Study of Deep Convolutional Networks and Enhanced Residual Networks for Image Super-Resolution

Maryam Gholami Shiri
Maryam.gholamishiri@studenti.unipd.it

Claudio Palmeri
claudio.palmeri@studenti.unipd.it

Abstract

In this study, we explore and compare the performances of two different models in an image super-resolution task. The former, first proposed by Chao Dong et al. [4], implements a novel deep-learning framework for super-resolution. This framework is built upon a deep convolutional neural network (CNN) that learns a direct end-to-end mapping between low and high-resolution images. This method stands out by jointly optimizing all network layers, diverging from traditional sparse-coding-based approaches. The latter, first proposed by Bee Lim et al. [9], introduces the Enhanced Deep Super-Resolution Network (EDSR). This model consists of a Residual network with ad-hoc modified residual blocks for this specific task. We then trained the models and compared the results on Set5 and Set14: Two standard benchmark dataset for super-resolution models.

1. Introduction

Image super-resolution is a machine-learning task that develops a model capable of enhancing the quality of an image by increasing the resolution of the original photo without losing its content and defining characteristics.

This famous and often tackled problem in computer vision has the following property: there is no "correct" solution since the task is inherently ill-defined. This happens because more than one image will be down-sampled in the exact low-resolution counterpart since our task is equivalent to mapping a high-dimensional space into a lower one. Thus, the mapping cannot be injective, and the inverse function needs to be better defined. Nonetheless, much research has been done in this field thanks to the many possible applications of this.

- **Surveillance applications**: We can enhance security camera footage and eventually pair it up with a face recognition algorithm that will benefit from the improvements [6].
- **Medical applications**: We can improve the resolution of any image taken during a medical exam, such as X-rays, to help the doctor make a better diagnosis [7].
- **Computer vision applications**: We can use a super-resolution model as a data pre-processing component in any computer vision task where we could benefit from improving the input's resolution.
- **"Historical" application**: We can enhance old photos that were taken using old (and thus less capable) technology.

All these applications stem from the fact that obtaining and memorizing a high-resolution image is more demanding from a memory perspective and often requires more expensive equipment.

We can also observe that we can extend our super-resolution models to make them able to process videos instead of still images by applying super-resolution frame-by-frame.

In our study, we developed two different models:

- CNN: A fully convolutional neural network specifically designed for this task. This model takes an already up-sampled, using more traditional and deterministic methods, low-resolution image and betters it.
- ResNet: Residual blocks are themselves fully convolutional. We can combine this with a transposed convolution, which is capable of increasing the scale of our inputs in order to match the expected output.

We trained different models for different down-sampling factor. In this paper we considered 2x,4x as possible factors.

In order to compare the performances we tested our models both on a test set made from a subset of the original training set that wasn't seen by our model and on standards benchmark datasets such as Set5 [2] and Set14 [12]. Our code was posted on Git-Hub

2. Related Work

Single Image Super Resolution SISR methods aim to recover an HR image from a single LR input image. We

can classify recent popular SISR methods into prediction models, edge-based techniques, statistical methods, and patch-based (or example-based) approaches [11]. Patch-based methods, known for their superior performance, have attracted significant research attention. In this method, patches can be cropped from the training images to learn mapping functions [8].

Recently, deep learning techniques have been successfully applied on SR. The pioneering work is termed the Super-Resolution Convolutional Neural Network (SRCNN) proposed by Dong et al. [3]. Motivated by SRCNN, problems such as face hallucination and depth map super-resolution have achieved state-of-the-art results [5].

This demonstrated that an end-to-end deep learning approach could effectively map low-resolution images to high-resolution counterparts. This method outperformed traditional techniques in terms of accuracy and proved more practical for real-world applications.

Building on SRCNN's foundation, researchers have developed more advanced CNN architectures to enhance SR performance. In SRCNN, the network must pre-serve all input detail since the image is discarded and the output is generated from the learned features alone. With many weight layers, this becomes an end-to-end relation requiring very long-term memory. For this reason, the vanishing/exploding gradients problem can be critical. To solve this problem, Kim et al [8]. introduced a residual learning framework. Mao et al [10]. extended this concept with an encoder-decoder architecture featuring symmetric skip connections, facilitating faster convergence and improving training efficiency.

