RADBOUD UNIVERSITY

Computing Science

---

**Mathematical Foundations of Computer Science**

# Diffusion Models And Their Use In Renal Histopathology

**Daily supervisor:**
Thomas de Bel

**Author:**
Mees Meuwissen

**First supervisor/assessor:**
Dr. Twan M. van Laarhoven

**In cooperation with:**
Aiosyn

**Second assessor:**
Dr. Johannes C. Textor

July 4, 2024

**Abstract**

In the field of digital pathology, obtaining suitable datasets for training robust models is a significant challenge. This thesis delves into generative modeling, specifically through the use of Latent Diffusion Models, to explore their potential in generating high-quality images of kidney tissue for use in downstream tasks such as segmentation model training. We train our own Latent Diffusion Model with histopathology images of kidney tissue and demonstrate that the trained model is capable of producing detailed images of specific renal structures using text-based guidance. We evaluate the effectiveness of our guidance system and quantitatively assess the realism of the generated images through a questionnaire with relevant experts. Additionally, we illustrate how human-annotated synthetic data can enhance real-world training datasets, improving performance of segmentation models by reducing incorrectly segmented areas by up to 34%.

# Acknowledgements

I would like to express my gratitude to Thomas de Bel, my daily supervisor, for his guidance, support and patience throughout my time at Aiosyn. Also, I would like to thank Twan van Laarhoven for his feedback and helpful comments throughout the thesis process. A special thank you goes out to all the wonderful people at Aiosyn, who granted me this opportunity and provided me with the perfect environment to write my thesis in, allowing me to stay motivated throughout. In particular, I want to thank Michael Deen and the AI team for being there for advice, friendly conversations, and even a table tennis match when needed.

# Contents

# 1. Introduction

Histopathology is the study of tissue specimens to diagnose and understand diseases at a microscopic level. By examining cellular structures and tissue architecture up close, pathologists can identify abnormalities, characterize diseases, and guide treatment decisions [1]. The field of *renal* histopathology focuses on analyzing kidney tissue samples and plays a major role in diagnosing several harmful conditions affecting the kidneys [2]. Traditionally, this diagnostic process relies on the expertise of pathologists, who interpret visual patterns and morphological changes through a microscope. In recent years, the use of Whole Slide Scanners has become prevalent for viewing histopathological images on computers. The digitization of the images has enabled the application of existing deep learning techniques, and has sparked interest in augmenting histopathological analysis with computational tools. These technologies offer the potential to automate repetitive tasks and uncover subtle patterns that may not be apparent to the human eye alone. However, the application of deep learning in histopathology is often constrained by the scarcity and variability of available annotated datasets. In order to make structures visible on a microscope, tissue undergoes *histological staining*. The most commonly used stain material is Hematoxylin and Eosyn (H&E), which stains the nuclei of cells a deep purplish blue, and the surrounding tissues various grades of pink [3]. This dual staining technique provides a contrast that allows pathologists to distinguish different tissue types and cellular structures with greater clarity, see Figure 1.1.

However, the process of histological staining is highly sensitive to variations in tissue preparation, staining protocols, and environmental conditions such as air humidity, water acidity and more. Even subtle differences in staining duration or dye concentration can significantly impact the appearance and interpretation of tissue samples under the microscope. Fur-

thermore, variations in scanner settings can introduce additional differences in datasets curated at different institutions [4]. Human pathologists are able to correct for these fluctuations, but they can significantly hinder Deep Learning based methods [5].

This sensitivity to variation means that available datasets are not always suitable for training new models, as the training data may not generalize well to the target data. Additionally, the difficulty of publishing datasets due to privacy constraints [6] further exacerbates the challenge of finding fitting training data. This situation can leave researchers and industry professionals with a limited pool of high-quality, standardized data for training robust and generalizable models.

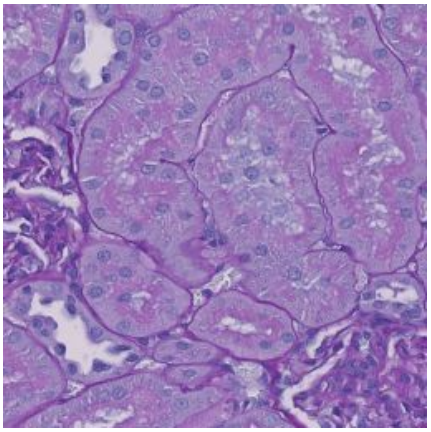


**Figure 1.1:** Microscopic image of stained kidney tissue.

This thesis explores the application of deep learning techniques to synthesize high-quality data for use in other downstream tasks [7]. Specifically, it investigates the feasibility of utilizing synthetic data generated by latent diffusion models [8] to supplement limited real-world datasets, aiming to improve the robustness and generalization of computer-aided diagnostic systems in renal pathology. Various models have been developed to improve diagnostics in histopathology, most of which focussed on discriminative tasks such as classification or segmentation [9]. Utilizing generative models in pathology is by comparison much underresearched. However, deep learning synthesis of data can help alleviate data scarcity as generated data is not bound by the same privacy regulations surrounding patient samples. Furthermore, generated images of rare subtypes of diseases can be

used in training of pathologists [10], providing another usecase for synthetic data.

This thesis was conducted at Aiosyn, an AI company working within the field of digital pathology. They allowed us access to their database of annotated kidney tissue, giving us the opportunity to explore the use of diffusion models within this specialization of pathology. To this end, we will develop a latent diffusion model and research its performance.

In Chapter 2, we give a brief overview of the existing research done in the area of image synthesis in general and in histopathology. Then, in Chapter 3, we explain the mathematics behind latent diffusion models in detail and derive an elegant formulation for optimization of them. Chapter 4 discusses the development of our model in particular, as well as outlines the methods used to gauge the performance of it. We go on to analyze the results in Chapter 5. Code used in the thesis is publicly available on our GitHub repository[1].

In this thesis, we explore whether synthetic data generated by diffusion models can be effectively utilized in renal histopathology to aid in training other models, such as segmentation model. Additionally, we assess if the images can play a role in training of pathologists, through the ability to create realistic images of specific structures on command.

[1] https://github.com/MeesMeuwissen/generationLDM

# 2. Related Work

Over the past few years, a lot of progress has been made in the development of generative models, with text-based models like OpenAIs ChatGPT [11] being among the most well-known. Generative Adverserial Networks (GAN), introduced by Goodfellow et al. in 2014 [12], were one of the first succesful models for various generative tasks, such as music synthesis [13] and image generation [14]. Levine et al. showed how GANs were capable of synthesizing multiple different types of cancer pathology images, proving their usefulnes to extend beyond generating natural images. [15]. Furthermore, Zhou et al. used a Unet based GAN for image synthesis in the context of histopathology to create a data augmentation method [16] to increase performance over a range of different tasks, including segmentation, detection and classification. However, GANs suffer from issues such as mode collapse, and are notoriously difficult to train [17][18], prompting researchers to continue looking alternative generative models.

Partly because of these factors, in natural image generation, GANs are already being replaced by different synthesis methods. This includes methods such as Flow Matching [19], but diffusion models [20] in particular are gaining a lot of attention, beating GANs on performance since 2021 [21]. Two of the most well-known image generation models with state-of-the-art performance are OpenAIs DALLE models [22] and Stability AIs Stable Diffusion [8]. More recently, diffusion models have been used to generate realistic video, as demonstrated by OpenAIs Sora, launched in early 2024 [23]. All of these models belong to a class of likelihood-based models that generate an image by gradually removing noise from an image (or another type of data) until only data remains (see Chapter 3 for the details of how a diffusion model works). Nichol and Dhariwal demonstrated that these models can scale remarkably well with model capacity and training compute, paving the way to new research in this area [24].

In the context of image synthesis in histopathology, several authors have used diffusion models for the generation of miscroscopic images of tissue, such as Moghadan et al. [25], who focussed on synthesizing images of brain cancer. Müller-Franzes et al. [26] demonstrated diffusion models can outperform GANs on medical imagery with their model, Medfusion. Furthermore, in [27], Harb et al. presented a technique using diffusion models to generate Whole Slide Images (WSIs) of tissue, which are gigapixel pathology images. To our knowledge, this has not been achieved using GANs.

Recently, Yellapragada et al. harnessed the power of Large Language Model GPT-3.5 [11] to create succinct and accurate descriptions of pathology images, which were subsequently used to accurately guide their model to create specific structures on command [28]. Their model, named PathLDM, used a *latent* diffusion model [8], meaning they created images in a compressed latent space before upscaling to pixel space. This technique allowed them to achieve state-of-the-art performance while reducing computational demand.

There exist notable differences between natural images and histopathology images. Natural images, capturing scenes of people and diverse environments, contain a broad spectrum of colors and textures inherent to their subjects. Objects can be lit in different manners, viewed from different angles and can vary in distance from the camera, to name a few. In contrast, histopathology images tend to look quite similar to each other, at least to the untrained eye. However, in this uniformity lies a critical challenge: these images contain intricate details crucial for accurate clinical assessment. Therefore, generative models tailored for histopathology images must be able to preserve and reproduce these minuscule structures with high accuracy. In our thesis, we will use PathLDM's trained network as a starting point, given its demonstrated effectiveness in this area.

# 3. Mathematical Background

In this chapter, we will explain how diffusion models function mathematically, as well as provide you with some insights into their implementation. We shall take a deep-dive into the mathematics behind it all, and is largely based on the excellent overview by Calvin Luo [29].

## 3.1 Generative Models

Generally speaking, the goal of a generative model is the following: Given observed data $x$, we want to learn to model the underlying true data distribution $p(x)$. Once we have a model for $p(x)$, we can *generate* new data by sampling from the distribution $p(x)$ which will closely match the observed data. One method of achieving this is through a diffusion model.

Oftentimes, we can think of the data we observe as generated or influenced by an associated, unseen, *latent* variable. For sake of clarity, let us denote the latent variable with $z$. A frequently used method to explain the concept of latent variables is that of Plato's Allegory of the Cave [30]. In the allegory, you are to imagine a group of people in a cave. They are chained to the wall, and their heads are secured so that they can only look forward, to the opposing wall. On that wall, shadows of various objects are visible. To a prisoner, having lived their entire life only seeing two-dimensional shadows, this is all that is real. They would never be able to comprehend that what they are seeing is simply a result of a higher-dimensional object being passed before a fire. In this allegory, the shadow is the data, and the object is the latent variable that produces the data. One caveat about this analogy: We generally try to learn *lower* dimensional latents in machine learning. This way we are able to learn representations that contain semantically meaningful and important properties. This can be seen as a form of compression, storing only the most defining properties of a datapoint.

In the same vein, objects, data, and everything else that we encounter in the world may be generated by some higher-dimension representations. For example, representations may carry information about the abstract properties of objects, like color, size, orientation and more. Our three-dimensional observation of an object can be interpreted as an instantiation of this latent variable, just as the shadow visible in the cave can be seen as an instantiation of the object in front of the fire, just in a lower dimension.

As an example of latent variables in machine learning, consider the images of handwritten digits in MNIST [31]. They are $28 \times 28$ pixels, meaning they exist in a 784-dimensional space. However, a lot of the information can be stored in only a few semantically meaningful variables. For example, perhaps we can use one variable to encode the digit, one for the slant, another for the stroke-width, and so on. With only a handful dimensions, one is able to accurately describe an image. By removing unnecessary information from data, machine learning models can focus on the most relevant features of a given datapoint.

## 3.2 Evidence Lower Bound

Mathematically, we can consider the latent variables $z$ and the data we observe $x$ to be modeled by a joint distribution $p(x, z)$. The approach diffusion models take, dubbed 'likelihood-based', is to learn a model to maximize the likelihood $p(x)$ over all observed $x$. Put differently, we have a dataset of observations, named $x$. We want to tune the parameters of a function $p$ in order to maximize the probability of observing the data, given the parameters of $p$.

There are two ways to manipulate the joint distribution to obtain a representation of the likelihood. The first is by explicitly *marginalizing* out the latent $z$:

$$p(x) = \int p(x, z) dz \tag{3.1}$$

The second method is by using the chain rule of probability:

$$p(x) = \frac{p(x,z)}{p(z|x)} \tag{3.2}$$

However, neither method is easy or even feasible. Marginalizing out $z$ involves integrating out all variables (so, dimensions) in $z$, which becomes intractable for even slightly complex models. Using Equation (3.2) requires use of a ground-truth latent encoder, $p(z|x)$. This function calculates the probability of a latent $z$ when given an observation $x$. This is referred to as the *posterior* of a distribution. This posterior is also not easy to obtain, as we will show later in this chapter. However, using the equations, it is possible to derive another equation, the **E**vidence **L**ower **Bo**und (ELBO), which is often used in practice. As the name suggests, this equation gives a lower bound of the evidence, which in this case is the log likelihood of the observed data.

The idea is as follows: If we optimize (increase as much as possible) the ELBO, we must also increase the evidence, and therefore the likelihood of the data. Formally, the equation for the ELBO is:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[ \log \frac{p(x,z)}{q_\phi(z|x)} \right] \tag{3.3}$$

Let us analyse the equation part by part: $q_\phi(z|x)$ is a flexible distribution with parameters $\phi$ that we wish to optimise. We aim for $q_\phi(z|x)$ to approximate $p(z|x)$. In the fully optimized case $q_\phi(z|x)$ becomes exactly equal to the true posterior $p(z|x)$. To achieve this, we adjust the parameters $\phi$ based on our observations $x$. To gain some intuition, you can think of $q$ being a neural network, and $\phi$ being its parameters. We train the network using data $x$. As we will see later in this chapter, optimizing the parameters to maximize the ELBO gives us access to components that can be used to model the true

data distribution. This in turn enables us to sample from it, effectively creating a generative model. For now, let us analyze why the ELBO is worth maximizing.

It is possible to derive the ELBO using Equation (3.1) and Jensen's Inequality [32], but it obscures a lot of the intuitive reasons as to *why* the ELBO is actually a lower bound as Jensen's Inequality does all the work for us. Additionally, simply knowing the ELBO is true does not tell us why we want to maximize it. A much more informative derivation starts at Equation (3.2). We make use of the fact that the integral over $q_\phi(z|x)$ equals 1 as it is a probability distribution.

$$
\begin{aligned}
\log p(x) &= \log p(x) \int (q_\phi(z|x) dz \\
&= \int q_\phi(z|x)(\log p(x)) dz \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log p(x)\right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p(x,z)}{p(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p(x,z) q_\phi(z|x)}{p(z|x) q_\phi(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p(x,z)}{q_\phi(z|x)}\right] + \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{q_\phi(z|x)}{p(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p(x,z)}{q_\phi(z|x)}\right] + D_{\mathrm{KL}}(q_\phi(z|x) \parallel p(z|x)) \\
&\geq \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p(x,z)}{q_\phi(z|x)}\right]
\end{aligned}
$$

In the final step, we used the fact that the Kullbeck-Leibler (KL) divergence (the term denoted as $D_{\mathrm{KL}}(\ldots \parallel \ldots)$ of two distributions is always $\geq 0$. This also makes clear the relationship between the evidence and the ELBO, and why optimizing the ELBO makes any sense at all: The KL Divergence term quantifies how similar the approximate posterior $q_\phi(z|x)$ and the true posterior $p(z|x)$ are. Ultimately, we want these two to match as

closely as possible and thus reduce their KL Divergence to zero. Sadly, we cannot directly compute the KL Divergence, since we do not have access to the ground truth $p(z|x)$ (If we did, we would not have to approximate it). However, we notice that the evidence ($\log p(x)$) does not depend in any way on the parameters of the approximate posterior, $\phi$, instead remaining constant. Therefore, since the ELBO term and the KL Divergence term sum up to this constant, any increase of the ELBO must necessarily be paired with an equal decrease of the KL Divergence term. Thus, optimization of the ELBO can be used as a proxy for minimizing the KL Divergence, which in turn optimizes our approximation of the true posterior distribution.

