Reimplementation of "Enhanced Deep Residual Networks for Single Image Super-resolution"

Robin Gratz Sönke Lülf Ludger Masch

April 4, 2021

1 Introduction/Task

The majority of people these days would probably not leave their apartment or house without a very personal item: their smartphone. Even though some would like to deny it, this device has become very important in our everyday lives to support us navigating to the next fancy restaurant, taking pictures of our holiday trip or chat with our friends about the homework for the next day. It's incredible how powerful these tiny computers became in the past decade. They yearly increase in computational performance, the efficient usage of energy and especially in pushing the limits of picture and video quality by not only developing better camera sensors or lenses but also using the power of machine learning and AI (or in general very clever algorithms) to enhance the image quality to an even higher extent. Especially thanks to the continuous improvement of these algorithms, it has become possible for example to get a higher dynamic range, generate real looking bokeh effects around a subject and drastically increase the amount of detail/resolution in an image. These big improvements in image quality over the last years resulted in a kind of uselessness in small digital cameras, people were just switching to their smartphones which they would carry around nevertheless.

A big topic in these image enhancement techniques is called Single Image Super-Resolution (SR). It's task is to increase the resolution of a single low resolution (LR) image to a certain amount (most of the time x2, x3 or x4) to get a high resolution (HR) image. Super-Resolution is not only applied in the camera software of mobile devices but is also for example used for surveillance to recognize faces from low-resolution images, satellite images, in media to reduce server costs or speed up transmission speeds of high resolution video or photo material, and in medicine where capturing high-resolution MRI images is very hard. Using Deep Learning approaches to tackle this problem drastically improved these reconstructions and outperform most of the previous used methods. Obviously there are many different proposed neural network models and even different architectures. We will discuss some other relevant ones in the chapter "Related approaches". For our project we chose a very pioneering architecture, the single-scale baseline model of Lim et al. [3], who used a deep

residual convolutional neural network (CNN). With their work, they won the NTIRE challenge on Single Image Super-Resolution in 2017 [5]. In this yearly challenge, scientists compete for the best results of their models on a validation data set.

In Super-Resolution tasks like in this one it's often the case that the input is a bicubic downsampled image and the target is the original high-resolution image. However, sometimes the model is also tested and evaluated on other resampling methods. Other image-enhancement tasks include for example blur- and noise removal.

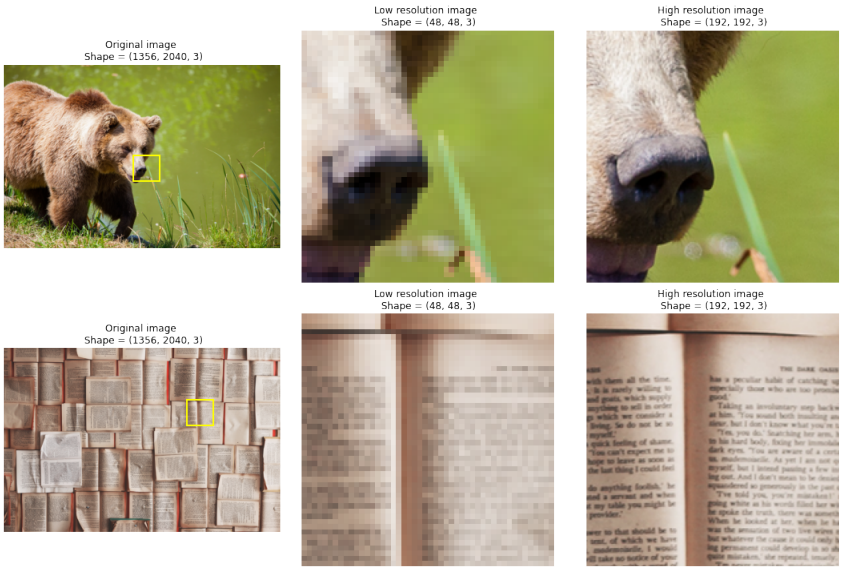

Figure 1: Comparison of low-resolution and high-resolution image (scale x4)

2 Related approaches

In this chapter we will talk about the development of (especially Single Image-) Super-Resolution with Deep Learning, also focusing on important loss functions and metrics like PSNR and SSIM.

