Learning models for intelligent photo editing

Garoe Dorta

A thesis submitted for the degree of Doctor of Engineering

University of Bath

Department of Computer Science

August 2019

Signature of Author ....

Garoe Dorta

Abstract

This thesis addresses the task of photo-realistic semantic image editing, where the goal is to provide intuitive controls to modify the content of an image, such that the result is indistinguishable from a real image. In particular the focus is on editing applied to human faces, although, the proposed models can be readily applied to other type of images. We build on recently proposed deep generative models, which allow learning the image editing operations from data. However, there are a number of limitations in these models, two of which are explored in this thesis: the difficulty of modelling high-frequency image details, and the inability to edit images at arbitrarily high resolutions.

The difficulty of modelling high-frequency image details is typical of methods with explicit likelihoods. This work presents a novel approach to overcome this problem. This is achieved by surpassing the common assumption that the pixels in the image noise distribution are independent. In most scenarios, breaking away from this independence assumption leads to a significant increase in computational costs. Additionally, it introduces issues in the estimability of the distribution due to the considerable increment in the number of parameters to be estimated. To overcome these obstacles, we present a tractable approach for a correlated multivariate Gaussian data likelihood, based on sparse inverse covariance matrices. This approach is demonstrated on variational autoencoder (VAE) networks.

An approach to perform image edits using generative adversarial networks (GAN) at arbitrarily high-resolutions is also proposed. The method relies on restricting the types of edits to smooth warps, *i.e.* geometric deformations of the input image. These warps can be efficiently learned and predicted at a lower resolution, and easily upsampled to be applied at arbitrary resolutions with minimal loss of fidelity. Moreover, paired data is not needed for training the method, *i.e.* example images of the same subject with different semantic attributes. The model offers several advantages with respect to previous approaches that directly predict the pixel values: the edits are more interpretable, the image content is better preserved, and partial edits can be easily applied.

Acknowledgements

I would like to thank my many supervisors for their unwavering support, advice and patience. Neill Campbell and Ivor Simpson for never failing to produce a plethora of amazing research ideas. They were patient enough to explain to me all the intricacies of the Bayesian approach, and for emphasising that I should never lose sight of it in this deep learning era. Sara Vicente for being the voice of reason, especially for her help in focussing on the ideas that were more promising, and providing priceless insight into the theoretical and practical problems that came along. Lourdes Agapito for her valuable help, though unfortunately, not for the whole duration on the EngD.

I would like to thank my unofficial supervisors at Anthropics Technology Ltd., Simon Prince for his inestimable wisdom regardless of the topic, and Tony Polichroniadis for his involvement in many discussions.

I would like to thank my former supervisor, Yong-Liang Yang for being able to transform my problematic first year of the EngD into a useful learning experience.

Many friends and colleagues made these years far more enjoyable that what it would have been without them. Including all the people that I crossed paths with at the University of Bath, University College London and Anthropics Technology Ltd. In particular, I would like to thank Ieva Kazlauskaite, a treasured friend, for being there to discuss research, helping with proof-reading and taking part in fun activities outside of research.

No less important is the financial support provided by Anthropics Technology Ltd., and similarly by the Engineering and Physical Sciences Research Council, with grant EP/L016540/1.

Finally, I wish to thank my parents and sister for their steadfast encouragement, care, understanding and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Image editing for photo enhancement is ubiquitous in modern society, where millions of images are uploaded per day to social media services such as Snapchat$^©$, Instagram$^©$ or Facebook$^©$ [Smith, 2013]. The majority of image editing methods generally seek to modify the content of an input image such that the manipulated image is indistinguishable from a natural unedited image. Modifying an image to the point where it becomes obvious that is has been edited commonly leads to harsh public scrutiny, as demonstrated by any internet search with terms such as "Photoshop fails".

There are a number of reasons why it might be interesting to perform edits on an image. For example, inpainting techniques seek to fill in a missing region, which has applications in removing unwanted objects [Criminisi et al., 2004] or restoring damaged photographs [Bertalmio et al., 2000], as shown in Fig. 1-1, left. Image defocusing [Barron et al., 2015] allows obtaining shallow-depth-of-field images, that traditionally require using expensive DSLR cameras, from all-in-focus images obtained by affordable cellphone cameras, as shown in Fig. 1-1, right. Image denoising [Vincent et al., 2008] seeks to remove noise from an image, where the noise is usually introduced by the sensors that captured the image. Semantic editing [Blanz and Vetter, 1999] modifies an image according to a high-level semantic concept, such as changing the expression of a face in an image. This type of editing will be discussed in more detail below.

Image editing applied to human faces has a long history in computer vision [Beier and Neely, 1992, Blanz and Vetter, 1999, Liu et al., 2001, Mohammed et al., 2009] and has been made increasingly relevant with the rise in the number of pictures people take of others or themselves. For instance, 350 million photos per day were uploaded to Facebook$^©$ as of 2013 [Smith, 2013]. This thesis treats editing human faces as an

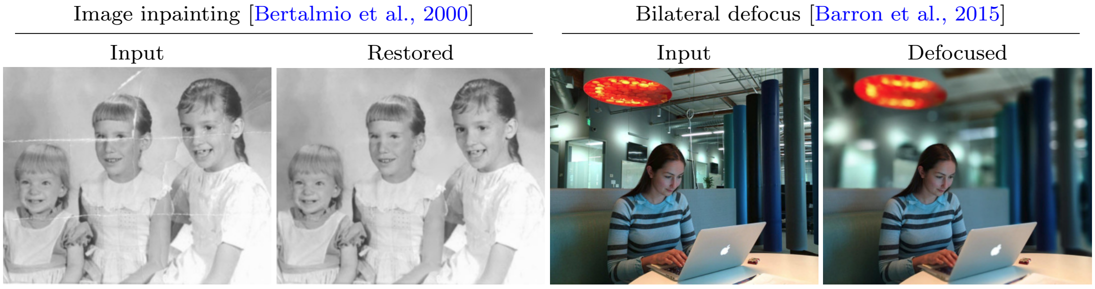


Image inpainting [Bertalmio et al., 2000]

Bilateral defocus [Barron et al., 2015]

Input

Restored

Input

Defocused

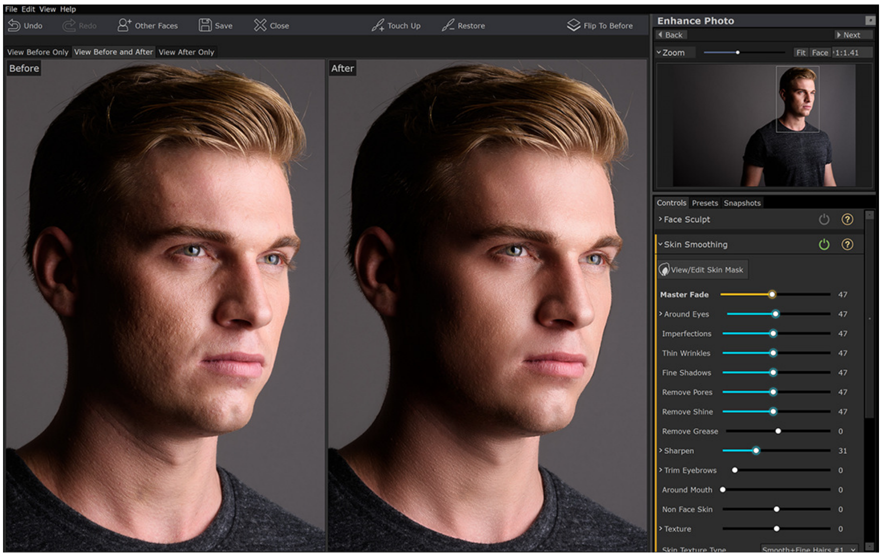
Figure 1-1: Example of image editing methods. An image inpainting method [Bertalmio et al., 2000] can be used to restore old photographs that have been damaged. A shallow-depth-of-field effect can be added to an all-in-focus image [Barron et al., 2015].[1]

example application, however the presented models can be readily applied to other data types and examples for non-face images can be found in Chapters 3 and 4.

The most naïve approach to editing would involve letting the user directly modify the pixel values in the image. However, this is extremely time consuming and it typically generates unrealistic images, as it is very easy to diverge from the manifold of real images using this approach [Zhu et al., 2016]. Higher level tools are also available, which allow editing groups of pixels. For example the liquify filter is used to deform a user-defined region in the image, and it can be found in most editing software. Still, employing these type of tools is time consuming and it requires a high level of expertise to produce good quality edits. Face editing is particularly demanding since humans can easily detect minor defects in other human faces [Mori et al., 2012], which adds extra challenges to research in this area. Any minor error produced during the editing operation could lead to an image that would not be considered realistic.

This research has been partly funded by Anthropics Technology Ltd.©, which produces intelligent image editing software, with a focus on human faces. The types of edits available in this software are of a semantic nature, as demonstrated by the example shown in Fig. 1-2. In contrast to low-level operations, these are high-level operations with an adaptive spatial scale, which depends on the particular edit. This leads to faster editing and makes the system more user-friendly, especially for novice users. In the example shown in Fig. 1-2, the user edits the input image by using a single slider that controls the smoothness of the skin in the face. Previously, users would require more technical skills and the use of general purpose tools on other professional editing software like Photoshop© to be able to achieve good quality editing. In contrast, Anthropics' software focuses on semantic editing with sliders to control the magnitude

[1] Images courtesy of [Bertalmio et al., 2000] and [Barron et al., 2015].



Master Fade 47
Around Eyes 47
Imperfections 47
Thin Wrinkles 47
Fine Shadows 47
Remove Pores 47
Remove Shine 47
Remove Grease 0
Sharpen 31
Trim Eyebrows 0
Around Mouth 0
Non Face Skin 0
Texture 0
Skin Texture Type Smooth+Fine Hairs #1

Figure 1-2: Semantic image editing with PortraitPro©, where the user adjusts a single slider corresponding to the "skin smoothing" attribute to manipulate the image.[2]

of the effect. In addition to the semantic edits, the software is able to generate photo-realistic results at interactive speeds without restrictions on the resolution of the input image. Furthermore, the identity of the subject in the picture is preserved after the edit, *i.e.* modifying the image to an extent where the subject can no longer be recognised is undesirable. Thus, these properties will be the guiding factors for the techniques explored in this thesis.

Anthropics Technology Ltd.© also develops editing tools for images containing human bodies, PortraitPro Body©, and for outdoor landscape images, LandscapePro©. The approach of editing via intuitive, high level semantic controls is shared by all of these. For example, PortraitPro Body© includes tools to easily modify the shape of different parts of the body, while LandscapePro© contains methods for manipulating the appearance of the sky, as well as other semantic parts of the image. The key idea for each piece of software is to employ domain specific knowledge in order to be able to implement the tools. For example, in PortraitPro© the skin must be detected automatically, as it is needed for a fully automated skin smoothing tool.

An important limitation in most of these tools is that even though the computer vision part of the process is automated, *e.g.* the skin detection for PortraitPro© or sky segmentation in LandscapePro©, most of the computer graphics parts are not. In other words, most of the transformations that are applied to the image are hand-designed.

[2] Image courtesy of Anthropics Technology Ltd.©

This requires a significant amount of work for the implementation of new edits, as these transformations are not trivial. A particular aspect that makes the design of these transformations cumbersome is that they must guarantee that the resulting image remains realistic and achieves the target edit. Moreover, careful consideration of the edge cases is required, such as extreme face poses, as these hand-designed edits might not work as well in those settings.

In this thesis we explore ways to automate the creation of new semantic edits. The approach that we follow is learning how to perform these transformations from data, as this allows us to rely less on designing complex algorithms and more on using labelled data. Unfortunately, there are number of issues that make this task non-trivial:

(i) There is the curse of **dimensionality**, images are high-dimensional mathematical objects, where each pixel is a three-dimensional vector containing the colour at that location in the image. Doubling the resolution of an image, leads to a quadratic increase in the number of pixels. Yet, the editing tools must operate at the original image resolution.

(ii) For most interesting image transformations **data labelling** is a process that is costly, noisy and ambiguous. For example, if we were to design a machine learning method to replicate the work of experts in Photoshop©, we would normally need a dataset of edited images, which is costly, as each expert would take a significant amount of time to edit each image. The data would be noisy, as some of them might make mistakes. Moreover, the results would be ambiguous, as even when aiming to achieve the same effect, each expert would edit the same image in a different manner.

(iii) The problem is **non-linear** in pixel space, as these transformations are of semantic nature. Therefore, learning them is by definition a non-trivial task.

(iv) In order to learn these image transformations, a common approach is to employ a generative model that captures the distribution of natural images. However, this requires unreasonable amounts of data, model complexity and computational resources. **Approximations** in the model may induce a lack of representation power, for example leading to the generation of blurry results.

In recent years, a number of academic methods have shown promise in learning these type of transformations from image data [Brock et al., 2017, Zhu et al., 2016, Yan et al., 2016], as shown in Fig. 1-3. These methods usually learn a low-dimensional representation of the data, where semantic editing corresponds to manipulating this

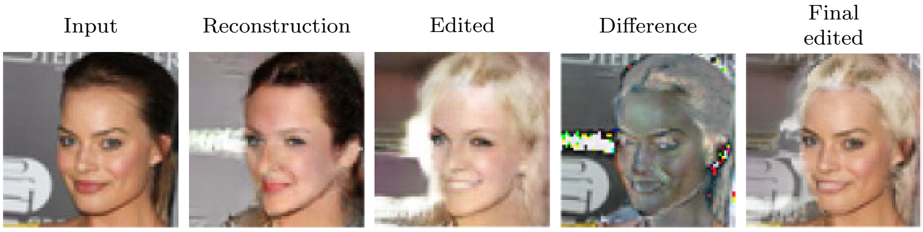


Figure 1-3: Example edits with neural photo editing [Brock et al., 2017]. The model operates on low dimensional representations of images. The reconstruction is produced by evaluating the low dimensional representation of the input image, and generating an image from it. A modification of this representation leads to an edited image. The difference between the input and the reconstruction is used to add some of the image content that was lost in the reconstruction step, producing the final result.[3]

representation. Henceforth, we will use latent space to refer to the space where the low-dimensional representations of the images live. Generally, semantic editing requires complex non-linear image manipulations, while manipulating the low-dimensional representation is frequently a simpler operation. Moreover, if there is a one-to-one mapping between a semantic concept a specific dimension in the representation, each dimension can correspond to a semantic slider in PortraitPro©.

The aforementioned methods are based on deep learning, a machine learning approach that employs neural networks, which will be discussed in more detail in Chapter 2. The deep learning approach has not only become the standard in face editing, but also in many other areas such as inpainting [Yang et al., 2017], super-resolution [Ledig et al., 2017], semantic segmentation [Luc et al., 2016] and depth estimation [Godard et al., 2017].

In particular, the image editing techniques mentioned above use deep generative models. For natural image data, these models seek to learn the distribution of natural images given a set of example images, and in principle these example images are not labelled. We only consider the subset of generative models that assume that the data is explained by a latent space. This is a typical approach in machine learning methods, that will be discussed in more detail in Section 2.1.

A key factor is that these models learn (or impose) some distribution on this low-dimensional representation. This has two important consequences: firstly, the models can be used to generate novel images by drawing samples from this distribution. Secondly, having some structure on the low-dimensional representation usually induces

[3] Images courtesy of [Brock et al., 2017].

some form of smoothness on the space, which is required to have smooth variations in the image space when traversing the low-dimensional space. Therefore, image editing can be performed in these models by finding out the location of the image in the latent space, moving to a new location according to the desired edit, and projecting back to the image space.

Two deep generative models are particularly popular: Generative Adversarial networks (GAN) [Goodfellow et al., 2014] and Variational Autoencoders (VAE) [Rezende et al., 2014, Kingma and Welling, 2014]. In relation to the editing techniques described above, neural photo editing [Brock et al., 2017] and the work by Zhu *et al.* [Zhu et al., 2016] employ GAN networks, while Attr2Img [Yan et al., 2016] uses a VAE.

Both, VAE and GAN will be explained in detail in Chapter 2. For now, we note that GAN models are capable of generating realistic images, yet they lack an inference method, *i.e.* they lack an encoder network to project an input image to the latent space. On the contrary, VAE models are known for generating blurry images. However, they have efficient inference and sampling mechanisms and hence may be more suitable for image editing. Despite this, we will come back to discussing GAN models later in the chapter.

1.1 Variational Autoencoders (VAE)

VAEs [Rezende et al., 2014, Kingma and Welling, 2014] consist of an encoder network that transforms from the image space to the latent space, *i.e.* it finds a low-dimensional representation of an input image, and a decoder network, which does the reverse process, *i.e.* it generates an image given its low-dimensional representation. This model has been used for a variety of applications, including predicting future frames from video data [Walker et al., 2016], object pose estimation [Prokudin et al., 2018] and semantic attribute editing on face images [Yan et al., 2016].

During training the model performs an autoencoding task, *i.e.* it finds the low dimensional representation of an image using the encoder network, and it reconstructs the image given this information using the decoder network. A probabilistic reconstruction loss is used to evaluate how well the model is reconstructing the input images. The residual image, $\boldsymbol{\epsilon} = \mathbf{x} - \boldsymbol{\mu}$, gives the empirical error per pixel, and it is defined as the image difference between the input, $\mathbf{x}$, and the mean output, $\boldsymbol{\mu}$, of the decoder network. The distribution of these residual images is modelled by the VAE as variance in the predicted values, where this variance indicates how confident the model is about the

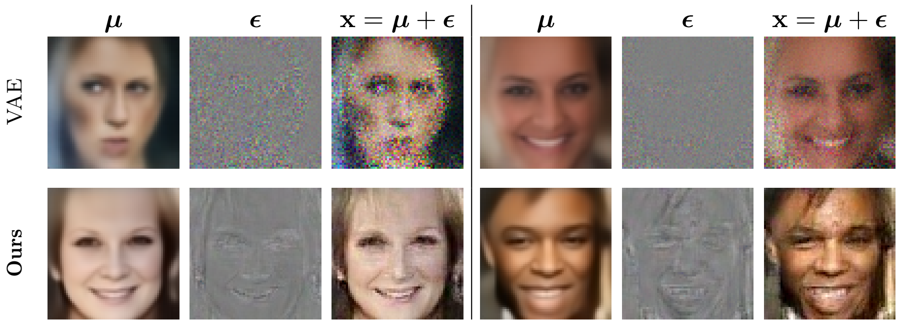

$\boldsymbol{\mu}$
$\boldsymbol{\epsilon}$
$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\epsilon}$
$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$
VAE
Ours

Figure 1-4: Samples from a VAE [Rezende et al., 2014, Kingma and Welling, 2014] and from the model presented in Chapter 3. VAEs use a forward model for the data where $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$, which correspond to a mean, $\boldsymbol{\mu}$, and a residual, $\boldsymbol{\epsilon}$. It can be seen how a VAE generates noisy images, $\mathbf{x}$, with blurry means, $\boldsymbol{\mu}$. Our model generates images, $\mathbf{x}$, with structured residuals, $\boldsymbol{\epsilon}$, that contain plausible high-frequency content.

predicted pixel values. In other words, the images are modelled as a mean component and a residual component, $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$, as shown in Fig. 1-4. The estimated residual component should be similar to the real residual. The images shown in the figure are samples from the model, not reconstructions, thus, the real residual is unknown.

It is common practice to model the residual distribution of each pixel value as being i.i.d. (independently and identically distributed), which is a simplifying approximation of the type discussed in item (iv). In practice, the model is often uncertain about its predictions for most data types, including images. High uncertainty results in high levels of noise when sampling from the i.i.d. distribution, which in turn produces unrealistic images, as shown in Fig. 1-4. For this reason, researchers tend to only show the mean component [Larsen et al., 2016, Yan et al., 2016], rather than a true sample from the model, which would include $\boldsymbol{\epsilon}$. Unfortunately, this severely limits the applicability of the model for editing, as the output images are either blurry or noisy.

Hypothesis

We postulate that the real residuals in VAE models are highly structured, *i.e.* that they are not independently and identically distributed, and reflect limitations in model capacity. Additionally, we posit that these structured residual distributions can be tractably learned by the model.

Proposed solution

We therefore propose to model the residual distribution using a Gaussian model with dense covariance matrices to capture the pixel-wise correlations. This follows a similar line of work as previous methods that employ more complex shapes for the data distribution in VAEs [Larsen et al., 2016, Gulrajani et al., 2017b, Hou et al., 2017]. Our method allows the VAE model to generate images that contain realistic high-frequency details, as shown in Fig. 1-4. In practice, this approach helps to reduce the errors due to item (iv), as the approximation assumptions are relaxed. This is not an easy task, as it implies that the model must estimate a full covariance matrix from a single sample. Our tractable method to predict structured uncertainty is discussed in Chapter 3.

1.2 Generative Adversarial Networks (GAN)

During the development of the aforementioned method, relevant extensions to GAN [Goodfellow et al., 2014] models were being concurrently proposed. These models have become increasingly relevant in computer vision tasks, including super-resolution [Ledig et al., 2017], image inpainting [Yang et al., 2017], face editing [Portenier et al., 2018, Shu et al., 2017, Geng et al., 2018] and semantic segmentation [Luc et al., 2016].

GANs also contain a decoder network, which is known as a generator network, that transforms a low-dimensional vector into an image. However, instead of training by performing reconstructions, GAN models employ an alternative approach. An auxiliary discriminator network is concurrently trained to evaluate if an image comes from the real data distribution, or if it was produced by the generator. Meanwhile, the generator tries to fool the discriminator into classifying its images as belonging to the real data distribution.

Contrary to VAEs, GAN models are able to produce images with realistic high-frequency details. However, GANs doe not provide any inference method to recover the corresponding low-dimensional representation of an input image. A number of methods [Odena et al., 2017, Dumoulin et al., 2017, Donahue et al., 2017, Huang et al., 2018] have been explored to address this issue, which will be discussed in Section 2.5.3. However, a reconstruction from one of these GAN-based methods, despite containing high frequency details and being realistic, will rarely correspond to the input image, limiting their direct use as an image editing method.

Regardless of this limitation, state-of-the-art methods now commonly use adversarial

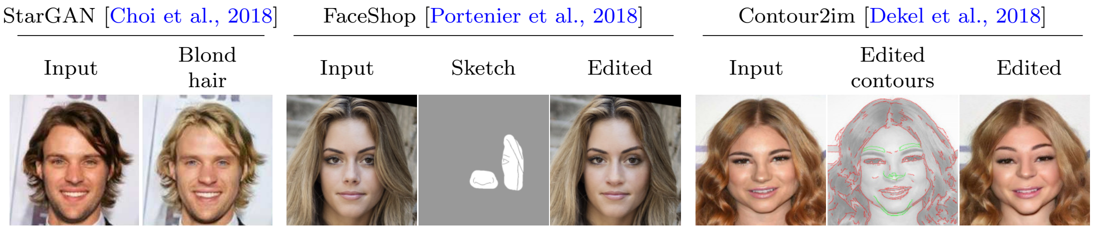

StarGAN [Choi et al., 2018]
Input
Blond hair
FaceShop [Portenier et al., 2018]
Input
Sketch
Edited
Contour2im [Dekel et al., 2018]
Input
Edited contours
Edited

Figure 1-5: Example edits with several state of the art methods that use adversarial losses. StarGAN [Choi et al., 2018] takes an input image and generates an edited result based on a binary attribute; in this example the attribute indicates the presence of "blond hair". FaceShop [Portenier et al., 2018] edits the image by inpainting a region and using a sketch as guidance; in this example changing the shape of the nose and unoccluding the eye. Contour2im [Dekel et al., 2018] edits the image by manipulating a set of contours; in this case the green contours have been moved and scaled.[4]

losses to ensure that the generated images are realistic [Choi et al., 2018, Portenier et al., 2018, Dekel et al., 2018], as shown in Fig. 1-5. For example, StarGAN [Choi et al., 2018] edits an image by changing binary semantic attribute labels, such as blond hair to brunette hair. FaceShop [Portenier et al., 2018] allows editing by drawing rough sketches, and Contour2im [Dekel et al., 2018] by scaling, translating or copying a set of image contours. Hence, these models offer a wide variety of high level controls to perform the editing operations. More importantly, they offer means to learn the image transformations from data, which fits with our objective of developing automated transformations for the semantic sliders. Of relevance to our objective of learning image transformations from data are image-to-image translation methods [Isola et al., 2017], and in particular extensions of the base model such as Cycle-GAN [Zhu et al., 2017] and the aforementioned StarGAN model.

Cycle-GAN and StarGAN

Cycle-GAN [Zhu et al., 2017] demonstrated that image editing operations can be learned from unpaired data. In the example shown in Fig. 1-6, Cycle-GAN learns a horses to zebras transformation without paired data, *i.e.* without a real example of how the horse in "Input A" would look if it were a zebra, and vice versa with zebras to horses. A Cycle-GAN method cannot be used to sample novel images, nevertheless, it overcomes the aforementioned encoding issues typical of GAN-based methods. Therefore, this is no longer a generative model, but a discriminative one, where the input data are the original images, and the labels are the ground truth edited images.

[4] Images courtesy of [Choi et al., 2018], Portenier [Portenier et al., 2018] and [Dekel et al., 2018].

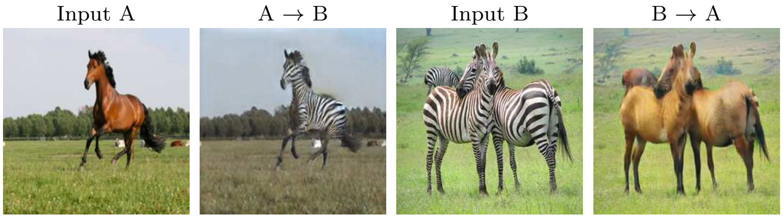


Figure 1-6: Example edits with Cycle-GAN [Zhu et al., 2017]. The model learns how to translate between domains A and B. In this case, domain A corresponds to images of horses, and domain B to images of zebras. The model is trained without paired data, *i.e.* a ground truth images for the A → B and the B → A transformations are not required.[5]

However, as previously stated, ground truth edited images are not actually needed for training this model. Moreover, being able to learn transformations without paired data is useful, as collecting paired data is generally a costly process, as discussed in item (ii).

This approach fits better with our objective of learning automated transformations for semantic sliders. However, Cycle-GAN models, and extensions such as StarGAN [Choi et al., 2018], have a number of limitations. Firstly, these methods can only operate at the same (or similar) resolution as the training data. Due to the data dimensionality limitations discussed in item (i), the resolution of the training images is usually low, which is problematic. An example is shown in Fig. 1-7, where the model learned how to transform a face with a neutral expression to a face with a sad expression, as well as the opposite transformation, which is not shown. The model produces good quality edits for images at the same resolution as the training data, yet, it fails when the resolution is increased. An obvious alternative is to downsample the input image to the training data resolution. However, this would entail a significant loss of high frequency detail when upsampling back to the original resolution, as observed by comparing the StarGAN result at low resolution to the high resolution result of the method that will be outlined below. This is a limitation that is also shared by most deep learning methods. A second limitation is that due to the lack of paired data, these models tend to inadvertently edit parts of the image that should have remained fixed. For example, in Fig. 1-5, StarGAN not only makes the hair blonde but the skin lighter. Intuitively, this can be understood by noticing that blond haired people frequently have lighter skin tones, and the model is unable to decorrelate both characteristics during training.

[5] Images courtesy of [Zhu et al., 2017].

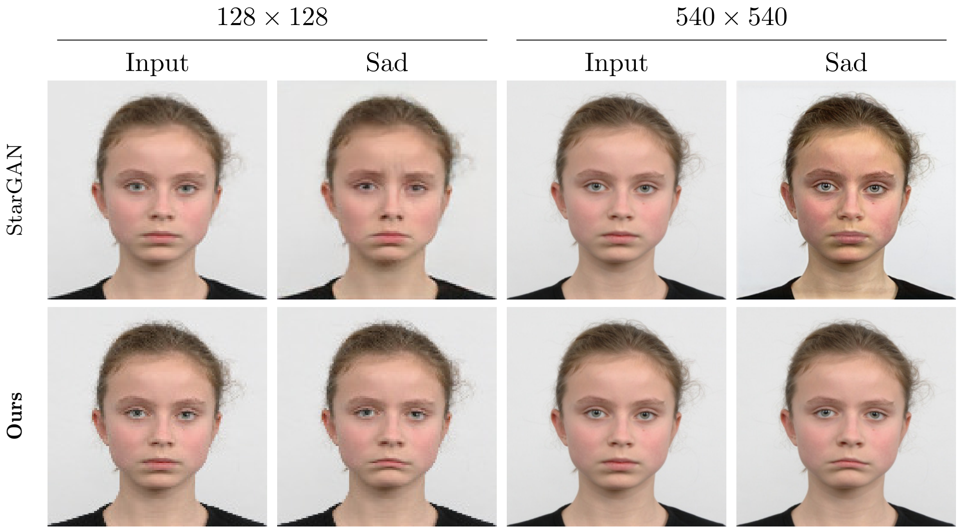


Figure 1-7: Image editing with a StarGAN [Choi et al., 2018] model and with the model presented in Chapter 4. Both models attempt to transform the neutral expression into a sad expression, and they are trained with images with a resolution of 128 × 128. StarGAN fails when the resolution of the input image differs significantly (540 × 540) from the training data. Our model successfully transfers the edit at all resolutions. (Zoom in for details)

Hypothesis

We postulate that a restricted set of semantic image transformations can be tractably learned from unpaired data in an image-to-image translation model, such as StarGAN, and used for editing at arbitrary resolutions.

Proposed solution

Inspired by previous methods learning from paired data [Yeh et al., 2016, Geng et al., 2018, Ganin et al., 2016], we propose to learn warp fields (geometric deformations) for semantic image editing in a StarGAN model, as show in Fig. 1-7. By being able to operate at arbitrary resolutions this model addresses item (i) in the restricted case of edits that only require geometric deformations. This approach also helps in reducing the amount of content that is edited in the image that should have remained fixed, as it is restricted to only model geometric deformations of the input image. Moreover, as the method is an extension of Cycle-GAN methods, it retains the ability to be trained with unpaired data. Our warping-based image-to-image translation method is presented in Chapter 4.

1.3 Evaluation

An important question is how to evaluate the performance of the different models presented in this thesis. Traditionally, in image editing applications a simple metric to measure the performance is whether the model produced the target edited image. For most edits this is very hard to evaluate quantitatively without resorting to user studies as there are ambiguities for many types of edits, such as smiling, even with paired data. Instead, the performance of the models on image editing will be measured under these three aims

1. Did the method achieve the target edit?
2. Is the edited image realistic?
3. Is the method computationally efficient?
4. How complex is the method?

Ideally a representative set of users would be used to evaluate the first two by employing user studies, where the edited images from the different methods are compared. Unfortunately, this is too expensive to be continuously used while developing the models. Therefore, automatic approximations to these metrics must be used in all other cases. For the first, if labelled data is available, a classifier can be trained and used to estimate the presence of a target edit, where the classifier will be a deep neural network. Evaluating image realism is a more complex task, as realism is a subjective metric, as it relies in its core on human perception. However, for human faces, part of what makes an edit realistic is whether the subject identity is preserved after the edit. Such identity scores are commonly used in face re-identification tasks [Schroff et al., 2015], where a trained model outputs a score for any given pair of original and edited images. Finally, we would prefer methods that are simple and efficient, rather than employing models that are inefficient or complicated.

In summary, the following metrics will be reported for image editing methods

1. Quantitative metrics:
    (a) the presence of the target attribute is evaluated both automatically with a classification network, and ground truth values are reported by carrying out user studies,

    (b) user studies are carried out to asses the perceptual realism of the generated images,

i. an identity score is reported to asses if the subject identity is preserved on the manipulated image, which is measured by a face re-identification network,

(c) the number of floating point operations required to edit an image is reported as a measure of model efficiency,

(d) the number of learnable parameters is reported as an indicator of model complexity,

2. Qualitative metrics:

   (a) example edited produced by the models are shown for qualitative evaluations.

We are also interested in evaluating purely unsupervised deep generative models, such as VAEs, as they may be used for image editing. In a Bayesian framework, model evaluation follows Occam's principle, the simplest model that explains the data is preferable [MacKay, 1992]. This implies modelling the distribution of the model parameters, as the model evidence is measured by marginalising over the distributions of model parameters and the data distribution. Unfortunately, for most deep learning methods evaluating the distribution over the model parameters given a dataset is not tractable. Approximations which reduce the computational burden exist, such as the Bayesian information criteria [Schwarz, 1978] and minimum description length approaches [Rissanen, 1978]. Unfortunately, for implicit likelihood methods such as GAN [Goodfellow et al., 2014], even computing the likelihood on test data is intractable. Thus, a combination of alternative tractable metrics is used to evaluate the generative models developed in this thesis:

1. Quantitative metrics:

   (a) when possible the likelihood on a test set is reported, where a higher likelihood is usually indicative of the model better capturing the unknown real data distribution,

   (b) the number of learnable parameters is reported as an indicator of model complexity,

2. Qualitative metrics:

(a) samples are drawn for generative models, and these are shown for qualitative evaluations,

(b) latent space interpolations are shown for latent generative models, in order to evaluate the quality of the latent space.

Despite the fact that these metrics are standard evaluation tools in the computer vision research community [Rezende et al., 2014, Kingma and Welling, 2014, Burda et al., 2016, Kingma et al., 2016, Goodfellow et al., 2014, Radford et al., 2016, Oord et al., 2016], it is known that they have several flaws. For example, test likelihoods are a popular metric for the evaluation of deep generative models. However, it has been shown [Theis et al., 2016] that test likelihoods and the quality of the samples from the model are largely independent of each other for high dimensional data such as images. Moreover, the test likelihood for most deep learning models does not take into account the complexity of the model. Measuring the quality of a latent space is also an open problem. For most deep generative models the quality of the images generated when traversing the latent space is defined subjectively as a perceptual metric, as ground truth data only exists for trivial cases and synthetic scenarios [Lucic et al., 2018, Higgins et al., 2017]. In conclusion, the evaluation of deep generative models is still an open area of research.

1.4 Publications

The work in this thesis has been presented in the following conference papers:

Dorta, G., Vicente, S., Agapito, L., Campbell, N. D. F., and Simpson, I. (2018b). Structured uncertainty prediction networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Dorta, G., Vicente, S., Campbell, N. D. F., and Simpson, J. A. I. (2020). The GAN that warped: Semantic attribute editing with unpaired data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

patent (under review):

Dorta, G., Campbell, N. D. F., and Simpson, I. (2018a). Method of modifying digital images. UK Patent, Application Number 1818759.1

and technical report:

Dorta, G., Vicente, S., Agapito, L., Campbell, N. D. F., and Simpson, I. (2018c). Training VAEs under structured residuals. *arXiv preprint:1804.01050*

1.5 Thesis outline

Chapter 2 describes deep generative models and relevant previous work on image editing for faces. Both VAE [Rezende et al., 2014, Kingma and Welling, 2014] and GAN [Goodfellow et al., 2014] models are described in detail, including limitations and several extensions to both models.

Chapter 3 explores a tractable method to predict structured uncertainty. Previous work on uncertainty prediction is discussed. Applications of the predicted uncertainty distribution are shown, including editing by navigating the latent space and denoising.

Chapter 4 describes an image-to-image translation editing technique based on geometric deformations (warping) of the input image. Existing methods for using deep learning techniques at high image resolutions are discussed. Semantic image editing results at arbitrary resolutions are shown, including partial edits.

Chapter 5 summarises the ideas presented in the thesis, draws conclusions and presents potential future work.

Chapter 2

Background

In the previous chapter the advantages of learning image transformations from data were stated in the context of developing semantic tools for image editing. Two latent variable deep generative models were introduced as particularly relevant for the task: Variational Autoencoders (VAE) [Rezende et al., 2014, Kingma and Welling, 2014] and Generative Adversarial Networks (GAN) [Goodfellow et al., 2014]. Additionally, Cycle-GAN [Zhu et al., 2016], a regression method that builds upon the GAN model was also discussed.

In this chapter we discuss machine learning and the deep learning approach to machine learning as prerequisites to understand the VAE, GAN and Cycle-GAN models. In order to put in context VAEs and GANs, the whole family of deep generative models is discussed. This highlights the advantages and disadvantages of both models, and why those are particularly interesting for our purposes. We proceed by discussing the VAE model in detail, including several limitations, and a number of extensions that attempt to address them. This is followed by a similar discussion of the GAN model, and of image-to-image translation methods, which reviews the Cycle-GAN model and several extensions. Finally, a brief history of image editing models is provided, with a particular focus on methods that address face editing. Most of the modern image editing approaches employ the aforementioned deep learning techniques, where GAN-based methods are particularly popular.

In summary, an introduction to machine learning is provided in Section 2.1, followed by a general introduction to deep learning in Section 2.2. Generative models, within the context of deep learning, are discussed in Section 2.3. Two relevant deep generative methods, VAE and GAN, are described in sections 2.4 and 2.5, respectively. Image-to-image translation methods are examined in Section 2.6. Finally, a description of face editing methods is provided in Section 2.7.

2.1 Machine learning

Most state of the art methods for image editing, including the ones discussed in the previous chapter, are based on *machine learning* techniques. Machine learning methods define a *model* to explain the *data* of interest. Additionally, the model is used to make predictions about new data, where this process is commonly known as *inference*. A set of *parameters* are used to define the model, where generally speaking models with more parameters will be able to explain more complex processes, while models with fewer will be more limited. However, models with too many parameters are computationally more expensive to evaluate and they might overfit to the training data, *i.e.* they might be highly predictive on the data they were trained with, yet fail to generalise on novel data. The process of finding the best set of parameters for some given data is denoted as *learning*.

An alternative to machine learning approaches are physics-based methods. These techniques are useful if the equations governing the system are known and can be simulated efficiently. One of the reasons why machine learning approaches for image understanding are prevalent, is that the underlying physics governing the image formation process are complex. A main cause of complexity is due to intrinsic ambiguities in this process. Examples of this include occlusions, which might hide important information, and colour ambiguities, *e.g.* the apparent color of an object might be due to white light shone on a coloured object, or due to coloured light shone on a white object. Machine learning approaches allow learning useful models that are assumed to implicitly approximate this process from data, without the need to explicitly know and encode in the model the physical equations.

Machine learning methods have been traditionally divided in two distinct groups: *supervised* methods and *unsupervised* methods. The former assumes that for each image we have a corresponding *label* that represents the information that we are interested in learning. For example, we might be given a dataset of images of faces and labels indicating the head orientation. *Unsupervised* methods on the other hand are concerned with learning the data generator process and/or uncover trends and other interesting underlying characteristics given only a collection of images.

Formally these ideas can be expressed in terms of probabilities. In the supervised case, we have a set of observations, $\{\mathbf{x}_j\}_{j=1}^J$, with their corresponding labels, $\{\mathbf{c}_j\}_{j=1}^J$, where each item is assumed to be an independent sample from a distribution $p(\mathbf{c}|\boldsymbol{\theta}, \mathbf{x})$, where $\boldsymbol{\theta}$ denotes the learnable model parameters. For the unsupervised case, the label information is not available, and the goal is to model $p(\mathbf{x}|\boldsymbol{\theta})$. Learning usually consists of finding the model parameters that maximise the probability of the observed data.

Additionally, models can be also be described as either *discriminative* or *generative*. The former are concerned with directly modelling the distribution of the labels given the images, $p(\mathbf{c}|\boldsymbol{\theta}, \mathbf{x})$. The latter aim to learn the distribution of images given the labels, $p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{c})$, and it employs Bayes' rule to infer the labels distribution given the observations,

$$p(\mathbf{c}|\boldsymbol{\theta}, \mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{c})p(\mathbf{c})}{\int p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{c})p(\mathbf{c})d\mathbf{c}}, \tag{2.1}$$

where $p(\mathbf{c})$ is a prior distribution over the labels. In the supervised case, the task will be to learn the model parameters, $\boldsymbol{\theta}$, given the data $\mathbf{x}$ and $\mathbf{c}$. Unsupervised models are usually only applied to generative settings, where $\boldsymbol{\theta}$ must be found given only $\mathbf{x}$.

Models can be augmented by *latent variables*, $\mathbf{z}$, which are usually employed to account for underlying characteristics in the data that are unobserved. Formally, the data is conditioned on $\mathbf{z}$, and these latent variables must be marginalised out when evaluating the data distribution

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{z})p(\mathbf{z})d\mathbf{z}. \tag{2.2}$$

A common instance in unsupervised generative settings is to assume that the data can be indexed by a parameter from an unknown latent space, $\mathbf{z}$. Formally, this would correspond to

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = f(\mathbf{z}, \boldsymbol{\theta}) \tag{2.3}$$

where f is a function with learnable parameters $\boldsymbol{\theta}$. The latent parameters, $\mathbf{z}$, could correspond to the physical parameters that actually produced the data, and the model is expected to learn them in order to be able to model the data well. For example, we might be given a dataset of images of faces from which we wish to learn a latent variable that would correspond to head orientation.

Generative models are of interest for image editing. In particular, for models with latent variables, new data, $\mathbf{x}^*$, can be sampled given an edited set of latents, $\mathbf{z}^*$. The latent variables, $\mathbf{z}$, can be inferred, $p(\mathbf{z}|\boldsymbol{\theta}, \mathbf{x})$, from a given input image, $\mathbf{x}$, and then a particular parameter in the latents can be altered to construct $\mathbf{z}^*$. With the edited latents, a new image can be sampled from the learned distribution, $p(\mathbf{x}^*|\boldsymbol{\theta}, \mathbf{z}^*)$, such that it contains the desired changes. For example, if $\mathbf{x}$ models face images, some parameters in $\mathbf{z}$ might control the head pose and orientation. We might change those values, and sample a new image with the desired head pose and orientation. In practice, the learned latents can rarely be used directly for editing. A common alternative is to use target labels, $\mathbf{c}^*$, containing the parameters that we are interested in editing. The values in the target labels would be different from the original image labels, $\mathbf{c}$, and edited images may be sampled from $p(\mathbf{x}^*|\boldsymbol{\theta}, \mathbf{z}, \mathbf{c}^*)$.

Generative models with latent variables also come with a set of disadvantages. Inference, *i.e.* evaluating the posterior distribution, $p(\mathbf{z}|\boldsymbol{\theta}, \mathbf{x})$, can be computationally challenging, as it requires employing Bayes' rule to evaluate it given $p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{z})$, formally,

$$p(\mathbf{z}|\boldsymbol{\theta}, \mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{p(\mathbf{x}|\boldsymbol{\theta})} = \frac{p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{z})p(\mathbf{z})}{\int p(\mathbf{x}|\boldsymbol{\theta}, \mathbf{z})p(\mathbf{z})d\mathbf{z}}, \tag{2.4}$$

where $p(\mathbf{z})$ is a prior distribution over the latent variables.

Image editing can also be approached with discriminative models as a regression problem. In this case, $\{\mathbf{x}, \mathbf{c}\}$ reflects a pair of images, where in each pair we have the image before the editing operation, $\mathbf{x}$, and another after editing, $\mathbf{c}$. We would be interested in learning the distribution of edited images given the original images, $p(\mathbf{c}|\boldsymbol{\theta}, \mathbf{x})$. The main issue with this method commonly lies in collecting the pairs of images needed to train the model. For example, if a model is required to approximate a complex editing operation performed by experts on a particular software, the experts are required to edit a large number of images in order to build a training dataset. This process can easily become too costly to be practical.

2.2 Deep learning

In the last few years, machine learning approaches, which make use of deep neural networks (DNN), have become the standard in most computer vision tasks, including inpainting [Yang et al., 2017], super-resolution [Ledig et al., 2017] and face editing [Portenier et al., 2018]. Deep networks are a set of non-linear models that are flexible and powerful and, which have been shown to be universal function approximators [Hornik et al., 1989, Cybenko, 1989]. Networks are composed of interconnected layers, where each layer applies a non-linear transformation to its input. An in depth discussion of deep learning methods can be found in [Goodfellow et al., 2016]. As most of the models of interest for our purposes are generative with latent variables, we define deep networks as

$$\mathbf{x} = f_n(\cdots f_2(f_1(f_0(\mathbf{z})))\cdots), \tag{2.5}$$

$$f_i(\mathbf{y}_i) = g_i(\mathbf{A}_i\mathbf{y}_i + \mathbf{b}_i), \tag{2.6}$$

where $\mathbf{z}$ are the latent inputs, $\mathbf{x}$ are vectorised images, $\mathbf{y}_i$ is the input of the f_i layer, g_i is a predefined non-linear transformation, such as a sigmoid or a ReLU [Nair and Hinton, 2010], $\mathbf{A}_i$ and $\mathbf{b}_i$ are the layer learnable parameters, a weight matrix and bias vector, respectively. Thus, $\boldsymbol{\theta} = \{\mathbf{A}_0, \mathbf{A}_1, \cdots, \mathbf{A}_n, \mathbf{b}_0, \mathbf{b}_1, \cdots, \mathbf{b}_n\}$ is used to denote the entire set of learnable parameters in the DNN.

For image data, convolutional neural networks [LeCun et al., 1998] are the de facto standard. They employ 2D convolutions, which are equivalent to imposing a local neighbourhood sparsity pattern on $\mathbf{A}_i$, for vectorised input images, $\mathbf{y}_i$. The convolution operation is also location invariant, which significantly reduces the number of parameters that are learned in $\mathbf{A}_i$ and $\mathbf{b}_i$. Non-linear downsampling techniques, such as max-pooling [Ranzato et al., 2007], are also a popular tool to reduce the dimensionality of $\mathbf{A}_i$.

There are a number of approaches for learning the parameters, $\boldsymbol{\theta}$. The simplest one is *maximum likelihood*, which finds for each parameter a point estimate under which the data is most likely. For a generative model this is defined as

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left[\prod_{j \in \mathcal{D}} p(\mathbf{x}_j | \boldsymbol{\theta}) \right], \tag{2.7}$$

where $\mathcal{D} = \{\mathbf{x}_j\}_{j=1}^{J}$ is the dataset and $\hat{\boldsymbol{\theta}}$ are the most likely parameters.

The use of large datasets and complex distributions leads to intractable integrals when employing more complex methods [Neal, 2012] for learning the parameters, $\boldsymbol{\theta}$. Moreover, for complex models it can be significantly challenging to add an informative prior on the model parameters, as the parameters do not have any intuitive interpretation. Therefore, in most commonly used state of the art deep learning methods a maximum likelihood approach is used. For a random initialisation of the parameters, Stochastic Gradient Descent (SGD) methods are used to find a local maximum solution. The required gradients can be automatically obtained via automatic differentiation packages [Abadi et al., 2015]. Popular SGD methods for deep learning models include Adam [Kingma and Ba, 2015] and RMSprop [Tieleman and Hinton, 2012].

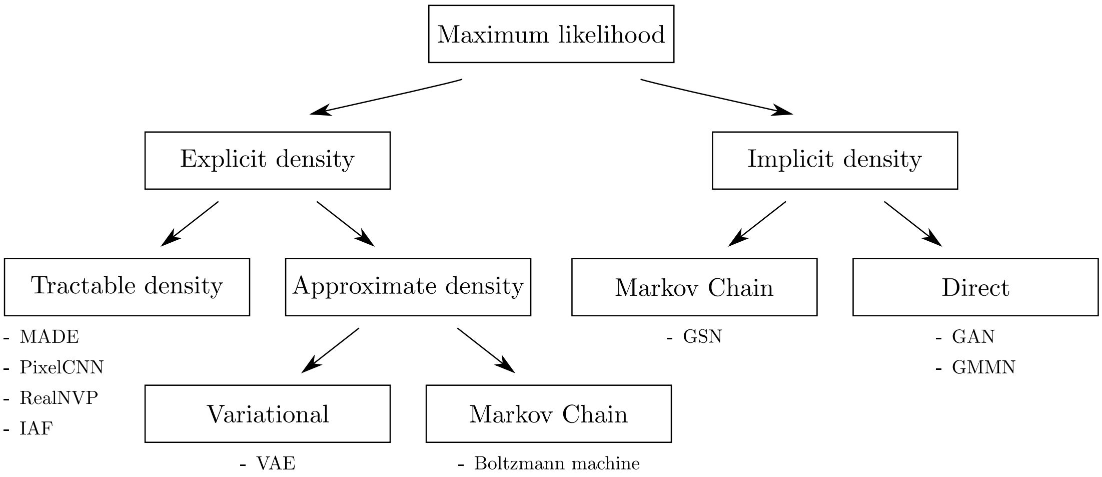


Figure 2-1: Overview of deep generative learning models, which employ maximum likelihood optimisation for estimating their parameters. Explicit methods directly optimise the likelihood by using likelihoods with tractable densities, or employing approximations. Implicit ones optimise the likelihood via sampling, where the samples are obtained with Markov Chain methods, or directly by the model.[1]

As discussed in the previous chapter, deep generative methods are of particular interest for learning image transformations from data. Therefore, we turn our attention in the next section to generative models, and in particular to those suitable for deep learning.

2.3 Deep generative models

In this section several deep generative models are discussed, including VAEs and GANs, as this serves to place these two models within the wider context of deep generative models. In Section 2.1 generative models were defined as characterising the data distribution $p(\mathbf{x}|\boldsymbol{\theta})$, potentially conditioned on a latent variable $p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})$. Deep generative models seek to learn this distribution, where the model parameters, $\boldsymbol{\theta}$, correspond to that of a deep neural network.

These models can be classified regarding the form of their likelihood distribution, $p(\mathbf{x}|\boldsymbol{\theta})$ or $p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})$, as shown in Fig. 2-1. These divisions arise due to the fact that for many methods, directly optimising the likelihood is not tractable. On the one hand, there are methods that define an *explicit* distribution for the likelihood, which can be in the form of an intractable integral. On the other, there are methods which have an *implicit* likelihood distribution, which only allows to take samples from the distribution. For all

[1] Diagram adapted from [Goodfellow, 2016].

the methods, a point estimate of the model parameters, $\boldsymbol{\theta}$, is evaluated, as discussed above. However, for the methods that rely on latent variables, $\mathbf{z}$, these parameters are treated differently from other model parameters, $\boldsymbol{\theta}$, as the latent variables are modelled with approximate/tractable distributions.

Explicit density

Examples of explicit methods include providing bounds to the likelihood via variational (VAE) [Kingma and Welling, 2014, Rezende et al., 2014] or Markov chain approximations [Smolensky, 1986]. Another line of work includes methods that have a tractable but complex density. Either by employing autoregressive distributions [Oord et al., 2016], stacking change of variable transformations with triangular Jacobians [Germain et al., 2015, Dinh et al., 2017] or both [Kingma et al., 2016].

Tractable density methods employing autoregressive distributions are computationally costly [Oord et al., 2016], as they require evaluating the full model independently for each pixel. Tractable methods that employ change of variable transformations [Germain et al., 2015, Dinh et al., 2017] present a better approach, as they do not have this constraint. However, they usually require composing a large number of transformations to be able to match the complex densities of real data distributions [Kingma and Dhariwal, 2018]. Moreover, as they are restricted by construction to be invertible functions, they learn latent spaces that have the same dimensionality as the input images. In the previous chapter, we established a preference for models in which manipulating a single dimension in the latent space corresponds to manipulating a semantic concept in the image. Hence, these models are not suited for image editing applications, as in general, a dimension in their latent space will not directly correspond to a semantic attribute.

The performance of approximate methods is heavily tied to the quality of the approximation that is employed. Markov chain methods [Smolensky, 1986] struggle to scale beyond modest resolutions [Goodfellow, 2016]. To aggravate matters further, there is no trivial way to evaluate when the chain has converged to a sample from the model. Thus, samples from this model are either computationally costly, or biased. Variational methods [Kingma and Welling, 2014] fare better, as they provide efficient mechanisms for learning and inference. However, they are known to struggle to generate high-frequency content for image data.

Implicit density

On the other hand, implicit density methods provide means to optimise the distribution via samples. Popular implicit likelihood methods include kernelised moment matching [Li et al., 2015, Dziugaite et al., 2015] and approximating the ratio of the data density versus the modelled density, with an auxiliary network (GAN) [Goodfellow et al., 2014]. Another approach relies on running a Markov chain to be able to sample from the model [Bengio et al., 2014].

A significant limitation of moment matching techniques is that they require choosing an appropriate kernel, large batch sizes and empirically have shown inferior performance over density ratio methods [Li et al., 2017a]. Markov chain implicit density methods share similar limitations as the aforementioned markov chain explicit density approaches. Ratio estimation approaches are the best performing implicit density methods, and have been shown to generate photo-realistic images. However, they are known to have unstable training behaviour, and to only model a (small) subset of the training data distribution [Radford et al., 2016, Arjovsky and Bottou, 2017].

Given the aforementioned considerations, in the following sections we will discuss in more detail Variational Autoencoders (VAE) [Kingma and Welling, 2014], as the representative for the explicit likelihood methods, and Generative Adversarial Networks (GAN) [Goodfellow et al., 2014], as the most relevant of the implicit likelihood methods. Both methods have in common the use of a latent space that has lower dimensionality than the input, thus the latent space is potentially more meaningful than in the alternative methods. For a more gentle introduction to both models, we refer the readers to Doersch [Doersch, 2016] and Goodfellow [Goodfellow, 2016], for VAE and GAN respectively.

2.4 Variational Autoencoders (VAE)

In this section we provide a more detailed description of the Variational Autoencoder (VAE) [Rezende et al., 2014, Kingma and Welling, 2014] model and several of its extensions.

A VAE is a latent variable model, which learns the probability distribution, $p(\mathbf{x} \,|\, \mathbf{z}, \boldsymbol{\theta})$, of the input data, $\mathbf{x}$, conditioned on a low-dimensional representation, $\mathbf{z}$, in a latent space. The likelihood of the input data, $p(\mathbf{x} \,|\, \mathbf{z}, \boldsymbol{\theta})$, is modelled by a decoder, which outputs the parameters of this distribution. The posterior probability distribution, $p(\mathbf{z} \,|\, \mathbf{x}, \boldsymbol{\theta})$, is intractable. Instead, an encoder is used for inference to provide an approximation to this distribution, by employing a tractable variational approach, $q(\mathbf{z}\,|\,\mathbf{x},\boldsymbol{\phi}) \approx p(\mathbf{z}\,|\,\mathbf{x},\boldsymbol{\theta})$. In practice, neural networks parametrised by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are used to model the distributions. To simplify notation, subscripts will be used to denote model parameters, where $p_{\boldsymbol{\theta}}(\mathbf{x}\,|\,\mathbf{z}) = p(\mathbf{x}\,|\,\mathbf{z},\boldsymbol{\theta})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x}) = q(\mathbf{z}\,|\,\mathbf{x},\boldsymbol{\phi})$.

The model jointly learns $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, *i.e.* it learns jointly the data likelihood and the inference mechanism. The parameters of the neural networks are estimated such that the marginal likelihood of the input data, $p_{\boldsymbol{\theta}}(\mathbf{x})$, is maximised under a variational Bayesian approximation:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = D_{\mathrm{KL}}[q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x})\,||\,p_{\boldsymbol{\theta}}(\mathbf{z}\,|\,\mathbf{x})] + L_{\mathrm{VAE}}, \tag{2.8}$$

where the variational lower bound is

$$L_{\mathrm{VAE}} = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x})}\left[\log\ p_{\boldsymbol{\theta}}(\mathbf{x}\,|\,\mathbf{z})\right] - D_{\mathrm{KL}}\left[q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z})\right], \tag{2.9}$$

where D_{KL} is the Kullback-Leibler divergence, which is defined as

$$D_{\mathrm{KL}}\left[\,p(\mathbf{x})\,||\,q(\mathbf{x})\,\right] = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\log\frac{p(\mathbf{x})}{q(\mathbf{x})}\right], \tag{2.10}$$

and it will be referred to as the KL divergence henceforth.

On the right-hand side of equation 2.8, the first term measures the distance between the approximate posterior, $q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x})$, and the true unknown posterior, $p_{\boldsymbol{\theta}}(\mathbf{z}\,|\,\mathbf{x})$. On the right-hand side of the variational lower bound, the first term, log $p_{\boldsymbol{\theta}}(\mathbf{x}\,|\,\mathbf{z})$, is the reconstruction error, and the second term is the KL divergence between the encoder distribution, $q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x})$, and a known prior, $p(\mathbf{z})$. The KL divergence between any two distributions is by construction a non-negative metric. Therefore, maximising the bound, L_{VAE}, will be approximately equivalent to maximising the marginal likelihood, $\log p_{\boldsymbol{\theta}}(\mathbf{x})$, as long as the approximate posterior distribution is complex enough.

For continuous data, the approximate posterior and the data likelihood usually take the shape of multivariate Gaussian distributions with factorised covariance matrices

$$q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x}) = \mathcal{N}\big(\,\boldsymbol{\rho}(\mathbf{x}), \boldsymbol{\omega}(\mathbf{x})^2\,\mathbf{I}\,\big), \tag{2.11}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}\,|\,\mathbf{z}) = \mathcal{N}\big(\,\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\sigma}(\mathbf{z})^2\,\mathbf{I}\,\big), \tag{2.12}$$

where $\mathbf{x}$ is the image as a column vector, the means $\boldsymbol{\mu}(\mathbf{z})$, $\boldsymbol{\rho}(\mathbf{x})$ and variances $\boldsymbol{\sigma}(\mathbf{z})^2$, $\boldsymbol{\omega}(\mathbf{x})^2$ are (non-linear) functions of their arguments. A diagram of a VAE model using Gaus-

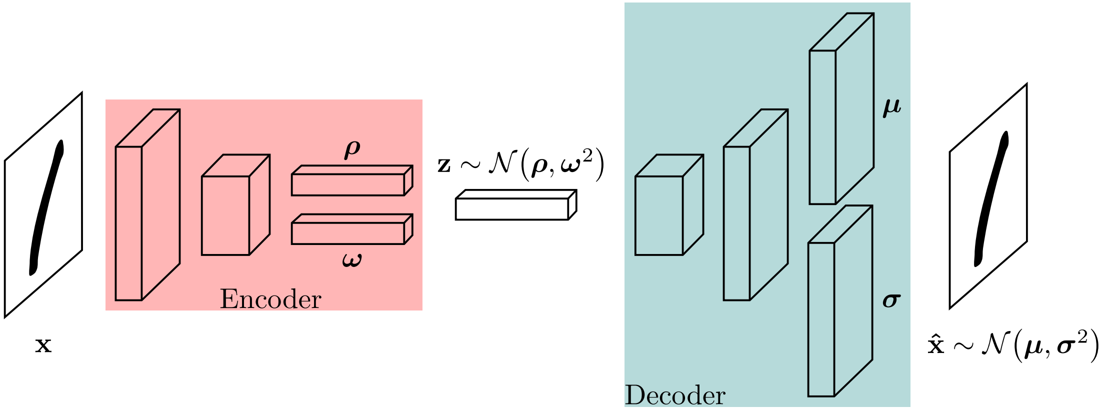

$\boldsymbol{\rho}$
$\boldsymbol{\omega}$
Encoder
$\mathbf{x}$
$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\rho}, \boldsymbol{\omega}^2)$
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}$
Decoder
$\hat{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$

Figure 2-2: Diagram of a Variational Autoencoder (VAE) [Rezende et al., 2014, Kingma and Welling, 2014] employing Gaussian distributions, as indicated in equations 2.11 and 2.12. The encoder infers the parameters, $\boldsymbol{\rho}$ and $\boldsymbol{\omega}$, of the latent distribution, $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$, of each input image, $\mathbf{x}$. A sample $\mathbf{z}$ is drawn from the latent distribution and used as input for the decoder. This network does the reverse process, transforming from the latent variable to the data distribution, $p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z})$, characterised by $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Again, a sample $\hat{\mathbf{x}}$ might be drawn from the data distribution.

sian distributions for the data likelihood and the approximate posterior is shown in Fig. 2-2. The decoder network outputs $\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\sigma}(\mathbf{z})^2$, which defines the probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z})$. Similarly, the encoder network outputs $\boldsymbol{\rho}(\mathbf{x}), \boldsymbol{\omega}(\mathbf{x})^2$, defining $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$. A slight abuse of notation is used to highlight the factorised nature of the distribution, where $\boldsymbol{\sigma}(\mathbf{z})^2\,\mathbf{I}$ denotes a diagonal matrix, such that $\mathrm{diag}(\boldsymbol{\sigma}(\mathbf{z})^2\,\mathbf{I}) = \boldsymbol{\sigma}(\mathbf{z})^2$, and similarly for $\boldsymbol{\omega}(\mathbf{x})^2\,\mathbf{I}$. Thus, the data likelihood is equivalent to the forward model:

$$\mathbf{x} = \boldsymbol{\mu}(\mathbf{z}) + \boldsymbol{\epsilon}(\mathbf{z}), \tag{2.13}$$

where $\boldsymbol{\epsilon}(\mathbf{z}) \sim \mathcal{N}\big(\mathbf{0}, \boldsymbol{\sigma}(\mathbf{z})^2\mathbf{I}\big)$ is commonly considered as unstructured noise inherent in the data.

The prior for the latent variables distribution is usually set to an isotropic Gaussian distribution

$$p(\mathbf{z}) = \mathcal{N}\big(\mathbf{0}, \mathbf{I}\big), \tag{2.14}$$

as this simplifies the KL term in equation 2.9 to

$$D_{\mathrm{KL}}\left[q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z})\right] = \frac{1}{2}\sum_{k=1}^{K}\left(\rho(x)_k^2 + \omega(x)_k^2 - \log\left(\omega(x)_k^2\right) - 1\right), \tag{2.15}$$

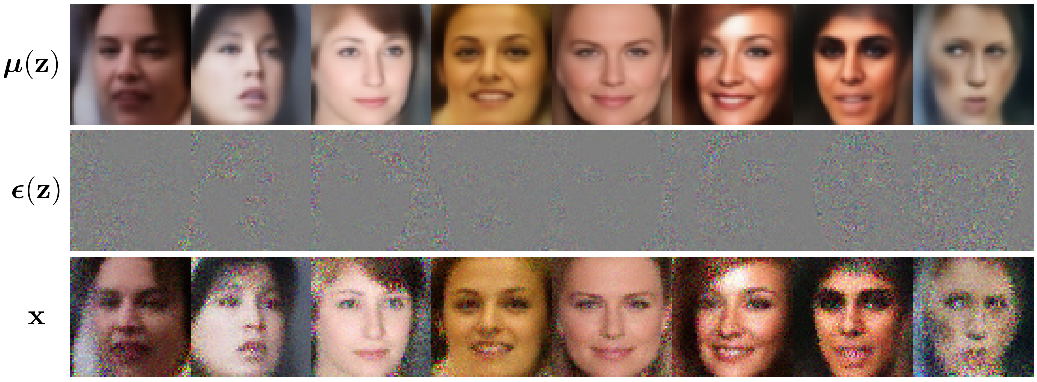

$\boldsymbol{\mu}(\mathbf{z})$
$\boldsymbol{\epsilon}(\mathbf{z})$
$\mathbf{x}$

Figure 2-3: Samples from VAE model trained on the CelebA faces dataset [Liu et al., 2015] at a resolution of 64×64. For the predicted Gaussian distribution, $\mathbf{x} \sim \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\sigma}(\mathbf{z})^2\,\mathbf{I}\big)$, in the first row we show $\boldsymbol{\mu}(\mathbf{z})$, in the second $\boldsymbol{\epsilon}(\mathbf{z})$, and in the third $\mathbf{x} = \boldsymbol{\mu}(\mathbf{z}) + \boldsymbol{\epsilon}(\mathbf{z})$. The mean images, $\boldsymbol{\mu}(\mathbf{z})$, lack the high-frequency detail present in natural face images.

where $\rho(x)_k$ and $\omega(x)_k$ are the k^{th} elements in $\boldsymbol{\rho}(\mathbf{x})$ and $\boldsymbol{\omega}(\mathbf{x})$, respectively, and K is the dimensionality of the latent space vector, $\mathbf{z}$.

VAE models are known to generate blurry outputs, as demonstrated in Fig. 2-3. The factorised Gaussian likelihood, described above, assumes that the data contains low levels of unstructured noise. It is expected that the noiseless data would be well modelled by the $\boldsymbol{\mu}(\mathbf{z})$, while $\boldsymbol{\epsilon}(\mathbf{z})$ could be ignored as we would not be interested in adding the noise back. As the blurriness appears in the mean images, $\boldsymbol{\mu}(\mathbf{z})$, this entails high levels of noise in $\boldsymbol{\epsilon}(\mathbf{z})$.

There are few possible explanations for the observed blurriness. Firstly, the direction of the KL divergence in equation 2.8 pushes the model to mode covering. In other words, to place mass of the distribution in areas where there is no data, in order to cover disconnected regions in the observed distribution. Secondly, the lack of complexity in the approximate posterior distribution could be negatively affecting the model. The real posterior, $p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, \mathbf{x})$, might be a complex distribution, yet the approximate posterior, $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$, is limited to be a factorised Gaussian. Thirdly, the likelihood model, $p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z})$, is also a factorised Gaussian, which implies an unreasonable requirement of exact pixel correspondence between the input and reconstructed images during training.

Extensions of the model can be classified in three distinct groups, where the last two seek to address the blurriness issues:

1. **approximate posterior regularisation** techniques seek to improve the quality of the learned latent space,

2. employing **complex approximate posterior** distributions (instead of factorised Gaussians), in order to reduce the approximation error in equation 2.9,

3. employing **complex likelihood** distributions (instead of factorised Gaussians), to more effectively model the data distribution.

Each approach will be discussed in more detail in subsequent sections.

2.4.1 Approximate posterior regularisation

Several regularisation terms have been proposed in order to improve the quality of the learned latent space. Defining the properties of a good latent space is not a trivial task, and it is hard to measure quantitatively. A popular objective is that of a disentangled representation, which seeks to find independent factors of variations in an unsupervised manner. For example, in a dataset of face images we might wish to uncover head pose or hair colour as one of the independent factors. A detailed overview of such methods applied to VAEs is provided in [Tschannen et al., 2018], which we summarise in this section.

These methods can be seen as adding some regularisation term to either the approximate posterior or the aggregated approximate posterior

$$L_{\text{RVAE}} = L_{\text{VAE}} - \lambda_1 \mathbb{E}_{p(\mathbf{x})}\left[R_1(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))\right] - \lambda_2 R_2(q_{\boldsymbol{\phi}}(\mathbf{z})), \tag{2.16}$$

where the aggregated approximate posterior, $q_{\boldsymbol{\phi}}(\mathbf{z}) = \frac{1}{J}\sum_{j=1}^{J} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_j)$, is a marginalisation of the approximate posterior over the dataset, R_1 and R_2 are regularisation functions, and λ_1 and λ_2 are user-defined hyper-parameters. Although, $q_{\boldsymbol{\phi}}(\mathbf{z})$ cannot be tractably computed, acceptable approximations can be evaluated from the mini-batch during each training step.

An overview of the different regularisation functions that have been proposed is shown in Table 2.1. Better matching the factorised prior has been shown to lead to more disentangled representations [Higgins et al., 2017], as this more tightly constraints the shape of the latent space. Objectives derived with the aim to increase the mutual information between $\mathbf{x}$ and $\mathbf{z}$, $I_{q_\phi}(x; z)$, have also been shown to help [Zhao et al., 2017b, Kim and Mnih, 2018, Chen et al., 2018a]. This mutual information regularisation has also been found useful when employing powerful decoder distributions, which are capable of modelling the data without using the latent variables [Makhzani and Frey, 2017].

Method	R_1	R_2
β-VAE [Higgins et al., 2017]	$D_{\mathrm{KL}}(q_\phi(z\mid x)\\|p(z))$	
InfoVAE [Zhao et al., 2017b]	$D_{\mathrm{KL}}(q_\phi(z\mid x)\\|p(z))$	$D_{\mathrm{KL}}(q_\phi(z)\\|p(z))$
FactorVAE [Kim and Mnih, 2018]		$\mathrm{TC}(q_\phi(z))$
PixelGAN-AE [Makhzani and Frey, 2017]	$-I_{q_\phi}(x;z)$	
DIP-VAE [Kumar et al., 2018]		$\\|\mathrm{Cov}_{q_\phi(z)}[z]-I\\|_{\mathrm{F}}^2$
HFVAE [Esmaeili et al., 2018]	$-I_{q_\phi}(x;z)$	$R_{\mathcal{G}}(q_\phi(z))+\lambda_2' \sum_{G\in\mathcal{G}} R_{\mathcal{G}}(q_\phi(z))$
HSIC-VAE [Lopez et al., 2018]		$\mathrm{HSIC}(q_\phi(z_{G_1}), q_\phi(z_{G_2}))$
Info. dropout [Achille and Soatto, 2018]	$D_{\mathrm{KL}}(q_\phi(z\mid x)\\|p(z))$	$\mathrm{TC}(q_\phi(z))$

Table 2.1: Overview over different choices of the regularisation functions for the approximate posterior $R_1(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))$ and the aggregated approximate posterior $R_2(q_{\boldsymbol{\phi}}(\mathbf{z}))$ of VAE [Kingma and Welling, 2014, Rezende et al., 2014] models. The loss function in which these regularisation methods are applied is defined in Eq. 2.16.[2]

Directly encouraging disentanglement by matching the covariance of $q_{\boldsymbol{\phi}}(\mathbf{z})$ to an isotropic Gaussian prior has also been explored [Kumar et al., 2018]. Another line of works seeks independence between clusters in the latent space as a way to achieve disentangled representations [Esmaeili et al., 2018, Lopez et al., 2018].

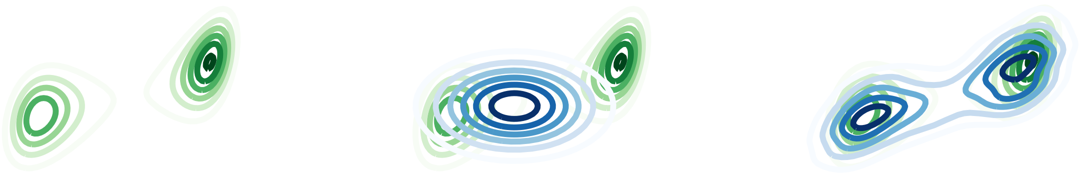
Derivations from the information bottleneck objective [Achille and Soatto, 2018] also lead to regularised objectives. Alemi *et al.* [Alemi et al., 2017] arrive at the aforementioned objective of Higgins *et al.* [Higgins et al., 2017], which seeks to better match the factorised prior. An alternative derivation [Achille and Soatto, 2018] also includes a total correlation term to increase the dependence between $\mathbf{x}$ and $\mathbf{z}$.

These methods show great promise in learning useful unsupervised representations from complex high-dimensional data, such as natural images. However, they still require extensive hyper-parameter tuning, λ_1 and λ_2, and there are no guarantees that they will uncover useful factors of variation for a given dataset.

2.4.2 Complex approximate posterior

In a standard VAE formulation there are two approximation errors. First, the error due to employing a variational approximation to the true posterior. In other words, the factorised multivariate Gaussian (eq. 2.11) has been considered a poor approximation to the true posterior (eq. 2.8) [Cremer et al., 2018]. The second error is due to the inference process being amortised over the dataset, as a network is used for the task, rather than using a method that would provide the optimal lower bound. This is done

[2] Table reproduced from [Tschannen et al., 2018].



(a) True posterior

(b) Gaussian approximate posterior

(c) Flow approximate posterior

Figure 2-4: Approximate posterior distributions for a VAE with a 2D latent space. The true multimodal posterior distribution is shown in a). A standard Gaussian approximation for the posterior is unable to capture the multi-modality of the true posterior. A more complex flow approximation for the posterior [Rezende and Mohamed, 2015] is able to better capture the true distribution.[3]

due to computational constraints, as evaluating the optimal bound is not trivial. An empirical analysis of the errors due to both approximations is provided in [Cremer et al., 2018].

In this section we describe several methods that seek to reduce the error due to employing a variational approximation to the true posterior, by allowing a more complex approximate distribution, as shown in Fig. 2-4. The hypothesis being that the performance of the model might be improved by providing a tighter variational bound, eq. 2.9.

This has been achieved by applying transformations with tractable inverses to the posterior distributions [Rezende and Mohamed, 2015]. The type of allowed transformations involves a restricted set of functions that have triangular Jacobians, which makes the log determinant evaluation tractable, such as planar flows [Rezende and Mohamed, 2015]. Moreover they are required to be composable, as to provide a simple way to increase their complexity by stacking several of them. The type of functions has also been extended to include a family of complex autoregressive transformations [Kingma et al., 2016] which have the aforementioned properties.

An alternative approach to increase the complexity of the approximate posterior is to switch to an implicit distribution. As implicit methods are based on sampling, this means that the KL divergence term in equation 2.9 must be computed with samples. Two approaches have been proposed, one based on a GAN model, which uses

[3] Images courtesy of [Cremer et al., 2018].

a discriminator to distinguish between samples from the prior and samples from the approximate posterior [Mescheder et al., 2017]. The other approach uses the Stein operator to approximate the distribution based on a set of particles [Pu et al., 2017a].

An importance weighted approach [Burda et al., 2016] has also been explored, where a fixed number of samples from the approximate posterior are weighted to provide a tighter variational bound. This was later re-interpreted as defining an implicit distribution over the approximate posterior [Cremer et al., 2017]. By having a tighter lower bound the test likelihood is improved with this approach. However, as the model is allowed to generate several reconstructions during training, not all of which need to be good, the quality of individual samples is on average reduced. Moreover, picking only the good samples is not straightforward, as the likelihood of the samples cannot be evaluated easily.

Later work showed tighter bounds can be detrimental for the encoder, and this can affect the performance of the whole model [Rainforth et al., 2018]. To address this issue, Rainforth *et al.* propose to use the importance weighted loss [Burda et al., 2016] for the decoder, and the VAE loss with multiple samples for the encoder.

The gains obtained by these approaches have been marginal in terms of model likelihood. They have been shown to be more useful in terms of the quality of the latent space distribution, which is useful for our image editing objective. Still, measuring the quality of the latent space is an open problem, as there are no reliable metrics [Higgins et al., 2017], and ground truth data only exists for simplified synthetic scenarios.

2.4.3 Complex likelihood

The use of a factorised multivariate Gaussian to model the data likelihood, in equation 2.12, has been challenged as too constraining, and as the main cause for the blurry outputs commonly observed when generating images with a VAE [Larsen et al., 2016, Hou et al., 2017, Gulrajani et al., 2017b]. Several extensions, which seek to employ more complex likelihood distributions have been proposed.

PixelVAE [Gulrajani et al., 2017b] substituted the factorised Gaussian distribution with a PixelCNN [Oord et al., 2016] distribution, as shown in Fig. 2-5. PixelCNN defines a tractable distribution which is defined autoregressively, such that there is one categorical distribution per pixel, which depends on the previous pixels. In PixelVAE,

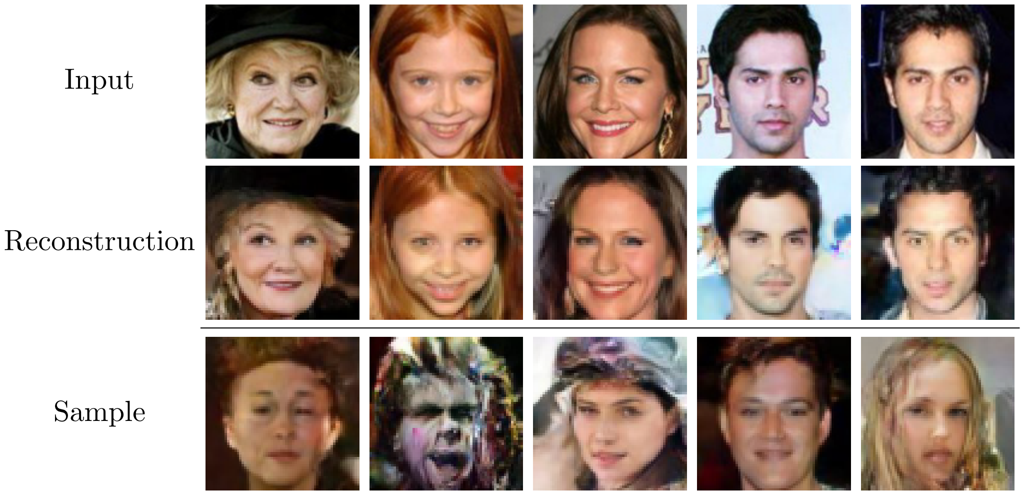


Figure 2-5: Reconstructions and samples drawn from a PixelVAE [Gulrajani et al., 2017b]. The images produced by the model contain high-frequency details. However, these additional details do not correspond to the input images.

the autoregressive distribution is also conditioned on the VAE latent variable:

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \prod_i p\left(x_i|\mathbf{z}, \{x_k\}_{k\in K}\right) \tag{2.17}$$

$$p\left(x_i|\mathbf{z}, \{x_k\}_{k\in K}\right) = \text{Cat}\left[\boldsymbol{\lambda}\left(\mathbf{z}, \{x_k\}_{k\in K}\right)\right], \tag{2.18}$$

where $K = [0, 1, \cdots, i-2, i-1]$ is the set of pixel values that were generated before pixel i, and $\boldsymbol{\lambda}(\mathbf{z}, \{x_k\}_{k\in K})$ is a learnable non-linear function of the input variables, *i.e.* a neural network. Due to computational constraints, the set of pixels K is usually reduced to a small neighbourhood around x_i. Employing this distribution allows modelling correlations in the pixel values, which in turn enables the model to produce high-frequency details. The autoregressive nature of the distribution implies that each pixel must be sampled sequentially. As x_i is a vectorised image, this formulation induces directionality in the likelihood distribution, *i.e.* the pixels are sampled in fixed order, from left to right and top to bottom. The sampling process involves as many forward passes of the network as the number of pixels, which is computationally demanding. Optimisations in the sampling process have been explored [Salimans et al., 2017, Reed et al., 2017], nevertheless it still remains a costly process. Moreover, as shown in Fig. 2-5 the additional high-frequency detail does not correspond to the inputs during reconstructions, which limits the applicability of the model for image editing tasks.

Models that build upon this technique have found that the PixelCNN distribution is complex enough to be able to model the data without using any information from the latent variables [Makhzani and Frey, 2017]. In other words, if the latents are not needed for reconstructing the inputs during training, the model can find a trivial solution for the KL term (in eq. 2.9) by setting the latents to follow the prior. This has led to employing some of the regularisation techniques presented in the previous section. A more detailed discussion on this issue and possible solutions can be found in the work by Chen *et al.* [Chen et al., 2017].

A perceptual loss [Gatys et al., 2015] has been used to evaluate the data likelihood [Hou et al., 2017]. The perceptual loss is defined over some hidden layers of a pretrained classification network as

$$L_{per} = \|\boldsymbol{\psi}(\mathbf{x}) - \boldsymbol{\psi}(\hat{\mathbf{x}})\|_2, \tag{2.19}$$

where $\boldsymbol{\psi}$ are the values of the hidden layer in the classification network, $\hat{\mathbf{x}}$ is the image modelled by the network and $\mathbf{x}$ is the real data.

Combinations with implicit likelihood methods have also been explored. The VAE/GAN model [Larsen et al., 2016] evaluates the data likelihood using a perceptual loss. However, it uses the features from the intermediate layers of a discriminator, rather than using a pretained classification network. The discriminator is added as part of an adversarial loss, which is optimised jointly with the lower bound in equation 2.9. Two scalar hyper-parameter are needed to control the relative weight of the adversarial and the perceptual loss, with respect to the KL divergence. This model was simplified by merging the encoder and the discriminator into a single network [Huang et al., 2018]. The perceptual and the adversarial terms are substituted respectively, by an L2 reconstruction loss in pixel space, and an energy-based adversarial term [Zhao et al., 2017a] on the latents.

These approaches [Larsen et al., 2016, Huang et al., 2018, Hou et al., 2017] are able to generate images with high-frequency content, however the added detail need not correspond with the original image. Furthermore, not evaluating the likelihood in pixel space has a number of drawbacks. It requires a weighting hyper-parameter for combining the data likelihood, which may be un-normalised such as the one in equation 2.19, and the KL divergence in equation 2.9. Moreover, it complicates model comparison, as the likelihood in pixel space on a test set can no longer be easily evaluated.

A tractable method that allows generating images with high-frequency details with VAEs will be presented in Chapter 3. Unlike PixelVAE, our method only requires a single forward pass of the network for sampling and likelihood evaluation. In contrast to the methods that combine the VAEs with implicit likelihood approaches, our model evaluates a normalised likelihood in pixel space.

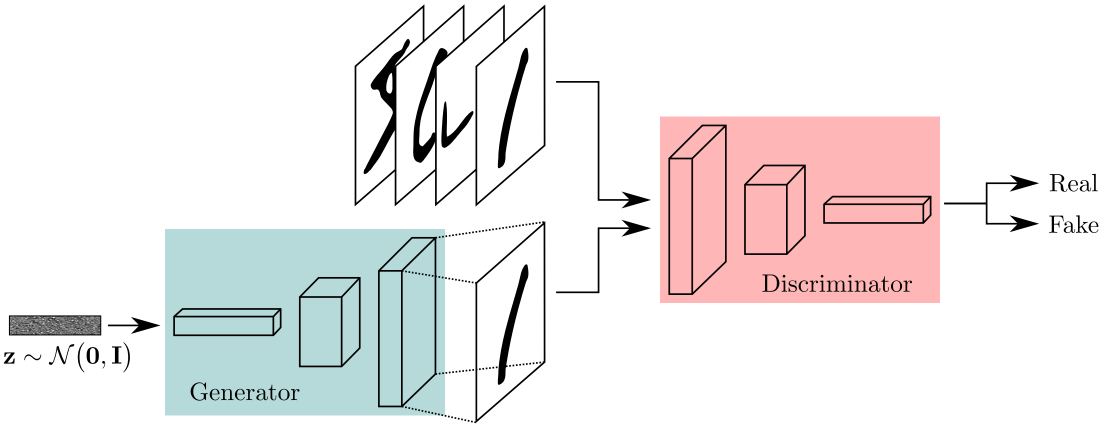


Figure 2-6: Diagram of a Generative Adversarial Network (GAN) [Goodfellow et al., 2014]. The generator transforms random noise, $\mathbf{z}$, into realistic samples, while the discriminator tries to tell apart the real images from the generated data.

2.5 Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN) [Goodfellow et al., 2014] consist of two parts, a generator and a discriminator, as shown in Fig. 2-6. The generator produces samples that resemble the data distribution samples, and the discriminator classifies data samples as real or fake. The discriminator is trained with the real examples drawn from a training set and the fake examples as the output of the generator. The generator is trained to fool the discriminator into classifying generated samples as real. Formally, GANs are defined by a minimax game objective:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))], \tag{2.20}$$

where $\mathbf{x}$ denotes input data from an empirical distribution $p_{\text{data}}(\mathbf{x})$ (the training set), $\mathbf{z}$ is a random variable drawn from an arbitrary distribution $p(\mathbf{z})$, G is the generator and D is the discriminator. Thus, generating samples from the model can be done trivially by

$$\mathbf{z} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{I}\big), \tag{2.21}$$

$$\mathbf{x} = G(\mathbf{z}), \tag{2.22}$$

where the latent variable is sampled from a simple arbitrary distribution, and the generator transforms the latent variable to image space. Example images generated by this model are shown in Fig. 2-7.

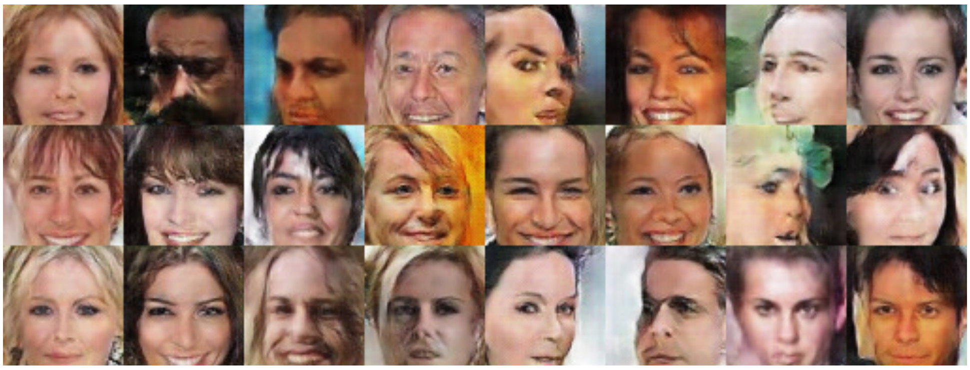


Figure 2-7: Example images generated by a GAN model trained on the CelebA faces dataset [Liu et al., 2015] at a resolution of 64×64. [4]

In practice, both the generator and the discriminator are implemented as neural networks, with learnable weights and biases. With a slight abuse of notation, G and D also refer to the parameters that are optimised in equation 2.20. These parameters are learned in alternating steps, where the generator parameters are updated for the min, and the discriminator ones are updated for the max.

Note that the optimum of equation 2.20 is a saddle point, where both the discriminator and the generator have the optimal parameters. However, it is known that there are oscillating situations where neither network improves over time [Goodfellow, 2016].

An additional problem, empirically found in GAN models, is that the second term in equation 2.20 saturates early in training (*i.e.* the discriminator confidently rejects the generator samples), providing zero gradients for the generator. In practice a more robust alternative [Goodfellow et al., 2014] is commonly used for the generator

$$\max_G \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\,\log(D(G(\mathbf{z})))\,]. \tag{2.23}$$

In equation 2.20, the target for the generator was constructed by flipping the sign of the discriminator output. In this version, the target itself is flipped, as the generator is now maximising the probability of the discriminator being mistaken, *i.e.* of classifying generated data as real. This formulation ensures that the generator still receives gradients, even if the discriminator is highly confident when rejecting the generator samples.

GANs have also been interpreted [Goodfellow et al., 2014] as minimising the Jensen-Shannon divergence between the generator distribution, p_g, and the data distribution,

[4] Images courtesy of [Radford et al., 2016].

p_{data}. First, we define $L(D, G)$ as the loss that the discriminator maximises

$$L(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\, \log(D(\mathbf{x}))\,] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\, \log(1 - D(G(\mathbf{z})))\,]. \tag{2.24}$$

For an optimal discriminator, D^*, this loss is equivalent to

$$L(D^*, G) = 2\,(\text{JSD}\,(\, p_{\text{data}} \,||\, p_g\,) - \log(2))\,, \tag{2.25}$$

where the optimal discriminator is defined as

$$D^* = \frac{p_{\text{data}}}{p_{\text{data}} + p_g}, \tag{2.26}$$

and JSD is the Jensen-Shannon divergence, which is defined as:

$$\text{JSD}\,(\, p \,||\, q\,) = \frac{1}{2} D_{\text{KL}} \left(p \,||\, \frac{1}{2}(p+q) \right) + \frac{1}{2} D_{\text{KL}} \left(q \,||\, \frac{1}{2}(p+q) \right), \tag{2.27}$$

where p and q are two arbitrary distributions, and D_{KL} is the Kullback-Leibler divergence defined in equation 2.10. GAN models have also been extend to optimise other f-divergences [Nowozin et al., 2016], such as the Kullback-Leibler divergence or the Pearson χ^2 divergence. However, this view does not correspond to how GAN models are used in practice, as this derivation is only valid when using the original objective, described in equation 2.20. For the modified objective in equation 2.23, the loss for an optimal discriminator corresponds to a combination of KL and JSD divergences [Arjovsky and Bottou, 2017]. Still, this interpretation is also dubious. It has been demonstrated [Arjovsky and Bottou, 2017] that the variance of the gradients for updating the generator increases as the discriminator approaches optimality. Therefore, the discriminator is rarely trained to optimality in practice.

Extensions to this model can be classified into four groups:

1. improving the model **stability**, as the modified objective in equation 2.23 is not enough to guarantee stable training,

2. reducing **mode collapse**, which manifests as the model only learning about a small part of the real data distribution,

3. addressing the lack of an **inference** mechanism in the model,

4. providing reliable **evaluation** metrics is also an area of interest, as the likelihood on a test set cannot be easily evaluated,

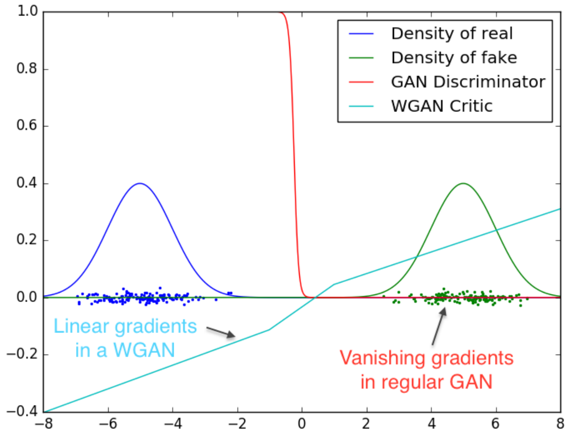


Figure 2-8: Example of vanishing gradients for a two-dimensional synthetic distribution, when there is no overlap between the generated samples and the real data. An optimal GAN discriminator, which corresponds to equation 2.20, has a constant loss on both the real and the generated data points. Consequently, producing gradients of little value for the generator. An optimal W-GAN [Arjovsky et al., 2017] discriminator, which corresponds to equation 2.28, has a linear loss function, that reflects distance between the distributions (an Integral Probability Metric [Müller, 1997]). Therefore, providing gradients that push the generator towards the real data even when the distributions do not overlap.[5]

Each approach will be discussed in more detail in the sections below.

2.5.1 Stability

Even when employing the modified objective defined in equation 2.23, GAN models are notoriously hard to train [Radford et al., 2016]. The stability issues with GAN training have been mostly attributed to the discriminator providing either zero or unbounded gradients to the generator [Arjovsky and Bottou, 2017]. Attempting to learn data distributions that lie on low-dimensional manifolds is believed to be the underlying cause of the issue. This allows the discriminator to easily discriminate between the generated and the data distribution as there can be little to not overlap between them. In such a case, the discriminator loss for the generated data is constant, thus providing no gradient information for the generator, as shown in Fig. 2-8.

A line of work to address the stability issues focused on carefully tuning a number of hyper-parameters, such as the architecture and the learning rates to achieve more stable behaviour [Radford et al., 2016, Metz et al., 2017, Zhang et al., 2018, Karras

[5] Image courtesy of [Arjovsky et al., 2017].

et al., 2019]. These methods provide significant gains in terms of stability and performance. Still, these techniques are mostly empirically found heuristics, which do not fully address the underlying issues in GAN training.

A simple solution to enforce overlapping between the learned and the data distributions is to add noise, either to the labels or to the images [Salimans et al., 2016, Arjovsky and Bottou, 2017]. However, this can lead to degraded performance and the amount of noise required is data and architecture dependant.

A more principled approach is to substitute the Jensen-Shannon divergence with a metric that has the concept of distance between points [Arjovsky and Bottou, 2017], where Integral Probability Metrics [Müller, 1997] are a family of such metrics. In other words, a metric that induces a weak topology on the loss, thus making it easier for the generated distribution to converge to the real data distribution. The Wasserstein or earth mover distance has been proposed [Arjovsky et al., 2017] as one such metric. This leads to the following alternative objective for the GAN model:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\,D(\mathbf{x})\,] - \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\,D(G(\mathbf{z}))\,]. \tag{2.28}$$

Employing the Wasserstein distance requires imposing certain smoothness constrains on the discriminator. Namely, the discriminator must be a 1-Lipschitz continuous function. For a function, f, that takes as input a scalar, K-Lipschitz continuity is defined as

$$|f(x_0) - f(x_1)| \leq K|x_0 - x_1|, \tag{2.29}$$

where x_0 and x_1 are any two inputs of the function, and $K = 1$ for 1-Lipschitz continuity. Intuitively, it implies that the growth and wiggling of the function is bounded. Several approaches have been proposed to enforce 1-Lipschitz continuity in the discriminator, including clipping the weights [Arjovsky et al., 2017] and penalising its gradients [Gulrajani et al., 2017a]. The gradient penalty term is defined as

$$L_{gp} = \mathbb{E}_{\dot{\mathbf{x}}}\left[(\|\nabla_{\dot{\mathbf{x}}} D(\dot{\mathbf{x}})\|_2 - 1)^2\right], \tag{2.30}$$

where $\dot{\mathbf{x}}$ is sampled uniformly along a straight line between a real and a generated image. This loss is weighted by scalar and added in the discriminator step of equation 2.28. Therefore, this approach adds a new hyper-parameter to the model, which is the weight of the gradient penalty loss.

Later work [Miyato et al., 2018] employs a spectral weight normalisation technique

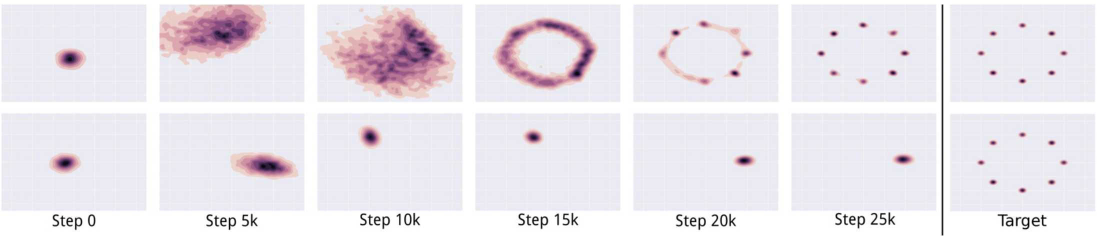


Figure 2-9: Example of mode collapse in GANs on a synthetic two-dimensional distribution. The target distribution is shown on the right-most column, which consists of 8 Gaussians. The bottom row shows the distribution learned by a GAN model, which only covers a single Gaussian. The top row shows the distribution learned by an Unrolled GAN [Metz et al., 2017], which is able to cover all the modes.[6]

to constrain the discriminator (and the generator) to be a 1-Lipschitz function, while using the non-saturating loss in equation 2.23. This was shown to improve the stability of the model in a similar degree as the previous method, with the advantage that it is hyper-parameter free.

GAN stability is still an open problem, as empirical studies [Brock et al., 2019] on generating images at large resolutions show that the previous regularisation techniques significantly affect the model performance. Thus they advocate for using the non-saturating loss and to simply keep the model on the last iteration before the gradients explode or vanish.

2.5.2 Mode collapse

GAN models are also prone to mode collapse [Arjovsky and Bottou, 2017]. This issue is observed as the generator learns to model only a small part of the data distribution, as shown in Fig. 2-9. This is due to the discriminator evaluating the realism of each image independently, thus it cannot evaluate whether the samples from the generator cover the whole data distribution or are concentrated on a small portion of the space with high density. In the limit, a valid optimum for the generator is to memorise a single training image and reproduce it regardless of the value of the latent vector.

One avenue of work seeks to provide the discriminator with additional knowledge by allowing it to evaluate some batch statistics, as an approximation of evaluating joint statistics on the whole dataset. This includes mini-batch discrimination [Salimans et al., 2016], which changes the discriminator architecture such that the features from

[6] Image courtesy of [Metz et al., 2017].

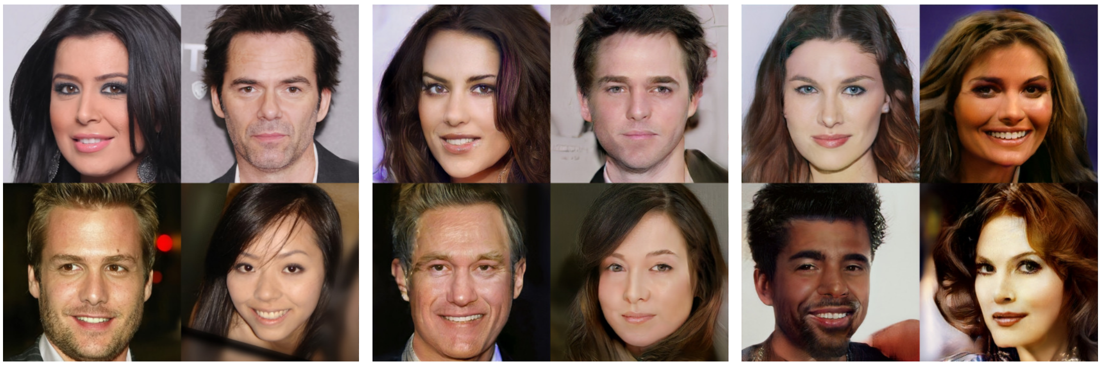
intermediate layers from one image in the batch are concatenated with the features from the rest of the batch, thus allowing the discriminator to make the decision for each image using side information from the other elements in the batch. An average over the standard deviation of each feature over the minibatch has also been used as additional information for the discriminator [Karras et al., 2018]. The idea is that if the standard deviation is significantly smaller on the generated batch, the discriminator will be able to classify the batch as fake. The main issue with these methods is that the approximation is limited by the batch size, which for moderate image sizes is usually no larger than 16 due to hardware constrains [Portenier et al., 2018, Choi et al., 2018, Karras et al., 2018].

Jointly training an encoder network to project images back to the latent space has also been shown to reduce mode collapse [Donahue et al., 2017, Dumoulin et al., 2017]. The optimal generator and encoders are inverses of each other. As the encoder must project all input images to the latent space, the generator is encouraged to cover the input space. This idea was further developed to include a reconstruction term, $\|\mathbf{z} - F(G(\mathbf{z}))\|_2$, where F is the encoder network, on the latent space [Srivastava et al., 2017]. One of the advantages of this approach is that it allows to provide gradients to the generator when the discriminator is constant. Moreover, the latent space is by convention Gaussian distributed, which justifies the use of an L2 reconstruction on the latent vectors.

Allowing the generator to predict several outputs has also been explored [Ghosh et al., 2018]. This is achieved by replicating the last layer of the generator c times. The discriminator is also extended to classify the images in $c + 1$ domains, where the first c are considered fake. Thus, the discriminator is trying to guess which of the c output layers in the generator, produced a specific sample. This pushes the generator to produce distinct images in each output layer. However, the scalability of this method is limited as it requires layer replication in the networks.

In a similar fashion when labelled data is available, this extra information has been used [Miyato et al., 2018] to encourage images from all labels to be represented. However, this does not ensure that mode collapse will not arise for intra-label samples, and labelled data is not always available.

The aforementioned methods are able to reduce the mode collapse problem with different degrees of success, however none of them address it completely. It is still an open question whether mode collapse is inevitable when learning the data distribution of images with GAN models.



(a) Input (b) Reconstructions (c) Samples

Figure 2-10: Images generated from GAN model with an inference network [Huang et al., 2018]. The reconstructions and the samples contain high-frequency details, yet the detail in the reconstructions does not match the input.[7]

2.5.3 Inference

The original GAN model lacks an inference method, *i.e.* finding the most likely latent vector, $\mathbf{z}$, for a given input image. This operation is a useful property for editing, as discussed in Section 2.1. An input image can be edited by manipulating its latent vector and generating a new image from the edited latent vector.

Several methods have been explored to provide an inference mechanism to GAN models. The simplest option is to include an encoder network that is trained post hoc to invert the generator function [Odena et al., 2017]. Another technique employs a gradient descent method with an L2 or similar loss in image space, as the generator network is differentiable. This can be further improved by initialising the gradient descent method with the encoder result from the previous approach [Zhu et al., 2016]. Nevertheless, there are problems with these methods as the generator is unaware of the encoder network.

Extensions have been proposed, which allow training the generator and the encoder jointly in a coherent fashion [Donahue et al., 2017, Dumoulin et al., 2017, Srivastava et al., 2017, Huang et al., 2018]. Nonetheless, these models generate reconstructions that only reproduce global features and pose, with the remaining content not matching the inputs, *i.e.* for images containing human faces, there is a loss of identity, as shown in Fig. 2-10.

A likely cause of the mismatch between the input and the reconstruction is the mode

[7] Images courtesy of [Huang et al., 2018].

collapse problem discussed in the previous section. Adding an inference mechanism helps to reduce the mode collapse problem, yet, none of the models address it completely. If significant parts of the input space are not being modelled, the model would struggle to accurately reconstruct images that are far from the subset of the training data that it managed to model. Therefore, making it less suitable for image editing, as the output image must resemble the input.

2.5.4 Evaluation

Evaluation of unsupervised generative models is a complex topic. In explicit likelihood methods, a standard approach is to compare the data likelihood of a test dataset. However, as discussed in Section 1.3, there are substantial limitations in employing this metric. This issue is exacerbated in implicit density methods like GAN, where a direct evaluation of the likelihood is not possible. Initial work attempted to use Parzen window estimates as means to approximate the likelihood [Goodfellow et al., 2014]. However, such strategies fail when applied on high dimensional data such as images [Theis et al., 2016]. Instead, a combination of alternative metrics is commonly used for model comparison.

Popular metrics used by current research include the Inception Score [Salimans et al., 2016], the Fréchet Inception Distance [Heusel et al., 2017] and the sliced Wasserstein distance [Karras et al., 2018]. The Inception metrics are based on features from a neural network trained for classification, which means that the metric will be insensitive to the image properties that the classification network learned to generalise over. For example, it is known that classification networks are mostly insensitive to pose and background changes. The sliced Wasserstein distance on the other hand, performs an initial approximation as it is measured over patches rather than whole images, and a second one that projects the 5-dimensional pixels (position and colour) to 1D [Rabin et al., 2012]. Both of these approximations inevitably introduce errors with respect to the true Wasserstein distance.

User studies have also been used for model comparison [Denton et al., 2015, Salimans et al., 2016], by asking a human whether an image is real or generated by a computer. However, user studies require significant effort by the researcher, in terms of design of the survey, recruitment of participants, analysis of the results, as well as monetary costs of running the study. Reproducibility can be an issue for modest number of participants. Moreover, these studies are usually only concerned with image quality, thus neglecting to evaluate mode collapse. These factors limit the scalability and large scale application as a metric, which in practice is only used once in the final stage of a publication.

An important limitation of the aforementioned metrics and user studies is that they do not measure generalisation [Theis et al., 2016]. In other words, a method that memorises a subset of the training data would score perfectly under these metrics.

2.6 Image-to-Image translation

Image-to-Image translation models, such as Pix2Pix [Isola et al., 2017], learn to transform an image from a source domain to a target domain using an adversarial loss [Goodfellow et al., 2014]. A generator network learns to predict a transformed version of the input image such that it appears to belong to the target domain. The discriminator observes image pairs, with one image from each domain, and assesses their collective realism. This approach requires paired training data; *i.e.* each image in the source domain must have a corresponding image in the target domain. Given this restriction, the method is often applied to problems where collecting paired data is easier, such as colourisation, or semantic labels to RGB.

Cycle-GAN [Zhu et al., 2017] overcomes the Pix2Pix limitation of having to operate with paired data, by means of a cycle consistency loss. Given two data domains, A and B, Cycle-GAN learns a pair of transformations $G : A \rightarrow B$ and $H : B \rightarrow A$. The cycle consistency loss is used to learn coherent transformations that preserve a reasonable amount of image content, where the loss is defined as

$$L^a_{\text{cycle}} = \mathbb{E}_{\mathbf{x}_a}\left[\, \|\mathbf{x}_a - H(G(\mathbf{x}_a))\|_1 \,\right], \tag{2.31}$$

where $\mathbf{x}_a$ is a sample image from domain A. The cycle loss for domain B is similarly defined as

$$L^b_{cycle} = \mathbb{E}_{\mathbf{x}_b}\left[\, \|\mathbf{x}_b - G(H(\mathbf{x}_b))\|_1 \,\right]. \tag{2.32}$$

These models are interesting from an image editing perspective, as they allow to learn the transformations from unpaired data. Moreover, the issue of poor reconstructions is significantly diminished in this approach. This is a typical limitation of GAN models, and it was discussed in Section 2.5.3. However, Cycle-GAN models are limited in that they require 2 generators and 2 discriminators for each domain pair, and this does not scale well with an increase in the number of domains.

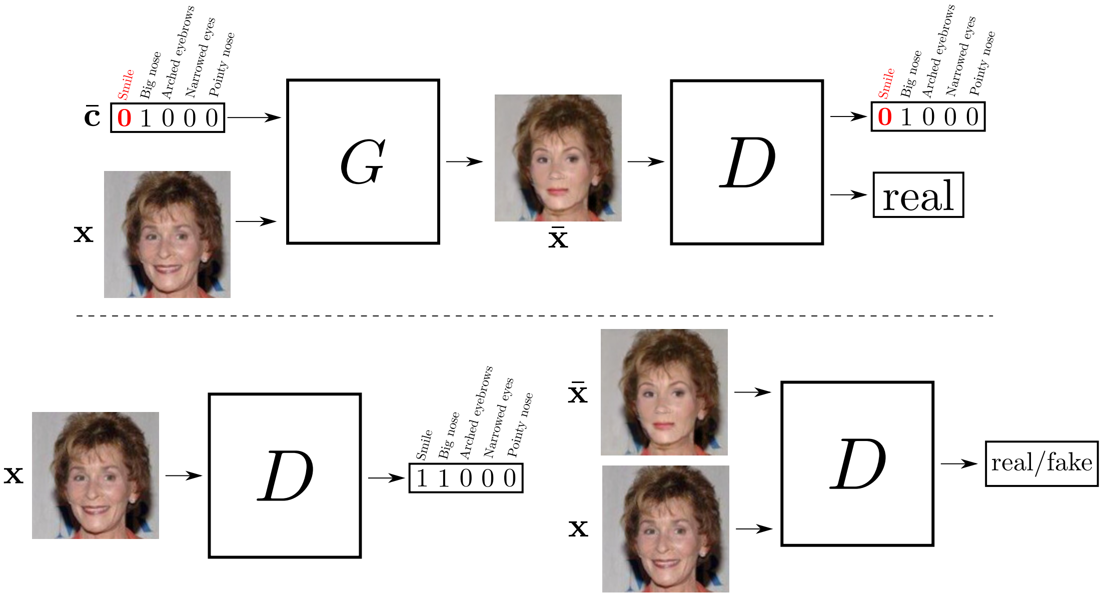


Figure 2-11: Overview of StarGAN [Choi et al., 2018], a state of the art image-to-image translation model. Top row, an input image, $\mathbf{x}$, is modified by a generator, G, given some target domain labels, $\bar{\mathbf{c}}$. The generator tries to fool the discriminator, D, into classifying the generated image, $\bar{\mathbf{x}}$, as real and also as having the target domain labels. Bottom row, the discriminator is trained to accurately predict the original domain labels of the input image. Also, it tries to classify the input image as real, and the generated ones as fake. Not shown, a cycle consistency loss is also employed, where the generator is used again with the generated images and the input domain labels to reconstruct the input image.

Cycle-GAN was generalised by StarGAN [Choi et al., 2018] to require only a single generator and discriminator to translate between multiple domains, as shown in Fig. 2-11. Here, each image $\mathbf{x}$ has a set of domain labels, represented as a binary vector $\mathbf{c}$. The generator, $G(\mathbf{x}, \bar{\mathbf{c}})$, transforms $\mathbf{x}$ to match some target domains indicated by $\bar{\mathbf{c}}$. A classifier is added such that it outputs the probability that $\mathbf{x}$ has the associated domains. This classifier learns the real domain labels from the training data, and it is used on the generator output to ensure the translated image matches the target domains.

A common issue with Cycle-GAN [Zhu et al., 2017] and its extensions is that they find undesired correlations within a domain, leading to excessive image modification. For example, when transforming from dogs to wolves, it tends to edit grass backgrounds to snowy ones, as wolves are usually found in such environments. A number of methods have been proposed to overcome this issue: RESGAN [Shen and Liu, 2017] edits the image via a sparse residual image, which allows restricting the quantity of edited pixels.

Residuals are an overcomplicated representation for edits such as geometric deformations, as the network must model the whole content in a region in order to be able to remove it and add it back in another location. Restricting the edits to a region given by a soft mask, which is also predicted by the generator has also been explored [Mejjati et al., 2018, Pumarola et al., 2018]. The model is complicated by the additional mask prediction network, and it also requires extra care to avoid poor solutions.

CycleGAN-based models have been shown to encode additional information on the generated images that is imperceptible to humans [Chu et al., 2017]. This information is used by the network to be able to reconstruct the original image as required by the cycle loss in equations 2.31 and 2.32. In StarGAN models, this issue manifest as the generator learning to ignore the input domain labels for images which have already been edited, as the edited images are only used at training time for the cycle loss. Adding a triple consistency loss has been shown to diminish this issue [Sanchez and Valstar, 2018], where the loss is defined as

$$L_{\text{tri\_cycle}} = \mathbb{E}_{\mathbf{x},\bar{\mathbf{c}},\hat{\mathbf{c}}}\left[\, \|G(\mathbf{x},\bar{\mathbf{c}}) - G(G(\mathbf{x},\hat{\mathbf{c}}),\bar{\mathbf{c}})\|_1 \,\right], \tag{2.33}$$

where we use the notation of StarGAN with a single generator G, controlled by target labels $\bar{\mathbf{c}}$, where $\mathbf{x}$ is the input image and $\hat{\mathbf{c}}$ denotes intermediate random labels for the domains. The loss encourages the network to produce consistent results regardless of any intermediate steps, $G(\mathbf{x},\hat{\mathbf{c}})$.

2.7 Face image editing

In this section we discuss methods for image editing, with a focus on techniques designed for manipulating faces. There has been much work in this area, where most of the methods presented here follow a machine learning approach, while modern techniques employ deep learning.

The types of edits that are allowed by these methods include: expression changes [Guenter et al., 1998], adding makeup [Guo and Sim, 2009], ageing [Kemelmacher-Shlizerman et al., 2014] or relighting [Blanz and Vetter, 1999]. As it was discussed in the previous chapter, these methods provide controls that allow the direct modification of these high level semantic aspects of the image, while preserving the identity of the subject.

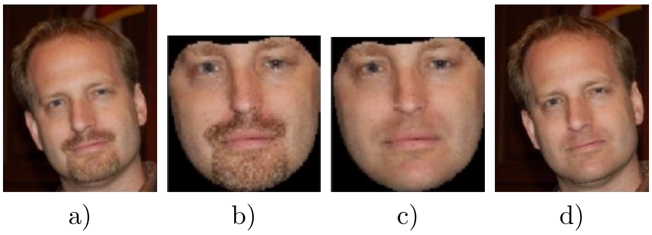


Figure 2-12: Example of image editing with a linear subspace model [Nguyen et al., 2008]. The input image, a), is aligned to a mean face, b), the beard is removed by performing a robust reconstruction in a non-beard linear space, c), and the edited face is projected back to image space, d).[8]

Traditional generative models

Early editing methods are based on modelling faces as a linear combination of components. In the most basic form, this can be written as $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{A}\mathbf{b}$, where $\mathbf{x}$ is the vectorised input image, $\bar{\mathbf{x}}$ is a mean face, $\mathbf{A}$ is a matrix, and $\mathbf{b}$ is a vector containing the mixing coefficients. The $\mathbf{A}\mathbf{b}$ operation can be seen as a linear combination of basis vectors, and novel images may be generated from the model by drawing a sample from the mixing coefficients vector, $\mathbf{b}$, and evaluating $\mathbf{x}$ from it.

Eigen faces [Turk and Pentland, 1991] employ PCA on a training set to estimate the basis matrix, A. This method was originally designed for face recognition, yet, extensions of the model have been used for graphics applications. For example, this approach allows for simple editing operations, such as removing beards in frontal and aligned images [Nguyen et al., 2008], as shown in Fig. 2-12. This is achieved by modelling the image as a sum of two linear combinations, $\mathbf{x} = \mathbf{A}\mathbf{b} + \mathbf{C}\mathbf{d} + \boldsymbol{\epsilon}$, one that models the beard part of the faces, $\mathbf{A}\mathbf{b}$, one that models the non-beard part, $\mathbf{C}\mathbf{d}$, and a residual component, $\boldsymbol{\epsilon}$. Although attractive for their simplicity, these methods have limited capability due to the linearity assumptions.

Active appearance models [Cootes et al., 1998] add some complexity by separating shape from texture. Thus, describing face shape as a mean shape plus a weighted combination of basis, and face texture as another mean plus weighted combination. A face image is constructed by first generating the texture, and then warping it according to the shape. In contrast to previous approaches, this method can handle more complex expressions and texture changes.

Morphable models [Blanz and Vetter, 1999] are an extension of active appearance

[8] Image courtesy of [Nguyen et al., 2008].

models to 3D. Thus, requiring the estimation of shading parameters and fitting a 3D shape model. This better captures the real physics under the image formation process, leading to improved results in most cases. However, the shading parameters are restricted to be Phong coefficients, and in practice the optimisation procedure struggles with real images due to occlusions and large changes in illumination.

In a similar fashion, blendshape models [Guenter et al., 1998] describe a face as a linear combination of shape, and another of texture. However, instead of learning an orthogonal basis from the training set using PCA, the basis matrix is constructed for a single individual. Usually, each component corresponds to a different face expression. This allows the model to provide interpretable controls, as the weighting coefficients correspond directly to a given facial expression. The requirement of several images of the same face in different expressions limits the applicability of this approach, which is further restricted as it also requires video [Garrido et al., 2013] or multiple cameras to be able to reconstruct the 3D shape model.

A probabilistic approach to edit and synthesise face images has also been explored [Mohammed et al., 2009]. This model uses a factor analyser [Bishop, 2006], which is equivalent to adding a noise term to the linear combination of components. Thus, it can be written as $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{A}\mathbf{b} + \boldsymbol{\epsilon}$, where $p(\mathbf{b}) = \mathcal{N}\big(\mathbf{0}, \mathbf{I}\big)$, $p(\boldsymbol{\epsilon}) = \mathcal{N}\big(\mathbf{0}, \boldsymbol{\Sigma}\big)$ and $\boldsymbol{\Sigma}$ is a diagonal covariance matrix. As the images generated by the factor analyser are overly smooth, a patch-based texture synthesis method is used to refine the generated images. Two editing applications were demonstrated: first, directly painting over the image generated by the factor analyser. Second, editing a single expression, where the model is trained with paired images where the subject is captured with a neutral expression, and with the target expression. The probabilistic approach enhances the capabilities of the model, and it also allows using it as an inpainting method.

These linear combination methods are mostly limited by the number of components that they require. In general, these components would cover only the modes of variation that are more common. Thus, failing for less frequent cases that are not well covered in the training set.

Exemplar-based methods

Several methods for editing have been proposed that require (at least) two images during inference, the input image to be edited, and an example image that describes the target edit. This task is difficult as the face in the example image is usually not well

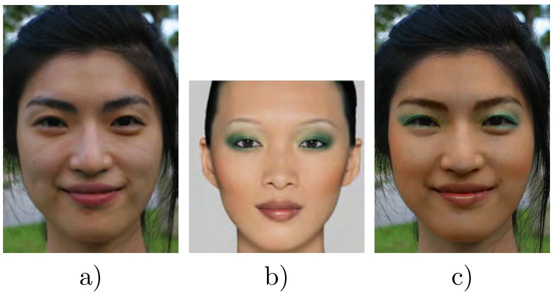


Figure 2-13: Example of image editing by decomposing a face into a face structure layer, a skin detail layer, and colour layer [Guo and Sim, 2009]. The example image b) is warped to match the input image a). The input and the warped example are decomposed into the aforementioned layers. And the editing is performed by combining layers, producing image c).[9]

aligned with the face in the input image in terms of pose. There might be significantly different lighting conditions in both images, and depending on the edit, only a small subset of the texture in the example image should be transferred to the input.

An avenue of work consists of decomposing the image into different layers that simplify the transfer of content from one to the other. This is similar to the aforementioned beard removal approach of [Nguyen et al., 2008], where the face was divided into beard and non-beard components. For instance, makeup transfer has been demonstrated by decomposing the image into three layers: face structure, skin detail and colour [Guo and Sim, 2009], as shown in Fig. 2-13. An alternative consists of a Laplacian pyramid decomposition, which allows for image enhancement and style transfer, *e.g.* for matching more general features, such as overall colour and high-frequency details [Shih et al., 2014]. Later work showed that under a similar decomposition, an example image was not needed for a number of edits [Boyadzhiev et al., 2015]. A pyramid is constructed by splitting the image into low and high-frequency components, splitting those into low and high amplitude components, and finally splitting again into positive and negative components. The authors showed that employing simple operations on the components, such as scaling or addition, lead to consistent image changes. For example, this method allows editing the skin such that it appears either, more oily, smoother or with more blemishes.

Modern approaches that do not rely on deep learning have also proven successful. For example the method of [Averbuch-Elor et al., 2017] operates on a sparse set of landmark correspondences between the input and the exemplar faces. The technique is originally

[9] Image courtesy of [Guo and Sim, 2009].

presented for example videos, rather than a single example image. For simplicity, we will assume an example video containing a single frame in the following discussion. The method first aligns the example face to the input face using a similarity transform. Once aligned, the input image is warped based on the landmarks, which are extended by hallucinating additional landmarks in the background. The hallucinated landmarks serve to avoid undesirable warps outside of the face region. The mouth interior is directly transferred from the example image using Poisson blending [Pérez et al., 2003] and fine scale details are transferred with a ratio image technique [Liu et al., 2001]. The use of these heuristics for editing limits the example image to be somewhat similar to the input image, in order to generate a good quality edit.

By restricting the example image to be an image from the same subject, a number of applications have been developed. For example, editing an image with several people, such that everyone in the picture is looking at the camera and smiling [Agarwala et al., 2004] or automatic deblur and exposure correction [Joshi et al., 2010]. Transfer of open eyes, for correcting images where subject appears with closed eyes [Shu et al., 2016], and expression transfer [Yang et al., 2011] have also been explored.

Both types of exemplar methods, with or without the restriction that the example image belongs to the same person, are designed for restricted tasks, where the domain assumptions limit their applicability to more general settings.

Modern deep learning approaches

Recent image editing methods [Portenier et al., 2018, Dekel et al., 2018, Choi et al., 2018, Geng et al., 2018] employ deep learning techniques. Example images edited by these models are shown in Fig. 1-5 in Chapter 1, and in Fig. 2-14. Particularly popular are the use of GANs, which were discussed in Section 2.5. Especially, GANs in a similar form as the Pix2Pix networks [Isola et al., 2017] discussed in Section 2.6, where the generator is trained with an adversarial loss and an L1 loss to the ground truth image.

Several methods that learn how to reconstruct images from contours have been proposed. Image editing is defined in these models as manipulations of the countours. TalkingHeads [Zakharov et al., 2019] and MakeAFace [Qian et al., 2019] use sparse contours. In TalkingHeads semantic information is added to the contours, indicating which face region they correspond to, *e.g.* mouth, eyes or nose. In the case of MakeAFace, a VAE with a Gaussian mixture prior on the latent space is used, which helps in learning a more disentangled representation of shape (as encoded by the contours) and tex-

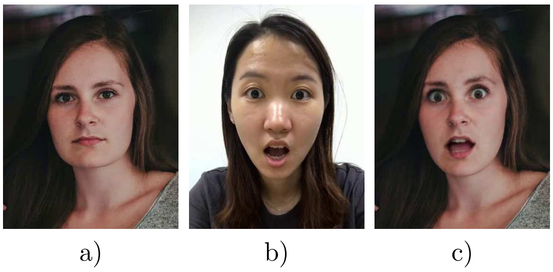


Figure 2-14: Example of expression transfer using Warp-Guided GAN [Geng et al., 2018]. The expression is transferred from an exemplar image, b), into an input image, a), generating an edited image, c). The approach does a rough warp of the input based on the exemplar and then uses two deep neural networks. One to refine the result after warping, and second to hallucinate previously occluded content in the mouth area.[10]

ture, which is in line with the research discussed in Section 2.4.2. FaceShop [Portenier et al., 2018] relaxes the assumption of access to precise contours, by employing less accurate sketches for training, which leads to increased robustness during inference. Contour2im [Dekel et al., 2018] employs denser contours, which also include additional colour gradient information.

The aforementioned methods require paired data, where the distance between the synthesized and the ground truth images is measured with an L1 pixel loss. An adversarial loss is used to improve the realism in TalkingHeads, Contour2im and FaceShop, while MakeAFace relies on a perceptual loss [Gatys et al., 2015]. Editing based on sketches or contours significantly simplifies the data collection process, as semantic labels are not required, and contours and sketches can be obtained automatically. However, these type of manipulations are more cumbersome than editing via semantic knobs.

Another line of work is based on the 3D Morphable model (3DMM) [Blanz and Vetter, 1999], which was discussed at the beginning of this section. An extensive review of these methods is provided by Zollhöfer *et al*. [Zollhöfer et al., 2018], and we only discuss here the most relevant techniques. Fitting a 3DMM is a highly non-linear problem, where deep learning techniques can be used to bypass this issue by directly predicting the parameters of the blendshape model [Richardson et al., 2016, Tuan Tran et al., 2017, Tewari et al., 2017]. These methods are commonly trained without ground truth data for the 3DMM parameters, as this data is expensive to acquire. Some techniques employ synthetic data [Richardson et al., 2016], which leads to reduced performance on real images. If several images of each subject are available, an approximation of the ground truth data can be obtained by estimating the 3DMM parameters for each image

[10] Image courtesy of [Geng et al., 2018].

and averaging the values per subject. Neural networks trained on this data have been shown to achieve competitive levels of accuracy [Tuan Tran et al., 2017]. A different approach [Tewari et al., 2017] relies on placing reconstruction losses directly on the rendered images, while employing additional regularisation losses to make the problem tractable. However, training directly with images requires employing approximations in the rendering pipeline, as the renderer must be differentiable in order to backpropagate the gradients to the neural network.

An alternative approach that overcomes the issues with differentiable renders has been explored. The key idea is to employ a network to fix the artefacts present in the image edited by the 3DMM. This has been used both for video re-enactment [Kim et al., 2018, Geng et al., 2018] and for animating 3D avatars in real time [Nagano et al., 2018]. Similarly to previously discussed approaches, the networks employed in these methods are variations of Pix2Pix.

Particularly relevant to the work presented in this thesis is Warp-Guided GAN [Geng et al., 2018]. An example of an edit from this method is shown in Fig. 2-14. This model is an extension of the method of [Averbuch-Elor et al., 2017], which was discussed above. Recall that this method aligns the faces, warps the input face based on landmarks, transfers fine details via ratio images and transfers the mouth interior with Poisson blending. In Warp-Guided GAN a 3DMM model is fitted to the video, which leads to better warps. The ratio images are substituted by a Pix2Pix network, which adds the high and medium frequency details that could not be added by warping alone. A second Pix2Pix network is used to inpaint the mouth area, which is an improvement over directly transferring the mouth from the example face to input face image using Poisson blending. An important limitation of this approach is the requirement of paired data for training, which is shared with all previous methods employing Pix2Pix networks.

Alternative approaches that do not use GAN networks have also been proposed. For example, Facelet-Bank [Chen et al., 2018b] edits images with an autoencoder network. First, the network is trained by performing image reconstructions with an L1 pixel loss and a feature loss. The feature loss encourages the encoding of the input image to be similar to the encoding of the output image. Then, the decoder is fixed, and an additional network learns how to modify the decoder features, so that rather than reconstructions, the decoder outputs an edited image. The features used to train the additional network are constructed by averaging the features of neighbouring images with the target attribute.

As discussed in the previous chapter, a significant limitation of deep learning methods is that they are constrained to operate at the same (or similar) resolution of the training images. Some deep learning methods have been proposed that are capable of operating at arbitrary resolutions [Yeh et al., 2016, Gharbi et al., 2017]. However, these techniques require paired data, while many of the aforementioned approaches employing adversarial or perceptual losses do not. In contrast to these approaches, a method that is able to edit images at arbitrary resolutions, that can be trained with unpaired data, and that does not require example images to describe the target edit will be presented in Chapter 4.

Chapter 3

Structured uncertainty

3.1 Introduction

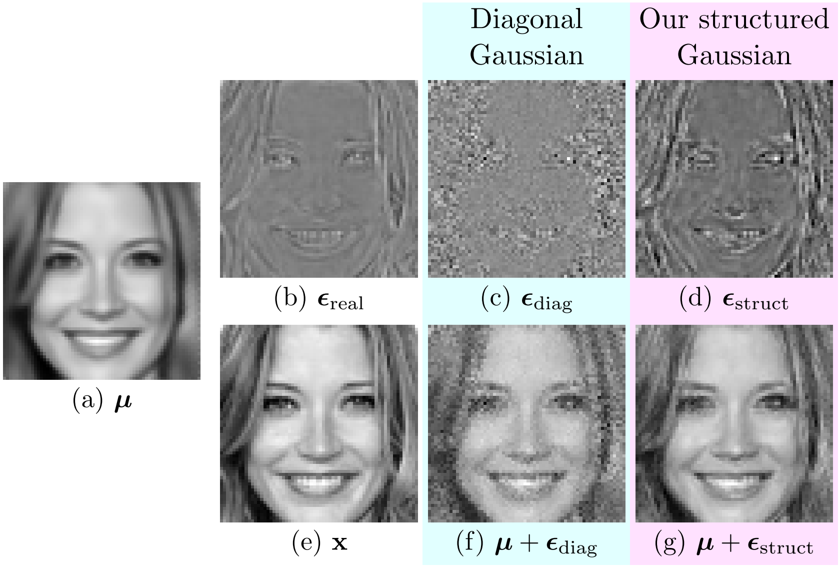
In Chapter 1, the Variational Autoencoder (VAE) [Rezende et al., 2014, Kingma and Welling, 2014] model was described as a promising method for image editing. However, as discussed in Section 2.4, this method is known to generate blurry samples.

In this chapter we describe a method that allows VAE models to generate images that contain high frequency details. Our method operates on Multivariate Gaussian distributions, as it predicts structured uncertainty distributions over residual images. A naïve approach to predict full covariance matrices would be computationally intractable, yet we employ covariance matrices with sparse inverses, which are a tractable subset of these distributions. Moreover, the method can be applied to any model that is trained for reconstructions and that outputs the mean of a Gaussian distribution.

3.1.1 Motivation

Deep probabilistic generative models have recently become the most popular tool to synthesise novel data. Particularly of interest for our applications is their use for image editing. Generative models require solving a density estimation problem. We concentrate on latent variable models where an explicit representation of the data density is given by the model, such as the VAE.

For latent variable models over continuous data, it is common to use a factorised multivariate Gaussian distribution to model the data likelihood conditioned on the latent



(a) $\boldsymbol{\mu}$ (b) $\boldsymbol{\epsilon}_{\text{real}}$ (c) $\boldsymbol{\epsilon}_{\text{diag}}$ (d) $\boldsymbol{\epsilon}_{\text{struct}}$ (e) $\mathbf{x}$ (f) $\boldsymbol{\mu}+\boldsymbol{\epsilon}_{\text{diag}}$ (g) $\boldsymbol{\mu}+\boldsymbol{\epsilon}_{\text{struct}}$

Figure 3-1: A VAE with a diagonal Gaussian likelihood learns the mean of the distribution, $\boldsymbol{\mu}$, as a smooth reconstruction of an input image, $\mathbf{x}$. The residual for this reconstruction is $\boldsymbol{\epsilon}_{\text{real}} = \mathbf{x} - \boldsymbol{\mu}$. The diagonal Gaussian only models unstructured residuals, where $\boldsymbol{\epsilon}_{\text{diag}}$ is a residual sample. When $\boldsymbol{\epsilon}_{\text{diag}}$ is added to $\boldsymbol{\mu}$ it generates an unrealistic image (f), demonstrating a failure to capture the real residual structure, $\boldsymbol{\epsilon}_{\text{real}}$. In contrast, we learn a structured residual model, with residual samples like $\boldsymbol{\epsilon}_{\text{struct}}$ that, when added to $\boldsymbol{\mu}$, generate a plausible and realistic image (g).

variable. This can be stated as $p(\mathbf{x} \,|\, \mathbf{z}) = \mathcal{N}\big(\boldsymbol{\mu}, \boldsymbol{\Sigma}\big)$, where $\mathbf{x}$ is an image flattened as a column vector, $\mathbf{z}$ is a latent variable, and the parameters of the Gaussian distribution are a mean vector, $\boldsymbol{\mu}$, and a covariance matrix, $\boldsymbol{\Sigma}$. This corresponds to the forward model $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a residual image, which is usually interpreted as noise in the data, and it is distributed as $\boldsymbol{\epsilon} \sim \mathcal{N}\big(\mathbf{0}, \boldsymbol{\Sigma}\big)$.

In factorised multivariate Gaussian distributions, the factorisation occurs in the covariance matrix, $\boldsymbol{\Sigma}$, which rather than a full matrix, is modelled either as a diagonal or scaled identity matrix, and may also be assumed to be shared for the whole dataset. The most extreme factorisation is that of the mean squared error, which assumes the errors at all pixels are i.i.d. (independent and identically distributed). Formally, using a mean squared error loss is equivalent to setting $\boldsymbol{\Sigma} = \mathbf{I}$. Diagonal covariance models [Kingma and Welling, 2014, Burda et al., 2016] are an improvement, as they allow some local estimation of noise level but maintain the strong and flawed assumption that pixel values in the residual are uncorrelated, as shown in Fig. 3-1b, 3-1c and 3-1d. These common choices of likelihood imply that if one were to draw a sample from the model ($\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$), which would include the additive noise term, $\boldsymbol{\epsilon}$, as opposed to just the mean, $\boldsymbol{\mu}$, white noise would be added to the generated image, $\mathbf{x}$, which is unlikely to ever appear realistic, as shown in Fig. 3-1f. For these reasons, most researchers only show the mean of the distribution, an overly-smooth $\boldsymbol{\mu}$, when employing these models [Larsen et al., 2016, Yan et al., 2016]. This emphasises the incoherence in generative models caused by these simplifications, and this is addressed in this chapter.

3.1.2 Proposed solution

We postulate that the residuals are highly structured and reflect limitations in model capacity – we therefore propose to estimate the conditional data likelihood, $p(\mathbf{x}|\mathbf{z})$, using a multivariate Gaussian distribution with a full covariance matrix, to capture pixel-wise correlations which will in turn improve the sampled reconstructions, as shown in Fig. 3-1d and 3-1g. In more detail, a deep neural network is used to predict full covariance matrices to model the residual distributions.

Providing the VAE with a structured noise model endows it with the capability of modelling complex, high-frequency features, which do not reliably occur in the same location (e.g. hair), stochastically rather than deterministically. In a VAE this approach implies learning to predict the parameters of a multivariate Gaussian distribution to explain a single image sample. This is a poorly conditioned problem, which requires informative priors to avoid model overfitting and other poor solutions.

A few techniques are explored in this chapter; however they were found to either restrict the space of solutions excessively or to be insufficient to be able to train the model reliably. Interestingly, we found two alternative approaches that work well in practice. The first method consists of training a VAE with a factorised Gaussian likelihood, and learning a separate network to model the residual distribution of the trained VAE. The second approach employs two regularisation losses as an approximation to an informative prior on the multivariate Gaussian distributions.

A naïve approach to training a VAE with a structured Gaussian likelihood would involve predicting a covariance matrix with $((3n)^2 + 3n)/2$ unique parameters and a mean with $3n$ unique parameters, where n is the number of pixels in an RGB image. This is computationally infeasible with standard strategies for training deep neural networks, and it would require very informative priors, as there is only one residual sample to learn the distribution from. Instead, we propose to operate on the restricted set of multivariate Gaussian distributions where the covariance matrices have sparse inverses. The inverse of a covariance matrix is known as the precision matrix, and our proposed approach models these precision matrices, rather than operating with covariance ones.

Sparse precision matrices allow tractable evaluations of the likelihood (needed for training) and provide tractable methods for sampling (needed for inference). Moreover, the proposed sparsity pattern is based on how close pixels are in image space, and this sparse approximation has a complexity that grows linearly with the number of pixels. A key aspect is that the covariance matrix is *not* sparse, even though its inverse matrix is sparse. Thus, it can model long range correlations, *e.g.* strands of hairs, as long as sufficient correlations are induced by the local structure in the precision matrix.

We demonstrate that a network to estimate these matrices can be tractably learned without ground truth data for the matrices, by employing maximum likelihood estimation on the residual images. Moreover, the model can predict covariance matrices that generalise well to previously unseen examples.

A key advantage of our approach for modelling structured uncertainty with a separate network, is that it can be readily applied to any black box method that outputs the mean of a Gaussian distribution; for example, to methods trained with mean squared error. Furthermore, the results could in principle be extended to other approaches in future work, such as implicit likelihood methods [Goodfellow et al., 2014], or methods that measure the data likelihood in a feature space [Hou et al., 2017].

3.2 Previous work

Our method builds upon the Variational Autoencoder (VAE) model, which was described in Section 2.4. Several approaches to regularise the approximate posterior distribution, as well as increase the complexity of the approximate posterior and the likelihood distributions were discussed. Most relevant to the content of this chapter are the methods that seek to employ more complex likelihood distributions in the model, which were presented in Section 2.4.3.

Those methods either model the likelihood as an autoregressive distribution [Gulrajani et al., 2017b], or transform the VAE into an implicit likelihood method, by adding terms that do not fit within an explicit likelihood generative model [Larsen et al., 2016, Hou et al., 2017, Huang et al., 2018]. The former requires as many forward passes of the network as the number of pixels in the image, which makes the method too slow for image editing applications. The latter are less principled approaches that involve hyper-parameter tuning for the added terms, that lack the means to evaluate the likelihood in pixel space on a test set for model comparison, and whose reconstructions do not resemble the inputs well. Thus, also rendering these methods inadequate for image editing.

The focus of this chapter is on statistical quantification of uncertainty. Therefore, we devote the remainder of this section to related methods for structured uncertainty prediction.

3.2.1 Structured uncertainty prediction

Statistical quantification of uncertainty has been an area of interest for a long time. Many traditional statistical estimation models provide some measure of uncertainty on the inferred parameters of a fitted model [Tipping and Bishop, 1999, Bartholomew et al., 2011]. However, as discussed in Section 2.2 quantification of parameter uncertainty is, in general, not tractable for deep learning generative methods.

Instead, we focus on uncertainty on the predictions made by the model, *i.e.* how confident is the model in its predictions. This uncertainty is usually dominated by two sources of noise. First, noise inherent in the data, which is usually introduced when acquiring the data. Second, noise due to deficiencies in the model, such as limited number of parameters or optimisers that are unable to reach the global optima during the learning phase.

This uncertainty on the model outputs is often modelled as a factorised multivariate Gaussian distribution. A common issue with these models is the false assumption of independence between pixels in the reconstruction residual image. In standard methods only a few samples of the residual are available (usually only one), which limits the ability to reliably estimate more complex uncertainty models. Previous work on modelling correlated Gaussian noise is limited, it has been used for small data scenarios [Nikias and Pan, 1988], for temporally correlated noise models [Woolrich et al., 2001] and in Gaussian processes [Rasmussen and Williams, 2006].

The most recent related work on uncertainty prediction for deep models has been the prediction of heteroscedastic Gaussian noise for encoder/decoder models [Kendall and Gal, 2017]. This approach is similar to the variational autoencoder with a Gaussian likelihood with a diagonal covariance [Kingma and Welling, 2014], but can be applied when the input and the generated output are different, for example in semantic segmentation tasks, as shown in Fig. 3-2. The authors claim that the predicted variance

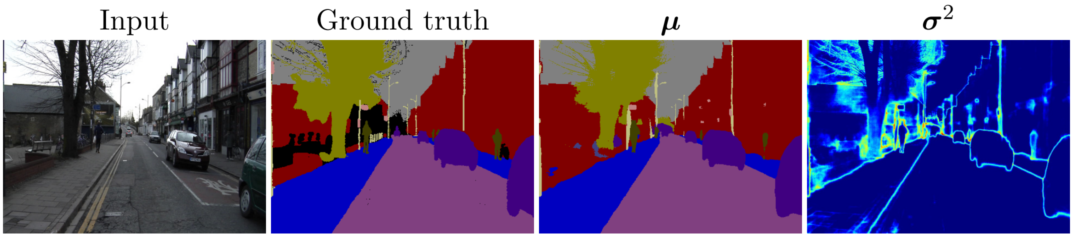


Figure 3-2: Uncertainty prediction in encoder-decoder models for semantic segmentation [Kendall and Gal, 2017]. The method is able to predict independent per-pixel uncertainty, $\boldsymbol{\sigma}^2$, on the predicted segmentation maps, $\boldsymbol{\mu}$.[1]

maps, denoted as $\boldsymbol{\sigma}^2$ in the figure, are indicative of regions with high acquisition noise. However, these maps correspond well to high frequency content in the image, similarly to the VAE with a diagonal noise model (see Fig. 3-25). Thus, indicating regions that the model consistently struggles to accurately predict. Hence, areas which should be modelled as having low variance and high covariance are being modelled as regions with high variance, as their approach is by construction unable to model covariance. This is further encouragement for the work presented in this chapter, which is able to cope with these regions, as will be shown in Fig. 3-25 for a reconstruction task.

