Denoising Monte Carlo Images with Machine Learning

Team 5: Cheolmin Lee, Minki Jo, Nick Heppert

INDEX

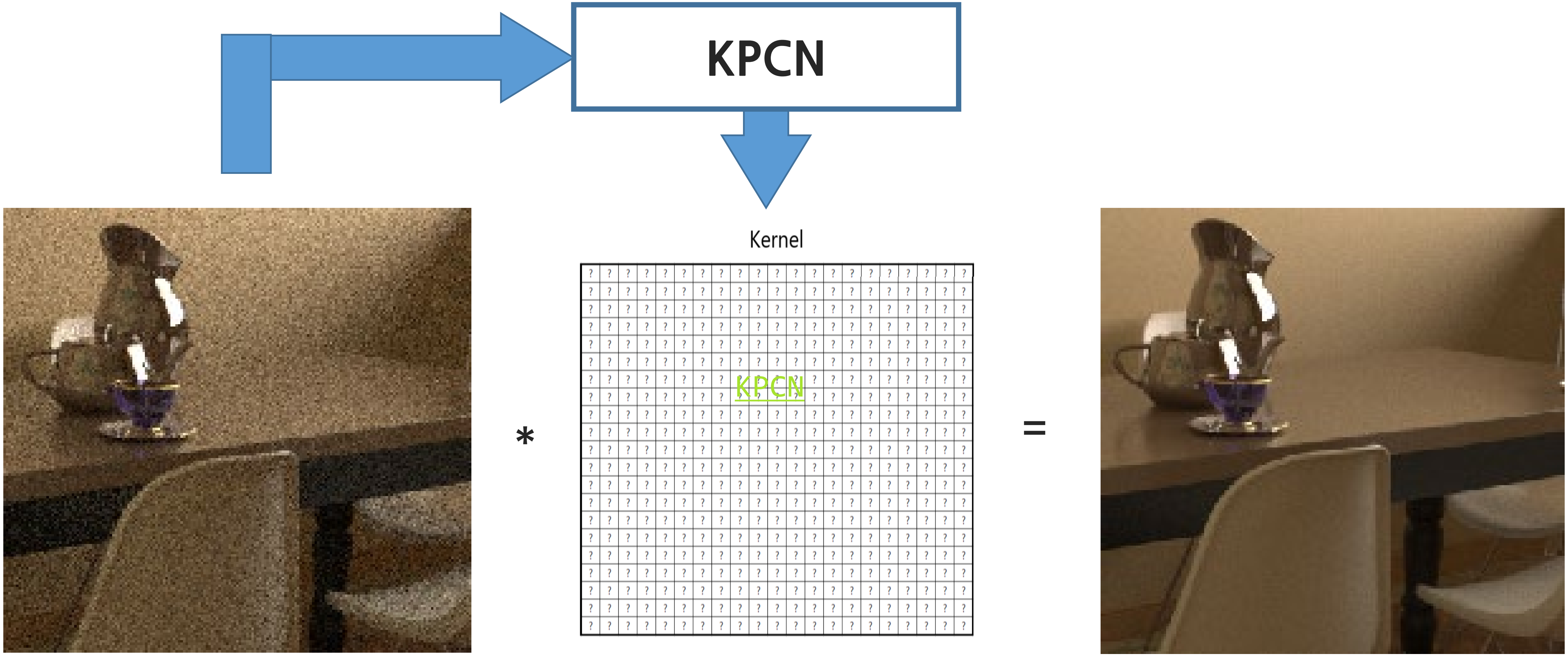
1 Introduction

2 Approach

3 Discussion

Introduction

Review



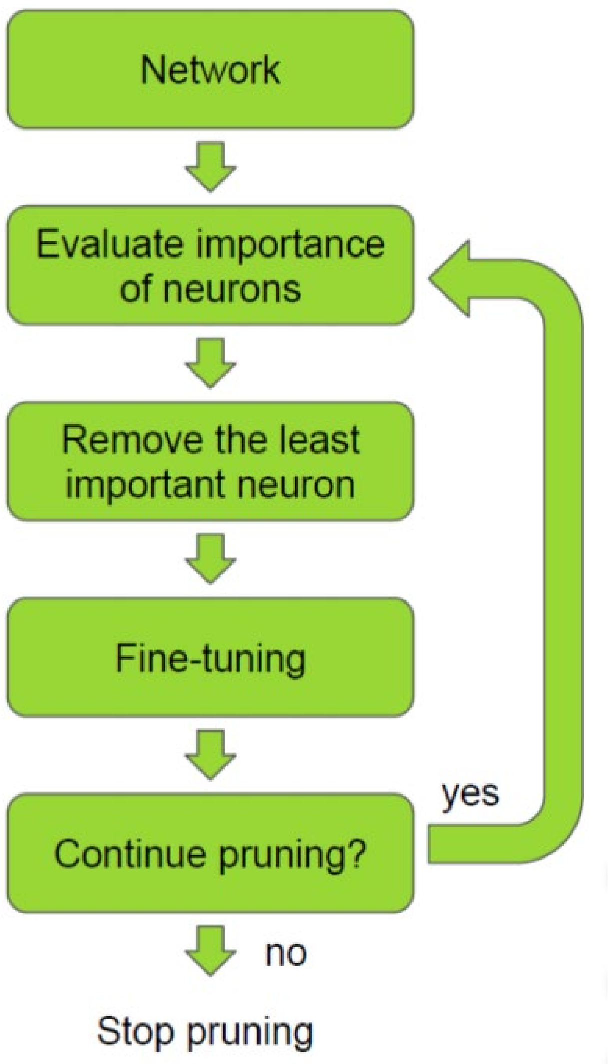
Noisy Image * Kernel = Clear Image

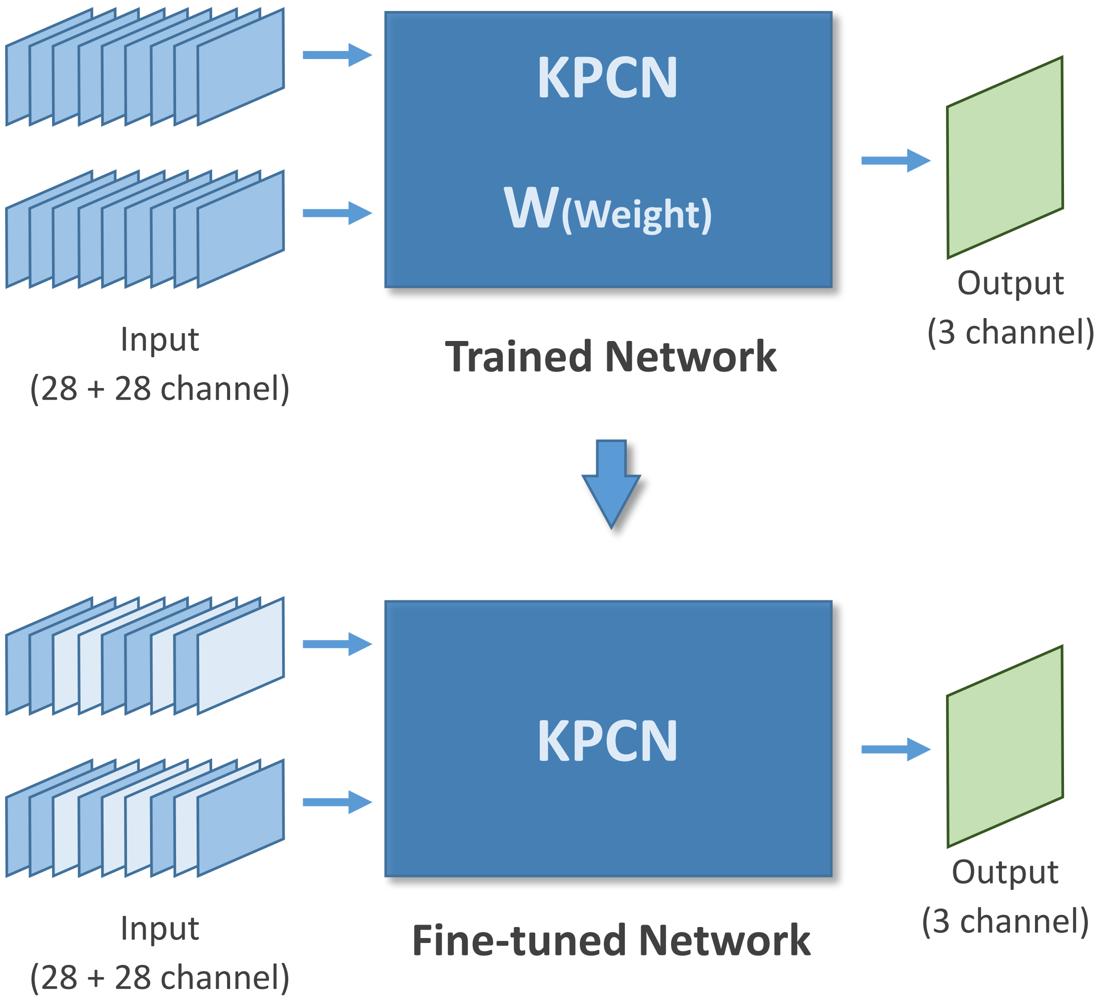
Changing Subject

- Previous Subject - Channel Pruning
  - Reduce the number of channels
- Current Subject - Modify Network Structure
  - Improve performance of denoising

