DTU Compute
Department of Applied Mathematics and Computer Science

Image Super-Resolution using Deep Variational Autoencoders

Master Thesis

Image Super-Resolution using Deep Variational Autoencoders

Master Thesis
August, 2021

By
Darius Chira and Ilian Haralampiev

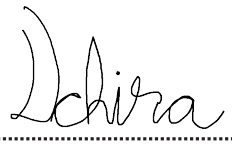
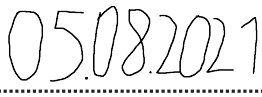
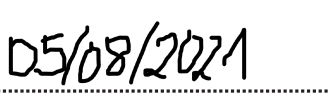
Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Approval

This thesis has been prepared over six months at the Department of Applied Mathematics and Computer Science, at the Technical University of Denmark, DTU, in partial fulfilment for the degree Master of Science in Computer Science and Engineering.

It is assumed that the reader has a basic knowledge in the areas of Machine Learning, Deep Learning and Image Analysis.

Darius Chira - s193157	Ilian Haralampiev - s193163
	
Signature	*Signature*
	
Date	*Date*

Abstract

Image super-resolution (SR) techniques are used to generate a high-resolution image from a low-resolution image. Deep Generative Models (DGMs) such as autoregressive (AR) models, Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have proven to be effective at modelling high-resolution images. In particular, GAN-based SR models have achieved state-of-the-art SR performances. Whereas VAEs used to be criticised for their feeble generative performances, there is now strong evidence that deep VAEs can outperform both GANs and AR models for high-resolution image generation. The project aims at exploiting the most recent deep VAE, specifically the state-of-the-art VDVAE generative model to improve upon existing SR models.
This project uses transfer learning to utilize the VDVAE model combined with an extra component in order to support the task of image super-resolution with four times upscaling (x4). The model we present, which we call VDVAE-SR, was able to output sharper images than a purely Convolutional Neural Network method EDSR (winner of NTIRE17 SR Challenge) in terms of FID score and even comparing to the popular GAN-based SR method ESRGAN (winner of PIRM-2018 SR Challenge).

The source code can be found on: https://github.com/dman14/VDVAE-SR

Acknowledgements

[Ole Winther], [Professor], [Department of Applied Mathematics and Computer Science]

[Andrea Dittadi], [Phd Student], [Department of Applied Mathematics and Computer Science]

[Valentin Victor David Julien Lievin], [Phd Student], [Department of Applied Mathematics and Computer Science]

We would like to thank our Professor, Ole Winther, for helping and offering us the opportunity to work on this Master Thesis after participating in his Deep Learning course at DTU, which provided us with the prerequisites needed for this work.

Also, we would like to give our thanks to Andrea and Valentin, acting as our supervisors throughout the whole thesis period, helping us with feedback, the development of the project, and all the other problems that we encountered on the way.

Contents

1 Introduction

1.1 Image super-resolution

Image super-resolution has long been considered as one of the harder tasks in image processing until the emerging of deep learning with its huge progress in the field of computer vision. The super resolution task and more precisely single image super-resolution SISR consists of producing a high-resolution image from its low-resolution counterpart. Great improvements in the field have started with purely Convolutional Neural Network models and currently dominated by GAN-based and autoregressive methods.

1.1.1 Main Objective

Until now, Variational Autoencoders (VAEs) based models haven't made an impact in the field, which could be explained as they have been criticised for their feeble generative performances compared to other state-of-art generative models. However, recent improvements in VAEs with the introduction of hierarchical structure of latent variables and findings that performance can be enhanced by greater stochastic depth among others, show strong evidence that deep VAEs can outperform both GANs and AR models for high-resolution image generation.

The main objective of this thesis is to use such models and adapt a state-of-the-art deep VAE for the task of super-resolution.
Some of the intentions of this thesis is to achieve:

- Review state-of-the-art Deep VAE models (LVAE, NVAE, VDVAE)
- Review state-of-the-art techniques for SR (SRGAN, ESRGAN)
- Implement and test baseline SR models
- Develop an SR model based on VDVAE
- Design an experimental protocol to compare our methods against existing methods

1.1.2 Applications

Even though the project and the topic are more research oriented, there are vast amount of applications and services using similar techniques in the industry. This can range from graphic processing tasks, as super-sampling for spatial anti-aliasing, or virtual super-resolution for rendering, digital zoom in cameras, or photo resolution restoration.
During the recent years, as Deep Learning became more popular as well, upscalling methods using it started to emerge. One example of this is the Deep Learning Super Sampling (DLSS) developed by NVIDIA that by using dedicated tensor core were able to achieve higher FPS (frames-per-second) and lower computation strain in video processing.
This is one area where Deep VAEs could improve the present state-of-the-art as the synthesis time is generally much lower for them when compared to autoregressive models. Child 2020.

2 Related Work

In this section, we present a carried out literature review divided in two parts. The first part focuses on deep variational autoencoders, where more details about two recent and best performing works in the field are given, as well as a brief comparison between them. The second part consists of different methods for image super-resolution that have achieved best results and they have been divided in categories depending on the type of the used model.

2.1 Generative models

The deep generative models are separated into three categories: Variational autoencoders (VAEs), Generative adversarial networks (GANs) and Autoregressive models (AR). The VAEs (Kingma and Welling 2013) generate new data, sampling from a learned latent space. They model an explicit but intractable density function, which cannot be optimized directly, so the lower-bound of likelihood is optimized instead. GANs (Goodfellow et al. 2014), on the other hand, perform an implicit density estimation and learn to generate from true data distribution using a game theory approach. The third category - autoregressive models, use explicit density estimation as the VAEs, but the density function is tractable, meaning that they are able to optimize the exact likelihood. An example of an autoregressive model is the PixelCNN (Oord et al. 2016), which generates pixels sequentially depending on previous pixels of the image.

In terms of comparison between the three methods, the generation process of the autoregressive model such as PixelCNN is much slower as it is performed sequentially. Based on training, GANs have long been considered as tricky and unstable to train, but able to achieve very good sample quality once training is carried properly. The VAEs have mostly been known for their not so good sample quality, but latest research show significant improvement by using more flexible approximations and adding structure to the latent variables. In the next section we are describing some of the most recent papers that use VAE architecture, while achieving state-of-the-art results in image generation.

2.2 Deep Variational Autoencoders

Standard Variational Autoencoders have been criticized for their inability to accurately model complex datasets and that they generate blurry images, which has been considered to be because of the usage of a similarity metric in pixel space (Dosovitskiy and Brox 2016). More recent work however, argues that the poor performance of VAEs has been due to lack of model expressiveness when modelling more complex input distributions and shows that deep hierarchical VAEs can model complex data structures and generate sharp natural images (Maaløe et al. 2019).

Standard VAEs that consist of several layers of stochastic latent variables have been problematic to train because of the so called prior collapse, where the top stochastic latent variable collapses into the prior. The Ladder Variational Autoencoder (LVAE) (Sønderby et al. 2016) is one of the first works that handles this issue by proposing a new inference model which shares information with the generative model resulting in an interaction between the bottom-up and top-down model of the VAE. However, it has been observed that in practice LVAEs, with a very deep hierarchy of stochastic latent variables, experience variable collapse (Maaløe et al. 2019).

One of the most recent works in the field is the Nouveau VAE (NVAE) (Vahdat and Kautz 2020), which is a deep hierarchical VAE built for image generation. The authors claim that their goal is to ”make VAEs great again” and they introduce a network architecture with a main building block being depthwise convolutions in the generative model. The depthwise separable convolutions (Chollet 2017) increase the receptive field of the network, which improves modelling long-range correlations in data and specifically for faces - the uniform skin tone and left-right symmetry. Another difference from previous work is that the NVAE uses batch normalization (BN), which the authors have investigated that it is an important component for successful results of deep VAEs. Additionally, they propose techniques for solving instability issues during training, such as residual parameterization of the approximate posteriors and spectral regularization. Another important technique that NVAE uses is the inverse autoregressive flows (IAF) (Kingma, Salimans, et al. 2016) in the encoder network, which is a type of normalizing flow with the main feature that it scales well to high-dimensional space. This technique provides more expressive distributions and since its only used in the encoder network, sampling time is not increased because of it. In terms of quantitative results however, the authors find out that the performance of the NVAE is only slightly improved by the usage of flows. Nevertheless, the NVAE outperforms the then-current state-of-the-art non-autoregressive and VAE models and closes the gap with autoregressive ones. When looking into the qualitative results, the generated images of faces look sharp with much better hair details and diversity compared to two other flow models images presented in the paper. Vahdat and Kautz 2020

Another recent work in the field of deep VAEs is the Very Deep Variational Autoencoder (VDVAE) (Child 2020). In the beginning of the paper, the authors argue that deep VAEs should at least match the performance of autoregressive models. In this line of thought, they claim that depth is key in order to improve performance and achieve better results than autoregressive models, while proposing an architecture that can scale to a higher number of stochastic layers. This network architecture is composed by residual blocks similar to ResNet blocks (He et al. 2016), but not the same as the depthwise separable convolutions of the NVAE generative model. Another difference from the NVAE model is the usage of residual scaling, nearest-neighbor upsampling and gradient skipping for performance and stability improvement. When experimenting with a 48 layer network, where layers were not conditioned on each other, the authors of VDVAE found out that statistical depth, independent of other factors, is correlated with performance. In terms of quantitative results, VDVAE outperforms NVAE and PixelCNN type autoregressive models, achieving higher likelihoods with fewer parameters. Comparing it with autoregressive models specifically, the VDVAE generates samples much faster, which could be very beneficial in real-time applications. Child 2020

Comparing the NVAE and VDVAE model, the NVAE introduces some powerful components in their network architecture like depthwise separable convolutions, residual normal distributions, spectral regularization and normalizing flows, and provides quite a few techniques for improving the training of the model. On the other hand, the VDVAE focuses more on the idea that depth of the stochastic layers of the VAE matters on performance and shows that it can outperform other VAE and some autoregressive models, and achieving this without using the complex components of the NVAE.

2.3 Super-resolution

2.3.1 CNN Based

One of the first studies that achieved quite good results in the area of super-resolution is the SR-CNN (Dong, Loy, He, et al. 2015), based on a three-layer CNN structure, using a

bicubic interpolated low-resolution image as an input to the network. Afterwards, with the proposal of residual neural networks (ResNet) (He et al. 2016), which provide fast training and better performance for deep architectures, numerous works have adapted ResNets and its modifications. One such work is the SR-ResNet (Ledig et al. 2017), that was used in GAN implementation and another work is SR-DenseNet (Tong et al. 2017), where the model size was further expanded in order to improve performance. Another popular work, which we have seen many super-resolution methods compare with, is the Enhanced Deep Super Resolution network (EDSR) (Lim et al. 2017). It again uses ResNets similar to the architectures of SR-ResNet but with the modification that the authors remove batch normalization from the residual block. They achieve impressive results and get the first place on the NTIRE2017 Super-Resolution Challenge.

2.3.2 GAN Based

One of the currently best performing image super-resolution algorithms is the ESRGAN (X. Wang et al. 2018), which is an enhanced version of the SRGAN (Ledig et al. 2017). The SRGAN is based on gated adversarial networks with the idea that the low-resolution image is passed as an input to the generator network and the discriminator network tries to distinguish between the super-resolution output produced by the generator network and the original high-resolution image. The authors of SRGAN argue that PSNR does not necessarily reflect perceptually better SR results and use an extensive mean opinion score (MOS) for evaluating perceptual quality. With that in mind, SRGAN introduces a perceptual loss different from previous work, based on adversarial as well as content loss. The ESRGAN builds up on SRGAN by improving the architecture adding Residual-Residual Dense Block (RRDB) for better performance compared to the ResNet used in SRGAN. Also, enhanced discriminator is used based on Relativistic GAN (Jolicoeur-Martineau 2018) and the features before the activation loss are used to improve perceptual loss. With these modifications, ESRGAN is able to achieve first place on the PIRM-SR Challenge 2018 (Blau et al. 2018) with the best perceptual index.

2.3.3 Transformer Based

Since the introduction of Transformers by (Vaswani et al. 2017), Transformers and attention based architectures have been dominating the field of Natural Language Processing, quickly becoming best performing when compared to other methods used at the time as LSTMs or RNNs. Some of the most popular implementations being models as: BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2019), GPT (Generative Pre-trained Transformer) (Radford and Narasimhan 2018)(Liu et al. 2019), or T5 (Text-to-Text Transfer Transformer) (Raffel et al. 2020).

With the impressive results in other fields, transformers entered in vision too, being used for multiple tasks as image classification, object detection, and even super-resolution, with models such as Yang et al. 2020 which achieve really good results in super-resolution, more specifically in texture recovery. In order to achieve this, multiple parts come together in the final model, including a Texture Transformer consisting of a learnable texture extractor, relevance embedding module, hard-attention module for feature transfer, and a soft-attention module for feature synthesis. Another detail about the model is that in order to attain more details about the low-resolution textures, the model uses 4 images as input: the low-resolution image, the high-quality reference image, a 4x bicubic upsampled low-res image, and a ref image that has been down-sampled and up-sampled sequentially.

Transformers also showed impressive results in image generations, as shown in Esser, Rombach, and Ommer 2021 which showed the potential of them in high-resolution Images by heavily improving over previous state-of-the-art techniques in related fields as image

completion. Also with the advancements of Dosovitskiy, Beyer, et al. 2020, transformers have been seen achieving really good results when compared with CNNs over the same amount of training time.

2.3.4 VAE Based

One recent work that has adopted a VAE architecture for image super-resolution is the srVAE (Gatopoulos, Stol, and Tomczak 2020). In this work, the authors use a bijective model to obtain a richer prior distribution corresponding to RealNVP (Dinh, Sohl-Dickstein, and Bengio 2016). Furthermore, the architecture consists of three latent variables: y is a downscaled version of the original image and u and z are stochastic latent variables. The interaction between the variables is represented on the probabilistic graphical model on fig. 2.1.

(a) Generative part

(b) Variational part

Figure 2.1: srVAE Graphical model (Gatopoulos, Stol, and Tomczak 2020)

The joint probability is described as:

$$p(p, z, y, u) = p(x|y, z)p(z|y, u)p(y|u)p(u)$$

And the approximate posterior is expressed as:

$$q(z, y, u|x) = q(z|y, x)q(u|y)q(y|x)$$

The idea behind using the second stochastic variable u is that it allows to capture the low-level information as it is based on the compressed representation y, while the latent variable z produces the high-level features of the data.

In order to generate a super-resolution image given its low-resolution counterpart y, the process is as follows: $u \sim q(u|y) \rightarrow z \sim p(z|y, u) \rightarrow x \sim p(x|z, y)$.

For the encoder and decoder, the authors use DenseNet (G. Huang et al. 2017) architecture. Gaussian as well as discretized logistic mixture likelihood (Salimans et al. 2017) probability distributions are used in the model. Based on the qualitative results, the authors conclude that the novel two-step process is able to successfully first capture the global structure of the image through the latent variable u and then add the local structure using the latent variable z resulting in a sharp image.

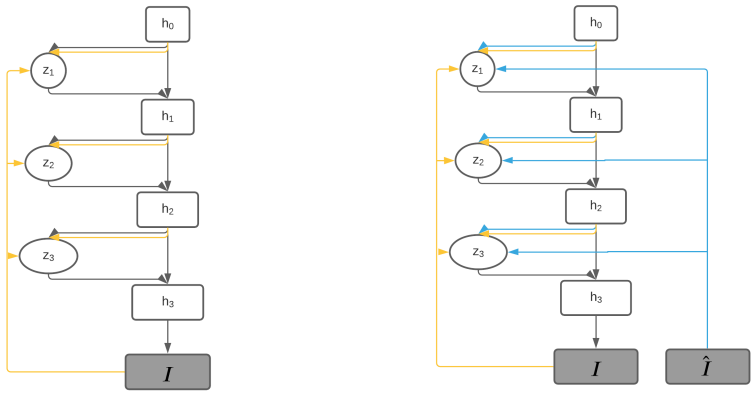
Another relevant work introduces the AIPO and AIPO-R models (Harvey, Naderiparizi, and Wood 2021) in the field of image completion with the possibility to be extended for super-resolution tasks as well. The authors of the paper use the aforementioned VDVAE model taking advantage of its hierarchical architecture in order to produce images with high-level details. In order to condition the VAE on the incompleted image, they propose a partial encoder $\hat{q}(z|\hat{I})$ where $\hat{I}$ is the corrupted image. The probability distribution of the completed images given their incompleted counterparts is approximated as:

$$p_{model}(I|\hat{I}) \approx \int p_{model}(I|z)\hat{q}(z|\hat{I})dz$$

and sampling from this approximation can be achieved by: $z \sim \hat{q}(\cdot|\hat{I}) \rightarrow I \sim p_{model}(I|z)$.

