UTRECHT UNIVERSITY
Faculty of Science
Department of Information and Computing Sciences
MSc Artificial Intelligence

REMOVAL AND INPAINTING OF OBJECTS FROM STREET-VIEW SCENES USING DIFFUSION MODELS

A THESIS BY
Dimitar Milenov Angelov
2339463

Project supervisor Assist. Prof. dr. Itir Onal Ertugrul
Daily supervisor Jeroen Guelen
Daily supervisor Sara Abdulaziz
Second examiner Assoc. Prof. Dr. ir. Ronald W. Poppe

Abstract

Inpainting is the process of reconstructing missing parts of an image, with the goal of producing a convincing result. This research, done in collaboration with Cyclomedia, investigates whether latent diffusion models (Rombach et al., 2022) can be used to inpaint the missing regions after an object has been removed from a street-view image. Cyclomedia semantic object masks were refined using the SAM model (Kirillov et al., 2023) to produce high-quality and accurate object coverage for inpainting. Fine-tuning was evaluated for increasing the accuracy and quality of inpainting results. A partial loss function was proposed, implemented, and evaluated. Lastly, a feature-based measure of image complexity was used to evaluate the training data and a model was trained on a subset of the most complex training images. The evaluation process includes both computational metrics and a qualitative user study. We found that the fine-tuning process improves the generative performance of the models, but that the partial loss and data filtering techniques did not result in an improvement. We speculate on reasons why that may be the case and share recommendations for future research directions.

Table of Contents

1. Introduction

Image inpainting is a fundamental task in computer vision that involves filling in missing or damaged parts of an image. The ability to reconstruct images that are incomplete or corrupted is critical in many applications, two of which this work will focus on - object removal and image restoration. In the context of supplying street-view imagery, removing objects such as people and cars can be used as a privacy measure, in place of blurring faces and license plates. An image in which such objects are removed rather than blurred would be more natural for an end user. However, this is a difficult task, because removing these objects from an image leaves missing information, creating a hole. This is not acceptable, as it is important that the resulting image appears natural - one should not be able to tell that there ever was an object.

Traditional methods for image inpainting rely on handcrafted features and priors, which limit their ability to handle complex image semantics, making them unsuitable for such applications (Rojas et al., 2020). Generative Adversarial Networks (GANs) have been able to create realistic images, but struggle with learning the structures of complex semantically-rich datasets, such as street-view imagery. Recently, deep learning Diffusion Models (DMs) have achieved impressive results for generating natural images and complex scenes, garnering a large amount of research and media attention (Croitoru et al., 2022). They have been applied to inpainting and achieved state-of-the-art results, beating both traditional and GAN-based methods (Lugmayr et al., 2022; Rombach et al., 2022). The prospects of natural removal of privacy-sensitive information and impressive results of DMs motivate this thesis, which will be performed at Cyclomedia.

The goal is to investigate whether DMs are a suitable method for the task of object removal on Cyclomedia data. It is important for Cyclomedia that the inpainting is natural and accurate - meaning no inaccurate data is hallucinated in the inpainted areas. To this end, several research questions and associated methods are motivated and proposed.

In this introduction, a short overview of Cyclomedia, as well as the research questions of this thesis, are given. An extensive literature review of the generative modelling and image inpainting fields are presented. The proposed methodology contains an overview of the dataset which will be used, three methods to be investigated, and the evaluation strategy. The proposal ends with an overview of the plan for Phase II of the thesis.

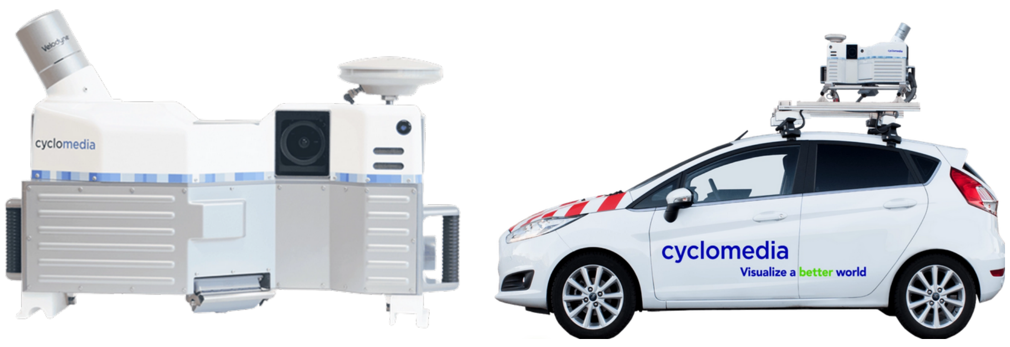


Figure 1. Cyclomedia recording setup. Left: Digital Cyclorama Recorder (DCR) system. Right: Car-mounted DCR system.

Cyclomedia is a company which specializes in providing street-view and aerial imagery as well as LIDAR point clouds. This thesis will make use of part of their street-view image dataset, which is recorded annually with high location accuracy. The company provides full coverage in the Netherlands and partial coverage in other countries, including France, Germany, the United States, and Scandinavian countries. Cyclomedia's customers include municipalities, utilities, infrastructure, and insurance companies, who leverage their data for things such as virtual inspections, inventory management, and infrastructure planning without physical site visits. The recorded data is available via an online viewer called StreetSmart, which includes various features such as distance and surface measurements on maps and image data.

The street-view data is captured using the Digital Cyclorama Recorder (DCR) system, which is mounted on top of recording cars. This is shown in Figure 1. The DCR system comprises of five cameras with their focal points on a line parallel to the driving direction, enabling the capture of a parallax-free 360° panoramic image, or Cyclorama, while driving. This patented method ensures that the 100 megapixel images are precisely aligned and captured exactly every 5 meters (Heuvel et al., 2011). While driving a Velodyne LIDAR scanner is continuously scanning the environment. The scanner is tilted backward, enabling the recording of a high-density point cloud of the entire street, including the road, adjacent areas, and higher buildings. This point cloud is used to calculate the distance for each pixel in an image. The depth can be determined by measuring the distance between the location of the camera and the intersection point of a ray from the camera location and the point cloud. This is done automatically for every image in the Cyclomedia pipeline.

As mentioned already, one of the critical aspects of Cyclomedia's data is the highly accurate positioning and calibration of the images. GPS alone would be unreliable, particularly in urban areas, where a signal might be difficult to receive. Therefore, the position of the recordings is calculated by combining several sensors, such as GPS and IMU, resulting in a relative position accuracy within 2 centimeters between Cycloramas. In addition, a dataset of images annotated with semantic maps, created by professional human labelers is available. Both the semantic-labelled data and location information will be leveraged during this thesis by the methods in this thesis. This is covered in more details in Section 3.

1.1 Research Questions

Main research question: How effective are latent diffusion models at inpainting natural street-view scenes?

Subquestions:

1. What is the impact of fine-tuning on the generative performance of the model in inpainting street-view scenes?
2. How does a loss function which focuses on specific parts of the image influence the generative performance of the model in inpainting street-view scenes?
3. How does the intricacy and complexity of the data used to fine-tune the data affect the generative performance in inpainting street-view scenes?
4. Is there a significant user preference when comparing the generative performance of the fine-tuned model against the baseline for inpainting street-view scenes?

2. Literature Review

2.1 Generative Models

Generative models are a type of machine learning model that focus on estimating the underlying probability distribution that generates the input data. Unlike discriminative models (such as logistic regression, SVM, and deep neural networks), which estimate the probability of a target variable Y given an input X ($P(X|Y)$) by learning the distinguishing characteristics of Y, generative models learn the entire distribution of $P(Y|X)$. The Naïve Bayes classifier (Webb, 2010) is the simplest example of an algorithm which uses $P(Y|X)$ and $P(X)$ and through Bayes' rule calculates $P(X|Y)$, and is considered a generative model. Usually calculating $P(Y|X)$ directly is not possible, therefore a generative model is used to estimate the distribution. Given that a classifier has estimated such a distribution correctly, one can sample from $P(X|Y)$ arbitrarily to create realistic (fit to the data distribution) data samples X.

Training a generative model takes substantially longer than a discriminative model because a much higher number of correlations need to be learned to recreate the entire probability distribution versus learning to discriminate between samples lying on the distribution. For instance, a convolutional neural network (CNN) (Venkatesan and Li, 2017) classifier can distinguish between images of cats and dogs by identifying only a few significant differences between them, while a deep convolutional generative adversarial network (DCGAN) (Radford et al., 2016), on the other hand, must learn all features of cats and dogs to generate realistic images of them. Discriminative models define a decision boundary in the data space, while generative models must comprehend the entire distribution of the data – a fundamentally more complex task, as pointed out in Harshvardhan et al. (2020).

Since they are based on such a powerful idea, it should be no surprise that generative models have found a variety of applications in a range of fields, from computer vision, to language processing, music, drug discovery, and more Harshvardhan et al. (2020). This section of the review will focus on generation specifically within the domain of computer vision, going over the ideas, applications, and architectures behind three of the main families of models – Autoencoders, Generative Adversarial Networks, and Diffusion Models. The intention is to provide readers with the necessary background information on the field of image generation, building up from techniques proposed over four decades ago to the state-of-the-art techniques disrupting the art world today.

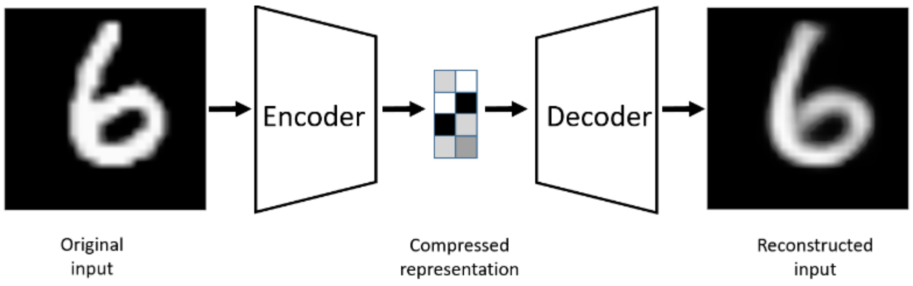


Figure 2. Example of an autoencoder architecture. The input image is encoded to a compressed representation and then decoded. Courtesy Bank et al. (2021)

2.1.1 Autoencoders

Autoencoding models, initially introduced in Rumelhart et al. (1986), are a type of unsupervised learning technique that aim to learn compact representations of data by reconstructing the input data through an encoding-decoding process. The goal is to learn a function that maps the input data to a lower-dimensional latent space (also called the "hidden" space) and a second function which maps the latent representation back to the original data space. Those functions are typically called an encoder and decoder, respectively, and are each learned via two independent neural networks. See Figure 2 for a high-level illustration of an autoencoder model.

The encoding process can be thought of as a compression step, where the input data is transformed into a more concise form while still capturing the most important aspects of the raw data. The latent representation is typically much smaller in size than the original input data, making it significantly more efficient to store and process. This restriction is called a bottleneck. It is also a form of regularization, which increases the generalization power of the learned model at the cost of accuracy. If a model were to not be regularized in some way, the encoder and decoder would simply learn the identity function between the input data (x) and the reconstructed data (x'). This is an important point, and many other regularization techniques have been proposed in subsequent research. The relevant techniques will be explored in this section.

The decoding process can be thought of as a reconstruction step, where the latent representation is transformed back into the original data space. As such, the input is a learned latent representation of the input data and the output is the reconstructed data x'. The networks are trained to minimise a reconstruction loss function, which measures the difference between x and x'. This is further discussed in 2.1.1.

A key advantage of autoencoders compared to supervised learning models is that autoencoders do not need labelled data. This means that one can use natural, unlabelled image data from the internet for training, without needing to spend person-hours labelling it beforehand.

Learning a latent representation of data is a powerful idea, which can be applied to a variety of problems. It is therefore not surprising that autoencoders have been applied to a range of applications beyond image generation, including (but not limited to) classification, clustering, anomaly detection, recommendation systems, and dimensionality reduction (Bank et al., 2021).

Reconstruction Loss Functions The **Mean Squared Error** (MSE) is calculated by taking the sum of the squares of the differences between the reconstructed data (x') and the original input data (x), and dividing by the number of data points (N).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_i')^2 \tag{2.1}$$

The MSE is differentiable and has nice mathematical properties, such as being convex. However, it can be sensitive to outliers, as large errors can have a significant impact on the loss value.

The **Binary Cross-Entropy** (BCE) is calculated by taking the negative sum of the product of the original input data and the log of the reconstructed data, and the product of the complement of the original input data and the log of the complement of the reconstructed data.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}(x_i * \log(x_i') + (1 - x_i) * \log(1 - x_i')) \tag{2.2}$$

$$BCE = \begin{cases} -log(f(s_1)) & if \quad t_1 = 1 \\ -log(1 - f(s_1)) & if \quad t_1 = 0 \end{cases} \tag{2.3}$$

Regularization **L1** regularization, also known as Lasso regularization, encourages the weights to be sparse, meaning that some of them will be set to zero, resulting in a more interpretable model with fewer features. This is done by adding a penalty term to the loss function proportional to the absolute values of the weights:

$$L_{\text{L1}} = L + \lambda \sum_{i=1}^{n} |w_i| \tag{2.4}$$

where L is the original loss function, w_i are the weights of the model, n is the number of weights, and λ is the regularization strength hyperparameter.

L2 regularization, also known as Ridge regularization, encourages the weights to be small, resulting in a smoother and more generalized model. This is done by the addition of a penalty term to the loss function proportional to the squared values of the weights:

$$L_{\text{L2}} = L + \lambda \sum_{i=1}^{n} w_i^2 \tag{2.5}$$

L2 regularization is computationally more efficient than L1 regularization, as it has a closed-form solution.

A combination of L1 and L2 regularization can be used simultaneously by adding both penalty terms to the loss function:

$$L_{\text{L1+L2}} = L + \lambda_1 \sum_{i=1}^{n} |w_i| + \lambda_2 \sum_{i=1}^{n} w_i^2 \tag{2.6}$$

where λ_1 and λ_2 are the regularization strength hyperparameters for L1 and L2 regularization, respectively. This encourages a balance between bias and variance.

The bias-variance trade-off is an important trade-off in the design of autoencoders. On one hand, the architecture should effectively reconstruct the input with minimal error. On the other hand, it should also produce a low-dimensional representation that can be generalized to other inputs effectively. Various methods and architectures have been proposed which attempt to balance these competing goals, as well as improve the models' ability to learn rich representations and capture important information.

Sparse Autoencoders (Ng, 2011) are a method of introducing a bottleneck in the information flow in a neural network without reducing the number of dimensions of the latent space. Instead, the loss function is modified via a sparsity penalty term, which discourages the network from activating too many hidden units at the same time. This has the effect of forcing the network to use only a portion of the available neurons to encode and decode a given input. This is a different approach to regularization, which typically involves regularizing the weights of a network, not the activations.

As a result, the trained network will focus individual nodes in the hidden layer to specific attributes of the input data. Since only a portion of the available neurons in the hidden layer are used, a limitation is imposed on the network's ability to memorize the training data, while not limiting its ability to extract features from it. Additionally, the sparsity constraint separates the size of the latent state representation and the regularization of the network. This allows the dimensionality of the encoding to be chosen based on the context of the data, without running into issues due to under- or over-regularization.

There are two main sparsity terms which can be added to the loss function: L1 regularization (2.4) and KL-divergence. L1 regularization penalizes the absolute value of the activations, while KL-divergence compares the observed activations to an ideal probability distribution and penalizes the network when the activations deviate from this distribution.

The KL-divergence is a function which measures the difference between two probability distributions. In order to use it as a sparsity constraint, a sparsity parameter ρ is defined, which denotes the expected average activation of a neuron over a subset of the dataset (M). In essence, by constraining the average activation of a neuron over a collection of samples neurons are encouraged to fire only for a subset of the observations.

$$\rho_j = \frac{1}{M} \sum_{i=1}^{m} [a_i^{(h)}(x)], \tag{2.7}$$

where the subscript j denotes the specific neuron in layer h, summing the activations over M.

As part of our loss function, the KL-divergence ($KL(||)$) is used to compare the observed ρ to a Bernoulli random variable distribution of ρ. Usually, the value of ρ is kept close to 0, e.g. 0.05, with the goal of keeping the average activation of a neuron close to 0. A high value for $KL(||)$ would mean a large difference between the distribution of ρ over M and an ideal distribution (Ng, 2011).

The KL-divergence and a reconstruction loss a KL-divergence regularization term have the following form:

$$KL = \sum_{j=1}^{l(h)} \rho \log \frac{\rho}{\rho^j} + (1-\rho) \log \frac{1-\rho}{1-\rho^j} \tag{2.8}$$

$$Loss = L(x, x') + \sum_{j=1}^{n} KL(\rho||\rho^j) \tag{2.9}$$

Denoising Autoencoder A limitation of AE models is their sensitivity to noise in the input data, which leads to an inability to reconstruct the original signal accurately when presented with a noisy input. Denoising autoencoder (DAE) models, proposed in Pascal et al. (2008), aim to solve this by adding noise to the input signal during training called the "stochastic corruption process" (See Fig. 3). With this approach, it is impossible for DAEs to learn a direct mapping by memorizing the training data, as the inputs the model is trained on and the outputs it is trying to reconstruct are not the same. Additionally, DAEs are relatively easy to train and not significantly more computationally expensive as the only addition is the stochastic corruption process, which is computationally inexpensive.

The stochastic corruption process works by randomly setting a certain percentage of the input elements to zero, or to a random value. The percentage of elements that are corrupted is controlled via a corruption level hyper-parameter (c). This process is done independently for each input element, and is applied to the input signal during training. For example, for an image dataset, the stochastic corruption process would add noise to the image by randomly setting a certain percentage of the pixels to zero or to a random value. This process can be done in a variety of ways, such as masking out pixels in a random rectangular region or masking out pixels randomly across the entire image. Pascal et al. experimentally found that a corruption level of around 50% ($c = 0.5$) gave good results.

An algorithmic formulation of the stochastic corruption process could be:

- Let x be the original input vector and c the corruption level (a scalar between 0 and 1)
- For each element x_i in x:
  - With probability c, set x_i to 0 (or, alternatively, a random value).
  - With probability $1 - c$, keep x_i unchanged.

DAEs have been applied in a wide range of applications such as image denoising, audio denoising, text denoising, and anomaly detection (Patil and P.m, 2020).

Variational Autoencoders (VAEs, Kingma and Welling (2013)) are a class of generative models that have gained significant attention in recent years for their ability to learn complex distributions and generate new samples from them. VAEs differ from traditional

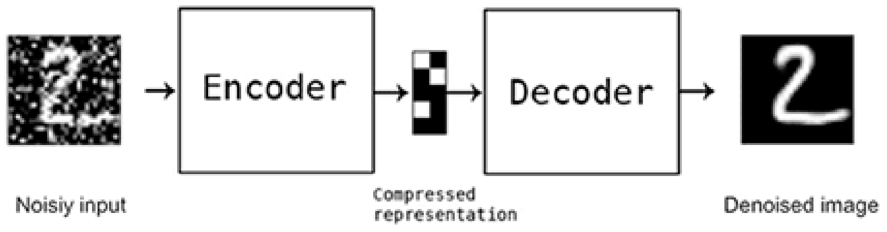


Figure 3. Denoising Autoencoder architecture utilizing stochastic corruption process for unsupervised feature learning and noise reduction in images. The stochastic corruption algorithm randomly applies noise to the input data, encouraging the autoencoder to learn robust features that are less sensitive to small perturbations in the data. Courtesy Chollet (2016).

Autoencoders (AEs) in that they introduce a probabilistic approach to the encoding and decoding process, which allows for the generation of new samples that are similar to the training data but not identical.

VAEs were first proposed as a way to improve upon traditional AEs. In traditional AEs, the encoder and decoder are trained to minimize the reconstruction error between the input and the output. However, this approach has several limitations, including the inability to generate new samples from the learned distribution and the difficulty of training the model when the data is highly correlated. VAEs address these limitations by introducing a probabilistic approach to the encoding and decoding process.

The basic idea behind VAEs is to introduce a latent variable z, which is assumed to be a random variable with a prior distribution $p(z)$. The encoder, or recognition model, is trained to approximate the posterior distribution $p(z|x)$ for a given input x. The decoder, or generative model, is trained to reconstruct the input x from the latent variable z. The main advantage of this approach is that it allows for the generation of new samples by sampling from the prior distribution $p(z)$ and passing them through the decoder.

The training process of VAEs is based on the optimization of the lower variational bound, which is a lower bound on the log-likelihood of the data. The lower variational bound is defined as:

$$\log p(x) \geq \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathrm{KL}(q(z|x)||p(z)) \tag{2.10}$$

$$Loss = L(x, x') + \sum_{j=1}^{n} KL(q_j(z|x)||p(z)) \tag{2.11}$$

where $p(x)$ is the true data distribution, $q(z|x)$ is the approximated posterior distribution, $p(x|z)$ is the likelihood, and $KL(q(z|x)||p(z))$ is the KL divergence between the approximated posterior and the prior distribution.

The lower variational bound can be seen as a trade-off between the reconstruction error and the regularization term. The first term, $\mathbb{E}_{q(z|x)}[\log p(x|z)]$, represents the reconstruction error, and the second term, KL$(q(z|x)||p(z))$, represents the regularization term. The lower variational bound is optimized by maximizing the likelihood of the data while keeping the regularization term small.

Several methods have been proposed to improve the performance of VAEs. One of the most popular methods is the use of deep neural networks as the encoder and decoder. This approach, known as the deep VAE (DVAE), has been shown to be effective in learning complex distributions and generating high-quality samples (Rezende et al., 2014). Another popular method is the use of adversarial training, where the decoder is trained to generate samples that are indistinguishable from the real data, while the encoder is trained to discriminate between real and generated samples. This approach, known as the adversarial VAE (AVAE), has been shown to improve the quality of the generated samples and the stability of the training process (Makhzani et al., 2016).

Another important aspect of VAEs is the choice of the prior distribution p(z). The most commonly used prior distribution is the standard normal distribution, but other distributions have also been proposed, such as the Laplace distribution (Rezende et al., 2014) and the mixture of Gaussians (Tomczak and Welling, 2017). These alternative prior distributions have been shown to improve the quality of the generated samples and the ability of the model to capture complex distributions.

