Faculty of Fundamental Problems of Technology
Wrocław University of Science and Technology

# Analysis of Super-resolution methods

Bartosz Rajczyk

Master's thesis
supervised by
Piotr Syga, PhD

Politechnika
Wrocławska
Wrocław 2022

# Contents

# Introduction

Image interpolation techniques have been a topic of research for several decades and has gone through developement of simple mathematical model, through more sophisticated techniques and reach its current peak-performance state with machine learning approaches. It is of no suprise, as the need for upscaling images became more and more critical in the recent years, in the age of the Internet and the social media, due to heavy downsizing and compression applied by the most popular social networks. Those services try to optimize for the data size and transfer speeds, usually delivering multiple sizes of the media depending on user's needs. However, in many cases, images (usually photos) stored there are the only copies left for their owners, who are thus left with low-quality, degenerate versions of them. The most recent approaches [5, 27, 14, 9, 16, 23] promise to overcome this issue, by interpolating the missing details and resolution by using networks trained of thousands of examples, making them aware of the photos' context and not solely rely on the mathematical properties.

This thesis strives to introduce the most popular interpolation methods, describe how they work, present results produced by them and then compare them to their recent, machine learning based counterparts. The design and concepts behind a *super resolution* neural network are described and its performance evaluated against multiple loss functions. A simple yet robust network architecture is introduced to achieve this.

There are two unique aspects of this analysis. Firstly, the dataset used is completely proprietary, created by the author from his original work. This has multiple benefits which are discussed later. Secondly, the results are evaluated not only by artifical scores like mean square error or structural similarity, but also by two additional approaches, machine learning based and human based. Both professional photographers and casual users were queried to achieve the most impartial judgement.

# Theoretical background

This chapter focuses on describing the theoretical basis behind the previous approaches as well as the machine learning ones. It gives several examples of both of them with brief descriptions.

## 2.1 Image representation

Image interpolation and super resolution is done for *raster images*, which is a matrix of *pixels* (picture elements), which are the smallest piece of color information available in the image. The pixel is defined as a square for this thesis's applications. A single image is a rectangle of a particular *resolution*, which is the number of pixels horizontally and vertically. Each pixel represents a color, which is coded as an intensity of there base colors - red, green and blue - between 0 (representing none of a particular color) and 255 (representing all of particular color), which is an 8-bit coding.

## 2.2 Classical approaches

Or non-machine learning approaches, often called just image interpolation methods not to be confused with super-resolution methods. They rely on mathematical models which became more elaborate over the years. This chapters describes them briefly and gives several examples [10].

### 2.2.1 Nearest-neighbor interpolation

This is one of the simplest interpolations methods available. For image to be upscaled, it projects the original image pixels' as points located at their original positions, applies a linear transformation scaling the image to the desired length and overlays it with a pixel grid of the new resolution. Then, for each new pixel, a color of the nearest old but scaled pixel is used. The process is illustrated by figure 2.1.

### 2.2.2 Linear interpolation

Linear interpolation is created by constructing a linear function connecting color values of each neighboring pixel on $2 \times 2$ grid and deciding a new color by evaluating this function using the pixel's coordinates to get its new color. Figure 2.2 illustrates this process.

### 2.2.3 Cubic interpolation

Cubic interpolation works similarly to the linear interpolation, but constructs a cubic function on a $4 \times 4$ grid, resulting in smoother-looking edges. The simple examples given for two previous interpolation

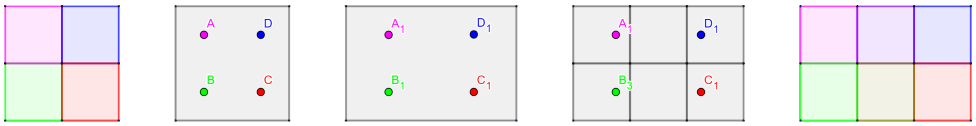



Figure 2.1: **Nearest-neighbor interpolation.** Firstly, the original four-pixel image is shown, then the colors are mapped to the center coordinate of each pixel, then the coordinate system is resized, a bigger pixel grid is applied and the nearest pixel color is used. You may notice that the two middle pixels are the same distance from two points; one of them have to be chosen and this is decided by a concrete implementation of the algorithm.

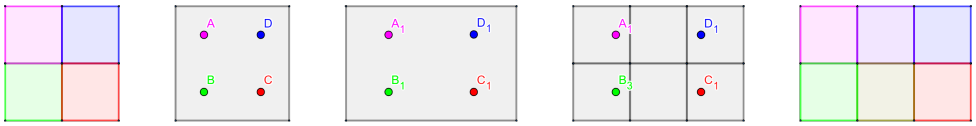



Figure 2.2: **Linear interpolation.** It proceeds similarly to figure 2.1, but the two middle pixels, each being the same distance away from their outermost neighbors, receive a color which is an arithmetic average of them

Figure 2.3: **Interpolations comparison.** Beginning on the left, the original picture is shown; then, in order, nearest-neighbor, linear and cubic interpolation for upscaling the image 300%. Notice the differences in the feeling of *smoothness* of the methods.

methods are thus not large enough to include the cubic interpolation, but figure 2.3 illustrates it on a regular picture.

### 2.2.4 Other interpolations

Apart from those mentioned, there are many more, less known, interpolation methods available, examples being Hanning, Lanczos or Hermite interpolation. They are omitted for sake of this research, as they are not widely available to the average consumer in leading image processing software.

### 2.2.5 Additional image enhancement

Interpolation is not the sole process that the upscaled images are put through when using a more sophisticated software. Additional methods can be used to make the picture appeal better to the viewer, such as additional sharpening or anti-aliasing. Some software decide the interpolation method based on the image contents. Today, however, many of those things are machine-learning supported when automated, and quite tedious to deal with manually (as they require the user to adjust multiple parameters, different for each image), and thus will be excluded from this research.

## 2.3 Machine learning approaches

Image super resolution differs significantly from image upscaling. All of the classical methods work based solely on the information contained within the input; the machine learning methods may leverage additional "knowledge" that is stored within them. The focus of this thesis are neural networks, which have shown considerable performance characteristics and good results since their conception.

It is important to note that the problem is ill-posed, meaning that there are multiple correct solutions to adding the missing detail to the picture, as, by concept, in evaluation phase there exists no high resolution version of the input.

