MASTERTHESIS
Moritz Tizian Wallrodt

Combining image inpainting and optical flow for integration of pathologies to deep learning-based endoscopic trainings simulators

FACULTY LIFE SCIENCES
Department Medical Engineering

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Moritz Tizian Wallrodt

Combining image inpainting and optical flow for integration of pathologies to deep learning-based endoscopic trainings simulators

Master thesis ∼ Submitted as part of master's exam
in the course of study Biomedical Engineering
at the department Medical Engineering
of the faculty Life Sciences
at the University of Applied Sciences Hamburg

First supervisor: Prof. Dr. Thomas Schiemann
Second supervisor: Dr. med. Rüdiger Schmitz, MSc.
Mentored by: Prof. Dr. René Werner

Submitted on: 22.05.2024

Moritz Tizian Wallrodt

Thema der Arbeit

Kombination von Inpainting-Algorithmen und Optical flow zur Integration von Pathologien in Deep Learning basierenden endoskopischen Trainingssimulatoren

Stichworte

Deep Learning, Endoskopische Trainingsimulatoren, Generative Adversarial Networks, Diffusion Models, Bildkomposition, Optischer Fluss

Kurzzusammenfassung

Endoskopische Simulatoren für das Training von Untersuchungen des Magen-Darm-Trakts werden üblicherweise aus Silikon hergestellt. Trotz einer qualitativ guten Nachbildung der Organe, bleibt das äußere Erscheinungsbild und das Trainingsgefühl weit von einer echten Untersuchung entfernt, wobei das Szenario bei jeder Untersuchung identisch bleibt. Aktuelle DeepLearning-Ansätze können Bilder von Trainingssimulatoren in synthetische Ansichten umwandeln, bisher jedoch meist nur für die Darstellung unauffälliger gesunder Schleimhaut. In dieser Arbeit wird das Potenzial von Generative Adversarial Nets (GANs) und Denoising Diffusion Probabilistic Models (DDPMs) zur Integration von Pathologien in synthetische Ansichten gesunder Schleimhaut und deren zeitkonsistente Verschiebung in kurzen Videosequenzen mittels Optical Flow untersucht. DDPMs liefern qualitativ hochwertige Ergebnisse beim Einzeichnen von Pathologien, haben jedoch eine signifikant längere Inferenzzeit als GAN-basierte Modelle. Der Optical Flow ermöglicht das Erstellen kurzer, konsistenter Videosequenzen, zeigt jedoch Limitationen bei der Anpassung der Lichtverhältnisse und Positionierung von Pathologien. Die Ergebnisse legen nahe, für zukünftige Methodenentwicklungen Pathologien auf Einzelbildebene einzuzeichnen und nur die dafür verwendeten Masken mit Optical Flow zu transformieren, statt - wie in dieser Arbeit - die Pathologie selbst. Ferner sollte die Implementierung der Modelle in Trainingssimulatoren weiter erforscht werden.

Moritz Tizian Wallrodt

Title of Thesis

Combining image inpainting and optical flow for integration of pathologies to deep learning-based endoscopic trainings simulators

Keywords

Deep Learning, Endoscopic training simulators, Generative Adversarial Networks, Diffusion models, Image inpainting, Optical flow

Abstract

Endoscopic simulators for training examinations of the gastrointestinal tract are usually made of silicone. Despite a good quality replica of the organs, the external appearance and training experience remains far away from a real examination, with the scenario remaining identical for each examination. Current deep learning approaches can convert images of training simulators into synthetic views, but so far mostly for the visualization of inconspicuous healthy mucosa. This work investigates the potential of Generative Adversarial Nets (GANs) and Denoising Diffusion Probabilistic Models (DDPMs) to integrate pathologies into synthetic views of healthy mucosa and their time-consistent displacement in short video sequences using optical flow. DDPMs provide high quality results when drawing pathologies, but have a longer inference time than GAN-based models. Optical flow enables the creation of short, consistent video sequences, but shows limitations in the adjustment of lighting conditions and positioning of pathologies. The results suggest that for future method developments, pathologies should be drawn at the single image level and only the masks used for this should be transformed with Optical Flow instead of the pathology itself, as done in this work. Furthermore, the integration of the models to training simulators should be further researched.

Contents

List of Figures

List of Tables

1 Introduction

Diseases of the gastrointestinal tract (GI-tract) lead to a considerable reduction in quality of life, as they not only cause pain or problems with digestion, but also represent an enormous psychological burden for those affected. In Germany alone, 2.5 million patients are treated in hospitals with diseases of the GI-tract, the liver, bile ducts, or pancreas. Whereby, over 60.000 patients die due to the diseases. Although the quality of medical care is constantly improving, it is estimated that the number of affected people will increase by 22% until 2032 [29].

Besides it is known that a good prevention and early detection is crucial regarding the progression of the disease [10, 21]. Although the progress in technology leads to invention which supports the daily routine in hospitals, very well-trained staff is still needed. Endoscopy is a procedure that uses a long flexible tube having a video camera at the end to investigate the GI-tract without any surgery. Especially in the field of endoscopy there are already algorithms that aims to detect pathologies (e.g. polyps) and thus improve the detection rate [18, 30]. However, these algorithms are trained for one particular disease and cannot cover all the complex pathologies which can occur. Hence, the the full diagnosis must still be made by a very well-trained doctor.

In order to prepare the medical staff for an endoscopy, there are a lot of training simulators (Figure 1.1), which allows to train the procedure without performing it on the real patient first.

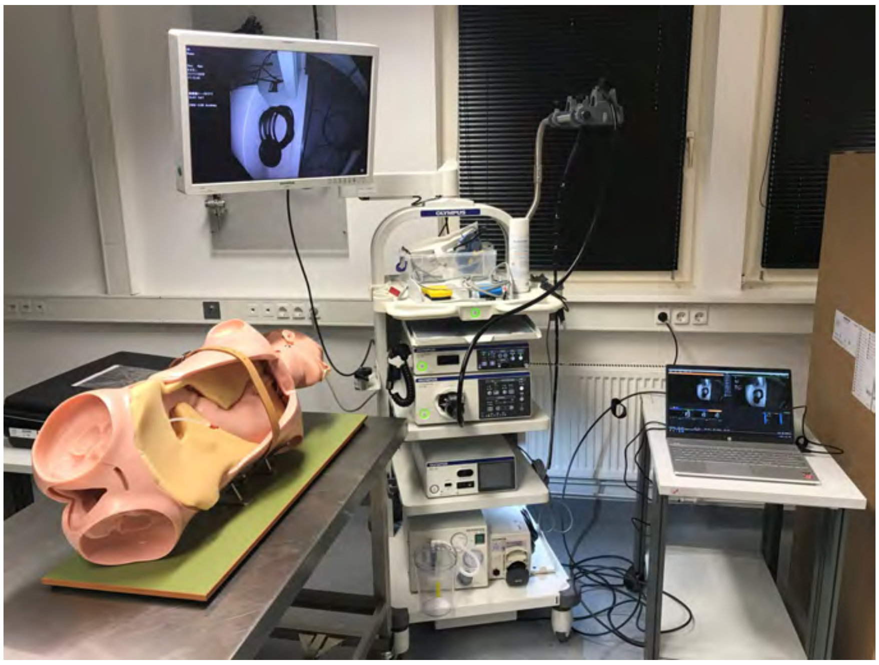


Figure 1.1: Endoscopic training simulator setup with dummy and endoscopy tower.

With these training simulators it is possible to train the handling with the endoscope, which is not easy at all. Also thanks to the morphological replication of a GI-tract, it is possible to get a feeling for proportions and structure. Nevertheless, the simulators are far away from being a perfect training simulator, since the visual appearance shows a lack of reality, as the dummy is made out of silicone (Figure 1.2).

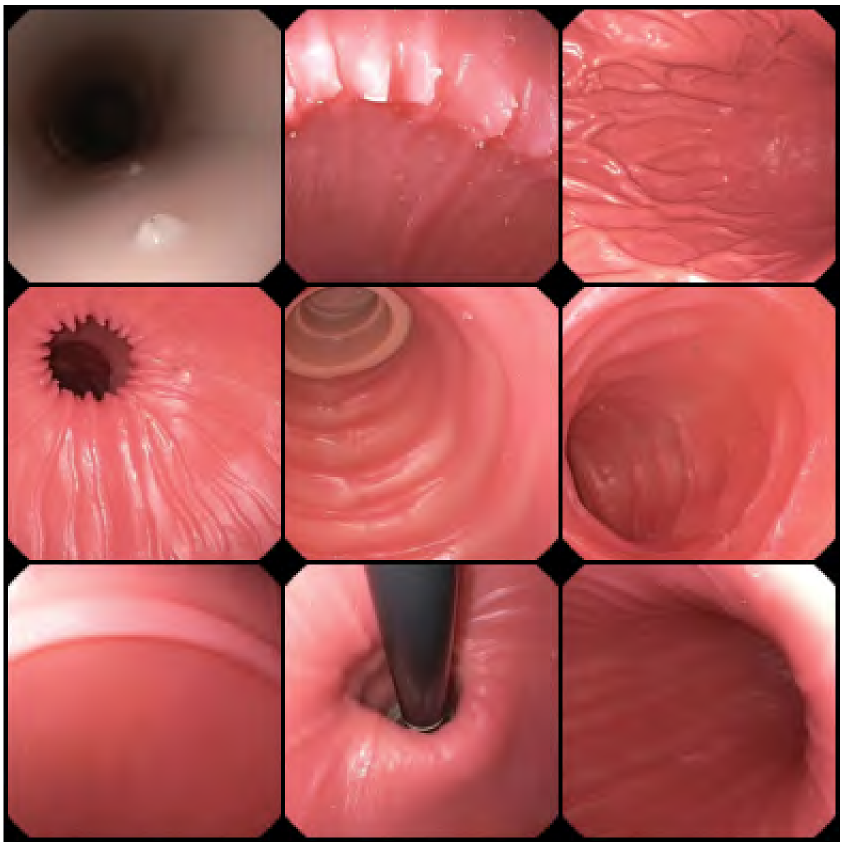


Figure 1.2: Example images of the inner view of a standard endoscopy simulator dummy.

This leads to two major drawbacks when training with these kind of simulators. The first one is that the trainees experience a rapid saturation of the learning curve, since the dummy looks always the same. After a few passes, the trainees will no longer pay attention to the environment and only focus on guiding the endoscope through the dummy. Thereby, it is precisely this ability to operate the endoscope and observe the surroundings at the same time that you want to train with these kind of simulators. The second big disadvantage is that the dummy can not cover different training scenarios, e.g, different pathologies and patients (Figure 1.3).

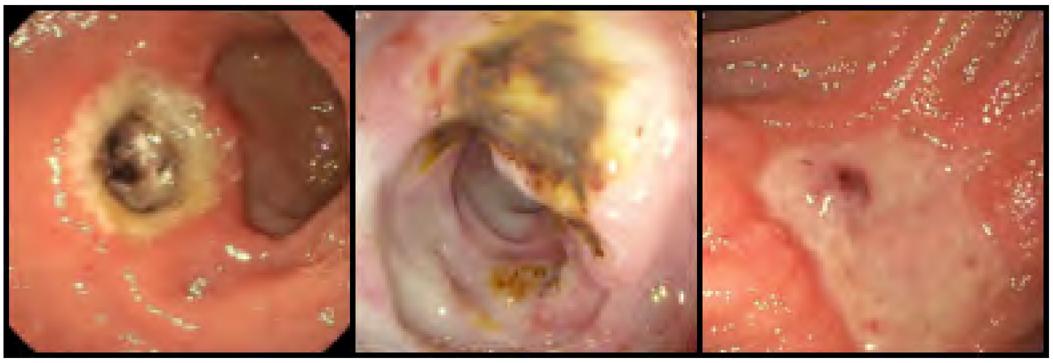


Figure 1.3: Examples of different pathologies that occurred during real examinations.

Although all three examples belong to the pattern of an ulcer, they may need different treatments. In order to be able to choose the correct treatment, the ulcer must be thoroughly examined. For this, it is necessary that the camera of the endoscope is steered in the correct position, which can be a challenging task and needs a lot of practice. At the same time, the medical staff must decide what the correct treatment is. Since in some cases it is necessary that action must be taken immediately with the help of the operating channel within the endoscope, the images cannot be evaluated after the examination. This example makes clear why it is extremely important that the medical staff is able to handle the endoscope perfectly while the main focus is on evaluating the surroundings. So far, these scenarios cannot be trained with conventional dummies.

Thankfully, improvements in the area of generative neural networks show promising approaches, which could help to overcome these shortcomings. Since a lot of work was already done considering the transformation of a dummy into a healthy patient [61], this work will focus on the integration of patholgies. For this, the potential of recent methods from the area of image inpainting and image composition will be investigated. In order to also ensure a good time consistency and thus a realistic appearance, the optical flow within a video sequence will be used to move the inpainted area according to the movement of the endoscope.

2 State of the Art

Since Ian Goodfellow introduced a novel neural network architecture named *Generative Adversarial Networks (GANs)* [20], generative networks become more and more popular. With these networks, it is possible to generate high quality and realistic looking images which belong to the data distribution of the given data set. In 2017, the architecture was extended such that an unpaired image-to-image transformation is possible and, because of the architecture and training concept, it was named cycleGAN [69]. Due to this work, images could now be transformed from one domain to another without loosing the underlying structure and content and the need of paired training samples. For example, a picture from a landscape in the summer could be transformed to the same landscape but with snow in the winter without having these pairs of images. Based on this, work was done to use these networks in the medical field. One of the first attempts to improve the realism of a endoscopic training simulator was done by a research group based in Heidelberg [15]. Thereby, the focus was on the temporal consistency between the transformed images. Since the original cycleGAN only considers single images during the transformation, a lack of temporal consistency was observed when transforming whole video sequences. The newly introduced tempCycleGAN considers two consecutive frames during the transformation, which leads to an improvement of the temporal consistency by showing less light flickering and artefacts. As the developed tempCycleGAN architecture allows the extension to consider more than two consecutive frames for one transformation, the influence of different frame set length was also investigated [61]. Within this work, the first attempt was made to integrate virtual pathologies by conditioning the architecture on an additional label. Although the results for the generated pathologies look good at single image level, the behaviour over a whole video sequence show a tremendous lack of realism and temporal consistency. Further research showed that the mentioned neural networks have a high potential to improve the medical training simulators, but the temporal consistency remains a challenging task. Instead of using pixel-wise constraints during the training process a new attempt tried to use the information contained in the optical flow of consecutive frames. This motion-guided architecture is called Mocycle-GAN and showed promising results [8]. Within a study, the tempCycleGAN and the Mocycle-GAN were compared to improve a endoscopic training simulator. The results revealed that both architecture show major improvements considering both the realism and the temporal consistency compared to the standard cycleGAN model. As the Mocycle-GAN is more lightweight than the tempcycleGAN it has a lower inference time and thus a better real time capability for the subsequent applications. However, the Mocycle-GAN only uses the optical flow implicitly which still can not solve the temporal consistency issues regarding the pathologies [53].

3 Fundamentals

3.1 Deep Learning

The terms artificial intelligence (AI), machine learning and deep learning are often used interchangeably, but they represent different approaches and techniques. AI is the overarching term that aims to solve problems through intelligent behavior. Rule-based approaches, a simple form of AI, make decisions according to predefined rules. However, these approaches are limited to simple problems. For more complex problems, machine learning (ML) approaches are used, where algorithms are applied to data sets to identify relevant structures and patterns. These algorithms can classify data or predict future values, but they come up against limits if the problem is highly non-linear. Deep learning, an advanced form of machine learning, uses deep neural networks to independently learn rules for problems within a given data set. This approach is able to learn highly non-linear patterns whereby it can be used for complex tasks such as autonomous driving and speech recognition. A classic example where machine learning techniques reach their limits is the description of a logical gate functions. Even if the task does not seem extraordinarily complex at first glance, modeling this using a normal machine learning algorithm such as linear regression is only possible to a limited extent, if at all [19]. Figure 3.1 shows the output signal of an XOR gate as a function of two input signals x_1 and x_2.

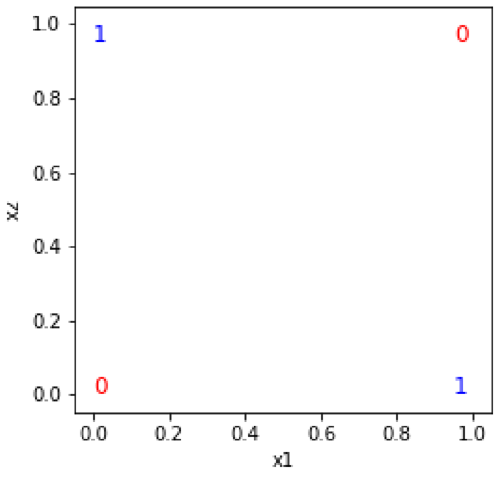


Figure 3.1: Outputs of a XOR gate depending on two inputs x_1 and x_2.

The visualization shows that there is no linear separation of the possible outputs that exactly describes the behavior of the gate. The methods from the field of deep learning are, however, able to solve this problem. Therefore, so-called non-linear activation functions φ (Figure 3.2) are important components of a neural network in order to introduce non-linearity allowing it to learn and represent complex relationships in data.

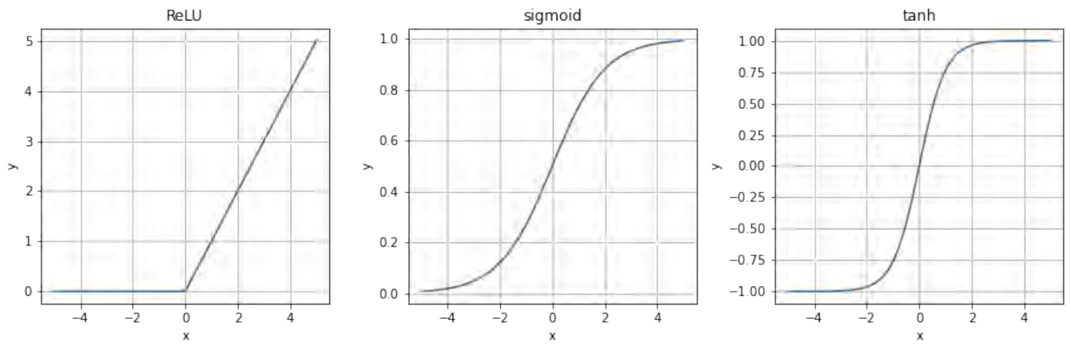


Figure 3.2: Visualization of the activation functions ReLU, sigmoid and tangent hyperbolicus

In general, a deep neural network consists of many layers. The first layer among other things specifies the shape of the input, which is expected by the neural network. This is followed by many hidden layers and concludes with an output layer, which holds the final result. The shape of the input layer is adapted to the data that is to be processed. For example, the shape of the first layer whose input is a gray-scaled picture with a size of 64x64 pixels could be a vector $\mathbf{x}$ of $64^2 = 4096$ values representing the single pixel values. The size of the output layer depends on the tasks. In case the neural network should classify if a given input belongs to a certain class or not, a binary output is sufficient. In case the given input should be assigned to one class out of ten, the output should be a vector. Figure 3.3 show an example of a neural network with three inputs, two hidden layers and two output neurons.

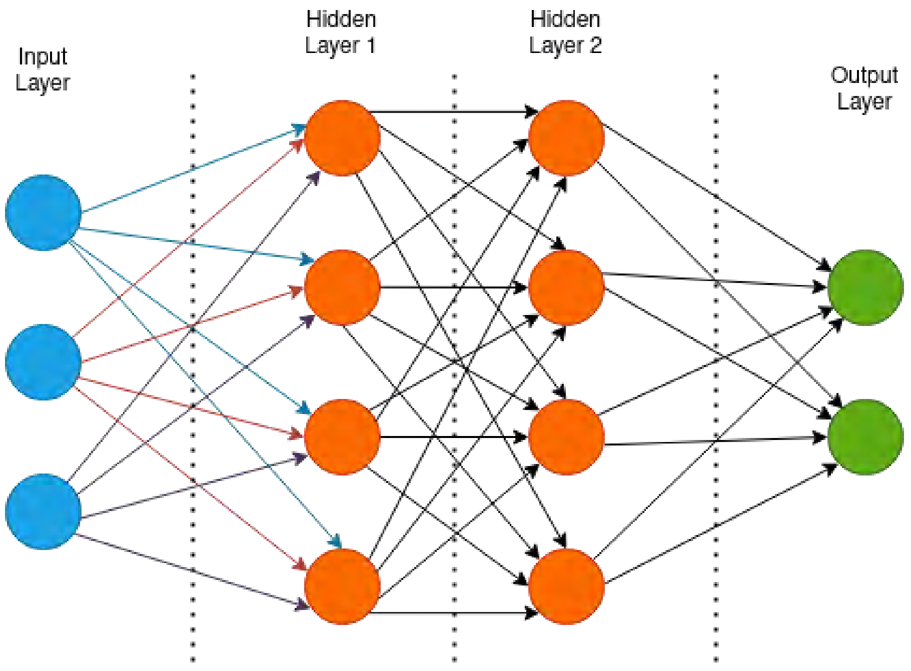


Figure 3.3: Structure of a neural network

A perceptron is the simplest form of a neural network model. It consists of a single layer of input units (neurons), each connected to an output unit through weighted connections. It takes multiple input values, each multiplied by a corresponding weight, and sums them up. This sum is then passed through an activation function to produce the output of the perceptron. Since real world data can contain systematic errors or distortions the layers can additionally have a bias term b to better fit the pattern within the data set. The whole structure of the perceptron is shown in Figure 3.4.

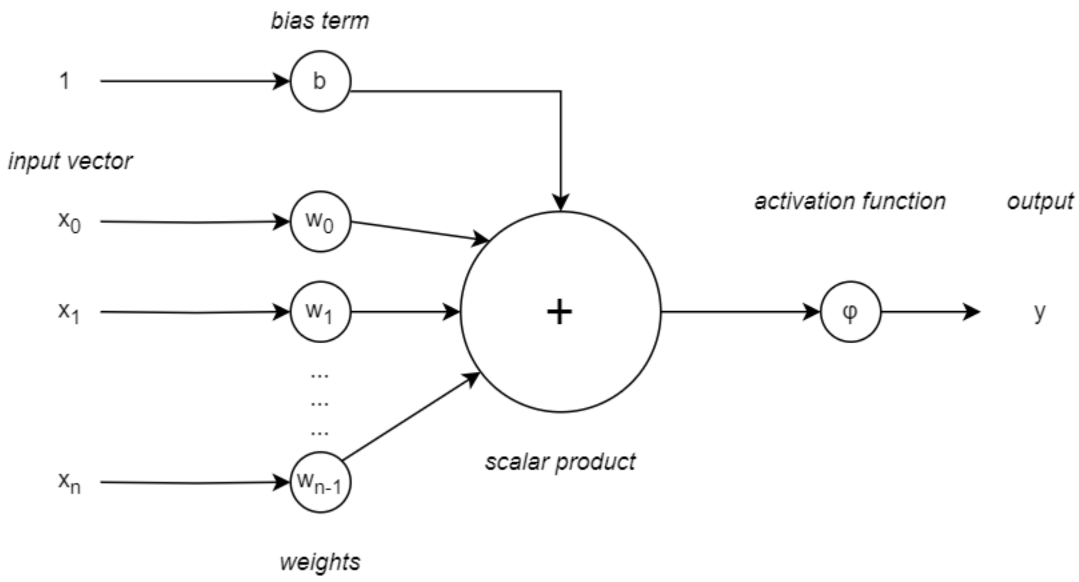


Figure 3.4: Structure of the perceptron

In order to optimize the weights θ of a neural network during training, a target function $J(\theta)$ (also known as loss function) must be defined beforehand. This function calculates the deviation between the expected outcome y and the predicted output of the neural network $\hat{y}$. Since a neural network can process multiple inputs (batch) at the same time, the loss term is divided by the number of outputs N in order to smoothen the error and thus stabilize the training. The choice of the loss function must be made carefully, as otherwise the neural network will not pursue a suitable goal and therefore cannot be expected to perform well regarding the given problem. For example, if the neural network should learn a regression, the mean squared error could be a suitable loss function.

$$J(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} (y_{\mathrm{i}} - \hat{y}_{\mathrm{i}})^2 \tag{3.1}$$

In case the problem is a binary classification the binary cross entropy can be used.

$$J(\theta) = -\frac{1}{N}\sum_{i=0}^{N-1} y_{\mathrm{i}} \cdot \log(p(y_{\mathrm{i}})) + (1 - y_{\mathrm{i}}) \cdot \log(1 - p(y_{\mathrm{i}})) \tag{3.2}$$

Independently of the chosen loss function, the resulting value is then used to optimize the weights of the neural network. Although, the input is passed from the first layer to the last layer (feed forward network), the weights are adjusted beginning at the output layer going to the input layer. This is called backpropagation [33], which is the key principle of a neural network. In this process, the influence of each individual weight on the error must be determined using the partial derivative,

$$\Delta w_{kj}^{l} = -\alpha \frac{\partial J(\theta)}{\partial w^{\mathrm{l}}{}_{\mathrm{kj}}} \tag{3.3}$$

where $J(\theta)$ is the error determined using the loss function and w_{kj}^{l} is the k-th neuron of a layer l, which is connected to the j-th neuron of the subsequent layer. The weight w_{kj}^{l} is adjusted in the opposite direction of the gradient Δw_{kj}^{l} and is additionally weighted with a factor α, which is referred to as the learning rate. The learning rate is also one of the hyperparameters for the training and must also be chosen carefully. Since the main idea of the whole optimisation process is to find a global minimum, a small learning rate tends to get stuck in a local minima. On the other hand a large learning rate can miss an actual global minima. There are different algorithms for the optimizer which differ in how they update the model parameters and how they adapt the learning rate during training. The choice of the optimizer can significantly impact the convergence speed and the final performance of the neural network [13].

