Report: Super Resolution Image Reconstruction using a Deep Learning Architecture

Christopher Murray and Guillaume Dufau

December 2018

1 Introduction

Super Resolution Image Reconstruction (SR) has been an active field of study in Computer Vision for the last three decades. Recently, the advent of Deep Learning has directed research efforts in this area towards the use of new and creative deep learning architectures. This effort has shown to be successful as modern techniques using Deep Convolution Neural networks and Generative Adversarial Networks are consistently outperforming more traditional methods. The idea of this project was to build on the Very Deep Convolutional Neural Network (VDCNN) [1] to see how well such an architecture could be adapted to do SR from multiple images, as well as to see how kernel size of convolutional layers affect the ability of the VDCNN.

For this project, we decided first to implement the VDCNN proposed by Kim et al. [**?**], and then attempt to implement some adjustments to see if it could be improved

2 Implementation of the paper

2.1 Algorithm

The VDCNN is trained such that given a low-resolution image, the network produces the high-frequency information that the low-resolution image is missing. Every layer of the network is of the same width and height, so every layer is padded with a single pixel of zeros so that convolution by a 3×3 kernel produces a layer of the same dimensions. The structure described above is illustrated in Figure 1.
The input is a low-resolution image x_i, which has been interpolated to match the dimensions of its corresponding high-resolution analogue y_i. For our implementation, the input tensor took the following shape:

$$[number\ of\ images, weights, heights, number\ of\ channels] = [n, imagewidth, imageheight, 3] \tag{1}$$

An outputted image, residual $f(x_i)$, keeps the same shapes than the input so we can easily reconstruct the generated high-res image $\hat{y_i} = x_i + f(x_i)$. Therefore, to train the Neural Network we use

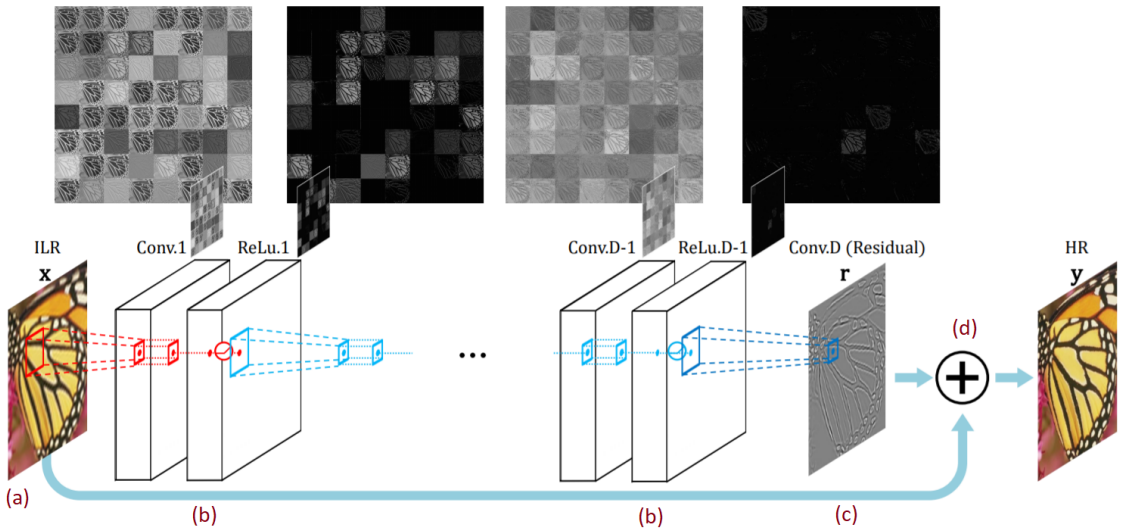


Figure 1: Very Deep CNN: (a) Input interpolated low-res image, 3 channel entry (b) Layers of convolution / deconvolution (with ReLu activation). To keep patches with same size we add a padding after each deconv sub-layer (c) Output of the Network being the residual (supposed difference between high res and low res image), with 3 channel (d) Sum the residual and the low-res input image giving the generated high-res image.

the following loss function:

$$loss = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 + \sum_{j=1}^{nbr\ weights} w_j^2 \tag{2}$$

with: n the number of inputted images (batch size).

The first summing term of the loss function drives the learning of the residual differences between low-resolution and high-resolution images. The second summing term prevents an exploding gradient from happening while learning. This is a minor adjustment from the original paper as gradient clipping was used in for their representation. Without the gradient explosion term, the network can often produce images tinted by one of the colour channels.

