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 UNet-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. If time and performance evaluation permits, the model is 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 INTRODUCTION

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.

Deep learning is fit for this project as this task does not require interpretability. It will also be able to recover finer details that traditional methods like interpolation-based, reconstruction-based, and learning-based miss. Since our project leverages a large dataset of chest X-ray images, and neural architectures excel at learning complex patterns from extensive data, it deems a good fit for the task. 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.

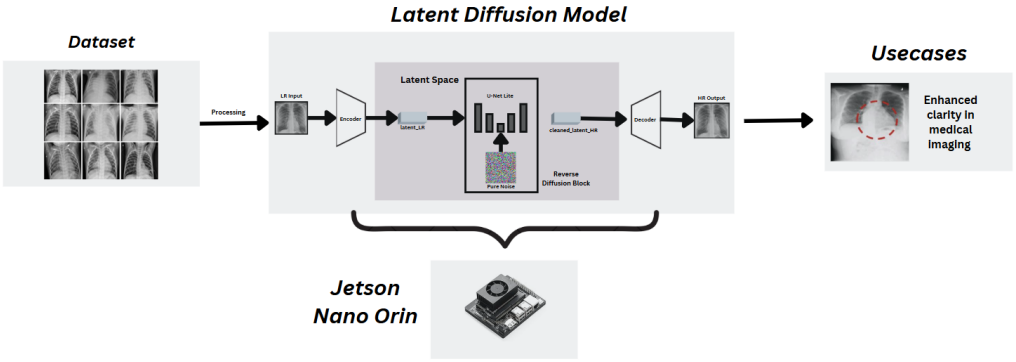

Figure 1: The proposed model.

2 Background And Related Work

2.1 A Brief History of Super Resolution (SR)

Super-resolution (SR) is the task of reconstructing high-resolution (HR) images from low-resolution (LR) counterparts. Early SR methods predominantly involved traditional interpolation techniques like bicubic interpolation, in which weighted averages of surrounding pixels are used to approximate unknown pixel values. It is a general purpose resampling technique used for both upsampling and downsampling. Although computationally efficient, this lacks the ability to capture fine detail and results tend to introduce blur. This is confirmed by the Data Processing Inequality which roughly states that you can not create information that was not present (Cover & Thomas, 2012). As a result, there have been recent shifts to exploring Deep Neural Network based frameworks as a means to inject learned information from the Neural Networks, into low-resolution images in order to increase resolution in a manner that actually adds informative content.

2.2 Evaluation Metrics in Super-Resolution

Traditional SR approaches typically evaluate quality using metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) (Huynh-Thu & Ghanbari, 2008; Wang et al., 2003). PSNR quantifies the reconstruction quality based on pixel-wise errors. It is calculated as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right) \tag{1}$$

where

MAX	Maximum possible pixel value (typically 255 for 8-bit images)
MSE	Mean squared error between the original and reconstructed images

SSIM assesses perceptual quality by comparing luminance, contrast, and structure.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2}$$

where

μ_x, μ_y	Mean intensity of images x and y
σ_x^2, σ_y^2	Variance of x and y

σ_{xy} Covariance between x and y
C_1, C_2 Constants to stabilize the division when the denominators are small

PSNR is widely used in papers as it is simple and fast. SSIM, by contrast, tends to favor structural similarity and captures visual structure better than PSNR (Wang et al., 2003).

2.3 CNN-Based Super-Resolution

Some of the earliest SR involved the use of CNN based architectures. Dong et al. (2015)'s pioneering work, SRCNN, demonstrated CNNs' superiority over traditional methods by achieving substantial improvements in PSNR metrics. These CNN models utilized pixel-wise losses such as L1 and L2 loss, leading to overly smoothed outputs that were not as sharp.

2.4 GAN-Based Super-Resolution

In 2017, Generative Adversarial Networks (GANs) showed promising breakthroughs in SR. GANs are a framework that uses two networks in order to generate new data. A generator network generates synthetic data that looks like real data in order to fool a discriminator network who tries to distinguish between real and fake data. The goal is to train the generator to the point where it excels in synthetic data generation by becoming capable enough to fool even the discriminator. SRGAN by Ledig et al. (2017) significantly improved perceptual quality through adversarial training. This work was further extended by ESRGAN, enhancing GAN training stability and introducing deeper architectures (Wang et al., 2019) . Despite these successes, GAN-based approaches suffer from being notoriously difficult to train. By their very nature, they require delicate balancing between generator and discriminator networks to achieve stable convergence.

