Residual Dense Network for Image Super-Resolution

Aastha Malhotra
B.Tech CSE (Batch of '27)
Shiv Nadar University
Delhi, India
am565@snu.edu.in

Venkat Ravi KB
B.Tech CSE (Batch of '26)
Shiv Nadar University
Delhi, India
vk839@snu.edu.in

Raashi Sharma
B.Tech CSE (Batch of '27)
Shiv Nadar University
Delhi, India
rs708@snu.edu.in

Dhanya Girdhar
B.Tech CSE (Batch of '27)
Shiv Nadar University
Delhi, India
dg218@snu.edu.in

***Abstract*—Residual Dense Networks (RDNs) carry out Single Image Super Resolution (SISR) by using hierarchical features through dense and residual connections. In this paper we implement RDN in PyTorch and extend its architecture by increasing the number of residual dense blocks and evaluate its performance. We analyse the saturation of the PSNR value and attempt explaining it through various phenomena. Our study provides insights into the limitations of simply deepening architectures for SISR and thoughts on improved training strategies.**

***Index Terms*—Single Image Super-Resolution (SISR), Residual Dense Networks (RDN), Deep Learning,**

I. Introduction

The aim of SISR is to reconstruct a single high resolution (HR) image from a single low-resolution input image (LR). In recent years, convolutional neural networks (CNNs) have had very good results in SISR by learning end to end mappings from LR to HR images. [1]

RDN however, introduced an architecture that fully utilises hierarchical features across all existing convolutional layers. By combining dense connectivity with local residual learning and global feature fusion, RDN is aimed to preserve information without the computational overhead of pre-upscaling LR images.

In this paper, we implement the RDN model proposed in Y. ZHang et al. [2] in PyTorch as provided in this Github repository [3] with tweaks of our own. Our study looks into practical limitations in the existing architecture and provides insights into directions for future SISR research.

II. Related Work and Background

Early deep learning approaches to SISR started out with the Super-Resolution CNN (SRCNN)1, which first established end-to-end mapping between interpolated low-resolution (LR) inputs and their high-resolution (HR) counterparts. The Very Deep Super-Resolution (VDSR) network [4] built upon this and used deeper architectures and residual learning to further improve reconstruction quality.

Efforts since then on this particular area focused on memory and feature utilisation. MemNet [5] introduced persistent memory blocks to capture long-term dependencies across layers, though it worked by primarily using pre-upscaled LR images as input. This increased computational complexity. SRDenseNet6 used dense blocks for feature extraction but did not have any mechanisms for global feature aggregation. Limitations such as these were addressed in Residual Dense Networks [2]. It combines dense connectivity, local residual learning, global feature aggregation all while preserving all required information without the computational overhead of pre-upscaling images such as in previous models such as VDSR and MemNet.

III. Methodology

RDN architecture [2] is made up of four components. Firstly, we have shallow feature extraction network (SFENet) which uses two convolutional layers to extract low-level features from the input image. Initial shallow features are produced by the first layer after which, the second layer prepares the input for deeper processing. After this, core feature extraction is performed by a sequence of RDBs (Residual Dense Blocks), each containing densely connected convolutional layers enhanced with local feature fusion (LFF) and local residual learning (LRL). This is done to ensure best feature propagation [2]. The outputs from all the RDBs are then processed by the GFF (Global Feature Fusion) Module which concatenates all the features and applies a 1×1 then a 3×3 convolution to fuse hierarchical information [1], [6]. The final reconstruction of a HR image is done by sub-pixel convolution [7] using UPNet (Upsampling Network) and supports scale factors of $2\times$, $3\times$ or $4\times$.

The PyTorch implementation that we have used in this work closely follows the original RDN architecture and also modularizes the components, which gives us more flexibility than before. The RDB class contains the dense block structure, implementing stacked densely connected convolutions with local feature fusion. The RDN class contents multiple RDBs and applies GFF while also implementing shallow feature extraction with upsampling. The model configuration can be modified to change the number of RDBs used, the number of convolutional layers used per RDB and the growth rate.

In this paper, to explore whether deeper or wider architectures improve super-resolution performance, we increased the number of blocks in the RDN. Our goal was to improve the model's representational capacity to boost PSNR (Peak Signal to Noise Ratio).

