University of Turin

Master in Computer Science

Artificial Intelligence and Information Systems

Generative Adversarial Networks for Histopathological Image Synthesis and Augmentation

Supervisor:
Prof. Marco Grangetto
Co-supervisor:
Carlo Alberto Barbano
Examiner:
Prof. Francesca Cordero

Candidate:
Desislav Nikolaev Ivanov

ACADEMIC YEAR
2021 / 2022

Abstract

Generative Adversarial Neural Networks (GANs) have recently achieved state-of-art performance for image synthesis and have been used to generate high-quality images of faces, animals, and 3D scenes. In this work, we explore the use of GANs in the medical computer vision field, where datasets are usually more scarce and require expert annotations. We demonstrate that GANs can be used to generate high-detailed images of medical objects, such as high-resolution microscopical organ tissue scans. In particular, we focus on the UniToPatho dataset, which contains Hematoxylin and Eosin stained (H&E) colorectal histopathological scans, and evaluate the performance of synthetic images used as an augmentation approach for classification tasks. To further improve the classification accuracy, we propose a new image-to-image StyleGAN-based model that is able to transform segmentation masks of nuclei into realistic tissues. Our model drastically improves the generation quality over previous approaches for the UniToPatho dataset and, most importantly, allows to exploit prior medical knowledge about the tissue morphology, by manually guiding the synthesis process for a more precise augmentation.

Contents

1 Introduction **1**
1.1 Colorectal cancer . . . 2
1.2 Focus of our research . . . 2
1.3 Contributions . . . 3

2 Introduction to Neural Networks **5**
2.1 Automatic Differentiation . . . 5
2.1.1 Stochastic Gradient Descent . . . 7
2.2 Architecting Neural Networks . . . 8
2.2.1 Artificial Neuron . . . 8
2.2.2 Layers of neurons and Multi-Layer Perceptron . . . 8
2.3 Training Neural Networks . . . 9
2.4 Convolutional Neural Networks . . . 11
2.4.1 Residual connections . . . 12
2.5 Autoencoders . . . 14
2.6 Variational Autoencoders . . . 14
2.7 U-Nets for Image Segmentation . . . 15
2.8 Generative Adversarial Neural Networks . . . 15
2.8.1 Deep Convolutional GANs . . . 17
2.8.2 Wasserstein GAN . . . 18
2.8.3 Conditional GANs . . . 18
2.8.4 Adversarial image-to-image translation . . . 18
2.8.5 StyleGAN . . . 19
2.9 Regularization for GANs . . . 20
2.9.1 Gradient penalty regularization . . . 20
2.9.2 Perceptual path-length regularization . . . 20
2.10 Quality metrics for GANs . . . 22
2.10.1 Fréchet Inception Distance . . . 22

3 Datasets **23**
3.1 UniToPatho dataset 23
3.2 PanNuke Dataset 24

4 Related Work **26**
4.1 GANs for Medical Computer Vision 26
4.2 Related work on UniToPatho 27

5 Generation **30**
5.1 Method 30
5.2 Setup 31
5.3 Interpolation 32
5.4 Experts' evaluation 34
5.5 FID Results and discussion 36

6 Classification with GAN-augmentation **37**
6.1 Method 37
6.2 Setup 38
6.3 Results and discussion 38

7 Image-to-image Translation **41**
7.1 Method 41
7.2 Setup 44
7.3 Results and discussion 44

A More Generations **48**

Chapter 1

Introduction

The advances of deep learning in classification and computer vision have pushed the frontier of automatic medical diagnosis and computer-aided diagnosis. Recently, the application of deep learning to medical diagnosis has become a topic of great interest, including the fields of radiology, cardiology, and pathology. In particular, deep learning in pathology has shown promising results, with the development of algorithms that can be trained on large amounts of data to automatically detect and segment tumor tissue in digitized pathology images. These computer-aided tumor detection and segmentation algorithms can potentially support pathologists in their diagnostic task. The adoption of such models can have an enormous impact on the clinical decisions made by pathologists, so it is crucial to ensure that the models are accurate and robust enough. One way to achieve high precision and accuracy without over-fitting the models is to use large amounts of data to catch the variety of cases that can be seen in the clinical world. However, collecting large databases of real world data for a niche diagnosis field can be challenging. First, the data collection process is costly, because it requires the use of special equipment, trained medical personnel, and increasingly, regulatory approval. Second, the data should be collected from multiple sources and institutions to avoid bias[41]. Finally, data sharing is also challenging because of privacy and regulatory concerns.

In this context, generative models have shown great promise because they can generate realistic synthetic images that can be used to train models and to perform data augmentation. Using synthetic images can have several benefits, including the generation of large datasets that can be used to train robust models, and the ability to generate images that can augment the training dataset to improve the performance of existing models. In particular, Generative Adversarial Networks (GANs)[15] have shown noticeable results in generating realistic images in a wide range of applications, including synthesis of realistic photographs of faces, natural scenes, and medical images [27]. In addition, GANs can be used to generate new images from control points [6], to interpolate between existing images [64], and to generate sequences of images [63]. Furthermore, the idea of using GANs for medical imaging has been researched in a lot of different studies and tasks [32, 58, 60], ranging from segmentation[57, 35], to synthesis[7, 22] and semi-supervised classification[23].

1.1 Colorectal cancer

Colorectal cancer is the third most common type of cancer in both men and women[52], although it can often be prevented through regular screenings and early detection. It usually starts as a small polyp, a clump of cells forming on the colon or rectum lining. Most polyps are benign; however, some can become cancer over time. There are various parameters for determining the danger of this transformation that pathologists evaluate: the type of polyp, their size and the degree of dysplasia. The type of polyp is determined by its appearance and the three most common types are tubular adenomas, villous adenomas and tubulovillous adenomas. The size of the polyp is also taken into account when determining its danger level. Smaller polyps have a lower chance of becoming cancer than larger ones; however, even small polyps should be removed if they show any signs of dysplasia (abnormal cell growth). Dysplasia can be graded on a scale from low to high: low-grade dysplasia means that the abnormal cells are only slightly different from normal cells and are unlikely to become cancer; high-grade dysplasia means that the abnormal cells look very different from normal cells and have a greater chance of turning into cancer. Polyps with high-grade dysplasia should be removed immediately as they pose the greatest threat.

1.2 Focus of our research

We focus our research on UniToPatho [3], a dataset containing Hematoxylin and Eosin stained (H&E) colorectal histopathological scans. One of the main purposes of the dataset is to enable the development of automatic prediction of dysplasia grade of colorectal polyps to support pathologists for more accurate diagnosis.

The dataset has been studied in three main areas. Firstly, Barbano et al.[3] developed a model for automatic tissue classification. Secondly, Di Luccio et al.[9] introduced a segmentation model that is capable of extracting and classifying nuclei contained in the images. Finally, Rubinetti et al.[49] proposed a generative model that is able to reconstruct segmentation masks of nuclei into realistic tissues.

All of the studies encountered difficulties mainly related to the small size of the dataset. In particular, the core obstacle of the dataset is driven by the lack of high dysplasia grade samples, which produces a strong imbalance in the data distribution.

For these reasons, we focus on building new generative models capable of synthesising new samples that can be used to augment the UniToPatho dataset. The purpose of such augmentations is to allow for an improved precision of automatic diagnosis.

1.3 Contributions

The main contributions of this thesis are the analysis of the application of GANs to the UniToPatho dataset and the improvement over prior generative techniques.

Automatic image synthesis

First, we propose different models of GANs applied to the UniToPatho dataset. We show that GANs are capable of learning the salient charachteristics of the dataset on different resolutions and can be used to generate high-quality tissues undistinguishable from real ones. We analyze the quality of the GANs by exploring the characteristics of the latent space through interpolation and by exploiting the Discriminator as an unsupervisedly-trained feature extractor that can be extended for classification. Subsequently, we synthesize various datasets and use them as augmentation for classification models. We show that this augmentation approach allows us to achieve slight improvements in accuracy, but the GANs still suffer from the imbalance of the dataset.

Improved image-to-image translation

With the goal of further increasing the classification precision, we extend our GAN model with additional inputs that can guide the Generator in the synthesis process. The generation of images in GANs is usually guided by normally distributed random values, as described later in section 2.8. However, the synthesis can be further conditioned by prior factors, which in our case are segmentation masks of nuclei. A similar approach has already been proposed by Rubinetti et al.[49]. We show that with an additional Encoder network that injects the segmentation masks of nuclei into the synthesis process of a sophisticated StyleGAN [28] based model, we are able to dramatically improve the quality of the generations over prior work. This allows us to further exploit prior expert medical knowledge to generate more meaningful synthetic samples by manually guiding the synthesis towards the underrepresented high-grade dysplasia class, and eventually increase the classification accuracy.

Chapter 2

Introduction to Neural Networks

In this chapter we provide a brief introduction to all the concepts of neural networks and deep learning used later in the thesis.

2.1 Automatic Differentiation

Neural networks are machine learning models capable of approximating arbitrary mappings from one finite dimensional space to another [21]. A neural network can be seen as a function $\mathbf{f}$ parameterized by a set of parameters $\boldsymbol{\theta}$ and applied to an input $\mathbf{x}$:

$$\hat{\mathbf{y}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$$

Internally, the neural network consists of a sequence of differentiable operations manipulating both $\mathbf{x}$ and $\boldsymbol{\theta}$, producing a final result $\hat{\mathbf{y}}$. The core idea behind neural networks is to approximate a real mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ by using a finite set of pairs of inputs $\boldsymbol{x} \in \mathcal{X}$ and outputs $\boldsymbol{y} \in \mathcal{Y}$ and properly adjust the parameters $\boldsymbol{\theta}$ such that the function $\mathbf{f}$ approximates f. One of the most common ways to learn these parameters is to minimize a differentiable loss function L that computes an error measure between $\hat{\mathbf{y}}$ and $\mathbf{y}$.

Since every operation of the loss L is differentiable by choice, one way to minimize it is to apply the chain rule of calculus and minimize its partial derivative with respect to $\boldsymbol{\theta}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}} L(\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}), \mathbf{y})$$

The number of parameters of neural networks and functions applied to them has grown rapidly, with billions of parameters being the norm nowadays, which makes the computation of the closed-form of the derivatives computationally too expensive.

Instead of computing the full closed-form derivatives, another way to solve the problem is to view the function as a Directed Acyclic Graph (DAG) defined over the operations used internally. During the computation of the function, also referred to as the forward pass, the intermediate results are memoized, thus allowing gradients to be efficiently computed in a backward pass with the same time complexity of the forward pass.

For example, suppose we have the following function:

$$f(x, w_1, w_2, w_3) = w_1x^2 + w_2x + w_3 \tag{2.1}$$

We can decompose all the operations in the following equivalent way:

$$\begin{aligned} y &= d + w_3 \\ d &= b + c \\ b &= w_1g \\ g &= x^2 \\ c &= w_2x \end{aligned}$$

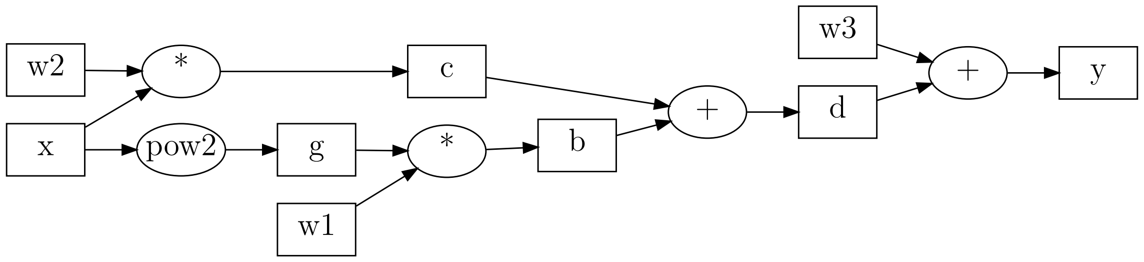


Figure 2.1: The DAG describing the computation of the function defined in 2.1. Square nodes refer to variables and intermediate results. Ellipse nodes represent operations with arguments coming from the result of the source nodes of incoming arrows.

Now, to compute the derivative of y with respect to all the intermediate variables we can simply apply the chain rule going backward from the final output y in the computation graph shown in Figure 2.1:

$$\begin{aligned}
\frac{\partial y}{\partial d} &= 1 \\
\frac{\partial y}{\partial w_3} &= 1 \\
\frac{\partial y}{\partial b} &= \frac{\partial y}{\partial d}\frac{\partial d}{\partial b} = 1 \\
\frac{\partial y}{\partial c} &= \frac{\partial y}{\partial d}\frac{\partial d}{\partial c} = 1 \\
\frac{\partial y}{\partial g} &= \frac{\partial y}{\partial b}\frac{\partial b}{\partial g} = w_1 \\
\frac{\partial y}{\partial w_1} &= \frac{\partial y}{\partial b}\frac{\partial b}{\partial w_1} = g \\
\frac{\partial y}{\partial w_2} &= \frac{\partial y}{\partial c}\frac{\partial c}{\partial w_2} = x \\
\frac{\partial y}{\partial x} &= \frac{\partial y}{\partial g}\frac{\partial g}{\partial x} + \frac{\partial y}{\partial c}\frac{\partial c}{\partial x} = 2w_1 x + w_2
\end{aligned} \tag{2.2}$$

As shown in Equation 2.2, once we reached a node, we were able to locally compute the partial derivatives by simply considering the precomputed derivative of directly connected nodes, without having to apply the full chain rule down to the root. This was possible because we intentionally followed a strict order in the derivation, a topological order starting from the output y, which guarantees that once a node is reached, all its connected nodes' gradients are computed. This process of efficiently propagating gradients through the network is also known as backpropagation, which is a specific use of the more general method of automatic differentiation and is automatically applicable only to acyclic graphs.

Automatic differentiation is at the core of today's most advanced neural network engines such as PyTorch[42] and JAX[4].

2.1.1 Stochastic Gradient Descent

In order to minimize the loss function, the weights must be adjusted properly. This adjustment can be guided by the gradient, which determines in which direction to update the weights such that the error decreases, also known as Gradient Descent. The most accurate gradient computation can be obtained by computing the loss over the whole dataset. However, this would result in the whole computation graph for all the examples being kept in memory, which is not applicable to large datasets.

For this reason, Stochastic Gradient Descent is usually preferred instead. SGD approximates the full gradient by applying the backpropagation step with randomly selected examples. An extension of SGD is the mini-batch SGD, which accumulates the gradients of batches of examples and then applies backpropagation.

2.2 Architecting Neural Networks

Neural networks nowadays scale to billions of parameters, so defining them as plain functions at a low level, as shown in the previous section, is not applicable. Instead, a more sophisticated approach is used, which allows to define high-level components such as Neurons and Layers that can be combined together in any differentiable way.

2.2.1 Artificial Neuron

The most basic component of a neural network is the artificial neuron, also known as the Perceptron[48]. It is a single unit containing a set of weights $\mathbf{w}$. In the forward pass, the perceptron takes a vector of inputs $\mathbf{x}$, weights them through its weights $\mathbf{w}$, and (optionally) applies a differentiable activation function *act*:

$$f(\mathbf{x}) = act(\sum_i x_i w_i + b)$$

Additionally, the perceptron can have an optional bias parameter that is added to the sum before the activation.

The weights of the perceptron are updated during training with gradient descent. Given a learning rate η which scales the gradients and a loss function L, following the idea of automatic differentiation described in section 2.1, each weight w_i is updated according to the partial derivative:

$$w_i' = w_i - \eta \frac{\partial L}{\partial w_i}$$

The same rule also applies for the bias.

2.2.2 Layers of neurons and Multi-Layer Perceptron

A layer of neurons contains multiple neurons stacked together. Each input is propagated through each neuron and the output of the layer is the set of outputs of each neuron. A layer in-between the inputs and the outputs of the network is referred to as a hidden layer.

Recall the idea that an artificial neuron is a function f, a layer of n neurons is thus a sequence of functions $f_1, f_2, \ldots f_n$. The forward pass of the layer can be described as:

$$g(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots f_n(\mathbf{x})\}$$

Another way to view a layer of n neurons is to consider all the weights of the neruons together in a single matrix $\mathbf{W}$ of size $m \times n$, where m is the size of the inputs, with an additional vector of biases $\mathbf{b}$ containing the bias of each neuron. Under this perspective, the forward function can be defined as:

$$g(\mathbf{x}) = act(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Similarly to neurons, layers can also be stacked sequentially, producing a composition of functions. A sequence of fully-connected layers defines the well-known architecture of the Multi-Layer Perceptron.

