Deep Back-Projection Networks for Single Image Super-resolution

Muhammad Haris, Greg Shakhnarovich, Member, IEEE, and Norimichi Ukita, Member, IEEE

Abstract—Previous feed-forward architectures of recently proposed deep super-resolution networks learn the features of low-resolution inputs and the non-linear mapping from those to a high-resolution output. However, this approach does not fully address the mutual dependencies of low- and high-resolution images. We propose Deep Back-Projection Networks (DBPN), the winner of two image super-resolution challenges (NTIRE2018 and PIRM2018), that exploit iterative up- and down-sampling layers. These layers are formed as a unit providing an error feedback mechanism for projection errors. We construct mutually-connected up- and down-sampling units each of which represents different types of image degradation and high-resolution components. We also show that extending this idea to several variants applying the latest deep network trends, such as recurrent network, dense connection, and residual learning, to improve the performance. The experimental results yield superior results and in particular establishing new state-of-the-art results across multiple data sets, especially for large scaling factors such as $8\times$.

Index Terms—Image super-resolution, deep cnn, back-projection, deep concatenation, large scale, recurrent, residual

1 INTRODUCTION

Significant progress in deep learning for vision [1], [2], [3], [4], [5], [6], [7] has recently been propagating to the field of super-resolution (SR) [8], [9], [10], [11], [12], [13], [14], [15]. Single image SR (SISR) is an ill-posed inverse problem where the aim is to recover a high-resolution (HR) image from a low-resolution (LR) image. A currently typical approach is to construct an HR image by learning non-linear LR-to-HR mapping, implemented as a deep neural network [12], [13], [14], [16], [17], [18], [19]. These networks compute a sequence of feature maps from the LR image, culminating with one or more upsampling layers to increase resolution and finally construct the HR image.

In contrast to this purely feed-forward approach, the human visual system is believed to use a feedback connection to simply guide the task for the relevant results [20], [21], [22]. Perhaps hampered by lack of such feedback, the current SR networks with only feed-forward connections have difficulty in representing the LR-to-HR relation, especially for large scaling factors.

On the other hand, feedback connections were used effectively by one of the early SR algorithms, the iterative back-projection [23]. It iteratively computes the reconstruction error, then uses it to refine the HR image. Although it has been proven to improve the image quality, results still suffer from ringing and chessboard artifacts [24]. Moreover, this method is sensitive to choices of parameters such as the number of iterations and the blur operator, leading to variability in results.

Inspired by [23], we construct an end-to-end trainable architecture based on the idea of iterative up- and down-sampling layers: Deep Back-Projection Networks (DBPN). Our networks are not only able to remove the ringing and chessboard effect but also successfully perform large scaling factors, as shown in Fig. 1. Furthermore, DBPN has been proven by winning SISR challenges.

On NTIRE2018 [27], DBPN is the $1^{st}$ winner on track 8× Bicubic downsampling. On PIRM2018 [28], DBPN got $1^{st}$ on Region 2, $3^{rd}$ on Region 1, and $5^{th}$ on Region 3.

Our work provides the following contributions:

1) Iterative up- and down-sampling units. Feed-forward architectures, which are considered as a one-way mapping, only map rich representations of the input to the output space. This approach is unsuccessful to map LR and HR image, especially in large scaling factors, due to limited features available in the LR spaces. Our networks focus not only on generating variants of the HR features using the up-sampling unit but also on projecting it back to the LR spaces using the down-sampling unit. It is shown in Fig. 2(d), alternating between up- (blue box) and down-sampling (gold box) units, which represent the mutual relation of LR and HR features. This procedure can also be considered as features augmentation to represent various image degradation and HR components. The detailed explanation can be seen in Section 3.2.
Fig. 2. Comparisons of Deep Network SR. (a) Predefined upsampling (e.g., SRCNN [16], VDSR [12], DRRN [14]) commonly uses the conventional interpolation, such as Bicubic, to upscale LR input images before entering the network. (b) Single upsampling (e.g., FSRCNN [17], ESPCN [18]) propagates the LR features, then construct the SR image at the last step. (c) Progressive upsampling uses a Laplacian pyramid network to gradually predict SR images [13]. (d) Iterative up- and down-sampling approach is proposed by our DBPN that exploit the mutually connected up- (blue box) and down-sampling (gold box) units to obtain numerous HR feature maps in different depths.

(2) **Error feedback.** We propose an iterative error-correcting feedback mechanism for SR, which calculates both up- and down-projection errors to guide the reconstruction for obtaining better results. Here, the projection errors are used to refine the initial features in early layers. The detailed explanation can be seen in Section 3.3.

(3) **Deep concatenation.** Our networks represent different types of image degradation and HR components produced by each up- and down-sampling unit. This ability enables the networks to reconstruct the HR image using concatenation of the HR feature maps from all of the up-sampling units. Our reconstruction can directly utilize different types of HR feature maps from different depths without propagating them through the other layers as shown by the red arrows in Fig. 2(d).

(4) **Variants of DBPN.** We show that DBPN can be modified into several variants. We propose four improvements for DBPN: dense connected projection unit (Section 4.1), Recurrent DBPN (Section 4.2), Residual DBPN (Section 4.3), and Photo Realistic DBPN (Section 6).

## 2 Related Work

### 2.1 Image super-resolution using deep networks

Deep Networks SR can be primarily divided into four types as shown in Fig. 2.

(a) **Predefined upsampling** commonly uses interpolation as the upsampling operator to produce a middle resolution (MR) image. This scheme was firstly proposed by SRCNN [16] to learn MR-to-HR non-linear mapping with simple convolutional layers. Later, the improved networks exploited residual learning [12], [14] and recursive layers [19]. However, this approach has higher computation because the input is the MR image which has the same size as the HR image.

(b) **Single upsampling** offers a simple way to increase the resolution. This approach was firstly proposed by FSRCNN [17] and ESPCN [18]. These methods have been proven effective to increase the resolution and replace predefined operators. Further improvements include residual network [25], dense connection [29], and channel attention [15]. However, they fail to learn complicated mapping of LR-to-HR image, especially on large scaling factors, due to limited feature maps from the LR image. This problem opens the opportunities to propose the mutual relation from LR-to-HR image that can preserve HR components better.

