Project 1: Text Classification with TensorFlow

ABSTRACT :

This project focuses on building a text classification model using TensorFlow. The model is designed to classify text data into predefined categories, demonstrating the entire workflow from data preprocessing to model training and evaluation. The 20 Newsgroups dataset is utilized for this purpose, showcasing practical applications of machine learning techniques in natural language processing.

OBJECTIVE :

The primary objective of this project is to develop a neural network model capable of classifying text documents into specific categories using TensorFlow. The project aims to illustrate the processes of data preprocessing, model construction, training, evaluation, and prediction within a machine learning context.

INTRODUCTION :

Text classification is a crucial task in natural language processing, enabling the organization and categorization of text data into various categories based on their content. Applications range from spam detection in emails to sentiment analysis in customer reviews. This project employs the 20 Newsgroups dataset, a popular benchmark in text classification, to build a TensorFlow-based neural network model. By following a systematic approach, this project demonstrates how to effectively preprocess text data, construct a neural network, train the model, and evaluate its performance.

METHODOLOGY :

1.Data preparation

The 20 Newsgroups dataset is fetched using scikit-learn's fetch_20newsgroups function. The dataset is divided into training and testing sets.

2. Data processing

Tokenization and padding are essential steps to convert text data into a numerical format suitable for neural network training.

3. Model Building

A neural network model is constructed using TensorFlow's Keras API. The model consists of an embedding layer, a global average pooling layer, and dense layers.

4. Model Training

The model is trained on the training dataset, with performance monitored on the validation dataset.

5. Evaluation

The model's performance is evaluated using accuracy and loss metrics, and the results are visualized through plots.

6. Prediction

The model is used to classify new text samples, demonstrating its practical application