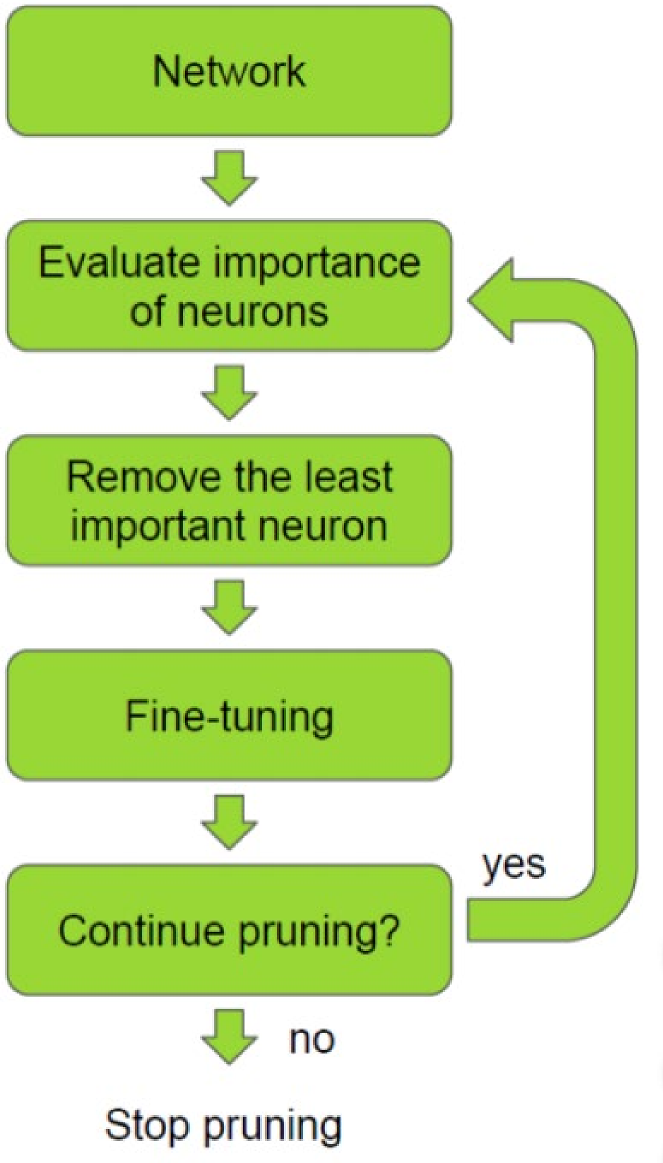
Review – Channel Pruning

Learning Algorithm





Problem of Channel Pruning



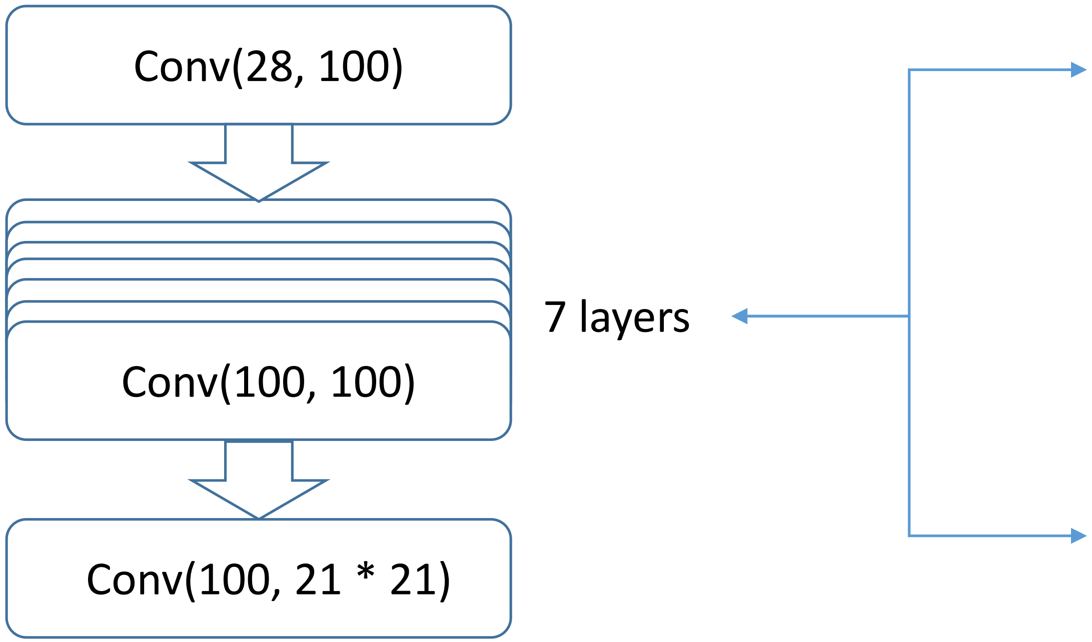
1 loop = 1day

- Learning iteration takes too long time
- 1 day per 1 loop
- Almost 20 days are required to complete one experiment

Prunning은 학습을 20회정도 반복해야 하는데, 1회 학습이 안정적으로 수렴하기까지 1일 이상이 소요되어 현실적으로 불가능 하다고 판단함.

Modify Network Structure

KPCN structure



SENet structure

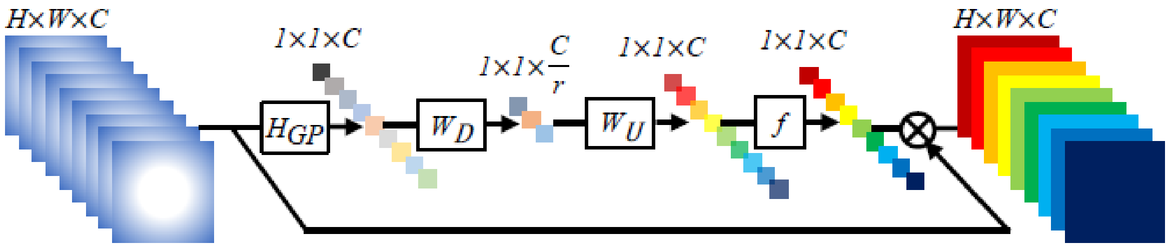
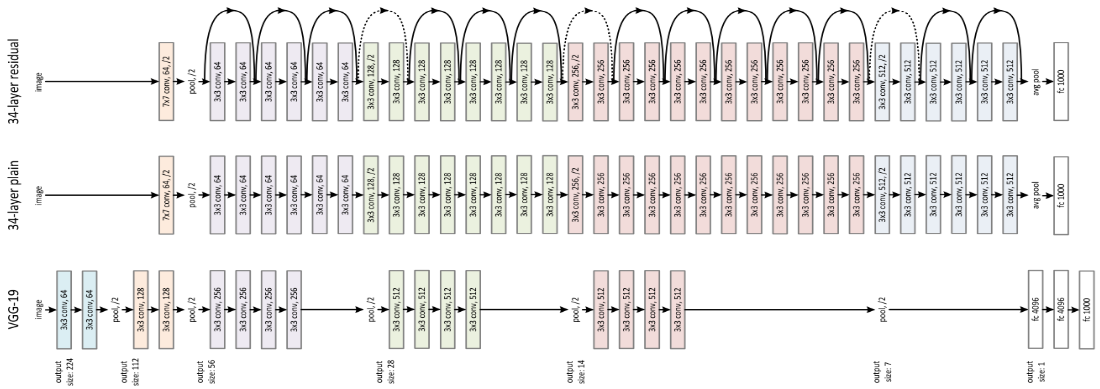


