CNNs for Single Image Super Resolution

Michele Bortone

michele.bortone@studenti.unipd.it

Alessio Lazzaron

alessio.lazzaron@studenti.unipd.it

Abstract

Super-resolution is the process to create high-resolution images from low-resolution images. Single image super-resolution (SISR), refers to the goal of recovering one high-resolution image from one low-resolution image.
In this study we report a comparison between some convolutional neural networks architectures that can solve this task, analyzing its performance. Our goal was to determine if the increase in complexity between the various architectures, would lead to an increase in performance.
We have analyzed three different architectures and evaluated performance with common image compression quality metrics.

1. Introduction

In most computer vision tasks, high resolution images are usually desired for image processing and analysis. Single Image Super Resolution is widely used in some medical imaging applications (e.g. magnetic resonance imaging), and security and surveillance imaging as well. Moreover, object detection problems can be accomplished using Super-resolution which helps when low resolution is caused by the long distance between the target and the imaging sensor and objects are too small. Super-resolution (SR) refers to the task of restoring high-resolution images (HR) from one or more low-resolution (LR) observations of the same scene. According to the number of input LR images, the SR can be classified into single image super-resolution (SISR) and multi-image super-resolution (MISR). Compared with MISR, SISR is much more popular because of its high efficiency. Many SISR methods have been studied in the digital age, including interpolation tecniques such as Nearest neighbor or bicubic.
Recent popular methods are based on neural networks and most architectures studied for this task are Convolutional Neural networks and Deep Generative models.
The deep learning models capability is particularly suitable for this type of task because simpler approaches like bicubic interpolation use only local information in an LR image to compute pixel values in the corresponding SR image, deep learning approaches, on the other hand, learn mapping functions from LR images to HR images from a large number of examples.

2. Related work

Deep learning can be used to estimate the high-resolution image given a low-resolution image. By using the HR image as a target (or ground-truth) and the LR image as an input, we can treat this like a supervised learning problem. There are a lot of approaches to affront this problem however most of them use CNNs because they get better performances. Some approaches are listed below and represent a broad outline of the current state of the art.

2.1. Pre-upsampling

In this method, the low-resolution images are first interpolated to obtain a "coarse" high resolution image. Now, CNNs are used to learn an end-to-end mapping from the interpolated low resolution images to the high resolution images. The intuition was that it may be easier to first upsample the low-resolution images using traditional methods (such as Bicubic interpolation) and then refine the resultant than learn a direct mapping from a low-dimensional space to a high-dimensional space. The advantage is that since the upsampling is done by traditional methods, the CNN only needs to learn how to refine the coarse image, which is simpler.
In our comparison, we implemented and analyzed two pre-upsampling networks, that are SRCNN [**?**] (Super Resolution Convolutional Neural Network) and VDSR [**?**] (Very Deep Super Resolution).

2.2. Post-upsampling

This method works in reverse way respect to the previous one, indeed the low-resolution images are passed to the CNNs as such. Upsampling is performed in the last layer using a learnable layer. The advantage of this method is that feature extraction is performed in the lower dimensional space (before upsampling) and so the computational complexity is reduced. Also, by using an learnable upsampling layer, the model can be trained end-to-end.

This network architecrture is suitable when we are interested to produce the output quickly because we can skip the pre-upsampling process. In this this report we have compared FSRCNN [**?**] (Fast Super Resolution Convolutional Neural Network) with the two Pre-upsampling CNNs cited above.

2.3. Progressive upsampling

The above approaches use only a single upsampling convolution to not increase the computational complexity. This makes the learning process harder for large scaling factors. To avoid it these models use a cascade of CNNs to progressively reconstruct high resolution images. By decomposing a difficult task into simpler tasks, the learning difficulty is greatly reduced and better performance can be obtained.

2.4. Iterative Up and Down sampling

Another popular model architecture is the hourglass structure that alternating between the process of upsampling and downsampling. These models can better find the deep relations between the LR-HR image pairs and thus provide higher quality reconstruction results.

