Real world image super resolution with LapSRN

Machine learning exam

Francesco Benedetti

Prof. **Claudia d'Amato, Nicola Di Mauro**

a.a. 2020/2021

Abstract

This report describes an attempt at improving one of the several methods that over the years have been proposed for the task of Single Image Super Resolution (SISR). The method is based on **LapSRN** (Laplacian Super Resolution Network). Differently from the common practice of taking HR images and downscaling them by using the same function, the images in the dataset on which the model was trained and tested are downscaled by random functions, that change from an image to the other. This simulates a real world scenario where the downscaling function is not known and is not always the same.

1 Introduction

Image Super Resolution (ISR) is the task of improving the quality of an image by increasing its resolution. Several domains, such as the medical domain or security and surveillance, can gain advantages from this task. The progresses made in the machine learning and deep learning fields have allowed researchers to design and test a plethora of algorithms and models to deal with this task. Neural networks' capabilities of grasping end-to-end mappings between a low resolution input and the high resolution output can be exploited to create super resolution models.
The architectures for networks devoted to this task are mainly three [1]. **Pre-upsampling** networks upsample the LR image at the beginning, and then refine the higher resolution image by computing operations, mainly convolutions. This approach is computationally expensive since the operations are performed in an high dimensional space.
Post-upsampling frameworks perform operations in the opposite order: calculations are made in low dimensional space, and at the end the feature maps are expanded to the desired scale. This leads to lighter models, but the upsampling phase could introduce some noise which would need to be further refined.
A compromise between the aforementioned structures consists in the **iterative upsampling**: the image is iteratively upscaled and computations are performed until the desired scale is reached. The pioneer work adopting this approach is LapSRN [2], which is the base model for the case study being described.
Most of the state of the art techniques are supervised. The LR-HR image pairs are obtained by taking the HR image and adding it some noise in order to artificially decrease its quality. The most popular downscaling technique is bicubic degradation. The models trained with this data are not suitable for a real world scenario, since they learn to upscale images that have been polluted by the same function while in reality the image's resolution can be affected by a lot of factors. Real world ISR (RISR) deals with real world

data.
This case study consists in an attempt at applying LapSRN to a real world dataset.

2 Related works

Prior to the introduction of learning based ISR, images were upsampled by interpolation. Bicubic interpolation [3] is the most effective and is still used by learning based techniques as upsampling method inside the network. One of the pioneer works in learning-based ISR is SRCNN [4], that uses the pre-upsampling approach followed by two convolutional layers. Residual learning [5] has proved to be very effective for computer vision tasks, including ISR, and has been widely adopted. For instance, RFDN [6] is a lightweight residual network. Initially, the LR image is convolved to produce feature maps that are then processed by a **feature distillation block**. At each step, it splits the maps in two: the first part is retained and the other is fed to the next distillation step. A shallow residual block is attached at the end, consisting of a convolution, an identity connection and the activation unit. Lim et al [6] propose EDSRN, a deep residual network with post-upsampling, obtaining impressive results.
For the real world setting, the techniques are prevalently GAN-based and are mainly unsupervised, even though there are datasets containing images taken from the same position but changing the camera settings, in order to produce LR-HR image pairs. An example of supervised approach is KernelGAN [7].

2.1 LapSRN

LapSRN is one of the few approaches that adopt iterative upsampling for image super resolution. The model is composed of two branches: the **feature extraction block** and the **image reconstruction block**. The former is made of a stack of convolutional layers, each preceded by an activation layer (pre-activation) and without batch normalization since the authors of [6] found out that normalizing the features negatively affects the final result. The feature maps extracted become the input for a transposed convolution layer [8] that upsamples them by a scale of 2. A convolutional layer follows for compressing the feature maps into a single channel representation.
The image reconstruction block directly upsamples the image by applying transposed convolution. The result is summed with the output of the feature extraction block to obtain the output image, whose resolution is twice as the resolution of the input. Multiple pyramid levels can be chained one after the other. If we want a model that takes an input and upscales it by a factor S, we will need to chain $log_2 S$ pyramid levels. The structure of a network with three pyramid levels is depicted in Figure 1.

