Image Super Resolution Using SRCNN and ResDenseNet

Hongyang Li, Yimeng Zhang, Yumeng Lu,

Abstract

Single image super-resolution is a challenging image processing task, which aims at recovering a high-resolution image from a single low-resolution image. Due to the resolution limitation of large-scale imaging equipment, super-resolution technology enjoys wide range of real-world applications, especially in aerospace and medical fields. One big challenge is how to let the machine know that the output images are high-resolution ones. Here, the key idea is to transform the unsupervised task into a supervised one by generating input low-resolution images through a pre-process of downsizing and upsizing high-resolution images, and the high-resolution ones are taken as the ground truth images. In this paper, we implement two supervised CNN structures to solve the super-resolution problem, which are the super-resolution convolutional neural network (SRCNN) and the residual dense network (RDN). To improve the model training efficiency, we optimize the SRCNN model with batch normalization. Simulation results show that our SRCNN, SRCNN+BN and RDN models all improve the quality of output images. Among them, SRCNN has the best PSNR and SSIM performance, but RDN has great potential to achieve greater performance by increasing the number of training epoches and the size of training set. Our codes are attached here.

1 Introduction

Image super-resolution is the task of recovering a higher-resolution image from a lower-resolution image. This image processing technique is important in computer vision and image processing. It enjoys a wide range of real-world applications, such as medical imaging, satellite imaging, surveillance, and security, amongst others. In this paper, we focus on supervised learning methods with CNNs to solve the unsupervised super-resolution problem, by using higher-resolution images as target and lower-resolution images as input.

Among several strategies for image super-resolution, SRCNN is a popular machine learning method to process lower-resolution images into high-resolution ones. Proposed by Dong in 2015 [1], this model encodes photographic characteristics by using learned kernels with nonlinear activations and can add the structures lost in the low-resolution input. This model is a simple model with only three layers, but it performs well in image super-resolution tasks and has been widely applied.

However, because SRCNN does not make full use of the hierarchical features from the original low-resolution images, RDN was considered to deal with the image super-resolution problem. In 2018, Zhang proposed the RDN model for image super-resolution, which fully exploit the hierarchical features from all the convolutional layers [3]. The model involves several residual dense block (RDB) to extract abundant local features via dense connected convolutional layers and allow direct connections from the state of preceding RDB to all the layers of current RDB.

Therefore, in this paper, we try to implement and improve the two models introduced above, and want to compare their super-resolution performance. Our performance testing is based on PSNR and SSIM metrics. The former one is based on the MSE loss, and the latter one analyzes structural similarity between output images and ground truth images.

2 Datasets

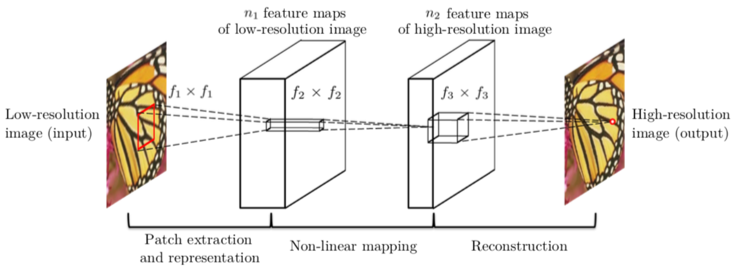
Both SRCNN and RDN's training dataset are generated from the 91-image dataset. For SRCNN, we decompose the 91-image dataset into 21884 sub-images with an extraction stride of 14, and each sub-image is of width and height 33 pixels. As is evaluated in Dong's paper [1], generating training dataset in this way is able to capture sufficient variability of natural images and avoid overfitting problem given the relatively small network of SRCNN.

For RDN, given the constraints on GPU resources, we only crop each image in the 91-image dataset by four corners and a central crop. Yet it is better to include more training data considering the high complexity of RDN model. We will further discuss about this problem in section 5.

For model validation, we use the Set14 dataset for both SRCNN model and RDN model. To test the performance of SRCNN and RDN, we use the Set5 dataset. We first downsize all the images from the training sets and validation sets using bicubic with scale factor $\frac{1}{3}$, and then upsize them with factor 3 to generate low resolution image.

