Image Reconstruction with Predictive Filter Flow

Shu Kong, Charless Fowlkes

Dept. of Computer Science, University of California, Irvine

{skong2, fowlkes}@ics.uci.edu

[Project Page], [Github], [Slides], [Poster]

Abstract

We propose a simple, interpretable framework for solving a wide range of image reconstruction problems such as denoising and deconvolution. Given a corrupted input image, the model synthesizes a spatially varying linear filter which, when applied to the input image, reconstructs the desired output. The model parameters are learned using supervised or self-supervised training. We test this model on three tasks: non-uniform motion blur removal, lossy-compression artifact reduction and single image super resolution. We demonstrate that our model substantially outperforms state-of-the-art methods on all these tasks and is significantly faster than optimization-based approaches to deconvolution. Unlike models that directly predict output pixel values, the predicted filter flow is controllable and interpretable, which we demonstrate by visualizing the space of predicted filters for different tasks.[1]

1. Introduction

Real-world images are seldom perfect. Practical engineering trade-offs entail that consumer photos are often blurry due to low-light, camera shake or object motion, limited in resolution and further degraded by image compression artifacts introduced for the sake of affordable transmission and storage. Scientific applications such as microscopy or astronomy, which push the fundamental physical limitations of light, lenses and sensors, face similar challenges. Recovering high-quality images from degraded measurements has been a long-standing problem for image analysis and spans a range of tasks such as blind-image deblurring [4, 28, 13, 43], compression artifact reduction [44, 33], and single image super-resolution [39, 57].

Such image reconstruction tasks can be viewed mathematically as inverse problems [48, 22], which are typically ill-posed and massively under-constrained. Many contemporary techniques to inverse problems have focused on regularization techniques which are amenable to computational optimization. While such approaches are interpretable as Bayesian estimators with particular choice of priors, they are often computationally expensive in practice [13, 43, 2]. Alternately, data-driven methods based on training deep convolutional neural networks yield fast inference but lack interpretability and guarantees of robustness [46, 59]. In this paper, we propose a new framework called *Predictive Filter Flow* that retains interpretability and control over the resulting reconstruction while allowing fast inference. The proposed framework is directly applicable to a variety of low-level computer vision problems involving local pixel transformations.

As the name suggests, our approach is built on the notion of filter flow introduced by Seitz and Baker [42]. In filter flow pixels in a local neighborhood of the input image are linearly combined to reconstruct the pixel centered at the same location in the output image. However, unlike convolution, the filter weights are allowed to vary from one spatial location to the next. Filter flows are a flexible class of image transformations that can model a wide range of imaging effects (including optical flow, lighting changes, non-uniform blur, non-parametric distortion). The original work on filter flow [42] focused on the problem of estimating an appropriately regularized/constrained flow between a given pair of images. This yielded convex but impractically large optimization problems (e.g., hours of computation to compute a single flow). Instead of solving for an optimal filter flow, we propose to directly predict a filter flow given an input image using a convolutional neural net (CNN) to regress the filter weights. Using a CNN to directly predict a well regularized solution is orders of magnitude faster than expensive iterative optimization.

Fig. 1 provides an illustration of our overall framework. Instead of estimating the flow between a pair of input images, we focus on applications where the model predicts both the flow and the transformed image. This can be

---

[1] Due to that arxiv limits the size of files, we put high-resolution figures, as well as a manuscript with them, in the project page.
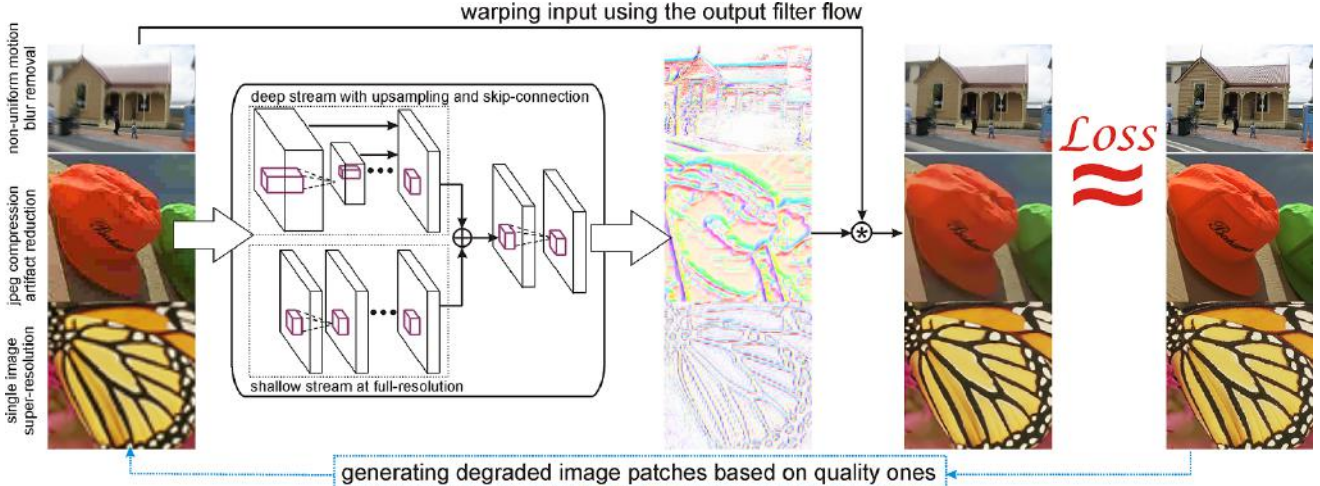
Figure 1: Overview of our proposed framework for *Predictive Filter Flow* which is readily applicable to various low-level vision problems, yielding state-of-the-art performance for non-uniform motion blur removal, compression artifact reduction and single image super-resolution. Given a corrupted input image, a two-stream CNN analyzes the image and synthesizes the weights of a spatially-varying linear filter. This filter is then applied to the input to produce a deblurred/denoised prediction. The whole framework is end-to-end trainable in a self-supervised way for tasks such as super-resolution where corrupted images can be generated automatically. The predicted filters are easily constrained for different tasks and interpretable (here visualized in the center column by the mean flow displacement, see Fig. 6).

viewed as "blind" filter flow estimation, in analogy with blind deconvolution. During training, we use a loss defined over the transformed image (rather than the predicted flow). This is closely related to so-called self-supervised techniques that learn to predict optical flow and depth from unlabeled video data [15, 16, 21]. Specifically, for the reconstruction tasks we consider such as image super-resolution, the forward degradation process can be easily simulated to generate a large quantity of training data without manual collection or annotation.

The lack of interpretability in deep image-to-image regression models makes it hard to provide guarantees of robustness in the presence of adversarial input [29], and confer reliability needed for researchers in biology and medical science [34]. Predictive filter flow differs from other CNN-based approaches in this regard since the intermediate filter flows are interpretable and transparent [50, 12, 32], providing an explicit description of how the input is transformed into output. It is also straightforward to inject constraints on the reconstruction (e.g., local brightness conservation) which would be nearly impossible to guarantee for deep image-to-image regression models.

To evaluate our model, we carry out extensive experiments on three different low-level vision tasks, non-uniform motion blur removal, JPEG compression artifact reduction and single image super-resolution. We show that our model surpasses all the state-of-the-art methods on all the three tasks. We also visualize the predicted filters which reveals filtering operators reminiscent of classic unsharp masking filters and anisotropic diffusion along boundaries.

