# SPARSITY FOCUSED SUPER RESOLUTION WITH DEEP LEARNING

**Hershen Nair, Srijan Pal, Vasisht Natamai**
Super Resolutionists
nair0066@umn.edu | pal00036@umn.edu | natam002@umn.edu

## ABSTRACT

This paper explores the hypothesis that integrating explicit sparse representation priors into a deep Super-Resolution CNN through feature map regularization will yield a superior trade-off between pixel fidelity (PSNR and SSIM) and perceptual quality (LPIPS and FID) compared to using either the dense CNN architecture or the classical sparse coding method alone. Moving beyond the limitations of the shallow SRCNN and standard loss functions, we utilize a deeper, residual network architecture. The core innovation is the direct integration of sparse representation by imposing an $L_1$ sparsity constraint on intermediate feature maps. This sparsity constraint compels the network to maintain a high-quality, edge-preserving feature representation, directly addressing the need for a more substantial hybrid model than simple regularization. The model is optimized using a hybrid loss incorporating fidelity and perceptual terms. Current work focuses on implementing Deep Sparse Feature Super-Resolution (DSFSR), which imposes an $L_1$ sparsity constraint on intermediate feature maps to balance fidelity and perceptual quality, with the intent to highlight a superior trade-off between these metrics compared to standard sparse coding or the dense CNN mentioned earlier. An ablation study is also performed to support our hypothesis, which determines the best place to put the $L_1$ loss in our DSFSR Model. All the relevant code and documentation for this project can be found in our GitHub Repository linked here: EE5561_Final_Project.

## 1 INTRODUCTION

The need for High-Resolution (HR) images is critical in fields like medicine, surveillance, and remote sensing. Single Image Super-Resolution (SISR) is the inverse problem of reconstructing a visually correct HR image from a single Low-Resolution (LR) input, requiring effective prior knowledge to succeed.

**Project Motivation and Evolution:** Our initial proposal, which focused on combining the shallow and dated SRCNN architecture with standard $L_1$ regularization, was deemed insufficient in complexity for a graduate-level project. In response, this project evolved to a more advanced, integrated hybrid approach. The primary motivation is to integrate the structural and edge-preserving benefits of sparse feature representation directly into a deeper, residual-based Convolutional Neural Network (CNN). This technical shift ensures greater novelty and a more complex foundation.

**Problem Definition:** This project addresses the challenge of creating a two-stage Super-Resolution system using a fundamental integration of sparse optimization.

- **Final Report Focus (DSFSR) :** Current work at this stage of the project focuses on Deep Sparse Feature Super-Resolution (DSFSR), a model that imposes an $L_1$ sparsity constraint on intermediate feature maps to balance fidelity and perceptual quality. Specifically, this report focuses on determining where, how, and why the $L_1$ sparsity constraint is applied to the intermediate feature maps of the DSFSR model. We support our findings with an ablation study explained in Section 3.
- **Future Work (EDLSR and SR-INR) :** Future work for this project involves the implementation of a more complex End-to-End Dictionary Learning system, where the sparse

priors are learned jointly with the network weights. The parallel objective is to explore SR using a Generative Implicit Prior, where the traditional dictionary is replaced by an Implicit Neural Representation (INR).

The evaluation goal for DSFSR is to demonstrate superior performance across a diverse set of metrics, balancing pixel fidelity (PSNR, SSIM) with perceptual quality (BRISQUE, LPIPS, FID).

## 2 Related Works (Updated from proposal)

### 2.1 Super-Resolution Convolutional Neural Network (SRCNN)

Convolutional Neural Networks (CNNs) are a type of feed-forward neural network trained using gradient descent. CNNs are used a lot in image classification and restoration problems due to their unique characteristics of prioritizing local connectivity and weight sharing mechanisms Xi et al. (2025). The Super Resolution Convolutional Neural Network (SRCNN) established the first end-to-end pipeline, using a pre-upsampling strategy (input up-sampled via Bicubic interpolation) followed by feature extraction, non-linear mapping, and reconstruction.

While the Super-Resolution Convolutional Neural Network (SRCNN) is a crucial step in improving image enhancement, it's not without its limitations. The paper by Ward et al. (2019) identified some core limitations by evaluating SRCNN on metrics of BRISQUE, SSIM, and PSNR.

- SRCNN struggled to reconstruct images degraded with JPEG compression as it isn't trained to correct compression artifacts, leaving those low-resolution artifacts to persist in the final image
- Successive image correcting tends to highlight unnatural edges, leading to an unnatural image.
- Increasing the scaling factor reduces image quality. Lower scaling factors produce clear improvements in sharpening/enhancing the image.

### 2.2 Sparsity

The sparse representation method of super-resolution is a powerful alternative to traditional methods like interpolation. This method, proposed by Yang et al. (2010), is that small image patches can be well-represented by a sparse linear combination of elements from an over-complete dictionary learned from training data. These elements include features like edges and patterns. The algorithm is as follows, two dictionaries ($D_l$ and $D_h$) represent elements from both low-resolution and corresponding high-resolution images, respectively, from the training data. These dictionaries are trained jointly such that the same sparse coefficient ($\alpha$) describes corresponding structures in both the low- and high-resolution patches. Once trained, the algorithm finds the sparse coefficient, $\alpha$, of each low-resolution patch from $D_l$ and uses it to index $D_h$ to get the corresponding high-resolution features and reconstruct the high-resolution patch. Finally, these patches are reconstructed to form a high-res version of the original low-res image. This method replaces the need for large patch databases with a compact learned representation, letting us recover high-frequency components like edges and textures more efficiently. It also ensures these edges and textures are sharper compared to many traditional methods. One limitation is that dictionary training is computationally expensive.

