GNR638: Mini Project 2

Avadhoot Gorakh Jadhav(210050027)
Gohil Megh Hiteshkumar(210050055)

April 2024

Abstract :

The goal of the project is to deblur the given blur images to get sharp images. In this project, we will try different models and loss functions as well as hypertune parameters to increase the performance of the model. The metric that we will use in this project is PSNR (peak signal to noise ratio) value.

(a) Blur image

(b) Sharp image produced by Model

(c) Blur image

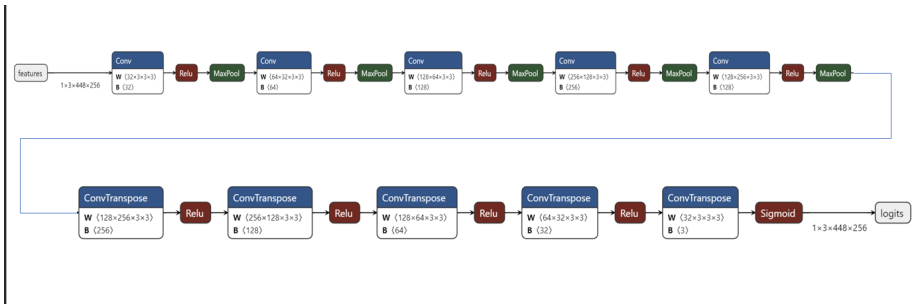
(d) Sharp image produced by Model

Pre-processing of data :

We have a total of 24000 sharp images. First we downsampled the images to 448×256 size. To generate images for training, we are applying Gaussian blurring with radius values 0.3, 1, 1.6 with kernel size 3, 7, 11 respectively. After the sharp images are blurred, we have total 72000 images for training.

Models

1. DeblurNet Model



This Model is an encoder-decoder model. The architecture of the model is as follows:

Encoder: The encoder part of the network is made of many 2D convolutional layers followed by batch normalization (to prevent internal covariate shift) and ReLU activation functions.The encoder of the network is useful for extracting features from the input image. Batch normalization normalizes the output of each layer, which helps to stabilize and speed up the training process. ReLU activation functions is used to get non-linearity to the model so as to learn complex relationships between input and output.The exact structure of convolutional layer, Batch Normalization layer and activation layers is as follows:

- Conv2d(3, 32, 3) : In channels = 3,Output channels = 32, Kernel size = 3
- BatchNorm2d(32) : Number of features = 32
- ReLU : Activation function = ReLU
- MaxPool2d(2) : Kernel size = 2, Stride = 2

The above layers are repeated with increasing number of output channels

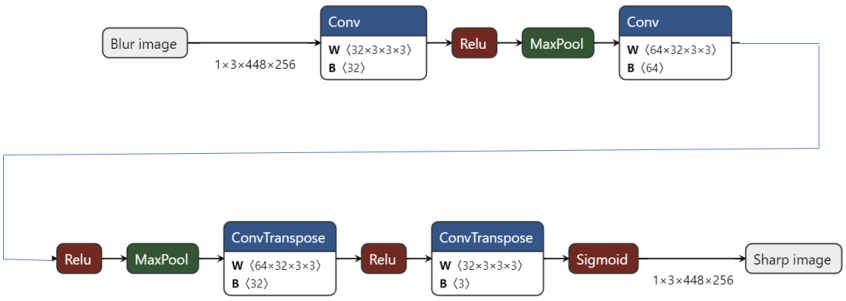
Decoder: The decoder of the network upsamples the feature maps learned by the encoder to reconstruct the blurred image to get a sharp image. It consists of several transposed convolutional layers followed by ReLU activation functions. The exact structure of all the layers is as follows:

- ConvTranspose2d(128, 256, 3) : Input channels = 128, Output channels = 256, Kernel size = 3
- ReLU : Activation function
- ConvTranspose2d(128, 256, 3) : In channels = 128, Output channels = 256

- ConvTranspose2d(256, 128, 3) : In channels = 256, Output channels = 128
- ConvTranspose2d(128, 64, 3) : In channels = 128, Output channels = 64
- ConvTranspose2d(64, 32, 3) : In channels = 64, Output channels = 32
- ConvTranspose2d(32, 3, 3) : layer to reconstruct RGB image in 3 channels
- Sigmoid : Activation function

Total number of parameters of this model: 1.3 Million

2. Reduced DeblurNet



This Model has similar architecture as DeblurNet Model, but have less number of convolutional layer. Some of the convolutional layer from Deblurnet are removed to get this model.