3.3 Methodology

In this section we introduce an extension of the VAE model, which is trained with a structured Gaussian likelihood. We also discuss a restricted standalone structured uncertainty prediction model that can be applied to models with a Gaussian likelihood. This section builds on the VAE model, and we use the notation introduced in Section 2.4.

For easier reading, the relevant equations are repeated below. The variational approximation, equation 2.9, used in the model was defined as

$$L_{\text{VAE}} = \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})} \left[\log \; p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}) \right] - D_{\text{KL}} \left[q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z}) \right], \tag{3.1}$$

where $\mathbf{x}$ is a column vector containing a vectorised image with n pixels, and $\mathbf{z}$ is the latent vector. The data likelihood, equation 2.12, is that of a multivariate Gaussian:

$$p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}) = \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\sigma}(\mathbf{z})^2 \, \mathbf{I} \big). \tag{3.2}$$

[1] Images courtesy of [Kendall and Gal, 2017].

The aforementioned likelihood corresponds to the forward model:

$$\mathbf{x} = \boldsymbol{\mu}(\mathbf{z}) + \boldsymbol{\epsilon}(\mathbf{z}), \tag{3.3}$$

where $\boldsymbol{\epsilon}(\mathbf{z})$ is known as the residual, and it is defined as

$$\boldsymbol{\epsilon}(\mathbf{z}) \sim \mathcal{N}\big(\,\mathbf{0}, \boldsymbol{\sigma}(\mathbf{z})^2\mathbf{I}\,\big). \tag{3.4}$$

As previously stated, most multivariate Gaussian models employ a factorised likelihood, equation 3.2, which assumes that the pixels in the residual, $\boldsymbol{\epsilon}(\mathbf{z})$, are independently distributed. In contrast, we extend the noise model to use a multivariate Gaussian likelihood with a full covariance matrix

$$p_{\boldsymbol{\theta}}(\mathbf{x}\,|\,\mathbf{z}) = \mathcal{N}\big(\,\boldsymbol{\mu}(\mathbf{z}), \boldsymbol{\Sigma}(\mathbf{z})\,\big), \tag{3.5}$$

where $\boldsymbol{\mu}(\mathbf{z})$ and $\boldsymbol{\Sigma}(\mathbf{z})$ are a parametric (non-linear) functions; this is equivalent to $\boldsymbol{\epsilon}(\mathbf{z}) \sim \mathcal{N}\big(\,\mathbf{0}, \boldsymbol{\Sigma}(\mathbf{z})\,\big)$. The covariance matrix captures the correlations between pixels to allow sampling of structured residuals as demonstrated in our experiments.

In our method, the covariance captures structured information about reconstruction uncertainty of the generative model. This means that drawing a sample from $\mathcal{N}\big(\,\mathbf{0}, \boldsymbol{\Sigma}(\mathbf{z})\,\big)$ and adding that to the mean, $\boldsymbol{\mu}(\mathbf{z})$, may produce a result that is more representative of the target image, as shown in Fig. 3-1. However, any given sample will not necessarily have a lower reconstruction error under a mean squared error loss. For example, in the case of images of human faces, the covariance might capture high-frequency details, that cannot be exactly modelled by the VAE, like hair details. Hair details sampled in this way will not necessarily align with the input hair, yet they make the image look more realistic.

A maximum likelihood approach is used to train the model. The aforementioned modifications only affect the likelihood term in equation 3.1, leaving the KL term unchanged. When using a full covariance matrix, the likelihood term is expanded as

$$\log\ p_{\boldsymbol{\theta}}(\mathbf{x}\,|\,\mathbf{z}) = -\frac{1}{2}\Big(\log\big(\big|\boldsymbol{\Sigma}(\mathbf{z})\big|\big)\ + \big(\mathbf{x} - \boldsymbol{\mu}(\mathbf{z})\big)^{\mathsf{T}}\big(\boldsymbol{\Sigma}(\mathbf{z})\big)^{-1}\big(\mathbf{x} - \boldsymbol{\mu}(\mathbf{z})\big) + c\Big)\,, \tag{3.6}$$

where $c = 3n\log(2\pi)$ is a constant term that does not affect the optimisation.

We will assume the input images are grey-scale, as this simplifies the discussion. In Section 3.3.4 we will show that our grey-scale derivation can be easily adapted to colour images. To simplify notation, in subsequent sections $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$ and $\boldsymbol{\epsilon}$ are used to denote

$\mathbf{\Sigma}(\mathbf{z})$, $\boldsymbol{\mu}(\mathbf{z})$ and $\boldsymbol{\epsilon}(\mathbf{z})$ respectively.

3.3.1 Covariance estimation considerations

The decoder network is used to estimate the covariance matrix, $\mathbf{\Sigma}$, and the means, $\boldsymbol{\mu}$, from the latent vector $\mathbf{z}$. The part of the decoder network responsible for estimating the covariance matrix will henceforth be referred to as the covariance network. This learning task is challenging on two fronts. First, the task is computationally challenging, with a complexity of $\mathcal{O}(n^2)$, as the covariance matrix contains $(n^2 + n)/2$ unique elements, where n is the number of pixels in the image. Second, it is ill-posed as no ground truth covariance matrices are available for real data. Moreover, for each training example, a full covariance matrix must be estimated from a single residual image, which increases the probabilities of overfitting.

For any covariance estimation method there are four aspects to consider:

(i) how difficult is it to sample from the covariance matrix?

(ii) how difficult is it to compute the terms (or their gradients) in equation 3.6?

(iii) how difficult is it to impose symmetry and positive definiteness on the estimated matrix?

(iv) what are the memory requirements and computational costs?

The standard sampling approach for $\mathcal{N}\big(\boldsymbol{\mu}, \mathbf{\Sigma}\big)$, given a decomposition of the covariance matrix in the form $\mathbf{\Sigma} = \mathbf{M}\mathbf{M}^{\mathrm{T}}$, is defined as $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = \mathbf{M}\mathbf{u}$, and $\mathbf{u} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{I}\big)$ is a vector of standard Gaussian samples. In this decomposition $\mathbf{M}$ is constrained to contain only real values. Note that this decomposition of $\mathbf{\Sigma}$ is not, in general, unique. Cholesky decompositions and eigendecompositions are popular choices for numerically estimating $\mathbf{M}$, and they will be explained in the subsequent section.

By definition, the covariance matrix is symmetric and positive definite.[2] A matrix is symmetric if $\sigma_{i,j} = \sigma_{j,i}$, where $\sigma_{i,j}$ is a single element in $\mathbf{\Sigma}$, and the condition must be fulfilled for all possible i and j pairs. Positive definiteness is defined as $\mathbf{x}^{\mathrm{T}}\mathbf{\Sigma}\mathbf{x} > 0$, for any $\mathbf{x}$ that contains real numbers.

[2] Technically the matrix may be positive semi-definite $\mathbf{x}^{\mathrm{T}}\mathbf{\Sigma}\mathbf{x} \geq 0$. Contrary to the positive definite case, this allows for $|\mathbf{\Sigma}| = 0$. If this occurs, the probability density function is degenerate, *i.e.* the density of $\mathbf{x}$ is not defined. Usually, this arises when $\mathbf{x}$ and $\mathbf{\Sigma}$ live on a lower dimensional manifold, and in that manifold the matrix is positive definite, *i.e.* the density of $\mathbf{x}$ is defined if it is evaluated on the manifold. In order to avoid this degenerate state, only positive definite matrices are modelled.

If $\mathbf{\Sigma}$ is the direct output of the covariance network, it needs to be inverted to calculate the negative log-likelihood in equation 3.6. Hence, it is more practical to estimate the precision matrix $\mathbf{\Lambda} = \mathbf{\Sigma}^{-1}$ as this term appears directly in the log likelihood, and the log determinant term can be equivalently computed as $\log(|\mathbf{\Sigma}|) = -\log(|\mathbf{\Lambda}|)$. The inverse of a symmetric positive definite matrix is also symmetric and positive definite. Therefore, it is enough to enforce these constraints on $\mathbf{\Lambda}$ to ensure that they appear in $\mathbf{\Sigma}$. Since sampling still requires a decomposition of the covariance matrix, this approach will be in general less efficient in that regard. However, a number of techniques will be presented to mitigate this issue.

3.3.2 Precision matrix parametrisations

In this section we discuss previously used parametrisations for the precision matrix, including the **eigendecomposition** and the **Cholesky decomposition**. Multivariate Gaussian distributions are widely used, and a number of parametrisations for the precision (or covariance) matrix have been previously explored, including the aforementioned ones. When directly predicting the precision (or covariance) matrix, imposing positive definiteness, and evaluating the log determinant, can be computationally demanding tasks, and these parametrisations may be used to alleviate these issues.

For each parametrisation of the precision matrix, we discuss the four aspects defined in the previous section, namely: (i) sampling, (ii) log likelihood evaluation, (iii) symmetry and positive definiteness, and (iv) memory and computational costs.

To simplify notation, we will use $\mathbf{r}$ to refer to the residual images, $\mathbf{r} = \mathbf{x} - \boldsymbol{\mu}$. Thus, the squared error term in equation 3.6 can be expressed in a more compact form as

$$(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \mathbf{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = \mathbf{r}^{\mathrm{T}} \mathbf{\Lambda} \mathbf{r}. \tag{3.7}$$

Eigendecomposition

An eigendecomposition is defined as $\mathbf{\Lambda} = \mathbf{Q}\mathbf{V}\mathbf{Q}^T$, where $\mathbf{Q}$ is an orthogonal matrix, where each column is denoted as an eigenvector, and $\mathbf{V}$ is a diagonal matrix with strictly positive elements in the diagonal, which are referred to as eigenvalues. The network only needs to output the eigenvector matrix $\mathbf{Q}$, and the diagonal elements of the eigenvalue matrix $\mathbf{V}$.

(i) The matrix $\mathbf{M}$ needed for sampling can be computed as:

$$\mathbf{M} = \mathbf{Q}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}^{T}. \tag{3.8}$$

As $\mathbf{V}$ is diagonal, $\mathbf{V}^{-\frac{1}{2}}$ only involves evaluating the inverse of the square root of each element in the diagonal.

(ii) The log likelihood evaluation is also simple to compute, the log determinant is defined as

$$\log(|\mathbf{\Sigma}|) = -\sum_{i=0}^{n-1} \log(v_{ii}), \tag{3.9}$$

where v_{ii} is the i^{th} diagonal element in $\mathbf{V}$. The squared error term is defined as

$$\mathbf{r}^{\intercal}\mathbf{\Lambda}\mathbf{r} = \mathbf{w}^{\intercal}\mathbf{w}, \tag{3.10}$$

where

$$\mathbf{w} = \mathbf{V}^{\frac{1}{2}}\mathbf{Q}^{T}\mathbf{r}, \tag{3.11}$$

where this formulation avoids evaluating the $\mathbf{Q}\mathbf{V}\mathbf{Q}^{T}$ matrix multiplications needed to construct $\mathbf{\Lambda}$.

(iii) As stated above, $\mathbf{V}$ must be positive and $\mathbf{Q}$ must be an orthogonal matrix to guarantee that $\mathbf{\Lambda}$ is symmetric and positive definite. The former can be easily achieved by having the network predict the $\log(\mathbf{V})$, and the later with an additional orthogonality loss:

$$L_{\text{ortho}} = \|\mathbf{Q}^{T}\mathbf{Q} - \mathbf{I}\|^{2}. \tag{3.12}$$

(iv) The computational cost and memory constraints can be reduced at the cost of employing an approximated precision matrix. This can be modelled by only predicting the n_{v} largest eigenvalues and their corresponding eigenvectors. If that is the case, equation 3.11 can be evaluated from right to left using the reduced size matrices, without explicitly building $\mathbf{\Lambda}$, which significantly increases the efficiency of the operation.

A significant limitation of this approach with a reduced number of eigenvectors is that $\mathbf{\Lambda}$ is not invertible by construction, as the smallest eigenvalues are set to zero, and this breaches the constraint that the precision matrix must be positive definite. Moreover, the number of eigenvectors that are required to have a good model of $\mathbf{\Lambda}$ is in general too large to be practical. After initial investigations, we empirically found these limitations to be too restrictive.

Eigendecomposition with diagonal

The eigendecomposition can be extended by adding a diagonal term $\mathbf{\Lambda} = \mathbf{QVQ}^T + a\mathbf{I}$, where a is a positive scalar. In contrast to the previous eigendecomposition approximation, this ensures that the estimated matrices are invertible. For this method, the additional parameter, a, can be estimated by the network or set as a hyper-parameter.

(i) Sampling can be achieved by solving the following system of equations

$$\mathbf{L}^{\mathrm{T}}\boldsymbol{\epsilon} = \mathbf{u}, \tag{3.13}$$

for $\boldsymbol{\epsilon}$, where $\mathbf{u} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{I}\big)$ and $\mathbf{L}$ is a numerically estimated Cholesky decomposition $\mathbf{LL}^T = \mathbf{QVQ}^{\mathrm{T}} + a\mathbf{I}$, which will be defined in the subsequent section. The proof of how sampling is equivalent to solving this system is given in Section A.1.5.

(ii) The log determinant is computed as

$$\log(|\mathbf{\Sigma}|) = -2\sum_{i=0}^{n-1}\log(\ell_{ii}), \tag{3.14}$$

where ℓ_{ii} is the i^{th} diagonal element in $\mathbf{L}$, and the squared error is computed as indicated in equation 3.7.

(iii) The same positive definite constraints described in the previous section can be employed, as the sum of positive definite matrices is positive definite, and the inverse is also positive definite.

(iv) This approach requires significant computational and memory resources. Contrary to the previous approach, the matrix $\mathbf{\Lambda}$ must be explicitly constructed for the squared error term. Additionally, a costly Cholesky decomposition must be numerically estimated for each evaluation of the likelihood.

In practice, this method suffers from similar limitations as the previous one, where the number of eigenvectors that are required to have a good model of $\mathbf{\Lambda}$ is in general too large to be practical.

Cholesky decomposition

A Cholesky decomposition is defined as: $\mathbf{\Lambda} = \mathbf{L}\mathbf{L}^{\intercal}$, where $\mathbf{L}$ is a lower triangular matrix, and the covariance network only explicitly estimates the non-zero elements in $\mathbf{L}$.[3] It will be shown below that all the quantities of interest can be evaluated directly from $\mathbf{L}$, *i.e.* without explicitly computing $\mathbf{\Lambda}$ or $\mathbf{\Sigma}$.

(i) Sampling from $\mathbf{\Sigma}$ involves solving the triangular system of equations

$$\mathbf{L}^{\intercal}\boldsymbol{\epsilon} = \mathbf{u}, \tag{3.15}$$

for $\boldsymbol{\epsilon}$ with backwards substitution, which requires $O(n^2)$ operations. Proof of how sampling is equivalent to solving this system is given in Section A.1.5.

(ii) Using the Cholesky decomposition, the terms in the negative log likelihood are simple to compute. The log determinant is

$$\log(|\mathbf{\Sigma}|) = -2\sum_{i=0}^{n-1}\log(\ell_{ii}), \tag{3.16}$$

where ℓ_{ii} is the i^{th} element in the diagonal of $\mathbf{L}$. The reconstruction error is defined as

$$\mathbf{r}^{\intercal}\mathbf{\Lambda}\mathbf{r} = \mathbf{w}^{\intercal}\mathbf{w}, \tag{3.17}$$

where

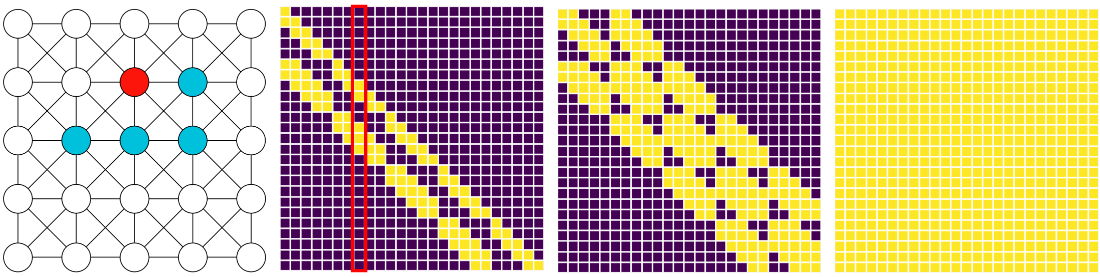
$$\mathbf{w} = \mathbf{L}^{\intercal}\mathbf{r}, \tag{3.18}$$

where this formulation avoids evaluating the $\mathbf{L}\mathbf{L}^{\intercal}$ matrix multiplication needed to construct $\mathbf{\Lambda}$.

(iii) By construction, the estimated precision matrix $\mathbf{\Lambda}$ is symmetric. To ensure that it is also positive definite it is sufficient to constrain the diagonal entries of $\mathbf{L}$ to be strictly positive, *e.g.* by having the network estimate $\log(\ell_{ii})$ element-wise.

(iv) Estimating this matrix directly is only a feasible solution for datasets with small dimensionality, as the number of parameters to be estimated increases quadratically with the number of pixels in $\mathbf{x}$.

[3] For the sake of completeness, we include a discussion of employing a Cholesky parametrisation on the covariance matrix, $\mathbf{\Sigma} = \mathbf{M}\mathbf{M}^{\intercal}$, in Section A.1.2 in the appendix. In this case, the network estimates the non-zero elements in the lower triangular matrix $\mathbf{M}$.



(a) Neighbourhood

(b) $\mathbf{L}$

(c) $\mathbf{\Lambda} = \mathbf{L}\mathbf{L}^{\intercal}$

(d) $\mathbf{\Sigma} = \mathbf{\Lambda}^{-1}$

Figure 3-3: Example of the sparsity patterns in the sparse Cholesky decomposition parametrisation for a 5×5 image. A spatial connectivity of $n_\mathrm{f} = 3$ is illustrated in (a), where each circle denotes a pixel in the image and lines connecting them indicate non-zero values between the pixel pair in $\mathbf{L}$. This leads to a band-diagonal lower-triangular Cholesky matrix $\mathbf{L}$, as shown in (b). The band-diagonal precision matrix, $\mathbf{\Lambda}$, is shown in (c) and the resulting dense covariance matrix, $\mathbf{\Sigma}$, in (d). The highlighted column in (b) corresponds to the red pixel in (a). The non-zero elements in this column correspond to the red pixel itself and its right and bottom neighbours highlighted in light blue. Dark blue is used in (b), (c) and (d) to denote values that are zero by construction, while yellow indicates the opposite.

3.3.3 Sparse Cholesky Decomposition

In this section we present our novel parametrisation for the precision matrix. Our approach is based on the Cholesky decomposition that was introduced in the previous section. We propose to reduce the computational complexity by imposing a fixed sparsity pattern in the matrix $\mathbf{L}$, and only estimate the non-zero values of the matrix via the covariance network. This allows us to scale our method to larger resolution images. Our fixed sparsity pattern only affects the lower-triangular part of the matrix, as the upper-triangular part is already sparse, as per definition of the Cholesky decomposition.

A key aspect to consider is that, in general, a sparse matrix $\mathbf{\Lambda}$, has a dense inverse matrix, $\mathbf{\Sigma} = \mathbf{\Lambda}^{-1}$. In practice, this means that having a sparse precision does not prevent strong correlations between distant pixels. However, this does not apply to the covariance matrix, as directly imposing sparsity in $\mathbf{\Sigma}$ removes the possibility of correlations between any pair of pixels that has a zero weight in $\mathbf{\Sigma}$.

The sparsity pattern imposed should depend on the type of data being modelled. For image data, we propose that elements in $\mathbf{L}$, denoted as ℓ_{ij}, are only non-zero if $i \geq j$ and i and j are neighbours in the image plane, where pixels i and j are neighbours if a patch of size n_f centred at i contains j, as shown in Fig. 3-3. As the patch is centred on the pixel, n_f is constrained to be an odd positive integer. This reduces the

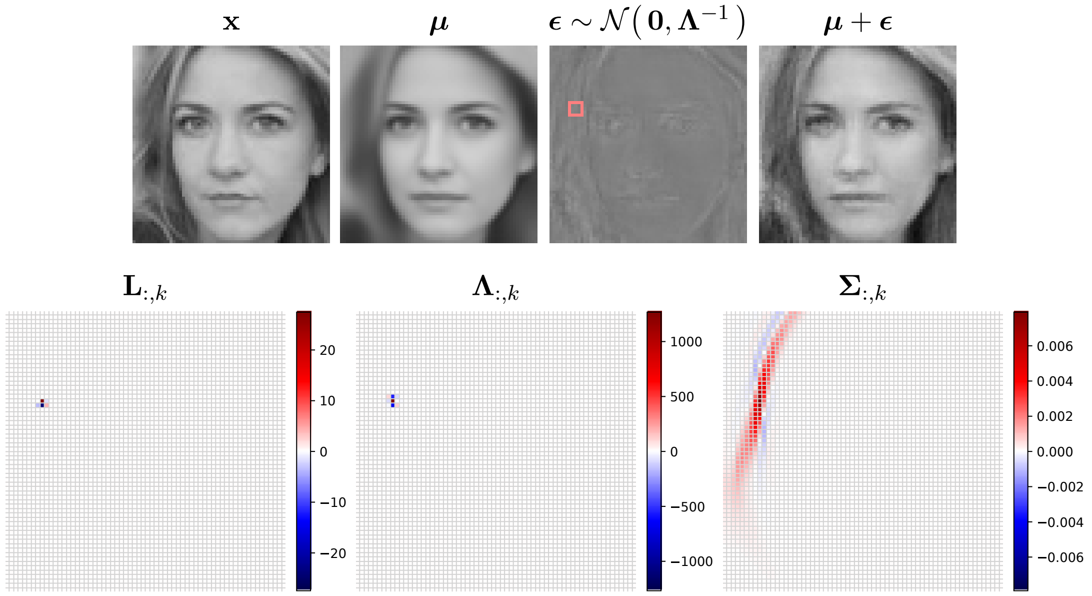


Figure 3-4: Demonstration of long correlations being modelled by a sparse precision matrix, $\mathbf{\Lambda}$. For an input image, $\mathbf{x}$, a VAE learns a mean, $\boldsymbol{\mu}$, and the sparse Cholesky factor, $\mathbf{L}$, of a precision matrix $\mathbf{\Lambda}$. A neighbourhood with size $n_\mathrm{f} = 3$ is highlighted by a red square in $\boldsymbol{\epsilon}$ for a single pixel, k. The reshaped column corresponding to pixel k is shown for each matrix. In $\mathbf{L}_{:,k}$ and $\mathbf{\Lambda}_{:,k}$ the zero values are so by construction, while in $\mathbf{\Sigma}_{:,k}$ they are not. Long range correlations between pixels, for example the long strands of hair, can be modelled by the covariance matrix, $\mathbf{\Sigma} = \mathbf{\Lambda}^{-1}$, which is not *necessarily* sparse.

number of non-zero elements in the matrix $\mathbf{L}$ to $n((n_\mathrm{f}^2 + 1)/2)$, where $n_\mathrm{f} \ll n$ and $n_\mathrm{f} > 1$.[4] Per pixel, a maximum of $(n_\mathrm{f}^2 + 1)/2$ values are estimated. For example, employing a neighbourhood of size 3, implies a maximum of 5 values per pixel in $\mathbf{L}$, as shown in Fig. 3-3a and 3-3b. Note that patch size and neighbourhood size will be used interchangeably to refer to n_f.

The capability of the covariance matrix to model strong correlations between distant pixels, despite the sparsity imposed in the precision, is shown in Fig. 3-4. In this example, a neighbourhood of size 3 is used. For the pixel of interest, many values in $\mathbf{\Sigma}$ are not zero, and those with strong correlations are well beyond the neighbourhood size. This allows modelling long range image structure, like hair.

The matrix $\mathbf{L}$ resulting from this sparsity pattern is both lower-triangular and band-diagonal as shown in Fig. 3-3. This leads to a precision matrix $\mathbf{\Lambda}$ with a similar sparsity

[4] The number of non-zero elements in $\mathbf{L}$ is further reduced, as pixels near the edge of the image have fewer neighbours.

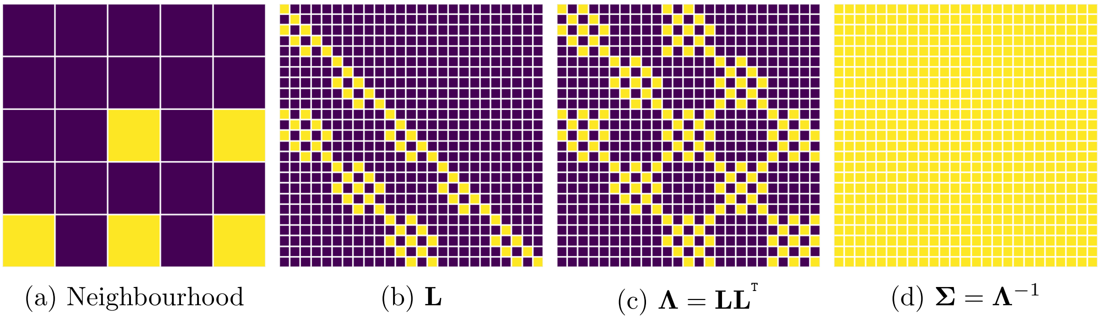

(a) Neighbourhood (b) $\mathbf{L}$ (c) $\boldsymbol{\Lambda} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$ (d) $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{-1}$

Figure 3-5: Example of the dilated sparsity patterns for the sparse Cholesky decomposition parametrisation. Similar to Fig 3-3. To avoid clutter, the neighbourhood in a) is shown by reshaping a single column of $\mathbf{L}$, where the element in the diagonal of $\mathbf{L}$ is placed in the centre of the neighbourhood image. In this example a sparsity pattern of $n_{\mathrm{f}} = 5$ is created by dilating by a factor of 2 a 3×3 kernel.

pattern with additional bands. Sampling, log likelihood evaluation and enforcing a valid precision matrix follows the same procedure as in the dense approach, with the reduced cost from using a sparse representation. Moreover, this approach is amenable to parallelisation on the GPU, as each patch can be evaluated independently. Efficient GPU evaluation is further discussed in Section 3.3.4.

An alternative sparsity pattern can be constructed such that the neighbouring structure is similar to dilated convolutions [Yu and Koltun, 2015], as shown in Fig. 3-5. Dilated convolutions have been shown to be a good approximation to large dense filters, by stacking dilated layers with different dilation rates. In our experiments we saw only marginal gains when using the dilated sparsity pattern. However, we believe such dilated-like sparsity patterns might be useful for larger resolution images.

Connection to Gaussian Conditional Markov Random Fields

Our multivariate Gaussian model for each image can be interpreted as a Gaussian conditional Markov Random Field (G-CRF) [Murphy, 2012]. A G-CRF has a log probability defined as:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) &= \mathcal{N}\left((\boldsymbol{\Lambda}(\mathbf{z}))^{-1}\boldsymbol{\eta}(\mathbf{z}), (\boldsymbol{\Lambda}(\mathbf{z}))^{-1} \right), && (3.19)\\ &\propto (\boldsymbol{\eta}(\mathbf{z}))^{\mathsf{T}}\mathbf{x} - \frac{1}{2}\mathbf{x}^{\mathsf{T}}(\boldsymbol{\Lambda}(\mathbf{z}))\,\mathbf{x}, && (3.20)\end{aligned}$$

where $\mathbf{x}$ is a vectorised image, $\mathbf{z}$ is a latent vector, $\boldsymbol{\eta}(\mathbf{z})$ and $\boldsymbol{\Lambda}(\mathbf{z})$ are parametric functions, with learnable parameters $\boldsymbol{\theta}$. The G-CRF log probability is proportional to the log likelihood of a multivariate Gaussian distribution with mean $\boldsymbol{\mu} = (\boldsymbol{\Lambda}(\mathbf{z}))^{-1}\boldsymbol{\eta}(\mathbf{z})$, and covariance matrix $\mathbf{\Sigma} = (\mathbf{\Lambda}(\mathbf{z}))^{-1}$, as denoted in equation 3.19. Therefore, the forward model is defined as

$$\mathbf{x} = (\mathbf{\Lambda}(\mathbf{z}))^{-1}\boldsymbol{\eta}(\mathbf{z}) + \boldsymbol{\epsilon}, \tag{3.21}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}\big(\mathbf{0}, (\mathbf{\Lambda}(\mathbf{z}))^{-1}\big)$. Thus, showing how to interpret our uncertainty model as a Gaussian conditional Markov Random Field. A detailed proof of this equivalence is provided in Section A.1.1 in the appendix.

A key difference with respect to our model is that CRF models are commonly used in discriminative settings, where $\mathbf{z}$ corresponds to the input image and $\mathbf{x}$ are labels. In particular, previous work has explored Gaussian conditional Markov Random Fields for image segmentation [Jancsary et al., 2012, Chandra and Kokkinos, 2016, Chandra et al., 2017], where $\mathbf{z}$ are input RGB images and $\mathbf{x}$ contains a semantic label per pixel. In this setting, the models tend to generate a $\boldsymbol{\eta}(\mathbf{z})$ that is a noisy estimate of the semantic labels, and commonly, $(\mathbf{\Lambda}(\mathbf{z}))^{-1}$ corresponds to an edge aware smoothing filter, which improves the final prediction $\boldsymbol{\mu}$, as the noise component, $\boldsymbol{\epsilon}$, is usually ignored for the forward model.

Compared to our method, there are two limitations in the G-CRF parametrisation. First, solving a sparse system of equations, $\mathbf{\Lambda}\boldsymbol{\mu} = \boldsymbol{\eta}$, is required both during training and inference to predict the mean decoding, $\boldsymbol{\mu}$. While, in our approach, a decoder network directly predicts $\boldsymbol{\mu}$ in a single forward pass. Second, a partition function is needed in order to ensure that the right hand side of equation 3.20 is a valid log probability, as this equation is used as an approximation to equation 3.19. Previous work, either defines a pseudo-likelihood approximation [Jancsary et al., 2012], which requires summing over the states of each node in the G-CRF, which is only tractable for discrete data, or ignores the partition function [Chandra and Kokkinos, 2016, Chandra et al., 2017], by setting the problem as an optimisation to find $\boldsymbol{\mu}$. In contrast, we offer a tractable approach, in equation 3.16, to evaluate the partition function, which only requires a sum over n elements.

An interesting property of G-CRF is the conditional independence of a pixel given a set of other pixels. For a given pixel x_i, we can separate all the other pixels into two sets: x_A, containing all pixels connected to x_i, *i.e.* $\forall p \in A,\ \lambda_{ip} \neq 0$, and x_B, containing all the pixels not connected to x_i, *i.e.* $\forall p \in B, \lambda_{ip} = 0$, where $x_S = \{x_p \,|\, p \in S\}$ and S is either A or B. Given this separation in two sets, the joint probability for the pixels is factorised as

$$p(x_i, x_A, x_B) = p(x_i|x_A)p(x_A, x_B). \tag{3.22}$$

In other words, x_i is conditionally independent of all pixels in the set B, given all pixels in the set A, which can be written as $p(x_i|x_A, x_B) = p(x_i|x_A)$. A proof of this conditional independence statement is provided in Section A.1.1 in the appendix. As our decoder model is equivalent to a G-CRF, the same property applies to it. This and similar Markov properties have been extensively used to model images [Murphy, 2012, Bishop, 2006]. For our model, as we only have conditional dependence between neighbours, this means that if two distant pixels are modelled as correlated (outside of their neighbourhood), there must be a path of correlated pixels between them, as illustrated in Fig. 3-4.

Low Rank Sparse Cholesky Decomposition

To model larger images, the size of the neighbourhood should be increased accordingly. Yet, the number of non zero-elements of the precision matrix increases quadratically with the size of the neighbourhood, which is not tractable. A solution that was explored is to reduce the dimensionality of $\mathbf{L}$ by approximating it with a learnable basis

$$\mathbf{L} = s(\mathbf{B}\mathbf{W}) = s(\mathbf{T}), \tag{3.23}$$

where $\mathbf{B}$ is a $n_\mathrm{c} \times n_\mathrm{b}$ dense matrix containing the basis, $\mathbf{W}$ is a $n_\mathrm{b} \times n$ dense matrix of weights, n is the number of pixels in the input image, n_b is the number of basis vectors, $n_\mathrm{c} = (n_\mathrm{f}^2 + 1)/2$ and $\mathbf{T}$ is a $n_\mathrm{c} \times n$ dense matrix containing the non-zero elements in $\mathbf{L}$. The operator $s(\cdot)$ rearranges and pads with zeroes the values in $\mathbf{T}$ to generate the $\mathbf{L}$ matrix, and it is explained in Section A.1.3 in the appendix. For now, we note that the diagonal of $\mathbf{L}$ is stored in the first row of $\mathbf{T}$, *i.e.* $\ell_{i,i} = t_{0,i}$, where $t_{0,i}$ is the i^{th} element in the first row of $\mathbf{T}$.

Under this formulation, the covariance network output becomes $\mathbf{W}$, while the basis matrix, $\mathbf{B}$, is learned at training time and it is shared for all the images. Note that this formulation only provides savings over the previous approach if $n_\mathrm{b} < n_\mathrm{c}$, *i.e.* if the basis is not complete.

We now turn our attention to how practical this parametrisation is:

(i) Sampling is performed with the procedure discussed in the preceding section, which requires solving a sparse system of equations.

(ii) The reconstruction error, $\mathbf{r}^\intercal \mathbf{\Lambda} \mathbf{r}$, is also computed as in the above section. The log determinant term can be evaluated as

$$\log(|\mathbf{\Sigma}|) = -2 \sum_{j=0}^{n-1} \log\left(t_{0,j}\right). \tag{3.24}$$

(iii) Ensuring that the estimated precision matrix is positive definite and symmetric requires constraining $\mathbf{W}$ and $b_{0,j}$ to be positive, where $b_{0,j}$ is the j^{th} element in the first row of B. We achieve this by directly predicting log values for that row.

(iv) The computational complexity is $\mathcal{O}(n_{\mathrm{c}}\, n_{\mathrm{b}} + n_{\mathrm{b}}\, n)$. This is tractable if $n_{\mathrm{b}} \ll n$ and $n_{\mathrm{f}} \ll n$, in other words, the growth is linear in the number of pixels.

Experimentally, we saw only marginal gains when using either the basis, or the dilated sparsity pattern with the basis. However, the basis might be useful when using sparsity patterns with larger neighbourhood size, n_{f}, or image size, n.

This approach is equivalent to not using a basis if $n_{\mathrm{b}} = n_{\mathrm{c}}$ and $\mathbf{B} = \mathbf{I}$. In this case, the log determinant evaluation is simplified to

$$\log(|\mathbf{\Sigma}|) = -2 \sum_{j=0}^{n-1} \log\left(w_{0,j}\right), \tag{3.25}$$

where $w_{0,j}$ is the j^{th} element in the first row of W. The positive definite constraint is fulfilled if the first row of W is positive, which again is achieved by predicting log values. As there will be benefits in employing this basis formulation, even with a fixed matrix $\mathbf{B} = \mathbf{I}$, we will continue to use it in the following sections.

3.3.4 Efficiency

Two assumptions, which limit the applicability of the method have been made through this chapter. The first consists of limiting the input to be grey-scale images, and the second is the assumption that the deep learning framework of choice supports sparse matrix operations.

In this section we present an approximation to be able to model multivariate Gaussian distributions for colour images with the same parameter budget as for grey-scale images. Subsequently, we describe a tractable implementation of our model in terms of dense matrix operations, which is amenable to parallelisation on the GPU.

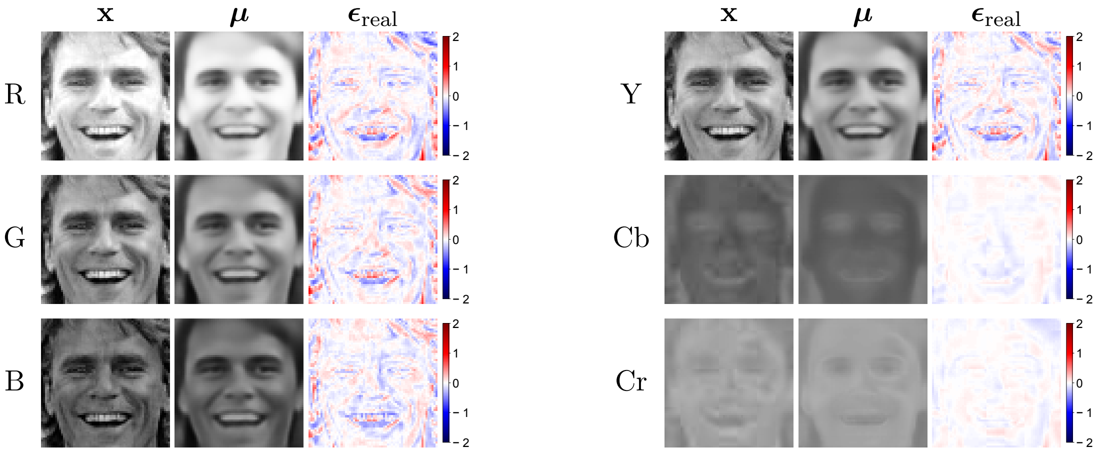


Figure 3-6: Input, $\mathbf{x}$, reconstructions, $\boldsymbol{\mu}$, and residuals, $\boldsymbol{\epsilon}_{\text{real}} = \mathbf{x} - \boldsymbol{\mu}$, in RGB and YCbCr colour spaces for a VAE with diagonal covariance trained with RGB images. In RGB space all the channels contain highly structured residuals, and in each channel the residual has a similar magnitude. In the YCbCr space the Cb and Cr residuals have lower magnitude than the Y channel residual.

Colour images

We present a structured uncertainty approach that can model colour images with a minimal increment in the number of parameters over modelling grey-scale images. To achieve this, we first observe that in a luminance colour space, such as YCbCr, the high-frequency details of the image are mostly encoded in the luminance channel, Y. This fact has been used by image compression algorithms like JPEG, where the colour channels Cb and Cr are spatially quantised with minimal loss of quality in the resulting images [Wallace, 1992]. As previously described, VAEs struggle to model high-frequency details, leading to highly structured residuals for the luminance channel, in contrast the Cb and Cr, which are smooth by nature, are easier for the VAE to model, leading to lower amplitude residuals, as shown in Fig. 3-6.

Therefore, we propose to model each YCbCr channel independently:

$$p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{\text{Y}} \,|\, \mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}_{\text{Cb}} \,|\, \mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}_{\text{Cr}} \,|\, \mathbf{z}), \tag{3.26}$$

where $\mathbf{x}_{\text{Y}}$, $\mathbf{x}_{\text{Cb}}$, $\mathbf{x}_{\text{Cr}}$ denote the Y, Cb and Cr channels in $\mathbf{x}$.

The luminance channel is modelled using a structured Gaussian likelihood, while the remaining channels use factorised Gaussian likelihoods

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{\mathrm{Y}} \,|\, \mathbf{z}) = \mathcal{N}\big(\, \boldsymbol{\mu}_{\mathrm{Y}}(\mathbf{z}), \boldsymbol{\Sigma}(\mathbf{z}) \,\big), \tag{3.27}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{\mathrm{Cb}} \,|\, \mathbf{z}) = \mathcal{N}\big(\, \boldsymbol{\mu}_{\mathrm{Cb}}(\mathbf{z}), \boldsymbol{\sigma}^2_{\mathrm{Cb}}(\mathbf{z})\, \mathbf{I} \,\big), \tag{3.28}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{\mathrm{Cr}} \,|\, \mathbf{z}) = \mathcal{N}\big(\, \boldsymbol{\mu}_{\mathrm{Cr}}(\mathbf{z}), \boldsymbol{\sigma}^2_{\mathrm{Cr}}(\mathbf{z})\, \mathbf{I} \,\big), \tag{3.29}$$

where $\boldsymbol{\mu}_{\mathrm{Y}}(\mathbf{z})$, $\boldsymbol{\Sigma}(\mathbf{z})$, $\boldsymbol{\mu}_{\mathrm{Cb}}(\mathbf{z})$, $\boldsymbol{\sigma}^2_{\mathrm{Cb}}(\mathbf{z})$, $\boldsymbol{\mu}_{\mathrm{Cr}}(\mathbf{z})$, $\boldsymbol{\sigma}^2_{\mathrm{Cr}}(\mathbf{z})$ are non-linear functions of the latent variable $\mathbf{z}$, which are implemented as neural networks. A common decoder is used as a backbone, which leads to sharing most of the network parameters. Each output only adds a single layer to the decoder at the corresponding image resolution.

Many datasets contain images compressed in JPEG, and as aforementioned this format spatially quantizes the Cb and Cr channels. Usually, during the JPEG encoding the Cb and Cr channels are downsampled to half of the image resolution. Thus, JPEG decoders upsample these channels back to the original resolution. The loss of information due the downsampling/upsampling process can be problematic. The colour channels contain less information than the luminance one, so they should be treated differently. In order to equalise the amount of information per pixel across channels, the Cb and Cr channels are downsampled to the same resolution that was used for the JPEG encoding.

GPU Implementation

Our approach can be implemented efficiently in modern GPU architectures with frameworks that lack support for sparse tensor operations. We use the low rank sparse Cholesky decomposition discussed in the previous section. For the likelihood evaluation, we reformulate the squared error term in the log likelihood as a 2D convolution operation, as shown in Fig. 3-7. Our solution for drawing random samples tractably from the distribution is based on a hybrid GPU/CPU approach.

For the likelihood evaluation, a reshape operator $f(\cdot)$ is defined, which does the reverse of vectorisation. This is required as the 2D convolution is evaluated in image space, *i.e.* before vectorising the inputs images, which have an image height and width of n_{h} and n_{w} respectively. The operator, $f(\cdot)$, reshapes its input to $n_{\mathrm{h}} \times n_{\mathrm{w}} \times c$ where c is a variable number of channels depending on the input size. Thus, the dimensionality of the input to the reshape operator must be a multiple of the number of pixels, $n = n_{\mathrm{h}} n_{\mathrm{w}}$.

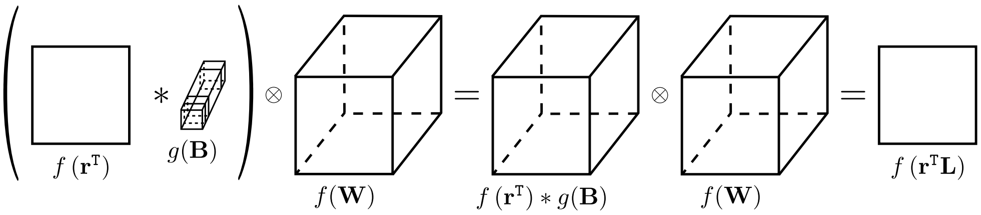

$f\left(\mathbf{r}^{\mathrm{T}}\right)$
$*$
$g(\mathbf{B})$
$\otimes$
$f(\mathbf{W})$
$=$
$f\left(\mathbf{r}^{\mathrm{T}}\right) * g(\mathbf{B})$
$\otimes$
$f(\mathbf{W})$
$=$
$f\left(\mathbf{r}^{\mathrm{T}}\mathbf{L}\right)$

Figure 3-7: Tractable GPU evaluation of $\mathbf{r}^{\mathrm{T}}\mathbf{\Lambda}\mathbf{r}$ using the dense matrix representation of $\mathbf{B}$ and $\mathbf{W}$, where $\mathbf{r}$ is the residual, $f(\cdot)$ and $g(\cdot)$ are reshape operators defined in the text. Firstly, the operation $f(\mathbf{r}^{\mathrm{T}}) * g(\mathbf{B})$ is evaluated, where the residual image, $f(\mathbf{r}^{\mathrm{T}})$, is convolved with the basis kernels, $g(\mathbf{B})$. Secondly, the tensor resulting from the convolution is linearly combined with the weights $f(\mathbf{W})$, where $\otimes$ denotes element-wise multiply and summing over the channel dimension. Thirdly, a vector-vector multiplication $(\mathbf{r}^{\mathrm{T}}\mathbf{L})^{\mathrm{T}}(\mathbf{r}^{\mathrm{T}}\mathbf{L})$ is needed to get the final value.

The squared error term, equation 3.7, in the likelihood is defined as

$$\mathbf{r}^{\mathrm{T}}\mathbf{\Lambda}\mathbf{r} = \mathbf{r}^{\mathrm{T}}\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{r} \tag{3.30}$$

where

$$\mathbf{r}^{\mathrm{T}}\mathbf{L} = f^{-1}\left(\left(f\left(\mathbf{r}^{\mathrm{T}}\right) * g(\mathbf{B})\right) \otimes f(\mathbf{W})\right), \tag{3.31}$$

where $g(\cdot)$ is a reshape and zero pad operator to $n_{\mathrm{f}} \times n_{\mathrm{f}} \times n_{\mathrm{b}}$, which is described in more detail below, $*$ denotes 2D convolution, $\otimes$ is a linear combination operator, which does element-wise multiply and sums over n_{b}, the channel dimension. Thus, this operation can be directly evaluated using the dense matrices $\mathbf{B}$ and $\mathbf{W}$.

In more detail, the weights matrix, $\mathbf{W}$, is reshaped to a $n_{\mathrm{h}} \times n_{\mathrm{w}} \times n_{\mathrm{b}}$ tensor by $f(\mathbf{W})$, and the residual, $f(\mathbf{r}^{\mathrm{T}})$, has a shape of $n_{\mathrm{h}} \times n_{\mathrm{w}} \times 1$. Each column in $\mathbf{B}$ corresponds to an individual kernel of size $n_{\mathrm{f}} \times n_{\mathrm{f}}$ for the convolution operation. For example, for a $n_{\mathrm{f}} = 3$ and $n_{\mathrm{b}} = 2$, the basis matrix is defined as

$$\mathbf{B} = \begin{bmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \\ b_{2,0} & b_{2,1} \\ b_{3,0} & b_{3,1} \\ b_{4,0} & b_{4,1} \end{bmatrix}. \tag{3.32}$$

The reshape and zero pad operation, g, leads to a $3 \times 3 \times 2$ tensor

$$g(\mathbf{B}) = \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & b_{0,0} & b_{1,0} \\ b_{2,0} & b_{3,0} & b_{4,0} \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & b_{0,1} & b_{1,1} \\ b_{2,1} & b_{3,1} & b_{4,1} \end{bmatrix} \right\}. \tag{3.33}$$

The log determinant term in the log likelihood does not require any modifications. Therefore, it is evaluated as described in equation 3.24, or using the simplified version of equation 3.25, if no basis matrix is used, *i.e.* $\mathbf{B} = \mathbf{I}$.

In practice, this approach is implemented in a fully convolutional fashion, where the network directly outputs a $n_\mathrm{h} \times n_\mathrm{w} \times n_\mathrm{c}$ tensor which corresponds to $f(\mathbf{W})$, and another of shape $n_\mathrm{h} \times n_\mathrm{w} \times 1$ for $f(\boldsymbol{\mu})$. Therefore, the only reshape operators used in equation 3.31 are $f^{-1}(\cdot)$, which vectorises the input tensor from $n_\mathrm{h} \times n_\mathrm{w} \times 1$ to n, and $g(\cdot)$. The log determinant evaluation of equation 3.24 also requires using $f^{-1}(\cdot)$, while the simplified one in equation 3.25 can be stated in terms of $f(\mathbf{W})$ as

$$\log(|\boldsymbol{\Sigma}|) = -2 \sum_{i=0}^{n_\mathrm{h}-1} \sum_{j=0}^{n_\mathrm{w}-1} \log\left(f(\mathbf{W})_{i,j,0}\right). \tag{3.34}$$

Interestingly, for the case $\mathbf{B} = \mathbf{I}$, the convolution operation corresponds to shifting the image in an axis-aligned manner, where the shift is constrained to be an integer for each dimension. An example of this shifting is shown in Section A.1.4 in the appendix.

The sampling operation, as described in equation 3.15, can be problematic in frameworks that do not support sparse tensors, and sparse system of equations solvers. However, as sampling is only required at inference time, this allows for hybrid solutions, such as evaluating $\mathbf{T}$ on GPU, and efficiently solving the system on CPU with an off-the-self sparse solver.

3.3.5 Priors

As our generative model estimates both a mean and a precision matrix per input image, we must employ some regularisation to avoid poor solutions. Intuitively these poor solutions arise from the underconstrained nature of the problem, as the VAE network is tasked with estimating the parameters of a multivariate Gaussian distribution given a single data sample, the input image.

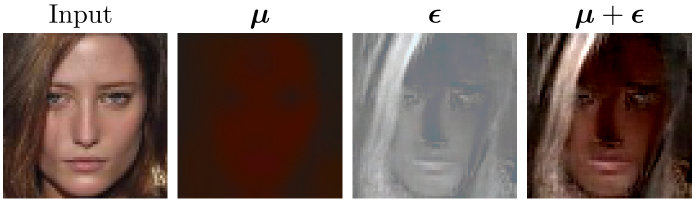


Figure 3-8: Example of a reconstruction from a VAE trained with a multivariate Gaussian distribution with our sparse Cholesky decomposition approach, *i.e.* $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}\big(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\big)$, where $\boldsymbol{\Lambda} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$. If trained without any regularisation, the image content is mostly modelled on the residuals, $\boldsymbol{\epsilon}$. Thus, producing means, $\boldsymbol{\mu}$, which do not resemble the input image.

The estimated variance should be lower bounded by the acquisition noise in the data, *i.e.* the lower bound is proportional to the expected noise in the images. Intuitively, for images this usually entails that little variance is expected on the predicted covariance matrix, i.e. the model should be certain about its predicted mean at any given pixel, assuming a complex enough model. Experimentally, we found that without any regularisation a randomly initialised network has a tendency to model most of the image content in the precision matrix, $\boldsymbol{\Lambda}$, rather than in the mean, $\boldsymbol{\mu}$, as shown in Fig. 3-8, which is undesirable. Another issue is the prediction of spurious correlations, which are not well supported by the data, but arise from only having a single residual sample per image.

Following a Bayesian route, we explore the addition of a prior distribution over the predicted parameters of the multivariate Gaussian distribution. For an input image, $\mathbf{x}$, an informative prior for the predicted mean, $\boldsymbol{\mu}$, would be centred on the input image itself. Therefore, we lack a good prior for the mean, as priors do not have access to the input data. Instead, we focus on adding a prior on the predicted precision matrix, $\boldsymbol{\Lambda}$. Accordingly, the variational bound of the VAE is augmented to

$$L = L_{\text{VAE}} + \mathbb{E}_{\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}\,|\,\mathbf{x})} \left[\log\ p_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}\,|\,\mathbf{z})\right], \tag{3.35}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}\,|\,\mathbf{z})$ is the prior distribution over the estimated matrices, $\boldsymbol{\Lambda} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$. Note that this addition does not transform $\boldsymbol{\Lambda}$ (nor $\mathbf{L}$) into a random variable, i.e. a distribution over $\boldsymbol{\Lambda}$ (or $\mathbf{L}$) matrices is not being estimated. Therefore, samples are not drawn from the $p_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}\,|\,\mathbf{z})$ distribution, only the probability density function of the distribution is evaluated in the above equation. To simplify notation, the dependency on $\boldsymbol{\theta}$ and $\mathbf{z}$ is dropped for the remainder of this section, i.e. $p(\boldsymbol{\Lambda})$ is used to denote $p_{\boldsymbol{\theta}}(\boldsymbol{\Lambda}\,|\,\mathbf{z})$.

We consider four different prior distributions: a **Wishart** distribution, a **Cholesky-**

Wishart distribution, a **Sparse Cholesky-Wishart** distribution, and a combination of **Gamma-Gaussian** distributions.

Wishart distribution

A common prior used for covariance matrices is the inverse Wishart distribution, as this is a conjugate prior for the covariance matrix of a Gaussian distribution. In our case, as we are working with precision matrices, $\mathbf{\Lambda}$, we use a Wishart distribution over precision matrices.

The probability density function of a Wishart distribution is defined as

$$p(\mathbf{\Lambda}) = W(\mathbf{\Lambda}|\mathbf{V}, p) = \frac{1}{2^{pn/2}|\mathbf{V}|^{p/2}\Gamma_n\left(\frac{p}{2}\right)}|\mathbf{\Lambda}|^{(p-n-1)/2}e^{-(1/2)\mathrm{tr}(\mathbf{V}^{-1}\mathbf{\Lambda})} \tag{3.36}$$

where the normalisation factor, Γ_n, is a multivariate Gamma function defined as

$$\Gamma_n\left(\frac{p}{2}\right) = \pi^{n(n-1)/4}\prod_{j=1}^{n}\mathbf{\Gamma}\left(\frac{p}{2} - \frac{j-1}{2}\right), \tag{3.37}$$

$$\mathbf{\Gamma}(a) = \int_0^\infty t^{a-1}e^{-t}dt, \tag{3.38}$$

where $\mathbf{\Gamma}(\cdot)$ is known as the Gamma function, n denotes the dimensionality of $\mathbf{\Lambda}$, *i.e.* $\mathbf{\Lambda}$ is a $n \times n$ matrix, $\mathbf{V}$ is a $n \times n$ scale matrix, and p is a scalar defining the degrees of freedom of the distribution. The density function is only defined for symmetric positive definite matrices, $\mathbf{\Lambda}$ and $\mathbf{V}$. The degrees of freedom parameter is constrained to $p \geq n$, with $p = n$ being an uninformative prior, and larger values being more informative.

This approach seems to require access to the precision matrix, $\mathbf{\Lambda}$, while we only have tractable access to the Cholesky matrix, $\mathbf{L}$. However, the probability density function, equation 3.36, can be re-written using the Cholesky factors $\mathbf{L}$, as will be shown in equation 3.42.

From that definition, all the terms can be efficiently computed as long as the scale matrix, $\mathbf{V}$, is diagonal. Luckily, this is the case, as a preference for removing spurious correlations, and having little noise in the data corresponds to a diagonal covariance matrix $\mathbf{\Sigma}_0 = \kappa^2\mathbf{I}$, where κ^2 is a scalar with the expected noise variance. As the prior will be used in an optimisation setting, it will push the precision matrices predicted by the covariance network towards the mode of the prior.

The mode of the Wishart distribution is defined as

$$\mathbf{\Lambda}_{\text{mode}} = (p - n - 1)\mathbf{V}, \tag{3.39}$$

which is only valid for $p \geq n+1$. Therefore, the scale matrix that encourages generating $\mathbf{\Lambda}$ matrices that produce diagonal covariance matrices, $\mathbf{\Sigma}_0 = \kappa^2\mathbf{I}$, is defined as

$$\mathbf{V} = \frac{1}{(p-n-1)\kappa^2}\mathbf{I}. \tag{3.40}$$

A prior with a scaled identity matrix, such as this one, which expects little noise in the data, is intended to force the network to model most of the image content in the means, $\boldsymbol{\mu}$. However, as there is only a single image per learned distribution, the estimated precision matrix might be close to singular, which would lead to probability density functions that are not well behaved, leading to flawed solutions. This issue might arise in either $p(\mathbf{\Lambda}) = W(\mathbf{V}, p)$ or $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}\big(\boldsymbol{\mu}, \mathbf{\Lambda}^{-1}\big)$, as the precision matrix appears in both distributions.

Implementation details on how to efficiently evaluate equation 3.36 using our dense basis representation, $\mathbf{BW}$, can be found in Section A.1.9 in the appendix.

Cholesky-Wishart distribution

In this section we provide a distribution over Cholesky matrices $\mathbf{L}$, which is partly based on the work by [Dillon et al., 2017]. This is more consistent with our method, than using a Wishart distribution over precision matrices, as it is a prior on the parameters that are estimated, $\mathbf{L}$, rather than on a function of them $\mathbf{\Lambda} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$.

A change of variables $q(\mathbf{L}) = \mathbf{\Lambda} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$ and $\mathbf{L} = q^{-1}(\mathbf{\Lambda})$ is applied on the Wishart distribution, which leads to the Cholesky-Wishart distribution, with a probability density function defined by

$$p(\mathbf{L}) = W^c(\mathbf{L}|\mathbf{V}, p) = W(q(\mathbf{L})|\mathbf{V}, p)\left|\frac{\partial(q(\mathbf{L}))}{\partial\mathbf{L}}\right| = W(\mathbf{L}\mathbf{L}^{\mathsf{T}}|\mathbf{V}, p)\left|\frac{\partial(\mathbf{L}\mathbf{L}^{\mathsf{T}})}{\partial\mathbf{L}}\right|. \tag{3.41}$$

Replacing the precision matrix, $\mathbf{\Lambda}$, for its Cholesky factors, $\mathbf{L}\mathbf{L}^{\mathsf{T}}$, in equation 3.36 leads to

$$W(\mathbf{L}\mathbf{L}^{\mathsf{T}}|\mathbf{V}, p) = \frac{1}{2^{pn/2}|\mathbf{V}|^{p/2}\Gamma_n\left(\frac{p}{2}\right)}|\mathbf{L}|^{(p-n-1)}e^{-(1/2)\text{tr}(\mathbf{V}^{-1}\mathbf{L}\mathbf{L}^{\mathsf{T}})}, \tag{3.42}$$

and the second term is equivalent to

$$\left|\frac{\partial(\mathbf{L}\mathbf{L}^{\intercal})}{\partial \mathbf{L}}\right| = 2^{n} \prod_{i=0}^{n-1} \ell_{i,i}^{n-i}. \tag{3.43}$$

This distribution has similar computational costs as the Wishart distribution, as the additional term in equation 3.43, only involves the diagonal values of the $\mathbf{L}$ matrix, which are also used in the log determinant in equation 3.16. A derivation of the Cholesky-Wishart distribution can be found in Section A.1.9 in the appendix.

As we maintain the preference for a diagonal covariance matrix, $\mathbf{\Sigma}_{\text{mode}} = \kappa^{2}\mathbf{I}$ in the prior, we proceed to define how to set the scale matrix to pursue this objective. First, we inspect the mode of the distribution, which for a diagonal $\mathbf{V}$ matrix is defined as

$$\hat{\lambda}_{i,i} = v_{i,i}\left(\frac{i(1-n)}{n-1} + p - 1\right), \tag{3.44}$$

$$\hat{\lambda}_{i,j} = 0, \quad \forall i,j \quad \text{s.t.} \quad i \neq j \quad \text{and} \quad j < i, \tag{3.45}$$

where $\hat{\lambda}_{i,i}$ and $\hat{\lambda}_{i,j}$ are the diagonal and off-diagonal elements of $\mathbf{\Lambda}_{\text{mode}}$, and $v_{i,i}$ denotes the i^{th} element in the diagonal of $\mathbf{V}$. Therefore, the diagonal scale matrix, $\mathbf{V}$, must be set to

$$v_{i,i} = \frac{\kappa^{-2}(n-1)}{i(1-n) + (n-1)(p-1)}, \tag{3.46}$$

where κ is a scalar with the expected noise standard deviation. A proof that the result presented above is indeed the mode of the distribution can be found in Sections A.1.6 and A.1.10 in the appendix.

Both the Wishart and Cholesky-Wishart distribution define the probability density function over dense precision matrices, $\mathbf{\Lambda}$, and lower triangular matrices, $\mathbf{L}$, respectively. As we have sparse lower triangular matrices, this implies adding a constant term proportional to the number of zero elements. This could be numerically problematic for the optimisation, and complicates model comparison. Therefore, we proceed to derive a prior distribution that accounts for the sparsity pattern in the Cholesky matrix, $\mathbf{L}$.

Gamma-Gaussian distribution

The Cholesky-Wishart distribution (for a lower triangular matrix $\mathbf{L}$ and a diagonal scale matrix $\mathbf{V}$) is equivalent to a square root Gamma distribution for the diagonal values in the Cholesky matrix and a Gaussian for the off-diagonal ones

$$p(\mathbf{L}) = W^c(\mathbf{L}|\mathbf{V}, p) = \prod_i p(\ell_{i,i}|\mathbf{V}, p) \prod_{i,j} p(\ell_{i,j}|\mathbf{V}, p), \tag{3.47}$$

$$p(\ell_{i,i}|\mathbf{V}, p) = \mathrm{Ga}^{\frac{1}{2}}\left(0.5\left(\frac{i(1-n)}{n-1} + p\right), \frac{0.5}{v_{i,i}}\right), \tag{3.48}$$

$$p(\ell_{i,j}|\mathbf{V}, p) = \mathcal{N}\big(0, v_{i,i}\big), \quad \forall i, j \quad \text{s.t.} \quad i \neq j \quad \text{and} \quad j < i, \tag{3.49}$$

where $i \in [0, 1, \cdots, n-1]$, $v_{i,i}$ denotes the i^{th} element in the diagonal of $\mathbf{V}$, and $p(\ell_{i,j})$ is only evaluated for the non-zero off-diagonal elements in $\mathbf{L}$. This equivalence is proved in Section A.1.6 in the appendix. The square root Gamma distribution arises from applying a change of variables in the Gamma distribution as demonstrated in Section A.1.7, where the shape and rate parametrisation is used. Moreover, the square root Gamma distribution is equivalent to the Nakagami distribution [Nakagami, 1960] and the chi distribution. Henceforth, we refer to this combined distribution as the Gamma-Gaussian distribution.

As discussed in Section A.1.6, the equivalence between the Cholesky-Wishart and the Gamma-Gaussian distribution only holds for diagonal scale matrices, $\mathbf{V}$, which was the case considered in the previous section. Therefore, the result for the mode of the distribution, equations 3.44 and 3.45, and for the scale parameter, equation 3.46, can be directly used in the Gamma-Gaussian distribution.

There are several advantages of the Gamma-Gaussian approach, with respect to the Wishart or Cholesky-Wishart distributions. The likelihood is easier to evaluate, as the distributions have simpler probability density functions. Moreover, this approach only evaluates the likelihood for the parameters that are defined under the model, *i.e.* it is aware about the sparsity pattern in $\mathbf{L}$.

For completeness, a Cholesky-Wishart distribution, which is equivalent to the Gamma-Gaussian approach for sparse matrices $\mathbf{L}$ is defined below.

Sparse Cholesky-Wishart distribution

As discussed above, the Cholesky-Wishart distribution evaluates the probability for all the elements in the lower triangular part of $\mathbf{L}$. Under the Gamma-Gaussian prior the likelihood for the zero elements is not evaluated. However, if it were, the likelihood for those elements would be a constant, as by construction the elements are constrained to be zero. Therefore, by removing the likelihood terms corresponding to those elements from the Cholesky-Wishart density function, a new distribution which accounts only for the non-zero elements can be defined. Formally we denote this distribution as the sparse Cholesky-Wishart, and it is defined by the following probability density function

$$p(\mathbf{L}) = W^c_{\text{sp}}(\mathbf{L} \,|\, \mathbf{V}, p) = \frac{W^c(\mathbf{L} \,|\, \mathbf{V}, p)}{\prod_{\ell_{i,j} \in K} \mathcal{N}\big(\, 0 \,|\, 0, v_{i,i} \,\big)}, \tag{3.50}$$

where K is the set of elements that are zero by construction in the lower-triangular part of $\mathbf{L}$. Note that when evaluating the probability density function of this distribution, the denominator is a constant.[5] The Sparse Cholesky-Wishart distribution is only defined for diagonal scale matrices, $\mathbf{V}$, as the equivalence between the Gamma-Gaussian and the Cholesky-Wishart only holds for that choice of hyper-parameters.

Discussion of the prior distributions

The aforementioned approaches for learning multivariate Gaussian distributions proved to be challenging in practice. For any values of the hyper-parameters in the priors, p and $\mathbf{V}$, we found that the optimisation was dominated by the cost of the prior due to the number of covariance parameters being estimated. Both the sparse Cholesky-Wishart and the Gamma-Gaussian priors made the learned precision matrices too strongly diagonal, and compared to a model without any regularisation, the improvements on the predicted means were minor.

Intuitively, the number of parameters in the precision matrix, $\mathbf{\Lambda}$, is much larger than the number of parameters in the means, $\boldsymbol{\mu}$. For example, there are a maximum of five parameters per pixel in $\mathbf{L}$ for a 3×3 neighbourhood in a grey-scale image. In general, this indicates that a prior on the precision would have a larger weight than one on the

[5] $\mathcal{N}\big(\, 0 \,|\, 0, v_{i,i} \,\big)$ is the probability density function of a normal distribution with zero mean and $v_{i,i}$ variance, evaluated at 0. Hence, it can be evaluated as:

$$\mathcal{N}\big(\, 0 \,|\, 0, v_{i,i} \,\big) = \frac{1}{\sqrt{v_{i,i}}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{0-0}{\sqrt{v_{i,i}}}\right)} = (2\pi v_{i,i})^{-\frac{1}{2}} \,.$$

means. An uninformative prior has roughly a uniform high cost regardless of the values in $\mathbf{L}$. An informative one has a low cost for values that agree with the prior, and a potentially very high cost for values that disagree. Therefore, the prior must cover well the desired distribution of covariance matrices, which is unknown, and be informative enough to avoid dominating the cost of the optimisation.

A clarifying example of how informative the prior would need to be, can be illustrated for the Wishart distribution. In our case a neural network, which amortises different observations over a dataset, is used; yet, for this example, a standard estimator is assumed for simplicity. The degrees of freedom hyper-parameter, p, indicates how informative the prior distribution is, and it has been interpreted as indicating that p precision matrices were observed [Gelman and Hill, 2007, Lunn et al., 2012]. In our case, only a single residual is observed, which would imply a value of $p < 1$ for the prior, as several residual images are needed to accurately estimate a single precision matrix. Unfortunately, the distribution is not well defined if $p < n$, as samples from the distribution are no longer constrained to be positive definite matrices.

A principled approach to learn the parameters of the prior distribution is to employ a hierarchical model with hyper-priors. However, this would entail adding an additional network to predict the parameters from the latent representation. Moreover, it is known that deep hierarchical models are difficult to train [Neal, 2012].

3.3.6 Regularised precision matrix estimation

In the previous section regularisation techniques based on prior distributions were discussed, which proved to be challenging in practice. Two alternative approaches for regularising the estimation of precision matrices are proposed in this section. These methods are not as principled as the priors, yet, empirically they were found to work well.

Independent parameter estimation (IPE)

In this method we propose to learn the distribution in two separate steps, as shown in Fig. 3-9. In the first step, the model is trained with a factorised Gaussian likelihood until convergence. This ensures that the decoder network learns to model the image as closely as possible on the means, $\boldsymbol{\mu}$. In the second step, the covariance network is trained with the rest of the model fixed, *i.e.* the “Encoder” and “$\boldsymbol{\mu}$-Decoder” remain

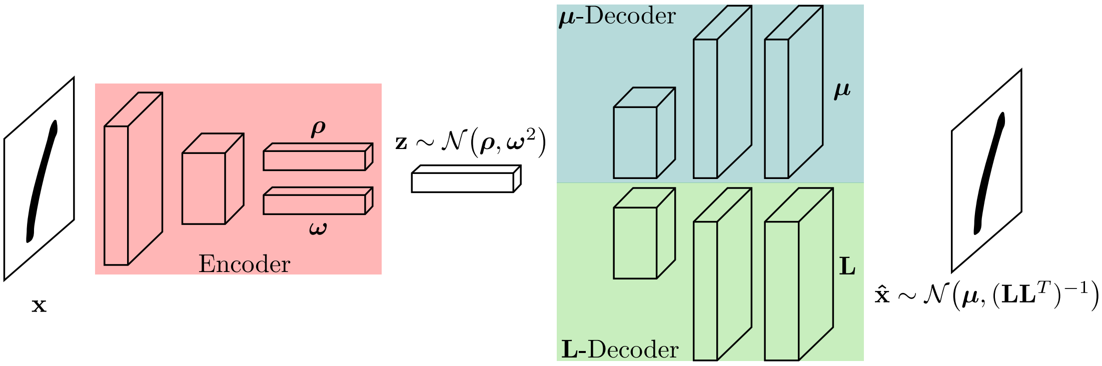

$\mathbf{x}$
Encoder
$\boldsymbol{\rho}$
$\boldsymbol{\omega}$
$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\rho}, \boldsymbol{\omega}^2)$
$\boldsymbol{\mu}$-Decoder
$\boldsymbol{\mu}$
$\mathbf{L}$
$\mathbf{L}$-Decoder
$\hat{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\mu}, (\mathbf{L}\mathbf{L}^T)^{-1})$

Figure 3-9: Structured uncertainty prediction in two steps. First, in a VAE, $\boldsymbol{\mu}$ and $\mathbf{z}$ are learned using a factorised Gaussian likelihood. Second, the encoder (highlighted in red) and the $\boldsymbol{\mu}$-decoder (highlighted in blue) are fixed, and the $\mathbf{L}$-decoder (highlighted in green) is learned by maximum likelihood estimation.