2.5 Pixel-Space UNets and their Limitations

The U-Net architecture, initially proposed by Ronneberger et al. (2015). for biomedical image segmentation, features an encoder-decoder structure. They excel at image-to-image tasks. Its uniqueness lies in the fact that it included skip connections, in which each encoder's block is saved and passed directly to the corresponding decoder block with the goal of preserving information from earlier layers while still capturing global context via the bottleneck. U-nets operate in pixel-space, and as a result, end up being memory-intensive and computationally heavy, making them unsuitable for deployment on edge devices and also slow to train.

2.6 Latent Diffusion Models

Latent Diffusion Models (LDMs), introduced by Rombach et al. (2022), offer computational efficiency by performing diffusion processes in a compressed latent space obtained via a Variational Autoencoder (VAE). Diffusion models iteratively add and remove noise to generate realistic outputs. The key advantage of LDMs lies in reduced computational cost. Latent spaces significantly decrease the input dimensions into the U-Net, thus reducing computation and memory requirements.

LDMs have been effectively employed in state-of-the-art image synthesis frameworks like Stable Diffusion, demonstrating high-quality image generation capabilities (Rombach et al., 2022). More recently, Edge-SD-SR by Noroozi et al. (2025) showcased latent diffusion-based SR's practicality on mobile and edge devices, dramatically reducing model size and inference latency while preserving quality. Edge-SD-SR achieved impressive results with models as small as 169M parameters and inference times under 40ms on smartphones, highlighting the potential for real-time applications.

3 Data Processing

The NIH ChestX-ray14 dataset that will be used to train this model is a public medical imaging dataset consisting of 112,120 chest X-ray images (1024 × 1024 pixels) from 30,805 unique patients. To manage storage and training time, three image archive batches (approximately 30,000 images) will be selected. These images will be filtered for PA (posteroanterior) view X-rays because they

are the clinical standard and offer minimal distortion compared to AP views. Corrupted and non-standard images will also be discarded. Approximately 10,000–15,000 high-resolution X-rays will remain for training, validation, and testing.

The following preprocessing methods will be used by the team as needed:

1. **Generating HR and LR Image Pairs:** Due to memory limitations and hardware constraints, the original 1024^2 images will be resized to a resolution of 256 × 256 pixels using bicubic interpolation (OpenCV's `INTER_CUBIC`). These resized images will serve as the ground-truth high-resolution (HR) set. A further downsampled copy of each 256^2 image, also using bicubic interpolation, will be generated at 64 × 64 pixels to simulate low-resolution (LR) inputs for the super-resolution task. Matching LR and HR images will be stored as pairs with identical filenames in two separate folders (`HR_256` and `LR_64`) for ease of access.
2. **Data Loading and Tensor Conversion:** After disk pre-processing, each LR-HR image pair will be loaded. PNG images will first be loaded from disk into NumPy arrays as 8-bit integers with pixel values ranging from 0 to 255. These arrays will be converted into PyTorch tensors of datatype float32 and normalized by dividing by 255.0, scaling pixel values into the range [0.0, 1.0]. We will further standardize these tensors by shifting and scaling them to a range of [-1, 1], which is standard in latent-diffusion-based models to accelerate training convergence.
3. **Data Augmentation:** To improve model generalization, identical random horizontal flips, ± 15° rotations, and slight brightness jitter will be applied consistently to each LR-HR image pair using a shared pipeline from `torchvision.transforms`. This approach ensures both images in a pair maintain perfect alignment after augmentation.
4. **Mini-Batch Assembly:** Finally, a PyTorch DataLoader will assemble mini-batches consisting of tensors shaped `(B × 1 × 64 × 64)` for LR images and `(B × 1 × 256 × 256)` for HR images, feeding them directly into the diffusion-based super-resolution model during training and evaluation.
5. **Dataset Splitting:** The processed dataset will be divided into training, validation, and testing subsets with a split of 70% training, 20% validation, and 10% testing.

4 ARCHITECTURE

The project aims to perform SR via a latent diffusion based architecture. It has critical simplifications to enable edge deployment and efficient training. The system will consist of a pretrained Variational Autoencoder, a custom denoising U-Net lite which will be implemented, and a diffusion noise scheduler. The VAE and diffusion noise scheduler will be borrowed from Hugging Face open source library.

