Super Resolution using Reference Image

Shreesh Ladha
UMass Amherst
sladha@cs.umass.edu

Sree Ram Sreekar Reddy
UMass Amherst
schegireddy@cs.umass.edu

Abstract

Traditional Image super-resolution is the task of inferring a high-resolution image from a single low-resolution input. We aimed to build upon it and improve the generated output by providing an additional reference image as input during training. Our experiments have been performed on a feed forward fully convolutional deep network. We explore different network structures and other parameter settings to improve performance and get better reconstruction quality. While we were not able to outperform our baseline model, the results obtained were on par with it in most cases.

1. Introduction

Super resolution (SR) is defined as the task of recovering a high resolution (HR) image from its low resolution (LR) counterpart. This problem has been there for many years but has been receiving tremendous attention in recent years owing to the deep learning revolution. The task is not trivial though, on account of its ill-posed nature - Many different HR images map to the same LR image. Especially for large downsampling ratios, high-frequency information is lost which renders the textured information in super-resolved images overly smoothed and blurry.

In our work we tried to overcome this issue by providing an additional HR image along with the input during training. This image (which we'll henceforth refer to as reference image) is similar to the LR image. The motivation behind the idea is that perhaps the network might draw some relevant information from the reference image to improve the generated HR image. We super resolve the image by a factor of $\times 4$

This is similar to example based super resolution which also uses external images. The difference is that we only use a single reference image as opposed to a large dataset of external images used for the former. Also example based SR involves multiple stages of learning as opposed to a single end to end learning strategy that is employed for neural networks. We experimented with several ideas and architectures to incorporate this additional information. To our knowledge, this is the first experiment of its kind using CNN's.

(a) (b)

Figure 1. Example Super Resolution : Left image is a low resolution input image while the right image is its super resolved counterpart obtained using a Deep CNN (ours)

2. Related Work

Image super-resolution is a classical problem with several approaches proposed over a decade. Early interpolation methods such [6] based on sampling theory, achieved blurry results. A large number of better performing algorithms have been surveyed by Yang et al [4].

Other popular classical approaches include learning mappings between low and high resolution image patches [14], [8]. Further, dictionary based methods that learn a sparse representation of the image patches [18], [11], [15] as well as neural network based approaches that use CNNs [10], [3], [2] have become popular.

The images that these models produce tend to be blurry because Mean Square loss minimization is used. Alternative losses such as perceptual loss, that calculates the L2 dif-

(a) (b) (c)

Figure 2. Example Triplet. From left to right : LR image, HR image and the reference image

Figure 3. Different scrambled I_{ref} obtained using L1, L2 and hypercolumn respectively for I_{LR} at the leftmost

ference in the feature space obtained after passing it through a pre-trained CNN like VGG have been suggested [5], [12]. The current state of the art method [13] uses a Residual network based architecture coupled with a generative adversarial network. We have used an architecture similar to theirs.

3. Method

In SR, the aim is to produce an HR image that has been downsampled to a low-resolution image

$$I_{LR} = d_\alpha(I_{HR})$$
$$I_{est} = f(I_{LR})$$

using a downsampling operator d_α. The task of image super-resolution is to provide an approximate inverse f of d, estimating I_{HR} from I_{LR}. The traditional deep learning approach is to learn f by a CNN, minimizing the L2 distance between the estimated HR image, I_{est} and ground truth I_{HR}. For our case this equation becomes,

$$I_{est} = f(I_{LR}, I_{ref})$$

where I_{ref} is the reference image additionally provided to the network. The size of I_{ref} is same as that of I_{HR}. An assumption that we worked on is that the network should already know which part of I_{ref} should be used while reconstruction of a specific portion of I_{LR}. For this purpose we scrambled the I_{ref} initially in patches and fed that to our network. Details are provided in the Experiments section.

