GradNet Image Denoising

by Yang Liu, Saeed Anwar, Liang Zheng and Qi Tian,
CVPR 2020

A demonstrative presentation by Bretan Cezar-Alexandru

April 4th, 2025

Overview

- Introduction – Image Denoising
- Proposed Method – GradNet
- Data
- Tested Alternatives
- Experiments
- Results
- Conclusions

Introduction – Image Denoising

- Image Denoising = removal of unwanted signals from data captured by any imaging device that manifest as random pixel-level variations in color or intensity information
- Usages:
  - Modern consumer-grade Digital Still Cameras;
  - A step in restoration of existing data captured with other imaging devices (old cameras)

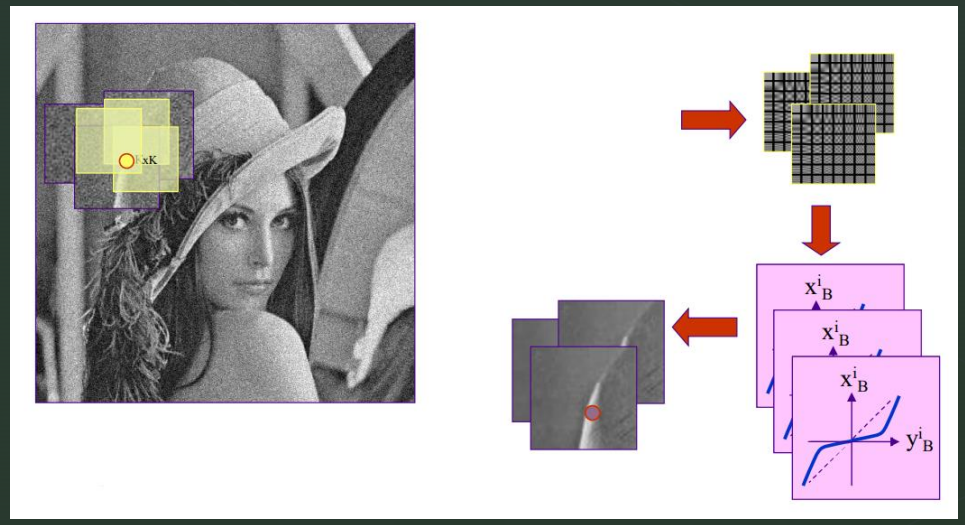


Excerpts from SIDD dataset, taken with Motorola Nexus 6:
1) ISO 100 2) ISO 1600 3) ISO 3200

Evolution of Image Denoising

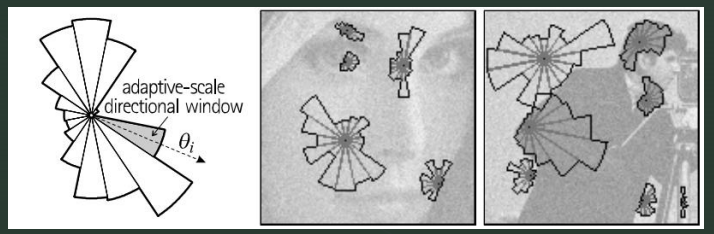
- Ubiquitous filters: Contraharmonic, Median, Adaptive Median…
- Transform-based: Wavelet Shrinkage (1995) [1]

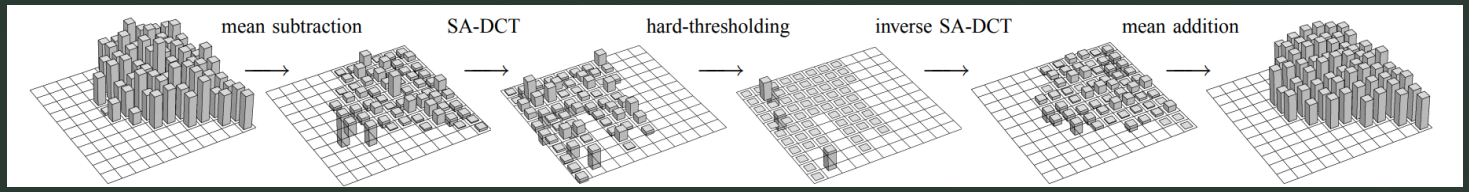
- Transform + Window-based: Sliding-window DCT (1999, TUT) [2]



Evolution of Image Denoising

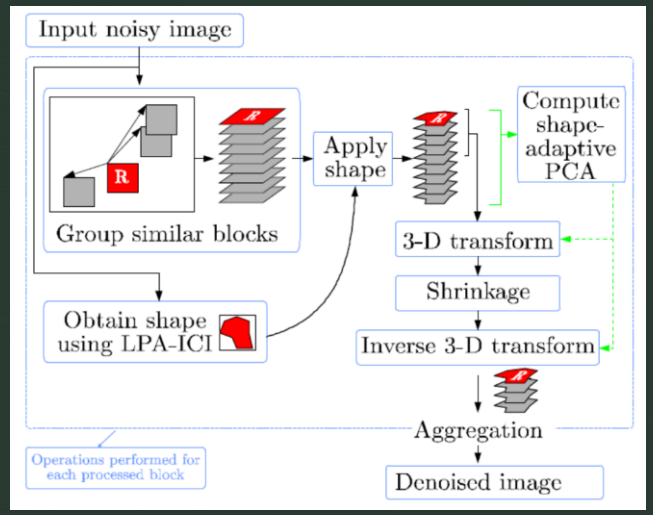
- Transform + Asymmetric Window-based: Pointwise SA-DCT (2005, TUT) [3]





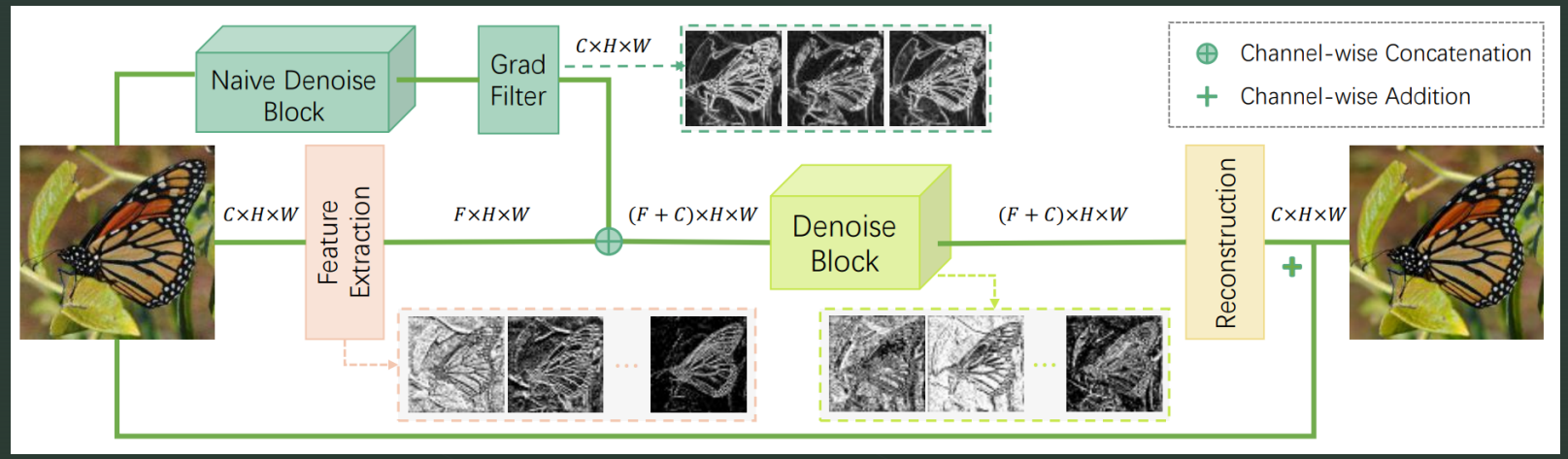
Evolution of Image Denoising

- Pre-Deep Learning state-of-the-art: BM3D-SAPCA (2009, TUT) [4], [5]