Encoder:

- Conv2d(3, 32, 3) : Input channels = 3,
- Output channels = 32, Kernel size = 3
- Stride = 1, Padding = 1
- BatchNorm2d(32) : Number of features = 32
- ReLU : Activation function = ReLU
- MaxPool2d(2) : Kernel size = 2, Stride = 2
- Conv2d(32, 64, 3) : Input channels = 32,

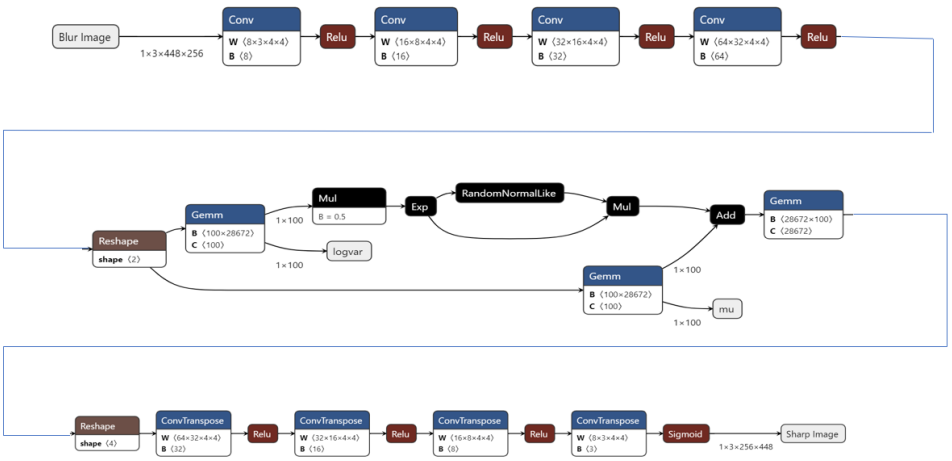
- Output channels = 64, Kernel size = 3
- Stride = 1, Padding = 1
- BatchNorm2d(64) : Number of features = 32
- ReLU : Activation function = ReLU
- MaxPool2d(2) : Kernel size = 2, Stride = 2

Decoder: The structure of all the layers is as follows:

- ConvTranspose2d(128, 256, 3) : Input channels = 128, Output channels = 256, Kernel size = 3
- Stride = 2, Padding = 1, Output padding = 1
- ReLU : Activation function
- ConvTranspose2d(64, 32, 3) : Input channels = 64, Output channels = 32
- ConvTranspose2d(32, 3, 3) : layer to reconstruct RGB image in 3 channels
- : Activation function

Total number of parameters of this model: 50 Thousand

3. Variation Auto Encode (VAE):



This Model is composed of an encoder, Latent space, and a decoder. The architecture of the model is as follows:

Encoder:

The encoder part of the network consists of many convolutional layers followed by fully connected layers to generate the mean (μ) and log variance ($\log \sigma^2$) which will be used in the latent space. The layers are as follows.

- Conv2d(3, 8, 4) : In channels = 3, Out channels = 8, Kernel size = 4
- ReLU : Activation function
- Conv2d(8, 16, 4) : In channels = 8, Out channels = 8, Kernel size = 4
- Conv2d(16, 32, 4) : In channels = 3, Out channels = 16, Kernel size = 4
- Conv2d(32, 64, 4) : In channels = 32, Out channels = 64, Kernel size = 4

Latent Space:

The two vectors produced at the end of the encoder μ and $\log \sigma^2$, representing the mean and log variance of the latent space, respectively are used to create latent space representation. These vectors are used to sample from a distribution in the latent space.

