MEDICAL IMAGE SUPER RESOLUTION

Ayushi Agrawal
Student# 1010015145
ayushi.agrawal@mail.utoronto.ca

Saaim Raad
Student# 1010292853
saaim.raad@mail.utoronto.ca

Pakhi Gupta
Student# 1010475807
pakhi.gupta@mail.utoronto.ca

Md Nazmus Saad
Student# 1010243194
md.saad@mail.utoronto.ca

ABSTRACT

This project proposes a latent diffusion-based super resolution (SR) architecture to enhance low resolution chest X-ray images, aiming to improve the diagnostic accuracy in medical settings. The model integrates a pretrained Variational Autoencoder (VAE), a custom U-Net-lite denoiser, and a noise scheduler to perform diffusion in the latent space, reconstructing high-resolution images with improved structural fidelity and computational efficiency.

Trained on a subset of the NIH ChestX-ray14 dataset. The model is set to be optimized for edge deployment, with plans to implement it on hardware platforms like the NVIDIA Jetson Nano. For now, inference and quantization (to INT8 via TensorRT) are conducted in a simulated environment. The model is benchmarked against bicubic baselines using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), offering a scalable solution for medical image enhancement in resource-limited settings.

—-Total Pages: 10

1 BRIEF PROJECT DESCRIPTION

High-quality medical images are essential for accurate diagnosis and treatment, particularly for detecting subtle abnormalities in chest X-rays, such as early signs of heart and lung infections, tumors, or cancer. However, imaging devices often face limitations due to hardware capabilities, environmental conditions, and operational constraints, resulting in low-resolution (LR) images that can obscure critical details. This reduction in image quality poses significant challenges for both human experts and automated Computer-Aided Diagnosis (CAD) systems, potentially compromising clinical decisions (Umirzakova et al., 2023).

Our project aims to develop a Super-Resolution (SR) system based on a latent diffusion architecture that efficiently reconstructs high-resolution images from low-resolution inputs. By combining a pretrained Variational Autoencoder (VAE) with a denoising U-Net and a diffusion noise scheduler, our approach enables high-quality image enhancement while being optimized for deployment on edge devices like the NVIDIA Jetson Nano (Figure 1).

Deep learning is fit for this project as this task does not require interpretability. It will also be able to recover finer details that are missed by traditional methods like interpolation-based, reconstruction-based and learning-based. Also, our project leverages a large dataset of chest X-ray images, and neural architectures excel at learning complex patterns from extensive data. The use of diffusion models, paired with latent space encoders, allows for a flexible and powerful way to upscale LR images. This makes our solution both practically useful and scalable for real-world medical settings, where improving image resolution can directly impact patient diagnosis.

2 INDIVIDUAL CONTRIBUTIONS AND RESPONSIBILITIES

Overall, the project is progressing on schedule. Team communication is smooth and members are fulfilling their responsibilities on time.

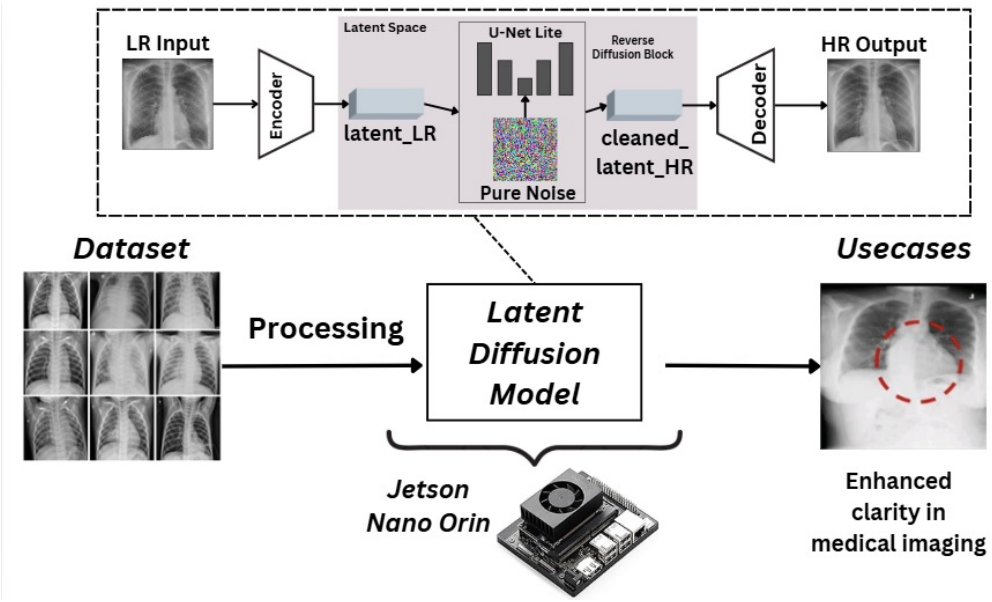

Figure 1: The proposed model.

