IMAGE DEBLURRING AND SUPER-RESOLUTION USING DEEP CONVOLUTIONAL NEURAL NETWORKS

Fatma Albluwi, Vladimir A. Krylov & Rozenn Dahyot

School of Computer Science & Statistics, Trinity College Dublin, Ireland

ABSTRACT

Recently multiple high performance algorithms have been developed to infer high-resolution images from low-resolution image input using deep learning algorithms. The related problem of super-resolution from blurred or corrupted low-resolution images has however received much less attention. In this work, we propose a new deep learning approach that simultaneously addresses deblurring and super-resolution from blurred low resolution images. We evaluate the state-of-the-art super-resolution convolutional neural network (SRCNN) architecture proposed in [1] for the blurred reconstruction scenario and propose a revised deeper architecture that proves its superiority experimentally both when the levels of blur are known and unknown a priori.

Index Terms— Image super-resolution, deblurring, deep learning, convolutional neural networks.

1. INTRODUCTION

Single image super-resolution (SR) is an essential application in computer vision that proves useful in multiple areas such as remote sensing image processing, security systems, medical imaging, etc. The SR task is an ill-posed problem, where one LR image has many solutions for HR image. In the early 1980s, Tsai [2] has addressed the topic of image super-resolution for the first time. Since then the problem has received a lot of attention, and typical state-of-the-art methods perform as example-based approaches where learned prior information alleviates the problem of multiple solutions [3].

The example-based methods are divided into two kinds: internal methods [4, 5, 6] and external example-based methods [7, 8, 9, 10, 11, 12]. The pipeline of most external example-based approaches where learned prior information alleviates the problem of multiple solutions [3]. The example-based methods are divided into two kinds: internal methods [4, 5, 6] and external example-based methods [7, 8, 9, 10, 11, 12]. The pipeline of most external example-based approaches where learned prior information alleviates the problem of multiple solutions [3].

We begin by briefly reviewing the SRCNN architecture proposed by Dong et al. [1] and its optimization procedure in Sec. 2.1. To tackle simultaneously deblurring and SR we propose a novel DBSRCNN architecture presented in Section 2.2.

2. SUPER RESOLUTION AND DEBLURRING

The (9-1-5) architecture of SRCNN corresponds to a relatively small network that contains 8,032 parameters and is composed of 2 hidden layers: added to the input and the output layers there are 4 layers in total. The objective is to learn a mapping function $F$ performing three tasks: patch extraction, non-linear mapping, and reconstruction. The structure of the network is defined as follows [1]:

- **Input Layer**: the input $x$ is 2-dimensional representation of the sub-image with $c = 1$ for grey level image...
Patch extraction and representation The first hidden layer extracts overlapping patches from the input sub-image and represents each patch as a high-dimensional vector. It uses a Rectified Linear Unit (ReLU) activation function \( F_1 \), kernel size \( f_1 = 9 \) and contains \( n_1 = 64 \) feature maps:

\[
F_1(x) = \max \left(0, W^{(1)} \ast x + b^{(1)}\right),
\]

where \( W^{(1)} \) contains \( n_1 \) filters of size \( c \times f_1 \times f_1 \) that produce \( n_1 \) feature maps. \( b^{(1)} \) is a \( n_1 \)-dimensional bias vector; \( \ast \) denotes the convolution operation.

Second order mapping The second hidden layer maps each high-dimensional vector of the previous layer to another high-dimensional vector, which is the representation of a high-resolution patch. ReLU activation function is employed with \( n_2 = 32 \) feature maps and filter size \( f_2 = 1 \):

\[
F_2(x) = \max \left(0, W^{(2)} \ast F_1(x) + b^{(2)}\right),
\]

where \( W^{(2)} \) involves \( n_2 \) filters of size \( n_1 \times f_2 \times f_2 \) to produce \( n_2 \) feature maps. \( b^{(2)} \) is the bias vector.

Reconstruction operation The output layer produces the reconstructed SR image by aggregating the patch-wise representations:

\[
F(x) = W^{(3)} \ast F_2(x) + b^{(3)}
\]

where \( W^{(3)} \) includes \( c \) filters of size \( n_2 \times f_3 \times f_3 \) with filter size \( f_3 = 5 \). \( b^{(2)} \) is a \( c \)-dimensional bias vector, associated to the number of image channels.

The structure of SRCNN can be written as a network with 3 layers (9-1-5) (64-32-1).