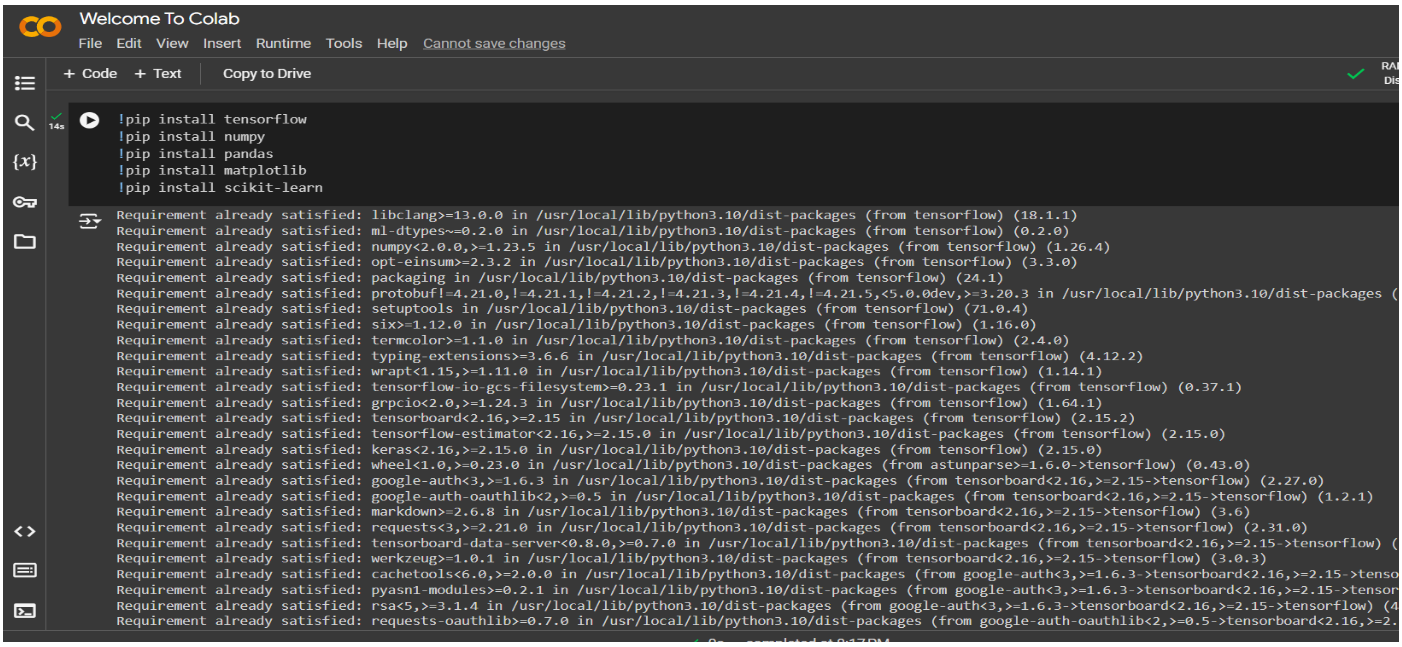
CODE :

```
!pip install tensorflow
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install scikit-learn


from sklearn.datasets import fetch_20newsgroups
import pandas as pd

newsgroups_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42)
newsgroups_test = fetch_20newsgroups(subset='test', shuffle=True, random_state=42)

train_data = newsgroups_train.data
train_labels = newsgroups_train.target

test_data = newsgroups_test.data
test_labels = newsgroups_test.target


from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Define parameters
vocab_size = 10000
max_length = 200
trunc_type = 'post'
padding_type = 'post'
oov_tok = "<OOV>"

# Tokenize and pad sequences
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train_data)
```

```
train_sequences = tokenizer.texts_to_sequences(train_data)
train_padded = pad_sequences(train_sequences, maxlen=max_length,
padding=padding_type, truncating=trunc_type)

test_sequences = tokenizer.texts_to_sequences(test_data)
test_padded = pad_sequences(test_sequences, maxlen=max_length, padding=padding_type,
truncating=trunc_type)


import numpy as np

train_labels = np.array(train_labels)
test_labels = np.array(test_labels)

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D,Dense

model = Sequential([
    Embedding(vocab_size, 16, input_length=max_length),
    GlobalAveragePooling1D(),
    Dense(24, activation='relu'),
    Dense(20, activation='softmax')
])
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()


history = model.fit(train_padded, train_labels, epochs=30, validation_data=(test_padded,
test_labels), verbose=2)

import matplotlib.pyplot as plt

def plot_graphs(history, string):
    plt.plot(history.history[string])
    plt.plot(history.history['val_' + string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_' + string])
    plt.show()

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")

new_text = ["This is a sample text to classify"]
new_seq = tokenizer.texts_to_sequences(new_text)
new_padded = pad_sequences(new_seq, maxlen=max_length, padding=padding_type,
truncating=trunc_type)

predicted_class = model.predict(new_padded)
print(f"Predicted class: {np.argmax(predicted_class)}")
```

output :

