MASTER'S THESIS 2023

Enhancing Satellite Images Using Super-Resolution

Nils Olén

ISSN 1650-2884
LU-CS-EX: 2023-28

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2023-28

Enhancing Satellite Images Using Super-Resolution

Förbättring av satellitbilder med hjälp av super-resolution

Nils Olén

Enhancing Satellite Images Using Super-Resolution

Nils Olén
ni3552ol-s@student.lu.se

June 27, 2023

Master's thesis work carried out at Tactel AB.

Supervisors: Michael Doggett, michael.doggett@cs.lth.se
Jonas Bondesson, jonas.bondesson@tactel.se
Tobias Leksell, tobias.leksell@tactel.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Super-resolution refers to the concept of enhancing the resolution of an image. This project investigates the use of machine learning based super-resolution to upscale satellite images for use in an interactive map application that is displayed in airplanes. The purpose is to store lower quality satellite images and upscale them instead of storing the high-resolution images, to ultimately save storage space. The research includes evaluating different super-resolution implementations, as well as a conventional bicubic image interpolation algorithm, to determine which method performs best in terms of image quality (PSNR and SSIM), upscaling speed and storage space saved. It was concluded that, out of the methods tested, EDSR produces the images that are closest in similarity to the original images. Findings also include that upscaling speed is a problem, because some of the deeper network architectures result in a good quality image, but with an upscaling speed that is too slow.

Keywords: Super-resolution, Neural network, Machine learning, Deep learning, Image interpolation, Satellite imagery

Acknowledgements

I would like to thank my supervisor Michael Doggett for the helpful discussions regarding the content of this thesis. I would also like to thank my supervisors at Tactel, Jonas Bondesson and Tobias Leksell, for helping me setup and giving me the resources for carrying out the research needed for the project. Finally, I would like to thank Tactel for giving me a comfortable and friendly space to work in, as well as providing me with the equipment that I required.

Contents

Chapter 1
Introduction

These days, machine learning is becoming increasingly common for solving computer related tasks. As with many other problems, this also applies to image upscaling, where the use of convolutional neural networks has popularised the concept known as super-resolution.

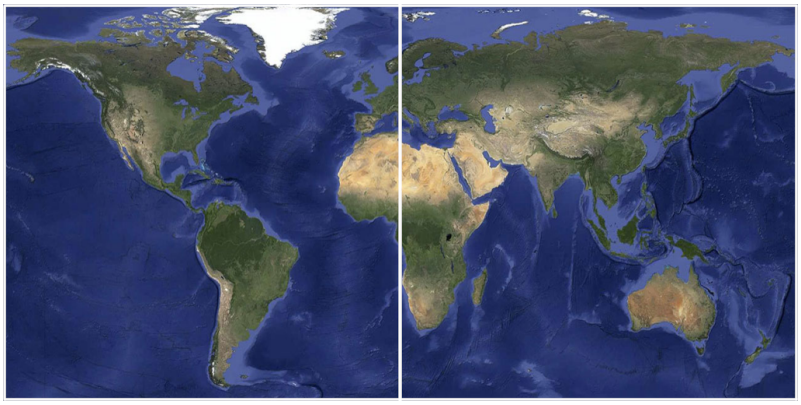
Single image super-resolution, often simplified as super-resolution, refers to the concept of reconstructing a high-resolution image using a low-resolution image as input in a trained neural network. The network is trained using many pairs of low- and high-resolution versions of the same image. The training process teaches the network how to upscale these images, and when the training process is finished, provided that the training image dataset was varied enough, the trained network can be used to upscale other previously unseen images.

When dealing with large amounts of data, the issue of storage space is always present. Super-resolution can be used to decrease the size of needed storage space by removing the need of storing high-resolution images, by upscaling lower-resolution images when needed instead.

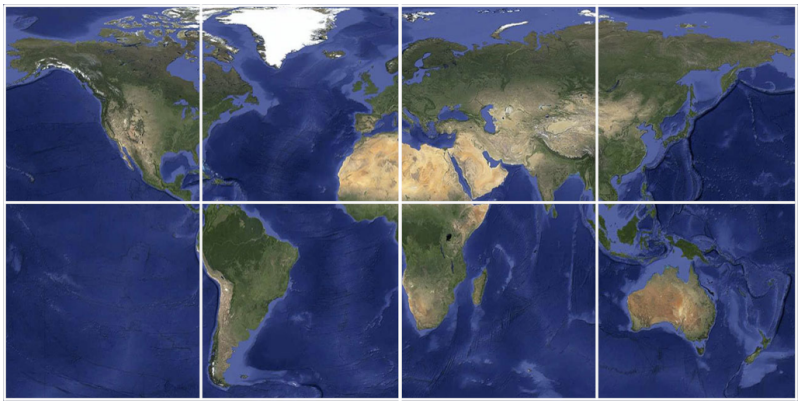
Arc [1] is an interactive map service, developed by Malmö-based company Tactel, that is designed to be displayed to passengers in airplanes. The map is built out of satellite images, which are arranged in layers called zoom levels. When zooming in on the map, new details emerge because higher detailed satellite images are being utilised the closer a user zooms in.

The Arc system follows a tiled map model [2]. At its most zoomed out level, zoom level 1, it originally consists of two 512x512 images which display the whole world map. When increasing the zoom level to level 2, each of these images are replaced by four new 512x512 images, which occupy the same space as the image they replace, resulting in a more detailed map which is displayed using eight 512x512 images. This pattern is followed all the way to the last level, zoom level 13, which contains over 20 million images. An example of the tiled map model is shown in Figure 1.1.

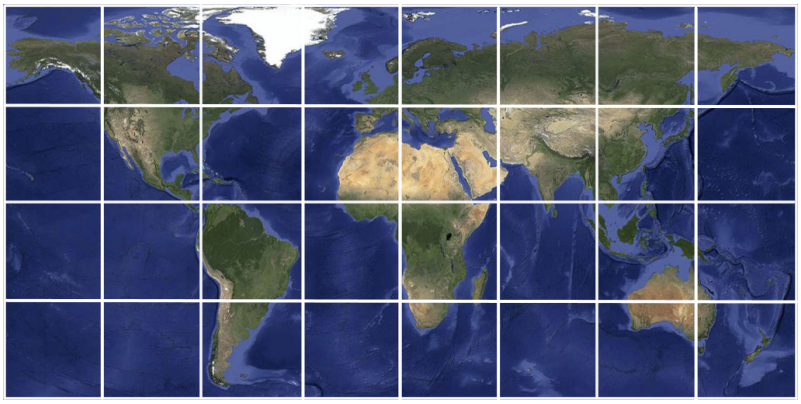
The images on zoom level 13 use about 241 GB of storage space, which is almost 4 times as many as zoom level 12, which uses about 68 GB of storage space. Using super-resolution to upscale the images from zoom level 12 to zoom level 13, instead of storing the actual high-resolution images, could free up a significant amount of space on the airplane's storage space,



(a) Zoom level 1.



(b) Zoom level 2.



(c) Zoom level 3.

Figure 1.1: An example of how the tiled map model works for the first three zoom levels. Each square has a size of 512x512 pixels.

which could then be used to store other desired content. To upscale the images from zoom level 12 to zoom level 13, each image on level 12 is split into four images of size 256x256 pixels, after which each of these four images is upscaled to a size of 512x512 pixels to generate the tiles used to build zoom level 13.

As with any other computer related task, super-resolution comes with a number of challenges. The upscaled images are estimations of the high-resolution images. This estimation

needs to be close enough to be a viable option to replace the high-resolution images. There is also the problem of upscaling speed. In a map service like Arc, the images need to be upscaled at a speed fast enough to not cause a too distracting delay. The trained network also uses up storage space. The space used by the network can not be so large that the decrease in storage space use is no longer useful.

1.1 Aim

The aim of this project is to identify and study different methods of implementing super-resolution to find out how well they are suited for usage on satellite images in an interactive map system. This will be done by evaluating a test set of images which are upscaled from zoom level 12 to zoom level 13 using the different super-resolution methods. To explore the possibility of using super-resolution as an alternative to storing high-resolution satellite images, the different methods will be evaluated on image quality of the produced images, upscaling speed and storage space saved.

1.2 Research Questions

- Out of a selection of super-resolution models, which method is best suited for upscaling satellite images in regards to resulting image quality, space saved and upscaling speed?
- Could it be used as a potential replacement for the real high-resolution images in a map application?

Chapter 2
Background

To put the research in this project into context, this chapter presents the foundation on which it is built. The chapter serves as an introduction to the theory behind super-resolution, and image upscaling in general, as well as a short history of super-resolution research, both in the broad sense and specifically for satellite imagery.

2.1 Theory

To understand how super-resolution works, its fundamental building blocks need to be explained. This section outlines the theory behind central concepts such as conventional interpolation methods, how neural networks work and how super-resolution uses neural networks to function. This section also includes how various performance factors can be measured and how these metrics are calculated.

2.1.1 Digital Image Interpolation

When increasing the spatial resolution of an image, also known as upscaling, the information in that image is stretched out over an area containing a larger amount of pixels than before, so a decision has to be made concerning what to do with the new pixels that appear between the old pixels, i.e. which colour to display. The process of estimating the missing pixels is called interpolation. A visual example of this shown in Figure 2.1 and Figure 2.2.

Image interpolation is a technique that is widely used in image processing, and works as a component of many image processing algorithms, including super-resolution, where the interpolation is performed using a machine learning approach. Some of the most common digital image interpolation techniques include nearest-neighbour, bilinear, bicubic and Lanczos interpolation.

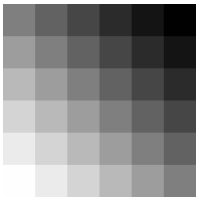


Figure 2.1: A 6x6 image that is about to be resized.

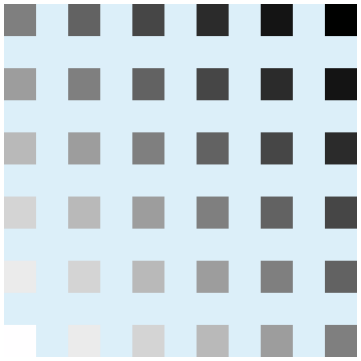


Figure 2.2: The 6x6 image resized to 11x11, the new pixels that appear between the old pixels need to be interpolated.

Nearest-Neighbour Interpolation

With nearest-neighbour interpolation we simply look at the pixel that is closest to the undecided pixel and display the same colour as that pixel. The effect of this is that in the enlarged version of the image, each pixel from the original image becomes bigger. So for the example in Figure 2.2, the new pixels would take on the colours as shown in Figure 2.3.

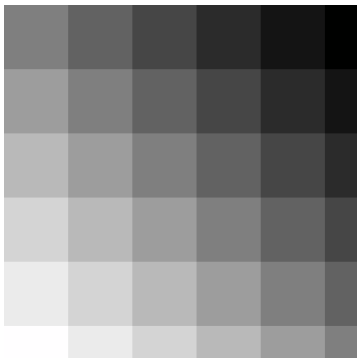


Figure 2.3: The resized 11x11 image with nearest-neighbour interpolation applied.

Bilinear/Bicubic Interpolation

Bilinear, based on repeated linear interpolation, looks at an area of 2x2 pixels to calculate the weighted average for every new pixel. Linear interpolation looks at the value in between two known values and assigns that value to the new pixel. Bilinear means that we do linear interpolation on each axis, e.g. the x-axis and then on the y-axis. Bilinear interpolation is

fast and produces reasonably good results. The results of using bilinear interpolation on the example image is shown in Figure 2.4

Bicubic interpolation, which is commonly used in photo editing software [3, p. 120], looks at an area of 4x4 pixels and computes the weighted average to decide the new colour display value for each new pixel. Since it requires more calculating, bicubic is generally slower than bilinear, and it uses more memory. Figure 2.5 shows an example that illustrates the differences between bilinear and bicubic interpolation.

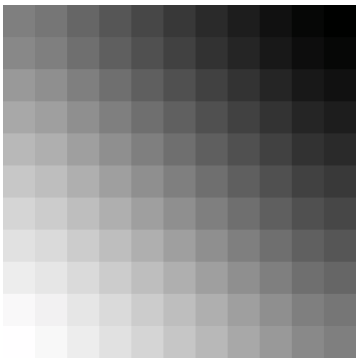


Figure 2.4: The example image resized to 11x11 using bilinear interpolation.

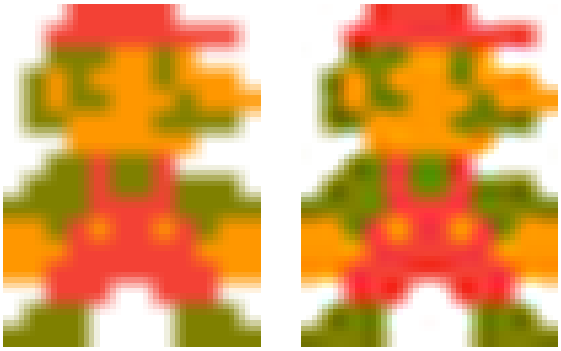


Figure 2.5: 8-bit Mario upscaled x4 using bilinear and bicubic interpolation.

Lanczos Interpolation

Lanczos interpolation is a more sophisticated method, which also means that it is the slowest and uses the most memory out of the mentioned methods. An example of using Lanczos interpolation is shown in Figure 2.6.

2.1.2 Neural Networks

An artificial neural network (ANN), often simplified as neural network, is a machine learning technique that is based on the structure and function of biological neurons in an organism's brain [4, p. 1]. Neural networks are built out of an input layer, a number of hidden layers,

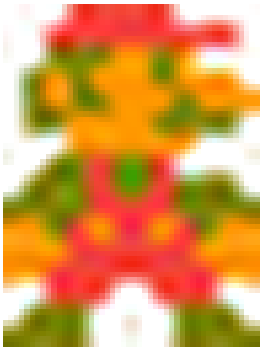


Figure 2.6: Mario again, upscaled using Lanczos interpolation.

and an output layer. Each layer contains nodes (neurons) where data is passed through an activation function, which decides if and what that node will output to the nodes in the next layer. The connections between nodes have weights, so when a value goes toward the input to a node it is scaled by a weight before being summed with the other inputs to that node. These weights start off as random numbers and are then estimated in the training process to map the inputs to the outputs.

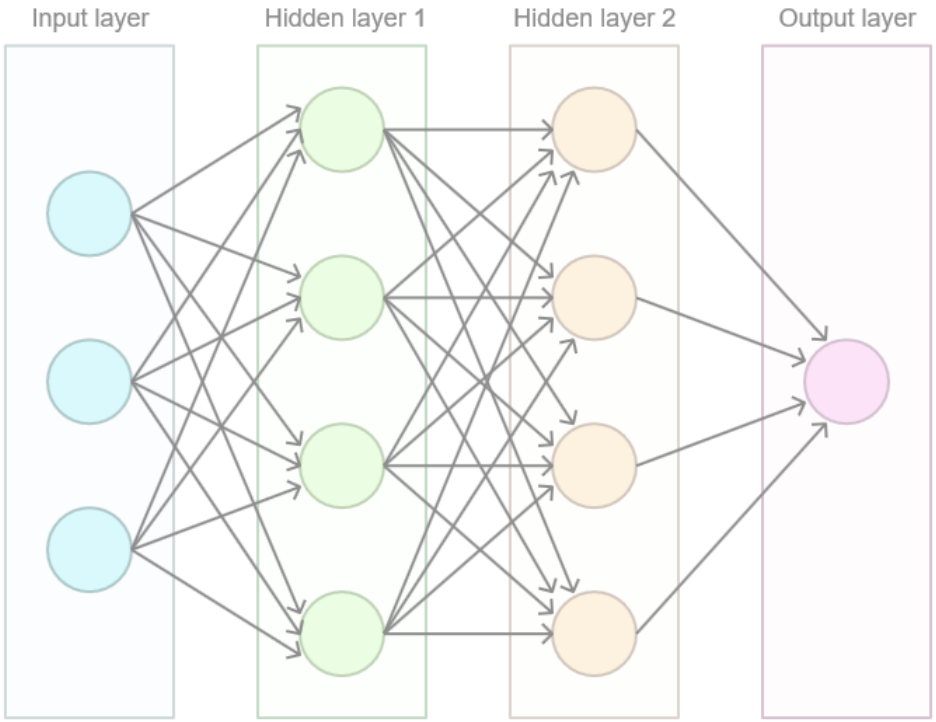


Figure 2.7: An example of a simple neural network.

