BME 2025 Deep Learning Project

Enhancing Image Super-Resolution with Generative Adversarial Networks (GANs)

Team VEO:
Vince Szigetvari (OWC5ZP)
Eduardo Meza Medina (IKUL3K)
Onur Aktan (H55CGX)

1 Introduction

Image Super-Resolution is a fundamental computer vision task focused on reconstructing high-resolution (HR) images from low-resolution (LR) inputs, essential for applications ranging from medical imaging to surveillance. While traditional interpolation methods are computationally efficient, they often fail to recover high-frequency details, resulting in blurry outputs [5].

Deep learning approaches, specifically Convolutional Neural Networks (CNNs), improved upon this but often produce overly smooth textures when minimizing Mean Squared Error [1]. To address this, Generative Adversarial Networks (GANs) [2] introduced adversarial training, allowing a generator to create perceptually realistic images by competing against a discriminator. The Super-Resolution GAN (SRGAN) [6] pioneered this approach by combining adversarial loss with perceptual loss [7] to generate sharper details. Building on this, the Enhanced Super-Resolution GAN (ESRGAN) [8] refined the architecture by introducing Residual-in-Residual Dense Blocks and relativistic adversarial loss [4] to improve training stability and visual fidelity.

This project implements and trains both SRGAN and ESRGAN on the DIV2K dataset to evaluate their architectural choices and training dynamics. We focus on critical tuning aspects, such as the removal of batch normalization [3] and the balancing of adversarial and perceptual losses to mitigate artifacts. While our results show that traditional metrics like PSNR fell slightly below the bicubic baseline, both GAN models demonstrated significantly superior perceptual realism. ESRGAN, in particular, offered the most convincing visual results with sharper textures, highlighting the limitations of pixel-wise metrics in capturing human perception. Ultimately, this study underscores the effectiveness of adversarial learning in bridging the gap between mathematical reconstruction and human visual expectations.

2 Methodology

In this project, we explore deep generative models for image super-resolution, specifically focusing on Generative Adversarial Networks (GANs) [2]. A GAN consists of two competing neural networks: a generator that aims to produce realistic images, and a discriminator that distinguishes between real and generated samples. Through this adversarial training process, the generator learns to synthesize perceptually convincing and visually appealing high-resolution images.

Our work primarily investigates two well-known GAN-based architectures for ISR: the Super-Resolution GAN (SRGAN) [6] and the Enhanced Super-Resolution GAN (ESRGAN) [8]. SRGAN was the first to achieve photo-realistic single-image super-resolution by combining adversarial loss with a perceptual loss computed from VGG feature maps [7], leading to significantly sharper textures than models trained solely with pixel-wise losses such as Mean Squared Error (MSE). Building upon SRGAN, ESRGAN introduces several architectural improvements—such as Residual-in-Residual Dense Blocks (RRDBs), the removal of batch normalization, and a relativistic adversarial loss—to further enhance perceptual realism and training stability.

In this study, we implement and evaluate both SRGAN and ESRGAN using Python and PyTorch. By comparing their quantitative performance (PSNR, SSIM) and qualitative results against classical interpolation methods, we aim to analyze how adversarial and perceptual learning contribute to generating visually superior high-resolution images. LLM tools were utilized to assist coding, mainly Github Copilot and ChatGPT.

2.1 SRGAN Overview

2.2 SRGAN Architecture

The SRGAN includes a generator and a discriminator network. The generator generates HR outputs from the LR images, and the discriminator is used to differentiate between real HR images and those that the generator produces. The adversarial loss, content loss, and perceptual loss altogether contribute to the training objective that guides the generator in generating visually realistic textures, together with structurally accurate reconstructions.

2.2.1 Discriminator Network

In SRGAN, the discriminator network is used as a binary classifier to distinguish between the real and generated high-resolution images. Its architecture is based on principles established in Deep Convolutional GANs (DCGAN) by Radford *et al.* (2016), employing a deep convolutional design with no pooling layers.

