Back–Projection Pipeline

Pablo Navarrete Michelini, 1 Hanwen Liu, 1 Yunhua Lu, 1 Xingqun Jiang, 1
1 BOE Technology Co., Ltd.

Abstract
We propose a simple extension of residual networks that works simultaneously in multiple resolutions. Our network design is inspired by the iterative back–projection algorithm but seeks the more difficult task of learning how to enhance images. Compared to similar approaches, we propose a novel solution to make back–projections run in multiple resolutions by using a data pipeline workflow. Features are updated at multiple scales in each layer of the network. The update dynamic through these layers includes interactions between different resolutions in a way that is causal in scale, and it is represented by a system of ODEs, as opposed to a single ODE in the case of ResNets. The system can be used as a generic multi–resolution approach to enhance images. We test it on several challenging tasks with special focus on super–resolution and raindrop removal. Our results are competitive with state–of–the–arts and show a strong ability of our system to learn both global and local image features.

Introduction
Image enhancement is the process of taking an impaired image as input and return an image of better quality. The current trend to achieve this target is to learn a mapping between impaired and enhanced images using example data. Deep–learning is leading this fast–growing quest in a number of applications, including: denoise(Lefkimmiatis 2018), deblur(Tao et al. 2018), super–resolution(Timofte et al. 2018), demosaicking(Kokkinos and Lefkimmiatis 2018), compression removal(Lu et al. 2018), dehaze(Ancuti et al. 2018b), derain(Wang et al. 2019), raindrop removal(Qian et al. 2018a), HDR(Wu et al. 2018), and colorization(He et al. 2018). Progress in network architectures often succeeds in image enhancement, as seen for example in image super–resolution, with CNNs applied in SRCNN (Dong et al. 2014), ResNets (He et al. 2016) applied in EDSR (Lim et al. 2017), DenseNets (Huang et al. 2017) applied in RDN (Zhang et al. 2018d), attention (Hu, Shen, and Sun 2018) applied in RCAN (Zhang et al. 2018a), and non–local attention (Wang et al. 2018) applied in RNAN (Zhang et al. 2019). In all these examples, arguably the most influential practice is the use of residual networks (ResNets). Here, we define the network state as the internal representation of an image in a network, commonly referred to as latent or feature space in the literature. The idea of ResNets is to represent an impaired image as a network state and progressively change it by adding residuals, as seen in Figure 1. This gives a compositional hierarchy(Poggio et al. 2017) of progressive local processing steps (e.g. convolutional layers) that transforms the input image. The update strategy of residual networks can be seen as a dynamical system where depth represents time and a differential equation models the evolution of the state(Liao and Poggio 2016).

Our proposed system, a Back–Projection Pipeline (BPP), works as a residual network that carries many (instead of one) resolution states at a given time step as seen in Figure 1. Although similar in spirit to U–Nets (Ronneberger, Fischer, and Brox 2015), this multi–resolution state is fundamentally different. U–Nets hold high resolution states to re–enter the network in later stages, whereas in BPP the state is created as initial conditions in multiple resolutions and get updated synchronously through the network. Another distinctive property of BPPs is scale causality. Namely, after initialization, low resolution states do not depend on higher resolution states. Information travels forward in depth, same as in ResNets, and upwards in scale, as shown in Figure 1. Scale causality is inspired by scale–space (Lindeberg 1994) and multi–resolution analysis(Mallat 1998) to express the nested nature of details. A simple example is that when we see an image of a keyboard we expect to see letters, but not necessarily the other way around. Finally, the interpretation...
of BPPs as an extension of ResNets becomes more clear from the dynamic of the network. We will show that BPP updates can be modeled by a non-autonomous system of differential equations, as opposed to a single ODE for ResNets.

**Related Work.** With regard to applications, BPP gives us a generic multi-resolution approach to transform images into a desired target. Current benchmarks in image enhancement often use different architectures for different tasks. It is important to distinguish between local and global targets. In the problem of super-resolution, for example, we need to calculate pixel values around a local area, and distant pixels become less relevant. In a different problem, contrast enhancement, we want to change the histogram of an image, which contains statistics that represent global features. General image enhancement is gaining interest in research and has been considered in the context of:

- **Mixed Local Problems:** In (Zhang et al. 2019), for example, authors solve denoising, super-resolution and deblur tasks using a single architecture and different parameters for each problem. In (Gharbi et al. 2016; Ehret et al. 2019) authors solve joint demosaicking and denoising, and in (Qian et al. 2019) authors additionally solve super-resolution, all through using a single architecture and same model parameters. In (Zhang, Zu, and Zhang 2018) authors tackle super-resolution and deblur, and train a single system to handle different image degradations.

- **Global and Local Problems:** Authors in (Soh, Park, and Cho 2019; Kim, Oh, and Kim 2019; Kinoshita and Kiya 2019) consider the joint solution of low-to-high dynamic range enhancement as well as image-SR. In (Kim, Oh, and Kim 2019) authors generate an image in HDR display format, whereas (Soh, Park, and Cho 2019; Kinoshita and Kiya 2019) use the same input and output format. They both use U-Net configurations, while (Soh, Park, and Cho 2019) uses a two-stage Retinex decomposition network.

Regarding architecture, BPP uses a multi-resolution workflow, which is different from U-Nets (Ronneberger, Fischer, and Brox 2015). This workflow follows from the Iterative Back-Projection (Irani and Peleg 1991) (IBP) algorithm. In this respect, Multi-Grid Back-Projection (Navarrete Michelini, Liu, and Zhu 2019) (MGBP) is the closest super-resolution system that is state-of-the-arts for lightweight systems with small number of parameters. It is based on a multi-resolution back-projection algorithm that uses a multigrid recursion (Trottenberg and Schiller 2001). This recursion violates scale causality as it sends network states back to low-resolution to restart iterations. We also notice that BPP follows the wide-activation design in (Yu et al. 2018) in the sense that features are increase before activations and reduced before updating. BPP shows a workflow structure similar to the Multi-scale DenseNet architecture in (Huang et al. 2018), except in the latter scale causality moves downwards in scale, it does not use back-projections and it focuses on a label prediction problem. The WaveNet architecture (Oord et al. 2016) also shares the property of scale causality but without back-projections, moving information upwards in scale without any step back. Finally, a similar causality and adaptation in the number of channels per scale exists in the SlowFast architecture (Feichtenhofer et al. 2019) but again without back-projections.

**Contributions.** Our major contribution is the introduction of a new network architecture that extends ResNets from single to multiple resolutions, with a clear representation in terms of ODE dynamic. Our main focus is to evaluate this extension and to prove that it is beneficial with respect to conventional ResNets. We also verify that the multi-scale dynamic of the network is being used to achieve improved performance and we visualize the dynamic of the network in solving different problems. BPP can be used to solve joint local problems, as well as combinations of global and local problems, getting state-of-art results in image-SR and competitive results for other problems using a single network configuration. Finally, we also show empirical evidence that BPP effectively uses both local and global information to solve problems.

