University of Bergen
Department of Informatics

Colorizing Scanning Electron Microscopy Images With Diffusion Models

Author: Emiel Venema
Supervisors: Stefan Bruckner

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

September, 2023

Abstract

We present a user-guided method for colorizing scanning electron microscopy (SEM) images using a conditional image-to-image diffusion model. Applying color to SEM images is an important part for researchers to communicate their findings and enhancing the visual appeal of their published work. Seeking to automate this otherwise manual and time-consuming process, we propose "AB-diffuser", a conditional image-to-image diffusion model that can generate SEM colorization both automatically and user guided with a few clicks. By limiting the diffusion process and data generation to color-data only, we ensure that none of the structural information of the researchers' valuable data is altered. We evaluated AB-diffuser's performance by testing its ability to replicate SEM colorizations from various published works with minimal user input. The results indicate that the model effectively captures the communicative intent and visual enhancements researchers had with their original colorizations.

Acknowledgements

I would first and foremost like to express my deepest appreciation to my partner and family, who in many regards have looked forward to this day more than I have. Their love, support and patience despite my mental and physical absence during this period in my life has been beyond invaluable. I am forever grateful.

I would also like to extend my gratitude to my supervisor, Stefan Bruckner who has provided excellent guidance, feedback and insight throughout this project. His support has been instrumental in maintaining my direction and focus on completing this project. I appreciate the effort and time he dedicated to guiding me, especially given his transition to the University of Rostock. His support has been invaluable in maintaining my direction and focus on completing this project.

Emiel Venema

Friday 1st September, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

SEM or scanning electron microscopy has been an important instrument for a wide range of scientific research and industrial applications. For any scientific discipline or industry that requires examination and analysis of nanoscopic structures or surfaces, SEM is a powerful tool that provides much higher resolution imagery than any optical microscope can provide. While human color perception relies on the interaction of photons with cone cells in our eyes, SEM images are formed by detecting electrons, thereby producing highly detailed but inherently grayscale images. In this thesis we will develop a conditional image-to-image diffusion model for the task of SEM image colorization.

1.1 Motivation and Problem Statement

In SEM, an electron gun fires a focused beam of electrons, scanning the specimen surface while interacting with its atoms. Images are formed by detecting the signals emitted from the sample due to these interactions. Commonly used signals for image formation are secondary electrons (SE), emitted due to the excitation of the stricken atoms, or backscattered electrons (BSE), which are electrons that undergo multiple scattering events, usually from deeper within the sample. An SEM image is constructed by encoding the signal intensity, which corresponds to the proportion of electrons emitted from the sample surface at that specific location, to a pixel value.

Researchers have since the early days of electron microscopy, been developing methods to produce color images. In 1940, Dr. Heinrich Herbst [12] patented a method for

visualizing the electron energy spectrum. By arranging bundles of cathode ray emitters of different strength in a ring, aimed at a phosphor screen, that emitted different colors depending on the energy of the exciting electrons, he successfully produced the first color image from an electron microscope.

Color vision is an important feature of human perception, enhancing our ability to discern and interpret objects and structures in our environment. The Gestalt principle of similarity states that humans tend to group objects together based on their similarity in color, shape or size, and one can see in the bottom row of Figure 1.1 how helpful color is to discern the various captured structures.

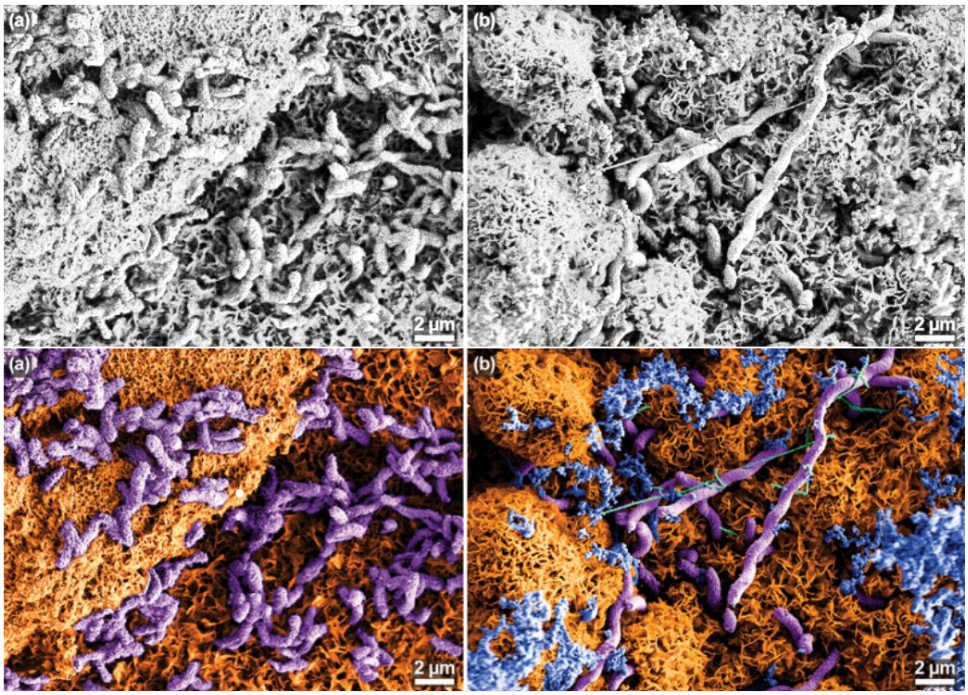

Figure 1.1: Desulfovibrio alaskensis bacterium biofilm present on iron sulfide, grayscale versus pseudocolored. Purple: Desulfovibrio alaskensis cells, Gold: iron sulfide minerals, Blue: carbon, Green: extracellular filaments
Source: Krantz et al. [18], used with permission from Taylor & Francis.

One can use data captured to color code SEM images. Backscattered electrons interact within the depth of the sample, losing energy as they are detected upon return. This loss of energy is dependent on the atomic numbers of the stricken region, and results in an image with atomic contrast. These gray tonal-levels can be assigned colors via a lookup table. Other signals, if captured, such as characteristic X-rays and secondary electrons can also be color encoded.

Manual colorization of SEM images via photo editing software is also a common practice. Figure 1.1 is an example of this. With manual colorization, there are no limiting factors on color assignment due to availability or composition of captured data, and one will have full artistic freedom to convey desired information or enhance the visual appeal of the image. This, however is laborious and time-consuming process. While there exists commercial solutions for SEM colorization such as MountainsSEM, there exists no viable alternatives that are either free or open source.

Bringing color to monochromatic images has been a well researched topic in the field of computer vision. Deep learning-based image colorization has been applied to a wide range of image domains and modalities such as natural images, infrared imagery, radar, manga and other types of line art. Auto-encoders, U-nets and generative adversarial networks have all been used to produce impressive results. Diffusion models have recently showed great results in other computer vision tasks such as image synthesis [7, 14] and image super-resolution [33, 36], however, research into their applicability to natural image colorization [37, 26] is still in its early stages. This is particularly true for diffusion models for user-guided colorization, where researchers have only unveiled significant findings in the past few months [3, 23, 22].

1.2 Research Question

Considering the tedious process of colorizing SEM images, the limited availability of dedicated tools, and the emerging potential of diffusion models in image-to-image translation tasks, we aim to address the following questions:

- Can diffusion models effectively colorize SEM images to enhance their visual interpretability?
- Can it produce results comparable to manual colorization and solutions dedicated to the task?

To answer these questions, we will develop and train a diffusion model for the task of SEM image colorization with support for user guidance. We will colorize grayscale SEM images from a series of publications, to asses the model's ability to capture and enhance the interperability of the features referenced in the original research. The model will also be evaluated against a set of colorized SEM images to assess if our model can provide

comparable results to common approaches, in terms of communicative intent and visual appeal.

As an addition to this thesis, we will also compare our model against other deep learning based image colorization methods on CUB-200-2010 [46] and Oxford 102 Flowers [28], two common datasets used to evaluate natural image colorization models. We present our attained peak signal to noise ratio (PSNR) against four similar baseline models.

1.3 Contributions

The main contributions of this thesis are: AB-diffuser, a conditional image-to-image diffusion model for the task of SEM image colorization with support for user guidance. The diffusion model only operates on the chromatic channels of images in cieLAB space, using grayscale images as a conditioning signal, along with optional user inputs for user-guided colorization. AB-diffuser was trained on our dataset consisting of 2000 colored SEM images, and evaluated against modern methods for SEM image colorization.

1.4 Thesis Structure

This thesis is organized into the following chapters:

- **Chapter 2: Theoretical Background**
  Introduction to neural networks, U-Nets, and denoising diffusion probabilistic models.
- **Chapter 3: Related Work**
  Review of contributions and materials we base our work on and draw inspiration from including: deep learning-based colorization models, diffusion models for image colorization, and SEM image colorization solutions.
- **Chapter 4: Methodology**
  Detailed explanation of the approach and architecture of AB-diffuser, our diffusion model used for SEM image colorization.
- **Chapter 5: Implementation and Prototyping**
  Description of the software tools used and the steps taken in the implementation of the proposed model.

- **Chapter 6: Results**
  Presentation and analysis of the model's performance on SEM image colorization tasks, and evaluating its performance against other deep learning-based image colorization solutions.
- **Chapter 7: Discussion**
  Interpretation of results, comparison with existing methods, discussion of the model's strengths and limitations and suggestions for future research directions.
- **Chapter 8: Conclusion**
  Summary of the research, key findings, and answers to research questions.

Chapter 2

Theoretical Background

This chapter outlines the basic principles Artificial Neural Networks and machine learning, as well as an introduction to Denoising Diffusion Probabilistic Models (DDPM).

2.1 Neural Networks

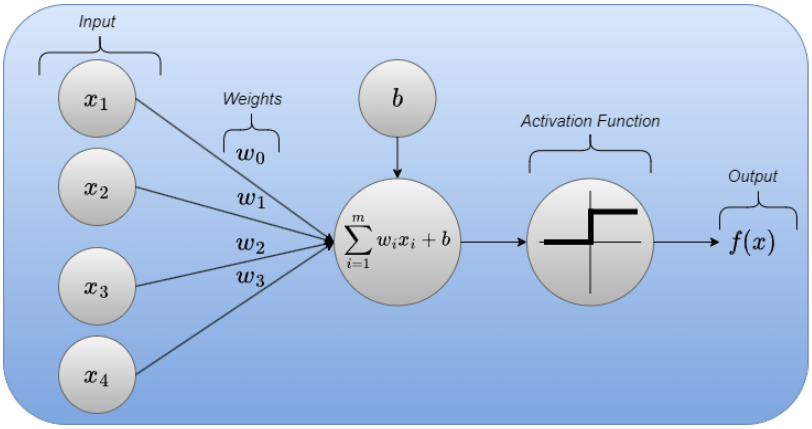

Figure 2.1: Diagram of a perceptron. A weighted sum of input data x and weights w is added with a bias term b and passed through the activation step-function to produce output $f(x)$

Artificial Neural Networks or ANN's belong to a class of machine learning (ML) models. ANN are functions that have the goal is to approximate some function f. One of the reasons we see ANN's everywhere today is their great ability to approximate functions,

and that they are able to perform a wide range of tasks, from classification and regression, to facial recognition and natural language processing.

The strengthening of signals between the simultaneously firing neurons when conducting repetitive motor or cognitive tasks plays a central role in behavioral learning and formation of memories Citri et al. [5]. ANN's are inspired by this neuron-to-neuron, weighted input-output relationship. Input data enters a processing unit called an artificial neuron, whose output is multiplied with a connection weight before being received by the next artificial neuron, analogous to the synaptic-connectivity strength between neurons in the brain.

2.1.1 Feedforward Neural Networks

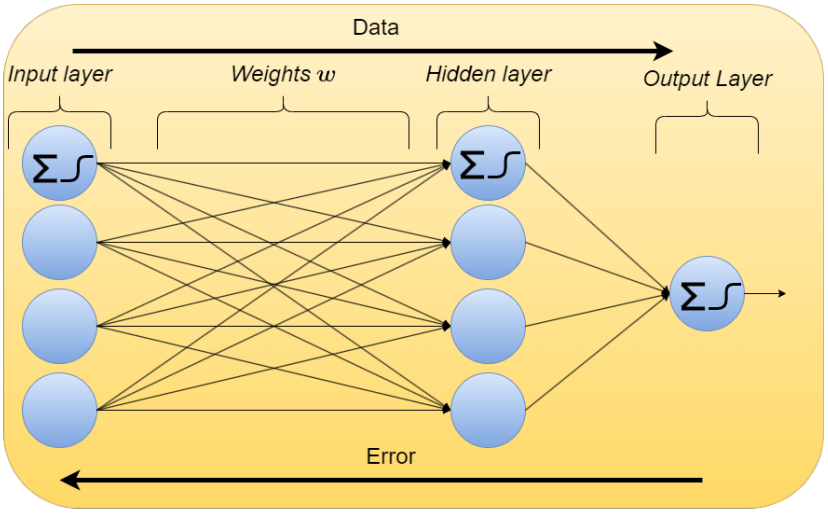

Figure 2.2: Diagram of a fully connected feed forward neural network. The input layer, consisting of 4 artificial neurons receives the data, and each neuron computes its weighted sum and activation. The activations are propagated through the hidden layers, to the output layer consisting of one neuron which computes the final output.
Note the bottom arrow, representing the flow of error back through the network when performing backpropagation during training.

FNNs are the simplest form of Artificial Neural Networks, where in contrast to Recurrent Neural Networks (RNN), the flow of information goes only one way: through the input layer, to the hidden layers if any, and finally throughout the output layer. They are built up by layers of artificial neurons or perceptrons.

Perceptrons: The perceptron was developed in the 1950's by Frank Rosenblatt. While other kinds of artificial neurons are mostly used in today's ANN's, many of the principles are the same. A perceptron is, in the strictest sense, a type of binary classifier that maps a weighted vector of m binary inputs or features $x = x_1, x_2, \ldots, x_m$ to a single binary number $f(x)$, as illustrated in Figure 2.1. An activation function, in this case the Heaviside step function, consists of a bias term b added with the dot product of features x and a vector of m real-valued weights $w = (w_1, w \ldots., w_m)$ and can be expressed as such:

$$f(x) = \begin{cases} 1 \text{ if } b + x \cdot w > 0 \\ 0 \text{ otherwise} \end{cases} \tag{2.1}$$

where $w \cdot x$ is the weighted sum $\sum_{i=1}^{m} w_i x_i$ of the feature vector x and weights w, and b is the bias term.

Perceptrons are a special case of the more general term artificial neuron. Artificial neurons can have a wide range of input data types, and activation functions allowing them more expressiveness and to handle more complex data and tasks. We can chain these neurons together in layers so that the outputs of neurons in the first layer, also called activations, become the inputs of the neurons in the next layers and so on. This yields us the feed forward network.

Figure 2.2 illustrates a three layer deep feedforward neural network. By adjusting the weights and biases of the network depending on the data we want to classify or perform regression on, we can control the output. This can be done manually, or ideally, via a learning rule that decides how the individual weights and biases should be manipulated, and a training process that feeds the network data. When applying the learning rule repeatedly no longer yields improvement in the network's performance, the network is said to have achieved convergence.

2.1.2 Training Neural Networks

Training a neural network is the process of adjusting the networks parameters, weights and biases, so that it learns a mapping from the input to specific outputs. This is often done in a supervised manner. In supervised learning, features of labeled training data is fed into the network, and the distance between the networks output and the label is measured. The networks parameters are then adjusted as to minimize this distance called "loss", with the goal of having the neural network producing outputs closer to this label

or ground truth the next time it encounter similar data, while also minimizing the overall average loss over the whole training set, the "cost".

The choice of loss function is important and one must take into account what task the network is trying to solve, and what minimization of the loss function yields. For regression problems mean squared error (MSE) is often used:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{2.2}$$

where y_i is the true value, and $\hat{y}_i$ is the predicted value.

If we base our cost function on the MSE so that $C(\vec{w}, \vec{c}) = \frac{1}{m}\sum_{i=1}^{m} \|y_i - \hat{y}_i\|^2$, the goal of training the network is to find the vector of weights w and biases b that minimizes average squared error over the training set.