The authors propose two approaches for training, where the first one is derived from maximising $logp_{model}(I|\hat{I})$ and the second one is based on maximising a lower bound on $logp_{model}(\hat{I})$ named AIPO (amortized inference with partial observations) and AIPO-R (amortized inference with partial observations and the reverse KL) respectively. For the AIPO model The lower bound $p_{model}(I|\hat{I})$ is similar to an unconditional VAE lower-bound $logp_{model}(I)$ with the difference that the prior $p(z)$ is replaced with the partial encoder $\hat{q}(z|\hat{I})$.

A graphical representation of the AIPO model is observed on fig. 2.2 on the right, while a hierarchical VAE unconditioned model is shown on the left. The architecture consists of 3 groups of latent variables, the blocks $h_o, ..., h_L$ represent the deterministic hidden states of the decoder, the orange lines represent the encoder, the black lines represent the decoder and the partial encoder is represented by the blue lines.



(a) Unconditioned VAE (b) AIPO

Figure 2.2: AIPO Graphical model (Harvey, Naderiparizi, and Wood 2021)

In order to speed up the training process, the authors of the AIPO model have used a pretrained generator and encoder, while training only the partial encoder. Based on the achieved quantitative and qualitative results, the authors conclude that the AIPO model outperforms the baselines that is compared with producing diverse and plausible image completions. Harvey, Naderiparizi, and Wood 2021.

2.4 Overview

Based on the conducted literature review, we observe that recent works on deep variational autoencoders show impressive results in generation tasks and especially the VD-VAE that focuses on depth as a key component of the architecture achieving state-of-the-art performance. In the field of image super-resolution, GAN-based are currently one of the best performing in the field, while emerging Transformer-based methods show a great potential. To the best of our knowledge, there are not any studies that use deep variational autoencoders for super-resolution. Based on the notable results of VDVAE for image generation, we believe that with the right adaptation, it should be able to obtain results at least as close to the current top performing algorithms in the area of super-resolution.

2.5 Experimental protocol

In order to compare our model with other image super-resolution techniques, we built an experimental protocol where we include EDSR - Convolutional-based and ESRGAN - GAN-based methods. We were able to find code and official pre-trained models from the creators of those methods on Github: Son 2017 and X. Wang 2018.

To make sure the output the same results as in their corresponding papers, we tested them on common datasets for all of them. We use the bicubic interpolation to downscale the images and then calculate the quality measure metrics PSNR and SSIM on the YcbCr channel. These processing techniques are common for the two papers and for most of the super-resolution methods overall, which we explain further in section 3.4.

We show the PSNR and SSIM scores on table 2.1 , where it can be seen that the produced results for each model are very close to the official metrics on the papers.

Model/	Set5		Set14		Urban100	
Dataset	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
EDSR exp. prot.	32.09	0.88	28.33	0.77	26.04	0.78
EDSR official	32.46	0.90	28.80	0.79	26.64	0.80
ESRGAN exp.prot.	32.71	0.88	28.74	0.78	27.04	0.81
ESRGAN official	32.73	0.90	28.99	0.79	27.03	0.81

Table 2.1: Comparison between official paper scores and our experimental protocol

Numerical differences can be expected as some of the functions, for example the bicubic interpolation is used on the actual MATLAB software for some of the papers and others use a non-official Python implementation of that function. However, when performing the comparison of our model to the others in chapter 6, we use the same functions as implemented in our experimental protocol to produce quality measure scores for all the models.

3 Set-Up

This chapter will focus on the general pre-requirements for the development of the model, as the general set-up, used data and data sets, preprocessing of the data, and training and test methods employed for the development and for the benchmarking of the models made.

3.1 Data sets

During the development of the project, multiple datasets were used in order to satisfy the criteria that we have set.
For the reasons of making the benchmarks and comparisons closer to the truth later on, it was decided to use the same datasets that most of the other models are using for both testing and training.

3.1.1 Training Dataset

For training, the DIV2K dataset, introduced by Agustsson and Timofte 2017, was used. The DIV2K dataset consists of 800 RGB high definition high resolution images for training, 100 images for validations, and 100 pictures for testing. As seen in fig. 3.1, there are a diversity of pictures and types of pictures in the dataset, not limited by any restrictions as the type of shot (portrait, scenery, object shot, etc.).

Figure 3.1: Example images from the DIV2K dataset

3.1.2 Testing Datasets

For the testing sets, the most popular benchmarking datasets commonly used for this purpose were used. between them there are:

- Set5: Firstly introduce by Bevilacqua et al. 2012 it's a dataset consisting of 5 pictures often used with the purpose of testing performance of image related tasks.
- Set14: Introduced by Zeyde, Elad, and Protter 2010 consists of 14 images. Adopted by the image processing field, these images are often seen and widely recognised.
- T91: is a dataset from *BasicSR (Basic Super Restoration)* n.d. mainly consisting of cropped images of flowers.
- General100: Found in Dong, Loy, and Tang 2016, the dataset made for super-resolution, consists of 100 pictures with a varying sizes from 710 x 701 to 131 x 112.
- Urban100: From J.-B. Huang, Singh, and Ahuja 2015 is a dataset of 100 images consisting of building and urban scenes.

- BSD100: Seen firstly in Martin et al. 2001, the Berkeley Segmentation Dataset and Benchmark was made with the purpose of creating a more standardized process of comparing results and papers.
- Manga109: compiled and introduced by Matsui et al. 2016, it consists of 109 images of manga volumes drawn by manga artists from Japan. intended to be used for academic research on the media processing of Japanese manga, it is often used as a benchmark dataset.

Figure 3.2: Example images from the test datasets. In order: Set14, BSD100, Urban100, Manga109

3.2 Data preprocessing

Before using the data as input to the networks, firstly it will be preprocessed to accommodate the fixed input size of the models and to ease the process of down-scaling and up-scaling. There are two steps: the dataset preprocessing, and the model specific preprocessing.

3.2.1 Dataset preprocessing

When creating a dataloader for a specific model, all the images are going to get preprocessed in order to deal with different restrictions. This is exemplified in fig. 3.3

- After loading the data, as the size of all images vary, they will be resized according to the specification of the used model, or specific resolution that will be tested.
- After the resize, some of the images have different ratios, and in order to normalize this aspect, the images are going to be center cropped to get square inputs.
- As a last adjustment before extracting the low-resolution image, the image is checked to be divisible by the scale factor, otherwise it will be readjusted again by one pixel.
- The image resulted afterwards is considered the high resolution image, or the ground truth in the analysis afterwards. The same image is going to be down scaled by the scaling factor and the low-resolution image, or the input to the networks will be obtained.
- As a last step, all the images are transformed into tensors in order to be able to be used as inputs to the models.

3.2.2 Model Specific preprocessing

After the standard preprocessing of the dataset, based on specific requirements of the different models implemented, another preprocessing function will be used. For the case of the VDVAE-SR, the images require a few extra adjustments before being used.
Firstly the scale of the data is going to be changed from a 0 to 1 range to -1 to 1. The

channels are going to be permuted, and then it's going to be multiplied by a shift and scaled based on the training dataset statistics. This can be seen in the last image in fig. 3.3.

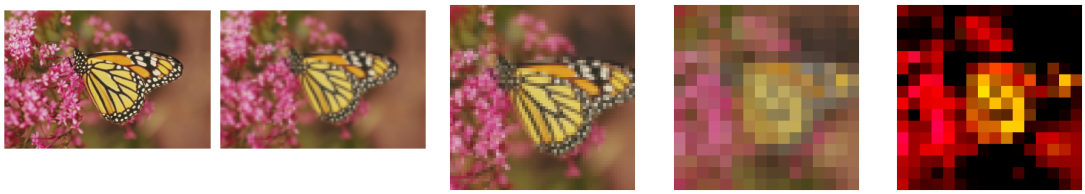


Figure 3.3: Example image through the pre-processing step. First Image is the original one, followed by a rescaled version, center cropped, down-scaled, and then normalized version.

3.2.3 Bicubic Interpolation

Both for downsampling the images for the pre-processing, and for upsampling them for testing purposes (ex. baseline) the Bicubic Interpolation method is used.
Bicubic Interpolation is a method often used for image processing, and super-resolution used as a baseline. The advantages of it are that is an easy to use method, fast, and better then others like bilinear or nearest-neighbor methods. The methods uses 4x4 pixel patches of the images, with the general formulation of it, as described in Gavade and Sane 2013, is:
Suppose the function values f and the derivatives fx, fy, and fxy are known at the four corners $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$, and of the unit square. The interpolated surface can then be written as:

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

The interpolation problem then consists of determining 16 coefficients a_{ij}.

3.3 Train Set-up

For both the development and training, multiple tools were used to ease the process. For the convenience of having access to a GPU, and to be able to work in the cloud with access to more data, Google Colab was used. Google Colab is a cloud service that gives access to a jupyter notebook environment with GPU capabilities. Asides from that, WandB was used in order to be able to track all the experiments, plot the needed graphs and extract samples from the models during training in an organized manner.
The Training Set-up was made in a way to be able to change models, hyper-parameters, datasets, and different options for the training in the fastest way in order to make the training, testing and development more streamlined for the higher number of needed experiments.

3.4 Test Set-up

In order to be able to test and compare the developed networks, a test set-up was made with multiple metrics to track. Then the testing routine accepts either a picture or a set of pictures that will be ran through the model, after which the comparison metrics are going to be calculated based on the output image and the reference image. The used metric are:

- PSNR (Peak Signal-to-noise Ratio): commonly used for measuring the quality of reconstructions for super-resolution, image generation, or compression algorithms, it is expressed as the ratio between maximum possible power of a signal and the power of corrupting noise, as described in Horé and Ziou 2010. Usually it is defined as:

$$PSNR = 10 \cdot log_{10}(\frac{MAX_I^2}{MSE})$$

  where MAX_I is the maximum possible value of a pixel (255 generally), and MSE is the mean squared error between the reconstructed image and the reference image.

- SSIM (Structural Similarity index measure): is a metric used for the perceived quality, measuring the similarity between two pictures.It is usually regarded as a closer metric to human perception and in some cases used for the same reason even though in works like Nilsson and Akenine-Möller 2020 it is shown that it is not adhering to human visual properties.Even with this being true, the measurement grew vastly in popularity after it's appearance, being very useful in benchmarks and comparisons. In Z. Wang et al. 2004 it is defined as:

$$SSIM_{(x,y)} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2) + c_2}$$

  Depending on factors such: luminance, contrast, and structure.

- FID (Fréchet inception distance) is another metric used to measure the the quality of pictures, usually from created images from generative models such as GANs or VAEs. the FID is based on a comparison between the distribution of two datasets: a generated one, and a real one. Instead of using and comparing the samples directly, the FID compares the activations of the last layer of a inception-v3 model, using the generated and real images as input. Heusel et al. 2018 proposed using the the Fréchet distance between multivariate gaussians like:

$$FID = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

  where X_r and X_g represent the multi-dimensional activations of a inception-v3 pool3 layer, for both the real and generated images respectively.
  Usually a 2048-dimensional activations are used, representing a dataset with 2048 samples, but methods with either higher and lower number of samples have been experimented with. In our case, most of the test datasets have a lower number of images.

All pictures are converted from RGB to YCbCr and the metrics are run on the Y channel (luma component) of the pictures. The reason for this is that it has been seen(as in Pisharoty, Jadhav, and Dandawate 2013) that the performance of the evaluation on the luminosity channel of a YCbCr instead of RGB provides better results and closer to the actual perceived structural noise of the image. It also serves for the convenience of benchmarking and comparisons as most of the related works also calculated their metrics on this representation of the images. Another note to be made is that this method of picture conversion is used during the testing phase exclusively, the training and validation still only using the RGB scale.

4 Deep Learning Methods

The Deep Learning Methods Chapter describes the methods and techniques used for the development of the model. Alongside VAEs, CVAEs, and Deep VAEs, other techniques as ResNets, CNNs, and specific activation functions will be presented, which are used in the model we are adapting in the next chapter.

4.1 Fundamentals

4.1.1 Convolutional Neural Networks

The main difference between an artificial neural network (ANN) and a convolutional neural network (CNN) is that in an ANN each neuron in a layer is connected to all the other neurons of the previous layer, while in a CNN each unit receives input only from a small subset of units known as the unit's receptive field.

The procedure is done using kernels where a kernel (ex. 3x3, 5x5 etc.) has learnable weights and is slid over the input performing element-wise multiplication and summation which results in convolved features. Afterwards, an element-wise non-linear transformation is performed. Figure 4.1 shows an example of an image, kernel filter and the resulting convolved features.

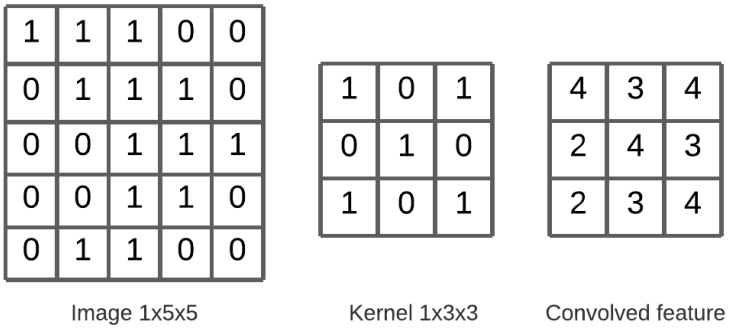


Figure 4.1: Convolution feature extraction

Pooling operations can also be performed and are usually used for down-sampling the input. Such operations do not involve any weights and they are a simple way to change the dimension of the data by just selecting the size of a window/patch that is slid over the input. Most popular pooling techniques are max and average pooling, where for max pooling, the maximum value of the window is the output for each window. For average pooling, the output of the pooling operation for each window corresponds to calculating the mean value.

4.1.2 Activation functions

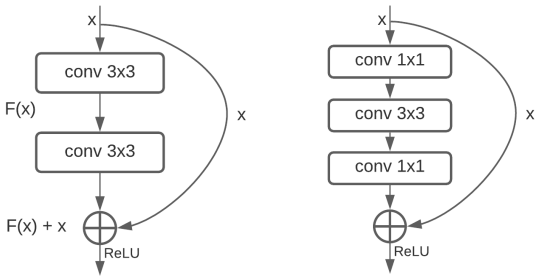
The activation functions are used to add non-linearity and are usually applied on each layer of a neural network. One of the most popular activation functions for deep neural networks is the ReLU, which maps all negative values to zero while keeping the same value for all the positive ones: $ReLU(x) = max(0, x)$. Another activation function that we need to describe as it is used in the model we are adapting for image super-resolution is the Gaussian Error Linear Unit or GeLU (Hendrycks and Gimpel 2016) which is performed

by multiplying the input with the Cumulative Distribution Function (CDF) for Gaussian Distribution $\Phi(x)$ as: $GeLU(x) = x \cdot \Phi(x)$.

4.1.3 ResNets

Residual networks or ResNets were first introduced in He et al. 2016 in order to tackle the problem where the accuracy of networks with many number of layers rapidly decreases when the model is about to converge, also known as the $degradation$ problem in deep neural networks.

The ResNet consists of neural networks with skipping connections as shown on fig. 4.2a which represents one residual block.



(a) ResNet block (b) ResNet bottleneck

Figure 4.2: ResNet building blocks

The skip connections perform an identity mapping without any extra parameters and are added to the output of the stacked layers resulting in $F(x) + x$. This way, in the case where the identity mapping x is the optimal solution, it can be easily produced by just setting $F(x)$ to zero, instead of being learned by $F(x)$ in the conventional case without the skip connection. The original ResNet consists of 34 convolutional layers with 3x3 filters and ReLU activation function on the output of each block.

Another version of the ResNet is the bottleneck architecture made for deeper networks with a building block that consists of three layers instead of two where the middle one serves as a bottleneck with 3x3 convolutions, while the other two layers are 1x1. A diagram is shown on fig. 4.2b. In terms of results, the ResNets tested with 50, 101 and 152 number of layers outperform the 34-layer one. He et al. 2016

4.2 Vanilla Variational Autoencoder

The derivation of the Variational Autoencoder in this section mainly follows the notation used in the Deep Learning course at DTU (DTU 2020) and Doersch 2016.

