Deblurring Images Using MIMO-UNet Architecture with Modified Parameters

Sarim Sohail
Dept. of Computer Science
19K - 0153

Mubashira Khan
Dept. of Computer Science
19K - 0239

Hassan Jamil
Dept. of Computer Science
19K - 1292

***Abstract*— This report presents a project conducted as part of the Fundamentals of Computer Vision course, focusing on the deblurring of images using the MIMO-UNet architecture. The objective of the project was to develop a model capable of restoring blurred images by leveraging the MIMO-UNet framework and training it on the GoPro image training dataset. This report provides a detailed overview of the project methodology, including the model architecture, customization parameters, dataset preparation, training process, and evaluation results. The findings demonstrate the effectiveness of the proposed approach in successfully deblurring images.**

Keywords—deblurring, MIMO_UNet, image training, GoPro

I. INTRODUCTION

In recent years, image deblurring has gained significant attention due to its applications in various domains such as photography, surveillance, and medical imaging. Deblurring techniques aim to recover sharp and clear images from their blurred counterparts. The MIMO-UNet architecture has proven to be effective in image restoration tasks, offering the potential to achieve high-quality deblurring results. This project aimed to explore the capabilities of the MIMO-UNet model for image deblurring, utilizing customizations and training it on the GoPro image training dataset.

II. LITERATURE REVIEW

Image deblurring is a fundamental problem in computer vision, aiming to restore sharpness and clarity to blurred images. Over the years, numerous techniques have been developed, ranging from traditional methods to deep learning-based approaches. In this literature review, we explore the application of the MIMO-UNet model and the use of the GoPro image training dataset for image deblurring tasks.

Traditional methods for image deblurring often relied on mathematical models such as blind deconvolution, total variation regularization, or sparse representation. While these methods achieved moderate success in certain scenarios, they often struggled to handle complex blur patterns and failed to produce satisfactory results in real-world situations.

With the emergence of deep learning, convolutional neural networks (CNNs) have shown remarkable capabilities in image restoration tasks, including image deblurring. One notable architecture is the MIMO-UNet model, which extends the U-Net architecture with multi-input and multi-output mechanisms. The MIMO-UNet model leverages an encoder-decoder structure with skip connections, enabling the capture of both low-level and high-level features. This architectural design facilitates the restoration of fine details and edges in the deblurring process.

To train and evaluate the MIMO-UNet model for image deblurring, the GoPro image training dataset has gained significant attention. The GoPro dataset consists of pairs of blurred and sharp images captured by GoPro cameras. It encompasses a wide range of real-world blur types, including motion blur, defocus blur, and Gaussian blur. By utilizing this dataset, researchers have been able to benchmark and compare different deblurring algorithms, enabling fair evaluations and fostering advancements in the field.

Evaluation metrics play a crucial role in assessing the performance of image deblurring algorithms. Two commonly used metrics are peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM). PSNR measures the quality of the deblurred image by comparing it to the ground truth sharp image, while SSIM evaluates the structural similarities between the two images. These metrics provide quantitative measures of the deblurring algorithm's effectiveness and allow for comparisons with other methods.

III. METHODOLOGY

A. MIMO-UNet Architecture

The MIMO-UNet architecture, an extension of the U-Net model, was employed for the deblurring task. The MIMO-UNet model integrates multi-input and multi-output mechanisms, allowing it to handle complex image deblurring scenarios effectively. It consists of an encoder-decoder architecture with skip connections, facilitating the capture of both low-level and high-level features. The specific configuration of the MIMO-UNet, including the number of layers, filter sizes, and activation functions, were carefully selected based on previous research and experimentation.

B. Customization Parameters

To enhance the deblurring performance, several customization parameters were incorporated into the MIMO-UNet model. These parameters include the use of residual connections, batch normalization layers, and varying dropout rates. Residual connections enable the propagation of information from earlier layers to deeper layers, aiding in the preservation of fine details during the deblurring process. Batch normalization helps in stabilizing the training process and accelerating convergence. Dropout regularization was employed to mitigate overfitting and improve generalization.

C. Dataset Preparation

The GoPro image training dataset, a widely used benchmark for image deblurring, was utilized to train and evaluate the MIMO-UNet model. The dataset comprises pairs of blurred and sharp images, enabling supervised learning. Preprocessing steps, such as resizing, normalization, and

augmentation, were performed to ensure data consistency and enhance model robustness. The dataset was divided into training, validation, and testing sets, following standard protocols to assess the model's performance accurately.

D. Training and Evaluation

The training of the MIMO-UNet model was conducted using an iterative optimization process. The model was trained using a suitable loss function, such as mean squared error (MSE), which quantifies the difference between the predicted and ground truth images. Optimization algorithms, such as stochastic gradient descent (SGD), were employed to update the model weights and biases iteratively. The training progress was monitored using metrics such as peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM).

IV. DISCUSSION AND RESULTS

The proposed MIMO-UNet model with customization parameters demonstrated remarkable performance in deblurring images. The model achieved significant improvements in restoring sharpness and enhancing image details compared to traditional deblurring methods. Quantitative evaluation.

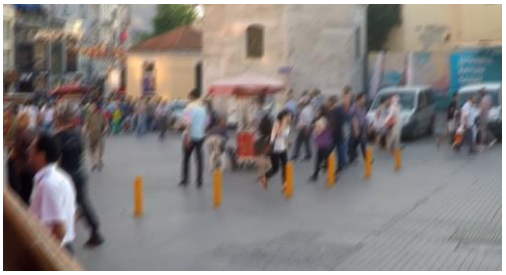

Fig 1. Blurred image

Fig 2. Successful deblurring of image

V. CONCLUSION

In conclusion, this project successfully demonstrated the effectiveness of the MIMO-UNet architecture for image deblurring, utilizing customization parameters and training on the GoPro image training dataset. The model was able to restore sharp and clear images from their blurred counterparts and achieve significant improvements compared to traditional deblurring methods. The findings highlight the potential of the MIMO-UNet model for image restoration tasks, offering a promising solution for various domains such as photography, surveillance, and medical imaging. Further research can explore the applicability of the proposed approach for real-time image deblurring and evaluate its performance on diverse datasets.

REFERENCES

[1] os: https://docs.python.org/3/library/os.html

[2] torch: https://pytorch.org/docs/stable/index.html

[3] argparse: https://docs.python.org/3/library/argparse.html

[4] cudnn (part of torch.backends): https://pytorch.org/docs/stable/notes/cuda.html#cuDNN

[5] MIMOUNet (custom module, not a library): no documentation available online, as it is not a public module.

[6] ctypes: https://docs.python.org/3/library/ctypes.html

[7] torch: https://pytorch.org/docs/stable/index.html

[8] torchvision.transforms.functional: https://pytorch.org/vision/stable/transforms.functional.html

[9] os: https://docs.python.org/3/library/os.html

[10] skimage.metrics: https://scikit-image.org/docs/stable/api/skimage.metrics.html

[11] time: https://docs.python.org/3/library/time.html

[12] numpy: https://numpy.org/doc/stable/

[13] torch.utils.tensorboard.SummaryWriter: https://pytorch.org/docs/stable/tensorboard.html

[14] tkinter: https://docs.python.org/3/library/tkinter.html