3 Models

3.1 SRCNN



n_1 feature maps of low-resolution image
n_2 feature maps of high-resolution image
$f_1 \times f_1$
$f_2 \times f_2$
$f_3 \times f_3$
Low-resolution image (input)
High-resolution image (output)
Patch extraction and representation
Non-linear mapping
Reconstruction

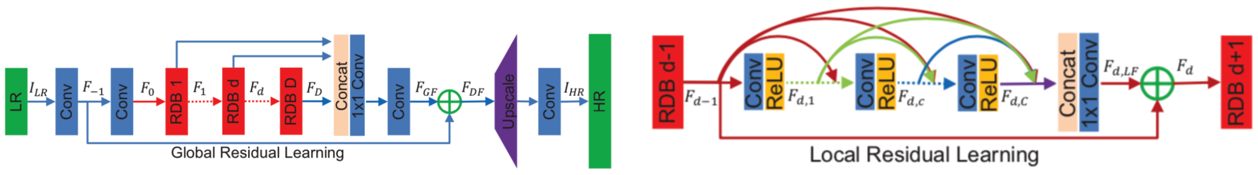
3.1.1 SRCNN Architecture

The SRCNN model proposed by Dong in 2014 consists of three operations: patch extraction, non-linear mapping, and reconstruction. The structure of the SRCNN model is illustrated in above figure. During patch extraction, filters extract patches from the low-resolution image and form a set of feature maps. Non-linear mapping generates a new set of feature maps that represents high-resolution patches. The reconstruction step aggregates the high-resolution patch-wise representations passed from the non-linear mapping step, and in this way, the predicted high-resolution image is created. The goal of our training process is to make the output image as similar as our ground truth high-resolution image.

3.1.2 SRCNN Settings

In Dong's paper [1], different settings of the hyperparameters in the SRCNN are discussed. Among them, we choose the setting with the best testing performance as the hyperparameters in our SRCNN model. The first layer involves 64 filters and each filter is of size 9×9, with image padding $= 4$. In this way, the first layer expands a three-channel image into 64 feature maps. The second layer applies 32 filters with size 5×5, and thus turns the 64-channel image with 2 padding to a 32-channel image. The third layer uses a 5×5 kernel to generate the output image. Both the first and second convolutional layers are succeeded by a ReLU activation function.

3.2 Residual Dense Network Model



3.2.1 Overall RDN Architecture

The Residual Dense Network model proposed by Zhang in 2018 consists of four parts: SFENet, RDBs, DFF, UPNet. It aims to make full use of the input low resoultion image's information by learning both local and global features, together with residuals. Given the input low resolution image I_{LR}, the model aims to generate a high resolution image I_{HR}. In SFENet phase, I_{LR} goes through two Conv layers with filter size 3×3 to extract shallow features, the corresponding outputs are F_{-1} and F_0. Then in the RDBs phase, the model goes through D residual dense blocks(RDBs). Here D is a hyperparameter. For block i, F_{i-1} is the input which is calculated from the previous block and F_i is the output. The detailed structure inside a RDB block will be discussed in the next section. After D residual dense blocks, the model generates a sequence of outputs, $\{F_0, F_1, F_2, \cdots F_D\}$. Concatenating the preceding results and go through a 1×1 and a 3×3 Conv layers, the model generate F_{GF}. Yet F_{GF} left F_{-1} untouchable. So for the final feature-map of DFF phase, the model concatenates F_{GF} and F_{-1}: $F_{DF} = F_{GF} + F_{-1}$. For the upsampling phase, the model utilizes ESPCN in UPNet followed by one Conv layer to enlarge image to the original size.

3.2.2 Residual Dense Block and Settings

Inside each residual dense blocks(RDBs), the input F_{d-1} goes through C Conv layers, each layer with G feature-maps to generate F_d (except the first layer only has G_0 feature-maps). Here d denotes the number of blocks and C, G, G_0 are hyperparameters. We denote block d, layer i's output as $F_{d,i}$. For layer i, it utilizes all the preceding outputs: $F_{d,i} = \sigma[W_{d,i}(F_{d-1}, F_{d,1}, F_{d,2}, \cdots F_{d,i-1})]$. Here $W_{d,i}$ is the weights corresponding to block d and layer i that concatenates the preceding results linearly. σ denotes the activation function, by default is ReLu. For each of the C layers, the model generates a corresponding $F_{d,i}$. The final stage of a RDB first passes all these results to a 1×1 Conv layer in the d-th RDB: $F_{d,LF} = H^d_{LFF}[F_{d-1}, F_{d,1}, F_{d,2}, \cdots, F_{d,C}]$ to learn the local feature fusion. Then the network concatenates previous outputs with F_{d-1} to output F_d: $F_d = F_{d-1} + F_{d,LF}$. As discussed in the original paper, the larger C, D, G are, the better the model's performance. So in our implementation, we use $D = 16$, $C = 8$ and $G = 64$ as our hyperparameter value.

3.3 Metrics

3.3.1 Loss Functions for SRCNN

To train the SRCNN model, we use mean squared error (MSE) as our loss function. The formula is given as

$$L_{MSE}(\theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathrm{X}_i;\theta) - \mathrm{Y}_i||^2, \tag{1}$$

where n is the number of SRCNN training samples, Y_i is the ground truth high-resolution image i, X_i is the input low-resolution image i, θ represents model parameters, and $F(\mathrm{X}_i;\theta)$ is the reconstructed image. The aim for training SRCNN is to $\min L_{MSE}$.

3.3.2 Loss Functions for RDN

To train the RDN model, we use the L_1 loss function. The formula is given as

$$L_{MAE}(\gamma) = \frac{1}{m}\sum_{j=1}^{m} |G(\mathrm{X}_j;\gamma) - \mathrm{Y}_j|, \tag{2}$$

where m is the number of RDN training samples, Y_j is the ground truth high-resolution image j, X_j is the input low-resolution image j, γ represents RDN model parameters, and $G(\mathrm{X}_j;\gamma)$ is the reconstructed image. The aim for training RDN is to $\min L_{MAE}$.

3.3.3 PSNR

We use two evaluation metrics to evaluate the similarity between the output image and the ground truth image. We firstly choose the Peak Signal-to-Noise Ratio (PSNR) to show the performance of SRCNN and RDN respectively. PSNR is commonly used to quantify reconstruction quality for images to lossy compression, and it performs well for simulating human perception of reconstructed image quality. PSNR is defined as

$$PSNR = 20 \cdot \log_{10} MAX_I - 10 \cdot \log_{10} MSE, \tag{3}$$

where MAX_I is the maximum possible pixel value of the image. Thus, the higher PSNR illustrates the better reconstruction performance.

3.3.4 SSIM

Another evaluation metric we use is the Structure Similarity Index (SSIM). Proposed in Wang's paper [2], SSIM tests the similarity between the output image and the ground truth image based on structural similarity. It quantifies human perception differences from three perspectives: luminance comparison, contrast comparison, and structure comparison. The formula of SSIM between the output image x and the ground truth y is given as

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)}, \tag{4}$$

where μ_x and μ_y are the average of x and y respectively, σ_x and σ_y are the variance of x and y respectively, and σ_{xy} is the covariance of x and y. $C_1 = (0.1L)^2$, $C_2 = (0.3L)^2$ are two variables to stabilize the division with weak denominator, and L is the dynamic range of the pixel-values. The higher SSIM means that the output image has higher structural similarity with the ground truth high-resolution image, which implies the tested model has better reconstruction performance.

4 Results

4.1 Baseline Models

We implemented three different non-neural baselines: bicubic, bilinear, and nearest neighbor.

4.2 Settings for Gradient Descent

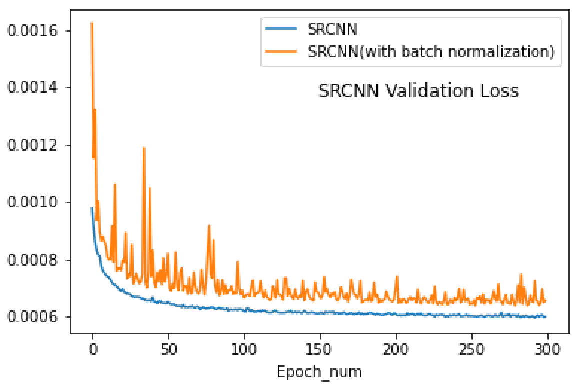
We use mini-batch gradient descent to train both the SRCNN model and the RDN model. For SRCNN, the hyperparameters for the mini-batch gradient descent are as follows: the learning rate is set as $1e^{-5}$, each batch contains 16 training samples, and the number of epoches is 300. For RDN, the learning rate and epoch number is chosen the same as SRCNN, but each batch only contains 4 training samples. To further increase the parameter learning efficiency and accuracy, we use Adam optimizer for both SRCNN and RDN.

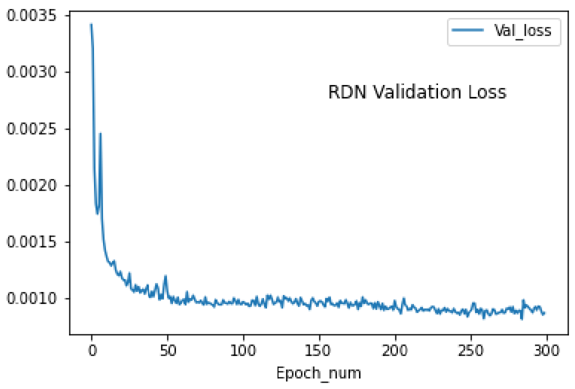
4.3 Batch Normalization for SRCNN

To accelerate our training speed of the SRCNN model, we add batch normalization to its first layer and second layer. The result for the first epoch shows a great time reduction. The basic SRCNN model takes 8.22 seconds to go through the first epoch, while the batch normalized SRCNN model just take 4.44 seconds. However, for the rest of the epochs, batch normalization does not display an obvious time advantage.

4.4 Validation Loss Comparision: SRCNN V.S. SRCNN+BN V.S. RDN

The following two figures illustrate the validation loss for SRCNN, SRCNN+BN and RDN. Our training datasets do not overfit the three models because the validation losses are all decreasing. The left figure illustrates that the validation loss for basic SRCNN is smoother, while the validation loss for SRCNN+BN has frequent fluctuations. Also, the validation loss for SRCNN+BN is a bit larger than the basic SRCNN model.

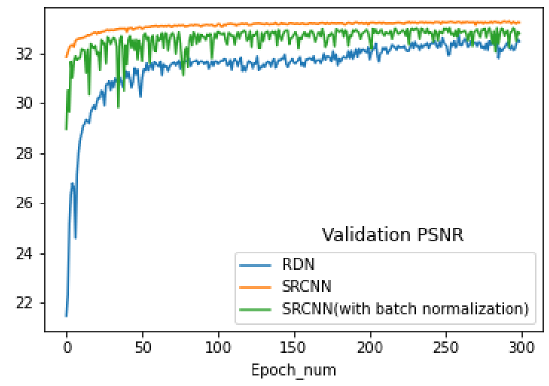




4.5 Performance Comparision: SRCNN V.S. SRCNN+BN V.S. RDN

The following two figures illustrate the performance of SRCNN, SRCNN+BN and RDN based on PSNR and SSIM metrics. By comparing their PSNR and SSIM values, we can find that SRCNN with batch normalization does not perform better than basic SRCNN. Instead, the super-resolution images it generates have a bit lower quality than what basic SRCNN generates. Moreover, our RDN model performs worse than SRCNN model from the perspectives of PSNR and SSIM.

However, by looking at the growth trends of PSNR and SSIM, we find that after about 100 epoches, the growth trends for SRCNN and SRCNN+BN become very small and are even almost equal to 0, while the growth trend for RDN is still very obvious even until reaching 300 epoches. Thus, we infer that if we increase the epoch number for RDN, it may perform much better. A further discussion for improving the RDN model is in section 5.



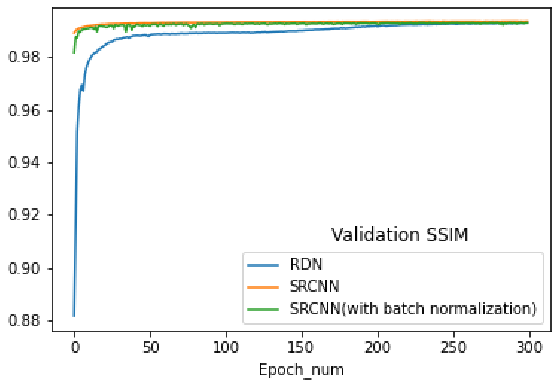


Table 1 below shows the validation performance and test performance for SRCNN, SRCNN+BN, RDN and the three baseline models (Bicubic, Bilinear, and Nearest Neighbors). We need to point out that the first three metrics: the max Val PSNR, the max Val SSIM and the max Val Loss respectively represent the maximum validation PSNR, the maximum validation SSIM and the minimum validation SSIM via all the epoches, while the max test PSNR and max test SSIM respectively represent the maximum test PSNR and maximum test SSIM among the test image samples. During the training process, we take the parameters in the epoch according to the maximum validation PSNR as the final model parameters. For the basic SRCNN model, the best epoch with the maximum validation PSNR = 33.266 is epoch 289. For the SRCNN model with batch normalization, the best epoch with the maximum validation PSNR = 33.003 is epoch 253. For the RDN model with batch normalization, the best epoch with the maximum validation PSNR = 32.767 is epoch 284.