### 2.3.1 Neural networks concepts

#### Neural network

A *neural network* is a *machine learning algorithm*, meaning that it is a method which iteratively improves itself based on the data which is fed into it. Neural networks were created based on real life workings of a brain; they consist of interconnected *neurons*. Each neuron takes some number of weighted inputs, sums them and evaluates this sum in an *activation function*, which is then given as an output of

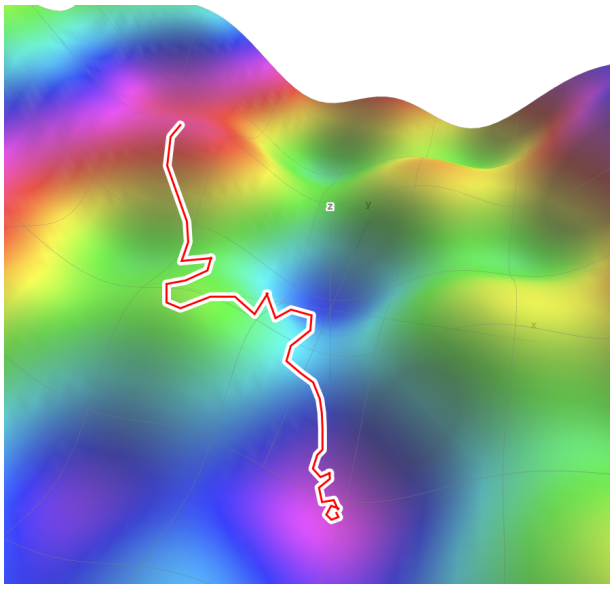


Figure 2.4: **Stochastic gradient descent example.** The loss function value is plotted and each next "move" of the red line represents the change of the value, searching for the minimum. Red color at the top indicates higher loss function values, purple at the bottom - lower. The descent tends towards the lower values.

this neuron. This output can be then connected as a weighted input to the next neurons. Neurons are organized in *layers*. There are at least two layers in a network, the input and the output, but usually there are multiple *hidden layers*, called like that because the end user does not access those layers directly, in between them, handling the main workload of the network.

Execution of the neural network begins with a *forward propagation*, which is feeding input through each layer, evaluating each of the neurons in series. Then, the output $Y'$ is compared against the expected output $Y$ using a *loss function*. This function can differ a lot depending on the task the networks aims to achieve. It is usually a mean absolute error, mean squared error or cross entropy loss. The process of learning the neural network tries to minimize the loss function for given inputs. To achieve this, it uses a *gradient descent* algorithms. The last step is back propagation, which pushes the error calculated by the loss function through the network backwards, starting from the outputs, and adjusts the weights of the neural inputs on each layer [19].

#### Stochastic gradient descent

As mentioned, the goal of a neural network training is to minimize an objective loss function $J$, which takes a multidimensional input $\Theta \in \mathbb{R}^d$. To achieve this, an iterative method called stochastic gradient descent is usually used. It updates the $\Theta$ iteratively, taking an additional parameter $\rho$ called learning rate. For training example $\omega \in \mathbb{R}^{d-1}$ from $\Theta$, it uses following formula to update its value in each step

$$\omega := \omega - \eta \cdot \nabla J(\omega)$$

This slowly "descents" over the gradient of the loss function to reach its candidate minimum, shown in figure 2.4. This process involves avoiding local minima and converging as fast as possible. Multiple gradient descent optimizers are used for this purpose, including the most popular Adagrad, Adadelta or Adam.

There are also other approaches to the minimization problem, like batch gradient descent, which uses the entire dataset $\Theta$ to perform each step, but the SGD is preferred due to large sizes of the input to machine learning applications and making the solution to the problem an online-algorithm, which is better in distributed systems [22].

### 2.3.2 Neural network types

Across the years, multiple network architectures have been proposed and tested. This is a brief summary of most significant designs.

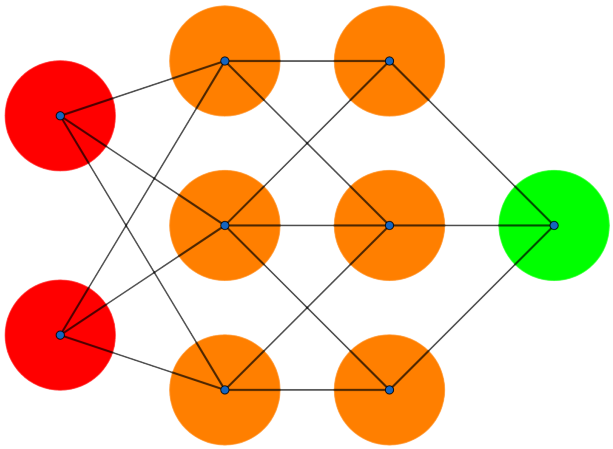


Figure 2.5: **Feed-forward network.** Two-neuron input layer, two three-neuron hidden layers and one-neuron output layer.

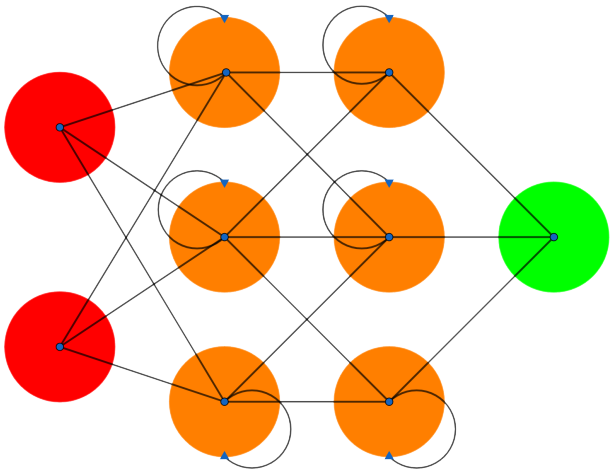


Figure 2.6: **Recurrent network.** Two-neuron input layer, two three-neuron hidden layers with recurrency and one-neuron output layer.

### Feed-forward networks

Feed-forward networks are the most basic neural networks than can be constructed [7]. They consist of basic neural layers which do none more than take the input in, calculate the activation function and feed it foward. When using multiple hidden layers, they can be called *deep feed-forward networks*. A simple feed-forward network can be seen in the figure 2.5.

These network were used primarly in the early days of machine learning research, later replaced by more specialised ones.

### Recurrent networks

Recurrent networks introduce a concept of *recurrency* in some of its layers; apart from feeding the neurons' results forward, they are re-fed as input to the neurons themselves after some delay [7]. Data flow in the network is illustrated by figure 2.6.