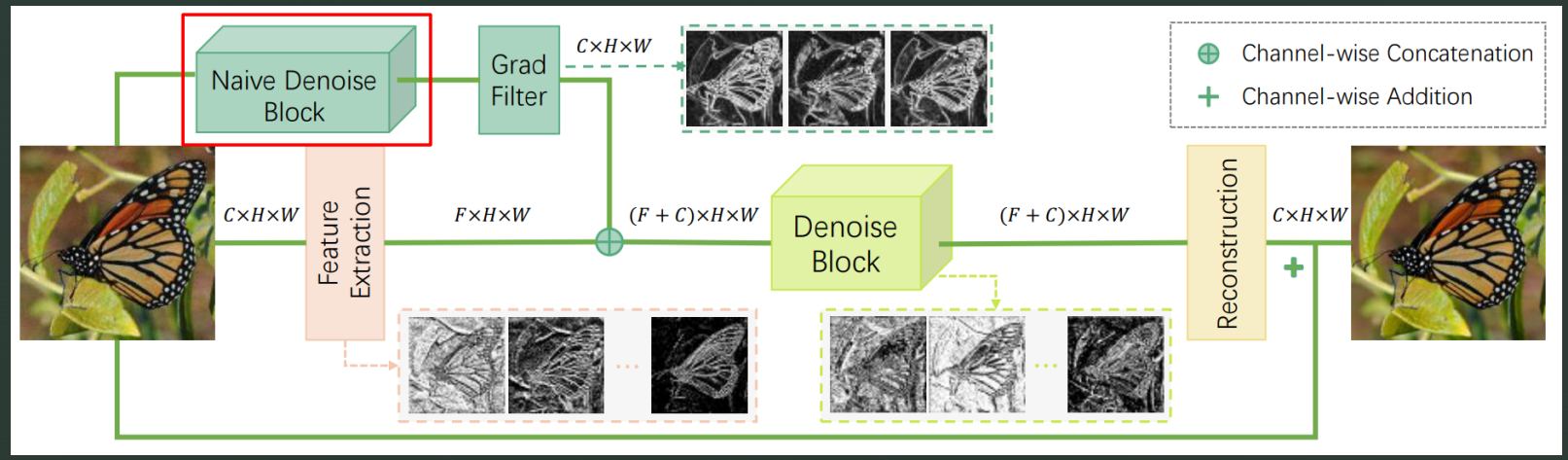
- 2010s - Advent of Deep Learning methods: great run-time performance, on par with some methods based on transforms & pre-defined filters and relatively easy to develop using already existing solutions.

GradNet Overview



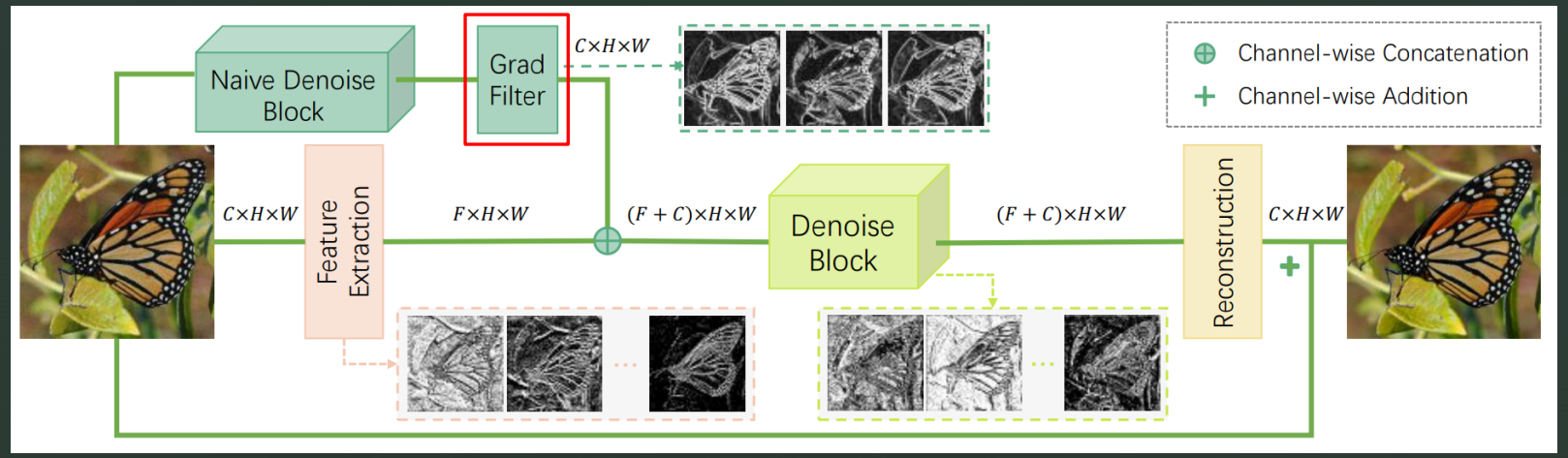
- Novel contributions of the paper:
  - MSKResNet architecture;
  - Clean image gradient fusion into latent features;
  - Gradient consistency loss;
  - End-to-End DL-based denoising pipeline that utilizes a pre-existing denoising algorithm.

