

Final

by Wenhao Zhong

Submission date: 18-Apr-2024 02:10PM (UTC+0100)

Submission ID: 229835698

File name: 50294_Wenhao_Zhong_Final_571450_1432746892.docx (474.85K)

Word count: 5446

Character count: 31907



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Artificial Intelligence

Title:

Application of meta ensemble deep learning techniques for sentiment
analysis on IMDB dataset

Student number:

2304553

Student name:

Wenhao Zhong

Date of submission

12/04/2024

Contents

I.	Introduction.....	3
II.	Related work in the literature	3
III.	Methodology of base models and developed ensemble learning models	4
	3.1. Convolutional Neural Network (CNN)	5
	3.2. Long Short-Term Memory network (LSTM)	6
	3.3. Gated Recurrent Unit (GRU).....	6
	3.4. Bidirectional Encoder Representations from Transformers (BERT)	6
	3.5. Developed ensemble learning model.....	7
IV.	Results and analysis	7
V.	Conclusions.....	8

Application of meta ensemble deep learning techniques for sentiment analysis on IMDB dataset

Wenhao Zhong*

*Wales Institute of Science and Art, University of Wales Trinity Saint David, Swansea SA1 8EW, United Kingdom
Email: 2304553@student.uwtsd.ac.uk

Abstract— This study investigates the performance of convolutional neural network (CNN), long short-term memory network (LSTM), gated recurrent unit (GRU), and bidirectional encoder representations from transformer (BERT) models, both individually and in an ensemble setting, on the IMDB dataset for sentiment analysis. Additionally, a stacked ensemble model combining CNN, LSTM, GRU, and BERT using weighted averaging technique is proposed and evaluated. The experimental results demonstrate that the ensemble model outperforms individual models, achieving an accuracy of approximately 85.9%. The study highlights the effectiveness of combining diverse deep learning architectures and ensemble techniques for improving sentiment analysis tasks.

Keywords- Convolutional Neural Network; Long Short-Term Memory Network; Gated Recurrent Unit; Bidirectional Encoder Representations from Transformer; Ensemble Learning; Sentiment Analysis;

I. INTRODUCTION

In the realm of artificial intelligence (AI), the fusion of deep learning and ensemble learning techniques has emerged as a powerful paradigm for enhancing predictive performance and robustness in various applications. Deep learning, characterized by its ability to automatically learn hierarchical representations from data, and ensemble learning, which leverages the collective wisdom of multiple models [1]. This synergistic combination, known as meta ensemble deep learning, holds promise for addressing complex tasks such as sentiment analysis [2].

Sentiment analysis, a subfield of natural language processing (NLP), plays a pivotal role in numerous domains. By discerning the sentiment conveyed in textual content, businesses can gauge customer satisfaction, monitor public opinion, and make data-driven decisions. Maas *et al.* [3] introduced a large dataset of movie reviews which is called IMDB. The Internet Movie Database (IMDB), a vast repository of movie reviews and ratings, serves as a robust benchmark for sentiment analysis, with sentiments ranging from praise to criticism in user-generated reviews.

The primary objective of this study is to explore the application of meta ensemble deep learning techniques for sentiment analysis on the IMDB dataset. Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), Gated Recurrent Units (GRUs), and Bidirectional Encoder Representations from Transformers (BERT) are experimented and CNN, LSTM, GRU are combined using stacking as well as weighted averaging technique to implement an ensemble learning model.

The remaining parts of this study are organized as follows. In Section II, related previous studies are discussed. Section III is used to describe the methodology of comparing four base models including CNN, LSTM, GRU and BERT and the ensemble learning model with various configurations and

hyperparameters. In Section IV, the results of experiments including accuracy, precision, recall, F1-score and confusion matrix are presented and discussed based on Section III. Performances of developed models are compared with some of the previous studies. Finally, the conclusion will be derived in Section V.

II. RELATED WORK IN THE LITERATURE

In Section II, this statement is summarized/also describe some existing deep learning and deep learning methods for sentiment analysis including CNN, LSTM, GRU, BERT and various ensemble learning models.

Sentiment analysis aims to automatically classify text documents into predefined sentiment categories such as positive, negative, or neutral. However, accurately analyzing sentiment from unstructured text data poses several challenges, including the presence of ambiguity, sarcasm, and context-dependent language usage. Furthermore, traditional machine learning approaches often struggle to capture the complex relationships present in natural language. In this context, deep learning techniques, including CNN, LSTM, GRU and BERT, have emerged as powerful tools for sentiment analysis. Moreover, single models are combined together to get better performance usually in existing literatures.

Single deep learning models can outperform traditional ML models and keep lower time complexity than combined models in general. I. Kanwal *et al.* [4] introduced a hybrid model combining Stacked Auto-Encoder (SAE) for feature extraction and LSTM for sentiment classification, achieving 87% accuracy on IMDB dataset. M. Sen and K. Chaturvedi [5] utilized GRU architecture for sentiment analysis, achieving high accuracy and low loss. the importance of sentiment analysis in various industries is discussed and the effectiveness of deep learning models is highlighted, particularly GRU. K. Arora *et al.* [6] employed Bidirectional Encoder Representations from Transformers (BERT) for sentiment analysis on IMDB movie reviews. The proposed scheme achieves high precision, recall, and F1-Score for both positive and negative classes, with an overall accuracy of 92.40%.

