Feedback Residual Dense Network for Image Super Resolution

Mannan Bajaj (24/A04/048)

1. Introduction:

Image Super Resolution is a classical problem in computer vision, where the goal is to increase the resolution of an image, while maintaining its content and details as much as possible. Historically there are numerous methods use for this problem, with each one being better in some aspects than the previous one. Starting from sparse coding methods while being simpler, required high computational costs and quite a lot of algorithmic work for the feature learning process, then came shallow CNNs like SRCNN, making computation cheaper, but not yielding better results than its successors, then came deeper CNNs, leveraging gradient clippings, skips connections and other techniques to achieve convergence even with a lot of layers, then came hierarchical approaches (top to bottom) like RDN (Residual Dense Network) and bottom up approach in SRFBN (Super Resolution Feedback Block Network), which will both be talked about in the proposed model. The model which is named FRDN (Feedback Residual Dense Network) leverages both the technologies namely RDN and SRFCN and creates a best-of-both-worlds scenario for FRDN to thrive. It learns low level features using SFENet from RDN, then the feature maps go through several RDBs and UpNet, all while the training process is done in recurrent feedback loops with hidden state from the previous iteration passed on to the current for better learning and optimization. It uses L1 loss for optimization (same as SRFBN). The final evaluation metrics are as follows:

2. Thought Process:

Image Super Resolution being a supervised learning task, requires the solution to come up with as accurate a mapping between low resolution image and its high-resolution counterpart as it possibly can. Several solutions try to strengthen their learning process of this mapping.

2.1. Literature Review:

1. **Sparse Coding Methods**: These methods use a smaller number of features if we compare them to deep learning methods, but to learn these features, they require complex algorithms backing this process, which are extremely expensive from a computation standpoint.
2. **Interpolation Based Methods**: These methods include shallow as well as deep CNN based models which came out from 2014 to 2017, these models use bicubic interpolation for preprocessing of the image, which raises two issues: (i). Bicubic Interpolation is computationally expensive. (ii). The model learns a mapping from Interpolated LR image to the HR image instead of the original LR image, which limits its performance.
3. **RDN**: The Residual Dense Network (RDN) leverages residual and dense connections to enhance image super-resolution. It combines hierarchical feature extraction with deep supervision, allowing efficient gradient flow and improved reconstruction of high-resolution images. RDN's architecture excels in capturing both local and global features, achieving superior performance on benchmark datasets by balancing computational efficiency and visual quality.
4. **SRFBN:** The Super-Resolution Feedback Network (SRFBN) introduces a feedback mechanism to iteratively refine image details for super-resolution tasks. By incorporating recurrent connections and feedback blocks, SRFBN enhances feature reuse and error correction across iterations. This approach results in sharper, more detailed high-resolution images, outperforming traditional methods on complex textures while maintaining robustness across various scaling factors.

2.2. Coming up with the approach:

With Residual Dense Network's excellence in hierarchical feature extraction with dense but efficient gradient flow and contiguous memory mechanism leveraging the input of the previous RDB at any given RDB, the whole structure of the

RDN with SFENet, RDBs, DFF and UpNet, it works well on learning complex mapping between LR and HR images. But since the task in hand suits a feedback-based model more, the residual dense network mechanism should work better with a recurrent feedback loop implemented alongside. It does not exactly follow the usual bottom-up approach as discussed with respect to SRFBN, but it works on the same recurrent feedback loop structure, It runs t iterations (t is a hyperparameter, we need to set), optimizing the output of each iteration separately then passing on a hidden state to the next iteration, so that the next iteration can improve on the features learned in the previous iteration, it works better than the existing architectures because it leverages the powerful feature refinement of the used RDBs repeatedly.

3. Blockers:

1. **Small Dataset**: Most super-resolution models are trained on large datasets like DIV2K or Flickr2K with thousands of image pairs. However, my dataset was limited to only 100 pairs, which restricted the model's ability to learn diverse features and generalize effectively, the model had to be dense so that it can learn generalised mapping even without a huge dataset.
2. **Too many research paper options**: The super-resolution domain is dense with advanced models — from RDN, SRFBN, RCAN, to SwinIR. Choosing one for implementation was overwhelming and time-consuming, especially when constrained by compute and dataset size.
3. **Computation Constraints**: Training feedback-based models with deep RDBs and large image sizes requires significant GPU memory. On modest hardware (like Colab or a single GPU), I had to use small batch sizes and also keep the model not too deep for the available computational constraints.
4. **Feedback Loop Debugging**: Implementing the feedback loop was technically challenging. Maintaining and correctly updating the hidden state across iterations required careful design to prevent memory leaks or inconsistent gradients. Ensuring weight sharing across RDBs while preserving iterative refinement logic added complexity. Debugging convergence issues due to recursive connections made training more sensitive and less stable than typical feedforward networks.

4. Approach

4.1. Shallow Feature Extraction (SFENet): The architecture begins with a shallow feature extraction module composed of two consecutive convolutional layers. This module processes the low-resolution (LR) input to extract a low-level feature map F_o, which remains constant throughout the feedback iterations and provides foundational details for reconstruction.

4.2. **Feedback Loop and Hidden State:** The core of the model operates as a recurrent feedback loop unrolled for T iterations. At each iteration t, the model receives two inputs: the fixed shallow feature F_o, and the hidden state H_{t-1}, which stores progressively refined features. These two are concatenated and passed through a 1×1 convolutional layer to fuse and compress them into a single representation.

4.3. **Residual Dense Blocks (RDBs):** The fused representation is processed by a stack of Residual Dense Blocks (RDBs), which form the shared feedback refinement block. These blocks leverage dense connectivity and local feature fusion to effectively extract hierarchical information. Their weights are shared across all iterations to maintain parameter efficiency and feedback consistency.

4.4. **Channel Attention Blocks (CAB):** Within the RDB itself, at the very end after concatenation of the input with the resulting feature map, there's a CAB which performs global average pooling, it enhances the important features and suppress noise to some extent so that our model doesn't focus on anything unnecessary, this is vital because evey noise that's generated will be carried on as it is to the SR image, it there is no attention layer.

4.5. **Upsampling Network (UPNet):** The refined feature HtH_tHt is passed through the upsampling module, which uses convolutional layers followed by PixelShuffle operations. The network predicts a residual image, which is then added to a bicubic-upsampled version of the input for final super$-resolved\ output$ I_{sr}^{t}.

4.6. **Preprocessing and Training Procedure:** The dataset consists of 100 paired images, with 80 used for training and 8 for validation. All HR images are dynamically cropped to align with their corresponding LR dimensions at a scale factor of 4. The model was trained for 10 epochs using the Adam optimizer (learning rate: 1e-4) and L1 loss averaged across all iterations.

4.7. **Evaluation Metrics:** Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) were used to evaluate performance on a separate test set of 20 images. PSNR measures pixel-level fidelity, while SSIM evaluates perceptual quality by comparing structural similarity between the predicted and ground truth images.

5. Results:

5.1. Evaluation Metrics:

PSNR (Peak Signal-to-Noise Ratio): PSNR quantifies pixel-wise similarity between the super-resolved and ground truth images. It measures the logarithmic ratio of maximum pixel intensity to mean squared error. This metric is widely used in super-resolution tasks to evaluate fidelity and reconstruction quality, especially for tasks requiring sharp, high-accuracy outputs.

SSIM (Structural Similarity Index): SSIM assesses perceptual similarity by comparing structural, luminance, and contrast components between images. Unlike PSNR, it correlates better with human visual perception. It was chosen for this task to evaluate how well the model preserved important textures and structural details in urban images beyond simple pixel accuracy.

```python
def forward(self, x):
    f1 = self.sfe1(x)
    F0 = self.sfe2(f1)

    hidden = torch.zeros_like(F0)

    upsampled_lr = nn.functional.interpolate(x, scale_factor=self.upscale_factor, mode='bicubic', align_corners=False)

    outputs = []
    for t in range(self.T):
        fused_input = self.feedback_fuse(torch.cat([F0, hidden], dim=1))
        hidden = self.refine(fused_input)
        residual = self.upnet(hidden)
        sr = residual + upsampled_lr
        outputs.append(sr)

    return outputs
```

Fig. 1. Code Snippet of Forward propagation process in FRDN architecture.

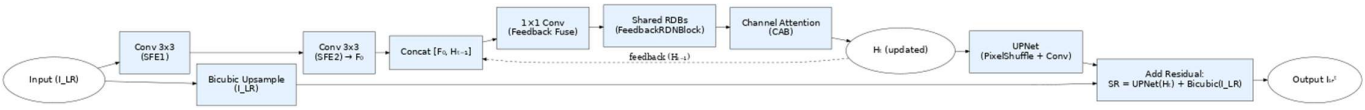

Fig. 2. Architecture diagram of FRDN, highlighting feedback loops and Residual Dense Blocks (RDBs).

5.2. Results for Feedback Residual Dense Network (FRDN)

This section outlines the initial performance of the proposed Feedback Residual Dense Network (FRDN) for image super-resolution. The FRDN architecture was developed by integrating concepts from the Residual Dense Network (RDN) and the Super-Resolution Feedback Network (SRFBN), aiming to leverage RDN's strong feature extraction within an iterative refinement framework inspired by SRFBN.

5.3. Experimental Setup Overview

- **Upscaling Factor:** x4
- **Number of Feedback Iterations (T):** 4
- **Number of RDBs within each iteration's refinement block:** 6

- **num_features**=64: The number of feature maps after the initial shallow feature extraction.
- **num_blocks=6**: the number of RDBs in the Main Refinement Block that is executed *within each iteration* of the feedback loop.
- **num_layers=4:** The number of convolutional layers (C) *inside each* RDB.
- **Loss Function:** L1 Loss, averaged over T iterations
- **Optimizer:** Adam with learning rate 1e-4

Performance Metrics: The performance of the FRDN model was evaluated using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM).

5.4. **Results:** The following results were obtained for the FRDN model on the specified test dataset:

- **After 5 Epochs of Training:**
  - **PSNR:** 27.11 dB
  - **SSIM:** 0.7059
- **After 10 Epochs of Training:**
  - **PSNR:** 27.68 dB
  - **SSIM:** 0.7390

Initial Observations: The initial results demonstrate that the FRDN model is learning and improving with increased training epochs. The PSNR increased by 0.53 dB and SSIM by 0.0308 when training was extended from 5 to 10 epochs.

Fig. 3. Visual comparison of Low-Resolution (LR) input, FRDN-predicted Super-Resolution (SR) output, and Ground Truth (HR) images

	Bicubic	SRCNN	MemNet	RDN	SRFBN	FRDN (Ours)
PSNR	23.14	24.52	25.50	24.92	26.73	27.68
SSIM	0.6577	0.7221	0.7630	0.7362	0.8043	0.7390

Fig4. Comparison of existing models vs our model, these values are taken from reference [8.3]

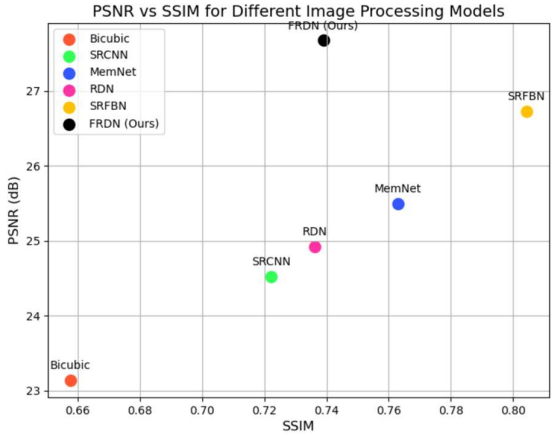

Fig5. Graphical Representation of the comparison made above

6. Comparative Study:

PSNR and SSIM values for different state of the art models trained on different datasets, **extracted from the SRFBN paper from 2019[8.3.]**. The performance of FRDN can't be directly compared with these values because FRDN is trained on a completely different dataset, which is completely different from these datasets in size and complexity. Although the Urban100 has almost similar characteristics as our very own dataset, it has the same level of complexity of images, the same scale at which we're creating high resolution images and also the exact same number of examples. So the performance of the state-of-the-art models on Urban100 can be directly compared to the performance of FRDN on our dataset. We can see that FRDN performs better than SRFBN, RDN, SRCNN and Bicubic interpolation having either better or almost equal SSIM and a better PSNR score. And this score will only increase with increasing number of epochs and allowing the model to train for longer time as the existing one.

This is because of the fact that it learns a more complex mapping from LR to HR than the rest of the models, RDN learns hierarchical features exceptionally but doesn't gather information about the actual input image throughout all the RDBs, so over the forward pass, the low level features somehow get lost throughout the process, In SRFBN, the model amazingly implement a recurrent feedback loop, but being more focussed on large scale tasks, it can't perform with that level of excellency for small scale tasks because it doesn't focus of feature learning as much as RDN or FRDN.

6.1. **Strengths:** FRDN fixes both of these shortcomings and implements a model that learns low level features very efficiently and also takes care that the low-level features don't get lost through the training period, that's why it performs better than RDN and SRFBN.

6.2. **Shortcomings:** Considering FRDN performs well on the given dataset, it took 14-15 minutes on average for a single epoch in the training process, because of its complex structure containing multiple RDBs, T iterations of recurrent feedback loop, and dense connections. It is not very efficient to work with large datasets with the current architecture. But dividing the data into batches can ease the process quite well. Where it lacks is the latest multi-head self-attention concept which significantly reduces training time because it works wonders on GPU, FRDN can't yet leverage that, but with some tweaks, it can sure compete with state-of-the-art transformer models.

7. Future Prospects:

7.1. **Curriculum Learnin**g: Curriculum learning, which gradually increases the difficulty of the learned target, is well known as an efficient strategy to improve the training procedure. Early work of curriculum learning mainly focuses on a single task, but it can be implemented for multiple tasks in a sequential manner. We could use a classification model to classify the images from our dataset into easy, medium and hard on the basis of their complexity when scaled by a factor, and then we can improve the training process of our model by feeding it the easy training examples first, then the medium ones, then the hard ones. And we can also deliberately misclassify some examples so that it can also generalize to every possible input.

7.2. **GAN-like optimization technique**: Since we're already running T iterations for every forward propagation of the overall architecture, so the feature map after every iteration can be considered as synthetic data with respect to the high resolution image, so our entire model can work as a generator for a GAN, and a discriminator can be designed which will give a score to the similarity of the generated feature map in every iteration with the high resolution image, be it SSIM. And it will try to maximize this score, this will penalize the generator if the synthetic images are not quite similar to HR images. This generator and

discriminator optimization will work after every iteration and after T iterations, our model will produce a much better SR image.

7.3. **Masked Image Modelling:** Masked image modelling is also a powerful concept which has been a part of transformers like BERT for masked modelling of textual inputs. It encourages the model to better learn the image structure and important features that constitute it, by masking part of the input and letting the model deal with generating it in its own way helps improve the performance quite a bit. While models like SRCNN, MemNet, RDN work well on just smaller scales, by adding masked image modelling, FRDN will be able to work on even larger scales than it currently can and will be useful for other image restoration tasks as well.

8. References:

8.1. Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In ECCV, 2014

8.2. Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In CVPR, 2018.

8.3. Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, Wei Wu. Feedback Network for Image Super-Resolution. In CVPR, 2019.

8.4. MPRNet: Multi-Path Residual Network for Lightweight Image Super Resolution. Armin Mehri, Parichehr B.Ardakani, Angel D. Sappa. In CVPR, 2020.