Single Image Super Resolution using an Efficient Transformer-based Sub-Pixel Convolution Neural Network

Md Asif Tanvir
Department of Computer Science
Missouri State University
Springfield, MO, USA
mt5864s@missouristate.edu

Sher Yar Karim
Department of Computer Science)
Missouri State University
Springfield, MO, USA
sk3746s@MissouriState.edu

***Abstract*—Image restoration is a very popular and old problem in the domain of computer vision. The goal is to restore a higher resolution of an image from a lower resolution and noisier image. This method is also referred to as single-image super-resolution. It has useful impacts in multiple domains like the medical sector, surveillance, and satellite image enhancements. While there are super-resolution methods using a convolutional neural network, with the introduction of transformers, there are multiple super-resolution model that uses the transformer architecture. But one pivotal problem with the transformer models is that they are computationally expensive, creating the opportunity for providing an optimized transformer-based solution for this problem. In this project, we propose an efficient transformer-based hybrid architecture, ETSPCN, to generate higher-resolution images from lower-resolution images. Although the initial results are not great, there are noticeable improvements in the computational performance. This can pave the way for future research in building efficient transformers for image super-resolution.**

***Index Terms*—Image restoration, Efficient Transformers, Pixel-Shuffle**

I. Introduction

Single Image Super-Resolution (SISR) focuses on recovering a high-resolution (HR) image from a single low-resolution (LR) version. Even though this task appears simple at first glance, it is one of the more challenging problems in computer vision. The difficulty arises from the fact that the LR image contains limited visual information, and the model must reconstruct details that are either weak or completely missing. This makes SISR an important and widely studied topic, especially because many real-world applications rely heavily on image clarity.

SISR plays a meaningful role in several areas. In medical imaging, enhancing resolution can help identify patterns that may not be visible in low-quality scans. In satellite and aerial imaging, weather conditions, distance, and sensor limitations often lead to low-resolution data that needs improvement before it becomes usable. Surveillance footage also frequently requires enhancement, especially when the original video is compressed or captured under poor lighting. In consumer photography, old or low-quality images often benefit from restoration. Because these fields depend heavily on visual detail, even a moderate improvement in clarity can lead to significant practical value.

Before deep learning became popular, researchers mainly used interpolation methods such as nearest-neighbor, bilinear, or bicubic upsampling. These techniques are simple and fast, but they usually produce blurred and unrealistic results because they do not actually restore high-frequency information. Later methods explored dictionary learning and sparse representations, where the idea was to match LR patches to HR patches. Although these approaches were more advanced than interpolation, they required engineered pipelines, heavy computation, and often failed to generalize well to different image types.

Deep learning introduced a major shift in how SISR models were designed. Convolutional neural networks (CNNs) showed that models could learn the LR-to-HR mapping directly from large image datasets. Architectures like ESPCN [1] introduced efficient ideas such as the pixel-shuffle operation, which avoided the artifacts often seen with transposed convolutions. These CNN-based approaches produced much sharper outputs than earlier techniques. However, CNNs operate mainly through local kernels, which limits their ability to understand global patterns in an image. In many super-resolution tasks, global context matters. For example, a repeated pattern in a texture or structural relationship between different parts of an object might be important for rebuilding fine details.

This limitation led to the introduction of transformer-based models into the super-resolution domain. Transformers, which originally became popular in natural language processing, use attention mechanisms that allow the model to compare distant regions of an image. Models such as ESRT [3] and SwinIR [2] adapted this idea for vision tasks and achieved strong results. They demonstrated that capturing long-range relationships helped the model generate sharper edges, more stable textures, and better structural consistency. However, these transformer models tend to be large, memory-intensive, and computationally expensive. They often require powerful GPUs to train and may not be suitable for deployment on lightweight devices.

Our goal in this project was to explore whether it is possible

to benefit from transformer-style global reasoning without relying on a heavy or complex model. Instead of building a deep transformer architecture, we focused on designing a compact model that is easier to train and experiment with. We built a hybrid approach that combines Efficient Transformer blocks with the lightweight pixel-shuffle upsampling method from ESPCN [1]. The Efficient Transformer blocks reduce computational cost by compressing features before attention is applied, allowing us to retain some of the transformer's advantages without the full computational burden.

