Microarray Image Denoising Leveraging Autoencoders and Attention-Based Architectures with Synthetic Training Data

by

Chris Czarnecki

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2024

© Chris Czarnecki 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Author's contributions to the research project, which this thesis applies to were as follows:

- Creation of the UArr package in Python, enabling plug-and-play microarray denoising workflows.
- Synthetic data generation conceptualization and implementation.
- Proposal of the MaxPool Readout Algorithm.
- Selection of denoising methods, their implementation and/or integration into the UArr package.
- Proposal and implementation of the Standard Assessment of Denoising in Gene-chip Evaluation (SADGE) metric.
- Conceptualization and implementation of the noising protocol, including Blob Noise.
- Analysis of the error associated with the MAGIC Tool readout and comparison against the MaxPool Readout Algorithm.
- Implementation of training and testing loops for all models used in the study.
- Generation of peak signal-to-noise ratio (PSNR), f-PSNR and SADGE results for all models used in the study.
- Introduction of the Dot Expression Loss (DEL) loss term.
- Introduction of the Elementwise Attention-like Transform for Microarray Enhancement (EATME) module.

Contributions of others involved:

- Krish Shah
  - Adaptation of the MAGIC Tool to a Java library usable from Python via JPype for the UArr package, maintenance activities for the UArr-MAGIC repository.
  - Implementation of the MaxPool Readout Algorithm in Python.

- Implementation of step-gradient image generation for readout method error evaluation.
- Addition of optional residual connections in the implementation of the Denoising Autoencoder in the UArr package.
- Assistance with trials of microarray readout software.

- Siddharth Viswanath
  - Fitting RNA-Seq experimental data to an appropriate distribution using the Fitter Package [1].

Abstract

Microarray technology has for many years remained a golden standard in transcriptomics. However, preparation of physical slides in wet labs involves procedures which tend to introduce occasional dirt and noise into the slide. Having to repeat experiments due to environmental noise present in the scanned images leads to increased reagent and labor costs. Motivated by the high costs of repeated wet lab procedures we explore denoising methods in the narrow subfield of microarray image analysis. We propose SADGE, a domain-relevant metric to quantify the denoising power of methods considered. We introduce a synthetic data generation protocol which permits the creation of very large microarray image datasets programmatically and provides noise-free ground truth useful for objective quantification of denoising. We also train several deep learning architectures for the denoising task, with several of them beating the current state-of-the-art method on both PSNR and SADGE metrics. We propose a new training modality leveraging EATME module to condition the image reconstruction on ground-truth expression values and we introduce an additional loss term (DEL) which further enhances the denoising capabilities of the model while ensuring minimal information loss. Collectively, innovations outlined in our work constitute a significant contribution to the field of microarray image denoising, influencing the cost-effectiveness of microarray experiments and thus impacting a wide range of clinical and biotechnological applications.

Acknowledgements

I would like to thank all the individuals who made working on this thesis possible, in particular my inspiring and generous supervisor Prof. Alexander Wong.

I would also like to give special thanks to Prof. Yuhao Chen for supporting me with guidance and assistance in working on research projects at VIP Lab.

I also want to thank Krish Shah, who significantly contributed to the practical part of the Microarray Denoising project under my guidance, helping with numerous implementation tasks.

Finally, a big thank you to all my friends at VIP Lab: for good advice, for a helping hand whenever I needed it and for creating a welcoming and intellectually stimulating environment, in which I grew as a researcher and as a person.

Dedication

I would like to dedicate this thesis to my family members, who supported me on this journey mentally, financially and physically. In particular I would like to dedicate this work to uncle Krzysztof and aunt Katja. Thank you for bearing with me throughout the entirety of my Masters degree and for creating a happy and nurturing household in which I could truly focus on learning and where I felt supported and safe every step of the way.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract v

Acknowledgements vi

Dedication vii

List of Figures xi

List of Tables xii

List of Abbreviations xiii

List of Symbols xiv

1 Introduction 1

1.1 Microarrays 1

1.1.1 What are microarrays? 1

1.1.2 Microarray images 2

1.1.3 RNA-Seq vs. microarrays 3

1.2 Noise . . . 3
1.2.1 Environmental noise in microarray images . . . 3
1.3 Problem statement . . . 5
1.4 Image denoising . . . 5
1.4.1 Classical denoising methods . . . 5
1.4.2 Denoising methods leveraging deep learning . . . 7

2 Background **14**
2.1 Prior works on microarray denoising leveraging classical methods . . . 14
2.2 Prior works on microarray denoising leveraging deep learning . . . 15
2.3 Prior works on measuring expression values from microarray images . . . 17
2.3.1 Software which did not pass the criteria . . . 17
2.3.2 MAGIC Tool . . . 17

3 Methods **19**
3.1 Data . . . 20
3.1.1 Baseline data . . . 20
3.1.2 Synthetic data generation . . . 20
3.1.3 Image noise injection . . . 22
3.1.4 Image representation . . . 23
3.2 Metrics . . . 24
3.2.1 PSNR and f-PSNR . . . 24
3.2.2 SADGE . . . 25
3.3 Evaluation on real microarray data . . . 31
3.4 Denoising methods . . . 31
3.4.1 Wavelet denoising . . . 31
3.4.2 Baseline autoencoder . . . 32
3.4.3 Autoencoder trained on synthetic data . . . 32

3.4.4 Restormer trained on synthetic data . . . 32
3.4.5 Autoencoder with residual connections . . . 33
3.4.6 Autoencoder trained on normally-distributed data . . . 33
3.4.7 Autoencoder conditioned on ground-truth expression values . . . 33

4 Results **36**
4.1 Evaluating MAGIC Tool on synthetic data . . . 37
4.2 SADGE calculated on input images . . . 38
4.3 Baseline . . . 38
4.3.1 Wavelet denoising . . . 38
4.3.2 Baseline autoencoder . . . 40
4.4 Autoencoder trained on synthetic data . . . 40
4.5 Restormer trained on synthetic data . . . 40
4.6 Autoencoder with residual connections . . . 43
4.7 Autoencoder trained on normally-distributed data . . . 44
4.8 Autoencoder with cross-attention . . . 44
4.9 Comparison of results . . . 45

5 Conclusion **51**
5.1 Recommendations and considerations . . . 51
5.2 Future work . . . 52

References **53**

APPENDICES **58**

A Implementation details **59**
A.1 Model training hyperparameters . . . 59

Glossary **60**

Nomenclature **61**

List of Figures

1.1 Microarray glass slide 2
1.2 A genome of a hypothetical virus 4
1.3 An example of a microarray with environmental noise 5
1.4 Transformer architecture 11
1.5 Restormer architecture 12

2.1 Denoising autoencoder architecture 16

3.1 Emulation of environmental noise by *blob noise* 23
3.2 PSNR limits 25
3.3 Reference image for relative dot intensity measurement 27
3.4 MAE vs. Log-MAE 28
3.5 Step-gradient image 29
3.6 Autoencoder with EATME 35

4.1 MAE plots for different microarray readout methods 39
4.2 Performance of wavelet denoising on microarrays 41
4.3 Visual comparison of results for different denoising methods for the synthetic test set 48
4.4 Restormer model inaccuracies 49
4.5 Visual comparison of results for different denoising methods for the DeRisi test set 50

List of Tables

3.1 Hyperparameters used to instantiate the Restormer model. 32

4.1 Comparison of gene expression readout methods 37
4.2 SADGE values for input images 38
4.3 Quantitative results for wavelet denoising 40
4.4 Results for the baseline Autoencoder 42
4.5 Results for the autoencoder trained on synthetic data. 42
4.6 Quantitative results for the Restormer model trained on synthetic data. . . 43
4.7 Quantitative results for the autoencoder model with residual connections trained on synthetic data. 43
4.8 Quantitative results for the autoencoder model trained on a normal distribution and tested on the chi-squared-following data 44
4.9 Quantitative results for the autoencoder with EATME and DEL. 45
4.10 SADGE scores on the synthetic dataset test split 45
4.11 PSNR scores on the synthetic dataset test split 46
4.12 f-PSNR scores on the DeRisi dataset test split 46

List of Abbreviations

DEL Dot Expression Loss. i, iii, v, 34, 44, 45, 48–51

DNA Deoxyribonucleic acid. i, 1

EATME Elementwise Attention-like Transform for Microarray Enhancement. i, iii, v, 33, 35, 44, 45, 48–51

MAE mean absolute error. i, 26, 33, 37

MSE mean squared error. i, 24, 54

PSNR peak signal-to-noise ratio. i, iii, v, 24, 44–46, 51

RNA Ribonucleic acid. i, 1

SADGE Standard Assessment of Denoising in Gene-chip Evaluation. i, iii, v, 26, 36, 44–47, 51

List of Symbols

G Ground-truth image, an image with guaranteed zero noise. i, 24, 48

I Input image, potentially containing noise. i, 24

I_η An image with noise intentionally added. i, 24, 25, 36, 48, 50

O Denoised image, output from a denoising model. i, 24, 25

R An image used as a relative measurement reference of microarray dot intensity. i

ι Measured absolute pixel intensity associated with a dot corresponding to a microarray probe in the scanned microarray image. i, 26

$\mathbf{v}$ A lowercase bold letter denotes vectors. i

e Measured expression value associated with a dot in a microarray image. Unlike the absolute pixel intensity, the expression value is always relative to some reference value. i, 26

Chapter 1

Introduction

1.1 Microarrays

1.1.1 What are microarrays?

Microarrays (Figure 1.1), often called *gene chips*, are microfluidic systems for the general purpose of quantitative biological interaction analysis involving nucleic acids such as Deoxyribonucleic acid (DNA) or Ribonucleic acid (RNA). DNA microarrays are a specific subtype of microarrays, which consist of a 2D grid of DNA probes affixed to a glass slide. Each DNA sequence is unique and complementary to some target sequence, which the microarray is supposed to detect. Studies have applied microarrays in human ancestral allele analysis [3], developed microarray methods for large-scale screening of genetically-modified crop [4].

They are used primarily in large-scale biological studies. One particular use of such devices is in the field of transcriptomics, where the set of currently expressed genes is compared against the set of expressed genes in some control sample. The impact of transcriptomics is widespread, from profiling heterogeneity of cells within tumors [5], to interspecies studies [6]. An intuitive example could be the usage of microarrays in melanoma biomarker determination, where the gene expression patterns of healthy cells are compared to cancerous cells [7]. The microarray method has therefore a profound meaning in modern cancer diagnostics.

The name *transcriptomics* stems from the process of transcription, i.e. the process where RNA Polymerase reads the DNA sequence and synthesizes an RNA molecule. In

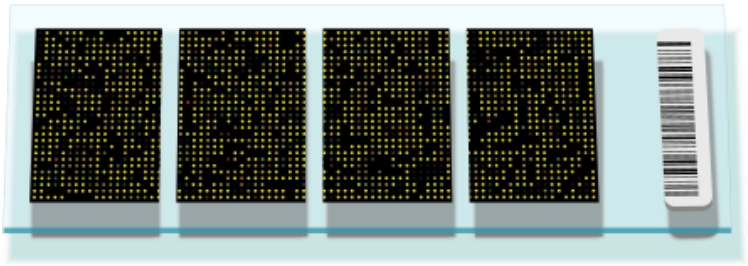


Figure 1.1: Microarray glass slide (DataBase Center for Life Science (DBCLS), CC BY 4.0, via Wikimedia Commons) [2].

microarrays for transcriptomic applications the RNA fraction from a cell or tissue is extracted and converted back to DNA in the process known as reverse transcription [8]. DNA tends to be more stable, which is why this step is usually essential in the preparation of microarray slides [9]. As a result of reverse transcription *complementary DNA* (cDNA) is obtained. Before a plain glass slide can become a microarray, DNA probes must be affixed to its surface. The fixation is performed in a regular grid-like pattern and multiple copies of the same probe are packed together tightly around one (x, y) position in the 2D grid forming an ellipse or circle-like pattern commonly called a dot or *spot*. Then, either the probes or the analyte must attach a fluorophore, which will enable imaging. In the process of reverse transcription, fixation of the fluorescent dyes is highly efficient [10], which is yet another reason that motivates the addition of the reverse transcription step to the procedure.

In the process of hybridization cDNA molecules bind to the DNA probes on the surface of the microarray. This activates fluorophore markers which radiate out light of a specific wavelength after having absorbed UV-spectrum wavelengths. The exact excitation and emission wavelengths depend on the fluorophore used [11]. A scanner is used to read the fluorescent signal, typically leveraging a laser for fluorophore excitation, a system of lenses and a photodiode for signal detection [11].

1.1.2 Microarray images

Microarray images typically consist of two channels to account for the test and control sample [12]. Each of the probe aggregates appears as a dot on a black background and the

pixel intensity value within the region of the dot can be interpreted as a quantitative signal. The brighter the dot, the more sequences have bound to the probes within the region. In the context of transcriptomics this can be interpreted as a stronger gene expression, the brighter the dot.

To illustrate this better, one could imagine a hypothetical virus, of which genome consists of merely 4 genes: $\gamma_1, \gamma_2, \gamma_3, \gamma_4$. Such viruses do indeed exist in nature – for example one of the smallest known viruses, the Circovirus, has only 2 genes in its entire genome [13]. Post-infection, the host's cells will contain the RNA of the hypothetical virus and depending on which of the genes are currently active in what proportion, the relative amount of said RNA would vary. One might ask a question: which of the genes contributes to which stage of the disease progression and perform multiple microarray experiments at specific timepoints using a 4-probe microarray (Figure 1.2). It is therefore one thing to know the sequence of a gene, and another to know when and where it is active. Both modalities increase our understanding about the mechanisms of infection, disease progression and might influence treatment outcomes.

1.1.3 RNA-Seq vs. microarrays

In many laboratories microarrays have largely been superseded by a more direct approach of sequencing RNA molecules, RNA-Seq. RNA-Seq yields quantitative information akin to the microarrays. However, microarrays are not an entirely forgotten technology. First, RNA-Seq data analysis might be more challenging due to some RNAs assuming secondary structures. This is less problematic with cDNA microarrays [14]. Microarrays also remain more cost-effective for certain large-scale analyses [15] with companies like Illumina offering a wide range of microarray solutions for e.g. genotyping or methylation analysis [16].

1.2 Noise

1.2.1 Environmental noise in microarray images

Microarray images are a particular case of images, where the intensity of pixels in given positions in the image must be maintained in order to obtain a trustworthy experimental result. As described before, the slide preparation process, calibration of the scanner, the cleanliness of the environment, where the extract is added to the probes prior to hybridization, all these factors influence trustworthiness of a gene expression value readout from a scanned microarray. We present an example of environmental noise in Figure 1.3.