**Architecture Design**

In Figure 2 (a) we observe the workflow of the Iterative Back-Projections (Irani and Peleg 1991) (IBP) algorithm:  
\[ h^{0} = P x, \quad h^{t+1} = h^{t} + P e(h^{t}), \quad e(h^{t}) = x - R h^{t}. \]  

IBP upscales an image \( x \) with a linear operator \( P \) and sends it back to low-resolution to verify the downsampling model represented by a linear operator \( R \). Now, we propose to extend the IBP algorithm to multiple scales by using the data pipeline approach shown in Figure 2 (b). Specifically, as soon as we get the first upscale image, we take it as reference and start a new upsampling to a higher resolution. Next, we downscale the second high-resolution image to verify the downsampling model. However, the reference image has been changed by the back-projection update at the lower level. At the lowest resolution the image never changes, and upper level iterations need to keep track of the lower level updates. In Figure 2 (b) we identify the essential computational block to assemble the pipeline: the Flux unit. The Flux unit is what makes scale travel possible by connecting input and output images from different levels.

**Network Architecture.** Without loss of generality, we tackle the image enhancement problem with an input res-
Algorithm 1 Back–Projection Pipeline (BPP)

\[
\text{BPP}(\text{input}, L, D):
\]

\[
\begin{align*}
\text{Input:} & \quad \text{Image input.} \\
\text{Input:} & \quad \text{Integer } L \geq 1, D \geq 1. \\
\text{Output:} & \quad \text{Image output.} \\
1: & \quad x_1^2 = \text{input} \\
2: & \quad \text{for } k = L - 1, \ldots, 1 \text{ do} \\
3: & \quad s_k^0 = \text{Scaler}^1_k(x_{k+1}^2) \\
4: & \quad \text{end for} \\
5: & \quad x_2 = \text{Analysis}^1_k(s_k^0) \\
6: & \quad \text{for } k = 1, \ldots, L - 1 \text{ do} \\
7: & \quad x_k = \text{Analysis}^2_k(s_k^1) \\
8: & \quad s_k^1 = \text{Scaler}^2_k(x_k^2) \\
9: & \quad p_k = \text{Analysis}^2_k(s_k^1) \\
10: & \quad \text{end for} \\
11: & \quad \text{for } l = 1, \ldots, D \text{ do} \\
12: & \quad x, p = \text{FluxBlock}(x, p, L) \\
13: & \quad \text{end for} \\
14: & \quad \text{output} = \text{input} + \text{Synthesis}(x_L)
\end{align*}
\]

Figure 3: Back–Projection Pipeline network diagram. On the left, a detailed diagram shows all back–projection modules. On the right, the diagram is simplified by using Flux units.

The full BPP algorithm and network configuration is specified in Algorithm 1 and Figure 3. To extend the pipelining approach into a network configuration, first, we initialize the network states \( x_k \) and down–projections \( p_k \) using linear down scalers and single convolutional layers in the Analysis modules to increase the number of channels. Second, states are updated using the Flux–Blocks defined in Algorithm 1, calculating residuals \( e_k \) and updating states upwards in scale with flux units. Third, the output state in the highest resolution is converted into a residual image by a convolutional layer in the Synthesis module and added to the input image.

### Network Dynamic

The restriction operators \( R_k \) (Downscale module) and interpolation operators \( P_k \) (Upscale module) are now non–linear and do not share parameters (time dependent). When we interpret depth as time \( t \), the dynamic is described in Figure 4 and leads to the following set of difference equations with their correspondent extension to continuous time:

\[
\begin{align*}
\frac{dh_k}{dt} & = P_k(R_k(h_k, t), h_{k-1}^+, t) \\
\frac{dh_1}{dt} & = h_1^+, t \\
\end{align*}
\]

\[
\begin{align*}
\frac{dh_k}{dt} & = P_k(R_k(h_k, t), h_{k-1}^+, t) \\
h_1(x, y, t) & = h_1(x, y, 0) .
\end{align*}
\] (2)

In the case of ResNets, the dynamical systems is given by \( h^{t+1} = h^t + f(h^t, t) \) and \( \frac{dh}{dt} = f(h, t) \) in continuous time. Therefore, BPP extends the model of ResNets from a single ODE to a system of coupled equations. Scale–causality follows from (2) as state \( h_k \) only depends on \( h_{k-1}, h_{k-2}, \ldots \). The multi–scale nature follows from the spatial dimension of state vectors \( h_k \), explicitly expressed in operators \( P_k : \mathbb{R}^{2} \times \mathbb{W} \rightarrow \mathbb{R}^{D} \times \mathbb{W} \) and \( R_k : \mathbb{R}^{D} \times \mathbb{W} \rightarrow \mathbb{R}^{2} \times \mathbb{W} \). In continuous space we could also express the multi–scale nature of the equations by using initial conditions \( h_{k-1}(x, y, t = 0) = h_k(2^s x, 2^s y, t = 0) \) with \( s \in \mathbb{N} \) with no filtering needed in continuous space, since aliasing effects do not exist. We observe that initial conditions are self–similar in scale(Mallat 1998). Whether this property is maintained in time depends on the evolution of the network state. In
the continuous time model, the restriction operator \( R_k \) in (2) represents a renormalization–group transformation of the network state, similar to those used in particle physics and ODEs to ensure self–similarity (Fisher 1974; Chen, Goldenfeld, and Oono 1996). In this sense, using different parameters at each scale allows the model to adjust the level of self–similarity that works better for a given problem. On the other hand, using different parameters in time can also be beneficial. It has been observed in (Liao and Poggio 2016) that normalization layers do not work well in recurrent networks, which share parameters in time. But in time–dependent systems, these layers become beneficial. Since the BPP configurations in our experiments use IN–layers, we chose to use different parameters in time. This does not have a significant effect in performance, because the flux–block structure in Algorithm 1 uses inline updates that avoid storage of old network states. During training, a checkpoint strategy can effectively reduce the memory footprint (Chen et al. 2016).

Using pipelining to extend IBP into multiple scales is simple and this is the major strength of this approach. There are several ways to extend IBP to multiple scales. We mentioned MGBP as a relevant but different approach. BPP is simpler, and that simplicity translates to a clear ODE model that is difficult to obtain otherwise. Most importantly, this ODE model is very expressive about the connection to IBP. It is direct from (2) that if the composition of \( P \) and \( R \) operations forms a contraction mapping then the ODE model will converge, which is the same argument used in convergence proofs of IBP in the linear case (Irani and Peleg 1991). At this point BPP departs from IBP. Because BPP is trained in a supervised fashion, we do not know a priori how this dynamic going to be driven towards the target. Overall, the BPP model inherits the essence of IBP in terms of an iteration that updates residuals upwards in scale, which can now be trained to reach diverse targets in a non–linear fashion using convolutional networks. The main purpose of our investigation is: first, to generalize the IBP dynamic to multiple scales in sequence; and second, to study how powerful is this dynamic so solve more general problems.

Finally, we note that the continuous model in (2) allows BPP to work as a Neural–ODE system (Chen et al. 2018). For the sake of simplicity, in this work we do not explore this direction. However, it stands as an interesting direction for future research.

Experiments

In our experiments we found that using IN–layers to activate ReLU units, as shown in Figure 3, could help converge faster in early training and doing so independent of initialization. Figure 5 (b) shows this effect and we also see that IN–layers are not required for BPP in the long run. In early training IN layers placed before ReLUs force a 50% activation in all flux units across all scales. This strategy shows to be a good choice to initialize parameters. Alternatively, we found that the most effective way to avoid IN–layers is using Dirac–kernels to initialize weights and adding Gaussian noise. This initialization was used in the learning curve BPP \((\text{no IN})\) in Figure 5 (b) and it is the closest we have found to avoid normalization layers.

