PRABAL GHOSH ROLL-40 NLP PROJECT

```
!nvidia-smi
```

```
Fri Nov 12 18:42:48 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   47C    P8    29W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

```
import numpy as np
import pandas as pd
```

Saving...

```
import tensorflow_datasets as tfds

import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12, 8)
from  IPython import display

import pathlib
```

```
import shutil
import tempfile

!pip install -q git+https://github.com/tensorflow/docs

import tensorflow_docs as tfdocs
import tensorflow_docs.modeling
import tensorflow_docs.plots

print("Version: ", tf.__version__)
print("Hub version: ", hub.__version__)
print("GPU is", "available" if tf.config.list_physical_devices('GPU') else "NOT AVAILABLE")

logdir = pathlib.Path(tempfile.mkdtemp())/"tensorboard_logs"
shutil.rmtree(logdir, ignore_errors=True)
```

```
  Building wheel for tensorflow-docs (setup.py) ... done
Version:  2.7.0
Hub version:  0.12.0
GPU is available
```

Task 3: Download and Import the Quora Insincere Questions Dataset

A downloadable copy of the Quora Insincere Questions Classification data can be found https://archive.org/download/fine-tune-bert-

Saving...

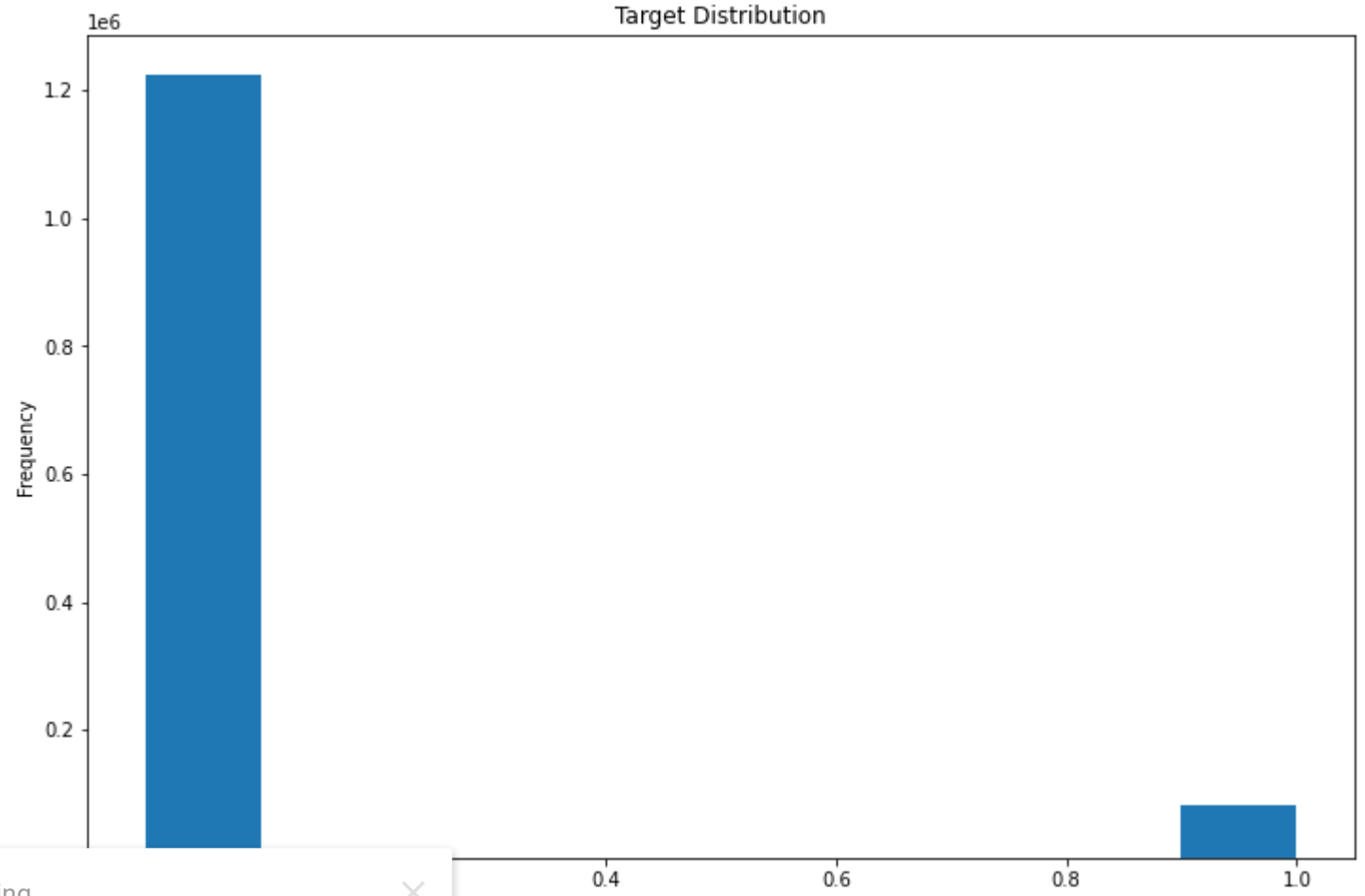
mpress and read the data into a pandas DataFrame.