Input & Output: The discriminator takes an HR image patch of size 96×96 pixels and outputs a scalar probability that the input image is real or generated by the generator.

Feature Extraction: It uses several convolutional layers, each with an increasing amount of feature maps ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512$). Instead of pooling, the network applies strided convolutions at each layer, so downsampling is learned rather than fixed.

Activation and Normalization: Each convolutional layer is followed by a LeakyReLU activation ($\alpha = 0.2$) to avoid neuron saturation and improve the flow of gradients. Batch normalization is used to stabilize training and improve convergence [3].

Fully Connected Layers: The final convolutional feature maps are flattened and go through dense layers, which finally leads to a sigmoid output neuron representing the probability of the input being real. The objective function of the discriminator follows the standard binary cross-entropy loss used in GANs:

$$\mathcal{L}_D = -\mathbb{E}[\log D(I^{\mathrm{HR}})] - \mathbb{E}[\log(1 - D(G(I^{\mathrm{LR}})))]. \tag{1}$$

The discriminator aims to maximize this objective by correctly classifying real and generated samples, while the generator attempts to minimize it to successfully fool the discriminator.

We used 8 residual blocks in our implementation.

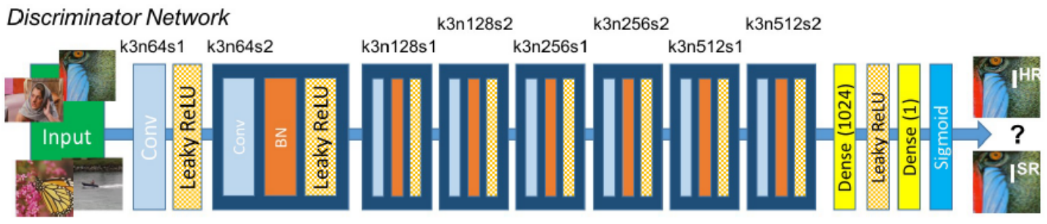


Figure 1: Discriminator Architecture

2.2.2 Loss Functions

Content (Perceptual) Loss

Instead of using a simple pixel-wise MSE loss, SRGAN incorporates a VGG-based perceptual loss. This measures differences between feature representations extracted from a pre-trained VGG-19 network [7], encouraging the generated images to match the high-level perceptual features of the ground-truth HR images rather than just pixel values.

$$\mathcal{L}_{\text{content}} = \frac{1}{WH} \sum_{x,y} \left(\phi_{i,j}(I^{\text{HR}})_{x,y} - \phi_{i,j}(G(I^{\text{LR}}))_{x,y} \right)^2 \tag{2}$$

Here, $\phi_{i,j}$ represents the feature maps extracted from the j-th convolution layer before the i-th pooling layer of the VGG-19 network. In our implementation, we extracted features from the ReLU activation after the 8th convolutional layer before the 3rd max-pooling layer (first 18 layers in the VGG network).

Adversarial Loss

The adversarial loss encourages the generator to produce images that the discriminator classifies as real:

$$\mathcal{L}_{\text{adv}} = -\mathbb{E}\left[\log D(G(I^{\text{LR}}))\right] \tag{3}$$

Total Generator Loss

The total generator loss is a weighted combination of perceptual and adversarial losses:

$$\mathcal{L}_G = \mathcal{L}_{\text{content}} + 10^{-3} \times \mathcal{L}_{\text{adv}} \tag{4}$$

This weighting factor balances perceptual fidelity with texture realism, allowing the generator to produce sharper, more natural-looking images. If the adversarial loss weight is too low, the discriminator will be ignored, thus reducing the feedback to the generator about the realism of generated images. This behavior is seen during training when the discriminator loss quickly becomes 0. During training, we observed this behavior and hence increased the adversarial weight to 5×10^{-2}.

2.3 ESRGAN

2.3.1 Modified Generator Network

The Enhanced Super-Resolution Generative Adversarial Network (ESRGAN), proposed by Wang *et al.* (2018), builds upon SRGAN to further enhance perceptual quality and stabilize training. The main architectural innovation is the use of Residual-in-Residual Dense Blocks (RRDBs), which combine dense connections and residual learning to improve feature representation and gradient flow.