In the example image in Figure 2.7 we see a simple neural network with one input layer, two hidden layers and one output layer. It also shows the nodes in the network: three input nodes, four in each of the hidden layers and one output node.

Deep learning refers to there being multiple hidden layers. The more layers that are added to a network, the deeper the network is. A network can have any number of hidden layers depending on how deep you want the network to be. There could be hundreds or even

thousands of hidden layers if that is desired.

The training process for a neural network generally works like this: if $y = F(x)$, where x is the input and y is the output of the network, we want to estimate F, which represents the network itself. What we have is many examples of input-output pairs (x, y), which form a training dataset and a validation dataset. With machine learning we estimate F "automatically", by presenting these input-output pairs and letting the network adapt (changing the weights) itself to become an estimation of F. It does this by estimating the output from each input in the training dataset, which is compared to the true output using a loss function. It "corrects" itself using backpropagation and estimates again, and repeats this process for a specified number of iterations.

Activation Function

The activation function, as mentioned earlier, decides what a node will send to the next layer. If a value of zero is sent, this node is not activated, hence the name, and does not influence the computations in the next layer. One of the most commonly used activation functions is the rectified linear unit (ReLU):

$$f(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where x is the input to the node and $f(x)$ is the output from the node.

There is also Leaky ReLU. This variant of ReLU lets the node be active even when x is negative, by letting through a small value, which is useful when a network suffers from too few nodes being activated.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases} \tag{2.2}$$

Leaky ReLU has an expansion called Parametric ReLU. This activation function makes the leakage coefficient a into a trainable parameter:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ a \cdot x & \text{otherwise} \end{cases} \tag{2.3}$$

Loss Function and Backpropagation

The loss function is used to measure how close the network's estimated values are to the true values. The goal is to minimise the loss function. For the loss function, one of the most commonly used is mean squared error (MSE):

$$MSE = \frac{1}{n} \sum_{i=0}^{n} (Y_i - \hat{Y}_i)^2. \tag{2.4}$$

where n is the number of samples, Y is the true output of sample i and $\hat{Y}$ is the estimated output of sample i.

After calculating the loss function, backpropagation is used to compensate for the loss by adjusting the weights in the network. It calculates the gradient of the loss function with

respect to the weights in the network. The weight updates are then (commonly) done using an optimisation algorithm called stochastic gradient descent.

Hyperparameters

To control how the neural network learns, there are a number of configuration settings that can be used to manipulate the learning process called hyperparameters. The number of epochs decides the number of training iterations, i.e. the number of times the network gets to evaluate and correct itself. The learning rate determines how fast the network learns by adjusting how large the correction steps are in each epoch. Batch size decides how many training samples to go through before calculating the loss and updating the weights. If the batch size is smaller than the size of the training dataset, the weights are updated multiple times per epoch, and the optimisation method is called batch gradient descent, instead of stochastic gradient descent.

2.1.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of neural network that are commonly used in computer vision tasks [4, p. 40]. In CNNs, the layers are 3-dimensional, where the shape often corresponds to image width, image height and number of image channels. They work similarly to regular neural networks, but they include an important component between the regular fully connected layers called a convolutional layer.

The convolutional layers work as filters that perform a convolutional operation on images between layers in the network. They are 3-dimensional filters of weights with the same depth as the previous layer but with a smaller size when it comes to width and height. The weights in these filters are estimated during the training process to create a convolutional operation that produces the desired outputs.

Figure 2.8 shows a simple CNN. The smaller square which connect the layers represent the convolutional operations. The small squares are the convolution filters, which slide over the matrices, calculating feature maps which creates the next layer. The stride of a convolution filter refers to how big steps it takes when it slides over a matrix. So a larger stride gives smaller resulting feature maps. The "dense" part of the network is a fully connected neural network where the previous 3-dimensional layer has been flattened into a layer of nodes. This particular network has 128 output nodes.

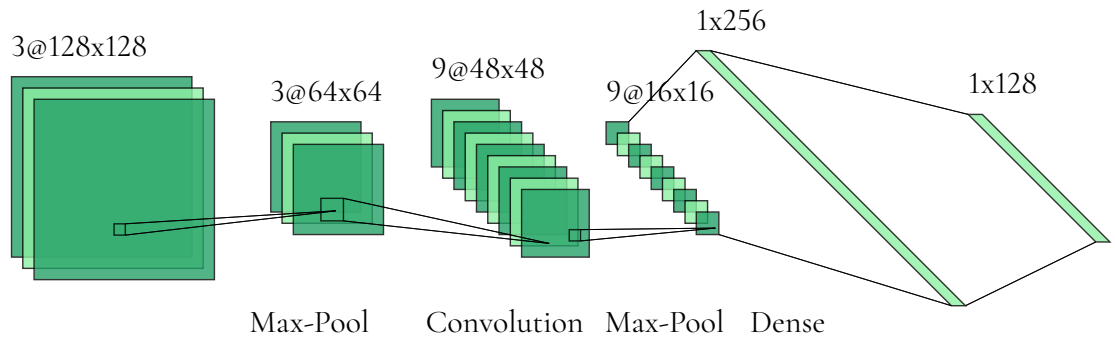


Figure 2.8: A simple convolutional neural network.

Max-pooling refers to the operation of calculating the max value of patches in a feature map, and using the values to create a new feature map. It results in a downsampled, smaller feature map.

For CNNs there are publicly available pre-trained networks which are commonly used instead of training a network from scratch [4, p. 42].

2.1.4 Generative Adversarial Networks

A generative adversarial network (GAN) is a machine learning framework that uses two neural networks: a generator network and a discriminator network [4, p. 45].

The generator network takes random Gaussian noise as input data and generates new data that is intended to resemble a sample from the real dataset. The discriminator network takes in both the real data and the generated data from the generator network and tries to distinguish between them.

During training, the two networks are trained in an adversarial manner: the generator network tries to generate data that can fool the discriminator network into classifying it as real, while the discriminator network attempts to correctly classify the real data and the generated data into their respective classification categories.

As the two networks are trained together, the generator network learns to generate more realistic data, while the discriminator network becomes better at distinguishing between the real data and the generated data. Eventually, the generator network learns to generate data that is indistinguishable from the real data, at which point the training process can be stopped. The GAN training process is visualised in Figure 2.9.

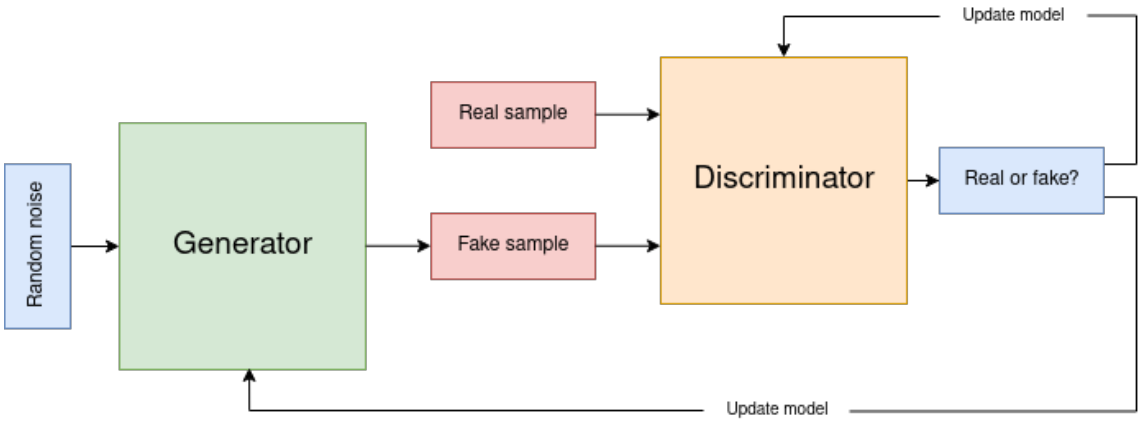


Figure 2.9: A diagram of how a GAN works.

2.1.5 Super-Resolution using Neural Networks

If we use neural networks to apply digital image interpolation we get super-resolution. In super-resolution, the input to the neural network is a low-resolution image and the output from the network is the same image in a higher resolution. It is the neural network's job to estimate a mapping between these input-output pairs to be able to upscale new, previously

unseen images to a higher resolution. Super-resolution methods can be broadly categorised into two types: single-image super-resolution and multi-image super-resolution.

Single-image super-resolution, which is the type of SR that has been referred to so far in this text, aims to reconstruct a high-resolution image from a single low-resolution input image. Since little input information is available, single-image super-resolution has a tendency to display upscaling artefacts in the resulting images.

Multi-image super-resolution utilises multiple low-resolution images of the same scene to generate a high-resolution output. By exploiting the redundancy across multiple images, these algorithms are able to resolve more details. Since the satellite images provided only contain one image per scene, single-image super-resolution is the focus in this project.

2.1.6 Metrics

There are many metrics that can be used to measure similarity between images. Out of these, the most commonly used metrics for super-resolution are peak signal-to-noise ratio and the structural similarity index measure.

Peak Signal-to-Noise Ratio

Peak signal-to-noise ratio (PSNR) is a value that is calculated as the ratio of the maximum possible pixel value to the mean squared error between the original and processed images. With a ground truth image I and a noisy representation of that image K, the MSE becomes:

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2. \tag{2.5}$$

In short, this calculation means summing the squared difference of each pixel and dividing by the image size.

The PSNR is:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right), \tag{2.6}$$

where MAX_I is the maximum pixel value, i.e. 255.

When using colour images, the MSE is calculated by adding the three colour channels' squared difference and then dividing by 3 multiplied by the image size.

PSNR is expressed in decibels (dB), and higher PSNR values indicate closer similarity between the images. Typical values for a compressed image fall between 30 and 50 dB. A value over 40 dB is considered very good and a value below 20 dB is generally considered poor [5, p. 135].

While PSNR is a widely used metric for image similarity, it has some limitations. For example, it does not take into account perceptual differences between images, such as differences in texture. Additionally, it can be sensitive to compression artefacts and may not always correspond well with subjective human evaluations of image quality. Despite this, PSNR remains a useful tool for evaluating image processing techniques.

Structural Similarity Index

The structural similarity index measure (SSIM) is a measure of image quality that takes into account both the structural information and the pixel values of an image or video. It is designed to be more perceptually accurate than metrics like PSNR, which only measure the difference between pixel values.

The SSIM is calculated by comparing the structural and pixel information of two images. It takes into account three measures of similarity: luminance, contrast, and structure.

With x and y being the images being compared, the SSIM is calculated using the following formula:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.7}$$

where

- μ_x is the pixel sample mean of x,
- μ_y is the pixel sample mean of y,
- σ_x^2 is the variance of x,
- σ_y^2 is the variance of y,
- σ_{xy} is the covariance of x and y,
- $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator,
- $L = 2^{b-1}$ is the dynamic range of the pixel values, where b is the number of bits per pixel,
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

The SSIM ranges from -1 to 1, with values closer to 1 indicating closer similarity between the two images.

SSIM has several advantages over PSNR. It takes into account perceptual differences between images, such as differences in texture, colour and contrast. Additionally, it is more sensitive to compression artefacts, making it more accurate for evaluating the quality of compressed images.

Other Metrics

With image quality accounted for, more metrics are needed to measure other aspects of the performance. To measure the space saved by using a particular super-resolution model, we simply take the amount of storage space the images from zoom level 12 use and add the size of the trained network. To measure the upscaling speed of an SR model, a built-in Python function is used to measure how long it takes to upscale each image in a test set.

2.1.7 Downsampling

The low-resolution image equivalents, will have to have been downsampled using some method. The method used in the Arc system is known as pixel binning. In this method, blocks of 2x2 pixels are averaged to produce one pixel in the low-resolution output image. However, many super-resolution models are optimised for using bicubic downsampling.

2.2 Previous Research

This section contains research previously done regarding the task of super-resolution. The history of using neural networks to perform super-resolution is outlined, and some previous examples of using super-resolution for enhancing satellite imagery are described.

2.2.1 A History of Super-Resolution

The following is a brief history of super-resolution using neural networks. It is structured into different model categories which are presented separately, and contains the models on which this work is based on. Implementations of some of the described models will be used to evaluate the research questions.

Linear Networks

The first use of deep learning in super-resolution dates back to 2014 and is presented in the paper "Image Super-Resolution Using Deep Convolutional Networks" by Dong et al. [6]. It introduces a novel approach in performing single image super-resolution using a deep convolutional neural network called SRCNN (Super-Resolution Convolutional Neural Network).

This CNN architecture consists of three layers. The first is a patch extracting layer which extracts dense patches from the low resolution input image and represents them as high-dimensional vectors which make up a set of feature maps. The second layer is a non-linear mapping layer which consists of 1x1 convolutional filters which map the high-dimensional vectors to another set of high-dimensional vectors that ultimately comprise a new set of feature maps. These vectors represent high-resolution patches. The final layer aggregates the representations of the high-resolution patches to reconstruct the high-resolution image.

The SR method Very Deep Super-Resolution (VDSR) [7], propsed in 2015, is an improvement on SRCNN and uses (as the name suggests) a very deep convolutional network which is based on the VGG-net architecture [8]. Instead of learning the direct mapping between the low and the high resolution images, it only learns a residual mapping that is the difference between the LR and the HR image. It also uses a learning rate that is 10^4 times greater than SRCNN, to speed up training. This method performs better than previously existing methods, and supports the theory that a deeper network architecture can provide a better contextualisation [9].

While SRCNN and VDSR are examples of early upsampling designs, i.e. they upsample the LR image to the size of the HR image before performing the SR operation, the following SR methods employ late upsampling. A late upsampling design performs feature extraction on the LR image and upsamples at the end, which significantly reduces the required computational cost [9].

Fast Super-Resolution Convolutional Neural Network (FSRCNN) [10] is an upgrade of SRCNN which was developed in 2016 to combat the high computational cost that hinders SRCNN from being used practically, essentially by making it faster. The redesign includes, among other changes, the introduction of a deconvolution layer at the end of the network, which is responsible for upsampling the image. This means that the mapping is learned directly from the LR image to the HR image. This model reaches speeds of more than 40 times faster than SRCNN, while also achieving a better image reconstruction.

The next model, Efficient Sub-Pixel Convolutional Neural Network (ESPCN) [11], also from 2016, introduces a sub-pixel convolutional layer which performs an operation at the end of the network that is similar to the deconvolution layer in FSRCNN. The sub-pixel convolution works by taking the depth of the last layer and converting it into the spatial dimension, essentially aggregating the feature maps from the low-resolution space and buiding the upscaled image in a single step. ESPCN is the first network model capable of real-time super-resolution on 1080p videos on a single GPU, while also producing higher quality images than previous CNN-based models [11].

Residual Networks

Residual learning is a technique that differs from linear networks by incorporating skip connections in the network design. This approach is used to mitigate the issue of vanishing gradients, which in turn makes working with very deep networks feasible [9]. In a residual network for super-resolution, the network learns the difference, or "residue", which mainly consists of high frequency information.

