Deep Learning (COSC 2779) – Assignment 2 – 2024

Arvindh Bharadwaj Venkatesan(s3967377)

1 Problem Definition and Background

The goal of this assignment is to develop a **multi-modal deep learning system** designed to classify multiple **propaganda techniques** within both text and image data. By focusing on meme content derived from the **SemEval-2021 Task 6**, this work aims to enhance **media literacy** and improve critical thinking skills among users. Propaganda techniques are categorized into **22 labels**, including "Appeal to authority," "Flag-waving," and "Loaded Language," necessitating a sophisticated approach to multi-label classification.

To achieve this, the system integrates **DistilBERT** for textual feature extraction and **ResNet50** for image processing. The dataset comprises **2,000 memes**, each potentially annotated with multiple labels, presenting challenges related to data variability and subjective interpretations of persuasion techniques. Detecting these techniques is vital for understanding and critiquing media narratives, making this project relevant in today's information-rich environment.

2 Evaluation Framework

The model's performance is evaluated using several key metrics:

- **Accuracy**: Measures the percentage of correct predictions.
- **Precision**: Indicates the proportion of true positive predictions relative to the total positive predictions.
- **Recall**: Reflects the ability of the model to identify all relevant instances (true positives).
- **F1 Score**: Combines precision and recall into a single metric for model evaluation.
- **Area Under the Curve (AUC)**: Assesses the model's ability to discriminate between classes.

The final evaluation yielded the following scores on the test dataset:

- **Loss:** ~0.02
- **Precision**: 0.17 (average)
- **Recall**: 1 (average)
- **F1 Score**: 0.24 (average)
- **AUC Score**: 0.5178

3 Approach & Justifications

Data Preprocessing

Data preprocessing is critical for optimizing model performance:

- **Text Data**: Tokenization is performed using the **DistilBERT tokenizer**, which transforms raw text into input IDs suitable for model training. Texts are padded to a maximum length of 128 tokens.
- **Image Data** Images are resized and normalized to standardize input dimensions for the **ResNet50** model, enhancing the model's ability to learn spatial features and improving convergence during training..

Exploratory Data Analysis (EDA)

EDA was conducted to inform model development:

- **Textual Insights**: Analyzed word count distributions, identifying a mean length of 25 words per text with a maximum length of 128 words.
- **Image Characteristics**: Displayed sample images and evaluated color intensity distributions across the RGB channels, identifying any quality issues.
- **Label Distribution**: A significant class imbalance was noted, with certain labels (e.g., "Loaded Language") appearing more frequently than others, necessitating class-weighting adjustments during training.

Model Architecture

The model employs a **multi-input architecture**:

- **Text Path**: Processes text through **DistilBERT** followed by a Dense layer for classification, leveraging its contextual embeddings for effective language understanding.
- **Image Path**: Employed **ResNet50** for feature extraction, capturing complex visual patterns, which are then processed through additional Dense layers for prediction. The outputs from both paths are concatenated and processed through additional layers before classification.

This architecture allows the model to learn effectively from both textual and visual modalities, enhancing its ability to identify propaganda techniques

4 Experiments & Tuning

Hyperparameter Tuning

The model's performance was optimized through careful tuning of key hyperparameters:

- **Learning Rate**: Adjusted within the range of **1e-5** to **1e-3** to balance convergence speed and stability.
- **Batch Size**: Tested values of **16**, **32**, and **64** to find the optimal trade-off between training efficiency and memory usage.
- **Epochs**: Trained for a maximum of **25 epochs**, implementing early stopping based on validation loss to prevent overfitting.
- **Dense Units**: Various configurations (128 to 512) of Dense layers were tested, with the final model utilizing **128 units** in the classification layers to capture complex relationships in the data.
- **L2 Regularization**: Applied L2 regularization with values from **1e-6** to **1e-2** to the Dense layers to prevent overfitting, ensuring that the model generalizes better to unseen data and the best L2 we got from random search was **0.00011824014373138818 (L2)**

Training Process

The training results were monitored through key metrics:

- **Training Loss**: Decreased from **0.6** to **0.02**, indicating effective learning.
- **Validation Accuracy**: Achieved **0.1256**, reflecting the model's classification capability.
- **Validation Token AUC**: Reached a score of **0.5**, demonstrating moderate class discrimination ability during testing.

