Decouple Learning for Parameterized Image Operators (*Supplementary Material*)

1 Outline

In the supplementary material, we provide more detailed analysis and results that are not included in the main paper due to space limitations.

- In **Section** 2, we explore more ablation study to discuss the performance of our proposed framework.
- In **Section** 3, we display more results that aren't included in the main paper due to space limitation.
- In **Section** 4, we provide more analysis and deeper understanding of our learned convolution kernels.
- In **Section** 5, we describe the detailed architecture of our *base* network.

2 More Ablation Study

To justify the effectiveness of our *weight learning* network, we compare our framework with the naive baseline which contains only the *base* network. Then more complicated network structures of the *weight learning* network are experimented with to explore the potential power of the proposed framework. Finally we demonstrate the interpolation ability of the proposed framework on unseen parameter values.

2.1 Comparison with Baseline

We compare our proposed framework with one naive approach that employs only the *base* network with additional input channels as [1], which indicates the parameter values and image operators separately. Each additional channel is occupied with a single value.

The results are shown in Table 1, which trains three different image operators including both image filtering and restoration tasks together. We can see that the baseline from [1] lags behind us on all the different parameter settings across all three image operators. The potential reason for this phenomenon could be that it is difficult to learn unified convolution weights to be suitable for tasks different in both goals and implementations. By contrast, the convolutional weights of our *base* network are adaptively changed for different tasks and input parameters. Theoretically speaking, our learned network should have the capability to express the base network with numerous possibilities.

Table 1: Numerical comparison (PSNR) between our proposed framework and the baseline [1] that are all jointly trained over three image operators.

RTV			Super resolution			Denoising		
λ	ours	baseline	s	ours	baseline	σ	ours	baseline
0.002	**36.60**	35.11	2	**30.83**	30.59	15	**30.90**	30.67
0.004	**37.14**	35.25	3	**28.30**	28.13	25	**28.63**	28.50
0.010	**37.18**	35.01	4	**26.95**	26.82	50	**25.66**	25.57
0.022	**36.83**	34.89						
0.050	**35.60**	32.75						
ave.	**36.67**	34.60	ave.	**28.69**	28.51	ave.	**28.39**	28.24

Table 2: Quantitative evaluation of a few variants of the proposed network trained on the WLS filter [2]. We experiment separately by training only the base network on a fixed single parameter value ("single (base)"), extending the weight learning network to two or more fully connected layers trained on numerous random parameter values with ("nume. (fc2R)") and without ("nume. (fc2)") ReLU layers inbetween.

λ	single		single (base)		nume.		nume. (fc2R)		nume. (fc2)	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.100	44.00	0.994	44.16	0.994	42.12	0.993	42.02	0.992	42.36	0.993
0.215	43.14	0.993	43.08	0.993	42.64	0.993	42.23	0.993	42.43	0.993
1.000	41.93	0.992	40.61	0.991	41.63	0.991	40.25	0.991	40.28	0.991
4.641	39.42	0.987	38.01	0.988	39.64	0.989	37.31	0.988	37.35	0.988
10.00	39.13	0.986	36.83	0.986	38.51	0.987	36.00	0.985	35.94	0.986
ave	41.52	0.990	40.54	0.990	40.91	0.990	39.56	0.990	39.67	0.990

Table 3: Quantitative evaluation of our proposed framework trained only with a set of fixed parameter values ("various (fixed)") on the L_0 smoother [3]. The parameters used for training our framework are taken from the 5 non-boldface parameters between [0.002, 0.2] in the table. The extra 4 parameters with boldface are only used in the test stage. The absolute difference between the network trained on a single parameter ("single") and various fixed parameters ("various (fixed)") is displayed in the bottom.

		0.0020	**0.0025**	**0.0033**	0.0043	0.0200	0.0928	**0.1200**	**0.1600**	0.2000	average
single	PSNR	40.69	40.19	39.77	38.96	36.07	33.08	31.78	31.13	31.75	35.94
	SSIM	0.989	0.987	0.986	0.986	0.982	0.977	0.973	0.972	0.973	0.981
various (fixed)	PSNR	39.61	39.33	38.95	38.51	35.37	31.80	31.40	30.69	30.54	35.13
	SSIM	0.988	0.987	0.986	0.986	0.979	0.972	0.972	0.971	0.970	0.979
diff	PSNR	1.08	**0.86**	**0.82**	0.45	0.7	1.28	**0.38**	**0.44**	1.21	0.81
	SSIM	0.001	**0**	**0**	0	0.003	0.005	**0.001**	**0.001**	0.003	0.002

2.2 Analysis of more variants of our proposed network

Training the *base* network only Since training a fully convolutional network alone has been employed frequently by previous image processing papers [4, 5, 1], which presents a strong baseline, we experiment with this alternative ("nume. (base) ") which only leverages the *base* network by training it on one specific parameter configuration. As show in Table 2, our proposed framework ("single") achieves even better performance than the baseline under the PSNR error metric.