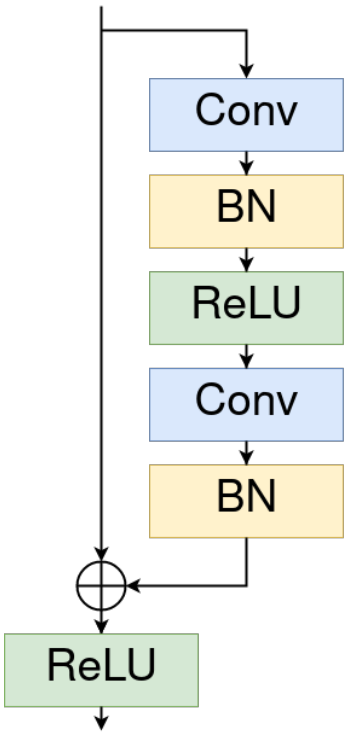


Figure 2.10: A residual block from the ResNet architecture. BN stands for Batch Normalisation.

The Enhanced Deep Super-Resolution (EDSR) [12] network, proposed in 2017, builds upon the ResNet architecture [13], which consists of multiple residual blocks like the one shown in Figure 2.10. EDSR demonstrates improvements in accuracy by excluding batch normalisation layers, which normalise the input and restricts the network's range. Removal of these layers can also yield a reduction in memory usage of up to 40%, which makes training more efficient [14]. This paper also introduces a Multi-Scale Deep Super-Resolution (MDSR) [12] system which works similarly and can upscale images using different upscaling factors.

Cascading Residual Network (CARN) [15], from 2018, is also a modification of the ResNet architecture. While being similar to EDSR, it also incorporates a cascading mechanism that feeds information from all previous layers as input to any convolutional layer. The cascading also works on a local level where each block between the convolutional layers incorporates a similar cascading mechanism internally. CARN produces results that are comparable to state-of-art-methods at the time of its conception.

EDSR and CARN use a single-stage design, which means that they are comprised of a single network. Several architectures employ a multi-stage design to enhance performance by extracting features separately in the low-resolution space and high-resolution space. In this approach, the first stage predicts the coarse features, which are further refined in the later stage [14].

An example of a multi-stage approach for residual networks is Balanced Two-Stage Residual Network (BTSRN) [16]. It is comprised of two parts: a low-resolution stage, which has 6 residual blocks, and a high-resolution stage, which includes 4 residual blocks. Given the larger input size in the HR stage, the convolution process has a higher computational cost

than in the LR stage. To achieve a balance between accuracy and performance, the number of blocks in each stage has been chosen accordingly. The model ranked among the best at NTIRE SR 2017 in terms of both accuracy and speed [16].

Recursive Networks

Recursive networks utilise recursively connected convolutional layers or recursively linked units. This approach aims to progressively break down the complex super-resolution problem into a series of simpler ones, which are easier to tackle [9].

An example of this is the Deep Recursive Convolutional Network (DRCN) [17]. Developed as early as 2015, it applies the same convolutional layers repeatedly, which helps to maintain a constant number of parameters for multiple recursions. DRCN is composed of three smaller networks: the embedding net, inference net, and reconstruction net. First, the embedding network, converts the input into feature maps. Secondly, the inference net performs the super-resolution task by using a single recursive layer to analyse a large image region, and applying the same convolutional layer multiple times while increasing the size of the receptive field after each recursion. Lastly, the reconstruction net transforms the feature maps that were output by the inference net into the final image. According to the authors, this method outperforms previous methods by a large margin [17].

Deep Recursive Residual Network (DRRN) [18] combines the recursive and residual methods by improving DRCN with the introduction of residual layers. This results in better overall performance compared to DRCN.

Progressive Reconstruction Networks

CNN models usually predict the output in one step, which may not be feasible when it comes to upscaling with large scaling factors. To address this issue, certain methods predict the output in multiple stages. For example, if we want to upscale by 4x, we separate the operation into steps, and first do 2x followed by 4x [9].

Laplacian Pyramid Super-Resolution Network (LapSRN) [19], proposed in 2017, utilises this method of upscaling in a pyramidial framework. It consists of three sub-networks that progressively generate residual images up to a factor of 8x. These residual images are added to the input low-resolution image to obtain super-resolution images. The first sub-network generates a residue of 2x, the second predicts the 4x and the final sub-network provides the 8x residual image. The resulting residual images are then added to the respective scaled upsampled images to obtain the final image [9]. The model performs favourably against state-of-the-art methods at the time of the paper's publication [19].

Densely Connected Networks

Super-resolution algorithms that fall under the category of densely connected networks are inspired by the success of the DenseNet architecture for image classification [20]. The idea behind this approach is to incorporate hierarchical cues available at different depths of the network that are combined to improve flexibility and achieve richer feature representations [9].

One of these designs is the 2017 model SRDenseNet [21], which uses skip connections in a very deep network. The dense skip connections help to combine low-level and high-level

features, which leads to better reconstruction performance. Additionally, the connections enable direct short paths from the output to each layer, which counteracts the vanishing-gradient issue that is common in very deep networks. To further enhance efficiency, the network includes deconvolution layers for learning upsampling filters and speeding up the reconstruction process. The method, according to the authors, sets a new state of the art for super-resolution, at the time of publication [21].

Multi-Branch Networks

Training very deep networks is difficult partly due to the problem of information flow. Multi-branch networks address this issue by dividing the information flow into a number of branches in which information can be allowed to pass [14]. This results in a fusion of information that stem from different receptive fields, allowing for better high-resolution reconstructions [9].

Cascaded Multi-Scaled Cross Network (CMSC) [22], published in 2018, is a multi-branch network which is built out of a feature extraction layer, a sequence of cascading subnetworks and a reconstruction layer. The feature extraction layer and the reconstruction layer works similarly to earlier examples. The mentioned subnetworks, which is the new introduced element, consist of two branches, each of which utilise different filter sizes which result in different receptive fields that capture contextual information at different scales. The result is a fusion of parts of information that complement each other, leading to a higher quality image reconstruction.

Another example of a multi-branch network, also from 2018, is the Information Distillation Network (IDN) [23]. With a similar architecture to CMSC, it consists of a feature extraction block, multiple stacked information distillation blocks and a reconstruction block. The distillation block is made up of an enhancement unit, which consists of 6 convolutional layers, and a compression unit. Halfway through the enhancement unit, the multi-branching happens. The output is sliced, where one half passes on through the enhancement unit and the other is returned and concatenated with the input of the block. This concatenation is then added to the output of the enhancement unit. Lastly, the compression unit uses a 1x1 convolutional filter to reduce the number of channels in the output of the enhancement unit. The purpose of this new block is the same as in CMSC, to identify features at different scales. This model performs particularly better in terms of speed [23].

Attention-Based Networks

Previously mentioned super-resolution models treat all spatial locations and channels as being of equal importance. However, in some cases, it may be beneficial to focus only on a select few features at a given layer. This is where attention-based models can be useful as they allow for flexibility by considering that not all features are crucial for super-resolution and instead have different levels of significance [9].

SelNet [24], proposed in 2017, introduces a selection unit that is put at the end of convolutional layers that can decide which information to pass on. In total, there are 22 convolutional layers with each one being connected to a selection unit. Additionally, it also uses residual learning and includes a sub-pixel layer at the end, similar to ESPCN.

Residual Channel Attention Network (RCAN) [25], from 2018, is also an attention-based

network. It employs a recursive residual-in-residual design, and consists of several residual groups with long skip connections, inside of which a couple of residual blocks sit with short skip connections. This structure allows the individual blocks to focus on low-frequency details, while the main network identifies high-frequency information. Each of the local residual blocks include an attention channel mechanism, which selectively passes on information in a way that is comparable to SelNet.

Generative Models

Generative models use a design based on generative adversarial networks (GANs). This means that, unlike previously mentioned models, they are not based on a pixel-wise loss function. Humans do not see differences between images pixel by pixel, but in perceptual quality [14], which is what the following models generally try to replicate.

One of these models is SRGAN [26], which was published in 2016. It introduces a perceptual loss function which consists of three parts: MSE for capturing pixel-wise loss, perceptual similarity loss which looks for differences in a high-level space and adversarial loss from the discriminator network, which is trained to differentiate between SR images and the original images.

EnhanceNet [27], also from 2016, uses an architecture based on the Fully Convolutional Network [28] which uses residual learning. The model, like with SRGAN, introduces a perceptual loss function which is supposed to enhance finer details and counteract the over-smoothed images that can result from using only pixel-based loss.

ESRGAN, which stands for Enhanced SRGAN [29], is an extension of SRGAN, and was made in 2018 to combat visual artefacts that appear in upscaled images. The main differences include: a residual-in-residual dense block without batch normalisation as the main building block for the network, a relative realness output from the discriminator network instead of a binary real-or-fake decision, and improvements on the perceptual loss function.

Recent Methods

The models in this section are recent super-resolution methods that introduce new elements into their networks, which are briefly presented.

Content Adaptive Resampler (CAR) [30], proposed in 2020, addresses the issue of downsampling by using a network called ResamplerNet to learn a task specific image downscaling method that is adapted to the super-resolution operation. The SR network used in CAR is based on the EDSR network.

Super-resolution models generally use one loss function to optimise the network, which can result in visual artefacts. Super-Resolution using Optimal Objective Estimation (SROOE) [31], a model published in 2022, incorporates multiple loss functions that are adaptively applied to different regions where they are deemed optimal.

Hybrid Attention Transformer (HAT) [32] is a a Transformer [33] based super-resolution method developed in 2022. It is designed to activate more pixels from the input image.

2.2.2 Super-Resolution on Satellite Imagery

Using super-resolution in combination with satellite imagery is not a new concept. The challenge of enhancing images of the earth's surface has been done before, not only for the sake of getting nice looking maps, but for different purposes as well, such as object detection and improving image segmentation.

The article "Enhancing Satellite Imagery with Deep Learning: A Practical Guide" [34] deals with the general problem of getting clearer images for a map application. It proposes a neural network architecture based on the Fully Convolutional Network [28] to solve this problem, finding regular interpolation methods to not produce good enough results.

One example [35] deals with the problem of semantic segmentation in satellite images. The challenge is to locate crop fields and identify which crop type it contains, which requires high-resolution clear images capable of showing small details. In the article, it was found that only using the images was not good enough to get an accurate labelling of the crop fields, so a solution incorporating super-resolution was tested. The results are clearer images which have the capability to significantly help the segmentation process and crop field labelling.

Another example [36] of using super-resolution for satellite imagery implements GAN-based approach based on the SRGAN architecture. This article uses large scale images to evaluate the approach, i.e. images that cover a large space, and the model shows significant improvement over a bicubic interpolation algorithm for these images.

The article "Super-Resolution of Multispectral Satellite Images Using Convolutional Neural Networks" [37] takes advantage of pan-sharpening, i.e. the process of merging high-resolution panchromatic imagery with low-resolution multispectral imagery to create a single high-resolution colour image, that is often applied to satellite images to investigate a novel approach of satellite image super-resolution. The article evaluates multiple super-resolution methods while incorporating pan-sharpening and finds RedNet30 to give the overall best performance.

Multi-image super-resolution (MISR) has also been used in conjunction with satellite images by utilising multiple images of the same area to find hidden fine details in low-resolution images that can be resolved in a super-resolution image that is generated by a fusion of multiple input images [38]. MISR is also useful for making clearer satellite images by, for example, removing clouds that partly obscure the images in different areas.

In an article entitled "The Missing Ingredient in Deep Multi-Temporal Satellite Image Super-Resolution" [39] a multi-image super-resolution solution is proposed in the form of a model called PIUnet. The model, at the time of the article's publication, outperformed similar methods in terms of PSNR while also being more computationally efficient.

Chapter 3
Methodology

For this project to go from thought to realisation, many things would need to be accounted for. This includes tasks like: setting up various bits of code to use for things like downscaling images and calculating image quality measurements from the upscaled images, choosing which super-resolution models to use for upscaling the test images and implementing these models in Python to produce the upscaled versions of the images from the test set.

3.1 Thoughts and Concerns

This section addresses some thoughts and potential problems that came up early in the process of this project.

One concern was the edges of the tiles. In the map application, these tiles will sit next to each other, so after upscaling there might be visible lines between the tiles since the upscaling does not take into account any pixels from the adjacent tiles outside the given image.

It is not uncommon for some of these models requiring upwards of 20 GB of dedicated GPU memory to run, which is a lot more than what was available for this project. So training some of the deeper networks would have to be ruled out.

As mentioned in Section 2.1.7, the images in the Arc application are downsampled using a binning style method. Many super-resolution models are optimised towards using bicubic downsampling for the low-resolution input images. If using binning became a problem, there may have been a need to resort to testing on bicubic downsampled images instead.

3.2 Setting up Metric Functions

To measure the performance of the upscaling techniques, both the conventional interpolation and super-resolution, methods for producing the metrics needed to be implemented.

Python has a built-in function for measuring time, which was used to measure how many seconds it takes to upscale an image.

There are also built-in functions that can measure values such as PSNR and SSIM. However, these come from external packages. The one that was used in this project is from the Python package sewar [40].

Since PSNR and SSIM can be quite deceptive at times, a visual assessment can also be a valuable "metric" for measuring how good the upscaled image is. The images are in fact made for humans to look at. Therefore, a visual interpretation and evaluation was done for each upscaled image produced for this report.

3.3 Conventional Interpolation Tests

To test the conventional interpolation methods from Section 2.1.1 a built-in function from the Python package OpenCV [41] was used. The function takes an image and resizes it to a set size using a selected interpolation algorithm. To measure the resulting image quality, the Python package sewar was used for the metrics.

(a) Nearest-neighbour **(b)** Bilinear **(c)** Bicubic

(d) Lanczos **(e)** Original image

Figure 3.1: Results from using conventional interpolation methods

The image that was resized is a 256x256 image of Malmö which shows an area of about 2 km^2, and mainly consists of buildings and roads. The results of the interpolation is shown in Figure 3.1. These images were resized to a size of 512x512 using each of the interpolation methods from Section 2.1.1, and in the figure they are cropped and zoomed in to show closer details of the different interpolation methods.

	Nearest-neighbour	Bilinear	Bicubic	Lanczos
Time (ms)	0.36	0.46	0.52	3.99
PSNR (dB)	34.51	37.59	39.38	39.54
SSIM	0.9395	0.9632	0.9769	0.9776

Table 3.1: The measured results of using the interpolation methods on the Malmö image.

As shown in Table 3.1, Lanczos gives the best values, both for PSNR and SSIM, which is also supported by a visual assessment of the images in Figure 3.1 but it has an execution time which is significantly slower than the rest. This, however, should be fast enough for a map application, but since it still is almost 10 times slower than bicubic without a huge improvement, bicubic interpolation is used in further comparisons to the super-resolution images.

3.4 Description of the Datasets

This project uses different datasets, depending on if the weights are pre-trained or if the network is trained from scratch.

The dataset for training from scratch is a collection of 1000 satellite images with a size of 512x512 that were randomly selected from a single column of images from the north pole all the way to the south pole. The images were downsampled to a size of 256x256 to form 1000 low-res high-res pairs. Out of the 1000 images, 800 were used for training and 200 were used for validation.

Pre-trained weights are trained using various datasets, one of which is the DIV2K dataset [42] which was used in the super-resolution challenges at NTIRE 2017 [43] and 2018 [44], and is commonly used for pre-trained weights. It consists of 1000 images, containing a wide range of subjects, which are divided into 800 training images, 100 validation images and 100 test images.

3.5 Early Experiments

One of the very first experiments consisted of using images from zoom level 3 and 4 to test a small training set, and what super-resolution can do with such a small dataset. This was trained using a method based on ESPCN from Section 2.2.1. The resulting images from upscaling using this network were worse than the low-resolution images, i.e. the network essentially took the low-resolution images and made the images look less similar to the original images.

The same images, from zoom level 3, were then used as input in a network with pre-trained weights based on ESRGAN from Section 2.2.1. In Figure 3.2, which shows the result of this upscaling, we can see that a significant improvement of the low-resolution image has been done, with a PSNR of about 30 dB.