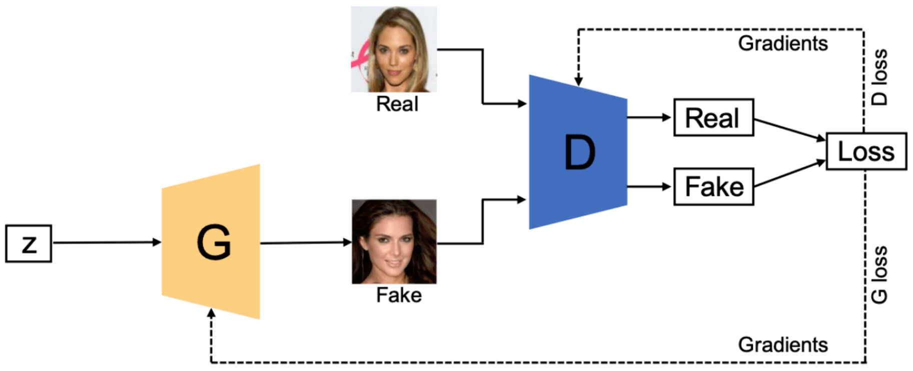


Figure 4. Generative Adversarial Network (GAN) architecture, consisting of a generator (G) and a discriminator (D) network. The two networks are trained synchronously in an adversarial manner, where the generator aims to fool the discriminator by generating realistic synthetic samples, while the discriminator tries to correctly identify the real and fake samples. Courtesy Wang et al. (2021).

2.1.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs), introduced in Goodfellow et al. (2014), are deep neural network architectures consisting of two main components. A generator network (G) and a discriminator network (D). The generator learns to generate new examples from a random noise input, while the discriminator learns to distinguish between real data and data generated by the generator network. The two networks are trained adversarially, with the generator receiving feedback from the discriminator on its ability to outperform it by generating high quality images. The architecture is shown in Figure 4.

The adversarial training aims to maximize the ability of the generator to produce realistic data, and to minimize the ability of the discriminator to distinguish between real and generated data. This idea is encapsulated in the loss function, called adversarial loss.

$$\min_G \max_D \mathbb{E}_{x\sim p_r} \log[D(x)] + \mathbb{E}_{z\sim p_z} \log[1 - D(G(z))] \tag{2.12}$$

Where G is the generator network, D is the discriminator network, x is a sample of the real data, and z is noise.

Since the time GAN architecture was published by Goodfellow et al., it has gained more attention and became a standard architecture that is being used in many applications for image generation and restoration. Figure 5 shows a road-map overview of the research done on GANs and the many variants which have been proposed since the original Goodfellow et al. paper. This large interest is due to the fact they learn in an unsupervised fashion and, perhaps most importantly, their power to generate high-quality images. They also have three major advantages to prior generative models, specifically VAEs, as highlighted in Wang et al. (2021):

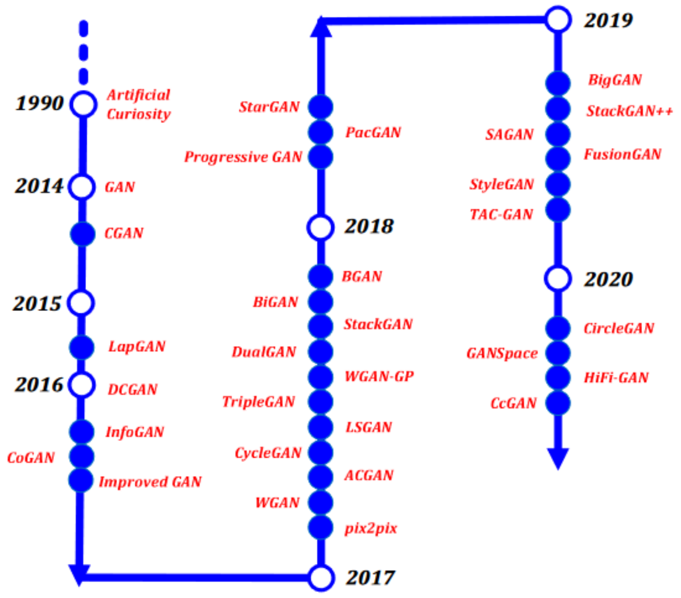


Figure 5. A visual timeline of the research on Generative Adversarial Networks (GANs) and its variants, showcasing the evolution of GANs since their inception in 2014. The graphic highlights the different types of GANs proposed in literature, including Deep Convolutional GANs (DCGAN), Wasserstein GANs (WGAN), Progressively Growing GANs (PGGAN), and StyleGAN, among others. Courtesy Farajzadeh-Zanjani et al. (2022).

1. GANs are able to produce any type of probability distribution, unlike VAEs which are mainly driven to approximate the data distribution to a unit Gaussian prior that limits their ability to learn complex features effectively (Goodfellow, 2017).
2. The GAN framework can train any type of generator network.
3. There is no restriction on the size of the latent variable.

In spite of their high capabilities, GANs have several limitations. One of the most significant is that they are difficult to train and evaluate. In terms of training, it can be challenging for the discriminator and generator to reach a Nash equilibrium, and it is common for the generator to fail to capture the full distribution of the dataset, getting stuck in a suboptimal state, and resulting in an issue known as "mode collapse" (Arjovsky and Bottou, 2017). This results in the generated samples being limited to a specific subset of the data, rather than having a diverse range of samples that cover the entire data distribution. In terms of evaluation, the primary issue is determining the dissimilarity between the real distribution of the target data (p_r) and the generated distribution (p_g). As explained in 2.1.1, it is not possible to precisely estimate the real distribution, making it challenging to accurately compare it to the generated distribution.

Apart from image generation, GAN-based models have been applied to various other computer vision sub-fields such as image-to-image translation, image super-resolution, and image completion. They have also been applied to different domains, such as natural language processing, time series synthesis, semantic segmentation, and others (Wang et al., 2021).

This rest of this section will cover some of the most significant GAN variants relevant to this work, introducing the idea of each one and highlighting their advances and limitations.

Fully-connected GAN (FCGAN) In the work which proposed the GAN architecture, Goodfellow et al. (2014), the GAN uses fully-connected neural networks for both the generator and discriminator, and was applied to the relatively simple image datasets MNIST (Li Deng, 2012), CIFAR-10 (Krizhevsky and Hinton, 2009), and Toronto Faces.

Goodfellow et al. suggest a training strategy of k steps optimizing the discriminator and one step optimizing the generator to prevent overfitting of D. It was found that using the adversarial loss (2.12) as the loss function results in an extremely low loss signal (known as vanishing gradients) for G, and instead Goodfellow et al. maximize $\log D(G(z))$ for training G. The architecture uses a maxout activation for the discriminator and a mixture of ReLU and sigmoid activations for the generator. It is a very influential paper, but the results of the final model are limited and it does not generalize well to complex images.

Deep Convolutional GAN (DCGAN, Radford et al. (2016)) is a foundational support for GAN research and is considered a major milestone in the history of GANs. DCGAN builds upon the original GAN architecture by using deep convolutional networks, rather than fully connected networks, in both the generator and discriminator. The architecture in Radford et al. (2016) is made up from four convolution layers in both the generator and discriminator and notably uses strided convolutional and transpose convolutional (also called fractional or de-convolutional) layers (Zeiler and Fergus, 2013) in each, respectively. See the architecture of the generator network in Figure 6 - the inverse is used for the discriminator.

DCGAN also made several modifications to the GAN architecture with the aim of stabilizing the training process. These modifications include the use of batch normalization (to center the generated samples and real samples at zero), and different activation functions in the networks. In G, ReLU activation is used in the hidden layers and Tanh in the output, and in D leaky ReLU is used (to prevent vanishing gradients when the activation input is smaller than 0). The same adversarial-based loss function as the GAN proposed in Goodfellow et al. (2014) (2.12) is used.

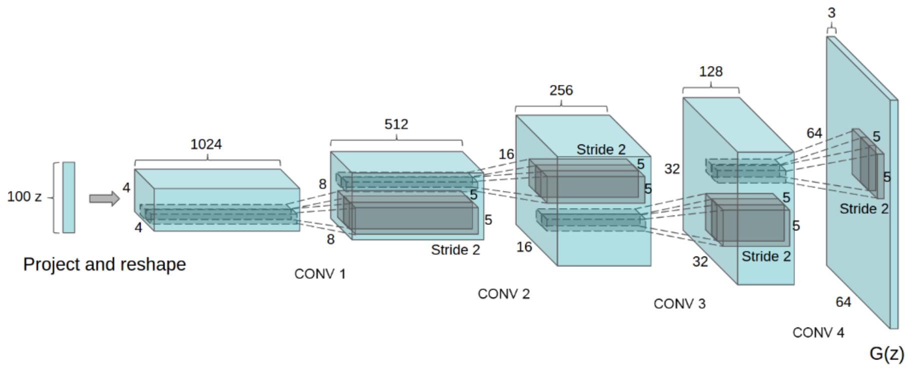


Figure 6. Deep Convolutional GAN (DCGAN) generator architecture, consisting of four transpose convolutional layers which gradually increase the image size from a latent representation to a 64^2 pixel image. No fully connected or pooling layers are used. Courtesy Radford et al. (2016).

The training process was done using stochastic gradient descent with a mini-batch size of 128 and the Adam optimizer with a learning rate of 0.0002 and momentum term of 0.5. It was trained on the 64x64 images from the Large-scale Scene Understanding (LSUN) (Yu et al., 2016), ImageNet (Deng et al., 2009) and the customized-assembled face datasets.

While DCGAN has been successful in producing realistic low-resolution images, it is limited in its capacity to generate diverse and high-resolution images. However, it remains an important milestone in the field of GAN research and the transpose convolutional architecture used in the generator has become widely adopted in GAN research (Wang et al., 2021).

Wasserstein GAN (WGAN) The Wasserstein Generative Adversarial Network (WGAN) (Arjovsky et al., 2017) uses a structure almost identical to the standard Generative Adversarial Network (GAN). However the loss function is changed to the Wasserstein distance, defined in the same work, in an effort to overcome the problem of mode collapse. The Wasserstein distance, also known as Earth-Mover's (EM) distance, is defined as

$$W(P_r, P_g) = \inf_{\gamma \in \prod(P_r, P_g)} E_{(x,y)\sim\gamma}[||x - y||] \tag{2.13}$$

where $\prod(P_r, P_g)$ refers to a collection of all mutual proportions, and P_r and P_g, in the range of $\gamma(x, y)$, are the real and generated data respectively. Since it is a comparison between two distributions, one may recall the KL-divergence (see 2.8). Compared to it, the Wasserstein distance is able to calculate a distance even when the two distributions do not overlap, and is continuous, making it ideal for providing a meaningful gradient for training

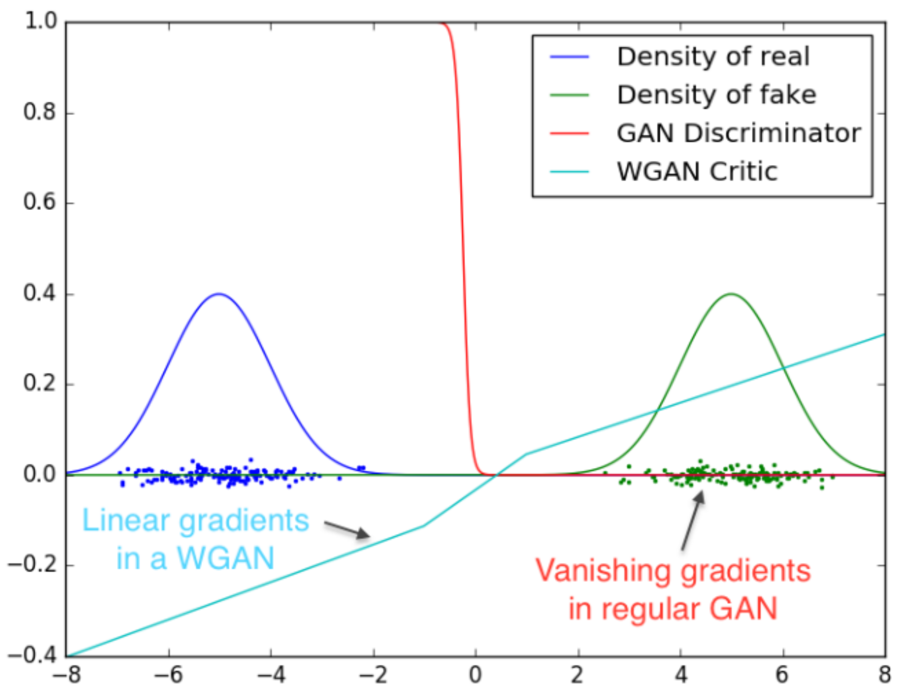


Figure 7. A comparison of the gradients provided by a typical Generative Adversarial Network (GAN) discriminator and a Wasserstein GAN (WGAN) critic, which highlights the advantage of WGANs over traditional GANs. The figure illustrates that the gradients provided by the GAN discriminator tend to saturate, leading to vanishing gradients and slow training. On the other hand, the WGAN critic provides very clean gradients on all parts of the space, enabling faster and more stable training. The WGAN architecture is designed to optimize the Wasserstein distance between the real and fake distributions, rather than the Jensen-Shannon divergence, as in the case of GANs. This results in more stable training and higher quality generated samples, making WGANs a popular choice for generative modeling tasks. Courtesy Arjovsky et al. (2017)

the generator. Figure 7 illustrates the gradient of WGAN comparing to the original GAN.

The $\inf$ in equation 2.13 is intractable, but Arjovsky et al. demonstrates that the Wasserstein distance can be estimated by D as

$$\max_{w \sim W} E_{x \sim P_r}[f_w(x)] - E_{z \sim p_z}[f_w(G(z))] \tag{2.14}$$

where f_w can be realized by D, w is the parameters in the discriminator and z is the input noise for the generator. The discriminator aims to maximize this equation in order to make the optimization distance equivalent to the Wasserstein distance. So the loss for G is

$$-\min_{G} E_{z \sim p_z}[f_w(G(z))] \tag{2.15}$$

This highlights the second architectural difference between WGAN and the original GAN - the function of D. The D in the original work is used as a binary classifier, but the function of D in WGAN is to estimate the Wasserstein distance, which is a task with a continuous output. Thus, the sigmoid in the last layer of D is removed in WGAN architectures.

WGAN was shown to be more stable during training (on the LSUN 64x64 image dataset (Yu et al., 2016)) and to produce better results in terms of mode collapse problem. However, the training was still found to be unstable at times when using momentum based optimizers such as Adam, thus RMSProp (Hinton et al., 2012) is used for training. A very deep WGAN does not converge easily.

Several subsequent variations of the WGAN architecture have been proposed in the literature, such as Wasserstein Generative Well-intentioned Network (GWIN) (Cosentino and Zhu, 2019), GS-WGAN (Chen et al., 2021), and Banach WGAN (Adler and Lunz, 2018). Gulrajani et al. (2017) propose a technique to improve the training procedure. These works have further improved the performance of WGAN and have subsequently been applied to various tasks such as image synthesis, text generation, and privacy-preserving data generation (Farajzadeh-Zanjani et al., 2022).

Progressive Growing GAN (PGGAN) Progressive GAN (PGGAN) (Kim et al., 2021) is a generative adversarial network that modifies the traditional GAN structure and training methodology in order to reduce training time, increase the variation of generated images, and stabilize the training process for generating large high-quality images. PGGAN achieves this by progressively expanding the depth of G and D in synchrony, adding layers step-wise during the training process (see Fig. 8). The progressive nature of the training enables the networks to first learn the large-scale structure of the image and then precisely learn the accurate scale details, rather than learning all scales of images at the same time.

PGGAN uses the idea of progressive neural networks, which allows for the deployment of prior knowledge via lateral connections to previously learned features and does not suffer from forgetting, making it widely applied for learning complex task sequences. The training process starts with low resolution 4x4 pixel images, and the size of both G and D is increased over the training process. All variables remain trainable throughout this growing process, which enables substantially more stable learning for both networks. By gradually increasing the resolution, the networks are faced with a much simpler task that increases in complexity over time. This approach gradually leads to the ultimate goal of discovering a mapping from latent vectors to high-resolution images.

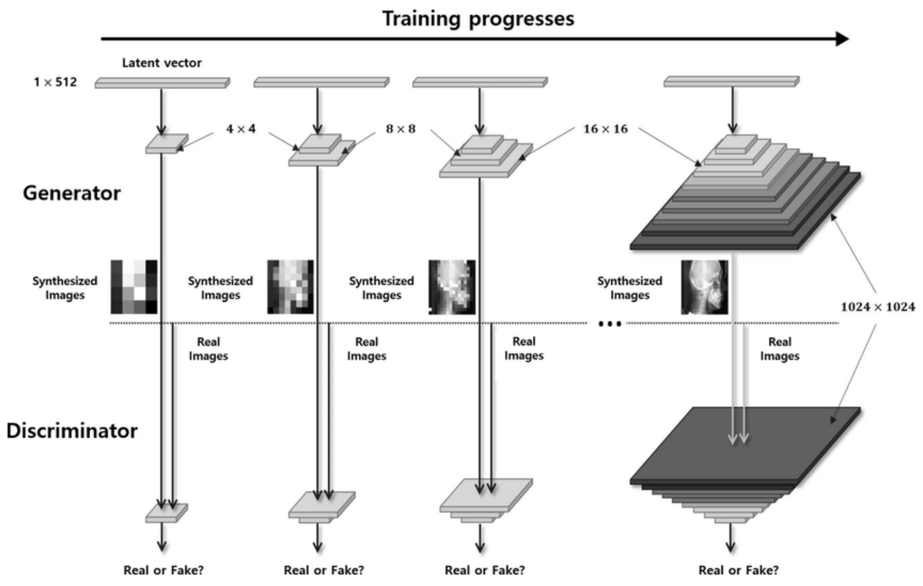


Figure 8. A visualization of the Progressively Growing GAN (PGGAN) architecture, which works by gradually increasing the size of the networks during the training process. The architecture starts with low-resolution 4^2 pixel images and increases the size of both the generator (G) and discriminator (D) over time, while keeping all variables trainable. This allows for more stable learning as the networks are gradually faced with a more complex task, leading to the ultimate goal of generating high-resolution images from latent vectors. Courtesy Kim et al. (2021)

The PGGAN architecture uses Leaky ReLU with leakiness 0.2 for all layers of both D and G except the last layers, which use linear activation. Only pixelwise normalization of the feature vectors after each convolutional 3x3 layer in the generator is deployed, i.e., no batch normalization, layer normalization, or weight normalization in either network. The Adam optimizer is utilized for training D and G and the mini-batch size is gradually decreased as the network size increases in order to deal with limitations on memory capacity. The WGAN-GP (variant of Wasserstein distance (Gulrajani et al., 2017)) loss is used.

PGGAN has been trained on several datasets such as CIFAR-10 (32x32 pixel images) (Krizhevsky and Hinton, 2009), LSUN (256x256 pixel images) (Yu et al., 2016), and CelebA-HQ (1024x1024 pixel images) (Liu et al., 2015). PGGAN has been able to generate large high-quality images, reduce training time, and solve GAN instability training problems.

StyleGAN StyleGAN (Karras et al., 2019) is a style-based generative adversarial network that utilizes the idea of progressive training and a redesigned generator structure to achieve stylistic control over the generated images. The generator network maps the input latent code into an intermediate latent space that is disentangled, allowing for regulation of the style of the generated images through the use of a mapping network and noise layers. An improved version of StyleGAN, known as StyleGAN2 (Karras et al., 2020), was later proposed and achieved better results in image quality, efficiency, diversity, and disentanglement.

One of the major contributions of StyleGAN is its ability to control specific features of the generated images, such as pose, hair, and facial features, without compromising overall image quality. This is in contrast to earlier GANs such as PGGAN, which had limited control over specific features. The input features are divided into three types: (i) coarse features - pose, hair, face, shape; (ii) medium features - facial features, eyes; (iii) fine features - color scheme.

However, StyleGAN does have some characteristic artifacts, such as blob-shaped artifacts and phase artifacts, which can be attributed to the use of instance layer normalization and the progressive growing phenomena. Despite these limitations, StyleGAN and its improved version StyleGAN2 have made significant contributions to the field of image synthesis, and have been widely used and researched over the last few years.

2.1.3 Diffusion Models

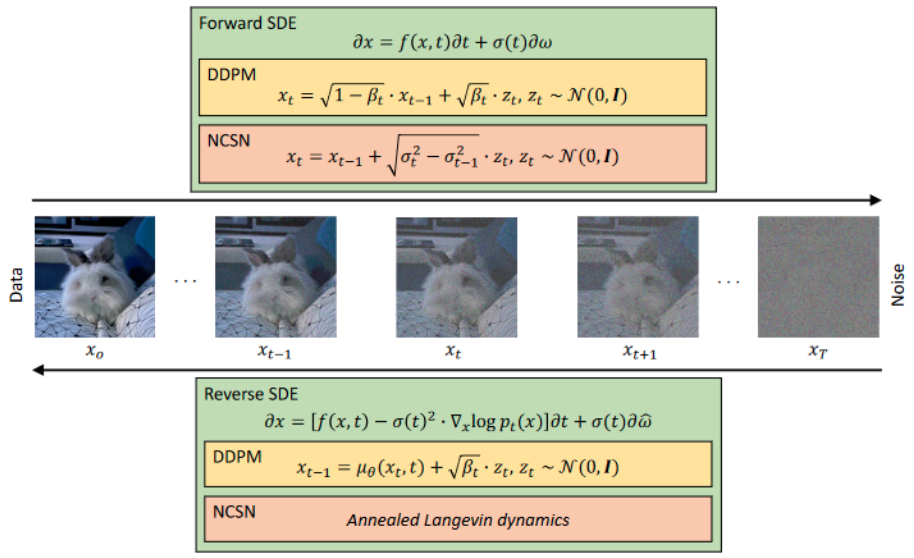
Over the past ten years, generative adversarial networks (GANs) have been the focus of a lot research effort for their ability to generate high-quality novel data. However, recent advancements have led to the establishment of the diffusion model (DM) architecture as the new state-of-the-art in deep generative models. Diffusion models have been able to generate higher quality data compared to GANs, while also having more stable training and not compromising on the diversity of the generated samples. In addition, DMs do not suffer from mode collapse, convergence failure, and the overhead of adversarial learning – all of which are known limitations of GANs.