(c) **Progressive upsampling** was recently proposed in LapSRN [13]. It progressively reconstructs the multiple SR images with different scales in one feed-forward network. For the sake of simplification, we can say that this network is a stacked of single upsampling networks which only relies on limited LR feature maps. Due to this fact, LapSRN is outperformed even by our shallow networks especially for large scaling factors such as 8× in experimental results.

(d) **Iterative up- and down-sampling** is proposed by our networks [26]. We focus on increasing the sampling rate of feature maps in different depths from iterative up- and down-sampling layers, then, distribute the tasks to calculate the reconstruction error on each unit. This scheme enables the networks to preserve the HR components by learning various up- and down-sampling operators while generating deeper features.

### 2.2 Feedback networks

Rather than learning a non-linear mapping of input-to-target space in one step, the feedback networks compose the prediction process into multiple steps which allow the model to have a self-correcting procedure. Feedback procedure has been implemented in various computing tasks [30], [31], [32], [33], [34], [35], [36].

In the context of human pose estimation, Carreira et al. [30] proposed an iterative error feedback by iteratively estimating and applying a correction to the current estimation. PredNet [36] is an unsupervised recurrent network to predictively code the future frames by recursively feeding the predictions back into the model. For image segmentation, Li et al. [33] learn implicit shape priors and use them to improve the prediction. However, to our knowledge, feedback procedures have not been implemented to SR.
2.3 Adversarial training

Adversarial training, such as with Generative Adversarial Networks (GANs) [37], has been applied to various image reconstruction problems [3, 6, 8, 38, 39]. For the SR task, Johnson et al. [8] introduced perceptual losses based on high-level features extracted from pre-trained networks. Ledig et al. [38] proposed SRGAN which is considered as a single upsampling method. It proposed the natural image manifold that is able to create photo-realistic images by specifically formulating a loss function based on the euclidian distance between feature maps extracted from VGG19 [40]. Our networks can be extended with the adversarial training. The detailed explanation is available in Section 6.

2.4 Back-projection

Back-projection [23] is an efficient iterative procedure to minimize the reconstruction error. Previous studies have proven the effectiveness of back-projection [41, 42, 43, 44]. Originally, back-projection in SR was designed for the case with multiple LR inputs. However, given only one LR input image, the reconstruction procedure can be obtained by upsampling the LR image using multiple upsampling operators and calculate the reconstruction error iteratively [24]. Timofte et al. [44] mentioned that back-projection could improve the quality of the SR images. Zhao et al. [21] proposed a method to refine high-frequency texture details with an iterative projection process. However, the initialization which leads to an optimal solution remains unknown. Most of the previous studies involve constant and unlearned predefined parameters such as blur operator and number of iteration.

To extend this algorithm, we develop an end-to-end trainable architecture which focuses to guide the SR task using mutually connected up- and down-sampling units to learn non-linear mutual relation of LR-to-HR image. The mutual relation between LR and HR image is constructed by creating iterative up- and down-projection unit where the up-projection unit generates HR feature maps, then the down-projection unit projects it back to the LR space as shown in Fig. 2 (d). This enables the networks to preserve the HR components by learned various up- and down-sampling operators and generates deeper features to construct numerous LR and HR feature maps.

3 Deep Back-projection Networks

Let \( H^i \) and \( \bar{I} \) be HR and LR image with \((M^h \times N^h)\) and \((M^l \times N^l)\), respectively, where \( M^l < M^h \) and \( N^l < N^h \). The main building block of our proposed DBPN architecture is the projection unit, which is trained (as part of the end-to-end training of the SR system) to map either an LR feature map to an HR map (up-projection), or an HR map to an LR map (down-projection).

3.1 Projection units

The up-projection unit is defined as follows:

- **scale up:** \( H^0_0 = (L^0_1 \ast p_t) \uparrow_s \) \( \quad \text{(1)} \)
- **scale down:** \( L^0_1 = (H^0_1 \ast g_t) \downarrow_s \) \( \quad \text{(2)} \)
- **residual:** \( e^0_l = L^0_1 - L^{-1} \) \( \quad \text{(3)} \)
- **scale residual up:** \( H^0_0 = (e^0_l \ast q_t) \uparrow_s \) \( \quad \text{(4)} \)
- **output feature map:** \( H^t = H^0_1 + H^0_0 \) \( \quad \text{(5)} \)

where \( \ast \) is the spatial convolution operator, \( \uparrow \) and \( \downarrow \) are, respectively, the up- and down-sampling operator with scaling factor \( s \), and \( p_t, g_t, q_t \) are (de)convolutional layers at stage \( t \).

The up-projection unit, illustrated in the upper part of Fig. 3, takes the previously computed LR feature map \( L^{-1} \) as input, and maps it to an (intermediate) HR map \( H^0_0 \); then it attempts to map it back to LR map \( L^0_1 \) ("back-project"). The residual (difference) \( e^0_l \) between the observed LR map \( L^{-1} \) and the reconstructed \( L^0_1 \) is mapped to HR again, producing a new intermediate (residual) map \( H^0_0 \); the final output of the unit, the HR map \( H^t \), is obtained by summing the two intermediate HR maps.

The down-projection unit, illustrated in the lower part of Fig. 3, is defined very similarly, but now its job is to map its input HR map \( H^t \) to the LR map \( L^t \).

- **scale down:** \( L^t = (H^t \ast g_t') \downarrow_s \) \( \quad \text{(6)} \)
- **scale up:** \( H^t = (L^t \ast p_t') \uparrow_s \) \( \quad \text{(7)} \)
- **residual:** \( e^t_l = H^t - H^t \) \( \quad \text{(8)} \)
- **scale residual down:** \( L^t = (e^t_l \ast q_t) \downarrow_s \) \( \quad \text{(9)} \)
- **output feature map:** \( L^t = L^0_0 + L^0_1 \) \( \quad \text{(10)} \)

We organize projection units in a series of stages, alternating between \( H \) and \( L \). These projection units can be understood as a self-correcting procedure which feeds a projection error to the sampling layer and iteratively changes the solution by feeding back the projection error.

The projection unit uses large sized filters such as \( 8 \times 8 \) and \( 12 \times 12 \). In the previous approaches, the use of large-sized filters is avoided because it can slow down the convergence speed and might produce sub-optimal results. However, the iterative up- and down-sampling units enable the mutual relation between LR and HR and take benefit of large receptive fields to perform better performance especially on large scaling factor where the significant amount of pixels is needed.

3.2 Network architecture

The proposed DBPN is illustrated in Fig. 4. It can be divided into three parts: initial feature extraction, projection, and reconstruction, as described below. Here, let \( \text{conv}(f, n) \) be a convolutional layer, where \( f \) is the filter size and \( n \) is the number of filters.

1. **Initial feature extraction.** We construct initial LR feature maps \( L^0 \in \mathbb{R}^{M^l \times N^l \times n_0} \) from the input using \( \text{conv}(3, n_0) \). Then \( \text{conv}(1, n_R) \) is used to reduce the dimension from \( n_0 \) to \( n_R \) before entering projection step where \( n_0 \) is the number of filters used in the initial LR features extraction and \( n_R \) is the number of filters used in each projection unit.
2) **Back-projection stages.** Following initial feature extraction is a sequence of projection units, alternating between construction of LR and HR feature maps ($L^t \in \mathbb{R}^{M^t \times N^t \times n_R}$ and $H^t \in \mathbb{R}^{M^h \times N^h \times n_R}$). Later, it further improves by dense connection where each unit has access to the outputs of all previous units (Section 4.1).

3) **Reconstruction.** Finally, the target HR image is reconstructed as $I^{sr} = f_{Rec}(H^1, H^2, ..., H^t)$, where $f_{Rec}$ use $\text{conv}(3, 3)$ as reconstruction and $[H^1, H^2, ..., H^t]$ refers to the concatenation of the feature-maps produced in each up-projection unit which called as deep concatenation.

Due to the definitions of these building blocks, our network architecture is modular. We can easily define and train networks with different numbers of stages, controlling the depth. For a network with $T$ stages, we have the initial extraction stage (2 layers), and then $T$ up-projection units and $T - 1$ down-projection units, each with 3 layers, followed by the reconstruction (one more layer). However, for the dense projection unit, we add $\text{conv}(1, n_R)$ in each projection unit, except the first three units as mentioned in Section 4.1.

### 4 The Variants of DBPN

In this section, we show how DBPN can be modified to apply the latest deep learning trends.

#### 4.1 Dense projection units

The dense inter-layer connectivity pattern in DenseNets [1] has been shown to alleviate the vanishing-gradient problem, produce improved features, and encourage feature reuse. Inspired by this we propose to improve DBPN, by introducing dense connections in the projection units called, yielding Dense DBPN.

Unlike the original DenseNets, we avoid dropout and batch norm, which are not suitable for SR, because they remove the range flexibility of the features [25]. Instead, we use $1 \times 1$ convolution layer as the bottleneck layer for feature pooling and dimensional reduction [11, 45] before entering the projection unit.

In Dense DBPN, the input for each unit is the concatenation of the outputs from all previous units. Let the $L^t$ and $H^t$ be the input for dense up- and down-projection unit, respectively. They are generated using $\text{conv}(1, n_R)$ which is used to merge all previous outputs from each unit as shown in Fig. 6. This improvement enables us to generate the feature maps effectively, as shown in the experimental results.

#### 4.2 Recurrent DBPN

Here, we propose recurrent DBPN which is able to reduce the number of parameters and widen the receptive field without increasing the model capacity. In SISR, DRCN [19] proposed recursive layers without introducing new parameters for additional convolutions in the networks. Then, DRRN [14] improves residual networks by introducing both global and local residual learning using a very deep CNN model (up to 52-layers). DBPN can also be treated as a recurrent network by sharing the projection units across the stages. We divided recurrent DBPN into two variants as mentioned below.

(a) **Single pair of projection unit (DBPN-R)** utilizes only one up-projection unit and one down-projection unit which is shared across all stages and does not utilize dense connection as shown in Fig. 6.

(b) **Multiple pairs of projection units (DBPN-MR)** utilizes multiple up- and down-projection units as shown in Fig. 7. However, instead of taking the output from each up-projection unit, DBPN-MR takes the HR features only from the last up-projection unit, then concatenate the HR features from each iteration. Here, the output from the last down-projection unit is the input for the next iteration. Then, the last up-projection unit will receive the output of all previous down-projection units on the corresponding iteration.

#### 4.3 Residual DBPN

Residual learning helps the network to converge faster and make the network have an easier job to produce only the difference between HR and interpolated LR image. Initially, residual learning
has been applied in SR by VDSR [12]. Residual DBPN takes LR image as an input to reduce the computational time. First, LR image is interpolated using Bicubic interpolation; then, at the last stage, the interpolated image is added to the reconstructed image to produce final SR image.

## 5 Experimental Results

### 5.1 Implementation and training details

In the proposed networks, the filter size in the projection unit is various with respect to the scaling factor. For $2 \times$, we use $6 \times 6$ kernel with stride = 2 and pad by 2 pixels. Then, $4 \times$ use $8 \times 8$ kernel with stride = 4 and pad by 2 pixels. Finally, the $8 \times$ use $12 \times 12$ kernel with stride = 8 and pad by 2 pixels.

We initialize the weights based on [46]. Here, standard deviation (std) is computed by $\sqrt{\frac{2}{n_f}}$ where $n_f = \frac{t}{t_{depth}}n_k$, $t_{depth}$ is the filter size, and $n_k$ is the number of filters. For example, with $t_{depth} = 3$ and $n_k = 8$, the std is 0.111. All convolutional and deconvolutional layers are followed by parametric rectified linear units (PReLU), except the final reconstruction layer.

We trained all networks using images from DIV2K [47] with augmentation (scaling, rotating, flipping, and random cropping). To produce LR images, we downscale the HR images on particular scaling factors using Bicubic. We use batch size of 16 with size $40 \times 40$ for LR image, while HR image size corresponds to the scaling factors. The learning rate is initialized to $1 \times 10^{-4}$ for all layers and decrease by a factor of 10 for every $5 \times 10^5$ iterations for total $10^6$ iterations. We used Adam with momentum to 0.9 and trained with L1 Loss. All experiments were conducted using PyTorch 0.3.1 and Python 3.5 on NVIDIA TITAN X GPUs. The code is available in the internet [48].