Because of memory limitations we used a patch–based training strategy, where smaller–sized patches are taken from training set images. Patch–based learning reduces the receptive field of the network during training. At inference the performance of the network reduces if the mean and variance of IN–layers are computed on an image larger than the training patches. To solve this problems we: first, divide input images into overlapping patches (of same size as training patches); second, we multiply each output by a Hamming window (Harris 1978); and third, we average the results. In all our experiments we use overlapping patches separated by 16 pixels in vertical and horizontal directions. The weighted average helps to avoid blocking artifacts.

On one hand, this approach introduces redundancy and reduces performance for medium size images. On the other hand, it also allows the algorithm to run on very large images (e.g. 8K) and can be massively parallelized by batch processing in multiple GPUs.

Configuration. In the following experiments we use a BPP configuration with 16 back–projection layers (flux–blocks), 4 resolution levels and 256, 128, 64 and 48 features per level from lowest to highest resolution, respectively. All convolutional layers use \( 3 \times 3 \) as kernel size, and scalers are initialized with bicubic filters of size \( 9 \times 9 \) and trained as additional parameters. A fully unrolled diagram is shown in the Appendix. The configuration was tuned according to validation performance for the most challenging problems (e.g. SR–8\(\times\)). By fixing the configuration we can potentially have the architecture hardwired in silicon and update its model parameters to switch between different problems.

Performance. The BPP architecture is multi–scale and sequential. The so–called Flux–Block in Algorithm 1 represents the sequential block and consists of one Flux unit per level. This sequential structure is more convenient for memory performance as it avoids buffering of features from previous blocks. Architectures such as Dense–Nets, U–Nets and MGBP need to buffer features in skipped connections and thus need more memory. Because the configuration is fixed, the performance of the system can be roughly estimated from average statistics. The system has a total of 19 million parameters and it can process 1.7 million pixels per second on a Titan X GPU using 16–bit floating point precision. This means, for example, that it takes 3.7 seconds to process a Full–HD image in RGB format \((3 \times 1920 \times 1080 \text{ pixels})\).
Table 1: Quantitative evaluation for SR. A more extensive comparison is available in the Appendix.

| Algorithm         | Set14 | BSDS100 | Urban100 | Manga109 |
|-------------------|-------|---------|----------|----------|
|                   | PSNR–Y_{M} | SSIM–Y_{M} | PSNR–Y_{M} | SSIM–Y_{M} | PSNR–Y_{M} | SSIM–Y_{M} | PSNR–Y_{M} | SSIM–Y_{M} |
| Bicubic           | 30.34  | 0.870   | 29.56    | 0.844    | 26.88     | 0.841     | 30.84     | 0.935     |
| MSLaSRN           | 33.28  | 0.915   | 32.05    | 0.898    | 31.15     | 0.919     | 37.78     | 0.976     |
| D-DBPN 2×         | 33.85  | 0.919   | 32.27    | 0.900    | 32.70     | 0.931     | 39.10     | 0.978     |
| EDSR              | 33.92  | 0.919   | 32.32    | 0.901    | 32.93     | 0.935     | 39.10     | 0.977     |
| RDN 4×             | 34.28  | 0.924   | 32.46    | 0.903    | 33.36     | 0.939     | 39.74     | 0.979     |
| BPP–SRx2x3x4x8    | 33.27  | 0.913   | 31.21    | 0.879    | 31.07     | 0.921     | 38.31     | 0.975     |
| BPP–SRx2         | 34.23  | 0.922   | 31.63    | 0.886    | 33.07     | 0.935     | 39.19     | 0.977     |
| Bicubic           | 27.55  | 0.774   | 27.21    | 0.739    | 24.46     | 0.735     | 26.95     | 0.856     |
| MSLaSRN           | 29.97  | 0.836   | 28.93    | 0.800    | 27.47     | 0.837     | 32.68     | 0.939     |
| EDSR 3×           | 30.52  | 0.846   | 29.25    | 0.809    | 28.80     | 0.865     | 34.17     | 0.948     |
| RDN 8×             | 30.74  | 0.850   | 29.38    | 0.812    | 29.18     | 0.872     | 34.81     | 0.951     |
| BPP–SRx2x3x4x8    | 30.23  | 0.838   | 28.81    | 0.794    | 28.43     | 0.852     | 33.75     | 0.943     |
| BPP–SRx3          | 30.78  | 0.848   | 29.14    | 0.804    | 29.56     | 0.873     | 34.49     | 0.948     |
| Bicubic           | 26.10  | 0.704   | 25.96    | 0.669    | 23.15     | 0.659     | 24.92     | 0.789     |
| MSLaSRN           | 28.26  | 0.774   | 27.43    | 0.731    | 25.51     | 0.768     | 29.54     | 0.897     |
| D-DBPN 4×         | 28.82  | 0.786   | 27.72    | 0.740    | 26.53     | 0.795     | 31.18     | 0.914     |
| EDSR              | 28.80  | 0.788   | 27.71    | 0.742    | 26.64     | 0.803     | 31.02     | 0.915     |
| RDN 8×             | 29.01  | 0.797   | 27.85    | 0.745    | 27.01     | 0.812     | 31.74     | 0.921     |
| BPP–SRx2x3x4x8    | 28.95  | 0.778   | 27.43    | 0.728    | 26.48     | 0.799     | 30.81     | 0.909     |
| BPP–SRx4          | 29.07  | 0.791   | 27.89    | 0.745    | 27.55     | 0.819     | 31.63     | 0.918     |
| Bicubic           | 23.19  | 0.568   | 23.67    | 0.547    | 20.74     | 0.516     | 21.47     | 0.647     |
| MSLaSRN           | 24.57  | 0.629   | 24.65    | 0.592    | 22.06     | 0.598     | 23.90     | 0.759     |
| D-DBPN 8×         | 25.13  | 0.648   | 24.88    | 0.601    | 22.83     | 0.622     | 25.30     | 0.799     |
| EDSR              | 24.94  | 0.640   | 24.80    | 0.596    | 22.47     | 0.620     | 24.58     | 0.778     |
| RDN                | 25.38  | 0.654   | 25.01    | 0.606    | 23.04     | 0.644     | 25.48     | 0.806     |
| BPP–SRx2x3x4x8    | 25.10  | 0.642   | 24.89    | 0.598    | 22.72     | 0.626     | 24.78     | 0.785     |
| BPP–SRx8          | 25.53  | 0.655   | 25.11    | 0.607    | 23.17     | 0.649     | 25.28     | 0.800     |

P1: Image Super–Resolution. We use DIV2K (Agustsson and Timofte 2017) and FLICKR–2K datasets for training and the following datasets for test: Set–14(Zeyde, Elad, and Protter 2010), BSDS–100(Martin et al. 2001), Urban–100(Huang, Singh, and Ahuja 2015) and Manga–109(Matsui et al. 2017). Impaired images were obtained by downscaling and then upsampling ground truth images, using Bicubic scaler, and scaling factors: 2×, 3×, 4× and 8×. Here, we consider two cases: we trained models BPP–SR×f for each upsampling factor f = 2, 3, 4 and 8; and we also trained a single model BPP–SRx2x3x4x8 to restore impaired images with unknown upsampling factors. We use 16 patches per mini–batch with patch size 48f × 48f for known upsampling factor f, and 192 × 192 for unknown upsampling factor, all at high resolution.