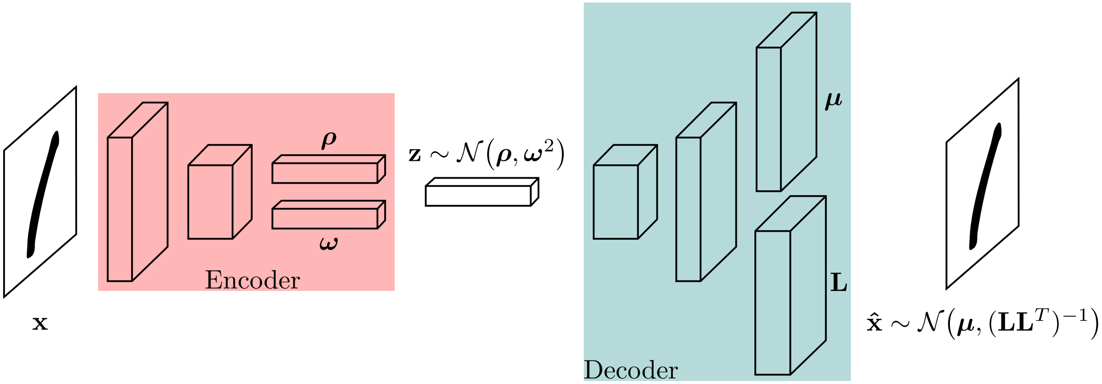
fixed in Fig. 3-9. In practice, this entails removing the layer that is used as output for the factorised covariance matrix estimation in the first step, and replacing it with our sparse Cholesky model.

In most VAE architectures the decoder shares features for both the means and the factorised covariance prediction, which limits the covariance network to only fine tune the final layers exclusively responsible for the covariance prediction. Moreover, the features learned by the shared decoder network usually lead to unstable learning for our approach. A simple solution consists of adding a full decoder for the covariance prediction, which does not share any features with the means prediction network, as shown in Fig. 3-9. Although this approach requires more parameters to be learned, its main advantage is that the covariance network can be applied to any black box method that outputs the mean of a Gaussian distribution.

Shared decoder with regularisation (SDR)

We also propose an alternative approach, which consists of adding two regulariser functions to the VAE loss. Thus, it enables learning in the covariance network with shared parameters with the $\boldsymbol{\mu}$-Decoder, as shown in Fig. 3-10. This also means that the encoder receives gradients from the uncertainty prediction part of the network.

We empirically obtained good results with a mean squared error term and an L_1 regulariser for the off-diagonal elements in $\mathbf{L}$. Both of these terms are intended to approximate the effect of an informative prior, such as an automatic relevance determi-



$\mathbf{x}$
Encoder
$\boldsymbol{\rho}$
$\boldsymbol{\omega}$
$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\rho}, \boldsymbol{\omega}^2)$
Decoder
$\boldsymbol{\mu}$
$\mathbf{L}$
$\hat{\mathbf{x}} \sim \mathcal{N}(\boldsymbol{\mu}, (\mathbf{L}\mathbf{L}^T)^{-1})$

Figure 3-10: Structured uncertainty prediction with shared features on the decoder. The VAE decoder is augmented to use a structured Gaussian likelihood. In order to avoid over fitting due to the covariance network, $\mathbf{L}$, the regularised loss function from equation 3.51 is used.

nation [Mackay, 1995]. The model maximises the following objective

$$L = L_{\text{VAE}} - \alpha \|\mathbf{x} - \boldsymbol{\mu}\|_2^2 - \gamma \sum_{i,j \in K} |\ell_{i,j}|. \tag{3.51}$$

where α and γ are scalar hyper-parameters and K is the set of elements that are not zero by construction in the lower-triangular part of $\mathbf{L}$. The α term ensures that most of the image content is modelled on the means, while the γ term is helpful in removing spurious correlations.

The L_1 regulariser can be computed using the dense basis approach as

$$\gamma \sum_{i,j \in K} |\ell_{i,j}| = \gamma \sum_{i=1}^{n_c - 1} \sum_{j=0}^{n-1} |t_{i,j}|, \tag{3.52}$$

where $t_{i,j}$ is a single element in $\mathbf{T}$.

We found it beneficial to anneal the α weight, which is set initially to a large value. This ensures that the network learns to model the data on the $\boldsymbol{\mu}$, as the variance minimiser term dominates the optimisation. The weight is subsequently lowered, allowing the model to learn meaningful correlations as the objective switches to being dominated by the variational lower bound, L_{VAE}.

3.4 Results

In this section we evaluate the two methods described in the section above. As the **independent parameter estimation** (IPE) approach is more restricted than the **shared decoder with regularisation** (SDR) method, we evaluate the former first on a set of synthetic tasks. This is followed by a comparison of both methods on real data, and an empirical justification of the design decisions taken in this chapter. We proceed by comparing the SDR method to factorised VAEs on two datasets. Next, additional results of the SDR approach that give some insights into the model are provided. Finally, denoising is shown as an example application of our structured uncertainty prediction approach.

3.4.1 Implementation details

Independent parameter estimation (IPE)

The IPE model is evaluated on two custom-made synthetic datasets to demonstrate the capability of the model to accurately describe known residual distributions. We also demonstrate our model on grey-scale cropped face images from the CelebA [Liu et al., 2015] dataset for sampling high frequency details to improve reconstructions. We show some examples of image denoising that takes advantage of the predicted precision matrix to better preserve structure in the denoised image.

All our models are implemented in the TensorFlow [Abadi et al., 2015] framework and they are trained on a single Titan X GPU using the Adam [Kingma and Ba, 2015] optimizer. Unless otherwise stated, the input data for all the experiments is normalised to the range $[-1, 1]$. Additional implementation details are given in each section, and in appendix A.2.1.

Shared decoder with regularisation (SDR)

The SDR model is evaluated on the CelebA [Liu et al., 2015] and LSUN Outdoor Churches [Yu et al., 2015] datasets. The models are trained on the YCbCr colour space for both datasets, and also using only the Y channel (grey-scale) for CelebA. This approach is used to demonstrate that learning the residual distribution on colour images only incurs in a minimal increase in the number of parameters, with respect to learning the residual distribution on grey-scale images. Moreover, we show that this model can achieve similar results as the IPE method with a much smaller parameter budget. Additional implementation details for the experiments can be found in appendix A.2.2.

Baselines

Factorised VAEs Two factorisations are tested for VAEs, a spherical covariance for both the Y, Cb and Cr channels, denoted as VAE (Sph), and a diagonal covariance for Y and spherical covariance for Cb and Cr, denoted VAE (Diag). Employing different noise models for the different channels in the image is justified by the asymmetry in the amount of information that they contain, as discussed in Section 3.3.4. For the experiments employing grey-scale images, the Cb and Cr channels are not used to trained the networks.

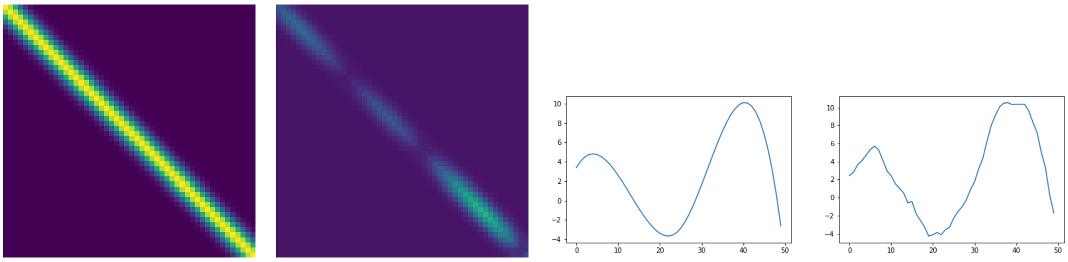
β-VAE We found that VAEs with a diagonal Gaussian likelihood overfitted to the reconstruction error, thus neglecting the KL term for the prior on the latent space, which in turn produces low quality samples. β-VAE [Higgins et al., 2017] addresses this issue by increasing the weight of the KL term in the log likelihood. Therefore, we also show results using this model with a factorised Gaussian likelihood.

3.4.2 Synthetic datasets

The goal of the two synthetic experiments is to evaluate the feasibility of training a covariance network to accurately estimate the residual distribution, where the true mean and covariance matrix are available for validation purposes.

Since the goal here is to evaluate the covariance prediction network, we simplify these experiments by bypassing the use of a generative model, and directly predicting $\mathbf{\Sigma}$ from $\boldsymbol{\mu}$, the noise free data. This means that the input to our covariance prediction network is $\boldsymbol{\mu}$.

Both datasets are constructed by generating a set of $\boldsymbol{\mu}$, and then adding a random sample of correlated noise to them: $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma}(\boldsymbol{\mu}))$, where $\mathbf{\Sigma}(\boldsymbol{\mu})$ is a predefined function of $\boldsymbol{\mu}$. Therefore, the added noise is dependent on the structure of the mean, which imitates the situation on real data. This also reflects the main assumption of our model: that there is sufficient information in the latent variables $\mathbf{z}$ (or in $\boldsymbol{\mu}$ for the synthetic experiments) to estimate the residual distribution $\mathbf{\Sigma}$. Each dataset contains 35,000 training examples and 1,000 test examples.



(a) Prototype covariance matrix, $\mathbf{\Sigma}^{\text{proto}}$.

(b) Example covariance matrix, $\mathbf{\Sigma}$, which is built using equation 3.53.

(c) Example mean $\boldsymbol{\mu}$.

(d) Example spline, $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{\Sigma}\big)$.

Figure 3-11: The splines dataset is synthesised using a prototype covariance matrix (a), that is transformed to generate a number of example covariances (b), where each transformation is a function of a different spline (c). Each example (d) in the dataset is constructed by taking a single sample from a covariance (b) and adding it to the corresponding spline (c).

We emphasise that despite the true covariance matrices being known for these synthetic experiments, we do not use them for training. We train the prediction network using the objective in equation 3.6, which makes no use of the true covariance and mimics the situation with real datasets.

Splines

The first synthetic dataset is composed of 1-D splines, *i.e.* one dimensional signals, with 50 points per example, as shown in Fig. 3-11d. Each spline is comprised of a low frequency component and a correlated high-frequency one. The high-frequency component is produced by a unique covariance matrix per example, that is generated by a deterministic function that takes as input the low-frequency signal. In more detail, given a predefined prototype covariance matrix, $\mathbf{\Sigma}^{\text{proto}}$ (shown in Fig. 3-11a), the covariance matrix, $\mathbf{\Sigma}$, for a particular example is constructed as

$$\sigma_{i,j} = \sigma_{i,j}^{\text{proto}} |\mu_i|. \tag{3.53}$$

This corresponds to scaling each row by the corresponding low frequency component, as shown in Fig. 3-11b.

Finally, a single random sample is drawn from the example covariance matrix and added to the mean $\boldsymbol{\mu}$, which generates the final example $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$, as shown in Fig. 3-11d. Therefore, there is a single example, $\mathbf{x}$, for each unique covariance matrix, $\mathbf{\Sigma}$, which

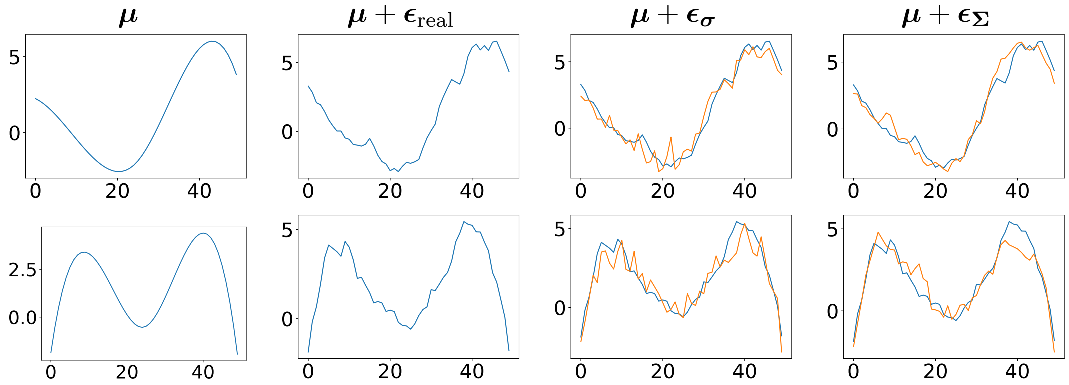


Figure 3-12: Reconstructions with samples from the estimated covariance matrix. Each row corresponds to a different spline on the test dataset. The first column contains the smooth mean, $\boldsymbol{\mu}$, the second shows the spline after adding the sample from the ground truth covariance matrix, $\boldsymbol{\mu}+\boldsymbol{\epsilon}_{\text{real}}$. The third and fourth columns show samples from the diagonal covariance and a dense covariance model. For easier comparisons, in the third and fourth columns the ground truth spline, depicted in blue, is copied from the second column, and the estimated one is shown in orange. The high-frequency component from the dense covariance model seems more correlated than the one from the diagonal model.

replicates the situation on real data.

The covariance prediction network is a multi-layer perceptron (MLP) with two layers of 100 units with ReLU activations and batch normalisation [Ioffe and Szegedy, 2015], and a final layer with 1275 units without an activation function. The final layer directly outputs the lower triangular part of the matrix $\mathbf{L}$, after applying an exponential activation to the diagonal elements. This network does not use our proposed sparse Cholesky decomposition, rather a dense Cholesky matrix is estimated. A dense matrix prediction is manageable in this setting due to the low dimensionality of the data. The diagonal covariance baseline has a similar architecture, with a final layer with 50 units, which correspond to the diagonal elements in $\boldsymbol{\Lambda}$. The models are trained with a learning rate of 1e-4 for 200 epochs. Additional implementation details are given in appendix A.2.1.

Reconstructions for this dataset are obtained by adding a sample from predicted $\boldsymbol{\Sigma}$ to $\boldsymbol{\mu}$. We show results for reconstructions in Fig. 3-12, for the diagonal and dense covariance model. The dense covariance model is able to add a plausible high-frequency component to the input $\boldsymbol{\mu}$. The high-frequency component added by the diagonal covariance approach seems less correlated, which is an indication of its inability to accurately model the ground truth covariance distribution.

	$-\log p(\mathbf{x}\,\vert\,\boldsymbol{\Sigma}(\boldsymbol{\mu}))$	KL	$\Vert\boldsymbol{\Sigma}(\boldsymbol{\mu}) - \boldsymbol{\Sigma}_{gt}\Vert_2$
Ground truth	18.65 ± 0.75	-	-
Diagonal model	42.64 ± 0.83	72.06 ± 0.30	3.81 ± 1.21
Ours	**21.12 ± 0.79**	**3.56 ± 0.18**	**1.26 ± 0.74**

Table 3.1: Quantitative comparison of reconstructions on the splines dataset. KL denotes the KL divergence $D_{KL}(\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\mu}))\,||\,\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{gt}))$, where $\boldsymbol{\Sigma}_{gt}$ is the ground truth covariance matrix, and $\boldsymbol{\Sigma}(\boldsymbol{\mu})$ is the estimated covariance. Our model obtains significant improvements under all metrics over a diagonal covariance model.

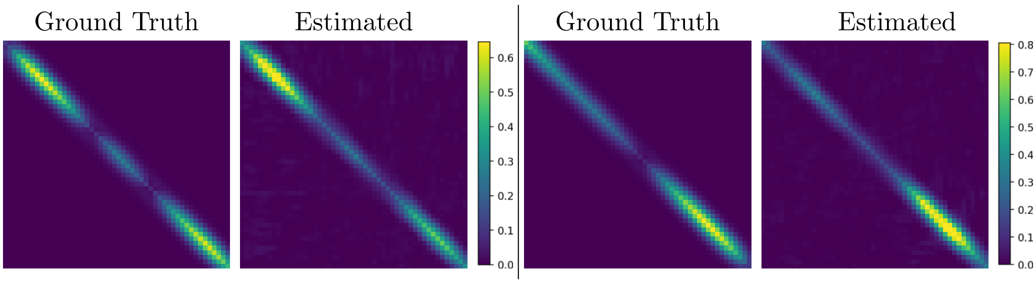


Figure 3-13: Estimated covariance matrices for the spline dataset. Left, the ground truth and estimated covariance matrix for a spline example on the test set, right, the corresponding matrices for a different example. Our model is able to learn the variations in the structured residual distributions.

Quantitative results are presented in Table 3.1. We compare with an uncertainty model that only estimates a diagonal covariance matrix. As the ground truth covariances contain off-diagonal structure, the diagonal covariance model is bound to fail in representing it. Our model achieves a negative log likelihood similar to the one evaluated using the real covariance matrices.

As we have access to the ground truth covariance matrix of each test example, we can directly compare the estimated matrix with the ground truth one. We show qualitative results in Fig. 3-13. Note how the model is able to recover most of the off-diagonal structure in the covariance matrix.

Ellipses

The second synthetic dataset was built to evaluate the covariance prediction network for images and to highlight the limitations of estimating a dense matrix $\mathbf{L}$.

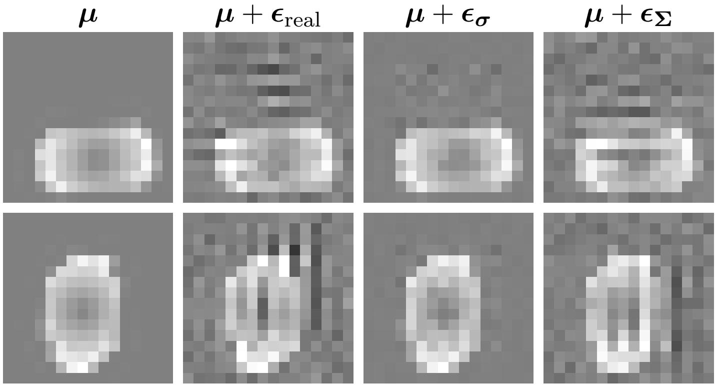

$\boldsymbol{\mu}$
$\boldsymbol{\mu} + \boldsymbol{\epsilon}_{\text{real}}$
$\boldsymbol{\mu} + \boldsymbol{\epsilon}_{\boldsymbol{\sigma}}$
$\boldsymbol{\mu} + \boldsymbol{\epsilon}_{\boldsymbol{\Sigma}}$

Figure 3-14: Reconstructions from the diagonal model and from our approach. From left to right: input $\boldsymbol{\mu}$, original $\mathbf{x}$, and reconstructions using a sample from the learned residual distribution. The diagonal approach, $\boldsymbol{\mu} + \boldsymbol{\epsilon}_{\boldsymbol{\sigma}}$, is limited to produce salt and pepper noise, while our dense covariance method, $\boldsymbol{\mu} + \boldsymbol{\epsilon}_{\boldsymbol{\Sigma}}$, learns how the noise depends on the rotation angle of the ellipse.

We generate a dataset of synthetic grey-scale 16 × 16 images. For each example, the mean image contains an ellipse with random width, height, position and rotation angle. The prototype covariance matrix for this dataset generates lines and is rotated by the same random rotation angle that was used for the ellipse, thus generating random lines that are aligned with the ellipse.

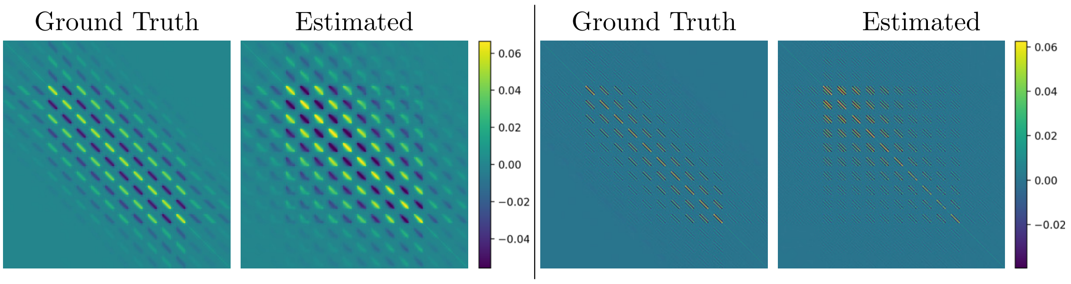
For this dataset, estimating directly a dense Cholesky matrix $\mathbf{L}$ requires 32,896 values per image. Even at this limited image size we were unable to train a dense prediction model, as training was too unstable due to the high number of parameters. Instead, we use the sparse Cholesky model defined in Section 3.3.2, with a neighbourhood of size 5 × 5, which only requires 3,328 values per image. The diagonal covariance baseline has a similar architecture, with a final layer that only predicts the diagonal elements in $\boldsymbol{\Lambda}$. The models are trained for 200 epochs with a learning rate of 1e-3. Additional implementation details are given in appendix A.2.1.

Reconstructions on the test set are shown in Fig. 3-14, where we show results of taking a sample from $\boldsymbol{\Sigma}$ which is added to $\boldsymbol{\mu}$ for a diagonal covariance model and our sparse Cholesky method. The diagonal covariance method fails to model the structured nature of the residuals. In contrast, the covariance prediction network from our approach is successful in mapping the uncertainty distribution from the mean, $\boldsymbol{\mu}$. The samples from this dense covariance matrix exhibit high-frequency detail that matches the true residual.

A quantitative comparison with a diagonal Gaussian model is presented in Table 3.2.

	$-\log p(\mathbf{x} \mid \boldsymbol{\Sigma}(\boldsymbol{\mu}))$	KL	$\lVert\boldsymbol{\Sigma}(\boldsymbol{\mu}) - \boldsymbol{\Sigma}_{gt}\rVert_2$
Ground truth	−286 ± 2.8	–	–
Diagonal model	−149 ± 30.8	743 ± 38.1	1.80 ± 0.21
Ours	**−259 ± 2.7**	**113 ± 2.6**	**1.06 ± 0.34**

Table 3.2: Quantitative comparison on the ellipses dataset (see Table 3.1 for a description of the metrics). Our model is able to better model the real covariance matrices with its more complex uncertainty distribution than a diagonal model.



Ground Truth Estimated Ground Truth Estimated

Figure 3-15: Covariance matrices estimated by our model, where we show the ground truth covariance and the prediction from our model for two examples in the test set. Much of the structure of the real covariance matrices is recovered.

Our model achieves a negative log likelihood similar to the one evaluated using the real covariance matrices. This shows that the method is not over-fitting to the training data, and thus, it is able to generalise to test images.

Examples of ground truth and estimated covariances are shown in Fig. 3-15 and illustrate the accuracy of the covariance prediction network. The covariance structure for this model is more complex than the one for the splines, and yet the model is still able to recover it effectively with our sparse estimation of the Cholesky matrix.

3.4.3 Ablation studies

In this section the efficacy of the different parts of the model, and the assumptions made in this chapter are empirically tested by means of ablation studies. Namely:

1. evaluating models trained on RGB and YCbCr colour spaces,
2. comparing the IPE and SDR models,
3. demonstrating the need for the regularised IPE and SDR approaches.

For each of the aforementioned items, quantitative and qualitative results are presented as required.

Implementation details

All the ablation studies are evaluated using CelebA [Liu et al., 2015], a dataset with real images, where the ground truth covariance matrices for the structured residual distributions are unknown. The aligned and cropped version of the dataset is used, where a further cropping and resizing to 64 × 64 is performed. The dataset consists of 202,599 images of faces, which we split into 182,637 for training and 19,962 for testing as recommended by the authors.

The images are converted to YCbCr colour space, and the Cb and Cr channels are blurred and downsampled to 16 × 16 pixels. The authors stored the images in JPG format, which justifies the downsampling of the Cb and Cr channels, as discussed in Section 3.3.4. For the grey-scale experiments only the Y channel is modelled, and the Cb and Cr channels are discarded. The patch size, n_f, for our covariance prediction for the Y channel is set to 3, which reduces the number of non-zero elements in $\mathbf{L}$ from $8,390,656$ to $20,480$. The models are trained for 110 epochs using a batch size of 64 and a learning rate of 0.0005. For data augmentation we employ simple left-right flips of the images. The VAE architecture from [Pu et al., 2017b] is used for the encoder and decoder networks.

For the SRD model, the hyper-parameter values for the regularisers are $\alpha = 10$ and $\gamma = 0.001$. However, for the first 10 epochs we set $\alpha = 1e4$. Moreover, the network weights are initialised with the weights of a VAE trained with Gaussian likelihood with a spherical covariance, *i.e.* VAE (Sph) is used as pretraining for VAE (SDR). The VAE models with a factorised Gaussian likelihood, VAE (Sph) and VAE (Diag) were introduced in Section 3.4.1. Additional implementation details are given in appendix A.2.2.

Metrics

In Section 1.3 several metrics were discussed for the evaluation of unsupervised generative models. Those metrics are reported on the test set, and two new metrics, the KL and the MSE are included as well:

1. **NLL**: denotes the approximate negative log likelihood $-\log p(\mathbf{x})$, as there is no close form solution for the likelihood, a lower bound is evaluated by numerically integrating over 500 $\mathbf{z}$ samples per image as described in [Burda et al., 2016].

2. **# params**: is the number of learnable parameters in the neural network, which indicates model complexity.

3. **KL**: is the Kullback-Leibler divergence of the approximate posterior, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, to the prior on the latent variables $p(\mathbf{z})$. As samples from the model are drawn from the prior distribution, usually, a high value is indicative of low quality samples.

4. **MSE**: is the mean squared error between the input image and its reconstruction. It is measured using the RGB images before the conversion to YCbCr colour space, and it is computed using only the predicted means, i.e. $\boldsymbol{\epsilon}$ is set to zero. The images are transformed to the range $[0, 255]$ before evaluating the metric.

RGB vs YCbCr

In order to isolate the effects of switching from the RGB to the YCbCr colour space, we train two VAEs with a diagonal covariance Gaussian likelihood, one for each colour space. Reconstruction from these models are shown in Fig. 3-16, which include a sample from the predicted residual distribution. It can be clearly seen that the coloured salt and pepper noise observed in the RGB model, is now mostly replaced by noise in the luminance channel in the YCbCr model. This agrees with our initial findings on the real residual images, which were shown in Fig. 3-6.

IPE vs SDR

In this section, we evaluate both the IPE and the SDR methods on grey-scale images from the CelebA dataset. In order to have a fair comparison between the IPE and the SDR models, we use the VAE trained with the SDR approach for the encoder and decoder networks ($\boldsymbol{\mu}$ and $\mathbf{z}$) that are needed for the IPE method.

Quantitative results for both models are shown in Table 3.3. Although IPE and SDR perform similarly in terms of negative log likelihood, IPE is more complex (in # params) as it requires a separate decoder network for the structured covariance prediction.

In order to avoid needless repetition, qualitative results are not presented here, as they will be shown in subsequent sections for both models. Moreover, as implied by the quantitative analysis, the models are qualitatively indistinguishable from each other.

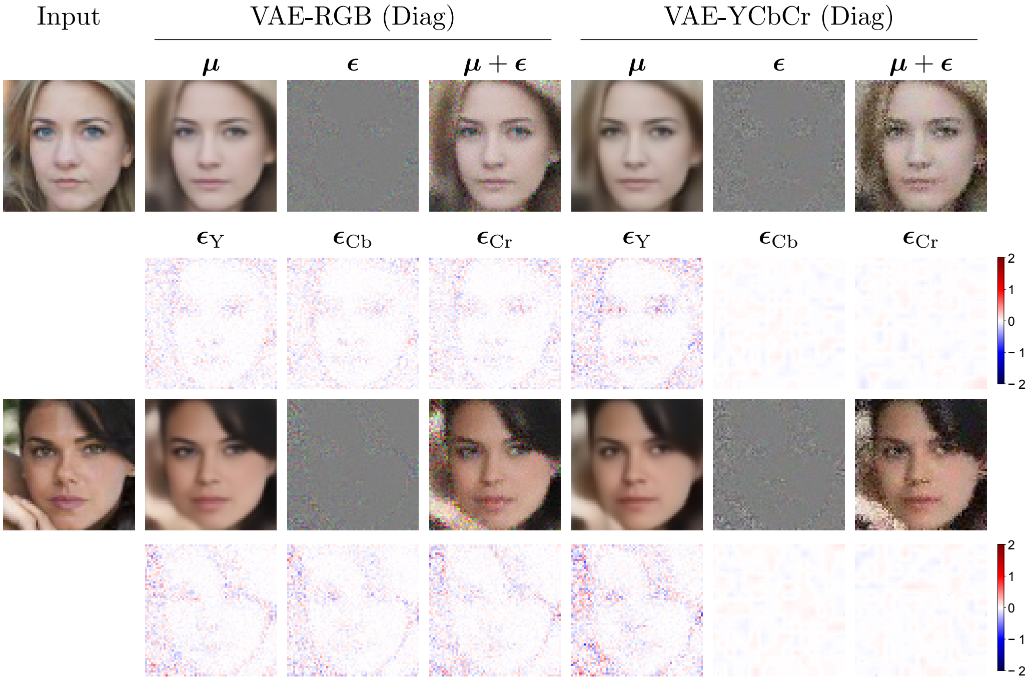

Input
VAE-RGB (Diag)
VAE-YCbCr (Diag)
$\boldsymbol{\mu}$
$\boldsymbol{\epsilon}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\epsilon}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\epsilon}_{\mathrm{Y}}$
$\boldsymbol{\epsilon}_{\mathrm{Cb}}$
$\boldsymbol{\epsilon}_{\mathrm{Cr}}$
$\boldsymbol{\epsilon}_{\mathrm{Y}}$
$\boldsymbol{\epsilon}_{\mathrm{Cb}}$
$\boldsymbol{\epsilon}_{\mathrm{Cr}}$
2
1
0
−1
−2

Figure 3-16: Reconstructions for diagonal covariance Gaussian likelihood VAEs trained on RGB images and on YCbCr images. Below each row, the predicted residuals per channel in YCbCr colour space are shown. Training on the YCbCr colour space allows the model to restrict the residual content to appear mostly on the Y channel, while in RGB salt and pepper noise of similar magnitude is observed in all the channels.

Model	**NLL**	**KL**	**# params**
VAE (IPE)	**–8308 ± 1455**	**269.76**	9.78e6
VAE (SDR)	–8297 ± 1455	**269.76**	**6.72e6**

Table 3.3: Quantitative comparison of density estimation error measured as the negative log likelihood (NLL), KL and # params for the grey-scale CelebA dataset, lower is better. The SDR model is able to achieve a likelihood similar to IPE, with a significant reduction in model complexity.

Naïve training and priors

In this section we demonstrate the need for regularised approaches to learn structured uncertainty in VAE models. We explore several models that were discussed in the methodology section, including: a model without regularisation (VAE ($\boldsymbol{\Sigma}$)), which corresponds to the SDR approach with $\alpha = 0$ and $\gamma = 0$, and VAEs with different prior distributions on the covariance matrix, which also use our sparse Cholesky approximation. We experimented with two priors, a sparse Cholesky-Wishart distribution

Model	**NLL** (with / without prior)	**KL**	**MSE**
VAE ($\mathbf{\Sigma}$)	1.48e6 ± 2.33e2 / **−8.67e3 ± 1.31e3**	236.88	1.20e4 ± 5.00e3
VAE ($\mathbf{\Sigma}$-$\text{Ga}^{\frac{1}{2}}\mathcal{N}$)	**3.28e4 ± 1.79e4** / 3.52e4 ± 1.48e4	221.55	3.82e3 ± 1.55e3
VAE ($\mathbf{\Sigma}$-W^{c}_{sp})	3.29e4 ± 1.79e2 / 3.56e4 ± 1.50e4	**214.61**	3.84e3 ± 1.59e3
VAE (SDR)	1.46e6 ± 4.37e2 / **−8.67e3 ± 1.42e3**	269.76	**2.57e2 ± 1.28e2**

Table 3.4: Quantitative comparison of density estimation error measured as the negative log likelihood (NLL), KL and MSE for the CelebA dataset, lower is better. The NLL is reported with the prior cost on the $\mathbf{L}$ matrices, and without it. For VAE (SDR), without prior refers to the likelihood without the regularisation losses. Although VAE ($\mathbf{\Sigma}$) has a similar likelihood as the SDR variant, this does not translate to qualitative performance, as highlighted by the large gap in MSE, and as shown in Fig. 3-17 and 3-18.

(VAE ($\mathbf{\Sigma}$-W^{c}_{sp})) and a Gamma-Gaussian distribution (VAE ($\mathbf{\Sigma}$-$\text{Ga}^{\frac{1}{2}}\mathcal{N}$)), as discussed in Section 3.3.5. These two models are trained by maximising the objective defined in equation 3.35.

For both prior distributions, equation 3.46 is used to set $\mathbf{V}$, with $p = n + 10$ and $\kappa = 0.09$, where these hyper-parameters are chosen such that the distributions are weakly informative and they have a preference for small magnitude residuals. The real amount of noise in the data is not known, thus, we use the average predicted variance for a VAE (Sph) trained on this dataset to choose the value for κ. The covariance matrix for VAE (Sph) is defined as $\mathbf{\Sigma} = \sigma\mathbf{I}$, and the value of κ is the average σ over the test set. As discussed in Section 3.3.5, the value of κ is the standard deviation of the expected noise. Intuitively, for pixel values in the range $[-1, 1]$, this choice of κ assumes noise of up to 8% of the signal, as a normal distribution with a variance of κ^2 has 95% of the mass between the range $[-0.182, 0.182]$. In other words, the expected noise is approximately ±23 for pixel values in a range of $[0, 255]$.

Quantitative results are shown in Table 3.4, where the negative log likelihood (NLL), the KL divergence to the prior on the latents, and the mean squared error (MSE) are reported. An uninformative Gamma-Gaussian prior on $\mathbf{L}$ is used for the models that were trained without priors, VAE (SDR) and VAE ($\mathbf{\Sigma}$). This prior is added only for the likelihood calculation, and consists of a square root Gamma distribution, as defined in equation A.36, with $a = b = 1e^{-6} \approx 0$ for the diagonal terms, and a Gaussian distribution with zero mean and a standard deviation of $3.4e^{37} \approx \infty$ for the off-diagonal terms.

Training a model without any regularisation, VAE ($\mathbf{\Sigma}$), leads to similar likelihood

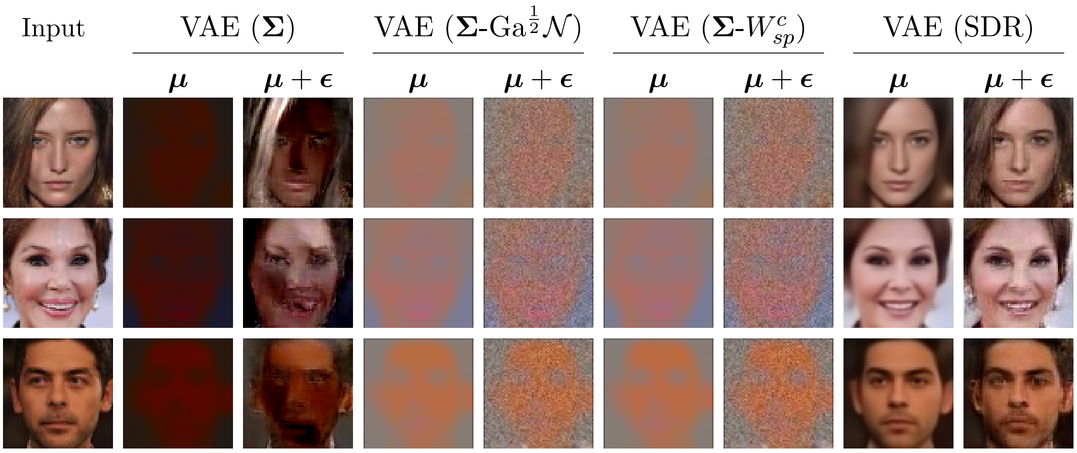

Input
VAE ($\mathbf{\Sigma}$)
VAE ($\mathbf{\Sigma}$-Ga$\frac{1}{2}\mathcal{N}$)
VAE ($\mathbf{\Sigma}$-W^c_{sp})
VAE (SDR)
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$

Figure 3-17: Comparison of image reconstructions for the different models. A model without regularisation, VAE ($\mathbf{\Sigma}$), encodes much of the information in the input image in the structured residual. Employing regularisation with priors, VAE ($\mathbf{\Sigma}$-Ga$\frac{1}{2}\mathcal{N}$) and VAE ($\mathbf{\Sigma}$-W^c_{sp}), leads to unstructured noise and poor means. Only our approach with additional loss functions, VAE (SDR), is able to generate good reconstructions.

(with or without priors) and KL terms as VAE (SDR). However, the VAE ($\mathbf{\Sigma}$) model struggles to generate realistic images, as was shown in Fig. 3-8. The models trained with priors offer a mild improvement in terms of MSE with respect to not using any regularisation. However, they overly concentrate on maximising the prior likelihood of the predicted parameters in the $\mathbf{L}$ matrix. Thus, neglecting the data term during the optimisation. Varying how informative the prior is does not solve the problem, as making it less informative leads to a higher cost per parameter in the $\mathbf{L}$, while more informative also has a higher cost for the values that deviate from $\mathbf{V}$. This experiment also serves to empirically demonstrate how both prior distributions are equivalent, as proved in Section A.1.6 in the appendix.

Qualitative results are shown in Fig. 3-17 and 3-18, which display reconstructions and samples respectively. These results highlight the advantages of using our regularised approach, as all models with or without prior regularisation fail to reliably produce good quality images with high frequency details. In the model trained without any regularisation, VAE ($\mathbf{\Sigma}$), in order to achieve a high likelihood, the Gaussian has a choice between modelling data on the means (with low variance) or on the covariance (with very high covariance terms). In practice, we observe the later, with means, $\boldsymbol{\mu}$, that have little image content, and the data being modelled mostly as stochastic residuals. Employing a sparse Cholesky-Wishart or a Gamma-Gaussian prior leads to covariance matrices that are too strongly diagonal, without a corresponding improvement on the

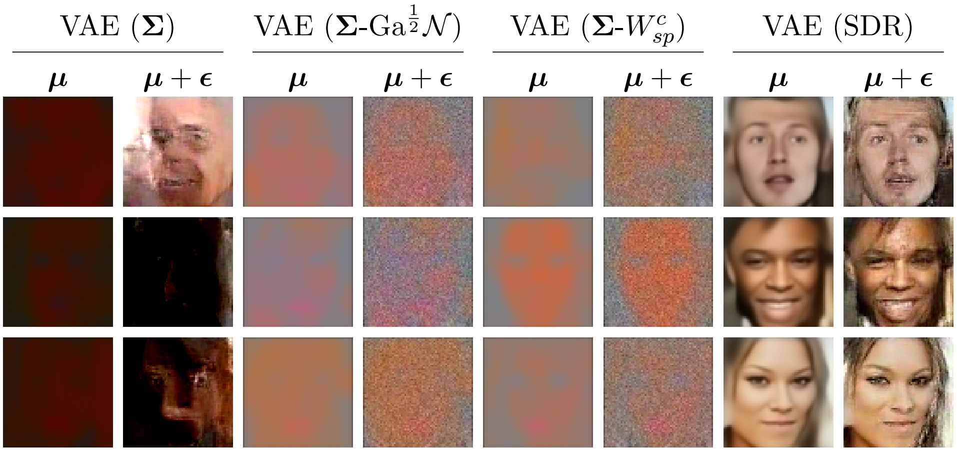

VAE ($\boldsymbol{\Sigma}$)
VAE ($\boldsymbol{\Sigma}$-Ga$\frac{1}{2}\mathcal{N}$)
VAE ($\boldsymbol{\Sigma}$-W_{sp}^{c})
VAE (SDR)
$\boldsymbol{\mu}$
$\boldsymbol{\mu} + \boldsymbol{\epsilon}$

Figure 3-18: Comparison of samples generated by the different models. Not employing regularisation, VAE ($\boldsymbol{\Sigma}$), leads to modelling the image distribution in the structured residual. Employing regularisation with priors, VAE ($\boldsymbol{\Sigma}$-Ga$\frac{1}{2}\mathcal{N}$) and VAE ($\boldsymbol{\Sigma}$-W_{sp}^{c}), leads to unstructured noise and poor means. Only our approach with additional loss functions, VAE (SDR), is able to generate good samples.

means, $\boldsymbol{\mu}$.

These experiments have been carried out with the SDR approach; however, they also serve to demonstrate the need for the IPE approach. As it was demonstrated above, both methods produce similar results. The difference is that in the SDR approach, the regularisation occurs in the form of explicit losses, while in the IPE method, the regularisation emerges from the training procedure.

3.4.4 Comparison to previous work

In this section we compare the SDR approach to previous methods, which include different covariance factorisations for Gaussian likelihood VAEs, and the β-VAE approach, as discussed in Section 3.4.1. The SDR approach is chosen as it was previously demonstrated that the SDR and IPE methods perform similarly. Yet, the SDR approach offers significant savings in terms of number of parameters in the neural network, and it allows for the encoder and decoder networks to be trained end-to-end.

The models are trained on two datasets in YCbCr colour space, CelebA [Liu et al., 2015] and LSUN [Yu et al., 2015]. As previously discussed, CelebA is a relatively simple dataset of aligned images of human faces. The LSUN dataset provides a harder

Model	**NLL**	**KL**	**# params**
VAE (Sph)	-4517 ± 1308	322.94	6.68e6
VAE (Diag)	-4598 ± 2747	341.92	**6.67e6**
β-VAE (Diag)	-5574 ± 1083	**199.16**	**6.67e6**
VAE (SDR)	$\mathbf{-8669 \pm 1424}$	281.91	6.70e6

Table 3.5: Quantitative comparison of density estimation error measured as the negative log likelihood (NLL), KL and # params for the YCbCr CelebA dataset, lower is better. The SDR model outperforms previous methods in terms of NLL, with a similar parameter budget. β-VAE obtains a lower KL divergence, yet, the samples produced by our model are of similar quality as theirs, as shown in Fig. 3-20.

challenge, as it contains unaligned images of outdoor churches.[6]

CelebA

Quantitative results for models trained on YCbCr images are shown in Table 3.5. For β-VAE we report the negative log likelihood (NLL) after setting $\beta = 1$, *i.e.* after removing the regulariser term. Our method achieves significantly lower likelihood than competing methods with a similar parameter budget.

The KL divergence between the approximate posterior and the prior shows that jointly learning a correlated residual distribution is also beneficial in order to achieve a latent distribution that better matches the prior distribution. However, the KL divergence can be a misleading indicator of sample quality. Experiments employing a lower β weight for the β-VAE produced worse quality samples, with a KL divergence that was still lower than in our SDR model.

β-VAE may be interpreted as adding a correction factor in the KL divergence that is meant to compensate for the factorised Gaussian assumption. In contrast, in our model the data is well represented, which leads to a latent space that produces good samples without the need to add a correction factor. However, a disadvantage of our model is the addition of the α and γ hyper-parameters in equation 3.51.

Reconstructions from the models are shown in Fig. 3-19. The means, $\boldsymbol{\mu}$, obtained with our model are comparable to competing methods. The samples, $\boldsymbol{\epsilon}$, from the learned residual distribution add unrealistic noise in the image for VAE (Sph). The diagonal

[6] The dataset contains other scene categories that we will not use.

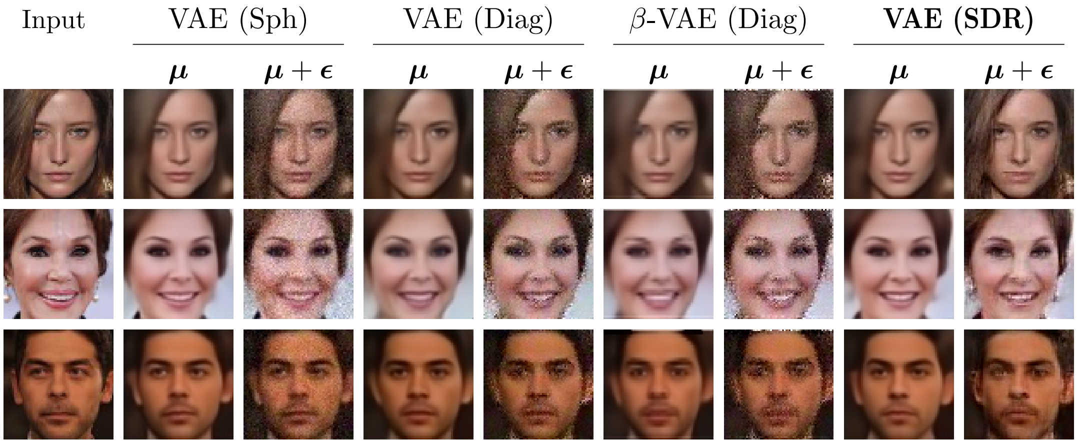

Input
VAE (Sph)
VAE (Diag)
β-VAE (Diag)
VAE (SDR)
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$

Figure 3-19: Comparison of image reconstructions for the different models. In contrast to previous work, our model is able to learn structured residuals. This structured residuals add plausible high-frequency content to the means, $\boldsymbol{\mu}$, such as hair details in the first row, or tooth details in the second row.

model of VAE (Diag) and β-VAE (Diag) is able to improve on this, and it does not include unrealistic noise in areas that are modelled well, such as the white background on the lower-right side of the image in the second row. In contrast to previous work, the residuals from our model add plausible details, like hair.

Samples from all the models are shown in Fig. 3-20. Overfitting to the data term can be observed in VAE (Sph) and more so in VAE (Diag), as denoted by the high value in the KL divergence shown in Table 3.5. Consequently, this leads to a latent space that does not follow the prior distribution and thus to the poor samples observed. β-VAE is able to produce samples that are of similar quality to its reconstructions. Our method is able to produce good quality samples, where the covariance network is again able to model plausible high frequency details, unlike competing methods.

LSUN

In this section we analyse the SDR approach on the LSUN [Yu et al., 2015] dataset, where we train the models only on the *church outdoors* category. There are $126,227$ training images, and 300 for validation for this category. As the test data is not available we use the validation set instead. We found it beneficial to increase the number of training epochs to 150, as this dataset is more complex than CelebA. All other hyper-parameters are carried over from the CelebA experiments in the previous section.

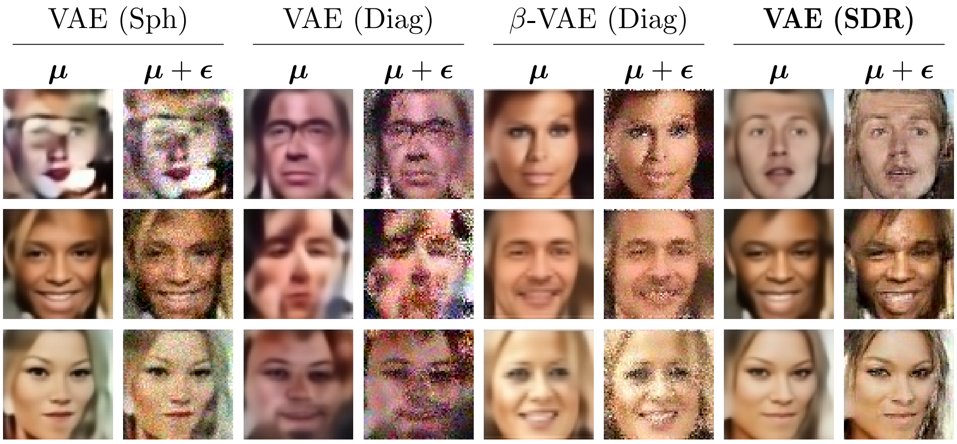


Figure 3-20: Samples for all models, where our method is the only able to include high-frequency details such as wrinkles. Neither the means, $\boldsymbol{\mu}$, nor the residuals, $\boldsymbol{\epsilon}$, generated by VAE (Sph) and VAE (Diag) are realistic. VAE (SDR) not only generates realistic residuals, but also improves the quality of the sampled means, $\boldsymbol{\mu}$, over the VAE (Sph) model, which is used for pretraining.

Model	**NLL**	**KL**	**MSE**
VAE (Sph)	–2440 ± 1176	303.83	**566 ± 285**
VAE (Diag)	–4464 ± 1924	331.06	728 ± 327
β-VAE (Diag)	–4213 ± 2055	**205.88**	800 ± 354
VAE (SDR)	**–6918 ± 2423**	313.92	614 ± 286

Table 3.6: Quantitative comparison of density estimation error measured as the negative log likelihood (NLL), KL and MSE for the LSUN dataset, lower is better. Our model is able to achieve a significant improvement in likelihood in comparison with VAEs with factorised noise models.

Quantitative results for reconstructions are presented in Table 3.6, where we measure the mean squared error (MSE) with respect to the input, as well as the negative log likelihood (NLL). A VAE with a spherical covariance is able to achieve the lowest reconstruction cost. However, as previously discussed, real samples from this model include unrealistic salt and pepper noise. Our model is able to outperform competing methods in terms of marginal log likelihood.

Reconstructions are shown in Fig. 3-21, where we find again that our model is able to produce better means than competing methods, with the exception of a VAE with spherical covariance. However, the residuals modelled by a spherical covariance VAE are quite limited, as they are forced to have high levels of noise throughout the image, including areas which are trivial to model, like flat regions of the sky. Our structured

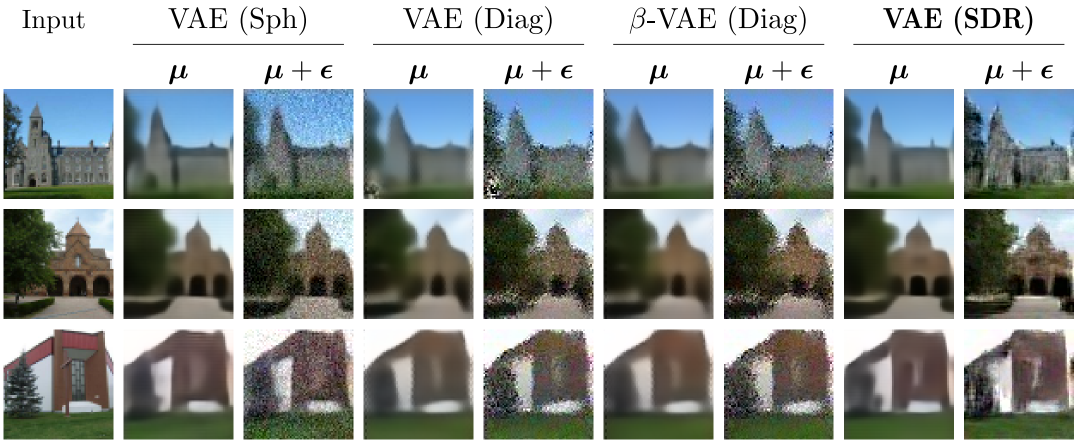

Input
VAE (Sph)
VAE (Diag)
β-VAE (Diag)
VAE (SDR)
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$

Figure 3-21: Comparison of image reconstructions for the different models. Our model is able to improve over the noisy images generated by previous work. However, the increased complexity of this dataset has a detrimental effect on the quality of the images generated by all the models.

residuals add fine detail, however as this dataset is more complex than CelebA, all the models struggle to reconstruct the input. A factor that plays a role in the complexity on this dataset is the lack of alignment between the images, as it is not obvious how to align churches.

Samples from the models are shown in Fig. 3-22, where we find that the VAE with diagonal covariance struggles to generate anything meaningful. β-VAE is able to generate recognizable shapes, which are significantly blurry, while our model is able to produce means similar to VAE (Sph) with better noise samples.

3.4.5 Model insights

In this section we show additional results that provide further insights into our model. The VAE (SDR) model trained on the CelebA dataset that was used in the previous section is employed in all the experiments, unless otherwise stated.

Multiple residual samples In order to highlight the stochastic nature of the residuals, in Fig. 3-23 several residual images sampled from the same covariance matrix are shown, *i.e.* the covariance matrix is kept fixed for each input image. It can be seen how there is significant variability between the samples, covering different hair positions

[7] https://garoe.github.io/thesis/supplement/animated_edits.html

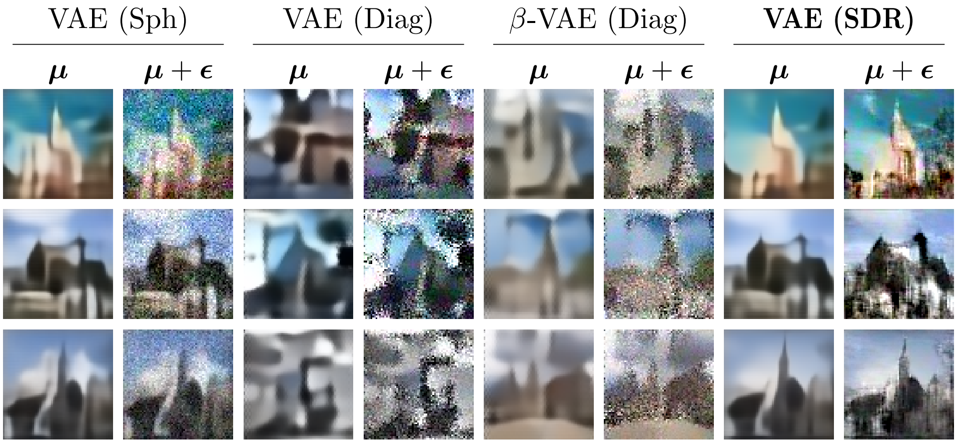


Figure 3-22: Samples drawn from the models. VAE (Diag) fails to learn a useful latent space and β-VAE produces blurrier images than our model. The images produced by our approach contain structured high-frequency content, still, all the models are unable to generate realistic images.

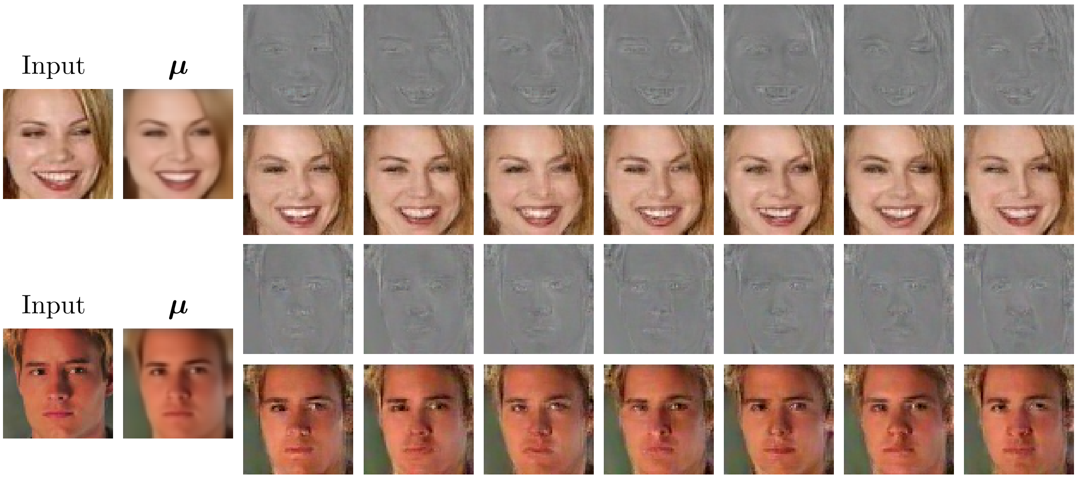


Figure 3-23: Variability of the residual images for reconstructions using the VAE (SDR) model. In the first row, each column shows a residual sample from the same covariance matrix, and those are added to the mean, μ, in the second row. Variations of hair position or teeth shape can be observed. Additional results can be found in the supplemental video[7].

or teeth variations. This demonstrates that the estimated covariance matrices span a space of plausible, yet, distinct residual images.

Latent space interpolations To evaluate the generalisation of the model to different regions in the latent space, we show in Fig. 3-24 the result of interpolating between

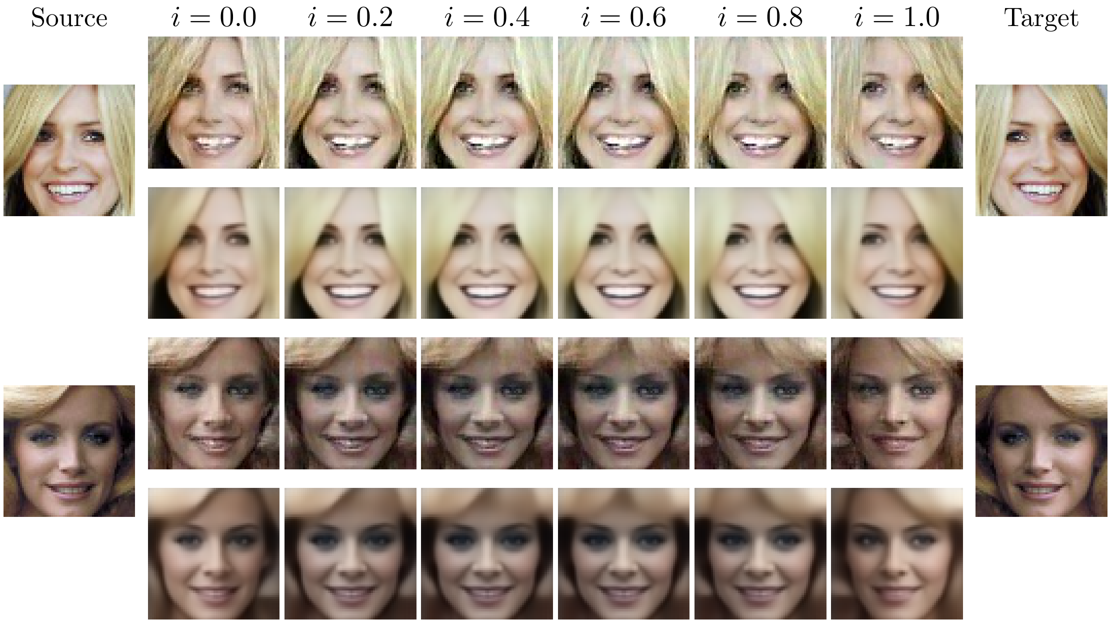


Figure 3-24: Samples drawn with our model while interpolating on the latent space, from the source to target. The i value on each column corresponds to equation 3.54. For every pair of rows, the first contains the sample from the model, $\boldsymbol{\mu} + \boldsymbol{\epsilon}$, and the second contains the mean, $\boldsymbol{\mu}$.

an image and its x-flipped mirror image. The latent values, $\mathbf{z}$, between the image pair are interpolated using polar interpolation[8]:

$$\mathbf{z}_i = \mathbf{z}_s\sqrt{1-i} + \mathbf{z}_t\sqrt{i}, \tag{3.54}$$

where $\mathbf{z}_i$ is the latent code for the i^{th} step, $\mathbf{z}_s$ is the latent code of the source image, $\mathbf{z}_t$ is the latent code of the target image, and i is an interpolation factor in the range $[0, 1]$. Using a fixed noise vector $\mathbf{u}$, a single sample is drawn from our model for each interpolation step as $\mathbf{x}_i = \boldsymbol{\mu}_i + \mathbf{M}_i\mathbf{u}$, where the mean, $\boldsymbol{\mu}_i$, and the covariance, $\boldsymbol{\Sigma}_i = \mathbf{M}_i\mathbf{M}_i^T$, are estimated from the corresponding $\mathbf{z}_i$. The generated images are plausible, and the sampled residuals are consistent across the interpolated images.

Variance maps To further highlight the differences between the diagonal and dense covariance models, a variance map for the Y channel is shown in Fig. 3-25 for both mod-

[8] Polar interpolation leads to better results than linear interpolation. During model learning and inference, the latent vectors, $\mathbf{z}$, have roughly the same norm. However, linear interpolation tends to produce vectors with significantly smaller norm, which usually correspond to parts of the latent space that are not well modeled by the decoder. A more in depth discussion of this issue can be found in [Huszar, 2017].

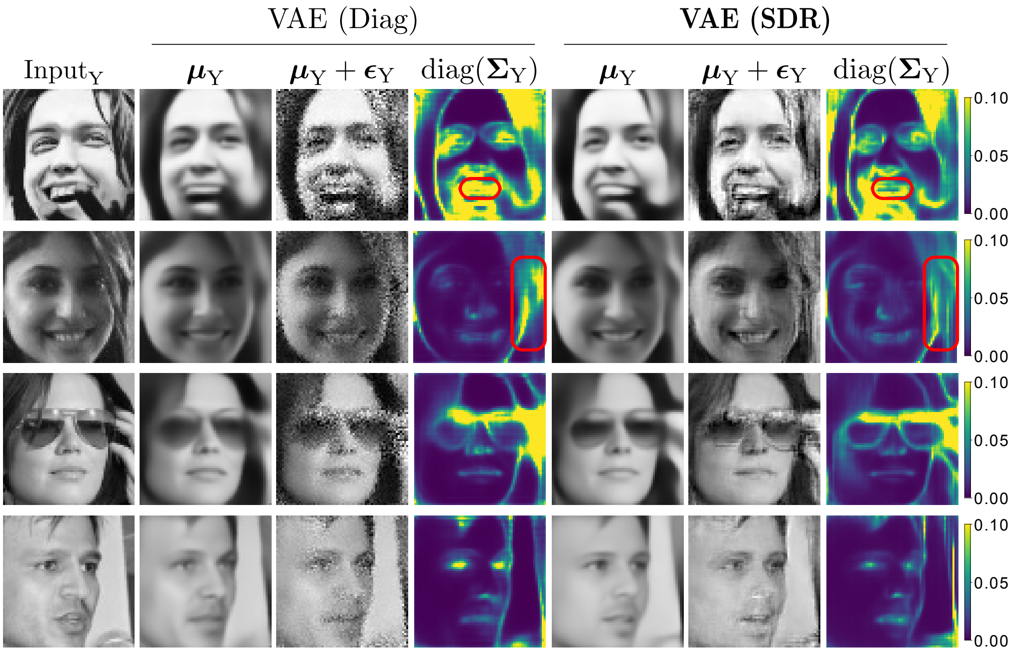

VAE (Diag)
VAE (SDR)
Input$_{\mathrm{Y}}$
$\boldsymbol{\mu}_{\mathrm{Y}}$
$\boldsymbol{\mu}_{\mathrm{Y}} + \boldsymbol{\epsilon}_{\mathrm{Y}}$
diag($\boldsymbol{\Sigma}_{\mathrm{Y}}$)
0.10
0.05
0.00

Figure 3-25: Variance maps for the Y channel in a VAE with diagonal covariance Gaussian likelihood and in a VAE employing the SDR approach for a dense covariance matrix. The diagonal noise estimation model mistakenly identifies teeth (red rounded rectangle in the first row) or hair (red rounded rectangle in the second row) as high variance regions, whereas our covariance model properly identifies them as regions with high covariance, yet lower variance.

els. The variance is defined as the diagonal part of the predicted covariance matrix. The VAE (Diag) model must explain all the errors with variance, while VAE (SDR) is able to explain some with correlations. The effect of this is evident when sampling residuals from the estimated covariance matrix, and when analysing the high variance areas in the predicted residual distribution. For example, the diagonal model characterises hair and teeth areas as high variance regions, while the dense covariance approach models those regions with high covariance and lower variance.

Restricted vs unrestricted input features for the covariance network We now demonstrate that the IPE approach is able to learn structured residual distributions from an unrestricted latent space. For this experiment, we train both an Autoencoder (AE) [Cottrell et al., 1987] and a VAE (Diag) model. The AE is trained with a mean squared error, and learns an unrestricted latent space, **z**, which is used as input for the covariance network. The latent space learned by the VAE (Diag) is approximately Gaussian distributed, due to the isotropic Gaussian prior that is used

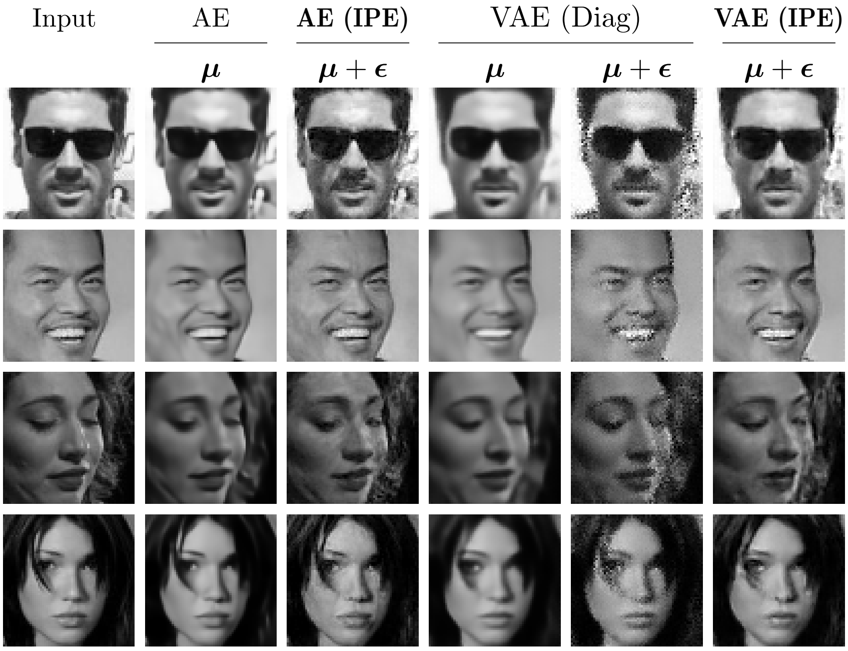

Input
AE
AE (IPE)
VAE (Diag)
VAE (IPE)
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$

Figure 3-26: Comparison of image reconstructions when applying our IPE model to an AE and a VAE (Diag). For both the AE and VAE, our IPE model adds plausible high-frequencies from a single sample drawn from the predicted uncertainty distribution. The distribution of the latent features in a VAE (Diag) is approximately Gaussian, while the AE latent space distribution is unrestricted. Thus, demonstrating that structured residuals can be learned from the unrestricted latent space learned by the AE.

in the model.

For this experiment we train the models using grey-scale images, and the neighbourhood size in $\mathbf{L}$ is increased to 7×7. The covariance networks are trained with a learning rate of 1e-3 for 50 epochs. Additional implementation details are given in appendix A.2.1.

Example reconstructions for the models are shown in Fig. 3-26. The means, $\boldsymbol{\mu}$, produced by the AE are sharper than those produced by the VAE (Diag), yet they still lack high-frequency details. The residuals learned by our model appears to be of the same quality, regardless of the shape of the latent space, *i.e.* Gaussian distributed for the VAE (Diag) and unrestricted for the AE. These residuals add plausible high-frequency details, like teeth and hair, to the means produced by the VAE (Diag) and the AE models.

A quantitative comparison for the models is presented in Table 3.7, where $-\log p(\mathbf{x} \,|\, \mathbf{z})$ is reported. This metric is used instead of the NLL, as the NLL cannot be tractably evaluated for the AE and the AE (IPE) models. This occurs because the AE is not a probabilistic model, as it does not have an explicit distribution over the latent space,

Model	$-\log p(\mathbf{x} \mid \mathbf{z})$
VAE (Diag)	–6079 ± 936
AE (IPE)	–8242 ± 433
VAE (IPE)	**–8386 ± 1339**

Table 3.7: Quantitative comparison of density estimation error, for VAE (Diag), AE (IPE) and VAE (IPE), lower is better. Our IPE approach achieves a similar conditional likelihood for both VAE and AE. Thus, demonstrating that there is no loss in performance when using the unrestricted latent space from an AE as input for the covariance network.