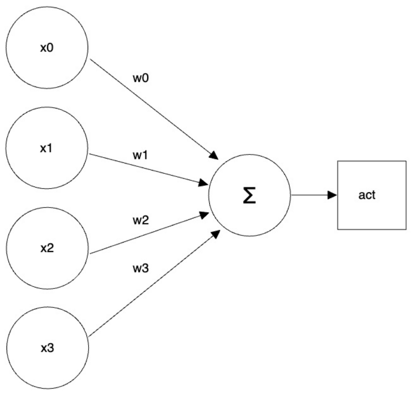


Figure 2.2: Example of perceptron.

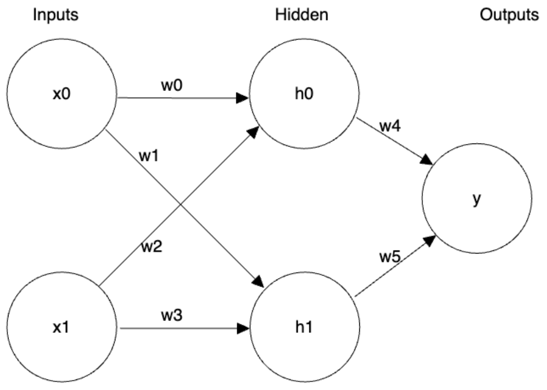


Figure 2.3: Neural network with a hidden layer.

2.3 Training Neural Networks

Suppose we are given the dataset $D = \{([0,0],0),([0,1],1),([1,0],1),([1,1],0)\}$, where in each pair the first element is the input and the second element is the output, and that we want to learn a mapping between the data. We can use a neural network to learn to predict the outputs given only the inputs. A suitable network for this task is the one described in Figure 2.3. It has two hidden neurons and one output neuron. As an activation function for each neuron we can use the sigmoid function, which has the same codomain of our data:

$$\begin{aligned} g &: \mathbb{R} \to \mathbb{R} \\ x &\mapsto \frac{1}{1+e^{-x}} \end{aligned}$$

The full forward pass of our network is then:

$$f(x_0, x_1) = g(w_4 g(w_0 x_0 + w_2 x_1 + b_0) + w_5 g(w_1 x_0 + w_3 x_1 + b_1) + b_2)$$

Now, our goal is to find a set of ws and bs such that for all $([x_0, x_1], y)$ in the dataset $f(x_0, x_1) = y$. In order to achieve this goal, we can use gradient descent to minimize a loss function that determines an error measure between the outputs of f and the expected ys in the dataset. A useful loss function for this task is the binary cross-entropy:

$$H = -\frac{1}{|D|} \sum_{x_0, x_1, y \in D} (y \cdot \log f(x_0, x_1) + (1-y) \cdot \log(1 - f(x_0, x_1)))$$

Now we can randomly initialize the weights and biases and repeatedly update them in the direction of the gradient. For example, suppose that we initialize the weights and biases to:

w_0	w_2	b_0	w_1	w_3	b_1	w_4	w_5	b_2
-0.27	-1.38	-0.51	0.35	-0.72	0.18	-1.03	0.19	2.62

If we compute the forward pass for the inputs we obtain $f(0,0) = 0.91$, $f(0,1) = 0.93$, $f(1,0) = 0.92$, and $f(1,1) = 0.93$. The loss function applied to the full dataset results in 1.32. Now, if we apply the backpropagation pass described in section 2.1, we compute the following gradients:

gw_0	gw_2	gb_0	gw_1	gw_3	gb_1	gw_4	gw_5	gb_2
-0.02	-0.02	-0.07	0.01	0.01	0.02	0.10	0.21	0.42

Now we apply the update rule and simply subtract the gradients to the relative weights, obtaining the following new weights:

w_0	w_2	b_0	w_1	w_3	b_1	w_4	w_5	b_2
-0.25	-1.36	-0.44	0.34	-0.73	0.16	-1.13	-0.02	2.20

As a result, if we compute the loss function with these new parameters, it decreases to 1.09. If we repeat the same process for 1000 steps, the loss decreases to 0.02 with the following weights:

w_0	w_2	b_0	w_1	w_3	b_1	w_4	w_5	b_2
-6.84	-6.55	2.59	4.56	4.53	-7.03	-9.83	-9.93	4.85

With the above operations we have effectively trained our neural network to approximate a suitable mapping for our dataset. The predictions of our network with these weights are $f(0,0) = 0.01 \approx 0$, $f(0,1) = 0.98 \approx 1$, $f(1,0) = 0.98 \approx 1$, $f(1,1) = 0.02 \approx 0$, which are the expected values that we wanted to learn to predict given only the inputs. The initial landscape of the function before the training is visible in Figure 2.4, whereas the landscape at the end of the training is visible in Figure 2.5.

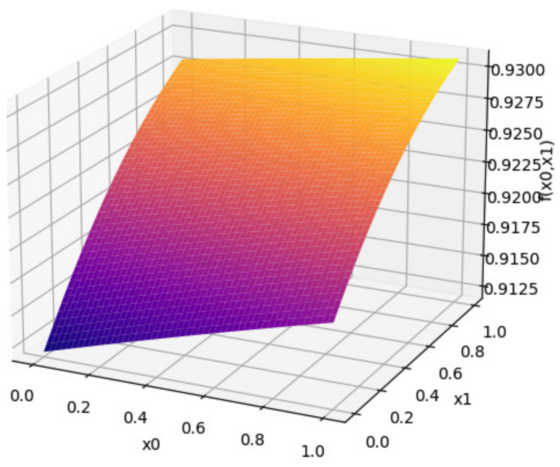


Figure 2.4: The initial landscape of the function f introduced in section 2.3

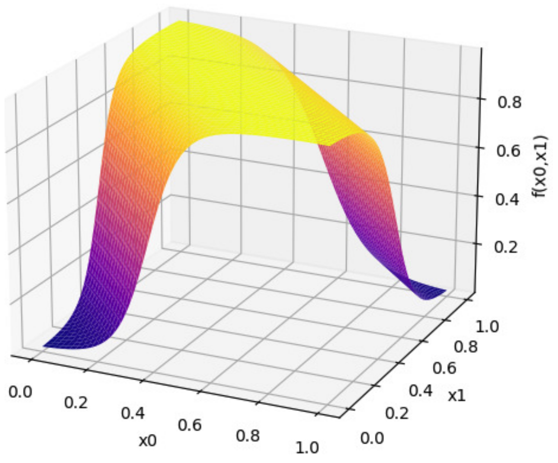


Figure 2.5: The learned landscape of the function f introduced in section 2.3

2.4 Convolutional Neural Networks

When dealing with image datasets, fully connected layers can produce very heavy networks in terms of number of parameters. For instance, a layer with just 100 outputs applied to an image of $512 \times 512 \times 3$ (RGB pixels) size requires more than $7M$ parameters. Even though nowadays state-of-the-art networks exploit such large layers with sophisticated layers [56, 11, 54, 5], most deep neural networks usually employ particular layers called Convolutional Layers, which, as the name suggests, apply a convolution over the inputs. The parameters of convolutional layers are the weights of the kernels with a fixed small size. These layers are also very efficient to train, since the small size forces the learning of generally applicable filters, such as shape extraction. For instance, given an input matrix $\mathbf{X}$ of size $n \times n$ and a kernel $\mathbf{K}$ of size $m \times m$, the result of the convolution (assuming a stride of 1) is a new matrix

$\mathbf{Y}$ of size $n - m + 1 \times n - m + 1$ defined as follows:

$$Y_{ij} = \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} \mathbf{K}_{k,l} * \mathbf{X}_{i+k,j+l}$$

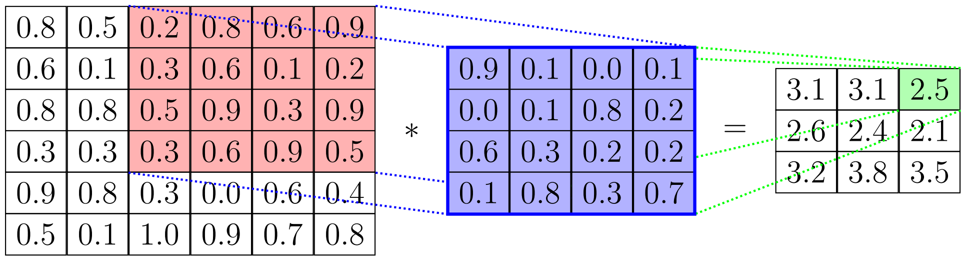


Figure 2.6: The execution of a convolution. On the left the input matrix. In the middle the kernel. On the right the output.

2.4.1 Residual connections

Stacking deep neural networks has provided a lot of advantage when dealing with complex machine learning tasks [53, 36]. However, up until 2015, extremely deep networks were actually starting to hit a bottleneck, where the increase of depth complicated the propagation of gradient, resulting in vanishing gradients. Figure 2.7 shows how the depth increase was not leading to overfitting, which is expected to happen theoretically, instead it was leading to a degradation into a local minima. By the end of 2015, a breakthrough solution was proposed by He et al.[19], which introduced the use of residual connections. The idea is to use particular convolutional blocks that facilitate the propagation of inputs down to the deepest outputs. The output of each block of convolutions becomes, in the simplest case, the plain input added to the output of the convolutions, as shown in Figure 2.8. This simple yet powerful solution allowed to scale the advantages of deep architectures by enabling the gradients to flow deeper in the backpropagation pass (Figure 2.9). The models proposed by the authors are called ResNets and are categorized by their depth, ranging from 18 layers (ResNet-18) to 152 layers (ResNet-152). The ResNet blocks have become the building blocks of today's state-of-the-art models.

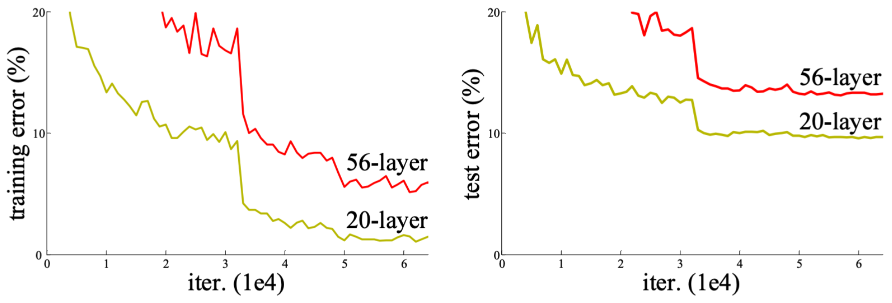


Figure 2.7: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks [19].

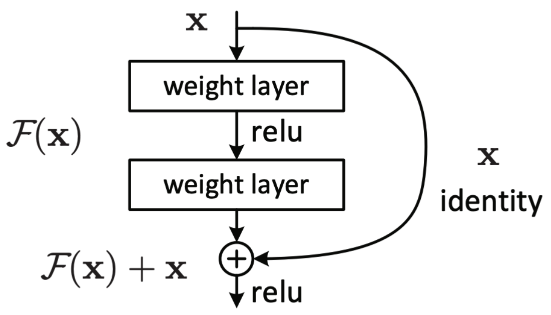

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.8: Residual connections [19]

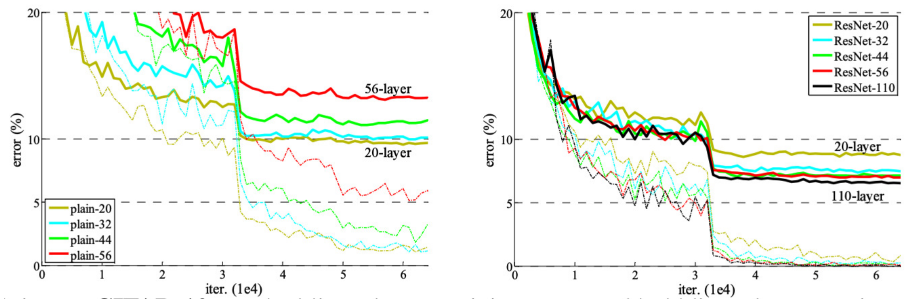


Figure 2.9: Training on CIFAR-10. Dashed lines denote training error, and bold lines denote testing error. Left: plain networks. Right: ResNets[19].

2.5 Autoencoders

Autoencoders are a class of neural networks composed of two main components: an Encoder and a Decoder (Figure 2.10). The idea of the autoencoder is to learn an encoding function f that encodes the input into a latent space $\mathbf{h} = f(\mathbf{x})$ and then decode it with an additional decoding function g such that $\hat{\mathbf{x}} = g(\mathbf{h})$. In other words, the autoencoder network learns the identity function $I = f \circ g$. Training the network consists in minimizing a reconstruction loss function:

$$L(g(f(\mathbf{x})), \mathbf{x})$$

One of the purpose of these models is to learn a compression function capable of mapping input data to an intermediate space containing the most salient characteristics of the inputs, so that the decoder can losslessly reconstruct them. Furthermore, once the encoding function is learned, it can be reused for different tasks, such as classification and regression.

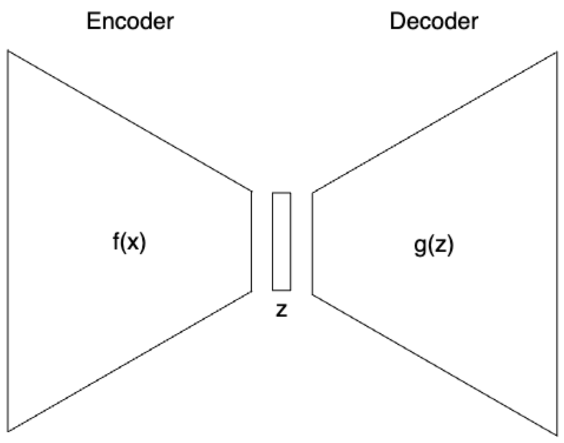


Figure 2.10: The autoencoder network.

2.6 Variational Autoencoders

Variational Autoencoders (VAE) were introduced in 2013 by Kingma and Welling [34]. The idea is to extend autoencoders within a probabilistic setting, by forcing them to learn a mapping to a probability distribution. Instead of using the encoder as a direct mapping from the input to a latent space, it is used as a function that computes the parameters of a probability distribution. For instance, if a Gaussian distribution is used, the encoder maps the input to the mean and variance of the distribution. A random latent vector is then sampled from the parametrized distribution. The encoder can be seen as an estimator for a posterior $q(z|x)$, and the reparametrization can be interpreted as a random sampling from $q(z|x)$. Finally, the decoder represents an estimator for the true posterior distribution $p(x|z)$. Under this perspective, the loss function for a VAE consists of two terms[45]

$$-D_{\mathrm{KL}}(q(z|x)\|p(z)) + \log p(x|z).$$

The first term is the Kullback-Leibler divergence, which enforces the approximate posterior distribution $q(z|x)$ and the model prior $p(z)$ to converge. The second term is the reconstruction penalty in the form of log-likelihood. The full measure is also known as the Evidence Lower BOund (ELBO), or variational lower bound.

The structure of VAEs allows to learn a continuous latent space capable of generating new samples by randomly drawing latent vectors from a given set of parameters.

2.7 U-Nets for Image Segmentation

U-Net [47] is a convolutional neural network architecture developed specifically for biomedical image segmentation. In addition to the architecture, the authors also provide insights on how to deal with low amount of data through data augmentation. The network is similar to an autoencoder, except for the fact that there are skip connections from each downsampling output to its corresponding resolution upsampling layer (Figure 2.11). Since these networks are only composed by convolutions, they can be applied to any input size. The network is trained on images and their segmentation maps, with the inputs being the raw images and outputs representing the segmentations. The proposed loss function for training these models is the cross-entropy applied on the soft-max on channel-wise scale, where each channel corresponds to a segmentation class.

