CHAPTER **1**

Denoising Scanning Tunneling Microscope Images Using Deep Learning

1.1 Motivation

Data acquisition with a Scanning Tunneling Microscope (STM) can be challenging, primarily because the tip is highly susceptible to changes as it scans across the surface. Even when the tip remains relatively stable, most images exhibit striped noise in random locations. For instance, in the study of $ZrTe_3$, nearly 5,000 images were collected over several years of scanning. However, among all these images, only a few dozen were suitable for analysis and are presented in this thesis.

During a deep learning course, I realized that the solution to this problem lies in computer vision. I learned about various deep learning methods for denoising signals. While many existing tools can denoise images, they are primarily designed for simpler types of noise found in conventional images. Through my research, I discovered that CNN Autoencoders [1] have been used for image denoising, particularly in the MNIST dataset. They have also been extensively used for denoising medical images [2][3][4]. This algorithm was initially implemented in TensorFlow and designed for small images with a specific type of noise. To make the model more suitable for STM applications, I adapted the algorithm to work in the PyTorch framework, enabling it to process larger images and datasets. More importantly, I designed a custom noise model that mimics the striped noise observed in STM images by applying randomly generated striped noise to clean images. The CNN autoencoder performed well in removing minor and moderate noise levels but struggled with severe noise. However, this is not necessarily a drawback—major noise often indicates that the STM tip has encountered a significant defect. In such cases, reconstructing the image and introducing artificial features would be undesirable, as STM image analysis relies heavily on preserving specific topographic details to study structural properties.

When reviewing previous work on denoising STM images, I found only one study that applied a similar technique to simulated graphene images [5]. Since then, a paper published on arXiv in July 2024 used an autoencoder to reconstruct STM images [6], and another paper published on arXiv in February 2025 [7] used a more advanced protocol (PDA-Net) framework for unsupervised image denoising. While their results remained average, this indicates a growing interest in applying machine learning methods to STM image processing. Although I am not an expert in deep learning, I consider myself an active user, and I strongly believe that the STM community can greatly benefit from advancements in computer vision.

In this section, I will go over the model step by step and present the results obtained on multiple images and samples.

1.2 Preprocessing

1.2.1 Data Selection

A total of 20 high-quality, clean STM images were selected as the base dataset.

1.2.2 Data Augmentation

To increase the dataset size and improve model generalization, various augmentation techniques were applied:

- Rotation within a range of -10° to $+10^\circ$.
- Horizontal and vertical flipping.
- Image inversion.
- Addition of Gaussian blur.
- Addition of salt-and-pepper noise.

These augmentations expanded the dataset to approximately 2,000 images.

1.2.3 Image Loading, Resizing, and Cropping

All images were resized and cropped to:

- Remove empty spaces introduced by rotation.

- Prevent the model from learning false structural features.

1.2.4 Normalization

Each grayscale image was normalized by dividing pixel intensity values by 255, ensuring all values fall within the range $[0, 1]$. This normalization improves model efficiency and stability, as it simplifies the weight and bias calculation, by decreasing the calculation load.

1.2.5 Noise Addition

To simulate the striped noise observed in STM images, a custom noise model was designed:

1. A zero-valued mask of the same size as the image was created.
2. The number of stripes, `num_stripes`, determined how many horizontal stripes were added.
3. Each stripe had:
    - A random starting row.
    - A random width.
4. An empty array was initialized to store the noisy images.
5. Each image was processed as follows:
    - A noise mask was generated.
    - Random noise values matching the image shape were generated.
    - Noise was applied selectively:
        - Where `mask` = 1, the original pixel was replaced with a random noise value.
        - Where `mask` = 0, the original pixel remained unchanged.

This preprocessing pipeline ensures that the training dataset closely resembles real STM images, allowing the denoising model to learn relevant features effectively.