The most simple approach and earliest solution when it comes to increasing the size of an image is to use interpolation schemes. The low resolution image is resized by a certain factor, pixels which were added in between the already known pixels are then interpolated. There are many different techniques like bilinear, bicubic, nearest neighbor or lanczos interpolation. Obviously, those resampling methods have their limitations. When convolutional neural networks became very successful in solving Computer Vision tasks like image classification or segmentation it was just a matter of time when they were also applied

in Super-Resolution tasks like Dong et. al. [1] did. Their CNN called SRCNN (for Super-Resolution CNN) consists of three layers, the patch extraction-, non-linear- and reconstruction layer. Here the image is upsampled by bicubic interpolation first and then fed into the previously mentioned layers. This is called Pre-Upsampling Super-Resolution since feature extraction is done in the high resolution space. The network is then trained by the MSE loss.

As the development of CNNs went on, ResNets (residual CNNs with skip connections) allowed for deeper and more powerful models with better accuracy and were also applied for Super-Resolution tasks which showed significantly improved results: Ledig et. al. [2], with their SRResNet, were the first who brought this improved architecture to this specific task. In comparison to the previously mentioned model, feature extraction is done in the lower resolution space before upsampling the image in the very end (this is called Post-Upsampling). Therefore, the computational cost is reduced a lot, making training a lot faster. Furthermore, instead of using bicubic interpolation, upsampling is done by a transposed convolution layer, such that the model becomes end-to-end trainable. The EDSR model by Lim et. al. [3] which we have reimplemented is a further development of the SRResNet with even more improved results in terms of PSNR (peak signal-to-noise ratio) and SSIM (structural similarity index measure). Both models were trained on a pixel-wise loss function, SRResNet on MSE and EDSR on the mean absolute value. However, there is one drawback using these kinds of loss functions: the resulting images lack perceptual quality, meaning that they appear very smooth without any high-frequencies and structure. Ledig et. al. [2] therefore proposed the SRGAN with the perceptual loss, a very different architecture than the ones used before. This generative adversarial network (GAN) provides very photo-realistic looking images with high perceptual quality. The generator here learns to generate high-resolution images from low-resolution images while the discriminator then learns to distinguish generated images from real images. However, the PSNR is on average much lower than images produced from EDSR or SRResNet models since it's also computed pixel-wise. Therefore, other measures have to be considered as well.

3 Our models

Because we are dealing with images, using convolutional NNs is a natural choice. Over the last years residual networks have proven to be a very powerful method when it comes to super-resolution- and computer vision tasks in general.

3.1 Residual blocks

Inspired by Lim and colleagues, we keep the residual blocks as simple as possible. One block consists of a convolutional layer, followed by a ReLU activation layer, followed by another convolutional layer. We do not use any batch normalization because the normalization reduces the flexibility of feature range and increases

the demand for memory [3, p.3]. We used 64 filters for the convolutional layers of each residual block. Further, we chose a kernel size of three and set padding to 'same' in order not to change the input size for each layer.

3.2 Upsampling layers

Because the target has a higher resolution than the input, the number of pixels differ. Therefore, in order to increase the quality we have to increase the number of pixels. The factor by which the number of pixels needs to be increased is the same as the factor by which the HR image was downsampled to create the LR image.

Thus, we need different configurations for each model. For simplicity and because we want to make use of weight initialization (see : 4.3) we differentiated the different cases within our universal upsampling block using if-conditions.

