Modified NAFNet for Image Deblurring through Weighted Loss Functions

Sainandan Dore
School of Engineering
University of Dayton
Dayton, USA
dores1@udayton.edu

Samuel Conley
School of Engineering
University of Dayton
Dayton, USA
conleys8@udayton.edu

Kassam Madmouj
School of Engineering
University of Dayton
Dayton, USA
madmoujk1@udayton.edu

Alexander Walker
School of Engineering
University of Dayton
Dayton, USA
walkera35@udayton.edu

***Abstract*—This project presents the development of a lightweight yet effective image deblurring model based on the NAFNet (Nonlinear Activation-Free Network) architecture. The primary goal is to optimize deblurring performance while maintaining low computational cost, making the model suitable for real-world applications with limited hardware resources. Experiments were conducted using the GoPro image dataset, which consists of pairs of blurred and sharp images exhibiting various levels and types of motion blur. Several aspects of the model were systematically evaluated and adjusted, including different loss functions (Charbonnier, L1, SSIM, Edge), corresponding weight parameters, model width, encoder block configurations, and learning rate strategies. After fine-tuning these variations, a modified version of NAFNet was implemented, incorporating GELU activations and a modifiable number of NAFBlocks to test different variations. The results demonstrate that with optimized settings and very limited computational resources, the model can consistently achieve peak signal-to-noise ratios (PSNRs) of around 24.4 dB, with an average runtime of only 18 minutes. These findings confirm the model's capability to deliver enhanced deblurring performance while remaining computationally efficient, highlighting its potential as a practical baseline for image deblurring.**

***Index Terms*—Image Deblurring, NAFNet, PSNR, Loss Functions, Lightweight Models**

I. Introduction

With the constant advancements of artificial intelligence and machine learning, various forms of image deblurring models have been introduced, each with its own trade-offs between complexity, speed, and image restoration quality. One of these approaches, a Non-linear Activation Free Network, or NAFNet, has gained popularity because of its simple and efficient methodology [1]. This tactic uses a single staged U-Net approach to reduce complexity between blocks, alongside a simple gate activation function to allow for quick computational speed [5]. The U-Net is a convolutional neural network specifically designed for image segmentation with its large number of feature channels and implementation of both contracting and expanding paths, allowing for a more accurate segmentation map. The NAFNET model simplifies this by decreasing its reliance on nonlinear activation functions and instead using depthwise convolutions and otherwise cleaner operations. This has shown to increase efficiency while simultaneously surpassing the U-Net in restoration quality, leading to this model being the baseline for image restoration [3].

This increase in efficiency and performance in image deblurring tasks has a direct real-world impact, as the need for effective image deblurring spans a variety of different fields such as medical images, astronomy, and biometrics [2]. In medical imaging, cleaner visuals are essential for accurate diagnoses. In astronomy and biometrics, deblurring improves telescope imagery and enhances the accuracy of fingerprint and facial recognition. Beyond these, everyday technologies such as surveillance systems, traffic cameras, and smartphone photography benefit from crystal clear imagery. Given the wide range of applications, developing faster and more reliable deblurring methods remains critically important.

To advance this important area of research, this project builds on the current NAFNet implementation presented by Chen et al. [1]. Specifically, it investigates how changes in model parameters, different loss functions and loss weightings, and different learning rate parameters and schedulers affect the model's performance with regards to PSNR and efficiency, using the same specialized GoPro dataset for deblurring as Chen et al.

II. Related Research

Image deblurring itself is the process of recovering a sharp image from a blurred observation, where the blur could be caused by many things such as camera shake or motion blur [14]. This causes problems, as multiple sharp images could correspond to the same blurred input. Accurate deblurring plays an important role in many fields like astronomy and biometrics, where fine details are critical for analysis and decision-making, and its importance has skyrocketed within the past few decades.

Before the rise of deep neural networks, image restoration relied on techniques such as Wiener filters and RL (Richardson-Lucy) filters. While useful, these processes failed to fully leverage the information present in the original images [8]. Other traditional methods, such as sparsity-based methods, attempted to utilize all available image data but lacked computational complexity, and large blur kernels were especially difficult to handle [8]. The emergence of neural networks, particularly Convolutional Neural Networks (CNNs), marked a turning point. CNNs enabled models to learn deblurring directly from data, adapting to various types of blur and

eliminating the need for hand-crafted filters by learning useful features automatically [9].