Table 1 and Figure 5 (a) show quantitative and qualitative results compared to other methods. We focus our comparison to the following methods: Bicubic (the baseline); EDSR (Lim et al. 2017), with major processing in 1 resolution level using a 32–layer ResNet; Dense–DBPN (Harris, Shakhnarovich, and Ukit 2018), with major processing in 2 resolution levels using 12 densely connected up/down back–projections; and RDN (Zhang et al. 2018c), with major processing in 1 resolution level using 20 densely connected residual–dense–blocks. We show EDSR and DBPN because they are both closely related to BPP in their residual and back–projection structures, respectively, and we show RDN as a top reference of current state–of–the–arts. Further comparisons with other methods can be found in the Appendix.

Overall, for the problem of super–resolution we find that BPP can get excellent results, reaching state–of–the–arts results in both quantitative and qualitative evaluations, but it decreases its performance when we test a more general problem. First, BPP–×f models get the best scores in most quantitative and qualitative evaluations, with RDN slightly outperforming BPP at 2× and 3× upsampling factors. This setting, including datasets for training and test, is the most common evaluation procedure for supervised SR technics. In terms of application this would be useful if we need to enhance an image upscaled with Bicubic scaler with a specific upsampling factor. It often happens that we have an image upscaled with an unknown factor and in this case we do not know which model parameters to load. In this case the BPP–SRx2x3x4x8 model offers a general upsampling...
solution. This performance of these BPP models decrease and not reach state-of-the-arts results. Although reasonably close to state-of-the-arts, often outperforming EDSR, we would have expected this model to perform better than BPP-×f if the architecture was able to generalize effectively to this more general setting. In fact, it has been observed in VDSR (Kim, Lee, and Lee 2016a) and MDSR (Lim et al. 2017) that training with unkown upscaling factors can improve the performance of the network. Therefore, these empirical results show that BPP can be very effective for fixed upscaling factors but does not generalize as well as other architectures for general upscaling factors.

P2: Raindrop Removal. We use the DeRaindrop (Qian et al. 2018a,b) dataset for training and test. This dataset provides paired images, one degraded by raindrops and the other one free from raindrops. These were obtained by use two pieces of exactly the same glass: one sprayed with water, and the other is left clean. In each training batch, we take 1 patch of size 528 × 528. We train a BPP model using L1 loss and patch size 456 × 456. More details of training settings are provided in the Appendix.

This problem is very different in nature to super-resolution. On one hand, a significant portion of pixels contain (uncorrupted) high-resolution information that must move to the output with little or no change. At the same time it needs to identify the irregular distribution of raindrops, with different sizes, and fill-in those areas by predicting the content within. In some images the content within raindrops is of little use, making the problem similar to inpainting. Thus, the problem requires processing of both local and global information in order to fill-in raindrops.

Even though we only trained our system with an L1 loss, our system performs similar to the state-of-the-arts DeRaindrop (Qian et al. 2018a) as seen in Table 2 and Figure 6. The DeRaindrop network in (Qian et al. 2018a) uses an attentive GAN approach that can estimate raindrop masks to focus on these areas for restoration. The PSNR score of BPP is better than DeRaindrop without adversarial training, and the SSIM score is better than all other systems in Table 2. The qualitative evaluation shows that BPP achieves a reasonable quality, considering the fact that it has not been trained using GANs. Here, the BPP architecture appears to be effective. In the next section we inspect properties of the network that reveal the undergoing mechanism used by BPP to obtain its solutions.

Other Problems. The performance in other problems, including mobile-to-DSLR photo translation, dehaze and joint HDR+SR are included in the Appendix.

Inspection of ODE updates. We conduct experiments to measure the magnitude of the updates in equation (2) to better understand the dynamic of the network when solving different problems. The arrange of Flux units in BPP networks forms an array of size L × D (number of levels times depth) and we compare the magnitude of residual updates in each one of these units. In Figure 7 we display the result of measuring

\[
\left\| \frac{dh_k}{dt} \right\|_2 = \left\| P_k(R_{h_k^t, t}, h_{k-1}^{t+1}, t) \right\|_2,
\]

for every flux unit, averaged over all images in the validation sets, and normalized to the maximum update (fixed to 100).

At the lowest resolution (k = 4) the reference image never changes and thus the updates is always zero.

Interestingly, we observe that the dynamic is far from the original contraction mapping design of IBP, that would result in an exponential decay of updates along depth. Here, we should remember that the dynamic is driven exclusively by the result of training the network in supervised manner.

Table 2: Quantitative results of raindrop removal.

| Method                | PSNR–Y_P | SSIM–Y_P |
|-----------------------|----------|----------|
| Eigen13               | 28.59    | 0.6726   |
| Pix2Pix               | 30.14    | 0.8299   |
| DeRaindrop (No GAN)   | 29.25    | 0.7853   |
| DeRaindrop            | 31.57    | 0.9023   |
| BPP                   | 30.85    | 0.9180   |
Instead of an exponential decay, the network consistently shows a bimodal statistic with one peak very close to the input and another very close to the output. Also, the highest resolution receives very small updates meaning that these feature move more or less unchanged with an increased update towards the end. The major processing goes on at the middle levels. In SR updates are stronger at the lower resolution ($k = 3$) and for RainDrop removal updates are stronger at the higher resolution ($k = 2$). The bimodal statistic is reminiscent of interpretability results for VGG networks in classification, that show higher contribution to label outputs very early and very late in a sequential configuration (Navarrete Michelini et al. 2019). Nevertheless, in BPP the updates focus on one or two resolution levels as opposed to VGG networks that are designed to process high resolutions early in the network and very low resolutions towards the end. Despite this important difference, these results suggest that sequential networks find solutions in two steps: analysis at the first layers, and fusion towards the very end.

**Interpretability.** We apply the LinearScope method from (Navarrete Michelini et al. 2019) to analyze the learning process in global and local problems. The general methodology is as follows. The BPP architecture contains several non-linear modules consisting on ReLUs and IN-layers. The decision of which pixels pass or stop in ReLUs, and what mean and variance to use in IN-layers, is non-linear. But the action of these layers are linear: masking and normalizing. For a given input image $x$, the action of all non-linear modules (ReLU and IN-layers) can be fixed as: $1/0–masks$ for ReLU and fixed mean and variance in IN-layers. This gives a linear system of the form $y = Fx + r$ that generates the same output as the non-linear system for the input $x$, and represents the overall action of the network on the input image.

The matrix $F$ represents the interpolation filters used by the network to solve the problem, and thus shows the local processing. The residual $r$ is a fixed global image created by non-linear modules. Figure 8 shows the local contributions, $Fx$, and global contributions, $r$, for three systems. We observe that EDSR almost purely relies on local processing to obtain an output. BPP, on the other hand, relies mostly on local processing but the contribution of $r$ is much larger than the one in EDSR. This shows a significantly different approach followed by BPP, compared to EDSR, to solve the super-resolution problem.

The BPP system for raindrop removal reveals a much larger contribution of $r$, that resembles a mask of raindrops. This means that BPP uses a local approach on areas without raindrops (using $Fx$) and a global approach on raindrops determined by the residual $r$. The mechanism used by the network to obtain the residual $r$ is non-linear. Overall, we observe that for this problem the BPP network divides the problem in two parts: a local adaptive filter in clean areas, to nearly copy–paste the input into the output; and a non-linear global approach to fill-in raindrop areas.

