Assignment-4

Text and Sequence

Name: Tejasree Gottam

student id: 811358524

```
!pip install tensorflow==2.12
```

```
Collecting tensorflow==2.12
  Downloading tensorflow-2.12.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.4 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (25.2.10)
Collecting gast<=0.4.0,>=0.2.1 (from tensorflow==2.12)
  Downloading gast-0.4.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (1.71.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (3.13.0)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (0.5.2)
Collecting keras<2.13,>=2.12.0 (from tensorflow==2.12)
  Downloading keras-2.12.0-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (18.1.1)
Collecting numpy<1.24,>=1.22 (from tensorflow==2.12)
  Downloading numpy-1.23.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.3 kB)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (24.2)
Collecting 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 (from tensorflow==2.12)
  Downloading protobuf-4.25.6-cp37-abi3-manylinux2014_x86_64.whl.metadata (541 bytes)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (1.17.0)
Collecting tensorboard<2.13,>=2.12 (from tensorflow==2.12)
  Downloading tensorboard-2.12.3-py3-none-any.whl.metadata (1.8 kB)
Collecting tensorflow-estimator<2.13,>=2.12.0 (from tensorflow==2.12)
  Downloading tensorflow_estimator-2.12.0-py2.py3-none-any.whl.metadata (1.3 kB)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (3.0.1)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow==2.12) (4.13.
Collecting wrapt<1.15,>=1.11.0 (from tensorflow==2.12)
  Downloading wrapt-1.14.1-cp311-cp311-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.meta
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow==
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow=
Requirement already satisfied: jaxlib<=0.5.2,>=0.5.1 in /usr/local/lib/python3.11/dist-packages (from jax>=0.3.15->tensorflow==2.
Requirement already satisfied: ml_dtypes>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from jax>=0.3.15->tensorflow==2.12) (
INFO: pip is looking at multiple versions of jax to determine which version is compatible with other requirements. This could tak
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.6.0-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.6.0,>=0.6.0 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.6.0-cp311-cp311-manylinux2014_x86_64.whl.metadata (1.2 kB)
Collecting ml_dtypes>=0.5.0 (from jax>=0.3.15->tensorflow==2.12)
  Downloading ml_dtypes-0.5.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (21 kB)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.5.3-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.5.3,>=0.5.3 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.5.3-cp311-cp311-manylinux2014_x86_64.whl.metadata (1.2 kB)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.5.1-py3-none-any.whl.metadata (22 kB)
  Downloading jax-0.5.0-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.5.0,>=0.5.0 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.5.0-cp311-cp311-manylinux2014_x86_64.whl.metadata (978 bytes)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.38-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.38,>=0.4.38 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.38-cp311-cp311-manylinux2014_x86_64.whl.metadata (1.0 kB)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.37-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.37,>=0.4.36 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.36-cp311-cp311-manylinux2014_x86_64.whl.metadata (1.0 kB)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.36-py3-none-any.whl.metadata (22 kB)
INFO: pip is still looking at multiple versions of jax to determine which version is compatible with other requirements. This cou
  Downloading jax-0.4.35-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.35,>=0.4.34 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.35-cp311-cp311-manylinux2014_x86_64.whl.metadata (983 bytes)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.34-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.34,>=0.4.34 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.34-cp311-cp311-manylinux2014_x86_64.whl.metadata (983 bytes)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.33-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.33,>=0.4.33 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.33-cp311-cp311-manylinux2014_x86_64.whl.metadata (983 bytes)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.31-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.31,>=0.4.30 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.31-cp311-cp311-manylinux2014_x86_64.whl.metadata (983 bytes)
Collecting jax>=0.3.15 (from tensorflow==2.12)
  Downloading jax-0.4.30-py3-none-any.whl.metadata (22 kB)
Collecting jaxlib<=0.4.30,>=0.4.27 (from jax>=0.3.15->tensorflow==2.12)
  Downloading jaxlib-0.4.30-cp311-cp311-manylinux2014_x86_64.whl.metadata (1.0 kB)
Requirement already satisfied: scipy>=1.9 in /usr/local/lib/python3.11/dist-packages (from jax>=0.3.15->tensorflow==2.12) (1.14.1
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.13,>=2.12->te
Collecting google-auth-oauthlib<1.1,>=0.5 (from tensorboard<2.13,>=2.12->tensorflow==2.12)
  Downloading google_auth_oauthlib-1.0.0-py2.py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.13,>=2.12->tensorfl
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.13,>=2.12->tens
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.13,>=2.12->tensorfl
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from google-auth<3,>=1.6.3->ten
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from google-auth<3,>=1.6.3->tens
```

```
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/dist-packages (from google-auth<3,>=1.6.3->tensorboard<
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from google-auth-oauthlib<1.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->ten
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorboard<2.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorboa
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorboa
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard<2.
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.11/dist-packages (from pyasn1-modules>=0.2.1->googl
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.11/dist-packages (from requests-oauthlib>=0.7.0->google-
Downloading tensorflow-2.12.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (586.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 586.0/586.0 MB 2.3 MB/s eta 0:00:00
Downloading gast-0.4.0-py3-none-any.whl (9.8 kB)
Downloading jax-0.4.30-py3-none-any.whl (2.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 80.3 MB/s eta 0:00:00
Downloading keras-2.12.0-py2.py3-none-any.whl (1.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 76.9 MB/s eta 0:00:00
Downloading numpy-1.23.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 103.8 MB/s eta 0:00:00
Downloading protobuf-4.25.6-cp37-abi3-manylinux2014_x86_64.whl (294 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.6/294.6 kB 22.6 MB/s eta 0:00:00
Downloading tensorboard-2.12.3-py3-none-any.whl (5.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.6/5.6 MB 108.3 MB/s eta 0:00:00
Downloading tensorflow_estimator-2.12.0-py2.py3-none-any.whl (440 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 440.7/440.7 kB 29.8 MB/s eta 0:00:00
Downloading wrapt-1.14.1-cp311-cp311-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (78 kB
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 78.4/78.4 kB 6.2 MB/s eta 0:00:00
Downloading google_auth_oauthlib-1.0.0-py2.py3-none-any.whl (18 kB)
Downloading jaxlib-0.4.30-cp311-cp311-manylinux2014_x86_64.whl (79.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.6/79.6 MB 28.6 MB/s eta 0:00:00
Installing collected packages: wrapt, tensorflow-estimator, protobuf, numpy, keras, gast, jaxlib, google-auth-oauthlib, tensorboa
  Attempting uninstall: wrapt
    Found existing installation: wrapt 1.17.2
    Uninstalling wrapt-1.17.2:
      Successfully uninstalled wrapt-1.17.2
  Attempting uninstall: protobuf
    Found existing installation: protobuf 5.29.4
    Uninstalling protobuf-5.29.4:
      Successfully uninstalled protobuf-5.29.4
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
  Attempting uninstall: keras
    Found existing installation: keras 3.8.0
    Uninstalling keras-3.8.0:
      Successfully uninstalled keras-3.8.0
  Attempting uninstall: gast
    Found existing installation: gast 0.6.0
    Uninstalling gast-0.6.0:
      Successfully uninstalled gast-0.6.0
  Attempting uninstall: jaxlib
    Found existing installation: jaxlib 0.5.1
    Uninstalling jaxlib-0.5.1:
      Successfully uninstalled jaxlib-0.5.1
  Attempting uninstall: google-auth-oauthlib
    Found existing installation: google-auth-oauthlib 1.2.1
    Uninstalling google-auth-oauthlib-1.2.1:
      Successfully uninstalled google-auth-oauthlib-1.2.1
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.18.0
    Uninstalling tensorboard-2.18.0:
      Successfully uninstalled tensorboard-2.18.0
  Attempting uninstall: jax
    Found existing installation: jax 0.5.2
    Uninstalling jax-0.5.2:
      Successfully uninstalled jax-0.5.2
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.18.0
    Uninstalling tensorflow-2.18.0:
      Successfully uninstalled tensorflow-2.18.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.23.5 which is incompatible.
tensorflow-text 2.18.1 requires tensorflow<2.19,>=2.18.0, but you have tensorflow 2.12.0 which is incompatible.
chex 0.1.89 requires numpy>=1.24.1, but you have numpy 1.23.5 which is incompatible.
grpcio-status 1.71.0 requires protobuf<6.0dev,>=5.26.1, but you have protobuf 4.25.6 which is incompatible.
tf-keras 2.18.0 requires tensorflow<2.19,>=2.18, but you have tensorflow 2.12.0 which is incompatible.
xarray 2025.1.2 requires numpy>=1.24, but you have numpy 1.23.5 which is incompatible.
bigframes 1.42.0 requires numpy>=1.24.0, but you have numpy 1.23.5 which is incompatible.
albumentations 2.0.5 requires numpy>=1.24.4, but you have numpy 1.23.5 which is incompatible.
scikit-image 0.25.2 requires numpy>=1.24, but you have numpy 1.23.5 which is incompatible.
orbax-checkpoint 0.11.12 requires jax>=0.5.0, but you have jax 0.4.30 which is incompatible.
albucore 0.0.23 requires numpy>=1.24.4, but you have numpy 1.23.5 which is incompatible.
blosc2 3.3.0 requires numpy>=1.26, but you have numpy 1.23.5 which is incompatible.
treescope 0.1.9 requires numpy>=1.25.2, but you have numpy 1.23.5 which is incompatible.
tensorflow-decision-forests 1.11.0 requires tensorflow==2.18.0, but you have tensorflow 2.12.0 which is incompatible.
imbalanced-learn 0.13.0 requires numpy<3,>=1.24.3, but you have numpy 1.23.5 which is incompatible.
ydf 0.11.0 requires protobuf<6.0.0,>=5.29.1, but you have protobuf 4.25.6 which is incompatible.
pymc 5.21.2 requires numpy>=1.25.0, but you have numpy 1.23.5 which is incompatible.
flax 0.10.5 requires jax>=0.5.1, but you have jax 0.4.30 which is incompatible.
Successfully installed gast-0.4.0 google-auth-oauthlib-1.0.0 jax-0.4.30 jaxlib-0.4.30 keras-2.12.0 numpy-1.23.5 protobuf-4.25.6 t
WARNING: The following packages were previously imported in this runtime:
```

```
WARNING: The following packages were previously imported in this runtime:
  [gast,jax,jaxlib,keras,numpy,tensorflow,wrapt]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

Loading the important libraries