Generally, combined models perform better than single models but have higher time complexity. S. Gogineni and A. Pimpalshende [7] contrasted classical machine learning algorithms and deep learning architectures on the IMDB movie review dataset. Focuses on text preprocessing and experiments with advanced techniques such as RNN, GRU, and LSTM. M. M. Rahman *et al.* [8] proposed a deep learning model combining CNN and LSTM for product feedback classification, achieving 92% accuracy on IMDB dataset, outperforming traditional ML algorithms. A. Yenter and A. Verma [9] developed a novel approach combining CNN and LSTM with multiple kernel branches for

sentiment analysis on IMDB dataset, achieving accuracy above 89% and surpassing previous models. M. M. Krishna *et al.* [10] utilized a hybrid approach combining CNN and BiLSTM for sentiment analysis on IMDB dataset, achieving superior results compared to traditional deep learning and machine learning methods. A. Rasool [11] introduced WRS, a novel word embedding method utilizing Word-to-Word Graph (W2WG) embedding, integrated with LSTM-CNN model for real-time sentiment analysis, outperforming traditional ML techniques and achieving effectiveness on Twitter and IMDB datasets. L. Zhang *et al.* [12] proposed an LSTM-Adaboost method for sentiment analysis on movie reviews, achieving higher accuracy compared to CNN and LSTM methods on the IMDB dataset. Y. Cheng *et al.* [13] introduced a multi-channel model combining CNN and bidirectional GRU with attention mechanism (MC-AttCNN-AttBiGRU) for text sentiment analysis, achieving better results than baseline models on IMDB and Yelp 2015 datasets. A. Zouzou and I. E. Azami [14] proposed a sentiment analysis method using GloVe word embedding and a CNN-GRU model on the IMDB dataset. The CNN-GRU model achieves a training accuracy of 86.34%, outperforming CNN and GRU models with other optimizers. C. Vielma *et al.* [15] explored CNN-GRU and CNN-bidirectional GRU models for sentiment analysis using IMDB movie reviews. Both models demonstrate high accuracy, with CNN-GRU achieving comparable results in less training time. L. B. N *et al.* [16] investigated deep learning methods (Bi-LSTM, GRU) and word embeddings (BERT, RoBERTa) for sentiment analysis on Twitter and IMDB datasets, emphasizing effectiveness and insights for real-world applications. K. Mouthami *et al.* [17] explored sentiment analysis of film reviews using Bi-LSTM and GRU models. It achieves high accuracy, with Bi-LSTM and GRU reaching 98.65% and 98.24% accuracy respectively, on IMDB movie reviews. R. I. Rudra and A. K. Gopalakrishnan [18] focused on sentiment analysis using RNN with different layers of GRU and BiGRU, utilizing GloVe word embeddings. Compares various algorithms on IMDB dataset, highlighting RNN-GRU and RNN-BiGRU models' superiority.

However, a part of metrics of base models can outperform combined models in some situations. M. Mishra and A. Patil [19] investigated sentiment classification of IMDB movie reviews using CNN, LSTM, and CNN-LSTM architectures, finding LSTM architecture outperforming others with 87% accuracy. Combination of CNN-LSTM reduces training time for larger datasets. M. R. Haque *et al.* [20] compared CNN, LSTM, and LSTM-CNN architectures for sentiment analysis on IMDB movie reviews, finding CNN to

outperform others with a 91% F-Score. M. Qamar *et al.* [21] proposed a domain-independent sentiment analysis model trained on datasets from various domains. It evaluates five deep learning models: CNN, CNN-GRU, CNN-LSTM, LSTM, and GRU. Results show GRU and LSTM models perform best, achieving high accuracy across different datasets, including IMDB movie reviews and a novel book review dataset from Amazon.

To improve accuracy, ensemble deep learning methods are introduced to tackle sentiment analysis tasks. Á. Kovács and T. Tajti [22] explored various machine learning models including BERT, LSTM, and GRU for sentiment analysis on IMDB movie reviews dataset. Ensemble methods such as voting functions and stacking are implemented to improve classification accuracy, demonstrating superiority over single-model approaches. Á. Kovács and T. Tajti [23] explored ensemble machine learning methods, including voting functions and stacking, various models such as BERT, LSTM, and GRU, are combined to improve classification accuracy for both positive and negative reviews. K. J. Sai *et al.* [24] proposed an ensemble machine learning approach for aspect-based sentiment analysis on IMDB reviews. Traditional lexicon-based methods are reviewed, and an ensemble model is developed, achieving 89% accuracy, outperforming the lexicon-based approach. The stacking classifier ensemble strategy is utilized, resulting in a 1% accuracy improvement by combining various machine learning algorithms. W. M. D. R. Ruwantha *et al.* [25] presented an LSTM and ensemble-based approach to predict movie success using both social media data and movie metadata. LSTM is used to identify sentiment values from Twitter posts, and an ensemble approach combines these sentiment values with movie metadata to predict success. The combined model achieves 81.2% accuracy, outperforming other models. A. Ö. Eren and M. Sert [26] introduced a method for predicting movie user ratings by leveraging both numerical and nominal features through ensemble learning. The approach is validated on IMDB performance dataset and outperforms existing methods, demonstrating the effectiveness of mixed data types in improving movie rating prediction accuracy. K. L. Tan *et al.* [27] introduced an ensemble hybrid deep learning model for sentiment analysis, combining RoBERTa, LSTM, BiLSTM, and GRU. RoBERTa transforms input sequences into embeddings, while LSTM, BiLSTM, and GRU capture long-range dependencies. Ensemble methods like averaging and majority voting enhance overall performance, achieving high accuracy across IMDB, Twitter US Airline Sentiment, and Sentiment140 datasets.

III. METHODOLOGY OF BASE MODELS AND DEVELOPED ENSEMBLE LEARNING MODELS

In this section, we use four base deep learning models including CNN, GRU, BERT and one ensemble learning model to classify reviews given by IMDB dataset. Section III is divided into five parts comprising comprehensive analysis of codes.

The introduction to the IMDB Dataset is given as follows. The IMDB dataset is used for binary sentiment analysis tasks, containing 50000 movie reviews along with their associated

sentiment labels. Maas *et al.* [3] provide a set of 25,000 training and 25,000 for testing. Both of labeled training and testing reviews consist of half positive reviews and negative reviews, where number 1 is denoted as positive and number 0 is denoted as negative.

In each part of Section III, there are three subparts including 1. Data acquisition and preprocessing, 2. Model architecture, 3. Training and evaluation. The functions of codes of data acquisition as well as preprocessing and training as well as evaluation for CNN, LSTM, GRU is

almost the same. Consequently, subpart one and three will be explained in the first part “Convolutional Neural Networks (CNN)” and briefly summarized in the following two parts.

3.1. Convolutional Neural Network (CNN)

CNN is primarily used for image recognition and classification tasks, leveraging convolutional layers to extract features from input images.

To be ready for data acquisition and preprocessing for CNN, in Fig.1, we imported necessary libraries comprising internal module of tensorflow, sklearn, matplotlib and numpy.

```
import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D, Flatten, Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# Load the IMDB dataset
num_words = 10000
maxlen = 100
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=num_words)

# Pad sequences to have the same length
X_train = pad_sequences(X_train, maxlen=maxlen)
X_test = pad_sequences(X_test, maxlen=maxlen)
```

Fig.1. Import necessary libraries for the CNN

To load the IMDB dataset, “num_words” parameter is introduced. In IMDB dataset, words are labelled as numbers according to occurring frequency. As the frequency goes up, the index number goes down. “num_words=10000” operation keeps the top 10000 most frequently occurring words as the same index number and sets others as number 2. This operation can simplify the calculation complexity.

In the part of data preprocessing, “pad Sequences” function is used to pad the sequences to have the same length, which is required for neural network models. The parameter “maxlen” is introduced to infer the maximum length of all sequences. Other parameters are default in the code. If a sequence is shorter than “maxlen”, the function will pad sequences at the beginning. If a sequence is longer than “maxlen”, the function will truncate it at the beginning to reduce the number of words in the sentence.

Model architecture of CNN is illustrated In Fig.2. We defined the CNN architecture as the combination of embedding layer, convolutional layer, pooling layer, flatten layer and two dense layers.

```
# Define the CNN architecture
model = Sequential([
    Embedding(num_words, 128),
    Conv1D(64, 5, activation='relu'),
    MaxPooling1D(pool_size=4),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1)

# Predictions
y_pred = (model.predict(X_test) > 0.5).astype(int)
```

Fig.2. Define the structure and hyperparameters of the CNN model

To break down each function used in the “Sequential” model construction: the Embedding layer is used to create

word embeddings, it takes the input vocabulary size (num_words=10000), the dimensionality of the embeddings (128), and the maximum length of input sequences (maxlen=100). Each word in the input sequences is replaced by its corresponding embedding vector. The Conv1D layer performs 1D convolution over the input sequences. Here, it has 64 filters with a kernel size of 5, applying the ReLU activation function to the convolution results. This layer is useful for extracting local patterns and features from the input sequences. The MaxPooling1D layer performs max pooling operation along the temporal dimension of the input sequences. It reduces the dimensionality of the feature maps obtained from the convolutional layer by taking the maximum value within each pooling window of size 4. This helps in capturing the most important features while reducing computational complexity. The Flatten layer is used to flatten the output from the previous layers into a one-dimensional vector. It converts the multi-dimensional feature maps into a single continuous vector, which can then be fed into the fully connected layers. Two Dense layers follow the flattened output. The first Dense layer has 64 units and uses the ReLU activation function. It serves as a hidden layer, learning complex patterns in the flattened feature representation. The second Dense layer has 1 unit with a sigmoid activation function, which outputs the binary classification prediction (positive or negative sentiment) for each input sequence.

In the part of compiling model, optimizer= “adam” sets the Adam optimizer, which is a popular optimization algorithm known for its efficiency and good performance in training deep neural networks. loss= “binary_crossentropy” sets the binary cross-entropy loss function, which is commonly used for binary classification problems. It measures the difference between the actual and predicted probability distributions. metrics=[“accuracy”] specifies that the accuracy metric should be computed during training.