2.1 Collaboration

The team uses a variety of different platforms for efficient communication and organization, as outlined in Table 1. These collaboration strategies remain largely unchanged from those laid out in the Project Proposal.

Weekly meetings are held on Thursdays at 6:00 PM in the Bahen building on campus and Mondays at 9:00 PM over Google Meet. Meetings had originally been scheduled for Saturdays at 12 PM, but this proved to be an inconvenient time and was adjusted accordingly.

Table 1: Platforms used for team collaboration and project execution.

Platform	**Purpose**	**Details of Use**
Discord	Main communication	Daily updates; meeting scheduling; link sharing.
Google Meet	Secondary communication	Virtual meetings.
Google Drive	File hosting	Shared document editing; resource hub.
GitHub	Code hosting	Version control; collaborative code editing.
ClickUp	Project management	Shared Gantt chart; hub for deadlines; meeting schedules; task assignment.

2.2 Task Management

Each team member was assigned responsibility for two focus areas in the Project Proposal, as in Table 2. A detailed breakdown of completed and future tasks is in Table 3. Most tasks have been completed on time and some within a small margin of the due date. The team has not encountered any issues with teamwork or deadlines, and the project is on track to be completed in the time available.

The major remaining tasks are model performance improvement and edge deployment. Two team members are assigned to each task to provide redundancy in case of unexpected roadblocks.

Table 2: Team member responsibilities and specialization areas.

Member	Primary Focus	Secondary Focus
A: Ayushi Agrawal	Data Processing & Evaluation	Deployment testing, slide design
B: Saaim Raad	Model Architecture	Dataset integration, training & docs
C: Md Nazmus Saad	Diffusion & Deployment	Model training & evaluation metrics
D: Pakhi Gupta	Documentation & UI	Data pairing, augmentation, visualization

Table 3: Project task breakdown and responsible team members.

Member	Tasks	Due	Completed
A	Download data and extract to folder automatically; Create LR and HR pairs; Create PyTorch Dataset and Dataloaders using appropriate transformations	June 17	June 20
D	Add data augmentation functionality to Dataset; Implement probabilistic random horizontal flip, rotations, resized crop and cutout consistently for LR and HR	June 22	June 23
C	Build U-Net-Lite architecture; Implement diffusion scheduler	July 5	July 7
B	Implement VAE encoder/decoder; Train initial model; Implement diffusion scheduler	July 9	July 9
Upcoming Tasks			
B, C	Additional hyperparameter tuning; Export model to ONNX	July 24	
C, B	Use TensorRT to quantize model and benchmark latency on Jetson	Aug 7	
A, C	Build an interface to show before/after on edge device	Aug 11	
D, A	Test model using evaluation metrics; Create final report and presentation	Aug 14	

3 Data Processing

The dataset used is the publicly available NIH ChestX-ray14 dataset (Wang et al., 2017) consisting of 112,120 1024x1024 images of chest X-rays from 30,805 unique patients.

The Indiana University Chest X-Ray Collection of more than 7000 images accessible through Open-i (Demner-Fushman et al., 2016) will be used as new testing data. It contains both lateral and frontal view X-rays, as opposed to the NIH set which contains only frontal X-rays. This will ensure that the model can be tested on images completely different from its training data.