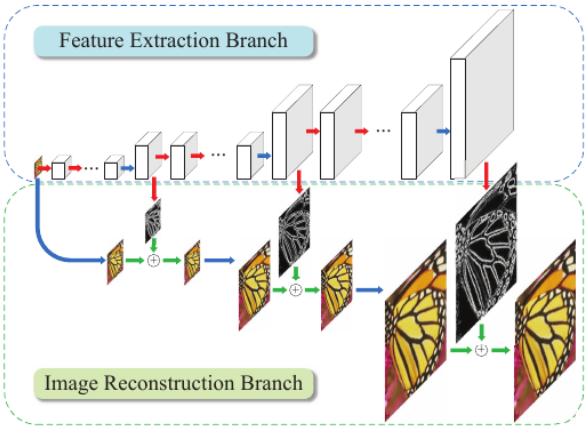


Figure 1: *Structure of the network*

3 Proposed procedure

The authors of LapSRN train the model on a dataset consisting of images downscaled by bicubic interpolation. My experiment consisted in making some changes to the model and training it on a more realistic dataset, with images downscaled by random functions. My hypothesis is that the progressive upsampling adopted by LapSRN could make this model accomplish good results when improving the resolu-

tion of images where the downscaling function is unknown.

3.1 Dataset

A real world dataset suitable for supervised learning is DIV2KRK[1] [7]. The authors took the HR images in the validation set of the popular DIV2K dataset [9] and blurred and downsampled each of them with a different, randomly generated kernel. Random blur kernels are 11×11 anisotropic gaussians, and each of them is with two independently distributed lengths $\lambda_1,\ \lambda_2 \sim U(0.6,\ 5)$ and a rotation angle $\theta \sim U[-\pi, \pi]$. Further, uniform multiplicative noise is applied to the blur kernel before normalizing its sum to one. This dataset is still synthetic, but closer to a real world one since the images are downscaled by random and different functions. The authors downscaled the HR images first by a factor of 2, then by a factor of 4.
Images in the dataset were converted to YCbCr format, and only the Y channel (luminance) was considered. This is a common practice for super resolution methods and the authors of LapSRN did the same.

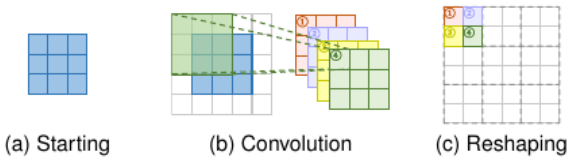


Figure 2: *Subpixel convolution*

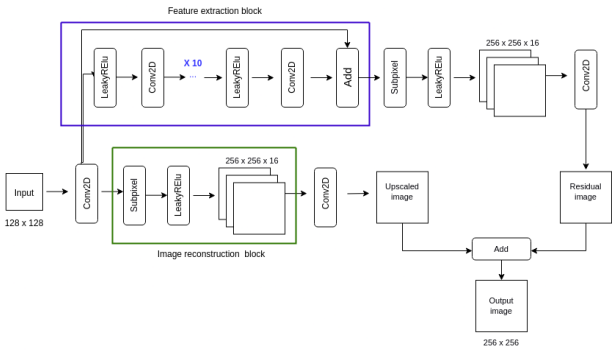


Figure 3: *Schema of one network level*

3.2 Model

The model is LapSRN, but some modifications were tried. The feature extraction block is made of 10 convolutional layers with LeakyRElu as pre-activation function. Local residual learning is used. More precisely, a skip connection links the input to the output of the last convolutional layer, which are summed. The authors of LapSRN use a transposed convolution layer to upsample the residual image, but this can cause uneven overlaps in the output, with consequent checkerboard-style artifacts as discussed in [10]. An alternative is represented by the **sub-pixel convolution**.

3.3 Subpixel convolution

Subpixel convolution was proposed in [11] and consists in upsampling the image by using the contribution of pixels coming from different channels. Those channels are generated by convolution without applying non linearities, and the number of channels to generate must be equal to the square of the scale. In our setting, we have an input of shape (W, H, C) and we want the layer to upscale it by a factor $S = 2$, hence the convolution must generate 4 channels. A reshaping operation is applied to produce outputs of shape $(Ws, Hs, C/S^2)$. The reshaping is carried out by a shuffling operator that, for computing the pixel value at a certain position, considers the contribute of the pixels in that position in each feature map. A graphic representation of how subpixel convolution works is shown in Figure 2.

In LapSRN, I replaced the transposed convolution with subpixel convolution. Figure 3 depicts the schema of one network level. The output can be used as input of an additional level which would make the model able to upsampling images by a scale of 4. An arbitrary number of levels can be stacked, provided that the appropriate LR-HR image pairs are available.

