VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Natural Language Processing (CO3085)

Comparing Deep Learning Models For Sentiment Analysis In Customer Reviews

Advisor: Assoc. Prof. Quan Thanh Tho
Student: Nguyen Phan Tri Duc - 2152528
Semester: 231

HO CHI MINH CITY, NOVEMBER 2023

Contents

1 Introduction

Sentiment analysis is a field in natural language processing (NLP) that seeks to understand and interpret the sentiments, opinions, and emotions expressed in textual data. In an era dominated by massive amounts of digital information, sentiment analysis has emerged as a crucial tool for extracting valuable insights from the opinions shared on various platforms, such as social media, product reviews, news articles, and customer feedback.

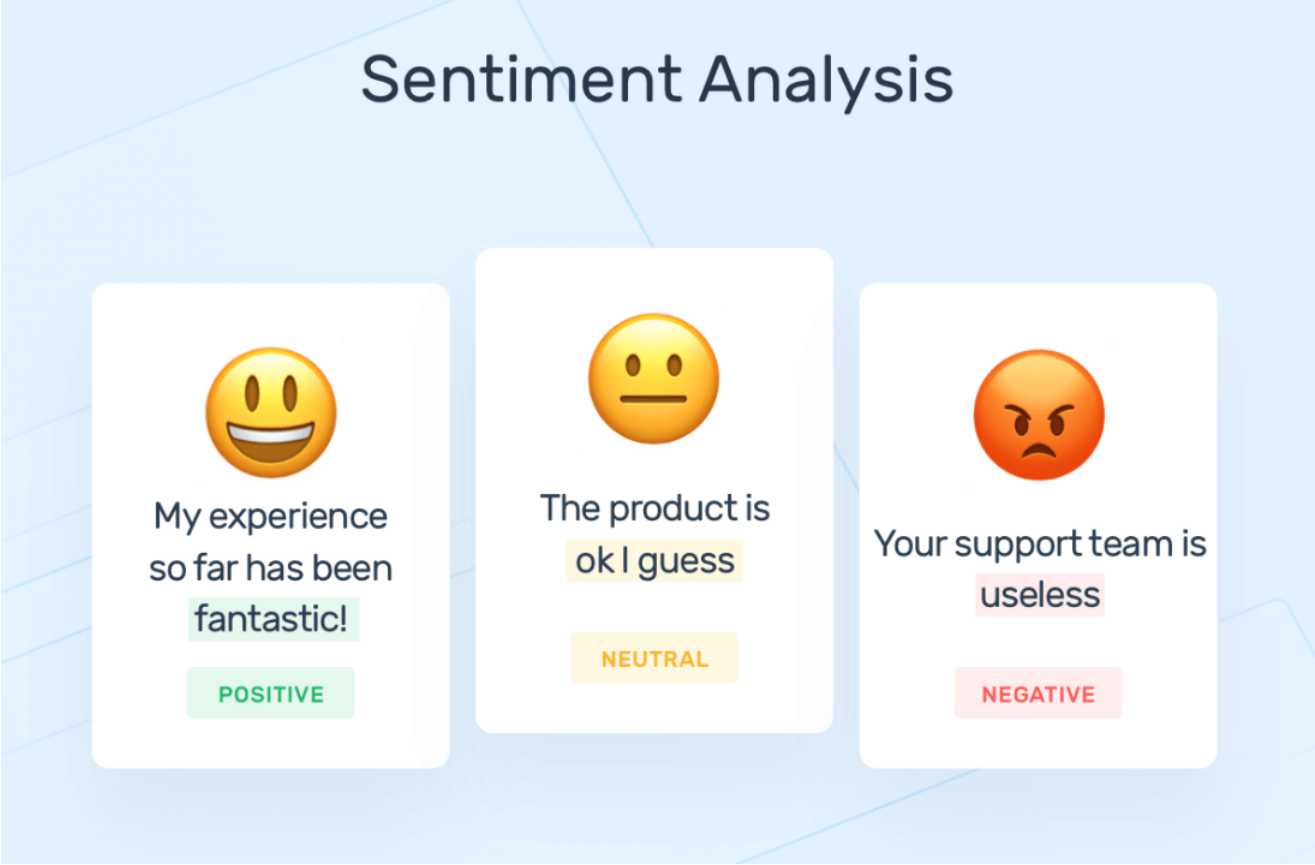

Figure 1: Example of diverse sentiments

The primary objective of sentiment analysis is to classify and analyze the subjective information present in text, categorizing it into predefined sentiment categories such as positive, negative, or neutral. This classification enables businesses, researchers, and individuals to gain a deeper understanding of public perceptions, customer sentiments, and trends within different domains.

One of the key challenges in sentiment analysis lies in the inherent complexity of human language. Language is rich, context-dependent, and often filled with nuances, making it challenging to accurately interpret the sentiment behind a piece of text. Ambiguities, variations in expression, and the use of sarcasm or irony further contribute to the intricacies involved in sentiment analysis.

The field of sentiment analysis encompasses a range of techniques, from traditional machine learning approaches to more advanced deep learning methods. Its role in distilling actionable insights from the vast sea of textual data makes sentiment analysis an invaluable tool for shaping business strategies, enhancing customer experiences, and understanding the ever-changing pulse of public sentiment.

2 Dataset

The dataset is from VLSP, comprises customer reviews in Vietnamese language on various tech products, categorized into three distinct classes: negative, neutral, and positive sentiments. There are **5100** records in training data and **1050** records in testing data.

```
[8] data_train = pd.read_csv("vlsp_sentiment_train.csv", sep='\t')
    data_test = pd.read_csv("vlsp_sentiment_test.csv", sep='\t')
    data_train
```

1 to 25 of 5100 entries Filter

index	Class	Data
0	-1	Mình đã dùng anywhere thế hệ đầu, quả là đầy thất vọng, hiện tại đang vứt xó. Giá thì đắt, ngốn pin như ăn gỏi, nặng
1	-1	Quan tâm nhất là độ trễ có cao không, dùng thi thoảng nó cứ trễ bực mình, đấy mới chỉ là dùng văn phòng chứ game thì chắc là ném đi từ lâu. Không biết con này có độ trễ không nhỉ. Dùng nhiều loại nhưng vẫn kết nhất con chuột sứ mitsumi, gọn bấm này tốt
2	-1	dag xài con cùi bắp trâu, mỗi tội đánh liên minh ức chế đập hết con
3	-1	logitech chắc hàng phải tiền triệu trở lên dùng mới thích chứ em dùng con có thấy được vài bữa là hư. chuyển sang eblue dùng được hơn năm chưa thấy hỏng.
4	-1	Đang xài con cùi mía , nhà xài nhiều chuột nên thử con này con kia chơi và kết quả là sau tháng và con chuột khác nhau đủ kiểu đa chức năng mà h chỉ còn lại là con cùi mía còn xài dc h đi mua chuột là xác định ko để mấy con nhân viên vấy ngắn nó dụ nửa cứ logitech mà fan
5	-1	Đang xài con nút chuột giữa hai con đều như hạch. xài chưa được năm , scroll đã chạy bậy rồi. Hay mua phải đồ đểu hĩ?
6	-1	Con Anywhere mình dùng bị double click cũng lần rồi... chán lắm
7	-1	Hàng cty cấp, cấp xong vứt ở nhà, xài con Xornet cho đỡ mỏi tay. Cơ mà vẫn thấc mắc là dong thấp nhất, nhưng éo bằng con misumi giá bằng
8	-1	Magic mouse mà ngon hơn mới lạ, Magic mouse chỉ được cái đẹp, pin thì ăn tuần hết, ko di trên mọi bề mặt được bạn nhé, bảo hành thì hãng bảo hành năm, còn Logitech thì năm nhé ## chuột này em đang dùng, chuột magic mouse bỏ tủ
9	-1	em giống y bác luôn, chán bluetooth kinh hồn, mà em không tưởng tượng được độ chuối của bluetooth
10	-1	Mình cũng giật kinh khủng, cảm thấy hơi nản về cái bluetooth của nó..... đã cập nhật El Capitan mới nhất, lúc đầu ko bị, thời gian sau lại giật lại như thường
11	-1	Mình đã lên nhưng vẫn bị, đúng là tắt wifi đi thì ngon hơn hẳn..... Chán vãi, mua em noá để khỏi dùng USB receiver, mà thấy cái bluetooth nản vãi
12	-1	mac mà ..... cảm thấy chuột ko mượt (khá là khó chịu
13	-1	di chuyển chuột ko mượt bạn ơi. Còn cáo Scroll cũng thấy bt nhưng chả biết sao cảm giác ko thích lắm. Mới đầu test train win thì ok nhưng giờ về nhà cắm mac the xảy ra mấy chi khi chịu thật.
14	-1	Cập Nhật tình hình sau khi đi bảo hành về bên logitech thông báo chuột bình thường khi kết nối bluetooth với mac air. Mình mang mac pro ra ngoài svhouse kết nối vs bluetooth con mv anywerè thì vẫn có hiện tượng lag ..... giờ mang về xài tạm chứ chả biết giải quyết sao luôn. Tính làm cái mail gửi svhouse để hỏi thúc bên logitech mà chả biết thế nào..... mấy bạn xài mac pro kết nối bluetooth thì bt à??? Chắc kiếm chuột khác cắm test thử xem sao quá.
15	-1	có ai mua con này dùng lúc rê nó bị đứt quãng không, e dùng thấy bị khó chịu quá, rê nó không được đều.
16	-1	Vừa đi bảo hành con này nút middle. Chuột logitech cao cấp hình như không tốt. Trước đây xài con cũng bị lỗi double click ko sửa được. Cũng may, Thành Nhân đổi cho con mới. Giờ đến con này thì cũng bị lỗi. Cho do mình xui
17	-1	chưa có cách khắc phục nhé mang đem đi bảo hành rồi mà vẫn bị chả hiểu sao logitech kêu xài bt ..... thôi về cắm cái cục kia vô xài luôn cho lành. đồ rẻ rẻ thì ok đụng vô đát tí thì bị tè le
18	-1	Sao mình dùng nó lag vãi linh hồn luôn, chán thật bluetooth mà giờ đang dùng usb reciver. Mình dùng mac pro ích, có cách nào khắc phục không chứ bỏ tiền ra mua còn này để dùng bluetooth mà giờ xót quá