To summarize our contribution: (1) we propose a novel,

end-to-end trainable, learning framework for solving various low-level image reconstruction tasks; (2) we show this framework is highly interpretable and controllable, enabling direct post-hoc analysis of how the reconstructed image is generated from the degraded input; (3) we show experimentally that predictive filter flow outperforms the state-of-the-art methods remarkably on the three different tasks, non-uniform motion blur removal, compression artifact reduction and single image super-resolution.

2. Related Work

Our work is inspired by filter flow [42], which is an optimization based method for finding a linear transformation relating nearby pixel values in a pair of images. By imposing additional constraints on certain structural properties of these filters, it serves as a general framework for understanding a wide variety of low-level vision problems. However, filter flow as originally formulated has some obvious shortcomings. First, it requires prior knowledge to specify a set of constraints needed to produce good results. It is not always straightforward to model or even come up with such knowledge-based constraints. Second, solving for an optimal filter flow is compute intensive; it may take up to 20 hours to compute over a pair of 500×500 images [42]. We address these by directly predicting flows from image data. We leverage predictive filter flow for targeting three specific image reconstruction tasks which can be framed as performing spatially variant filtering over local image patches.

Non-Uniform Blind Motion Blur Removal is an extremely challenging yet practically significant task of removing blur caused by object motion or camera shake on

a blurry photo. The blur kernel is unknown and may vary over the image. Recent methods estimate blur kernels locally at patch level, and adopt an optimization method for deblurring the patches [46, 2]. [53, 18, 46] leverage prior information about smooth motion by selecting from a predefine discretized set of linear blur kernels. These methods are computationally expensive as an iterative solver is required for deconvolution after estimating the blur kernel [9]; and the deep learning approach cannot generalize well to novel motion kernels [54, 46, 18, 41].

Compression Artifact Reduction is of significance as lossy image compression is ubiquitous for reducing the size of images transmitted over the web and recorded on data storage media. However, high compression rates come with visual artifacts that degrade the image quality and thus user experience. Among various compression algorithms, JPEG has become the most widely accepted standard in lossy image compression with several (non-invertible) transforms [51], i.e., downsampling and DCT quantization. Removing artifacts from jpeg compression can be viewed as a practical variant of natural image denoising problems [6, 20]. Recent methods based on deep convolutional neural networks trained to take as input the compressed image and output the denoised image directly achieve good performance [10, 47, 7].

Single Image Super-Resolution aims at recovering a high-resolution image from a single low-resolution image. This problem is inherently ill-posed as a multiplicity of solutions exists for any given low-resolution input. Many methods adopt an example-based strategy [56] requiring an optimization solver, others are based on deep convolutional neural nets [11, 30] which achieve the state-of-the-art and real-time performance. The deep learning methods take as input the low-resolution image (usually $4\times$ upsampled one using bicubic interpolation), and output the high-resolution image directly.

3. Predictive Filter Flow

Filter flow models image transformations $\mathbf{I}_1 \to \mathbf{I}_2$ as a linear mapping where each output pixel only depends on a local neighborhood of the input. Find such a flow can be framed as solving a constrained linear system

$$\mathbf{I}_2 = \mathbf{T}\mathbf{I}_1, \quad \mathbf{T} \in \mathbf{\Gamma}. \tag{1}$$

where $\mathbf{T}$ is a matrix whose rows act separately on a vectorized version of the source image $\mathbf{I}_1$. For the model 1 to make sense, $\mathbf{T} \in \mathbf{\Gamma}$ must serve as a placeholder for the entire set of additional constraints on the operator which enables a unique solution that satisfies our expectations for particular problems of interest. For example, standard convolution corresponds to $\mathbf{T}$ being a circulant matrix whose rows are cyclic permutations of a single set of filter weights

which are typically constrained to have compact localized non-zero support. For a theoretical perspective, Filter Flow model 1 is simple and elegant, but directly solving Eq. 1 is intractable for image sizes we typically encounter in practice, particularly when the filters are allowed to vary spatially.

3.1. Learning to predict flows

Instead of optimizing over $\mathbf{T}$ directly, we seek for a learnable function $f_{\mathbf{w}}(\cdot)$ parameterized by $\mathbf{w}$ that predicts the transformation $\hat{\mathbf{T}}$ specific to image $\mathbf{I}_1$ taken as input:

$$\mathbf{I}_2 \approx \hat{\mathbf{T}}\mathbf{I}_1, \quad \hat{\mathbf{T}} \equiv f_{\mathbf{w}}(\mathbf{I}_1), \tag{2}$$

We call this model Predictive Filter Flow. Manually designing such a function $f_{\mathbf{w}}(\cdot)$ isn't feasible in general, therefore we learn a specific $f_{\mathbf{w}}$ under the assumption that $\mathbf{I}_1, \mathbf{I}_2$ are drawn from some fixed joint distribution.

Given sampled image pairs, $\{(\mathbf{I}_1^i, \mathbf{I}_2^i)\}$, where $i = 1, \ldots, N$, we seek parameters $\mathbf{w}$ that minimize the difference between a recovered image $\hat{\mathbf{I}}_2$ and the real one $\mathbf{I}_2$ measured by some loss ℓ.

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \ell(\mathbf{I}_2^i - f_{\mathbf{w}}(\mathbf{I}_1^i) \cdot \mathbf{I}_1^i) + \mathcal{R}(f_{\mathbf{w}}(\mathbf{I}_1^i)),$$
$$\text{s.t. constraint on } \mathbf{w} \tag{3}$$

Note that constraints on $\mathbf{w}$ are different from constraints $\mathbf{\Gamma}$ used in Filter Flow. In practice, we enforce hard constraints via our choice of the architecture/functional form of f along with soft-constraints via additional regularization term $\mathcal{R}$. We also adopt commonly used L_2 regularization on $\mathbf{w}$ to reduce overfitting. There are a range of possible choices for measuring the difference between two images. In our experiments, we simply use the robust L_1 norm to measure the pixel-level difference.

Filter locality In principle, each pixel output $\mathbf{I}_2$ in Eq. 3 can depend on all input pixels $\mathbf{I}_2$. We introduce the structural constraint that each output pixel only depends on a corresponding local neighborhood of the input. The size of this neighborhood is thus a hyper-parameter of the model. We note that while the predicted filter flow $\hat{\mathbf{T}}$ acts locally, the estimation of the correct local flow within a patch can depend on global context captured by large receptive fields in the predictor $f_{\mathbf{w}}(\cdot)$.

In practice, this constraint is implemented by using the "im2col" operation to vectorize the local neighborhood patch centered at each pixel and compute the inner product of this vector with the corresponding predicted filter. This operation is highly optimized for available hardware architectures in most deep learning libraries and has time and space cost similar to computing a single convolution.

For example, if the filter size is 20×20, the last layer of the CNN model $f_{\mathbf{w}}(\cdot)$ outputs a three-dimensional array with a channel dimension of 400, which is comparable to feature activations at a single layer of typical CNN architectures [27, 45, 17].

Other filter constraints Various priori constraints on the filter flow $\hat{\mathbf{T}} \equiv f_{\mathbf{w}}(\mathbf{I}_1)$ can be added easily to enable better model training. For example, if smoothness is desired, an L_2 regularization on the (1st order or 2nd order) derivative of the filter flow maps can be inserted during training; if sparsity is desired, an L_1 regularization on the filter flows can be added easily. In our work, we add sum-to-one and non-negative constraints on the filters for the task of non-uniform motion blur removal, meaning that the values in each filter should be non-negative and sum-to-one by assuming there is no lighting change. This can be easily done by inserting a softmax transform across channels of the predicted filter weights. For other tasks, we simply let the model output free-form filters with no further constraints on the weights.

Self-Supervision Though the proposed framework for training Predictive Filter Flow requires paired inputs and target outputs, we note that generating training data for many reconstruction tasks can be accomplished automatically without manual labeling. Given a pool of high quality images, we can automatically generate low-resolution, blurred or JPEG degraded counterparts to use in training (see Section 4). This can also be generalized to so-called self-supervised training for predicting flows between video frames or stereo pairs.

3.2. Model Architecture and Training

Our basic framework is largely agnostic to the choice of architectures, learning method, and loss functions. In our experiments, we utilize to a two-stream architecture as shown in Fig. 1. The first stream is a simple 18-layer network with 3×3 convolutional layers, skip connections [17], pooling layers and upsampling layers; the second stream is a shallow but full-resolution network with no pooling. The first stream has larger receptive fields for estimating per-pixel filters by considering long-range contextual information, while the second stream keeps original resolution as input image without inducing spatial information loss. Batch normalization [19] is also inserted between a convolution layer and ReLU layer [38]. The Predictive Filter Flow is self-supervised so we could generate an unlimited amount of image pairs for training very large models. However, we find a light-weight architecture trained over moderate-scale training set performs quite well. Since our architecture is different from other feed-forward image-to-image regression CNNs, we also report the baseline per-

formance of the two-stream architecture trained to directly predict the reconstructed image rather than the filter coefficients.

For training, we crop 64×64-resolution patches to form a batch of size 56. Since the model adapts to patch boundary effects seen during training, at test time we apply it to non-overlapping tiles of the input image. However, we note that the model is fully convolutional so it could be trained over larger patches to avoid boundary effects and applied to arbitrary size inputs.

We use ADAM optimization method during training [24], with initial learning 0.0005 and coefficients 0.9 and 0.999 for computing running averages of gradient and its square. As for the training loss, we simply use the ℓ_1-norm loss measuring absolute difference over pixel intensities. We train our model from scratch on a single NVIDIA TITAN X GPU, and terminate after several hundred epochs[2].

4. Experiments

We evaluate the proposed Predictive Filter Flow framework (PFF) on three low-level vision tasks: non-uniform motion blur removal, JPEG compression artifact reduction and single image super-resolution. We first describe the datasets and evaluation metrics, and then compare with state-of-the-art methods on the three tasks in separate subsections, respectively.

4.1. Datasets and Metrics

We use the high-resolution images in DIV2K dataset [1] and BSDS500 training set [37] for training all our models on the three tasks. This results into a total of 1,200 training images. We evaluate each model over different datasets specific to the task. Concretely, we test our model for non-uniform motion blur removal over the dataset introduced in [2], which contains large motion blur up to 38 pixels. We evaluate over the classic LIVE1 dataset [52] for JPEG compression artifacts reduction, and Set5 [5] and Set14 [58] for single image super-resolution.

To quantitatively measure performance, we use Peak-Signal-to-Noise-Ratio (PSNR) and Structural Similarity Index (SSIM) [52] over the Y channel in YCbCr color space between the output quality image and the original image. This is a standard practice in literature for quantitatively measuring the recovered image quality.

4.2. Non-Uniform Motion Blur Removal

To train models for non-uniform motion blur removal, we generate the 64×64-resolution blurry patches from clear

---

[2]Models with early termination ($\sim$2 hours for dozens of epochs) still achieve very good performance, but top performance appears after 1–2 days training. The code and models can be found in https://github.com/aimerykong/predictive-filter-flow

Figure 2: Visual comparison of our method (*PFF*) to *CNN [Sun, et al.]* [46] and *patch-optim [Bahat, et al.]* [2] on testing images released by [2]. Please be guided with the strong edges in the filter flow maps to compare visual details in the deblurred images by different methods. Also note that the bottom two rows display images from the real-world, meaning they are not synthesized and there is no blur ground-truth for them. Best view in color and zoom-in.

Table 1: Comparison on motion blur removal over the non-uniform motion blur dataset [2]. For the two metrics, the larger value means better performance of the model.

	Moderate Blur				
metric	[55]	[46]	[2]	CNN	**PFF**
PSNR	22.88	24.14	24.87	24.51	**25.39**
SSIM	0.68	0.714	0.743	0.725	**0.786**
	Large Blur				
metric	[55]	[46]	[2]	CNN	**PFF**
PSNR	20.47	20.84	22.01	21.06	**22.30**
SSIM	0.54	0.56	0.624	0.560	**0.638**

we generate training pairs on the fly during training, so our model can deal with a wide range of motion blurs. This is advantageous over methods in [46, 2] which require a pre-defined set of blur kernels used for deconvolution through some offline algorithm.

In Table 1, we list the comparison with the state-of-the-art methods over the released test set by [2]. There are two subsets in the dataset, one with moderate motion blur and the other with large blur. We also report our CNN models based on the proposed two-stream architecture that outputs the quality images directly by taking as input the blurry ones. Our CNN model outperforms the one in [46] which trains a CNN for predicting the blur kernel over a patch, but carries out non-blind deconvolution with the estimated kernel for the final quality image. We attribute our better performance to two reasons. First, our CNN model learns a direct inverse mapping from blurry patch to its clear counterpart based on the learned image distribution, whereas [46] only estimates the blur kernel for the patch and uses

ones using random linear kernels [46], which are of size 30×30 and have motion vector with random orientation in $[0, 180°]$ degrees and random length in $[1, 30]$ pixels. We set the predicted filter size to be 17×17 so the model outputs $17 \times 17 = 289$ filter weights at each image location. Note that

an offline optimization for non-blind deblurring, resulting in some artifacts such as ringing. Second, our CNN architecture is higher fidelity than the one used in [46], as ours outputs full-resolution result and learns internally to minimize artifacts, e.g., aliasing and ringing effect.

From the table, we can see our PFF model outperforms all the other methods by a fair margin. To understand where our model performs better, we visualize the qualitative results in Fig. 2, along with the filter flow maps as output from PFF. We can't easily visualize the 289 dimensional filters. However, since the predicted weights $\hat{T}$ are positive and L_1 normalized, we can treat them as a distribution which we summarize by computing the expected flow vector

$$\begin{bmatrix} v_x(i,j) \\ v_y(i,j) \end{bmatrix} = \sum_{x,y} \hat{T}_{ij,xy} \begin{bmatrix} x-i \\ y-j \end{bmatrix}$$

where ij is a particular output pixel and xy indexes the input pixels. This can be interpreted as the optical flow (delta filter) which most closely approximates the predicted filter flow. We use the the color legend shown in top-left of Fig. 6.

The last two rows of Fig. 2 show the results over real-world blurry images for which there is no "blur-free" ground-truth. We can clearly see that images produced by PFF have less artifacts such as ringing artifacts around sharp edges [46, 2]. Interestingly, from the filter flow maps, we can see that the expected flow vectors are large near high contrast boundaries and smaller in regions that are already in sharp focus or which are uniform in color.

Although we define the filter size as 17×17, which is much smaller than the maximum shift in the largest blur (up to 30 pixels), our model still handles large motion blur and performs better than [2]. We assume it should be possible to utilize larger filter sizes but we did not observe further improvements when training models to synthesize larger per-pixel kernels. This suggests that a larger blurry dataset is needed to validate this point in future work.

