Mini Project 2 :Image Deblurring

COURSE NAME

GNR:638

Authors:

Vaibhav Rathore (23M2156)
Shubranil B (23M2162)
Vaibhav Bhatnagar (23M1062)

April 10, 2024

Indian Institute of Technology

Contents

1 StripFormer

In our Project we used a modified version of StripFormer and used the model for **14.96 Miilion parameters**[1].Here is the Figure containing the model architecture, see Figure 4.

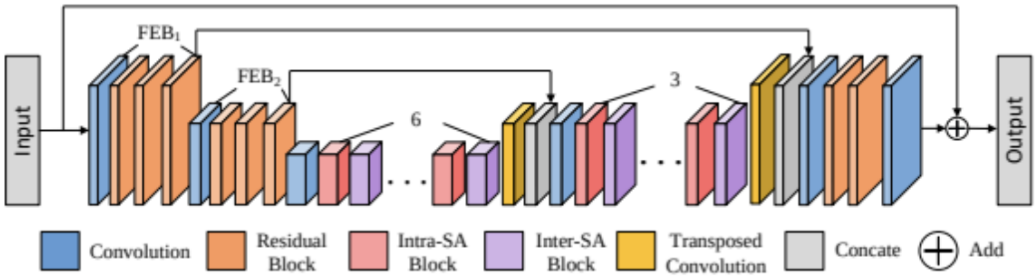

Figure 1: Stripformer : Transformer-based architecture

1.1 Architecture Details

The Stripformer is a transformer-based architecture designed to address blurring in images captured in dynamic scenes, where blur artifacts may vary in orientation and magnitude. It leverages intra- and inter-strip tokens to extract patterns of blur with different orientations and magnitudes. Intra-strip attention focuses on horizontal and vertical strip-wise features, while inter-strip attention examines relationships between strips. These attention mechanisms allow for the reweighting of features to accommodate both short- and long-range blur magnitudes.

The architecture of Stripformer consists of a residual encoder-decoder framework starting with two Feature Embedding Blocks (FEBs) to generate embedding features. These FEBs downsample the input, resulting in an output resolution one-fourth of the input size. The model then employs a stack of convolution layers interleaved with Intra-SA and Inter-SA blocks on the smallest and second-smallest scales. These blocks perform horizontal and vertical intra-strip or inter-strip attention to capture blur with various magnitudes and orientations.

Upsampling is achieved using transposed convolution, and the output features are concatenated with those from the encoder on the same scale. The architecture concludes with two residual blocks and a convolution layer with a residual connection to the input blurred image.

1.1.1 FEATURE EMBEDDING BLOCK (FEB)

The Feature Embedding Block (FEB) is a key component of the Stripformer architecture, designed to address issues associated with traditional transformer models. Instead of flattening image patches into tokens, which can lead to the loss of spatial pixel correlations and require excessive parameters, the FEB employs convolutional layers and residual blocks. This approach enables the FEB to generate feature embeddings while preserving spatial information, enhancing the model's ability to capture complex patterns in the input image.

1.1.2 INTRA-SA and INTER-SA BLOCKS

The core of Stripformer is Intra-SA and Inter-SA blocks. We detail their designs as follows:

Intra-SA block : In Stripformer comprises two parallel branches: horizontal intra-strip attention (Intra-SA-H) and vertical intra-strip attention (Intra-SA-V). Input features, represented as X with dimensions $HxWxC$, undergo LayerNorm and 11 convolution (Conv) operations with C filters. This processing is described as: Equation 1.

