Low-Light Image Reconstruction for Computationally Limited Systems

Matthew Alegrado
ECE Department
UC San Diego
San Diego, CA
malegrado@ucsd.edu

Abstract

This project focuses on the development of a lightweight deep generative image model that will be able to reconstruct dark images into a light counterpart that conveys more usable information than the given dark image. The limitations of this model will be tested in order to determine feasibility when deploying the model on a system with computational cost constraints.

1 Introduction

Computer vision has become more and more common as technology has progressed, from cars to assist with driving to identifying people's faces in pictures. However, in many cases, images are noisy, whether it be from dirty cameras, blur, or most important to my case, low light levels. These images are far from the clean images that often populate datasets that scientists and engineers use to train their models, and without a way to clean dirty test images, these models suffer in their performance post-deployment.

One method to tackle this issue for low light images in particular is to add a preprocessing step before the computer vision (CV) task, and I will be exploring using a deep generative model to infer what the properly lighted image would look like given a dark image as input. A dark image is akin to a lossy transformation of a regularly lit image, as there is less information available per pixel and things like colors and edges may be lost, which means that generative AI could be used to reconstruct that missing information and return a good approximation of the original image.

In particular, I will choose to focus on constructing a lightweight model that could be deployed on systems with computational constraints, e.g., drones, cars, phones. Systems that work in real time often do not have the power to deploy AI models because a large computational cost per image would slow down processing time, which in something like a self-driving car would be counterproductive to the goal of improving performance of another CV task.

1.1 Previous Work

The paper "Restoring Extremely Dark Images in Real Time" (Lamba et al., 2021) [1] covers a very similar idea as my project, using a generative model to restore dark images fast enough to use in a real-time system. The authors used a parallel encoder-decoder and estimated light amplification in order to improve computational speed and reduce size of current, similar models.

The paper "Personalized Generative Low-light Image Denoising and Enhancement" (Wang et al., 2024) [3] uses a smartphone photo gallery to train a personalized diffusion model to improve on diffusion-based denoising approaches and reduce the inaccuracies that arise out of a low signal-to-noise ratio.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

The paper "Autoencoding beyond pixels using a learned similarity metric" (Larsen et al.) [5] uses a GAN as a more effective training method for a VAE to make the loss function less about pixel mean squared error and more about generating perceptibly realistic images.

1.2 Significance and Novelty

While existing works demonstrate real-time restoration and personalized enhancement, they do not explicitly target highly resource-constrained platforms. My approach focuses on maximizing processing speed at the expense of output resolution and quality, downsampling inputs to lower resolutions when necessary. This trade-off is particularly applicable to systems that require rapid but not necessarily high-fidelity vision, such as drones navigating near the ground or low-latency obstacle detection in autonomous vehicles.

However, image fidelity is still important even at lower resolutions as most image datasets have clean images, and models are trained on those clean and realistic images. As such, using both a GAN and a method to generate a lightweight and sparse model that can still generate realistic looking and clean images should be new and impactful as an effective pipeline component in any camera-sensing robotics vehicle.

2 Methodology

For the deep generative model, I will be using a U-Net trained using a generative adversarial network (UNet-GAN) to reconstruct the images. The U-Net consists of a downsampling encoder that uses ResBlocks, and an unsampling encoder consisting of bilinear upsampling followed by a convolutional layer. A crucial addition is the attention gate skip connections that follow each ResBlock and connects each similarly sized output of the encoder to the input of the decoder.

The core strength of this approach comes from the bottleneck combined with the skip connections. As the encoder downsamples images, each of the filters learns more and more global context, while the decoder learns to interpret those cues while still retaining each step's previous encoder output at the current level, meaning the decoder utilizes both learned global and local context at each upsample. This is important to help the model mimic the distortions of low light images, such as how shadows interact with objects. That example requires both local and global knowledge, as the model must understand how the positioning of global light sources can affect local shadow patterns.

Following [6], I used residual blocks with two 3x3 convolutions, the first one being strided for the downsampling in the encoder. This helps with keeping gradients from vanishing as the layers deepen, and also adding another convolution at each hidden dimension, compared to only using one and then moving on to the next hidden layer size.

To ensure the final model is lightweight and robust, I will be using a teacher-student method to train the model, with a larger teacher model training using the GAN and a student model that has a smaller memory footprint and reduced parameter counts that learns from the optimized teacher.

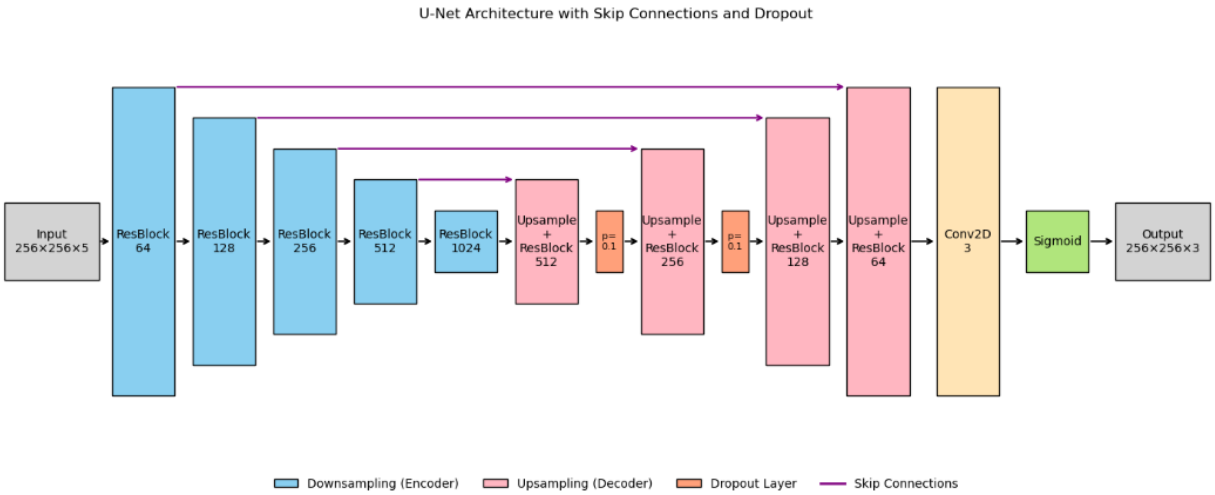


Figure 1: U-Net architecture: contracting path on the left, expanding path on the right, with skip connections. Note the 5-channel input corresponds to the RGB plus 2 augmented channels feeding in ISO and exposure time values.

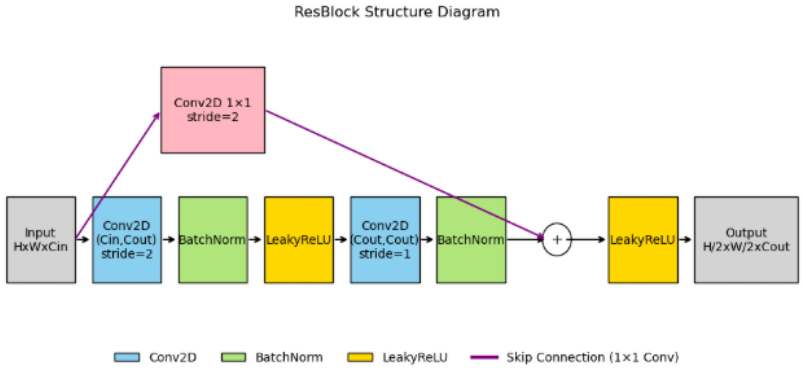


Figure 2: ResBlock architecture.

For the generative loss, I focus on two aspects: reconstruction and structural similarity. For simplicity and to encourage the model to reconstruct images sharply and to not overdo blurring, I chose to use L1 loss over L2 loss. The Structural Similarity Index Measure (SSIM) is a function that outputs in the range (-1, 1), ranging from -1 being complete anti-similarity, 0 being zero similarity, and 1 being complete similarity. It can be used as a loss function if one subtracts the SSIM score from 1 to encourage similarity, as the model will minimize the distance from complete similarity on the outputs.

SSIM has been used as a metric [10] to determine how good the reconstruction is, but I took it a step further and forced the model to optimize over it as an augment to the reconstruction loss. I instead used another metric from the same paper, the PSNR, to measure how much of the image qualities have been amplified relative to the noise that pervades the images due to the low exposure.

Between two images, here is the SSIM score. μ represents the pixel sample mean, and σ represents sample covariance.

$$l(x,y) = \frac{2\,\mu_x\,\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1},$$

$$c(x,y) = \frac{2\,\sigma_x\,\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2},$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\,\sigma_y + c_3}.$$

$$\text{SSIM}(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma}$$

This is the general form of the SSIM score, with constants c_1, c_2, c_3 and α, β, γ used to tune the properties and strength of the luminosity, constrast, and structure elements. For my project, I used the SSIM's default implementation in Torch Metrics, which implements a special version of this general SSIM function where $c_3 = c_2/2$ and all of the exponents are equal to 1.

The generator will be generating an image from the input image, and this will be compared against the input image itself in order to ensure images match the training set well. The GAN will be responsible for discriminating between the two images, so this will be a feedforward ResBlock CNN with a similar structure to the encoder.

My dataset is the See In the Dark (SID) dataset, which comes the researchers behind "Learning to See in the Dark" (Chen et al., 2018) [11]. It includes 5094 raw short-exposure images, with 424 distinct long-exposure images. Multiple short-exposure images correspond to a long-exposure one, usually with varying exposure times or ISO values. There were both indoor and outdoor shots, with the indoor shots having between 0.03 and 0.3 lux, while the outdoor shots have between 0.2 and 5 lux.

Because of the massive overhead that comes with loading raw images into a state usable for my model and for data collection, i.e. printing the images for my own comparison, I created a preprocessing script that converts the raw .ARW image files into .png files. I also added a transformation that converts the dark input images into 5 channel images, RGB plus two channels for camera ISO and exposure time. This gave the model capacity to factor in the properties and different noise artifacts that may arise from those variables and take into account their effect when reconstructing images. I also added a random horizontal flip on the training set with a probability of 0.5. To make training feasible on my own device, I downscaled images to 256x256 when training the teacher, and 128x128 when training the student.

The following table reflects the optimizers and schedulers used in the training process. These had the best overall results for training when compared with other methods used for training.

Component	**Optimizer**	lr	wd	β
Teacher Generator	RAdam	1e-3	1e-5	(0.5, 0.999)
Discriminator	RMSprop	2e-4	5e-5	–
Student Generator	Adam	1e-3	1e-5	(0.9, 0.999)

Table 1: Optimizers and their hyperparameters

Scheduler	**Optimizer**	$T_{\max}$	$\eta_{\min}$	**Metric**
CosineAnnealingLR	Teacher Generator	50	1×10^{-5}	N/A
CosineAnnealingLR	Discriminator	50	1×10^{-5}	N/A
ReduceLROnPlateau	Student Generator	N/A	N/A	L1 Validation Loss

Table 2: Learning-rate schedulers

While training, I used a cosine annealing scheduler to keep both the generator and discriminator's learning rate to proportionally decrease at the same rate. The cosine shape of the scheduler helped

train with a limited number of epochs, as it allows for a high starting LR and decreases it sharply while letting it stay near the minimum for some time. This helped the model converge under a relatively small number of 50 epochs, useful because of my limited access to computational power. Because of the simple loss function on the student, I opted to reduce the learning rate whenever validation loss plateaued

2.1 Attention Gates

The U-Net implementation utilizes skip connections to retain high detail portions of the image at each step, but implementing this naively would add unnecessary clutter to each layer, as unimportant parts of the encoding step may end up being processed through the skip in the decoder, wasting some of the model's capacity cleaning that away.

Attention mechanisms have proven to be effective at improving the learning capacity of models without needing to deepen them [8]. However, attention modules are computationally expensive, requiring multiple fully connected layers or convolutional layers, depending on the type of attention, per layer. This stacks up quickly and I wanted most of the model's limited capacity to be focused on reconstruction rather than visual cleaning.

A good balance between the two sides are attention gates, which use convolutions to encode local information into a skip connection. Following [7], I implemented these gates as two 1x1 convolutions that pass through a channel bottleneck. The reference paper used addition sums, but I instead concatenated channels as that is more natural with a convolutional design. The bottleneck is always half of the incoming channels, so for example, the second to last encoding layer outputs 512 channels. The attention gate would pass those 512 channels through a 1x1 convolution, bringing it down to 256 channels, then another 1x1 convolution that brings it back to 512 channels. Then that is concatenated with the corresponding decoder block's input along the channels, doubling the amount of input channels.

Although this method does not have the full benefits of attention, namely the global context that attention uses to learn overarching dependencies between elements of the image, it requires a mere fraction of the computational cost. Using 1x1 convolutions, this method is incredibly cheap compared to the large projection matrices of attention and full self-attention, while still allowing the skip connections to be a learned part of the model.

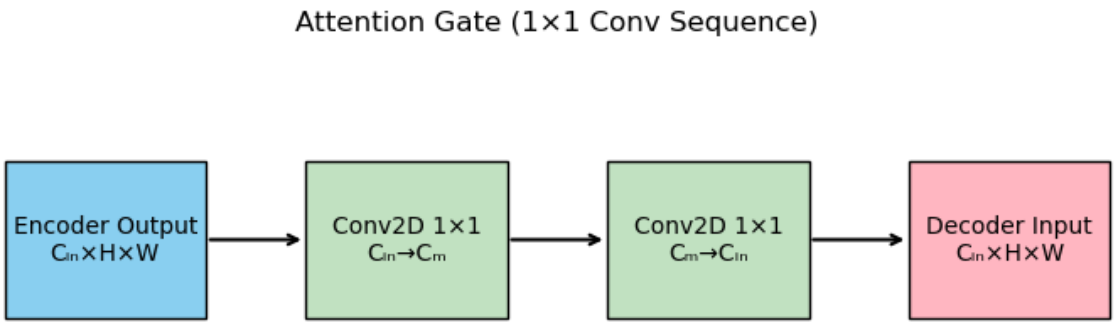


Figure 3: Attention gate implementation. In general, $C_m = \frac{C_{in}}{2}$.

2.2 GAN

The GAN will be a key part of optimizing my model as it will ensure the images are clean and realistic. Image generators often can generate images in such a way that creates artifacts that are not present in real images. For this project's purpose as a step in a larger CV pipeline, these generated images could end up blurring key details, like the text on a sign, for example, which may be important despite the shape of the sign still being accurately captured for the most part. A standard loss function may not be able to optimize these artifacts away, which is why I chose to introduce a discriminator to add adversarial training.

The generative portion of the network, as previously discussed, is the U-Net itself. The discriminator is based on a feedforward ResBlock network and trained seperately from the generative portion of the model. This network will focus on distinguishing between the U-Net's generated outputs and the ground truth image that was used to generate the image. To train this, I use the GAN hinge loss.

$$\mathcal{L}_{Discriminator} = \mathbb{E}_{x\sim p_{\text{data}}}\big[\max\big(0,\ 1 - D(x)\big)\big] + \mathbb{E}_{z\sim p_z}\big[\max\big(0,\ 1 + D\big(G(z)\big)\big)\big],$$
$$\mathcal{L}_{Generator} = -\mathbb{E}_{z\sim p_z}\big[D\big(G(z)\big)\big]$$

Here, $D(x)$ is the output of the discriminator network on input x, while G(z) is the output of the generator, i.e. the image generated by the VAE. I chose to use hinge loss as it prioritizes distance between classification, which is particularly relevant here as there are various types of different low-light conditions, and I wanted to maximize the margins for all the different conditions.