Fig. 3. Channel attention (CA). $\otimes$ denotes element-wise product

ResNet structure



[Zhang et. al. 18] Image Super-Resolution Using Very Deep Residual Channel Attention Networks, ECCV2018

[He et. al. 16] Deep Residual Learning for Image Recognition, CVPR2016

Approach
&
Experiments

Road to Baseline

1. In-Official Re-Implementation
2. No dataset
3. Patch storage
4. Large-scale training
5. Instability of loss
    a. Skip batch with NaN kernel
    b. Patches with infinity
6. Finally Baseline

Baseline Results

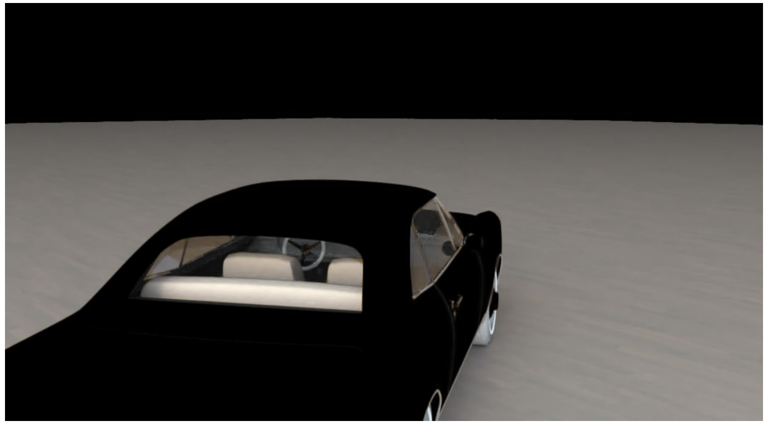
Specular Network (s)

Prediction

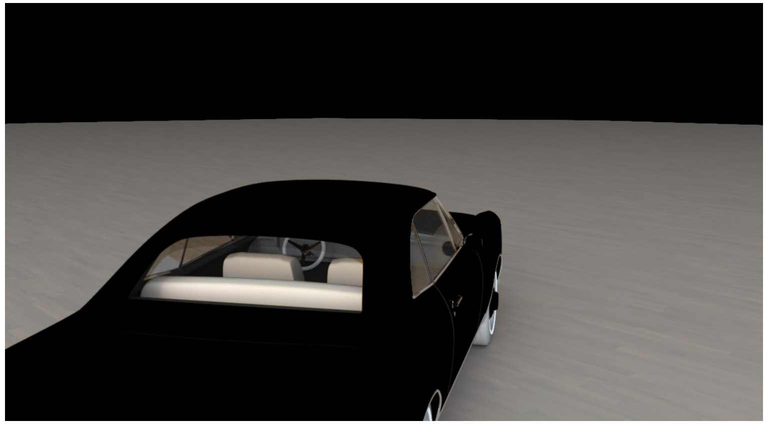
Ground Truth

Diffuse Network (d)

Prediction



Ground Truth



$$f = d * (a + \epsilon) + \exp(s)$$

Baseline Results

Prediction

Ground-
Truth

Experiments on Baseline

- L1-Loss on a 100-patch test-set
  - Layer increase: 11.40
  - Batch-normalization: 11.04
  - Layer increase + dropout: 9.170
  - Dropout: 8.137
  - Baseline: 7.916
  - Combined loss: 7.901
  - Learning rate reduction: 7.824

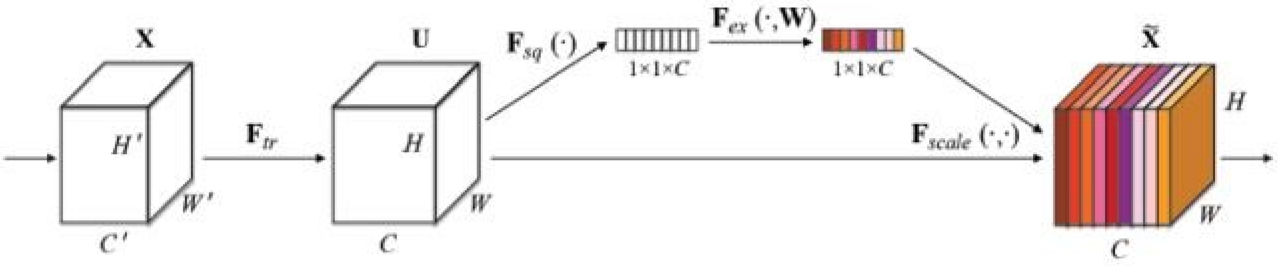
Comparison: Baseline vs. Combined Loss

Minor idea

Recent novel network architectures

- Channel attention

  This method can break the high correlation between channels and improve the performance of the model.



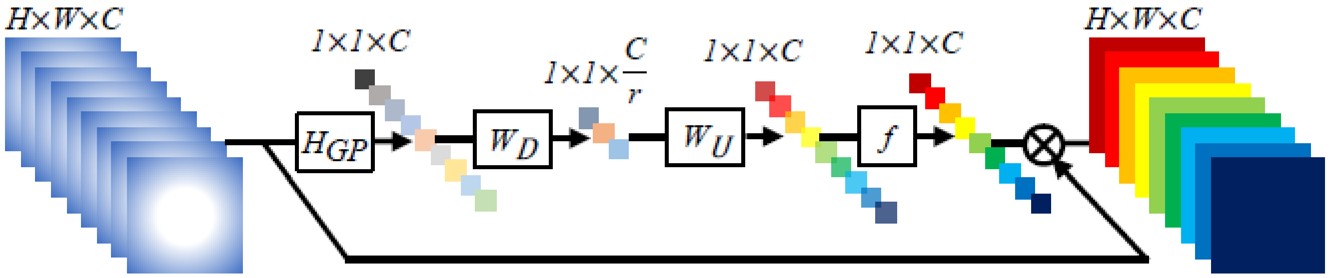


Fig. 3. Channel attention (CA). ⊗ denotes element-wise product

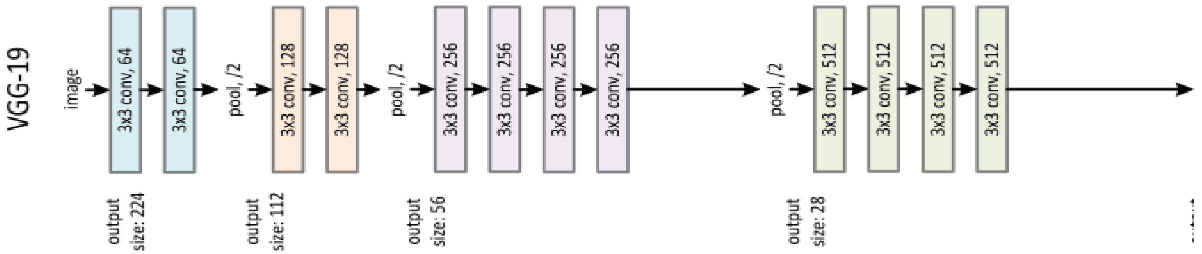
[Zhang et. al. 18] Image Super-Resolution Using Very Deep Residual Channel Attention Networks, ECCV2018