## 3.3 Variational Autoencoders

One Deep Learning construct that uses the ELBO is the Variational Autoencoder (VAE) [33]. In that setting, the ELBO is directly optimized, although it is often stated in a more dissected form. Let us explore this dissection here.

$$\begin{aligned}
\mathbb{E}_{q_\phi(z|x)}\left[\frac{p(x|z)}{q_\phi(z|x)}\right] &= \mathbb{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x,z)p(z)}{q_\phi(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log(p_\theta(x|z)\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log\frac{p(z)}{q_\phi(z|x)}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log(p_\theta(x|z)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q_\phi(z|x) \parallel p(z))}_{\text{prior matching term}} \qquad (3.4)
\end{aligned}$$

In the first step, we are able to split the joint distribution $p(x,z)$ into components $p_\theta(x|z)$ and prior $p(z)$. In essence, $p_\theta(x|z)$ can be considered the reverse of $q_\phi(z|x)$: autoencoders are trained to predict (i.e., reconstruct) input data from an intermediate *bottlenecked* form. $q_\phi(z|x)$ is the encoder, transforming observations $x$ to bottlenecked latents $z$, and $p_\theta(x|z)$ converts the latents back into observations (Section 3.3). These two models are trained simultaneously, optimizing both $\phi$ and $\theta$.

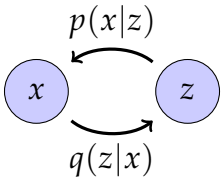



**Figure 3.1:** A graphical representation of a Variational Autoencoder. $q(z|x)$ encodes observations $x$ to latents $z$, and $p(x|z)$ decodes latents into observations.

Both terms in Equation (3.4) are intuitively interpretable: the first term measures a reconstruction loss, steering the parameters toward effective latents which allow for original data to be recoverd accurately. The second term, prior matching term, is used to guide the learned distribution toward a prior held belief over the latent variables. This encourages the encoder to learn a usable distribution (such as a normal distribution) enabling easy sampling from the latent space, possibly to generate more observations. Furthermore, fixing $p(z)$ to a known distribution allows us to calculate the KL Divergence term of the ELBO analytically.

Commonly, the encoder of the VAE is chosen to be a multivariate Gaussian with a diagonal covariance matrix, and the prior is a standard multivariate Gaussian. This partially fixed the distribution, but we can optimize the mean and exact standard deviation by adjusting parameters $\phi$:

$$q_\phi(z|x) = \mathcal{N}(z; \boldsymbol{\mu}_\phi(x), \boldsymbol{\sigma}^2_\phi(x)\mathbf{I})$$
$$p(z) = \mathcal{N}(z, \mathbf{0}, \mathbf{I})$$

The reconstruction loss can be approximated by using a Monte Carlo estimate (randomly selecting a few samples from the set of observations, and doing the calculations with that). Ordinarily, this would lead to issues, since latent $z \sim \mathcal{N}(z; \boldsymbol{\mu}_\phi(x), \boldsymbol{\sigma}^2_\phi(x)\mathbf{I})$ is stochastically generated, which is generally non-differentiable (meaning it won't be possible to update parameters $\phi$ by doing back-propagation). To overcome this difficulty, the *reparameterization trick* is employed. In this trick, samples from an arbitrary normal distri-

bution $x \sim \mathcal{N}(x; \mu, \sigma^2)$ with mean $\mu$ and standard deviation $\sigma$ are rewritten as:

$$x = \mu + \sigma\epsilon \quad \text{with } \epsilon \sim \mathcal{N}(\epsilon; 0, \mathrm{I})$$

That is, arbitrary Gaussian distributions are simply standard Gaussians that have their mean shifted by $\mu$, and their variance stretched by $\sigma^2$. Essentially, we can now rewrite a random variable as a deterministic function of an external noise variable $\epsilon$. After applying this trick, it is possible to determine gradients of the ELBO with respect to parameters $\phi$, optimizing $\boldsymbol{\mu}_\phi$ and $\boldsymbol{\sigma}_\phi$ as desired. Therefore, we can optimize the ELBO jointly over $\phi$ and $\theta$ by utilizing the reparameterization trick.

After training a VAE, it becomes possible to easily generate new data by sampling from the latent distribution $p(z)$ and running it through the optimized decoder $p_\theta(x|z)$. Usually, VAEs are employed with a latent space that is smaller than the observation space, as we then learn compact, useful representations of otherwise bulky data. Additionally, when we also learn a semantically meaningful latent space, latent vectors can be carefully crafted to include or exclude certain aspects of the generated data [34].

## 3.4 Hierarchical Variational Autoencoders

Now, let us generalize the concept of a VAE. A Hierarchical Variational Autoencoder introduces *hierarchies* for latent variables [35]. In essence, this captures the idea that latent variables themselves can be generated by more abstract, higher-level latent variables. Intuitively, if we regard our three-dimensional objects as objects generated by a higher-level latent, then the shadows the imprisoned people see can be interpreted as generated by a latent hierarchy of degree two (or more).

In general, in HVAEs, each latent is allowed to depend on all latents of higher levels. However, for diffusion models, it is only required to consider a special type of HVAE, namely a Markovian HVAE (MHVAE). In this case,

all latents are only allowed to condition on the latent exactly one level up, making the generative process a Markov Chain. That is to say, decoding latent $z_t$ only conditions on latent $z_{t+1}$, not on $z_{t+2}$ or others. Intuitively, this boils down to simply stacking VAEs on top of eachother, see Figure 3.2. Mathematically, we first need to introduce some new notation to represent the joint distribution $p(x, z_{1:T}$ and posterior $q_\phi(z_{1:T|x})$:

$$p(x, z_{1:T}) = p(z_T) \prod_{t=2}^{T} p_\theta(z_{t-1}|z_t) \tag{3.5}$$

$$q_\phi(z_{1:T}|x) = q_\phi(z_1|x) \prod_{t=2}^{T} q_\phi(z_t|z_{t-1}) \tag{3.6}$$

Here, we simply continuously roll out the term into its components, all with the appropriate conditioning information that makes it Markovian. Next, we can extend the ELBO as follows, this time using equation Equation (3.1) for brevity:

$$\begin{aligned}
\log p(x) &= \log \int p(x, z_{1:T}) dz_{1:T} && \text{(Apply equation Equation (3.1))} \\
&= \log \int \frac{p(x, z_{1:T}) q_\phi(z_{1:T}|x)}{q_\phi(z_{1:T}|x)} dz_{1:T} && \text{(Multiply by } 1 = \frac{q_\phi(z_{1:T}|x)}{q_\phi(z_{1:T}|x)}\text{)} \\
&= \log \mathbb{E}_{q_\phi(z_{1:T}|x)} \left[ \frac{p(x, z_{1:T})}{q_\phi(z_{1:T}|x)} \right] && \text{(Definition of Expectation)} \\
&\geq \mathbb{E}_{q_\phi(z_{1:T}|x)} \left[ \log \frac{p(x, z_{1:T})}{q_\phi(z_{1:T}|x)} \right] && \text{(Apply Jensen's Inequality)}
\end{aligned}$$

Note the similarity between this extended ELBO and Equation (3.3): The only difference is that we no longer talk about a single $z$, instead writing $z_{1:T}$ for the latents at the different levels. Later on, we will decompose this term further into interpretable parts.

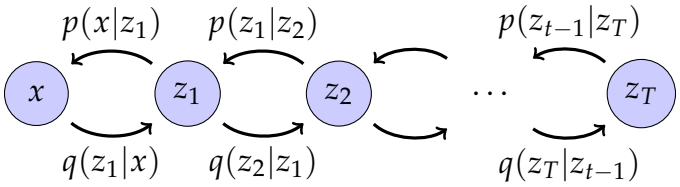



**Figure 3.2:** A graphical representation of a Markovian Hierarchical Variational Autoencoder with T hierarchical levels. Note that all decoding steps $p$ are conditioned only on the previous latent.

## 3.5 Diffusion Models

Now, we are ready to formally define our concept of a Diffusion Model (DM), also known as a Variational Diffusion Model (VDM). It is simply a Markovian HVAE with a couple of extra restrictions:

- The dimensionality of all latents $z_t$ is the same as the dimensionality of the data $x$.
- The structure of the latent encoders $q_\phi(z_1|x)$ and $q_\phi(z_t|z_{t-1})$ is not learned. Instead, it is predefined as a Gaussian distribution centered around the output of the previous timestep.
- The parameters of the encoding Gaussians vary in such a way that the final latent $z_T$ is distributed as the standard Gaussian, $\mathcal{N}(z_T, \mathbf{0}, \mathbf{I})$.

In our context of image generation, this simply means that we start with a completely noise-free image $x$, and progressively add noise to the image in $T$ steps, so that the final image $z_T$ is indistinguishable from pure noise. Note also that all latents $z_t$ remain in the dimension of the original image $x$, for example $256 \times 256$ pixels.

Because $z_t$ has the same dimension as $x$, and is simply a noisier version of it, we can abuse notation and write $x_t$ for both true data and latent samples. $x_0$ represents true data samples (i.e., unnoised) and $x_t$ with $t \in [1, T]$ a latent in hierarchy $t$. Using this notation and the fact we no longer learn the

encoders, the equation for the posterior (Equation (3.6)) simplifies:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{3.7}$$

Note the absence of parameter $\phi$ in the above equation.

From the second restriction, we know the latent encoders are centered around its previous latent. In fact, in our models, we will also fix the variance of the encoders, meaning they are completely determined before the model is trained. This is standard for many VDMs [20], although variants where the variance is also learned have been explored as well [36]. We set up our Gaussian encoders with $\boldsymbol{\mu}_t(x_t) = \sqrt{\alpha_t}x_{t-1}$, $\boldsymbol{\Sigma}_t(x_t) = (1-\alpha_t)\mathbf{I}$, where $\alpha_t$ are hyperparameters set beforehand. These hyperparameters determine *how fast* the images are turned into noise. Mathematically, this means we characterize the encoders as

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathbf{I})$$

The final restriction places an assumption on the final latent distribution $p(x_T)$, namely that it is a standard Gaussian. This updates the joint distribution to become

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

where $p(x_T) = \mathcal{N}(x_T; \mathbf{0}, \mathbf{I})$.

Together, these three assumptions lead to a process that can be described as progressively adding noise to a picture until it is pure noise. See Figure 3.3 for the process visualised.

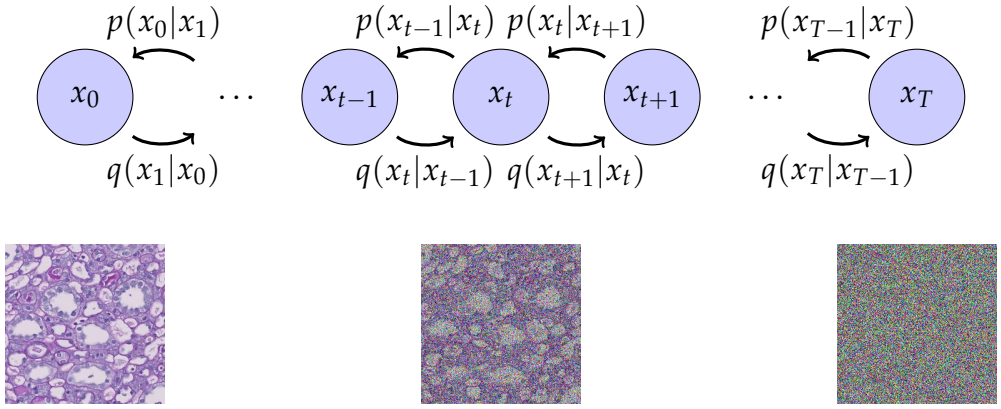



**Figure 3.3:** Variational Diffusion Model: $x_0$ is a noise-free image. Latents $x_t$ are progressively more noisy, with $x_T$ being complete Gaussian noise. $q(x_t|x_{t-1})$ are Gaussians centered around the previous output mean.

Since we no longer have parameters $\phi$ in the encoders $q$, the only parameters left to train are the parameters $\theta$ defining the approximate denoising transition $p_\theta(x_{t-1}|x_t)$. Once trained, these can be used to generate new data by simply sampling pure noise from $p(x_T)$, and running it through each of the denoising transitions.

Training the VDM is done by maximizing the ELBO, the derivation of which can be found in Derivation A.1

$$\begin{aligned}
\log p(x) &= \log \int p(x_{0:T}) dx_{0:T} \\
&= \dots \\
&= \underbrace{\mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0|x_1)\right]}_{\text{reconstruction term}} - \underbrace{\mathbb{E}_{q(x_{T-1}|x_0)}\left[D_{KL}(q(x_T|x_{T-1})\|p(x_T))\right]}_{\text{prior matching term}} \\
&\quad - \sum_{t=1}^{T-1} \underbrace{\mathbb{E}_{q(x_{t-1},x_{t+1}|x_0)}\left[D_{KL}(q(x_t|x_{t-1})\|p_\theta(x_t|x_{t+1}))\right]}_{\text{consistency term}}
\end{aligned} \tag{3.8}$$

In this form we can seperate the ELBO into three distinct parts:

- First, the reconstruction term $\mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0|x_1)\right]$. This can be interpreted as the log probability of having the original data sample $x_0$ given the first latent $x_1$. This term also appears in the optimization of

vanilla VAEs.

- The prior matching term $\mathbb{E}_{q(x_{T-1}|x_0)}\left[D_{KL}(q(x_T|x_{T-1})\|p(x_T))\right]$: This term is subtracted, which means maximizing the ELBO requires this term to me minimal. It is minimized when $q(x_T|x_{t-1})$ exactly matches $p(x_T)$ (so, a standard Gaussian). Fortunately, this depends on our choice of $\alpha$, allowing us to ensure this condition is met. Furthermore, this term has no parameters and can thus not be optimized *during* the training process.
- The consistency terms $\mathbb{E}_{q(x_{t-1},x_{t+1}|x_0)}\left[D_{KL}(q(x_t|x_{t-1})\|p_\theta(x_t|x_{t+1}))\right]$. Like the prior matching term, this is made up from a KL Divergence term, this time between encoders $q$ and decoders $p_\theta$. Its purpose is to make the distribution at $x_t$ consistent, no matter if you arrive there by adding noise to ($q(x_t|x_{t-1})$) or removing noise from ($p_\theta(x_t|x_{t+1})$) a neighboring image.

Optimizing the ELBO mainly boils down to optimizing the consistency terms, since there are so many of them.

In our derivation, all terms are expressed as expected values, which makes it perfect to optimize by once again using Monte Carlo estimates. However, note that the consistency term depends on two random variables: $x_{t-1}$ *and* $x_{t+1}$. This has the effect that it might exhibit quite high variance, leading to poorer optimization speeds. Instead, it is possible to derive a form of the ELBO in which each term only depends on a single random variable. The key to this derivation is the following insight: Instead of adding a bit of noise to an image $x_{t-1}$, we can equivalently *remove* a bit of noise from $x_{t+1}$, *if* we know what the denoised image $x_0$ looks like. Mathematically speaking, we simply add $x_0$ as an extra conditioning term to $q(x_t|x_{t-1})$: it then becomes $q(x_t|x_{t-1}, x_0)$. Then, using Bayes rule, we can do the following rewriting:

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

Making use of this equation, the dissection of the ELBO then results in the

following (Full derivation found in Derivation A.2)

$$\begin{aligned}\log p(x) &\geq \dots \\ &= \underbrace{\mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0|x_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{KL}(q(x_T|x_0) \parallel p(x_T))}_{\text{prior matching term}} \\ &\quad - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))\right]}_{\text{denoising matching term}} \end{aligned} \tag{3.9}$$

- Here, the reconstruction term remains the same as in Equation (3.8).
- The prior matching term features updated conditioning for the encoding function $q$. But, as previously discussed, it has no trainable parameters and is set to zero under our restrictions.
- The consistency term does meaningfully change here however: It becomes a *denoising matching term*. Note that in the KL Divergence term, the encoder is now written as $q(x_{t-1}|x_t, x_0)$, accomplishing the desired effect of *removing* a slight bit of noise from $x_t$ to arrive at $x_{t-1}$. In this sense, it acts as a ground truth *denoising* function, one which $p_\theta(x_{t-1}|x_t)$ is trying to learn. Crucially, it now conditions on the same random variable $x_t$ that our approximate denoising step $p_\theta(x_{t-1}|x_t)$ conditions on. $q$ also conditions on $x_0$, but that is not a random variable, so it does not increase the variance of our Monte Carlo estimates of the ELBO. An overview of this process can be found in Figure 3.4, where the goal is to tune parameters $\theta$ such that the red arrow matches the teal one.