Insights and Adjustments

1. **Class Imbalance**: Observed variability in precision and recall emphasized the need for strategies like class weighting to enhance model performance on minority classes.
2. **Model Generalization**: While the model effectively learned from training data, further tuning and additional data are necessary to improve generalization across all classes.

The tuning experiments revealed critical insights into the model's behavior, informing future adjustments aimed at enhancing its robustness and accuracy in classifying propaganda techniques.

5 Ultimate Judgment, Analysis & Limitations

The model exhibits promising capabilities in classifying propaganda techniques, as evidenced by the following outcomes:

1. **Performance Metrics**:
    - **Validation Accuracy**: **0.1256**
    - **Validation Token AUC**: **0.3635**
    - **Training Loss**: Decreased from **0.6** to **0.2**, indicating effective learning and adaptation.
2. **Strengths**:
    - **Multi-modal Integration**: The model effectively processes both text and image data, utilizing **DistilBERT** for textual analysis and **ResNet50** for image feature extraction, capturing complex patterns in propaganda techniques.
    - **Improved Learning**: Notable reduction in training loss demonstrates the model's ability to learn effectively from the training dataset.

3. **Areas for Improvement**:
    - **Model Generalization**: Despite decent performance, the model struggles with generalization across less frequent classes, necessitating enhancements in training strategies.

4. Limitations

a. **Class Imbalance**:
    - The dataset exhibits significant class imbalance, with certain techniques being underrepresented. This can lead to skewed results and affect model predictions, making techniques like **oversampling** or **undersampling** essential.

b. **Subjectivity in Labeling**:
    - Inconsistencies in labeling can result in varying interpretations of techniques, leading to challenges in the training dataset quality. This inconsistency affects precision and recall across different classes.

c. **Computational Resource Requirements**:
    - The model's training demands substantial computational power, particularly for processing high-resolution images through **ResNet50**. This requirement can limit the ability to conduct extensive training or experiments, impacting the model's scalability.

d. **Evaluation Metrics Limitations**:
    - The model achieved moderate scores, with a validation AUC of **0.3635** indicating room for improvement in discriminating between classes, especially for minority labels.

5. Conclusion

In summary, while the model lays a strong foundation for identifying propaganda techniques, addressing unique challenges such as class imbalance, improving labeling consistency, and optimizing resource usage are vital for enhancing its overall effectiveness and applicability.

6 References:

- Bose, A. (2024). *Enhanced Detection of Hateful Memes Using Multimodal Learning*. [online] Medium. Available at: https://abhishekbose550.medium.com/enhanced-detection-of-hateful-memes-using-multimodal-learning-70c3a1dab74a [Accessed 17 Oct. 2024].
- MIND-Lab (2021). *GitHub - MIND-Lab/MEME: MEME benchmark dataset for misogyny detection*. [online] GitHub. Available at: https://github.com/MIND-Lab/MEME [Accessed 17 Oct. 2024].
- GianRomani (2022). *Detection-of-Persuasion-Techniques-in-Texts-and-Images/README.md at main · GianRomani/Detection-of-Persuasion-Techniques-in-Texts-and-Images*. [online] GitHub. Available at: https://github.com/GianRomani/Detection-of-Persuasion-Techniques-in-Texts-and-Images/blob/main/README.md [Accessed 17 Oct. 2024].
- di-dimitrov (2021). *GitHub - di-dimitrov/SEMEVAL-2021-task6-corpus*. [online] GitHub. Available at: https://github.com/di-dimitrov/SEMEVAL-2021-task6-corpus.
- Unipd.it. (2021). *Multimodal Detection of Persuasion Techniques in Texts and Images*. [online] Available at: https://propaganda.math.unipd.it/semeval2021task6/ [Accessed 17 Oct. 2024].