For the scales two and three we use one convolutional layer with the same number of filters as in the residual blocks, a kernel size of four in order to avoid checkerboard artifacts [4] and padding set to 'same'. In order to increase the number of pixels by the right amount we set strides to the factor, by which the image should be scaled up to. When handling a scale of four we do not set the strides to four but instead use two transposed convolution layers successively with strides set to two, to create the same effect, but with more trainable parameters.

3.3 General architecture

Because upsampling is done in the very end of our model, the EDSR is a post-upsampling network. Our three final models (one for each scale of x2, x3 and x4) start with one convolutional layer (64 filters, kernel size of 3 and the padding option set to 'same'). After that, the output of this layer gets saved for a skip-connection. Then our models have 16 residual blocks. This is followed by one more convolutional layer (again 64 filters, kernel size of 3 and the padding option set to 'same'). At this point, the saved output of the first convolutional layer is added and the result is fed into the output block. This block consists of the upsampling layers (3.2) and a final convolution layer. The last convolution layer has three filters in order to construct the three colour channels of an image.

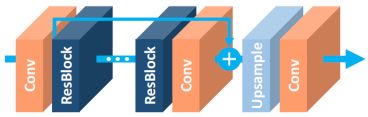

Figure 2: Achritecture (Fig. by Lim et al. [3])

3.4 Functionality

- Input layer: Applies 64 filters for the first time so that every residual block gets the same input shape so that adding input and output is possible.
- Residual blocks: The residual blocks extract the features of the LR image.
  - ReLU: As usual for convolutional layers we chose ReLU as the activation function.
- Second convolutional layer: Feature extraction in the end.
- Skip-connection: Makes the training process more stable by avoiding vanishing gradients.
- Upsample layers: Takes the extracted features of the residual blocks and reproduces them on a larger scale.
- Readout layer: Reduces the number of filters from 64 to three and turns the abstract features back into something that can be interpreted as an image with colour.
  - Sigmoid function: makes sure that the values of each pixel are in the same range as the target pixels (0-1).

4 Training

4.1 Data handling

In the input pipeline we first linearly scale the input images (low resolution) normally around 0 and normalize the label images (high resolution) to a range from zero to one. As a result, labels and predictions are in the same value range because the readout-layer of our EDSR network uses a sigmoid function. After that, we resize the images to a quadratic shape. The labels have a size of 600 by 600 while the input images have a fraction of that depending on the scaling factor (e.g. 300 by 300 with scale factor 2). We chose a mini-batch size of four because we trained our models using Google Colaboratory and greater mini-batch sizes exceeded the RAM limitations. We prefetched two mini-batches, again because of the trade-off between training speed and the limitations.

We trained the network using quadratic patches of 48 by 48 pixels (like in the original paper) as input and the respective area of the HR image as the label. These patches were augmented by randomly flipping or rotating them by 90 degrees.

4.2 Hyperparameters

- Epochs: We chose 200 epochs because the training times were feasible and the results already show the potential of the network.

- Number of blocks: The number of blocks is set to 16 as in the baseline model of Lim et al. that we tried to reimplement [3].
- Number of filters: Inspired by the baseline model again we use 64 filters for each residual block [3].
- Learning rate: We initialized the learning rate with 10^{-4} and halved it four times (every 40th epoch) in order to make the training process more stable.
- Optimizer: Like Lim et al. we use ADAM as an optimizer [3, p.5].
- Running average factor: Based on the course we used a running average factor of 0.95.
- Loss: In the past L2 loss has been used for image restoration and also for super-resolution. However, Zhao et al. [6] have shown that L1 loss can result in a better performance. We therefore use the mean absolute error.

4.3 Weight initialization

We started our training process by training the x2 model for 200 epochs. Before training, we initialized the x3 and x4 models with the weights learned by the x2 model for faster convergence and a better performance [3, p.4].

5 Results

This faster convergence can be seen in the results where the loss of both initialized models starts at a lower point and decreases faster during the fist updates (See: Fig 3).