3.1 Data Loading

The following steps were used to obtain and process the dataset, of which a few sample images are shown in Figure 2:

1. **Data downloading and organization:** A downloading script allows the user to download a specified number of batches from the dataset source, extracts them into the specified folder, and prevents overwriting of existing data.
2. **Generating HR and LR image pairs:** The `generate_downsampled_pairs` function downsamples the original 1024x1024 images twice using OpenCV's `resize` function with bicubic interpolation. The 256x256 downsampled image is saved into the `HR_256` folder and the 64x64 version is saved into the `LR_64` folder.
3. **Creating a custom PyTorch dataset:** The `PairDataset` class reads images from their folders, transforms them into tensors of type `float32` and normalizes them to [-1,1] for faster training convergence. It assembles mini-batches of dimension $B\times3\times64\times64$ or $B\times3\times256\times256$ for input images and labels (ground truths) respectively.
4. **Dataset splitting:** The `PairDataset` class also allows for configurable splits between train/validation/test data. A 70-20-10 split is currently in use, which provides 38,527 training, 10,907 validation, and 5,500 test images. One of the challenges here was to keep images from the same patient together, which was solved using careful parsing of filename while generating split indices.

The same steps will be followed for testing on the new data (Indiana University). The only required change will be to set the data directory (a configurable argument) to the directory where the new data has been extracted, and to ensure images are cropped to squares before resizing.

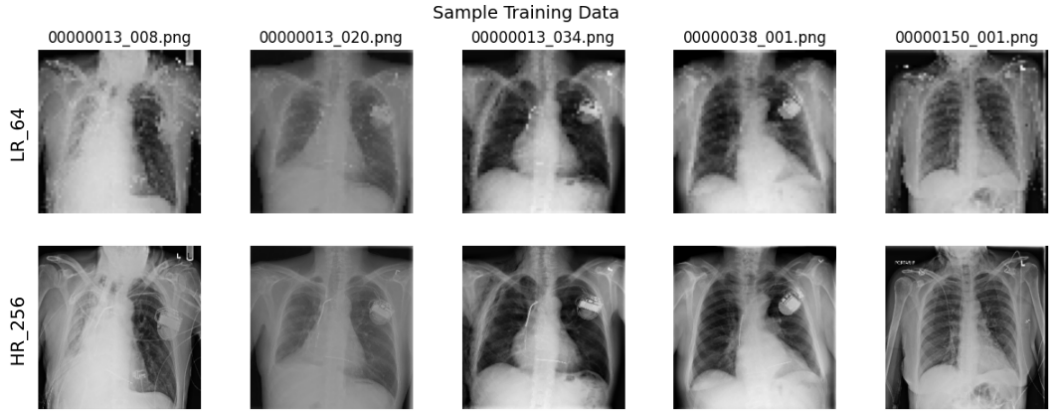

Figure 2: Some random samples of the training data without data augmentation.

3.2 Data Augmentation

To improve generalization and robustness, data augmentation was applied to 70% of the training samples. This percentage was chosen based on empirical studies suggesting that moderate augmentation helps prevent overfitting while maintaining data fidelity, which is particularly important in the medical imaging domain where fine-grained details are critical (Athalye & Arnaout, 2023; Sanaat et al., 2022). Each augmentation was implemented considering spatial alignment, ensuring the LR and HR image pairs remained synchronized. The augmentation techniques used are as follows:

- **Random Horizontal Flip (50% chance):** Both the LR and HR images were flipped along the vertical axis using `torchvision.transforms.functional.hflip`. This augmentation helps the model become invariant to left-right orientation, which is useful for X-ray images where anatomical symmetry is common and directional bias can lead to misclassification.

- **Random Rotation (90°, 180°, or 270°):** Implemented using `F.rotate`, this augmentation rotates both images in discrete steps. These rotations simulate different acquisition angles, which can occur in real-world imaging scenarios due to variations in patient posture or X-ray machine positioning. Using discrete angles helps preserve anatomical integrity while still adding valuable diversity.
- **Random Resized Crop (80% of original size):** A random crop of 80% of the image was selected and then resized back to the original dimensions. The same crop coordinates were applied to both LR and HR images. This helps the model learn robust features across various subregions of the image, which is essential in X-rays where abnormalities can appear anywhere.
- **Cutout Augmentation (30% chance):** A square patch (covering 20% of the image size) was blacked out at a random location using direct tensor masking. This simulates occlusions or artifacts, such as medical markers or foreign objects, often present in X-rays. Training with cutout encourages the model to rely on surrounding context, improving resilience to incomplete data.

As shown in Figure 3, these augmentation techniques provide diverse and medically plausible variations during training.