Figure 2: Training data

```
data_train = pd.read_csv("vlsp_sentiment_train.csv", sep='\t')
data_test = pd.read_csv("vlsp_sentiment_test.csv", sep='\t')
data_test
```

1 to 25 of 1050 entries Filter

index	Class	Data
0	-1	Nói thiệt là mình thì thì chuột nào mình cũng chơi tuốt, chỉ trừ 1 hãng ra: Razer. Mình đang sở hữu 1 con DA black, xài được 6 tháng nó bị double click, đem sửa xong xài được them 2 tháng nữa nó bị hư nút cuộn... Trong khi con SS Sensei mình xài 3 năm mới bị double click và rít nút cuộn.
1	-1	Đang dùng mx1. Cũng ngon nhưng chưa đầy năm mà đã 3 lần tháo ra thay 3 nút bấm rồi. May còn lột được từ máy khác mà bỏ vào!
2	-1	Chưa thấy đc điểm thuyết phục để mua, nhất là vụ pin
3	-1	Những phần xem báo tra cứu bản đồ, dịch vụ.. dùng ip cho lành. Màn hình bé tí thế kia xem chắc nổi nóng
4	-1	ĐÚNG LÀ MUA Ở VIỆT NAM KHÔNG ỨNG DỤNG ĐƯỢC GÌ NHIỀU. NGOÀI MÁY CÁI NOTIFICATION RA
5	-1	Thế thì bạn sẽ dễ hiểu sai lắm bạn ạ. Vì bọn báo chí nó giật tít lắm lúc chả phải cái nó viết. Với cả. Dù là xem đầu đề bài báo thì cũng là mệt với cái màn hình tí hon đó. Mình đã đi coi aw rồi nên mới thấy ko khả thi.
6	-1	phạm vi của bluetooth rất ngắn chưa kể nếu che chắn quá kĩ như cốp xe là nghe tiếng được tiếng mất trừ khi bác luôn kè kè cái đt bên mình thì chất lượng cuộc gọi mới tốt được, đó là chưa kể đến nếu dùng đt cũ hoặc sw của tàu thì bluetooth chỉ ở v3 hạn hữu có cả v2 thì chất lượng cuộc gọi còn tệ nữa, còn nếu ngoài phạm vi bt thì nó lại là vấn đề của 1 cái sim khác nói chung thì dù cho sw có sim hay không sim gì thì bác vẫn kè kè cái smp bên người thôi
7	-1	Dầy hự
8	-1	Sao trông dầy thế nhỉ
9	-1	- Chết quên... cho phát biểu lại Apple watch là best. - À lại quên. LG LTE 2 là con đẻ tiện thứ 2 khi ăn cắp thiết kế của các hãng tổng hợp nhưng ít ra nhìn nó cũng ít đẻ tiện hơn nhà sam. ... còn muốn con nào best chịu khó tổng hợp vào. Chứ ra lấy xe ở p
10	-1	Rất đẹp, nhưng phù hợp với đeo cổ chân người Việt hơn.
11	-1	Giá mắc quá. 330 đồng có quá nhiều lựa chọn
12	-1	Mình mới tậu đc 1 e.Thử chức năng tự mở khóa mà thất vọng quá (mình khoái nhất cái này).Trước nó cho phép mở khóa luôn nhưng h chỉ cho duy trì trạng thái mở khóa thôi, nghĩa là vẫn phải đăng nhập bằng tay trước. Đồng thời app phải cho chạy ngầm... Nói thật nghi ông Mi này thu thập dữ liệu người dùng lắm
13	-1	Vọc, restart máy... tốn bao thời gian, bực mình xoá Mi Fit đi cài lại thì OK, thật là éc.
14	-1	Suốt ngày dùng mấy vòng đeo này thì đừng hỏi sao vô sinh, ung thư, bệnh tật
15	-1	Nhầm rồi bạn. SW của Huawei nhìn đẹp hơn nhiều
16	-1	vỏ nhựa thì thua rồi, tưởng Asus thế nào thời buổi này vẫn còn kiểu build kém tiết kiệm giảm chất lượng
17	-1	ha ha bạn mua đi. SS cứ hết bảo hành là có bệnh :v
18	-1	. Hình Batman của Samsung xấu quá, không bằng phiên bản Nokia 710 Baman. Hình Batman Nokia 710 đẹp hơn. Tôi đang sử dụng nè. Có thể nói Nokia 710 phiên bản Batman chắc chỉ có 1 cái duy nhất của tôi tại VN
19	-1	Quá bình thường so với giá trị của nó, Tính ra SS6 vẫn đẹp hơn.
20	-1	ai cũng chê iphone nhưng cuối cùng cũng SS bắt trước iphone vè màu sắc, thiết kế jack 3.5 phía dưới, loa có lỗ nhỏ, đường viền dải anten

Figure 3: Testing data

Data pre-processing:

- Encode labels to one-hot vectors.
  - [1, 0, 0] for negative.
  - [0, 1, 0] for neutral.
  - [0, 0, 1] for positive.

```
labels = data_train.iloc[:, 0].values
labels_test = data_test.iloc[:, 0].values

encoded_labels = np.array(
    [
            [1, 0, 0] if label == -1
        else [0, 1, 0] if label == 0
        else [0, 0, 1]
             for label in labels
    ]
)

encoded_labels_test = np.array(
    [
            [1, 0, 0] if label == -1
        else [0, 1, 0] if label == 0
        else [0, 0, 1]
             for label in labels_test
    ]
)

labels = encoded_labels
labels_test = encoded_labels_test
```

- **Remove words with digits in each reviews:** Word with digits is usually unique(product names, prices...) and hard to trained to actually contribute to the sentiment analysis.

Word with digits is usually unique and hard to trained to actually contribute to the sentiment analysis.

```
[ ] # Function to print words with digits
    def printWordWithDigits(review) :
        for word in review.split(' '):
            for char in word:
                if char.isdigit():
                    print(word, end=' ')
                    break
        return review

    data_train['Data'] = data_train['Data'].apply(printWordWithDigits)
    data_test['Data'] = data_test['Data'].apply(printWordWithDigits)
```

```
98k....pin 2 400k m175 3 4 m175 :3 2 M185, 1 3 1/3. 2 1 1tr7 1 10.11.3 10.11.3 13" 2015 2 10.11.3 M905 1tr8
```

Figure 4: Testing data

Apply the remove word with digits function to the 'Data' column.

```python
# Function to detect word with digits
def contains_digits(word) :
    for char in word:
        if char.isdigit():
            return True
    return False

# Function to remove word with digits
remove_word_with_digits = lambda review: ' '.join(
    word for word in review.split(' ') if not contains_digits(
        word)
)

# Apply the function to the 'Data' column
data_train['Data'] = data_train['Data'].apply(
    remove_word_with_digits)
data_test['Data'] = data_test['Data'].apply(
    remove_word_with_digits)

reviews = data_train.iloc[:, 1].values
reviews_test = data_test.iloc[:, 1].values
```

	Class	Data
0	-1	Mình đã dùng anywhere thế hệ đầu, quả là đầy t...
1	-1	Quan tâm nhất là độ trễ có cao không, dùng thi...
2	-1	dag xài con cùi bắp 98k....pin trâu, mỗi tội đ...
3	-1	logitech chắc hàng phải tiền triệu trở lên dùn...
4	-1	Đang xài con m175 cùi mía , nhà xài nhiều chuộ...
...	...	...
5095	0	Mình mua máy về đc 1 ngày mà điện thoại khác g...
5096	0	Có bạn nào dùng f1w ko.mình dùng m cảm thấy qu...
5097	0	Dùng oppo mà bộ nhớ 4gb thì k chơi games ...
5098	0	Sao tui thích xài hàng oppo mà lựa toàn mấy đứ...
5099	0	mới mở hộp ,oy mở vào camera mà đã có ảnh chụp...

5100 rows × 2 columns

(a) Training data before apply

	Class	Data
0	-1	Mình đã dùng anywhere thế hệ đầu, quả là đầy t...
1	-1	Quan tâm nhất là độ trễ có cao không, dùng thi...
2	-1	dag xài con cùi bắp trâu, mỗi tội đánh liên mi...
3	-1	logitech chắc hàng phải tiền triệu trở lên dùn...
4	-1	Đang xài con cùi mía , nhà xài nhiều chuột nên...
...	...	...
5095	0	Mình mua máy về đc ngày mà điện thoại khác gọi...
5096	0	Có bạn nào dùng ko.mình dùng m cảm thấy quanh ...
5097	0	Dùng oppo mà bộ nhớ thì k chơi games đc đ...
5098	0	Sao tui thích xài hàng oppo mà lựa toàn mấy đứ...
5099	0	mới mở hộp ,oy mở vào camera mà đã có ảnh chụp...

5100 rows × 2 columns

(b) Testing data after apply

Figure 5: Comparison of training and testing data

- **Tokenization:** Using pyvi module to tokenize each reviews.

```
# Use PyVi for Vietnamese word tokenizer
word_reviews = []
word_reviews_test = []

for review in reviews:
    review = ViTokenizer.tokenize(review.lower())
    word_reviews.append(review.split())

for review in reviews_test:
    review = ViTokenizer.tokenize(review.lower())
    word_reviews_test.append(review.split())
```

- **Embedding Dimensions and Sequence Length:**
  - EMBEDDING_DIM: Specifies the dimensionality of the word vectors.
  - MAX_VOCAB_SIZE: Defines the maximum number of unique words to be used in the word embedding.
  - MAX_SEQUENCE_LENGTH: Sets the maximum number of words in a review.

The Tokenizer class from Keras is used to vectorize the text corpus, turning each word into a numerical sequence. First I fit the tokenizer on the training data (word_reviews), building the vocabulary, then convert Texts to Sequences and Padding, and converts the tokenized training data into sequences of integers. Pads the sequences to ensure uniform length (MAX_SEQUENCE_LENGTH).