Gradient descent with backpropagation is the standard approach to achieve this. By calculating the partial derivatives of the cost function with respect to the weights and biases, the direction of the steepest ascent can be found. In order to move in the opposite direction, the respective gradients are subtracted from the vectors of weights and biases as such:

$$\vec{\mathbf{b}'} := \vec{\mathbf{b}} - \eta \begin{bmatrix} \frac{\partial C}{\partial b_1} \\ \frac{\partial C}{\partial b_2} \\ \vdots \\ \frac{\partial C}{\partial b_n} \end{bmatrix} \qquad \vec{\mathbf{w}'} := \vec{\mathbf{w}} - \eta \begin{bmatrix} \frac{\partial C}{\partial w_1} \\ \frac{\partial C}{\partial w_2} \\ \vdots \\ \frac{\partial C}{\partial w_m} \end{bmatrix} \tag{2.3}$$

with hyperparameter η being the learning rate, the size of the steps taken towards the minimum. Higher learning rates increase the risk of overshooting the minimum and oscillating around it, while lower may lead to slower convergence or getting "stuck" in local minima. Avoiding the very computationally expensive task of calculating the loss over a large training set for every step, mini-batch gradient descent is used instead. The steps are taken on the gradient of loss over a shuffled, smaller subset of training data. Even tough it will not converge directly towards a minimum, it will still converge faster due the higher amount of steps taken during a pass over the whole training set, an "epoch".

In practice, the gradients are calculated by propagating the loss backwards through the network, a process called backpropgotation. Backpropagation aims to determine the contribution of a given weight and bias to the cost by calculating the gradient of the cost with respect to them. Going from the gradient on the output layer, and working backwards layer by layer, using the chain rule of calculus to compute the partial derivatives in the former layers.

2.1.3 Activation Functions

The job of the activation function is to transform the weighted input of the artificial neuron to an output. These functions are usually simple, but the choice of activation function will greatly affect what kinds of data the network can be trained on effectively, and what kinds of functions it can approximate. A network consisting of step-functions tends to be sensitive; small adjustments to the weights and biases can be just enough to push the activation from 0 to 1 and vice versa. This can have consequences for the layers receiving this output. While the adjustment may produce a more accurate prediction for the intended class, it might be to the detriment of another [9]. With a linear activation function, the network's decision boundaries separating the classes will also be linear, making it less effective on complex non-linear data.

One such example is the Sigmoid activation function, a bounded S-shaped function denoted $\sigma(x) = \frac{1}{1+e^{-x}}$ with a continuous range of $[0, 1]$. The smoothness of the function ensures that small changes in the weights and biases will only cause small changes in the output. The non-linearity it introduces also allows the network to learn more complex decision boundaries. In fact the Universal Approximation Theory "arbitrary width" case was proven using Sigmoid neurons in the hidden layers [6].

2.1.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a class of neural networks that specializes in processing data in matrix structures, such as images. While conventional neural networks with 1D feature vectors as input are able to process such data, they fail to capture its spatial structures well. CNNs are able to process images in their native structure, maintaining the spatial integrity of lines, curves and other features present. CNNs are simply ANN's with one or more convolutional layers.

In the convolutional layers, the input is convolved with a filter called the kernel. The kernel has the same depth as the input, but a smaller height and width and consists of weights that determines the output values of the convolution. The kernel slides a certain distance for each convolution, the stride, over the whole height and width of the input matrix. The dot product of the kernel and the covered input values of the input matrix is computed at each step. These values are then stored in a new matrix called the feature map, before being passed through an activation function.

The dimensions of the feature map are determined by the size of the kernel, the stride and the padding. For instance, a 3x3 kernel with a stride of 1 convolved over a 6x6 input matrix will produce a 4x4 feature map. To preserve the spatial dimensions of the input, one can pad the same input matrix with 0's along its height and width, resulting in a 6x6 activation map. Multiple kernels are used for each convolutional layer, producing one feature map each.

So instead of having full connectivity for each input feature and activations, we have sparse connectivity where each activation is determined by a smaller subset of the input features, known as the kernels receptive field. This reduces the amount of parameters needed, since the kernel weights are shared across the whole set of inputs. Even tough we have this sparse connectivity, local features further down the network are still indirectly affected by parameters further up the network.

We can reduce the number of parameters even further by using pooling layers. Pooling layers are similar to convolutional layers, where a filter is slid over the activation map, but instead, a pooling function determines the output of the values in the receptive field. Common pooling functions are max-pooling and average-pooling. Like the name suggest, for max pooling, the maximum value within the receptive field is selected, reducing the spatial size of the resulting feature map, while maintaining dominant features.

The network is made up by several of these convolutional and down-sampling pooling layers, where in the shallower layers the network extracts high-level features such as lines and curves while the deeper layers learns more complex features as their outputs are the "condensed" products of the previous layers.

2.1.5 U-Nets

The U-net is a CNN architecture, originally developed for biomedical image segmentation by Ronneberger et al. [34], but has been utilized other computer vision tasks such as

image-colorization [21, 52], image generation [14] and various image-to-image translation tasks [37]. U-nets are symmetric "U-shaped" networks, consisting of one contracting or down-sampling path called the encoder network, followed by an up-sampling or expanding path called the decoder network.

The encoder network is made up of convolutional blocks that double the feature map depth, followed by down-sampling max-pooling layers, who halve the feature map's spatial resolution. The decoder network replaces the max-pooling with up-convolutions where the feature maps are bilinearly scaled up by a factor of 2 before convolution, restoring the features spatial resolution, and halving their depth. In between the convolutional blocks, there are skip connections that concatenate activations from the encoder network to the decoder blocks. This allows the decoder network to recover spatial information of the features present in the shallower layers otherwise lost in the down-sampling process.

Even though the earlier layers provide better spatial information, they lack quality feature representation compared to the deeper layers. To compensate for this one may utilize attention gates at the skip connections. [29] In Figure 2.3 we see how the attention heads are constructed. Each attention gate learns an importance map of attention coefficients α from the skip connections $\mathbf{g}$ and the previous decoder layer $\mathbf{x}$, which is multiplied element-wise with the skip connection activations, scaling them according to their importance.

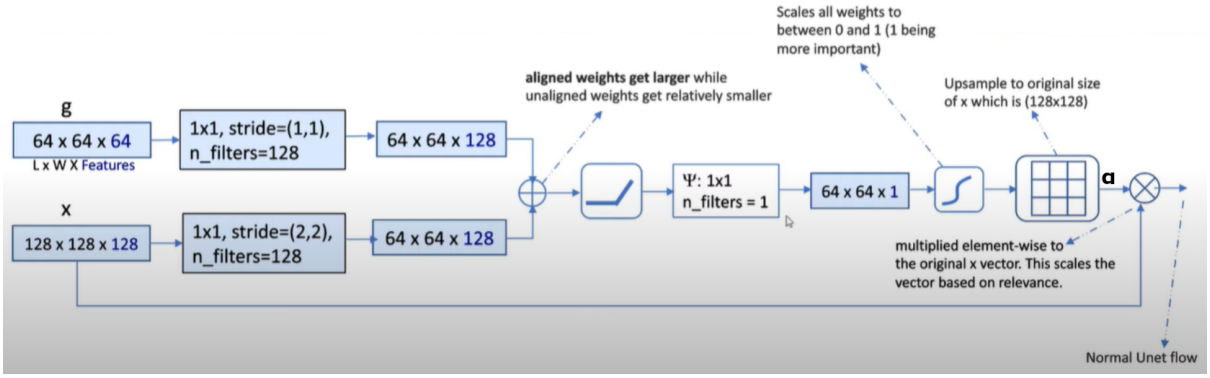

Figure 2.3: Attention head at the skip connections of the U-net. $\mathbf{g}$: previous decoder layer, $\mathbf{x}$: skip connection from encoder network, and can be seen multiplied attention coefficients α
Source: https://youtu.be/KOF38xAvo8I?t=746 with permission.

2.2 Denoising Diffusion Probabilistic Models

Generative models in machine learning have the aim of capturing the underlying distribution of a given data set. In other words, if we assume that the data, be it image-data, text, audio, are samples from a probability distribution $p_{data}(x)$. Generative models try to learn the approximated probability distribution $p_\theta(x)$. By drawing from this approximated distribution we are able to generate new samples.

DDPM's or simply "diffusion models" are a class of generative models that in recent years has emerged as a new state of the art of the deep generative model family. They have achieved great success in the field of image synthesis, even out performing generative-adversarial-networks [7] They have shown potential in the domain of computer vision tasks, such as image segmentation [2] and image-to-image translation tasks like colorization, JPEG restoration, inpainting and uncropping [37]. Text-to-image diffusion models such as DALL-E2 [30] and Stable Diffusion [33] have gained enormous popularity over the last year, being able to generate high quality and resolution images that are true to their text prompts.

This section is mostly based on the work of Ho et al. [14], who presented such substantial improvements to the original formulation of diffusion models by Sohl-Dickstein et al. [**?**], that much of the recent developments of diffusion models can be attributed to them. Note that for the sake of brevity, we have omitted much of the derivations leading the various terms and expressions. This section outlines the mathematical and practical details of DDPMs, but at a higher level.

2.2.1 The Diffusion Processes

The diffusion process giving the name to diffusion models has two phases, the forward and backwards diffusion process. The former is a Markov chain that transforms data to noise, and the latter is a reverse Markov chain that transforms noise back to data. The forward diffusion process begins with a given data distribution $x_0 \sim q(x)$ and an analytically tractable distribution $\pi(y)$ as a target, usually the standard gaussian distribution.

Given a sample x_0 from $q(x)$, the forward diffusion process gradually adds a small amount of gaussian noise to the sample using a Markov chain of $\mathcal{T}$ steps, with a variance or beta schedule given by $\{\beta_t \in (0,1)\}_{t=1}^{\mathcal{T}}$, giving us a sequence of increasingly noisy

random variables $x_1, x_2, .., x_T$ with a density of $q(x_t \mid x_{t-1})$. Using the Markov property of the transitions and the chain rule of probability, and the standard gaussian distribution as the target, one can factorize the joint distribution of the random variables conditioned on x_0 and define the forward process as:

$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1}) \tag{2.4}$$

with density

$$q(x_t \mid x_{t-1}) := \mathcal{N}(x_t; \boldsymbol{\mu_t} = \sqrt{1-\beta_t} x_{t-1}, \boldsymbol{\Sigma_t} = \beta_t \boldsymbol{I}) \tag{2.5}$$

where $\beta_t \boldsymbol{I}$ denotes the covariance for all dimensions of x_t

As Ho et al.[14] noted, to obtain a sample of x_t without applying q to the whole sequence $x_1, x_2, \ldots, x_{t-1}$, one can reparameterize the forward process to obtain a closed form:

Let $\epsilon_0, \ldots, \epsilon_{t-2}, \epsilon_{t-1} \sim \mathcal{N}(0, \boldsymbol{I})$, $\alpha_t := 1-\beta_t$ and $\bar{\alpha}_t := \prod_{i=1}^{T} \alpha_i$ with reparameterization $\mathcal{N}(\mu, \sigma^2) = \mu + \sigma \cdot \epsilon$, x_t can be written as:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \epsilon \tag{2.6}$$

This enables us to precompute a variance schedule with β_t as a hyperparameter where $1-\bar{\alpha}_t$ gives the variance of the noise at a given timestep, and to sample x_t at any timestep. The densisty becomes:

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t)\boldsymbol{I}) \tag{2.7}$$

Training: Going from isotropic gaussian noise to some sample of $q(x_0)$ is performed by sampling $x_T \sim \mathcal{N}(0, \boldsymbol{I})$, then sampling the reverse steps $q(x_{t-1} \mid x_t)$ back to x_0.

$$q(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \boldsymbol{I}), 1 < t \leq \mathbf{T} \tag{2.8}$$

The problem is the intractability of $q(x_{t-1} \mid x_t)$, so it is instead approximated by a Neural Network p_θ.

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{2.9}$$

$$p_\theta(x_{t-1} \mid x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t)), 1 < t \leq \mathbf{T} \tag{2.10}$$

Note that in the work of Ho et al. [14] they decided to only have the neural network learn the mean $\mu_\theta(x_t, t)$, and set the variance $\mathbf{\Sigma}_\theta(\mathbf{x}_t, t)$ to a time depended constant $\sigma_t^2 \boldsymbol{I}$, using either $\sigma_t^2 \boldsymbol{I} = \beta_t$ or $\sigma_t^2 \boldsymbol{I} = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, both producing similar results.

The network is trained by optimizing the minimum log-likelihoods variational lower bound as such:

$$L := \mathbb{E}_q \Bigg[\overbrace{\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}}^{L_{vlb}} \Bigg] \tag{2.11}$$

L_T is referred as the constant term and is ignored while training, since $q(\mathbf{x}_T|\mathbf{x}_0)$ and $p(\mathbf{x}_T)$ are both isotropic Gaussians with zero mean when $t \to T$, leading to a negligible KL-divergence.

L_{t-1} is the step-wise denoising term that compares p_θ with the forward process posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ conditioned on x_0.

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \tag{2.12}$$

$$\text{where } \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \tag{2.13}$$

L_0 is the reconstruction term, which is also ignored since its already approximated in L_{t-1} by the neural network.

2.2.2 Simplified Loss

Another discovery by Ho et al. [14] was that L_{t-1} can be further simplified. By rewriting x_0 in terms of the reparameterization of the forward process (see Equation 2.6)

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t + \sqrt{1-\bar{\alpha}_t}\epsilon\right) \tag{2.14}$$

The mean of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ can then be expressed as:

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\left(\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t + \sqrt{1-\bar{\alpha}_t}\epsilon)\right) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) \tag{2.15}$$

The authors decided to have the neural network approximate $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t)$, and now since x_t is also a input of the model, and all other quantities of $\tilde{\boldsymbol{\mu}}_t$ are known, $\mu_\theta(x_t, t)$ can have the same form:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) \tag{2.16}$$

Now the neural network only predicts the noise of x_t at timestep t, and the target and approximated mean can be compared with a MSE loss function after reconstruction. The loss is then:

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right\|^2\right] \tag{2.17}$$

Experimentaly, better results and easier implementation was achieved when ignoring the scaling term in Equation 2.17, leading to a even more simplified loss function:

$$L_{simple} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right\|^2\right] \tag{2.18}$$

In practical terms the process of training a DDPM is done as follows.

1. Define a variance or noise schedule β_t. Ho et al. [14] uses a linear scedule from $\beta_1 = 10^-4$ to $\beta_T = 0.02$, with $T = 1000$. Precompute α_t and $\bar{\alpha}_t$.

2. Select timestep $t \sim \mathbf{U}(\{1, ..., T\})$ and add noise according to schedule β_t to x_0, creating noisy variable x_t following Equation 2.6

3. Input timestep embedding t and x_t into neural network, an U-net, to predict the noise.

4. Compare output with noise $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$, taking gradient descent step on the loss.

5. Repeat 2-4 until convergence.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: **until** converged

Figure 2.4: Training algorithm for DDPM by Ho et al. [14]

2.2.3 Backward Process and Sampling

Now the final piece of the puzzle is how we go backwards from noisy data $x_T \sim \mathcal{N}(0, \boldsymbol{I})$ to a clean generated sample x_0. Reparameterizing $p_\theta(\mathbf{x}_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right), \beta_t)$ gives us:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sqrt{\beta_t}\epsilon \tag{2.19}$$

Now all thats left to do is to sample gaussian noise $\mathbf{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$, input it into the neural network together with the timestep $t = T$ iteratively for $t = T, ..., 1$ to produce the final generated sample x_0, as illustrated in Figure 2.5.

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 2.5: Sampling algorithm for DDPM by Ho et al. (2020)

Chapter 3

Related Work

Putting SEM image colorization using DDPM's into context requires discussion of the involving disciplines, their development and current state. We examine contributions to Deep Learning Image Colorization, Diffusion Models and SEM Image Colorization that we base our work on, and draw inspiration from.

3.1 Image Colorization With Convolutional Neural Networks

The colorization task has traditionally received less attention in the field of computer vision than prominent tasks like semantic segmentation, object detection and image generation. Still great strides have been made in the recent years with the rise of deep learning and convolutional neural networks, and increasingly available large image datasets. We will discuss the task itself, with some prominent approaches and their respective use-cases.

3.1.1 Image Colorization Task

In image colorization in a deep learning context, we leverage modern CNN architecture's ability to learn semantics about colors in the visual world from large image datasets to assign plausible color values to pixels in grayscale images.

Deep learning colorization arose due to the combination of multiple factors: the potential of CNNs as a tool to solve computer vision tasks, the lack of quality as well as need

for human intervention in the colorization methods at that time. The first deep learning colorization method was "Deep Colorization" by Cheng et al. [4] it is a fully automatic colorization solution utilizing a CNN consisting of 3 hidden layers, and a two neuron output layer corresponding to the UV chromatic channels in YUV color space. A dataset consisting of 2668 images from the Sun dataset [48] with 47 object categories was used for training. For each pixel in the luminance channel Y, a set of feature descriptors were computed to be used as input for the model. This set includes a 7x7 patch of surrounding pixel values, the DAISY [42] feature descriptor, and an annotation label indicating its object category.