Training with more fc layers We also try a deeper *weight learning* network with more fully connected layers. Here we simply add one more fully connected layer to the default *weight learning* network, and demonstrate the performance of its two variants with ("nume. (fc2R)") and without ("nume. (fc2)") ReLU between these two layers respectively. As shown in Table 2, they all achieve comparable performance with that of single layer ("nume.") used in our paper. The potential reason for this phenomenon is that this one-layer *weight learning* network is sufficient for adaptively learning various parameter settings, while adding more weights/complexity to the network does not contribute to the performance much.

2.3 Interpolation ability of the proposed framework on unseen parameters.

Since the *weight learning* network contains a single fully connected layer with no non-linear activation layers, the predicted convolution weights should be a linear transformation of the input parameters. Given such a fact, any convolutional kernels of a specific parameter should be the linear interpolation of the other two. Hence we are curious about the interpolation ability of the proposed framework.

To verify such a property, we train the network using only a few fixed parameter values, which corresponds to the 5 non-boldface parameter values in Table 3. But in the test stage, we use another 4 parameter values (boldface in Table 3) that have not been seen by the network in the training stage but are between the lower and upper bound of its parameter range. As shown in Table 3, we can see that the network performs similarly to the one trained with only one parameter value, but more importantly for the interpolated boldface parameter values that the network does not recognize, it also surprisingly achieves very comparable results.

This means that a few parameter values are already sufficient for learning a good linear transformation in the *weight learning* network from input parameters to convolution weights. However, as in a real scenario, the number of such fixed training parameters is usually difficult to decide, due to many different parameter ranges of image operators. As a result, we choose to sample random parameter values instead of only a few of them for training in our paper.

3 Complement to Table 1 in the main paper

In Table 4, we show the results on two more image filters that are not included in Table 1 of the main paper due to space limitations.

Table 4: Quantitative absolute difference between the network trained with a *single* parameter value and *numerous* random values on two other filters (RGF [6] and WMF [7]).

		RGF				WMF		
metric	λ	single	nume.	diff	λ	single	nume.	diff
PSNR	1.00	41.77	37.03	4.74	1.00	39.06	36.79	2.27
	3.25	38.36	38.27	0.09	3.25	39.78	38.76	1.02
	5.50	38.11	38.35	0.24	5.50	39.94	38.53	1.41
	7.75	37.65	37.99	0.34	7.75	40.06	39.20	0.86
	10.0	37.52	37.08	0.44	10.0	39.49	38.72	0.77
	ave.	38.68	37.74	**0.93**	ave.	39.66	38.40	**1.26**
SSIM	1.00	0.994	0.981	0.013	1.00	0.985	0.972	0.013
	3.25	0.986	0.986	0	3.25	0.985	0.979	0.006
	5.50	0.985	0.986	0.001	5.50	0.986	0.981	0.005
	7.75	0.984	0.985	0.001	7.75	0.986	0.985	0.001
	10.0	0.984	0.982	0.002	10.0	0.986	0.984	0.002
	ave.	0.986	0.984	**0.002**	ave.	0.985	0.980	**0.005**

4 More analysis

In this section, we provide a more detailed and deeper understanding of the convolutional weights learned by our proposed framework.

Deeper understanding of the weight learning network One interesting phenomenon worthy of discussion is the surprising potential of the proposed framework for jointly training multiple different operators within one single network. We use the L_0 and WLS operators for jointly training as an example. With the similar strategy used in training multiple operators, besides the original parameter (γ_1) of these two operators, an extra discrete task id (γ_2, e.g.,"1","2") is input into the *weight learning* network $\mathcal{N}_{weight}$ to tell the network which task should be used. The most interesting thing is that how the proposed network can differentiate these two tasks when the same value of γ_1 is adopted for L_0 and WLS. In these cases, the difference of the learned weights for these two operators is only from the term $\gamma_2 A_{i2}$ because A_{i1} is shared. In fact, the weight difference is just a certain value A_{i2} $(2 * A_{i2} - A_{i2})$, and is the proposed framework still able to separate the target solution space for L0 and WLS? The answer is "yes". To verify it, given the jointly trained model, we set the task id of L_0 as "1" (correct) and "2"(incorrect) then test the performance relative to the original L_0 smoothing ground truth respectively. The final PSNR value is $34\ vs\ 26$. In some sense, this simple experiment not only demonstrates the great task separating ability of our proposed framework, but the diversity of solution space of the base network.

For better theoretical understanding, we finally link our proposed framework with evolutionary computing, where it is often difficult to directly get a optimal solution in a large search space. To help the searching process, some structure constraints are added for the target weight space, like Discrete Cosine Transform (DCT) used in [8]. Similarly, for our task, the weight space of the *base* network $\mathcal{N}_{base}$ is also very large, and there may exists many different parameter solutions (local minimas) which can obtain the same/similar

performance. When the *weight learning* network $\mathcal{N}_{weight}$ follows the linear formulation, like DCT used in [8], it is an extra constraint, which constrains that the weight spaces of different parameters $\overrightarrow{\gamma}$ are related with affine transform.