Due to their remarkable generative abilities, there has been an increasing research interest in both improving DM architectures and applying DMs to a variety of tasks in computer vision – image generation, image super-resolution, image editing, image-to-image translation, and, notably for this work, image inpainting (Croitoru et al., 2022; Cao et al., 2022).

Diffusion models utilize a two-stage approach. In the first stage, called the forward diffusion stage, Gaussian noise is added to the input data over T steps. The training data is gradually degraded until it becomes pure Gaussian noise. In the second stage, known as the reverse/backward diffusion stage, a generative model is trained to gradually undo the diffusion process and recover the original input data from the diffused (noisy) data, one step t at a time. The generative model estimates the noise that should be subtracted at each step and is typically based on a U-Net architecture (Ronneberger et al., 2015) to allow for the preservation of the dimensionality of the data at each step.

It should be noted that there are at least three sub-categories of diffusion models, shown in Figure 9 (Croitoru et al., 2022). The first sub-category consists of denoising diffusion probabilistic models (DDPMs) (Sohl-Dickstein et al., 2015), which draw inspiration from non-equilibrium thermodynamics theory. DDPMs are latent variable models that use latent variables to estimate the probability distribution. DDPMs can be seen as a variety of Variational Autoencoders (VAEs), where the encoding process inside VAE corresponds to the forward diffusion stage, and the decoding process corresponds to the reverse diffusion stage. This review will focus on the DDPM variant, as it is the most relevant to prior works related to inpainting. The second sub-category is noise conditioned score networks (NCSNs) (Song and Ermon, 2020), which train a shared neural network via score matching to estimate the score function (defined as the gradient of the log density) of the perturbed data distribution at different noise levels.

An alternative approach to modeling diffusion is through stochastic differential equations (SDEs, Song et al. (2021)), which form the third sub-category of diffusion models. In



Forward SDE
$\partial x = f(x,t)\partial t + \sigma(t)\partial\omega$
DDPM
$x_t = \sqrt{1-\beta_t}\cdot x_{t-1} + \sqrt{\beta_t}\cdot z_t,\ z_t \sim \mathcal{N}(0,\mathbf{I})$
NCSN
$x_t = x_{t-1} + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}\cdot z_t,\ z_t \sim \mathcal{N}(0,\mathbf{I})$
Data
x_o
x_{t-1}
x_t
x_{t+1}
x_T
Noise
Reverse SDE
$\partial x = [f(x,t) - \sigma(t)^2\cdot\nabla_x \log p_t(x)]\partial t + \sigma(t)\partial\hat{\omega}$
DDPM
$x_{t-1} = \mu_\theta(x_t,t) + \sqrt{\beta_t}\cdot z_t,\ z_t \sim \mathcal{N}(0,\mathbf{I})$
NCSN
Annealed Langevin dynamics

Figure 9. A generic framework showing the three formulations of diffusion models based on: stochastic differential equations (SDEs), denoising diffusion probabilistic models (DDPMs) and noise conditioned score networks (NCSNs). The SDE formulation is a generalization of the other two. In general, a diffusion model consists of two processes. The first one, called the forward process, transforms data into noise, while the second one is a generative process that reverses the effect of the forward process. This latter process learns to transform the noise back into data. Courtesy Croitoru et al. (2022).

Croitoru et al. (2022), this is shown as a generalization of the other two variants. The forward SDE process, shown in the top of Figure 9, shows that a change over time in x is modeled by a function f plus a stochastic component $\partial\omega \sim N(0, \partial t)$ scaled by $\sigma(t)$. Different choices of f and σ will lead to different diffusion processes, hence why the SDE formulation is a generalization of DDPMs and NCSNs. Modeling diffusion using forward and reverse SDEs leads to efficient generation strategies and robust theoretical results (Croitoru et al., 2022).

Denoising Diffusion Probabilistic Models (DDPMs) In this section follows an explanation of the formulation of DDPMs, divided into the forward and reverse diffusion processes. The DDPM architecture pipeline is shown in Figure 10 The formulation and explanations are based on Croitoru et al. (2022); Cao et al. (2022); Weng (2021)

In the **forward process**, Gaussian noise is added to an image x over T steps until the data is corrupted into pure noise. x_t is the image x at time-step $t \in T$. In order to obtain noised versions of x_0 $x_1, x_2, ..., x_T$, a Markovian process is formulated:

$$p\left(x_t \mid x_{t-1}\right) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\cdot x_{t-1}, \beta_t \cdot \mathbf{I}\right), \forall t \in \{1, \ldots, T\}, \qquad (2.16)$$

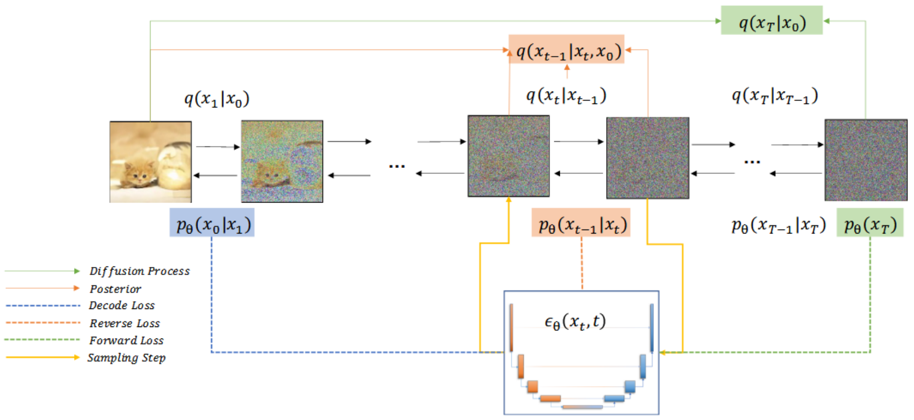


Figure 10. The arrows pointing from left to right indicate the diffusion process and the arrows pointing in the reverse direction indicate the reverse process. The colored background transition terms are components of ELBO: the blue part stands for decode loss L_0, the green part represents forward loss L_T, and the orange part constitutes the reverse loss L_t. Dashed lines with different colors show the training pattern of the noise prediction model ϵ_θ. Besides, in any step $1 \leq t \leq T$, the yellow lines denote the ancestral sampling process. Courtesy Cao et al. (2022).

where $\beta_1, ..., \beta_T \in [0, 1)$ are hyperparameters controlling the amount of noise added at each step, $\mathbf{I}$ is the identity matrix with the same dimensions at x, and $\mathcal{N}(x; \mu, \sigma)$ is the normal distribution with mean μ and covariance σ which produces x.

This recursive formulation allows for sampling x_t directly via:

$$p\left(x_t \mid x_0\right)=\mathcal{N}\left(x_t ; \sqrt{\hat{\beta}_t} \cdot x_0,\left(1-\hat{\beta}_t\right) \cdot \mathbf{I}\right), \tag{2.17}$$

where $\hat{\beta}_t=\prod_{i=1}^t \alpha_i$ and $\alpha_t=1-\beta_t$. In this way, given x_0 and a variance schedule β_t, any version x_t can be sampled in a single step.

When choosing a variance schedule $(\beta_t)_{t=1}^T$, it is important that each step is small, $(\beta_t)_{t=1}^T \ll 1$. When this is the case, the reverse steps $p\left(x_{t-1} \mid x_t\right)$ have the same functional form as the forward process $p\left(x_t \mid x_{t-1}\right)$, as it becomes more likely that x_{t-1} comes from a region close to where x_t is observed, which allows us to model this region with a Gaussian distribution. For example, in the work of Ho et al. (2020), $(\beta_t)_{t=1}^T$ is comprised of linearly increasing constants between $\beta_1 = 10^{-4}$ and $\beta_T = 2 \cdot 10^{-2}$, where $T = 1000$.

In the **reverse process**, new samples from $p(x_0)$ are generated by starting from a sample $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following the reverse steps $p\left(x_{t-1} \mid x_t\right)=\mathcal{N}\left(x_{t-1} ; \mu\left(x_t, t\right), \Sigma\left(x_t, t\right)\right)$. These steps are approximated via a neural network that receives as input the noisy image x_t and the embedding at time step t, and learns to predict the mean $\mu_\theta\left(x_t, t\right)$ and the covariance $\Sigma_\theta (x_t, t)$ of a Gaussian distribution.

Ho et al. (2020) propose significant simplifications to the previously-used formulation for the loss function of the network (Croitoru et al., 2022; Sohl-Dickstein et al., 2015). As a result, the network does not predict the mean and covariance directly - instead, it predicts the noise from the image. The loss function measures, for a given step t of the forward process, the distance between the real noise z_t and the estimation of the network $z_\theta (x_t, t)$ It is formulated as follows:

$$\mathcal{L} = \mathbb{E}_{t\sim[1,T]}\mathbb{E}_{x_0\sim p(x_0)}\mathbb{E}_{z_t\sim\mathcal{N}(0,\mathbf{I})} \left\| z_t - z_\theta (x_t, t) \right\|^2 , \tag{2.18}$$

where $\mathbb{E}$ is the expected value, and $z_\theta (x_t, t)$ is the network predicting the noise in x_t.

This formulation allows for images to be sampled from random Gaussian noise. Each time, due to variations accumulating over the reverse diffusion process, a different image will be generated. However, the model cannot be guided into generating a specific image. For this, a form of conditioning is required.

Conditional generation In conditional diffusion models, an additional input y is given to the generative network, which is trained to model the conditional distribution $p(x|y)$. y can be of any form, such as (an embedding of) a sequence of text, a label, a semantic map, an image (Dhariwal and Nichol, 2021), or other extracted features Baranchuk et al. (2022). In theory, it is possible to include y during the training process and have the model learn how y affects x at each step as $z_\theta (x_t, t, y)$. However, in practice, this is not an effective strategy (Dhariwal and Nichol, 2021).

As a solution, Dhariwal and Nichol introduce classifier guidance. The technique uses a pre-trained classifier to direct the reverse diffusion process in the direction of the gradients of $p(x|y)$. This method is very effective and flexible, as the gradients of any pre-trained model can be used, regardless of the modality of y. It is the conditioning method which is used in most works. It has the limitation of being dependant on a separately trained classification model, but that is also an advantage as the changes resulting from modifications to the classifier can be measured independently of the generative model itself.

Latent diffusion While diffusion models are capable of generating very realistic results, their practicality is limited by their computational requirements. The main reason for this is that, during inference, a sample image needs to pass through the generative network T

times. Improving the sampling speed has been an active area of research, and a number of approaches have been put forward.

One of the most significant improvements came in the form of latent diffusion, proposed in Rombach et al. (2022). The technique utilizes a pre-trained autoencoder network (Esser et al., 2021) trained with a combination of a perceptual loss and adversarial objective to encode an image x into a latent representation $z = \mathcal{E}(x)$ via the encoder $\mathcal{E}$. Using the DDPM formulation, noise is added to z over T steps and it is iteratively denoised by a generative U-net. The resulting denoised image is reconstructed from the latent space via the decoder, to produce the final generated image $\tilde{x}$:

$$\tilde{x} = \mathcal{D}(z) = \mathcal{D}(\mathcal{E}(x)) \tag{2.19}$$

This architecture is shown in Figure 11.

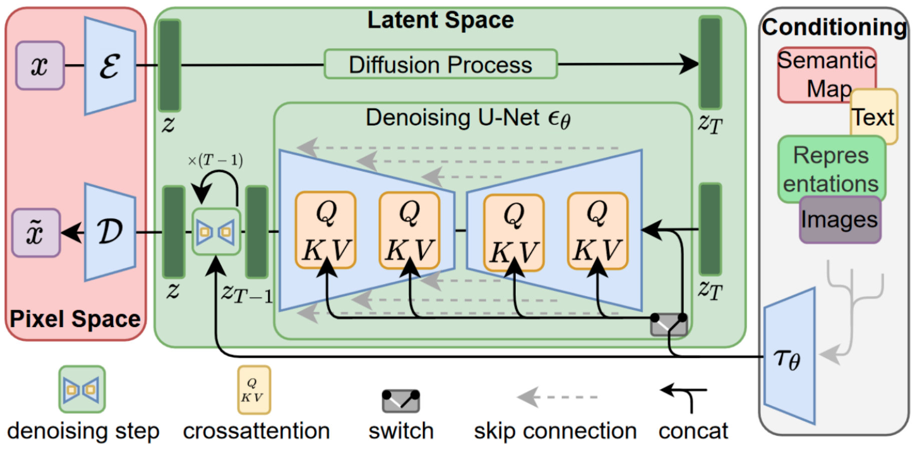


Figure 11. Latent Diffusion model architecture. Courtesy Rombach et al. (2022)

The final model produced improved results compared to non-latent models (Dhariwal and Nichol, 2021) while being less computationally demanding on inference. The authors point out several advantages to this approach: (i) The model is much more efficient because sampling is performed on a low-dimensional space. (ii) Due to U-net backbone of DMs, which makes them particularly effective for data with spatial structure, there is no need for aggressive compression which would reduce the quality of the decoded samples. (iii) Conditioning can be performed more effectively in the latent space.

2.1.4 Datasets

Supervised DL techniques learn by mapping data, in our case images, to their labels. There exist a variety of datasets, all with different images and labels for different specific applications. Initially, a specific dataset would be used to address application-specific problems in inpainting. However, as the field of computer vision has progressed to larger and deeper networks, which require more data to be trained, models are now trained with a combination multiple datasets. An advantage of this approach beyond an increase in the amount of data that is available is that the learned models tend to be more robust and generalize better.

In table 1, a quick overview of the most used datasets across all the works to train inpainting models is presented. This is intended as a reference for the reader when considering what the models of certain works were trained to to do. Of particular relevance to this project would be any works that used the Paris StreetView and/or Cityscapes datasets, as they most closely resembles the data of Cyclomedia.

Table 1. An overview of the most commonly-used image generation and inpainting datasets.

Dataset	**Description**	**Citation**
Places2	A large-scale image dataset of over 10 million natural scene images collected from various places around the world.	Zhou et al. (2018)
ImageNet	A large image dataset with over 14 million images categorized into more than $20,000$ classes, including animals, scenes, objects, and more.	Deng et al. (2009)
CelebA	A large-scale face attributes dataset with over $200,000$ celebrity images, annotated with 40 binary attributes such as "Smiling" or "Wearing Glasses."	Liu et al. (2015)
CelebA-HQ	A high-quality version of the CelebA dataset, with images generated using Generative Adversarial Networks (GANs) to increase the resolution and variability of the faces.	Karras et al. (2018)
DTD	A diverse textures dataset with images of textures such as fabric, paper, stone, and more, organized into 47 categories.	Cimpoi et al. (2014)
CMP Facade	A dataset of facade images of buildings, collected in the city center of Dresden, Germany.	Tyleček and Šára (2013)
Paris StreetView	A dataset of street scenes from Paris, France, with a focus on fine-grained recognition tasks, such as object and scene recognition.	Doersch et al. (2012)
Cityscapes	A large-scale dataset of urban scenes, with a focus on semantic understanding of street scenes in terms of object and structure recognition.	Cordts et al. (2016)

2.1.5 Evaluation Metrics

Evaluation metrics are used to assess the performance and effectiveness of algorithms. Different evaluation metrics are used to measure different aspects of an algorithm's performance. In this section, the most common and prevalent image restoration and inpainting evaluation metrics (Xiang et al., 2023) will be presented. The advantages and disadvantages of each will be discussed.

The **Mean Squared Error** (MSE) is a simple and widely used metric for evaluating the quality of image inpainting and reconstruction. It measures the average squared difference between the predicted and actual pixel values, which can provide a quantitative measure of the reconstruction error. Its value is sensitive to small changes in image quality, which makes it useful for comparing different reconstruction methods. However, MSE does not take into account the human perception of image quality, which limits its usefulness in some applications. In practice, the Normalized Mean Squared Error (NMSE), which is normalized by the variance of the original image, is more commonly used to make a comparison.

$$MSE = \frac{1}{C_j \times H_j \times W_j} ||I_{gt} - I_{out}||_2^2 \tag{2.20}$$

$$NMSE = \frac{||I_{gt} - I_{out}||_2^2}{||I_{gt}||_2^2} \tag{2.21}$$

Where I_{gt} is the original image, I_{out} is the output image, and they are of size $C \times H \times W$.

The **Peak Signal to Noise Ratio** (PSNR) is another widely used simple metric for evaluating the quality of image inpainting and reconstruction. It measures the ratio between the maximum possible power of a signal and the power of the noise present in the signal. Like MSE, it is sensitive to small changes in image quality and can provide a quantitative measure of the reconstruction error. However, like MSE, it does not take into account the human perception of image quality, which may limit its usefulness in some applications.

$$PSNR = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right) \tag{2.22}$$

Where MAX is the maximum possible value of a pixel, usually 255.

The **Structural Similarity Index** (SSIM) is a more advanced metric for evaluating the quality of image inpainting and reconstruction. It takes into account the human perception of image quality by measuring the structural similarity between two images, taking into account the luminance, contrast, and structure of the images. This makes it more suitable for evaluating the quality of real images or for tasks where human perception is important.

$$SSIM = [l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{\gamma} \tag{2.23}$$

Where α, β, and γ are parameters that control the relative importance of luminance, contrast, and structure, denoted by l, c, and s, respectively.

The **Learned Perceptual Image Patch Similarity** (LPIPS, Zhang et al. (2018)) is a distance metric that is specifically designed to capture the perceptual similarity between two images. It is one of the most commonly used metrics for evaluating deep learning inpainting & restoration systems. It uses a modified version of AlexNet (Krizhevsky et al., 2017) that has been fine-tuned on a large dataset of pairs of images, with each pair labeled according to their perceptual similarity. Specifically, the network is trained to predict the similarity between pairs of image patches using a Siamese network architecture, where two identical networks share weights and are trained to minimize the difference between the predicted similarity and the true similarity. It has a large number of filters in the lower layers, allowing it to capture a wide range of low-level image features, and uses local response normalization (LRN) and max pooling layers to increase its robustness to small image deformations and variations. LPIPS can provide more accurate and nuanced evaluations of image quality than traditional metrics like MSE or PSNR.

$$LPIPS = \sum_{l} \frac{1}{H_l W_l} \left\| w_l \odot \left(\Phi^l_{gt} - \Phi^l_{out} \right) \right\|_2^2 \tag{2.24}$$

Where l denotes a layer in the neural network, $\Phi_{\text{gt},l}$ and $\Phi_{\text{out},l}$ are the feature maps at layer l for the ground truth and output images, respectively.

The **Inception Score** (IS, Salimans et al. (2016)) measures how well the quality and the diversity of the images. The quality is measured based on how well the generated images can be classified by a pre-trained image classification network, usually Inception V3 (Szegedy et al., 2016). The diversity is measured by the entropy of the individual probability distributions, which captures the range of object categories covered by the generated images. A high IS score indicates that the generated images are both diverse and of high quality, while a low IS score indicates that the generated images are either low quality, not diverse, or both. It is important to note that the IS has some limitations, such as its sensitivity to the choice of the pre-trained image classification network and the dataset used for training it. Therefore, the IS should be used in conjunction with other evaluation metrics to provide a comprehensive analysis of the performance of a generative model. However, calculating an accurate IS requires a large sample of generated images, limiting its use for models which have a very high inference time (Lugmayr et al., 2022).

$$IS = \exp\left(\mathbb{E}_{\hat{x} \sim \mathbb{P}_g} \left[\log D_{KL} \left(p(y|x) \| p(y) \right) \right] \right) \tag{2.25}$$

The **Fréchet Inception Distance** (FID, Heusel et al. (2017)) is another measure based on the Inception V3 network (Szegedy et al., 2016). However, it uses the activations of the high-level features of the CNN to compare the distribution of the original and generated images, rather than its classification results. A low FID score indicates that the feature distributions of the real and generated images are similar, while a high FID score indicates that the feature distributions are dissimilar, implying that the generated images are of low quality or not diverse enough. Compared to the IS, the FID has the advantage of being less sensitive to changes in image quality and resolution, as it focuses on the high-level features extracted by the pre-trained CNN, rather than the pixel-level features of the images. However, the FID requires more computational resources than the IS, as it involves computing the mean and covariance of the feature representations of both the real and generated images, and computing the Fréchet distance between the resulting distributions. Also, much like IS, the FID requires enough samples to allow for a result to be calculated over a probability distribution.

$$FID = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\left(\sigma_r + \sigma_g - 2\left(\sigma_r \sigma_g\right)^{\frac{1}{2}}\right) \tag{2.26}$$

Where $\boldsymbol{\mu}_r$ and $\boldsymbol{\mu}_g$ are the mean feature vectors of the real and generated images, respectively, and $\boldsymbol{\Sigma}_r$ and $\boldsymbol{\Sigma}_g$ are the covariance matrices.

A **user study** is also often used, though not always due to the time, effort, and expenses associated with performing one. It is a subjective evaluation method that involves people rating the quality of the generated images. It takes into account the human perception of image quality, which is often the most important factor in assessing the effectiveness of an algorithm in real-world scenarios. They can be divided into two types - a single image study and a multiple images study. In a single image user study, either a real or generated image is shown to the volunteers, who attempt to guess whether the image is real or not. For the multiple images user study, multiple images generated from different algorithms are shown to volunteers. The volunteers are required to rank them based on some perceptual measure, such as realism. For image inpainting, a multiple image user study comparing against several algorithms is the most effective way to judge the subjective quality and realism of the images generated by the model.

2.2 Image Inpainting & Restoration

2.2.1 Non-Neural Methods

Geometry-based One of the seminal works, “Image Inpainting” by Bertalmio et al., proposes a method to “replicate the techniques used by professional restorators”, drawing on previous works in the fields of film restoration, texture synthesis, and disocclusion. The method uses partial differential equations (PDEs) to minimise an energy function, which is defined by two terms: a data fidelity term, which measures how well the inpainted region fits the known area, and a regularization term, which encourages smoothness in the result. The algorithm updates the image iteratively, moving inverse to the gradient of the energy function, until a minimum is reached and the image is inpainted.

This method and derivative geometry-based methods tend to propagate local structures from the border around the mask to the interior. They can produce convincing results over small regions (e.g., for image restoration), but they fail when inpainting large areas and tend to produce flat-looking, unrealistic images (Buyssens et al., 2015). A simple case in which Bertalmio et al.’s algorithm fails is shown in Figure 12.