Problems with desaturation and diversity of colors in fully automatic methods were addressed in the work of Zhang et al. [51]. They observed that, in natural image datasets, the distribution of colors tends to be biased towards desaturated values. They also argued that using Euclidean MSE between ground truth and prediction, led to a grayish and desaturated colorization, because the optimal solution is essentially the mean of a set of plausible colors. Working in the CIELAB color space, they proposed a multimodal classification approach where the model was trained to predict a quantized probability distribution of the AB channels gamut. Using a cross-entropy loss function with a weighting term to rebalance the loss for rare color-classes, they could account for the strong bias towards desaturated colors.

Levels of Automation Fully automatic methods are able to produce good results for natural image colorization, with the work by Zhang et al. [51] achieving 32% fool rate on test subjects. Challenges arise in scenes that do not conform to higher level semantics like grass is usually green, and a clear sky is blue. Man-made objects like clothes, cars and buildings may have a wide variety of different colors, often totally outside any semantic context derivable from the scene or present in the training set. This also applies to SEM imagery, where specimen in many cases are far too small to be perceived by optical microscopy, meaning that ground truth color, if any, would be hard to obtain. Colorization is an ill-posed problem, and fully automatic methods can not entirely account for all its uncertainty, and can be prone to producing results not meeting specific requirements of the user, e.g., wanting to highlight a structure with green color in a SEM image.

3.1.2 Real-Time User-Guided Image Colorization With Learned Deep Priors

The problem of ambiguity was one of the driving factors for the work of Zhang et al. [52]. Proposing a semi-automatic colorization method, they leveraged the speed and quality of recent automatic colorization solutions, but allowing the user to guide the colorization by providing a set of inputs.

A U-net was used as the main colorization network which receives the L channel of the image to be colorized. Two variants of the network were trained to accommodate different user inputs. One "global hints network" θ_g^* where the user provides a reference image of which its global histogram and average saturation is used as input in addition to the L input of the main network. The other "local hints network" θ_l^* is provided with user generated inputs $\boldsymbol{U_l} = \boldsymbol{\{X_{ab}, B_{ab}\}}$ where $\boldsymbol{X}_{ab} \in \mathbb{R}^{HxWx2}$ is a sparse tensor consisting of user provided AB values, at the locations specified in the binary mask $\boldsymbol{B}_{ab} \in \mathbb{R}^{HxWx1}$. The local hints network also has a sub structure to the main U-net that is trained to output a histogram of AB values for each pixel in the grayscale input image based on the likelihood.

During training of the local hints network, simulated user inputs are generated. Authors expected users to click on points closer to the center of images, thus point locations for the binary mask $\boldsymbol{B}_{ab}$ are sampled from a 2D-Gaussian where $\mu = \frac{1}{2}[H, W]^T$ and $\Sigma = \left(\left[\left(\frac{H}{4}\right)^2, \left(\frac{W}{4}\right)^2\right]\right)$. Average AB values are sampled from the ground truth image from uniformly drawn 1x1 to 9x9 patches around the point location and stored in $\boldsymbol{X}_{ab}$. The number of generated points are drawn from a geometric distribution with $p = \frac{1}{8}$, meaning that 12.5% of training instances have no user input, to ensure that the network still is able to perform fully automatic colorization.

3.1.3 Image Colorization With Diffusion Models

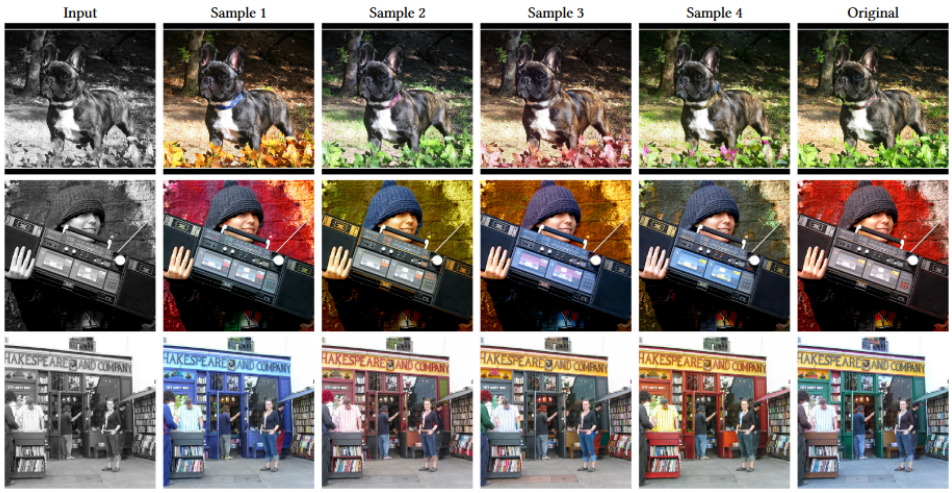

Figure 3.1: Colorization output from Palette, showing differently generated samples from their respective conditioning input. Figure sourced from Saharia et al. [37]

Palette: Image-to-Image Diffusion Models The work done by Saharia et al. [37] is a good example of the versatility of diffusion models in the field of computer vision. It is a unified diffusion model framework for the tasks of colorization, JPEG restoration, uncropping and inpainting as illustrated in Figure 3.2. The approach is able to outperform regression and GAN correspondents when trained for individual tasks, as well as achieving competitive results when trained on all tasks simultaneously.

Palette's models are conditional diffusion models that are on the form $p(\mathbf{y}|\mathbf{x})$ where y is the output image and x as an input signal, an image, that the models is conditioned on. In contrast to the approximated backward process for DDPM's in Equation (2.10), Palette iteratively removes Gaussian noise from an image y_T conditioned on an input image x by sampling $p_\theta(y_{t-1}|y_t, x)$ until a final color image y_0 is produced.

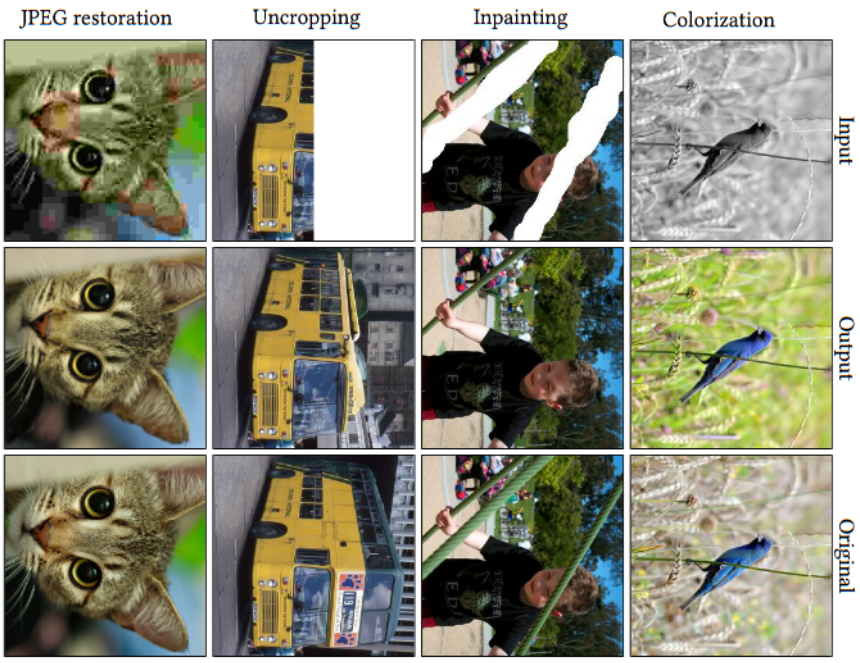

Figure 3.2: Palette task examples. Figure sourced from Saharia et al. [37]

To achieve conditional generation, Palette is trained by concatenating x onto y_t during training per. Saharia et al. [36]. They use a U-net architecture following the work of Ho et al. [14], but increasing the depth and adding self-attention blocks at 32x32, 16x16 and 8x8 resolutions, instead of only at 16x16.

To train Palette for the colorization task, grayscale images were used was used as conditioning signal. The authors opted for the RGB color space, instead of the more conventional CIELAB color space to maintain generality when training for the other tasks. Palette performed well compared to other automatic colorization solutions, achieving 45.9% average human evaluation fool rate on the ImageNet test set with 5 second displays, beating the scores of PixColor [11] and ColTran [19] of 26.0 % and 34.9%, respectively. Figure 3.1 illustrates Palette's ability to generate diverse colorizations.

Color Diffusion by Millon [26] is another diffusion model for the task of automatic image colorization. In contrast to Palette, Color Diffusion is works in the cieLAB color space, and the diffusion process is applied to the AB color channels only. The L channel is used as a conditioning signal, and the model is trained to predict the AB channels of the color image, resulting in a colorized image.

Concurrently with our work, a solution for semi-automatic colorization with diffusion models has now been proposed by Liu et al. [23]. They leveraged a pretrained text-to-

image latent diffusion model, Stable Diffusion [33], which is trained on large scale text-image datasets, and exploit its learned semantic information between text and natural images as priors for colorization.

Latent diffusion models circumvent the computationally expensive problem of working in the high dimensional space of images, by exploiting the fact that modern Variational-Auto-Encoders(VAE) are able to reconstruct images from a low dimensional latent space.

Instead of having a de-noising network working on relatively high dimensional images, the same noising and de-noising process can be applied to the latent vectors from the encoder network of a VAE.

Their approach is to train a "diffusion guider" $\mathcal{F}_\theta(I_g, I_h)$ where I_g is the grayscale images, and I_h are the user hint maps similar to the work of Zhang et al. [52]. $\mathcal{F}$ guides the backward diffusion process of the pretrained latent diffusion model $\epsilon_\theta(t, text)$ to generate colorized images I_c. The diffusion guider is of a similar architecture to the pretrained model, but having an encoded latent vectors of grayscale images and user hints as input.

Artifacts and loss of textural and structural information may not be a huge concern when encoding latent image-vector generated by a diffusion model, where there is no original image to compare to. Colorization with generative models, on the other hand, is a task where one would like to have textures and structures intact, and the generative part only entail the color information of the image. I_h and I_g are encoded using a custom grayscale encoder $\mathcal{G}$ that works with decoder $\mathcal{E}$ of the VAE used with the pretrained diffusion model. By adding grayscale feature maps from the down-sampling layers of $\mathcal{G}$, into corresponding up-sampling layers of $\mathcal{E}$, the authors ensured pixel-aligned colorization of the grayscale images when decoding the de-noised latent vectors.

During training weights of ϵ_θ and $\mathcal{D}$ were frozen to reduce complexity and speed up training. 50% of text captions of training images were removed or replaced with dummy captions such as "color photo" to ensure automatic colorization capabilities.

The result is a diffusion model that can colorize images based on both user provided hints, as well text prompts. Experiments showed superior performance compared to other both semi-automatic and fully-automatic colorization solutions, including the aforementioned Palette [37], and local-hints network by Zhang et al.[52].

3.2 SEM Image Colorization Solutions

Even tough SEM image colorization is mostly done as a post-processing step with general image editing software like Photoshop and GIMP, there are some parametric and/or commercial solutions that are tailored to the task.

Goytom et al. [10] proposed two parametric methods for colorizing SEM images. One "End-to-end CNN" using Encoder-Decoder architecture with a fusion layer where outputs of a pretrained image classifier, InceptionV3 [41], are concatenated together with the output of the decoder, based on the work of Iizuka et al. [16]. The network is trained to predict the AB channels of grayscale SEM images, leveraging feature extraction capabilities of the pretrained classifier. Since there were no large scale datasets of colorized SEM images publicly available, the authors argued that features extracted with this classifier trained on 1.2 million images from ImageNet [35] could compensate for the low amount of available training data.

The other method dubbed "Neural Style Transfer CNN", uses colors from a second input image as a reference for the colorization process. An encoder-decoder network is trained to predict the AB values of a single reference image. After convergence the same network is fed a grayscale SEM image. This AB output is then put through a post-processing step where similar L channel values are assigned one AB color value from the output, presumably some kind of averaging operation. The authors argued that it gave a more uniform colorization of the background, though they did not provide much insight into this process.

MountainsSEM is a commercial software solution for SEM image data analysis. Features include 3D topography reconstruction, measurement tools and SEM image enhancements by colorization. The colorization functionality is powered by a user adjustable segmentation and object detection algorithm, details on the implementation are not publicly available at this time. The user load SEM image data, and segmentation maps around objects and structures are automatically generated. The user can click on a structure in the image with their chosen color, and the object will be colorized as such. With the help of the object recognition technology, the auto-colorization feature can infer the colorization of the rest of the image, based on objects shape or size. With the highlight option, the rest of the image will be colorized in a contrasting color to the selected object.

Chapter 4

Methodology

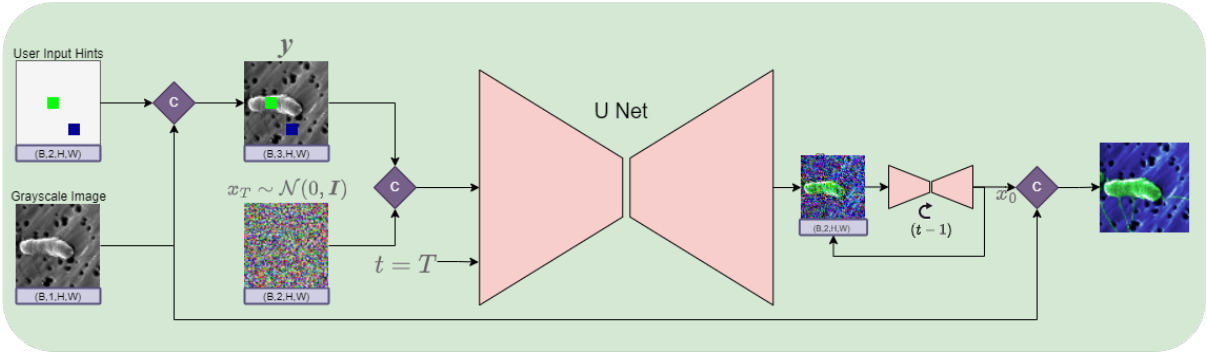

Figure 4.1: Diagram of the AB diffuser inference process. Conditional signal y consists of the grayscale image's "L" channel, concatenated with optional user provided color hints. Vector x_T consisting of pure Gaussian noise is de-noised through the reverse diffusion process until generated color data x_0 is produced.

'AB Diffuser' is a conditional probabilistic diffusion model for SEM image colorization. It is capable of both fully automatic and semi-automatic colorization if provided with user inputs. AB Diffuser colorizes SEM-images by generating AB color-channels in CIELAB color space, conditioned on the grayscale images themselves with a set of optional user provided color hints, granting them influence over the colorization process. We have also included a simple GUI running in Jupyter Notebook, where users can load their grayscale SEM-images, colorize them and export the results. AB-diffuser is trained to colorize SEM images at resolution 256x256, but is capable of colorizing images of arbitrary aspect ratios as well.

In this chapter, we will discuss our approach for colorizing SEM images with diffusion models. We begin by describing the various components and concepts facilitating the

training and inference process, such as data, color space and user input generation. Then we will take a closer look at the architecture of the training and inference pipelines, and the U-net itself. Lastly we will present our GUI implementation, enabling user interaction with the model.

4.1 Data

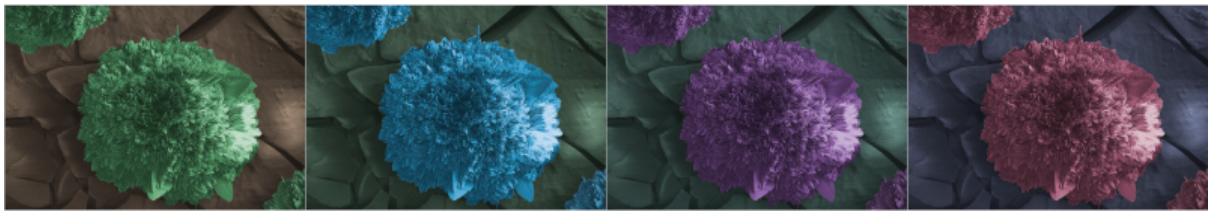

Figure 4.2: Colorized SEM images of pollen grains from our SEM dataset. The original colorization (a) augmented with color jittering to increase the color diversity of the training data (b-d).
Source: LorDesposti, CC BY 4.0 ¡https://creativecommons.org/licenses/by/4.0¿, via Wikimedia Commons

Multiple datasets were used to train AB diffuser. The final model was trained on ImageNet ILSVRC 2012 [35], a common subset of ImageNet containing 1.2 million training images spanned across 1000 classes. It also contains 50.000 and 100.000 images for validation and testing respectively. AB diffuser is further fine-tuned on a colorized SEM dataset we put together by combining 600 images from the colorized SEM dataset SEM-COLORFUL1.0 by Goytom et. al [10] with 1400 images scraped from Bing and Google using WFDownloader App [47]. Our dataset contains a total of 2000 images, where 1700 are used for training, 100 for validation and 200 for testing.