```
df=pd.read_csv('https://archive.org/download/fine-tune-bert-tensorflow-train.csv/train.csv.zip',compression='zip',low_memory=
df.shape
# here compress='zip' is used to unzip the ziped file
```

```
(1306122, 3)
```

the data has 1.3 Milion (1306122) entries or rows

```
df['target'].plot(kind='hist',title='Target Distribution');
```



from this histogram we can find that majority of the questions are non toxic. Here for non toxic data the label is 1 and for toxic data classification label is 0

```
from sklearn.model_selection import train_test_split
```

```
train_df,remaining=train_test_split(df,random_state=42,train_size=0.01,stratify=df.target.values)   # here we take small por
validation_df,_=train_test_split(remaining,random_state=42,train_size=0.001,stratify=remaining.target.values)
train_df.shape,validation_df.shape
```

```
((13061, 3), (1293, 3))
```

```
train_df.target.head(15).values
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0])
```

Here 11th target is toxic as it is labeled as 1 out of 1st 15 questions.

```
train_df.question_text.head(15).values
```

```
array(['What is your experience living in Venezuela in the current crisis? (2018)',
       'In which state/city the price of property is highest?',
       'Do rich blacks also call poor whites, “White Trash”?',
       'Should my 5 yr old son and 2 yr old daughter spend the summer with their father, after a domestic violent relat
       'Why do we have parents?',
       'Do we experience ghost like Murphy did in Interstellar?',
       'Are Estoniano women beautiful?',
       'There was a Funny or Die video called Sensitivity Hoedown that got pulled. Does anyone know why?',
       'Is it a good idea to go in fully mainstream classes, even if I have meltdowns that might disrupt people?',
       'What classifies a third world country as such?',
       'Is being a pilot safe?',
       'Who is Illiteratendra Modi? Why does he keep with him a Rs 1 lakh pen?',
       'Have modern management strategies such as Total supply Chain Management applied to education? Can they be?',
       'Why are Lucky Charms considered good for you?',
       'How many people in India use WhatsApp, Facebook, Twitter and Instagram?'],
```


Here 11th question is ....""Who is Illiteratendra Modi? Why does he keep with him a Rs 1 lakh pen? "" ...which is toxic...as in ths question they try to disrespect our priminister Modi.

Task 4: TensorFlow Hub for Natural Language Processing

Our text data consits of questions and corresponding labels.

You can think of a question vector as a distributed representation of a question, and is computed for every question in the training set. The question vector along with the output label is then used to train the statistical classification model.

The intuition is that the question vector captures the semantics of the question and, as a result, can be effectively used for classification.

To obtain question vectors, we have two alternatives that have been used for several text classification problems in NLP:

word-based representations and context-based representations

Word-based Representations A word-based representation of a question combines word embeddings of the content words in the question. We can use the average of the word embeddings of content words in the question. Average of word embeddings have been used for different NLP tasks.

Examples of pre-trained embeddings include: Word2Vec: These are pre-trained embeddings of words learned from a large text corpora. Word2Vec has been pre-trained on a corpus of news articles with 300 million tokens, resulting in 300-dimensional vectors. GloVe: has been pre-trained on a corpus of tweets with 27 billion tokens, resulting in 200-dimensional vectors.

Context-based Representations Context-based representations may use language models to generate vectors of sentences. So, instead of learning vectors for individual words in the sentence, they compute a vector for sentences on the whole, by taking into account the order of words and the set of co-occurring words.

Examples of deep contextualised vectors include:

Embeddings from Language Models (ELMo): uses character-based word representations and bidirectional LSTMs. The pre-trained or of 1024 dimensions. ELMo is available on Tensorflow Hub.

Saving...

encoder uses a Transformer architecture that uses attention mechanism to incorporate information about the order and the collection of words. The pre-trained model of USE that returns a vector of 512 dimensions is also available on Tensorflow Hub.

Neural-Net Language Model (NNLM): The model simultaneously learns representations of words and probability functions for word sequences, allowing it to capture semantics of a sentence. We will use a pretrained models available on Tensorflow Hub, that are

trained on the English Google News 200B corpus, and computes a vector of 128 dimensions for the larger model and 50 dimensions for the smaller model.

Tensorflow Hub provides a number of modules to convert sentences into embeddings such as Universal sentence ecoders, NNLM, BERT and Wikiwords.

Transfer learning makes it possible to save training resources and to achieve good model generalization even when training on a small dataset. In this project, we will demonstrate this by training with several different TF-Hub modules.

