TURUN YLIOPISTO

# Super-Resolution Diffusion Model as a Data Augmentation Method in Convolutional Neural Networks


Data Analytics

Master's Degree Programme in Information and Communication Technology

Department of Computing, Faculty of Technology

Master of Science in Technology Thesis

Author:

Oona Leppänen

Supervisor:

Professor Timo Knuutila (University of Turku)


November 2025

**Master of Science in Technology Thesis**
**Department of Computing, Faculty of Technology**
**University of Turku**

**Subject**: Data Analytics
**Programme:** Master's Degree Programme in Information and Communication Technology
**Author**: Oona Leppänen
**Title**: Super-Resolution Diffusion Model as a Data Augmentation Method in Convolutional Neural Networks
**Number of pages:** 96 pages, 17 appendix pages
**Date**: November 2025

This thesis will examine if the SeeSR diffusion model can be used as a data augmentation method by generating super-resolution image data that is given to a CNN. The results of the CNN will be compared to results of two other CNNs that have low-resolution and high-resolution image data as their inputs to evaluate the performance of the CNN trained with super-resolution image data. In addition, a component of the SeeSR model will be changed to enhance the detail generation capability of SeeSR. The effect of this change will be investigated by comparing the new version to the original SeeSR model.

The new version of SeeSR will be evaluated by calculating Peak-Signal-to-Noise Ratio, Structural Similarity Index Measure, Fréchet Inception Distance and Natural Image Quality Evaluator metrics and comparing the results to the results of the same metrics of the original SeeSR model. The CNN models will be evaluated by calculating accuracy, precision, recall and F1-score and comparing the results to one another.

The new SeeSR version performs worse with the reference Image Quality Assessment metrics and Fréchet Inception Distance but overperforms to an unrealistic level on Natural Image Quality Evaluator. The CNN model trained with super-resolution image data performs best of the CNN models which implies that a diffusion model can be beneficial as a data augmentation method. To improve the pipeline, the data could be more robust, the CNN models could be more optimized and a newer diffusion model with better detail generation capability could be utilized.

**Keywords:** Diffusion Model, Convolutional Neural Network, CNN, Super-Resolution, Deep Learning, Data Augmentation

**Diplomityö**
**Tietotekniikan laitos, Teknillinen tiedekunta**
**Turun yliopisto**

**Oppiaine**: Data-analytiikka
**Tutkinto-ohjelma:** Tieto- ja viestintätekniikka
**Tekijä**: Oona Leppänen
**Otsikko**: Super-Resolution Diffusion Model as a Data Augmentation Method in Convolutional Neural Networks
**Sivumäärä**: 96 sivua, 17 liitesivua
**Päivämäärä**: Marraskuu 2025

Tämä tutkielma tutkii, voiko SeeSR-diffuusiomallilla toteutettu super-resoluutio toimia data-augmentointimenetelmänä generoimalla super-resoluutiokuvadataa, joka syötetään CNN:lle. Tämän CNN:n tuottamia tuloksia verrataan kahteen muuhun CNN:ään, jotka on koulutettu matalaresoluutioisella ja korkearesoluutioisella kuvadatalla. Tarkoituksena on näin arvioida super-resoluutiodataa käyttävän CNN:n suoriutumiskykyä. Lisäksi eräs SeeSR-mallin osista vaihdetaan toiseen, jotta kyseisen mallin yksityiskohtien generointikyky parantuisi. Muutosta tutkitaan vertaamalla tätä SeeSR:n uutta versiota alkuperäiseen SeeSR:ään.

SeeSR:n uusi versio arvioidaan käyttämällä seuraavaksi lueteltuja menetelmiä ja vertaamalla niiden tuloksia alkuperäisen SeeSR:n tuloksiin: huippusignaali-kohinasuhde, rakenteellinen samankaltaisuusindeksimenetelmä, Fréchetin aloitusetäisyys-menetelmä ja luonnollisen kuvalaadun arvioitsija. Super-resoluutiodatalla opetettu CNN arvioidaan käyttämällä täsmällisyys-, tarkkuus- ja herkkyysmenetelmiä sekä F1-arvoa ja tuloksia verrataan matalaresoluutioisella ja korkearesoluutioisella datalla koulutettujen CNN:ien tuloksiin.

Uusi versio SeeSR:stä suoriutuu huonommin kuvan laadun arvioinnin viitemitoilla ja Fréchetin aloitusetäisyys-menetelmällä, mutta se ylisuoriutuu epärealisten hyvin luonnollisen kuvalaadun arvioitsijalla. Super-resoluutiodatalla opetettu CNN-malli suoriutuu parhaiten verrattuna muihin CNN-malleihin. Tämä viittaa siihen, että diffuusiomallia voisi käyttää data-augmentointimenetelmänä. Jotta tätä liukuhihnaa voisi parantaa, käytetty data voisi olla vankempaa, CNN-malleja voitaisiin optimoida ja uudempaa diffuusiomallia voisi kokeilla nykyisen sijasta, jotta voitaisiin saavuttaa parempi yksityiskohtien generointikyky.

**Asiasanat:** Diffuusiomallit, Konvolutiivinen neuroverkko, CNN, Super-resoluutio, Syväoppiminen, Data-augmentointi

# Foreword

Special thanks to Doctoral Researcher Rami Ilo for assistance and supervision on this thesis from 22.3.2024 to 1.10.2025.

Turku, November 2025
Oona Leppänen

# Table of contents

**Foreword** 4

**1 Introduction** 7

**2 Background** 10

**2.1 Image Restoration** 10

**2.2 Super-Resolution** 10

**2.3 Deep Learning** 12

2.3.1 Convolutional neural networks 14

2.3.2 U-Net 16

**2.4 Diffusion Model** 17

2.4.1 Stochastic Differential Equation and Denoising Diffusion Probabilistic Model 22

2.4.2 Supervised and Zero-Shot Learning 23

2.4.3 Conditioning 25

2.4.4 Guidance 26

2.4.5 Attention 27

2.4.6 Latent Space and Variational Autoencoder 30

2.4.7 Challenges of Diffusion Models 32

**2.5 Models for experiment** 36

2.5.1 Text-to-Image Diffusion Models and Stable Diffusion 36

2.5.2 ControlNet 38

2.5.3 Semantic-Aware SR 39

2.5.4 Content Consistent Super-Resolution 47

**2.6 Evaluation of Diffusion and CNN models** 49

2.6.1 Peak Signal-to-Noise Ratio 50

2.6.2 Structural Similary Index Measure 50

2.6.3 Fréchet Inception Distance 51

2.6.4 Natural Image Quality Evaluator 52

2.6.5 Common Model Evaluation Metrics 53

**2.7 Data Augmentation** 55

**3 Initial Dataset** 58

**4 Experimental Setups** 61

**4.1 Preprocessing** 65

**4.2 Diffusion Model Setup** 67

4.3 Metric Component Setup 69
4.4 CNN Model Setups 71
5 Experiment Results and Discussion 75
5.1 Diffusion Model's Results 75
5.2 Comparison of High-Resolution and Super-Resolution Datasets 78
5.3 Classification Results 86
6 Conclusion 93
References 97
Appendixes 105
Appendix 1. Example Images of Classes in the Datasets 105
Appendix 2. Example Images of SeeSR and the Modified SeeSR 109
Appendix 3. Example Images of Non-Human Class with Different Resolutions 115

# 1 Introduction

Convolutional neural networks, also called CNNs, are a type of Deep Learning that is used to process image, video, signal and other tensor data. When training CNNs, some data augmentation methods are typically applied to the input data for multiple reasons. The most important of them is that the input dataset is too small despite many datasets being freely available on the Internet. For example, good quality data from different seasons, time of day or wildlife in their natural inhabitants are hard to get access in large quantities.

Deep learning needs input data in huge amounts to guarantee successful learning, and one way to attempt to solve this issue is by using data augmentation. For example, nighttime could be simulated by applying different data augmentation methods that affect the colorization and other properties of the input images.

The available data augmentation methods don't currently include a method that could increase image size. This could be done with a super-resolution method that enhances low-resolution data into super-resolution data. It would improve the quality of the training data in some respects, like blurriness and details, while keeping the relative image sizes constant across the dataset. Super-resolution is an image restoration task where the size of an image is increased to make the image sharper and more detailed. Additionally, the amount of existing high-resolution data is rather small so being able to increase the amount of high-resolution data with super-resolution opens more possibilities for high-resolution data usage.

In this thesis a diffusion model is utilized in creating super-resolution data. Diffusion models are a type of deep learning that are used in multiple different fields and tasks but especially in visual computing such as image restoration tasks. A variety of different deep learning alternatives exist for super-resolution, but diffusion models were chosen because they are relatively new models that can achieve better results than the state-of-the-art models, Generative adversarial networks (GANs), especially in visual generation tasks [1].

The idea of using super-resolution for data augmentation isn't entirely new, but very little research exists on the topic yet. Ruikai Zou and Yan Cang proposed a data augmentation algorithm that is based on super-resolution in their paper "Super-resolution-based data augmentation algorithm for few-shot bearing defect detection" published in July 2025 [2]. Qiqi Zhu et al. used super-resolution GAN for augmenting remote sensing images (RSIs) for CNN classifier with results of improved accuracy in their study "Super Resolution Generative

Adversarial Network Based Image Augmentation for Scene Classification of Remote Sensing Images" published in 2020 [3]. Wentian Qu et al. use super-resolution to support and enhance their data augmentation algorithm in their paper "HOGSA: Bimanual Hand-Object Interaction Understanding with 3D Gaussian Splatting Based Data Augmentation" published in January 2025 [4].

For the purposes of this thesis, it isn't possible to build and train a diffusion model with the resources available, so a pretrained diffusion model is chosen to do the super-resolution task. Because the classification task needs sharp and clear details, but the chosen diffusion model, SeeSR [5], produces somewhat blurry details, detail enhancement is performed for the model. It is done by changing the decoder of the variational autoencoder (VAE) of SeeSR to the decoder of the VAE of the CCSR [6] model that might be able to enhance the details of SeeSR. This leads to a question regarding the quality of the new version of the SeeSR model.

The aim of this thesis is to generate high-quality and detailed super-resolution images and test them against their high-resolution and low-resolution variants by utilizing a CNN classifier. The CNN classifier is used to evaluate the effectiveness of super-resolution data in a classification task. To do this, a high-resolution dataset is created by assigning class labels to a high-resolution dataset acquired from the internet [7]. Then a low-resolution version of it is created by downscaling the high-resolution dataset and, finally, the super-resolution dataset is created by using the modified SeeSR model on the low-resolution dataset. The performance of the modified SeeSR model is tested using some typical image quality metrics, and lastly a base CNN classifier architecture is created and by utilizing it three different CNN classifier versions are trained with the different datasets. The results of the CNN models are compared to each other for examining the performance of the super-resolution data compared to the other datasets.

The research question of this thesis regarding the SeeSR model is

**RQ1**: How does the changing of the variational autoencoder of the SeeSR model affect to the performance of SeeSR?

The research questions of this thesis regarding the CNN evaluations are

**RQ2**: How well does super-resolution diffusion model perform as a data augmentation method from the perspective of a CNN classifier?

**RQ3**: How well does super-resolution data generated with a diffusion model approximate the original high-resolution data counterpart from the perspective of a CNN classifier?

**RQ4**: How does image resolution affect the classification performance of a CNN model?

With RQ1 the aim is to find out the performance changes that happened after the diffusion model is modified to determine did the change affect the model and its outputs in any way, how it changed the model and was the change an improvement. In RQ2 the aim is to compare the super-resolution and low-resolution data to find out how the increase of image sizes and the sharpness and clearness of the data affects the classification compared to the slightly blurry data with small image size.

In RQ3 the super-resolution dataset is compared to its high-resolution counterpart to find out the quality of the approximation. With RQ4 a baseline for the previous two comparisons is set. Answering it tells how much classification accuracy is lost by lowering the resolution of the data. The research questions RQ2-4 are studied from the perspective of a CNN classifier.

This thesis expects the reader to have basic knowledge about CNNs, and common metrics used with CNNs, like accuracy, precision and recall. Beneficial, but not required, prior knowledge include the U-Net, attention, latent space and VAEs.

The outline of this thesis is the following: In chapter 2 the background information for super-resolution, deep learning, diffusion models, CNNs, chosen diffusion models and different evaluation metrics are discussed to gain all necessary knowledge for the experiments. Chapter 3 discusses the datasets used in this thesis and how they are formed. Chapter 4 sets up the experiment discussing the preprocessing and the setups of the used deep learning models. In chapter 5 results of the experiments are presented and discussed. Chapter 6 is the final chapter that concludes the thesis and discusses future work possibilities.

# 2 Background

## 2.1 Image Restoration

Image restoration is a hard topic [1] that has existed multiple decades. It's a low-level vision task which purpose is to either recover the original image from a degraded image [8] or remove undesired objects from images, such as haze, shadow or rain. [9] Because the original images are inferred from the degraded images, i.e. the causes from the observations, image restoration tasks are inverse problems. Inverse problems are typically ill-posed meaning that the solution doesn't fulfil at least one of the following criteria: a solution exists, it is unique or that it is stable. Different regularization methods have been applied to image restoration tasks to counter the non-stability of the solution. [8]

Because many imaging applications produce degraded images, there exists a need for image restoration methods. This degradation can occur during formation, transmission or storage of an image. It can be caused by sensor blur (resulting from spatial averaging at photo sites), noise, motion blur (shaking of a camera or movements of an object), optical aberrations (e.g. out-of-focus blur), atmospheric distortions (e.g. aerosol scattering) and so on. For example, the photographs may be taken in low-light conditions which exposes the images to noise, out-of-focus photographs are blurry, and image taken from a license plate of a moving car may be unreadable because of motion blur. [8]

Most existing image restoration solutions usually concentrate on one of the tasks such as removing blur although some solutions attempt to solve multiple tasks simultaneously [1]. Super-resolution, deblurring and denoising are examples of tasks recovering high-resolution images from low-resolution images and shadow removal, dehazing and deraining are some examples of tasks removing undesired objects form images. [9] Image restoration can be used in various fields, like photography, surveillance, astronomy and medical imaging to name a few [8].

## 2.2 Super-Resolution

Super-resolution (SR) is one of the image restoration tasks which aims at restoring high-frequency content that has been lost during image formation [8], or in other words it improves the resolution of the images [11]. Super-resolution is inverse and ill-posed problem because the number of possible high-resolution images that are consistent with a low-resolution image

is infinite. More precisely expressed, when the generated output images are compared to the ground truth their loss values are similar although they look different subjectively view because of their colour, brightness or some other factors [10]. However, using regularization as part of the super-resolution techniques the ill-posed problem can be changed into better-posed problem. The idea is to use prior information to produce a better-posed solution. [8]

Super-resolution is done by producing a high-resolution image based on a set of low-resolution images. The set consists of one or more low-resolution images. [8] The process can be divided into two categories based on the number of low-resolution images in the set. If the set contains only one low-resolution image, the super-resolution process is called single-image super-resolution (SISR). Correspondingly, the super-resolution process is called multi-image super-resolution (MISR) if the set contains more than one low-resolution image. MISR isn't as challenging as SISR because it has much more information to use for reconstruction of image features. [11]

Nowadays, deep learning-based super-resolution techniques are the state-of-the-art techniques, and they are utilized in SISR because they can do SISR processes faster and more efficiently than the traditional methods. Therefore, SISR is used widely in many different computer vision applications, such as video enhancement, medical image reconstruction and surveillance imaging. Deep learning counters the ill-posed nature of SISR through its loss function and architecture. Because this thesis is about a deep learning-based super-resolution, SISR is used as the approach. [11]

Super-resolution deep learning models typically require training sets consisting of image pairs to learn the features and achieves the ability to produce high-resolution images from low-resolution images. An image pair consists of a high-resolution image and a low-resolution image that should be the same image but with different resolutions. However, the low-resolution image isn't usually a real image but rather a simulated image because it's hard to take the exact same image twice but with different resolutions. This is why most of the existing models utilize simulated datasets for training and testing. [11]

So, usually only a high-resolution image is taken, and the low-resolution image is degraded from it by utilizing a degradation pattern. The test sets consist of only the low-resolution images, so the high-resolution image generation ability of the deep learning models can be tested. [11]

The artificial degradation of images entails problems. They aren't typically as complex as captured images due to many disturbances during the life cycle of the captured images, such as motion blur, compression artifacts or sensor noise. Because of this, the SISR models trained with degraded images perform poorly when dealing with the captured images. To answer this problem, some real-world/blind super-resolution models have been developed, for example the SR3+ model [12] and SeeSR [5]. [11]

SISR has multiple benchmark datasets for training and testing. The most used dataset for training is DIV2K [13] with 1000 high-resolution images. Other training sets are for example Flickr2K [14], BSDS300 [15] and DF2K [7]. For effective testing of model performance, a lot of benchmark test sets exist, like Urban100 [16], Set5 [17], Set14 [18] and BSD100 [19]. Some of the benchmark datasets weren't originally created just for super-resolution but were created for other tasks, such as ImageNet [20] and CelebA [21] which are for object recognition [20] and face attribute recognition, respectively. Real-world SISR has its own dataset called RealSR. [11]

## 2.3 Deep Learning

Deep learning is a very popular type of machine learning because it has shown huge potential in different machine learning tasks, for example in computer vision. See figure 1. One of the reasons for its triumph over other machine learning types in many cases is its capability to process huge amounts of data [22]. Therefore, deep learning can be used in various contexts, such as computer vision, natural language processing (NLP), bioinformatics, processing speech and audio, robotics, chemistry, online advertising, video games, finance and search engines [22].

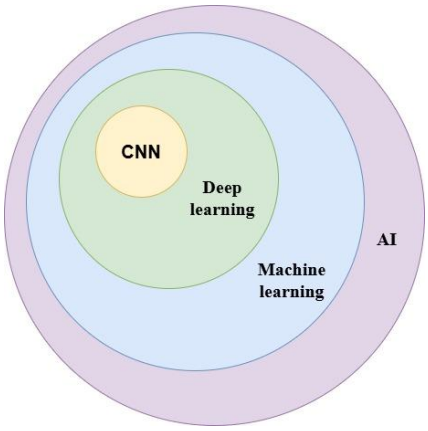


Figure 1. Deep learning in the context of AI. [22]

Deep learning is based on our knowledge regarding applied mathematics, statistics and human brains. It represents inputs, like images or text, as a nested hierarchy of concepts where every concept has been determined based on a simpler concept. After representing the inputs, it makes predictions based on the network it created by representing the inputs. For example, an image consists of pixels that determine edges. These edges can determine different corners and contours that can determine object parts of an image. Therefore, it can be said that a deep learning model gets pixels of an image as an input, and it predicts an output based on the object parts. [22]

Each of the different concept dwells in a different layer. Therefore, deep learning model consists of an input layer, at least two hidden layers and an output layer in corresponding order. The inputs are given to the model in the input layer, predictions are produced in the output layer, and the hidden layers determine the features of the inputs such as edges and then do the task the model is made for, such as classification or object detection. See figure 2. The name “deep learning” refers to the many hidden layers deep learning models consist of. [22]

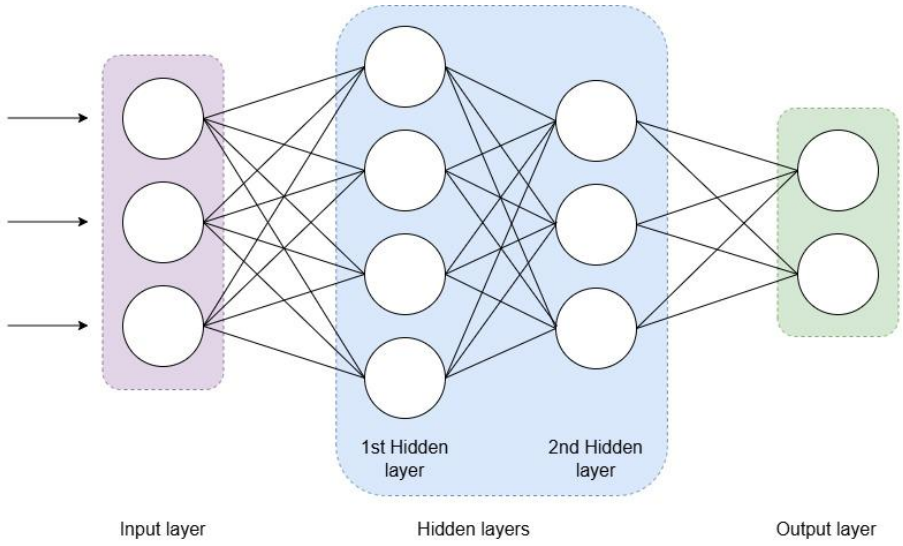


Figure 2. A fully connected deep learning architecture. [22]

As can be seen from figure 2, each layer consists of nodes, and the layers are joined via connections. If all nodes of a layer are joined to all nodes of the next layer, the model is said to be fully connected. Otherwise, the model is partially connected. To end up in the node(s) of the next layer, the input data is multiplied with weights, and some bias is added to it. Weights are learnable parameters that determine how much each feature affects the prediction of the model. Bias is a parameter for regulating the activation of the nodes. In the node the result goes through an activation function that determines whether the node activates. When it

activates, the data flows to the next node. While flowing, the data is multiplied with new weights, and a new bias is added. This process of multiplication, addition and activation is repeated until the output layer is reached.

When the process is done, a cost function, which is also called a loss function, is calculated. It reports the error between the prediction and the ground truth that is the "real answer". The error, loss, is a value that states how much the prediction differs from the ground truth. After the loss if achieved, backpropagation is done with an optimizer. Backpropagation is a method for updating the parameters in the model. The optimizer is an optimization algorithm that minimizes the cost function by updating the parameters during backpropagation. SGD, Adam and RMSprop are popular examples of optimizers. Learning rates are hyperparameters that are used for tuning optimizers.

### 2.3.1 Convolutional neural networks

Convolutional neural network (CNN), or convolutional network, is a form of deep learning that is a very popular deep learning model type nowadays especially in computer vision [23] and NLP. CNNs process data that has a grid-like topology, such as images and videos, and their architecture is flexible to different input dimensions, like one-dimensional (1D) data such as text, or two-dimensional (2D) data like images. [22] Because of their popularity in the computer vision field, their simpler architecture and their effectiveness, CNNs are chosen to be the classification model in this thesis.

Concepts and architecture of the CNNs are gone through in this section briefly. To familiarize yourself with more information about the CNNs, it is recommended that you examine chapter 9 in the Deep Learning book by Ian Goodfellow et al. [22], a CNN course of Stanford university [24] or an introduction paper to CNNs by Keiron O'Shea and Ryan Nash [25].

One relevant concept of deep learning is convolution that describes an operation that consists of many parallel convolutions i.e. multichannel convolution. Usually, cross-correlation is used in CNNs instead of convolution although it is still referred to as convolution. The difference between these operations is that in convolution the kernel is flipped but in cross-correlation it isn't. They are otherwise the same operation. [22]

The kernel is square "window" that moves over the data matrix and implements convolution in each location it "sees". For example, 2x2 kernel moves over a 5x5 image seeing four pixels

at a time and doing the convolution operation on them. [22] The typical kernel sizes are 3x3, 5x5 and 7x7, but the 3x3 kernel is the most common choice and it is also used in this thesis.

A CNN model architecture consists of three kinds of building blocks that are stacked on top of each other: convolutional layers, pooling layers and fully connected (FC) layers. Feature extraction is carried out in convolutional and pooling layers, and the final output is mapped from the features in FC layers. A CNN model for classification outputs a class for the input as its final output. [23]

The main component of a CNN architecture is the convolutional layer that consists of convolution operations [23]. In the convolutional layer convolution or cross-correlation is performed. A CNN model starts with at least one convolutional layer. Each convolutional layer is always followed by an activation function. Because the convolutional layers produce a linear activation, it is changed into a nonlinear activation with an activation function. [22] Rectified linear unit (ReLU) is the most used activation function today [23] and it is also utilized in this thesis.

Pooling layer comes after one or more convolutional layers and their corresponding activation functions. The pooling layer changes the output of previous layers by replacing the elements of their output array with the summary statistics of nearby elements. [22] The hyperparameters of the pooling layer are stride, padding and filter size. Stride is the distance between the locations of two consecutive kernels. It is measured as the number of elements in the input matrix. [23] Padding is used to avoid reducing the size of the output during the pooling operation [22]. Zero padding is the most common choice, but other padding choices can be made as well like same padding, which is used in this thesis.

The two most common pooling operations are max pooling and average pooling. The filter of the max pooling starts from the left corner of the given matrix, then it checks its overlapping area for finding the maximum value of the elements in the area and chooses that value to represent the elements the filter overlaps. Then the filter moves to the next area it did not overlap. The output of max pooling layer consists of those maximum values and is given to the next layer as an input. See figure 3. Max pooling is the most used pooling operation with its typical filter size of 2x2 and stride size of 2 [23]. It is used in this thesis with those typical hyperparameter values.

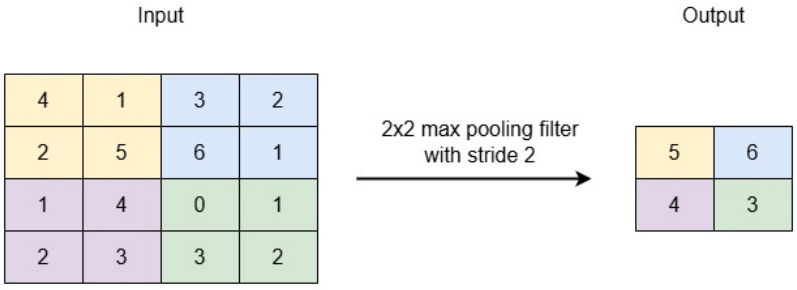


Figure 3. Max pooling with 2x2 filter and stride 2. [23]

The pooling layer isn't necessary, but it is widely used. It can be left out of a CNN model if the reduction of spatial dimensions is undesirable. In that case instead of leaving out the whole pooling layer unit stride can be used to control overfitting. However, pooling enhances statistical efficiency giving it relevancy in CNNs. [22]

FC layers are the last layers in a CNN model, but it isn't necessary to use them at all. Typically, one or more FC layers are added to the end of a model. These layers produce the final output from the feature maps, such as a class for classification or a mask for segmentation.

