Video Super-Resolution

Abhinav Poddar	Naitik Malav	Rajesh Chavre	Samar Garg	Brijesh Aghav
EP19BTECH11023	CS19BTECH11026	CS19BTECH11030	CS19BTECH11033	CS19BTECH11047

Abstract

We proposed a deep learning based method for converting low resolution video to high resolution video. There are many feasible methods to perform the same objective. We have used already existing SRCNN and FSRCNN models to convert low resolution frame to high resolution frame. Our main goal is to use deep learning to improve the speed and restoration quality of the video. We will try to address a few drawbacks of traditional approaches. We will explore different models and analyze their output to achieve our desired result. We have discussed the formulation part of both models below. We have also compared the output results with the low resolution images and various plots.

1. Introduction

The usage of computer vision for different applications is increasing day by day. In this project, we aim to recover a high-resolution video from a given low-resolution video. The main motivation behind this project is that we would be able to restore old videos which are in lower resolution to higher resolution videos. This will help make it a more comfortable experience for the viewers to watch the video. Another application where it would be useful are various video streaming platforms like YouTube, Netflix which would greatly benefit from the reduced resolution of the video due less storage space of those videos on their servers. The model can convert those low-resolution video to high-resolution video on the client side of network in real time. Thus we can provide high-quality videos on low bandwidth networks. The basic pre-processing in starting is to sample the video into frames or images and then testing these images on CNN model which produces high resolution images. Later these high resolution images added according to the start order which results in high resolution video. CNN directly learns an end-to-end mapping between low and high-resolution images, which makes the method much faster and efficient.

2. Literature Review

Model-1: Image Super-Resolution Using Deep Convolutional Networks [1]

Our first model is based on the image super resolution technique based on the research paper Image Super-Resolution Using Deep Convolutional Networks. So far we have done the basic pre-processing of the video. That is we have sampled it into the list of images. For eg- if we have a 30fps and 60secs long video we have divided the video into 1800 images, so the model is tested only on these images.

SRCNN, a fully convolutional neural network for image super resolution directly learns an end-to end mapping between low and high-resolution images, with little pre/post-processing beyond the optimization. SRCNN is improved by increasing the filter size in the nonlinear mapping layer, and then investigating deeper structures by adding nonlinear mapping layers. In comparison to other image super resolution models, SRCNN is faster at speed, so it is not only a quantitatively superior method, but also a practically useful one.

Formulation: A single low-resolution frame(or image) from video is up-scaled to the desired size using bi cubic interpolation. Let us denote the interpolated image as Y. Our goal is to recover an image F(Y) from image Y that is as similar as possible to the ground truth high-resolution image X. We wish to learn a mapping F, which conceptually consists of three operations.

- **Patch Extraction and representation** - This technique extracts patches from the low-resolution frame Y and stores each patch as a high-dimensional vector. These vectors are made up of a collection of feature maps, the number of which is equal to the vectors dimensions. Formally, our first layer is expressed as an operation F_1:

  $$F_1(Y) = max(0, W_1 * Y + B_1),$$

  where '*' is a convolution operation and W_1 corresponds to n_1 filters of size c$\times$f$_1$ $\times$ f_1, where c being