While implementing these techniques, a few challenges were faced. Since the low- and high-resolution image pairs had different sizes, keeping them aligned during transformations like cropping meant the coordinates had to be scaled carefully. It was also important to avoid any unrealistic augmentations like color jitter and warping, which were left out to ensure the medical value of the X-rays was not compromised. Choosing to apply augmentation to around 70% of the data helped strike a good balance between adding useful variation and still keeping enough original examples for the model to learn from accurately.

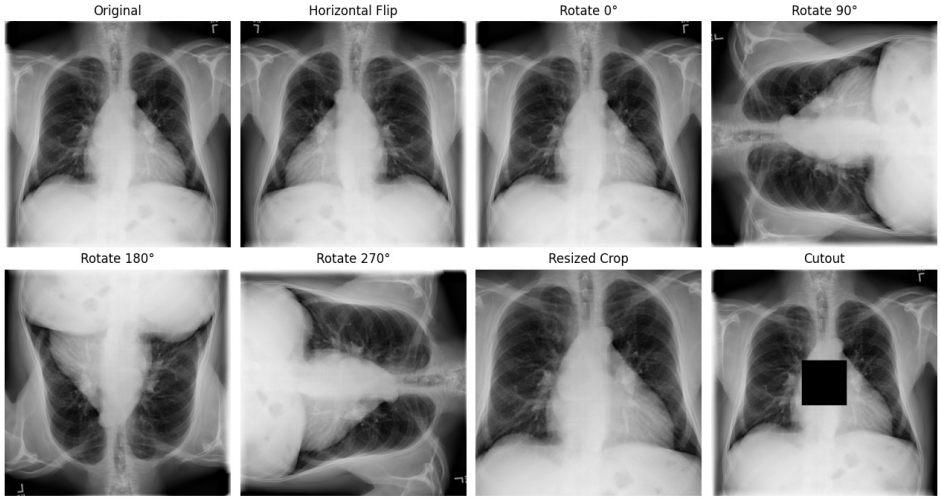

Figure 3: Demonstration of augmentation techniques used in model training.

4 Baseline Model

The baseline model that will be used to compare with our primary model is bicubic upsampling. First, the LR image was taken as input and resized to an HR image via bicubic upsampling. After that, PSNR, which quantifies the reconstruction quality based on pixel-wise errors and SSIM, which assesses perceptual quality by comparing luminance, contrast, and structural information, were computed (Figure 4). The bicubic sampling is performed via PyTorch's `F.interpolate` function.

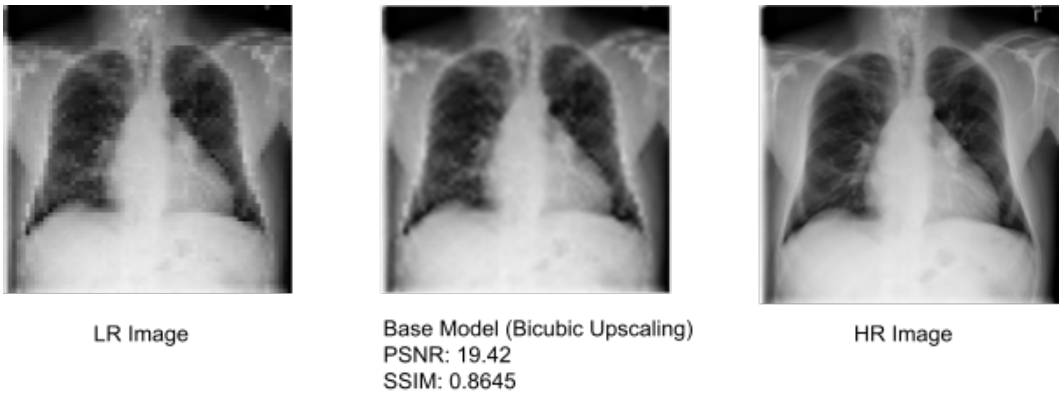

Figure 4: One example showing the baseline models' improvements over the LR image and the computed PSNR/SSIM.

5 Primary Model

The primary model performs super-resolution on low-resolution chest X-rays using a latent diffusion-based architecture. As shown in Figure 5, the system includes a pretrained VAE, custom diffusion schedulers, and a custom U-Net, totaling 813,220 trainable parameters. All training was conducted on an NVIDIA GeForce RTX 4090.