2.8 Generative Adversarial Neural Networks

Generative Adversarial Networks [15] (GANs) are a class of neural networks used for generative modeling. They are formed by two neural networks competing against each other in an adversarial way. One is the generator network and the other is the discriminator: the generator network g is responsible for generating samples $\boldsymbol{x} = g(\boldsymbol{z}; \boldsymbol{\theta}^g)$, while the discriminator network d learns to distinguish between real

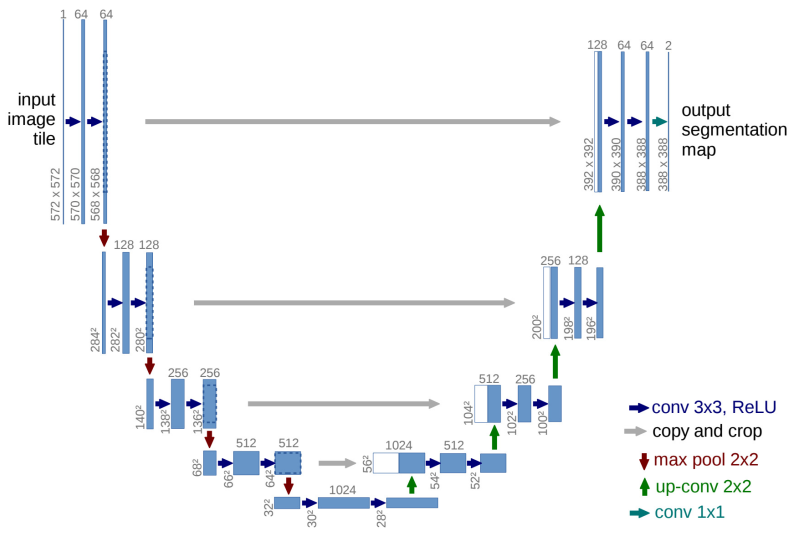


Figure 2.11: U-Net[47]

and generated samples. More precisely, the discriminator computes a probability $d(\boldsymbol{x}; \boldsymbol{\theta}^d)$ indicating whether $\boldsymbol{x}$ is a real example drawn for the real dataset or is a generated example produced by the generator. In practice, the generator network receives a random vector as input, usually drawn from a normal distribution, where each value is responsible for determining some characteristic of the generation.

Both networks are trained independently with two losses. The discriminator network learns to maximize its output when shown real samples, and minimize it when shown fake ones. On the other hand, the generator network never gets to see real samples. In fact, the generator network learns to maximize the output of the discriminator, trying to "fool" it into predicting that the generated images are actually real. In other words, the two networks are competing in a zero-sum game, where a function $J(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$ determines the reward of the discriminator, which results in the opposite reward for the generator $-J(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$. Under this perspective, one network is trained to minimize the reward of the other and vice versa:

$$\min_g \max_d J(d, g) = \mathbb{E}_{\boldsymbol{x} \sim p_{\mathtt{data}}(\boldsymbol{x})}[\log d(\boldsymbol{z})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - d(g(\boldsymbol{z})))]$$

However, it can be shown that using this loss the gradients for the generator network do not flow sufficiently, since $\log(1 - d(g(\boldsymbol{z})))$ saturates [15]. For this reason, instead of training the generator to minimize $\log(1 - d(g(\boldsymbol{z})))$, Goodfellow et al. [15] suggest to heuristically train the generator to maximize $\log(d(g(\boldsymbol{z})))$, also known as non-saturating loss.

2.8.1 Deep Convolutional GANs

As shown earlier, convolutional layers allow to dramatically increase the scalability and efficiency of neural networks when dealing with images. This also holds for GANs used for image generation. The first use of convolutions for image generation was proposed with the Deep Convolutional Generative Adversarial Networks (DC-GAN) [43]. The idea of DCGANs is to use convolutions in a *reverse* way during the generation, with this operation being known as Deconvolution (or transposed convolution) [61]. The generator network proposed by Radford et al. [43] uses a randomly sampled latent vector z of size 100 as input, which gets projected into a $1024 \times 4 \times 4$ space through a fully connected layer. After this projection, transposed convolutions are used to halve the number of channels and double the resolution size of previous layer's output. After each convolution the authors apply a ReLU activation function with Batch Normalization[25]. Finally, the output of the model is passed through a *tanh* activation function. On the other hand, the discriminator network is based on standard convolutional layers, LeakyReLU and Batch Normalization. The introduction of Deconvolutions in the GANs allowed to achieve impressive results in image synthesis and become the standard building block for state-of-the-art GANs.

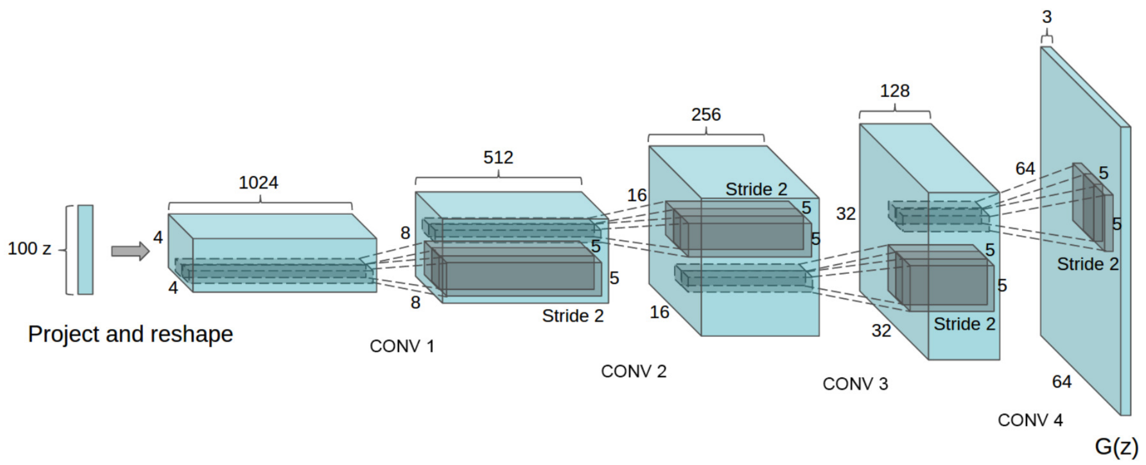


Figure 2.12: The generator network of a Deep Convolutional Generative Adversarial Network [43].

2.8.2 Wasserstein GAN

The original optimization technique proposed for GANs resulted in convergence issues and encouraged research towards alternative approaches. One remarkable solution to the convergence problem was proposed by Arjovsky et al.[2] with the introduction of Wasserstein GANs. The authors propose a loss function based on the Wasserstein-1 (or Earth Mover, EM) distance. They prove that there are many probability distributions (such as a uniform distribution on unit interval) that are mathematically impossible to approach each other under the Kullback-Leibler or Jensen-Shannon divergences, but successfully converge under the EM distance. However, the authors also show that using this loss requires hard constraints on the weights of the networks to be enforced. In particular, they show that every parameter must lie in a compact space (Lipschitz constraint). They propose two variants of enforcing this constraint: one variant is a simple clipping of the weights to some constant range, and the other is a projection of the weights to a sphere. However, these constraints inevitably lead to vanishing gradients, so the research on the most suitable enforcement of the Lipschitz constraint in neural networks is still open. The authors still succeed in employing the loss in a GAN and show remarkable advantages in the gradient descent, as shown in Figure 2.13. Nevertheless, the research on efficient GAN losses and regularization is still under discussion [16, 39, 40].

2.8.3 Conditional GANs

GANs can additionally be conditioned during the generation: instead of sampling images just from random vectors, conditional GANs also make use of high-level features, such as hair style, pose and emotion in the context of face generation. These labels are injected into the input of both the generator and the discriminator networks, forcing the discriminator to learn to distinguish different labels and the generator to generate samples with specific labels.

2.8.4 Adversarial image-to-image translation

Conditional GANs successfully generate random images with manually selected features such as high-level object classes. An even deeper conditioning of GANs was introduced with the Pix2Pix[26] paper, where Isola et al. achieve remarkable results in image-to-image translation through GANs. The idea is to use a U-Net as a generator network and a traditional convolutional discriminator and training them in an adversarial way. The generator uses images as both inputs and outputs and learns

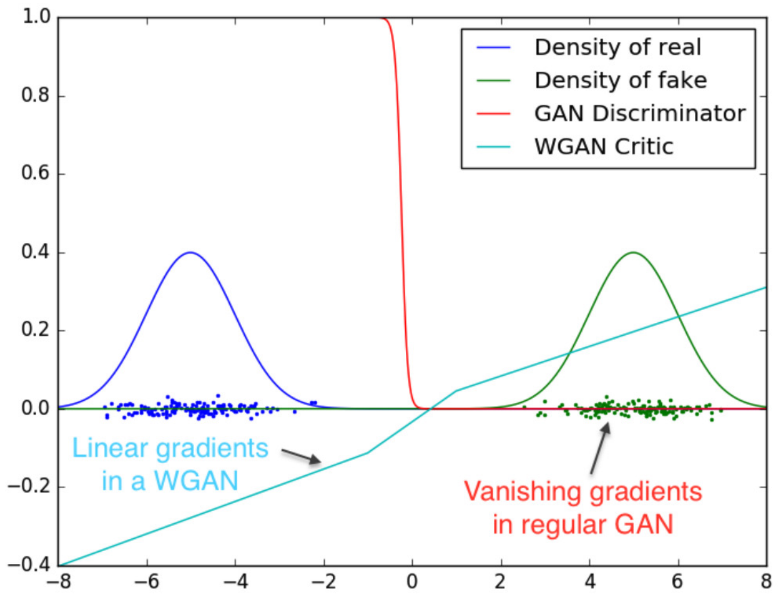


Figure 2.13: Two Gaussian distributions describing the real and fake samples used to compare the gradients of minimax GANs and WGANs. The GAN discriminator is able to very accurately distinguish real and fake samples and provides no reliable gradient information. On the other hand, the WGAN critic provides clean and linear gradient in all parts of the space. The horizontal axis describes the single feature of the data, whereas the vertical axis describes the density function of the two Gaussians and the predictions of both the GAN Discriminator and WGAN Critic.

the mapping from the two spaces. The adversarial loss allows the networks to produce high-resolution images indistinguishable from real ones. The authors show that the simple use of adversarial loss allows the network to be applicable to practically any image-to-image translation task.

2.8.5 StyleGAN

The most advanced GAN architecture for image generation is StyleGAN[28]. It was introduced by Karras et al. in 2018 and still remains state-of-the-art in realistic generative image modeling. It is based on the idea of style transfer[14], which is the process of taking the style of one image and transferring it to another image. However, in the generative world, this style is actually learned and extracted from the latent space. In fact, the StyleGAN model has two latent spaces: the first is the usual

$\mathcal{Z}$ space, where all the values are picked randomly following a normal distribution. The $\mathcal{Z}$ vectors are then mapped through a sequence of fully connected layers, along with high-level conditional features, into a new intermediate latent space, the $\mathcal{W}$ space. The Generator network is indeed divided into two sub-networks, a mapping network $f : \mathcal{Z} \mapsto \mathcal{W}$ and a synthesis network $g : \mathcal{W} \mapsto \mathcal{Y}$. The synthesis network starts from a learned constant input of size $512 \times 4 \times 4$ which is propagated through a sequence of synthesis blocks. Each synthesis block consists of the following transformations: an upsampling of the previous block, a 3×3 convolution, an additional random noise, an adaptive instance normalization[24] layer guided by the $\mathcal{W}$ vectors, another convolutional 3×3 layer, an additional noise input, and finally a final normalization layer guided by the $\mathcal{W}$ vectors. The loss function used by the authors is based on a non-saturating loss (as introduced in section 2.8) along with path-length regularization and an R_1 gradient penalty regularization[39].

2.9 Regularization for GANs

2.9.1 Gradient penalty regularization

A common technique used in GANs to improve convergence is to introduce a gradient penalty to penalize the discriminator from deviating from the Nash equilibrium[51]. One way to do so is to penalize the gradient of the discriminator only on real data. This comes from the idea that when the generator reaches a true real data distribution level, the discriminator should not produce any gradient without suffering the adversarial loss. This leads to a R_1 regularization term proposed in [39]:

$$R_1 = \frac{\gamma}{2}\mathbb{E}_{x\sim P_x}[\|\nabla D(x)\|^2]$$

A similar regularization idea can be applied to fake data, where the discriminator gradient is penalized on the generator distribution, also known as R_2 regularization:

$$R_2 = \frac{\gamma}{2}\mathbb{E}_{z\sim P_z}[\|\nabla D(G(z))\|^2]$$

2.9.2 Perceptual path-length regularization

Perceptual path-length was initially introduced in [28] as a metric to evaluate the quality of the Generator network. Subsequently, the authors proposed to use this quantity as a regularization term[29]. Path-length regularization is used to encourage the generator to generate images that are close to the training data in latent space

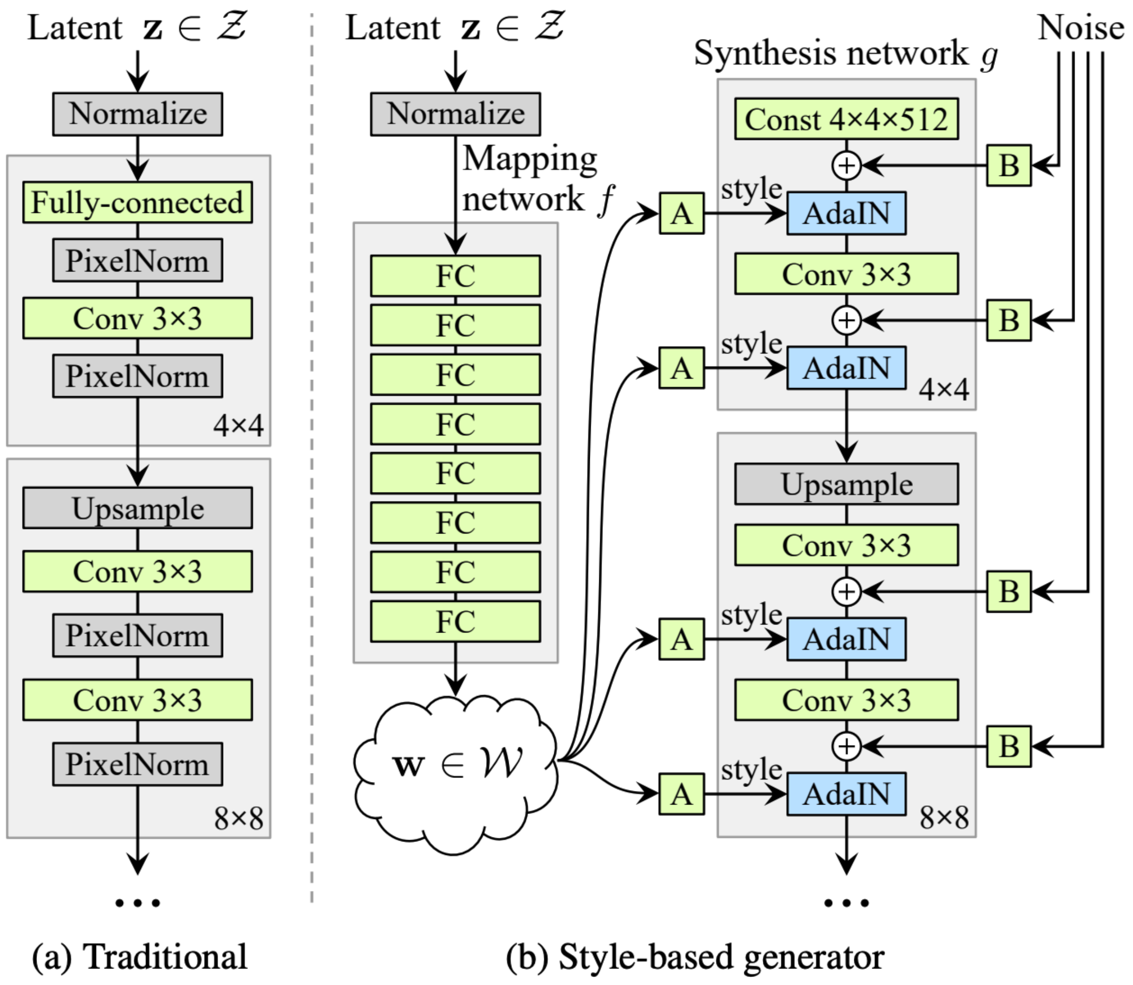


Figure 2.14: The StyleGAN Model [28].

and to avoid drastic changes in the neighbourhood of the latent vectors. The path-length regularization term is added to the loss function as follows,

$$\mathbb{E}_{\boldsymbol{w},\boldsymbol{y}\sim\mathcal{N}(0,\boldsymbol{I})}(\|\boldsymbol{J}_{\boldsymbol{w}}^T\boldsymbol{y}\| - a)^2,$$

where $\boldsymbol{y}$ is a matrix of random images with normal pixel intensity distribution, $a \in \mathbb{R}$ defines the desired scale of the gradient, $\boldsymbol{w} \sim f(\boldsymbol{z})$, with $\boldsymbol{z}$ following a normal distribution, and $\boldsymbol{J}_{\boldsymbol{w}}$ is the Jacobian matrix $\partial g(\boldsymbol{w})/\partial \boldsymbol{w}$ of the generator function (which is not required to be explicitly computed, as it is efficiently derived through backpropagation[2.1]).