$$(X^h, X^v) = Conv(Norm(X)) \tag{1}$$

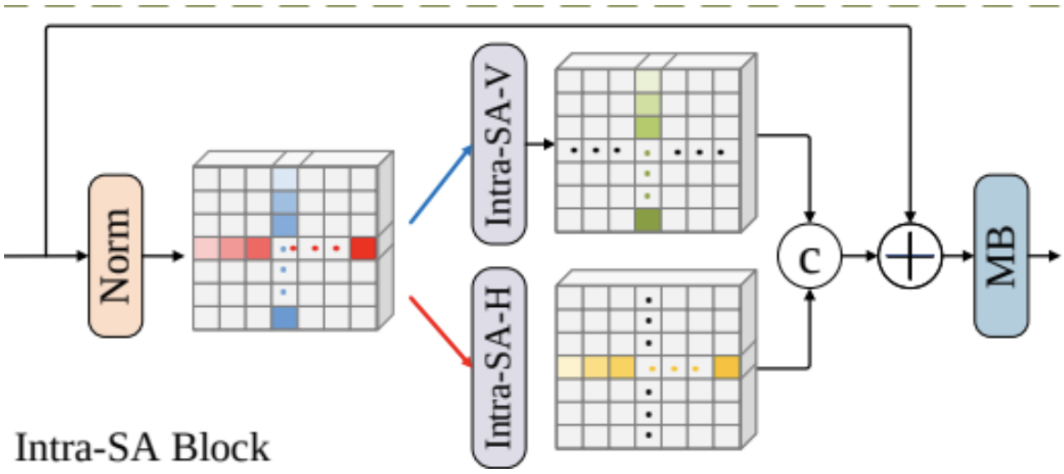

Figure 2: Intra-SA block

For the horizontal intra-strip attention, we split the input features X^h into H non-overlapping horizontal strip $X_i^h \in R^{WxD}$, $i = \{1, 2, ..., H\}$. Each strip X_i^h has W tokens with D dimensions. Next, we generate queries, keys, and values associated with X_I^h as Q_{ij}^h , K_{ij}^h , and $V_{ij}^h \in R^{WxD/M}$ m for the multi-head attention mechanism as
Equation 2.

$$(Q_{ij}^h, K_{ij}^h, V_{ij}^h) = (X_i^h P_j^Q, X_i^h P_j^K, X_i^h P_j^V) \tag{2}$$

The multi-head attended feature $O_{ij}^h \in R^{WxD/m}$ m for one horizontal strip is calculated as : Equation 3.

$$O_{ij}^h = \text{softmax}\left(\frac{Q_{ij}^h (K_{ij}^h)^T}{D\sqrt{D/m}}\right) V_{ij}^h \tag{3}$$

whose space complexity is $O(W^2)$. We concatenate the multi-head horizontal features $O_{ij}^h \in RWxD/m$ along the channel dimension to generate $O_i^h \in RWxD$ and fold all of them into three-dimensional tensors $O^h \in R^{HxWxD}$ as the Intra-SA-H output. Symmetrically, the vertical intra-strip attention produces the multi-head attended feature for one vertical strip, denoted as $O_{ij}^V \in R^{HxD/m}$, whose space complexity is $O(H^2)$. After folding all the vertical features, the Intra-SA-V output denotes as $O^v \in R^{HxWxD}$. We then concatenate them to feed into a 11 convolution layer with a residual connection to the original input features X to obtain the attended features $O^attn \in R^{HxWxC}$ as

1.1.3 Loss Functions

Contrastive Learning, an effective self-supervised technique , enables models to derive universal features from data, even in the absence of labeled information. By discerning similarity and dissimilarity among data points, contrastive learning has found utility in various vision tasks , facilitating the clustering of "positive" (similar) pairs while pushing apart "negative" (dissimilar) pairs within the feature space. Motivated by recent advancements , we incorporate contrastive learning into our framework to ensure that the deblurred output closely resembles the ground truth image while maintaining dissimilarity with the blurred input. Considering a blurred input X, its corresponding deblurred result R, and the sharp ground truth S, where X, R, and $S \in R^{HW3}$, we treat X, R, and S as the negative, anchor, and positive samples, respectively. The contrastive loss function is formulated as: Equation 4.

$$L_con = \frac{L1((S) - (R))}{L1(\psi(X) - \psi(R))} \tag{4}$$

Here, ϕ extracts hidden features from conv3-2 of the pre-trained VGG-19 model , and L1 represents the L1 norm. Minimizing Lcon facilitates pulling the deblurred result R close to the sharp ground truth S (the numerator), while simultaneously pushing R away from its blurred input X (the denominator) within the same latent feature space.

Optimization: The loss function of Stripformer for deblurring is as follows, where L_{char} and L_{edge} represent the Charbonnier loss and edge loss, respectively, akin to those employed in MPRNet, and L_{con} denotes the contrastive loss. Here, we set $\lambda_1 = 0.05$, and $\lambda_2 = 0.0005$. Equation 5.

$$L = L_{char} + \lambda 1 L_{edge} + \lambda 2 L_{con} \tag{5}$$

At any given stage S, instead of directly predicting a restored image XS, the proposed model predicts a residual image RS to which the degraded input image I is added to obtain: $XS = I + RS$. We optimize our MPRNet end-to-end with the following loss function: Equation 6.

$$L = \sum_{S=1}^{3} [L_{char}(X_S, Y) + \lambda L_{edge}(X_S, Y)] \tag{6}$$

Here, Y represents the ground-truth image, and Lchar is the Charbonnier loss, given by: Equation 7.

$$L_{char} = \sqrt{\|X_s - Y\|^2 + \epsilon^2} \tag{7}$$

with ϵ empirically set to 10^-3 for all experiments. Additionally, Ledge is the edge loss, defined as: Equation 8.

$$L_{edge} = \sqrt{\|\Delta(X_s) - \Delta Y\|^2 + \epsilon^2} \tag{8}$$

1.2 Training Details

1.2.1 Loss Function:

The loss function used is $Stripformer_Loss()$, which is a custom loss function for the StripFormer model designed for Image deblurring.