```
import matplotlib.pyplot as plt
%matplotlib inline
import logging
logging.getLogger('tensorflow').disabled = True
```

Importing TensorFlow and Keras:

```
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from keras import preprocessing
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Flatten, Dense, Embedding, LSTM,  Conv1D, MaxPooling1D, GlobalMaxPooling1D, Dropout
from keras.models import load_model
from keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.optimizers import Adam
from google.colab import files
import re, os
```

Considering the IMDB example from Chapter 6. Re-running the example and modifying the by implementing a cutoff for reviews after 150 words, Validation Sample - 10000, Consider only the top 10,000 words

Model 1: Basic model just using embedded layer with Training Sample - 100Creating the training sample with 100 obs , validation with 10,000 obs and test with 5000 obs

```
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
import numpy as np

vocab_limit = 10000
sequence_length = 150

(raw_train_data, raw_train_labels), (raw_test_data, raw_test_labels) = imdb.load_data(num_words=vocab_limit)

padded_train_data = pad_sequences(raw_train_data, maxlen=sequence_length)
padded_test_data = pad_sequences(raw_test_data, maxlen=sequence_length)

combined_data = np.concatenate((padded_train_data, padded_test_data), axis=0)
combined_labels = np.concatenate((raw_train_labels, raw_test_labels), axis=0)

x_train_small, x_val_set, y_train_small, y_val_set = train_test_split(
    combined_data, combined_labels, train_size=100, test_size=10000, random_state=42, stratify=combined_labels)

_, x_test_final, _, y_test_final = train_test_split(
    padded_test_data, raw_test_labels, test_size=5000, random_state=42, stratify=raw_test_labels)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 1s 0us/step
```

```
x_train_small.shape
```

```
(100, 150)
```

```
x_val_set.shape
```

```
(10000, 150)
```

```
x_test_final.shape
```

```
(5000, 150)
```

Model Building :

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Flatten, Dense

simple_model_v1 = Sequential()

simple_model_v1.add(Embedding(input_dim=10000, output_dim=8, input_length=sequence_length))

simple_model_v1.add(Flatten())

simple_model_v1.add(Dense(units=1, activation='sigmoid'))

simple_model_v1.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

simple_model_v1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 150, 8)            80000

 flatten (Flatten)           (None, 1200)              0

 dense (Dense)               (None, 1)                 1201

=================================================================
Total params: 81,201
Trainable params: 81,201
Non-trainable params: 0
_________________________________________________________________
```

Model Execution

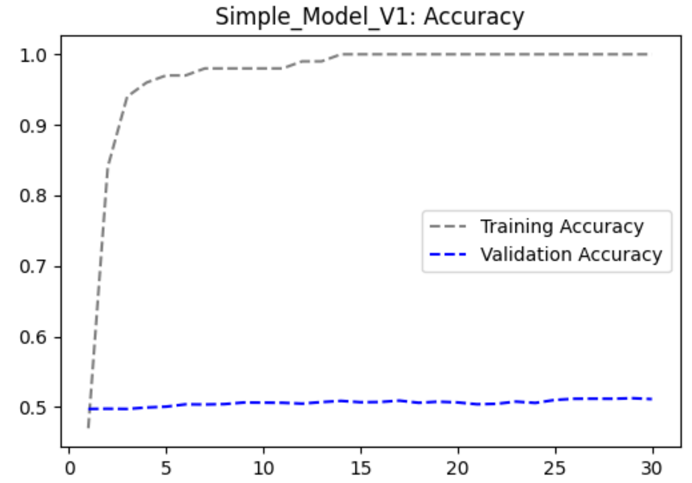
```
from tensorflow.keras.callbacks import ModelCheckpoint

model_checkpoint_v1 = ModelCheckpoint(
    filepath="simple_model_v1.h5",
    save_best_only=True,
    monitor="val_loss"
)

history_v1 = simple_model_v1.fit(
    x_train_small, y_train_small,
    epochs=30,
    batch_size=32,
    validation_data=(x_val_set, y_val_set),
    callbacks=[model_checkpoint_v1]
)
```

```
Epoch 2/30
4/4 [==============================] - 0s 131ms/step - loss: 0.6686 - accuracy: 0.8400 - val_loss: 0.6938 - val_accuracy: 0.4977
Epoch 3/30
4/4 [==============================] - 0s 130ms/step - loss: 0.6524 - accuracy: 0.9400 - val_loss: 0.6938 - val_accuracy: 0.4974
Epoch 4/30
4/4 [==============================] - 0s 135ms/step - loss: 0.6382 - accuracy: 0.9600 - val_loss: 0.6937 - val_accuracy: 0.4994
Epoch 5/30
4/4 [==============================] - 0s 129ms/step - loss: 0.6247 - accuracy: 0.9700 - val_loss: 0.6938 - val_accuracy: 0.5006
Epoch 6/30
4/4 [==============================] - 0s 128ms/step - loss: 0.6116 - accuracy: 0.9700 - val_loss: 0.6937 - val_accuracy: 0.5039
Epoch 7/30
4/4 [==============================] - 0s 129ms/step - loss: 0.5979 - accuracy: 0.9800 - val_loss: 0.6938 - val_accuracy: 0.5037
Epoch 8/30
4/4 [==============================] - 0s 136ms/step - loss: 0.5840 - accuracy: 0.9800 - val_loss: 0.6939 - val_accuracy: 0.5042
Epoch 9/30
4/4 [==============================] - 0s 134ms/step - loss: 0.5695 - accuracy: 0.9800 - val_loss: 0.6942 - val_accuracy: 0.5065
Epoch 10/30
4/4 [==============================] - 0s 134ms/step - loss: 0.5548 - accuracy: 0.9800 - val_loss: 0.6941 - val_accuracy: 0.5063
Epoch 11/30
4/4 [==============================] - 0s 138ms/step - loss: 0.5390 - accuracy: 0.9800 - val_loss: 0.6940 - val_accuracy: 0.5061
Epoch 12/30
4/4 [==============================] - 0s 130ms/step - loss: 0.5230 - accuracy: 0.9900 - val_loss: 0.6938 - val_accuracy: 0.5049
Epoch 13/30
4/4 [==============================] - 0s 139ms/step - loss: 0.5068 - accuracy: 0.9900 - val_loss: 0.6936 - val_accuracy: 0.5071
Epoch 14/30
4/4 [==============================] - 0s 137ms/step - loss: 0.4904 - accuracy: 1.0000 - val_loss: 0.6935 - val_accuracy: 0.5088
Epoch 15/30
```