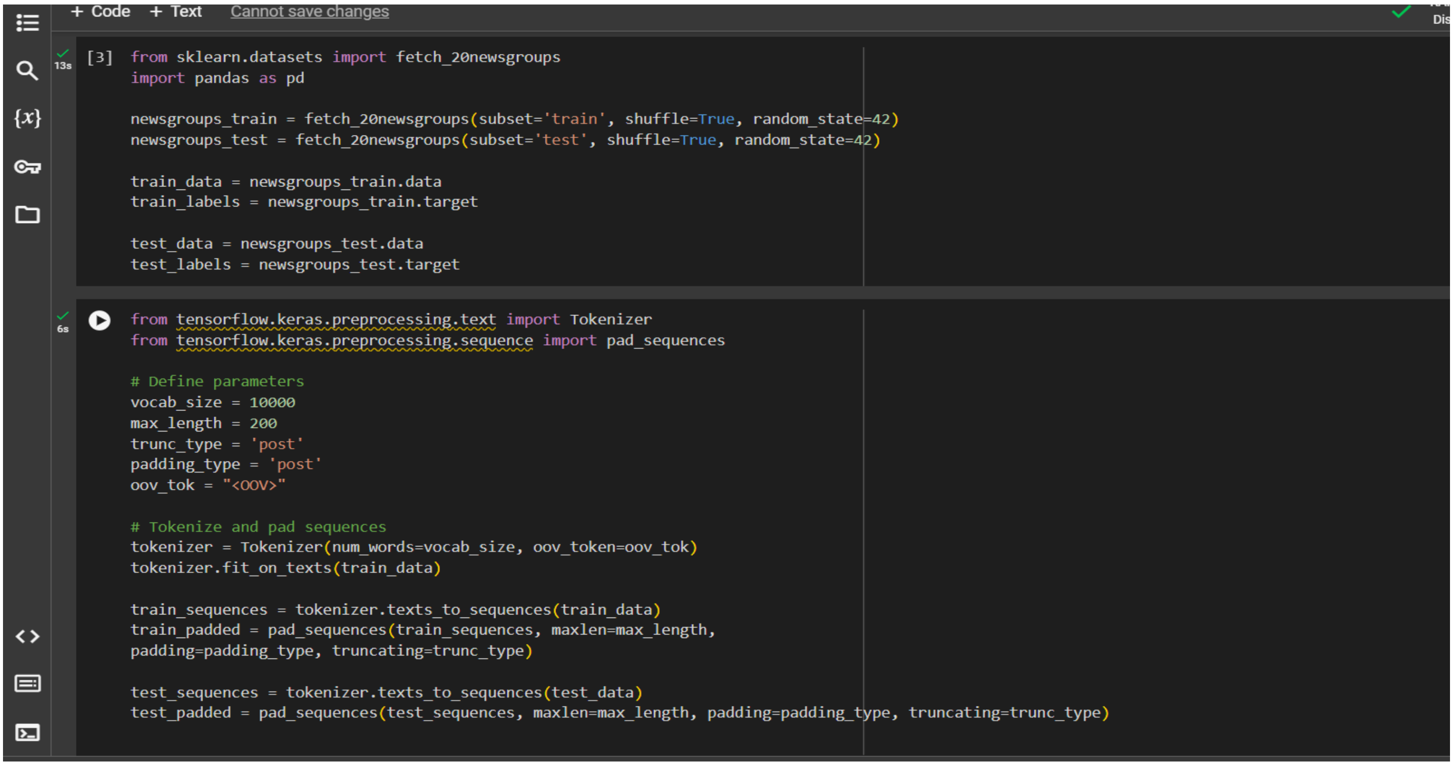
1. Setup Environment


Welcome To Colab
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
14s

```
!pip install tensorflow
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install scikit-learn
```

```
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.43.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (1.2.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.6)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.0.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tenso
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensor
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (4
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.
```

