Image Super-Resolution Using Deep Recurrent Residual Networks (DRRN)

Abstract

This report explores the use of Deep Recurrent Residual Networks (DRRN) for the task of image super-resolution. The aim is to investigate the impact of residual connections, recurrent blocks, and recursive layers on training and validation performance. We analyze the trade-offs between accuracy, loss, and computational efficiency, and present experimental results highlighting the significance of residual connections in training deep models for image super-resolution.

1 Introduction

Image super-resolution is a task aimed at reconstructing high-resolution (HR) images from low-resolution (LR) inputs. Super-resolution techniques have widespread applications in fields such as medical imaging, satellite imagery, and video enhancement. Deep learning-based methods, particularly residual networks and recurrent models, have recently demonstrated remarkable performance in this area.

The focus of this project is the implementation of a Deep Recurrent Residual Network (DRRN) to enhance image resolution. We experiment with different configurations of residual and recursive blocks and evaluate their effect on training stability and model performance.

2 Methodology

2.1 Network Architecture

The architecture used for this project is a Deep Recurrent Residual Network (DRRN). The DRRN model is designed with:

- An initial convolutional layer.
- Recursive blocks composed of residual units.
- A final convolutional layer to output the super-resolved image.

- Global residual learning to enhance training by adding the LR input to the final HR output.

2.2 Dataset Preparation

We followed the methodology outlined in the base paper [1] and extracted 31×31 patches from a dataset containing 601 images. The dataset patches were generated with a stride of 21, leading to a total of 103,455 training samples. These patches were used for both the training and validation stages.

2.3 Training Procedure

We trained the DRRN model using Mean Squared Error (MSE) as the loss function and the Adam optimizer with a learning rate of 0.001. Gradient clipping was employed to tackle issues related to exploding gradients, and early stopping criteria were set to monitor validation loss. Training was performed for 50 epochs, with early stopping activated if the validation loss showed no improvement for 5 consecutive epochs.

3 Mathematical Formulation

In the Deep Recurrent Residual Network (DRRN), the primary mathematical operations involve residual learning and recursive block structures. Below are the key formulations:

3.1 Residual Block

In a residual block, the input x is passed through a series of convolutional layers to learn a residual function $F(x, \{W_i\})$. The output of the block is computed by adding the learned residual to the input:

$$y = x + F(x, \{W_i\})$$

where:

- x is the input feature map.
- $F(x, \{W_i\})$ is the residual function, with W_i representing the weights of the convolutional layers.

3.2 Recursive Block

A recursive block is a structure composed of multiple residual units. The output of each recursive block is computed by stacking residual units, and the same weights are shared among these units:

$$y_{\text{rec}} = x + \sum_{i=1}^{n} F_i(x, W_i)$$

where:

- y_{rec} is the output of the recursive block.
- n is the number of residual units in the recursive block.
- $F_i(x, W_i)$ represents the transformation applied by the i^{th} residual unit.

3.3 Global Residual Learning

At the network level, the input LR image x is passed through several recursive blocks, and the output is the HR image reconstruction y, computed as the sum of the LR input and the learned residual:

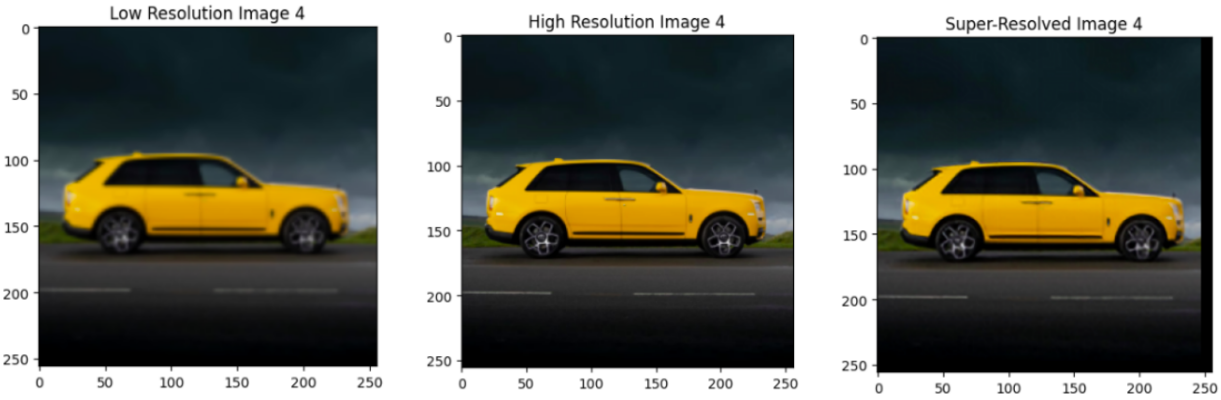
$$y = x + G(x, \{W_i\})$$

where:

- $G(x, \{W_i\})$ is the output of the final convolutional layer after all recursive blocks.

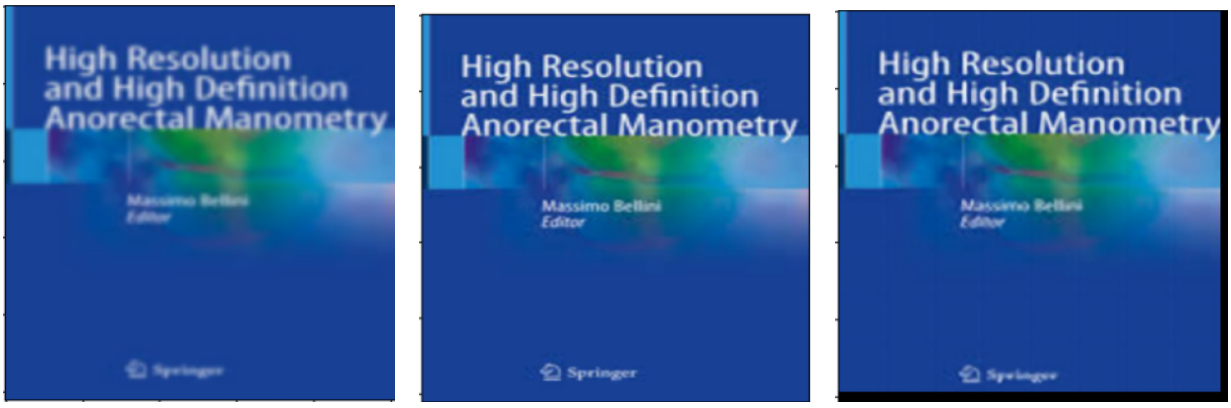
4 Results

The following section discusses the model's performance with residual connections and the effect of removing them and changing the parameters like recursive blocks and layers.



(a) Low Resolution Image(From Dataset) (b) High Resolution Image(From Dataset) (c) Super Resolution Image(From Model)

Figure 1: Comparing the input data to the output image generated



(a) Low Resolution Image(From Dataset) (b) High Resolution Image(From Dataset) (c) Super Resolution Image(From Model)

Figure 2: Comparing the input data to the output image generated

(a) Low Resolution Image(From Dataset) (b) High Resolution Image(From Dataset) (c) Super Resolution Image(From Model)

Figure 3: Comparing the input data to the output image generated

4.1 Effect without Residual Connections

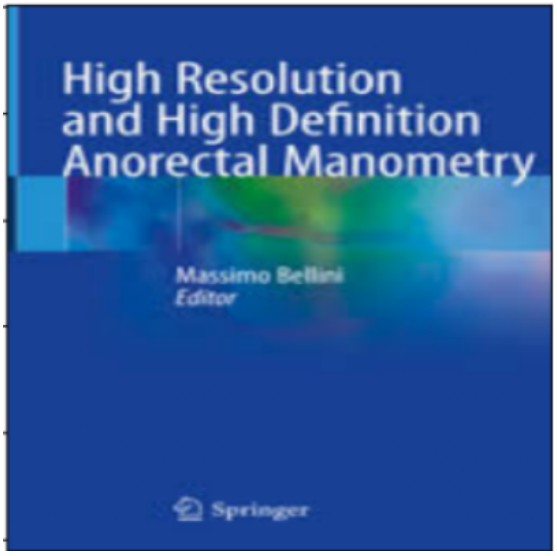
We just changed 1 line in code that is x+input to just x as output and the results changed drastically, model's accuracy forget increasing were decreasing with increasing epochs.