Similar steps are repeated for the testing data.

```
EMBEDDING_DIM = 400 # how big is each word vector
MAX_VOCAB_SIZE = 10000 # how many unique words to use (i.e num
    rows in embedding vector)
MAX_SEQUENCE_LENGTH = 300 # max number of words in a comment
   to use

tokenizer = Tokenizer(num_words=MAX_VOCAB_SIZE, lower=True,
   char_level=False)
tokenizer.fit_on_texts(word_reviews)

sequences_train = tokenizer.texts_to_sequences(word_reviews)
data = pad_sequences(sequences_train, maxlen=
   MAX_SEQUENCE_LENGTH)

word_index = tokenizer.word_index

sequences_test = tokenizer.texts_to_sequences(
   word_reviews_test)
data_test = pad_sequences(sequences_test, maxlen=
   MAX_SEQUENCE_LENGTH)
```

```
[ ]  print('Shape of X train and X validation tensor:', data.shape)
     print('Shape of label train and validation tensor:', labels.shape)

     print('Shape of X test and X validation tensor:', data_test.shape)
     print('Shape of label test and validation tensor:', labels_test.shape)
```

```
Shape of X train and X validation tensor: (5100, 300)
Shape of label train and validation tensor: (5100, 3)
Shape of X test and X validation tensor: (1050, 300)
Shape of label test and validation tensor: (1050, 3)
```

Figure 6: Shape of training and testing data

- **Embedding matrix:** I utilize pre-trained word vectors from the 'vi-model-CBOW.bin' file, load them into 'word_vectors'. An 'embedding_matrix' is initialized as a matrix of zeros with dimensions corresponding to the determined vocabulary size and the chosen embedding dimension ('EMBEDDING_DIM'). The matrix is populated by the corresponding vector, otherwise a random vector. The final step involves creating an Embedding layer in Keras ('embedding_layer') with the determined vocabulary size, embedding dimension, and weights set to the constructed embedding matrix.

```
word_vectors = KeyedVectors.load_word2vec_format('vi-model-
    CBOW.bin', binary=True)

vocabulary_size=min(len(word_index)+1,MAX_VOCAB_SIZE)
embedding_matrix = np.zeros((vocabulary_size, EMBEDDING_DIM))
for word, i in word_index.items():
    if i>=MAX_VOCAB_SIZE:
        continue
    try:
        embedding_vector = word_vectors[word]
        embedding_matrix[i] = embedding_vector
    except KeyError:
        embedding_matrix[i]=np.random.normal(0,np.sqrt(0.25),
            EMBEDDING_DIM)

del(word_vectors)

from keras.layers import Embedding
embedding_layer = Embedding(vocabulary_size,
                            EMBEDDING_DIM,
                            weights=[embedding_matrix],
                            trainable=True)
```

3 Deep Learning Models

3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep neural networks designed for processing and analyzing visual data, they are particularly well-suited for tasks such as classification, object detection, and facial recognition, where the spatial relationships and patterns within images play a crucial role. The key innovation of CNNs lies in their ability to automatically learn hierarchical and spatial feature representations from the input data.

One of the strengths of CNNs is their parameter sharing, meaning that the same filter is applied to different parts of the input, enabling the network to learn translation-invariant features. This property makes CNNs highly effective in recognizing patterns and objects in various positions and orientations within an image.

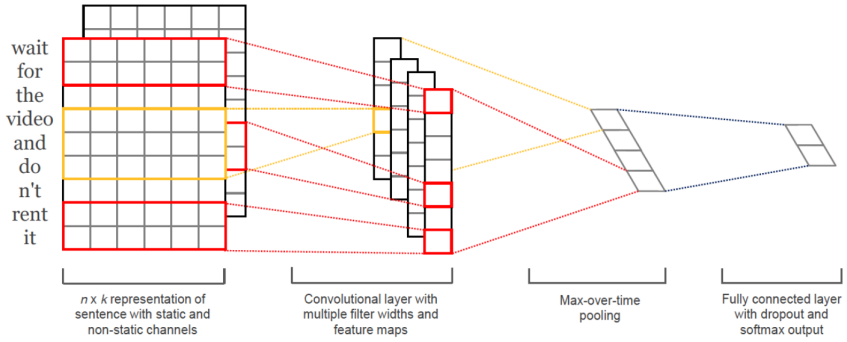

Figure 7: CNN model visualized in NLP

CNNs employ convolutional layers that apply convolutional operations to input images, extracting local features through the use of filters. These filters detect specific patterns like edges, textures, or more complex structures within the input data. The use of pooling layers further reduces the spatial dimensions of the data, emphasizing the most important features and enhancing the network's ability to generalize.

CNNs have achieved remarkable success in a wide range of applications, including image recognition, object detection, and even tasks beyond computer vision, such as natural language processing. Their ability to automatically learn hierarchical features has positioned CNNs as a foundational architecture in the field of deep learning, playing a pivotal role in advancing the capabilities of artificial intelligence systems.

3.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) represent a class of artificial neural networks designed for sequential data processing, making them particularly effective for tasks involving patterns across time or sequences. Unlike traditional feedforward neural networks, RNNs maintain hidden states that allow them to retain information from previous inputs, enabling them to capture dependencies and relationships within sequential data. This architecture makes RNNs well-suited for various applications, including natural language processing, speech recognition, and time series analysis. However, standard RNNs have limitations in capturing long-term dependencies, known as the vanishing and exploding gradient problems, which can hinder their performance on tasks requiring memory over extended sequences.

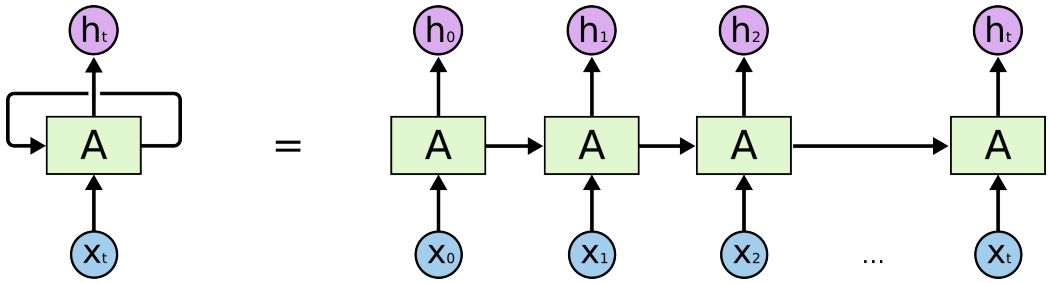

Figure 8: RNN simple structure

Long Short-Term Memory (LSTM) networks emerged as a solution to the challenges posed by standard RNNs. LSTMs are a type of RNN that incorporates memory cells and gating mechanisms to better manage and control the flow of information. The key innovation of LSTMs lies in their ability to selectively learn, remember, and forget information over long sequences. The architecture includes three gates - input, forget, and output gates - which regulate the flow of information through the network. LSTMs are particularly effective for tasks requiring the understanding of context and long-range dependencies, such as machine translation, sentiment analysis, and speech recognition. Their success has made LSTMs a foundational component in the field of deep learning, providing a powerful tool for modeling sequential data with improved memory and context retention.

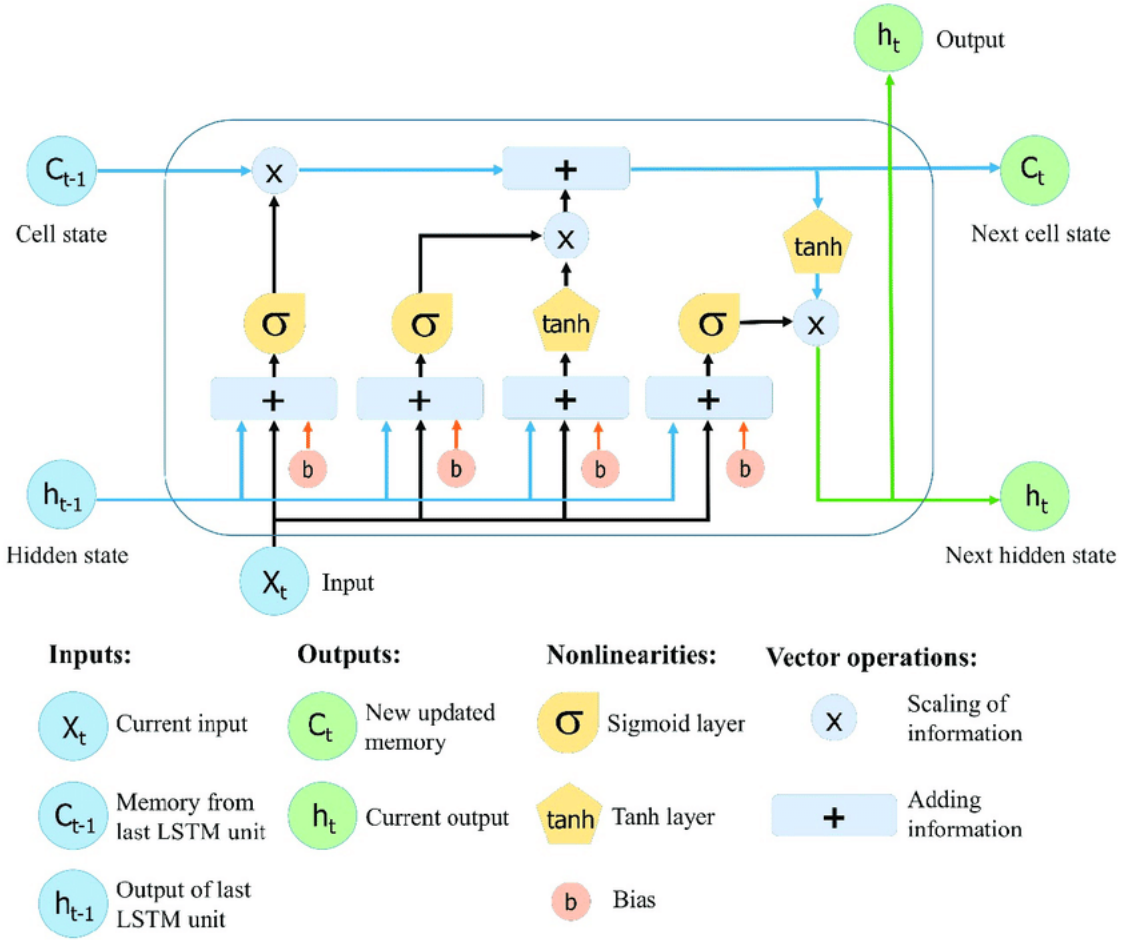

Figure 9: LSTM structure

The core strength of LSTMs lies in their ability to selectively remember or forget information over extended time intervals, allowing them to capture dependencies over long sequences more effectively. This is achieved through the use of memory cells, input, output, and forget gates, each serving a distinct purpose in the learning process.