To facilitate training this model for the colorization task we use a dataloader that converts the images from RGB to CIELAB color space, and splits the LAB channels into grayscale L and color AB channels. During the fine-tuning process on the limited colorized SEM data, the dataloader is configured to do data augmentation operations, such as random cropping and flipping. Given that colorized SEM images does not have to adhere to the relatively stricter set of semantics and priors that natural image colorization have to follow, color jittering has proved to be a particular useful augmentation. The user should be able to apply any color to an object or structure in the image, and color

jittering will aid the model in learning that by providing it a more diverse set of color representations as seen in fig. 4.2.

All channels are also normalized to the range of $[-1, 1]$ to ensure that the reverse process $p_\theta(\mathbf{x}_{0:T}|y)$ operates on consistently scaled inputs, starting from $x_T \sim \mathcal{N}(0, \boldsymbol{I})$. [14]

4.2 Colorspace

In contrast to Palette by Saharia et al. [37], AB diffuser operates strictly on color data. Operating in RGB color space was a conscious design choice for Palette, it needed consistency across the multiple tasks it performs. The problem with working with RGB in colorization tasks is that it requires the model to effectively generate all the data in the image. This poses a problem when the model is tasked with colorizing SEM images in particular, as we would not want to risk distorting their inherent structural details. SEM images are often utilized for precise measurements, material characterization, and understanding micro-to-nano scale phenomena. It is therefore crucial that we preserve the structural integrity of the underlying SEM data, and only enhance the visualization through colorization.

AB-diffuser does this by limiting diffusion process and U-Net output to only the AB channels like Color Diffusion [26]. To colorize a SEM image, we concatenate the generated color data to the SEM image, whose structure is preserved within the L channel. This is a common approach in image-to-image translation tasks, especially for colorization. [4, 15, 52].

4.3 Hint Generation

Just like using the grayscale image itself as a conditioning signal, we also use a set of user inputs to condition the model.

Inspired by the work on point-click colorization by Zhang et al. [52], we decided to use their approach for hint generation when training our diffusion model. For each image in batch of data during training, the Hint Generator outputs a batch of hint masks $\boldsymbol{B}_{ab} \in \mathbb{R}^{HxWx2}$, where H and W are the height and width of the image respectively. Each hint is a uniformly drawn 1x1 to 9x9 pixel patch. The average A and B color values of

the ground truth image at the hint location is sampled and stored in the mask, while maintaining the rest of the values at 0. Each mask contains n amount of hints drawn from a geometric distribution with probability $p = \frac{1}{8}$, meaning that 12.5% of training instances have no user input, to encourage the model to perform well on images without user guidance. To reinforce that the model should replicate the colors from the user inputs to the output, the whole AB channel of the ground truth image is copied over to the hint mask in 1% of training instances.

Expecting that users will be more inclined to select objects and structures in the center of the image, the hint locations in $\boldsymbol{B}_{ab}$ are sampled from a 2D-Gaussian where $\mu = \frac{1}{2}[H, W]^T$ and $\Sigma = \left(\left[\left(\frac{H}{4}\right)^2, \left(\frac{W}{4}\right)^2\right]\right)$.

4.4 Model Architecture

The goal of this AB-diffuser is to colorize SEM images with and without user guidance, necessitating a conditional design or a way to guide the data generation process. Since colorization is an image-to-image translation task, the model must be able to generate color data that comply to the underlying structure and content of the grayscale images. While other diffusion models usually generates data that can independently be interpreted and understood, be it image or audio-synthesis, the data generated by this model would only become meaningful given the context of a grayscale image.

There are many ways to guide the data generation process of diffusion models towards a desired distribution. We chose a proven approach used by the authors of Palette: image-to-image diffusion models[37] to great effect for not only colorization, but also other image-to-image translation tasks. By simply providing the conditioning signal as an additional input to the U-Net, we can let the attention mechanisms and convolutional layers learn the relationship between the grayscale images, and the input noised color data. Achieving success with this approach for automatic colorization, we extended the conditional signal to also include user inputs.

In this section we will present the model architecture, training and sampling pipelines of AB-diffuser. We will also touch on our chosen noise schedule, and the reasoning behind it.

4.4.1 Training Pipeline

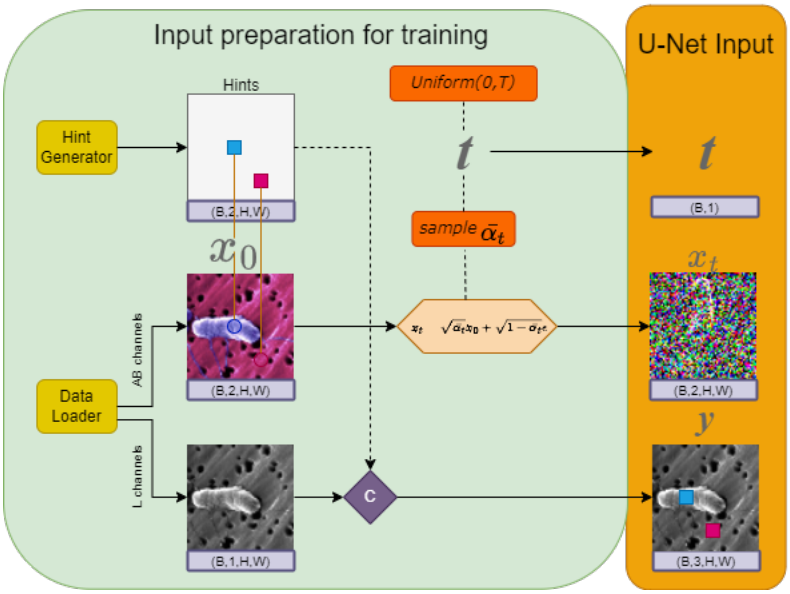

Figure 4.3: Model input during training. y denotes the conditioning signal consisting of the grayscale image's "L" channel, and simulated user hints with colors sampled from the ground truth. x_t denotes the AB color channels of the image at forward diffusion step t. $\bar{\alpha}_t$ denotes the noise schedule. $\boldsymbol{\epsilon}$ denoted the noise

AB-diffuser follows the general training procedure for diffusion models as described in Section 2.2.1, with a few differences due to the colorization task and training objective. For every batch of colorized images, the grayscale images are converted to cieLAB color space, and split into their lightness and color channels, L and AB respectively. Binary hint maps are generated for each image, and the average AB color values of x_0 are sampled at the hint locations. We will refer to the AB channels as x_{0-T} from now on. A uniformly sampled batch of integers $t \sim \mathcal{U}(0, T)$ is used to sample noise schedule $\bar{\alpha}_t$ from wich we compute the forward diffusion step to generate the noised input x_t. The hint maps are concatenated to grayscale images forming the conditioning signal y as seen in Figure 4.3. Inputs t,x_t and y is received by the U-Net $v_\theta(\mathbf{x}_t, t, y)$, which outputs a tensor of shape $[b, 2, H, W]$.

Recalling the simplified loss L_{simple} in Equation. 2.18, where the training objective is the noise term $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$, training the model by minimizing $L_{simple} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y)\|^2\right]$. We use an alternative training objective $v = \sqrt{\bar{\alpha}_t}\boldsymbol{\epsilon} - \sqrt{1-\bar{\alpha}_t}\mathbf{x}_0$ as described in the work of Salimans and Ho [38], minimizing $\|v_t - v_\theta(\mathbf{x}_t, t, y)\|^2$. We find

that, experimentally, using this objective results in faster convergence and more stable training. We discuss this further in Section 5.2.2.

To overcome the problem of having such a limited amount of colorized SEM data to work with, training is done in two stages, a large-scale training session on 1.2 million natural images from ImageNet, followed by a fine-tuning session on our smaller colorized SEM dataset. The first stage ensures that we have a robust model that learns the relationship between the generated color hints and color data, and how to apply them to a large and diverse set of shapes, objects and structures. This would not be feasible with our much smaller dataset of colorized SEM images alone. Details on training, fine-tuning and hyperparameters can be found in the Implementation chapter, Section 5.2.2.

4.4.2 Sampling Pipeline

During inference, a batch of conditioning signal y consisting of the grayscale SEM images to be colorized, and optional user provided color hints is prepared. x_T is sampled from an isotropic Gaussian distribution $\mathcal{N}(0, \boldsymbol{I})$.

The prepared inputs are taken through the reverse denoising diffusion process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, y)$, by iteratively computing $\hat{x}_{t-1}$ until $t = 0$ giving our generated AB channels $\hat{x}_0$. These are concatenated to the L channel of the grayscale image to form the final batch of colorized SEM image.

4.4.3 U-Net

At the heart of our diffusion model is our U-Net. During training, U-Net $\epsilon_\theta(x_t, t, y)$, is tasked with predicting the noise term ϵ added to the input x_0. As illustrated in Figure 4.3, the U-Net receives the conditioning signal y, and a noised input x_t. The U-Net architecture is based on the U-Net used in the original DDPM paper by Ho et al. [14], Each down-and up sampling layers consisting of two convolution ResNet blocks, with linear self attention at each resolution. The timestep t provides the U-Net with context on where in the Markov chain of the diffusion process the input x_t is, and is transformed through a sinusoidal positioning embedding before being concatenated to the input of each ResNet block.

4.4.4 Noise Schedule

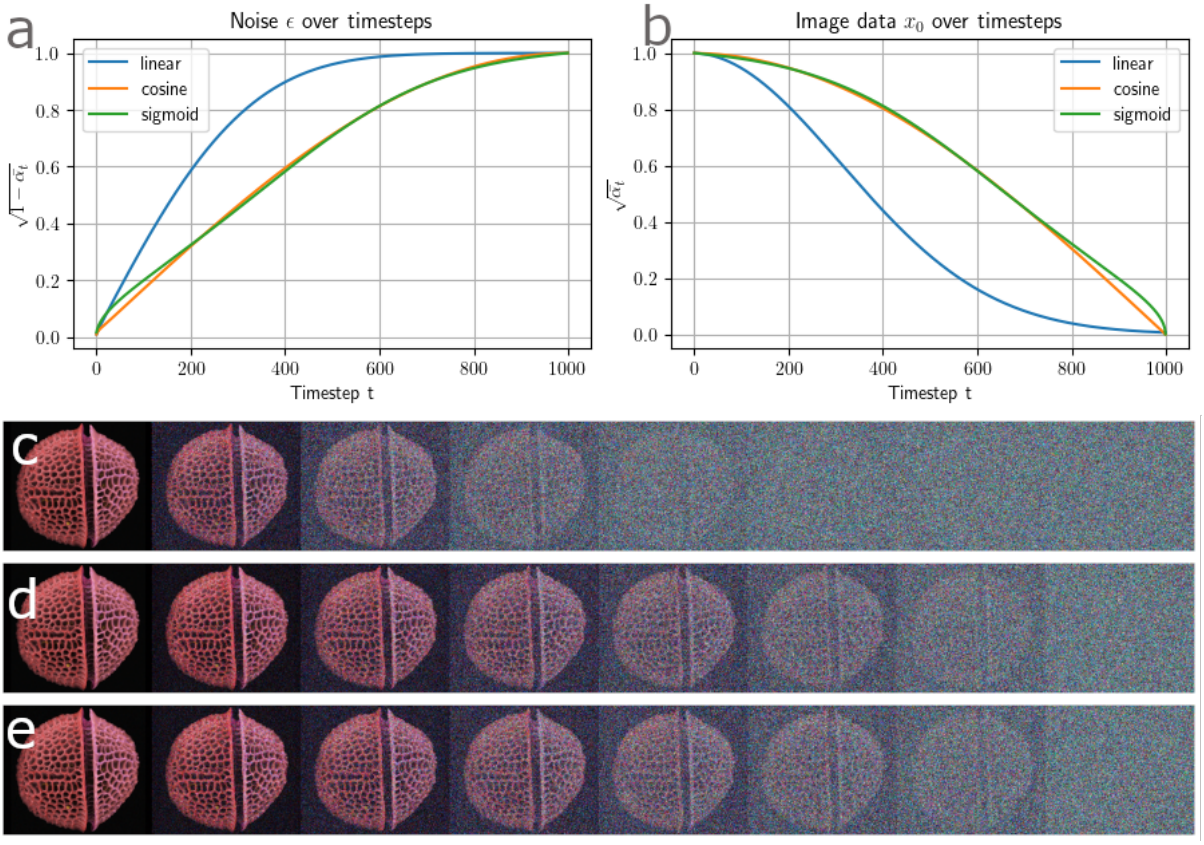

Figure 4.4: Comparisons of noise schedules $\bar{\alpha}_t$ and their effect on the forward diffusion process.
a: The amount of noise ϵ present in x_t over the forward diffusion process with linear, cosine and sigmoid schedules
b: The amount of clean data x_0 present in x_t over the forward diffusion process with linear, cosine and sigmoid schedules.
c, **d**, **e**: Forward diffusion process on an image, linear, cosine and sigmoid schedules respectively.

Our model uses a sigmoid noise schedule as proposed by Jabri et al. [17]. A sigmoid noise schedule was found to increase sampling quality on larger image sizes, and in our experiments increase stability of training and validation loss during training.

Recalling from Section 2.2 that the forward diffusion process is computed as such: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$, where ϵ denotes the Gaussian noise term and x_0 denotes clean image data.

The given amount of noise applied to the AB color channels during the forward diffusion process $q(x_t \mid x_0)$, see Equation 2.7, is determined by a fixed variance or noise-schedule $\bar{\alpha}_t$. As noted by Nichol et al. [27], the linear noise schedule used in the original DDPM paper by Ho et al. [14] has a sharp drop off in the middle of the forward diffusion process, destroying the information relatively quickly. This is illustrated in Figure 4.4 **a** and **c** where one can see that there is no significant increase in noise applied to the image after time step 600, and may cause difficulties for the model to learn the difference of noise levels at those later time steps. The third alternative is the cosine schedule proposed by Nichol et al. [27], tough similar in shape to the sigmoid schedule, we find the training and validation loss to be more unstable than with the sigmoid schedule.

4.5 GUI

A simple GUI was deviced for users to interact with the model with the point-click interaction. The GUI runs within Jupyter notebook, and allows users to load their greyscale SEM image to be colorized. After selecting a color from the palette, the user can click on the image and provide color hints to guide the colorization. The GUI has support for multiple colorization outputs, and an export button.

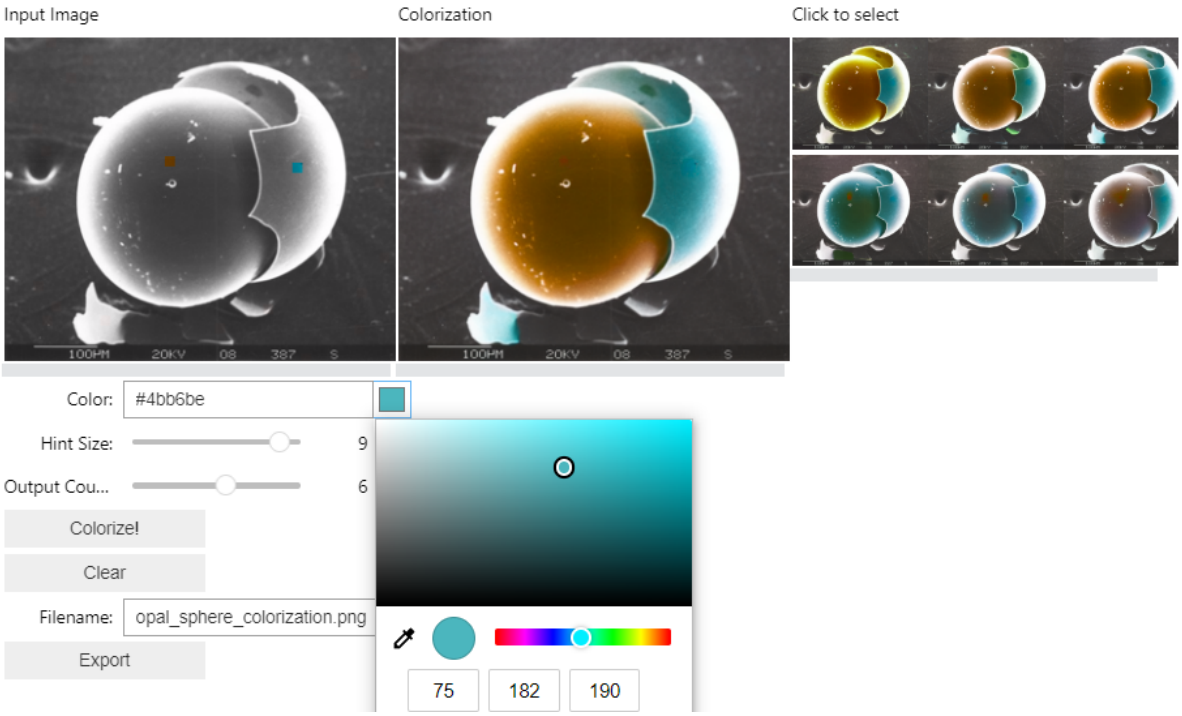

Figure 4.5: GUI for colorizing SEM images with AB-diffuser, running within a Jupyter notebook.
Left: Visualization of the models input, with user provided hints.
Middle: Currently selected colorization for export.
Right: Additional colorization outputs, that one can select by clicking. Image size: 312x256

When clicking the greyscale image with a selected color, they are converted to cieLAB color space, and applied to the hint mask to create the conditioning signal y. y is then duplicated into a batch, whose size is determited by the "Output Count" slider, and put trough the reverse diffusion process to generate the colorized images. By clicking on the additional colorization outputs, the user can select which one to export.