GradNet – Pipeline Components



- Naive Denoise Block: Existing denoising solution that receives the input image.
- Examples given in paper: BM3D [4], **DnCNN** [6]

GradNet – Pipeline Components



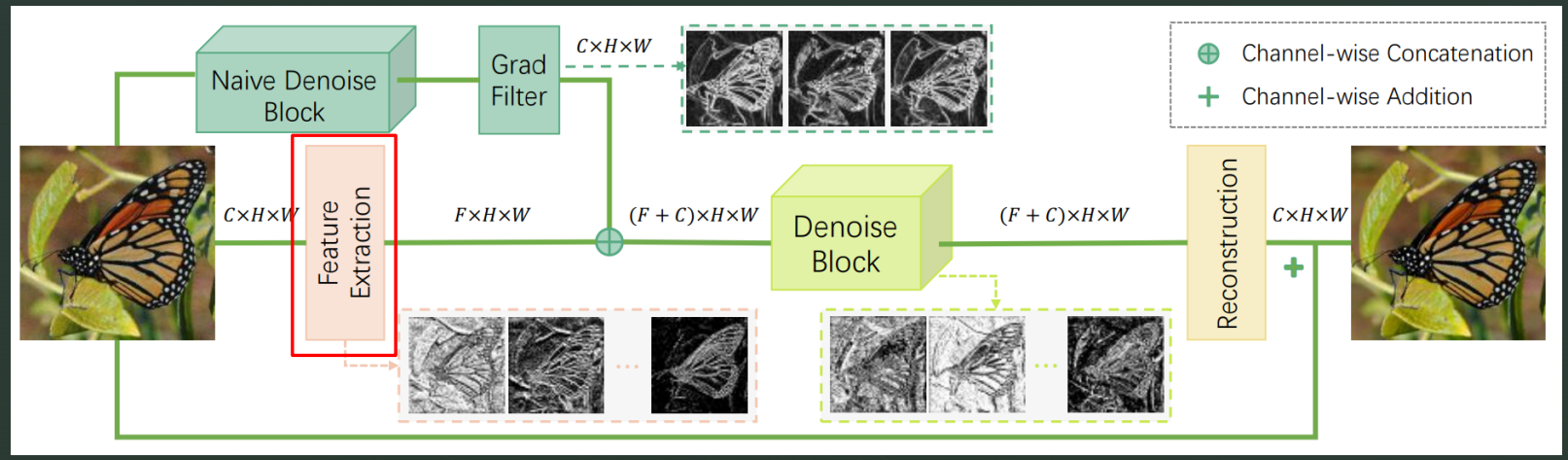
- Grad Filter: Extraction of per-channel Sobel gradient magnitudes from the intermediary denoised image.

$$S_H = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, S_V = \begin{bmatrix} -1 & -2 & +1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

$$\boldsymbol{G_H(x) = S_H * x, G_V(x) = S_V * x}$$

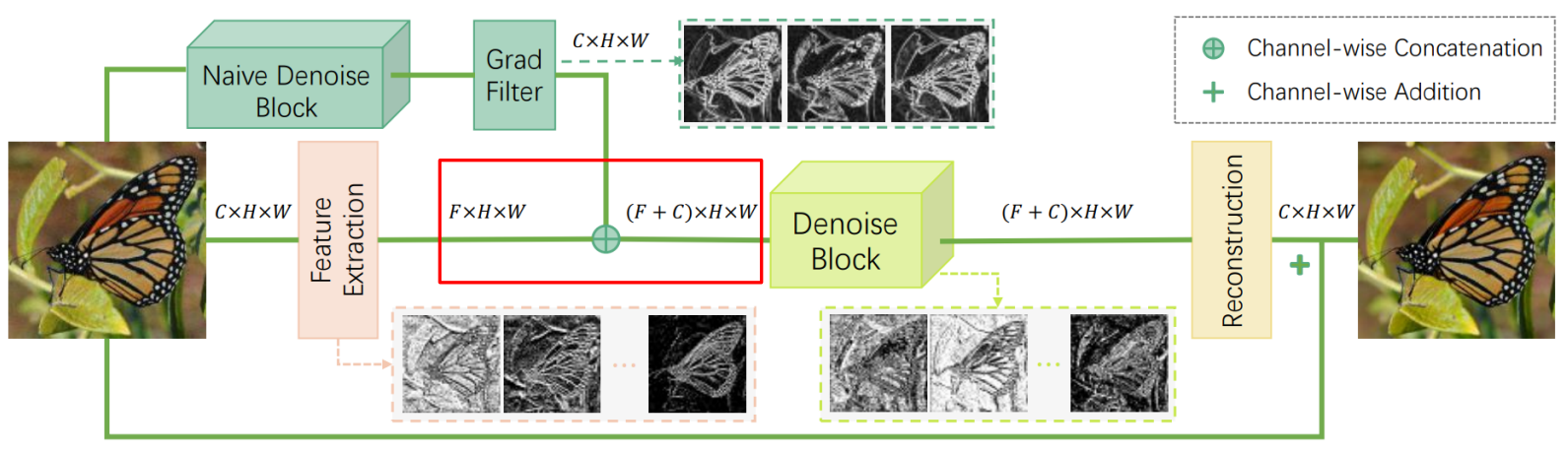
$$\boldsymbol{G(x) = \sqrt{G_H(x)^2 + G_V(x)^2}}$$

GradNet – Pipeline Components

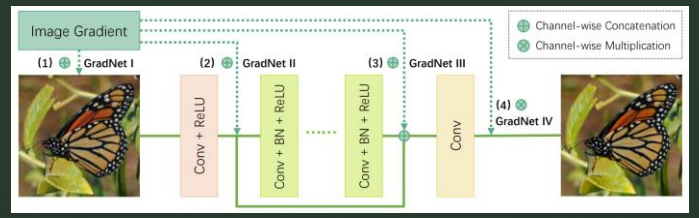


- Feature Extraction: Basic 2D Convolutional Block followed by ReLU activation. Extracts multi-channel features from the input image.

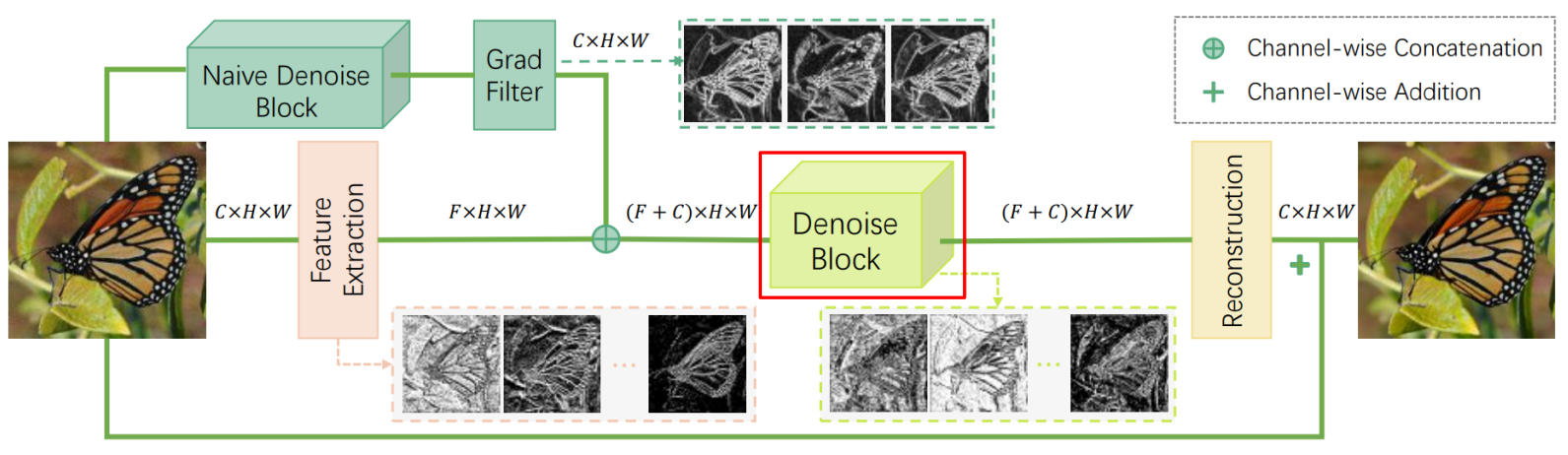
GradNet – Pipeline Components



- **GRADIENT FUSION**: Channel-wise concatenation of the intermediary denoised image gradients and extracted features from the noisy image.
- Motivation: Image gradients are similar to the outputs of the early CNN layers, whose kernels are responsive mainly to edges, yielding outputs that are, in a sense, homogenous to the gradients' meaning.
- Proof-of-Concept study:

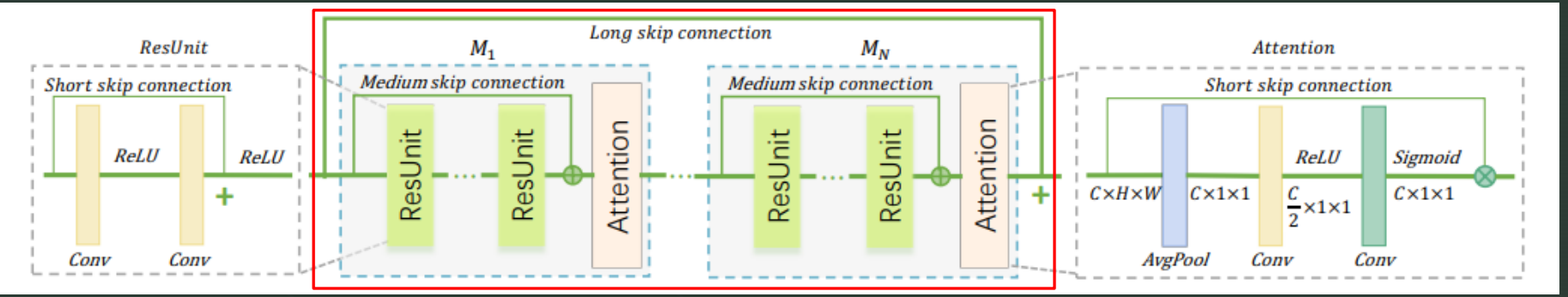


GradNet – Pipeline Components



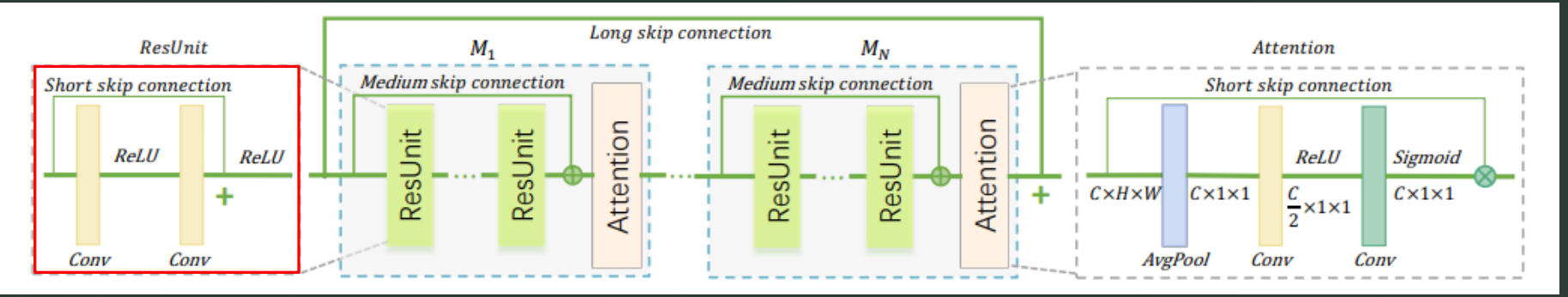
- Denoise Block: Multiple Skip-Connection Residual Network (MSKResNet) for the actual denoising of the feature maps.

GradNet – MSKResNet architecture



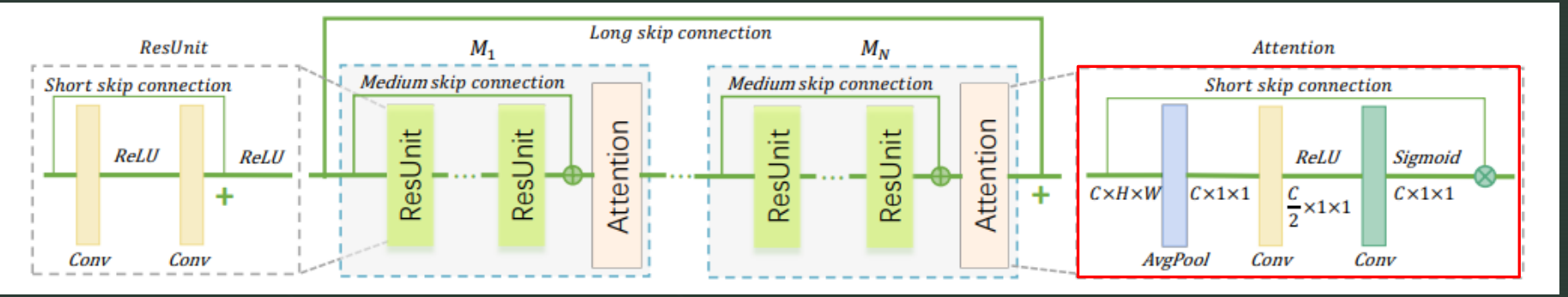
- Main components:
  - Residual Modules
    - Residual Units
    - Attention Units
  - Skip connection from the denoise block input to the final output that performs an addition operation.

GradNet – MSKResNet architecture



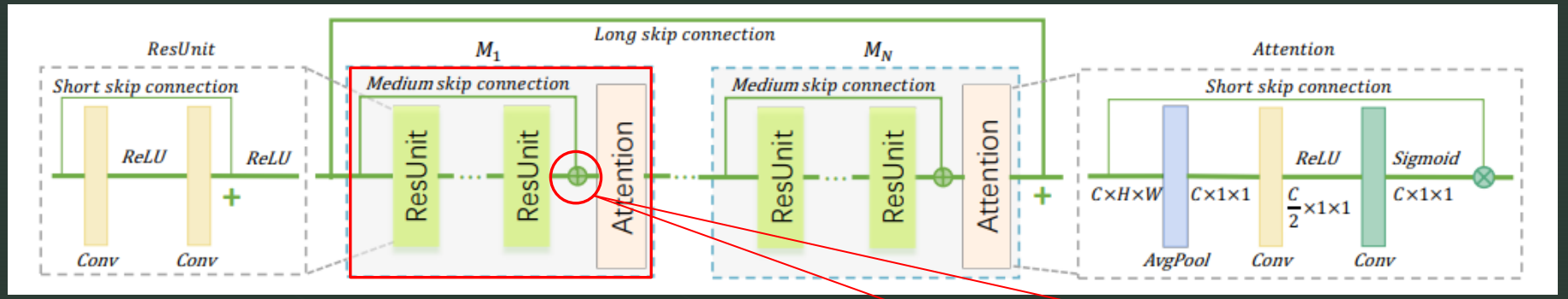
- Residual Unit: 2 Convolutional layers with ReLU activation, with a skip connection from the input that performs addition.

