Heidelberg University

Generating Breast Cancer Immunohistochemical Images with Deep Learning

Xichu Yu, Yuchen Li, Maiqi Zhou
Practical Biomedical Image Analysis

Summer Semester 2024
Supervisor: PD Dr. Karl Rohr
15.4. - 26.7.2024

July 23, 2024

1 Introduction and Motivation

Breast cancer continues to be a prominent cause of global female mortality. The key factor of accurate diagnosis and therapy is histopathological examination. This extensive examination begins with the creation of hematoxylin and eosin (HE) stained slices from tumor samples, illustrated in Figure 1 (Left). These slices are meticulously evaluated by pathologists, either under a microscope or via a digitized whole-slide imaging (WSI) system.

A critical part of any breast cancer diagnosis is the detailed analysis of key proteins such as the human epidermal growth factor receptor 2 (HER2). Overexpression of HER2 is characteristic of certain aggressive forms of breast cancer. Therefore, its accurate detection is imperative for the formulation of appropriate treatment strategies. HER2 expression is typically assessed via immunohistochemical (IHC) techniques which involve the preparation of an IHC-stained slice, as demonstrated in Figure 1 (Right), and subsequent inspection by pathologists to determine HER2 expression levels. [1]

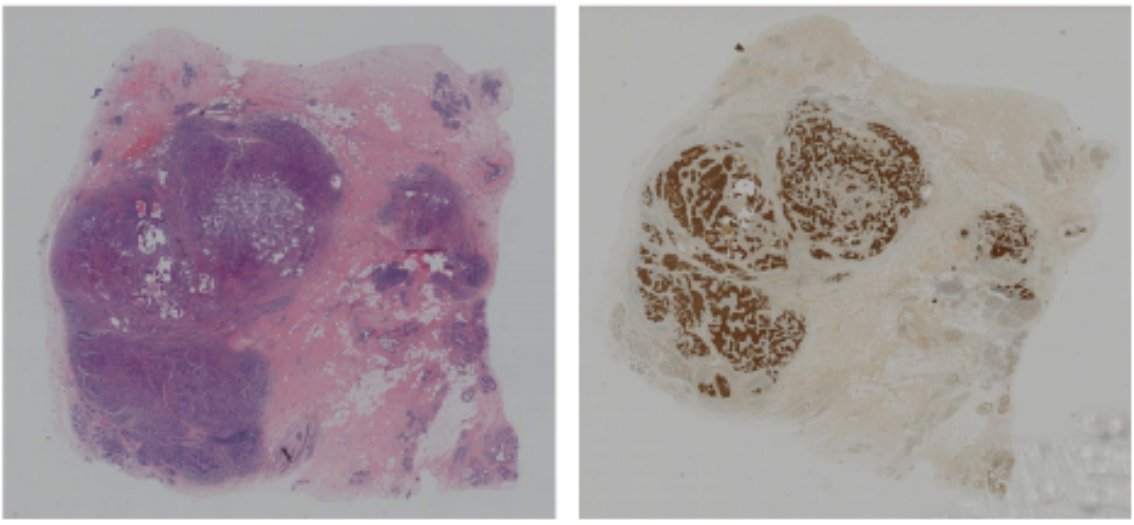

Figure 1: (Left) HE stained slice and (Right) IHC stained slice. [2].

However, this approach presents certain limitations. Preparing an IHC-stained section is a costly and time-consuming endeavor. Furthermore, the inherent heterogeneity of tumors poses a significant challenge as the standard practice of IHC staining typically encompasses only a single pathological section. Consequently, this could potentially result in an inadequate representation of the tumor's actual state [2].

To address these challenges, we developed several deep-learning-based methods to translate images from Hematoxylin and Eosin (HE) to Immunohistochemistry (IHC) domain, reducing the need for costly IHC staining and revealing tumor heterogeneity. We implemented pix2pix-based [3] and diffusion-based [4] models. In addition, we followed the method of challenge winner in [5] to filter the provided dataset using open-source pathological software HistoQC. We also developed user-friendly software based on QuPath, open-source software for digital pathology and whole slide image analysis. The software works as an extension that can connect the proposed models and the QuPath and present the converted IHC image. We evaluated the generation results on 3 commonly used metrics and thoroughly discussed the experimental results and findings during this project.

2 Related Work

2.1 HE to IHC Image Conversion

2.1.1 Pyramid Pix2pix Model

The Pyramid Pix2pix model uses a unique image transformation method, which involves applying the same scale transformation to the generated image and ground truth. The transformation process

includes two key steps: 1) using a Gaussian kernel with a standard deviation of 1 as a low-pass filter to smooth the image, 2) downsampling the smoothed image to reduce its resolution and eliminate redundant pixels. As the Gaussian filtering process proceeds, the image becomes increasingly blurry, which is a necessary step for subsequent transformations.

For each resolution level or octave, the model performs multiple Gaussian convolutions to achieve scale transformation. The Pyramid pix2pix model is designed with multiple octaves. The first layer of each octave is obtained by downsampling the last image of the previous octave. Each octave consists of 5 layers and undergoes 4 Gaussian blurs. Each output of the octave is defined as a scale.

In Gaussian pyramid, the first layer image of each octave is extracted to calculate the loss. The loss at each scale is denoted as Si (where i = 1, 2, 3, etc.). This innovative approach enables the Pyramid Pix2pix model to effectively address the unique challenges posed by the breast cancer immunohistochemistry (BCI) dataset [5], demonstrating the potential of deep learning in medical image generation and analysis.

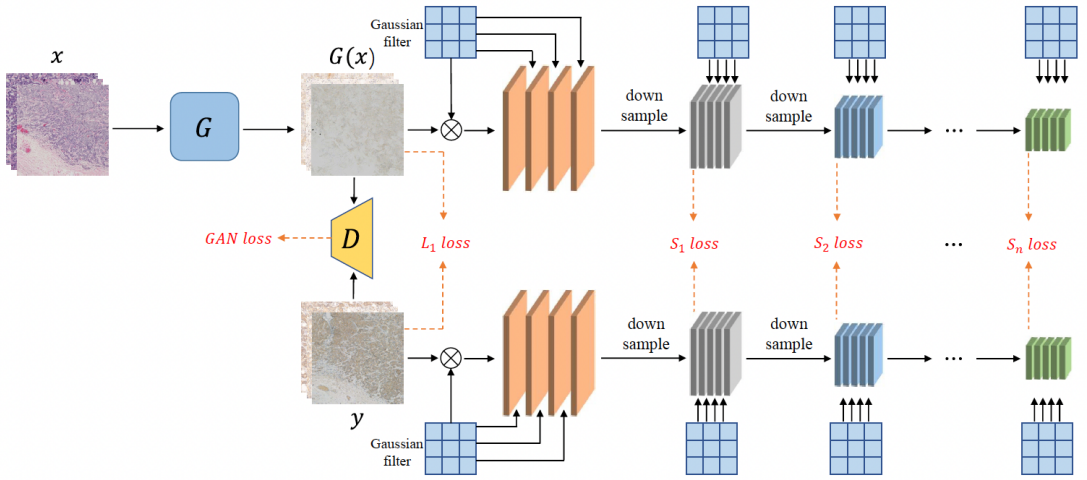

Figure 2: The framework of the Pyramid pix2pix. [1]

2.1.2 Review of BCI Challenge

In [5], Zhu et al. summarized the participant models and presented further analysis for each model.

The team arptidec5 built the solution with the Pyramid pix2pix model, which is a type of conditional generative adversarial network (GAN). They extended the method by filtering the image pairs in the dataset. Before the images were used as input to the model for training, they went through a quality control process to ensure images with artifacts, cracked tissue, or blurriness were removed from the training process.

The team Just4Fun's solution was constructed utilizing the architecture of GAN. They employed an encoder to extract features from HE images for classification purposes. The classification outcome serves as guidance for the generator within the framework.

The team lifangda02 implemented the resnet-9-blocks generator in their solution, incorporating an innovative paired InfoNCE [6] contrastive loss, which extends the InfoNCE loss. In this approach, an output patch serves as the query, with the corresponding IHC-stained patch designated as the positive sample, while non-corresponding patches are designated as negatives.

The team stan9 used a weakly supervised deep generative network called WeCREST. Their method involves a multi-class discriminator to distinguish between real and synthetic image styles and incorporates a classifier that operates on the discriminator to sort images based on the HER2 expression status, labeled according to the corresponding IHC stained images.

The team vived23 employed a conditional generative adversarial network (cGAN) based on the paired images. They applied the discrete wavelet transform to extract 4 channels of spatial and frequency domain features from HE images.

Team	Final Rank	PSNR(dB)/Rank	SSIM/Rank
arpitdec5	1	19.736 / 2	0.574 / 1
Just4Fun	2	22.929 / 1	0.559 / 2
lifangda02	3	17.927 / 5	0.555 / 3
stan9	4	17.959 / 4	0.543 / 4
guanxianchao[1]	5	19.560 / 3	0.497 / 5
vivek23	6	15.271 / 6	0.493 / 6

Figure 3: Results of participating teams. (Team guanxianchao didn't submit their method.) [5]

2.1.3 Parallel Studies

A similar task is discussed in [7]. The authors proposed a novel multitask dual diffusion architecture for virtual staining. The proposed method was inspired by the success of multi-task deep learning models with limited dataset size. The method leveraged the affinity between similar tasks of segmentation and staining (i.e. HE to IHC conversion) to achieve high generation quality with limited training samples. However, the proposed method utilized a non-open dataset with segmentation information. And the method is not open-source either. As a result, we did not put much effort into this direction.