We also considered an iterative variant of our model in which we feed the resulting deblurred image back as input to the model. However, we found relatively little improvement with additional iterations (results shown in the appendix). We conjecture that, although the model was trained with a wide range of blurred examples, the statistics of the transformed image from the first iteration are sufficiently different than the blurred training inputs. One solution could be inserting adversarial loss to push the model to generate more fine-grained textures (as done in [30] for image super-resolution).

4.3. JPEG Compression Artifact Reduction

Similar to training for image deblurring, we generate JPEG compressed image patches from original non-compressed ones on the fly during training. This can be
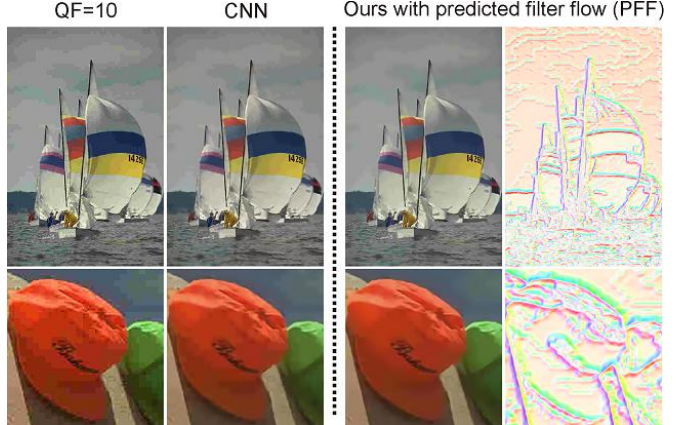

Figure 3: Visual comparison of our methods (PFF and CNN). Strong edges in the expected flow map (right) highlight areas where most apparent artifacts are removed. More results can be found in the appendix. Best viewed in color and zoomed-in.

Table 2: Comparison on JPEG compression artifact reduction over LIVE1 dataset [52]. PSNR and SSIM are used as metrics listed on two rows respectively in each macro row grid (the larger the better).

QF	JPEG	SA-DCT [14]	AR-CNN [10]	L4 [47]	CAS-CNN [7]	MWCNN [35]	**PFF**
10	27.77	28.65	29.13	29.08	29.44	29.69	**29.82**
	0.791	0.809	0.823	0.824	0.833	0.825	**0.836**
20	30.07	30.81	31.40	31.42	31.70	32.04	**32.14**
	0.868	0.878	0.890	0.890	0.895	0.889	**0.905**
40	32.35	32.99	33.63	33.77	34.10	34.45	**34.67**
	0.917	0.940	0.931	—	0.937	0.930	**0.949**

easily done using JPEG compression function by varying the quality factor (QF) of interest.

In Table 2, we list the performance of our model and compare to the state-of-the-art methods. We note that our final PFF achieves the best among all the methods. Our CNN baseline model also achieves on-par performance with state-of-the-art, though we do not show in the table, we draw the performance under the ablation study in Fig. 4. Specifically, we study how our model trained with single or a mixed QFs affect the performance when tested on image compressed with a range of different QFs. We plot the detailed performances of our CNN and PFF in terms of absolute measurements by PSNR and SSIM, and the increase in PSNR between the reconstructed and JPEG compressed image.

We can see that, though a model trained with QF=10 overfits the dataset, all the other models achieve generalizable and stable performance. Basically, a model trained on a single QF brings the largest performance gain over images compressed with the same QF. Moreover, when our model is trained with mixed quality factors, its performance
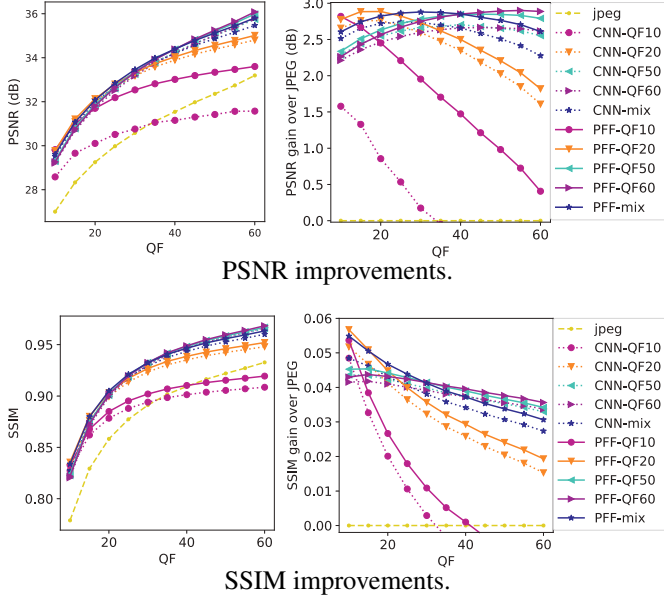
PSNR improvements.

SSIM improvements.

Figure 4: Performance vs. training data with different compression quality factors measured by PSNR and SSIM and their performance gains, over the LIVE1 dataset. The original JPEG compression is plotted for baseline.

is quite stable and competitive with quality-specific models across different compression quality factors. This indicates that our model is of practical value in real-world applications.

In Fig. 3, we demonstrate qualitative comparison between CNN and PFF. The output filter flow maps indicate from the colorful edges how the pixels are warped from the neighborhood in the input image. This also clearly shows where the JPEG image degrades most, e.g., the large sky region is quantized by JPEG compression. Though CNN makes the block effect smooth to some extent, our PFF produces the best visual quality, smoothing the block artifact while maintaining both high- and low-frequency details.

4.4. Single Image Super-Resolution

In this work, we only generate pairs to super-resolve images 4× larger. To generate training pairs, for each original image, we downsample $\frac{1}{4}\times$ and upsample 4× again using bicubic interpolation (with anti-aliasing). The 4× upsampled image from the low-resolution is the input to our model. Therefore, a super-resolution model is expected to be learned for sharpening the input image.

In Table 3, we compare our PFF model quantitatively with other methods. We can see that our model outperforms the others on both test sets. In Fig. 5, we compare visually over bicubic interpolation, CNN and PFF. We can see from the zoom-in regions that our PFF generates sharper boundaries and delivers an anti-aliasing functionality. The filter flow maps once again act as a guide, illustrating where the
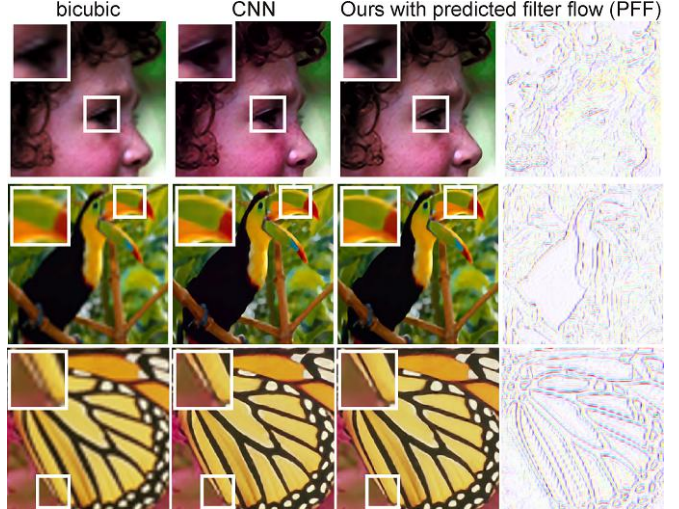

Figure 5: Visual comparison of our method (PFF) to CNN, each image is super-resolved (×4). More results can be found in the appendix. Best view in color and zoom-in.