### 5.2 Model analysis

There are six types of DBPN used for model analysis: DBPN-SS, DBPN-S, DBPN-M, DBPN-L, D-DBPN-L, D-DBPN, and DBPN-M. The detailed architectures of those networks are shown in Table [49].

1. We found these settings to work well based on general intuition and preliminary experiments.
2. The implementation is available here.

---

**Depth analysis.** To demonstrate the capability of our projection unit, we construct multiple networks: DBPN-S ($T = 2$), DBPN-M ($T = 4$), and DBPN-L ($T = 6$). In the feature extraction, we use $n_0 = 128$ and $n_R = 32$. Then, we use conv(1, 1) for the reconstruction. The input and output image are luminance only.

The results on $4 \times$ enlargement are shown in Fig. 8. DBPN outperforms the state-of-the-art methods. Starting from our shallow network, DBPN-S gives the higher PSNR than VDSR, DRCN, and LapSRN. DBPN-S uses only 12 convolutional layers with smaller number of filters than VDSR, DRCN, and LapSRN. At the best performance, DBPN-S can achieve 31.59 dB which better $0.24 \text{ dB}$, $0.06 \text{ dB}$, $0.05 \text{ dB}$ than VDSR, DRCN, and LapSRN, respectively. DBPN-M shows performance improvement which better than all four existing state-of-the-art methods (VDSR, DRCN, LapSRN, and DRRN). At the best performance, DBPN-M can achieve 31.74 dB which better $0.39 \text{ dB}$, $0.21 \text{ dB}$, $0.20 \text{ dB}$, $0.06 \text{ dB}$ than VDSR, DRCN, LapSRN, and DRRN respectively. In total, DBPN-M uses 24 convolutional layers which has the same depth as LapSRN. Compare to DRRN (up to 52 convolutional layers), DBPN-M undeniable shows the effectiveness of our projection unit. Finally, DBPN-L outperforms all methods with $21 \text{ dB}$ which better than all four existing state-of-the-art methods (VDSR, DRCN, LapSRN, and DRRN). At the best performance, DBPN-L can achieve $31 \text{ dB}$ which better $0.51 \text{ dB}$, $0.33 \text{ dB}$, $0.32 \text{ dB}$, $0.18 \text{ dB}$ than VDSR, DRCN, LapSRN, and DRRN, respectively.

The results of $8 \times$ enlargement are shown in Fig. 9. Our networks outperform the current state-of-the-art for $8 \times$ enlargement which clearly show the effectiveness of our proposed networks on large scaling factors. However, we found that there is no significant performance gain from each proposed network especially for DBPN-L and DBPN-M networks where the difference only 0.04 dB.

### Number of parameters

We show the tradeoff between performance and number of network parameters from our networks and existing deep network SR in Fig. 10 and 11.

For the sake of low computation for real-time processing, we construct DBPN-SS which is the lighter version of DBPN-S ($T = 2$). We use $n_0 = 64$ and $n_R = 18$. However, the results outperform SRCNN, FSRCNN, and VDSR on both $4 \times$ and $8 \times$ enlargement. Moreover, DBPN-SS performs better than VDSR with $72\%$ and $37\%$ fewer parameters on $4 \times$ and $8 \times$ enlargement, respectively.

DBPN-S has about $27\%$ fewer parameters and higher PSNR than LapSRN on $4 \times$ enlargement. Finally, D-DBPN has about $76\%$ fewer parameters, and approximately the same PSNR, com-
TABLE 1
Model architecture of DBPN. "Feat0" and "Feat1" refer to first and second convolutional layer in the initial feature extraction stages. Note: \( \text{conv}(f, n, s, p) \) where \( f \) is filter size, \( n \) is number of filters, \( s \) is striding, and \( p \) is padding.

| Scale | DBPN-SS | DBPN-S | DBPN-M | DBPN-L | D-DBPN-L | D-DBPN | DBPN |
|-------|---------|--------|--------|--------|----------|--------|-------|
| Input/Output | Luminance | Luminance | Luminance | Luminance | Luminance | RGB | RGB |
| Feat0 | \( \text{conv}(3, 64, 1, 1) \) | \( \text{conv}(3, 128, 1, 1) \) | \( \text{conv}(3, 128, 1, 1) \) | \( \text{conv}(3, 128, 1, 1) \) | \( \text{conv}(3, 128, 1, 1) \) | \( \text{conv}(3, 256, 1, 1) \) | \( \text{conv}(3, 256, 1, 1) \) |
| Feat1 | \( \text{conv}(1, 18, 1, 0) \) | \( \text{conv}(1, 32, 1, 0) \) | \( \text{conv}(1, 32, 1, 0) \) | \( \text{conv}(1, 32, 1, 0) \) | \( \text{conv}(1, 32, 1, 0) \) | \( \text{conv}(1, 64, 1, 0) \) | \( \text{conv}(1, 64, 1, 0) \) |
| Reconstruction | \( \text{conv}(1, 1, 0) \) | \( \text{conv}(1, 1, 0) \) | \( \text{conv}(1, 1, 0) \) | \( \text{conv}(1, 1, 0) \) | \( \text{conv}(1, 1, 0) \) | \( \text{conv}(1, 1, 0) \) | \( \text{conv}(1, 1, 0) \) |
| BP stages | 2× | \( \text{conv}(6, 18, 2, 2) \) | \( \text{conv}(6, 32, 2, 2) \) | \( \text{conv}(6, 32, 2, 2) \) | \( \text{conv}(6, 32, 2, 2) \) | \( \text{conv}(6, 64, 2, 2) \) | \( \text{conv}(6, 64, 2, 2) \) |
| Parameters (k) | 2× | 106 | 337 | 779 | 1221 | 1230 | 5819 | 8811 |
| Depth | 12 | 12 | 24 | 36 | 40 | 52 | 76 |
| No. of stage (T) | 2 | 2 | 4 | 6 | 6 | 10 |
| Dense connection | No | No | No | No | Yes | Yes | Yes |

Fig. 9. The depth analysis of DBPN on Set5 dataset for 8× enlargement. DBPN-S (T = 2), DBPN-M (T = 4), and DBPN-L (T = 6)pared to EDSR on 4× enlargement. On the 8× enlargement, D-DBPN has about 47% fewer parameters with better PSNR compare to EDSR. This evidence show that our networks has the best trade-off between performance and number of parameter.