2.5. Generative Adversarial Networks (GANs)

This type of network have been increasingly used for several image based applications including Super Resolution. GANs typically is composed of two neural networks: the Generator and the Discriminator, that dueling each other in a zero-sum game. Given a set of target samples, the Generator tries to produce samples that can fool the Discriminator into believing they are real. The Discriminator tries to recognize the real (target) samples from fake (generated) samples. At the end we obtain a Generator that is really good at generating samples similar to the target samples.

3. Dataset

For our purpose we use three different dataset, the first is *T91* that is the smallest dataset, it contains 91 mixed images of small size that representing fruits, vegetables, objects and people, every images is in Bitmap format. The second is *General100* that contains 100 bmp-format images with no compression. The size of these 100 images ranges from 710 x 704, the largest, to 131 x 112, the smallest. And the third dataset is *Berkeley Segmentation Dataset (BSDS200)*, ot contain 200 images and it is created for developing new boundary detection algorithms and for developing a benchmark for that task. For each neural network that we implement, we also adjust the number of epochs for the training phase, in fact, since the size of the datasets were different it was necessary to customize this hyperparameters to have, more or less, the same performances on the validation test and avoid overfitting.

The validation set is obtained by split the dataset in two parts: the biggest one, almost the 80% of the dataset, for the training set, and the rest 20% for the validation set. However using different type of training set didn't change the models performance so much.

To testing the performance of the models and to understand how much the models are able to reconstruct from the low-resolution images to high resolution images we use, as testing dataset, *Set5* and *Set14* that are common evaluation dataset for Super Resolution task and contains various images of buildings to animal faces.

Official web pages of SRCNN, FSRCNN and VDSR provide all datasets used. We don't use directly the dataset but we apply some preprocessing. First off all we convert the training set's images from RGB into YCbCr format because the models is training with only the luma, the brightness, of the images. After this, we normalize the images in order to have floating point values between 0 and 1, so in this way we make the convergence of the loss function faster. The low-resolution images is made by down and up sampling the images, inside the dataset, with a bicubic interpolation and we analyze the model's performance with different scale factors.

In order to perform training operations with larger dataset we do some data augmentation. We crop small images called patches from each images in the training set. Patches are overlapping sub-images densely cropped from the original image with low size. The stride, which controls how many images are cropped, and size are hyperparameters.

Patch extraction is the process of extracting a large set of small image patches, from a single larger image. This type of data augmentation is frequently used in image-to-image regression problems such as Super Resolution, where networks can be trained on very small input image sizes. Patch extraction is widely used in recent Super Resolution methods.

Also in deep models we use rotated images to create a larger dataset for the training phase.

4. Method

We want to verify if the complexity of the model increases also its performance do too. So we implement three convolutional neural network that are different for the complexity of the architecture.

4.1. SRCNN

The Super Resolution CNN [**?**] is relatively simple and can see as an ordinary CNN that can approximates the complex mapping between the LR and HR spaces in an end-to-end way As shown in figure 1, SRCNN is composed by three main parts:
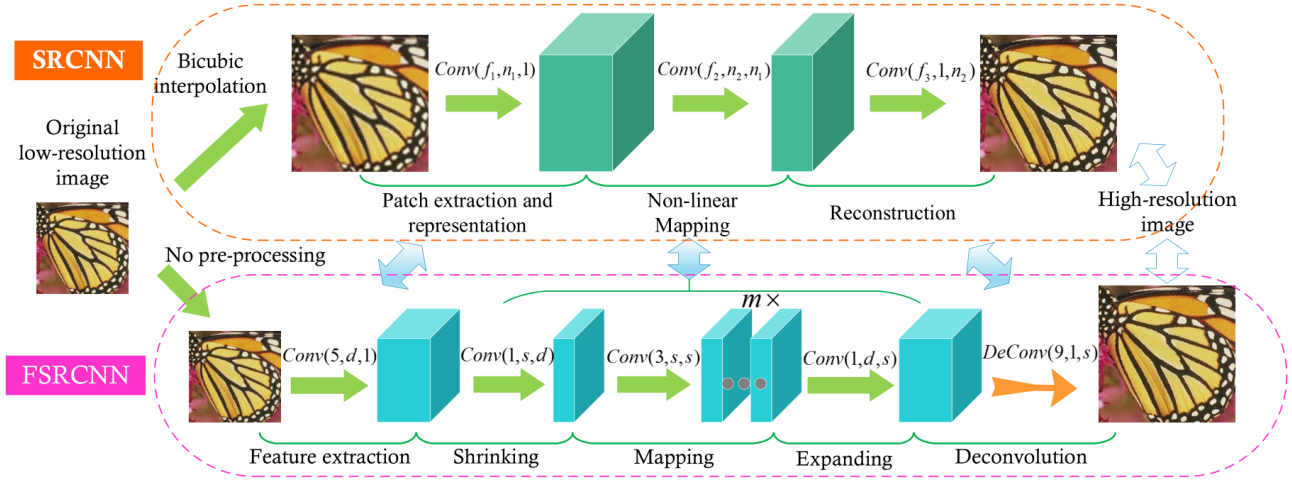
Figure 1. Comparison between SRCNN and FSRCNN architectures [**?**]