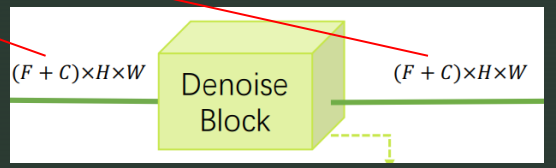
GradNet – MSKResNet architecture



- Attention Unit – Squeeze-and-Excitation block [7]:
  - 1 Average Pooling layer for extracting global information for each channel of the feature map received as input;
  - 1 Convolution layer with ReLU activation that halves the number of features to learn the dependency between channels;
  - 1 Convolution layer with sigmoid activation that restores the number of channels;
  - Skip connection from the input to the output that performs element-wise multiplication

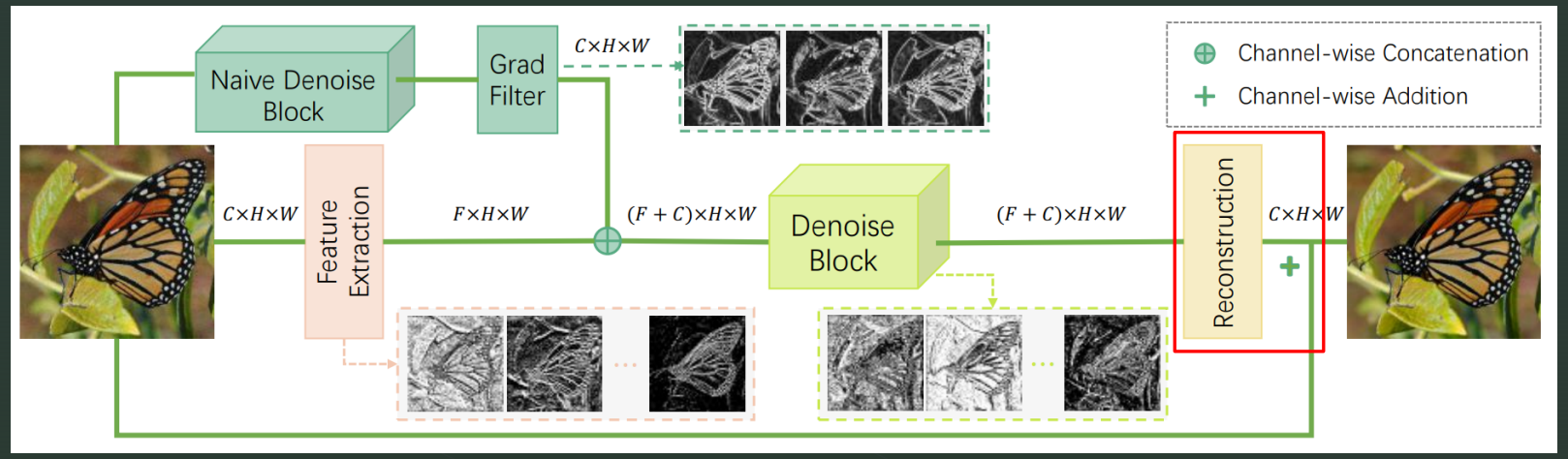
GradNet – MSKResNet architecture





- Residual Modules – paper suggests 4 of them in series:
  - Residual Units – paper suggests 4 of them in series;
  - Attention Unit;
  - Skip connection from the input to right before the Attention Unit. The figure from paper states a channel-wise concatenation being performed, which results in an increase in the number of channels within the feature map as it passes through each residual module. However, this contradicts with the paper's statement that the entire denoise block should output the same number of channels as the input. Therefore, in my implementation, an addition operation is performed instead.

GradNet – Pipeline Components



- Reconstruction: Basic 2D Convolutional layer that changes the multi-channel features back to 3 channels and a global skip connection that performs a sum with the original image. The final output is the denoised RGB image.

GradNet – Gradient Consistency Loss

- Another contribution the paper brings is the employment of the L1 error between the gradients of the ground truth and the predicted denoised image as a regularization loss.

$$L_p = \frac{1}{N}\sum_{i=1}^{N} ||\text{GradNet}(\boldsymbol{y_i}, G(\hat{\boldsymbol{x}}_{\boldsymbol{i}})) - \boldsymbol{y_i}||_1$$

$$L_g(\widetilde{\boldsymbol{x}}_{\boldsymbol{i}}) = ||G_H(\widetilde{\boldsymbol{x}}_{\boldsymbol{i}}) - G_H(\boldsymbol{x_i})||_1 + ||G_V(\widetilde{\boldsymbol{x}}_{\boldsymbol{i}}) - G_V(\boldsymbol{x_i})||_1.$$

$$L = L_p + \lambda * L_g(\widetilde{\boldsymbol{x}}_{\boldsymbol{i}})$$

Data Setup

- The paper makes a very good point that noise (as illuded to earlier, can caused by factors like high ISO sensitivity used) in capturing real images is actually different from Additive White Gaussian Noise:
  - As seen in the example below, AWGN actually reduces contrast and makes the image seem overall brighter.
  - AWGN is spatially invariant, while real noise shows a degree of spatial covariance;
  - Real noise differs (in aggressiveness and character) from sensor to sensor;
- It is unclear whether the authors mixed up AWGN-corrupted samples with the real noise-corrupted ones for a single training run, or they did separate runs for the two types of noise.

‘ground-truth’

ISO 3200

AWGN $\sigma = 50$

Data Setup

- Real noise datasets: Smartphone Image Denoising Dataset (SIDD), Darmstadt Noise Dataset (DnD), RENOIR, PolyU. $\sigma \in [8,64]$.
- Regular image datasets, corrupted by AWGN noise at training and validation time: DIV2K, BSD, Kodak24, CBSD68, McMaster.
- Augmentations at training time: Flipping (3/4 chance to flip in one of 3 possible directions) and Rotating (3/4 chance to rotate in one of 3 possible angles).
- 80x80 random cropping, 4 matching pairs of patches for real noise samples due to relatively reduced amount of them.
- Total training samples per epoch: 1385 with AWGN + 1600 with real noise.
- Total validation samples per epoch: 126 with AWGN + 320 with real noise.