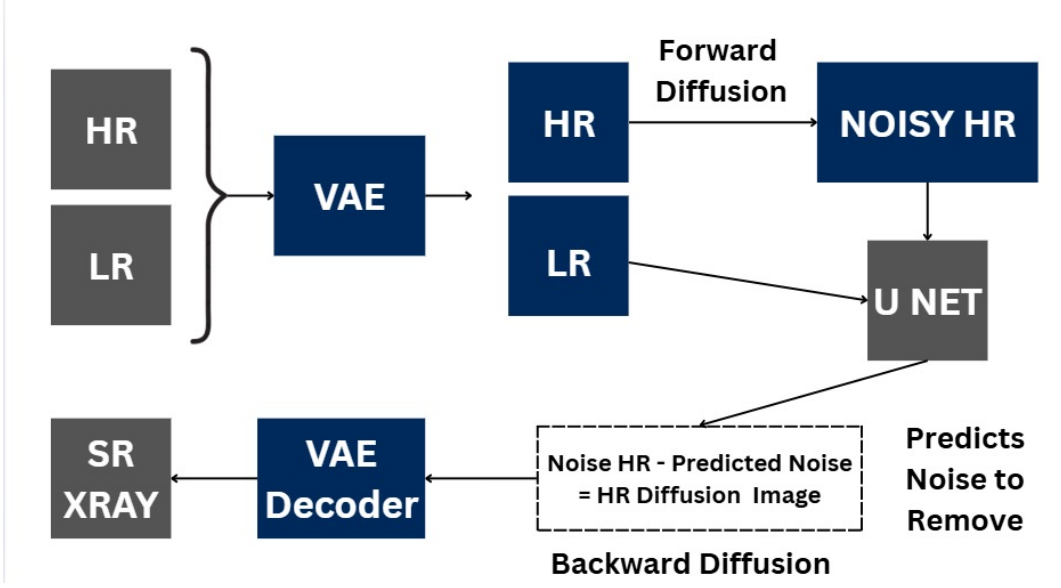

Figure 5: High level block diagram of architecture used for both training and inference.

5.1 Model Architecture and Design Choices

5.1.1 Latent Diffusion-Based Super-Resolution

The model takes paired LR and HR images (3×64×64 and 3×256×256). LR images are first upsampled to 3×256×256 via bicubic interpolation, then both LR and HR images are encoded to latent space separately using Hugging Face's pretrained AutoencoderKL VAE, yielding 4×32×32 latents by reducing spatial resolution 8× and expanding channels from 3 to 4.

5.1.2 Forward Diffusion and Timestep Conditioning

Training uses the forward diffusion process, where Gaussian noise is progressively added to the HR latent over multiple time steps. Rather than sequentially applying noise at each step, we use the closed-form DDPM formula for efficiency:

$$z_t = \sqrt{\bar{\alpha}_t} \cdot z_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$$

This closed form calculates the final noisy HR latent given some number of timesteps $t \in [0, 300)$. The timestep t controls the noise level, as a higher t implies more steps that add noise. z_0 is the clean HR latent, $\epsilon \sim \mathcal{N}(0, 1)$, and

$$\bar{\alpha}_t = \prod_{i=1}^{t} (1 - \beta_i)$$

with β linearly scheduled from 0.0001 to 0.02, as per the original DDPM paper (Ho et al., 2020). The noisy HR latent is concatenated with the LR latent to form an 8-channel tensor passed to the U-Net, alongside the timestep t.

U-Net uses an encoder-decoder structure with skip connections (Figure 6) (Ronneberger et al., 2015). Through conditioning available from the LR latent and t, the U-Net learns to predict the noise added to the HR latent, trained using MSE loss against the true ϵ added during the forward diffusion process.