Optimizing this derivation of the ELBO mainly comes down to optimizing the denoising matching terms, since they are so numerous, especially since models typically have large $T$. For example, in our models, we use $T = 1000$ timesteps.

In order to make calculation of the KL Divergence term tractable, we can

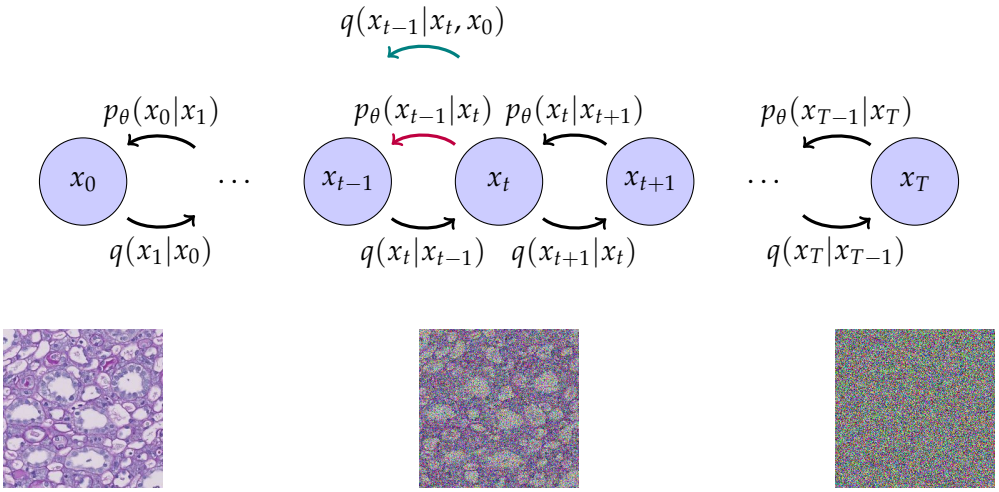



**Figure 3.4:** In order to train a VDM with lower variance, we compute a ground-truth denoising step in the form of $q(x_{t-1}|x_t, x_0)$ using Bayes rule. We use KL Divergence to minimise the difference between it and our approximate denoising step $p_\theta(x_{t-1}|x_t)$.

do some rewriting. Bayes rule gives us the following:

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \tag{3.10}$$

Paired with the fact that (by restriction)
$q(x_t|x_{t-1}, x_0) = q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathbf{I})$, we must only derive tractable forms for $q(x_t|x_0)$ and $q(x_{t-1}|x_0)$. However, the knowledge that the encoder transitions are linear Gaussians helps us here: Recall the reparameterization trick, and apply it to $x_t \sim q(x_t|x_0)$. It can be rewritten in terms of $x_{t-1}$ as

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon \text{ with } \epsilon \sim \mathcal{N}(\epsilon; \mathbf{0}, \mathbf{I})$$

Similarly, $x_{t-1}$ can be rewritten in terms of $x_{t-2}$ as

$$x_{t-1} = \sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon \text{ with } \epsilon \sim \mathcal{N}(\epsilon; \mathbf{0}, \mathbf{I})$$

which can be substituted in the formula for $x_t$. Continuing this way, it is possible to write $q(x_t|x_0)$ neatly through continuous application of the reparameterization trick. In fact, we can can rewrite (Full derivation available in Derivation A.3) a sample $x_t \sim q(x_t|x_0)$ as:

$$\begin{aligned} x_t &= \dots \\ &= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0 \sim \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}) \end{aligned} \tag{3.11}$$

where $\bar{\alpha}_t$ is defined as the product of all $\alpha$'s before it: $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

In a similar way, we derive the equation describing $q(x_{t-1}|x_0)$. Armed with these equations, we can derive a formula for $q(x_{t-1}|x_t, x_0)$, continuing from Equation (3.10):

$$\begin{aligned} q(x_{t-1}|x_t, x_0) &= \frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)} \\ &= \frac{\mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathbf{I})\mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}x_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})} \\ &= \dots \\ &\propto \mathcal{N}(x_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t}}_{\mu_q(x_t, x_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\Sigma_q(t)}) \end{aligned} \tag{3.12}$$

Most crucially, with this derivation we have shown that for any timestep $t$, $x_{t-1}$ is normally distributed with mean $\mu_q(x_t, x_0)$ (thus dependent on $x_0$) and variance $\Sigma_q(t)$ (dependent only on time $t$ and hyperparameters $\alpha$). The coefficients $\alpha$ are fixed beforehand, allowing us easy calculation of the variances at each timestep.

Recall that our goal is to learn a model $p_\theta(x_{t-1}|x_t)$ to match the ground truth denoising step $q(x_{t-1}|x_t, x_0)$ as closely as possible. Therefore, it makes

sense to model $p_\theta$ as a Gaussian, reflecting the ground truth. Furthermore, we have fixed all coefficients $\alpha$ beforehand, meaning we can calculate the appropriate variance of our predicted Gaussian, and can focus on predicting the mean of each transition. We denote it by $\mu_\theta(x_t, t)$. Note how $\mu_\theta$ parameterizes only on $x_t$ and $t$, and not on $x_0$, since $p_\theta(x_{t-1}|x_t)$ does not condition on $x_0$. For brevity, we will simply write $\mu_q$ for the mean of $q(x_{t-1}|x_t, x_0)$ and $\mu_\theta$ for the mean of $p_\theta(x_{t-1}|x_t)$.

Mathematically, optimizing the function to match as closely as possible is done with the KL Divergence. In an equation, we want to find the $\theta$ that minimizes their divergence:

$$\arg\min_\theta D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \tag{3.13}$$

$$= \arg\min_\theta D_{\mathrm{KL}}(\mathcal{N}(x_{t-1}; \mu_q, \Sigma_q(t) \parallel \mathcal{N}(x_{t-1}; \mu_\theta, \Sigma_q(t)) \tag{3.14}$$

Since the variances of the two distributions match exactly, the KL Divergence should be optimized precisely when the means $\mu_q$ and $\mu_\theta$ match. In fact, if we write $\Sigma_q(t) = \sigma_q^2(t)\mathbf{I}$, we can write out the definition of KL Divergence. Doing so leads us to the following equation (Derivation A.4):

$$\begin{aligned}
&\arg\min_\theta D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \\
&= \dots \\
&= \arg\min_\theta \frac{1}{2\sigma_q^2(t)} \left[ \left\| \mu_\theta - \mu_q \right\|_2^2 \right]
\end{aligned}$$

This makes precise our intuition of having to match the means of the two

distributions. Recall from Equation (3.12) the definition of $\mu_q$:

$$\mu_q(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t}$$

As for $\mu_\theta$, only the $x_0$ term is unknown to it. Thus, if we parameterize a neural network to predict it from $x_t$ and $t$, we can match it closely to $\mu_q$ by setting

$$\mu_\theta(x_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{x}_\theta(x_t, t)}{1-\bar{\alpha}_t}$$

where $\hat{x}_\theta(x_t, t)$ is the neural networks prediction of $x_0$. Plugging these representations for $\mu_q$ and $\mu_\theta$ into Equation (3.14) and rewriting (Derivation A.5), the problem simplifies to:

$$\begin{aligned}
&\arg\min_\theta D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \\
&= \dots \\
&= \arg\min_\theta \frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2)}{(1-\bar{\alpha}_t)^2} \left[\|\hat{x}_\theta(x_t, t) - x_0\|_2^2\right]
\end{aligned}$$

The constants in front of the norm are not dependent on $\theta$ and thus do not influence the optimal choice of $\theta$. Therefore, optimizing the diffusion model boils down to an elegant essence: train a neural network to predict original, noise-free images from arbitrarily noised versions of it. Doing this well allows us to effectively learn how to perform a single denoising step $x_t$ to $x_{t-1}$, as we will be able to accurately predict the normal distribution determining $x_{t-1}$. In order to decrease *all* the denoising matching terms from Equation (3.9), we must train our network for all $t \in \{1, \dots, T\}$ equally. This is achieved by randomly sampling $t$ uniformly during training.

So, to recap the whole learning process: We have a dataset of data (images) from which we would like to sample more, similar data. To do this,

we use a diffusion model. In this model, we add noise to our data in many (in our case 1000) controlled steps, creating intermediate data distributions. By carefully controlling the manner in which the noise is added, we ensure the distributions are all Gaussian. Additionally, after all noising steps are completed, we are left with pure Gaussian noise, which is easy to sample from.

We then train a model to reverse this process: In the trained model, we start with pure Gaussian noise ($x_T$), and repeatedly remove noise from the image by calculating the means of the intermediate distributions ($\mu_{T-1}$, $\mu_{T-2}, \ldots$) which are then used to sample intermediate results from ($x_{T-1}$, $x_{T-2}, \ldots$). This must be done sequentially: In order to calculate $\mu_{T-2}$, we need access to sample $x_{T-1}$, which can only be obtained using $\mu_{T-1}$, and so on. After going through all denoising steps, we arrive at $x_0$, which is a noise-free generated datapoint, hopefully similar to true data.

Obviously, the more accurate the intermediate distributions are, the better our generated data. The variances are fixed through hyperparameters $\alpha$, and the means $\mu_t$ are approximated. This is done by having a neural network predict $x_0$ from $x_t$, denoted $\hat{x}_\theta(x_t, t)$ which is used to calculate an approximate mean $\mu_\theta$.

Optimization of the process is captured by optimizing the ELBO. In practice, this is done by randomly choosing a time $t$ and a datapoint $x_0$. Noisy $x_t$ is calculated using $\bar{\alpha}_t$, which is a fixed hyperparameter. Then, the model predicts $\hat{x}_\theta$, which is compared to $x_0$, updating parameters $\theta$ in the process.

### 3.5.1 Predicting ground truth noise

In our previous approach, we train a neural network to predict $x_0$, the noise-free image. However, in practice, it turns out that predicting *the noise that was added to* $x_0$ works better [20][37]. Let us derive this alternative yet equivalent interpretation.

Recall that $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon_0$ in Equation (3.11). We can rewrite

this equation to obtain a formula for $x_0$ in terms of $x_t$ and $\epsilon_0$:

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}_t}}$$

Using this equation in our formula for the mean $\mu_q(x_t, x_0)$ we arrive (Derivation A.6) at the following equation:

$$\begin{aligned}\mu_q(x_t, x_0) &= \dots \\ &= \frac{1}{\sqrt{\alpha}_t} x_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha}_t}\epsilon_0\end{aligned}$$

Therefore, we can calculate our approximate mean $\mu_\theta(x_t, x_0)$ by:

$$\mu_\theta(x_t, x_0) = \frac{1}{\sqrt{\alpha}_t} x_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha}_t}\hat{\epsilon}_\theta(x_t, t)$$

Then, Equation (3.14) becomes

$$\begin{aligned}&\arg\min_\theta D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \\ &= \dots \\ &= \arg\min_\theta \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \left[\|\epsilon_0 - \hat{\epsilon}_\theta(x_t, t)\|_2^2\right]\end{aligned} \tag{3.15}$$

In this equation, $\hat{\epsilon}_\theta(x_t, t)$ is the output of a neural network with parameters $\theta$. Once again, we are not interested in the constants in front of the norm. Thus, the neural net can be optimized in the same way as the network for $\hat{x}_\theta(x_t, t)$, i.e. via Monte Carlo optimization.

### 3.5.2 Guided Diffusion

So far, we have outlined a method of generating samples that look like they belong to a distribution dataset of observations $x$. However, for many applications, it is valuable to have more control over the generated data by using conditioning information $y$. For example, $y$ can be a class label or a description of the generated data, as used in popular available diffusion models such as DALLE-2 [38] and Imagen [37].

The easiest way to turn a DM into a conditional DM is to simply add the conditioning information $y$ to the denoising conditioning information: Instead of trying to learn $p_\theta(x_{t-1}|x_t)$, we learn a transition $p_\theta(x_{t-1}|x_t, y)$. This translates into learning a neural network to predict or $\hat{x}_\theta(x_t, t, y) \approx x_0$ or $\hat{\epsilon}_\theta(x_t, t, y) \approx \epsilon_0$. In a sense, time $t$ can be viewed as conditioning information just as $y$, only $t$ is less complex. In our case, we are trying to generate images, and the conditioning information is an embedding of a textual description of the images content.

Furthermore, we make use of a technique called *Classifier Free Guidance* [39]. In this strategy, we run the model twice when sampling: once *with* conditioning information, once *without*. A parameter $\gamma$, called the guidance scale, controls the importance of the conditioning. If $\gamma = 0$, the model completely ignores the output of the conditional model call, and the model behaves like a unconditional diffusion model. If $\gamma = 1$, the unconditional output is ignored. Furthermore, $\gamma$ values greater than 1 are possible. In this situation, the model not only prioritizes the conditional output, it even moves *away* from the unconditional output, in the direction of the conditional output. Basically, it decreases the probability of creating output that ignores the conditioning information. We make use of this technique when sampling to make sure the conditioning information is taken into account.

### 3.5.3 Latent Diffusion Model

We now have a conceptual understanding of how diffusion models work mathematically. In practice, they turn out to be quite cumbersome and heavy to train, particularly for large images. This leads us to consider train-

ing a diffusion model in a smaller, more compressed latent space of a pretrained autoencoder. This enables training using limited resources while retaining high perceptual quality, and it has been used to deliver state-of-the-art image synthesis tools [8].

Training a diffusion model in a latent space requires the use of an encoder/decoder pair, known as the *autoencoder*. It can be trained seperately using conventional methods [40], resulting in an encoder $\mathcal{E}$ and decoder $\mathcal{D}$. Essentially, we have a dataset $x$ of images for which we want to create more synthetic samples. In order to do so, we train our diffusion model on images in the latent space of the autoencoder. That means to obtain training data for the DM, we pass real, pixel-space images through the encoder $\mathcal{E}$ to obtain training samples in the latent space. Subsequently, they are used to train the DM (Figure 3.6), which is carried out as explained in the section before.

When obtaining new samples, we sample noise in the latent space, run it through the diffusion model, and pass the resulting image through the decoder $\mathcal{D}$ (Figure 3.7).

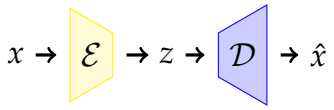



**Figure 3.5:** Functioning of the autoencoder. Data $x$ is passed through encoder $\mathcal{E}$ to obtain a compressed, lower-dimensional representation $z$. Then, this $z$ is passed through decoder $\mathcal{D}$ in order to reconstruct $\hat{x}$, which should approximate $x$.

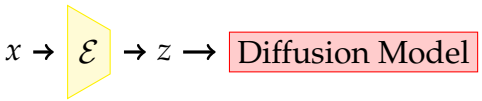



**Figure 3.6:** Training the diffusion model. Images in pixel space are compressed by the pretrained encoder, and the latent representation $z$ is used as training input for the diffusion model.

Summarising, we have defined Variatonal Diffusion Models as a special case of a Hierarchical Autoencoder using three key restrictions. With these restrictions, optimization becomes tractable and scalable, in a surprisingly elegant way. Then, we extended our definition to be able to handle outside

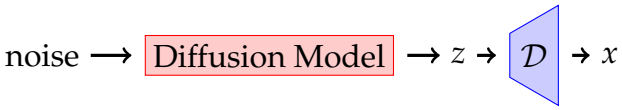



**Figure 3.7:** Sampling from the diffusion model. A latent sample $z$ is generated by the diffusion model from latent noise. Since output $z$ is in latent space, we must decode it with decoder $\mathcal{D}$ to obtain a regular sample in full pixel space.

conditioning information such as class labels or textual descriptions. Lastly, we explained how to improve computational performance by deploying the diffusion model in a latent space of a pretrained autoencoder.