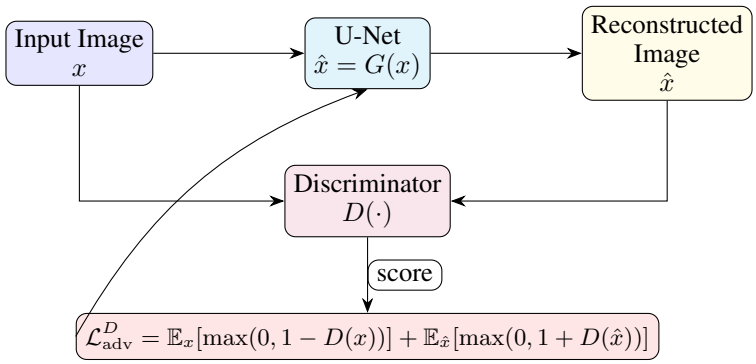


Figure 4: U-Net GAN architecture, showing adversarial loss feeding back into the U-Net generator.

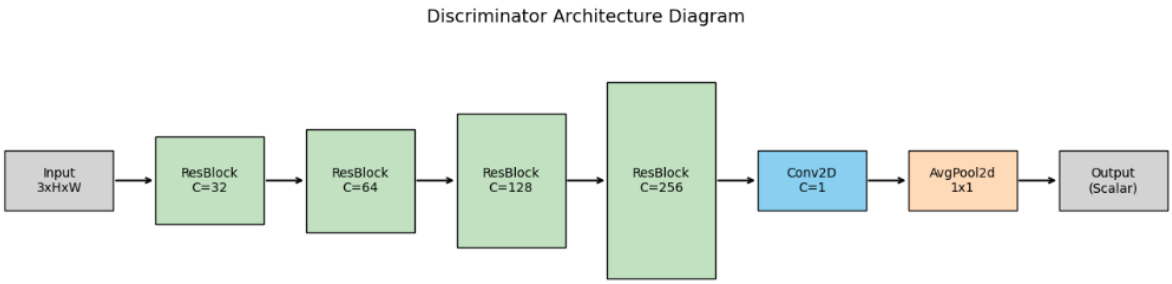


Figure 5: Feedforward CNN discriminator with ResBlocks.

When training the model, I will integrate the various loss functions (SSIM loss, reconstruction loss, discriminator loss) by summing them up and optimizing over them. The overall loss will then be a weighted sum of these terms, with hyperparameters α, β, γ controlling the three respective losses.

In practice, I deal with the expectations on the adversarial loss by taking the sample mean at run time. It is reflected below:

$$\mathcal{L}^{\mathcal{G}} = \alpha\,\mathcal{L}_1 + \beta\,\mathcal{L}_{\text{SSIM}} + \gamma\,\mathcal{L}^{G}_{\text{adv}}$$
$$\mathcal{L}^{\mathcal{D}} = \mathcal{L}^{D}_{\text{adv}}$$

$$\mathcal{L}_1 = \frac{1}{N}\sum_{i=1}^{N} \big| x^{(i)}_{bright} - x^{(i)}_{gen} \big|$$

$$\mathcal{L}_{\text{SSIM}} = 1 - \frac{1}{N}\sum_{i=1}^{N} \textbf{MS-SSIM}(x^{(i)}_{bright}, x^{(i)}_{gen})$$

$$\mathcal{L}^{D}_{\text{adv}} = \frac{1}{N}\sum_{i=1}^{N}\Big[\max\big(0,\, 1 - D(x_i)\big)\Big] + \frac{1}{N}\sum_{i=1}^{N}\Big[\max\big(0,\, 1 + D\big(G(z_i)\big)\big)\Big],$$

$$\mathcal{L}^{G}_{\text{adv}} = -\frac{1}{N}\sum_{i=1}^{N} D\big(G(z_i)\big).$$

Because the discriminator and generator are optimzing opposite sides of an objectives, their is an inherent instability as they each train to beat the other. This is less of an issue for my model since the adversarial weight is only one component of the three for the generator loss, but the instability is still something I wanted to minimize. To that end, I implemented a warmup phase and a discriminator cooldown hyperparameter. During the warmup phase, the generator focused on optimizing without any discriminator, and after the warmup phase ends, the discriminator starts working.

I found that the discriminator learned much quicker, so I also added a cooldown, which is how many epochs it will wait before it trains again. This keeps the training more balanced, as the generator gets more time to optimize over the discriminator's learning, which is important as it also needs to optimize reconstruction and SSIM loss. More details regarding the effectiveness in section 5.

2.3 Teacher & Student

Although the discriminator is useful in training the model to ensure the images are realistic and match the dataset, it will be disregarded at test time, and the model will only use the trained U-Net that has learned to fool the discriminator. But to ensure there is no model collapse during the adversarial training, the model needs to be strong enough to learn how to generate realistic images. This is counterproductive to my original goal, so to remedy that I will employ a student model that learns from the teacher model.