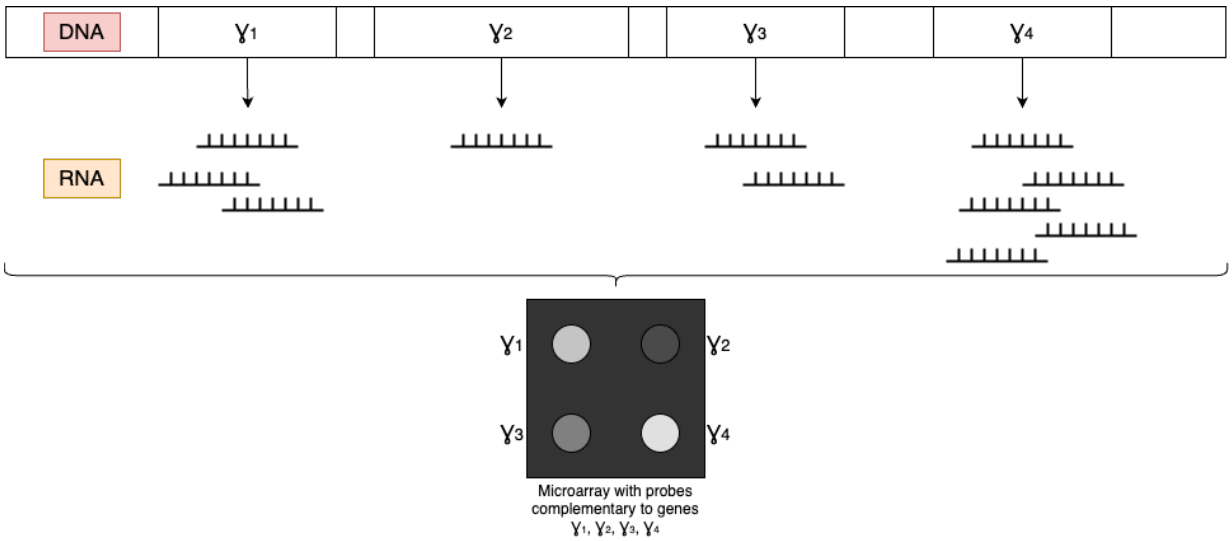


Figure 1.2: A genome of a hypothetical virus. Transcription from DNA to RNA is the first step in the expression of genetic information. Intercepting RNA, converting it to cDNA by means of reverse transcription and hybridization of the cDNA product to the microarray performed over the course of infection progression can yield important information about the illness progression. The diagram illustrates the relationship between the intensity of each dot on the scanned microarray image and the number of RNA transcripts from hypothetical genes $\gamma_1, \gamma_2, \gamma_3, \gamma_4$. The more RNA transcript, the more copies will hybridize to a probe complementary to a given gene, leading to a brighter dot post-scanning.

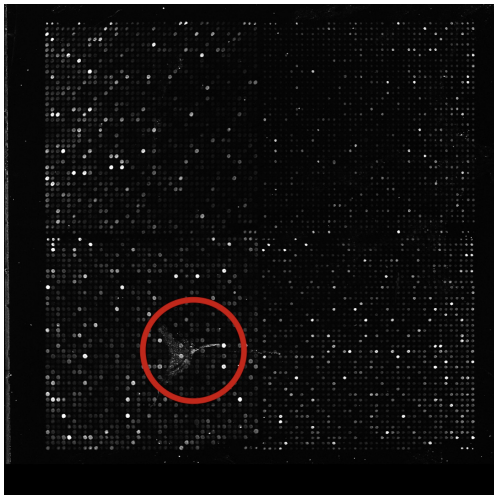


Figure 1.3: An example of a microarray image with a prominent speck of environmental noise close to the lower left quadrant. Courtesy of J. DeRisi [17].

1.3 Problem statement

Because many impurities can be introduced onto the microarray slides both as a result of manufacturing imperfections, as well as environmental debris, denoising of microarray images has been a topic of many studies [18, 19, 20, 21]. The noise impacts the accuracy of relative comparison between the test and control samples. If the microarray image is of poor quality with visible artefacts after scanning, the experiment needs to be repeated which involves additional costs of chips and reactants, which could otherwise be avoided, should a reliable noise-removal method be proposed.

1.4 Image denoising

1.4.1 Classical denoising methods

Classical image denoising methods are inspired by signal processing theory, drawing on concepts such as Fourier transforms [22], wavelets [18], and statistical analysis to filter out noise while preserving important image details. These techniques often involve decomposing an image into different frequency components, identifying and isolating the noise, and

then reconstructing the image to achieve a cleaner and more accurate representation. By leveraging mathematical models and algorithms, classical methods aim to improve image quality in various applications, from medical imaging to photography.

While robust and effective against noise in general real-world images, they are not necessarily optimal for images encoding particular experimental semantics, such is the case with microarray images. Nevertheless some classical methods, in particular wavelet denoising, have been in the past successfully applied to microarray images.

What follows is background information on wavelet denoising and deep-learning-based denoising methods. Their concrete applications to microarray images will be covered in Chapter 2.

Wavelet denoising

One widely used image denoising method which is often applied to microarray images is wavelet denoising, leveraging wavelet transform to decompose the image into a set of basis functions called wavelets. Traditionally this method has been in the past applied to time/frequency domain signals. In the case of image denoising, the time domain is replaced by the spatial domain (position of pixels in the 2D image grid).

The discrete wavelet transform (DWT) of a signal $f(t)$ can be expressed as:

$$W_{j,k} = \langle f(t), \psi_{j,k}(t) \rangle$$

where $\psi_{j,k}(t)$ are the wavelet basis functions at scale j and position k, and $\langle \cdot, \cdot \rangle$ denotes the inner product. The wavelet coefficients $W_{j,k}$ capture the signal's details at various scales and positions.

Wavelet denoising involves three main steps: decomposition, thresholding, and reconstruction.

First, the noisy signal $f(t)$ is decomposed into wavelet coefficients using the DWT:

$$\{W_{j,k}\} = \mathrm{DWT}(f(t))$$

The wavelet coefficients are then modified by applying a thresholding function to reduce the influence of noise. Two common thresholding techniques are hard thresholding and soft thresholding.

- Hard Thresholding:

$$\hat{W}_{j,k} = \begin{cases} W_{j,k} & \text{if } |W_{j,k}| > \lambda \\ 0 & \text{if } |W_{j,k}| \leq \lambda \end{cases}$$

- Soft Thresholding:

$$\hat{W}_{j,k} = \begin{cases} \text{sign}(W_{j,k})(|W_{j,k}| - \lambda) & \text{if } |W_{j,k}| > \lambda \\ 0 & \text{if } |W_{j,k}| \leq \lambda \end{cases}$$

Here, λ is the threshold parameter that determines which coefficients are considered noise and which are signal. In the case of Soft Thresholding, the subtraction of λ from $|W_{j,k}|$ yields smoother transitions with fewer discontinuities in the denoised signals. Soft Thresholding typically removes noise more effectively but can occasionally lead to a reduction of the signal, sometimes shrinking signal-related coefficients.

Ultimately, the denoised signal $\hat{f}(t)$ is reconstructed from the modified wavelet coefficients using the inverse discrete wavelet transform (IDWT):

$$\hat{f}(t) = \text{IDWT}(\{\hat{W}_{j,k}\})$$

The choice of threshold λ is crucial for effective denoising. A common method for selecting λ is the universal threshold, given by:

$$\lambda = \sigma\sqrt{2\log N}$$

where σ is the estimated noise standard deviation, and N is the number of data points in the signal.

In the case of images the wavelet transform is applied in two dimensions, along the rows and columns of the image.

Notably, noise is assumed to be sampled from a Gaussian distribution.

1.4.2 Denoising methods leveraging deep learning

Denoising autoencoders

A denoising autoencoder (DAE) is a type of neural network designed to remove noise from data. It learns to reconstruct a clean version of the input from its noisy counterpart. In a U-Net-style convolutional autoencoder, the architecture typically consists of an encoder, and a decoder.

Model Architecture The model architecture generally consists of two branches connected sequentially:

1. encoder – transforms the input $\mathbf{x}$ into a lower-dimensional representation using convolutional layers and max-pooling operations.

$$\mathbf{h} = f_{\text{encoder}}(\mathbf{x})$$

2. decoder – the decoder reconstructs the clean signal or image from the compressed representation using up-sampling operations and convolutional layers.

$$\hat{\mathbf{x}} = f_{\text{decoder}}(\mathbf{h})$$

The encoder and decoder are typically laid out in a U-Net architecture by including skip connections (residual connections) between corresponding layers in the encoder and decoder. These connections allow the model to retain high-resolution features that might be lost during down-sampling.

Let $\mathbf{x}_{i,j}$ represent the feature map at the j-th layer of the encoder, and $\mathbf{y}_{i,j}$ represent the feature map at the j-th layer of the decoder. The skip connections are represented as:

$$\mathbf{y}_{i,j} = f_{\text{decoder},j}(\mathbf{h}_{i,j}, \mathbf{x}_{i,j})$$

where $\mathbf{h}_{i,j}$ is the output of the previous layer in the decoder and $f_{\text{decoder},j}$ includes both the up-sampling and convolution operations.

The goal is to train the model f to minimize the reconstruction error between the clean input $\mathbf{x}$ and the denoised output $\hat{\mathbf{x}}$. The supervised learning task involves minimizing the loss function $\mathcal{L}$, typically the mean squared error (MSE), defined as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$$

where $\mathbf{x}_i$ is the i-th clean input, $\hat{\mathbf{x}}_i = f(\mathbf{x}_i + \mathbf{n}_i)$ is the reconstructed output from the noisy input $\mathbf{x}_i + \mathbf{n}_i$, and $\mathbf{n}_i$ represents the noise added to the input. Here, N is the number of samples in the training dataset.

The denoising autoencoder is trained in a supervised fashion by using pairs of clean and noisy datapoints. During training, the model f learns to map noisy inputs to their clean counterparts by minimizing the reconstruction error. The U-Net-style architecture, with its skip connections, helps preserve spatial information and improve the quality of the denoised output. This approach is particularly effective for image denoising tasks, where retaining fine details is crucial.

Transformer-based denoising architectures

In recent years Transformer-based architectures have gained a lot of popularity. Unlike recurrent neural networks [23], they can be easily parallelized by splitting attention heads across a number of computational units. Unlike Long short-term memory networks [24], they tend to capture long-range dependencies in the input space more effectively. They also have been shown empirically to scale better: more training data and deeper models with a larger number of parameters typically yield superior performance, following a power law [25]. For these reasons we have investigated Transformer-based architectures for the denoising problem.

Transformer architecture The Transformer architecture, introduced by Vaswani *et al.* [26] revolutionized the field of natural language processing by relying entirely on self-attention mechanisms instead of recurrent or convolutional neural networks. The architecture has subsequently been adapted to work with images, leveraging patch embeddings.

The architecture consists of an encoder-decoder structure, where both the encoder and decoder are composed of multiple layers of self-attention and feed-forward neural networks. The encoder processes the input sequence to create a latent representation, while the decoder generates the output sequence based on this representation and previously generated tokens [26].

Attention Attention mechanisms allow the Transformer to weigh the importance of different tokens in the input sequence when encoding a particular token, enabling the model to capture relationships and dependencies across the entire sequence. Each token's representation is computed as a weighted sum of the other tokens' representations, with the weights determined by learned attention scores. This process enables the model to focus on relevant parts of the sequence, making it capable of handling long-range dependencies and providing more context-aware representations. Two fundamental types of the attention mechanism are additive attention and multiplicative attention. They differ in the alignment score function definition. For additive attention the alignment score function is:

$$Attn(\mathbf{e}_i, \mathbf{d}_j) = \mathbf{w}^T \tanh(\mathbf{W}[\mathbf{e}_i; \mathbf{d}_j]) \tag{1.1}$$

where:

- $\mathbf{e}_i$ – hidden state of the encoder for the i^{th} token in the input sequence

- $\mathbf{d}_j$ – hidden state of the decoder for the j^{th} token in the input sequence
- $\mathbf{w}$ – scaling factor (trained alongside the weight matrix)
- W – weight matrix
- $[\mathbf{e}_i; \mathbf{d}_j]$ – concatenation of the hidden state of the encoder and the decoder

Multiplicative attention can be characterized by the following alignment score function:

$$Attn(\mathbf{e}_i, \mathbf{d}_j) = \mathbf{e}_i^T \mathbf{W} \mathbf{d}_j \tag{1.2}$$

Scaled Dot-Product Attention Scaled dot-product attention is a specific sub-flavor of the multiplicative attention mechanism. It has been introduced alongside and used in Transformers [26]. It operates by first computing the dot product between the query and key vectors to obtain raw attention scores. This is equivalent to the encoder hidden state in a conventional multiplicative attention. These scores are then scaled by the square root of the dimensionality of the key vectors, which helps mitigate the issue of large dot product values causing gradient instability while training. The scaled scores are passed through a softmax function to obtain the attention weights, which are subsequently used to compute a weighted sum of the value vectors. This process results in the final attention output, which effectively captures the relevant information from the input sequence for each token. The formula for scaled dot-product attention is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{1.3}$$

Here, Q, K, and V represent the query, key, and value matrices, respectively, and d_k is the dimensionality of the key vectors. The matrices are stacked query, key and value tensors respectively. The usage of matrices enables efficient parallel computation of the attention scores. The softmax function ensures that the attention scores sum to one, providing a probability distribution over the input tokens.

Restormer A more recent denoising architecture based largely on the Transformer model is the Restormer architecture proposed by Zamir *et al.* [27].

The authors recognize that convolutional neural networks are limited by their receptive field. Transformers (Figure 1.4), with their ability to model long-range dependencies, were

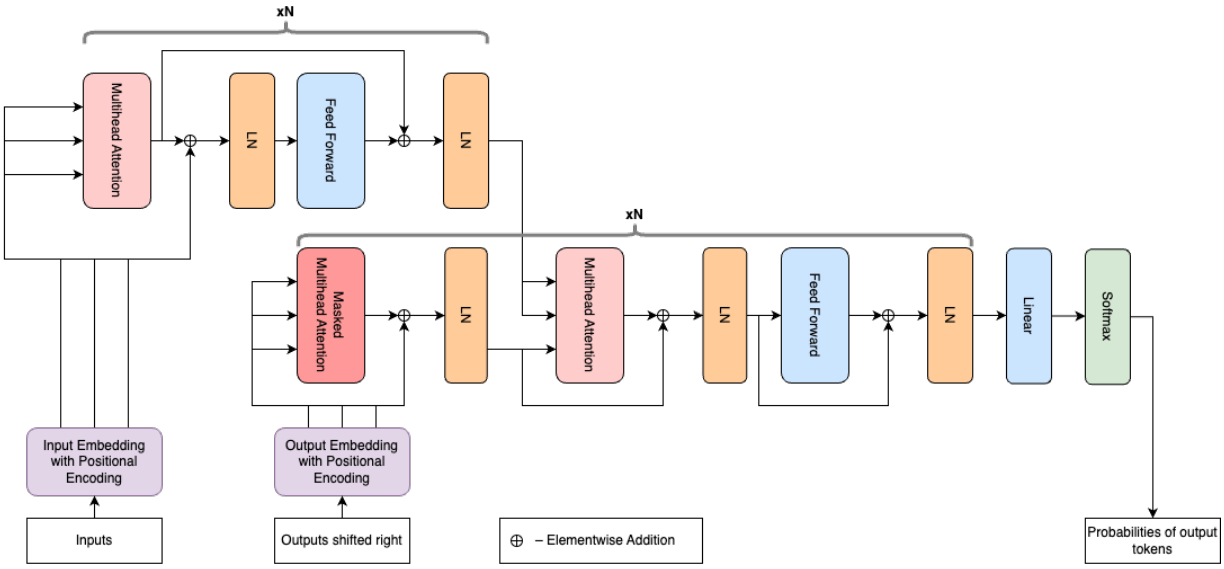


Figure 1.4: The original Transformer architecture designed for sequence-to-sequence tasks. $\times N$ indicates the number of blocks, a hyperparameter controlling the depth of the model.