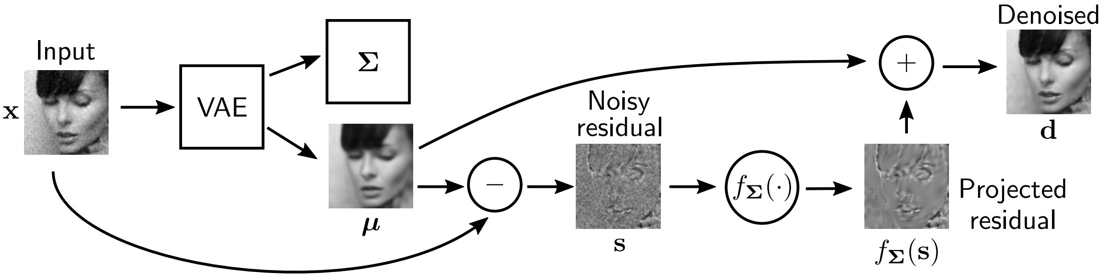


Figure 3-27: Overview of the proposed denoising technique with our covariance prediction network. The noisy input, $\mathbf{x}$, is reconstructed, generating $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$. A noisy residual, $\mathbf{s} = \mathbf{x} - \boldsymbol{\mu}$, is projected on the space spanned by the predicted covariance, producing a clean residual, $f_{\mathbf{\Sigma}}(s)$. This projected residual is added to the mean, $f_{\mathbf{\Sigma}}(s) + \boldsymbol{\mu}$, to generate the denoised image, $\mathbf{d}$. The projection step, $f_{\mathbf{\Sigma}}(s)$, is explained in the text.

$\mathbf{z}$. For the $-\log p(\mathbf{x} \mid \mathbf{z})$ evaluation in VAE (Diag) and VAE (IPE), we simply take the mean vector predicted by the encoder for $\mathbf{z}$. This is intended to replicate the situation with the AE. Our model performs similarly regardless of the model that is used for the means, VAE (Diag) or AE. Thus, demonstrating that a structured residual distribution can be learned from the unrestricted latent space of an AE.

3.4.6 Application: denoising

One possible application of the covariance prediction network is image denoising, which is demonstrated using the IPE method with a VAE (Diag) from the previous section. This model was trained with grey-scale images on the CelebA dataset. We hypothesise that our predicted covariance matrices will only span the space of valid face residuals, thus projecting a noisy residual in that space will remove the noise.

An overview of our denoising method is shown in Fig. 3-27. Here, the noisy image

Model	**MSE**	**PSNR**	**SSIM**
DAE	0.005 ± 0.003	28.89 ± 1.69	0.90 ± 0.03
Ours	**0.003** ± **0.001**	**31.38** ± **0.92**	**0.92** ± **0.02**

Table 3.8: Quantitative comparison for denoising in terms of mean squared error (MSE, lower is better), peak signal-to-noise ratio (PSNR, higher is better) and SSIM (higher is better) with respect to the noise-free input. Our model is on average able to produce better results than an Autoencoder trained for denoising.

is reconstructed using the VAE, which was *not* trained with noisy data. Thus, $\boldsymbol{\mu}$ is a denoised image, which lacks high-frequency details. The residual between the noisy input and the reconstruction is computed, and projected onto $\hat{\mathbf{\Sigma}}$, which is created by taking the n_{v} eigenvectors of $\mathbf{\Sigma}$ with the largest eigenvalues. This projection step is denoted as $f_{\mathbf{\Sigma}}(\mathbf{s})$ in the figure, and it is defined as $f_{\mathbf{\Sigma}}(\mathbf{s}) = \mathbf{s}\hat{\mathbf{Q}}\hat{\mathbf{Q}}^{\intercal}$, where $\hat{\mathbf{Q}}$ is a $n \times n_{\text{v}}$ matrix with the eigenvectors. The projected residual is added to the VAE reconstruction, $\boldsymbol{\mu}$, to produce the final denoised image.

The number of eigenvectors acts a control over how aggressive the denoising process is, where using a small number removes more noise, yet, it also discards useful high-frequency content. For this experiment we use $n_{\text{v}} = 1000$, where the hyper-parameter was manually chosen such that the images appear realistic after denoising.

As a baseline we train an Autoencoder explicitly for denosing (DAE) using a mean squared error loss, and with the same architecture as the VAE. The noisy images are constructed by adding salt and pepper noise to the noise-free images. The noise is sampled from $\mathcal{N}\left(0, \left(\frac{0.25}{3}\right)^2\right)$.

Quantitative results for this experiment are shown in Table 3.8, where mean squared error (MSE), the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) [Wang et al., 2004] is reported for the first 2000 images in the test set. Our approach achieves lower error than the DAE, which was trained specifically for this task.

Qualitative results from our model and the DAE are shown in Fig. 3-28. Note how the use of the structure covariance model is able to filter the noise to generate plausible structured high-frequency details, while the DAE fails to recover those details.

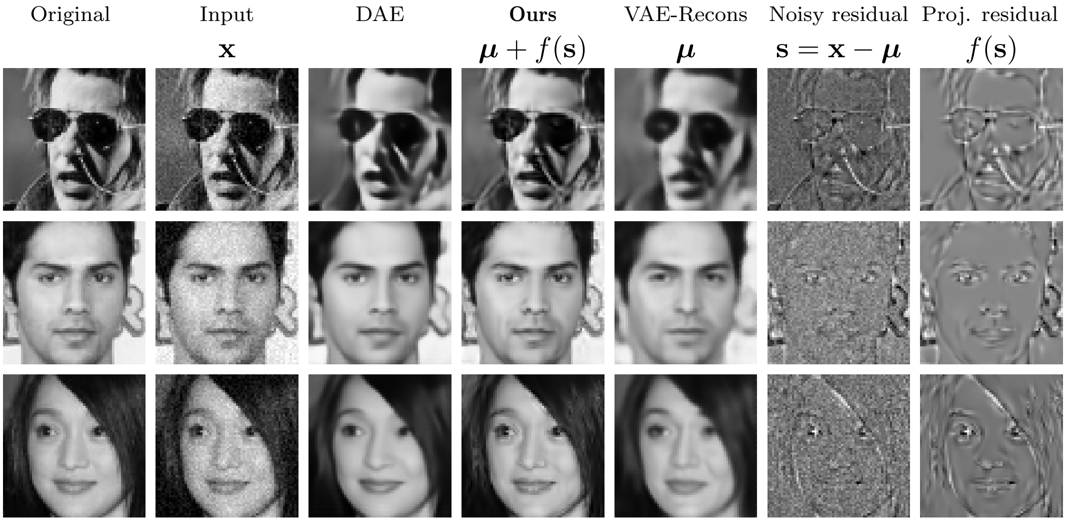


Figure 3-28: Denoising experiment, left column: original image without noise, second column: image with added noise, third column: denoising autoencoder (DAE) result, fourth column: our result, fifth column: VAE reconstruction from the noisy input, sixth column: residual between the VAE reconstruction and the noisy input, right column: the noisy residual projected on $\hat{\boldsymbol{\Sigma}}$, the matrix constructed with 1000 eigenvectors of $\boldsymbol{\Sigma}$. Our result is the sum of the projected residual and the VAE reconstruction. Our model is able to recover fine details that are lost with the DAE approach.

3.5 Discussion

In this chapter, we have proposed two approaches for training a deep neural network to predict dense covariance matrices for the residual image distribution of unseen image reconstructions. Our results show that ground truth covariances can be learned for toy data, and samples of better quality than previous work can be generated for CelebA [Liu et al., 2015] and LSUN [Yu et al., 2015] data. Additionally, we have shown a practical application of this approach for denoising face images.

We have demonstrated how to endow Variational Autoencoders (VAE) with a structured likelihood model. Our results show that VAEs can be successfully trained to predict structured output uncertainty, subject to additional regularisation terms, and that such models have similar mean reconstructions than those obtained with a factorised likelihood model. Moreover, we have shown that under a complex enough likelihood distribution, VAEs are able to generate images with high frequency details.

There are still open questions, such as: what is the best input data to use for the covariance network? Is it best to learn directly from $\mathbf{z}$ or $\boldsymbol{\mu}$ or some other pre-learned function of either of these? Is it possible to substitute the regularisation losses used in VAE (SDR) for a more principled approach, such as automatic relevance determination techniques [Mackay, 1995] with hierarchical priors? Could this approach be extended to obtain better quality results on complex datasets like LSUN?

Finally, we have only examined the residual image distribution for reconstruction models trained with a Gaussian likelihood. An interesting avenue to explore would be a structured pixel uncertainty distribution for other reconstruction error metrics, such as perceptual [Hou et al., 2017] or adversarial losses [Larsen et al., 2016, Huang et al., 2018]. This might help with complex datasets where the current model struggles, such as LSUN. Additionally, the approach might also be applied for models that use the mean squared error for tasks other than reconstructing, such as segmentation or depth estimation.

Chapter 4

Warping GAN

4.1 Introduction

In the previous chapter an unsupervised generative method based on Variational Autoencoders [Rezende et al., 2014, Kingma and Welling, 2014] was discussed. The novelty of the method was the inclusion of a tractable approach for modelling structured uncertainty. The model was able to provide high-level editing by interpolating in the latent space between different images. However, most deep learning models, including the one presented in the previous chapter, are limited to operate on the image resolution it was trained with.

This chapter describes a model for face editing that predicts smooth geometric deformations (warping) of the input image. As the deformations are smooth, they can be easily upsampled and applied at arbitrary resolutions. Thus, overcoming the weaknesses of the model described in the previous chapter, as long as the editing operation is amenable to be modelled by warps. Moreover, the technique is based on StarGAN [Choi et al., 2018], one of the image-to-image translation models discussed in Section 2.6, which allows the model to be trained without explicit regression targets.

4.1.1 Motivation

We are interested in photo-realistic image editing, in which deep learning image-to-image translation methods have shown promising results [Zhu et al., 2017, Choi et al., 2018, Pumarola et al., 2018]. As discussed in Chapter 1, we also focus on methods that

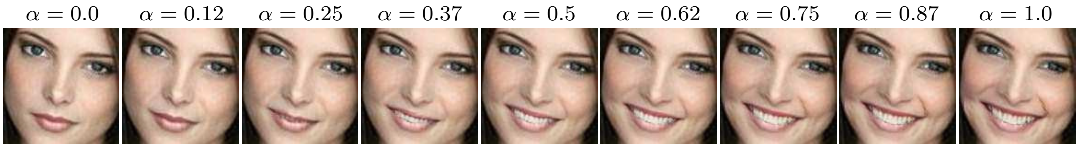


Figure 4-1: Example of partial edits on the GANimation [Pumarola et al., 2018] model. The input image, $\alpha = 0.0$, is progressively edited to match the facial action units [Ekman and Friesen, 1976] of a smiling face, $\alpha = 1.0$.[1]

provide a simple interface for users to edit images, *i.e.* a single control per semantic characteristic, as this makes it easier for novice users.

Most deep learning editing methods [Choi et al., 2018, Dekel et al., 2018, Portenier et al., 2018, Pumarola et al., 2018, Geng et al., 2018, Yan et al., 2016], including the aforementioned image-to-image translation methods, predict the pixel values of an edited image directly. Example edits from these methods are shown in Chapter 1 in Fig. 1-5. A consequence of this approach is that these methods are limited to only being effective on images that have a similar resolution to the training data. A further disadvantage of current methods is the difficulty of applying a partial or exaggerated edit, as shown in Fig. 4-1, as opposed to a binary attribute change. For this to be possible, an extensive collection of soft attribute data is required. In this example, the annotation process consisted of assigning a soft score from zero to one indicating the strength of a number of facial action units [Ekman and Friesen, 1976]. This is labour intensive, and at test time each intermediate value requires another forward pass of the network, creating increased computational expense [Pumarola et al., 2018].

Recently, some interesting approaches that allow edits at arbitrary resolutions have been proposed. They proceed by estimating the edits at a fixed resolution and then applying them to images at a higher resolution. The types of possible edits are restricted to either warping [Yeh et al., 2016] or local linear colour transforms [Gharbi et al., 2017]. However, these approaches are limited by requiring paired data, *i.e.* for each source image in the dataset, they need the corresponding edited image. For example the paired data approach by Yeh *et al.* [Yeh et al., 2016] employs images of the same subject under different expressions and camera poses. This data was collected with an expensive setup with multiple synchronised cameras, and with laboratory controlled illumination.

In contrast, many pixel-based methods [Choi et al., 2018, Dekel et al., 2018, Portenier et al., 2018, Pumarola et al., 2018] do not need paired data for training. This is of

[1] Image courtesy of [Pumarola et al., 2018].

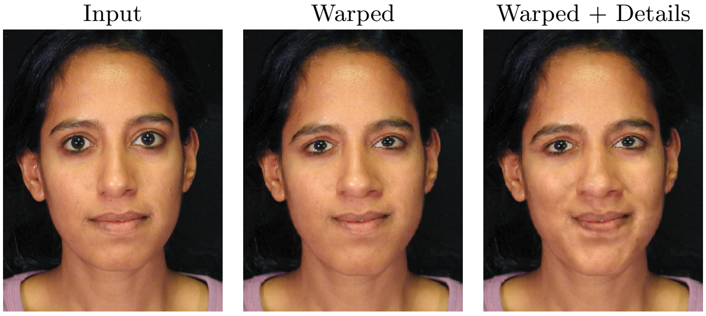


Figure 4-2: Example of expression editing, where a neutral expression face, input image, is edited to a smiling expression. If warping alone is used, the edit appears too subtle. Adding details like shadows and wrinkles improves the quality of the edit.[2]

particular importance, as collecting paired data is significantly more expensive and time consuming than collecting unpaired data. Some methods [Choi et al., 2018, Pumarola et al., 2018] achieve this by extending the Cycle-GAN [Zhu et al., 2017] approach, which was described in Section 2.6. While others [Dekel et al., 2018, Portenier et al., 2018] automatically produce paired data in a self supervised manner. This regularises learning enough, such that applying an adversarial loss to the outputs during training is enough to ensure good quality results.

An interesting combination of the aforementioned methods would be a model that allows partial semantic editing at high-resolution without requiring paired data. Such a model can be used for an editing system where each attribute is associated with a slider. Thus, providing simple yet powerful controls over the image editing process, while being usable at a wide range of resolutions.

4.1.2 Proposed solution

In this chapter, we introduce an approach to learn warp fields for semantic image editing without the requirement of paired training data samples. This is achieved by exploiting recent approaches for learning edits from unpaired data with cycle-consistency checks. Our proposed model uses a similar framework to StarGAN [Choi et al., 2018] to predict warp fields that model the requested edits. As the predicted warp fields are smooth, they can be trivially upsampled and applied at high resolutions.

An evident disadvantage of the proposed model is that there are clear limitations to the types of edits possible through warping. Namely, warping can only model deformations

[2] Images courtesy of [Liu et al., 2001].

in the object shape. Thus, edits that require colour transformations or synthesising new content are not possible. For instance, changing the hair colour of a person from blonde to brunette is unattainable with this method.

Furthermore, for transformations amenable to be characterised in this way, the edit might still not be fully representable. For example, moderate changes in face expressions have been shown to be modelled well by warping [Geng et al., 2018]. However, details like changes in illumination and appearance, such as wrinkles, are not captured well [Liu et al., 2001], as shown in Fig. 4-2.

We argue that, for the changes that *can* be described by warping, there are several benefits of doing so. The advantages of our proposed model with respect to pixel translation models can be summarised as:

1. Smooth warp fields can be upsampled and applied to higher resolution images with a minimal loss of fidelity. This is opposed to upsampling images, which commonly results in the loss of high frequency details. We show plausible edits produced by warp fields upscaled by up to a 30× factor of the resolution they were estimated at.

2. Modelling only geometric transformations makes it easier to add priors to regularise against unrealistic edits. For example, it is easy to restrict the deformations to be locally consistent.

3. Warp fields lead to a model that is better at preserving a subject's identity. This is demonstrated quantitatively by measuring an identity score based on a face re-identification network.

4. Warp fields are more interpretable than pixelwise differences, which are the transformations produced by most pixel-based approaches. Particularly with respect to identifying potentially erroneous or unrealistic edits. We illustrate this by providing maps showing the degree of local stretching or squashing.

5. Warp fields are more suited to allow partial edits than pixel-based approaches. In other words, the model can produce partial edits while being trained only with binary labelled data. We demonstrate the simplest implementation of this by scaling the warp field using a single scalar parameter. Interpolation between the input image and the complete edit is demonstrated by varying the scalar from zero to one, while values larger than one demonstrate extrapolation, *i.e.* exaggerated edits. We qualitatively show that edits produced in this way are plausible.

An additional contribution presented in this chapter is to improve the specificity of editing attributes in StarGAN-based models. We have observed that when these models are trained with several binary attributes, they can transform more than one attribute of the image, even if only a single attribute edit was intended. This is caused by the model having no indication of the attributes that should be edited, only of the final expected attribute labels. For example, when enlarging the nose of a subject that has a slight smile, the model will not only make the nose bigger, but also make the smile more pronounced. In order to overcome this limitation, we propose to transform the binary labels used for each attribute to inform the model of which attributes should be edited, and which should remain fixed. This produces only the expected changes, and it does not require any extra attribute annotation. Moreover, it removes the need to rely on attribute classifiers during inference.

4.2 Previous work

Modern image-to-image translation methods can be trained with unpaired data, in particular, models based on the Cycle-GAN [Zhu et al., 2017] approach, as discussed in Section 2.6. This is a key component for our method, and as such we choose to build upon StarGAN [Choi et al., 2018], a state of the art model for image-to-image translation, which was described in the same section.

4.2.1 StarGAN

An overview of the characteristics of the model discussed in this chapter compared to previous methods is shown in Table 4.1. As highlighted in the table, StarGAN operates on interpretable and easy to collect labels for the attributes, as well as running at interactive speeds. However, it cannot operate at arbitrary resolutions, nor can it perform partial edits when trained on binary labelled data. Thus, we will discuss in more detail several deep learning models that focus on generating images at high resolutions, some of which support partial edits.

4.2.2 Methods for high resolution

Methods for editing images at high resolution can be divided in two categories: (i) methods that use intermediate representations that are designed to upsample well to

Method	Unpaired Data	High Resolution	Forward Pass	Partial edits
Pix2Pix [Isola et al., 2017]			✓	
CycleGAN [Zhu et al., 2017]	✓		✓	
StarGAN [Choi et al., 2018]	✓		✓	
FaceShop [Portenier et al., 2018]	✓		✓	
FlowVAE [Yeh et al., 2016]		✓	✓	✓
CWF [Ganin et al., 2016]		$\sim$	✓	$\sim$
DBL [Gharbi et al., 2017]		✓	✓	✓
iGAN [Zhu et al., 2016]	✓	$\sim$		✓
DFI [Upchurch et al., 2017]	✓	$\sim$		✓
Ours	✓	✓	✓	✓

Table 4.1: Compared to previous work, our method is the only one that is able to edit high-resolution images in a forward pass of the network, without the need for paired data during training. The model also allows to perform partial edits of the input image. The symbol $\sim$ denotes partial fulfilment of a criterion.

arbitrary resolutions, and (ii) methods that directly predict pixel values at high resolutions.

Methods designed for upsampling

These approaches are based on predicting intermediate representations that are relatively agnostic to image resolution; *e.g.* warp fields, local colour affine transformations or blendshape weights.

Warp fields, if sufficiently smooth, can be predicted at a lower resolution, upsampled and applied at high resolution with minimal loss of accuracy. Previous methods have applied them to redirecting eye gaze [Ganin et al., 2016], editing emotional expressions [Yeh et al., 2016] and synthesising objects in novel views [Zhou et al., 2016]. By scaling the predicted warp field, partial edits are supported by these methods. However, these models require paired training data. Additionally, in the eye gaze redirection approach [Ganin et al., 2016] a supplementary CNN is used after warping to directly edit the pixels in the output image, which limits the upsampling effectiveness of the whole model.

Spatial Transformer GANs [Lin et al., 2018] predict a series of global affine deformations that can be estimated at a lower resolution and applied at arbitrary resolutions. The model is designed to automatically compose a foreground image into a background one, where the foreground image is aligned to the background one using the global transformations. This method is limited in that it only supports a global affine transformation to a foreground image, which is then pasted on top of a background image. Thus, using this method for face editing would require an infeasibly large dataset of suitable face parts to use as foreground images in order to preserve identity.

Local data dependent linear transformations of pixel intensities have been estimated from low resolution images to model the effect of image enhancement filters [Chen et al., 2016], and this was demonstrated to upsample well to high resolution images at test time. The method requires a pair of low-resolution images, before and after the edit, to fit the local transformations. An extension where the transformations are predicted by a deep network, was proposed by Gharbi *et al*. [Gharbi et al., 2017]. To perform a partial edit, a linear interpolation can be evaluated between the identity matrix and the estimated linear transformations. Paired data is needed for both methods, and it was generated by editing pictures using Photoshop filters and other image enhancement tools. Therefore, these methods appear to be more suited for image enhancement.

While the previously discussed methods directly predict the intermediate representations, Zhu *et al*. [Zhu et al., 2016] train a low-resolution GAN and then fit a dense warp field and local affine colour transformation between an input image and a GAN generated image at low resolution, 64×64. The generated image is created by reconstructing a rough edited input image using the GAN generator, where an encoder is trained to project input images to the generator latent space. The optimisation to find the warp field and the local transformations does not have a closed form solution. Thus, the authors employ an iterative method that fits either the warps or the local transformations on each step. Partial editing is supported as the method uses warp fields and local affine colour transforms. However, the generator is unaware that these restricted transformations will be fitted to its outputs, as it is trained with the standard GAN framework. Therefore, its capacity is potentially wasted in learning how to generate images that lead to edits that are not representable by such transformations.

Blendshape weights have also been used as an intermediate representation to edit expressions in the context of video reenactment [Thies et al., 2016]. This method uses a handcrafted metric to compute the expression mapping in terms of blendshape weights. An extension where the blendshape weights of a face were transformed by a Cycle-GAN has been explored [Ma and Deng, 2018]. Similar to our approach, the blendshape weights are resolution independent, and partial edits can be achieved by smoothly varying the blendshape weights. However, these methods require several input video frames to reconstruct the face for the blendshape model.

Direct prediction at high resolution

A number of techniques have been proposed in order to scale deep image synthesis methods to larger image resolutions. These include, synthesising images in a pyramid of increasing resolutions [Denton et al., 2015], employing fully convolutional networks trained on patches [Ledig et al., 2017], and directly in full resolution using multi-GPU training with small batch sizes and long training times [Brock et al., 2019, Karras et al., 2018] (approx 2 weeks for 512×512 and 1 month for 1024×1024). Image editing applications using the aforementioned methods have been explored [Ignatov et al., 2017, Portenier et al., 2018]. However, direct or pyramid-based approaches do not scale well beyond modest resolutions, and training on patches assumes that global image information is not needed for the edit. Moreover, these methods are only applicable at similar resolutions as the training data that is was used to learn the model parameters.

A gradient descent based method for editing the image has also been explored [Upchurch et al., 2017]. The deep features of a pre-trained classifier network are extracted for an input image. The features of neighbouring images containing the desired attribute are also extracted. Both features are linearly interpolated, and the edited image is produced by modifying the input using a gradient descent method until it encodes to the interpolated features. Partial edits can be achieved by varying the interpolation rate between the deep features. However, this approach starts failing when the resolution of the test image differs significantly from the input data. Moreover its applicability is limited, as generating a 1000×1000 image takes approximately 2 minutes on a high end desktop GPU.

StarGAN

As our model builds upon StarGAN [Choi et al., 2018], we define the method in more detail in this section. In order to train a StarGAN model a dataset of labelled images is needed. A single image is denoted by $\mathbf{x}$, and each image has a set of associated domains represented as a binary vector $\mathbf{c}$. As we focus on semantic face editing, we use semantic attributes to refer to the domains encoded in $\mathbf{c}$. The model employs three networks, a generator $G(\cdot)$, a discriminator $D(\cdot)$, and a classifier $C(\cdot)$. The generator,

$G(\mathbf{x}, \bar{\mathbf{c}})$, transforms $\mathbf{x}$ to match the target attributes indicated by $\bar{\mathbf{c}} \sim p(\mathbf{c})$, where $p(\mathbf{c})$ is the empirical distribution of binary attribute vectors, $\mathbf{c}$. The model is trained with:

i. an adversarial loss:

$$L_{\text{adv}} = \mathbb{E}_{\mathbf{x}} \left[\log(D(\mathbf{x})) \right] + \mathbb{E}_{\mathbf{x},\bar{\mathbf{c}}} \left[\log(1 - D(G(\mathbf{x}, \bar{\mathbf{c}}))) \right], \tag{4.1}$$

ii. a cycle consistency loss:

$$L_{\text{cycle}} = \mathbb{E}_{(\mathbf{x},\mathbf{c}),\bar{\mathbf{c}}} \left[\|\mathbf{x} - G(G(\mathbf{x}, \bar{\mathbf{c}}), \mathbf{c})\|_1 \right], \tag{4.2}$$

iii. and attribute classification losses:

$$\begin{aligned} L^d_{\text{cls}} &= \mathbb{E}_{(\mathbf{x},\mathbf{c})} \left[-\log(C(\mathbf{x}, \mathbf{c})) \right] && (4.3) \\ L^g_{\text{cls}} &= \mathbb{E}_{\mathbf{x},\bar{\mathbf{c}}} \left[-\log(C(G(\mathbf{x}, \bar{\mathbf{c}}), \bar{\mathbf{c}})) \right], && (4.4) \end{aligned}$$

where $C(\mathbf{x}, \mathbf{c})$ is a discriminative function that outputs the probability that $\mathbf{x}$ has associated attributes $\mathbf{c}$. In the expectations, the data is sampled in pairs, when an input image, $\mathbf{x}$, is sampled, its corresponding attribute vector, $\mathbf{c}$, is also taken, and this is indicated as $\mathbb{E}_{(\mathbf{x},\mathbf{c})}[\cdot]$.

The classifier learns using the training set (eq. 4.3) and it also ensures the translated image matches the target attributes (eq. 4.4). The adversarial loss (eq. 4.1) ensures that the generated images are realistic, while the cycle consistency loss (eq. 4.2) is used to avoid the trivial solution where the generator collapses to producing a single image per attribute regardless of the input.

In practice due to the stability issues discussed in Section 2.5.1, in StarGAN the adversarial loss (eq. 4.1) is substituted by the Wassertein distance objective (eq. 2.28):

$$L_{\text{adv}} = \mathbb{E}_{\mathbf{x}} \left[D(\mathbf{x}) \right] - \mathbb{E}_{\mathbf{x},\bar{\mathbf{c}}} \left[D(G(\mathbf{x}, \bar{\mathbf{c}})) \right], \tag{4.5}$$

and the Wasserstein gradient penalty term in eq. 2.30 is added to constrain the discriminator to be a 1-Lipschitz continuous function.

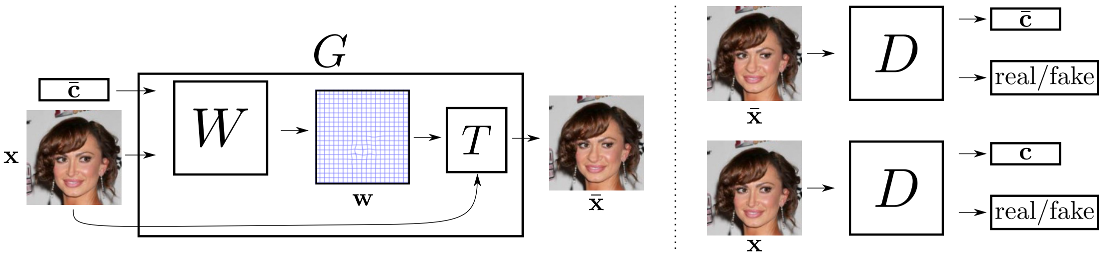


Figure 4-3: Overview of our model, which consists of a generator, G, and a discriminator, D. The generator contains a warping network, W, and a warping operator, T. The inputs to the warping network are an RGB image, $\mathbf{x}$, and a target attribute vector, $\bar{\mathbf{c}}$. The output is a dense warp field, $\mathbf{w}$, which is used by T to deform the input image and produce the output image, $\bar{\mathbf{x}}$. The discriminator evaluates both the input image, $\mathbf{x}$, and the generated image, $\bar{\mathbf{x}}$, for realism and the presence of attributes that agree with the labels. In this example, the only change between $\bar{\mathbf{c}}$ and $\mathbf{c}$ is the label for the attribute "big nose".

4.3 Methodology

We employ the StarGAN [Choi et al., 2018] framework as the basis for our model, and use the notation introduced in the previous section. We modify the generator such that the set of transformations is restricted to non-linear warps of the input image:

$$G(\mathbf{x}, \bar{\mathbf{c}}) = T(\mathbf{x}, W(\mathbf{x}, \bar{\mathbf{c}})), \tag{4.6}$$

where $W(\mathbf{x}, \bar{\mathbf{c}}) = \mathbf{w}$ is a parametric function that generates the warp parameters. T is a predefined warping function that applies a warp to an image. W is chosen to be a neural network. An overview of our system is shown in Fig. 4-3.

4.3.1 Warp Parametrisations

The non-linear warp fields, $\mathbf{w}$, can be parametrised in a number of different ways. Two possible approaches are landmark-based and dense warps.

Landmark-based For landmark-based methods, displacements are defined only at a sparse set of landmarks on the object to be deformed. A smooth dense warp field can then be constructed through the use of interpolation techniques, such as thin-plate splines [Bookstein, 1989] or mesh grids [Wolberg, 1998]. Such a parametrisation has the advantage of predicting a reduced parameter set, making the model easier to train. However, this comes at the cost of reduced deformation flexibility, as the number of

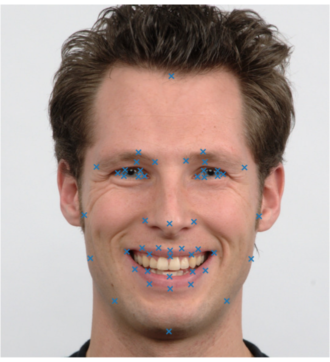


Figure 4-4: An example of typical landmark locations for face images. In particular, 49 face points that are detected, which are indicated by the blue crosses.

landmarks is usually much sparser than the number of pixels in the image, as shown in Fig. 4-4. Additionally, this approach relies on accurate and robust landmark finding.

Dense warps A displacement vector at each pixel defines a dense warp, which allows for arbitrary deformations of the input image. This gives flexibility at the cost of model complexity. Such an approach by itself gives no guarantees on the smoothness of the warp field, which is crucial for application at arbitrary resolutions. In order to ensure smoothness, some regularisation terms must be employed.

Block-based warps Another option consists of defining a shared displacement vector for a number of pixels in a neighbourhood. For example, a displacement vector for every patch of four pixels in the image. This is suitable for settings where defining a displacement vector per pixel is not desirable, and landmark information is not available or applicable for the given data. Hence, the number of parameters to be predicted is significantly reduced with respect to the aforementioned dense approach.

Constrained dense warps The flexibility of dense warps can be problematic. As a result, several approaches to constrain the dense warps have been explored. For example, invertibility is a desirable property that has been enforced by means of velocity field parametrisations [Ceritoglu et al., 2013], which ensure that the warp transformations are diffeomorphic. An approach to avoid warps which would produce image folding has also been proposed [Shu et al., 2018], by parametrising the warps by their gradients and constraining the gradient values to be positive. In this parametrisation, the displacements can be obtained by numerical integration of the gradients.

Discussion We choose to use dense warps, given their additional flexibility over the landmark-based approach, which was found to be too restrictive in preliminary experiments. The reduced number of parameters to be predicted in the block-based approach would likely regularise the learning task for the neural network. However, this would come at the cost of reduced performance, as the model would lose fine control over the estimated warp fields. We empirically found that the block-based and the dense constrained approaches were not needed for our purposes, as the regularisation techniques described in the following section proved to be sufficient. However, it is interesting to notice that training the network with low resolution images and applying the warps to higher resolution images during inference, as will be explained in Section 4.3.4, is analogous to learning low resolution block-based warps from high resolution images.

4.3.2 Learning

We use the same adversarial loss (eq. 4.5), gradient penalty loss (eq. 2.30) and attribute classification loss (eq. 4.3) as StarGAN.

Warp cycle loss The cycle consistency loss (eq. 4.2) is modified to produce warp fields that are inverse consistent, i.e. the composition of the forward and backward transformations yields an identity transformation:

$$L_{\text{cycle}} = \mathbb{E}_{(\mathbf{x},\mathbf{c}),\bar{\mathbf{c}}}\left[\|T(T(\mathbf{A},\mathbf{w}),\bar{\mathbf{w}})) - \mathbf{A}\|_2^2\right], \tag{4.7}$$

where $\bar{\mathbf{w}} = W(G(\mathbf{x},\bar{\mathbf{c}}),\mathbf{c})$ is a warp field that reverts the initial transformation, and $\mathbf{A}$ is a two channel image where each pixel takes the value of its coordinates. This loss provides additional guidance to the network in terms of dual learning [Shen and Liu, 2017]. Also, it is more informative than equation 4.2, as a pixel loss provides no information for warps inside constant colour regions.

Smoothness loss The generator network estimates an independent deformation per pixel. Consequently, there are no guarantees that the learned warps will be smooth. Smooth warps are desirable as they can be effectively upsampled to arbitrary resolutions. Therefore, an $L2$ penalty on the gradients of the warp field is added to encourage smoothness. In practice a finite-difference approximation is used as

$$L_{\mathrm{s}} = \mathbb{E}_{\mathbf{x},\bar{\mathbf{c}}} \left[\frac{1}{n} \sum_{(i,j)\in P} \|w_{i+1,j} - w_{i,j}\|_2^2 + \|w_{i,j+1} - w_{i,j}\|_2^2 \right], \tag{4.8}$$

where P is the set of all pixels in the warp field, n is cardinality of P, and $w_{i,j}$ is the displacement vector at pixel coordinates (i, j).

Additional regularisation Due to the lack of paired data, it has been observed that image-to-image translation methods easily find undesirable correlations in the data. Previous work [Mejjati et al., 2018, Pumarola et al., 2018] addressed this by employing unsupervised attention mechanisms, which restrict the extent of the edits. However, we did not observe this behaviour in our models. This is probably due to the restriction of editing only by means of geometric deformations.

An additional issue with image-to-image translation models is the embedding of information in the form of imperceptible high-frequency content in the edited images. This is described in more detail in Section 2.6. Again, our model is unaffected by this, due to only being able to manipulate the image via geometric deformations. Therefore, a further advantage of our model with respect to pixel-based methods is that we do not need to employ the triple consistency loss defined in equation 2.33. Qualitative results demonstrating this are shown in Section 4.4.

Complete objective The joint losses for the discriminator and the generator are defined as

$$L_D = -L_{\mathrm{adv}} + \lambda_{\mathrm{gp}} L_{\mathrm{gp}} + \lambda_{\mathrm{cls}} L_{\mathrm{cls}}^d, \tag{4.9}$$

$$L_G = L_{\mathrm{adv}} + \lambda_{\mathrm{cls}} L_{\mathrm{cls}}^g + \lambda_{\mathrm{cycle}} L_{\mathrm{cycle}} + \lambda_{\mathrm{s}} L_{\mathrm{s}}, \tag{4.10}$$

where L_D is the discriminator loss, L_G is the generator loss and λ_{cls}, λ_{gp}, λ_{cycle} and λ_{s} are hyper-parameters that control the relative strength of each loss.

4.3.3 Signed labels

As mentioned in Section 2.6, a StarGAN type model can change unexpected parts of the image when editing specific attributes. At inference time, the classifier is used to infer the labels for all of the attributes. Then, these labels can either be changed or

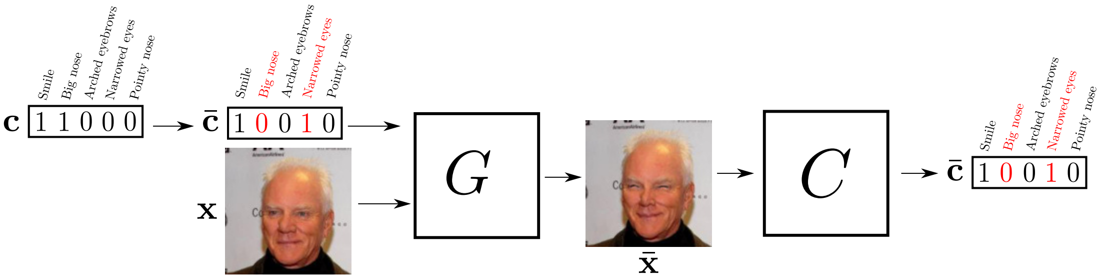


Figure 4-5: Use of binary attribute labels in a StarGAN model during training. The edited attribute labels, $\bar{\mathbf{c}}$, are constructed by copying the values from the input attribute labels, $\mathbf{c}$, and changing the labels of one or more attributes to a target value. In this case, the edited attributes are “big nose” and “pointy nose”, which are highlighted in red. The generator and the classifier cannot distinguish between the edited and the copied attributes, which encourages all attributes to be magnified. Especially noticeable is the accentuation of the smile.

directly copied to the target vector, depending on the desired edits. This means that the model cannot distinguish between the edited attributes and the copied ones, as shown in Fig. 4-5. Thus, the model tends to accentuate the copied attributes, since this increases the chances of being classified correctly by the discriminator.

In order to address this drawback, we propose to explicitly instruct the generator on which attributes should be edited, as shown in Fig. 4-6. The target labels used as input for the generator are transformed to contain three values, $[-1, 0, 1]$, where -1 indicates that the attribute should be reversed, 0 that it should remain unaffected, and 1 that it should be added. We name these new target labels as signed labels, as they contain positive and negative values.

This approach has two distinct benefits:

1. it leads to more localised edits, as the network is able to distinguish attributes that should remain fixed from attributes that should be edited,

2. it removes the need for a classifier network during inference, as the unedited entries in the signed target labels can be set to zero.

The classifier loss for the generator (initially defined in eq. 4.4) is modified to only penalize the attributes that should be edited:

$$L^{g}_{\text{cls}} = \mathbb{E}_{\mathbf{x},\hat{\mathbf{c}}}\left[-h\sum_{i=0}^{r-1} |\hat{c}_i| \log\left(C(G(\mathbf{x},\hat{\mathbf{c}}),\bar{c}_i)\right)\right], \tag{4.11}$$

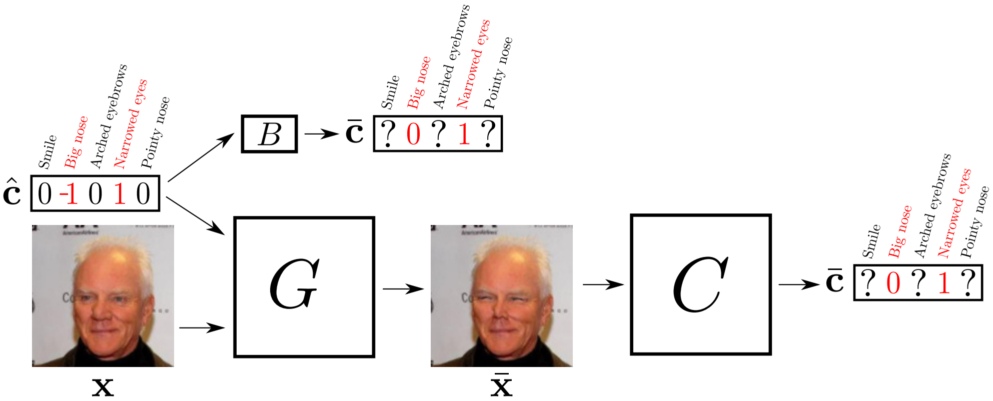


Figure 4-6: Use of signed attribute labels during training. The attributes that should remain fixed are set to 0 in the signed labels, $\hat{\mathbf{c}}$, and the ones that should change are set to their target value of 1 or -1. In this example, the target attributes are "big nose" and "pointy nose", which are highlighted in red. A label operator, $B(\cdot)$, which is described in the text, converts the signed labels in $\hat{\mathbf{c}}$, to binary labels in $\bar{\mathbf{c}}$. There is no loss associated with the labels that were not changed, represented by a "?" in $\bar{\mathbf{c}}$.

where $\hat{\mathbf{c}}$ are the signed labels, r is the number of attributes, and $h = r/\|\hat{\mathbf{c}}\|_1$ is a normalisation factor, which ensures that there is no bias with respect to the number of edited attributes. During training, the signed target label for each attribute, $\hat{c}_i$, is sampled independently from a Categorical distribution with probabilities $[0.25, 0.5, 0.25]$. A label operator, $\bar{c}_i = B(\hat{c}_i)$, is used to transform the signed labels to binary labels, which is defined as

$$B(\hat{c}_i) = \begin{cases} 0, & \text{if } \hat{c}_i = -1. \\ 1, & \text{if } \hat{c}_i = 1. \end{cases} \tag{4.12}$$

For $\hat{c}_i = 0$, the function does not need to be defined, as the classification loss for those attributes is zero by construction.

The classification loss in equation 4.11 is used for images with several not mutually exclusive binary attributes, and equation 4.4 is used otherwise.

4.3.4 Inference

Once the model parameters have been optimised, an input image, $\mathbf{x}$, of arbitrary size can be edited in a single forward pass of the network. The process is fully automated, as the user only needs to specify the semantic attribute that they would like to edit. For models that do not use the signed labels, the input attribute labels need to be inferred by the classifier, as the labels for all attributes are needed for the generator as shown

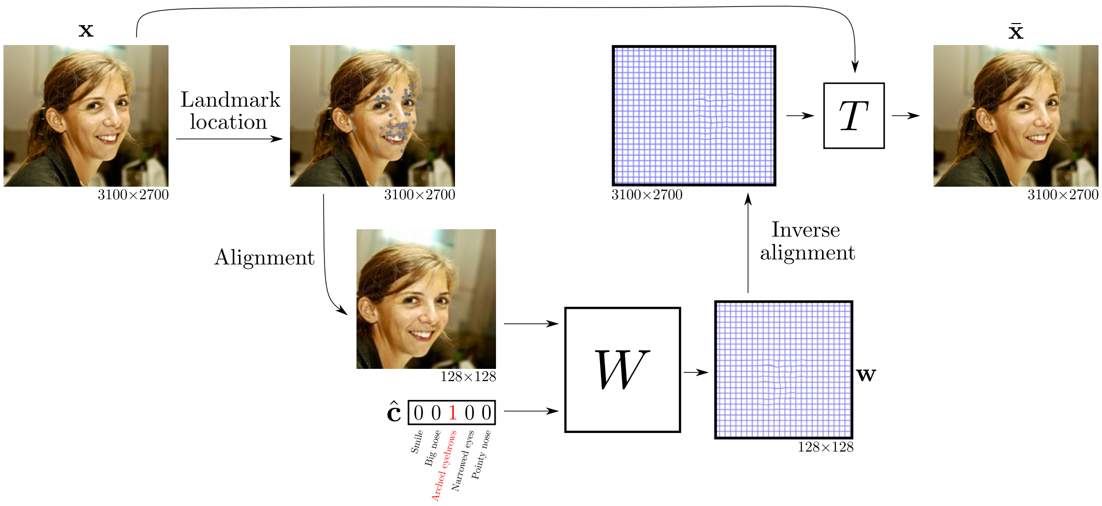


Figure 4-7: Overview of inference at arbitrary resolutions with our model. The resolution is shown below each image. The input is a single RGB image, $\mathbf{x}$, with an arbitrary resolution. Automatic landmark detection is performed, and the detected landmarks are used to align the image with the train data. One or more labels are set, in this example "arched eyebrows", to create the target attribute vector, $\hat{\mathbf{c}}$, with the signed labels. With this information, the warp network, W, produces a low resolution warp, $\mathbf{w}$, which is resized using the inverse alignment transformation. This resized warp field is used to deform the original input image, generating the final result, $\bar{\mathbf{x}}$, at the original resolution.

in Fig. 4-5. The inference process with signed labels is illustrated in Fig. 4-7, which also includes an alignment step specific for faces that is explained in Section 4.4.1.

In a more general setting, where object landmarks are not available, inference proceeds as follows. First, the input image is resized to match the resolution of the training data, and the labels in $\hat{\mathbf{c}}$, are set according to the target edit. Then, the resized image and attribute vector with the target labels are fed into the warping network, which produces a suitable warp field, $\mathbf{w}$. The warp field displacement vectors are rescaled and resampled to the original image resolution. Lastly, the original image is warped using the resampled warp field to produce the final edited image.

Partial edits

Another advantage of our model is that once a warp field has been computed for an input image, partial edits can be applied by simply scaling the predicted displacement vectors by a scalar, α. Formally, this is defined as

$$G(\mathbf{x}, \hat{\mathbf{c}}, \alpha) = T(\mathbf{x}, \alpha\mathbf{w}) = \bar{\mathbf{x}}, \tag{4.13}$$

where $\mathbf{x}$ is the input image, $\mathbf{w} = W(\mathbf{x}, \hat{\mathbf{c}})$ the predicted warp field, and $\bar{\mathbf{x}}$ is the partial edited image. This is a cheap operation as it does not require to run the network for each new value of α, as the displacement field, $\mathbf{w}$, is reused. For this reason, our method allows for edits to be performed at interactive speeds, in contrast with previous methods that allow for partial edits [Pumarola et al., 2018]. However, the edits are not trivially composable, *i.e.* only a single attribute may be edited is this way, as editing a second attribute requires composing the warps from both edits. This operation, if done naïvely, would lead to unrealistic warps.

4.4 Results

In this section we evaluate the two contributions described in the section above. First, we demonstrate editing at arbitrary resolutions by warping with an image-to-image translation method trained with unpaired data. Including learning how to perform partial edits from binary labelled data, and how to interpret the edits by providing maps showing squashing and stretching due to the warp fields. Second, we demonstrate the advantages of employing signed labels to avoid accentuating attributes that should remain fixed.

4.4.1 Datasets

We evaluate our method and baselines quantitatively and qualitatively on two face datasets, CelebA [Liu et al., 2015] and RafD [Langner et al., 2010].

Face alignment We make use of face landmark locations to align and resize the images to 128×128 using a global affine transformation, at both training and test time. Prediction of face landmark locations is a well studied problem, and there are several off-the-shelf methods [King, 2009] to locate landmark points in a face. In particular, we use an internal face landmark detection network to extract 49 points per face, as shown in Fig. 4-4. Details of alignment at training time are provided below for each dataset. At test time, the inverse of the aforementioned global affine transformation is used to transform the warp fields, as shown in Fig. 4-7. The warp is then applied

Figure 4-8: Examples of the train images found on the CelebA dataset.

directly to the original image. This is in contrast with previous methods, that would edit the aligned image and then warp the edited image to the original space.

CelebA The CelebA dataset [Liu et al., 2015] contains 202,599 images of faces and we use the train/test split recommended by the authors. We use the aligned and cropped version of the dataset, where the images have a size of 178×218. A few examples of the images used for training are shown in Fig. 4-8. The faces are centre-cropped to 178×178, and resized to 128×128, following the protocol in Choi *et al.* [Choi et al., 2018]. The face landmarks provided by the authors are limited to 5 points per face, thus we employ our internal detection method during inference to extract the points for aligning images that are not in the test set. Importantly, from the 40 binary attributes provided, we choose the ones more amenable to be characterised by warping, namely: smiling, big nose, arched eyebrows, narrow eyes and pointy nose.

RafD The RafD dataset [Langner et al., 2010] contains images of 67 subjects in 8 expressions, as shown in Fig. 4-9. For each expression, the subjects were recorded from 5 camera angles and with 3 different eye gaze directions. We discarded the two most extreme camera angles, leaving a total of 4,824 images. Contrary to previous work [Choi et al., 2018, Pumarola et al., 2018], which kept images from the same subject on the train and test set, we reserve all images of subjects 58, 63, 64, 71 and 72 as test data. Face landmarks were automatically detected in all images, the images were aligned to the mean CelebA [Liu et al., 2015] face and resized to 128×128. We only consider transformations from neutral to all other expressions and vice-versa.

Even though this dataset contains paired data we treat the images as unpaired. This allows us to perform qualitative comparisons on the edits produced by the different methods with respect to plausible targets. However, the images cannot be directly employed for quantitative evaluations. This is due to global misalignment issues, as shown in Fig. 4-9. Moreover, the space of edits is multi-modal, *i.e.* there are several ways to express a facial expression, and the dataset only contains a single observation for each expression per subject.

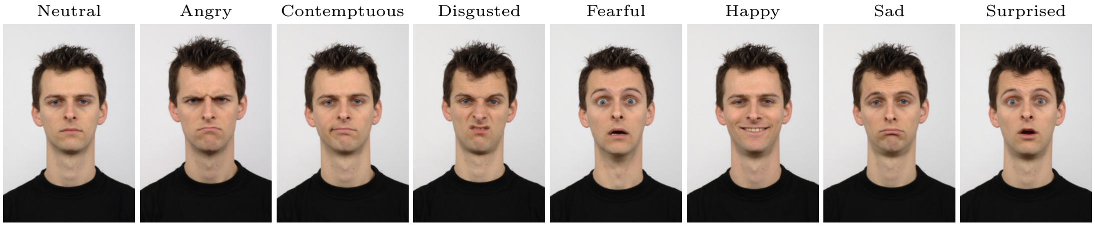


Figure 4-9: Examples of the train images found on the RafD dataset for subject 5. For each expression two additional images are recorded where the subject has different gaze directions. In addition, each instance is also captured from two additional camera angles.

4.4.2 Models

Our main baseline is StarGAN [Choi et al., 2018], a state of the art model for image-to-image translation, which is also the most similar method to our approach. We define three novel models to evaluate our contributions separately. WarpGAN is used to denote models that output a warp field. We append a "+" to the name of all models that employ signed labels. Thus, StarGAN+ is used to evaluate the effect of using signed labels, and WarpGAN+ is our final proposed model.

An obvious alternative to our model consists of fitting a dense flow field to the results generated by StarGAN. This approach is similar to the iGAN [Zhu et al., 2016] model, which was discussed in Section 4.2. We tested it on CelebA using the dense optical flow matching technique described in [Zach et al., 2007] and [Sánchez Pérez et al., 2013][3], and we denote this method by SGFlow. An example of SGFlow is shown in Fig. 4-10, where results for two values of the data term, λ, of the optical flow method [Zach et al., 2007, Sánchez Pérez et al., 2013] are shown. Warping based on optical flow can lead to artefacts when the method fails to find good correspondences. Constraining a StarGAN model to generate images that are amenable to optical flow estimation is not trivial. Thus, we do not consider this model in the remaining experiments.

We also experimented with the GANimation [Pumarola et al., 2018] approach using the code provided by the authors. However, we were unable to generate meaningful results when training the method with binary attributes. In the original work the method was trained with the EmotioNet dataset [Benitez-Quiroz et al., 2016], which is semi-automatically annotated with soft action unit attributes [Ekman and Friesen, 1976]. We therefore suspect that the method is heavily dependent on that type of data, which specify facial expressions in great detail.

[3] An OpenCV implementation of the method is available as `cv2.DualTVL1OpticalFlow_create()`

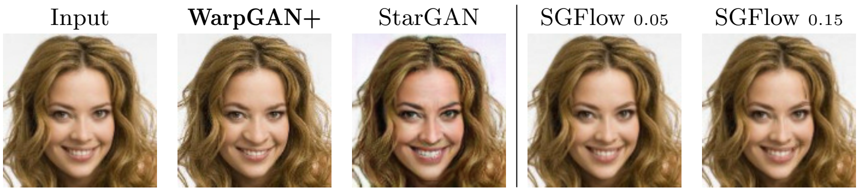


Figure 4-10: Employing a dense flow method [Zach et al., 2007, Sánchez Pérez et al., 2013] to transfer a "big nose" edit from StarGAN [Choi et al., 2018]. Results for the flow method with $\lambda = 0.05$ and $\lambda = 0.15$ are shown. StarGAN has edited the input to such lengths that good correspondences between the input and output cannot be found by the flow method. A value of $\lambda = 0.05$ does not transfer enough content, while $\lambda = 0.15$ produces visible artefacts, for example on the forehead.

4.4.3 Implementation details

All models were trained on a single Titan X GPU using the TensorFlow [Abadi et al., 2015] framework. The Adam optimizer [Kingma and Ba, 2015] is used with a learning rate of 0.0001, with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The model hyper-parameters are shared for all datasets: $\lambda_{\text{cls}} = 2$, $\lambda_{\text{gp}} = 10$, $\lambda_{\text{cycle}} = 10$ and $\lambda_{\text{s}} = 125$.

The network architectures used are shown in the appendix in Tables B.1 and B.2. During training, function T warps the input according to the generated displacement field, $\mathbf{w}$, using bilinear interpolation[4]. To improve image quality at inference time we use a bicubic interpolant[5], as an increase in quality in the upsampled warp field leads to superior edits.

The networks were trained on CelebA for 20 epochs, and on RafD for 200 epochs (due to the reduced size of this dataset). A batch size of 16 was used for all the experiments. For data augmentation, the images are left-right flipped randomly.

For the StarGAN baseline we employ the implementation provided by the authors, where we keep all their recommended hyper-parameters except for $\lambda_{\text{cls}} = 0.25$. The choice of λ_{cls}, for StarGAN and our models, is informed by the results shown in Fig. 4-14.

In StarGAN the discriminator weights are updated 5 times for each generator update, as Wasserstein GAN [Arjovsky et al., 2017] methods are advised to train the discriminator up to convergence. However, we found it sufficient for the WarpGAN models to train the discriminator for a single step, for each generator step.

[4] T is implemented with TensorFlow as $T(\mathbf{x}, \mathbf{w})$ = `tf.contrib.image.dense_image_warp(x, w)`.

[5] T is implemented with OpenCV as $T(\mathbf{x}, \mathbf{w})$ = `cv2.remap(x, w, interpolation=cv2.INTER_CUBIC)`.

4.4.4 Quantitative metrics

Quantitative evaluation is challenging for our setting. The use of unpaired data means that we do not have target images to compare with the results of the model. For datasets that have paired data, such as the RafD dataset, in principle, the target images could be used to evaluate the method. However, as the model is free to learn alternative plausible transformations, we could be penalising the model for generating plausible yet different edits. Instead, we provide a methodology based on separately trained models.

Attribute classifier network We train a classifier, $\hat{C}(\cdot)$, for the image attributes using the training data. The network has the same architecture as the classifier used for StarGAN, and is trained with the cross entropy loss of equation 4.3. This classifier is used to estimate quantitatively if the edited images have the requested attributes. It achieves an average accuracy of 82.46% on the real test data; we note that the classifier is indicative but should not be considered ground truth.

Face re-identification network We also use a pretrained face re-identification model, Facenet [Schroff et al., 2015], to evaluate whether the edits preserve the subject's identity. In more detail, the Facenet model trained on the MS-Celeb-1M dataset [Guo et al., 2016] is used. This dataset consists of 10 million images and 100k unique identities. As both CelebA and MS-Celeb-1M were collected from publicly available Internet images, we expect some overlap between both datasets. This pretrained model is provided by the authors and is publicly available[6].

Attribute accuracy The first metric that we use is the rate of images classified as having the target attribute, which for methods employing binary labels is defined as

$$L_{\text{acu}} = \frac{1}{k}\sum_{\mathbf{x},\bar{\mathbf{c}}}[\hat{C}(G(\mathbf{x},\bar{\mathbf{c}}),\bar{\mathbf{c}}) \geq 0.5], \tag{4.14}$$

where k is the number of elements in the test set. For the methods using signed labels, the metric is similarly defined as

$$L_{\text{acu}} = \frac{1}{k}\sum_{\mathbf{x},\hat{\mathbf{c}}}[C(G(\mathbf{x},\hat{\mathbf{c}}),B(\hat{\mathbf{c}})) \geq 0.5]. \tag{4.15}$$

[6] https://github.com/davidsandberg/facenet

For each image, $\mathbf{x}$, in the test set a random target attribute vector, $\bar{\mathbf{c}}$ or $\hat{\mathbf{c}}$, is drawn. The target attribute is constrained to edit only a single attribute in each image. Hence, the classifier, $\hat{C}(\cdot)$, is evaluated exclusively for that particular attribute.

Identity score The second metric is an identity preservation score, which for methods employing binary labels is defined as

$$L_{\text{id}} = \frac{1}{k} \sum_{\mathbf{x},\bar{\mathbf{c}}} \left[1 - \|\boldsymbol{\psi}(\mathbf{x}) - \boldsymbol{\psi}(G(\mathbf{x}, \bar{\mathbf{c}}))\|_2^2\right], \tag{4.16}$$

and for the methods using signed labels is similarly defined as

$$L_{\text{id}} = \frac{1}{k} \sum_{\mathbf{x},\hat{\mathbf{c}}} \left[1 - \|\boldsymbol{\psi}(\mathbf{x}) - \boldsymbol{\psi}(G(\mathbf{x}, \hat{\mathbf{c}}))\|_2^2\right], \tag{4.17}$$

where $\boldsymbol{\psi}(\cdot)$ are the features in the last layer of the face re-identification network. A distance larger than 1.2 (score lower than -0.2) has been used to indicate that two faces belong to different people [Schroff et al., 2015].

There is a trade-off between attribute accuracy and identity score. On one extreme, a new face that has the target attribute and does not match the original face would achieve maximal attribute accuracy with negative identity score. On the other, not modifying the input image has maximal identity score, yet it does not achieve the target edit. A perfect model would modify the image just enough to contain the target attribute, thus, maintaining as much content as possible from the input into the edited image.

4.4.5 Ablation study

An ablation study was performed where different versions of the model were trained by removing one of the losses. Qualitative results are shown in Fig. 4-11, where a common image is edited by each of the models. Without the cycle loss, L_{cycle}, the model is not encouraged to generate invertible transformations, which leads to artefacts in the images. Removing the smoothness loss, L_{s}, seems to have little effect, yet, this is needed for applying the warps at higher resolutions. Without the adversarial loss, L_{adv}, the model produces images that are less realistic. We qualitatively observed this in our experiments, and in this example, it is particularly noticeable for the "smile" attribute. Removing the classification loss, L_{cls}, leaves the generator with a trivial solution, *i.e.*

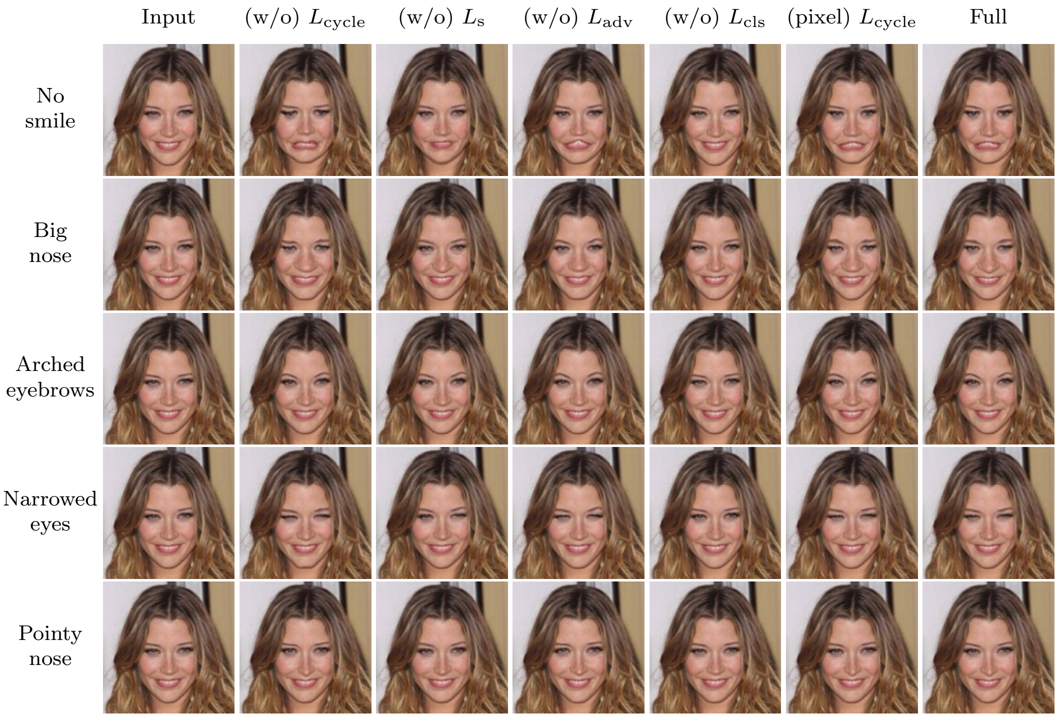


Figure 4-11: Ablation study, where we remove different losses in our model. It can be seen that all the losses play a part in generating realistic edits.

the network predicts an identity transformation. Additionally, we also experimented with substituting our warp-based cycle loss (equation 4.7) with the pixel-based cycle loss (equation 4.2) of StarGAN, and denote this experiment as (pixel) L_{cycle}.

In order to quantitatively evaluate the effect of each loss in the model, the attribute accuracy and the identity score are reported in Fig. 4-12, for the different ablated models. It can be seen how removing each loss has a negative effect either on the attribute or the identity score with respect to the full model. There is an exception when removing the adversarial loss, L_{adv}. However, the identity score does not model realism directly. This can be seen qualitatively in Fig. 4-11, where the edited image for the “smile” attribute appear less realistic without this loss.

Our warp-based cycle loss seems to have little to no effect under the attribute accuracy and identity score metrics. However, there are several advantages of using this loss instead of the pixel-based alternative. On a theoretical level it is more coherent to have a loss on the parameters that are estimated by the model, rather than on a surrogate task. We also expect invertible geometric transformations to be better behaved during partial edits. Finally, the loss is more informative for areas of similar colours, *i.e.* under

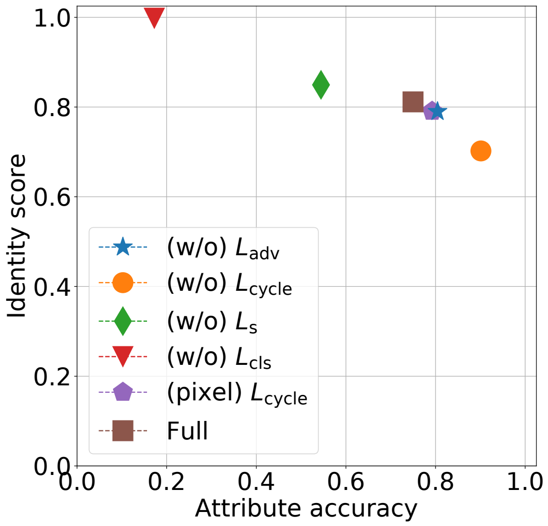
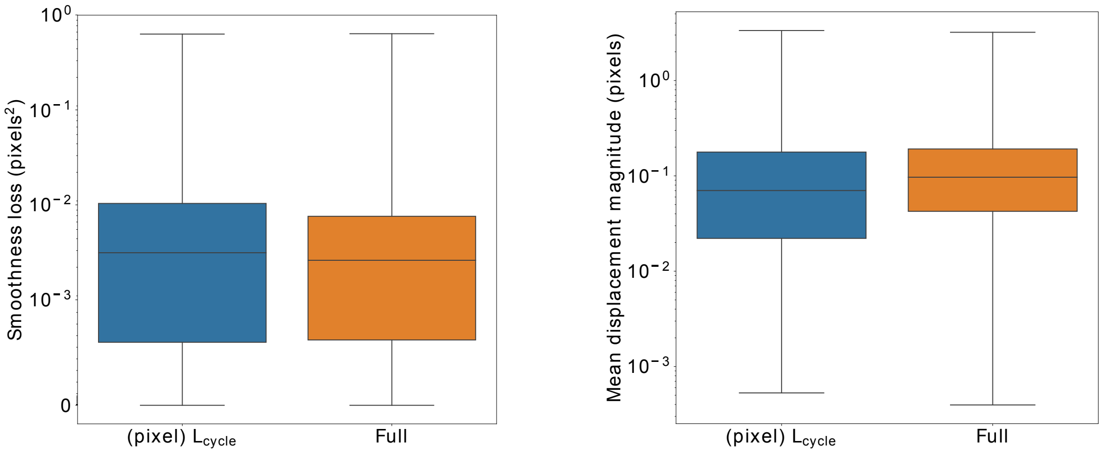


Figure 4-12: Presence of the edited attribute (x-axis) vs face re-identification score (y-axis), higher is better for both axes. Removing each loss in our model has a detrimental effect in either accuracy or identity preservation. Based on these metrics, the adversarial loss and our warp-based cycle loss seem to have little effect. However, we qualitatively observed that without the adversarial loss, the edited images were less realistic. The effects of the warp-cycle loss will be further analysed in Fig. 4-13a.

a pixel-based cycle loss, the model is free to permute the pixels without incurring a cost under this loss. In order to quantitatively analyse the last effect, we measure the smoothness on the displacement vectors in similar colour regions, as shown in Fig. 4-13a. The input images are segmented into super-pixels using the approach of [Achanta et al., 2012], and a super-pixel is considered to contain similar colour pixels if the standard deviation averaged across channels is smaller than 0.1 for pixel values in the range $[-1, 1]$. Additionally, a skin segmentation network is employed to only select the pixels in the image that belong to the face, as our model does not in general warp regions of the image outside of the face area. Hence, the smoothness loss of equation 4.8 is evaluated only for displacements of similar coloured regions inside the face. A mild increase in smoothness of the predicted warps can be observed in the model that employs our warp-based cycle loss, thus empirically demonstrating our hypothesis. Note that both models are also trained with the smoothness loss in equation 4.8, which might explain why there is not a larger gap between both models. The mean displacement magnitude in those regions is also reported in Fig. 4-13b. This shows how the model using the pixel-based cycle loss is estimating very small warps in those areas, which are probably inconsequential for the editing operation.



(a) Smoothness of the displacement vectors, where the unit for the y-axis is pixels^2, lower is smoother.

(b) Mean displacement magnitude of the displacement vectors, where the unit for the y-axis is pixels.

Figure 4-13: Quantitative comparison of a WarpGAN+ model trained with the pixel-based cycle loss (equation 4.2), and with the warp-based cycle loss (equation 4.7). The smoothness and average magnitude of the displacement vectors in the warp field for similar coloured regions in the face are shown. Note that the axis is on a logarithmic scale. Employing our proposed warp-based cycle loss leads to smoother warps in those regions and that contain fewer very small (and presumably pointless) displacements.

4.4.6 Quantitative results

In this section we evaluate our model with respect to previous work. This is done in terms of the attribute accuracy and identity score metrics that were defined in the previous section, and with a user study to estimate perceptual results. For this set of experiments the same attribute per image is edited for all models.

Accuracy vs identity preservation The attribute accuracy and the identity preservation score are plotted for StarGAN, StarGAN+ and WarpGAN+ for a range of λ_{cls} values, as shown in Fig. 4-14. An ideal editing model would be located on the top-right. Changing StarGAN to use signed labels (StarGAN+) moves the curve towards higher attribute accuracy with comparable identity score. Our warping approach (WarpGAN+) allows for stronger identity preservation than StarGAN+. Overall, our approach better preserves identity than StarGAN, for similar levels of attribute accuracy. For the rest of the experiments, we picked λ_{cls} based on these results: choosing the value that leads to both higher attribute accuracy and identity score.

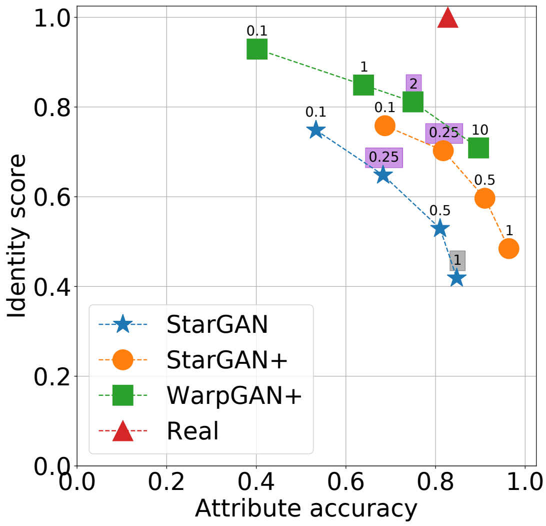


Figure 4-14: Presence of the edited attribute (x-axis) vs face re-identification score (y-axis), higher is better for both axes. The classification loss weight, λ_{cls}, is shown on top of each marker. Highlighted in grey is the value used by the StarGAN authors for this dataset, and in purple the ones used in this chapter. Compared to previous work, our model produces edits that better preserve identity.