(a) Low-resolution **(b)** Super-resolution **(c)** High-resolution

Figure 3.2: Result from early experiment using ESRGAN.

3.6 Selection of Test Images

The selection of test images for the report includes 10 images of cities, famous landmarks and different terrain. This selection is based on the assumption that a user most likely would like to zoom in on cities and landmarks. Figure 3.3 shows one of the test images in its uncropped form as an example.

Figure 3.3: Uncropped New York image.

3.7 Selection of Models

Since a lot of super-resolution models exist, a selection of a number of models to test had to be made. The resulting selection was based on using models from different categories in 2.2.1, as well as if they were considered interesting, unique or easy to implement. The models were implemented with Python using the PyTorch package.

3.8 SRCNN

The implementation used in this section is based on SRCNN which was introduced in Section 2.2.1. For SRCNN, the input images are pre-processed by using bicubic interpolation to resize the images into the desired size, i.e. from 256x256 to 512x512. This is the only pre-processing that is done for the images in this method. So the "low-resolution" input image shown in Figure 3.4 is actually a bicubic interpolated image with the same pixel size as the output image.

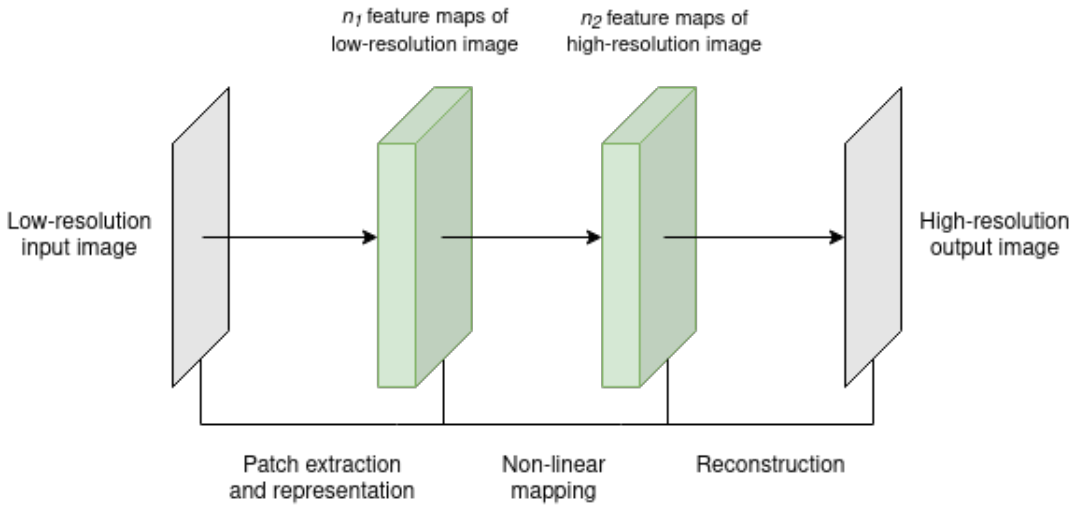


Figure 3.4: The architecture of SRCNN. The arrows represent the convolutional operations.

The first experiment on the SRCNN model was done with a small dataset which consists of 24 images of Paris. This was trained for 5 epochs, mostly to test if the method worked, which resulted in a PSNR of 34.15 dB on the same image of Malmö used in the experiment shown in Figure 3.2. This was then increased to 120 images, still 5 epochs, which gave a PSNR of 34.24 dB, which was only a slight increase. For these experiments the inference time was about 200 ms.

After these smaller experiments, all of which took a surprisingly long time to train, the network was trained on the whole map dataset for 100 epochs. The training process took around 10 hours, and the full results are shown in the results section. Another SRCNN test was done using pre-trained weights. These weights were trained using the T91 dataset. For the PSNR value, the pre-trained weights actually performed slightly better than the ones

trained from scratch. Tests with this SRCNN network is referred to as SRCNN-p later in the text.

Since training took a very long time, even for the network with the simplest architecture, and because the weights trained with the T91 dataset seem to have a similar performance in terms of both image quality and speed, a decision was made to shift focus towards the use of pre-trained weights in later tests.

3.9 EDSR

For EDSR, an architecture based on the model described in Section 2.2.1 was used. The architecture is shown in Figure 3.5. It consists of 16 residual blocks and 64 filters as proposed in the paper for the baseline model on which this implementation is based.

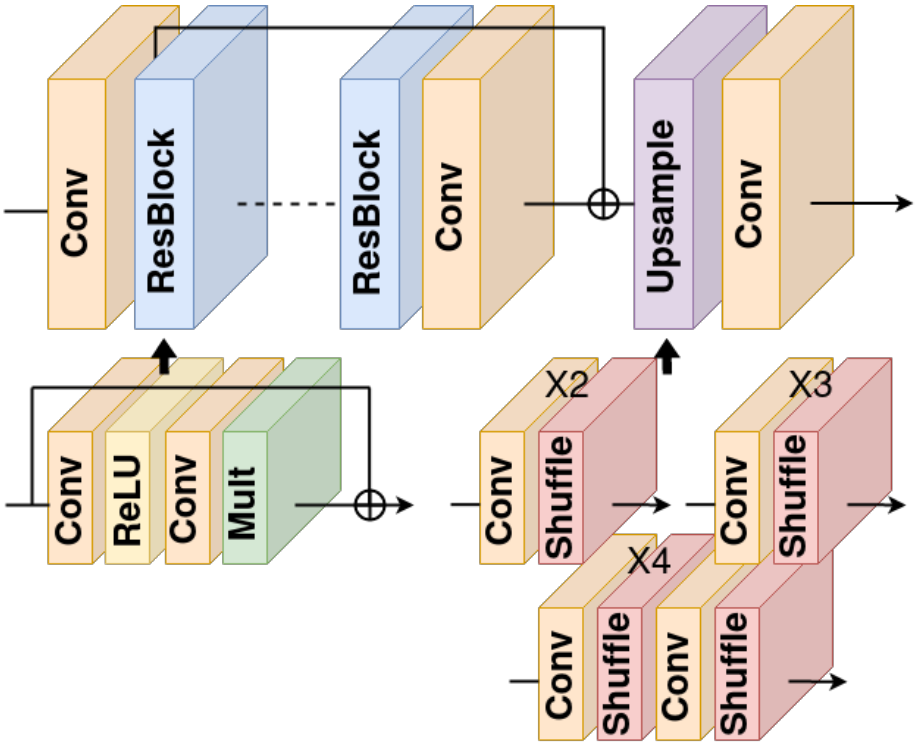


Figure 3.5: The architecture of EDSR. The bottom block architectures represent the residual block and the upsampling block respectively.

Experiments were done initially by training from scratch in a manner similar to the SRCNN training, by gradually increasing the size of the dataset. However, the training time was too slow to be able to produce any usable results. So a test with weights trained with the DIV2K dataset was used in the final comparison. Additionally, bicubic downsampling resulted in a much greater PSNR than binning, so the downsampling method was changed as well.

3.10 SRGAN

The SRGAN model is based on the super-resolution technique described in Section 2.2.1. A visual representation of the generator architecture is shown in Figure 3.6. The PReLU blocks shown refer to parametric ReLU activation function. Figure 3.7 shows the discriminator network, which uses leaky ReLU for activation as well as a sigmoid activation function at the end of the network. The dense part of the network starts with 1024 nodes which all connect to a single node, which ultimately determines whether or not the input image is real or fake.

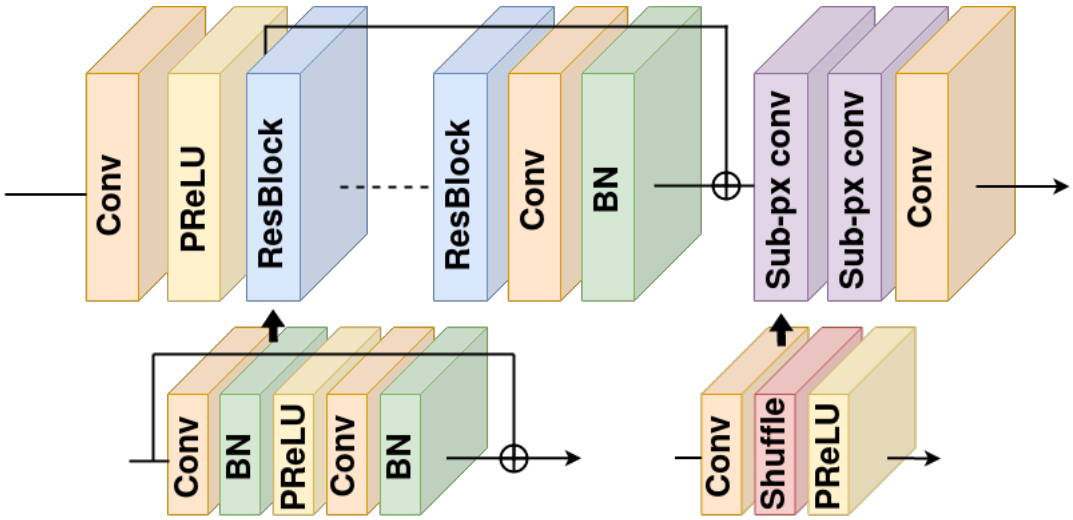


Figure 3.6: The architecture of the generator network in SRGAN.

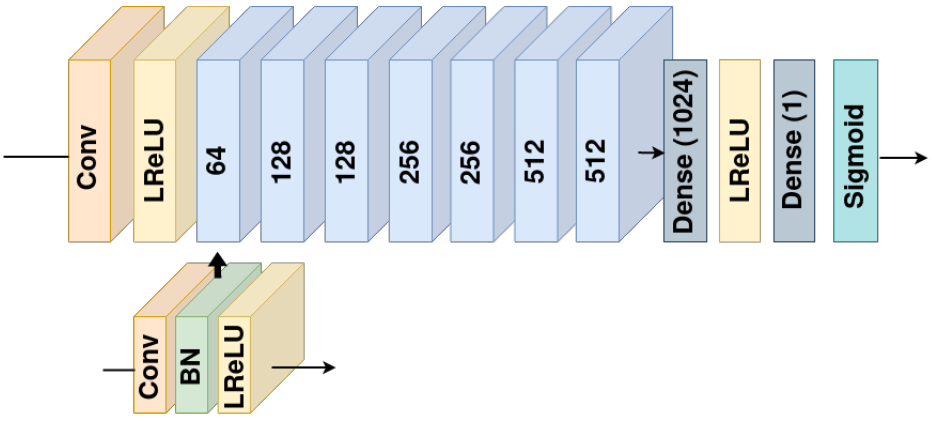


Figure 3.7: The architecture of the discriminator network in SRGAN. The numbers on the convolutional layers represent the number of feature maps.

The SRGAN model was more difficult than anticipated to set up. When time came to train the network it ultimately demanded too much video memory, and would not work. The result images were upscaled using pre-trained weights that were trained on the DIV2K dataset.

3.11 ESRGAN

Since a network with pre-trained weights was used in one of the early experiments, the same model was used on the test images to produce the pictures in the final results. The generator network model is very similar to SRGAN in Figure 3.6, but batch normalisation is removed, and the residual blocks are replaced by residual-in-residual blocks, which are depicted in Figure 3.8.

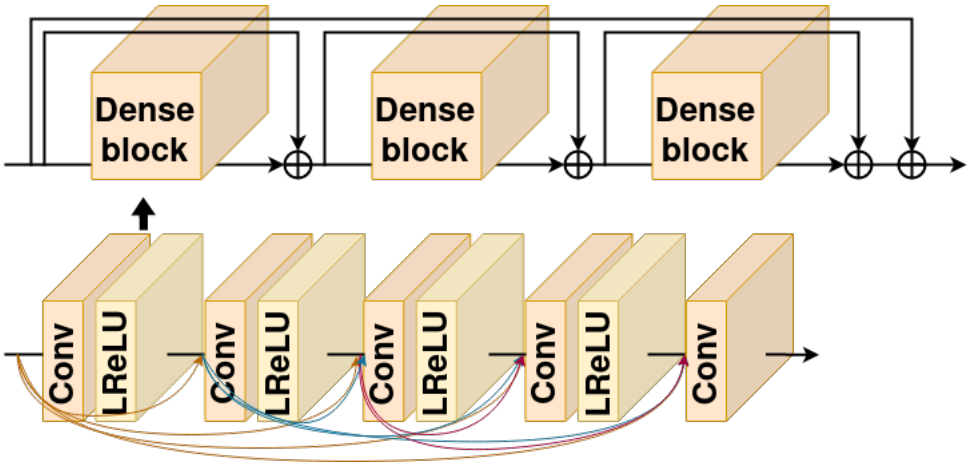


Figure 3.8: The architecture of a Residual-in-Residual Dense Block (RRDB).

3.12 HAT

The HAT implementation is based on the model mentioned in Section 2.2.1. As shown in Figure 3.9, it is a deep and complex model, and it requires around 20 GB of video memory to train, so the intention was, from the start, to use pre-trained weights for this model. The upscaling took a much longer time for this model than the previous models, which makes sense when considering the complexity of the network architecture.

3.13 Test on Edges Between Tiles

To test if a problem could appear where the upscaling creates visible lines between the tiles, due to the upscaling operation only taking into account only a single image at a time, two images were appended next to each other after upscaling them using SRCNN. The result of this is shown in Section 4.13.

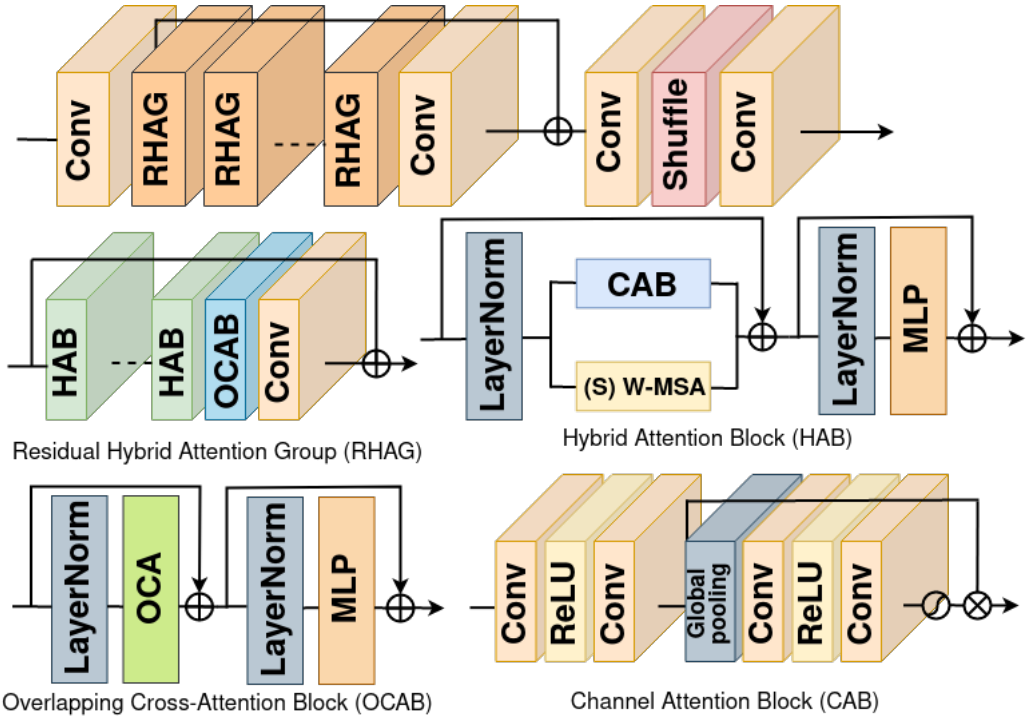


Figure 3.9: The architecture of the HAT super-resolution model.