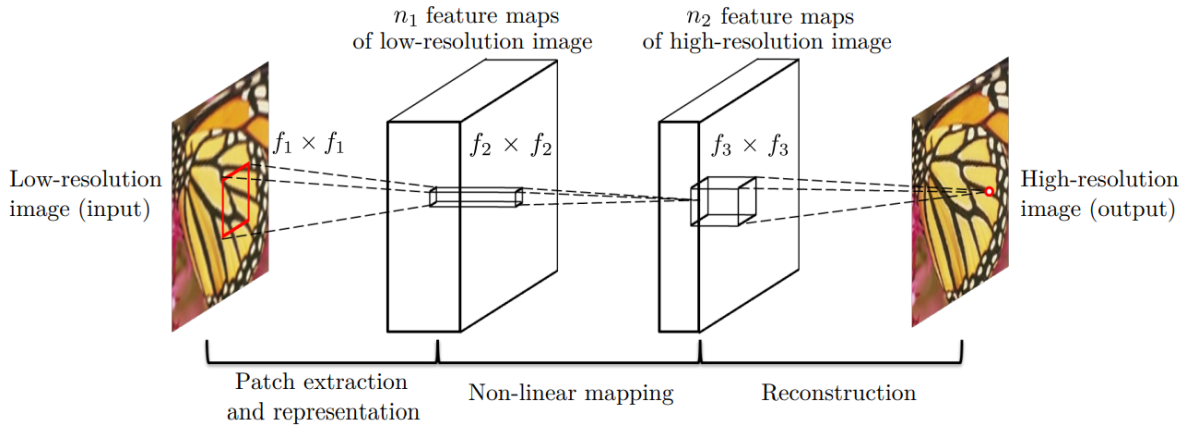

Figure 1. SRCNN Model

the no. of channels in the input image and f_1 being the spatial size of a kernel, and B_1 represents the biases. The output has n_1 feature maps and we have also applied ReLU(Rectified Linear Unit) on the filter responses.

- **Non-Linear Mapping** - Each high-dimensional vector is non linearly mapped onto another high-dimensional vector. Each mapped vector represents a high-resolution patch conceptually. These vectors comprise another set of feature maps. That is the first layer extracts an n_1-dimensional feature for each patch. In the second operation, we map each of these n_1-dimensional vectors into an n_2-dimensional one. The operation of the second layer is:

$$F_2(Y) = max(0, W_2 * F_1(Y) + B_2),$$

where W_2 contains n_2 filters of size $n_1 \times f_2 \times f_2$, and B_2 is n_2-dimensional.

- **Reconstruction** - It aggregates the above high resolution patch wise representations to generate the final high resolution frame. Averaging can be thought of as a pre-defined filter applied to a collection of feature maps, each point being the "flattened" vector version of a high-resolution patch. We have defined a convolutional layer to produce the final high-resolution images as:

$$F(Y) = W_3 * F_2(Y) + B_3,$$

where W_3 is a set of linear filters which corresponds to c filters of size $n_2 \times f_3 \times f_3$, and B_3 is a c-dimensional vector.

We have put all three operations together and form a convolutional neural network as shown in the figure. In this model, all the filtering weights and biases are to be optimized. The generated output of high resolution images added together to form the high-resolution video.

In short, the SRCNN's first convolutional layer produces a series of feature maps. The second layer converts these feature maps to high-resolution patch representations in a nonlinear fashion. The last layer combines the predictions within a spatial neighbourhood to produce the final high-resolution image F(Y).

Training: The end-to-end mapping function F requires the parameters Θ = W1, W2, W3, B1, B2, B3. It can be achieved by minimizing the loss between the reconstructed images F(Y, Θ) and the corresponding ground truth high resolution images X. We have used Mean Squared Error(MSE) as the loss function for the given set of high-resolution images Xi and their corresponding low-resolution images Yi as defined below:

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \|F(Y_i, \Theta) - X_i\|^2$$

where n is the no. of training samples. Mean squared error(MSE) is used because the loss function favours a high PSNR. The PSNR is a widely-used metric for quantitatively evaluating image restoration quality, and is at least partially related to the perceptual quality. The loss is minimized using stochastic gradient descent with a standard backpropogation. The weight matrices are updated as -

$$\Delta_{i+1} = 0.9 * \Delta_i - \eta$$
$$W_{i+1}{}^l = W_i{}^l + \Delta_{i+1}$$