Deep concatenation. Each projection unit is used to distribute the reconstruction step by constructing features which represent different details of the HR components. Deep concatenation is also well-related with the number of \( T \) (back-projection stage), which shows more detailed features generated from the projection units will also increase the quality of the results. In Fig. 12, it is shown that each stage successfully generates diverse features to reconstruct SR image.

Error Feedback. As stated before, error feedback (EF) is used to guide the reconstruction in the early layer. Here, we analyze how error feedback can help for better reconstruction. We conduct experiments to see the effectiveness of error feedback procedure. On the scenario without EF, we replace up- and down-projection unit with single up- (deconvolution) and down-sampling (convolution) layer.

We show PSNR of DBPN-S with EF and without EF in Table 2. The result with EF has 0.53 dB and 0.26 dB better than without EF on Set5 and Set14, respectively. In Fig. 13 we visually show how error feedback can construct better and sharper HR image especially in the white stripe pattern of the wing.

Moreover, the performance of DBPN-S without EF is interestingly 0.57 dB and 0.35 dB better than previous approaches such as SRCNN [16] and FSRCNN [17], respectively, on Set5. The results show the effectiveness of iterative up- and downsampling layers to demonstrate the LR-to-HR mutual dependency.

Filter Size. We analyze the size of filters which is used in the back-projection stage. As stated before, the choice of filter size in the back-projection stage is based on the preliminary results. For the 4× enlargement, we show that filter 8×8 is 0.08 dB and 0.09 dB better than filter 6×6 and 10×10, respectively, as shown in Table 3.
Table 2: Analysis of EF using DBPN-S on $4 \times$ enlargement. Red indicates the best performance.

| Algorithm | Scale | Set5 | Set14 |
|-----------|-------|------|-------|
| SRCNN     | 16    | 30.49| 27.61 |
| FSRCNN    | 17    | 30.71| 27.70 |
| Without EF|       | 31.06| 27.95 |
| With EF   |       | 31.59| 28.21 |

Table 3: Analysis of filter size in the back-projection stages on $4 \times$ enlargement from D-DBPN. Red indicates the best performance.

| Filter size | Striding | Padding | Set5 | Set14 |
|-------------|----------|---------|------|-------|
| 6           | 4        | 1       | 32.39| 28.78 |
| 8           | 4        | 2       | 32.47| 28.82 |
| 10          | 4        | 3       | 32.38| 28.79 |

Table 4: Analysis of input/output color channel using DBPN-L. Red indicates the best performance.

| Channel    | Set5 | Set14 |
|------------|------|-------|
| RGB        | 31.88| 28.47 |
| Luminance  | 31.86| 28.47 |

Luminance vs RGB In D-DBPN, we change the input/output from luminance to RGB color channels. There is no significant improvement in the quality of the result as shown in Table 4. However, for running time efficiency, constructing all channels simultaneously is faster than a separated process.

5.3 Comparison of each DBPN variant

Dense connection. We implement D-DBPN-L which is a dense connection of the $L$ network to show how dense connection can improve the network’s performance in all cases as shown in Table 5. On $4 \times$ enlargement, the dense network, D-DBPN-L, gains 0.13 dB and 0.05 dB higher than DBPN-L on the Set5 and Set14, respectively. On $8 \times$, the gaps are even larger. The D-DBPN-L has 0.23 dB and 0.19 dB higher than DBPN-L on the Set5 and Set14, respectively.

Comparison across the variants. We compare six DBPN variants: DBPN-R64-10, DBPN-R128-5, DBPN-MR64-3, DBPN-RES-MR64-3, DBPN-RES, and DBPN. First, DBPN, which was the winner of NTIRE2018 [27], and PIRM2018 [28], uses $n_0 = 256$, $n_R = 64$, and $t = 10$ for the back-projection stages, and dense connection between projection units. In the reconstruction, we use $\text{conv}(3, 3)$. DBPN-R64-10 uses $n_R = 64$ with 10 iterations to produce 640 HR features as input of reconstruction layer. DBPN-R128-5, uses $n_R = 128$ with 5 iterations, produces 640 HR features. DBPN-MR64-3 has the same architecture with D-DBPN but the projection units are treated as recurrent network. DBPN-RES-MR64-3 is DBPN-MR64-3 with residual learning. Last, DBPN-RES is DBPN with residual learning. All variants are trained with the same training setup.

The results are shown in Table 5. It shows that all variants successfully have better performance than D-DBPN [26]. DBPN-R64-10 has the least parameter compare to other variants, which is suitable for mobile/real-time application. It can reduce $10 \times$ number of parameter compare to DBPN and maintain to get good performance. We can see that increasing $n_R$ can improve the performance of DBPN-R which is shown by DBPN-R128-5 compare to DBPN-R64-10. However, better results is obtained by DBPN-MR64-3, especially on Urban100 and Manga109 test set compare to other variants. It is also proven that residual learning can slightly improve the performance of DBPN. Therefore, it is natural that we performed the combination of multiple stages recurrent and residual learning called DBPN-RES-MR64-3 which performs the best results and has lower parameter than DBPN.

5.4 Comparison with the-state-of-the-arts on SR

To confirm the ability of the proposed network, we performed several experiments and analysis. We compare our network with ten state-of-the-art SR algorithms: A+ [48], SRCNN [16], FSRCNN [17], VDSR [12], DRGCN [19], DRRN [14], LapSRN [13], D-DBPN [26], EDSR [25], and RCAN [15]. We carry out extensive experiments using 5 datasets: Set5 [49], Set14 [50], BSDS100 [51], Urban100 [52], and Manga109 [53]. Each dataset has different characteristics. Set5, Set14 and BSDS100 consist of natural scenes; Urban100 contains urban scenes with details in different frequency bands; and Manga109 is a dataset of Japanese manga.

Our final network, DBPN-RES-MR64-3, combines dense connection, recurrent network and residual learning to boost the performance of DBPN. It uses $n_0 = 256$, $n_R = 64$, and $t = 7$ with 3 iteration. In the reconstruction, we use $\text{conv}(3, 3)$. RGB
color channels are used for input and output image. It takes around than 14 days to train.