These networks are used mainly when dealing with temporal data, like time series, as they are aware of the order the data comes in.

### Convolutional network

These networks are characterized by having at least a single *convolutional* layer s [12]. This layer type is responsible for applying convolution to data, meaning that it goes over the original input matrix with a *kernel*, calculating dot-product of it and sub-matrices, both treated as vectors, constructing a *feature map*. A graphic representation of a single step of a convolution is given in figure 2.7.

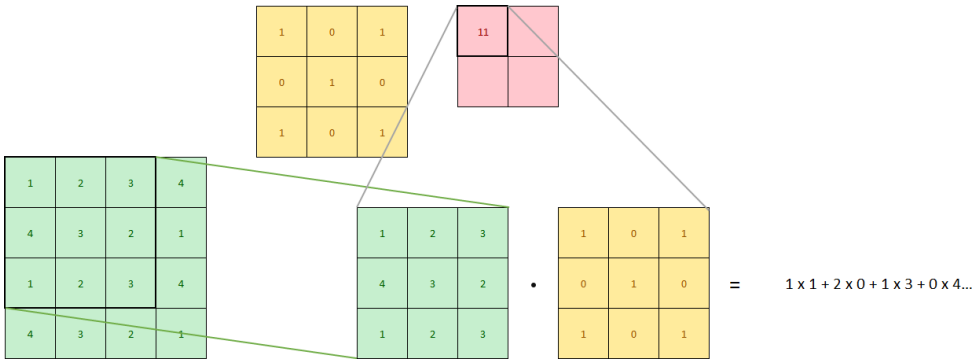



Figure 2.7: **Convolution.** A sub-matrix is chosen from the input (in green), which is treated like a vector before calculating a dot product, which is put in the output (in red). This will be repeated for times by moving the sub-matrix window one row right and then repeating this for one column below.

Convolutional networks found widespread use in image and audio processing; the convolution step is critical in recognizing patterns and important features of the dataset.

### Autoencoders

The main idea behind autoencoders is to create a network which specializes in constructing a data-specific encoding which captures the most important features of the dataset and then decoding it to retrieve as much information from the limited output [12]. The latent vector representation of the encoder part is the focus of an autoencoder; it can be modified before being fed to the decoder to achieve almost continuous results. Figure 2.8 illustrates a simple autoencoder.

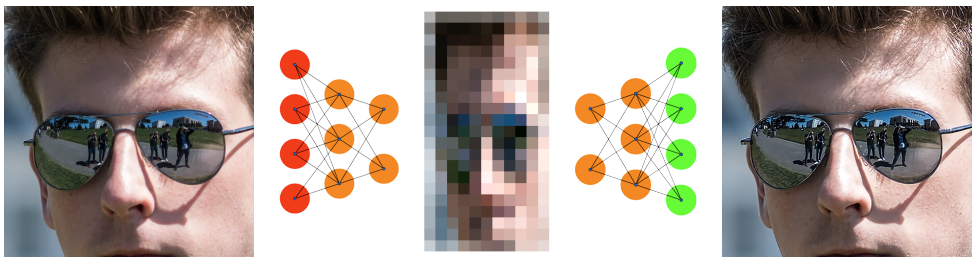


Figure 2.8: **Autoencoder network.** A simplistic illustration of an autoencoder and its main components. The left hand side shows encoding process, the middle is the latent vector and the right hand side - decoding.

### Generative adversarial networks

Introduction of the GANs was a major breakthrough in the computer vision applications, as they allowed to create almost photorealistic images when used in generative imagining [7]. The main idea behind a GAN is to create not one, but two network which are competing against each other. Those are

- **the generator** - a network responsible for creating the results; those can be for example the upscaled images
- **the discriminator or the adversary** - a network responsible for telling if the result it is fed comes from the generator or the original dataset

This creates a continuous rivalry between the two, each one getting better at its task.

# Related work

This chapter introduces other designs and briefly describes them.

## 3.1 Machine learning super resolution progress

First learning based approaches emerged around 2000s, with notable examples being "Example Based super-resolution" [11] from 2002, which relied on reconstructing images using high-resolution texture patches [25, 24].

## 3.2 Examples

Multiple works from recent years have been chosen for comparison and evaluation against the proposed solution, varying both in complexity and performance.

### 3.2.1 SCRNN

Sometimes referred to as a benchmark for future solutions [24], SCRNN is a convolutional network with three layers, trained using MSE loss function [8]. It uses cubic-interpolated low resolution to high resolution input, similar to the solution of this paper.

### 3.2.2 SRGAN

SRGAN, as the name suggests, is a generative adversarial network, with a generator part created from 16 residual blocks (which are two convolution layers, batch normalization layer and ReLU activation function, with skip connections from the start to the end added) and a convolutional discriminator [16].

### 3.2.3 VDSR

It is based on VGG, a convolutional neural network with multiple convolutions followed by max pooling with additional fully connected layers at the end [14]. It uses relatively high learning factors and gradient clipping (restricting the output of the network to some fixed interval) to prevent gradient explosion (unwanted divergance from the result). It also uses interpolated low resolution to high resolution as an input.

### 3.2.4 EDSR

The main idea behind building the EDSR was to use the residual blocks introduced for image recognition tasks [13] and modify them to better fit the super resolution use case [17]. The original ResNet architecture assumed a residual block containing two convolutions followed by batch normalization layers, with a single ReLu activation between those pairs. EDSR proposed solution was to remove the normalization step. The paper assumes that this improves flexibility of the network and thus - the results.

### 3.2.5 ESRGAN

An abbreviation for "Enhanced SRGAN", it uses the similar ideas like previously described SRGAN paper, but improves it by using different type of residual blocks, similar to those found in EDSR, described above [23]. Apart from removing the batch normalization layer, it also introduces additional residual skips to the blocks, which allow to skip to any following layer in the block.

### 3.2.6 NLSN

An abbreviation for "Non local sparse attention (network)", this is one of the most novel designs, introduced at CVPR 2021 [20]. Two main concepts behind this design are sparsity constraints and non local attention. The prior is defined as limiting the number of maximum active neurons on particular layers of the network and is quoted to be a powerful tool driving many image reconstruction models. The non local prior is about looking for patterns in images globally and registering those similarities across all the dataset. It is computationally expensive, but delivers robust results.