3.2 Basic architectures

Over the years, many different architectures became popular. They vary widely in complexity, design, and functionality, each tailored to address specific tasks and challenges. As already mentioned in the previous chapter, one of the fundamental architectures is the feed forward neural network. It consist of layers with neurons where information flows in one direction, from the input layer through hidden layers to the output layer.

These networks excel in tasks like regression and classification, with popular variations such as deep neural networks (DNNs), which stack multiple hidden layers for increased representational power.

Another architecture is the recurrent neural network (RNN), designed to handle sequential data by incorporating feedback loops (Figure 3.5) within the network, allowing it to maintain an internal state or memory. RNNs are well-suited for tasks such as time series prediction, language modeling, and speech recognition, thanks to their ability to capture temporal dependencies [22, 52].

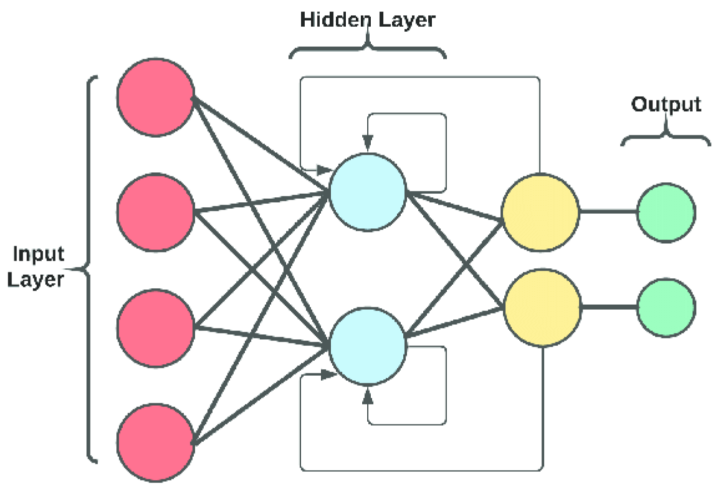


Figure 3.5: Recurrent neural network architecture diagram [2].

Especially in the field of computer vision and image processing, convolutional neural networks (CNNs) have emerged as a cornerstone in the field of computer vision. CNNs leverage specialized layers like convolutional layers to automatically extract hierarchical features from input images. This done by performing a convolution to an input (Figure 3.6) with different kernels which are learned by the network. These architectures have achieved remarkable success in image classification, object detection, and image segmentation tasks, often outperforming traditional computer vision techniques [19, 36].

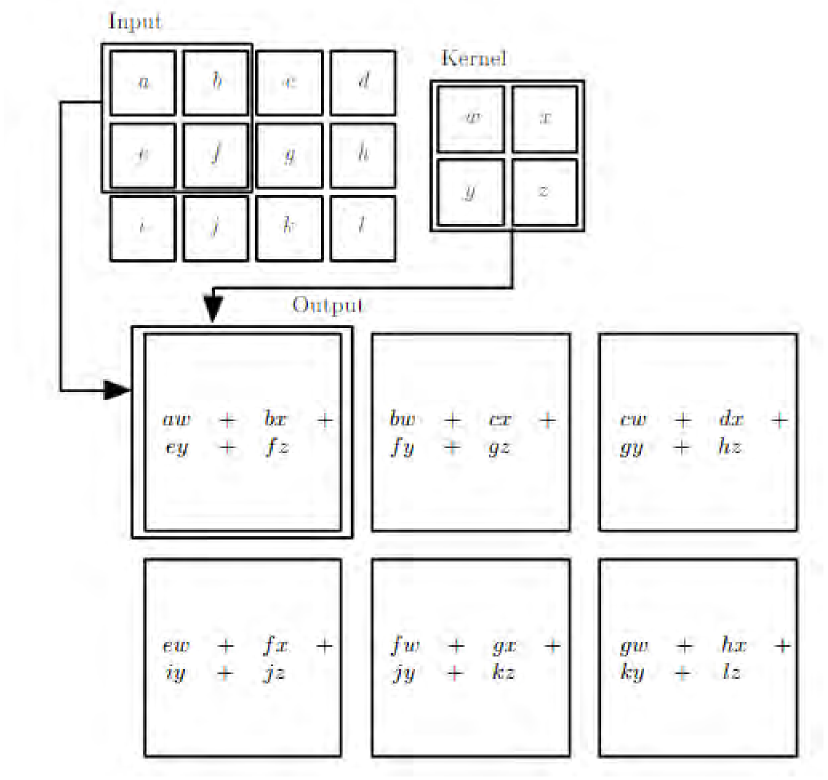


Figure 3.6: 2-D convolution of an 4x3 input with a 2x2 kernel. [19]

As the field continues to evolve, novel architectures and hybrid models emerge, combining elements from different paradigms to tackle increasingly complex challenges. As an example, an auto-encoder is a type of neural network architecture designed for unsupervised learning and dimensionality reduction. It consists of an encoder network that compresses the input data into a lower-dimensional representation z, and a decoder network that reconstructs the original input from this compressed representation (Figure 3.7). By learning to reconstruct the input data accurately, auto-encoders capture meaningful features and patterns, making them useful for tasks like data denoising, anomaly detection, and feature learning [59].

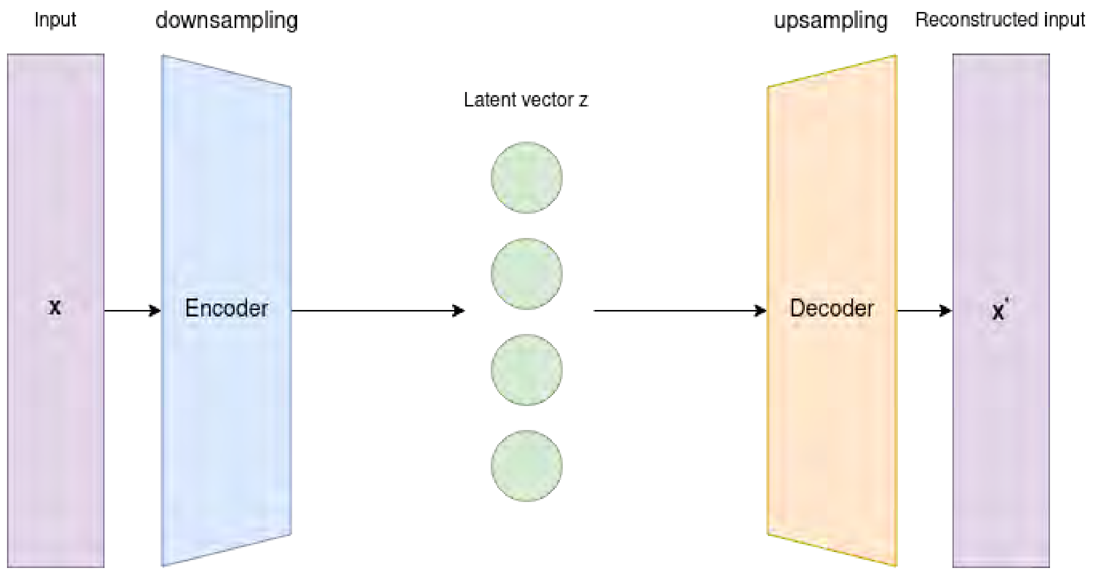


Figure 3.7: Structure of an autoencoder architecture.

Another widely used architecture is the U-Net. It is a convolutional neural network architecture designed for semantic segmentation tasks in computer vision. Its U-shaped architecture consists of a encoder and decoder part, which are connected via skip connections to preserve spatial information (Figure 3.8). By concatenating feature maps from the encoder to the decoder, skip connections enable the decoder to access both low-level and high-level features [50].

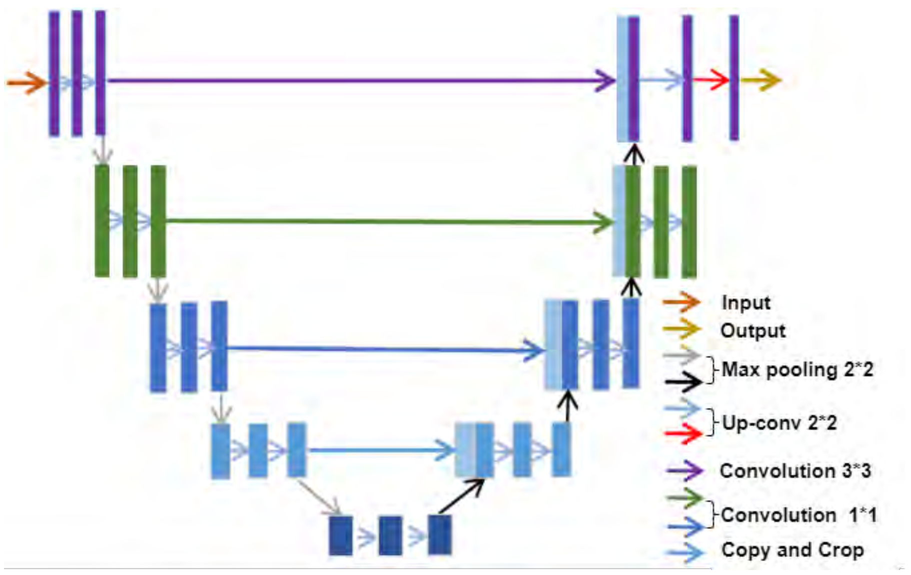


Figure 3.8: Structure of an U-Net architecture (changed according to [14]).

U-Net is widely employed in medical image segmentation and other fields where detailed pixel-level classification is required [27].

3.3 Transformer

In 2017, transformer networks [60] have been published and since then they have emerged as a groundbreaking architecture in the field of deep learning, revolutionizing natural language processing, computer vision, and various other domains. They have become the cornerstone of many state-of-the-art models due to their ability to capture long-range dependencies and process sequential data efficiently [24]. The so called self-attention mechanism is the key feature of the transformer architecture, enabling the model to weight different parts of the input sequence differently during processing. Thereby, each element of the input sequence is called a token. This mechanism allows transformers to consider the relationships between all tokens in a sequence simultaneously which helps to overcome the limitations of traditional network architectures like CNNs when it comes to capturing long-range dependencies.

In general, the first step within a transformer network is to associate each token of the input with a high dimensional vector. This is done in a so called embedding layer. Within this high dimensional space, tokens with a closer relationship are located in the same area. But the problem is that a network initially has no idea about the meaning of the single tokens, which are represented as numbers. As an example the two phrases "A neural network model" and "A fashion model" both contain the same word "model" but the context is completely different. In case the transformer should generate some text based on one of these phrases, all words within the sentence play a crucial role and therefore must be considered carefully in order to end up with a meaningful text. So that the network is able to learn the meaning the self-attention mechanism is used. It computes a set of attention scores for each element in the input sequence based on its relationships with all other elements. These scores are then used to weight the importance of each element in the context of the entire sequence, which allows transformers to capture contextual information effectively. In order to calculate these attention scores, three different components are considered, namely keys (K), queries (Q) and values (V)

The queries Q are vectors with a lower dimension than the embedding layer. They a calculated by multiplying a matrix W_Q with the embeddings $\vec{E}$,

$$\vec{Q_i} = W_Q \odot \vec{E_i} \tag{3.4}$$

whereby the matrix W_Q contains learnable parameter of the transformer network and the index i corresponded to the position of a token within a sequence, i.e the query for the first token is called $\vec{Q_1}$. In the same way, further vectors called keys K are calculated using another matrix W_k

$$\vec{K_i} = W_K \odot \vec{E_i} \tag{3.5}$$

After all queries and keys for each token are calculated, the dot product is applied between each query and key pair. The resulting number is a measure how close the tokens are located within the high dimensional space. Since these numbers can reach from $-\infty$ to $+\infty$, the values are normalised using the softmax function, which ensures that the values are scaled between 0 and 1.

The third component, the values V, are vectors associated with each token in the input sequence, similar to keys. However, unlike keys, values are not directly involved in computing attention scores. Instead, they are used to compute the weighted sum that forms the context vector $\vec{V_i}$ in the attention mechanism.

$$\vec{V_i} = W_V \odot \vec{E_i} \tag{3.6}$$

All these components are glued together, which forms the self-attention mechanism (Equation 3.7). The factor $\frac{1}{\sqrt{d_k}}$ normalizes the values depending on the chosen dimensionalty of the vectors and ensuring better numerical stability [60].

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.7}$$

To sum it up once again, queries represent the information the model is focusing on, keys capture information about different parts of the input sequence, and values represent the content that will be attended to based on the computed attention scores. Together, these components allow the transformer network to effectively model relationships and dependencies within the input sequence. One of these pairs of W_Q, W_K and W_V are associated with one layer which are called attention heads. Calculating multiple matrices for the same pairs of queries, keys and values is called multi-head attention (Figure 3.9).

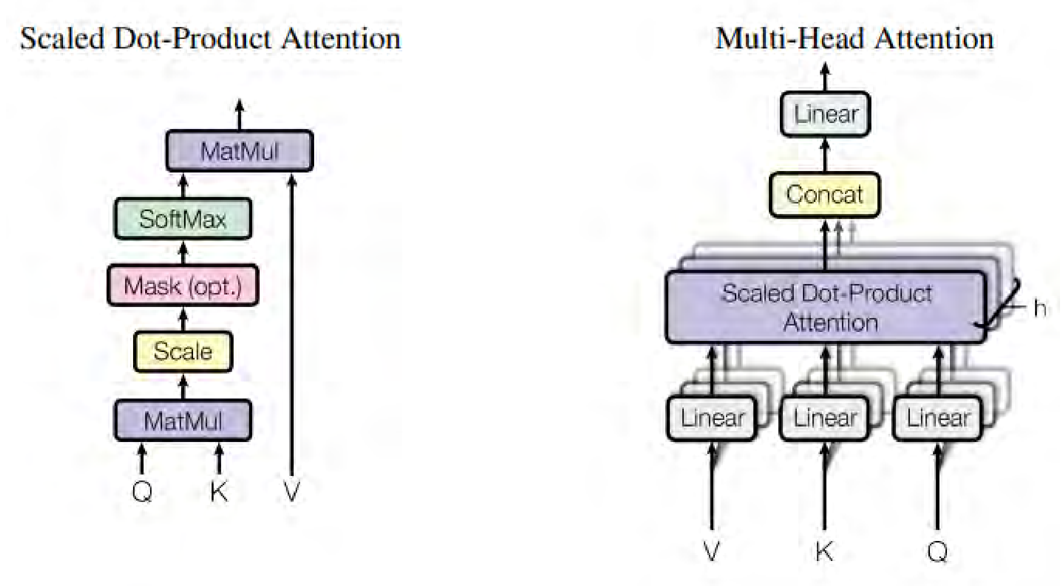


Figure 3.9: Scaled Dot-Product Attention (left). Multi-Head Attention consists of several attention layers running in parallel (right) [60].

These self-attention layers can now be used together with other common layer types, building the final architecture of a transformer network (Figure 3.10).

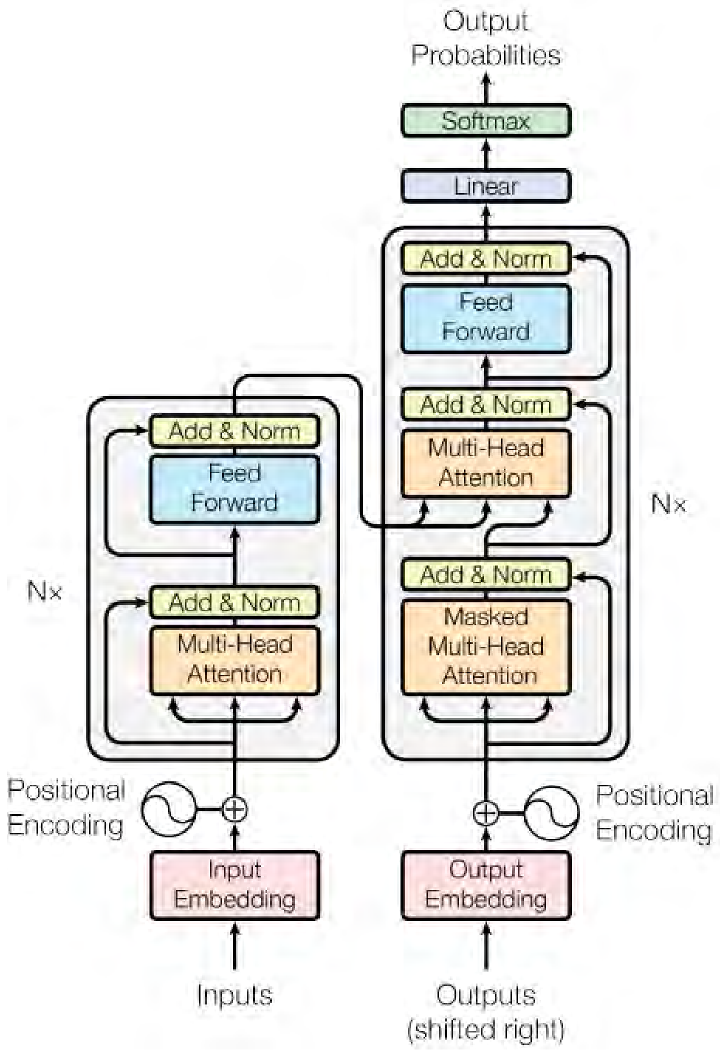


Figure 3.10: Architecture of a transformer network [60].

3.4 Generative Neural Networks

When Ian Goodfellow and his colleagues published his research on generative adversarial networks [20] in 2014, the interest and potential in generative networks increased strongly. His idea was to use a network architecture with two networks which compete against each other. One network is called the generator and the other one discriminator. The task of the generator is to produce outputs which belongs to the distribution of a given data set, while the discriminator should learn to distinguish between the real data and the fake outputs produced by the generator.

The training of a Generative Adversarial Network (GAN) involves two main loss functions. From the perspective of the generator, the goal is to minimize the probability that the discriminator assigns the generated data to fake. This is commonly expressed using the negative log likelihood of the discriminator predicting the generated samples as real.

$$\mathcal{L}_G = -log(D(G(z))) \tag{3.8}$$

The input z of the generator is a vector containing random noise sampled from a normal distribution. Since the task of the discriminator is to distinguish between real and synthetic data, the corresponding loss function consists of two terms.

$$\mathcal{L}_D = -log(D(x)) - log(1 - D(G(z))) \tag{3.9}$$

Here x is the real data sampled from the data set and $G(z)$ is again the output generated by the generator. Whereby, the generator can be trained on different data like speech signals or images. The fact that these two networks compete against each other makes the training process highly prone to be unstable (mode collapse). That is why a lot of work was done finding solutions to avoid this. One idea is to change the network architecture. For example, the Patch-GAN approach uses a discriminator which returns a patch instead of one single value which indicates if the input belongs to the real or fake distribution. Each value in this patch indicates the probability that the corresponding area of the input is real or fake. This is particularly used when images should be generated. Thus, the generator gets a more detailed feedback which area of the generated images has to be optimized. With this proposed architecture, considerable improvements regarding undesired artifacts and noise in the generated outputs were achieved [11].

3.5 Image-to-Image Transformation

Image-to-image transformation involves converting an input image from one domain to another, catering to various applications. For instance, one may want to enhance a low-resolution image to a high-resolution one without resorting to basic resizing and interpolation techniques. In this scenario, the domains are low-resolution and high-resolution image, where having a data set with paired examples or an actual ground truth is highly advantageous. However, often the available data lacks such paired samples, and at times, it is even impossible to obtain them. When the task involves transferring the style of one image to that of another domain, the images required may not be accessible in both styles. For example, the objective might be to alter the style of a generic image to resemble the work of a renowned artist. Despite possibly having numerous images of landscapes and works by a famous artist, paired examples of these may not exist.

For the task of unpaired image-to-image transformation, a powerful deep learning architecture named cycleGAN (Cycle-Consistent Generative Adversarial Network) was introduced in 2017 [69]. Since the goal is to translate a given image from one domain into another without losing semantic features, the approach makes use of a special training procedure. Whereby, the architecture (Figure 3.11) consists of two generators and two discriminators (one for each domain).

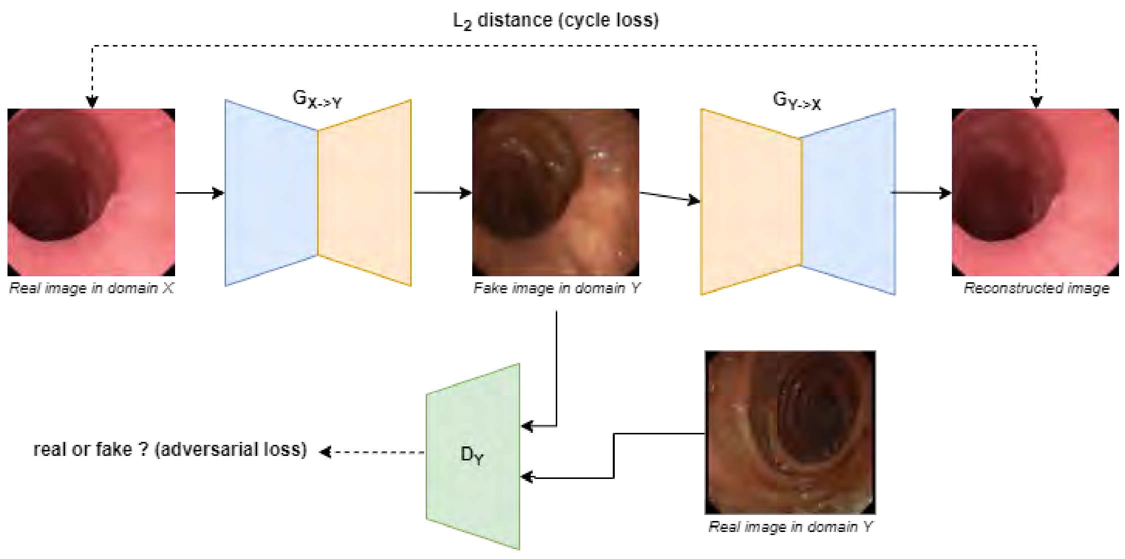


Figure 3.11: Architecture of the cycleGAN

During training, an image from the domain X is translated to the domain Y with the first generator $G_{X->Y}$ such that it belongs to the distribution of the target domain. As with the original GAN approach, a corresponding discriminator network D_X or D_Y, respectively, is trained to distinguish between real and generated data.

$$\mathcal{L}_{\text{adv}}(G_{\text{X}}, D_{\text{X}}, X) = \mathbb{E}_{\text{x}\sim\text{p}_{\text{data}}(\text{x})}[\log D_{\text{X}}(x)] + \mathbb{E}_{\text{y}\sim\text{p}_{\text{data}}(\text{y})}[\log(1 - D_{\text{X}}(G_{\text{X}}(y))] \tag{3.10}$$

To ensure the preservation of the semantic features, the translated image will be transformed back into the original domain using the second generator $G_{Y->X}$. By calculating the pixel-wise distance between original image from domain X and the reconstructed image (cycle loss), the generator is forced to keep this features. Since the second generator $G_{Y->X}$ needs to be also trained, the same procedure is done but with respect to domain Y.

$$\mathcal{L}_{\text{cycle}}(G_{\text{Y}}, G_{\text{X}}) = \mathbb{E}_{\mathbf{x}\sim\text{p}_{\text{data}}(\mathbf{x})}[||G_{\text{X}}(G_{\text{Y}}(x)) - x||_1] + \mathbb{E}_{\mathbf{y}\sim\text{p}_{\text{data}}(\mathbf{y})}[||G_{\text{Y}}(G_{\text{X}}(y) - y)||_1 \tag{3.11}$$

The authors of the cycleGAN architecture observed an improved of the results when using an additional loss function named *identity loss*. Therefore, the generator receives also images from the actual target domain. Since these images already belong to the correct distribution, the generator should not change them.

$$\mathcal{L}_{\text{identity}}(G_{\text{X}}, G_{\text{Y}}) = \mathbb{E}_{\mathbf{y}\sim\text{p}_{\text{data}}(\mathbf{y})}[||G_{\text{Y}}(y) - y||_1] + \mathbb{E}_{\mathbf{x}\sim\text{p}_{\text{data}}(\mathbf{x})}[||G_{\text{X}}(x) - x||_1] \tag{3.12}$$

The overall loss function for the cycleGAN consists of three different terms for the generator.

$$\mathcal{L} = \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{cycle}} + \mathcal{L}_{\text{identity}} \tag{3.13}$$

Since the publication of this architecture in 2017, a lot of research has been done in order to improve the output quality [5, 62], allow to guide the generation process [38, 9, 70] or modifying the architecture for further use cases [57, 31]. Whereby, one use case is the unpaired video-to-video transformation. Although, it is quite close to the image-to-image transformation task, it brings a lot of additional challenges with it. In particular, the temporal consistency between the consecutive frames plays a crucial role. Interestingly, two research groups already explored the potential of unpaired video-to-video transformation for the usage in medical training simulators [16, 48]. The proposed tempCycleGAN [16] approach is based on the original cycleGAN architecture but considers two consecutive frames for one translation. By using another temporal discriminator and additional loss functions the architecture show better temporal consistency than using the standard cycleGAN. Another idea is to use the information about the optical flow within a video sequence. The proposed MoCycleGAN (Motion-guided CycleGAN) [8] leverages the optical flow between two consecutive frames to restrict the generator through an appropriate loss function (Figure 3.12).