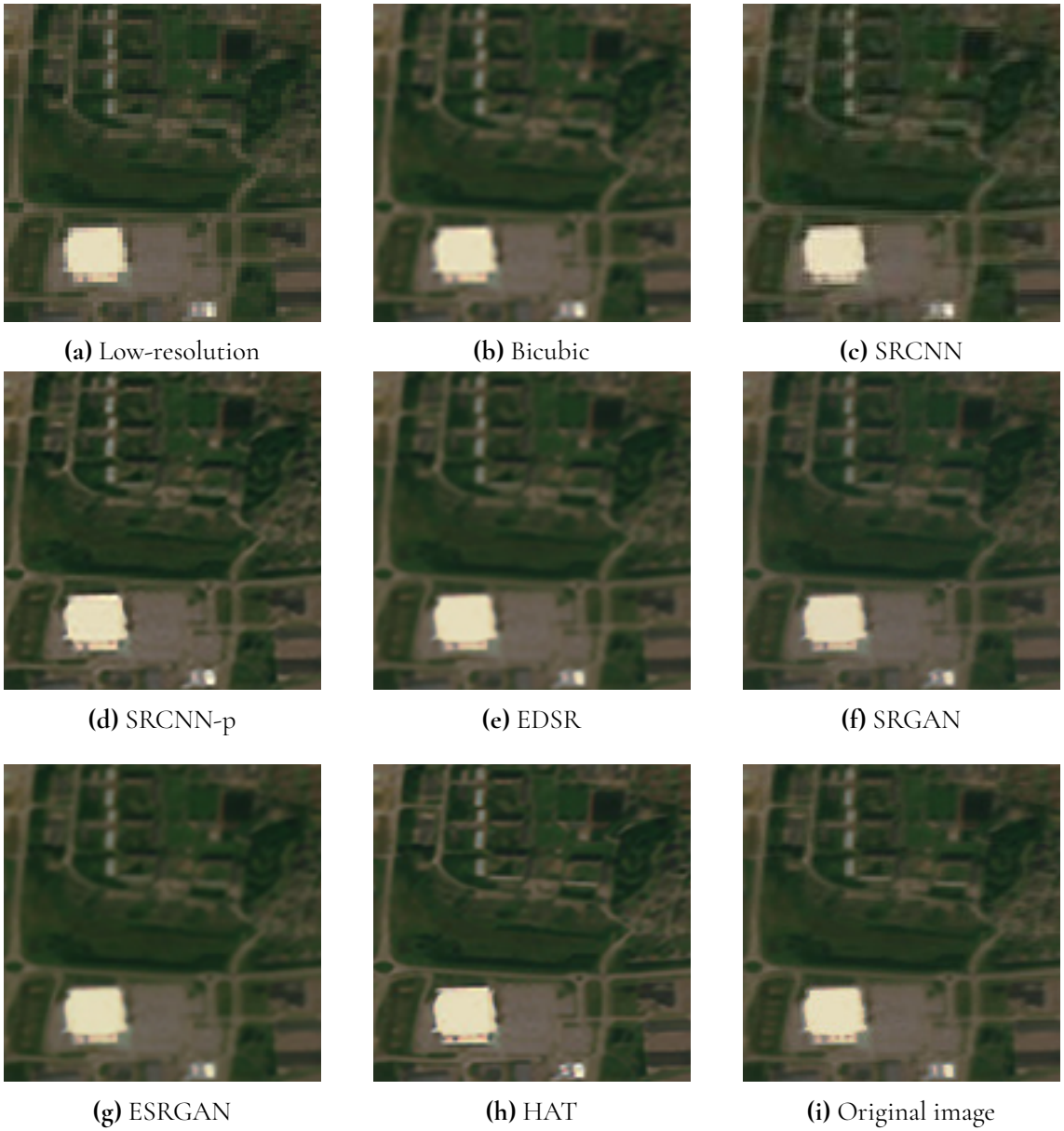
Chapter 4
Results

This chapter presents the results of every upscaling technique on every image in the test set. First, every image is presented separately with a short discussion, as well as a visual assessment, of that image's results with the different super-resolution methods. Then, every metric is presented separately to see which method performs best overall. Lastly, the results from a test of the edges between tiles are shown.

For any time results on SRCNN, it should be noted that the time of the bicubic upscaling should technically be added to the SRCNN time. However, the bicubic upscaling is much faster than SRCNN, so it does not add a significant time delay. Other things to note include that all images shown are zoomed in heavily to see closer upscaling details and that the images for SRGAN were upscaled x4 and then downscaled. This, however, does not add a significant time delay.

4.1 Image 1 - Malmö

The results of the upscaling techniques on the Malmö image are seen in Figure 4.1 and Table 4.1. While bicubic interpolation and EDSR perform best according to the metrics, SRCNN-p and HAT both produce sharper images. Interestingly, these models get a relatively low PSNR and SSIM. It might be that the produced images are too sharp, so the similarity metrics determine that the image is not as similar to the original image as the blurrier results. It is also worth mentioning that HAT is more than ten times slower than ESRGAN which is the second slowest.



(a) Low-resolution **(b)** Bicubic **(c)** SRCNN

(d) SRCNN-p **(e)** EDSR **(f)** SRGAN

(g) ESRGAN **(h)** HAT **(i)** Original image

Figure 4.1: Malmö image upscaled using different super-resolution methods.

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00052	0.23	0.19	2.94	4.98	8.51	95
PSNR (dB)	34.51	39.38	35.14	35.95	41.16	38.01	37.27	36.09
SSIM	0.9395	0.9769	0.9383	0.9566	0.9833	0.9687	0.9774	0.9586

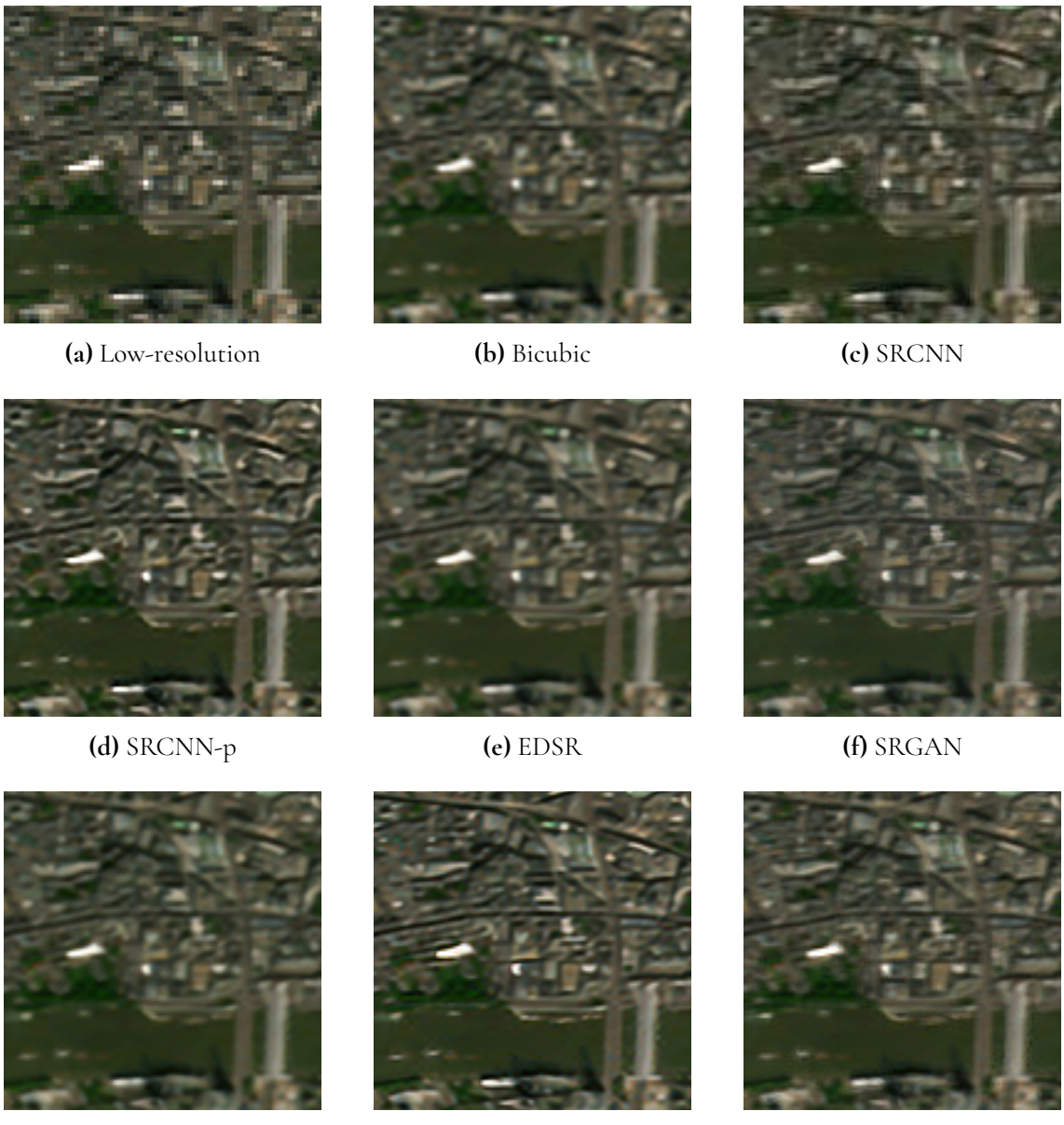
Table 4.1: Image 1 - Malmö

4.2 Image 2 - London

According to Table 4.2, this image, which includes smaller details than the Malmö image, gets closer results between bicubic and SRCNN, while SRCNN and SRCNN-p differ more,

which were closer in the previous image.

The images, shown in Figure 4.2, show that SRCNN-p and HAT are the sharpest again. They do however feature upscaling artefacts which might not be desirable. The metrics also show that EDSR and ESRGAN are the only methods that get a better image quality than bicubic. The images also support this because they look the most similar to the original image.



(a) Low-resolution **(b)** Bicubic **(c)** SRCNN

(d) SRCNN-p **(e)** EDSR **(f)** SRGAN

(g) ESRGAN **(h)** HAT **(i)** Original image

Figure 4.2: London image upscaled using different super-resolution methods.

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.0018	0.28	0.20	2.89	4.95	8.43	115
PSNR (dB)	27.95	31.48	30.04	28.02	33.11	30.88	31.59	27.91
SSIM	0.8921	0.9443	0.9202	0.9061	0.9594	0.9310	0.9523	0.9073

Table 4.2: Image 2 - London

4.3 Image 3 - Central Park

(a) Low-resolution **(b)** Bicubic **(c)** SRCNN

(d) SRCNN-p **(e)** EDSR **(f)** SRGAN

(g) ESRGAN **(h)** HAT **(i)** Original image

Figure 4.3: Central Park image upscaled using different super-resolution methods.

Again, only EDSR and ESRGAN produce a better image quality than bicubic interpola-

tion according to the PSNR and SSIM values in Table 4.3. HAT produces an image that gets a lower PSNR but a higher SSIM than the low-resolution image.

In the image results (Figure 4.3), the ones that stick out are SRCNN-p, SRGAN and HAT. SRCNN-p and SRGAN looks sharper than the results for this image as well and SRGAN produces a colour on the buildings that is slightly off. EDSR looks the most similar to the original image, which the metrics also indicate.

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00064	0.23	0.21	2.93	4.93	8.48	109
PSNR (dB)	25.21	29.07	27.73	25.15	30.89	26.57	29.53	24.90
SSIM	0.8951	0.9498	0.9276	0.9122	0.9658	0.9106	0.9573	0.9090

Table 4.3: Image 3 - Central Park

4.4 Image 4 - Countryside

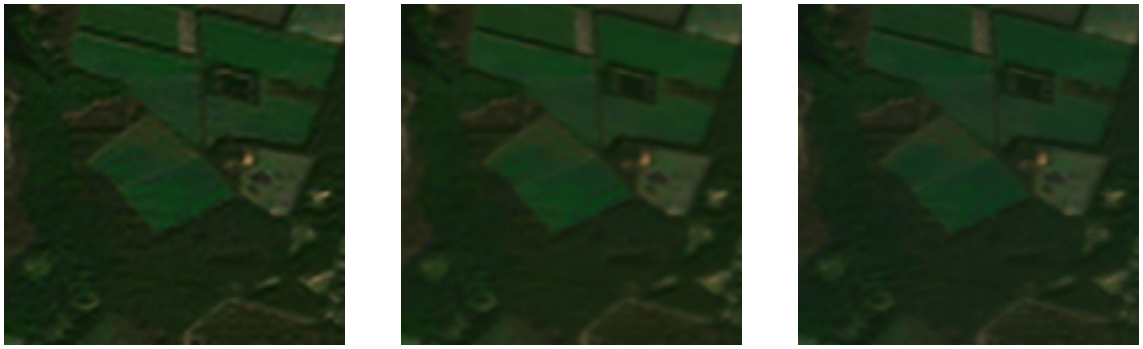
For this image, the measured values in Table 4.4 show that all methods except SRCNN are an improvement over the low-resolution image. EDSR is the only method that outperforms bicubic interpolation. It might also be worth noting that, SRCNN-p, SRGAN, ESRGAN and HAT are all similar in PSNR, but SRCNN-p and HAT are significantly lower in SSIM.

In the produced images shown in Figure 4.4, it is hard to discern differences between the images, but it seems that HAT is sharpest again and that SRCNN-p contains many upscaling artefacts. Otherwise, the resulting images look quite similar.

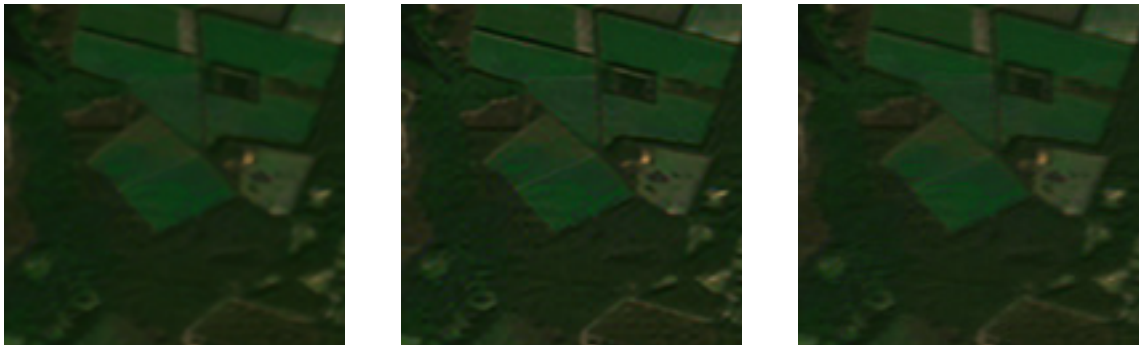
	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00051	0.23	0.21	2.89	4.97	8.83	100
PSNR (dB)	36.27	39.85	36.07	37.05	41.22	37.33	37.71	37.04
SSIM	0.9403	0.9689	0.9220	0.9509	0.9752	0.9624	0.9653	0.9494