Inspired by [9], I implemented a smaller scale student model that uses less and smaller hidden layers, while still retaining the main design points of the teacher, such as the attention gated skips and ResBlocks. The paper used L1 loss and KL divergence, but since I don't use a VAE, I simply stuck to L1 loss for maximum sharpness when reconstructing per-pixel.

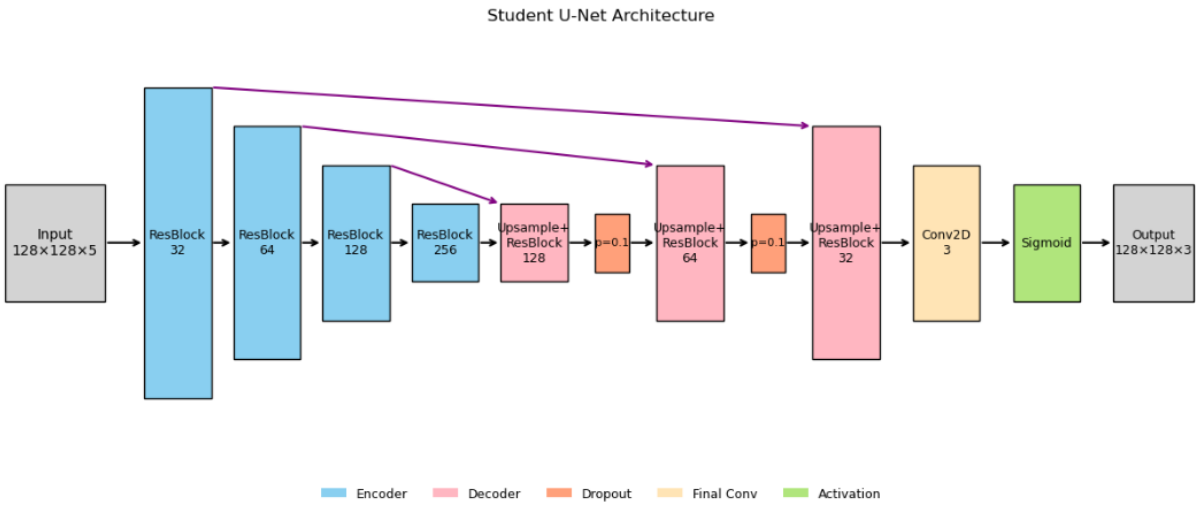


Figure 6: Student U-Net architecture, following similar design paradigms to the teacher U-Net.

The teacher will be the model that trains with the GAN and is optimized to both generate realistic images and generate images that resemble the input image. In other words, it is a model that is strong enough to fool the discriminator without losing sight of its primary objective of reconstructing dark images. The student will be a smaller model, with less parameters and smaller dimensionality, that will learn to mimic the teacher's outputs. More specifically, the teacher will be given a training image and output its reconstructed image, and the reconstruction is used as the students "ground truth" when it trains. The student model will learn to reconstruct images in the teacher's style, which ideally is both realistic and accurate to the dark image.

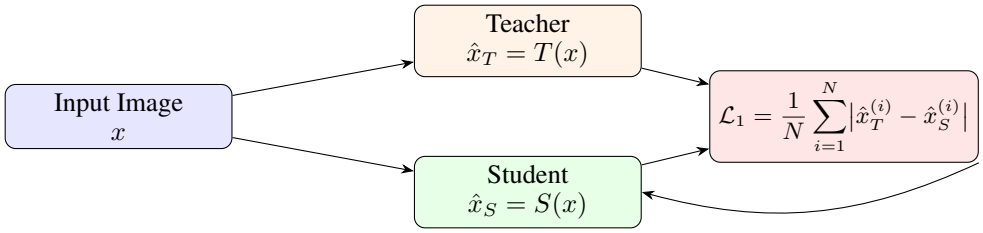


Figure 7: Student and Teacher run in parallel on the same input x; their reconstructions $\hat{x}_T$ and $\hat{x}_S$ are compared via an L1 loss.

Here, the student will take inputs directly from the teacher's outputs, which are locked to inference mode to prevent further training, and it uses that as the ground truth image instead of the true long-exposure image. The reason I did this was to ensure that the student learned to mimic the teacher as closely as possible. Recall that I am doing this knowledge distillation because the model is necessarily complex in order to train well on the original set. Once that has been learned with the large parameter U-Net teacher, I simply condense those network patterns by having the student match the teacher outputs directly.

3 Limitations

One of the biggest disadvantages of this model is the large amount of training that needs to be done. This model must train multiple components which are each quite large: the U-Net encoder, U-Net decoder, and discriminator. Training CNNs is computationally expensive, especially for high resolution images like those in the SID dataset is extremely expensive, and even with rescaling to 256x256, training on at least 100 epochs takes a very long time. Of course, downscaling images comes at a cost of faithfully reproducing high resolution images, which limits this project's applications for cases that require high resolution reconstructions, but the focus is on reproducing images quickly on

an embedded system so rescaling is less detrimental and even expected, as the extra reconstruction resolution is likely not worth the computational cost.