4. Approach

4.1. Dataset

For the purposes of our experiment we used VGG's flowers dataset which contains 102 categories, each having more than 40 images. The images are of different sizes. We took a center crop to remove unnecessary background information and then scaled down the image to a size of 160×160. This served as our I_{HR}. We downsampled this again by a factor of 4, to get I_{LR}. Downsampling was done using 'bicubic' interpolation.

From each of these classes, we used three images which formed the dataset for our baseline model (model without I_{ref}). For all other experiments using this same subset of data, we created triples of the form $(I_{LR}, I_{HR}, I_{ref})$. Here the I_{ref} was chosen from the remaining two images of the particular class. During training, one of these I_{ref}'s was randomly picked. We felt this would help prevent the model from overfitting. An example triplet has been shown in Figure 2. We used data augmentation only for our baseline model.

4.2. Experiments

In this section we have reported the experimentation details [1]. We first mention the experiments that worked and later talk about the ones that didn't.

4.2.1 Baseline model

Our baseline model is a fully convolutional net similar to the one used by [13]. We used 6 residual blocks as opposed to 18, that they used in their experiments. Their network is based on the Residual architecture with a small modification. They have an additional skip connection between the start and the end of the residual blocks as can be seen in Figure 4. For upsampling they used a pixel shuffle layer proposed by [17] which is a much more computationally efficient. The network was trained on (I_{LR}, I_{HR}) image pairs using Adam optimizer with MSE as the loss. The image is not downsampled at any stage in the network since we don't want to lose out on the information that we already have.

4.2.2 Baseline + Scrambled I_{ref} image

As described earlier, before passing the I_{ref} to the network we scrambled its pixel-patches according to the similarity of LR image pixel-patches. More concretely, for each pixel-patch in the I_{LR}, the closest pixel patch in the I_{ref} was found. This new 'scrambled' I_{ref} is inputted into the network. This additional step is to help the network understand where to get the information from. We experimented

[1] Code can be accessed from : https://github.com/shreeshl/SISR

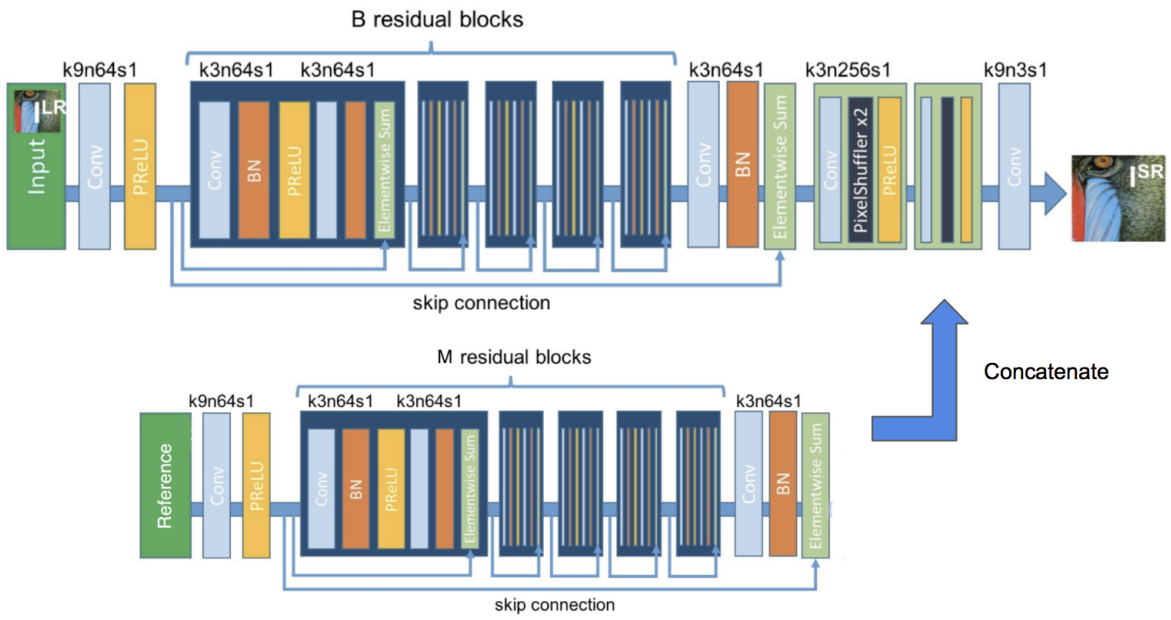

Figure 4. The architecture that we used (top portion of the image borrowed from [13]). Reference in the bottom network refers to the I_{ref}

