جامعة بيرزيت
BIRZEIT UNIVERSITY

Faculty of Engineering & Technology

Electrical & Computer Engineering Department

Intelligent Systems Laboratory ENCS5141

Case Study #2

Character Recognition and Subjectivity Detection

Prepared by:

Hiba Jaouni 1201154

Instructor: Dr. Mohammad Jubran

Section: 3

Date: Jan 8th 2025

Abstract

This report explores two machine learning tasks: character recognition and subjectivity detection. In character recognition, both a Convolutional Neural Network (CNN) and a pre-trained ResNet-18 model were tested, with ResNet-18 outperforming the CNN through transfer learning. In subjectivity detection, Long Short-Term Memory (LSTM) networks and BERT-based transfer learning were compared, with BERT achieving superior performance. The study emphasizes the impact of hyperparameter tuning and transfer learning in enhancing model accuracy and performance across different tasks.

Table of Contents

Table of Tables

1. Introduction

This report focuses on the implementation and evaluation of two tasks: **Character Recognition** and **Subjectivity Detection**, employing state-of-the-art deep learning techniques and models. The assignment is divided into two parts, each addressing a specific problem using different methodologies to explore the strengths of various machine learning approaches.

In the first part, **Character Recognition**, the task involves identifying characters from pixel-based image data using a Convolutional Neural Network (CNN) and a pre-trained ResNet-18 model. The CNN is built from scratch to learn character patterns directly from the dataset, while the ResNet-18 model leverages transfer learning to enhance accuracy and generalization. The comparison highlights the performance benefits of pre-trained models, which utilize knowledge from large datasets, making them more robust and efficient compared to training models from scratch.

The second part, **Subjectivity Detection**, focuses on classifying text data as subjective or objective by applying Long Short-Term Memory (LSTM) networks and the pre-trained BERT model. The LSTM network captures sequential dependencies within the text, making it well-suited for analyzing contextual information. In contrast, the BERT model utilizes transfer learning to provide a deep contextual understanding of language, outperforming traditional models with its ability to handle nuanced linguistic patterns.

This report explores the effectiveness of these techniques in solving their respective tasks, comparing performance metrics to assess the impact of architecture choice, hyperparameters, and pre-training.

2. Task 1: Character Recognition

Task one focuses on implementing character recognition using a dataset represented in two CSV files: one for training and the other for testing. Each file contains data where the first column represents the label, and the remaining 784 columns correspond to pixel values of the character images. The label column is mapped to its corresponding ASCII value using the provided mapping file, mapping.txt. This mapping allows the raw pixel data to be interpreted as meaningful characters or letters.
Two approaches are employed to process this dataset: a Convolutional Neural Network (CNN) and a pre-trained ResNet-18 model. The CNN is trained from scratch to learn patterns from the pixel data, while the ResNet-18 model leverages transfer learning to enhance performance. Hyperparameter tuning, including adjustments to learning rate and batch size. The pre-trained ResNet-18 outperformed the CNN, providing a stronger foundation for the subsequent NLP-based subjectivity detection task.

2.1 Character Recognition Using CNN Model

The table below evaluates the performance of the model using various hyperparameter configurations, including learning rate, batch size, and the two configurable convolution kernel sizes: Kernel Size1 (first convolution layer) and Kernel Size2 (second convolution layer).

Table 2.1: CNN Hyperparameter-tuning Results

Learning Rate	Batch Size	Kernel Size1	Kernel Size2	Accuracy	Precision	Recall	F1 Score
0.01	32	5	3	0.6491	0.6710	0.6476	0.6423
0.01	32	5	5	0.6486	0.6731	0.6490	0.6462
0.01	32	3	3	0.0206	0.0004	0.0213	0.0009
0.01	32	3	5	0.0217	0.0005	0.0213	0.0009
0.01	64	5	3	0.6758	0.6837	0.6758	0.6725
0.01	64	5	5	0.6713	0.7046	0.6714	0.6623
0.01	64	3	3	0.6316	0.6488	0.6321	0.6229
0.01	64	3	5	0.6221	0.6378	0.6238	0.6177
0.001	32	5	3	0.7632	0.7698	0.7636	0.7594
0.001	32	5	5	0.7743	0.7840	0.7724	0.7662
0.001	32	3	3	0.7565	0.7653	0.7563	0.7524
0.001	32	3	5	0.7597	0.7812	0.7585	0.7601
0.001	64	5	3	0.7735	0.7851	0.7739	0.7719
0.001	64	5	5	0.7716	0.7805	0.7710	0.7682
0.001	64	3	3	0.7616	0.7748	0.7621	0.7562
0.001	64	3	5	0.7587	0.7664	0.7601	0.7570
0.003	32	5	3	0.7507	0.7647	0.7532	0.7495
0.003	32	5	5	0.7452	0.7604	0.7485	0.7450
0.003	32	3	3	0.7368	0.7503	0.7374	0.7342