The adversarial training scheme presents some difficulties. One issue that may arise is instability from the GAN loss, as there is no guarantee that as the generator improves and the discriminator learns, that they both will reach some equilibrium point. Another issue may be that a generative network that is not sufficiently powerful enough to fool the discriminator will collapse into producing only a small set of images that is good at fooling the discriminator. Solving all of these issues involves heavy hyperparameter tuning, which made the runtime of this project even longer.

4 Timeline

- **Week 3:** Project Proposal
- **Week 5:** Have a draft of architecture design
- **Week 7:** Finish implementing architecture draft, begin training on dataset
- **Week 8:** Modify architecture as needed, optimize hyperparameters
- **Week 9:** Minimize model size as much as possible, record processing speed at test time
- **Week 10:** Visualize results, collect examples, and compile final report

5 Results & Analysis

5.1 Teacher U-NetGAN

(a) Reconstruction Loss

(b) SSIM Loss

(c) Discriminator vs. Generator Loss

(d) PSNR (dB)

Figure 8: Training curves for key metrics: (a) full loss, (b) SSIM loss, (c) GAN losses, (d) reconstruction loss.

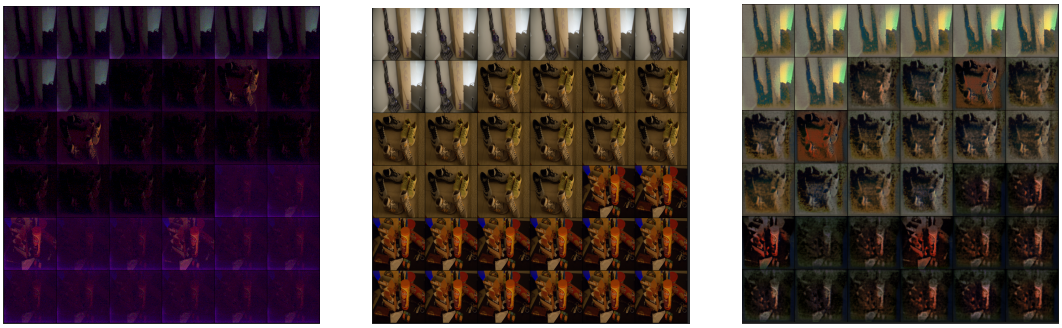
Here are the plots from training and validation across 50 epochs, labeled by training step on the x-axis. We can see that while the reconstruction loss optimized decently well for a multiple termed loss function over 50 epochs, the SSIM struggled to lower significantly on the training set, starting at around 0.57 and ending around 0.51. For reference, the chosen weights on the loss function were 0.5 for reconstruction, 0.15 for adversarial, and 0.35 for SSIM loss. Despite accounting for a third of the loss weight, the SSIM loss only went down about 10%. Strangely enough, it decreased the SSIM loss much better on the validation set, and similarly for reconstruction, meaning the training set may be much more difficult than the validation set.

Looking at the discriminator loss, there was a long period of imbalance where the discriminator was a fair bit stronger than the generator. However, by the end of training, the two seem to balance out. This was expected; recall the discriminator had a learning rate at 20% that of the generator, and the generator had 15% of the loss function dedicated to adversarial penalties. It seems giving a slight edge to the discriminator kept training stable and kept adversarial artifacts to a minimum.

Overall, it seems that training the U-Net using a GAN worked well, but the overly strong emphasis on the SSIM seemed to be detrimental, as the model failed to reliably reduce it and subsequently increase the SSIM score. It may be better suited as a metric rather than a loss term. The model did a decent job of increasing PSNR, reflecting its ability to retain the image information in the presence of noise that the darkness and low exposure times add to the images. However, a value of 18 dB is still relatively poor, and better models aim for at least 25 dB [10].

Metric	**Test Set**
Full loss	0.1984
Reconstruction loss	0.0972
SSIM loss	0.4279
PSNR (dB)	18.3982

Table 3: Key evaluation metrics on the test dataset.



(a) Input (high ISO) (b) Ground truth (bright) (c) Reconstruction

Figure 9: High-ISO test example: (a) noisy input, (b) long-exposure ground truth, (c) model reconstruction.

(a) Input (low ISO) (b) Ground truth (bright) (c) Reconstruction

Figure 10: Low-ISO test example: (a) noisy input, (b) long-exposure ground truth, (c) model reconstruction.

Looking at the test set examples, the model did a decent job at reconstruction, but zooming into individual examples makes its weakness apparent. On the low-ISO set, some reconstructions (corresponding to lowest-exposure) images are dark and green. The hue is an artifact from the discriminator training, as it showed up during training at times when the adversarial training hadn't stabilized yet. The higher low-exposure images show good reconstructions of the foreground, but fail to capture the

(a) Input (dark) (b) Ground truth (bright) (c) Reconstruction

Figure 11: Validation example: (a) dark input, (b) long-exposure ground truth, (c) model reconstruction.

background accurately, especially in the top shots of the full basketball court. The reconstructions also failed to capture dark objects, as evidenced by the base of the pole missing in the bottom shots of the basketball close-up.