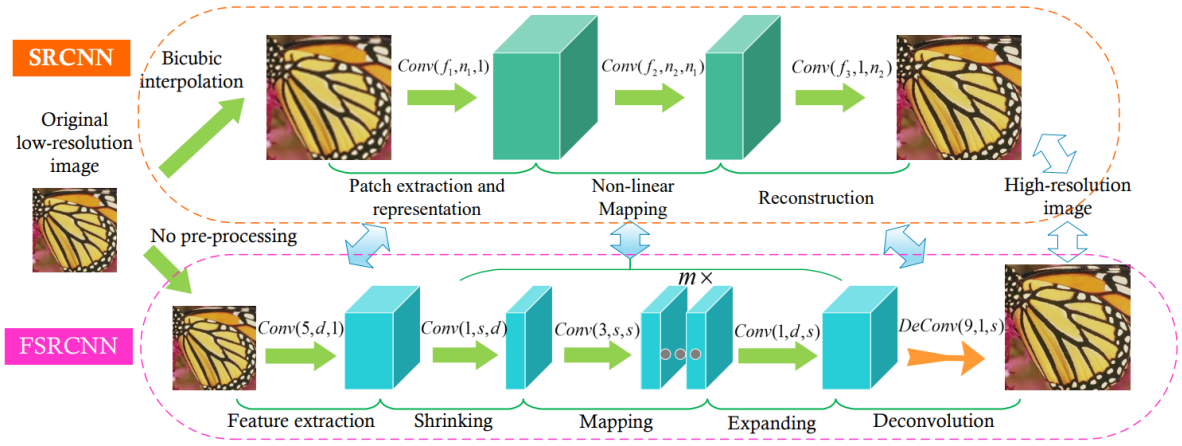

Figure 2. Comparison between architecture of FSRCNN and SRCNN Model

Model-2: Accelerating the Super-Resolution Convolutional Neural Network [2]

This paper aims to accelerate the current SRCNN, and propose a compact hourglass-shape CNN structure for faster and better super resolution. So here re-designing the SRCNN structure mainly in three aspects-,

- FSRCNN adopts the original low-resolution frame as input without bicubic interpolation. A deconvolution layer is introduced at the end of the network to perform upsampling.
- Reformulating the mapping layer by shrinking the input feature dimension before mapping and expanding back afterwards.
- FSRCNN adopts smaller filter sizes and a deeper network structure.

These improvements provide FSRCNN with better performance and lower computational cost than SRCNN.

Formulation: According to the Fig2, FSRCNN can be decomposed into five parts – feature extraction, shrinking, mapping, expanding and deconvolution. Out of these 5 parts, the first four parts are convolution layers, while the last one is a deconvolution layer.

- **Feature Extraction** - It is similar to SRCNN. FSRCNN performs feature extraction on the original LR image without interpolation.
- **Shrinking** - Shrinking layer is added after the feature extraction layer to reduce the low resolution feature dimension, which results in the low computation complexity of the mapping step. By adopting a smaller filter number which is very less here in comparison to SRCNN, the low resolution feature dimension is also reduced. In short, this strategy greatly reduces the number of parameters.
- **Non Linear Mapping** -The non-linear mapping step is the most essential component impacting SR performance, and the most influential parameters are the mapping layer's width (i.e., the number of filters in a layer) and depth.
- **Expanding** - Shrinking operation reduces the number of LR feature dimension for the sake of the computational efficiency. However, restoration quality will be poor if we generate the high resolution frame straight from these low-dimensional characteristics. As a result, after the mapping section, we add an extending layer to expand the HR feature dimension.
- **Deconvolution** - The last part is a deconvolution layer, which aggregates the previous features with a set of deconvolution filters resulting into high resolution frame.
- **PReLU** - We have used the Parametric Rectified Linear Unit instead of ReLU as an activation function. PReLU helps in avoiding the dead features caused by zero gradients in ReLU. We can make full use of all parameters to test the maximum capacity of different network designs. PReLU activation fucntion is applied after each convolutional layer.
- **Cost Function** - Similar to SRCNN, the mean squared error(MSE) is used as the cost function. The optimization objective is represented as -

$$min_\theta \frac{1}{n}\sum_{i=1}^{n}\|F(Y_s^i,\theta) - X^i\|_2^2$$

where Y_s^i and X^i are the i-th low resolution and high resolution images in training data. $F(Y_s^i, \theta)$ is the high resolution output of Y_s^i with parameters θ. All parameters are optimized using stochastic gradient descent with the standard backpropagation.

3. Experimental Results

Dataset preprocessing - We have used DIV2K dataset for training purpose. It can be downloaded via link. The first model is trained on 1000 images and it is tested on 100images. The first model performs super resolution with a scale factor of 2. The second model is trained using 7300 high resolution images and it is tested on 100images. The second model performs super resolution with a scale factor of 3. There are several interpolation methods that can be used to resize images; however, we will be using bicubic interpolation.