[1] http://www.wisdom.weizmann.ac.il/ vision/kernelgan/

3.4 Loss functions

Given a low resolution image I^{LR} and an HR image I^{HR}, we want to learn a function f, able to predict an HR image $\hat{I}$ as similar as possible to I^{HR}. So $\hat{I} = f(I^{LR}, \theta)$. θ is the set of network parameters that the model must learn in order to provide the best prediction. There are several loss functions particularly suitable for this task. In the one proposed by the authors of LapSRN, the loss for a single pairing is computed as:

$$Loss_S(I^{HR}, \hat{I}) = \rho(I^{HR} - \hat{I})$$

where ρ is the **Charbonnier penalty function** and is defined as $\rho(x) = \sqrt{x^2 + \epsilon^2}$. The value for ϵ is set to 1×10^{-3}. The loss for a batch is the average of the losses computed for the single images.

Perceptual loss

An other loss function is the perceptual loss, introduced in [12]. It uses the VGG19 [13] which, being trained on a very large set of images such as Imagenet, can act as a sort of judge who has to evaluate the result of the super resolution process. This loss is aimed at producing images that look better to humans, at the expense of achieving lower results in metrics like the PSNR, that are roughly based on pixel differences and ignore some details that the human eye can catch. We indicate with ϕ_{ij} the feature map extracted from the j^{th} layer of the network before the i^{th} pooling layer. Then, the features that the network will compute at that layer can be interpreted as the scores that the network gave to the input image. The value of the loss will be the Euclidean distance between the feature maps computed for the ground truth HR image and the ones computed for the predicted image.

$$L_S(I^{HR}, \hat{I}) = \frac{1}{W_{ij} H_{ij}} \sum_{x=1}^{W_{ij}} \sum_{y=1}^{H_{ij}} (\phi_{ij}(I^{HR})_{xy} - \phi_{ij}(\hat{I})_{xy})^2$$

This loss function was tried in this case study, to see if it could be helpful for real world SR and produce better results with respect to the Charbonnier loss. However, this didn't happen. The perceptual loss values were low since the beginning epochs, and this prevented the model from adjusting its parameters. In fact, the PSNR results got stuck between 6 and 7 for both the models, a very low value. Probably this happened because the VGG19, which is trained on RGB images, was asked to compute features for images consisting of one channel (luminance) concatenated thrice with itself to simulate a three-channels representation. This is an unusual input for the network that ended up producing similar feature maps for both the ground truth images and the images produced by the model during training. If the feature maps are close, their distance (hence the loss) is low. Moreover, this loss function significantly slows down the training process.

3.5 Implementation and training details

The project was realized in Python 3.9, using the Keras library for implementing the models. Each feature extraction block contains 10 convolutional layers. Each convolutional layer has 64 filters with kernel size of 3. Zero padding is used to prevent the convolution from decreasing the image shape. I used Adam [14] as optimizer. As mentioned, the non linear activation function is the LeakyRElu [15], with $\alpha = 0.02$. The training environment was Google Colab, which offers a Nvidia Tesla K80 GPU. The models were trained for 100 epochs with early stopping based on the PSNR values obtained at each epoch. The learning rate was set to 1×10^{-3} and decayed every 1000 steps by exponential decay with decay rate equal to 0.95. The batch size for the $\times 2$ model is 16, while for the $\times 4$ model it is 4.

The original set of 100 HR images (along with their LR counterparts) was split as follows: **80** images for the training set, **15** for the test set and **5** for validation. Each pixel in the images is scaled to the 0-1 range. From each of the downscaled images, 100 random patches of size 128×128 were extracted. Each patch was randomly rotated of 90°, 180° or 270° and randomly flipped. For each LR patch, the corresponding patch in the HR image was extracted in order to create a LR-HR pair. At the end of the procedure, two datasets were obtained: one for the $\times 4$ super resolution model (where the HR patches are of size 512×512) and one for the $\times 2$ super resolution

model (HR patches are of size 256×256). In both datasets, the training set contains 8000 patches, the test set contains 1500 patches and the validation set is made of 500 patches. Due to computational limitations, the dimension of the $\times 4$ dataset was reduced to 2000 patches for training, 100 for validating and 500 for testing.

4 Experiments

The models were evaluated on the test sets, using Peak Signal-to-Noise Ratio (PSNR) as evaluation metric. Given the ground truth image I and the predicted image $\hat{I}$, PSNR is defined by the formula:

$$PSNR = 10 \times log_{10}(\frac{L^2}{\frac{1}{N}\sum_{i=1}^{N}(I_i - \hat{I}_i)^2})$$

Where L is the maximum pixel value (in our case it is equal to 1 since all the pixels fall in the 0-1 range), and N is the number of pixels. To actually use the system for upscaling an image we must:

- Convert the image to YCbCr format and consider its Y channel;
- Feed it to the network, which will produce its prediction;
- Take the Cb and Cr channels of the original image and upsample them via bicubic interpolation;
- Stack the upsampled channels and convert the resulting image back to RGB.