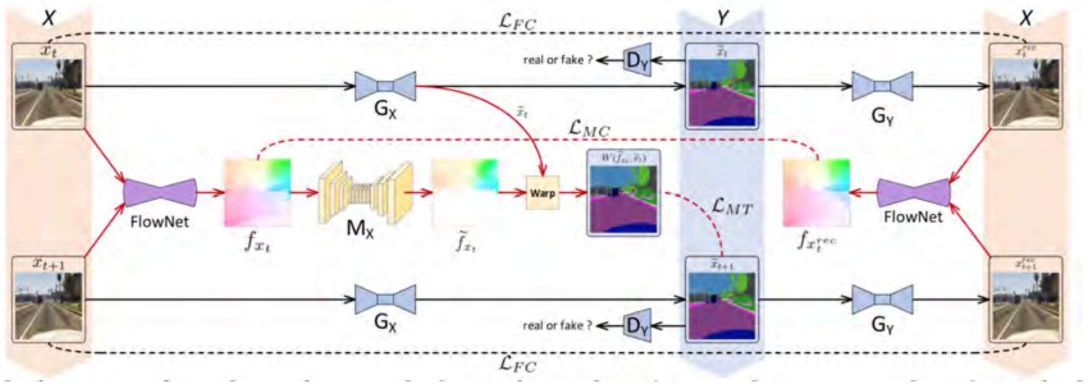


Figure 3.12: Architecture of the MoCycleGAN [8]

Although, the MoCycleGAN shows promising results with simultaneous minimization of the inference time compared to the tempCycleGAN, it still uses the optical flow only implicitly. This means that the results are not always satisfactory.

3.6 Image Inpainting & Composition

Image inpainting with neural networks is a cutting-edge technique in computer vision and image processing. The goal of inpainting is to reconstruct missing or damaged regions in an image seamlessly, making the alterations visually imperceptible (Figure 3.13). Deep learning models have shown remarkable success in addressing this task. The applications of image inpainting with neural networks are diverse, ranging from restoring old photographs and removing unwanted objects to medical image restoration and video editing [34, 39, 40, 51].

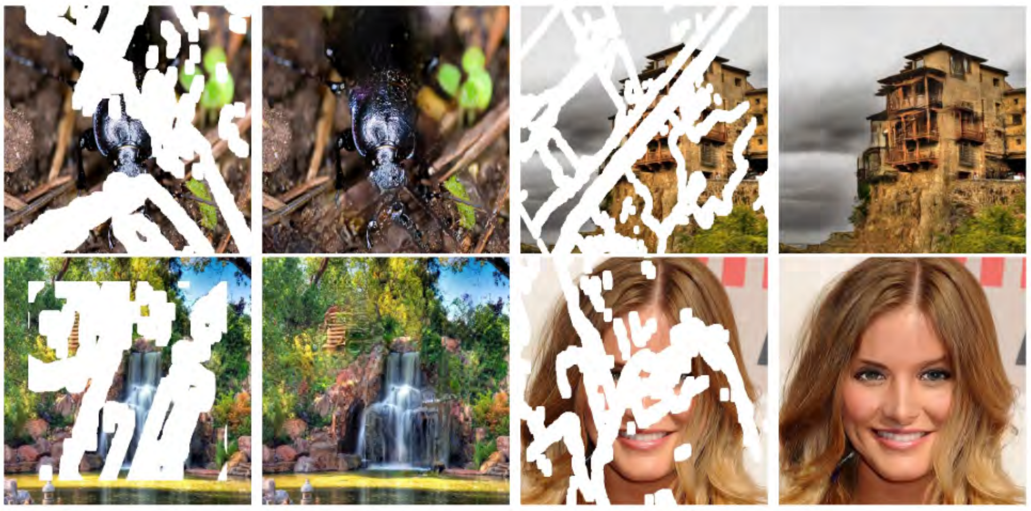


Figure 3.13: Images with masked regions (white) and the corresponding reconstructed images. Reconstruction is done with the help of a deep neural network [37].

In some cases the model should not simply fill the missing area such that it looks realistic, but the filled region should contain a particular object (Figure 3.14). The approach where an inpainting model is guided with an object is called image composition [7, 55, 64].

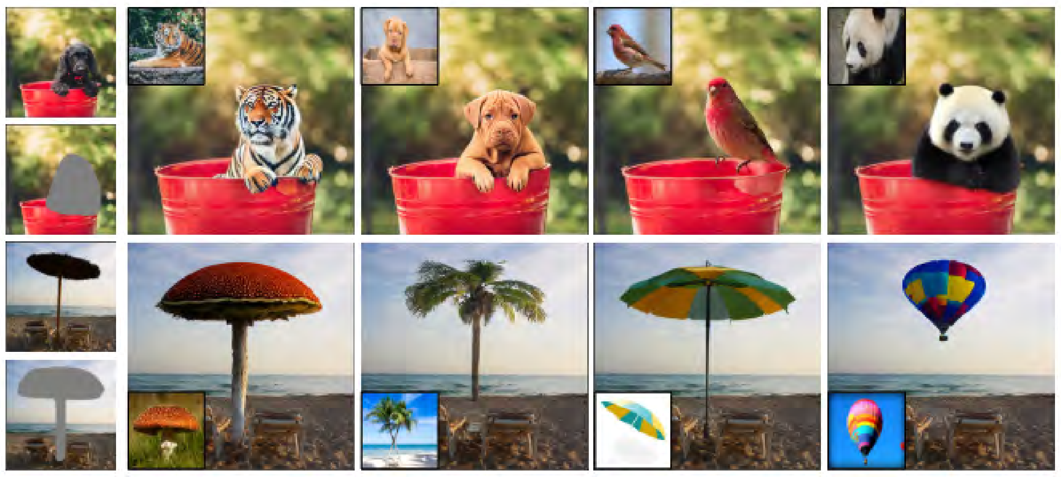
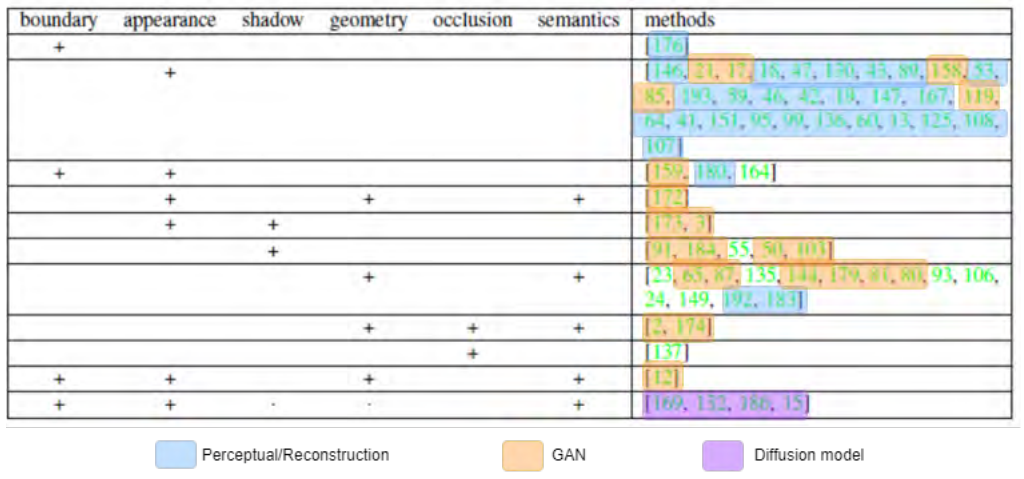


Figure 3.14: Examples for the task of image composition. The small images on the left show the original image and the masked (grey) image. The other columns show the result of the image composition with the given reference object in the top left or bottom left corner respectively [64].

Although, maybe the task to fill a region of interest with missing pixels sounds not too difficult, on closer inspection, the task can be quite challenging even for the complex architectures. The problem of image inpainting or composition, respectively, can be decomposed into six different sub-tasks. The first one is the *boundary* problem, which means that the filled pixels should have a smooth transition from foreground to background. The *appearance* (e.g., illumination) should also be adjusted to fit the overall context of the image. The generation of a *shadow* is important in order to be close to reality, but it is quite challenging. Regarding the inpainting of an object the size and *geometry* should be adjusted according to the overall image. Furthermore, the *semantic* context of an object should be obtained. And last but not least it should be possible to copy with the unreasonable *occlusion*. A survey [44] from 2023 summarized existing methods for each of the sub-tasks which are shown in Figure 3.15.



boundary	appearance	shadow	geometry	occlusion	semantics	methods
+						[176]
	+					[146, 21, 17, 18, 47, 130, 43, 89, 158, 53, 85, 193, 39, 46, 42, 19, 147, 167, 119, 64, 41, 151, 95, 99, 136, 60, 13, 125, 108, 107]
+	+					[159, 180, 164]
	+		+		+	[172]
	+	+				[173, 3]
		+				[91, 184, 55, 50, 103]
			+		+	[23, 65, 87, 135, 144, 179, 81, 80, 93, 106, 24, 149, 192, 183]
			+	+	+	[2, 174]
				+		[137]
+	+		+		+	[12]
+	+	·	·		+	[169, 132, 186, 15]

🟦 Perceptual/Reconstruction 🟧 GAN 🟪 Diffusion model

Figure 3.15: Issues to be solved in image inpainting/composition and the corresponding deep learning methods to solve these issue. (Changed according to [44])

Whereby the different methods investigated can be categorized into three different architectures. The first approach is to use an encoder/decoder architecture with perceptional and reconstruction losses (marked in light blue). These methods seem to be able to solve mainly the task of the boundary conditions and the appearance, but fail at the other tasks. Methods that are based on GAN (marked in orange) architectures appear to be able to learn more complex tasks and can also cover the shadow generation or geometry. But still they are not able to cover all of the six tasks at once. Recently published methods which are using diffusion models (marked in violet) show promising results when trying to cover all tasks [44].

3.7 Optical Flow

Non-linear registration is a crucial technique in medical image analysis that enables the alignment of images with complex spatial deformations. Unlike linear registration, which assumes global affine transformations (e.g. scaling, rotation, etc.), non-linear registration accounts for local variations and distortions within the images. Non-linear registration aims to find a transformation φ that aligns two or more images $I_{k=1,\ldots n}$ by warping one image I_i onto another image I_j. This transformation is typically represented by a deformation field, which describes how each point in the moving image is mapped to the corresponding point in the fixed image. Unlike linear transformations, which preserve straight lines and parallelism, non-linear transformations can model complex deformations such as bending, stretching, and twisting. Non-linear registration involves optimizing a cost function $\mathcal{D}$ that measures the discrepancy between a reference image R and a target image T under the deformation field. Often the reference image R is called fixed image and the image T is referred to as moving image, since the transformation φ is applied to T. Mathematically, the resulting optimization problem can be expresses like

$$J[R, T \circ \varphi] := \mathcal{D}[R, T \circ \varphi] + \mathcal{S}[\varphi] \rightarrow min \tag{3.14}$$

The term $\mathcal{S}$ is introduced as an additional regularization to provide particular smoothness [63]. As cost function J, the NCC (Normalized Cross-Correlation) is commonly used since it quantifies the similarity between two images [68]. Mathematically, the NCC between two images R and T at a given displacement (x,y) can be expressed as:

$$NCC(R,T) = \frac{\sum_{x,y}(R(x,y) - \bar{R}) \cdot (T(x,y) - \bar{T})}{\sqrt{\sum_{x,y}(R(x,y) - \bar{R})^2 \cdot \sum_{x,y}(T(x,y) - \bar{T})^2}} \tag{3.15}$$

Whereby, $\bar{R}$ and $\bar{T}$ denote the means of images R and T respectively. The NCC value lies between -1 and 1, where a value of 1 indicates perfect similarity, 0 indicates no correlation, and -1 indicates perfect negative correlation. Due to this, the goal in this case is to minimize the negative of the NCC value, since most optimization algorithms are designed for minimization. For the optimization process itself, techniques like gradient descent are employed to find the optimal transformation parameters that minimize the cost function. During each iteration of the optimization process, a Gaussian kernel can be applied to the deformation field. It helps to regularize the optimization process, making it more stable and less sensitive to noise and local variations in the images. Since the decision of the size of the kernel is a trade-off between capturing fine details and overall stability, it has to be chosen carefully, depending on the application.

To further improve the efficiency of the optimization, a coarse-to-fine strategy can be employed. This strategy estimates a deformation field efficiently by first obtaining a coarse-level estimate, then refining it iteratively at finer resolutions. It involves representing images and deformation fields in a pyramid structure, refining the estimation using hierarchical optimization or interpolation, which lead to a better stability and faster convergence [42]. After the optimisation converged or reached a given number of iterations, the final result will be the motion field u. This field now contains the information about how to move each pixel of the moving image such that it aligns with the fixed image (Figure 3.16).

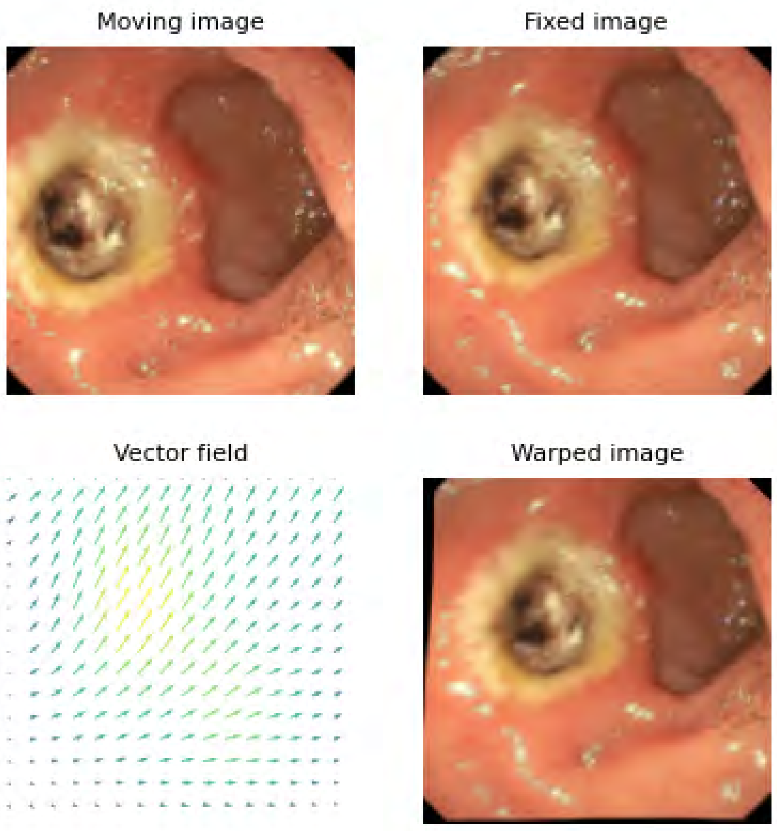


Figure 3.16: Warping an image (moving) at time step t of a video sequence to align it with a frame (fixed) at time step $t + 6$ using the calculated vector field.

Here, only every eighth vector of the motion field is illustrated for better clarity. It is clearly visible by observation that the ulcus in the fixed image is moved more towards the upper right corner compared to the moving image. This is also confirmed when looking at the directions of the vectors and thus the warped image looks quite similar to the fixed image. However, the warped images shows black pixels on the left, which were not there before. This is due to the fact that there are no known pixels at the edges of the images which could be mapped to the corresponding positions. Therefore, these pixels were set to a default values which is typically 0.

4 Denoising diffusion probabilistic models

Deep Diffusion Probabilistic Models (DDPMs) [25] are a recent breakthrough in generative modeling, combining principles of probabilistic modeling and deep neural networks. Unlike traditional approaches, DDPMs explicitly model the diffusion process, which originates from stochastic processes and statistical physics. Stochastic processes describe the evolution of random variables over time, and diffusion processes specifically model the random movement of particles or information through a medium [56]. In the context of DDPMs, the diffusion process is adapted to iteratively transform noise-corrupted inputs into realistic samples by gradually reducing the level of noise. This idea draws inspiration from physical diffusion, where particles spread out from regions of high concentration to regions of low concentration, gradually achieving equilibrium. By modeling this diffusion process, DDPMs can effectively capture complex data distributions and generate high-quality samples [25].

4.1 Theory

The generation of samples is done in two stages. The first stage is called the forward process q in that the noise is applied.

$$q(x_{\mathrm{t}}|x_{\mathrm{t\text{-}1}}) = \mathcal{N}(x_{\mathrm{t}}, \sqrt{1-\beta_{\mathrm{t}}}x_{\mathrm{t\text{-}1}}, \beta_{\mathrm{t}}\mathcal{I}) \tag{4.1}$$

Here, the noise is sampled from the normal distribution $\mathcal{N}$. The output sample x_t is the result depending on the previous sample x_{t-1} noised with the mean of $\sqrt{1-\beta_t}$ and variance of β_t. A smaller number regarding the time step t means in this case a lower amount of noise and a higher number of t means a high amount of noise, respectively. The value of β_t belongs to a given schedule which will be explained later.

One possibility to noise the sample is to iteratively apply the forward process for each time step. However, it is possible to do all time steps in one pass. In order to derive this, a few notations are important.

$$\alpha_{\text{t}} = 1 - \beta_{\text{t}} \tag{4.2}$$

In addition the cumulative product of all alphas will be used for better clarity.

$$\bar{\alpha}_{\text{t}} = \prod_{s=1}^{t} \alpha_{\text{s}} \tag{4.3}$$

With the help of the so called reparameterization trick $\mathcal{N}(\mu, \sigma_2) = \mu + \sigma \cdot \epsilon$ the forward process can be rewritten.

$$q(x_{\text{t}}|x_{\text{t-1}}) = \sqrt{1 - \beta_{\text{t}}} x_{\text{t-1}} + \sqrt{\beta_{\text{t}}} \epsilon \tag{4.4}$$

Whereby, ϵ is taken from a normal distribution $\epsilon \sim \mathcal{N}(0, 1)$ with 0 mean and a standard deviation of 1. The previously introduced notation for α_t can be used to rewrite the formula again.

$$q(x_{\text{t}}|x_{\text{t-1}}) = \sqrt{\alpha_{\text{t}}} x_{\text{t-1}} + \sqrt{1 - \alpha_{\text{t}}} \epsilon \tag{4.5}$$

It turns out that it is possible to calculate x_t at each time step by just using all necessary values of α_t. Even more interesting is the fact that the final noised sample x_t can be directly calculated from the initial sample x_0.

$$\begin{aligned} q(x_{\mathrm{t}}|x_{\mathrm{t-1}}) &= \sqrt{\alpha_{\mathrm{t}}}x_{\mathrm{t-1}} + \sqrt{1-\alpha_{\mathrm{t}}}\epsilon \\ &= \sqrt{\alpha_{\mathrm{t}}\alpha_{\mathrm{t-1}}}x_{\mathrm{t-2}} + \sqrt{1-\alpha_{\mathrm{t}}\alpha_{\mathrm{t-1}}}\epsilon \\ &= \sqrt{\alpha_{\mathrm{t}}\alpha_{\mathrm{t-1}}\alpha_{\mathrm{t-2}}}x_{\mathrm{t-3}} + \sqrt{1-\alpha_{\mathrm{t}}\alpha_{\mathrm{t-1}}\alpha_{\mathrm{t-2}}}\epsilon \\ &= ... \\ &= \sqrt{\alpha_t\alpha_{t-1}...\alpha_1\alpha_0}x_0 + \sqrt{1-\alpha_t\alpha_{t-1}...\alpha_1\alpha_0}\epsilon \end{aligned}$$

Using the accumulated product $\hat{\alpha}_{\mathrm{t}}$, the expression can be further simplified.

$$q(x_t|x_{t-1}) = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{4.6}$$

With this formula it is possible to noise a sample in one pass independently of the amount of the time steps, which makes the forward process fast and easy.