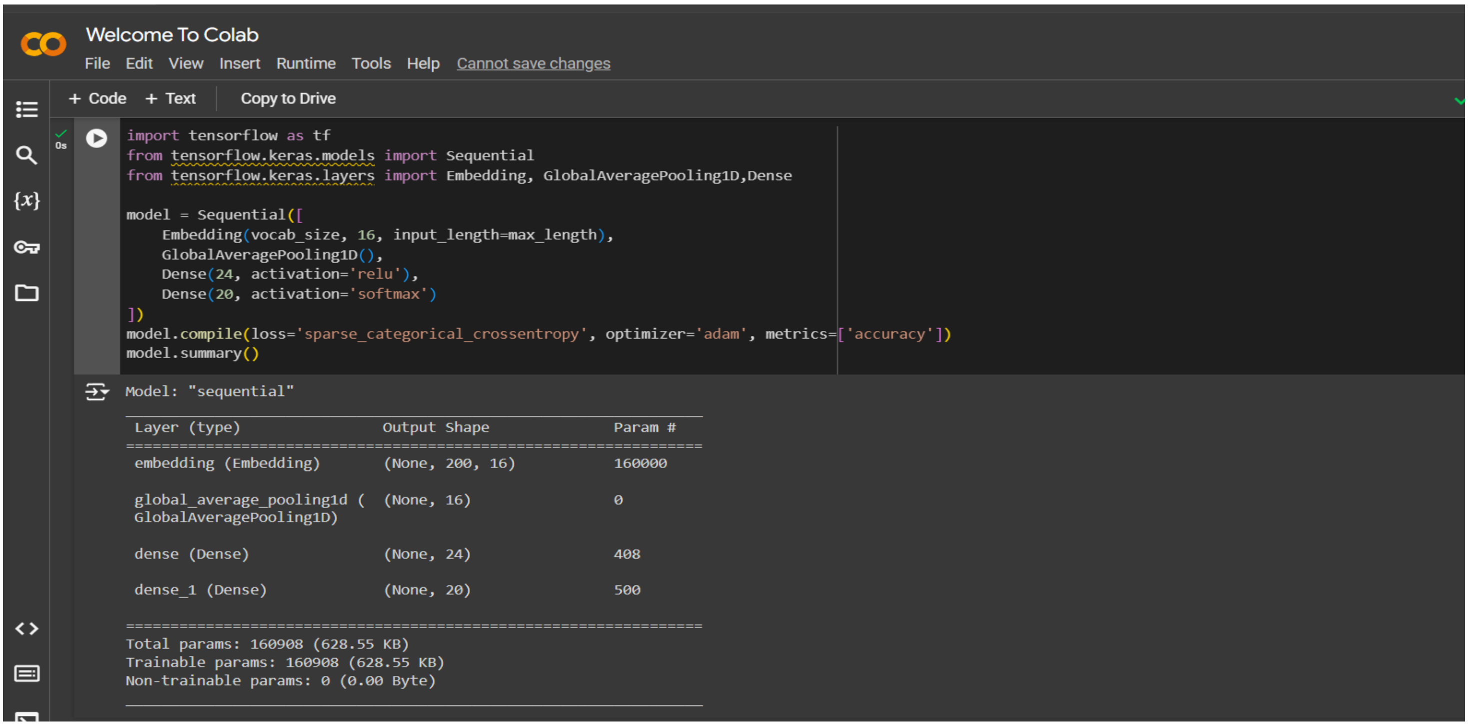
2. Data Preprocessing



```
[3] from sklearn.datasets import fetch_20newsgroups
    import pandas as pd

    newsgroups_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42)
    newsgroups_test = fetch_20newsgroups(subset='test', shuffle=True, random_state=42)

    train_data = newsgroups_train.data
    train_labels = newsgroups_train.target

    test_data = newsgroups_test.data
    test_labels = newsgroups_test.target
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Define parameters
vocab_size = 10000
max_length = 200
trunc_type = 'post'
padding_type = 'post'
oov_tok = "<OOV>"

# Tokenize and pad sequences
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(train_data)

train_sequences = tokenizer.texts_to_sequences(train_data)
train_padded = pad_sequences(train_sequences, maxlen=max_length,
padding=padding_type, truncating=trunc_type)

test_sequences = tokenizer.texts_to_sequences(test_data)
test_padded = pad_sequences(test_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)
```