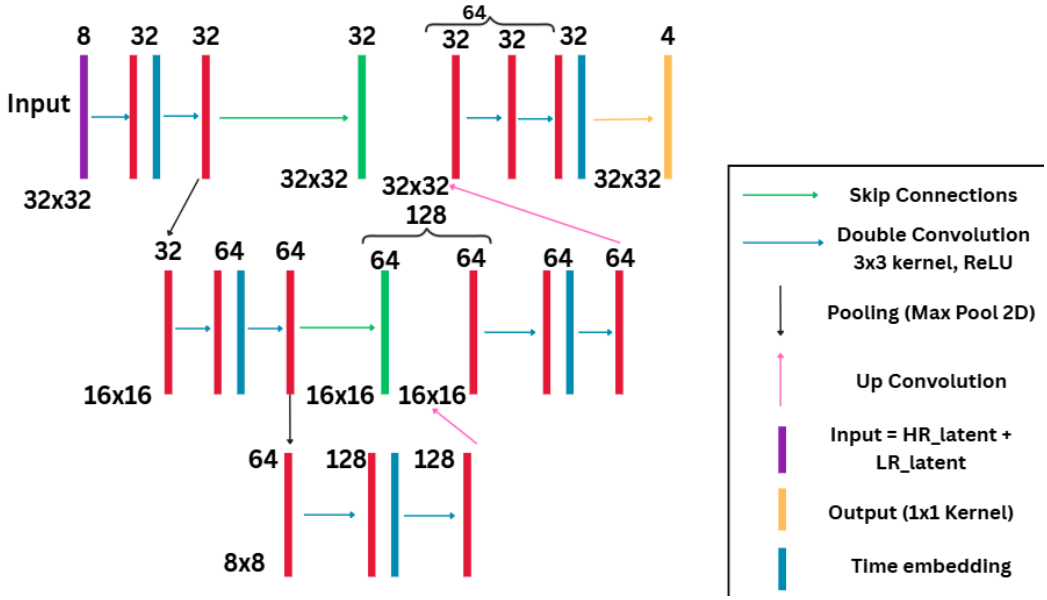

Figure 6: Structure of the U-Net.

5.1.3 Inference and Reverse Diffusion

During inference, the process starts from pure Gaussian noise in latent space ($4 \times 32 \times 32$). Reverse diffusion is performed from $t = 299$ to 0, iteratively denoising with the trained U-Net. At each step, the LR latent is concatenated with the noisy HR latent, and the predicted $\hat{\epsilon}_t$ is used to reverse the noise. After all steps, the final latent is decoded via the VAE to produce a $3 \times 256 \times 256$ super-resolved image. PSNR is used to measure performance against ground-truth HR images, with bicubic interpolation as a baseline.

5.2 Model Performance

5.2.1 Verifying Training Function

To validate the implementation, the model was overfitted on a 10-image subset. After 500 epochs, the loss decreased to 0.144 but did not reach zero (Figure 7), as diffusion models aim to predict

stochastic noise rather than exact targets. The minimum MSE is lower-bounded by noise variance, VAE reconstruction error, and the KL-divergence term. Visually, the diffusion output surpassed bicubic upsampling (Figure 8), confirming correct training behavior.

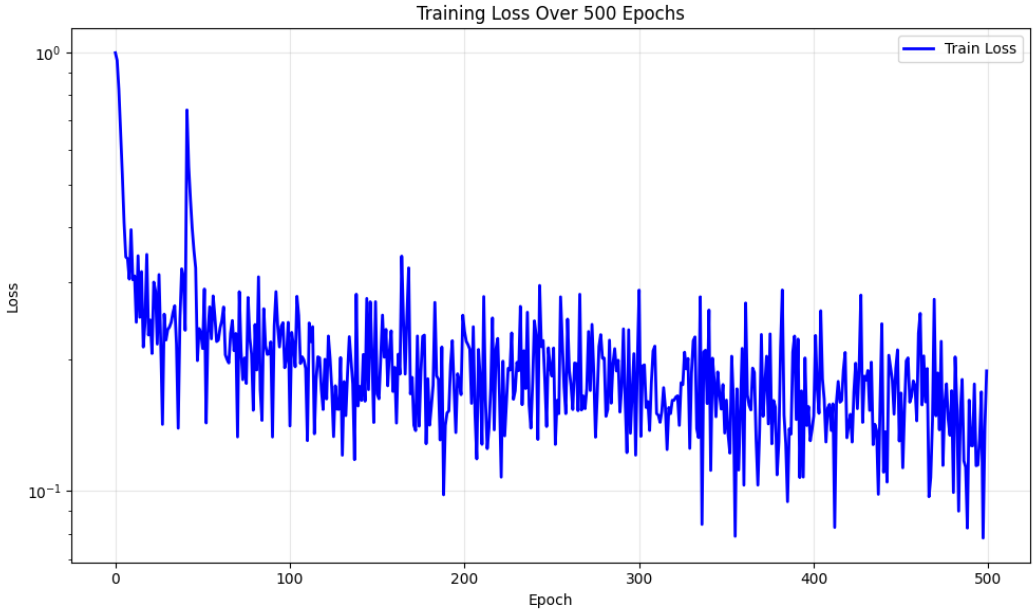

Figure 7: Training loss graph during overfitting verification.