In the training section, “epochs=5” denotes the entire training dataset is passed forward and backward through the neural network five times. “batch_size=32” defines there are 32 samples that will be propagated through the network. “validation_split=0.1” splits off a fraction (10%) of the training data as validation data. The model's performance on this validation data is computed after each epoch, allowing us to monitor its generalization ability and detect overfitting.

Training and evaluation process of CNN model is shown in Fig.3. Metrics including accuracy, precision, recall, F1 score and confusion matrix are calculated and displayed. Moreover, confusion matrix is visualized using matplotlib and seaborn libraries.

```
# Evaluation
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(conf_matrix)

import matplotlib.pyplot as plt
import seaborn as sns

# Plot confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

Fig.3. Train and evaluate the CNN model

3.2. Long Short-Term Memory network (LSTM)

LSTM is a type of recurrent neural network architecture designed to capture long-range dependencies in sequential data, making them well-suited for tasks such as natural language processing and time series prediction.

The model architecture of LSTM is shown in Fig.4. The structure of embedding layer as well as dense layer and corresponding hyperparameters are the same, the only difference is LSTM Layer. It is a type of recurrent neural network (RNN) architecture that is well-suited for sequence prediction problems. In this code, "LSTM (64)" creates an LSTM layer with 64 memory units. This layer processes sequences of input data and maintains an internal state, allowing it to capture long-term dependencies in the data.

```
# Define the LSTM architecture
model = Sequential([
    Embedding(num_words, 128),
    LSTM(64), # LSTM layer with 64 units
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.1)

# Predictions
y_pred = (model.predict(X_test) > 0.5).astype(int)
```

Fig.4. Define the structure and hyperparameters of the LSTM model

As for the activation function used in this study, ReLU is used in the convolutional layer and second to last dense layer of CNN model. Sigmoid is used in the last dense layer, that is, output layer of CNN, LSTM, GRU. The reason is that in existing literatures, ReLU is commonly used in hidden layers of deep neural networks due to its simplicity, computational efficiency, and effectiveness in preventing the vanishing gradient problem. Moreover, Sigmoid is often used in the output layer of binary classification models to produce probability scores.

3.3. Gated Recurrent Unit (GRU)

GRU is a simpler variant of recurrent neural network designed to capture long-range dependencies in sequential data more efficiently than LSTM, often used as an alternative for tasks like machine translation and sentiment analysis in natural language processing.

It is shown in Fig.5 that "GRU (64)" creates a GRU layer with 64 units. Each unit processes input sequences and maintains its internal state, which allows the network to capture temporal dependencies in the data. In this case, there are 64 units, which means the layer will have 64 internal computational nodes or memory cells. These cells process input data and update their state based on the input and previous states. Almost the same as LSTM model, the GRU architecture contains one embedding layer, one GRU layer and one dense layer, adam optimizer, crossentropy loss function, accuracy metrics, aimed at a simpler structure for less time complexity and easier comparing results.

```
# Define the GRU architecture
model = Sequential([
    Embedding(num_words, 128),
    GRU(64), # GRU layer with 64 units
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1)

# Predictions
y_pred = (model.predict(X_test) > 0.5).astype(int)
```

Fig.5. Define the structure and hyperparameters of the GRU model

3.4. Bidirectional Encoder Representations from Transformers (BERT)

BERT is a transformer-based model known for its ability to pre-train large-scale language representations on vast amounts of text data, achieving state-of-the-art performance on various natural language understanding tasks through fine-tuning.

It is worth noting that, to use BERT, it only needs to load a pre-trained BERT model from the Hugging Face Transformers library rather than define its architecture like traditional neural networks including CNN, LSTM and GRU. The processes of importing libraries, loading IMDB dataset and Padding sequences are almost the same but in order to load pre-trained BERT model we need to import BertTokenizer and TFBertForSequenceClassification from transformers. The figure is shown in Fig.6.

```
import numpy as np
import tensorflow as tf
from transformers import BertTokenizer, TFBertForSequenceClassification
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load the IMDB dataset
num_words = 10000
maxlen = 100
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=num_words)

# Pad sequences to have the same length
X_train = pad_sequences(X_train, maxlen=maxlen)
X_test = pad_sequences(X_test, maxlen=maxlen)
```

Fig.6. Import necessary libraries for the BERT

The process of defining hyperparameters and training the BERT model is shown in Fig.7. It contains eight steps totally.