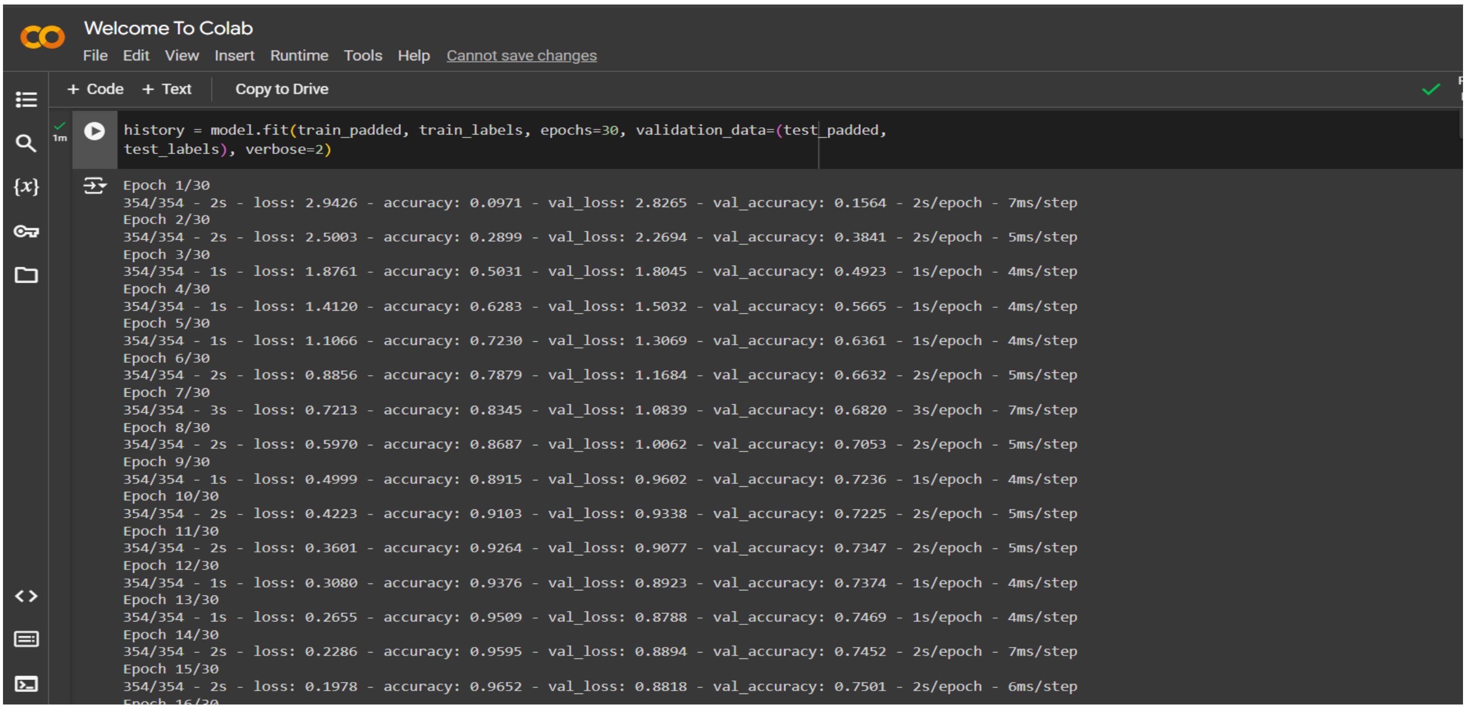
3. Building the model


Welcome To Colab
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
0s

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D,Dense

model = Sequential([
    Embedding(vocab_size, 16, input_length=max_length),
    GlobalAveragePooling1D(),
    Dense(24, activation='relu'),
    Dense(20, activation='softmax')
])
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 16)           160000

 global_average_pooling1d (  (None, 16)                0
 GlobalAveragePooling1D)

 dense (Dense)               (None, 24)                408

 dense_1 (Dense)             (None, 20)                500

=================================================================
Total params: 160908 (628.55 KB)
Trainable params: 160908 (628.55 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