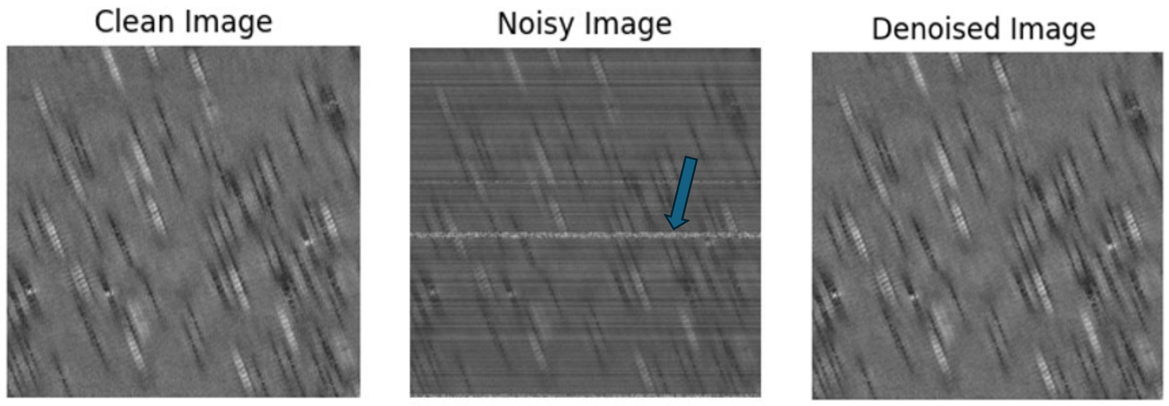


Figure 1.1: (Left) Clean $ZrTe_3$ topography taken at 90K. (Center) Topography with added horizontal noise, used as input for the CNN autoencoder. (Right) Output denoised topography.

1.2.6 Reshaping for Variational Autoencoder

Each grayscale image was reshaped to include a channel dimension, ensuring compatibility with the Variational Autoencoder (VAE) input format, which requires a shape of:

$$[\text{batch size}, \text{channels}, \text{height}, \text{width}] \tag{1.1}$$

Since the images are grayscale with a resolution of 630 pixels, a single channel was added, resulting in a final shape of:

$$[\text{batch size}, 1, 630, 630] \tag{1.2}$$

This adjustment ensures the data is correctly formatted for the VAE model.

1.3 Autoencoder Model

1.3.1 Encoder

The encoder compresses the input image into a lower-dimensional latent representation by progressively extracting features and reducing spatial dimensions.

- **First Convolutional Layer:**
  - A `Conv2D` layer with 32 filters and a kernel size of 3×3 is applied to extract low-level features.
  - The activation function used is ReLU, though other functions, such as sigmoid, were tested without noticeable improvement.

- **Downsampling via Max Pooling:**
  - A 2×2 max-pooling layer is applied to reduce the spatial dimensions by half.
  - This downsampling step preserves essential features while reducing computational complexity.
- **Second Convolutional Layer:**
  - Another `Conv2D` layer with 32 filters is applied to refine feature extraction.
  - Followed by a ReLU activation function.
- **Second Downsampling via Max Pooling:**
  - A second 2×2 max-pooling layer further reduces the spatial dimensions.
  - The result is a compressed latent representation of the image.
- **Feature Representation:**
  - This step is crucial as it ensures that the model learns key structural details of the image.
  - The encoding process is comparable to Fast Fourier Transform (FFT) filtering, which isolates key signals while discarding redundant information.
- **Model Complexity Considerations:**
  - Additional layers can be added to the encoder, but this increases computation time without necessarily improving the final results.
  - Excessive layers may even degrade the quality of the denoised images.

The implemented encoder structure is as follows:

```
self.encoder = nn.Sequential(
    nn.Conv2d(1, 32, kernel_size=3, padding=1),
    nn.ReLU(True),
    nn.MaxPool2d(2, 2),

    nn.Conv2d(32, 32, kernel_size=3, padding=1),
    nn.ReLU(True),
    nn.MaxPool2d(2, 2)
)
```

This encoder structure effectively reduces the input image dimensions while retaining essential features for reconstruction in the decoder.

1.3.2 Decoder

The decoder reconstructs the image from the compressed latent representation by progressively upsampling and refining the feature maps.