Table 4.4: Image 4 - Countryside

(a) Low-resolution **(b)** Bicubic **(c)** SRCNN



(d) SRCNN-p **(e)** EDSR **(f)** SRGAN



(g) ESRGAN **(h)** HAT **(i)** Original image

Figure 4.4: Countryside image using different sr methods

4.5 Image 5 - Miami Beach

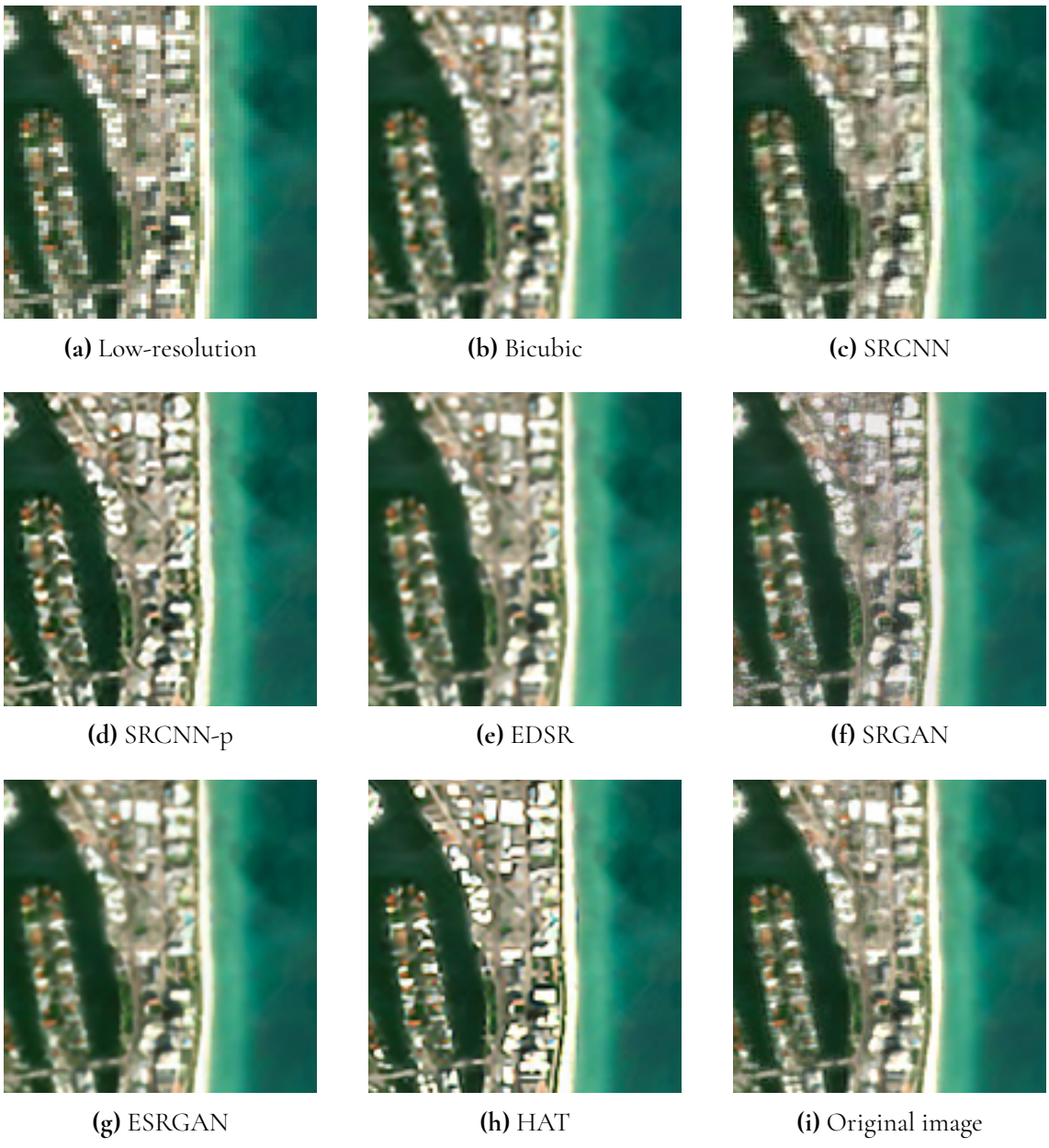
For this image, HAT produces an improvement over the low-resolution image when measured in SSIM, but worse when measured in PSNR. Like in some of the earlier results, EDSR and ESRGAN scores a higher SSIM and PSNR than bicubic interpolation. SRCNN produces a significant improvement over the LR image in PSNR, but gets a worse SSIM. All the measured values are shown in Table 4.5.

As is evident from Figure 4.5, the SRCNN-p image looks quite messy. The sharpness is still there, but the high frequency details do not seem to go well with it. SRGAN produces a strange gray-like texture on the buildings in the center of the image. The image produced by the HAT network has a high contrast, compared to the other images. The sharpness does not

make it look as messy as SRCNN-p, but the buildings are still too bright and defined. Most of the methods resolve the clearer high frequency details in the lower part of the image but some of them struggle with the more cluttered details further up.

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00048	0.23	0.20	2.90	4.96	8.43	116
PSNR (dB)	25.67	28.99	27.38	26.16	30.53	27.05	29.56	24.80
SSIM	0.9327	0.9641	0.9312	0.9431	0.9732	0.9382	0.9645	0.9364

Table 4.5: Image 5 - Miami Beach



(a) Low-resolution **(b)** Bicubic **(c)** SRCNN

(d) SRCNN-p **(e)** EDSR **(f)** SRGAN

(g) ESRGAN **(h)** HAT **(i)** Original image

Figure 4.5: Miami image upscaled using different super-resolution methods.

4.6 Image 6 - Rocky Landscape

(a) Low-resolution

(b) Bicubic

(c) SRCNN

(d) SRCNN-p

(e) EDSR

(f) SRGAN

(g) ESRGAN

(h) HAT

(i) Original image

Figure 4.6: Rocky Landscape image upscaled using different super-resolution methods.

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00053	0.23	0.19	2.92	4.92	8.43	116
PSNR (dB)	40.24	43.98	39.08	40.68	44.75	39.65	38.84	41.36
SSIM	0.9638	0.9785	0.9354	0.9626	0.9815	0.9608	0.9723	0.9647

Table 4.6: Image 6 - Rocky Landscape

For these results (Table 4.6), HAT gets a higher PSNR than ESRGAN but it is the other

way around for SSIM. ESRGAN gets the lowest PSNR of all methods, including the low-resolution image, which is the first time this has happened for ESRGAN. Out of all models, only EDSR is higher than bicubic for both metrics. Furthermore, SRCNN, SRGAN and ESRGAN all receive a worse PSNR than the low-resolution image, but ESRGAN gets a higher SSIM.

This time the differences in the produced images, shown in Figure 4.6, are hard to see. The HAT image is the most clear this time. The lack of small details in the image seems to be a benefit to the sharper results that HAT produces. The image looks like it has a higher resolution than the rest, including the original image. Since the original image is blurry, HAT obviously does not look very similar to it compared to other methods.

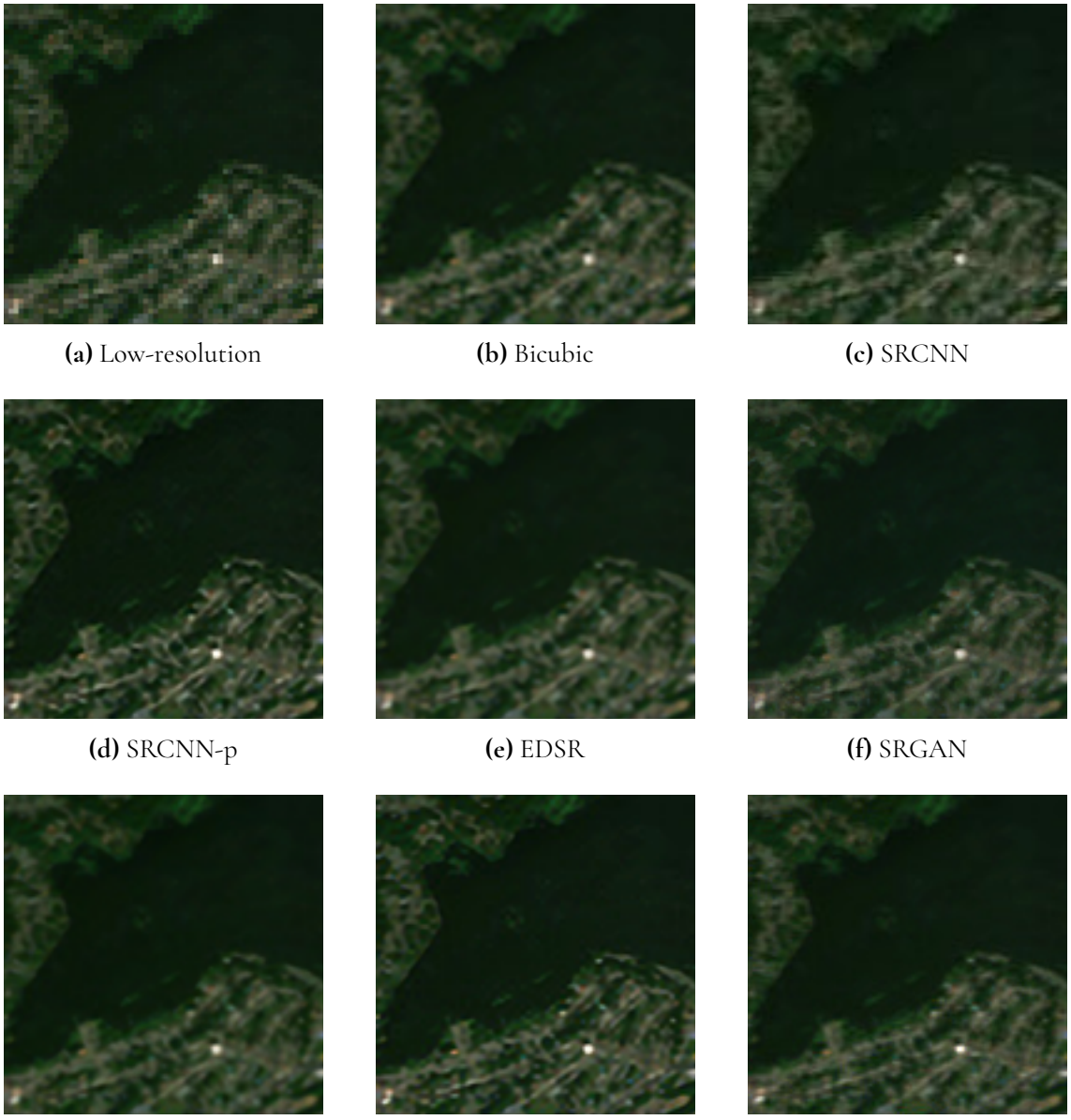
4.7 Image 7 - Lake

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00055	0.24	0.21	2.83	4.93	8.63	98
PSNR (dB)	31.90	35.86	32.96	32.93	37.81	34.72	35.43	32.53
SSIM	0.9259	0.9581	0.9140	0.9360	0.9675	0.9395	0.9551	0.9313

Table 4.7: Image 7 - Lake

Out of all the network models, EDSR is the only method which is an improvement over bicubic, both for PSNR and SSIM. This time, all methods get a higher PSNR than the low-res image, which seems to be a rare occurrence. This is true for SSIM too, except for SRCNN, which gets a lower value. Interestingly, as opposed to the previous image, HAT is lower than ESRGAN again for PSNR. This image, especially in its uncropped form, has more high frequency details than the previous, which could be an explanation for why HAT performs worse.

As evident from Figure 4.7, all the upscaled images look similar except for SRCNN-p and HAT, which seems to be a pattern. Further details between these images, are hard to discern.



(a) Low-resolution **(b)** Bicubic **(c)** SRCNN

(d) SRCNN-p **(e)** EDSR **(f)** SRGAN

(g) ESRGAN **(h)** HAT **(i)** Original image

Figure 4.7: Lake image upscaled using different super-resolution methods.

4.8 Image 8 - Paris

From Table 4.8, we can see that HAT scores the lowest PSNR, even lower than the low-resolution image, which was also true for the Miami image. SRCNN and SRCNN-p are significantly different in PSNR, while for the previous image, they were very close. EDSR and ESRGAN are the only improvements over bicubic interpolation. There seems to be a pattern where EDSR always takes the top spot and the second spot alternates between being occupied by ESRGAN and bicubic interpolation. For SSIM, all the super-resolution methods score higher than the low-resolution image. EDSR is has the highest SSIM, followed by ESRGAN, which aligns with the PSNR scores.

In the images, shown in Figure 4.8, HAT works out the building details better this time.

However, HAT (and SRCNN-p) also seems to amplify the small high contrast bits in the image, which the other methods do not. Overall, EDSR, SRGAN and ESRGAN look the most similar to the original image.

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00052	0.23	0.19	2.89	4.94	8.53	119
PSNR (dB)	26.42	29.65	28.24	26.33	31.18	28.52	29.90	26.17
SSIM	0.8712	0.9310	0.9042	0.8900	0.9500	0.9053	0.9431	0.8904

Table 4.8: Image 8 - Paris

(a) Low-resolution **(b)** Bicubic **(c)** SRCNN

(d) SRCNN-p **(e)** EDSR **(f)** SRGAN

(g) ESRGAN **(h)** HAT **(i)** Original image

Figure 4.8: Paris image upscaled using different super-resolution methods. The dark blob in the lower right corner is the Eiffel Tower.

4.9 Image 9 - Versailles

(a) Low-resolution

(b) Bicubic

(c) SRCNN

(d) SRCNN-p

(e) EDSR

(f) SRGAN

(g) ESRGAN

(h) HAT

(i) Original image

Figure 4.9: Versailles image upscaled using different super-resolution methods.

According to the metrics in Figure 4.9, every method improves the low-resolution image, both for PSNR and SSIM. This time, only EDSR is better than bicubic interpolation again, pushing down ESRGAN to a close third spot.

In the images in Figure 4.9, all the upscaling methods seem to have produced images that do not look bad. As with all the other test images, SRCNN-p and HAT's resulting images are sharper than the others, with HAT looking slightly cleaner.

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00047	0.23	0.22	2.89	4.95	8.41	113
PSNR (dB)	29.19	32.69	30.48	29.52	34.23	31.83	32.60	29.34
SSIM	0.9047	0.9499	0.9114	0.9193	0.9628	0.9345	0.9551	0.9194

Table 4.9: Image 9 - Versailles

4.10 Image 10 - Abstract Sea Image

(a) Low-resolution **(b)** Bicubic **(c)** SRCNN

(d) SRCNN-p **(e)** EDSR **(f)** SRGAN

(g) ESRGAN **(h)** HAT **(i)** Original image

Figure 4.10: Sea image upscaled using different super-resolution methods.

As shown in Figure 4.10, this is the only test image where bicubic interpolation gets

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Time (s)	N/A	0.00049	0.25	0.21	2.91	4.93	8.50	111
PSNR (dB)	45.82	49.31	42.09	45.66	49.35	39.58	39.53	45.82
SSIM	0.9864	0.9922	0.9657	0.9854	0.9916	0.9750	0.9855	0.9887

Table 4.10: Image 10 - Sea

the highest score, which it does for SSIM. For PSNR, EDSR is slightly higher than bicubic interpolation.

It is unclear what the image, shown in 4.10, depicts. In the full image, it looks like it might be the sea and some islands, and some clouds seem to be visible as well. It is quite hard to see a significant difference between the super-resolution techniques for this image. Like in some of the preceding examples, HAT seems to work well with this low-detailed image. Other than that, any method, except maybe SRCNN, resolve the details in the image in a satisfying manner.

4.11 Metric Tables

Image	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Malmö	0.00052	0.23	0.19	2.94	4.98	8.51	95
London	0.0018	0.28	0.20	2.89	4.95	8.43	115
Park	0.00064	0.23	0.21	2.93	4.93	8.48	109
Country	0.00051	0.23	0.21	2.89	4.97	8.83	100
Miami	0.00048	0.23	0.20	2.90	4.96	8.43	116
Rocky	0.00053	0.23	0.19	2.92	4.92	8.43	116
Lake	0.00055	0.24	0.21	2.83	4.93	8.63	98
Paris	0.00052	0.23	0.19	2.89	4.94	8.53	119
Versailles	0.00047	0.23	0.22	2.89	4.95	8.41	113
Sea	0.00049	0.25	0.21	2.91	4.93	8.50	111

Table 4.11: Upscaling time for every image in seconds.