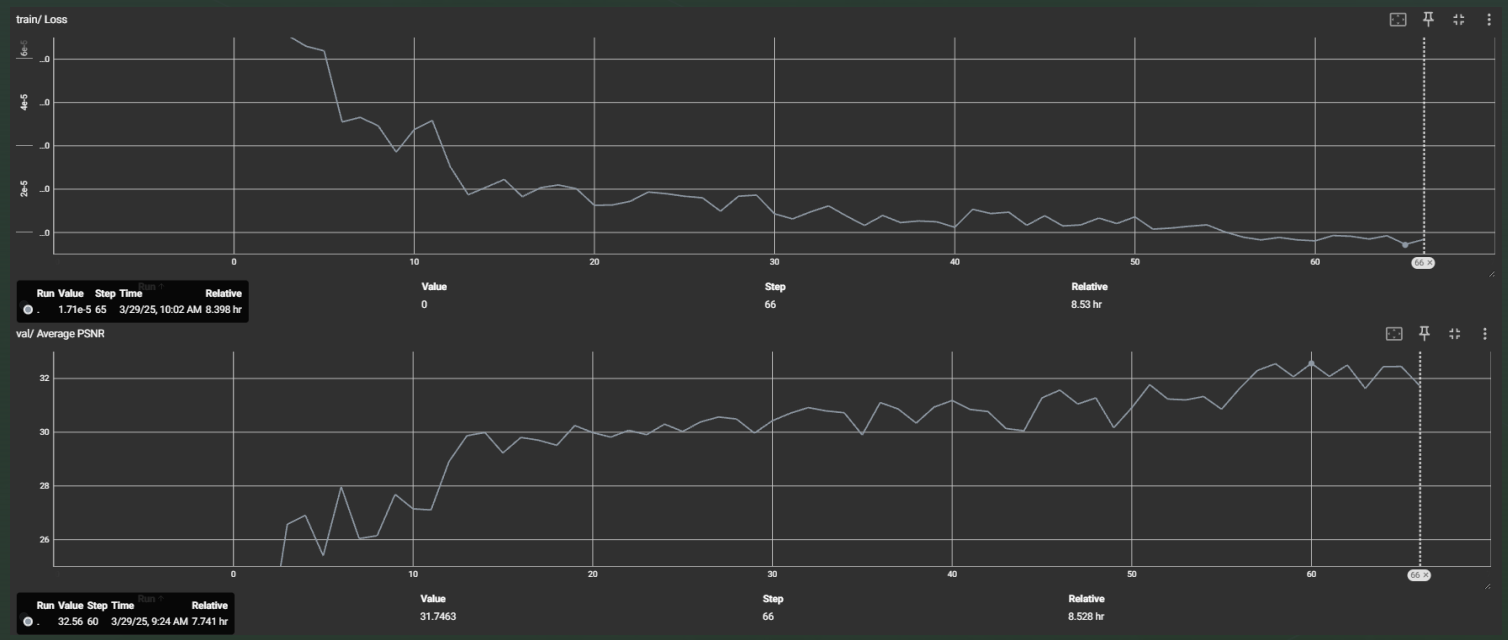
```python
class NoiseType(StrEnum):
    AWGN = "awgn"
    REAL = "real"


@dataclass
class ImageData:
    original_image: Path
    noise_type: NoiseType
    sigma_min: Union[float | None]
    sigma_max: Union[float | None]
    ref_image: Union[Path | None]
    flip: bool
    rotate: bool
```

Tested Alternatives

- BM3D – Python package: `pip install bm3d` courtesy of TUNI
  - BM3D requires the noise variance as a known variable and it is not known in the case of real noise samples
  - The `skimage.restoration.estimate_sigma()` noise variance estimator looked like a solution, though it normally undershoots the variance; after some observations, decided it was better to scale the values given by the variance by some value (5.0) than assume an arbitrary value for all real noise samples
- DnCNN – Own model trained, using the same data setup as GradNet;
- Used as stand-alone comparison points and ‘naive’ denoisers in GradNet pipeline.

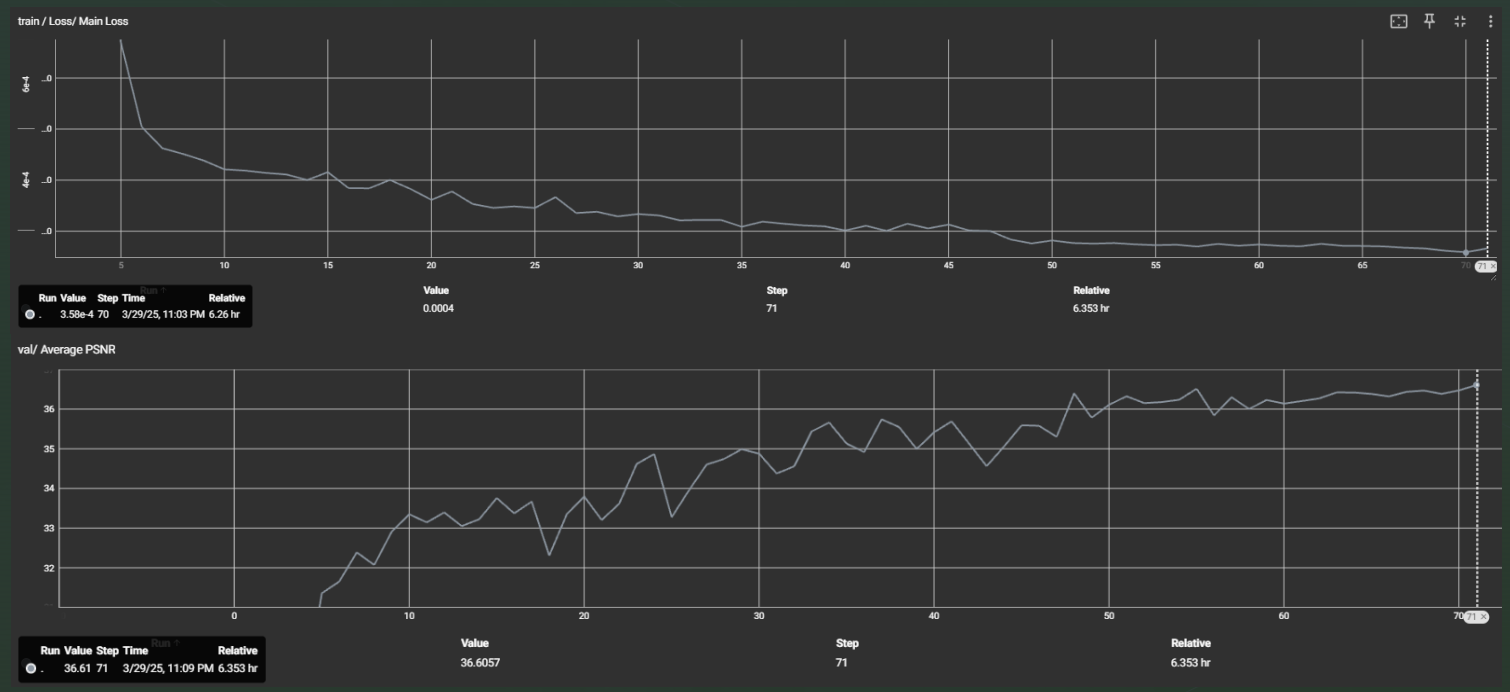
Experiments

- 1. Training DnCNN-S, the 16-layer variant
  - Managed to train using the ground truth image as the target instead of the residual noise as the target, as described in the original paper