```
# Initialize BERT tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# Convert string sequences to strings
X_train_strings = [' '.join(map(str, review)) for review in X_train]
X_test_strings = [' '.join(map(str, review)) for review in X_test]

# Tokenize input data
X_train_tokens = tokenizer(X_train_strings, padding=True, truncation=True, max_length=maxlen, return_tensors='tf')
X_test_tokens = tokenizer(X_test_strings, padding=True, truncation=True, max_length=maxlen, return_tensors='tf')

# Convert labels to tensors
y_train = tf.convert_to_tensor(y_train)
y_test = tf.convert_to_tensor(y_test)

# Load pre-trained BERT model for sequence classification
model = TFBertForSequenceClassification.from_pretrained("bert-base-uncased")

# Compile the model
optimizer = 'adam' # Changed from tf.keras.optimizers.Adam to 'adam'
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

# Train the model
history = model.fit(X_train_tokens, y_train, epochs=10, batch_size=32, validation_split=0.1)

# Predictions
y_pred = model.predict(X_test_tokens)[0].argmax(axis=-1)
```

Fig.7. Load pre-trained BERT model and define hyperparameters

In the first line, "BertTokenizer.from_pretrained('bert-base-uncased')" creates a BERT tokenizer object. It loads the

pre-trained tokenizer provided by the transformers library with the "bert-base-uncased" configuration which indicates that loading the tokenizer for the BERT model with an uncased vocabulary. This vocabulary includes only lowercase letters and special tokens. The second step is to convert integer sequences to strings. The reason is that the `imdb.load_data()` function returns sequences of integers representing words while the BERT tokenizer needs strings input. In the third step, "tokenizer" function converts the input strings (`X_train_strings`) and (`X_test_strings`) into tokenized sequences, pads them to a maximum length (`maxlen`), and returns TensorFlow tensors (`return_tensors="tf"`). BERT model expects input in a specific tokenized format, and this step ensures that the input data is properly tokenized and formatted for use with BERT. "tf.convert_to_tensor" function converts Python objects (in this case, NumPy arrays) to TensorFlow tensors since TensorFlow models typically expect input data to be in tensor format. Converting the labels to tensors allows them to be used directly with TensorFlow models without any further conversion steps. The remaining steps and evaluation are similar to those described in CNN part.

3.5. Developed ensemble learning model

In this part, an ensemble model comprised with CNN, LSTM and GRU as base models using stacking technique is established and experimented. The figure is shown in Fig.8. These three base models offer a good balance between complexity and performance. Moreover, they have a much smaller complexity than BERT.

The reason for choosing stacking technique is that stacking combines the predictions of multiple base models using a meta-model, which learns how to best combine these predictions. It allows for flexibility in choosing diverse base models, which can capture different aspects of the data and

improve the overall predictive power. Additionally, stacking provides a framework for effectively leveraging the predictions of multiple models, which can lead to better performance compared to using only a subset of the available models. Last but not least, their predictions are combined using weighted averaging and the model performance is optimized by adjusting the number of epochs, batch size, model architectures and ensemble configurations.

```
# Ensemble Predictions
cnn_pred = cnn_model.predict(X_test)
lstm_pred = lstm_model.predict(X_test)
gru_pred = gru_model.predict(X_test)

# Experiment with different ensemble configurations (e.g., weighted averaging)
ensemble_pred = (0.4 * cnn_pred + 0.3 * lstm_pred + 0.3 * gru_pred)

# Convert predictions to binary labels
ensemble_pred_labels = np.round(ensemble_pred)

# Evaluation
ensemble_accuracy = accuracy_score(y_test, ensemble_pred_labels)
ensemble_precision = precision_score(y_test, ensemble_pred_labels)
ensemble_recall = recall_score(y_test, ensemble_pred_labels)
ensemble_f1 = f1_score(y_test, ensemble_pred_labels)
ensemble_conf_matrix = confusion_matrix(y_test, ensemble_pred_labels)