Table 3: Comparison on single image super-resolution (×4) over the classic Set5 [5] and Set14 [58] datasets. The metrics used here are PSNR (dB) and SSIM listed as two rows, respectively.

	Bicubic	NE+LLE [8]	KK [23]	A+ [49]	SRCNN [11]	RDN+ [59]	**PFF**
Set5	28.42	29.61	29.69	30.28	30.49	32.61	**32.74**
	0.8104	0.8402	0.8419	0.8603	0.8628	0.9003	**0.9021**
Set14	26.00	26.81	26.85	27.32	27.50	28.92	**28.98**
	0.7019	0.7331	0.7352	0.7491	0.7513	0.7893	**0.7904**

smoothing happens and where sharpening happens. Especially, the filter maps demonstrate from the strong colorful edges where the pixels undergo larger transforms. In next section, we visualize the per-pixel kernels to have an indepth understanding.

5. Visualization and Analysis

We explored a number of techniques to visualize the predicted filter flows for different tasks. First, we ran k-means on predicted filters from the set of test images for each the three tasks, respectively, to cluster the kernels into K=400 groups. Then we run t-SNE [36] over the 400 mean filters to display them in the image plane, shown by the scatter plots in top row of Fig. 6. Qualitative inspection shows filters that can be interpreted as performing translation or integration along lines of different orientation (non-uniform blur), filling in high-frequency detail (jpeg artifact reduction) and deformed Laplacian-like filters (super-resolution).

We also examined the top 10 principal components of the predicted filters (shown in the second row grid in Fig. 6).
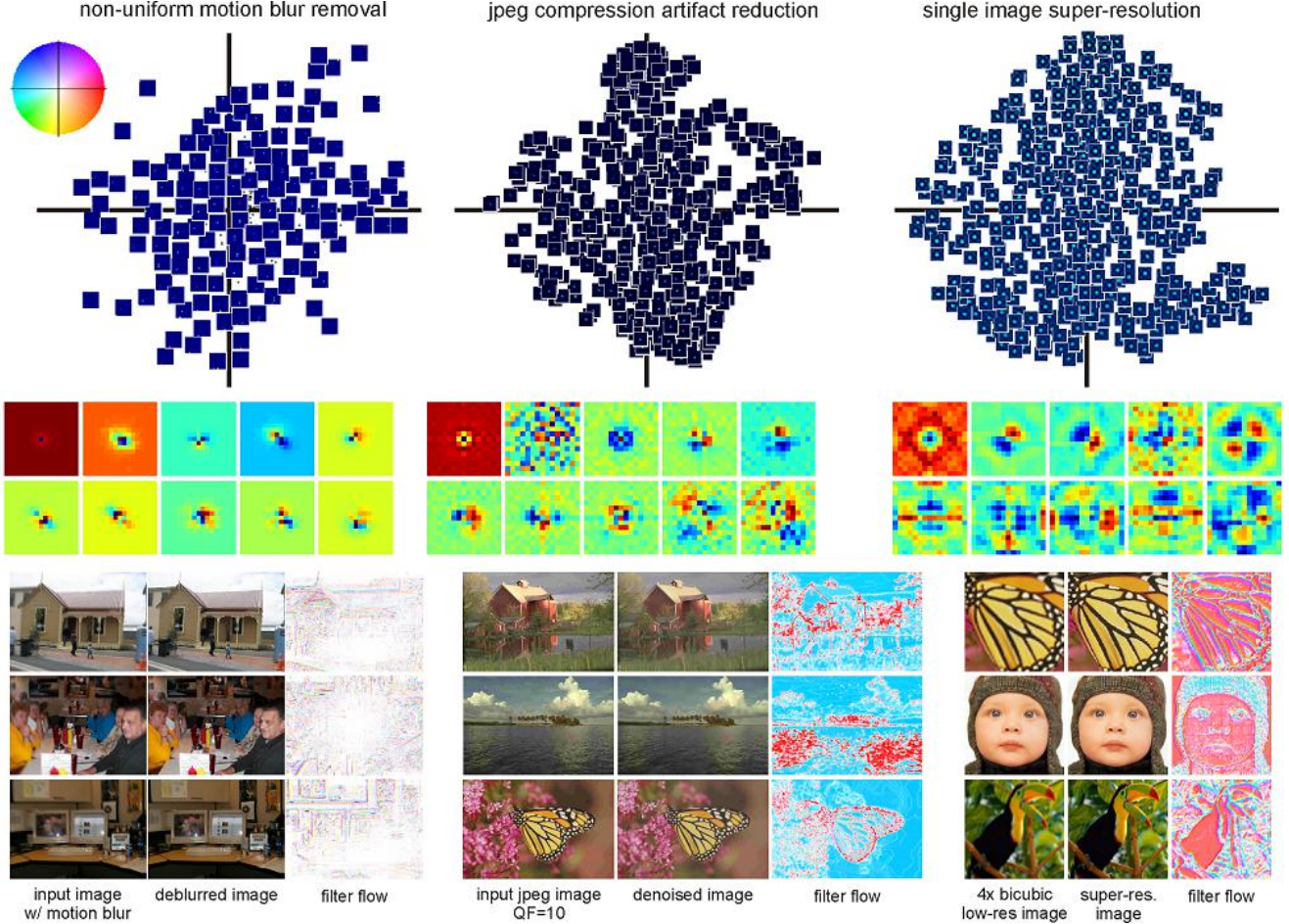
non-uniform motion blur removal   jpeg compression artifact reduction   single image super-resolution

input image w/ motion blur   deblurred image   filter flow   input jpeg image QF=10   denoised image   filter flow   4x bicubic low-res image   super-res. image   filter flow

Figure 6: Three row-wise panels: (1) We run K-means (K=400) on all filters synthesized by the model over the test set, and visualize the 400 centroid kernels using t-SNE on a 2D plane; (2) top ten principal components of the synthesized filters; (3) visualizing the color coded filter flow along with input and quality image. Each pixels filter is assigned to the nearest centroid and the color for the centroid is based on the 2D t-SNE embedding using the color chart shown at top left.

The 10D principal subspace capture 99.65%, 99.99% and 99.99% of the filter energy for non-uniform blur, artifact removal and super resolution respectively. PCA reveals smooth, symmetric harmonic structure for super-resolution with some intriguing vertical and horizontal features.

Finally, in order to summarize the spatially varying structure of the filters, we use the 2D t-SNE embedding to assign a color to each centroid (as given by the reference color chart shown top-left), and visualize the nearest centroid for the filter at each filter location in the third row grid in Fig. 6. This visualization demonstrates the filters as output by our model generally vary smoothly over the image with discontinuities along salient edges and textured regions reminiscent of anisotropic diffusion or bilateral filtering.

In summary, these visualizations provide a transparent view of how each reconstructed pixel is assembled from the degraded input image. We view this as a notable advantage over other CNN-based models which simply perform image-to-image regression. Unlike activations of intermediate layers of a CNN, linear filter weights have a well defined semantics that can be visualized and analyzed using well developed tools of linear signal processing.

6. Conclusion and Future Work

We propose a general, elegant and simple framework called Predictive Filter Flow, which has direct applications to a broad range of image reconstruction tasks. Our framework generates space-variant per-pixel filters which are easy to interpret and fast to compute at test time. Through extensive experiments over three different low-level vision tasks, we demonstrate this approach outperforms the state-of-the-art methods.