As it was done in the paper, the network creates stacks of 64 patches between convolutional layers keeping the inputted height and width of the image. Also, the learning rate had to be fine tuned so that the network would properly converge.

3 Innovations

3.1 Multiple image low resolution representation

Our approach to SR from multiple image sources was rather simple since the current efforts in Deep Learning SR focus exclusively on single-image SR. The idea was to create five different low-

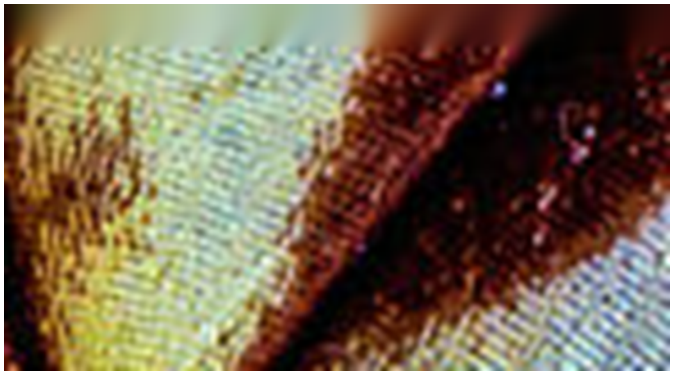


Figure 2: Effect on the image of an in-painting method, applied here on top of the image

resolution representations of the original ground-truth image using weighted kernels, and then see if the network would learn a better way to map the high-frequency information from the sum of the low-resolution representations. One of the kernel is shown in Equation 3.

$$Left - oriented\ kernel = \begin{bmatrix} 0.4 & 0.1 \\ 0.4 & 0.1 \end{bmatrix} \tag{3}$$

In order to gather information from those 5 images we changed the input of the CNN. Instead of a 3 channel image, the input for the neural network increased to 15 channels where the 3 first channels correspond to those of the usual low-res image, the next 3 to those of the image with left-oriented kernel, etc.

For the network to learn from multiple inputs, we adapted the loss function to use the squared difference $(y_i - \hat{y_i})^2$ where the product of the network is added to the low-resolution image taken from an unbiased kernel.

3.2 Resampling padding

Another idea to build upon the VDCNN architecture was to increase the size of the kernel. However, this can be problematic since the layers need to maintain the same dimensions, meaning that there would be increased padding with zeros. This causes a decrease in the receptive field as described here [3]. We thought it would be interesting to see if in-painting an expanded border would lower the influence of the zero padding when training kernel weights. An example of in-painting is shown Figure 2.

4 Results

The presented results were obtained after 2 hours training with our personal computers. The network learned from 400 images. Looking at the loss values in the following table, an increased time of training might lead to even better results. The result shown in Figure 3 are obtained on an image left aside during training.

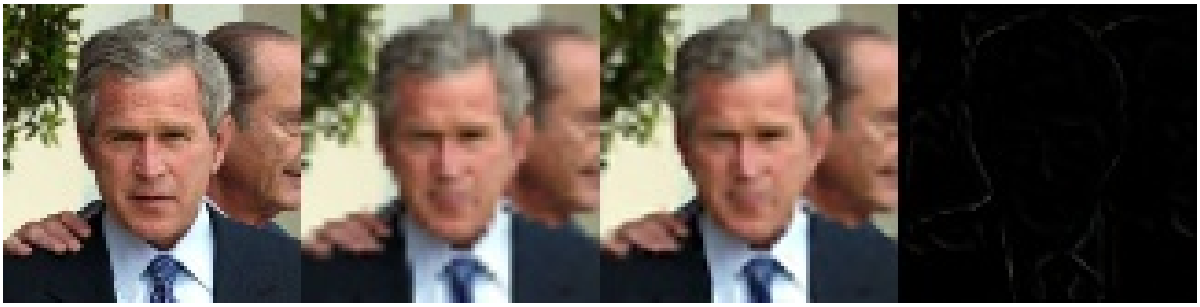


Figure 3: From left to right: Original image (downsized to 81x81) / the interpolated Low-res image / the generated High-res image (sum of low-res and residual) / The residual image, output from the network