A critical challenge in CNN-based SR methods involves managing multi-scale inputs effectively. Traditional approaches often require the input image to be upsampled before feeding it into the network, as seen in methods like VDSR. This preprocessing step increases computational load. Recent approaches have shifted towards incorporating upsampling layers at the end of the network, reducing the computational burden without compromising model capacity. However, these methods must still address multi-scale training within a single framework.

3. Dataset

In our project, we utilized training data from the DIV2K dataset. This dataset has recently gained prominence in image restoration due to its high-quality 2K resolution images. The DIV2K [1] dataset comprises 900 meticulously curated training and 100 test images. However, since the ground truth for the test dataset has not been released, we report and compare performance metrics using a subset of the training set. Picture 1 is an example of images present in the DIV2K dataset.

Figure 1. An example of original image taken from the DIV2K dataset.

- CNN: For this model, we followed the following procedure:

  We divided the 900 images of the DIV2K dataset into 880 training images and 20 validation images.

  For each high-resolution training image, we extracted $32 \times 32 \times 3$ RGB patches with a stride of 96. Thus since this dataset contains images of different shapes and a variable number of training example will be extracted from each image.

  The total number of training images is 262773 for an average of 298 patches for each training image. This patch-based approach facilitates efficient memory management and accelerates the training process since we are working with smaller images during the training process.

  In order to compute the corresponding low resolution patches, we down-sampled them and up-sampled them by the same factor using bicubic interpolation. This re-sampling technique considers the nearest 4x4 neighborhood of known pixel values surrounding the target pixel. This method yields smoother images compared to nearest-neighbor or bilinear interpolation.

  The high-resolution validation set was left unchanged, and we applied a similar procedure to the training images, i.e., down and the up-sample with the same scale factor, giving us 20 images for each of the 4 down-sampling factors we considered.

  Finally, since all our data was composed of integers between 0 and 255 we decided to divide each of them by 255 in order to have our variables in the [0,1] interval

- EDSR: For this model, we followed several critical steps to optimize the dataset for super-resolution tasks:

  To generate low-resolution images, we utilized bicubic interpolation. The images were downscaled by a factor of 4.

We extracted RGB input patches of size 48×48 from the low-resolution (LR) images alongside their corresponding high-resolution (HR) patches.

To enhance model robustness, we augmented the training data using random horizontal flips and 90-degree rotations. These augmentation techniques increase the diversity of training samples, thereby improving model generalization.

As part of the preprocessing steps, we subtracted the mean RGB value of the DIV2K dataset from all images. This normalization step helps mitigate variability in the dataset's lighting conditions and color distributions.

4. Methods

4.1. Evaluation metrics and Loss function

The objective of our task is to reconstruct the original image as closely as possible. Thus, we decided to use as the Loss function the Mean Squared Error, which is defined as:

$$MSE_{total} = \frac{1}{N}\sum_{i=1}^{N} MSE(Y_i, Y_i^{pred})$$

where N is the number of training examples, Y_i is the correct output and Y_i^{pred} is the output predicted by the model. more precisely:

$$MSE(Y_i - Y_i^{pred}) = \frac{1}{h * w * 3}\sum_{j=1}^{h}\sum_{k=1}^{w}\sum_{l=1}^{3}(Y_{i,j,k,l} - Y_{i,j,k,l}^{pred})^2$$

Where h and w are the size of the 3 coloured image. We also used another metric to evaluate the model's perperformances: The mean PSNR or Peak to Noise Ratio:

$$PSNR_{total} = \frac{1}{N}\sum_{i=1}^{N} PSNR(Y_i, Y_i^{pred})$$

more precisely:

$$PSNR(Y_i, Y_i^{pred}) = 20 * log_{10}\frac{(MAX_i)}{\sqrt{MSE(Y_i - Y_i^{pred})}}$$

where $MAX_i = 1$ is the maximum intensity in our image, which is one since we divided each pixel value by 255.

We implemented the following two models:

4.2. Image Super-Resolution Using Deep Convolutional Networks

In this study, the author proposes using a tailored deep convolutional neural network for the super-resolution task. The layers are structured as follows:

4.2.1 Patch extraction and representation

Sparse coding methods extract the relevant features of the original images by using an already curated and tested set of dictionaries. This can be thought of as convolving the original image by a set of already pre-determined filters.

Instead of using those filters, we optimize randomly generated filters while training the model. Thus, the first layer of our model tales the original low-resolution input X and transforms it into:

$$F_1(X) = Relu(W_1 * X + B_1)$$

Where '*' denotes the convolution operation and W_1 and $B1$ are, respectively, the weights and the bias of our first convolutional layer. We also apply the activation function $Relu(x) = max(x, 0)$, which serves as a way to make the function not linear.