On the high-ISO set, we can see the inputs have lots of noisy information and it is easy to make out the rough image just on the low exposure set. The model did a good job of cleaning up the foreground on the higher exposure dark images but once again failed to consistently generalize to the lowest exposure. On the shoes, it is hard to tell where one shoe ends and another begins. Clearly, the model was not able to learn how to denoise the high ISO images and extrapolate shapes from very small pixel values on the low ISO data.

I suspect this may partially be an issue with the training set, as the high ISO (>5000) images only consist of about 20% of the training data. For reference, the high ISO test images were at 8000 ISO. There simply may not have been enough emphasis on learning that during training. The loss function may also be unsuited for denoising, as a large emphasis was put on structural similarity. This may have shifted the focus away from denoising and more towards simply roughly matching per-pixel reconstruction and general structural elements, and the discriminator's weight would be too weak to truly get rid of the noise.

I would follow work done in [12] and convert my discriminator to a similar U-Net and focus the training to solely be on the adversarial terms, making it a true GAN, as this may encourage actual denoising and a good discriminator should push the generator to have good structural similarity metrics too, even if not directly optimizing for it. Other works [10] also suggest using a perceptual loss using a pretrained VGG as a method of improving image visual acuity and reducing the reconstruction noise.

On the test set, it is clear which images were generated from the lowest exposure (0.04s) and which were generated from the other exposure (0.1s). Some of the images are far darker than their other counterparts, and the way the dataset is organized shows repetitions of similar ISO values but differing exposure times. In fact, the more poorly reconstructed images always corresponds to the shortest exposure time, regardless of ISO. Comparing the test reconstructions between the first images (low ISO) and last images (high ISO), we can see this relationship still hold.

Despite inputting the ISO and exposure time into the model as a separate channel, it does not seem to have had a significant effect in helping the model "customize" its reconstructive transformation. It is also possible that there simply is not enough information in the 0.04s exposure images to cleanly reconstruct it. Also, there is a notable hue difference in all of the images, which is a weakness of the L1 and SSIM loss, as they don't penalize those at all. In the future I may look into other loss functions which penalize differences in hue that could serve as small additional penalties to help the model generate more faithful reconstructions.

5.2 Student U-Net

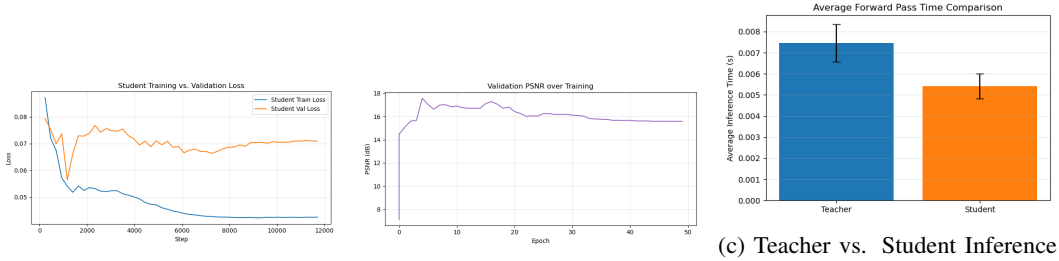
In terms of cost saving, the teacher model had 33.6 million parameters, while the student model had 2.4 million. This is over ten-fold decrease in the model's memory footprint, which is significant

Name	Params (M)	Size (MB)
U-NetGAN	43.0	172.0
Teacher U-Net	38.1	152.4
Discriminator	4.9	19.6
Student U-Net	2.4	9.6

Table 4: Parameter counts and estimated memory footprint (assuming 4 bytes per parameter). The U-NetGAN consists of both the Teacher U-Net and the Discriminator combined.

for systems with processors with just a few megabytes of memory to spare for this step in the CV pipeline. In terms of computational speed, the teacher averages 7.5 milliseconds while the student averaged 5.5 milliseconds. Not nearly as large of an improvement, taking about 75% of the time on the student compared to the teacher, but significant nonetheless.

Although these results are not as meaningful without a timed benchmark of existing, similar models on my own machine, it still demonstrates the effectiveness of knowledge distillation in reducing both model complexity and inference time without sacrificing a large amount of reconstruction accuracy and generalizability.