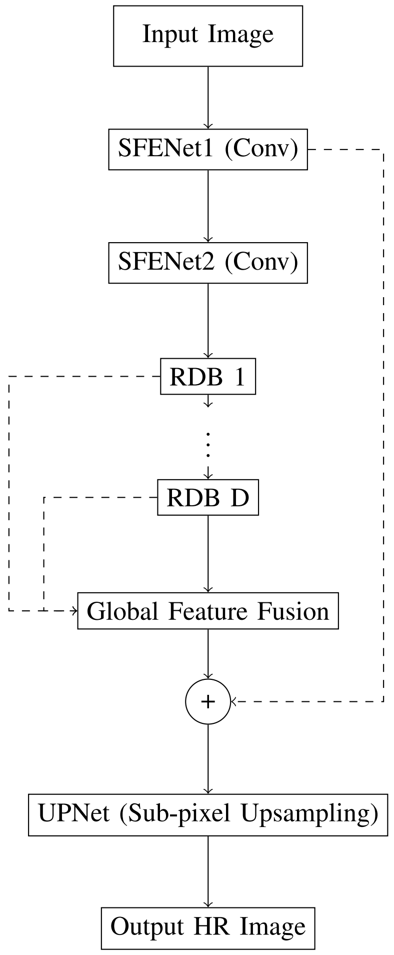

Fig. 1. Overview of the Residual Dense Network (RDN) architecture.

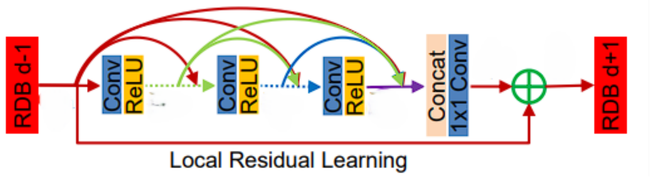

Fig. 2. RDB Structure

IV. Experimental Setup

Datasets and Metrics. The experiments that we conducted used the DIV2K dataset2 [8], which is a standard benchmark for SISR tasks. This dataset consists of 800 high-resolution training images with 2K resolution, along with 100 validation images and 100 test images. The bicubic downsampling with scaling factor $\times 4$ LR counterparts was used in this paper for model training and evaluation. Our model was trained using the 800 training images part of the training dataset, and we used the validation set (801-810) for evaluation during training. For testing, we used four standard benchmark datasets: Set5 [9], Set14 [10], B100 [11], and Urban100 [12]. The super-resolution results were evaluated using PSNR on the Y channel (luminance) of the transformed YCbCr space.

Training Settings. In our implementation, we used patch sizes of 192×192 from the HR images. The batch size was set to 16. We trained the model using the ADAM optimizer [13] and the learning rate was initialized to 5×10^{-5}. The model was trained for 20 epochs with 16,000 iterations per epoch because the model stopped showing significant improvement after 20 epochs. We used the L1 loss function for optimization. In the implementation of our code we used PyTorch with CUDA acceleration to optimize computational efficiency to the best of our ability during both training and evaluation phases.

V. Experimental Results

The source code of the implemented RDN model can be found on the original RDN GitHub repository [3].

A. Results and Analysis

Network Configuration. We implemented the RDN model that consisted of 16 Residual Dense Blocks (RDBs), each of these RDBs contained 8 convolutional layers with a growth rate (Number of filters per convolutional layers) of 64 (3×3 kernel size). This growth rate determines the number of feature maps that each convolutional layer produces, this results in dense feature concatenation all throughout the network. The model uses local and global feature fusion as mentioned in the original RDN architecture. [14].

Quantitative Results on DIV2K. Table I is used to visually represent the PSNR performance of the RDN model that we trained at different epochs on the benchmark DIV2K validation dataset. The model showed steady improvement all throughout the training process and achieved the highest PSNR value of 29.144 dB at epoch 18 for $4\times$ super-resolution.

Epoch	PSNR (dB)	Epoch	PSNR (dB)
1	27.422	11	28.671
2	27.830	12	28.906
3	28.184	13	28.960
4	28.401	14	28.985
5	28.494	15	28.805
6	28.593	16	29.101
7	28.441	17	29.099
8	28.751	18	**29.144**
9	28.770	19	29.080
10	28.795	20	29.080

TABLE I
PSNR Performance of RDN on DIV2K Validation Set ($4\times$ SR)

Loss Convergence. Figure 3 shows the L1 loss convergence that has happened during our training period. The loss is seen to be decreasing rapidly in the first couple epochs, from 31.64 at the beginning of epoch 1 to approximately 8.09 by the end of epoch 2. The convergence then slows down in the later epochs, with the value of loss stabilizing around 6.7 after epoch 15. This pattern shows us that the learning the model is undergoing is effective and is approaching convergence within the duration of training.

Benchmark Dataset Results. We tested our trained model on standard benchmark datasets. Table V-A shows the results

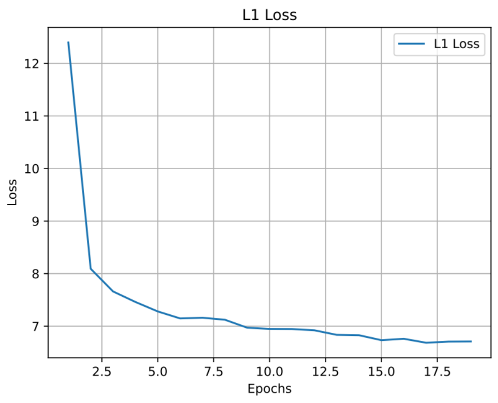

Fig. 3. L1 loss convergence over training epochs.

in terms of PSNR for 4× super-resolution on Set5, Set14, B100, and Urban100 datasets. The model achieved 31.565 dB, 28.230 dB, 27.304 dB, and 25.265 dB on these datasets, respectively.

Dataset	PSNR (dB)
Set5	31.565
Set14	28.230
B100	27.304
Urban100	25.265

TABLE II
PSNR PERFORMANCE (DB) OF RDN ON BENCHMARK DATASETS (4× SR)

B. Discussion

The results from the experiment that we conducted show that the the RDN architecture is very effective for single image super-resolution tasks. A significant PSNR improvement of 1.722 dB was achieved by the model from epoch 1 to its peak performance at epoch 18 on the DIV2K validation set. The dense skip connections and hierarchical features that are being used in RDN improved the information flow throughout the network significantly, this contributed to superior reconstruction quality.

These benchmark results are a statement to the model's generalization capability across varying datasets with different image characteristics. The performance on Set5 (31.565 dB) is particularly notable, since it represents high-quality reconstruction for 4× super-resolution. The relatively lower PSNR on Urban100 (25.265 dB) reflects the challenge of reconstructing complex urban scenes with intricate details and structures.

The small fluctuations in PSNR values observed in later epochs (18-20) can be a sign that the model had approached convergence. Choosing to continue training after 20 epochs might give us marginal improvements but it would likely result in diminishing returns.

C. Visual Results

To visually represent the prowess of the RDN model that we have ran, we have created a qualitative comparison on an example image from the Set5 dataset. Fig. 4 shows the ground truth high-resolution image alongside the bicubic interpolation result and our RDN super-resolution output. The RDN model clearly recovers more detailed structures and produces sharper edges compared to the bicubic interpolation, which returns blurring artifacts.

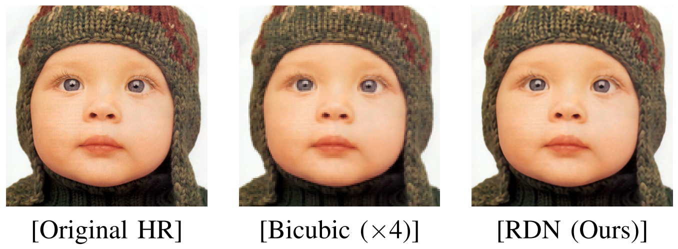

Fig. 4. Visual comparison on "Baby" image from Set5 dataset for 4× SR. Our RDN model recovers more detailed textures and produces sharper edges compared to bicubic interpolation.

VI. MODEL DEPTH EXPLORATION

We looked at in detail how increasing network depth affects the performance of the Residual Dense Network (RDN) for image super-resolution. The baseline model followed the original RDN design, using 16 Residual Dense Blocks (RDBs), each with eight convolutional layers, a growth rate of 64, and 3×3 kernels.

To test the effect of depth, we changed the architecture by increasing the number of RDBs from 16 to 32, and kept all the other settings the same. Our aim in doing this was to evaluate whether adding layers could improve the model's ability to reconstruct the high-resolution images that we are looking for.

The updated model, as reflected in the logs, included:

- Two initial convolution layers (SFENet1 and SFENet2) for feature extraction
- 32 RDBs, each with:
  - Eight densely connected convolution layers
  - ReLU activation after each convolution
  - Local feature fusion via a 1×1 convolution
- Global feature fusion using 1×1 and 3×3 convolutions
- An upsampling module (UPNet) with PixelShuffle layers for 4× scaling

A. Training Protocol

Both of the network configurations were trained using the same protocols to make sure that the comparison is fair. The training process, as documented in the log files, followed these specifications:

- Dataset: DIV2K training set with validation on DIV2K validation set
- Scale factor: 4× super-resolution
- Loss function: L1 loss (Mean Absolute Error)

- Learning rate: 5.00e-5, maintained constant throughout training
- Batch processing: Progress reported every 1600 iterations
- Training duration: 20 epochs for both configurations

The training logs reveal clearly distinguishable learning patterns between the two configurations. For the 32-RDB network, the L1 loss went down from its initial values of 28.3160 in epoch 1 to 6.6241 by epoch 19. The PSNR for the validation improved correspondingly from 27.592 dB to reach its peak value of 29.264 dB as seen in the results. Each epoch ran for 51-52 seconds per 1600 iterations, with additional 10-20 seconds for model validation in our experiment.