In our experiments here, we only train light-weight models over patches, However, we believe global image context is also important for these tasks and is an obvious direction for future work. For example, the global blur structure con-

veys information about camera shake; super-resolution and compression reduction can benefit from long-range interactions to reconstruct high-frequency detail (as in non-local means). Moreover, we expect that the interpretability of the output will be particularly appealing for interactive and scientific applications such as medical imaging and biological microscopy where predicted filters could be directly compared to physical models of the imaging process.

Acknowledgement

References

[1] 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. 4

[2] Y. Bahat, N. Efrat, and M. Irani. Non-uniform blind deblurring by reblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3286–3294, 2017. 1, 3, 4, 5, 6, 12, 13

[3] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 468–475. IEEE, 2017. 11

[4] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995. 1

[5] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *British Machine Vision Conference*, 2012. 4, 7

[6] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005. 3

[7] L. Cavigelli, P. Hager, and L. Benini. Cas-cnn: A deep convolutional neural network for image compression artifact suppression. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 752–759. IEEE, 2017. 3, 6

[8] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004. 7

[9] S. Cho, J. Wang, and S. Lee. Handling outliers in non-blind image deconvolution. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 495–502. IEEE, 2011. 3

[10] C. Dong, Y. Deng, C. Change Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584, 2015. 3, 6

[11] 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*, 38(2):295–307, 2016. 3, 7

[12] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017. 2

[13] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *ACM transactions on graphics (TOG)*, volume 25, pages 787–794. ACM, 2006. 1

[14] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *IEEE Transactions on Image Processing*, 16(5):1395–1411, 2007. 6

[15] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016. 2

[16] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017. 2

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[18] M. Hradiš, J. Kotera, P. Zemcík, and F. Šroubek. Convolutional neural networks for direct text deblurring. In *Proceedings of BMVC*, volume 10, page 2, 2015. 3

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4

[20] V. Jain and S. Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*, pages 769–776, 2009. 3

[21] J. Y. Jason, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision*, pages 3–10. Springer, 2016. 2

[22] J. Kaipio and E. Somersalo. *Statistical and computational inverse problems*, volume 160. Springer Science & Business Media, 2006. 1

[23] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE transactions on pattern analysis & machine intelligence*, (6):1127–1133, 2010. 7

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[25] S. Kong and C. Fowlkes. Pixel-wise attentional gating for parsimonious pixel labeling. *arXiv preprint arXiv:1805.01556*, 2018. 11

[26] S. Kong and C. Fowlkes. Recurrent scene parsing with perspective understanding in the loop. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 11

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In

Advances in neural information processing systems, pages 1097–1105, 2012. 4

[28] D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE signal processing magazine*, 13(3):43–64, 1996. 1

[29] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 2

[30] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017. 3, 6, 11

[31] K. Li, B. Hariharan, and J. Malik. Iterative instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 11

[32] Z. C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, 2018. 2

[33] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *IEEE transactions on circuits and systems for video technology*, 13(7):614–619, 2003. 1

[34] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. Van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017. 2

[35] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo. Multi-level wavelet-cnn for image restoration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[36] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 7

[37] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 416–423. IEEE, 2001. 4

[38] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 4

[39] S. C. Park, M. K. Park, and M. G. Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*, 20(3):21–36, 2003. 1

[40] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *ECCV*, 2016. 11

[41] C. Schuler, M. Hirsch, S. Harmeling, and B. Scholkopf. Learning to deblur. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):1–1, 2016. 3

[42] S. M. Seitz and S. Baker. Filter flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 1, 2

[43] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *Acm transactions on graphics (tog)*, volume 27, page 73. ACM, 2008. 1

[44] M.-Y. Shen and C.-C. J. Kuo. Review of postprocessing techniques for compression artifact removal. *Journal of visual communication and image representation*, 9(1):2–14, 1998. 1

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

[46] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 769–777, 2015. 1, 3, 4, 5, 6, 12, 13

[47] P. Svoboda, M. Hradis, D. Barina, and P. Zemcik. Compression artifacts removal using convolutional neural networks. *arXiv preprint arXiv:1605.00366*, 2016. 3, 6

[48] A. Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005. 1

[49] R. Timofte, V. De Smet, and L. Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1920–1927, 2013. 7

[50] A. Vellido, J. D. Martín-Guerrero, and P. J. Lisboa. Making machine learning models interpretable. In *ESANN*, volume 12, pages 163–172. Citeseer, 2012. 2

[51] G. K. Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 3

[52] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 4, 6

[53] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *International journal of computer vision*, 98(2):168–186, 2012. 3

[54] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, pages 1790–1798, 2014. 3

[55] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1107–1114, 2013. 5, 12

[56] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. In *European Conference on Computer Vision*, pages 372–386. Springer, 2014. 3

[57] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010. 1

[58] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, pages 711–730. Springer, 2010. 4, 7

[59] 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. 1, 7

Appendix

In the supplementary material, we first show more visualizations to understand the predicted filter flows, then show if it is possible to refine the results by iteratively feeding deblurred image to the same model for the task of non-uniform motion blur removal. We finally present more qualitative results for all the three tasks studied in this paper.

1. Visualization of Per-Pixel Loading Factors

As a supplementary visualization to the principal components by PCA shown in the main paper, we can also visualize the per-pixel loading factors corresponding to each principal component. We run PCA over testing set and show the first six principal components and the corresponding per-pixel loading factors as a heatmap in Figure 7. With this visualization technique, we can know what region has higher response to which component kernels. Moreover, given that the first ten principal components capture $\geq 99\%$ filter energy (stated in the main paper), we expect future work to predict compact per-pixel filters using low-rank technique, which allows for incorporating long-range pixels through large predictive filters while with compact features (thus memory consumption is reduced largely).

2. Iteratively Removing Motion Blur

As the deblurred images are still not perfect, we are interested in studying if we can improve performance by iteratively running the model, i.e., feeding the deblurred image as input to the same model one more time to get the result. We denote this method as PFF+1. Not much surprisingly, we do not observe further improvement as listed in Figure 4, instead, such a practice even hurts performance slightly. The qualitative results are shown in Figure 8, from which we can see the second run does not generate much change through the filter flow maps. We believe the reason is that, the deblurred images have different statistics from the original blurry input, and the model is not trained with such deblurred images. Therefore, it suggests two natural directions as future work for improving the results, 1) training explicitly with recurrent loops with multiple losses to improve the performance, similar to [3, 31, 40, 26, 25], or 2) simultaneously inserting an adversarial loss to force the model to hallucinate details for realistic output, which can be useful in practice as done in [30].