- Linear(64×28×16, latent_dim) : Maps the flattened mean features to Latent space
- Linear(64×28×16, latent_dim) : Maps the flattened varaince features to Latent space
- Linear(latent_dim, 64×28×16) : Convert the Latent space features to reconstruct the sharp image

Decoder: The decoder part has to reconstruct the sharp image with correct size. So, it uses the latent space output and then uses transposed convolutions to get image in correct shape.

ConvTranspose2d$(64, 32, 4, 2, 1)$: Input channels $= 64$, Output channels $= 32$, Kernel size $= 4$

ConvTranspose2d$(32, 16, 4, 2, 1)$: Input channels $= 32$, Output channels $= 16$, Kernel size $= 4$

ConvTranspose2d$(16, 8, 4, 2, 1)$: Input channels $= 16$, Output channels $= 8$, Kernel size $= 4$

ConvTranspose2d$(8, 3, 4, 2, 1)$: Final layer to reconstruct RGB image

Sigmoid : Activation function = Sigmoid

Total number of parameters of this model: 8 Million

Loss Functions :

For this project we tried using different loss functions as follows,

MSE loss (mean squared loss) :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

MSE loss is used for base loss function to compare with other loss functions. It can give us the difference between the deblurred image and actual image. But it lacks capturing distributed characteristics of the image. So to avoid this we tried using SSIM

SSIM (structural similarity index measure) :

$$\text{SSIM}(x, y) = \frac{2 \cdot \mu_x \cdot \mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \cdot \frac{2 \cdot \sigma_{xy} + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$$

SSIM loss helps in capturing luminance, constrast and structure of images.

PSNR (peak signal to noise ratio) :

$$\text{PSNR} = -20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{\text{MSE}}}\right)$$

We used negation of PSNR value as loss function. Since the metric used for evaluation is PSNR. By using it as loss function, we got better result with every model we have used so far.

BCE + KLD (specifically for VAE models) :

BCE :

$$\text{BCE}(x, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(x_i) + (1 - y_i)\log(1 - x_i)]$$

KLD :

$$\text{KLD}(P||Q) = \sum_i P(i) \cdot \log\left(\frac{P(i)}{Q(i)}\right)$$

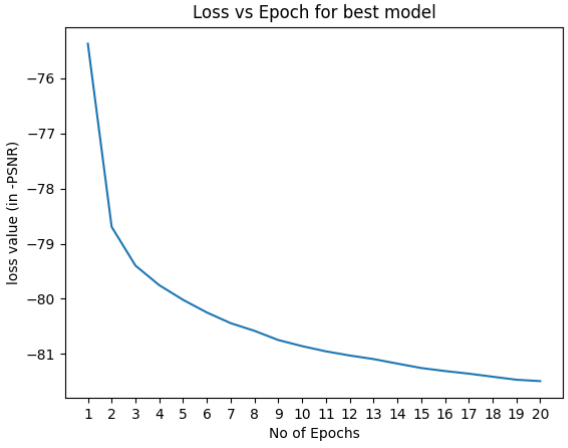
BCE i.e. Binary Cross Entropy loss + KLD Kullback-Leibler Divergence loss were specifically used for VAE model. BCE helps in measuring difference between deblurred image and original image. So it encourages the model to generate outputs similar to original image. It measures the discrepancy between the learned latent distribution and a predefined prior distribution (usually a Gaussian distribution).

Training:

Model	Reduced DebluNet
Loss function	PSNR
Batch size	8
Learning Rate	0.005
Epochs	20
Optimizer	Adam
Batch Normalization	present