The second stage of DDPMs is the reverse diffusion process $p(x_{\mathrm{t-1}}|x_{\mathrm{t}})$. This time, it takes a noised sample x_t and with the help of a neural network it produces a less noised sample x_{t-1}.

$$p(x_{\mathrm{t-1}}|x_{\mathrm{t}}) = \mathcal{N}(x_{\mathrm{t-1}}, \mu_\Theta(x_{\mathrm{t}}, t), \Sigma_\Theta(x_{\mathrm{t}}, t)) \tag{4.7}$$

Whereby Σ_Θ is the neural network to predict the variance and μ_Θ is a network to predict the mean of the noise. Often the variance is fixed during the sampling procedure, which means no network is necessary for this parameter. Therefore the task is only to predict the mean using μ_Θ [25]. As already mentioned the training of a neural network p_Θ is based on the minimization of a loss function. In theory, the negative log-likelihood could be chosen, since it would indicate if a generated sample belongs to the data distribution of the training data or not:

$$\mathcal{L} = -log(p_\Theta(x_0)) \tag{4.8}$$

In practice, this is not possible, because $p_\Theta(x_0)$ depends on all the time steps before and therefore, it is not easy to keep track of all the gradients in each step. As a solution, the variational lower bound is used as a more nicely calculable target. The idea is that in case a given function $f(x)$ can not be calculated and it can be proven that there is a function $g(x)$ which is always smaller than $f(x)$, this function will increase when maximizing $g(x)$. Here, $f(x)$ is the negative log-likelihood and $g(x)$ is the Kullback-Leibler divergence (KL-divergence) [25].

$$-log(p_\Theta(x_0)) \leq -log(p_\Theta(x_0)) + D_{KL}(q(x_{1:T}|x_0)||p_\Theta((x_{1:T}|x_0)) \tag{4.9}$$

The KL-divergence is a measure how close two distributions are and will not become negative.

$$D_{KL}(p||q) = \int_x p(x) log(\frac{p(x)}{q(x)})\, dx \tag{4.10}$$

Since $-log(p_\Theta(x_0))$ still occurs on both sides and therefore, it is not calculable so it has to be further reformulated.

$$D_{KL}(q(x_{1:T}|x_0)||p_\Theta((x_{1:T}|x_0)) = log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{1:T}|x_0)}) \tag{4.11}$$

After that, the Baysian rule is applied to the lower term $p_\Theta(x_{1:T}|x_0)$.

$$\begin{aligned} p_\Theta(x_{1:T}|x_0) &= \frac{p_\Theta(x_0, x_{1:T}) p_\Theta(x_{1:T})}{p_\Theta(x_0)} \\ &= \frac{p_\Theta(x_0, x_{1:T})}{p_\Theta(x_0)} \\ &= \frac{p_\Theta(x_{0:T})}{p_\Theta(x_0)} \end{aligned}$$

Thus, the log-term in Equation 4.11 can be rewritten.

$$log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{1:T}|x_0)}) = log(\frac{q(x_{1:T}|x_0)}{\frac{p_\Theta(x_{0:T})}{p_\Theta(x_0)}}) \tag{4.12}$$

Now the bottom term of the denominator can be pulled up and the log rule applied afterwards.

$$log(\frac{q(x_{1:T}|x_0)}{\frac{p_\Theta(x_{0:T})}{p_\Theta(x_0)}}) = log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) + log(p_\Theta(x_0)) \tag{4.13}$$

After reaching that point, this term is inserted in the original formula (Equation 4.9) which is the starting point. By doing this, the term $log(p_\Theta(x_0))$ cancels each other out and the quantity, which is not calculable, vanishes. This ends up in the variational lower bound which can be minimized during the training of the network $\mu_\Theta(x_\mathrm{t}, t)$.

$$-log(p_\Theta(x_0)) \leq log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) \tag{4.14}$$

Looking at the variational lower bound it is noticeable that the upper term $q(x_{1:T}|x_0)$ is just the forward process and the lower can be further rewritten as the following.

$$p_\Theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\Theta(x_{t-1}|x_t) \tag{4.15}$$

Here, $p(x_T)$ can be computed using the neural network and $p_\Theta(x_{t-1}|x_t)$ is the output of at time step t. At this point, the authors of the original DDPM paper [25] decided to do some more reformulation steps to arrive at a more easy calculable expression which can be later implemented properly. For this, the current formulation for the lower bound (Equation 4.14) is rewritten and log-rules were applied.

$$log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) = log(\frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{p(x_T)\prod_{t=1}^{T} p_\Theta(x_t|x_{t-1})}) \tag{4.16}$$

$$= -log(p(x_T)) + log(\frac{\prod_{t=1}^{T} q(x_t|x_{t-1})}{\prod_{t=1}^{T} p_\Theta(x_t|x_{t-1})}) \tag{4.17}$$

$$= -log(p(x_T)) + \sum_{t=1}^{T} log(\frac{q(x_t|x_{t-1})}{p_\Theta(x_t|x_{t-1})}) \tag{4.18}$$

At this point the authors decided to take out the first summand for $t = 1$, which will make more sense in the end.

$$log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) = -log(p(x_T)) + \sum_{t=2}^{T} log(\frac{q(x_t|x_{t-1})}{p_\Theta(x_t|x_{t-1})}) + log(\frac{q(x_1|x_0)}{p_\Theta(x_0|x_1)}) \tag{4.19}$$

Next the nominator $q(x_t|x_{t-1})$ is rewritten by using the Bayesian rule.

$$q(x_t|x_{t-1}) = \frac{q(x_{t-1}|x_t)q(x_t)}{q(x_{t-1})} \tag{4.20}$$

But there is the problem with this formulation, since $q(x_t|x_{t-1})$ depends on x_t and x_{t-1} which can have a high variance because up to this point there is no information about the initial x_0. Therefore, the authors reduced the high variance by conditioning Equation 4.20 also on x_0. These conditioning reduces the high variance and the output get more certain.

$$q(x_t|x_{t-1}) \Rightarrow \frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)} \tag{4.21}$$

Replacing the new term for $q(x_t|x_{t-1})$ in 4.19 will lead to the following.

$$log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) = -log(p(x_T)) + \sum_{t=2}^{T} log(\frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{p_\Theta(x_{t-1}|x_t)q(x_{t-1}|x_0)}) + log(\frac{q(x_1|x_0)}{p_\Theta(x_0|x_1)}) \tag{4.22}$$

By having a closer look at the formula it is now more clear why the authors decided to take out the first summand. If they had not done it, the formula would end up in a loop after conditioning on x_0.

$$\boxed{q(x_1|x_0)} = \frac{q(x_0|x_1)q(x_1)}{q(x_0)} \Rightarrow \frac{q(x_0|x_1, x_0)\boxed{q(x_1|x_0)}}{q(x_0|x_0)} \tag{4.23}$$

For further simplification, the sum in Formula 4.22 is split up again which will result in following term.

$$\begin{aligned} log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) = -log(p(x_T)) &+ \sum_{t=2}^{T} log(\frac{q(x_{t-1}|x_t, x_0)}{p_\Theta(x_{t-1}|x_t)q(x_{t-1}|x_0)}) \\ &+ \sum_{t=2}^{T} log(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}) \\ &+ log(\frac{q(x_1|x_0)}{p_\Theta(x_0|x_1)}) \end{aligned}$$

When writing out the second summation, it becomes apparent that a lot of terms cancel each other out, which can be demonstrated by choosing $T = 4$ as an example.

$$\begin{aligned} \sum_{t=2}^{4} log(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}) &= log(\prod_{t=2}^{4} \frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}) \\ &= log(\frac{q(x_2|x_0)q(x_3|x_0)q(x_4|x_0)}{q(x_1|x_0)q(x_2|x_0)q(x_3|x_0)} \end{aligned}$$

Only the last term in the nominator and the first one in the denominator remain. With this observation, the sum can be dramatically simplified.

$$\sum_{t=2}^{T} log(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}) = log(\frac{q(x_T)|q(x_0)}{q(x_1)|q(x_0)}) \tag{4.24}$$

This outcome can be inserted in the Equation 4.24.

$$\begin{aligned} log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) = -log(p(x_T)) + \sum_{t=2}^{T} log(\frac{q(x_{t-1}|x_t, x_0)}{p_\Theta(x_{t-1}|x_t)q(x_{t-1}|x_0)}) \\ + log(\frac{q(x_T)|q(x_0)}{q(x_1)|q(x_0)}) \\ + log(\frac{q(x_1|x_0)}{p_\Theta(x_0|x_1)}) \end{aligned}$$

The last two terms can be further rewritten using the logarithmic rules.

$$\begin{aligned} log(\frac{q(x_T|x_0)}{q(x_1|x_0)}) + log(\frac{q(x_1|x_0)}{p_\Theta(x_0|x_1)}) = log(q(x_T|x_0)) - log(q(x_1|x_0)) \\ + log(q(x_1|x_0)) - log(p_\Theta(x_0|x_1)) \end{aligned}$$

Whereby, the term two and three cancel each other out. At this point, the authors decided to also bring the first term to the front and fuse it with $-log(p(x_T))$.

$$log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) = -log(\frac{q(x_T|x_0)}{p(x_T)}) + \sum_{t=2}^{T} log(\frac{q(x_{t-1}|x_t, x_0)}{p_\Theta(x_{t-1}|x_t)}) - log(p_\Theta(x_0|x_1)) \tag{4.25}$$

The Equation 4.25 can be also rewritten using the KL-divergence notation which ends up in a nice analytical expression.

$$\begin{aligned} log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) = & D_{KL}(q(x_T|x_0)||p(x_T)) \\ & + \sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\Theta(x_{t-1}|x_t)) \\ & - log(p_\Theta(x_0|x_1)) \end{aligned}$$

Since the first KL-divergence term does not contain any learn-able parameter, it can be ignored in this case. Also due to the fact that $q(x_T|x_0)$ will converge to a normal distribution and $p(x_T)$ is sampled from it, the difference will be really small anyways.

$$log(\frac{q(x_{1:T}|x_0)}{p_\Theta(x_{0:T})}) = \sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\Theta(x_{t-1}|x_t)) - log(p_\Theta(x_0|x_1))$$

As already mentioned in the beginning, $p_\Theta(x_{t-1}|x_t)$ can be expressed like the following.

$$\mathcal{N}(x_{t-1}; \mu_\Theta(x_t, t), \Sigma_\Theta(x_t, t)) = \mathcal{N}(x_{t-1}; \mu_\Theta(x_t, t), \beta I)$$

Whereby, the variance Σ is a fixed schedule and these closed form solution also exits for the forward process.

$$\mathcal{N}(x_{t-1}; \hat{\mu}_t(x_t, t), \hat{\beta} I)$$

For the sake of simplicity, the derivation of $\hat{\mu}$ and $\hat{\beta}$ is not covered here. It can be proven that the final version of $\hat{\mu}$ and $\hat{\beta}$ looks like the following [25].

$$\hat{\mu}_t(x_t, x_0) = \frac{\sqrt{\alpha_t(1-\hat{\alpha}_{t-1})}}{1-\hat{\alpha}_t}x_t + \frac{\sqrt{\hat{\alpha}_{t-1}}\beta_t}{1-\hat{\alpha}_t}x_0 \tag{4.26}$$

$$\hat{\beta}_t = \frac{1-\hat{\alpha}_{t-1}}{1-\hat{\alpha}_t} \cdot \beta_t \tag{4.27}$$

Since $\hat{\beta}$ is fixed, the focus only lays on $\hat{\mu}$. The closed form solution for the forward process can be expressed like the following (see Equation 4.5).

$$x_t = \sqrt{\hat{\alpha}_t}x_0 + \sqrt{1-\hat{\alpha}_t}\epsilon$$

Whereby, the formula can be rewritten in terms of x_0.

$$x_0 = \frac{1}{\hat{\alpha}_t}(x_t - \sqrt{1-\hat{\alpha}_t}\epsilon)$$

This term can be inserted into the Equation 4.27, which leads to a formulation for $\hat{\mu}_t$ that does not depend on x_0 anymore.

$$\begin{aligned}\hat{\mu}_t &= \frac{\sqrt{\alpha_t(1-\hat{\alpha}_{t-1})}}{1-\hat{\alpha}_t}x_t + \frac{\sqrt{\hat{\alpha}_{t-1}}\beta_t}{1-\hat{\alpha}_t}\frac{1}{\hat{\alpha}_t}(x_t - \sqrt{1-\hat{\alpha}_t}\epsilon)\\ &= \frac{1}{\hat{\alpha}_t}(x_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\epsilon)\end{aligned}$$

The authors [25] decided to use a simple L_2-norm for the loss function and calculate the difference between the actual mean $\hat{\mu}_t(x_t, t)$ and the predicted mean $\mu_\Theta(x_t, t)$ and due to the fact that x_t is already available as input to the model and thus does not have to be predicted. Therefore, it is sufficient to only calculate the mean squared error between the predicted and the actual noise.

$$\begin{aligned}
\mathcal{L}_t &= \frac{1}{2\sigma_t^2}||\hat{\mu}_t(x_t,t) - \mu_\Theta(x_t,t)||^2 && (4.28)\\
&= \frac{1}{2\sigma_t^2}||\frac{1}{\hat{\alpha}_t}(x_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\epsilon) - \frac{1}{\hat{\alpha}_t}(x_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\epsilon_\Theta(x_t,t)||^2 && (4.29)\\
&= \frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\hat{\alpha}_t)}||\epsilon - \epsilon_\Theta(x_t,t)||^2||^2 && (4.30)
\end{aligned}$$

Interesting is that, better sampling quality was experienced when ignoring the scaling factor in front. Which leads to the final version of the loss function for the neural network [25].

$$\mathcal{L}_t = ||\epsilon - \epsilon_\Theta(x_t,t)||^2 \qquad (4.31)$$

Once the neural network is trained to predict the noise, it can be used to remove the noise within the sample with the help of the reparameterization trick.

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\epsilon_\Theta(x_t,t)) + \sqrt{\beta_t}\epsilon \qquad (4.32)$$

Commonly, the neural network is based on a basic U-Net architecture, but additionally the time step t is embedded and integrated in the each stage of the U-Net to condition the model on this parameter (4.1).

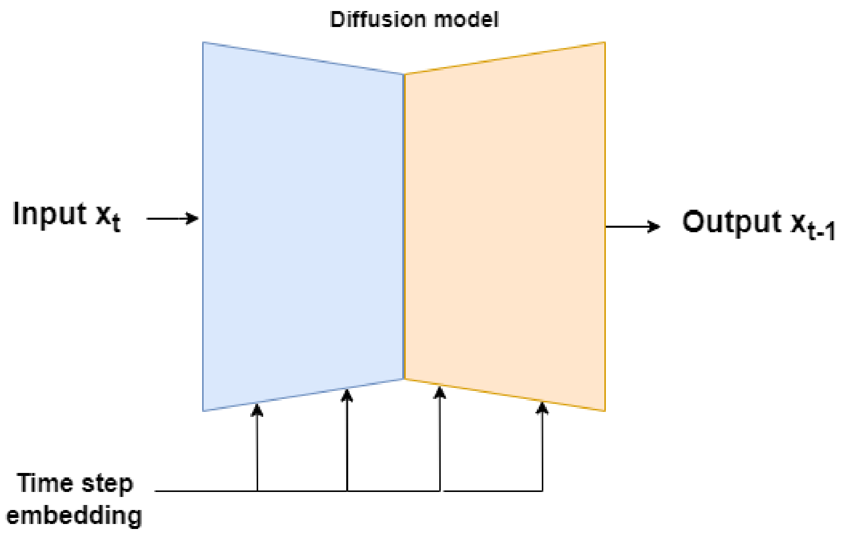


Figure 4.1: Diffusion model based on a U-Net architecture with time step embedding.

4.2 Noise Scheduling

The way how the noise is applied to the image plays a crucial role when training a diffusion model. Several methods have already been tested and the optimal noise scheduling depends on the specific task. For example, when increasing the image size it is recommended to use a more noisier schedule due to increased redundancy in pixels [6]. In order to get a brief intuition of the noise scheduling, the linear and the cosine scheduler will be compared. In general $\hat{\alpha}_t$ is sampled and calculated using a function $f(t)$.

$$\hat{\alpha}_t = \frac{f(t)}{f(0)} \tag{4.33}$$

For a linear schedule the values for $\hat{\alpha}_t$ are simply interpolated between a given start and end value.

$$f(t) = \frac{\beta_{end} - \beta_{start}}{steps} \cdot t \tag{4.34}$$

In case a cosine schedule is used, the $\hat{\alpha}_t$ are sampled from a cosine function.

$$f(t) = \cos(\frac{\frac{t}{T} + s}{1 + s} \cdot \frac{\pi}{2})^2 \tag{4.35}$$

The parameter s is used as a small offset to prevent β_t from being too small near $t = 0$ [43]. Independently of the schedule, the calculation of the corresponding β_t stays the same.

$$\beta_t = 1 - \frac{\hat{\alpha}_t}{\hat{\alpha}_t - 1} \tag{4.36}$$

The Figure 4.2 shows the difference when noising an image with the linear and the cosine schedule. It is clearly visible that the features and structures of the images are maintained for longer when using the cosine schedule.

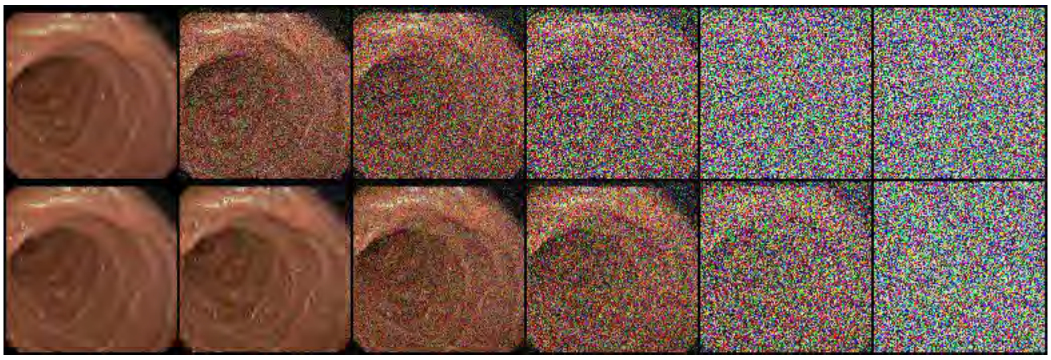


Figure 4.2: Comparison of linear and cosine noise scheduling at time steps 0, 100, 150, 300, 600 and 999.

Overall, the scheduling principle for a DDPM involves determining the schedule of diffusion steps during training and inference to effectively denoise images while optimizing performance and convergence [6].

4.3 Further Improvements

There are two major modifications to the original training procedure, which lead to better results. The first one is the principle of classifier free-guidance (CFG) and the second is the use of the exponential moving average (EMA).

The CFG approach [26] introduces a new variable to condition the prediction. For example, the output should belong to a certain class and should not be an arbitrary image sampled from the data distribution. Therefore, an additional information about this class needs to be provided to the network in order to control the generation process. Since the model is already conditioned on the sampling step t, the easiest way is to add the label y to the time step. For this, the label y and the time step t must have the same dimension and since the number of classes is most likely not the number of time steps there must be an additional embedding of y in order to add these two. Interestingly, the authors decided to train the model only in 90% of the training steps with a conditional class label, the other 10% the class label will be ignored. This allows that the model learns to generate images both conditional and unconditional. Also it is observed that the results show a better quality. In the sampling process, both results will be calculated and linearly interpolated between these outputs afterwards [26]:

$$\epsilon_t = \epsilon_{t,\Theta} + w \cdot (\epsilon_{t,c} - \epsilon_{t,,\Theta}) \tag{4.37}$$

Whereby $\epsilon_{t,c}$ is the predicted noise using the class label information and $\epsilon_{t,\Theta}$ is the prediction without. The parameter w indicates a weight which influences the impact of the class label to the output. A higher value of w will ensure that the result will belong to the given class but it will also decrease the quality.

The second improvement is the use of EMA, which has also been used in other training procedures in different model architectures. The goal is to smoothen the training process regarding updating the weights. In case the training process leads to a noise update of the weights meaning that the direction of the gradient changes a lot, EMA can smoothen this and ensure a more robust outcome. The implementation for this strategy is quite simple. In the beginning a copy of the model must be made. The weights of the normal model p_Θ will updated as usually using the loss and a given optimizer. The weights of the EMA model $p_{\Theta,EMA}$ will updated depending on the old and new weights.

$$w = \beta \cdot w_{old} + (1 - \beta) \cdot w_{new} \tag{4.38}$$

This is simply an interpolation between the old weights of the model and the new one weighted with the parameter β. Usually, β is around 0.99 which means that the influence of the new weights is very low. This avoids that big outliers during the training process will have a large impact on the model weights.

4.4 Stable Diffusion Model

The Stable Diffusion model [49] is a text-to-image model, which was invented among others by CompVis Group at Ludwig Maximilian University of Munich and released in 2022. The main difference compared to a normal diffusion model is that the denoising process is applied in the latent space and not on the image itself. In order to transform the images into a suitable latent space representation, a pretrained auto-encoder network is used (Figure 4.3).

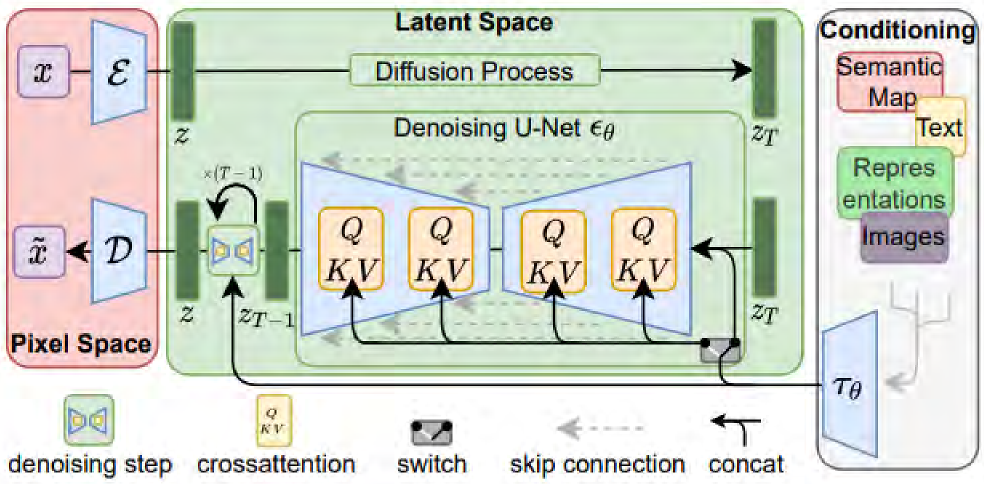


Figure 4.3: Architecture of the Stable Diffusion model. [49]

The goal remains the same, which is to predict the noise and therefore, the loss function remains the same as discussed before. Finally, the whole training procedure of the Stable Diffusion model is splitted into three stages. The first stage is the training of the encoder and decoder. After that, these networks were used to transform the images into latent space in order to apply it to the U-Net model, which is trained in the second stage. Once these two training phases were carried out, the U-Net model can be trained once again but conditioned on an additional label.

5 Methods

As already mentioned, there are a lot of architectures for the task of image inpainting and image composition, respectively. In this work, four of them are chosen to test the performance on this particular use case. Thereby, two of the architectures are based on GANs and the other two uses DDPM. First, the StyleMapGAN [35] will be investigated since it is designed to learn the feature representation in a latent space and can than be used as an improved „copy & paste“ technique. It was chosen as it should also work when having a low amount of data available like it is the case for images with pathologies. The second GAN approach is the SAC-GAN [4], which has the advantage that it allows a free form inpainting at any given location, which is not the case for the StyleMapGAN. Additionally, the architecture is designed such that it can be conditioned on a label and thus control the content of the inpainted area. The third architecture that will be investigated is the standard DDPM. Since it can be trained and used for this task without any further modification it is perfect to get a first intuition of the performance of DDPMs for the inpainting task. This also allows to compare the results of GAN-based methods and DPPMs. The last architecture will be a more advanced DDPM named Anydoor [7], which is based on the Stable Diffusion architecture [49]. Whereby, this model is not designed as an inpainting model as the other approaches. It is made for image composition, i.e. the inpainting of a reference object into a target image. With this, the following methods covers both the comparison of GAN-based architectures and DDPMs as well as image inpainting and image composition.

5.1 StyleMapGAN

Recent approaches of GAN-based image generation not only focus on the generation of high quality images, but also to be able to influence and guide the image synthesis. Most of the techniques do this by manipulating the latent space [28, 46, 54]. Although this works, it comes with a few drawbacks which lower the quality of the generated images. First the projection to a single latent vector can be very time consuming and the manipulation highly depends on a accurate embedding of the encoder. In order to address these problems, the StyleMapGAN [35] uses an intermediate latent space representation instead of a single vector. This maintains higher features of the images, which improves the results regarding local image editing and image interpolation.

The architecture of the StyleMapGAN consists of four networks F, G, E and D (Figure 5.1). The synthesis network G is trained to reconstruct real images in terms of both pixel-level and perceptual-level. The encoder E learns to reconstruct the stylemap when $G(F)$ generated an image using the a vector sampled from Gaussian distribution. As usual, the discriminator D tries to distinguish between the real and generated images. For the inference itself, only the encoder E and the synthesis network G are needed.

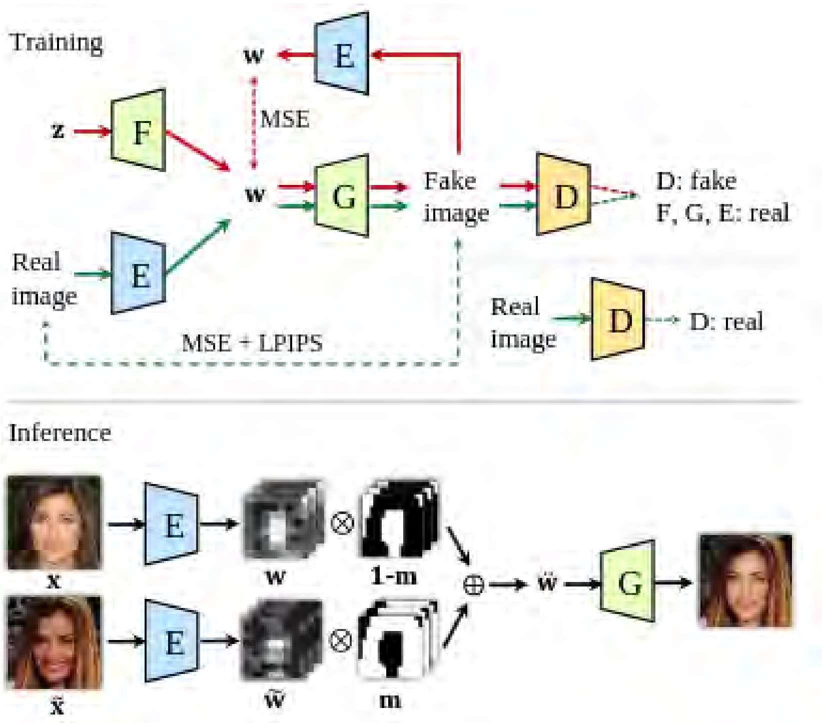


Figure 5.1: Training and inference principle of StyleMapGAN [35].

During the training process several loss functions are used to optimize the parameters of the networks. The standard adversarial loss is applied to the synthesis network G and the discriminator D. Additional D, is stabilized using the R1-regularisation [41].

$$R_1 = \lambda \sum_{i=1}^{n} |w_i| \tag{5.1}$$

It adds a penalty to the objective function that discourages complexity in the model by shrinking the weights w towards zero. Whereby, the strength of the regularisation can be controlled with the parameter λ. This helps to prevent over-fitting and improves the generalization ability of the model, especially in high-dimensional data sets where feature selection is crucial. The mean squared error is used for the latent reconstruction of the encoder and the image reconstruction of E and G. Furthermore, the learned perceptual image patch similarity (LPIPS) [66] is applied to E and G as perceptual loss. This helps to decrease the difference between original and reconstructed images.