4.2.2 Non-linear mapping

Similarly, our second layer is structured in the same way:

$$F_2(X) = Relu(W_2 * F_1(X) + B_2)$$

And it mirrors how sparse coding methods use other curated dictionaries to extract the relevant features from the previous layer's output.

4.2.3 Reconstruction

Finally, we compute the final reconstruction layer as

$$F_3(X) = W_3 * F_2(X) + B_3$$

Picture 2 shows the overall structure of our network.

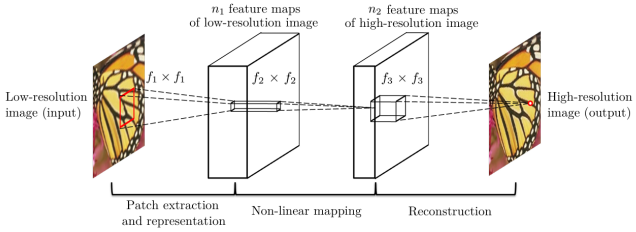

Figure 2. Overall structure of the CNN model.

4.3. Enhanced Deep Residual Networks for Single Image Super-Resolution

The author proposes an enhanced residual network (ResNet) architecture tailored for image super-resolution in this study. Slight modifications were made to the original ResNet to increase model performance and training time. The model is composed of 3 sections:

- Convolutional layer
- Residual blocks
- An Upsampling block

We will delve now into more details about each section:

Convolutional layer An initial convolutional layer that encodes our images' relevant features, which are then processed with a Relu activation function.

Residual Blocks: Residual blocks work by adding to the input of the block itself after having been processed by convolutional layers.

The original proposal for processing in the Residual blocks consists of the following:

- Convolutional layer
- Batch normalization layer
- Applying the Relu activation function
- Convolutional layer
- Batch normalization layer

After adding the original input of the layer with its results, we apply the Relu once again before transmitting the result to the next residual block.

The modified version does not include both batch normalization layers since they drastically impact the training time. It also does not include the application of the last Relu function.

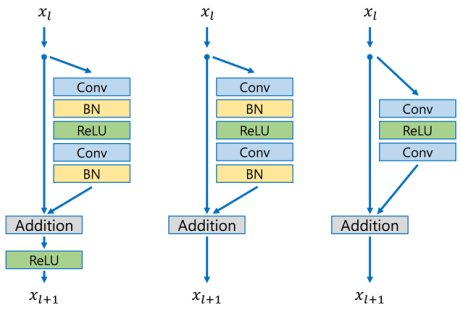

Figure 3. Difference between the proposed EDSR model and the other residual neural networks in the past.

Upsampling Block: This block's purpose is to increase the spatial resolution of its input in order to match the size of the original high-resolution image. To accomplish this, we utilize transposed convolution with a stride of 2. A final convolutional layer is then applied at the very end of the processing.

Picture 4 shows the overall structure of the EDSR model.

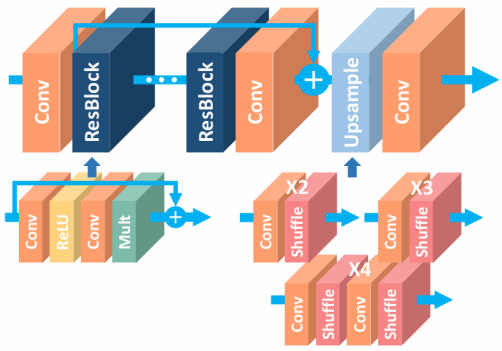

Figure 4. Overall structure of the EDSR model

5. Experiments

5.0.1 CNN

Since reconstructing an image that was compressed with a factor of 2 is necessarily more straightforward than reconstructing an image down-sampled with a factor of 4, we decided to train our model on the former images first.

After doing so, we exploited the already trained model to use it as a starting point for the 4x images. This technique, known as pre-training, permits us to achieve an overall better performance.

We experimented with different learning rates and filter sizes to get the best possible performance on the 2x training set. Then we only trained the better performing model on the other, harder to reconstruct, dataset 2 times:both from scratch and by using the model already trained on the 2 times down-sampled images as pre-training.

We implemented the base model presented while varying its learning rate between 0.001,0.0001, and 0.00001.