- **Patch extraction and representation:** extracts patches from the low-resolution image and represents each patch as a high-dimensional vector;

- **Non-linear mapping:** maps each high-dimensional vector into another high-dimensional vector;

- **Reconstruction:** aggregates the above high-resolution patch representations to generate the final high-resolution image;

So SRCNN has only three layer where the first layer has 64 filters and each of them as a dimension of 9x9, the second layer has 32 filters with dimension 1x1 and the output layer has only one filter with dimension 5x5. Every layer not have padding and for the first two layers we use a relu activation function and for the last layer we adopt a linear activation function. The loss that the model want to minimize is the mean square error, between the low resolution images and the output high resolution images, using stochastic gradient descent with the standard backpropagation. More precisely we adopt Adam as optimization algorithm

4.2. FSRCNN

FSRCNN is presented by the authors like an evolution of SRCNN. The main goal of this architecture is to reduce the the time complexity of the output image computation.
To solve the computational cost problem, the bicubic interpolation step computed in SRCNN is replaced with a deconvolutional layer. In FSRCNN there is no pre-processing and the the input is the low resolution image.
As shown in figure 1, FSRCNN is composed by five main parts:

- **Feature Extraction:** performs feature extraction on the low resolution image;

- **Shrinking:** reduces the number of feature maps so parameters as well;

- **Non-Linear Mapping:** multiple layers that map feature maps to high resolution representation;

- **Expanding:** Inverse process of the shrinking layer;

- **Deconvolution;** aggregates features and create upsampled image.

We can see the deconvolution operation like the inverse of the convolution. The output of this layer will be the input size multiplied by the stride. The stride of the deconvolutional operation is set to the desired upscaling factor.
Another way to see the deconvolution operation is to think it like an upsampling method such as interpolation but with parameters to learn with optimization algorithms.
The number of filters on feature extraction d, the number of filters after shrinking operation s and the number of non-linear mapping layers m are hyperparameters.
The non-linear mapping step in SRCNN is replaced by shrinking, mapping, and expanding step. The non-linear mapping operation is composed by multiple smaller layers than the single wide layer used in SRCNN, so this operation is more efficient due to the deep networks advantages.
Finally shrinking and expanding operations are able to reduce drastically the number of parameters to learn during training. Smaller number of parameters and the low resolution input produce a faster output than SRCNN.