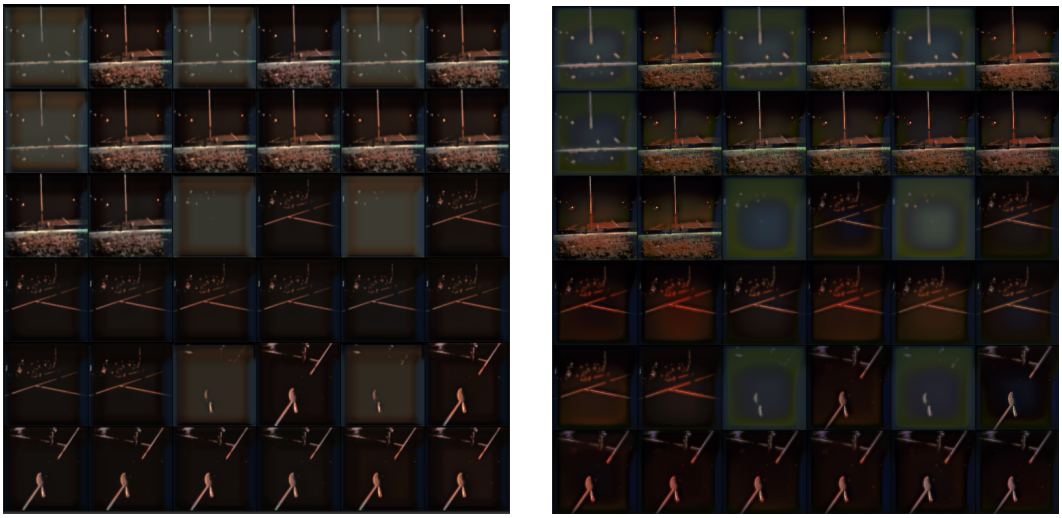


(a) Student Full Loss (b) Student Validation PSNR (c) Teacher vs. Student Inference Time

Figure 12: Student model performance: (a) full training & validation loss, (b) validation PSNR over epochs, (c) average inference time comparison.

The plots suggest that the student overfitted on the model, which is strange because the student still had a decent amount of weight decay and dropout in the decoder. The validation PSNR was slightly worse than the teacher, hovering around 16 which decreased as the model trained. However, the reconstructions were still very good relative to the teacher outputs.

Looking at the test images, the student captures the likeness of the teacher incredibly well. It keeps the distortions and general patterns of the teacher, like how it reconstructs foreground vs. background, replacing dark parts like the pole as a shadow, and pretty closely matches the color too, with a slight reduction in overall hue. Despite the imperfect reconstructions of the teacher, the student clearly is able to generalize the teacher's outputs even with more than 10x fewer parameters.



(a) Student Model Output (b) Teacher Model Output

Figure 13: Comparison between the student network's output the teacher network's output on the same test images.

6 Conclusion

Looking over all the data, it seems I may have chosen sub-optimal training goals for the model. Including SSIM loss seemed to provide no practical benefit, and the adversarial training was not influential enough to have made the images resemble ground truth without all the noise added from ISO and other low-exposure artifacts. The L1 term is standard and did seem to optimize well, so future work could include a 50/50 split between reconstruction and adversarial loss, between two U-Nets, instead of one U-Net and one feedforward CNN.

Despite this, the knowledge distillation pipeline showed promising results, with the tiny student model being able to reconstruct the teacher's outputs with a high degree of accuracy and generalizability for the fraction of the memory and computational cost. If I improved the teacher model using the methods outlined above, the student could easily produce realistic bright reconstructions of dark images even with variable ISO and exposure times. Further refinements could push the image quality to a powerful and cheap tool for nighttime computer vision applications in real-time, given how fast the inference time is.

References

[1] Lamba, M., & Mitra, K. (2021). Restoring Extremely Dark Images in Real Time. The Computer Vision Foundation.

[2] Zhao, L., Li, X., & Meng, R. (2022). A low-light image enhancement method based on AUDD-GAN. Journal of Physics: Conference Series.

[3] Wang, X., Chennuri, P., Yuan, Y., Ma, B., Zhang, X., & Chan, S. (2024). Personalized Generative Low-light Image Denoising and Enhancement. arXiv preprint arXiv:2401.01234.

[4] Lin, R., Anantnasirichai, N., Huang, G., Lin, J., Sun, Q., Malyugina, A., & Bull, D.R. (2024). BVI-RLV: A Fully Registered Dataset and Benchmarks for Low-Light Video Enhancement. arXiv preprint arXiv:2402.05678.

[5] Larsen, A.B.L., Sønderby, S.K., Larochelle, H., & Winther, O. (2016). Autoencoding Beyond Pixels Using a Learned Similarity Metric. In *ICML*, arXiv preprint arXiv:1512.09300.

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *CVPR*. https://arxiv.org/abs/1512.03385

[7] Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., et al. (2018). Attention U-Net: Learning Where to Look for the Pancreas. arXiv preprint arXiv:1804.03999. https://arxiv.org/pdf/1804.03999

[8] Xu, G., Wang, X., Wu, X., Leng, X., & Xu, Y. (2024). *Development of Skip Connection in Deep Neural Networks for Computer Vision and Medical Image Analysis: A Survey*. arXiv:2405.01725. https://arxiv.org/abs/2405.01725

[9] Zhang, Y., & Yan, D. (2025). *Knowledge Distillation for Image Restoration: Simultaneous Learning from Degraded and Clean Images*. arXiv:2501.09268. https://arxiv.org/abs/2501.09268

[10] Feijoo, D., Benito, J.C., García, Á., & Conde, M.V. (2025). *DarkIR: Robust Low-Light Image Restoration*. arXiv:2412.13443v2. https://arxiv.org/abs/2412.13443v2

[11] Chen, C., Chen, Q., Xu, J., & Koltun, V. (2018). *Learning to See in the Dark*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 3291-3300. arXiv preprint arXiv:1805.01934. https://arxiv.org/abs/1805.01934

[12] Schönfeld, E., Schiele, B., & Khoreva, A. (2020). A U-Net Based Discriminator for Generative Adversarial Networks. arXiv preprint arXiv:2002.12655. https://arxiv.org/abs/2002.12655