CS F437: Generative Artificial Intelligence

Assignment 2

Submission Time and Date: 2350hrs, 30th April 2024

Instructions:

- This assignment is a coding project and is expected to be done in groups. Each group can contain at most three members. Make sure that all members of the group are registered for this course.
- This assignment is expected to be done in Python using standard libraries like NumPy, Pandas, and Matplotlib unless otherwise specified. Jupyter Notebook/Google Colab can be used. Refrain from directly copying codes/snippets from other groups or the internet, as all codes will be checked for plagiarism.
- The deliverable items include the code files (.py or .ipynb) and a report with the documented results. All deliverable items should be put together in a single .zip file. Rename this file as follows before submission: A1_<id-of-first-member>_<id-of-second-member>_<id-of-third-member>
- Submit the zip file on CMS on or before the aforementioned deadline. Please note that this is a hard deadline and no extensions/exemptions will be given. The demos for this assignment will be held later which shall be conveyed to you. All group members are expected to be present during the demo.

Part-A: Image Generation

The objective of this task is to train generative models to generate images given random vectors sampled from the latent vector space. The following three algorithms are to be used:

- Variational Autoencoders
- Generative Adversarial Networks
- Diffusion Models

Dataset

The dataset for this assignment is the 'MNIST' dataset, which consists of images of handwritten digits from 0 to 9. The dataset consists of 60,000 training images and 10,000 testing images.

- This dataset is available on the internet, or you can also use any library that has the MNIST dataset to load it.
- Do not load the labels associated with the images. We will only be working with the images and not their labels.

Part A-I: Variational Autoencoder

The objective is to build and train a Variational Autoencoder model for the Image Reconstruction task. You will be required to build the models with the following dimensions of latent variables: 2, 4, 8, 16, 32, 64, thus giving 6 VAE models in total.

The trained decoder will be used for image generation.

What needs to be documented?

Randomly sample 5 points from the latent vector space and generate output images for them. Put these images in the report. This should be done for all 6 VAE models.

Also, note down the mean and the variance of the latent vector space for dimensions: 2, 4, 8, and 16.

Part A-II: Generative Adversarial Network

The objective is to build and train a Generative Adversarial Network (GAN) model for the Image Generation task. You will be required to build the models with the following dimensions of latent variables: 2, 4, 8, 16, 32, 64, thus giving 6 GAN models in total.

What needs to be documented?

Randomly sample 5 points from the latent vector space and generate output images for them. Put these images in the report. This should be done for all 6 GAN models.

Part A-III: Diffusion Model

The objective is to build and train a Diffusion model for the Image Generation task. You will be required to build the models with the following dimensions of latent variables: 8, 16, 32, 64, thus giving 4 Diffusion models in total.

What needs to be documented?

Randomly sample 5 points from the latent vector space and generate output images for them. Put these images in the report. This should be done for all 4 Diffusion models.

Implementation Requirements for Parts A-I, II, and III:

1. The following libraries are permitted: Scikitlearn, Tensorflow, Keras, and PyTorch
   -> These libraries should only be used to define the network layers and train the models. No pre-existing implementation of VAE, GAN, or Diffusion model should be used.
2. The architecture of any constituent neural network is up to the student. Both Fully-connected and CNN layers can be used.
   -> A minimum of 4 hidden layers must be present in each network.
3. The activation function is from any of the following functions as per requirement: tanh, sigmoid, ReLU, LeakyReLU.
4. The optimization algorithm is from any of the following: Stochastic Gradient Descent, Adam optimizer.

Part-B: Image-to-Image Translation

The image-to-image translation task involves translating or converting a source image into a target image while modifying certain visual properties of the image as per the requirement and preserving the rest of the properties. The objective of this task is to build image-to-image translation models using the following two variants of the Generative adversarial networks (GANs) and analyze the differences in their translational abilities.

- Deep Convolutional GAN (DCGAN)
- CycleGAN

Dataset

The dataset for this task is the 'CelebA' dataset, which consists of more than 200K celebrity images, each with 40 binary attribute annotations. The images and annotations of the dataset are present in this gdrive link.

- The '`Align&Cropped Images`' are to be used for this task.
- The attributes corresponding to these images are present in the '`list_attr_celeba.txt`' file.

- The train, validation, and test splits are mentioned in the '`list_eval_partition.txt`' file.

The details of the image source, the attributes file, and the train-validation-test split file are provided in the README file of the gdrive folder. Please refer to that.

Part B-I: Deep Convolutional GAN (DCGAN)

DCGAN consists of a CNN-based generator and a discriminator that satisfies certain properties to ensure stable training. The details of the model architecture and the guidelines that the generator and discriminator CNNs need to satisfy are given in the DCGAN Paper. Please read the paper thoroughly before starting this part of the assignment.

- The generator and discriminator are expected to have at least five Convolutional layers satisfying the architecture guidelines for stable Deep Convolutional GANs.

NOTE: The fractional-strided convolutional layer refers to the transposed convolutional layer or deconvolutional layer.

DCGAN Training:

Train the model using the standard GAN training objective. The hyperparameters mentioned in the DCGAN paper should be used for training. Train the model on the 'train split' of the images.

Encoder Architecture and Training:

The DCGAN generator takes the 100-dimension 'z vector' as input to generate the images. The next step in this pipeline is to train a CNN encoder on top of the DCGAN model to generate the 'z vectors' from the input images.

- Keeping the DCGAN generator fixed, the encoder should be trained using the 'Image Reconstruction' objective with a Mean Squared Error loss between the input image to the encoder and the output image from the generator.
- Adam Optimizer should be used for training. Try out a few values for the hyperparameters of the optimizer (learning rate, beta1) and select the set of hyperparameters that maximize the performance on the validation split of the data.

- The Encoder is expected to have at least six Convolution layers. Except for the last convolutional layer, all other layers must be augmented with Leaky ReLU activation and batch normalization layers.
- The following libraries are permitted: Scikitlearn, Tensorflow, Keras, and PyTorch -> These libraries should only be used to define the network layers and train the model. No pre-existing DCGAN implementation should be used.

Vector Arithmetics for Image-to-image translation:

The vector arithmetics mentioned below should be used to achieve image-to-image translation. As mentioned in the DCGAN paper, three images from each category must be randomly sampled and averaged for the vector arithmetics. The test split images should be used for this purpose.

Following are the vector arithmetics to be performed (over the 'z vectors'):

1. Men without glasses + People with glasses - People without glasses
2. Men with glasses - Men without glasses + Women without glasses
3. Smiling Men + People with Hat - People with Hat + People with Mustache - People without Mustache

For each of the above vector arithmetics, generate 5 outputs keeping the following points in mind:

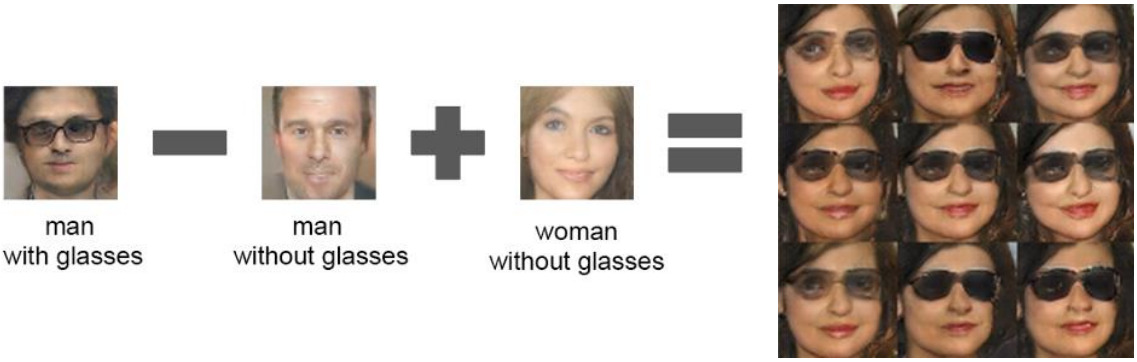
- Each output must be a 3x3 grid output of images (as explained in the vector arithmetics section of the DCGAN paper).

For each of the above vector arithmetics, again sample three images at random for each category and generate 1 translated output. But this time, don't perform the vector arithmetics in the 'z space'. Instead, perform it in the input image space itself before passing the image to the encoder.

What needs to be documented in the report?

1. For each of the vector arithmetics:

- For all 5 outputs, show the average of the three images in each category and the 3x3 output as follows:



Hence, a total of 15 outputs are excepted (3 vector arithmetic operations * 5 outputs each)

- Next, the output for the input image-level arithmetics for all three cases should be in a format similar to the example above (3x3 grid output isn't required; only the single output image is fine).

Part B-II: CycleGAN

Implement the CycleGAN model as taught in class for the image-to-image translation task. More details about the generator and discriminator architectures, as well as the objective functions, can be found in the CycleGAN Paper.

- Follow the generator architecture as explained in Section 4 of the CycleGAN paper.
  -> As mentioned in the paper, 'Instance Normalization' should be used for all Convolutional layers. **'PyTorch'** API can be used for this purpose.
  -> ReLU / LeakyReLU should be used as layer activations.
  -> Adam Optimizer must be used
- For the discriminator, 4 convolutional layers (along with Instance Normalization and LeakyReLU) and a final convolutional layer to get the desired output must be used.
- Try out a few values for the hyperparameters of the optimizer (learning rate, beta1) and select the set of hyperparameters that maximize the performance on the validation split of the data.
- Train the model using the objective functions taught in class and explained in the CycleGAN paper.

- The following libraries are permitted: Scikitlearn, Tensorflow, Keras, and PyTorch
  -> These libraries should only be used to define the network layers and train the model. No pre-existing CycleGAN implementation should be used.

Image-to-image Translation:

- Train the model on the following two datasets (from the training split of CelebA):
  1. Men without glasses and men with glasses
  2. Men with glasses and women with glasses
- Test the model for the following:
  -> Men without glasses to men with glasses (5 images) and vice versa (5 images)
  -> Men with glasses to women with glasses (5 images) and vice versa (5 images)

What needs to be documented in the report?

- The model outputs for the testing data as mentioned above.