4.3. VDSR

Very Deep CNN exploits a large number of layers to restore image and improve quality. Its architecture includes a cascade of hidden layers which include small filters in equal
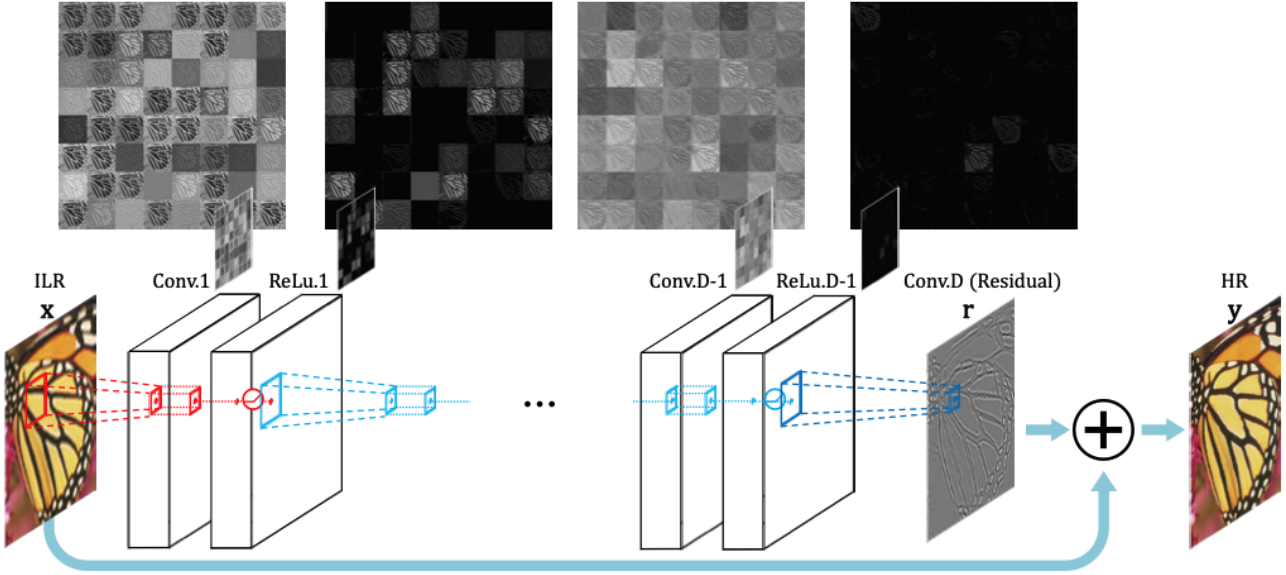
Figure 2. VDSR architecture[?]

number.

Each hidden layer has 64 filters and each of them as a dimension of 3x3. The number of hidden layers is an hyperparameter. The main idea behind this network is the implementation of residual learning. In the previous models, the exact copy of the input has to go through all layers until it reaches the output layer. This becomes an end-to-end relation between input and output and in a deep model can create serious problems such as difficult to train the network and vanishing/exploding gradients. The input of VDSR mis the interpolated image x and the target is the high resolution image y. Since x and y are pretty similar we can learn only the residual image $\mathbf{r} = \mathbf{y} - \mathbf{x}$.

After the generation of the output is possible to reconstruct the HR image summing the interpolated image with the residual.

In the original paper they have demonstrated that is possible to train this network to learn Super Resolution in different scale factors if we train the network with a dataset that contains images down-sampled in multiple scale factors.

5. Experiments

We have implemented these three networks using python libraries:

- OpenCV: Images processing;

- Tensorflow: CNN building and training;

- Skimage: Computing metrics;

- Google Colab: Development environment, although the free version is limited, the computing power is superior to normal laptops

We have measured performance using classical image compression metrics:

- Mean Squared Error (MSE) : The mean of squared difference between ground truth image pixels and restored image pixels. The lower the MSE, the better the quality of the reconstructed image.

- Peak Signal to Noise Ratio (PSNR) : Ratio between the maximum possible value and the value of distorsion in the image. Resolve the MSE problem, that is the dependence with the pixel scale. The higher the PSNR, the better the quality of the reconstructed image.

- Structural similarity (SSIM) : method that measures the perceptual difference between two similar images. Unlike PSNR, SSIM is based on visible structures in the image. The higher the SSIM, the better the quality of the reconstructed image.

Our aim is to verify if the increase of complexity, in the architectures used, also brings a considerable increase on the performance in the SISR task. To compare the results between the different models and the papers we trained all the network with T91. We trained SRCNN with 16024 input's patch and 5688 patch for the validation, also we used a learning rate as 0.001

6. Conclusion

References

[1] Authors. The frobnicatable foo filter, 2014. Face and Gesture submission ID 324. Supplied as additional material `fg324.pdf`.