2.10 Quality metrics for GANs

2.10.1 Fréchet Inception Distance

Since GANs are trained in an unsupervised manner and the adversarial losses often result in constant oscillation, it is difficult to know when to interrupt the training. One of the most used methods to evaluate the quality of the generator distribution is the use of classification models pretrained on big datasets, such as ImageNet[8], and then measure the similarity between the features extracted from the real samples and the generated samples. One such metric is the Fréchet Inception Distance[20] introduced in 2017. Given a Gaussian (μ_r, C_r) and a Gaussian (μ_g, C_g), the FID is defined as

$$d^2 = ||\mu_r - \mu_g||^2 + Tr(C_r + C_g - 2\sqrt{C_r \cdot C_g}),$$

where Tr is the sum along diagonals of the matrix.

In the context of GANs, one of the Gaussian distributions is parametrized by the mean and covariance of the features extracted through a pretrained model from the real dataset. The other distribution is represented by the mean and covariance of the features extracted from a synthetic dataset with the same pretrained model. It is common in literature to see results mentioning FID10k or FID50k: the suffix refers to the size of the synthetic dataset, which in this case is 10 000 and 50 000 respectively.

Chapter 3

Datasets

In this chapter we provide a description of the core datasets that we use later.

3.1 UniToPatho dataset

UniToPatho [3] is a histological dataset comprising different colorectal polyps samples collected from patients undergoing cancer screening, consisting of 292 whole-slide hematoxylin and eosin (H&E) stained images. The slides are acquired through a Hamamatsu Nanozoomer S210 scanner at 20× magnification (0.4415 μm/px). Each of the 292 slides is related to a unique patient and is annotated by expert pathologists. There are 6 different classes:

- NORM: Normal tissue
- HP: Hpyerplastic Polyp
- TA.HG: Tubular Adenoma, High-Grade dysplasia
- TA.LG: Tubular Adenoma, Low-Grade dysplasia
- TVA.HG: Tubulo-Villous Adenoma, High-Grade dysplasia
- TVA.LG: Tubulo-Villous Adenoma, Low-Grade dysplasia

The dataset is provided with a split ratio of 70% for the training set and 30% for the test set, resulting in 204 slides and 88 slides respectively. The authors also provide two non-overlapping cropped versions consisting of 8669 square images with side of 800μm (1812×1812 pixels patches) and 867 square images with side of 7000μm (15855×15855 pixels patches). The main two tasks for the dataset are:

- polyp type classification between NORM, HP, TA, TVA
- dysplasia grade classification between LG and HG

Crop Size	HP	NORM	TA.HG	TA.LG	TVA.HG	TVA.LG	Total
Whole-slides	41	21	26	146	20	38	292
7000μm	59	74	98	411	93	132	867
800μm	545	950	454	3618	916	2186	8699

Table 3.1: UniToPatho class distribution

The dataset distribution is shown in Table 3.1. One characteristic of the dataset is its size, with only 292 total samples. Additionally, it also contains a strong imbalance in the class distribution. Out of 6 different classes, the TA.LG class alone counts for half of the total dataset. Furthermore, when dealing with the dysplasia grade classification task there are a total of 184 whole-slide samples for the low-grade class, while for the high-grade class there are just 46 whole-slide samples. Considering the provided train-test split, this means that there are only 32 unique whole-slide samples available for training. As we show later, these imbalances have shown to be a crucial obstacle when developing automatic classification algorithms.

Some examples of the dataset are visible in Figure 3.1.

3.2 PanNuke Dataset

PanNuke[13] is a histological dataset containing 19 different tissue types. Each tissue is semi-automatically annotated, providing the classification of nuclei in 5 different classes: inflammatory, neoplastic, connective, dead, and epithelial. It is built upon more than 20k whole slide images collected at different magnifications and from multiple data sources. The dataset allows to train models capable of generalizing segmentations to new tissue types. This generalization power has been further confirmed for the UniToPatho dataset[9].

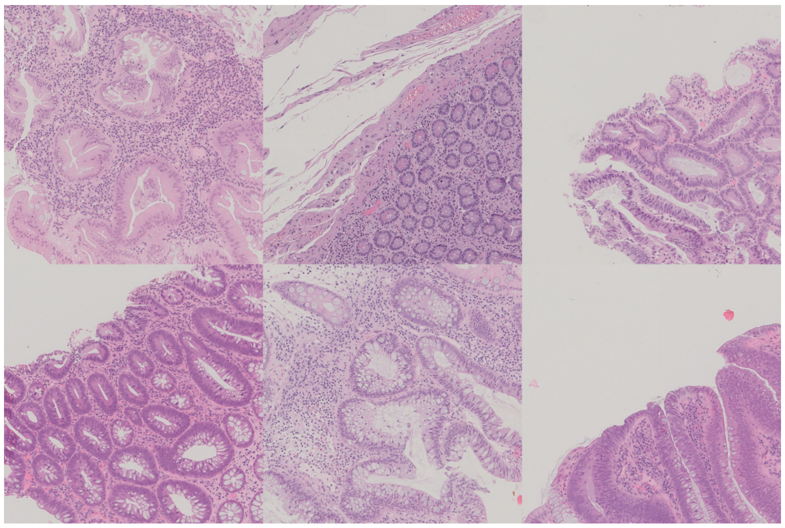


Figure 3.1: 1812px resolution images from the UniToPatho dataset. Classes from left to right, top to bottom: HP, NORM, TA.HG, TA.LG, TVA.HG, TVA.LG

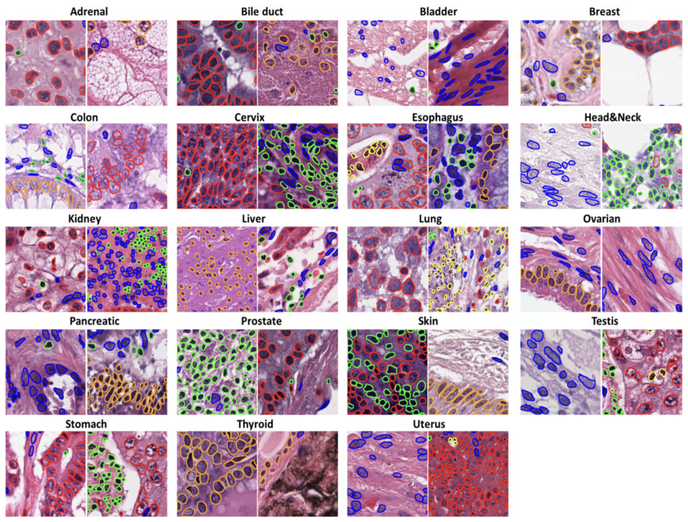


Figure 3.2: Samples from exhaustively annotated PanNuke dataset, which contains image patches from 19 tissue types for nuclei instance segmentation and classification (Red: Neoplastic; Green: Inflammatory; Dark Blue: Connective; Yellow: Dead; Orange: Epithelial)[13]

Chapter 4

Related Work

In this chapter we provide an overview on related works covering UniToPatho [3] and the applications of GANs to medical computer vision.

4.1 GANs for Medical Computer Vision

The generative power of GANs has raised a lot of interest for medical imaging, especially when dealing with small and unbalanced datasets. Many works [32, 58, 60] show that GANs can be used to generate synthetic data that can be used to improve classification tasks.

Stain normalization

Histological images use stain to highlight different parts of the biological tissues. The stain process can vary from one laboratory to another, so it is common to encounter inconsistencies between datasets. For this reason, Xu et al. [59] focus on a GAN-based model with the goal of normalizing the stain of different datasets. They propose a conditional CycleGAN [63] network to transform H&E stained images into IHC stained images. They show that even with a limited amount of data they are able to succeed in the normalization task.

Style transfer

Hou et al. [22] propose a GAN applied to histopathological data as a style-transfer method. They first train a model capable of generating synthetic histopathology images with desired characteristics such as locations and sizes of nuclei. Then they use a GAN model that modifies the synthetic images by transfering the style of randomly selected real samples. Finally, they build a synthesis pipeline for a histopathological segmentation task and achieve a 6% improvement in segmentation error.

Synthesis for augmentation

Frid-Adar et al. [12] apply GAN models to a limited dataset of computed tomography (CT) images of liver lesions. They first train a GAN model on the real data and then generate a synthetic dataset to augment the training set. After that, they train a CNN on the real training set and a CNN on the augmented dataset for classification. As a result, they achieve a 7.1% increase in sensitivity (from 78.6% to 85.7%) and a 4.0% increase in specificity (from 88.4% to 92.4%) thanks to the synthetic images.

4.2 Related work on UniToPatho

The UniToPatho dataset has been studied in three main areas ranging from Classification, to Segmentation and Generation.

Classification

Barbano et al. [3] propose different solutions to the classification task on UniToPatho, tackling it at different sizes. They use an ensamble of multiple cascading ResNet-18 [19] models trained on both the $\sigma = 800$ and $\sigma = 7000$ crops. The model has an initial ResNet-18 that discriminates between HP and non-HP class at $\sigma = 800$. After this discrimination, the non-HP classes are then discriminated at $\sigma = 7000$ with another ResNet-18 which classifies between NORM, TA and TVA class. Finally, for the TA and TVA predictions, another ResNet-18 is used at $\sigma = 800$ which classifies between LG and HG dysplasia. Overall, the final model achieves a balanced accuracy of 67.0% over the 6 classes.

The most clinically important task for pathologists is the differentiation between the two grades of dysplasia. By focusing only on this task, the authors achieve a balanced accuracy of 80.5% after training a ResNet-18 model on the 1812px crops. This leaves the challenge for accurate automatic grade classification still open.

Segmentation

In another work, Di Luccio et al.[9] propose a segmentation model for nuclei of UniToPatho images. The segmentation consist in the extraction and classification of all the nuclei in a given image. Since the UniToPatho dataset does not have segmentation masks, Di Luccio et al. apply transfer learning starting from the PanNuke dataset. They train a UNet++[62] network on top of annotated colon scans of the PanNuke dataset and achieve great results, with an IoU (Intersection over Union) score of 0.84. An example of the segmentation is shown in Figure 4.1.

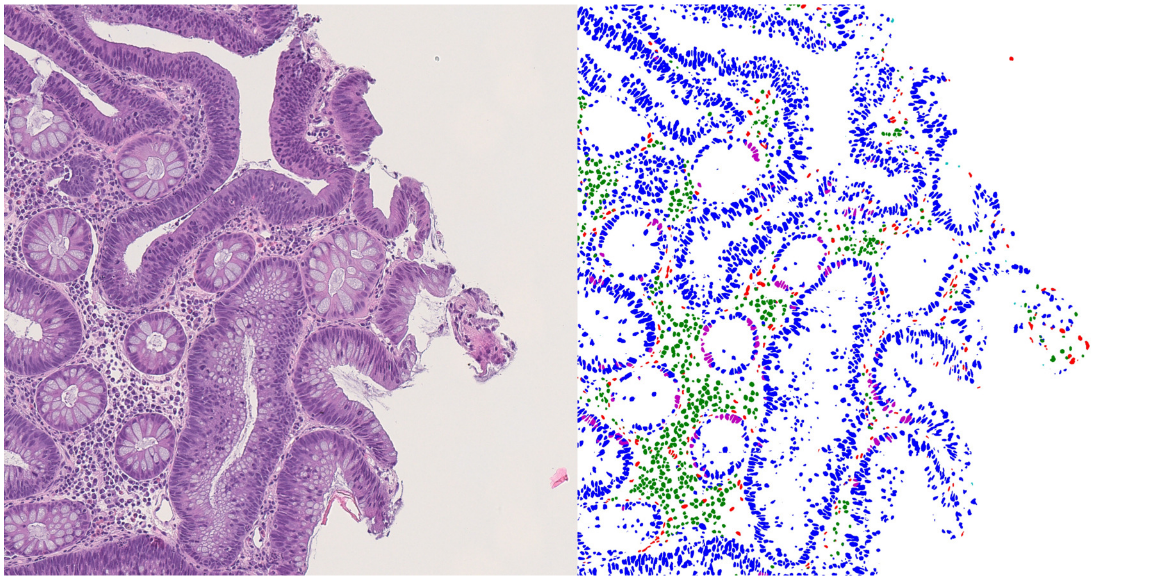


Figure 4.1: On the left, an UniToPatho image. On the right, its segmentation [9]. Blue: Neoplastic cells. Green: Inflammatory. Red: Connective/Soft tissue cells. Cyan: Dead Cells. Magenta: Epithelial.

Generation

In a subsequent work, Rubinetti et al.[49] used the segmentation masks extracted in [9] to build a generative model. Their goal was to build a model able to transform segmentation masks into realistic histological images with the aim to use prior medical knowledge to generate new samples for the UniToPatho dataset. They trained a standard Pix2Pix[26] model on both PanNuke and UniToPatho and succeeded in their generative task at a 256px crop size. However, when scaling their model to higher resolutions, such as the most significant 1812px resolution[3], the quality degrades significantly.

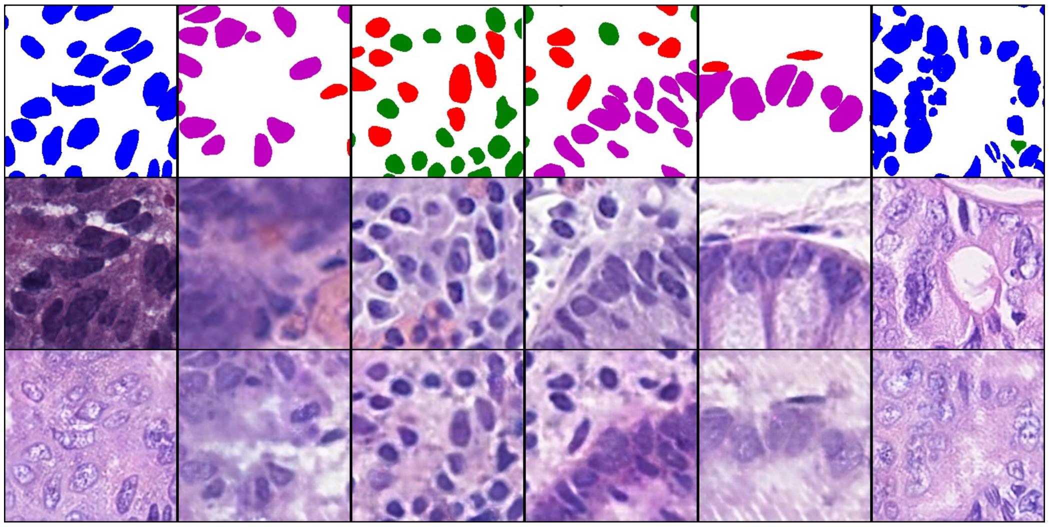


Figure 4.2: Prior generation work on UniToPatho at low resolution[49]. First row: segmentation masks. Middle row: real images. Right row: reconstructed images.

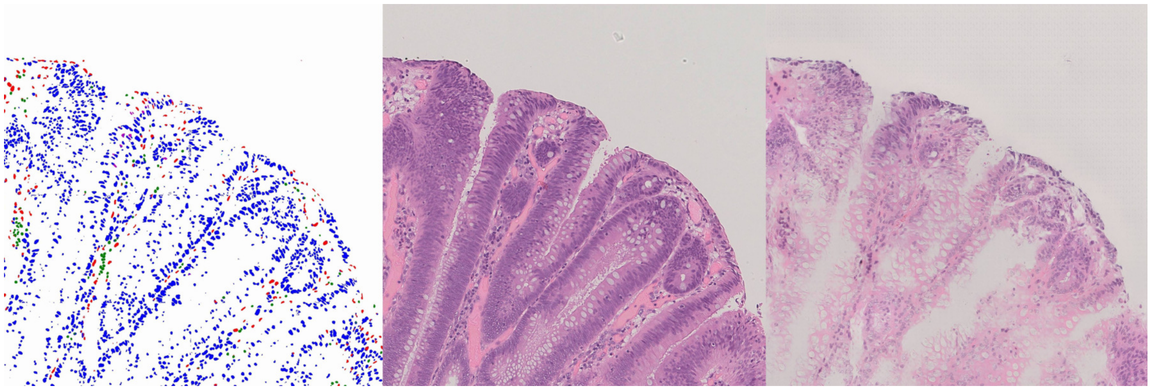


Figure 4.3: Prior generation work on UniToPatho at high resolution[49]. Left image: segmentation masks. Middle image: real images. Right image: reconstructed images.