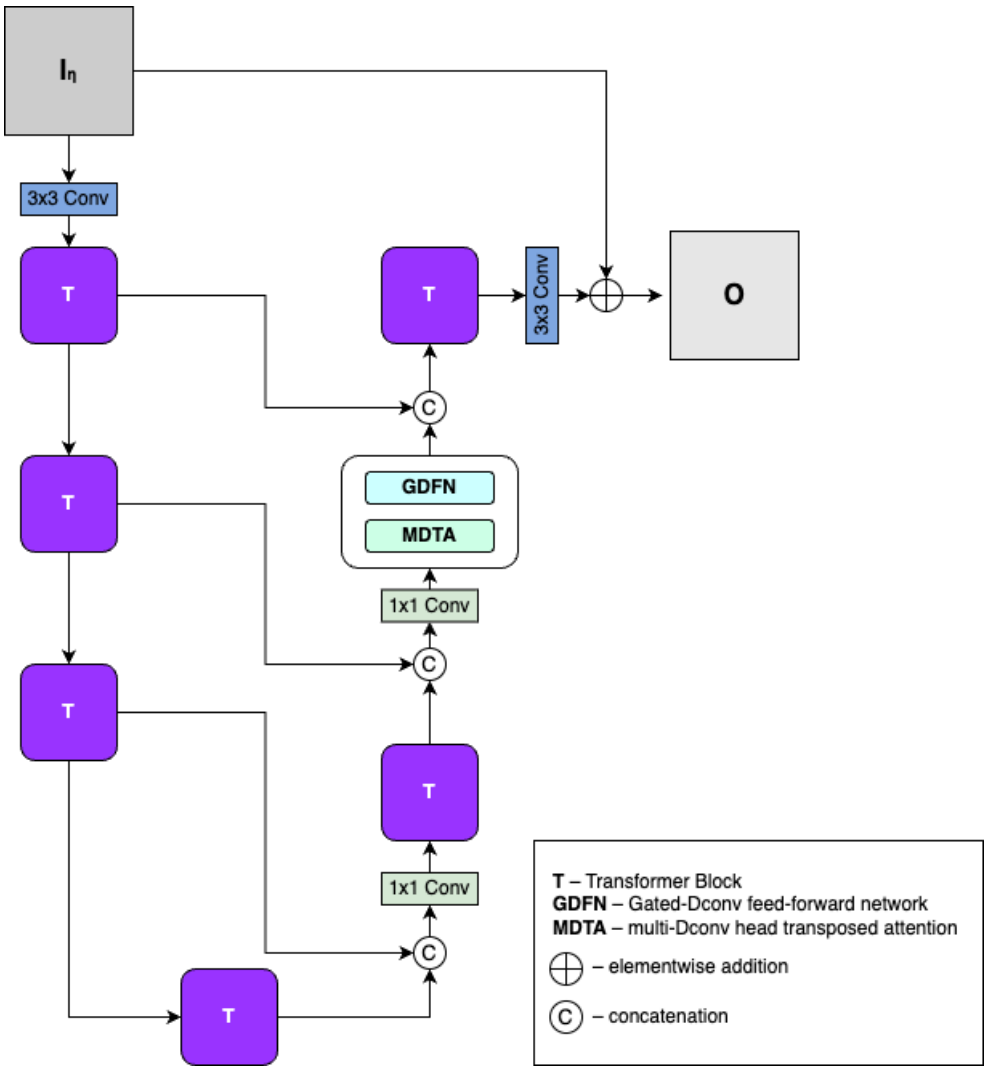


Figure 1.5: Restormer [27] architecture.

previously explored for this task, but their high computational complexity has traditionally posed a challenge for high-resolution images [27]. Two novel elements are introduced into the architecture (Figure 1.5) which led to its state-of-the-art performance:

- **Multi-Dconv Head Transposed Attention (MDTA)**: a module that efficiently aggregates local and non-local pixel interactions, making it suitable for high-resolution images. This module reduces computational overhead by computing self-attention across channels rather than spatial dimensions.
- **Gated-Dconv Feed-Forward Network (GDFN)**: a module that performs controlled feature transformation, enhancing informative features and suppressing less useful ones. It utilizes depth-wise convolutions to emphasize local context before feature transformation.

Since Restormer is one of the top-performing denoising methods for general image denoising problems and it is computationally feasible to train within the scope of our resources, we explore it as a more modern alternative microarray denoising method.

Chapter 2

Background

This chapter will briefly introduce related works on microarray denoising that extend beyond the introductory material presented in Chapter 1.

2.1 Prior works on microarray denoising leveraging classical methods

Historically, microarray denoising has been performed using wavelet transforms [18, 19] and Gaussian scale mixture models of complex wavelets [20]. These methods all constitute classical denoising methods, where no trainable model is used.

Wavelet transform-based denoising leverages the multi-resolution analysis capability of wavelets to decompose microarray images into different frequency components. By applying thresholding techniques to the wavelet coefficients, noise can be effectively reduced while preserving significant features in the data. This approach is particularly advantageous for microarray images, which often contain localized noise and artifacts that can obscure the biological signals of interest.

The Gaussian scale mixture (GSM) model [20] further refines the wavelet-based approach by modeling the statistical properties of wavelet coefficients. In this method, wavelet coefficients are assumed to follow a Gaussian distribution at each scale, allowing for more sophisticated thresholding and noise reduction. The GSM model can capture the complex statistical dependencies between wavelet coefficients, resulting in improved denoising performance compared to simple thresholding methods.

These classical denoising methods have been widely adopted due to their robustness and the mathematical rigor underlying wavelet theory. They do not require any training data or learned parameters, making them relatively easy to implement and apply to various types of microarray images. However, their performance is inherently limited by their reliance on predefined wavelet bases and statistical models, which may not fully capture the intricate noise patterns present in microarray data, such as semi-translucent or opaque debris.

Despite these limitations, wavelet transform-based methods and GSM models remain foundational techniques in the field of microarray image denoising. They provide a benchmark against which newer, more advanced methods can be compared, and they continue to be used in applications where simplicity and computational efficiency are paramount.

2.2 Prior works on microarray denoising leveraging deep learning

The current state-of-the-art for microarray denoising is the work of Mohandas *et al.* [21], where a denoising autoencoder architecture is used. This is the first application of a deep learning method to the narrow subfield of microarray denoising.

Mohandas *et al.* propose a denoising autoencoder method for the purpose of microarray image denoising [21]. The autoencoder has been trained on two combined datasets of real microarray images, of which only the DeRisi dataset is still available online [28]. To the best of our knowledge the work of Mohandas *et al.* is until now not only the first but also the only application of a deep learning method to microarray image denoising.

The architecture used by Mohandas *et al.* which serves as a baseline for our work is a U-Net-style architecture, where the downsampling branch interleaves Max Pooling layers in between convolutional layers. The architecture is depicted in Figure 2.1.

An attractive research direction was to apply more modern architectures to the problem and to address some of the shortcomings of the prior work. First, the authors used 560 images for training, which is a relatively small training dataset. Second, all input samples consisted of real microarray images with extra noise added. As shown in Chapter 1, real images already contain a considerable amount of noise due to environmental factors. Third, the reliability of PSNR as a metric for denoising success in the context of this problem is dubious, as will soon become apparent. Fourth, the autoencoder architecture proposed by Mohandas *et al.* does not leverage skip-connections between the Encoder and Decoder layers.

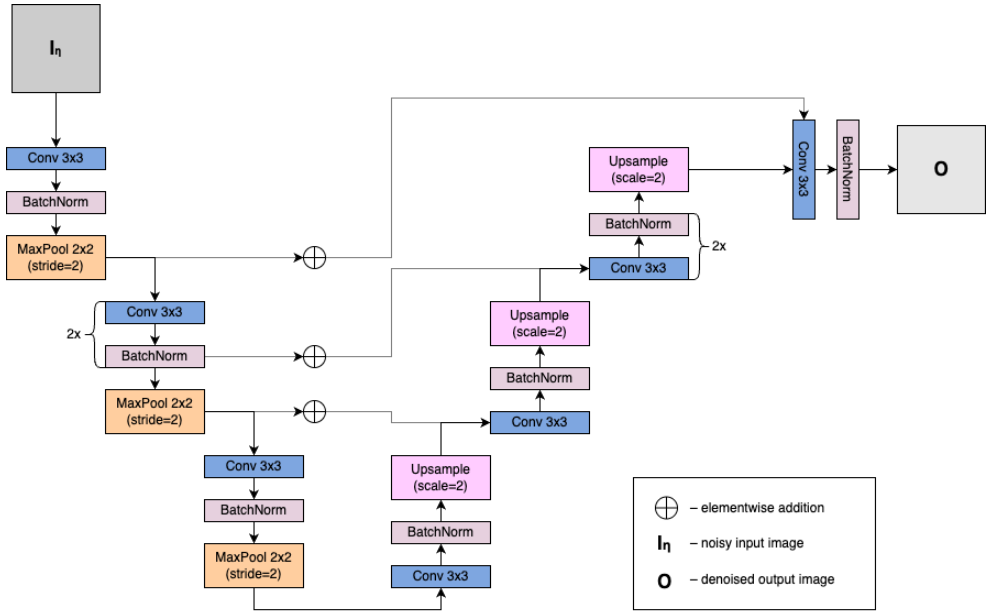


Figure 2.1: Denoising autoencoder architecture. Mohandas *et al.* [21] do not leverage the residual connections.

2.3 Prior works on measuring expression values from microarray images

Since one of the components of our work will involve measuring expression values from a microarray image (from this point onwards we shall refer to this process as *microarray readout*), we have conducted a literature search on existing readout methods and searched for free and open-source software to facilitate this process.

While it will soon become apparent that for our approach Algorithm 1 is optimal with respect to the readout error, we wished to compare Algorithm 1 against existing methods.

2.3.1 Software which did not pass the criteria

We have found a number of software applications for microarray data analysis. However, the most significant shortcoming of most of the tools we have encountered was that they were primarily designed for post-readout scenarios, where the user would be working with measured expression values. For instance, the AFFY package available for the R programming language and maintained by Thermo Fisher Scientific (after the acquisition of Affymetrix) only supports the CEL file format, which already contains measured intensities [29]. Other tools, such as MAIA [30] are outdated and cannot be run on modern systems. Some tools such as MIMAGER [31] only seem to support visualization of microarray image samples but not their analysis.

2.3.2 MAGIC Tool

Ultimately, we have decided to use the MicroArray Genome Imaging and Clustering Tool (MAGIC Tool) [28]. MAGIC Tool is an open-source microarray analysis suite implemented in the Java Programming Language which has been developed by Heyer *et al.* [28] and is available for download from the Davidson College website. This software suite aims to provide a user-friendly environment for researchers, enabling them to perform a wide range of analyses on microarray data from image processing to statistical analysis and visualization. While packages such as AFFY do not expose the interface for a direct analysis of images, MAGIC does permit raw microarray image loading.

MAGIC includes a suite of tools for microarray image analysis, focusing on dot identification and segmentation. The software offers 3 distinct algorithms for dot segmentation, including Fixed Circle, Adaptive Circle, and Seeded Region Growing. The Fixed Circle

method is parameterized by a user's selection of a circle radius r for dot segmentation. The Adaptive Circle method accepts a minimum and maximum radius ($r_{\min}$ and $r_{\max}$) and uses a circle for dot segmentation within that range. It is also thresholded by τ which indicates the percentile of pixels within the grid region that should be regarded as belonging to the dot. Seeded Region Growing starts with a singular foreground pixel and a background region and grows the segmented area pixel-by-pixel. This process is also controlled by a percentile threshold τ.

The software also supports various normalization techniques to correct for systematic biases in the data and supports clustering algorithms such as k-Means or QT-Clust for gene expression profile analysis. It also includes cluster visualization tools.

MAGIC Tool was designed to be highly flexible, supporting various microarray formats and accommodating different experimental designs. The software integrates multiple functions into a single platform, reducing the need for multiple tools and streamlining the analysis process. Indeed, the ability to support different microarray shapes and layouts is crucial for the success of a microarray analysis platform.

Because MAGIC Tool supports loading raw images and was simple to convert to a library callable from Python, we have chosen this software to facilitate microarray readouts for our work.

Chapter 3

Methods

In this chapter we cover the methodology for the following experiments:

- Wavelet Denoising (baseline) (method covered in Section 3.4.1)
- Autoencoder (baseline) (method covered in Section 3.4.2)
- Autoencoder trained on synthetic data (method covered in Section 3.4.3)
- Restormer trained on synthetic data (method covered in Section 3.4.4)
- Autoencoder with residual connections trained on synthetic data (method covered in Section 3.4.5)
- Autoencoder trained on normally-distributed synthetic data (method covered in Section 3.4.6)
- Autoencoder conditioned on ground-truth expression values (method covered in Section 3.4.7)

All results for the aforementioned methods are collectively discussed in Chapter 4. Before we cover individual denoising methods, we shall briefly look at the data and metrics used throughout the experiments and we come back to the denoising methods in Section 3.4.

3.1 Data

3.1.1 Baseline data

For the baseline training our intention was to reuse the exact same datasets that Mohandas *et al.* [21] have used in their work to ensure the most fair comparison possible. Their data corpus consisted of two datasets: DeRisi and SIB dataset.

Regrettably, the SIB dataset has been as of 2023 permanently removed from the internet. Searches on the Wayback Machine [32] as well as contacting the authors of the baseline work have ensured that no copy of the SIB dataset exists, and was therefore inaccessible to us.

Images in the DeRisi dataset are 2-channel monochromatic images, as expected from DNA microarrays. Consistent with Mohandas *et al.* , we have selected the cleaner channels from each pair of channels for the training dataset. Mohandas *et al.* do not state what criterion has been used for the selection of a cleaner channel. Indeed, without idealized clean images, assessing the noise quantitatively is not viable, hence we used visual heuristics for selection.