with different metrics (such as L2, L1, HOG, hypercolumn vector [9]) to measure patch similarity. We stuck with L2 loss which seemed to work better than the rest. An illustration of these different metrics can be seen in Figure 3. A patch size of 40×40 was used for all experiments. Patches of smaller sizes were biased towards a few handful patches hence those sizes weren't used. A patch in I_{LR} is compared with all patches of size 40 in I_{ref} with a stride of 10 (for computational efficiency).

This scrambled image is now passed through a smaller but similar architecture as the baseline. This had 3 instead of 6 residual blocks. The convolutional volume obtained after passing the scrambled I_{ref} through this network is then concatenated to the volume after the upsampling layer of the baseline. We append this after upsampling layer for the same reason that was mentioned earlier. We don't want to lose out on the information of I_{ref} by downsampling it. There is a layer of filters that operates on this concatenated volume and finally reduces it to our required image size.

4.3. Baseline + Scrambled I_{ref} grayscale image

A similar experiment as above was tried but with a grayscale I_{ref}. This was to see if reducing the colour dependency makes the network focus more on other details.

4.4. Pretrained Baseline + Scrambled I_{ref} image

A similar experiment as above but with a pretrained baseline model. Network convergence was slow with the earlier models. Also since ideally this network should also be able to produce images atleast as good as the baseline, we reinforced this constraint by using the pretrained baseline. The results and convergence both were better.

4.5. Experiments that didn't work

We didn't find any prior work similar to ours hence there wasn't a clear guideline regarding what would work. A few such failures listed below.

- Minimizing the texture loss (used by Sajjadi et al [16] for SR, inspired by the style loss of Gatys et al [7]) between matched patches in the I_{est} and I_{ref}. The network didn't seem to converge.
- Since patch matching is more of a coarse similarity, we thought of using a more fine grained similarity. We passed both I_{LR} and I_{ref} through a pretrained VGG16 network. Convolutional volume obtained after 'conv3' layer is extracted for both images. This volumes are normalized and then multiplied to get a correlation matrix (Each volume has size $h \times w \times c$. This is reshaped to $hw \times c$. The correlation matrix thus obtained has size $hw \times hw$). This matrix essentially captures the similarity between any two pixels. We apply a softmax over it to get a probability distribution over the pixels in I_{ref}. A new 'dense' scrambled I_{ref} volume is obtained by multiplying this correlation matrix with I_{ref}'s original volume. In this 'dense' volume, each pixel in a particular location is a weighted combination of all pixels in I_{ref} (weighted by its similarity to pixels in I_{LR}). Since this volume has a size of $40 \times 40 \times c$, it