PSNR and structural similarity (SSIM) \(^{54}\) were used to quantitatively evaluate the proposed method. Note that higher PSNR and SSIM values indicate better quality. As used by existing networks, all measurements used only the luminance channel (Y). For SR by factor \(s\), we crop \(s\) pixels near image boundary before evaluation as in \(^{17}, 25\). Some of the existing networks such as SRCNN, FSRCNN, VDSR, and EDSR did not perform \(8 \times \) enlargement. To this end, we retrained the existing networks by using author’s code with the recommended parameters.

Figure \(^{14}\) shows that EDSR tends to generate stronger edge than the ground truth and lead to misleading information in several cases. The result of EDSR shows the eyelashes were interpreted as a stripe pattern. Our result generates softer patterns which is subjectively closer to the ground truth. On the butterfly image, EDSR separates the white pattern and tends to construct regular pattern such as circle and stripe, while D-DBPN constructs the same pattern as the ground truth.

We show the quantitative results in the Table \(^{7}\) Our network outperforms the existing methods by a large margin in all scales except RCAN. For \(4 \times\), EDSR has 0.26 dB higher than D-DBPN but outperformed by DBPN-RES-MR64-3 with 0.44 dB margin on Urban100. Recent state-of-the-art, RCAN \(^{15}\), performs better results than our network on \(2 \times\). However, on \(4 \times\), our network has 0.26 dB higher than RCAN on Urban100. The biggest gap is shown on Manga109, our network has 0.52 dB higher than RCAN.

Our network shows its effectiveness on \(8 \times \) enlargement which outperforms all of the existing methods by a large margin. Inter-
Fig. 15. Qualitative comparison of our models with other works on $8 \times$ super-resolution.
TABLE 7
Quantitative evaluation of state-of-the-art SR algorithms: average PSNR/SSIM for scale factors 2×, 4×, and 8×. Red indicates the best and blue indicates the second best performance.

| Method              | Scale | Set5 PSNR | SSIM | Set14 PSNR | SSIM | BSDS100 PSNR | SSIM | Urban100 PSNR | SSIM | Manga109 PSNR | SSIM |
|---------------------|-------|-----------|------|------------|------|--------------|------|--------------|------|--------------|------|
| bicubic             | 2     | 29.56     | 0.884 | 26.88      | 0.844| 30.84        | 0.935| 35.33        | 0.967|
| A+                  | 2     | 29.56     | 0.884 | 26.88      | 0.844| 30.84        | 0.935| 35.33        | 0.967|
| SRCNN               | 4     | 28.34     | 0.863 | 26.06      | 0.829| 30.39        | 0.941| 37.16        | 0.974|
| FSRCNN              | 8     | 27.39     | 0.847 | 25.76      | 0.810| 32.52        | 0.920| 41.52        | 0.982|
| VDSR                | 12    | 27.39     | 0.847 | 25.76      | 0.810| 32.52        | 0.920| 41.52        | 0.982|
| DRCN                | 16    | 27.39     | 0.847 | 25.76      | 0.810| 32.52        | 0.920| 41.52        | 0.982|
| DRRN                | 20    | 27.39     | 0.847 | 25.76      | 0.810| 32.52        | 0.920| 41.52        | 0.982|
| LapSRN              | 24    | 27.39     | 0.847 | 25.76      | 0.810| 32.52        | 0.920| 41.52        | 0.982|
| D-DBPN              | 28    | 27.39     | 0.847 | 25.76      | 0.810| 32.52        | 0.920| 41.52        | 0.982|

TABLE 8
Runtime evaluation with input size 64×64. Red indicates the best and blue indicates the second best performance. * indicates the calculation using function timer in Torch, and N.A. indicates that the algorithm runs out of GPU memory.

| Method              | Scale | 2× (128×128) | 4× (256×256) | 8× (512×512) |
|---------------------|-------|--------------|--------------|--------------|
| VDSR [12]           |       | 0.02225      | 0.03225      | 0.06856      |
| DRRN [14]           |       | 0.25413      | 0.32893      | N.A.         |
| *EDSR [25]          |       | 0.8379       | 1.2458       | 1.1477       |
| DBPN-SS             |       | -            | 0.01672      | 0.02692      |
| DBPN-S              |       | -            | 0.02073      | 0.03812      |
| DBPN-M              |       | -            | 0.05411      | 0.08106      |
| DBPN-L              |       | -            | 0.06971      | 0.12635      |
| D-DBPN              |       | 0.15331      | 0.19356      | 0.31851      |

6 Perceptually optimized DBPN
We also can extend DBPN to produce HR outputs that appear to be better under human perception. Despite many attempts, it remains unclear how to accurately model perceptual quality. Instead, we incorporate the perceptual quality into the generator by using adversarial loss, as introduced elsewhere [38], [39]. In the adversarial settings, there are two building blocks: a generator (G) and a discriminator (D). In the context of SR, the generator G produces HR images (from LR inputs). The discriminator D works to differentiate between real HR images and generated HR images (the product of SR network G). In our experiments, the generator is a DBPN network, and the discriminator is a network with five hidden layers with batch norm, followed by the last, fully connected layer.

The generator loss in this experiment is composed of four loss terms, following [39]: MSE, VGG, Style, and Adversarial loss.

\[
L_G = w_1 \cdot L_{mse} + w_2 \cdot L_{vgg} + w_3 \cdot L_{adv} + w_4 \cdot L_{style}
\]

- MSE loss is pixel-wise loss which calculated in the image space \( L_{mse} = ||I^h - I^sr||_2^2 \).
• VGG loss is calculated in the feature space using pretrained VGG19 network \[^{40}\] on multiple layers. This loss was originally proposed by \[^{55}\]. Both \(I^h\) and \(I^{sr}\) are first mapped into a feature space by differentiable functions \(f_i\) from VGG multiple max-pool layers \((i = 2, 3, 4, 5)\) then sum up each layer distances. \(L_{vgg} = \sum_{i=2}^{5} ||f_i(I^h) - f_i(I^{sr})||^2\).

• Adversarial loss. \(L_{adv} = -\log(D(G(I^h)))\), where \(D(x)\) is the probability assigned by \(D\) to \(x\) being a real HR image.