4.2.1 Generative process and Intractability

As briefly introduced in 2.1, the Variational Autoencoder (VAE) is a generative model that generates new data by sampling from a learned sampled space using probabilistic principles. This way, the generative process includes the latent variable z, which is sampled from the prior $p_\theta(z)$ and the observation variable sampled from the observation model $p_\theta(x|z)$:

$$z \sim p_\theta(z), x \sim p_\theta(x|z)$$

In order to find a model that fits well the data, it can be defined by maximizing the total probability $p(x)$ as:

$$\theta^{\star} := \underset{\theta}{\text{argmax}}\, p_{\theta}(x) = \int_{z} p_{\theta}(x|z)p(z)dz \tag{4.1}$$

where $p(x)$ is parameterized by θ and the joint distribution $p_{\theta}(x, z)$ is expressed as the likelihood $p_{\theta}(x|z)$ and prior $p(z)$.

As the idea is to produce meaningful latent variables from which we can generate new data, we can define a posterior distribution according to Bayes Theorem:

$$p_{\theta}(z|x) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(x)}$$

However, in practice, $p(x)$ is intractable as it needs to be evaluated over all the different configurations of z, which makes the posterior $p(z|x)$ also intractable.

4.2.2 Variational Inference

In order to solve the intractability issue, approximation of the posterior is required and it is performed using Variational Inference (VI). The approximated posterior is introduced as:

$$q_{\phi}(z|x) \approx p_{\theta}(z|x)$$

and usually chosen among the Gaussian family, parameterized by the mean $\mu_{\phi}(x)$ and variance $\sigma_{\phi}(x)$.

With the introduction of the approximate posterior $q_{\phi}(z|x)$, the marginal probability $p_{\theta}(x)$ can be expressed as an expectation over the approximate posterior if we multiply and divide by it inside the integral as:

$$p_{\theta}(x) = \int \frac{q_{\phi}(z|x)}{q_{\phi}(z|x)} p_{\theta}(x, z)dz = E_{q_{\phi}(z|x)}\left[\frac{p_{\theta}(x|z)p_{\theta}(z)}{q_{\phi}(z|x)}\right]$$

4.2.3 Evidence Lower Bound

Based on Jensen's Inequality, which states that a function over an expectation is greater than or equal to the expectation of the function with respect to some random variable X, if the function is concave: $logE(X) \geq Elog(X)$, taking the logarithm on both sides of the previous equation and applying Jensen's Inequality results in:

$$logp_{\theta}(x) = logE_{q_{\phi}(z|x)}\left[\frac{p_{\theta}(x|z)p_{\theta}(z)}{q_{\phi}(z|x)}\right]$$

$$logE_{q_{\phi}(z|x)}\left[\frac{p_{\theta}(x|z)p_{\theta}(z)}{q_{\phi}(z|x)}\right] \geq E_{q_{\phi}(z|x)}\left[log\frac{p_{\theta}(x|z)p_{\theta}(z)}{q_{\phi}(z|x)}\right]$$

where the right hand side is equivalent to the Evidence Lower Bound (ELBO) and can be further derived to:

$$ELBO = E_{q_{\phi}(z|x)}\left[logp_{\theta}(x|z) + logp_{\theta}(z) - logq_{\phi}(z|x)\right]$$

The ELBO can be further expressed using KL divergence, which is a measure of similarity between two distributions and has the properties of being non-negative and not symmetric. The KL divergence for two discrete probability distributions P and Q is calculated by the following formula:

$$D_{KL}(P||Q) = \sum_x P(X) log \frac{P(X)}{Q(X)}$$

When applying the KL divergence, the ELBO can be written as:

$$ELBO = E_{q_\phi(z|x)} \left[log p_\theta(x|z) \right] - D_{KL}(q_\phi(z|x)||p_\theta(z)) \tag{4.2}$$

where the first term on the right hand side of the equation is known as the reconstruction error and the second term is the regularization. When optimizing the ELBO, there is a trade-off between these two terms. The reconstruction term measures the reconstruction quality, where the expectation value is taken over the approximate posterior $q_\phi(z|x)$ and the regularization term is pulling the approximate posterior to match the prior $p_\theta(z)$ as the prior is usually fixed.

4.2.4 Reparameterization trick

As the ELBO is the lower bound to the log-likelihood $log p_\theta(x)$, in order to maximize the likelihood, the ELBO needs to be maximized. This can be achieved using the backpropagation algorithm as the approximated posterior and generative model are implemented using neural networks. However, backpropagation is not possible to flow in the layer when sampling z from the approximate posterior $z \sim q_\phi(z|x)$ as there is a stochastic node within the network. To be able to backpropagate through that layer, the so called "reparameterization trick" is used, which changes the stochastic process to happen in an input node. This is achieved by first sampling a noise variable from a fixed distribution $\epsilon \sim N(0, I)$ and then computing z using a deterministic and differentiable transformation $z = \mu(X) + \Sigma^{1/2}(X) * \epsilon$. Doersch 2016.

4.2.5 Implementation

The implementation of Variational Autoencoders is similar to the classical autoencoders, which have an encoder and decoder architecture. For VAEs, the encoder represents the approximate posterior $q_\phi(z|x)$ that tries to produce meaningful latent variables from X and the decoder represents the observation model $p_\theta(x|z)$ that generates data points using the latent variables z. The approximate posterior is usually chosen as a diagonal Gaussian distribution:

$$q_\phi(z|x) = N(z|\mu_\phi(x), \sigma_\phi(x)I)$$

where the mean $\mu_\phi(x)$ and variance $\sigma_\phi(x)$ are the outputs of the neural network.

The prior $p_\theta(z)$ is usually chosen as a Gaussian with a mean of zero and variance of one $p_\theta(z) = N(0, I)$. The observation model $p_\theta(x|z)$ is implemented using a neural network where the input latent features are sampled from the approximate posterior during training and sampled from the prior when in generation mode.

4.3 Conditional Variational Autoencoder

In this section, we introduce the conditional variational autoencoder (CVAE) as in order to perform super-resolution we need to generate very specific data and this is not possible using only a vanilla VAE.

The loss for the conditional variational autoencoder is derived similarly as for a normal variational autoencoder, where again we want to maximize the marginal probability similar to 4.2 but this time conditioned on a random variable which could be for example a low-resolution image y:

$$p_\theta(x|y) = \int_z p_\theta(x|y,z)p(z|y)dz \tag{4.3}$$

Using Bayes Theorem with multiple conditions, the inference model can be expressed as:

$$p_\theta(z|x,y) = \frac{p_\theta(x|z,y)p_\theta(z|y)}{p_\theta(x|y)} \tag{4.4}$$

where again $p_\theta(x|y)$ is intractable and needs to be approximated using Variational Inference:

$$q_\phi(z|x,y) \approx p_\theta(z|x,y)$$

The ELBO can be derived once again using the Jensen's Inequality corresponding to:

$$logp_\theta(x|y) = logE_{q_\phi(z|x,y)}\left[\frac{p_\theta(x|z,y)p_\theta(z|y)}{q_\phi(z|x,y)}\right] \geq E_{q_\phi(z|x,y)}\left[log\frac{p_\theta(x|z,y)p_\theta(z|y)}{q_\phi(z|x,y)}\right] = ELBO$$

And further shown as:

$$ELBO = E_{q_\phi(z|x,y)}\left[logp_\theta(x|z,y) + logp_\theta(z|y) - logq_\phi(z|x,y)\right]$$

When applying KL Divergence, the ELBO can be expressed as the reconstruction and regularization term:

$$ELBO = E_{q_\phi(z|x,y)}\left[logp_\theta(x|z,y)\right] - D_{KL}(q_\phi(z|x,y)||p_\theta(z|y)) \tag{4.5}$$

In terms of implementation, the random variable y is added as an input to the encoder as well as the decoder. In addition, the prior becomes conditioned on the random variable y resulting in $p_\theta(z|y)$ which can be implemented as a neural network.

4.4 Very Deep Variational Autoencoder (VDVAE)

As briefly explained in chapter 2 the Very Deep Variational Autoencoder is the current state-of-the-art, showing the best results compared to other VAE-based models and even outperforming some autoregressive ones. Their work is mainly focused on increasing performance with greater depth, which has also been shown in previous works as BIVA (Maaløe et al. 2019) and NVAE (Vahdat and Kautz 2020). In this section, we will go more into detail about the main propositions of the paper, the type of variational autoencoders their research is based on, the architecture of the VDVAE and their experimental findings.

4.4.1 Hierarchical Variational Autoencoders

Hierarchical Variational Autoencoders, first introduced in Ladder Variational Autoencoder Sønderby et al. 2016, consist of multiple layers of latent variables which are dependent on each other. This results in a more flexible prior and posterior compered to just a diagonal Gaussian prior like the one used in Vanilla VAE section 4.2 which could be too limiting. The VDVAE conditioning structure is adapted from the LVAE, where an iterative interaction between a "bottom-up" and "top-down" layers is achieved due to parameter sharing between the inference and generative models in each layer. The prior and the approximate posterior for a K-layer model in the VDVAE paper are expressed as:

$$p_\theta(z) = p_\theta(z_0)p_\theta(z_1|z_0)...p_\theta(z_K|z_{<K})$$

$$q_\phi(z|x) = q_\phi(z_0|x)q_\phi(z_1|z_0, x)...q_\phi(z_K|z_{<K}, x)$$

where $p_\theta(z_0)$ is represented as a diagonal Gaussian distribution $N(z_0|0, I)$ and the latent variable group z_0 is at the top layer that corresponds to small number of latent variables at low resolution. Intuitively, z_K is at the bottom of the network having larger number of latent variables at high resolution. (Child 2020).

4.4.2 Main propositions

The authors of the VDVAE have two main propositions in their argumentation that depth is an important factor for hierarchical variational autoencoders.

The first one states that hierarchical VAEs generalize autoregressive models when the number of stochastic layers of the VAE equals the dimension of the data D. The authors show theoretically that if the approximate posterior $q(z|x)$ just outputs the observed variables in a given order and if the generator $p(x|z)$ performs the identical function, the ELBO becomes $logp_\theta(x) = logp_\theta(z) = \sum_{i=1}^{N} logp_\theta(z_i|z_{<i}) = \sum_{i=1}^{N} logp_\theta(x_i|x_{<i})$, which is the equivalent of an autoregressive model over the observed variables. (Child 2020)

The second proposition states that shorter procedures such as a K-layer VAE, where K < D, are also learnable if such latent representation exist, which is usually the case for images. This means that these models could learn a low-dimensional representation of the data. The authors argue that the lowest possible value for K may be deeper than the VAEs implemented so far and this is why they test hierarchical VAEs with a depth above 30 stochastic layers. (Child 2020)

4.4.3 Architecture of VDVAE

Bottom-up path

The architecture of the VDVAE is shown on the diagram on fig. 4.3 where on the bottom-up path, the image passes through a number of residual blocks, followed by average pooling. These concepts were previously explained in section 4.1 where the residual block is with bottleneck architecture.

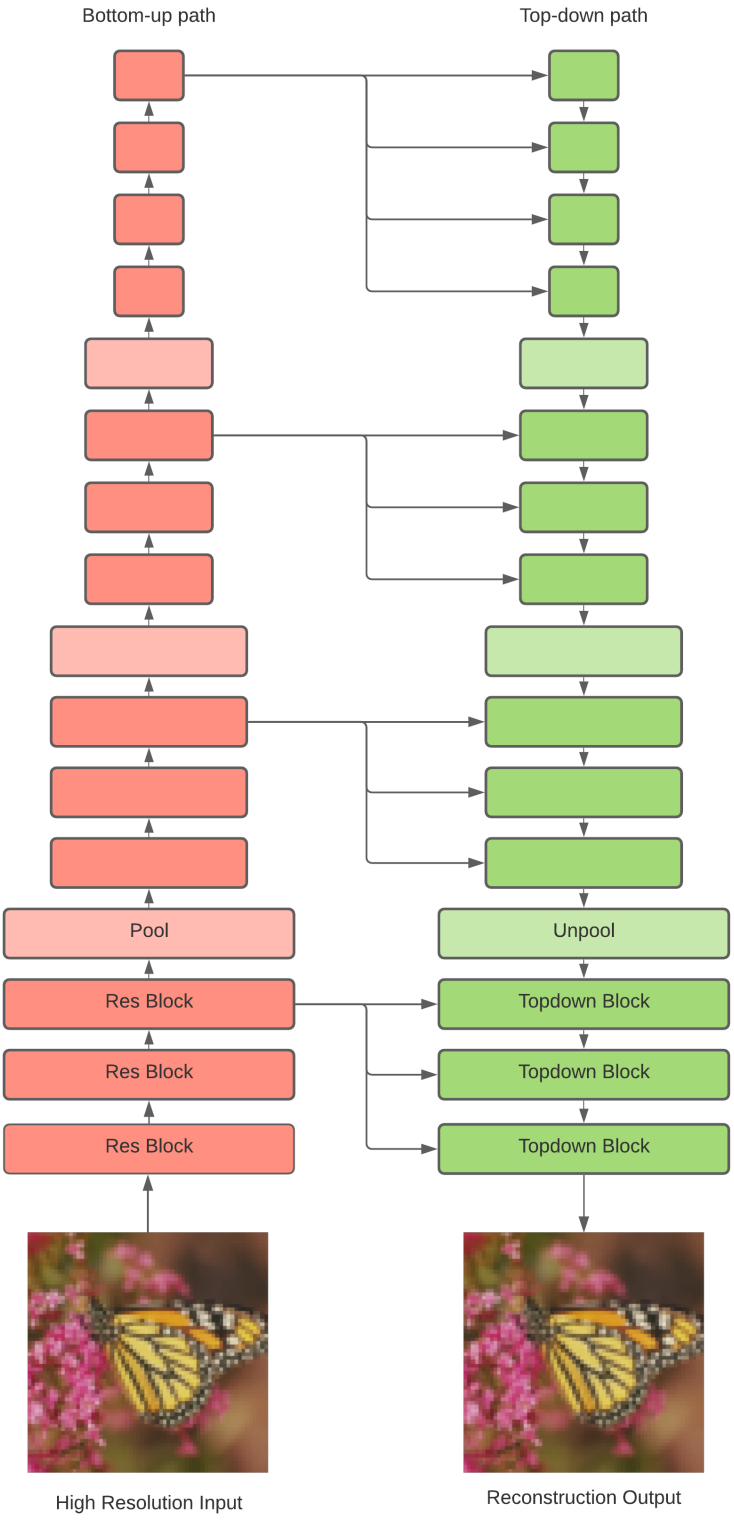


Figure 4.3: VDVAE Architecture

This procedure is repeated in order to calculate the features or so called ”activations” up to the top-most layer of the bottom-up path. Each convolutional layer of the residual block

shown on fig. 4.4 is followed by a GELU activation function that was previously mentioned in section 4.1.