B. Efficiency vs. Performance Analysis

Table III is a visual representation that gives us a detailed comparison between the 16-RDB and 32-RDB configurations. The deeper 32-RDB network presented us with a peak PSNR value of 29.264 dB at epoch 19, which is only a 0.12 dB improvement over the 16-RDB configuration's peak PSNR value of 29.144 dB at epoch 18. This marginal performance gain came at a quite a large computational cost as a trade-off, with the training time per epoch increasing almost 3 times as much.

TABLE III
COMPARISON OF NETWORK CONFIGURATIONS

Network Config.	**Peak PSNR (dB)**	**Peak Epoch**	**Relative Training Time**
16-RDB	29.144	18	$1\times$
32-RDB	29.264	19	$\sim 3\times$

The computational overhead becomes particularly evident when taking a look at the practical implications of this: the 32-RDB model needed approximately three times more training time while only delivering a 0.12 dB PSNR improvement.

VII. CONCLUSION

The attempt to add residual dense blocks not being able to very effectively improve the PSNR suggests that deeper architectures alone could be insufficient for further gains in results. This can be explained by a couple different angles. Firstly, as network depth increases, the problem of vanishing gradients becomes more pronounced, making it difficult to effectively train the earliest layers through backpropagation [15]. Also, deeper networks are more prone to overfitting which means that the model can learn to memorize noise, artifacts and patterns in the training data, rather than working towards the goal of generalizing to unseen data2. It is known that techniques like residual connections which are implemented in this paper help work around this issue to an extent, but the exponential growth of feature combinations in dense networks could still lead to redundant feature learning and reduced generalization capability [16].

Future work for this paper could explore alternative strategies to improve gains beyond architecture scaling that could include gradient clipping or learning-rate warmup. These refinements in training techniques could help improve evaluation metrics such as PSNR. Overall, in this paper, we successfully implemented RDN for SISR, proving its architectural effectiveness through our experiments. Our results clearly show that RDN achieves competitive performance, although there is definitely room for future research in the area.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv preprint arXiv:1512.03385*, 2015. [Online]. Available: https://arxiv.org/pdf/1501.00092

[2] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," *arXiv preprint arXiv:1802.08797*, 2018. [Online]. Available: https://arxiv.org/pdf/1802.08797

[3] S. Son, "*EDSR-PyTorch*," GitHub repository. [Online]. Available: https://github.com/sanghyunson/EDSRPyTorch/blob/master/src/model/rdn.py (accessed: Apr. 29, 2025).

[4] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," *arXiv preprint arXiv:1511.04587*, 2015. [Online]. Available: https://arxiv.org/pdf/1511.04587

[5] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Image Super-Resolution Using Very Deep Residual Channel Attention Networks," *arXiv preprint arXiv:1708.02209*, 2017. [Online]. Available: https://arxiv.org/pdf/1708.02209

[6] X. Tong, G. Li, R. Tao, Q. Lu, and Y. Wang, "Image Super-Resolution Using Dense Skip Connections," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 4809–4817, 2017. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2017/papers/Tong_Image_Super_Resolution_Using_ICCV_2017_paper.pdf

[7] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "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 (CVPR)*, pp. 1874–1883, 2016. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Shi_Real-Time_Single_Image_CVPR_2016_paper.pdf

[8] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, *et al.*, "NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[9] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.

[10] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse representations," in *Proceedings of the 7th International Conference on Curves and Surfaces*, 2010.

[11] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 416–423, 2001.

[12] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5197–5206, 2015.

[13] S. J. Reddi, S. Kale, and S. Kumar, "A proof of local convergence for the Adam optimizer," in *2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 126–130, 2019.

[14] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," *arXiv preprint arXiv:1802.08797*, 2018. [Online]. Available: https://arxiv.org/abs/1802.08797

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

[16] B. Knyazev, W. S. Liu, A. Lin, M. Sepliarskaia, C. Häne, and G. W. Taylor, “Image Super-Resolution Using Cross-Scale Attention,” *arXiv preprint arXiv:1608.06993*, 2016. [Online]. Available: https://arxiv.org/pdf/1608.06993