- **First Convolutional Layer:** A `Conv2D` layer with 32 filters is applied to refine feature extraction, followed by a ReLU activation function.
- **Upsampling Layer:**
  - Doubles the spatial dimensions, reversing the downsampling effect of max pooling.
  - Uses nearest-neighbor interpolation to expand the feature map.
- **Second Convolutional Layer:** Another `Conv2D` layer with 32 filters is applied, followed by a ReLU activation function.
- **Second Upsampling Layer:** Further restores the spatial resolution to match the original image size.
- **Final Convolutional Layer:** A final `Conv2D` layer with an output channel of 1 is applied to reconstruct the grayscale image.
- **Activation Function:** A sigmoid activation function is used at the output to ensure pixel intensity values remain within the range $[0, 1]$.

The implemented decoder structure is as follows:

```
self.decoder = nn.Sequential(
    nn.Conv2d(32, 32, kernel_size=3, padding=1),
    nn.ReLU(True),
    nn.Upsample(scale_factor=2, mode='nearest'),

    nn.Conv2d(32, 32, kernel_size=3, padding=1),
    nn.ReLU(True),
    nn.Upsample(scale_factor=2, mode='nearest'),

    nn.Conv2d(32, 1, kernel_size=3, padding=1),
    nn.Sigmoid()
)
```

This architecture ensures that the image is reconstructed while preserving key structural features learned during encoding. see the figure below for visualization of the encoder decoder process.

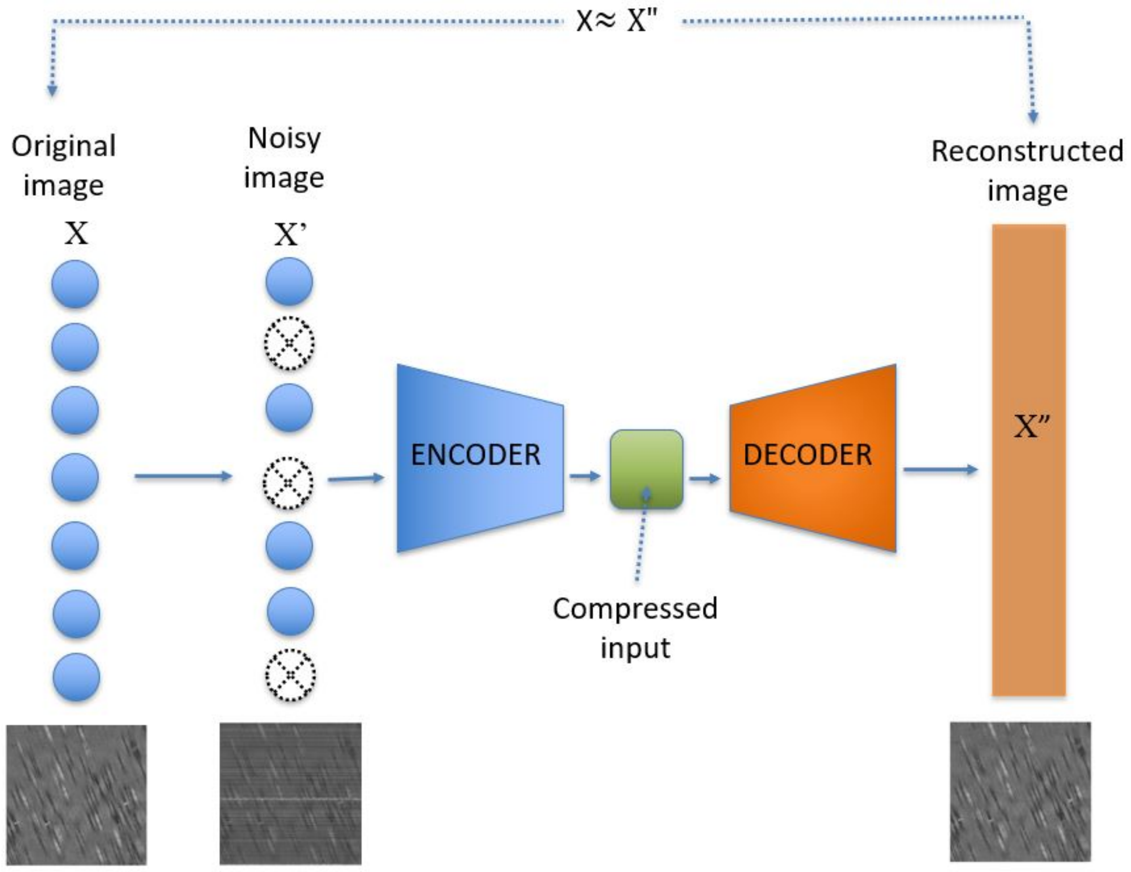
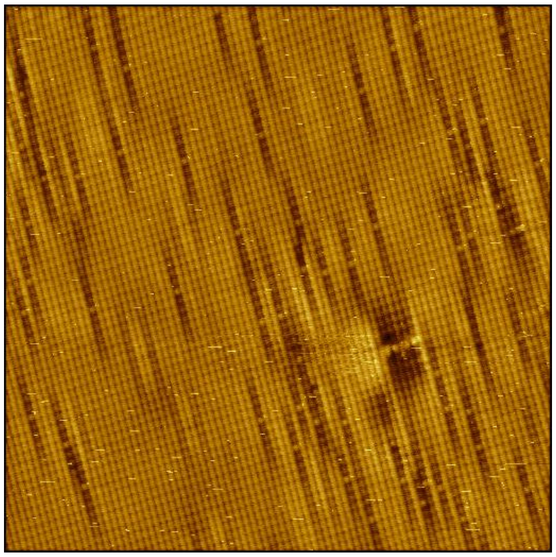