Chapter 5

Generation

GANs have recently shown astonishing results in the photo-synthesis field, generating high-quality photo-realistic images of faces, animals and even 3D scenes. We experiment with the use of such models in the medical computer-vision field. In our case, we use a GAN model applied to the UniToPatho dataset in an attempt to generate new samples with the goal to increase the accuracy of classification models. We only focus on the dysplasia grade label of the dataset, since the distinguishment between low-grade and high-grade dysplasia is the most influential target under clinical perspective [17, 18, 38, 50]. In this chapter, we provide a description of the models and methods used for training the GAN, along with some initial results for the quality of the generations.

5.1 Method

For all of the following GAN models we use a StyleGAN2[29] architecture, with 8 mapping layers, and a size of 512 for both the $\mathcal{Z}$ and $\mathcal{W}$ latent spaces. We use a non-saturating adversarial loss, path-length regularization and R_1 gradient penalty. The overall objectives are:

$$\mathcal{L}_g = -\mathbb{E}_{z\sim P_z}[\log(d(g(z)))] + \gamma_{pl}\mathbb{E}_{w\sim P_w, y\sim\mathcal{N}(0,\boldsymbol{I})}(\|\boldsymbol{J}_{\boldsymbol{w}}^T\boldsymbol{y}\| - a)^2 \tag{5.1}$$

$$\mathcal{L}_d = -\mathbb{E}_{x\sim P_x}[\log(d(x))] - \mathbb{E}_{z\sim P_z}[\log(1-d(g(z)))] + \gamma_{R_1}\mathbb{E}_{x\sim P_x}[\|\nabla_x d(x)\|^2] \tag{5.2}$$

As described in section 2.8, this adversarial loss is the alternative loss to the original min-max loss proposed by Goodfellow et al. [15] that avoids the saturation of the Generator network. The regularization term in Equation 5.1 is the path-length regularization that encourages the Generator to avoid drastic changes in the neighbourhood of sampled latent vectors mapped into $\mathcal{W}$ space. The regularization term in Equation 5.2 is the R_1 gradient penalty, which forces the discriminator to avoid producing gradients without suffering from the adversarial loss when the generator reaches the true real data distribution. The structure of the two losses is the one proposed in [29].

Since our dataset is quite small, we apply an augmentation pipeline to the images shown to the discriminator, as proposed by Karras et al. [30]. We include horizontal and vertical flipping with $p = 0.5$ and random 90-degree rotation with $p = 0.5$. Additionally we use random saturation, contrast and brightness manipulation with $p = 0.5$ and a random intensity in the range of ± 0.2.

Initially, we train a conditional GAN on dysplasia grade with 2-sized one-hot encoding of the label. The purpose of this model is to have a single GAN that learns the distributions of both the high-grade and low-grade samples. However, since the two classes are very similar, we hypothesise that given the strong imbalance in the dataset, the conditional GAN trained on both classes could eventually lean only towards the majority class (low-grade). For this reason, we also train two separate unconditional models, one for the low-grade class and one for the high-grade class. This way, the two networks never see the other class and cannot mix their features. We train the three models on both 512px and 2048px and compare them through a FID50k score and an initial experiment with experts.

5.2 Setup

For the training we use the Adam[33] optimizer with $\beta_0 = 0$, $\beta_1 = 0.99$, $\epsilon = 1e-8$, learning rate of 0.0025. For the R_1 regularizer we use a weight of 6.5 and for the path-length term we use 2.0 as weight. These hyperparameters were used by Karras et al. [31] and are usually suitable for a lot of different datasets and GAN settings.

Data preprocessing

We extract two versions from the UniToPatho dataset that we later use for GANs: a cropped version at 512px resolution and a resized version at 2048px resolution:

- 512px resolution: we extract 16 partially overlapping 512px resolution crops out of each image. By doing so we obtain 139.2k images starting from 8.6 images. We notice that some of the crop images contain very low information, to such an extent that some are completely white. We manually label some of these low-content crops and fit a simple SVM model on the outputs of a pretrained ResNet on ImageNet to filter out the dataset. As a result, we end up with 79.3k images, of which 7.5k high-grade and 39.9k low-grade samples are used for training;

- 2048px resolution: using power-of-two resolutions when training GANs allows to easily scale and transfer the weights of lower resolution models to higher resolution ones, thus we resize the 1812px UniToPatho images to 2048px with a bilinear interpolation.

Training on 512px resolution

Karras et al. [29] released a pretrained GAN on BreCaHAD[1], a histological dataset for breast cancer analysis. In a first experiment, we extend their model and fine-tune it for UniToPatho. The BreCaHAD model is trained on 512px resolution images, so we use the 512px version of the UniToPatho dataset. We train the GAN with a batch size of 32 until the FID50k metric starts diverging, resulting in a total of 4.1M examples shown to the discriminator after 5 days and 19 hours on a single Nvidia A40 GPU.

Training on 2048px resolution

State of the art GANs are usually trained up to 1024px resolutions, but since the dysplasia-grade is better classified at higher resolutions [3] we chose to train the GAN model on the full-sized 2048px resolution images.

Similarly to the previous version, we train with a batch size of 32 for 8 days and 5 hours on a single GPU, for a total of 940k images shown to the Discriminator.

5.3 Interpolation

As an initial evaluation of our model, we generate some examples to check their visual quality (Figure 5.1). Additionally, we explore the latent space through the common technique of interpolation between vectors. If we pick two random latent vectors following the initial normal distribution in $\mathcal{Z}$, the generator creates two mostly different images. However, if we interpolate between these two latent vectors and pick some points in the interpolation path, we can visualize the samples in-between the two generations. This is also a frequent approach used to determine whether the generator is actually generating new images or it has just memorized the samples. Sharp transformations in the interpolated generations are usually a sign of overfitting. In our case, we take evenly-spaced steps in each of the values of the latent vectors from one to the other and, as visualized in Figure 5.2, our model shows natural transformations.

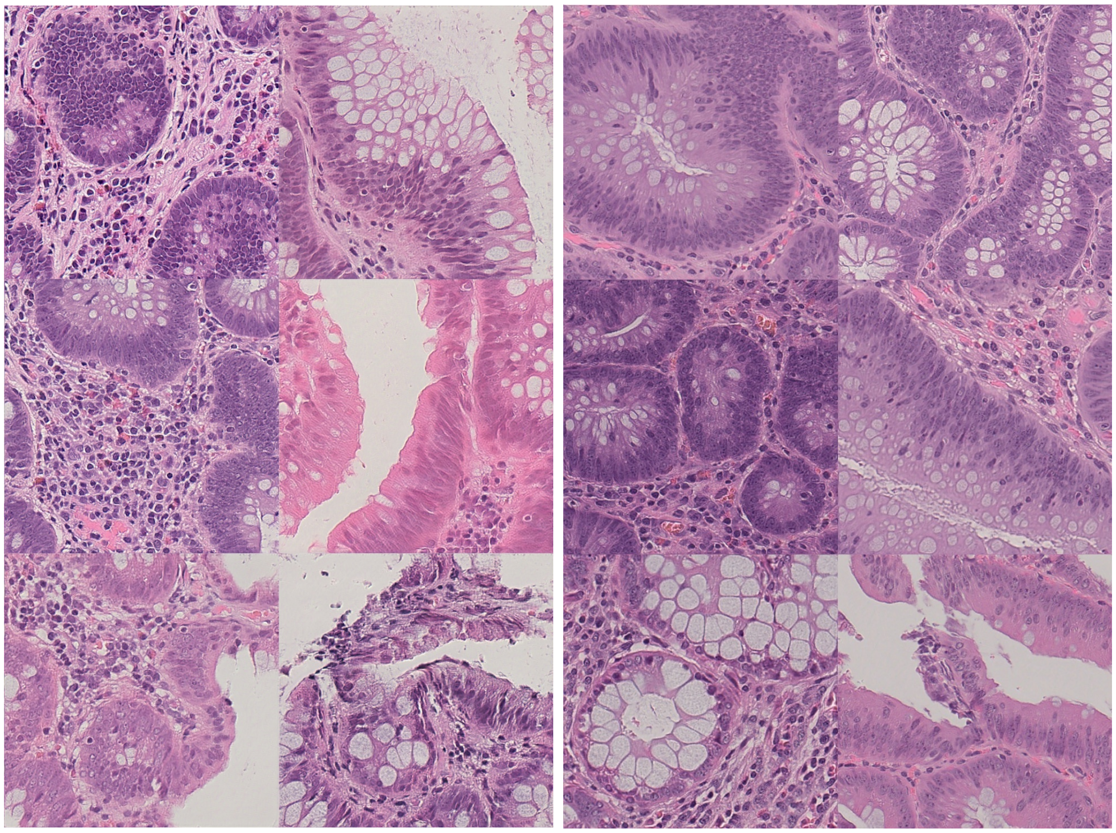


Figure 5.1: Comparison between real (left two columns) and generated (right two columns) images at 512px resolution.

Learned evolution of tissues visualized through interpolation

Doing interpolations in the latent space $\mathcal{Z}$ does not allow to interpolate between different classes, since the classes are represented as a discrete one-hot encoding. However, our GAN model mixes both the latent space $\mathcal{Z}$ and the label into a continuous latent space $\mathcal{W}$, thus allowing for interpolation between different classes. If we use the same latent vector $\mathcal{Z}$ and map it with two different classes into $\mathcal{W}$, we can visualize how the generator distinguishes the two labels over the same encoding. Furthermore, we can interpolate between the two corresponding vectors in $\mathcal{W}$ and visualize how a low-graded tissue evolves into a high-grade tissue with the same characteristics encoded by the same latent vector $\mathcal{Z}$, as shown in Figure 5.3.

5.4 Experts' evaluation

To additionally check whether or not the GAN is generating realistic images, we run two experiments with expert pathologists from the Department of Medicine of the University of Turin.

Initially, we run a classification experiment by sampling 10 low-grade images and 10 high-grade images at 2048px resolution, and ask the experts to annotate them. The resulting accuracy is 85%, where 20% of the high-grade examples got misclassified as low-grade ones, and 10% of the low-grade examples got misclassified as high-grade ones.

As a second test, we run a fake/real experiment. We randomly pick 10 real

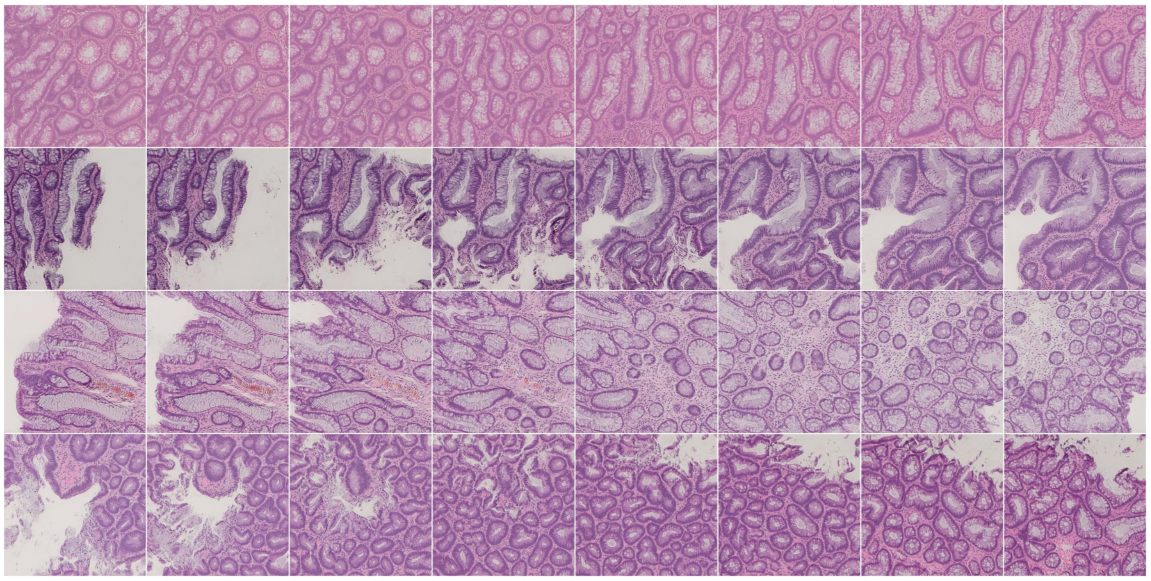


Figure 5.2: Visualization of the interpolation between random latent $\mathcal{Z}$ vectors of the same class. In each row, the first and last images are generated from randomly sampled vectors. The images in-between are the visualizations of each sequential step taken from the first latent vector towards the last one (2048px model).

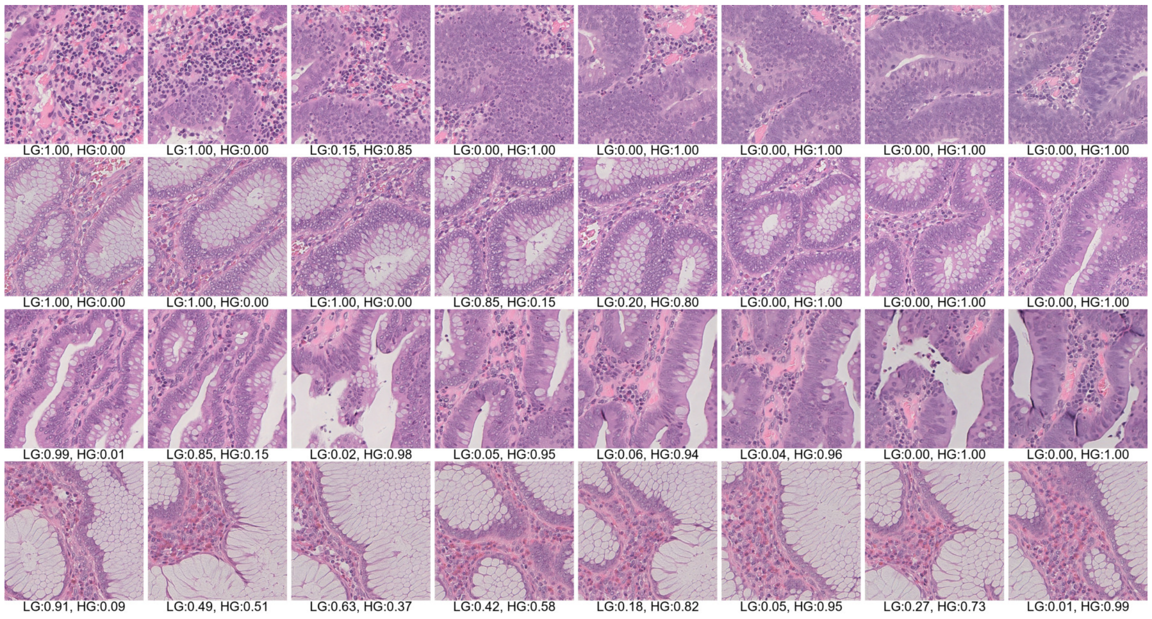


Figure 5.3: Visualization of the interpolation between low-grade latent vectors and their corresponding high-grade latent vector. All examples on each row start from the same latent vector in $\mathcal{Z}$. On the first column the $\mathcal{Z}$ vectors are mapped into $\mathcal{W}$ with a low-grade label, whereas the last column corresponds to the $\mathcal{Z}$ vectors mapped into $\mathcal{W}$ with a high-grade label. The images in-between are the visualization of the interpolations between the two $\mathcal{W}$-space latent vectors in each row. Under each image we show the classification prediction of our ResNet18 model trained on real examples (512px model).

examples and randomly synthesize 10 fake examples. We run the experiment for both 512px and 2048px resolutions and ask the experts to check which examples are generated and which ones are real. The resulting accuracy for the 512px resolution images is as low as 52.6%, which demonstrates that the quality of the generated samples is very high, to such an extent that they are indistinguishable from real ones, even by experts. Regarding the 2048px resolution, the pathologists achieve 65.7% accuracy, which shows that on higher resolutions the quality slightly decreases. Experts also noted that the 2048px fake images are easier to spot because there's more space for unrealistic patterns.

Experiment	Resolution	Accuracy
Synthetic labeling	2048px	85.0%
Fake vs Real	2048px	65.7%
Fake vs Real	512px	52.6%

Table 5.1: Results of the evaluation by experts

5.5 FID Results and discussion

We evaluate all the GAN models with a FID50k metric. As shown in Table 5.2, lower resolution models perform better than higher resolutions ones, with a minimum FID50k of 3.6, which further confirms the initial observations by experts in the previous section. The reason for this is that the images at 2048px resolution, aside from having higher size, contain additional global information about the pattern of the tissue which is absent from the low-grade images. Additionally, for the 2048px resolution there are only 5k examples compared to the 40k examples at 512px resolution.