In this thesis CNNs are utilized to evaluate a super-resolution dataset against a high-resolution and low-resolution datasets by finding differences between CNN model performances caused by the different datasets. More of this can be read from chapter 4.4 where the CNN model setup and reasonings are discussed.

### 2.3.2 U-Net

U-Net is a deep learning architecture that was developed for medical segmentation tasks. Besides segmentation, U-Net is used in many other tasks and models like in diffusion models. [26]

The structure of the U-Net consists of an encoder and decoder parts that are also called as contracting path and expansive path, respectively [26]. In the middle of the U-Net is a middle block which is the end point of the encoder and the starting point of the decoder. The encoder and decoder can consist of different kinds of blocks. See figure 4. All blocks consist of a set of neural network layers that are typically used together. The layers can be convolutional layers, transformer layers, resnet layers and multi-head attention layers just to name a few.

For example, U-Net backboned Stable diffusion (SD) consists of convolutional layers, resnet layers and vision transformers (ViTs). [27]

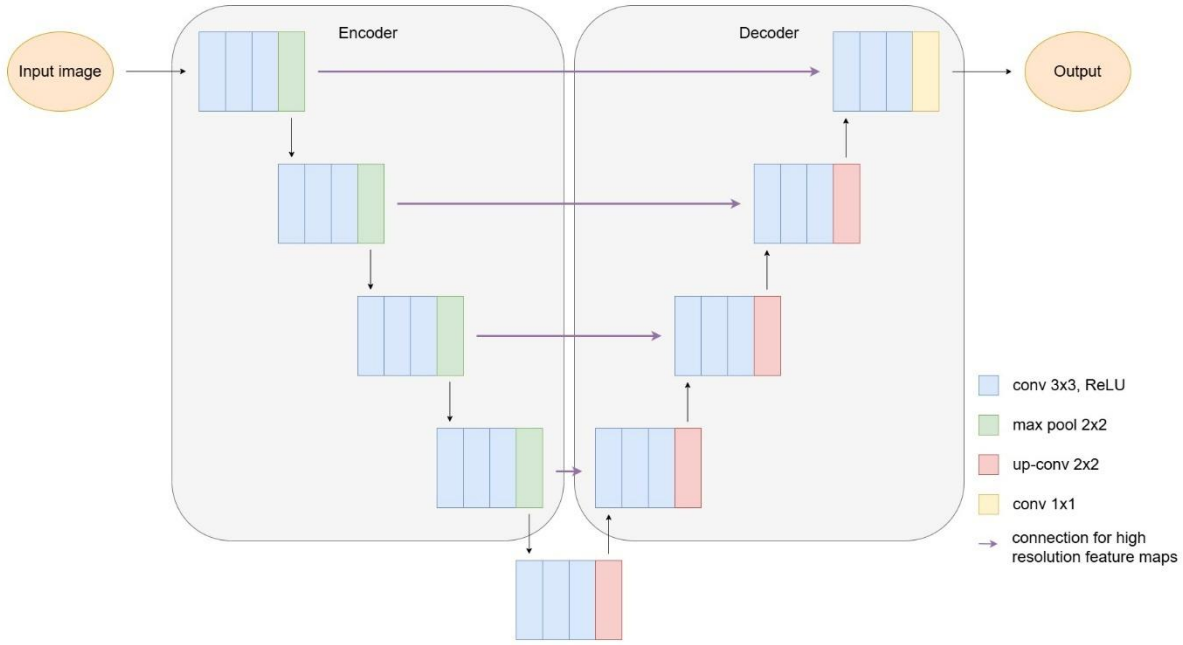


Figure 4. The architecture of the initial U-Net that is only composed of convolutional layers and introduced in [26].

Initially, U-Net was based on fully convolutional network that was modified, expanded and the pooling operations of its latter part were replaced with upsampling operations. With the replacement, the resolution of the output can be increased.

To get a more exact output and to ensure minimal data loss during the process the encoder's blocks are connected to decoder's same level blocks for joining the high-resolution features to upsampled output. [26] These connections are called skip connections. Those high-resolution features are spatial details, and the decoder deals with broad contextual features. By combining these features, the U-Net is also able to obtain fine-grained structures from the data. The usage of skip connections usually improves the performance of the model U-Net is part of and quickens its training. [28] Typically the encoder and decoder parts are relatively symmetrical which makes the structure of the U-Net to resemble upper case letter U giving the structure the name "U-Net". [26]

## 2.4 Diffusion Model

Diffusion probabilistic model, usually called diffusion model, is a probabilistic deep learning model that is widely used in visual computing nowadays [1]. They have proved to be very

effective and have superior generative capability [1] despite their challenges in different aspects of the models. The challenges are discussed in chapter 2.4.7. Some application fields, where diffusion models are effective and where they are utilized, are data generation and editing, [29] image restoration and text to image tasks [1]. Thus, the data given to the models can be text, tags, images, videos or 3D or even 4D data and so on [29]. This thesis discusses diffusion models from the context of the super-resolution and image restoration tasks.

Diffusion models have been chosen to be used as the super-resolution deep learning model in this thesis because they perform better than the state-of-the-art models such as GANs in computer vision field. Moreover, the diverse generation results of high-fidelity diffusion models are much better than the results of lower fidelity GANs. Additionally, while GANs tend to fail at recovering the texture of objects, diffusion models don't have this same limitation allowing diffusion models to surpass them in this aspect as well. [1]

Next, some general information about the architecture and implementation of the diffusion models are discussed starting from the architecture of the diffusion models: There are two different architectures in research: the diffusion model can be based on U-Net with CNN or on transformer. The U-Net architecture was especially popular in earlier studies, but transformers have become very popular in recent years. The U-Net architecture has been refined by adding a cross-attention module, multi-head attention, position encoding and group normalization to it. However, the transformers are good at uniting the different tasks of a model and modelling the long-range dependencies achieving better predictions of noise in the backward process. [1] So, it seems like the transformers are replacing the U-Net architectures.

The operating principle of diffusion models flows in the following way: The core of the diffusion pipeline changes the input data distribution into some reference distribution. This reference distribution is typically Gaussian noise. This change is done by utilizing an iterative process such as Markov chains to add noise to the data until the data satisfies the target reference distribution. [9] After that, the pipeline learns to reverse the previous process to sample structured data from the reference distribution [9] by denoising the noised input with noise prediction or score estimation to mimic the original input data as well as possible [1].

Both processes are done iteratively. [30] The first process is called forward process, diffusion process or forward diffusion [1]. The latter process, which denoises the data, is called reverse process, backward process or backward diffusion [1] and the steps of that process are called denoising steps [30]. Backward process is a characteristic that evolved the field of generative

models significantly by leaving out the unstable and complex generation process replacing it with a stable and independent iterative backward processes [1]. A U-Net architecture was initially implemented to diffusion models because of their suitable nature to predict denoised data iteratively [30].

The idea of the diffusion models is expressed mathematically in the following way: All data examples are picked independently from a data distribution $p_{data}(\boldsymbol{x})$ that is composed of the training set. Because the idea is to synthetize more data examples from $p_{data}(\boldsymbol{x})$, a diffusion model is fit to $p_{data}(\boldsymbol{x})$. Then independent and identically distributed (i.i.d.) noise with variance $\boldsymbol{\sigma}^2$ is added to the example $\boldsymbol{x}$ as it goes through the forward process ending up at noisy data example distribution $p(\boldsymbol{x}, \boldsymbol{\sigma})$. The noise levels range from $\boldsymbol{\sigma}_{max} > \ldots > \boldsymbol{\sigma}_0 = 0$ and if the $\boldsymbol{\sigma}_{max}$ is large enough the data is hidden by the noise almost entirely. In that case the noise and $p(\boldsymbol{x}, \boldsymbol{\sigma})$ can't be differentiated from each other. If this happens, it enables sampling a noise data example $\boldsymbol{x}_T \sim N(0, \boldsymbol{\sigma}_{max}^2)$ from $p(\boldsymbol{x}, \boldsymbol{\sigma})$ so that it will be the starting point for the backward process for denoising $\boldsymbol{x}_T \sim N(0, \boldsymbol{\sigma}_{max}^2)$. [29]

In each step $\boldsymbol{x}_T \sim N(0, \boldsymbol{\sigma}_{max}^2)$ is denoised $\boldsymbol{x}_t \sim p(\boldsymbol{x}, \boldsymbol{\sigma}_t)$. When $\boldsymbol{x}_t$ reaches $\boldsymbol{x}_0$, $\boldsymbol{x}_t$ approximates the initial data example $\boldsymbol{x}$ [29]. The symbol $t$ means a time step, or diffusion step, that either happens forward or backward in time. In the forward process $q$, $t$ increases as we go forward in time and in the backward process $p$ $t$ decreases as we move backwards in time reaching zero (0) when $\boldsymbol{x}_T \sim N(0, \boldsymbol{\sigma}_{max}^2)$ is fully denoised. [10]

In image restoration the pipeline of diffusion models flows in this way: (See figure 6 for reference.) In the stage 1, dataset $D$ consists of low-resolution and high-resolution (LR-HR) image pairs: $D = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$. In the stage 2 data flows through the forward process where noise is added iteratively to the high-resolution (HR) image translating it from image space to corruption space. The current state $\boldsymbol{z}_t$ lies between the image space and the corruption space. In the forward process apply an assumption of $\boldsymbol{z}_t \sim q(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})$ and noise is added additively.

For finite cases the end point is $z_T$ where $T$ is the maximal time step ($0 < t \leq T$). [10]

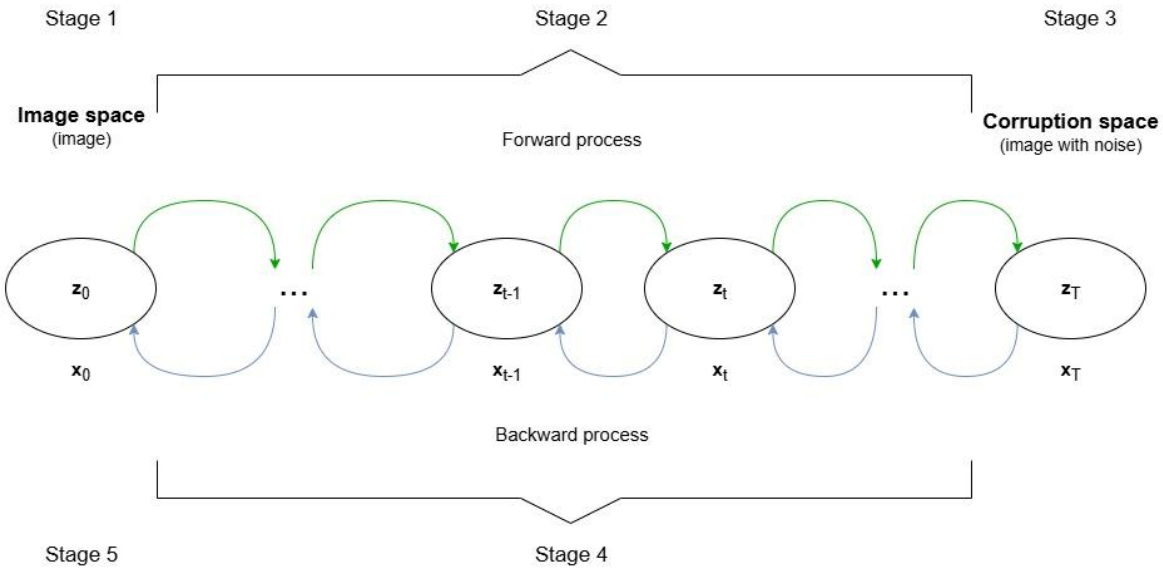


Figure 5. The concept of a diffusion model where z is state and x is image. [10]

The corruption space is the space that is reached as the final step of forward diffusion. Corruption that is injected to the input data is usually Gaussian noise, but it can be some other noise as well. More and wider research about the corruption is needed regarding the paper [10] by Brian B. Moser et al. from June 2024. [10]

In the stage 3, the image has been converted into the corruption space, and it contains noise. In the stage 4 the backward process is performed. Now the flow goes from the corruption space to the image space, and the transition is done by iterating the state $z_t$ towards the initial data distribution $\boldsymbol{z}_0 \sim q(\boldsymbol{x})$ where $t$ is zero ($t = 0$). The ideal denoised distribution is the assumption $\boldsymbol{z}_{t-1} \sim p(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t)$, but it is unattainable and, therefore, it is approximated by a parametrized model $p_\theta(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})$ where $p_\theta$ is an approximation of $p$. In the stage 5, the image space is accomplished, and the noise has been removed from the image. [10]

The more specific details about the mathematics and the diffusion model depends heavily on the diffusion model base that is used. There exist three basic bases among the diffusion models that other diffusion models commonly utilize in super-resolution tasks: Denoising Diffusion Probabilistic Model (DDPM), Score-based Generative Model (SGM) and Stochastic Differential Equation (SDE). [10] Noise conditioned score network (NCSN) is utilized as a base in image restoration tasks [1]. More on these models can be found below in the next chapter 2.4.1.

As indicated the diffusion models are much more than the base. They utilize a lot of different concepts such as state domain, corruption space, conditioning, guidance and attention [10]. Attention, conditioning and guidance are explored later in their corresponding chapters 2.4.3-2.4.5. The state domain is the domain where the input data is presented when it is introduced to the diffusion model [10]. The SeeSR model [5] utilizes latent domain as the state domain so only the latent space is in the scope of this thesis, and it can be found from chapter 2.4.6.

Diffusion models have also some other concepts such as optimization strategies that include optimizing variance and noise schedule during forward and backward processes to improve the performance and stability of diffusion models [1]. However, these are left outside the scope of this background chapter since they aren't important in the big picture of this thesis.

When it comes to the super-resolution task, the performance of the super-resolution diffusion models is good when they are used to generate low-resolution images. These models have still huge performance gaps in tasks where they generate high-resolution images as was stated in the paper [30] from April 2025. [30] Despite that they have proved notable efficacy in super-resolution and other image restoration tasks [1].

This has led to a situation where there exist many different diffusion models available for the super-resolution task without even mentioning other image restoration tasks. Classic super-resolution diffusion models are, for example, SR3 [31], SRDiff [32] and StableSR [33]. SR3 uses Gaussian noise and the denoising happens in many different levels. SRDiff was the first SISR model that was based on diffusion (uses Gaussian noise and Markov chains) and StableSR uses a pretrained text-to-image diffusion model as a base and fine-tunes it. [11]

Diffusion model bases and other relevant concepts regarding the diffusion models are choices done based on the task(s) at hand. There isn't a universal best choice for all tasks, and some methods or other choices may not be explored or don't work with all tasks. The following subchapters of the thesis will only cover topics relevant to the SeeSR model and super-resolution task specifically and less focus is placed on topics outside of this scope.

The sections of this chapter are about the relevant diffusion models and methods regarding the SeeSR model and the disadvantages of diffusion models. In the following order the sections are: 2.4.1 Stochastic Differential Equation and Denoising Diffusion Probabilistic Model, 2.4.2 Supervised and Zero-Shot Learning, 2.4.3 Conditioning, 2.4.4 Guidance, 2.4.5 Attention, 2.4.6 Latent Space and Variational Autoencoder and 2.4.7 Challenges of Diffusion Models.

### 2.4.1 Stochastic Differential Equation and Denoising Diffusion Probabilistic Model

J. Solh-Dickstein, E. Weiss, N. Maheswaranathan and S. Ganguli can be considered to have created the first prototype of diffusion models in their paper "Deep unsupervised learning using nonequilibrium thermodynamics" [34]. DDPM, NCSN and SDE have since developed the concept of diffusion models in aspects such as in model architecture, optimization strategy and sampling efficiency. These three models are usually used as a base for other diffusion models. [1] SDE and DDPM together with SGM are the common choices for base diffusion model in super-resolution. [10]

Stochastic differential equation (SDE), or Score SDE, is a mathematical process that describes a stochastic process by using stochastic differential equations (SDEs). Diffusion models utilize SDE in the forward and backward diffusion processes where SDE is a generalization for continuous and infinite time steps or in other words for infinite scaling of noise. The DDPM, NCSN and SGM are discretizations of SDE. [10]

The forward process q created by SDE is

$$d\boldsymbol{z} = f(\boldsymbol{z}, t)dt + g(t)d\boldsymbol{w} \tag{1}$$

where *f* is a drift function or a drift coefficient of ***z(t)*** [1], *g* is diffusion function or a diffusion coefficient ***z(t)*** [1], and $\boldsymbol{w}$ is standard Wiener process or Brownian motion. [10] The drift coefficient is designed for verifying the Gaussian distribution and the diffusion coefficient is the degree of random noise disturbance [1]. The discretizations of SDE can be presented uniformly by using the Equation 1 as a base. For example, the DDPM is

$$d\boldsymbol{z} = -\frac{1}{2}\alpha(t)\boldsymbol{z}dt + \sqrt{\alpha(t)}d\boldsymbol{w} \tag{2}$$

where $\alpha\left(\frac{t}{T}\right) = T\alpha_t$ when $T \rightarrow \infty$. [10] The backward process is done with reverse-time SDE that is

$$d\boldsymbol{z} = [f(\boldsymbol{z}, t) - g(t)^2 \nabla_{\boldsymbol{z}} \log q_t(\boldsymbol{z})]dt + g(t)d\tilde{\boldsymbol{w}} \tag{3}$$

where *dt* means an infinitesimal negative time step and $\tilde{\boldsymbol{w}}$ means standard Wiener process with backwards going time. [10]

Denoising Diffusion Probabilistic Model (DDPM) is a diffusion model where the forward process $q$ and backward process $p$ utilize Markov chains, and the used noise is typically Gaussian noise. The number of time steps is finite in DDPM. The forward process is:

$$q(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}) = N(\boldsymbol{z}_t \mid \sqrt{1-\ \alpha_t}\boldsymbol{z}_{t-1}, \alpha_t \boldsymbol{I} \tag{4}$$

where hyperparameter $\alpha$ is a variance of added noise at each time step, or in other words the noise schedule, and $0 <\ \alpha_{\ 1:T} < 1$. Formula (1) can be expressed with a single step:

$$q(\boldsymbol{z}_t \mid \boldsymbol{z}_0) = N(\boldsymbol{z}_t \mid \sqrt{\gamma_t}\boldsymbol{z}_0, (1-\gamma_t)\boldsymbol{I} \tag{5}$$

where $\gamma_t =\ \Pi_{i=1}^{t}(1-\alpha_i)$. This enables sampling $z_t$ directly:

$$\boldsymbol{z}_t =\ \sqrt{\gamma_t}\boldsymbol{z}_0 +\ \sqrt{1-\gamma_t}\epsilon,\ \epsilon \sim N(0, \boldsymbol{I}). \text{ [10]} \tag{6}$$

The backward process learns reverse forward process and generates a distribution similar to the prior $\boldsymbol{x}_0$. In super-resolution the prior $z_0$ is typically a high-resolution image. The backward process $p_\theta$ is:

$$p_\theta(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t) = N(\boldsymbol{z}_{t-1} \mid \mu_\theta(\boldsymbol{z}_t, \gamma_t), \Sigma_\theta(\boldsymbol{z}_t, \gamma_t)) \tag{7}$$

where $\Sigma_\theta$ and $\gamma_\theta$ are learnable. When forming a conditional DDPM, the formula is:

$$p_\theta(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t, \boldsymbol{x}) = N(\boldsymbol{z}_{t-1} \mid \mu_\theta(\boldsymbol{z}_t, \boldsymbol{x}, \gamma_t), \Sigma_\theta(\boldsymbol{z}_t, \boldsymbol{x}, \gamma_t)) \tag{8}$$

where $\boldsymbol{x}$ is the condition such as a low-resolution image. [10]

Because DDPMs must learn only the so-called noise network, U-Net is used as the base architecture of the DDPMs [9]. U-Net architecture is discussed in chapter 2.3.2.

### 2.4.2 Supervised and Zero-Shot Learning

Image restoration tasks have been researched with diffusion models a lot because of the superior texture recovery ability of diffusion models. This research can be divided into two categories from the perspective of the training strategies used in research: supervised learning and zero-shot learning. In short, supervised learning builds and trains a model from scratch using conditioning, guidance and other building blocks of diffusion models, while zero-shot learning uses a pretrained model that may be fine-tuned to a task. [1] In this section these two categories are discussed in the context of diffusion model-based image restoration methods.

Supervised diffusion model-based image restoration methods use image pairs as the dataset for training the diffusion model, where one image pair consists of a clean image and a distorted image. However, the image pair requirement is a notable restriction for these methods. The methods need large amounts of paired data for training which isn't easy to gather not to mention the huge difficulties in collecting the paired dataset from the real world. More of this challenge can be read from chapter 2.4.7. In addition, including the degraded or low-quality image into the condition of the diffusion model effectively is a challenge with these methods. For example, SR3 [31] is a supervised diffusion model for super-resolution.

Zero-shot diffusion model-based image restoration methods have learned their tasks already. This means they don't need to be trained nor need image pairs like the supervised methods unless they are fine-tuned. In that case they only need distorted images which are typically used as a condition to guide the image restoration process. Zero-shot models contain the prior knowledge of a used pretrained model including the texture and structure priors. These priors have been composed of a real-world training dataset. Zero-shot diffusion model-based image restoration methods such as Iterative Latent Variable Refinement (ILVR) [35] are promising in image restoration in low-level vision field. [1]

However, this approach has also its own challenges. One is retaining the data structure of the distorted images while extracting the perceptual priors. Another is to preserve the consistency of the data between the low-quality and high-quality images because the pretrained models don't retain the data distribution in pixel-wise data consistency. The last one is getting the perceptual knowledge from the low-quality images in models with conditioning where the images have higher requirements. [1]

Both zero-shot and supervised diffusion model -based image restoration methods use mostly synthetic distortions for their low-resolution images which don't perform well in out-of-distribution (OOD) blind/real-world degradation. Real-world image restoration, or blind image restoration, is image restoration where the distorted image from the image pair contains distortions found from an image taken from the real world. The synthetic distortion challenge can be somewhat countered by adding distortion invariant learning, real-world distortion simulation, domain translation or kernel estimation to the diffusion model. [1] More information about these methods can be found from [1]. More of the OOD degradations and blind image restoration can be read from chapter 2.4.7.

### 2.4.3 Conditioning

Controlling data generation of the diffusion models via conditions, that a user has set, is one of the most important properties of generative models [29]. In image restoration the basic architecture of a diffusion model consists of a noise predictor and a condition module. The noise predictor is a U-Net or transformer consisting of the forward and backward processes, and the condition module is a component that takes care of the conditioning of the diffusion model. The condition module is linked to the noise predictor, and it typically guides the backward process. Different types of conditions exist, and they can vary from sketches, semantic maps, texts [29] and other kinds of textual prompts to images, class labels, latent features and segmentation maps. All these different possible conditions ease the development of different applications that have different requirements, such as Stable Diffusion [36] and ControlNet [27]. [1]

Conditioning is very important in image restoration because of the nature of the tasks. The idea in image restoration is to produce a clean high-quality image based on a corrupted low-quality image, such as a low-resolution or blurred image. In these tasks the low-quality images typically guide the generation. [9] Usually, when no conditioning is used in a diffusion model, the sampling is done with the unconditional distribution $p(z)$ in backward process. A diffusion model with conditioning samples the data from the conditional distribution $p(z|\boldsymbol{x})$ where $\boldsymbol{x}$ is the condition. [29] See the formula 5 about the conditioning in DDPMs from chapter 2.5.1.

If conditioning is used, a diffusion model needs an effective condition strategy to connect the conditioning to the noise predictor [1]. One way to do this is to use simple concatenation where the condition is concatenated to the outputs of each intermediate steps. However, concatenation isn't tied to any specific location of the diffusion model architecture. Concatenation is used in tasks like uncropping, in-painting and colorization. [29] Other conditional mechanisms can be, for example, a usage of an additional classifier to guide the diffusion model via its gradient or addition of condition to noise prediction or score estimation without a separate classifier. The latter mechanism is called classifier-free conditional mechanism. [1]

In super-resolution the conditioning in diffusion models guides the sampling to generate better high-resolution images causing the diffusion models to become highly dependent of the conditioning. The low-resolution images are used as a condition, but conditioning can be

implemented with other mechanisms as well: feature reference, super-resolved reference, low-resolution reference and text-to-image information. The point of the feature reference mechanism is to extract the features with relevance from the pretrained model as shown by SRDiff [32]. The idea of the super-resolved reference mechanism is to predict a reference image by utilizing the priors of a pretrained super-resolution model as a condition. An example of this method can be seen in ResDiff [37]. [10]

The low-resolution reference mechanism uses channel concatenation to concatenate the low-resolution image and a denoised image created at time step *t-1*. The low-resolution image is used as the conditioning input for noise prediction in each time step *t* as shown in ILVR [35]. The use of text-to-image information as a conditioning method rose from a desire to use information other than low-resolution images for conditioning. This mechanism uses a pretrained text-to-image model that helps to improve image interpretation and synthesis in tasks like super-resolution. The pretrained model can be fine-tuned to create better textual descriptions. SeeSR is one example of a model that utilizes a pretrained text-to-image model as conditional mechanism. [10]

### 2.4.4 Guidance

Conditioning is essential for most diffusion models, but some cases like DDPM are formed in a way where conditioning has no effect [10]. This happens because there isn't a good way to fine-tune the strength of the conditions in conditioning. To solve this problem, conditioning information weighting is controlled via a principle called guidance [10] which gives a better opportunity to guide the diffusion trajectory. Guidance is typically applied after training. [29]

Guidance can be divided into classifier and classifier-free guidance depending on if a separate model is used or not. Classifier guidance utilizes a classifier, a separate model, to guide the diffusion by combining the gradient of the classifier to the score estimation of the diffusion model. In this guidance the diffusion model is unconditional [29]. The score function of classifier guidance is

$$\nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t \mid \boldsymbol{x}) = \nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t) + \lambda \nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{x} \mid \boldsymbol{z}_t) \tag{9}$$

where $\boldsymbol{x}$ is conditional information, $\boldsymbol{z}_t$ is current state and $\lambda$ is a hyperparameter that controls weight and $\lambda \in \mathbb{R}^+$. [10]

Classifier guidance has a trade-off between sample fidelity and mode coverage. It also suffers from the need for an additional trained classifier that can handle inputs with arbitrary noise. Most of the pretrained image classification models can't handle those kinds of inputs. [10] Also, there exists a risk where the irrelevant information captured by a diffusion model leads to a poorly defined gradient [29].