SRCNN Optimization The filter weights in each layer are initialized by drawing randomly from a Gaussian distribution with zero mean and standard deviation 0.05, the biases set to 0, and the learning rate equal to 0.001. The aim is to recover SR image \( F(x) \) from LR \( x \) that is as similar as possible to the original HR image \( y \). The estimation of \( \Theta = \{W^{(1)}, W^{(2)}, W^{(3)}, b^{(1)}, b^{(2)}, b^{(3)}\} \) is required to specify the end-to-end mapping function \( F \). To this end, the cost minimization between the reconstructed images \( F(x, \Theta) \) and its original HR images \( y \) is performed. The MSE function \( C(\Theta) \) is employed as the cost function:

\[
C(\Theta) = \frac{1}{n} \sum_{i=1}^{n} ||F(x_i; \Theta) - y_i||^2
\]

This cost function is minimized using Adam [22], which is used to optimize the network at faster convergence rates.

De-blurring with DBSRCNN

The proposed network aims to learn an end-to-end mapping \( F \), which takes the blurred LR image \( x \) as input, and directly provides the deblurred HR reconstruction \( F(x) \). Our architecture includes four convolutional layers in addition to a concatenation layer as demonstrated in Figure 1.

The first layer is feature extraction to compute the low-level features and contains 32-feature maps (32 filters) with filter size \( 9 \times 9 \). The second layer is a feature enhancement layer which provides enhanced features from the output of the first layer, and contains 32-features maps with filters of size \( 5 \times 5 \). The third layer concatenates features from the first two layers creating a merged vector with low-level and enhanced features. This layer contains 64-features maps or 32-features maps depending on the operation defining the merger procedure. Such operations include summation, maximum, subtraction, averaging, multiplication, concatenation. All these operations except the last one take the same size of inputs and return the same shape. Concatenation allows inputs of different sizes. We have empirically observed the best performance associated with the concatenation operation. The fourth layer performs the second-order mapping. The fifth layer reconstructs the output HR image. The operations of the proposed network can be described as follow:

\[
F_i(x) = \max \left(0, W^{(i)} \ast F_{i-1}(x) + b^{(i)}\right), \quad i \in \{1, 2, 4\}
\]

\[
F_3(x) = \text{merge} \left(F_1(x), F_2(x)\right)
\]

\[
F(x) = W^{(5)} \ast F_4(x) + b^{(5)}
\]

where \( W^{(i)} \) and \( b^{(i)} \) are the filters and biases. \( W^{(i)} \) is comprised of \( n_i \) filters and \( n_o \) is the number of channels in the input image. \( F_i(x) \) are the feature maps and \( F(x) \) is the reconstructed output image which has the same size as the input image. The activation functions used are ReLU. Mean Squared Error (MSE) is used as the cost function, and the cost function is minimized using Adam optimization.

Experimental Results

Dataset & image degradation model The training dataset is composed of 91 images taken from Yang et al. [23]. The test datasets are denoted “Set5” (5 images) [8] and “Set14” (14 images) for SR reconstruction. The results are consistently better than state-of-the-art methods, as observed in Table 1.
images) [24]. We make this choice of training and test data to allow a fair comparison with [1] where they were employed. To fully exploit the available data we rely on augmentation: HR images from the training set are randomly cropped to obtain \( f_{sub} \times f_{sub} \times c \) pixel sub-images. We employ sub-images of size \( f_{sub} = 33 \), thus the 91 HR images can be divided into 21,824 training sub-images with stride 14. The model is trained on sub-images, and the inference on the whole image.

To create a single blurred LR sub-image \( x_i \) (input) for training and testing, the HR sub-images \( y_i \) are first blurred using a Gaussian kernel noted \( blur_{i} \) (with standard deviation \( \sigma = i \)). Secondly images are down-sampled using the down-scaling factor, and then up-sampled using bicubic interpolation to the HR input resolution. Thus, the sizes of the input and output images of our network are equal. The down/up scaling factor employed in this study is \( s = 3 \).

### Computation Time
The Python implementation of SRCNN [25] uses Keras-1 with Theano library as a backend. We converted the code to keras-2 and switched the backend to TensorFlow. To render the SRCNN training more computationally efficient, our implementation relies on Adam optimization. The training time of SRCNN in this implementation was 8.33 minutes with NVIDIA GTX 1050 GPU. Dong et al. [1] indicate that the (9-5-5) SRCNN network achieved better performance than (9-1-5) SRCNN network but at the cost of training time. In our implementation of (9-5-5) SRCNN network, the training time was around 11 minutes.