Memory cells act as storage units that store information over time. The input gate controls the flow of new information into the memory cell, while the forget gate regulates the removal of outdated or irrelevant information. The output gate then determines what information should be passed on to the next time step or output layer. This intricate architecture enables LSTMs to retain relevant information and discard unnecessary details, facilitating the learning of sequential patterns.

LSTMs have proven highly successful in a variety of applications, particularly in natural language processing tasks like language modeling, sentiment analysis, and machine translation. Their ability to model long-range dependencies makes them well-suited for scenarios where understanding the context over extended sequences is crucial. LSTMs have become a fundamental building block in deep learning architectures, contributing significantly to the advancement of sequence-based learning and enabling the development of more sophisticated and capable AI systems.

3.3 Combined Models

A neural network architecture that combines Convolutional Neural Networks (CNNs) with Long Short-Term Memory networks (LSTMs) is commonly referred to as a CNN-LSTM model. This hybrid architecture is particularly effective for tasks that involve sequential data with spatial dependencies, such as video analysis, action recognition, or even certain types of natural language processing tasks.

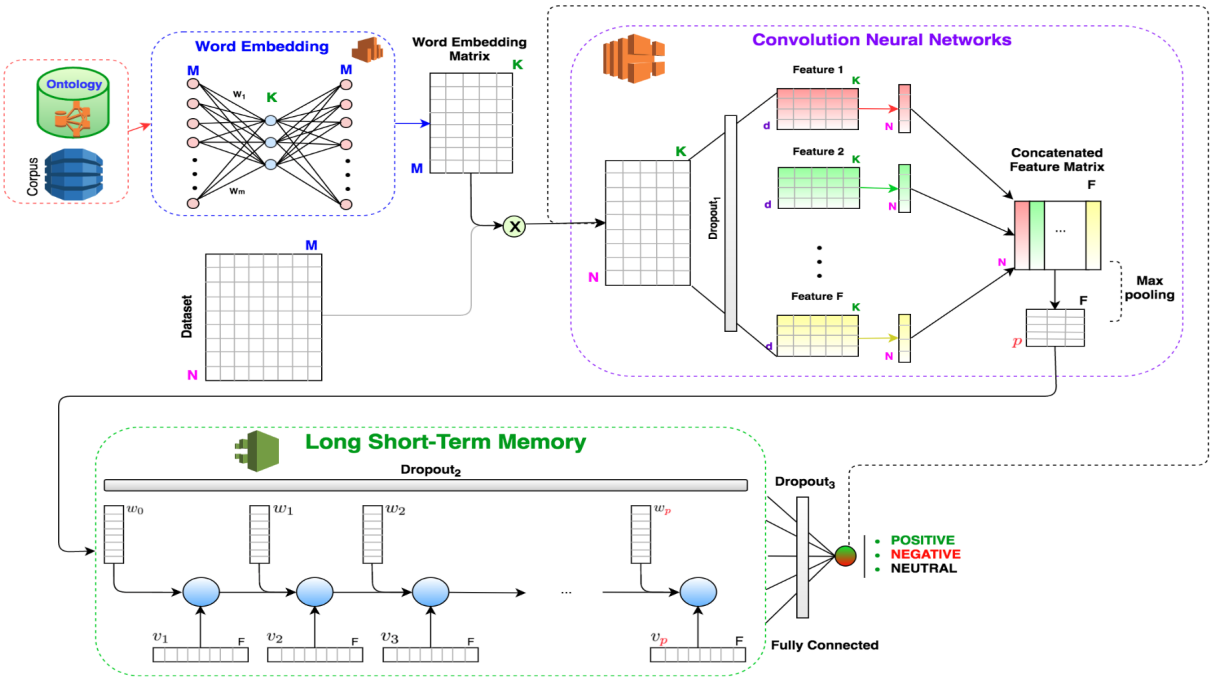

Figure 10: CNN combined with LSTM structure

In a typical CNN-LSTM architecture, the initial layers of the network are comprised of CNN layers. These convolutional layers are well-suited for capturing spatial features and patterns within the input data. In the context of image data, CNNs excel at learning hierarchical representations of visual features, enabling them to recognize patterns such as edges, textures, and object parts.

Following the CNN layers, the output is reshaped and fed into one or more LSTM layers. The LSTM layers are responsible for processing the sequential information, capturing temporal dependencies, and learning patterns over longer sequences. The combination of CNNs and LSTMs leverages the strengths of both architectures: CNNs for spatial feature extraction and LSTMs for sequential modeling.

This architecture is beneficial in scenarios where understanding both spatial and temporal aspects of the data is essential. For instance, in video analysis, CNNs can identify spatial features in individ-

ual frames, while LSTMs can effectively model the temporal evolution of these features over the entire sequence.

The CNN-LSTM model allows for more comprehensive and nuanced representations of complex data, making it a versatile choice for tasks requiring a combined understanding of spatial and temporal patterns. The combination of CNNs and LSTMs has been successfully applied in various domains, showcasing its efficacy in tasks ranging from video analysis to action recognition and beyond.

4 Reused Models for Sentiment Analysis

In the configuration of the reused models below, I tune the Adam optimizer with a learning rate of 0.0005 as the default value of 0.001 is a little bit high, result in uncontrolled oscillating validation loss through training instead of decreasing through testing. The categorical cross-entropy loss function is selected as the optimization criterion. During training, 15% of the data is reserved for validation to monitor the model's performance. The training process is set to run for a maximum of 100 epochs with a batch size of 512. Early stopping is implemented as a callback, which monitors the validation loss. If the validation loss does not improve by at least 0.01 over the course of 4 consecutive epochs, the training will be stopped early, preventing overfitting and optimizing the training time.

```
adam = Adam(learning_rate=0.0005, beta_1=0.9, beta_2=0.999,
    epsilon=1e-08)
model.compile(loss='categorical_crossentropy', optimizer=adam,
    metrics=['accuracy'])
...
early_stopping = EarlyStopping(monitor='val_loss', min_delta=0.01,
     patience=4, verbose=1)
callbacks_list = [early_stopping]

model.fit(data, labels, validation_split=0.15,
          epochs=100, batch_size=512, callbacks=callbacks_list,
             shuffle=True)
```

4.1 CNN

- **Structure:** A CNN with three convolutional layers, each with 100 filters, filter size respectively are 3, 4, and 5.

Layer (type)	Output Shape	Param #	Connected to
input_9 (InputLayer)	[(None, 300)]	0	[]
embedding (Embedding)	(None, 300, 400)	3110400	['input_9[0][0]']
conv1d_17 (Conv1D)	(None, 298, 100)	120100	['embedding[8][0]']
conv1d_18 (Conv1D)	(None, 297, 100)	160100	['embedding[8][0]']
conv1d_19 (Conv1D)	(None, 296, 100)	200100	['embedding[8][0]']
max_pooling1d_17 (MaxPooling1D)	(None, 1, 100)	0	['conv1d_17[0][0]']
max_pooling1d_18 (MaxPooling1D)	(None, 1, 100)	0	['conv1d_18[0][0]']
max_pooling1d_19 (MaxPooling1D)	(None, 1, 100)	0	['conv1d_19[0][0]']
concatenate_6 (Concatenate)	(None, 3, 100)	0	['max_pooling1d_17[0][0]', 'max_pooling1d_18[0][0]', 'max_pooling1d_19[0][0]']
flatten_6 (Flatten)	(None, 300)	0	['concatenate_6[0][0]']
dropout_8 (Dropout)	(None, 300)	0	['flatten_6[0][0]']
dense_8 (Dense)	(None, 3)	903	['dropout_8[0][0]']

Total params: 3591603 (13.70 MB)
Trainable params: 3591603 (13.70 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 11: CNN model details

- **Training results:** The metrics after training for 32 epochs before early stopping are measured as below.

Table 1: Model Evaluation

Model	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy
CNN	1.0251	0.9841	2.9828	0.1686

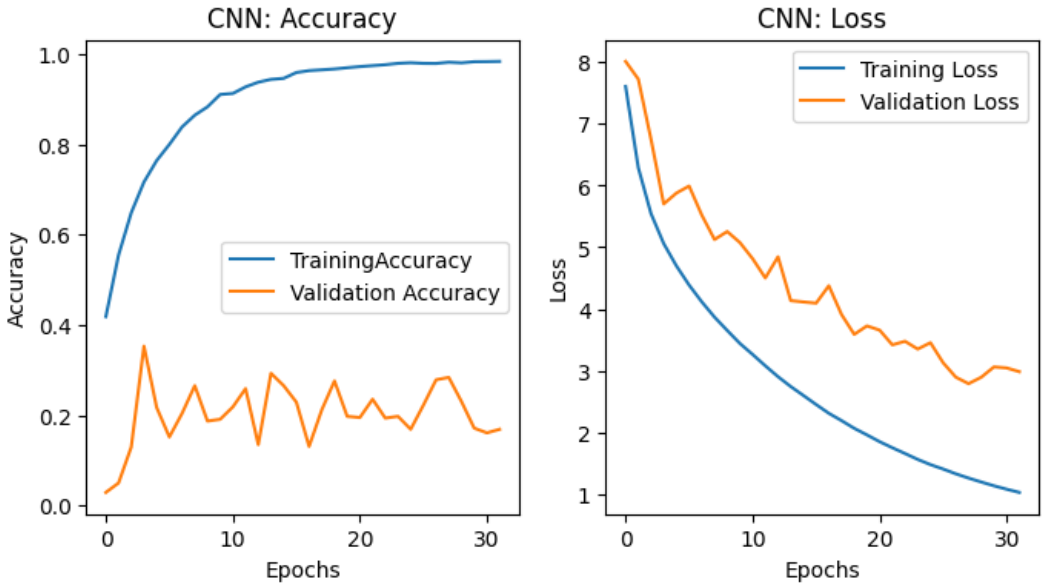

Figure 12: Training accuracy and loss

- **Testing results:** Accuracy: 65.33%(Loss: 1.7449).