When training deep neural networks, the vanishing gradient problem becomes particularly severe as gradients are backpropagated through many layers. During backpropagation, the chain rule is applied to compute gradients, resulting in the product of several small derivatives. In deeper networks, this leads to an exponential decay of gradient values as they move from the output layers to the earlier layers. As a consequence, gradients become exceedingly small, effectively approaching zero, and the earlier layers in the network fail to receive meaningful updates during training. Without sufficient gradient flow, these layers cannot adjust their weights properly, impairing the model's ability to learn complex

```
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2488 - loss: 0.0847 - val_accuracy: 0.3027 - val_loss: 0.0756
Epoch 16/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2315 - loss: 0.0846 - val_accuracy: 0.3120 - val_loss: 0.0755
Epoch 17/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2284 - loss: 0.0845 - val_accuracy: 0.1635 - val_loss: 0.0755
Epoch 18/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2520 - loss: 0.0847 - val_accuracy: 0.5097 - val_loss: 0.0755
Epoch 19/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2283 - loss: 0.0842 - val_accuracy: 0.1635 - val_loss: 0.0757
Epoch 20/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2398 - loss: 0.0843 - val_accuracy: 0.1684 - val_loss: 0.0756
Epoch 21/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2180 - loss: 0.0843 - val_accuracy: 0.1635 - val_loss: 0.0755
Epoch 22/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2067 - loss: 0.0839 - val_accuracy: 0.1749 - val_loss: 0.0755
Epoch 23/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2110 - loss: 0.0840 - val_accuracy: 0.1746 - val_loss: 0.0754
Epoch 24/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2062 - loss: 0.0843 - val_accuracy: 0.1632 - val_loss: 0.0756
Epoch 25/25
728/728 ━━━━━━━━━━━━━━━━━━━━ 123s 169ms/step - accuracy: 0.2091 - loss: 0.0843 - val_accuracy: 0.1635 - val_loss: 0.0755
```

Figure 4: Effect on Validation and accuracy by changing number of recursive layers

features and representations. This phenomenon not only slows down convergence but also hinders the overall learning capacity of the model, especially in very deep architectures.



(a) Ground-Truth Expected Output

(b) Output without Residual Connections

Figure 5: Comparison of outputs with and without residual connections

4.2 Effect of Recursive layers

In this project, we experimented with varying the number of residual layers within each recursive block to determine the optimal configuration. Our findings concluded that 4 to 6 residual layers per block provided the best balance, resulting in higher accuracy and lower loss, while maintaining an efficient training time. This configuration achieved the most favourable trade-off between performance and computational efficiency.
Increasing the number of residual layers beyond 6 led to diminishing returns, with only incremental improvements in performance and longer training times. Furthermore, models with 10-12 layers showed signs of overfitting, indicating that careful tuning of the residual layer count is crucial for achieving optimal results in image super-resolution tasks. Overall, this insight into residual layer optimisation is vital for enhancing model performance while ensuring practical training efficiency.