### 2.3 Very Deep Super Resolution (VDSR)

VDSR is a highly accurate single-image super-resolution (SR) method proposed by Jiwon Kim, Jung Kwon Lee and Kyoung Mu Lee in 2016. It uses a very deep convolutional network inspired by the VGG-net architecture. It addresses the limitations of earlier, shallower CNNs like SRCNN by increasing network depth to 20 weight layers Kim et al. (2015). The core idea of VDSR is that increasing network depth allows the model to utilize more contextual information and boost accuracy.

- **Utilize More Contextual Information:** By cascading small $3 \times 3$ filters many times, the network exploits contextual information over very large image regions (a large receptive field), which is crucial for detail recovery in the SR problem.

- **Boost Accuracy:** The depth shows a significant improvement in accuracy over shallow models. VDSR outperformed SRCNN by a large margin Kim et al. (2015).

### 2.4 End-To-End Super Resolution (SRDD)

The SRDD paper proposes an End-to-End Super-Resolution Network with a Deep Dictionary (SRDD), aiming to fuse traditional sparse-coding methods with deep learning Maeda (2022). Historically, sparse coding explicitly learned high/low-resolution dictionaries, a multi-step process that the SRCNN replaced entirely with implicit dictionary learning via a multi-layered CNN. The core motivation for SRDD is that conventional deep-learning methods, including state-of-the-art ones, suffer significant performance degradation on out-of-domain images because their implicitly learned HR dictionaries are "fragile" to differences in input degradation. SRDD solves this by explicitly learning the high-resolution dictionary $D_H$ using a generator network, where a separate main network predicts the coefficients ($\alpha$) for $D_H$ to reconstruct the imageMaeda (2022).

### 2.5 Sparsity Driven Compressed Implicit Neural Representations (SINR)

The SINR paper provides a key insight for SR models using Implicit Neural Representations (INR) as a generator Jayasundara et al. (2025). INRs are simple neural networks that define an image as a continuous function, freeing SR from fixed resolutions. The SINR concept shows that the weights of these INRs are naturally compressible, similar to how images are compressed, because the weights tend to follow a Gaussian distributionJayasundara et al. (2025). Using Compressed Sensing (CS), the INR's large weight vectors ($\mathbf{w}$) can be turned into a much smaller, sparse code ($\mathbf{x}$), requiring far less storage. This is achieved using a Random Sensing Matrix ($\mathbf{A}$) that the receiver can recreate using a simple seed, meaning no extra data needs to be sent. For SR models, this confirms that making the latent code ($\mathbf{z}$) sparse is effective, and, more importantly, it offers a path to building an INR generator ($\mathbf{G}_\theta$) that is highly compact and efficient, reducing the model's size significantly without losing image quality.

## 3 Methodology (updated)

### 3.1 Replication of Current Methods

To establish a foundational comparison, we began by replicating key incremental Super-Resolution (SR) models, evolving from the shallow SRCNN to include crucial architectural and regularization techniques. All baseline models were trained for 50 epochs using a consistent setup to expedite comparison, leveraging pre-trained SRCNN weights from an open-source source project Chandrasekhar (2023).

1. **Incremental Baseline Development**

| Baseline Model | Architecture | Core Improvement | Loss Function | Optimizer |
|---|---|---|---|---|
| SRCNN Base | 3-Layer CNN | NA | Standard MSE Loss | SGD |
| SRCNN + Sparsity | 3-Layer CNN | $L_1$ Regularization (on weights) | MSE + $L_1$ Penalty | AdamW |
| SRCNN + Residual | 3-Layer CNN | Residual Learning: Learning difference between LR and HR image | MSE on Residual | AdamW |
| SRCNN + Sparsity + Residual | 3-Layer CNN | Combines Residual Learning and $L_1$ regularization | MSE on Residual + $L_1$ Penalty | AdamW |

2. **Implementation Details**
   - All models were trained for 50 epochs. The SRCNN base model used SGD for historical fidelity, while all subsequent models utilized the AdamW optimizer.
   - We selected AdamW due to its adaptive learning rate and effective weight decay decoupling, which ensures the $L_2$ regularization term remains independent and effective against the adaptive scaling, unlike standard Adam.
   - In models incorporating a residual connection, the network learns the residual image (HR - LR) directly, significantly improving training stability and performance over learning the full HR mapping.

### 3.2 VERY DEEP SUPER RESOLUTION (VDSR)

The VDSR model uses a deep, 20-layer CNN architecture pulled from the VDSR paper Kim et al. (2015).

- **Input Preparation:** The low resolution input image is up scaled (using bicubic interpolation) to the desired high-resolution size before being fed into the network.
- **Deep Feature Mapping:** The network consists of a 20 layer stack of Conv2D (3x3 kernel, 64 channel) and ReLU layers. The depth allows the model to learn highly complex and non-linear mappings to capture a larger receptive field.
- **Global Residual Connection:** The VDSR uses a global residual connection, basically the input image is fed back in after the 20-layer network. The loss function is the standard MSE between the learned and true residual.
- The model is run for 50 epochs with AdamW as the optimizer. A scheduler is used to adjust the learning rate at 30 and 40 epochs by scaling the LR factor by 0.1. This helps the model avoid falling into local minima or a plateau.