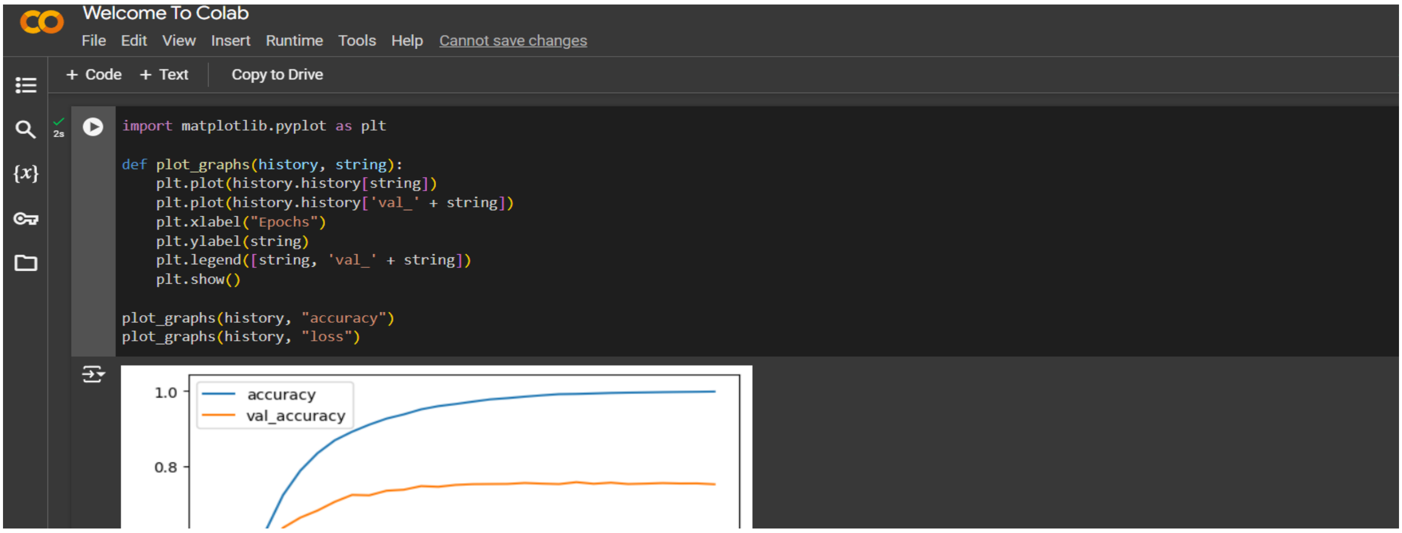
4. Training the Model


Welcome To Colab
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
1m

```
history = model.fit(train_padded, train_labels, epochs=30, validation_data=(test_padded,
test_labels), verbose=2)
```

