BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Department of Electrical and Computer Engineering
ENCS5141—Intelligent Systems Laboratory

Assignment #2

Prepared by: Alhasan Manasra - 1211705

Instructor: Dr.Mohammad Jubran
Assistant: Eng.Hanan Awawdeh
Date: January 8, 2025

Contents

Table of figures

Table Of Tables

1 Introduction

This case study focuses on the tasks to be discussed here relate to character recognition and subjectivity classification, to deal with the challenges in extracting meaningful insights from textual data represented in quite unconventional formats. In the present study, two main tasks were considered:

1.1 Task 1: Data Extraction and Preprocessing

The first task has been to train a CNN model, make use of pre-trained models to extract sentences out of images of individual characters, and perform tasks such as proper data extraction and preprocessing from given datasets like "characters.csv" and "characters-test.csv" with mapping files such as "mappings.txt." Then, extracted sentences had to be reconstructed for further processing.

1.2 Task 2: Subjectivity Classification

In the second task, there will be involved subjectivity detection on the textual data, which has entailed the training of models using Long Short-Term Memory (LSTM) and transfer learning by utilizing a pre-trained transformer model, BERT, in classifying sentences into either subjective or objective. Experimentation with some hyperparameters to best optimize models will relate to learning rates, batch sizes, and model architectures.

This report describes the methodologies adopted to perform these two tasks, the problems encountered during the studies, and the performance obtained. The first task gives an example of how effective CNN can be in character recognition and sentence reconstruction with the help of pre-trained models, while the second task evaluates different NLP models for subjectivity classification. Results are analyzed by comparing the performances of these models in terms of accuracy, precision, recall, and F1-scores. Further, the limitations of the various approaches and their applications on the respective datasets are discussed in this report.

2 Procedure and Discussion

Task 1: Data Extraction and Preprocessing

The first part of this case study focuses on the implementation of a character recognition system using a CNN model and transfer learning with a ResNet-18 model. The objective was to preprocess the given datasets, train the models, and evaluate their performance in recognizing characters. We will compare the CNN model with the pre-trained ResNet-18 model to understand the impact of transfer learning on model accuracy and generalization. It outlines data preprocessing, the model architectures, and training methodologies used, with their respective results.

Read the training and testing datasets from their respective CSV files. The pandas library was used to load data into structured DataFrame formats that efficiently allowed manipulation and preprocessing. The image data was reshaped into the format required for processing by the CNN model. The image data was reshaped into the format required for processing by the CNN model. It involved converting each image to a 1x28x28 tensor and normalizing the pixel values by 255.0 to scale them uniformly. Numerical labels were then mapped to their corresponding characters by using a mapping file. This mapping was necessary to interpret the model's predictions in terms of meaningful characters.
The aforementioned steps generated output, which was a well-structured dataset that could then be used to train and validate the CNN model.

Initial validation accuracy across epochs-as shown in the following output-varied from 83.34% for the first epoch to 86.80% at the fifth epoch, clearly an upward slope in model performance.

```
Epoch 1, Validation Accuracy: 83.34%
Epoch 2, Validation Accuracy: 85.24%
Epoch 3, Validation Accuracy: 86.32%
Epoch 4, Validation Accuracy: 86.66%
Epoch 5, Validation Accuracy: 86.80%
```

Figure 1 - validation accuracy of the CNN model over five epochs.

Performance evaluation was done for the trained CNN model on the test dataset. First, the model is moved to the GPU for efficient computation and set to evaluation mode so that no gradient calculations take place. Then, utilizing the test dataset, a comparison between predictions made by the model and the actual labels of images was done, and then the overall test accuracy was calculated. While considering real-world, completely unseen data, it performs with 86.22% accuracy in the CNN model.

Besides test accuracy, other detailed evaluation metrics involving Precision, Recall, and F1-Score were calculated in depth to gain an understanding of how the model would behave. These metrics were calculated using the true labels and predicted labels collected

during evaluation. Precision-the ratio of the number of true positives identified correctly out of all the positive predictions-was 86.57%. Recall is the proportion of true positive samples identified correctly among all the actual positives and was 86.22%. Finally, the F1-Score, the harmonic mean of Precision and Recall, was calculated as 86.00%. These all together give an indication that the CNN model was consistently good for all measures of evaluation and hence robust and reliable.

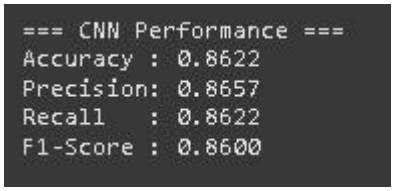

Figure 2 - Performance metrics for the CNN model

In this step, the ResNet-18 model was trained on the character dataset over five epochs with batch-wise loss monitoring and validation accuracy calculation after each epoch. The training loop processed the data in mini-batches, with the loss calculated after each batch using the CrossEntropyLoss function. The loss value was printed periodically (every 100 batches) to monitor the training progress. After all the batches within an epoch have been processed, the performance of the model is validated against a separate validation dataset. In sum, it disabled gradient calculations for evaluation, ran prediction for the labels of validation images, and then compared them against actual labels to calculate the accuracy for the validation set.