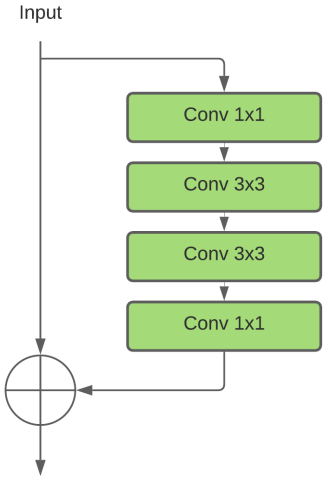


Figure 4.4: Res Block (residual block)

Top-down path
The top-down path consists of so called ”topdown” blocks shown on fig. 4.5

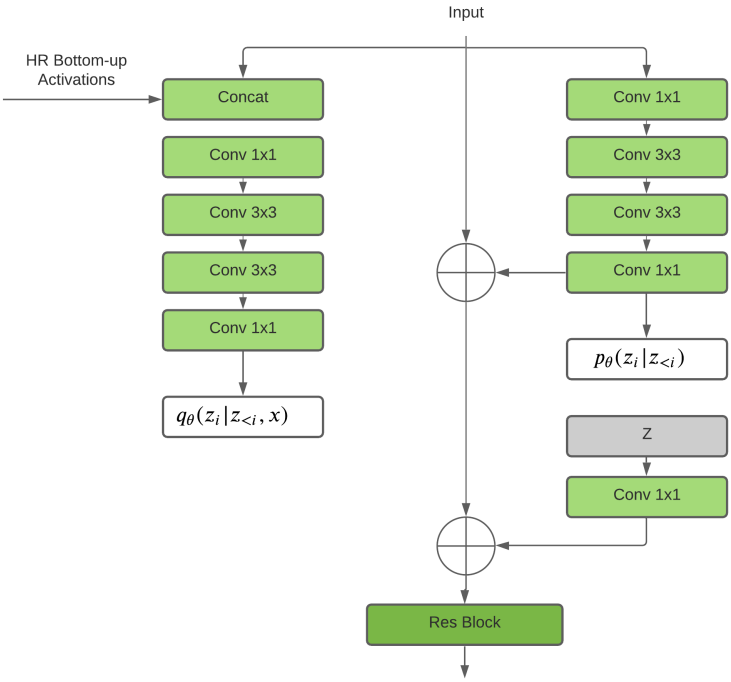


Figure 4.5: Top-down block

On the left side of the figure is where the approximate posterior for each layer $q_\phi(z_i|z_{<i}, x)$ is being calculated by concatenating the output of the previous topdown block with the activations from the bottom-up path and passing it through a convolutional neural network

that outputs the mean and variance vectors of the diagonal Gaussian distribution. The prior $p_\theta(z_i|z_{<i})$ on the right is also parameterized by a convolutional neural network and is also a diagonal Gaussian distribution. The unpooling layer consists of nearest-neighbor upsampling, which is a simple upsampling technique that just copies the value from the nearest pixel. During training, the latent variable z is sampled from the approximate posterior, and when in generative mode, z is sampled from the prior.

Additional components
For initialization of the weights throughout the network, the default PyTorch weight initialization is used. Only the final layer of each residual block is scaled by $\frac{1}{\sqrt{N}}$, with N being the depth. The authors quantitatively show that this scaling increases stability and performance for a model that consists of many number of layers. (Child 2020)

For training, the authors propose skipping the update of the weights when the gradient norm is above a specified threshold and they ensure smooth training of the model using this approach.

4.4.4 Experimental findings
Based on the performed tests, the authors found that statistical depth improves performance. They trained models on CIFAR-10, ImageNet-32 and ImageNet-64 with a larger number of stochastic layers, but with fewer parameters than related work and were able to achieve better results than all non-autoregressive models and outperforming the GatedPixelCNN/PixelCNN++ autoregressive model concluding that the stochastic depth explains the gap between VAEs and autoregressive models. (Child 2020).

Another finding is that VAEs are able to learn hierarchical representation of the data as the authors show that the latent variables at low-resolution explain almost the whole global structure of the data while being only less than 1% of all the latent variables. The authors argue that these hierarchical representations of the data are also efficient as higher-resolution latent variables look like spatially independent on the resulting images. This means that the latent variables at higher resolution can be emitted in parallel instead of conditioning them on each other and this way making the sampling process much faster. (Child 2020).

5 Development

In this chapter the development process of the SR-VDVAE is described. Each section reports on the different parts of the model, and their additions to the task at hand. We base our image super-resolution model on the Conditional Variational Autoencoder introduced in Section 4.3 and Very Deep Variational Autoencoder, Section 4.4.

5.1 Concept

We base our image super-resolution model on the Conditional Variational Autoencoder introduced in Section 4.3 and Very Deep Variational Autoencoder, Section 4.4. The goal is to generate a high-resolution image x, given its low-resolution counterpart y referring to maximizing of the conditional probability as stated in Equation 4.3.

5.2 Evidence lower bound

The evidence lower bound used for the image super-resolution can be expressed as the ELBO for conditional variational autoencoder, but with slight modifications. Stating the ELBO for CVAE here again as in Equation 4.5:

$$ELBO = E_{q_\phi(z|x,y)}\left[logp_\theta(x|z,y)\right] - D_{KL}(q_\phi(z|x,y)||p_\theta(z|y))$$

The approximate posterior $q_\phi(z|x,y)$ is conditionally dependent on both the high and low resolution images. However, as the low-resolution image information is contained inside the high-resolution one, we can omit conditioning the posterior on y. In the above equation, the prior $p_\theta(z)$ is conditioned on the low resolution image y, resulting in $p_\theta(z|y)$.

Based on these changes, the ELBO can be expressed as:

$$ELBO = E_{q_\phi(z|x)}\left[logp_\theta(x|z,y)\right] - D_{KL}(q_\phi(z|x)||p_\theta(z|y)) \tag{5.1}$$

The conditioned prior $p_\theta(z|y)$ can be expressed as a neural network that maps the low resolution image to a distribution over the latent variables z. In the next few sections, we experiment with a simple linear model as well as a more complex one, adopting the bottom-up path architecture from the VDVAE model and refer to it as $LR\,encoder$ because it is based on the inference network of the VDVAE. To make it more clear, we refer to the VDVAE original model encoder and decoder as the HR encoder and HR decoder, sometimes omitting the HR in front of the decoder as we don't have an LR decoder component. Fig. 5.4 shows the architecture of one of the VDVAE-SR model configurations we are using, which should make this a bit more clear.

5.3 Transfer learning

The VDVAE models, as described in 4.4, were trained for 2.5 week on 32 V100 GPUs in order to get the presented results. As a similar set-up was out of our computational capabilities, one of the main objective during the project was to be able to use a pre-trained VDVAE model in our architecture to drastically decrease the training time needed for the development of our model.

In the next subsection, the concept of using a pre-trained generative model with an LR Encoder for low resolution images is proved and it will serve as the first step in the development process.

For all the models that we train, we use the pre-trained HR encoder of the VDVAE model without updating its weights as we observed that the posterior samples are perfect quality - same as the original images.

5.4 Trivial

To serve as a proof-of-concept and to find out if the transfer learning is correctly implemented, the first model created was the simplest way to introduce an LR encoder, which we call a Trivial version. This model consists of a VDVAE with an additional LR encoder with only one convolutional layer that serves just as a linear transformation to the desired dimension (latent representation at the top of the network).

The equation for the conditioned prior using this method can be expressed as:

$$p_\theta(z|y) = p_\theta(z_0|y) \prod_{i=1}^{K} p_\theta(z_i|z_{<i}, y)$$

where $p_\theta(z_0|y) = N(\mu_{Linear}(y), \sigma^2_{Linear}(y))$

We decided to train two models based on this configuration. Taking advantage of the pre-trained model of the VDVAE, the difference between the two models is that the first one's decoder is trainable, while the second one is fixed, where on both of them the decoders are initialized with the pre-trained weights.

5.4.1 Trainable Decoder

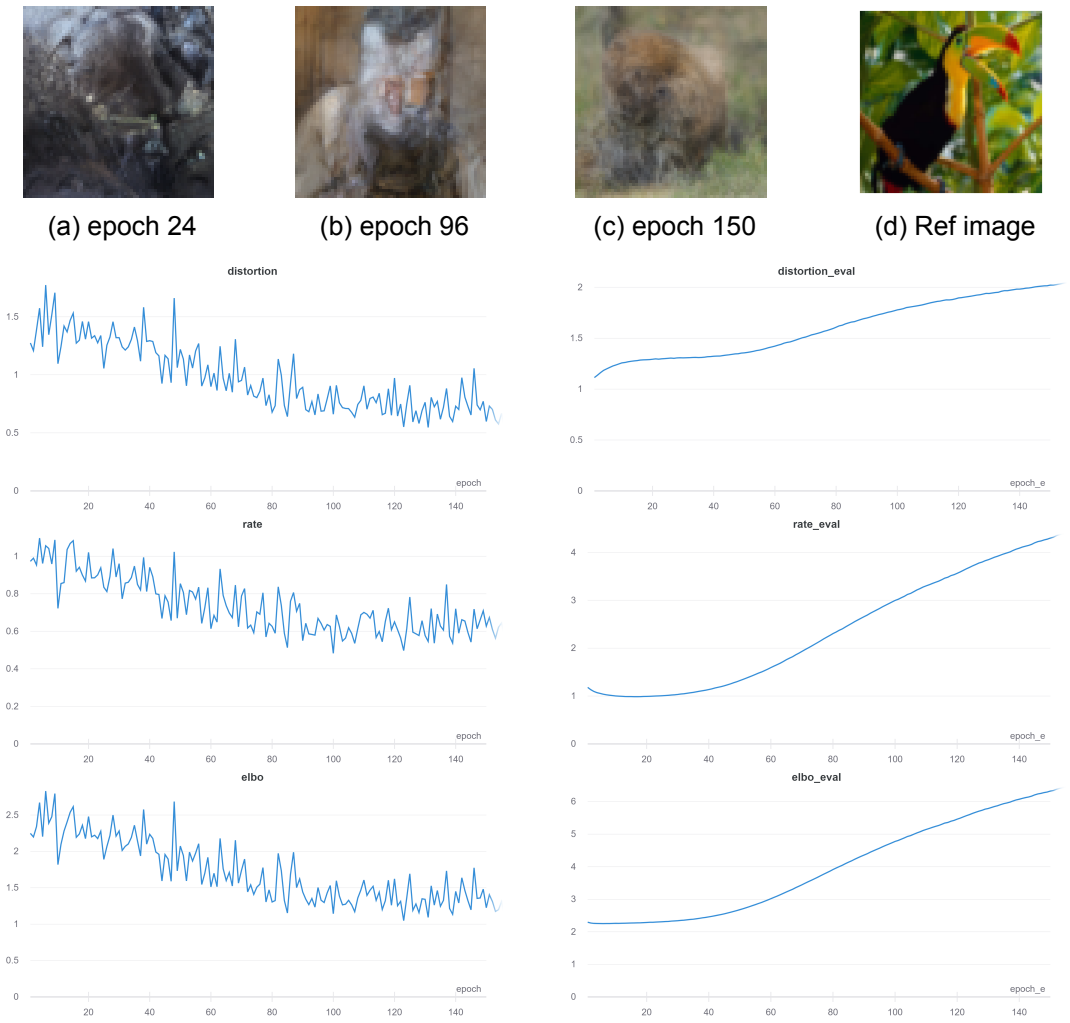
The results of the model with trainable decoder are shown on fig. 5.1 where the test samples are conditioned on the low-resolution picture of the bird from Set5. The idea is to show the visual improvement through the different phases of training. The "distortion" graph represents the reconstruction error of the ELBO, while the "rate" represents the KL divergence. The KL divergence presented on each of the following graphs is calculated as the average over all the layers. We also show the evaluation metrics on the validation dataset through training.

5.4.2 Fixed decoder

Another variation of the Trivial version was to see how the training performs when the decoder is not trainable (not updating the weight parameters). The produced test samples are shown on fig. 5.2. The metrics are quite similar except with the KL divergence where we don't see that decrease as in the previous model. Based on the quality of the images, these are a bit sharper compared to the one with trainable decoder but still far off the desired result.

5.4.3 Posterior Collapse

Based on the KL divergence metric and visual inspection of test samples, we believe that the averaging out of the test samples for the version where the decoder is trainable could be because of posterior collapse, a problem that often occurs in VAEs earliest described in Bowman et al. 2015. This problem can happen when the decoder is expressive enough and learns to ignore some or all of the latent variables resulting in the encoder to collapse into the prior: $q_\phi(z|x) \approx p(z)$. An example of pictures from a model with a heavily collapsed posterior can be seen in fig. 5.3.



(a) epoch 24 (b) epoch 96 (c) epoch 150 (d) Ref image

Figure 5.1: Picture (a), (b), and (c) are test samples taken during the training process, while picture (d) is the reference image. The plots represent the tracked parameters during training the Trivial model with trainable decoder. These include the distortion, rate (kl-divergence), and the elbo for both training and evaluation

(a) Sample1 (b) Reconstruction 1 (c) Sample 2 (d) Reconstruction 2

Figure 5.3: Example of images from a model with collapsed posterior.

5.5 VDVAE LR encoder architecture

The next step in the development process is to increase the complexity of the LR encoder as we concluded that the Trivial version is not expressive enough to produce latent variables from which we can generate good samples for super-resolution. Based on the VDVAE architecture, we create a smaller version of the HR encoder similar to the pre-

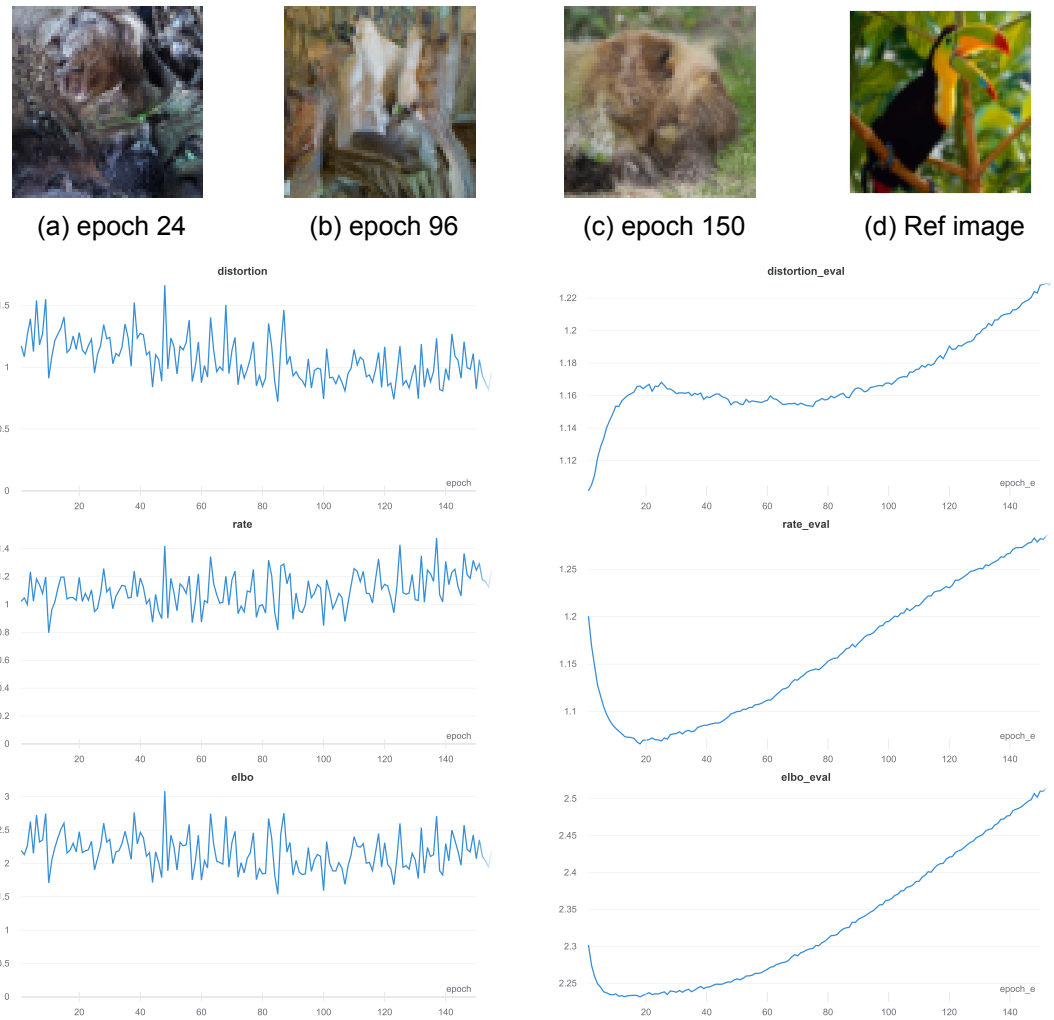


Figure 5.2: Picture (a), (b), and (c) are test samples taken during the training process, while picture (d) is the reference image. The plots represent the tracked parameters during training the Trivial model with fixed decoder. These include the distortion, rate (kl-divergence), and the elbo for both training and evaluation.

trained one, but without the bottom most layers that represent the 64 and 32 dimensions. A low-resolution 16x16 image as shown on fig. 5.4 is used as the input to the new LR encoder on the right.