Figure 1.2: CNN autoencoder visual representation.

1.4 Results for $ZrTe_3$ Topographies

The first attempt to apply the model was naturally on the $ZrTe_3$ sample, since the neural network was trained on images of $ZrTe_3$ acquired at different temperatures and under various scanning conditions, while maintaining the same size and pixel resolution for each image.


(a) $ZrTe_3$ topography taken at 90 K,

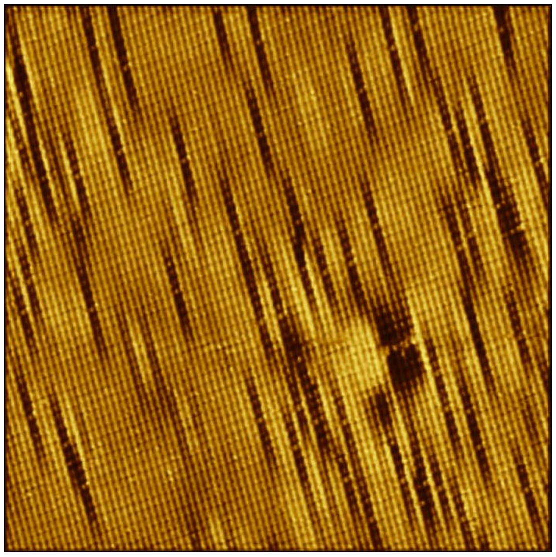
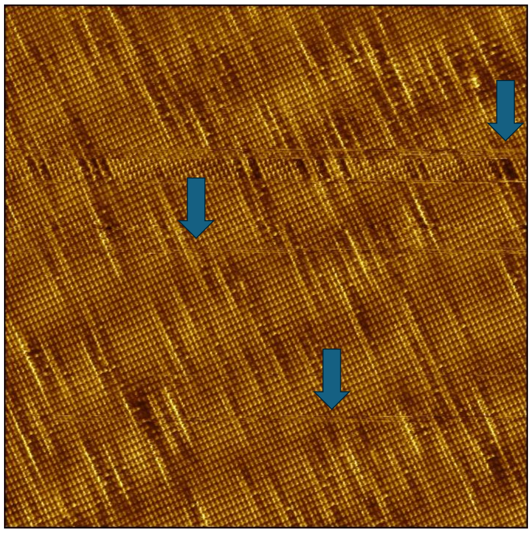
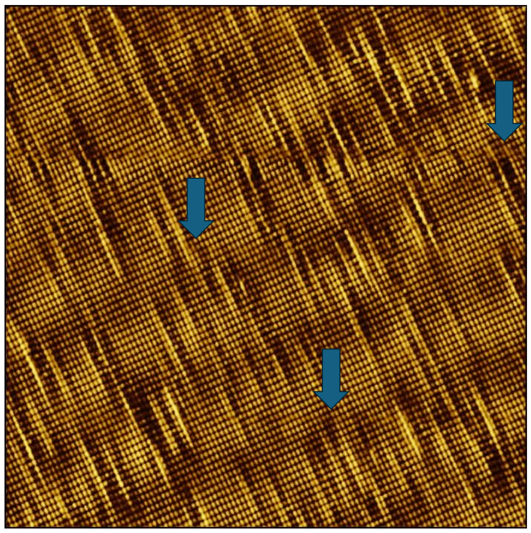

(b) Denoised $ZrTe_3$ topography.