Figure 4.5 shows a run of the model and GUI with 6 colorized outputs. One blue, one brown, and multiple grey hints around the sphere were provided as input along with the grayscale image.

Chapter 5

Implementation and Prototyping

In this section we discuss how we arrived to our solution for a diffusion based SEM colorization model and how we implemented it. First we go into technical detail about or data acquisition. Afterwards we go through the iterative improvements we made trough prototyping and testing, and finally we discuss the technical details of the training process of our final model and GUI implementation.

Note that the considerations and decisions in this chapter were taken before a recent notable contribution to the field of diffusion-based colorization models by Liu et al. [23]. In this work, the authors detail approaches to overcome the challenges of incorporating user inputs, and maintaining structural integrity of the colorized images in latent-space diffusion models.

5.1 Data acquisition

Due to the limited amount of research done on the topic of colorizing SEM images, there exist only one publicly available dataset of colorized SEM images, SEMCOLORFUL1.0 by Goytom et. al [10]. This dataset consist of 800 images, with a 90/10 train and test split. This set contained many duplicate images, and many of the training images were present in the validation set. To clean up this dataset we used Imagededup [1], a python package containing tools for finding and removing duplicate images in a dataset. By moving all the train and test images into one folder, and running Imagededup's perceptual hashing algorithm, we were able to remove 200 duplicate images, leaving us with only 600 unique images.

To give our model more data to learn from, we decided to utilize an image web-scraping tool. There are a lot of tools available for this, with varying degrees of functionality and ease of use. We decided to use WFDownloader App [47], a standalone image scraper, which features a GUI and support for multiple search engines. Using search terms such as "SEM image insect colorful" and "SEM image pollen colorized" we scraped over 3000 images from Bing and Google. Any grayscale images and files that were not in an image format were removed with a python script, and the remaining images were manually inspected to remove any non-SEM images. The last step was to combine the SEM-COLORFUL1.0 dataset with the scraped images, and use Imagededup to remove the remaining duplicates.

This left us with a final dataset of 2000 images, of which 1700 were used for training, 100 for validation and 200 for testing.

5.2 Prototyping

Literature on diffusion-based colorization models is limited. This was especially true for semi-automatic colorization with diffusion models. Using our only point of reference, Palette [37], we decided to start prototyping an automatic colorization model using their conditional approach. Considering that we wanted our model to only operate on color data, with a classical scheme of inputting L component, output AB components and concatenate them, we still needed to test if this was feasible with diffusion models.

Our objectives with these prototypes where to test the feasibility of our approach of limiting the diffusion process to the color channels in cieLAB color space, and if we could achieve user guidance by conditioning the model on generated color hits.

We modified an implementation of DDPMs by Wang [44]. The repository contained a Pytorch implementation of DDPMs, with support for multiple types of noise schedules, training objectives and various other diffusion models. It also contained infrastructure for large scale training, so modifications to appropriate the model to our conditional architecture was relatively simple.

First we wrote a custom Pytorch dataset class that would convert images to cieLAB color space, and separate the "L" channel from the "AB" channels. To ensure that the reverse process $p_\theta(\mathbf{x}_{0:T})$ operates on consistently scaled inputs, starting from $p(x^T) \sim \mathcal{N}(0, \boldsymbol{I})$ all channels were normalized to $[-1, 1]$ [14]. The functions related to the backward

and forward diffusion process $q(x_{1:T} \mid x_0)$ were modified to only operate on the "AB" channels, until the Unet received them as input. The U-net was configured to output 2 channels, corresponding to the "AB" channels of the color data.

5.2.1 Training the First Prototype

For the training data used the relatively lightweight "Tiny-ImageNet" dataset, a subset of the original ImageNet [35] consisting of 100k 64x64 natural images of 200 classes (500 images per class). The dataset was also supplemented with 1000 colorized SEM images down scaled to 64x64, from the dataset described in Section 1.1.

At this early stage, we were interested in testing the colorization capabilities of the model, and not how to best utilize the SEM dataset, relying on the class imbalance introduced by the additional 1000 colorized SEM images for now. We used a dataset for monitoring the model's colorization outputs during training, consisting of 10k images from Tiny-ImageNet's validation set, and 200 images from our SEM validation set. For logging and visualization of colorized images during training process we used Weights and Biases [45]. Training was done on a single Nvidia RTX 3080 10gb GPU, with a batch size of 128, Adam optimizer, and a learning rate of 1e-4 50k steps.

For this proof of concept we set the parameters and objectives to standard values from the literature. Timesteps T was set to 1000 and training objective was the noise term ϵ_t, and a cosine noise schedule.

Results

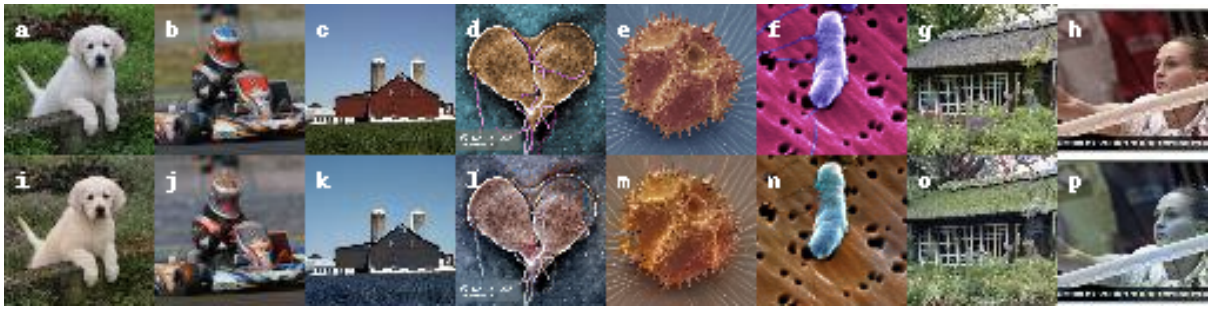

Figure 5.1: Colorization outputs from the initial prototype.
Ground truth: a-h, model outputs: i-p, image size: 64x64

The resulting colorization outputs looked very promising, with the model able to generate plausible color Data for natural images, albeit with some failure cases which were especially prominent with human subjects. The Colorized SEM images were also satisfactory.

5.2.2 Support for User Inputs

After the initial prototype proved to produce satisfactory automatic colorizations, we moved on to the second objective, adding support for user inputs. Simulation of user inputs was done in the same manner as in the work on point-click colorization by Zhang et al. [52]

For each image in batch of data, a binary hint mask $\boldsymbol{B}_{ab} \in \mathbb{R}^{HxWx1}$ is generated. Details on the hint generation is discussed in the Methodology chapter 4.3.

The model was trained on the same dataset, learning rate, hyperparameters and training objective as the first prototype. It was trained for 50k, with a smaller batch size of 40 due to the extra memory requirements of the hint masks. Gradients were accumulated for 2 steps before updating the model parameters to account for the smaller batch size. This took approxamently 25 hours on a single Nvidia RTX 3080 10gb GPU.

Results

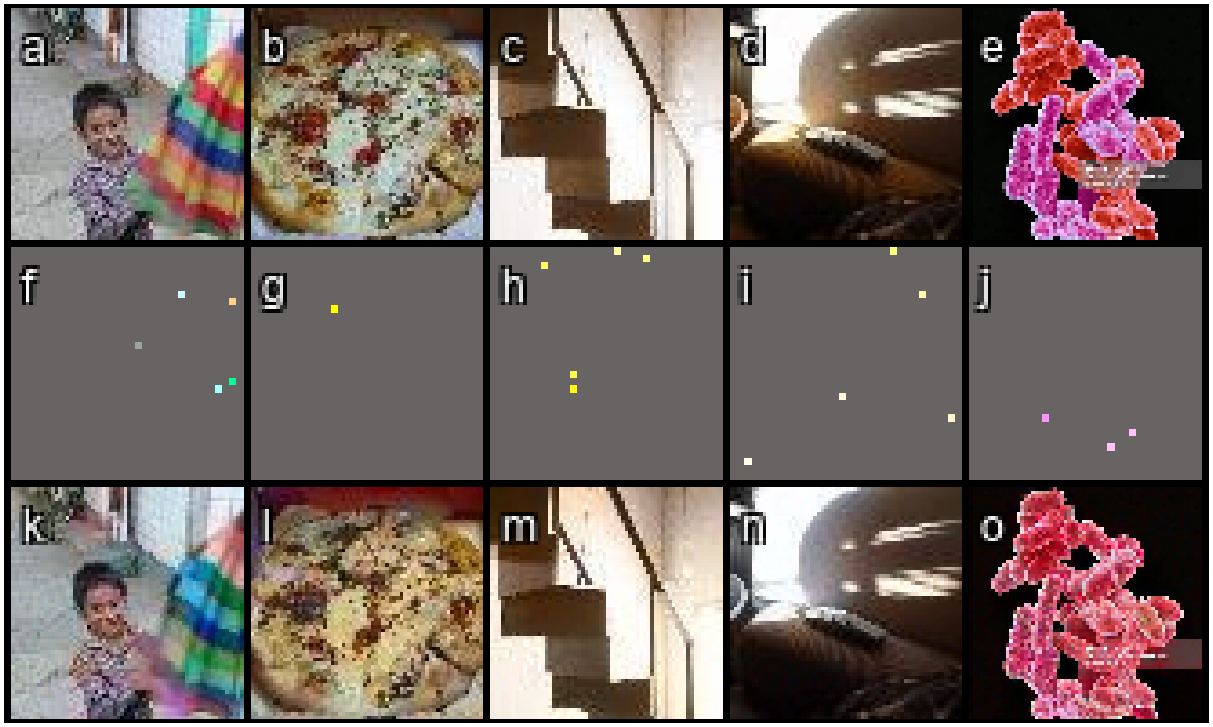

Figure 5.2: Colorization outputs during training with user hints. Ground truth: a-e, generated hints: f-j, model outputs: k-o image size: 64x64

The resulting colorization outputs were also in line with our hopes for this prototype, generating plausible color data for both natural and SEM images.

Evaluating the Prototype

Time constraints limited the amount of testing we could do, two noise schedules, cosine and sigmoid, and two training objectives, v and *epsilon* were considered. Three model candidates were trained for 100k steps on the Tiny-ImageNet training set at 64x64 resolution, and evaluated on its validation set.

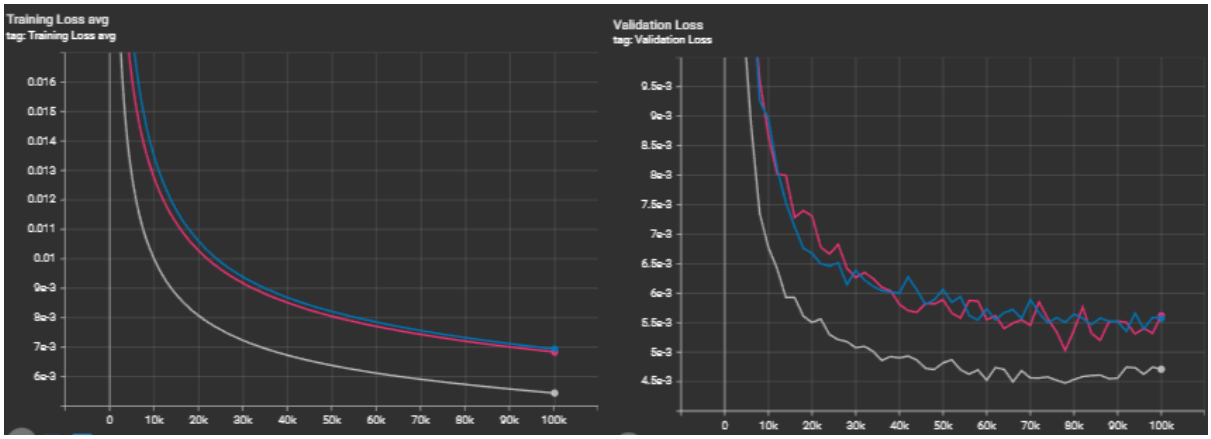

Figure 5.3: Blue: cosine noise schedule, ϵ objective. Purple: Cosine noise schedule, v objective. Gray: Sigmoid noise schedule, v objective.

Figure 5.3 shows the loss curves for the candidates, with the sigmoid noise schedule with v objective achieving the lowest validation and training loss.

Each model was evaluated on the validation set of Tiny-ImageNet. The models colorized each batch twice, with and without generated color hints, so we could evaluate both automatic and semi-automatic performance.

Peak signal to noise ratio (PSNR) and structural similarity metric (SSIM) are two common evaluation metrics for image colorization. They are both full-reference methods, meaning that the colorized images are evaluated against their ground truth. SSIM evaluates the perceptual difference between the ground truth and colorized images based on their differences in luminance, contrast and structural content. Higher SSIM indicates higher perceptual similarity between the ground truth and the output, and is given by:

$$\text{SSIM} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad -1 \leq \text{SSIM} \leq 1 \tag{5.1}$$

Where:

- μ_x and μ_y are the mean values of the pixel intensities in the ground truth image and the colorized image, respectively. These terms represent the average brightness of the two images.

- σ_{xy} is the covariance of the pixel intensities between the ground truth and colorized images. This term quantifies how changes in pixel intensities are related across the images and relates to the structural similarity of image pairs.

-σ_x^2 and σ_y^2 are the variances of the pixel intensities in the ground truth and colorized images, analogous to contrast. C_1 and C_2 are constants added for numerical stability.

PSNR is closely related to MSE. It can be interpreted as the ratio between peak possible signal strength (typically 255 for an 8-bit image), and the amount of noise introduced, in this case, by colorizing the image.

$$PSNR = 10log10(\frac{MAX^2}{MSE}) \tag{5.2}$$

The noise is measured in MSE, hence, higher values of PSNR equates to higher similarity between the image pairs.

As these metrics are measurements of pixel-level differences between ground truth and generated images, they will serve as an indicator on how well the model conditions on the user hints

Metric	**Cosine** v	**Sigmoid** v	**Cosine** ϵ
Automatic Colorization			
MSE ↓	0.016	**0.012**	0.014
PSNR ↑	20.58	21.28	**23.30**
SSIM ↑	**0.90**	0.89	0.87
Colorization w. User Hints			
MSE ↓	6.7e-3	**6.0e-3**	8.0e-3
PSNR ↑	24.03	**24.85**	24.58
SSIM ↑	**0.929**	**0.929**	0.928

Table 5.1: Evaluation metrics for model candidates.

Table 5.1 shows the resulting metrics for the candidates. Each candidate performed similarly, with sigmoid schedule and v objective achieving marginally better results . For colorizations with generated hints, we observed better results across all evaluation metrics, suggesting that the hints played a role in guiding the colorizations.

To test if the prototype was utilizing the provided hints as intended we implemented a GUI to interact with the model. The GUI was written in Python using IPyWigdet for Jupyter notebook. The GUI allows for a user to load a grayscale image, and then click on the image with a selected color to provide hints for the model. The hints are then passed to the model together with the grayscale image.