### 3.3 DSFSR: DEEP SPARSE FEATURE SUPER-RESOLUTION (COMPLETED WORK)

To create a novel and focused contribution, we implement Deep Sparse Feature Super-Resolution (DSFSR), a hybrid approach that integrates sparsity constraints directly into the internal feature representations of a deep SR network. Instead of using the shallow SRCNN architecture, we build DSFSR on top of an 18-layer VDSR model to leverage the larger receptive field and improved representation capacity of deep CNNs.
Unlike standard regularization that applies an L1 penalty on network weights or output images, DSFSR introduces a sparsity regularization term ($\mathcal{L}_{sparsity}$) on selected intermediate feature maps ($\mathbf{F}_i$) inside the network. The goal is to encourage sparse activations within the network, so that only important structures such as edges and textures produce strong responses, while smooth regions remain mostly inactive. The total loss function is defined as:

$$\mathcal{L}_{Total} = \mathcal{L}_{Reconstruction} + \lambda \cdot ||\mathbf{F}_i||_1$$

Unlike classical sparse coding, which enforces sparsity on linear combination coefficients, DSFSR enforces sparsity on intermediate activation maps after nonlinear transformations, encouraging selective feature responses rather than sparse linear representations.

1. **Extracting the Feature Map:** During the forward pass, the network outputs both the predicted residual image and intermediate feature maps. These feature maps correspond to the activations after the ReLU operation.
2. **Calculating the L1 Sparsity Loss:** The sparsity loss is computed by taking the L1 norm of the selected feature map(s), defined as the mean of the absolute values of all activations in the feature tensor. This penalty encourages many activations to be close to zero, resulting in sparse internal representations.
3. **Combining the Total Loss:** The total training loss is computed as the sum of the reconstruction loss (MSE on the predicted residual) and the sparsity regularization term weighted by a scalar $\lambda$. During backpropagation, gradients from both loss terms jointly update the network weights, enforcing sparsity in the internal feature representations while preserving reconstruction accuracy.

4. **Ablation Study on L1 Placement:** An ablation study is performed to determine where sparsity regularization is most effective within the 20-layer VDSR architecture. We evaluate six DSFSR variants by applying the L1 sparsity loss at different depths: the 5th layer, 10th layer, 18th layer, 5 & 10th layers, 10 & 18th layers, and 5 & 10 & 18th layers.

   The 5th layer represents early-stage features that capture low-level image information such as edges and gradients. The 10th layer corresponds to mid-level features where more complex textures and structures emerge. The 18th layer represents late-stage features that are closely related to the final reconstruction. By comparing these configurations, we aim to study how sparsity affects different levels of feature abstraction.
5. **Expected Outcome of the Ablation Study:** We expect that applying sparsity to very early layers may negatively affect performance, since early feature maps contain both low-frequency and high-frequency information and are not naturally sparse. In contrast, mid and late layers are expected to benefit more from sparsity regularization, as these layers primarily encode higher-level textures and structural details that are naturally sparse in images. Applying sparsity to multiple layers may further strengthen this effect but could also lead to over-regularization. This ablation study allows us to empirically identify the most effective placement of sparsity regularization in DSFSR.
6. **Ablation Study on L1 Penalty Strength ($\lambda$):** After identifying the most effective layer (10th layer) for applying sparsity, we perform an additional ablation study on the sparsity weight $\lambda$. We evaluate multiple values of $\lambda$ in the range $[1 \times 10^{-4}, 3 \times 10^{-3}]$. Smaller values of $\lambda$ impose a weak sparsity constraint and result in behavior similar to the VDSR baseline, while larger values increasingly suppress feature activations. We observe that increasing $\lambda$ improves some quantitative metrics but can also lead to smoother reconstructions, indicating over-regularization. We expect this ablation to highlight that $\lambda$ controls the balance between preserving texture details and enforcing sparse internal representations.

## 3.4 Future Work

### 3.4.1 EDLSR

One future objective is to implement a robust End-to-End Dictionary Learning system, conceptually derived from SRDD Maeda (2022), where the sparse HR dictionary ($\mathbf{D}_H \in \mathbf{R}^{C \times N \times s^2}$) is explicitly learned jointly with the network weights.

- Design a network that functions as a coefficient predictor, $\mathbf{E}_\phi$, which processes the LR input ($\mathbf{X}_{\text{LR}}$) to output the sparse coefficient map ($\boldsymbol{\alpha} \in \mathbf{R}^{N \times H \times W}$).
- The HR dictionary $\mathbf{D}_H$ is implemented as a set of learnable parameters (nn.Parameter) or generated by a sub-network (such as in SRDD).
- The final HR image ($\mathbf{Y}_{\text{HR}}$) is reconstructed via a weighted linear combination of the dictionary atoms:

$$\mathbf{Y}_{\text{HR}} = \sum_{n=1}^{N} \mathbf{D}_{H,n} \otimes \boldsymbol{\alpha}_n$$

- The system is optimized end-to-end to ensure the sparse priors are learned and specialized for the SR task.

### 3.4.2 SR-INR

A parallel objective is to explore SR using a Generative Implicit Prior, where the traditional dictionary is replaced by an Implicit Neural Representation INR $\mathbf{G}_\theta$ borrowed from the SINR paper concepts Jayasundara et al. (2025).

- Encoder Block ($\mathbf{E}_\phi$): A CNN (based on VDSR) acts as an encoder, processing the LR image ($\mathbf{X}_{\text{LR}}$) to predict a highly compact latent code ($\mathbf{z} \in \mathbf{R}^{D_{\text{latent}}}$).
- INR Block ($\mathbf{G}_\theta$): The INR is a small MLP that uses the latent code $\mathbf{z}$ as its continuous, non-linear prior to generate the HR image patch ($\mathbf{Y}_{\text{patch}}$). The INR is defined as a continuous function $f_\theta(\mathbf{c}, \mathbf{z}) \rightarrow \mathbf{v}$, where $\mathbf{c}$ is the coordinate and $\mathbf{v}$ is the signal value.