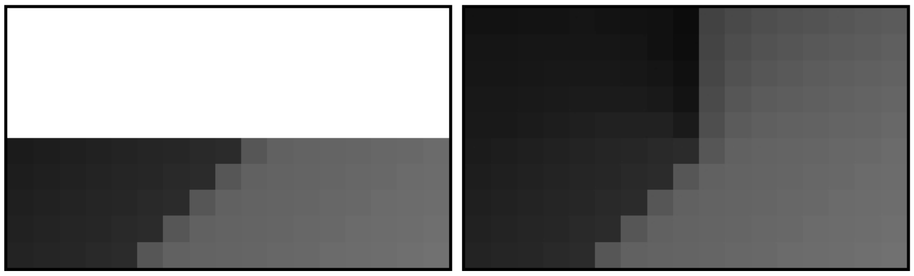


Figure 12. Left: Original image, mask in white. Right: Inpainted image using the Bertalmio et al. (2000) method.

Texture synthesis Texture synthesis is a computer vision technique that involves generating new textures based on existing textures with applications mainly in computer graphics & video games (Salem, 2021; Buyssens et al., 2015) (see Fig. 13). Initially, a simple stochastic model using Markov Random Fields Cross and Jain (1983) would be used, where textures are blended inwards from the border.

The Copy-Paste algorithm proposed in Efros and Leung (1999) is the pioneering work which introduced the concept of using image patches. The method is based on a self-similarity prior and the estimation of a pixel’s value based on a center patch. The nearest neighbors of the center patch are found through a search process, which is based on a metric that combines the vector of squared differences with a Gaussian weighting kernel. It works for simple and low-resolution textures, which were prevalent in computer graphics

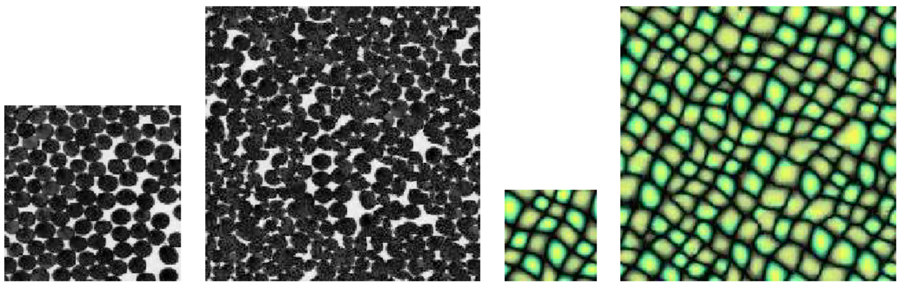


Figure 13. Results produced with the Wei and Levoy (2000) texture synthesis algorithm.

at the time, but it can produce repeating patterns and lacks the ability to capture large-scale patterns.

Wei and Levoy (2000) extends this approach by incorporating a multi-scale framework and using tree-structure vector quantization to speed up the patch search. Further improvements were made by Ashikhmin (2001), in which a more efficient search scheme that enhances the texture synthesis process while reducing the computational time is proposed. Both methods demonstrate the capability of synthesizing diverse textures while preserving the structure and properties of the reference texture. They produce smoother blending at faster speeds than the technique by Efros and Leung (1999).

Texture synthesis techniques produce good results for inpainting small missing regions in simple structure images, but they can not produce similar results if the image contains complex unrepeated structure. This is expected, as these techniques are intended to blend and synthesize texture files - a task which does not concern itself with semantics or structure. Due to this, such methods are not suitable for inpainting real-world scenes.

Exemplar-based In order to remove large objects from images, a new type of inpainting method was created. Exemplar-based methods, initially proposed in Criminisi et al. (2004); Bornard et al. (2002), attempt to fill in the missing region using patches of known image data. The algorithms work in 3 main steps, repeated until the region is filled in (Buyssens et al., 2015):

1. Select a pixel p on the border between the known region and masked area.
2. Select the patch $\Psi_{\hat{p}}$ that best matches patch Ψ_p, which is centred on p.
3. Paste the values of $\Psi_{\hat{p}}$ around p.

The order in which the image is filled (i.e., how p is selected) is critical. A naïve implementation would be the *onion-peel* method, used in Bornard et al. (2002), wherein the mask is filled iteratively from the boundary toward the centre. This method produces undesirable results, especially toward the centre of the mask (see Fig. 14, top). Contrary to Bornard et al.'s approach, Criminisi et al. (2004) propose a filling order in which pixels lying along the border are prioritised based on a confidence term C_p and a data term D_p. The data term ensures that the algorithm prioritises continuation of known structures that enter the masked area. The results of Criminisi et al.'s algorithm are shown in Figure 14 (bottom).

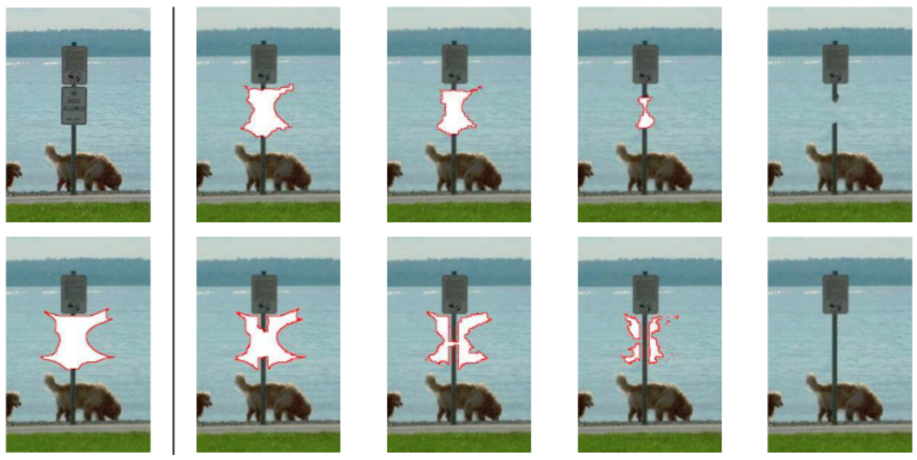


Figure 14. Top: Onion-peel filling order. Bottom: Data-aware filling order. Courtesy Criminisi et al. (2004).

Searching over the set of patches for the one that matches Ψ_p most closely is the most computationally expensive part of the algorithm. As Buyssens et al. show, this step is essentially a nearest-neighbour (NN) search, which has a computational complexity of $O(dN)$, where d is the dimensionality of the search space and N is the number of points (patches $\Psi_{\hat{p}}$) to be searched. Note that the search has to be performed for each iteration of the inpainting process.

There are many NN optimizations which can and have been applied to exemplar-based inpainting methods. For example, He and Sun use the KD-trees algorithm to do an Approximate Nearest Neighbor (ANN) search (Bentley, 1975), reducing the computational complexity to $O(\log(N))$, at the cost of needing to construct a KD-tree at each iteration. This is an approximate solution, wherein not all possible patches are considered, but in practice it has been found to give acceptable solutions. Another significant optimisation was proposed in Barnes et al. (2009). The PatchMatch algorithm is based on the idea that the search can be efficiently approximated by taking advantage of the local coherency of image patches. It works by first initializing the ANN field and then refining it iteratively in two phases: propagation and random search. The propagation phase improves a patch by looking to its neighbours, while the random search phase looks for a better source patch within a decreasing radius (See Fig. 15). See Ehret and Arias (2018) for an analysis of the complexity of PatchMatch and its many variants.

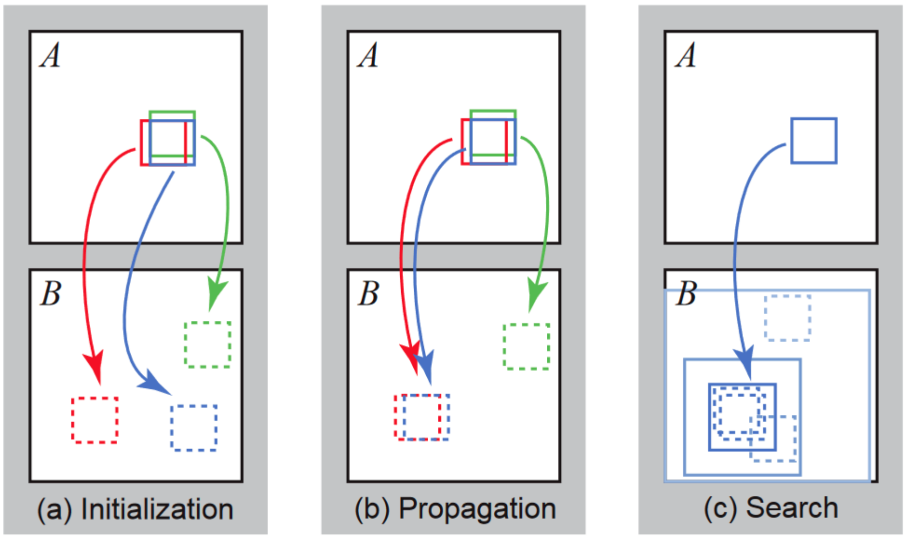


Figure 15. Phases of PatchMatch search algorithm: (a) patches are initialised randomly; (b) the blue patch checks green and red patches to see if they will improve its mapping, propagating good matches; (c) the patch searches randomly for improvements in concentric neighborhoods. Courtesy Barnes et al. (2009).

There are many variations based on Criminisi et al.'s exemplar-based algorithm. For example, Perez et al. (2004) uses information given by the user to constrain the search space, making it a semi-supervised approach. The methods in Wong and Orchard (2008); Guillemot et al. (2013) synthesise a patch $\Psi_{\hat{p}}$ by blending the K best-matching patches, but this tends to produce blurry textures in the missing regions (Buyssens et al., 2015).

Hybrid Geometry-based methods are effective at reconstructing global structures in images, but struggle with textural details. Exemplar-based and texture synthesis methods, on the other hand, are good at filling in texture but have difficulty reconstructing global structures, particularly when the missing part of a structure is not present in the rest of the image. As a result, hybrid methods that combine the different methods have been proposed, aiming to combine the strengths of each approach. Hybrid methods can be separated into two main classes.

The first class decomposes the image into structure and texture parts, which are then inpainted separately using a geometry-based method and a synthesis method respectively (Starck et al., 2005; Bertalmio et al., 2003). The inpainted structure and texture parts are then combined to reconstruct the final result. However, separating structure from texture is a challenging problem, especially in the presence of both small and large textures, which are common in natural images.

The second class uses a geometry-based method to construct an initial sketch of the strong edges of objects which enter the mask. This sketch is then used to define lookup areas for inpainting with an exemplar-based approach (Cao et al., 2011). These methods are able to reconstruct structures that are not present in the unmasked part of the image by continuing main structures that are broken by the mask, a major limitation of purely exemplar-based methods. Both classes of hybrid approaches can provide improved results over any of the other three methods, but they are limited in practice due to their high complexity and tendency to fail on natural images (Buyssens et al., 2015; Salem, 2021).

2.2.2 Neural Methods

Traditional image inpainting algorithms are prone to inconsistent image semantics, which leads to unsatisfactory results - especially when inpainting a large area in a semantically-rich context. Even hybrid methods, which balance the effectiveness of geometric, textural, and exemplar-based methods cannot achieve semantically consistent restoration at a large scale (Zhang et al., 2023). Since AlexNet (Krizhevsky et al., 2017), deep learning has shown remarkable results at a variety of tasks in computer vision and revolutionized the field. Deep learning-based inpainting algorithms have shown a much greater ability to capture semantic information and have been able to produce significantly improved results. This section will cover the most significant developments of deep learning-based approaches to image restoration and inpainting, from neural and convolutional methods, autoencoder and VAE-based approaches, through GAN architectures, and finally the most recent works with denoising diffusion probabilistic models.

One of the first attempts on neural image restoration focused on denoising. Burger et al. (2012) investigate whether a plain multi-layer perceptron (MLP) can outperform state-of-the-art non-neural methods of the time. They learn a simple direct mapping from noisy image patches to ones which are noise-free. While previous denoising works focus on specific types of noise, they demonstrate that the MLP-based approach can be adapted to any type of noise, given that it is trained on it. The results would not be surprising for us now, however this work marks a landmark in image restoration, as it is one of the first successful neural attempts relative to the state-of-the-art at the time.

Eigen et al. (2013) used a 3-layer CNN architecture for the task of blind image restoration of images corrupted by rain drops and dirt. They specifically target the real-world use case of images taken through a window, such as inside a vehicle or from outdoor security cameras. The CNN was trained on clean & corrupted image pairs and they demonstrate that it was able to learn the appearance of dirt and water droplets in natural images and map corrupted images to clean ones.

Image degradation - blurring, compression artifacts, noise - can be modelled via convolution. Xu et al. (2014) demonstrate that the inverse process - deconvolution, can be used to restore images. To do so, they propose a Deconvolutional Convolutional Neural Network (DCNN) - a 3-layer CNN. Note that they do not use any de-convolutional/transpose convolutional/fractional layers, but rather convolutional layers with learned kernels which reduce artifacts in a corrupted image. The proposed method achieves PSNR performance greater than a number of non-neural image restoration methods.

In Ren et al. (2015) propose an architecture which adapts the Shepard framework (Shepard, 1968) into a convolutional form and combines it with a 3-layer CNN, called Shepard Convolutional Neural Networks (ShCNN). They demonstrate the effectiveness of their architecture on the image inpainting and super-resolution tasks, achieving greater results than a deeper CNN architecture and a number of non-neural methods. The ShCNN network is composed of five layers, two of which are Shepard interpolation layers, and a ReLU activation function.

One of the first autoencoder-based works on inpainting focuses on blind text removal - blind meaning, no mask is provided to the model. Xie et al. (2012) train a stacked sparse denoising autoencoder (SSDA) - a series of autoencoders with a combination of sparsity constraint and stochastic corruption in the latent layers. They achieve PSNR results which are comparable to traditional denoising techniques (Portilla et al., 2003) and non-blind inpainting methods (Mairal et al., 2008).

Although these DL methods perform relatively well in simple cases, they still produce blurriness on the restored parts of the image and are unable to inpaint on complex scenes due to a lack of semantic understanding of the image. During this time, GAN models showed an impressive ability to generate complex images which were semantically sound. It did not take long before a GAN architecture would be applied to inpainting, to impressive results.

The work of Pathak et al. was the first to do so, with Context Encoders (Pathak et al., 2016). The architecture they propose is a combination of an autoencoder model and conditional DCGAN. They use the encoder to learn a compressed representation of the semantic features of the image surrounding the area to be inpainted, and condition the GAN generative network on those embeddings. In this way, they aimed to understand the content of the image in order to inform the generative network on the missing part(s). They train their network with a combination of MSE and adversarial loss and are able to generate a 64^2 pixel area in the center of 128^2 pixel images, while avoiding the blurring common when using (only) MSE loss. The model achieved exceptional results for the time - it was the first method which could recreate realistic inpainting results which are

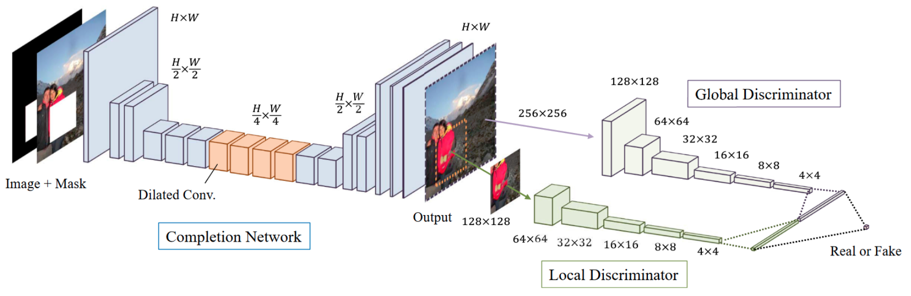


Figure 16. Architecture for "Globally and Locally Consistent Image Completion" by Iizuka et al.. Consists of a completion network and two auxiliary context discriminator networks used for adversarial training. The global discriminator network takes the entire image as input, while the local discriminator network takes only a small region around the completed area as input. The goal is to produce high-quality image completions that blend seamlessly into the surrounding area. Courtesy Iizuka et al. (2017).

semantically consistent with the rest of the image.

In Iizuka et al. (2017), an improvement of Context Encoders (CE) is proposed. Iizuka et al. introduce an extra discriminator in the GAN part of the architecture, creating a pair of discriminator networks, which determine whether an input image is real or fake. The extra discriminator is called a "Local Discriminator" and its input is only the generated section of the image, while the "Global Discriminator" gets the entire image as input. The adversarial loss function is then calculated as a combination of their outputs. The architecture is shown in Figure 16. The local discriminator ensures that there is local consistency in the inpainted part of the image and it greatly improves the performance of their model, especially when inpainting faces. In addition, their model uses dilated convolutions (Yu and Koltun, 2016) to expand the receptive field of the convolutions and Poisson blending on the output image for blending around the border of the inpainted area. Iizuka et al.'s approach improved on the generation ability of Context Encoders, and works with any mask and input image size. Jiang et al. (2020) propose a further improvement on CEs by adding skip-connections in the generator and by using Wasserstein loss, improving the stability of the training.

Despite the improvements proposed in Iizuka et al. (2017); Jiang et al. (2020), CNN models are inherently limited in their ability to borrow or copy information from distant parts of the image. This limits the ability of models to generate convincing images when there is a lot of structure in the scene, for example, when inpainting part of a fence. Borrowing information from distant parts of the image is something that traditional patch-based methods are explicitly designed to do, so a combination of a patch-based information gathering approach and GAN-based generator could be an effective solution.

This is exactly what is proposed in Yu et al. (2018) - a deep generative model which

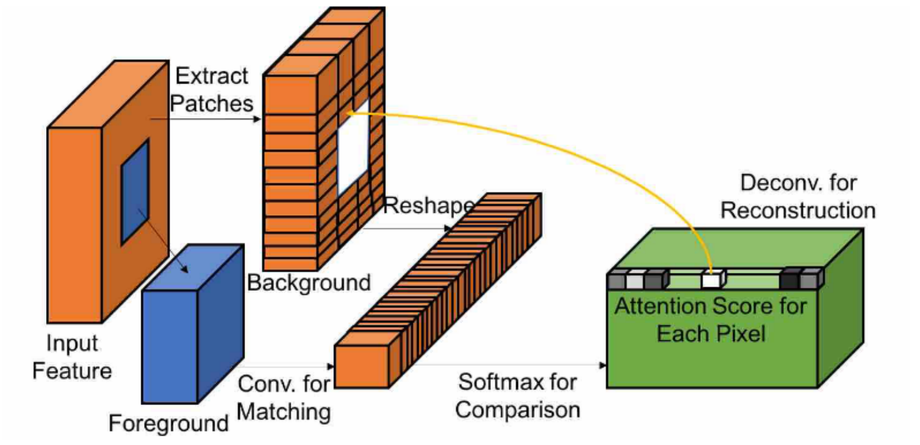


Figure 17. Architecture of the Contextual Attention Module (CAM). It computes matching scores between foreground and transformed background patches. Softmax is applied and an attention score for each pixel is obtained. The foreground patches are reconstructed with the attention scores using transpose-convolutional layers. Courtesy Yu et al. (2018).

could both generate novel image structures and explicitly borrow surrounding image patches to make better predictions. Their approach uses two stacked Wasserstein GANs - one for a coarse generation and one for refinement, and a global and local discriminator network, as inspired by Iizuka et al. (2017). The model learns to borrow information via a Contextual Attention Module (CAM) (see Jaderberg et al. (2015) on attention mechanisms), which determines where to borrow feature for the masked region by computing the cosine similarity between the background and foreground image patches (see Fig. 17).

The resulting model, a combination of, at the time, state-of-the-art techniques for image generation, is able to achieve very impressive results, but it does so at a high computational cost.

Yan et al. (2018) propose a new network called Shift-Net, which uses Deep Feature Rearrangement. Their architecture is based on a U-Net (Ronneberger et al., 2015), to which they add a novel shift-connection layer. Encoded features of the known region are shifted into the unknown area, to provide information for the generator. Guidance loss is used to encourage the connection between the information from the known region and the generated part of the image. The resulting network is able to produce comparable results to Yu et al. (2018); Iizuka et al. (2017) at a greater efficiency, especially excelling in inpainting missing regions which contain sharp structures and detailed textures. Also of note for this work is that they train on the Paris StreetView dataset.

In Liu et al. (2018), the authors propose a method for inpainting random masks. The architecture is based on a U-Net-like architecture, in which all convolutional layers are

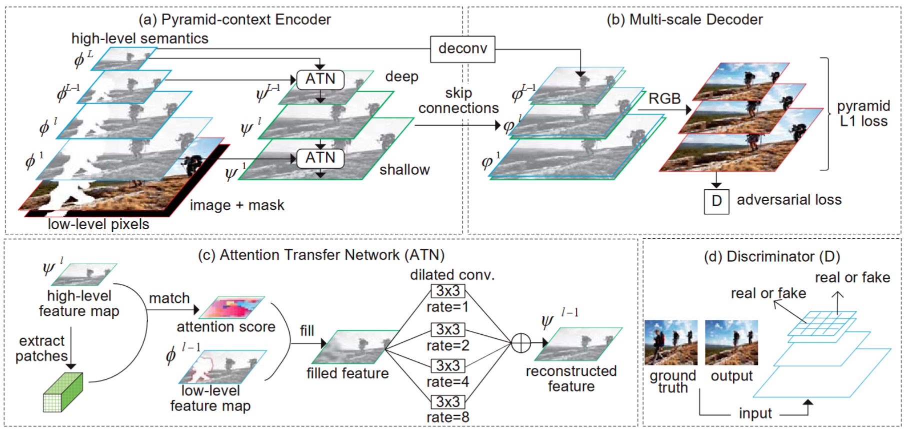


Figure 18. Pyramid-context ENcoder Network (PEN-Net) architecture, consisting of three specialized components: a pyramid-context encoder (a), a multi-scale decoder (b), and an adversarial training loss (d). The pyramid-context encoder effectively encodes compact latent features by utilizing the Attention Transfer Network (ATN) (c), which fills regions from high-level feature maps to low-level feature maps, thereby improving the encoding process. The multi-scale decoder takes reconstructed features from ATNs, latent features, and input data to generate output images. To optimize the network, pyramid L1 losses and an adversarial loss are minimized. Courtesy Zeng et al. (2019).

replaced with partial convolutional layers, in which the convolution result depends only on unmasked pixels. A joint loss function is used, comprised of a perceptual loss, style-loss (Gatys et al., 2015), and total variation (TV) loss (Johnson et al., 2016). The weights of these losses were determined by a hyperparameter search on 100 images from the validation set. However, this approach has some drawbacks, such as the high complexity of gating an un-learnable layer when multiplying it with the input feature map, and the limitation of each pixel to being either valid or invalid, which can lead to fading of valid pixels in deep layers.