```
module_url = "https://tfhub.dev/google/tf2-preview/gnews-swivel-20dim/1" #@param ["https://tfhub.dev/google/tf2-pre  w/gnews
```

module_url: https://tfhub.dev/google/tf2-preview/gn

here we take ""gnews-swivel-20dim small"" module with 20 dimension embedding vector. for that we use the url for that module

Tasks 5 & 6: Define Function to Build and Compile Models

```
def train_and_evaluate_model(module_url, embed_size, name, trainable=False):
  hub_layer=hub.KerasLayer(module_url,input_shape=[],output_shape=[embed_size],dtype=tf.string,trainable=trainable)
  model=tf.keras.models.Sequential([hub_layer, tf.keras.layers.Dense(256,activation='relu'),tf.keras.layers.Dense(64,activat
  model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),loss=tf.losses.BinaryCrossentropy(),metrics=[tf.met
  model.summary()
  history=model.fit(train_df['question_text'],train_df['target'],epochs=100,batch_size=32,
                    validation_data=(validation_df['question_text'], validation_df['target']),
                    callbacks=[tfdocs.modeling.EpochDots(),
                               tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=2,mode='min'),
                               tf.keras.callbacks.TensorBoard(logdir/name)],
                    verbose=0)
  return history
  # here trainable is False means we will stop to Fine tune the hidden layers of the pre trained model.
  # we use early stopping at 100 epochs
```

Saving...

this ""gnews-swivel-20dim small"" module is loadedinto the Keras layer using the tensorflow hub

Task 7: Train Various Text Classification Models

```
histories={}


# module_url = "https://tfhub.dev/google/tf2-preview/gnews-swivel-20dim/1" #@param ["https://tfhub.dev/google/tf2-preview/gn
# histories['gnews-swivel-20dim']=train_and_evaluate_model(module_url,embed_size=20,name='gnews-swivel-20dim',trainable=Fals


# module_url = "https://tfhub.dev/google/tf2-preview/nnlm-en-dim50/1" #@param ["https://tfhub.dev/google/tf2-preview/gnews-s
# histories['nnlm-en-dim50']=train_and_evaluate_model(module_url,embed_size=50,name='nnlm-en-dim50',trainable=False)
```

Double-click (or enter) to edit

```
module_url = "https://tfhub.dev/google/tf2-preview/nnlm-en-dim128/1" #@param ["https://tfhub.dev/google/tf2-preview/gnews-sw
```

module_url: https://tfhub.dev/google/tf2-preview/nnl

Saving...