# Proposed implementation

This chapter focuses on the technical and concrete side of the thesis. It covers the dataset, the network architecture and the preprocessing done before training.

## 4.1 Dataset description

The dataset for super resolution imaging must contain both high resolution and low resolution pictures. The number of different pictures will affect the network's ability to learn and adapt to novel inputs.

### 4.1.1 Contents

A unique opportunity for this thesis is an original dataset provided by the author. It contains over 1000 very high resolution images taken with a Sony A7rIV, creating 61 megapixel 14-bit raw images, using varying lenses. Those were individually developed in a dedicated software, Adobe Lightroom Classic [3], into 8-bit highest-quality JPEGs. Example images from the dataset can be seen in figure 4.1.

Choosing custom dataset over the readily available has multiple benefits, among them:

- it can easily be extended with more similar data
- it can be preprocessed in any other way desired by the user
- having the 14-bit raw files, it can be used for a later work focused on recreating them from the 8-bit source

### 4.1.2 Preprocessing

#### Data generation

The data has been pre-filtered manually to discard some of the similar images. Then the images were fed into a preprocessing script, which generates two parts of the dataset:

- **high res images** - resolution of 2016 by 2016 pixels, a middle square of a picture, no compression
- **low res images** - resolution of 992 by 992 pixels, smiliar area, jpeg compression level 80

Those sizes seem odd at first, but they take padding added later into account; all added padding makes them 2048 and 1024 respectively. The exact difference between high res and low res sets is illustrated by figure 4.2. This means that the entire data fed into the network was size of approximately $1050 \times (2048 \times 2048 \times 3 + 1024 \times 1024 \times 3)$, making it around 16.5 billion individual data points.

#### Data loading

The data is loaded and represented as a mutli-dimensional array of 32-bit floating point values, each representing single colour value of a single pixel, 0.0 being black and 1.0 being fully bright. Because of the size of the data, a lazy loading mechanism had to be implemented, which loads data in chunks when a particular picutre is needed.

1. Large, uniform surfaces

2. People

3. Sharp edges

4. Dense, detail-rich textures

Figure 4.1: **Examples of pictures in the dataset**

### Preparing for training

Due to limited resources available for this thesis, a network able to upscale image by the factor of 4 could not have been trained even on the 12 GB VRAM 10240 CUDA cores 320 TPU cores Nvidia RTX 3080 ti. Instead, a different approach was chosen. Each picture was split in 64 tiles ($252 \times 252$ pixels for high res, $124 \times 124$ in low res). The resulting tiles were then used for training purposes. For each tile, there's a two-pixel padding added on each of the edges, making them $256 \times 256$. Those additional padding pixels are creating by mirroring the exiting image from each side. Those serve as a border which is then cut away when gluing them together for a reconstruction; this prevents grid-like artifacts from happening.

One could ask why was it so important to keep the input $256 \times 256$. This is important when trying to speed up the training process. Modern GPUs contain TPUs, tensor processing units, which are specialized in matrix multiplication operators, making them orders of magnitude faster. TPUs are designed by multiply $4 \times 4$ 16-bit float matrix in a single processor cycle, but also require that the data sizes fed into the network are divisible by 8 [2]. There was a twofold increase in measured training speed when switching to TPU-based computations in this case.

Upscaling the test images is done the same way, by loading an image, splitting it into tiles, predicting an upscaled version of them and reconstructing a full picture to return to the user. This is illustrated by figure 4.3.

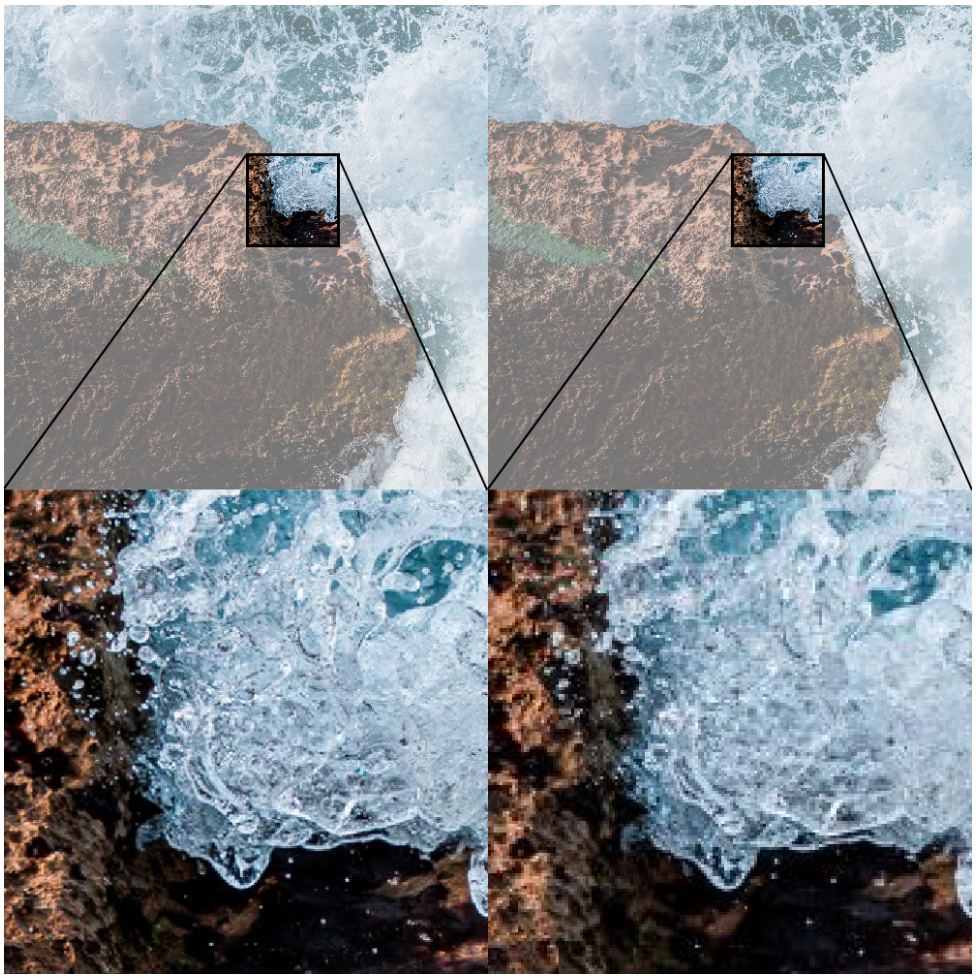


Figure 4.2: **High res and low res comparison.** Notice the detail difference in the crops below.