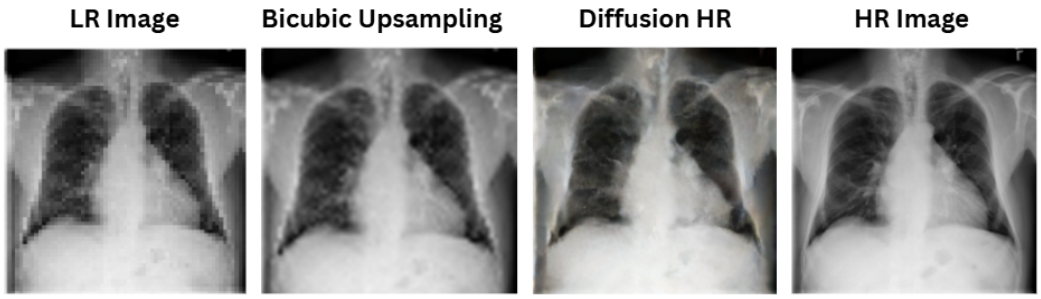

Figure 8: Visual example of overfitting result, showing more clarity than the baseline.

5.2.2 Model Version 1 Performance

For this first full experiment, the dataset was split 70% train, 20% validation, and 10% test. Training ran for 50 epochs, but only six of the twelve available data batches were loaded, so each epoch saw half the images (effective batch size = 9 632). Both training and validation losses dropped quickly at the start and then flattened out near 0.187 by roughly epoch 20 (Figure 9).

Although final outputs were blurrier than bicubic results, the model consistently generated lung-like structures from pure noise, indicating it had learned meaningful denoising patterns (Figure 10). Poor results were likely due to untuned hyperparameters, reduced data per epoch, a short training window, and limited U-Net capacity. Future experiments will include the full dataset, longer training, a deeper U-Net, and hyperparameter optimization (including β-scheduling) to improve output quality.

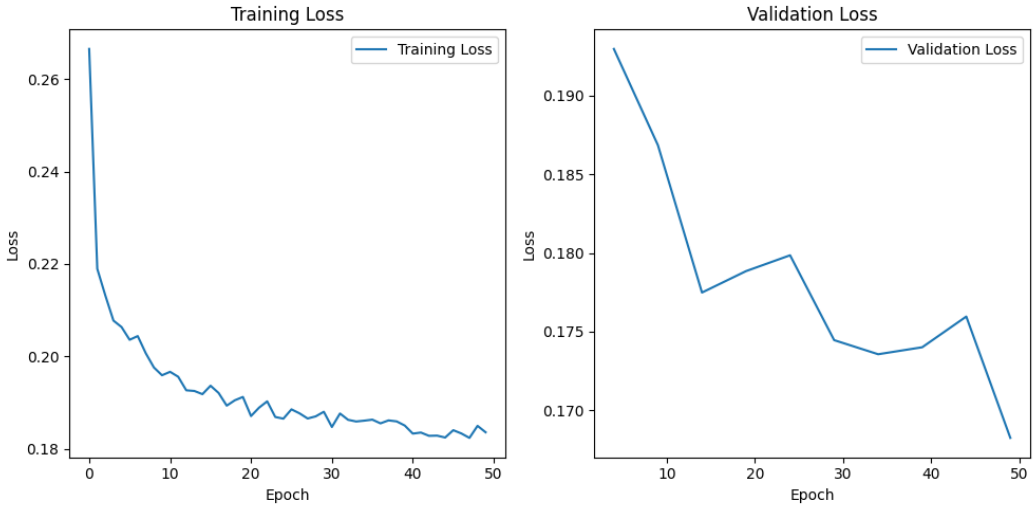

Figure 9: Training and validation curves for Version 1 of training.