5 Results and discussion

In order to provide some comparison, bicubic interpolation was used as baseline method for upsampling the images in the test set. The average PSNR results obtained by the two techniques are reported in Table 1. It is clear how poorly the X2 model performs. A visual proof is the comparison in figures 4 and 6, where we can see how the upsampling process significantly deteriorates the images without bringing any improvement. If you notice, the model for doubling the resolution of the image actually performs post upsampling, since it upsamples only once. The results suggest that a post upsampling model is not suitable for real world super resolution, or at least it would need to extract more features before being able to produce acceptable results.

	X2	X4
Interpolation	25.601	20.816
MyModel	12.957	21.777

On the other hand, the model for the X4 upsampling achieves higher PSNR than the baseline and, as showed in figures 5 and 7, produces better looking results since it enhances some details that interpolation does not catch. As a side effect, some artifacts can be noticed. This is one of the drawbacks of subpixel deconvolution, as discussed in [16].

6 Conclusions and future works

In this work, LapSRN was used for Real world Image Super Resolution, with some modifications to the original structure. The model performed very badly on X2 super resolution while the results on X4 SR were significantly better. This suggests that an iterative upsampling structure can work well in scenarios where images' resolution is downsized by unknown functions.

As future works, some changes on the network may be tried, such as increasing the number of convolution layers in the first level to see if they improve performance on X2 super resolution. Also, an other model could be implemented to realize X8 super resolution and see if the performances keep improving as the number of model components grows. This could be done only if an appropriate dataset is found or created. Lastly, the model could be trained and tested with RGB images instead of using only the Y channel of YCbCr images.

Figure 4: X2 model

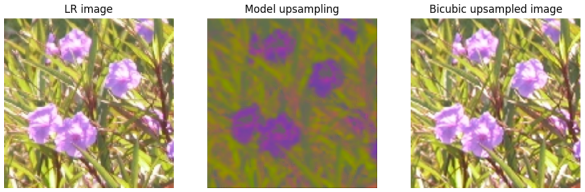


Figure 5: X4 model

Figure 6: X2 model

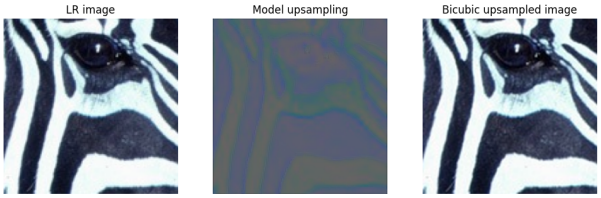
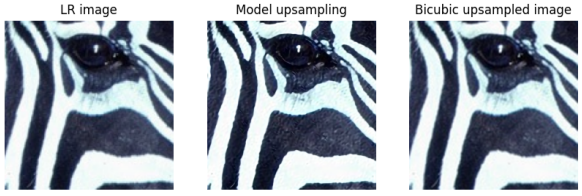


Figure 7: X4 model



References

[1] WANG, Zhihao; CHEN, Jian; HOI, Steven CH. Deep learning for image super-resolution: A survey. IEEE transactions on pattern analysis and machine intelligence, 2020, 43.10: 3365-3387.

[2] Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks

[3] R. Keys, "Cubic convolution interpolation for digital image processing," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 29, 1981.

[4] DONG, Chao, et al. Learning a deep convolutional network for image super-resolution. In: European conference on computer vision. Springer, Cham, 2014. p. 184-199.

[5] HE, Kaiming, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.

[6] LIM, Bee, et al. Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2017. p. 136-144.

[7] Bell-Kligler S, Shocher A, Irani M. Blind super-resolution kernel estimation using an internal-gan. Advances in Neural Information Processing Systems. 2019;32.

[8] GAO, Hongyang, et al. Pixel transposed convolutional networks. IEEE transactions on pattern analysis and machine intelligence, 2019, 42.5: 1218-1227.

[9] 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.

[10] Odena A, Dumoulin V, Olah C. Deconvolution and checkerboard artifacts. Distill. 2016 Oct 17;1(10):e3.

[11] Shi W, Caballero J, Huszár F, Totz J, Aitken AP, Bishop R, Rueckert D, Wang Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 1874-1883).

[12] Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, Shi W. Photo-realistic single image super-resolution using a generative adversarial network. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 4681-4690).

[13] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.

[14] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014 Dec 22.

[15] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in International Conference on Machine Learning, 2013

[16] AITKEN, Andrew, et al. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. arXiv preprint arXiv:1707.02937, 2017.