To compensate for a reduced number of samples, for each image in the DeRisi dataset we obtain 3 additional images via rotational augmentation, i.e. each image is rotated by 90°, 180°, 270° and added to the pool of available images. Following Mohandas *et al.* , we generate 20 noise-added images from one image in the pool, using Gaussian noise ($\mu = 0$, $\sigma = 30$ in $[0; 255]$ pixel intensity range).

We perform a train-test-validation split in proportions of $80\% : 10\% : 10\%$, arriving at 224 total training images, 28 validation and 28 test images.

We also introduce two variants of the DeRisi dataset, one that consists of images in their original dimensions with a sizeable background margin around the actual probe field and a cropped variant. We refer to these as *cropped* and *uncropped* respectively.

3.1.2 Synthetic data generation

For the synthetic data generation we have appreciated the uncanny regularity of a microarray image, where each dot corresponding to a DNA probe is an ellipsoid shape, closely resembling a circle on a black background. The intensity of each such probe could be easily modeled by the intensity of a patch of pixels close to the centroid of the shape. Therefore, we have decided to generate a large number of noise-free idealized DNA microarray images with no position shifts of the probes in the putative glass slide.

Generation

We have used the OpenCV [33] library to algorithmically draw 6724 circles in a fixed 82×82 grid on a perfectly black background. To emulate the natural irregularities in the shapes of the probe readouts, we have introduced a random-axis 1-pixel skew and a 1-pixel blur border to each of the circles. The diameter of each of the dots has been set to 8 pixels and the distance between dots has been set to 10 pixels. We generate margin-less microarray images but our data generation pipeline supports generation of arrays of padded images, permits flexible up- and down-scaling of the images and allows the user to define the shape and the number of generated dots in a flexible fashion. This implies our data generation pipeline can be used to cover nearly any shape and size of a microarray, which makes it usable as a dataset generator for circumstances where the shapes do not align with images that we have trained and tested our models on.

Determining the distribution

The intensity of each probe is not a fixed value. Hypothesizing that the best possible denoising method for microarray chips would involve a distribution of dot intensities closely resembling real-life gene expression values, we wished to model dot intensity using the correct distribution. The distribution of gene expression values has been estimated from prior RNA-Seq experiments.

We have searched for RNA-Seq samples in the Genomic Data Commons portal. We have selected a random RNA-Seq sample from the TCGA database (sample UUID: 26e28409-2493-4bd8-86f5-dc7b85846081) from the NCI Cancer Model Development for the Human Cancer Model Initiative study [34] [1] and used the Python Fitter package [1] to estimate which probability distribution would fit that sample. Chi-squared distribution was the distribution that fit the sample best (p-value: 0.0001) and we used the estimated values degrees-of-freedom (1), mean (-7.2125×10^{-29}) and standard deviation (2.217) parameters.

When generating dot intensity values we started with random normal variables:

$$X_i \sim \mathcal{N}(\text{mean}, \text{std_dev}^2) \quad \text{for } i = 1, 2, \ldots, \text{df}$$

We then converted X_i to standard normal variables Z_i using the transformation:

[1] The Cancer Model Development Center (CMDC) datasets are part of the Human Cancer Model Initiative (HCMI), supported by NCI contract HHSN261201500003I. The data used for this analysis are available at the NCI Genomic Data Commons (https://gdc.cancer.gov/). Information about HCMI can be found at https://www.cancer.gov/ccg/research/functional-genomics/hcmi

$$Z_i = \frac{X_i - \text{mean}}{\text{std_dev}}$$

Finally we summed the squares of the standard normal variables:

$$Q = \sum_{i=1}^{\text{df}} Z_i^2$$

The sum Q follows a chi-squared distribution with df degrees of freedom given by:

$$Q \sim \chi^2(\text{df})$$

Synthetic dataset

Ultimately we generate a total of 10000 synthetic microarray images, where dot intensity values are sampled from the distribution mentioned in the previous section. We perform a conventional 80%/10%/10% training/test/validation split.

3.1.3 Image noise injection

For all training protocols we required a way of adding noise to the input images. In the case of the baseline autoencoder method used by Mohandas *et al.* , additional noise has been applied to already naturally-noisy images and reconstruction loss was calculated between the denoised image and the input image to drive the training. We follow the same noise injection protocol with Gaussian noise and salt-and-pepper noise added to the appropriate images. Parameters used for noise injection in the case of Mohandas *et al.* [21] are unclear. In our case we have decided to apply Gaussian noise with $\mu = 0.0, \sigma = 30.0$ and we set the salt-and-pepper noise probability at 10%.

In the case of synthetic data we use the same noise injection configuration but we make the denoising problem more challenging by adding Blob Noise, i.e. noise consisting of circular semi-translucent patches emulating environmental sources of noise such as dust particles or mistakes in microarray slide preparation. Our data generation framework also supports adding random-polygon shaped blob noise. A visual comparison between a DeRisi microarray with such environmental noise and a synthetic microarray image with blob noise added can be viewed in Figure 3.1.

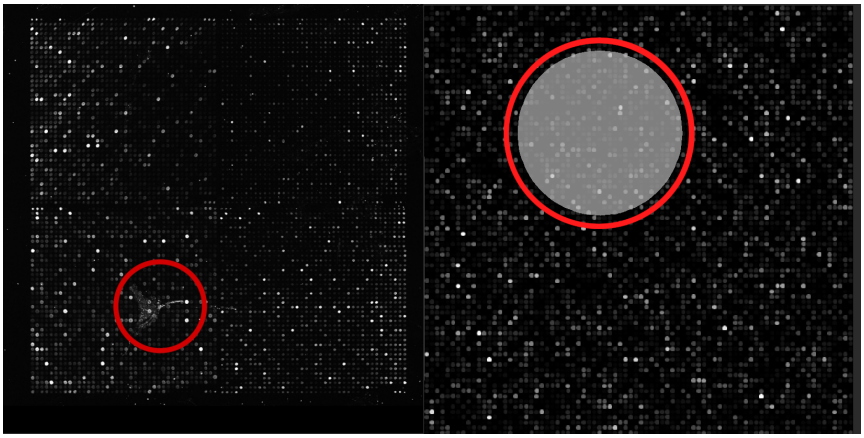


Figure 3.1: Comparison of the environmental noise present in real microarray images (courtesy of J. DeRisi [17]) from the DeRisi dataset and the Blob Noise added by us to the synthetic images.

3.1.4 Image representation

Before proceeding, it is worth discussing a few nuances in image representation that we assume in this work. The DeRisi images required normalization. Loading them directly without normalization resulted in abnormal color saturation. The synthetic images have been prepared against a perfectly black background and without any high-intensity pixel artefacts (free of noise), thus applying normalization would shift the pixel intensity distribution leading to misrepresented images. For this reason we normalize only real-world images of microarrays in our experiments.

We do however apply *pixel value scaling* to all loaded images to keep pixel values in the range $[0.0; 1.0]$. We do this to normalize datasets and enable relative comparisons between samples. This operation does not induce information loss or distribution skew that would be significant from the experimental point of view, since microarrays are a quantitative method that is inherently relative (requiring a reference image for measurement). From this point onward we will refer to this representation bounded in the range $[0.0; 1.0]$ as floating-point image representation.

3.2 Metrics

3.2.1 PSNR and f-PSNR

Following Mohandas *et al.* mean squared error (MSE) has been chosen as reconstruction loss and adopted as a training criterion. We also report two qualitative metrics: PSNR for our own dataset and flipped peak signal-to-noise ratio f-PSNR for the DeRisi dataset.

$$PSNR[\text{db}] = 10 \cdot \log_{10} \left(\frac{(\text{Max possible pixel value})^2}{MSE} \right) \tag{3.1}$$

The two methods differ in the way MSE is calculated. Flipped-PSNR corresponds to the definition of PSNR used by Mohandas *et al.* , who treat the *output* image as the noise free image. Let I_η be the noise-added input image, O the denoised output image, I the original image with no extra noise added and G a ground-truth image. Then, the MSE calculation for the Flipped-PSNR metric can be expressed with the following equation:

$$MSE_{(I_\eta,O)} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (I_\eta(i,j) - O(i,j))^2 \tag{3.2}$$

Where possible, i.e. wherever we can use ground-truth images which are guaranteed to have no noise, we use the PSNR metric, calculated between the ground-truth image (G) and the output image (O). In this context the definition of MSE is:

$$MSE_{(G,O)} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (G(i,j) - O(i,j))^2 \tag{3.3}$$

A clear distinction between PSNR and f-PSNR metrics is necessary to account for the fact that the synthetic dataset we have developed provides idealized clean images, while the DeRisi dataset consists of naturally-noisy real microarray images. Obtaining a ground truth for real datasets could be done by means of domain expert evaluations, which is out of the scope of this work.

PSNR is thus in the context of this work only meaningful for the synthetic dataset and the larger its value the greater the denoising power of the model in question. f-PSNR is conversely meaningful only for the real-life microarray images and the smaller its value, the greater the denoising power of the model. Intuitively, if the noise-added image (I_η) and

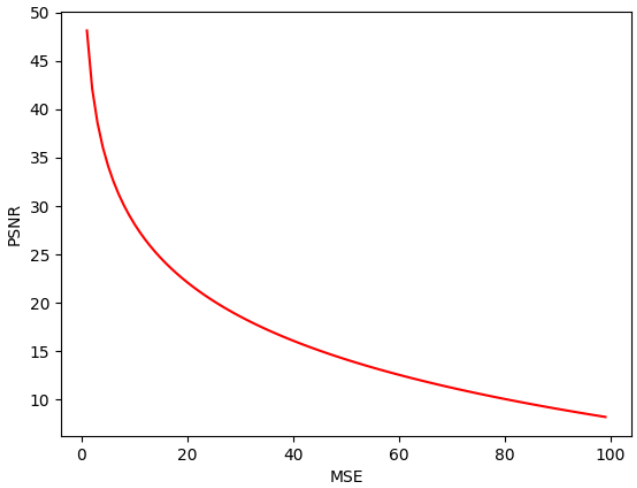


Figure 3.2: PSNR tends to infinity as MSE tends to zero. Conversely, when the MSE term tends to infinity, PSNR tends to zero.

output image (O) are very similar, then $MSE_{(I_\eta,O)}$ should be close to zero. This happens when the model has failed to capture and remove noise from I_η. As MSE tends to zero and at a fixed value of the maximum possible pixel intensity, f-PSNR increases since the MSE term is present in the denominator (Figure 3.2).

Consequently, for the DeRisi dataset we report the results of f-PSNR and we do not report the results for PSNR. Due to the presence of natural environmental noise in the input image, it is not possible to treat these images as ground-truth and this way of quantification of noise would be meaningless. Conversely, for the synthetic dataset, we report only the PSNR metric.

3.2.2 SADGE

Although PSNR is a well-established metric used to quantify the success of the denoising process in general image denoising tasks, neither PSNR nor f-PSNR bear any biological significance and fail to quantify how successful the denoising process was in the context of microarrays' function. Namely, DNA microarrays are in a normal laboratory workflow scanned and converted to relative gene expression strength values. PSNR and f-PSNR do not account in any way for how accurate or precise the microarray readout remains after a

denoising model has been applied. In other words: PSNR and f-PSNR do not inform about whether the denoising model captured only noise or also inadvertently removed relevant information.

One could imagine a simple scenario where a given denoising model yields visually spectacular results, where almost no noise is perceptible to the human eye and the confidence in the model's capability is reinforced by the fact that e.g. the PSNR metric on the test split of the synthetic dataset has been comparatively high. Yet, it might be the case that the model has in fact slightly altered the intensities of dots (ι), leading to an erroneous assessment of gene expression values (e).

We address this glaring issue through the introduction of a new metric applicable to the evaluation of denoising models for the specific problem of microarray denoising.

We leverage existing an existing tool to facilitate conversion of images to gene expression values. We use the *MicroArray Genome Imaging & Clustering Tool* (MAGIC Tool) [28] for this purpose. Later, we will also show a custom algorithm that outperforms methods offered by the MAGIC Tool.

MAGIC accepts a two-channel microarray image, applies a grid to the image, performs dot segmentation and measures relative intensity of the dots. After normalization, these relative intensity values are representative of a transcriptomic profile of a given tissue or genomic signature of a given organism. We consider these measurements pertinent to the qualitative assessment of denoising methods as they bear crucial biological meaning. We discuss the capabilities of MAGIC and compare its precision against a simplistic implementation of microarray reader in Python programming language in Section 4.1.

Notably, MAGIC Tool requires a reference image and performs only relative intensity measurements. This is consistent with the experimental protocol for microarray scanning and post-processing, we thus approach the problem of measuring intensities from the same standpoint. To measure the intensities in each image against the same baseline, we generate a **reference** microarray image of the same shape as the image being measured. The reference image consists of the same number of dots, each of which is set to a constant pixel value intensity of 0.5 in a floating-point representation (Figure 3.3).

We then propose a new metric for the specific task of microarray denoising and we name it SADGE. SADGE is defined as mean absolute error (MAE) between measured gene expression values from a ground-truth image and gene expression values from a denoised image:

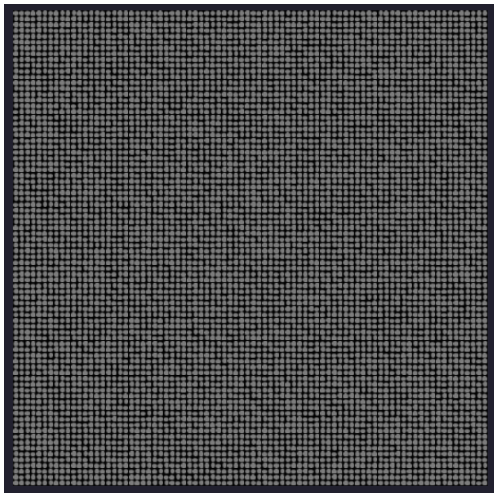


Figure 3.3: Reference image for relative dot intensity measurement. Each dot is set at exactly 50% of the maximum pixel intensity (0.5 in floating point representation)

$$SADGE = \log \frac{1}{nm} \sum_{i}^{n} \sum_{j}^{m} |\text{imeasure}(R_{ij}, G_{ij}) - \text{imeasure}(R_{ij}, O_{ij})| \tag{3.4}$$

Where:

- n and m are the rows and columns for a gridded microarray image and should be equal to the number of dots in the image
- imeasure is the pixel intensity measurement method which varies depending on the configuration
- R is the reference image

Intuitively, if the MAE between the two sets of values is large, that indicates the model has not successfully removed all of the noise from the input image. We choose to apply a decimal logarithm to MAE to make the it more responsive to small differences between gene expression values as demonstrated in Figure 3.4. Log-MAE also has the desirable property, where all values below 0 indicate an average error on the order of less than 100%

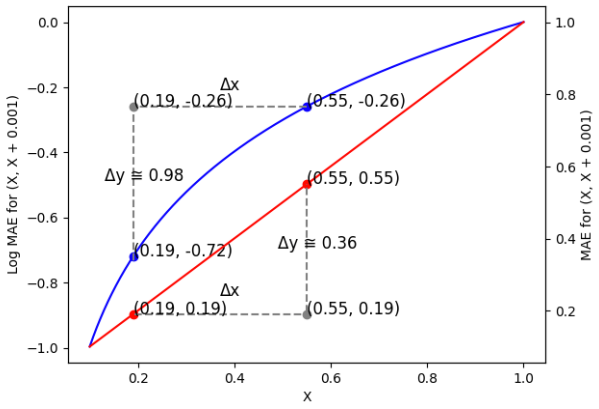


Figure 3.4: A comparison of MAE (red) and Log-MAE (blue). For small error values (< 1), Log-MAE has a higher resolution than MAE.

for images in the floating-point image representation. Conversely a SADGE value > 2 would indicate that on average the measured gene expression values are > 2 orders of magnitude away from their ground truth values.