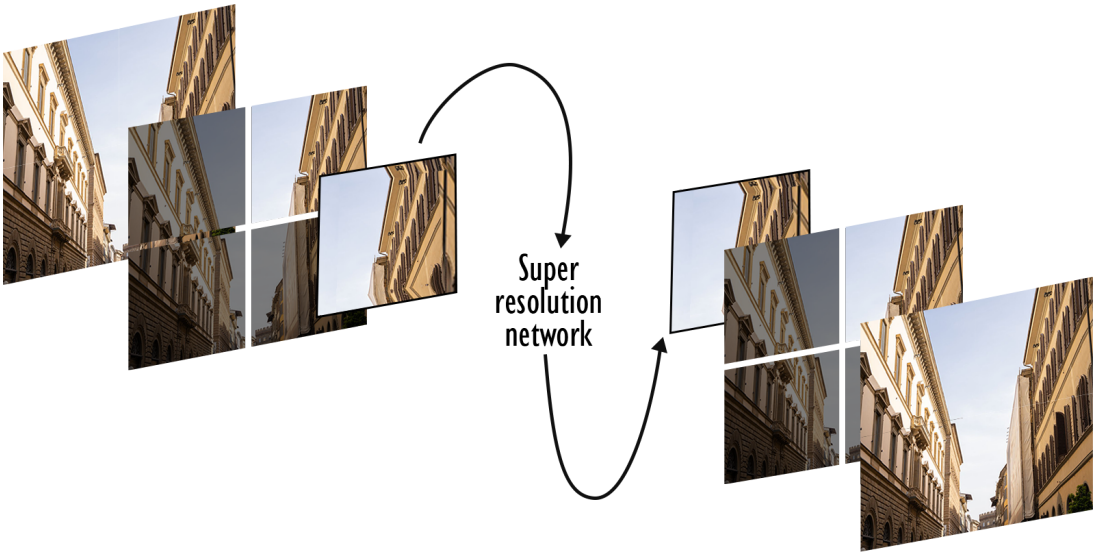



Figure 4.3: **Data flow in the network.** This is a simpler example in which the image is split into 4 tiles; in actual model, each image is split in 64 tiles.

## 4.2 Network architecture

### 4.2.1 Overview

The architecture of choice is an convolutional autoencoder. This is due to its properties which are desirable when it comes to this paper's case of super resolution, including

- Autoencoders specialize in "compressing" the data using only the most important features and then decompressing them as faithful as possible. This is the exact property needed for upscaling the images; the low resolution data contains only limited amount of information, of which the compression artifacts and noise is to be omitted and the shape or colour data - brought forward.
- Convolution is a very important step when working with image data; the feature maps it creates are crutial for an ability to recognize patterns, which are learned by the network and applied accordingly.
- Other popular state-of-the-art architectures such as GANs are resource-intensive and highly unstable when training [15]. Due to nature of this research, a stable and resource-efficient network architecture was preferred. This allows for experimental use at home and omits need for the use of specialized computational systems.

The convolutional autoencoder enables good results in resource-aware fashion, as can be seen in Section 5.

### 4.2.2 Description

A precise description of the architecture used by the paper is show in figure 4.4. The first step, not shown in the picture, is a naive upscaling done by a simple cubic interpolation; the input to the network is shaped similarly to the output, $256 \times 256$ pixels. This makes the learning process faster as the network does not have to learn how to upscale the image, only how to create back the missing details. Following this, a series of convolutional layers creating feature maps from which a max pooling layer chooses the most important ones. This reduces the dimensionality one time to $128 \times 128$ and then to $64 \times 64$. Then there is a decoding phase, which contain two transposed convolution layers, upscaling the feature space. A simple upscaling layers could be used, but those are not trainable and would not benefit the final result. Finally, two residual connections are added, which should help with the noise created by the decoder.

## 4.3 Training and testing environment

### 4.3.1 Physical environment and software

The model was run in Keras with Tensorflow backend version 2.9.0, running on Python 3.9.12 on 64-bit Windows 11, 24-core AMD Ryzen 5900X CPU, 64 GB of RAM and Nvidia RTX3080Ti. The training of all considered models took several hours and changed according to the optimal stopping estimation for each of the loss functions.

### 4.3.2 Loss function choice

#### Mean squared error

Abbreviated MSE, is the basic loss function of choice for many machine learning applications. It calculates a square of a difference between the predicted and actual vectors. Let $Y$ be the actual result and $Y'$ the predicted data, $n$ be the number of data points, then

$$\mathrm{MSE}(Y, Y') = \frac{1}{n} \sum_{i=1}^{n} (Y_i - Y_i')^2$$

is the metric used as a loss function. This is good enough for basic applications of the network, however this metric is hardly related to the way that real people perceive image similarity.