The idea of the classifier-free guidance is to reach to the same results as the classifier guidance does but without an additional classifier. [10] The purpose of this guidance was to counter the problems of classifier guidance. For this classifier-free guidance changes the training regimen so that the same network acts as the unconditional model and the conditional model at the same time. [29] This happens by changing the conditioning information to null in the unconditional score function. In this option the computational cost is quite high. The other option is to train two diffusion models: an unconditional and a conditional diffusion model. However, this option has even higher computational cost. [10] The computational cost isn't the only challenge but the samples that the classifier-free guidance produces are typically low-quality regardless of the guidance scale [29].

The score function of the classifier-free guidance is

$$\nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t \mid \boldsymbol{x}) = (1-\lambda)\nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t) + \lambda \nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t \mid \boldsymbol{x}) \tag{10}$$

with the same parameters as in the formula 9 regarding classifier guidance. [10]

A general challenge regardless of the guidance approach is a trade-off between diversity and sample quality: The larger the guidance scale the better the sample quality. However, as the guidance scale increases diversity of sample will suffer. [29] According to the paper [10] the classifier or classifier-free approaches of guidance aren't applied in super-resolution before summer 2024.

### 2.4.5 Attention

Attention, is a very popular deep learning technique [38] which idea is to draw the attention of the deep learning model to the most important areas, for example, in an image and, therefore, ignore the irrelevant parts [39]. Guiding of attention is done by assigning weights to the features extracted from the input so that the weights express the importance of each feature directly. Attention increases the performance of deep learning models [39] and our in-depth

understanding of relationships of inputs and outputs and the behaviour of the deep learning models. [38]

Attention is very popular in many different tasks including computer vision tasks like super-resolution, image generation and object detection [39]. The reasons for its popularity are the ability to train the attention as a part of a base model, the possibility to achieve state-of-the-art results, how it has enabled the success of transformers and its ability to increase the interpretability of the deep learning models. [38]

A common challenge of diffusion models regardless of the model or its effectiveness is to capture the nontrivial relationships between features and their interactions and to be able to maintain them. The purposes of diffusion models are the improvement of prediction accuracy, learning of evolving dynamic patterns and securing of controlled generation of outputs by adapting to the characteristics of the data and to changing inputs. They achieve their goals by using attention that guides the model attention by weighting and aligning the features dynamically. [30] For example, attention can assist a diffusion model by doing adjustment of weighting between parts of an image which leads to an exact controlling of the spatial details or by capturing local features available in the image during generation of outputs. The more nuanced interpretation of text and verification of the compatibility between an image and its description are possible because of the flexibility in weighting which attention makes viable. [30]

So, attention is a response to the previously presented challenge because it allows the diffusion models to learn the complex dependencies between features. With attention the quality, accuracy and interpretability of the results are enhanced. In addition, diffusion models have consistency challenges that may be avoided with attention because it verifies the alignment of the input conditions and the generated outputs. For example, in generative tasks such as text-to-image generation attention is crucial for the alignment of textual and visual representations. While diffusion models enhance the temporal and spatial control, or local information in short, they leave the maintaining of the global consistency to attention enabling it to focus on the vast contextual information. [30]

Attention is also useful for advancing the stability of the diffusion models and smoothing the transitions in the generation process by combining the information of each time step. However, attention increases the computational complexity of diffusion models. To maintain

high-quality in the generation process and to improve computational efficiency, more efficient attention mechanisms have been developed such as sparce attention. [30]

As explained attention is critical to the diffusion models. Self-attention and cross-attention are the attention mechanisms used widely in diffusion model research. Self-attention models the spatial dependencies that lie in the input types. Cross-attention pays attention to the alignment of the input types and mapping features. [30]

The spatial control mentioned earlier is necessary to the diffusion models because with it the models can control the relationships between image sections or across different input types. Attention allows spatial control by ensuring that the generated content and intended target are correctly aligned. Focusing on spatial control is especially important to achieve the high-quality results in image-to-image and text-to-image generation tasks and other similar tasks. Cross-attention is mostly used today for gaining good spatial control and it is implemented using a cross-attention layer. Attention, regardless of the mechanism, is typically positioned in the middle or higher levels in an encoder-decoder structure of a U-Net. [30] The cross-attention mechanism is used multiple times in the SeeSR model in this thesis.

Next, some mathematics about the self-attention and cross-attention. Intermediate outputs are produced in the diffusion model processes, and some features are derived from them. These features are expressed in values $\boldsymbol{V}$, keys $\boldsymbol{K}$ and queries $\boldsymbol{Q}$ in self-attention and the output of self-attention is $\boldsymbol{A}\cdot\boldsymbol{V}$ where $\boldsymbol{A} = Attention(\boldsymbol{Q}, \boldsymbol{K})$. The $\boldsymbol{V}$ and $\boldsymbol{Q}$ are d-dimensional projection matrices. Contextual information is derived between $\boldsymbol{V}$ and $\boldsymbol{Q}$ by the attention mechanism in the following way:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}) = Softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right) \tag{11}$$

where $d$ is the dimension. [29]

Cross-attention functions in roughly the same way, but it utilizes conditioning such as text prompts to control the generation process. In image synthesis an encoder $\tau$ that is specific to some domain preprocesses a condition $c$. The result is an intermediate projection $\tau(c)$ that is also called projected condition. The projection is added to different layers in the middle of the denoising U-Net using cross-attention

$$\boldsymbol{Q} = \boldsymbol{W}_Q * \varphi(\boldsymbol{z}_t), \boldsymbol{K} = \boldsymbol{W}_K * \tau(\boldsymbol{c}), \boldsymbol{V} = \boldsymbol{W}_V * \tau(\boldsymbol{c}) \tag{12}$$

where $\varphi(\boldsymbol{z}_t)$ is a result from the middle of the denoising U-Net, $\boldsymbol{Q}$ is activation projection, $\boldsymbol{V}$ and $\boldsymbol{K}$ are derived from a given condition projection and $\boldsymbol{W}_K$, $\boldsymbol{W}_V$ and $\boldsymbol{W}_Q$ are learnable projection matrices. [29]

### 2.4.6 Latent Space and Variational Autoencoder

Depending on the desired state domain for a diffusion model different methods need to be utilized to achieve the specific domain. All previous formulas are presented in pixel space, which is the most common domain, because images are naturally in that space. However, diffusion models that are in pixel space are computationally intensive because they are iterative models that do their high-dimensional calculations in RGB space. To solve this challenge, other domains are utilized to achieve lower computational requirements. The other domains are frequency space, latent space and residual space and to utilize them an extra step is needed: first, images are mapped from pixel domain to the desired domain such as latent space and then the images are given to a diffusion model. [10]

The point of using the latent domain is to ease sampling and training costs [1] by reducing the dimensionality and fitting data to approximate a specific data distribution [40]. Because of latent space, some diffusion models, such as Stable diffusion [36], DiffIR [41] and SeeSR, [1] have achieved multiple advantages that vary between each diffusion model. Such advantages are, for example, faster execution, comparable or better results than in pixel space and lower resource requirements with the same performance. [10]

A common way for transferring data into latent space is to use an autoencoder such as VAE [10]. Otherwise, latent space can be achieved through methods like compressing an input image or training a latent diffusion model such as a transformer-based model that generates compact priors to guide another model [1].

Variational autoencoder (VAE) is a likelihood-based generative model that performs probability density estimation. The idea of the model is to generate new data based on the distribution of the input data. [40] VAE has an encoder-decoder structure consisting of Bayesian networks where the encoder and decoder can be called recognition model and generative model, respectively. Between them lies the probabilistic latent space that consists of the input data distributions. [42] See figure 5.

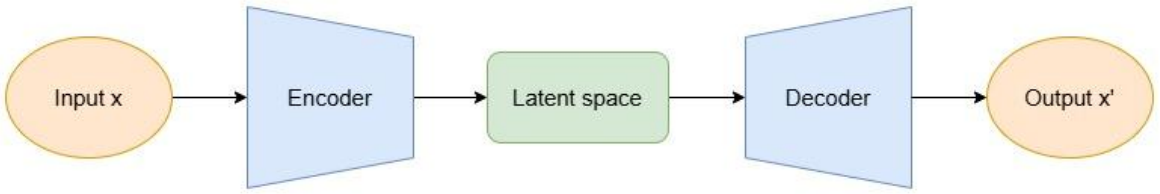


Figure 6. The architecture of VAE. [40]

Encoder converts the input data into the latent space from another space [42]. While doing this, the encoder generates parameters that are used in Gaussian distribution in the latent space [40]. After the conversion, the decoder attempts to recreate the input data from the latent version (distribution) of the data by converting the encoded data back into the input space. [42] Decoder utilizes the Gaussian distribution for output generation by approximating the parameters that describe the probability distribution of the input data [40].

When changing the distribution in latent space the output of the VAE can be modified to match the desired properties. As an example, the model could take a crying human face image as an input, encode it to the latent space, change the variables that affect to the expression of the face to get smiling face and then generate the smiling face image. [42]

Latent space offers couple of advantages to VAEs in general. In generative models like VAEs latent space enables dimensionality reduction, examination of the data in the latent condition and therefore gaining better understanding of the data. [42] The dimensionality reduction allows efficient data compression [40].

VAEs have turned out to be a good platform to build upon. They have been extended to contain attention or multiple levels of stochastic latent variables, and some dynamical models are built based on them. VAEs have been utilized in different fields, such as in image (re-)synthesis, NLP or even in astronomy. [42] In this thesis a VAE is used in the SeeSR diffusion model to convert the input data into latent space.

Utilization of latent domain flows as follows: The starting point is the input space such as a pixel space from where the data is transformed into latent representation and then given to the diffusion model. In the model the data is transformed from latent space to corruption space and then back into the latent space. After the model the latent result is transformed back into

the input space. [10] See figure 7.

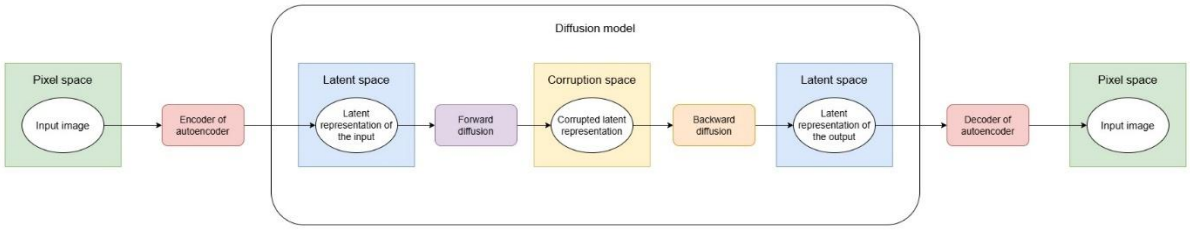


Figure 7. The architecture of a latent diffusion model. [1]

### 2.4.7 Challenges of Diffusion Models

Diffusion models are a very effective way to solve many tasks, but they have multiple challenges that have yet to be solved. They are difficult to develop, and one may have hard time following their development process because the related benchmarks and tasks are complicated and diverse [1]. Also, the number of publications grows all the time making it hard to keep up with the latest developments [10]. This sets a great challenge for beginners in the field and complicates the identification of possible research directions and trends [10]. In addition, the significant computational costs and large amount of paired data needed to train the supervised diffusion model-based image restoration can be exorbitant for some researchers [1].

#### *2.4.7.1 Computational cost*

Computational costs are very high in most of the diffusion-based image restoration models because these methods need a significant amount of time steps, usually a thousand, for high-quality output image generation [9]. For example, DDPMs [43] need 113.7 million and SR3 [31] needs 155.3 million parameters [1]. Because of the computational costs, diffusion models are challenging to implement into different real-world applications [9]. The huge size of diffusion models exacerbates the existing computational demands of deep learning [29].

There are different ways to try to ease these challenges. Efficient sampling techniques and latent diffusion are possible solutions, but they don’t fit every image restoration task because efficient sampling decreases the quality of image generation, and latent diffusion often causes colour shifting. Another potential solution is to design the diffusion process more towards degraded images by starting the process from low-quality images and not from Gaussian noise. This increases the performance of the sampling process to a point where under a

hundred diffusion steps are usually needed. The design can be improved, for example, with more effective design of SDE. [9]

Colour shifting is a challenge that may occur because of the high computational cost. This is a problem especially at the cases where hardware is constrained. The constrictions lead to the usage of shorter training periods and smaller batch sizes which causes the colour shifting effect. Colour normalization is performed in the StableSR [33] model by adjusting the variance and mean of a generated image with the mean and variance of the corresponding low-resolution image. Also, properly defined diffusion design and architecture are suggested as counters to colour shifting. [10]

#### *2.4.7.2 Data*

Many different diffusion model versions need clean-distorted image pairs for training. Images are typically distorted artificially, so it isn't an authentic distortion coming from the real world. The real-world distortions are usually unknown and diverse and, therefore, hard to mimic synthetically. However, the blind image restoration discussed in chapter 2.4.2 might be a solution to this problem. [1]

Blind image restoration doesn't guarantee success because its challenges cause synthetically distorted data that often generalize poorly. Because real-world distortions are unknown, desired degradation modes are hard to recognize. Also, it is extremely hard or even impossible to collect image pairs from the real-world. Earlier, these problems have been tried to solve with methods like unsupervised learning or simulating real-world degradations, but diffusion models require different methods, such as kernel estimation and domain translation. [1] More on the different methods and their descriptions can be read from [1].

Diffusion models need a lot of data samples. There exists enough 2D image data for image processing, but some tasks in visual computing don't have enough data. For example, diverse and high-quality scene data and 3D object data aren't available in large enough quantities for training diffusion models. [1] Also, the scaling of the 2D image models for larger dimensions is a relevant challenge [29] not to mention the requirements of some models regarding the fixed resolution of images during training as demanded by SR3 [1].

#### *2.4.7.3 Backward process*

Sampling efficiency consists of strategies for generating samples based on noise with less time steps without compromising output image quality [10]. Generation quality depends on a huge number of sampling steps during backward diffusion which causes challenges to the efficiency of model applications in practice [1]. For example, to sample 50 000 images with 32x32 resolution using Nvidia 2080 Ti GPU DDPM needs about 20 hours. GAN does the same in under a minute. For 256x256 resolution images with otherwise same conditions it takes about 1 000 hours using DDPM. [10] Different ways to improve sampling efficiency have been suggested such as handcrafted sampling strategies, but the problem is still unsolved [1].

In the context of super-resolution training-free sampling and training-based sampling can be applied as sampling strategies. Training-free sampling methods make sampling quicker by minimizing the number of time steps while calculating the Probability Flow Ordinary Differential Equation (ODE) or SDE. Training-based sampling methods use a sampler for generating data quicker and they do it in the backward process instead of a typical numerical solver. The described process can be partial or complete depending on the application. [10]

Even though diffusion models produce photorealistic results the backward process includes some inconsistencies between output and input images of the diffusion models. This concerns especially text information and texture generation, and it is caused by the stochasticity of noise injection occurring in each iteration step and by an intrinsic bias in multistep noise or score estimation. A couple of solutions have been proposed to solve this challenge. One of them is to attach a new predictor at the beginning of a diffusion pipeline so that the initial high-quality images can be generated by it. For generation L1 loss is used and then more details are added gradually by utilizing the diffusion model. However, to do this an additional network must be trained and utilized causing the performance of the diffusion model to be highly dependent of the new network. Another solution is to use optimal transport or flow matching to create straight-line trajectories in inference. These straight-line trajectories have been proven to be much more efficient than the curved paths that diffusion models use normally. [9]

#### *2.4.7.4 Unstable results and out-of-distribution degradations*

Diffusion models may generate unstable results when compared to the ground truth or input image. The instability is caused by visual blurriness or over-enhancement of the image details or the details being unfaithful to the input image. In super-resolution models this affects the content consistency and fidelity of the model results directly. This challenge can be, at least partially, countered by using less iteration steps for generation, but that causes degradation of visual generation performance. [6]

The processing of OOD degradations is difficult and a diffusion model applying the OOD data has typically low or average performance. Its outputs also contain visually unappealing artefacts. Stable diffusion that has a feature control module in it has been suggested as a solution to this challenge, but it must be refined with a specific dataset to suit to the desired image restoration task. [9]

Generally, synthetic data simulates known degradations, such as blur, noise and compression, and it can't cover all the corruption types occurring in the real-world applications at the same time [9]. Therefore, there have been attempts to develop blind image restoration to answer the problem, but diffusion models don't usually support blind image restoration because they are built for solving synthetic data distortions. [1]

The recent approaches that are inspired by the success of vision-language models and large language models have explored solutions for the OOD and blind image restoration tasks by utilizing an image restoration with language-based representations of images. The idea functions as follows: first a clean textual description about a low-quality input image is created and then it is used to guide the image restoration process. The description should describe the content in the image without any unwanted degradation concepts. [9]

As stated earlier, diffusion models are mostly applied to one or a few specific degradations in the image restoration aspect and this success has happened fast. When these models are applied to distortion types and degrees that are unseen for that model, they usually suffer from unsatisfactory robustness in addition to the previously mentioned challenges. This challenge brings us a relevant question about a consistent image restoration model that would work with diverse distortion levels and types. [1]

## 2.5 Models for experiment

In this section, the diffusion models that are necessary to the experiment of this thesis are discussed. The diffusion model part of the experiment is based heavily on the SeeSR model, so it has been described the most extensively and the other models are discussed to support SeeSR or the experiment. First, text-to-image models and Stable diffusion are covered, then ControlNet and SeeSR. Finally, the CCSR model is discussed.

### 2.5.1 Text-to-Image Diffusion Models and Stable Diffusion

Text-to-image diffusion models are diffusion models that produce images from text prompts. Because of the huge development of these language models, they have become popular models in other tasks as well. [29] By utilizing their text-to-image information together with the information typically required for a task, pretrained text-to-image models can be used as a base model for other tasks. For example, in super-resolution low-resolution images as well as text-to-image information can be utilized as conditions. [10]

The usage and customization of text-to-image models to different tasks have been explored in recent years in hopes of achieving better generation outputs in the tasks. These models can be utilised, for example, by optimizing the text-to-image network weights or fine-tuning parts of the text-to-image network. [29] The fine-tuning can be done by adding some encoders or layers specified to the desired task. These additional parts guide the generation process by integrating text prompts to the process which may provide benefits like improvement of the interpretation and synthesis of images in the super-resolution task. [10]

Stable diffusion [44], [36], is a popular text-to-image diffusion model that functions in latent space. In fact, it is an implementation of latent diffusion models (LDMs). [44] LDMs [36] are diffusion models that use latent space instead of image space by utilizing an encoder-decoder architecture. The diffusion model itself consists of an autoencoder and a denoising U-Net that deploys cross-attention in multiple places. Text and other conditions are applied by the conditioning module. [36] See figure 8. Stable diffusion v2 is used as a base in the SeeSR model. More on the Stable diffusion can be read from [44] and [36].

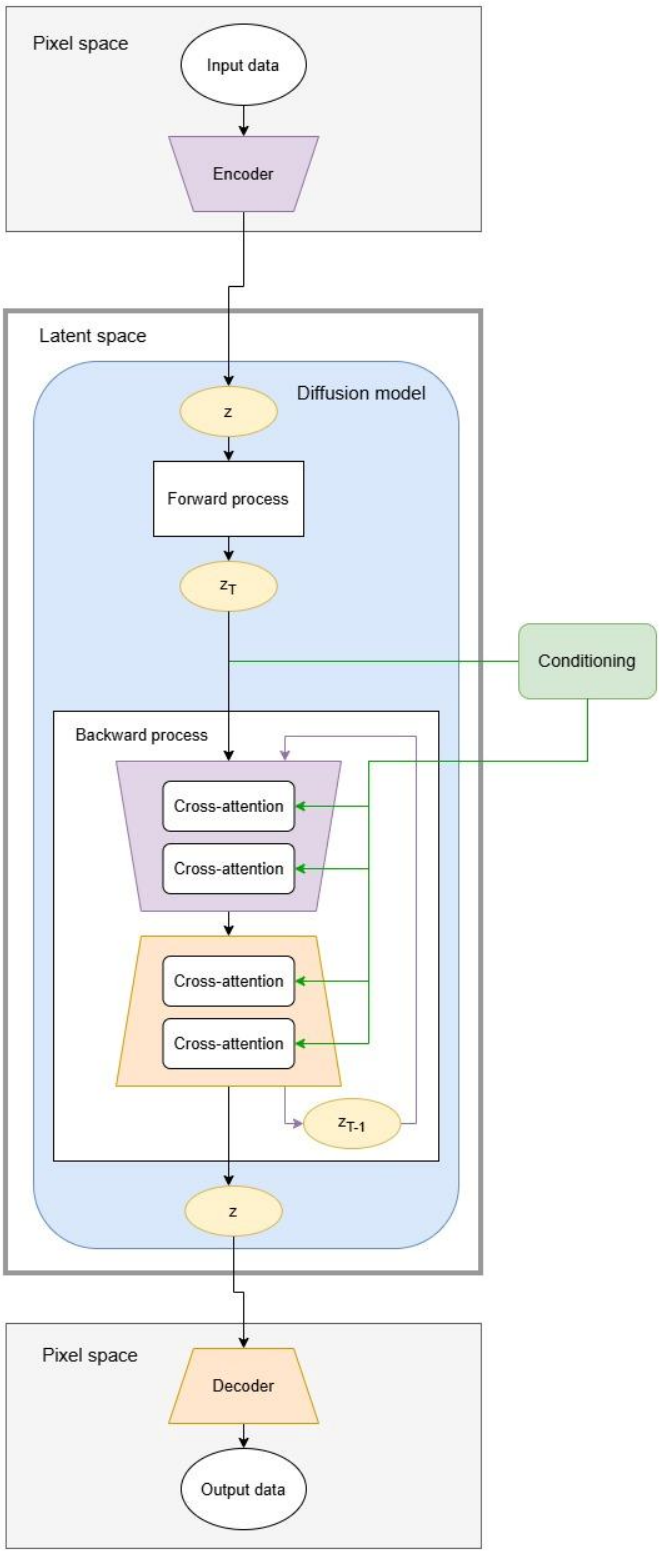


Figure 8. The architecture of LDM. [36]

### 2.5.2 ControlNet

ControlNet [27] is a deep learning architecture that brings conditioning for controlling large pretrained text-to-image diffusion models such as Stable diffusion. This supports the usage of a combination of conditions in diffusion models. These additional conditions are injected to the text-to-image model by ControlNet that can be scaled to different sized datasets. [27]

In creation of ControlNet a pretrained text-to-image model is frozen, and its encoder is copied concurrently to function as a trainable ControlNet. Copying and freezing of the text-to-image model enables ControlNet to retain the capabilities and quality of the text-to-image model. The text-to-image model needs to be robust, strong and deep for ensuring the quality of ControlNet. The encoder backbone is the one that learns the outputted diverse conditional controls of ControlNet. [27]

The copied encoder is connected to the frozen text-to-image model with a zero-initialized convolutional layer, also called zero-convolution, that is 1x1 convolution layer. Another zero-convolution layer is before the copied encoder. The parameters of these layers, weights and biases, are initialized as zeros. However, these parameters change during the training of ControlNet. Through this parameter initialization, it is guaranteed that harmful noise won't affect the training of ControlNet or the features of the backbone at the start of the training. The outputs of ControlNet are given to the decoder of the text-to-image model. [27] See figure 9.

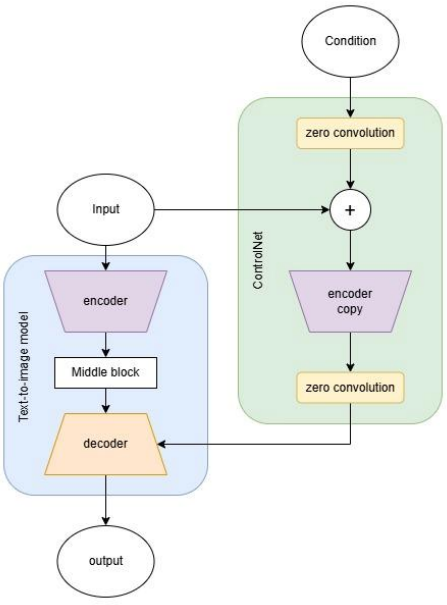


Figure 9. The architecture of ControlNet and its interaction with a text-to-image model. [27]

ControlNet architecture takes an input conditioning vector c which goes through a zero-convolution layer $Z(\cdot;\cdot)$. Then the result is combined with the input $z_t$ of the text-to-image model just before it is fed to the copy of the encoder of the text-to-image model. After the encoder copy comes another zero-convolution layer. The parameters of the first and second instances of zero-convolution layers are $\Theta_{z1}$ for first layer and $\Theta_{z2}$ for second layer. The output of ControlNet is

$$\boldsymbol{y}_c = \mathcal{F}(\boldsymbol{z}_t;\, \Theta) + \mathcal{Z}(\mathcal{F}(\boldsymbol{z}_t + \mathcal{Z}(c;\, \Theta_{z1});\, \Theta_c);\, \Theta_{z2}) \tag{13}$$

where $y_c$ means the output, $\Theta$ means the parameters of the text-to-image model and $\Theta_c$ means the parameters of the encoder copy. [27] Because the parameter values of both zero-convolutions are zero at the beginning, the output of the first training step is

$$\boldsymbol{y}_c = \boldsymbol{y}. \text{ [27]} \tag{14}$$

More information about ControlNet in Stable diffusion can be found from [27].

### 2.5.3 Semantic-Aware SR

In super-resolution the input images typically have low-resolution. If those images have been through a huge quality degradation, the local structures of the images have been destroyed. The destruction can cause ambiguous image semantics which can lead to semantic errors in high-resolution images. In that case the performance of a super-resolution model has been weakened. The Semantic-Aware SR (SeeSR) addresses the problem in a semantics-aware way that preserves better semantic fidelity occurring in Real-ISR. [5]

Real-ISR is super-resolution with distorted image pairs where one image has high-resolution, and the other one has low-resolution with real-world distortions [5]. Essentially, it is the same problem as blind image restoration but in a super-resolution context. Examples of Real-ISR methods are SeeSR, SR3+ [12] and CDFormer [45].