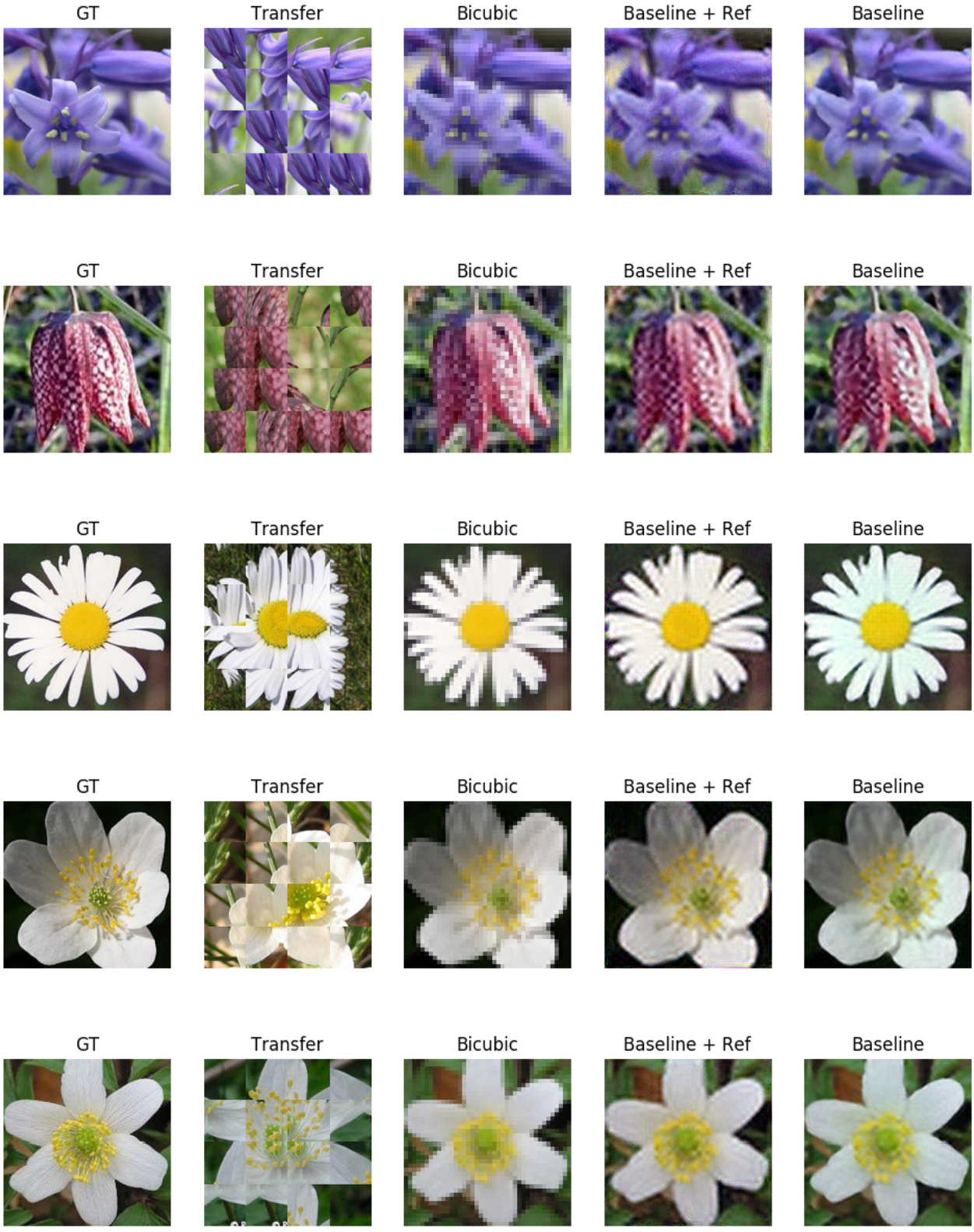

Figure 5. From left to right : Ground truth, Scrambled Reference Image, SR using Bicubic interpolation, SR with Pretrained Baseline + Reference, Baseline

couldn't be appended after upsampling layer since that had a size of $160 \times 160 \times \tilde{c}$. Therefore this was concatenated in the middle of the network. The network convergence was very slow and by looking at the I_{est}, it didn't look like it would converge.

5. Results

Multiple sets of images obtained using the Baseline and Pretrained Baseline + Scrambled have been displayed in Figure 6. The images used for testing are from another VGG flowers dataset which is a smaller dataset with 18 classes. As you can notice, the best results are obtained with baseline and the model with I_{ref} seems to give images with some checkerboard effects. The PSNR (Peak Signal to Noise Ratio) values have also been reported in Table 1. Although PSNR is a popular metric for SR evaluation, its values don't present a clear picture of the actual scenario. When you compare the SR of bicubic to SR with reference image, although noisier, the generated images looks perceptually much better for the latter but only have a slightly better PSNR.