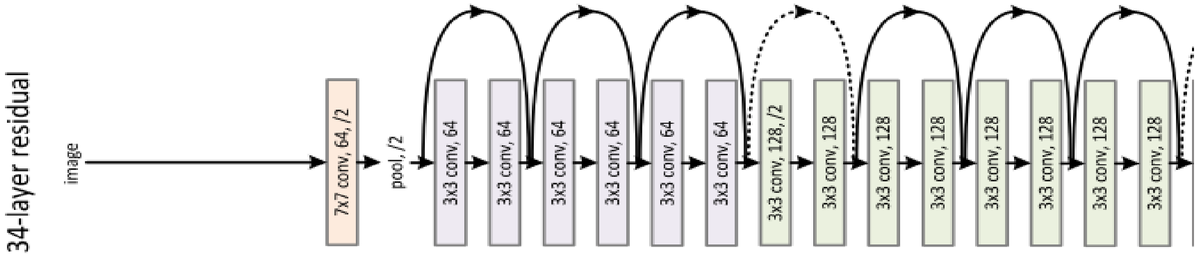
KPRCN

Residual learning

- KPCN follows the design of the VGG net (2014)
  - Small receptive field
  - Deep layers

- Therefore, we applied the residual learning (2016) technique to the KPCN.
  - No bottle neck, no batch normalization layer





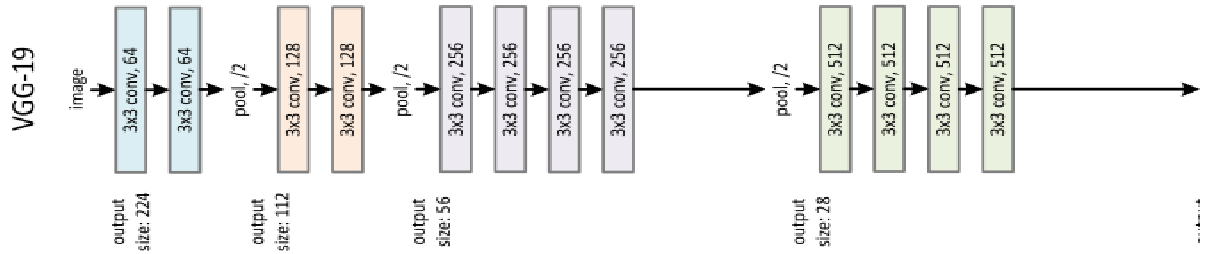
[He et. al. 16] Deep Residual Learning for Image Recognition, CVPR2016

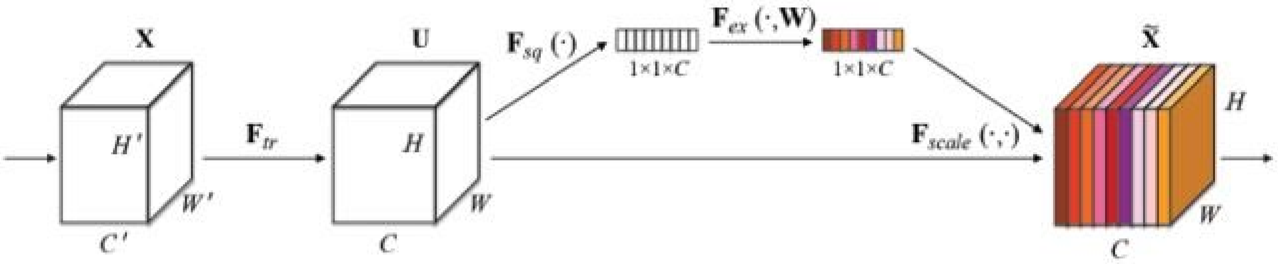
KPRCAN

Channel Attention

- Squeeze-Excitation network(2017) uses the channel attention over the convolutional layer.

- The KPRCAN uses the channel attention block.





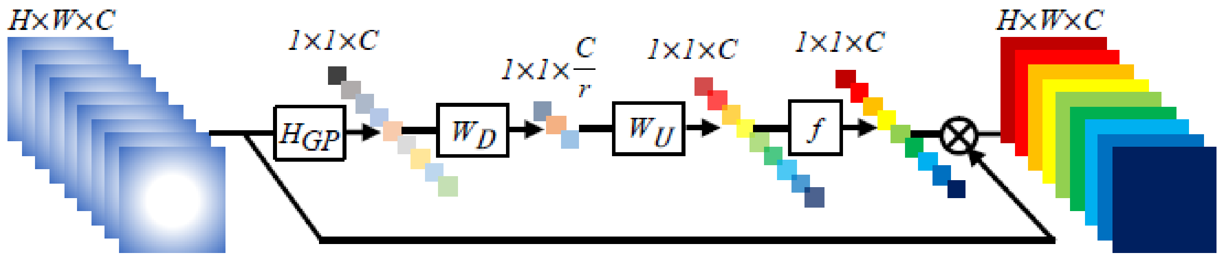


Fig. 3. Channel attention (CA). ⊗ denotes element-wise product

[Zhang et. al. 18] Image Super-Resolution Using Very Deep Residual Channel Attention Networks, ECCV2018

Direct Prediction model

Channel Attention

- For the new models, we also implemented the **direct prediction model** by employing the recent image restoration technique.
  - Recent denoising models does not use Kernel prediction method, but use Direct prediction.
  - Divide the model into head, body, tail block, and add the skip connection that cross the body block