• Style loss is used to generate high quality textures. This loss was originally proposed by \[^{56}\]. Style loss using the same differentiable function \(f\) as in VGG loss. \(L_{style} = \sum_{i=2}^{5} ||\theta(f_i(I^h)) - \theta(f_i(I^{sr}))||^2\) where Gram matrix \(\theta(F) = F F^T \in \mathbb{R}^{n \times n}\).

The training objective for \(D\) is

\[
L_D = -\log(D(I^h)) - \log(1 - D(G(I^f))).
\]

As is common in training adversarial networks, we alternate between stages of training \(G\) and training \(D\). We use pre-trained DBPN model which optimized by MSE loss only, then fine-tuned with the perceptual loss. We use batch size of 4 with size 60 \(\times\) 60 for LR image, while HR image size is 240 \(\times\) 240. The learning rate is initialized to \(1e{-4}\) for all layers for \(2 \times 10^5\) iteration using Adam with momentum to 0.9.

This method was included in the challenge associated with PIRM2018 \[^{28}\], in conjunction with ECCV 2018. In the challenge, evaluation was conducted in three disjoint regimes defined by thresholds on the RMSE; the intuition behind this is the natural tradeoff between RMSE and perceptual quality of the reconstruction. The latter is measured by combining the quality measures of Ma \[^{57}\] and NIQE \[^{58}\] as below,

\[
\text{Perceptual index} = 1/2((10 - Ma) + NIQE).
\] (12)

The three regimes correspond to Region 1: RMSE \(\leq 11.5\), Region 2: \(11.5 < \text{RMSE} \leq 12.5\), and Region 3: \(12.5 < \text{RMSE} \leq 16\). We select optimal parameter settings for each regime. This process yields

• Region 1 \((w_1 : 0.5, w_2 : 0.05, w_3 : 0.001, w_4 : 1)\)

• Region 2 \((w_1 : 0.1, w_2 : 0.2, w_3 : 0.001, w_4 : 1)\)

• Region 3 \((w_1 : 0.03, w_2 : 0.2, w_3 : 0.001, w_4 : 10)\)

Our method achieved 1\(^{st}\) place on Region 2, 3\(^{rd}\) place on Region 1, and 5\(^{th}\) place on Region 3 \[^{28}\]. In Region 3, it shows very competitive results where we got 5\(^{th}\), however, it is noted that our method has the lowest RMSE among other top 5 performers which means the image has less distortion or hallucination w.r.t the original image.

We show qualitative results from our method which is shown in Fig. \[^{16}\]. It can be seen that there are significant improvement on high quality texture on each region compare to MSE-optimized SR image.

7 CONCLUSION

We have proposed Deep Back-Projection Networks for Single Image Super-resolution which is the winner of two single image SR challenge (NTIRE2018 and PIRM2018). Unlike the previous methods which predict the SR image in a feed-forward manner, our proposed networks focus to directly increase the SR features using multiple up- and down-sampling stages and feed the error predictions on each depth in the networks to revise the sampling results, then, accumulates the self-correcting features from each upsampling stage to create SR image. We use error feedbacks from the up- and down-scaling steps to guide the network to achieve a better result. The results show the effectiveness of the proposed network compares to other state-of-the-art methods. Moreover, our proposed network successfully outperforms other state-of-the-art methods on large scaling factors such as 8\(\times\) enlargement. We also show that DBPN can be modified into several variants to follow the latest deep learning trends to improve its performance.

ACKNOWLEDGMENTS

This work was partly supported by FCRAL, and by AFOSR Center of Excellence in Efficient and Robust Machine Learning, Award FA9550-18-1-0166.

REFERENCES

[1] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv preprint arXiv:1512.03385, 2015.

[3] E. L. Denton, S. Chintala, R. Fergus et al., “Deep generative image models using a laplacian pyramid of adversarial networks,” in Advances in neural information processing systems, 2015, pp. 1486–1494.

[4] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, “Learning from simulated and unsupervised images through adversarial training,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[5] G. Larsson, M. Maire, and G. Shakhnarovich, “FracTalNet: Ultra-deep neural networks without residuals,” arXiv preprint arXiv:1605.07648, 2016.

[6] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv preprint arXiv:1511.06434, 2015.

[7] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul 2017.

[8] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in European Conference on Computer Vision. Springer, 2016, pp. 694–711.

[9] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia, “Detail-revealing deep video super-resolution,” in Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 2017, pp. 22–29.

[10] M. S. Sajjadi, R. Vemulapalli, and M. Brown, “Frame-recurrent video super-resolution,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6626–6634.

[11] M. Haris, M. R. Widyanto, and H. Nobuhara, “Inception learning super-resolution,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[12] J. Kim, J. Kwon Lee, and K. Mu Lee, “Accurate image super-resolution using very deep convolutional networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2016, pp. 1646–1654.

[13] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Deep laplacian pyramid networks for fast and accurate super-resolution,” in IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[14] Y. Tai, J. Yang, and X. Liu, “Image super-resolution via deep recursive residual network,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[15] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 286–301.

[16] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 2, pp. 295–307, 2016.

[17] C. Dong, C. C. Loy, and X. Tang, “Accelerating the super-resolution convolutional neural network,” in European Conference on Computer Vision. Springer, 2016, pp. 391–407.
Fig. 16. Results of DBPN with perceptual loss.

[18] W. Shi, J. Caballero, F. Huszár, 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,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874–1883.

[19] J. Kim, J. Kwon Lee, and K. Mu Lee, “Deeply-recursive convolutional network for image super-resolution,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1637–1645.

[20] D. J. Felleman and D. C. Van Essen, “Distributed hierarchical processing in the primate cerebral cortex.” Cerebral cortex (New York, NY: 1991), vol. 1, no. 1, pp. 1–47, 1991.

[21] D. J. Kravitz, K. S. Saleem, C. I. Baker, L. G. Ungerleider, and M. Mishkin, “The ventral visual pathway: an expanded neural framework for the processing of object quality,” Trends in cognitive sciences, vol. 17, no. 1, pp. 26–49, 2013.

[22] V. A. Lamme and P. R. Roelfsema, “The distinct modes of vision offered by feedforward and recurrent processing,” Trends in neurosciences, vol. 23, no. 11, pp. 571–579, 2000.

[23] M. Irani and S. Peleg, “Improving resolution by image registration,” CVGIP: Graphical models and image processing, vol. 53, no. 3, pp. 231–239, 1991.