# 4. Methodology

In this chapter, we will outline how we developed our Latent Diffusion Model, as well as list several different methods of evaluating the performance of it. The results of the tests will be detailed in Chapter 5.

## 4.1 Model Development

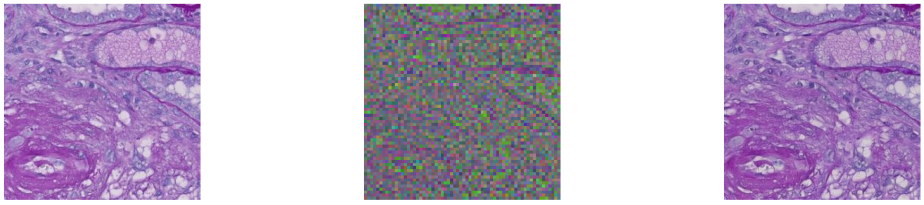
We built our model using the infrastructure used by PathLDM [28], which in turn builds on a code-base built by Rombach et al. at Computer Vision and Learning LMU Munich, called latent-diffusion [8]. Their code is available on GitHub [1], as are PathLDMs version [2] and our fork [3].

### 4.1.1 Autoencoder

When developing a Latent Diffusion Model, the choice of autoencoder is crucial because it is an integral component of the overall model. Autoencoders typically reduce the dimensionality of their input by a fixed amount. In this process, we need to balance the trade-off between the autoencoder's ability to accurately reconstruct the input and the computational efficiency of the diffusion model, which operates on the latent space data. Since histopathological images contain fine details and structures that must be preserved, we should be careful when reducing their dimensionality. To this end, we employ the pretrained VAE used by PathLDM[4], which uses 3 output channels and a downsampling factor of 4. That is to say, input images are scaled down by 4 when passing through the encoder, meaning we are left with $\frac{1}{16}^{th}$ of the pixels after encoding. The authors found that this downsampling fac-

[1] https://github.com/CompVis/latent-diffusion
[2] https://github.com/cvlab-stonybrook/PathLDM
[3] https://github.com/MeesMeuwissen/generationLDM
[4] https://drive.google.com/drive/folders/1_urgkNKIMmFoATiRtDwgjGuwjEt_fdvG

tor manages to preserve image detail, while not needlessly slowing down training of the LDM. The VAE is obtained by finetuning VQ-f4 from Rombach et al. on breast tissue samples from the well-studied Cancer Genome Atlas Breast Invasive Carcinoma (TCGA-BRCA) dataset [41] [42]. In our thesis, we treat the encoder and decoder as black-box components of the model and do not further train them. The functioning of the VAE is illustrated in Figure 4.1, where we can see what an image in the latent space of the VAE looks like. In Figure 4.1b, the image looks quite noisy to the naked eye, but this is a result of the compression by encoder $\mathcal{E}$, and contains all the relevant information to reconstruct the image, as can be seen by comparing the reconstructed image with the input.



**(a)** Unaltered input image $x$.

**(b)** Encoded image $\mathcal{E}(x)$.

**(c)** Reconstructed image $\mathcal{D}(\mathcal{E}(x)) \approx x$.

**Figure 4.1:** Functioning of the Variational Autoencoder. Note that the encoded image is $64 \times 64$ pixels, while the other two are $256 \times 256$ pixels, highlighting the compressive function of the encoder.

### 4.1.2 Denoising Unet

As the backbone of the DM, we use a Unet denoiser designed by Rombach et al., which is a Unet [43] with some additional structure to incorporate a method to guide the diffusion process. This is achieved by integrating a cross-attention mechanism [44] in all layers of the Unet, ensuring proper guidance. See Figure 4.2 for an overview. In our primary training run, we do not train the Unet from scratch. Instead, we take as a starting point the trained Unet from PathLDM[5] which is capable of producing high-quality breast tissue images. This has several advantages: First, it speeds up training time since the model is already proficient in removing noise from images

[5]https://drive.google.com/drive/folders/1_urgkNKIMmFoATiRtDwgjGuwjEt_fdvG

and does not have to learn this. Second, the model will already be capable of dealing with the fine detail found in histopathology imageres, as opposed to a Unet pretrained on the data from Imagenet, which is one of the models provided by Rombach et al.. However, experiments have shown that starting at that model checkpoint yields comparative results, at least to the naked eye, as can be seen in Figure 4.3. In the image, samples from a partially trained model are shown, and no discernable difference can be easily spotted. Nevertheless, because of the theoretical performance increase outlined earlier, we conduct all our experiments using a model initialized with the PathLDM Unet, capable of producing high-quality images of tissue right of the bat.

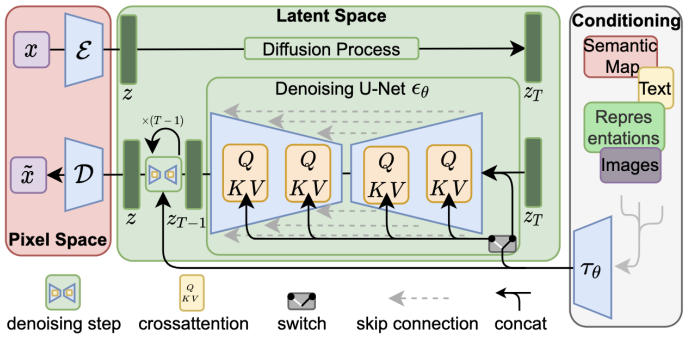
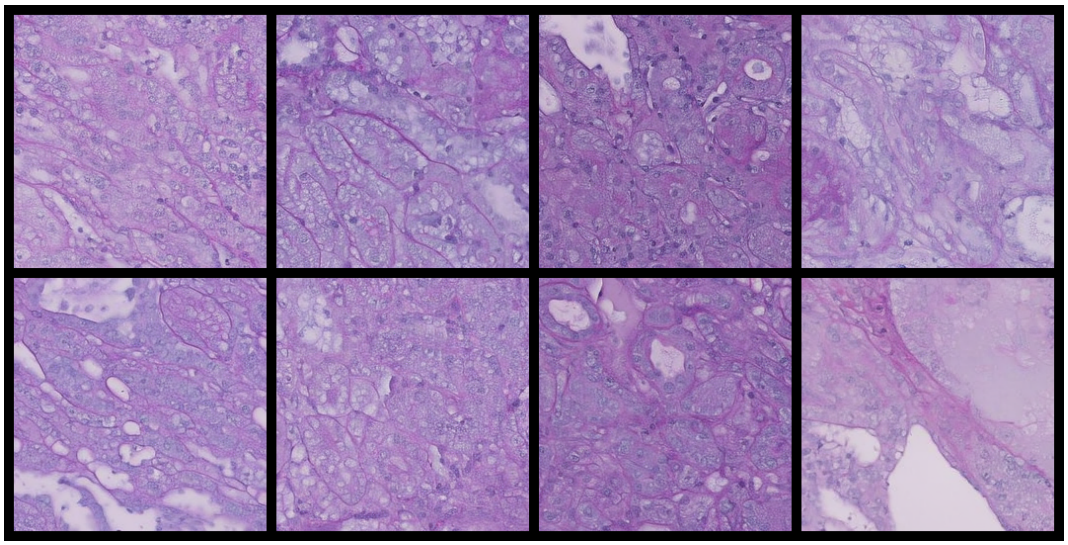


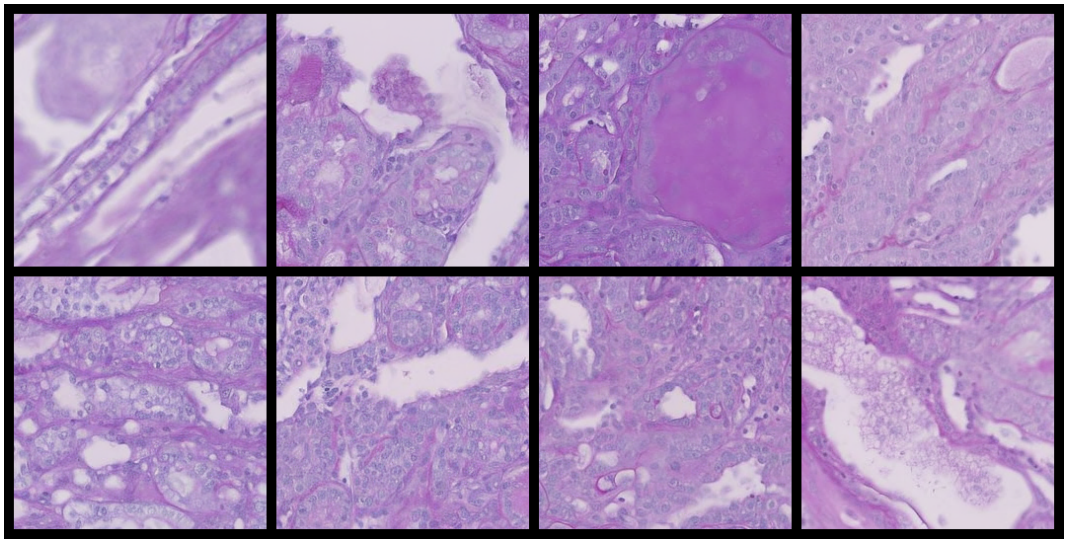

**Figure 4.2:** Structure of our Latent Diffusion Model. Source: [8].

### 4.1.3 Data

We trained our Latent Diffusion model using an internal custom dataset consisting of 100 000 patches, each measuring 256x256 pixels. This dataset was derived from a whole slide image (WSI) of rat kidney tissue. The decision to use rat kidney resections rather than human kidney biopsies stems from the availability of WSIs at Aiosyn: Resections provide a significant advantage due to their larger size, ensuring nearly all patches are entirely filled with tissue with hardly any patches of edges between tissue and background, as well as providing a lot more data. In contrast, human kidney biopsies are thin tissue strands extracted with a hollow needle, resulting in many patches being situated on the boundary between tissue and non-tissue. We refer to Figure 4.4 for a visual comparison of biopsies and resec-



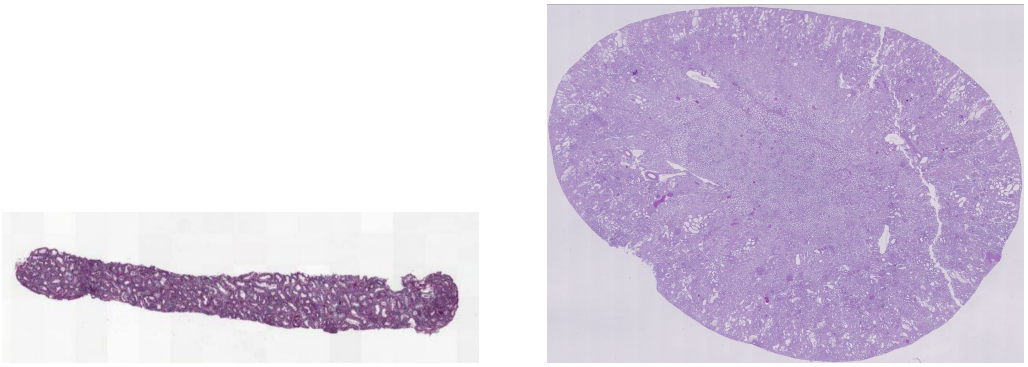
**(a)** ImageNet starting point.



**(b)** TCGA-BRCA starting point.

**Figure 4.3:** Two selections of samples of synthetic images produces by the models after 1 hour of training. Above, the denoising Unet was initialized with a checkpoint trained to create data found in ImageNet. Below, the starting checkpoint was trained to produce breast tissue from the TCGA-BRCA dataset. By eye, no clear difference can be seen between the two selections.

tions.



**(a)** A biopt of human kidney tissue. **(b)** A resection of rat kidney tissue.

**Figure 4.4:** Comparison between a biopt and a resection of kidney tissue. Intuitively, patch datasets constructed from a resection contain fewer patches with the border between tissue and background. Note the large difference in size between the two images.

One downside of using rat WSIs is that they are unannotated, i.e. no ground truth layer segmenting the tissue is available. Thus, we artificially created such annotations using Aiosyns segmentation model. This is an AI tool which analyzes the entire WSI and segments it into 8 distinct classes such as Arteries, Glomeruli or Tubuli. It also discriminates between sclerotic (scarred) and non-sclerotic (healthy) glomeruli. After running the unlabelled data through the segmentation model, we make a patch dataset out of it, using the models prediction as a ground truth. With every patch of kidney tissue comes a mask, detailing which parts of the image are which tissue type. This mask will be used to create a textual description of the contents of the image, so absolute accuracy is not important as only the broad structure of the image dictates the descriptions. The patch-mask pairs will serve as our training data.

### 4.1.4 Guidance

In principle, any medium can be used as conditioning with our model setup. All that is required is a powerful enough encoding component to compute meaningful embeddings of the conditioning data. We opted to use PLIP [45] to compute our embeddings, which is a pathology specific language encoder. It was trained using over 200 000 pathology images paired with

natural language descriptions of those images. It works by computing two embedding vectors, one for the image and one for the description. Embedding computations were tuned so embeddings would match for pairs, and be dissimilar for non-pairs using contrastive learning, pioneered by CLIP [46].

Contrary to the method proposed by Rombach et al., we do not train the guidance mechanism while optimizing the diffusion model. Instead, we simply treat PLIP as part of the fixed dataloader and use it as a black box tool.

To leverage the powerful PLIP encoder, we need a suitable textual description for each image in our training dataset. This process involves analyzing the mask associated with each image. For each of the 8 classes potentially present in the mask, we determine the proportion of pixels belonging to that class. Drawing inspiration from methods detailed in [28], we categorize these proportions into one of three classes: classes with less than 20% prevalence are labeled as *low*, those between 20% and 40% as *medium*, and anything above 40% as *high*. Subsequently, a descriptive sentence is generated for each class present in the mask, indicating both the class identity and its prevalence level (low, medium, or high). Additionally, an introductory sentence is appended to the beginning of each caption, irrespective of the mask contents. This approach results in captions structured as follows:

> This image showcases various types of tissue structures found in renal tissue.
>
> The prevalence of Tubuli is medium.
>
> There is a lot of Glomeruli visible in the image.

and

> This image showcases various types of tissue structures found in renal tissue.
>
> The image shows a low amount of Atrophic Tubuli.
>
> There is a lot of Tubuli visible in the image.
>
> The image shows a low amount of Sclerotic Glomeruli.
>
> The prevalence of other kidney tissue is medium.

Classes which do not occur anywhere in the patch are omitted from the description. The caption is then run through PLIP to encode the information in an embedding, which is used as guidance during the diffusion process. Ul-

timately, when trained, we will use descriptions like these to sample images from the model with predetermined contents.

### 4.1.5 Model Configuration

To train the model, we use Pytorch and the standard training loop from Pytorch Lightning[6] version `1.9.5`. As optimizer, we used AdamW [47], which is standard Adam [48] with added weight decay. We employ a learning rate of $2.5 \times 10^{-5}$, with 1 000 warmup steps. The loss function is, in accordance with Equation (3.15), MSELoss. To perform sampling, we make use of a technique called DDIM [49] with 50 steps, with classifier free guidance scale of $\gamma = 1.5$. Unconditional model calls are simulated by simply setting the caption to the empty string. DDIM can be viewed as an extension of DDPM focussed on the sampling process specifically: It builds on the trained diffusion model to significantly speed up sampling speeds.

When preprocessing the input data, we normalize the images to $[-1, 1]$, as well as stochastically flipping the images horizontally or vertically, both with probability 0.5.