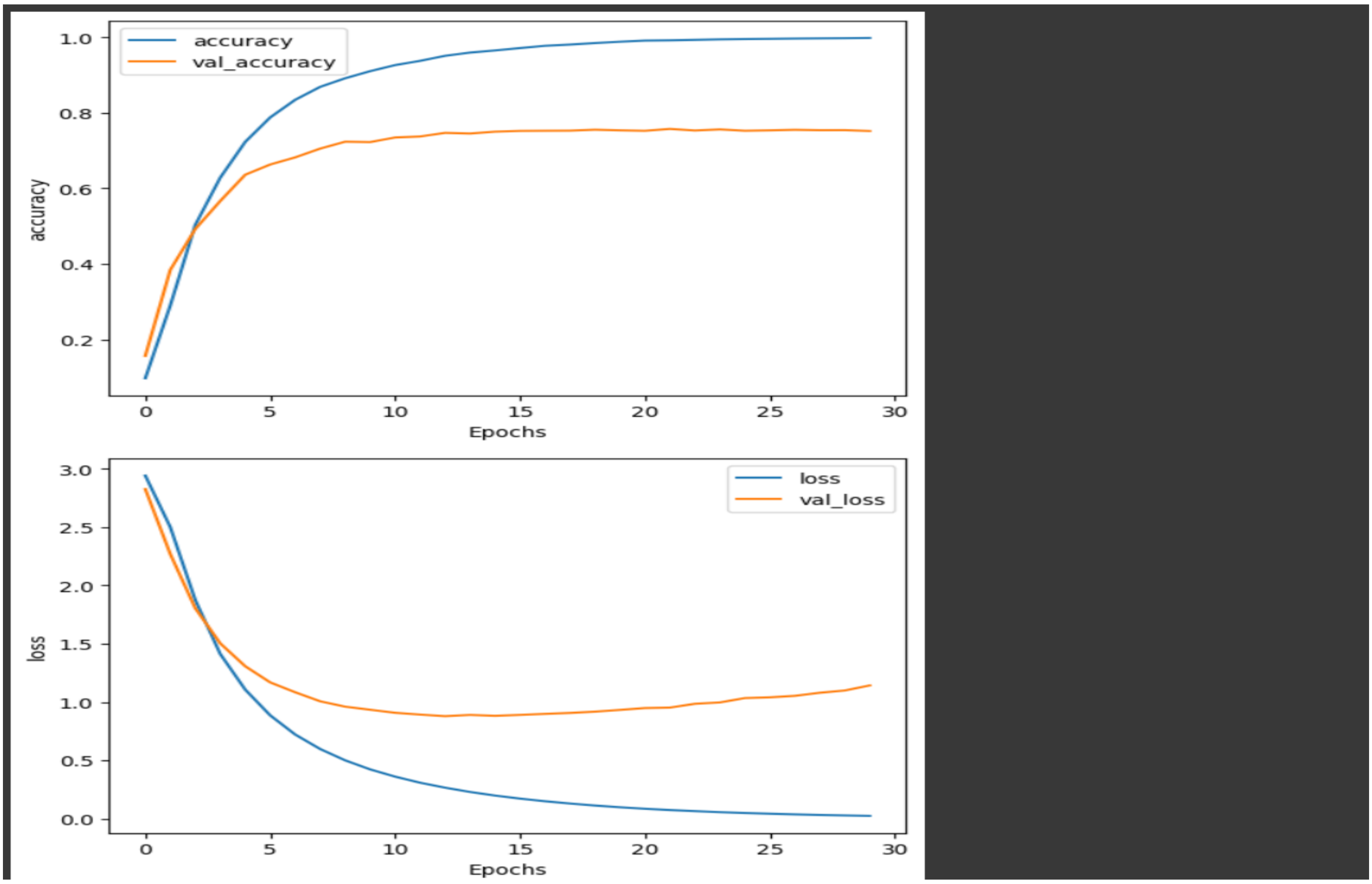
```
Epoch 1/30
354/354 - 2s - loss: 2.9426 - accuracy: 0.0971 - val_loss: 2.8265 - val_accuracy: 0.1564 - 2s/epoch - 7ms/step
Epoch 2/30
354/354 - 2s - loss: 2.5003 - accuracy: 0.2899 - val_loss: 2.2694 - val_accuracy: 0.3841 - 2s/epoch - 5ms/step
Epoch 3/30
354/354 - 1s - loss: 1.8761 - accuracy: 0.5031 - val_loss: 1.8045 - val_accuracy: 0.4923 - 1s/epoch - 4ms/step
Epoch 4/30
354/354 - 1s - loss: 1.4120 - accuracy: 0.6283 - val_loss: 1.5032 - val_accuracy: 0.5665 - 1s/epoch - 4ms/step
Epoch 5/30
354/354 - 1s - loss: 1.1066 - accuracy: 0.7230 - val_loss: 1.3069 - val_accuracy: 0.6361 - 1s/epoch - 4ms/step
Epoch 6/30
354/354 - 2s - loss: 0.8856 - accuracy: 0.7879 - val_loss: 1.1684 - val_accuracy: 0.6632 - 2s/epoch - 5ms/step
Epoch 7/30
354/354 - 3s - loss: 0.7213 - accuracy: 0.8345 - val_loss: 1.0839 - val_accuracy: 0.6820 - 3s/epoch - 7ms/step
Epoch 8/30
354/354 - 2s - loss: 0.5970 - accuracy: 0.8687 - val_loss: 1.0062 - val_accuracy: 0.7053 - 2s/epoch - 5ms/step
Epoch 9/30
354/354 - 1s - loss: 0.4999 - accuracy: 0.8915 - val_loss: 0.9602 - val_accuracy: 0.7236 - 1s/epoch - 4ms/step
Epoch 10/30
354/354 - 2s - loss: 0.4223 - accuracy: 0.9103 - val_loss: 0.9338 - val_accuracy: 0.7225 - 2s/epoch - 5ms/step
Epoch 11/30
354/354 - 2s - loss: 0.3601 - accuracy: 0.9264 - val_loss: 0.9077 - val_accuracy: 0.7347 - 2s/epoch - 5ms/step
Epoch 12/30
354/354 - 1s - loss: 0.3080 - accuracy: 0.9376 - val_loss: 0.8923 - val_accuracy: 0.7374 - 1s/epoch - 4ms/step
Epoch 13/30
354/354 - 1s - loss: 0.2655 - accuracy: 0.9509 - val_loss: 0.8788 - val_accuracy: 0.7469 - 1s/epoch - 4ms/step
Epoch 14/30
354/354 - 2s - loss: 0.2286 - accuracy: 0.9595 - val_loss: 0.8894 - val_accuracy: 0.7452 - 2s/epoch - 7ms/step
Epoch 15/30
354/354 - 2s - loss: 0.1978 - accuracy: 0.9652 - val_loss: 0.8818 - val_accuracy: 0.7501 - 2s/epoch - 6ms/step
Epoch 16/30
```