Not much of interest can be deduced from the table of upscaling times (4.11). Bicubic interpolation is the fastest, and HAT is, by far, the slowest. EDSR, which is the best performing method according to the image quality metrics, has one of the faster upscaling times when comparing to the other models, but it is still about 15 times slower than the SRCNN implementations.

The Central Park and Miami images consistently get the lowest PSNR, as seen in Table 4.12. The images with many buildings, like the city images, seem to be harder to upscale overall. The table also confirms that the second best upscaling technique alternates between bicubic interpolation and ESRGAN. Bicubic is second best for all the non-city images, so it might be a pattern there, but it is also second best for the Malmö and the Versailles images, which contain city textures.

Image	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Malmö	34.51	39.38	35.14	35.95	**41.16**	38.01	37.27	36.09
London	27.95	31.48	30.04	28.02	**33.11**	30.88	31.59	27.91
Park	25.21	29.07	27.73	25.15	**30.89**	26.57	29.53	24.90
Country	36.27	39.85	36.07	37.05	**41.22**	37.33	37.71	37.04
Miami	25.67	28.99	27.38	26.16	**30.53**	27.05	29.56	24.80
Rocky	40.24	43.98	39.08	40.68	**44.75**	39.65	38.84	41.36
Lake	31.90	35.86	32.96	32.93	**37.81**	34.72	35.43	32.53
Paris	26.42	29.65	28.24	26.33	**31.18**	28.52	29.90	26.17
Versailles	29.19	32.69	30.48	29.52	**34.23**	31.83	32.60	29.34
Sea	45.82	49.31	42.09	45.66	**49.35**	39.58	39.53	45.82
Average	32.31	36.03	32.92	32.75	**37.42**	33.41	34.20	32.60

Table 4.12: PSNR for every image. Numbers in bold indicate the highest value and underlined numbers indicate the second highest value.

Image	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
Malmö	0.9395	0.9769	0.9383	0.9566	**0.9833**	0.9687	0.9774	0.9586
London	0.8921	0.9443	0.9202	0.9061	**0.9594**	0.9310	0.9523	0.9073
Park	0.8951	0.9498	0.9276	0.9122	**0.9658**	0.9106	0.9573	0.9090
Country	0.9403	0.9689	0.9220	0.9509	**0.9752**	0.9624	0.9653	0.9494
Miami	0.9327	0.9641	0.9312	0.9431	**0.9732**	0.9382	0.9645	0.9364
Rocky	0.9638	0.9785	0.9354	0.9626	**0.9815**	0.9608	0.9723	0.9647
Lake	0.9259	0.9581	0.9140	0.9360	**0.9675**	0.9395	0.9551	0.9313
Paris	0.8712	0.9310	0.9042	0.8900	**0.9500**	0.9053	0.9431	0.8904
Versailles	0.9047	0.9499	0.9114	0.9193	**0.9628**	0.9345	0.9551	0.9194
Sea	0.9864	**0.9922**	0.9657	0.9854	0.9916	0.9750	0.9855	0.9887
Average	0.8265	0.9614	0.9270	0.9362	**0.9710**	0.9426	0.9628	0.9356

Table 4.13: SSIM for every image. Numbers in bold indicate the highest value and underlined numbers indicate the second highest value.

The SSIM results (Table 4.13) seem to follow PSNR, for the most part, in terms of which method is best and second best. For SSIM there is a trend where the London, Central Park and Paris images get the lowest overall values. It is also for SSIM where the only time EDSR is not the best method happens. Bicubic interpolation gets a higher SSIM score for the sea image.

The models and their average PSNR and SSIM scores ranked are shown in Table 4.14. All the methods produced a better image overall, than simply keeping the low-resolution image.

Ranking	Method	PSNR
1	EDSR	37.42
2	Bicubic	36.03
3	ESRGAN	34.20
4	SRGAN	33.41
5	SRCNN	32.92
6	SRCNN-p	32.75
7	HAT	32.60
8	LR	32.31

Ranking	Method	SSIM
1	EDSR	0.9710
2	ESRGAN	0.9628
3	Bicubic	0.9614
4	SRGAN	0.9426
5	SRCNN-p	0.9362
6	HAT	0.9356
7	SRCNN	0.9270
8	LR	0.9252

Table 4.14: PSNR and SSIM rankings.

4.12 Space Saved

Table 4.15 shows how much storage space the weights require. As mentioned in the introduction chapter, zoom level 12 and zoom level 13 use 68 GB and 241 GB of storage space respectively, so in relation to the size of the images, the sizes of these network are insignificant. The storage space saved is around 173 GB.

Model	Size
Bicubic	413 B
SRCNN	231.1 kB
SRCNN-p	231.6 kB
EDSR	6.2 MB
SRGAN	6.5 MB
ESRGAN	20.6 MB
HAT	82.6 MB

Table 4.15: Storage space used by the different models.

4.13 Edges Between Tiles

As clarified in the lowermost image in Figure 4.11, an edge appears between tiles. The figure shows three levels of zoom on the edge between the tiles. The edge, however, is subtle, and can only be seen when zooming in to a degree which would most likely not be done during common use.

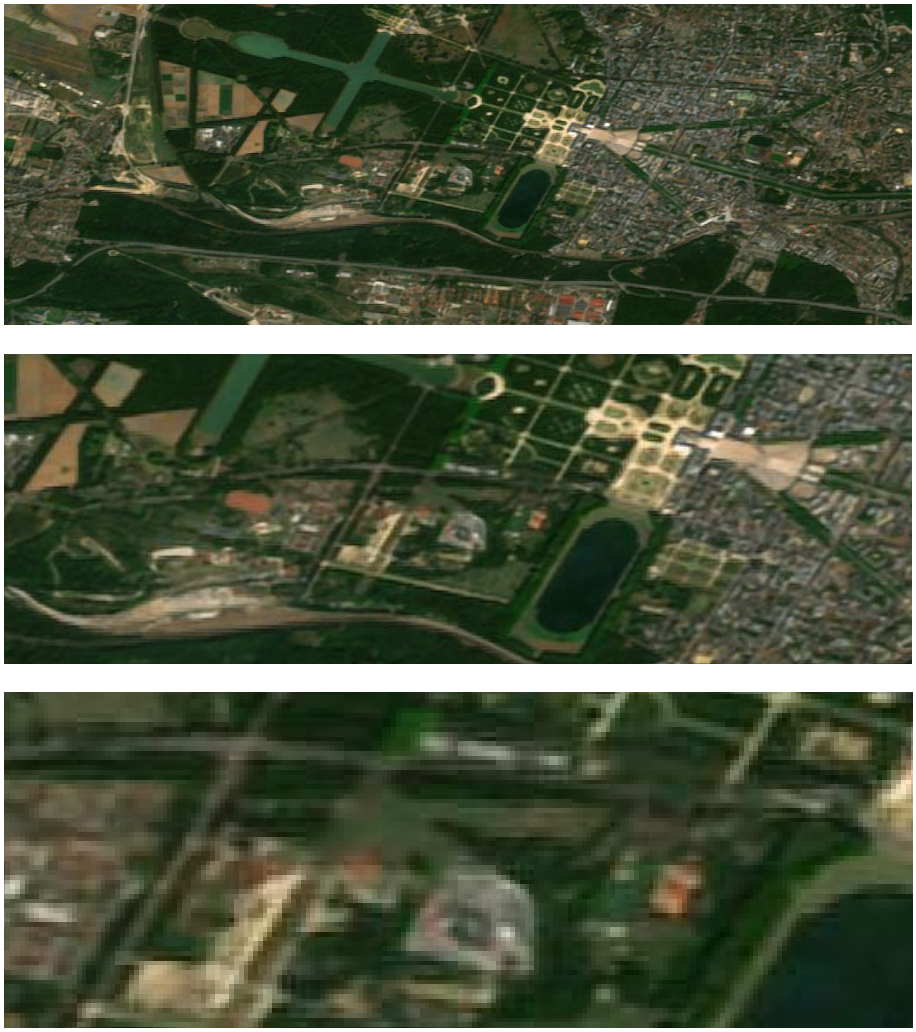


Figure 4.11: Result of displaying two adjacent SRCNN-upscaled images. A subtle vertical line is visible in the middle of the bottom image.

Chapter 5
Discussion

This chapter contains a discussion of the results shown in the previous chapter, as well as various topics derived from the process of producing results during the course of this project.

5.1 Ambiguous Results

One of the main dilemmas surrounding super-resolution is that it should not be used when ambiguous results can be a problem. The information that the upscaling creates is only estimated and, therefore, does not actually exist. There is no guarantee that the details we see in the upscaled image are real.

For a map application ambiguous results is not an issue, to a degree. Most important is probably that the images look good to the users of the application, and if some details are not resolved entirely correctly, this is most likely not an issue. It becomes an issue when they start becoming distracted by strange upscaling artefacts.

5.2 Original Image Similarity

In some of the upscaled results, the super-resolution image is actually clearer than the original image. As mentioned, the details that show up in the SR image are not actually real details. The question then is what a user would prefer. They might look at the less blurry SR image and think that it looks better, unaware of the fact that what they are seeing is fake. The clearer results can also result in a lower image quality value (PSNR and SSIM), which is a reason why those values can be misleading.

If what you are after is an as accurate representation of the original image as possible, then high metric values are good, but if you want an image that looks as "good" as possible, which is obviously subjective, the values do not accurately represent this. This is shown, for

example, in the images that lack smaller details, where HAT creates a clear and defined result but receives a PSNR and SSIM that is among the lower scores.

5.3 Difficulty of Upscaling Satellite Imagery

Images at zoom level 12 have very small details, particularly the images depicting cities. This essentially means that the images contain unpredictable textures, which are not ideal for using with super-resolution, and machine learning in general.

There is also the issue of downsampling. When using images downsampled with binning for upscaling, the results did not come out well. So to use super-resolution in a system like this, the downsampling method used to generate the lower resolution images would need to be bicubic.

5.4 Comparison of SR Methods

Upscaling time is an issue for the deeper networks, which is not ideal for a map application, where the images need to be displayed when a user zooms in, without a too distracting delay. Bicubic interpolation, SRCNN and SRCNN-p, with their fast upscaling speeds, would probably work without any issues. However, HAT can be ruled out, because it takes almost two minutes for it to upscale one image. Of course, it is not only up to me to decide what is too slow, but ESRGAN at roughly 8.5 seconds, and SRGAN at around 5 seconds, cause a delay which seems like it would be distracting. EDSR sits on the limit at around 3 seconds which is still quite slow, but considering its good image results, it could work.

For some images with no small details, like the sea image, the upscaling method does not seem to matter that much, and any upscaling technique seems to produce an image that looks good enough. Because, while the measured values vary, the images are not very easy to distinguish between.

From the upscaled images, it becomes clear that HAT performs better on images that do not contain a lot of small details, so different models work better on different types of images. While it is not viable to use HAT, because its upscaling speed is too slow, there might be another super-resolution method which also carries this attribute, but also has a more useful upscaling speed. This could be used in a system that classifies satellite images into different types, and uses a corresponding super-resolution model for upscaling that image.

Looking at Table 4.12 and Table 4.13, it is clear that EDSR generates the best quality images in relation to the original images out of the methods tested. This is also supported by a visual assessment of the upscaled images; the images produced by EDSR consistently look the most similar to the original images. Bicubic interpolation gets high metric results, while in the shown images it is one of the blurrier results. The images produced still look good, and the roughness in the low-resolution images has definitely been smoothed out, so considering its fast upscaling speed it is still a useful alternative. ESRGAN's results look very similar to EDSR and, considering its complexity and its slower upscaling speed, it is not very useful for this task. This is also the case for SRGAN, which is also slower and produces lower quality results compared to EDSR. SRCNN is low in the ranking of the models, but in a visual assessment the results are similar to bicubic interpolation. However, considering that bicubic

interpolation has a speed of around 0.5 ms, while SRCNN takes around 230 ms, SRCNN becomes redundant. SRCNN-p's sharper images, might be useful for some applications. The upscaling artefacts, which it is prone to, are undesirable. The results produced by HAT do not go together well with the methods of measurement, but some of the images look good, and they have a sharpness that could be desirable. However, as mentioned, HAT is clearly too slow to be used in an application such as this.

Chapter 6
Conclusions

Out of the methods tested, EDSR gives the best results according to the metrics, which is also supported by the images. Its upscaling speed, which is about 3 seconds, makes it uncertain if it could be used in an application such as Arc. The size of the network in terms of storage space needed, was found to be of no issue.

Regarding if super-resolution could be used to replace the original images in a map application, the potential is there. While it is possible to produce images which are very similar to the original images, the biggest issue is the upscaling speed. If a speed of 3 seconds per image is tolerable, EDSR is definitely a candidate, otherwise another method would have to be identified and tested.

6.1 Further Research

Further research of this project includes trying more models to find if one can get better upscaling speed than EDSR with similar results, as well as training EDSR from scratch to see if it can produce improved results.

If a system that follows the tiled map model does not implement any super-resolution, it could still be a good idea to use bicubic interpolation to create a fake zoom level 14, since it needs little storage space, has a fast upscaling speed and, as shown by the metric results, produces reasonably good quality images. Machine learning based image upscaling could also, of course, be applied to the images in zoom level 13 to produce a zoom level 14.

Another thing that would be interesting to explore would be using a larger scaling factor, like 4x for example. While this would not resolve any more details, it could result in a less pixelated image, which in turn would mean that a user could zoom in further while maintaining smooth edges. An issue with this though is that the produced images would be 1024x1024, so the system would need to be made compatible with displaying twice as large images in the same space as the 512x512 images are displayed.

References

[1] Tactel, "Arc: Exploring the world below from the sky above." URL: https://tactel.se/en/cases/exploring-the-world-below-from-the-sky-above.

[2] Wikipedia, "Tiled web map." URL: https://en.wikipedia.org/wiki/Tiled_web_map.

[3] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2022. ISBN: 9783030343712.

[4] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer International Publishing, 2018. ISBN: 9783319944623.

[5] D. R. Bull and F. Zhang, *Intelligent Image and Video Compression*. Oxford: Academic Press, second ed., 2021. ISBN: 9780128203538.

[6] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *arXiv preprint arXiv:1501.00092*, 2015. URL: http://arxiv.org/abs/1501.00092.

[7] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1646–1654, 2016.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[9] S. Anwar, S. Khan, and N. Barnes, "A deep journey into super-resolution: A survey," *ACM Computing Surveys*, vol. 53, pp. 1–34, May 2020.

[10] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Computer Vision – ECCV 2016*, pp. 391–407, Springer International Publishing, 2016.

[11] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1874–1883, IEEE Computer Society, June 2016.

[12] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1132–1140, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[14] A. C. N. Matcha, "Image super-resolution: A comprehensive review," 2020. URL: `https://blog.paperspace.com/image-super-resolution/`.

[15] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Computer Vision – ECCV 2018*, pp. 256–272, Springer International Publishing, 2018.

[16] Y. Fan, H. Shi, J. Yu, D. Liu, W. Han, H. Yu, Z. Wang, X. Wang, and T. S. Huang, "Balanced two-stage residual networks for image super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1157–1164, 2017.

[17] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1637–1645, 2016.

[18] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2790–2798, 2017.

[19] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5835–5843, 2017.

[20] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, IEEE Computer Society, July 2017.

[21] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4809–4817, 2017.

[22] Y. Hu, X. Gao, J. Li, Y. Huang, and H. Wang, "Single image super-resolution via cascaded multi-scale cross network," *arXiv preprint arXiv:1802.08808*, 2018. URL: `https://arxiv.org/abs/1802.08808`.

[23] Z. Hui, X. Wang, and X. Gao, "Fast and accurate single image super-resolution via information distillation network," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 723–731, IEEE Computer Society, June 2018.

[24] J.-S. Choi and M. Kim, "A deep convolutional neural network with selection units for super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1150–1156, 2017.

[25] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Computer Vision – ECCV 2018*, pp. 294–310, Springer International Publishing, 2018.