The motivation behind this design was not to outperform large transformer models, but to develop a transformer-based model for SISR in a simplified and efficient setting. We wanted to experiment whether such blocks could still help the model learn useful patterns and whether the combination with pixel-shuffle would lead to any noticeable improvements. Our experiments show that while the performance does not reach the level of modern state-of-the-art models, the hybrid model significantly reduces parameters and computation. This makes it a promising approach for scenarios where resources are limited or where the model needs to run on low-power hardware.

Overall, this project allowed us to explore the balance between model complexity and practical efficiency. It also helped us understand how transformer-based ideas can be adapted into smaller architectures. The insights gained from this work may contribute to future studies focused on building lightweight super-resolution models that combine global understanding with efficient upsampling.

II. Methodology

Figure 1 demonstrates the architecture flow of our proposed model, ETSPCN. The design of ETSPCN model was guided by a simple idea: try to capture some of the benefits of transformer-based architectures while keeping the overall structure lightweight and easy to train. Instead of building a large model with many layers, we focused on creating a clear and compact pipeline that combines the strengths of convolutional operations, efficient transformer concepts, and pixel-shuffle upsampling. The architecture consists of five main components: shallow feature extraction, Efficient Transformer blocks, a refinement layer, pixel-shuffle upsampling, and a residual branch that helps preserve structural information. Each part plays a specific role, and together they form a model that balances performance and efficiency.

A. Shallow Feature Extraction

We begin by applying a single convolution layer of 4×4 to the input image. This layer increases the number of channels from three (RGB) to 64, which allows the model to represent more abstract visual features. Even though this step is simple, it is important because it gives the model a basic understanding of edges, corners, and textures before the more complex operations take place. Many deep networks use several layers at this stage, but we wanted the model to remain lightweight. During development, we observed that this single convolution was enough to provide stable features for the next stage.

B. Efficient Transformer Blocks

The output of the shallow extraction layer is passed through multiple Efficient Transformer blocks. These blocks form the core of the model. Traditional transformer layers, like SwinIR [2], apply full self-attention, which is computationally expensive. To make the process more efficient, we use a reduced version where the number of channels is temporarily lowered before attention is computed. This helps decrease the memory required for attention and speeds up the computation. This architecture is taken from the paper ESRT [3].

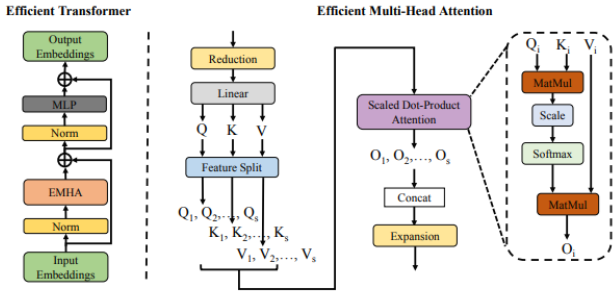

Fig. 2: Efficient Transformer block structure used in our model from the paper ESRT [3]

Figure 2 demonstrates the Efficient Transformer block structure. First, the input is normalized to stabilize the learning process. Then the number of channels is reduced, and multi-head attention is applied to this compressed representation. After attention, the features are projected back to the original number of channels. A small feed-forward network processes the output further, adding nonlinearity and helping the model learn more complicated patterns. Residual connections are used around both the attention and the feed-forward parts to prevent the model from losing information and to make training more stable. By stacking multiple of these blocks, the network is able to learn relationships across different regions of the image without requiring the heavy computational load of a full transformer.

C. Feature Refinement

After the transformer blocks, we use a single 3×3 convolution layer to refine the feature maps. Although this layer is simple, it plays an important role. The attention operations sometimes introduce small inconsistencies or artifacts in the features. The refinement layer smooths these irregularities and prepares the representation for the final upsampling stage. Without this refinement, the reconstructed images tended to show slight noise or uneven textures during our early experiments, so this step helps improve both stability and visual quality.