Stable diffusion has been utilized in SeeSR as a noise predictor and the tag-style has been chosen to be text prompt style. Each tag prompt has one object class, and one image can have multiple tag prompts. This causes the descriptions of images to be more detailed than in the caption-style where a prompt is a description describing an image with one sentence. Tag prompts don't offer information about the location of objects in an image. However, that doesn't affect a text-to-image model since it can couple the class prompts and their corresponding locations because of its semantic segmentation capability. [5]

The tag-style models are vulnerable to semantic distortion and erroneous semantic cues in the results which stem from image degradation. The researchers of SeeSR noticed that if they can make the prompts of the tag-style degradation aware it could help the text-to-image model to generate high-quality Real-ISR results with correct image semantics. [5]

So, the idea of the SeeSR model is, in short, to extract high-quality tag-style prompts and then train a super-resolution diffusion model that utilizes the prompts as conditions. First, the degradation-aware prompt extractor (DAPE) is trained to extract soft and hard prompts from the low-resolution images that contain accurate semantic information regardless of the strength of the degradation in the image. After training the purpose of DAPE is to produce the soft and hard tags to the pretrained text-to-image model as conditions from the low-resolution images. [5]

The DAPE consists of two components: tagging head and image encoder. See figure 10. The tagging head produces tags as outputs that are called hard (semantic) prompts. Their point is to enhance the local perception capability of the text-to-image model. The image encoder produces feature representations called soft (semantic) prompts that compensate the hard

prompts by offering additional representation information. Both prompts support the semantically accurate and detailed results. [5]

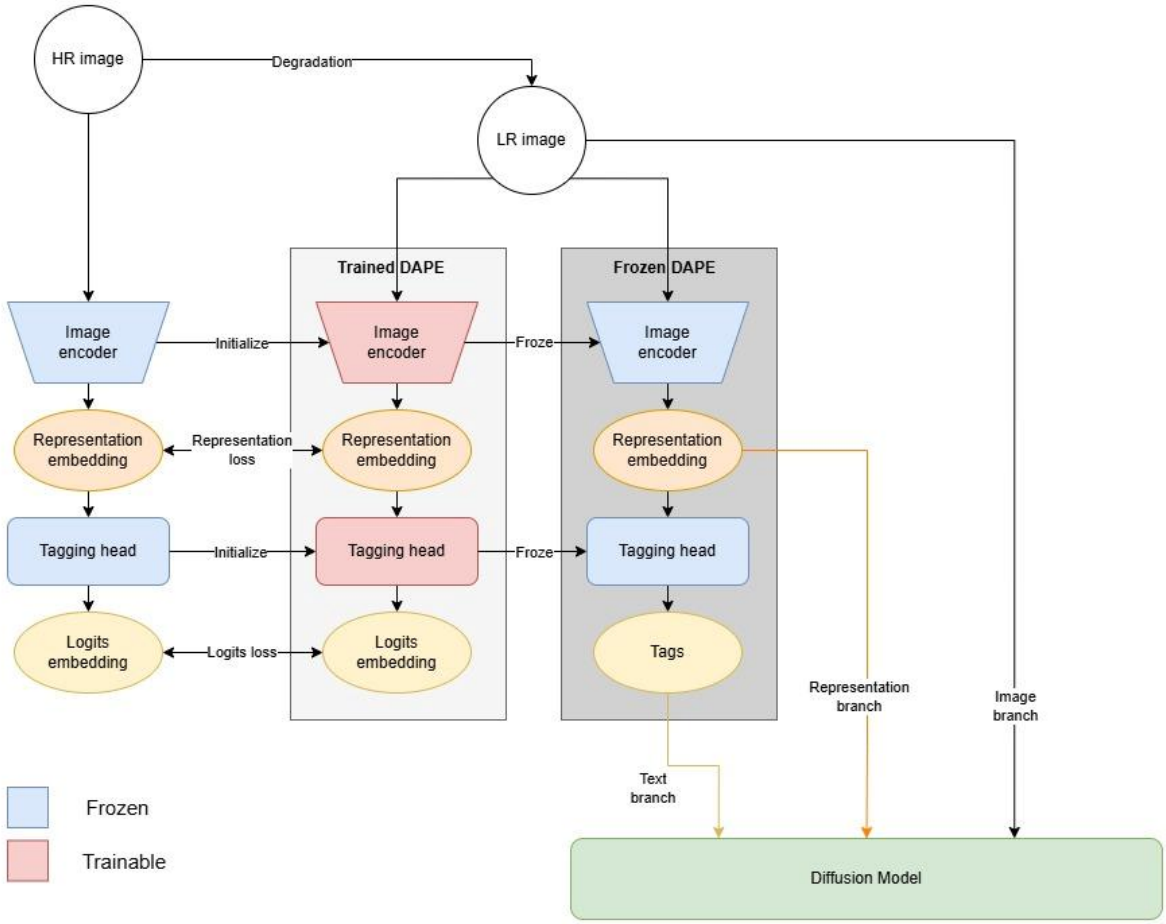


Figure 10. The creation of DAPE. [5]

To be exact the DAPE is a fine-tuned version of a tagging model called Recognize anything model (RAM) [5]. Ram is a transformer based zero-shot model that takes images, tags and texts as inputs to produce output tags. The architecture of the model consists of a text generation encoder-decoder structure, an image-tag recognition decoder and an image encoder. RAM has learned almost 6 500 common labels for tagging and can produce more tags with open-vocabulary recognition with high accuracy. [46]

The training of DAPE flows in the following way: A high-resolution image $x$ is fed through frozen RAM to get the representation embeddings, feature embeddings, $f_x^{rep}$ and the logits embeddings, tags, $f_x^{logits}$ as outputs to guide the training. Then the low-resolution images $y$ are achieved by applying random degradations to the high-resolution images. Next, those outputs are used to train this RAM version. The outputs from the RAM version, where low-

resolution images are used, are forced to be very similar to the outputs of frozen RAM using the high-resolution images. This ensures the image degradation robust DAPE. [5]

The branch with low-resolution images is now called low-resolution branch and the branch with high-resolution images is called the high-resolution branch. By aligning the outputs of the branches, RAM, which can now be referred to as DAPE, learns how to predict high-quality semantic prompts from the low-resolution images. The training objective of the DAPE is

$$\mathcal{L}_{DAPE} = \mathcal{L}_r\left(f_y^{rep}, f_x^{rep}\right) + \lambda\mathcal{L}_l(f_y^{logits}, f_x^{logits}) \quad (15)$$

where $\mathcal{L}_r$ is the mean squared error, $f_x^{rep}$ is the representation embedding from low-resolution branch, $\lambda$ is a balance parameter, $\mathcal{L}_l$ is the cross-entropy loss and $f_x^{logits}$ is the logits embedding from low-resolution branch. [5]

The number of hard prompts is controlled by a preset threshold. However, with too high threshold the recall rate of the model may not hold true, but the accuracy of the predicted classes increases. The soft prompts aren't controlled with the threshold. [5]

When the learning of DAPE is over the diffusion model part of SeeSR takes care of the reliable semantic prompts. See figure 10 above. The soft prompts of DAPE are passed straight to the frozen encoder and decoder parts of the Stable diffusion U-Net and to the trainable ControlNet. The hard prompts are first fed to the frozen text encoder of the Stable diffusion model from where they are passed to the same parts of the SeeSR model as the soft prompts. [5]

Inside the encoder and decoder of the Stable diffusion U-Net are text cross-attention (TCA) modules; one per each coder. TCA is a cross-attention module that guides the model's attention regarding text. An attention module is needed for soft prompts as well, so the researchers decided to implement the cross-attention module of the PASD model [47] for guiding semantics. These representation cross-attention (RCA) modules are added straight after the TCA modules in the Stable diffusion model, and they are initialized randomly. [5]

Before the ControlNet model comes an image encoder that takes low-resolution images as input and transforms them to the latent space. These latent versions are then passed to ControlNet. The structure of the image encoder isn't altered. ControlNet is a copy of the encoder of the pretrained Stable diffusion U-Net with the modifications that have been done

to the Stable diffusion discussed earlier. The outputs of ControlNet are used as a conditioning for SeeSR together with the text and representation branch. The flow of low-resolution images through the image encoder and ControlNet is called the image branch. [5]

Finally, there is a decoder of a pretrained VAE at the end of the SeeSR model. It transforms the generated image from latent space back to image space. The text encoder, U-Net and VAE decoder are frozen during the training and the image encoder, the ControlNet and RCA modules in the U-Net are trainable. The Stable diffusion model is frozen to avoid huge training cost it causes. [5] See figure 11 for the architecture of SeeSR.

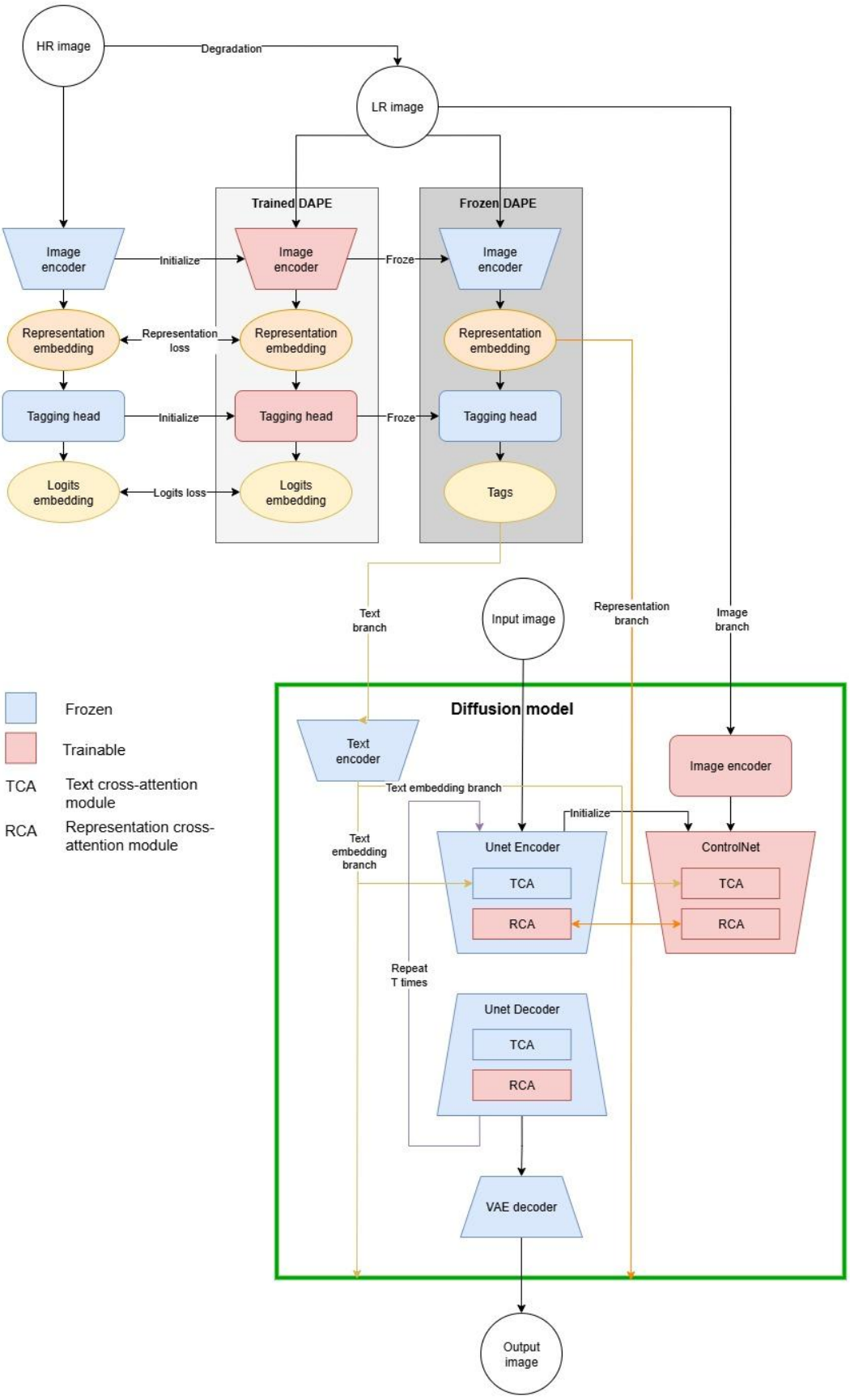


Figure 11. The architecture of SeeSR. [5]

The training of SeeSR goes as follows. A high-resolution image is fed into the pretrained VAE encoder to get the latent representation $z_0$. In the forward process noise is added to the latent representation $z_0$ to create a noisy latent $z_t$. Estimation of the noise in the noisy latent $z_t$ is the training objective of SeeSR and the model's optimization objective is

$$\mathcal{L} = \mathbb{E}_{\mathbf{z}_0,\mathbf{z}_{lr},t,p_h,p_s,\epsilon\sim N}\left[\|\epsilon - \epsilon_\theta(\boldsymbol{z}_t,\boldsymbol{z}_{lr},t,p_h,p_s)\|_2^2\right] \quad (16)$$

where $\epsilon_\theta$ is the SeeSR network, $z_t$ is the noisy latent, $z_{lr}$ is the latent low-resolution image, $t$ is the diffusion step (i.e. time step), $p_h$ are the hard prompts and $p_s$ are the soft prompts. [5]

There still exists one limitation in the backward process after the training of SeeSR. The pretrained text-to-image model don't transform an image completely into Gaussian noise but the starting point of the backward process of the Stable diffusion-based Real-ISR models is usually random Gaussian noise. This inconsistency between inference and training may lead to a situation where a diffusion model believes that the degradation in the images must be enhanced. Especially smooth areas in the images such as sky are prone to enhancement. [5]

Therefore, the SeeSR model embeds the low-resolution images straight to the initial random Gaussian noise to avoid inconsistency. The researchers call this strategy low-resolution embedding (LRE) strategy. This strategy relieves the inconsistency by reducing artifacts in smoothed areas, and it offers more reliable starting points to SeeSR and most of the Stable diffusion-based Real-ISR models. [5]

The researchers of SeeSR decided to concentrate on the x4 Real-ISR task such as the Real-ESRGAN [48] which means that inputs are four times of their original size when they are fed to the diffusion model. Other scaling factors could have been used as well. Next, the training and evaluation of SeeSR are opened from the practical view and it has been compared to other Real-ISR methods in terms of capability. [5]

SeeSR has been trained with two datasets: LSDIR and a dataset that consists of 10 000 first face images of FFHQ. The degradation pipeline of Real-ESRGAN is utilized to synthesize the low-resolution and high-resolution image pairs for training. DAPE is fine-tuned with LORA ($r$ = 8) method, and it takes 20 000 iterations. The batch size is 32 and the learning rate is 10-4. The controlled Stable diffusion 2-based model is trained with 150 000 iterations using Adam optimizer, batch size of 192, learning rate of 5x10-5, 512x512 resolution images as input and 8 NVIDIA Tesla 32G-V 100 GPUs. The spaced DDPM sampling, 50-time steps and lambda 1 are used in inference. [5]

The evaluation of SeeSR uses four different datasets. One of them is DIV2K-Val that is produced based on the DIV2K validation set. First the validation set is cropped so that the result is 3 000 patches of images with 512x512 resolution. Then the patches are degraded with the same pipeline that is used in training, Real-ESRGAN degradation pipeline, and now the DIV2K-Val dataset is ready. DRealSR and RealSR are real-world datasets containing low-resolution images that are cropped to 128x128 resolution. RealLR200 is made by the researchers entirely. It contains 200 low-resolution images split as follows: 47 images from DiffBIR [49], 65 images collected from the Internet by the researchers, 38 images from recent literature and 50 images from the VideoLQ dataset. The images from the VideoLQ dataset are the last frames of the videos in the dataset. [5]

For evaluation purposes nine reference and no-reference metrics are used. The reference metrics are PSNR and SSIM which are calculated on the Y channel in YCbCr space, DISTS and LPIPS. The no-reference metrics are MUSIQ, NIQE, CLIPIQA and MANIQA. The FID metric belongs to neither of the groups. The SeeSR model is compared to publicly available state-of-the-art Real-ISR methods that can be divided into recent diffusion-based methods and GAN-based methods. The diffusion methods are StableSR [33], PASD [47], LDM, DiffBIR and ResShift [50]. The GAN methods are LDL [51], DASR [52], Real-ESRGAN, BSRGAN [53] and FeMaSR [54]. All methods aren’t necessarily compared to SeeSR in all experiments. [5]

SeeSR has been evaluated and compared in four different ways that are quantitative and qualitative comparisons, user study and testing the semantic preservation of the model. The quantitative comparison is done with the aforementioned metrics, and the qualitative comparison is done by looking at the generated images and comparing them to each model and to the ground truth. The user study is about giving ground truth and generated images to test participants to examine, and the semantic preservation test is done by giving the generated images to the pretrained OpenSeeD model that functions as segmentator and detector. SeeSR surpasses some of the other models or all models in every test showing strong semantics preservation capability and ability to produce richer details and more accurate semantics than other models. [5]

SeeSR has some limitations. DAPE may predict wrong tags if the used images are highly degraded which leads to incorrectly restored objects. The alignment between areas and tags of the low-resolution images may be inaccurate in grave degradation cases, but this might be

relieved by using an additional mask to gain extra information. It is challenging for the model to reconstruct small-scale scene text images as it is a general problem for methods based on Stable diffusion. [5]

### 2.5.4 Content Consistent Super-Resolution

Content consistent super-resolution (CCSR) [6], [55], is a diffusion model that is based on ControlNet, SeeSR, BasicSR [56] and ADDSR [57]. [55] It utilizes a diffusion model, that is built on Stable diffusion and ControlNet, as its base and a GAN model in the decoder of the latent VAE for generating super-resolution outputs. This division is based on the observations of the researchers of the CCSR paper [6]. They noticed that diffusion models are powerful tools for reproducing object and image structures present in the structures of low-resolution input images. The researchers also noticed that GANs are better for producing texture and details for the generated images than diffusion models. This is the case, especially, when the main structures of the output image have been already generated, so the GAN can only focus on producing synthetic and fine-grained details to it in an efficient way. Therefore, a GAN model is used for texture and detail enhancement in CCSR. [6]

The first stage of the CCSR model applies a non-uniform time step sampling strategy that is used for information extraction and structure generation based on low-resolution input images to enhance fidelity and stability. See figure 12. The stage begins with a single step sampling that extracts some coarse information from the given inputs. After that may come a couple of reverse time steps that reconstruct the main structures of the output, but they are optional. These additional steps can be skipped especially when efficiency of the model is needed. This is possible because one step can produce most of the generated low-frequency information that is acquired from a low-resolution image. At the end of the first stage truncation is done for the intermediate diffusion steps to make it possible to them to evade the details that aren’t similar to the details in low-resolution images. [6]

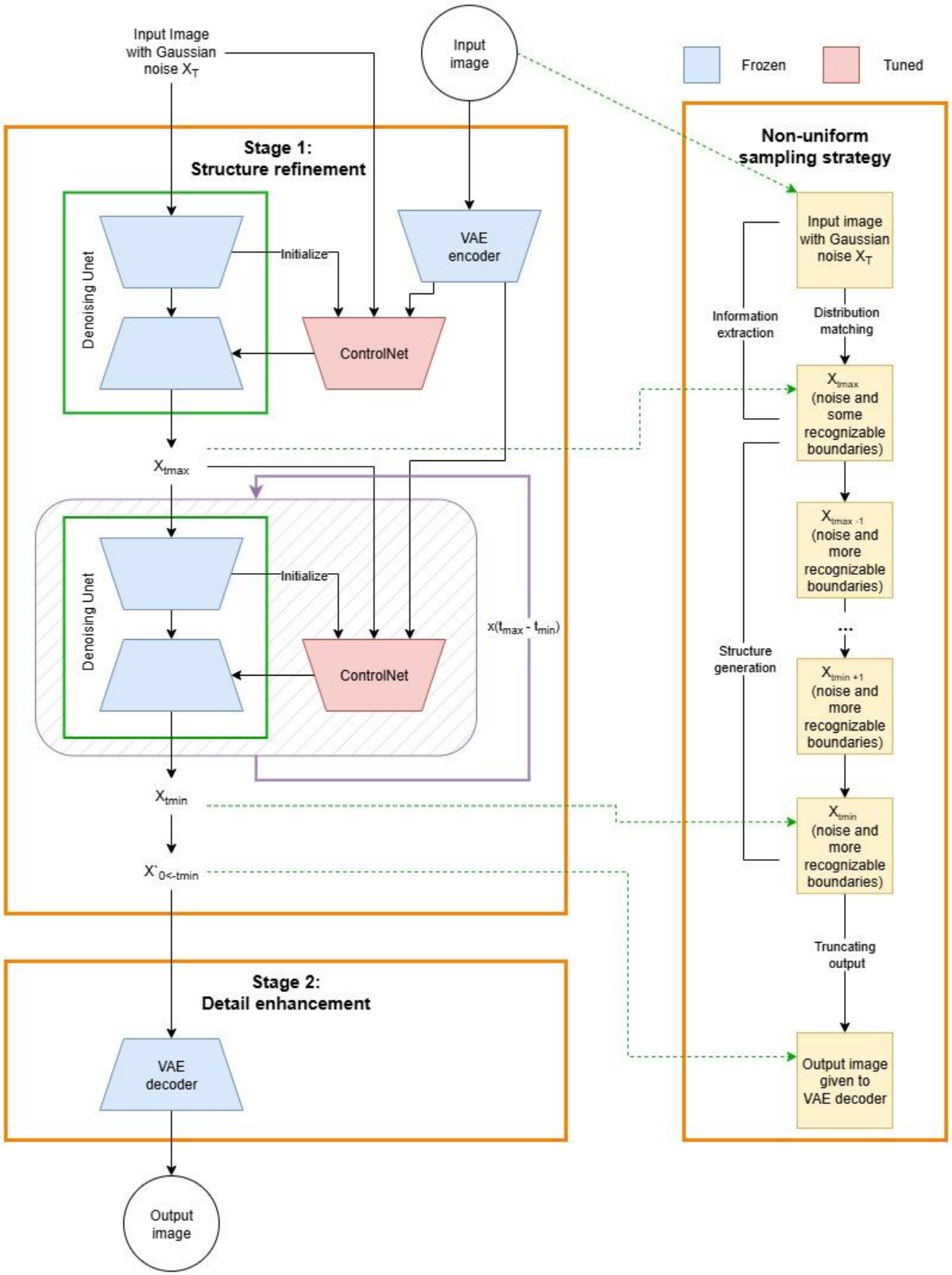


Figure 12. The architecture of CCSR. [6]

At the second stage of the CCSR a pretrained VAE decoder is fine-tuned using the adversarial loss of the GAN to achieve deterministic detail enhancement. It gets the output of the

previous stage as its input. The fine-tuned VAE decoder does the decoding of the latent features at the same time it does the detail enhancement. This causes no additional computational costs. More specifically, training loss of the fine-tuned decoder is the same as it is in the original VAE decoder. [6]

CCSR allows usage of different diffusion steps in the backward process without any new or follow-up training. So, single step or multiple step diffusion can be used for generation of high-resolution images because the model does single-timestep and multi-timestep sampling at the same time. This makes it possible for users to use different diverse perception-distortion balances depending on their preferences which balances the generation capacity and efficiency. CCSR enhances the visual quality and content consistency present in the super-resolution outputs. [6]

## 2.6 Evaluation of Diffusion and CNN models

Assessment of image quality isn't an easy task due to the wide range of different aspects associated with quality, like contrast, sharpness and noise [10]. Therefore, effectively and robustly assessing the quality of images generated by diffusion models requires more specialized evaluation metrics [29]. Image Quality Assessment (IQA) metrics are an answer to the problem although they aren't perfect. IQA metrics are all metrics that try to evaluate generated images that are close to realistic images. In addition, those metrics need to resemble perceptual evaluations made by humans. [10]

IQA metrics can be divided into objective and subjective methods [29] or reference and no-reference methods [10]. Objective methods use fair and simple formulations for computations. They numerically measure pixel recovery in images, but the actual visual effects in the images aren't easy to measure. Subjective methods evaluate the perceptual quality of the generated images, and their assessment is based on judgements done by humans. [11] The reference methods need a reference image in addition to a generated image so that they can compare the images against each other. No-reference methods don't need the reference image and are based on other techniques. [10]

In image restoration context Peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), Fréchet Inception Distance (FID), Learned Perceptual Image Patch Similarity (LPIPS) and Natural Image Quality Evaluator (NIQE) are commonly used metrics [1]. In this thesis PSNR, SSIM, FID and NIQE are used to assess the modified SeeSR. Next,

they are discussed in the respective order along with common model evaluation metrics that will be utilized in the evaluation of the CNN classifier.