We find SADGE simpler to interpret and more meaningful in the context of microarray denoising than PSNR, since it relates the biologically-significant expression values post-denoising to a ground-truth value and this relationship is preserved at all granularity levels, from a single gene to the entire microarray.

Applicability of the SADGE metric

SADGE cannot be meaningfully interpreted for situations in which no ground truth image exists. Any environmental noise that is present in a real-life microarray image might affect the intensities of the dots and thus might modify the measured expression values. Obtaining real-world microarray images which are considered clean would require domain expert evaluators, who could judge images as clean vs. noisy. Building such a dataset would require a considerable amount of effort and resources and would constitute a separate publication, outside of the scope of innovations we present in this work. Moreover, there is no guarantee that the real images contain no noise whatsoever. Therefore, we do not report results for the SADGE metric when testing on the DeRisi dataset.

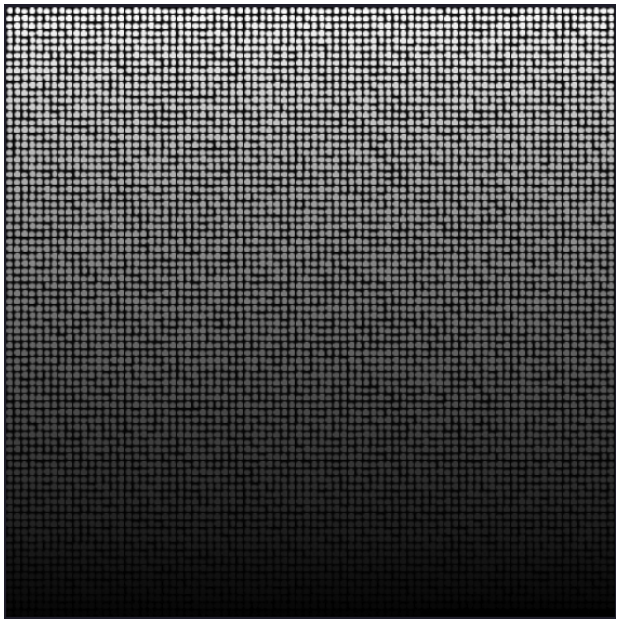


Figure 3.5: Step-gradient image we used to test the consistency of microarray readouts from the MAGIC Tool [28] and using the MaxPool Readout (Algorithm 1).

Dot intensity measuring methods

In Equation 3.4 we have left the imeasure operator loosely defined. How each individual dot intensity is measured from an image can vary. We compare 3 different dot segmentation methods offered by the MAGIC Tool which we previously mentioned in Section 2.3.2 and we present the results in Section 4.1. We also introduce a custom microarray readout algorithm (Algorithm 1) which we dub the MaxPool Readout.

We apply the strategy of measuring relative intensities of each dot against a reference image. As the test image we created a step-gradient image (Figure 3.5).

For the microarray shape used (82×82, a total of 6724 dots), we set the upper-left corner dot intensity to 1.0 and the bottom-right corner dot intensity to 0.0. The remaining 6722 dots are processed left-to-right and top-to-bottom with each successive dot intensity set to $\iota_{i-1} + \iota_{i-1}\frac{1}{6722}$. In general terms the same strategy can be applied to any microarray image of the shape $n \times m$:

Algorithm 1 MaxPoolReadout – Custom microarray expression value readout algorithm based on the concept of Max Pooling.

Input: img, ref_img, k, ϵ
Ensure: img.height = ref_img.height
Ensure: img.width = ref_img.width
  $h \leftarrow$ img.height
  $w \leftarrow$ img.width
  $s \leftarrow k$
  procedure MaxPool2D(img, k, s)
    Initialize output with zeros of size $\left\lfloor \frac{h-k}{s} + 1 \right\rfloor \times \left\lfloor \frac{w-k}{s} + 1 \right\rfloor$
    for each i from 0 to h with step s **do**
      for each j from 0 to w with step s **do**
        extract patch from img of size $k \times k$ starting at (i, j)
        find the maximum value in the patch
        assign this maximum value to the corresponding position in the output
      end for
    end for
    return output
  end procedure
  procedure Flatten(output)
    Initialize flat_output as an empty list
    for each row in output **do**
      for each value in row **do**
        append value to flat_output
      end for
    end for
    return flat_output
  end procedure
  output $\leftarrow$ Flatten(MaxPool2D(img, k, s))
  ref_output $\leftarrow$ Flatten(MaxPool2D(ref_img, k, s)) + ϵ
  return $\frac{\text{output}}{\text{ref_output}}$

$$\iota_{ij} = \begin{cases} 0.0 & \text{if } (i,j) = (0,0) \\ 1.0 & \text{if } (i,j) = (n,m) \\ \iota_{i-1j-1} + \iota_{i-1j-1}\frac{1}{nm-2} & \end{cases} \tag{3.5}$$

The reference image used can be viewed in Figure 3.3 and the step-gradient image can be viewed in Figure 3.5. Equation 3.5 also informs about the expected gene expression values:

$$e_{ij} = \frac{\iota_{ij}^{(\text{sample})}}{\iota_{ij}^{(\text{ref})}} \tag{3.6}$$

These are used to compare different readout methods and discussed further in Section 4.1.

3.3 Evaluation on real microarray data

Denoising models would be in fact useless if their performance on real microarray image data had been poor. To assess the generalization capabilities of the denoising models trained on synthetic data, we have held out a test set from the augmented DeRisi dataset and we evaluated each of the trained models.

Because for real-world data no objective ground-truth can be derived, for this test split we only calculate PSNR (or f-PSNR in the case of the baseline autoencoder model) and we do not report results for SADGE.

3.4 Denoising methods

In this section we describe methods and models used for denoising. For results, please refer to Chapter 4.

3.4.1 Wavelet denoising

We have used used the default implementation of wavelet denoising provided by the SKIMAGE Python package. We have opted to use the BAYESSHRINK method as recommended by the documentation with the default parameters.

Hyperparameters	Values Used	Default Values
NUM_BLOCKS	$(1, 2, 2, 2)$	$(4, 6, 6, 8)$
NUM_HEADS	$(1, 4, 4, 4)$	$(1, 2, 4, 8)$
CHANNELS	$(6, 12, 24, 48)$	$(48, 96, 192, 384)$
NUM_REFINEMENT	1	4

Table 3.1: Hyperparameters used to instantiate the Restormer model.

3.4.2 Baseline autoencoder

The denoising autoencoder architecture presented by Mohandas *et al.* [21] has been used to provide an experimental baseline. The architecture diagram can be found for reference in the Figure 2.1. The architecture follows the UNet design pattern [35] with progressively smaller output sizes from the pooling layers on the downward branch and progressively larger output sizes from the upsampling layers on the upward branch. No residual connections were present in the architecture chosen by the authors of the prior work.

3.4.3 Autoencoder trained on synthetic data

For the first part of our experimental work we leverage an identical architecture to that used by Mohandas *et al.* and we simply train it on a much larger corpus of synthetic data. Real-life microarray images have been completely left out from the training process, as we wished to validate the generalization capabilities of the model post-training without leveraging data with natural environmental noise. More discussion about the data can be found in Section 3.1.

3.4.4 Restormer trained on synthetic data

We used the official Restormer architecture implementation made available by Zamir *et al.* [27].

To make training of a Restormer architecture feasible on the compute resources we had available to us, we have reduced the depth of the model, the number of attention heads, the number of channels in the blocks and we set the number of refinement cycles to 1. Hyperparameters used to instantiate the model are outlined in Table 3.1.

3.4.5 Autoencoder with residual connections

Since Mohandas *et al.* have used a basic autoencoder architecture without any residual connections, we have decided to add skip connections between the encoder and decoder branches for one of our experiments, to investigate whether residuals would improve reconstruction capabilities of the model.

3.4.6 Autoencoder trained on normally-distributed data

To assess the impact of distribution shifts in training data on test-stage performance, we have also trained an autoencoder architecture on normally-distributed data with $\mu = 0.0$ and $\sigma = 0.2$. Subsequently we tested the trained model on the test split from the data generated according to the chi-squared distribution (see Section 3.4.3).

3.4.7 Autoencoder conditioned on ground-truth expression values

Because for the synthetic data ground truth expression values can be measured prior to the training process, we decided to investigate whether conditioning the autoencoder architecture on measured gene expression values could improve the denoising power of a model.

As will be described in detail in Section 4.6, residual connections have not improved SADGE scores, therefore we decided to start with the same architecture originally used by Mohandas *et al.* [21] and introduce a cross-attention mechanism between the representation of an image within the autoencoder and the corresponding ground-truth expression values.

At training, a noise-added input image I_η has this time been paired with its corresponding ground truth image G and a relative intensity measurement has been obtained from G for each dot in the grid using the MaxPool Readout Algorithm (see Algorithm 1). We have only used the aforementioned algorithm to measure the intensities, since by this stage in our work we have proven that this way of measuring dot intensities is characterized by a significantly lower MAE, as discussed in detail in Section 4.1.

The obtained expression values $\mathbf{e}_{\text{gt}}$ have been fed to a cross-attention-like module which we dub EATME. Given a vector of ground-truth expression values $\mathbf{e}$, we reshape it to the same grid configuration as the configuration of dots on the microarray slide (h_g, w_g) and

for each position in the grid we upsample the image to fit the spatial dimensions of the reconstructed image (h_O, w_O):

$$E_{\text{upsampled}} = \text{resize}(\text{reshape}(\mathbf{e}_{\text{gt}}, (h_g, w_g)), (h_O, w_O)) \tag{3.7}$$

Then we use the following cross-attention-like module:

$$EATME(O_\xi, \mathbf{e}) = \sigma(E_{\text{upsampled}} * W * O_\xi) \tag{3.8}$$

Here, O_ξ stands for the pre-output-block feature map with the same spatial dimensions as the input image (for example if the input tensor was of the shape $(b, 1, h, w)$ then the feature map will be of the shape $(b, 32, h, w)$ for the autoencoder architecture we used).

Finally the EATME scores are applied elementwise to the O_ξ feature map before the last convolutional block:

$$O = \text{out}(EATME(O_\xi, \mathbf{e}) * O_\xi) \tag{3.9}$$

out stands for the final convolutional block in the autoencoder, which yields a reconstructed single-channel image given a feature map of the same spatial dimensions. Each of the upsampled expression values in $E_{\text{upsampled}}$ covers a grid cell, where the dot in the reconstructed image is expected and provides the exact information as to how bright the dot is expected to be.

At inference, the expression values have not been passed to the model at all and only the latent image representation was used for reconstruction by the decoder.

We have also used an additional loss term which we call DEL to further enhance the capability of the model to maintain the correct dot intensities at each position:

$$\mathcal{L}_{DEL} = MSE(\mathbf{e}_G, \mathbf{e}_O) \tag{3.10}$$

Where $\mathbf{e}_G$ indicates the expression value vector collected from the ground-truth image, while $\mathbf{e}_O$ indicates the expression value vector collected from the denoised image.

A visual summary of the architecture is presented in Figure 3.6.

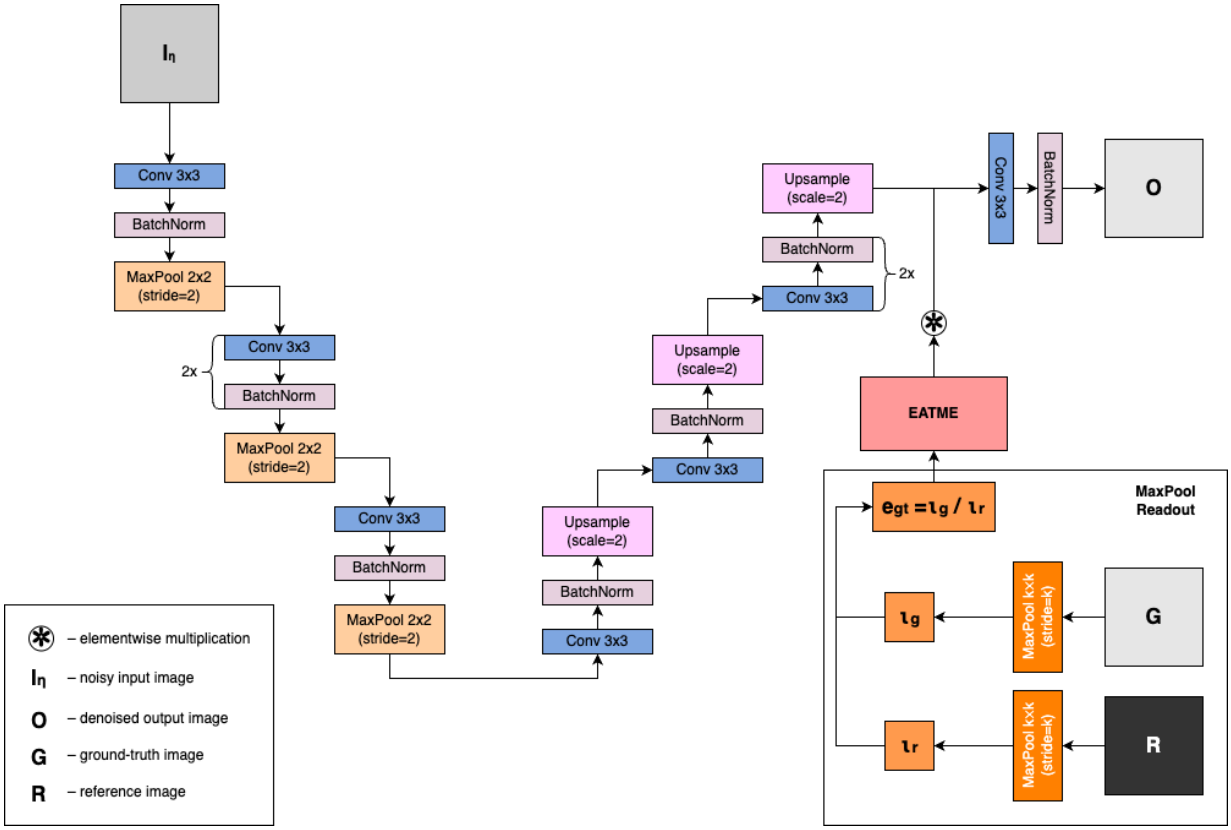


Figure 3.6: Autoencoder architecture with EATME module.