This consists of using kernel sizes for the three layers respectively: 9,1,5; several filters equal 128 and 64 in the first and second layers, respectively. We ran the model for ten epochs and printed the results in Figure 5. We can observe that the performance for the model with a learning rate equal to 0.0001 achieves the best performance.

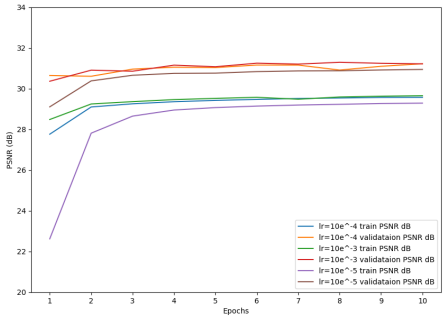

Figure 5. Performance by learning rate.

After fixing the learning rate to 0.0001 we then both tried to increase and decrease the filter sizes to 200, 100 and 64, 32 respectively. Unfortunately increasing it heavily increase both the Ram usage and especially consume more time to training the model.

For this reason and because the model was not improving, we only trained it for four epochs and printed the results in Figure 6. The labels in the plot indicate the number of filters present in the first layer of our convolutional network.

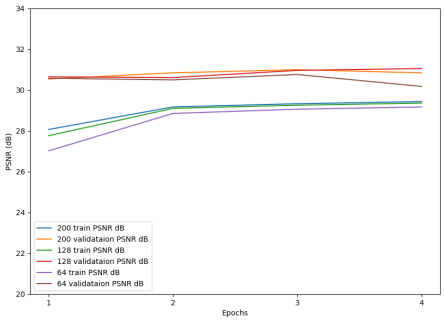

Figure 6. Performance by filter size rate.

As we can observe from the plot, neither increasing nor decreasing the filter bettered the performance of the validation set. Thus, the model with a number of filters equal to 128 and 64, kernel sizes of 9,1,5, and a learning rate of 0.0001 is our better-performing model on our 2 times down-sampled dataset.

We then tested our model on Set5 and Set14, which yielded the following final performance: 30.22 dB and 30.14 dB, respectively.

We focus now on the 4 times down-sampled dataset and we compared the performances of a model with the aforementioned structure between training it from scratch and using the training on the twice down-sampled dataset as pre-training.

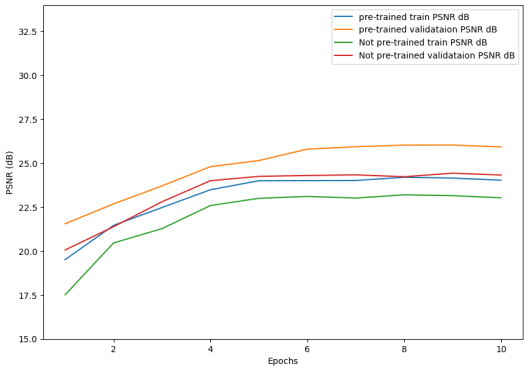

Figure 7. Performance on the 4x dataset.

As we can see the effects of pre-training are noticeable and improve the overall performance of the model. Finally we tested it on Set5 and Set14 down-sampled with a factor of 4 and we got a PSNR value of respectively: 24.53 dB and 24.71 dB

5.0.2 EDSR

In this work, we used the Enhanced Deep Super-Resolution (EDSR) model to reconstruct high-resolution images from their low-resolution counterparts. For this experiment, we utilized a training dataset (train-ds) and a validation dataset (val-ds). Each dataset contained low-resolution (LR) and high-resolution (HR) images. The model architecture can be explained with the following components:

- Residual Blocks: The residual blocks employed in the Central to the EDSR architecture. Each block consisted of two convolutional layers with 32 filters and a kernel size 3x3 surrounding a rectified linear unit (ReLU) activation. The output of the second convolutional layer is then added to the input of the block.
- Upsampling Layers: In this model, we used two upsampling stages, each doubling the resolution, resulting in a total upscaling factor of 4.
- Initial and Final Convolutional Layers: The model began with an initial convolution layer to project the input LR image into a high-dimensional feature space. After passing through the residual blocks and upsampling layers, a final convolutional layer was used to generate the output image, ensuring it had the same number of channels as the original HR image.
- Normalization: The input and output images were normalized using a rescaling layer to ensure that the pixel values remained within the appropriate range during processing, with the inputs scaled to [0, 1] and the outputs scaled back to [0, 255].