### 2.6.1 Peak Signal-to-Noise Ratio

The Peak Signal-to-Noise Ratio (PSNR) is one of the most popular IQA methods and one of the most used metrics in SISR [10]. Some papers claim it to be the most popular metric in image restoration [1] or in SISR [11]. PSNR measures peak-signal-to-noise-ratio which is computed by calculating the pixel-wise distance between the ground truth image $\boldsymbol{y}$ and the distorted image $\hat{\boldsymbol{y}}$ using the Mean Square Error (MSE) method [1] and taking a ratio between MSE and the maximum pixel value L:

$$PSNR(\boldsymbol{y}, \hat{y}) = 10 * \log_{10}\left(\frac{L^2}{\frac{1}{N}\sum_{i=1}^{N}[\boldsymbol{y} - \hat{y}]^2}\right). \quad (17)$$

In super-resolution the images are a high-resolution image and a super-resolution image, respectively. [10]

Despite the huge popularity of PSNR it may not perform well in human perceptual manner because it focuses on computing MSE pixel-wise [11]. Therefore, if the pixels are moved even a little bit PSNR result may be worse. However, this movement doesn't influence human perception. Because of the pixel-wise calculations, the models that are trained on loss based on pixels achieve usually higher PSNR results. [10]

### 2.6.2 Structural Similary Index Measure

The Structural Similarity Index Measure (SSIM) is a traditional and popular IQA metric. The popularity of SSIM is based on its stable performance and conveniency when assessing perceptual quality of images [11] and its relatively quick computational speed. The metric evaluates image quality from a human perceptual view. [1] To do that, it examines three structural features of images: contrast, luminance and structures [10]. SSIM compares the structural similarity of ground truth and distorted images by comparing these three features. The perceptual information about the structures of an image is gained from the interdependent pixels that lay spatially next to each other. [11]

So, SSIM consists of three comparative measures that are weighted and combined to form the metric [11]. The luminance $\mu$ is estimated as a mean of intensity and the contrast $\sigma$ is estimated as the standard deviation of luminance. For an image $\boldsymbol{y}$, the luminance $\mu_y$ is

$$\mu_{\boldsymbol{y}} = \frac{1}{N_y}\sum_{p\in\Omega_y} \boldsymbol{y}_p \tag{18}$$

and contrast $\sigma_y$ is

$$\sigma_{\boldsymbol{y}} = \frac{1}{N_y-1}\sum_{p\in\Omega_y}[\boldsymbol{y}_p - \mu_{\boldsymbol{y}}]^2. \tag{19}$$

Because similarity of different similarity features needs to be computed, a comparison function S is used:

$$S(x, y, c) = \frac{2x\boldsymbol{y}+c}{x^2+\boldsymbol{y}^2+c}. \text{ [10]} \tag{20}$$

In the function the $x$ and $y$ are scalar variables, and $c$ is a constant for numerical stability. The $x$ and $y$ are compared, and $c$ is $c = (k \cdot L)^2, 0 < k \ll 1$ to be precise. The comparisons of high-resolution image $y$ and the approximated version of it, $\hat{y}$, for contrast ($C_c$) and luminance ($C_l$) are $C_c(\boldsymbol{y}, \hat{\mathrm{y}}) = S(\sigma_{\boldsymbol{y}}, \sigma_{\hat{\mathrm{y}}}, c_2)$ and $C_l(\boldsymbol{y}, \hat{\mathrm{y}}) = S(\mu_{\boldsymbol{y}}, \mu_{\hat{\mathrm{y}}}, c_2)$ where $c_1, c_2 > 0$. The structure comparison ($C_s$) can be achieved from an empirical covariance $\sigma_{\boldsymbol{y},\hat{\mathrm{y}}}$ :

$$\sigma_{\boldsymbol{y},\hat{\mathrm{y}}} = \frac{1}{N_y-1}\sum_{p\in\Omega_y}(\boldsymbol{y}_p - \mu_{\boldsymbol{y}}) * (\hat{\mathrm{y}}_p - \mu_{\hat{\mathrm{y}}}) \tag{21}$$

$$C_s(\boldsymbol{y}, \hat{\mathrm{y}}) = \frac{\sigma_{\boldsymbol{y},\hat{\mathrm{y}}}+c_3}{\sigma_{\boldsymbol{y}}\cdot\sigma_{\hat{\mathrm{y}}}+c_3} \tag{22}$$

where the $C_s$ appears to be a correlation coefficient between the images $y$ and $\hat{y}$ and $c_3 > 0$. [12] By utilizing all three comparison formulas, the SSIM can be defined as

$$SSIM(\boldsymbol{y}, \hat{\mathrm{y}}) = [C_l(\boldsymbol{y}, \hat{\mathrm{y}})]^{\alpha} * [C_c(\boldsymbol{y}, \hat{\mathrm{y}})]^{\beta} * [C_s(\boldsymbol{y}, \hat{\mathrm{y}})]^{\gamma} \tag{23}$$

where parameters $\alpha, \beta, \gamma > 0$ are tunable for the purpose of defining the importance of each component. [12] The bigger values SSIM produces the bigger structural similarity exists between a ground truth image and a distorted image [11].

### 2.6.3 Fréchet Inception Distance

The Fréchet Inception Distance (FID) metric is used to assess the quality of generated images. It was created to be an improvement of Inception Score (IS) to assess the performance of GANs but has been since used for evaluation of other types of models as well. [58] In diffusion models it measures the fidelity and diversity of images [29].

To compute FID a pretrained CNN, usually Inception v3, is utilized to produce feature vectors. The input size is 299x299 by default for Inception v3. A feature vector is produced by removing the output layer of the Inception v3 model so that its output is the 2048 activations of the last pooling layer, called coding layer, instead of classes. These activations are the activation features produced from the inputs. The ground truth image dataset and generated image dataset are both given to the Inception v3 model resulting in two sets of feature vectors. After the activation feature sets have been produced the covariance and mean are calculated for both sets to summarize them and the results are used for calculating the Fréchet distance between both image datasets. The result is the output of the FID metric. [58]

The Fréchet distance is

$$dist_F^2(P,Q) = \left\|\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q\right\|_2^2 + Tr(\sum_P + \sum_Q - 2(\sum_P \sum_Q)^{\frac{1}{2}}) \quad (24)$$

where $P$ and $Q$ are multivariate normal distributions, $\sum_P$ and $\sum_Q$ are covariances of $P$ and $Q$ and $\boldsymbol{\mu}_P$ and $\boldsymbol{\mu}_Q$ are means of the $P$ and $Q$. This closed-form formula works only with multivariate normal distributions [59].

When calculating FID, it is assumed that feature vectors abide a Gaussian distribution [29]. However, this isn't the case with all Inception embeddings, and they aren't always normally distributed either. In those cases, the image quality assessed by FID may be inaccurate. [59] When the differences between the probability distributions of the real and generated images are small, the generated images are considered to resemble the ground truth images. [58]

FID has other disadvantages and weaknesses as well. All metrics based on the Inception v3 have some fundamental limitations such as the biases of the model and their dependency on the pretrained Inception model and its performance [29]. Also, the training of the Inception model has been done with a million images consisting of a thousand classes. These amounts are small in the long run leading to the difficulties to represent images that are complex and rich. [59] Standard metrics for measuring image quality like FID may make assumptions that aren't desirable regarding the distributional similarity in a dataset, and these metrics don't line up with human judgement all the time [29].

### 2.6.4 Natural Image Quality Evaluator

Natural Image Quality Evaluator (NIQE) is a no-reference IQA metric that computes values based on the natural scene statistic (NSS) model. It doesn't know anything about the expected

distortions in the data, and it only uses deviations measured from statistical regularities that can be detected from the natural high-resolution images. [11]

When computing NIQE, NSS model computes a set of local features for distorted images which are then given to a multivariate Gaussian model. The local features for natural high-resolution images are already computed and given to another multivariate Gaussian model. The distance between these multivariate Gaussian models is then computed. Mathematically, NIQE is

$$D(v_1, v_2, \sum_1, \sum_2) = \sqrt{((v_1 - v_2)^T \left(\frac{\sum_1 + \sum_2}{2}\right)^{-1} (v_1 - v_2))} \quad (25)$$

where $v_1$ and $v_2$ are mean vectors of multivariate Gaussian models of high-resolution and super-resolution images, $\sum_1$ and $\sum_2$ are covariance matrices of the same multivariate Gaussian models. The lower the perceptual quality of the image the higher the result of NIQE. [11]

### 2.6.5 Common Model Evaluation Metrics

Confusion Matrix, precision, recall and accuracy are common model evaluation metrics that are used for analysing a model for parameter training or comparing performances of two different models. From these metrics precision, recall and accuracy can be utilized for binary and multi-class classifier testing, and they are based on the confusion matrix. The relevant information regarding the classification rule performance and the model can be obtained from the confusion matrix which is why multiple metrics are based on it. [60] More of these metrics can be found from [60].

Confusion Matrix is a matrix where the performance of a classifier is examined by comparing predicted labels produced by the classifier to the actual labels. The actual labels are typically expressed by the rows of the matrix and the predicted labels by the columns, and they are in the same order. See figure 13. The crossing elements of the label headers are the amount of actual-predicted label pairs for each combination. The combination reveals if a class is predicted correctly or incorrectly and the way it is correct or incorrect such as in pair A-A the prediction is correct but in pair A-C the prediction is incorrect. [60]

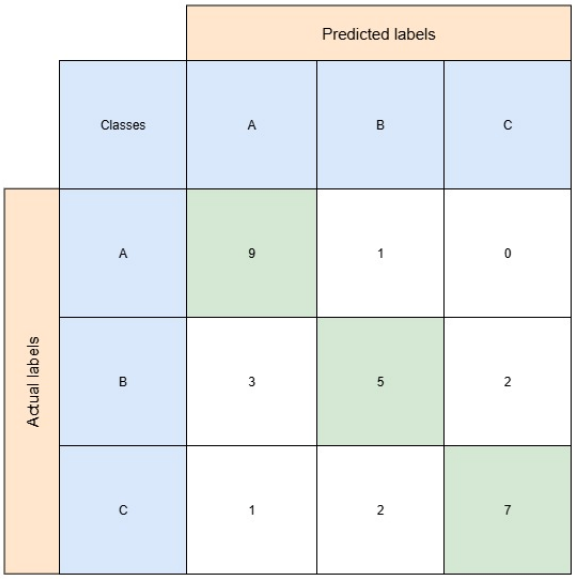


Figure 13. Confusion Matrix. [60]

The possible combinations can be divided into four classes: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). The class is TP when a label is predicted as positive and the actual label is positive. It is TN when the label is predicted as negative and the actual label is negative. FP is the case where the label is predicted wrongly positive while the actual label is negative and FN is when the label is predicted wrongly negative although the actual label is positive. TP values can be found from the main diagonal of the matrix from top-left to bottom-right. [60]

Recall expresses the accuracy of a model when predicting positive labels which means that it tells how good the model's ability is to predict positive labels for a dataset. The formula to calculate recall is

$$Recall = \frac{TP}{TP+FN} \tag{26}$$

that is the amount of the TP labels divided by the amount of all truly positive labels. [60]

Precision denotes the reliability of a model when predicting positive labels. This means that it tells the rate of how many predicted positive labels are truly positive. The formula of precision is

$$Precision = \frac{TP}{TP+FP}. \tag{26}$$

So, precision is TP labels divided by the amount of all labels that has been predicted as positives. [60]

Accuracy is an intuitively understandable metric that is very popular in tasks like multi-class classification. It expresses the probability for a model predicting successfully a label for a random sample. The formula for accuracy is

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad (27)$$

Therefore, the accuracy is the number of correct labels divided by the amount of all labels. The value given by the accuracy is between closed interval [0, 1]. Misclassification Rate is the probability of the incorrectly predicted labels and is calculated by subtracting the accuracy from the maximum value 1. [60]

F1-score evaluates the performance of a classification model by calculating the harmonic mean of precision and recall. The harmonic mean is used for finding an optimal trade-off between those two metrics. In the F1-score precision and recall are weighted equally, so they contribute the same amount to the score. If precision or recall is close to zero, it causes the F1-score value to drop because the harmonic mean weights smaller values higher than other values. [60]

Only TPs are used in the binary case and TNs are used along with them in the multiclass case. The values of the F1-score vary in the same closed interval [0, 1] as accuracy. The formula of F1-score is

$$F1 - score = \left(\frac{2}{precision^{-1} + recall^{-1}}\right) = 2\left(\frac{precision * recall}{precision+recall}\right). \quad (28)$$

This is the same as the weighted average of precision and recall. F1-score gives more credits to models that have the same precision and recall, and it gives more weight to smaller classes over larger ones. However, precision or recall can be weighted in the F1-score if needed. [60]

## 2.7 Data Augmentation

The most important thing for deep learning models to achieve good performance is to have a lot of training data that is rich and representative. However, it is challenging to achieve this kind of dataset, especially for different kinds of practical tasks, even today. The current datasets such as ImageNet [20] lack variability that could guarantee good performance. [61]

Another problem for deep learning models caused by small datasets is overfitting. If a model overfits, it doesn't learn the less important features of training data even though those features are necessary. This leads to a worse generalization ability. [62]

Data augmentation solves these problems. With it the accuracy and robustness of a deep learning model can be increased together with the model performance despite the usage of a poorly representative and small dataset. To achieve this, proper data augmentation techniques need to be utilized. Data augmentation, when used properly, simplifies model architecture and decrease the needed complexity of the model which both lead to better generalization capability. [61]

Data augmentation generates new training samples based on the training data of a model, or it generates them from scratch by adjusting appearance of the data for example. [61] This way the old samples offer more information that would otherwise be missed which increases the generalization of the model. Data augmentation also preserves the labels of an original image in the data augmented images, so the augmented images have no need to be labelled. [62]

Multiple regularization methods can be utilized for generalizing CNNs regarding the input data and the architecture of the models. For example, data augmentation, batch normalization and dropout are regularization methods that can be used in CNNs. With CNNs data augmentation has been proven to be a good trick for enhancing input data. [62]

Depending on research papers the taxonomy of classifying data augmentation techniques differs. From the papers [61] and [62] the taxonomy presented in paper [61] will be used. The papers have much in common, but differences can be found as well. For example, the flipping technique can be found from the geometric transformation class or from non-geometric transformation class.

Data augmentation is divided into data transformation and data synthesis in paper [61]. The super-resolution data augmentation falls somewhere in the data synthesis approaches, probably in the generative modelling category. Data transformation is divided into input space and feature space augmentations from where input space augmentation consists of techniques which change the input images directly with simple and intuitive transformations. These transformations are typically added directly into the learning process where they change the training data at each epoch lowering amount of needed memory. The point of these input space data augmentation techniques is to increase the variability of the data to increase the

generalization capability of a model. This class can be further divided into traditional and advanced data augmentation from where the traditional data augmentation can be divided into geometric and photometric data augmentation. [61]

The geometric transformations class includes vast variety of techniques that change the geometric properties of images by moving their pixels. For example, the aspect ratio, orientation and position of the images can be changed [62]. These changes are done with techniques, like translation, rotation and shearing, for mimicking the changes the real-world causes to images, such as the transitions of perspective, scale and viewpoint. [61]

In translation an image is moved to the direction of one cardinal direction and the part of the image, that would be cut off, is moved to the empty space the image leaves after the shifting. This must be made with a great care to ensure that no substantial changes are done for the image. This technique aims at diversifying the data representation. Flipping is a technique that flips an image vertically or horizontally. When using this technique, one must be careful to apply it only to suitable datasets as it can cause significant changes in datasets like MNIST [63]. [62]

The photometric transformations class techniques change the values in the pixels but not their spatial location. They typically change the combination of the RGB channel values with different functions. Therefore, it can be said that they change the visual appearance of the images. These techniques are usually based on digital image processing techniques, such as contrast, saturation and brightness. Photometric transformations are done by mimicking the photometric effects that real-world images would suffer. Those effects may be caused by colour artifacts, the camera or shooting conditions such as time of the day. By using these techniques, the accuracy and robustness of models can be enhanced. [61]

# 3 Initial Dataset

The dataset used in this research is a modification of a standard dataset for super-resolution called DF2K [7]. DF2K consists of 3850 RGB images divided into a training and test set. The training set has 3450 images and test set has 400 images. DF2K dataset is a combination of two datasets: DIV2K [13] and Flickr2K [14]. DIV2K dataset consists of 800 training images and 100 test images, and they are all high-resolution images. Therefore, Flickr2K consists of 2650 high-resolution training images and 300 low-resolution test images. DIV2K test set is called DIV2K_valid, and Flickr2K test set is called OutdoorSceneTest300. The resolution of the high-resolution images in DF2K typically varies between 1100x1100 pixels to 2040x2040 pixels, but some outliers exist such as 2040x816 pixel image. The low-resolution images in DF2K test set typically vary between 256x256 pixels to 720x720 pixels, but there exist some outliers in that set as well such as 768x512 pixel image. [7]

Because this thesis is about super-resolution, the data needs to be paired images. This means that high-resolution images must exist, and they must have low-resolution versions of them as their pair. DF2K dataset is used because it was one of the few datasets found from the Internet that was possible to be downloaded for free in a way that the high-resolution of images was preserved and the images could be labelled into multiple classes. The dataset was downloaded from Kaggle [7].

The training sets of DIV2K and Flickr and the test set DIV2K_valid have been combined to form an initial dataset. OutdoorSceneTest300 has been left out because it doesn’t contain high-resolution images and is, therefore, not sufficient for the purpose of this thesis. To achieve the goals of the thesis, the initial dataset was labelled into two classes: human and non-human. Other options existed, but humans are clear objects that always look quite similar and have roughly the same properties unlike animals, for example. Also, human images were available in significant quantities in DF2K. The initial dataset consists of 3327 images, and the classes human and non-human have 990 and 2337 images, respectively.

An image was labelled as human if an image contained at least one easily recognizable human or large head, arm or leg of a human. If it contained a very small human(s) with only their head and arms visible or their head, one arm and one leg visible, these images were also labelled as human such as figure 14. Example images of humans and non-humans can be found from Appendix 1.

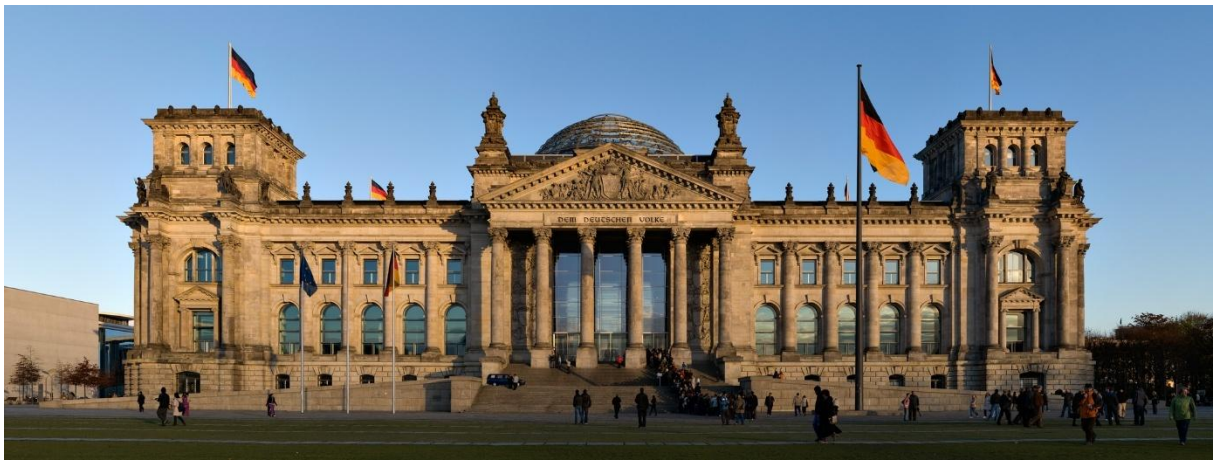

Figure 14. An example image labelled as human. [7]

A human can be easily recognized from an image where the human fills most of the image and should be, therefore, easy to produce with the modified SeeSR model. The most interesting images in the initial dataset are images where humans are small, but they are recognizable enough that the images are labelled as humans. This is done even if the images are mostly covered with a street view of a city or something similar. Otherwise, those images were labelled to the non-human class.

Because of this variation, the labelling of humans is a difficult task that depends on the human doing the labelling. Therefore, the labelling is not infallible. The principles and rules were set, and the labelling was made by a human, so errors and inconsistent decisions can't be avoided entirely. Therefore, there might be some images in the non-human class that should belong to the human class and vice versa.

The initial plan for the dataset was to label images into four classes. During this labelling process some images were decided to be too ambiguous to label cleanly into one class and these were collected and removed from the dataset. When the initial four label split was adjusted to binary labels to better support the objective of this thesis, the unclear images were left out. Duplicate images were found during the labelling process and removed.

Having only two classes that are hard to differentiate from each other supports the goal of the thesis more clearly than four classes. Also, the initial dataset is rich in content and not meant for classification which makes it a more difficult classification dataset and input dataset for the modified SeeSR. Because of these reasons, the CNN classifier doesn't need to focus on multiple classes at the same time but can focus on finding the differences between different versions of the same dataset. This way the modified SeeSR is challenged to produce clear

enough images where humans can still be recognized, while the classifier works as a metric for assessing the capability of the modified SeeSR at generation and, therefore, as a data augmentation method.

# 4 Experimental Setups

Libraries, pipelines, preprocessing, modified SeeSR, CNNs and metrics are discussed in this chapter, and two coding environments are used to create code. Python files are created in Notepad++ and ipynb files are created and run in Jupyter notebook. SeeSR has been created using Pytorch [64] and its code is spread across multiple python files. Tensorflow [65] and Keras [66] are used in ipynb files and for the code that runs Python files in Jupyter notebook. Google Colab is the environment used for running all the files to gain access to its GPUs. See table 2.

Table 1. Used file types, environments and frameworks divided by file type.

| File type | Creation environment | Framework / API | Running environment |
|---|---|---|---|
| python | Notepad++ | Pytorch | Google Colab |
| ipynb | Jupyter Notebook | Tensorflow, Keras | Google Colab |

Pytorch and Tensorflow are frameworks that are meant for coding machine learning and deep learning models meanwhile Keras is an API of the Tensorflow framework for building more high-level deep learning solutions. Torchvision is an image transformation package for Pytorch which is necessary for image processing. Diffusers package is Huggingface's package to implement and load different diffusion pipelines. Other packages with prominent role are Accelerate, Transformers, Numpy, PIL, torcheval and metrics of skimage. These and other packages with their specific versions can be found from tables 3 and 4.

SeeSR was implemented with specific packages, so to use it the same packages must be used. Not all packages with their correct version are marked in the requirements text file provided in GitHub of SeeSR [67]. Because of this, the missing packages must be loaded separately which leads to a situation where all package versions in the requirements file doesn't match with the loaded missing package versions. All packages needed for SeeSR with their working versions are listed in table 3. The packages are recommended to run in the listed order.

Table 2. Used packages and their versions in the modified SeeSR diffusion model.

| Package | Version |
|---|---|
| torch | 2.3.1+cu121 |
| torchvision | 0.18.1+cu121 |
| Python | 3.9.21 |
| huggingface_hub | 0.16.4. |

| Package | Version |
|---|---|
| numpy | 1.24.2 |
| safetensors | 0.5.3 |
| scipy | 1.13.1 |
| timm | 0.9.12 |
| scikit-image | 0.18.3 (0.25.2 may work) |
| torcheval | latest version (0.0.7) |
| tokenizer | 0.13.3 |
| torchmetrics | 1.5.2 |
| transformers | 4.33.3 |
| xformers | 0.0.27 |
| accelerate | 0.31.0 |
| diffusers | 0.23.0 |
| einops | 0.8.1 |
| fairscale | 0.4.13 |
| loralib | 0.1.2 |
| pandas | 2.0.1 |
| pydantic | 1.10.6 |
| pytorch-lightning | 2.3.3 |

Table 3. Packages used in other code than the modified SeeSR. Most of them use latest packages versions.

| Packages of other code | Version |
|---|---|
| sklearn | 1.7.2 |
| os | 3.14.0 |
| glob | 3.14.0 |
| numpy | 1.24.2 |
| keras | 3.11.1 |
| matplotlib | 3.10 |
| argparse | 3.14.0 |
| scipy | 1.13.1 |
| statistics | 3.13 |
| torch | 2.3.1+cu121 |
| torchvision | 0.18.1+cu121 |
| torcheval | 0.0.7 |
| skimage | 0.25.2 |
| shutil | 3.14.0 |
| mmetrics | 1.1.0 |

The experiment itself contains two pipelines. These pipelines are a classification pipeline and a metric pipeline. The datasets for classification are created in the classification pipeline along with the classification tasks of CNNs. The other pipeline creates datasets for assessing the diffusion model with different metrics so that the results are comparable with the results of the SeeSR research paper [5]. For pipelines see figure 15.

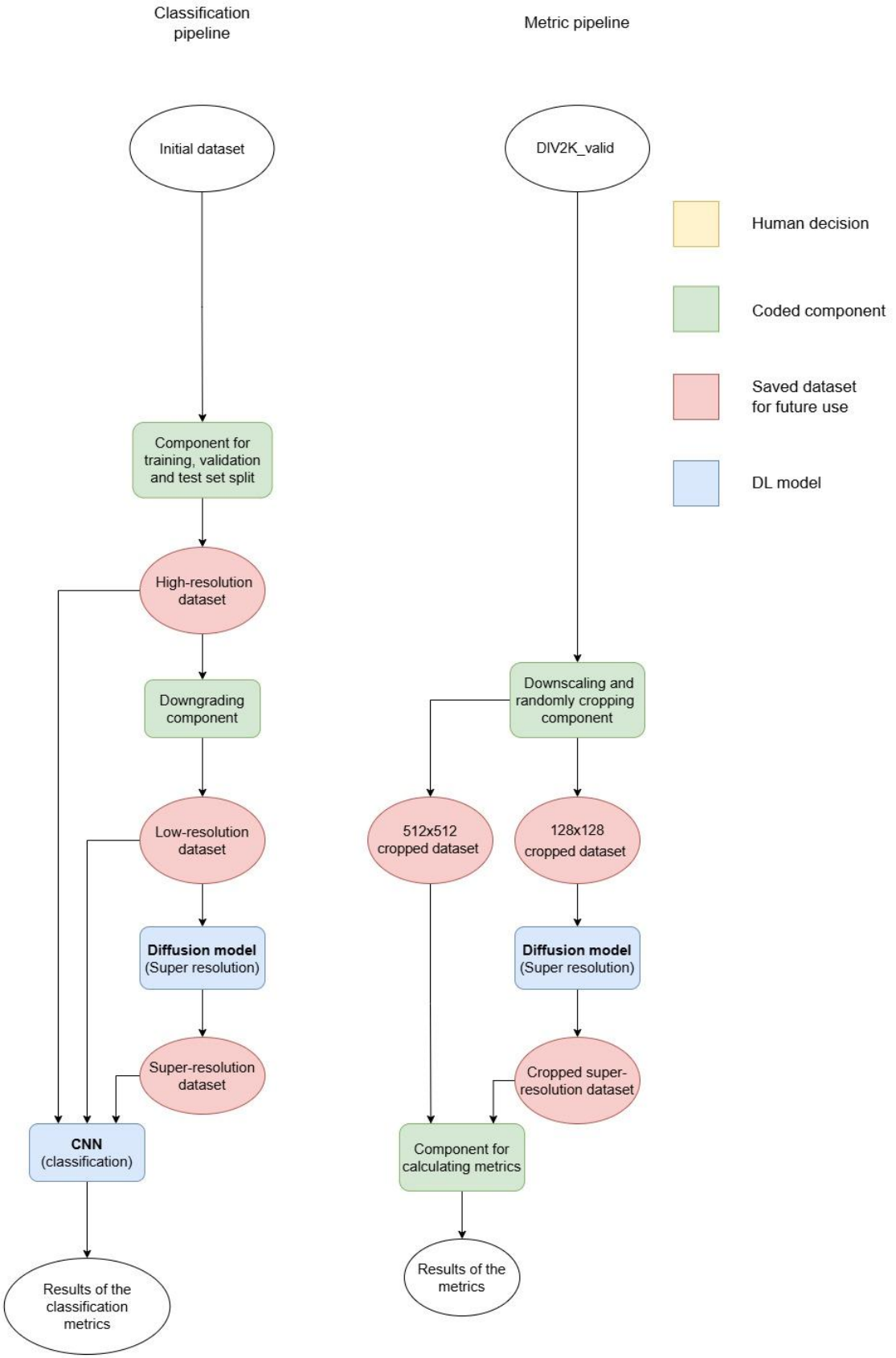


Figure 15. The classification and metric pipelines.