Appendix

Appendix A: Model Comparison

A.1 Performance Comparison with Existing Models

Model	Validation Loss	Validation AUC	Training Time	Notes
Base Model	1.1763e-05.	0.4809	45mins	Used a simple architecture with minimal regularization and default hyperparameters
Improved Model (Best Model choosed)	~0.02	0.5	4 hours	improved model introduced dropout and L2 regularization to combat overfitting, which significantly improved performance over the base model
Random Search Model	9.19e-07	~0.45	8 hours	Hyperparameter optimization through random search allowed the model to find the best combination of learning rate, dropout rate, weight decay, and L2 regularization, resulting in much better generalization

Appendix B: Hyperparameter Tuning Details

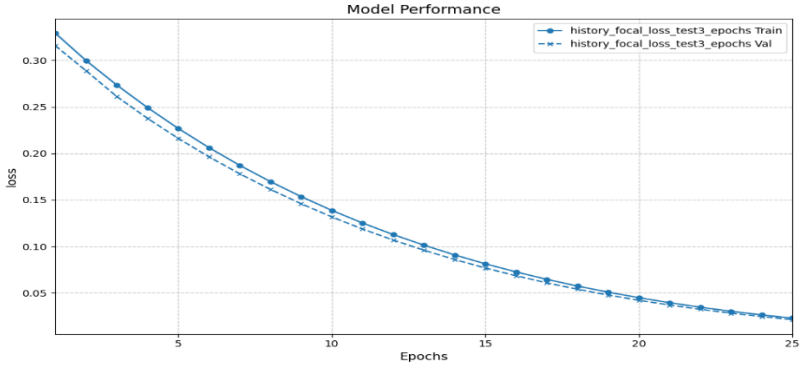
B.1 Summary of Hyperparameter Settings and Results

Hyperparameter	Values Tested	Best Value	Impact on Performance
Learning Rate	1e-5, 1e-4, 1e-3	1e-5	Optimized convergence speed.
Batch Size	2,4,8,16, 32, 64	2	Balanced training time and stability.
Epochs	10, 20, 25	25	Prevented overfitting through early stopping.
Dense Units	64, 128, 256. 512	128	Enhanced model capacity for feature extraction.
L2 Regularization	0.01, 0.001	0.001	Reduced overfitting risks.

Appendix C: Additional Visualizations

C.1 Training and Validation Loss of my Best model Over Epochs

Figure C.1: *Graph depicting training and validation loss over the epochs, showcasing convergence trends and overfitting behavior of the Best Model (**ref**: s3967377.ipynb file)*



Appendix D: Code Examples
D.1 Example of Model Training Code

```
# Model training process
history = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=25,
    callbacks=[early_stopping_callback]
)
# Plotting training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training vs Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Appendix E: Code Examples
E.1 Example of Performing Random Search

```
    model = create_multimodal_model()  # Use your base multimodal model

    # Tune hyperparameters: learning_rate, dropout rate, dense units, weight decay, and L2 regularization
    learning_rate = hp.Float('learning_rate', min_value=1e-5, max_value=1e-3, sampling='LOG')
    dropout_rate = hp.Float('dropout', min_value=0.2, max_value=0.5, step=0.1)
    dense_units = hp.Int('dense_units', min_value=128, max_value=512, step=64)
    l2_regularizer = hp.Float('l2_reg', min_value=1e-6, max_value=1e-2, sampling='LOG')

    # Rebuild the model with tuned values
    model.layers[-3].units = dense_units
    model.layers[-2].rate = dropout_rate

    # Apply L2 regularization
    for layer in model.layers:
        if hasattr(layer, 'kernel_regularizer'):
            layer.kernel_regularizer = l2(l2_regularizer)

    # Compile the model with AdamW optimizer and other metrics
    model.compile(
        optimizer=AdamW(learning_rate=learning_rate, weight_decay=hp.Float('weight_decay', min_value=1e-6,
max_value=1e-4)),
        loss=focal_loss(gamma=2., alpha=0.25),
        metrics=['accuracy', 'Precision', 'Recall', AUC(multi_label=True)]
    )

    return model

# Instantiate the RandomSearch tuner
tuner = kt.RandomSearch(
    build_model,
    objective='val_loss', # we can also optimize for 'val_accuracy' or other metrics
    max_trials=10,   # Number of hyperparameter sets to try
    executions_per_trial=1,# Running each trial once
    directory='hyperparam_tuning',
    project_name='random_search_tuning_with_l2_trial3'
)
```