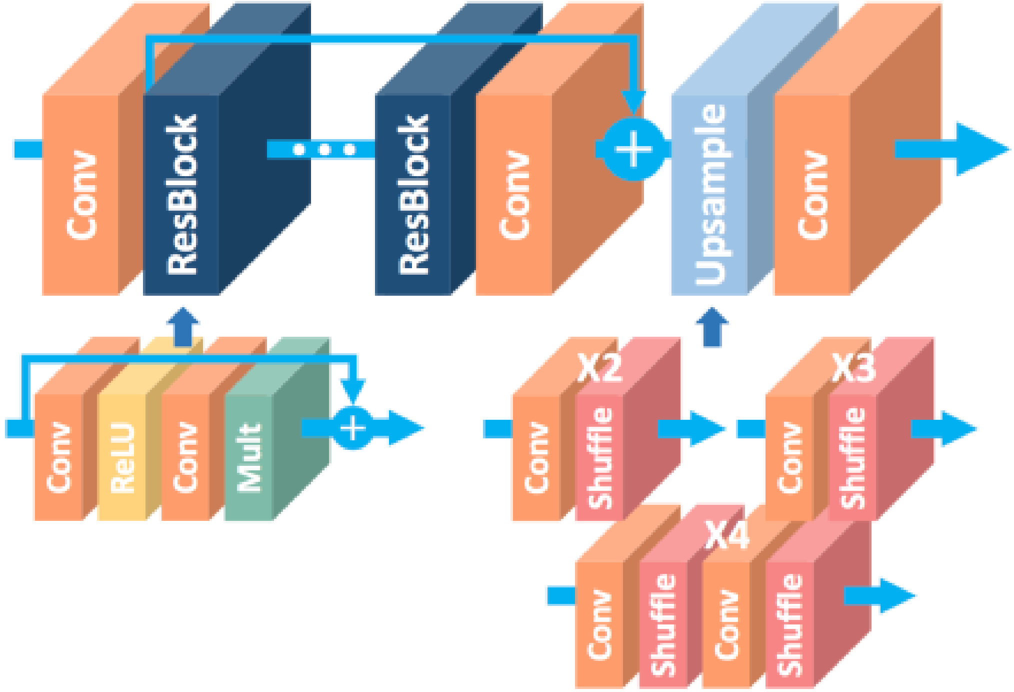


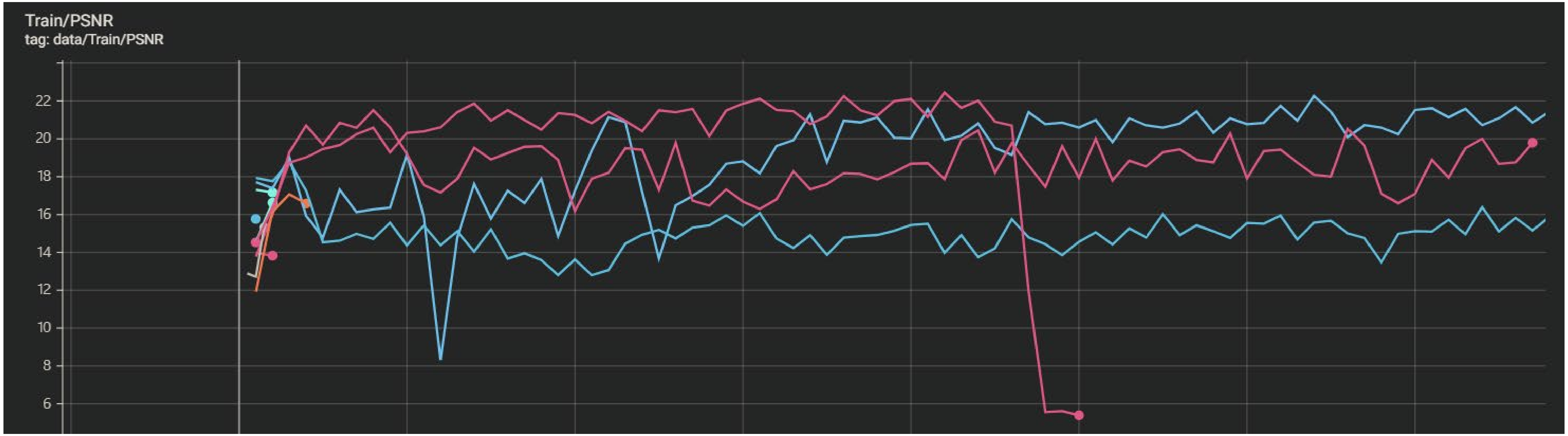
Figure 3: The architecture of the proposed single-scale SR network (EDSR).

[Zhang et. al. 18] Image Super-Resolution Using Very Deep Residual Channel Attention Networks, ECCV2018

Direct Prediction model

Direct prediction

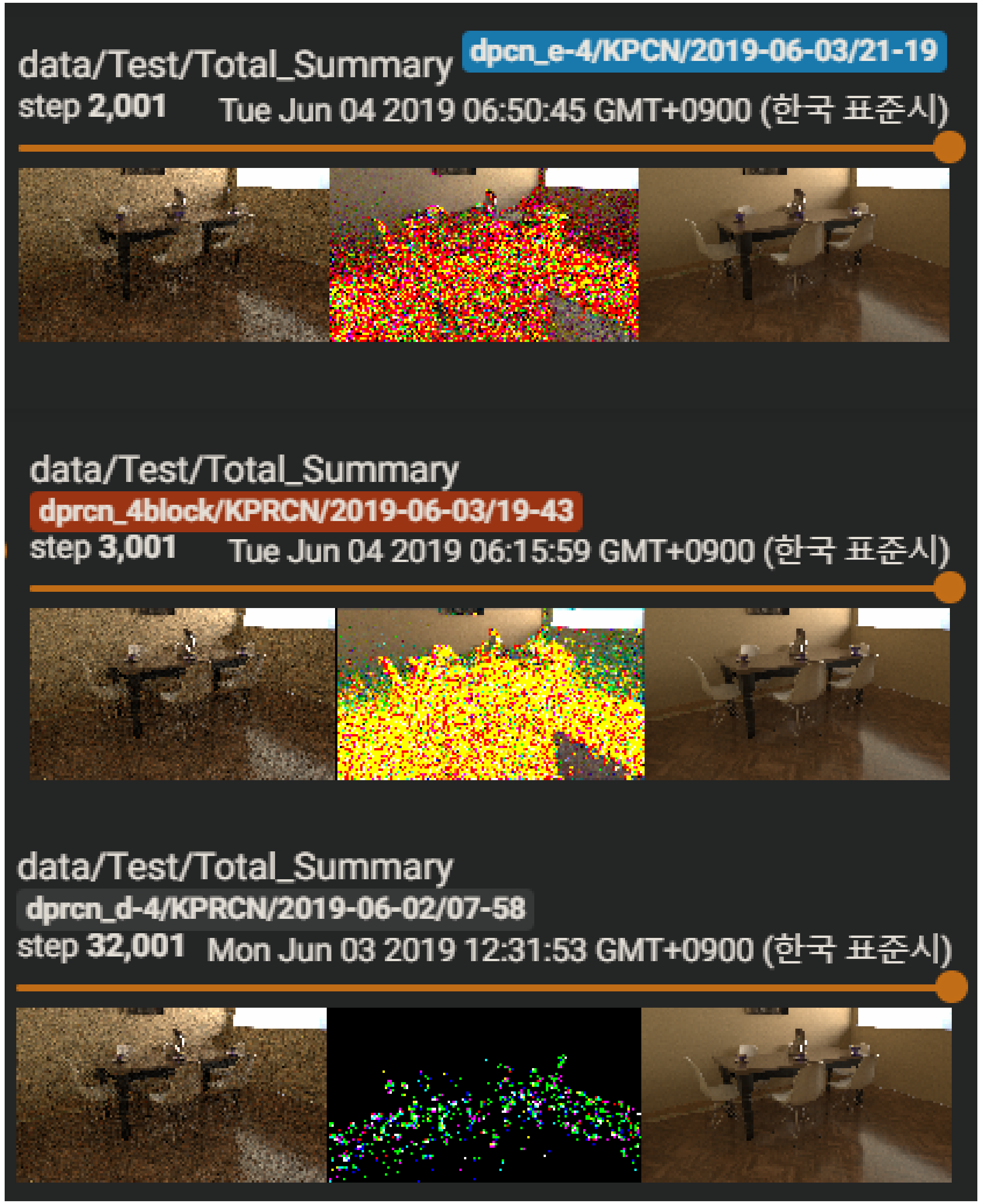
- All of the Direct prediction models does not trained well.



Direct Prediction model

Direct prediction

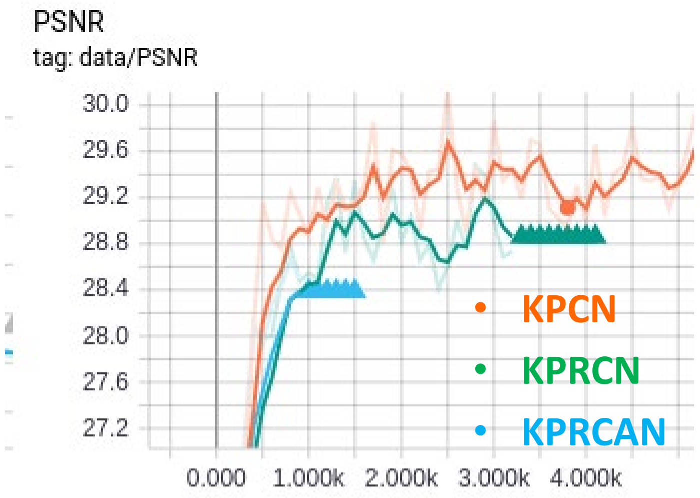
- All of the Direct prediction models does not trained well.
- The model easily exploded.