Zeng et al. (2019) propose a novel model called PEN-Net - "The Pyramid-context ENcoder Network". PEN-Net, built on a U-Net structure, restores images by encoding contextual semantics from the full resolution input and decoding the learned semantic features back into images. The pyramid-context encoder learns region affinity through attention from a high-level semantic feature map and transfers this attention to previous low-level feature maps, ensuring both visual and semantic coherence for inpainting. Additionally, the network includes a multi-scale decoder with deeply-supervised pyramid losses and an adversarial loss, resulting in fast convergence during training and more realistic results during testing. The network architecture is shown in Figure 18. They test the trained network on Facade, DTD, CelebA-HQ, and Places2 and show state-of-the-art performance, at the cost of high computation and, as a result, a low output resolution.

Sagong et al. (2019) improves on Yu et al. (2018) with a model called parallel extended-decoder path for semantic inpainting (PEPSI). PEPSI has a single shared encoding network and a parallel decoding network with a coarse and inpainting path, reducing the number of convolution operations by almost half compared to conventional coarse-to-fine networks. The coarse path generates a preliminary inpainting result, which trains the encoding network to predict features for the CAM, while the inpainting path uses refined features reconstructed by the CAM to create a higher-quality result. In Shin et al. (2021) Diet-PEPSI is proposed - an improved architecture with fewer network parameters. A new type of convolutional layer called the rate-adaptive dilated convolutional layer is employed to use fewer hardware resources. The layer achieves this by modulating the shared weights with different scaling and shifting values for each dilation rate. By using these layers, Diet-PEPSI reduces the number of network parameters compared to using multiple standard dilated convolutional layers. The rate-adaptive dilated convolutional layers are implemented as residual blocks, called Diet-PEPSI units (DPUs). By replacing multiple dilated convolutional layers with DPUs, Diet-PEPSI is able to achieve the same receptive field size as PEPSI while having fewer parameters. PEPSI and Diet-PEPSI have improved computational efficiency while outperforming other models (such as those in Pathak et al. (2016); Iizuka et al. (2017); Yu et al. (2018)) in terms of PSNR and SSIM.

In Suvorov et al. (2022), large mask inpainting in complex scenes is attempted, with a method called Large Mask inpainting (LaMA). Large masks are difficult to inpaint with conventional convolutional layers due to the limited size of their receptive field - an issue that each of the last few architectures that were covered have tried to resolve. LaMa does away with convolutional layers in the first layers entirely. Instead, Fast Fourier convolution (Chi et al., 2020) layers are used, which have a receptive field size spanning the entire image. This enables the generator to consider global context right from the early layers, which is essential for inpainting of high-resolution images. This results in improved efficiency as trainable parameters can be utilized for reasoning and generation directly, rather than "waiting" for information to propagate from the outer parts of the image into the masked area. In addition, they use a novel high receptive field perceptual loss (in combination with an adversarial loss), optimized for the larger receptive field in the first layers. The results are very impressive - at the time of writing, this is the state-of-the-art in terms of GAN-based inpainting. The model shows very impressive semantic understanding and ability, but the results are still not always perfect. Removal of foreground objects in very busy, semantically rich scenes shows the limitations best - blurring and haziness appear, especially at the center of the inpainted area (see Fig. 20.

A previous thesis project, Uittenbogaard (2018), attempted to tackle inpainting at Cyclomedia. The method made use of the background information in multiple images adjacent to the source image which would be inpainted. Using the depth information captured as part of the Cycloramas, the adjacent images were reprojected such that the background information would match the area which is being inpainted. The masked image, mask, and matching masked background information from the four adjacent reprojected images were given as input into a GAN model, which would generate the information for the masked area. This method was used with the aim of preserving the truthfullness in the image, to ensure no objects that are not known to be there would be hallucinated. The approach is shown in Figure 19. The original, pre-processed, and generated data for the project are available and could potentially be used as part of this thesis. It can also be a useful method to evaluate against.

Diffusion-based Several works have evaluated the use of diffusion models for inpainting. The initial attempts (Meng et al., 2022; Song et al., 2021) adapted unconditional models to be used for conditional tasks (such as image inpainting, super-resolution, and colorization). The models are given as input the image with the masked area replaced by Gaussian noise, which is then de-noised over N steps. However, as this was a side application of the models they propose, the authors do not share implementation details nor quantitative evaluation metrics specific to inpainting. The first work to thoroughly evaluate such a model for inpainting is Chung et al. (2022). The authors train an unconditional model and apply it to a number of conditional tasks by changing the inference scheme which is used. The main advantage of the approach they propose is that they achieve a significant reduction in the number of sampling steps required at inference. This is done by not starting the denoising process from pure Gaussian noise, but from a better initialization, such as one-step correction created via a neural network. As a result, the network is capable of producing an output significantly faster than SDEdit, while having similar performance for inpainting.

Image-to-image diffusion models are conditional models, denoted as $p(y|x)$, where both x and y are images. Such models are presented in Dhariwal and Nichol (2021); Saharia et al. (2021) and applied to image super-resolution. The first work which examined diffusion models for inpainting, Saharia et al. (2022), employed an image-to-image conditional diffusion model. Palette is based on a modified U-Net architecture (Ho et al., 2020), inspired by the approaches shown in Dhariwal and Nichol (2021); Saharia et al. (2021); Song et al. (2021). The architecture is based on a 256×256 U-Net model with class-conditional layers, as proposed in Dhariwal and Nichol (2021), but differs in two main ways: (i) it does not include class-conditioning, and (ii) it includes additional conditioning of the source image through concatenation, following the approach in Saharia et al. (2021). For inpainting, the model is explicitly fine-tuned by removing random regions of training

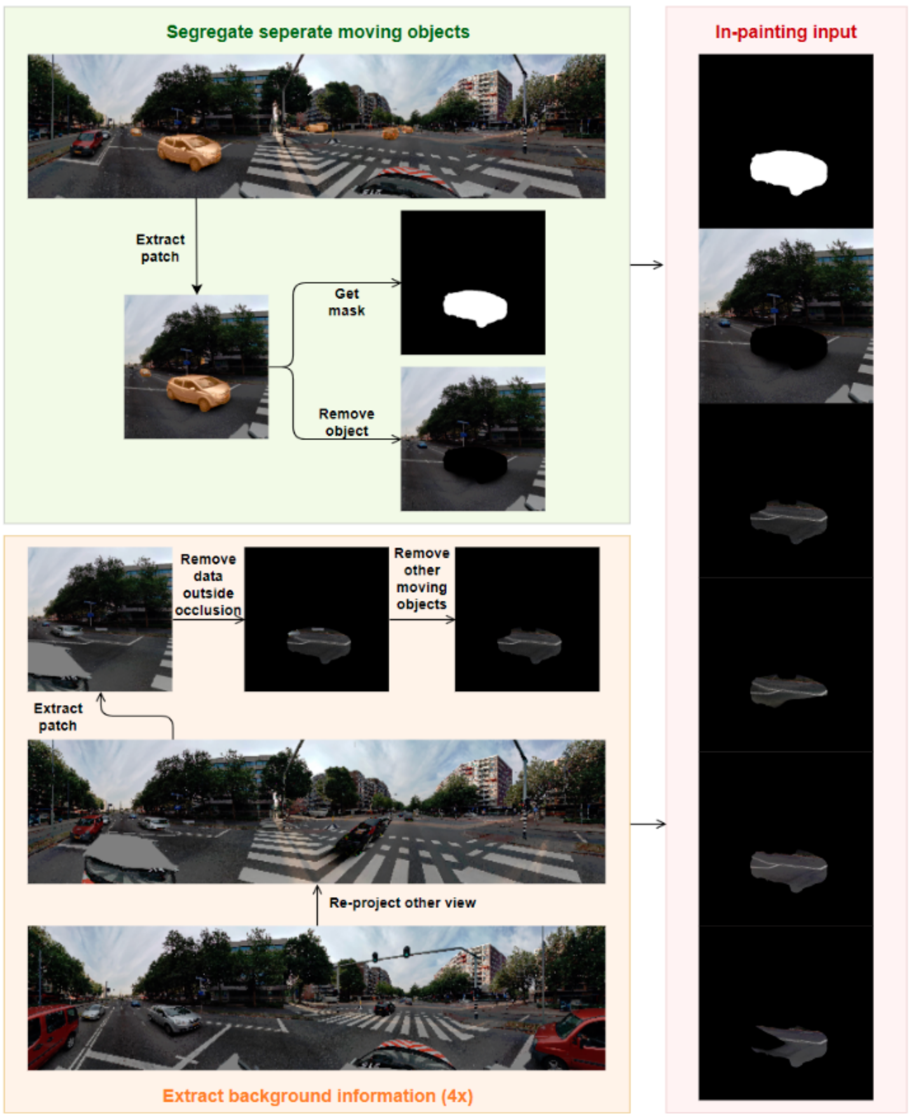

Segregate seperate moving objects
Extract patch
Get mask
Remove object
In-painting input
Remove data outside occlusion
Remove other moving objects
Extract patch
Re-project other view
Extract background information (4x)

Figure 19. An overview of the inpainting pipeline for a previous thesis project done at Cyclomedia. In the segregation part (top left), first one of the moving objects is selected. A patch centered around the object is extracted. From this patch, the moving object is removed and the result of this is an input for the inpainting network, together with the mask of the moving object. In the background information part (top left), four images taken close to the image of interest are reprojected to the point of view of the image of interest. A patch of the same area is extracted. From this patch, first the area outside of the occlusion is removed. After that, the moving objects within the occluded area are removed. After repeating this process four times, the four pieces of background information are fed to a GAN. Courtesy Uittenbogaard (2018).

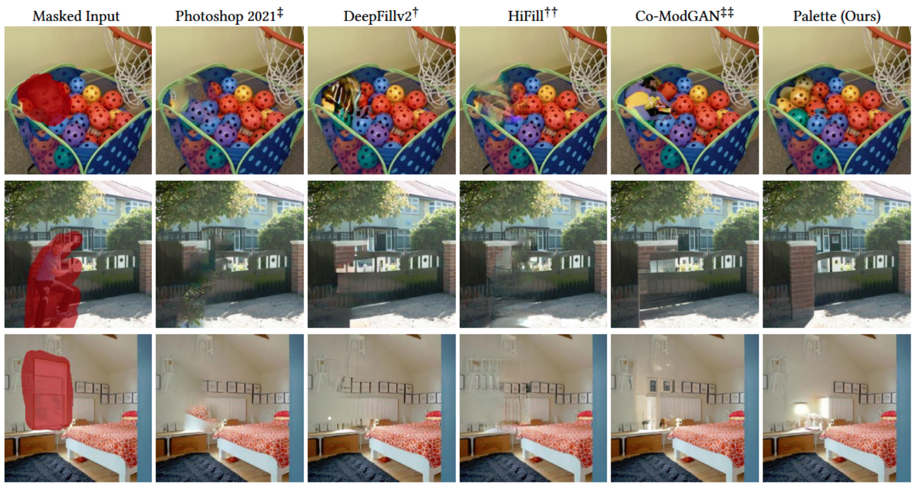


Figure 20. Comparison of several SOTA inpainting methods on object removal. Left to right: Masked input image; Result using Content-aware fill tool by Photoshop, which is built on the Patch-Match method (Barnes et al., 2009); GAN-based DeepFillv2 (Yu et al., 2019); GAN-based HiFill (Yi et al., 2020); Co-ModGAN (Zhao et al., 2021); and DDPM-based Palette (Saharia et al., 2022). An equivalent image comparing results between the methods described in this review could not be found, but the results are intended to show a general difference between the results produced by non-neural, GAN-based, and Diffusion-based methods. Courtesy Saharia et al. (2022).

examples, and adding an additional mask channel to the architecture. They evaluate the difference in results between an L_1 and L_2 loss function and find that an L_1 loss produces significantly lower sample diversity compared to L_2. Their approach fills the masked area with Gaussian noise, which the model then perturbs into a realistic image over 1000 steps. Both free-form and center masks are evaluated and compared to a number of SOTA GAN-based models in terms of a number of evaluation metrics including the Inception Score (IS) and Fréchet Inception Distance (FID). They achieve significantly improved results on the validation sets of the ImageNet and Places2 datasets. A results comparison image can be seen in Figure 20. The Palette model was also applied to colorization, outpainting, and JPEG restoration. It achieved SOTA performance on all of these tasks without any task-specific engineering to the model architecture.

The method presented in Lugmayr et al. (2022), called RePaint, is of particular interest for this work. This is the only method that was created specifically for inpainting and uses a diffusion model. Their method is based on the pre-trained guided diffusion model from Dhariwal and Nichol (2021), adapted in two ways: **(i)** it makes use of information from the unmasked regions, and **(ii)** it uses a custom denoising schedule to improve the harmony between the generated and the known part.

To explain **(i)**, consider that for each step $t \in T$, two images exist - one created by adding

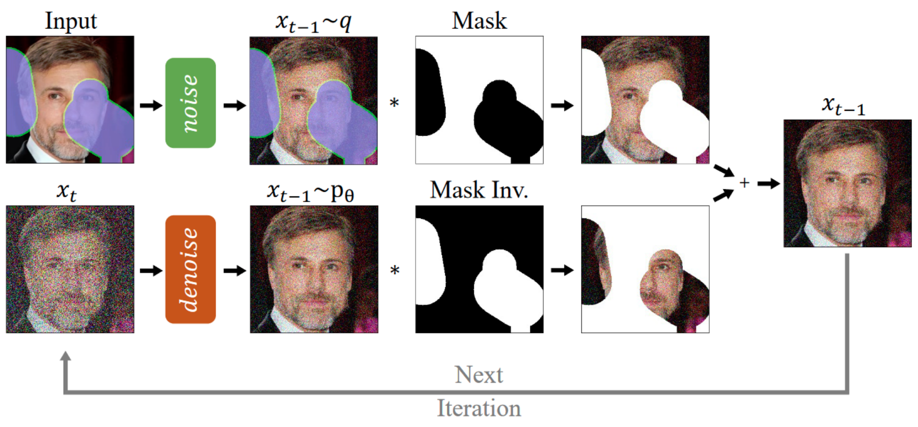


Figure 21. RePaint conditioning on the known region. At each step, the known region (top) is sampled from the input, while the inpainted part is sampled from the output of the DDPM (bottom). Courtesy Lugmayr et al. (2022)

noise, and one created by removing noise via the denoising U-Net. These two versions are denoted as $(x_t|x_{t-1})$ and $(x_t|x_{t+1})$ respectively, based on whether they were derived by a previous or later stage along T. $(x_t|x_{t-1})$ is the version the denoising network is trying to recreate and can be considered the ground truth version. To condition on the known region, at each step t, $(x_t|x_{t-1})$ and $(x_t|x_{t+1})$ are combined by taking $(x_t|x_{t+1})$ for the masked area and $(x_t|x_{t-1})$ for the known area. This approach is also illustrated in Figure 21.

When using solely this conditioning, Lugmayr et al. found that while the model would leverage the context of the known region, the result would not harmonize well with the rest of the image. The authors suggest that this disharmony is because rough boundaries are created between the known and unknown regions over and over again, at each step t, and the network does not have enough time and flexibility to fully converge. This is exacerbated from the variance schedule of β_t, due to which the maximum change in the image decreases with t. To combat this, **(ii)** is used - the output of the DDPM x_{t-1} is diffused back to x_t and denoised by the DDPM again. This scales back the output, but some information from the generated region of x_{t-1} is still kept in x_t and it gives more time for the network to harmonize the information of the real and generated regions. The jump size j is a hyper-parameter which controls how many steps are taken back, and r controls the number of times the resampling is done. In their experiments, which explored a variety mask types on the ImageNet, CelebA-HQ, and Places2 datasets, they find that $j < 10$ tends to produce blurry results, and an increased number of resamplings r improves the overall image consistency (saturating around $r = 10$).

RePaint (Lugmayr et al., 2022) is currently the best performing inpainting method in literature. It achieves greater LPIPS score than diffusion-based SDEdit (Meng et al., 2022) and GAN-based LaMa Suvorov et al. (2022), and was its results were preferred in a user study. There is a major caveat to these results however - computation time. Due to the resampling strategy, each step of the diffusion process has to be taken multiple times, computationally having an effect similar to significantly increasing the size of T. This greatly increases the number of sequential steps performed during inference. The pre-trained DDPM RePaint is based on, Dhariwal and Nichol (2021), is not a latent model and works in pixel space. If a DDPM model operating in a latent space of an autoencoder (Rombach et al., 2022; Vahdat et al., 2021) is used instead, the computational requirements would potentially be reduced. However, this approach has not been explored yet.

3. Methodology

In this section, the methodology used to answer the research questions and sub-questions is explained. First, the data pre-processing pipeline for both training and inference is defined, including the mask-generation and refinement processes. Subsequently, the fine-tuning methodology and training process for the latent diffusion model is detailed. Next, we introduce the custom partial loss function - a modification of the native latent diffusion loss function, which was used with the aim of improving the quality of the inpainting results. An explanation of the data filtering technique that was applied in order to analyze and select the most suitable images from the training set for fine-tuning the model follows. Finally, the quantitative and qualitative evaluation methods are defined.

3.1 Data Preprocessing

The data for this project is a subset of the overall Cyclomedia dataset. It consists of ~ 4000 images, which were semantically labelled for over 40 classes by human annotators, prior to the beginning of this research. Most importantly for this research, it contains labels for people and different types of vehicles.

Prior to any experiments being conducted, the data was split in an $8{:}1{:}1$ ratio to create a training, validation, and test set. The images were separated and the test set was used only for final evaluation.

3.1.1 Training Images

Crop Generation The latent diffusion architecture we utilize is designed to process images of size 512x512. However, the Cyclomedia dataset comprises of cubic images of size 4068x4068. Therefore, it is essential to resize the images to enable processing by the model. One possible technique for downscaling the images is bicubic interpolation Hirahara et al. (2021), but such an aggressive downsampling approach would limit the upper boundary of the quality of the generated samples.

To address this concern, an alternative approach for generating 512x512 image crops is used. The naive approach of simply dividing the original images into fixed-sized crops may result in several limitations. Since the original images capture a wide field of view, a significant portion of the images constitute sky or street regions, which offer limited semantic information - since there are usually no people or cars in the sky. Including these semantically poor crops in the training process could potentially cause overfitting and result in inflated performance metrics. Additionally, during the inference stage, objects

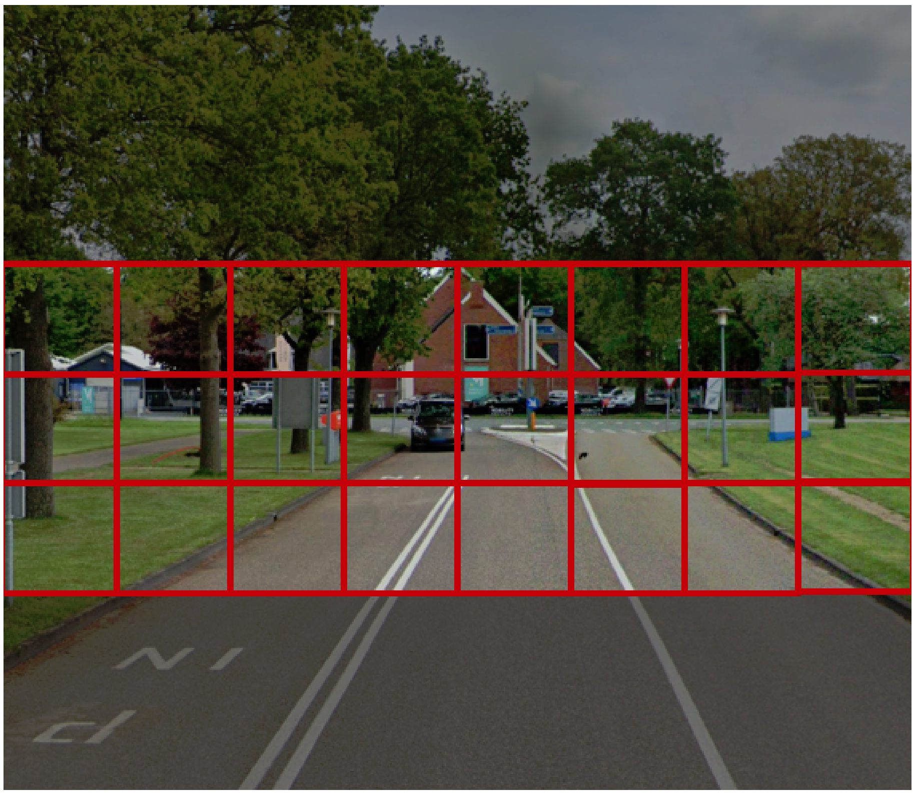


Figure 22. Training crops generation process. The darkened areas on the top and bottom of the image are discarded, and 8x3 crops of size 512x512 are created in a sliding-window fashion. The crops are highlighted in red. Note that the licence plate was blurred manually for publication.

close to the camera, such as cars, might occupy an area larger than 512x512 and would need to be downscaled to fit into the model. Ignoring these cases during training would likely lead to suboptimal generative performance.

Considering these factors, we adopt a two-pronged approach to cropping the images. Firstly, 512x512 crops from the areas where the objects of interest are most likely to appear are selected. Specifically, the region between the upper $1/3$ and lower $1/4$ of the original image is targeted. This targeted selection ensures that the model receives informative input and aims to evade overfitting on semantically poor areas. The crops are created in a sliding-window fashion, until the entire focus area is covered. A visual example of this can be seen in Figure 22.

Secondly, larger crops exceeding the 512x512 size are created. To accomplish this, a point within the image is selected and a random integer greater than 512 is chosen and used as the dimensions of the crop. Subsequently, image crops are extracted from the selected area surrounding the previously chosen point. These resulting crops are then downscaled to a 512x512 size using bicubic interpolation. This process is repeated until 10 larger crops are obtained for each image.