When training, it proved crucial to perform all operations in `float32` precision, and *not* in `float16`. Initial training runs utilized the autocasting capabilities of torch's Automatic Mixed Precision[7] package, and seemingly training occurs without error. However, during early experiments, we noticed some weights remained stagnant. We ultimately found it to be caused by their gradients being smaller than the minimum positive value supported by half precision floats. Therefore, the gradients were rounded down to zero, meaning their corresponding weights do not update during training. This has the effect that only *part* of the model gets trained, which means our model output changes, but without using the entire potential of the model. Thus, the images created will look like a strange mix between breast tissue and kidney tissue, see Figure 4.5. Training the model in `float32` ensures *all* weights get updated appropriately, and learning can

[6] https://lightning.ai/docs/pytorch/stable/
[7] https://pytorch.org/docs/stable/amp.html

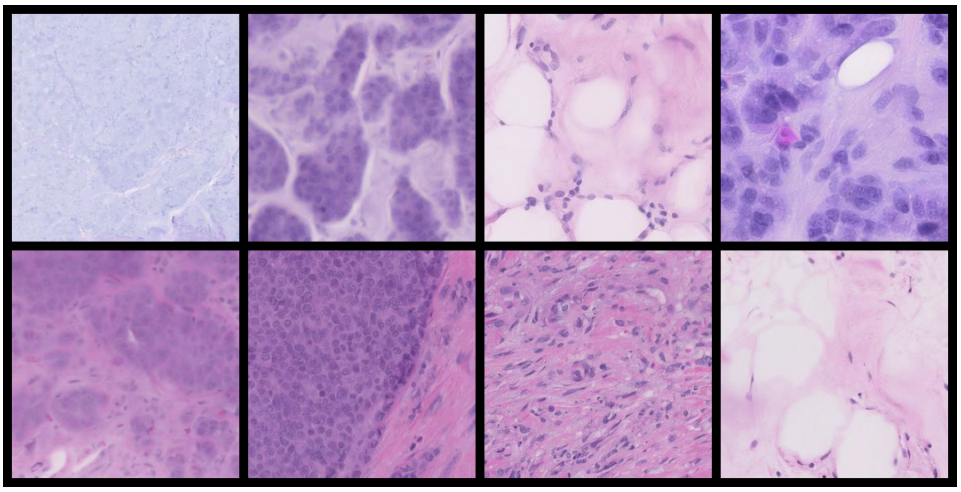


**Figure 4.5:** Mix between breast tissue and kidney tissue, as created when training with 16-bit precision.

occur.

## 4.2 Model Analysis

In this section, we outline how we evaluate our trained diffusion model, exploring its effectiveness and limitations through three distinct testing methodologies. These methodologies offer both a quantitative and a qualitative perspective, allowing us to evaluate the model's performance from multiple angles and gain insights into its behavior. They also allow us to identify potential weaknesses and areas of improvement for possible future work.

### 4.2.1 Caption Accuracy

We want to be able to use our trained diffusion model to create images of specific renal structures. As written in Section 4.1.4, training was guided by a PLIP-embedding of text. Consequently, we can steer the synthesis of images using a similar natural language caption. To test if we can succeed in guiding the model towards specific structures, we will segment synthetic images using Aiosyns segmentation model and compare the segmentation against the generating caption.

We limited ourselves to analysing the prevalence of Glomeruli in synthetic images, as they are an example of a scarce and interesting renal struc-

ture. The choice to focus on Glomeruli is motivated by our further experiments, as we will be detailed in Section 4.2.3. There, we will use a dataset of synthetically generated Glomeruli to *train* a segmentation model. Therefore it makes sense to analyse the capability of the model to generate this specific structure. To enable analysis, we first need a representative dataset. Thus, we created a dataset of 400 $256 \times 256$ pixel samples all with the same caption:

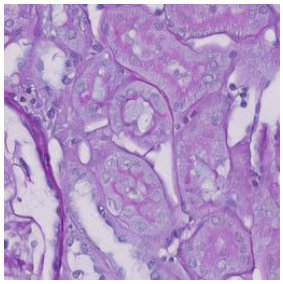
> This image showcases various types of tissue structures found in renal tissue.
>
> The prevalence of Tubuli is medium.
>
> There is a lot of Glomeruli visible in the image.

This caption will be referred to as the *generating* caption. Captions such as this were present during training, which means the model should have an understanding of the image that it is supposed to create.

Next, we analyze the synthetic images using a segmentaion model. For the image-mask pairs in the training dataset, the caption information is based on the entirety of the mask of the patch ($256 \times 256$ pixels), which means that the entirety of the image should be taking into account when trying to compare caption accuracy. In other words, if we sample an image with a low amount of a specific structure, it is entirely possible that that tissue occurs somewhere along the edges. However, Aiosyns segmentation model is a convolutional model, trained using valid padding (i.e., no padding). This means that, given an input of $W \times H$ pixels, only the centremost $(W - 184) \times (H - 184)$ pixels will be segmented. This poses an issue for us, as we need to segment the entire image to give a fair assessment of the caption accuracy.

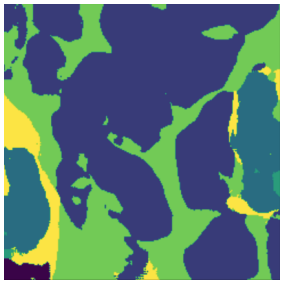
There are several solutions to circumvent this issue, and we opted for a simple solution where we mirror the image at the edges to stretch it out by 92 pixels on each border. Then, if we run the padded image through the segmentation model, those outer edges will be convolved off, and we are left with a segmentation of the original image. The process is shown in Figure 4.6, starting with a synthetic image in Figure 4.6a, the mirror-padded image in Figure 4.6b and finally the generated segmentation of the image in Figure 4.6c.

Unfortunately, due to internal workings of the segmentation model, it is not possible to feed it an image of $440 \times 440$ pixels. This is due to the fact that the U-net makes use of skip-connections where it stores and later concatenates some of the encoder outputs. When using valid padding, we must ensure even dimensions throughout all pooling steps to ensure the dimensions of the skip-connections match up. Otherwise, the shape of the encoder output does not exactly match that of the decoder output, therefore they cannot be concatenated. We decided to resolve this by padding with 86 pixels instead of 92, which is the largest padding value lower than 92 that still allows the Unet model to function. Using 86 padding means we feed the Unet an image of shape $428 \times 428$ pixels, and the result is a segmented image of shape $244 \times 244$ pixels. Consequently, we do not segment a boundary of 12 pixels. We deem this loss of accuracy acceptable as large structures are highly unlikely to be present *only* in this boundary.



**(a)** Unmirrored synthetic image that must be segmented.

**(b)** Mirrored image used as input for the segmentation model.



**(c)** Segmented image.

**Figure 4.6:** Segmentation process.

Next, the segmented image is processed in the same way that is done for training: The probability that a random pixel belongs to a specific class is computed by counting how many pixels are of that class, and dividing by the total number of pixels. Next, we iterate over each class probability and categorizes the prevalence of each tissue structure into 'low', 'medium' or 'high', based on the thresholds 0.2 and 0.4. With these categorizations, a human-readable caption is created like explained in Section 4.1.4, containing all information per class (non-present classes are omitted from the caption). This caption will be referred to as the *predicted* caption of a synthetic image.

Finally, all we need to do is compare the generating caption with the predicted caption to see how well the model was guided toward the caption. We will be testing for Glomerulus presence, as well as *how much* of it is present.

### 4.2.2 Human Perception

One of the goals of the generative model is to be able to generate realistic-looking images. In order to test whether or not our model is capable of this, we created a survey containing both real and synthetic images, and asked participants to classify images as real or synthetic. More specifically, we randomly selected 20 images of $256 \times 256$ pixels images from the rat-tissue dataset. The selected images were verified to contain different types of tissue, including glomeruli, tubuli and other structures. Additionally, we selected 20 synthetic images by hand. These images were selected from a larger synthetic dataset by Thomas de Bel, an AI engineer at Aiosyn with a lot of experience with renal images. Selection was done to filter out the images that are dead-giveaway synthetic, as that is not the purpose of this test. However, it is worth noting that the majority of generated images pass this pre-selection, and that seriously wrong images are rare. Testing the model's best images provides the most accurate assessment of its ability to generate realistic images.

The (in total 40) images were randomly shuffled, and participants were asked to classify the images as Real or Synthetic. They were not given a time-limit to do so. Additionally, respondents were asked to rate their own experience with renal histopathology on a scale of 1 to 5, with 1 being 'Barely any experience', and 5 being 'Pathologist'. As a final question, participants had the option to disclose if they noticed any patterns or had a strategy to decide between real and synthetic.

The survey was sent to employees of Aiosyn as well as a few pathologists. The results are analysed using Cohen's Kappa, which is a statistical measure of inter-rater agreement [50]. It takes a value $\kappa \in [-1, 1]$, where 1 means complete agreement, $-1$ complete disagreement and 0 chance agree-

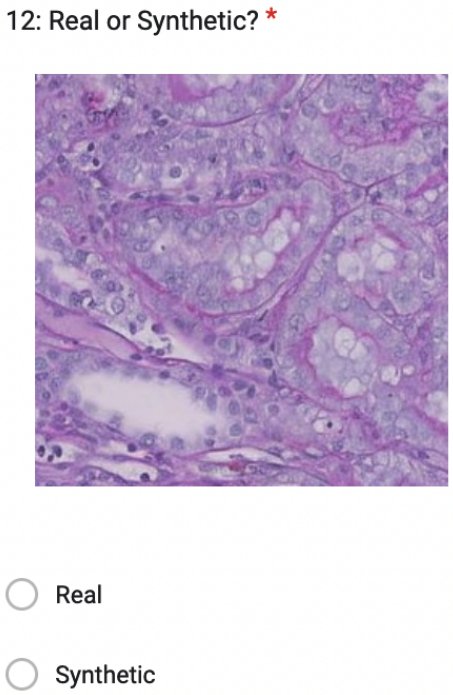



**Figure 4.7:** Example question from the survey. Correct answer: Synthetic.

ment (i.e., the agreement one would expect if the raters are completely uncorrelated). In our analysis, the ground-truth answers will be considered one rater and the respondent the other.

### 4.2.3 Use in Model Training

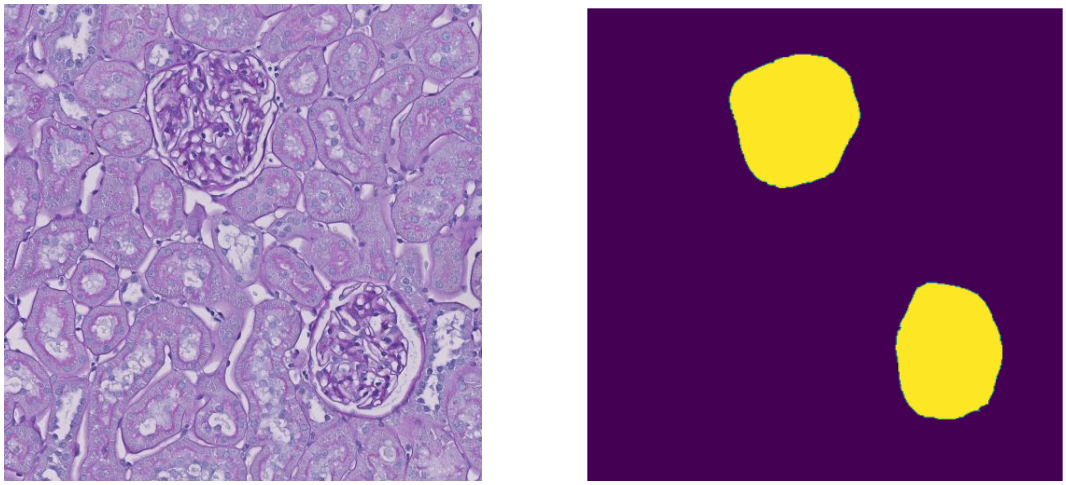
Real-life datasets can suffer from limited data availability. By supplementing datasets with synthetic images, we could potentially reduce chance of overfitting and allow for more robust training. Moreover, generated images can be tailored to include rare or underrepresented pathological features to increase performance across a range of different situations.

With the aim of trying to test whether there is any merit to using our generated images for training, we set up the following experiment: We trained 6 models to segment glomeruli in real-world tissue patches, using various ratios of real/synthetic training data, ranging from solely real data to solely synthetic data. In all training runs except one, the total number of training samples remains the same. In that one exceptional case, the real dataset is *supplemented* (not replaced) by the synthetic dataset, effectively doubling

the size of the training data. The datasplits can be found in Table 4.1. See Figure 4.8 for an illustration of the segmentation task.

**Table 4.1:** Experiment Setup

| Experiment Name | Real Share | Synthetic Share | Training Patches (#) |
|---|---|---|---|
| 100_0 | 1.00 | 0.00 | 198 |
| 75_25 | 0.75 | 0.25 | 198 |
| 50_50 | 0.50 | 0.50 | 198 |
| 25_75 | 0.25 | 0.75 | 198 |
| 0_100 | 0.00 | 1.00 | 198 |
| 100_100 | 1.00 | 1.00 | 396 |



**(a)** Real image to be segmented. **(b)** Correct segmentation.

**Figure 4.8:** Training image-mask pair.

We created the training patch dataset of synthetic data in the following way:

- To start, we sampled 1600 patches of 512 $\times$ 512 pixels, all with the same caption (the generating caption used in Section 4.2.1) guiding the images to contain a large amount of Glomeruli, with medium prevalence of Tubuli. This took around 10 hours in total on Amazon Web Services' servers.
- The patches were stitched together, side to side, to form one large 'synthetic whole slide' `.tif` file, containing 40 by 40 patches. See Figure 4.9.

- Part of the stitched `.tif` file was then annotated, creating 740 annotations of glomeruli. Note that these annotations were done by an unexperienced annotator, so their quality and accuracy is not guaranteed.
- A dataset of 512 $\times$ 512 pixel patches was created from the annotated `.tif` file, containing 198 patch-mask pairs to indicate what part of the tissue is glomeruli and what part is not. This dataset will be referred to as the Synthetic Training (ST) dataset.

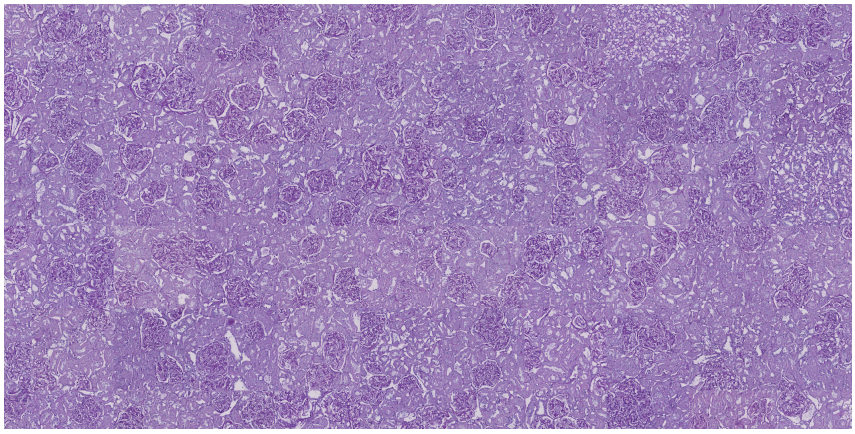


**Figure 4.9:** Small part of the stitched synthetic whole slide. Note the visible borders between two patches of 512 $\times$ 512 pixels are clearly visible.

To be able to directly compare the ST dataset with real data, we curated a dataset of equal size from real data, referred to as the Real Training (RT) dataset. The RT dataset was annotated by Aiosyn's annotators, and only the annotations pertaining to glomeruli were taken into account. All other parts of the image were considered as background.

For all experiments, the validation (dubbed the Real Validation (RV) dataset) dataset exists solely of real patches of annotated rat tissue, from the same source as the RT dataset, but without overlap. This means that experiment 100_0 can be considered a baseline test, where we expect great performance on this relatively simple tasks.