- Sparsity Objective: The method requires enforcing sparsity directly on the latent code $\mathbf{z}$ via an $\mathbf{L1}$ regularization term in the overall loss function ($\mathcal{L}_{\text{sparsity}} = \lambda||\mathbf{z}||_1$).

### 3.5 Score Metrics

To evaluate the different SR models, we use a combination of fidelity-based metrics (PSNR, SSIM) and perceptual metrics (BRISQUE, LPIPS, FID).

- **Fidelity -based Metrics: PSNR** (Peak Signal-to-Noise Ratio) measures pixel-level reconstruction accuracy. Higher PSNR indicates better fidelity. **SSIM** (Structural Similarity Index) compares structural information and is more perceptually relevant than PSNR.
- **BRISQUE** (Blind/Referenceless Image Quality Score) is a no-reference metric that evaluates how natural an image looks. Lower scores indicate more realistic textures. In SR, BRISQUE helps detect oversmoothing, which is common in SRCNN and VDSR.
- **LPIPS** (Learned Perceptual Image Patch Similarity Zhang et al. (2018)) measures perceptual distance between SR and HR images using deep features from a trained network (VGG/AlexNet). Lower scores indicate closer perceptual similarity. In SR, LPIPS is an effective indicator of whether fine textures and high-frequency details are preserved.
- **FID** (Frechet Inception Distance Heusel et al. (2017)) compares the feature distributions of an entire set of SR images to the HR ground-truth set using an Inception network. Lower scores indicate that the SR outputs collectively look closer to real high-resolution images.

Evaluation: All metrics are computed on the same 150-image set. PSNR, SSIM, BRISQUE, and LPIPS are computed per image, while FID is calculated once per model as a set statistic.

### 3.6 Google CoLab

All implementations will be run on Google Colab as it provides ideal Python support and GPU access. Our models will be run with either the Nvidia A100, L4, and T4 GPUs. The specific type depends on its performance and availability on Colab.

## 4 Current Results

### 4.1 Test Images

Figures 1 and 3 shows the visual output of all the SR models on test images compared with their corresponding HR ground truth and LR input images.

### 4.2 Violin Plots

Figures 2 and 4 present violin plots of PSNR, SSIM, BRISQUE, and LPIPS scores for each model. These plots visualize both the distribution of metrics and the spread/variance of model performance across the dataset. Figure 4 shows the distribution specifically for the DSFSR models, highlighting the effect of feature-map sparsity on the overall metrics.
**Interpretation:** A wide violin shape indicates high variability between images, while a narrow or peaked shape indicates consistent performance.

## 5 Discussion and Conclusion

### 5.1 Ablation Study Recap

The focus of this report is determining the best placement for the L1 regularizer on intermediate feature maps. Our ablation study will determine where in the 20 layer VDSR model is best to put the L1 regularizer loss. One model places L1 regularizer loss after the 5th layer. A snippet of code in the forward pass logic allows us to determine where we want to place the L1 loss; It is shown on the following page.

```
# Iterate through the sequential module to get the intermediate feature
    map
feature_map_i = x
for i, module in enumerate(self.residual_layers):
    feature_map_i = module(feature_map_i)

    # Assuming an 20-layer VDSR
    # The 5th layer is technically the 10th item since each layer has
        both conv and ReLU (2*5-1 = 9)
    if i == 9:
        # Save the intermediate feature map F_i (This is the key for
            DSFSR)
        intermediate_feature_map = feature_map_i

#The output of the last ReLU in the residual layers
x = feature_map_i
```

Here are the 6 DSFSR Models we will compare: 5th Layer, 10th Layer, 18th Layer, 5th & 10th Layer, 10th & 18th Layer, 5th & 10th & 18th Layer. After determining the most suitable for introducing sparsity, we conduct an additional ablation study on the sparsity weight $\lambda$, testing values in the range $[1 \times 10^{-4}, 3 \times 10^{-3}]$.

### 5.2 DSFSR Ablation Study Results

To analyze the effect of sparsity placement, we compare the mean metric scores in Table 1 together with the corresponding violin plots in Figure 4 for six DSFSR variants, where the L1 sparsity regularization is applied at different layers of the 20-layer VDSR model. Based on this comparison, we therefore fix the sparsity placement at one layer and perform an additional ablation study by sweeping the sparsity weight $\lambda$ to understand its impact on model behavior.

- Overall Impact of Sparsity Placement: DSFSR variants perform similarly to VDSR, showing that sparsity acts as a mild regularizer without significantly harming reconstruction quality or destabilizing training.
- L1 Loss at Early Layer (5th): Applying sparsity at the 5th layer results in slightly improved BRISQUE and LPIPS but no clear gain in PSNR or SSIM. This suggests that early feature maps, which contain a mix of low-frequency and high-frequency information, do not strongly benefit from sparsity constraints.
- L1 Loss at Mid-Layer (10th): Applying sparsity at the 10th layer yields the best overall trade-off among the tested configurations. This variant achieves the highest mean PSNR (35.17 dB) and the lowest mean LPIPS (0.1106), indicating improved perceptual similarity while maintaining strong reconstruction fidelity. This supports the intuition that mid-level feature maps primarily encode textures and structural patterns that are naturally sparse.
- L1 Loss at Later Layer (18th): Sparsity applied at the 18th layer produces comparable PSNR and SSIM to the mid-layer variant, while achieving the lowest FID. This suggests that sparsifying late-stage features may slightly improve the generated images globally, though the perceptual gains are less consistent than those observed for mid-layer sparsity.
- Multi-layer Sparsity (5 & 10, 10 & 18, 5 & 10 & 18): Applying sparsity to multiple layers does not lead to further improvements and in some cases, slightly degrades perceptual metrics such as LPIPS and FID. This indicates that enforcing sparsity across too many stages of the network may over-regularize the model and limit its representational flexibility.
- Violin Plot Analysis: The violin plots in Figure 4 show that all DSFSR variants produce tightly clustered PSNR and SSIM distributions, indicating consistent reconstruction quality across the dataset. Differences between variants are more visible in LPIPS and BRISQUE, where the mid-layer sparsity model exhibits a slightly more compact distribution, suggesting more consistent perceptual behavior.
- Qualitative Results: Visually, the HR outputs of test images across different sparsity placements in Figure 3 reveals only subtle differences, which is expected given the strong VDSR