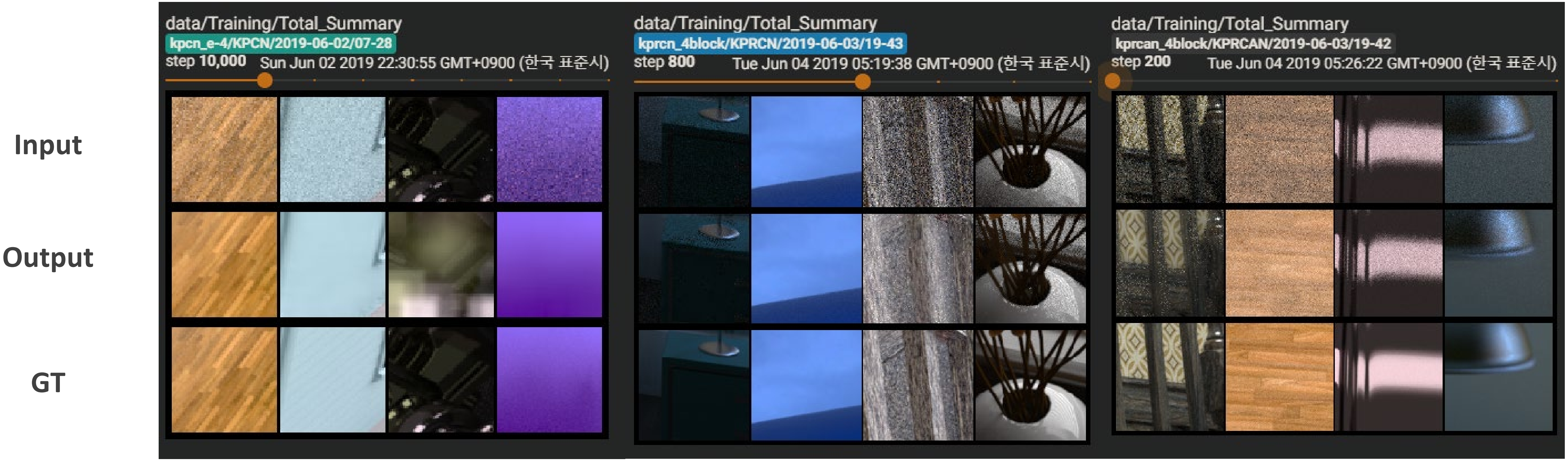
Kernel Prediction model

Kernel prediction

- For the patches, the performance of the KPCN is the best.



Test accuracy for patches

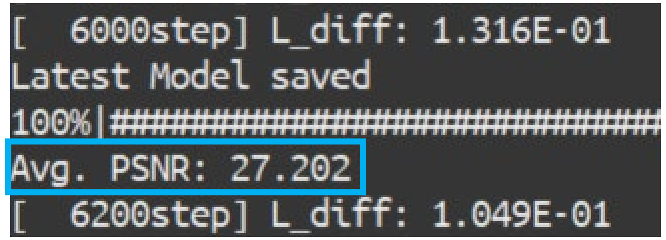


Kernel Prediction model

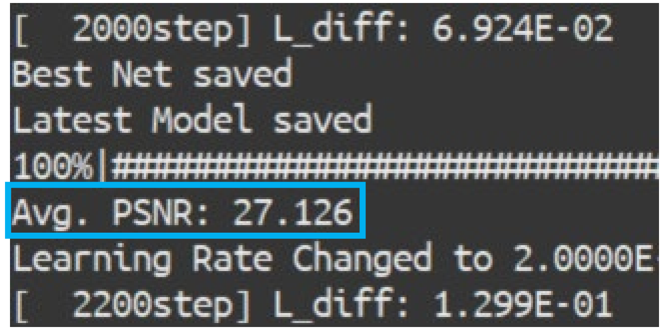
Kernel prediction

- For the test images, the performance of the KPRCAN is the best.

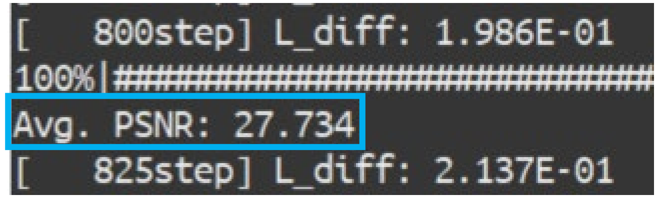
KPCN



KPRCN



KPRCAN



Kernel Prediction model

Kernel prediction

- For the **test images**, the performance of the KPRCAN is the best.
- But if we set the **same layer number** for the KPRCAN, it achieve the best performance.
- However more careful modification is required. (Explained later)

KPCN

```
[  6000step] L_diff: 1.316E-01
Latest Model saved
100%|##################################
Avg. PSNR: 27.202
[  6200step] L_diff: 1.049E-01
```

KPRCN

```
[  2000step] L_diff: 6.924E-02
Best Net saved
Latest Model saved
100%|##################################
Avg. PSNR: 27.126
Learning Rate Changed to 2.0000E
[  2200step] L_diff: 1.299E-01
```

KPRCAN

```
[   800step] L_diff: 1.986E-01
100%|##################################
Avg. PSNR: 27.734
[   825step] L_diff: 2.137E-01
```

KPRCAN 4 Block

```
Learning rate changed to 1.28E-0
100%|##################################
Avg. PSNR: 30.256
[   225step] L_diff: 1.592E-01
```

Final result - Input

Final result – Output(KPRCAN)

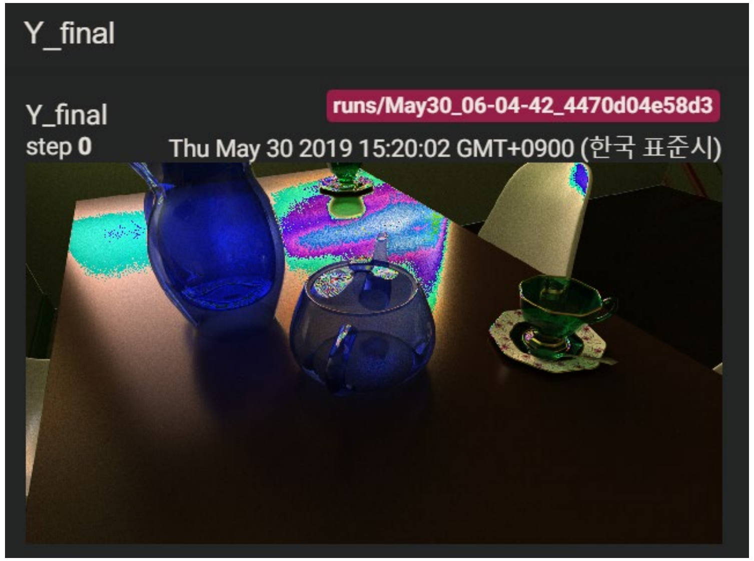
Final result - GT

Discussion

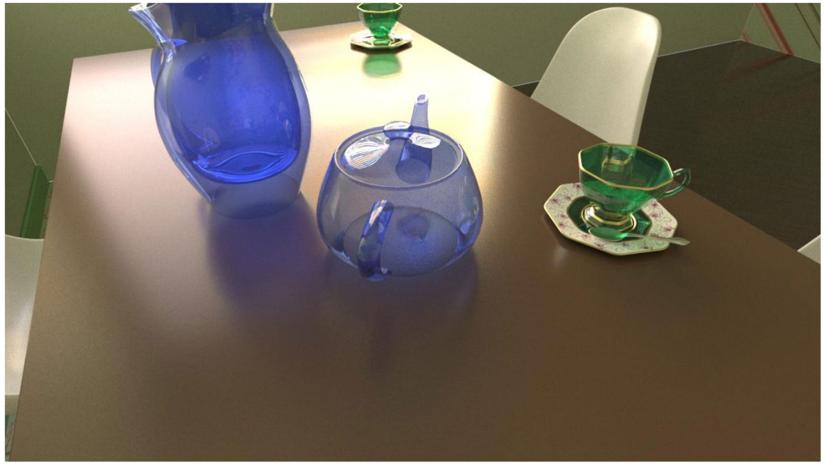
Discussion

Data format

- The original data is stored in HDR format. Which has th e range [0, inf]. Therefore, the inference range is also [0, inf] while most of the points have the value in [0,1].
- However when we evaluate the data, we convert the H DR data into RGB data by clamping the tensor value. Which means, we don’t need to exactly infer the value over 1.