Training and Optimization: In the training session, we compiled the model using the Adam optimizer, which was chosen for its efficiency and ability to handle sparse gradients. The learning rate was managed using a piecewise constant decay schedule, starting with a learning rate of 1×10^{-4} and decreasing to 5×10^{-5} after 5000 steps. This strategy aimed to stabilize training by reducing the learning rate as the model converged. The loss function used was the Mean Absolute Error (MAE), which is effective for pixel-wise comparisons between the predicted and target images. PSNR was chosen as the primary metric to evaluate the model's performance. A higher PSNR indicates a closer resemblance between the super-resolved image and the ground truth HR image. We only trained the model over 50 epochs 8, each consisting of 100 steps because it was

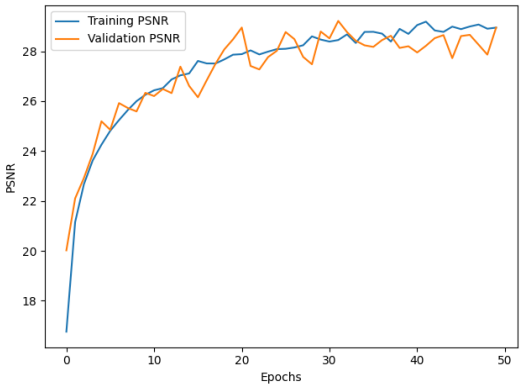

Figure 8. PSNR metric on EDSR model for train and validation dataset after 50 epochs.

computationally expensive and heuristically it looked like we were close to convergence.

Testing the model on the benchmark datasets Set5 and Set 14 yielded a PSNR performance of respectively: 28.17 dB and 27.86 dB.

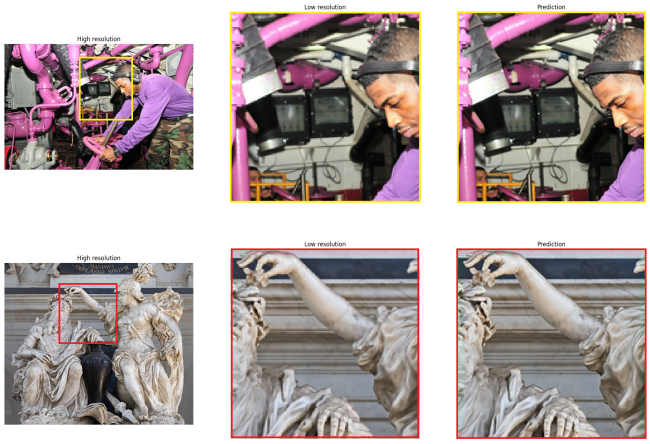

Figure 9. Examples of the EDSR model applied to a 4 times down-sampled image.

6. Conclusion

We implemented two models capable of augmenting the resolution and quality of an image that was down-sampled with a factor of 4. Based on a purely convolutional neural network, the former achieved a performance of respectively 24.53 dB and 24.71 dB on the Set5 and Set14 benchmark datasets. Based on residual network architecture, the latter achieved a performance of 28.17 dB and 27,86 dB, respectively, on the same aforementioned datasets. Thus, our best-performing model is the EDSR model. Further work should consider trying to implement a Multi-scale model that is capable of handling images that were down-sampled with a variable factor as highlighted in Bee Lim et all [9]

GitHub link of our project

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[2] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi Morel. Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding. In *British Machine Vision Conference (BMVC)*, Guildford, Surrey, United Kingdom, Sept. 2012.

[3] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 184–199, Cham, 2014. Springer International Publishing.

[4] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks, 2015.

[5] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 391–407, Cham, 2016. Springer International Publishing.

[6] Seiichi Gohshi. Real-time super resolution algorithm for security cameras. In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, volume 05, pages 92–97, 2015.

[7] Jithin Saji Isaac and Ramesh Kulkarni. Super resolution techniques for medical image processing. In *2015 International Conference on Technologies for Sustainable Development (ICTSD)*, pages 1–6, 2015.

[8] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution, 2016.

[9] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution, 2017.

[10] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections, 2016.

[11] Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. Single-image super-resolution: A benchmark. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 372–386, Cham, 2014. Springer International Publishing.

[12] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In Jean-Daniel Boissonnat, Patrick Chenin, Albert Cohen, Christian Gout, Tom Lyche, Marie-Laurence Mazure, and Larry Schumaker, editors, *Curves and Surfaces*, pages 711–730, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.