baseline. However, models with sparsity applied at the mid and late layers tend to produce slightly cleaner textures and fewer minor artifacts, aligning with the small but consistent improvements observed in LPIPS and FID scores.

- Results on sweeping the L1 Penalty Strength ($\lambda$): Based on the overall performance, applying sparsity at the mid-layer (10th layer) consistently provides the best balance between reconstruction fidelity and perceptual quality. We therefore fix the sparsity placement at the 10th layer and sweep the sparsity weight $\lambda$ over a range of values ($1 \times 10^{-4}$ to $3 \times 10^{-3}$). For smaller values of $\lambda$ ($3 \times 10^{-4}$ and $5 \times 10^{-4}$), the DSFSR model shows behavior close to the VDSR baseline, with only minor changes in metric scores. As $\lambda$ is increased to $1 \times 10^{-3}$, we observe consistently lower FID across two independent runs, indicating that stronger sparsity encourages outputs that are closer to the HR dataset. However, this improvement is not consistently reflected in other metrics, and larger values of $\lambda$ do not lead to further gains. Overall, this ablation suggests that the sparsity weight primarily controls a trade-off between conservative, distribution-aligned reconstructions and preserving fine texture details, with moderate values of $\lambda$ providing stable performance and larger values leading to diminishing returns. Unlike dropout, which randomly suppresses activations during training, the proposed L1 regularization deterministically encourages sparsity in feature activations and remains active during inference.

### 5.3 Difficulties

- A single greatest difficulty in constructing the CNNs, specifically for VDSR and DSFSR, was ensuring proper dimensionality changes between input to output and to testing later on. A lot of debugging had to be done to make sure the images were in the proper tensor format to input to the network and then brought back to the correct format for testing. Preprocessing and post-processing took more time than the actual network for some of the CNNs!
- GPU access in Google Colab was limited due to only a set amount of compute units available. So at times we had to use the CPU, which was more time-intensive.
- Formatting images for submission became difficult due to Overleaf formatting constraints. One suggestion that we tried was putting all our results on a single row (Ground Truth, Low-Res image, model 1, model 2, ...). We attempted to reformat 1 and 3, however, placing more than 6 images in a row led to the images either being too small or too blurry to actually make out the super-resolved effect.
- Due to the continuous changes our project went through,it was difficult to expand too deep into a single model or focus. We had more trouble finding a consistent narrative for our project since the notion of sparsity in SR tasks is already developed in recent literature. We settled on the DSFSR model since we had not found any existing ablation study or sufficient research on where exactly sparsity penalty had the best effect (in regards to feature maps). This idea was also influenced by the TA and Professor feedback, which greatly helped this project fit into a consistent narrative.

### Work Done and Lessons Learned by Each Member

**Hershen Nair:** Analyzed Sparse Representation model. Write-ups for Abstract, Sections 1, 2, 3.1, 3.8, 5.1, 5.3, Updates to Sections 3.3 and 3.4. This project helped me deepen my understanding of how Sparse Representation works along with dictionary training to construct high-resolution images from low-resolution ones. I also learned a great deal about the importance ablation studies, especially how varying carefully chosen ablations can give meaningful insights on a model's behavior under different conditions.
**Srijan Pal:** Score metrics implementation and ablation study of DSFSR model. Write-ups for Sections 3.3, 3.5, 4, 5.2. This project helped me improve my understanding of ablation studies, result analysis, and how they are used together to understand model behavior and performance.
**Vasisht Natamai:** Developing and implementing DSFSR model. Write-ups for sections 3.2, 3.3, 3.4, 3.5, 3.6, 5.2. This project really helped me learn the impact of sparsity on deep neural networks. It also helped me understand how to conduct robust ablation studies to help determine a correlation between variables.

## REFERENCES

Aditya Chandrasekhar. Image super resolution. https://www.kaggle.com/datasets/adityachandrasekhar/image-super-resolution/data, 2023. Kaggle dataset.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Dhananjaya Jayasundara, Sudarshan Rajagopalan, Yasiru Ranasinghe, Trac D. Tran, and Vishal M. Patel. Sinr: Sparsity driven compressed implicit neural representations, 2025. URL https://arxiv.org/abs/2503.19576.

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. *CoRR*, abs/1511.04587, 2015. URL http://arxiv.org/abs/1511.04587.

Shunta Maeda. Image super-resolution with deep dictionary, 2022. URL https://arxiv.org/abs/2207.09228.