The influence of the size of the stylemaps were also investigated and the results are illustrated in Figure 5.2. It turns out that when choosing a too low spatial resolution the outputs suffer from poor reconstruction, i.e high level features get lost. This matches with the observations of approaches using a single vector as latent representation. The other way around if the stylemap dimension is too high, too many high level features are preserved and the results are not harmonious. The best output were achieved at a stylemap size of 8x8.

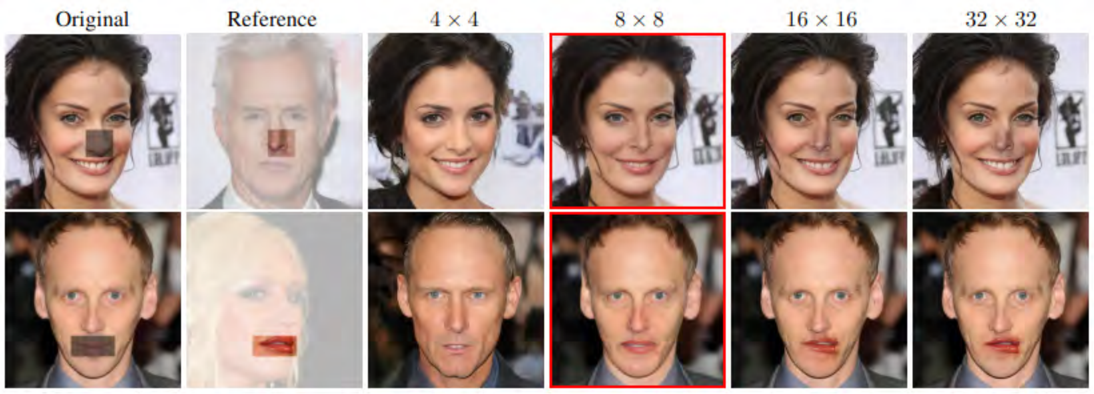


Figure 5.2: Influence of dimension size for style maps [35].

Once the training is done, the image editing can be performed by simply transforming two images into the stylemap representation and blending the region of interest. Whereby, the region can be masked on the original images as the mask is shrunk by max pooling in order to align it to the corresponding location in the stylemaps. After projecting the original image and the reference image through the encoder E into the latent representation, the edited stylemap $\ddot{w}$ can be calculated using the stylemaps w and $\tilde{w}$ together with the mask m.

$$\ddot{w} = m \odot \tilde{w} + (1 - m) \odot w \tag{5.2}$$

The Figure 5.3 shows examples of synthesized images using aligned and unaligned transplantation. It demonstrates that it is possible to copy an arbitrary number and size of areas of a given reference image into another image using the StyleMapGAN.

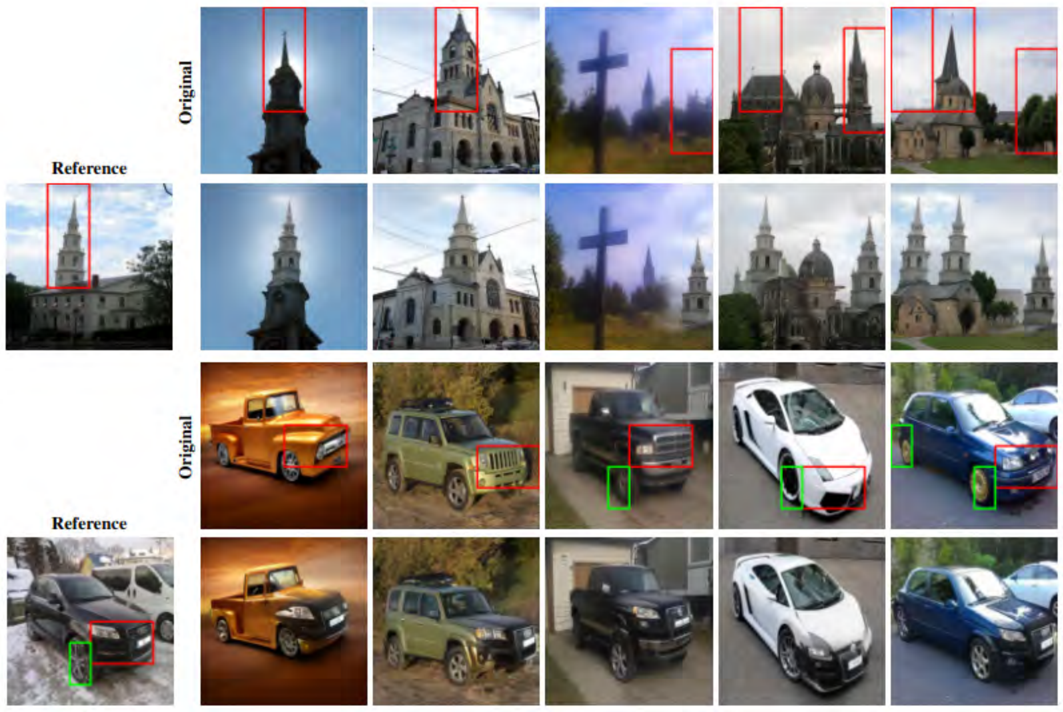


Figure 5.3: Examples of transplantation mode of the StyleMapGAN [35].

Although it is possible that the transplantation can be performed even if the location does not align, the problem that the masks need to have the same size still exists. However, since the goal of the architecture is to simply learn the latent representation of the images within a data set it does not play a crucial role if images of a certain class are underrepresented.

5.2 SAC-GAN

Based on the idea of representing images as style maps introduced with the StyleMap-GAN, the spacial-aware attribute controllable GAN (SAC-GAN) [4] tries to improve the results and simultaneously avoid the main drawbacks. This time it is specially developed for filling missing areas in faces. Again a basic encoder and decoder architecture is used, but this time the shape and location of the masks can be chosen arbitrarily and additionally the style of the filled area is controllable via spatial style maps (Figure 5.4).

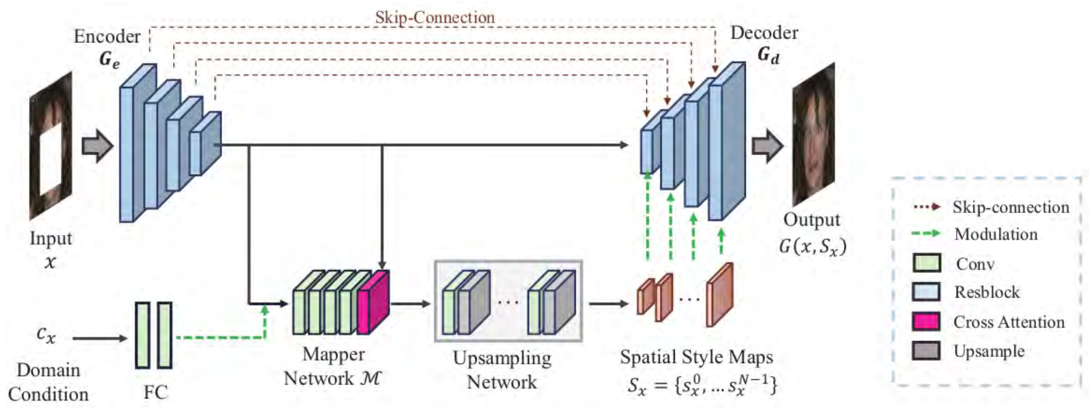


Figure 5.4: SAC-GAN architecture [4].

The domain condition c_x is represented as a one-hot encoded vector and expresses the desired facial attribute, e.g. smiling, angry, etc. In contrast to the StyleMapGAN, the spatial style maps are not the output of the encoder. The style maps were obtained by using the latent representation of the encoder network G_e, which are then resized using the mapper network $\mathcal{M}$ followed by upsampling layers. The resulting spatial style maps are used as additional information in decoder network G_d through applying them to the decoder layers by weight modulation. One special feature is that a cross attention module is used in the mapper network for enhancement of contextual consistency in feature space to achieve long-range dependency between image feature and spatial style map [4].

Since the architecture should be capable of filling missing pixels within a given image the network is trained on the masked images without using the complete images as inputs. This makes is possible not only to change an area in a complete image but also to restore an already damaged one. To enable the model to learn this task, four loss functions are used which are illustrated in Figure 5.5. As it is a GAN-based approach, the typical adversarial loss is used in combination with a discriminator network D. Since the model should learn to only change the pixels within the masked area, a pixel-wise reconstruction loss is adopted. Because the mean squared error has a drawback of making the reconstructed image blurry, the L1 distance between each pixel is calculated in this case [4]. The encoder Ge_e and the mapper network $\mathcal{M}$ are trained with a style consistency loss to ensure that both extract the identical style maps from the reconstructed and the ground-truth image. Between these two style maps the MSE is calculated. The identity preserving loss should guarantee that reconstructed outputs preserve the same identity as the original input image. It is implemented by using a pretrained face identity network R called ArcFace [12].

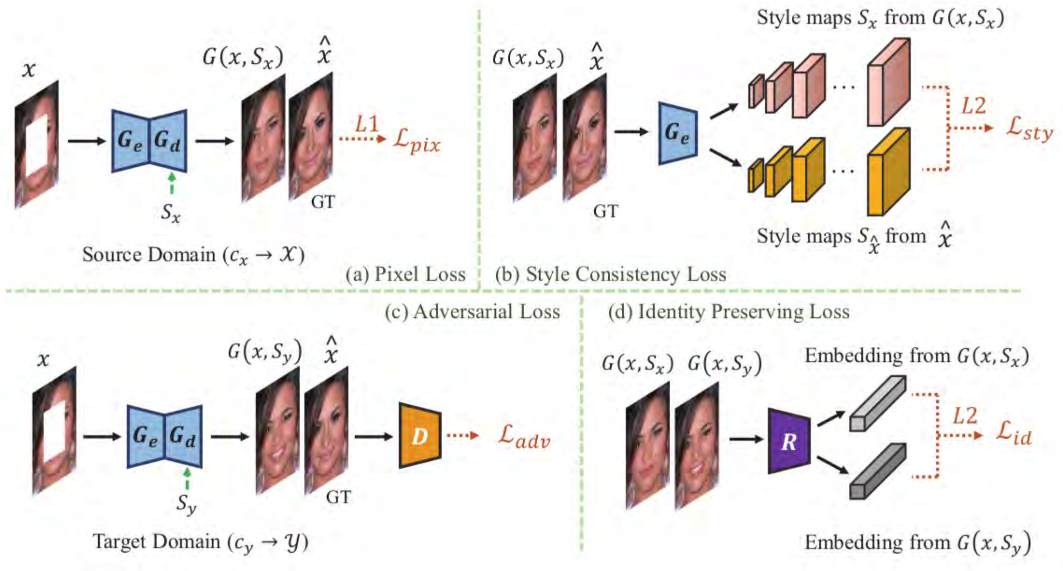


Figure 5.5: Loss function of SAC-GAN [4].

The training procedure is splitted into two phases. In the first phase, the reconstruction is done within the source domain, i.e the image together with the original attribute. Here, the encoder G_e, the decoder G_d and the mapper network $\mathcal{M}$ were updated by using the reconstruction and style preserving loss. In the second phase, the reconstruction is done with a different attribute (target domain). Afterwards, the adversarial and identity preserving losses are used to optimize the networks again, this time also including the discriminator.

In Figure 5.6, two examples generated with the SAC-GAN are shown. The model is able to fill missing areas of arbitrary shape in images of faces and generate outputs with certain attributes.

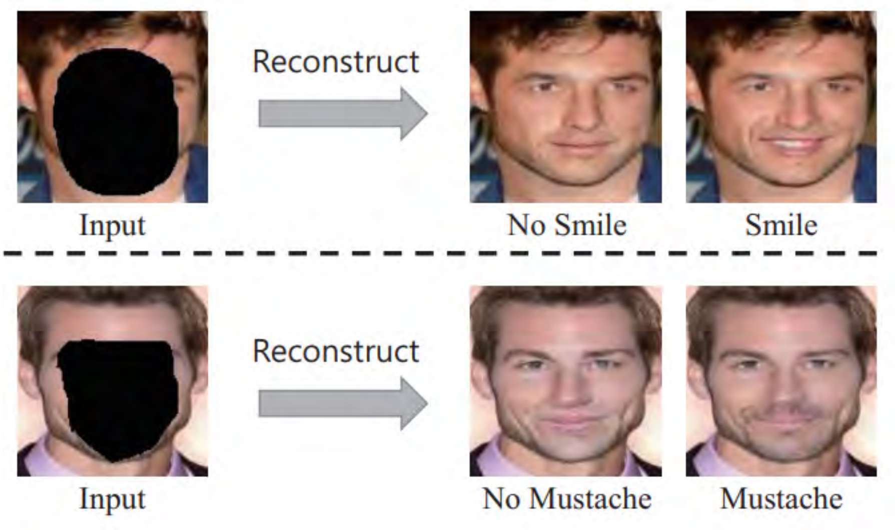


Figure 5.6: Examples of face image inpainting with SAC-GAN and attribute manipulation [4].

As the performance of the model was only investigated on images of faces, it will be interesting to see how it performs on other domains like endoscopic images. The advantage when training a model on face is that the main structure and position of these images are always quite similar. Therefore, this task is much more easier to learn compared to other difficult and varying scenes.

5.3 DDPM

The special thing about DDPMs is that no further or other training process is needed when the model should be used for an inpainting tasks. The only thing that changes is the inference algorithm. Here, the model is guided with the already known image area and thus the model only denoises the unknown masked area (Figure 5.7).

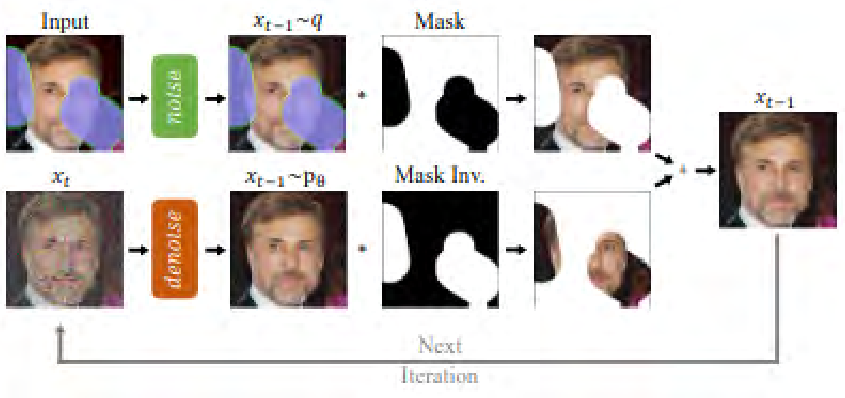


Figure 5.7: DDPM inference for inpainting task [39].

Therefore, the standard U-Net model of the original DDPM [25] can be used and just needs to be retrained on the specific data set. The original inference procedure (Algorithm 2) is later changed such that the backward process is guided with the already known area (Algorithm 3).

Algorithm 1 DDPM training [25]

```
1: repeat
2:     $x_0 \sim q(x_0)$
3:     $t \sim Uniform(1, ..., T)$
4:     $\epsilon \sim \mathcal{N}(0, I)$
5:     $\nabla_\Theta ||\epsilon - \epsilon_\Theta(\sqrt{\hat{\alpha}_t} x_0 + \sqrt{1 - \hat{\alpha}_t}\epsilon, t)||^2$
6: until converged
```

Algorithm 2 DDPM inference [25]

```
1: $x_t \sim \mathcal{N}(0, I)$
2: for $t = T, ..., 1$ do
3:     $z \sim \mathcal{N}(0, I) if t > 1, else z = 0$
4:     $x_{t-1} = \frac{1}{\sqrt{\hat{\alpha}_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\hat{\alpha}_t}}\epsilon_\Theta(x_t, t)) + \sigma_t z$
5: end for
6: return $x_0$
```

Although, the results often already look promising, i.e the transitions between known and inpainted area, the content within the area tends to look semantically not meaningful (Figure 5.8).

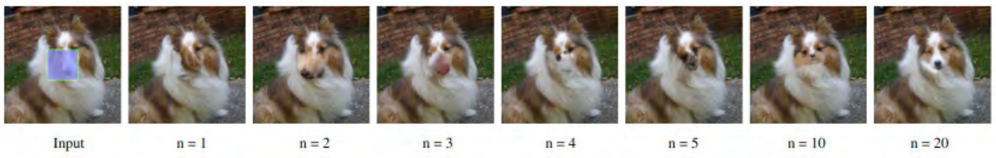


Figure 5.8: Results of Repaint algorithm with different number of resampling steps [39].

This is because the sampling of the known pixels is performed without considering the generated parts of the image, which introduces disharmony and get worse with every time step. One possibility to overcome this problem is to use a resampling approach [39], meaning during the backward process, the sampling process is performed multiple times before advancing to the next time step (Algorithm 3).

Algorithm 3 Inpainting using RePaint approach with U resamplings [39].

```
$x_T \sim \mathcal{N}(0, I)$
for $t = T$ to 1 do
    for $u = 1$ to $U$ do
        $\epsilon \sim \mathcal{N}(0, I)$ if t > 1, else $\epsilon = 0$
        $x_{t-1}^{known} = \sqrt{\hat{\alpha}_t} x_0 + (1 - \hat{\alpha}_t)\epsilon$
        $z \sim \mathcal{N}(0, I) if t > 1, else z = 0$
        $x_{t-1}^{unknown} = \frac{1}{\hat{\alpha}_t}(x_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\epsilon_\Theta(x_t, t)) + \sigma_t z$
        $x_{t-1} = m \odot x_{t-1}^{known} + (1 - m) \odot x_{t-1}^{unknown}$
        if $u < U and t > 1$ then
            $x_t \sim \mathcal{N}(\sqrt{1 - \beta_{t-1} x_{t-1}}, \beta_{t-1} I)$
        end if
    end for
end for
```

Due to this resampling, the model has more time to harmonize the known x_t^{known} and the unknown area $x_t^{unknown}$, but it also increases the inference time depending on the number of resampling steps.

5.4 Anydoor

The overall architecture of the Anydoor model [7] involves many components (Figure 5.9). The main component for the image generation is a pretrained Stable Diffusion model, whereby the input for the guidance differ from the original approach. The conditional tokes which normally come from the CLIP [47] network are replaced by an ID extractor. This ID extractor consists of a pretrained DINO-V2 transformer model [45] as the backbone and additional linear layer in order to align the output to the embedding space of the Stable Diffusion model. Additionally, a detail extractor is used to create detail maps which are then used as information for the skip connection of the U-Net model. This detail extractor is the U-Net encoder of the ControlNet-style [65] architecture.

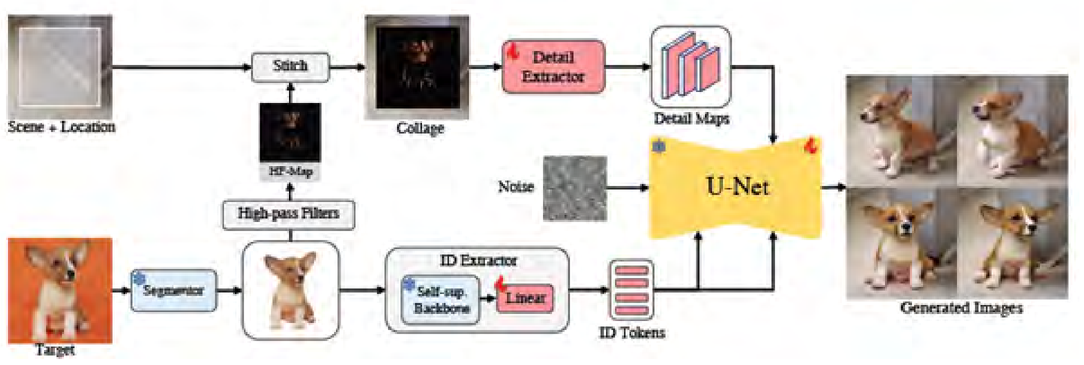


Figure 5.9: Anydoor pipeline [7].

For the training process, not every component of the architecture is trained. In Figure 5.9, the fixed model weights are indicated by a blue snow flake, which means that the segmentor and the backbone (DINO-V2) of the ID extractor are not trained. In contrast to this, the weights of the detail extractor, the linear layer of the ID extractor and the decoder network of the Stable Diffusion model are optimized during the training process.

In order to generate high quality outputs, Anydoor used as many a priori knowledge as possible during the generation process (Figure 5.10). Therefore, a high frequency map of the target object is calculated by a high-pass filter to extract the prominent features. Furthermore, the attention maps generated by the ID extractor additionally refer to important structures within the target image.

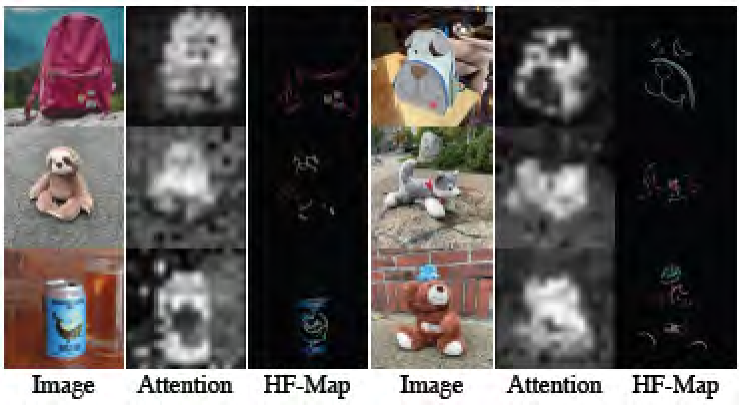


Figure 5.10: Example of focus region of ID extractor and detail extractor [7].

Another interesting approach of the Anydoor model is the data preparation pipeline for training. Since it is important that the model learns how to place an object in any new position regardless of the conditions (geometry, lightning, etc.), it is necessary that images of an object are available from different perspectives. Therefore, images of video sequences are used for the training, as the point of view changes of an object changes a lot over the time. For the training, simply two frames containing the same object but at different times are selected (Figure 5.11). Then the object in one frame is treated as the target (used for the detail and ID extractor) and the other used for the supervision.

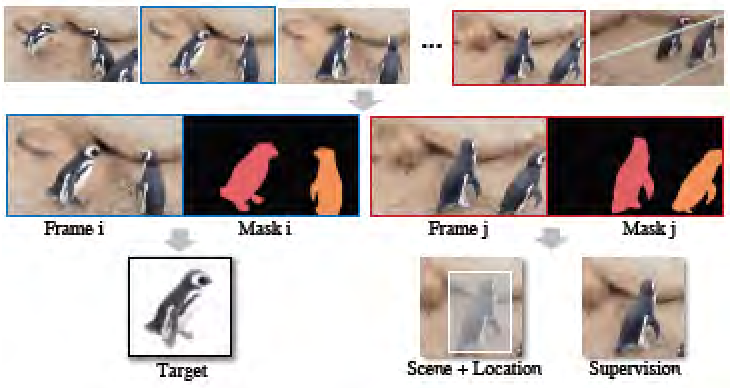


Figure 5.11: Data preparation pipeline for Anydoor [7].

For the inference process, an area must be masked in which the reference object should be inpainted and the actual reference object must be segmented. This means that two masks are needed for the inference which differ from the other architectures which only required one mask. Important to know is that although the mask of the target area can be given in any arbitrary shape, the model will convert this mask into a bounding box covering the whole mask. This means that even though a free form mask was given in the beginning, it is not guaranteed that the final inpainted object matches with the shape of the given mask. But it will definitely be located within the bounding box.

6 Experiments

Since the generation of virtual pathologies is defined as a two-stage problem, the aim is first to find the best method for the inpainting of a pathology in the initial frame of a video sequence. Therefore, the four suggested models will be used and the results evaluated. Once the best method is selected, the inpainting results are used for the optical flow part in order to generate temporal consistent videos. The training and experiments were run on a workstation with the following specifications.

- AMD Ryzen 7 5800X @3.6 MHz
- NVIDIA GeForce RTX 3090 24GB
- 64GB RAM @3600 MHz
- Ubuntu 22.04.3 LTS

For the training procedure, 63 videos (204866 frames) of real patients are available. Whereby, 61 videos (202780 frames) belong to healthy patients and the other two videos contain ulcera. Although in both cases the diagnosis is an ulcer, their visual appearance differ a lot. Hence, these videos were splitted into three groups (Figure 6.1). The first group contains only healthy looking patients. Group two holds a superficial ulcer (1098 frames) and the third group shows a deeper ulcer (988 frames).

