Aniket Tiwari MDS202308
Bibek Paul MDS202317
Himanshu MDS202327

Assignment 02 Report

Task 1

- Renamed the column names to `label` and `message` and mapped `{'ham': 0, 'spam': 1}`.
- Split the data into train and test sets with `message` acting as input and `label` acting as output and then **tokenized** the input.
- Found out the maximum sequence length for **padding**.
- Trained on the padded sequences using RNN and LSTM both models reached high training accuracy, but **LSTM showed a slight advantage** in validation accuracy across epochs.
- The RNN model reached **~98% accuracy** and **~100% precision**, while the LSTM model achieved **~99% accuracy** and **~91% precision** on the test dataset.
- We give more importance to **precision** because it's important to reduce false positives as per our model.

Task 2

- Developed RNN and LSTM models to predict the second half of an SMS given the first half, with an `<END>` token to signal the end of the output.
- Text cleaning included tokenization and padding.
- Padding length was chosen 40 as most of the sentences have very small sequence length and RNN and LSTM models will face vanishing gradient problem.
- We pre padded the first half with `max_len=20` and post padded the second half with `max_len=20`.
- **RNN Model**: Consisted of Embedding, Simple RNN, Time Distributed and Dense layers.
- **LSTM Model**: Consisted of Embedding, LSTM, Time Distributed and Dense layers, using similar parameters (128, 128, 64) to maintain comparability.
- Both models were trained with sparse categorical cross-entropy loss and evaluated on a test set.
- In the examples, the LSTM model displayed slightly better convergence, producing more coherent message continuations than the RNN model.
- **LSTM** was found to be more effective than RNN for sequential text generation, as it better preserved context over longer sequences with a loss of 3.45 compared to loss of 3.95 in RNN.
- Although both the models didn't provide satisfactory results.

Task 3

- Loaded the Fashion MNIST dataset, which consists of 70,000 grayscale images of clothing items, and split it into training and test sets for model training and evaluation.

- Normalized the image pixel values to a range between -1 and 1 for better convergence during training.
- Labels were one-hot encoded to facilitate class-specific image generation.
- **Model Architecture:**
    - **Generator:** Takes a random noise vector and class label to generate an image matching the given label, using layers like dense, reshape, and transposed convolution.
    - **Discriminator**:Receives an image and class label, determining if the image is real or fake through convolutional and dense layers.
    - This architecture is chosen because it enables the generation of realistic, class-specific images by conditioning the output on class labels, allowing for controlled and varied image synthesis in the GAN framework.
- **Training Loop:** Alternates training of the generator and discriminator to balance generation quality with classification accuracy. In a GAN, the discriminator's accuracy should ideally converge around 50% as it becomes equally challenged by real and generated images, indicating a balance. At the start, discriminator accuracy is often high (close to 100%) as it easily identifies the initial poor-quality fakes from the generator. As training progresses, generator quality improves, and the discriminator accuracy should drop to around 50%.
- **Loss Functions:** Uses binary cross-entropy loss for both models because through it encourages the generator to produce realistic images that fool the discriminator, while the discriminator focuses on distinguishing real from fake images accurately. It also effectively measures the probability error for distinguishing between real and fake images, which is central to GAN training.
- **Image Generation:** After training, the generator creates images by combining random noise with specific class labels, allowing it to produce various images across clothing categories.
- **Results:** Displayed 100 random generated images after 50 and 100 epochs. Also plotted 4 generated images per class, demonstrating the model's ability to create unique clothing items per label.