print("Ensemble Accuracy:", ensemble_accuracy)
print("Ensemble Precision:", ensemble_precision)
print("Ensemble Recall:", ensemble_recall)
print("Ensemble F1 Score:", ensemble_f1)
print("Ensemble Confusion Matrix:")
print(ensemble_conf_matrix)
```

Fig.8. Define the ensemble model using weighted averaging

The process is that, establish CNN, LSTM, GRU models and predict IMDB dataset labels individually to get various probabilities of labels. Then three different probabilities are summed using weighted averaging. In order to do binary classification, "np.round" function is introduced to convert numbers ranged from 0 to 1 to 0 or 1. Generally this process can combine different predictions and lead to a better model performance. The remaining processes are almost the same as single models demonstrated above including CNN, LSTM and BERT.

IV. RESULTS AND ANALYSIS

Based on the model research in Section III, the insights can be integrated gained into Section IV, which used to demonstrate the performance of different models, optimization of the ensemble model and the contribution of the individual base models to the ensemble's performance.

To avoid showing too many similar images, only the evaluation of the ensemble model is showed in Fig.9. The remaining metrics evaluation results are showed in TABLE I according to different epochs and weighted averaging numbers.

It can be seen that among 4 base models, BERT's computational complexity is much greater than others while the accuracy is much lower. It only gains 50% accuracy but is several hundred times longer than others as for average epoch computing time. It proves that it's not suitable for laptop programmer and needs further data pre-processing for higher accuracy. By reason of this characteristic, it is not included in the developed ensemble model in this study. Based on the results, performances of 4 base models are analysed and the contribution of the individual base models (CNN, LSTM, GRU) to the ensemble's performance is analyzed and interpreted as follows.

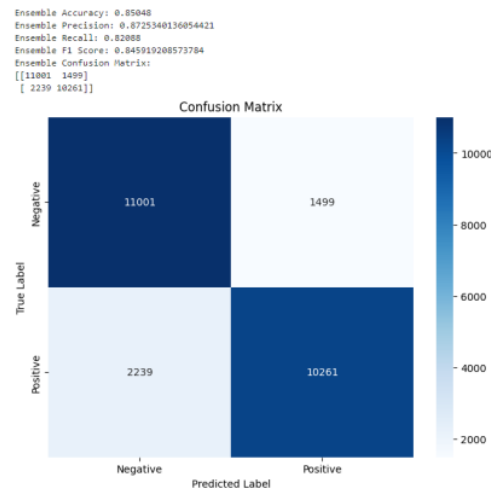


Fig.9. Evaluation of the ensemble model

TABLE I

ESTABLISHED METRICS RESULTS OF DEEP LEARNING MODELS.

Model/Metrics	Epoch	Weighted averaging	Accuracy	Precision	Recall	F1 Score	Average epoch computing time
CNN	5	/	0.84568	0.84701	0.84376	0.84538	9.0s
CNN	10	/	0.84868	0.84600	0.85256	0.84926	10.2s
LSTM	5	/	0.83628	0.83364	0.84024	0.83693	25.0s
LSTM	10	/	0.83052	0.81579	0.85384	0.83438	25.2s
GRU	5	/	0.83232	0.80549	0.87624	0.83937	28.0s
GRU	10	/	0.83108	0.82254	0.84432	0.83329	28.4s
BERT	3	/	0.50000	0.25000	0.50000	0.33333	9383.7s
BERT	10	/	0.50000	0.25000	0.50000	0.33333	9527.6s
Ensemble	3	2/5, 3/10, 3/10	0.85884	0.85551	0.86352	0.85950	67.4s
Ensemble	3	1/3, 1/3, 1/3	0.85876	0.85538	0.86352	0.85943	69.1s
Ensemble	3	1/2, 1/4, 1/4	0.85448	0.83077	0.89032	0.85951	65.7s
Ensemble	5	2/5, 3/10, 3/10	0.85048	0.87253	0.82088	0.84592	67.8s
Ensemble	5	1/3, 1/3, 1/3	0.85440	0.84843	0.86296	0.85564	64.7s
Ensemble	5	1/2, 1/4, 1/4	0.85356	0.86282	0.84080	0.85167	65.1s

CNN achieved an accuracy of approximately 84.6% after 5 epochs, which improved slightly to 84.9% after 10 epochs. It showed consistent improvement in precision, recall, and F1 score with more epochs. The average epoch computing time for the CNN model was approximately 9.6 seconds.

LSTM achieved an accuracy of approximately 83.6% after 5 epochs, which decreased slightly to 83.1% after 10 epochs. Precision and F1 score remained relatively stable, but there was a slight decrease in recall after 10 epochs. The LSTM model had a longer average epoch computing time compared to CNN, approximately 25.1 seconds.

GRU achieved an accuracy of approximately 83.2% after 5 epochs, which remained stable around 83.1% after 10 epochs. It showed slight improvement in precision and F1 score after 10 epochs, but there was a decrease in recall. The average epoch computing time for the GRU model was approximately 28.2 seconds, longer than both CNN and LSTM.

BERT: The results for the BERT model after both 3 and 10 epochs show an accuracy of 50.0%, indicating poor performance on the sentiment analysis task. The precision, recall, and F1 score are also low, with all metrics at 25.0% and 50.0% respectively. Moreover, the confusion matrix is [0 12500; 0 12500] which reveals that the model predicted all instances as belonging to the positive sentiment class, resulting in a lack of true negative predictions and a high number of false positives.

Ensemble Model: The ensemble model, consisting of CNN, LSTM, and GRU, achieved the highest accuracy among all models, with an accuracy of approximately 85.9%. Precision, recall, and F1 score were also high, indicating balanced performance in correctly classifying both positive and negative instances. This ensemble model demonstrates the effectiveness of combining predictions from diverse deep learning architectures. The average epoch computing time for the ensemble model was approximately 66.4 seconds, which is higher compared to individual models but still reasonable.

The ensemble model, comprising CNN, LSTM, and GRU, outperformed individual base models, underscoring the benefits of combining diverse architectures. CNN, LSTM, and GRU models showed relatively similar performance. BERT exhibited significantly lower performance compared to other models. The choice of weighted averaging scheme in the ensemble model influenced its performance, with different schemes leading to slightly different results.