## 4.1 Preprocessing

In this chapter the three preprocessing methods of classification pipeline and one preprocessing method of the metric pipeline are explained in detail and each choice related to these methods are justified. Both pipelines begin with the initial dataset prepared in chapter 3.

Starting with the classification pipeline the high-resolution dataset is first created from the initial dataset by using split component that creates the training, validation and test set split. The only choice made in this component was to decide how to split the data into each set. Typically, the split is done the way that the training set has 60%-80% of the data, the validation set has 10%-20% of the data and the test set has 10%-20% of the data.

In this thesis the training, validation test set split is done by dividing the whole data into training, validation and test set at the same time in a ratio 80%, 10% and 10%, respectively. This choice was based on a limited understanding of the Pytorch library since the method for doing the split of all sets at the same time was known to me but other ways to do it with Pytorch were not. Therefore, the simultaneous split was used. This choice was made, since only 30% of the data in the initial dataset belongs to the human class. Therefore, the number of training images in human class is wanted to be maximized to achieve better accuracy of the CNN models, so the size of the training set was maximized.

The training set of the high-resolution dataset has total of 2664 images, the validation set has total of 332 images, and the test set has total of 331 images. The number of images in each class per set can be found from table 1.

Table 4. Number of images in each class per set.

| Class | Training set | Validation set | Test set |
|---|---|---|---|
| human | 792 | 99 | 99 |
| non-human | 1872 | 233 | 232 |
| Total | 2664 | 332 | 331 |

Then a low-resolution dataset is created based on the high-resolution dataset by downgrading it. The scale of the downgrading is the only decision that must be made in the downgrading component. The sizes of the high-resolution images are relevant to know for deciding a suitable scaling factor. The image heights and widths typically vary between 1100 to 2040 forming all kinds of combinations. All images that have at least 1 million pixels (1000x1000

= 1 million pixels) in them such as 2040x816 (2040x816 = 1 664 640 pixels) image are counted as a high-resolution image and all images that have under 1 million pixels in them are counted as low-resolution images. In the original SeeSR the low-resolution images given to the SeeSR model have size 128x128 [5].

With these in mind, I decided to aim about the same size as the 128x128 pixel images with my low-resolution dataset and I used, therefore, scaling factor number 6. Too low resolutions such as 64x64 could lead to unbearable data loss. In that case small humans could disappear from the low-resolution images which led to the wrong classification of human labelled images by a CNN model. At the same time the modified SeeSR couldn't generate the humans for the super-resolution image.

Because of the same reason, the images aren't scaled into the same size. The different image sizes in the data would lead to the alteration of image shapes that led to data loss. The modified SeeSR needs enough relevant data to be able to generate good looking images, so that the CNN model using that data can perform its classification task successfully.

The scale factor 6 works well with different sized high-resolution data since 1000 divided by 6 is about 166,7 and 2040 divided by 6 is 340. These values are still close to 128x128 pixels, but they probably retain relevant information such as small humans.

The idea behind the evaluation of the modified SeeSR is to compare the metric results of it and the original SeeSR. To ensure the same conditions between the models and, therefore, the validity of the comparison, the metric calculations have been made as similarly as possible as the SeeSR paper [5] describes. Therefore, the same initial dataset, DIV2K_valid, is used for both models. It consists of 100 images [7], and its image count is raised to 3000 by random cropping each image 30 times in the SeeSR research paper [5]. The size of the random cropped images is 128x128 pixels [5].

In this thesis the metric pipeline begins from the DIV2k_valid test set, and it has only one preprocessing component for the data. DIV2k_valid is given to the component that random crops the images resulting low-resolution images with size 512x512. Those images form a dataset of 3000 images that is hence called 512x512 cropped dataset and is used as a ground truth dataset in the evaluation of the modified SeeSR. Then that dataset is downscaled by downscaling factor 4 to form another dataset with 128x128 images that is hence called 128x128 cropped dataset.

## 4.2 Diffusion Model Setup

The starting point of the decision-making process was to decide if the diffusion model would be a pretrained model or crafted from the start. Diffusion models and their specialized versions for super-resolution have grown large after they have started emerging. They require significant prior knowledge about many deep learning concepts as can be seen from the wide range of topics introduced in chapter 2. So, this means that it is more efficient to use a pretrained model considering this thesis' time management aspect.

Also, available computational resources, such as memory and computational power, were limited. This emphasized the need to use a pretrained diffusion model. These computational limitations forced the use of Google Colab's premium Pro version despite using a pretrained model. Still, this model, along with CNNs and evaluation metrics of the modified SeeSR, was limited on computational aspects.

A GitHub web page [68] was used to survey different appropriate diffusion models to choose the best for the purposes of this thesis. The survey was done during January 2025 from the README's Papers section "1.2.1. Super Resolution (SR)" in part "1.2. Task-Specific Image Restoration" in section "1. Diffusion Models For Natural Image Restoration" in [68]. At the time the "OFTSR: One-Step Flow for Image Super-Resolution with Tunable Fidelity-Realism Trade-offs" paper [69] published in ArXiv 2024 and released in December 2024 was the newest super-resolution diffusion model.

To use a pretrained model it was necessary that researchers would have released the trained parameters of their model, so all models without publicly available parameters were instantly excluded. This included all papers without an available GitHub page and GitHub pages that did not have any publicly available parameters. Another criterion was that a GitHub page needed to have provided the model under a public licence, so that there was no requirement to ask for permission to use the model from the publishing researchers.

With these restrictions there were still a lot of different possible diffusion models available in [68], so transformer-based models were chosen to be excluded from the possible model pool. This was done because much more new information would have to have been gather and internalized than with the more familiar U-Net structure. After these criteria, only a couple of options were available from which to choose a pretrained model and SeeSR was chosen among them. SeeSR was roughly one year old at the time of selection, and it had been used as

a part of the more recent CCSR model. So, it was relatively new compared to some of the other models, but it was established enough to be used in other works as well.

However, the detail restoration capabilities of SeeSR seemed to be limited and, therefore, this thesis attempts to improve this detail restoration by changing an aspect of the original model. The rest of the non-transformer-based models that had a GitHub page were searched for possible detail enhancement options with the CCSR model becoming a noteworthy option. CCSR does the detail enhancement via its VAE decoder which is quick to implement to the similarly structured SeeSR model. Also, the VAE of the CCSR model seemed an interesting piece to experiment with because it decodes the latent features while also performing detail enhancement without additional computational cost. Because of the quickness and simplicity of the detail enhancement implementation and its need for no additional computational recourses, the VAE decoder of the CCSR model was decided to be used in the SeeSR model as a replacement of the original VAE of SeeSR. The original SeeSR code can be found from [67].

After replacing the VAE of SeeSR, the model has two purposes: generate a asuper-resolution dataset and a cropped super-resolution dataset for classification and metric calculations, respectively. The super-resolution dataset is created by giving the low-resolution dataset to the modified SeeSR which then outputs the super-resolution version. This happens in the classification pipeline.

T4 GPU with 15.0 GB RAM is used for the modified SeeSR to generate the super-resolution dataset. The RAM of the system is 51.0 GB. It took 230.5 hours, which is about 9 days and 14.5 hours, to produce the super-resolution dataset which consists of 3327 images.

The creation of cropped super-resolution dataset happens in the metric pipeline, and it is done by giving the 128x128 cropped dataset to the modified SeeSR which then outputs the cropped super-resolution dataset with image size 512x512. This is the other dataset utilized in the diffusion model metric calculations. It took 14,5 hours using L4 GPU with 22.5 GB RAM to generate cropped super-resolution dataset with the modified SeeSR. The RAM of the system is 53.0 GB.

To enable better comparison between the original and modified SeeSRs all parameter choices set as default in SeeSR are kept unchanged in the modified SeeSR. The only change is to save

the tags of each image, but it doesn't affect the performance. The default parameter choices are presented in table 5.

Table 5. Default parameters and their values of the SeeSR model.

| Hyperparameter/information | Value(s) |
|---|---|
| Tags given for each image | clean, high-resolution, 8k |
| User input tags | - |
| Tensor type | torch.cuda |
| Upscaling | x4 |
| Guidance scale | 5.5 |
| Conditioning scale | 1.0 |
| Number of inference steps | 50 |
| Blending alpha | 1.0 |
| Negative prompts | dotted, noise, blur, lowres, smooth |
| VAE encoder tiled size | 1024 |
| VAE decoder tiled size | 224 |
| latent tiled size | 96 |
| latent tiled overlap | 32 |
| start point | lr, noise |
| start steps | 999 |

## 4.3 Metric Component Setup

The last component in the metric pipeline is a component where different metrics are computed for evaluating the quality of the generated outputs of the modified SeeSR. These metrics are SSIM, PSNR, FID and NIQE and they are all used in the SeeSR paper as well.

The NIQE metric only needs the cropped super-resolution dataset as input, but the other metrics need image pairs of low-resolution and super-resolution images. This is why the 512x512 cropped dataset was created. The images in the image pairs given to the SSIM and PSNR metrics need to be the same size. Therefore, because the output images of the modified SeeSR are size 512x512, the image size of the 512x512 cropped dataset matches to the cropped super-resolution dataset.

When calculating the SSIM and PSNR metrics, the images are changed into YCbCr colour space. This is done because the SeeSR paper describes the same operation done when computing these metrics for SeeSR. SSIM and PSNR are computed by using

structural_similarity and peak_signal_noise_ratio methods, respectively. They are from the scikit-image API [70]. SSIM takes just one image pair at a time, so its scores are computed for each image pair and then the mean of the scores is taken. The PSNR functions the same way. Each method for metric calculations and their packages can found from table 6. The CUDA API of torch is used for quickening computations of all metrics.

Table 6. Methods and their packages used in metric computations.

| Package | Method |
|---|---|
| skimage.metrics | structural_similarity |
| skimage.metrics | peak_signal_noise_ratio |
| statistics | mean |
| torcheval.metrics | FrechetInceptionDistance |
| metrics.image_quality | niqe |

FID calculations are usually implemented through custom function calls, but there exist some APIs and libraries where it has been implemented as a method for more convenience. One of these libraries is TorchEval which FID method FrechetInceptionDistance is used in this thesis to calculate FID for saving time. See table 6 above. This method utilizes the Inception v3 model.

Because the computational cost is too high for updating all the images to the FID method, the size of the images is reduced from 512x512 to 299x299 which is the original input size of Inception v3. This memory saving action is needed to ensure a better FID score. For the same reason, the images updated to FID are given in four batches. It is the largest batch size that can be given to an L4 GPU before the memory runs out and that can still divide the test set data of 3000 images evenly. This means that the batch size is 750.

The FID score is then calculated in two different ways: one FID score is computed by giving all batches to FID and then computing the score once. The other FID score is computed by giving batches to FID one at a time and taking a mean of the four FID scores got in this way. This is done to enable better comparability because FID score depends heavily on the batch size and the exact batch size used to calculate FID for SeeSR hasn’t been specified in the SeeSR paper.

NIQE is calculated by using a library called mmetrics and using the package's method for it. See table 6 above. It takes just one image at a time, so it produces an index for all images from which their mean is then taken.

For all metric computations the L4 GPU is used because it has larger GPU RAM memory than T4. Its RAM is 22.5 GB while the RAM of the system is 53.0 GB and the disk's memory is 235.7 GB. The larger GPU memory is needed especially for the FID computation to enable larger batch sizes.

The reason to choose exactly these metrics lies in the popularity of the metrics. FID is a standard metric in IQA. NIQE is a popular no-reference metric that is used to bring variety because the other three metrics are reference metrics. PSNR might be the most popular IQA metric, and it is at least one of the most used metrics in SISR [10]. SSIM is another traditional and very popular IQA metric [1]. SSIM and PSNR are quite similar metrics, but FID and NIQE are completely different from them and from each other, so using FID and NIQE gives variation to the metrics and helps to better understand the capability of the modified SeeSR. Together these metrics occur in wide range of studies more constantly than other metrics. So, using them in this thesis is a natural continuation in the line-up not to mention their usage in the SeeSR paper.

## 4.4 CNN Model Setups

The classification pipeline continues with the CNN models that perform the classification tasks. In this section of the code the data is loaded, the initial CNN is built and then it is customized for the low-resolution, high-resolution and super-resolution datasets one by one. Therefore, each dataset has its own CNN model. The customization is done because the datasets have different properties and by customizing the CNNs a little, the models perform better. Then these CNNs are trained and evaluated with different metrics. All models are run with L4 GPU that has 22.5G RAM along with 53G system RAM.

In this thesis only the choices made regarding the code are discussed. The choices are related to loading the datasets, building the CNN models and model training and evaluation. When it comes to the choices regarding data, CNN can't take data with multiple input sizes. Because the image sizes differ within the datasets, either the size of the images must be standardised, or the CNN model must be changed to a fully convolutional network that can take different sized images as input. The latter option includes several changes to the basic structure of a

CNN model, such as using Global pooling instead of a flatten layer and using “None” as the input size. Because this option would have needed a lot of additional research, padding the input to a fixed size was chosen instead. Other reasons to use padding were the possible problems with arrays and the threat of being forced to use batch size of one in a fully convolutional network.

The size of the padding depends on a dataset used. In the low-resolution dataset the maximum width and height are both 340 pixels and in the super-resolution dataset the maximum width and height are both 1360 pixels, so these widths and heights are used as padding size in the respective datasets. See table 7. The maximum sizes are used to keep the sizes of the maximum sized images the same. By doing this, the loss of the information in images are minimized because there is no need to reduce the image sizes. Instead, the smaller images are padded into the same size in a way that preserves their aspect ratio. pad_to_aspect_ratio method is utilized for padding the images.

Table 7. Widths and lenghts of the three datasets.

| Dataset | Maximum width and height (in pixels) | Used width and height |
|---|---|---|
| Low-resolution | 340 | 340 |
| High-resolution | 2040 | 1360 |
| Super-resolution | 1360 | 1360 |

However, the high-resolution dataset is an exception. It uses the same sized padding as the super-resolution dataset to achieve a fair baseline for evaluation. If the high-resolution images are much larger than the super-resolution images, the CNN model with the high-resolution dataset gets more information about the dataset. However, if the images have the same size, the differences in performance can’t be explained by the amounts of information but rather the different information in the images.

Another reason is the computational restrictions that force the image size to be smaller than 2040x2040 pixels. With 2040x2040 sized images the RAM of L4 GPU can take only a batch of five images at a time. When using mixed precision [71], the maximum batch size is eight. The proposed CNN model doesn’t learn with that small batch size and through testing it was discovered that it needs at least a batch size of 18. This can be achieved by reducing the image size to 1360x1360 which is the same size as the maximum sizes of images in the super-

resolution dataset. Mixed precision changes some parts of a CNN model to use float16 instead of float32 saving some memory that enables larger batch sizes [71].

Because of this struggle between image and memory sizes, the batch size of high-resolution dataset is set to 18. Using the 1360x1360 image size and batch size of 18 20.7G out of 22.5G of the L4 GPURAM is used when training with both datasets. The batch size of low-resolution dataset is 64 and it could be larger because of the small maximum size of the images in the dataset.

When loading the training and validation sets, the data is shuffled. However, the test sets aren't shuffled because that causes the labels and images to be shuffled in such a way that it becomes impossible to tell which label belongs to which image. The labels, human and non-human, are expressed as numbers zero and one, respectively.

After loading the input images some data augmentation method layers are used because of the small sizes of the datasets. Five data augmentation methods are applied to the data: two geometric transformations and three photometric transformations. The geometric transformations are horizontal flip and translation with values 0.1 and 0.1, respectively. The photometric transformations are saturation in range [0.4, 0.6], contrast with value 0.2 and brightness with value 0.1.

The datasets include images that are easy to classify correctly, but they also include images that are very hard to classify correctly. That is because the humans in those images are very small and sometimes not easily spotted. To avoid situations that even humans can't classify correctly the images because of blurriness or pixel shifts in the images, the mentioned five data augmentation methods are chosen and used with small values. For example, if randomCrop method would be used, it may had cropped out humans from the target image leading to a situation where the image is impossible to classify correctly. The data augmentation methods are chosen from the image augmentation web page of Keras [72].

After the augmentation layers comes a rescaling layer where the input images are rescaled to be in range [0, 1]. Then comes eight CNN layer blocks that consist of a convolutional layer, an activation function and a pooling layer, respectively. The convolutional layer has "same" padding and kernel size 3x3, while the number of filters changes between 64, 128 and 256. ReLU is utilized as the activation function and max pooling as the pooling layer in all blocks. The pool size of the max pooling layers is 2x2, stride is 2 and padding is "same".

To enable CNN model learning, the pre-flatten layer size of the convolution kernel needed to be reduced to 2x2xN. This means that with the low-resolution dataset the initial CNN can be used as it is, but other datasets need two additional convolutional blocks.

After all necessary blocks comes the rest of the structure of the initial model including a flatten layer, two dense layers, three activation functions and an output layer. The number of neurons in the dense layers are 512 and 200, respectively, and the two activation functions that follow are ReLUs. The dense output layer has two output neurons followed by a SoftMax activation function configured to use data type float32 for the purpose of mixed precision. Usually when using binary data one output neuron and sigmoid activation function are used but, in this case, CNNs won't learn if those choices are applied. Therefore, the multiclass choice is utilized.

RSMProp is used as an optimizer in these CNNs since the models were incapable of learning with Adam and there isn't enough time to optimize SGD. Learning rate of the optimizer for the CNN model using the low-resolution dataset is the default 0.001, and for other models it's 0.0001. Its values are chosen such that they enhance the learning of the models. Loss parameter is sparce_categorical_crossentropy and accuracy parameter is sparce_categorical_accuracy. The multiclass choices are used based on the output layer of the CNN base model.

The CNNs are trained for 120 epochs to see the points when they start to overfit to the datasets. The best models for each dataset based on validation accuracy are saved and used for evaluation. For high-resolution and super-resolution datasets mixed precision is used to enable larger batch sizes. The evaluation metrics are confusion matrix, FID-score, precision and recall as they are comprehensive metrics meant for evaluation of classification models.

# 5 Experiment Results and Discussion

In this chapter the metrics computed for the modified SeeSR are compared to the results reported for the original SeeSR model in the SeeSR paper [5]. Then example images in low-resolution, high-resolution and super-resolution datasets are presented and compared to show the differences between each dataset. Last, the metrics calculated for CNNs are compared and discussed.

## 5.1 Diffusion Model's Results

The SSIM, PSNR, FID and NIQE metrics have been computed for the modified SeeSR. These results can be found from table 8 along with the corresponding results reported for SeeSR and two other diffusion models in the SeeSR paper [5]. For FID two versions were computed: FID that was calculated from the whole dataset (full-set FID) and FID that was calculated from one batch at a time (mean FID) with batch size of 750. The result of full-set FID is 15.85 and the result of mean FID is 48.19.

Table 8. The results of metrics for the modified SeeSR, SeeSR and two other models.

| Metric | SeeSR with CCSR VAE | SeeSR | StableSR | ResShift |
|---|---|---|---|---|
| SSIM | 0.5676 | 0.5386 | 0.4887 | 0.5422 |
| PSNR | 13.35 | 21.19 | 20.84 | 21.75 |
| Full-set FID | 15.85 | 31.93 | 36.57 | 55.77 |
| Mean FID | 48.19 | 31.93 | 36.57 | 55.77 |
| NIQE | 0.2485 | 4.9275 | 4.6551 | 6.9731 |

As reported in table 8 the SSIM value of the modified SeeSR is slightly worse than the SSIM value of SeeSR (0. 5386). However, the values aren't too dissimilar and, therefore, the result suggests that the modified SeeSR is roughly equivalent to SeeSR. When comparing SSIM of the modified SeeSR to SSIMs of StableSR and ResShift with values 0.4887 and 0.5422, respectively, the modified SeeSR is almost as good as ResShift but falls far behind StableSR.

PSNR of the modified SeeSR is also plausible but not very good. It is 13.35, while SeeSR has 21.19, StableSR has 20.84 and ResShift has 21.75 as their PSNR value. PSNR of the modified SeeSR falls badly behind the results of other models. None of these results are strong PSNR values and given that the modified SeeSR is a derivative of SeeSR, it was unlikely that it would perform well here either. However, the modified SeeSR falls far behind SeeSR as well.

This might be a consequence of the way PSNR is calculated and the dataset that is given to it. PSNR uses pixel-wise MSE as part of its calculation. This combined with randomly cropped data that is likely different from the cropped dataset used in the SeeSR paper which leads to different PSNR values. This may not explain the whole gap, but it effects the results somewhat.

PSNR of the modified SeeSR is also plausible but not very good. It is 13.35 while, the PSNR value of SeeSR, StableSR and ResShift are 21.19, 20.84 and 21.75, respectively. PSNR of the modified SeeSR falls badly behind the other values. This can be caused by multiple reasons such as the test set used for the modified SeeSR not being completely same as the one used for other models since while creating the test set, random crop crops the set randomly.

Another challenge may be caused by the MSE computed for calculating PSNR values. Because the modified SeeSR generates sharper and clearer images than SeeSR, the pixel values inside the output images generated by the modified SeeSR are larger. Therefore, they differ more from the low-resolution input images than the output images of SeeSR resulting with a worse PSNR value. The modified SeeSR may over-emphasize details which may cause larger pixel-wise distance in MSE and, therefore, worse PSNR values. So, PSNR isn't necessarily a good estimate for the quality of the output images of the modified SeeSR.

Figure 16 shows an input image for SeeSR and the modified SeeSR and their output images, respectively. The size of the input image is 128x128 and, therefore, it seems blurry. The output images are super-resolution versions of the input image and that's why they are much sharper and clearer. However, clear differences can be seen between these two outputs. The image from the modified SeeSR is sharper and more detailed than the image from SeeSR which can be seen from the horn-like structure of the creature and from its raised limb. The differences show that the VAE component distinctly improves the detail creation of SeeSR leading to sharper and more detailed results. Other example images can be found from Appendix 2.

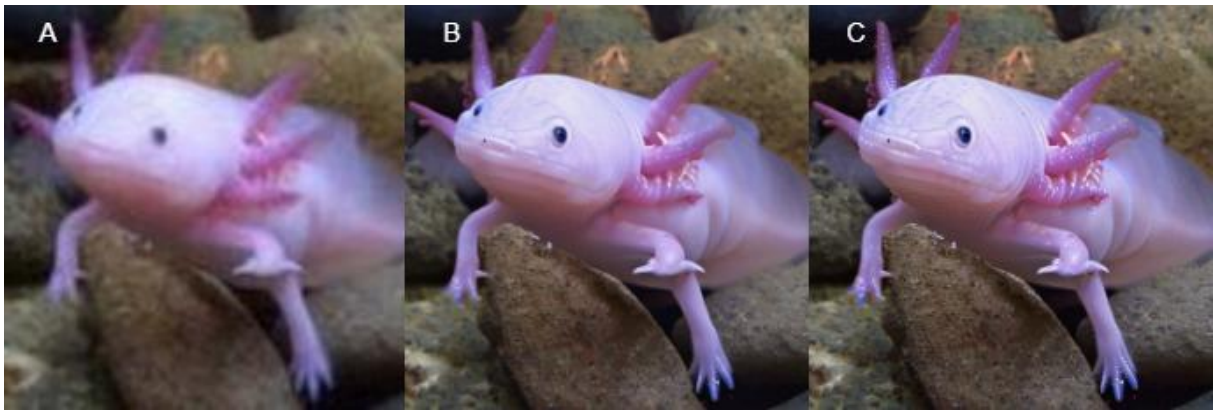


Figure 16. Image A is input image for SeeSR and the modified SeeSR, images B and C are output images of SeeSR and the modified SeeSR, respectively. Images A and B are from [67].

When comparing full-set FID of 15.85 to the FID of original SeeSR of 31.93, the full-set FID is much better than FID of SeeSR. The same occurs with the FID scores of StableSR and ResShift with values are 36.57 and 55.77, respectively. However, when comparing the mean FID of 48.19 to the FID scores of SeeSR and the other models, it is clearly much worse than the scores of SeeSR and StableSR but better than the score of ResShift.

When comparing the full-set FID and the mean FID of the modified SeeSR while considering the reported results, the mean FID seems like the more reliable and realistic score than full-set FID. This is because the batch sizes used to calculate the FID scores of the comparison models have not been mentioned in the SeeSR paper and, therefore, their batch sizes are unknown. If the batch sizes would be known, full-set FID could be a reliable score. So, mean FID is focused on this thesis from now on.

The mean FID result seems plausible and demonstrates reasonable performance. However, as implied it is not completely flawless. The batch size used to compute it may not match to the batch size used for the original FID which could cause differences between the FID scores of the modified SeeSR and SeeSR. Other reasons for these differences are the change of a component in the modified SeeSR and that the VAE component of the modified SeeSR isn't fine-tuned at all in this model. The change done for the SeeSR model isn't massive, but it makes evident changes to images even humans can see without inconvenience.

The final metric is NIQE. The modified SeeSR performed too well to be reliable as its NIQE index is 0.2485. It is expected that the index would be about as good as the index of SeeSR, a score of 4.9275, but not that much better. The indexes of other models were 4.6551 and 6.9731. Something might be wrong with the computations done by the NIQE method. There isn't much information shared on the page from where the method was imported, so the

method computations are hard to check. Therefore, the metric should be computed again with a more reliable method, or it should be implemented from scratch to ensure a correct result.

There is a possibility that NIQE emphasizes sharp and clear details that the modified SeeSR can provide. This would cause the better NIQE index of the modified SeeSR. However, the index is so large compared to the index of SeeSR that it is not plausible to achieve it with only one component changed and without fine-tuning. Not to mention that no radical changes are observable from the example images in figure 16 nor in Appendix 2.

