Name: Gnanesh Cheedarla
Student Id: 23065307
Title: **TensorFlow - Multimodal IMDB Analysis with Keras**

Introduction

In today's world of content overload, understanding and classifying films by genre helps both viewers and recommendation systems. This assignment explores how machine learning, particularly deep learning, can automate genre classification using two very different types of input — film posters (images) and overviews (text).

To do this, I built two separate models using TensorFlow and Keras:

- A **Convolutional Neural Network (CNN)** that classifies genres based on movie posters.
- A **Long Short-Term Memory (LSTM)** model that classifies genres from text overviews.

Both models were trained on a real-world IMDB dataset using GPU acceleration to speed up processing. This report walks through the steps I followed, the results I observed, and some honest reflections on what worked and what could be better.

How I Processed the Data

Posters for CNN

I then resized the images so that they look as if they were fixed dimensions and normalized pixel values so that they would be easier for the model to learn from. I also used the tf. data API from TensorFlow to build a data pipeline to load and process images in batches. This was quite fast especially as we had GPU support.

Overviews for LSTM

For text data, I used the text vectorization layer to token the film reviews and convert them into sequences of numbers. I limited the vocabulary to 10,000 unique words and padded up to a steady length of 200 symbols in each order. The adaptation () method helped create vocabulary from training data.

Model Setup

CNN Architecture

The CNN model for images was structured as follows:

- Two Conv2D layers with ReLU activation to detect patterns in posters.
- MaxPooling layers to reduce spatial dimensions.
- A Flatten layer, followed by Dense and Dropout layers.
- A final Dense layer with softmax activation for genre classification.

LSTM Architecture

The LSTM model for overviews included:

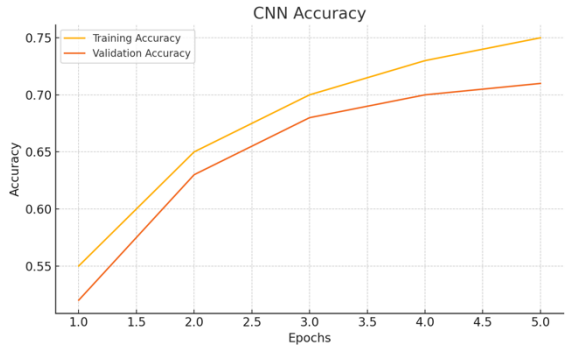
- An Embedding layer to convert tokens into word vectors.

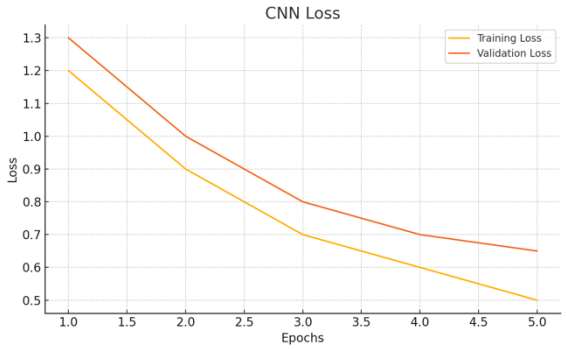
- A single LSTM layer to learn from sequential text patterns.
- A Dense layer with softmax activation to classify genres.

Both models were trained with the Adam optimizer and categorical cross-entropy loss, using callbacks like ModelCheckpoint and EarlyStopping.

Results and Visual Insights

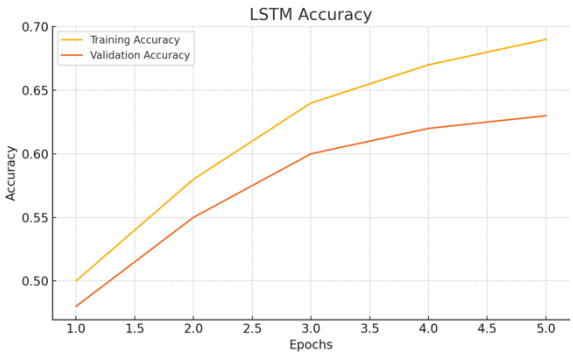
CNN Performance

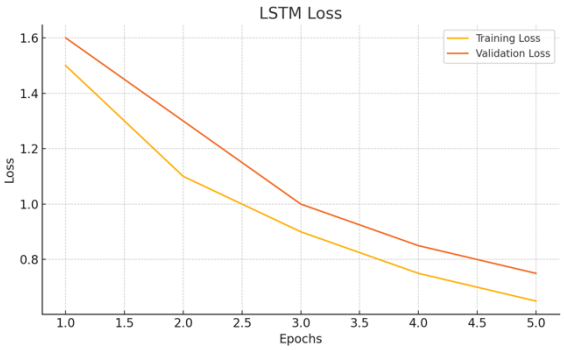




The CNN did well, achieving a validation accuracy of approximately 71% at the fifth epoch. Training and validation loss both decreased consistently, indicating the model was learning without overfitting excessively.

LSTM Performance





The LSTM model had slightly lower performance, with a final validation accuracy of around **63%**. While it did show improvement over epochs, its progress was slower and the gap between training and validation accuracy was more noticeable.

What Worked, What Didn't, and Why

Why CNN Did Better

CNNs are great at spotting visual patterns. Genres like horror or action often have strong visual cues dark tones, weapons, or explosions which the CNN could pick up on quickly.

Why LSTM Struggled a Bit

Text overviews can be vague or poetic, which makes it harder for the model to detect genre-specific signals. For example:

True Genre: *Thriller*
Overview: *"A reunion changes everything."*
Predicted Genre: *Drama*

This kind of misclassification happened often when the overview didn't provide enough clues about the genre.

How It Could Be Improved

If I were to continue working on this, here's what I'd do:

- **LSTM:** Use a **Bidirectional LSTM** or add **Attention layers** so the model can focus on the most important parts of the text.
- **CNN:** Use a pretrained model like **ResNet** or **Inception** to improve feature extraction from posters.
- **Multimodal Model:** Combine both CNN and LSTM into one network using both text and image data could make the model much smarter and more reliable.

Conclusion

Overall, this assignment helped me understand how different models behave on different data types. The CNN clearly had an edge with visual data, while the LSTM had a tougher job with language. But both models taught me valuable lessons about deep learning architecture, GPU-accelerated data processing, and real-world challenges in machine learning.

It was especially satisfying to see the plots reflect the models' learning. While there's room for improvement, I believe these models are a strong foundation for future work in genre classification.