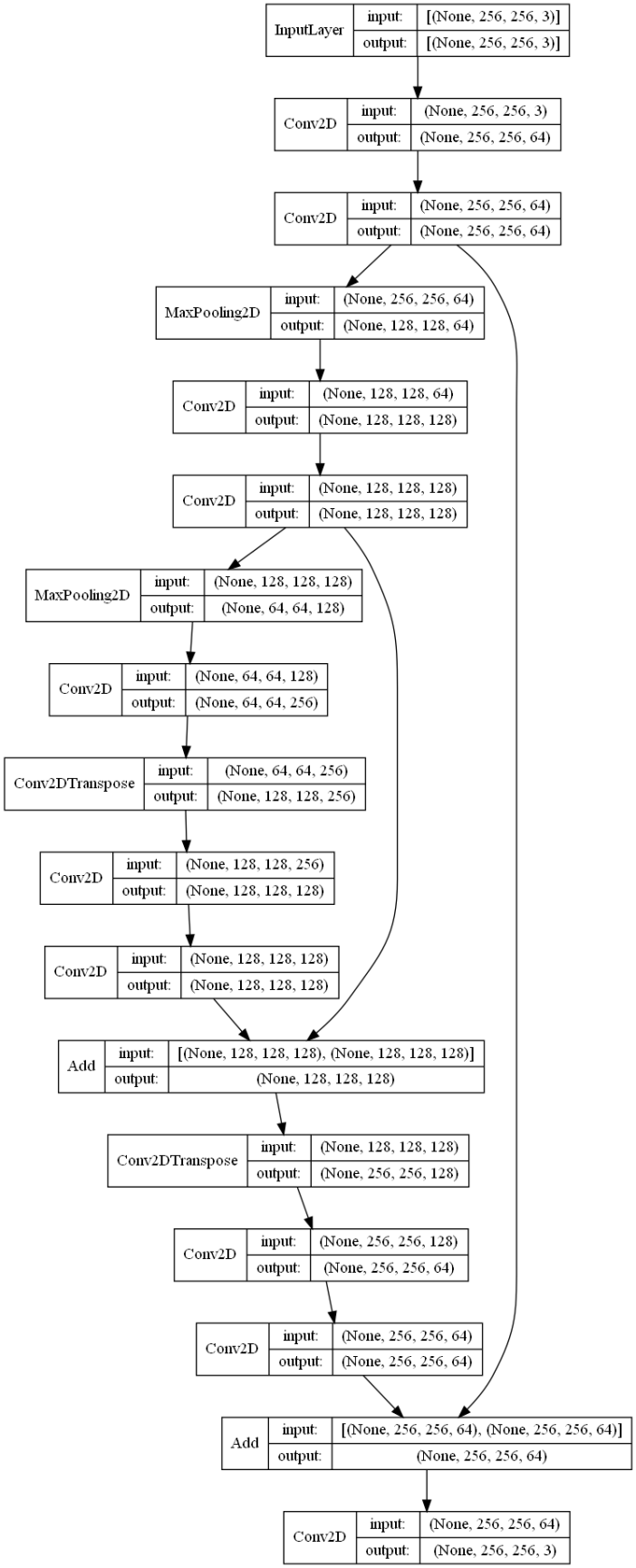



Figure 4.4: **Summary of the model**, described above. Autogenerated by `tf.keras.plot_model`.

**Peak signal-to-noice ration**

Abbreviated PSNR, measures the ratio between the strength of a signal (the image) and noise (upscaling artifacts). For the networks use case, it is defined as

$$\mathrm{PSNR}(Y, Y') = 10 \cdot log_{10} \frac{1}{\mathrm{MSE}(Y, Y')}$$

The 1 in the nominator is actually the maximum value of the input, but in case of the 32-bit float images, this is always 1. Because the MSE and PSNR are proportional, only MSE will be tested.

**Structural similarity**

Abbreviated SSIM, is used to compare images in terms of structural changes, meaning local contrast or brightness. It is defined as

$$\mathrm{SSIM}(Y, Y') = \frac{(2\mu_Y \mu_{Y'} + c_1)(2\sigma_{YY'} + c_2)}{(\mu_Y^2 + \mu_{Y'}^2 + c_1)(\sigma_Y^2 + \sigma_{Y'}^2 + c_2)}$$

where

$$c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$$

and $k_1$, $k_2$ and $L$ are constant, $k_1 = 0.01$, $k_2 = 0.03$ as suggested by the original paper and $L = 2^8 - 1$, which is the dynamic range of the pixel values.

**Perceptual loss**

The perceptual loss is a metric which aims to predict how a real human would rate the result. This paper's approach is based on a previous work [26], which discovered that the image recognition network create deep feature layers which yield surprising result as for predicting the human evaluation of image quality. This metric was done using a modified - updated to work on the modern Tensorflow implementation and improved to code responsible for loading the model to get read of reading the disc every time the function is evaluated - implementation of this metric, called `lpips` [21, 4].

It returns $p \in [0, 1]$, where 0 is a perfect perceptual similarity and 1 - complete dissimilarity. This property should make it a candidate for a loss function, however some other sources [1] seem to disagree. This was included as an experiment for a simple perceptual loss function.

## 4.4 Training

### 4.4.1 Data input

The input data was split 60% training, 20% validation and 10% test. The training data is fed to the network in the training phase and errors from the loss function are then back propagated through the network. The validation set is used to test the network after it has been trained and to see if the results it achieved for the training data also work for data that is not included in the back propagation stage. This is repeated every training epoch; every time the data in those sets is shuffled as well. For this particular configuration, each training epoch took over 22 minutes to complete, making the entire process a multi-hour endeavor.

### 4.4.2 Hyper parameters setup

The training process was run multiple times and analyzed in terms of practical training hyper parameters. The optimizer was set to Adam with a learning rate of 0.001 or 0.0005 depending on the model, constant throughout all iterations. Other parameters were left at default values. Most training was over 7 epochs for each model.

# Results

In this chapter, we present, compare and discuss results gotten from the paper's network, as well as introduce results from the related work.

## 5.1 Examples and comparisons

The comparisons were created by generating a high resolution predictions based on low resolution images and putting them side by side to the high resolution originals. This was done for hand picked set of images presented here for demonstrational reasons, being the best representation of different image contents.

### 5.1.1 Perceptual loss

The experimental quality of forementioned loss function turned out to be harder to tackle than anticipated. Cited difficulties and very long time of training the network using this metric (over an hour for each epoch) resulted in failure to deliver appropriate results. This is why it was omitted from the tables below.

### 5.1.2 Small detail

This particular photo, presented in figure 5.1, contains a lot of small detail which is blurred in the low resolution photo. It is regained in both MSE and SSIM models.

1. High resolution 2. Low resolution 3. MSE model 4. SSIM model

Figure 5.1: **Results for small details.** High resolution and two results are $2016 \times 2016$, low resolution is originally $992 \times 992$, all are $200 \times 200$ crops.

### 5.1.3 Edges and flat surfaces

This image was chosen for its smooth, flat surfaces and hard edges. It checks if the network does not introduce an unwated noise and can upscale egdes without blurring them. The results are positive both for MSE and SSIM models, show in figure 5.2.

### 5.1.4 Faces

Faces are crutial for the aforementioned use on social media pictures; upscaling them incorrectly makes viewers notice this instantly. The results from the paper are show in 5.3, all satisfactory. Both models brought back the wrinkles which were barely visible in the low resolution input.

1. High resolution 2. Low resolution

3. MSE model 4. SSIM model

Figure 5.2: **Results for edges and flat surfaces.** High resolution and two results are $2016 \times 2016$, low resolution is originally $992 \times 992$, all are $400 \times 400$ crops.

1. High resolution 2. Low resolution

3. MSE model 4. SSIM model

Figure 5.3: **Results for faces.** High resolution and two results are $2016 \times 2016$, low resolution is originally $992 \times 992$, all are $400 \times 400$ crops.