Chapter 4

Results

We start the discussion of results by looking at factors driving the SADGE metric:

- Evaluating the MAGIC Tool [28] as a microarray readout tool for the calculation of SADGE (results covered in Section 4.1).
- Calculating SADGE values on noise-added input images (I_η) (results covered in Section 4.2).

Next, we present results for the following experiments:

- Wavelet Denoising (baseline) (results covered in Section 4.3.1)
- Autoencoder (baseline) (results covered in Section 4.3.2)
- Autoencoder trained on synthetic data (results covered in Section 4.4)
- Restormer trained on synthetic data (results covered in Section 4.5)
- Autoencoder with residual connections trained on synthetic data (results covered in Section 4.6)
- Autoencoder trained on normally-distributed synthetic data (results covered in Section 4.7)
- Autoencoder conditioned on ground-truth expression values (results covered in Section 4.8)

Finally, we summarize the results collectively in Section 4.9 to arbitrate the best-to-date denoising model for microarray images.

Readout method	MAE (expected vs. observed)
MaxPool Readout	**0.0045**
Fixed Circle ($r = 4$)	0.0961
Fixed Circle ($r = 5$)	0.1061
Seeded Region Growing ($\tau = 5$)	0.1702
Seeded Region Growing ($\tau = 25$)	0.1615
Seeded Region Growing ($\tau = 50$)	0.1701
Adaptive Circle ($\tau = 25$)	0.0297
Adaptive Circle ($\tau = 50$)	0.0297
Adaptive Circle ($\tau = 75$)	0.0297

Table 4.1: Comparison of gene expression readout methods. MaxPool Readout is our proposed method (see Algorithm 1). Fixed Circle, Seeded Region Growing and Adaptive Circle are dot segmentation methods offered by the MAGIC Tool.

4.1 Evaluating MAGIC Tool on synthetic data

To evaluate the error levels of the MAGIC Tool when preparing the relative gene expression readouts from microarray images, we have applied a range of dot segmentation methods as well as a range of parameters for those segmentation methods. We have also compared the results from MAGIC to the MaxPool Readout Algorithm (see Algorithm 1).

Equipped with the expected expression values (as described in Section 3.2.2) we calculated the relative intensity of each of the dots and compare it against the expected value by taking MAE. We apply this protocol to 3 different dot segmentation methods offered by the MAGIC Tool as well as to our own MaxPool Readout implementation. The comparison between all the scanning methods mentioned here can be found in Table 4.1.

For the Adaptive Circle method we used $r_{\min} = 2$ and $r_{\max} = 4$. Of the methods offered by the MAGIC Tool, the best-performing was the Adaptive Circle method, where the threshold parameter did not result in any variability in performance. The runner-up was Fixed Circle with $r = 4$, which was the first choice for computation of the SADGE metric.

Notably, using a MaxPool Readout method, which we proposed, resulted in a much better performance than any other of the methods offered by the MAGIC Tool. Nevertheless, reliability of this method is hard to evaluate on real-life microarray data, unless a large

dataset pairing RNA-Seq and microarray experiments from the same biological samples is created. This is cost-prohibitive and we leave this research question lingering for future studies.

In the case of dot segmentation methods offered by the MAGIC Tool, the high-intensity dots suffered from higher-error readouts as demonstrated in Figure 4.1.

4.2 SADGE calculated on input images

To quantify by how much the noise added to the synthetically-generated images which were used as an input for training the denoising models has skewed the gene expression values in comparison with the ground truth, we have calculated the SADGE metric for two of the synthetically-generated datasets used in our work. The results are presented in Table 4.2. A higher SADGE value for the dataset generated according to the chi-squared distribution indicates that the same noise-addition protocol resulted in a much larger corruption of expression values.

Dataset	**Mean SADGE (MP)**
Synthetic Chi-squared	0.2253
Synthetic Gaussian	-0.0083

Table 4.2: SADGE values for input images. MP – MaxPool Readout method

4.3 Baseline

4.3.1 Wavelet denoising

Since wavelet denoising does not involve learning a model, we report results only for the test splits. Results are summarized in Table 4.3.

Qualitatively, the denoising process has largely failed to capture most of the added noise. In Figure 4.2 we show a close-up on the corner of a synthetic microarray image and we notice that the amount of noise removed is insignificant. This stays consistent with the quantitative results, where f-PSNR on the DeRisi dataset remains quite high (over 25 for

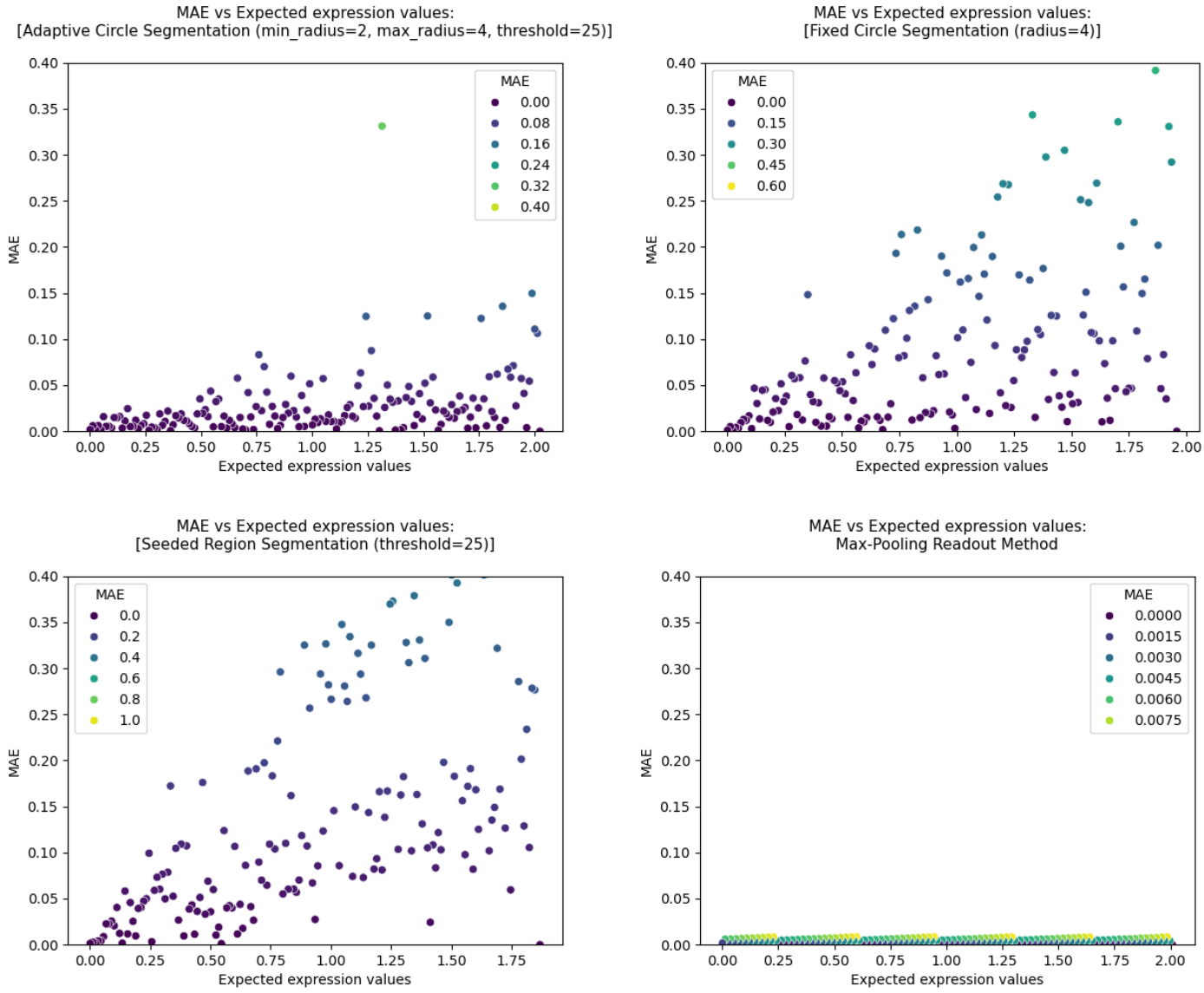


Figure 4.1: MAE between expected binned expression values (bin size 40) and detected expression values using different readout methods. Left-to-right, top-to-bottom: Adaptive Circle method, Fixed Circle method, Seeded Region Growing method, MaxPool Readout method

Dataset	**PSNR** [dB]	**f-PSNR** [dB]	**SADGE (MP)**	**SADGE (AC)**
Synth Test	11.9464	N/A	0.1998	2.2700
DeRisi Test (no crop)	N/A	25.2812	N/A	N/A
DeRisi Test (crop)	N/A	22.0768	N/A	N/A

Table 4.3: Quantitative results for wavelet denoising. MP – MaxPool Readout, AC – Adaptive Circle

the uncropped dataset and over 22 for the cropped dataset) and the PSNR value remains very low for the synthetic dataset (below 12).

4.3.2 Baseline autoencoder

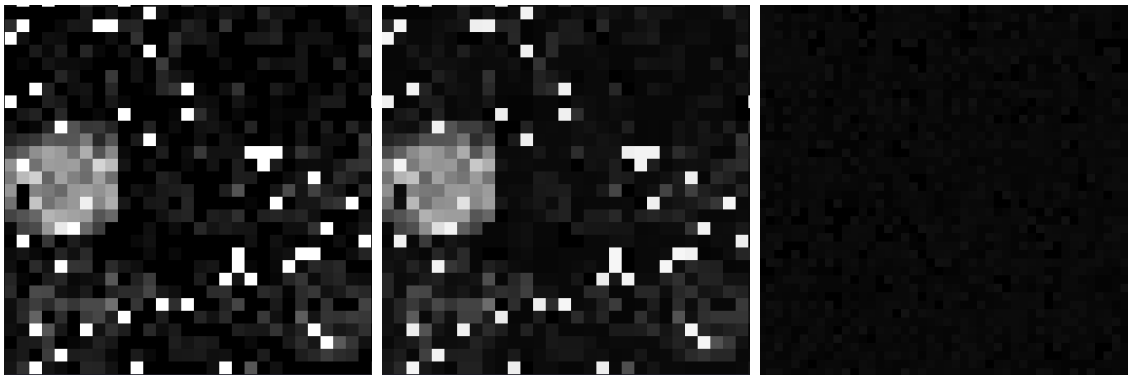
For the baseline autoencoder method we report results in the Table 4.4. We notice a slight improvement in PSNR values on the synthetic dataset test split as well as a modest improvement in terms of f-PSNR for the DeRisi test split. SADGE values calculated according to the MaxPool Readout method are also now below 0 which indicates a significant improvement in terms of expression readout error. A comparison of the input, output and ground truth for the synthetic test split can be viewed in Figure 4.3.

4.4 Autoencoder trained on synthetic data

Our first attempt at training a new denoising model introduced solely a change in the training data. In place of the DeRisi train split, we used 8000 synthetically-generated images for training and 1000 for validation. We held out 1000 images for testing on a holdout set. Generation and noise injection parameters for the input images have been described in Section 3.1.2 and Section 3.1.3 respectively. No hyperparameter change occurred before training and no changes to architecture have been introduced. The results are summarized in Table 4.5. A visual comparison of the outputs can be found in Figure 4.3.

4.5 Restormer trained on synthetic data

We trained a 2022 state-of-the-art denoising architecture on our custom synthetic microarray dataset and we have observed a significant improvement in PSNR for the test split



(a) Input into the wavelet denoising algorithm with Gaussian and salt-and-pepper noise (left-bottom-corner close-up).

(b) Output from the wavelet denoising algorithm (left-bottom-corner close-up).

(c) Difference between (a) and (b) representing the noise captured by the wavelet denoising algorithm. The noise captured is of very low intensity.

Figure 4.2: An example of a poor qualitative performance of wavelet denoising.

Dataset	PSNR [dB]	f-PSNR [dB]	Loss	SADGE (MP)	SADGE (AC)
DeRisi Train (no crop)	N/A	26.0758	0.0024	N/A	N/A
DeRisi Val (no crop)	N/A	26.2424	0.0024	N/A	N/A
DeRisi Train (crop)	N/A	23.2468	0.0047	N/A	N/A
DeRisi Val (crop)	N/A	22.9231	0.0051	N/A	N/A
Synth Test (trained on no-crop)	16.2730	N/A	0.0247	-0.5749	2.4523
Synth Test (trained on crop)	17.7472	N/A	0.0184	-0.6397	2.2459
DeRisi Test (no crop)	N/A	26.2684	0.0024	N/A	N/A
DeRisi Test (crop)	N/A	23.4593	0.0046	N/A	N/A

Table 4.4: Results for the baseline Autoencoder, following Mohandas *et al.* [21].

Dataset	PSNR [dB]	f-PSNR [dB]	Loss	SADGE (MP)	SADGE (AC)
Synth Train	28.0678	N/A	0.0015	-1.1037	2.2449
Synth Val	18.4170	N/A	0.0143	-0.6499	2.3399
Synth Test	28.1942	N/A	0.0015	-1.1172	2.2395
DeRisi Test (no crop)	N/A	22.5481	0.0112	N/A	N/A
DeRisi Test (crop)	N/A	22.0562	0.0090	N/A	N/A

Table 4.5: Results for the autoencoder trained on synthetic data.

but worse f-PSNR on the DeRisi test splits in comparison with the autoencoder trained on synthetic data (Section 4.4). Interestingly SADGE values for the synthetic test split were significantly worse than for the autoencoder, despite the fact that the outputs look at a first glance visually more appealing than those produced by the autoencoder model. Upon closer inspection one can notice that the dot intensities are significantly skewed in the

output image prepared by Restormer (Figure 4.3). Quantitative results are summarized in Table 4.6.

Dataset	**PSNR** [dB]	**f-PSNR** [dB]	**Loss**	**SADGE (MP)**	**SADGE (AC)**
Synth Train	30.7834	N/A	0.0168	-1.0798	2.3882
Synth Val	18.4625	N/A	0.0636	-0.5092	2.4790
Synth Test	29.4749	N/A	0.0208	-1.0804	2.3886
DeRisi Test (no crop)	N/A	24.5517	0.0410	N/A	N/A
DeRisi Test (crop)	N/A	22.5572	0.0442	N/A	N/A

Table 4.6: Quantitative results for the Restormer model trained on synthetic data.

4.6 Autoencoder with residual connections

Adding residual connections to the autoencoder architecture originally used by Mohandas *et al.* [21] did not improve SADGE, PSNR and f-PSNR. We observed marginally faster model convergence with the residual connections enabled at the expense of reconstruction quality (convergence at epoch 13 vs. epoch 15 for the autoencoder without residuals). Results are summarized in Table 4.7.