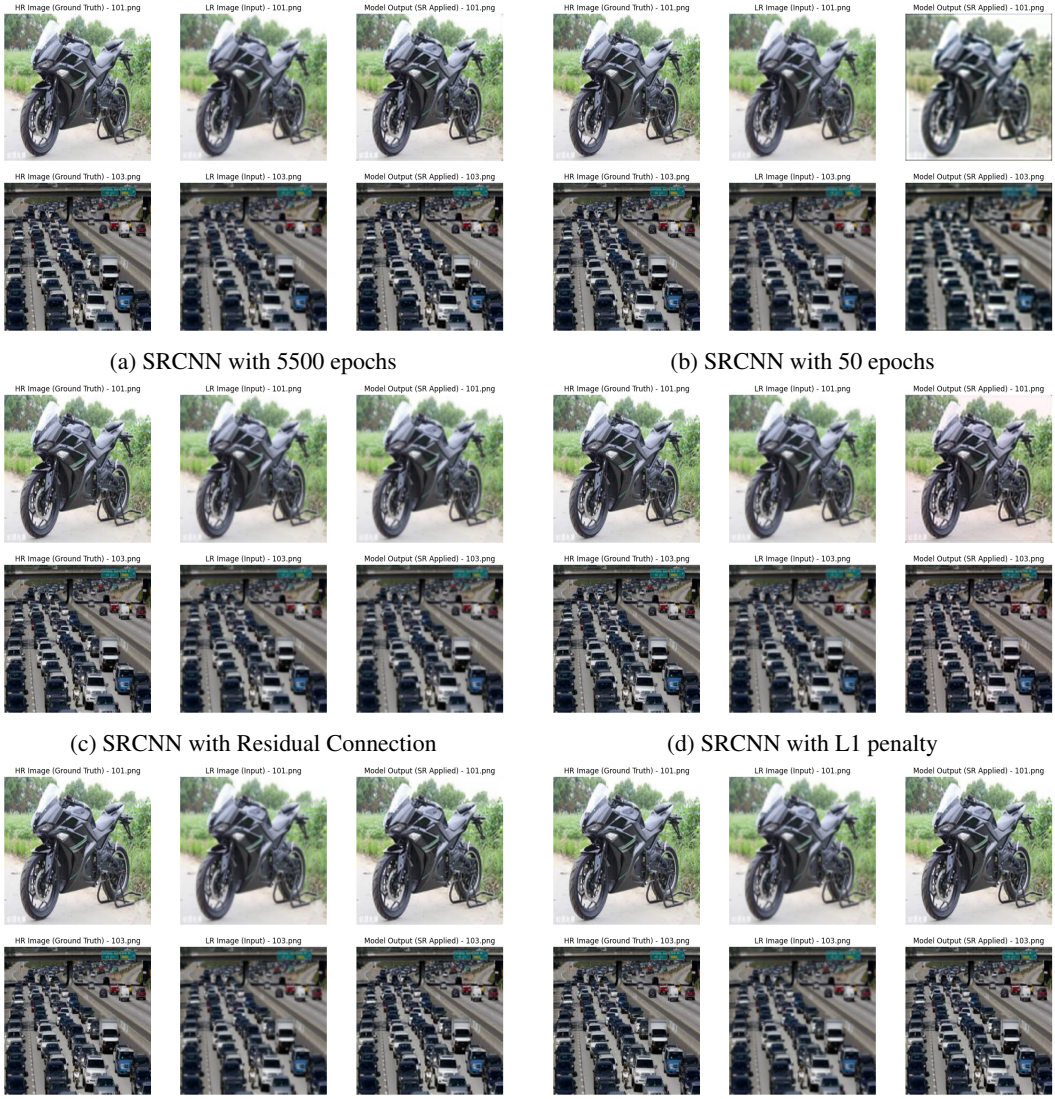
Chris M Ward, Josh Harguess, Brendan Crabb, and Shibin Parameswaran. Image quality assessment for determining efficacy and limitations of super-resolution convolutional neural network (srcnn). *arXiv preprint arXiv:1905.05373v1*, 2019.

Wenqiang Xi, Zairila Juria Zainal Abidin, Cheng Peng, and Tadiwa Elisha Nyamasvisva. A review of deep learning-based image super-resolution reconstruction methods. *Journal of Computing and Electronic information Management*, 17, 06 2025. doi: 10.54097/phfrck02.

Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010. doi: 10.1109/TIP.2010.2050625.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. doi: 10.1109/CVPR.2018.00068.