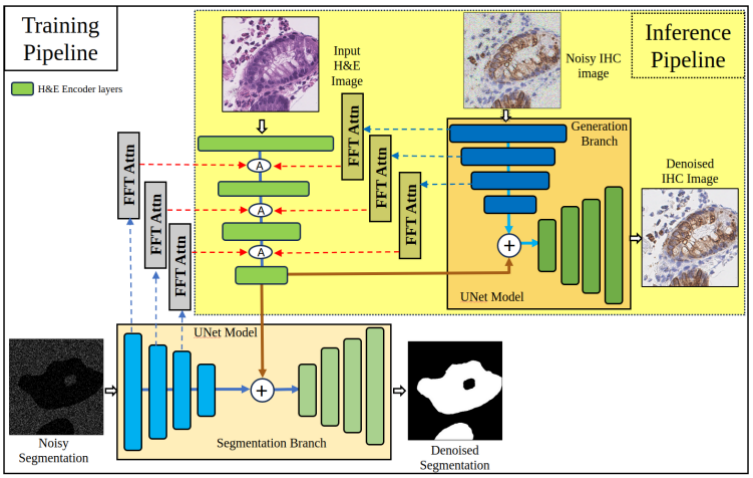

Figure 4: The architecture of StainDiffuser proposed in [7]

2.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs)[3] have since become one of the most popular methods in generative modeling. GANs consist of two neural networks, a generator and a discriminator, which are trained simultaneously through an adversarial process. The generator creates synthetic data samples, while the discriminator evaluates them against real data samples. The generator aims to produce data that is indistinguishable from real data, while the discriminator strives to correctly classify the input data as real or fake. This adversarial training continues until the generator produces high-quality data that the discriminator cannot reliably distinguish from real data. The training process of GANs involves the following steps:

1. **Generator Training**: The generator takes random noise as input and generates synthetic data. The objective of the generator is to produce data that the discriminator classifies as real. The loss function for the generator is:

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

where $G(z)$ is the generated data and $D(G(z))$ is the discriminator's probability that the generated data is real.

2. **Discriminator Training**: The discriminator receives both real and synthetic data and learns to classify them correctly. The loss function for the discriminator is:

$$\mathcal{L}_D = -\left[\log(D(x)) + \log(1 - D(G(z)))\right]$$

where x is the real data and $G(z)$ is the generated data.

3. **Adversarial Process**: The generator and discriminator are trained alternately, with the generator improving its data generation capabilities while the discriminator enhances its classification accuracy.

GANs have been successfully applied in various domains, including image synthesis, image translation, and data augmentation.
Based on the foundational GAN architecture, Conditional GANs (cGANs)[8] introduce an additional layer of control over the data generation process. CGANs condition both the generator and the discriminator on some extra information, such as class labels or data from other modalities. This conditioning allows the model to generate data that adheres to specific requirements or constraints, making cGANs particularly useful in applications where controlled output is essential. The architecture of cGANs involves the following modifications:

1. **Conditional Inputs**: Both the generator and the discriminator receive an additional input that provides conditional information. For example, in image generation tasks, this input could be the class label of the image to be generated.

2. **Generator Loss Function**: The generator's loss function is modified to incorporate the conditional input:

$$\mathcal{L}_G = -\log(D(G(z|y)|y))$$

where y is the conditional input.

3. **Discriminator Loss Function**: Similarly, the discriminator's loss function is also modified to consider the conditional input:

$$\mathcal{L}_D = -\left[\log(D(x|y)) + \log(1 - D(G(z|y)|y))\right]$$

cGANs have been employed successfully for tasks such as image-to-image translation and text-to-image synthesis. For example, in image-to-image translation, cGANs can convert images from one domain to another based on the given condition, such as transforming sketches into realistic photographs. By incorporating conditional inputs, cGANs improve the flexibility and capability of GANs in more diverse condition.

2.3 Pix2Pix

Pix2Pix[3] is a conditional generative adversarial network for image-to-image translation tasks. The Pix2Pix framework improve the performance of GANs by conditioning both the generator and discriminator on input images, enabling the generated output images could fulfill specific input constraints. This framework has been successfully applied to various image-to-image translation problems, including image colorization, semantic segmentation and style transfer.
Generator
The generator in Pix2Pix is based on the U-Net architecture, which consists of an encoder-decoder structure with skip connections between corresponding layers of the encoder and decoder. The key components of the generator are:

1. **Encoder**: The encoder compresses the input image into a lower-dimensional representation through a series of downsampling layers. Each layer consists of a 2D convolutional layer followed by a LeakyReLU activation, with batch normalization applied to all but the first layer. These layers progressively reduce the spatial dimensions while increasing the depth, capturing high-level features at different scales.

2. **Skip Connections**: Skip connections are used between corresponding layers of the encoder and decoder. These connections concatenate feature maps from the encoder to the decoder, preserving spatial information and enabling the generator to produce more detailed and accurate images.

3. **Decoder**: The decoder reconstructs the image from the compressed representation using a series of upsampling layers. Each upsampling layer includes an upsampling operation, a 2D convolutional layer, batch normalization, and a ReLU activation function, with dropout applied to the first three layers. These layers progressively increase the spatial dimensions, allowing the network to reconstruct the image from the learned features.

Discriminator The discriminator in Pix2Pix uses a PatchGAN architecture, which classifies the input image as real or fake rather in pixel-level than classifying the entire image. This approach helps the discriminator focus on high-frequency details and textures, leading to more realistic generated images. The key components of the discriminator are:

1. **Downsampling Layers**: The discriminator consists of several downsampling layers to progressively reduce the spatial dimensions while increasing the depth.

2. **Final Layer**: The final layer uses a sigmoid activation function to classify the patches as real or fake.

Pix2Pix has been successfully applied to various image-to-image translation problems, such as image colorization and semantic segmentation. By conditioning on input images and using skip connections, Pix2Pix generates high-quality and detailed images that closely resemble the target outputs. Also, the PatchGAN discriminator improves the learning of high-frequency details, resulting in more realistic textures and finer details in the generated images. In this case, pix2pix has become a foundational model in the field of image-to-image translation.

2.4 Attention mechanism

Attention mechanisms[9] have become a pivotal innovation in the field of deep learning, particularly in tasks involving sequential data and image processing, which enables models to focus on relevant parts of the input data dynamically.

Self-attention computes the representation of a sequence by relating different positions within that sequence. This allows the model to focus on relevant parts of the input when producing a given output, making it especially powerful for tasks requiring an understanding of long-range dependencies. The self-attention mechanism can be described by the following steps:

1. **Input Transformation**: The input sequence is transformed into three matrices: queries (Q), keys (K), and values (V), through learned linear projections.

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

   where X is the input sequence, and W^Q, W^K, W^V are the projection matrices.

2. **Attention Scores**: The attention scores are computed by taking the dot product of the query with all keys, scaling by the square root of the dimension of the keys (d_k), and applying a softmax function to obtain the weights.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

3. **Weighted Sum**: The resulting weights are used to create a weighted sum of the values, producing the final attention output.

In recent years, attention mechanisms are widely used in image processing problems such as image segmentation and image translation. For image segmentation, by focusing on critical parts of an image, attention mechanisms can enhance the performance of segmentation and detection tasks, making it easier to identify and delineate objects within an image. Also, it can dynamically highlight important regions of an image, allowing the network to focus on relevant features and improving the quality of generated images.

2.5 Diffusion models

GANs [10] is universally utilized in the previous work of the challenge organizers [1] and chosen by all challenge participants as the backbone [5]. Nevertheless, studies have demonstrated that GANs capture less diversity compared to the best likelihood-based models. [11] Additionally, training GANs can be challenging due to the problem of instability. [11] In contrast to GAN [10], diffusion probabilistic models have emerged as a powerful framework for generative modeling, which offers a new approach to synthesizing data by iteratively refining samples through a diffusion process. This section reviews the key advancements and research that have contributed to the development of the diffusion model and several variations related to our method.

2.5.1 Diffusion Probabilistic Model

The concept of diffusion processes has its roots in statistical physics and stochastic processes, with early applications in modeling physical phenomena such as heat distribution and particle motion. In the realm of machine learning, these processes have been adapted to develop new probabilistic models that generate data through iterative refinement. Sohl-Dickstein et al. were among the first to introduce diffusion probabilistic models in the context of generative modeling [12]. Their work demonstrated the feasibility of using diffusion processes to generate data, providing a novel perspective on probabilistic model design.

$$p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:N})\, d\mathbf{x}_{1:N} \tag{1}$$

where:

- $p_\theta(\mathbf{x}_0)$ is the marginal probability distribution of the variable $\mathbf{x}_0$.
- $\mathbf{x}_1, \ldots, \mathbf{x}_T$ are latents of the same dimensionality as the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$.