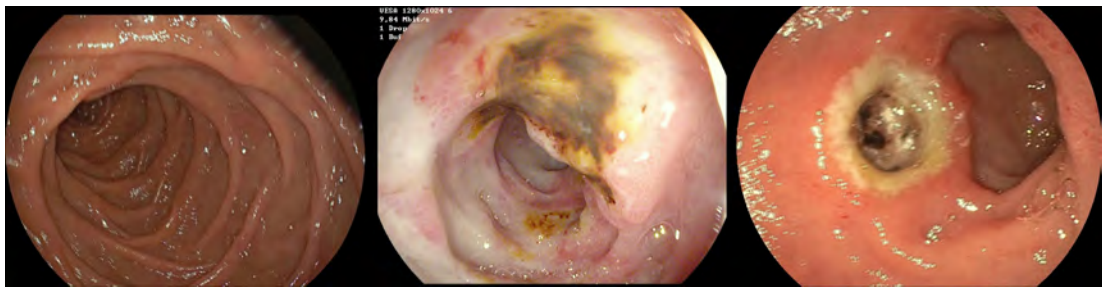


Figure 6.1: Examples images of the different training classes. From left to right: healthy patient, superficial ulcer, and deeper ulcer.

The training of the StyleMapGAN follows the standard procedure as described in the original paper [35]. All hyperparameters were maintained as suggested and during the inference, the transplantation mode is used for local editing. The SAC-GAN is trained twice with two different approaches. The first attempt only uses random generated circles as masks whereby, the missing area is selected randomly all over the image. For the second attempt, the pathologies in the real images of the data set are segmented and the corresponding masks are used to select the missing pixel values. Since there are no such segmentation masks available for the healthy patients, synthetic masks (Figure 6.2) are generated using another generative neural network. For this, the ProGAN [32] model is trained with the real segmentation masks. This technique was already used for inpainting of polyps and showed promising results [17].

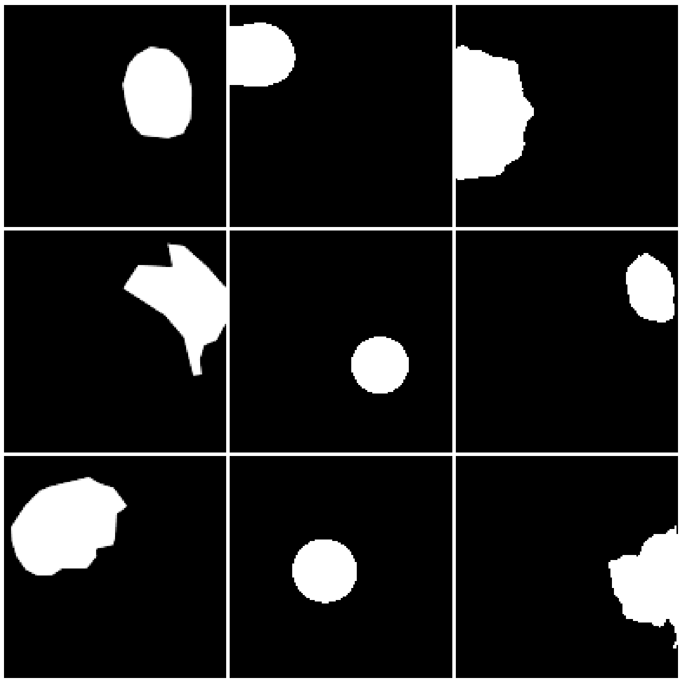


Figure 6.2: Generated masks for training process. Left: real masks; middle: masks generated with circles and random noise; right: masks generated with ProGAN.

For the training process itself, all the hyperparameters are maintained. However, the identity preserving loss is not taken into account since the face identity recognizer of the original work is not applicable to our scenario.

As previously noted, the original model is employed in relation to the DDPM approach. It is retrained on the given data set and the inference procedure is adapted according to the algorithm presented in chapter 5.3. During the training, the model is conditioned in 90% of the training steps and the rest is trained unconditioned, as suggested in [26]. For the inference step, the EMA model is used as final model.

Since the architecture of the Anydoor model is complex, it requires a lot of memory and time for the training. The authors recommended to use a NVIDIA A100, which has about 80 gigabyte of memory. Retraining is therefore not suitable with the available hardware setup and therefore only the pretrained model is used to explore the potential of the approach. For model inference, the gap between the different image resolution must be closed. Currently, the available models for transforming the dummy into synthetic patients output images of the size 128x128 pixels. Since the Anydoor model expects inputs images to have a size of 512x512 pixels, the transformed images must be resized before they can be used. One method to do this is to use cubic interpolation, which is simple but the quality of the resized images can be decreased. Another approach is to use neural networks which are trained for the super resolution task. Such models can upscale the input image whereby, the quality of the images is preserved (Figure 6.3).

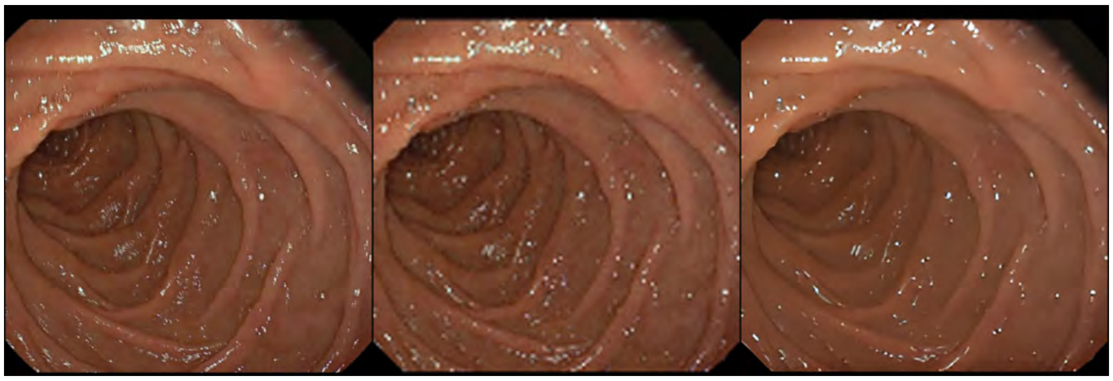


Figure 6.3: Transformed dummy images resized from 128x128 to 512x512 pixels (original left) by cubic interpolation (middle) and Real-ECRGAN (right)

The optical flow between two consecutive frames is calculated by using a variational non-linear intensity-based registration approach [63]. Whereby, the algorithm is run a maximum of 800 iterations with an early stopping delta of 1e-5. The registrations is done using the coarse-to-fine approach with three levels. The intermediate results of the vector field will be smoothen by using a Gaussian kernel with $\sigma_x = \sigma_y = 5$ pixel. Additionally, the the pretrained deep learning network RAFT [58] is used as an alternative.

For the video generation process the following procedure is implemented in order to ensure the temporal consistency of the inpainted pathology. First, a pathology is chosen and inpainted in the initial frame ($I_{0,inpainted}$) of a video sequence (V). Second, the optical flow field (f) is calculated using the method (F) mentioned above. After that, the corresponding segmentation mask (M) and cropped pathology (P) is warped and inserted in the subsequent frame I_{i+1} using the Hadamard product $\odot$.

Algorithm 4 Procedure of video generation with pathology

```
Require: $I_{0,inpainted}$, $V$, $M$, $P$, $F$
1: $V[0] \leftarrow I_{0,inpainted}$                    ▷ Assign inpainted frame as initial frame
2: for i=0 ;i<len(V)-1; i++ do                          ▷ Loop through all frames
3:     $f_{i \to i_{i+1}} \leftarrow F(V_i, Vi_{i+1})$
4:     $M \leftarrow$ warp M with $f_{i \to i_{i+1}}$                  ▷ Get new mask
5:     $P \leftarrow$ warp P with $f_{i \to i_{i+1}}$                  ▷ Get updated pathology
6:     $V[i+1] \leftarrow M \odot P + (1 - M) \odot V[i+1]$    ▷ Copy pathology into frame
7: end for
8: return $V$                       ▷ V is the video sequence with inpainted pathology
```

7 Results

7.1 StyleMapGAN

Once again, the StyleMapGAN simply learns to encode the images into a latent space and than replace the masked region with the desired features. Therefore, it is to be expected that the inpainted pathologies have a good quality although the data set does only contain very few training examples. The assumption is underpinned by our findings (Figure 7.1 and 7.2).

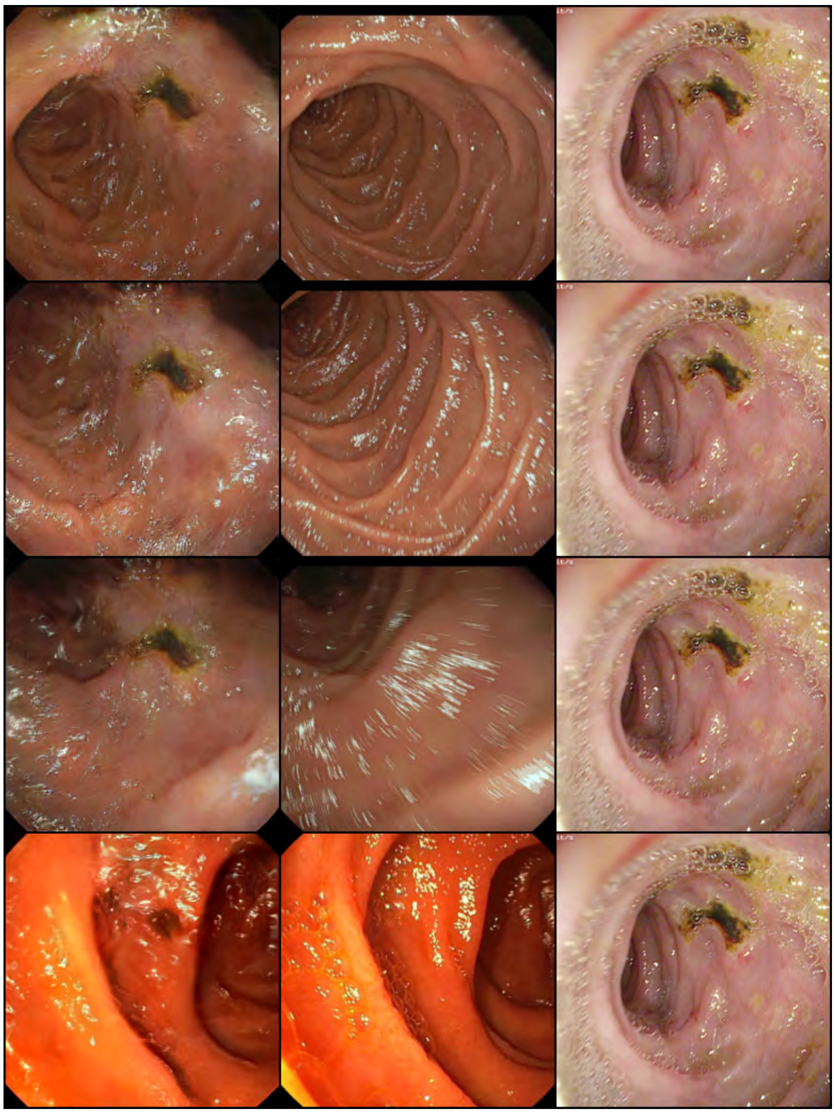


Figure 7.1: Results of StyleMapGAN for superficial ulcer. The same reference pathology (right column) inpainted in four different images (middle column). The first column contains the final inpainted images.

The inpainted region looks quite close to the reference pathology. However, it is clearly visible that not only the masked region but also parts of the surrounding area are changes during the inpainting process.

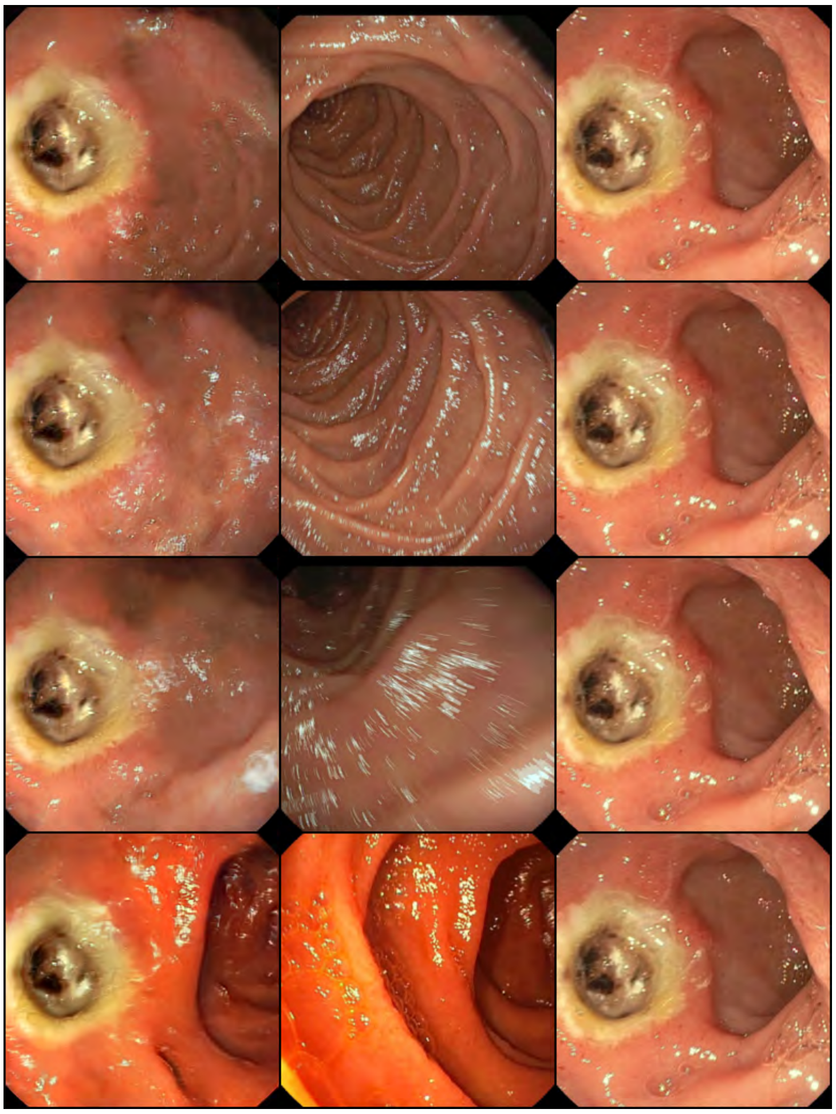


Figure 7.2: Results of StyleMapGAN for deeper ulcer. The same reference pathology (right column) inpainted in four different images (middle column). The first column contains the final inpainted images.

Furthermore, the observation of the authors [35] that the quality of the results highly depends on the structural similarity between the target and reference image can be confirmed. The greater this difference is the more of the surrounding area is changed and therefore the quality is very poor. This can be clearly seen in Figure 7.2 in the first three rows. In contrast the structural similarity between the target and reference image in the last row is higher and the inpainted result looks more reasonable.

7.2 SAC-GAN

The results of the first attempt (using random areas masked with circles) are shown in Figure 7.3. It is clearly visible that the model fails to fill the missing area properly. The transitions are not smooth and thus the shape of the original mask is still visible. Furthermore, the inpainted areas do not show the expected pathologies. It seems that the model arbitrarily inpaint areas of the mucosa corresponding to the particular patient.

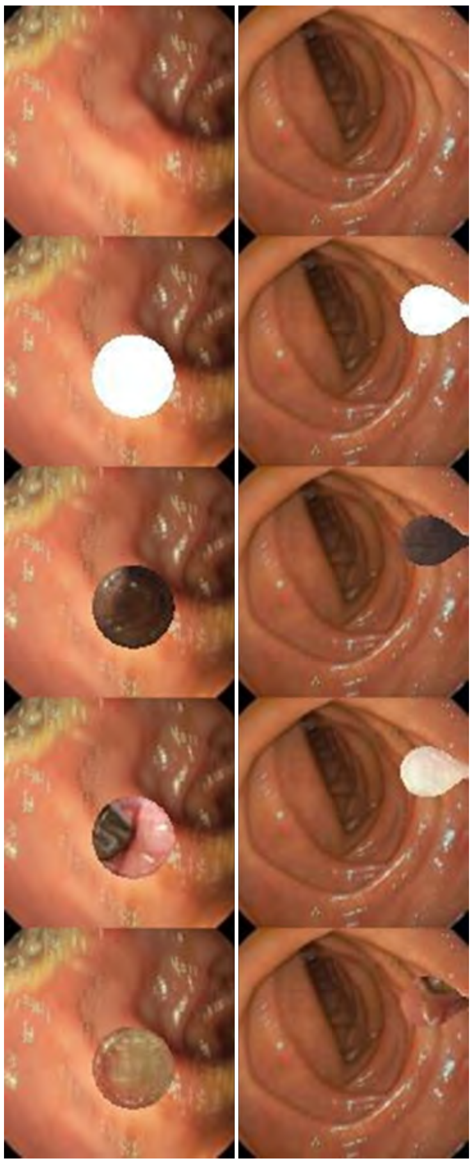


Figure 7.3: Results of SAC-GAN with circle masks (left) and randomly generated masks (right). From top to bottom: original image, original image with mask, healthy, superficial ulcer, deeper ulcer)

The model is therefore not able to fill the masked areas in a meaningful way when training only with randomly generated masks. But it seems that the model learned to only fill the masked area without changing the adjacent pixels. The results of the second attempt (using segmentation masks of the pathologies) are illustrated in Figure 7.4.

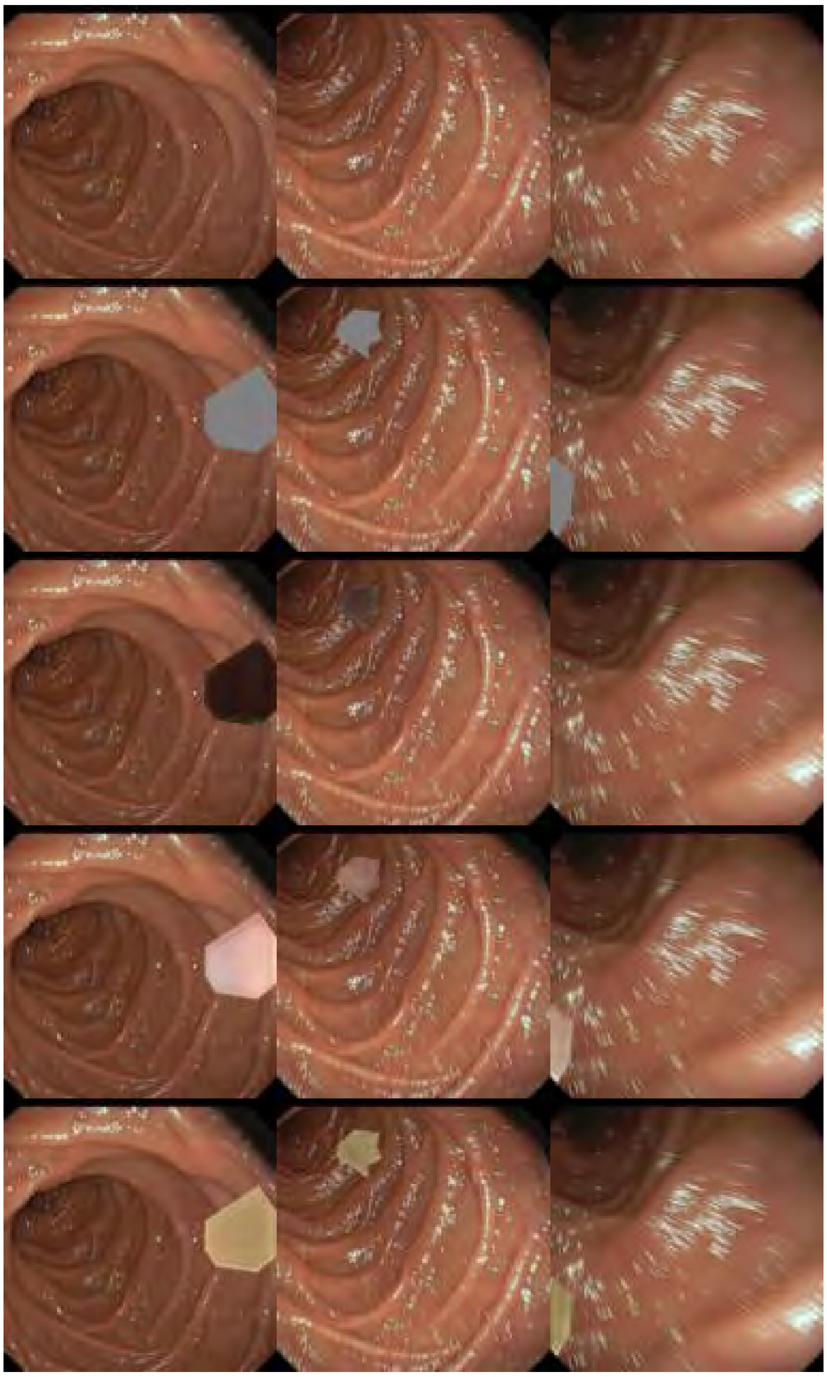


Figure 7.4: Results of SAC-GAN with segmentation masks. From top to bottom: original image, masked original image, inpainted healthy patient, superficial ulcer, deeper ulcer.

Interestingly, the updated model appears to be capable of filling defect areas with healthy mucosa, which can be seen in the third row of Figure 7.4. However, the results of the other two cases still do not show the desired pathologies. The inpainted areas for the superficial ulcer again look like the mucosa of the patient. In this respect, the areas of the deeper ulcer show some matching colors with the actual ulcer but the main features were not visible. As the results for the healthy patient look promising and the inpainting of the pathologies still do not work, it leads to the conclusion that the model is not able to learn to transfer a pathology into an area surrounded by a different mucosa.

7.3 DDPM

By use of DDPMs, the filled region looks realistic and the transitions to the known area is smooth and consistent (Figure 7.5). Also only pixels within the masked area were changes and no other pixel in the surrounding area like observed with the StyleMapGAN. As a result, it is even hard to find the inpainted areas. However, the filled regions do not show the expected pathologies. Since the inpainted areas for the ulcer do not distinguish from the healthy inpainted pixels it seems that the DDPM has the same problem as observed with the SAC-GAN.

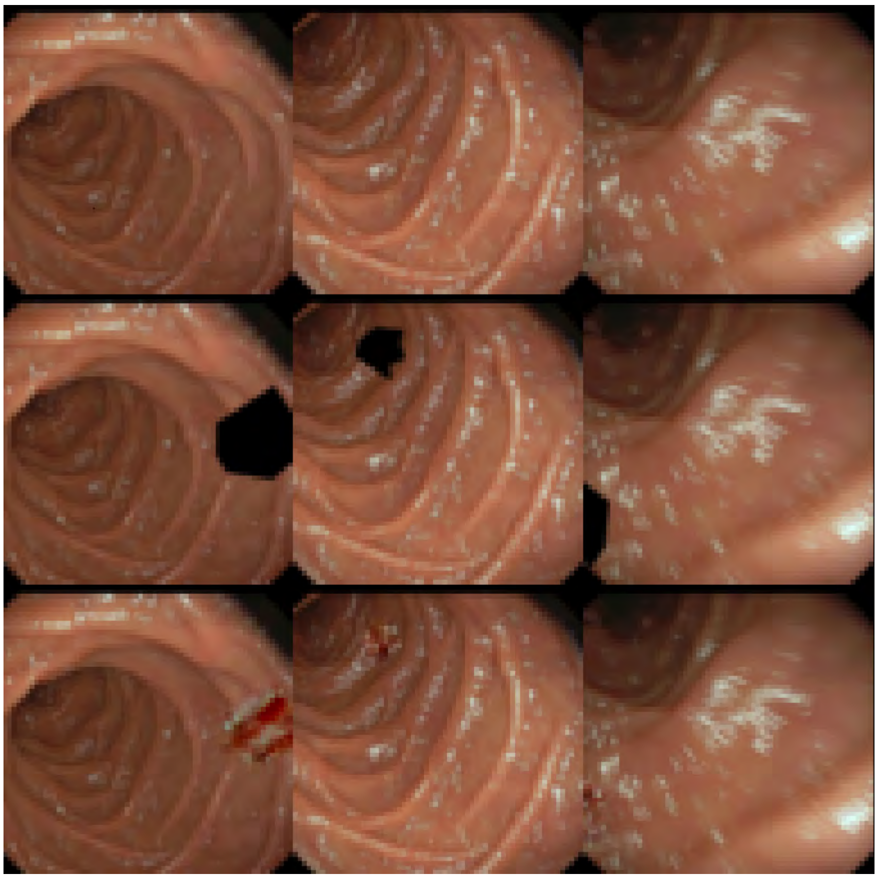


Figure 7.5: Results with DDPM as inpainting algorithm (original images (top), masked images (middle), inpainted images (bottom, from left to right: healthy, superficial ulcer, deep ulcer).