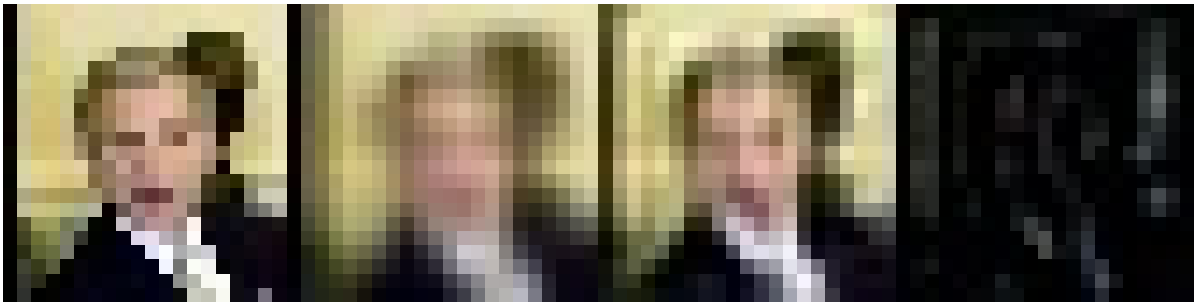
Loss values during training		
Averaged error	Number of epoch	Running time(min)
3200	1	4
1835	15	60
1450	28	110
1360	30	120

The generated image has sharper edges, in some points similar to the original case. However the modification inside the object (face here) is fairly small. We believe a longer training time might help getting better results on this point.

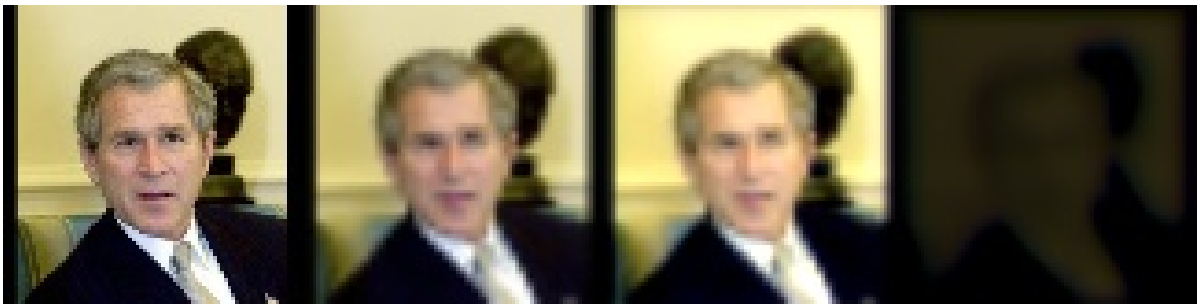
Following these convincing results are the results with multi-image input. Shown in Figure 4. A difficulty during this project was to find the best possible learning rate for the network to converge. It highly depends on the size of the inputs as well as the number of images. Knowing that, for the 81x81 sized images, the running time was around 1 or 2 hours. On the sample (b) of Figure 4, as mentioned earlier, there was a gradient explosion. Due to a lack of time we couldn't rerun it enough to find the final best learning rate and so results for multi-image case.

However we feel pretty confident on the idea that multi-image input is helpful in our case. Indeed, the sample (a), which converged well, shows promising results for a multi-image VDCNN architecture.

Unfortunately, we were unable to test the in-painting border, we couldn't find a way to correctly implement such a personalized padding in tensorflow. The idea was to define no padding for in between layers, and then manually extract layers and output a larger layer such that the convolutions of a larger filter would produce the proceeding layer to be the same size of the original image. The language requires that the graph is fully defined before it is executed, so such a padding policy requiring OpenCV functions could not be implemented as it is impossible to inpaint on a placeholder.



(a) 21x21 sized images



(b) 81x81 sized images

Figure 4: Output when using multiple images as input with different sizes, same order than above description

5 Conclusion

We found that both the single image and multiple image VDCNNs produced more visually appealing results compared to the standard bilinear interpelation. For future work, the idea of a multiple image Deep Learning architecture could be explored further. Another idea that would be interesting to follow up on would be to implement the border inpainting padding and see how the results of that padding scheme compare to the usual zero padding scheme.

References

[1] Jiwon Kim, Jung Kwon Lee and Kyoung Mu Lee *Accurate Image Super-Resolution Using Very Deep Convolutional Networks* Seoul National University · 2016.

[2] Yuan Yuan et al. *Learning a Deep Convolutional Network for Image Super-Resolution.* University of Hong Kong · 2014.

[3] Wenjie Luo et al. *Understanding the Effective Receptive Field in Deep Convolutional Neural Networks.* University of Toronto · 2017.