[24] S. Dai, M. Han, Y. Wu, and Y. Gong, “Bilateral back-projection for single image super-resolution,” in Multimedia and Expo, 2007 IEEE International Conference on. IEEE, 2007, pp. 1039–1042.

[25] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced deep residual networks for single image super-resolution,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, July 2017.

[26] M. Haris, G. Shakhnarovich, and N. Ukiti, “Deep back-projection networks for super-resolution,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[27] M. Haris, G. Shakhnarovich, and N. Ukiti, “Enhanced deep residual networks for single image super-resolution,” in The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018 IEEE Conference on, 2018.

[28] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, “2018 pirm challenge on perceptual image super-resolution,” arXiv preprint arXiv:1809.07517, 2018.

[29] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual dense network for image super-resolution,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[30] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, “Human pose estimation with iterative error feedback,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4733–4742.

[31] S. Ross, D. Munoz, M. Hebert, and J. A. Bagnell, “Learning message-passing inference machines for structured prediction,” in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011, pp. 2737–2744.

[32] Z. Tu and X. Bai, “Auto-context and its application to high-level vision tasks and 3d brain image segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 10, pp. 1744–1757, 2010.

[33] K. Li, B. Hariharan, and J. Malik, “Iterative instance segmentation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3659–3667.

[34] A. R. Zamir, T.-L. Wu, L. Sun, W. Shen, J. Malik, and S. Savarese, “Feedback networks,” arXiv preprint arXiv:1612.09508, 2016.

[35] A. Shrivastava and A. Gupta, “Contextual priming and feedback for faster r-cnn,” in European Conference on Computer Vision. Springer, 2016, pp. 330–348.

[36] W. Lotter, G. Kreiman, and D. Cox, “Deep predictive coding networks for video prediction and unsupervised learning,” arXiv preprint arXiv:1605.08104, 2016.

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, 2014, pp. 2672–2680.

[38] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang et al., “Photo-realistic single image super-resolution using a generative adversarial network,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul 2017.

[39] M. S. Sajjadi, B. Schölkopf, and M. Hirsch, “Enhancenet: Single image super-resolution through automated texture synthesis,” arXiv preprint arXiv:1612.07919, 2016.
K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *ICLR*, 2015.

Y. Zhao, R.-G. Wang, W. Jia, W.-M. Wang, and W. Gao, “Iterative projection reconstruction for fast and efficient image upscaling,” *Neurocomputing*, vol. 226, pp. 200–211, 2017.

M. Haris, M. R. Widyananto, and H. Nobuhara, “First-order derivative-based super-resolution,” *Signal, Image and Video Processing*, vol. 11, no. 1, pp. 1–8, 2017.

W. Dong, L. Zhang, G. Shi, and X. Wu, “Nonlocal back-projection for adaptive image enlargement,” in *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE, 2009, pp. 349–352.

R. Timofte, R. Rothe, and L. Van Gool, “Seven ways to improve example-based single image super resolution,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1865–1873.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

E. Agustsson and R. 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.

R. Timofte, V. De Smet, and L. Van Gool, “A+: Adjusted anchored neighborhood regression for fast super-resolution,” in *Asian Conference on Computer Vision*. Springer, 2014, pp. 111–126.

M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, “Low-complexity single-image super-resolution based on nonnegative neighbor embedding,” in *British Machine Vision Conference (BMVC)*, 2012.

R. Zeyde, M. Elad, and M. Protter, “On single image scale-up using sparse-representations,” in *Curves and Surfaces*. Springer, 2012, pp. 711–730.

P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

J.-B. Huang, A. Singh, and N. Ahuja, “Single image super-resolution from transformed self-exemplars,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5197–5206.

Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa, “Sketch-based manga retrieval using manga109 dataset,” *Multimedia Tools and Applications*, pp. 1–28, 2016.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, 2004.

A. Dosovitskiy and T. Brox, “Generating images with perceptual similarity metrics based on deep networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 656–664.

L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

C. Ma, C.-Y. Yang, X. Yang, and M.-H. Yang, “Learning a no-reference quality metric for single-image super-resolution,” *Computer Vision and Image Understanding*, vol. 158, pp. 1–16, 2017.

A. Mittal, R. Soundararajan, and A. C. Bovik, “Making a” completely blind” image quality analyzer,” *IEEE Signal Process. Lett.*, vol. 20, no. 3, pp. 209–212, 2013.

---

**Muhammad Haris** Muhammad Haris received S. Kom (Bachelor of Computer Science) from the Faculty of Computer Science, University of Indonesia, Depok, Indonesia, in 2009. Then, he received the M. Eng and Dr. Eng degree from Department of Intelligent Interaction Technologies, University of Tsukuba, Japan, in 2014 and 2017, respectively, under the supervision of Dr. Hajime Nobuhara. Currently, he is working as postdoctoral fellow in Intelligent Information Media Laboratory, Toyota Technological Institute with Prof. Norimichi Ukita. His main research interests are low-level vision and image/video processing.

---

**Greg Shakhnarovich** Greg Shakhnarovich has been faculty member at TTI-Chicago since 2008. He received his BSc degree in Computer Science and Mathematics from the Hebrew University in Jerusalem, Israel, in 1994, and a MSc degree in Computer Science from the Technion, Israel, in 2000. Prior to joining TTI Greg was a Postdoctoral Research Associate at Brown University, collaborating with researchers at the Computer Science Department and the Brain Sciences program there. Greg’s research interests lie broadly in computer vision and machine learning.

---

**Norimichi Ukita** Norimichi Ukita is a professor at the graduate school of engineering, Toyota Technological Institute, Japan (TTI-J). He received the B.E. and M.E. degrees in information engineering from Okayama University, Japan, in 1996 and 1998, respectively, and the Ph.D degree in Informatics from Kyoto University, Japan, in 2001. After working for five years as an assistant professor at NAIIST, he became an associate professor in 2007 and moved to TTI-J in 2016. He was a research scientist of Precursory Research for Embryonic Science and Technology Japan Science and Technology Agency (JST), during 2002 - 2006. He was a visiting research scientist at Carnegie Mellon University during 2007-2009. He currently works also at the Cybermedia center of Osaka University as a guest professor. His main research interests are object detection/tracking and human pose/shape estimation. He is a member of the IEEE.