[26] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, IEEE Computer Society, July 2017.

[27] M. M. Sajjadi, B. Schölkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4501–4510, IEEE Computer Society, Oct. 2017.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, IEEE Computer Society, June 2015.

[29] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Computer Vision – ECCV 2018 Workshops*, pp. 63–79, Springer International Publishing, 2019.

[30] W. Sun and Z. Chen, "Learned image downscaling for upscaling using content adaptive resampler," *IEEE Transactions on Image Processing*, vol. 29, pp. 4027–4040, 2020.

[31] S. H. Park, Y. S. Moon, and N. I. Cho, "Perception-oriented single image super-resolution using optimal objective estimation," *arXiv preprint arXiv:2211.13676*, 2022. URL: `https://arxiv.org/abs/2211.13676`.

[32] X. Chen, X. Wang, J. Zhou, and C. Dong, "Activating more pixels in image super-resolution transformer," *arXiv preprint arXiv:2205.04437*, 2022. URL: `https://arxiv.org/abs/2205.04437`.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[34] E. Alimohammadi, "Enhancing satellite imagery with deep learning: A practical guide (super-resolution)," 2019. URL: `https://itnext.io/satellite-view-images-enhancement-with-deep-learning-super-resolution-92040d6b22b6`.

[35] J. Tan, "Enhancing satellite imagery through super resolution," 2020. URL: `https://omdena.com/blog/super-resolution/`.

[36] M. Tayba and P. Rivas, "Enhancing the resolution of satellite imagery using a generative model," in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 20–25, 2021.

[37] M. U. Müller, N. Ekhtiari, R. M. Almeida, and C. Rieke, "Super-resolution of multispectral satellite images using convolutional neural networks," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. V-1-2020, pp. 33–40, Aug. 2020.

[38] M. Rifat Arefin, V. Michalski, P.-L. St-Charles, A. Kalaitzis, S. Kim, S. E. Kahou, and Y. Bengio, "Multi-image super-resolution for remote sensing using deep recurrent networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 816–825, 2020.

[39] D. Valsesia, "The missing ingredient in deep multi-temporal satellite image super-resolution," 2021. URL: `https://towardsdatascience.com/the-missing-ingredient-in-deep-multi-temporal-satellite-image-super-resolution-78cac0f063d9`.

[40] A. Khalel, "sewar." URL: `https://pypi.org/project/sewar`.

[41] OpenCV-Team, "Opencv." URL: `https://pypi.org/project/opencv-python/`.

[42] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1122–1131, 2017.

[43] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, *et al.*, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[44] R. Timofte, S. Gu, J. Wu, L. Van Gool, L. Zhang, M.-H. Yang, M. Haris, *et al.*, "Ntire 2018 challenge on single image super-resolution: Methods and results," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

Appendices

Appendix A
Unused Metrics

This section includes the measured values of each image using metrics that were not used in the main evaluation of the super-resolution methods.

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9962	0.9987	0.9958	0.9970	0.9991	0.9974	0.9955	0.9973
MSSSIM	0.9853	0.9923	0.9820	0.9876	0.9961	0.9911	0.9944	0.9904
PSNRB	34.0228	39.7241	34.4985	35.9239	41.3250	37.5973	36.4331	36.2512
ERGAS	3559.3009	2021.8384	3566.4328	3050.7063	1699.1113	2618.1332	2910.2305	2938.4011
RASE	512.6029	291.2837	512.6750	439.8728	244.7969	376.0097	417.7597	423.1265
SAM	0.0841	0.0486	0.0795	0.0725	0.0397	0.0537	0.0454	0.0702
SCC	0.4103	0.6713	0.4819	0.6629	0.7168	0.5992	0.6814	0.7157
VIFP	0.5525	0.7279	0.5606	0.6692	0.7717	0.6785	0.7381	0.6697

Table A.1: Image 1 - Malmö

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9880	0.9943	0.9912	0.9878	0.9959	0.9924	0.9915	0.9878
MSSSIM	0.9649	0.9782	0.9768	0.9693	0.9882	0.9795	0.9863	0.9743
PSNRB	27.4814	31.1706	29.3754	27.8474	32.7114	30.7230	30.9703	27.6432
ERGAS	6586.1995	4381.9118	5422.4065	6565.3180	3704.7203	4909.5923	4596.1186	6527.1815
RASE	949.0123	631.7395	781.1944	946.9939	534.0997	707.5762	661.8260	940.8257
SAM	0.1498	0.1002	0.1185	0.1475	0.0831	0.1048	0.0874	0.1481
SCC	0.4345	0.6633	0.5240	0.6396	0.7042	0.5730	0.6828	0.6696
VIFP	0.4271	0.5435	0.4662	0.4834	0.5928	0.5133	0.5756	0.4850

Table A.2: Image 2 - London

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9806	0.9910	0.9862	0.9800	0.9937	0.9833	0.9865	0.9793
MSSSIM	0.9665	0.9820	0.9795	0.9761	0.9915	0.9772	0.9895	0.9769
PSNRB	25.0427	28.7419	27.1620	24.9654	30.3947	26.9984	28.9594	24.6500
ERGAS	8545.3060	5636.3204	6972.6243	8698.7893	4683.3123	7527.6813	5888.9409	8753.4121
RASE	1231.0257	811.0623	1002.3485	1253.4716	673.5650	1083.3214	846.0130	1260.1007
SAM	0.1966	0.1266	0.1478	0.1920	0.1026	0.1657	0.1086	0.1977
SCC	0.4210	0.6232	0.4607	0.6079	0.6690	0.4587	0.6263	0.6402
VIFP	0.3991	0.5090	0.4410	0.4543	0.5630	0.4376	0.5444	0.4490

Table A.3: Image 3 - Central Park

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9931	0.9965	0.9888	0.9930	0.9973	0.9875	0.9867	0.9936
MSSSIM	0.9861	0.9911	0.9788	0.9875	0.9951	0.9875	0.9925	0.9894
PSNRB	35.3735	39.6173	34.9554	36.8017	40.9778	35.9387	36.8445	35.9022
ERGAS	4342.3999	2948.1398	4983.7209	4110.6069	2621.6105	4531.4890	4297.0406	4054.4045
RASE	626.0818	424.3901	714.3115	592.1483	377.2822	645.4366	615.5797	583.5349
SAM	0.1131	0.0761	0.1190	0.1051	0.0652	0.0925	0.0742	0.1039
SCC	0.4325	0.6113	0.3877	0.5951	0.6324	0.4886	0.5879	0.6630
VIFP	0.5567	0.6901	0.5084	0.6474	0.7250	0.6100	0.6860	0.6469

Table A.4: Image 4 - Countryside

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9913	0.9956	0.9892	0.9906	0.9967	0.9875	0.9860	0.9905
MSSSIM	0.9795	0.9874	0.9823	0.9836	0.9932	0.9839	0.9915	0.9842
PSNRB	25.3233	28.6526	26.9295	25.8812	30.1590	27.3320	28.9858	24.5105
ERGAS	4490.2316	3105.6903	4525.5856	4468.6090	2681.8146	4380.0511	3947.0072	4662.1604
RASE	647.1001	446.9789	650.6114	643.2951	385.8298	631.4595	564.3150	671.2120
SAM	0.1471	0.1005	0.1211	0.1384	0.0842	0.1220	0.0892	0.1595
SCC	0.4155	0.5378	0.3234	0.5025	0.5411	0.3791	0.4880	0.6057
VIFP	0.4740	0.5640	0.4511	0.5131	0.6040	0.4922	0.5773	0.5078

Table A.5: Image 5 - Miami

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9971	0.9987	0.9944	0.9970	0.9989	0.9933	0.9912	0.9976
MSSSIM	0.9898	0.9939	0.9774	0.9908	0.9962	0.9898	0.9938	0.9927
PSNRB	37.5559	42.4730	36.1490	39.7216	43.5395	37.4052	37.4391	37.2015
ERGAS	2790.9666	1842.1291	3332.2995	2697.4249	1703.5892	3143.1733	3466.5030	2483.7986
RASE	402.6991	265.7635	480.7575	389.1246	245.7662	453.7227	497.4480	358.3862
SAM	0.0686	0.0452	0.0799	0.0645	0.0414	0.0630	0.0494	0.0608
SCC	0.4440	0.6331	0.4049	0.6166	0.6415	0.4706	0.5839	0.7047
VIFP	0.6065	0.7531	0.5316	0.7209	0.7703	0.6187	0.7131	0.7314

Table A.6: Image 6 - Rocky Landscape

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9869	0.9924	0.9821	0.9866	0.9942	0.9838	0.9787	0.9861
MSSSIM	0.9819	0.9882	0.9799	0.9841	0.9938	0.9856	0.9914	0.9850
PSNRB	30.8627	34.8589	31.8631	32.3102	36.5943	33.6627	34.1981	31.4402
ERGAS	6099.9323	4330.6042	6459.3125	5909.3727	3740.7051	5880.6273	5451.6181	6099.8000
RASE	872.6355	616.1607	918.3478	844.5736	531.2901	827.9621	775.0439	868.6778
SAM	0.1511	0.0962	0.1347	0.1335	0.0769	0.1047	0.0888	0.1382
SCC	0.4297	0.5891	0.3620	0.5708	0.6194	0.4713	0.5741	0.6272
VIFP	0.4934	0.6153	0.4745	0.5635	0.6660	0.5610	0.6351	0.5605

Table A.7: Image 7 - Lake

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9875	0.9938	0.9910	0.9872	0.9957	0.9916	0.9922	0.9871
MSSSIM	0.9563	0.9739	0.9712	0.9637	0.9866	0.9740	0.9848	0.9701
PSNRB	26.0083	29.1018	27.3605	26.0135	30.4420	28.3343	29.0549	25.8796
ERGAS	6912.8039	4699.2639	5681.1864	6915.4604	3947.4553	5404.7512	4677.1253	6971.3147
RASE	996.1857	677.2244	818.3867	997.3763	568.8456	778.9047	673.6191	1004.6470
SAM	0.1509	0.1045	0.1229	0.1512	0.0876	0.1162	0.0918	0.1528
SCC	0.4192	0.6474	0.5170	0.6389	0.6987	0.5449	0.6779	0.6674
VIFP	0.3873	0.4942	0.4237	0.4446	0.5469	0.4548	0.5315	0.4452

Table A.8: Image 8 - Paris

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9900	0.9951	0.9899	0.9902	0.9965	0.9914	0.9902	0.9901
MSSSIM	0.9711	0.9820	0.9754	0.9751	0.9906	0.9822	0.9887	0.9795
PSNRB	28.7362	32.1387	29.6362	29.1980	33.5489	31.3329	31.7280	29.0286
ERGAS	5796.0077	3903.7220	5479.1779	5596.9694	3309.4481	4753.3372	4409.9715	5639.2587
RASE	833.9686	561.5305	786.8188	806.1803	475.9314	679.9109	632.8703	811.0757
SAM	0.1374	0.0927	0.1194	0.1323	0.0776	0.0990	0.0824	0.1337
SCC	0.4206	0.6307	0.4420	0.6125	0.6702	0.5360	0.6406	0.6585
VIFP	0.4484	0.5661	0.4534	0.5111	0.6153	0.5293	0.5946	0.5126

Table A.9: Image 9 - Versailles

	LR	Bicubic	SRCNN	SRCNN-p	EDSR	SRGAN	ESRGAN	HAT
UQI	0.9999	0.9999	0.9996	0.9998	0.9999	0.9990	0.9989	0.9999
MSSSIM	0.9979	0.9984	0.9893	0.9970	0.9988	0.9944	0.9971	0.9982
PSNRB	42.2454	47.9105	40.1054	44.6718	48.0719	39.1476	38.4511	42.0462
ERGAS	503.9697	377.3061	894.4186	600.3008	391.4067	1250.1844	1295.1236	459.0874
RASE	72.6700	54.3802	128.7177	86.5733	56.4303	180.2369	185.8405	66.1454
SAM	0.0133	0.0090	0.0198	0.0125	0.0089	0.0207	0.0133	0.0116
SCC	0.4589	0.4694	0.2053	0.4247	0.4035	0.1973	0.2945	0.6194
VIFP	0.7831	0.8606	0.6079	0.8398	0.8365	0.6869	0.7676	0.8636

Table A.10: Image 10 - Sea

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2023-06-02

EXAMENSARBETE Enhancing Satellite Images Using Super-Resolution
STUDENT Nils Olén
HANDLEDARE Michael Doggett (LTH), Jonas Bondesson (Tactel), Tobias Leksell (Tactel)
EXAMINATOR Jacek Malec (LTH)

AI-uppskalning av satellitbilder

POPULÄRVETENSKAPLIG SAMMANFATTNING **Nils Olén**

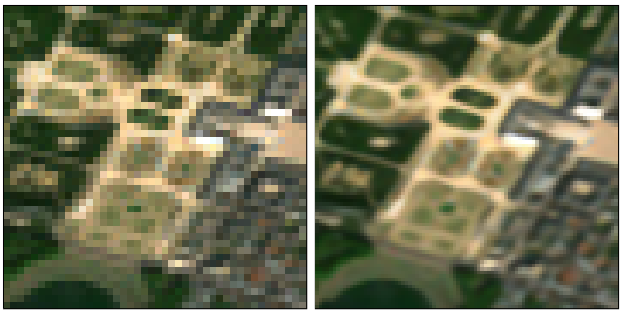
Super-resolution är en populär maskininlärningsmetod som används för att skala upp bilder till en högre upplösning. I detta arbete utvärderas flera implementationer av super-resolution efter hur bra resultat de producerar vid uppskalning av satellitbilder.

Kartapplikationer som visar satellitbilder följer ofta en så kallad Tiled web map-modell. Denna modell innebär att man har ett antal zoomnivåer, där varje ökad zoomnivå innebär att fler bilder av en bestämd konstant upplösning används för att visa kartan. Detta innebär att antalet bilder, och lagringsutrymmet som krävs för att lagra dessa bilder, ökar exponentiellt.

Karttjänsten Arc, utvecklad av Malmöbaserade Tactel, som jag har arbetat med är gjord för att visas för passagerare på flygplan, och eftersom ett flygplan har ytterligare begränsningar gällande lagringsutrymme, krävs effektiv komprimering. Kartmodellen har 13 zoomnivåer, där den 13:e kräver nästan fyra gånger mer lagringsutrymme än den föregående. Detta projekt utforskar möjligheten att lagra bilderna på den föregående zoomnivån 12 istället, och skala upp bilderna till en estimering av zoomnivå 13 när de behöver visas. För att åstadkomma detta använde jag en metod av AI-uppskalning som kallas super-resolution.

Super-resolution är en maskininlärningsteknik som innebär att man tränar ett artificiellt neuronnät med ett dataset av bildpar som består av en lågupplöst och en högupplöst version av samma bild. Målet är att nätverket ska lära sig att skala upp nya osedda bilder.

I mitt examensarbete har jag undersökt om super-resolution ger tillräckligt bra resultat för att vara ett alternativ för att slippa lagra de högupplösta bilderna på flygplanet. Därför har jag identifierat och utvärderat ett antal olika implementationer av super-resolution för att ta reda på vilken som fungerar bäst för att skala upp satellitbilder.



Figur 1: En bild på slottet i Versailles, uppskalad med EDSR-modellen.

Under testandet av implementationerna kom det fram att mer komplexa modeller har en för långsam uppskalningshastighet för att vara användbara i en kartapplikation. Efter en utvärdering. som baserades på bildkvalitet och uppskalninghastighet, ansågs modellen EDSR (Enhanced Deep Super-Resolution Network) ge bäst resultat av de implementationer jag testade. Jag kunde också konstatera att eftersom mätvärdena är baserade på vilken uppskalad bild som liknar originalbilden mest, fick vissa bilder ett sämre betyg då de var skarpare än originalbilden.