These outputs below reflect the progressive increase in model performance over the epochs. For example, the validation accuracy in the first epoch was 87.42%, gradually increasing to 88.70% in the fifth epoch. This shows that the model learned the pattern in the training data with increased performance on the validation dataset. The detailed batch-wise loss values give an insight into how the model converges during training, losses generally decrease as training progresses, which indicates the ability of the model to minimize the error and improve the accuracy of predictions.

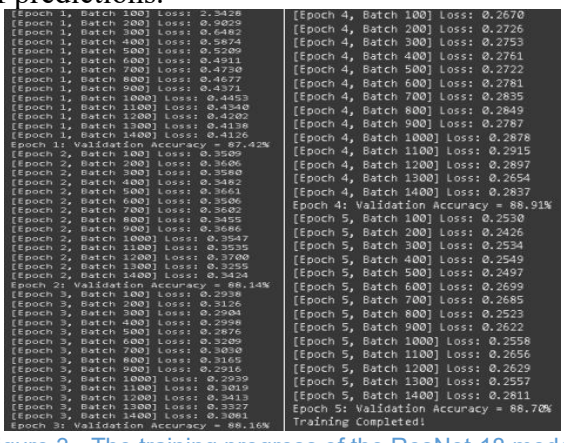

Figure 3 - The training progress of the ResNet-18 model.

The ResNet-18 model was tested on the test dataset for classification. Set the model in evaluation mode and expand grayscale images to three channels before making inferences. Predictions were made for each batch, and metrics such as accuracy, precision, recall, and F1-score were calculated. This model achieved an accuracy of 88.56%, precision of 89.02%, recall of 88.56%, and F1-score of 88.43%. These reflect very good consistency in performance, whereby the model was able to correctly classify characters of the dataset in balance, considering both precision and recall.

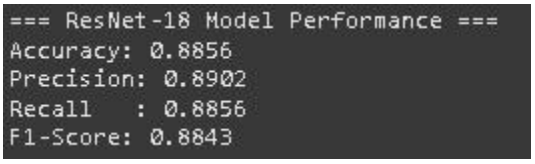

Figure 4 - ResNet-18 Model Performance Metrics

Metric	CNN Model	ResNet-18 Model
Accuracy	0.8622	0.8856
Precision	0.8657	0.8902
Recall	0.8622	0.8856
F1-Score	0.8600	0.8843

Table 1 - Model Comparison: CNN vs ResNet-18

The table compares the performance of two models, CNN and ResNet-18, based on four key metrics: Accuracy, Precision, Recall, and F1-Score. ResNet-18 consistently outperforms CNN across all metrics, achieving higher Accuracy 88.56%, Precision 89.02%, Recall 88.56%, and F1-Score 88.43%. This indicates that ResNet-18 provides better overall performance and is more effective for the given task compared to the CNN model.

Part 2: Comparative Analysis of Classification Techniques

I do the subjectivity classification in Task 2 using two methods: an LSTM model and transfer learning of pre-trained transformers. The task starts by preparing the data and loading the training and testing datasets from the TSV files provided. I structurize the dataset using the pandas library for convenience in manipulation. First, check if the GPU is available to ensure training speed.

I used the BERT tokenizer for text preprocessing, tokenizing sentences into numerical input IDs with regard to attention masks that can be understood by transformer-based models. After that, label encoding for target labels was done to represent categorical labels in numerical format.

Then, I will define a class for PyTorch Dataset that will handle both the tokenized inputs and labels for the training and testing datasets. It will create some DataLoader objects

helpful for batch processing during the training and evaluation steps. Preparation of data with the view to LSTM Model and transformers integration.

The architecture will include embedding, LSTM, dropout, and finally, a fully connected layer to do the output classification. Moving on, the training of this model over five epochs resulted in capturing loss and accuracy from each step, hence gradual improvements could be depicted on the performance of the trained model.
Finally, the performance measures-accuracy, precision, recall, and F1-score-have been used to evaluate the LSTM and BERT model on the test set. The results of the test will give good marks for strengths and weaknesses of the proposed LSTM and BERT model to act as the ground for other results produced by transfer learning-based models. This step laid a systematic foundation for subjectivity classification and further experimentation using advanced techniques.