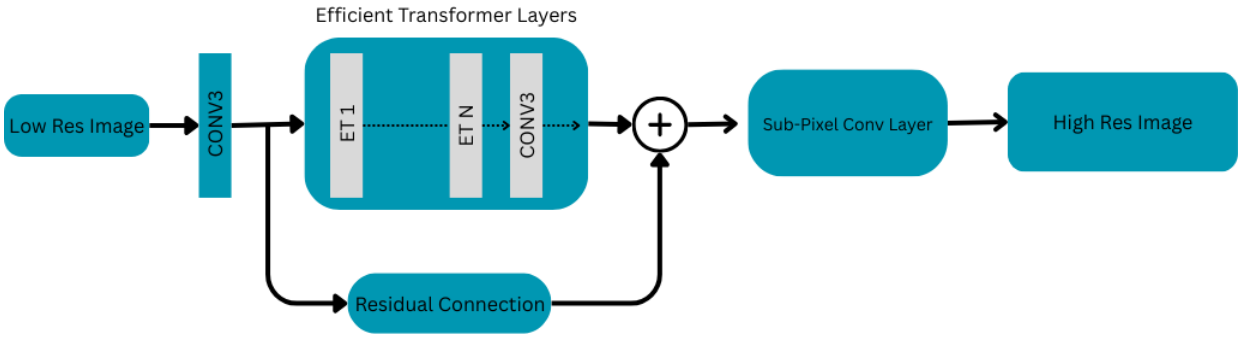

Fig. 1: Architecture overview of ETSPCN

D. Pixel-Shuffle Upsampling

To reconstruct the high-resolution image, we use pixel-shuffle, an approach introduced in ESPCN [1]. This method rearranges elements from the channel dimension into spatial resolution, which increases the width and height of the image while maintaining efficiency (Figure 3). Pixel-shuffle is much lighter than transposed convolution and avoids the checker-board artifacts that sometimes appear in other upsampling methods. It also allows the model to produce sharp edges and cleaner textures. For a lightweight architecture like ours, pixel-shuffle was a practical and effective choice, and it contributed noticeably to the quality of the final outputs.

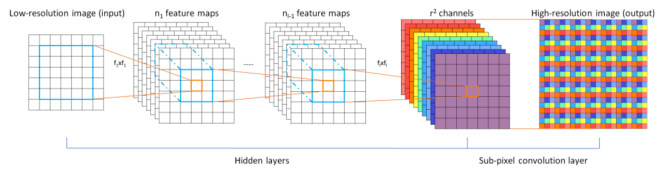

Fig. 3: Pixel-shuffle (sub-pixel convolution) upsampling process from ESPCN [1]

E. Residual Upsampling Path

Alongside the main reconstruction path, we include a residual branch that processes the original low-resolution input. Initially, we implemented the residual branch using a bicubic upsampling approach. As this was resulting in weaker outputs, we also decided to use a small 3×3 convolution layer followed by pixel-shuffle for the residual connection. The output of this branch is added to the main reconstruction. The idea behind this addition is that not all information needs to be learned from scratch. The residual branch preserves low-frequency structure and general color distribution, which helps the transformer blocks focus on improving fine details rather than relearning the overall content of the image. This approach also stabilizes training by giving the model an easier gradient path.

F. Complete Model Flow

Putting everything together, the ETSPCN model follows a simple and efficient sequence. The input image first goes through shallow feature extraction, then moves through several Efficient Transformer blocks that capture both local and global patterns. A refinement layer cleans up the features, and pixel-shuffle converts them into a higher-resolution output. The residual branch reinforces structure and improves consistency. Even though the model is compact, the combination of these components allows it to take advantage of transformer ideas while remaining computationally friendly.

III. Experiments

A. Training Setups and Datasets

For training our model, we have used the Div2K dataset [4]. This dataset consists of 100 high-resolution images, with each image's resolution being 2048×2048. For training, we down-sampled the data by the *scaling factor*, which became our main training data. The original images stayed as the ground truth labels for the model to train against. We trained the model for 300 epochs using the following settings:

- We randomly cropped 16 low-resolution image patches with a size of 48×48. These patches are then used as inputs for each epoch.
- For data augmentation, we randomly flipped some of the images horizontally. Also, 90-degree rotations were used for some of the images.
- We set the initial learning rate to 2×10^{-4}. This learning rate was decreased to half after every 200 epochs.
- As the optimizer, we used the *Adam* optimizer. *L1 loss* was used as the loss function.