Testing Low Resolution Images - For image testing purpose we have down scaled few 100 images by respective scale factor. To ensure that our image quality metrics are being calculated correctly and that the images were effectively degraded, we have calculated the PSNR, MSE, between our reference images and the degraded images that we just prepared. Below are the few graphs. Fig 3 has MSE plots on FSRCNN model using scale factor-2 and Adam, and RMSProp optimizer. Fig 4 has MSE plots on FSRCNN model using scale factor-3 and Adam optimizer.

Testing low resolution video - We have submitted python scripts for breakdown of video into images and clubbing images to get back video. These generated images then used for testing purpose. We have also submitted the generated high resolution video and the original low resolution video.

PSNR-

The PSNR block calculates the peak signal-to-noise ratio between two pictures in decibels. This ratio is used to compare the quality of the original and compressed images. The better the quality of the compressed or reconstructed image, the higher the PSNR. MSE is not a good metric for the image similarity as multiple images can result to same MSE value. So we have also used PSNR. One of helpful qualities of PSNR is that it can be expressed easily in terms of MSE. The PSNR is described by the equation-

$$PSNR = 20 * \log_1 0(\frac{max^2}{\sqrt{mse}})$$

where max is the highest possible value of the signal, and mse is mean squared error.

Fig 5 has PSNR plots on FSRCNN using scale factor-2 and, Adam and RMSProp as optimizer. Fig 6 has PSNR