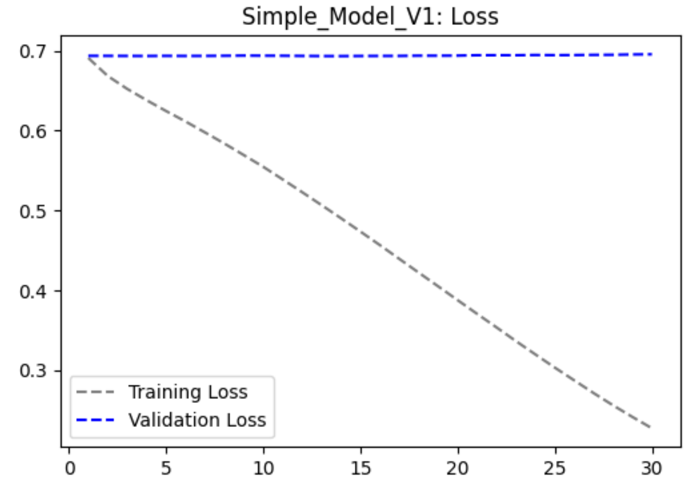
```
4/4 [==============================] - 0s 128ms/step - loss: 0.4567 - accuracy: 1.0000 - val_loss: 0.6937 - val_accuracy: 0.5072
Epoch 17/30
4/4 [==============================] - 0s 137ms/step - loss: 0.4394 - accuracy: 1.0000 - val_loss: 0.6938 - val_accuracy: 0.5092
Epoch 18/30
4/4 [==============================] - 0s 136ms/step - loss: 0.4222 - accuracy: 1.0000 - val_loss: 0.6941 - val_accuracy: 0.5061
Epoch 19/30
4/4 [==============================] - 0s 135ms/step - loss: 0.4050 - accuracy: 1.0000 - val_loss: 0.6940 - val_accuracy: 0.5077
Epoch 20/30
4/4 [==============================] - 0s 133ms/step - loss: 0.3877 - accuracy: 1.0000 - val_loss: 0.6942 - val_accuracy: 0.5066
Epoch 21/30
4/4 [==============================] - 0s 123ms/step - loss: 0.3706 - accuracy: 1.0000 - val_loss: 0.6946 - val_accuracy: 0.5040
Epoch 22/30
4/4 [==============================] - 0s 127ms/step - loss: 0.3536 - accuracy: 1.0000 - val_loss: 0.6947 - val_accuracy: 0.5046
Epoch 23/30
4/4 [==============================] - 0s 128ms/step - loss: 0.3364 - accuracy: 1.0000 - val_loss: 0.6946 - val_accuracy: 0.5078
Epoch 24/30
4/4 [==============================] - 0s 128ms/step - loss: 0.3197 - accuracy: 1.0000 - val_loss: 0.6949 - val_accuracy: 0.5059
Epoch 25/30
4/4 [==============================] - 0s 129ms/step - loss: 0.3032 - accuracy: 1.0000 - val_loss: 0.6947 - val_accuracy: 0.5101
Epoch 26/30
4/4 [==============================] - 0s 129ms/step - loss: 0.2872 - accuracy: 1.0000 - val_loss: 0.6948 - val_accuracy: 0.5118
Epoch 27/30
4/4 [==============================] - 0s 130ms/step - loss: 0.2712 - accuracy: 1.0000 - val_loss: 0.6951 - val_accuracy: 0.5119
Epoch 28/30
4/4 [==============================] - 0s 126ms/step - loss: 0.2558 - accuracy: 1.0000 - val_loss: 0.6952 - val_accuracy: 0.5118
Epoch 29/30
4/4 [==============================] - 0s 130ms/step - loss: 0.2412 - accuracy: 1.0000 - val_loss: 0.6956 - val_accuracy: 0.5126
Epoch 30/30
4/4 [==============================] - 0s 127ms/step - loss: 0.2271 - accuracy: 1.0000 - val_loss: 0.6958 - val_accuracy: 0.5113
```

Plotting the Accuracy and loss for training and validation

```
import matplotlib.pyplot as plt

train_acc_v1 = history_v1.history['accuracy']
val_acc_v1 = history_v1.history['val_accuracy']

train_loss_v1 = history_v1.history["loss"]
val_loss_v1 = history_v1.history["val_loss"]

epochs_v1 = range(1, len(train_acc_v1) + 1)

plt.figure(figsize=(6, 4))
plt.plot(epochs_v1, train_acc_v1, color="grey", linestyle="dashed", label="Training Accuracy")
plt.plot(epochs_v1, val_acc_v1, color="blue", linestyle="dashed", label="Validation Accuracy")
plt.title("Simple_Model_V1: Accuracy")
plt.legend()
plt.figure()

plt.figure(figsize=(6, 4))
plt.plot(epochs_v1, train_loss_v1, color="grey", linestyle="dashed", label="Training Loss")
plt.plot(epochs_v1, val_loss_v1, color="blue", linestyle="dashed", label="Validation Loss")
plt.title("Simple_Model_V1: Loss")
plt.legend()
plt.show()
```




```
<Figure size 640x480 with 0 Axes>
```




```
from tensorflow.keras.models import load_model

# Loading the saved model
loaded_model_v1 = load_model('simple_model_v1.h5')

# Evaluating the model on the test data
eval_results_v1 = loaded_model_v1.evaluate(x_test_final, y_test_final)

# Printing the results (Loss and Accuracy)
print(f'Loss: {eval_results_v1[0]:.3f}')
print(f'Accuracy: {eval_results_v1[1]:.3f}')
```

```
157/157 [==============================] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.5078
Loss: 0.693
Accuracy: 0.508
```

The model built with training sample of 100 and just one embedded layer gave the loss as 0.693 and accuracy of 0.512 that means the model has performed well for a training sample of 100. Now let us try building the model with the training sample of 5000.