When it comes to all previous metrics discussed, the results of the metrics achieved with the modified SeeSR can also be worse than the results of SeeSR because the modified SeeSR may not be as good model as the original model is. By comparing the metrics, it seems like that might be the case. NIQE of the modified SeeSR is also expected to perform worse than SeeSR and that's why its excellent performance is so suspicious. However, above output images tell a different story and, therefore, to be surer about the performance of the modified SeeSR more metrics should be computed for it and compared to SeeSR such as LPIPS. More of this and other future work possibilities can be found from chapter 6.

## 5.2 Comparison of High-Resolution and Super-Resolution Datasets

Now, some example output images of the modified SeeSR are presented and compared to their high-resolution variants. The corresponding low-resolution versions of these images are also presented. By comparing the images, a better understanding can be achieved about the super-resolution dataset and especially the mistakes the modified SeeSR makes. Next, four example super-resolution images of humans and their low and high-resolution counterparts are presented and discussed. The humans are presented in different sizes, contexts, and difficulty levels in the images. Non-human images can be found in Appendix 3 for comparison since their successful detailed recreation is less important in this thesis. It matters, of course, but the exact texture material of waves in the ocean or a bit quirky beak of a penguin will not affect the classification in the same manner as a weirdly shaped human head.

All images presented in this chapter and in all Appendixes are originally from [7]. Some of the images have been distorted either via SeeSR, the modified SeeSR or downgrading. This has been noted in the figure captions in some of the following ways: super-resolution image (distorted by the modified SeeSR), low-resolution image (distorted by downgrading) or mentioning the diffusion model used to generate the image.

Starting from an image where there are five humans covering most of the image. This first image pair consists of figures 17 and 18. Under them figure 19 is presented. It is a low-resolution version of the images in figures 17 and 18. By inspecting the images, a lot of differences can be detected such as the textual mistake the diffusion model has made. All mistakes discussed in this text are surrounded by a red square in the images to highlight the areas of interests. The textual mistake can be found from the right edge of the image.


Figure 17. High-resolution image of a group of people posing to the camera.


Figure 18. Super-resolution image of a group of people posing to the camera.

Figure 19. Low-resolution image of a group of people posing to the camera.

Despite multiple mistakes in the super-resolution image the humans in the images are the most interesting areas. Going on from left to right the first red square marks a human face completely distorted in the super-resolution variant. The second square is from one of the

foreground men that has features of his face completely changed along with his totally changed facial expression. The third square highlights the cap of one of the foreground men and his hair, but in the super-resolution variant the hair is missing and the cap formed with a weird shape covers his whole head. The fourth square has a human with glasses in the super-resolution variant although he should not have them.

The fact that the diffusion model could not generate even the foreground humans correctly but made huge mistakes like generating glasses and beard out of nowhere is a bit alarming. However, this doesn't tell everything because it is just one image and generation can differ depending on the difficulty level of images. The next high-resolution image example is more challenging in the aspect of generating humans. This image and its counterparts can be found from figures 20-22.


Figure 20. High-resolution image where two humans sit outside in the background of the image.


Figure 21. Super-resolution image where two humans sit outside in the background of the image.


Figure 22. Low-resolution image where two humans sit outside in the background of the image.

There are just two humans in the background of the images. The super-resolution image depicts the humans with a red skin and in different clothes. However, they are still clearly humans – or at least the man in the left is. The other one is recognizable to human because of

the context provided in a form of counterpart images and the other human sitting opposite him.

Moving to a more challenging image example that is about a human peeking behind a giant tree. The small human blends very well to the dark background and, therefore, he isn't distinguishable very well. This can be seen from a mistake the diffusion model has made. It has interpreted that the human is part of the tree branches that can be seen behind him. Consequently, any human can't be found from the super-resolution image and hence the image can't be classified correctly. See figures 23-25 for these images.


Figure 23. High-resolution image of a human peeking behind a giant tree.

Figure 24. Super-resolution image of a human peeking behind a giant tree.

Figure 25. Low-resolution image of a human peeking behind a giant tree.

The last example image that can be found from figures 26-28 is about human standing on a street while he is half behind a sapling. The human is very small and no details other than the body parts can be recognized from him. In the super-resolution image, his head and legs have

disappeared, and the rest of his body has been merged into the background. Therefore, the human isn't likely to be classified correctly.

Figure 26. High-resolution image of a street view with a small human on the left side of the image.

Figure 27. Super-resolution image of a street view with a small human on the left side of the image.

Figure 28. Low-resolution image of a street view with a small human on the left side of the image.

When inspecting all the images presented in this section, it can be noticed that the more a human blends to the background and the smaller they are the more poorly the modified SeeSR has been able to generate them. The model makes notable mistakes even with humans photographed nearby. This foreshadows difficulties for CNN trying to learn based on the super-resolution dataset. However, here were only presented couple of images as examples and too straightforward conclusions can't be derived with this.

## 5.3 Classification Results

CNN model that has low-resolution dataset as its input is called low-resolution CNN from now on. Similarly, the high-resolution version is high-resolution CNN and super-resolution version is super-resolution CNN. With these terms set accuracy, precision, recall and F1-score of each classification model can be presented and compared along with their confusion matrices effectively.

As can be seen from table 9 high-resolution CNN performs worst, and super-resolution CNN performs best out of all CNNs with all metrics. When comparing low-resolution and high-resolution CNNs, the difference isn't too large. The same is true for low-resolution and super-resolution CNNs but for the high-resolution and super-resolution pair these differences start to become more noticeable.

Table 9. The metric results of the CNN models with low-resolution, high-resolution and super-resolution datasets.

| Model with Dataset | Test Set Accuracy (%) | Precision (%) | Recall (%) | F1-score |
|---|---|---|---|---|
| Low-resolution | 80.36 | 76.59 | 76.44 | 0.7651 |
| High-resolution | 77.95 | 73.89 | 71.24 | 0.7225 |
| Super-resolution | 82.48 | 79.74 | 77.08 | 0.7817 |

Low-resolution dataset has blurred images due to its low pixel amount that erases or blurs the details from the images. This may lead to a situation where some of the images are difficult or impossible to classify correctly due to the challenging dataset. The diffusion model takes the low-resolution dataset as input to produce the super-resolution dataset. This may cause the blurred details to be reconstructed poorly when compared to the high-resolution variant of the images. Also, the erased details can't be recovered which leads to the missing information such as missing humans from the images.

However, the missing or blurred details may be a benefit for low-resolution CNN, since it can focus on clearer and more observable humans than high-resolution CNN. This may indirectly benefit super-resolution CNN, because low-resolution dataset has been used to generate super-resolution dataset and, therefore, super-resolution dataset has clearer and more observable humans than low-resolution dataset.

With the super-resolution dataset, the performance of the modified SeeSR affects the results significantly as well because it must be able to recognize and produce good-looking humans for super-resolution CNN to be able to classify the images correctly. It can be inferred from the results that the modified SeeSR has been able to produce reasonably good images for super-resolution CNN to outperform others.

At the same time, the modified SeeSR makes the images and their details so sharp and clear that the images resemble the high-resolution variants more closely in these aspects. This may give the super-resolution CNN better tools to recognize human images than the low-resolution model. It is also possible that the modified SeeSR can't produce all humans to look like humans to the human eye, but it makes some mistakes that share similar features with well-produced humans that are recognizable for super-resolution CNN. This could lead to the better classification performance.

To say more about why high-resolution CNN performs so poorly, more results need to be inspected. Before that table 9 provides one more notable thing: all accuracies stay close to 80% accuracy. It isn't a great outcome but implies that the models learn the classification task. All metric values of high-resolution CNN are poor, but super-resolution CNN performs decently.

This is probably caused partially by the difficult datasets given to these models. They are very challenging because of the small humans in the images of the datasets. The idea was to be able to emphasize the differences between the CNN models which are caused by the different resolutions of the datasets. However, it looks like the datasets are so difficult that the models have some hard time learning to classify the data, which is even more likely with the difficult images.

The limitations of the GPU memory also cause the differences between the datasets. Most of the high-resolution images have been downscaled from maximum of 2040x2040 pixels to 1360x1360 pixels to enable a large enough batch size for high-resolution CNN. This downscaling blurs the images which causes sharpness and clearness of images in the high-resolution dataset to suffer and, therefore, complicates the classification task. Now, the images that may have been classified as humans or learned easier to be classified as humans may be classified incorrectly. The batch size 18 of high-resolution and super-resolution CNNs complicates the learning of these models by losing some of the stabilizing effect of large batch sizes. This may cause lower performance of super-resolution CNN and that high-resolution CNN don't learn the task properly. Also, the usage of mixed precision causes small data loss inside high-resolution and super-resolution CNNs to enable larger GPU memory. This might affect their learning negatively.

When inspecting the class wise precision and recall for all datasets, a couple of trends can be seen among the metrics. These results can be found from table 10 from which can be seen that the models don't succeed very well with the human class and succeed much better with the non-human class. This was expected since the datasets are challenging and the classes are imbalanced, but the variation range is larger than expected. The largest gap between the human class precision and non-human class precision is about 18.5 percentage points and it resides in low-resolution CNN. The smallest gap is about 11.5 percentage points, and it can be found from super-resolution CNN. When comparing the recall values, the largest gap between

human and non-human classes is about 33.5 percentage points found from high-resolution CNN and the smallest gap is about 19.5 percentage points residing in low-resolution CNN.

Table 10. Class-wise metric results of the CNN models with different datasets.

| Model with Dataset | Human Class Precision (%) | Non-Human Class Precision (%) | Human Class Recall (%) | Non-Human Class Recall (%) |
|---|---|---|---|---|
| LR | 67.34 | 85.84 | 66.66 | 86.20 |
| HR | 65.85 | 81.92 | 54.53 | 87.93 |
| SR | 74.10 | 85.63 | 63.62 | 90.51 |

Comparing the CNN models to each other high-resolution CNN has the smallest precision values for both classes. The other models have about the same precision value for non-human class, but super-resolution CNN outperforms other models in the human class significantly in precision.

When comparing the recall values of CNNs, high-resolution CNN performs terribly with the human class. The others are a bit better but not by much. However, the models achieve good values with non-human class. Surprisingly, low-resolution CNN performs the worst while super-resolution CNN outperforms the others again with an excellent score of 90.51%.

The results of precisions and recalls imply that CNNs predict the non-human labels very well but make a lot of mistakes when trying to predict the human class. This is likely to be partially caused by the class imbalance in the training data which biases the models to prioritize the non-human class.

This class includes about 70% of all data leaving to the human class 30% of the data. This imbalance between the classes complicates the performed task because accuracy weights the larger class higher seeing it to be more relevant than the smaller class. This causes some of the classification mistakes the three models make, and it should have been taken better into account while labelling the datasets or training CNNs. For example, more human data could have been searched and added to the dataset to balance the classes or misclassification rates of loss functions of CNNs could have been weighted differently.

The differences in the confusion matrices are similar to the differences presented for precision, recall and F1-score results. High-resolution CNN falls behind the other two CNNs while super-resolution CNN outperforms the others. The results of the confusion matrices can

be found from figures 29-31. The super-resolution dataset is referred to as the diffusion dataset in the figure 31.

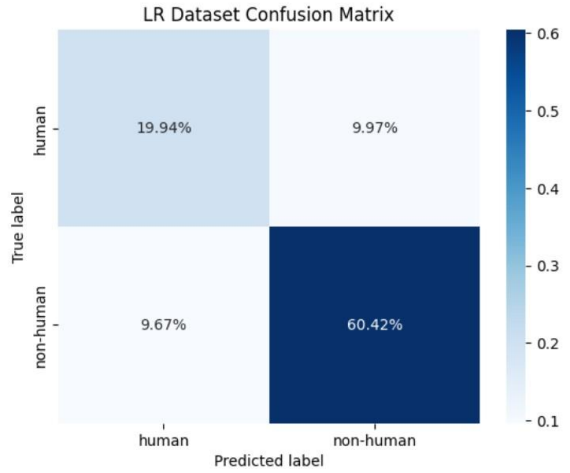


Figure 29. Confusion matrix of low-resolution CNN.

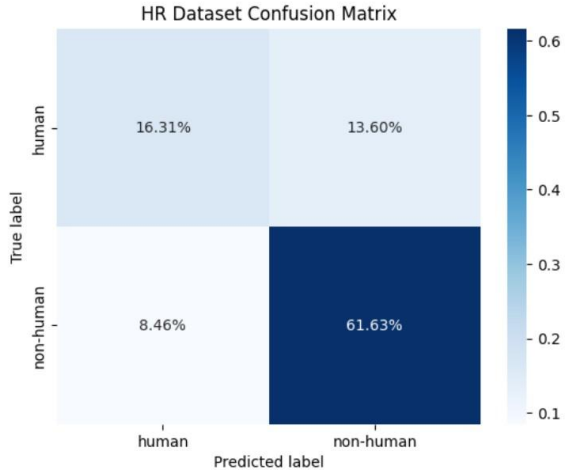


Figure 30. Confusion matrix of high-resolution CNN.

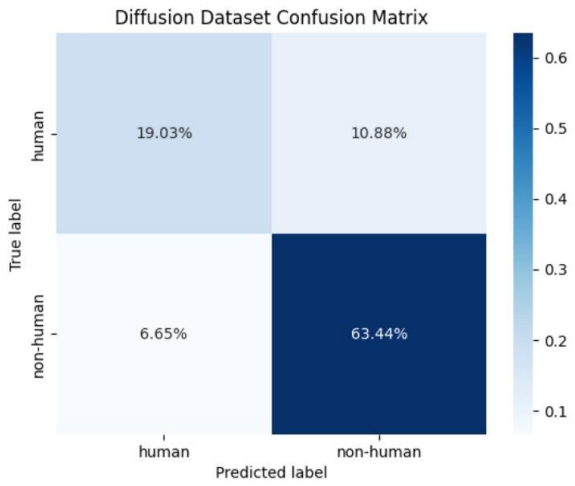


Figure 31. Confusion matrix of super-resolution CNN.

From these results can be seen that super-resolution CNN classifies the non-human class best although just barely. At the same time, it makes least mistakes when classifying non-humans. Low-resolution CNN performs best when classifying humans but makes about same amount of classification mistakes for both classes. High-resolution CNN falls behind the others in correctly classified humans. However, it makes much less classification mistakes when classifying non-humans managing to outperform low-resolution CNN with making less mistakes with non-humans.

The confusion matrices show that low-resolution CNN learns to classify the easy human and non-human images, but it guesses the hard ones resulting 50%-50% division between the classes. When it comes to high-resolution CNN, it learns the classes poorly and prioritizes guessing the inputs into the non-human class likely because of class imbalances and other previously discussed reasons. However, super-resolution CNN starts to learn both classes with an especially high success rate on the non-human class.

The blurriness of the images in the low-resolution dataset may cause the evenly divided mistakes since the details are erased or blurred badly from the difficult images. These images have humans that are very small and, therefore, they probably disappear during the downscaling process. But as stated, this leads low-resolution CNN to classify easy images correctly and to guess the difficult ones. That also benefits super-resolution CNN to perform better with the enhanced details of the known humans and probably with some of them being more observable than in low-resolution dataset.

However, the high-resolution dataset has much more details available in the images and, therefore, high-resolution CNN has the best competence to recognize the humans in the input

data. This means that it is facing the biggest challenge. The brutal data with small humans that are hard to separate from the background may be too hard for the high-resolution CNN in a way that the model tries to optimize to learn the non-human class to the detriment of the human class. This optimization is the effect of the imbalanced classes in the data.

There is a chance that all the variations in the classification results are just caused by variance. Because super-resolution and high-resolution accuracies differ by 4 percentage points, it implies that super-resolution CNN is likely to be better than high-resolution CNN. See table 9. To find out the significance of the results and to evaluate the classifiers, McNemar's test [73] is applied to CNNs. It is a statistical test meant for large models that can't be evaluated on multiple datasets to compare performance [73].

When comparing the high-resolution and super-resolution CNNs, the *p*-value from the McNemar's test is 0.0275 which is less than the standard confidence value of 0.05. This means that the performance differences between super-resolution CNN and high-resolution CNN can be said to be statistically significant. The *p*-value of low-resolution and high-resolution CNN pair is 0.3222 and the *p*-value of low-resolution and super-resolution CNN pair is 0.4011. Therefore, these pairs have a significant chance that their results are caused by variance. If the data had been stratified and more GPU memory had been available, the results could have plausibly been enhanced.

# 6 Conclusion

The idea behind the thesis was to find out if a super-resolution diffusion model could be used as a data augmentation method. So, the thesis aimed to generate super-resolution image data with a SeeSR diffusion model and test it against the low-resolution and high-resolution variants of the data. This comparison was done by utilizing CNN classifiers. Next, the research questions are answered one by one. Discussion related to the research questions can be found from chapter 5.

**RQ1**: How does the changing of the variational autoencoder of the SeeSR model affect to the performance of SeeSR?

Changing the VAE component of SeeSR affects performance negatively. The visual differences in output images aren't large, but the differences in metric results are significant. The modified SeeSR performs worse with SSIM, PSNR and FID than the original SeeSR but overperforms unrealistically with NIQE. More metrics should be tested to ensure the performance effects with other kinds of metrics such as subjective IQA methods.

**RQ2**: How well does super-resolution diffusion model perform as a data augmentation method from the perspective of a CNN classifier?

The super-resolution data generated with the modified SeeSR produces best classification results out of all datasets with all classification metrics. The difference isn't large but by enhancing the CNN models, utilizing some misclassification weighting in loss function and using a less complex dataset, the real and bigger differences between datasets may be discovered.

**RQ3**: How well does super-resolution data generated with a diffusion model approximates the original high-resolution data counterpart from the perspective of a CNN classifier?

Super-resolution CNN performs better than high-resolution CNN. The difference is notable and may be caused by batch size and image size issues. The performances of CNNs imply that the super-resolution dataset approximates the high-resolution dataset quite well but differently, since the results of the CNN models differ significantly. The McNemar's test shows that there exists a statistically significant difference in the models.

**RQ4**: How does image resolution affect the classification performance of a CNN model?

When comparing the results of CNNs, it can be seen that image resolution matters. The high-resolution dataset performs worst, and the super-resolution dataset performs best. These results are probably partially caused by the image size and batch size limitations faced during the training of the high-resolution and super-resolution CNN models. Another big problem was the lack of stratification of the datasets to improve the class imbalance. Solving memory issues, using less complex data, considering the class imbalance and enhancing CNNs may cause different results of CNNs and therefore future work is important to validate the results of this thesis.

As discussed at chapter 5.3, there are positive signs that a super-resolution diffusion model could be utilized as a data augmentation method based on the performance of super-resolution CNN compared to other CNN versions. However, there are many ways to improve and continue this thesis. The directions are related to the data, the diffusion model, the CNN models, changing part of the pipeline or the task and possible research questions. All those directions are discussed next.

Starting from data, to improve the classification capabilities of the CNN models, the initial dataset from which the three datasets are generated should suit the classification task better. Finding a dataset that isn't as hard but is still difficult in the classification aspect would support the emphasis on the differences of the three datasets. Also, the initial dataset should be a dataset meant for classification and contain more data to enable better training of CNNs. Alternatively, it could be created from the scratch by photographing different classes while being faithful to the suggestions just made.

The improvement to the classification capabilities of the CNN models could also have achieved by considering the imbalance in the classes of the datasets better. This could have been done by weighting the misclassification rate in the loss function in CNNs or using stratification while splitting the initial dataset into training, validation and test set.

The structures of the pipelines could also be altered. The modified SeeSR could be tested with more subjective methods, like LPIPS and DISTS, to determine the other aspects of the generalization capability of the model better. The detail generation capability of the model could be enhanced because it makes apparent mistakes with details when generating super-resolution images. Also, different general tag prompts could be used, or the current prompts could be polished to see if they had affected the generation capability of the model in a positive way.

The CNN models can be enhanced a lot. The original image size of the high-resolution images was 2040x2040 pixels but because of the memory limitations the smaller image size (1360x1360 pixels) was used during training super-resolution and high-resolution CNNs. If more GPU memory was available, the bigger image size could have been used for both high-resolution and super-resolution datasets. This change may emphasize the differences between the high-resolution and super-resolution datasets and potentially improve the performance of the models.

A critical improvement for high-resolution and super-resolution CNNs is to be able to use larger batch size in those models. Now both of their batch sizes are 18 which is extremely small for the purposes of this thesis. By increasing it, the performance of the models could potentially be improved or, at least, better understanding of the topic and models could be achieved. The hyperparameters of CNNs could be fine-tuned to better suit the datasets and to support the training of the models. For example, the learning rate could be optimized by changing it incrementally. Alternatively, a whole new structure for the CNN models could be discovered and used.

Another way to inspect the improvements of the pipelines is to use another diffusion model. There are a lot of new diffusion models that have emerged during the years 2024 and 2025 that could be used instead of SeeSR, such as CCSR or OFTSR. By replacing SeeSR model with another diffusion model, the detail generation performance of super-resolution model in the classification pipeline can potentially be increased, so that it improves the classification performance of super-resolution CNN. Another option is to change the latent text-to-image model, Stable diffusion, inside SeeSR with a newer model like one of the transformer-based models such as SDXL [74]. Also, this change has the potential to increase the performance of super-resolution CNN. The classification task can also be changed to another task like segmentation for testing the datasets through different means.

As implied the test arrangement is very limited. Only one CNN architecture is used with only one train-validation-test set split and with relatively small datasets and image sizes. Also, SeeSR was needed to apply as a pretrained version. Next, this test could be implemented as larger and more comprehensive way for gaining better understanding and more reliable results of the effects of super-resolution data as a data augmentation method.

The super-resolution data augmentation could also be combined and examined with other data augmentation methods. In this thesis data augmentation is made for the super-resolution

dataset in super-resolution CNN because of the small amount of data in the datasets. Different combinations of data augmentation methods could be tested as done in this thesis but in more throughout way by using only super-resolution data at first but adding more data augmentation methods later to compare the results and see only the impact of the super-resolution data.

Follow-up research questions could be: What is the lowest resolution of images that can be given to a diffusion model in a way that the diffusion model is able to make a sufficient super-resolution output image? How does incremental change of resolution of images affect to the quality of the super-resolution output images? For answering these questions, the resolution of low-resolution images could be started to increase, and the images and their super-resolution variants could be compared to each other and to other image pairs. With this it would be possible to pinpoint the sufficient low resolutions to generate sufficient super-resolution data.

In this thesis the super-resolution data is discussed in a way that it replaces the input data of a deep learning model entirely but there is an alternative way to work with it as well. Generated super-resolution data could be joined together with high-resolution data for increasing the amount of data available for deep learning model training. The research question related to that direction could be how well a model trained with super-resolution and high-resolution data performs compared to a model trained with only the high-resolution data.

## References

[1] Xin Li et al., “Diffusion Models for Image Restoration and Enhancement– A Comprehensive Survey” arXiv, Aug. 18, 2023. Accessed: Nov. 22, 2024. [Online]. Available: https://arxiv.org/abs/2308.09388v1

[2] Ruikai Zou and Yan Cang, “Super-resolution-based data augmentation algorithm for few-shot bearing defect detection” Proc. SPIE 13692, Fourth International Conference on Electronics Technology and Artificial Intelligence (ETAI 2025), 136927A, Jul. 24, 2025. Accessed: Sep. 24, 2025. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13692/136927A/Super-resolution-based-data-augmentation-algorithm-for-few-shot-bearing/10.1117/12.3068539.short

[3] Qiqi Zhu et al., “Super Resolution Generative Adversarial Network Based Image Augmentation for Scene Classification of Remote Sensing Images” IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium, Waikoloa, HI, USA, 2020, pp. 573-576. Accessed: Sep. 24, 2025. [Online]. Available: https://ieeexplore-ieee-org.ezproxy.utu.fi:2443/document/9324043

[4] Wentian Qu et al., “HOGSA:Bimanual Hand-Object Interaction Understanding with 3D Gaussian Splatting Based Data Augmentation” arXiv, Jan. 6, 2025. Accessed: Sep. 24, 2025. [Online]. Available: https://arxiv.org/abs/2501.02845

[5] Rongyuan Wu et al., “SeeSR: Towards Semantics-Aware Real-World Image Super-Resolution” arXiv, Jun. 4, 2024. Accessed: Jun. 23, 2025. [Online]. Available: https://arxiv.org/abs/2311.16518v2

[6] Lingchen Sun et al., “Improving the Stability and Efficiency of Diffusion Models for Content Consistent Super-Resolution” arXiv, Sep. 25, 2024. Accessed: Jun. 23, 2025. [Online]. Available: https://arxiv.org/abs/2401.00877v2

[7] Thaihoa1476050, “DF2K + OST” Kaggle, 2021. Accessed: Mar. 11, 2025. [Online]. Available: https://www.kaggle.com/datasets/thaihoa1476050/df2k-ost?select=train

[8] Bahadir K. Gunturk, and Xin Li, “Image Restoration : Fundamentals and Advances”. 1st ed., CRC Press, Jul. 11, 2012. Accessed: Nov. 25, 2024. [Online]. Available: https://doi.org/10.1201/b12693

[9] Ziwei Luo et al., “Taming Diffusion Models for Image Restoration: A Review” arXiv, Oct. 22, 2024. Accessed: Dec. 4, 2024. [Online]. Available: https://arxiv.org/html/2409.10353v2

[10] Brian B. Moser et al., “Diffusion Models, Image Super-Resolution And Everything: A Survey” arXiv, Jun. 23, 2024. Accessed: Dec. 5, 2024. [Online]. Available: https://arxiv.org/abs/2401.00736v3

[11] Juncheng Li et al., “ASystematic Survey of Deep Learning-based Single-Image Super-Resolution” arXiv, Apr. 12, 2024. Accessed: Nov. 25, 2024. [Online]. Available: https://arxiv.org/abs/2109.14335v2

[12] Hshmat Sahak, Daniel Watson, Chitwan and David Fleet, “Denoising Diffusion Probabilistic Models for Robust Image Super-Resolution in the Wild” arXiv, Feb. 15, 2023. Accessed: Nov. 25, 2024. [Online]. Available: https://arxiv.org/abs/2302.07864

[13] Eirikur Agustsson and Radu Timofte, "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study," IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 2017, pp. 1122-1131. Accessed: Oct. 10, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8014884

[14] Bee Lim et al., “Enhanced Deep Residual Networks for Single Image Super-Resolution” arXiv, Jul. 10, 2017. Accessed: Oct. 10, 2025. [Online]. Available: https://arxiv.org/abs/1707.02921

[15] Pablo Arbelaez, Charless Fowlkes and David Martin, “The Berkeley Segmentation Dataset and Benchmark” web page, Jun. 2007. Accessed: Oct. 15, 2025. [Online]: Available: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[16] Jia-Bin Huang, Abhishek Singh and Narendra Ahuja, “CVPR 2015 Open Access” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5197-5206, 2015. Accessed: Oct. 10, 2025. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2015/html/Huang_Single_Image_Super-Resolution_2015_CVPR_paper.html

[17] Marco Bevilacqua et al., “Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding” ResearchGate, Sep. 2012. Accessed: Oct. 10, 2025. [Online]. Available: https://www.researchgate.net/publication/260351242_Low-Complexity_Single_Image_Super-Resolution_Based_on_Nonnegative_Neighbor_Embedding

[18] Roman Zeyde, Michael Elad and Matan Protter, ”On Single Image Scale-Up Using Sparse-Representations” in: Boissonnat, JD., *et al.* Curves and Surfaces, 2010, Lecture Notes in Computer Science, vol 6920. Springer, Berlin, Heidelberg, 2012. Accessed:

Oct. 10, 2025. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-27413-8_47

[19] D. Martin et al., "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, Vancouver, BC, Canada, 2001, pp. 416-423 vol.2. Accessed: Oct. 10, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/937655

[20] Stanford Vision Lab, Stanford University, Princeton University. “ImageNet” web page, Mar. 11, 2021. Accessed: Nov. 27, 2024. [Online]. Available: https://www.image-net.org/index.php

[21] Ziwei Liu et al., ”Large-scale CelebFaces Attributes (CelebA) Dataset” web page, 2015. Accessed Oct. 10, 2025. [Online]. Available: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[22] Ian Goodfellow, Yoshua Bengio and Aaron Courville, “Deep Learning” 2016th ed., MIT Press 2016.