It is important to note that the source of the RT and RV datasets is the same source that was used to train the diffusion model, which means that the ST dataset should theoretically be as similar as possible to the RV set. Model performance on the RV set is measured using a combination of Categorical Cross Entropy (which, in our simplified case, becomes Binary Cross

Entropy loss) as well as Dice Score Coefficient (DSC). DSC can be used to measure how well a segmentation models performs [51]. It ranges between 0 and 1, where 1 is ideal performance, meaning all pixels are classified correctly.

Aside from the training data, all training loops are identical, with identical seeding as well. No data augmentations were made aside from stochastically flipping horizontally and vertically, as well as randomly cropping. This provides us with the optimal situation to directly compare the impact of swapping the training data from real to synthetic. All training runs were allowed to continue until stopped by the EarlyStopping callback, which was configured to monitor the validation loss.

# 5. Results

In this chapter, we will present the outcomes of the experiments defined in Chapter 4.

## 5.1 Caption Accuracy

As explained in Section 4.2.1, we created 400 256 $\times$ 256 pixel images using the generating caption

> This image showcases various types of tissue structures found in renal tissue.
>
> The prevalence of Tubuli is medium.
>
> There is a lot of Glomeruli visible in the image.

We tested the predicted caption of the synthetic image, and in 91.3% of all generated images, there is mention of Glomeruli (sclerotic or non-sclerotic) in the image (See Table 5.1. This means that the model succeeded in generating renal tissue that contains glomerulus reliably. However, in only 4.2% of those images were the glomeruli deemed not sclerotic. It seems that the model is able to capture the general look and structure of glomeruli, but fails to generate it cleanly. Small errors introduced can quickly make the generated glomerulus look unhealthy, i.e. sclerotic. It is worth noting that the data used to train the diffusion model comes from a mix of healthy and unhealthy rat tissue, so perhaps the training data for healthy glomeruli was somewhat lacking.

One other takeaway from Table 5.1 is that the model is not great at creating large patches of glomeruli. Even when we combine sclerotic and non-sclerotic glomeruli, they only cross the 'High' threshold (40% of the image) in less than 5% of cases. It is much more common to create a Medium amount of combined sclerotic and non-sclerotic glomeruli, at 28.02%. But in most cases, a Low amount is produced, at just below 60% (See Table 5.1c).

In only 3.86% of images glomeruli were present, but no sclerotic glomeruli.

| Amount | % |
|---|---|
| None | 12.34 |
| Low | 77.38 |
| Medium | 9.77 |
| High | 0.50 |
| Total | 100.00 |

**(a)** Glomeruli (non-sclerotic).

| Amount | % |
|---|---|
| None | 12.60 |
| Low | 78.92 |
| Medium | 7.46 |
| High | 1.03 |
| Total | 100.00 |

**(b)** Sclerotic Glomeruli.

| Amount | % |
|---|---|
| None | 8.74 |
| Low | 58.35 |
| Medium | 28.02 |
| High | 4.88 |
| Total | 100.00 |

**(c)** Sclerotic Glomeruli and Glomeruli combined.

**Table 5.1:** Occurence of Glomeruli in the synthetic dataset.

Conversely, 3.60% of images contained only sclerotic glomeruli, but no healthy glomeruli. Therefore we can say that the model tends to generate a mix of sclerotic and healthy glomeruli, as far as the segmentation model can tell. This might be indicative of a lack of clear differentiation between the two structures by the model, but we cannot say this for certain without more research.

In conclusion, the model demonstrates reliable capability in generating images of glomeruli. However, it faces challenges in producing images where a large portion is comprised of glomeruli. Furthermore, we observe limited control over the glomeruli's state, with both healthy and sclerotic conditions occurring at comparable rates despite only guiding towards healthy tissue.

## 5.2 Human Perception

Recall that to test how realistic synthetically generated images look, we conducted a survey containing 40 images, of which 20 were real and 20 were synthetic. Participants were asked to determine whether each image was real or synthetic. The survey was sent out to employees of Aiosyn, as well as to some people working actively in the field of pathology. We received 13 responses from people with varying levels of experience with renal histopathology.

Here, we analyse the results using Cohen's Kappa, ranging from $-1$ (complete disagreement) to 1 (complete agreement). $\kappa = 0$ means chance

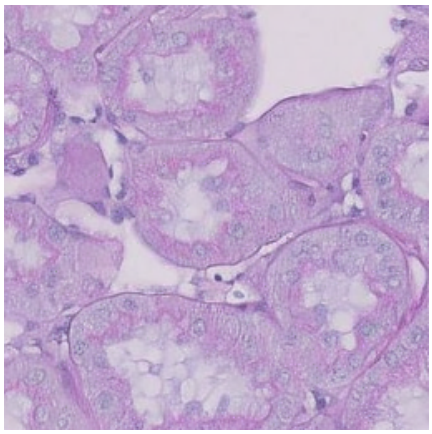
agreement, the agreement one would expect if the raters are completely uncorrelated. If respondents are unable to tell synthetic from real, a $\kappa$ value of 0 would be expected.

Interestingly, mean $\kappa$ achieved was $-0.08$, indicating that participants actually performed slightly worse than would be expected of someone randomly guessing. In fact, the maximum $\kappa$ a participant got was 0.30, (which corresponds to 25 correct, 15 incorrect answers) and the minimum was $-0.40$ (12 correct, 28 incorrect). These results indicate that, generally, people cannot tell the difference between real and synthetic images.

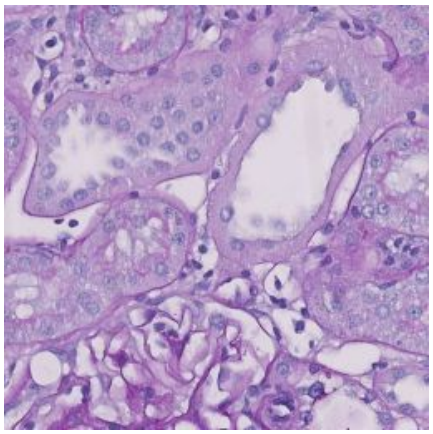
Additionally, there seems to be no correlation between experience level and performance on the survey. Participants with an experience of 1 or 2 scored $-0.09$ on average, while respondents with experience of at least 3 had an average $\kappa = -0.18$.

On average, people had 18.5 answers correct. The majority vote also did not lead to convincing results, being correct in a mere 17 out of 40 questions.

Some images were identified correctly more often than others: Question 9 shows a synthetic image, but was incorrectly identified as real in 11 out of 13 instances, see Figure 5.1a. Question 15 also had only 2 correct answers, see Figure 5.1b. For an overview of all answers, see Appendix B, which contains all responses for each question, as well as the ground truth answers.



**(a)** Synthetic image most often classified as real.



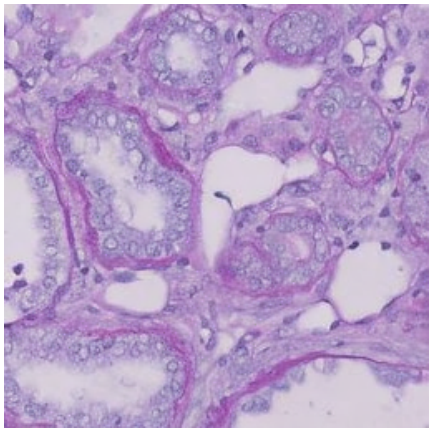
**(b)** Real image most often classified as synthetic.

**Figure 5.1:** Images most often classified incorrectly.

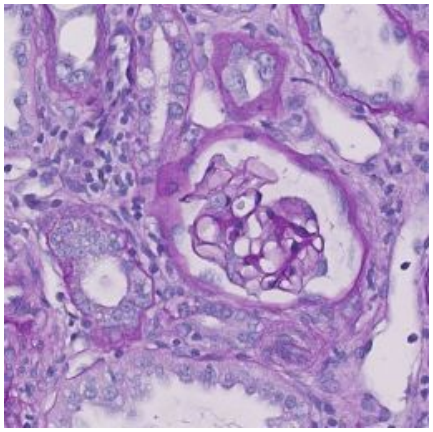
An example of a poor synthetic image is given in Figure 5.2a. This image

was identified as synthetic by 11 out of 13 participants. The real image with the most correct answers is shown next to it, with 10 right answers. On average, questions were answered correctly by 6 people.

Exactly why certain images have profoundly different rates of correct answers is unclear. One respondent reported looking at the white areas, explaining that (in their opinion) they looked slightly too big in some images, while another looked at smaller details, such as incorrectly stained nuclei. Neither strategy produced significantly better results, forcing us to question the validity of their claims. It seems likely that the variance observed is simply due to a relatively low amount of respondents.



**(a)** Synthetic image most often classified as synthetic.



**(b)** Real image most often classified as real.

**Figure 5.2:** Images most often classified correctly.

In summary, it is clear the model is capable of producing synthetic images that are indistinguishable from real, thereby opening the door to its use in renal histopathological training.

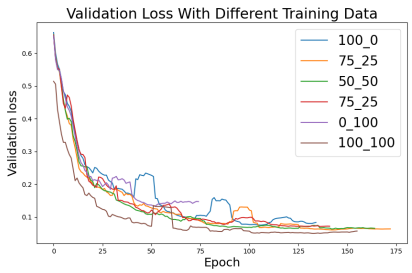
## 5.3 Use in Model Training

Remember that experiment 100_0 uses solely real training data, and serves as a baseline to compare other experiments against. Since the validation set remains consistent throughout all experiments, it is valuable to analyze the validation loss. In Figure 5.3a, the validation loss of each of the training runs is shown (smoothed, with a window size of 10). Some experiments

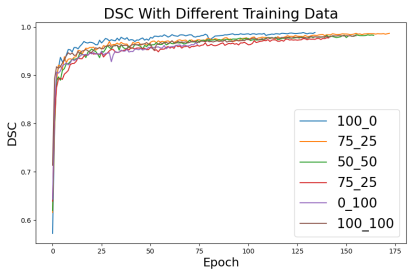
trained for longer than others, this is down to the EarlyStopping callback breaking off training after validation loss stagnated. From Section 5.3, a few things stand out. First of all, all models learn, and the validation loss drops to a value close to 0. However, training *solely* with synthetic data did the worst out of our experiments, obtaining a loss of roughly double the best score, which was achieved by using an even split of real and synthetic data (50_50).

Recall that DSC is a measure ranging from 0 to 1, where 1 means every pixel is correctly classified. Table 5.2 shows per experiment what the final validation loss and DSC was. Keep in mind these are the scores achieved at the end of training, not the highest achieved during training. Still, from these results it is clear that having a mix of real and synthethic data during training can outperform *only* using real data.

Furthermore, 50_50 even got a better final DSC than 100_100, despite having a training dataset half the size. Generally speaking, we expect 100_100 to perform the best since it simply has more available training data. This is clearly visible when analyzing the validation loss, but the advantage slims when we look at the graph for DSC (Figure 5.3b).




**(a)** Smoothed validation loss curves for the experiments.




**(b)** DSC curves for the experiments.

Another aspect to notice is training speed: 100_100 understandably achieves its low validation loss and DSC score in fewer epochs, since it has more data. As for the other experiments, they seem to be on par with eachother.

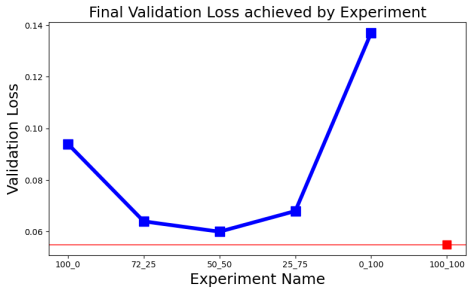
If we compare experiments 100_0 and 100_100, we can draw conclusions as to the effects of *supplementing* an existing dataset with synthetic data. On first glance, a DSC score increase from 0.956 to 0.971 might seem insignificant, but it is not. In fact, this reduces the area of incorrectly segmented

**Table 5.2:** Achieved Validation Loss and DSC at end of training.

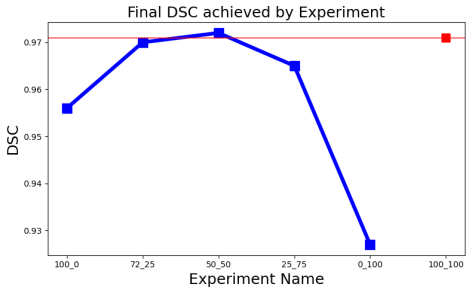
| Experiment Name | Validation Loss (lower is better) | DSC (higher is better) |
|---|---|---|
| 100_0 | 0.094 | 0.956 |
| 75_25 | 0.064 | 0.970 |
| 50_50 | **0.060** | **0.972** |
| 25_75 | 0.068 | 0.965 |
| 0_100 | 0.137 | 0.927 |
| 100_100 | 0.055 | 0.971 |

tissue by more than 34%.

The curves in Figure 5.4a and Figure 5.4b are particularly insightful: They show explicitly the difference in performance between the different experiments. Evidently, there is an optimal balance to be found between real and synthetic data, since using a combination of the two is scoring considerably better than using only real or only synthetic data. The 50_50 mix of real and synthetic performs best in both our metrics, and even rivals experiment 100_100, which contains twice as much data.




**(a)** Validation loss achieved at end of training.




**(b)** DSC achieved at end of training.

Exactly why a mix performs so well is yet to be determined conclusively, but we hypothesise that it might be a result of increased variety in the training samples. This trains the model to deal with more diverse validation samples by forcing the model to develop a quite general understanding of glomeruli. Ordinarily, training data variety can be achieved simply by increasing the size of the training dataset or performing various data augmentations, which are well-known to increase performance. It is possible that the synthetic dataset allows the model to learn a much broader notion of a

glomerulus, which translates well to performance on the validation set.

# 6. Discussion

In this chapter we will discuss several shortcomings, opportunities and implications of the research conducted. There are several interesting avenues to be explored that we did not have the time for. In this section, we want to highlight a few of them.

## 6.1 Guidance

In Section 4.1.4, we discuss our method of generating a caption. To generate a caption for a given image, we perform some analysis on the corresponding mask to determine the prevalence of certain renal structures. Using these prevalences, static sentences are chained together to create a short paragraph broadly describing the contents of the image. This basic strategy admits only a small amount of different captions, and many different images are described using the exact same caption. Furthermore, we use simple language without any domain-specific jargon, which means we might not use PLIPs full capabilities. It would therefore be interesting to tune the captions more finely to investigate if it improves model performance, specifically performance of the guidance mechanism. It is clear from our tests that the trained model is capable of producing realistic-looking images at the drop of a hat. However, generating *specific* images is not as simple: It requires some tuning of the caption to generate exactly what is asked. As the results in Section 5.1 show, the model tends to fill up the image with tissue types that were not present in the caption. Potentially, increasing the specificity of textual descriptions can alleviate some of this struggle.

Another way to enhance the guidance system is by moving away from textual guidance altogether. Instead, we could use an embedding of the mask directly. Unlike the pretrained PLIP, this encoder would need to be trained concurrently with the denoising diffusion model, as outlined in [8].

Alternatively, one could pretrain an encoder for masks, simplifying the training process for the diffusion model. Conditioning your model on a mask offers the added benefit of having a ground-truth mask readily available for generated images. In our current setup with captions, synthetic data requires manual annotation before being used in downstream tasks such as model training. With mask-based guidance, sampling would be simplified to providing a mask and synthesizing an accompanying image, meaning synthetic data could be used in down-stream tasks without requiring human intervention. However, we want to highlight the advantage of manually annotating as well. If the masks were to be generated concurrently the images, there would be no outside injection of information; training a segmentation model with *fully* synthetic data is more like transferring the understanding of renal tissue from one model (the generative model) to another (for example, a segmentation model), known as *Transfer Learning* [52]. With manually annotated images, there is some expert knowledge *added* to the synthetic data, reducing cyclicity and possibly making data more effective for learning downstream tasks.

## 6.2 Synthesizing WSIs

Additionally, one compelling direction for further exploration is the challenge of creating large, coherent images. When generating the synthetic patch dataset, we initially stitched together 1600 images side by side, forming a patchwork blanket of renal tissue images. However, creating a patch dataset from this mosaic is complicated by the necessity to avoid borders between synthetic images within each patch. This constraint imposes limitations on the types of patch datasets that can be created, as well as on the overall quality and consistency of the patches.

