Neural Networks and Deep Learning (Spring 2021)

Homework #4: Deep ConvNet and GAN

Due date: 22nd Mordad 1400

In order to do this homework, go through theories and concepts from Deep Learning, Convolutional Neural Networks (ConvNets), and Generative Adversarial Networks (GAN). If you're coding in the Python or Matlab, you can use any library for GAN. People who like to implement GAN without using any library are viewed positively.

Problem Definition

GAN consists of two primary networks: Generator, $G(z)$, and Discriminator, $D(x)$. The generator module is used to create artificial samples of data by incorporating feedback from the discriminator. Its objective is to deceive the discriminator into classifying the generated data as belonging to the original dataset and ultimately minimize the cost function:

$$\min_G \max_D E_{x\sim P_{\text{data}}(x)} \log D(x) + E_{z\sim P(z)} \log(1 - D(G(z)))$$

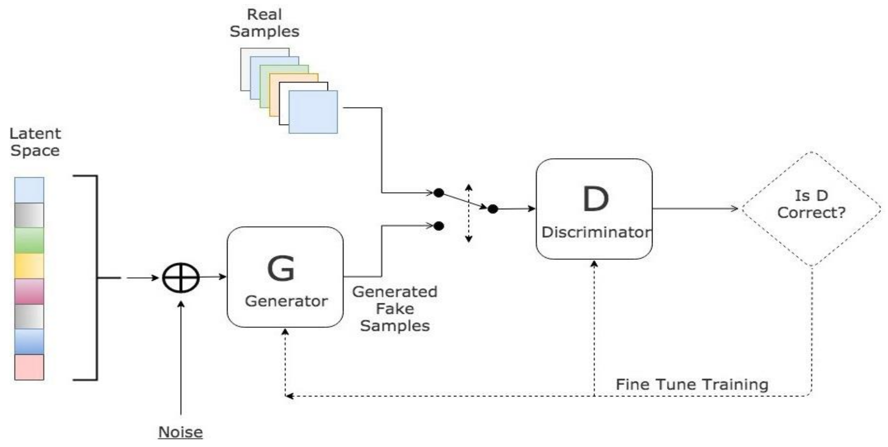

Figure 1. GAN architecture.

During training, as you can see in Figure 1, the generator encapsulates the probability distribution of original data and is trained to generate data that maximizes the probability of the discriminator mistaking the fake data to be real. The end goal of the generator is such that the discriminator is no longer able to differentiate between real data or synthetically generated data.

From another perspective, the generator G takes as input a random noise vector z and outputs an image $x_{\text{fake}} = G(z)$. The discriminator D receives as input either a training image or a synthesized image from the generator and outputs a probability distribution $P(s|x) = D(x)$ over possible image sources. The discriminator is trained to maximize the log-likelihood it assigns to the correct source:

$$L = E[\log P(s = \text{real} \mid x_{\text{real}})] + E[\log P(s = \text{fake} \mid x_{\text{fake}})]$$

The Auxiliary Classifier GAN (AC-GAN) is simply an extension of GAN that requires the discriminator to not only predict if the image is ‘real’ or ‘fake’, but also provide the ‘source’ or the ‘class label’ of the given image.

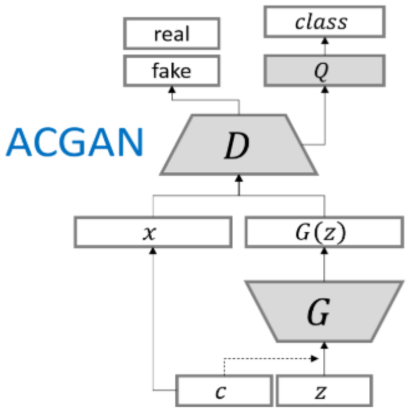

Fig.2- AC-GAN architecture

In AC-GAN, each generated sample has a corresponding class label, $c \sim P(c)$, in addition to the noise z. Generator G tries to generate $x_{\text{fake}} = G(c, z)$. The discriminator D gives probability distributions over both sources and class labels, $P(s|x), P(c|x) = D(x)$. The objective function has two parts: the loglikelihood of the correct source, L_s, and the log-likelihood of the correct class, L_c.

$$L_s = E[\log P(s = \text{real} \mid x_{\text{real}})] + E[\log P(s = \text{fake} \mid x_{\text{fake}})]$$

$$L_c = E[\log P(c = \text{real} \mid x_{\text{real}})] + E[\log P(c = \text{fake} \mid x_{\text{fake}})]$$

D is trained to maximize $L_c + L_s$ while G is trained to maximize $L_c - L_s$. For more information, refer to AC-GAN paper.

Your Task

Dataset: ***COVID, Non-COVID*** CT IMAGES

https://drive.google.com/drive/folders/1kVIe0HIYz_k9Jcjn27ViHPe51AG9y_fr?usp=sharings

Part A: In this section, you will learn a deep ConvNet to classify COVID and Non-COVID images.

1. Divide the data into 80% for training and 20% for test of model. Note that there is more than one image for a patient and the images of a particular patient should not be in both training and test data.
2. Do the necessary pre-processing on the images and resize them suitably for entering the network. The images in the database have different sizes.
3. Train a deep ConvNet as a classifier to distinguish between COVID and Non-COVID CT images.
4. You can use pre-trained networks (e.g., VGG-16) to extract features, or you can train a new network.
5. Calculate the precision, recall, F1-score, accuracy and AUC criteria and report them. In python, you can use SKlearn library for these metrics.
6. Plot the ROC curve.

Part B: The purpose of this part is to generate more images than the given dataset. You will train an AC-GAN to generate images while predicting their classes.

1. Do steps 1-2 of part A.
2. Train an AC-GAN to can generate COVID and Non-COVID images with labels. You can use any deep structure for its generator and discriminator.
3. Use the ConvNet, trained in part A, and test it with the generated (labeled) images in this part. Report the same metrics, calculated for them.
4. Give 20% of the original test images along with the generated images as test images to the trained ConvNet and report the same metrics.
5. Plot the ROC curve.

Notes:

- **Pay extra attention to the due date. It will not extend.**
- **Be advised that submissions after the deadline would not grade.**
- **Prepare your full report in PDF format and include the figures and results.**
- **Please do not send us the database when you send your homework.**
- **You can use any library for ConvNet and GAN in Matlab, Python or other languages.**
- **Submit your assignment using a zipped file with the name of "StdNum_FirstName_LastName.zip" to compuscien@gmail.com with "NNDL-Spring 2021-HW#4" subject.**