3. More Qualitative Results

In Figure 9, 10 and 11, we show more qualitative results for non-uniform motion blur removal, JPEG compression artifact reduction and single image super-resolution, respectively. From these comparisons and with the guide of filter
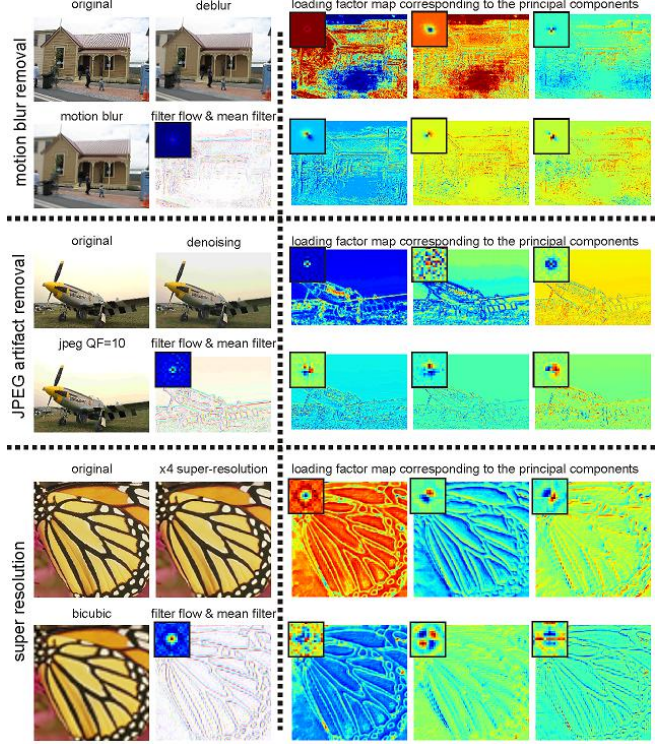

Figure 7: We show the original image, low-quality input and the high-quality output by our model as well as the mean kernel and filter flow maps on the left panel, and the first six principal components and the corresponding loading factors as heatmap on the right panel. Best seen in color and zoom-in.

flow maps, we can see at what regions our PFF pays attention to and how it outperforms the other methods.

Table 4: Comparison on motion blur removal over the non-uniform motion blur dataset [2]. PFF+1 means we perform PFF one more time by taking as input the deblurred image by the same model.

metric	Moderate Blur					
	[55]	[46]	[2]	CNN	**PFF**	PFF+1
PSNR	22.88	24.14	24.87	24.51	**25.39**	25.28
SSIM	0.68	0.714	0.743	0.725	**0.786**	0.783
metric	Large Blur					
	[55]	[46]	[2]	CNN	**PFF**	PFF+1
PSNR	20.47	20.84	22.01	21.06	**22.30**	22.21
SSIM	0.54	0.56	0.624	0.560	**0.638**	0.633