There are multiple ways one could try to solve this issue. Our trained model is a convolutional Unet. Consequently, samples of arbitrary size can be generated. However, we quickly run into hardware limitations: creating $512 \times 512$ images could only be done in batch size of 4 on an AWS EC2 instance. Potentially, we could reduce the batch size to increase the sampling

image size, but we will not be able to generate images of substantial size, simply due to limited memory of current GPUs.

In [27], the authors successfully generate whole slide images (WSIs) at resolutions up to $65\,536 \times 65\,536$ pixels by gradually increasing the resolution of the generated image. Initially, they use a diffusion model to sample a low-resolution image of $512 \times 512$ pixels of a WSI. This image is then refined sequentially in multiple steps, with each step producing a higher-resolution image guided by the previous lower-resolution image to ensure consistency in high-level features and coarse structures. This approach enables the generation of realistic WSIs of breast tissue using the TCGA-BRCA dataset [42]. The authors avoid stitching artifacts through a technique called grid-shift, achieving results where pathologists could not reliably distinguish between real and synthetic WSIs. Employing such methods to generate a consistent WSI could make annotating synthetic data easier.

## 6.3 Annotation quality

Another potential limiting factor of our developed model is the input data: We use unannotated rat tissue to train our model. The way we obtain conditioning information is by using a trained segmentation model, which might be less accurate than manual annotation by trained professionals. Additionally, this creates a highly circular training structure (segmentation model is needed to train the diffusion model, which is used to train a segmentation model), which may lead to deterioration of performance, although we did not observe this in our tests.

## 6.4 Finetune VAE on renal tissue

Additional performance improvements might be achieved by finetuning the VAE on histopathology images of renal tissue (see Section 3.5.3 for details). Our studies utilized a VAE trained on the TCGA-BRCA dataset of breast tissue, which demonstrated strong potential in encoding renal tissue. However, finetuning on a renal tissue dataset could further enhance its perfor-

mance. Due to time constraints, this was not pursued in this thesis.

## 6.5 Use in Model Training

In Section 5.3, we observe that training a segmentation model with a synthetic dataset can yield better performance than using a real dataset of the same size. However, this comparison might be considered unfair. The synthetic dataset benefits from the extensive data used to train the diffusion model, indirectly enhancing its quality. It's plausible that using the vast amounts of data directly to train the segmentation model could lead to even better results. This is something to be explored in more detail in later works.

Furthermore, in our setup, we train a segmentation model to differentiate between glomeruli and other tissue, which is an almost trivial task for powerful segmentation models. It would be compelling to research if the found results translate to more complex segmentation tasks involving more than two classes, or other downstream tasks in general.

# 7. Conclusion

In this work, we showed how to develop a latent diffusion model capable of synthesizing high quality images of kidney tissue. We queried industry professionals to tell real from synthetic images, but found nobody could do so reliably, proving our models synthetic images are highly realistic, paving the way for use in renal histopathological training. Furthermore, our model is text-conditional, allowing for us to sample specific structures on command. We explored the efficacy of our guidance system and found its accuracy to be fair, but with room for improvement. Finally, we used our model to sample 1600 images and created a synthetic training-dataset out of it by manually annotating a portion of the images. With this dataset, we trained a toy segmentation model. We then compared the performance of this model against models trained with different ratios of real to synthetic data. Our findings indicate that models trained with synthetic data can match or even outperform those trained solely with real data.

## 7.1 Future Research

As explained in Chapter 6, there are a lot of parameters to finetune during the model training process. It would be interesting to compare outputs of two differently trained LDMs to see where we can maximize performance. Additionally, we showed synthetic data can be used to increase training performance on a simple task. Investigating whether similar results can be achieved on more complex tasks would be an interesting endeavor.

# Appendices

# A. Appendix: Mathematical Derivations

In this Appendix, we include several derivations of formulas that are omitted from Chapter 3 for brevity.

## A.1 Derivation of ELBO for VDMs

Derivation of the ELBO:

$$\log p(x) = \log \int p(x_{0:T}) dx_{1:T} \tag{A.1}$$

$$= \log \int \frac{p(x_{0:T}) q(x_{1:T}|x_0)}{q(x_{1:T}|x_0)} dx_{1:T} \tag{A.2}$$

$$= \log \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \frac{p(x_{0:T})}{q(x_{1:T}|x_0)} \right] \tag{A.3}$$

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_{0:T})}{q(x_{1:T}|x_0)} \right] \tag{A.4}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^{T} q(x_t|x_{t-1})} \right] \tag{A.5}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T) p_\theta(x_0|x_1) \prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)}{q(x_T|x_{T-1}) \prod_{t=1}^{T-1} q(x_t|x_{t-1})} \right] \tag{A.6}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T) p_\theta(x_0|x_1) \prod_{t=1}^{T-1} p_\theta(x_t|x_{t+1})}{q(x_T|x_{T-1}) \prod_{t=1}^{T-1} q(x_t|x_{t-1})} \right] \tag{A.7}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T) p_\theta(x_0|x_1)}{q(x_T|x_{T-1})} \right] \tag{A.8}$$

$$+ \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \prod_{t=1}^{T-1} \frac{p_\theta(x_t|x_{t+1})}{q(x_t|x_{t-1})} \right] \tag{A.9}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)} [\log p_\theta(x_0|x_1)] \tag{A.10}$$

$$+ \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)}{q(x_T|x_{T-1})} \right] + \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \sum_{t=1}^{T-1} \log \frac{p_\theta(x_t|x_{t+1})}{q(x_t|x_{t-1})} \right] \tag{A.11}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)} [\log p_\theta(x_0|x_1)] \tag{A.12}$$

$$+ \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)}{q(x_T|x_{T-1})} \right] + \sum_{t=1}^{T-1} \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p_\theta(x_t|x_{t+1})}{q(x_t|x_{t-1})} \right] \tag{A.13}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)} [\log p_\theta(x_0|x_1)] \tag{A.14}$$

$$+ \mathbb{E}_{q(x_{1:T}|x_0)} \left[ \log \frac{p(x_T)}{q(x_T|x_{T-1})} \right] + \sum_{t=1}^{T-1} \mathbb{E}_{q(x_{t-1},x_t,x_{t+1}|x_0)} \left[ \log \frac{p_\theta(x_t|x_{t+1})}{q(x_t|x_{t-1})} \right] \tag{A.15}$$

$$= \underbrace{\mathbb{E}_{q(x_1|x_0)} [\log p_\theta(x_0|x_1)]}_{\text{reconstruction term}} - \underbrace{\mathbb{E}_{q(x_{T-1}|x_0)} [D_{KL}(q(x_T|x_{T-1}) \| p(x_T))]}_{\text{prior matching term}} \tag{A.16}$$

$$- \sum_{t=1}^{T-1} \underbrace{\mathbb{E}_{q(x_{t-1},x_{t+1}|x_0)} [D_{KL}(q(x_t|x_{t-1}) \| p_\theta(x_t|x_{t+1}))]}_{\text{consistency term}} \tag{A.17}$$

## A.2 Derivation of ELBO using Bayes rule

We continue the derivation at Equation (A.4) of the previous derivation, applying log rules and telescoping series along the way.

$$\log p(x) \geq \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_{0:T})}{q(x_{1:t}|x_0)}\right] \tag{A.18}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^{T} q(x_t|x_{t-1})}\right] \tag{A.19}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)\prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)}{q(x_1|x_0)\prod_{t=2}^{T} q(x_t|x_{t-1})}\right] \tag{A.20}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)p_\theta(x_0|x_1)\prod_{t=2}^{T} p_\theta(x_{t-1}|x_t)}{q(x_1|x_0)\prod_{t=2}^{T} q(x_t|x_{t-1}, x_0)}\right] \tag{A.21}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{\frac{q(x_{t-1}|x_t,x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)}}\right] \tag{A.22}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{\frac{q(x_{t-1}|x_t,x_0)\cancel{q(x_t|x_0)}}{\cancel{q(x_{t-1}|x_0)}}}\right] \tag{A.23}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_T)p_\theta(x_0|x_1)}{\cancel{q(x_1|x_0)}} + \log \frac{\cancel{q(x_1|x_0)}}{q(x_T|x_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)}\right] \tag{A.24}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_T)p_\theta(x_0|x_1)}{q(x_T|x_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)}\right] \tag{A.25}$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log p_\theta(x_0|x_1)\right] + \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p(x_T)}{q(x_T|x_0)}\right] \tag{A.26}$$
$$+ \sum_{t=2}^{T} \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)}\right]$$

$$= \mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0|x_1)\right] + \mathbb{E}_{q(x_T|x_0)}\left[\log \frac{p(x_T)}{q(x_T|x_0)}\right] \tag{A.27}$$
$$+ \sum_{t=2}^{T} \mathbb{E}_{q(x_t,x_{t-1}|x_0)}\left[\log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)}\right]$$

$$= \underbrace{\mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0|x_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{KL}(q(x_T|x_0) \parallel p(x_T))}_{\text{prior matching term}} \tag{A.28}$$

$$- \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(x_t|x_0)}\left[D_{KL}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))\right]}_{\text{denoising matching term}} \tag{A.29}$$

where in Equation (A.23), we make use of the telescoping nature of the product wrapped in a logarithm.

## A.3 Derivation of tractable formula for $x_t \sim q(x_t|x_0)$

Suppose we have a set of $2T$ noise variables $\{\epsilon_t^*, \epsilon_t\} \sim^{\text{iid}} \mathcal{N}(\epsilon; \mathbf{0}, \mathbf{1})$. Then, we can rewrite $x_t \sim q(x_t|x_0)$ as follows:

$$
\begin{aligned}
x_t &= \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}^* && \text{(A.30)}\\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-2}^*) + \sqrt{1-\alpha_t}\epsilon_{t-1}^* && \text{(A.31)}\\
&= \sqrt{\alpha_t\alpha_{t-1}} x_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-2}^* + \sqrt{1-\alpha_t}\epsilon_{t-1}^* && \text{(A.32)}\\
&= \sqrt{\alpha_t\alpha_{t-1}} x_{t-2} + \sqrt{\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}^2 + \sqrt{1-\alpha_t}^2}\epsilon_{t-2} && \text{(A.33)}\\
&= \sqrt{\alpha_t\alpha_{t-1}} x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon_{t-2} && \text{(A.34)}\\
&= \dots && \text{(A.35)}\\
&= \sqrt{\prod_{i=1}^{t}\alpha_i} x_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\epsilon_0 && \text{(A.36)}\\
&= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0 && \text{(A.37)}\\
&\sim \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}) && \text{(A.38)}
\end{aligned}
$$

In step A.32, we summed two Gaussians. The resulting distribution has as mean the sum of the two means, an variance the sum of the two variances. When written using the reparameterization trick, we can consider $\sqrt{1-\alpha_t}\epsilon_{t-1}^*$ as a sample from $\mathcal{N}(\mathbf{0}, (1-\alpha_t)\mathbf{I})$ and $\sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-2}^*$ as a sample from $\mathcal{N}(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I})$. Then, we can consider their sum as a sample from $\mathcal{N}(\mathbf{0}, (1-\alpha_t+\alpha_t-\alpha_t\alpha_{t-1})\mathbf{I}) = \mathcal{N}(\mathbf{0}, (1-\alpha_t\alpha_{t-1}\mathbf{I}))$. Writing this sample using the reparameterization trick once more gives us the expression $\sqrt{1-\alpha_t\alpha_{t-1}}\epsilon_{t-2}^*$, as seen in Equation (A.34). Furthermore, $\bar{\alpha}_t$ is shorthand notation for $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$.

## A.4 Derivation of optimisation using $\mu_\theta(x_t, t)$

$$
\begin{aligned}
&\arg\min_\theta D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \\
&= \arg\min_\theta D_{\mathrm{KL}}(\mathcal{N}(x_{t-1}; \mu_q, \Sigma_q(t)) \parallel \mathcal{N}(x_{t-1}; \mu_\theta, \Sigma_q(t))) \\
&= \arg\min_\theta \frac{1}{2}\left[\log\frac{|\Sigma_q(t)|}{|\Sigma_q(t)|} - d + \mathrm{tr}(\Sigma_q(t)^{-1}\Sigma_q(t)) + (\mu_\theta - \mu_q)^T\Sigma_q(t)^{-1}(\mu_\theta - \mu_q)\right] \\
&= \arg\min_\theta \frac{1}{2}\left[\log 1 - d + d + (\mu_\theta - \mu_q)^T\Sigma_q(t)^{-1}(\mu_\theta - \mu_q)\right] \\
&= \arg\min_\theta \frac{1}{2}\left[(\mu_\theta - \mu_q)^T\Sigma_q(t)^{-1}(\mu_\theta - \mu_q)\right] \\
&= \arg\min_\theta \frac{1}{2}\left[(\mu_\theta - \mu_q)^T(\sigma_q^2(t)\mathbf{I})^{-1}(\mu_\theta - \mu_q)\right] \\
&= \arg\min_\theta \frac{1}{2\sigma_q^2(t)}\left[\left\|\mu_\theta - \mu_q\right\|_2^2\right]
\end{aligned}
$$

In this derivation, we write $\mu_q$ for $\mu_q(x_t, x_0)$ and $\mu_\theta$ for $\mu_\theta(x_t, t)$. In the second step, we make use of the standard formula for the KL Divergence between two Gaussian distributions.

## A.5 Derivation of optimisation using $\hat{x}_\theta(x_t, t)$

$$
\begin{aligned}
&\arg\min_\theta D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)) \\
&= \arg\min_\theta D_{\mathrm{KL}}(\mathcal{N}(x_{t-1}; \mu_q, \Sigma_q(t)) \parallel \mathcal{N}(x_{t-1}; \mu_\theta, \Sigma_q(t))) \\
&= \arg\min_\theta \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{\sqrt{\alpha}_t(1-\alpha_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{x}_\theta(x_t,t)}{1-\bar{\alpha}_t} - \frac{\sqrt{\alpha}_t(1-\alpha_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t}\right\|_2^2\right] \\
&= \arg\min_\theta \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{x}_\theta(x_t,t)}{1-\bar{\alpha}_t} - \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t}\right\|_2^2\right] \\
&= \arg\min_\theta \frac{1}{2\sigma_q^2(t)}\left[\left\|\frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}(\hat{x}_\theta(x_t,t) - x_0)\right\|_2^2\right] \\
&= \arg\min_\theta \frac{1}{2\sigma_q^2(t)}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2)}{(1-\bar{\alpha}_t)^2}\left[\left\|\hat{x}_\theta(x_t,t) - x_0\right\|_2^2\right]
\end{aligned}
$$

## A.6 Derivation of $\mu_q(x_t, x_0)$ in terms of $x_t$ and $\epsilon_0$.

$$\mu_q(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t} \tag{A.39}$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}_t}}}{1-\bar{\alpha}_t} \tag{A.40}$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + (1-\alpha_t)\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\alpha_t}}}{1-\bar{\alpha}_t} \tag{A.41}$$

$$= \frac{\sqrt{\alpha}_t(1-\bar{\alpha}_{t-1})x_t}{1-\bar{\alpha}_t} + \frac{(1-\alpha_t)x_t}{(1-\bar{\alpha}_t)\sqrt{\alpha}_t} - \frac{(1-\alpha_t)\sqrt{1-\bar{\alpha}_t}\epsilon_0}{(1-\bar{\alpha}_t)\sqrt{\alpha}_t} \tag{A.42}$$

$$= \left(\frac{\sqrt{\alpha}_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} + \frac{(1-\alpha_t)}{(1-\bar{\alpha}_t)\sqrt{\alpha}_t}\right)x_t - \frac{(1-\alpha_t)\sqrt{1-\bar{\alpha}_t}\epsilon_0}{(1-\bar{\alpha}_t)\sqrt{\alpha}_t} \tag{A.43}$$

$$= \left(\frac{\alpha_t(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)\sqrt{\alpha}_t} + \frac{(1-\alpha_t)}{(1-\bar{\alpha}_t)\sqrt{\alpha}_t}\right)x_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha}_t}\epsilon_0 \tag{A.44}$$

$$= \frac{\alpha_t - \bar{\alpha}_t + 1 - \alpha_t}{(1-\alpha_t)\sqrt{\alpha}_t}x_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha}_t}\epsilon_0 \tag{A.45}$$

$$= \frac{1-\bar{\alpha}_t}{(1-\alpha_t)\sqrt{\alpha}_t}x_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha}_t}\epsilon_0 \tag{A.46}$$

$$= \frac{1}{\sqrt{\alpha}_t}x_t - \frac{(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha}_t}\epsilon_0 \tag{A.47}$$

Note that in Equation (A.40), some terms $\bar{\alpha}_t$ lose their overline, which might be easy to miss.