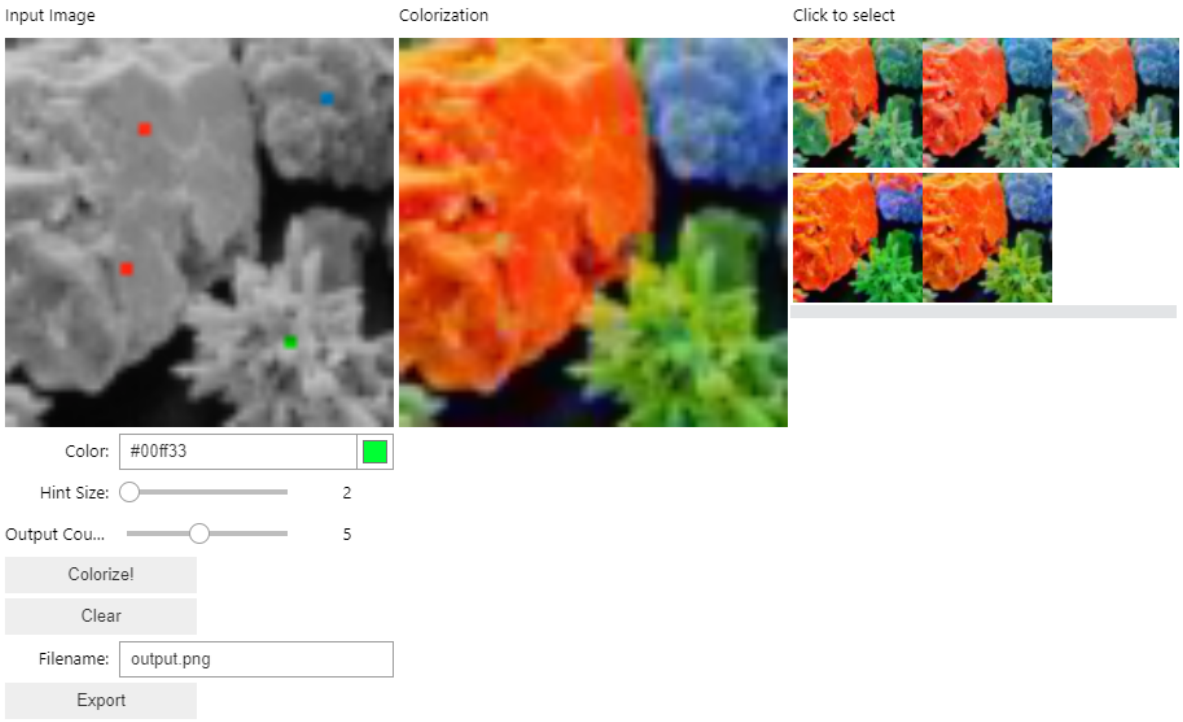

Figure 5.4: GUI for colorizing SEM images with AB-diffuser, running within a Jupyter notebook.
Left: Visualization of the models input, with user provided hints.
Middle: currently selected colorization for export.'
Right: additional colorization outputs, controlled by a slider. Image size: 64x64

There were no significant differences between the models when tested with the GUI, all propagating color from the hints to the regions and structures well.

5.3 Training the Final Model

Moving over to full scale training, we decided to use an implementation of DDPMs by Wang [44]. The repository contained a Pytorch implementation of DDPMs, with support for multiple types of noise schedules, training objectives and various other diffusion models. It also contained infrastructure for large scale training, so modifications to appropriate the model to our conditional architecture was relatively simple.

As mentioned in Section 1.1, we performed the training of AB-diffuser in two stages. During the first stage we trained the model on the full ImageNet training set on 256x256 resolution, and an effective batch size of 64 with gradient accumulation, for 100k steps.

We used a learning rate of 1e-4, with ADAM optimizer. Training took approximately 60 hours.

The fine-tuning stage was done on the SEM dataset, with a batch size of 32 for 20k steps, which took 6 hours. The data was augmented with random cropping, rotations, flips and color-jittering. We kept the weights as they were, while reducing the learning rate to 7e-5 to avoid over fitting on the much smaller dataset.

Chapter 6

Experiments and Results

In this chapter we discuss our conducted experiments, and presentqualitative results of the performance of AB-diffuser on SEM image colorization.

Although the primary focus of this thesis i SEM image colorization, we also evaluate AB-diffusers performance on natural image colorization, by comparing quantitative results against a set of user-guided colorization models.

All experiments are conducted on a NVIDIA A100 80 GPU, and images are sampled from 1000 diffusion steps.

6.1 SEM Image Colorization Performance

AB-diffuser was trained to colorize SEM images, with one of the aims being to enhance the interpretability of the data presented in them. Authors will often discuss their subject's morphology, using symbols and/or descriptive text to communicate their findings.

In this section, we present the performance of our approach on published SEM image results, where authors did not use color to present their findings. Since this is one of the main reasons one would colorize SEM images, we want to assess if we can capture the referenced features by providing AB-diffuser with color hints, or trough automatic colorization without hints.

Afterwards, we validate our model against well established methods, by replicating SEM images that have been colorized with tools such as MountainsSEM and Photoshop.

For each colorization attempt we produce 9 outputs, this lets us better review our hint placements effect on the output, and infer how we can adjust their position and color value to iterate towards an desirable result. For further consistency, all hints provided are of size 9x9. Figure 6.1 depicts our setup.

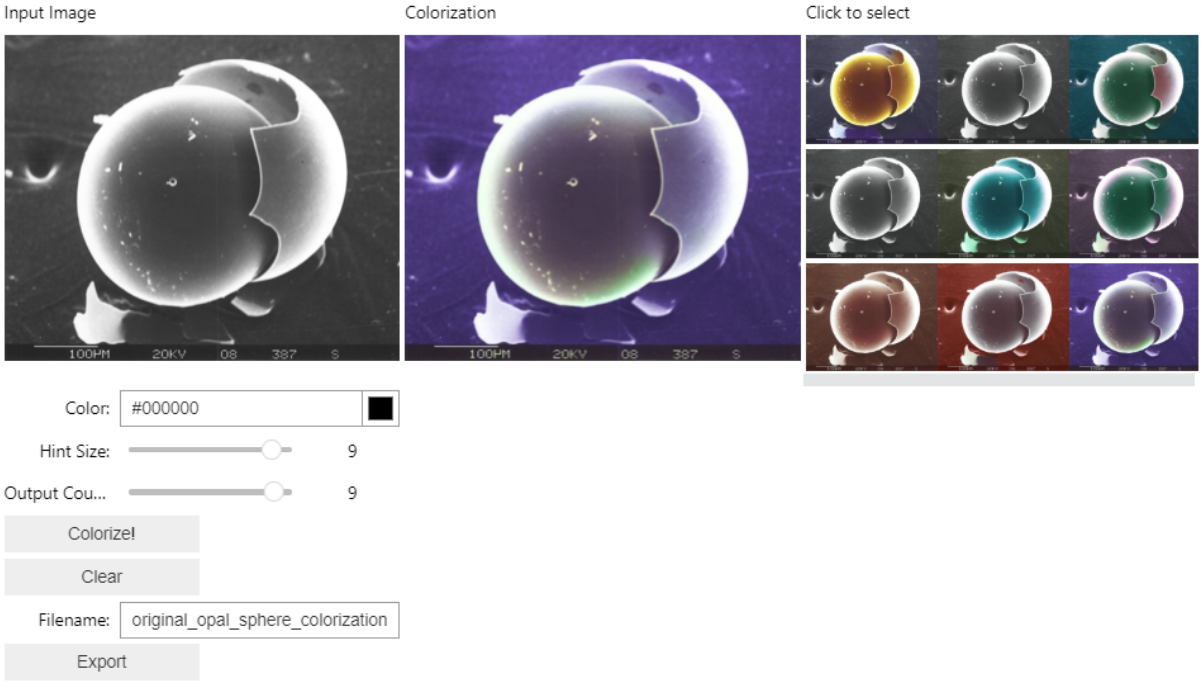

Figure 6.1: Example of our GUI setup for when conducting experiments. Note the lack of color hints, this is an automatic colorization.

A desirable result is achieved when the regions discussed in the text and image captions, or highlighted via markers and symbols are colorized in a visually appealing manner with the intended colors we provide with hints. In addition, we perform automatic colorizations, where a desirable result is achieved if we deem colorization to have visually enhanced the image, or if the colorization captures any prominent or important features. We also measure the time it takes to complete each colorization, while counting the attempts.

Note that during the initial experiments, the expected behaviour of the model was to assign contrasting colors to the regions not provided with color hints. We found that populating these areas with "gray" hints helped the model restrict its colorization to the desired areas. "Gray" refers to AB values close to zero, indicating minimal color for the model. This approach became a consistent strategy during the following experiments.

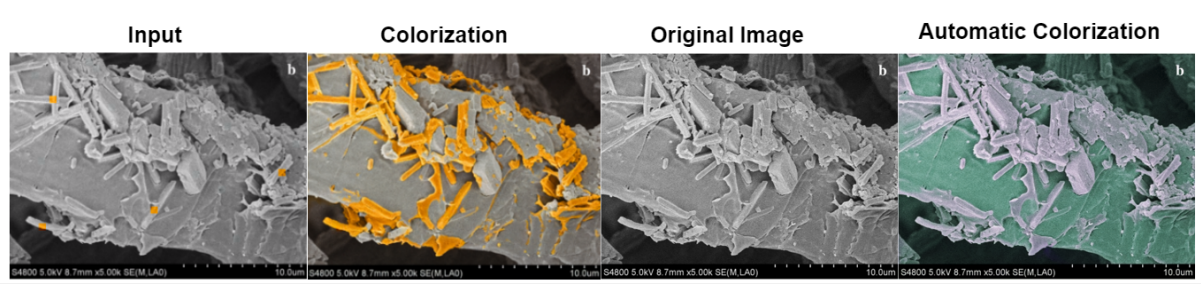

Figure 6.2: Rod-Like particles (orange) on Chinese handmade Lajian paper-fibers, colored green in the automatic colorization result.
Source: Figure colorized using AB-diffuser and adapted from Luo et al. [24] (licensed under CC BY).

Figure 6.2 from Luo et al [24] depicts rod-like particles surrounding fibers from the Lajian fibers. Through energy-dispersive spectroscopy, authors determined that these particles mainly contain carbon and oxygen, suggesting that they might be organic dyes. The paper itself is described by the authors to have an "orange-red" color, we reflect that by coloring them orange.

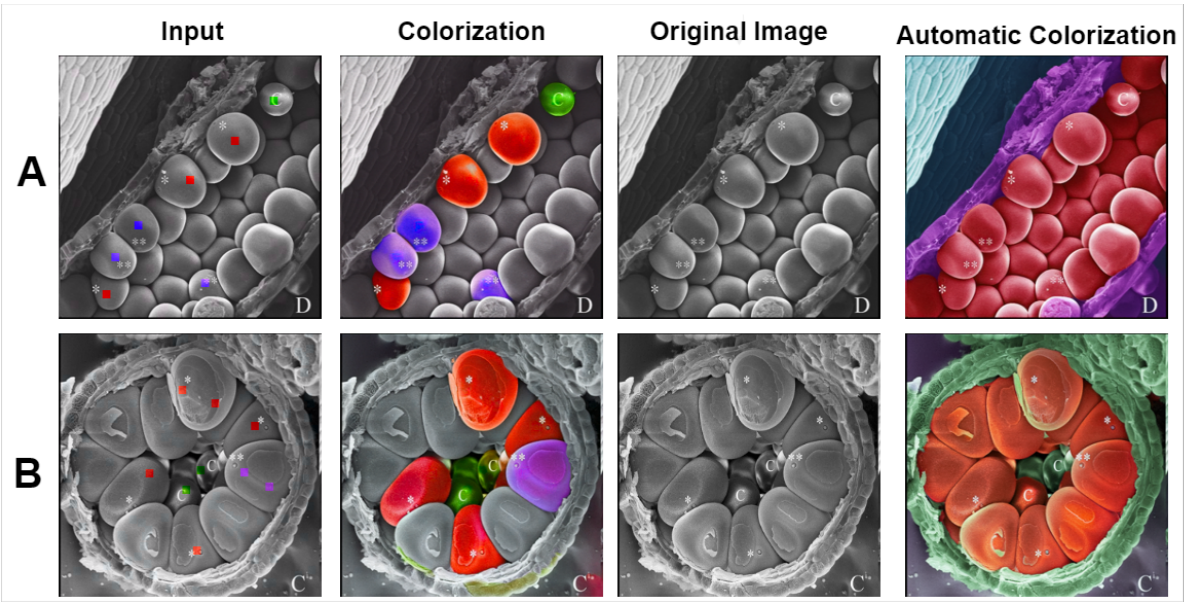

Figure 6.3: SEM images depicting morphology of maize athers. Red grains have laterally facing apertures, purple grains have their towards the central cavity. Central, immature grains are colored green.
A and **B**: Longitudinal and transverse cross-section respectively, of maize athers containing pollen grains. **Source**: Figure is adapted from Tsou et al. [43] (licensed under CC BY).

In Figure 6.3 from Tsou et al [43], two SEM cross-sections of maize-anthers, containing pollen are presented. Here the authors have opted for symbols to point out the

relevant features of the image. Some mature, peripheral grains have their aperture facing laterally, or towards the central cavity marked "*" and "**" respectively. The apertures are minuscule, and may be challenging to discern for those unfamiliar with maize-pollen anatomy, see Figure 6.4.

Central immature grains, marked with "C", are smaller and spherical, with a specular appearance due to their lower content of light-scattering starch-granules. For each type of marking we assign one color, this maintains consistency between the longitudinal and transverse cross-sections.

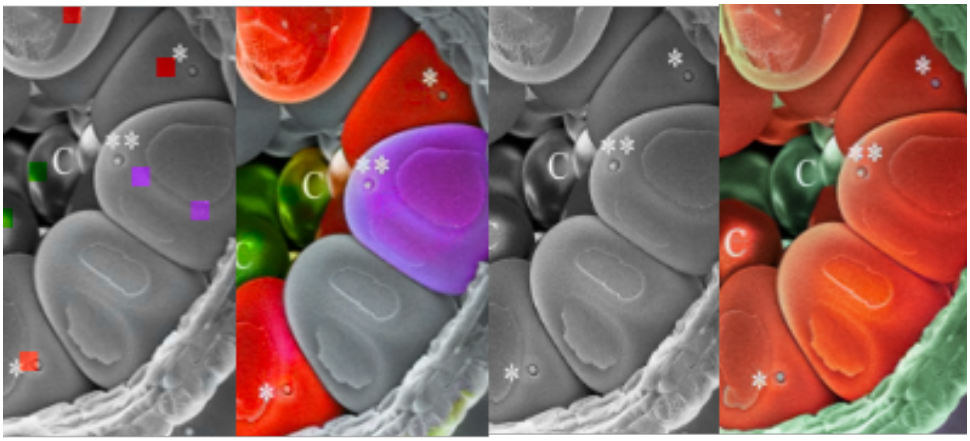

Figure 6.4: Close-up on pollen apertures in Figure 6.3 **B**.
From left to right: Hint placements, semi-automatic colorization with hints, orignal grayscale image, automatic colorization. Notice the red hint placed directly on an aperture, the model is still able to isolate it. **Source**: Figure is adapted from Tsou et al. [43] (licensed under CC BY).

Sushma et al. [40] presents an approach of tuning surface morphology of europium-doped strontium zirconate to attain red-emitting luminescent properties by applying concentrated aloe vera gel to the particles. Figure 6.5 illustrates the structural changes on the particle surface from increasing amounts of gel. In the text, the surfaces are described growing "hexagonal disc-like assemblies" and "pyramidal assembly units" for Figure 6.5 **A** and **B** respectively, and we color these structures red.

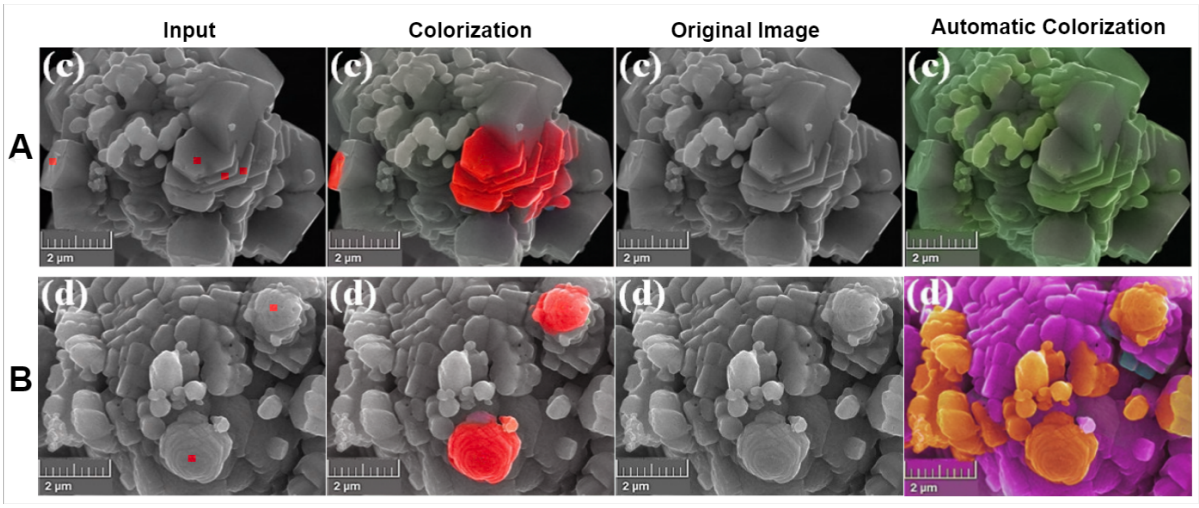

Figure 6.5: SEM micrograph depicting changes surface morphology of europium-doped strontium zirconate, as a result of adding increasing amounts of aloe vera gel
A 10ml AV gel, **B** 15ml AV gel.

Source: Figure is adapted from Sushma et al. [40], used with permission from Elsevier.