### Conclusions

We propose Back–Projection Pipeline as a simple yet non-trivial extension of residual networks (ResNets) to run in multiple resolutions. The update dynamic through the layers of the network includes interactions between different resolutions in a way that is causal in scale, and it is represented by a system of ODEs. We use it as a generic multi-resolution approach to enhance images. The focus of our investigation is to evaluate this multi–scale residual approach. Overall, our empirical results show that BPP can achieve excellent results in traditional supervised learning. Our BPP configuration gets state–of–the–art results in SR for fixed upscaling factors and competitive results for raindrop removal as well as other problems (see Appendix). We also observe a lack of generalization for the problem of SR with unknown upscaling factors. Inspection of the residual updates in the network shows that all resolution levels are being used, with higher intensity in lower resolutions, showing that supervised training gives preference to the multi–scale setting over traditional residual networks. Based on our results, we cannot conclude that scale causality is beneficial. Nevertheless, we can at least conclude that this strong simplification in the flow of network information, inherited from IBP, does not prevent the architecture to achieve competitive results. Further investigation is necessary in this regard (especially regarding generalization) and it could open interesting research directions in network architecture search and design.
Figure 9: Detail diagram of the 4-level, 16-layers BPP configuration used in our experiments.

References

Agustsson, E.; and Timofte, R. 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. 5, 11

Ancuti, C.; Ancuti, C. O.; and Timofte, R. 2019. NTIRE–2019 Dehaze Evaluation code. https://competitions.codalab.org/my/datasets/download/a85c0d2-cf8b-4ec8-bf83-243c7bda515. [Online; accessed 20-May-2019]. 11

Ancuti, C.; Ancuti, C. O.; and Timofte, R. 2019. NTIRE–2019 Workshops Conference on Computer Vision and Pattern Recognition (CVPR). 5, 12

Ancuti, C.; Ancuti, C. O.; Timofte, R.; Van Gool, L.; Zhang, L.; and Yang, M. 2018b. NTIRE 2018 Challenge on Image Dehazing: Methods and Results 891–901. 1

Ancuti, C. O.; Ancuti, C.; Sbert, M.; and Timofte, R. 2019. Dense Haze: A benchmark for image dehazing with dense-haze and haze-free images. CoRR abs/1904.02904. URL http://arxiv.org/abs/1904.02904. 4

Chen, T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. neural information processing systems 6571–6583. 4

Chen, T.; Xu, B.; Zhang, C.; and Guenter, C. 2016. Training Deep Nets with Sublinear Memory Cost. arXiv: Learning. 4

Dong, C.; Loe, C. C.; He, K.; and Tang, X. 2014. Learning a Deep Convolutional Network for Image Super–Resolution. In in Proceedings of European Conference on Computer Vision (ECCV). 1, 12

Dong, C.; Loe, C. C.; and Tang, X. 2016. Accelerating the Super–Resolution Convolutional Neural Network. In in Proceedings of European Conference on Computer Vision (ECCV). 12

Ehret, T.; Davy, A.; Arias, P.; and Facciolo, G. 2019. Joint demoising and denoising by overfitting of bursts of raw images. 2

Feichtenhofer, C.; Fan, H.; Malik, J.; and He, K. 2019. SlowFast Networks for Video Recognition. In The IEEE International Conference on Computer Vision (ICCV). 2

Fisher, M. E. 1974. The renormalization group in the theory of critical behavior. Reviews of Modern Physics 46(4): 597–616. 4

Gharbi, M.; Chaurasia, G.; Paris, S.; and Durand, F. 2016. Deep Joint Demosaicking and Denoising. ACM Trans. Graph. 35(6): 191:1–191:12. ISSN 0730-0301. doi:10.1145/2980179.2982399. URL http://doi.acm.org/10.1145/2980179.2982399. 2

Haris, M.; Shakhnarovich, G.; and Ukita, N. 2018. Deep Back–Projection Networks for Super–Resolution. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 5, 12

Harris, F. J. 1978. On the use of windows for harmonic analysis with the discrete Fourier transform. Proceedings of the IEEE 66(1): 51–83. 4

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. computer vision and pattern recognition 770–778. 1

He, M.; Chen, D.; Liao, J.; Sander, P. V.; and Yuan, L. 2018. Deep exemplar-based colorization. ACM Transactions on Graphics (TOG) 37(4): 47. 1

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-Excitation Networks. computer vision and pattern recognition 7132–7141. 1

Hu, X.; Mu, H.; Zhang, X.; Wang, Z.; Tan, T.; and Sun, J. 2019. Meta-SR: A Magnification–Arbitrary Network for Super-Resolution. arXiv: Computer Vision and Pattern Recognition. 3

Huang, G.; Chen, D.; Li, T.; Wu, F.; Der Maaten, L. V.; and Weinberger, K. Q. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. International Conference on Learning Representations. 2

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 1, 10

Huang, J.; Singh, A.; and Aluja, N. 2015. Single image super-resolution from transformed self-exemplars 5197–5206. 5, 11

Ignatov, A.; Kobyshev, N.; Timofte, R.; Vanhoey, K.; and Van Gool, L. 2017. DSLR-quality photos on mobile devices with deep convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, 3277–3285. 11

Irani, M.; and Peleg, S. 1991. Improving Resolution by Image Registration. CVGIP: Graph. Models Image Process. 53(3): 231–239. ISSN 1049-9652. doi:10.1016/1049-9652(91)90045-L. URL http://dx.doi.org/10.1016/1049-9652(91)90045-L. 2, 4

Kim, J.; Lee, J. K.; and Lee, K. M. 2016a. Accurate Image Super–Resolution Using Very Deep Convolutional Networks. In The IEEE Conference on Computer Vision and Pattern Recognition. 6, 12
Kim, J.; Lee, J. K.; and Lee, K. M. 2016b. Deeply-Recursive Convolutional Network for Image Super-Resolution. In The IEEE Conference on Computer Vision and Pattern Recognition. 12

Kim, S. Y.; Oh, J.; and Kim, M. 2019. Deep SR-ITM: Joint Learning of Super-Resolution and Inverse Tone-Mapping for 4K UHD HDR Applications. In The IEEE International Conference on Computer Vision (ICCV). 2

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. international conference on learning representations. 11, 17

Kinoshita, Y.; and Kiya, H. 2019. Convolutional Neural Networks Considering Local and Global features for Image Enhancement. In The IEEE International Conference on Image Processing (ICIP). 2

Kokkinos, F.; and Lefkimmiatis, S. 2018. Deep image demosaicking using a cascade of convolutional residual denoising networks. In Proceedings of the European Conference on Computer Vision (ECCV), 303–319. 1

Kundu, D.; Ghadiyaram, D.; Bovik, A. C.; and Evans, B. L. 2017a. Evaluation code for HIGRADE metric. http://live.ece.utexas.edu/research/Quality/higradeRelease.zip. [Online; accessed 20-May-2019]. 10, 11

Kundu, D.; Ghadiyaram, D.; Bovik, A. C.; and Evans, B. L. 2017b. No-Reference Quality Assessment of Tone-Mapped HDR Pictures. IEEE Transactions on Image Processing 26(6): 2957–2971. 10, 11

Lai, W.; Huang, J.; Ahuja, N.; and Yang, M. 2018. Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(3): 2599–2613. 12