5. Evaluation

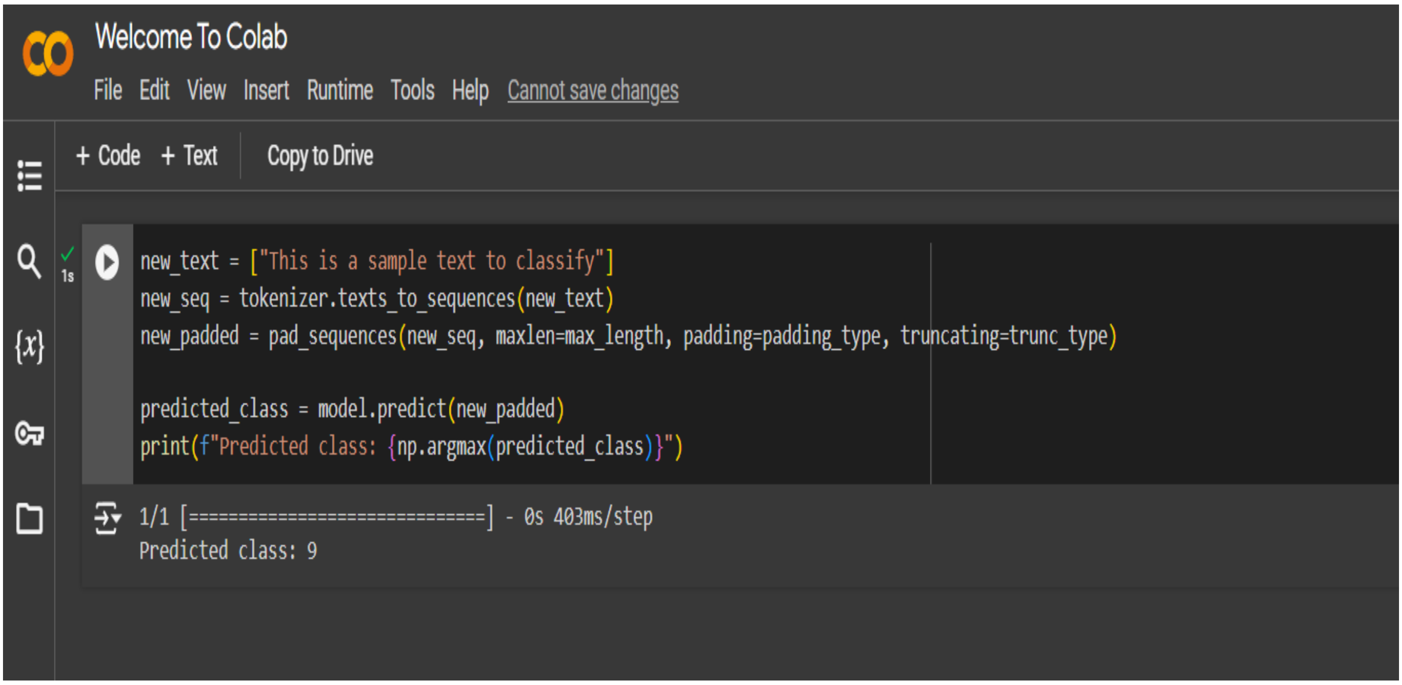


Welcome To Colab
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
1.0
0.8
accuracy
val_accuracy

```
import matplotlib.pyplot as plt

def plot_graphs(history, string):
    plt.plot(history.history[string])
    plt.plot(history.history['val_' + string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_' + string])
    plt.show()

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
```



6. Prediction


Welcome To Colab
File Edit View Insert Runtime Tools Help Cannot save changes
+ Code + Text Copy to Drive
1s

```
new_text = ["This is a sample text to classify"]
new_seq = tokenizer.texts_to_sequences(new_text)
new_padded = pad_sequences(new_seq, maxlen=max_length, padding=padding_type, truncating=trunc_type)

predicted_class = model.predict(new_padded)
print(f"Predicted class: {np.argmax(predicted_class)}")
```

```
1/1 [==============================] - 0s 403ms/step
Predicted class: 9
```

Conclusion :

This project successfully demonstrates the process of building a text classification model using TensorFlow. By preprocessing the text data, constructing a neural network, and training the model on the 20 Newsgroups dataset, we achieved a model capable of classifying text into various categories. The model's performance was evaluated using accuracy and loss metrics, with visualizations aiding in understanding the training process. This project underscores the potential of machine learning in natural language processing tasks and provides a foundation for further exploration and refinement in text classification models.