0.003	32	3	5	0.7401	0.7535	0.7423	0.7390
0.003	64	5	3	0.7463	0.7587	0.7476	0.7452
0.003	64	5	5	0.7510	0.7648	0.7513	0.7492
0.003	64	3	3	0.7316	0.7463	0.7327	0.7292
0.003	64	3	5	0.7359	0.7498	0.7364	0.7336

The results highlight that a learning rate of 0.001 consistently provided superior performance, yielding higher accuracy, precision, recall, and F1 scores across most configurations. This is likely because a lower learning rate allows the model to converge more gradually, avoiding overshooting the optimal weights during training. In contrast, a higher learning rate (e.g., 0.01) likely caused instability, particularly with smaller kernel sizes (e.g., Kernel Size1: 3, Kernel Size2: 3), leading to lower performance due to the model failing to settle into a stable optimization path. Larger kernel sizes (e.g., Kernel Size1: 5, Kernel Size2: 5) tended to perform better as they capture broader spatial relationships in the input data, which may enhance feature extraction and ultimately improve model predictions.

Batch size also influenced performance, with Batch Size: 32 achieving slightly better precision and F1 scores compared to Batch Size: 64. Smaller batch sizes often provide a more robust gradient estimation due to frequent updates, which can help in scenarios where the dataset has variability or noise. However, larger batch sizes can stabilize training but may miss finer-grained adjustments, potentially explaining the slightly reduced performance. The best-performing configuration, Learning Rate: 0.001, Batch Size: 32, and Kernel Size1: 5, Kernel Size2: 5, likely succeeded by maintaining a balanced optimization process and effectively capturing the underlying data patterns.

2.2 Character Recognition Using Pre-trained ResNet-18 Model'

This section presents the results of experiments conducted to evaluate the performance of a pretrained ResNet18 model under varying hyperparameters. The model was fine-tuned with different learning rates, batch sizes, and kernel sizes in its convolutional layers. Specifically, the learning rates tested were 0.01, 0.001, and 0.003, while batch sizes of 32 and 64 were used. The experiments aimed to identify the optimal combination of these parameters by analyzing performance metrics such as accuracy, precision, recall, and F1-score. The table below summarizes the outcomes of these experiments.

Table 2.2: ResNet18 Hyperparameter-tuning Results

Learning Rate	Batch Size	Accuracy	Precision	Recall	F1-Score
0.01	32	0.7349	0.7404	0.7349	0.7319
0.01	64	0.7430	0.7484	0.7430	0.7413
0.001	32	0.7716	0.7736	0.7716	0.7706
0.001	64	0.7486	0.7532	0.7486	0.7463
0.003	32	0.7610	0.7635	0.7610	0.7586
0.003	64	0.7534	0.7554	0.7534	0.7519

The results from the experiments highlight the significant impact of learning rate and batch size on the performance of the pretrained ResNet18 model. A learning rate of 0.001 consistently outperformed the other values, achieving the highest Accuracy (0.7716), Precision (0.7736), and F1-Score (0.7706). This lower rate allowed for stable training, resulting in better convergence and generalization. On the other hand, higher learning rates like 0.01 caused the model to converge too quickly, leading to suboptimal performance due to large weight updates that skipped the optimal minima.
Regarding batch size, batch size 32 generally provided better performance than batch size 64, with the former achieving the best results in combination with a learning rate of 0.001. Smaller batch sizes tend to help with better generalization, despite longer training times. The best-performing configuration was learning rate 0.001 with batch size 32, which provided the highest overall metrics, making it the recommended setup for further fine-tuning.

The pre-trained ResNet-18 model outperformed the CNN by leveraging transfer learning, enabling it to achieve better accuracy and generalization. Pre-trained models benefit from prior knowledge gained on large datasets, making them more effective than models built from scratch, which require more data and time to learn patterns.

3. Task 2: Subjectivity Detection

This section focuses on detecting subjectivity in sentences provided in the training and testing TSV files, classifying each sentence as either subjective (SUBJ) or objective (OBJ). Two models are utilized to achieve this: Long Short-Term Memory (LSTM) and Transfer Learning with BERT. Hyperparameter tuning, including adjustments to batch size, learning rate, and the number of layers, is performed to optimize model performance. The results indicate that Transfer Learning with BERT significantly outperformed the LSTM model.

3.1 Subjectivity Detection using LSTM

The training results for the LSTM model show that variations in hyperparameters like learning rate, batch size, and the number of layers lead to modest performance changes. Increasing the learning rate improves accuracy and precision up to a point, after which larger values cause instability. Lower learning rates slow convergence but improve generalization, especially with more layers. Smaller batch sizes allow for frequent updates but introduce noisy gradients, while larger sizes stabilize training but may reduce accuracy. Deeper networks show marginal improvements, with configurations of 4 or 5 layers performing similarly. These variations highlight the subtle interplay between hyperparameters in optimizing the model.

For the LSTM model training, the following hyperparameters were used:

- Learning Rates: [0.01, 0.001, 0.003, 0.005, 0.0001]
- Batch Sizes: [32, 64, 128]
- Number of Layers: [2, 3, 4, 5]

The table below shows some results obtained from different models during hyperparameter tuning and their corresponding performance metrics.