Lai, W.-S.; Huang, J.-B.; Ahuja, N.; and Yang, M.-H. 2017. Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution. In IEEE Conference on Computer Vision and Pattern Recognition. 12

Lefkimmiatis, S. 2018. Universal denoising networks: a novel CNN architecture for image denoising. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3204–3213. 1

Liao, Q.; and Poggio, T. 2016. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. arXiv preprint arXiv:1604.03640. 1, 4

Lim, B.; Son, S.; Kim, H.; Nah, S.; and Lee, K. M. 2017. Enhanced Deep Residual Networks for Single Image Super–Resolution. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. 1, 5, 6, 11, 12

Lindeberg, T. 1994. Scale-Space Theory in Computer Vision. ISBN 0-7923-9418-6. 9

Luo, D.; Ouyang, W.; Xu, D.; Zhang, X.; Gao, Z.; and Sun, M.-T. 2018. Deep Kalman filtering network for video compression artifact reduction. In Proceedings of the European Conference on Computer Vision (ECCV), 568–584. 1

Ma, C.; Yang, C.-Y.; Yang, X.; and Yang, M.-H. 2017. Learning a No-Reference Quality Metric for Single-Image Super-Resolution. Computer Vision and Image Understanding 1–16. 11

Ma, C.; Yang, C.-Y.; Yang, X.; and Yang, M.-H. 2018. Evaluation code for Ma–metric. https://github.com/chaoma99/sr-metric. [Online; accessed 20-May-2019]. 11

Mallat, S. 1998. A Wavelet Tour of Signal Processing. Academic Press. 1, 3

Martin, D. R.; Fowlkes, C. C.; Tal, D.; and Malik, J. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics 2: 416–423. 5, 11

Matsui, Y.; Ito, K.; Aramaki, Y.; Fujimoto, A.; Ogawa, T.; Yamasaki, T.; and Aizawa, K. 2017. Sketch-based manga retrieval using manga109 dataset. Multimedia Tools and Applications 76(20): 21811–21838. 5, 11

Navarrete Michelini, P.; Liu, H.; Lu, Y.; and Jiang, X. 2019. A Tour of Convolutional Networks Guided by Linear Interpreters. In The IEEE International Conference on Computer Vision (ICCV). IEEE. URL https://arxiv.org/abs/1908.05168. 7

Navarrete Michelini, P.; Liu, H.; and Zhu, D. 2019. Multigrid Back-projection Super–Resolution and Deep Filter Visualization. In Proceedings of the Thirty–Third AAAI Conference on Artificial Intelligence (AAAI 2019). AAAI. 2. 7, 12

Nemoto, H.; Korshunov, P.; Hanhart, P.; and Ebrahimi, T. 2015. Visual attention in LDR and HDR images. Technical report. URL https://nmmsg.epfl.ch/downloads/hdr-eye/. 11

Oord, A. v. d.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; and Kavukcuoglu, K. 2016. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499. 2

Paszke, A.; Gross, S.; Chintalas, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In NIPS-W. 17

Poggio, T.; Mhaskar, H.; Rosasco, L.; Miranda, B.; and Liao, Q. 2017. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. International Journal of Automation and Computing 14(5): 503–519. 1

Qian, G.; Gu, J.; Ren, J. S.; Dong, C.; Zhao, F.; and Lin, J. 2019. Trinity of Pixel Enhancement: a Joint Solution for Demosaicking, Denoising and Super-Resolution. arXiv e-prints arXiv:1905.02538. 2

Qian, R.; Tan, R. T.; Yang, W.; Su, J.; and Liu, J. 2018a. Attentive Generative Adversarial Network for Raindrop Removal From a Single Image. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1, 6, 17

Qian, R.; Tan, R. T.; Yang, W.; Su, J.; and Liu, J. 2018b. De-Raindrop dataset. https://drive.google.com/open?id=1e7R76s6wUx1LOcAsthgDLPSnOrQ49K. [Online; accessed 20-May-2019]. 6, 17

Qian, R.; Tan, R. T.; Yang, W.; Su, J.; and Liu, J. 2018c. Evaluation code for Raindrop removal. https://github.com/ru1996/DeRaindrop/blob/master/metrics.py. [Online; accessed 20-May-2019]. 11

Reinhard, E.; and Devlin, K. 2005. Dynamic range reduction inspired by photoreceptor physiology. IEEE Transactions on Visualization and Computer Graphics 11(1): 13–24. 11

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. medical image computing and computer assisted intervention 234–241. 1, 2

Soh, J. W.; Park, J. S.; and Cho, N. I. 2019. Joint High Dynamic Range Imaging and Super-Resolution from a Single Image. arXiv:1905.00933 [cs, ess] URL http://arxiv.org/abs/1905.00933. ArXiv: 1905.00933. 2, 11

Tao, X.; Gao, H.; Shen, X.; Wang, J.; and Jia, J. 2018. Scale-recurrent Network for Deep Image Deblurring. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1
Wang, S.; Zheng, J.; Hu, H.-M.; and Li, B. 2013. Naturalness preserved enhancement algorithm for non-uniform illumination images. *IEEE Transactions on Image Processing* 22(9): 3538–3548.

Trottenberg, U.; and Schuller, A. 2001. *Multigrid*. Orlando, FL, USA: Academic Press, Inc. ISBN 0-12-701070-X.

Zhang, K.; Zuo, W.; and Zhang, L. 2018. Learning a Single Convolutional Super-Resolution Network for Multiple Degradations. *Computer Vision and Pattern Recognition (CVPR) Workshops*. 11

Figure 9 shows an expanded diagram of the single BPP configuration used in our experiments. It uses 16 back-projection layers (flux blocks), 4 resolution levels and 256, 128, 64 and 48 features per level from lowest to highest resolution, respectively. All convolutional layers use $3 \times 3$ as kernel size, and scalers are initialized with bicubic filters of size $9 \times 9$ and trained as additional parameters.

We observe that, after initialization, the lowest-resolution network state (at the bottom of the diagram) never changes. Thus, the highest-resolution state (at the top of the diagram) is always 3-layers away from this fixed state. This is similar to a long-range skip-connection in DenseNet (Huang et al. 2017), but in BPP these shortcut moves through a different resolution. Because of scale causality, the next low-resolution level moves relatively close to the fixed state and we can interpret it as a shorter-range skip-connection. Thus, the particular structure of BPP allows quick paths from the output to every layer of the network, similar to DenseNets, which is convenient for the gradient flow during back-propagation steps.

**Evaluation Metrics**

Quantitative evaluations in our experiments include three objective metrics: PSNR, SSIM and HIGRADE–2. From these, PSNR and SSIM are reference-based metrics that measure the difference between an impaired image and ground truth. Higher values are better in both cases. The PSNR (range 0 to $\infty$) is a log-scale version of mean-square-error and SSIM (range 0 to 1) uses image statistics to better correlate with human perception. Full expressions are as follows:

$$PSNR(X,Y) = 10 \cdot \log_{10} \left( \frac{255^2}{MSE} \right),$$  

$$SSIM(X,Y) = \frac{(2\mu_X\mu_Y + c_1)(2\sigma_{XY} + c_2)}{\left(\mu_X^2 + \mu_Y^2 + c_1\right)\left(\sigma_X^2 + \sigma_Y^2 + c_2\right)},$$

where $MSE = \mathbb{E}[(X - Y)^2]$ is the mean square error of the difference between $X$ and $Y$; $\mu_X$ and $\mu_Y$ are the averages of $X$ and $Y$, respectively; $\sigma_X^2$ and $\sigma_Y^2$ are the variances of $X$ and $Y$, respectively; $\sigma_{XY}$ is the covariance of $X$ and $Y$; $c_1 = 6.5025$ and $c_2 = 58.5225$.