Model 2: Basic model just using embedded layer with Training Sample - 5,000

```
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
import numpy as np

max_vocab = 10000
max_seq_len = 150

((raw_x_train, raw_y_train), (raw_x_test, raw_y_test)) = imdb.load_data(num_words=max_vocab)
padded_x_train = pad_sequences(raw_x_train, maxlen=max_seq_len)
padded_x_test = pad_sequences(raw_x_test, maxlen=max_seq_len)

merged_inputs = np.concatenate((padded_x_train, padded_x_test), axis=0)
```

```
merged_labels = np.concatenate((raw_y_train, raw_y_test), axis=0)

x_train_rnn, x_val_rnn, y_train_rnn, y_val_rnn = train_test_split(
    merged_inputs, merged_labels, train_size=5000, test_size=10000, random_state=42, stratify=merged_labels
)

_, x_test_rnn, _, y_test_rnn = train_test_split(
    padded_x_test, raw_y_test, test_size=5000, random_state=42, stratify=raw_y_test
)
```

```
x_train_rnn.shape
```

```
(5000, 150)
```

```
x_val_rnn.shape
```

```
(10000, 150)
```

```
x_test_rnn.shape
```

```
(5000, 150)
```

Model Building :

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Flatten, Dense

flat_model_rnn = Sequential()
flat_model_rnn.add(Embedding(input_dim=10000, output_dim=8, input_length=max_seq_len))
flat_model_rnn.add(Flatten())
flat_model_rnn.add(Dense(units=1, activation='sigmoid'))

flat_model_rnn.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

flat_model_rnn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 150, 8)            80000

 flatten_1 (Flatten)         (None, 1200)              0

 dense_1 (Dense)             (None, 1)                 1201

=================================================================
Total params: 81,201
Trainable params: 81,201
Non-trainable params: 0
_________________________________________________________________
```

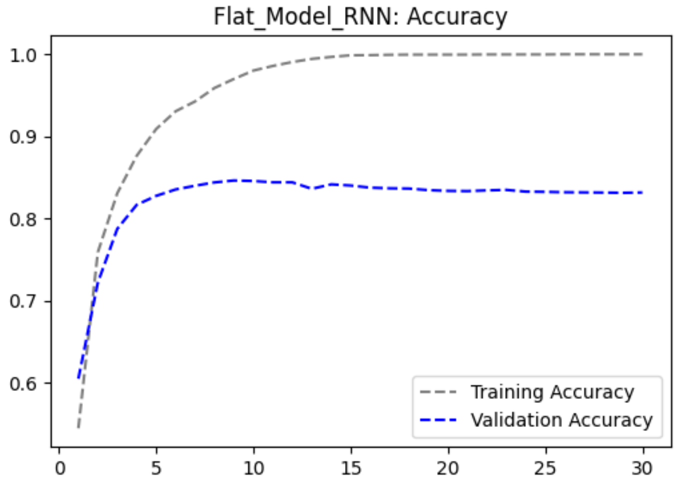
```
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint_rnn = ModelCheckpoint(
    filepath="flat_model_rnn.h5",
    save_best_only=True,
    monitor="val_loss"
)

history_rnn = flat_model_rnn.fit(
    x_train_rnn, y_train_rnn,
    epochs=30,
    batch_size=32,
    validation_data=(x_val_rnn, y_val_rnn),
    callbacks=[checkpoint_rnn]
)
```

```
Epoch 1/30
157/157 [==============================] - 1s 6ms/step - loss: 0.6903 - accuracy: 0.5446 - val_loss: 0.6840 - val_accuracy: 0.604
Epoch 2/30
157/157 [==============================] - 1s 5ms/step - loss: 0.6459 - accuracy: 0.7588 - val_loss: 0.6261 - val_accuracy: 0.721
Epoch 3/30
157/157 [==============================] - 1s 5ms/step - loss: 0.5362 - accuracy: 0.8308 - val_loss: 0.5246 - val_accuracy: 0.787
Epoch 4/30
157/157 [==============================] - 1s 5ms/step - loss: 0.4102 - accuracy: 0.8764 - val_loss: 0.4425 - val_accuracy: 0.816
Epoch 5/30
157/157 [==============================] - 1s 5ms/step - loss: 0.3147 - accuracy: 0.9090 - val_loss: 0.3983 - val_accuracy: 0.827
```

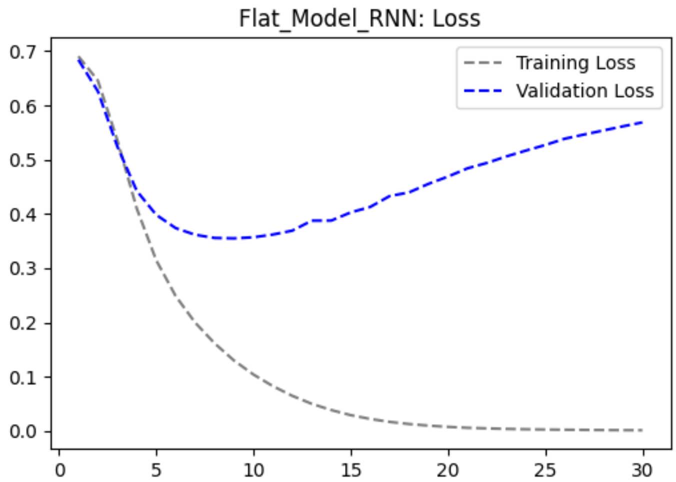
```
Epoch 6/30
157/157 [==============================] - 1s 5ms/step - loss: 0.2488 - accuracy: 0.9306 - val_loss: 0.3739 - val_accuracy: 0.835
Epoch 7/30
157/157 [==============================] - 1s 4ms/step - loss: 0.1998 - accuracy: 0.9426 - val_loss: 0.3615 - val_accuracy: 0.839
Epoch 8/30
157/157 [==============================] - 1s 5ms/step - loss: 0.1617 - accuracy: 0.9590 - val_loss: 0.3557 - val_accuracy: 0.844
Epoch 9/30
157/157 [==============================] - 1s 5ms/step - loss: 0.1299 - accuracy: 0.9700 - val_loss: 0.3549 - val_accuracy: 0.846
Epoch 10/30
157/157 [==============================] - 1s 5ms/step - loss: 0.1041 - accuracy: 0.9802 - val_loss: 0.3569 - val_accuracy: 0.845
Epoch 11/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0826 - accuracy: 0.9858 - val_loss: 0.3619 - val_accuracy: 0.844
Epoch 12/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0647 - accuracy: 0.9906 - val_loss: 0.3691 - val_accuracy: 0.844
Epoch 13/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0501 - accuracy: 0.9944 - val_loss: 0.3875 - val_accuracy: 0.836
Epoch 14/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0386 - accuracy: 0.9968 - val_loss: 0.3876 - val_accuracy: 0.841
Epoch 15/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0292 - accuracy: 0.9988 - val_loss: 0.4027 - val_accuracy: 0.840
Epoch 16/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0222 - accuracy: 0.9990 - val_loss: 0.4127 - val_accuracy: 0.837
Epoch 17/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0168 - accuracy: 0.9994 - val_loss: 0.4327 - val_accuracy: 0.836
Epoch 18/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0129 - accuracy: 0.9996 - val_loss: 0.4396 - val_accuracy: 0.836
Epoch 19/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0098 - accuracy: 0.9996 - val_loss: 0.4553 - val_accuracy: 0.834
Epoch 20/30
157/157 [==============================] - 1s 4ms/step - loss: 0.0076 - accuracy: 0.9996 - val_loss: 0.4687 - val_accuracy: 0.833
Epoch 21/30
157/157 [==============================] - 1s 4ms/step - loss: 0.0059 - accuracy: 0.9996 - val_loss: 0.4839 - val_accuracy: 0.833
Epoch 22/30
157/157 [==============================] - 1s 4ms/step - loss: 0.0048 - accuracy: 0.9998 - val_loss: 0.4938 - val_accuracy: 0.834
Epoch 23/30
157/157 [==============================] - 1s 4ms/step - loss: 0.0038 - accuracy: 0.9998 - val_loss: 0.5058 - val_accuracy: 0.834
Epoch 24/30
157/157 [==============================] - 1s 5ms/step - loss: 0.0032 - accuracy: 0.9998 - val_loss: 0.5165 - val_accuracy: 0.832
Epoch 25/30
157/157 [==============================] - 1s 4ms/step - loss: 0.0027 - accuracy: 0.9998 - val_loss: 0.5269 - val_accuracy: 0.832
Epoch 26/30
157/157 [==============================] - 1s 4ms/step - loss: 0.0023 - accuracy: 1.0000 - val_loss: 0.5384 - val_accuracy: 0.832
Epoch 27/30
157/157 [==============================] - 1s 4ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.5462 - val_accuracy: 0.831
Epoch 28/30
157/157 [==============================] - 1s 4ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.5537 - val_accuracy: 0.831
```

Plotting the Accuracy and loss for training and validation