After CNNs became widely utilized, advances kept happening at a very quick rate, with restoration quality of images constantly improving. The Scale-Recurrent Network introduced refining blurry images across multiple scales, assisting with large blur such as camera shakes without wasting computational resources, but it struggled with having a limited memory capacity [10]. DeblurGAN used adversarial learning to create realistic sharp images by prioritizing perceptual quality, but sometimes hallucinated details [11]. And transformer-based models such as Restormer captured long-range pixel interactions through a feed-forward network and attention mechanisms, but it made models very computationally complex [12].

While each new model in image deblurring built upon the strengths of its predecessors, they also introduced their own challenges due to differences in architecture, capabilities, and parameter choice. CNNs and transformer-based architectures achieved significant improvements, but they came at the cost of increased model complexity, larger parameter counts, and slower training times [13]. Deep hierarchies or expensive attention modules were required for a lot of implementations, which meant their applicability for real-world problems was reduced. This motivated the development of lightweight and efficient models.

In response to this, Chen et al. introduced the NAFNet model. This model removed nonlinear activations commonly used by other models, focused on efficient deblurring while maintaining accuracy, and used lightweight gating mechanisms [1]. Unlike previous methods, NAFNet uses a simple U-Net backbone and does not generally use linear activations such as RELU, GELU, and Swish, reducing the information lost while the features are being transformed. And the utilization of simple gates inside each NAFNet block assist with learning useful features, while different intra-block design schemes reduce memory and complexity [1].

Overall, computational costs are reduced from previous state-of-the-art (SOTA) methods while maintaining, and in some cases increasing, the peak signal-to-noise ratio (PSNR) on various datasets. NAFNet demonstrated that transformer-based designs are not the only way to advance the field of image deblurring, as efficient and lightweight models can perform very similarly while being resource-friendly. Its flexibility means it can lend itself to other tasks, such as image denoising and super-resolution, creating yet another advantage via its transferability.

This paper improves upon these adjustments by varying the types of losses used, including Charbonnier, L1, SSIM, and Edge loss; the parameters used in calculating the total losses, including beta values for changing the importance of SSIM, gamma values for adjusting the weight of Edge Loss, and delta values for changing the weight of the L1 Loss; and the parameters of the model itself, including the width of the model and the size of the encoder blocks, which varies the complexity of the NAFNet model.

Adjusting all of these parameters allows for evaluating the model's performance under various conditions, with the goal of maintaining a high PSNR while improving computational efficiency by reducing GPU memory usage and minimizing training and inference time. Specifically, the learning rate parameters are adjusted in hopes of positively affecting the speed of convergence. The specific methodologies and implementations explored throughout the experimentation process are discussed in the following section, where the different loss functions and NAFNet architecture, among other things, are explored.

III. Methodology

In order to implement the modified NAFNet model, a dataset containing paired blurred and sharp images was collected. The model is run on the blurred images, and then compared against the corresponding sharp images. Residual images are generated by comparing the original blurred image with both its sharp ground truth and the output produced by the modified NAFNet model to easily determine the model's performance.

The model is comprised of various components, some of which differ from a typical NAFNet implementation. First, the model extracts the features of the image via a 2-D convolution layer using GELU activation. This is the first disparity between the NAFNet model created by Chen et al, which does not use any sort of linear activation, but instead uses a simple multiplication activation function called SimpleGate [1]. The modified application makes the activation function smoother and better at picking out nuances within the features.

The next component of the model is the modified NAFBlock. There are three variations of these blocks, each with different functions [1]. The encoder NAFBlocks are used to increase dimensionality through depthwise convolution and to extract features at each level. The middle NAFBlocks are where the most intense computational processes are run, and they are collectively referred to as the bottleneck [1]. Within these blocks, GELU activation is used, which further adds to the complexity. Finally, the decoder blocks upsample the features from the previous levels to reconstruct a sharper image in two dimensions. In the modified model, an arbitrary number of encoding, decoding, and middle NAFBlocks can be used. The total number decides the complexity and computational resources necessary for the model to be run [1].

Before producing the final output, the image passes through one last convolution layer to ensure it can be visualized in three RGB channels. After processing through the decoding blocks and upsampling layers, the network's features still reside in a high-dimensional space, meaning they are not yet interpretable as a standard RGB image. This final convolution reduces the feature maps to exactly three channels, corresponding to the red, blue, and green parts of the image. This layer is then followed by a residual connection, where the original blurred input image is added back to the network's output. This residual learning strategy helps the model focus on predicting the differences between the sharp and blurred images, improving training stability and reducing convergence time.