V. CONCLUSIONS

In this study, we explored the performance of CNN, LSTM, GRU, and BERT models, as well as an ensemble model comprising CNN, LSTM, GRU, on the IMDB dataset for sentiment analysis. The experimental results indicate that while individual models show competitive performance, the ensemble model significantly outperforms them, achieving an accuracy of approximately 85.9%. This highlights the importance of leveraging ensemble techniques to combine predictions from diverse models, leading to improved classification performance. Furthermore, the stacked ensemble model, utilizing weighted averaging technique, demonstrates the effectiveness of integrating multiple deep learning architectures for sentiment analysis tasks. Overall, this study underscores the potential of ensemble methods in enhancing the performance of deep learning models for natural language processing tasks like sentiment analysis.

ACKNOWLEDGEMENT

The author sincerely thanks Seena Joseph who teaches Artificial Intelligence at the University of Wales Trinity Saint David, Swansea, Wales, United Kingdom, for providing high quality lectures and Q&A services.

VIDEO LINK

<https://youtu.be/CoJw5IQMHY>

REFERENCES

- [1] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 2, pp. 757–774, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.014.
- [2] R. Kora and A. Mohammed, "An enhanced approach for sentiment analysis based on meta-ensemble deep learning," *Soc. Netw. Anal. Min.*, vol. 13, no. 1, p. 38, Mar. 2023, doi: 10.1007/s13278-023-01043-6.
- [3] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. [Online]. Available: <http://www.aclweb.org/anthology/P11-1015>
- [4] I. Kanwal, F. Wahid, S. Ali, A. -U. Rehman, A. Alkhayyat, and A. Al-Radaei, "Sentiment Analysis Using Hybrid Model of Stacked Auto-Encoder-Based Feature Extraction and Long Short Term Memory-Based Classification Approach," *IEEE Access*, vol. 11, pp. 124181–124197, 2023, doi: 10.1109/ACCESS.2023.3313189.
- [5] M. Sen and K. Chaturvedi, "Sentiment Analysis of IMDB Movies Dataset Using Deep Learning Based GRU Model," in *2023 International Conference on Sustainable Communication Networks and Application (ICSCNA)*, Nov. 2023, pp. 176–181. doi: 10.1109/ICSCNA58489.2023.10370081.
- [6] K. Arora, N. Gupta, and S. Pathak, "Sentimental Analysis on IMDb Movies Review using BERT," in *2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, Jul. 2023, pp. 866–871. doi: 10.1109/ICESC57686.2023.10193688.
- [7] S. Gogineni and A. Pimpalshende, "Predicting IMDB Movie Rating Using Deep Learning," in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, Jun. 2020, pp. 1139–1144. doi: 10.1109/ICCES48766.2020.9137994.
- [8] M. M. Rahman, M. M. Jahangir Kabir, M. Kabir, and M. R. Karim, "Categorization of Product Feedback using a Deep CNN-based LSTM Model and a Comparative Study of Traditional ML Algorithms," in *2022 IEEE 7th International conference for Convergence in Technology (I2CT)*, Apr. 2022, pp. 1–6. doi: 10.1109/I2CT54291.2022.9825165.
- [9] A. Yenter and A. Verma, "Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, Oct. 2017, pp. 540–546. doi: 10.1109/UEMCON.2017.8249013.
- [10] M. M. Krishna, B. Duraisamy, and J. Vankara, "Hybrid Deep Learning Techniques for Sentiment Analysis on IMDb Datasets," in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Apr. 2022, pp. 689–693. doi: 10.1109/ICACITE53722.2022.9823898.
- [11] A. Rasool, Q. Jiang, Q. Qu, and C. Ji, "WRS: A Novel Word-embedding Method for Real-time Sentiment with Integrated LSTM-CNN Model," in *2021 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Jul. 2021, pp. 590–595. doi: 10.1109/RCAR52367.2021.9517671.
- [12] L. Zhang, M. Wang, M. Liu, and H. Li, "Sentiment Analysis of Movie Reviews Based on LSTM-Adaboost," in *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Dec. 2022, pp. 186–190. doi: 10.1109/IMCEC55388.2022.10019969.
- [13] Y. Cheng, L. Yao, G. Xiang, G. Zhang, T. Tang, and L. Zhong, "Text Sentiment Orientation Analysis Based on Multi-Channel CNN and Bidirectional GRU With Attention Mechanism," *IEEE Access*, vol. 8, pp. 134964–134975, 2020, doi: 10.1109/ACCESS.2020.3005823.
- [14] A. Zouzou and I. E. Azami, "Text sentiment analysis with CNN & GRU model using GloVe," in *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*, Oct. 2021, pp. 1–5. doi: 10.1109/ICDS53782.2021.9626715.
- [15] C. Vielma, A. Verma, and D. Bein, "Sentiment Analysis with Novel GRU based Deep Learning Networks," in *2023 IEEE World AI IoT Congress (AIoT)*, Jun. 2023, pp. 0440–0446. doi: 10.1109/AIIoT58121.2023.10174396.
- [16] L. B. N. A. R. K. V. K. K. A. Appaji, P. D. Shenoy, and V. K. R., "Sentiment Analysis of Textual Data using Word Embedding and Deep Learning Approaches," in *2023 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*, Nov. 2023, pp. 1–6. doi: 10.1109/NKCon59507.2023.10396439.
- [17] K. Mouthami, N. Yuvaraj, K. K. Thilageswaran, and K. J. Lokeshvar, "Text Sentiment Analysis of Film Reviews Using Bi-LSTM and GRU," in *2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, Jul. 2023, pp. 1379–1386. doi: 10.1109/ICESC57686.2023.10193121.