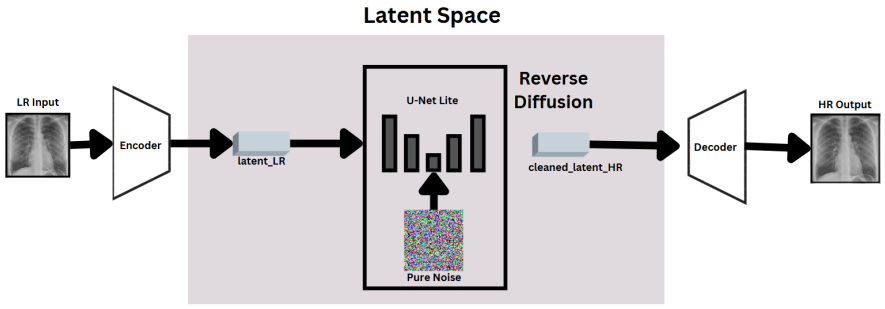

Figure 2: Proposed model architecture.

4.1 Training Pipeline

During training, each high-resolution (HR) and low-resolution (LR) image pair is first encoded by a frozen Variational Autoencoder (VAE) encoder into latent representations (`latent_HR` and `latent_LR`). The HR latent then undergoes forward diffusion, in which Gaussian noise of intensity t (referred to commonly as the timestep), is added according to a scheduler in a stochastic manner, yielding a noisy latent (`noisy_latent_HR`). The custom UNet-Lite model takes this noisy latent, along with the LR latent and timestep, to predict the exact noise that was added. The training objective is to minimize the mean squared error (MSE) between the predicted noise and the true added noise, allowing the UNet-Lite model to learn effective denoising patterns at various noise levels.

4.2 Inference Pipeline

At inference (deployment on Jetson Nano), only the LR image is available, which is first encoded by the frozen VAE encoder into `latent_LR`. The inference process starts from pure Gaussian noise (`latent_N`), which is iteratively denoised through a reverse diffusion loop using the trained UNet-Lite model. At each reverse diffusion step, the model predicts and removes noise, conditioned on `latent_LR`, gradually refining the latent representation. After completing 20-30 reverse diffusion steps, the final clean latent is decoded by the frozen VAE decoder into the high-resolution, super-resolved output image.

4.3 Edge AI Deployment

After training, the custom UNet-Lite model and VAE decoder will be exported from PyTorch into ONNX format and quantized to INT8 precision using NVIDIA's TensorRT, significantly reducing inference latency and memory usage. The optimized INT8 models will be deployed on a Jetson Nano development kit. During inference on the Jetson device, a lightweight Python interface will accept 64×64 LR images, run the entire latent diffusion super-resolution pipeline, and display the generated 256×256 super-resolved outputs along with inference times.

5 Baseline Model

The model's outputs will be compared to a baseline such as bicubic upsampling, where the LR image will be taken as input and resized to a HR image via bicubic. Then, PSNR/SSIM will be computed against ground truth and a baseline number will be set up. Ideally the model should exceed or at least match the baseline number with better perceptual quality.

6 Ethical Considerations

While this project aims to improve diagnostic support by enhancing low-resolution chest X-ray images through latent diffusion, several ethical considerations and limitations must be addressed.

6.0.1 Data and Labeling Limitations

The dataset that will be used in this project consists of chest X-rays images with labels extracted from radiology reports via Natural Language Processing (NLP). Although the labelling method achieves over 90% accuracy, they are not manually verified by radiologists. This introduces labeling noise and some positive findings may actually be false or ambiguous in the image itself (Olatunji et al., 2019).

6.1 Bias and Generalizability

Public datasets, like the one being used for this project usually lack demographic diversity, which risks bias and reduced performance on underrepresented populations (Galanty et al., 2024).

6.2 Model and Practical Constraints

The diffusion-based architecture relies on generative modeling in a latent space, which may introduce hallucinated details not present in the original scan. While the UNet-Lite reduces complexity for edge deployment, it may limit the model's ability to capture subtle clinical features. Additionally, using a pretrained VAE not tailored to medical images may lead to domain mismatch.

6.3 Clinical Risks

Super-resolution can enhance clarity but may introduce artifacts or hallucinations. These enhancements must not be mistaken for real anatomical features, as they could mislead diagnoses if not carefully validated (Shin et al., 2024).

Overall, while promising, this system should be used with caution and further validated for safe, fair clinical use.

7 Project Plan

7.1 Communication

Communication primarily occurs through Discord for day-to-day coordination, while shared Google Drive is used to store and organize documents. GitHub manages code contributions, and ClickUp is used to plan tasks and track progress with Gantt charts. To avoid code conflicts, the team uses feature branches and pull requests in GitHub, ensuring changes are reviewed and merged systematically.

To maintain effective collaboration, the team has scheduled regular weekly meetings on Thursdays at 6:00 PM and Saturdays at 12:00 PM. While in-person attendance is preferred, team members who are unavailable can join virtually via Google Meet.