Original GT data



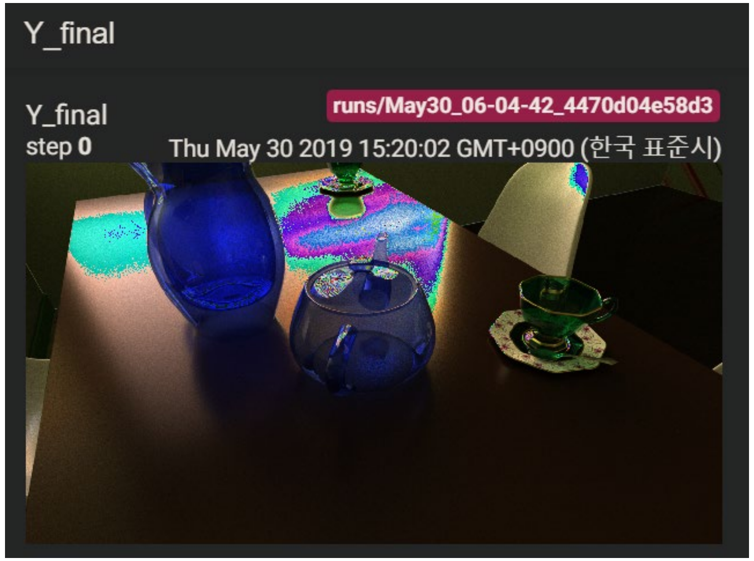
GT data in RGB

데이터 학습(최적화)는 HDR에서 하는데, 성능 평가는 RGB에서 한다.
서로 다른 특성으로 인해 성능이 나빠질 수 있다.

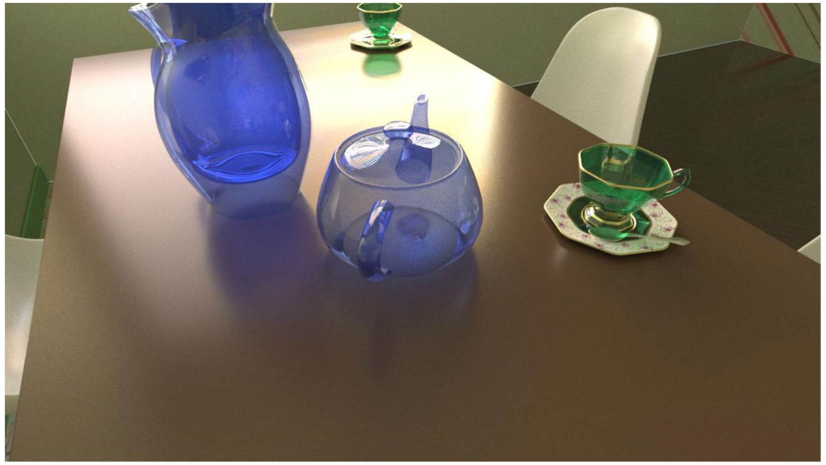
Discussion

Data format

- In short, we take loss from the HDR but evaluate in RGB where those format have different data range.
- This cause numerical extrapolation problem.
  (the value used for optimization and evaluation has different data range, different characteristic)



Original GT data



GT data in RGB

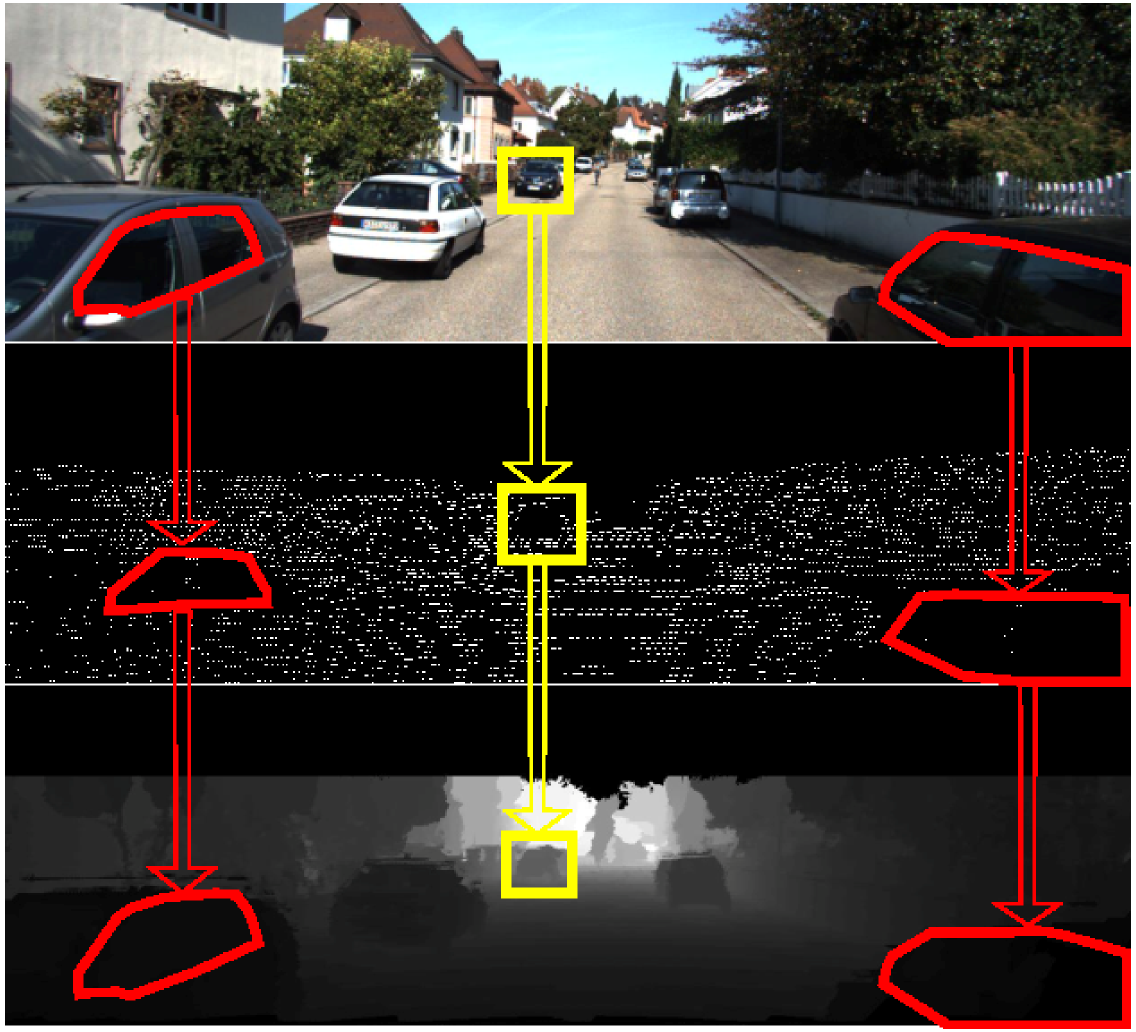
데이터 학습(최적화)는 HDR에서 하는데, 성능 평가는 RGB에서 한다.
서로 다른 특성으로 인해 성능이 나빠질 수 있다.

Discussion

Data format

- When we have a invalid value for the GT, we can **detach the loss from those invalid data points** in order to avoid irrelevant training.

- What if we applied this method? Or clamp the data at the training session.



Invalid point에서 로스를 받지 않도록 detach 할 수 있다. 이런 기법 혹은 데이터 전처리를 통해 성능을 개선할 수 있을지도 모른다.