Figure 1.3: Comparison of a noisy STM image of $ZrTe_3$-Hf dopped taken at 90k (a) and its denoised version (b). The original image exhibits minor random small horizontal striped noise, while the denoised version shows a significant reduction in noise.

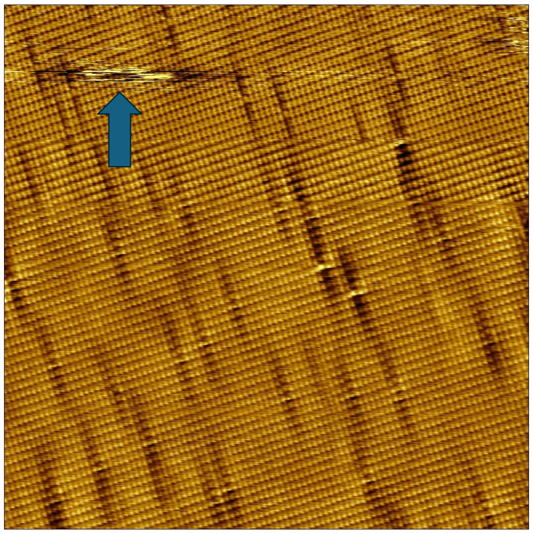


(a) $ZrTe_3$ topography taken at 9K, showing clear features disrupted by four horizontal noise artifacts.

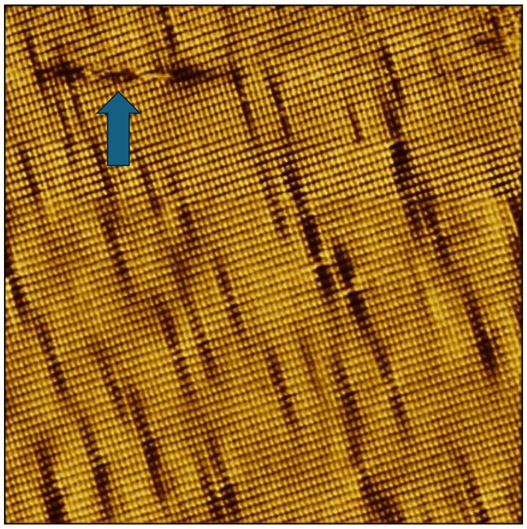


(b) Denoised topography. The second and third noise artifacts are completely removed, while the first noise remains noticeable.

Figure 1.4: Average noise.Comparison of a noisy STM image (a) and its denoised version (b). the blue arrows point to the striped noise location in (a) and same location denoised striped in (b)



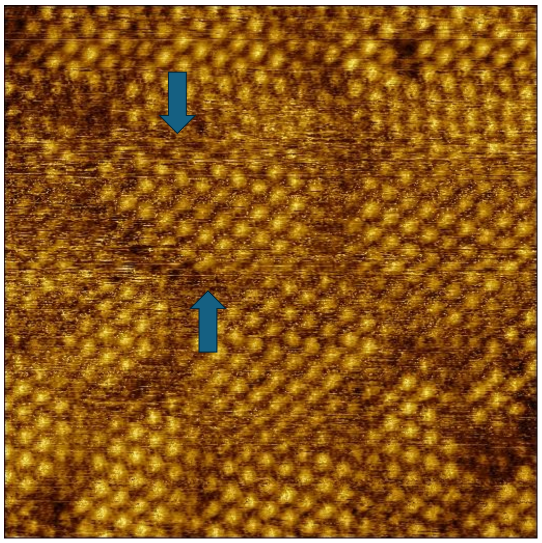
(a) $ZrTe_3$ topography taken at 90K, showing clear features disrupted by major horizontal noise artifacts, indicated by the blue arrow.