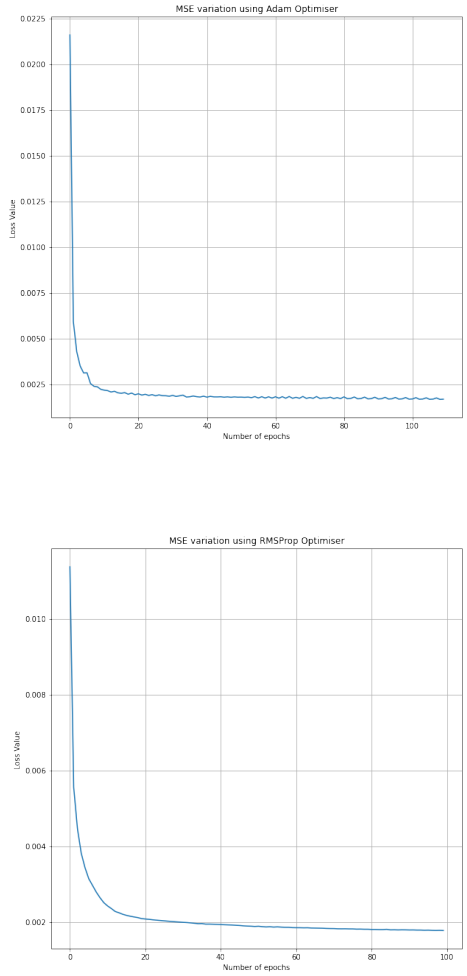

Figure 3. MSE Plots on FSRCNN model(scaling-2), (a)Adam optimizer, (b)RMS prop

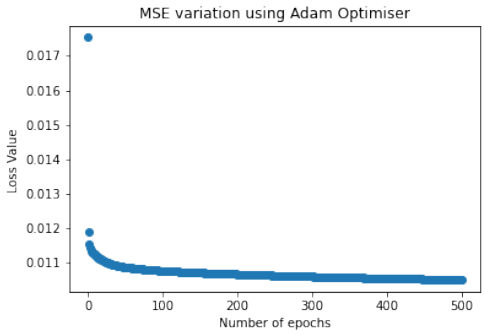

Figure 4. MSE Plots on FSRCNN model(scaling-3), Adam optimizer

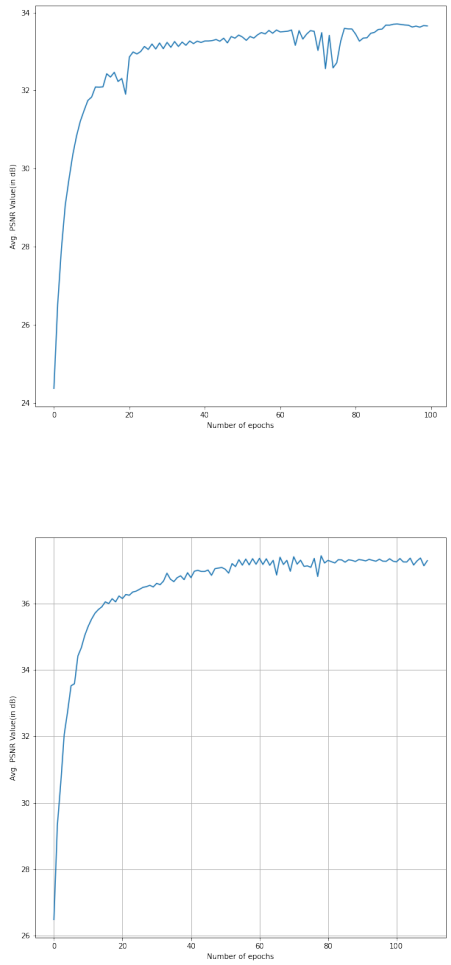

Figure 5. PSNR Plots on FSRCNN model(scaling-2), (a)Adam optimizer, (b)RMS prop

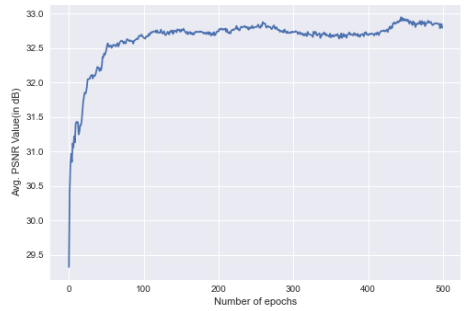

Figure 6. PSNR Plots on FSRCNN model(scaling-3), Adam optimizer

plots on FSRCNN model using scale factor-3 and Adam optimizer.

Results -

Low resolution frames and high resolution frames from both of the videos are as follows. We have shown one frame and it's output with different models in figure 7. In figure 8 we have shown the original low resolution image, interpolated image and high resolution image generated by our model FSRCNN.

Figure 7. First image is of size 426x240p, second is 854x480p, and third is 1280x720p

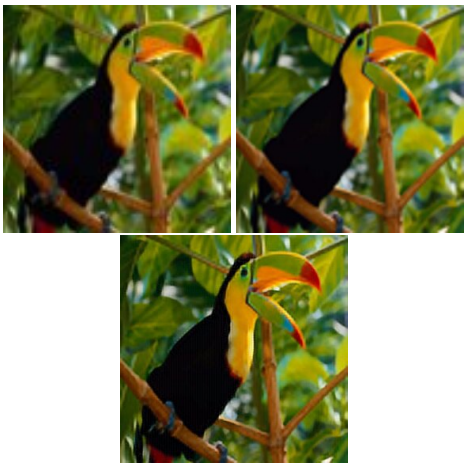

Figure 8. Low resolution, Interpolated, and High resolution.

4. Improvements

SSIM - The structural similarity index measure (SSIM) is a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos. SSIM is used for measuring the similarity between two images. Other techniques, such as MSE or PSNR, estimate absolute mistakes, whereas these methods measure relative errors. Structural information refers to the assumption that pixels have a lot of interdependencies, especially when they're close together in space. These dependencies contain crucial information about the structure of the visual scene's elements. For eg. Figure 8

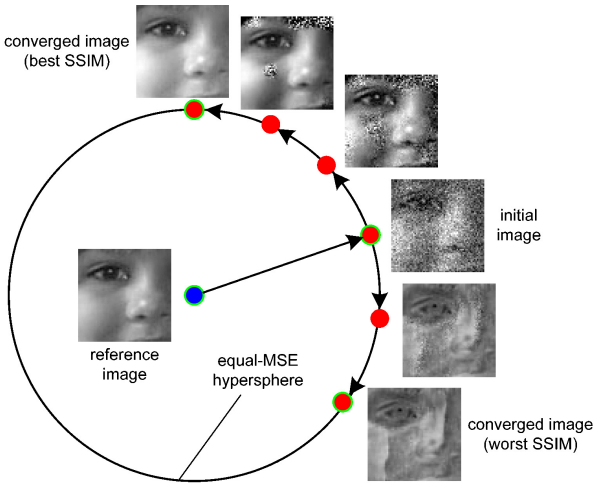

Figure 9. SSIM example

References

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE*, 2015. 1

[2] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. *IEEE*, 2016. 3