```
d_evaluate_model(module_url,embed_size=128,name='nnlm-en-dim128',trainable=False)
```

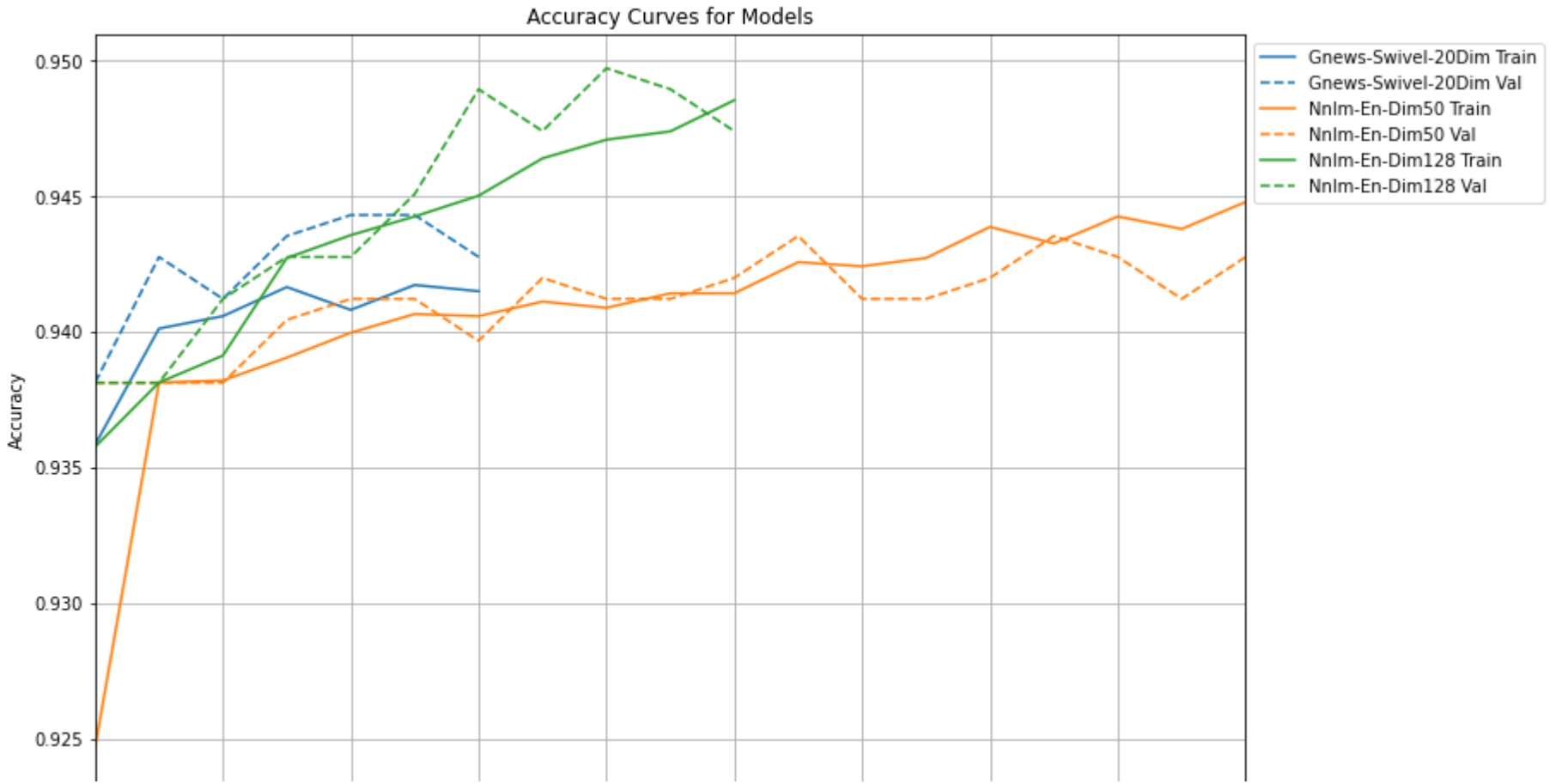
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 keras_layer_2 (KerasLayer)  (None, 128)               124642688

 dense_6 (Dense)             (None, 256)               33024

 dense_7 (Dense)             (None, 64)                16448

 dense_8 (Dense)             (None, 1)                 65

=================================================================
Total params: 124,692,225
Trainable params: 49,537
Non-trainable params: 124,642,688
_________________________________________________________________

Epoch: 0, accuracy:0.9358,  loss:0.3056,  val_accuracy:0.9381,  val_loss:0.2067,
...........
```

it only train last 3 dense layers as we set trainable=False... so fine tuning is not performed here.

Task 8: Compare Accuracy and Loss Curves

```
plt.rcParams['figure.figsize'] = (12, 8)
plotter = tfdocs.plots.HistoryPlotter(metric = 'accuracy')


plt.legend(bbox_to_anchor=(1.0, 1.0), loc='upper left')
plt.title("Accuracy Curves for Models")
plt.show()
```