(b) Denoised topography. The minor noise artifacts are partially removed, while the primary noise remains.

Figure 1.5: Comparison of a noisy STM image (at) and its denoised version (b).

1.5 Result for Non-Hex Domains 1T-TaS$_2$ Topography


(a) Noisy 1T-TaS$_2$ topography showing non-hexagonal domains in the NC phase, with average striped noise indicated by blue arrows.

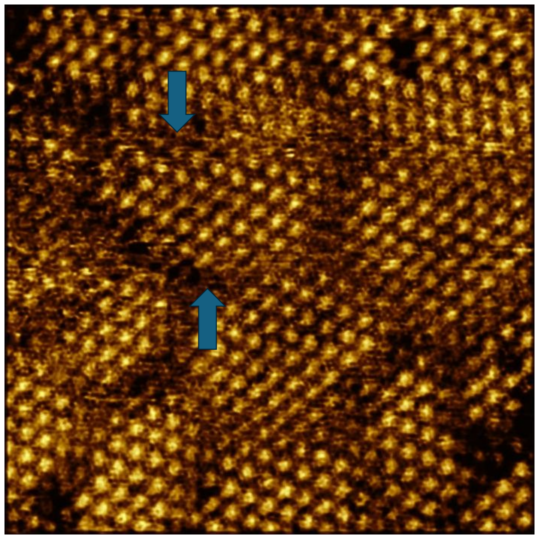
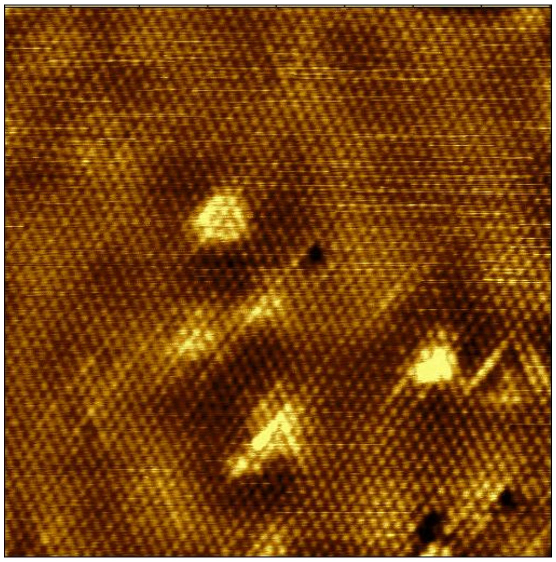

(b) Denoised topography. While some noise remains, it is significantly reduced.

Figure 1.6: Comparison of a noisy STM image (a) and its denoised version (b).

1.6 Result for $TiSe_2$ Topography


(a) $TiSe_2$ topography taken at 175K, showing low striped noise.

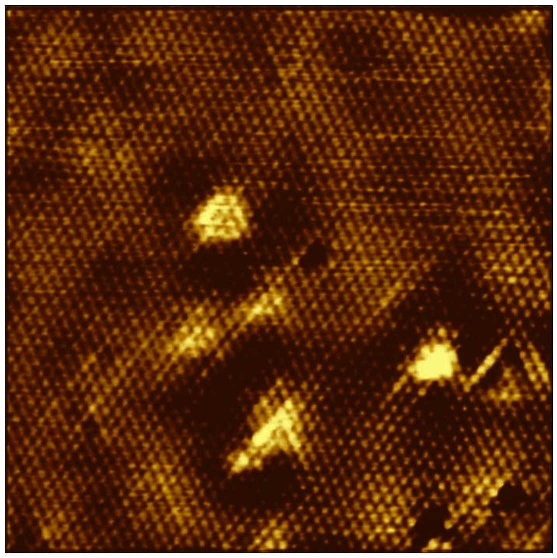

(b) Denoised topography. The horizental striped noise is almost completely removed.

Figure 1.7: Comparison of a noisy STM image (a) and its denoised version (b).