7.4 Anydoor

Even though Anydoor is applied without any retraining and the original model has been trained on entirely different data, a remarkable performance is achieved. Figure 7.6 shows some examples of inpainted pathologies. For the sake of clarity, only the original masked image, the reference image and the output of the Anydoor model are shown. The chosen reference area is represented with a red line in the reference image.

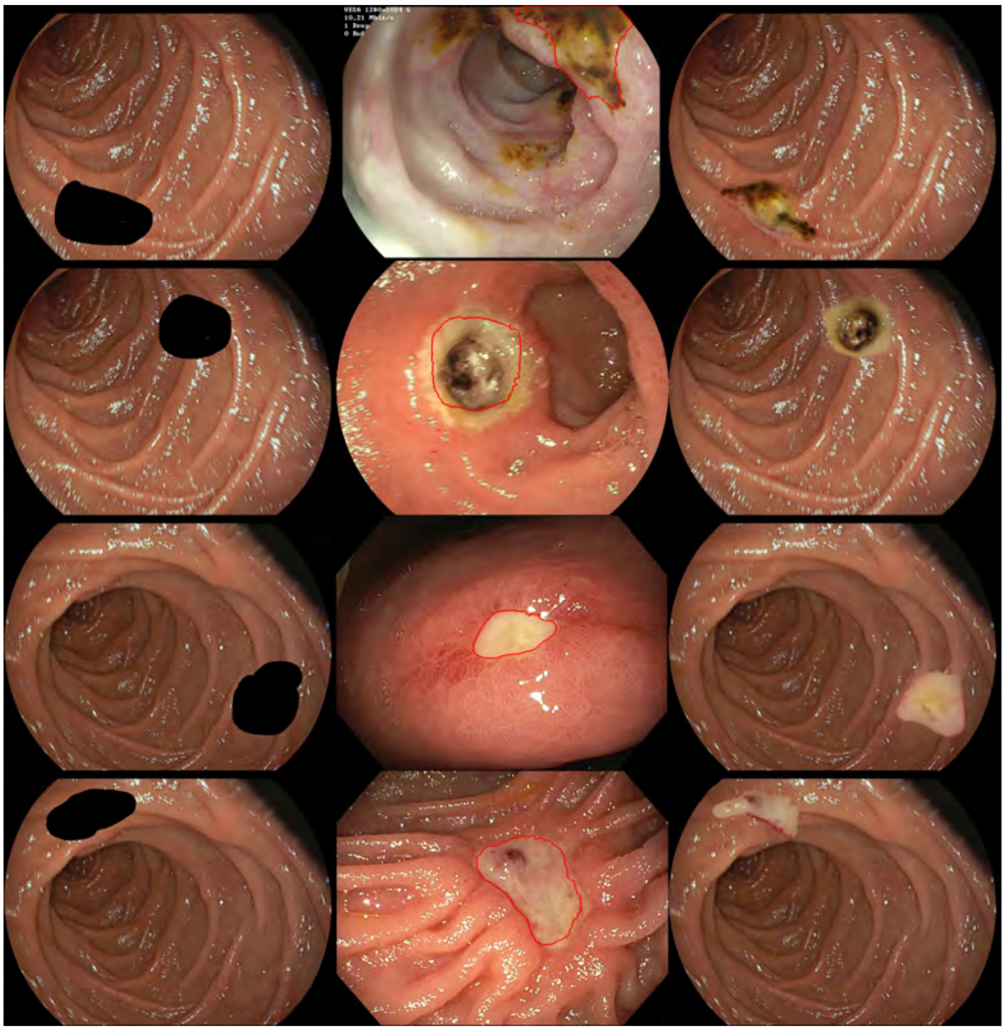


Figure 7.6: Results with Anydoor algorithm (From left to right: Original masked image, reference area (red line), inpainted image).

The features of the reference areas are well presented and the transitions to the known areas are smooth and consistent. The quality is also good in cases where the shape of the masked region and the reference pathology differ a lot. It also seems that the model is robust against the different backgrounds (mucosa) and thus is able to transfer a pathology to a completely new background. With this, it perform significantly better than all the other models discussed before. Nevertheless, there are cases in which the model does not deliver satisfactory results (Figure 7.7).

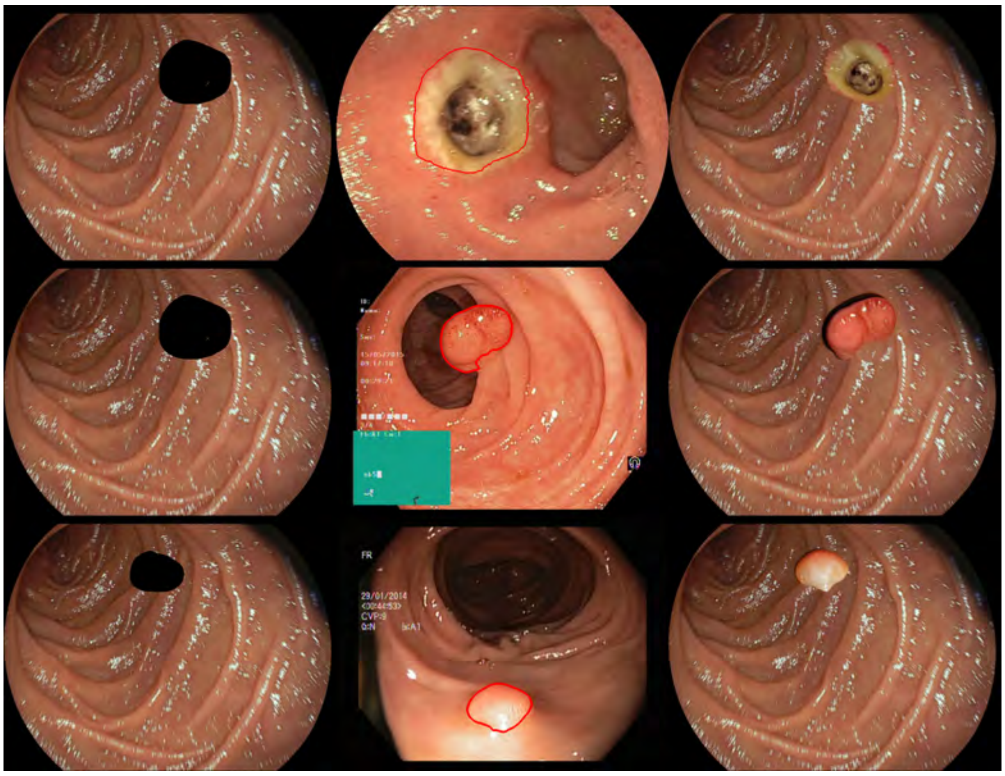


Figure 7.7: Failure cases of Anydoor (From left to right: Original masked image, reference area (red line), inpainted image).

In cases where the mucosa of the target and reference image differ and the reference region also contains parts of it, the transitions from the pathology to the background are not that smooth anymore (see Figure 7.7 top row). For pathologies that are not only flat, but also stick out into the lumen (e.g. polyps), the quality of the inpainted region is decreased. Although the main features are retained, the spatial representation is poor (see Figure 7.7 middle and bottom row).

7.5 Video generation pipeline

Since the Anydoor model shows the best results, it will be used for the further video generation. In order to generate the videos according to the procedure (Algorithm 4) explained in chapter 6, some work needs to be done to make the different models compatible. First of all a dummy frame will be transformed into the patient domain using the MoCycleGAN. Afterwards the Real-ESRGAN will be used to increase the image resolution from 128x128 to 512x512 pixels. The high resolution image will then be used as target input for the Anydoor model in which a given reference pathology will be inpainted. Since the specified target mask and the actual shape of the inpainted pathology can differ, the SAM architecture is used to extract the final mask of the inpainted pathology. Subsequently the mask and the pathology is warped using the calculated optical flow between two consecutive frames whereas the warped pathology is inserted into the new frame. The whole pipeline is illustrated in Figure 7.8. Since the use of the RAFT model as an alternative to calculate the optical flow did not yield good results, it will not be considered and only the variational registration (VARREG) [63] approach is used.

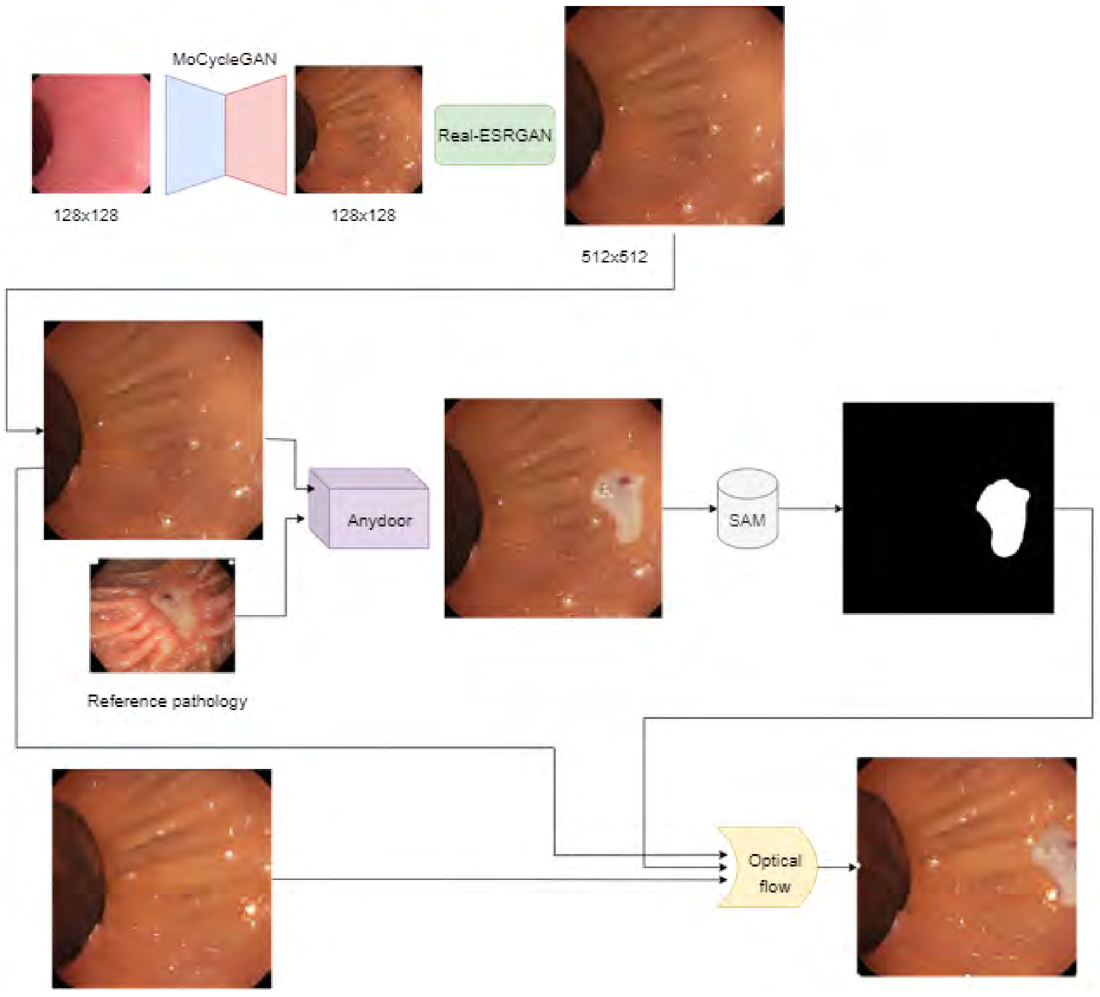


Figure 7.8: Process of the video generation.

Although only the Anydoor model was used for the inpainting process the inference times and number of model parameters of all models discussed are given in Table 7.1 for a better evaluation.

Table 7.1: Inference times and model parameters.

Model	Inference time (ms)	Model parameters	Input image size
Inpainting			
StyleMapGAN	345.16	97M	256x256
SAC-GAN	6.04	57M	128x128
DDPM	8140	23M	64x64
Anydoor	7273	245M	512x512
Optical flow			
VARREG	548	N/A	N/A
RAFT	29	5M	N/A
Dummy transformation			
tempCycleGAN	6.67	8M	128x128
MoCycleGAN	2.12	8M	128x128
Others			
Real-ESRGAN	50.3	17M	512x512
SAM	30.5	641M	N/A

It is clear to see that GAN´s are significantly faster than DDPMs regarding the inpainting tasks. However, the Anydoor model is faster than the standard DDPM, although it has nearly 10 times more model parameters and works on a 8 times higher image resolution. This clearly shows the impact of an optimal noise schedule and gives hope for future speed ups. Also interesting is the comparison of the inference time between the VARREG approach and the RAFT model. The RAFT model is nearly 19 times faster. Unfortunately, initial tests have shown that the model is not able to calculate the optical flow when using the available pretrained models. As the authors have already said, the model needs a proper retraining to adapt it to the specific use case [58]. If it then achieves the desired results, then this approach will be very interesting in the future in terms of real-time capability. For the image-to-image transformation itself both models show low inference times. Nevertheless, the MoCycleGAN model is 3 times faster. The Real-ESRGAN has a measured inference time of 50.3 ms which is quite slow. Since it is only needed to bridge the gab between the output resolution of the MoCycleGAN (and tempCycleGAN respectively) and the input resolution of the Anydoor model, another possibility should be looked at. Ideally, the result of the image-to-image transformation should correspond directly to the requirements of the Anydoor model.

Adding up all the relevant inference times for the video generation, it results in a total time of 600.42 ms for one frame.

$$T_{total} = T_{MoCycleGAN} + T_{Real-ESRGAN} + T_{Varreg} = 600.42ms \tag{7.1}$$

Due to the fact that the pathology is inpainted only in the first frame, the inference time of the Anydoor and SAM model only applies there. For the first frame the total inference time amount to

$$T_{total,first} = T_{total} + T_{Anydoor} + T_{SAM} = 8770.92ms \tag{7.2}$$

Both times are significantly to high in order to use this pipeline in a real-time system. In this case the total inference time should be less than 40 ms in order to achieve a frame rate of 25 frames per second. Despite this the inference time of the Anydoor model is by far the slowest.

8 Discussion

8.1 Quality of inpainted pathologies

Two of the four model architectures showed visiually convincing results. SAC-GAN failed with respect to the inpainting task. The DDPM managed to create smooth boundary transitions but was not able to inpaint the desired pathologies. In contrast, StyleMap-GAN was able to inpaint the targets but due to the changes in the surrounding area and the restriction in the positioning of the pathogies it does not seem to be the right approach to this task. The Anydoor architecture produced the best results and the target area can be chosen independently from the reference. Another advantage is the fact that the model performs well without retraining, even though pathologies are not included in the training data, suggesting a good model robustness as well as potential for further improvement. Although it fits exactly the needs regarding the inpainting tasks the architecture has two drawbacks. First of all the model has by far the most model parameters in comparison to the other approaches, which leads to a no lightweight implementation and longer training times in the future. The other and more important disadvantage is that the inference time is enormously outside of real-time capability. In order to be able to use these model in a deep-learning based training simulator, inference time must be extremely reduced. Fortunately, this problem is already tackled by current research [1, 3] and it is important to incorporate these findings in the future.

8.2 Inpainting artifacts

In some cases, artifacts were observed in the inpainted area. Although, it does not happen with every pathology for some constellations of reference pathology and masks artifacts like a human eye appeared within the inpainting area (Figure 8.1).

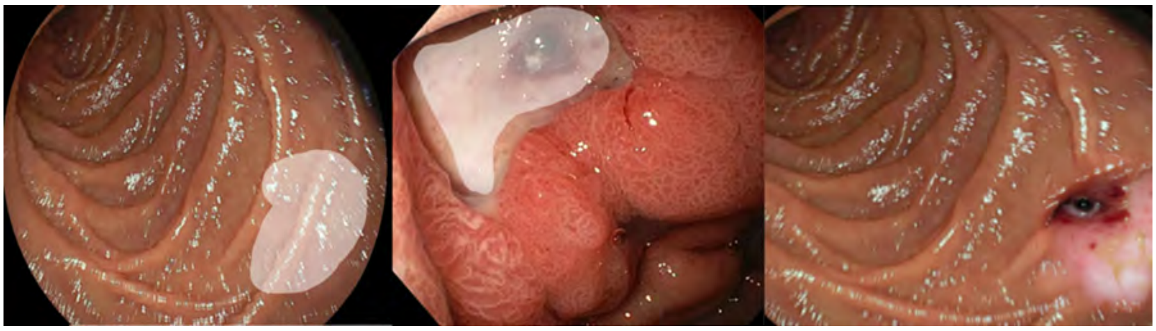


Figure 8.1: Artifact of an eye within the pathology after inpainting.

This behaviour can be explained by the fact that the Anydoor model was trained on data containing images of human faces and since the model was still not trained on medical images yet these artefacts are quite understandable. Therefore, it is reasonable to assume that this problem can be mediated or overcome by domain-specific fine-tuning in future research.

While experimenting with different masks for the same pathology it became clear that the quality of the inpainted pathology depends on the chosen mask. When looking at the mentioned failure cases of the Anydoor model as illustrated in Figure 7.7 (top row), it is visible that the transition between healthy mucosa and the pathology does not look realistic. This is because the masked area of the reference pathology also includes some pixels which belongs to the patients mucosa and the model tries to also include this into the target area. This is an undesirable behaviour since the quality of the inpainting should not be lowered in cases were the segmentation of the pathology is not that accurate. Fortunately, using objects which the model already knows from the training process do not show these artifacts (8.2).

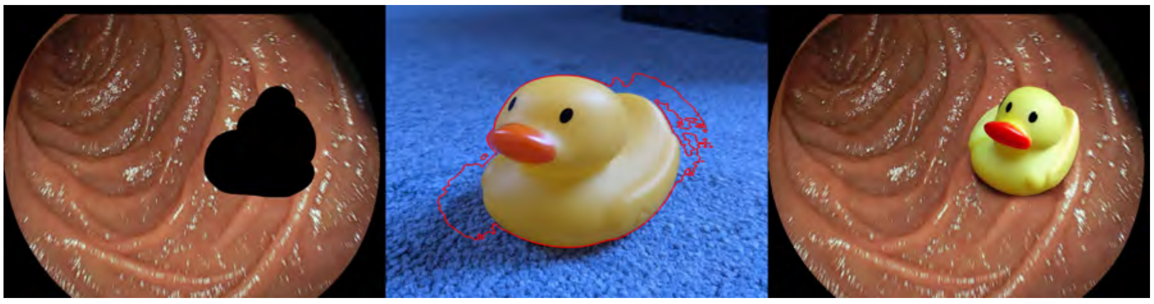


Figure 8.2: Result of inpainting an object which was already included in the training data.

Although the segmentation mask of the duck (red line around the duck) not accurate and the area contains a lot of pixel from the background, the inpainting result does not show any artifacts. This observation leads to the assumption that these types of artifacts can also be avoided when fine-tuning the model on the given pathologies.

8.3 Temporal consistency

Due to the way how the video frames were preprocessed, all frames contain black pixel values in each corner. It turns out that this leads to problem when calculating the optical flow field since at these points there is a hard transition and thus the algorithm is struggling to maintain the temporal consistency when the pathology reaches one of the corners (Figure 8.3).

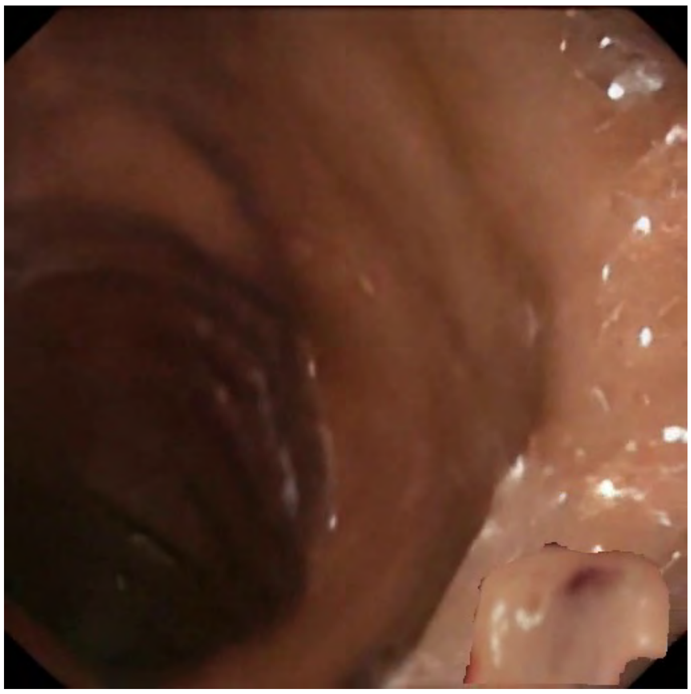


Figure 8.3: Issue when warping pathology into corner of image.

In cases were a pathology should move out of the image in one of the corners the pathology is not able to fade out of the scene. It will be stuck in the corner and shows unnatural behavior which decreases the temporal consistency. In the future a solution must be found. One way to solve this would be to find another preprocessing which avoids black corners. Alternatively, the black corners could be removed when calculating the optical flow by a suitable padding of the pixels nearby. Furthermore, it was observed that prominent features such like strong light reflections which were added during the image-to-image transformation lead to unnatural movements of the pathology. Since they are easy to register, they will have a large impact on the resulting motion field and as the movement of these reflections do not fit with the expected behaviour of the pathology the temporal consistency will be poor. An attempt was therefore also made to calculate the motion field on the original dummy frames, but since the dummy frames show only very few features the optical flow can not be calculated well and especially strong movements will not be registered.

A further problem is that the optical flow does not consider the structural components of the video sequence. This leads to implausible movements and light conditions of the pathology when the scene changes a lot. Also the appearance of the pathology do not adapt its visual appearance (shadow, illumination, etc.) to the current scene which will result in a bad temporal consistency. The Figure 8.4 shows the starting frame of a video sequence with an inpainted area (left) and the same pathology only a few more frames further on (right). It can be observed that the size and position of the pathology has been adapted to the progression of the videos due to the optical flow, but not the illumination.

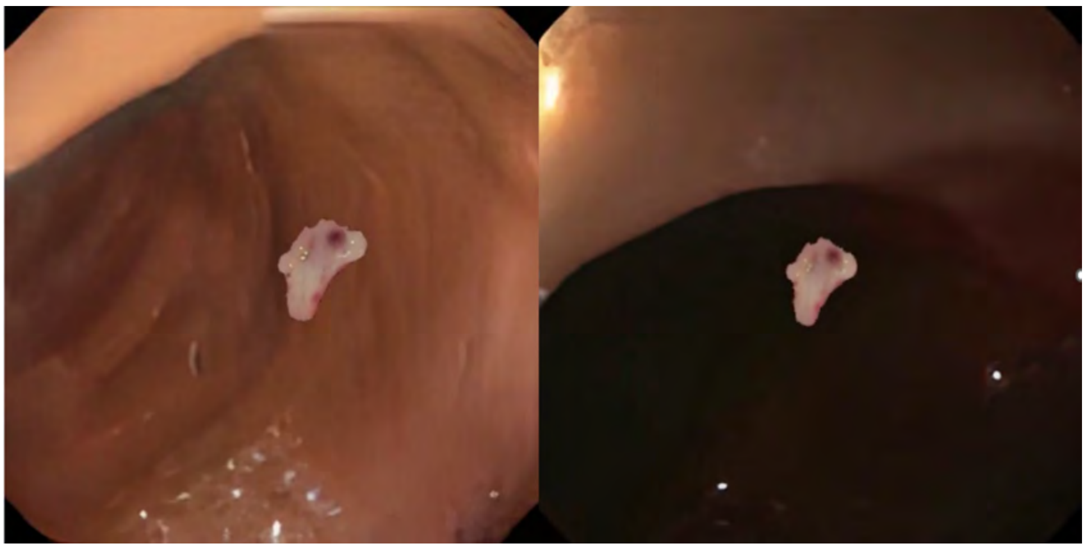


Figure 8.4: No adaption of the inpainted pathology to the background.

The last two problems strengthen the idea that the temporal consistency should ideally be ensured also by the inpainting network. Further work could focus on this and expand the architecture of the Anydoor model such that it can consider also previous frames during the inpainting process. But even if the inpainting shows temporal consistency also in complex scenes there is still the question when and were the pathologies should be inpainted. Until now the videos start already with a pathology, but it is hard to slowly fade it into visibility area. Application of fiducial markings to the physical training dummy would serve as a reasonable approach to this problem.