# B. Appendix: Survey Results

| 2 | Truth | | 1 | 2 | 1 | 2 | 2 | 2 | 1 | 5 | 1 | 1 | 1 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | Real | | Real | Synthetic | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Real | Real | Real |
| 4 | Synthetic | | Synthetic | Synthetic | Synthetic | Synthetic | Real | Real | Synthetic | Synthetic | Synthetic | Real | Synthetic | Real | Synthetic |
| 5 | Real | | Synthetic | Synthetic | Synthetic | Real | Real | Real | Real | Real | Synthetic | Real | Synthetic | Synthetic | Synthetic |
| 6 | Real | | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Real | Real | Synthetic | Synthetic | Real | Real | Real |
| 7 | Synthetic | | Real | Real | Synthetic | Real | Synthetic | Synthetic | Synthetic | Real | Real | Real | Real | Real | Real |
| 8 | Real | | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Real | Synthetic | Real | Synthetic |
| 9 | Real | | Synthetic | Synthetic | Real | Real | Real | Synthetic | Synthetic | Synthetic | Real | Real | Synthetic | Real | Real |
| 10 | Synthetic | | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Real | Real |
| 11 | Synthetic | | Real | Real | Real | Synthetic | Real | Real | Real | Synthetic | Real | Real | Real | Real | Real |
| 12 | Real | | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Real | Synthetic | Synthetic | Real |
| 13 | Real | | Synthetic | Synthetic | Synthetic | Real | Synthetic | Synthetic | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Real |
| 14 | Synthetic | | Synthetic | Real | Synthetic | Synthetic | Synthetic | Real | Real | Synthetic | Real | Real | Synthetic | Real | Synthetic |
| 15 | Synthetic | | Real | Real | Real | Synthetic | Synthetic | Real | Synthetic | Real | Real | Real | Real | Real | Real |
| 16 | Synthetic | | Real | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Real | Real | Real | Real | Synthetic |
| 17 | Real | | Synthetic | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic |
| 18 | Real | | Real | Real | Real | Real | Real | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic |
| 19 | Real | | Real | Real | Real | Real | Real | Synthetic | Real | Synthetic | Synthetic | Real | Real | Synthetic | Real |
| 20 | Real | | Synthetic | Synthetic | Real | Synthetic | Real | Synthetic | Real | Real | Real | Synthetic | Synthetic | Synthetic | Synthetic |
| 21 | Real | | Real | Synthetic | Real | Synthetic | Synthetic | Synthetic | Real | Synthetic | Real | Real | Synthetic | Real | Real |
| 22 | Real | | Real | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Real | Real | Real | Synthetic |
| 23 | Synthetic | | Real | Real | Synthetic | Real | Real | Real | Real | Real | Real | Real | Synthetic | Synthetic | Real |
| 24 | Real | | Real | Real | Real | Synthetic | Synthetic | Synthetic | Real | Real | Real | Real | Real | Synthetic | Real |
| 25 | Real | | Real | Real | Real | Real | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic |
| 26 | Synthetic | | Real | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Real | Synthetic |
| 27 | Synthetic | | Real | Synthetic | Real | Synthetic | Real | Real | Synthetic | Real | Real | Real | Real | Real | Synthetic |
| 28 | Synthetic | | Real | Real | Real | Real | Real | Synthetic | Real | Real | Synthetic | Real | Synthetic | Real | Synthetic |
| 29 | Synthetic | | Real | Real | Real | Real | Synthetic | Real | Real | Real | Real | Real | Real | Real | Synthetic |
| 30 | Synthetic | | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic |
| 31 | Real | | Synthetic | Real | Real | Real | Real | Synthetic | Synthetic | Synthetic | Real | Synthetic | Synthetic | Real | Synthetic |
| 32 | Synthetic | | Real | Real | Real | Synthetic | Real | Real | Real | Synthetic | Real | Real | Real | Real | Synthetic |
| 33 | Real | | Synthetic | Real | Synthetic | Real | Real | Synthetic | Real | Synthetic | Synthetic | Real | Synthetic | Real | Synthetic |
| 34 | Synthetic | | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Synthetic | Synthetic | Real | Real | Real |
| 35 | Real | | Synthetic | Synthetic | Real | Synthetic | Synthetic | Real | Real | Real | Real | Real | Synthetic | Synthetic | Real |
| 36 | Real | | Real | Real | Real | Real | Real | Real | Synthetic | Real | Real | Synthetic | Synthetic | Real | Real |
| 37 | Synthetic | | Real | Real | Real | Synthetic | Synthetic | Real | Real | Real | Real | Real | Real | Real | Real |
| 38 | Synthetic | | Real | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic |
| 39 | Real | | Real | Synthetic | Real | Synthetic | Real | Real | Real | Real | Real | Real | Synthetic | Synthetic | Real |
| 40 | Synthetic | | Synthetic | Real | Real | Synthetic | Real | Synthetic | Real | Real | Synthetic | Real | Real | Real | Real |
| 41 | Synthetic | | Real | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Real | Synthetic | Synthetic | Real |
| 42 | Synthetic | | Real | Real | Synthetic | Synthetic | Synthetic | Synthetic | Real | Synthetic | Real | Synthetic | Real | Synthetic | Synthetic |

**Figure B.1:** All responses to the survey. Participants are shown in columns, experience level is on top. Green means correct answer, red means incorrect. Left-most column shows question number and ground truth.

# Bibliography


[1] S. W. Jahn, M. Plass, and F. Moinfar, "Digital pathology: Advantages, limitations and emerging perspectives," *Journal of Clinical Medicine*, vol. 9, no. 11, p. 3697, Nov. 2020, ISSN: 2077-0383. DOI: 10.3390/jcm9113697. [Online]. Available: http://dx.doi.org/10.3390/jcm9113697.

[2] M. T. Eadon, T.-H. Schwantes-An, C. L. Phillips, *et al.*, "Kidney histopathology and prediction of kidney failure: A retrospective cohort study," *American Journal of Kidney Diseases*, vol. 76, no. 3, pp. 350–360, Sep. 2020, ISSN: 0272-6386. DOI: 10.1053/j.ajkd.2019.12.014. [Online]. Available: http://dx.doi.org/10.1053/j.ajkd.2019.12.014.

[3] G. W. Gill, "H&e stain," in *Cytopreparation: Principles & Practice*. New York, NY: Springer New York, 2013, pp. 207–216, ISBN: 978-1-4614-4933-1. DOI: 10.1007/978-1-4614-4933-1_12. [Online]. Available: https://doi.org/10.1007/978-1-4614-4933-1_12.

[4] C. Dunn, D. Brettle, M. Cockroft, E. Keating, C. Revie, and D. Treanor, "Quantitative assessment of h&e staining for pathology: Development and clinical evaluation of a novel system," *Diagnostic Pathology*, vol. 19, no. 1, Feb. 2024, ISSN: 1746-1596. DOI: 10.1186/s13000-024-01461-w. [Online]. Available: http://dx.doi.org/10.1186/s13000-024-01461-w.

[5] F. M. Howard, J. Dolezal, S. Kochanny, *et al.*, "The impact of site-specific digital histology signatures on deep learning model accuracy and bias," *Nature Communications*, vol. 12, no. 1, Jul. 2021, ISSN: 2041-1723. DOI: 10.1038/s41467-021-24698-1. [Online]. Available: http://dx.doi.org/10.1038/s41467-021-24698-1.

[6] *AMA Journal of Ethics*, vol. 18, no. 8, pp. 817–825, Aug. 2016, ISSN: 2376-6980. DOI: 10.1001/journalofethics.2016.18.8.stas1-1608. [Online]. Available: http://dx.doi.org/10.1001/journalofethics.2016.18.8.stas1-1608.

[7] S. I. Nikolenko, *Synthetic Data for Deep Learning*. Springer International Publishing, 2021, ISBN: 9783030751784. DOI: 10.1007/978-3-030-75178-4. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-75178-4.

[8] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, 2021. arXiv: 2112.10752 [cs.CV].

[9] C. L. Srinidhi, O. Ciga, and A. L. Martel, "Deep neural network models for computational histopathology: A survey," *Medical Image Analysis*, vol. 67, p. 101 813, Jan. 2021, ISSN: 1361-8415. DOI: 10.101

6/j.media.2020.101813. [Online]. Available: http://dx.doi.org/10.1016/j.media.2020.101813.

[10] V. A. Fajardo, D. Findlay, C. Jaiswal, *et al.*, "On oversampling imbalanced data with deep conditional generative models," *Expert Systems with Applications*, vol. 169, p. 114 463, 2021, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114463. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420311155.

[11] OpenAI, *Chatgpt: Language model*, Accessed: 2024-06-19, 2023. [Online]. Available: https://www.openai.com/chatgpt.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661.

[13] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, *Jukebox: A generative model for music*, 2020. arXiv: 2005.00341.

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, *Improved techniques for training gans*, 2016. arXiv: 1606.03498 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1606.03498.

[15] A. B. Levine, J. Peng, D. Farnell, *et al.*, "Synthesis of diagnostic quality cancer pathology images by generative adversarial networks," *The Journal of Pathology*, vol. 252, no. 2, pp. 178–188, Sep. 2020, ISSN: 1096-9896. DOI: 10.1002/path.5509. [Online]. Available: http://dx.doi.org/10.1002/path.5509.

[16] Q. Zhou and H. Yin, "A u-net based progressive gan for microscopic image augmentation," in *Medical Image Understanding and Analysis*, G. Yang, A. Aviles-Rivero, M. Roberts, and C.-B. Schönlieb, Eds., Cham: Springer International Publishing, 2022, pp. 458–468, ISBN: 978-3-031-12053-4.

[17] Z. Xiao, K. Kreis, and A. Vahdat, *Tackling the generative learning trilemma with denoising diffusion gans*, 2022. arXiv: 2112.07804.

[18] A. Brock, J. Donahue, and K. Simonyan, *Large scale gan training for high fidelity natural image synthesis*, 2019. arXiv: 1809.11096.

[19] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, *Flow matching for generative modeling*, 2023. arXiv: 2210.02747.

[20] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[21] P. Dhariwal and A. Nichol, *Diffusion models beat gans on image synthesis*, 2021. arXiv: 2105.05233 [cs.LG].

[22] OpenAI, A. Ramesh, M. Pavlov, G. Goh, and S. Gray, *Dall·e: Creating images from text*, Accessed: 2024-06-19, 2021. [Online]. Available: https://openai.com/index/dall-e/?stream=science.

[23] T. Brooks, B. Peebles, C. Holmes, *et al.*, "Video generation models as world simulators," 2024. [Online]. Available: https://openai.com/research/video-generation-models-as-world-simulators.

[24] A. Nichol and P. Dhariwal, *Improved denoising diffusion probabilistic models*, 2021. arXiv: 2102.09672.

[25] P. A. Moghadam, S. V. Dalen, K. C. Martin, *et al.*, *A morphology focused diffusion probabilistic model for synthesis of histopathology images*, 2022. arXiv: 2209.13167.

[26] G. Müller-Franzes, J. M. Niehues, F. Khader, *et al.*, "A multimodal comparison of latent denoising diffusion probabilistic models and generative adversarial networks for medical image synthesis," *Scientific Reports*, vol. 13, no. 1, Jul. 2023, ISSN: 2045-2322. DOI: 10.1038/s41598-023-39278-0. [Online]. Available: http://dx.doi.org/10.1038/s41598-023-39278-0.

[27] R. Harb, T. Pock, and H. Müller, "Diffusion-based generation of histopathological whole slide images at a gigapixel scale," *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 5119–5128, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265157934.

[28] S. Yellapragada, A. Graikos, P. Prasanna, T. Kurc, J. Saltz, and D. Samaras, "Pathldm: Text conditioned latent diffusion model for histopathology," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 5170–5179. DOI: 10.1109/WACV57701.2024.00510.

[29] C. Luo, *Understanding diffusion models: A unified perspective*, 2022. arXiv: 2208.11970 [cs.LG].

[30] Wikipedia contributors, *Allegory of the cave — Wikipedia, the free encyclopedia*, [Online; accessed 10-June-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Allegory_of_the_cave&oldid=1226543546.

[31] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[32] Wikipedia contributors, *Jensen's inequality — Wikipedia, the free encyclopedia*, [Online; accessed 10-June-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Jensen%27s_inequality&oldid=1212437916.

[33] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: 1312.6114 [stat.ML].

[34] D. Bouchacourt, R. Tomioka, and S. Nowozin, *Multi-level variational autoencoder: Learning disentangled representations from grouped observations*, 2017. arXiv: 1705.08841 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1705.08841.

[35] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, *Improving variational inference with inverse autoregressive flow*, 2017. arXiv: 1606.04934 [cs.LG].

[36] D. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational diffusion models," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 21 696–21 707. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/b578f2a52a0229873fefc2a4b06377fa-Paper.pdf.

[37] C. Saharia, W. Chan, S. Saxena, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *ArXiv*, vol. abs/2205.11487, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248986576.

[38] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *ArXiv*, vol. abs/2204.06125, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248097655.

[39] J. Ho and T. Salimans, *Classifier-free diffusion guidance*, 2022. arXiv: 2207.12598 [id='cs.LG'].

[40] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019, ISSN: 1935-8245. DOI: 10.1561/2200000056. [Online]. Available: http://dx.doi.org/10.1561/2200000056.

[41] S. Abousamra, R. Gupta, L. Hou, *et al.*, "Deep learning-based mapping of tumor infiltrating lymphocytes in whole slide images of 23 types of cancer," en, *Front. Oncol.*, vol. 11, p. 806 603, 2021.

[42] W. Lingle, B. J. Erickson, M. L. Zuley, *et al.*, *The cancer genome atlas breast invasive carcinoma collection (tcga-brca)*, 2016. DOI: 10.7937/K9/TCIA.2016.AB2NAZRP. [Online]. Available: https://www.cancerimagingarchive.net/collection/tcga-brca/.

[43] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597.

[44] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[45] Z. Huang, F. Bianchi, M. Yuksekgonul, T. J. Montine, and J. Zou, "A visual–language foundation model for pathology image analysis using medical twitter," *Nature Medicine*, pp. 1–10, 2023.

[46] A. Radford, J. W. Kim, C. Hallacy, *et al.*, *Learning transferable visual models from natural language supervision*, 2021. arXiv: 2103.00020.

[47] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: 1711.05101.

[48] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980.

[49] J. Song, C. Meng, and S. Ermon, *Denoising diffusion implicit models*, 2022. arXiv: 2010.02502.

[50] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, Apr. 1960, ISSN: 1552-3888. DOI: 10.1177/001316446002000104. [Online]. Available: http://dx.doi.org/10.1177/001316446002000104.

[51] K. H. Zou, S. K. Warfield, A. Bharatha, *et al.*, "Statistical validation of image segmentation quality based on a spatial overlap index," *Academic Radiology*, vol. 11, no. 2, pp. 178–189, Feb. 2004, ISSN: 1076-6332. DOI: 10.1016/s1076-6332(03)00671-8. [Online]. Available: http://dx.doi.org/10.1016/s1076-6332(03)00671-8.

[52] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191.