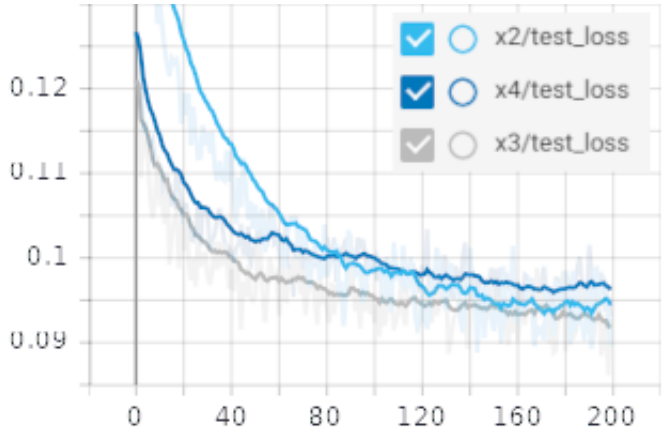

Figure 3: Loss comparison

One strategy to further improve the results on a super-resolution task is to train multiple networks, feed them the same input and calculate the mean over all outputs. Because training multiple networks is computationally expensive Lim et al. used a strategy called 'Geometric Self-ensemble' [3, p.5]. Instead of feeding multiple different networks the same input, they feed the same network eight augmentations (all combinations of flips and rotations, including the original) of the input. After inverting the augmentations, the mean over all outputs gets calculated. We used the same method and marked it as 'EDSR+'.

Figure 4 shows what one of our basic models (the x3 model) is capable of after only 200 epochs of training. Furthermore, it shows how the Geometric Self-ensemble method improves the perceived quality of the image, by getting rid of artifacts to the left of the wing and the measured quality by increasing the PSNR value.

Figure 4: Example image of the x3 model

Because of computational limitations we could only reimplement a lightweight version of the proposed network. This makes it hard to compare the results to the original paper, but even this lightweight version clearly shows the potential of this approach as you can see in the example images at the end of the document.

References

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *arXiv:1501.00092*, 2014.

[2] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv:1609.04802*, 2016.

[3] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. July 2017.

[4] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. https://distill.pub/2016/deconv-checkerboard/, 2016.

[5] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[6] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for neural networks for image processing. *arXiv:1511.08861*, 2015.