Discussion

Kernel prediction

- For the **test images**, the performance of the KPRCAN is the best.
- But if we set the **same layer number** for the KPRCAN, it achieve the best performance.
- However more careful modification is required. (Explained later)

KPCN

```
[   6000step] L_diff: 1.316E-01
Latest Model saved
100%|##################################
Avg. PSNR: 27.202
[   6200step] L_diff: 1.049E-01
```

KPRCN

```
[   2000step] L_diff: 6.924E-02
Best Net saved
Latest Model saved
100%|##################################
Avg. PSNR: 27.126
Learning Rate Changed to 2.0000E
[   2200step] L_diff: 1.299E-01
```

KPRCAN

```
[    800step] L_diff: 1.986E-01
100%|##################################
Avg. PSNR: 27.734
[    825step] L_diff: 2.137E-01
```

KPRCAN
4 Block

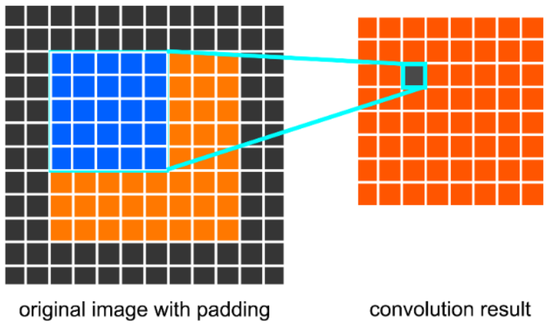
```
Learning rate changed to 1.28E-0
100%|##################################
Avg. PSNR: 30.256
[    225step] L_diff: 1.592E-01
```

Discussion

Network structure

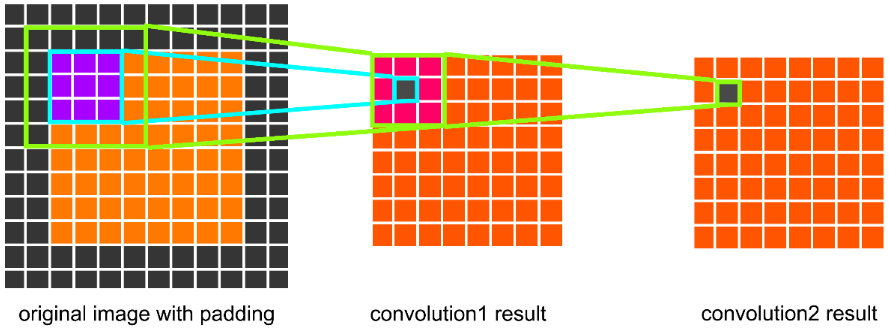
- In terms of the receptive field, stacking two 3x3 layer has same size of receptive field with one 5x5 layer.
- In addition, ResBlock require at least two block between the skip connection

5x5 conv



3x3 conv

3x3 conv



5x5 conv

5x5 conv

5x5 conv

3x3 conv

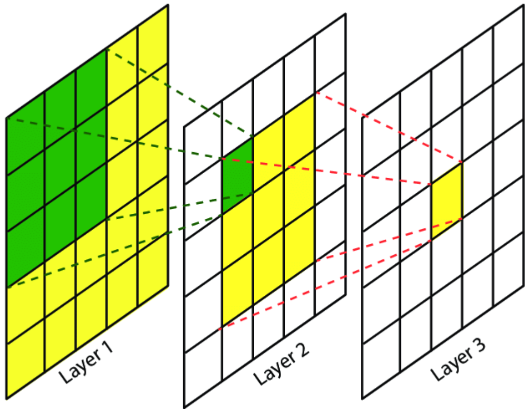
3x3 conv

3x3 conv

3x3 conv

3x3 conv

3x3 conv



Same receptive field, double number of layers

3x3 conv 2개와 5x5 conv 1개는 다른 레이어 수를 갖지만 같은 크기의 영역을 커버한다.

Kernel Prediction model

Network structure

- The KPRCAN 4 Block has same size of receptive field and number of layers with the KPCN.
- Since this network does not use the Batch normalization, it cannot use large number of layers.
- Furthermore, this model is extremely sensitive so we need to modify this network carefully.

8 of 5x5 conv layer body (Total 10 layer) — **KPCN**

```
[  6000step] L_diff: 1.316E-01
Latest Model saved
100%|##################################
Avg. PSNR: 27.202
[  6200step] L_diff: 1.049E-01
```

8 of 5x5 ResBlock body (Total 18 layer) — **KPRCN**

```
[  2000step] L_diff: 6.924E-02
Best Net saved
Latest Model saved
100%|##################################
Avg. PSNR: 27.126
Learning Rate Changed to 2.0000E
[  2200step] L_diff: 1.299E-01
```

8 of 5x5 SEBlock body (Total 18 layer) — **KPRCAN**

```
[   800step] L_diff: 1.986E-01
100%|##################################
Avg. PSNR: 27.734
[   825step] L_diff: 2.137E-01
```

4 of 3x3 SEBlock body (Total 10 layer) — **KPRCAN 4 Block**

```
Learning rate changed to 1.28E-0
100%|##################################
Avg. PSNR: 30.256
[   225step] L_diff: 1.592E-01
```

앞의 슬라이드 처럼 KPCN과 KPRCAN이 같은 영역을 커버하도록 층수와 커널 수를 조절하니 확연히 다른 결과를 얻었다.

BN등을 사용할 수 없어 깊은 층 수를 갖게되면 학습이 되지 않고 모델 자체가 매우 민감한 특성 등으로 인해 세심한 조정이 필요하다.

Team Contribution

- Cheolmin: Baseline Code
- Nick: Experiments on Baseline
- Minki: Extended Models

Thank You!

Reference

[Liu et. al. 17] Learning Efficient Convolutional Networks through Network Slimming, ICCV2017

[Zhang et. al. 18] Image Super-Resolution Using Very Deep Residual Channel Attention Networks, ECCV2018

[Lehtinen et. al. 18] Noise2Noise: Learning Image Restoration without Clean Data, ICML2018

[Bako Et al. 17] “Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings.” ACM Transactions on Graphics 36, no. 4 (July 20, 2017)

Back up

Table 2: Quantitative results about color image denoising. Best results are **highlighted**.

Method	Kodak24				BSD68				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70
CBM3D	36.57	30.89	28.63	27.27	35.91	29.73	27.38	26.00	36.00	30.36	27.94	26.31
TNRD	34.33	28.83	27.17	24.94	33.36	27.64	25.96	23.83	33.60	27.40	25.52	22.63
RED	34.91	29.71	27.62	26.36	33.89	28.46	26.35	25.09	34.59	29.02	26.40	24.74
DnCNN	36.98	31.39	29.16	27.64	36.31	30.40	28.01	26.56	36.21	30.28	28.16	26.17
MemNet	N/A	29.67	27.65	26.40	N/A	28.39	26.33	25.08	N/A	28.93	26.53	24.93
IRCNN	36.70	31.24	28.93	N/A	36.06	30.22	27.86	N/A	35.81	30.28	27.69	N/A
FFDNet	36.81	31.39	29.10	27.68	36.14	30.31	27.96	26.53	35.77	30.53	28.05	26.39
RNAN (ours)	**37.24**	**31.86**	**29.58**	**28.16**	**36.43**	**30.63**	**28.27**	**26.83**	**36.59**	**31.50**	**29.08**	**27.45**

Table 3: Quantitative results about **gray-scale** image denoising. Best results are **highlighted**.

Method	Kodak24				BSD68				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70
BM3D	34.39	29.13	26.99	25.73	33.31	27.76	25.62	24.44	34.47	28.75	25.94	24.27
TNRD	34.41	28.87	27.20	24.95	33.41	27.66	25.97	23.83	33.78	27.49	25.59	22.67
RED	35.02	29.77	27.66	26.39	33.99	28.50	26.37	25.10	34.91	29.18	26.51	24.82
DnCNN	34.90	29.62	27.51	26.08	33.88	28.36	26.23	24.90	34.73	28.88	26.28	24.36
MemNet	N/A	29.72	27.68	26.42	N/A	28.43	26.35	25.09	N/A	29.10	26.65	25.01
IRCNN	34.76	29.53	27.45	N/A	33.74	28.26	26.15	N/A	34.60	28.85	26.24	N/A
FFDNet	34.81	29.70	27.63	26.34	33.76	28.39	26.29	25.04	34.45	29.03	26.52	24.86
RNAN (ours)	**35.20**	**30.04**	**27.93**	**26.60**	**34.04**	**28.61**	**26.48**	**25.18**	**35.52**	**30.20**	**27.65**	**25.89**