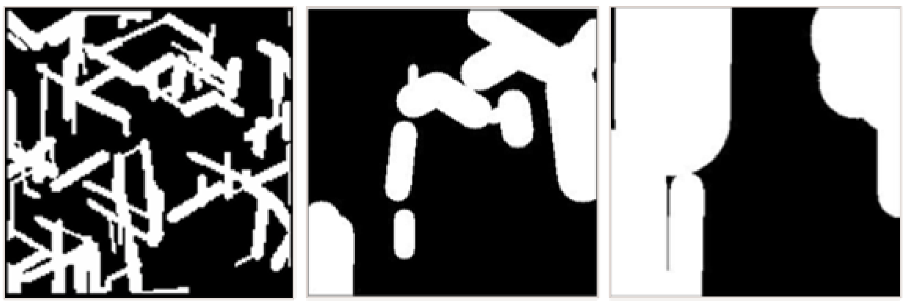


Figure 23. Different types of polygonal synthetic masks generated following the Suvorov et al. (2021) method. Left: Thin mask. Middle: Medium mask. Right: Thick mask.

Thus, for each original image, 34 crops are created: 10 downscaled crops and 24 native crops. This approach ensures a diverse set of training samples that encompass various object scales and semantic regions within the images.

Synthetic Masks For training, the method of Suvorov et al. (2021) is followed to create synthetic masks. It offers two types of masks: polygonal chain masks and box masks, at three sizes - thin, medium, and thick. For polygonal chain masks, the algorithm samples the number of line segments and their respective parameters, such as starting points, angles, lengths, and widths. The algorithm then draws each segment on the mask array.

In the case of medium and thick masks, there is a 30% chance that the mask will be a box mask instead of a polygonal chain mask. A box mask is created as a combination of a random number of boxes, with a random size within their respective medium/thick range, placed randomly within the image. The configuration for generating each type of mask was kept the same one used in Suvorov et al. (2021). For each image crop, 6 masks are created - 1 thin, 3 medium, and 2 thick. An example of each mask is shown in Figure 23.

3.1.2 Inference Images

Mask Refinement The Cyclomedia dataset we used contains human-annotated object labels and segmentations for 23 classes including People, Cars, Trucks, and more. The object labels were accurate, but the available segmentations had low quality, often missed large parts of the object, and were insufficient for masking the objects of interest. An example image masked based on the available Cyclomedia segmentation is shown in Figure 26.

The accuracy of the object masks is a very important for the performance of object-removal techniques, due to the fact that the network leverages any information in the image to complete it in the most realistic manner. If the mask does not cover the object fully, the

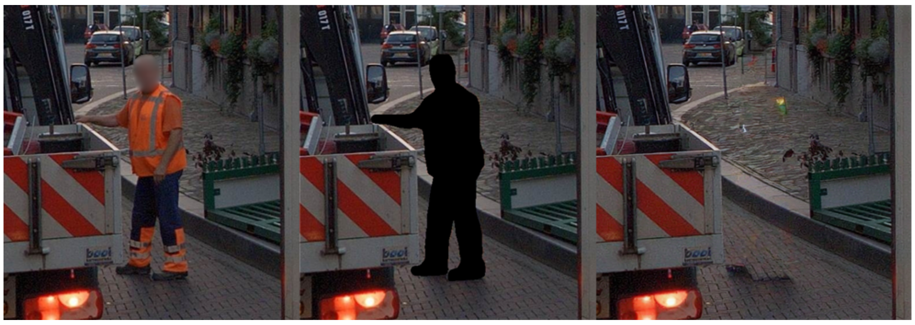


Figure 24. Example of artifact caused by imperfect mask. A single orange pixel last left unmasked from the right vest of the person, which results in a large orange artifact. Note that private information was blurred manually for publication.

model, in its aim of creating a realistic image, would continue and sometimes entirely replace the remnants of the object with an artificially-created one in its place. We found that if even a single pixel remained from the edge of the object, the network would propagate it into the inpainted area and would result in significant artifacting. An example of this is shown in Figure 24, where a single orange pixel from the vest of a person was left unmasked, and caused a large orange artifact to be left in the inpainted result.

To address the challenge of accurate automatic mask generation, we leverage a recently published model for image segmentation, called SAM (Segment Anything Model) (Kirillov et al., 2023). SAM is a foundation model that can accurately segment objects from natural images, with the ability to generate complex boundaries, even ones of occluded or partially visible objects. It can be prompted by one or several points, or a bounding box surrounding the object. SAM is based on the Vision Transformer (ViT) architecture (Dosovitskiy et al., 2021), which uses self-attention to process images as a sequence of patches. SAM extends ViT by adding a segmentation head that predicts a binary mask for each patch, and then upsamples the mask to the original image resolution. Prompts are encoded into a latent vector, which is then concatenated with the patch embeddings. The architecture of SAM is shown in Figure 25. SAM is trained on a large-scale dataset of over 1 billion masks on 11 million images, covering a wide range of objects and scenes. It is designed to be used in natural images and does not require any fine-tuning to produce high-quality segmentations. When combined with the existing object labels and bounding boxes, SAM is ideal for accurately selecting the regions of the image in which an object is present.

The center point and bounding boxes of the existing masks are calculated and used as the input prompt. While SAM produces very accurate results, it can sometimes miss the absolute edges of an object, and the issue of having absolutely full coverage of the object is still present. As such, the output mask is dilated with a filter of size $7 * 7$ (Yu and Koltun, 2016) to ensure the entire object is removed. An example of a final refined mask can be

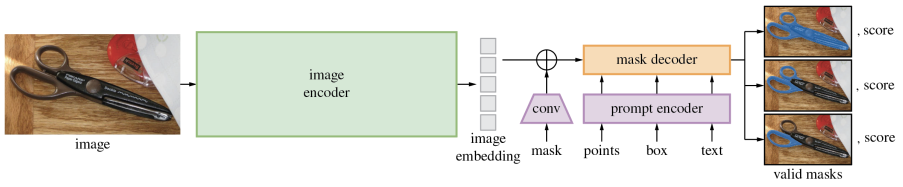


Figure 25. Architecture of Segment Anything Model (SAM). Diagram courtesy Kirillov et al. (2023).

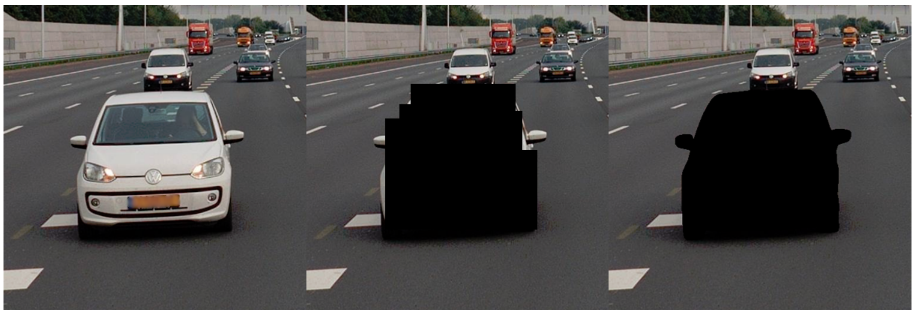


Figure 26. Left: Unmasked image crop; note that private information was blurred manually for publication. Middle: Image masked based on Cyclomedia segmentation. Right: Image masked based on refined mask, generated with SAM (Kirillov et al., 2023).

seen in Figure 26.

3.2 Fine-Tuned Latent Diffusion

The **Fine-Tuned** model uses a configuration closely based on the **Baseline** model. The settings for the diffusion process, e.g. the number of denoising steps t, the settings for the attention head, and others, were not changed. The encoder-decoder network was kept static for the training runs, and was not modified in any way. This was done so that the two networks would be closely comparable, with the difference being in the fine-tuning process. The changes in the modified network should be due to its exposure to novel training images, rather than anything stemming from model settings.

The training data for the **Fine-Tuned** model was pre-processed as described in Section 3.1.1. However, due to the long training runs and limited compute time, not all image crops were used. Instead, the model was trained on a random selection of 10% of the crops.

A significant number of tests were done to determine an optimal training strategy. Between training runs, the results of changes to the learning rate and different optimizers were tested on the validation set, until appropriate settings were found. Setting the learning rate to a value greater than 10^{-3} resulted in the loss exploding and the model outputting noise, while a value lower than 10^{-8} did not allow for the model to change sufficiently and

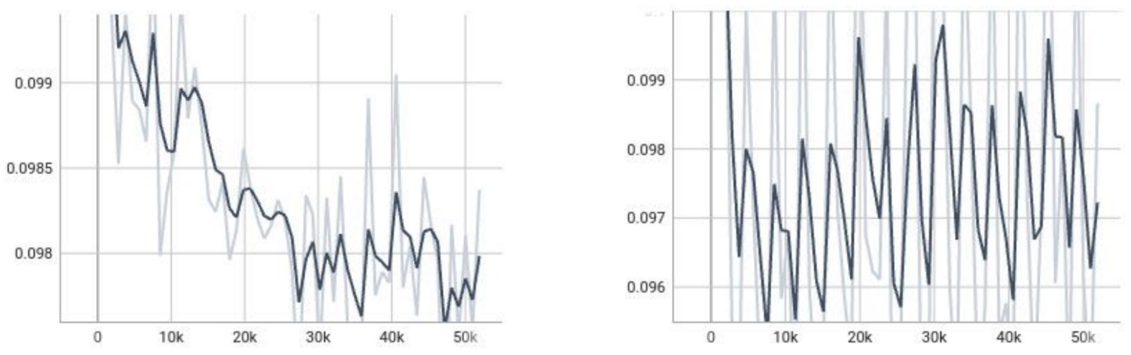


Figure 27. Left: Training loss. Right: Validation loss. Note the low loss value from the beginning of training.

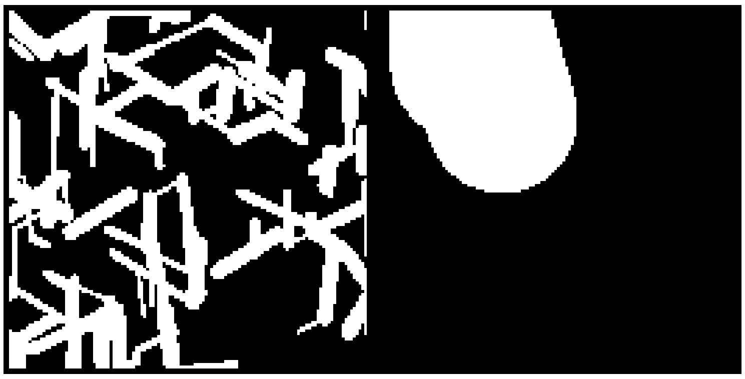
prevented learning as a result. As such, a static learning rate of 10^{-6} at a batch size of 10, with the Adam optimizer (Kingma and Ba, 2017) were used. The **Fine-Tuned** model was trained for 50 epochs.

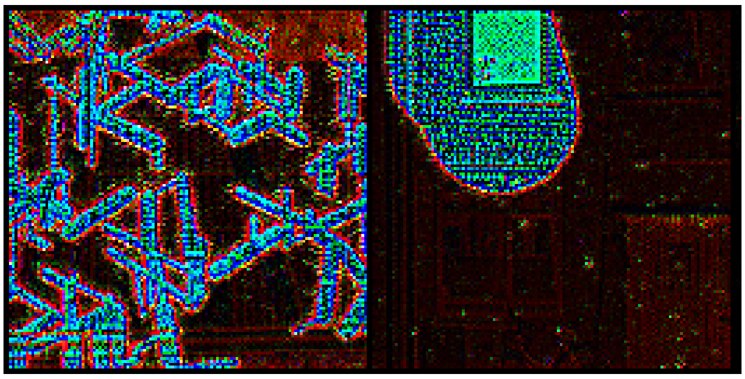
After finding an optimal learning configuration for the model, based on the validation set, the model was learning and showing improvements. However, when looking at both the training and validation loss during training, shown in Figure 27, it became apparent that the initial loss very low and reductions were small. This could be explained by the high performance of the baseline, and a limitation of the loss function to find sufficient differences between the target and model output. This is addressed in the following section, where a new inpainting-specific loss for the Latent Diffusion architecture is proposed.

3.3 Partial Loss Function

In the Latent Diffusion architecture, the training loss is computed based on the difference between a target encoded image and the output of the denoising U-Net. This is computed at each step t of the denoising process. In the case of inpainting, the target is the latent masked image, being compared against the output of the U-Net at each timestep t. However, consider that we are only interested in the area which is being changed - the masked area, a proportionally small area of the whole image. The rest of the image may also be changed due to the encoding process, which would introduce unwanted noise when computing the loss, potentially impeding the training process.

The difference caused by the encoding process can be seen in Figure 28b, where a visualisation of the difference of the unmasked encoded image and masked encoded image is shown. This visualisation is produced by subtracting the two encoded image matrices. Ideally, in the case where the two encodings are identical outside the mask, there would be no noise in the unmasked area. Figure 28b shows that this is not the case. This noise


(a) Synthetic image masks. Masked areas shown in white.


(b) Difference between unmasked and masked image embeddings. The two embeddings are not identical in the unmasked area and the differences are showing in the form of noise.

would cause some effect on the loss-computation, and may have been impeding learning.

To curb this, we propose the Partial Loss function - a training loss function which only takes into account the masked areas of the image. The parts of the image which should remain unchanged are not taken into consideration. This is done by setting the values of the encoded image matrices to be identical in the unchanged regions. This is done by creating an inverse mask (i.e. pixels within the masked area are set to 1 and ones in the unmasked area to 0) and computing the Hadamard product of the two matrices. The loss can be computed as before, as the difference between the two resulting image matrices.

The final loss is then weighted by the inverse of the proportion of the masked area. This is done in order to keep the overall value of the loss consistent and provide a strong enough signal for training the model, even on images with a very small training mask.

The formula for calculating the Partial L1 Loss is shown below, in equation 3.1.

$$\text{Loss} = \frac{\left|M_I^{-1} \odot I_{pred} - M_I^{-1} \odot I_{gt}\right|}{1 - \sum(M)/(I_X * I_Y)} \tag{3.1}$$

Where I is the embedded image, M_I is the corresponding image mask, M^{-1} is the inverse of the mask, $\sum(M)$ is the total number of masked pixels, I_X and I_Y are the X and Y dimensions of the embedded image, I_{pred} and I_{gt} are the model prediction and ground truth respectively, and $\odot$ is the element-wise product.

The Partial Loss method has the advantage of being both very computationally efficient and being compatible with any method for computing the difference between the images. Models trained with both a Partial L1 and a Partial L2 loss were tested and evaluated on the validation set, the results of which are presented in Table 3. Additionally, a test was done with an Unweighted Partial L1 loss, the results of which are reported in the appendix (A, B).

The loss of the final **Partial Loss** model is weighted and based on the L1 loss, as these settings were found to be optimal on the validation set. It was trained following the training configuration used for the **Fine-Tuned** model. This was done to ensure that the two models would be comparable and the effects on the results would be based on the modifications to the loss function.

3.4 Data Filtering

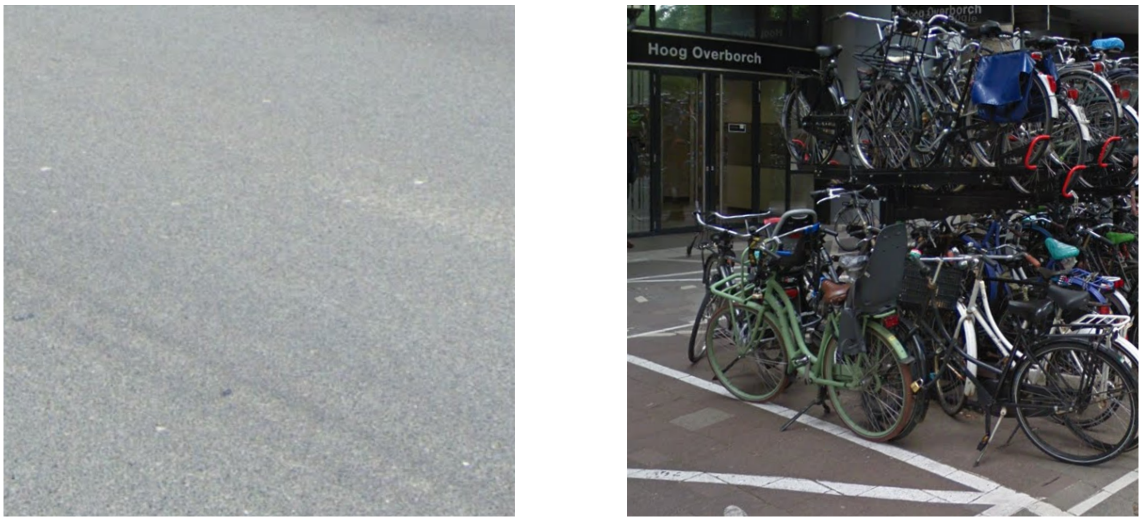


Figure 29. Comparison of one of the lowest and one of the highest complexity images, as reported by the segmentation-based complexity measure. Left: Image with low complexity. Right: Image with high complexity.

During the training process, we found that there were issues with the image cropping when generating the training data. Despite the efforts made to include meaningful and informative data, described in Section 3.1.1, a significant number of image crops contained little to no meaningful information, being comprised of entirely sky or ground.

In order to combat this, a more sophisticated method of data filtering was necessary. The method had to be able to recognize image crops which do not contain much information, i.e. crops with a low complexity. There are different techniques for measuring image complexity, such as spatial information measures (Yu and Winkler, 2013), compression-based techniques (Yu and Winkler, 2013), entropy-based measures (Zhou et al., 2019), and feature-based techniques.

We chose to use a feature-based technique, particularly SAM, because, as Kirillov et al. demonstrate, it has the ability to produce very accurate segmentation maps and works very well on natural images. Similarly to other feature-based techniques, the number of segmentations correlates with the complexity of the image. That is to say, a more complex image will result in a higher number of segments being output by SAM. You can see an example of a low- and high-complexity image in Figure 29.

All of the training crops were processed and the complexity of each one was stored. A histogram of the results and the number of segments in each image is shown in Figure 30. Note that, while there is a peak of images with around 50 segments, there is a large number which have less than that, as well as a long tail of highly-complex crops, with the highest being 400. We chose to focus on the most complex images, the ones lying in the long tail on the right of the chart.

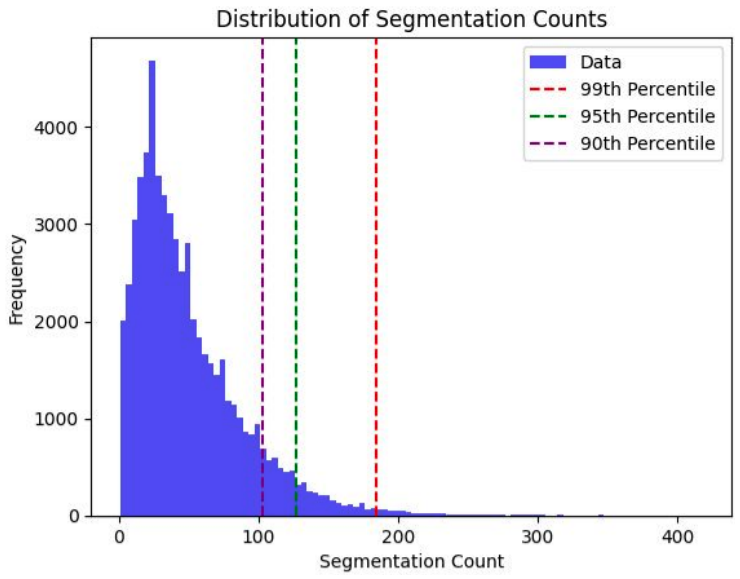


Figure 30. A histogram showing the distribution of images relative to the number of segmentations found in them. Notice the large number of images with few segmentations (i.e. low complexity) and the long tail on the right.

A new subset of the data was thus created, keeping the same size as the data used to train the **Fine-Tuned** and **Partial Loss** models. This dataset focuses on the 90th percentile most-complex images, all of which have more than 100 segmentations, as can be seen in Figure 30.

The model fine-tuned on this data was based on the **Partial Loss** model, and is referred to as the **Top-N** model. Only the training data was changed - the validation crops were kept identical between all three models. No other configuration changes were made, to ensure comparability between the **Top-N** and **Partial Loss** models.

3.5 Evaluation

3.5.1 Quantitative Measures

Quantitative measures evaluate how close the generated image is to the original image, both in terms of pixel values and perceptual similarity. The four key metrics which are considered are the Peak Signal-to-Noise Ratio (PSNR, 2.22), the Structural Similarity Index (SSIM, 2.23), the Learned Perceptual Image Patch Similarity (LPIPS, 2.24), and the Fréchet Inception Distance (FID, 2.26). All of these measures are widely used in literature when evaluating image generation and inpainting algorithms (Xiang et al., 2023; Benny et al., 2021).

PSNR and SSIM are two common quality measures that compare the generated image and the original image pixel by pixel. PSNR measures the reconstruction error (a higher value indicates a lower reconstruction error and a better image quality), while SSIM measures the structural similarity (ranging from -1 to 1, where 1 means identical images and -1 means completely different images). See 2.22 and 2.23 for more details on how the measures are computed. Pixel-based metrics compare the generated image and the ground truth image pixel by pixel, but they do not capture the perceptual aspects of image quality, such as texture, color, and style.

LPIPS is a measure based on a deep neural network that learns to predict human judgments of image similarity. Because of this, it can better reflect how humans perceive image similarity than pixel-based metrics. It can capture these aspects by comparing the feature representations of the images at different layers of a pre-trained network. LPIPS can also handle variations in image content, such as object pose, shape, and occlusion, that are common in image generation tasks. It ranges from 0 to 1, where 0 means identical images and 1 means very different images - therefore, a lower LPIPS score indicates a higher quality image.

Diversity measures how varied the generated images are, given different random noises as inputs. A good inpainting model should be able to generate diverse images that are consistent with the observed regions and the image context. FID is a diversity measure that compares the feature distributions of the generated images and the real images using a pre-trained Inception-V3 network (Szegedy et al., 2016). It can capture both the realism and the diversity of the generated images. Because of this, it has been shown to correlate well with human judgments of image quality (Benny et al., 2021). A lower FID value means that the generated images are more similar to the real images in terms of feature statistics.