HIGRADE–2 (Kundu et al. 2017b) is a non-reference image quality metric based on gradient scene–statistics defined in the LAB color space and it is often used to evaluate high-dynamic-range images. Here, we used the Matlab code available in (Kundu et al. 2017a).

**Appendix**

### Diagrams

In an effort to make diagrams easy to read, concise and carrying a precise meaning, we introduce the notation in Figure 10. This is, lines connected to the left-side of any given module represent different inputs to that module. Every module can have several inputs but only one output.
In the case of PSNR and SSIM metrics, we follow existing benchmarks that use different versions of these metrics. We used the following three definitions in our experiments:

- **PSNR/SSIM – Y_{M}:** Based on the Matlab code available in (Zhang et al. 2018b), computes PSNR/SSIM on the Y’ channel. Matlab uses a conversion of RGB to YUV color-spaces following the BT.709 standard, including offsets that are often avoided in other implementations.

- **PSNR/SSIM – Y_{P}:** Based on the Python code available in (Qian et al. 2018c), computes PSNR/SSIM on the Y channel. The code uses an OpenCV function to convert from RGB to YCbCr color-space.

- **PSNR/SSIM – RGB:** Based on the Python code available in (Ancuti, Ancuti, and Timofte 2019), computes the average PSNR/SSIM for pairs of RGB images.

### Training Settings

#### Image Super–Resolution
We use DIV2K (Agustsson and Timofte 2017) and FLICKR–2K datasets for training and the following datasets for test: Set–14(Zeyde, Elad, and Protter 2010), BSDS–100(Martin et al. 2001), Urban–100(Huang, Singh, and Aghuja 2015) and Manga–109(Matsui et al. 2017). Impaired images were obtained by downsampling and then upsampling ground truth images, using Bicubic scaler, with scaling factors: $2 \times$, $3 \times$, $4 \times$ and $8 \times$. Our target is to recover the ground truth so we use a loss function that measures the $L_1$ distance between impaired images and ground truth. For evaluation we measure PSNR and SSIM on the $Y$–channel using the Matlab code from (Zhang et al. 2018b).

We follow the training settings from (Lim et al. 2017). In each training batch, we randomly take 16 impaired patches from our training set (800 DIV2K plus 2,650 FLICKR–2K images). We consider two cases: we train a model BPP–SR × $f$ for each upsampling factor $f = 2, 3, 4$ and 8; and we also train a model BPP–SR × 2 × $3 \times 4 \times 8$ to restore impaired images with unknown upsampling factor. We use patch size $48 f \times 48 f$, for $f = 2, 3$ and 4, and $192 \times 192$ for $f = 8$ and unknown upsampling factor. We augment the patches by random horizontal/vertical flipping and rotating $90^\circ$. We use Adam optimizer(Kingma and Ba 2015) with learning rate initialized to $10^{-4}$ and decreased by half every 200,000 back–propagation steps.

The training data used for the BPP–SR × 3 × 4 × 8 model includes all images used for training the upsampling factors $f = 2, 3, 4$ and 8. We could have chosen to train our model using a random and fractional upsampling factor $2.0 \leq f \leq 8.0$, but this would have made it difficult to reproduce the training settings.

#### Mobile–to–DSLR Photo Translation
We use the DPED(Ignatov et al. 2017) dataset for training and test. This dataset provides $100 \times 100$ aligned patches taken from iPhone–mobile photos (impaired) and DSLR–Canon photos (ground truth). There are 160,471 patches available for training and 4,353 patches for test. We take 400 patches from the test set for validation during training. We use full size iPhone images from DPED for qualitative results. For loss function we use the negative SSIM between impaired and ground truth patches. We find SSIM to be more effective than $L_1$ and MSE losses in this problem. For evaluation we measure the average PSNR and SSIM metrics for RGB pairs, using the code from (Ancuti, Ancuti, and Timofte 2019), and the non–reference metric HIGRADE–2(Kundu et al. 2017b) using the Matlab code available from (Kundu et al. 2017a).

In each training batch, we take 16 patches of size $100 \times 100$. We use Adam optimizer(Kingma and Ba 2015) with learning rate initialized to $10^{-4}$ and decreased by half every 200,000 back–propagation steps. We do not observe improvements after 200 epochs.

#### Image Dehaze
We use the following real haze datasets: I–Haze(Anuci et al. 2018a), O–Haze(Anucut et al. 2018c) and Dense–Haze(Anucuti et al. 2019). We follow the training setting from (Zhang, Sindagi, and Patel 2018). In each training batch, we take 1 patch of size $528 \times 528$. The training set is augmented by rescaling the images, using bicubic scaler, to $1.25 \times$, $1 \times$, $0.625 \times$ and $0.3125 \times$ the original size. We use Adam optimizer(Kingma and Ba 2015) with learning rate initialized to $10^{-4}$ and decreased by half every 200,000 back–propagation steps. We train the system for 10,000 epochs.

#### Joint HDR and Super–Resolution
We use the HDR–Eye(Nemoto et al. 2015) dataset for training and Wang LDR(Wang et al. 2013) dataset for test. HDR–Eye(Nemoto et al. 2015) provides HDR images constructed from multi–exposure photographs. Following the training configuration in (Soh, Park, and Cho 2019), we select 40 from a total of 46 standard–exposed and HDR–constructed pairs of images (we excluded images C01.png, C04.png, C13.png, C28.png, C38.png and C42.png, because of visible misalignment problems in the HDR image constructions). Then, we take each standard–exposed image and we: first, downscale it by factor 2; and then upscale it by factor 2 (both with bicubic scaler), and use this output as impaired image. We follow the configuration in (Soh, Park, and Cho 2019) although their tone–mapping algorithms are not specified and tone–mapped images are not provided. We use several tone–mapping algorithms until being able to produce competitive quantitative and qualitative outputs. For our final results we used the OpenCV implementation of Reinhard–Devlin tone–mapping (Reinhard and Devlin 2005) with parameters gamma = 2.2, intensity = 0, lightadapt = 0, and coloradapt = 0.