```
import matplotlib.pyplot as plt

val_acc_rnn = history_rnn.history['val_accuracy']
train_loss_rnn = history_rnn.history['loss']
val_loss_rnn = history_rnn.history['val_loss']
train_acc_rnn = history_rnn.history['accuracy']

epochs_rnn = range(1, len(train_acc_rnn) + 1)

plt.figure(figsize=(6, 4))
plt.plot(epochs_rnn, train_acc_rnn, color="grey", linestyle="dashed", label="Training Accuracy")
plt.plot(epochs_rnn, val_acc_rnn, color="blue", linestyle="dashed", label="Validation Accuracy")
plt.title("Flat_Model_RNN: Accuracy")
plt.legend()
plt.figure()

plt.figure(figsize=(6, 4))
plt.plot(epochs_rnn, train_loss_rnn, color="grey", linestyle="dashed", label="Training Loss")
plt.plot(epochs_rnn, val_loss_rnn, color="blue", linestyle="dashed", label="Validation Loss")
plt.title("Flat_Model_RNN: Loss")
plt.legend()
plt.show()
```




```
<Figure size 640x480 with 0 Axes>
```




```
from tensorflow.keras.models import load_model

loaded_rnn_model = load_model('flat_model_rnn.h5')

eval_results_rnn = loaded_rnn_model.evaluate(x_test_rnn, y_test_rnn)
print(f'Loss: {eval_results_rnn[0]:.3f}')
print(f'Accuracy: {eval_results_rnn[1]:.3f}')
```

```
157/157 [==============================] - 1s 1ms/step - loss: 0.3169 - accuracy: 0.8622
Loss: 0.317
Accuracy: 0.862
```

Model 3: Basic model just using embedded layer with Training Sample - 10,000

```
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
import numpy as np

token_limit = 10000
seq_length = 150

((train_raw_seq, train_raw_lbl), (test_raw_seq, test_raw_lbl)) = imdb.load_data(num_words=token_limit)
train_seq_padded = pad_sequences(train_raw_seq, maxlen=seq_length)
test_seq_padded = pad_sequences(test_raw_seq, maxlen=seq_length)

all_padded_seq = np.concatenate((train_seq_padded, test_seq_padded), axis=0)
all_seq_labels = np.concatenate((train_raw_lbl, test_raw_lbl), axis=0)

x_train_seq, x_val_seq, y_train_seq, y_val_seq = train_test_split(
    all_padded_seq, all_seq_labels, train_size=10000, test_size=10000, random_state=42, stratify=all_seq_labels
)

_, x_test_seq, _, y_test_seq = train_test_split(
    test_seq_padded, test_raw_lbl, test_size=5000, random_state=42, stratify=test_raw_lbl
)
```

```
x_train_seq.shape
```

```
(10000, 150)
```

```
x_val_seq.shape
```

```
(10000, 150)
```

```
x_test_seq.shape
```

```
(5000, 150)
```

Model Building :

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Flatten, Dense

flat_model_seq = Sequential()
flat_model_seq.add(Embedding(input_dim=10000, output_dim=8, input_length=seq_length))
flat_model_seq.add(Flatten())
flat_model_seq.add(Dense(units=1, activation='sigmoid'))

flat_model_seq.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

flat_model_seq.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 150, 8)            80000

 flatten_2 (Flatten)         (None, 1200)              0

 dense_2 (Dense)             (None, 1)                 1201

=================================================================
Total params: 81,201
Trainable params: 81,201
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint_seq = ModelCheckpoint(
    filepath="flat_model_seq.h5",
    save_best_only=True,
    monitor="val_loss"
)

history_seq = flat_model_seq.fit(
    x_train_seq, y_train_seq,
    epochs=30,
    batch_size=32,
```

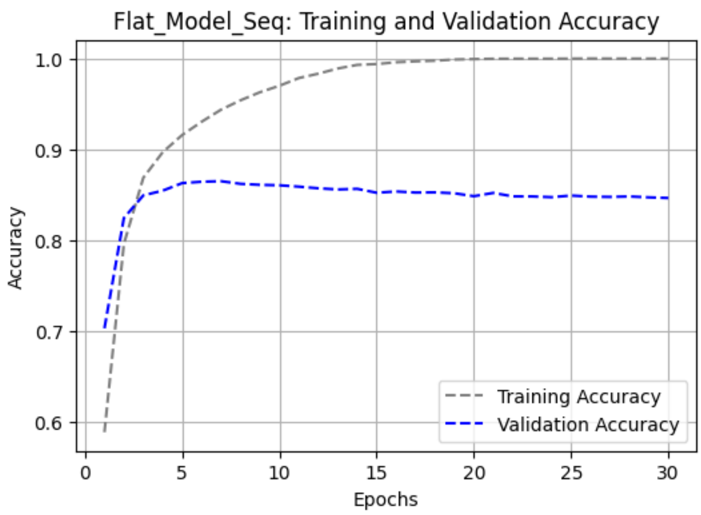
```
    validation_data=(x_val_seq, y_val_seq),
    callbacks=[checkpoint_seq]
)
```

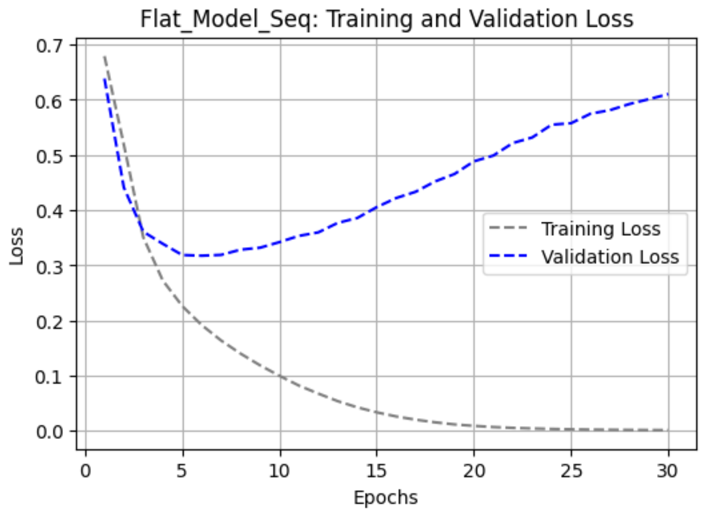
```
Epoch 1/30
313/313 [==============================] - 2s 4ms/step - loss: 0.6793 - accuracy: 0.5884 - val_loss: 0.6386 - val_accuracy: 0.702
Epoch 2/30
313/313 [==============================] - 1s 3ms/step - loss: 0.5184 - accuracy: 0.7957 - val_loss: 0.4410 - val_accuracy: 0.824
Epoch 3/30
313/313 [==============================] - 1s 3ms/step - loss: 0.3507 - accuracy: 0.8686 - val_loss: 0.3610 - val_accuracy: 0.849
Epoch 4/30
313/313 [==============================] - 1s 3ms/step - loss: 0.2727 - accuracy: 0.8968 - val_loss: 0.3391 - val_accuracy: 0.854
Epoch 5/30
313/313 [==============================] - 1s 3ms/step - loss: 0.2264 - accuracy: 0.9157 - val_loss: 0.3188 - val_accuracy: 0.862
Epoch 6/30
313/313 [==============================] - 1s 3ms/step - loss: 0.1920 - accuracy: 0.9303 - val_loss: 0.3175 - val_accuracy: 0.864
Epoch 7/30
313/313 [==============================] - 1s 3ms/step - loss: 0.1639 - accuracy: 0.9436 - val_loss: 0.3190 - val_accuracy: 0.864
Epoch 8/30
313/313 [==============================] - 1s 3ms/step - loss: 0.1401 - accuracy: 0.9539 - val_loss: 0.3285 - val_accuracy: 0.861
Epoch 9/30
313/313 [==============================] - 1s 3ms/step - loss: 0.1190 - accuracy: 0.9628 - val_loss: 0.3320 - val_accuracy: 0.860
Epoch 10/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0999 - accuracy: 0.9699 - val_loss: 0.3422 - val_accuracy: 0.860
Epoch 11/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0822 - accuracy: 0.9784 - val_loss: 0.3535 - val_accuracy: 0.858
Epoch 12/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0677 - accuracy: 0.9832 - val_loss: 0.3598 - val_accuracy: 0.857
Epoch 13/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0540 - accuracy: 0.9889 - val_loss: 0.3770 - val_accuracy: 0.855
Epoch 14/30
313/313 [==============================] - 1s 4ms/step - loss: 0.0428 - accuracy: 0.9929 - val_loss: 0.3858 - val_accuracy: 0.856
Epoch 15/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0338 - accuracy: 0.9938 - val_loss: 0.4055 - val_accuracy: 0.852
Epoch 16/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0263 - accuracy: 0.9958 - val_loss: 0.4220 - val_accuracy: 0.853
Epoch 17/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0204 - accuracy: 0.9968 - val_loss: 0.4333 - val_accuracy: 0.852
Epoch 18/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0156 - accuracy: 0.9975 - val_loss: 0.4520 - val_accuracy: 0.852
Epoch 19/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0118 - accuracy: 0.9988 - val_loss: 0.4655 - val_accuracy: 0.851
Epoch 20/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0092 - accuracy: 0.9993 - val_loss: 0.4885 - val_accuracy: 0.848
Epoch 21/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0070 - accuracy: 0.9996 - val_loss: 0.4990 - val_accuracy: 0.851
Epoch 22/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0056 - accuracy: 0.9997 - val_loss: 0.5216 - val_accuracy: 0.848
Epoch 23/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0045 - accuracy: 0.9997 - val_loss: 0.5318 - val_accuracy: 0.848
Epoch 24/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0036 - accuracy: 0.9997 - val_loss: 0.5550 - val_accuracy: 0.847
Epoch 25/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0030 - accuracy: 0.9998 - val_loss: 0.5576 - val_accuracy: 0.849
Epoch 26/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0025 - accuracy: 0.9999 - val_loss: 0.5749 - val_accuracy: 0.847
Epoch 27/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0022 - accuracy: 0.9998 - val_loss: 0.5813 - val_accuracy: 0.847
Epoch 28/30
313/313 [==============================] - 1s 3ms/step - loss: 0.0018 - accuracy: 0.9998 - val_loss: 0.5924 - val_accuracy: 0.848
Epoch 29/30
```

```
####Plotting the Accuracy and loss for training and validation
```

```
import matplotlib.pyplot as plt

train_acc_seq = history_seq.history['accuracy']
val_acc_seq = history_seq.history['val_accuracy']

train_loss_seq = history_seq.history["loss"]
val_loss_seq = history_seq.history["val_loss"]

epochs_seq = range(1, len(train_acc_seq) + 1)

plt.figure(figsize=(6, 4))
plt.plot(epochs_seq, train_acc_seq, color="grey", linestyle="dashed", label="Training Accuracy")
plt.plot(epochs_seq, val_acc_seq, color="blue", linestyle="dashed", label="Validation Accuracy")
plt.title("Flat_Model_Seq: Training and Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.grid()
plt.show()
```

```
plt.figure(figsize=(6, 4))
plt.plot(epochs_seq, train_loss_seq, color="grey", linestyle="dashed", label="Training Loss")
plt.plot(epochs_seq, val_loss_seq, color="blue", linestyle="dashed", label="Validation Loss")
plt.title("Flat_Model_Seq: Training and Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.grid()
plt.show()
```






```
from tensorflow.keras.models import load_model

loaded_seq_model = load_model('flat_model_seq.h5')

eval_results_seq = loaded_seq_model.evaluate(x_test_seq, y_test_seq)

print(f"Test Loss: {eval_results_seq[0]:.3f}")
print(f"Test Accuracy: {eval_results_seq[1]:.3f}")
```

```
157/157 [==============================] - 0s 1ms/step - loss: 0.2850 - accuracy: 0.8764
Test Loss: 0.285
Test Accuracy: 0.876
```

PreTrained Models

GloVe, or Global Vectors for Word Representation, is an unsupervised learning algorithm for generating vector representations of words based on their co-occurrence statistics in large text corpora. Developed by researchers at Stanford University, GloVe aims to capture semantic relationships and meanings of words by considering their global statistical information. We are getting the data from ai.stanford.edu.

```
!curl -O https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
!tar -xf aclImdb_v1.tar.gz
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 80.2M  100 80.2M    0     0  7400k      0  0:00:11  0:00:11 --:--:-- 14.7M
```

```
!rm -r aclImdb/train/unsup
```

```
import os

dataset_path = '/content/aclImdb'
train_path = os.path.join(dataset_path, 'train')

review_labels = []
review_texts = []

for sentiment in ['neg', 'pos']:
    sentiment_dir = os.path.join(train_path, sentiment)
    for filename in os.listdir(sentiment_dir):
        if filename.endswith('.txt'):
            with open(os.path.join(sentiment_dir, filename)) as file:
                review_texts.append(file.read())
            review_labels.append(0 if sentiment == 'neg' else 1)
```

```
print('No. of Samples', len(review_texts))
```

```
No. of Samples 25000
```

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

seq_maxlen = 150
n_train_samples = 100
n_val_samples = 10000
vocab_size_limit = 10000

text_tokenizer = Tokenizer(num_words=vocab_size_limit)
text_tokenizer.fit_on_texts(review_texts)
review_sequences = text_tokenizer.texts_to_sequences(review_texts)

word_to_index = text_tokenizer.word_index
print('Found %s unique tokens.' % len(word_to_index))

padded_data = pad_sequences(review_sequences, maxlen=seq_maxlen)
label_array = np.asarray(review_labels)
print('Shape of data tensor:', padded_data.shape)
print('Shape of label tensor:', label_array.shape)

shuffle_indices = np.arange(padded_data.shape[0])
np.random.shuffle(shuffle_indices)
padded_data = padded_data[shuffle_indices]
label_array = label_array[shuffle_indices]

x_train_text = padded_data[:n_train_samples]
y_train_text = label_array[:n_train_samples]
x_val_text = padded_data[n_train_samples: n_train_samples + n_val_samples]
y_val_text = label_array[n_train_samples: n_train_samples + n_val_samples]

# Test data loading
test_path = os.path.join(dataset_path, 'test')

test_labels = []
test_texts = []

for sentiment_type in ['neg', 'pos']:
    sentiment_path = os.path.join(test_path, sentiment_type)
    for test_file in sorted(os.listdir(sentiment_path)):
        if test_file.endswith('.txt'):
            with open(os.path.join(sentiment_path, test_file)) as file:
                test_texts.append(file.read())
            test_labels.append(0 if sentiment_type == 'neg' else 1)

test_sequences = text_tokenizer.texts_to_sequences(test_texts)
x_test_text = pad_sequences(test_sequences, maxlen=seq_maxlen)[:5000]
y_test_text = np.asarray(test_labels)[:5000]
```

```
Found 88582 unique tokens.
Shape of data tensor: (25000, 150)
Shape of label tensor: (25000,)
```

```
x_train_text.shape
```

```
(100, 150)
```

```
x_val_text.shape
```

```
(10000, 150)
```

```
x_test_text.shape
```

```
(5000, 150)
```

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
!unzip -q glove.6B.zip
```

```
--2025-04-20 23:48:24--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2025-04-20 23:48:25--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2025-04-20 23:48:25--  https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  5.02MB/s    in 2m 39s

2025-04-20 23:51:05 (5.17 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