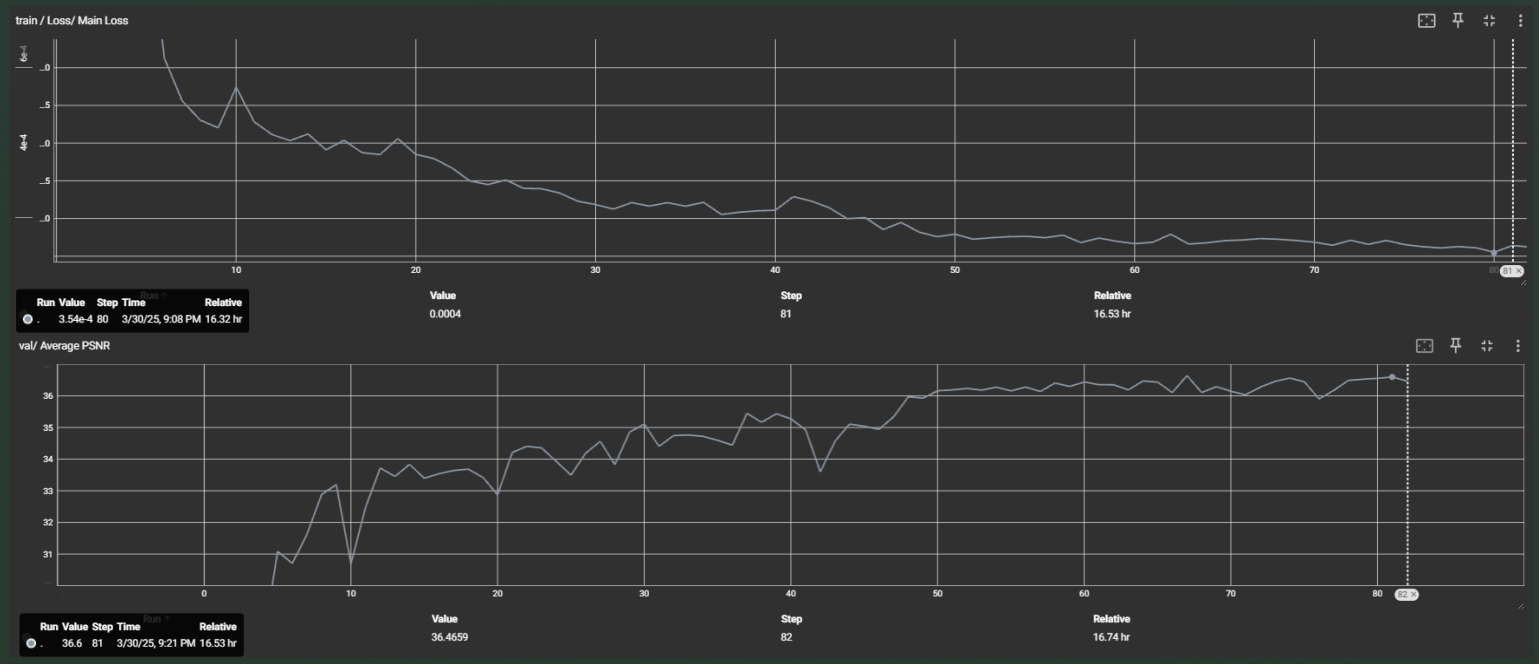
Experiments

- 2. Training GradNet with DnCNN-S as the naive denoiser:
  - 2.1. Initial feature size = 64, concatenated with 3 gradient channels, 4 Residual Modules with 4 Residual Units each.



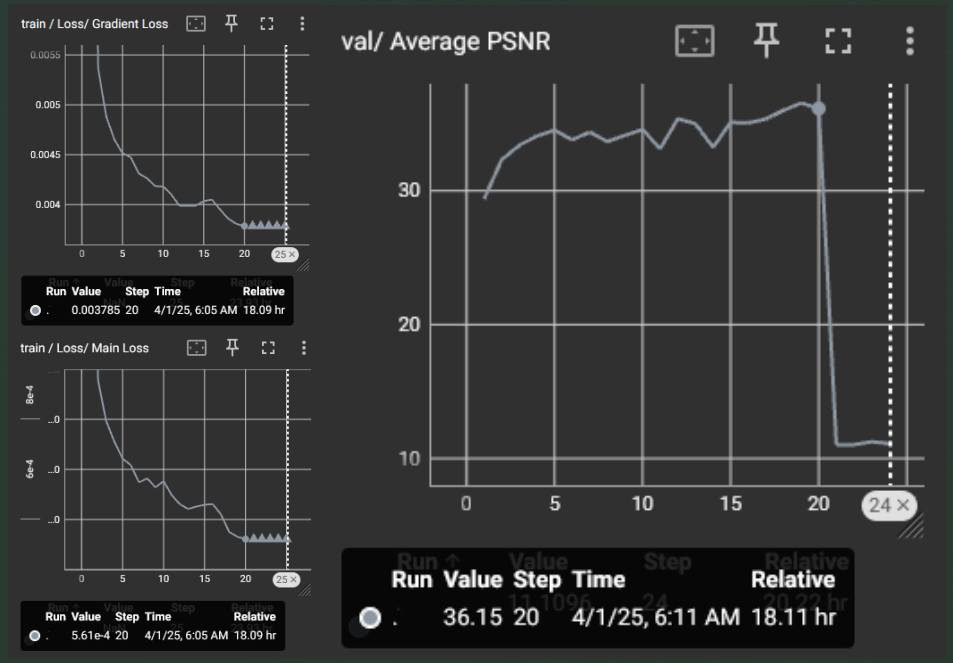
Experiments

- 2. Training GradNet with DnCNN-S as the naive denoiser:
  - 2.2. Initial feature size = 64, the 3 gradient channels replicated 12 times in an interlacing manner along the feature map channel dimension in an attempt to boost the weight of the approximated gradients (referring to it as ‘gradient mixup’), same MSKResNet structure as the previous experiment.



Experiments

- 3. Training GradNet with BM3D as the naive denoiser
  - Extremely long, BM3D runs on CPU
  - Weirdly enough, the only experiment where the gradient loss wasn't 0.0 for most of the training
  - A lot of image patches break BM3D during a normalization step when the implementation performs YCbCr / opponent color transform.




```python
if self.__naive_dn == bm3d_rgb:

    # Guard against color patches that break bm3d
    while True:
        crop_coords = (
            int(uniform(0, im_full.shape[0]-self.__crop_size[0])),
            int(uniform(0, im_full.shape[1]-self.__crop_size[1]))
        )
        sample = im_full[
            crop_coords[0]:crop_coords[0]+self.__crop_size[0],
            crop_coords[1]:crop_coords[1]+self.__crop_size[1],
            :
        ]

        o = array(sample).reshape([sample.shape[0] * sample.shape[1], 3]) @ self.__opp.T
        o_rng = o.max(axis=0) - o.min(axis=0)

        if all(o_rng > self.__eps):
            break

        del o
        del o_rng

        # print(f"Suspicious image detected: {str(data.original_image)}")

    del im_full
```