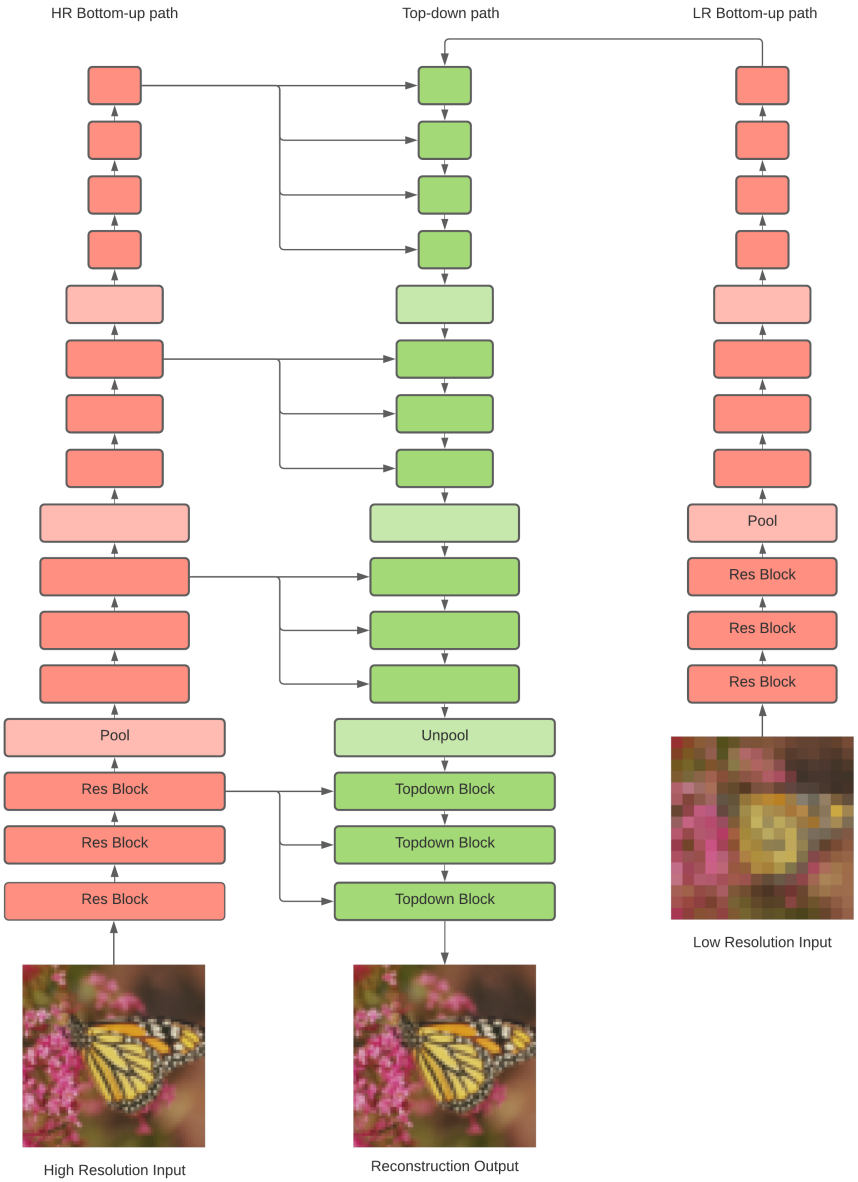


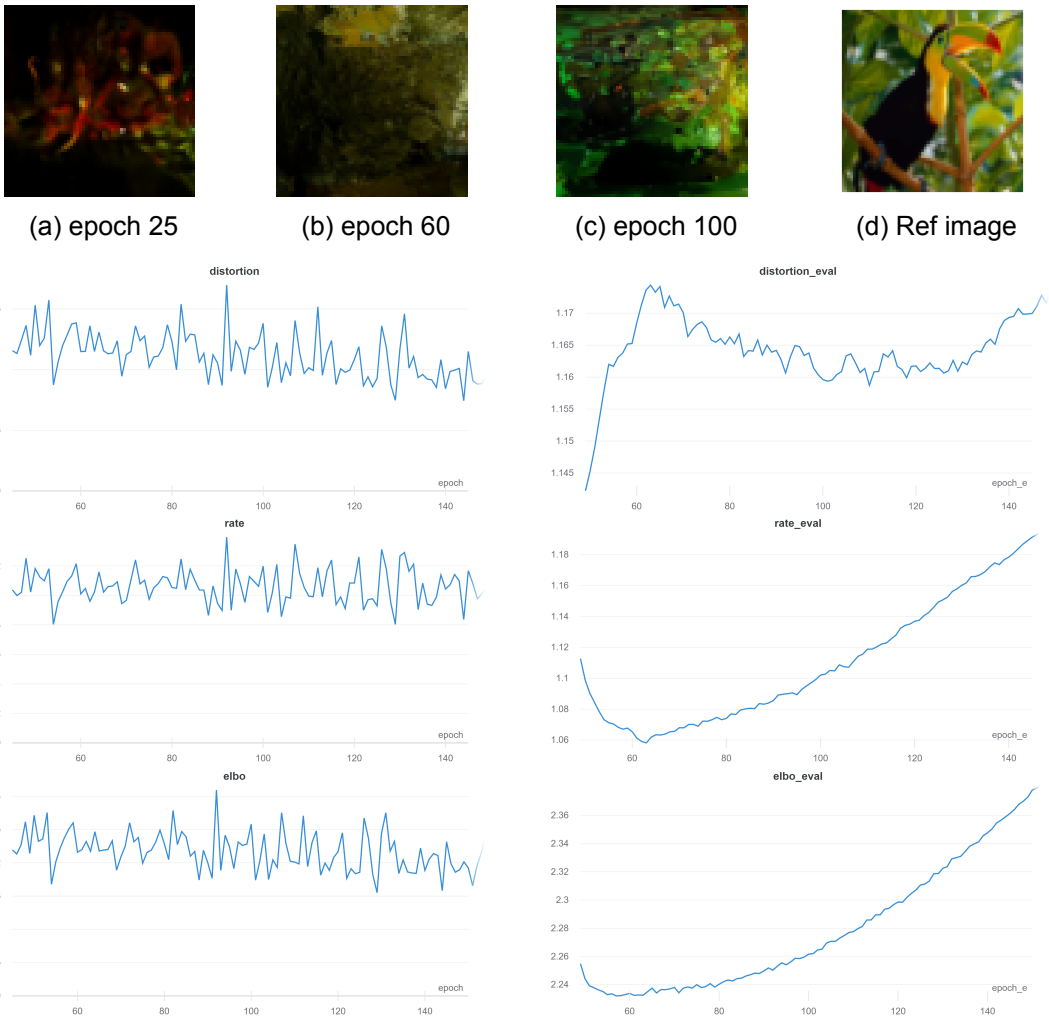
Figure 5.4: VDVAE-SR architecture

The prior equation is similar to the Trivial version with the difference that z_0 is sampled from a Gaussian distribution with mean and variance outputs of the $LR_{encoder}$:

$$p_\theta(z_0|y) = N(z_0|\mu_{LR_{enc}}(y), \sigma^2_{LR_{enc}}(y))$$

We again experiment with two versions - one with a trainable and the other with fixed decoder to find out differences in training and visual results. The results from the model with trainable decoder are shown on fig. 5.6 where the metrics of a fixed decoder are shown on fig. 5.5.
Based on the shown results, we can conclude that the global features are better captured than the Trivial models as we see black, yellow and green colors dominating the test samples that should refer to the main colours of the toucan image.

However, once again, the individual shapes on the image are not produced when sampling. Again, the KL divergence of the model with trainable decoder follows a similar pattern as the Trivial model but we don't observe samples that seem to average out. This is a good sign as we hypothesize that having a trainable decoder is more useful when we pass in more information from the LR encoder (not only on the top layer) as modified in the next section.



(a) epoch 25 (b) epoch 60 (c) epoch 100 (d) Ref image

Figure 5.5: Picture (a), (b), and (c) are test samples taken during the training process, while picture (d) is the reference image. The plots represent the tracked parameters during training the VDVAE-SR model with fixed decoder. These include the distortion, rate (kl-divergence), and the elbo for both training and evaluation

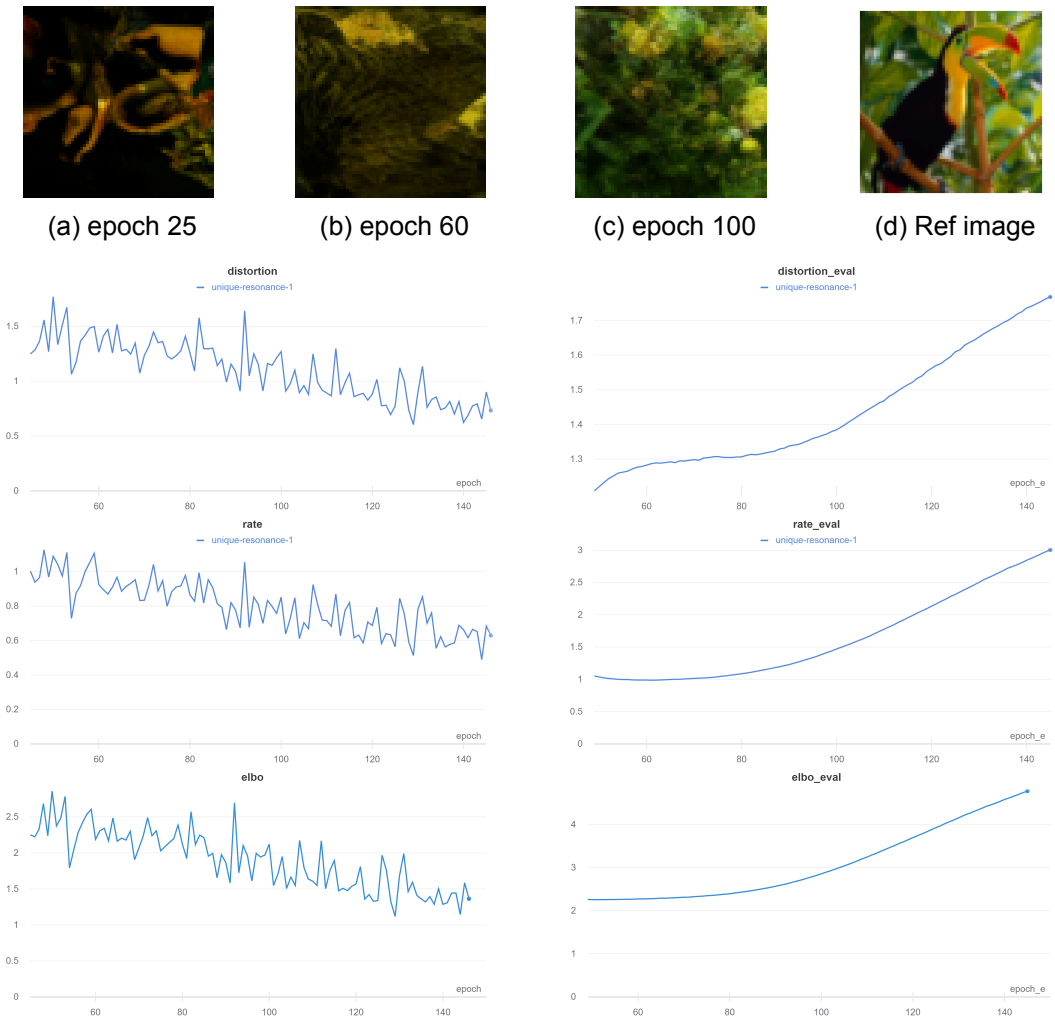


Figure 5.6: Picture (a), (b), and (c) are test samples taken during the training process, while picture (d) is the reference image. The plots represent the tracked parameters during training the VDVAE-SR model with trainable decoder. These include the distortion, rate (kl-divergence), and the elbo for both training and evaluation

5.6 LR-activations in posterior

As we observed in the previous section, there was an improvement in capturing low-level details, however, the model still lacked capturing high-level ones. We hypothesize, that we need to add more information to the decoder on all layers, not only on the top one, in order to have better results.
Based on this argument, we made modifications in order to add the activations from the LR encoder with the activations from the HR encoder before they are concatenated with the input at each topdown block. This means that we are adding new information from the LR encoder on each layer only to the posterior in the top-down block. We still feed the activations of the top layer of the LR encoder to the top layer of the decoder as the previous model. The diagram of how the activations are added is shown on fig. 5.7.

Based on this architecture, we rely on the KL divergence term to make the prior distributions on each topdown block similar to the distributions of the approximate posterior, which now consists also of activations from the LR encoder. The results of this model are presented on fig. 5.8 and it can be seen that similarly to the model without added activa-

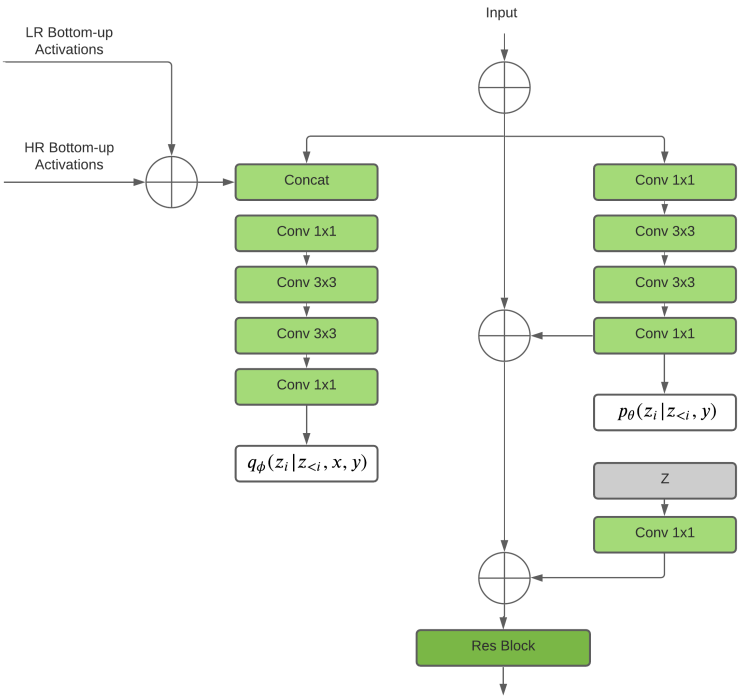


Figure 5.7: Top-down block adding activations in posterior

tions on each layer, this one is also not able to capture high-level features properly. Nevertheless, we learn that passing the extra information from the LR encoder only in the posterior is not enough,reflected also in eq. (5.1), as especially during generative mode, the model is not able to use the new information, which leads us to the next and final model explained in the next section.

5.7 Final Model

5.7.1 Architecture

As seen in the last iteration, the model was still not able to capture enough information besides global features. For this reason, instead of passing information from the LR encoder only at the top of the top-down path and in the posterior part of the decoder, we modified it so we add activations directly to the input on each top-down block. This can be seen in fig. 5.9. Using the LR activations in this manner, the new information will have an impact on both the prior and the posterior sampling.