1.3 Optimizer:

Adam optimizer is used with a learning rate of 0.001 and betas values set to (0.9, 0.999). The optimizer is initialized to optimize the parameters of the model.

1.3.1 Loading Pretrained Model and Optimizer:

Our code also loads a pretrained model and optimizer state from a specified checkpoint path $(checkpointpath)$ using torch.load(). The model's state dictionary and optimizer's state dictionary are loaded separately from the checkpoint and assigned to the corresponding model and optimizer. Both model and optimizer are then moved to the specified device (likely GPU) using .to(device). This is done to ensure that if the training is stopped , we can restart it from there.

1.3.2 Training Loop:

The training loop runs for a specified number of epochs (num epochs). Within each epoch, the training data loader (data loader) is iterated over. For each batch of images and corresponding sharp images:
The images and sharp images are moved to the specified device. The optimizer gradients are zeroed.
The model is used to predict outputs from the input images. The loss is calculated using the defined loss function (Stripformer Loss()).
Backpropagation is performed (loss.backward()) and the optimizer is stepped to update the model parameters (optimizer.step()).
Periodically, every 10 steps, training loss and PSNR (Peak Signal-to-Noise Ratio) are printed for monitoring the training progress.
Additionally, a random image from the test dataset is selected, processed through the model, and its PSNR is printed for monitoring the model's performance on unseen data.
After each epoch, a model checkpoint is saved, including the current epoch number, model state dictionary, and optimizer state dictionary

1.4 Quantitative Results

Due to limitations in computational resources, the model was trained with a batch size of 2 and for a limited number of epochs. This might explain the moderate PSNR value of 8 achieved on the test set.

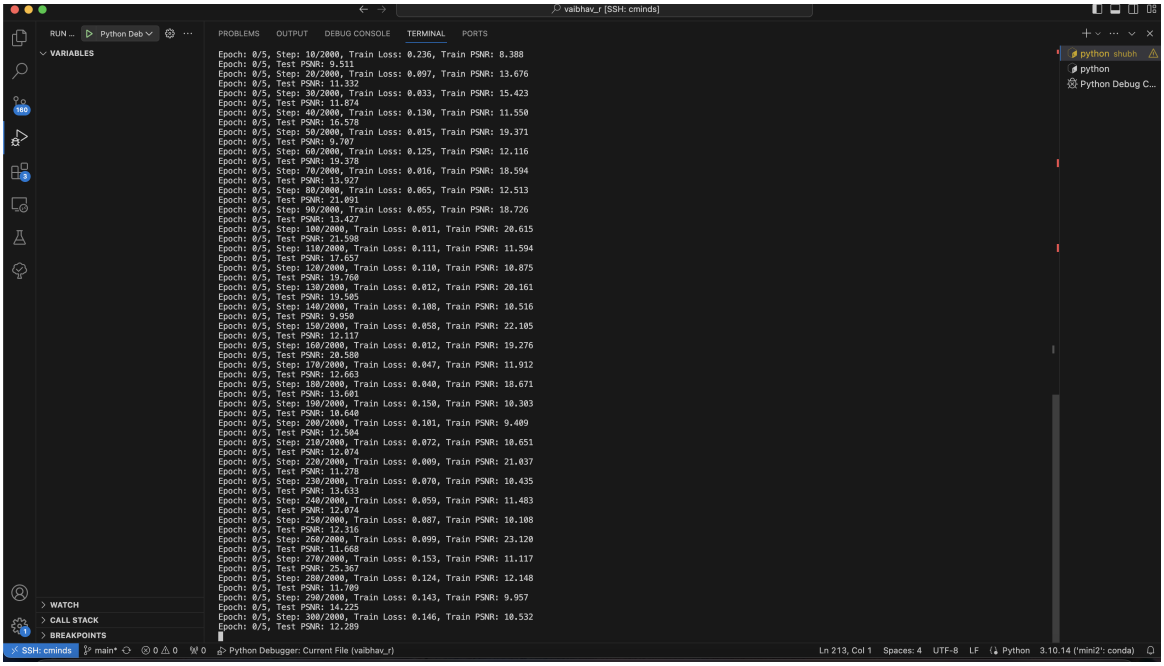

Figure 3: Training PSNR and Testing PSNR

1.5 Qualitative Results

The restored images are not fully sharp , but they are much improved quality .We are attaching the generated images in here.

Figure 4: Restored Image

References

[1] Tsai, F.-J., Peng, Y.-T., Lin, Y.-Y., Tsai, C.-C., and Lin, C.-W. (2022). Stripformer: Strip transformer for fast image deblurring.