Figure 8: We show deblurring results over some random testing images from the dataset released by [2]. We first feed the blurry images to PFF model, and obtain deblurred images; then we feed such deblurred images into the same PFF model again to see if this iterative practice refines the output. However, through the visualization that iteratively running the model changes very little as seen from the second filter flow maps. This helps qualitatively explain why iteratively running the model does not improve deblurring performance further.
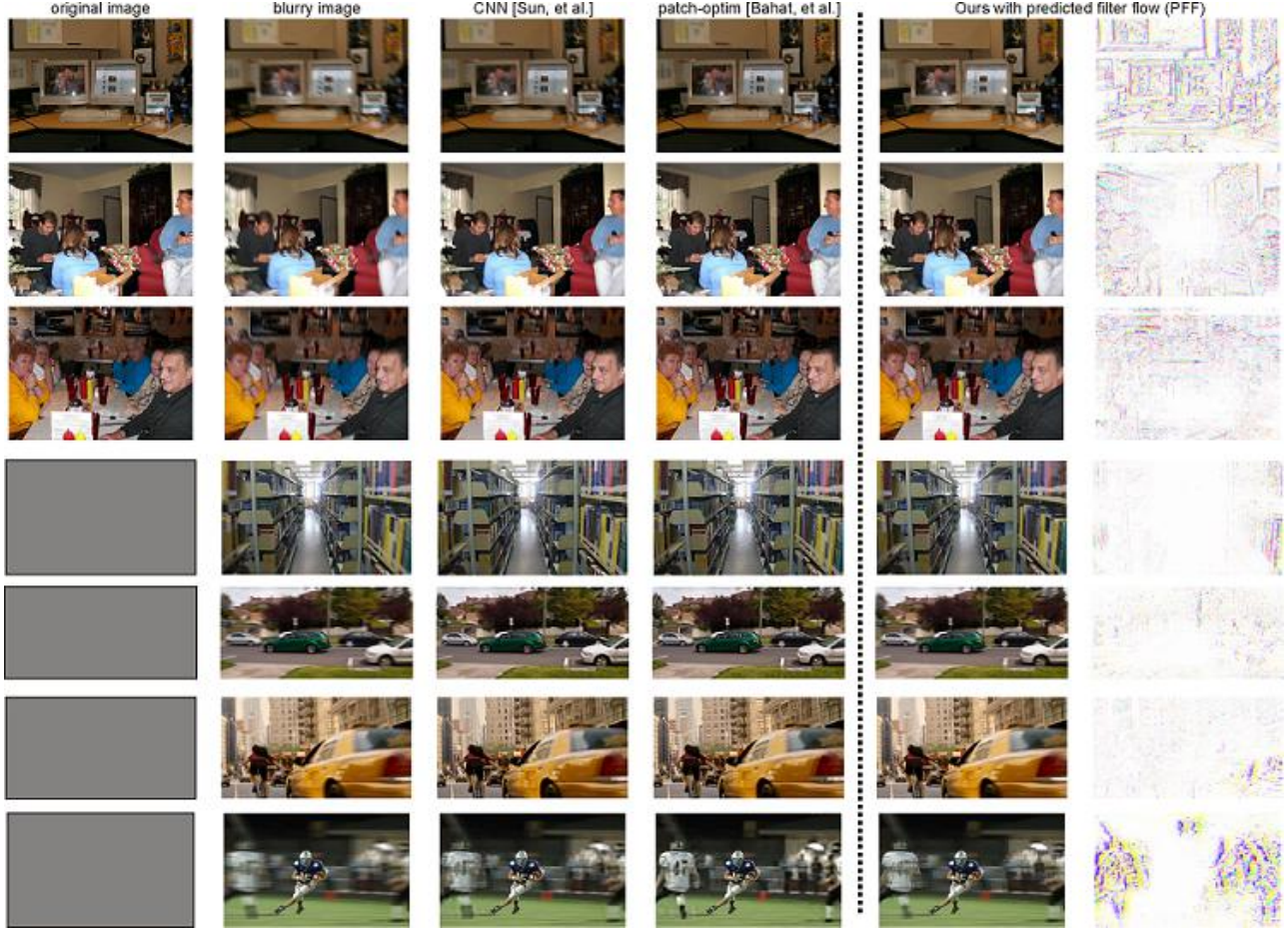
Figure 9: Visual comparison of our method (*PFF*) to *CNN [Sun, et al.]* [46] and *patch-optim [Bahat, et al.]* [2] on more testing images released by [2]. Please be guided with the strong edges in the filter flow maps to compare visual details in the deblurred images by different methods. The last four rows show real-world blurry images without "ground-truth" blur. Note that for the last image, there is very large blur caused by the motion of football players. As our model is not trained on larger kernels which should be able to cover the size of blur, it does not perform as well as *patch-optim [Bahat, et al.]* [2]. But it is clear that our model generates sharp edges in this task. Best view in color and zoom-in.
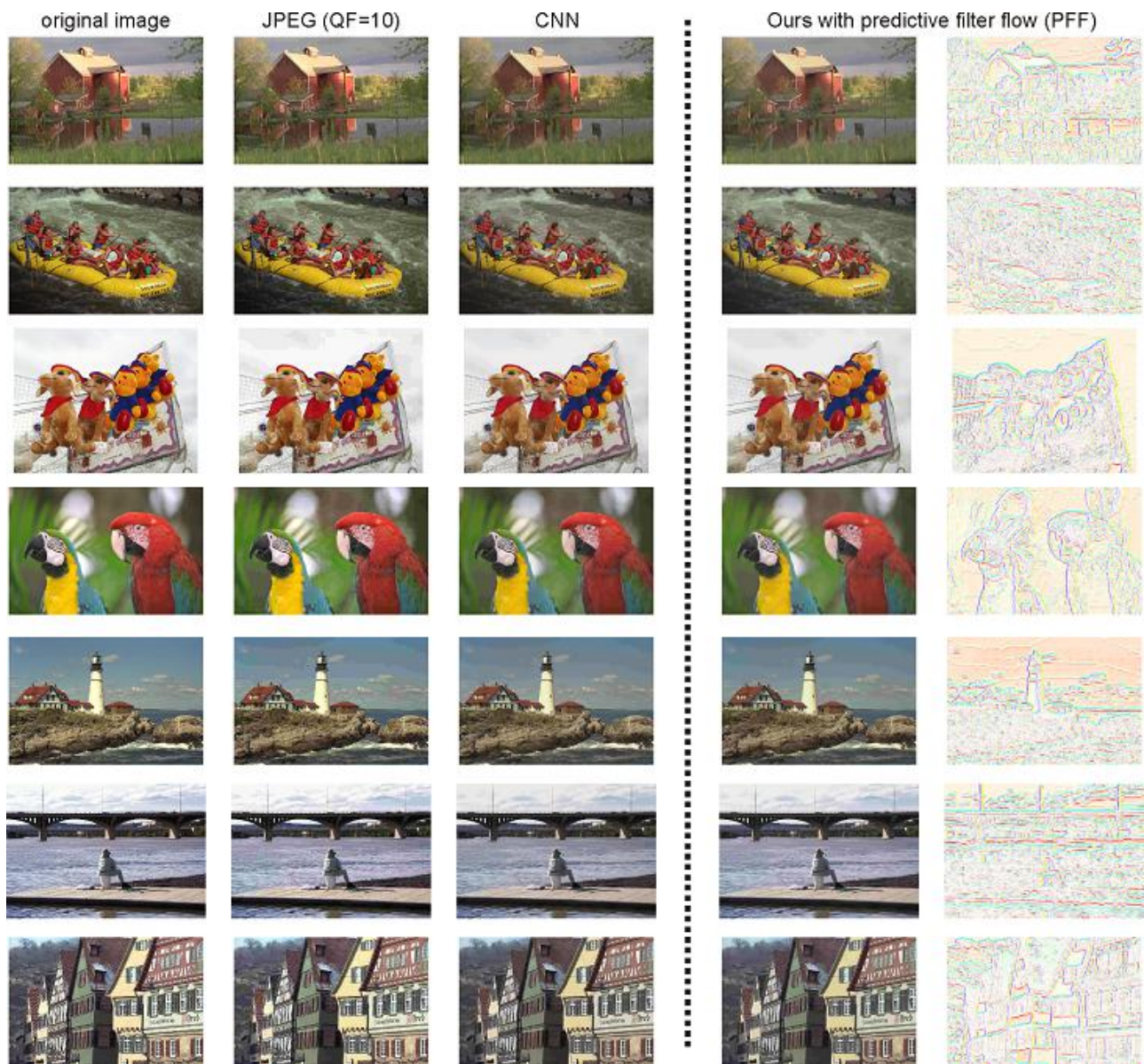
Figure 10: Visual comparison between CNN and our method (*PFF*) for JPEG compression artifact reduction. Here we compress the original images using JPEG method with quality factor (QF) as 10. Best view in color and zoom-in.
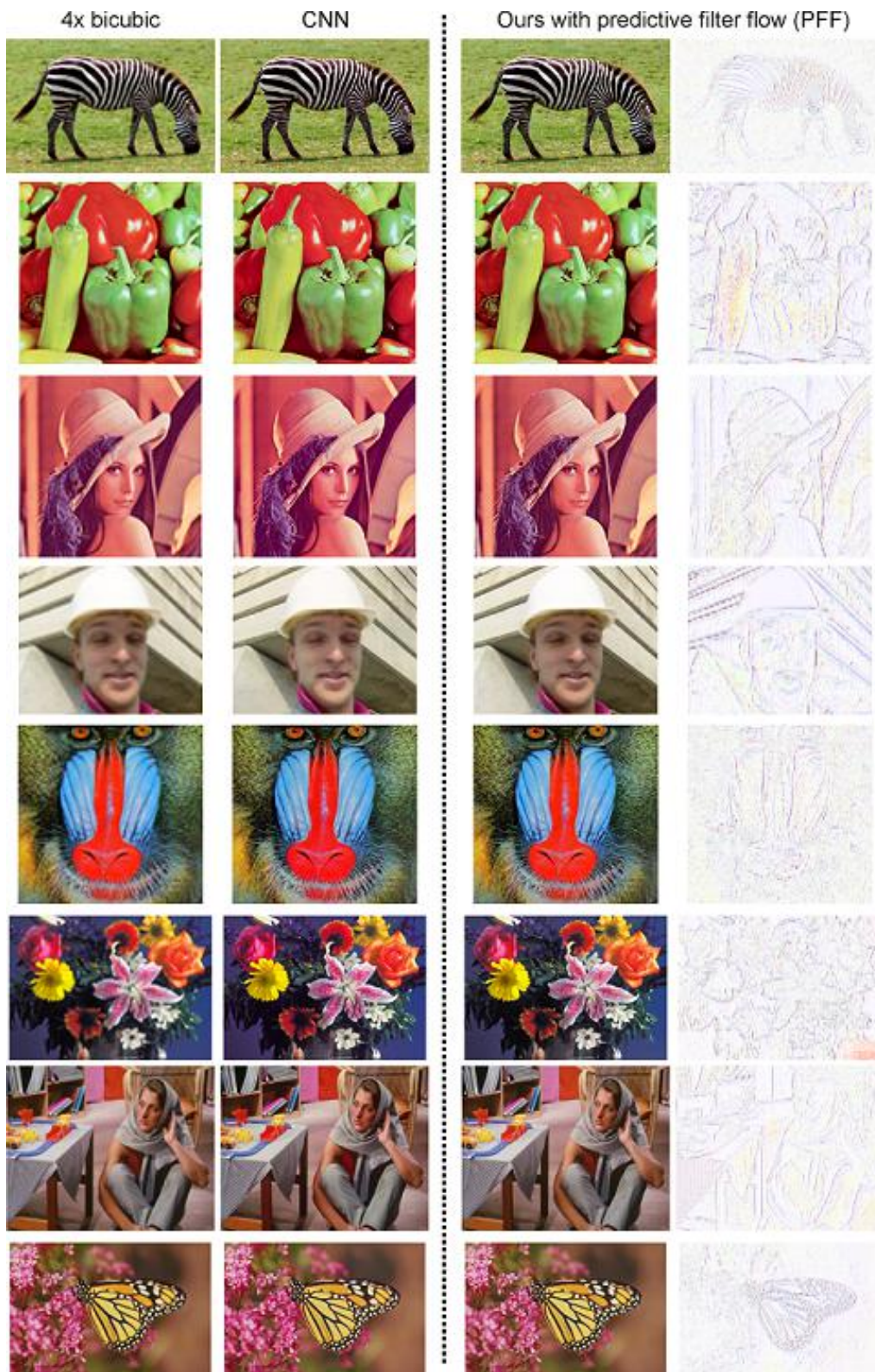
Figure 11: Visual comparison between CNN and our method (*PFF*) for single image super-resolution. Here all images are super-resolved by $4\times$ larger. We show in the first column the results by bicubic interpolation. Best view in color and zoom-in.