Conditioned models also perform slightly better than unconditioned ones, which is most probably caused by the higher amount of examples when considering both classes together. Furthermore, we notice that HG-only models achieve a very high FID50k compared to the LG ones. In fact, this is caused by the strong imbalance in the dataset, which has a much lower amount of examples for the high-grade class compared to the number of low-grade samples.

Resolution	Dysplasia Grade	FID50k
512px	both (conditioned)	3.6
512px	low grade	5.4
512px	high grade	7.9
2048px	both (conditioned)	9.1
2048px	low grade	8.6
2048px	high grade	24.2

Table 5.2: Comparison of the FID50k on different resolutions.

Chapter 6

Classification with GAN-augmentation

As mentioned earlier, our primary goal is to exploit the GAN for augmentation of the UniToPatho training set. We only focus on the 2048px images, because, as Barbano et al. [3] show, the dysplasia grade of the UniToPatho dataset is best detected at higher resolutions.

6.1 Method

For the classification task we use ResNet-18 models pretrained on ImageNet1k [8]. We replace the last fully-connected layer with a new fully-connected layer with 2 outputs, one for the probability of low-grade dysplasia and the other for the probability of high-grade dysplasia. The objective of the training is thus a cross-entropy.

First, we train a ResNet-18 model only on real data to determine the baseline accuracy limit of the training set. After that, we use our GAN to generate a dataset of the same size of the real training set for augmentation purposes. To evaluate the quality of this synthetic dataset, we first compute the Classification Accuracy Score(CAS) [44], which consist in training a separate classifier on top of only generated samples and evaluate its accuracy against a real test set. We then proceed with the augmentation method by training ResNet-18 models on the training data augmented with the synthetic dataset. We experiment with two proportions of augmentation:

- 1.5x augmentation: we add 50% samples per class of the training set, which is equivalent to 1499 low-grade samples and 572 high-grade samples;

- 2x augmentation: we add 100% samples per class, by in fact doubling the dataset size.

Finally, we follow the idea of using GANs as both trained feature extractors [43, 37] and for anomaly detection [10]. We take the Discriminator model trained only on low-grade 2048px images and extract the features of the last two convolutional blocks for all the real training set and test set. We flatten and concatenate these results and fit an SVM model. We apply a grid search on the training set to determine the most suitable kernel type, regularization parameter strength, and polynomial degree in case of polynomial kernel. We then evaluate the accuracy of the SVM on the test set features.

6.2 Setup

For all the ResNet-18 models trained for classification we always keep the same training setting in order to evaluate the impact of the synthetic samples. We use the Adam [33] optimizer with a batch size of 256, a learning rate of 0.01 and a weight decay of 1e−5. We also apply an augmentation pipeline with random horizontal and vertical flipping ($p = 0.5$), random rotation in the range of $[-90°, +90°]$ with reflection padding ($p = 0.5$) and random color jittering of brightness, contrast, hue and saturation in the range of $[0.8, 1.2]$. Additionally, we normalize the colors of the images to the mean and standard deviation of the images of ImageNet. Furthermore, as suggested by Barbano et al. [3], we balance the training set by randomly replicating the high-grade samples and randomly reducing the number of low-grade samples such that the final training set size is the same as the initial one but the number of examples per class is the same.

6.3 Results and discussion

The final combined classification results are shown in Table 6.1.

Baseline results. For the baseline ResNet-18, averaging from 3 runs we achieve a balanced accuracy of 79.1%, a sensitivity of 0.732 and a specificity of 0.852.

Synthetic-only results. As an initial check, we evaluate the trained baseline model on the synthetic dataset to classify the generated samples. As a result we

Experiment	Balanced Accuracy	Sensitivity	Specificity
Baseline Resnet18	79.2% ± 0.4%	**0.732 ± 0.048**	0.852 ± 0.051
Synthetic-only	78.2% ± 0.9%	0.686 ± 0.032	0.877 ± 0.020
Augmented with 50%	**79.8% ± 1.0%**	0.696 ± 0.017	0.899 ± 0.012
Augmented with 100%	78.7% ± 0.6%	0.722 ± 0.014	0.853 ± 0.011
Unsupervised LG GAN+SVM	77.2%	0.621	**0.919**

Table 6.1: Training results on the test set averaged out of 3 runs.

obtain a balanced accuracy of 84.1%, which is almost the exact same accuracy evaluated in the experiment with experts, as described in section 5.4.

Additionally, after training a new ResNet18 only on the generated samples, we achieve a balanced accuracy of 78.2% on the test set, averaged from 3 different runs. The resulting sensitivity decreases to 0.686, compared to the baseline model, whereas the specificity increases to 0.877.

Augmentation results. In the first version of 50% augmentation, the resulting balanced accuracy slightly improves over our baseline model with a 79.8% score, averaged from 3 different runs. However, similarly to the synthetic-only conclusion, there is a loss in sensitivity and a gain in specificity, with 0.705 and 0.927 respectively.

On the other hand, for the 100% augmentation we achieve a balanced accuracy of 78.7%, a sensitivity of 0.722, and a specificity of 0.853.

GAN Discriminator results. After applying a grid-search on the hyperparameters of the SVM (kernel type, regularization parameter strength, polynomial degree in case of polynomial kernel) on the training set, with a cross-validation K-fold of 5 we achieve a 77.2% balanced accuracy on the test set. The result is quite impressive: without ever showing high-grade examples to the GAN, the SVM almost approaches the accuracy of the baseline ResNet model.

Discussion

As shown in the previous sections, the GAN is capable of learning the characteristics of the two classes and generates samples that are very similar to the real data. The results of the SVM experiment show that the discriminator, after training only on low-grade samples, can be exploited as a feature extractor for anomaly detection and classification on previously unseen classes.

Furthermore, the results of the CAS experiment show that the generated samples are almost as representative as the real ones. This demonstrates that, at the price of a small accuracy loss, GANs can be used to anonymize real data by generating almost-equivalent synthetic datasets.

Finally, when using synthetic data for augmentation, the experiments show that it can achieve a slight (but not statistically significant) increase in the average accuracy. However, in all the experiments we noticed a decrease in sensitivity and an increase in specificity. This suggests that the GAN still suffers from the imbalance of the dataset and leans more towards the majority class (low-grade).

Chapter 7

Image-to-image Translation

Sampling random images for augmentation only from random latent vectors shows very promising results, but does not allow to fully solve the classification tasks on the UniToPatho dataset. This is mostly due to the fact that the high-grade dysplasia class has very few examples compared to the low-grade one, thus the GAN cannot learn the true distribution for HG samples. For this reason, we now focus on a different approach for the generation task. Instead of sampling images just from random noise, we encode high-level image features such as segmentation masks of nuclei into an intermediate latent space and use this information to guide the synthesis process. This enables to manually craft realistic and meaningful samples for augmentation by exploiting prior expert medical knowledge and focusing the generation on the minority class.

7.1 Method

The StyleGAN[29] architecture that we used in the previous experiments showed high-quality results. For this reason, we extend it to an image-to-image setting, as similarly done by Richardson et al. [46].

Initial idea

In a first experiment, we attempt to extend the StyleGAN model with an additional variational autoencoder setting[34]. Since the input of the Generator network of the StyleGAN follows a normal distribution, we introduce an encoder network that maps segmentation masks into an intermediate space describing the mean and variance of

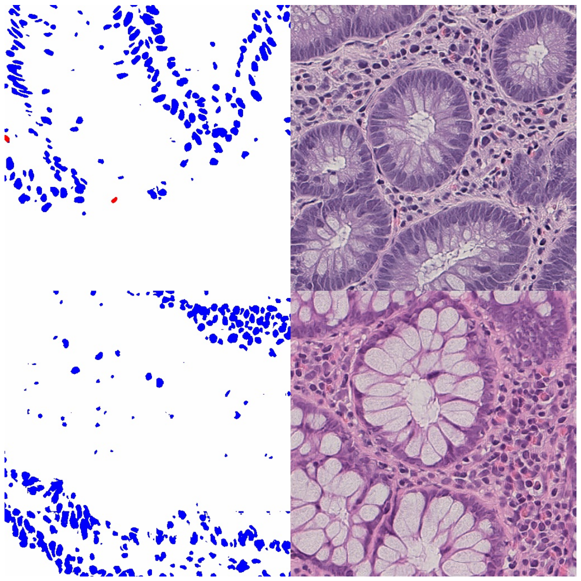


Figure 7.1: Result of the image-to-image model with variational encoding. The resulting images are of high-quality, but strongly unrelated from the segmentation masks.

the normal distribution. These parameters are then used to model a normal distribution and sample random vectors from it. We train the full network similarly to the previous experiments, with an additional loss for the Kullback–Leibler divergence[45] and reconstruction loss. After experimenting with different hyperparameters for the losses and encoder structure, we notice that all the generations are of arguably high quality, but strongly unrelated from the segmentation masks. An example is shown in Figure 7.1.

We hypothesise that this outcome is caused by the randomness of the reparametrization which eventually causes completely different masks to be mapped into very similar latent vectors, thus causing an instability in the training.

Final architecture

Instead of using a variational encoder, we experiment with a different approach that has less generative power, but allows the high-level details of the masks to flow properly into the generator network. We introduce a new Encoder that allows to inject the segmentation masks of nuclei directly into the synthesis network. This network is composed of ResNet[19] blocks and additional skip connections that, similarly to a U-Net[47], concatenate the output of each block into the corresponding resolution blocks in the synthesis network. Before the concatenation we use an additional instance normalization layer [55], and we keep the same mapping network and learned synthesis input of the StyleGAN architecture. A high-level design of the generator network is summarized in Figure 7.2.

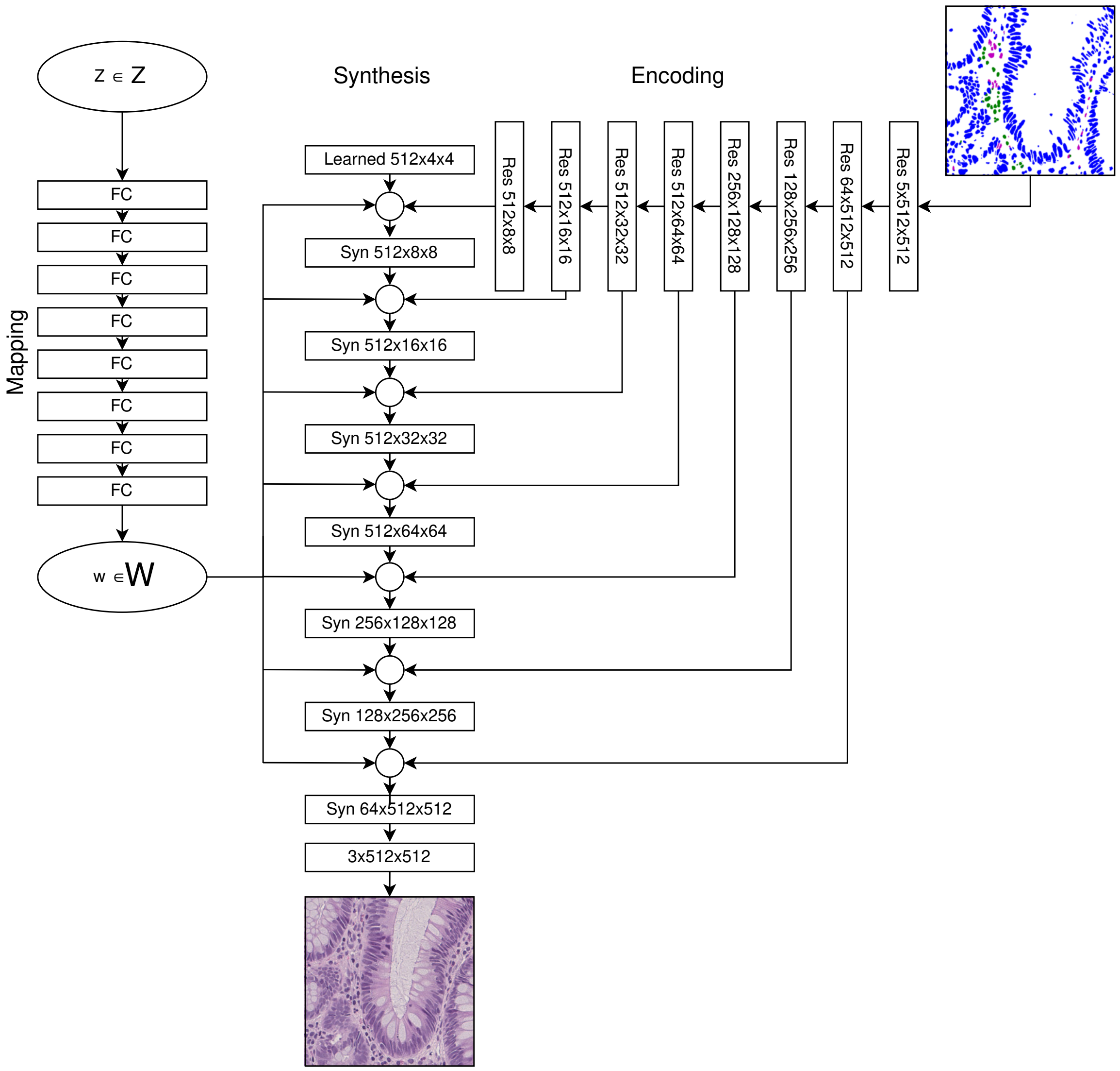


Figure 7.2: Architecture of the extended generator. The Mapping and Synthesis networks follow the StyleGAN2[29] architecture. The Encoder network uses skip connections similar to the U-Net[47] architecture.

For the Discriminator network we use the same architecture of the previous experiments with additional input channels for the segmentation masks. This allows the Discriminator to also learn a correct relationship between segmentation masks and real images, as done in [26].

The total number of parameters is 94.1M for the 512px model and 94.5M for the 2048px model.

We use the same adversarial loss and regularization terms used for the GANs in the previous experiment. Furthermore, we experiment with an additional L1 reconstruction loss, which usually works better with image-to-image translation[26]. The final objective functions are:

$$\mathcal{L}_g = -\mathbb{E}_{z,s}[\log(d(g(z,s)))] + \gamma_{pl}\mathbb{E}_{w,y\sim\mathcal{N}(0,\boldsymbol{I})}(\|\boldsymbol{J}_{\boldsymbol{w}}^T\boldsymbol{y}\| - a)^2 + \lambda\mathbb{E}_{z,s,x}[\|x - g(z,s)\|_1],$$

$$\mathcal{L}_d = -\mathbb{E}_{x,s}[\log(d(x,s))] - \mathbb{E}_{z,s}[\log(1 - d(g(z,s)))] + \gamma_{R_1}\mathbb{E}_{x,s}[\|\nabla_{x,s}d(x,s)\|^2],$$

where s refers to segmentations and λ is a hyperparameter referring to the weight of the reconstruction term.

7.2 Setup

We train all the models on a single Nvidia A40 GPU with the Adam [33] optimizer, a learning rate of 0.0025, a batch size of 32 and an augmentation pipeline consisting of random rotation, random flipping, and random jittering of hue, contrast, saturation and brightness.

For the 512px models, we train for an average of 14 days, resulting in 74M examples shown to the discriminator. We then train the higher-resolution model by initially copying the weights of the smaller one on the shared components of the networks. We train the 2048px models for an average of 8 days with a total of 1.1M examples shown to the discriminator.

7.3 Results and discussion

As shown in Figure 7.3, our model successfully produces high-quality and high-resolution reconstructions. We experiment with different weights for the reconstruction loss and, contrary to other works [26], this loss term does not provide any advantage in the synthesis. In fact, visual quality improves for lower weights of the term. In addition, the lowest FID50k result that we achieve is 3.46, without any reconstruction loss. We argue that this behaviour is caused by the fact that our synthesis is also guided by random latent vectors defining the style that are independent from the segmentation masks. This leads the reconstruction loss to try associating the same image to multiple style vectors, which creates instability in the training of the Generator. Furthermore, the removal of the reconstruction loss demonstrates that the Discriminator network is capable of autonomously learning the relationship between segmentations and images solely from the adversarial loss.

Reconstruction loss weight	FID50k
$\lambda = 0.0$	**3.46**
$\lambda = 2.0$	5.34
$\lambda = 10.0$	5.62

Table 7.1: Comparison of the FID50k with different reconstruction loss weight on the 512px resolution images.

