Image Deblurring Using Nonlinear Activation Free Network (NAFNet)

By Sai, Kassam, Sam, Alex

Why Image Deblurring Matters

- Used in medical imaging, astronomy, and biometrics
  - Clean visuals obtained for accurate diagnoses, quality of telescope imagery is enhanced, and face/fingerprint recognition accuracy increases
- Surveillance systems, traffic cameras, and everyday smartphone photography benefit from crystal clear imagery
- Blur from motion, camera shake & low light leads to loss of critical detail
- Real-world need: fast, efficient, and accurate restoration methods

Motion blur encountered in medical imagery

Example of low light blur

What is NAFNet?

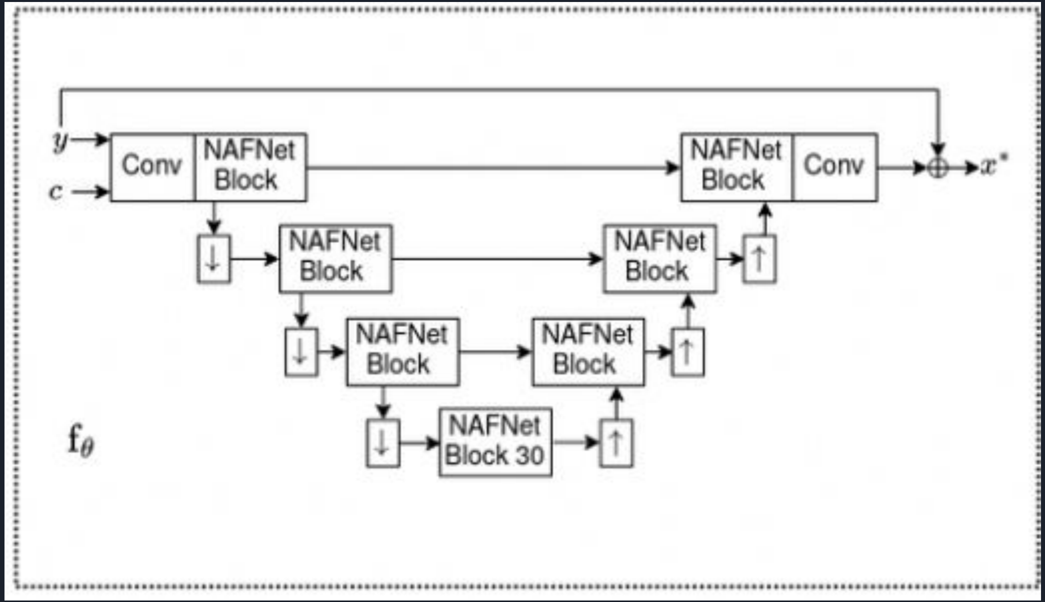
- Lightweight U-Net–style model with nonlinear activations
- Prioritizes speed and low resource usage
- Uses SimpleGate activation and residual learning
- Suitable for real-time, low-power applications

What We Changed:

- Used GELU activation
- Allowed variable numbers of encoder/middle/decoder blocks

Diagram of a Typical NAFNet Model

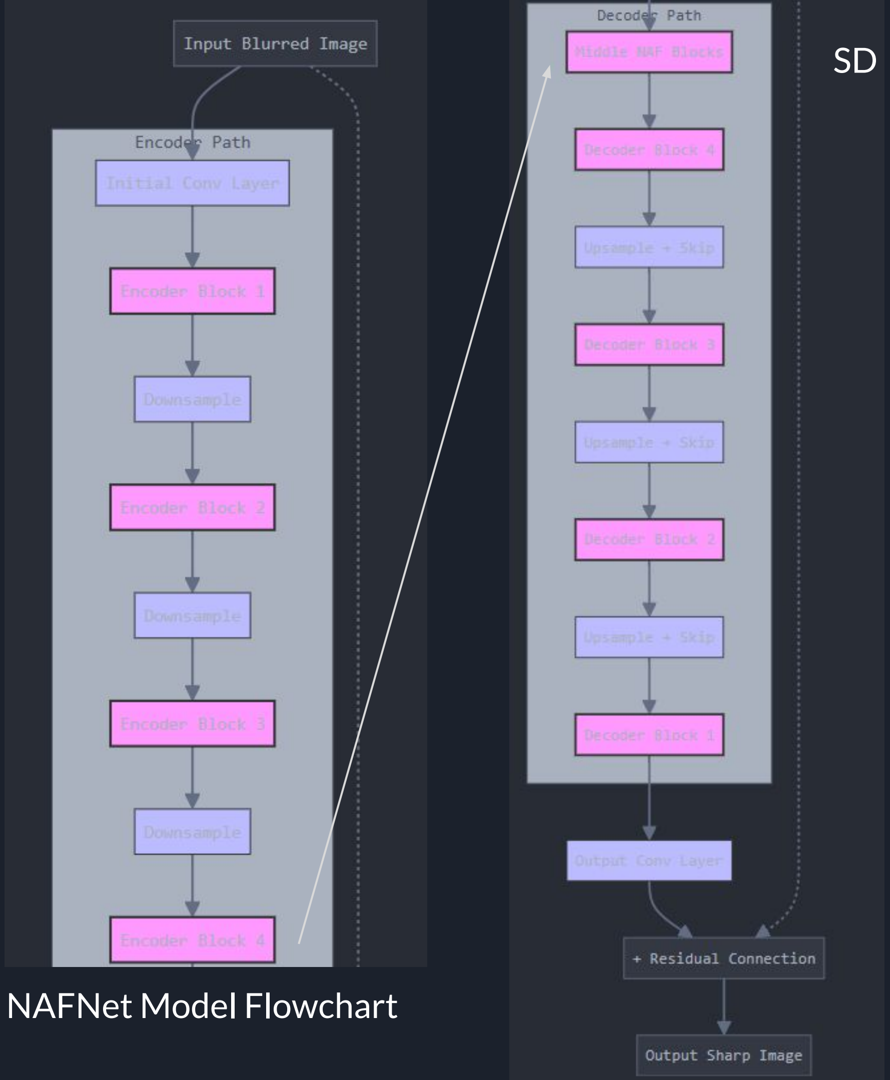
- Model utilizes three main phases: downsample and extract, deblurring procedure, upsampling and reconstruct
- This is where it gets its “U-Net” name
- Some features are allowed to “skip” blocks to preserve lower level features
- Implemented in our design as well