2.3.2 Key Architectural Changes Compared to SRGAN

Residual-in-Residual Dense Block (RRDB): Each RRDB merges the ideas of multiple densely connected convolutional layers within a residual framework. This design allows for reusing features across layers, helping the network reconstruct textures and fine details more effectively. Batch normalization is removed in RRDBs to reduce computation and avoid visual artifacts.

Batch Normalization Removal: Unlike SRGAN, ESRGAN omits all batch normalization layers. This prevents over-smoothing and color distortion caused by normalization statistics, resulting in sharper and more stable outputs [3].

Residual Scaling and Smaller Initialization: To stabilize training of deeper networks, ESRGAN applies a residual scaling factor (typically 0.2). Smaller weight initialization values also help control gradient magnitude, reducing the risk of divergence during adversarial training.

Deeper Generator Network: ESRGAN increases the number of residual blocks compared to SRGAN, which uses 16 basic residual blocks. This deeper design helps the network learn more complex features and finer textures, ultimately improving perceptual quality.

Generator Pipeline:

$$\text{Input Image} \rightarrow \text{Feature Extraction} \rightarrow N \times \text{RRDB} \rightarrow \text{Upsampling Layers} \rightarrow \text{Output Image}$$

We used 12 RRDB blocks in our implementation.

2.3.3 Modified Loss Functions

ESRGAN also refines the loss functions from SRGAN to achieve more realistic and visually pleasing results.

Perceptual Loss Based on Features Before Activation: Unlike SRGAN, which computes perceptual loss from activated VGG-19 feature maps, ESRGAN uses pre-activation features (before ReLU). These features contain richer information about texture, contrast, and color, helping to produce images that are perceptually sharper and more natural [7]. In our implementation, we extracted features from the convolutional layer before the 8th ReLU activation (after the 8th convolutional layer) (first 17 layers in the VGG network).

Relativistic Adversarial Loss: ESRGAN introduces the Relativistic average Discriminator (RaD), which doesn't just decide if an image is real or fake. Instead, it compares how much more realistic a real image is than a generated one [4]. This relativistic approach improves training stability and enhances the realism of fine details.

Total ESRGAN Generator Loss: The final generator loss in ESRGAN is a weighted combination of:

- Perceptual loss ($\mathcal{L}_{\text{percep}}$)
- Relativistic adversarial loss ($\mathcal{L}_G^{\text{Ra}}$)
- Pixel-level L_1 loss

$$\mathcal{L}_G^{\text{ESRGAN}} = \mathcal{L}_{\text{percep}} + \lambda_{\text{adv}} \cdot \mathcal{L}_G^{\text{Ra}} + \lambda_1 \|G(I_{\text{LR}}) - I_{\text{HR}}\|_1 \tag{5}$$

Here, λ_{adv} and λ_1 are weighting coefficients. The use of L_1 loss (instead of MSE in SRGAN) helps achieve sharper edges and better contrast. In our implementation, we set $\lambda_{\text{adv}} = 5 \times 10^{-3}$ and $\lambda_1 = 1 \times 10^{-2}$ based on recommendations from the original paper. We changed the adversarial weight to 5×10^{-2} to stabilize training.

3 Training

We trained both SRGAN and ESRGAN models on the DIV2K dataset. There are 19,570 train image-pairs, 4,193 validation image-pairs, and 4,195 test image-pairs in total. The size of the LR images is 64×64, and 256×256 for HR images. Originally, LR images were 256×256 (but more blurry), but we downscaled them to 64×64 using `cv2.INTER_CUBIC` to create a more challenging super-resolution task.

3.1 Training SRGAN

All training was executed on Google Colab using an NVIDIA L4 GPU. We monitored training progress and logged metrics using Weights & Biases (`wandb`).