Table 5: Comparison between the statistics of convolution kernels learned with random parameter values and a single parameter value. The numbers are generated based on the WLS filter while λ equals to 10.

layer index	1	2	3	4	5	6	7	8	9	10
correlation	0.005	-0.001	0.008	0.004	-0.005	0.007	0.008	0.010	-0.002	-0.012
mean (various)	0.219	0.066	0.004	-0.069	-0.227	-0.183	-0.214	-0.131	-0.152	-0.141
mean (fixed)	0.542	0.801	0.604	-0.571	-2.090	-0.685	-0.520	-1.638	-1.604	-1.326
variance (various)	15.514	12.166	15.511	18.231	17.676	20.408	16.578	19.420	16.447	18.292
variance (fixed)	542.28	373.70	490.53	419.55	482.71	559.00	532.14	505.67	471.50	437.69
layer index	11	12	13	14	15	16	17	18	19	20
correlation	0.011	0.184	-0.014	0.001	0.012	0.024	0.011	-0.001	-0.071	-0.017
mean (various)	-0.296	-0.168	-0.084	-0.310	-0.137	-0.201	0.003	-0.239	-0.407	0.634
mean (fixed)	-1.117	-1.134	-2.417	-1.437	-1.043	-1.977	-0.596	1.898	-3.203	3.358
variance (various)	17.704	20.141	18.767	21.080	28.461	23.795	19.475	14.592	13.703	4.220
variance (fixed)	549.51	500.61	585.75	507.40	792.44	588.06	581.66	501.08	487.62	140.33

Difference between convolutional kernels for jointly trained network and solely trained network To analyse the difference of the convolution weights between networks jointly trained on various random parameter values ("various") and a single parameter value ("fixed"), we compute their correlation coefficient, individual mean and variance for each layer as shown in Table 5.

As can be seen, the correlation coefficient is almost 0 everywhere, which means there is no linear relationship between the two groups of convolution kernels. The absolute mean and variance of jointly trained network is also significantly larger than that of the solely trained network. Therefore, in each way, their learned convolution weights are very different from each other, even if the learned smoothing effect is almost the same (PSNR/SSIM: 35.51dB/0.983 (various) *vs.* 35.83dB/0.982 (fixed)).

The above experiments have verified the fact that the solution space of an image processing task could be huge in the form of learned convolution kernels. Two exact same results can be represented by very different convolution weights.

5 Detailed Network Structure

The detailed network structure of the *base* network is shown in Table 6.

Table 6: The detailed network structure of the *base* network. "conv", "inst", "deconv","relu" represent the convolution layer, instance normalization layer, deconvolution layer and ReLU activation respectively. Each "residual block" consists of two convolution layers, both followed by instance normalization and relu.

Name	Kernel	Stride	Dilation.	Ch I/O	InpRes	OutRes
conv1 + inst + relu	3×3	1	1	$4/64$	448×448	448×448
conv2 + inst + relu	3×3	1	1	$64/64$	448×448	448×448
conv3 + inst + relu	3×3	2	1	$64/64$	448×448	224×224
residual block1	3×3	1	2	$64/64$	224×224	224×224
residual block2	3×3	1	4	$64/64$	224×224	224×224
residual block3	3×3	1	4	$64/64$	224×224	224×224
residual block4	3×3	1	8	$64/64$	224×224	224×224
residual block5	3×3	1	8	$64/64$	224×224	224×224
residual block6	3×3	1	16	$64/64$	224×224	224×224
residual block7	3×3	1	1	$64/64$	224×224	224×224
deconv4 + inst + relu	4×4	2	1	$64/64$	224×224	448×448
conv5 + inst + relu	3×3	1	1	$64/64$	448×448	448×448
conv6	3×3	1	1	$64/3$	448×448	448×448

References

[1] Chen, Q., Xu, J., Koltun, V.: Fast image processing with fully-convolutional networks. In: IEEE International Conference on Computer Vision. Volume 9. (2017) 1, 2, 3

[2] Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. In: ACM Transactions on Graphics (TOG). Volume 27., ACM (2008) 67 2

[3] Xu, L., Lu, C., Xu, Y., Jia, J.: Image smoothing via l 0 gradient minimization. In: ACM Transactions on Graphics (TOG). Volume 30., ACM (2011) 174 2

[4] Liu, S., Pan, J., Yang, M.H.: Learning recursive filters for low-level vision via a hybrid neural network. In: European Conference on Computer Vision, Springer (2016) 560–576 3

[5] Fan, Q., Yang, J., Hua, G., Chen, B., Wipf, D.: A generic deep architecture for single image reflection removal and image smoothing. In: Proceedings of the 16th International Conference on Computer Vision (ICCV). (2017) 3238–3247 3

[6] Zhang, Q., Shen, X., Xu, L., Jia, J.: Rolling guidance filter. In: European conference on computer vision, Springer (2014) 815–830 4

[7] Zhang, Q., Xu, L., Jia, J.: 100+ times faster weighted median filter (wmf). In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 2830–2837 4

[8] Koutnik, J., Gomez, F., Schmidhuber, J.: Evolving neural networks in compressed weight space. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation, ACM (2010) 619–626 4, 5