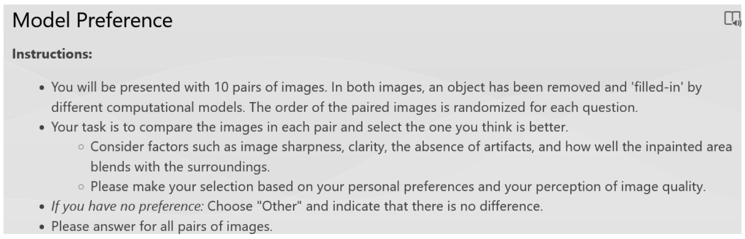
While quantitative measures are important for establishing a consistent and objective measure of comparison between different models, they can have limitations in terms of their ability to fully evaluate the quality of a generated image. Although perception-based metrics such as FID and LPIPS approximate how people perceive and evaluate images, quantitative measures alone may not capture human preferences fully.

3.5.2 Qualitative Measures

Qualitative measures are based on human judgments of the generated images, such as their realism, diversity, and consistency with the observed regions and the image context. They can be less consistent and more difficult to collect than than quantitative measures, as they depend on the subjective opinions of the participants. However, they can provide more insights into how humans perceive and evaluate the generated images, and complement the quantitative measures.

To collect qualitative feedback from the participants, we designed a survey with 2 sets of 10 questions. The first set of questions compared the **Baseline** and the **Fine-Tuned** models, and the second set compared the **Fine-Tuned** and **Top-N** models. Each question presented a pair of images, from each of which a person had been removed and inpainted using one of the models. The order of the images in each pair was randomized for each question, so the participants did not know which model generated which image.

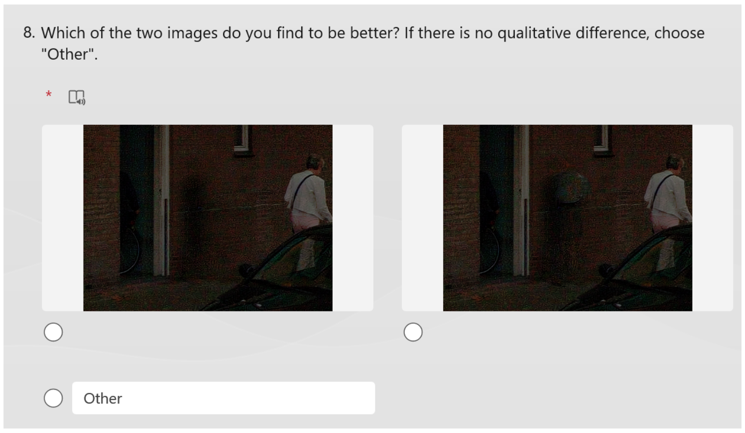
The participants were asked to compare the images in each pair and select the one they thought was better, based on their personal preferences and their perception of image quality. They were instructed to consider factors such as image sharpness, clarity, the absence of artifacts, and how well the inpainted area blends with the surroundings. If they had no preference, they could choose “Other” and indicate that there was no difference. You can see the survey instructions in Figure 31a, and a sample question in Figure 31b.


Model Preference

Instructions:

- You will be presented with 10 pairs of images. In both images, an object has been removed and 'filled-in' by different computational models. The order of the paired images is randomized for each question.
- Your task is to compare the images in each pair and select the one you think is better.
  - Consider factors such as image sharpness, clarity, the absence of artifacts, and how well the inpainted area blends with the surroundings.
  - Please make your selection based on your personal preferences and your perception of image quality.
- *If you have no preference:* Choose "Other" and indicate that there is no difference.
- Please answer for all pairs of images.

(a) Survey instructions shared with participants prior to being shown the questions.


8. Which of the two images do you find to be better? If there is no qualitative difference, choose "Other".

(b) Sample survey question. Each image was generated from a different model. The order of the images was randomized for each question.

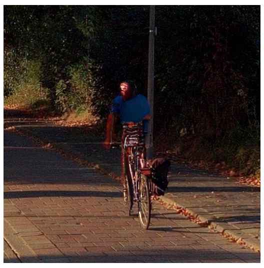
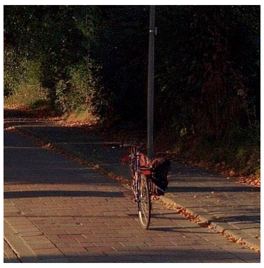
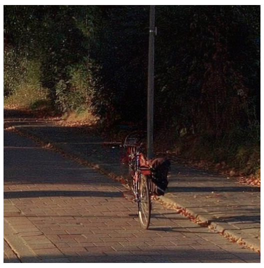
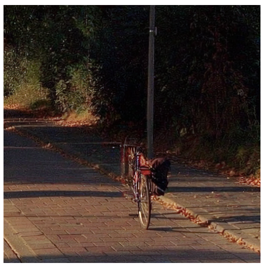
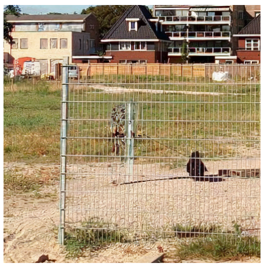
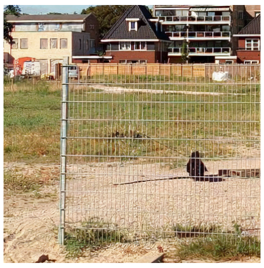
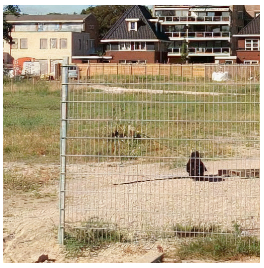
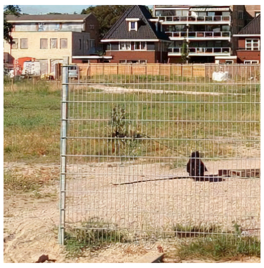
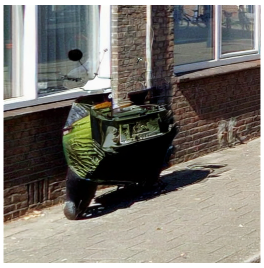
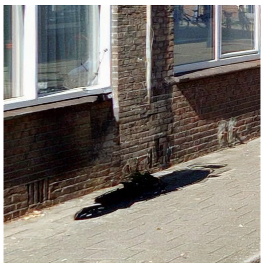
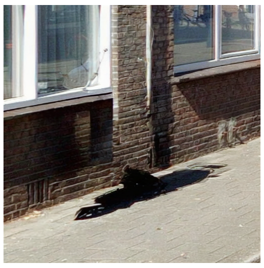
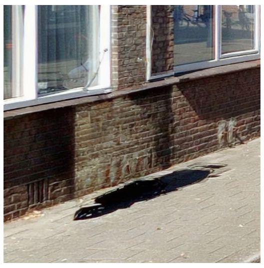
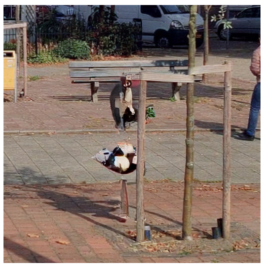
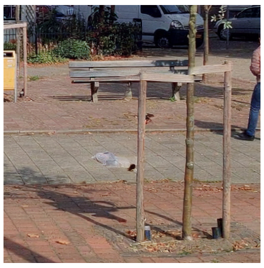
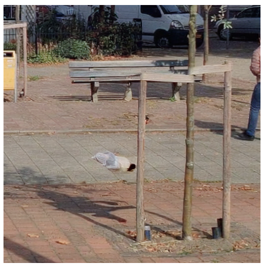
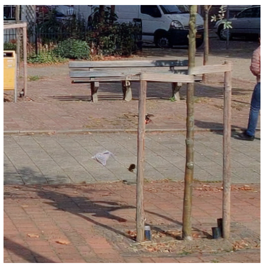
Figure 31. Qualitative user study - instructions and questions

Friends of the researchers and employees of Cyclomedia's R&D department were sent the survey. Prior to seeing the questions, participants were asked to self-report whether they had experience working in the field of photogammetry, computer vision, or related image-processing field. This made it possible to differentiate between computer vision experts and laypeople, and evaluate if there is any difference in preference between the two groups.

A preference score was calculated based on the survey, to quantify participants' inclination towards one model over the other. To calculate the score, we first assign a value of 1, -1, or 0 to each answer given by the respondents, depending on whether they prefer the first model, the second model, or have no preference, respectively. For each participant, the sum of the answers represents their individual preference score. An overall score is calculated as the average of all of the individual scores. The score ranges from -10 to 10, with values further from 0 indicating a stronger preference for one of the models. A score of 0 would mean the respondent is indifferent between the models.

4. Results

Table 2. Qualitative sample of inpainted results on test street-view scenes. A variety of objects are shown.

Baseline	**Fine-Tuned**	**Partial-Loss**	**Top-N**
			
			
			
			

The results of the experiments are presented in this section. Both quantitative and qualitative metrics are used for evaluation. The results show the **Fine-Tuned** model is able to produce more realistic results than the **Baseline**. The **Partial-Loss** and **Top-N** models are not able to conclusively outperform the **Fine-Tuned** model. In terms of human preference, the **Fine-Tuned** model is the most favourable. Sample outputs for each of the different models are shown in Table 2.

Table 3. Quantitative testing results of all models on the test set. Highest score for each measure is underlined. Parenthesis contain standard deviation.

Metric	**Baseline**	**Fine-tuned**	**Partial Loss**	**Top-N**
PSNR (higher is better)	26.114 (33.28)	<u>26.443</u> (32.17)	26.056 (31.39)	<u>26.443</u> (34.51)
SSIM (closer to 1 is better)	0.764 (0.016)	0.768 (0.015)	0.768 (0.015)	<u>0.772</u> (0.014)
FID (lower is better)	19.182	<u>17.287</u>	18.560	18.010
LPIPS (lower is better)	0.141	<u>0.134</u>	0.138	0.135

4.1 Quantitative Metrics

This section of this study encompasses the quantitative assessment of our methods' generative performance. To evaluate how well a model can perform inpainting, both the pixel-based and the perception-based quality of the generated images need to be considered. In order to do so, a multitude of objective metrics are employed. The four metrics that were used are PSNR 2.22, SSIM 2.23, LPIPS 2.24, and FID 2.26. Taken together, the metrics provide a means to quantitatively compare the models and their image generation capabilities.

The quantitative testing results for all methods are summarized in Table 3. Results were calculated on the test set. For each metric, the best result is underlined. Where applicable, the standard deviation is shown in parenthesis.

The results show that **Baseline** model is outperformed on every metric by all of the other models. When using the partial loss function, the performance is decreased in comparison to the **Fine-Tuned** model. Training on a carefully-curated subset of the data increases generative performance to the point of matching the **Fine-Tuned** model in terms of PSNR. The SSIM is also slightly improved, though the FID and LPIPS scores are lower, indicating a worse generated image.

4.2 Qualitative Metrics

In this section, the results of the survey are presented. The survey was conducted to examine user preferences for two different model settings: **Baseline** vs. **Fine-Tuned** and **Fine-Tuned** vs. **Top-N**. The study included a total of 28 participants, comprising of 16 laypeople and 12 experts. A paired t-test was performed on the preference score to evaluate if it is statistically significant against a null hypothesis assuming indifference between the models, and therefore a score of 0. The preference scores and t-tests are summarized in Table 4.

Table 4. Qualitative results, measured via user-study. Preference score calculated based on the responses of each participant.

Setting	Group	Preference Score		Statistical Significance	
		Mean	SD	T-Statistic	P-Value
Baseline vs **Fine-Tuned**	All	7.39	2.51	15.29	8.06×10^{-15}
	Experts	7.83	2.67	9.72	9.78×10^{-7}
	Laypeople	7.06	2.33	11.73	5.87×10^{-9}
Fine-Tuned vs **Top-N**	All	-4.25	2.29	-9.63	3.17×10^{-10}
	Experts	-4.0	2.0	-6.63	3.69×10^{-5}
	Laypeople	-4.44	2.47	-6.95	4.68×10^{-6}

For the first set of questions, all participants displayed a strong preference for the **Fine-Tuned** setting over the **Baseline** setting. The mean preference score for the **Fine-Tuned** setting was 7.39 (SD $= 2.51$). The T-test yielded a highly significant result, with a T-statistic of 15.29 and a p-value of 8.05×10^{-15}.

This holds within both the expert and layperson groups, with a mean preference score of 7.83 (SD $= 2.67$) and 7.06 (SD $= 2.33$) respectively. The T-test confirms the significance of these results, with a T-statistic of 9.72 and a p-value of 9.78×10^{-7} for the experts, and a T-statistic of 11.73 and a p-value of 5.87×10^{-9} for the laypeople. This indicates that the preference is stronger in the experts group than the laypeople group.

In the second set of questions, all participant groups favored the **Fine-Tuned** setting over the **Top-N** setting. The mean preference score for the second setting was -4.25 (SD $= 2.29$). Note the sign of the score is negative, showing a preference for the **Fine-Tuned** model. The T-test shows that the preference is significant, with a T-statistic of -9.63 and a p-value of 3.17×10^{-10}.

The preference holds within both subsets of the participants, with a mean preference score of -4.0 (SD $= 2.0$) among experts and -4.44 (SD $= 2.47$) among laypeople. The T-test shows both are statistically significant, with T-statistic of -6.63 and a p-value of 3.69×10^{-5} in the experts group and -6.95 and a p-value of 4.68×10^{-6} among laypeople.

These results demonstrate consistent preferences among all participant groups for the Fine-tuned setting in both the first and second set of questions. These findings provide insights into people's preferences for the different methods. They will be explored further in the subsequent "Discussion" section.

5. Discussion

5.1 Summary of Findings

In this thesis project, we aimed to investigate the effectiveness of deep learning diffusion models at inpainting and object removal for natural street-view scenes. We evaluated the impact of different factors, such as fine-tuning, modifications to the loss function, and data complexity, on the generative performance of the models. In this section, the main findings and implications of the experiments which were conducted are discussed.

The main research question of this thesis is:

- How effective are deep learning diffusion models at inpainting natural street-view scenes?

To answer this question, quantitative and qualitative experiments were conducted in order to evaluate the performance of latent diffusion models on the task of inpainting street-view scenes. We found that latent diffusion models are able to produce very realistic inpainting results, even in complex scenarios, however their performance is very sensitive to mask quality. There are additional factors outside of the generative performance of the model as well, which limit the effectiveness and usefulness of inpainting natural scenes in a realistic manner. For example, if inpainting a person, even if the mask were to perfectly capture the person, and the generation were to be indistinguishable from real, the shadow of the person would remain and the image would look very unnatural (see Figure 32). Externalities such as these are difficult to account for in the real world and were difficult to evaluate for within the scope of this research. Nonetheless, the results of this research shows that deep learning diffusion models are very effective at inpainting natural street-view scenes.

The first subquestion was:

- What is the impact of fine-tuning on the generative performance of the model in inpainting street-view scenes?

To answer this question, we compared the **Baseline** model, as created in Nichol and Dhariwal (2021), with a **Fine-Tuned** model, which was further trained on a subset of the Cyclomedia dataset. The Cyclomedia dataset comprises of high-quality natural street-view images of the Netherlands. The **Fine-Tuned** model was trained using synthetic masks,

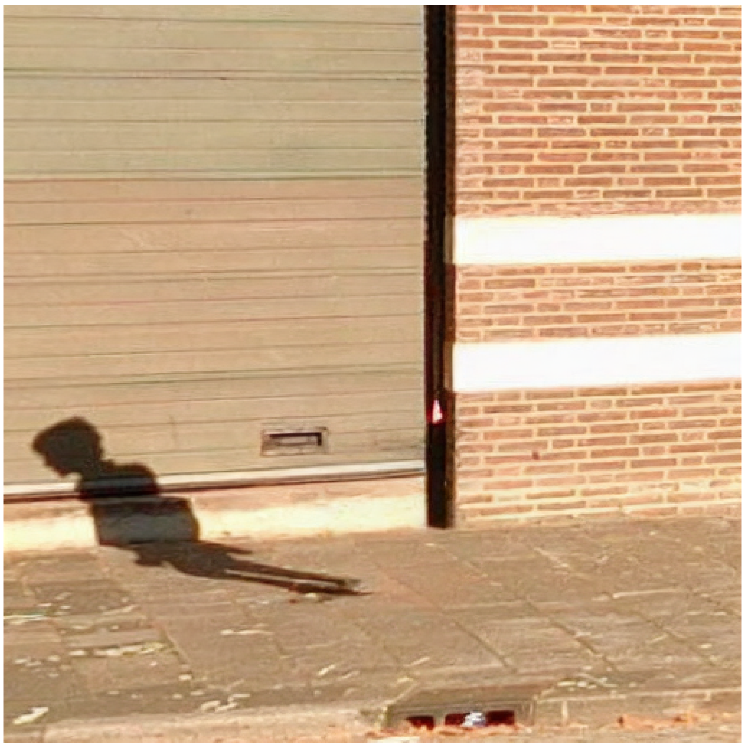


Figure 32. Example of an unnatural inpainting result due to remaining shadow. Despite a well-masked object, inpainted in a realistic manner, the result is insufficient because the mask remains. Image inpainted with the **Fine-Tuned** model.

generated following the approach of Suvorov et al. (2021).

The results showed that fine-tuning improved the generative performance of the model on the test set, as measured quantitatively by PSNR, SSIM, FID, and LPIPS metrics (see Table 3). The model improved on all of the above metrics against the baseline, but the improvements were limited in terms of absolute value. Despite the limited improvement in terms of quantitative measures, the qualitative results show a strong preference for the **Fine-Tuned** model against the baseline. The qualitative results show that both experts and laypeople have a strong, statistically significant preference for the images inpainted with the **Fine-Tuned** model over ones made with the **Baseline** model.

The discrepancy between the quantitative and qualitative metrics may be because the differences in quality are too small for the computational methods to detect, but significant enough to be noticeable by people. It may be that the results produced by the **Baseline** model are already so realistic and natural, that they are reaching a boundary of the quantitative metrics. Naturally, such a ceiling to detection would be higher for people.

The results indicate that fine-tuning is an effective way to improve the generative performance of diffusion models for inpainting street-view scenes. However, the improvements in generative performance come at the cost of training, which is a lengthy and very computationally-expensive prospect for diffusion models.

The second subquestion was:

- How does a loss function which focuses on specific parts of the image influence the generative performance of the model in inpainting street-view scenes?

To answer this question, we proposed a custom loss function for the latent diffusion model, the Partial Loss, which computes the reconstruction loss on only the masked parts of the image. The **Partial Loss** model was fine-tuned using the Partial Loss function, on the same data as the **Fine-Tuned** model. The results show that the **Partial Loss** model has a better FID and LPIPS score compared to the baseline, but worse in comparison to the **Fine-Tuned** model (see Table 3). These results suggest that modifying the loss function to focus on the inpainted region does not have a significant impact on the generative performance of diffusion models for inpainting street-view scenes.

The third subquestion was:

- How does the intricacy and complexity of the data used to fine-tune the model affect the generative performance in inpainting street-view scenes?

To answer this question, the images in the training set were sorted by their complexity and a model, called the **Top-N** model, was fine-tuned on the top 10% of the images. The quantitative results show that the **Top-N** model performed best in terms of SSIM and equally as well as the **Fine-Tuned** model in terms of PSNR, however it does not perform as well in FID and LPIPS (see Table 3). The qualitative survey also showed that both experts and laypeople have a significant preference for images inpainted with the **Fine-Tuned** model over the **Top-N** model. This is a surprising result, as the expected result would be that having exclusively images with high complexity in the fine-tuning dataset would result in more effective, faster training, and the model would be better able to produce higher-quality image inpaintings. However, the results show that this is not the case, and using more complex data to fine-tune the model does not improve the generative performance of diffusion models for inpainting street-view scenes.

One reason why this may be the case is that the images which the model is actually being used on are not necessarily the most complex ones. Because the test and training image crops are generated in the same way, we would expect them to have a similar distribution of complexity. When the model is being trained on only a certain part of the training crops, it is being trained on a sample which is out-of-distribution relative to the test set.

The fourth subquestion was:

- Is there a significant user preference when comparing the generative performance of the fine-tuned model against the baseline for inpainting street-view scenes?

To answer this question, we conducted a user study to collect preference scores from 28 participants (16 laypeople and 12 experts). The participants were asked to compare between an images inpainted by the **Baseline** model and the identical image inpainted by the **Fine-Tuned** model. Then, participants chose between one of the two images or indicated that they had no preference for either one (see an example question in Figure 31b). The results of the survey show that there is a statistically significant preference for the **Fine-Tuned** model (see Table 4). This result is true for both experts and laypeople. These results suggest that the **Fine-Tuned** model generates inpainted images that are more visually appealing and realistic to the human eye than the **Baseline** model, and that this preference is consistent across different levels of expertise.

The same survey was repeated, however the images compared between the **Fine-Tuned** and **Top-N** models. Users show a less-strong but still significant preference for the **Fine-Tuned** model. This indicates that the users found images generated by it to be more

The results showed that there was a consistent user preference for the **Fine-Tuned** model over the **Baseline** model, as measured by the mean preference score and the statistical significance (see Table 4). The mean preference score for the **Fine-Tuned** model was 7.39 for all participants, 7.83 for experts, and 7.06 for laypeople, indicating that the **Fine-Tuned** model was preferred by the majority of the participants. The preference score for the **Fine-Tuned** model was significantly higher than the preference score for the **Baseline** model, as indicated by the t-statistic and the p-value. These results suggest that the **Fine-Tuned** model generates inpainted images that are in some way more visually appealing, realistic, or otherwise higher quality to human perception than the **Baseline** model, and that this preference is consistent across different levels of expertise.

In summary, the experiments conducted in this thesis show that diffusion models are effective at inpainting natural street-view scenes, and that fine-tuning is an effective way of improving the generative performance of the models. We also showed that modifying the loss function to focus on the inpainted region does not have a significant impact on the generative performance, and that using more complex data to fine-tune the model does not improve the generative performance. Finally, a user survey showed that there is a consistent and significant user preference for the **Fine-Tuned** model over the **Baseline** model, and that this preference is independent of the level of expertise.

5.2 Limitations

There are limitations of the work in therms of the quantitative results. In spite of making use of a multitude of different complimentary metrics, following the approach of previous works in the field, the measures were not fully able to capture the differences between the models. The results of the tests are not significantly different enough, even though the qualitative user survey shows that there is a perceptible and appreciable difference between the model outputs.