SD

Our Contribution

- Built and tested a lightweight deblurring model using a modified version of NAFNet
- Explored impact of model size, loss functions, and learning rates
- Modified NAFNet with GELU activations and flexible block configurations



Our NAFNet Model Flowchart

Dataset Used

GoPro Image Deblurring Dataset

- 3,214 image pairs (blurred + sharp), 1280×720 resolution
- 2,103 used for training, 1,111 for testing
- Captures real-world blur: motion, camera shake, etc.
- Commonly used in deblurring research

Example Pair of Blurred and Sharp image

SC

Methodology

- Input passed through a modified encoder → bottleneck → decoder structure
- Last convolution layer reshapes to 3-channel RGB
- Final output = Network Output - Original Input (residual learning)

Loss Function Used:

Mix of:

- L1 Loss: pixel accuracy
- Charbonnier Loss: smoother, handles noise
- SSIM Loss: structure preservation
- Edge Loss: sharper boundaries

$$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{Charbonnier}}(x, y) + \beta \cdot \mathcal{L}_{\text{SSIM}}(x, y) + \gamma \cdot \mathcal{L}_{\text{Edge}}(x, y) + \delta \cdot \mathcal{L}_{\text{L1}}(x, y)$$

Loss Function Used

AW

Results: Loss Function Comparison

- Best results with all 4 losses (PSNR 24.34 dB, runtime 18:14)
- Removing losses like SSIM or L1 slightly reduced performance
- There is a little tradeoff between image quality and runtime

Test Number	Losses Used	PSNR (dB)	Runtime (min:sec)
1	Charbonnier, L1, Edge, SSIM	24.34	18:14
2	Charbonnier, L1, Edge	24.05	18:09
3	Charbonnier, L1, SSIM	23.99	17:55
4	Charbonnier, SSIM, Edge,	24.06	17:39

Results- Tuning Loss Weights

- Tuning **beta (SSIM), gamma (Edge), and L1 weight** improved results
- Best config: β = 0.2, γ = 0.01, L1 weight = 0.05
- Reached PSNR 24.39 dB with runtime under 18 minutes

Test	Beta	Gamma	L1	PSNR (dB)	Runtime
5	0.05	0.01	0.01	24.38	17:55
6	0.05	0.01	0.05	24.37	17:57
7	0.05	0.05	0.01	24.36	17:57
8	0.05	0.05	0.05	24.34	17:54
9	0.10	0.01	0.01	24.37	17:29
10	0.10	0.01	0.05	24.34	18:14
11	0.10	0.05	0.01	24.37	17:54
12	0.10	0.05	0.05	24.32	17:56
13	0.20	0.01	0.01	24.32	17:56
14	0.20	0.01	0.05	24.39	17:56
15	0.20	0.05	0.01	24.35	17:54
16	0.20	0.05	0.05	24.37	18:01

Results- Architecture & Learning Rate Effects

- Increased width and blocks led to **no major PSNR gain**
- Best model: Width = 32, Encoder = [1,1,2,6]
- Cosine annealing learning rate helped early convergence
- Constant LR (1e-3, 5e-4) performed worse

Test	Width	Encoder Blocks	PSNR (dB)	Runtime
17	32	[1,1,2,6]	24.39	17:56
18	32	[2, 2, 2, 6]	24.36	19:05
19	48	[1, 1, 2, 6]	24.33	19:05
20	48	[2, 2, 2, 6]	24.35	19:07

Test	Learning Rate	PSNR (dB)	Runtime
21	Cos. Annealing	24.39	17:56
22	5×10^{-4}	24.06	18:34
23	1×10^{-3}	24.27	18:36

SC

Results: NAFNet Output Images

Conclusion

- Lightweight modified NAFNet achieves **24.4 dB PSNR** in ~18 minutes
- Architecture tuning and loss weighting matter more than size
- Cosine annealing + all 4 losses = best combo
- Model is robust and trainable even with limited GPU

Future work:

- More training images and epochs
- Use of better hardware/GPU
- Explore perceptual loss and real-time applications

Thank You!