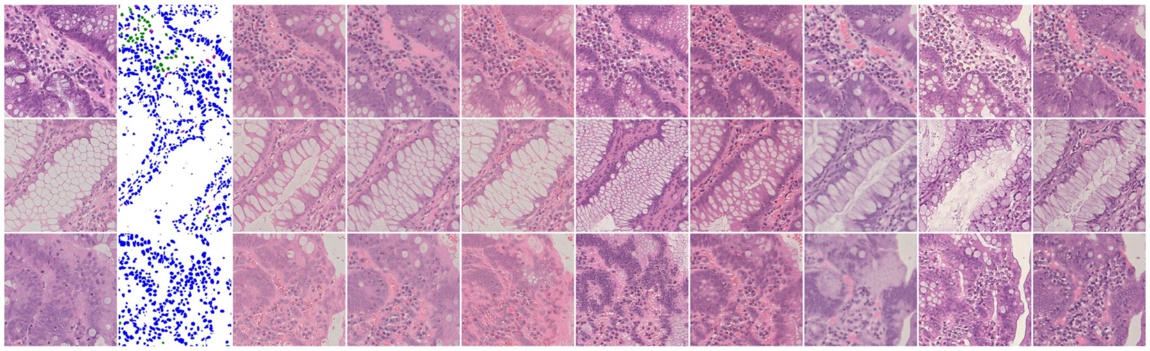


Figure 7.3: Images generated by our model. The first column contains the real images. The second column contains the segmentation masks. The rest of the columns are generated by the GAN. Each of the generations columns represents a randomly sampled $\mathcal{Z}$ vector, which enforces the same style across different input masks.

Compared to prior work on UniToPatho [9], our model drastically improves the visual quality, especially on higher resolutions, as shown in Figure 7.4. Additionally, the FID50k metric also determines significantly lower distances for our model, as described in Table 7.2. Furthermore, compared to the prior work which learns a deterministic mapping between segmentation masks and images, our model has more generative power thanks to the additional latent space that can be use to guide the style of the generation. As shown in Figure 7.3, we can randomly draw input vectors from the $\mathcal{Z}$ latent space and apply them to the same segmentation masks. By doing so, the generator keeps the positioning of nuclei enforced by the segmentation masks intact in the generation, while still producing different realistic variations of the tissues. On the other hand, when we apply the same latent vector to different segmentation masks, the generator keeps the same style in the generations. This demonstrates that our GAN learns to separate the input space of segmentation masks from the overall style space.

Model	Train 512px	Test 512px	Train 2048px	Test 2048px
Pix2Pix-based [49]	37.6	36.2	82.1	88.1
Ours	**3.46**	**4.32**	**5.33**	**6.12**

Table 7.2: Comparison of the FID metric between our model and prior work [49].

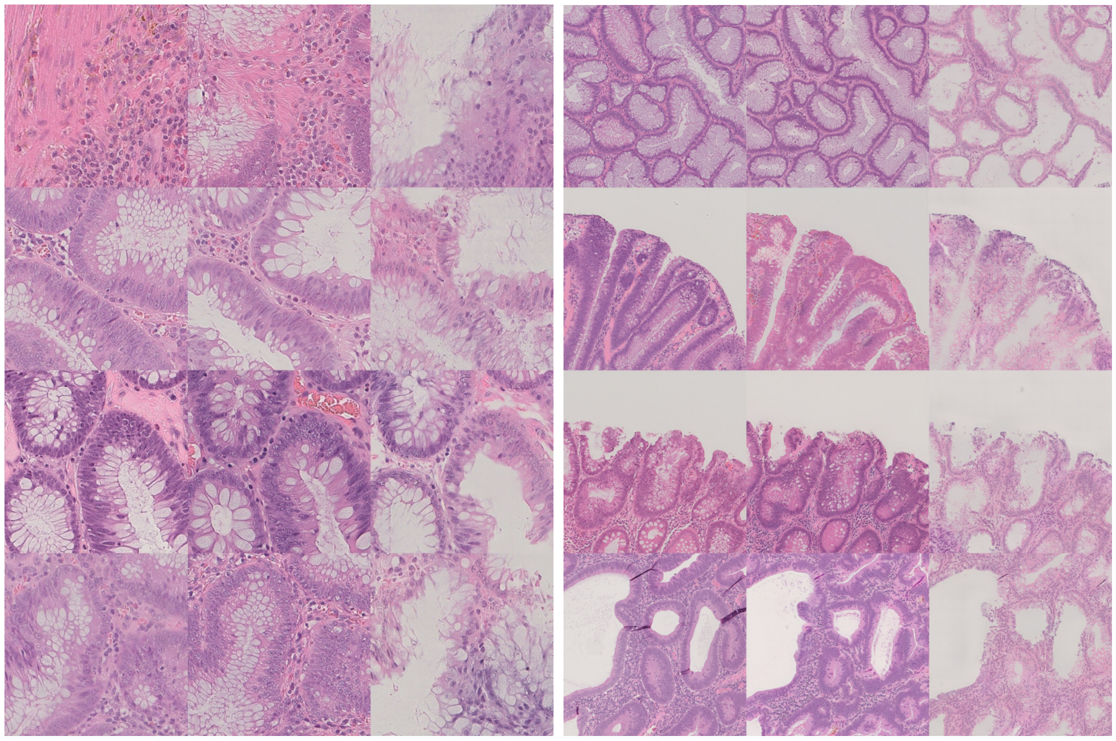


Figure 7.4: Comparison with prior work [49] at 512px resolution (left) and 2048px resolution (right). Columns in order: ground truth, ours, prior work.

Conclusion

We explored various applications of GANs on the UniToPatho dataset. In a first experiment we implemented and trained GANs on different resolutions with the goal of synthetizing new images for aumgmentation. Preliminary analysis on this approach showed interesting results, such as the learned tissue evolution between dysplasia grade, and the difficulty of experts to distinguish real eaxmples for generated ones. Subsequently, we trained different ResNet models on both the real dataset and multiple synthetic versions, achieving small (but not statistically significant) improvements in average accuracy. We showed that our GANs also suffer from the strong imbalance in the data, thus not enabling us to fully solve the classification task. For this reason, we studied a different approach for the generation that allows to guide the synthesis process with high level tissue features, such as segmentation masks of nuclei. For this purpose, we implemented a new generative model based on our initial GAN and improved the image quality and scalability of prior similar attempts. We succeeded in the training of this model and will further explore the augmentation path by exploiting expert medical knowledge for the synthesis of more meaningful samples.

Appendix A

More Generations

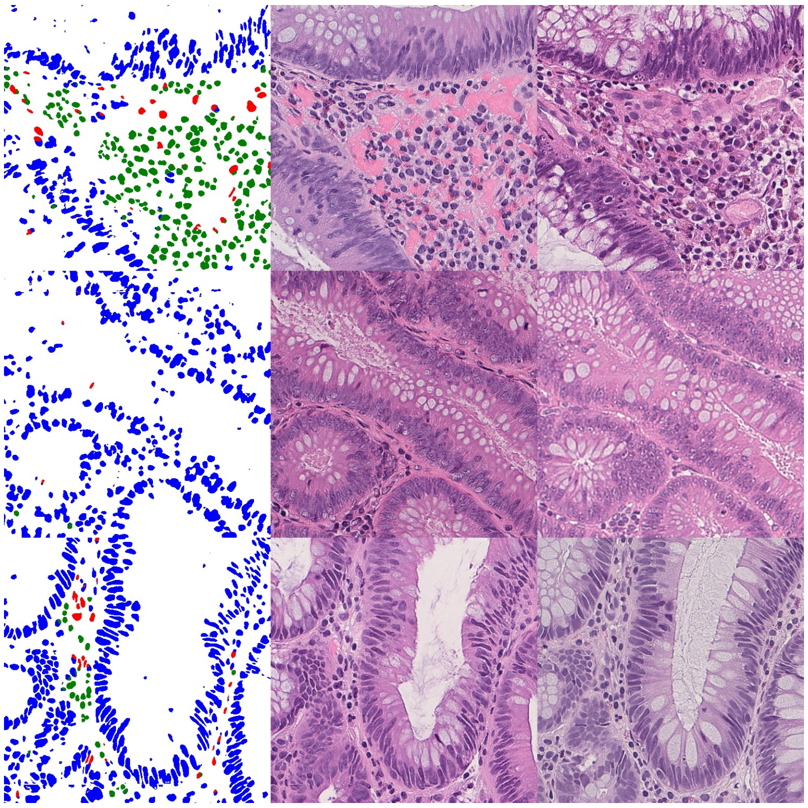


Figure A.1: Image-to-image generations at 512px resolution. Columns in order: segmented, generated, real.