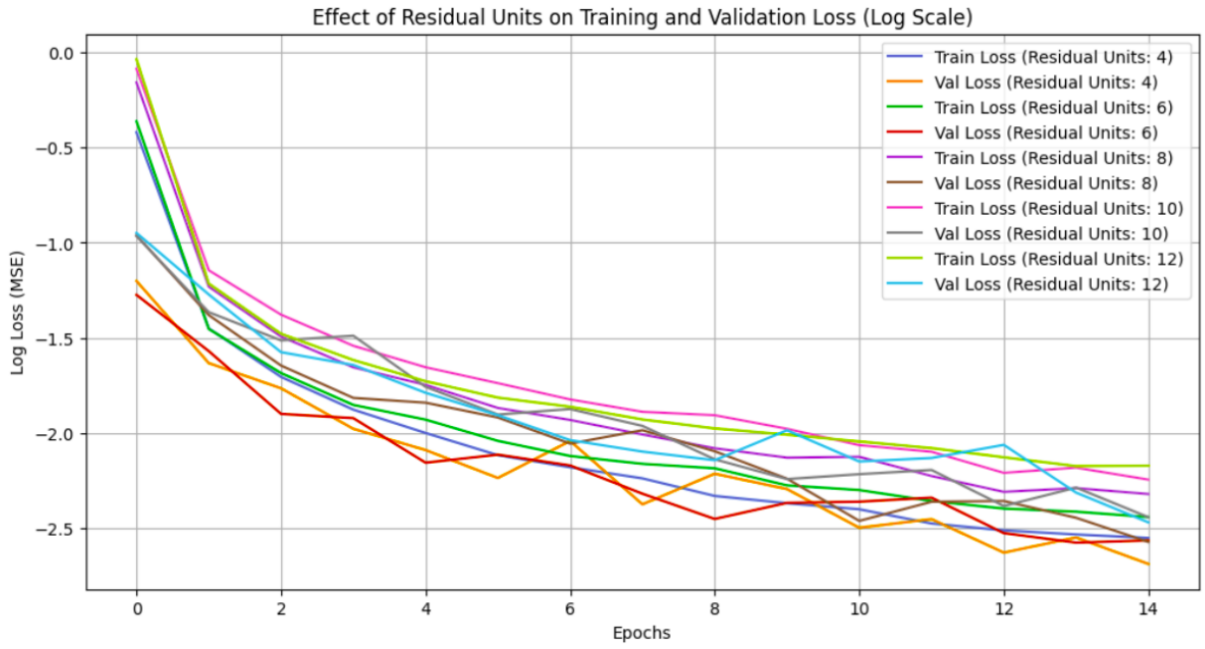
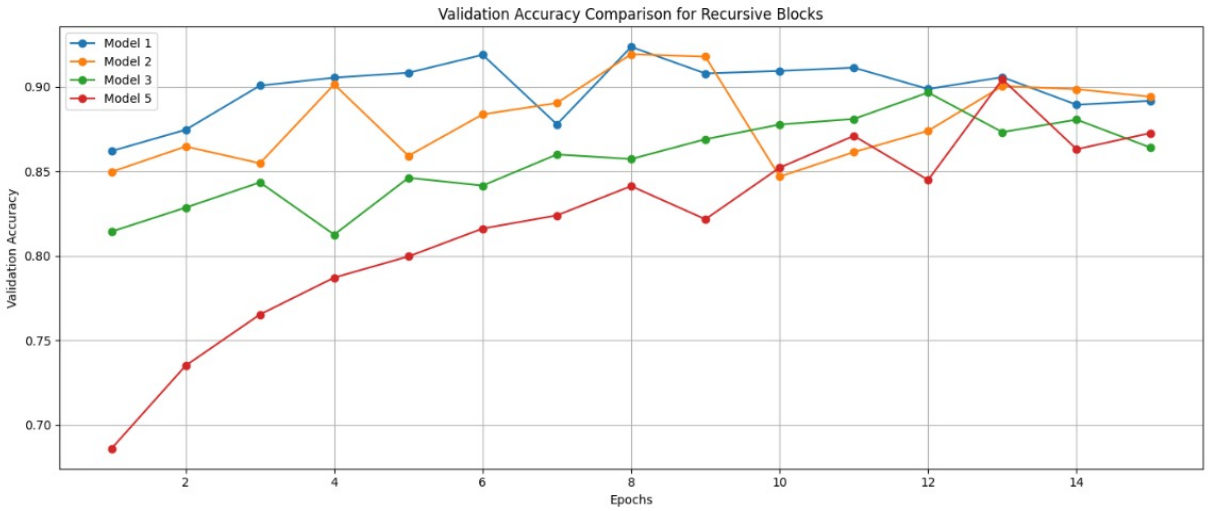