We first pretrained the SRGAN generator for 20 epochs using only MSE content loss to provide a good initialization for adversarial training. One epoch took around 4 minutes. The loss is visible in the plot below.

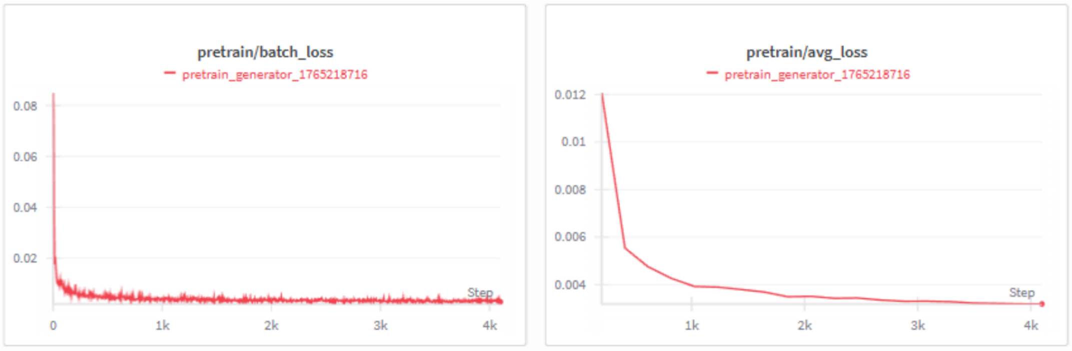


Figure 2: Batch Loss

After pretraining, we trained the full SRGAN model for 37 epochs, and one epoch took around 13.5 minutes. After 7 epochs, we noticed that the discriminator loss became very small (close to 0), meaning that the discriminator was dominating the training and the generator was not receiving useful feedback. To mitigate this, we increased the adversarial loss weight from 1×10^{-3} to 5×10^{-2}. This adjustment helped balance the training dynamics between the generator and discriminator. Below, the training loss plots are shown.