## 5.2 Metrics

The main metrics measured are MSEs and SSIMs of the naively interpolated images and the predicted ones when compared to the high resolution originals.

These were compared to the related works and results from their neural networks, including results for typical benchmark datasets, Set5 [6], Set14 [25] and BSD100 [18]. Those figure were taken from the papers cited in chapter 3. Results from this paper are in *cursive*, with different loss functions used in particular models as headers.

Additionally, a **Local** metric is shown, which is the result of the network being evaluated against the original 10% test dataset split.

| | **cubic** | **srcnn** | **srgan** | **vdsr** | **edrs** | **esrgan** | **nlsn** | ***mse*** | ***ssim*** |
|---|---|---|---|---|---|---|---|---|---|
| **Set5** | 33.66 | 38.30 | 29.40 | 37.53 | 32.46 | 38.11 | 38.34 | 29.00 | 28.32 |
| **Set14** | 30.24 | 29.00 | 26.02 | 33.03 | 28.80 | 33.92 | 34.08 | 28.23 | 27.60 |
| **BSD100** | 29.56 | - | 25.16 | 31.90 | 27.71 | 32.32 | 32.43 | 29.66 | 28.37 |
| **Local** | | | | | | | | 31.41 | 27.23 |

Table 5.1: **PSNR values**, higher = better.

| | **cubic** | **srcnn** | **srgan** | **vdsr** | **edsr** | **esrgan** | **nlsn** | ***mse*** | ***ssim*** |
|---|---|---|---|---|---|---|---|---|---|
| **Set5** | 0.930 | 0.863 | 0.847 | 0.959 | 0.897 | 0.960 | 0.962 | 0.936 | 0.938 |
| **Set14** | 0.869 | 0.786 | 0.740 | 0.912 | 0.788 | 0.920 | 0.923 | 0.909 | 0.914 |
| **BSD100** | 0.843 | 0.729 | 0.669 | 0.896 | 0.742 | 0.901 | 0.903 | 0.910 | 0.918 |
| **Local** | | | | | | | | 0.966 | 0.940 |

Table 5.2: **SSIM values**, higher = better. The SSIM values are, not surprisingly, better for a model trained

## 5.3 Human evaluation

Image quality is hardly ever correctly evaluated by algorithmic metrics like PSNR or SSIM. That is why a group of respondents with different backgrounds were asked to evaluate the image quality of the models' results for multiple datasets, similar how it was done on two other cited works.

The image quality survey was held online using Google Forms. It contained a single question asking about one's experience with photography, with two options available (no experience/some experience) and 25 questions with images upscaled using the network described in the paper. These were 5 photos from Set5, 5 photos from Set14, 7 photos from BSD100 and 8 photos from the test dataset from the paper. They were split evenly between MSE and SSIM models. There were two control images, one from the test dataset with bicubic compression, one with the original high resolution.

Respondents were asked to rate the photo based on the image quality itself, not the contents or how they liked it overall. They had to choose a number between 1 and 5 to do so, were 1 was the poorest quality and 5 was the best.

There were 56 answers to the survey, with 29 to 27 split in favor of "no photography experience" option. The highest rated image was, surprisingly, an SSIM model-upscaled image coming from the test set with the average of 4.55, not the high resolution high res, scoring at 4.28. This was also surpassed by a picture from Set14, with an average of 4.48. The average variance of the scores was 0.99, with the most "controversial" image having score variance of 1.86. The control bicubic photo scored 2.82 and was not the worst rated image in the survey, with an image from the Set5 reaching as low as 1.66, while also being the least controversial at a variance of 0.45.

All mentioned images are shown in figure A.1 and the results are shown in the table 5.3. The missing vdsr, edsr, edsr and nlsn columns are due to no such data being provided by the authors of these papers. Only this particular paper differences between different groups of respondents, results for cubic, srcnn and srgan are cited with no such distinction.

| | **cubic** | **srcnn** | **srgan** | ***mse*** | | ***ssim*** | |
|---|---|---|---|---|---|---|---|
| | | | | normal | photographers | normal | photographers |
| **Set5** | 1.97 | 2.57 | 3.58 | 2.08 | 2.00 | 2.83 | 2.65 |
| **Set14** | 1.20 | 2.34 | 3.72 | 2.38 | 2.33 | 3.63 | 3.73 |
| **BSD100** | 1.11 | 1.89 | 3.56 | 2.20 | 2.35 | 2.75 | 3.64 |
| **Local** | | | | 3.38 | 3.38 | 3.63 | 3.58 |

Table 5.3: **Mean opinion score values**, higher = better. First number is rating by people not acquainted with photography, the second is otherwise. The value marked with a star comes from a single control image and thus should not be considered a good comparison.

Figure 5.4: **Butterfly image.** It produced the lowest PSNR seen in tests.

## 5.4 Discussion of the results

There are several problems when comparing those results, including:

- The proposed network works with factor $\times 2$ upscaling only. Not all papers provided results for this factor or did not state the upscaling factor when presenting results.
- Other networks were trained with relatively small input size (200 - 500 pixels per image side) and large upscaling factor. The proposed solution works with big input sizes and splits into multiple tiles. This means that the datasets used for metrics are in-line with the training data of the other papers, but not with this paper's, as they are more zoomed into the detail.

This being said, some conclusions can be made:

- The PSNR values for the Set5 of the proposed models are noticeably below the other models from the comparison. After some investigation, a single image, show in figure 5.4, was found to score as low as 23.64. This is probably due to its nature being much different than anything seen in the training dataset. There are multiple black and white patches and a lot of yellow color, both absent from the prior.
- The mean opinion score for the Set5 is also much lower than the other models. The probable cause is the same as above.
- Despite being trained on a much smaller dataset than other solutions and taking much less resources to train, the network does well and delivers consistent performance.
- The SSIM and PSNR metrics are not directly related to the human perception of the image quality. This is shown by the mean opinion scores diverging from those two in multiple examples.
- The overall average rating from the respondents is 2.93 with a variance of 1.34. 57.7% of all votes cast for the pictures are 3 or more. This is a baseline to claim that most of the respondents found pictures produced by the models accetable.
- The results gathered from the professional photographers are almost always lower than the other group. This is not surprising, as the photographers are much more knowledgeable about the image quality factors.