The joint distribution $p_\theta(\mathbf{x}_{0:N})$ is called the reverse process which is defined as follows:

$$p_\theta(\mathbf{x}_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{2}$$

The corresponding forward process is fixed to a Markov chain that gradually adds Gaussian noise to the input, which is defined as:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{3}$$

The forward process utilizes a gradually increasing variance schedule $\beta_1, \ldots, \beta_T$. The goal is to reverse this process by predicting the noise added in each step with learned parameters θ. The training objective is to optimize the following variational bound on negative log-likelihood:

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}\right] = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})}\right] =: L \tag{4}$$

Following the foundational work, Ho et al. (2020) introduced Denoising Diffusion Probabilistic Models

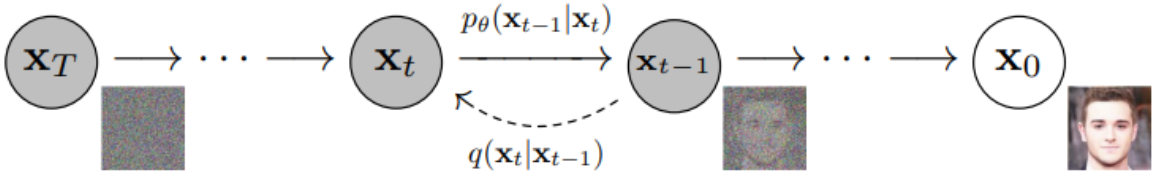

Figure 5: Forward and reverse process visualized with image [4]

(DDPM) [4], which refined the training process and achieved state-of-the-art results in image generation. They showed that if $\alpha_n := 1 - \beta_n$ and $\bar{\alpha}_n$ represents the cumulative product $\prod_{i=1}^{n} \alpha_i$, we can obtain:

$$q(\mathbf{x}^n \mid \mathbf{x}^0) = \mathcal{N}\left(\mathbf{x}^n; \sqrt{\bar{\alpha}_n}\mathbf{x}^0, (1 - \bar{\alpha}_n)\mathbf{I}\right). \tag{5}$$

Further, for the parametrization for p_θ in equation (2), they choose:

$$\mu_\theta(\mathbf{x}^n, n) = \frac{1}{\sqrt{\alpha_n}}\left(\mathbf{x}^n - \frac{\beta_n}{\sqrt{1 - \bar{\alpha}_n}}\epsilon_\theta(\mathbf{x}^n, n)\right) \tag{6}$$

where ϵ_θ denotes a neural network which predicts $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ from $\mathbf{x}^n$.

Finally, the training objective is simplified to:

$$\mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{\beta_n^2}{2\sigma_\theta\alpha_n(1 - \bar{\alpha}_n)}\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_n}\mathbf{x}^0 + \sqrt{1 - \bar{\alpha}_n}\epsilon, n\right)\right\|^2\right] \tag{7}$$

Diffusion probabilistic models have been extended in later research. In [13], Nichol and Dhariwal proposed an improved noise schedule and sampling procedure, making the sampling more efficient and proved that DDPMs [4] can achieve better log-likelihoods with little impact on the quality of output. Additionally, Song et al. explored score-based generative models in [14]. They used a stochastic differential equation (SDE) that transforms data distributions into noise and back, leveraging neural networks to estimate score functions and achieved improved generation performance.

Song et al. [15] proposed another kind of diffusion probabilistic model named Denoising Diffusion Implicit Model (DDIM) which takes a deterministic reverse process in contrast to 2:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1 - \alpha_t}\epsilon_\theta^t(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right) + \sqrt{1 - \alpha_{t-1}}\epsilon_\theta^t(\mathbf{x}_t) \tag{8}$$

Accordingly, the proposed forward distribution is defined as:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1 - \alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1 - \alpha_t}}\right), 0\right) \tag{9}$$

In reverse steps in DDIM, the variance in each step is zero, making the process deterministic. It is also proved that the original marginal distribution in DDPM [4] is maintained and the sample efficiency is further improved. In addition, the deterministic nature of DDIM broadens the use case of DPMs which will be discussed in the next section.

2.5.2 Diffusion Autoencoder

Diffusion Autoencoders (DiffAE) [16] is an innovative approach in the field of generative modeling. It takes use of the strengths of diffusion probabilistic models (DPMs) to create meaningful and decodable representations. Unlike traditional DPMs, which often struggle with the interpretability of latent variables, DiffAE aims to produce high-quality generative outputs while also encoding useful and semantic information for downstream tasks. Based on the deterministic character of DDIM [15], the noise map $\mathbf{x}_t$ is fixed and can be regarded as an encoding of a given input data $\mathbf{x}_0$. This process of adding noise and denoising is similar to the encoding and decoding. In this context, the authors proposed the Diffusion Autoencoder as the Figure 6 displays. The authors implemented a conditional diffusion model conditioned with $\mathbf{z}_{sem}$ where the $\mathbf{z}_{sem}$ is the output of a semantic encoder $Enc_\phi(\mathbf{x}_0)$. The equation 6 will now become:

$$\mu_\theta(\mathbf{x}_t, t, \mathbf{z}_{\text{sem}}) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \sqrt{1 - \alpha_t}\epsilon_\theta(\mathbf{x}_t, t, \mathbf{z}_{\text{sem}})\right) \tag{10}$$

By utilizing a semantically meaningful representation to guide the generation process, the diffusion autoencoder can achieve accurate reconstruction and style manipulation in latent space similar to StyleGAN [17] as Figure 7.

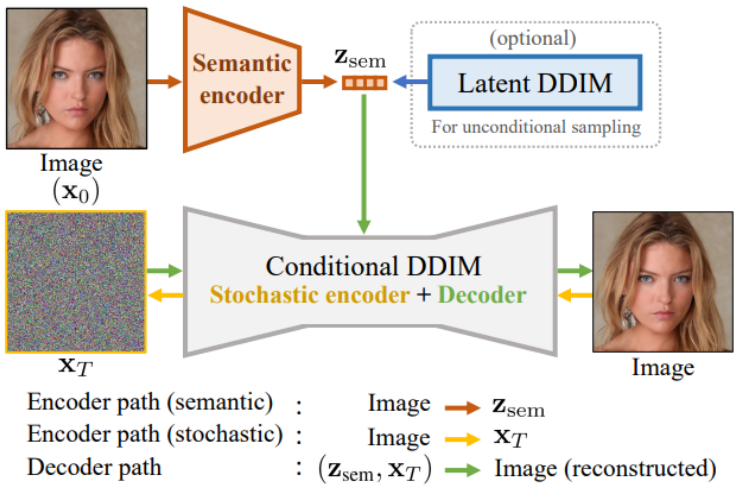

Figure 6: Architecture of Diffusion Autoencoder [16]

Figure 7: Style manipulation of two real images with DiffAE. [16]

3 Method and System

3.1 Pix2pix and Variant

3.1.1 Pix2pix(Attention U-Net)

Our modified model is based on the U-Net based Pix2Pix architecture, while the discriminator follows a convolutional PatchGAN classifier. In order to enhance the model's ability to focus on relevant image features, we integrate attention blocks into the existing U-Net structure.

Generator: The generator network is designed to convert input images to output images through 8 downsampling layers and 8 upsampling layers. Each downsampling layer consists of a 2D convolutional layer followed by a LeakyReLU activation, with batch normalization applied to all but the first layer. These layers progressively reduce the spatial dimensions of the input while increasing the depth, capturing high-level features at different scales. The upsampling layers include an upsampling operation followed by a 2D convolutional layer, batch normalization, and ReLU activation, with dropout applied to the first three layers. The upsampling layers progressively increase the spatial dimensions, allowing the network to reconstruct the image from the learned features. Skip connections are used between corresponding downsampling and upsampling layers to preserve spatial information.

Attention block: In the generator, attention blocks were applied at each skip connection in the upsampling path, specifically after the upsampling and before the concatenation with the corresponding downsampled feature map. The attention blocks consist of the following steps:

1. **Input Transformation**: Two separate convolutional layers transform the input feature maps from the skip connection and the upsampled path into a lower-dimensional space. This transformation ensures the inputs are compatible for subsequent operations.

2. **Feature Combination and Activation**: The transformed feature maps are combined using an element-wise addition, followed by a ReLU activation function. This step generates an intermediate feature map that captures the combined information from both inputs.

3. **Attention Map Generation**: A convolutional layer followed by a sigmoid activation function

generates the attention map. The attention map highlights important regions by assigning higher weights to relevant features.

4. **Feature Modulation**: The original skip connection features are modulated by the attention map through element-wise multiplication. This modulation enhances the important features while suppressing less relevant ones.

Discriminator: The discriminator employs a PatchGAN architecture with three downsampling layers and a final sigmoid activation to classify image patches as real or fake.
By introducing attention blocks, our model could dynamically highlight important regions in the feature maps, improving the generator's capability to produce high-quality images. The enhanced focus on salient features allows for more detailed and accurate image generation, leveraging the strengths of both the U-Net architecture and the attention mechanisms.

3.1.2 Pix2pix(TransUNet)

For this model, we tried to use TransUnet[18] as the generator structure of our model. The TransUnet model combines the strengths of convolutional neural networks (CNNs) and Transformers to enhance image generation quality. The architecture consists of a ResNet-based encoder, a Transformer module, and a decoder, forming an end-to-end network that leverages both local and global features.
Encoder: The encoder is based on a pretrained ResNet-50 model, which is divided into four stages:

1. **Stage 1**: Processes the input image using the initial ResNet layers, including 3 convolutional layers with batch normalization and ReLU activation.