Method	PSNR
Bicubic	23.69
Baseline	26.31
Baseline + Scrambled	24.07
Baseline + Greyscale Scrambled	24.09
PretrainedBaseline + Scrambled	24.53

Table 1. Comparison between PSNR scores of diff. methods

6. Discussion and Conclusion

In our project we aimed to experiment if SR could be improved by providing an additional reference image during training. Although it seemed likely that this would improve reconstruction, after our experiments we realized it is not a trivial task.

We experimented with different architectures to improve the traditional task of SR using a reference image. We weren't able to outperform the baseline scores. This might be due to our network not being deep enough or it needed more time to converge (our aim was not to get the best accuracy but to see what works and what doesn't) or just that our approach wasn't correct. We didn't perform extensive hyperparameter tuning too which could have improved the performance. The dataset that we worked with was a little harder as well. Perhaps a dataset related to buildings would have been a better choice to experiment with since the correlations are much easier to find. We were motivated to go ahead with the flowers dataset with the hope that, despite the reference image not being obviously similar, the network might be able to learn filters to detect the relevant information and improve reconstruction. Future work might also include incorporating the GAN framework in this experiment. It helped make the images more 'real' for SR in a recent paper by Ledig et al [13] and could work here too. Finally, the 'dense' way of scrambling the image seemed as the most general approach to tackle this and thus could be explored in more detail.

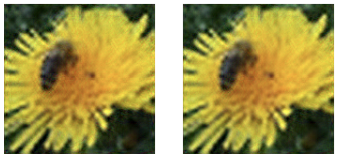

Figure 6. Example images obtained using Baseline + Scrambled and Pretrained Baseline + Scrambled. These images have a difference of 1.3 in their PSNR values but still no visible difference is noticeable

References

[1] Code based upon the implementation of https://github.com/twtygqyy/pytorch-srresnet.

[2] C. C. L. C. Dong and X. Tang. Accelerating the super- resolution convolutional neural network. *ECCV*, 2016.

[3] K. H. C. Dong, C. C. Loy and X. Tang. Learning a deep convolutional network for image super-resolution. *ECCV*, 2014.

[4] C. M. C.-Y. Yang and M.-H. Yang. Single-image super-resolution: a benchmark. *ECCV*, 2014.

[5] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. *NIPS*, 2016.

[6] C. Duchon. Lancazos filtering in one and two dimensions. *Journal of Applied Meteorology*, 18(8):10161022, 1979.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[8] D.-Y. Y. H. Chang and Y. Xiong. Super-resolution through neighbor embedding. *CVPR*, 2004.

[9] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 447–456, 2015.

[10] P. S. J. Bruna and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. *ICLR*, 2016.

[11] T. D. J. Donahue. Context encoders: Feature learning by inpainting. *CVPR*, 2016.

[12] A. A. J. Johnson and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *ECCV*, 2016.

[13] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial

network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017.

[14] C. G. M. Bevilacqua, A. Roumy and M.-L. A. Morel. ow-complexity single-image super-resolution based on nonnegative neighbor embedding. *BMVC*, 2012.

[15] R. R. R. Timofte and L. V. Gool. Seven ways to improve example-based single image super resolution. *CVPR*, 2016.

[16] M. S. M. Sajjadi and y. v. Bernhard Schölkopf and Michael Hirsch, journal=CoRR. Enhancenet: Single image super-resolution through automated texture synthesis.

[17] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[18] P. Y. Y. Y. X. Lu, H. Yuan and X. Li. Geometry constrained sparse coding for single image super-resolution. *CVPR*, 2012.

[19] H. Yue, X. Sun, J. Yang, and F. Wu. Landmark image super-resolution by retrieving web images. *IEEE Transactions on Image Processing*, 22:4865–4878, 2013.