Accuracy vs realism The attribute accuracy does not provide a ground truth answer to whether the images contain the target attribute. Moreover, the identity score measures how close two images are, yet, it does not model the perceptual realism of the edited image. A useful editing model has a high-level of realism and can produce the target edit. In order to quantitatively evaluate both aspects, we perform a user study on Amazon Mechanical Turk (MTurk) to evaluate the quality of the generated images, for StarGAN, StarGAN+ and our model. For each method, we use the same 250 test images from CelebA and edit the same attribute per image.

We conducted two experiments, one to evaluate the realism of the images, where the workers had to answer whether the image presented was real or fake, and another to evaluate attribute editing, where we asked the workers whether the image presents the target attribute. In both experiments, the workers were randomly shown a single image at a time: either an edited image or an unaltered original image. To evaluate the reliability of the workers, a number of easy to classify images were mixed with the data, and used as a control. Workers needed to give the right answer to at least 90% of the control images for their data to be considered reliable. Images with fewer than 3 annotations are discarded, as they are considered unreliable data. Finally, a simple majority voting scheme was used to determine the classification of each image.

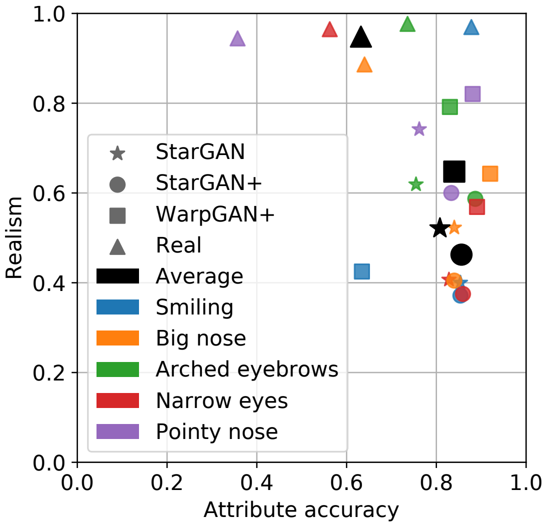


Figure 4-15: Human perception of the presence of the desired attribute (attribute accuracy) vs realism of the image, as indicated by the user study, higher is better for both axes. Images generated by our model are more realistic than those generated by previous work.

For both experiments, example images were shown to the workers for guidance before commencing the task. For the experiment evaluating realism, typical failure cases for all models were shown as examples of fake images. For the evaluation of the presence of the target attribute, curated examples from training data edited with our model were shown. This allows showing two images per attribute for guidance, where the only difference is the presence or lack of the attribute, as both depict the same subject.

Some images in the CelebA dataset contain border artifacts due to the alignment process that the authors used for the aligned version of the dataset. An example of such artifacts can be seen in the first image in Fig. 4-8. In order to get more reliable results, none of these images were included in the pool of 250 images used for the experiments.

Results of this user study are shown Fig. 4-15. For the real data, the workers were able to evaluate reliably the image realism, however they were often inconsistent with the original attribute labels. Nonetheless, the workers' performance on real data should not be taken as an upper bound for two reasons. First, the images in the dataset were annotated by a professional annotation company, while we employed untrained MTurk workers in our experiments. Second, as all methods tend to generate exaggerated edits to ensure that the images are classified correctly, the edited images are easier to classify than the original images. For the editing models, the attribute accuracy is consistent

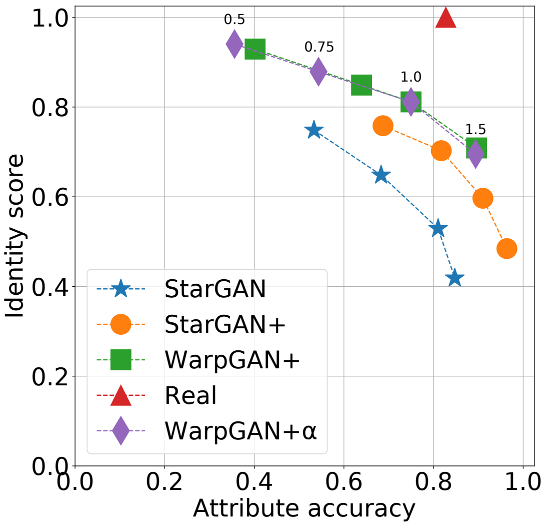


Figure 4-16: Presence of the edited attribute (x-axis) vs face re-identification score (y-axis), higher is better for both axes. For all models except WarpGAN+α, this figure is identical to Fig. 4-14. For WarpGAN+α the value of α is shown on top of each marker. Modifying the α value at test time in our model has a similar effect as training the model with different λ_{cls} values.

to that reported by the classifier network in Fig. 4-14. However, identity score and realism do not align, as they measure different notions. An image might contain only small edits, which the identity network is invariant to, yet those edits could include unrealistic artefacts that can be easily detected by humans. All editing models achieve good attribute accuracy, where the room for improvement is mostly on the realism axis. Our model (WarpGAN+) produces realistic results for most attributes, and it is able to generate images that are more realistic than StarGAN.

Effect of α We quantitatively evaluate the effect of scaling the displacement fields by a scalar α. For this experiment, we take WarpGAN+ trained with $\lambda_{\text{cls}} = 0.25$ and we evaluate the identity score and the attribute accuracy on the test set for different values of α. Results are shown in Fig. 4-16 for this model, which is denoted as WarpGAN+α. The curve produced by employing different values of α is very similar to the curve in Fig. 4-14, which was produced by modifying λ_{cls}. Hence, for $\lambda_{\text{cls}} = 2.0$, a similar effect to changing the value of λ_{cls} used during training can be achieved by choosing an alternative value of α at test time. This is in contrast to previous work [Zhu et al., 2017, Choi et al., 2018], where modifying the strength of the effects requires training a model with the new parameters.

Model	128×128	256×256	512×512	1024×1024	2048×2048
StarGAN	**19.9**	79.7	318.9	1275.4	5101.8
WarpGAN	26.3	**26.3**	**26.3**	**26.3**	**26.5**

Table 4.2: Comparison of model efficiency for StarGAN and WarpGAN for different image sizes. The number of floating point operations (GFLOPs) required to edit an image is shown, lower is better. In each column the first row indicates the input image resolution. The warping step in our method introduces additional operations when operating at the resolution of the training images, 128×128. However, the cost remains mostly constant in our method as the resolution increases, as only the warping operation is performed at the resolution of the input image.

Model complexity and efficiency Model complexity is measured by the number of learnable parameters in the neural networks. In StarGAN and WarpGAN this is almost identical, as the only change in the network architecture appears on the last layer, which in StarGAN in an RGB image with three channels, and in WarpGAN is a dense flow field of two channels. In more detail, there are 8,434,243 trainable parameters in the generator network in StarGAN, and 8,431,106 in the generator of WarpGAN.

Model efficiency is measured by the number of floating point operations (GFLOPs) required to process an input image, and it is shown in Table 4.2. As the neural network architectures are fully convolutional, this is the cost of running the network for an input image of the given resolution. In WarpGAN, there is a moderate increase in GFLOPs when operating at the lowest resolution, due to the the warping operation. Contrary to StarGAN, in our method the cost of editing is mostly independent of the image resolution, as estimating the flow fields at the training data resolution has a constant complexity. Note that this is a generous comparison for StarGAN, as it is unlikely that a network with the same number of learnable parameters would be able to learn the transformations at resolutions which differ significantly from 128×128. Therefore, the larger network that would be required, would further increase the computational cost in StarGAN.

4.4.7 Qualitative results

We show qualitative results on the CelebA dataset in Fig. 4-17. For each input image, we show the edited images corresponding to changing a single attribute. In order to give an intuition of the quantitative metrics used in the previous section, the attribute

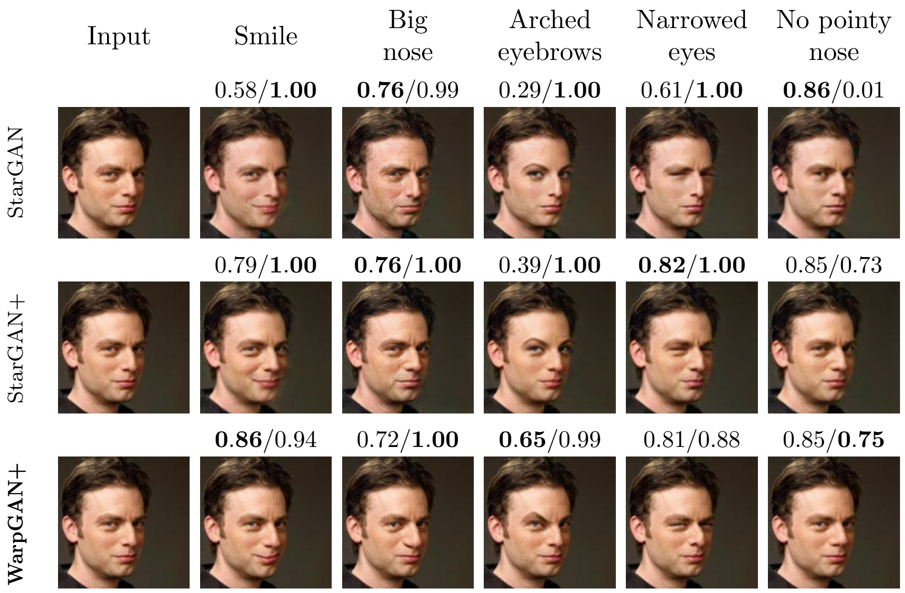


Figure 4-17: Comparison to previous work on the CelebA dataset. For a given input image, first column, each method attempts to transfer the semantic attribute in its corresponding column. The identity score and attribute probability are shown as (id/cls) on top of each image (higher is better for both). These scores were defined in section 4.4.6. Our approach edits the attributes of the input images while better preserving the identity of the subject. Please see the supplemental material for videos of these edits[7].

accuracy and the identity score are shown on top of each image. StarGAN [Choi et al., 2018] often changes characteristics that are not related to the edited attributes, such as the skin tone or the background colour. StarGAN+ produces more localised edits than StarGAN. Our approach generates changes that are less exaggerated, and better preserve the identity of the subject.

Qualitative results are shown for the RafD dataset on Fig. 4-18. As this dataset contains paired data, we can compare the edited images with plausible targets. For some of the expressions, our model produces more realistic edits than previous work. In addition, we can apply our results seamlessly at the original image resolution of 580×540, denoted as WarpGAN-HR, in contrast to the 128×128 resolution of StarGAN. However, in this dataset the subjects were recorded while enacting extreme expressions. This demonstrates the limits of editing by warping, as many of these extreme expressions require edits that cannot be modelled well by geometric deformations alone. For example, previously occluded content that becomes visible after the change in expression.

[7] https://garoe.github.io/thesis/supplement/animated_edits.html

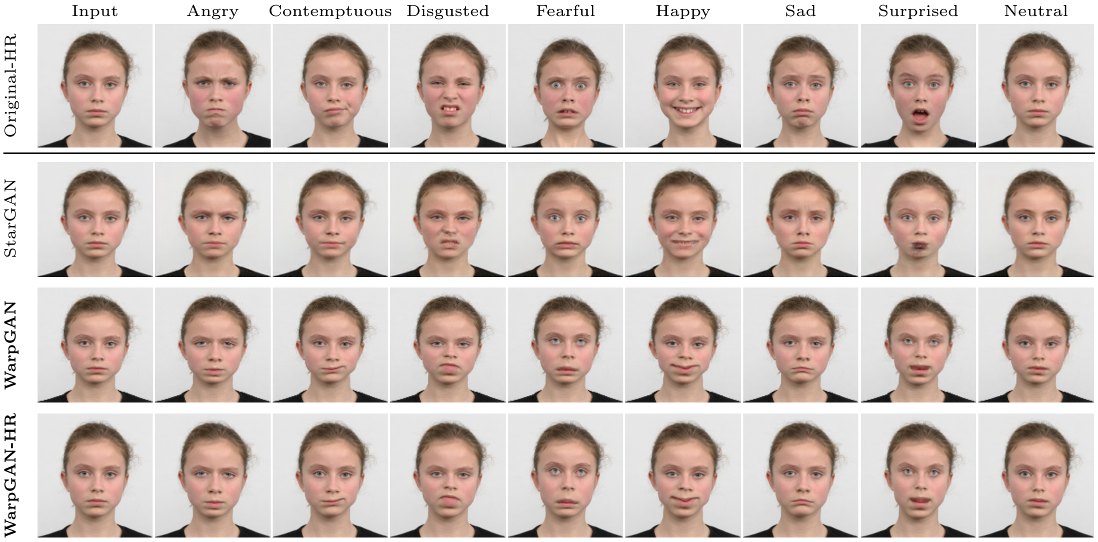


Figure 4-18: Comparison to previous work on the RafD dataset. For a given input image, first column, each method attempts to change to the expression in its corresponding column. This dataset contains paired examples, which we show in the top row. StarGAN is limited to operate at a fixed resolution (128×128), while our approach can be applied at the original resolution (580×540). Please see the supplemental material for videos of these edits[6]. (Zoom in for details)

Results for an in-the-wild image downloaded from Flickr are shown in Fig. 4-19. This demonstrates the power of the warping representation by operating at a far higher resolution (3456×5184) than can be achieved by direct methods. Please see the supplemental material for animated edits.

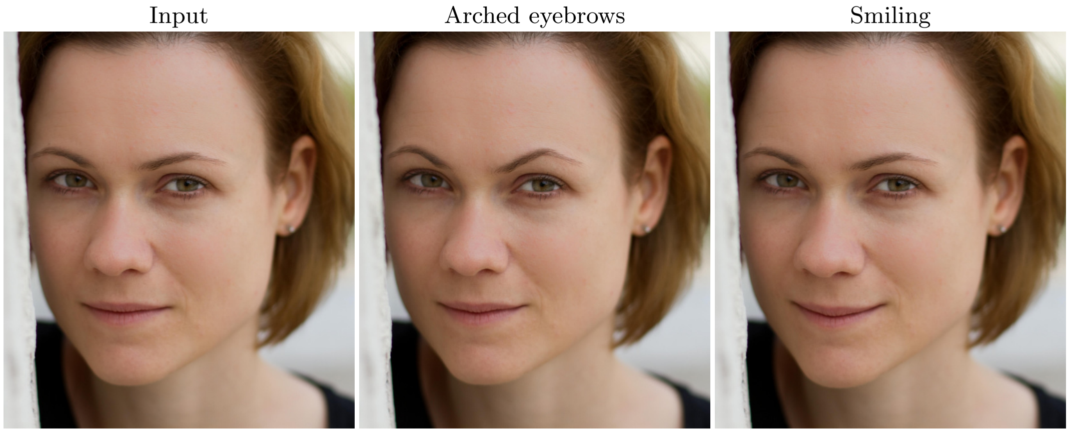


Figure 4-19: Semantic image editing at high resolution (2950×3450). The user requests a change in a semantic attribute and the input image is automatically transformed by our method into the attribute in the corresponding column. The intensity of edit is adjusted by a scalar hyper-parameter, α, which we set to 1.5 and 0.5, respectively. The identity and fine details of the original input are preserved. Please see the supplemental material for videos of these edits[6]. (Zoom in for details)[8]

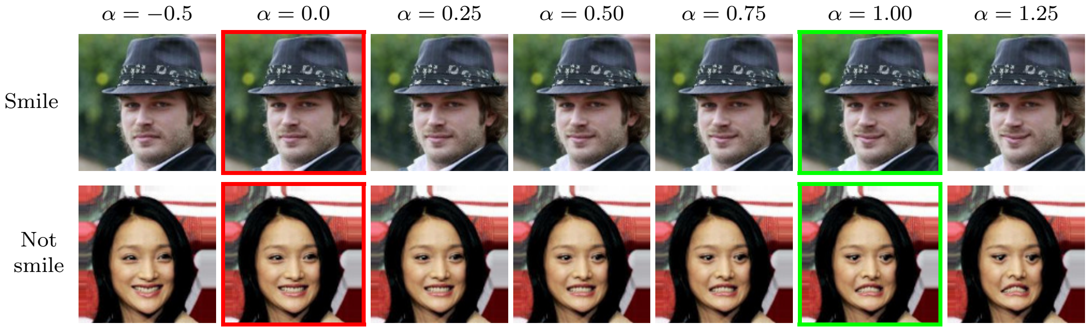


Figure 4-20: Partial editing with our model, for the "smile" attribute. A single warp is generated by our model, which is interpolated and extrapolated by scaling the magnitude of the displacement vectors by α. The input image, corresponding to $\alpha = 0$, is progressively edited in both directions. Please see the supplemental material for videos of these edits[6].

Partial edits Results of interpolation and extrapolation of warp fields generated by our model are shown in Fig. 4-20. In order to obtain these results, a single warp field per input image is generated by the network. The displacement vectors in the warp field are scaled by α as described in equation 4.13. The intermediate images vary smoothly in a realistic fashion, while preserving the identity of the subject. Moreover, extrapolation can be achieved by employing an α value outside of the $[0, 1]$ range.

[8] Input image courtesy of Flickr user Dawolf.

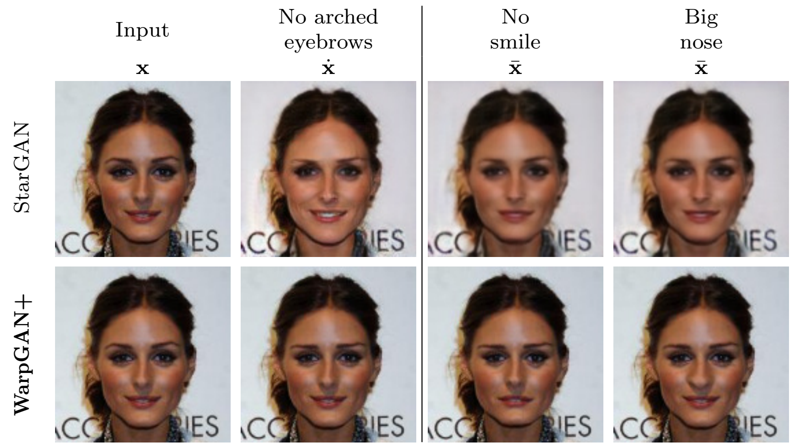


Figure 4-21: Composing edits, where the input image, $\mathbf{x}$, is first edited to "no arched eyebrows", $\dot{\mathbf{x}} = G(\mathbf{x}, \dot{\mathbf{c}})$. The "no arched eyebrows" image, $\dot{\mathbf{x}}$, is edited a second time, $\bar{\mathbf{c}} = G(\dot{\mathbf{x}}, \bar{\mathbf{c}})$, to the corresponding semantic attribute. As it has been previously demonstrated [Sanchez and Valstar, 2018], StarGAN [Choi et al., 2018] always tries to reconstruct the input, ignoring the labels in the second image transfer operation. In contrast, our method, due to its regularization via warping, is able to transfer the attributes in all cases.

Edit composition Composition of edits consists of editing an image repeatedly. For example, first changing the shape of the nose, and then change the shape of the eyebrows using the previously edited image as input, where the expected result is an image that contains both edits. Using the model in this way is demonstrated in Fig. 4-21. Formally this is defined as

$$\bar{\mathbf{x}} = G(G(\mathbf{x}, \dot{\mathbf{c}}), \bar{\mathbf{c}}), \tag{4.18}$$

where $\mathbf{x}$ is the input image, $\dot{\mathbf{c}}$ is an intermediate attribute vector, $\hat{\mathbf{c}}$ is the final attribute vector, and $\bar{\mathbf{x}}$ is the final edited image. For simplicity of notation, this is defined with respect to traditional binary labels, and it is obvious how to state it using signed labels. It would be expected that the model would edit the image consistently, regardless of any intermediate editing steps. Formally, this can be expressed as

$$G(G(\mathbf{x}, \dot{\mathbf{c}}), \bar{\mathbf{c}}) \approx G(\mathbf{x}, \bar{\mathbf{c}}). \tag{4.19}$$

However, as demonstrated in Sanchez and Valstar [Sanchez and Valstar, 2018], StarGAN ignores the final target attributes, $\bar{\mathbf{c}}$, and reconstructs the original image in all cases, as shown in Fig. 4-21. On the other hand, our model is unable to "hide" information on the edited images, as the edit operations are limited to warping. Thus, our method transfers the desired attributes in all the images.

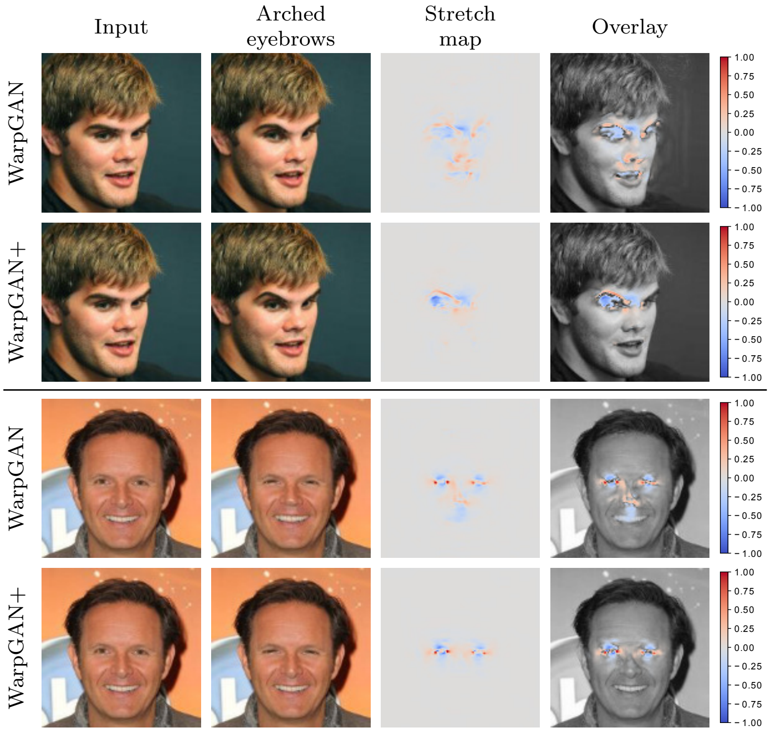


Figure 4-22: Stretch maps computed from the warp fields, for both WarpGAN and WarpGAN+. The log determinant of the Jacobian of the warp is shown, where blue indicates stretching and red corresponds to squashing. The signed labels used by WarpGAN+, leads to correctly localized edits, as opposed to the binary labels used by WarpGAN.

Visualising warp fields A further advantage of our model is the interpretability of its edits. This is demonstrated in Fig. 4-22, where we show the log determinant of the Jacobian of the warp field, which shows image squashing and stretching. It can be seen how employing signed labels leads to more localised edits. Moreover, the values from the stretch maps can potentially be used to automatically determine which areas have been stretched or compressed excessively by the network.

Please see appendix B.3 for additional qualitative results.

4.5 Discussion

In this chapter a novel way to describe semantic image edits from unpaired data using warp fields was introduced. We have demonstrated that, despite limitations on the set of edits that can be described using warping alone, there are several clear advantages

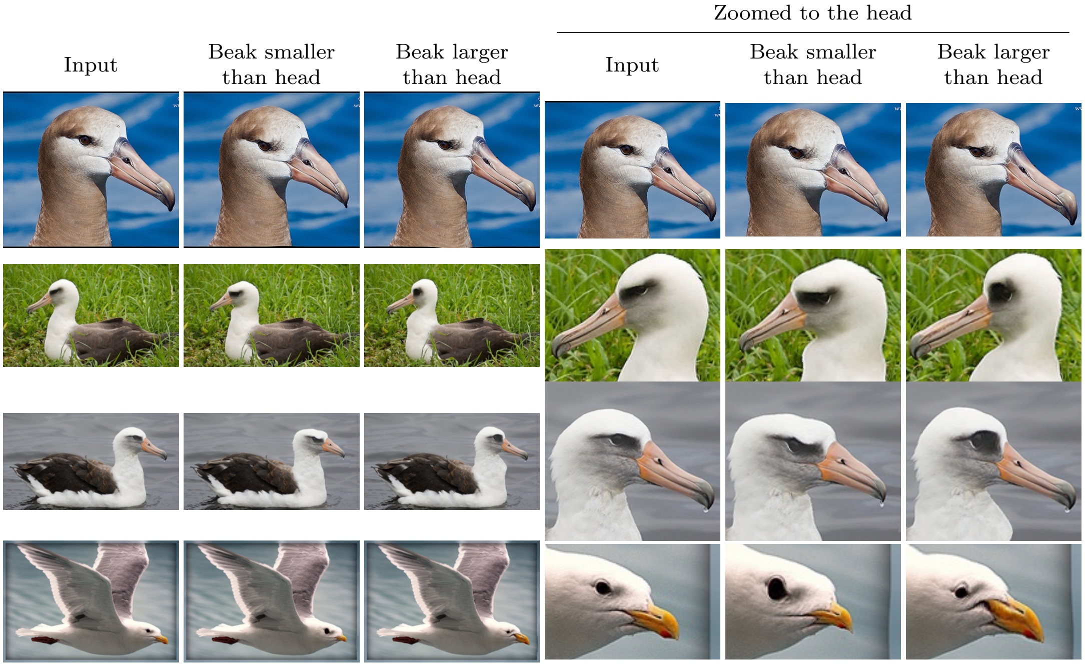


Figure 4-23: Preliminary results from our model on test images from the Cub200 dataset [Wah et al., 2011]. The model attempts to transfer the attribute (relative beak size) in each column to the input image. For easiness of comparison, a crop of the head area is shown in the last three columns.

to modelling edits in this way: they better preserve the identity of the subject, they allow for partial edits, and they are applicable at arbitrary resolutions.

In this chapter the model was only evaluated on aligned face data. An important question is whether the model is general enough to be successfully applied to other types of images. In order to test this, we performed some preliminary experiments on the Cub200 dataset [Wah et al., 2011]. This dataset contains 11,788 images of 200 bird species taken in the wild. The images are annotated with 15 body part locations, 312 binary attributes, and semantic masks of the bird body. The train/test split recommended by the authors uses 5,994 images for training, and the remaining 5,794 for testing. In a similar fashion as the datasets used in this chapter, the bird faces are aligned with an affine transform with common scaling and cropped to 128 × 128 using four landmark locations: the beak, the crown, the forehead and the right eye. If the right eye is not visible, the image is left-right flipped. For all images, the background is removed using the semantic masks.

Results of the model on this dataset are shown in Fig. 4-23, for the attribute "beak size relative to head". As the size is relative between two parts, it can be observed how the model learns to edit both the head and the beak in order transfer the requested attribute. Interestingly, in the fourth row, the model learns that larger eyes generally correspond to bigger heads, and edits the image accordingly. Still, this dataset is more complex and contains significantly fewer images than the CelebA dataset, which leads to deformations that are not always realistic, such as the neck areas in the second and third rows.

There are several avenues for future work, including different parametrisations for the warps, *e.g.* in the form of velocity fields [Ceritoglu et al., 2013], which ensures that the deformations are diffeomorphic. It might be the case that for data outside of the domain of faces, additional regularisation in this form might be needed. It is also possible that it would remove the need for the smoothness regularisation loss, as non-smooth warps are not invertible.

Additional intermediate representations that upsample well could be added to increase the model flexibility, such as local colour transformations [Chen et al., 2016]. This would allow the model to perform a wider range of edits, without sacrificing the ability to operate at arbitrary resolutions. Moreover, the viability of the approach has already been demonstrated [Gharbi et al., 2017], albeit with the constrains of assuming access to paired data, and that the edits on each image patch are independent from other patches.

An alternative avenue of future work could look at employing image inpainting methods [Pathak et al., 2016, Yang et al., 2017, Gilbert et al., 2018, Liu et al., 2018] in conjunction with our model. These methods could be applied only on small areas were the warp field has stretched or squashed the image excessively. These regions could be automatically detected with the log determinant of the Jacobian maps shown in the previous section. Moreover, by applying the method only on a small region, this approach would potentially alleviate their problems to scale to high-resolution images.

Chapter 5

Conclusions

5.1 Summary of contributions

In this thesis we have investigated two models that can learn semantic image transformations from data: a fully unsupervised generative model that included structured uncertainty prediction on its outputs, and a discriminative model that learns warp fields from unpaired data. We have discussed how these models can be used as the basis for an image editing application. In this chapter we summarise the contributions introduced in this thesis, and examine possible avenues of future work.

5.1.1 Structured uncertainty prediction

A model to learn structured uncertainty from reconstruction residuals was presented in Chapter 3. This is an unsupervised deep generative model; thus it is able to learn a latent space from which new images may be sampled. The learned latent space may also be used for interpolations, where face frontalisation was illustrated by a latent space interpolation between an image and its x-flipped mirror image.

We demonstrated that a dense covariance matrix, which has a spatially localised sparse inverse, can be tractably learned from a single residual image in deep generative models with a multivariate Gaussian likelihood. Samples from the pixel uncertainty distribution were shown to generate more plausible residual images, and the model was able to generalise these predictions to test data.

The learning task posed a severely unconstrained problem, and two approaches were proposed in order to regularise the problem. The first approach consisted of training two separate models, where one model learns how to estimate the mean of the Gaussian distributions, assuming a factorised Gaussian model (as is standard for VAEs). Subsequently, a separate model learns the structured distribution of the residual images, which involves estimating a sparse inverse matrix. The model for the mean of the Gaussian distributions was trained and fixed. Then, the structured uncertainty model is trained separately. This proved to be a reliable way to fit these models to image data. The second approach allowed learning both the mean and inverse covariance of the multivariate Gaussian distributions in a joint fashion, with the help of two regularisation functions. This reduced the overall number of parameters needed for the model, and in some cases, it appeared to improve quality of the learned latent space.

The model was quantitatively evaluated by reporting the likelihood on a test set, which is a typical metric in fully unsupervised generative models, as discussed in Section 1.3. The likelihood, evaluated with the covariance matrices estimated by the model, was similar to the one evaluated with ground truth covariance matrices on synthetic data. Moreover, when evaluated on real images, both of our models achieved better likelihoods than previous work.

5.1.2 WarpGAN

A model to learn warp fields for image editing was discussed in Chapter 4. By employing recent advances in adversarial networks [Goodfellow et al., 2014, Zhu et al., 2017], it does not require ground truth target images indicating how the edited images should look like.

We demonstrated that warp fields can be tractably learned from unpaired data and applied at arbitrary resolutions. Editing in this way provides additional benefits typically associated with warp fields. Namely, it results in a model that is more interpretable and easier to regularise than pixel-based methods. Additionally, efficient training of the model with low resolution images, 128 × 128, was demonstrated, which can subsequently be applied for inference with arbitrary resolution images.

As this model was designed specifically for image editing, it was evaluated with the metrics introduced in Section 1.3. Namely, by means of user studies, and with automated methods for detecting the presence of the edit, as well as identity scores. It was demonstrated under these metrics that the method is able to generate edits that are more realistic than previous work, and that are better at preserving the identity of the subject in the image.

5.2 Limitations and future work

Despite significant progress in the area of learning semantic image transformations from data, there are several limitations in current methods. In Chapters 1 and 2, we discussed how deep generative models are of interest for this task. In particular, Variational Autoencoders (VAE) [Rezende et al., 2014, Kingma and Welling, 2014] and Generative Adversarial Networks (GAN) [Goodfellow et al., 2014]. Moreover, regression approaches using GANs are also popular [Isola et al., 2017], and within those, unpaired image-to-image translation methods [Zhu et al., 2017] are particularly relevant. In this section, we discuss limitations and future work for these models, with a focus on the solutions presented in this thesis.

5.2.1 Variational Autoencoders (VAE)

Variational Autoencoders [Rezende et al., 2014, Kingma and Welling, 2014] were shown to be a powerful model to learn latent representations from data. However, one of the main limitations in the VAE-based models is that they are known to generate blurry outputs. Previous work [Larsen et al., 2016, Huang et al., 2018] attempted to address this issue by combining the model with GANs. However, this brought a number of limitations in GANs into these hybrid models. Namely, difficulty in training and mode collapse [Goodfellow, 2016].

In Chapter 3 a method to alleviate the issue of blurry outputs was presented, which was based on learning a structured residual distribution. However, training a VAE in this way proved to be a difficult task, and the approach did not perform well with complex data. A limitation of the structured uncertainty model lies in the linear assumption on the dependencies between the pixels in the residual. Therefore, an interesting avenue of future work would be to extend the model such that the data dependencies are nonlinear. An additional drawback lies in how the quality of the images generated by the model depended heavily on the value of the hyper-parameters used. Another line of work could explore learning these parameters from data in a fully automated fashion. Moreover, the noise due to limitations in the model, and the intrinsic noise in the data were entangled in the residual. Learning to disentangle these sources of reconstruction uncertainty could be interesting, as usually we would want to add the noise due to model limitations to the generated image, while the intrinsic noise is not of interest.

Samples from a VAE model are drawn using ancestral sampling. First, generating a latent space vector by sampling from the prior distribution, and then projecting the latent space vector to image space using the decoder, and a sample from the likelihood distribution in image space is drawn to produce the output image. A limitation in these models is that it is assumed that, after training, the latent space will closely follow the prior distribution. Empirically, this does not occur [Xu et al., 2019], and it usually leads to poor quality samples, as shown in Chapter 3. From an editing perspective this might cause a loss of quality in the edited images, as image manipulation is performed by interpolating in the latent space.

Recently, several methods that attempt to address this issue have been proposed [Tomczak and Welling, 2018, Xu et al., 2019, Klushyn et al., 2019]. These approaches follow a Bayesian route, where hierarchical priors are used, *i.e.* the parameters of the prior distribution are learned, and hyper-prior distributions are employed on those parameters. The latent space learned by these models is more complex and tends to better reflect the structure of the data. However, the training procedure becomes more brittle, and frequently involves tuning additional model parameters.

The model in Chapter 3 helped to some degree with the original issue, as a better data fit means that the model can devote more capacity to learn encodings in the latent space that better follow the prior. Still, measuring the quality of the latent space is not a well defined problem [Higgins et al., 2017].

5.2.2 Unpaired image-to-image translation models

Unpaired image-to-image translation models [Zhu et al., 2017] use generative adversarial networks (GAN) [Goodfellow et al., 2014]. GANs are capable of generating realistic images at large resolutions [Karras et al., 2018]. Despite much effort [Arjovsky et al., 2017], these models are still difficult to train, lack an inference mechanism, and are known to suffer from mode collapse.

Unpaired image-to-image translation methods [Zhu et al., 2017] diminish some of these issues by employing the model in a more restricted regression setting. The generator now takes as input an image from an initial domain, and is tasked to transform the image, such that it resembles an image from a second domain. As an image contains much more information than the latent space vectors used in the original GAN models, the job of the generator network is simplified. The use of additional losses, such as the cycle-loss [Zhu et al., 2017], makes training the network more stable, as the networks receive more information in the gradients. Moreover, as the output must resemble the input while also being edited, mode collapse manifests as the network is applying the same edit to different images, which in most cases is not a problem.

Despite this, unpaired image-to-image translation models have several limitations. Weak labels contain much less information than pixel-wise labels, the model only knowns that the image belongs to a given domain, and it is easy for it to pick up undesired intra-domain correlations. This leads to the model producing images that maximise the chance of being classified as belonging to the target domain, by performing exaggerated edits. The model presented in Chapter 4 partially addressed this issue in the restricted case of transformations that only require geometric deformations. However, even for those cases, there is no guarantee that the network will only learn the image transformation that we are interested in.

The uni-modality of the edits is another important issue, as for most image transformation there exists a plethora of plausible ways to achieve a realistic edit that fulfils the target criteria. Some approaches have been proposed [Ghosh et al., 2018, Chen et al., 2019] to extended the model such that it can generate multiple outputs from a single image. However, these methods offer limited control over the variations and assume that there exists a fixed number of variations in all cases.

A problem found in these models, which is also shared by most deep learning methods, is the limitation to only be able to operate at the same resolution that they were trained with. The method discussed in Chapter 4 introduced a way to address this by means of upsampling geometric transformations. However, this severely limits the type of edits that the network was able to model. Additional resolution independent image transformations tools exist [Chen et al., 2016, Guenter et al., 1998], and exploring principled ways to include them under this framework is a possible avenue of future work. In this case, our model may be viewed as the building block of a more general editing system, where the edits could be divided into a warping step and a colour transformation step. Our model would perform the warping step of the edit, while one of these methods would perform the colour transformation step.

Resolution independent methods can also be interpreted as ameliorating the issues raised by the over-parametrisation of deep neural networks. It has been empirically demonstrated [LeCun et al., 1990, Hassibi and Stork, 1993, Han et al., 2015, Li et al., 2017b] that large networks can be pruned down to significantly smaller networks with minimal loss in accuracy. This indicates that there is a significant amount of redundancies in current models, which in practice translates into considerable computational inefficiencies. Yet, attempts at directly training smaller networks have proven unfruitful. Accordingly, with current methods, large networks are needed in order to achieve good performance [Frankle and Carbin, 2019]. An interesting avenue of work would be to provide means that would allow the smaller networks to be directly trained to achieve the performance of pruned networks. For example, by improving current stochastic gradient descent methods, or by adding regularisation losses.

5.3 Final conclusions

In this thesis, we have explored techniques for learning image transformations from data for applications in image editing with semantic controls. The models employed state-of-the-art deep learning methods, which allowed them to achieve good performance and generalisation to test data. In a broad way, we can question the validity of using deep learning methods. A critique of these models from the machine learning perspective is the use of point estimates for the parameters (maximum likelihood approach). Contrary to previous methods, in most settings with deep neural networks, overfitting to the training data does not imply poor performance on test data. The current interpretation is that the networks are highly non-linear interpolators of the training data (or their learned features). Hence, with more training data the generalisation improves. However, there are many regimes where large amounts of data is not available, usually, due to the costs associated with the data labelling process.

Despite their limitations, Bayesian neural networks [Neal, 2012] are a promising direction of work that addresses this issue. This methodology has the potential of providing the good performance of deep neural networks, while adding uncertainty estimates, and guarantees against overfitting. Still, Bayesian networks are significantly harder to train than standard neural networks, both in terms of stability and computational resources. Moreover, these networks usually make factorised Gaussian assumptions on the distribution of the networks weights [Wu et al., 2019]. We believe that an approach inspired by the work presented in Chapter 3 might help with some of the issues found in these networks.

A different focus of research can be placed on the human part of the editing system, as these editing models should be useful for human users. The approach of learning image transformations with semantic sliders, exemplified by PortraitPro©, is well suited for image editing on personal computers. However, nowadays it is common to take a picture with a smartphone, perform some local editing, and share the image directly from the device, without using a desktop computer. The type of interfaces appropriate for image editing on these devices is significantly different than those used for desktop computers, as there are severe limitations in terms of screen space and due to the touch-based interactions with the system.

An avenue of future work would be to explore automated high-level techniques that could be applied in this setting, where many of the considerations for the models that have been used in this thesis still apply, in particular for the method presented in Chapter 4. For example, being able to generate photo-realistic results or producing edited images at interactive speeds. In summary, learning image transformations from data is a promising field for providing user friendly systems for image editing, and we hope that the work published in this thesis will inspire further research in this area.

Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Achanta et al., 2012] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(11):2274–2282.

[Achille and Soatto, 2018] Achille, A. and Soatto, S. (2018). Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(12):2897–2905.

[Agarwala et al., 2004] Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. (2004). Interactive digital photomontage. *ACM Transactions on Graphics (TOG)*, 23(3):294–302.

[Alemi et al., 2017] Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2017). Deep variational information bottleneck. *The International Conference on Learning Representations (ICLR).*

[Arjovsky and Bottou, 2017] Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. *The International Conference on Learning Representations (ICLR).*

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *The International Conference on Machine Learning (ICML)*, volume 70, pages 214–223, Sydney. PMLR.

[Averbuch-Elor et al., 2017] Averbuch-Elor, H., Cohen-Or, D., Kopf, J., and Cohen, M. F. (2017). Bringing portraits to life. *ACM Transactions on Graphics (TOG)*, 36(6):196.

[Barron et al., 2015] Barron, J. T., Adams, A., Shih, Y., and Hernandez, C. (2015). Fast bilateral-space stereo for synthetic defocus. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Bartholomew et al., 2011] Bartholomew, D., Knott, M., and Moustaki, I. (2011). *Latent variable models and factor analysis: a unified approach; 3rd ed.* Wiley series in probability and statistics. Wiley, Chichester.

[Beier and Neely, 1992] Beier, T. and Neely, S. (1992). Feature-based image metamorphosis. *SIGGRAPH '92*, 26(2):35–42.

[Bengio et al., 2014] Bengio, Y., Laufer, E., Alain, G., and Yosinski, J. (2014). Deep generative stochastic networks trainable by backprop. In *The International Conference on Machine Learning (ICML)*, volume 32 of *Proceedings of Machine Learning Research*, pages 226–234, Bejing, China. PMLR.

[Benitez-Quiroz et al., 2016] Benitez-Quiroz, C. F., Srinivasan, R., and Martinez, A. M. (2016). EmotioNet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Bertalmio et al., 2000] Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning.* Springer.

[Blanz and Vetter, 1999] Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 187–194.

[Bookstein, 1989] Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 11(6):567–585.

[Boyadzhiev et al., 2015] Boyadzhiev, I., Bala, K., Paris, S., and Adelson, E. (2015). Band-sifting decomposition for image-based material editing. *ACM Transactions on Graphics (TOG)*, 34(5):163:1–163:16.

[Brock et al., 2019] Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *The International Conference on Learning Representations (ICLR)*.

[Brock et al., 2017] Brock, A., Lim, T., Ritchie, J., and Weston, N. (2017). Neural photo editing with introspective adversarial networks. *The International Conference on Learning Representations (ICLR)*.

[Burda et al., 2016] Burda, Y., Grosse, R., and Salakhutdinov, R. (2016). Importance weighted autoencoders. *The International Conference on Learning Representations (ICLR)*.

[Ceritoglu et al., 2013] Ceritoglu, C., Tang, X., Chow, M., Hadjiabadi, D., Shah, D., Brown, T., Burhanullah, M., Trinh, H., Hsu, J., Ament, K., Crocetti, D., Mori, S., Mostofsky, S., Yantis, S., Miller, M., and Tilak Ratnanather, J. (2013). Computational analysis of LDDMM for brain mapping. *Frontiers in Neuroscience*, (7).

[Chandra and Kokkinos, 2016] Chandra, S. and Kokkinos, I. (2016). Fast, exact and multi-scale inference for semantic image segmentation with deep Gaussian CRFs. In *The European Conference on Computer Vision (ECCV)*, pages 402–418.

[Chandra et al., 2017] Chandra, S., Usunier, N., and Kokkinos, I. (2017). Dense and low-rank Gaussian CRFs using deep embeddings. In *The IEEE International Conference on Computer Vision (ICCV)*.

[Chen et al., 2016] Chen, J., Adams, A., Wadhwa, N., and Hasinoff, S. W. (2016). Bilateral guided upsampling. *ACM Transactions on Graphics (TOG)*, 35(6):203:1–203:8.

[Chen et al., 2018a] Chen, T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. (2018a). Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2615–2625. Curran Associates, Inc.

[Chen et al., 2017] Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. (2017). Variational lossy autoencoder. *The International Conference on Learning Representations (ICLR).*

[Chen et al., 2018b] Chen, Y.-C., Lin, H., Shu, M., Li, R., Tao, X., Shen, X., Ye, Y., and Jia, J. (2018b). Facelet-bank for fast portrait manipulation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Chen et al., 2019] Chen, Y.-C., Xu, X., Tian, Z., and Jia, J. (2019). Homomorphic latent space interpolation for unpaired image-to-image translation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Choi et al., 2018] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. (2018). StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Chu et al., 2017] Chu, C., Zhmoginov, A., and Sandler, M. (2017). CycleGAN: a master of steganography. *arXiv preprint:1712.02950.*

[Cootes et al., 1998] Cootes, T. F., Edwards, G. J., and Taylor, C. J. (1998). Active appearance models. In *Computer Vision — ECCV'98*, pages 484–498, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Cottrell et al., 1987] Cottrell, G. W., Munro, P., and Zipser, D. (1987). Learning internal representations from gray-scale images: An example of extensional programming. In *Conference of the Cognitive Science Society.*

[Cremer et al., 2018] Cremer, C., Li, X., and Duvenaud, D. (2018). Inference suboptimality in variational autoencoders. In *The International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 1078–1086. PMLR.

[Cremer et al., 2017] Cremer, C., Morris, Q., and Duvenaud, D. (2017). Reinterpreting importance-weighted autoencoders. *Workshop track - The International Conference on Learning Representations (ICLR).*

[Criminisi et al., 2004] Criminisi, A., Perez, P., and Toyama, K. (2004). Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

[Dekel et al., 2018] Dekel, T., Gan, C., Krishnan, D., Liu, C., and Freeman, W. T. (2018). Sparse, smart contours to represent and edit images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Denton et al., 2015] Denton, E. L., Chintala, S., szlam, a., and Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1486–1494. Curran Associates, Inc.

[Dillon et al., 2017] Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., and Saurous, R. A. (2017). Tensorflow distributions. *arXiv preprint:1711.10604*.

[Dinh et al., 2017] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real NVP. *The International Conference on Learning Representations (ICLR)*.

[Doersch, 2016] Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint:1606.05908*.

[Donahue et al., 2017] Donahue, J., Krähenbühl, P., and Darrell, T. (2017). Adversarial feature learning. *The International Conference on Learning Representations (ICLR)*.

[Dorta et al., 2018a] Dorta, G., Campbell, N. D. F., and Simpson, I. (2018a). Method of modifying digital images. UK Patent, Application Number 1818759.1.

[Dorta et al., 2018b] Dorta, G., Vicente, S., Agapito, L., Campbell, N. D. F., and Simpson, I. (2018b). Structured uncertainty prediction networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Dorta et al., 2018c] Dorta, G., Vicente, S., Agapito, L., Campbell, N. D. F., and Simpson, I. (2018c). Training VAEs under structured residuals. *arXiv preprint:1804.01050*.

[Dorta et al., 2020] Dorta, G., Vicente, S., Campbell, N. D. F., and Simpson, J. A. I. (2020). The GAN that warped: Semantic attribute editing with unpaired data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Dumoulin et al., 2017] Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. (2017). Adversarially learned inference. *The International Conference on Learning Representations (ICLR).*

[Dziugaite et al., 2015] Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, pages 258–267. AUAI Press.

[Ekman and Friesen, 1976] Ekman, P. and Friesen, W. V. (1976). Measuring facial movement. *Environmental psychology and nonverbal behavior*, 1(1):56–75.

[Esmaeili et al., 2018] Esmaeili, B., Wu, H., Jain, S., Bozkurt, A., Siddharth, N., Paige, B., Brooks, D. H., Dy, J., and van de Meent, J.-W. (2018). Structured disentangled representations. *arXiv preprint:1804.02086.*

[Frankle and Carbin, 2019] Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *The International Conference on Learning Representations (ICLR).*

[Ganin et al., 2016] Ganin, Y., Kononenko, D., Sungatullina, D., and Lempitsky, V. (2016). DeepWarp: Photorealistic image resynthesis for gaze manipulation. In *The European Conference on Computer Vision (ECCV)*, pages 311–326, Cham. Springer International Publishing.

[Garrido et al., 2013] Garrido, P., Valgaerts, L., Wu, C., and Theobalt, C. (2013). Reconstructing detailed dynamic face geometry from monocular video. *ACM Transactions on Graphics (TOG)*, 32(6):158:1–158:10.

[Gatys et al., 2015] Gatys, L., Ecker, A. S., and Bethge, M. (2015). Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 262–270. Curran Associates, Inc.

[Gelman and Hill, 2007] Gelman, A. and Hill, J. (2007). *Data Analysis using Regression and Multilevel/Hierarchical Models.* Analytical methods for social research. Cambridge University Press.

[Geng et al., 2018] Geng, J., Shao, T., Zheng, Y., Weng, Y., and Zhou, K. (2018). Warp-guided GANs for single-photo facial animation. *ACM Transactions on Graphics (TOG)*, 37(6).

[Germain et al., 2015] Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). MADE: Masked autoencoder for distribution estimation. In *The International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889. PMLR.

[Gharbi et al., 2017] Gharbi, M., Chen, J., Barron, J. T., Hasinoff, S. W., and Durand, F. (2017). Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):118.

[Ghosh et al., 2018] Ghosh, A., Kulharia, V., Namboodiri, V. P., Torr, P. H., and Dokania, P. K. (2018). Multi-agent diverse generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Gilbert et al., 2018] Gilbert, A., Collomosse, J., Jin, H., and Price, B. (2018). Disentangling structure and aesthetics for style-aware image completion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Godard et al., 2017] Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Goodfellow, 2016] Goodfellow, I. (2016). NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint:1701.00160*.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680.

[Guenter et al., 1998] Guenter, B., Grimm, C., Wood, D., Malvar, H., and Pighin, F. (1998). Making faces. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 55–66.

[Gulrajani et al., 2017a] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017a). Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5767–5777. Curran Associates, Inc.

[Gulrajani et al., 2017b] Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., and Courville, A. (2017b). PixelVAE: A Latent Variable Model for Natural Images. *The International Conference on Learning Representations (ICLR).*

[Guo and Sim, 2009] Guo, D. and Sim, T. (2009). Digital face makeup by example. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 73–79.

[Guo et al., 2016] Guo, Y., Zhang, L., Hu, Y., He, X., and Gao, J. (2016). MS-Celeb-1M: A dataset and benchmark for large scale face recognition. In *The European Conference on Computer Vision (ECCV)*. Springer.

[Han et al., 2015] Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1135–1143. Curran Associates, Inc.

[Hassibi and Stork, 1993] Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems (NIPS)*, pages 164–171.

[Heusel et al., 2017] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6626–6637. Curran Associates, Inc.

[Higgins et al., 2017] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. *The International Conference on Learning Representations (ICLR).*

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366.

[Hou et al., 2017] Hou, X., Shen, L., Sun, K., and Qiu, G. (2017). Deep feature consistent variational autoencoder. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141.

[Huang et al., 2018] Huang, H., li, z., He, R., Sun, Z., and Tan, T. (2018). IntroVAE: Introspective variational autoencoders for photographic image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 52–63. Curran Associates, Inc.

[Huszar, 2017] Huszar, F. (2017). Gaussian distributions are soap bubbles. https://www.inference.vc/high-dimensional-gaussian-distributions-are-soap-bubble/.

[Ignatov et al., 2017] Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., and Van Gool, L. (2017). WESPE: weakly supervised photo enhancer for digital cameras. *arXiv preprint:1709.01118*.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *The International Conference on Machine Learning (ICML)*, pages 448–456.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Jancsary et al., 2012] Jancsary, J., Nowozin, S., Sharp, T., and Rother, C. (2012). Regression tree fields an efficient, non-parametric approach to image labeling problems. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2376–2383.

[Joshi et al., 2010] Joshi, N., Matusik, W., Adelson, E. H., and Kriegman, D. J. (2010). Personal photo enhancement using example images. *ACM Transactions on Graphics (TOG)*, 29(2):12:1–12:15.

[Karras et al., 2018] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. *The International Conference on Learning Representations (ICLR).*

[Karras et al., 2019] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Kemelmacher-Shlizerman et al., 2014] Kemelmacher-Shlizerman, I., Suwajanakorn, S., and Seitz, S. M. (2014). Illumination-aware age progression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems (NIPS).*

[Kim et al., 2018] Kim, H., Garrido, P., Tewari, A., Xu, W., Thies, J., Niessner, M., Pérez, P., Richardt, C., Zollhöfer, M., and Theobalt, C. (2018). Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4).

[Kim and Mnih, 2018] Kim, H. and Mnih, A. (2018). Disentangling by factorising. In *The International Conference on Machine Learning (ICML)*, volume 80, pages 2649–2658.

[King, 2009] King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research (JMLR)*, 10:1755–1758.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR).*

[Kingma and Dhariwal, 2018] Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10236–10245. Curran Associates, Inc.

[Kingma et al., 2016] Kingma, D. P., Salimans, T., and Welling, M. (2016). Improving variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems (NIPS).*

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *The International Conference on Learning Representations (ICLR).*

[Klushyn et al., 2019] Klushyn, A., Chen, N., Kurle, R., Cseke, B., and van der Smagt, P. (2019). Learning hierarchical priors in VAEs. *arXiv preprint:1905.04982*.

[Kumar et al., 2018] Kumar, A., Sattigeri, P., and Balakrishnan, A. (2018). Variational inference of disentangled latent concepts from unlabeled observations. In *The International Conference on Learning Representations (ICLR).*

[Langner et al., 2010] Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D. H. J., Hawk, S. T., and van Knippenberg, A. (2010). Presentation and validation of the Radboud faces database. *Cognition and Emotion*, 24(8):1377–1388.

[Larsen et al., 2016] Larsen, A. B. L., Sønderby, S. K., and Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In *The International Conference on Machine Learning (ICML)*, volume 48, pages 1558–1566. JMLR.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[LeCun et al., 1990] LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in Neural Information Processing Systems (NIPS)*, pages 598–605.

[Ledig et al., 2017] Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Li et al., 2017a] Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Poczos, B. (2017a). MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2203–2213. Curran Associates, Inc.

[Li et al., 2017b] Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. (2017b). Pruning filters for efficient convnets. *The International Conference on Learning Representations (ICLR).*

[Li et al., 2015] Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *The International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 1718–1727, Lille, France. PMLR.

[Lin et al., 2018] Lin, C.-H., Yumer, E., Wang, O., Shechtman, E., and Lucey, S. (2018). ST-GAN: Spatial transformer generative adversarial networks for image compositing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Liu et al., 2018] Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., and Catanzaro, B. (2018). Image inpainting for irregular holes using partial convolutions. In *The European Conference on Computer Vision (ECCV).*

[Liu et al., 2015] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *The IEEE International Conference on Computer Vision (ICCV).*

[Liu et al., 2001] Liu, Z., Shan, Y., and Zhang, Z. (2001). Expressive expression mapping with ratio images. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 271–276. ACM.

[Lopez et al., 2018] Lopez, R., Regier, J., Jordan, M. I., and Yosef, N. (2018). Information constraints on auto-encoding variational bayes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6117–6128. Curran Associates, Inc.

[Luc et al., 2016] Luc, P., Couprie, C., Chintala, S., and Verbeek, J. (2016). Semantic Segmentation using Adversarial Networks. In *NIPS Workshop on Adversarial Training*, Barcelona, Spain.

[Lucic et al., 2018] Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2018). Are GANs created equal? a large-scale study. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 700–709. Curran Associates, Inc.

[Lunn et al., 2012] Lunn, D., Jackson, C., Best, N., Spiegelhalter, D., and Thomas, A. (2012). *The BUGS book: A practical introduction to Bayesian analysis.* Chapman and Hall/CRC.

[Ma and Deng, 2018] Ma, L. and Deng, Z. (2018). Real-time facial expression transformation for monocular RGB video. *Computer Graphics Forum*, 38(1):470–481.

[MacKay, 1992] MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

[Mackay, 1995] Mackay, D. J. C. (1995). Probable networks and plausible predictions a review of practical bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505.

[Makhzani and Frey, 2017] Makhzani, A. and Frey, B. J. (2017). PixelGAN autoencoders. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1975–1985. Curran Associates, Inc.

[Mejjati et al., 2018] Mejjati, Y. A., Richardt, C., Tompkin, J., Cosker, D., and Kim, K. I. (2018). Unsupervised attention-guided image to image translation. In *Advances in Neural Information Processing Systems (NeurIPS).* Curran Associates, Inc.

[Mescheder et al., 2017] Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *The International Conference on Machine Learning (ICML).*

[Metz et al., 2017] Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2017). Unrolled generative adversarial networks. *The International Conference on Learning Representations (ICLR).*

[Miyato et al., 2018] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *The International Conference on Learning Representations (ICLR).*

[Mohammed et al., 2009] Mohammed, U., Prince, S. J., and Kautz, J. (2009). Visiolization: generating novel facial images. *ACM Transactions on Graphics (TOG)*, 28(3):57.

[Mori et al., 2012] Mori, M., MacDorman, K. F., and Kageki, N. (2012). The uncanny valley [from the field]. *IEEE Robotics Automation Magazine*, 19(2):98–100.

[Müller, 1997] Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429443.

[Murphy, 2012] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective.* The MIT press.

[Nagano et al., 2018] Nagano, K., Seo, J., Xing, J., Wei, L., Li, Z., Saito, S., Agarwal, A., Fursund, J., and Li, H. (2018). PaGAN: Real-time avatars using dynamic textures. *ACM Transactions on Graphics (TOG)*, 37(6).

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *The International Conference on Machine Learning (ICML)*, pages 807–814.

[Nakagami, 1960] Nakagami, M. (1960). The m-distributiona general formula of intensity distribution of rapid fading. In Hoffman, W., editor, *Statistical Methods in Radio Wave Propagation*, pages 3 – 36. Pergamon.

[Neal, 2012] Neal, R. M. (2012). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

[Nguyen et al., 2008] Nguyen, M. H., Lalonde, J.-F., Efros, A. A., and De la Torre, F. (2008). Image-based shaving. *Computer Graphics Forum*, 27(2):627–635.

[Nikias and Pan, 1988] Nikias, C. L. and Pan, R. (1988). Time delay estimation in unknown gaussian spatially correlated noise. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(11):1706–1714.

[Nowozin et al., 2016] Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 271–279. Curran Associates, Inc.

[Odena et al., 2017] Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier GANs. In *ICML*, volume 70, pages 2642–2651. JMLR.

[Oord et al., 2016] Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. *The International Conference on Machine Learning (ICML).*

[Pathak et al., 2016] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Pérez et al., 2003] Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Transactions on Graphics (TOG)*, 22(3):313–318.

[Portenier et al., 2018] Portenier, T., Hu, Q., Szabó, A., Bigdeli, S. A., Favaro, P., and Zwicker, M. (2018). Faceshop: Deep sketch-based face image editing. *ACM Transactions on Graphics (TOG)*, 37(4):99:1–99:13.

[Prokudin et al., 2018] Prokudin, S., Gehler, P., and Nowozin, S. (2018). Deep directional statistics: Pose estimation with uncertainty quantification. In *The European Conference on Computer Vision (ECCV).*

[Pu et al., 2017a] Pu, Y., Gan, Z., Henao, R., Li, C., Han, S., and Carin, L. (2017a). VAE learning via Stein variational gradient descent. *Advances in Neural Information Processing Systems (NIPS).*

[Pu et al., 2017b] Pu, Y., Wang, W., Henao, R., Chen, L., Gan, Z., Li, C., and Carin, L. (2017b). Adversarial symmetric variational autoencoder. *Advances in Neural Information Processing Systems (NIPS).*

[Pumarola et al., 2018] Pumarola, A., Agudo, A., Martinez, A., Sanfeliu, A., and Moreno-Noguer, F. (2018). GANimation: Anatomically-aware facial animation from a single image. In *The European Conference on Computer Vision (ECCV).*

[Qian et al., 2019] Qian, S., Lin, K.-Y., Wu, W., Liu, Y., Wang, Q., Shen, F., Qian, C., and He, R. (2019). Make a face: Towards arbitrary high fidelity face manipulation. In *The IEEE International Conference on Computer Vision (ICCV)*.

[Rabin et al., 2012] Rabin, J., Peyré, G., Delon, J., and Bernot, M. (2012). Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*, pages 435–446, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Radford et al., 2016] Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *The International Conference on Learning Representations (ICLR)*.

[Rainforth et al., 2018] Rainforth, T., Kosiorek, A., Le, T. A., Maddison, C., Igl, M., Wood, F., and Teh, Y. W. (2018). Tighter variational bounds are not necessarily better. In *The International Conference on Machine Learning (ICML)*, volume 80, pages 4277–4285.

[Ranzato et al., 2007] Ranzato, M., Huang, F.-J., Boureau, Y.-L., and LeCun., Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT press Cambridge.

[Reed et al., 2017] Reed, S., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Chen, Y., Belov, D., and de Freitas, N. (2017). Parallel multiscale autoregressive density estimation. In *The International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 2912–2921. PMLR.

[Rezende and Mohamed, 2015] Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. *The International Conference on Machine Learning (ICML)*.

[Rezende et al., 2014] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *The International Conference on Machine Learning (ICML)*.

[Richardson et al., 2016] Richardson, E., Sela, M., and Kimmel, R. (2016). 3d face reconstruction by learning from synthetic data. In *Fourth International Conference on 3D Vision (3DV)*, pages 460–469.

[Rissanen, 1978] Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465 – 471.

[Salimans et al., 2016] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. (2016). Improved techniques for training GANs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2234–2242. Curran Associates, Inc.

[Salimans et al., 2017] Salimans, T., Karpathy, A., Chen, X., and Kingma, D. (2017). Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. *The International Conference on Learning Representations (ICLR).*

[Sanchez and Valstar, 2018] Sanchez, E. and Valstar, M. (2018). Triple consistency loss for pairing distributions in gan-based face synthesis. *arXiv preprint1811.03492.*

[Sánchez Pérez et al., 2013] Sánchez Pérez, J., Meinhardt-Llopis, E., and Facciolo, G. (2013). TV-L1 Optical Flow Estimation. *Image Processing On Line*, 3:137–150.

[Schroff et al., 2015] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Schwarz, 1978] Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

[Shen and Liu, 2017] Shen, W. and Liu, R. (2017). Learning residual images for face attribute manipulation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Shih et al., 2014] Shih, Y., Paris, S., Barnes, C., Freeman, W. T., and Durand, F. (2014). Style transfer for headshot portraits. *ACM Transactions on Graphics (TOG)*, 33(4):148:1–148:14.

[Shu et al., 2018] Shu, Z., Sahasrabudhe, M., Alp Guler, R., Samaras, D., Paragios, N., and Kokkinos, I. (2018). Deforming autoencoders: Unsupervised disentangling of shape and appearance. In *The European Conference on Computer Vision (ECCV).*

[Shu et al., 2016] Shu, Z., Shechtman, E., Samaras, D., and Hadap, S. (2016). Eye-Opener: Editing eyes in the wild. *ACM Transactions on Graphics (TOG)*, 36(1):1:1–1:13.

[Shu et al., 2017] Shu, Z., Yumer, E., Hadap, S., Sunkavalli, K., Shechtman, E., and Samaras, D. (2017). Neural face editing with intrinsic image disentangling. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Smith, 2013] Smith, C. (2013). Facebook users are uploading 350 million new photos each day. https://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9?IR=T.

[Smith and Hocking, 1972] Smith, W. B. and Hocking, R. R. (1972). Algorithm AS 53: Wishart variate generator. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 21(3):341–345.

[Smolensky, 1986] Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge.

[Srivastava et al., 2017] Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. (2017). VEEGAN: Reducing mode collapse in GANs using implicit variational learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3308–3318. Curran Associates, Inc.

[Tewari et al., 2017] Tewari, A., Zollhofer, M., Kim, H., Garrido, P., Bernard, F., Perez, P., and Theobalt, C. (2017). MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *The IEEE International Conference on Computer Vision (ICCV)*.

[Theis et al., 2016] Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. *The International Conference on Learning Representations (ICLR)*.

[Thies et al., 2016] Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C., and Niessner, M. (2016). Face2Face: Real-time face capture and reenactment of RGB videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[Tipping and Bishop, 1999] Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(3):611–622.

[Tomczak and Welling, 2018] Tomczak, J. and Welling, M. (2018). VAE with a VampPrior. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR.

[Tschannen et al., 2018] Tschannen, M., Bachem, O. F., and Lui, M. (2018). Recent advances in autoencoder-based representation learning. In *Bayesian Deep Learning Workshop, NeurIPS*.

[Tuan Tran et al., 2017] Tuan Tran, A., Hassner, T., Masi, I., and Medioni, G. (2017). Regressing robust and discriminative 3D morphable models with a very deep neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Turk and Pentland, 1991] Turk, M. A. and Pentland, A. P. (1991). Face recognition using eigenfaces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–591. IEEE.

[Upchurch et al., 2017] Upchurch, P., Gardner, J., Pleiss, G., Pless, R., Snavely, N., Bala, K., and Weinberger, K. (2017). Deep Feature Interpolation for image content changes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Vincent et al., 2008] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *The International Conference on Machine Learning (ICML)*, pages 1096–1103. ACM.

[Wah et al., 2011] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 Dataset. Technical report.

[Walker et al., 2016] Walker, J., Doersch, C., Gupta, A., and Hebert, M. (2016). An uncertain future: Forecasting from variational autoencoders. In *The European Conference on Computer Vision (ECCV)*.

[Wallace, 1992] Wallace, G. K. (1992). The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv.

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

[Wolberg, 1998] Wolberg, G. (1998). Image morphing: a survey. *The visual computer*, 14(8-9):360–372.

[Woolrich et al., 2001] Woolrich, M. W., Ripley, B. D., Brady, M., and Smith, S. M. (2001). Temporal autocorrelation in univariate linear modeling of FMRI data. *NeuroImage*, 14(6):1370 – 1386.

[Wu et al., 2019] Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernandez-Lobato, J. M., and Gaunt, A. L. (2019). Deterministic variational inference for robust bayesian neural networks. *The International Conference on Learning Representations (ICLR).*

[Xu et al., 2019] Xu, H., Chen, W., Lai, J., Li, Z., Zhao, Y., and Pei, D. (2019). On the necessity and effectiveness of learning the prior of variational auto-encoder. *arXiv preprint:1905.13452.*

[Yan et al., 2016] Yan, X., Yang, J., Sohn, K., and Lee, H. (2016). Attribute2Image: Conditional image generation from visual attributes. *The European Conference on Computer Vision (ECCV).*

[Yang et al., 2017] Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., and Li, H. (2017). High-resolution image inpainting using multi-scale neural patch synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[Yang et al., 2011] Yang, F., Wang, J., Shechtman, E., Bourdev, L., and Metaxas, D. (2011). Expression flow for 3D-aware face component transfer. *ACM Transactions on Graphics (TOG)*, 30(4):60:1–60:10.

[Yeh et al., 2016] Yeh, R. A., Liu, Z., Goldman, D. B., and Agarwala, A. (2016). Semantic facial expression editing using autoencoded flow. *arXiv preprint:1611.09961.*

[Yu and Koltun, 2015] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *The International Conference on Learning Representations (ICLR).*

[Yu et al., 2015] Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint:1506.03365.*

[Zach et al., 2007] Zach, C., Pock, T., and Bischof, H. (2007). A duality based approach for realtime TV-L1 optical flow. In *Pattern Recognition*, pages 214–223, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Zakharov et al., 2019] Zakharov, E., Shysheya, A., Burkov, E., and Lempitsky, V. (2019). Few-shot adversarial learning of realistic neural talking head models. In *The IEEE International Conference on Computer Vision (ICCV).*

[Zhang et al., 2018] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. (2018). Self-attention generative adversarial networks. *arXiv preprint:1805.08318.*

[Zhao et al., 2017a] Zhao, J., Mathieu, M., and LeCun, Y. (2017a). Energy-based generative adversarial network. *The International Conference on Learning Representations (ICLR).*

[Zhao et al., 2017b] Zhao, S., Song, J., and Ermon, S. (2017b). Infovae: Information maximizing variational autoencoders. *arXiv preprint:1706.02262.*

[Zhou et al., 2016] Zhou, T., Tulsiani, S., Sun, W., Malik, J., and Efros, A. A. (2016). View synthesis by appearance flow. In *The European Conference on Computer Vision (ECCV)*, pages 286–301.

[Zhu et al., 2016] Zhu, J.-Y., Krähenbühl, P., Shechtman, E., and Efros, A. A. (2016). Generative visual manipulation on the natural image manifold. In *The European Conference on Computer Vision (ECCV)*, pages 597–613, Cham. Springer International Publishing.

[Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV).*

[Zollhöfer et al., 2018] Zollhöfer, M., Thies, J., Garrido, P., Bradley, D., Beeler, T., Pérez, P., Stamminger, M., Nießner, M., and Theobalt, C. (2018). State of the art on monocular 3D face reconstruction, tracking, and applications. *Computer Graphics Forum*, 37(2):523–550.