Table 3.1: Part of Hyperparameter-tuning Results for LSTM

Learning Rate	Batch Size	Num Layers	Accuracy	Precision	Recall	F1	Remarks
0.005	32	5	0.5144	0.5524	0.5295	0.4633	Best performing configuration with highest F1 score
0.001	32	5	0.4979	0.6072	0.5186	0.3827	High precision but slightly lower F1
0.005	64	4	0.4856	0.5738	0.5071	0.3530	Strong balance between precision and recall
0.001	128	5	0.4979	0.5896	0.5182	0.3880	Similar performance to other 5-layer setups
0.003	32	5	0.4897	0.5596	0.5103	0.3726	Slightly lower accuracy but competitive F1
0.01	32	2	0.4774	0.2387	0.5000	0.3231	Low precision, basic configuration
0.01	32	3	0.4774	0.2387	0.5000	0.3231	Same as 2 layers, with no improvement
0.003	128	2	0.4733	0.4042	0.4953	0.3279	Lowest performance, especially in accuracy
0.01	128	2	0.4897	0.5987	0.5111	0.3612	High precision, but not the best overall F1
0.0001	128	5	0.4774	0.2387	0.5000	0.3231	Extremely low precision and recall

The configuration with a learning rate of 0.005, batch size of 32, and 5 layers is the best performing model, as it achieved the highest F1 score (0.4633), a good balance between precision (0.5524) and recall (0.5295), and the highest overall accuracy (0.5144).

A higher learning rate (e.g., 0.01) can cause the model to take large steps during training, leading to overshooting the optimal weights, which negatively impacts precision and recall, especially in simpler models with fewer layers. On the other hand, smaller batch sizes (e.g., 32) enable more frequent updates, allowing the model to learn more nuanced patterns, and deeper architectures (e.g., 5 layers) provide greater capacity to capture complex relationships, leading to improved performance.

The lowest performance occurs with configurations using a lower learning rate (0.0001) and larger batch sizes (128), especially with fewer layers, where both precision and recall are compromised. In practice, a very low learning rate (like 0.0001) can cause the model to converge too slowly, preventing it from learning meaningful patterns efficiently. Meanwhile, larger batch sizes (like 128) reduce the frequency of weight updates, resulting in slower adaptation and less fine-tuned adjustments, especially when the model is shallow with fewer layers.

3.2 Subjectivity Detection Using Transfer Learning with BERT

The Bidirectional Encoder Representations from Transformers (BERT) model was fine-tuned for the task of subjectivity detection using various combinations of hyperparameters, including different learning rates and batch sizes. The model's performance was evaluated across 10 training epochs, with metrics such as accuracy, precision, recall, and F1 score being recorded. The table below presents the results for each configuration, highlighting the impact of hyperparameter tuning on the model's effectiveness. Additionally, the best-performing model is identified based on the F1 score, which balances precision and recall.

Table 3.2: BERT Hyperparameter-tuning Results

Learning Rate	Batch Size	Accuracy	Precision	Recall	F1 Score
0.01	32	0.4774	0.4774	1.0000	0.6462
0.01	64	0.4774	0.4774	1.0000	0.6462
0.001	32	0.4774	0.4774	1.0000	0.6462
0.001	64	0.4774	0.4774	1.0000	0.6462
2e-05	32	0.7284	0.6812	0.8103	0.7402
2e-05	64	0.7037	0.6410	0.8621	0.7353

The results demonstrate that hyperparameter tuning significantly affects the performance of the BERT model. When the learning rate is set too high (e.g., 0.01), the model struggles to converge effectively, as indicated by the consistently low accuracy, precision, recall, and F1 scores. This occurs because a high learning rate causes the model to overshoot the optimal parameters during training, leading to suboptimal performance.

Conversely, lowering the learning rate to 2e-05 allowed the model to achieve better performance. This learning rate facilitates more gradual and precise updates to the model weights, enabling the model to converge effectively. Additionally, a batch size of 32 yielded slightly better results than a batch size of 64. A smaller batch size often allows the model to make more frequent updates to the weights, improving its ability to generalize.

The best-performing configuration for BERT used a learning rate of 2e-05 and a batch size of 32, achieving an F1 score of 0.7402. This balance of precision (0.6812) and recall (0.8103) highlights its ability to identify both positive and negative cases accurately.

The pre-trained BERT model outperformed LSTM by leveraging transfer learning, enabling it to achieve higher accuracy and better generalization. Pre-trained models like BERT benefit from extensive training on large datasets, allowing them to understand complex language patterns more effectively than models like LSTM, which rely solely on task-specific training and lack pre-existing knowledge.

4. Conclusion

In conclusion, this study demonstrates the effectiveness of advanced machine learning techniques in character recognition and subjectivity detection. The use of transfer learning with ResNet-18 significantly enhanced character recognition accuracy, while the BERT-based model outperformed LSTM networks in subjectivity detection. These findings underscore the importance of selecting the right models and utilizing hyperparameter optimization for improving performance.