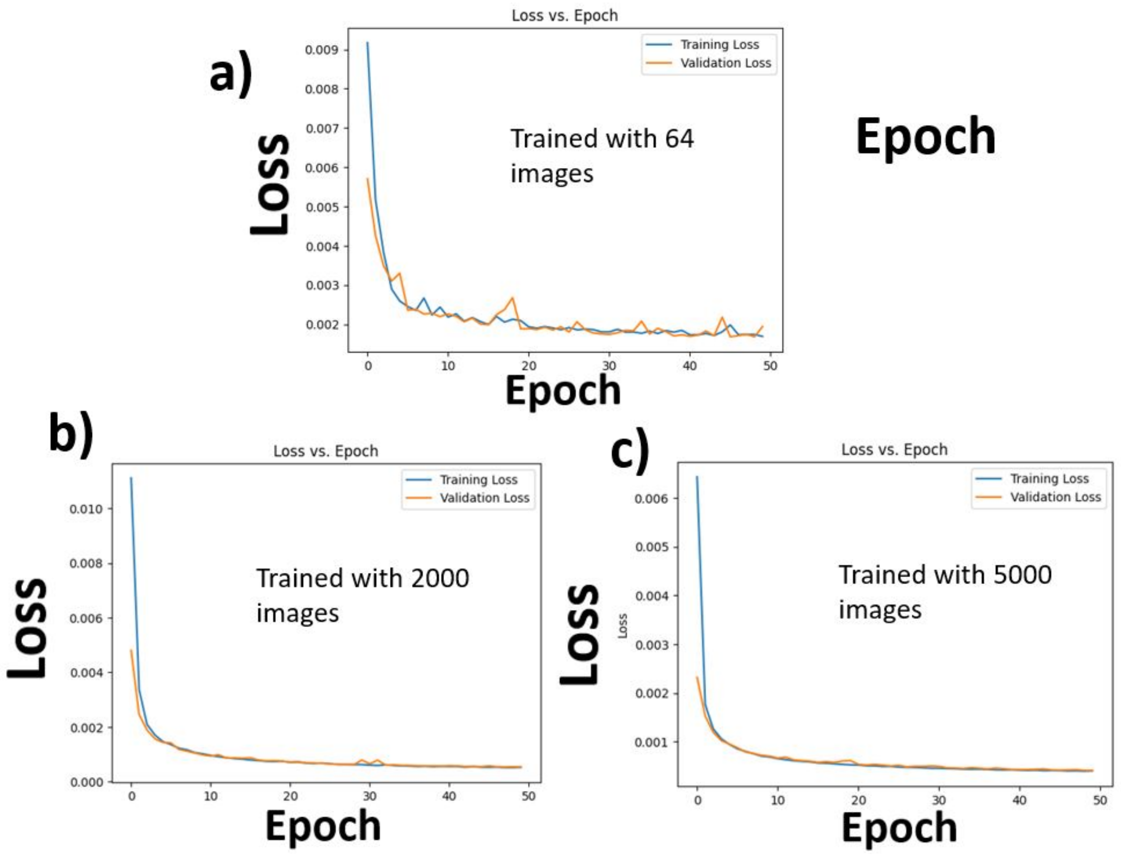


Figure 1.8: Loss vs. Epoch. (a) The plot for the model trained on 64 images shows overfitting, indicating that 64 images are insufficient. (b) and (c) correspond to training sets of 2000 and 5000 images, respectively, where no overfitting is observed. No further improvement is seen when increasing the dataset to 5000 images.