Appendix A

Structured uncertainty

A.1 Proofs and derivations

A.1.1 Gaussian Markov Random Fields

Equivalence to multivariate Gaussian likelihood

First we state in more detail the log probability in a G-CRF model, which is usually defined in terms of pair-wise and unary potential functions:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \propto \sum_{i=0}^{n-1} \sum_{j \in N_i} \psi_{ij}(x_i, x_j, \mathbf{z}) + \sum_{j=0}^{n-1} \psi_j(x_j, \mathbf{z}), \tag{A.1}$$

$$\psi_{ij}(x_i, x_j, \mathbf{z}) = -\frac{1}{2} x_i \left(\lambda(\mathbf{z})_{i,j}\right) x_j, \quad s.t. \quad \lambda(\mathbf{z})_{i,j} = \lambda(\mathbf{z})_{j,i} \tag{A.2}$$

$$\psi_j(x_j, \mathbf{z}) = -\frac{1}{2} \left(\lambda(\mathbf{z})_{j,j}\right) x_j^2 + \left(\eta(\mathbf{z})_j\right) x_j, \tag{A.3}$$

where x_i and x_j denote individual pixels, $\lambda_{i,j}(\mathbf{z})$ and $\eta_j(\mathbf{z})$ are functions parametrised by $\boldsymbol{\theta}$, N_i is the set of all neighbours of pixel x_i, $\mathbf{z}$ is the conditioning input vector, and $\psi_{ij}(\cdot)$ and $\psi_j(\cdot)$ are known as potential functions. The parametric functions, $\lambda_{i,j}(\mathbf{z})$ and $\eta_j(\mathbf{z})$, can be implemented as neural networks, where the model optimises their parameters $\boldsymbol{\theta}$. The η_j, $\lambda_{i,j}$ and $\lambda_{j,j}$ terms can be accumulated in matrix form as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \propto \left(\boldsymbol{\eta}(\mathbf{z})\right)^{\mathsf{T}} \mathbf{x} - \frac{1}{2} \mathbf{x}^{\mathsf{T}} \left(\boldsymbol{\Lambda}(\mathbf{z})\right) \mathbf{x}, \tag{A.4}$$

which is the log probability described in equation 3.20.

To see how the log likelihood of a Gaussian MRF, equation A.4, is proportional to that of a Multivariate Gaussian distribution, equation 3.6, we first need to define the log probability of a Multivariate Gaussian distribution in canonical form [Murphy, 2012]. The log probability with respect to the canonical parameters $\boldsymbol{\eta} \triangleq \boldsymbol{\Lambda}\boldsymbol{\mu}$ and $\boldsymbol{\Lambda} \triangleq \boldsymbol{\Sigma}^{-1}$ is defined as

$$\log \mathcal{N}^c(\mathbf{x}|\boldsymbol{\eta}, \boldsymbol{\Lambda}) = c + \frac{1}{2}\left(\log\left(|\boldsymbol{\Lambda}|\right) - \mathbf{x}^{\mathsf{T}}\boldsymbol{\Lambda}\mathbf{x} - \boldsymbol{\eta}^{\mathsf{T}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\eta} + 2\mathbf{x}^{\mathsf{T}}\boldsymbol{\eta}\right), \tag{A.5}$$

where $c = -\frac{n}{2}\log(2\pi)$ is a constant term, and the ^{c} in $\mathcal{N}^c(\cdot)$ is used to distinguish this parametrisation from the standard one with $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

The log probability of the Gaussian MRF in equation A.4, can be rewritten as

$$\log p(\mathbf{x})_{\boldsymbol{\theta}} \propto \boldsymbol{\eta}^{\mathsf{T}}\mathbf{x} - \frac{1}{2}\mathbf{x}^{\mathsf{T}}\boldsymbol{\Lambda}\mathbf{x} = \frac{1}{2}\left(-\mathbf{x}^{\mathsf{T}}\boldsymbol{\Lambda}\mathbf{x} + 2\mathbf{x}^{\mathsf{T}}\boldsymbol{\eta}\right), \tag{A.6}$$

where we dropped the dependency of $\mathbf{z}$ for clarity of notation. It can be seen that all the data terms, *i.e.* those involving $\mathbf{x}$, in the $\log \mathcal{N}^c(\mathbf{x}|\boldsymbol{\eta}, \boldsymbol{\Lambda})$, defined above appear in the log probability of the Gaussian MRF. Therefore, the Gaussian MRF is equivalent to a multivariate Gaussian, up to a normalisation factor.

Conditional independence proof

The conditional independence property in Gaussian CRF is known [Murphy, 2012] and we reproduce it here for completeness. The conditional independence assertion in equation 3.22 is stated with respected to sets x_A and x_B. For simplicity, in this proof we assume sets of size one, x_j and x_k, as the generalisation to larger size sets is straightforward. In other words, we have three pixels, where x_j is the pixel connected to x_i with $\lambda_{i,j} \neq 0$, and x_k is the pixel that is not connected to x_i with $\lambda_{i,k} = 0$. We also redefine the potential functions in the exponential space, $\tilde{\psi}_{ij} = e^{\psi_{ij}}$, and $\tilde{\psi}_j = e^{\psi_j}$, as this will simplify the notation for the proof.

We start by using the functional conditional independence theorem [Murphy, 2012]. The theorem states that x_i and x_k are conditionally independent given x_j, which can be written as

$$p(x_i, x_j, x_k) = p(x_i|x_j)p(x_j, x_k), \tag{A.7}$$

if the joint probability can be expressed in terms of functions as

$$p(x_i, x_k | x_j) = \tilde{\psi}_{ij}(x_i, x_j)\tilde{\psi}_{jk}(x_j, x_k) \quad s.t. \quad p(x_j) > 0. \tag{A.8}$$

This is the case for $\lambda_{i,k} = 0$, and it can be proved by noting that Markov Random Fields are constrained to pairwise connections. Thus, we can write

$$p(x_i, x_k | x_j) = p(x_i | x_j, x_k)p(x_k | x_j), \tag{A.9}$$

$$p(x_i, x_k | x_j) \propto \tilde{\psi}_{ik}(x_i, x_k)\tilde{\psi}_{ij}(x_i, x_j)\tilde{\psi}_{jk}(x_j, x_k). \tag{A.10}$$

Employing the definition for pairwise potentials in equation A.2 and given the aforementioned assumption that $\lambda_{i,k} = 0$, we see that $\psi_{ik}(x_i, x_k) = 0$, and equivalently, $\tilde{\psi}_{ik}(x_i, x_k) = 1$. Therefore, the term cancels out in the equation above, leading to

$$p(x_i, x_k | x_j) \propto \tilde{\psi}_{ij}(x_i, x_j)\tilde{\psi}_{jk}(x_j, x_k), \tag{A.11}$$

which has the same form as equation A.8, thus concluding the proof.

A.1.2 Directly modelling the Cholesky decomposition of the covariance matrix

If the covariance network only explicitly estimates the non-zero elements in $\mathbf{M}$, sampling from $\mathbf{\Sigma}$ is computed as a simple matrix-vector multiplication, $\boldsymbol{\epsilon} = \mathbf{M}\mathbf{u}$. However, the reconstruction error

$$\mathbf{r}^{\mathsf{T}} \left(\mathbf{\Sigma}^{-1}\right) \mathbf{r} = \mathbf{w}^{\mathsf{T}}\mathbf{w}, \tag{A.12}$$

requires solving a system of equations on each training step as

$$\mathbf{M}^{\mathsf{T}}\mathbf{w} = \mathbf{r}, \tag{A.13}$$

where the system is solved for $\mathbf{w}$. The log determinant is computed as

$$\log(|\mathbf{\Sigma}|) = 2\sum_{i=0}^{n-1} \log(m_{ii}), \tag{A.14}$$

where m_{ii} is a diagonal element in $\mathbf{M}$.

Due to the quadratic increase in the number of elements in $\mathbf{\Sigma}$, this approach is only tractable if a sparsity pattern similar to the one discussed in Section 3.3.3 is applied in $\mathbf{M}$. This assumes that deep learning frameworks that support sparse tensors and sparse system of equations solvers. Withal, the main limitation in a sparse approach applied on $\mathbf{M}$ is that long range correlations could no longer be modelled, as the sparsity pattern in $\mathbf{M}$ is propagated to $\mathbf{\Sigma}$ by construction.

A.1.3 Example of operator $s(\cdot)$

In this section we show how the $s(\cdot)$ operator modifies a dense matrix, where the operator includes padding and shifting. The operator can be understood as working in two stages. In the first, it generates an intermediate matrix by padding with zeros the input matrix, such that the sparsity pattern in the first column of $\mathbf{L}$ is replicated in each column of the output matrix of the first stage. In the second, it reorders the padded matrix, which is indexed by i, j into an output matrix indexed by k, m as follows

$$k = i + j \quad \text{and} \quad m = j, \tag{A.15}$$

where any k, m index in the output that would require a negative i is set to zero. In practice, the output matrix is constructed directly, without the intermediate matrix.

Below, we provide an example for a 4 × 4 grey-scale image, with a neighbourhood size of $n_\mathrm{f} = 3$ and $n_\mathrm{b} = 4$ is the number of vectors in the basis matrix. The input image is defined as

$$f(\mathbf{x}) = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} \\ x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,0} & x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix}.$$

The dense basis matrix is defined as

$$\mathbf{B} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ b_{4,0} & b_{4,1} & b_{4,2} & b_{4,3} \end{bmatrix}.$$

The dense weight matrix is defined as

$$\mathbf{W} = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} & w_{0,4} & w_{0,5} & w_{0,6} & w_{0,7} & w_{0,8} & w_{0,9} & w_{0,10} & w_{0,11} & w_{0,12} & w_{0,13} & w_{0,14} & w_{0,15} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} & w_{1,6} & w_{1,7} & w_{1,8} & w_{1,9} & w_{1,10} & w_{1,11} & w_{1,12} & w_{1,13} & w_{1,14} & w_{1,15} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} & w_{2,6} & w_{2,7} & w_{1,8} & w_{2,9} & w_{2,10} & w_{2,11} & w_{2,12} & w_{2,13} & w_{2,14} & w_{2,15} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} & w_{3,6} & w_{3,7} & w_{3,8} & w_{3,9} & w_{3,10} & w_{3,11} & w_{3,12} & w_{3,13} & w_{3,14} & w_{3,15} \end{bmatrix}.$$

The dense matrix $\mathbf{T} = \mathbf{BW}$, contains the non-zero values in $\mathbf{L}$. Therefore, after the matrix multiplication the values that would correspond to neighbours outside of the image are zeroed out as

$$\mathbf{T} = \begin{bmatrix} t_{0,0} & t_{0,1} & t_{0,2} & t_{0,3} & t_{0,4} & t_{0,5} & t_{0,6} & t_{0,7} & t_{0,8} & t_{0,9} & t_{0,10} & t_{0,11} & t_{0,12} & t_{0,13} & t_{0,14} & t_{0,15} \\ t_{1,0} & t_{1,1} & t_{1,2} & 0 & t_{1,4} & t_{1,5} & t_{1,6} & 0 & t_{1,8} & t_{1,9} & t_{1,10} & 0 & t_{1,12} & t_{1,13} & t_{1,14} & 0 \\ 0 & t_{2,1} & t_{2,2} & t_{2,3} & 0 & t_{2,5} & t_{2,6} & t_{2,7} & 0 & t_{2,9} & t_{2,10} & t_{2,11} & 0 & 0 & 0 & 0 \\ t_{3,0} & t_{3,1} & t_{3,2} & t_{3,3} & t_{3,4} & t_{3,5} & t_{3,6} & t_{3,7} & t_{3,8} & t_{3,9} & t_{3,10} & t_{3,11} & 0 & 0 & 0 & 0 \\ t_{4,0} & t_{4,1} & t_{4,2} & 0 & t_{4,4} & t_{4,5} & t_{4,6} & 0 & t_{4,8} & t_{4,9} & t_{4,10} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In practice, for the square error evaluation this process is done by padding the images with zeroes on the borders, as it is standard for 2D convolutions in deep learning frameworks.

The sparse matrix $\mathbf{L}$ is constructed after applying the $s(\cdot)$ operator, as $\mathbf{L} = s(\mathbf{T})$,

$$\mathbf{L} = \left[\begin{matrix} t_{0,0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_{1,0} & t_{0,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_{1,1} & t_{0,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t_{1,2} & t_{0,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_{3,0} & t_{2,1} & 0 & 0 & t_{0,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_{4,0} & t_{3,1} & t_{2,2} & 0 & t_{1,4} & t_{0,5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_{4,1} & t_{3,2} & t_{2,3} & 0 & t_{1,5} & t_{0,6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t_{4,2} & t_{3,3} & 0 & 0 & t_{1,6} & t_{0,7} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_{3,4} & t_{2,5} & 0 & 0 & t_{0,8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_{4,4} & t_{3,5} & t_{2,6} & 0 & t_{1,8} & t_{0,9} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t_{4,5} & t_{3,6} & t_{2,7} & 0 & t_{1,9} & t_{0,10} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & t_{4,6} & t_{3,7} & 0 & 0 & t_{1,10} & t_{0,11} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{3,8} & t_{2,9} & 0 & 0 & t_{0,12} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{4,8} & t_{3,9} & t_{2,10} & 0 & t_{1,12} & t_{0,13} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{4,9} & t_{3,10} & t_{2,11} & 0 & t_{1,13} & t_{0,14} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{4,10} & t_{3,11} & 0 & 0 & t_{1,14} & t_{0,15} \end{matrix}\right] \begin{matrix} x_{0,0} \\ x_{0,1} \\ x_{0,2} \\ x_{0,3} \\ x_{1,0} \\ x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_{2,0} \\ x_{2,1} \\ x_{2,2} \\ x_{3,3} \\ x_{3,0} \\ x_{3,1} \\ x_{3,2} \\ x_{3,3} \end{matrix},$$

where on the right side of the matrix the vectorisation of the input image, $f(\mathbf{x})$, into $\mathbf{x}$ is shown.

A.1.4 Example of operator $g(\mathbf{I})$

In this section we show how the $g(\mathbf{I})$ operator modifies a dense identity matrix, to generate a kernel tensor that can be used in a 2D convolution. The operator transform a $n_\mathrm{c} \times n_\mathrm{c}$ matrix into a $n_\mathrm{f} \times n_\mathrm{f} \times n_\mathrm{c}$ tensor. For this example we assume that $n_\mathrm{f} = 3$

and $\mathbf{B} = \mathbf{I}$. Therefore, $\mathbf{I}$ has a shape of 5×5, and $g(\mathbf{I})$ is a tensor of shape $3 \times 3 \times 5$ and with values

$$g(\mathbf{I}) = \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right\}. \tag{A.16}$$

Note that when this kernel is convolved with an input image of shape $n_\mathrm{h} \times n_\mathrm{w} \times 1$, a tensor of shape $n_\mathrm{h} \times n_\mathrm{w} \times 5$ is created. The image corresponding to the first channel in the tensor is a copy of the input image, and the image in the second channel corresponds to the input shifted one pixel to the left. Similarly, the third is shifted one pixel to the right and one pixel to the top, the fourth is shifted one pixel to the top, and the fifth is shifted one pixel to the left and one the top.

A.1.5 Sampling from a multivariate Gaussian distribution

In this section we show how to draw samples from a multivariate Gaussian distribution $\mathbf{x} \sim \mathcal{N}\big(\boldsymbol{\mu}, \boldsymbol{\Sigma}\big)$, using the Cholesky matrix $\mathbf{L}$. The covariance matrix, $\boldsymbol{\Sigma}$, can be expressed in terms of the Cholesky factors of the precision matrix,

$$\boldsymbol{\Sigma} = \mathbf{M}\mathbf{M}^{\mathsf{T}} = \boldsymbol{\Lambda}^{-1} = (\mathbf{L}\mathbf{L}^{\mathsf{T}})^{-1} = (\mathbf{L}^{\mathsf{T}})^{-1}\mathbf{L}^{-1}. \tag{A.17}$$

Thus, we can substitute $(\mathbf{L}^{\mathsf{T}})^{-1}$ for $\mathbf{M}$ in the sampling operation

$$\begin{aligned} \mathbf{x} &= \boldsymbol{\mu} + \boldsymbol{\epsilon} && \text{(A.18)} \\ &= \boldsymbol{\mu} + \mathbf{M}\mathbf{u} && \text{(A.19)} \\ &= \boldsymbol{\mu} + (\mathbf{L}^{\mathsf{T}})^{-1}\mathbf{u}, && \text{(A.20)} \end{aligned}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}\big(\mathbf{0}, \boldsymbol{\Sigma}\big)$ and $\mathbf{u} \sim \mathcal{N}\big(\mathbf{0}, \mathbf{I}\big)$.

The second term in the right hand side can be rearranged as

$$\begin{aligned} (\mathbf{L}^{\mathsf{T}})^{-1}\mathbf{u} &= \boldsymbol{\epsilon} && \text{(A.21)} \\ \mathbf{L}^{\mathsf{T}}\boldsymbol{\epsilon} &= \mathbf{u} && \text{(A.22)} \end{aligned}$$

which is the system that is solved for $\boldsymbol{\epsilon}$ to sample from the covariance matrix.

Note that $\mathbf{M}\mathbf{u} \neq (\mathbf{L}^{\mathsf{T}})^{-1}\mathbf{u}$, as $\mathbf{M}$ is lower triangular and $(\mathbf{L}^{\mathsf{T}})^{-1}$ is upper triangular. This means that although they lead to the same covariance matrix, samples drawn for the same $\mathbf{u}$ will generate different $\mathbf{x}$. In other words, both are valid Cholesky decompositions of $\mathbf{\Sigma}$, with one following the convention that the left matrix in the decomposition should be lower triangular, and the other assuming it should be upper triangular.

A.1.6 Equivalence between Cholesky-Wishart and Gamma-Gaussian distributions

We start the proof by defining how to draw random samples from a Wishart distribution. This is usually done via a Bartlett decomposition [Smith and Hocking, 1972] as

$$\mathbf{\Lambda} = \mathbf{C}\mathbf{D}\mathbf{D}^{\mathsf{T}}\mathbf{C}^{\mathsf{T}} \sim W(\mathbf{\Lambda}|\mathbf{V}, p), \tag{A.23}$$

where $\mathbf{C}$ is a Cholesky factor, $\mathbf{V} = \mathbf{C}\mathbf{C}^{\mathsf{T}}$, of the scale matrix $\mathbf{V}$, and $\mathbf{D}$ is a lower triangular matrix. Each element in the diagonal of $\mathbf{D}$ is an independent sample from a square root Gamma distribution, and the off-diagonal values are independently sampled from a Gaussian as

$$p(d_{i,i}) = \mathrm{Ga}^{\frac{1}{2}}\left(0.5\left(\frac{i(1-n)}{n-1}+p\right), 0.5\right), \tag{A.24}$$

$$p(d_{i,j}) = \mathcal{N}\big(0, 1\big), \quad \forall i, j \quad \text{s.t.} \quad i \neq j \quad \text{and} \quad j < i, \tag{A.25}$$

where $i \in [0, 1, \cdots, n-1]$, and the square root Gamma distribution is defined in Section A.1.7.

A sample from the Cholesky-Wishart distribution is similarly defined as

$$\mathbf{L} = \mathbf{C}\mathbf{D} \sim W^{c}(\mathbf{\Lambda}|\mathbf{V}, p). \tag{A.26}$$

We consider only diagonal scale matrices, $\mathbf{V}$, where the corresponding Cholesky factor is $\mathbf{C} = \sqrt{\mathbf{V}}$, where $\sqrt{\mathbf{V}}$ denotes the element-wise application of the square root operation to the matrix $\mathbf{V}$. Substituting this matrix in the sampling equation of the Cholesky-Wishart distribution leads to

$$\mathbf{L} = \mathbf{C}\mathbf{D} = \sqrt{\mathbf{V}}\mathbf{D}. \tag{A.27}$$

This equation can be simplified as

$$\mathbf{L} = \mathbf{D}, \tag{A.28}$$

by absorbing the $\sqrt{\mathbf{V}}$ matrix in each distribution as

$$p(d_{i,i}) = \mathrm{Ga}^{\frac{1}{2}}\left(0.5\left(\frac{i(1-n)}{n-1}+p\right), \frac{0.5}{v_{i,i}}\right), \tag{A.29}$$

$$p(d_{i,j}) = \mathcal{N}\big(0, v_{i,i}\big), \quad \forall i,j \quad \text{s.t.} \quad i \neq j \quad \text{and} \quad j < i. \tag{A.30}$$

Proof of how scaling a random variable is equivalent to this parametrisation is provided in Section A.1.8.

By inspection, the sample matrix $\mathbf{L} = \mathbf{D}$ is Gamma-Gaussian distributed. Therefore, if the samples from a Cholesky-Wishart distribution follow a Gamma-Gaussian distribution, the probability density function of both must be equal too, thus concluding the proof. Note that no samples are drawn when using the priors, they are only used for this proof.

A.1.7 Derivation of square root Gamma distribution

We start with the standard definition of the probability of a Gamma distribution:

$$\mathrm{Ga}\big(x|a,b\big) = \frac{b^a x^{a-1} e^{-bx}}{\mathbf{\Gamma}(a)}, \tag{A.31}$$

$$\mathbf{\Gamma}(a) = \int_0^\infty t^{a-1} e^{-t} dt, \tag{A.32}$$

where the distribution is parametrised by two scalars, where a_i is the shape and b_i is known as the rate, and both parameters are required to be positive.

A change of variables $q(y) = x = y^2$ and $q^{-1}(x) = y = \sqrt{x}$ is applied on the Gamma distribution, which leads to the square root Gamma distribution, with a probability density function defined by

$$\mathrm{Ga}^{\frac{1}{2}}\left(y|a,b\right) = \mathrm{Ga}\big(q(y)|a,b\big)\left|\frac{\partial(q(y))}{\partial y}\right| \tag{A.33}$$

$$= \mathrm{Ga}\big(y^2|a,b\big)\left|\frac{\partial(y^2)}{\partial y}\right| \tag{A.34}$$

$$= \frac{b^a (y^2)^{a-1} e^{-by^2}}{\mathbf{\Gamma}(a)}|2y|, \tag{A.35}$$

$$= \frac{2b^a y^{2a-1} e^{-by^2}}{\mathbf{\Gamma}(a)}, \tag{A.36}$$

The log probability, which is optimised, is defined by

$$\log \mathrm{Ga}^{\frac{1}{2}}\left(\, y|a, b\,\right) = (2a-1)\log(y) - by^2 + a\log(b) - \log\left(\boldsymbol{\Gamma}(a)\right) + \log(2). \tag{A.37}$$

A.1.8 Scaled Gaussian and square root Gamma variables

For both derivations a change of variables $q(y) = \alpha x = y$ and $q^{-1}(x) = x = \frac{1}{\alpha}y$ is required. This change of variables has a log determinant of the Jacobian defined as

$$\left|\frac{\partial q(y)}{\partial x}\right| = \left|\frac{\partial \alpha x}{\partial x}\right| = |\alpha| = \alpha. \tag{A.38}$$

For the normal distribution we want to prove that

$$\mathcal{N}\left(y|\alpha\mu, \alpha^2\sigma^2\right) = \mathcal{N}\left(\, x|\mu, \sigma^2\,\right) \quad s.t. \quad y = \alpha x. \tag{A.39}$$

Intuitively, this means that if x is normally distributed, $y = \alpha x$ is also normally distributed, with the α folded into the hyper-parameters.

$$\begin{aligned}
\mathcal{N}\left(y|\alpha\mu, \alpha^2\sigma^2\right) &= \left|\frac{\partial \alpha x}{\partial x}\right| \mathcal{N}\left(\, \alpha x|\alpha\mu, \alpha^2\sigma^2\,\right), && \text{(A.40)}\\
&= \alpha\frac{1}{\sqrt{2\pi\alpha^2\sigma^2}} e^{-\frac{(\alpha x - \alpha\mu)^2}{2\alpha^2\sigma^2}}, && \text{(A.41)}\\
&= \frac{\alpha}{\alpha\sqrt{2\pi\sigma^2}} e^{-\frac{\alpha^2(x-\mu)^2}{2\alpha^2\sigma^2}}, && \text{(A.42)}\\
&= \frac{\cancel{\alpha}}{\cancel{\alpha}\sqrt{2\pi\sigma^2}} e^{-\frac{\cancel{\alpha^2}(x-\mu)^2}{\cancel{\alpha^2}2\sigma^2}}, && \text{(A.43)}\\
\mathcal{N}\left(\, x|\mu, \sigma^2\,\right) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, && \text{(A.44)}
\end{aligned}$$

where in the final line we recognize the probability density function of a Gaussian distribution.

For the square root Gamma distribution we want to prove that

$$\mathrm{Ga}^{\frac{1}{2}}\left(\, y|a, \frac{b}{\alpha^2}\,\right) = \mathrm{Ga}^{\frac{1}{2}}\left(\, x|a, b\,\right) \quad s.t. \quad y = \alpha x. \tag{A.45}$$

Intuitively, this means that if x is square root Gamma distributed, $y = \alpha x$ is also square root Gamma distributed, with the α folded into the hyper-parameters.

$$\mathrm{Ga}^{\frac{1}{2}}\left(y|a, \frac{b}{\alpha^2}\right) = \left|\frac{\partial \alpha x}{\partial x}\right| \mathrm{Ga}^{\frac{1}{2}}\left(\alpha x|a, \frac{b}{\alpha^2}\right) \tag{A.46}$$

$$= \alpha \frac{2\left(\frac{b}{\alpha^2}\right)^a (\alpha x)^{2a-1} e^{-\frac{b}{\alpha^2}(\alpha x)^2}}{\mathbf{\Gamma}(a)}, \tag{A.47}$$

$$= \frac{\alpha 2 \alpha^{-2a} b^a \alpha^{2a-1} x^{2a-1} e^{-\frac{\alpha^2}{\alpha^2} b x^2}}{\mathbf{\Gamma}(a)}, \tag{A.48}$$

$$= \frac{2 \alpha^{-2a+1} b^a \alpha^{2a-1} x^{2a-1} e^{-\frac{\alpha^2}{\alpha^2} b x^2}}{\mathbf{\Gamma}(a)}, \tag{A.49}$$

$$= \frac{\bcancel{\alpha^{-(2a-1)}}\, \bcancel{\alpha^{2a-1}}\, 2 b^a x^{2a-1} e^{-\frac{\cancel{\alpha^2}}{\cancel{\alpha^2}} b x^2}}{\mathbf{\Gamma}(a)}, \tag{A.50}$$

$$\mathrm{Ga}^{\frac{1}{2}}\left(x|a, b\right) = \frac{2 b^a x^{2a-1} e^{-bx^2}}{\mathbf{\Gamma}(a)}, \tag{A.51}$$

where in the final line we recognize the probability density function of a square root Gamma distribution.

A.1.9 Cholesky-Wishart distribution

Derivation

This derivation is partly based on the matrix outer product bijector derivation of Dillon *et al*. [Dillon et al., 2017]. The starting point will be the definition of the Wishart density function in equation 3.36 and the change of variables transformation in equation 3.41.

First, we prove the change of variables result in equation 3.42. We substitute $\mathbf{\Lambda}$ for $\mathbf{L}\mathbf{L}^{\mathsf{T}}$ in equation 3.36, which results in

$$W(\mathbf{L}\mathbf{L}^{\mathsf{T}}|\mathbf{V}, p) = \frac{1}{2^{pn/2}|\mathbf{V}|^{p/2}\Gamma_n\left(\frac{p}{2}\right)} |\mathbf{L}\mathbf{L}^{\mathsf{T}}|^{(p-n-1)/2} e^{-(1/2)\mathrm{tr}(\mathbf{V}^{-1}\mathbf{L}\mathbf{L}^{\mathsf{T}})}, \tag{A.52}$$

$$= \frac{1}{2^{pn/2}|\mathbf{V}|^{p/2}\Gamma_n\left(\frac{p}{2}\right)} (|\mathbf{L}||\mathbf{L}^{\mathsf{T}}|)^{(p-n-1)/2} e^{-(1/2)\mathrm{tr}(\mathbf{V}^{-1}\mathbf{L}\mathbf{L}^{\mathsf{T}})}, \tag{A.53}$$

$$= \frac{1}{2^{pn/2}|\mathbf{V}|^{p/2}\Gamma_n\left(\frac{p}{2}\right)} (|\mathbf{L}|^2)^{(p-n-1)/2} e^{-(1/2)\mathrm{tr}(\mathbf{V}^{-1}\mathbf{L}\mathbf{L}^{\mathsf{T}})}, \tag{A.54}$$

$$= \frac{1}{2^{pn/2}|\mathbf{V}|^{p/2}\Gamma_n\left(\frac{p}{2}\right)} |\mathbf{L}|^{(p-n-1)} e^{-(1/2)\mathrm{tr}(\mathbf{V}^{-1}\mathbf{L}\mathbf{L}^{\mathsf{T}})}. \tag{A.55}$$

Next we turn our attention to the determinant of the Jacobian term in equation 3.41, namely

$$\left|\frac{\partial(\mathbf{L}\mathbf{L}^{\intercal})}{\partial\mathbf{L}}\right| = \left|\frac{\partial\mathbf{\Lambda}}{\partial\mathbf{L}}\right|, \tag{A.56}$$

where we will prove that

$$\left|\frac{\partial\mathbf{\Lambda}}{\partial\mathbf{L}}\right| = 2^n \prod_{j=0}^{n-1} \ell_{j,j}^{n-j}. \tag{A.57}$$

The derivative of a single element in $\mathbf{L}$ indexed by a, b, with respect to a single element in $\mathbf{\Lambda}$ indexed by i, j can be written as

$$\begin{aligned}\frac{\partial\lambda i, j}{\partial\ell_{a,b}} &= \frac{\partial(\boldsymbol{\ell}_{i,:}\boldsymbol{\ell}_{:,j}^{\intercal})}{\partial\ell_{a,b}}, &\text{(A.58)}\\ &= \sum_{d=0}^{n-1}(1_{i=a}1_{d=b}\ell_{j,d} + 1_{j=a}1_{d=b}\ell_{i,d}), &\text{(A.59)}\end{aligned}$$

where : is used to denote selecting all the elements along the given dimension, and $1_{i=j}$ is a scalar indicator variable with a value of 1 if i is equal to j and 0 otherwise.

In order to build the Jacobian matrix, both $\mathbf{L}$ and $\mathbf{\Lambda}$ must be vectorised, otherwise we would have a multidimensional Jacobian hyper-matrix for which computing the hyper-determinant is a complex task. A row-major mapping operation which drops all upper-triangular elements is used, which will be denoted as vec[·], where the indices are defined as

$$\begin{aligned}&k = i(i+1)/2 + j && i >= j &\text{(A.60)}\\ &drop && i < j, &\text{(A.61)}\end{aligned}$$

where i and j index rows and columns in the input matrix, k is the index in the output vector, and *drop* is used to denote elements that are dropped. For example a 4×4 matrix would be vectorised as

$$\begin{bmatrix} 0 & . & . & . \\ 1 & 2 & . & . \\ 3 & 4 & 5 & . \\ 6 & 7 & 8 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}. \tag{A.62}$$

The dropped elements in $\mathbf{L}$ are zero by construction and so is their gradient. As $\mathbf{\Lambda}$ is symmetric, the dropped elements correspond to the repeated values in the matrix.

Observe that $k(i,j) < k(a,b)$ iff (1) $i < a$ or (2) $i = a$ and $j < b$. In both cases

$$\frac{\partial \mathrm{vec}[\lambda_{i,j}]}{\partial \mathrm{vec}[\ell_{a,b}]} = 0 \tag{A.63}$$

since:

$$\begin{aligned} j \leq i < a \qquad & \text{thus} \qquad & i, j \neq a, \qquad & \text{(A.64)} \\ i = a > j \qquad & \text{thus} \qquad & i, j \neq a. \qquad & \text{(A.65)} \end{aligned}$$

As a result, the Jacobian matrix is lower-triangular, which simplifies the determinant to be the product of its diagonal elements. As we are only interested in the diagonal elements in the Jacobian, equation A.59 for these elements simplifies to

$$\frac{\partial \mathrm{vec}[\lambda_{i,j}]}{\partial \mathrm{vec}[\ell_{i,j}]} = \ell_{j,j} + 1_{i=j}\ell_{i,j}. \tag{A.66}$$

From the previous definition, observe that for a new row $(i+1,:)$ in $\mathbf{L}$, the first j gradient terms are $\{\ell_{0,0}, \ell_{1,1}, \cdots, \ell_{j,j}\}$, and the diagonal term is

$$\frac{\partial \mathrm{vec}[\lambda_{j+1,j+1}]}{\partial \mathrm{vec}[\ell_{j+1,j+1}]} = 2\ell_{j+1,\,j+1}. \tag{A.67}$$

Therefore, we complete the proof by aggregating all the terms

$$\left|\frac{\partial \mathbf{\Lambda}}{\partial \mathbf{L}}\right| = \left|\frac{\partial \mathrm{vec}[\mathbf{\Lambda}]}{\partial \mathrm{vec}[\mathbf{L}]}\right| = 2^n \prod_{j=0}^{n-1} \ell_{j,j}^{n-j}. \tag{A.68}$$

Efficient likelihood evaluation

In this section we show how to evaluate equations 3.42 and 3.43, in terms of the dense representation $\mathbf{T} = \mathbf{BW}$. As it is common practice in machine learning, the log probability is optimised instead of directly maximising the distribution probability density function:

$$\begin{aligned} \log W^c(\mathbf{L}|\mathbf{V},p) = -\frac{p}{2}\log\left(|\mathbf{V}|\right) - \log\left(\Gamma_n\left(\frac{p}{2}\right)\right) + (p-n-1)\log(|\mathbf{L}|) - \\ \frac{1}{2}\mathrm{tr}\left(\mathbf{V}^{-1}\mathbf{L}\mathbf{L}^{\intercal}\right) - \frac{n(2-p)}{2}\log(2) + \sum_{i=0}^{n-1}(n-i)\log(\ell_{i,i}) \end{aligned} \tag{A.69}$$

All the terms that do not involve $\mathbf{L}$ are ignored, as they are constants that can be precomputed.

The log determinant and the log determinant of the Jacobian can be computed as

$$(p-n-1)\log(|\mathbf{L}|) = (p-n-1)\sum_{j=0}^{n-1}\log(t_{0,j}), \tag{A.70}$$

$$\sum_{i=0}^{n-1}(n-i)\log(\ell_{i,i}) = \sum_{j=0}^{n-1}(n-j)\log(t_{0,j}). \tag{A.71}$$

The trace term is more problematic. Luckily, for a diagonal scale matrix, $\mathbf{V}$, it simplifies to

$$\operatorname{tr}\left(\mathbf{V}^{-1}\mathbf{L}\mathbf{L}^{\intercal}\right) = \operatorname{tr}\left(\mathbf{V}^{-1}\mathbf{\Lambda}\right) = \sum_{i=0}^{n-1}\frac{\lambda_{i,i}}{v_{i,i}}. \tag{A.72}$$

Thus, it can be efficiently computed as well, as the i^{th} element in the diagonal of $\mathbf{\Lambda}$ is the sum of the squared values over a single row:

$$\lambda_{i,i} = \sum_{j=0}^{n-1}\ell_{i,j}^2. \tag{A.73}$$

From our dense representation, $f(\mathbf{T})$, a convolution operator can be used to sum up the corresponding values as

$$\lambda_{i,i} = (f^{-1}(f(\mathbf{T}^2) * \mathbf{o}))_{i,i}, \tag{A.74}$$

where $\mathbf{T}^2$ applies an element-wise square operator and $f(\cdot)$ is the reshape operator defined in Section 3.3.4. Note that directly summing over the rows of $\mathbf{T}^2$ is not possible due to the previously defined operator $\mathbf{s}(\cdot)$, which shifts and pads with zeros the elements in $\mathbf{T}$. The filters for the convolution, which sum over the shifted values have $n_{\mathrm{c}} = (n_{\mathrm{f}}^2 - 1)/2 + 1$ channels, and all the elements in each channel are zeros except at a single location. For example, for a 3×3 sparsity pattern on $\mathbf{L}$, the kernel, $\mathbf{o}$, is defined as

$$\mathbf{o} = g(\mathbf{I}) = \left[\begin{array}{ccccc} \begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} & \begin{bmatrix}0&0&0\\0&0&1\\0&0&0\end{bmatrix} & \begin{bmatrix}0&0&0\\0&0&0\\1&0&0\end{bmatrix} & \begin{bmatrix}0&0&0\\0&0&0\\0&1&0\end{bmatrix} & \begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix} \\ k=0 & k=1 & k=2 & k=3 & k=4 \end{array}\right], \tag{A.75}$$

where k indicates the channel dimension in the kernel, and $g(\cdot)$ is the reshape and zero pad operator defined in Section 3.3.4.

A.1.10 Mode of the sparse Cholesky-Wishart distribution

In this section we will prove that the mode of a sparse Cholesky-Wishart distribution, $W^c_{\text{sp}}(\mathbf{L} \,|\, \mathbf{V}, p)$, whose probability density function was defined in equation 3.50, is a diagonal matrix, $\mathbf{\Lambda}_{\text{mode}}$, defined as

$$\hat{\lambda}_{i,i} = v_{i,i} \left(\frac{i(1-n)}{n-1} + p - 1 \right), \tag{A.76}$$

$$\hat{\lambda}_{i,j} = 0, \quad \forall i, j \quad \text{s.t.} \quad i \neq j \quad \text{and} \quad j < i, \tag{A.77}$$

where $\hat{\lambda}_{i,i}$ are the diagonal elements in $\mathbf{\Lambda}_{\text{mode}}$, $\hat{\lambda}_{i,j}$ are the off-diagonal ones and $\mathbf{L} \in \mathbb{R}^{n \times n}$.

Mode of the square root Gamma distribution

We start the proof by finding the the mode of a square root Gamma distribution. We need to take the derivative of the probability density function with respect to x and equate to zero.

$$\frac{\partial \text{Ga}^{\frac{1}{2}}\left(x|a, b \right)}{\partial x} = \frac{2b}{\Gamma(a)}(2a-1)x^{2a-2}e^{-bx^2} - \frac{2b}{\Gamma(a)}x^{2a-1}2bxe^{-bx^2}, \tag{A.78}$$

$$\Rightarrow \cancel{\frac{2b}{\Gamma(a)}}(2a-1)x^{2a-2}e^{-bx^2} - \cancel{\frac{2b}{\Gamma(a)}}x^{2a-1}2bxe^{-bx^2} = 0, \tag{A.79}$$

$$\Rightarrow (2a-1)x^{2a-2}e^{-bx^2} - x^{2a}2be^{-bx^2} = 0, \tag{A.80}$$

$$\Rightarrow e^{-bx^2}((2a-1)x^{2a-2} - 2bx^{2a}) = 0, \tag{A.81}$$

$$\Rightarrow e^{-bx^2}x^{2a-2}(2a-1-2bx^2) = 0, \tag{A.82}$$

where the cancellation of the $\Gamma(\cdot)$ terms is possible as $b > 0$ and $a > 0$, so $\frac{2b}{\Gamma(a)} > 0$.

Now we find the roots independently. The first root is

$$\begin{aligned} e^{-bx^2} &= 0, && \text{(A.83)} \\ -bx^2 &= \log(0), && \text{(A.84)} \\ -bx^2 &= -\infty, && \text{(A.85)} \\ x &= \pm\infty && \text{(A.86)} \end{aligned}$$

which is not a valid solution.

The second root is

$$\begin{aligned} x^{2a-2} &= 0, && \text{(A.87)} \\ x &= 0, && \text{(A.88)} \end{aligned}$$

which is also not a valid solution as $x > 0$.

The third root is

$$\begin{aligned} 2a - 1 - 2bx^2 &= 0, && \text{(A.89)} \\ x &= \sqrt{\frac{2a-1}{2b}}, && \text{(A.90)} \\ &= \sqrt{\frac{1}{2}}\sqrt{\frac{2a-1}{b}}, && \text{(A.91)} \\ &= \frac{\sqrt{2}}{\sqrt{2}}\frac{1}{\sqrt{2}}\sqrt{\frac{2a-1}{b}}, && \text{(A.92)} \\ &= \frac{\sqrt{2}}{2}\sqrt{\frac{2a-1}{b}}, && \text{(A.93)} \end{aligned}$$

where x is the mode of the distribution.

Mode in terms of $\mathbf{L}$, $\boldsymbol{\Lambda}$ and $\boldsymbol{\Sigma}$

Employing the mode derived from the square root Gamma distribution (eq. A.93), we now plug-in the hyper-parameters from the Cholesky-Wishart distribution (eq. 3.48), to find the Cholesky matrix, $\mathbf{L}_{\text{mode}}$, that corresponds to the mode of the distribution.

Formally, this is defined as

$$\hat{\ell}_{i,i} = \frac{\sqrt{2}}{2}\sqrt{\frac{2(0.5(\frac{i(1-n)}{n-1}+p))-1}{\frac{0.5}{v_{i,i}}}}, \tag{A.94}$$

$$= \frac{\sqrt{2}}{2}\sqrt{\frac{\cancel{2}(\cancel{0.5}(\frac{i(1-n)}{n-1}+p))-1}{\frac{0.5}{v_{i,i}}}}, \tag{A.95}$$

$$= \frac{\sqrt{2}}{2}\sqrt{\frac{v_{i,i}(\frac{i(1-n)}{n-1}+p-1)}{0.5}}, \tag{A.96}$$

$$= \frac{\sqrt{2}}{2}\sqrt{2}\sqrt{v_{i,i}\left(\frac{i(1-n)}{n-1}+p-1\right)}, \tag{A.97}$$

$$= \sqrt{v_{i,i}\left(\frac{i(1-n)}{n-1}+p-1\right)}, \tag{A.98}$$

where $\hat{\ell}_{i,i}$ is used to denote the diagonal elements in $\mathbf{L}_{\text{mode}}$.

The off-diagonal elements follow a Gaussian distribution, where it is known that the mode of the distribution is equal to the mean hyper-parameter, which is zero in our case. Therefore, $\mathbf{L}_{\text{mode}}$ is a diagonal matrix.

Rather than the Cholesky of the precision matrix, we are interested in the covariance matrix, which is more interpretable. Hence, we derive the covariance matrix that corresponds to the mode of the distribution. As the mode of the distribution is a diagonal matrix, $\mathbf{L}_{\text{mode}}$, the precision matrix $\mathbf{\Lambda}_{\text{mode}} = \mathbf{L}_{\text{mode}}\mathbf{L}_{\text{mode}}^{\intercal}$ is also diagonal, and we can simply square the result above

$$\hat{\lambda}_{i,i} = v_{i,i}\left(\frac{i(1-n)}{n-1}+p-1\right), \tag{A.99}$$

where $\hat{\lambda}_{i,i}$ denotes a diagonal element in $\mathbf{\Lambda}_{\text{mode}}$.

Similarly, the covariance matrix, $\mathbf{\Sigma}_{\text{mode}} = \mathbf{\Lambda}_{\text{mode}}^{-1}$, is diagonal as well. Thus, an inverse per element is enough to invert the matrix

$$\hat{\sigma}_{i,i}^{2} = \frac{1}{\hat{\lambda}_{i,i}} = \frac{1}{v_{i,i}\left(\frac{i(1-n)}{n-1}+p-1\right)}, \tag{A.100}$$

$$= \frac{n-1}{v_{i,i}(i(1-n)+(n-1)(p-1))}, \tag{A.101}$$

where $\hat{\sigma}_{i,i}^{2}$ denotes a diagonal element in $\mathbf{\Sigma}_{\text{mode}}$.

A.2 Network architectures

A.2.1 IPE models

The exp block in all the architectures removes the log in the diagonal values of the Cholesky matrix that is being estimated. However, as previously explained the log values are directly used for the log likelihood and prior evaluations.

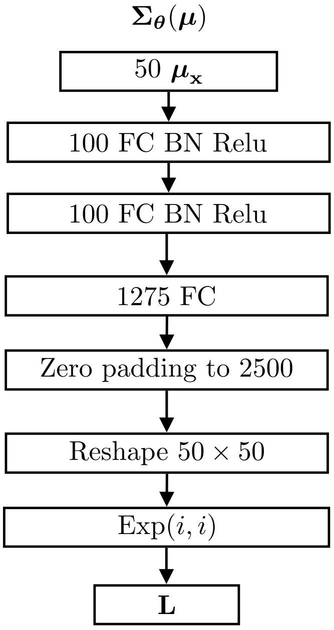


Figure A-1: **L**-Decoder network architecture for the splines dataset.

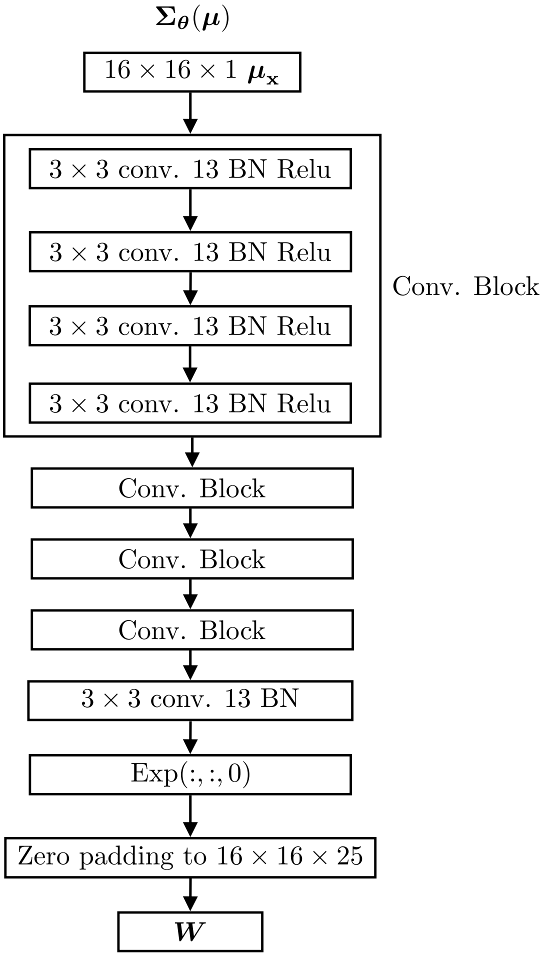


Figure A-2: **L**-Decoder network architecture for the ellipses dataset.

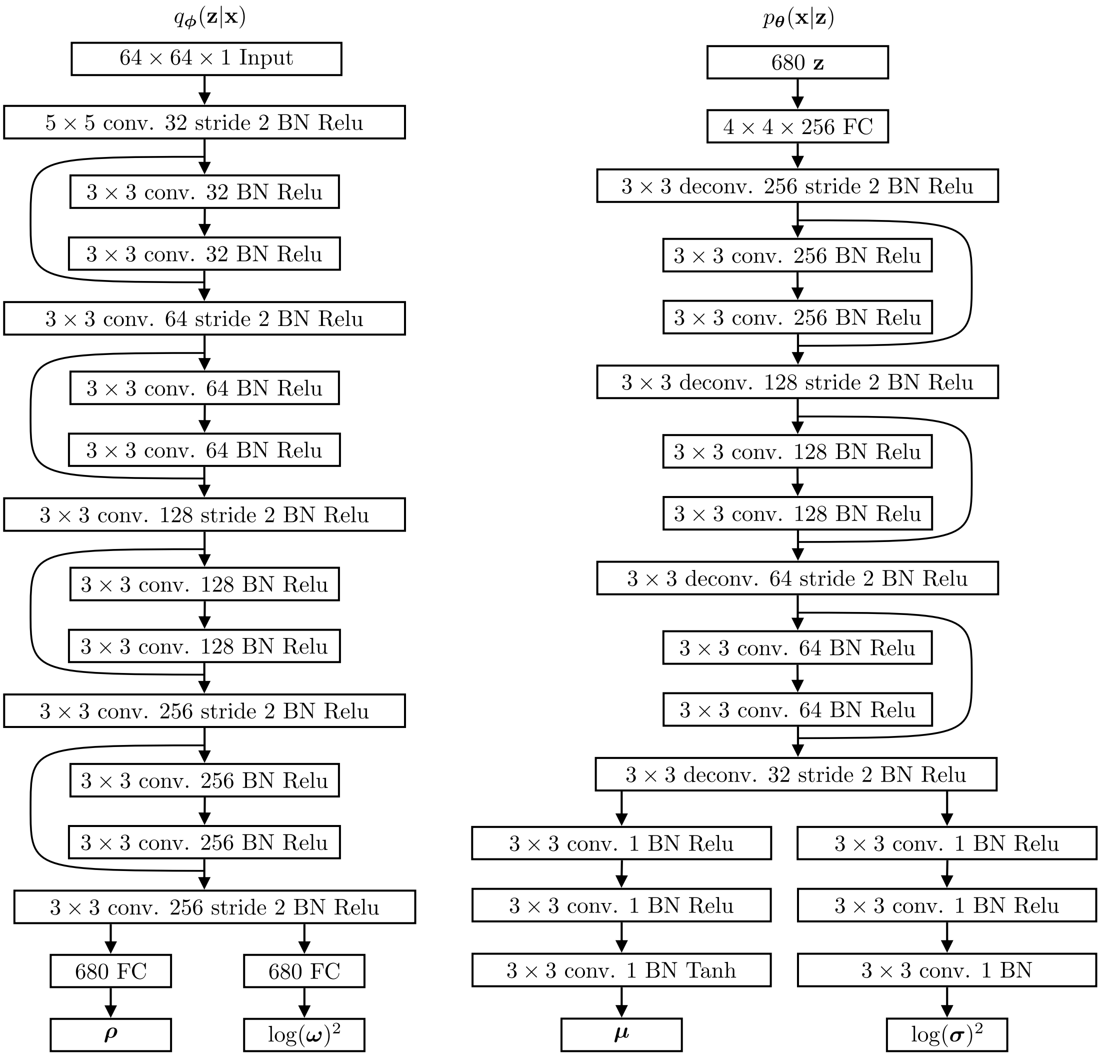

$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$
$64 \times 64 \times 1$ Input
5×5 conv. 32 stride 2 BN Relu
3×3 conv. 32 BN Relu
3×3 conv. 32 BN Relu
3×3 conv. 64 stride 2 BN Relu
3×3 conv. 64 BN Relu
3×3 conv. 64 BN Relu
3×3 conv. 128 stride 2 BN Relu
3×3 conv. 128 BN Relu
3×3 conv. 128 BN Relu
3×3 conv. 256 stride 2 BN Relu
3×3 conv. 256 BN Relu
3×3 conv. 256 BN Relu
3×3 conv. 256 stride 2 BN Relu
680 FC
680 FC
$\boldsymbol{\rho}$
$\log(\boldsymbol{\omega})^2$
$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$
680 $\mathbf{z}$
$4 \times 4 \times 256$ FC
3×3 deconv. 256 stride 2 BN Relu
3×3 conv. 256 BN Relu
3×3 conv. 256 BN Relu
3×3 deconv. 128 stride 2 BN Relu
3×3 conv. 128 BN Relu
3×3 conv. 128 BN Relu
3×3 deconv. 64 stride 2 BN Relu
3×3 conv. 64 BN Relu
3×3 conv. 64 BN Relu
3×3 deconv. 32 stride 2 BN Relu
3×3 conv. 1 BN Relu
3×3 conv. 1 BN Relu
3×3 conv. 1 BN Tanh
$\boldsymbol{\mu}$
3×3 conv. 1 BN Relu
3×3 conv. 1 BN Relu
3×3 conv. 1 BN
$\log(\boldsymbol{\sigma})^2$

Figure A-3: VAE architecture for the grey-scale CelebA dataset.

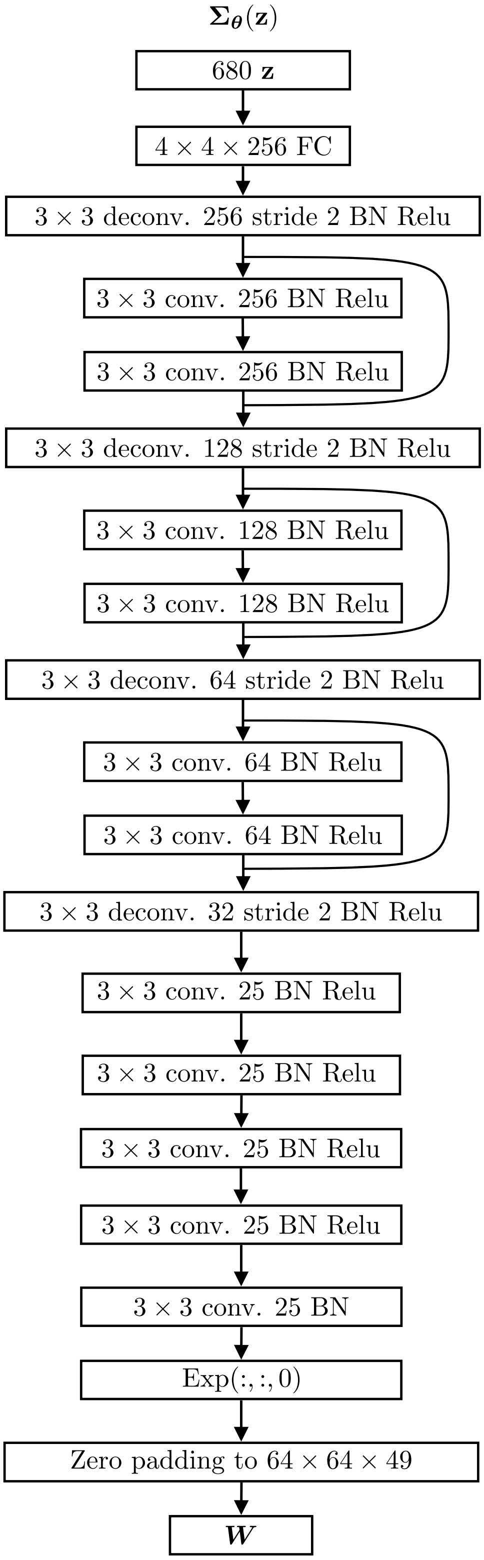

$\boldsymbol{\Sigma_\theta}(\mathbf{z})$
680 $\mathbf{z}$
$4 \times 4 \times 256$ FC
3×3 deconv. 256 stride 2 BN Relu
3×3 conv. 256 BN Relu
3×3 conv. 256 BN Relu
3×3 deconv. 128 stride 2 BN Relu
3×3 conv. 128 BN Relu
3×3 conv. 128 BN Relu
3×3 deconv. 64 stride 2 BN Relu
3×3 conv. 64 BN Relu
3×3 conv. 64 BN Relu
3×3 deconv. 32 stride 2 BN Relu
3×3 conv. 25 BN Relu
3×3 conv. 25 BN Relu
3×3 conv. 25 BN Relu
3×3 conv. 25 BN Relu
3×3 conv. 25 BN
Exp$(:,:,0)$
Zero padding to $64 \times 64 \times 49$
$\boldsymbol{W}$

Figure A-4: **L**-Decoder network architecture for the grey-scale CelebA dataset.

A.2.2 SDR models

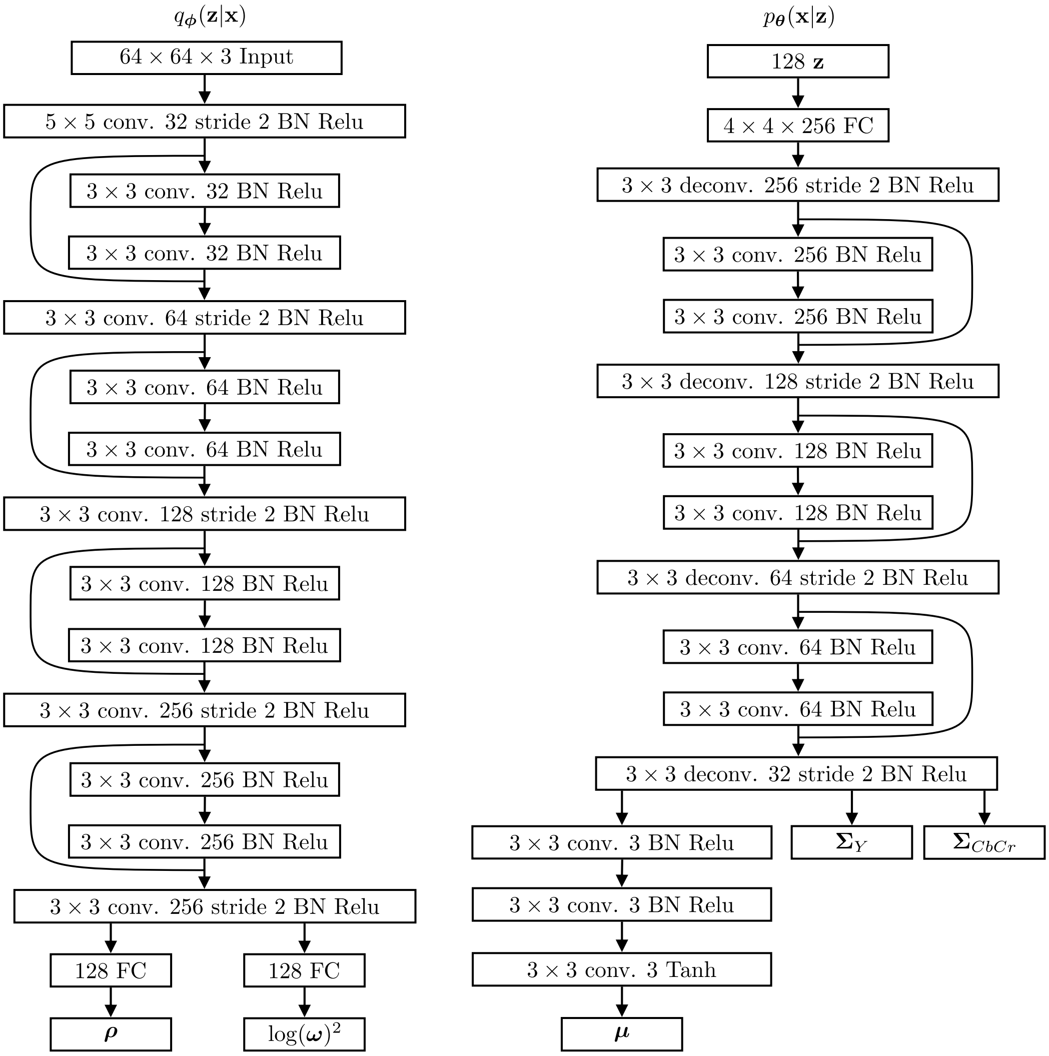

$q_{\phi}(\mathbf{z}|\mathbf{x})$
$64 \times 64 \times 3$ Input
5×5 conv. 32 stride 2 BN Relu
3×3 conv. 32 BN Relu
3×3 conv. 32 BN Relu
3×3 conv. 64 stride 2 BN Relu
3×3 conv. 64 BN Relu
3×3 conv. 64 BN Relu
3×3 conv. 128 stride 2 BN Relu
3×3 conv. 128 BN Relu
3×3 conv. 128 BN Relu
3×3 conv. 256 stride 2 BN Relu
3×3 conv. 256 BN Relu
3×3 conv. 256 BN Relu
3×3 conv. 256 stride 2 BN Relu
128 FC
128 FC
$\boldsymbol{\rho}$
$\log(\boldsymbol{\omega})^2$
$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$
128 $\mathbf{z}$
$4 \times 4 \times 256$ FC
3×3 deconv. 256 stride 2 BN Relu
3×3 conv. 256 BN Relu
3×3 conv. 256 BN Relu
3×3 deconv. 128 stride 2 BN Relu
3×3 conv. 128 BN Relu
3×3 conv. 128 BN Relu
3×3 deconv. 64 stride 2 BN Relu
3×3 conv. 64 BN Relu
3×3 conv. 64 BN Relu
3×3 deconv. 32 stride 2 BN Relu
3×3 conv. 3 BN Relu
$\boldsymbol{\Sigma}_Y$
$\boldsymbol{\Sigma}_{CbCr}$
3×3 conv. 3 BN Relu
3×3 conv. 3 Tanh
$\boldsymbol{\mu}$

Figure A-5: Common network architecture for all the models.

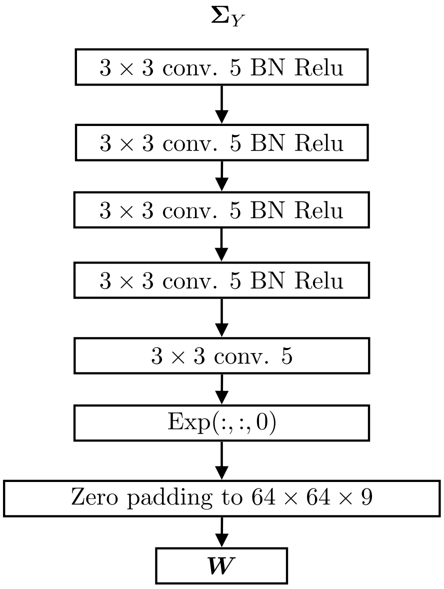

$\mathbf{\Sigma}_Y$
3×3 conv. 5 BN Relu
3×3 conv. 5 BN Relu
3×3 conv. 5 BN Relu
3×3 conv. 5 BN Relu
3×3 conv. 5
Exp(:, :, 0)
Zero padding to $64 \times 64 \times 9$
$\boldsymbol{W}$

Figure A-6: Covariance prediction branch for the Y channel in our model.

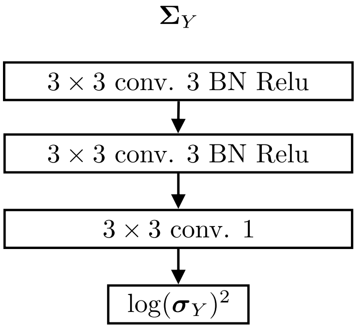

$\mathbf{\Sigma}_Y$
3×3 conv. 3 BN Relu
3×3 conv. 3 BN Relu
3×3 conv. 1
$\log(\boldsymbol{\sigma}_Y)^2$

Figure A-7: Diagonal covariance prediction branch for the Y channel in a VAE.

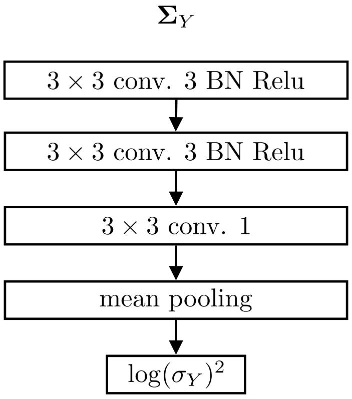


Figure A-8: Spherical covariance prediction branch for the Y channel in a VAE, where mean pooling does a mean across pixels, thus the output contains a single value.

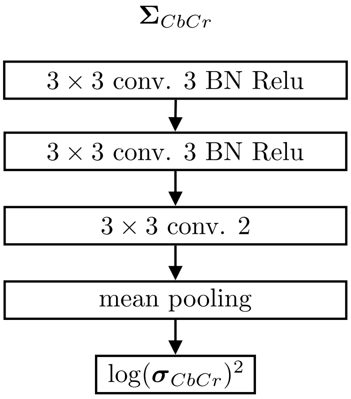

$\mathbf{\Sigma}_{CbCr}$
3×3 conv. 3 BN Relu
3×3 conv. 3 BN Relu
3×3 conv. 2
mean pooling
$\log(\boldsymbol{\sigma}_{CbCr})^2$

Figure A-9: Spherical covariance prediction branch for the Cb and Cr channels in a VAE, where mean pooling does a mean across pixels, thus the output contains two values.

A.3 Additional qualitative results

A.3.1 Ablation studies

Naïve training and priors: Reconstructions

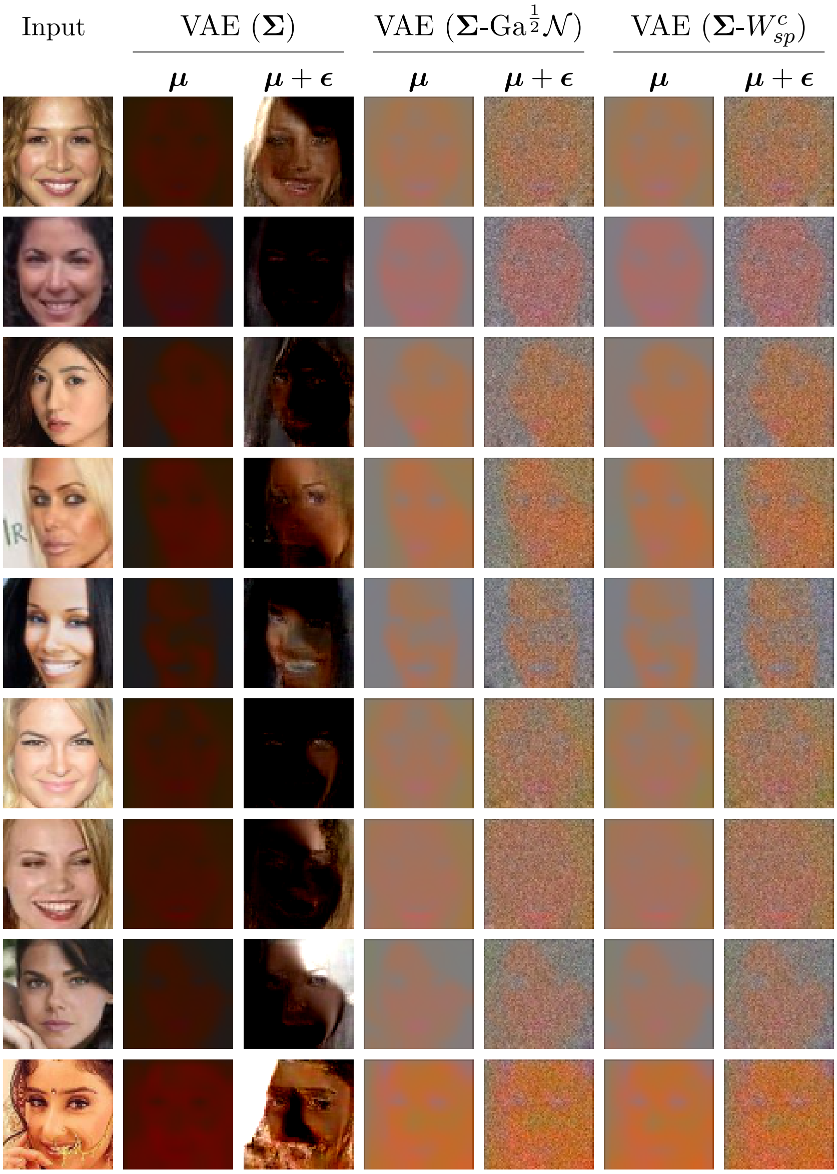

Input
VAE ($\boldsymbol{\Sigma}$)
VAE ($\boldsymbol{\Sigma}$-Ga$^{\frac{1}{2}}\mathcal{N}$)
VAE ($\boldsymbol{\Sigma}$-W^{c}_{sp})
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$

Figure A-10: Comparison of image reconstructions for the different models. A model without regularisation, VAE ($\boldsymbol{\Sigma}$), encodes much of the information in the input image in the structured residual. Employing regularisation with priors, VAE ($\boldsymbol{\Sigma}$-Ga$^{\frac{1}{2}}\mathcal{N}$) and VAE ($\boldsymbol{\Sigma}$-$W^{c}_{\text{sp}}$), leads to unstructured noise and poor means.