Experiments - Reflection

- Would’ve liked more, but hardware and time constraints happened along with several dataset implementation errors leading to more wasted time…
- Google Colab wasn’t an option, in my experience, it tends to disconnect at random for inactivity and wipe the workspace, unfeasible due to training runs being very long;
- Lab computers were unable to train using BM3D and PyTorch data loading optimization leveraging multiprocessing, my own computer could, weirdly enough;
- Wasted a lot of time thinking real noise datasets don’t normally have ground truth images, as the versions I initially found didn’t have the other half, leading me to think the PSNR is measured between the noisy image and the predicted denoised version;
- In the end, all experiments were run with:
  - nVIDIA GeForce RTX 2060 Max-Q 6GB
  - AMD Ryzen 7 4800HS
  - 16 GB RAM

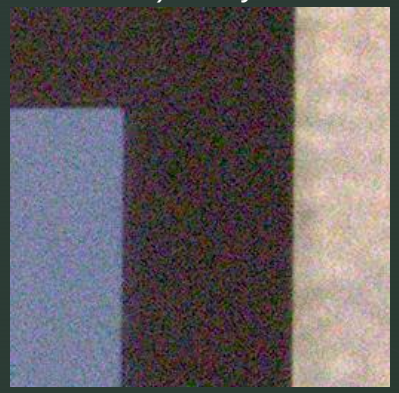
Results

256x256 image patches, PSNR averages for entire datasets	DnCNN	BM3D	GradNet + DnCNN	GradNet + DnCNN + gradient mixup	GradNet + DnCNN, with BM3D gradients fed on inference	GradNet + BM3D
Kodak24 + AWGN σ = 25	27.482	31.866	30.435 (paper: ~32)	30.205	30.435	29.642
Kodak24 + AWGN σ = 50	23.765	28.435	25.355 (paper: ~29)	25.236	25.355	25.763
SIDD	35.230 (paper: ~23)	37.139 (paper: ~25)	39.319 (paper: ~38)	39.152	39.319	39.112

- Claimed performance of GradNet with DnCNN gradients is higher for AWGN, possibly due to the auhors having 5000+ samples of this type in their dataset, but it is unclear whether they also had real noise samples mixed up in the same training suite;
- Stand-alone DnCNN results on SIDD are significantly higher, possibly due to the authors training it exclusively on AWGN-corrupted samples, as the original paper did;
- My BM3D results on SIDD are significantly higher than the ones on the paper, possibly due to the authors picking an arbitrary value for the noise variance;
- Results in the 3rd column are identical to the ones in the 5th, suggesting that a slightly more accurate approximation of the gradients at inference time doesn't help improve the results.

Results

1) Noisy

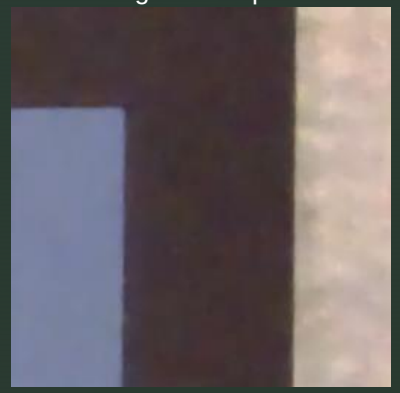


2) BM3D w/ est. σ

3) DnCNN

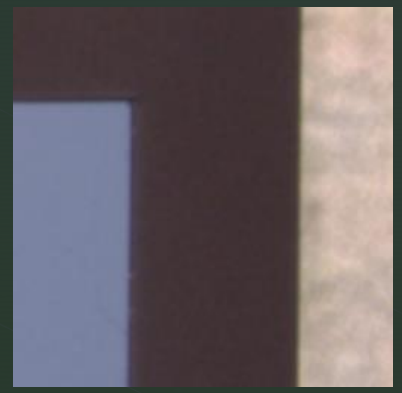
4) GradNet + DnCNN

5) GradNet + DnCNN w/ grad mixup



6) GradNet + BM3D

7) Ground Truth



PSNRs:

2) 33.317

3) 33.461

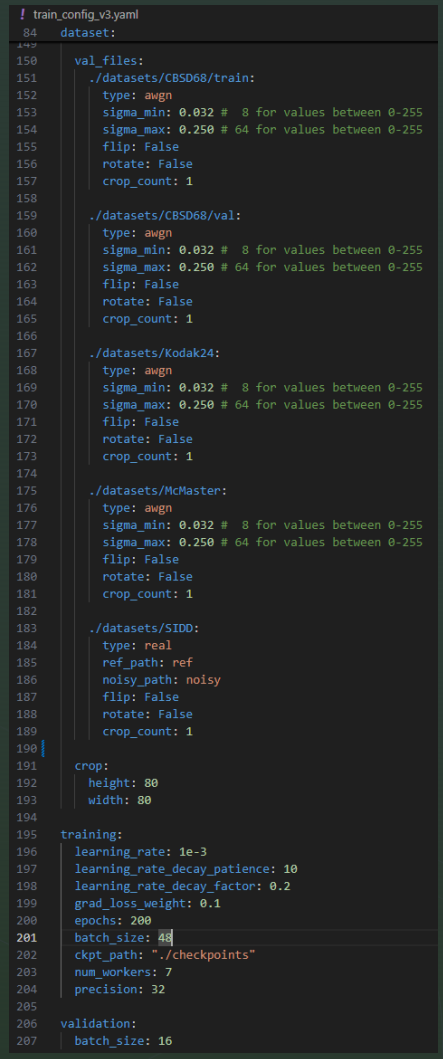
4) 37.713

5) 37.610

6) 37.282

Implementation info

- Full implementation can be found here: [Bretan-Cezar/gradnet-impl: Unofficial implementation of the "GradNet Image Denoising" paper]
- Dataset paths, augmentations, splits, noise ranges, network parameters, training hyperparameters and many more completely configurable via YAML



References

- [1] Donoho, David L., et al. “Wavelet Shrinkage: Asymptopia?” Journal of the Royal Statistical Society. Series B (Methodological), vol. 57, no. 2, 1995, pp. 301–69. JSTOR, http://www.jstor.org/stable/2345967
- [2] Egiazarian, K., Astola, J., Helsingius, M. & Kuosmanen, P. 1999. Adaptive denoising and lossy compression of images in transform domain. Journal of Electronic Imaging 8, 3, pp. 233-245
- [3] Alessandro Foi, Kostadin Dabov, Vladimir Katkovnik, and Karen Egiazarian "Shape-adaptive DCT for denoising and image reconstruction“
- [4] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian "COLOR IMAGE DENOISING VIA SPARSE 3D COLLABORATIVE FILTERING WITH GROUPING CONSTRAINT IN LUMINANCE-CHROMINANCE SPACE"
- [5] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian "BM3D Image Denoising with Shape-Adaptive Principal Component Analysis"
- [6] K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," in *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142-3155, July 2017, doi: 10.1109/TIP.2017.2662206
- [7] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2017

Q&A

Thank you for your attention!