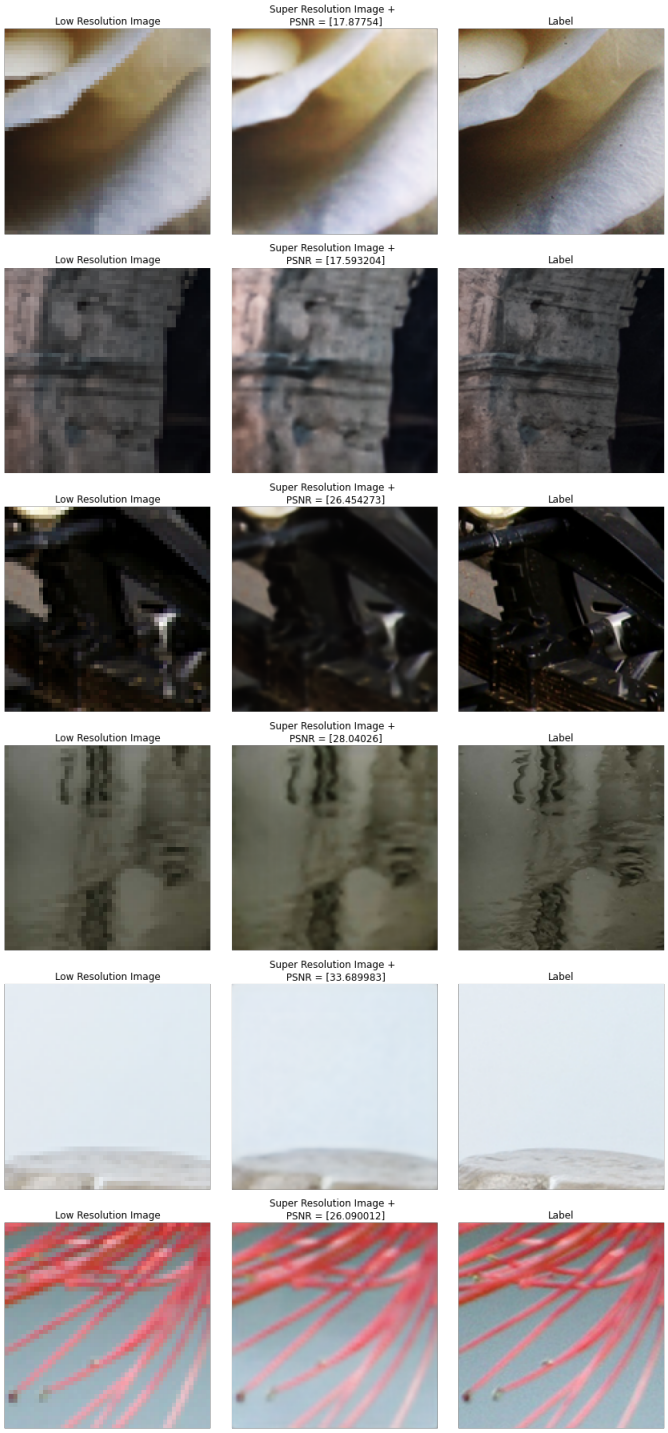

Figure 5: Overview of x3 Images

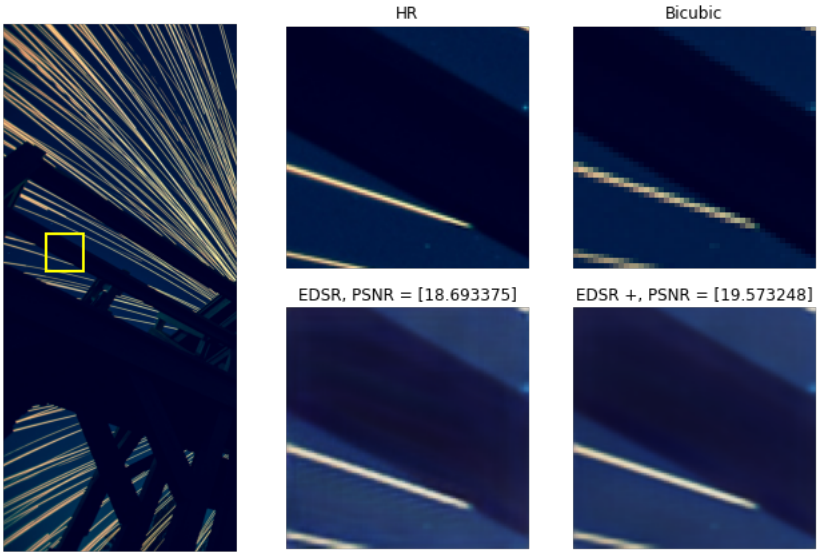

Figure 6: Example image of the x2 model

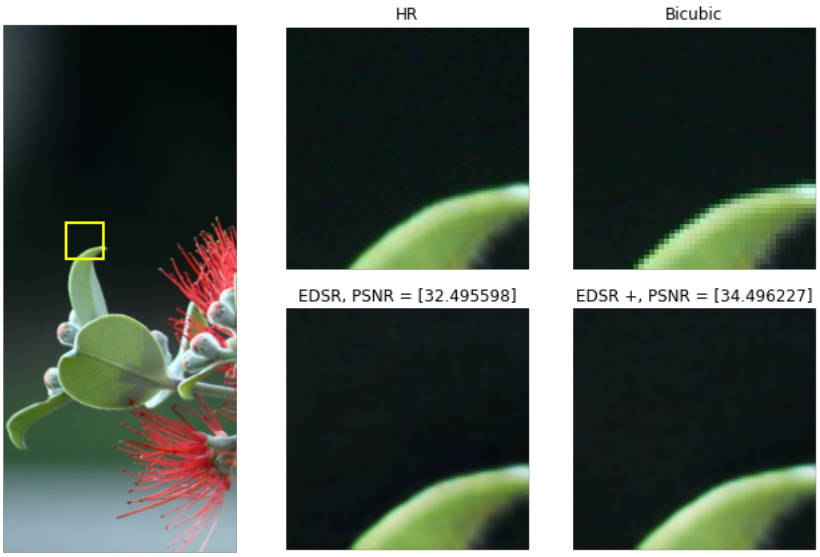