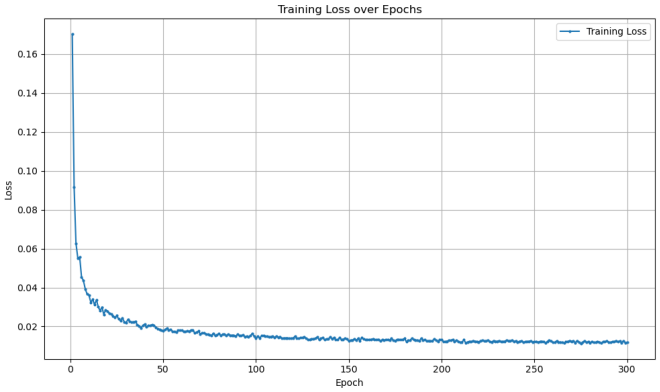

Fig. 4: Training loss over epochs

Figure 4 showcases the training loss for our model over the 300 epochs. It is evident from the image that the loss decreased smoothly, without any gradient explosion or vanishing.

B. Evaluation

1) Evaluation Metrics: For evaluating the performance of our proposed model, we used the traditional validation metrics in the image restoration domain, the PSNR and SSIM values. PSNR refers to Peak Signal to Noise Ratio, which quantifies the difference between the peak possible strength of a signal (ground truth image) and the strength of the noise that distorts it (low-resolution image). The PSNR values are expressed in decibels (dB), and a higher PSNR value refers to a good quality of the generated image. SSIM means the Structural Similarity Index, which is more of a perceptual metric. It evaluates the visual impact of changes between two images, like the structural information, luminance, and contrast difference. The SSIM values range from -1 to 1, where 1 refers to perfect structural similarity, and -1 is the exact opposite. To compare how good our model performs, we used 5 traditional datasets in the field of image super resolution, Set5 [5], Set12 [6], Set14 [7], BSD100 [8], and Urban100 [9].

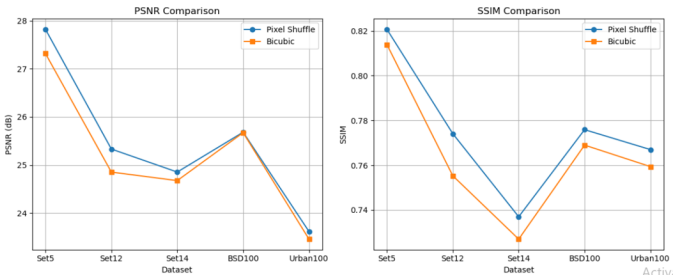

Fig. 5: PSNR and SSIM comparison between bicubic and pixel-shuffle as residual connection ($\times 4$ Scale)

As described in the model architecture, we experimented with two variants of our hybrid model, one with the bicubic upsampling as the residual connection and the other with the pixel-shuffle as the residual connection. We evaluated both of these models for 4x upscaling using the validation datasets and evaluation metrics described above. Figure 5 demonstrates that using the pixel-shuffle as the residual connection performs better than using a bicubic residual block unanimously across all the datasets. That is why we have used the pixel-shuffle as the residual block for all the evaluations you will see from this point onward.

2) Quantitative Analysis: From Table I, we get the quantitative results for our model for an upscale factor of 4 against the traditional methods, like ESRT [3], SwinIR [2], and bicubic extrapolation. Table II demonstrates the results for an upscale factor of 2. From both of these results, we can see that our hybrid model, though it performs better than the conventional bicubic method, fails to beat the existing works. The SwinIR and the ESRT methods both outperformed our model by a large margin for scale $\times 4$. For scale $\times 2$, the performance of our model increased, but it still could not outperform the

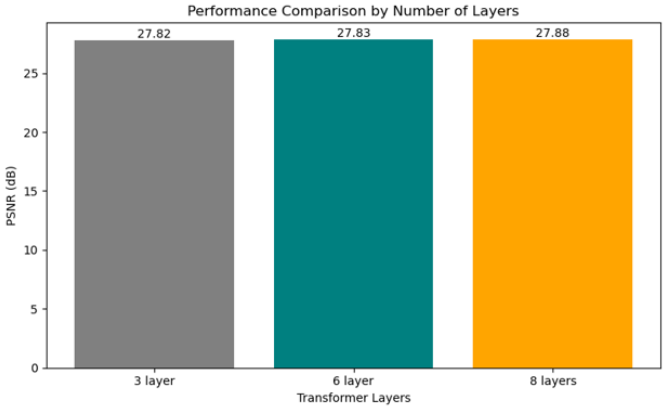

Fig. 6: PSNR values for different layers of Efficient Transformers on Set5 ($\times 4$ Scale)