Dataset	**PSNR** [dB]	**f-PSNR** [dB]	**Loss**	**SADGE (MP)**	**SADGE (AC)**
Synth Train	28.0415	N/A	0.0015	-1.0861	2.2484
Synth Val	18.4156	N/A	0.0144	-0.6423	2.3494
Synth Test	28.1569	N/A	0.0015	-1.1012	2.2415
DeRisi Test (no crop)	N/A	22.6085	0.0107	N/A	N/A
DeRisi Test (crop)	N/A	22.0000	0.0093	N/A	N/A

Table 4.7: Quantitative results for the autoencoder model with residual connections trained on synthetic data.

4.7 Autoencoder trained on normally-distributed data

As mentioned in 3.4.6, we have also trained the original autoencoder architecture on normally-distributed data to investigate the impact of a distribution shift. Testing on the data distributed according to the chi-squared distribution has shown that the autoencoder model cannot easily transfer across different distributions without re-training. PSNR for the synthetic test split was much lower than that for the autoencoder trained on data that was sampled from the same distribution. Results are summarized in Table 4.8.

Dataset	**PSNR** [dB]	**f-PSNR** [dB]	**Loss**	**SADGE (MP)**	**SADGE (AC)**
Synth Train (Gaussian)	33.5757	N/A	0.0004	-1.2470	2.2138
Synth Val (Gaussian)	18.2088	N/A	0.0151	-0.6321	2.2789
Synth Test (Chi-squared)	17.4355	N/A	0.0180	-0.4674	2.1297
DeRisi Test (no crop)	N/A	20.6271	0.0163	N/A	N/A
DeRisi Test (crop)	N/A	20.6842	0.0143	N/A	N/A

Table 4.8: Quantitative results for the autoencoder model trained on a normal distribution and tested on the chi-squared-following data as well as the DeRisi dataset.

4.8 Autoencoder with cross-attention

Activation of the EATME cross-attention module and addition of DEL to the total loss score of the model at training has significantly improved the capability of the autoencoder model to perform denoising at test stage. This method achieves the lowest SADGE score of all methods tested and outperforms the autoencoder trained on synthetic data on PSNR scores on the synthetic dataset test split. It achieves a slightly worse f-PSNR score on the DeRisi dataset.

Dataset	PSNR [dB]	f-PSNR [dB]	Loss	SADGE (MP)
Synth Train	28.2110	N/A	0.0015	-1.1207
Synth Val	18.5038	N/A	0.0141	-0.6463
Synth Test	28.1159	N/A	0.0015	-1.1243
DeRisi Test (no crop)	N/A	22.7248	0.0107	N/A
DeRisi Test (crop)	N/A	22.2672	0.0089	N/A

Table 4.9: Quantitative results for the autoencoder with EATME and DEL.

4.9 Comparison of results

To facilitate model-to-model comparison, we rolled up the results from individual tables. Table 4.10 shows SADGE scores for all tested methods, Table 4.11 shows PSNR scores pertinent to testing on the synthetic data, Table 4.12 shows f-PSNR scores pertinent to testing on the DeRisi dataset.

Method	SADGE (MP, Synth test)
AE Baseline	-0.6397
AE Synth	-1.1172
Restormer	-1.0804
AE Synth (with residuals)	-1.1012
AE Synth (normal)	-0.4674
AE Synth (EATME, DEL)	**-1.1243**

Table 4.10: SADGE scores on the synthetic dataset test split calculated using the MaxPool Readout method (lower is better). Bolded is the best performing denoising method in this context.

With respect to the lowest (best) SADGE value, the most successful model in our experiments has turned out to be the conventional Autoencoder architecture without residual connections, with DEL and EATME enabled.

An important observation is the fact that in order to assess the denoising power of a given model, visual analysis of the images is not sufficient post-denoising, nor is just looking solely at the PSNR metric. Under the PSNR metric, Restormer exhibits exceptionally

Method	**PSNR** [dB] (Synth test)
AE Baseline	17.7472
AE Synth	28.1942
Restormer	**29.4749**
AE Synth (with residuals)	28.1569
AE Synth (normal)	17.4355
AE Synth (EATME, DEL)	28.1159

Table 4.11: PSNR scores on the synthetic dataset test split (higher is better). Bolded is the best performing denoising method in this context.

Method	**f-PSNR** [dB] (DeRisi crop test)
AE Baseline	23.4593
AE Synth	22.0562
Restormer	22.5572
AE Synth (with residuals)	22.0000
AE Synth (normal)	**20.6842**
AE Synth (EATME, DEL)	22.2672

Table 4.12: f-PSNR scores on the DeRisi dataset test split (lower is better). Bolded is the best performing denoising method in this context.

good performance and the images reconstructed by the model *look clean*. However, a higher SADGE metric suggests that Restormer while performing reconstruction also tends to remove or modify relevant information in the image.

It's also worth discussing the visual results produced by each of the models in the study. Figure 4.3 shows a comparison between outputs. Baseline autoencoder was not capable of removing most of the blob noise (4.3c). Output samples from the autoencoder (Figure 4.3d,f) all show very good local representation of the dots with no apparent deviations in intensity visible to the naked eye. Meanwhile, the output image from the Restormer model (Figure 4.3e) confabulates different dot intensities in certain locations of the image (Figure 4.4). This poor performance is reflected in a comparatively high SADGE value for the model. If for a pair of images (G, O) there exists a single pair of dots $(g \in G, o \in O)$, such that all other dots have intensities exactly equal (i.e. $s = t \quad \forall s \in G, t \in o$ where $s \neq g, t \neq o$), then the SADGE formula simplifies to $\log(|g - o|)$. By taking the derivative

$\frac{\partial}{\partial o}\log(|g-o|) = \frac{1}{\ln(10)|g-o|}$, we can notice that the value of SADGE will increase by a positive value $\frac{1}{\ln(10)|g-\mathrm{d}o|}$ for every infinitesimal change in the value of o ($\mathrm{d}o$).

We also present a visual comparison of denoising results for the DeRisi dataset in Figure 4.5 and we show the results for the same microarray image we presented earlier in Figure 1.3. While none of the methods were capable of completely filtering out the visible environmental noise speck, the sharpness of individual dots in the area is visibly improved. Since no clear ground truth for this image exists, we cannot with certainty quantify the success of denoising process using the SADGE metric.

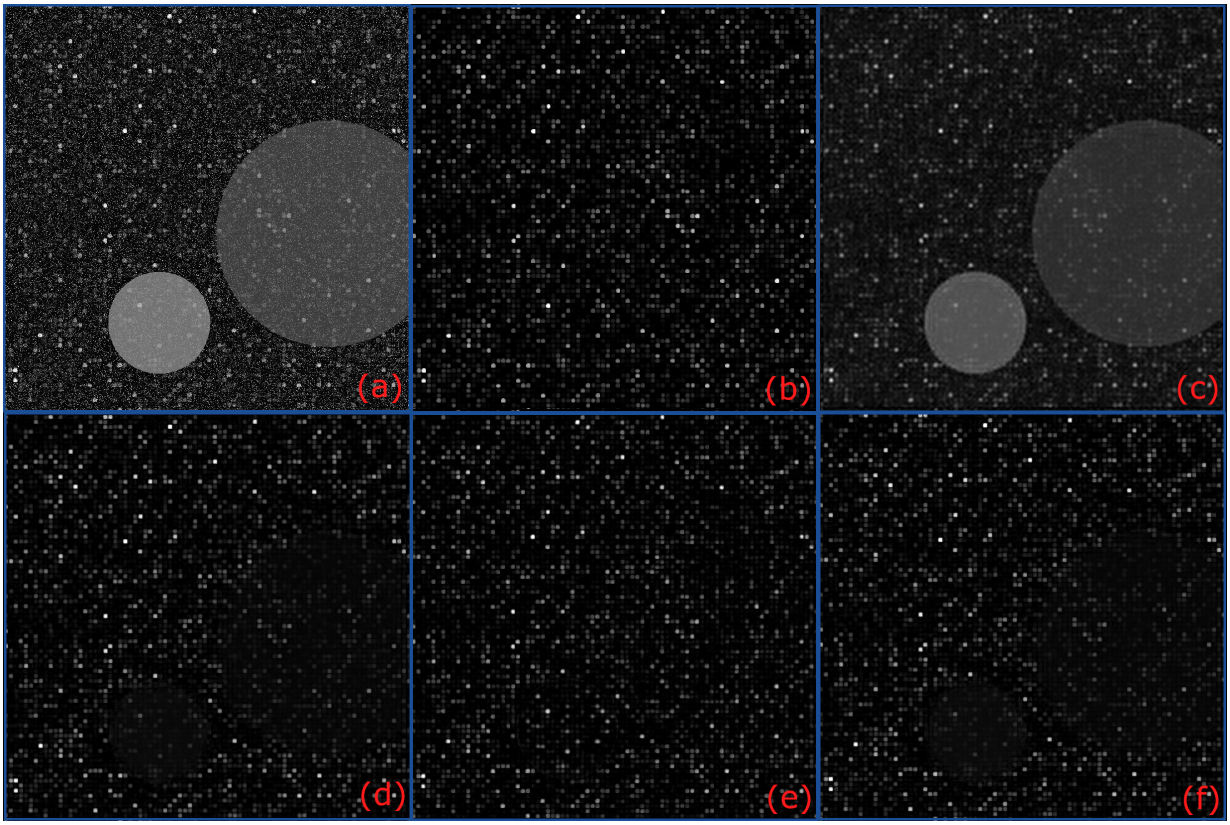


Figure 4.3: Visual comparison of the test-stage results from each denoising model considered in the experiment. (a) Noise-added image (I_η), (b) Ground-truth image (G), (c) Output image from the baseline autoencoder model, (d) Output image from the autoencoder trained on synthetic data, (e) Output image from the Restormer model, (f) Output image from the autoencoder model with EATME module and DEL activated.

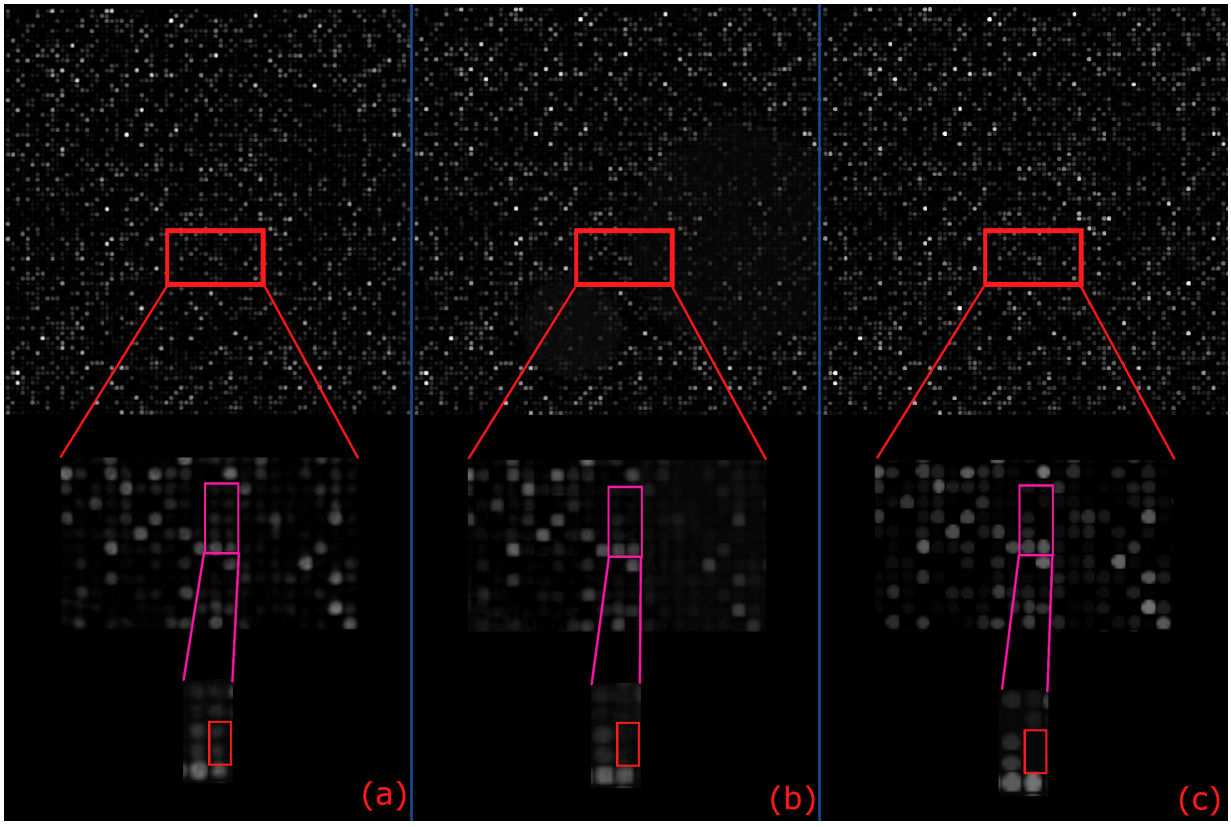


Figure 4.4: Restormer model tends to generate nonzero-intensity dots in locations where dot intensities should be zero; (a) Restormer output, (b) Autoencoder with EATME and DEL, (c) ground-truth.

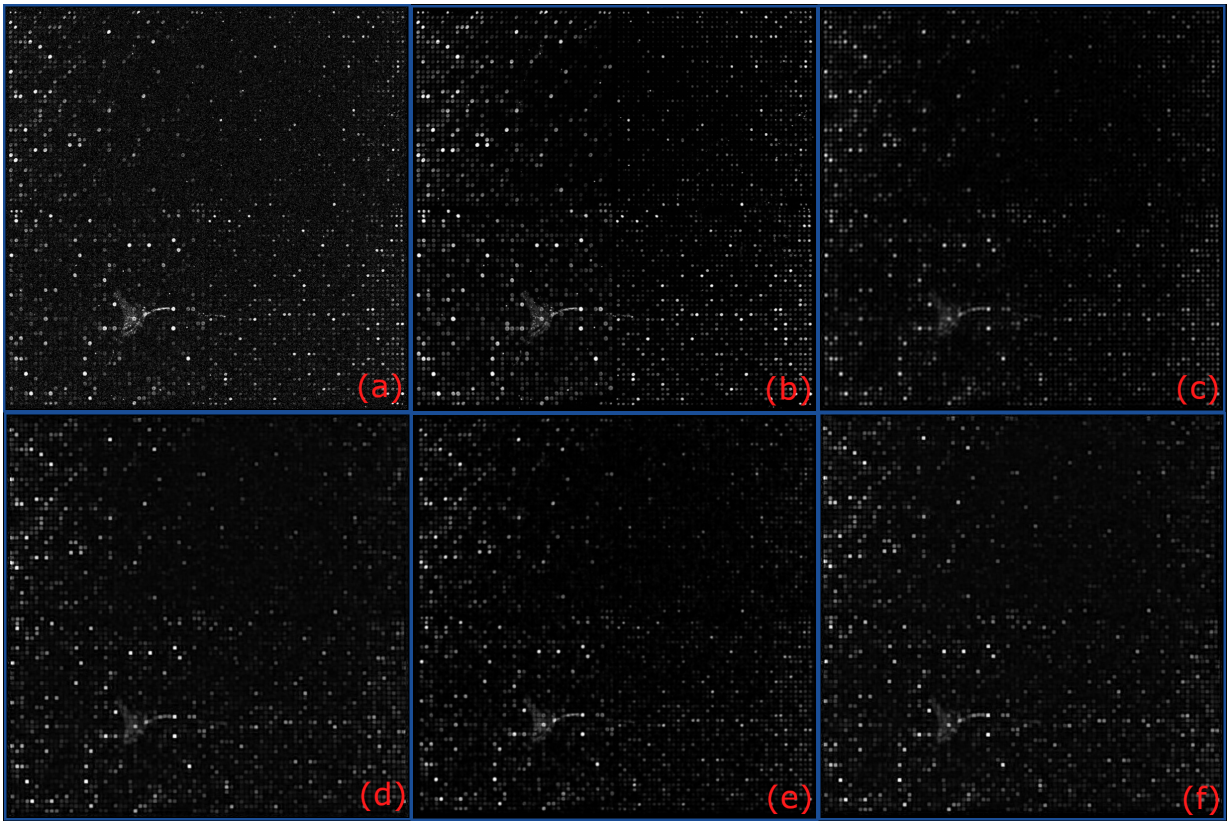


Figure 4.5: Visual comparison of results for different denoising methods for the DeRisi test set. (a) Noise-added image (I_η), (b) Original image from the DeRisi dataset, (c) Output image from the baseline autoencoder model, (d) Output image from the autoencoder trained on synthetic data, (e) Output image from the Restormer model, (f) Output image from the autoencoder model with EATME module and DEL activated.