Defect identification after denoising

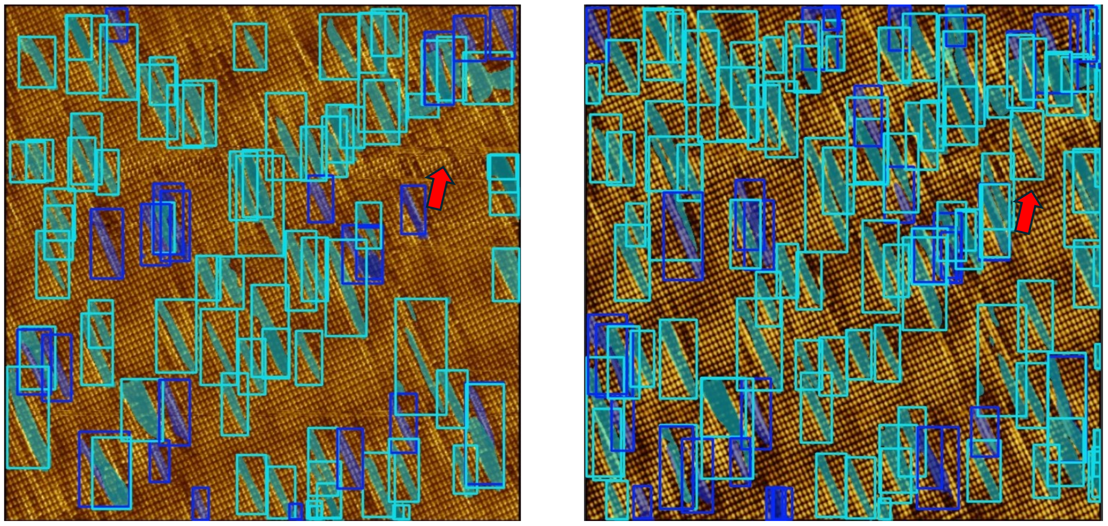


Figure 1.9: Defect segmentation using CNN YOLOv8 before and after denoising with a CNN autoencoder. (Left) Noisy topography shown in Figure 1.4a, with a dark defect near the horizontal noise indicated by a red arrow. The detected defect count is 22 dark defects and 73 bright defects. (Right) Defect segmentation after applying the denoising model. The dark defect is shown in red detection, with a new defect count of 32 bright defects and 98 dark defects.

1.7 Conclusion and Future Work

In summary, the adapted autoencoder algorithm with custom noise showed promising results, not only on the trained $ZrTe_3$ sample images but also on two other materials with different crystal structures, sizes, and pixel resolutions.

This algorithm is not only useful for visualization but also highly beneficial for analysis. For example, in Chapter 3, I demonstrated three different techniques to extract defect signals. These techniques can only be effectively applied if the image is completely free of noise, which is extremely rare. By minimizing noise, we can significantly improve defect signal extraction using these three methods.

Further improvements can be made if the noise is better characterized and higher-quality images are used for training. Additionally, expanding the training set with more diverse samples would enhance the model's accuracy. A potential future improvement involves incorporating a Swin Transformer[8][9] instead of a CNN autoencode algorithm, a state-of-the-art innovation in image denoising within the field of computer vision.

References

[1] Aswin Techguy. Deep cnn autoencoder - denoising image. Available online, 2022. GitHub repository.

[2] B. Nageshwar Rao and D. Lakshmi Sreenivasa Reddy. Brain mri noise reduction using convolutional autoencoder. In Ashwani Kumar, editor, *Artificial Intelligence and Data Science*, pages 348–362, Cham, 2022. Springer Nature Switzerland.

[3] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 241–246, 2016.

[4] Wenxi Lin, Mengting Gao, Chengtao Ruan, and Jianhua Zhong. Denoising for intracranial hemorrhage images using autoencoder based on cnn. In *2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*, pages 520–523, 2021.

[5] Frédéric Joucken, John L. Davenport, Zhehao Ge, Eberth A. Quezada-Lopez, Takashi Taniguchi, Kenji Watanabe, Jairo Velasco, Jérôme Lagoute, and Robert A. Kaindl. Denoising scanning tunneling microscopy images of graphene with supervised machine learning. *Phys. Rev. Mater.*, 6:123802, Dec 2022.

[6] Peter Binev, Joshua Moorehead, Ayush Parambath, Luke Parrella, Rori Pumphrey, and Miruna Savu. Stm image analysis using autoencoders, 2025.

[7] Jianxin Xie, Wonhee Ko, Rui-Xing Zhang, and Bing Yao. Physics-augmented deep learning with adversarial domain adaptation: Applications to stm image denoising, 2025.

[8] Chi-Mao Fan, Tsung-Jung Liu, and Kuan-Hsien Liu. Sunet: Swin transformer unet for image denoising. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, page 2333–2337. IEEE, May 2022.

[9] Jingyun Liang. Swinir: Image restoration using swin transformer. https://github.com/JingyunLiang/SwinIR, 2021.