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

Platform	**Purpose**	**Description**
Discord	Main communication	Used for daily team communication, quick updates, and coordination.
Shared Google Drive	File storage & collaboration	Central location for organizing and sharing documents, slides, and resources.
GitHub	Code collaboration	Repository for version control, code sharing, and collaborative development.
Google Meet	Virtual meetings	Used for remote team meetings, discussions, and check-ins.
ClickUp	Scheduling and Project Management	To plan weekly meetings, set deadlines, assign tasks, and coordinate teamwork sessions using Gantt charts.

7.2 Task division

The team has divided the project into clear components aligned with the architecture stages: data preparation, latent encoding, forward diffusion, denoising (training), reverse diffusion (inference), decoding, and deployment. Each team member is assigned responsibility for specific tasks within these stages, with deadlines set to ensure steady progress and timely completion.

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.

Task	Member	Due	Description
Download and extract data	A, D	June 17	Prepare base image folder
Filter images from metadata	A, D	June 17	Manually select relevant data
Resize and create HR–LR pairs	A, D	June 17	Downsample to 64×64 (LR), resize to 256×256 (HR), and pair for training
Implement PyTorch Dataset class	D, B	June 22	Code loading logic with augmentations; support LR–HR image pairing
Implement VAE encoder/decoder	B, C	June 24	Design latent-space VAE with encoder, decoder, and reparameterization trick
Build UNet-Lite architecture	B, A	July 2	Create U-Net with skip connections, tuned for latent noise prediction
Implement diffusion scheduler	C, B	July 5	Code forward and reverse noise processes, timestep handling, and β scheduler
Train initial model	C, B	July 9	Train with 10k steps, log loss, PSNR, and LPIPS on validation split
Test model using evaluation metrics	A, C	July 10	Add PSNR, LPIPS, FID, and validation logging for model evaluation
Draft progress report	All	July 11	Each member writes a section; final draft compiled and submitted as a group
Export model to ONNX	C, B	July 20	Convert trained model to ONNX format, validate output shape and consistency
Quantize with TensorRT (INT8)	C, B	July 25	Use TensorRT to quantize model and benchmark latency on Jetson
Create Streamlit/Jetson demo	C, A	July 29	Build interface to show before/after on edge device
Generate final poster/slides	D, A	Aug 10	Design final visuals, graphs, and flow diagrams for presentation

An example of task division using ClickUp Gantt charts for this project is shown below, illustrating how tasks are broken down and scheduled over time.

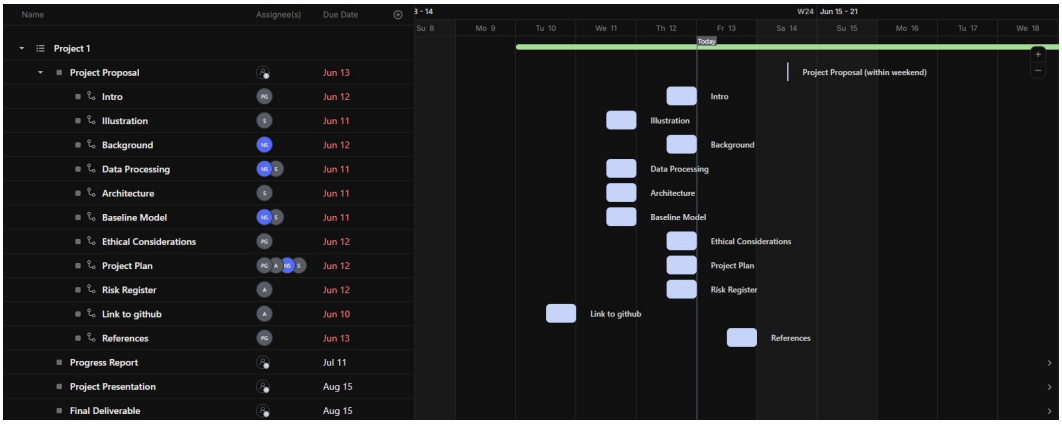

Figure 3: Gantt Chart using ClickUp.

8 Risk Register

Both technical and collaborative issues may arise during the course of the project. Some key risks are outlined in the table below. The impact of an event and its likelihood are scored on a scale of 1 to 5. The risk is calculated as the product of these two metrics.

Table 4: Risk register with impact, likelihood, and mitigation strategies.