```
import numpy as np

glove_path = "glove.6B.100d.txt"

glove_embeddings = {}
with open(glove_path) as glove_file:
    for line in glove_file:
        token, vector = line.split(maxsplit=1)
        vector = np.fromstring(vector, dtype="f", sep=" ")
        glove_embeddings[token] = vector

print(f"Found {len(glove_embeddings)} word vectors.")
```

```
Found 400000 word vectors.
```

```
embedding_size = 100

embedding_weights = np.zeros((vocab_size_limit, embedding_size))
for token, index in word_to_index.items():
    vector = glove_embeddings.get(token)
    if index < vocab_size_limit:
        if vector is not None:
            embedding_weights[index] = vector
```

Model 9:Pretrained Models with Training sample size 100- we are using GloVe model

Model Building :

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow import keras

lstm_model_embed = Sequential()
lstm_model_embed.add(Embedding(vocab_size_limit, embedding_size, input_length=seq_maxlen))
lstm_model_embed.add(LSTM(32))
lstm_model_embed.add(Dense(1, activation='sigmoid'))

lstm_model_embed.layers[0].set_weights([embedding_weights])
lstm_model_embed.layers[0].trainable = False

custom_adam = keras.optimizers.Adam(learning_rate=0.0001)
```

```
lstm_model_embed.compile(optimizer=custom_adam, loss='binary_crossentropy', metrics=['accuracy'])
lstm_model_embed.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 150, 100)          1000000

 lstm (LSTM)                 (None, 32)                17024

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 1,017,057
Trainable params: 17,057
Non-trainable params: 1,000,000
_________________________________________________________________
```

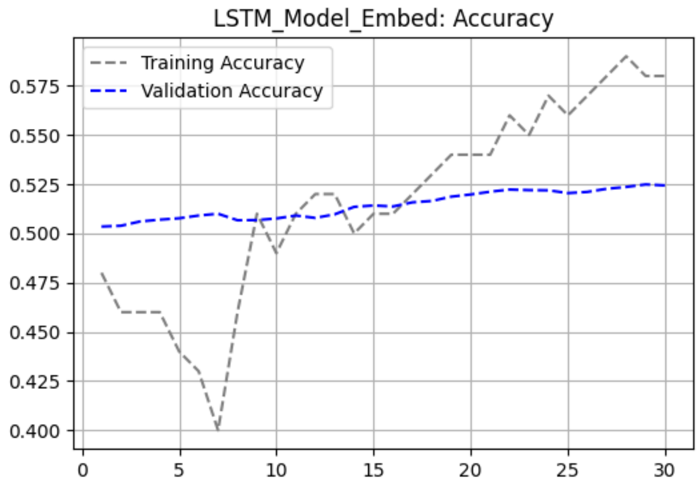
```
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint_embed_lstm = ModelCheckpoint(
    filepath="lstm_model_embed.keras",
    save_best_only=True,
    monitor="val_loss"
)

history_embed_lstm = lstm_model_embed.fit(
    x_train_text, y_train_text,
    epochs=30,
    batch_size=32,
    validation_data=(x_val_text, y_val_text),
    callbacks=[checkpoint_embed_lstm]
)
```

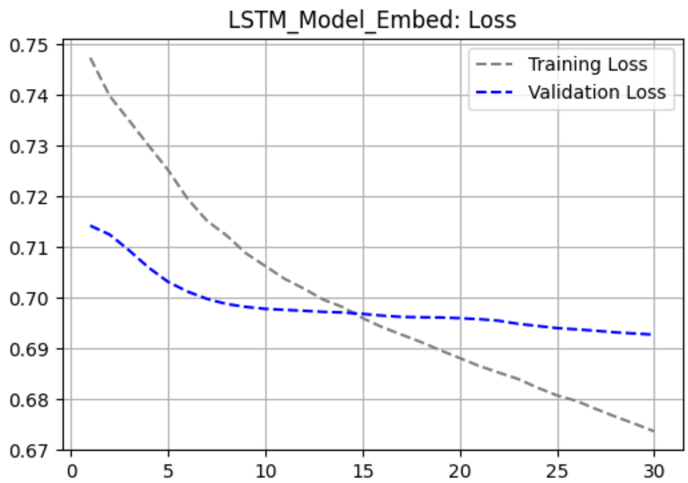
```
Epoch 1/30
4/4 [==============================] - 8s 2s/step - loss: 0.7473 - accuracy: 0.4800 - val_loss: 0.7142 - val_accuracy: 0.5035
Epoch 2/30
4/4 [==============================] - 5s 2s/step - loss: 0.7398 - accuracy: 0.4600 - val_loss: 0.7124 - val_accuracy: 0.5039
Epoch 3/30
4/4 [==============================] - 5s 2s/step - loss: 0.7349 - accuracy: 0.4600 - val_loss: 0.7093 - val_accuracy: 0.5061
Epoch 4/30
4/4 [==============================] - 5s 2s/step - loss: 0.7300 - accuracy: 0.4600 - val_loss: 0.7059 - val_accuracy: 0.5070
Epoch 5/30
4/4 [==============================] - 5s 2s/step - loss: 0.7252 - accuracy: 0.4400 - val_loss: 0.7031 - val_accuracy: 0.5077
Epoch 6/30
4/4 [==============================] - 5s 2s/step - loss: 0.7195 - accuracy: 0.4300 - val_loss: 0.7012 - val_accuracy: 0.5091
Epoch 7/30
4/4 [==============================] - 5s 2s/step - loss: 0.7152 - accuracy: 0.4000 - val_loss: 0.6998 - val_accuracy: 0.5100
Epoch 8/30
4/4 [==============================] - 5s 2s/step - loss: 0.7123 - accuracy: 0.4600 - val_loss: 0.6988 - val_accuracy: 0.5067
Epoch 9/30
4/4 [==============================] - 5s 2s/step - loss: 0.7088 - accuracy: 0.5100 - val_loss: 0.6982 - val_accuracy: 0.5068
Epoch 10/30
4/4 [==============================] - 5s 2s/step - loss: 0.7062 - accuracy: 0.4900 - val_loss: 0.6978 - val_accuracy: 0.5076
Epoch 11/30
4/4 [==============================] - 5s 2s/step - loss: 0.7037 - accuracy: 0.5100 - val_loss: 0.6976 - val_accuracy: 0.5091
Epoch 12/30
4/4 [==============================] - 5s 2s/step - loss: 0.7017 - accuracy: 0.5200 - val_loss: 0.6974 - val_accuracy: 0.5079
Epoch 13/30
4/4 [==============================] - 5s 2s/step - loss: 0.6996 - accuracy: 0.5200 - val_loss: 0.6972 - val_accuracy: 0.5097
Epoch 14/30
4/4 [==============================] - 5s 2s/step - loss: 0.6981 - accuracy: 0.5000 - val_loss: 0.6971 - val_accuracy: 0.5135
Epoch 15/30
4/4 [==============================] - 5s 2s/step - loss: 0.6960 - accuracy: 0.5100 - val_loss: 0.6968 - val_accuracy: 0.5142
Epoch 16/30
4/4 [==============================] - 5s 2s/step - loss: 0.6942 - accuracy: 0.5100 - val_loss: 0.6965 - val_accuracy: 0.5137
Epoch 17/30
4/4 [==============================] - 5s 2s/step - loss: 0.6927 - accuracy: 0.5200 - val_loss: 0.6962 - val_accuracy: 0.5158
Epoch 18/30
4/4 [==============================] - 5s 2s/step - loss: 0.6913 - accuracy: 0.5300 - val_loss: 0.6961 - val_accuracy: 0.5165
Epoch 19/30
4/4 [==============================] - 5s 2s/step - loss: 0.6896 - accuracy: 0.5400 - val_loss: 0.6961 - val_accuracy: 0.5187
Epoch 20/30
4/4 [==============================] - 5s 2s/step - loss: 0.6881 - accuracy: 0.5400 - val_loss: 0.6960 - val_accuracy: 0.5198
Epoch 21/30
4/4 [==============================] - 5s 2s/step - loss: 0.6865 - accuracy: 0.5400 - val_loss: 0.6958 - val_accuracy: 0.5212
Epoch 22/30
4/4 [==============================] - 5s 2s/step - loss: 0.6852 - accuracy: 0.5600 - val_loss: 0.6955 - val_accuracy: 0.5223
Epoch 23/30
4/4 [==============================] - 5s 2s/step - loss: 0.6840 - accuracy: 0.5500 - val_loss: 0.6948 - val_accuracy: 0.5220
Epoch 24/30
4/4 [==============================] - 5s 2s/step - loss: 0.6823 - accuracy: 0.5700 - val_loss: 0.6944 - val_accuracy: 0.5219
Epoch 25/30
4/4 [==============================] - 5s 2s/step - loss: 0.6807 - accuracy: 0.5600 - val_loss: 0.6940 - val_accuracy: 0.5205
Epoch 26/30
```

```
4/4 [==============================] - 5s 2s/step - loss: 0.6796 - accuracy: 0.5700 - val_loss: 0.6938 - val_accuracy: 0.5211
Epoch 27/30
4/4 [==============================] - 5s 2s/step - loss: 0.6781 - accuracy: 0.5800 - val_loss: 0.6935 - val_accuracy: 0.5227
Epoch 28/30
4/4 [==============================] - 5s 2s/step - loss: 0.6765 - accuracy: 0.5900 - val_loss: 0.6932 - val_accuracy: 0.5236
Epoch 29/30
```