Once the model has been completely run, the loss is calculated to guide model training. Several loss functions are combined to optimize both pixel-level accuracy and perceptual image quality. First, the L1 loss, or the mean absolute error (MAE), is calculated. This measures the direct pixel-wise difference between the output of the network and the sharp image, which helps encourage accurate reconstruction [15]. The equation of this is shown below in Equation 1 [19].

$$\mathcal{L}_{\text{L1}}(x, y) = \frac{1}{N} \sum_{i=1}^{N} |x_i - y_i| \tag{1}$$

Second, Charbonnier Loss is used, which is a smooth approximation of L1 loss that is more robust to noise and outliers [16]. As a result, the model will be protected against overcorrection, and will not overcompensate if a lot of blur exists. This equation is displayed in Equation 2.

$$\mathcal{L}_{\text{Charbonnier}}(x, y) = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(x_i - y_i)^2 + \epsilon^2} \tag{2}$$

Third, a Structural Similarity (SSIM) Loss is included to encourage the model to preserve important structural information, such as edges and textures. SSIM analyzes the difference between 2 images and outputs a result showing how similar the two images are, and the loss is one minus this value [17]. This equation is shown in Equation 3 below.

$$\mathcal{L}_{\text{SSIM}}(x, y) = 1 - \text{SSIM}(x, y) \tag{3}$$

Lastly, Edge Loss is applied by computing the Laplacian (edge maps) of both the network's output and sharp images and minimizing their difference, meaning the boundaries will be restored more efficiently. Edge pixels are directly optimized and sharper images are produced as a result [18]. This loss function is displayed in Equation 4.

$$\mathcal{L}_{\text{Edge}}(x, y) = \frac{1}{N} \sum_{i=1}^{N} |\Delta x_i - \Delta y_i| \tag{4}$$

These loss components are weighted together during training with specific tuning of the importance of each term, each of which are analyzed and varied during many tests. This final loss function is shown below in Equation 5. Alpha represents the weight for Charbonnier loss, beta for SSIM loss, gamma for edge loss, and delta for L1 loss.

$$\begin{aligned} \mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{Charbonnier}}(x, y) + \beta \cdot \mathcal{L}_{\text{SSIM}}(x, y) \\ + \gamma \cdot \mathcal{L}_{\text{Edge}}(x, y) + \delta \cdot \mathcal{L}_{\text{L1}}(x, y) \end{aligned} \tag{5}$$

This modified NAFNet model introduced many changes compared to Chen et al.'s paper. GELU activations are utilized, the number of NAFBlocks are varied by changing the amount of encoder blocks used, and a more complex and thorough loss function is used. The model takes a blurred input image, processes it through a streamlined encoder-bottleneck-decoder structure, and outputs a sharpened version of the input image, reinforced by residual learning. Training is conducted over 10 epochs, balancing available computational resources with the need for sufficient convergence. Throughout training, the model learns to minimize the constructed loss function, which can be adjusted both with the functions used and their respective parameters, promoting accurate pixel restoration. This methodology established a foundation for evaluating how variations in model parameters, loss parameters, and learning rates impact final model performance.

IV. Results and Discussion

The NAFNet model underwent numerous tests in order to find the most ideal results on the GoPro dataset. The dataset consists of pairs of blurred and sharp images and was specifically created for the purpose of image deblurring. Many different models have been trained using this dataset. There are 3214 blurred images, each with a size of 1280 x 720, with 2103 used for training and 1111 used for testing [21].

The first test conducted on the model was the loss test. Different tests were conducted, with each one incorporating a different combination of losses while outputting both the PSNR and the runtime (formatted like minutes:seconds). Charbonnier loss is incorporated into all of them, as it plays the biggest factor in the loss calculation, while the other losses are removed in some tests to determine their individual impact. As seen below in Table I, Test 1 has the highest PSNR, although it also has the highest runtime by a few seconds. Including all of the four losses leads to increased model complexity and allows for the tweaking of more parameters, shown in the next section of tests, while simultaneously maintaining the computational costs. This is why all four losses are used in the following tests. The respective parameters used are beta = 0.1, alpha = 1, gamma = 0.01, and delta = 0.05, while the number of training images remain constant at 500 for all tests.

TABLE I
Loss Test Results

Test	Losses Used	PSNR (dB)	Runtime
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, Edge, SSIM	24.06	17:39

An example set of images, consisting of the original blurred input image, corresponding sharp image, the NAFNet model output image, the residual image between the NAFNet output and the blurred image, and the residual image between the sharp and the blurred images for comparison, are all output to display the models capabilities. This is shown in Figure 1.

As shown in Figure 1, the model demonstrates effectiveness in highlighting areas requiring enhancement, as evidenced by the similarities between the two residuals. But at enhancing the images themselves the model struggles to provide a very clear enhancement from the blurred image. This is almost certainly due to the lack of testing epochs and training images, set to 10 and 500 respectively, which are limited due to GPU RAM and time restraints encountered due to Google Colab. Nonetheless, the makings of a very efficient and streamlined model are

Fig. 1. Example set of Images Output by the Model

shown, with the parameters being further tuned to increase reliability.

The second test conducted was the loss parameter test. This test included all of the loss types (like in test 1 of Table I above), but altered the beta, gamma, and delta parameters accordingly to see how each one affects the PSNR and runtime of the NAFNet model. The value of alpha was kept at 1 and is not altered in any of these tests. As can be seen by the results in Table II below, the values of the PSNR and runtime are generally around the values found in Test 1. Test 14 with the parameters beta = 0.2, alpha = 1, delta = 0.05, and gamma = 0.01 provided the best results with a 24.39 dB PSNR, the highest recorded PSNR thus far, and a 17:56 runtime, among the lowest of all the tests. Therefore, these parameters will be used for the following tests in determining the optimal model parameters (width and number of encoder blocks), and for determining the optimal learning rate.

TABLE II
LOSS WEIGHT TUNING RESULTS

Test	Beta	Gamma	Delta	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

The third set of tests involve changing the width of the model, along with adjusting the number of encoder blocks used, which inherently changes the total amount of NAFBlocks utilized. The goal is to see whether an increase in the width of the model or in encoder block amount will lead to an increase in model performance while maintaining run and training time. As seen in Table III , increasing the model width to 48 or increasing the amount of encoder blocks to 12 with size [2,2,2,6](expanding the total number of NAFBlocks to 30 as the number of middle blocks and decoder blocks stay constant at 10 and 8, respectively) do not have much of a positive impact on PSNR or runtime. Runtime increases by over a minute if either of these factors are increased, while PSNR slightly decreases even with the longer training time. After these tests are performed, it is clear that the configuration with the width equaling 32 and the encoder block configuration equaling [1,1,2,6] provides the most desired combination between PSNR and runtime. This configuration along with the previously selected values will be used in the following tests which determine the optimal learning rate.

TABLE III
ENCODER CONFIGURATION TEST RESULTS

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

The fourth set of tests include varying the learning rate to see how training is affected. For all of the previous tests, a cosine annealing learning rate is used, starting at 9.76e-4 for the first epoch and ending at 1e-6 for the tenth epoch. This means that the majority of the increase in PSNR is attributed to the first few epochs, as that is when learning is the strongest and the biggest changes will be made. The last few epochs demonstrate steady improvements using this learning rate, as it builds upon what is learned previously. In the following two tests, a constant learning rate of 5e-4 and a constant learning rate of 1e-3 are used to see how that will affect the rate of learning. Table IV below shows the results from tests 21-23 with the varying of this parameter, and it can be seen that cosine annealing is the best learning rate parameter with the fastest runtime and the highest PSNR.

TABLE IV
CONSTANT LEARNING RATE TEST RESULTS

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

V. CONCLUSION

While the current configuration produces reliable results, there are many important observations that are made. The balance between PSNR and runtime suggests that merely increasing the model width or the number of encoder blocks does not equate to better results, as the model with the least width and number of blocks outperformed the other tests. Additionally, loss tuning results indicate that a moderate weighting for

the different parameters provides an optimal tradeoff between sharpness and runtime. These findings suggest that targeted fine-tuning of architecture and loss parameters can produce effective lightweight models for deblurring tasks.

The final experiments involved evaluating fixed versus cosine learning annealing rates. The results showed that cosine annealing allowed for the PSNR to converge relatively quickly, as the learning rate decreasing throughout the epochs meant much of the learning happened earlier in training. The constant learning rate of 5e-4 performed pretty close to cosine annealing with its PSNR, but it was a little lower and the runtime was a little longer. The learning rate of 1e-3 proved to be too big, as its PSNR was the lowest. So the overall configuration that proved to be the best for the model is: width = 32, encoder blocks = [1,1,2,6], alpha = 1, beta = 0.2, gamma = 0.01, delta = 0.05, and a cosine annealing learning rate and all four loss functions.

Overall, these outcomes demonstrate the adaptability and practical value of the NAFNet, especially when trained under tight computational constraints. With extended training time, more training images, and higher-end GPUs, further gains in PSNR and visual quality are expected to be achieved, making this approach very scalable in real-world deployment.

ACKNOWLEDGMENT

Thank you to Dr. Theus Aspiras for guiding the team and providing the resources and information needed to complete this research.

REFERENCES

[1] L. Chen, X. Chu, X. Zhang, and J. SunG. Eason, "Simple baselines for image restoration," MEGVII Technology, Beijing, CN.

[2] C. Hansen, DTU, http://www.imm.dtu.dk/p̃cha/HNO/chap1.pdf.

[3] D. Cochard, "NAFNET : A machine learning model to Deblur images," Medium, https://medium.com/axinc-ai/nafnet-a-machine-learning-model-to-deblur-images-a0a03e94feae (accessed Apr. 29, 2025).

[4] W. Gu et al., "A deep learning model, NAFNET, predicts adverse pathology and recurrence in prostate cancer using mris," Nature News, https://www.nature.com/articles/s41698-023-00481-x.

[5] S. Jian, "Nafnet: Ai model details," nafnet — AI Model Details, https://www.aimodels.fyi/models/replicate/nafnet-megvii-research (accessed Apr. 29, 2025).

[6] Saiwa, "What is the image deblurring Deep learning method?," Medium, https://medium.com/@saiwadotai/what-is-the-image-deblurring-deep-learning-method-b40621406ae3.

[7] F. Albluwi, V. A. Krylov and R. Dahyot, "Image Deblurring and Super-Resolution Using Deep Convolutional Neural Networks," *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, Aalborg, Denmark, 2018, pp. 1-6, doi: 10.1109/MLSP.2018.8516983.

[8] Li, C. A Survey on Image Deblurring. (2022), https://arxiv.org/abs/2202.07456

[9] S. Nah, T. H. Kim, and K. M. Lee, "Deep multi-scale convolutional neural network for dynamic scene Deblurring," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 257–265, Jul. 2017. doi:10.1109/cvpr.2017.35

[10] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia, "Scale-recurrent network for deep image Deblurring," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8174–8182, Jun. 2018. doi:10.1109/cvpr.2018.00853

[11] Z. Li, "Image Deblurring using GAN," arXiv.org, https://arxiv.org/abs/2312.09496.

[12] S. W. Zamir *et al.*, "Restormer: Efficient Transformer for high-resolution image restoration," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022. doi:10.1109/cvpr52688.2022.00564

[13] T. Thangavel, "Limitations of transformer architecture," Medium, https://medium.com/@thirupathi.thangavel/limitations-of-transformer-architecture-4e6118cbf5a4.

[14] "Home," GoPhotonics.com, https://www.gophotonics.com/community/what-is-image-deblurring.

[15] A. Shekhar, "What are L1 and L2 loss functions?," Outcome School — Get High Paying Tech Job, https://outcomeschool.com/blog/l1-and-l2-loss-functions (accessed Apr. 29, 2025).

[16] "Loss functions¶," Loss Functions - machine learning note documentation, https://machine-learning-note.readthedocs.io/en/latest/basic/loss_functions.html).

[17] J. Roubaud, "Working with structural similarity index," Medium, https://jack-roubaud.medium.com/working-with-structural-similarity-index-86ace619f408 (accessed Apr. 29, 2025).

[18] G. Seif and D. Androutsos, "Edge-Based Loss Function for Single Image Super-Resolution," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, 2018, pp. 1468-1472, doi: 10.1109/ICASSP.2018.8461664.

[19] V. Yathish, "Loss functions and their use in neural networks," Towards Data Science, https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9/.

[20] S. A. Biyouki and H. Hwangbo, "A comprehensive survey on Deep Neural Image deblurring," arXiv.org, https://arxiv.org/abs/2310.04719.

[21] "Papers with code - gopro dataset," Dataset — Papers With Code, https://paperswithcode.com/dataset/gopro.