The output gives the training performances over five epochs of the LSTM model: per epoch, the total training loss and accuracy. The loss values are slightly decreasing over the epochs, meaning the model learns to lower its error in training. However, the accuracy is still constant at 64.10%, which does not seem to increase much by adding more epochs. This could mean that the model architecture, hyperparameters, or quality of the dataset is limiting further improvements in accuracy. This may call for further fine-tuning or adjustments in order to make the model effective.

```
Epoch 1/5, Loss: 34.1264, Accuracy: 0.6410
Epoch 2/5, Loss: 34.0484, Accuracy: 0.6410
Epoch 3/5, Loss: 34.0725, Accuracy: 0.6410
Epoch 4/5, Loss: 34.0623, Accuracy: 0.6410
Epoch 5/5, Loss: 33.9617, Accuracy: 0.6410
```

Figure 5 - The training performance of the LSTM model across five epochs.

The LSTM model's performance metrics, as shown in the results. The accuracy value rests at 47.74%, an indication that less than half of all its predictions have been correct. A precision of 23.87% reflects a very high number of false alarms, while for recall, this is 50%, meaning half of the related instances have been found. The F1-score, therefore, balanced precision and recall at 32.31%, which reflected that the model was limited in handling the classification task. These results confirm our hypothesis that the LSTM model could not generalize well, most likely due to being limited by its architecture, training data, or a hyperparameter.

```
========================================
LSTM Model Performance Metrics:
========================================
Metric          Score
----------------------------------------
Accuracy        0.4774
Precision       0.2387
Recall          0.5000
F1-Score        0.3231
========================================
```

Figure 6 - The LSTM model's performance metrics

In this step of Task 2, the BERT model was trained and evaluated for subjectivity classification. The model was trained over 5 epochs using the training data, with the loss calculated at each step using the CrossEntropyLoss function. After each epoch, the model was evaluated on the validation set to calculate accuracy. The predictions were made using the torch.argmax function, and the accuracy score was calculated by comparing predicted labels against the true labels. On the whole, it has a constant validation accuracy of about 73.25% across all epochs; this could mean that the model is stable but may need some more tuning or change in hyperparameters for improvement.

```
Epoch 1/5, Loss: 0.3809, Accuracy: 0.7325
Epoch 2/5, Loss: 0.4171, Accuracy: 0.7325
Epoch 3/5, Loss: 0.4158, Accuracy: 0.7325
Epoch 4/5, Loss: 0.3599, Accuracy: 0.7325
Epoch 5/5, Loss: 0.3990, Accuracy: 0.7325
```

Figure 7 - BERT Model Training and Validation Accuracy Across Epochs

The BERT model performance on this step is evaluation for a test dataset, to create predictions over every input from a test dataset and compare it against the ground truth labels. Hence, for computing the set of evaluation metrics in terms of performance for the trained classification model, accuracy, precision, recall, and F1-score provide comprehensive insight.

With this, the BERT model yielded an accuracy of 0.7325, which is around 73.25% of correct predictions. A precision of 0.7584 tells about the capability of the model in avoiding false positives. Recall equaled 0.7392 and highlighted the ability of the model in catching true positives, while the F1-score equaled 0.7289 and tried to balance both precision and recall, therefore giving a general view of how good the model was.

```
BERT Model Performance:
Accuracy: 0.7325
Precision: 0.7584
Recall: 0.7392
F1-Score: 0.7289
```

Figure 8 - BERT Model Performance Metrics for Subjectivity Classification

Metric	LSTM Model	BERT Model
Accuracy	0.4774	0.7325
Precision	0.2387	0.7584
Recall	0.5	0.7392
F1-Score	0.3231	0.7289

Table 2 - Comparison of LSTM and BERT Model Performance Metrics

3 Conclusion

This assignment consists of two tasks dealing with different domains within machine learning. In one, a trained and tested convolutional neural network (CNN) and pre-trained ResNet-18 for character classification on images were performed. It showed the relative superiority in scores of performance according to accuracy, precision, recall, and F1-score by ResNet-18 over CNN because it leverages pre-trained results on diverse, complex datasets.

The tasks were then performed with LSTM and BERT for subjectivity classification. The BERT model outperformed the LSTM in all parameters of evaluation, reflecting the power of transfer learning and robustness that the transformer-based architecture has for text classification tasks. These evaluation metrics really gave detailed insight into the strengths and limitations of each model, enabling thorough comparison.

Overall, the assignment showed the steps of proper pre-processing of the data, tuning, and evaluation, which are very necessary. The big point to take away is that pre-trained models do better since they can make use of prior knowledge like ResNet-18 or BERT, but they contain computational complexity, and their outputs have to be tuned with much care to get the best from them. Systematic implementation and heavy testing mitigated the challenges faced during data preprocessing and training to produce meaningful results and valuable learning outcomes.