Validation In order to validate our results and compare AB-diffuser agains a baseline, we replicate colored SEM micrographs from various published works that have been colorized with well established tools. To confirm that none of these colorized SEM images are present in our SEM dataset, we apply imagededup's [1] CNN trained to detect image duplications, followed by manual inspection of our dataset.

Figure 6.6 depicts our results against SEM images colorized with MountainsSEM, the state of the art tool for SEM micrograph analysis and colorization. MountainsSEM has many features lacking in AB-diffuser, who can only rely on provided hints to generate desired results.

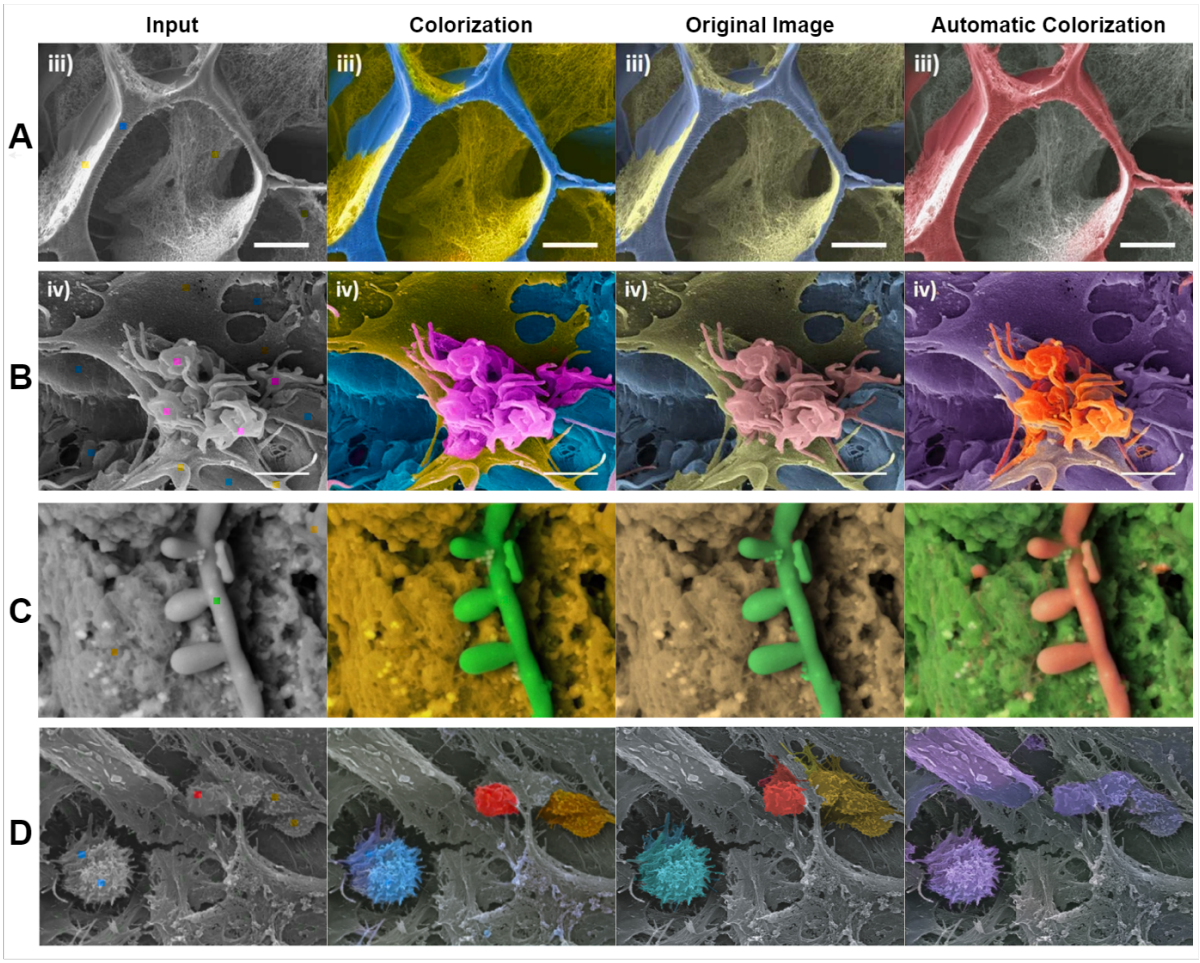

Figure 6.6: Qualitative comparison against colorization made with MountainsSEM
Source: **A** and **B** are adapted from Fernández-Colino et al. [8](licensed under CC BY); **C** and **D** are adapted from Relucenti et al. [31, 32](licensed under CC BY);

Due to time constraints we are not able to conduct our own timed testing with MountainsSEM to properly compare the two methods. We have one, timed example of a SEM image colorized with MountainsSEM, from their own demonstration video. Figure 6.7 displays the results. According to the video, the initial segmentation borders took 1 minute to set up, 12 minutes were spent on manually editing them, and 3 minutes was spent on the actual colorization. The process took 15 minutes in total. We spent one attempt each on the automatic, and semi-automatic colorization with color hints, taking 2:58 and 3:20 minutes respectively. The model used 2:30 minutes to process the colorizations each time.

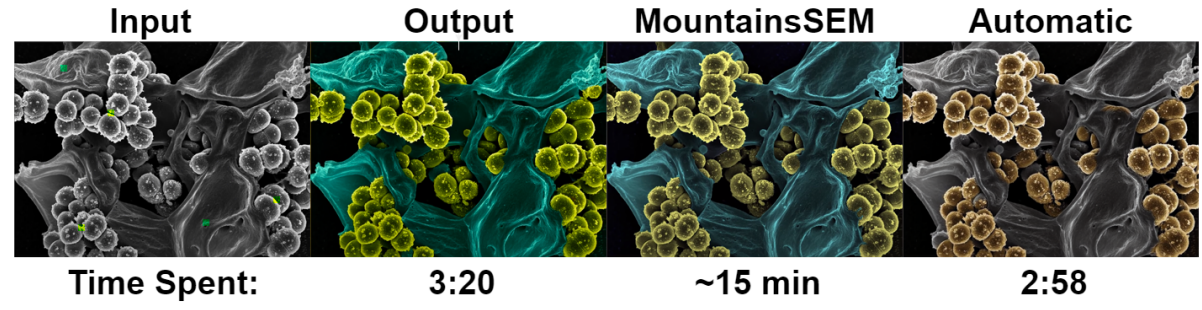

Figure 6.7: Timed Comparison with MountainsSEM
Source: Figure adapted from https://youtu.be/ROZlaOkHmoo?t=81

In Figure 6.8, our results are displayed against SEM images colored with Photoshop, a common tool for manual SEM image colorization.

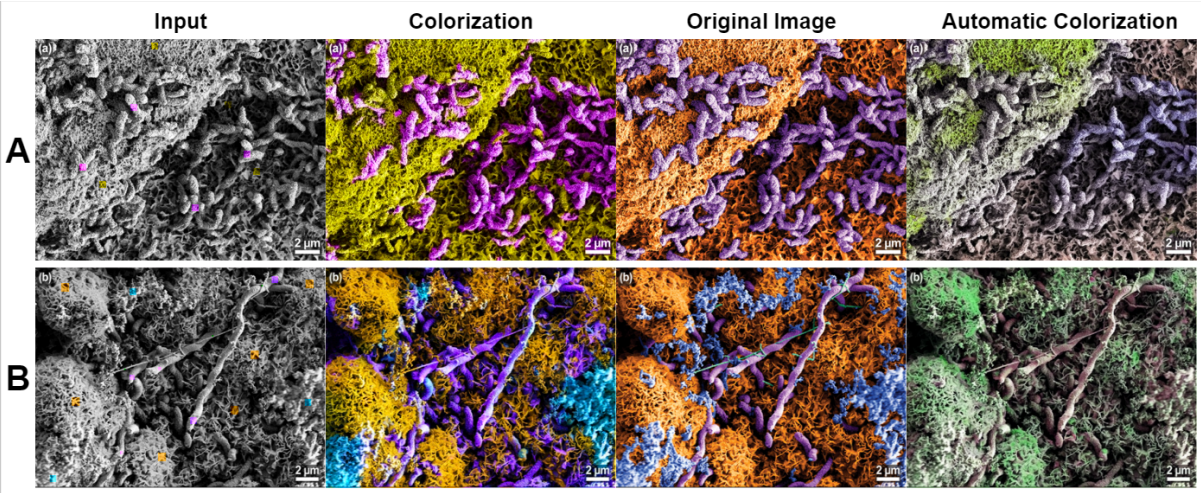

Figure 6.8: Comparisons between our results and Manual colorization with Photoshop.
Source: **A** and **B** are adapted from Krantz et al. [18], used with permission from Taylor & Francis.

Table 6.1 provides an overview of the recorded times and attempts for each colorization method. For the semi-automatic method, which involves user guidance, the recorded time starts when the first hint is applied, and ends when the export button is clicked. For automatic colorization without user guidance, the timer begins upon the first click of the "colorize" button in the GUI (See Figure 6.1). On average, the semi-automatic method took 9:41 minutes and approximately 3 attempts for each completed colorization. With each attempt averaging 3:31 minutes, the model required 2:30 minutes to process a batch of 9 images. This indicates that roughly 1 minute per attempt was dedicated to user interactions, such as selecting colors from the palette, applying hints, and reviewing the batch of 9 colorizations.

Figure	Semi-Automatic		Automatic	
	Attempts	**Time (min:sec)**	**Attempts**	**Time (min:sec)**
6.2	2	7:22	1	2:57
6.3 A	3	10:25	1	3:09
6.3 B	3	11:01	2	6:09
6.5 A	4	14:07	4	11:55
6.5 B	2	7:09	1	3:04
6.6 A	2	6:48	1	3:08
6.6 B	5	17:33	2	5:48
6.6 C	1	3:16	1	2:59
6.6 D	3	10:24	3	9:21
6.8 A	3	10:09	4	12:21
6.8 B	4	14:43	2	6:00
6.7	1	3:20	1	2:58
Average	2.7	9:41	1.9	5:49
Avg/Attempt	-	3:31	-	3:02
Fastest:	6.6 C	**Slowest**:	6.6 B	-

Table 6.1: Time taken and number of attempts for colorization of each figure. We output a batch of 9 images for each attempt, taking on average 2:30 minutes to process, on a NVIDIA A100 GPU.

6.2 Natural Image Colorization Performance

To evaluate our diffusion models performance on natural image colorization, we follow the IColoriT-protocol by Yun et al. [50], for point interactive colorization benchmarking. We test AB-diffuser on two datasets: CUB-200-2010 [46] and Oxford 102 Flowers [28], containing 3033 images of birds and 1020 colorful flowers respectively. Note that these are only two out of three datasets included in the iColoriT-protocol, where the third is ImageNet ctest10k [20].

Following this approach, hint locations are sampled uniformly, and their size are fixed at 2x2 pixels. Hint densities are set to specific values in the range: 1, 2, 5, 10, 20, 50, 100, and 200 hints per image. PSNR is then computed for each of these hint densities to assess the performance of the model at different amounts of provided hints.

We use the weights from the first training stage for this experiment. The model was trained on ImageNet [35] for 100k steps with an batch size of 64.

We compare AB-diffuser's achieved PSNR against the four-point interactive colorization methods:

1. Side Window Filtering (Optimization model) by Yin et al. [49].
2. Real Time User guided Colorization by Zhang et al. [52].
3. Instance-aware Image Colorization by Su et al. [39].
4. iColoriT by Yun et al. [50].

The three deep learning based models are trained on ImageNet [35], with the same style of interactivity for user guidance. And while they are are suitable baselines to compare AB-diffuser against, they have been trained significantly more than we have been able to, due to time constraints. This is reflected in the results displayed in Figure 6.9, where AB-diffuser achieves lower PSNR than the learning-based models. As PSNR is a measure of distance between image pairs, this lower score indicates that our generated images are not responding to the conditioning of the hints as well as the others.

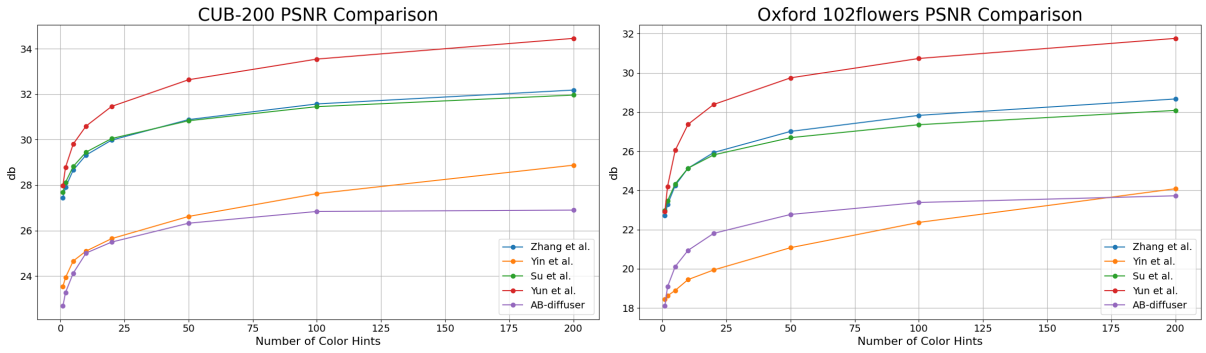

Figure 6.9: Our attained PSNR scores on CUB-200-2010 [46] and Oxford 102 Flowers [28] datasets against base-line models.

Chapter 7

Discussion

In this chapter we discuss our findings in Chapter 6 and their significance. We also address the current limitations of AB-diffusers as an tool for SEM image colorization, followed by a discussion on possible solutions to these limitations.

7.1 SEM Image Colorization

Based on the experiments in Section 6.1 we demonstrate that AB-diffuser is able to colorize SEM images with high precision and good visual clarity. Figures 6.2 and 6.5 shows that the user guidance system with hints allows for enough expressiveness that we are able to the isolate morphological features, to accentuate and highlight them with our intended color. This is a significant leap in interpretability when compared to the original publications, where these features are described solely in the text and figure captions.

The original greyscale SEM images in Figure 6.3 are typical examples on how authors often mark the relevant regions with gray symbols or arrows in order to point out, and refer to their findings in the text. We are able to further accentuate these marked pollen grains for clearer visualization, while keeping the assignment of color consistent between different viewing angles in Figure 6.3 A and B. This consistency lets reader quickly locate the relevant regions between figures when examining them. In Figure 6.4 we show that the model is able to identify and isolate minuscule features, even when a color hint is placed directly on top of it, demonstrating its ability to discern subtle features. It is important that critical, small details like these apertures are not lost in the colorization, as it would undermine the primary objective of aiding intepretability.

We also notice that the model is able to pick up on strong features and assign uniform colors to them automatically without providing any guidance. In Figure 6.2, the automatic colorization distinctly highlights the fiber in green, setting it apart from the surrounding gray particles. For the automatic colorization in Figure 6.3 A and B, we demonstrate the models ability to identify and assign consistent colors the main regions of the image, the pollen grains and the surrounding wall, without any user guidance. In this experiment, the majority of automatic outputs had the grains colored red. We suspect this may be due to the relatively high number of red-colored blood cells in our SEM image dataset, which may be an indication of the model overfiting on the data

In Figure 6.6, we demonstrate that AB-diffuser can closely replicate colorizations made with MountainsSEM, a frequently used tool for this task, although there are some noticeable discrepancies. These are even more prominent in our attempt at replicating the relatively complex SEM images in Figure 6.8, were color bleeding is apparent, and our model struggles with isolating the various features.

As we discuss in Section 1.1, the reason why one would highlight features with color in SEM, is that humans tend to group objects of similar colors together, this is a double edged sword when presenting colored SEM data with imperfections. Errors such as the ones displayed in Figure 7.1, can lead to misinterpretation of what authors intended to convey.