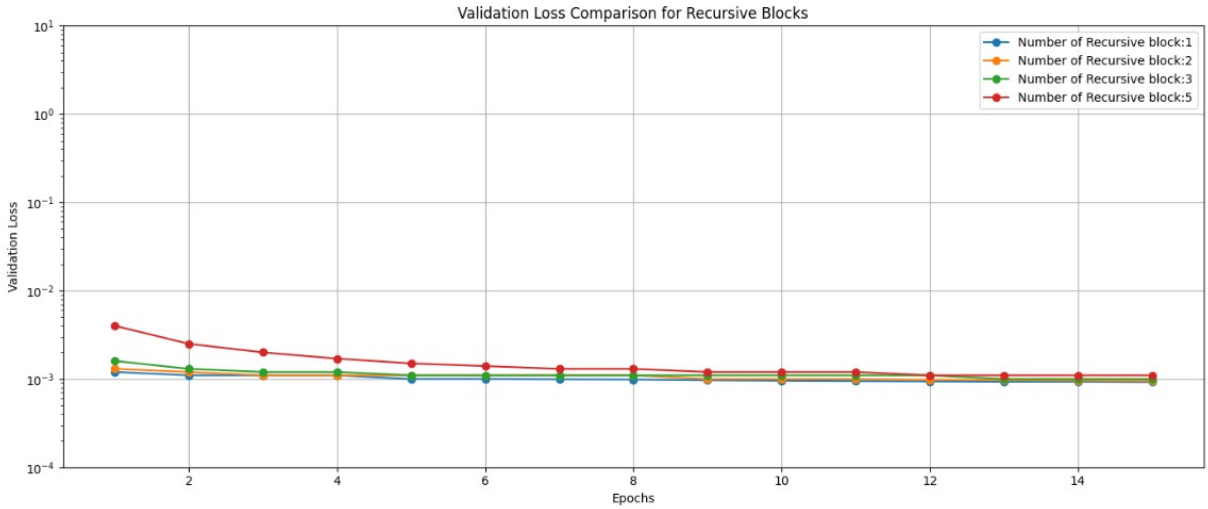
Figure 6: Effect on Validation and accuracy by changing number of recursive layers

4.3 Effect of Recursive Blocks

We experimented with various configurations of recursive blocks, finding that the performance (in terms of accuracy and loss) improved initially as the number of blocks increased but started to plateau after 3 or 4 blocks.



(a) Validation Accuracy



(b) Validation Loss

Figure 7: Validation Accuracy and Loss with Different Numbers of Residual Blocks

4.4 Training Time and Performance

Increasing the number of residual units and recursive blocks increased training time but only led to marginal performance gains beyond 6 residual units and 3 recursive blocks.

5 Conclusions

Through our experiments, we conclude that the use of residual connections significantly improves the training stability and performance of deep models in the image super-resolution

task. Additionally, an optimal balance between the number of residual units and recursive blocks must be achieved to avoid overfitting and excessive training time. Our findings suggest that 4 to 6 residual units per block, combined with 2 or 3 recursive blocks, provide a good trade-off between accuracy, loss, and computational efficiency.

6 Future Work

Future research can focus on optimizing the number of residual units and recursive blocks through more advanced hyperparameter tuning techniques. Furthermore, investigating alternative loss functions, such as perceptual loss, could improve the perceptual quality of the super-resolved images.

References

[1] Tai, Y., Yang, J., and Liu, X. (2017). *Image Super-Resolution via Deep Recursive Residual Network*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3147-3155.