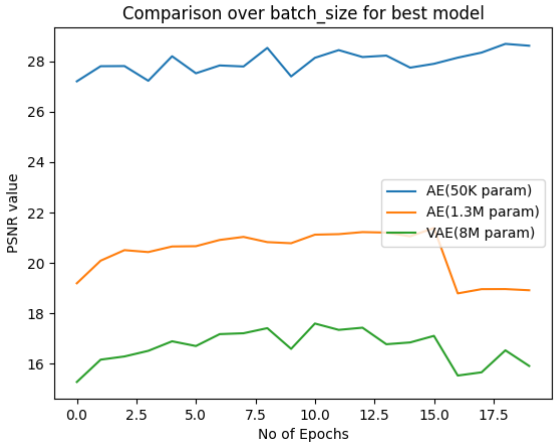
Training Parameters

For the above training parameters we got the best testing PSNR score 28.694143830005814 dB. The number of parameters of this best model are only 50000. Thus, the model is not only accurate but also computationally efficient.
To obtain the optimal values of parameters we used grid search on those attribute. The loss value is negation of PSNR score. the loss value was decreasing continuously for all 20 epochs, which can be seen in the loss vs epoch graph below.
Training Loss graph for the best model



Experiments:

Comparison of Models:
As we can see from the graph, the DeblurNet model is giving better PSNR value during testing than VAE model and reduced DeblurNet model is giving



better PSNR value than DeblurNet . Here the number of parameters of VAE model is about 8 million, on the other hand Deblurnet Model has around 1.3M params, while reduced DeblurNet has around 50K params. The deblurring task has low complexity, so models with high complexity will overfit during training and will be bias towards training data. So one thing we observed is that, decreasing complexity of models benefits us. Also the DeblurNet is more computationally efficient than VAE.

Comparison of Loss Functions:
The following table shows the performance of VAE model for different loss functions:
Here, we have shown data for VAE model, as the last loss function can be used

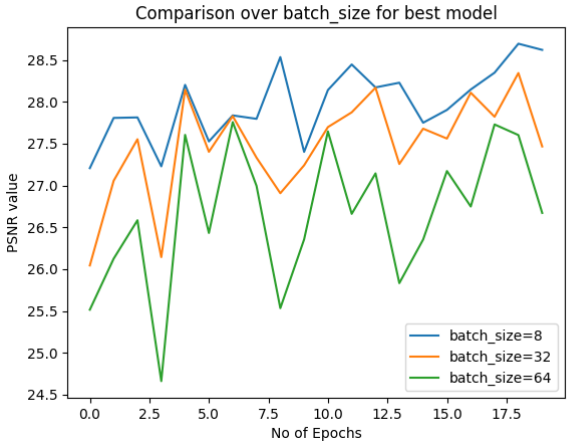
Loss Function	**PSNR score**
MSE	16.27
SSIM	16.89
PSNR	17.12
BCE + KLD	17.64

PSNR score of VAE model for different loss functions

only on VAE model. From table we can see that taking negation of PSNR as loss function benefits more than any other loss function. Since the goal is to reconstruct sharp image from blurred images, PSNR acts as quality measure between output image and original image, while MSE loss only penalises pixel to pixel errors. PSNR performs better than SSIM, beacause SSIM is more complex, also SSIM tries to capture structural similarity instead of quality between two images, PSNR performs better. SSIM performs better than MSE, because it captures more distributed characteristics than MSE. The BCE + KLD loss

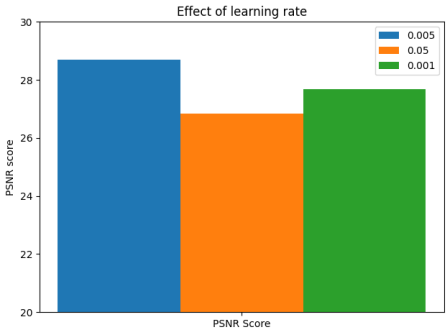
performs the best for VAE model as it is specifically designed to assist that model.

Effect of Batch Size: As we can see from graph of best model, increasing



batch size decreases the PSNR value. So optimum batch size that we calculated is 8.

Effect of learning rate:
We hypertuned the learning rate to increase the performance of the model.



The plot shows the PSNR score of our best model for different learning rate. It is clear that optimal value of learning rate for our model is 0.005.

Effect of Batch Normalization: Adding the batchNormalization layer after each convolutional layer has improved the performance of the model. This is because adding batch Normalization reduces internal covariate shift and add regularization effect.