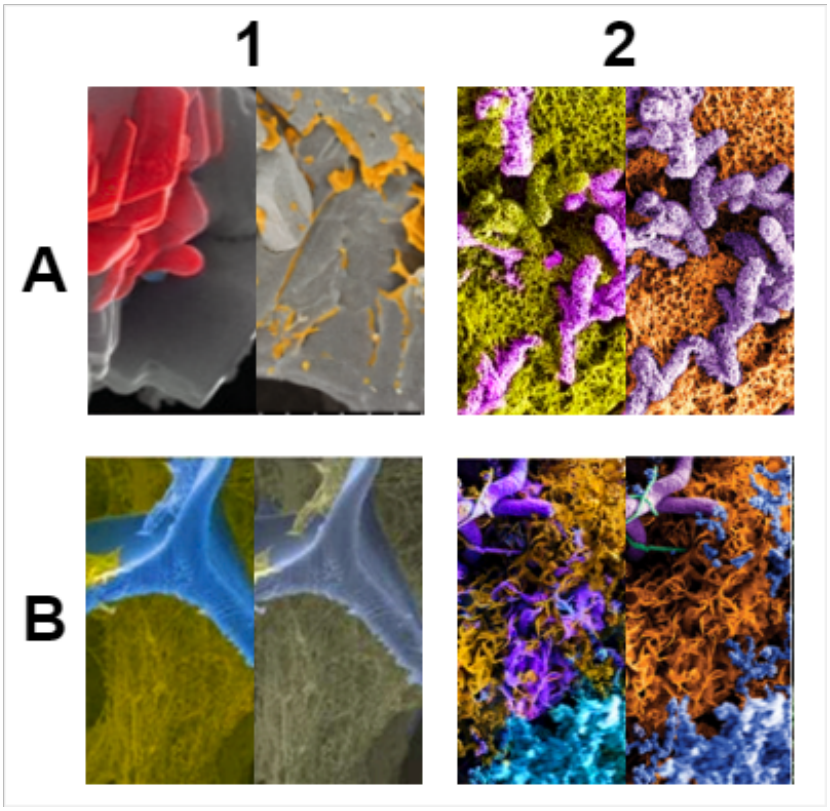

Figure 7.1: Examples of misleading colorizations.
A1: Red highlighting of hexagonal disks bleeds over to unrelated structure, and yellow highlighting of particles are present on unrelated cracks in the fiber.
A2: Yellow color assigned to iron sulfide, bleeding onto bacterium.
B1: Blue color assigned to cell structure covers part of the elastin material.
B2: Blue carbon is covered by yellow, and purple color bleeding over the iron sulfide.

7.2 Limitations

Impact of Inference Speed While errors such that as the ones we display in 7.1 are to be expected with parametric methods, our GUI does not allow for any further photo editing than applying color hints. This limits our options for error correction to repeated running of the model, until an acceptable result is achieved, and is a limitation that is further exacerbated by the slow inference speeds.

Table 6.6 shows that we spend on average ∼3 attempts on our semi-automatic colorization results, with each attempt taking 3:31 minutes. On average it takes 2:30 minutes for the model to output 9 images with a NVIDA A100 GPU, which accounts for ∼ 70% of the time for each attempt. Given the long waiting times for each batch to finish, we

are much more inclined to consider an colorization output as a "desirable result", even with such errors depicted in 7.1, that can be remedied by a few additional hints. Despite this, in a direct comparison, we demonstrate that AB-diffuser was able to colorize the SEM image showed in Figure 6.7 four times faster. Their software relies on manual editing of segmentation-lines to aid the colorization process, and they reported that 12 out of 15 minutes was spent doing so. With faster inference speed, significantly more time can be dedicated to adjust the colorizations, and as Figure 6.9 suggests, the performance increases as more hits are given to the model. The consistency of the colorizations between the two viewing angles in Figure 6.3 demonstrates that AB-diffuser allows for highly specific colorizations given enough well placed hints.

Natural Image Colorization As for natural image colorization, our model's performance lags behind the baseline when evaluated on PSNR. A higher PSNR value indicates that generated colorization's have less distortion or noise compared with their ground truth. Our lower scores suggest that our model introduce more noise or artifacts to the colorizations, compared to the baseline models. This outcome is anticipated, given that our training on natural images has not been nearly as extensive as the others.

For example, Instance-aware Image Colorization [39] has been trained for 9 epochs, or $\sim$ 180k steps with batch size 62 on ImageNet, while also being assisted by pretrained weights from the work of [52]. The best performer, iColorIt [50], has been trained for 2.5M steps with a batch size of 512.

Comparing a single evaluation metric is also not sufficient for a comprehensive study on its performance on natural image colorization. There are other relevant metrics that would be interesting to compare against a baseline such as FID [13], SSIM, and LPIPS [53] that provide a stronger indicator on perceptual quality than PSNR. We also lack qualitative comparisons between the models, because evaluation metrics that aim to quantify perceptual quality, can only indicate how good a model is performing. Time constraints have been the largest factor for this, both in terms of the amount of training we were able to do, and the amount of studies we were able to undertake.

Evaluating SEM colorization performance Ideally, to give have a more comprehensive assessment of AB-diffusers performance on SEM image colorization, conducting an expert interview would have been beneficial, not only for this study, but also during the development process. Lacking expertise in the field, we are only able to evaluate its performance based on what we have learned during this thesis, and feedback from an

expert would give us valuable insight on specific needs and standards for SEM image colorization, outside of our own research.

7.3 Future Improvements

In this section we discuss possible improvements to be made, and their possible implications.

Remedy For Inference Speed The slow inference speeds are typical for pixel-space diffusion models, and is one of the main reasons the literature is moving in the direction of latent-space diffusion models such as DALL-E2 [30] and Stable Diffusion [33]. There are, however, methods that accelerate inference speed of pixel-space diffusion models. Progressive distillation by Salimans et al. [38], is a method that has been shown to drastically reduce the amount of reverse diffusion steps needed to attain visual quality.

In this approach, the output of a fully-trained teacher model, is applied as the training target for a student model with half the diffusion steps. The process is iterative; after convergence, the student model, becomes the teacher for the next student with even fewer diffusion steps. This chain continues, progressively reducing the steps while retaining most of the generative ability of the original teacher model. Using this approach, Salimans et al. [38] were able to reduce the sampling steps in a diffusion model from 8,192 to 4, without significant loss in perceptual quality. In the work of Meng et al. [25], a variant of Stable Diffusion [33] was distilled from 2048 steps down to 8 steps, while retaining its original performance.

Implementing the training infrastructure to support progressive distillation would yield immediate positive result for our model. To strike a balance between quality and speed, we have settled on a moderate 1000 sampling steps. With progressive distillation we can train at a significantly higher amount of steps, and subsequently distill it down, while retaining better generative performance than AB-diffuser has in its current state. This would have an positive feedback loop on its usability as a tool for SEM colorization.

Faster processing times would enable users to quickly iterate and refine their color hints based on the model's outputs. As users see results more rapidly, they are more likely to fine-tune their inputs, leading to better and more precise colorizations. This iterative cycle, with immediate feedback influencing subsequent user input, would enhance the quality, while improving the user experience and confidence in the tool.

Exploiting the Utility Of "Gray" Hints As we briefly discussed in 6.1, we realized that we could attain highlighting-effect seen in Figures 6.5 and 6.3, by thoroughly saturating surrounding areas with "gray hints". This meant that we no longer had to rely on applying contrasting color hints to make certain features stand out. It also remedied problems with color bleeding, as we could simply apply more "gray" hints to problematic areas. As the current default value of the mask is 0, the model does not know when hints are sampled from gray areas during training, because 0 corresponds to "no color" in CIELAB space.

By applying an offset to the default value by a small amount, or a number outside the data-range, the model would be able to learn from hints sampled from these gray areas in the training data. This could reinforce the described behaviour, streamlining the colorization process for SEM images where only specific morphological features need to be accentuated.

Chapter 8

Conclusion

In this thesis we have built and trained AB-diffusion, a conditional diffusion model for SEM image colorization from the ground up. We have applied bleeding edge technology to a very specific and niche task. While we have uncovered many limitation in our approach, we have also achieved success. In this final chapter we will provide answers to our stated research questions, by discussing our findings, results, and problems we have encountered along the way.

Our stated aim with this thesis was to determine if we could leverage the emerging potential of diffusion models for image colorization to aid researchers in the process of colorizing SEM micrographs, posing the following questions:

- Can diffusion models be applied to the task of colorizing SEM images to enhance their visual interpretability?

In our results we present multiple colorized SEM images where features that otherwise only were referenced to or pointed out with text or symbols. We were able to isolate the the relevant areas with color, making them stand out from the rest of the images. The user guidance system allowed for enough expression that we were able to create consistent color representations across viewing angles of a specimen. The automatic colorization could for multiple instances, in as few as one attempt, isolate relevant features in color. While these findings indicate that we are able aid interpretability of the data with our approach, there are still certain limitations to address.

Color bleeding onto unintended structures was typical, and is a problem that we have argued to hurt interpretability of data. This is further exacerbated by one of the main

limitations of our diffusion model, inference speed. As it stands, our model takes on average 2:30 minutes to process each attempt. This had the consequence of us being willing to accept imperfections in the colorization, to avoid having to wait another 2:30 minutes, hoping that our adjustment had the intended effect.

- Can it produce results comparable to manual colorization and solutions dedicated to the task?

While we were able to accurately recreate many of the SEM images that were colorized by either MountainsSem or Photoshop, utilizing the user guidance system. There were some instances where the model struggled to accurately colorize the more complex scenes, with major color bleeding being apparent.

There was also a lack of direct comparison between the methods, both in terms of the colorization process itself, and time needed to colorize. It is difficult to say how much our approach can bring to the table compared to established methods, without such an study.

Bibliography

[1] Image Deduplicator (imagededup).
URL: `https://github.com/idealo/imagededup`.

[2] Tomer Amit, Tal Shaharbany, Eliya Nachmani, and Lior Wolf. SegDiff: Image segmentation with diffusion probabilistic models, 2022.

[3] Hernan Carrillo, Michaël Clément, Aurélie Bugeau, and Edgar Simo-Serra. Diffusart: Enhancing line art colorization with conditional diffusion models. In *Proc. CVPRW*, pages 3485–3489, June 2023.

[4] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proc. ICCV*, pages 415–423, 2015.

[5] Ami Citri and Robert C. Malenka. Synaptic plasticity: Multiple forms, functions, and mechanisms. *Neuropsychopharmacology*, 33(1):18–41, 2008. doi: 10.1038/sj.npp.1301559.

[6] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274.

[7] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Proc. NeurIPS*, 2021. doi: 10.48550/arXiv.2105.05233.

[8] Alicia Fernández-Colino, Frederic Wolf, Stephan Rütten, Thomas Schmitz-Rode, José Rodríguez-Cabello, Stefan Jockenhoevel, and Petra Mela. Small caliber compliant vascular grafts based on elastin-like recombinamers for in situ tissue engineering. *Frontiers in Bioengineering and Biotechnology*, 7:340, 11 2019. doi: 10.3389/fbioe.2019.00340.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[10] Israel Goytom, Qin Wang, Tianxiang Yu, Kunjie Dai, Kris Sankaran, Xinfei Zhou, and Dongdong Lin. Nanoscale microscopy images colorization using neural networks. *arXiv preprint arXiv:1912.07964*, 2019.

[11] Sergio Guadarrama, Ryan Dahl, David Bieber, Mohammad Norouzi, Jonathon Shlens, and Kevin Murphy. Pixcolor: Pixel recursive colorization, 2017.

[12] Heinrich Dr Herbst. Electron microscope with dark field illumination.
URL: `https://patents.google.com/patent/DE764812C/en`.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proc. NeurIPS*, pages 6840–6851, 2020.

[15] Shanshan Huang, Xin Jin, Qian Jiang, and Li Liu. Deep learning for image colorization: Current and future prospects. *Engineering Applications of Artificial Intelligence*, 114:Article 105006, 2022. doi: 10.1016/j.engappai.2022.105006.

[16] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics*, 35(4):1–11, 2016. doi: 10.1145/2897824.2925974.

[17] Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. *arXiv preprint*, arXiv:2212.11972, 2022. doi: 10.48550/arXiv.2212.11972.

[18] Gregory P. Krantz, Kilean Lucas, Erica L.-Wunderlich, Linh T. Hoang, Recep Avci, Gary Siuzdak, and Matthew W. Fields. Bulk phase resource ratio alters carbon steel corrosion rates and endogenously produced extracellular electron transfer mediators in a sulfate-reducing biofilm. *Biofouling*, 35(6):669–683, 2019. doi: 10.1080/08927014.2019.1646731.

[19] Manoj Kumar, Dirk Weissenborn, and Nal Kalchbrenner. Colorization transformer. *arXiv preprint arXiv:2102.04432*, 2021.

[20] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization, 2017.

[21] Yeongseop Lee and Seongjin Lee. Automatic colorization of anime style illustrations using a two-stage generator. *Applied Sciences*, 10(23):8699, 2020. doi: 10.3390/app10238699.

[22] Hanyuan Liu, Minshan Xie, Jinbo Xing, Chengze Li, and Tien-Tsin Wong. Video colorization with pre-trained text-to-image diffusion models, 2023.

[23] Hanyuan Liu, Jinbo Xing, Minshan Xie, Chengze Li, and Tien-Tsin Wong. Improved diffusion-based image colorization via piggybacked models, 2023.

[24] Yanbing Luo, Jiali Chen, Cheng Yang, and Yifan Huang. Analyzing ancient chinese handmade lajian paper exhibiting an orange-red color. *Heritage Science*, 7, 2019. doi: 10.1186/s40494-019-0306-6.

[25] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models, 2023.

[26] Erwann Millon. Color-diffusion. `https://github.com/ErwannMillon/Color-diffusion`, 2023. GitHub repository.

[27] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proc. ICML*, pages 8162–8171, 2021. doi: 10.48550/arXiv.2102.09672.

[28] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proc. icvgip*, pages 722–729, 2008. doi: 10.1109/ICVGIP.2008.47.

[29] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y. Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018.

[30] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[31] Michela Relucenti, Selenia Miglietta, Gabriele Bove, Orlando Donfrancesco, Ezio Battaglione, Pietro Familiari, Claudio Barbaranelli, Edoardo Covelli, Maurizio Barbara, and Giuseppe Familiari. Sem bse 3d image analysis of human incus bone affected by cholesteatoma ascribes to osteoclasts the bone erosion and vpsem dedx analysis reveals new bone formation. *Scanning*, 2020:1–9, 2020.
URL: `https://doi.org/10.1155/2020/9371516`.

[32] Michela Relucenti, Giuseppe Familiari, Orlando Donfrancesco, Maurizio Taurino, Xiaobo Li, Rui Chen, Marco Artini, Rosanna Papa, and Laura Selan. Microscopy methods for biofilm imaging: Focus on sem and vp-sem pros and cons. *Biology*, 10 (1), 2021. ISSN 2079-7737. doi: 10.3390/biology10010051.

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. CVPR*, pages 10684–10695, 2022. doi: 10.48550/arXiv.2112.10752.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241, 2015. doi: 10.1007/978-3-319-24574-4_28.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[36] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement, 2021. **URL:** `http://arxiv.org/abs/2104.07636`.

[37] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *Proc. SIGGRAPH*, Vancouver, BC, Canada, August 2022. doi: https://doi.org/10.1145/3528233.3530757.

[38] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022.

[39] Jheng-Wei Su, Hung-Kuo Chu, and Jia-Bin Huang. Instance-aware image colorization. In *Proc. CVPR*, June 2020.

[40] K.C. Sushma, R.B. Basavaraj, D.P. Aarti, M.B. Madhusudana Reddy, G. Nagaraju, M.S. Rudresha, H.M. Suresh Kumar, and K.N. Venkatachalaiah. Efficient red-emitting srzro3:eu3+ phosphor superstructures for display device applications. *Journal of Molecular Structure*, 1283:135192, 2023. ISSN 0022-2860. doi: https://doi.org/10.1016/j.molstruc.2023.135192.

[41] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proc. CVPR*, pages 2818–2826, 2016. doi: 10.48550/arXiv.1512.00567.

[42] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *Proc. CVPR*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587673.

[43] Chih-Hua Tsou, Ping Chin Cheng, Chiung-Maan Tseng, Hsiao-Jung Yen, Yu lan Fu, Tien-Rong You, and David B. Walden. Anther development of maize (zea mays) and longstamen rice (oryzalongistaminata) revealed by cryo-sem, with foci on locular dehydration and pollen arrangement. *Plant Reproduction*, 28:47 – 60, 2015. doi: 10.1007/s00497-015-0257-3.

[44] Phil Wang. Denoising-diffusion-pytorch, 2020.
URL: `https://github.com/lucidrains/denoising-diffusion-pytorch`.

[45] Weights & Biases. Weights & biases – developer tools for ml, 2023.
URL: `https://wandb.ai/site`.

[46] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200, 2010.

[47] WFDownloader. Wfdownloader app - free bulk image downloader and multi-purpose bulk downloader, 2023.
URL: `https://www.wfdownloader.xyz/`. Accessed: 2023-08-03.

[48] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, pages 3485–3492, 2010. doi: 10.1109/CVPR.2010.5539970.

[49] Hui Yin, Yuanhao Gong, and Guoping Qiu. Side window filtering. In *Proc. CVPR*, June 2019.

[50] Jooyeol Yun, Sanghyeon Lee, Minho Park, and Jaegul Choo. icolorit: Towards propagating local hint to the right region in interactive colorization by leveraging vision transformer, 2022.

[51] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *Proc. ECCV*, pages 649–666, 2016. doi: 10.1007/978-3-319-46487-9_40.

[52] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. Real-time user-guided image colorization with learned

deep priors. *ACM Transactions on Graphics*, 36(4):1–11, 2017. doi: 10.1145/3072959.3073703.

[53] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.