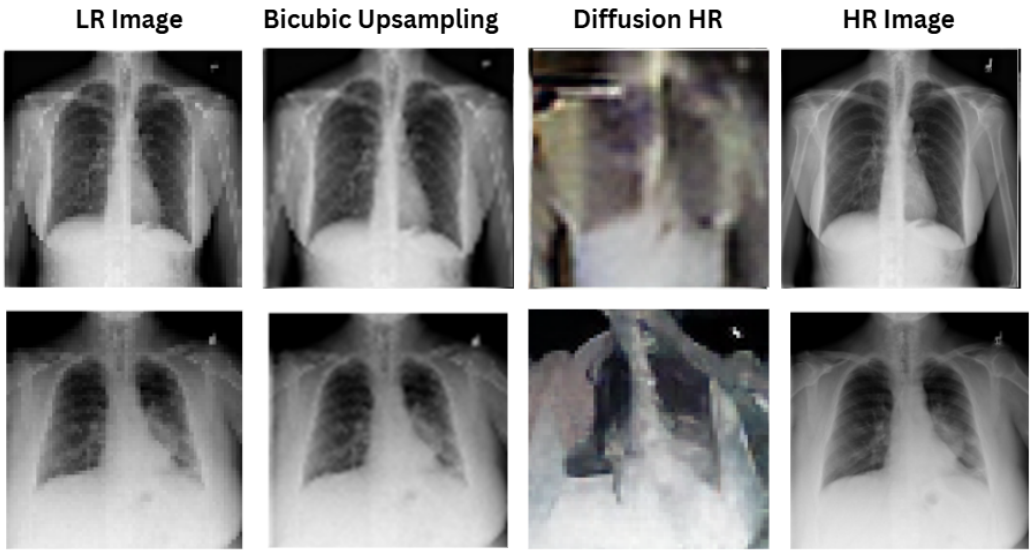

Figure 10: Visual example of training result.

5.3 Challenges

Key challenges involved designing and integrating the diffusion scheduler with the VAE and U-Net. As a latent diffusion model, the system required a solid understanding of discrete-time stochastic processes and variational inference to generate images from pure noise. A significant portion of time was spent reading relevant literature and working through the underlying math to ensure correct implementation. As a result, only one full training run was completed (approximately 6 hours), leaving inadequate time for hyperparameter tuning. Although metrics such as PSNR, SSIM, and test loss are straightforward to compute for evaluating diffusion model outputs on the test set, they were not included due to time constraints but will be incorporated shortly for a more complete evaluation, and in order to better compare bicubic upsampling with latent diffusion.

REFERENCES

Chinmayee Athalye and Rima Arnaout. Domain-guided data augmentation for deep learning on medical imaging. *PloS one*, 18(3):e0282532, 2023. ISSN 1932-6203. doi: 10.1371/journal.pone.0282532. URL https://europepmc.org/articles/PMC10035842.

Dina Demner-Fushman, Marc D. Kohli, Martin B. Rosenman, Sarah E. Shooshan, Laritza Rodriguez, Sameer Antani, George R. Thoma, and Clement J. McDonald. Preparing a collection of radiology examinations for distribution and retrieval. *Journal of the American Medical Informatics Association: JAMIA*, 23(2):304–310, March 2016. doi: 10.1093/jamia/ocv080.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pp. 234–241. Springer, 2015. URL http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a. (available on arXiv:1505.04597 [cs.CV]).

Amir Sanaat, Iraj Shiri, Somayeh Ferdowsi, et al. Robust-deep: A method for increasing brain imaging datasets to improve deep learning models' performance and robustness. *Journal of Digital Imaging*, 35(2):469–481, 2022. doi: 10.1007/s10278-021-00536-0. URL https://doi.org/10.1007/s10278-021-00536-0.

Sabina Umirzakova, Sevara Mardieva, Shakhnoza Muksimova, Shabir Ahmad, and Taegkeun Whangbo. Enhancing the super-resolution of medical images: Introducing the deep residual feature distillation channel attention network for optimized performance and efficiency. *Bioengineering*, 10(11):1332, 2023. doi: 10.3390/bioengineering10111332.

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3462–3471, 2017. doi: 10.1109/CVPR.2017.369.