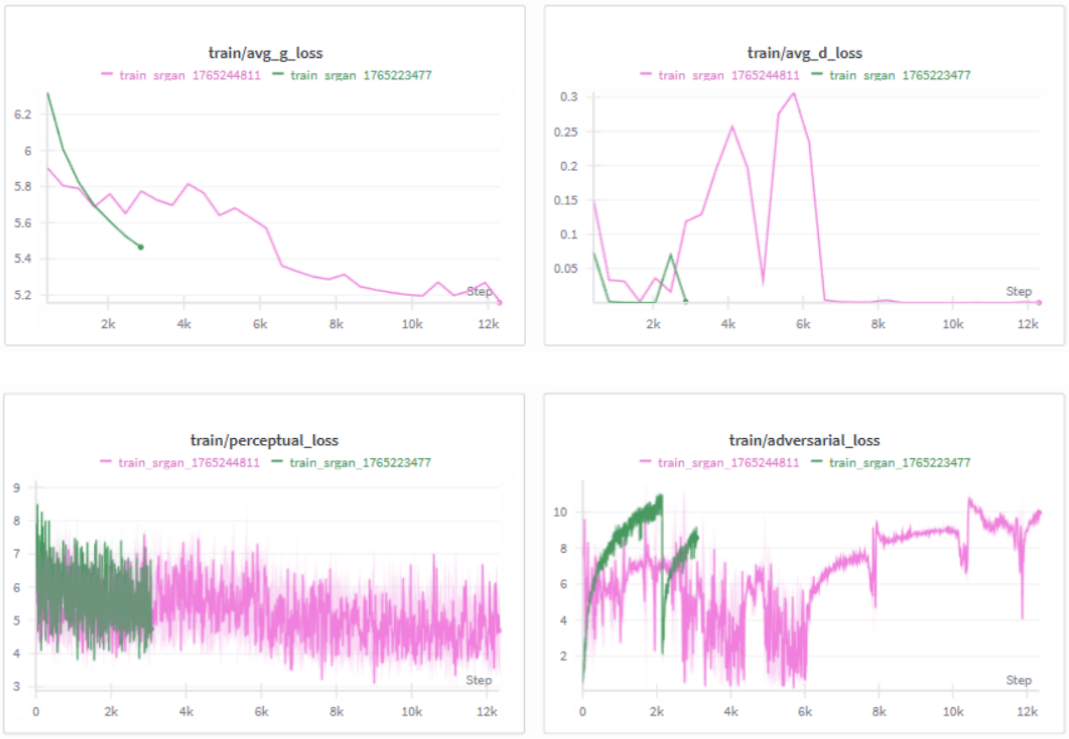


Figure 3: SRGAN Training Loss

3.2 Training ESRGAN

Similar to SRGAN, we first pretrained the ESRGAN generator only using L_1 loss to provide a good initialization for adversarial training. This time we stopped pretraining after 4 epochs as the loss plateaued. One epoch took around 13–14 minutes. The loss curves are shown below.

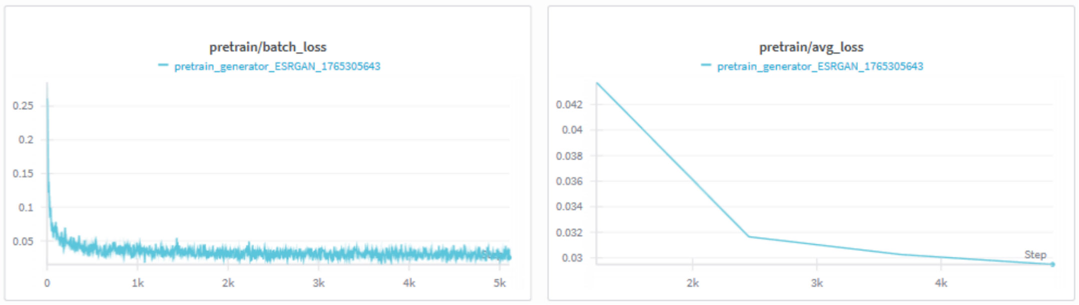


Figure 4: ESRGAN Training Loss

After pretraining, we trained the full ESRGAN model for 7 full epochs, and one epoch took around 24–25 minutes. We quickly noticed that the discriminator loss became very small again (close to 0), so we again increased the adversarial loss weight from 5×10^{-3} to 5×10^{-2}. This adjustment helped balance the training dynamics between the generator and discriminator. Below are the training loss plots.