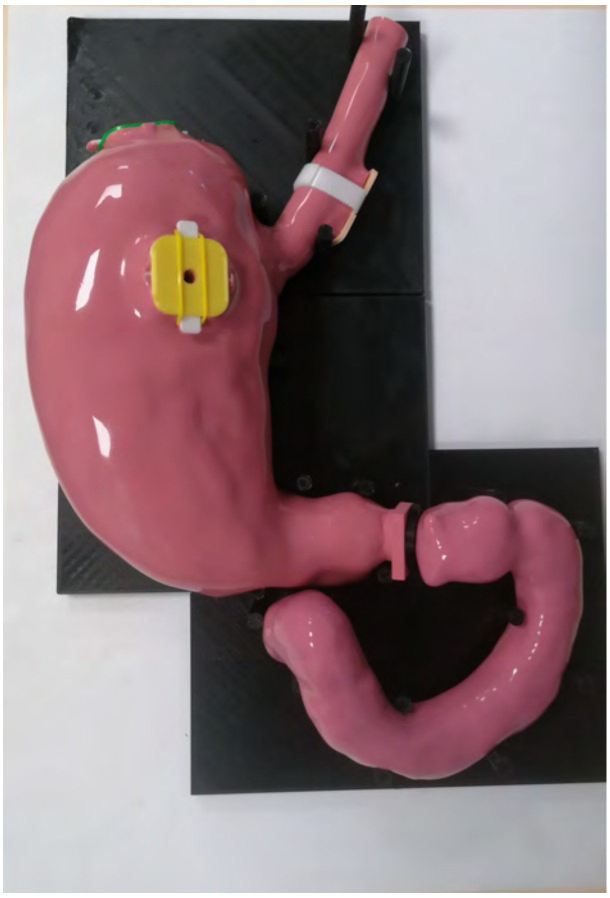


Figure 8.5: Endoscopic training dummy with inlay parts (yellow) for pathologies.

These markings in the model above can be exchanged quickly but also suffer from poor realism (Figure 8.6). Such models combined with the approach discussed here could lead to a significantly improved training experience.

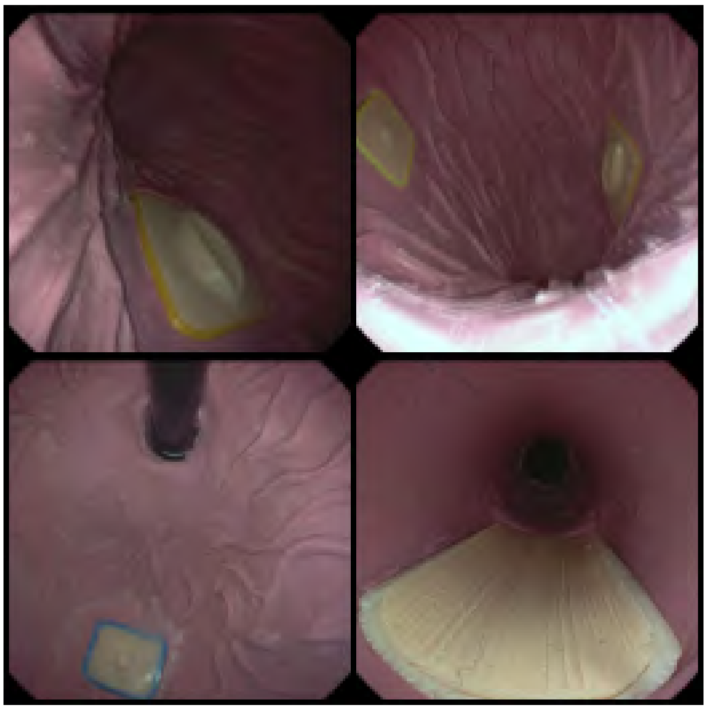


Figure 8.6: Inner view of the pathology inlays.

Although such inlays would help to overcome the difficulties to find suitable positions for the pathologies they would lead to a limited number of training scenarios, since the positions are fixed within the dummy.

9 Outlook

In following work, mainly three challenges could be tackled in order to improve the results. Since the whole idea only makes sense if it can be used in practice, it must be real-time capable. Therefore, it is necessary to decrease the inference time by optimisation of the backward process of the diffusion model. However, the measured time is far away from real-time one first step could be to experiment with the already optimizable parameters like sampling steps and scheduler. Furthermore, it would be interesting how the model performs on other hardware intended especially for running deep learning models.

This leads to the second challenge which is the implementation of the pipeline on a devices that can be used in the hospitals and training facilities. The hardware needs be able to run the model, capture the images coming from the endoscope and feed the transformed image back into the clinical setup simultaneously. Since the models used only for image-to-image transformation is already real-time capable the pipeline could be implemented without the integration of pathologies for first experiments.

The third major challenge is to improve the temporal consistency of the pathology even in complex video scenes. As this also includes the adjustment of the inpainted area to the changing illumination and structural components, it is necessary that the temporal consistency is covered by the inpainting model itself. Only moving the initial inpainted pixels using the optical flow does not allow any suitable adaption to the changing background. Fortunately, some progress has already been made in this area on which further work could base on [23, 67].

10 Conclusion

It is important that the practitioner is able to achieve both the safe handling of the endoscope and precise observation of the unknown environment simultaneously. Otherwise suspicious pathologies or disease pattern, will not be recognized which can have serious consequences for the patient. Simultaneity of these tasks cannot be trained with conventional silicone simulators until now. A first study verified that the discussed architectures for image-to-image transformation improving the realism of the endoscopic training simulator, but the integration of pathologies does not show promising results yet. The current approach tries to do the image-to-image transformation and generate the pathology using one model. This has multiple drawbacks like not having control over the positioning and movement of the pathology which leads to a poor temporal consistency. In this work, the problem of integrating pathologies is defined as two sub-tasks, whereby the first one is the general inpainting of a pathology within an image and the second task is to ensure the temporal consistency. It turns out that there are many approaches for the general inpainting problem, but the ones based on diffusion models perform significantly better than the GAN based architectures.

The described video generation pipeline allows to inpaint a given reference pathology into a frame and also is able to maintain the temporal consistency in simple scenarios using the optical flow between the consecutive video frames. In more difficult scenarios, i.e large changes in illumination and the structures within a video sequence, the optical flow approach is not sufficient to maintain a natural movement of the pathology as a result if which the quality of the generated videos is very poor. The discussed artifacts which occurs in some cases decreases the visual appearance of the pathologies, but as discussed these could be avoided by fine-tuning of the model with the medical image data.

Although it improves the quality of the generated videos regarding the realism it leads to a drastically increased inference time. Therefore, the pipeline is not real-time capable yet and needs further optimisation. As the hardware is constantly improving and current research deals with making the architectures faster, the inference time could be decreased which makes the pipeline still promising.

Even though the open challenges are not easy to solve, the implemented approach shows a high potential to integrate pathologies into Deep Learning-based endoscopic training simulators and thus allows to cover various training scenarios in the future.

Bibliography

[1] Andy, Shih ; Suneel, Belkhale ; Stefano, Ermon ; Dorsa, Sadigh ; Nima, Anari: Parallel Sampling of Diffusion Models. In: Oh, A. (Hrsg.) ; Neumann, T. (Hrsg.) ; Globerson, A. (Hrsg.) ; Saenko, K. (Hrsg.) ; Hardt, M. (Hrsg.) ; Levine, S. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 36, Curran Associates, Inc., 2023, S. 4263–4276

[2] Arias, Fernando ; Zambrano Nuñez, Maytee ; Guerra-Adames, Ariel ; Tejedor, Nathalia ; Vargas-Lombardo, Miguel: Sentiment Analysis of Public Social Media as a Tool for Health -Related Topics. In: *IEEE Access* 10 (2022), 01, S. 1–1

[3] Bao, Fan ; Li, Chongxuan ; Zhu, Jun ; Zhang, Bo: *Analytic-DPM: an Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models.* 2022

[4] Cha, Dongmin ; Kim, Taehun ; Lee, Joonyeong ; Kim, Dajin: *SAC-GAN: Face Image Inpainting with Spatial-Aware Attribute Controllable GAN.* S. 202–218, 03 2023. – ISBN 978-3-031-26292-0

[5] Chen, Hongqian ; Guan, Mengxi ; Li, Hui: ArCycleGAN: Improved CycleGAN for Style Transferring of Fruit Images. In: *IEEE Access* 9 (2021), S. 46776–46787

[6] Chen, Ting: *On the Importance of Noise Scheduling for Diffusion Models.* 2023

[7] Chen, Xi ; Huang, Lianghua ; Liu, Yu ; Shen, Yujun ; Zhao, Deli ; Zhao, Hengshuang: Anydoor: Zero-shot object-level image customization. In: *arXiv preprint arXiv:2307.09481* (2023)

[8] Chen, Yang ; Pan, Yingwei ; Yao, Ting ; Tian, Xinmei ; Mei, Tao: *Mocycle-GAN: Unpaired Video-to-Video Translation.* 2019

[9] Choi, Yunjey ; Choi, Minje ; Kim, Munyoung ; Ha, Jung-Woo ; Kim, Sunghun ; Choo, Jaegul: *StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation.* 2018

[10] Davey, Patrick ; Sprigings, David ; Keshav, Satish ; Kent, Alexandra: 1181Prevention of gastrointestinal disease. In: *Diagnosis and Treatment in Internal Medicine.* Oxford University Press, 08 2018. – URL https://doi.org/10.1093/med/9780199568741.003.0346. – ISBN 9780199568741

[11] Demir, Ugur ; Unal, Gozde: *Patch-Based Image Inpainting with Generative Adversarial Networks.* 2018

[12] Deng, Jiankang ; Guo, Jia ; Yang, Jing ; Xue, Niannan ; Kotsia, Irene ; Zafeiriou, Stefanos: ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (2022), Oktober, Nr. 10, S. 5962–5979. – URL http://dx.doi.org/10.1109/TPAMI.2021.3087709. – ISSN 1939-3539

[13] Desai, Chitra: Comparative analysis of optimizers in deep neural networks. In: *International Journal of Innovative Science and Research Technology* 5 (2020), Nr. 10, S. 959–962

[14] Ding, Yi ; Chen, Fujuan ; Zhao, Yang ; Wu, Zhixing ; Zhang, Chao ; Wu, Dongyuan: A Stacked Multi-Connection Simple Reducing Net for Brain Tumor Segmentation. In: *IEEE Access* PP (2019), 07, S. 1–1

[15] Engelhardt, Sandy ; De Simone, Raffaele ; Full, Peter M. ; Karck, Matthias ; Wolf, Ivo: *Improving Surgical Training Phantoms by Hyperrealism: Deep Unpaired Image-to-Image Translation from Real Surgeries.* S. 747–755. In: *Lecture Notes in Computer Science*, Springer International Publishing, 2018. – ISBN 9783030009281

[16] Engelhardt, Sandy ; De Simone, Raffaele ; Full, Peter M. ; Karck, Matthias ; Wolf, Ivo: Improving surgical training phantoms by hyperrealism: deep unpaired image-to-image translation from real surgeries. In: *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I* Springer (Veranst.), 2018, S. 747–755

[17] Fagereng, Jan A. ; Thambawita, Vajira ; Storås, Andrea M. ; Parasa, Sravanthi ; Lange, Thomas de ; Halvorsen, Pål ; Riegler, Michael A.: *PolypConnect: Image inpainting for generating realistic gastrointestinal tract images with polyps.* 2022

[18] Fitting, Daniel ; Krenzer, Adrian ; Troya, Joel ; Banck, Michael ; Sudarevic, Boban ; Brand, Markus ; Böck, Wolfgang ; Zoller, Wolfram G. ; Rösch, Thomas ; Puppe, Frank ; Meining, Alexander ; Hann, Alexander: A video based benchmark data set (ENDOTEST) to evaluate computer-aided polyp detection systems. In: *Scandinavian Journal of Gastroenterology* 57 (2022), Nr. 11, S. 1397–1403

[19] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning. Das umfassende Handbuch*. MIT Press, 2018. – ISBN 9783958457003

[20] Goodfellow, Ian J. ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: *Generative Adversarial Networks*. 2014

[21] Graeven, Ullrich: *Onko internetportal*. May 2021. – URL https://www.krebsgesellschaft.de/onko-internetportal/basis-informationen-krebs/krebsarten/darmkrebs/frueherkennung.html

[22] Grossberg, Stephen: Recurrent neural networks. In: *Scholarpedia* 8 (2013), Nr. 2, S. 1888

[23] Gu, Bohai ; Yu, Yongsheng ; Fan, Heng ; Zhang, Libo: *Flow-Guided Diffusion for Video Inpainting*. 2023

[24] Han, Kai ; Wang, Yunhe ; Chen, Hanting ; Chen, Xinghao ; Guo, Jianyuan ; Liu, Zhenhua ; Tang, Yehui ; Xiao, An ; Xu, Chunjing ; Xu, Yixing u. a.: A survey on vision transformer. In: *IEEE transactions on pattern analysis and machine intelligence* 45 (2022), Nr. 1, S. 87–110

[25] Ho, Jonathan ; Jain, Ajay ; Abbeel, Pieter: *Denoising Diffusion Probabilistic Models*. 2020

[26] Ho, Jonathan ; Salimans, Tim: *Classifier-Free Diffusion Guidance*. 2022

[27] Isensee, Fabian ; Jaeger, Paul F. ; Kohl, Simon A. ; Petersen, Jens ; Maier-Hein, Klaus H.: nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. In: *Nature methods* 18 (2021), Nr. 2, S. 203–211

[28] Jahanian, Ali ; Chai, Lucy ; Isola, Phillip: *On the "steerability" of generative adversarial networks*. 2020

[29] Jansen, Petra L.: *Krankheiten der Verdauungsorgane behandeln und vermeiden*. 2022. – Deutsche Gesellschaft für Verdauungs und Stoffweckselerkrankungen

[30] JOEL, Troya ; SUDAREVIC, Boban ; ADRIAN, Krenzer ; MICHAEL, Banck ; MARKUS, Brand ; BENJAMIN, Walter ; FRANK, Puppe ; WOLFRAM, Zoller ; ALEXANDER, Meining ; ALEXANDER, Hann: Direct comparison of multiple computer-aided polyp detection systems. In: *Endoscopy* 0 (2023), 08

[31] KANEKO, Takuhiro ; KAMEOKA, Hirokazu ; TANAKA, Kou ; HOJO, Nobukatsu: *CycleGAN-VC2: Improved CycleGAN-based Non-parallel Voice Conversion.* 2019

[32] KARRAS, Tero ; AILA, Timo ; LAINE, Samuli ; LEHTINEN, Jaakko: *Progressive Growing of GANs for Improved Quality, Stability, and Variation.* 2018

[33] KELLEY, Henry J.: Gradient theory of optimal flight paths. In: *Ars Journal* (1960), S. 947–954

[34] KHADER, Firas ; MUELLER-FRANZES, Gustav ; ARASTEH, Soroosh T. ; HAN, Tianyu ; HAARBURGER, Christoph ; SCHULZE-HAGEN, Maximilian ; SCHAD, Philipp ; ENGELHARDT, Sandy ; BAESSLER, Bettina ; FOERSCH, Sebastian ; STEGMAIER, Johannes ; KUHL, Christiane ; NEBELUNG, Sven ; KATHER, Jakob N. ; TRUHN, Daniel: *Medical Diffusion: Denoising Diffusion Probabilistic Models for 3D Medical Image Generation.* 2023

[35] KIM, Hyunsu ; CHOI, Yunjey ; KIM, Junho ; YOO, Sungjoo ; UH, Youngjung: *Exploiting Spatial Dimensions of Latent in GAN for Real-time Image Editing.* 2021

[36] LI, Zewen ; LIU, Fan ; YANG, Wenjie ; PENG, Shouheng ; ZHOU, Jun: A survey of convolutional neural networks: analysis, applications, and prospects. In: *IEEE transactions on neural networks and learning systems* 33 (2021), Nr. 12, S. 6999–7019

[37] LIU, Guilin ; REDA, Fitsum A. ; SHIH, Kevin J. ; WANG, Ting-Chun ; TAO, Andrew ; CATANZARO, Bryan: *Image Inpainting for Irregular Holes Using Partial Convolutions.* 2018

[38] LU, Yongyi ; TAI, Yu-Wing ; TANG, Chi-Keung: *Attribute-Guided Face Generation Using Conditional CycleGAN.* 2018

[39] LUGMAYR, Andreas ; DANELLJAN, Martin ; ROMERO, Andres ; YU, Fisher ; TIMOFTE, Radu ; GOOL, Luc V.: *RePaint: Inpainting using Denoising Diffusion Probabilistic Models.* 2022

[40] Meng, Xiangxi ; Gu, Yuning ; Pan, Yongsheng ; Wang, Nizhuan ; Xue, Peng ; Lu, Mengkang ; He, Xuming ; Zhan, Yiqiang ; Shen, Dinggang: *A Novel Unified Conditional Score-based Generative Framework for Multi-modal Medical Image Completion.* 2022

[41] Mescheder, Lars ; Geiger, Andreas ; Nowozin, Sebastian: *Which Training Methods for GANs do actually Converge?* 2018

[42] Mok, Tony C. W. ; Chung, Albert C. S.: *Affine Medical Image Registration with Coarse-to-Fine Vision Transformer.* 2022

[43] Nichol, Alex ; Dhariwal, Prafulla: *Improved Denoising Diffusion Probabilistic Models.* 2021

[44] Niu, Li ; Cong, Wenyan ; Liu, Liu ; Hong, Yan ; Zhang, Bo ; Liang, Jing ; Zhang, Liqing: *Making Images Real Again: A Comprehensive Survey on Deep Image Composition.* 2023

[45] Oquab, Maxime ; Darcet, Timothée ; Moutakanni, Théo ; Vo, Huy ; Szafraniec, Marc ; Khalidov, Vasil ; Fernandez, Pierre ; Haziza, Daniel ; Massa, Francisco ; El-Nouby, Alaaeldin ; Assran, Mahmoud ; Ballas, Nicolas ; Galuba, Wojciech ; Howes, Russell ; Huang, Po-Yao ; Li, Shang-Wen ; Misra, Ishan ; Rabbat, Michael ; Sharma, Vasu ; Synnaeve, Gabriel ; Xu, Hu ; Jegou, Hervé ; Mairal, Julien ; Labatut, Patrick ; Joulin, Armand ; Bojanowski, Piotr: *DINOv2: Learning Robust Visual Features without Supervision.* 2024

[46] Patel, Parth ; Kumari, Nupur ; Singh, Mayank ; Krishnamurthy, Balaji: Lt-gan: Self-supervised gan with latent transformation detection. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, S. 3189–3198

[47] Radford, Alec ; Kim, Jong W. ; Hallacy, Chris ; Ramesh, Aditya ; Goh, Gabriel ; Agarwal, Sandhini ; Sastry, Girish ; Askell, Amanda ; Mishkin, Pamela ; Clark, Jack ; Krueger, Gretchen ; Sutskever, Ilya: *Learning Transferable Visual Models From Natural Language Supervision.* 2021

[48] Rivoir, Dominik ; Pfeiffer, Micha ; Docea, Reuben ; Kolbinger, Fiona ; Riediger, Carina ; Weitz, Jürgen ; Speidel, Stefanie: *Long-Term Temporally Consistent Unpaired Video Translation from Simulated Surgical 3D Data.* 2021

[49] Rombach, Robin ; Blattmann, Andreas ; Lorenz, Dominik ; Esser, Patrick ; Ommer, Björn: *High-Resolution Image Synthesis with Latent Diffusion Models.* 2021

[50] Ronneberger, Olaf ; Fischer, Philipp ; Brox, Thomas: *U-Net: Convolutional Networks for Biomedical Image Segmentation.* 2015

[51] Rouzrokh, Pouria ; Khosravi, Bardia ; Faghani, Shahriar ; Moassefi, Mana ; Vahdati, Sanaz ; Erickson, Bradley J.: *Multitask Brain Tumor Inpainting with Diffusion Models: A Methodological Report.* 2023

[52] Salehinejad, Hojjat ; Sankar, Sharan ; Barfett, Joseph ; Colak, Errol ; Valaee, Shahrokh: Recent advances in recurrent neural networks. In: *arXiv preprint arXiv:1801.01078* (2017)

[53] Schmitz, Rüdiger ; Wallrodt, Moritz ; Ehlken, Hanno ; Madesta, Frederic ; Ohlhoff, Karsten ; Lux, Thomas ; Hann, Alexander ; Rösch, Thomas ; Werner, René: *Generative AI for the Live virtual Augmentation of Endoscopy Training Simulators – Breath of Life for an Inanimate Friend.* – In preparation for submission for publication to *Endoscopy, Innovations and brief communications (IBCs)* 2024

[54] Shen, Yujun ; Gu, Jinjin ; Tang, Xiaoou ; Zhou, Bolei: *Interpreting the Latent Space of GANs for Semantic Face Editing.* 2020

[55] Song, Yizhi ; Zhang, Zhifei ; Lin, Zhe ; Cohen, Scott ; Price, Brian ; Zhang, Jianming ; Kim, Soo Y. ; Aliaga, Daniel: *ObjectStitch: Generative Object Compositing.* 2022

[56] Spinney, Richard E. ; Ford, Ian J.: *Fluctuation relations: a pedagogical overview.* 2012

[57] Tang, Guangyi ; Ni, Jianjun ; Chen, Yan ; Cao, Weidong ; Yang, Simon X.: An Improved CycleGAN Based Model For Low-light Image Enhancement. In: *IEEE Sensors Journal* (2023), S. 1–1

[58] Teed, Zachary ; Deng, Jia: *RAFT: Recurrent All-Pairs Field Transforms for Optical Flow.* 2020

[59] Tschannen, Michael ; Bachem, Olivier ; Lucic, Mario: Recent advances in autoencoder-based representation learning. In: *arXiv preprint arXiv:1812.05069* (2018)

[60] VASWANI, Ashish ; SHAZEER, Noam ; PARMAR, Niki ; USZKOREIT, Jakob ; JONES, Llion ; GOMEZ, Aidan N. ; KAISER, Lukasz ; POLOSUKHIN, Illia: *Attention Is All You Need.* 2023

[61] WALLRODT, Moritz ; SCHULZ-ALSEN, Maximilian ; EHLKEN, Hanno ; RÖSCH, Thomas ; SCHMITZ, Rüdiger ; WERNER, René: Extending Tempcyclegan for Virtual Augmentation of Gastrointestinal Endoscopy Training Simulators. In: DESERNO, Thomas M. (Hrsg.) ; HANDELS, Heinz (Hrsg.) ; MAIER, Andreas (Hrsg.) ; MAIER-HEIN, Klaus (Hrsg.) ; PALM, Christoph (Hrsg.) ; TOLXDORFF, Thomas (Hrsg.): *Bildverarbeitung für die Medizin 2023.* Wiesbaden : Springer Fachmedien Wiesbaden, 2023, S. 3–8. – ISBN 978-3-658-41657-7

[62] WANG, Tongzhou ; LIN, Yihan: *CycleGAN with Better Cycles.* 2018

[63] WERNER, René ; SCHMIDT-RICHBERG, Alexander ; HANDELS, Heinz ; EHRHARDT, Jan: Estimation of lung motion fields in 4D CT data by variational non-linear intensity-based registration: A comparison and evaluation study. In: *Physics in Medicine and Biology* 59 (2014), 07, S. 4247–4260

[64] YANG, Binxin ; GU, Shuyang ; ZHANG, Bo ; ZHANG, Ting ; CHEN, Xuejin ; SUN, Xiaoyan ; CHEN, Dong ; WEN, Fang: Paint by Example: Exemplar-based Image Editing with Diffusion Models. In: *arXiv preprint arXiv:2211.13227* (2022)

[65] ZHANG, Lvmin ; RAO, Anyi ; AGRAWALA, Maneesh: *Adding Conditional Control to Text-to-Image Diffusion Models.* 2023

[66] ZHANG, Richard ; ISOLA, Phillip ; EFROS, Alexei A. ; SHECHTMAN, Eli ; WANG, Oliver: *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric.* 2018

[67] ZHANG, Zhixing ; WU, Bichen ; WANG, Xiaoyan ; LUO, Yaqiao ; ZHANG, Luxin ; ZHAO, Yinan ; VAJDA, Peter ; METAXAS, Dimitris ; YU, Licheng: *AVID: Any-Length Video Inpainting with Diffusion Model.* 2023

[68] ZHAO, Feng ; HUANG, Qingming ; GAO, Wen: Image Matching by Normalized Cross-Correlation. In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings* Bd. 2, 2006, S. II–II

[69] ZHU, Jun-Yan ; PARK, Taesung ; ISOLA, Phillip ; EFROS, Alexei A.: *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.* 2020

[70] Zhu, Jun-Yan ; Zhang, Richard ; Pathak, Deepak ; Darrell, Trevor ; Efros, Alexei A. ; Wang, Oliver ; Shechtman, Eli: Toward multimodal image-to-image translation. In: *Advances in Neural Information Processing Systems*, 2017

Declaration of independent work on a thesis

I hereby certify that I have independently written this without any help from others and that I have only used the and that I have used only the indicated aids. Verbatim or in the sense of other works taken passages are marked under the sources indicated.

Location	Date	Signature