# Summary

## 6.1 This paper

The goal of this paper was to introduce the main problems of the super resolution and propose a simple, resource-aware solution which achieves results comparable to the previously introduced solutions. This was done using an convolutional autoencoder, which is a well know, basic design, but proved to be easy to train and produced robust results. The survey conducted among real human subjects proved that the network delivers effects much better than the traditional alternatives available for the end user on the popular graphics editing software, regardless of the model (MSE, SSIM) used. The survey conducted among human subjects shown that the methods deliver acceptable performance in terms of human perception.

The results worth mentioning come from the SSIM model, both in terms of metrics and real human evaluation. The SSIM metric for all test sets for the mentioned model is not far off (usually about 1%) from the best other models, sometimes, like for BSD100, even surpassing them.

There are some shortcomings as well, like the unsuccessful use of the perceptual loss and therefore limited comparison options between the models. Other limitations, like largely varying performance across multiple metrics, come from the limited time and computational power spent on the training.

## 6.2 Possible future work

There are multiple challenges that can be derived from this paper and considered for a future improvements, including

- **Use working perceptual loss** - this failed as a research effort, probably due to an outdated implementation and no time to work out a better solution.
- **Improving the scale factor** - the proposed solution worked for $\times 2$ upscaling, creating four times as many pixels from the original image. Current state-of-the-art solutions usually focus on $\times 3$, $\times 4$ or even $\times 8$ scale factors, but also work with lower resolution inputs. Even though, improving the network's resolution performance should be achievable by extending its convolutional layers even further.
- **Creating a GAN** - generative adversarial networks have achieved by far the best test results and created a benchmark for modern super resolution work. They introduce some problems of their own (like high instability, mentioned before), but with enough time for training, they should be able to deliver better results.
- **Creating a spectral super resolution network** - the input to the network is 8-bit per color channel per pixel, but the original RAW photos are 14-bit per channel per pixel. This creates a future possibility to increase the spectral information of the input instead of the spatial one, which could be of use to the professional photographers, who rely on the RAW data for photo editing.

# Bibliography


[1] Experimenting with lpips metric as a loss function. https://medium.com/dive-into-ml-ai/experimenting-with-lpips-metric-as-a-loss-function-6948c615a60c. Accessed: 2022-06-16.

[2] Mixed precision. https://www.tensorflow.org/guide/mixed_precision. Accessed: 2022-06-16.

[3] Adobe. Photoshop lightroom classic. https://helpx.adobe.com/support/lightroom-classic.html.

[4] alexlee gk. lpips-tensorflow. https://github.com/alexlee-gk/lpips-tensorflow.

[5] M. I. Assaf Shocher, Nadav Cohen. "zero-shot" super-resolution using deep internal learning. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[6] M. Bevilacqua, A. Roumy, C. Guillemot, M. line Alberi Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. *Proceedings of the British Machine Vision Conference*, strony 135.1–135.10. BMVA Press, 2012.

[7] F. Chollet. *Deep Learning with Python*. Manning, List. 2017.

[8] C. Dong, C. C. Loy, K. He, X. Tang. Learning a deep convolutional network for image super-resolution. D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars, redaktorzy, *Computer Vision – ECCV 2014*, strony 184–199, Cham, 2014. Springer International Publishing.

[9] C. Dong, C. C. Loy, K. He, X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[10] S. Fadnavis. Image interpolation techniques in digital image processing: An overview. *International Journal Of Engineering Research and Application*, 4:2248–962270, 11 2014.

[11] W. Freeman, T. Jones, E. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.

[12] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[14] J. Kim, J. K. Lee, K. M. Lee. Accurate image super-resolution using very deep convolutional networks. *CoRR*, abs/1511.04587, 2015.

[15] N. Kodali, J. D. Abernethy, J. Hays, Z. Kira. How to train your DRAGAN. *CoRR*, abs/1705.07215, 2017.

[16] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[17] B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee. Enhanced deep residual networks for single image super-resolution. *CoRR*, abs/1707.02921, 2017.

[18] D. Martin, C. Fowlkes, D. Tal, J. Malik. 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*, wolumen 2, strony 416–423 vol.2, 2001.

[19] B. Mehlig. Artificial neural networks. *CoRR*, abs/1901.05639, 2019.

[20] Y. Mei, Y. Fan, Y. Zhou. Image super-resolution with non-local sparse attention. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, strony 3517–3526, June 2021.

[21] richzhang. Perceptualsimilarity. https://github.com/richzhang/PerceptualSimilarity.

[22] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[23] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, X. Tang. ESRGAN: enhanced super-resolution generative adversarial networks. *CoRR*, abs/1809.00219, 2018.

[24] W. Yang, X. Zhang, Y. Tian, W. Wang, J. Xue. Deep learning for single image super-resolution: A brief review. *CoRR*, abs/1808.03344, 2018.

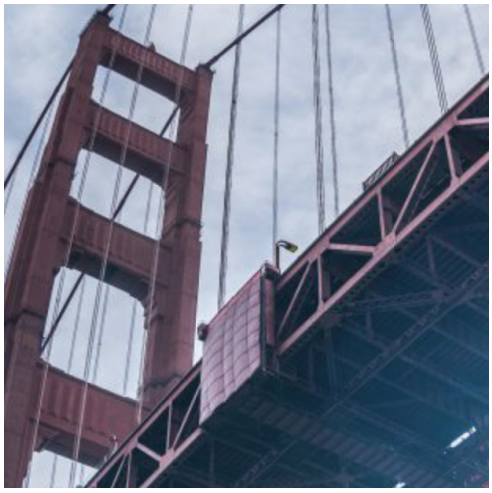
[25] R. Zeyde, M. Elad, M. Protter. On single image scale-up using sparse-representations. J.-D. Boissonnat, P. Chenin, A. Cohen, C. Gout, T. Lyche, M.-L. Mazure, L. Schumaker, redaktorzy, *Curves and Surfaces*, strony 711–730, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[26] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR*, 2018.

[27] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, Y. Fu. Image super-resolution using very deep residual channel attention networks, 2018.

# Image quality survey examples

1. Control high resolution image



2. Control low resolution image

3. Lowest rated image, SSIM model, Set5

4. Highest rated image, SSIM model, test set

5. Second highest rated image, SSIM model, Set14

6. Most controversial image, MSE model, BSD100

Figure A.1: **Images from the human evaluation survey.**

# CD contents

There are two files used to prepare the data for training and then do the training and gather metrics. Both are Project Jupyter files and require adequate runtime to work.