  Confusion matrix:

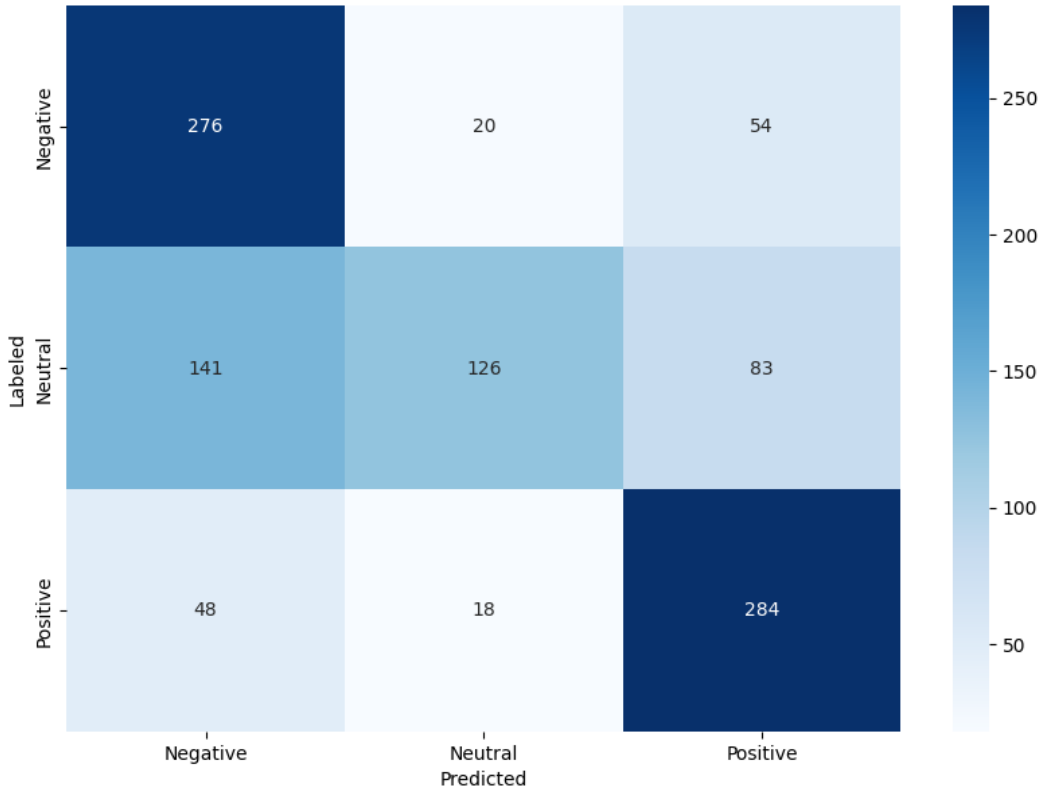

Figure 13: Confusion matrix

Table 2: Testing results

Model	Accuracy	Recall	Precision	F1-Score
CNN	0.6533	0.6533	0.6788	0.6348

4.2 LSTM one layer

- **Structure:** A simple LSTM model with 512 units.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 300)]             0

 embedding (Embedding)       (None, 300, 400)          3110400

 reshape_11 (Reshape)        (None, 300, 400)          0

 lstm_1 (LSTM)               (None, 512)               1869824

 dropout_6 (Dropout)         (None, 512)               0

 dense_6 (Dense)             (None, 3)                 1539

=================================================================
Total params: 4981763 (19.00 MB)
Trainable params: 4981763 (19.00 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 14: LSTM 1 layer model details

- **Training results:** The metrics after training for 11 epochs before early stopping are measured as

below.

Table 3: Model Evaluation

Model	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy
LSTM 1 layer	0.2616	0.9303	2.6989	0.2209

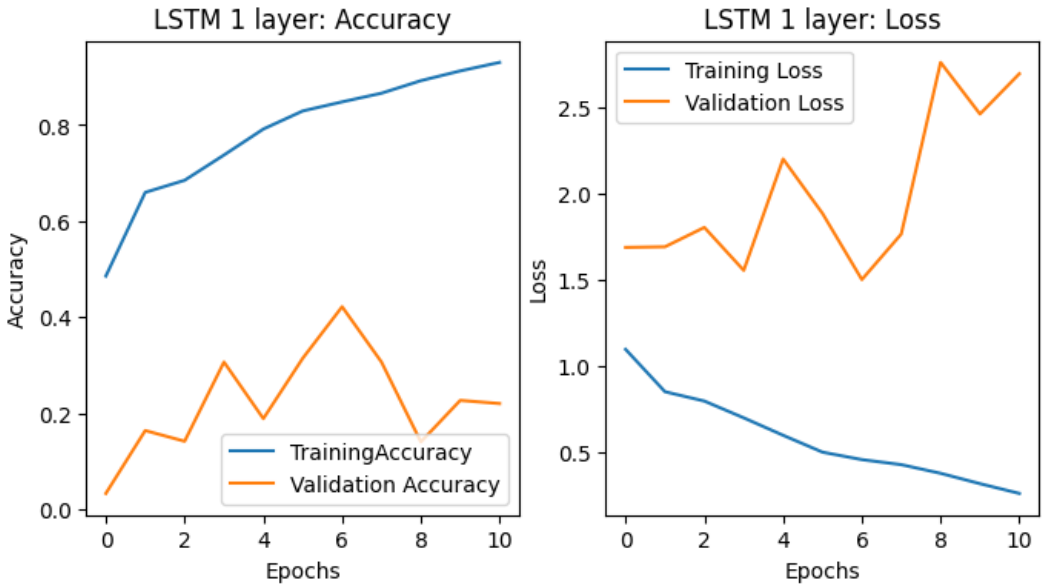

Figure 15: Training accuracy and loss

- **Testing results:** Accuracy: 65.33%(Loss: 1.0714).
  Confusion matrix:

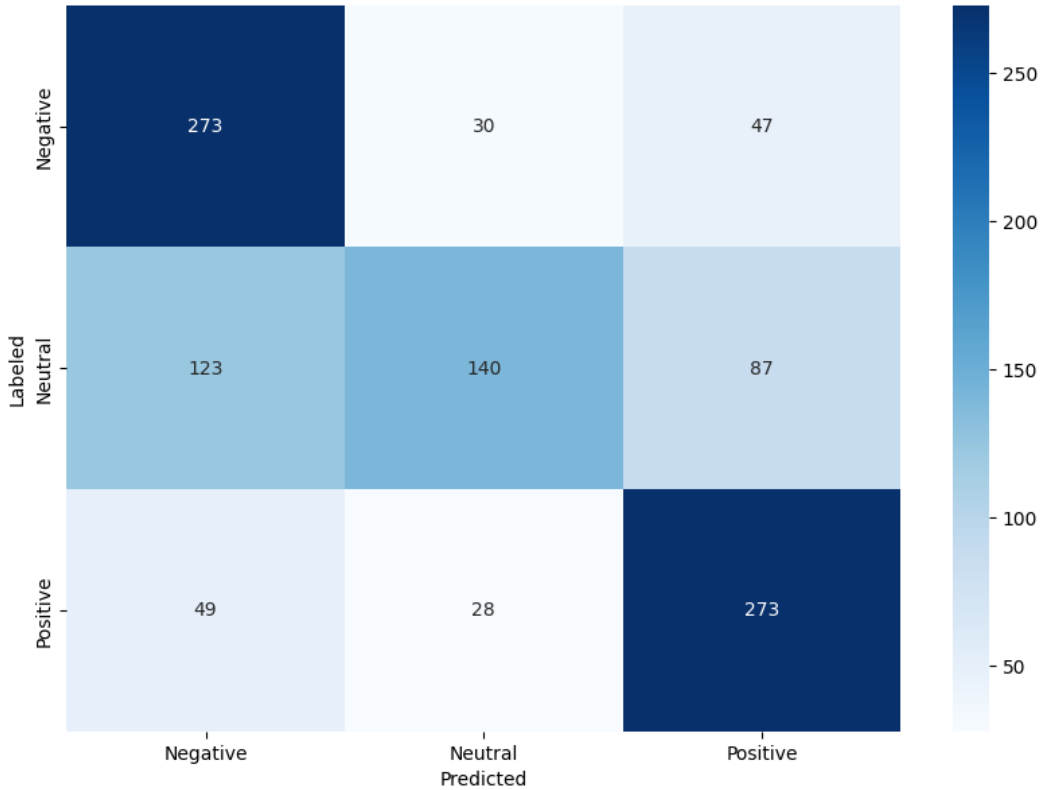

Figure 16: Confusion matrix

Table 4: Testing results

Model	Accuracy	Recall	Precision	F1-Score
LSTM 1 layer	0.6533	0.6533	0.6638	0.6397

4.3 LSTM three layer

- **Structure:** A LSTM model with 3 layers, unit sizes are 1024, 512, 256 respectively.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_15 (InputLayer)       [(None, 300)]             0

 embedding (Embedding)       (None, 300, 400)          3110400

 reshape_31 (Reshape)        (None, 300, 400)          0

 lstm_6 (LSTM)               (None, 300, 1024)         5836800

 lstm_7 (LSTM)               (None, 300, 512)          3147776

 lstm_8 (LSTM)               (None, 256)               787456

 dropout_14 (Dropout)        (None, 256)               0

 dense_14 (Dense)            (None, 3)                 771

=================================================================
Total params: 12883203 (49.15 MB)
Trainable params: 12883203 (49.15 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 17: LSTM 3 layers model details

- **Training results:** The metrics after training for 5 epochs before early stopping are measured as below.