Moreover, comparing our models with the three baseline models (bicubic, bilinear and nearest neighbour) from the perspectives of test PSNR and test SSIM, we find that our SRCNN models and RDN model improve the super-resolution performance. Furthermore, comparing the testing performance of basic SRCNN, SRCNN+BN and RDN, we find that our basic SRCNN model performs better than the rest two models by PSNR, and it achieves a test PSNR of 35.973 db. However, even though the test PSNR for RDN is much lower, it is still very high, reaching 35.592 db. When looking at the test SSIM, the three models have similar performance, but our basic SRCNN has a little advantage. Actual outputs for testing images are illustrated in page 7 of this paper.

Experiment	Val PSNR (Max)	Val SSIM (Max)	Val Loss (Min)	Test PSNR (Max)	Test SSIM (Max)
Basic SRCNN	33.266	0.9935	0.0005950	35.973	0.9979
SRCNN+BN	33.003	0.9930	0.0006371	35.680	0.9970
RDN	32.767	0.9934	0.0008141	35.592	0.9977
Bicubic	/	/	/	34.672	0.9970
Bilinear	/	/	/	33.616	0.9960
NN	/	/	/	30.087	0.9890

Table 1: Comparision of Model Performance

5 Discussions and Future Work

Though the SRCNN outperforms RDN slightly based on 300 training epochs, it does not suggest RDN is less efficient than SRCNN. Considering the high complexity of RDN (at least $((C-1) \times G + G_0) \times D + 2 + 2 + 1$ convolutional layers), it requires more training data. Yet to maintain consistency in training set to compare both models' performance as well as the constraints of GPU resources, we use FiveCrop method and generating 455 subimages from the 91-image set, which is far less than 21884 subimages in SRCNN. Another reason is that as was clearly stated in Dong's paper, the 91-image training set is the almost the best training set for SRCNN. The RDN's model, however, should use DIV2K with 800 training images and 100 validation images as default datasets. In this paper, with only around 2% size of SRCNN's training data, RDN performs as well as SRCNN. Therefore, it is foreseeable that with enough training data, RDN will outperforms SRCNN significantly.

We downloaded the pre-trained weights by the original inventor of RDN from GitHub-RDN and tested it on the same Set5 image. The corresponding results have a max-PSNR of 36.54 which is higher than our current RDN and echoes our previous assumption. Moreover, in terms of the fluctuations in the batch normalized SRCNN, we attribute it to SRCNN's self-supervised model-type, where each pixel's value matters. In such scenario, batch normalization might not be that helpful since it changes the value of pixels.

6 Acknowledgements

We would like to thank our professor Li Guo for her advice on improving the model.
We would like to thank our learning assistant Yichen (William) Huang in configuring HPC environment.

References

[1] Chao Dong et al. "Image Super-Resolution Using Deep Convolutional Networks". *CoRR abs/1501.00092 (2015)*. arXiv: 1501.00092.

[2] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, April 2004.

[3] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual Dense Network for Image Super-Resolution. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

(a) Ground Truth Image

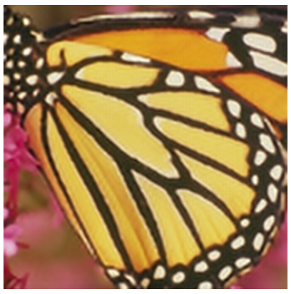
(b) Bicubic

(c) RDN

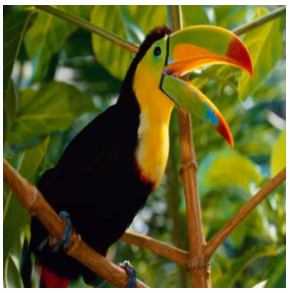
(d) SRCNN

(a) Ground Truth Image

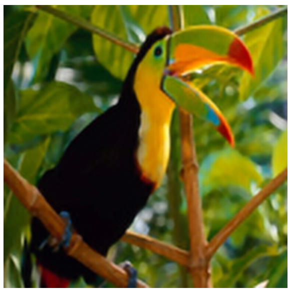
(b) Bicubic


(c) RDN

(d) SRCNN


(a) Ground Truth Image

(b) Bicubic


(c) RDN

(d) SRCNN

(a) Ground Truth Image

(b) Bicubic

(c) RDN

(d) SRCNN

(a) Ground Truth Image

(b) Bicubic

(c) RDN

(d) SRCNN