2. **Stage 2**: Extracts deeper features using the next ResNet block, consisting of 4 convolutional layers with batch normalization and ReLU activation.

3. **Stage 3**: Further refines the features with another ResNet block, which includes 6 convolutional layers with batch normalization and ReLU activation.

4. **Stage 4**: Produces high-level features with the final ResNet block, containing 3 convolutional layers with batch normalization and ReLU activation.

ResNet-50 has robust feature extraction capabilities and residual connections, which help in training deeper networks by mitigating the vanishing gradient problem. **Transformer Module:** The Transformer module in TransUnet is a Vision Transformer (ViT) that operates on the high-level features extracted by the encoder. It consists of:

1. **Patch Embedding**: The high-level features are divided into patches and embedded into a sequence.

2. **Transformer Layers**: Multiple transformer layers (10 in this case), each consisting of a multi-head self-attention mechanism and a feed-forward neural network, process the embedded patches to capture global dependencies and contextual information.

3. **Output Reshaping**: The transformer output is reshaped back to a feature map and upsampled to match the spatial resolution required for the decoder.

Decoder: The decoder reconstructs the high-resolution image from the features processed by the Transformer. It consists of multiple upsampling and decoding layers:

1. **Upsampling Layers**: These layers increase the spatial dimensions progressively using bilinear interpolation and convolutional layers (4 upsampling layers).

2. **Decoder Blocks**: Each block refines the features using convolutional layers, batch normalization, and ReLU activation (4 decoder blocks, each with 2 convolutional layers).

3. **Skip Connections**: Integrates features from corresponding encoder stages to preserve spatial information.

Discriminator: The discriminator in the TransUnet model remains similar to the standard Pix2Pix discriminator we metioned previously.
TransUnet is a useful model that combinet the local and global features by integrating CNNs for local feature extraction and Transformers for capturing global dependencies.

3.1.3 Loss function

In the Pix2Pix model, to enhance the performance of the model, we use two types of combination of loss functions. Specifically, we utilize adversarial loss, L1 loss, and VGG loss.

Adversarial Loss and L1 Loss: Firstly, we use the popular combination of loss functions: adversarial loss, L1 loss. The total loss for the generator is a weighted sum of the adversarial loss and L1 loss. The fomulas of $L_{\text{GAN loss}}$ and L1_loss are as follow:

$$L_{\text{GAN loss}} = L_{\text{BCE}}(D(G(z)), 1)$$

where $D(G(z))$ is the discriminator's output for the generated images $G(z)$.

$$L_{\text{L1 loss}} = \|G(z) - y\|_1$$

where $G(z)$ are the generated images and y are the target images.

$$L_{\text{Total}} = L_{\text{GAN}} + 100 \times L_{\text{L1}}$$

VGG Loss: In addition, we introduce VGG Loss to capture perceptual differences between the generated and target images. This loss leverages a pretrained VGG19 network to extract feature representations at different layers. The VGG loss is computed as the weighted sum of the L1 losses between the feature maps of the generated and target images at multiple layers of the VGG19 network. The formula of L_{VGG} is as follow:

$$L_{\text{VGG}} = \sum_{i=1}^{5} w_i \| L_{\text{VGG}(G(z))_i} - L_{\text{VGG}(y)_i} \|_1$$

where $L_{\text{VGG}(G(z))_i}$ and $L_{\text{VGG}(y)_i}$ are the feature maps at layer i of the VGG19 network for the generated and target images, respectively, and w_i are the weights for each layer.

$$L_{\text{Total}} = L_{\text{GAN}} + 100 \times L_{\text{L1}} + 10 \times L_{\text{VGG}}$$

3.2 Diffusion Model

3.2.1 Diffusion Process

The original DDPM [4] is designed to be probabilistic and unconditional. The noising process is stochastic and involves a sequence of steps where each step gradually adds Gaussian noise to the input data until it becomes a sample from the standard Gaussian distribution. The reverse diffusion process aims to reconstruct the input data from the pure Gaussian noise by progressively denoising it. Since the noise z introduced in every sampling step is drawn from a random normal Gaussian distribution, the output would also be stochastic. Besides, the original DDPM only takes training samples in a single domain, and it only learns the distribution of the input domain. Consequently, it is not suitable for image conversion tasks like in this project. As a result, we choose DDIM [15] for this project. According to [15], the forward process and the reverse process are defined as follows.

Forward Process: The forward process in DDIM maintains the same structure as in DDPM but provides flexibility in the number of steps and the type of noise added. The process can be expressed as:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I}), \tag{11}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ is the cumulative product of $\alpha_s = 1 - \beta_s$.

Reverse Process: The reverse process in DDIM is designed to be deterministic, making it suitable for applications requiring consistent outputs. It is expressed as:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{1 - \alpha_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1}}\epsilon_\theta(\mathbf{x}_t, t), \tag{12}$$

where ϵ_θ is the predicted noise.

This formulation allows DDIM to perform efficient and flexible sampling, making it an ideal choice for the image conversion tasks in this project.

3.2.2 Conditional Mechanisms

The timestep t plays a crucial role in the diffusion process, as it indicates the progression of noise addition or removal at each step. To effectively incorporate the temporal information into the model, we use a sinusoidal encoding function as is used in [4] [15] [16]. This encoding provides a unique representation for each timestep, allowing the model to distinguish between different stages of the diffusion process. The sinusoidal encoding function is defined as follows:

$$\psi(t) = \left[\sin\left(\frac{t}{10000^{2i/d}}\right), \cos\left(\frac{t}{10000^{2i/d}}\right)\right]_{i=0}^{d-1}, \tag{13}$$

where d is the dimensionality of the encoding, and i is the position within the encoding vector. The sinusoidal encoding captures both low-frequency and high-frequency variations of the timestep, providing a rich representation that can be used by the model. This encoding is then incorporated into the noise prediction network ϵ_θ as additional input, ensuring that the model is aware of the specific timestep at each stage of the diffusion process.

In practice, the timestep embedding is concatenated with the input data or intermediate features within the neural network. This integration helps the model learn the temporal dynamics of the diffusion process more effectively, leading to improved performance in denoising and data generation tasks. By using sinusoidal timestep embeddings, we ensure that the model has access to precise temporal information, which is essential for the accurate reconstruction of the input data from noise.

To guide the generation process, we introduce auxiliary information $\mathbf{y}$, which could be class labels, textual descriptions, or any other relevant data. In this experiment, the setting of the condition will only be considered as an image. This conditional information is integrated into the model at each step of the diffusion process. Specifically, the conditional diffusion process can be described as:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{y}) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, \mathbf{y}, t), \Sigma_\theta(\mathbf{x}_t, \mathbf{y}, t)) \tag{14}$$

where $\mathbf{y}$ denotes the conditional data, namely an HE image to be converted. The first option is to integrate the condition by concatenating two images along the channel dimension. The input to the model is defined as $\mathbf{x_{in}} = \mathbf{Cat}(\mathbf{x}_{HE}, \mathbf{x}_{IHC})$. In that case, the model takes an input tensor of 6 channels to predict the noise. This naive approach of concatenating was used in the preliminary experiment but did not receive good results.

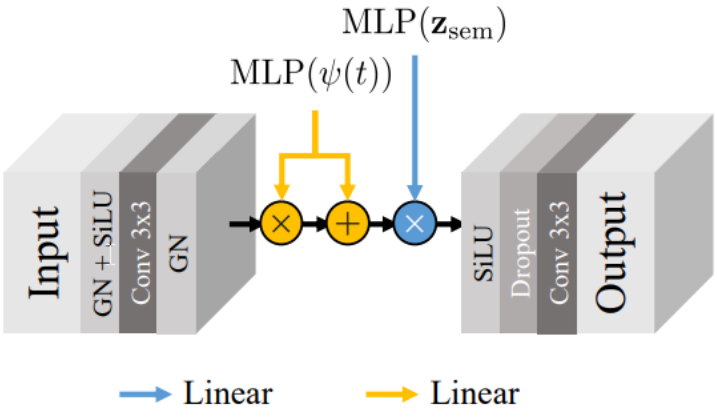

Figure 8: ResBlock + AdaGN in DiffAE[16]

Alternatively, we use the AdaGN mechanism introduced in [11] following DiffAE [16]. As is shown in Figure 8, $\psi(t)$ denotes the sinusoidal encoding and MLP is a multilayer perceptron consisting of several fully-connected layers. The time encoding and semantic encoding will be converted to one-dimensional embedding which will be applied to the input as follows:

$$\text{AdaGN}(\mathbf{h}, t, \mathbf{z}_{\text{sem}}) = \mathbf{z}_s(t_s \text{GroupNorm}(\mathbf{h}) + t_b) \tag{15}$$

The DiffAE [16] model uses the original image as the conditioning input to encourage the diffusion model to reconstruct the input image accurately. In contrast, we utilize HE images as the conditioning

input to guide the model with the semantic encoding derived from the HE image. This approach leverages the rich semantic information contained in HE images to improve the quality and relevance of the generated IHC outputs.