Table 5: Model Evaluation

Model	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy
LSTM 3 layers	0.4739	0.8544	2.6398	0.1346

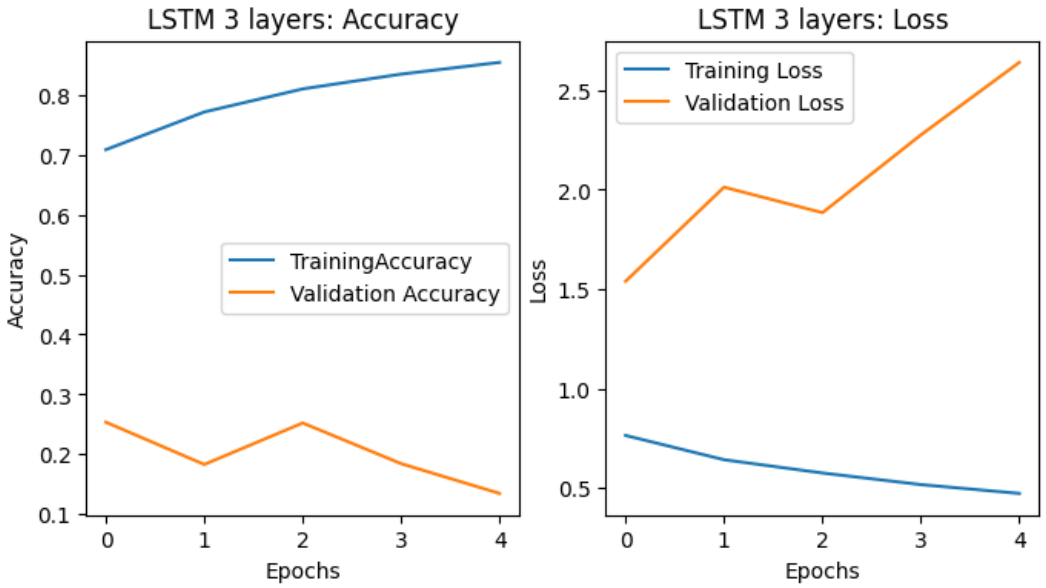

Figure 18: Training accuracy and loss

- **Testing results:** Accuracy: 61.33%(Loss: 1.1439).
  Confusion matrix:

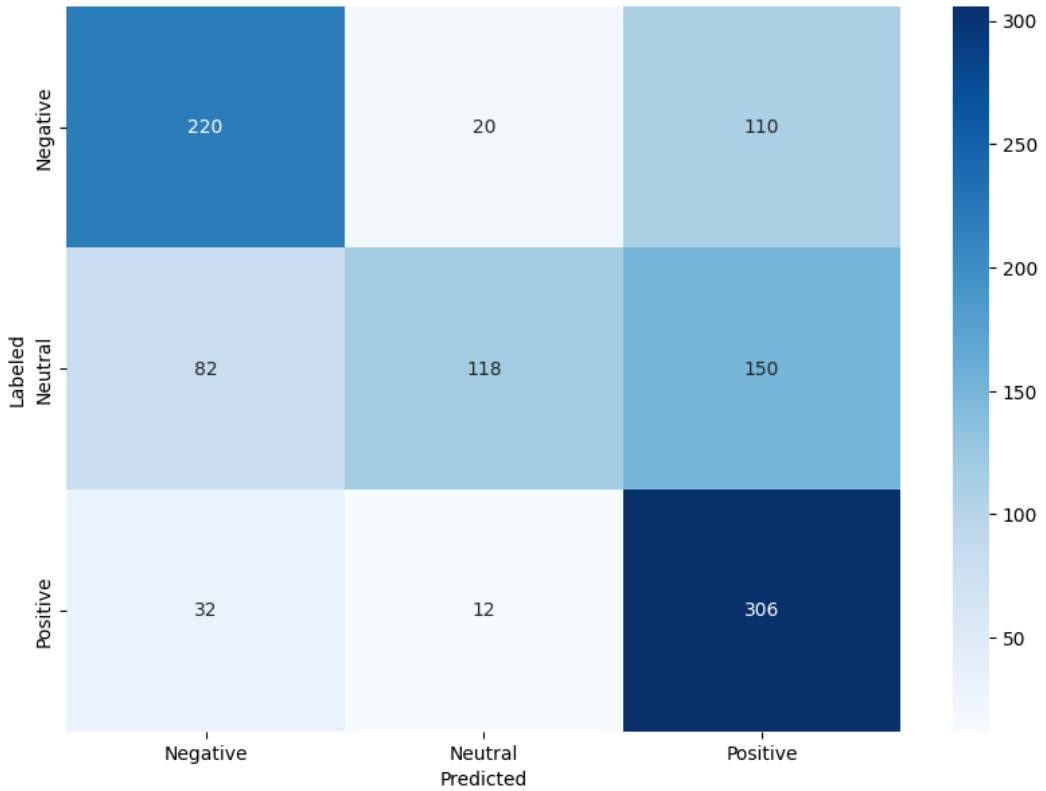

Figure 19: Confusion matrix

Table 6: Testing results

Model	Accuracy	Recall	Precision	F1-Score
LSTM 3 layers	0.6133	0.6133	0.6620	0.5945

5 Modified Models for Sentiment Analysis

Same tuning of the reuse models(learning rate, validation split, batch sizes...) are applied to the configuration and training progress of these models below.