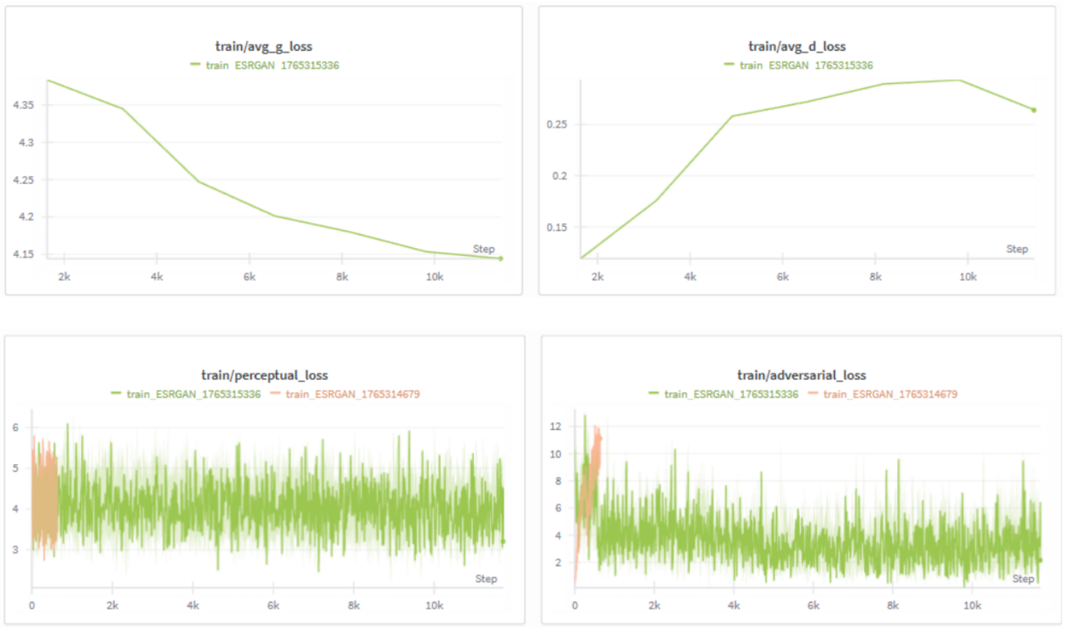


Figure 5: ESRGAN Training Loss

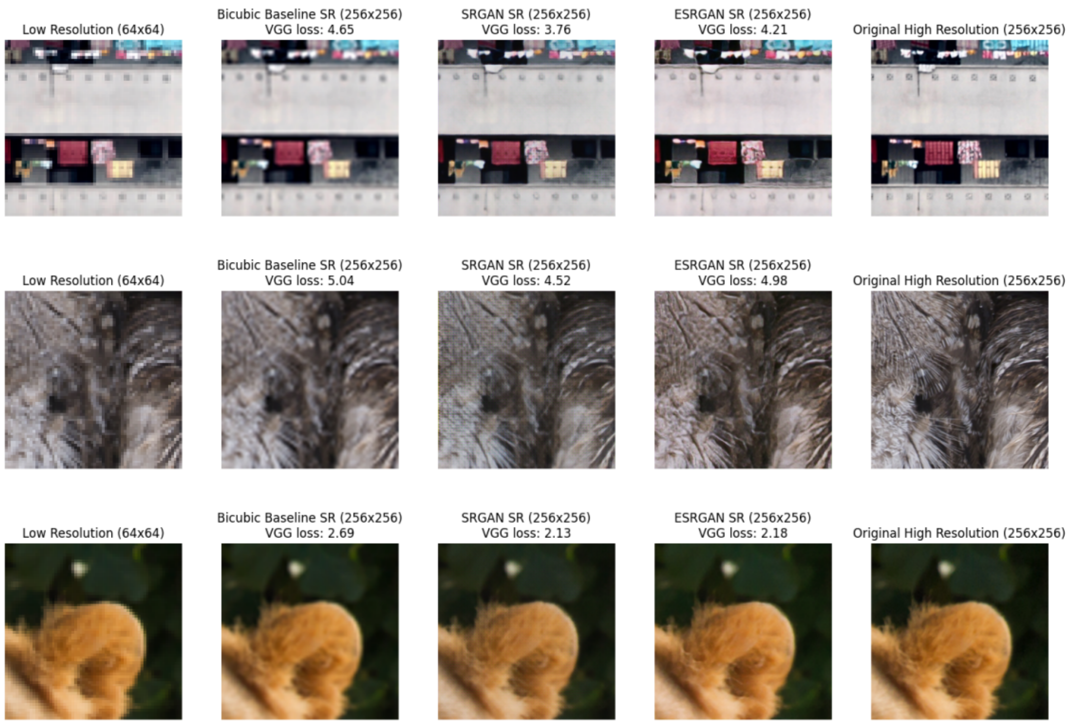
4 Evaluation

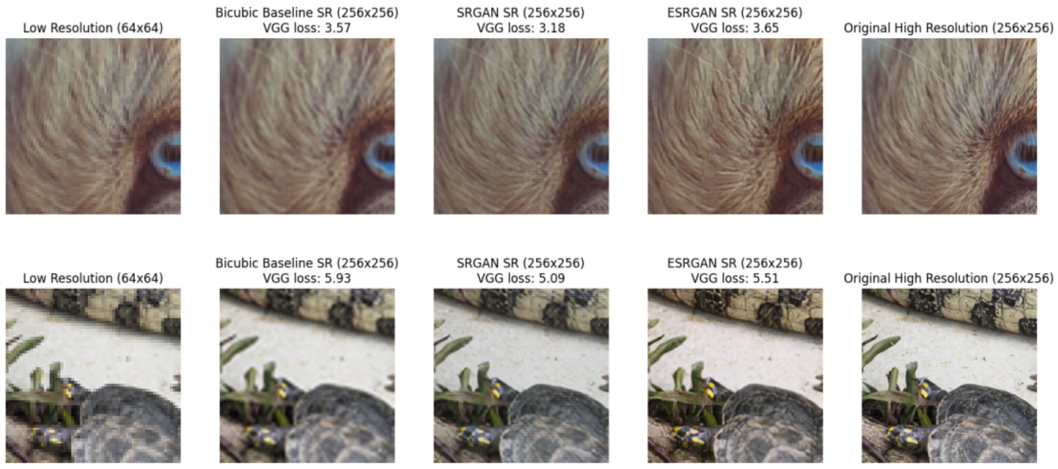
We used PSNR (Peak Signal-to-Noise Ratio) and VGG-based Perceptual Loss as quantitative metrics to evaluate the performance of the trained SRGAN and ESRGAN models on the DIV2K test set. For the VGG loss, we used the same layers as in the ESRGAN case (the first 17 layers before the 8th ReLU activation). The results on the test set are summarized in the table below:

Table 1: Quantitative results on the DIV2K test set.

Model	**PSNR (dB)**	**VGG Loss**
Bicubic Baseline	25.25	4.2891
SRGAN	23.48	3.5674
ESRGAN	23.70	3.8646

Based on metrics, both SRGAN and ESRGAN are slightly underperforming the bicubic baseline in terms of PSNR, which is expected as GAN-based models prioritize perceptual quality over pixel-wise accuracy. However, both models achieve lower VGG loss compared to bicubic interpolation, indicating better perceptual similarity to ground truth images. Based on these quantitative results, the SRGAN model outperforms ESRGAN in terms of perceptual loss, while ESRGAN has a slight edge in PSNR. However visually, the ESRGAN results appear to be more detailed and sharper compared to SRGAN. A known issue in SR tasks is that hard to capture quality with metrics alone. A common approach is to conduct Mean Opinion Score (MOS) tests, where human raters evaluate the visual quality of images. Due to time constraints, we could not perform a full MOS study, but we looked at results obtained from both models and generally preferred ESRGAN outputs for their enhanced texture details and realism. Below are some example outputs from both models compared to bicubic interpolation and ground truth HR images.





5 Conclusions