3.2.3 Model Architecture

The architecture of our model is based on the principles of Diffusion Autoencoders (DiffAE) [16], incorporating Residual Blocks (ResBlocks) and Adaptive Group Normalization (AdaGN) to process input images. The design leverages Multi-Layer Perceptrons (MLPs) to integrate time and semantic encodings, guiding the generation process through rich, context-aware transformations. We followed the convention of using UNet with ResBlock as the backbone as in [4], [15] and [16]. In addition, QKV attention mechanism [9] mentioned in our pix2pix model is also utilized in between ResBlocks.

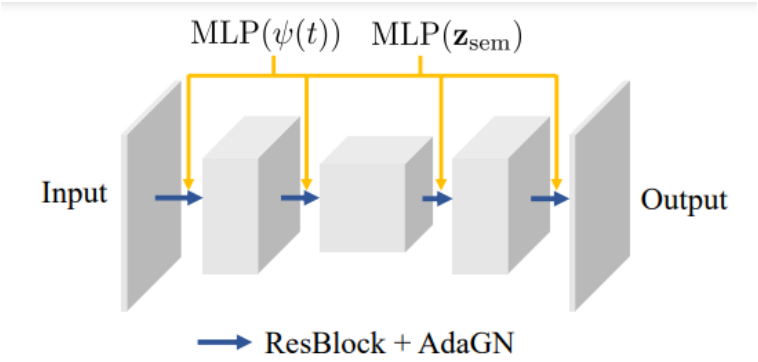

Figure 9: General architecture of our method based on [16]

The semantic encoder shares the same architecture as the encoder part of the denoising UNet which also follows the design of [16].

3.2.4 Loss Funtion

The training objective of the proposed model is simply defined as:

$$L = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_0,\epsilon_t} \left[\|\epsilon_\theta(\mathbf{x}_t, t, \mathbf{z}_{\text{sem}}) - \epsilon_t\|_2^2 \right] \tag{16}$$

The loss function is built based on [16]. The goal is to minimize the mean squared error (MSE) between the noise added in the forward process and the predicted noise given timestep t and semantic encoding z_{sem}

4 Implementation

4.1 Data Preprocessing

We utilized the officially available Breast Cancer Immunohistochemical (BCI) dataset, which contains 9,746 images (4,873 pairs). The dataset is already divided into training and test sets, with 3,897 pairs designated for training.

In our project of transforming Hematoxylin and Eosin (HE) images into Immunohistochemistry (IHC) images, effective data preprocessing is crucial. We leverage the capabilities of HistoQC [19], an open-source software, to filter out low-quality images and ensure the reliability of our input data. The filtering standards we used by HistoQC are as follows:

Flat Areas: Detects smooth (flat) areas in an image by convolving the image with an average filter and comparing it to the original image.

Small Tissue Filled Max Area: Sets the upper limit on the size of holes that should be filled within the tissue mask.

Blurry Removed Num Regions: Represents the number of regions (or areas) within an image that has been removed due to being identified as blurry.

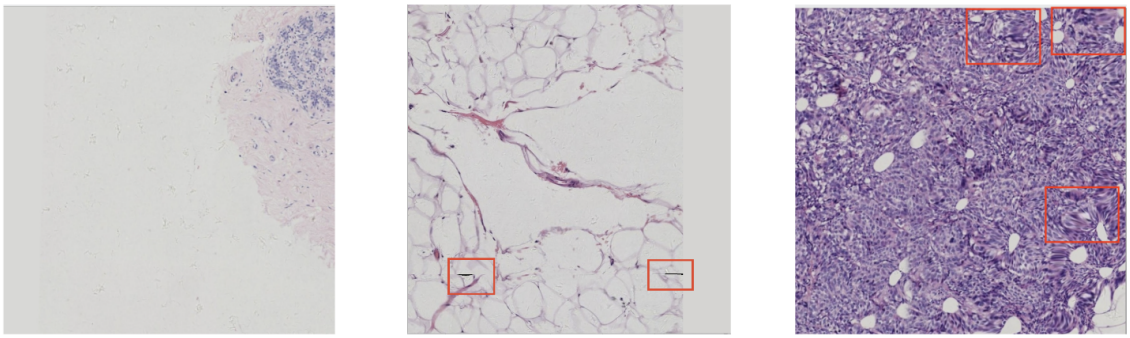

Figure 10: Examples of Filtering Criteria: (Left) Flat Areas, (Middle) Small Tissue Filled Max Area, (Right) Blurry Removed Num Regions

To test the efficacy of our preprocessing methods, we employed the Pyramid pix2pix model [1] for both the preprocessed and original images. This generative adversarial network (GAN) model is renowned for its capability in image-to-image translation tasks. We compared the quality of the resulting images using the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM).

We conducted two experiments to verify the effectiveness of the preprocessing. Both experiments utilized the Pyramid pix2pix model for 100 epochs with compressed images of size 256x256 pixels.

In the first experiment, we applied two conditions: Small Tissue Filled Max Area >100 and Blurry Removed Num Regions >1. After filtering, the dataset contained 3,656 pairs, resulting in a filter rate of 6.19%. The second experiment used the conditions: Small Tissue Filled Max Area >100 and Flat Areas >3. After filtering, the dataset contained 3,722 pairs, with a filter rate of 4.49%.

Original Pairs	After preprocessing Pairs	Filter Rate
3897	3656	6.19%

Table 1: Experiment 1 of Dataset Preprocessing

Original Pairs	After preprocessing Pairs	Filter Rate
3897	3722	4.49%

Table 2: Experiment 2 of Dataset Preprocessing

4.2 Diffusion Models

4.2.1 Dataset Preparation

In addition to the filtering method based on HistoQC, we downgraded the images from 1024 × 1024 to 512 × 512 pixels and subsequently applied a random crop to the resolution of 128 × 128 to reduce the GPU memory usage while keeping sufficient reception field. The random crop can also be regarded as a data augmentation to the original dataset.

Besides, we applied normalization to the images to transform the pixel values to a range of [-1, 1]. We did not introduce any further data augmentation to the dataset.

4.2.2 Hyperparameter Setting

We basically followed the hyperparameter setting in [16].

Batch Size	16
Channels	[64, 128, 256, 512]
Encoder Channels	[64, 128, 256, 512]
$\mathbf{z}_{\text{sem}}$ Size	512
β Scheduler	Linear
Learning Rate	2e-5
Optimizer	Adam
Training T	500

Table 3: Summary of hyperparameters, each number of channels corresponds to two ResBlocks

4.3 Pix2pix and Its Variant

For pix2pix and its variant, we used the filtered dataset. Before feeding the images into the model, we performed several preprocessing steps. All images were resized to 512x512 pixels, and the pixel values were normalized. And random cropping images to 256x256 pixels, which was applied to increase the diversity of the training data and prevent overfitting.

For the training details,we used the Adam optimizer with a learning rate of 0.5 for both the generator and discriminator. The models were trained with a batch size of 10. In addition, we introduce early stop in the training process. For Pix2pix(U-Net), we trained 100 epochs; Pix2pix(TransU-Net) was trained for 45 epochs; Pix2pix(Attention U-Net) with L1 and adversarial loss was trained for 69 epochs and Pix2pix(Attention U-Net) with VGG loss was trained for 91 epochs.

4.4 Software Development

4.4.1 Collaborative Design with QuPath

Our software development process utilized two main tools: QuPath and JavaFX. QuPath is a highly specialized platform designed for pathology image analysis, with particular relevance to whole slide image analysis in digital pathology. It provides a wide range of functionalities, handling both automatic and manual tasks across various complexity levels. These capabilities were indispensable for our project, as they allowed for comprehensive analysis and manipulation of pathology images, facilitating the integration of our model into the existing workflow.

QuPath supports customizable extensions, which was a crucial feature for our development process. This flexibility enabled us to create a user-friendly Java extension that seamlessly integrates our model into QuPath. By leveraging QuPath's extensibility, we ensured that our solution could be tailored to meet specific user requirements and adapt to future advancements in pathology image analysis.

Complementing QuPath, JavaFX played a vital role in ensuring the creation of modern, intuitive user interfaces for desktop applications. JavaFX is known for its ability to build sophisticated and visually appealing interfaces, which was essential for enhancing the usability of our QuPath extension. The use of JavaFX allowed us to deliver a seamless user experience, making it easier for users to interact with our model and perform complex image analysis tasks.

Our implementation strategy was heavily reliant on the use of JavaFX and FXML. FXML is an XML-based language that defines UI components while allowing the direct use of Java code. This combination facilitated the design and development of a sleek, modern user interface for our QuPath extension. The ability to use FXML for UI definition, alongside Java code for backend logic, made the construction of complex interfaces straightforward and efficient.

QuPath extension templates, which import JavaFX, were instrumental in simplifying the development process. These templates provided a robust foundation upon which we built our extension, ensuring consistency and reliability. The integration of JavaFX within QuPath allowed us to take full advantage of its capabilities, resulting in a highly functional and aesthetically pleasing interface.

Ultimately, the result was a robust Java extension with integrated model functions, available for direct use within QuPath. This integration simplified the user experience, allowing users to leverage our model's capabilities without needing to switch between different software platforms. Besides the user-friendly interaction with our model, the QuPath extension provides additional benefits. QuPath's capabilities extend beyond those of other open-source bioimage analysis tools, particularly focusing

on brightfield images commonly used in routine pathology, such as Hematoxylin and Eosin (HE) and Immunohistochemistry (IHC) staining.