5.1 Larger CNN

- **Structure:** A CNN with three convolutional layers, each with 200 filters, filter size respectively are 6, 10, and 14.

```
Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
input_10 (InputLayer)       [(None, 300)]             0         []

embedding (Embedding)       (None, 300, 400)          3110400   ['input_10[0][0]']

conv1d_20 (Conv1D)          (None, 295, 200)          480200    ['embedding[9][0]']

conv1d_21 (Conv1D)          (None, 291, 200)          800200    ['embedding[9][0]']

conv1d_22 (Conv1D)          (None, 287, 200)          1120200   ['embedding[9][0]']

max_pooling1d_20 (MaxPooli  (None, 1, 200)            0         ['conv1d_20[0][0]']
ng1D)

max_pooling1d_21 (MaxPooli  (None, 1, 200)            0         ['conv1d_21[0][0]']
ng1D)

max_pooling1d_22 (MaxPooli  (None, 1, 200)            0         ['conv1d_22[0][0]']
ng1D)

concatenate_7 (Concatenate  (None, 3, 200)            0         ['max_pooling1d_20[0][0]',
)                                                                'max_pooling1d_21[0][0]',
                                                                 'max_pooling1d_22[0][0]']

flatten_7 (Flatten)         (None, 600)               0         ['concatenate_7[0][0]']

dropout_9 (Dropout)         (None, 600)               0         ['flatten_7[0][0]']

dense_9 (Dense)             (None, 3)                 1803      ['dropout_9[0][0]']

==================================================================================================
Total params: 5512803 (21.03 MB)
Trainable params: 5512803 (21.03 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 20: Larger CNN model details

- **Training results:** The metrics after training for 20 epochs are measured as below.

Table 7: Model Evaluation

Model	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy
Larger CNN	2.7503	0.9896	4.7600	0.2131

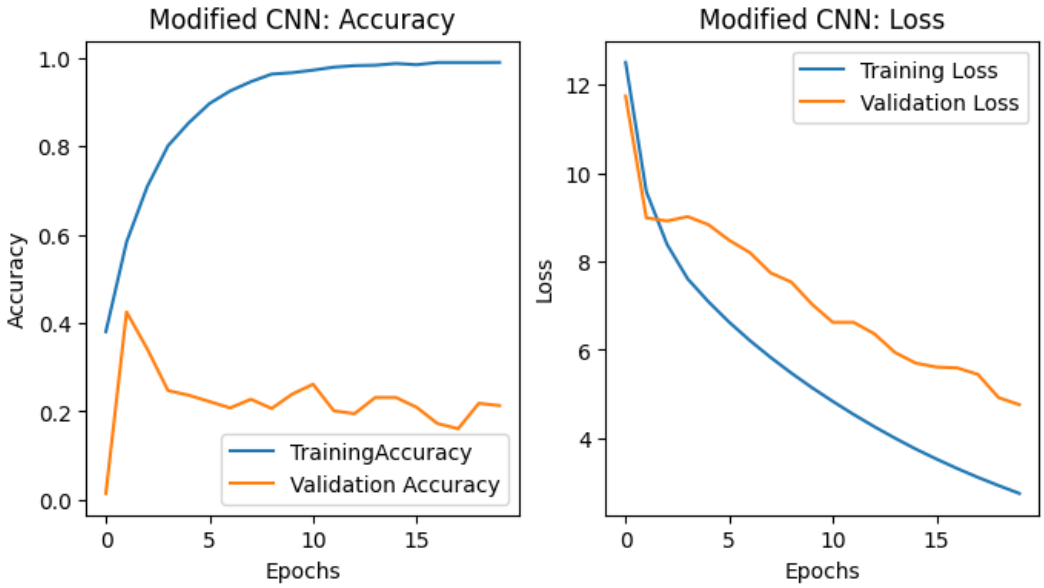

Figure 21: Training accuracy and loss

- **Testing results:** Accuracy: 66.29%(Loss: 3.3278).
  Confusion matrix:

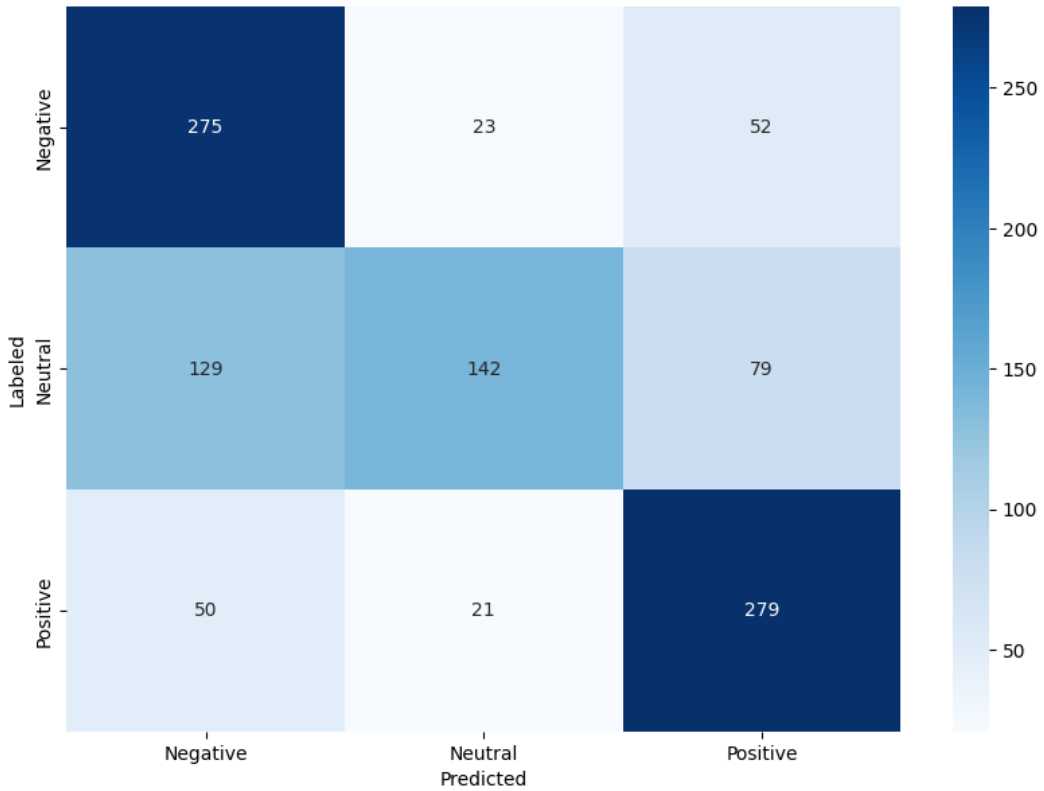

Figure 22: Confusion matrix

Table 8: Testing results

Model	Accuracy	Recall	Precision	F1-Score
Larger CNN	0.6629	0.6629	0.6832	0.6494

5.2 BiLSTM

- **Structure:** A Bidirectional Long Short-Term Memory (BiLSTM) is an architecture that enhances the capabilities of traditional LSTM networks. In a BiLSTM, the input sequence is processed in both forward and backward directions simultaneously by two separate hidden layers. This bidirectional processing allows the network to capture contextual information from both past and future inputs for each time step.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_16 (InputLayer)       [(None, 300)]             0

 embedding (Embedding)       (None, 300, 400)          3110400

 reshape_32 (Reshape)        (None, 300, 400)          0

 bidirectional_4 (Bidirecti  (None, 2048)              11673600
 onal)

 dropout_15 (Dropout)        (None, 2048)              0

 dense_15 (Dense)            (None, 3)                 6147

=================================================================
Total params: 14790147 (56.42 MB)
Trainable params: 14790147 (56.42 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 23: BiLSTM model details

- **Training results:** The metrics after training for 5 epochs before early stopping are measured as below.

Table 9: Model Evaluation

Model	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy
BiLSTM	0.2792	0.9239	1.9211	0.3595

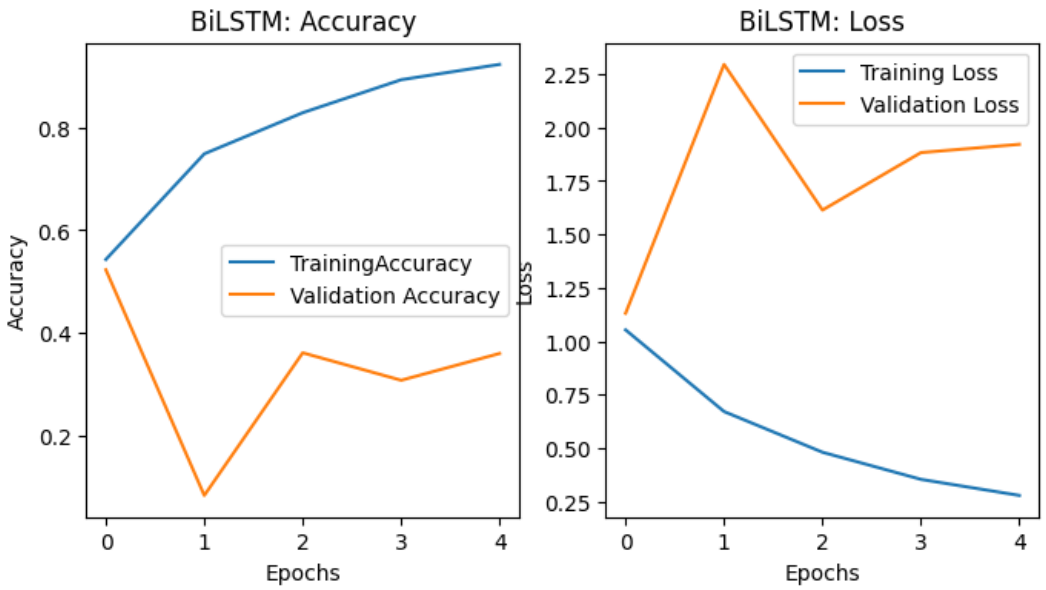

Figure 24: Training accuracy and loss

- **Testing results:** Accuracy: 66.10%(Loss: 3.3278).

  Confusion matrix:

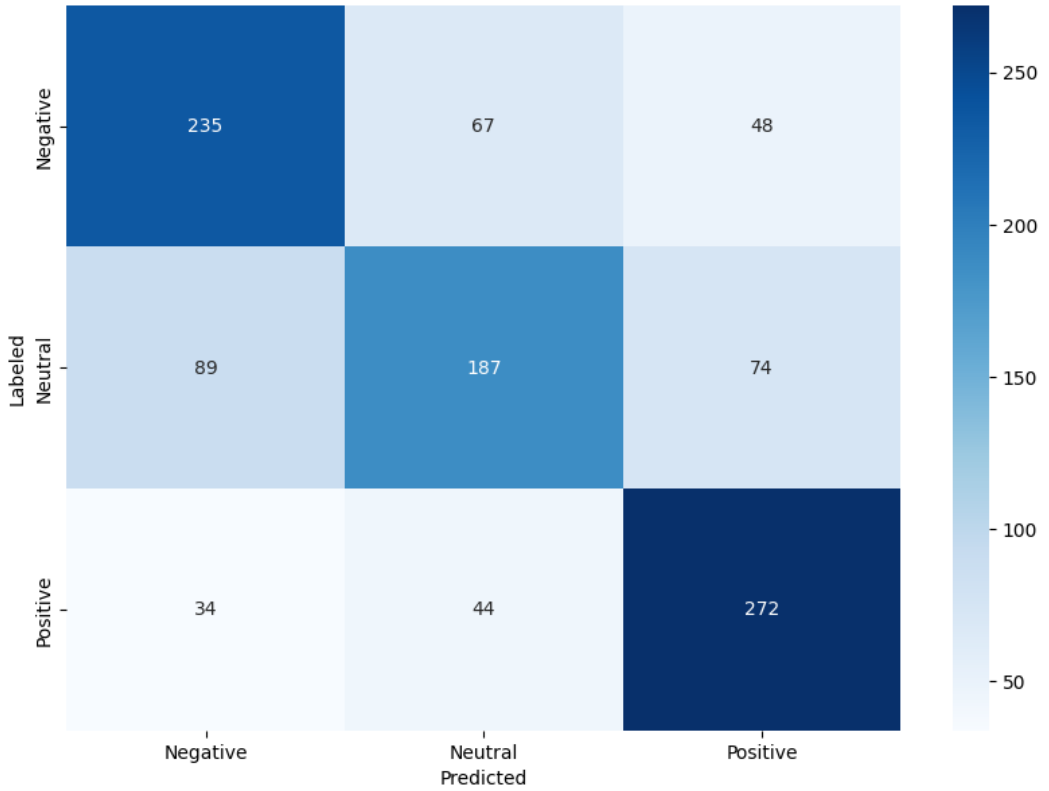

Figure 25: Confusion matrix

Table 10: Testing results

Model	Accuracy	Recall	Precision	F1-Score
BiLSTM	0.6610	0.6610	0.6581	0.6574

5.3 Larger CNN and BiLSTM combined

- **Structure:** Upon analysis above, it becomes clear that the Larger CNN model outperforms its original counterpart, while the BiLSTM model surpasses both the single-layer and three-layer LSTM models. Recognizing these performance trends, I am motivated to propose a hybrid model of Larger CNN and BiLSTM combined(CNN+BiLSTM) that leverages the strengths of both the CNN and BiLSTM architectures, anticipating enhanced results through their collaborative synergy.

Layer (type)	Output Shape	Param #	Connected to
input_14 (InputLayer)	[(None, 300)]	0	[]
embedding (Embedding)	(None, 300, 400)	3110400	['input_14[0][0]']
reshape_27 (Reshape)	(None, 300, 400)	0	['embedding[13][0]']
conv1d_29 (Conv1D)	(None, 300, 200)	480200	['reshape_27[0][0]']
conv1d_30 (Conv1D)	(None, 300, 200)	800200	['reshape_27[0][0]']
conv1d_31 (Conv1D)	(None, 300, 200)	1120200	['reshape_27[0][0]']
max_pooling1d_29 (MaxPooling1D)	(None, 1, 200)	0	['conv1d_29[0][0]']