- [18] R. I. Rudra and A. K. Gopalakrishnan, "Sentiment Analysis of Consumer Reviews Using Machine Learning Approach," in *2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Jul. 2023, pp. 49–54. doi: 10.1109/JCSSE58229.2023.10202106.
- [19] M. Mishra and A. Patil, "Sentiment Prediction of IMDb Movie Reviews Using CNN-LSTM Approach," in *2023 International Conference on Control, Communication and Computing (ICCC)*, May 2023, pp. 1–6. doi: 10.1109/ICCC57789.2023.10165155.
- [20] M. R. Haque, S. Akter Lima, and S. Z. Mishu, "Performance Analysis of Different Neural Networks for Sentiment Analysis on IMDb Movie Reviews," in *2019 3rd International Conference on Electrical,*

Computer & Telecommunication Engineering
(ICECTE), Dec. 2019, pp. 161–164. doi:
10.1109/ICECTE48615.2019.9303573.

- [21] M. Qamar, H. Rao, S. A. Farooq, and A. Swagat Bhuyan, “Sentiment Analysis using Deep Learning: A Domain Independent Approach,” in *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)*, Mar. 2023, pp. 1373–1380. doi: 10.1109/ICEARS56392.2023.10085676.
- [22] Á. Kovács and T. Tajti, “Enhancing Sentiment Analysis Accuracy on IMDB Reviews Through Ensemble Machine Learning Techniques,” in *2023 IEEE 21st Jubilee International Symposium on Intelligent Systems and Informatics (SISY)*, Sep. 2023, pp. 000289–000294. doi: 10.1109/SISY60376.2023.10417873.
- [23] Á. Kovács and T. Tajti, “Enhancing Sentiment Analysis Accuracy on IMDB Reviews Through Ensemble Machine Learning Techniques,” in *2023 IEEE 21st Jubilee International Symposium on Intelligent Systems and Informatics (SISY)*, Sep. 2023, pp. 000289–000294. doi: 10.1109/SISY60376.2023.10417873.
- [24] K. J. Sai, S. S. Chakravarthi, S. Sountharajan, and E. Suganya, “Ensemble Learning Solution for the Aspect-based Sentimental Analysis on IMDB reviews,” in *2023 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, Mar. 2023, pp. 01–08. doi: 10.1109/WiSPNET57748.2023.10134144.
- [25] W. M. D. R. Ruwantha, K. Banujan, and K. Btgs, “LSTM and Ensemble Based Approach for Predicting the Success of Movies Using Metadata and Social Media,” in *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, Sep. 2021, pp. 626–630. doi: 10.1109/3ICT53449.2021.9581601.
- [26] A. Ö. Eren and M. Sert, “Movie rating prediction using ensemble learning and mixed type attributes,” in *2017 25th Signal Processing and Communications Applications Conference (SIU)*, May 2017, pp. 1–4. doi: 10.1109/SIU.2017.7960604.
- [27] K. L. Tan, C. P. Lee, K. M. Lim, and K. S. M. Anbananthen, “Sentiment Analysis With Ensemble Hybrid Deep Learning Model,” *IEEE Access*, vol. 10, pp. 103694–103704, 2022. doi: 10.1109/ACCESS.2022.3210182.

Final

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Well done!

94/100

PAGE 1

PAGE 2

PAGE 3



Comment 1

Simply, Related work or Review of Literature.



Comment 2

Do you need to include this as the first sentence for review of literature.



Comment 3

Literature support ?



Comment 4

Literature support ?



Comment 5

Don't provide the initial



Comment 6

The research gap should be highlighted before state the aim of the study.



Comment 7

check the column template....



Comment 8

Need to rephrase, for instance, in this section, the methods and materials used to implement the four deep learning models including.....and ensemble models for sentiment analysis is presented.



Comment 9

Literature support

GRADING FORM: ACCA 7026 GRADING

WENHAO ZHONG

93.50

ENGAGEMENT WITH LIT. SKILLS

(including current research, referencing, academic conventions and academic honesty)



Exceptional engagement with an extensive range of relevant and credible literature, informed by the latest research. Minor errors in in-text citation.

14

KNOWLEDGE&UNDERSTANDING SKILLS

(A systematic, conceptual understanding of knowledge, and a critical awareness of current problems and/or new insights, much of which is at, or informed by, the forefront of the discipline; a comprehensive understanding of techniques applicable to their own research)



Excellent understanding of the concepts.

8.5

COGNITIVE&INTELLECTUAL SKILLS

(Critical evaluation of current research and methodologies and develop critiques of them and, where appropriate, to propose new hypotheses; make sound judgements in the absence of complete data.)



Excellent critical evaluation

24

PRACTICAL APPLICATION SKILLS

(Originality / creativity in the application of knowledge, tools and techniques and in tackling and solving problems in complex and unpredictable professional situations; practical use of established techniques of research and enquiry to create and interpret knowledge in the discipline.)



An advanced application of knowledge, methods, materials, tools and/or techniques. Results

38

presentation should have been improved by comparing and contrasting with the existing literature.

TRANSF. SKILL & PROF.PRACTICE

(Exercise of self-direction, autonomy and personal responsibility; plan and implement tasks at a professional level; independent learning; use appropriate media to communicate effectively and professionally to a variety of audiences; fluency of expression; systematic approach; clarity and effectiveness in presentation and organisation.)



A well written abstract.

Can communicate with an exceptionally high level of professionalism. Minor correction in in-text citation.

9