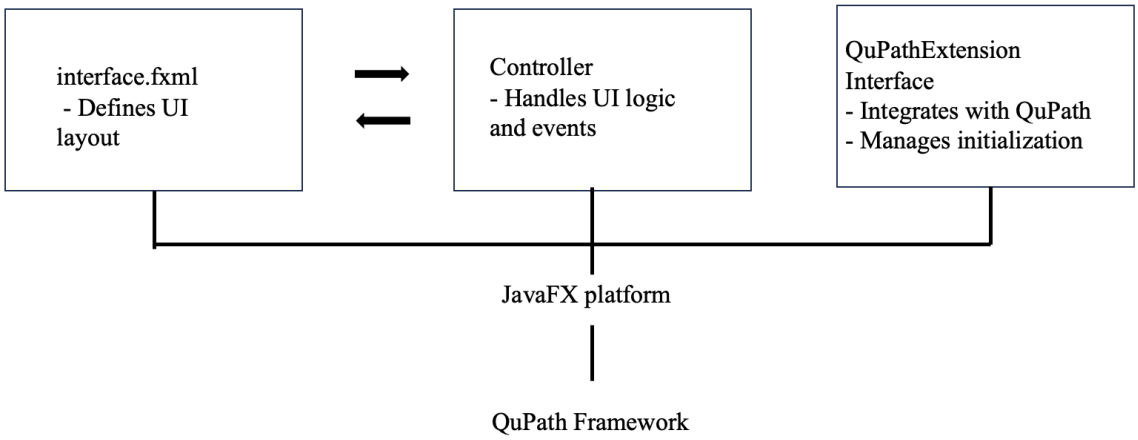

Figure 11: System Architecture: Integration of interface.fxml, Controller, and QuPathExtension Interface within the JavaFX and QuPath framework

Our decision to create a QuPath extension, rather than a standalone website for image transformation, was driven by several factors. First, it allowed us to harness QuPath's powerful image analysis tools alongside our image transformation functionalities. This synergy provided users with a comprehensive solution for diverse image analysis and transformation tasks. Second, integrating our model within QuPath ensured that users could access advanced image analysis capabilities in a single, cohesive environment. This integration reduced the complexity and learning curve associated with using multiple software tools.

4.4.2 Software Testing

To ensure the robustness and accuracy of the HE to IHC image conversion system, the implementation employed a comprehensive two-pronged testing approach: White Box Testing and Black Box Testing. Both testing methods were crucial in verifying the functionality, reliability, and performance of the system.

White Box Testing, also known as clear or glass box testing, was performed using TestFX and JUnit. TestFX, a framework for testing JavaFX applications, enabled us to simulate user interactions and verify the behavior of the user interface. JUnit, a widely-used testing framework for Java, facilitated rigorous unit testing of individual modules. This included the conversion algorithm, image loading process, and exception handling mechanisms. Extensive testing was conducted to ensure thorough coverage of the codebase. Boundary cases were meticulously examined, testing both extreme and normal values for inputs and outputs. This approach helped us identify and rectify potential issues related to the handling of edge cases and ensure the system's robustness. Additionally, the system's ability to handle exceptions, such as image file format errors, was rigorously assessed. By covering a broad spectrum of scenarios, we ensured that the system behaved as expected under various conditions, thus enhancing its reliability and stability.

Black Box Testing, on the other hand, focused on evaluating the outward functionality of the software without delving into its internal structures or workings. This approach was instrumental in validating the software's usability and functional aspects from the end-user's perspective. Three primary test cases were devised for this purpose: Regular HE to IHC Image Conversion, handling of Unsupported Image Formats, and handling of Incorrect File Paths.

Regular HE to IHC Image Conversion: This test case was designed to verify the core functionality of the software, ensuring that it could accurately convert Hematoxylin and Eosin (HE) stained images to Immunohistochemistry (IHC) stained images. The expected outcome was that the system would produce high-quality IHC images that closely resembled the target outputs. This test validated the effectiveness of the conversion algorithm and the overall performance of the system.

Handling of Unsupported Image Formats: In this test case, we evaluated the system's resilience when interacting with unsupported image formats. The software was tested with a variety of image file formats that were not supported by the conversion algorithm. The expected result was that the system would gracefully handle these files by generating appropriate error messages without crashing or exhibiting unintended behavior. This test was crucial in ensuring that the software could handle unexpected inputs robustly and provide clear feedback to users.

Handling of Incorrect File Paths: This test case focused on the software's ability to appropriately handle invalid or incorrect file paths. Users might inadvertently provide incorrect paths to image files, and the system needed to handle such scenarios gracefully. The expected outcome was that the software would return error messages indicating the issue with the file path, guiding the user to correct the input. This test validated the software's user-friendliness and error-handling capabilities.

The integration of JUnit in both White Box and Black Box Testing provided a structured and automated way to conduct these tests. JUnit's annotations and assertions enabled us to create comprehensive test suites that could be easily executed and maintained. Automated tests ensured consistent and repeatable testing processes, facilitating the detection of regressions and the validation of new features.

Overall, the combination of White Box and Black Box Testing ensured a thorough examination of the HE to IHC image conversion system. White Box Testing allowed us to scrutinize the internal workings and validate the correctness of individual components, while Black Box Testing focused on the overall functionality and user experience. This dual approach ensured that the system was not only functionally accurate but also reliable, user-friendly, and robust against various input scenarios.

4.4.3 Responding to User Suggestions

After our team used the software together, we came up with two suggestions. The first suggestion is to modify the QuPath extension's frame from a separate component to the initial interface of QuPath, as this would provide more parameters and make it easier to access information and use the functions. However, during our exploration, we only found the extension files provided by QuPath, which is the method we had adopted. Fortunately, we discovered an interface provided by QuPath called ImageJ, which is very useful for obtaining image information. Therefore, we decided to implement this interface.

The second suggestion was to enable users to enlarge images by scrolling with the mouse and to drag the image to view details. The solution was to add drag and zoom features to an ImageView. For the drag functionality, it records the mouse position when the mouse is pressed and calculates the movement delta when the mouse is dragged, updating the ImageView's position accordingly. For zoom functionality, it handles scroll events, adjusting the ImageView's scale based on the scroll direction: increasing the scale for scrolling up and decreasing it for scrolling down. This allows users to click and drag to move the image and use the scroll wheel to zoom in and out.

4.5 Evaluation Metrics

To accurately assess the performance of our image translation models, we employ a combination of three widely recognized metrics: LPIPS, PSNR, and SSIM. Each of these metrics offers a unique perspective on image quality and similarity, enabling a comprehensive evaluation of the generated images against the original images.

4.5.1 Learned Perceptual Image Patch Similarity (LPIPS)

Learned Perceptual Image Patch Similarity (LPIPS) quantifies the perceptual differences between two images by leveraging deep learning models to approximate human visual sensitivity to various image distortions. This metric evaluates the similarity between the generated image and its real counterpart by extracting and comparing deep features from pre-trained convolutional neural networks (CNNs). LPIPS has emerged as a preferred metric for tasks where the preservation of textural and color fidelity is paramount, as it aligns more closely with human perception than traditional metrics. The adoption of LPIPS in our evaluation strategy ensures that our model's performance is gauged not just on pixel-level accuracy but on perceptually relevant aspects, which is crucial for the subjective quality assessment of generated images.

4.5.2 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) is a traditional, widely utilized metric in the field of image processing for assessing the quality of reconstructed images. PSNR measures the ratio between the maximum possible power of a signal and the power of corrupting noise that affects its fidelity, expressed in decibels. A higher PSNR value indicates a higher quality of reconstruction, suggesting that the generated image is closer to the original image in terms of pixel-wise intensity. PSNR is particularly useful for evaluating the baseline accuracy of image generation models, providing a clear, objective measure of the noise level and pixel accuracy across the generated output.

4.5.3 Structural Similarity Index (SSIM)

The Structural Similarity Index (SSIM) is an advanced metric designed to improve upon traditional methods like PSNR by taking into account the perceived changes in structural information, luminance, and contrast of an image. Unlike PSNR, which evaluates images purely on pixel-level differences, SSIM assesses the visual impact of three characteristics of an image: luminance, contrast, and structure, thus providing a more holistic view of image quality. SSIM is particularly valuable for image generation tasks as it accounts for the structural integrity and textural details of the images, ensuring that the generation process preserves the essential attributes that contribute to the overall visual perception of the image. The inclusion of SSIM in our evaluation framework allows us to closely monitor the preservation of these critical image features, thereby ensuring the production of high-quality, visually pleasing generated images.

5 Experimental Results

5.1 Dataset Filtering Results

The results are presented in the table below. Higher PSNR and SSIM scores for the preprocessed images, compared to the original images, would confirm the effectiveness of our preprocessing methodologies for HE to IHC image transformations.

	PSNR	SSIM
original	17.46	0.59
filtered1	19.23	0.61
filtered2	17.53	0.57

Table 4: Comparison of PSNR and SSIM scores for original and preprocessed images

The results demonstrate a clear improvement in image quality for the first preprocessing condition. Specifically, the PSNR increased from 17.46 for the original images to 19.23 for the filtered images, indicating a higher signal-to-noise ratio and better image quality. Additionally, the SSIM score increased from 0.59 to 0.61, suggesting improved structural similarity.

For the second preprocessing condition, the improvements were less pronounced. The PSNR for the filtered images was 17.53, only slightly higher than the original, and the SSIM decreased to 0.57. This suggests that while some undesired features were filtered out, the overall image quality was not significantly enhanced. Therefore, we applied the first filtered dataset in subsequent analyses and transformations.

5.2 Benchmark Results

5.2.1 Pix2pix and Its Variant

Model	Average LPIPS	Average PSNR	Average SSIM
Pix2Pix (U-Net)	0.6718	17.1238	0.3174
Pix2Pix (TransUnet)	0.7110	16.4542	0.3032
Pix2Pix (Attention U-Net)	0.6959	6.6879	-0.0689
Pix2Pix (Attention U-Net with VGG-Loss)	0.5697	6.3755	-0.0591

Table 5: Evaluation Scores for Different Pix2Pix Models

Experimental results of pix2pix(U-Net):
The Pix2Pix (U-Net) model achieves a moderate level of performance. The average LPIPS score of 0.6718 suggests that there is a perceptual difference between the generated and target images, but it is within an acceptable range. The PSNR score of 17.1238 indicates that the model produces images with reasonable quality, as PSNR measures the ratio of signal power to the noise. The SSIM score of 0.3174 reflects moderate structural similarity, showing that the model captures some structural information but there is room for improvement.

As the figure shown, although the generated images basically maintain the overall structure, they are a bit blurry compared to the real images in terms of details and sharpness. This may be due to the fact that L1 loss tends to produce blurry images. Especially in terms of edges and textures, the generated images are not sharp enough to reproduce the details in the real images well. But for the overall performance, we can assume that the UNet-based pix2pix model is effective for image to image translation task.

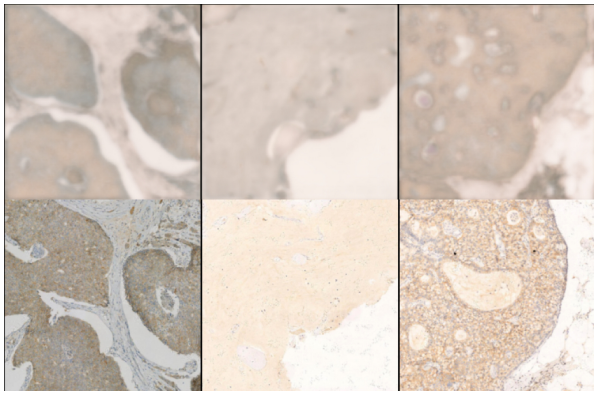

Figure 12: Qualitative results of pix2pix(U-Net)

Experimental results of pix2pix(TransUnet):
The TransUnet-based model shows a decrease in performance compared to the basic U-Net model. The LPIPS score of 0.7110 indicates a slightly higher perceptual difference. The PSNR score of 16.4542 is lower than that of the U-Net model, suggesting that the generated images have more noise or less fidelity. The SSIM score of 0.3032 also indicates a slight reduction in structural similarity.

Due to time and computational resource constraints, we were not able to train our own ResNet, so we used a pretrained ResNet50 as the base of the encoder part. However, ResNet50 is primarily designed for image classification and segmentation tasks, so the pretrained features were not well-suited for generating high-quality images in this context. Nevertheless, from the qualitative results, we can see that TransUnet has the ability to generate structures similar to the real images. If we can improve the residual blocks in the encoder, the results may be better.

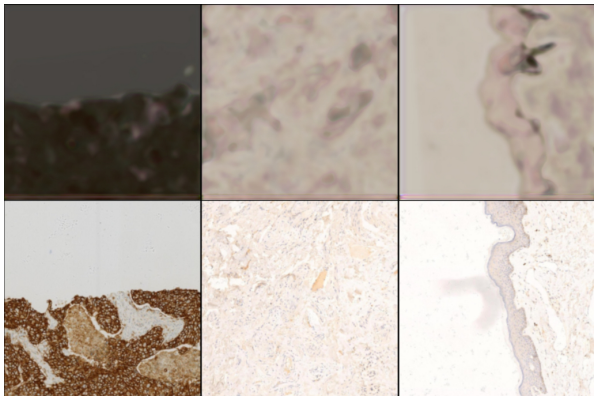

Figure 13: Qualitative results of pix2pix(TransU-Net)

Experimental results of pix2pix(Attention U-Net) with L1 loss and adversarial loss:
For the pix2pix model which generator is Attention U-Net, using a combination of L1 and adversarial loss functions, shows very low PSNR and SSIM values. The LPIPS score of 0.6959 is lower than pix2pix(U-Net), indicating better perceptual similarity. However, the PSNR score of 6.6879 and the negative SSIM score of -0.0689 suggest poor quantitative performance. The negative SSIM score is particularly concerning, indicating a significant lack of structural similarity. The poor scores may be attributed to the high noise levels in the images, which adversely affect these metrics.

The images generated by the Pix2Pix (Attention U-Net) model appear to be sharper and more detailed than those generated by the Pix2Pix (U-Net) model. The latter are blurrier, making it difficult to discern fine details. Although the generated images are still losing some details, in general, the performance of the addition of attention blocks, as implemented, provides some benefits.

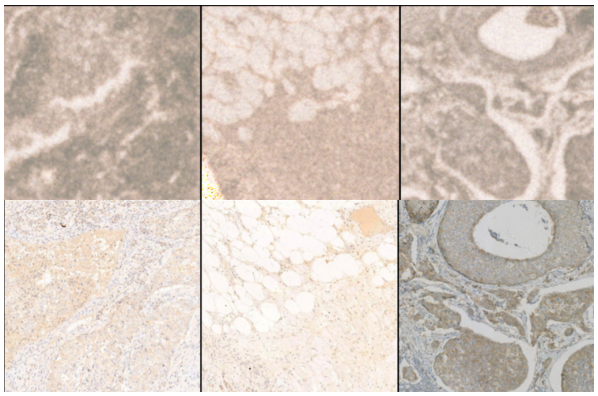

Figure 14: Qualitative results of pix2pix(Attention U-Net) with L1 and adversarial loss

Experimental results of pix2pix(Attention U-Net) with VGG loss:
For the other model pix2pix model which generator is Attention U-Net, incorporating an additional VGG loss in the loss function. The LPIPS score of 0.5697 indicates a significant improvement in perceptual similarity, suggesting that the generated images are closer to the target images in terms of human visual perception. However, the PSNR score of 6.3755 and the negative SSIM score of -0.0591 are still very low, similar to the previous Attention U-Net model. These results suggest that while the perceptual quality of the images is improved, the high noise levels continue to affect the quantitative metrics. Visual inspection confirms that the images appear better than what the PSNR and SSIM scores would suggest, indicating that the VGG loss helps in capturing perceptual quality that is not reflected in traditional metrics.

The generated images still appear to be less sharp and detailed compared to the real images. However, the images generated by the model with VGG Loss are better in terms of detail and sharpness. The structural details, such as boundaries and textures, are better captured by the model with VGG

Loss. The images generated with L1 and Adversarial Loss struggle more with reproducing these details accurately, leading to a more homogeneous and less detailed representation.

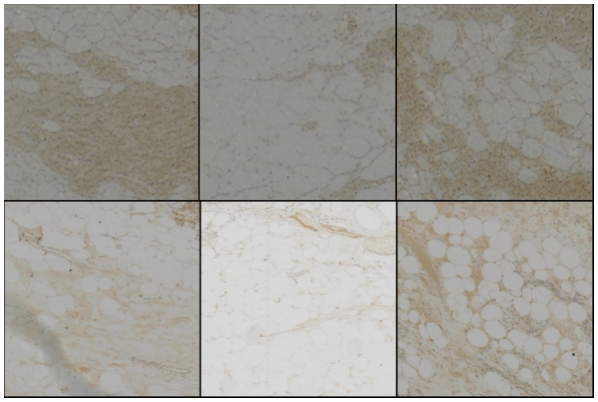

Figure 15: Qualitative results of pix2pix(Attention U-Net) with VGG loss

5.3 Software Testing Results

Test Case	Steps	Expected Results	Actual Results
Image Loading	1. Provide the file path of a valid image in a supported format. 2. Initiate image loading in the system.	The system should successfully load and display the image without errors.	The system successfully loaded and displayed the image.
Regular HE to IHC Image Conversion	1. Load a standard HE image. 2. Run the conversion algorithm.	IHC conversion image is successfully returned.	The system successfully returned the IHC conversion image.
Unsupported Image Formats	1. Input an image file in an unsupported format. 2. Initiate image loading or conversion.	The system should return an error message indicating the unsupported file format.	The system returned an error message indicating the unsupported file format.
Incorrect Image Path	1. Provide an incorrect file path for the image. 2. Attempt to load the image in the system.	The system should return an error message indicating the file path is incorrect.	The system returned an error message indicating the file path is incorrect.

Table 6: Test Cases for Image Processing and Conversion System

The table above outlines the test cases, the steps taken, the expected results, and the actual results.

Analysis of Test Results Image Loading: The system was tested for its ability to load valid image files without errors. The expected result was that the system would successfully load and display the images. The actual result confirmed that the system successfully loaded and displayed the image. For incorrect file paths, the system returned an appropriate error message.

Regular HE to IHC Image Conversion: This test verified the system's capability to convert HE images to IHC images accurately. The expected outcome was that the system would return a correctly converted IHC image. The actual result showed that the system successfully returned the IHC conversion image. For incorrect file paths, the system returned an appropriate error message.

Unsupported Image Formats: The system's response to unsupported image formats was tested. The expected result was that the system would generate an error message indicating the unsupported file format. The actual result confirmed that the system returned the expected error message. For incorrect file paths, the system returned an appropriate error message.

The detailed test results and the actual performance of the system demonstrate its reliability and highlight areas for potential improvement in the preprocessing and conversion pipeline.

5.3.1 Diffusion-based models

Due to the continuous change to the model, we only keep the final version of model for the evaluation. Preliminary results are abandoned early because the diffusion model takes too long to train. Besides, since the final results is not ideal as well, the preliminary results would not be worthy to tracking. As displayed in Table 7, the best test loss is found at 349 epochs and did not improve after hundreds of epochs. The model finally achieved a loss of approximately 0.026.

However, Table 8 illustrates the scores under LPIPS, PSNR and SSIM. As we can observe from the scores obtained, the results are extremely terrible. In addition, the qualitive results is displayed in Figure 16. The left most is the HE image as the condition, the one to the right is the real IHC image. The later two are both samples from the learned distribution which are the same because of the experimental setting. This also invalidates the results obtained. Further investigation indicated that although the model can denoise very well when timestep sampled is relatively low (i.e. only a little noise is added), the denoising works terribly when large T is sampled. As a result, the model output is very bad though the loss appears to be promising.

Model	Epochs	Best Epoch	Best Test Loss
Modified DiffAE	793	349	0.02585

Table 7: Loss and Epochs for Modified DiffAE

Model	Average LPIPS	Average PSNR	Average SSIM
Modified DiffAE	0.5235	6.7626	0.0114

Table 8: Evaluation Scores for Modified DiffAE

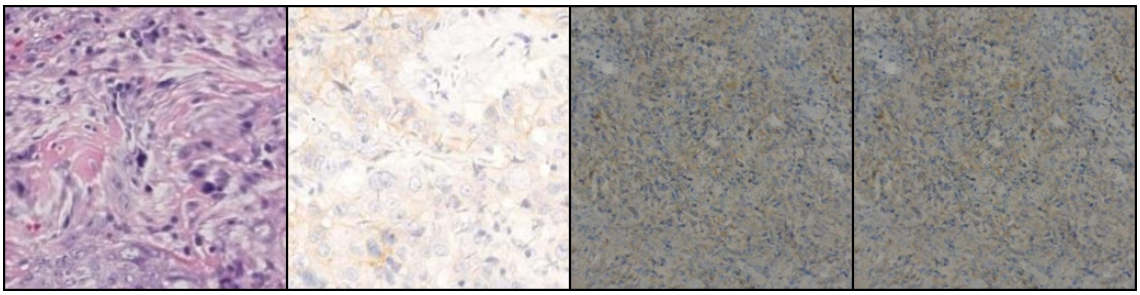

Figure 16: Qualitative results of modified DiffAE

6 Discussion and Outlook

6.1 Pix2pix model and variant

In this project, we implemented and evaluated several variants of the Pix2Pix model for image-to-image translation tasks, including the basic U-Net, TransUnet, and Attention U-Net architectures. The performance of these models was assessed using LPIPS, PSNR, and SSIM metrics, alongside qualitative visual comparisons.

From the experimental results, we could prove that Pix2Pix is a useful tool for image-to-image translation tasks. Pix2Pix (Attention U-Net) improved perceptual similarity (lower LPIPS) but still struggled with noise and structural accuracy, as indicated by low PSNR and negative SSIM scores. This suggests that while the perceptual quality of the images is improved, the high noise levels continue to affect the quantitative metrics. But, Visual inspection confirms that the images appear better than

what the PSNR and SSIM scores would suggest. We believe that Pix2Pix (Attention U-Net) has great potential for image translation task.

Based on the findings, several avenues for future research and improvements are identified. First, addressing the high noise levels observed in the generated images is crucial. This could involve using noise reduction techniques, both as a preprocessing step and as part of the model architecture, to improve PSNR and SSIM scores. Then, from the experimental result, we find integrating the attention mechanism could improve the model's performance in capturing detail and sharpness. We should focus on refining the implementation of attention blocks to ensure they effectively enhance important features and reduce noise. From the experimental result, we find that the integration of an attention mechanism could improve the model's performance in capturing detail and sharpness. This may involve experimenting with different attention architectures or integrating more sophisticated attention mechanisms.

Finally, instead of using existing pre-trained models, it would be beneficial to fine-tune these models on datasets more relevant to the specific image generation tasks. This could improve the transferability of features learned by the encoder.

By addressing these areas, we can work towards developing more robust and effective image generation models that produce high-quality, perceptually accurate, and structurally sound images.

6.2 Diffusion Model

Compared to the GAN [10] which works more implicitly and generate the target image directly, diffusion models learn added noise in each forward step. As discussed in the experimental results part, the model handles low-level noise easily but it is much more difficult to predict when noise level is high, resulting the model performed terribly. We propose possible refinements in two aspects. The first is to use more powerful models. For example, the original DiffAE [16] used more attention blocks which is not included in this project because it is extremely memory-expensive. We initiated another version using original DiffAE implementation but no effective result is collected yet. Alternately we could introduce extensions to the model, in [20], a novel method used autoregressive recurrent neural network (RNN) which is more suitable to deal with time series infomation to replace UNet-based architecture.

6.3 Software System

This software's functionalities include converting the HE images to IHC images, downloading the image, and seeing detailed parameters of images. These capabilities are essential for users who need to analyze and manipulate medical images efficiently. As mentioned before, although it acts as a part of the Qupath extension, it is isolated from the initial frame, making it hard to combine with other functions. Consequently, users may find it cumbersome to switch between different tools and may experience interruptions in their workflow. Enhancing integration with QuPath's initial frame would greatly improve the user experience by enabling smoother transitions and better utilization of all available features.

References

[1] A. Liu and et al., "Bci: Breast cancer immunohistochemical image generation through pyramid pix2pix," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1815–1824.

[2] *Breast cancer immunohistochemical image generation challenge*. [Online]. Available: `https://bci.grand-challenge.org/`.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[4] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *ArXiv*, 2020. [Online]. Available: `https://arxiv.org/abs/2006.11239`.

[5] A. Zhu and et al., "Breast cancer immunohistochemical image generation: A benchmark dataset and challenge review," *arXiv preprint arXiv:2305.03546*, 2023.

[6] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[7] T. Kataria, B. Knudsen, and S. Y. Elhabian, "Staindiffuser: Multitask dual diffusion model for virtual staining," *arXiv preprint arXiv:2403.11340*, 2024.

[8] M. Mirza and S. Osindero, *Conditional generative adversarial nets*, 2014. arXiv: `1411.1784 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/1411.1784`.

[9] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: `1706.03762 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/1706.03762`.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2672–2680.

[11] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *arXiv preprint arXiv:2105.05233*, 2021.

[12] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015, pp. 2256–2265.

[13] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021, pp. 8162–8171.

[14] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[15] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[16] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwajanakorn, "Diffusion autoencoders: Toward a meaningful and decodable representation," *ArXiv*, 2021. [Online]. Available: `https://arxiv.org/abs/2111.15640`.

[17] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[18] J. Chen, Y. Lu, Q. Yu, *et al.*, "Transunet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306*, 2021.

[19] A. Janowczyk, R. Zuo, H. Gilmore, M. Feldman, and A. Madabhushi, "Histoqc: An open-source quality control tool for digital pathology slides," *JCO Clinical Cancer Informatics*, 2019.

[20] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8857–8868.

A Task distribution

Task	Member
Preprocessing the Dataset	Maiqi Zhou
Development of Software System	Maiqi Zhou
Pix2Pix Model	Xichu Yu
Model Evaluation	Xichu Yu
Diffusion Models	Yuchen Li
Experimental Design	Yuchen Li

Table 9: Task Assignment

B Report distribution

parts of the report	Member
Introduction and Motivation	Yuchen Li
Preprocessing the Dataset	Maiqi Zhou
Development of Software System	Maiqi Zhou
Pix2Pix Model	Xichu Yu
Model Evaluation	Xichu Yu
Diffusion Models	Yuchen Li
Appendix	Maiqi Zhou

Table 10: Task Assignment

C Timeline change

There are some changes in our timeline.

1. Model Implementation: According to our initial plan, we aimed to complete the model implementation at the beginning of July. However, we delayed this until the day before the deadline. This is a common occurrence as we often hold out hope for better results until the last moment. Training the model was particularly time-consuming due to our limited GPU resources. The training process was slow, often requiring extended periods to see results, which then necessitated adjustments to the generated images. Additionally, we had to experiment repeatedly with different network structures.

2. Software Development: Our plan was to finish the software development by the end of June, allowing time for testing, debugging, and incorporating feedback from actual users. However, the completion was delayed by approximately 10 days. This delay was partly due to incorporating suggestions from others and overcoming challenges encountered while integrating the model.

3. Report Draft: We finished our draft nearly two days later than planned.