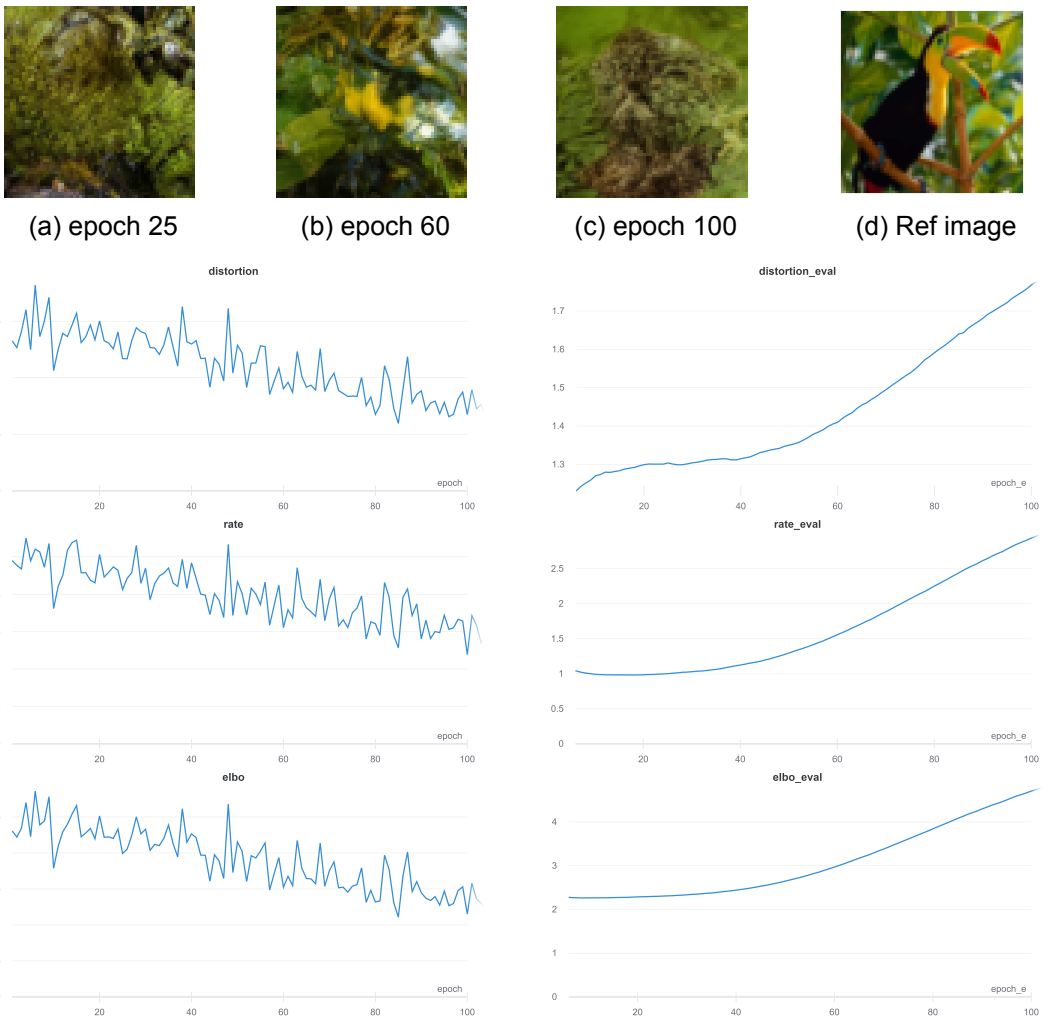


Figure 5.8: Picture (a), (b), and (c) are test samples taken during the training process, while picture (d) is the reference image. The plots represent the tracked parameters during training the VDVAE-SR model with activations in the posterior. These include the distortion, rate (kl-divergence), and the elbo for both training and evaluation

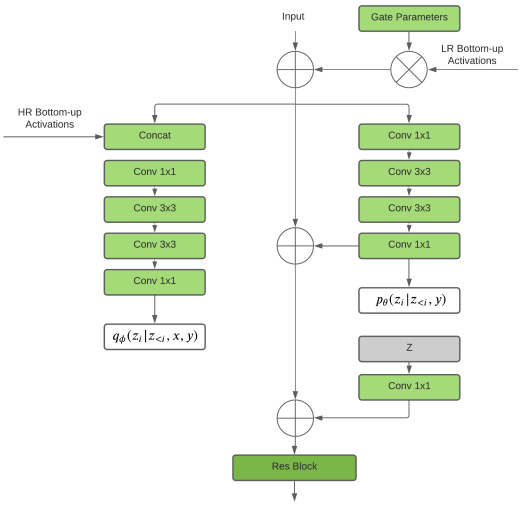


Figure 5.9: Top-down block adding activations to both prior and posterior

The conditioned prior is expressed again as:

$$p_\theta(z|y) = p_\theta(z_0|y) \prod_{i=1}^{K} p_\theta(z_i|z_{<i}, y)$$

where $p_\theta(z_0|y) = N(\mu_{LR_{enc}}(y), \sigma^2_{LR_{enc}(y)})$

The posterior can be expressed as:

$$q_\phi(z|x, y) = q_\phi(z_0|x, y) \prod_{i=1}^{K} q_\phi(z_i|z_{<i}, x, y)$$

5.7.2 Gate parameters

Another addition to the model are the gate parameters seen in fig. 5.9, similar to the concept introduced by Bachlechner et al. 2020. Using another trainable parameter like that, initialized with zeros, will gradually introduce the new information to the pre-trained model.
During the development of the architecture it was observed that the model became really hard to train and having really big gradients if the new information was introduced in an uncontrolled manner. Having the gate parameters greatly improved the stability, and in some case, made the models trainable where it was previously not able to train, exploding the gradient to infinity and receiving NaN values in the calculated ELBO.

5.7.3 Results

The results of the training can be seen on fig. 5.10, where now high-level details as well as global features are captured by the model. Adding together all the presented methods, end up in a model that is now able to more accurately sample a picture closer to a high resolution version of the input LR image, having a higher degree of fidelity in both color and shape estimation.
This can be seen even after only 200 epochs of training, on a smaller dataset (DIV2K). More examples of images, alongside metrics and comparison between multiple models can be seen in the next chapter, chapter 6.

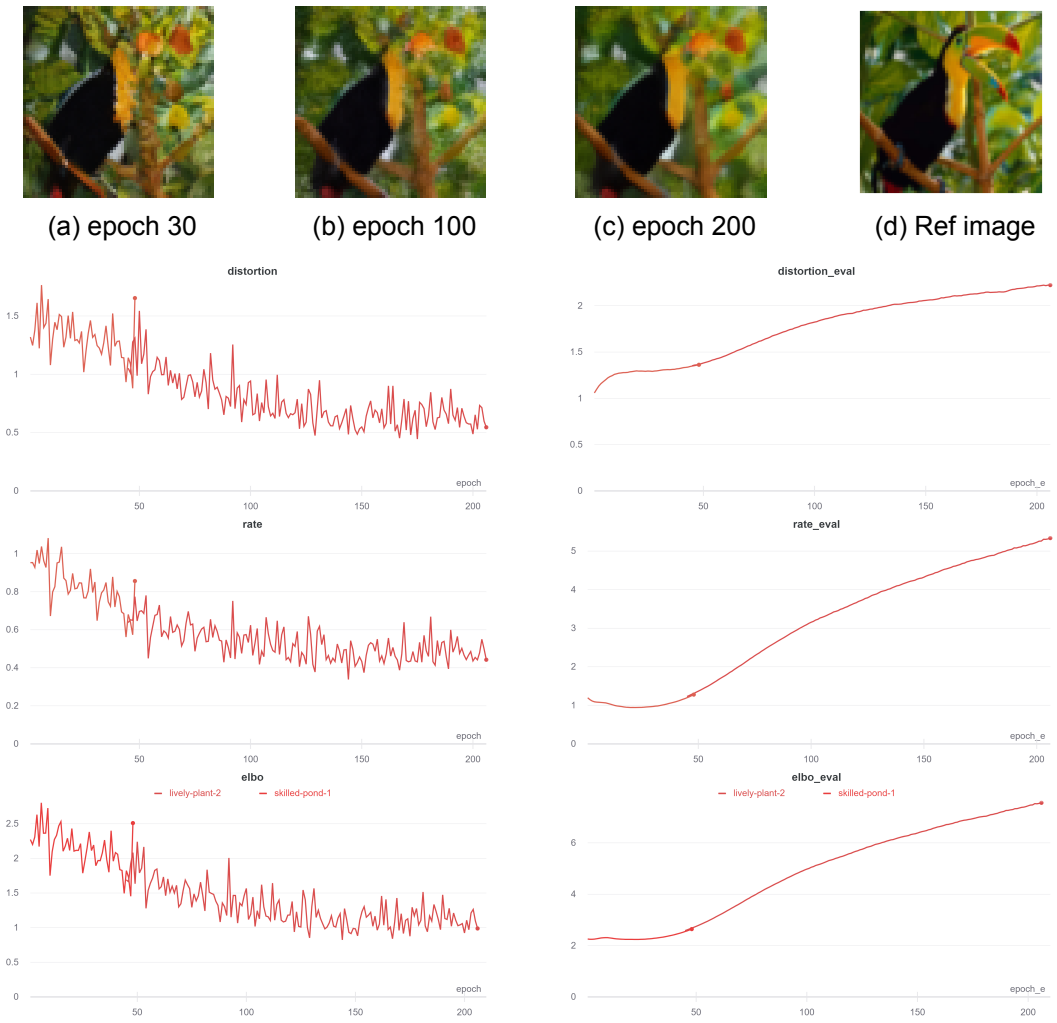


Figure 5.10: Picture (a), (b), and (c) are test samples taken during the training process, while picture (d) is the reference image. The plots represent the tracked parameters during training the VDVAE-SR final model architecture. These include the distortion, rate (kl-divergence), and the elbo for both training and evaluation

5.7.4 Overfitting

As observed on all the trained models from the evaluation graphs, the models seem to overfit on the training data as we see an increase in the validation loss. For the final model, this can be seen after around 25 epochs according to the elbo on fig. 5.10. However, we decided to continue training as we were still able to see improvement on the samples from the validation dataset as as it can also be seen on the bird from Set14 on fig. 5.10. A reason for this could be that the likelihood that we are trying to maximize and the perceptual quality are not necessarily correlated.

5.7.5 Comparison table

We show the differences between the models on section 5.7.5 to be more clear on what exactly configuration has been used on each model.

Model/ Configuration	$p_\theta(z_0\|y)$ (activations passed on top of the decoder)	Activations passed in the posterior on each layer	Activations passed in the prior on each layer	Gate parameters
Trivial	Yes	No	No	No
LR encoder	Yes	No	No	No
Activations in posterior	Yes	Yes	No	No
Final model	Yes	Yes	Yes	Yes

Table 5.1: Comparison between models based on the different configurations used

6 Results

In this section the final models for 64x64 images and 256x256 images, with all the modification presented in chapter 5 are presented. The results, training methods, and comparisons with other state-of-the-art models are shown in this chapter.

6.0.1 Training

The training of the two models are based on the final model architecture as described in section 5.7, which includes passing the activations from the LR encoder to the decoder on each layer as well as having the gate parameters for increased stability during training. The training hyperparameters for both models are kept the same as the ones from the VDVAE-model shown on fig. 6.1

Parameter	CIFAR-10	ImageNet-32	ImageNet-64	FFHQ-256	FFHQ-1024
Num layers	45	78	75	62	72
Hidden size	384	512	512	512	Varies
Bottleneck size	96	128	128	128	Varies
Latent dim per layer	16	16	16	16	16
Batch size	32	256	128	32	32
Learning rate	0.0002	0.00015	0.00015	0.00015	0.00007
Optimizer	Adam	Adam	Adam	Adam	Adam
Skip threshold	400	300	380	180	500
Weight Decay	0.01	0.0	0.0	0.0	0.0
EMA rate	0.0002	0.00015	0.00015	0.00015	0.00015
Training iterations	1.1M	1.7M	1.6M	1.7M	1.7M
GPUs	2 x V100	32 x V100	32 x V100	32 x V100	32 x V100
Training time	6 days	2.5 weeks	2.5 weeks	2.5 weeks	2.5 weeks
Parameters	39M	119M	125M	115M	115M

Figure 6.1: Figure of hyperparameters used during training. The table was taken from Child 2020.

64-resolution model

The 64-resolution model has been trained on the DIV2K dataset, having an LR-encoder trained from scratch and an HR encoder and decoder pre-trained on the ImageNet64 dataset, where the HR encoder's weight parameters are not updated during training, but the decoder's parameters are being updated. The stochastic depth of the used pre-trained VDVAE model is 75 layers.
The training has been performed for 200 epochs using a batch size of 4. Each epoch consists of 200 iterations on the training set (800 images) and 50 iterations on the validations set (200 images).

256-resolution model

Similar to the 64-resolution model, we are utilizing the pre-trained versions provided by VDVAE for the 256-resolution model as well, having the same weights update procedure in both encoders and decoder. The only difference this time is that we initialize the weights of the LR-encoder with the pre-trained model on ImageNet64 and the HR decoder and encoder are pre-trained on the FFHQ 256x256 dataset. The stochastic depth of the used pre-trained VDVAE model is 62 layers. The training has been performed for 95 epochs as training time takes much longer than the 64-resolution model. We use a batch size of 1, similar to the VDVAE model and so an epoch consists of 800 iterations for training and 200 iterations for validation. Samples conditioned on the low-resolution bird image from Set5 during training are shown on fig. 6.2

(a) epoch 5 (b) epoch 60 (c) epoch 95 (d) Ref image

Figure 6.2: Test samples throughout training for our model 256 resolution

6.0.2 Temperature

The ”temperature” t parameter between 0 and 1 is used in the VDVAE during sampling from prior in generative mode where by default is equal to 1 in the equation:

$$pv = pv + I \cdot log(t)$$

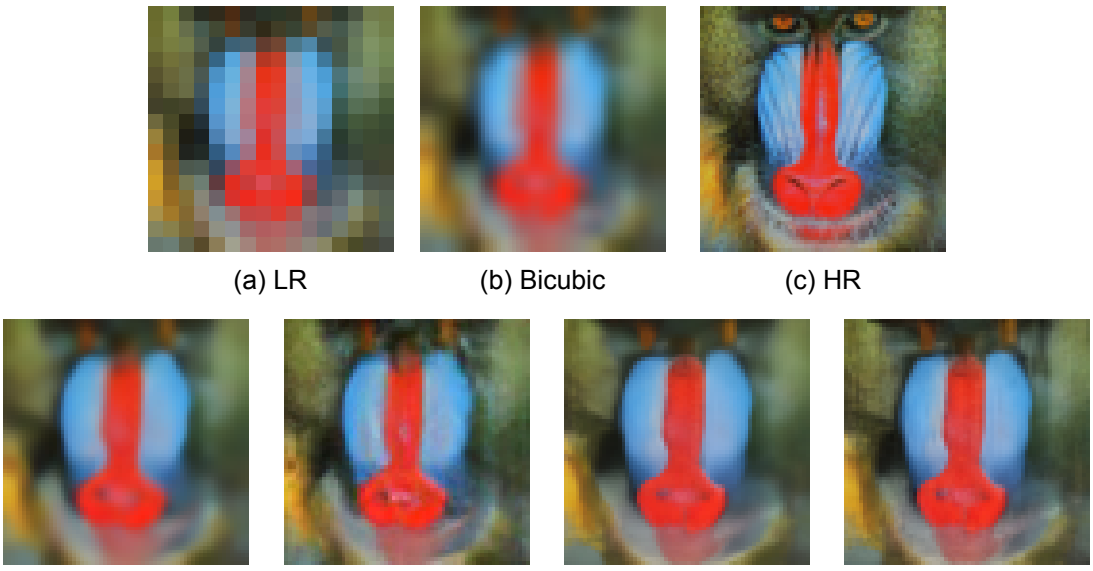
where pv is the variance and I is the identity matrix having the same dimension as pv. Reducing the temperature results in reducing the variance and so achieving more regularity in the generated samples. fig. 6.3 and fig. 6.4 show samples with different temperature for the 64 and 256 model respectively.

Figure 6.3: Prior sampling difference with varying temperature values for 64x64 pictures

Figure 6.4: Prior sampling difference with varying temperature values for 256x256 pictures

6.1 Model VDVAE-SR 64

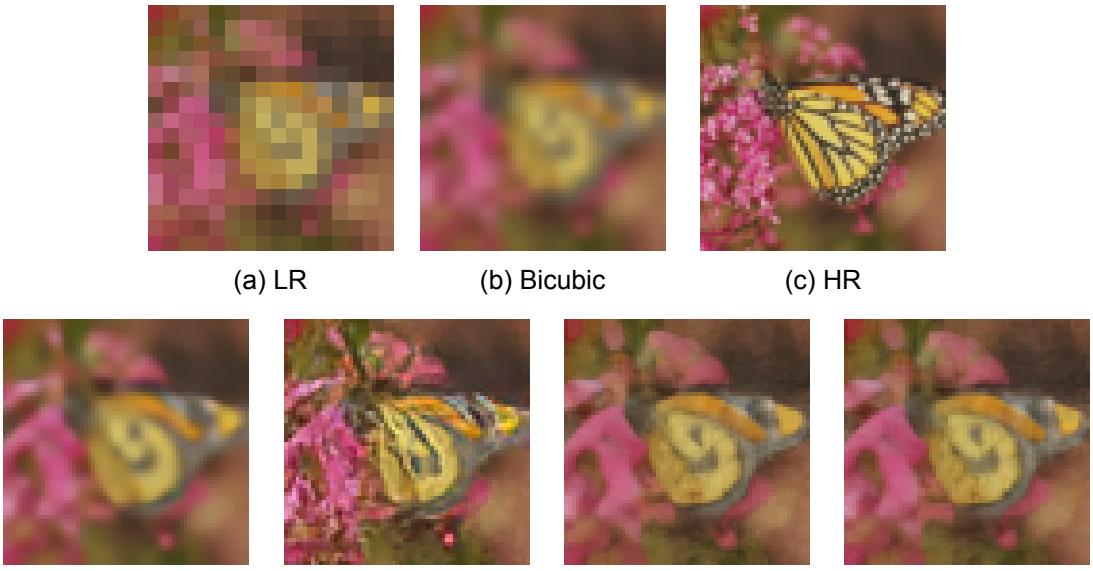
This section presents the results for the 64-resolution model showing test samples from our model with temperatures 0.2 and 0.8 compared to samples generated from the two other models that we use for comparison, namely the EDSR and ESRGAN. Additionally, we also include the images generated by bicubic interpolation. The baboon and monarch images from Set14 are used for comparison and shown on fig. 6.5 and fig. 6.6.