### 3.1. Evaluation of SRCNN
To fully evaluate the performance of SRCNN on SR of blurred images we test several different scenarios:

- **Non-blind scenario:** four pipelines are trained on images with blur \( N(0, \sigma) \) with \( \sigma = 1, 2, 3, 4 \) and tested on images having the same level of blur.

- **Blind scenario:** two pipelines are trained on images with blur \( N(0, \sigma) \) with \( \sigma \) varying between 1-3 and 1-4. Testing performed on images with various levels of blur.

The average of peak signal-to-noise ratio (PSNR) in dB between the blurred LR input (degraded images) and the original HR images on Set5 and Set14 is used to evaluate the performance of all pipelines and results are reported in Tables 1 and 2. The default baseline comparison is with the bicubic interpolation. In the non-blind scenario, all AI pipelines improve the PSNR. The same behavior is observed for the structural similarity index measure (SSIM) as well, see Table 3 for “Set14”. The performance improvement becomes less pronounced in the blind scenarios.

Table 4 presents the average PSNR over the merged test sets. Here we test all pipelines on all possible levels of the input blurring. As expected non-blind pipelines perform best on images presenting the correct level of blurring (the one used for their training), see diagonal PSNR in red, and outperform blind pipelines trained to tackle a range of blurring levels. But when information about the blurring level is unavailable or a wrong non-blind model is used, the blind networks allow to achieve the best performance.

### 3.2. Evaluation of DBSRCNN
Tables 5, 6 and 7 report average PSNR for “Set5”, “Set14”, and SSIM for “Set14”, respectively. One observes a clear improvement of DBSRCNN’s performance over SRCNN on blurred images. DBSRCNN architecture allows to improve the quality of the images as measured with PSNR and SSIM. A possible explanation of this performance is that the concatenation of features extracted at an early stage acts similarly to traditional image processing techniques such as unsharp masking that boosts relevant (high) frequencies partially lost in the blurring stage, to enhance the reconstructed image.

Reconstruction examples with various levels of blur are shown in Figure 2 for qualitative comparison: DBSRCNN al-

---

**Table 1.** Average of PSNR (dB) for SRCNN on “Set5”

<table>
<thead>
<tr>
<th>Method</th>
<th>Training images</th>
<th>L.R</th>
<th>Blur (\sigma = 1)</th>
<th>Blur (\sigma = 2)</th>
<th>Blur (\sigma = 3)</th>
<th>Blur (\sigma = 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Upsampling (bicubic)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-trained SRCNN</td>
<td></td>
<td>31.95</td>
<td>30.55</td>
<td>30.29</td>
<td>29.61</td>
<td>27.35</td>
</tr>
<tr>
<td>non-blind</td>
<td></td>
<td>28.16</td>
<td>30.02</td>
<td>27.54</td>
<td>24.43</td>
<td></td>
</tr>
<tr>
<td>mixed (\sigma = 1)</td>
<td></td>
<td>27.64</td>
<td>29.05</td>
<td>29.55</td>
<td>27.80</td>
<td>25.91</td>
</tr>
<tr>
<td>mixed (\sigma = 4)</td>
<td></td>
<td>28.49</td>
<td>30.65</td>
<td>28.29</td>
<td>26.43</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2.** Average of PSNR (dB) for SRCNN on “Set14”

<table>
<thead>
<tr>
<th>Method</th>
<th>Training images</th>
<th>L.R</th>
<th>Blur (\sigma = 1)</th>
<th>Blur (\sigma = 2)</th>
<th>Blur (\sigma = 3)</th>
<th>Blur (\sigma = 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Upsampling (bicubic)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original SRCNN[1]</td>
<td></td>
<td>23.35</td>
<td>26.86</td>
<td>25.37</td>
<td>24.04</td>
<td>23.05</td>
</tr>
<tr>
<td>Re-trained SRCNN</td>
<td></td>
<td>24.47</td>
<td>31.66</td>
<td>28.40</td>
<td>25.51</td>
<td>23.19</td>
</tr>
<tr>
<td>non-blind</td>
<td></td>
<td>26.86</td>
<td>27.57</td>
<td>27.16</td>
<td>25.34</td>
<td>23.88</td>
</tr>
<tr>
<td>mixed (\sigma = 1)</td>
<td></td>
<td>25.85</td>
<td>26.92</td>
<td>26.93</td>
<td>25.52</td>
<td>24.20</td>
</tr>
<tr>
<td>mixed (\sigma = 4)</td>
<td></td>
<td>26.48</td>
<td>27.07</td>
<td>26.46</td>
<td>25.87</td>
<td>24.30</td>
</tr>
</tbody>
</table>