It may be the case that computational evaluation metrics have an upper limit of sensitivity which has been reached with the advent of newer and better generative techniques. This is also the argument presented in Stein et al. (2023), in which FID is shown to not be able to fully capture the performance of diffusion models. Unfortunately this work could not use the custom inception network Stein et al. propose, because it was not available at the time of model evaluation.

Another limitation of the study to consider is that there are newer diffusion models, such as Stable Diffusion 2, which are shown to perform better out-of-the-box and be more efficient. In addition, there have been iterations on other architectures, notably GigaGAN (Kang et al., 2023), which is an updated GAN model, which have been shown to perform better and be more efficient that latent diffusion models. Due to the pace of the generative AI field, these methods could not be tested within this work.

When implementing the partial loss, the model is fine-tuned using the proposed loss, but most of its training was done with a loss that evaluates the entirety of the output. The results of this model were not improved, but that could change if the model is fully trained using the partial loss function. However, due to limitations of time and availability of compute resources, it was not feasible to test a model that was fully trained with the partial loss.

5.3 Future Work

For future research, it would be recommend to start by looking into the feasibility of using newer generative models, including ones based on the GAN architecture.

From our experience, unless there is the opportunity of doing an extremely long training run on very high-quality data, we would recommend focusing on mask quality initially. The quality of the object masks had a much greater impact on the efficacy of the models than fine-tuning did. Getting accurate masks is far from trivial - it is a task that should not be under-estimated. Additional research into how shadows could be masked may be the final step to creating a production-level system which can be used in a variety of real-world scenarios. This is because, no matter how accurate the mask and convincing the inpainting, the result cannot be convincing if the object's shadow is left.

Instead of fine-tuning, which is quite time-consuming, it may be useful to consider a "knowledge base" approach instead. Where the model is guided by and forced to reference a known set of images, through a process like Low-Rank Adaptation (LoRA, Hu et al. (2021)), textual inversion (Gal et al., 2022), or Dreambooth (Ruiz et al., 2023). The techniques above are applied to text-guided image generation, but it would be worth investigating their feasibility for inpainting. Finally, another approach to guidance could be based on the use of image semantic-maps.

6. Conclusion

This research investigated the generative performance of latent diffusion models (Rombach et al., 2022) in natural street-view scenes. Cyclomedia semantic object masks were refined using the SAM model (Kirillov et al., 2023) to produce high-quality and accurate object coverage for inpainting. Fine-tuning was evaluated for increasing the accuracy and quality of inpainting results, and we found it improved performance in terms of both computational metrics and human perception. A partial loss function was proposed, implemented, and evaluated. The performance when using a model fine-tuned with the partial loss did not show a significant improvement. Lastly, a feature-based measure of image complexity was used to evaluate the training data and a model was trained on a subset of the most complex training images. The performance of the model showed an improvement against baseline and the partial loss model, but it was not able to match the native fine-tuned model in terms of human perception.

Bibliography

Adler, J. and Lunz, S. (2018). Banach Wasserstein GAN.

Arjovsky, M. and Bottou, L. (2017). Towards Principled Methods for Training Generative Adversarial Networks. arXiv:1701.04862 [cs, stat].

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. arXiv:1701.07875 [cs, stat].

Ashikhmin, M. (2001). Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226. ACM.

Bank, D., Koenigstein, N., and Giryes, R. (2021). Autoencoders. arXiv:2003.05991 [cs, stat].

Baranchuk, D., Rubachev, I., Voynov, A., Khrulkov, V., and Babenko, A. (2022). Label-Efficient Semantic Segmentation with Diffusion Models. arXiv:2112.03126 [cs].

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):1–11.

Benny, Y., Galanti, T., Benaim, S., and Wolf, L. (2021). Evaluation Metrics for Conditional Image Generation. *Int J Comput Vis*, 129(5):1712–1731.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 417–424, USA. ACM Press/Addison-Wesley Publishing Co.

Bertalmio, M., Vese, L., Sapiro, G., and Osher, S. (2003). Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889. Conference Name: IEEE Transactions on Image Processing.

Bornard, R., Lecan, E., Laborelli, L., and Chenot, J.-H. (2002). Missing data correction in still images and image sequences. In *Proceedings of the tenth ACM international conference on Multimedia*, MULTIMEDIA '02, pages 355–361, New York, NY, USA. Association for Computing Machinery.

Burger, H. C., Schuler, C. J., and Harmeling, S. (2012). Image denoising: Can plain neural networks compete with BM3D? In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2392–2399. ISSN: 1063-6919.

Buyssens, P., Daisy, M., Tschumperle, D., and Lezoray, O. (2015). Exemplar-based Inpainting: Technical Review and new Heuristics for better Geometric Reconstructions. *IEEE Trans. on Image Process.*, pages 1–1.

Cao, F., Gousseau, Y., Masnou, S., and Pérez, P. (2011). Geometrically Guided Exemplar-Based Inpainting. *SIAM J. Imaging Sci.*, 4(4):1143–1179.

Cao, H., Tan, C., Gao, Z., Chen, G., Heng, P.-A., and Li, S. Z. (2022). A Survey on Generative Diffusion Model. arXiv:2209.02646 [cs].

Chen, D., Orekondy, T., and Fritz, M. (2021). GS-WGAN: A Gradient-Sanitized Approach for Learning Differentially Private Generators. arXiv:2006.08265 [cs, stat].

Chi, L., Jiang, B., and Mu, Y. (2020). Fast Fourier Convolution. In *Advances in Neural Information Processing Systems*, volume 33, pages 4479–4488. Curran Associates, Inc.

Chollet, F. (2016). Building Autoencoders in Keras.

Chung, H., Sim, B., and Ye, J. C. (2022). Come-Closer-Diffuse-Faster: Accelerating Conditional Diffusion Models for Inverse Problems through Stochastic Contraction. arXiv:2112.05146 [cs, eess, stat].

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). Describing Textures in the Wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, Columbus, OH, USA. IEEE.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. arXiv:1604.01685 [cs].

Cosentino, J. and Zhu, J. (2019). Generative Well-intentioned Networks. arXiv:1910.12481 [cs, stat] version: 1.

Criminisi, A., Perez, P., and Toyama, K. (2004). Region Filling and Object Removal by Exemplar-Based Image Inpainting. *IEEE Trans. on Image Process.*, 13(9):1200–1212.

Croitoru, F.-A., Hondru, V., Ionescu, R. T., and Shah, M. (2022). Diffusion Models in Vision: A Survey. arXiv:2209.04747 [cs].

Cross, G. R. and Jain, A. K. (1983). Markov random field texture models. *IEEE Trans Pattern Anal Mach Intell*, 5(1):25–39.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. ISSN: 1063-6919.

Dhariwal, P. and Nichol, A. (2021). Diffusion Models Beat GANs on Image Synthesis. arXiv:2105.05233 [cs, stat].

Doersch, C., Singh, S., Gupta, A., Sivic, J., and Efros, A. A. (2012). What makes Paris look like Paris? *ACM Trans. Graph.*, 31(4):1–9.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929 [cs].

Efros, A. and Leung, T. (1999). Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1033–1038 vol.2, Kerkyra, Greece. IEEE.

Ehret, T. and Arias, P. (2018). On the Convergence of PatchMatch and Its Variants. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1121–1129, Salt Lake City, UT. IEEE.

Eigen, D., Krishnan, D., and Fergus, R. (2013). Restoring an Image Taken through a Window Covered with Dirt or Rain. pages 633–640.

Esser, P., Rombach, R., and Ommer, B. (2021). Taming Transformers for High-Resolution Image Synthesis. arXiv:2012.09841 [cs].

Farajzadeh-Zanjani, M., Razavi-Far, R., Saif, M., and Palade, V. (2022). Generative Adversarial Networks: A Survey on Training, Variants, and Applications. In Razavi-Far, R., Ruiz-Garcia, A., Palade, V., and Schmidhuber, J., editors, *Generative Adversarial Learning: Architectures and Applications*, volume 217, pages 7–29. Springer International Publishing, Cham. Series Title: Intelligent Systems Reference Library.

Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G., and Cohen-Or, D. (2022). An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. arXiv:2208.01618 [cs].

Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A Neural Algorithm of Artistic Style. arXiv:1508.06576 [cs, q-bio].

Goodfellow, I. (2017). NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv:1701.00160 [cs].

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. arXiv:1406.2661 [cs, stat].

Guillemot, C., Turkan, M., Le Meur, O., and Ebdelli, M. (2013). Object removal and loss concealment using neighbor embedding methods. *Signal Processing: Image Communication*, 28(10):1405–1419.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Harshvardhan, G., Gourisaria, M. K., Pandey, M., and Rautaray, S. S. (2020). A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285.

He, K. and Sun, J. (2012). Statistics of Patch Offsets for Image Completion. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Computer Vision – ECCV 2012*, Lecture Notes in Computer Science, pages 16–29, Berlin, Heidelberg. Springer.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Heuvel, F. A. V. D., Beers, B. J., and Verwaal, R. G. (2011). Method and system for producing an image from a vehicle.

Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

Hirahara, D., Takaya, E., Kadowaki, M., Kobayashi, Y., and Ueda, T. (2021). Effect of the Pixel Interpolation Method for Downsampling Medical Images on Deep Learning Accuracy. *Journal of Computer and Communications*, 9(11):150–156. Number: 11 Publisher: Scientific Research Publishing.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. arXiv:2006.11239 [cs, stat].

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs].

Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2017). Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36(4).

Jaderberg, M., Simonyan, K., Zisserman, A., and kavukcuoglu, k. (2015). Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Jiang, Y., Xu, J., Yang, B., Xu, J., and Zhu, J. (2020). Image Inpainting Based on Generative Adversarial Networks. *IEEE Access*, 8:22884–22892. Conference Name: IEEE Access.

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 694–711, Cham. Springer International Publishing.

Kang, M., Zhu, J.-Y., Zhang, R., Park, J., Shechtman, E., Paris, S., and Park, T. (2023). Scaling up GANs for Text-to-Image Synthesis. arXiv:2303.05511 [cs].

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive Growing of GANs for Improved Quality, Stability, and Variation. arXiv:1710.10196 [cs, stat].

Karras, T., Laine, S., and Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405. ISSN: 2575-7075.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and Improving the Image Quality of StyleGAN. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116. ISSN: 2575-7075.

Kim, M., Kim, S., Kim, M., Bae, H.-J., Park, J.-W., and Kim, N. (2021). Realistic high-resolution lateral cephalometric radiography generated by progressive growing generative adversarial network and quality evaluations. *Scientific Reports*, 11.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs].

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *CoRR*.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. (2023). Segment Anything. arXiv:2304.02643 [cs].

Krizhevsky, A. and Hinton, G. (2009). Learning Multiple Layers of Features from Tiny Images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

Li Deng (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Process. Mag.*, 29(6):141–142.

Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., and Catanzaro, B. (2018). Image Inpainting for Irregular Holes Using Partial Convolutions. pages 85–100.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep Learning Face Attributes in the Wild. arXiv:1411.7766 [cs].

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. (2022). RePaint: Inpainting using Denoising Diffusion Probabilistic Models. arXiv:2201.09865 [cs].

Mairal, J., Elad, M., and Sapiro, G. (2008). Sparse Representation for Color Image Restoration. *IEEE Transactions on Image Processing*, 17(1):53–69. Conference Name: IEEE Transactions on Image Processing.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2016). Adversarial Autoencoders. arXiv:1511.05644 [cs].

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. (2022). SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations. arXiv:2108.01073 [cs].

Ng, A. (2011). Sparse autoencoder.

Nichol, A. and Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models. arXiv:2102.09672 [cs, stat].

Pascal, V., Hugo, L., Yoshua, B., and Pierre-Antoine, M. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1096–1103, New York, NY, USA. Association for Computing Machinery.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context Encoders: Feature Learning by Inpainting. pages 2536–2544.

Patil, B. H. and P.m, P. (2020). A Comprehensive Review on State-of-the-Art Image Inpainting Techniques. *Scalable Computing: Practice and Experience*, 21(2):265–276. Number: 2.

Perez, P., Gangnet, M., and Blake, A. (2004). PatchWorks: Example-Based Region Tiling for Image Editing.

Portilla, J., Strela, V., Wainwright, M., and Simoncelli, E. (2003). Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351. Conference Name: IEEE Transactions on Image Processing.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv:1511.06434 [cs].

Ren, J. S., Xu, L., Yan, Q., and Sun, W. (2015). Shepard Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. arXiv:1401.4082 [cs, stat].

Rojas, D. J. B., Fernandes, B. J. T., and Fernandes, S. M. M. (2020). A Review on Image Inpainting Techniques and Datasets. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 240–247. ISSN: 2377-5416.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752 [cs].

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241, Cham. Springer International Publishing.

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. (2023). DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation. arXiv:2208.12242 [cs].

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: explorations* *in the microstructure of cognition, vol. 1: foundations*, pages 318–362. MIT Press, Cambridge, MA, USA.

Sagong, M.-c., Shin, Y.-g., Kim, S.-w., Park, S., and Ko, S.-j. (2019). PEPSI : Fast Image Inpainting With Parallel Decoding Network. pages 11360–11368.

Saharia, C., Chan, W., Chang, H., Lee, C. A., Ho, J., Salimans, T., Fleet, D. J., and Norouzi, M. (2022). Palette: Image-to-Image Diffusion Models. arXiv:2111.05826 [cs].

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2021). Image Super-Resolution via Iterative Refinement. arXiv:2104.07636 [cs, eess].

Salem, N. (2021). A Survey on Various Image Inpainting Techniques.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. (2016). Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, ACM '68, pages 517–524, New York, NY, USA. Association for Computing Machinery.

Shin, Y.-G., Sagong, M.-C., Yeo, Y.-J., Kim, S.-W., and Ko, S.-J. (2021). PEPSI++: Fast and Lightweight Network for Image Inpainting. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):252–265. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics. arXiv:1503.03585 [cond-mat, q-bio, stat] version: 8.

Song, Y. and Ermon, S. (2020). Generative Modeling by Estimating Gradients of the Data Distribution. arXiv:1907.05600 [cs, stat].

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-Based Generative Modeling through Stochastic Differential Equations. arXiv:2011.13456 [cs, stat].

Starck, J.-L., Elad, M., and Donoho, D. (2005). Image decomposition via the combination of sparse representations and a variational approach. *IEEE Trans. on Image Process.*, 14(10):1570–1582.

Stein, G., Cresswell, J. C., Hosseinzadeh, R., Sui, Y., Ross, B. L., Villecroze, V., Liu, Z., Caterini, A. L., Taylor, J. E. T., and Loaiza-Ganem, G. (2023). Exposing flaws of generative model evaluation metrics and their unfair treatment of diffusion models. arXiv:2306.04675 [cs, stat].

Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., and Lempitsky, V. (2021). Resolution-robust Large Mask Inpainting with Fourier Convolutions. arXiv:2109.07161 [cs, eess].

Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., and Lempitsky, V. (2022). Resolution-Robust Large Mask Inpainting With Fourier Convolutions. pages 2149–2159.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. pages 2818–2826.

Tomczak, J. M. and Welling, M. (2017). Improving Variational Auto-Encoders using convex combination linear Inverse Autoregressive Flow. arXiv:1706.02326 [stat].

Tyleček, R. and Šára, R. (2013). Spatial Pattern Templates for Recognition of Objects with Regular Structure. In Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Weickert, J., Hein, M., and Schiele, B., editors, *Pattern Recognition*, volume 8142, pages 364–374. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Computer Science.

Uittenbogaard, R. (2018). Moving object detection and image inpainting in street-view imagery.

Vahdat, A., Kreis, K., and Kautz, J. (2021). Score-based Generative Modeling in Latent Space. arXiv:2106.05931 [cs, stat].

Venkatesan, R. and Li, B. (2017). *Convolutional Neural Networks in Visual Computing: A Concise Guide*. CRC Press, Boca Raton.

Wang, Z., She, Q., and Ward, T. E. (2021). Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy. *ACM Comput. Surv.*, 54(2):37:1–37:38.

Webb, G. I. (2010). Naïve Bayes. In Sammut, C. and Webb, G. I., editors, *Encyclopedia of Machine Learning*, pages 713–714. Springer US, Boston, MA.

Wei, L.-Y. and Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 479–488, USA. ACM Press/Addison-Wesley Publishing Co.

Weng, L. (2021). What are Diffusion Models? Section: posts.

Wong, A. and Orchard, J. (2008). A nonlocal-means approach to exemplar-based inpainting. In *2008 15th IEEE International Conference on Image Processing*, pages 2600–2603, San Diego, CA, USA. IEEE.

Xiang, H., Zou, Q., Nawaz, M. A., Huang, X., Zhang, F., and Yu, H. (2023). Deep learning for image inpainting: A survey. *Pattern Recognition*, 134:109046.

Xie, J., Xu, L., and Chen, E. (2012). Image Denoising and Inpainting with Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Xu, L., Ren, J. S., Liu, C., and Jia, J. (2014). Deep Convolutional Neural Network for Image Deconvolution. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Yan, Z., Li, X., Li, M., Zuo, W., and Shan, S. (2018). Shift-Net: Image Inpainting via Deep Feature Rearrangement. pages 1–17.

Yi, Z., Tang, Q., Azizi, S., Jang, D., and Xu, Z. (2020). Contextual Residual Aggregation for Ultra High-Resolution Image Inpainting. arXiv:2005.09704 [cs].

Yu, F. and Koltun, V. (2016). Multi-Scale Context Aggregation by Dilated Convolutions. arXiv:1511.07122 [cs].

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2016). LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. arXiv:1506.03365 [cs].

Yu, H. and Winkler, S. (2013). Image complexity and spatial information. In *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 12–17.

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. (2019). Free-Form Image Inpainting with Gated Convolution. arXiv:1806.03589 [cs].

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2018). Generative Image Inpainting With Contextual Attention. pages 5505–5514.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. arXiv:1311.2901 [cs].

Zeng, Y., Fu, J., Chao, H., and Guo, B. (2019). Learning Pyramid-Context Encoder Network for High-Quality Image Inpainting. pages 1486–1494.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595. ISSN: 2575-7075.

Zhang, X., Zhai, D., Li, T., Zhou, Y., and Lin, Y. (2023). Image inpainting based on deep learning: A review. *Information Fusion*, 90:74–94.

Zhao, S., Cui, J., Sheng, Y., Dong, Y., Liang, X., Chang, E. I., and Xu, Y. (2021). Large Scale Image Completion via Co-Modulated Generative Adversarial Networks.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2018). Places: A 10 Million Image Database for Scene Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(6):1452–1464.

Zhou, E. Y., Damiano, C., Wilder, J., and Walther, D. B. (2019). Measuring complexity of images using Multiscale Entropy. *Journal of Vision*, 19(10):96a.

A. Validation set results

Table 5. Quantitative testing results of all models on the validation set. Highest score for each measure is underlined.

Metric	**Baseline**	**Fine-tuned**	**Unweighted Partial Loss**	**Partial Loss**	**Partial L2 Loss**	**Top-N**
PSNR (higher is better)	25.784	26.228	22.779	26.124	26.125	26.012
SSIM (closer to 1 is better)	0.759	0.764	0.667	0.766	0.763	0.761
FID (lower is better)	39.988	36.131	75.331	38.823	39.412	39.643
LPIPS (lower is better)	0.141	0.134	0.263	0.136	0.138	0.138

B. Unweighted Partial Loss

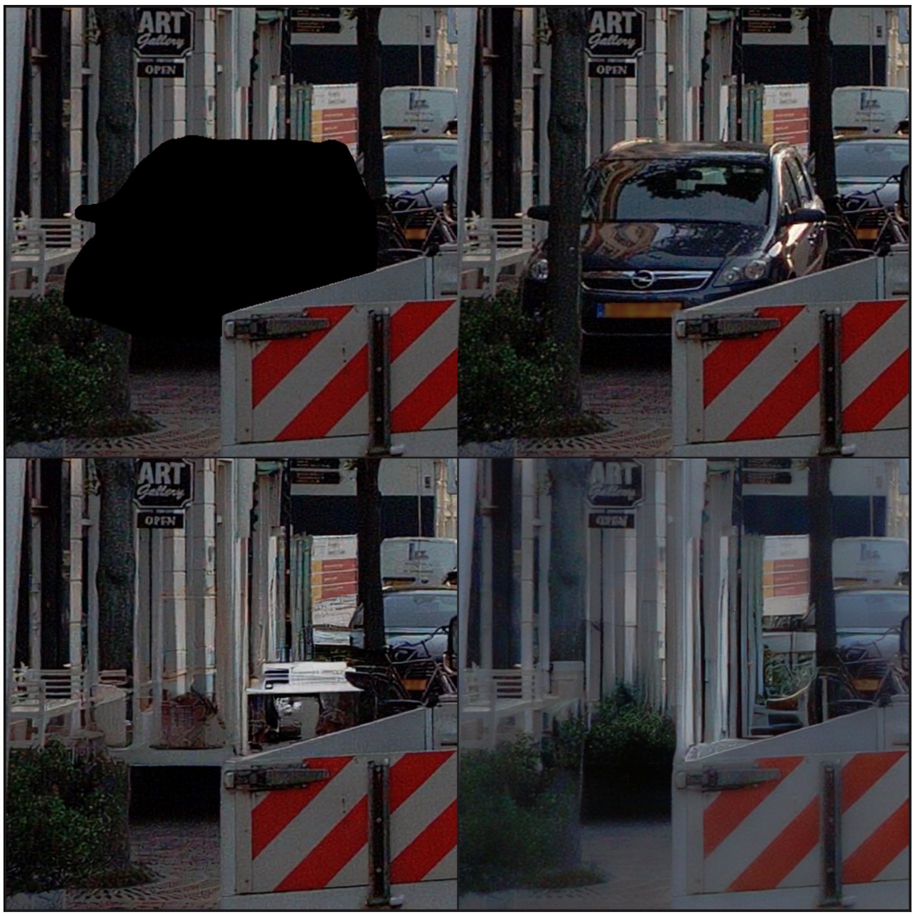


Figure 33. Top left: Masked image, mask was drawn by hand for testing purposes. Top right: Original image. Bottom left: **Baseline** model output. Bottom right: **Unweighted Partial Loss** model output. Note that the license plates were manually blurred for publication.