Plotting the Accuracy and loss for training and validation

```
train_acc_lstm = history_embed_lstm.history['accuracy']
val_acc_lstm = history_embed_lstm.history['val_accuracy']

train_loss_lstm = history_embed_lstm.history["loss"]
val_loss_lstm = history_embed_lstm.history["val_loss"]

epoch_range_lstm = range(1, len(train_acc_lstm) + 1)

plt.figure(figsize=(6, 4))
plt.plot(epoch_range_lstm, train_acc_lstm, color="grey", linestyle="dashed", label="Training Accuracy")
plt.plot(epoch_range_lstm, val_acc_lstm, color="blue", linestyle="dashed", label="Validation Accuracy")
plt.title("LSTM_Model_Embed: Accuracy")
plt.legend()
plt.grid()
plt.show()

plt.figure(figsize=(6, 4))
plt.plot(epoch_range_lstm, train_loss_lstm, color="grey", linestyle="dashed", label="Training Loss")
plt.plot(epoch_range_lstm, val_loss_lstm, color="blue", linestyle="dashed", label="Validation Loss")
plt.title("LSTM_Model_Embed: Loss")
plt.legend()
plt.grid()
plt.show()
```






```
loaded_lstm_model = load_model('lstm_model_embed.keras')
eval_results_lstm = loaded_lstm_model.evaluate(x_test_text, y_test_text)

print(f'Loss: {eval_results_lstm[0]:.3f}')
print(f'Accuracy: {eval_results_lstm[1]:.3f}')
```

```
157/157 [==============================] - 3s 16ms/step - loss: 0.7025 - accuracy: 0.4706
Loss: 0.703
Accuracy: 0.471
```

Model 10:Pretrained Models , 4 LSTM hidden layers with Training sample size 5000

```
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Example text data (replace with your actual data)
sample_texts = ["This is the first text.", "Another example text.", "Text data for training."]
sample_labels = [0, 1, 0]  # Example labels (replace with your actual labels)

# Parameters
max_sequence_length = 150
num_train_samples = 5000
num_val_samples = 10000
vocab_limit = 10000

# Tokenizer for text preprocessing
text_tokenizer = Tokenizer(num_words=vocab_limit)
text_tokenizer.fit_on_texts(sample_texts)
text_sequences = text_tokenizer.texts_to_sequences(sample_texts)

# Get word index
token_to_index = text_tokenizer.word_index
print(f'Found {len(token_to_index)} unique tokens.')

# Pad sequences to ensure uniform input length
padded_text_data = pad_sequences(text_sequences, maxlen=max_sequence_length)

# Convert labels to numpy array
label_array = np.asarray(sample_labels)
print('Shape of data tensor:', padded_text_data.shape)
print('Shape of label tensor:', label_array.shape)

# Shuffle the data and labels
shuffle_idx = np.arange(padded_text_data.shape[0])
np.random.shuffle(shuffle_idx)
padded_text_data = padded_text_data[shuffle_idx]
label_array = label_array[shuffle_idx]

# Split into training and validation sets
x_train_sample = padded_text_data[:num_train_samples]
y_train_sample = label_array[:num_train_samples]
x_val_sample = padded_text_data[num_train_samples:num_train_samples + num_val_samples]
y_val_sample = label_array[num_train_samples:num_train_samples + num_val_samples]
```

```
Found 10 unique tokens.
Shape of data tensor: (3, 150)
Shape of label tensor: (3,)
```

```
x_train_sample.shape
```

```
(3, 150)
```

```
x_val_sample.shape
```

```
(0, 150)
```

Model Building :

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dropout, Dense
from tensorflow.keras import optimizers

vocab_size = 10000
embed_dim = 150
sequence_length = 150

pretrained_embed_matrix = np.random.rand(vocab_size, embed_dim)

deep_lstm_model = Sequential()
deep_lstm_model.add(Embedding(vocab_size, embed_dim, input_length=sequence_length))

deep_lstm_model.add(LSTM(512, return_sequences=True))
```

```
deep_lstm_model.add(Dropout(0.5))

deep_lstm_model.add(LSTM(256, return_sequences=True))
deep_lstm_model.add(Dropout(0.5))

deep_lstm_model.add(LSTM(128, return_sequences=True))
deep_lstm_model.add(Dropout(0.5))

deep_lstm_model.add(LSTM(128))

deep_lstm_model.add(Dense(256, activation='relu'))
deep_lstm_model.add(Dropout(0.5))
deep_lstm_model.add(Dense(256, activation='relu'))
deep_lstm_model.add(Dropout(0.5))
deep_lstm_model.add(Dense(1, activation='sigmoid'))

deep_lstm_model.layers[0].set_weights([pretrained_embed_matrix])
deep_lstm_model.layers[0].trainable = False

custom_adam_opt = optimizers.Adam(learning_rate=0.0001)
deep_lstm_model.compile(optimizer=custom_adam_opt, loss='binary_crossentropy', metrics=['accuracy'])

deep_lstm_model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 150, 150)          1500000

 lstm_1 (LSTM)               (None, 150, 512)          1357824

 dropout (Dropout)           (None, 150, 512)          0

 lstm_2 (LSTM)               (None, 150, 256)          787456

 dropout_1 (Dropout)         (None, 150, 256)          0

 lstm_3 (LSTM)               (None, 150, 128)          197120

 dropout_2 (Dropout)         (None, 150, 128)          0

 lstm_4 (LSTM)               (None, 128)               131584

 dense_4 (Dense)             (None, 256)               33024

 dropout_3 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 256)               65792

 dropout_4 (Dropout)         (None, 256)               0

 dense_6 (Dense)             (None, 1)                 257

=================================================================
Total params: 4,073,057
Trainable params: 2,573,057
Non-trainable params: 1,500,000
_________________________________________________________________
```

```
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint_deep_lstm = ModelCheckpoint(
    filepath="deep_lstm_model.keras",
    save_best_only=True,
    monitor="val_loss"
)

history_deep_lstm = deep_lstm_model.fit(
    x_train_sample, y_train_sample,
    epochs=10,
    batch_size=12,
    validation_data=(x_val_sample, y_val_sample),
    callbacks=[checkpoint_deep_lstm]
)
```

```
Epoch 1/10
1/1 [==============================] - 10s 10s/step - loss: 0.7257 - accuracy: 0.0000e+00
Epoch 2/10
1/1 [==============================] - 1s 611ms/step - loss: 0.7159 - accuracy: 0.3333
Epoch 3/10
1/1 [==============================] - 1s 620ms/step - loss: 0.6704 - accuracy: 1.0000
Epoch 4/10
1/1 [==============================] - 1s 606ms/step - loss: 0.6963 - accuracy: 0.6667
```