max_pooling1d_30 (MaxPooling1D)	(None, 1, 200)	0	['conv1d_30[0][0]']
max_pooling1d_31 (MaxPooling1D)	(None, 1, 200)	0	['conv1d_31[0][0]']
reshape_28 (Reshape)	(None, 1, 200)	0	['max_pooling1d_29[0][0]']
reshape_29 (Reshape)	(None, 1, 200)	0	['max_pooling1d_30[0][0]']
reshape_30 (Reshape)	(None, 1, 200)	0	['max_pooling1d_31[0][0]']
concatenate_10 (Concatenate)	(None, 1, 600)	0	['reshape_28[0][0]', 'reshape_29[0][0]', 'reshape_30[0][0]']
bidirectional_3 (Bidirectional)	(None, 1024)	4558848	['concatenate_10[0][0]']
dropout_13 (Dropout)	(None, 1024)	0	['bidirectional_3[0][0]']
dense_13 (Dense)	(None, 3)	3075	['dropout_13[0][0]']

Total params: 10072923 (38.43 MB)
Trainable params: 10072923 (38.43 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 26: CNN+BiLSTM model details

- **Training results:** The metrics after training for 7 epochs are measured as below.

Table 11: Model Evaluation

Model	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy
CNN+BiLSTM	0.4330	0.9993	1.6749	0.5451

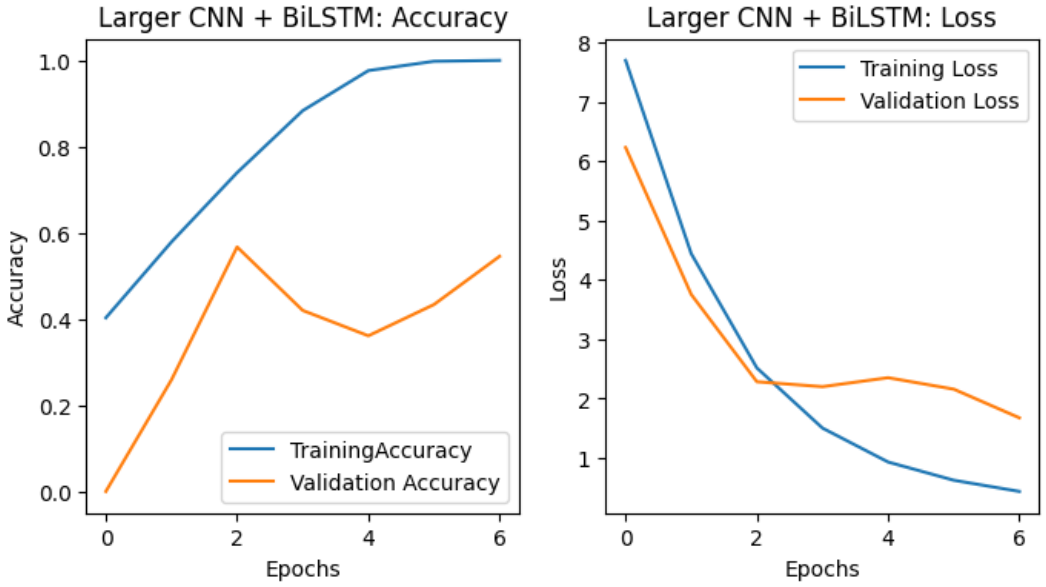

Figure 27: Training accuracy and loss

- **Testing results:** Accuracy: 70.00%(Loss: 1.1747).
  Confusion matrix:

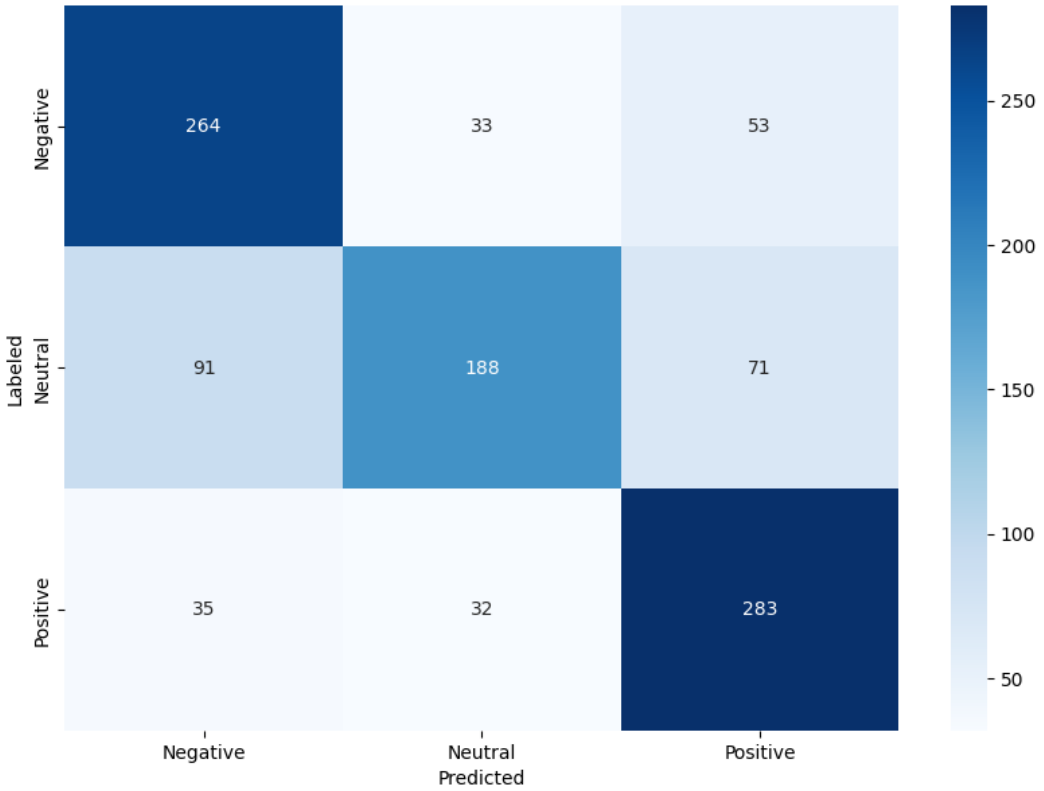

Figure 28: Confusion matrix

Table 12: Testing results

Model	Accuracy	Recall	Precision	F1-Score
CNN+BiLSTM	0.7000	0.7000	0.7051	0.6949

6 Results

The training and testing progress has notable similarities across the various models. While the models demonstrate rapid convergence to high training accuracy, a discernible disparity emerges in achieving commensurate validation accuracy. This suggests potential overfitting tendencies, where the models might excel in memorizing the training data but face challenges in generalizing well to new, unseen data during validation.

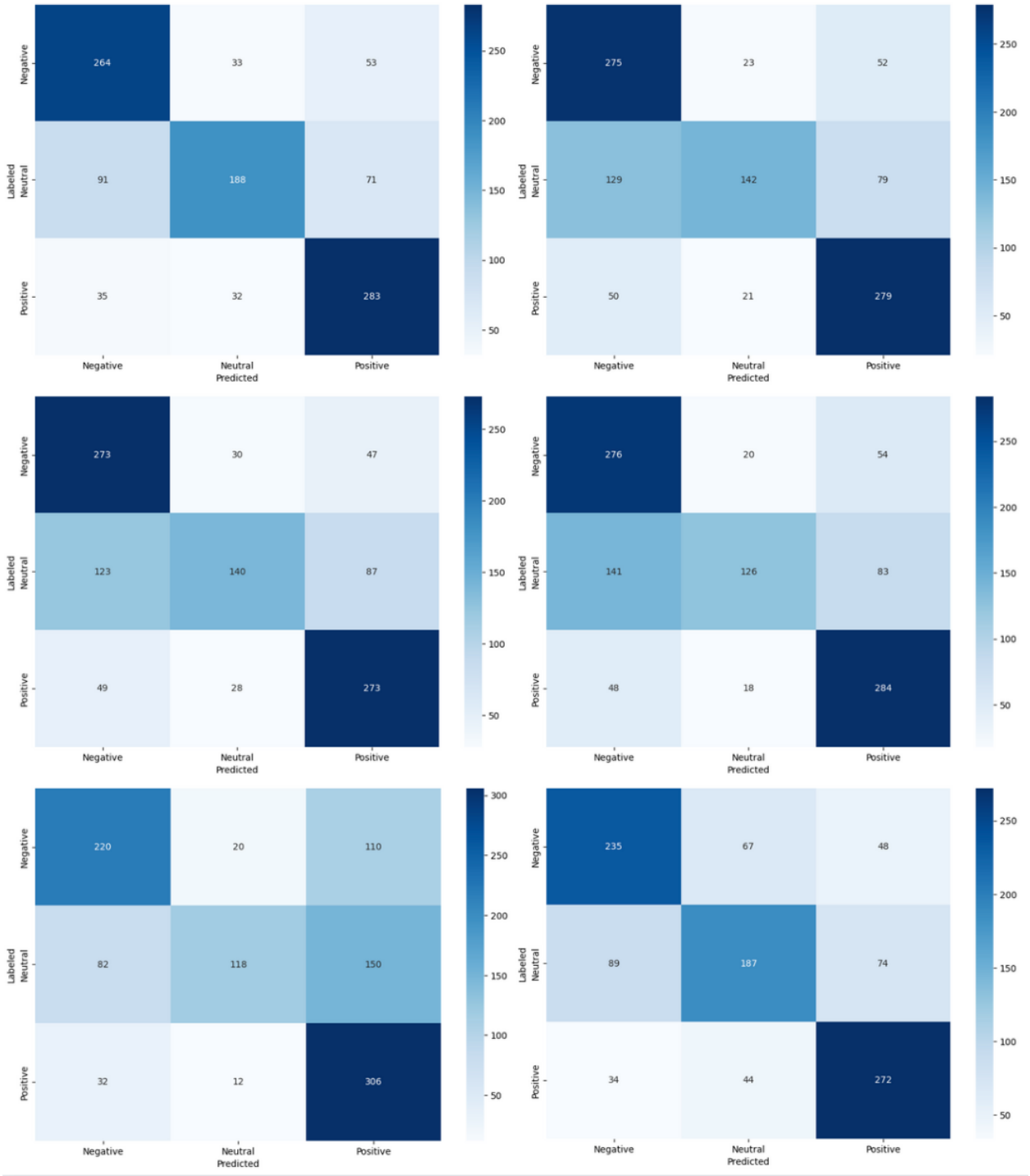

Figure 29: Confusion matrices

Analyzing the confusion matrices reveals a clear pattern in the models' proficiency in predicting sentiments labeled as Negative and Positive. These sentiments are generally handled with relative ease,

showcasing the models' capability in capturing the nuances of these categories. However, a substantial challenge surfaces when it comes to accurately classifying instances within the Neutral category. The models often struggle with this specific sentiment, resulting in a considerable number of misclassifications into other sentiment categories. Addressing this challenge is pivotal for enhancing the overall accuracy and effectiveness of sentiment classification.

These are the final results of the models.

Model	Test Accuracy
CNN+BiLSTM	70.00%
Larger CNN	66.29%
BiLSTM	66.10%
LSTM 1 layer	65.33%
CNN	65.33%
LSTM 3 layers	61.33%

In the pursuit of enhancing sentiment analysis performance, several modified and original models have been evaluated. The results indicate that the Larger CNN combined with BiLSTM model achieved the highest accuracy at 70.00%, outperforming both the Larger CNN (66.29%) and the standalone BiLSTM (66.10%). This suggests that the modifications introduced, particularly the combination of Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM), contributed to a more effective sentiment analysis framework, showcasing the importance of model architecture in achieving superior results.

As expected, Larger CNN or BiLSTM also surpassed the accuracy of the original models, namely LSTM with one layer (65.33%) and standalone CNN (65.33%). Surprisingly, BiLSTM performs better than LSTM with three layers (61.33%). This finding suggests that the Bidirectional LSTM, with its ability to capture contextual information from both past and future sequences, demonstrated superior performance compared to a deeper but unidirectional LSTM architecture. The efficiency of BiLSTM in understanding nuanced patterns within text sequences underscores its potential as a powerful tool for sentiment analysis tasks.

7 Conclusion

The presented accuracy percentages across various models highlight the inherent complexities of the sentiment analysis task. Achieving accurate sentiment classification is a challenging endeavor, as indicated by the modest performance across different architectures. Notably, the CNN+BiLSTM model stands out with an accuracy of 70.00%, surpassing other configurations. This suggests that combining Convolutional Neural Networks (CNN) with Bidirectional Long Short-Term Memory (BiLSTM) networks provides a more effective approach in capturing and understanding sentiment patterns within the given dataset. The superior performance of the CNN+BiLSTM model underscores the significance of leveraging both spatial hierarchies through CNN and contextual dependencies through BiLSTM for a more robust sentiment analysis solution.

Working Notebook: https://colab.research.google.com/drive/1AUk9NNu-WqArcDST1xOVNSMGQT7er521?usp=sharing