This project showed that GAN-based models can push super-resolution closer to what people expect visually, even when traditional metrics do not fully capture the improvement. SRGAN and ESRGAN both produced sharper textures than bicubic interpolation, and ESRGAN in particular tended to deliver the most convincing details in our visual comparisons. At the same time, training these models proved sensitive to loss weighting and discriminator behavior, which shaped much of the practical work in getting them to converge.

If we continued this project, we would focus on stronger data augmentation, longer and more stable training schedules, and better perceptual loss functions. A proper MOS study would also give a clearer picture of how people judge the outputs. Overall, our results suggest that adversarial and perceptual learning meaningfully improve the perceived quality of super-resolved images, even when standard metrics paint a more modest picture.

References

[1] Chao Dong et al. “Image Super-Resolution Using Deep Convolutional Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2016), pp. 295–307. DOI: `10.1109/TPAMI.2015.2439281`.

[2] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 27. 2014, pp. 2672–2680.

[3] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 2015, pp. 448–456.

[4] Alexia Jolicoeur-Martineau. “The Relativistic Discriminator: A Key Element Missing from Standard GANs”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[5] R. G. Keys. “Cubic convolution interpolation for digital image processing”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.6 (1981), pp. 1153–1160. DOI: `10.1109/TASSP.1981.1163711`.

[6] Christian Ledig et al. “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4681–4690. DOI: `10.1109/CVPR.2017.19`.

[7] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations (ICLR)*. 2015.

[8] Xintao Wang et al. “ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.