[23] Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9

[24] Stanford University, “CS231n Deep Learning for Computer Vision” web page, 2025. Accessed: Oct. 16, 2025. [Online]. Available: https://cs231n.github.io/convolutional-networks/

[25] Keiron O’Shea and Ryan Nash, “An Introduction to Convolutional Neural Networks” arXiv, Dec. 2, 2015. Accessed: Oct. 16, 2025. [Online]. Available: https://arxiv.org/abs/1511.08458v2

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation” arXiv, May 18, 2025. Accessed: Feb. 24, 2025. [Online]. Available: https://arxiv.org/abs/1505.04597

[27] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala, “Adding Conditional Control to Text-to-Image Diffusion Models” arXiv, Nov. 26, 2023. Accessed: Feb. 24, 2025. [Online]. Available: https://arxiv.org/abs/2302.05543v3

[28] Amith Kamath et al., “The impact of U-Net architecture choices and skip connections on the robustness of segmentation across texture variations” ScienceDirect, Oct. 2025. Accessed: Oct. 17, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482525014088?via%3Dihub

[29] Ryan Po et al., “State of the Arton Diffusion Models for Visual Computing” arXiv, Oct. 11, 2023. Accessed: Oct. 1, 2024. [Online]. Available: https://arxiv.org/abs/2310.07204

[30] Litao Hua et al., “Attention in Diffusion Model: A Survey” arXiv, Apr. 1, 2025. Accessed: Jun. 4, 2025. [Online]. Available: https://arxiv.org/abs/2504.03738

[31] Chitwan Saharia et al., “Image Super-Resolution via Iterative Refinement” arXiv, Jun. 30, 2021. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2104.07636v2

[32] Haoying Li et al., “SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models” arXiv, May 18, 2021. Accessed: Sep. 22, 2025. Available: https://arxiv.org/abs/2104.14951v2

[33] Jianyi Wang et al., “Exploiting Diffusion Prior for Real-World Image Super-Resolution” arXiv, Jun. 28, 2024. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2305.07015v4

[34] Jascha Sohl-Dickstein et al., ”Deep Unsupervised Learning using Nonequilibrium Thermodynamics” arXiv, Nov. 18, 2015. Accessed: Jun. 9, 2025. [Online]. Available: https://arxiv.org/abs/1503.03585

[35] Jooyoung Choi et al., “ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models” arXiv, Sep. 15, 2021. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2108.02938v2

[36] Robin Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models” arXiv, Apr. 13, 2022. Accessed: Jul. 1, 2025. [Online]. Available: https://arxiv.org/abs/2112.10752v2

[37] Shuyao Shang et al., “ResDiff: Combining CNN and Diffusion Model for Image Super-Resolution” arXiv, Feb. 2, 2024. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2303.08714v3

[38] Gianni Brauwers and Flavius Frasincar, “A General Survey on Attention Mechanisms in Deep Learning” arXiv, Mar. 27, 2022. Accessed: Feb. 25, 2025. [Online]. Available: https://arxiv.org/abs/2203.14263

[39] Meng-Hao Guo et al., “Attention Mechanisms in Computer Vision: A Survey” arXiv, Nov. 15, 2021. Accessed: Feb. 25, 2025. [Online]. Available: https://arxiv.org/abs/2111.07624

[40] Oskar Åström and Alexandros Sopasakis, ”Improved Anomaly Detection through Conditional Latent Space VAE Ensembles” arXiv, Oct. 16, 2024. Accessed: Mar. 3, 2025. [Online]. Available: https://arxiv.org/abs/2410.12328

[41] Bin Xia et al., “DiffIR: Efficient Diffusion Model for Image Restoration” arXiv, Aug. 16, 2023. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2303.09472v3

[42] Diederik P. Kingma and Max Welling, “An Introduction to Variational Autoencoders” arXiv, Dec. 11, 2019. Accessed: Feb. 27, 2025. [Online]. Available: https://arxiv.org/abs/1906.02691v3

[43] Jonathan Ho, Ajay Jain and Pieter Abbeel, “Denoising Diffusion Probabilistic Models” arXiv, Dec. 16, 2020. Accessed: Oct. 15, 2025. [Online]. Available: https://arxiv.org/abs/2006.11239v2

[44] Robin Rombach et al., “Stable Diffusion” GitHub, 2021, Accessed: Jul. 1, 2025. [Online]. Available: https://github.com/CompVis/stable-diffusion

[45] Qingguo Liu et al., “CDFormer: When Degradation Prediction Embraces Diffusion Model for Blind Image Super-Resolution” arXiv, Jun. 30, 2024. Accessed: Jul. 9, 2025. [Online]. Available: https://arxiv.org/abs/2405.07648v2

[46] Youcai Zhang et al., “Recognize Anything: A Strong Image Tagging Model” arXiv, Jun. 9, 2023. Accessed: Jul. 8, 2025. [Online]. Available: https://arxiv.org/abs/2306.03514v3

[47] Tao Yang et al., “Pixel-Aware Stable Diffusion for Realistic Image Super-Resolution and Personalized Stylization” arXiv, Jul. 9, 2024. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2308.14469v4

[48] Xintao Wang et al., “Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data” arXiv, Aug. 17, 2021. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2107.10833v2

[49] Xinqi Lin et al., “DiffBIR: Towards Blind Image Restoration with Generative Diffusion Prior” arXiv, Apr. 12, 2024. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2308.15070v3

[50] Zongsheng Yue, Jianyi Wang and Chen Change Loy, “ResShift: Efficient Diffusion Model for Image Super-resolution by Residual Shifting” arXiv, Oct. 18, 2023. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2307.12348v3

[51] Jie Liang, Hui Zeng and Lei Zhang, “Details or Artifacts: A Locally Discriminative Learning Approach to Realistic Image Super-Resolution” arXiv, Mar. 17, 2022. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2203.09195

[52] Jie Liang, Hui Zeng and Lei Zhang, “Efficient and Degradation-Adaptive Network for Real-World Image Super-Resolution” arXiv, Mar. 27, 2022. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2203.14216

[53] Kai Zhang et al., “Designing a Practical Degradation Model for Deep Blind Image Super-Resolution” arXiv, Sep. 30, 2021. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2103.14006v2

[54] Chaofeng Chen et al., “Real-World Blind Super-Resolution via Feature Matching with Implicit High-Resolution Priors” arXiv, Jul. 3, 2022. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2202.13142v2

[55] Lingchen Sun et al., “CCSR Improving the Stability and Efficiency of Diffusion Models for Content Consistent Super-Resolution” GitHub, Dec. 12, 2024. Accessed: Jul. 7, 2025. [Online]. Available: https://github.com/csslc/CCSR

[56] Kelvin C.K. Chan et al., ”BasicVSR: The Search for Essential Components in Video Super-Resolution and Beyond” arXiv, Apr. 7, 2021. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2012.02181v2

[57] Rui Xie et al., “AddSR: Accelerating Diffusion-based Blind Super-Resolution with Adversarial Diffusion Distillation” arXiv, Dec. 27, 2024. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2404.01717v4

[58] Yu Yu, Weibin Zhang and Yun Deng, “Frechet Inception Distance (FID) for Evaluating GANs” ResearchGate, Sep. 1, 2021. Accessed: Jul. 31, 2025. [Online]. Available: https://www.researchgate.net/publication/354269184_Frechet_Inception_Distance_FID_for_Evaluating_GANs

[59] Sadeep Jayasumana et al., “Rethinking FID: Towards a Better Evaluation Metric for Image Generation” arXiv, Jan. 25, 2024. Accessed: Jul. 31, 2025. [Online]. Available: https://arxiv.org/abs/2401.09603v2

[60] Margherita Grandini, Enrico Bagli and Giorgio Visani, “METRICS FOR MULTI-CLASS CLASSIFICATION: AN OVERVIEW” arXiv, Aug. 13, 2020. Accessed: Aug. 4, 2025. [Online]. Available: https://arxiv.org/abs/2008.05756

[61] Alhassan Mumuni and Fuseini Mumuni, “Data augmentation: A comprehensive survey of modern approaches” ScienceDirect, Nov. 15, 2022. Accessed: Aug. 7, 2025. Available: https://www.sciencedirect.com/science/article/pii/S2590005622000911

[62] Teerath Kumar et al., “Image Data Augmentation Approaches: A Comprehensive Survey and Future directions” arXiv, Mar. 12, 2023. Accessed: Aug. 7, 2025. [Online]. Available: https://arxiv.org/abs/2301.02830v4

[63] Li Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]" IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141-142, Nov. 2012. Accessed: Sep. 22, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/6296535

[64] Jason Ansel et al., “Pytorch” web page, 2024. Accessed: Oct. 28, 2025. [Online]. Available: https://pytorch.org/

[65] Martín Abadi et al., “Tensorflow” web page, 2015. Accessed: Oct. 28, 2025. [Online]. Available: https://www.tensorflow.org/

[66] Chollet, F. et al., “Keras: Deep Learning for humans” web page, 2015. Accessed: Oct. 28, 2025. [Online]. Available: https://keras.io/

[67] Rongyuan Wu et al., “SeeSR: Towards Semantics-Aware Real-World Image Super-Resolution” GitHub, 2024. Accessed: Jan. 13, 2025. [Online]. Available: https://github.com/cswry/SeeSR/tree/main

[68] EricShenYuQi, Chunming He and Chengyu Fang, “Awesome Diffusion Models in Low-level Vision” GitHub, Feb. 24, 2025. Accessed: Dec. 2, 2024. [Online]. Available: https://github.com/ChunmingHe/awesome-diffusion-models-in-low-level-vision?tab=readme-ov-file

[69] Yuanzhi Zhu et al., “OFTSR: One-Step Flow for Image Super-Resolution with Tunable Fidelity-Realism Trade-offs” arXiv, Sep. 3, 2025. Accessed: Sep. 22, 2025. [Online]. Available: https://arxiv.org/abs/2412.09465v2

[70] Stéfan van der Walt et al., “scikit-image: Image processing in Python” web page, Feb. 18, 2025. Accessed: Sep. 22, 2025. [Online]. Available: https://scikit-image.org/

[71] Tensorflow, “Mixed precision” web page, Mar. 23, 2024. Accessed: Aug. 18, 2025. [Online]. Available: https://www.tensorflow.org/guide/mixed_precision

[72] Keras, “Image augmentation layers” web page, 2025. Accessed: Aug. 4, 2025. [Online]. Available: https://keras.io/api/layers/preprocessing_layers/image_augmentation/

[73] Thomas Dietterich, “Approximate statistical tests for comparing supervised classification learning algorithms” MIT Press, Neural Computation, vol. 10, no. 7, pp. 1895–1923, Oct. 1, 1998. Accessed: Nov. 4, 2025. [Online]. Available: https://dl.acm.org/doi/10.1162/089976698300017197

[74] Dustin Podell et al., “SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis” arXiv, Jul. 4, 2023. Accessed: Oct. 31, 2025. [Online]. Available: https://arxiv.org/abs/2307.01952

# Appendixes

## Appendix 1. Example Images of Classes in the Datasets

Some example images from human and non-human classes are presented here from all three datasets: high-resolution, low-resolution and super-resolution. The example images are high-resolution variants to give better understanding of the content of the images and the datasets. All images are from DF2K dataset. First comes five human class examples followed by five non-human class examples.


Figure 32. Humans posing to the camera.


Figure 33. Humans sitting behind bushes.

Figure 34. Humans in front of building.

Figure 35. Human peeking behind a tree.

Figure 36. Human behind a bush on the left in the image.

Figure 37. Swimming pool surrounded by palm trees.

Figure 38. Statue of humans.

Figure 39. Ruined ancient building.

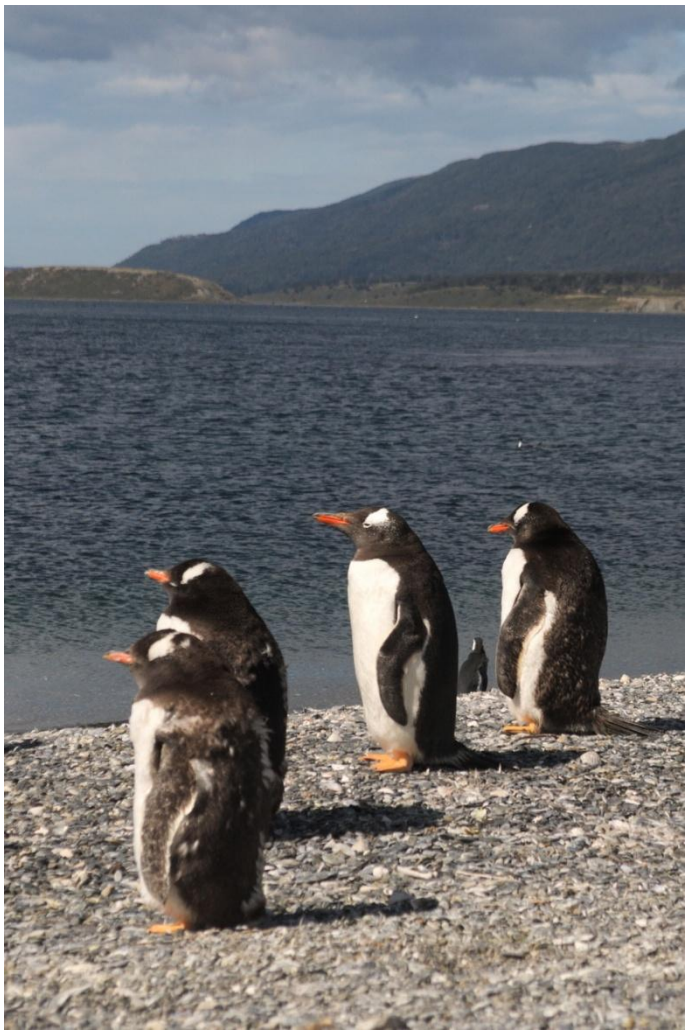

Figure 40. Penguins on the shore of an ocean.

Figure 41. Old bridge in the wilds.

## Appendix 2. Example Images of SeeSR and the Modified SeeSR

Three example images made by SeeSR and the modified SeeSR can be found here in respective order. The images can be compared to see the differences between the models.

Figure 42. Output image of original SeeSR.

Figure 43. Output image of the modified SeeSR.

Figure 44. Output image of original SeeSR.

Figure 45. Output image of the modified SeeSR.

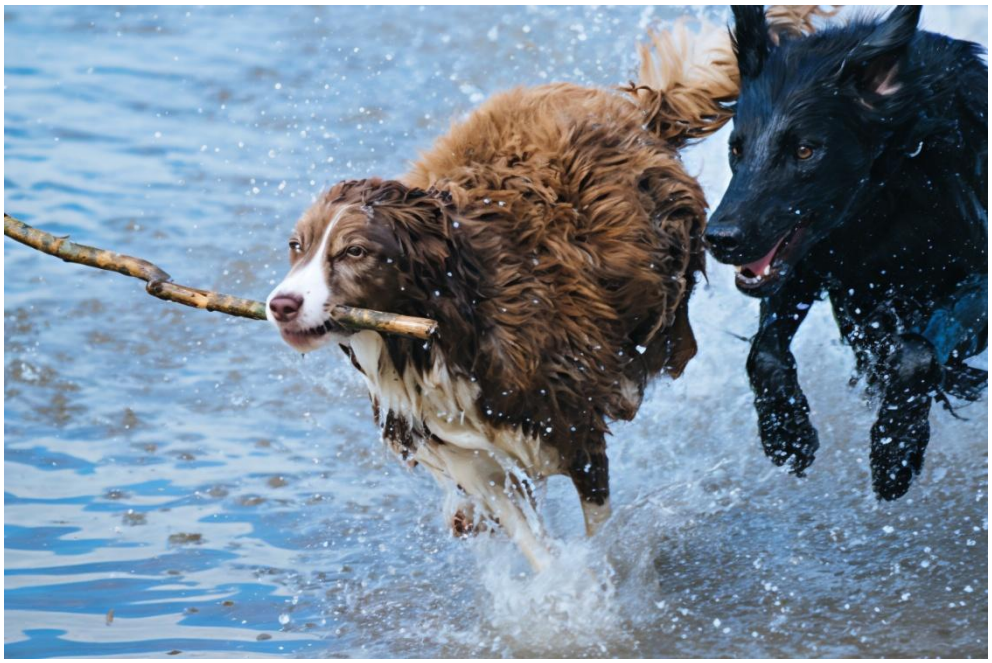

Figure 46. Output image of original SeeSR.

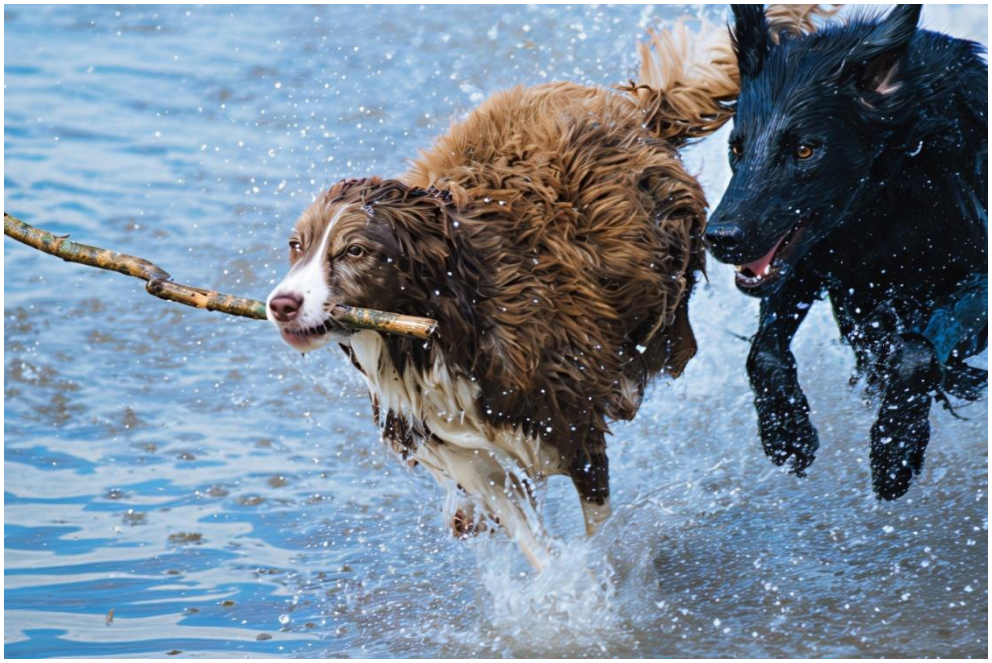

Figure 47. Output image of the modified SeeSR.

Figure 48. Output image of original SeeSR.

Figure 49. Output image of the modified SeeSR.

Figure 50. Output image of original SeeSR.

Figure 51. Output image of the modified SeeSR.

Figure 52. Output image of original SeeSR.

Figure 53. Output image of the modified SeeSR.

## Appendix 3. Example Images of Non-Human Class with Different Resolutions

Five example images of non-human class from high-resolution, super-resolution and low-resolution datasets, respectively, can be found here, so that the images can be compared to gain a better understanding of the quality of the images. The high-resolution images are the ones provided in Appendix 1 and the images from other datasets are same images but with different resolutions.

Figure 54. High-resolution image of a swimming pool area.

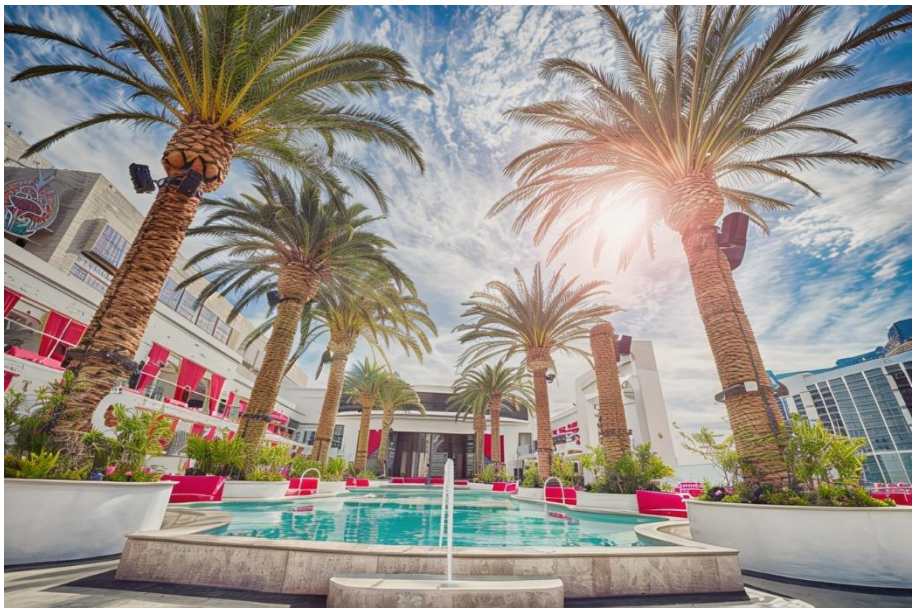

Figure 55. Super-resolution image of a swimming pool area.

Figure 56. Low-resolution image of a swimming pool area.

Figure 57. High-resolution image of a statue of humans.

Figure 58. Super-resolution image of a statue of humans.

Figure 59. Low-resolution image of a statue of humans.

Figure 60. High-resolution image of a ruined ancient building.

Figure 61. Super-resolution image of a ruined ancient building.

Figure 62. Low-resolution image of a ruined ancient building.

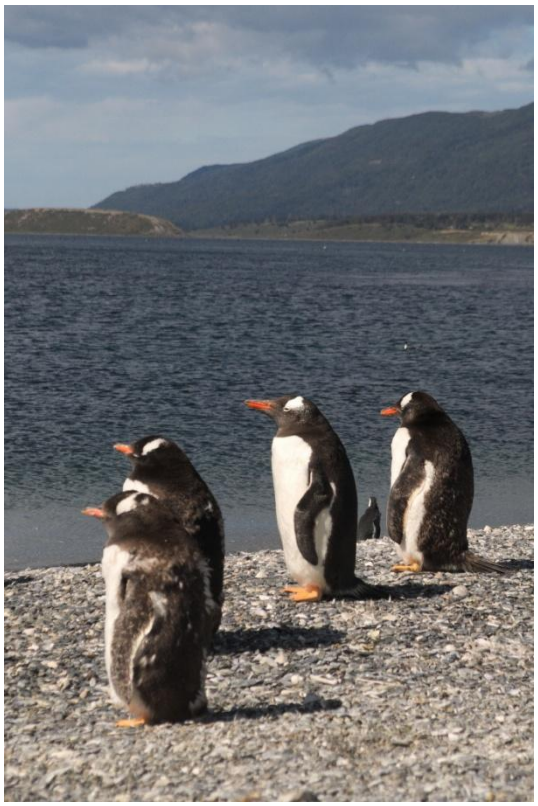

Figure 63. High-resolution image of penguins on the shore of an ocean.

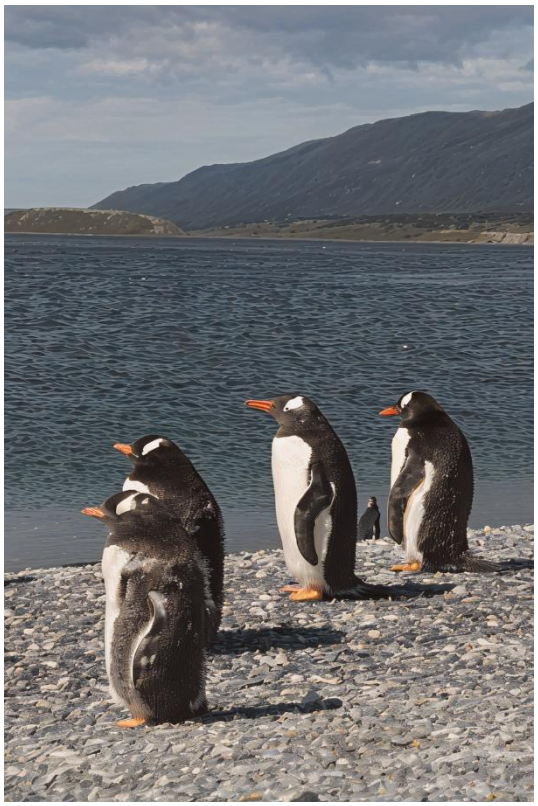

Figure 64. Super-resolution image of penguins on the shore of an ocean.

Figure 65. Low-resolution image of penguins on the shore of an ocean.

Figure 66. High-resolution image of an old bridge in the wilds.

Figure 67. Super-resolution image of an old bridge in the wilds.

Figure 68. Low-resolution image of an old bridge in the wilds.