We train our system using patches of size $456 \times 456$. Following the analysis in (Soh, Park, and Cho 2019), we use the non–reference image quality metrics: Ma(Ma et al. 2017, 2018), to evaluate SR improvements; and HIGRADE–2(Kundu et al. 2017b,a) to evaluate HDR improvements. We augment the patches by random horizontal/vertical flipping and rotating $90^\circ$. We use Adam optimizer(Kingma and Ba 2015) with learning rate initialized to $10^{-4}$ and decreased by half every 200,000 back–propagation steps. We train the system for 10,000 epochs.
Table 3: Extended quantitative evaluation for super-resolution.

| Algorithm                  | Set14 | BSDS100 | Urban100 | Manga109 |
|----------------------------|-------|---------|----------|----------|
|                            | PSNR−Y_m | SSIM−Y_m | PSNR−Y_m | SSIM−Y_m |
| Bicubic                    | 30.34 | 0.870   | 29.56    | 0.844    |
| A+ (Timofte and Smet 2014) | 32.40 | 0.906   | 31.22    | 0.887    |
| FSRCNN (Dong, Loy, and Tang 2016) | 32.73 | 0.909 | 31.51    | 0.891    |
| SRCNN (Dong et al. 2014)   | 32.29 | 0.903   | 31.36    | 0.888    |
| MSLapSRN (Lai et al. 2018) | 33.28 | 0.915   | 32.05    | 0.898    |
| VDSR (Kim, Lee, and Lee 2016a) | 32.97 | 0.913 | 31.90    | 0.896    |
| LapSRN (Lai et al. 2017)   | 33.08 | 0.913   | 31.80    | 0.895    |
| DRCN (Kim, Lee, and Lee 2018b) | 32.98 | 0.913 | 31.85    | 0.894    |
| MGBP (Naveerite Michelini, Liu, and Zha 2019) | 33.27 | 0.915 | 31.99    | 0.897    |
| D-DBPN (Hars, Shakhmarovitch, and Ukta 2018) | 33.85 | 0.919 | 32.27    | 0.900    |
| EDSR (Lim et al. 2017)     | 33.92 | 0.919   | 32.32    | 0.901    |
| RDN (Zhang et al. 2018c)   | 34.28 | 0.924   | 32.46    | 0.903    |
| RCAN (Zhang et al. 2018a)  | 34.12 | 0.921   | 32.41    | 0.903    |
| BPP−SRx2x3x4x8             | 33.27 | 0.913   | 31.21    | 0.879    |
| BPP−SRx2                   | 34.23 | 0.922   | 31.63    | 0.886    |
| Bicubic                    | 27.55 | 0.774   | 27.21    | 0.739    |
| SRCNN (Dong et al. 2014)   | 29.30 | 0.822   | 28.41    | 0.786    |
| MSLapSRN (Lai et al. 2018) | 29.97 | 0.836   | 28.93    | 0.800    |
| LapSRN (Lai et al. 2017)   | 29.87 | 0.832   | 28.82    | 0.798    |
| EDSR (Lim et al. 2017)     | 30.52 | 0.846   | 29.25    | 0.809    |
| RDN (Zhang et al. 2018c)   | 30.74 | 0.850   | 29.38    | 0.812    |
| BPP−SRx2x3x4x8             | 30.23 | 0.838   | 28.81    | 0.794    |
| BPP−SRx3                   | 30.78 | 0.848   | 29.14    | 0.804    |

Algorithm | Set14 | BSDS100 | Urban100 | Manga109 |
|----------|-------|---------|----------|----------|
| PSNR−Y_m | 25.96 | 0.669   | 23.15    | 0.659    |
| SSIM−Y_m | 25.96 | 0.669   | 23.15    | 0.659    |
| PSNR−Y_m | 25.96 | 0.669   | 23.15    | 0.659    |
| SSIM−Y_m | 25.96 | 0.669   | 23.15    | 0.659    |
| RCAN (Zhang et al. 2018a) | 30.85 | 0.873   | 31.74    | 0.921    |
| BPP−SRx3 | 30.78 | 0.848   | 29.14    | 0.804    |

Algorithm | Set14 | BSDS100 | Urban100 | Manga109 |
|----------|-------|---------|----------|----------|
| PSNR−Y_m | 26.10 | 0.704   | 25.96    | 0.669    |
| SSIM−Y_m | 26.10 | 0.704   | 25.96    | 0.669    |
| PSNR−Y_m | 26.10 | 0.704   | 25.96    | 0.669    |
| SSIM−Y_m | 26.10 | 0.704   | 25.96    | 0.669    |
| RCAN (Zhang et al. 2018a) | 30.85 | 0.873   | 31.74    | 0.921    |
| BPP−SRx3 | 30.78 | 0.848   | 29.14    | 0.804    |

Algorithm | Set14 | BSDS100 | Urban100 | Manga109 |
|----------|-------|---------|----------|----------|
| PSNR−Y_m | 28.78 | 0.785   | 27.74    | 0.745    |
| SSIM−Y_m | 28.78 | 0.785   | 27.74    | 0.745    |
| PSNR−Y_m | 28.78 | 0.785   | 27.74    | 0.745    |
| SSIM−Y_m | 28.78 | 0.785   | 27.74    | 0.745    |
| RCAN (Zhang et al. 2018a) | 30.85 | 0.873   | 31.74    | 0.921    |
| BPP−SRx3 | 30.78 | 0.848   | 29.14    | 0.804    |

Algorithm | Set14 | BSDS100 | Urban100 | Manga109 |
|----------|-------|---------|----------|----------|
| PSNR−Y_m | 24.39 | 0.586   | 23.67    | 0.547    |
| SSIM−Y_m | 24.39 | 0.586   | 23.67    | 0.547    |
| PSNR−Y_m | 24.39 | 0.586   | 23.67    | 0.547    |
| SSIM−Y_m | 24.39 | 0.586   | 23.67    | 0.547    |
| RCAN (Zhang et al. 2018a) | 30.85 | 0.873   | 31.74    | 0.921    |
| BPP−SRx3 | 30.78 | 0.848   | 29.14    | 0.804    |

Algorithm | Set14 | BSDS100 | Urban100 | Manga109 |
|----------|-------|---------|----------|----------|
| PSNR−Y_m | 24.82 | 0.635   | 24.67    | 0.592    |
| SSIM−Y_m | 24.82 | 0.635   | 24.67    | 0.592    |
| PSNR−Y_m | 24.82 | 0.635   | 24.67    | 0.592    |
| SSIM−Y_m | 24.82 | 0.635   | 24.67    | 0.592    |
| RCAN (Zhang et al. 2018a) | 30.85 | 0.873   | 31.74    | 0.921    |
| BPP−SRx3 | 30.78 | 0.848   | 29.14    | 0.804    |
Figure 11: Extended qualitative evaluation for super-resolution.
Figure 12: Extended qualitative evaluation for Mobile-to-DSLR photo translation.
Figure 13: Extended qualitative evaluation for image dehaze for Indoor/Outdoor datasets.
Figure 14: Extended qualitative evaluation for image dehaze for Dense dataset.

Figure 15: Extended qualitative evaluation for joint HDR+SR enhancement.
**Raindrop Removal**  We use the DeRaindrop (Qian et al. 2018a,b) dataset for training and test. This dataset provides paired images, one degraded by raindrops and the other one free from raindrops. In each training batch, we take 1 patch of size $528 \times 528$. We train a BPP model using $L_1$ loss and patch size $456 \times 456$. We use Adam optimizer (Kingma and Ba 2015) with learning rate initialized to $10^{-4}$ and decreased by half every 200,000 back-propagation steps. We train the system for 10,000 epochs.

**Computing Infrastructure**
All training processes run on Linux operating system, using implementations in Python language with software packages: Numpy, Pytorch (Paszke et al. 2017), Scilab, Pillow and OpenCV. We used NVIDIA Tesla M40 (24GB) GPU for training and NVIDIA Titan–X Maxwell (12GB) for tests.

**Reproducibility**
All output images of the BPP systems obtained in our experiments can be downloaded from the following link (2.77 GB) This can be used to reproduce all quantitative evaluations in our experiments. We have provided external links to all evaluation scripts used in our evaluations.
Figure 16: Extended qualitative evaluation for raindrop removal.