Event, Impact	Likelihood	Risk	Mitigation Strategy
Team member drops out or becomes unresponsive. **Score: 3**	Unlikely; members need the skills acquired from this course. **Score: 1**	3	(1) Maintain clear and consistent documentation and internal progress reports. (2) Write well-structured and commented code. (3) Reevaluate team goal and redistribute tasks.
Model training takes significantly longer than expected. **Score: 4**	Likely; we are attempting to train on large datasets without extensive resources. **Score: 5**	12	(1) Start with smaller datasets or lower resolution. (2) Consider transfer learning. (3) Parallelize where possible. (4) Monitor training time early and adjust if needed.
Model exceeds latency or memory limits on edge device. **Score: 5**	Possible; although device has 8GB RAM memory which should suffice. **Score: 4**	10	(1) Quantize aggressively. (2) Profile on-device early. (3) Adjust architecture (e.g., smaller U-Net). (4) Fall back to simpler SR model if absolutely needed (i.e., CNN-based or GAN).
Quantization severely degrades SR output quality. **Score: 3**	Likely; Super-resolution is a fine-grained image generation task, where 8-bit quantization can easily blur or destroy details. **Score: 5**	9	(1) Retrain model while simulating quantization from the start. (2) Use half precision (FP16 instead of INT8).
Generated images contain unrealistic textures or artifacts. **Score: 3**	Likely; models are unpredictable and outputs may be unexpected. **Score: 5**	9	(1) Increase UNet size. (2) Ensure diverse training data. (3) Tune noise schedule and regularization. (4) Visually monitor outputs during training.
Team falls behind due to unclear division of work or miscommunication. **Score: 3**	Unlikely; project plan is detailed and individually assigned. **Score: 2**	3	(1) Assign clear roles early. (2) Hold weekly sync-ups. (3) Use a shared Notion/GitHub board. (4) Create a communication protocol (e.g., Discord updates, weekly deliverables).
Deployment tools (e.g., TensorRT or ONNX) are incompatible or unstable.	Moderate; ONNX conversion for diffusion models can be tricky. **Score: 3**	9	(1) Test the pipeline early. (2) Document known compatibility issues. (3) Be ready to adjust pipeline (e.g., simplify model or use different tools). (4) Be prepared to try Snapdragon (Qualcomm interface).

9 Link to GitHub Repository

GitHub will be used to store and collaborate on code for this project. The link to the repository is: `https://github.com/petrichor-2/image-upscaler`

REFERENCES

T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 2012. ISBN 9781118585771. URL https://books.google.ca/books?id=VWq5GG6ycxMC.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *CoRR*, abs/1501.00092, 2015. URL http://arxiv.org/abs/1501.00092.

M. Galanty, D. Luitse, S.H. Noteboom, S.H. Noteboom, P.M.A. van Ooijen, M. Homan, M. Nagtegaal, J.A.W.M. van der Laak, and H. Huisman. Assessing the documentation of publicly available medical image and signal datasets and their impact on bias using the beamrad tool. *Scientific Reports*, 14:31846, 2024. doi: 10.1038/s41598-024-83218-5. URL https://doi.org/10.1038/s41598-024-83218-5.

Q. Huynh-Thu and M. Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics Letters*, 44:800–801, 2008. doi: 10.1049/el:20080522. URL https://digital-library.theiet.org/doi/abs/10.1049/el%3A20080522.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, 2017. doi: 10.1109/CVPR.2017.19.

Mehdi Noroozi, Isma Hadji, Victor Escorcia, Anestis Zaganidis, Brais Martinez, and Georgios Tzimiropoulos. Edge-sd-sr: Low latency and parameter efficient on-device super-resolution with stable diffusion via bidirectional conditioning, 2025. URL https://arxiv.org/abs/2412.06978.

Tobi Olatunji, Li Yao, Ben Covington, Alexander Rhodes, and Anthony Upton. Caveats in generating medical imaging labels from radiology reports, 2019. URL https://doi.org/10.48550/arXiv.1905.02283. Accepted workshop contribution for Medical Imaging with Deep Learning (MIDL), 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

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]).

Minji Shin, Mohan Seo, Kyung Lee, and Kyuri Yoon. Super-resolution techniques for biomedical applications and challenges. *Biomedical Engineering Letters*, 14(3):465–496, March 2024. doi: 10.1007/s13534-024-00365-4.

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.

Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In Laura Leal-Taixé and Stefan Roth (eds.), *Computer Vision – ECCV 2018 Workshops*, pp. 63–79, Cham, 2019. Springer International Publishing. ISBN 978-3-030-11021-5.

Zhou Wang, Alan Bovik, Hamid Sheikh, Student Member, and Eero Simoncelli. Image quality assessment: From error measurement to structural similarity. *IEEE Trans. Imgage Process.*, 13, 11 2003.