(a) LR (b) Bicubic (c) HR

(d) EDSR (e) ESRGAN (f) VDVAE-SR(T=0.8) (g) VDVAE-SR(T=0.2)

Figure 6.5: Prior Samples comparison between multiple models for 16x16 to 64x64 on the baboon picture of the Set14 DataSet



(a) LR (b) Bicubic (c) HR

(d) EDSR (e) ESRGAN (f) VDVAE-SR(T=0.8) (g) VDVAE-SR(T=0.2)

Figure 6.6: Prior Samples comparison between multiple models for 16x16 to 64x64 on the monarch picture of the Set14 DataSet

6.1.1 Quantitative results

In table 6.1, we compare the methods using the PSNR and SSIM scores previously explained in section 3.4 on each test dataset using our experimental protocol. Based on the calculated scores on the 64-resolution model, we see that EDSR performs best on both PSNR and SSIM, while our model is second where there is a very small difference between the temperatures 0.2 and 0.8. It is hard to say the results are justified comparing them visually to the images on fig. 6.5 and fig. 6.6 as the low-resolution one consists of small number of pixels only, so it is definitely difficult for all the models to output samples close to the high-resolution image. However it can be seen that the EDSR image seems a bit more blurry than ESRGAN and our model even though it achieves the best scores.

Table 6.1: Table with the evaluation of the chosen model using PSNR and SSIM on the Y Channel.The results are the average score on every dataset. All the images were downscaled to 64x64 resolution, the low-resolution input image being a 4x downscaled image of that as well (16x16)

Model/ DataSet	Bicubic		EDSR		ESRGAN		VDVAE-SR(0.2)		VDVAE-SR(0.8)	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Set5	23.13	0.6621	26.13	0.8082	23.61	0.7283	24.90	0.7610	24.79	0.7600
Set14	23.43	0.6385	25.08	0.7484	22.63	0.6454	24.44	0.7187	24.44	0.7168
T91	25.03	0.6722	27.07	0.7782	24.30	0.6671	26.04	0.7324	26.01	0.7291
General100	23.53	0.6730	25.97	0.7884	23.31	0.6890	25.10	0.7522	25.08	0.7506
Urban100	22.36	0.5428	23.96	0.6701	21.68	0.5837	23.37	0.6344	23.35	0.6338
BSD100	25.10	0.6741	26.87	0.7679	23.81	0.6472	26.00	0.7294	25.95	0.7265
Manga109	20.84	0.5840	22.39	0.7081	19.74	0.5966	21.93	0.6859	21.89	0.6833

6.2 Model VDVAE-SR 256

For the 256-resolution model, we show the produced super-resolution images from all models based on two random images from the BSD100 dataset on fig. 6.7 and fig. 6.8.

(a) LR (b) Bicubic (c) HR

(d) EDSR (e) ESRGAN (f) VDVAE-SR(T=0.8) (g) VDVAE-SR(T=0.2)

Figure 6.7: Prior Samples comparison between multiple models for 64x64 to 256x256 on a test picture from BSD100

(a) LR (b) Bicubic (c) HR

(d) EDSR (e) ESRGAN (f) VDVAE-SR(T=0.8) (g) VDVAE-SR(T=0.2)

Figure 6.8: Prior Samples comparison between multiple models for 64x64 to 256x256 on a test picture from BSD100

6.2.1 Quantitative results

We calculate PSNR and SSIM scores for the 256-resolution model the same way as the 64-resolution one and present the results for all the models and test datasets on table 6.2. Based on the calculated scores, once again, EDSR shows best results on all datasets. VDVAE-SR is again on second place but this time we see a bit bigger difference between the 0.2 and 0.8 temperatures, where the model with more regularity is better.

Table 6.2: Table with the evaluation of the chosen model using PSNR and SSIM on the Y Channel.The results are the average score on every dataset. All the images were downscaled to 256x256 resolution, the low-resolution input image being a 4x downscaled image of that as well (64x64)

Model/ DataSet	Bicubic		EDSR		ESRGAN		VDVAE-SR(0.2)		VDVAE-SR(0.8)	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Set5	28.88	0.8185	32.98	0.9025	30.92	0.8523	31.68	0.8850	31.06	0.8732
Set14	26.09	0.7261	28.75	0.8166	26.57	0.7330	28.04	0.7993	27.44	0.7787
T91	31.52	0.8452	35.32	0.9112	33.37	0.8606	34.02	0.8917	33.13	0.8731
General100	28.41	0.7957	32.14	0.8741	29.95	0.8128	31.15	0.8570	30.36	0.8382
Urban100	23.12	0.6229	25.58	0.7608	22.90	0.6704	24.96	0.7370	24.51	0.7188
BSD100	27.00	0.7101	28.94	0.7846	26.41	0.6921	28.24	0.7633	27.54	0.7370
Manga109	22.72	0.6867	25.78	0.8253	23.86	0.7645	25.22	0.8178	24.80	0.8029

The problem with PSNR and SSIM scores has been previously pointed out in several works including SRGAN and ESRGAN as previously mentioned in chapter 2 where SRGAN has even created their own score called MOS using people that manually evaluate test images giving them a score from 1 to 5 showing that the PSNR and SSIM scores actually differ from human visual perception especially when new high-level details are generated.

This can be observed on the test samples previously shown on fig. 6.7 and fig. 6.8 that EDSR lacks high-level details compared to ESRGAN and VDVAE models but it is still

favourite in the PSNR and SSIM scores. Because of this, we use the FID score metric described in section 3.4 using the implementation on PyTorch by Seitzer 2020. We show the results for the 256-resolution model on table 6.3 where lower numbers indicate better results. This time, the ESRGAN tops the table and EDSR is on third place. We also show results produced with different temperatures by the VDVAE-SR model, where this time the 0.8 temperature has better score than the 0.2 temperature samples. This can also be justified visually on the fig. 6.7 and fig. 6.8 when zooming in, observing that there are more high-level details synthesized by the 0.8 temperature images. The samples with highest variance introduce more high-level details as previously shown on fig. 6.4 and this is correctly reflected in the scores as well.

Table 6.3: FID metric Table

Model/ DataSet	Bicubic	EDSR	ESRGAN	VDVAE-SR(0.2)	VDVAE-SR(0.8)	VDVAE-SR(1)
T91	92.44	61.23	39.00	61.73	59.13	68.03
General100	113.68	68.99	42.52	67.22	63.78	70.72
Urban100	165.34	109.78	66.15	96.28	95.61	102.55
BSD100	146.36	124.45	66.87	95.89	82.27	85.75
Manga109	171.76	80.22	58.11	74.25	76.93	82.43

More output images from the different models with 256-resolution on test dataset Set14 can be seen in the appendix A.

7 Final Remarks

In this chapter we will summarize the project, discuss it's strengths and shortcomings, while also concluding the report. In the end some future works will be proposed and other aspects that could be improved or worked upon.

7.1 Discussion

7.1.1 Impact of images cropping on metrics

The final results we obtain from our experimental protocol, when comparing the different models based on the quality scores, are based on cropped images, so we can fit the appropriate image dimension in our model resulting in the same height and width for the output. However, this approach differs from other works like EDSR and ESRGAN where it was possible to input any image resolution to their models resulting in not a square image output as ours. This could be one of the reasons why we see difference between the scores in the Results section and table 2.1 for the same models and datasets, where table 2.1 was calculated without cropping the images. The other difference is the fact that we produce output images that are smaller in dimension compared to the original test datasets, while the other models calculate the scores with output samples that are the same or with only a few pixels difference of the original images.

7.1.2 Adding activations strictly in prior (gradient explosion)

Something that we also need to comment on is the fact that for the final model we feed the activations from the LR encoder to both the posterior and the prior in the top-down path, which is different from what we are based on in eq. (4.5) as according to it, the LR activations in the posterior could be omitted. We actually tried to train a model where we feed the activations from the LR encoder to the prior only, but the training was not stable resulting in many NaN values for the ELBO. This is why we had to go with the configuration explained in the final model, but future work might be able to achieve stable training possibly with the right hyperparameters.

7.1.3 Comparison against other VAE-based SR methods

Another model that could have been informative to compare with would be a VAE-based super-resolution model. However, there are only very few papers that we were able to find and the only one that had an implementation on GitHub, which we discussed in chapter 2 as srVAE by Gatopoulos, Stol, and Tomczak 2020, outputs images only up to 32x32 dimension using x2 upscaling instead of x4, making it unfeasible to compare with our model. It is important to mention that we also had an autoregressive model in our experimental protocol that we had to remove in the end as we found out later that the super-resolution task is executed in a manner that is different than single image super-resolution making it not comparable with the other models in our experimental protocol.

7.1.4 Training challenges

Another aspect of the project that could be discussed is the training and training environment used. As the network was really big (180 million parameters) even by using pre-trained parts of the network the training time needed was quite long. The pre-trained models used were trained for 2.5 weeks on 32 V100 GPUs. This is an amount of computing power that we did not have access to, but by using the pre-trained models in our development this time was cut to a time frame between 24-72 hours for most of our models, training on either a P100 or a V100 GPU, as those were the available GPUs in the

development platform that we chose (Google Colab). A problem that we had was when training the 256-resolution model as training was very slow and we had to wait and be lucky to get a v100 GPU from Google Colab to be able to train faster.

7.2 Conclusion

Through this thesis, we investigated the use of Deep Variational Autoencoders for the purpose of generating Super-Resolution images. Even though more research and development can be conducted in the future (presented in the next section), based on the results presented in chapter 6, we can conclude that the project and its results are satisfying as they are comparable to models that are both winners on super-resolution challenges from years 2017 and 2018 that still hold high ranking on official leaderboards. Based on our findings,it can be feasible to say that VAE-based models can achieve even greater results in the future of super-resolution.

7.3 Future Work

7.3.1 Overfitting and use of larger datasets

Most of the models shown during the report, even though PSNR and SSIM still increase over time, show clear signs of overfitting. This is mainly because of the choice of training datasets, specifically DIV2K dataset which only consists of 800 training pictures. This was done for the reason of being able to compare more directly with other models trained similarly, and because of the time constraints that we had. In the future, a good experiment could be to train on a much larger dataset, as ImageNet, to be able to see data and results from a longer training time. This is actually an experiment that was attempted by us as well, but it was not included in the report as we were not able to train for more than 24 hours at the time, and splitting training sessions with such a large dataset would have been quite difficult.

7.3.2 Training iterations

One aspect of training that we failed to do was to properly track quality measure scores as well besides loss metrics. By doing this a better view of the actual results could be seen. Although this was not implemented, by analysing saved models along a training session, it can be observed that the PSNR and SSIM scores continue to increase, even though the tracked metrics indicate that the network is slowly overfitting on the dataset. This can indicate that training more could in fact give better results, even using the same small dataset.

7.3.3 Probabilistic Distributions

Another improvement that can be made, even suggested by Child 2020, is the use of more flexible distributions, like discretized mixture of logits. Besides in the output sample where the discretized mixture of logits is used, Gaussian distributions are implemented at every block. This is a limitation that the original developers of VDVAE did take in account but did not implemented as the main study was about hypothesizing depth, not elementwise density, as more important for their specific task. By going around the restrictions that a Gaussian distribution holds upon the expressiveness of the network, better super-resolution samples could be hypothesized.

7.3.4 Hyperparameter search

Another aspect that can be improved is the chosen hyperparameters. Doing a hyperparameter search, and a more in-depth analysis of them could be very beneficial for such a deep network, as during our development phase we mainly used the hyperparameters given by the VDVAE developers in Child 2020, but as the task is different, this factor can be improved.

7.3.5 Layer Architecture and Distribution

The actual architecture of both the top-down network and bottom-up can be also changed. The configuration and distribution of layers at each resolution can greatly impact the performance of the network. A similar analysis for this aspect can also be seen in Child 2020, and experimenting with multiple layers for the higher resolution blocks, as 256x256, could have some success, as these blocks are the only ones which do not have any extra information or activations from the LR top-down encoder.

Bibliography

Child, Rewon (2020). “Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images”. In: *arXiv preprint arXiv:2011.10650*.

Kingma, Diederik P and Max Welling (2013). “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114*.

Goodfellow, Ian J et al. (2014). “Generative adversarial networks”. In: *arXiv preprint arXiv:1406.2661*.

Oord, Aaron van den et al. (2016). “Conditional image generation with pixelcnn decoders”. In: *arXiv preprint arXiv:1606.05328*.

Dosovitskiy, Alexey and Thomas Brox (2016). “Generating images with perceptual similarity metrics based on deep networks”. In: *arXiv preprint arXiv:1602.02644*.

Maaløe, Lars et al. (2019). “Biva: A very deep hierarchy of latent variables for generative modeling”. In: *arXiv preprint arXiv:1902.02102*.

Sønderby, Casper Kaae et al. (2016). “Ladder variational autoencoders”. In: *arXiv preprint arXiv:1602.02282*.

Vahdat, Arash and Jan Kautz (2020). “Nvae: A deep hierarchical variational autoencoder”. In: *arXiv preprint arXiv:2007.03898*.

Chollet, François (2017). “Xception: Deep learning with depthwise separable convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258.

Kingma, Diederik P, Tim Salimans, et al. (2016). “Improving variational inference with inverse autoregressive flow”. In: *arXiv preprint arXiv:1606.04934*.

He, Kaiming et al. (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Dong, Chao, Chen Change Loy, Kaiming He, et al. (2015). “Image super-resolution using deep convolutional networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.2, pp. 295–307.

Ledig, Christian et al. (2017). “Photo-realistic single image super-resolution using a generative adversarial network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690.

Tong, Tong et al. (2017). “Image super-resolution using dense skip connections”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 4799–4807.

Lim, Bee et al. (2017). “Enhanced deep residual networks for single image super-resolution”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 136–144.

Wang, Xintao et al. (2018). “Esrgan: Enhanced super-resolution generative adversarial networks”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.

Jolicoeur-Martineau, Alexia (2018). “The relativistic discriminator: a key element missing from standard GAN”. In: *arXiv preprint arXiv:1807.00734*.

Blau, Yochai et al. (2018). *PIRM challenge on perceptual image super-resolution (2018)*.

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].

Devlin, Jacob et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805 [cs.CL].

Radford, A. and Karthik Narasimhan (2018). “Improving Language Understanding by Generative Pre-Training”. In:

Liu, Yinhan et al. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv: 1907.11692 [cs.CL].

