Deep Learning for Computer Vision
Mini-Project 2 Report

Image Deblurring

Abhijeet Manoj Pal (200100107)
Pushkraj (200040112)
Aditya Malhotra (200040010)

April 9, 2024

1 Introduction

We have a dataset of 24000 Images. We initially downscaled/resized the images to size (256,448). Then we created the dataset B by randomly choosing a gaussian kernel out of the given three:

- Kernel size = 3X3, sigma = 0.3
- Kernel size = 7X7, sigma = 1
- Kernel size = 11X11, sigma = 1.6

Using this we obtained a set B of 24000 Images. Then we designed a network for Deblurring the images. The network was inspired by the Unet Architecture. The architecture is explained in the section below. Then we trained the model on this training data as constructed above and it was tested on the test dataset given of 150 images. Finally the results obtained are displayed in the Results section.

2 Architecture Used

The architecture used for deblurring was inspired by UNet. The architecture is as shown in the figure 5. The architecture image is also shared in the zip file as architecture.png.

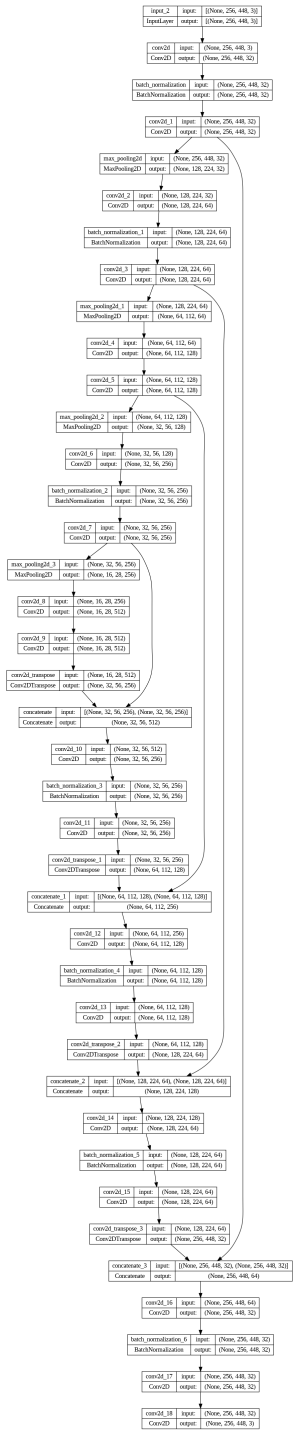

Figure 1: Architecture of Image Debluring Model

2.1 Architecture Explanation

Like the UNet architecture, the architecture is composed of a symmetric encoder decoder model – an encoder comes first, then a decoder. While the decoder uses transposed convolutions to enable localization, the encoder uses the convolutional and pooling layers to extract features and capture context from the image data. To maintain gradient passage and spatial information during training, skip connections are made between the encoder and the decoder. This helps with the reconstruction of intricate structures.

The encoder consists of multiple conv blocks, each comprising of convolutional layers followed by activation functions (ReLU) is used here. This is followed by batch normalization. The maxpooling layers help to reduce the spatial dimensions and increase the scope of the features in the original space of the image.

The decoder consists of upsampling layers which are used to restore the spatial dimensions. We then concatenate the feature maps from the encoding layers to help in precise localization of features and also in their reconstruction.

The skip connections concatenate features from the encoder to the decoder. The help in getting fine-grained details lost during downsampling in the maxpool layers of the encoder.

The final-layer consists of a convolutional layer with sigmoid activation function. Batch normalization is used to mitigate issues of vanishing gradients. Transposed convolutional layers help in upsampling and reconstruct the input image.

Table 1: Parameter Summary

Parameter Type	Parameters	Size
Total params	7763491 (7.7M)	29.62 MB
Trainable params	7761827	29.61 MB
Non-trainable params	1664	6.50 KB

3 Training Details

- The blurred and clear images are loaded using data generators named blurred_generator and clear_generator

- A custom generator called train_generator was defined to yield batches of blurred and clear training images
- We have used three loss functions namely MSE, MAE and SSIM index
- The number of training samples is calculated based on the number of files in the training set
- The number of steps per epoch is determined based on the batch size and the number of training samples
- We have defined two callbacks. EarlyStopping : This monitors the MSE loss on the validation set and stops the training if the loss does not improve for a number of specified epochs.
- ReduceLROnPleateau: This reduces the learning rate if MSE loss on the validation set does not improve for a certain number of epochs.
- The model is compiled using the Adam optimizer with learning rate of 0.001
- The loss function is set to MSE loss, we have also use metrics such as PSNR, MSE, MAE, SSIM are also printed during training
- The validation data path is obtained and loaded in batches using validation_generator
- **Training**:
- The model is trained for 25 epochs
- Training data is provided via train_generator()
- Validation data provided via validation_generator()
- Training is monitored, and is stopped if validation loss does not improve (early_stop_callback)
- Learning rate is reduced if validation loss plateaus (reduce_lr_callback)