Naïve training and priors: Samples

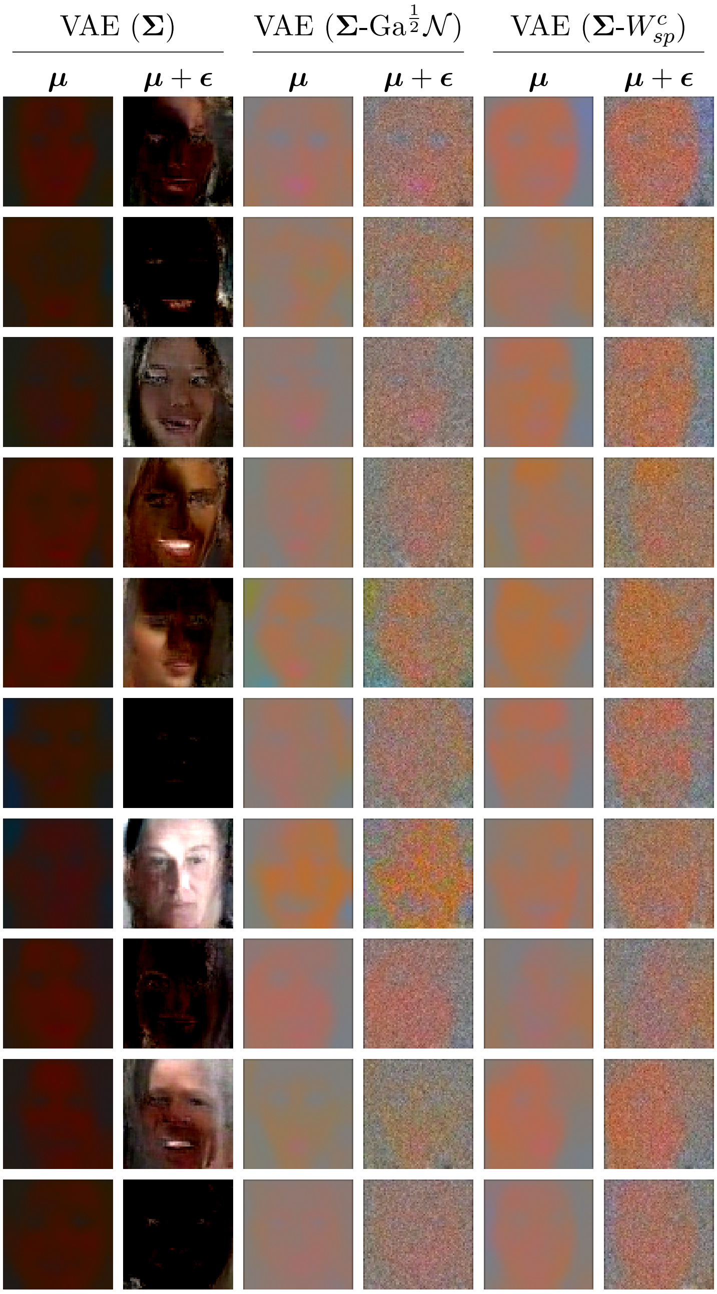

VAE ($\boldsymbol{\Sigma}$)
VAE ($\boldsymbol{\Sigma}$-Ga$^{\frac{1}{2}}\mathcal{N}$)
VAE ($\boldsymbol{\Sigma}$-W^{c}_{sp})
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$

Figure A-11: Comparison of samples generated by the different models. Not employing regularisation, VAE ($\boldsymbol{\Sigma}$), leads to modelling the image distribution in the structured residual. Employing regularisation with priors, VAE ($\boldsymbol{\Sigma}$-Ga$^{\frac{1}{2}}\mathcal{N}$) and VAE ($\boldsymbol{\Sigma}$-$W^{c}_{\mathrm{sp}}$), leads to unstructured noise and poor means.

A.3.2 Comparison to previous work

CelebA: Reconstructions

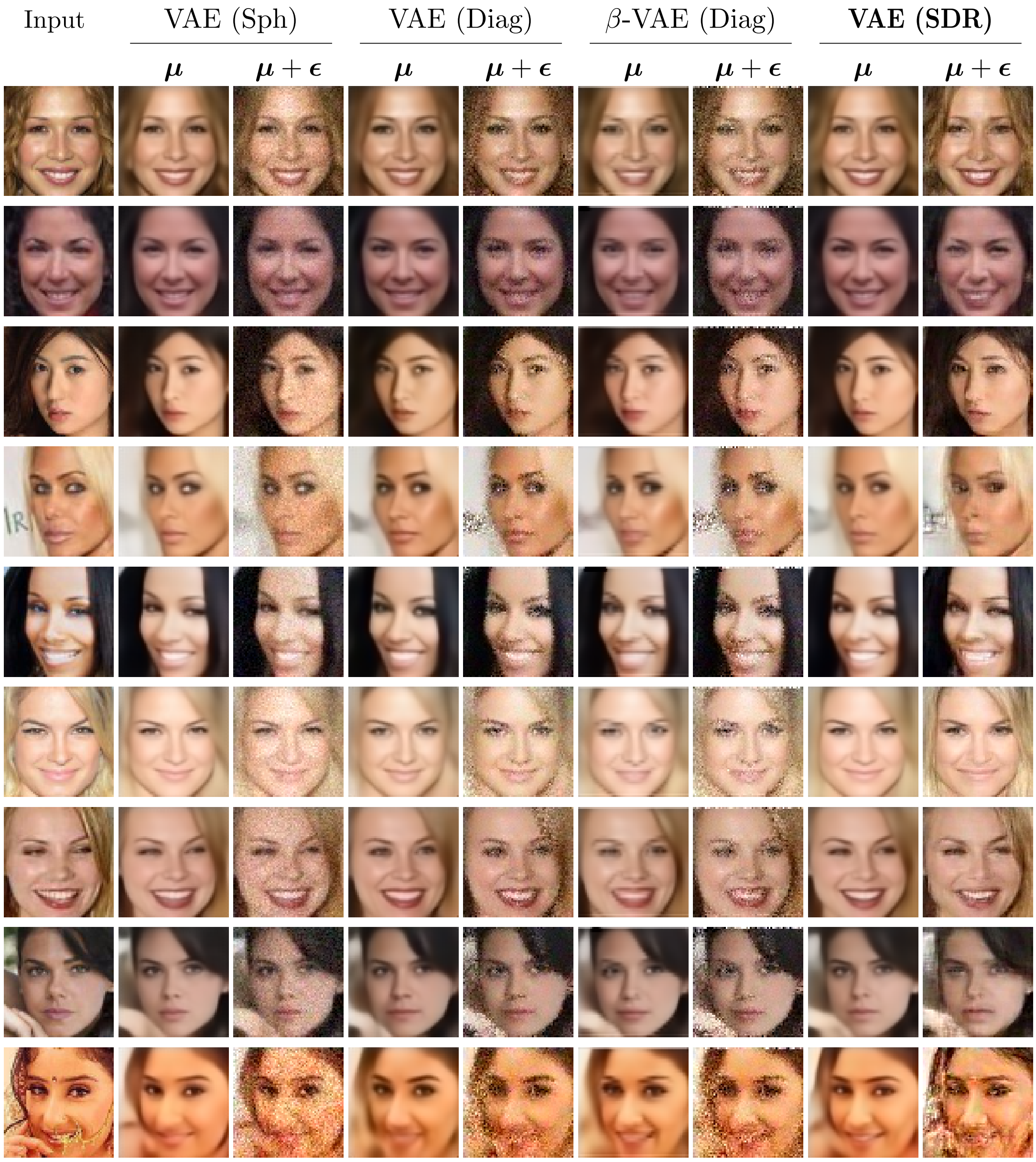

Input
VAE (Sph)
VAE (Diag)
β-VAE (Diag)
VAE (SDR)
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$

Figure A-12: Comparison of image reconstructions for the different models. In contrast to previous work, our model is able to learn structured residuals. This structured residuals add plausible high-frequency content to the means, $\boldsymbol{\mu}$, such as hair details in the first row, or tooth details in the second row.

CelebA: Samples

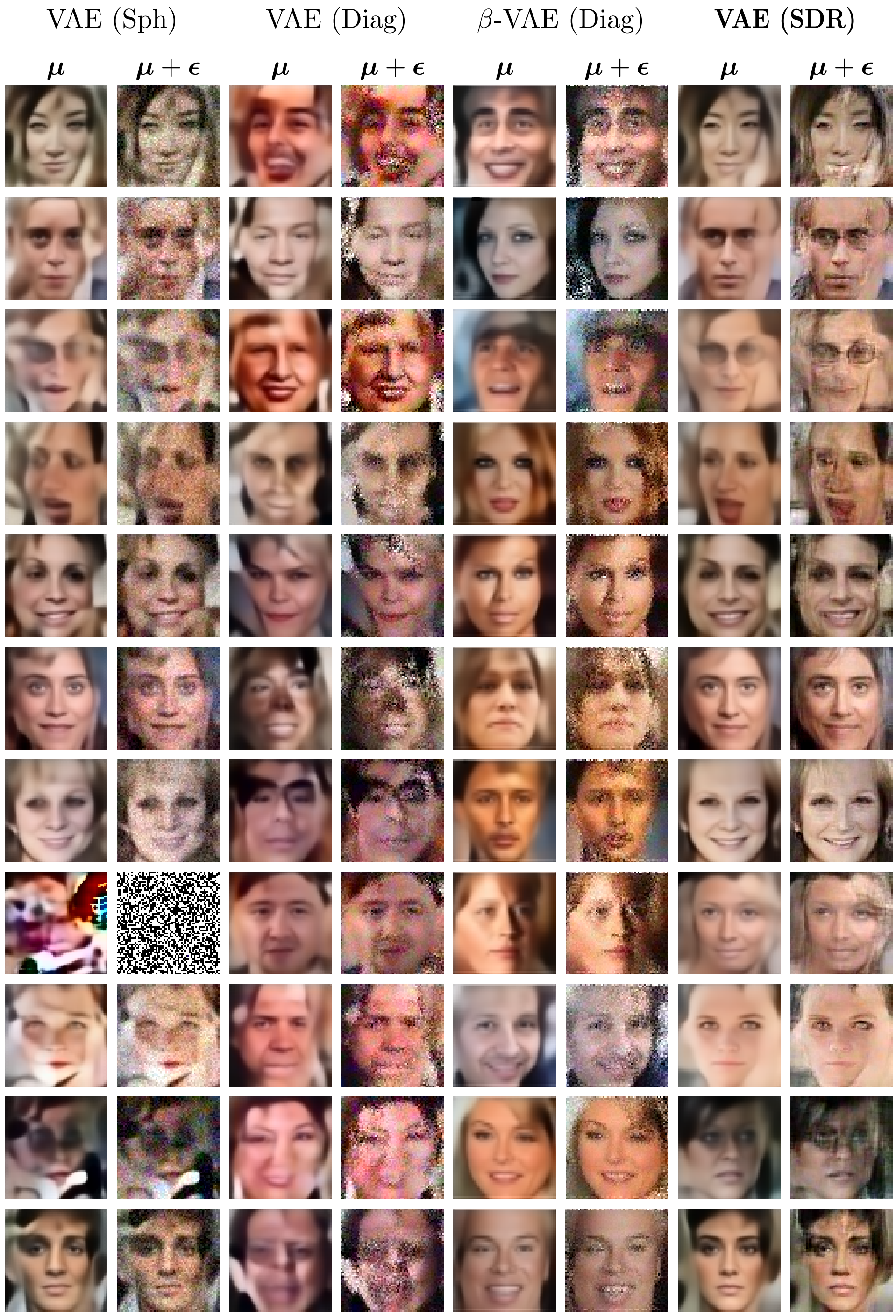

VAE (Sph)
VAE (Diag)
β-VAE (Diag)
VAE (SDR)
$\boldsymbol{\mu}$
$\boldsymbol{\mu} + \boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu} + \boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu} + \boldsymbol{\epsilon}$
$\boldsymbol{\mu}$
$\boldsymbol{\mu} + \boldsymbol{\epsilon}$

Figure A-13: Samples for all models, where our method is the only able to include high-frequency details such as wrinkles. Neither the means, $\boldsymbol{\mu}$, nor the residuals, $\boldsymbol{\epsilon}$, generated by VAE (Sph) and VAE (Diag) are realistic.

LSUN: Reconstructions

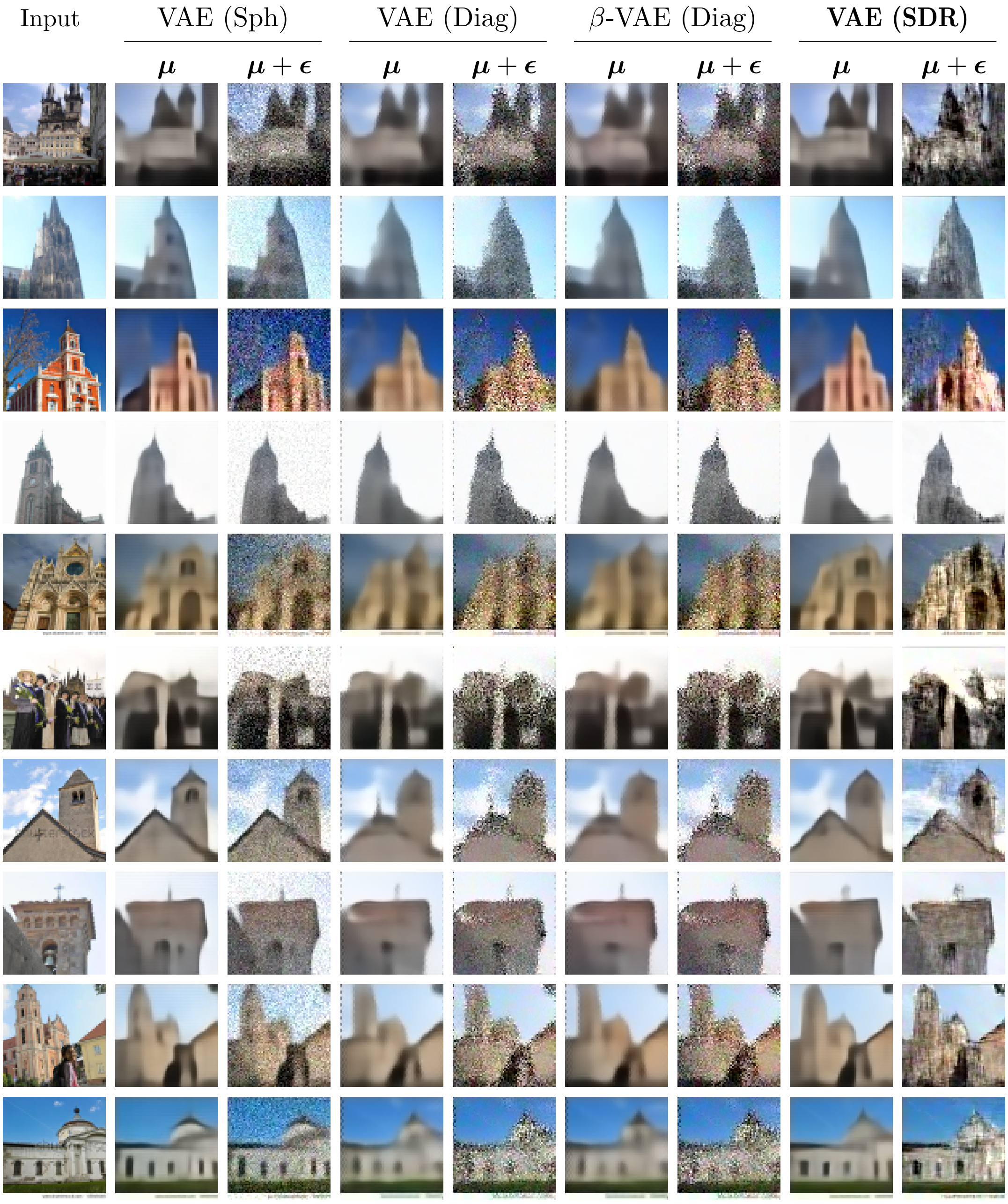


Figure A-14: Comparison of image reconstructions for the different models. Our model is able to improve over the noisy images generated by previous work. However, the increased complexity of this dataset has a detrimental effect on the quality of the images generated by all the models..

LSUN: Samples

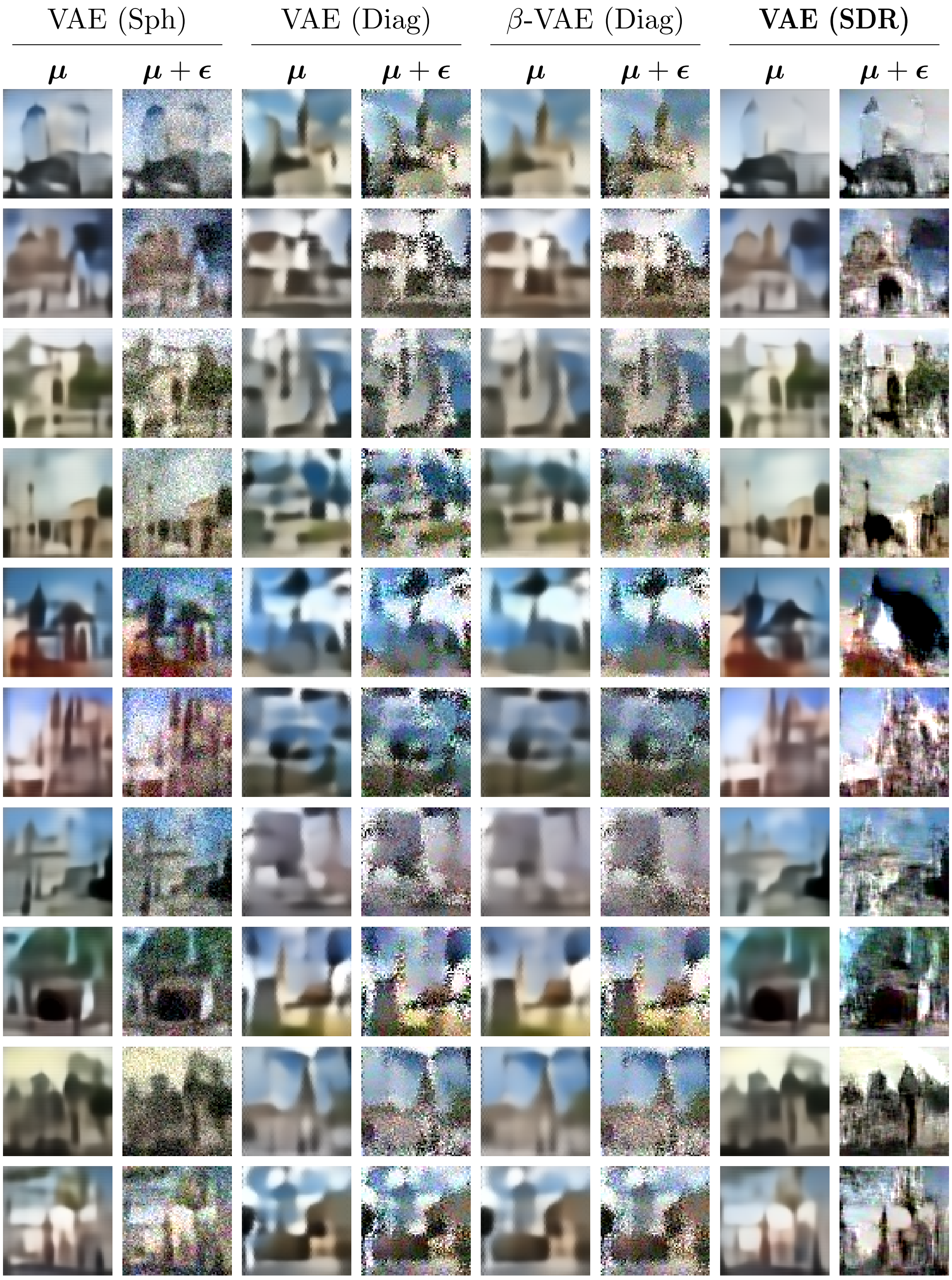


Figure A-15: Samples drawn from the models. VAE (Diag) fails to learn a useful latent space and β-VAE produces blurrier images than our model. The images produced by our approach contain structured high-frequency content, still, all the models are unable to generate realistic images.

Model insights: multiple residual samples

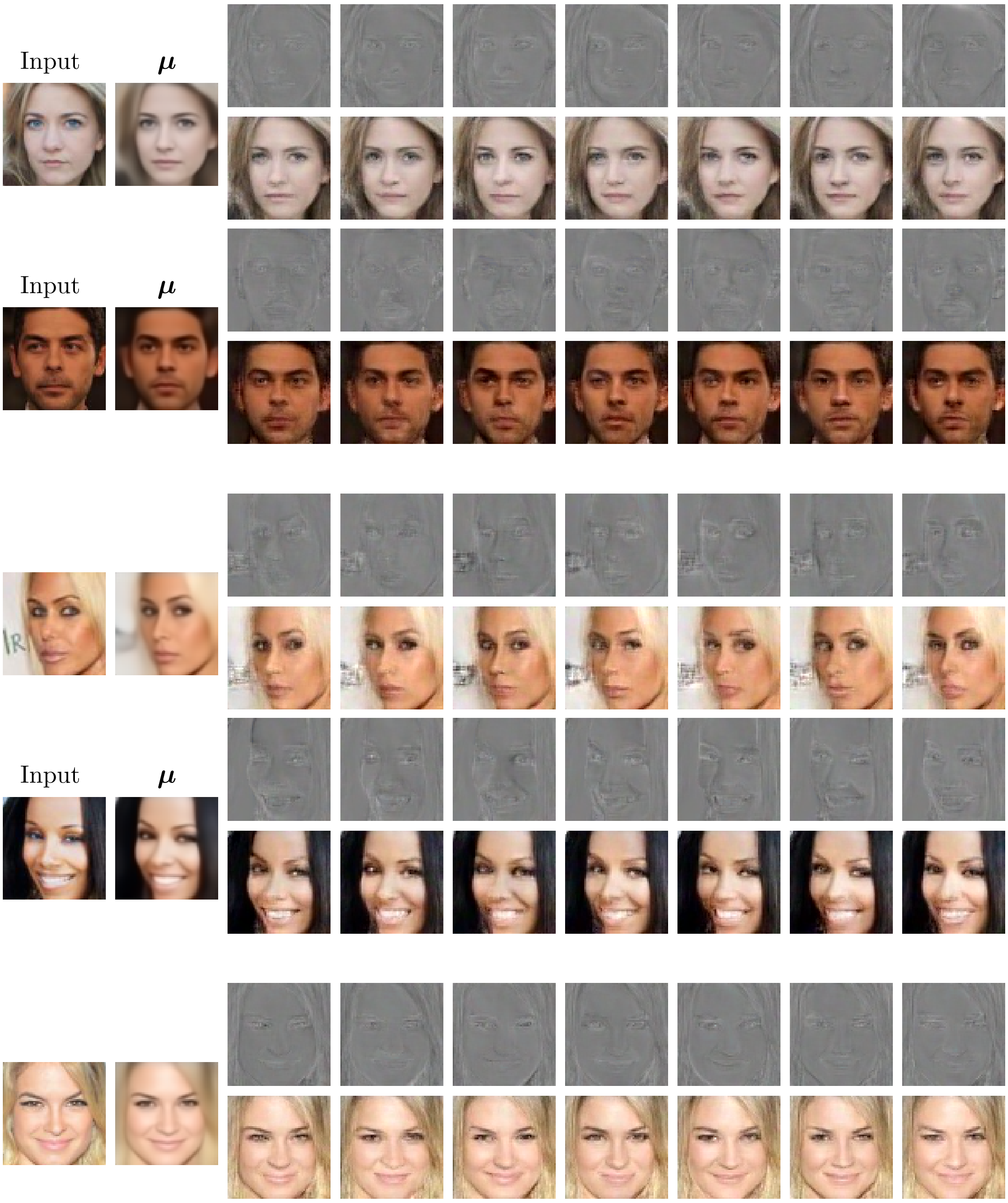


Figure A-16: Variability of the residual images for reconstructions using the VAE (SDR) model. In the first row, each column shows a residual sample from the same covariance matrix, and those are added to the mean, $\boldsymbol{\mu}$, in the second row. Variations of hair position or teeth shape can be observed. Additional results can be found in the supplemental video.

Model insights: interpolations

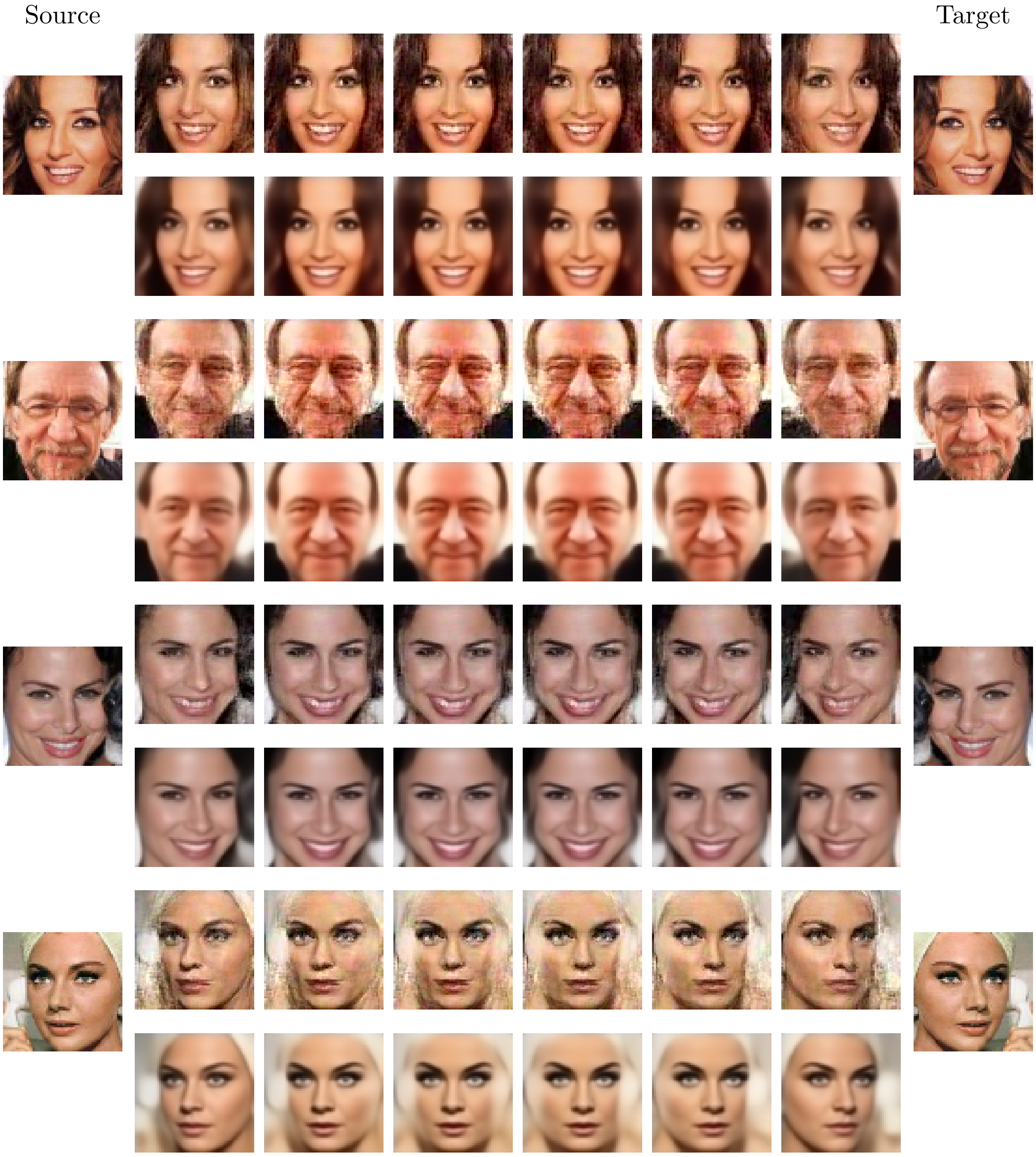


Figure A-17: Samples drawn with our model while interpolating on the latent space, from the source to target. The i value on each column corresponds to equation 3.54. For every pair of rows, the first contains the sample from the model, $\boldsymbol{\mu} + \boldsymbol{\epsilon}$, and the second contains the mean, $\boldsymbol{\mu}$.

Model insights: variance maps

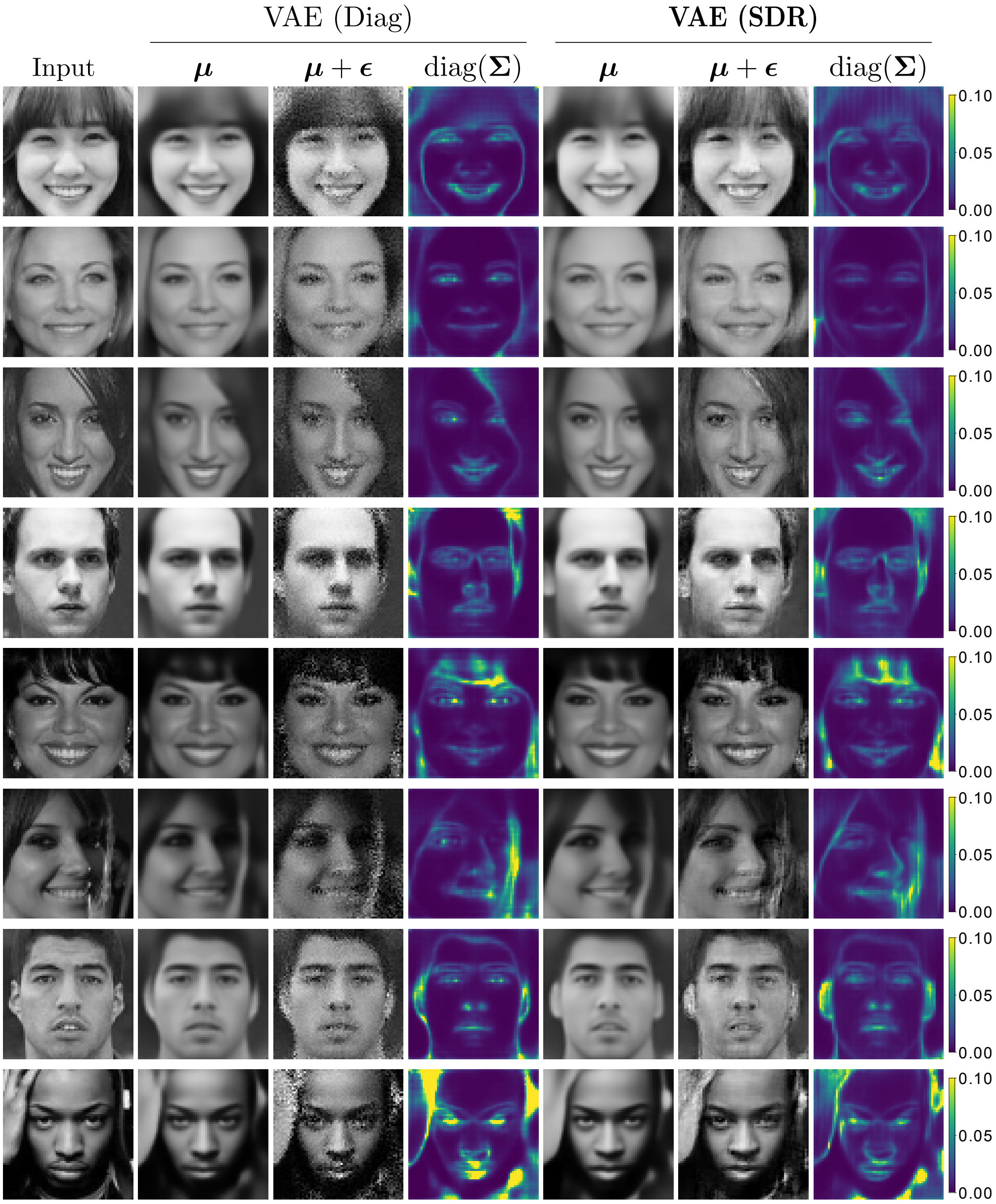

VAE (Diag)
VAE (SDR)
Input
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
diag($\boldsymbol{\Sigma}$)
$\boldsymbol{\mu}$
$\boldsymbol{\mu}+\boldsymbol{\epsilon}$
diag($\boldsymbol{\Sigma}$)
0.10
0.05
0.00

Figure A-18: Variance maps for the Y channel in a VAE with diagonal covariance Gaussian likelihood and in a VAE employing the SDR approach for a dense covariance matrix. The diagonal noise estimation model mistakenly identifies structured areas as high variance regions, whereas our covariance model properly identifies them as regions with high covariance, yet lower variance.

Restricted vs unrestricted features: reconstructions

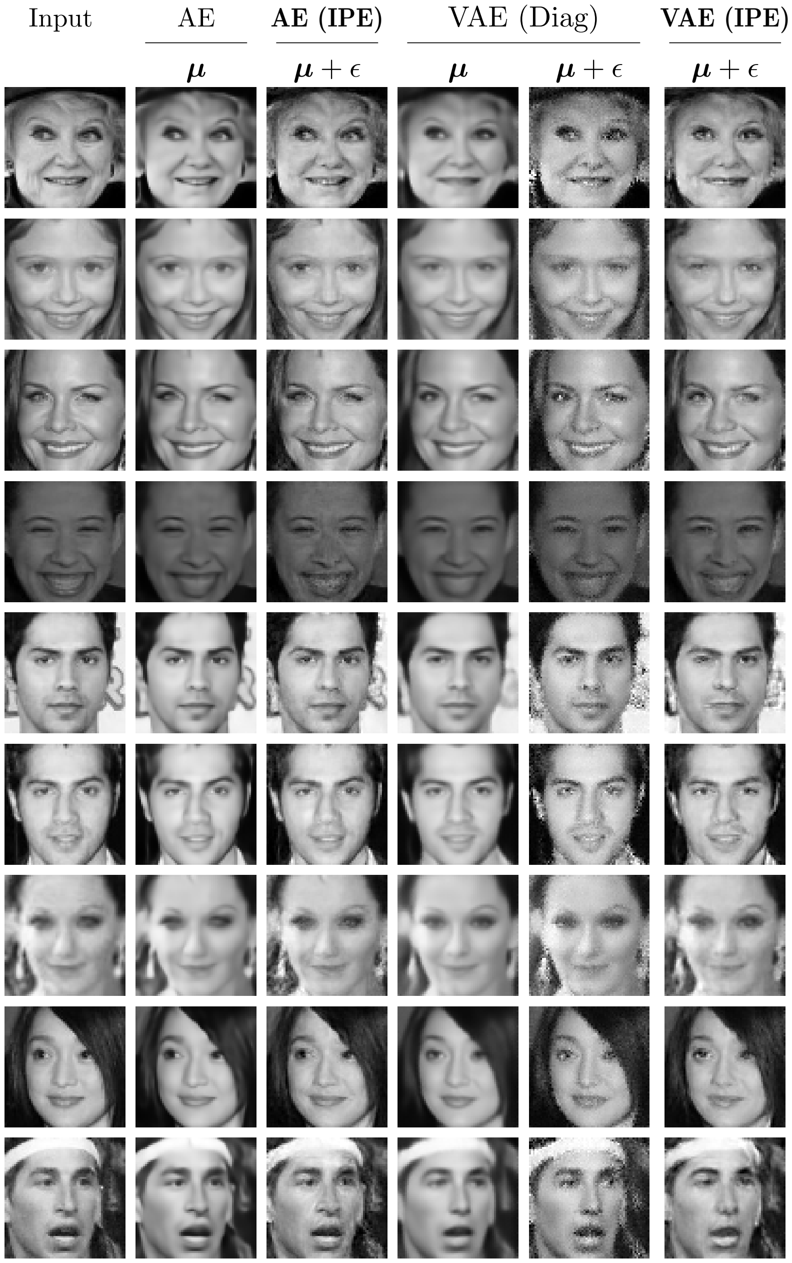

Input
AE
AE (IPE)
VAE (Diag)
VAE (IPE)
$\boldsymbol{\mu}$
$\boldsymbol{\mu} + \epsilon$
$\boldsymbol{\mu}$
$\boldsymbol{\mu} + \epsilon$
$\boldsymbol{\mu} + \epsilon$

Figure A-19: Comparison of image reconstructions for the different models. The AE and VAE both generate over-smoothed images in $\boldsymbol{\mu}$, and VAE (Diag) generates unrealistic unstructured noise in $\boldsymbol{\mu} + \boldsymbol{\epsilon}$. For both the AE and VAE, our IPE model adds plausible high-frequencies from a single sample drawn from the predicted uncertainty distribution.

A.3.3 Application: denoising

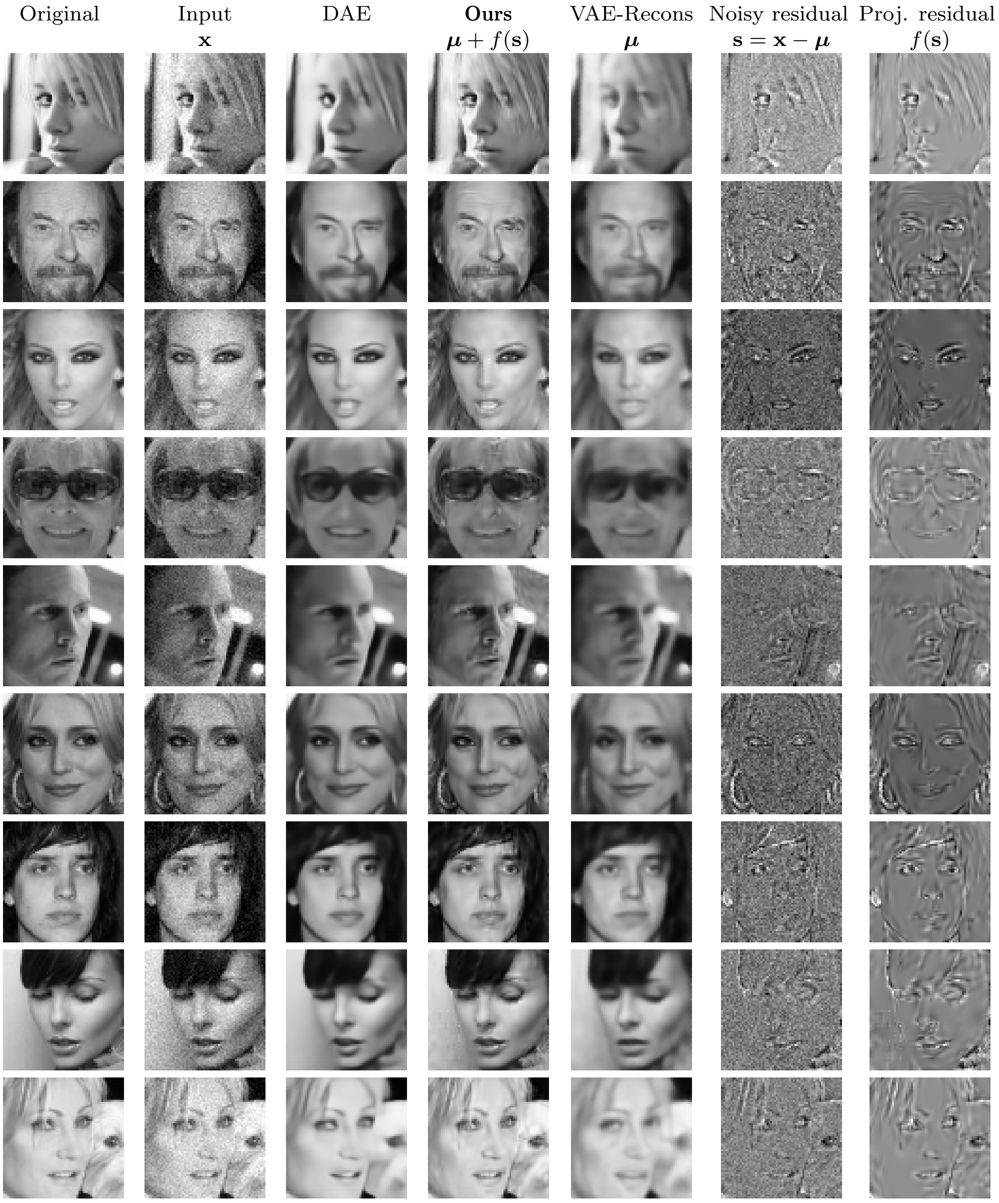


Figure A-20: Denoising experiment, left column: original image without noise, second column: image with added noise, third column: denoising autoencoder (DAE) result, fourth column: our result, fifth column: VAE reconstruction from the noisy input, sixth column: residual between the VAE reconstruction and the noisy input, right column: the noisy residual projected on $\hat{\boldsymbol{\Sigma}}$, the matrix constructed with 1000 eigenvectors of $\boldsymbol{\Sigma}$. Our result is the sum of the projected residual and the VAE reconstruction. Our model is able to recover fine details that are lost with the DAE approach.

Appendix B

WarpGAN

B.1 Network architectures

Our network architectures are based on the StarGAN model. In the generator all transpose convolutions are replaced with bilinear resizing followed by convolution, and instance normalisation is replaced by batch normalisation. For the discriminator the StarGAN architecture is used without any modifications. In both tables the following notation is used: N is the number of output channels, K is the kernel size, S is the stride size, P is the padding size, BN is a batch normalisation layer, r is the number of attributes, h and w denote the dimensionality of the input image.

Part	Input → Output Shape	Layer information
Downsampling	$(h, w, 3) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64, K4x4, S2, P1), Leaky ReLU
	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128, K4x4, S2, P1), Leaky ReLU
	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	CONV-(N256, K4x4, S2, P1), Leaky ReLU
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{16}, \frac{w}{16}, 512)$	CONV-(N512, K4x4, S2, P1), Leaky ReLU
	$(\frac{h}{16}, \frac{w}{16}, 512) \rightarrow (\frac{h}{32}, \frac{w}{32}, 1024)$	CONV-(N1024, K4x4, S2, P1), Leaky ReLU
	$(\frac{h}{32}, \frac{w}{32}, 1024) \rightarrow (\frac{h}{64}, \frac{w}{64}, 2048)$	CONV-(N2048, K4x4, S2, P1), Leaky ReLU
Output layer D	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (\frac{h}{64}, \frac{w}{64}, 1)$	CONV-(N1, K3x3, S1, P1)
Output layer C	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (1, 1, r)$	CONV-(N(r), K$\frac{h}{64}$x$\frac{w}{64}$, S1, P0)

Table B.1: Architecture for the discriminator and the classifier networks, D and C. The kernel weights in the downsampling layers are shared by D and C.

Part	Input → Output Shape	Layer information
Downsampling	$(h, w, 3 + r) \rightarrow (h, w, 64)$	CONV-(N64, K7x7, S1, P3), ReLU, BN
	$(h, w, 64) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	CONV-(N128, K4x4, S2, P1), ReLU, BN
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	CONV-(N256, K4x4, S2, P1), ReLU, BN
Bottleneck	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), ReLU, BN
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block
Upsampling	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{2}, \frac{w}{2}, 256)$	Bilinear resize
	$(\frac{h}{2}, \frac{w}{2}, 256) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	CONV-(N128, K4x4, S1, P1), ReLU, BN
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (h, w, 128)$	Bilinear resize
	$(h, w, 64) \rightarrow (h, w, 64)$	CONV-(N64, K4x4, S1, P1), ReLU, BN
	$(h, w, 64) \rightarrow (h, w, 2)$	CONV-(N2, K7x7, S1, P1)

Table B.2: Architecture for the warping network, W, the last layer is the displacement field $\mathbf{w}$. All residual blocks have the same structure, as indicated by the first block.

B.2 Extended quantitative results

Accuracy vs identity preservation

Model	Smiling	Big nose	Arched eyebrows	Narrow eyes	Pointy nose	Mean
StarGAN	0.65	0.60	0.64	0.66	0.68	0.65
StarGAN+	0.72	0.66	0.67	0.78	0.69	0.70
WarpGAN+	**0.83**	**0.73**	**0.81**	**0.87**	**0.82**	**0.81**
Real	1.00	1.00	1.00	1.00	1.00	1.00

Table B.3: Quantitative comparison of the identity score on real and generated images on the CelebA dataset evaluated with the face re-identification network, higher is better. These results correspond to the models highlighted in purple in Fig. 4-14, where the last column in this table contains the values for y-axis.

Model	Smiling	Big nose	Arched eyebrows	Narrow eyes	Pointy nose	Mean
StarGAN	0.84	0.60	0.69	0.65	0.62	0.68
StarGAN+	**0.92**	**0.73**	**0.87**	**0.75**	**0.82**	**0.82**
WarpGAN+	0.72	0.72	0.83	0.74	0.74	0.75
Real	0.92	0.81	0.82	0.88	0.72	0.83

Table B.4: Quantitative comparison of the attribute classification accuracy on real and generated images on the CelebA dataset evaluated with a separate classification network, higher is better. These results correspond to the models highlighted in purple in Fig. 4-14, where the last column in this table contains the values for x-axis.

Accuracy vs realism

Model	Smiling	Big nose	Arched eyebrows	Narrow eyes	Pointy nose	Mean
StarGAN	**0.85**	0.84	0.75	0.83	0.76	0.81
StarGAN+	**0.85**	0.84	**0.89**	0.86	0.83	**0.86**
WarpGAN+	0.63	**0.92**	0.83	**0.89**	**0.88**	0.84
Real	0.88	0.64	0.74	0.56	0.36	0.63

Table B.5: Quantitative comparison of the attribute classification accuracy on real and generated images on the CelebA dataset evaluated with a user study, higher is better. This table contains the x-axis values used in Fig. 4-15.

Model	Smiling	Big nose	Arched eyebrows	Narrow eyes	Pointy nose	Mean
StarGAN	0.40	0.52	0.62	0.41	0.74	0.52
StarGAN+	0.37	0.40	0.59	0.38	0.60	0.46
WarpGAN+	**0.42**	**0.64**	**0.79**	**0.57**	**0.82**	**0.65**
Real	0.97	0.89	0.98	0.96	0.94	0.95

Table B.6: Quantitative comparison of image realism both on real and generated images on the CelebA dataset evaluated with a user study, higher is better. This table contains the y-axis values used in Fig. 4-15.

B.3 Additional qualitative results

CelebA

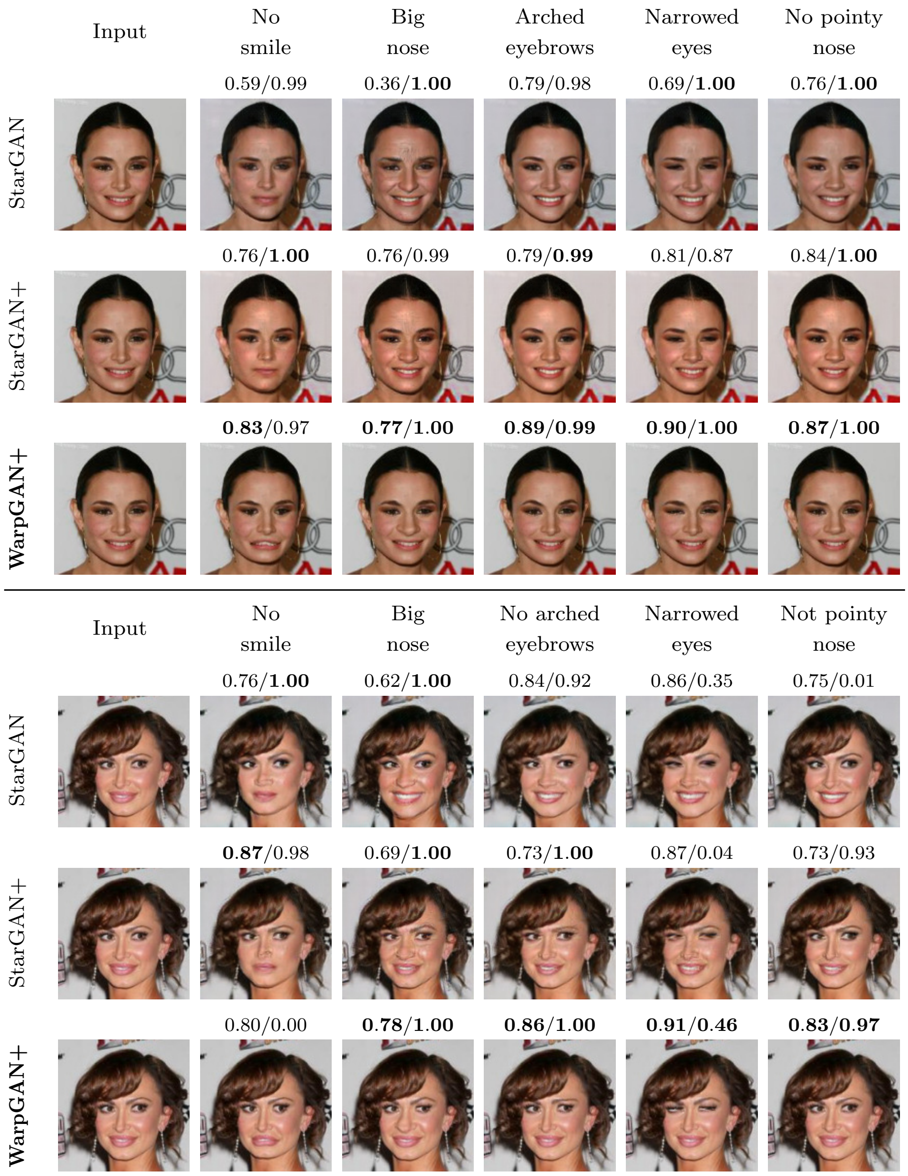


Figure B-1: Additional results on the CelebA dataset. From a given input image, first column, each method attempts to transfer the semantic attribute in its corresponding column. An identity score and attribute transfer score are shown as (id / cls) on top of each image (higher is better for both).

RafD

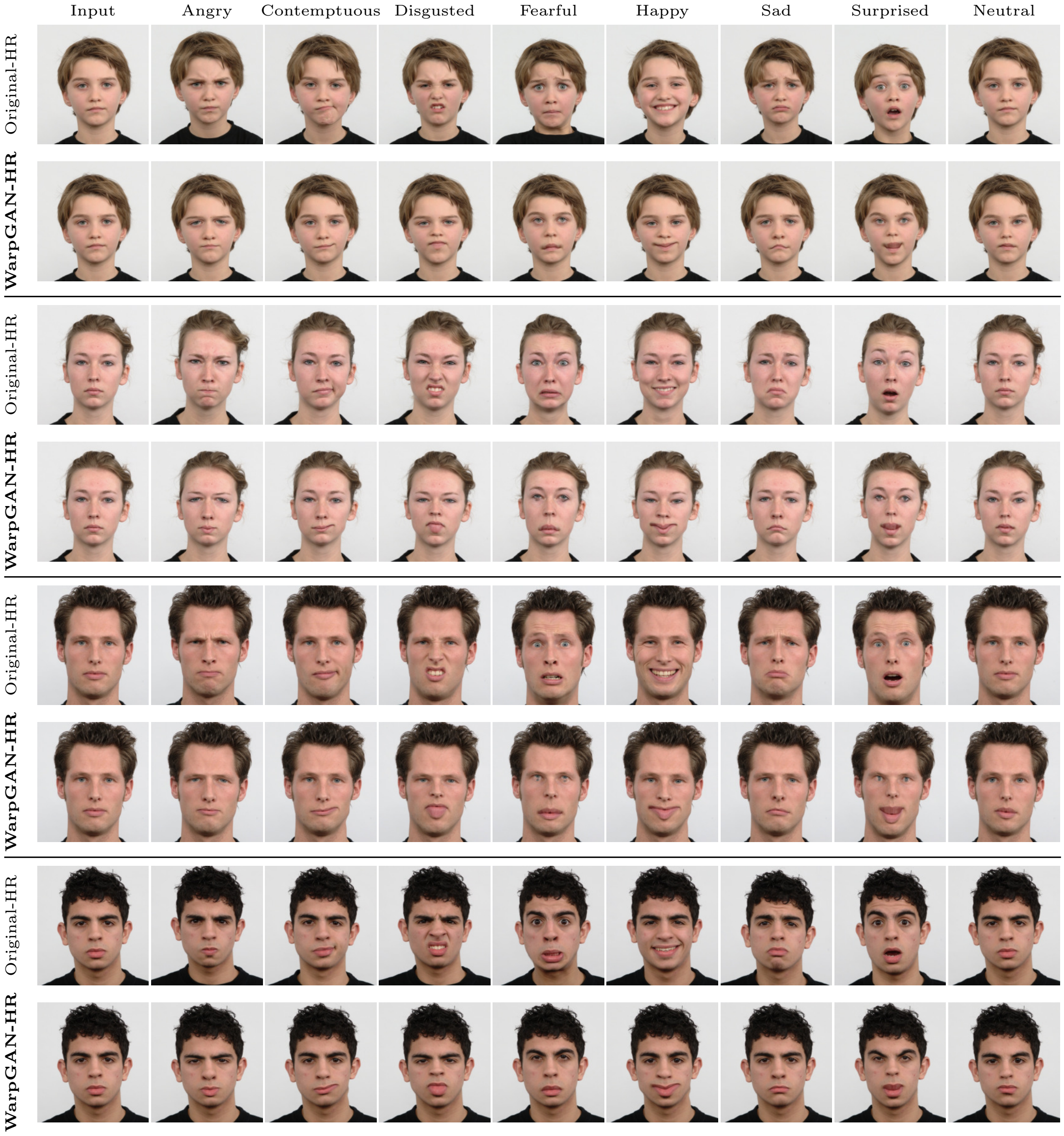


Figure B-2: Additional results on the RafD dataset for our model at the original dataset resolution (580×540). The input image is shown in the first column, and each method attempts to transfer the semantic attribute of the corresponding column. The dataset contains paired examples for each attribute, which are shown as Original-HR for each subject. (Zoom in for details)

Flickr-HR

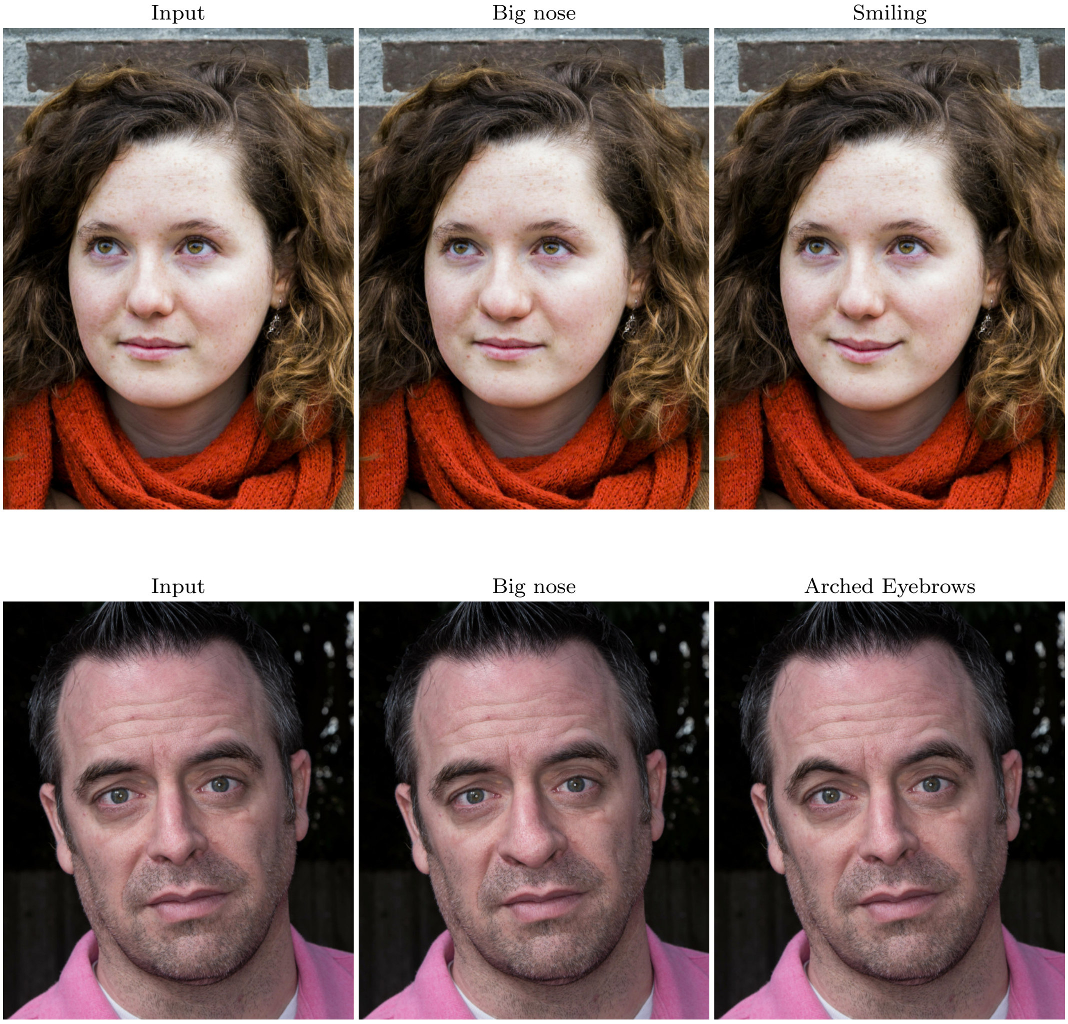


Figure B-3: Additional results of our model on high-resolution images. Our model predicts warps at low resolution that can then be resized and applied to high resolution images. The model is able to keep the content and identity at high resolution. Please see supplemental videos demonstrating animated edits. (Zoom in for details)[1]

[1] Input images courtesy of Flickr users Kenneth DM and Randall Pugh.

Partial edits

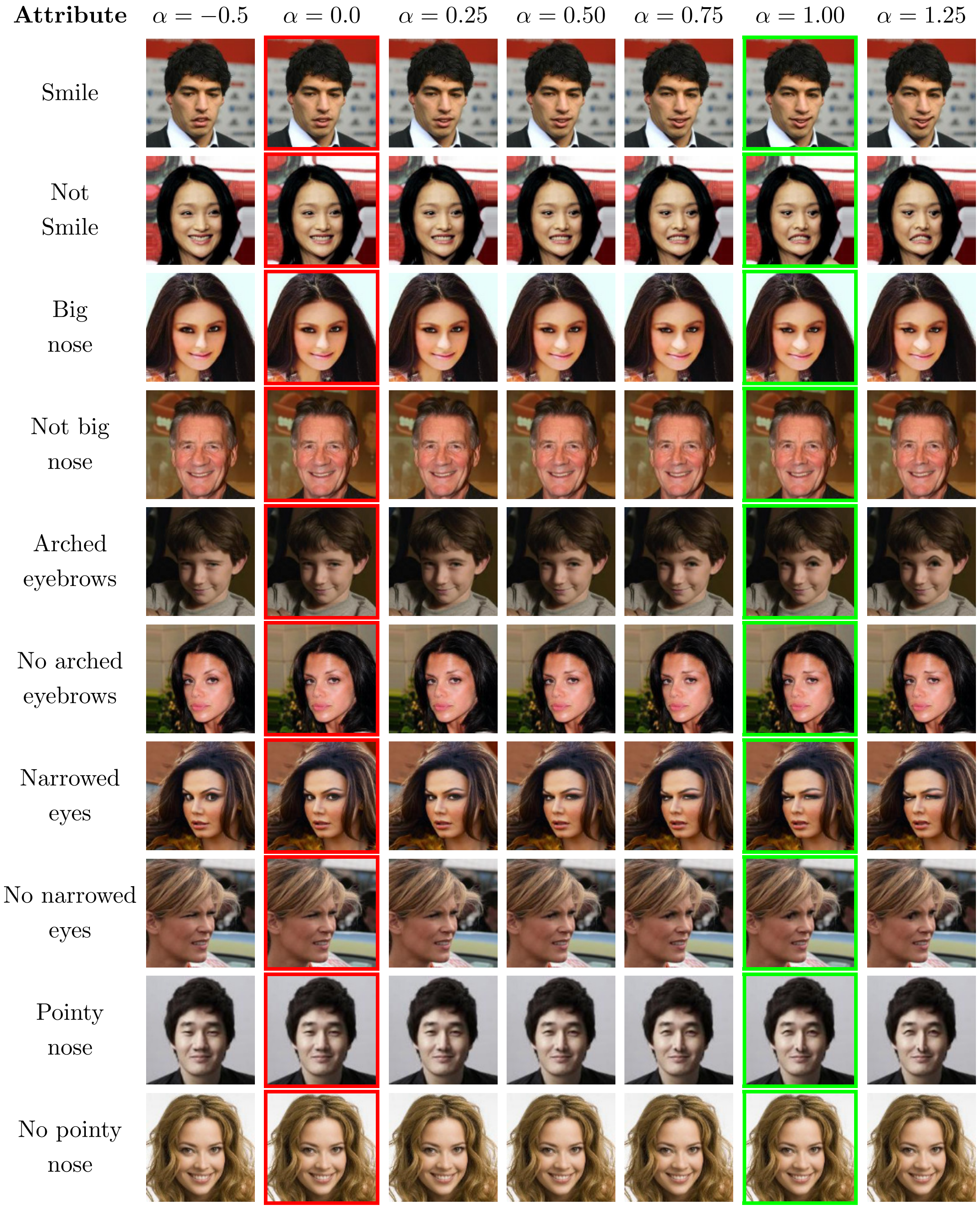


Figure B-4: Partial editing with our model, for the attribute indicated in the first column. A single warp is generated by our model, which is interpolated and extrapolated by scaling the magnitude of its values by α. The input image, $\alpha = 0$, is progressively edited in both directions. A red box denotes the input image, and a green one the output of the generator with $\alpha = 1$. Please see supplemental videos demonstrating animated edits.

Stretch maps

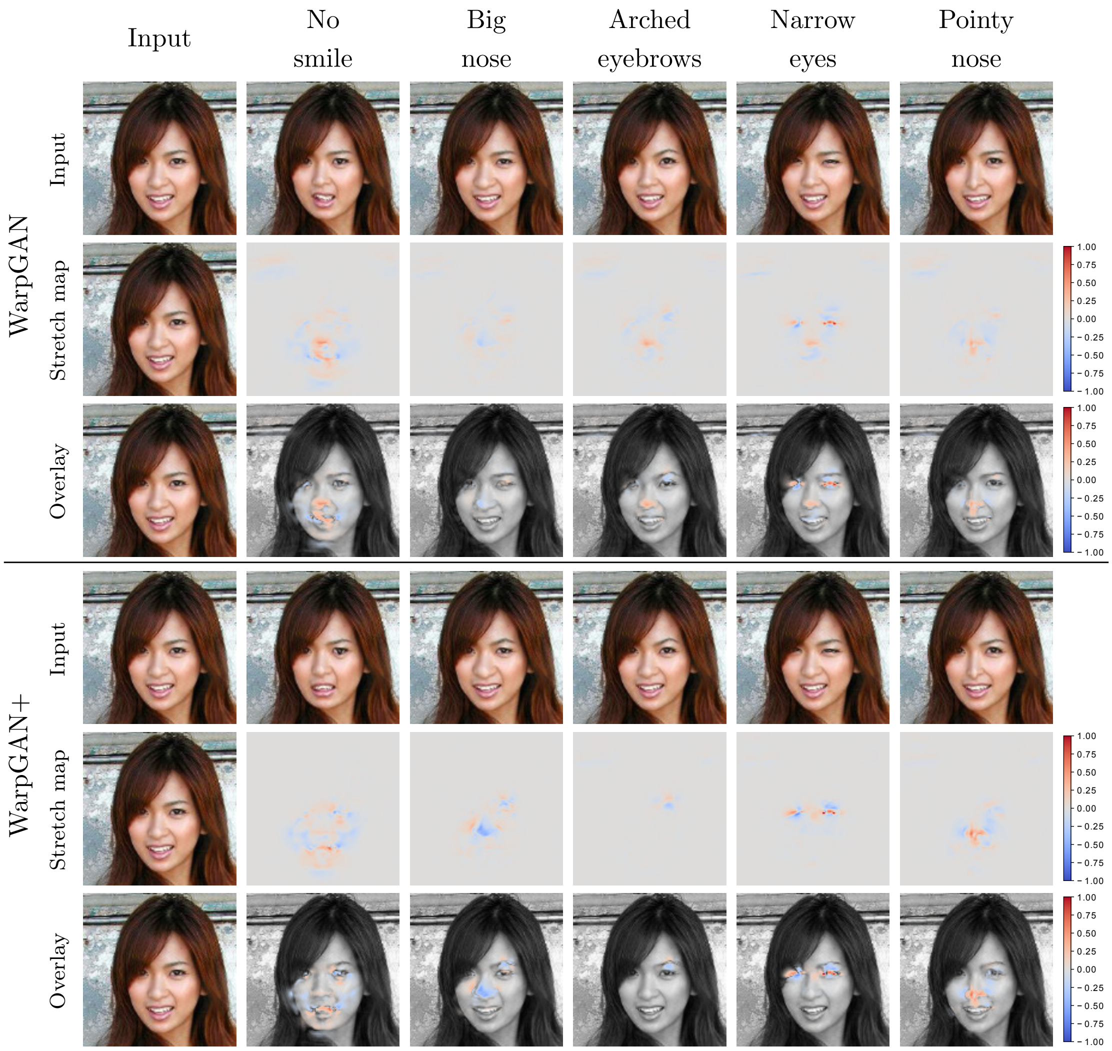


Figure B-5: Stretch maps computed from the warp fields, for WarpGAN and WarpGAN+. The log determinant of the Jacobian of the warp is shown, where blue indicates stretching and red corresponds to squashing. Using binary labels (WarpGAN) leads to an accentuation of all attributes, while using signed labels (WarpGAN+) leads to correctly localised edits for most attributes.

Cub200

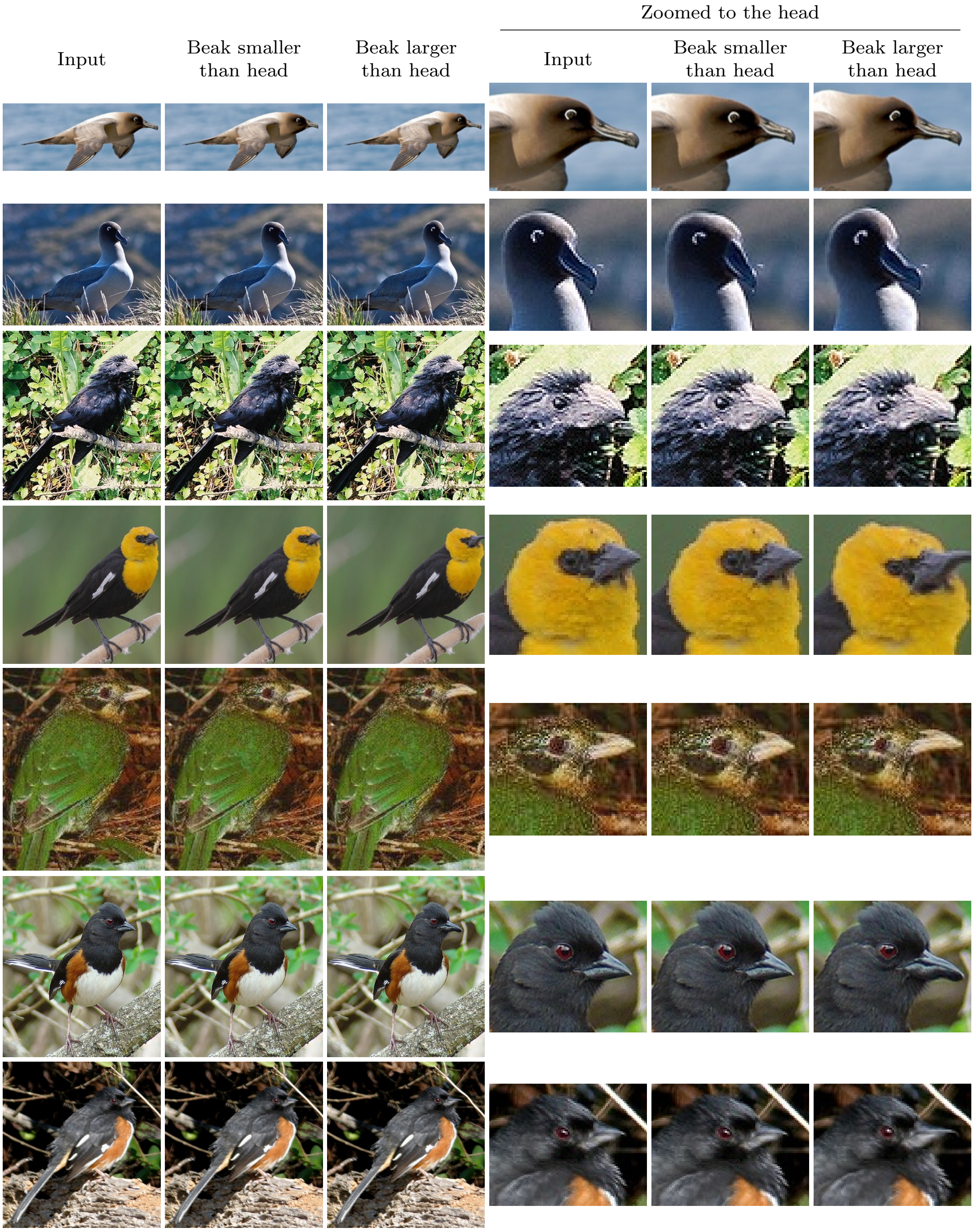


Figure B-6: Preliminary results from our model on test images from the Cub200 dataset [Wah et al., 2011]. The model attempts to transfer the attribute (relative beak size) in each column to the input image. For easiness of comparison, a crop of the head area is shown in the last three columns.