Figure 7: Example image of the x2 model

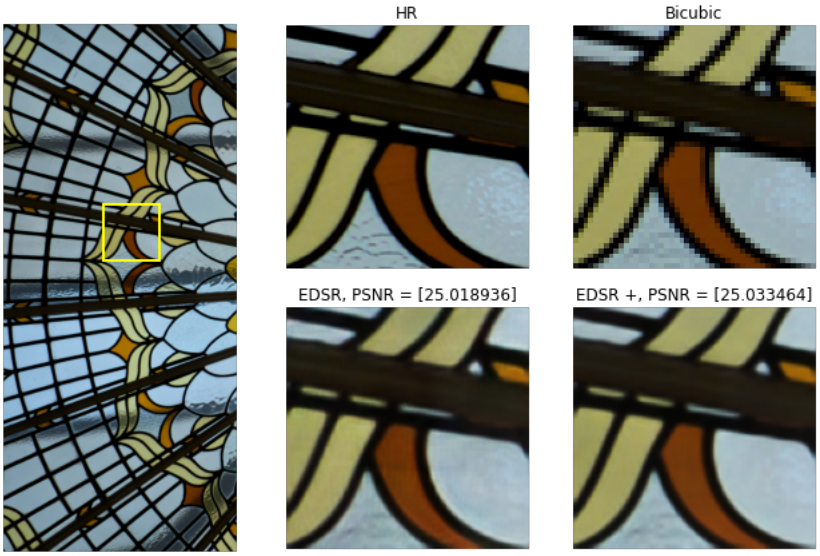

Figure 8: Example image of the x3 model

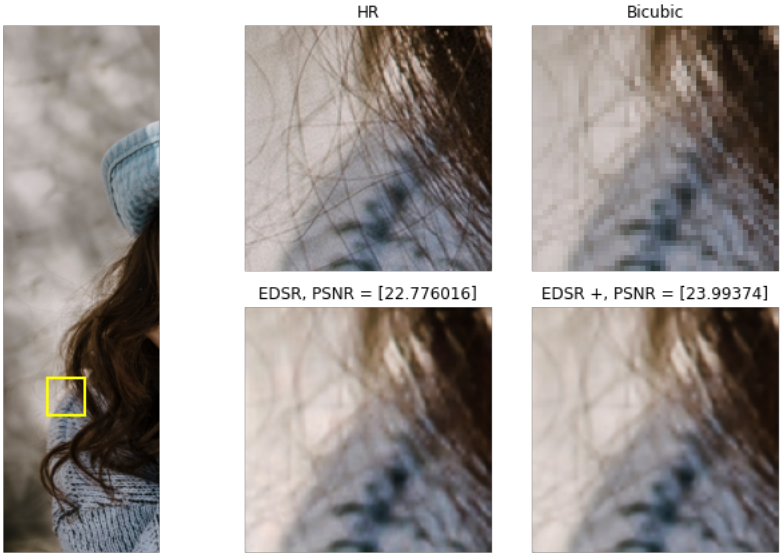

Figure 9: Example image of the x3 model