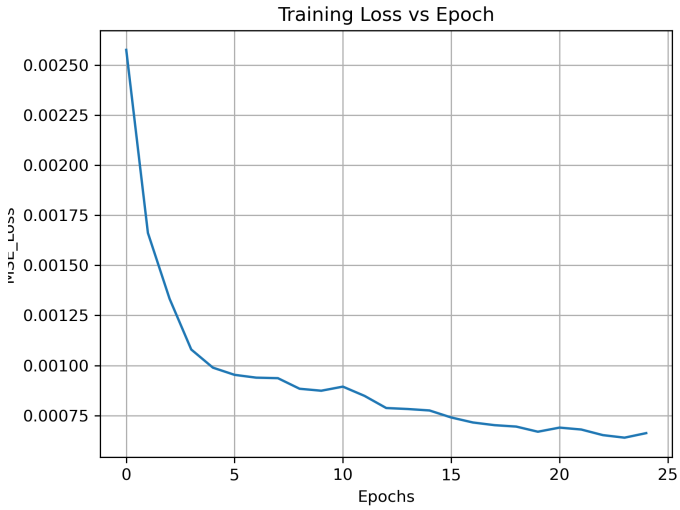

Figure 2: Training MSE loss vs epoch

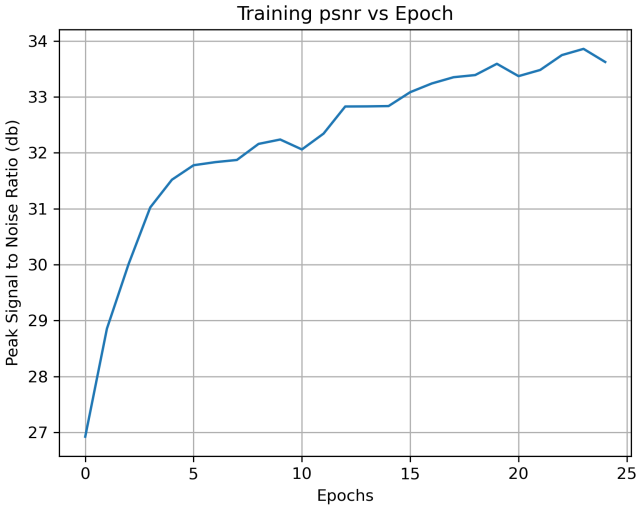

Figure 3: Training PSNR vs epoch

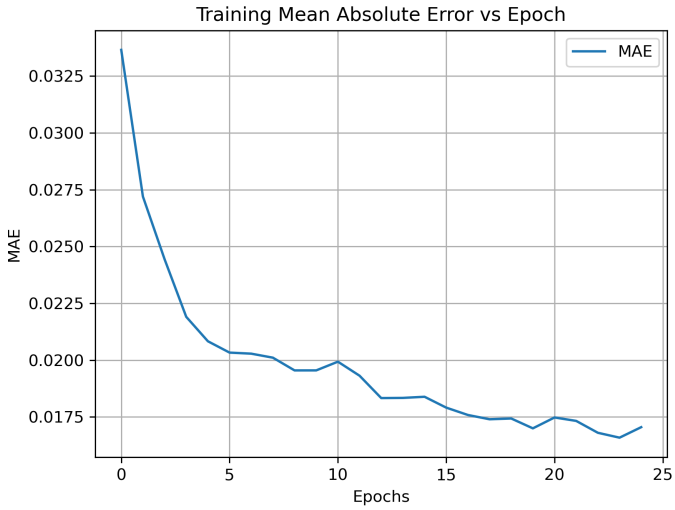

Figure 4: Training MAE vs epoch

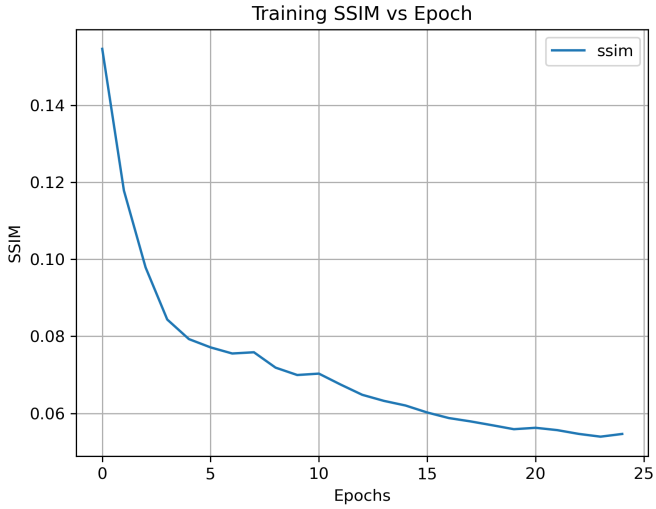

Figure 5: Training SSIM vs epoch

4 Results

We predict deblurred images using the trained model on the test dataset given in mp2_test. The results are shown. Then the given evaluation script is run and the average PSNR is calculated.
The obtained PSNR between the two folders is : 30.1106

Figure 6: Blur vs predicted vs Sharp

Figure 7: Blurred vs Predicted vs Sharp

4.1 Qualitative Results

We observe that the model is able to delblur the images quite well. Though it's not able to deblur the images perfectly but as seen in the image 6 we can see that the surroundings such as grass is clearly seen, though the face of the tiger is still quite blurred. But with open eyes, we are able to distinguish that it is a tiger not any other animal. Also the edges on the rock are clearly visible. Contrast is quite good of the predicted image. Also the trees in the background are much clear than the blurred image.
In the second image 7 we see that the face of the man's face is not visible in the blurred image whereas in the prediced_deblurred and the sharp_image the face is cearly visible same goes for the face of the girl.

4.2 Quantitative Results

Table 2: PSNR Values

Image	PSNR b/w blurred & sharp (dB)	PSNR b/w predicted & sharp (dB)
7	25.1385	29.83995
6	24.78206	28.65352
All Images Average	26.6817	30.1106