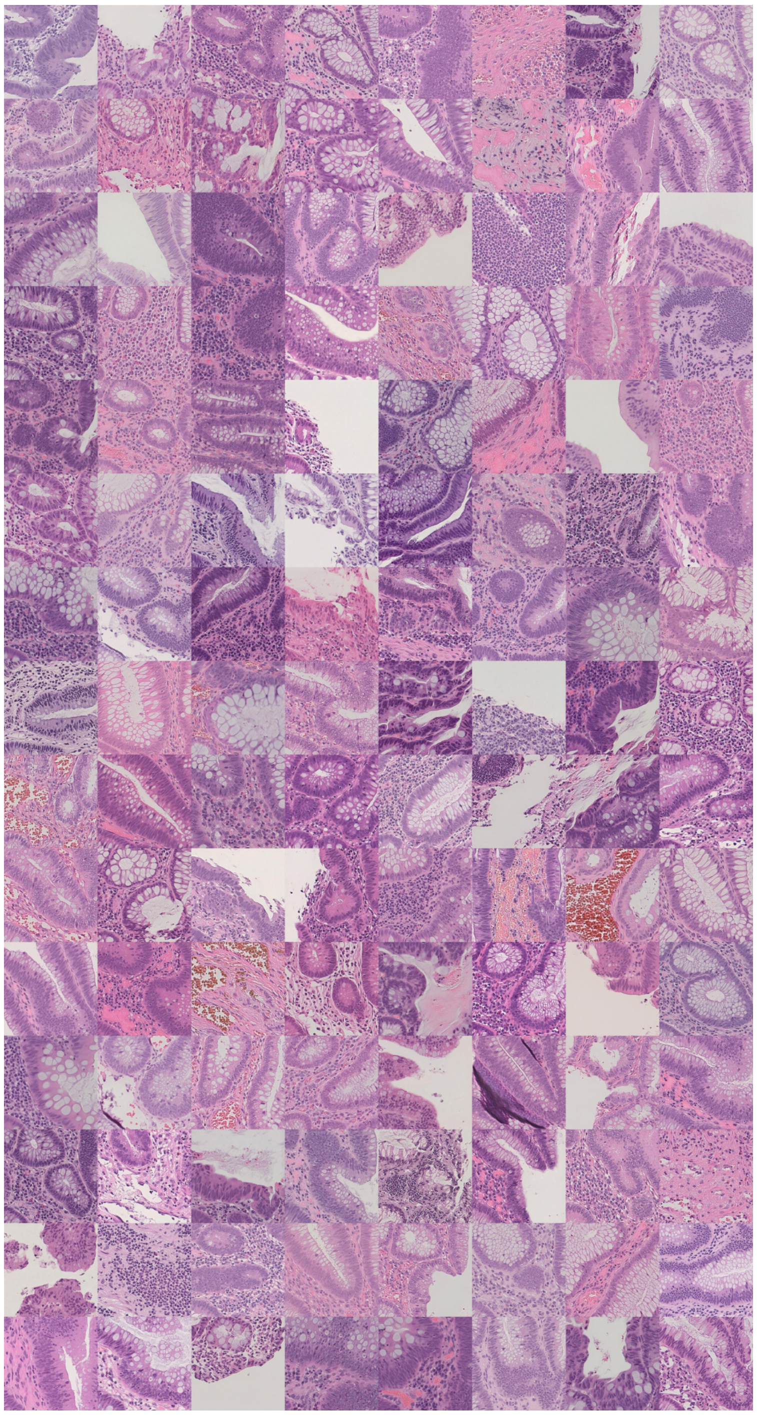


Figure A.2: Generations at 512px resolution

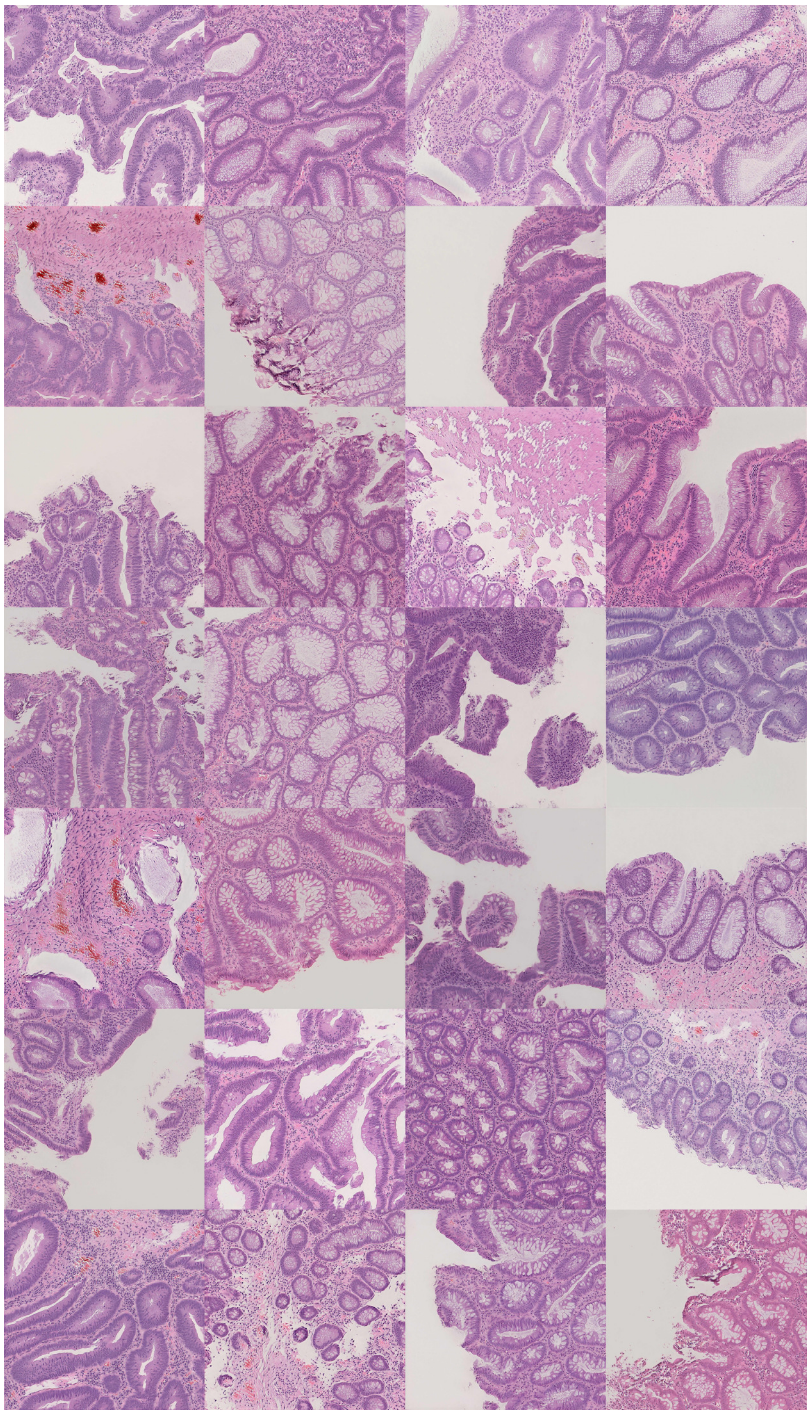


Figure A.3: Generations at 2048px resolution

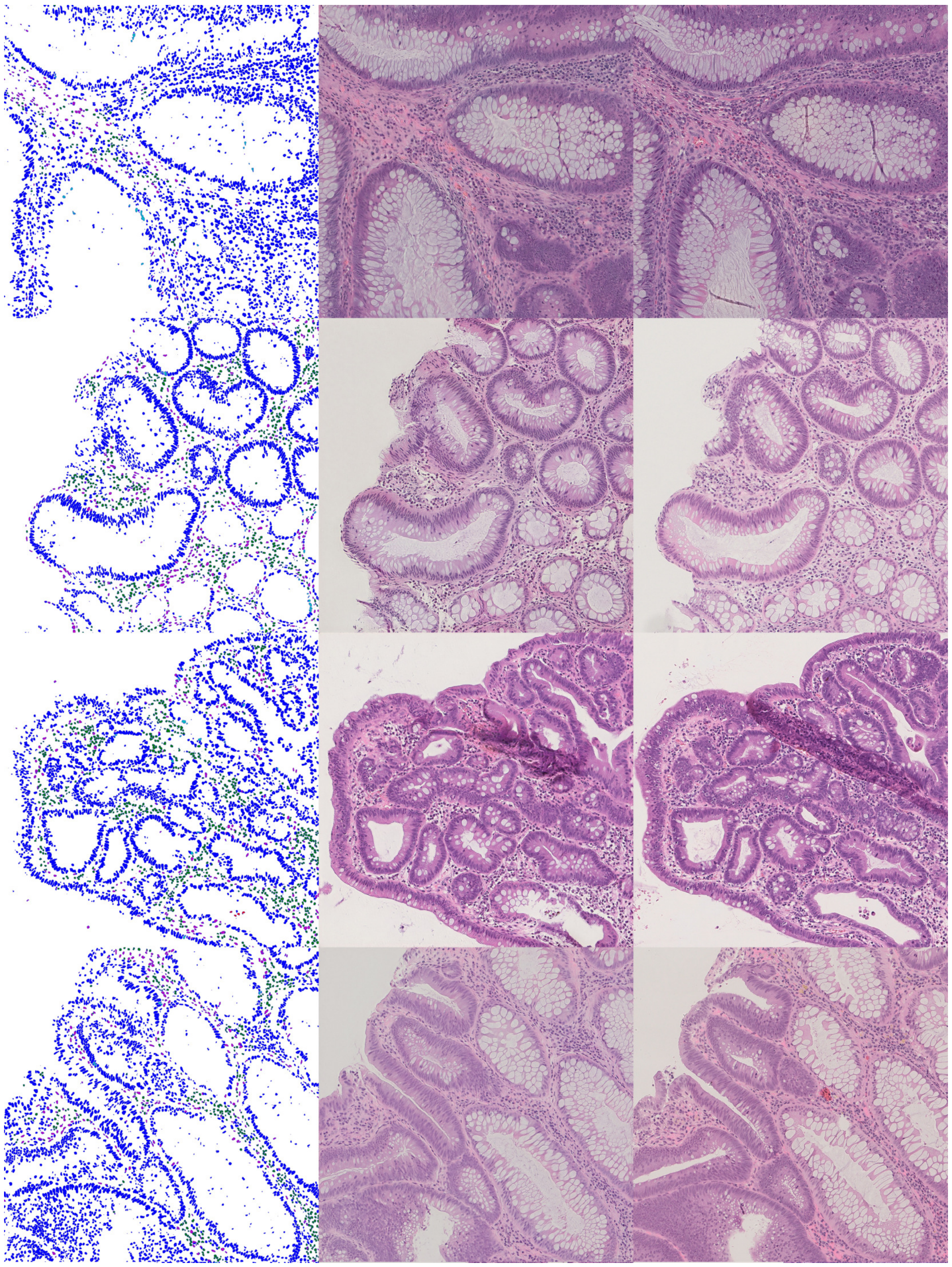
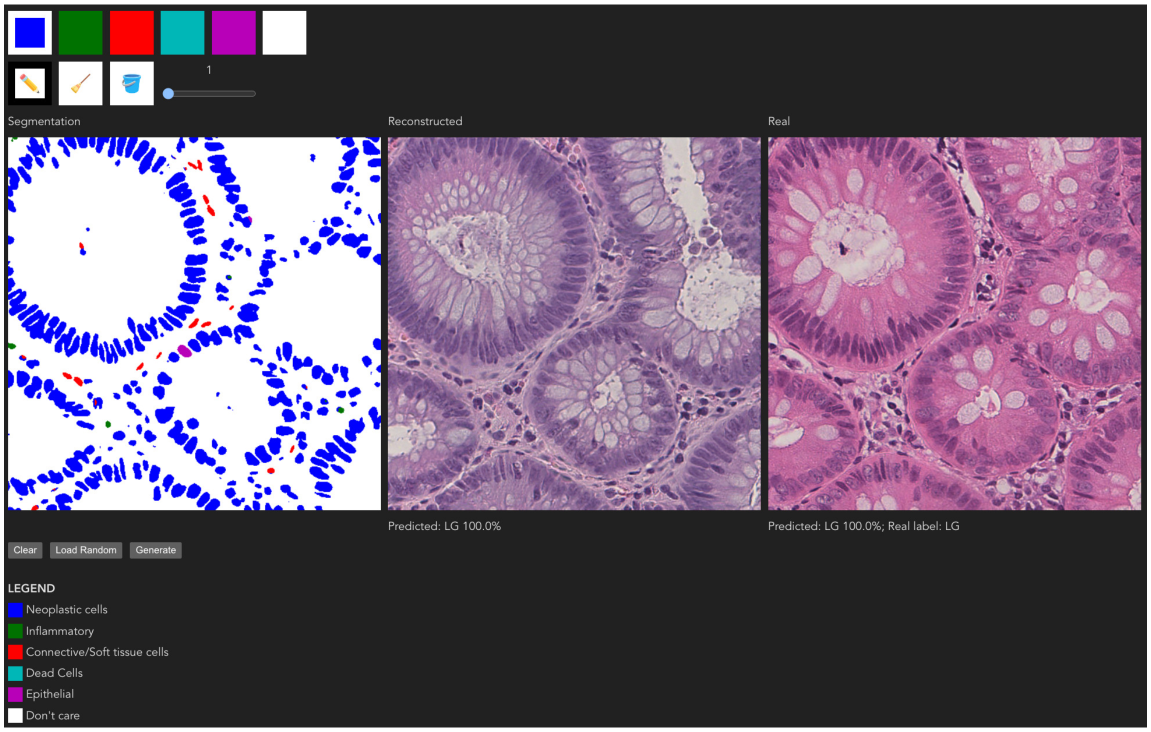


Figure A.4: Image-to-image generations at 2048px resolution. Columns in order: segmented, generated, real.



Predicted: LG 100.0%

Predicted: LG 100.0%; Real label: LG

LEGEND

🟦 Neoplastic cells
🟩 Inflammatory
🟥 Connective/Soft tissue cells
🟦 Dead Cells
🟪 Epithelial
⬜ Don't care

Figure A.5: The UI of our tool for image synthesis. Available at https://desi-ivanov.github.io/histopatho-drawer

Bibliography

[1] Alper Aksac, Douglas J. Demetrick, Tansel Ozyer, and Reda Alhajj. Brecahad: a dataset for breast cancer histopathological annotation and diagnosis. *BMC Research Notes*, 12(1):82, Feb 2019. ISSN 1756-0500. doi: 10.1186/s13104-019-4121-7. URL https://doi.org/10.1186/s13104-019-4121-7. 32

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017. URL https://arxiv.org/abs/1701.07875. 18

[3] Carlo Alberto Barbano, Daniele Perlo, Enzo Tartaglione, Attilio Fiandrotti, Luca Bertero, Paola Cassoni, and Marco Grangetto. Unitopatho, a labeled histopathological dataset for colorectal polyps classification and adenoma dysplasia grading, 2021. URL https://arxiv.org/abs/2101.09991. 2, 23, 26, 27, 28, 32, 37, 38

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax. 7

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165. 11

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maxi-

mizing generative adversarial nets, 2016. URL https://arxiv.org/abs/1606.03657. 2

[7] Hyungjoo Cho, Sungbin Lim, Gunho Choi, and title = Neural Stain-Style Transfer Learning using GAN for Histopathological Images publisher = arXiv year = 2017 copyright = arXiv.org perpetual, non-exclusive license Min, Hyunseok keywords = Computer Vision and Pattern Recognition (cs.CV), Artificial Intelligence (cs.AI), Machine Learning (cs.LG), FOS: Computer and information sciences, FOS: Computer and information sciences. URL https://arxiv.org/abs/1710.08543. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 22, 37

[9] Davide Di Luccio, Fiandrotti Attillo, Marco Grangetto, Carlo Alberto Barbano, and Enzo Tartaglione. Semantic segmentation of histopathological tissue with deep learning, 2021. 2, 24, 27, 28, 45

[10] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection, 2019. URL https://arxiv.org/abs/1906.11632. 38

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. URL https://arxiv.org/abs/2010.11929. 11

[12] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321: 321–331, dec 2018. doi: 10.1016/j.neucom.2018.09.013. 27

[13] Jevgenij Gamper, Navid Alemi Koohbanani, Ksenija Benet, Ali Khuram, and Nasir Rajpoot. Pannuke: an open pan-cancer histology dataset for nuclei instance segmentation and classification. In *European Congress on Digital Pathology*, pages 11–19. Springer, 2019. 24, 25

[14] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015. URL https://arxiv.org/abs/1508.06576. 19

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661. 1, 15, 16, 30

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017. URL https://arxiv.org/abs/1704.00028. 18

[17] Samir Gupta, David Lieberman, Joseph C Anderson, Carol A Burke, Jason A Dominitz, Tonya Kaltenbach, Douglas J Robertson, Aasma Shaukat, Sapna Syngal, and Douglas K Rex. Recommendations for Follow-Up after colonoscopy and polypectomy: A consensus update by the US Multi-Society task force on colorectal cancer. *Gastroenterology*, 158(4):1131–1153.e5, February 2020. 30

[18] Cesare Hassan, Giulio Antonelli, Jean-Marc Dumonceau, Jaroslaw Regula, Michael Bretthauer, Stanislas Chaussade, Evelien Dekker, Monika Ferlitsch, Antonio Gimeno-Garcia, Rodrigo Jover, Mette Kalager, Maria Pellisé, Christian Pox, Luigi Ricciardiello, Matthew Rutter, Lise Mørkved Helsingen, Arne Bleijenberg, Carlo Senore, Jeanin E van Hooft, Mario Dinis-Ribeiro, and Enrique Quintero. Post-polypectomy colonoscopy surveillance: European society of gastrointestinal endoscopy (ESGE) guideline - update 2020. *Endoscopy*, 52 (8):687–700, June 2020. 30

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385. 12, 13, 27, 42

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2017. doi: 10.48550/ARXIV.1706.08500. URL https://arxiv.org/abs/1706.08500. 22

[21] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208. 5

[22] Le Hou, Ayush Agarwal, Dimitris Samaras, Tahsin M. Kurc, Rajarsi R. Gupta, and Joel H. Saltz. Unsupervised histopathology image synthesis, 2017. URL https://arxiv.org/abs/1712.05021. 2, 26

[23] Bo Hu, Ye Tang, Eric I-Chao Chang, Yubo Fan, Maode Lai, and Yan Xu. Unsupervised learning for cell-level visual representation in histopathology images with generative adversarial networks. *IEEE Journal of Biomedical and Health Informatics*, 23(3):1316–1328, may 2019. doi: 10.1109/jbhi.2018.2852639. URL https://doi.org/10.1109%2Fjbhi.2018.2852639. 2

[24] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017. URL https://arxiv.org/abs/1703.06868. 20

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL https://arxiv.org/abs/1502.03167. 17

[26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2016. URL https://arxiv.org/abs/1611.07004. 18, 28, 44

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017. URL https://arxiv.org/abs/1710.10196. 2

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018. URL https://arxiv.org/abs/1812.04948. 3, 19, 20, 21

[29] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2019. URL https://arxiv.org/abs/1912.04958. 20, 30, 31, 32, 41, 43

[30] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020. URL https://arxiv.org/abs/2006.06676. 31

[31] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Proc. NeurIPS*, 2021. 31

[32] Salome Kazeminia, Christoph Baur, Arjan Kuijper, Bram van Ginneken, Nassir Navab, Shadi Albarqouni, and Anirban Mukhopadhyay. Gans for medical image analysis, 2018. URL https://arxiv.org/abs/1809.06222. 2, 26

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980. 31, 38, 44

[34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114. 14, 41

[35] Simon Kohl, David Bonekamp, Heinz-Peter Schlemmer, Kaneschka Yaqubi, Markus Hohenfellner, Boris Hadaschik, Jan-Philipp Radtke, and Klaus Maier-Hein. Adversarial networks for the detection of aggressive prostate cancer, 2017. URL https://arxiv.org/abs/1702.08014. 2

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 12

[37] Xin Mao, Zhaoyu Su, Pin Siang Tan, Jun Kang Chow, and Yu-Hsing Wang. Is discriminator a good feature extractor?, 2019. URL https://arxiv.org/abs/1912.00789. 38

[38] Takahisa Matsuda, Han-Mo Chiu, Yasushi Sano, Takahiro Fujii, Akiko Ono, and Yutaka Saito. Surveillance colonoscopy after endoscopic treatment for colorectal neoplasia: From the standpoint of the Asia-Pacific region. *Dig Endosc*, 28(3): 342–347, March 2016. 30

[39] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? 2018. doi: 10.48550/ARXIV.1801.04406. URL https://arxiv.org/abs/1801.04406. 18, 20

[40] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018. URL https://arxiv.org/abs/1802.05957. 18

[41] Natalia Norori, Qiyang Hu, Florence Marcelle Aellen, Francesca Dalia Faraci, and Athina Tzovara. Addressing bias in big data and AI for health care: A call for open science. *Patterns (N Y)*, 2(10):100347, October 2021. 1

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703. 7

[43] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. URL https://arxiv.org/abs/1511.06434. 17, 38

[44] Suman Ravuri and Oriol Vinyals. Classification accuracy score for conditional generative models, 2019. URL https://arxiv.org/abs/1905.10887. 37

[45] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014. URL https://arxiv.org/abs/1401.4082. 15, 42

[46] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 41

[47] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597. 15, 16, 43

[48] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958. 8

[49] Davide Rubinetti, Roberto Esposito, Marco Grangetto, Carlo Alberto Barbano, and Enzo Tartaglione. Generation of histopathological tissue with generative adversarial network, 2021. 2, 3, 28, 29, 46

[50] Matthew D Rutter, James East, Colin J Rees, Neil Cripps, James Docherty, Sunil Dolwani, Philip V Kaye, Kevin J Monahan, Marco R Novelli, Andrew Plumb, Brian P Saunders, Siwan Thomas-Gibson, Damian J M Tolan, Sophie Whyte, Stewart Bonnington, Alison Scope, Ruth Wong, Barbara Hibbert, John Marsh, Billie Moores, Amanda Cross, and Linda Sharp. British society of Gastroenterology/Association of coloproctology of great britain and Ireland/Public

health england post-polypectomy and post-colorectal cancer resection surveillance guidelines. *Gut*, 69(2):201–223, November 2019. 30

[51] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. URL https://arxiv.org/abs/1606.03498. 20

[52] Rebecca L Siegel, Kimberly D Miller, and Ahmedin Jemal. Cancer statistics, 2020. *CA Cancer J Clin*, 70(1):7–30, January 2020. 2

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL https://arxiv.org/abs/1409.1556. 12

[54] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer, 2022. URL https://arxiv.org/abs/2204.01697. 11

[55] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2016. URL https://arxiv.org/abs/1607.08022. 43

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762. 11

[57] Du Wang, Chaochen Gu, Kaijie Wu, and Xinping Guan. Adversarial neural networks for basal membrane segmentation of microinvasive cervix carcinoma in histopathology images. In *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 2, pages 385–389, 2017. doi: 10.1109/ICMLC.2017.8108952. 2

[58] Jelmer M. Wolterink, Konstantinos Kamnitsas, Christian Ledig, and Ivana Išgum. Generative adversarial networks and adversarial methods in biomedical image analysis, 2018. URL https://arxiv.org/abs/1810.10352. 2, 26

[59] Zhaoyang Xu, Xingru Huang, Carlos Fernández Moro, Béla Bozóky, and Qianni Zhang. Gan-based virtual re-staining: A promising solution for whole slide image analysis, 2019. URL https://arxiv.org/abs/1901.04059. 26

[60] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *Medical Image Analysis*, 58:101552, 2019. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2019.101552. URL https://www.sciencedirect.com/science/article/pii/S1361841518308430. 2, 26

[61] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, 2010. doi: 10.1109/CVPR.2010.5539957. 17

[62] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation, 2018. URL https://arxiv.org/abs/1807.10165. 28

[63] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017. URL https://arxiv.org/abs/1703.10593. 2, 26

[64] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation, 2017. URL https://arxiv.org/abs/1711.11586. 2