# 6 APPENDIX




(a) SRCNN with 5500 epochs

(b) SRCNN with 50 epochs

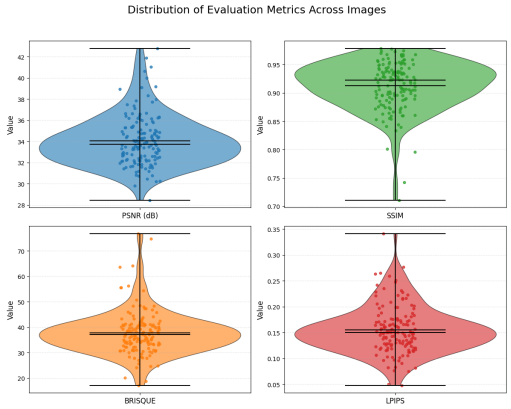
(c) SRCNN with Residual Connection

(d) SRCNN with L1 penalty

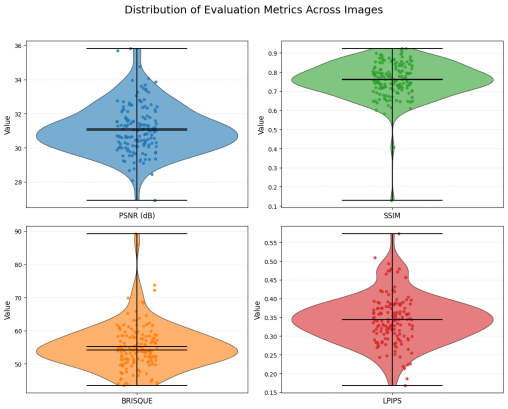
(e) SRCNN with Residual and L1 Penalty

(f) VDRS

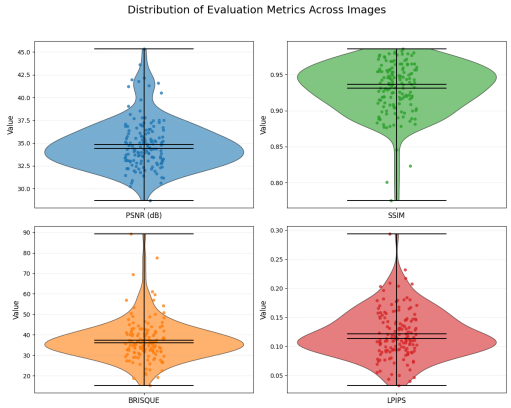
Figure 1: Test Images by Baseline Model



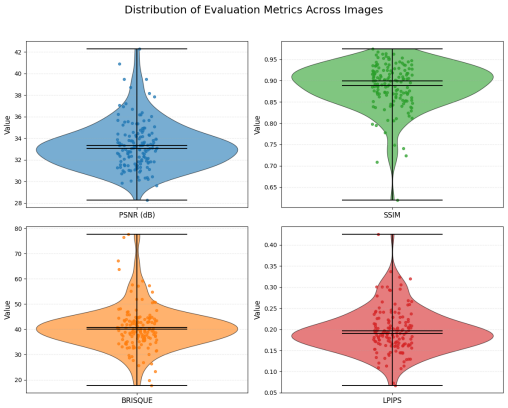

(a) SRCNN with 5500 epochs



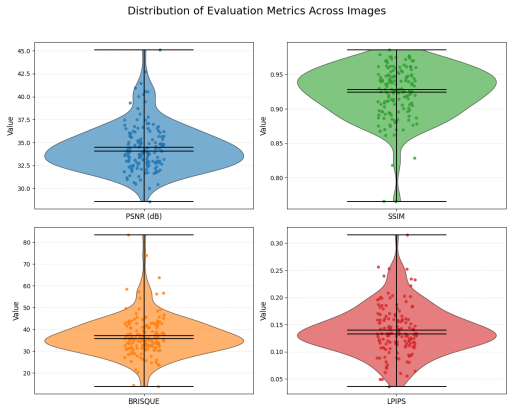

(b) SRCNN with 50 epochs



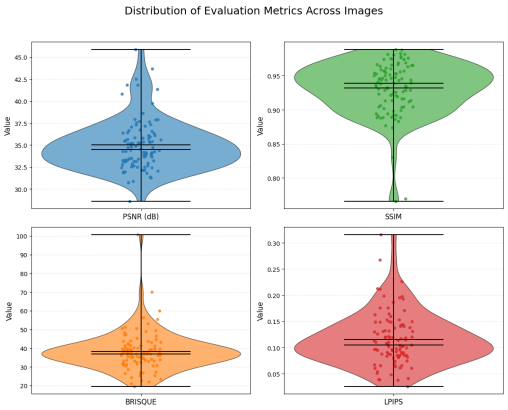

(c) SRCNN with Residual Connection




(d) SRCNN with L1 penalty




(e) SRCNN with Residual and L1 Penalty




(f) VDRS

Figure 2: Violin Plots by Baseline Model


(a) DSFSR (5th Layer L1 Penalty)


(b) DSFSR (10th Layer L1 Penalty)

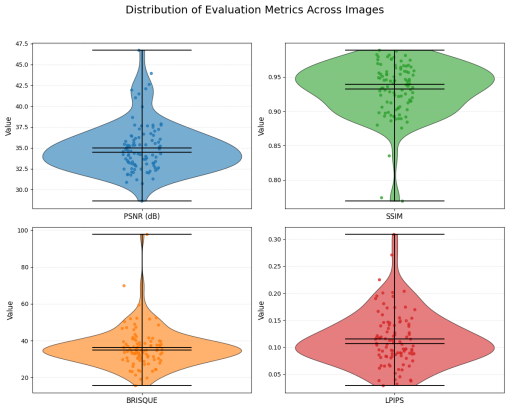

(c) DSFSR (18th Layer L1 Penalty)


(d) DSFSR (5th and 10th Layer L1 Penalty)

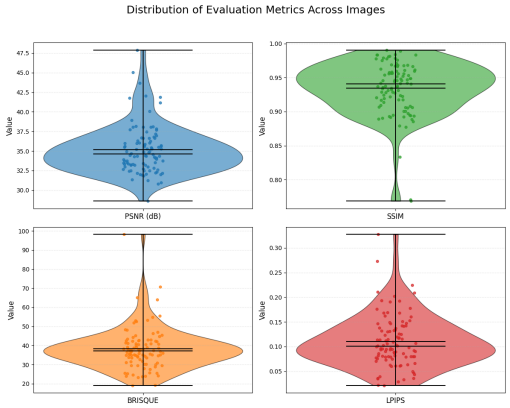

(e) DSFSR (10th and 18th Layer L1 Penalty)


(f) DSFSR (5th, 10th, and 18th Layer L1 Penalty)