Chapter 5

Conclusion

In this work we have presented a number of significant contributions addressing the microarray image denoising problem. We have shown that PSNR is not an optimal metric for the assessment of the denoising power of such models, since it fails to capture the semantic meaning of dot intensities in the image. We have proposed MaxPool Readout method as a lower-MAE dot intensity microarray readout method in comparison to what MAGIC Tool [28] offers. SADGE as an alternative metric which can be applied to synthetic microarray images and facilitates accurate evaluation of the denoising power of denoising models. We have shown that the Restormer model considered one of the state-of-the-art architectures in general image denoising problems performs exceptionally poorly when applied to microarray images. We have introduced two significant architectural modifications to the autoencoder model proposed by Mohandas *et al.* [21]: DEL and EATME. We show that our method outperforms any existing model for microarray image denoising by a large margin according to the SADGE metric.

5.1 Recommendations and considerations

For the microarray denoising problem, we recommend the f-PSNR metric to be supplanted by a metric exhibiting biological significance such as the SADGE metric proposed in our work. For training denoising models applicable to microarray images, we recommend the usage of fully-synthetic training datasets, which yield ground-truth images. This enables the usage of both classical PSNR and SADGE as criteria for successful denoising.

5.2 Future work

We recognize that methods and metrics we have proposed might be generalized further to other image processing subdomains, where pixel intensities within local regions have a precise semantic interpretation and the entirety of the image is susceptible to noise. In the future we would like to develop a generalized denoising framework for the field of Quantitative Immunocytochemistry [36], where microscopy slides from specific tissues are stained using fluorescent markers binding to specific structural regions within the tissue. This permits, for example, the analysis of accumulation of specific proteins in specific parts of the analyzed tissue, which might prove clinically-relevant in post-operative or post-biopsy histopathological tests.

References

[1] T. Cokelaer, A. Kravchenko, lahdjirayhan, msat59, V. Ferrari, A. Varma, B. L, C. E. Stringari, C. Brueffer, E. Broda, E. Pruesse, K. Singaravelan, S. A. Russo, Z. Li, m. padgham, and negodfre, "Cokelaer/fitter: V1.7.1," Zenodo, Jun. 2024.

[2] S. D. C. for Life, "DNA microarray(4x) image," Mar. 2022.

[3] J. G. Hacia, J.-B. Fan, O. Ryder, L. Jin, K. Edgemon, G. Ghandour, R. A. Mayer, B. Sun, L. Hsie, C. M. Robbins, L. C. Brody, D. Wang, E. S. Lander, R. Lipshutz, S. P. A. Fodor, and F. S. Collins, "Determination of ancestral alleles for human single-nucleotide polymorphisms using high-density oligonucleotide arrays," *Nature Genetics*, vol. 22, no. 2, pp. 164–167, 1999. [Online]. Available: https://www.nature.com/articles/ng0699_164

[4] S. Hamels, S. Leimanis, M. Mazzara, G. Bellocchi, N. Foti, W. Moens, J. Remacle, and D. E. G. Van. Microarray Method for the Screening of EU Approved GMOs by Identification of their Genetic Elements. JRC Publications Repository. [Online]. Available: https://publications.jrc.ec.europa.eu/repository/handle/JRC40406

[5] T. M. Gorges, A. Kuske, K. Röck, O. Mauermann, V. Müller, S. Peine, K. Verpoort, V. Novosadova, M. Kubista, S. Riethdorf, and K. Pantel, "Accession of Tumor Heterogeneity by Multiplex Transcriptome Profiling of Single Circulating Tumor Cells," *Clinical Chemistry*, vol. 62, no. 11, pp. 1504–1515, 2016. [Online]. Available: https://doi.org/10.1373/clinchem.2016.260299

[6] R. Ji, Y. Wang, Y. Cheng, M. Zhang, H.-B. Zhang, L. Zhu, J. Fang, and K. Zhu-Salzman, "Transcriptome Analysis of Green Peach Aphid (Myzus persicae): Insight into Developmental Regulation and Inter-Species Divergence," *Frontiers in Plant Science*, vol. 7, 2016. [Online]. Available: https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2016.01562/full

[7] H. Gogas, A. M. M. Eggermont, A. Hauschild, P. Hersey, P. Mohr, D. Schadendorf, A. Spatz, and R. Dummer, "Biomarkers in melanoma," *Annals of Oncology*, vol. 20, pp. vi8–vi13, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923753419570107

[8] H. Varmus, "Reverse Transcription," *Scientific American*, vol. 257, no. 3, pp. 56–65, 1987. [Online]. Available: https://www.jstor.org/stable/24979478

[9] M. Camacho-Sanchez, P. Burraco, I. Gomez-Mestre, and J. A. Leonard, "Preservation of RNA and DNA from mammal samples under field conditions," *Molecular Ecology Resources*, vol. 13, no. 4, pp. 663–673, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/1755-0998.12108

[10] D. Murphy, "Gene expression studies using microarrays: Principles, problems, and prospects," *Advances in Physiology Education*, vol. 26, no. 4, pp. 256–270, 2002. [Online]. Available: https://journals.physiology.org/doi/full/10.1152/advan.00043.2002

[11] K.-H. Kim, S.-Y. Lee, S. Kim, and S.-G. Jeong, "DNA microarray scanner with a DVD pick-up head," *Current Applied Physics*, vol. 8, no. 6, pp. 687–691, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1567173907001204

[12] Amersham-Biosciences. The microarray handbook. [Online]. Available: https://www.ualberta.ca/biological-sciences/media-library/mbsu/microarray/amersham-microarray-handbook.pdf

[13] C. L. Schoch, S. Ciufo, M. Domrachev, C. L. Hotton, S. Kannan, R. Khovanskaya, D. Leipe, R. Mcveigh, K. O'Neill, B. Robbertse, S. Sharma, V. Soussov, J. P. Sullivan, L. Sun, S. Turner, and I. Karsch-Mizrachi, "NCBI Taxonomy: A comprehensive update on curation, resources and tools," *Database*, vol. 2020, p. baaa062, 2020. [Online]. Available: https://academic.oup.com/database/article/doi/10.1093/database/baaa062/5881509

[14] N. Rachinger, S. Fischer, I. Böhme, L. Linck-Paulus, S. Kuphal, M. Kappelmann-Fenzl, and A. K. Bosserhoff, "Loss of Gene Information: Discrepancies between RNA Sequencing, cDNA Microarray, and qRT-PCR," *International Journal of Molecular Sciences*, vol. 22, no. 17, p. 9349, 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8430810/

[15] Pricing: Microarray and sequencing resource — boston university. [Online]. Available: https://www.bumc.bu.edu/microarray/pricing/

[16] Microarrays — Microarray analysis techniques and products. [Online]. Available: https://www.illumina.com/techniques/microarrays.html

[17] J. L. DeRisi, V. R. Iyer, and P. O. Brown, "Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale," *Science*, vol. 278, no. 5338, pp. 680–686, 1997. [Online]. Available: https://www.science.org/doi/10.1126/science.278.5338.680

[18] X. Wang, R. Istepanian, and Y. H. Song, "Microarray image enhancement by denoising using stationary wavelet transform," *IEEE Transactions on NanoBioscience*, vol. 2, no. 4, pp. 184–189, 2003. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1254520?casa_token=h5Geed78KmsAAAAA:Xo6iouwQK4pBpcAHh3kH3esrFFYivdPwFoXf-I1lamyZGkkr2Ol5mENjglVpLWu5AZ1F8tkxaQ

[19] Z. Gan, F. Zou, N. Zeng, B. Xiong, L. Liao, H. Li, X. Luo, and M. Du, "Wavelet Denoising Algorithm Based on NDOA Compressed Sensing for Fluorescence Image of Microarray," *IEEE Access*, vol. 7, pp. 13 338–13 346, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8606969

[20] L. Srinivasan, Y. Rakvongthai, and S. Oraintara, "Microarray Image Denoising Using Complex Gaussian Scale Mixtures of Complex Wavelets," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 4, pp. 1423–1430, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6800018/

[21] A. Mohandas, S. M. Joseph, and P. S. Sathidevi, "An Autoencoder based Technique for DNA Microarray Image Denoising," in *2020 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2020, pp. 1366–1371. [Online]. Available: https://ieeexplore.ieee.org/document/9182265/

[22] H. J. Nussbaumer, "The Fast Fourier Transform," in *Fast Fourier Transform and Convolution Algorithms*, H. J. Nussbaumer, Ed. Berlin, Heidelberg: Springer, 1982, pp. 80–111.

[23] L. V. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*. Pearson Education India, 2006.

[24] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6795963

[25] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling Laws for Neural Language Models. [Online]. Available: http://arxiv.org/abs/2001.08361

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[27] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang. Restormer: Efficient Transformer for High-Resolution Image Restoration. [Online]. Available: http://arxiv.org/abs/2111.09881

[28] L. J. Heyer, D. Z. Moskowitz, J. A. Abele, P. Karnik, D. Choi, A. M. Campbell, E. E. Oldham, and B. K. Akin, "MAGIC Tool: Integrated microarray data analysis," *Bioinformatics*, vol. 21, no. 9, pp. 2114–2115, 2005. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bti247

[29] L. Gautier, L. Cope, B. M. Bolstad, and R. A. Irizarry, "Affy—analysis of Affymetrix GeneChip data at the probe level," *Bioinformatics*, vol. 20, no. 3, pp. 307–315, 2004. [Online]. Available: https://doi.org/10.1093/bioinformatics/btg405

[30] E. Novikov and E. Barillot, "Software package for automatic microarray image analysis (MAIA)," *Bioinformatics*, vol. 23, no. 5, pp. 639–640, 2007. [Online]. Available: https://doi.org/10.1093/bioinformatics/btl644

[31] A. Wolen, "Aaronwolen/mimager." [Online]. Available: https://github.com/aaronwolen/mimager

[32] Wayback Machine. [Online]. Available: https://wayback-api.archive.org/

[33] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[34] J. C. Zenklusen, J. S. Boehm, K. L. Ligon, and D. A. Tuveson, "dbGaP Study," https://www.ncbi.nlm.nih.gov/projects/gap/cgi-bin/study.cgi?study_id=phs001486.v4.p4, 2024.

[35] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. [Online]. Available: http://arxiv.org/abs/1505.04597

[36] T. M. Mayhew, "Quantitative immunocytochemistry at the ultrastructural level: A stereology-based approach to molecular nanomorphomics," *Cell and Tissue Research*, vol. 360, no. 1, pp. 43–59, Apr. 2015.

[37] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. [Online]. Available: https://pytorch.org/assets/pytorch2-2.pdf

APPENDICES

Appendix A

Implementation details

Since Mohandas *et al.* [21] did not publish their code, nor the model weights, the denoising autoencoder architecture has been replicated from the figure provided in their paper. All deep learning models used in this paper were implemented using the PyTorch framework (version 2.3.1) [37].

Training for all models has been conducted on a single NVIDIA RTX A6000 GPU (Ampere architecture).

A.1 Model training hyperparameters

Autoencoders have been trained using the ADAM optimizer for 15 epochs with the exception of the baseline autoencoder (trained for 4 epochs to prevent overfitting due to small dataset size). MSE loss has been used as the reconstruction loss for all models leveraging the autoencoder architecture. The initial learning rate was set to 10^{-3}. Batch size was set to 8.

Restormer was trained using the RADAM optimizer for 25 epochs and the cosine-annealing learning rate scheduler and L1 loss have been used. The initial learning rate was set to 10^{-5}. Batch size was set to 4.

Glossary

dot Also called a *spot*; a circular pattern corresponding to a single type of probe hybridizing to a complementary nucleic acid sequence on a microarray. i, 2

Microarray A microfluidic system for the general purpose of biological interaction analysis. i, 1

probe A nucleic acid sequence, DNA or RNA, complementary to target sequence in biological interaction tests. i, 1

transcription The natural process of genetic information encoded in a DNA molecule being translated into an RNA molecule. i, 1

Nomenclature

Blob Noise Noise consisting of semi-translucent patches emulating environmental sources of noise such as dust particles or mistakes in microarray slide preparation. i, iii, 22, 23

Dot Expression Loss (DEL) An additional loss term which calculates the MSE between ground-truth expression values and expression values measured from a denoised image, defined in Equation 3.10. i

Elementwise Attention-like Transform for Microarray Enhancement (EATME) Proposed cross-attention-like mechanism which conditions the denoising models during training to attend to ground-truth expression values, defined in Equation 3.8. i

f-PSNR Flipped PSNR, where the MSE is calculated between the noise-added input image and the denoised image. i, iii, 24, 44–46, 51

floating-point image representation In the context of this work, image representation where all pixel values are in the range $[0.0; 1.0]$. i, 23, 28

MaxPool Readout A microarray readout algorithm which relies on fixed-stride MaxPooling operation applied to a 2D image, informed by the parametric shape of the microarray, defined in Algorithm 1. i, iii, 29, 33, 45, 51

microarray readout A procedure/algorithm which reads dot intensities from the microarray image against a chosen reference image and returns expression values. i, 25, 29, 36

Standard Assessment of Denoising in Gene-chip Evaluation (SADGE) A metric for the assessment of the power of denoising models in the context of DNA microarray denoising, as defined in Equation 3.4. i