Raffel, Colin et al. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. arXiv: 1910.10683 [cs.LG].
Yang, Fuzhi et al. (2020). *Learning Texture Transformer Network for Image Super-Resolution*. arXiv: 2006.04139 [cs.CV].
Esser, Patrick, Robin Rombach, and Björn Ommer (2021). *Taming Transformers for High-Resolution Image Synthesis*. arXiv: 2012.09841 [cs.CV].
Dosovitskiy, Alexey, Lucas Beyer, et al. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv: 2010.11929 [cs.CV].
Gatopoulos, Ioannis, Maarten Stol, and Jakub M Tomczak (2020). "Super-resolution variational auto-encoders". In: *arXiv preprint arXiv:2006.05218*.
Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2016). "Density estimation using real nvp". In: *arXiv preprint arXiv:1605.08803*.
Huang, Gao et al. (2017). "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.
Salimans, Tim et al. (2017). "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications". In: *arXiv preprint arXiv:1701.05517*.
Harvey, William, Saeid Naderiparizi, and Frank Wood (2021). "Image Completion via Inference in Deep Generative Models". In: *arXiv preprint arXiv:2102.12037*.
Son, Sanghyun (2017). *EDSR-PyTorch*. https://github.com/sanghyun-son/EDSR-PyTorch.
Wang, Xintao (2018). *ESRGAN*. https://github.com/xinntao/ESRGAN.
Agustsson, Eirikur and Radu Timofte (July 2017). "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
Bevilacqua, Marco et al. (2012). "Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, pp. 135.1–135.10. ISBN: 1-901725-46-4. DOI: http://dx.doi.org/10.5244/C.26.135.
Zeyde, Roman, Michael Elad, and Matan Protter (June 2010). "On Single Image Scale-Up Using Sparse-Representations". In: vol. 6920, pp. 711–730. ISBN: 978-3-642-27412-1. DOI: 10.1007/978-3-642-27413-8_47.
BasicSR (Basic Super Restoration) (n.d.). URL: https://github.com/xinntao/BasicSR/wiki.
Dong, Chao, Chen Change Loy, and Xiaoou Tang (2016). *Accelerating the Super-Resolution Convolutional Neural Network*. arXiv: 1608.00367 [cs.CV].
Huang, Jia-Bin, Abhishek Singh, and Narendra Ahuja (June 2015). "Single Image Super-Resolution From Transformed Self-Exemplars". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Martin, D. et al. (2001). "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics". In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 2, 416–423 vol.2. DOI: 10.1109/ICCV.2001.937655.
Matsui, Yusuke et al. (Nov. 2016). "Sketch-based manga retrieval using manga109 dataset". In: *Multimedia Tools and Applications* 76.20, pp. 21811–21838. ISSN: 1573-7721. DOI: 10.1007/s11042-016-4020-z. URL: http://dx.doi.org/10.1007/s11042-016-4020-z.
Gavade, Anil and Prasana Sane (Oct. 2013). "SUPER RESOLUTION IMAGE RECONSTRUCTION BY USING BICUBIC INTERPOLATION". In: pp. 205–209. DOI: 10.13140/RG.2.1.4909.0401.
Horé, Alain and Djemel Ziou (2010). "Image Quality Metrics: PSNR vs. SSIM". In: *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.
Nilsson, Jim and Tomas Akenine-Möller (June 2020). "Understanding SSIM". In:

Wang, Z. et al. (Apr. 2004). “Image Quality Assessment: From Error Visibility to Structural Similarity”. In: *IEEE Transactions on Image Processing* 13.4, pp. 600–612. DOI: 10.1109/TIP.2003.819861.

Heusel, Martin et al. (2018). *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. arXiv: 1706.08500 [cs.LG].

Pisharoty, Narayan, Manisha Jadhav, and Yogesh Dandawate (Apr. 2013). “Performance Evaluation of Structural Similarity Index Metric in Different Colorspaces for HVS Based Assessment of Quality of Colour Images”. In: *International Journal of Engineering and Technology* 5, pp. 1555–1562.

Hendrycks, Dan and Kevin Gimpel (2016). “Gaussian error linear units (gelus)”. In: *arXiv preprint arXiv:1606.08415*.

DTU (2020). *Deep Learning with PyTorch*. https://github.com/DeepLearningDTU/02456-deep-learning-with-PyTorch.

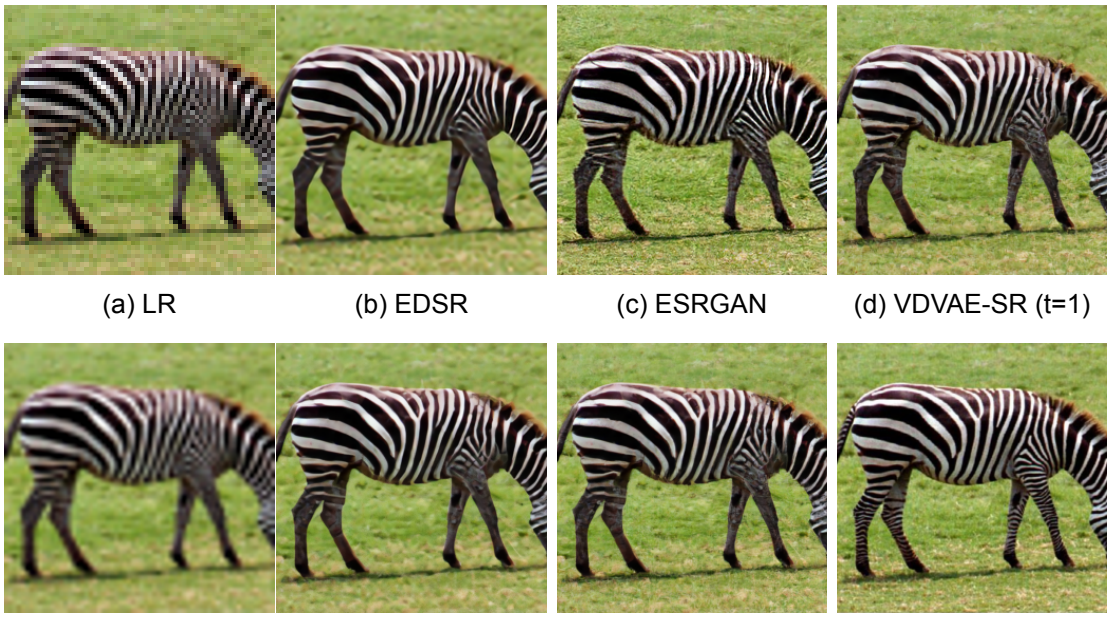
Doersch, Carl (2016). “Tutorial on variational autoencoders”. In: *arXiv preprint arXiv:1606.05908*.

Bowman, Samuel R et al. (2015). “Generating sentences from a continuous space”. In: *arXiv preprint arXiv:1511.06349*.

Bachlechner, Thomas et al. (2020). *ReZero is All You Need: Fast Convergence at Large Depth*. arXiv: 2003.04887 [cs.LG].

Seitzer, Maximilian (Aug. 2020). *pytorch-fid: FID Score for PyTorch*. https://github.com/mseitzer/pytorch-fid. Version 0.1.1.

A Appendix 1



(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.1: Set14 Zebra

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.2: Set14 Monarch

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.3: Set14 Comic

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.4: Set14 Man

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.5: Set14 ppt3

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

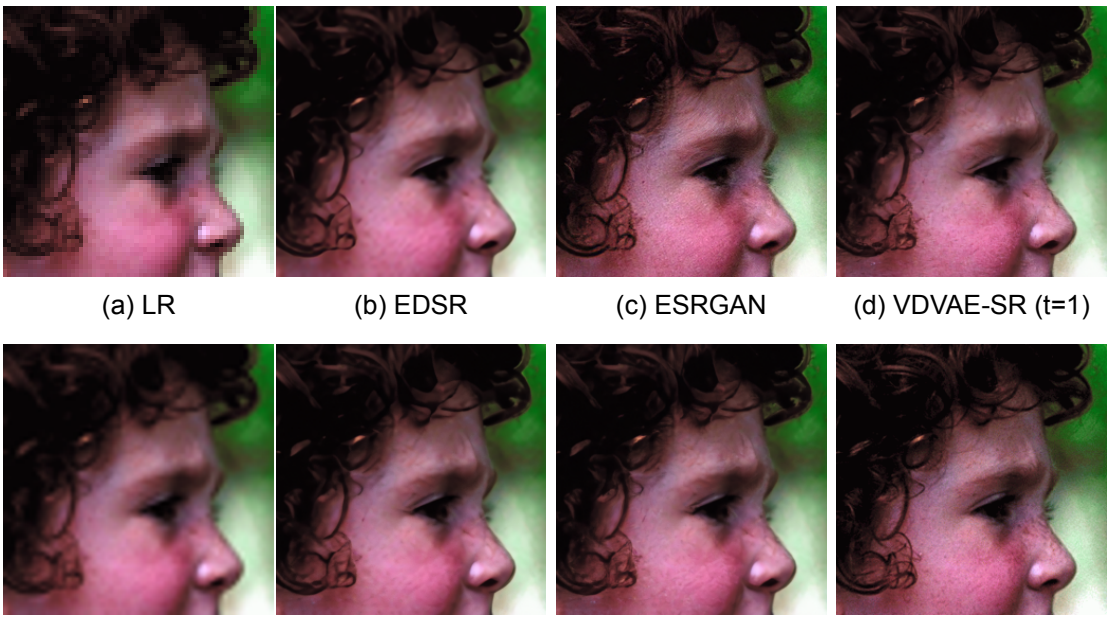
(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.6: Set14 Pepper

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.7: Set14 Foreman



(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.8: Set14 Face

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

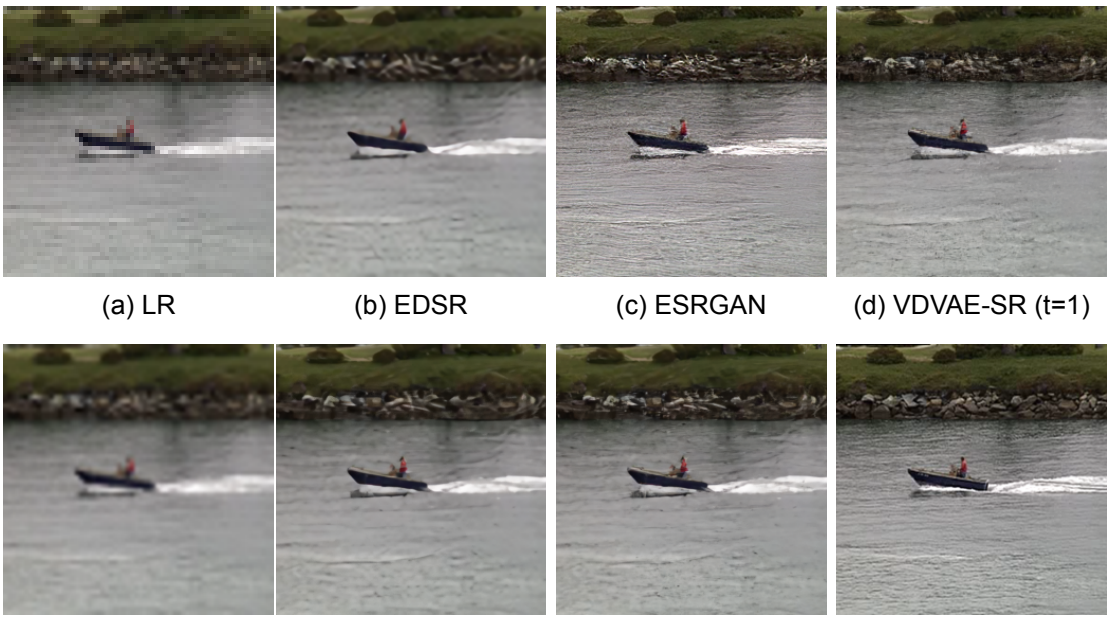
(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.9: Set14 Flowers

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

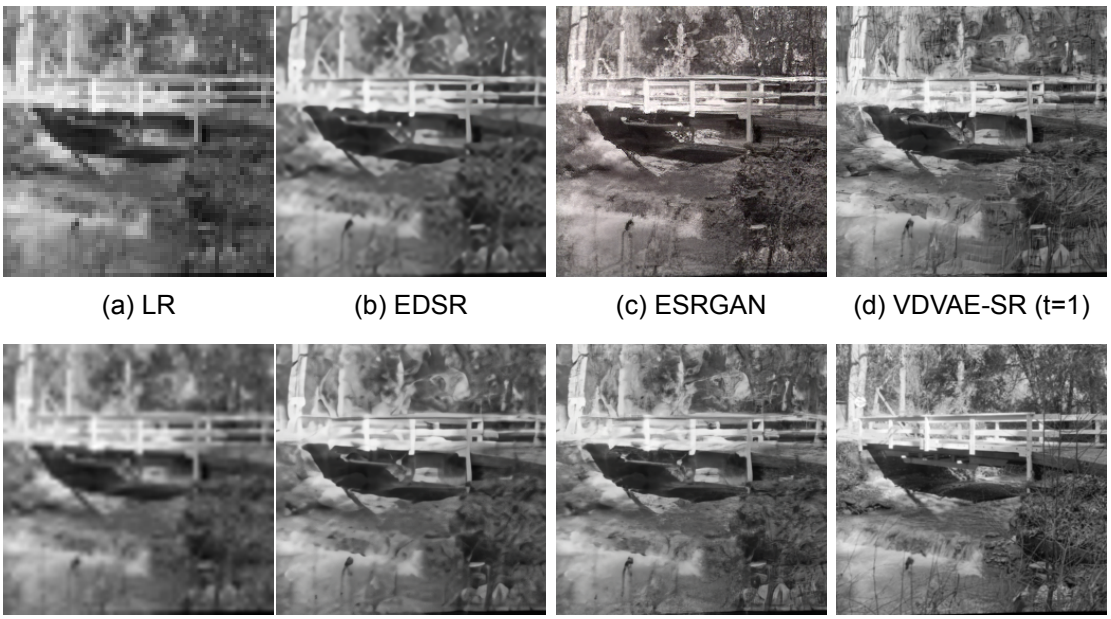
Figure A.10: Set14 Barbara



(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.11: Set14 Coastguard



(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

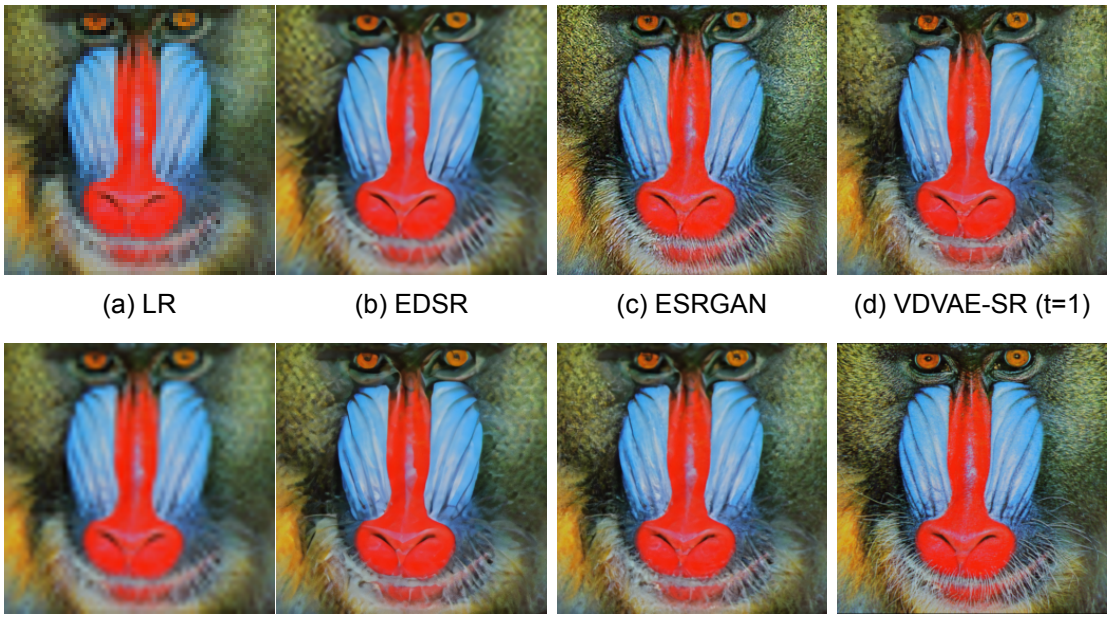
(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.12: Set14 Bridge

(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.13: Set14 Lenna



(a) LR (b) EDSR (c) ESRGAN (d) VDVAE-SR (t=1)

(e) bicubic (f) VDVAE-SR (t=0.2) (g) VDVAE-SR (t=0.8) (h) Reference

Figure A.14: Set14 Baboon

Image super-resolution (SR) techniques are used to generate a high-resolution image from a low-resolution image. Deep Generative Models (DGMs) such as autoregressive (AR) models, Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) have proven to be effective at modelling high-resolution images. In particular, GAN-based SR models have achieved state-of-the-art SR performances. Whereas VAEs used to be criticised for their feeble generative performances, there is now strong evidence that deep VAEs can outperform both GANs and AR models for high-resolution image generation. The project aims at exploiting the most recent deep VAE, specifically the state-of-the-art VDVAE generative model to improve upon existing SR models.
This project uses transfer learning to utilize the VDVAE model combined with an extra component in order to support the task of image super-resolution with four times upscaling (x4). The model we present, which we call VDVAE-SR was able to output sharper images than a purely Convolutional Neural Network method EDSR (winner of NTIRE17 SR Challenge) in terms of FID score and even comparing to the popular GAN-based SR method ESRGAN (winner of PIRM-2018 SR Challenge).

The source code can be found on: https://github.com/dman14/VDVAE-SR

Technical
University of
Denmark

Richard Petersens Plads, Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

www.compute.dtu.dk