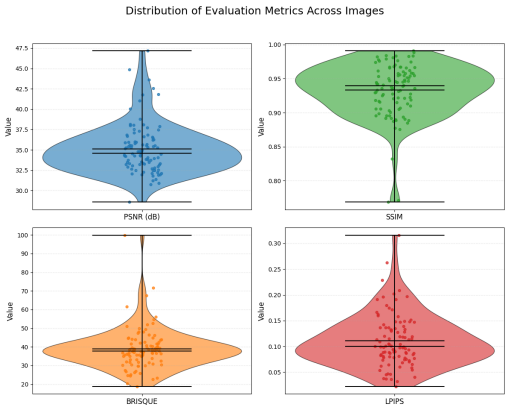
Figure 3: Test Images by DSFSR Model



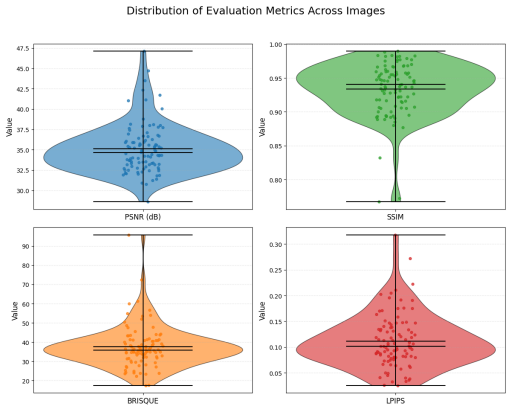

(a) DSFSR (5th Layer L1 Penalty)



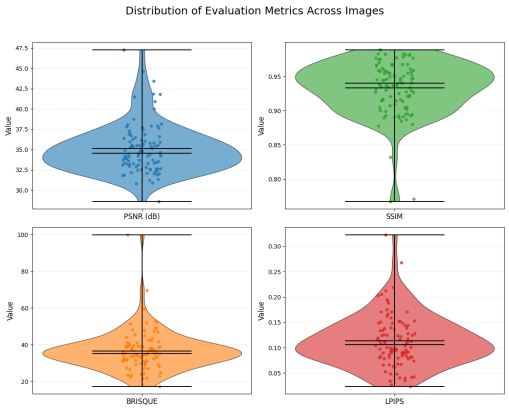

(b) DSFSR (10th Layer L1 Penalty)



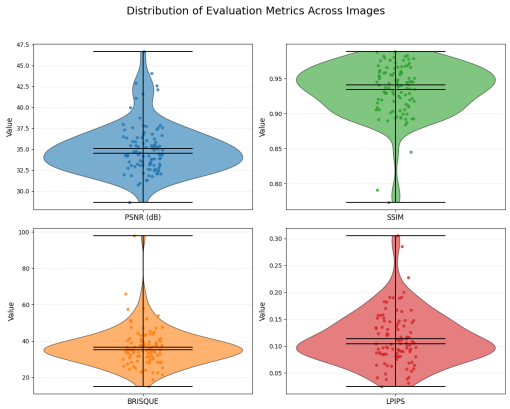

(c) DSFSR (18th Layer L1 Penalty)




(d) DSFSR (5th and 10th Layer L1 Penalty)




(e) DSFSR (10th and 18th Layer L1 Penalty)




(f) DSFSR (5th, 10th and 18th Layer L1 Penalty)

Figure 4: Violin Plots by DSFSR Model

| Model | Mean PSNR (dB) | Mean SSIM | Mean BRISQUE | Mean LPIPS | FID (HR set vs SR set) |
|---|---|---|---|---|---|
| SRCNN 5500 Epochs | 34.065 | 0.9128 | 37.82 | 0.1556 | 62.236 |
| SRCNN 50 Epochs | 31.101 | 0.7592 | 55.19 | 0.3437 | 129.732 |
| SRCNN + Residual | 34.866 | 0.9311 | 37.33 | 0.1217 | 51.166 |
| SRCNN + Sparsity | 33.331 | 0.8883 | 40.72 | 0.1964 | 75.431 |
| SRCNN + Residual + Sparsity | 34.476 | 0.9239 | 37.06 | 0.1396 | 57.432 |
| VDSR | 34.995 | 0.9318 | 38.35 | 0.1159 | 78.279 |
| DSFSR (5th Layer) | 35.007 | 0.9324 | 36.22 | 0.1150 | 77.821 |
| DSFSR (10th Layer) | 35.167 | 0.9343 | 38.25 | 0.1106 | 77.455 |
| DSFSR (18th Layer) | 35.119 | 0.9333 | 38.90 | 0.1111 | 76.479 |
| DSFSR (5th and 10th Layer) | 35.132 | 0.9339 | 37.54 | 0.1117 | 77.124 |
| DSFSR (10th and 18th Layer) | 35.091 | 0.9333 | 36.70 | 0.1140 | 78.112 |
| DSFSR (5th, 10th, and 18th Layer) | 35.067 | 0.9345 | 36.56 | 0.1141 | 79.467 |

Table 1: Evaluation Metrics Table

| $\lambda$ at DSFSR 10th Layer | Mean PSNR (dB) | Mean SSIM | Mean BRISQUE | Mean LPIPS | FID (HR set vs SR set) |
|---|---|---|---|---|---|
| 3e-3 | 35.208 | 0.9374 | 37.63 | 0.1103 | 77.024 |
| 1e-3 | 35.196 | 0.9351 | 37.55 | 0.1112 | 45.214 |
| 5e-4 | 35.029 | 0.9327 | 36.10 | 0.1135 | 77.577 |
| 3-e4 | 35.076 | 0.9339 | 36.43 | 0.1135 | 77.897 |
| 1e-4 | 35.167 | 0.9343 | 38.25 | 0.1106 | 77.455 |

Table 2: Evaluation Metrics Table