**Table 3.** Average of SSIM for SRCNN on “Set14”

<table>
<thead>
<tr>
<th>Method</th>
<th>Training images</th>
<th>L.R</th>
<th>Blur (\sigma = 1)</th>
<th>Blur (\sigma = 2)</th>
<th>Blur (\sigma = 3)</th>
<th>Blur (\sigma = 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Upsampling (bicubic)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original SRCNN[1]</td>
<td></td>
<td>0.8599</td>
<td>0.8325</td>
<td>0.7660</td>
<td>0.6957</td>
<td>0.6337</td>
</tr>
<tr>
<td>Re-trained SRCNN</td>
<td></td>
<td>0.8652</td>
<td>0.8815</td>
<td>0.8545</td>
<td>0.8047</td>
<td>0.7491</td>
</tr>
<tr>
<td>non-blind</td>
<td></td>
<td>0.8488</td>
<td>0.8707</td>
<td>0.8436</td>
<td>0.7588</td>
<td>0.6771</td>
</tr>
<tr>
<td>mixed (\sigma = 1)</td>
<td></td>
<td>0.8319</td>
<td>0.8561</td>
<td>0.8409</td>
<td>0.7717</td>
<td>0.6969</td>
</tr>
<tr>
<td>mixed (\sigma = 4)</td>
<td></td>
<td>0.8409</td>
<td>0.8707</td>
<td>0.8436</td>
<td>0.7588</td>
<td>0.6771</td>
</tr>
</tbody>
</table>

**Table 4.** Average of PSNR (dB) for SRCNN on “Set5+Set14”

<table>
<thead>
<tr>
<th>Method</th>
<th>Training images</th>
<th>L.R</th>
<th>Blur (\sigma = 1)</th>
<th>Blur (\sigma = 2)</th>
<th>Blur (\sigma = 3)</th>
<th>Blur (\sigma = 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Upsampling (bicubic)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-trained SRCNN</td>
<td></td>
<td>24.47</td>
<td>31.66</td>
<td>28.40</td>
<td>25.51</td>
<td>23.19</td>
</tr>
<tr>
<td>non-blind</td>
<td></td>
<td>26.86</td>
<td>27.57</td>
<td>27.16</td>
<td>25.34</td>
<td>23.88</td>
</tr>
<tr>
<td>mixed (\sigma = 1)</td>
<td></td>
<td>25.85</td>
<td>26.92</td>
<td>26.93</td>
<td>25.52</td>
<td>24.20</td>
</tr>
<tr>
<td>mixed (\sigma = 4)</td>
<td></td>
<td>26.48</td>
<td>27.07</td>
<td>26.46</td>
<td>25.87</td>
<td>24.30</td>
</tr>
</tbody>
</table>
**Fig. 2.** SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with different $\sigma$. Third and fifth column show non-blind scenario, and fourth and sixth correspond to blind scenario. Each result is accompanied by zoom and PSNR.
In this work we follow the third strategy and perform SR on a colour image along with relevant comparisons with SRCNN.

3.3. Extension to colour images

The majority of existing SR methods concentrate on single-band or grey-scale input imagery. There are several main approaches to perform super-resolution on colour images. The straight forward approach is to perform SR separately on the input colour channels and then merge them together into a colour image. Another approach consists in dealing with all three channels in a unified manner by expanding the sizes of layers in the deep architecture ($c = 3$). Finally, colour images can be super-resolved by casting the colour images to YCbCr colour space where the SR is performed solely on the luminance channel $Y$. Chroma components $Cb$ and $Cr$ are up-scaled by bicubic interpolation, and then all channels ($Y$, $Cb$, $Cr$) are combined again to produce output.

In this work we follow the third strategy and perform SR on $Y$ channel and bicubic interpolation on chroma components. Figure 3 demonstrates an example output of processing a colour input along with relevant comparisons with SRCNN.

4. CONCLUSION

In this work we have extensively evaluated performance of the recently proposed SRCNN architecture for recovering high resolution images from low resolution corrupted by blur / noise. We have proposed a new architecture DBSRCNN that enhances the reconstruction by boosting the relevant features that were originally lost in the SRCNN pipeline. Our experimental study with different levels of Gaussian blur demonstrates that our revised deeper architecture performs better in both blind and non-blind testing scenarios.
5. REFERENCES


