Deep Learning
Photo Reconstruction Challenge
Spring 2023
Group 14

Group Members:
Anand Manojkumar Parikh - 20CS10007
Anushka Srivastava - 20CS10009
Vivek Jaiswal - 20CS10077
Amit Kumar - 20CS30003

Bi-LSTM score on kaggle: 0.19862
GAN score on kaggle: 0.23275

Contents of DL_Asgn3_Grp14.zip:

1. DL_Asgn3_Grp14_Report	(the final report)
2. DL_Asgn3_Grp14_MidEvalReport	(mid-eval report, for reference)
3. GAN.ipynb	(GAN code)
4. BiLSTM.ipynb	(BiLSTM code + outputs)
5. GAN_output.log	(all outputs of GAN.ipynb)

Link to all models and submission files:
https://drive.google.com/drive/folders/1TX9gJuGIHZAPwF6kpxSyRgqZpvQyckLE?usp=share_link

Description of models in above link:
1. g_model.pth - Generator model (part of adversarial network)
2. d_model.pth - Discriminator model (part of adversarial network)
3. b_model.pth - BiLSTM model

NOTES:
1. The GAN outputs are provided in GAN_output.log
2. The BiLSTM outputs are provided in BiLSTM.ipynb itself

Description of GAN used by us:

- It consists of 2 networks: The generator and the discriminator.
- The generator's job is to produce inpainted images given a masked image.
- The discriminator's job is to accurately classify the original unmasked image as real and the generated image as fake.
- Both these networks are trained together.
- Finally, for testing, we use only the trained generator to create images given a masked image.

Generator Model's description and layers and output dimension at each layer:

```
torch.Size([1, 3, 256, 256])      // Input dimension
torch.Size([1, 64, 128, 128])
torch.Size([1, 64, 64, 64])
torch.Size([1, 128, 32, 32])
torch.Size([1, 128, 16, 16])
torch.Size([1, 256, 8, 8])
torch.Size([1, 512, 4, 4])
torch.Size([1, 4000, 1, 1])       // Context Encoding of Image features
torch.Size([1, 512, 4, 4])
torch.Size([1, 256, 8, 8])
torch.Size([1, 128, 16, 16])
torch.Size([1, 128, 32, 32])
torch.Size([1, 64, 64, 64])
torch.Size([1, 64, 128, 128])
torch.Size([1, 3, 256, 256])
torch.Size([1, 3, 256, 256])      // Output dimension
```

- The model takes in a batch of masked/damaged images, each of size 3 * 256 * 256.
- It is downsampled gradually and at each layer some features of the input are extracted.
- The context vector obtained after downsampling is of length 4000, which contains the features.
- Then, it is again upsampled gradually, and the full image is reconstructed.
- Finally, an output of size 3 * 256 * 256 is generated, which is full image (without masks)

Error function for Generator:

- We want the generator to make realistic fake images, so the error is defined as -
  0.999 * L1(g_out , original) + 0.001 * MSE(discriminator(g_out) , real)

  g_out = generated output image by generator
  original = original unmasked image
  discriminator(g_out) = classification of generated image by discriminator
  real = labels for real image (all 1's)

- This loss is basically the distance: "how far is generated output from real image" + "how far does discriminator classify the generated output from real image", with some appropriate weights given to both components.
- Minimizing this forces the generator to create better fake images.

Discriminator Model's description and layers and output dimension at each layer:

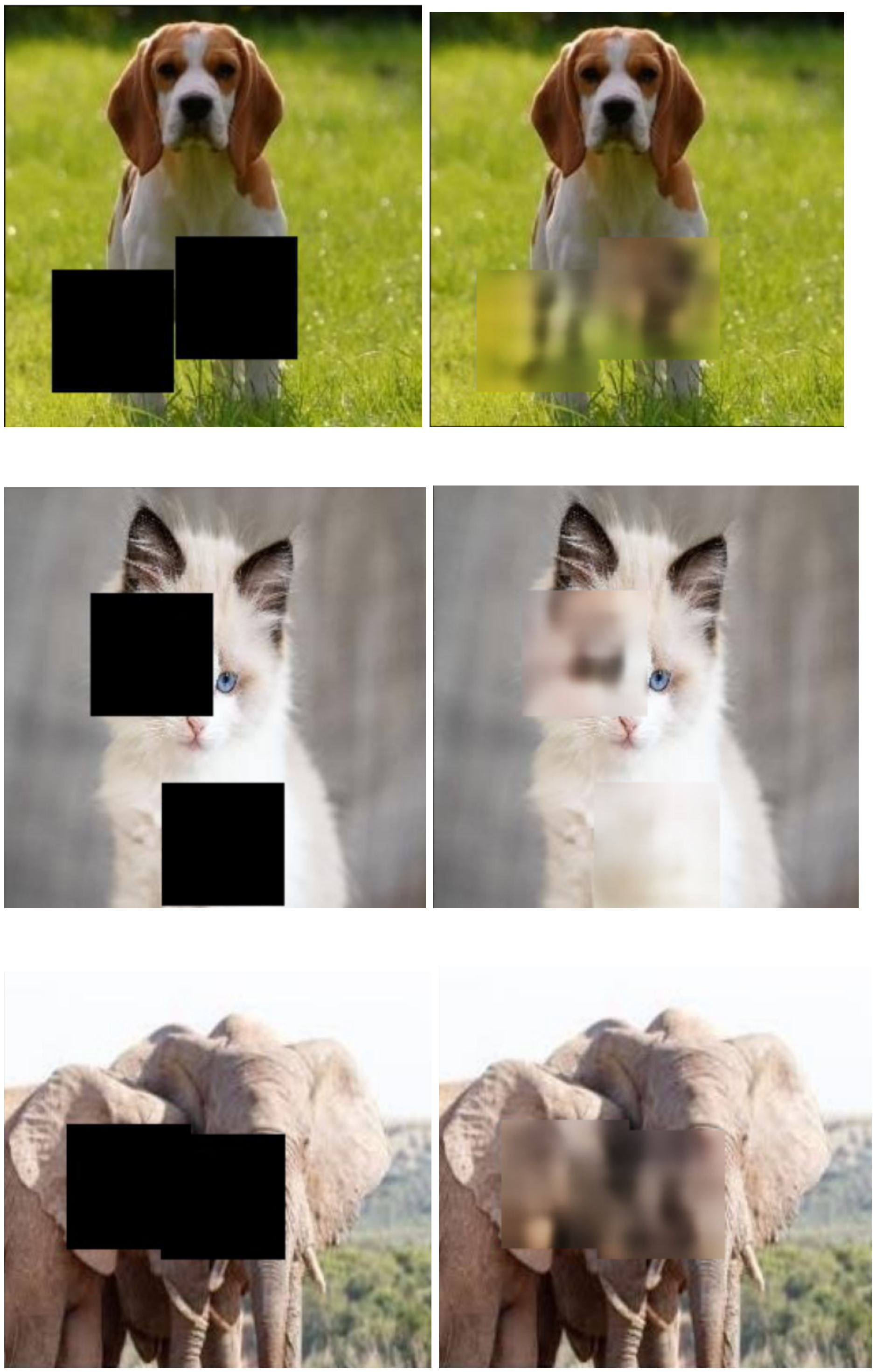
```
torch.Size([1, 3, 256, 256])       // Input Dimension
torch.Size([1, 64, 128, 128])
torch.Size([1, 128, 64, 64])
torch.Size([1, 256, 32, 32])
torch.Size([1, 512, 16, 16])
torch.Size([1, 1, 16, 16])
torch.Size([1, 1, 16, 16])         // Output Dimension
```

- In each epoch, the model is fed 2 images of size 3 * 256 * 256 :
  the generated (fake) image and the real image.
- It outputs an array of size 16 * 16 as the classification.

Error function for Discriminator:

- We want the discriminator to distinguish successfully between real and fake images, so error is:
  0.5*MSE(discriminator(g_out),fake) + 0.5*MSE(discriminator(original),real)
- This loss is basically the distance: "how far does the discriminator classify the fake image from fake labels" + "how far does the discriminator classify the real image from real labels"
- Minimizing this forces the discriminator to learn the difference between real and fake images.

Sample GAN Reconstructions:



GAN Hyper - Parameters used while training:

- NUM_EPOCHS = 80
- BATCH_SIZE = 16

Some extra details:

1. No. of training samples = 7000
2. No. of testing samples = 200
3. No. of training batches = 7000 / 16 = 438
4. Training Time = 4 hours

Possible Improvements:

1. Using random masking rather than using static masks provided in the training samples repeatedly.
2. Using image modifications like blurring, rotation, cropping, etc.
3. Using dropout technique to train faster.

GAN Sources:

1. https://medium.com/@renithprem/how-to-repair-your-damaged-images-with-deep-learning-cc404aec144
2. https://medium.com/ai-society/gans-from-scratch-1-a-deep-introduction-with-code-in-pytorch-and-tensorflow-cb03cdcdba0f

Description of BiLSTM used by us:

1. It consist of two layers:
    a. BiLSTM layer with input_size=256, hidden_size=128 and num_layers=2
    b. Linear Layer with (hidden_size*2, input_size) as parameters
2. We are sending whole image and train according to whole image with input size = (3,256,256) and its output is send to Linear layer as parameters and output is finally input size, i.e, (3,256,256)

Error function for BiLSTM:

1. Error Function is the class named ReconstructionLoss with Loss as MSELoss of complete image and its output
2. We are calculating MSELoss over all the pixels because the masked region can be anywhere so train all over the pixels.

BiLSTM Hyper - Parameters used while training:

- NUM_EPOCHS = 10
- BATCH_SIZE = 1
- input_size=256
- hidden_size=128
- num_layers=2
- Learning rate = 0.001

Sample BiLSTM Reconstructions: