AML assignment -2

Aishvarya S – MDS202302

Part – 1

Trained and evaluated RNN and LSTM models for SMS spam classification using PyTorch.

Starts by loading a labeled SMS dataset, encoding labels (spam = 1, ham = 0), and splitting data into train and test sets. Messages are tokenized, and a vocabulary is built for converting text into sequences. A custom PyTorch dataset class is defined for loading data, with padding and collating for variable-length sequences.

Two models, RNN and LSTM, are built with an embedding layer, recurrent layer (RNN or LSTM), and a fully connected layer for binary classification. The models are trained using cross-entropy loss and Adam optimizer for 20 epochs, reaching high accuracy (RNN: 97.40%, LSTM: 97.49%). Evaluation metrics, including precision, recall, and F1-scores, are reported for each model

Part – 2

Implemented and trained both RNN and LSTM models on a text prediction task using an SMS spam dataset. The dataset is tokenized, sequences are created by splitting messages in half, and models are trained to predict the second half based on the first. The vocabulary is tokenized, sequences are padded, and train/test splits are made.

Each model architecture consists of an embedding layer, an RNN or LSTM layer, and a linear output layer. Training runs for 100 epochs with CrossEntropy loss, where RNN and LSTM achieve losses of 0.8147 and 0.7688, respectively, on the test set. Prediction tests on unseen halves of SMS messages show the LSTM slightly outperforms the RNN, especially in longer, complex messages, producing more coherent sentence completions.

- **RNN Loss:** 0.8147, **Accuracy:** 92.04%
- **LSTM Loss:** 0.7688, **Accuracy:** 92.08%

Both models demonstrate strong performance on this text continuation task, with the LSTM model slightly better at capturing dependencies in longer sequences.

Part – 3

Implemented a conditional GAN (Generative Adversarial Network) for generating Fashion-MNIST images.

Dataset class loads Fashion-MNIST data from IDX files, handling the reading of images and labels. The transform argument allows preprocessing such as normalization.

The Generator class generates new images. It combines random noise (latent_dim) and label embeddings, then applies a series of upsampling and convolutional layers to produce realistic images.

The Discriminator class distinguishes real from fake images, embedding labels alongside the image data before passing them through linear layers with LeakyReLU activations and a sigmoid output

The training loop alternates between updating the generator and discriminator. The generator is trained to create realistic images that can "fool" the discriminator, while the discriminator is trained to distinguish real images from generated ones.Binary cross-entropy loss (BCELoss) is used to train both networks, with separate optimizers for the generator and discriminator.

The training process shows the evolving balance between generator and discriminator performance