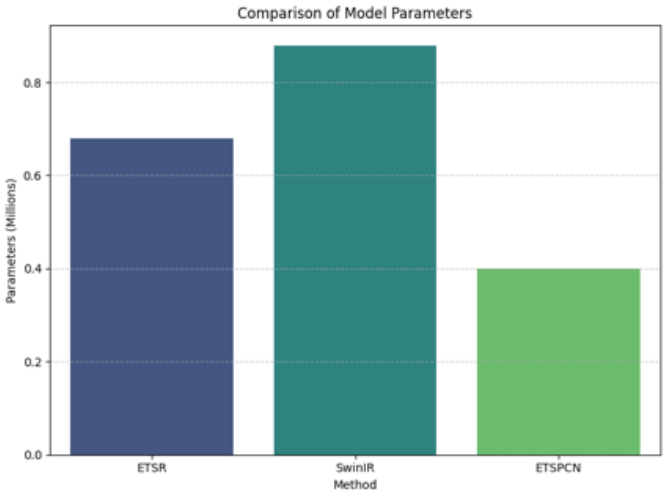

Fig. 7: Learnable parameters comparison between ESRT, SwinIR, and ESTPCN ($\times 2$ Scale)

SwinIr method, which achieved a 38.35 dB PSNR score for the Set5 dataset. The results from the ESRT [3], SwinIR [2] are taken from their paper, and they might have different training settings from ours. Also, SwinIR is trained on a larger dataset than ours, which can also contribute to their better performance. But still, the difference is too big to just lay the blame on their training settings. To investigate what could be the issue with our model's underperformance, we modified the model architecture by experimenting with deeper layers of efficient transformers. We ran the test for three different layers of efficient transformers: 3, 6, and 8. Figure 6 demonstrates the results for this experiment. From this figure, we can see that even increasing the transformer layer depths did not improve the performance of the PSNR values, indicating that the problem is elsewhere.

As another of our goals was to verify how efficient our model is, we also compared the total number of learnable parameters and total floating point operations (FLOPs) for our model against the SwinIR and ESRT models. The results were

TABLE I: Super-Resolution Performance Comparison ($\times 4$ Scale)

Method	Scale	Set5		Set14		BSD100		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	$\times 4$	26.91	0.8093	24.48	0.7228	25.69	0.7632	23.15	0.749
ESRT [3]	$\times 4$	32.19	0.8947	28.69	0.7833	27.69	0.7379	26.39	0.796
SwinIR [2]	$\times 4$	**32.44**	**0.8976**	**28.77**	**0.7858**	**27.69**	**0.7406**	**26.47**	**0.798**
ETSPCN (Ours)	$\times 4$	27.821	0.8207	24.852	0.7370	25.68	0.7759	23.61	0.767

TABLE II: Super-Resolution Performance Comparison ($\times 2$ Scale)

Method	Scale	Set5		Set14		BSD100		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	$\times 2$	32.17	0.9243	28.52	0.8709	27.22	0.8552	26.97	0.885
SwinIR [2]	$\times 2$	**38.35**	**0.9620**	**34.14**	**0.9227**	**32.47**	**0.903**	**33.51**	**0.941**
ETSPCN (Ours)	$\times 2$	33.37	0.9322	29.08	0.8809	26.98	0.8594	27.61	0.897

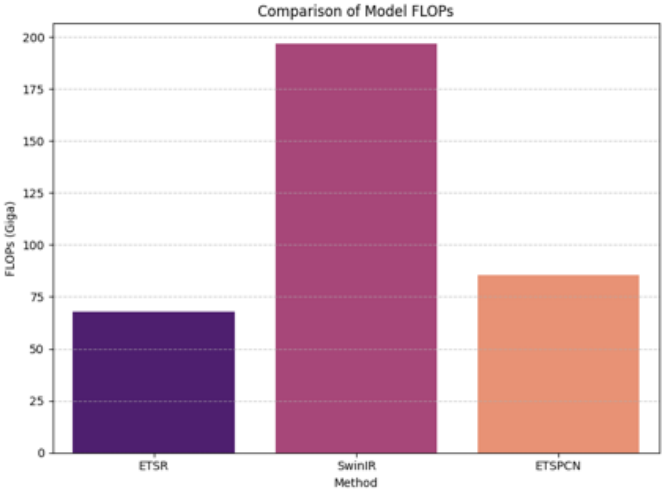

Fig. 8: Floating Point Operations (FLOPs) comparison between ESRT, SwinIR, and ESTPCN ($\times 2$ Scale)

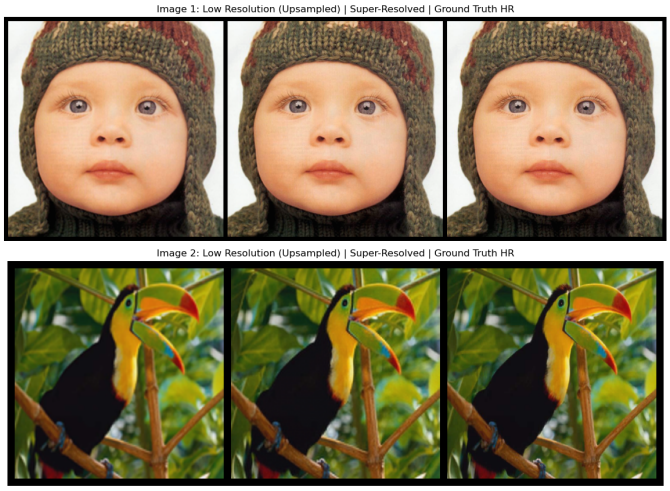

Fig. 9: Visual comparison of image upscaling - Set5 ($\times 2$ Scale).

better in this criterion, as shown in Figures 7, 8. The number of learnable parameters dropped a lot from the ESRT and SwinIR methods, validating the lightweight nature of our model. Also, the FLOP count for our model was 85.39 G Flops, which is slightly higher than ESRT (67.7 G Flops), but significantly lower than the FLOP count of SwinIR (196.6 G Flops). This performance gives us confidence that this model can decrease the overall runtime complexity of a transformer-based model.

3) Qualitative Analysis: To visualize the quality of our model in super-resolution tasks, we visualize some of the sample images from the validation datasets. For visualization, we only used the Set5 and Urban100 datasets. Figure 9 provides the visual examples of the $\times 2$ upscaling of a couple of images from the Set5 dataset. The leftmost image is the low-resolution image that we intend to upscale. The middle image is the upscaled image using our model, and the rightmost is the ground truth image. Similarly, Figure 10 visualizes the $\times 4$ upscaling for the same images from Set5. Although the $\times 2$ scaled images visually look much improved from the

Fig. 10: Visual comparison of image upscaling - Set5 ($\times 4$ Scale)

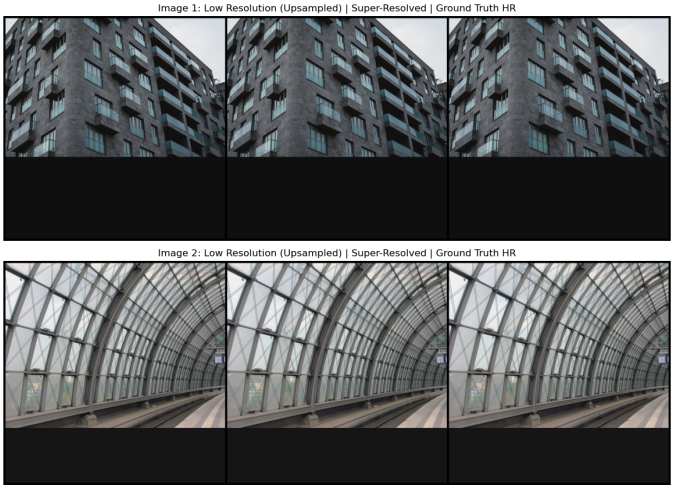

Fig. 11: Visual comparison of image upscaling - Urban100 ($\times 2$ Scale)

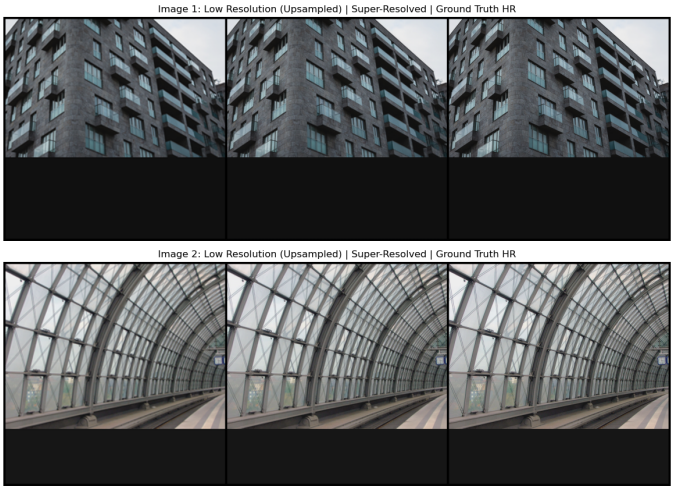

Fig. 12: Visual comparison of image upscaling - Urban100 ($\times 4$ Scale)

lower-resolution images, the $\times 4$ scaled images still produce blurry outputs. A similar conclusion can also be made for the Urban100 dataset visualization (Figures 11, 12).

IV. Conclusion and Future works

In this project, we implemented a hybrid image restoration model, ETSPCN, that utilizes the architectural details from multiple well-established models, like SwinIR [2], ESPCN [1], and ESRT [3]. ETSPCN used the efficient transformer from the ESRT [3] paper along with the pixel-shuffle from ESPCN [1] in a SwinIR [2] like pipeline for achieving a lightweight model that can also capture the global details from low-resolution images. The main idea was to use the positive features from the existing models, also while making them computationally inexpensive. Although we achieved computational efficiency, the overall image restoration performance of ETSPCN did not meet our expectation. Compared to the existing popular models, the performance of ETSPCN does not hold water.

There can be various reasons for the underperformance of ETSPCN. The efficient transformers reduce the number of channels by half, which can lose some information of the original image. In the original ESRT [3] paper, where the authors introduced the efficient transformer blocks, they also used high performance blocks (HPBs)using CNNs to preserve the information lost by the efficient transformer blocks. By to keep the model simple, we did not introduce HPBs in our architecture. Instead of simply using a global residual block, we need to think of introducing extra blocks in front of the efficient transformers to preserve the lost information. Also, instead of simply splitting the attention blocks and multiplying them together using a scaler product, an weighted multiplication approach can be experimented to improve the results from the efficient transformer. Overall, this project opens up multiple research questions and opportunities in developing a lightweight and accurate transformer-based image super resolution model.

References

[1] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D. and Wang, Z., 2016. Real-time single-image and video super-resolution using an efficient sub-pixel convolutional neural network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1874-1883).

[2] Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L., and Timofte, R. (2021). Swinir: Image restoration using Swin Transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1833-1844).

[3] Lu, Z., Li, J., Liu, H., Huang, C., Zhang, L., and Zeng, T., 2022. Transformer for single-image super-resolution. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 457-466).

[4] Agustsson, E., and Radu T., 2017. Ntire 2017 challenge on single image super-resolution: Dataset and study. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, (pp. 126-135).

[5] Bevilacqua, M., Roumy, A., Guillemot, C., and Morel, M., 2012. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In British Machine Vision Conference, (pages 135.1–135.10).

[6] Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2009, September). Non-local sparse models for image restoration. In 2009 IEEE 12th international conference on computer vision (pp. 2272-2279).

[7] Zeyde, R., Elad, M., and Protter, M. (2010, June). On single image scale-up using sparse-representations. In International conference on curves and surfaces (pp. 711-730).

[8] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001, July). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In Proceedings eighth IEEE international conference on computer vision. ICCV 2001 (Vol. 2, pp. 416-423).

[9] Huang, J. B., Singh, A., and Ahuja, N. (2015). Single image super-resolution from transformed self-exemplars. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5197-5206).