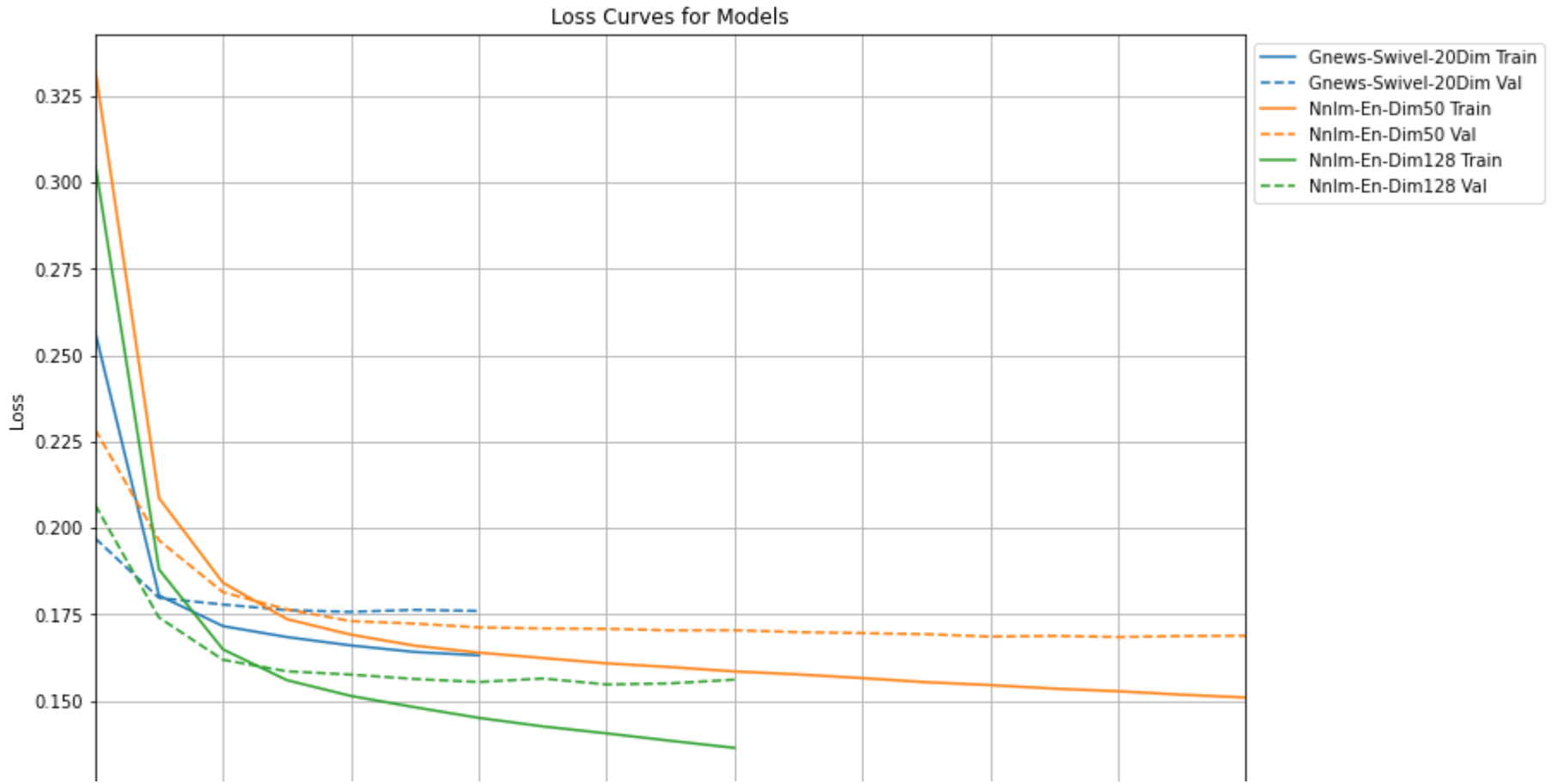


here we can understand that larger training model predict more accurately.nnlm-en-dim128 model has more accuracy....and nnlm-en-dim50 model is alsogood but it is sufferd from over fitting ..... here all these we understand from this epochs vs accuracy graph

```
plotter = tfdocs.plots.HistoryPlotter(metric = 'loss')
```

Saving...

```
plt.legend(bbox_to_anchor=(1.0, 1.0), loc='upper left')
plt.title("Loss Curves for Models")
plt.show()
```



large models has less loss ...we understand that from this epoch vs loss graph

Task 9: Fine-tune Model from TF Hub

Saving...

```
module_url = "https://tfhub.dev/google/tf2-preview/gnews-swivel-20dim/1" #@param ["https://tfhub.dev/google/tf2-pre  w/gnews
```

module_url: https://tfhub.dev/google/tf2-preview/gn

```
histories['gnews-swivel-20dim-finetuned']=train_and_evaluate_model(module_url,embed_size=20,name='gnews-swivel-20dim-finetun

#here we set trainable =True... that menas the model will fine tuned
```

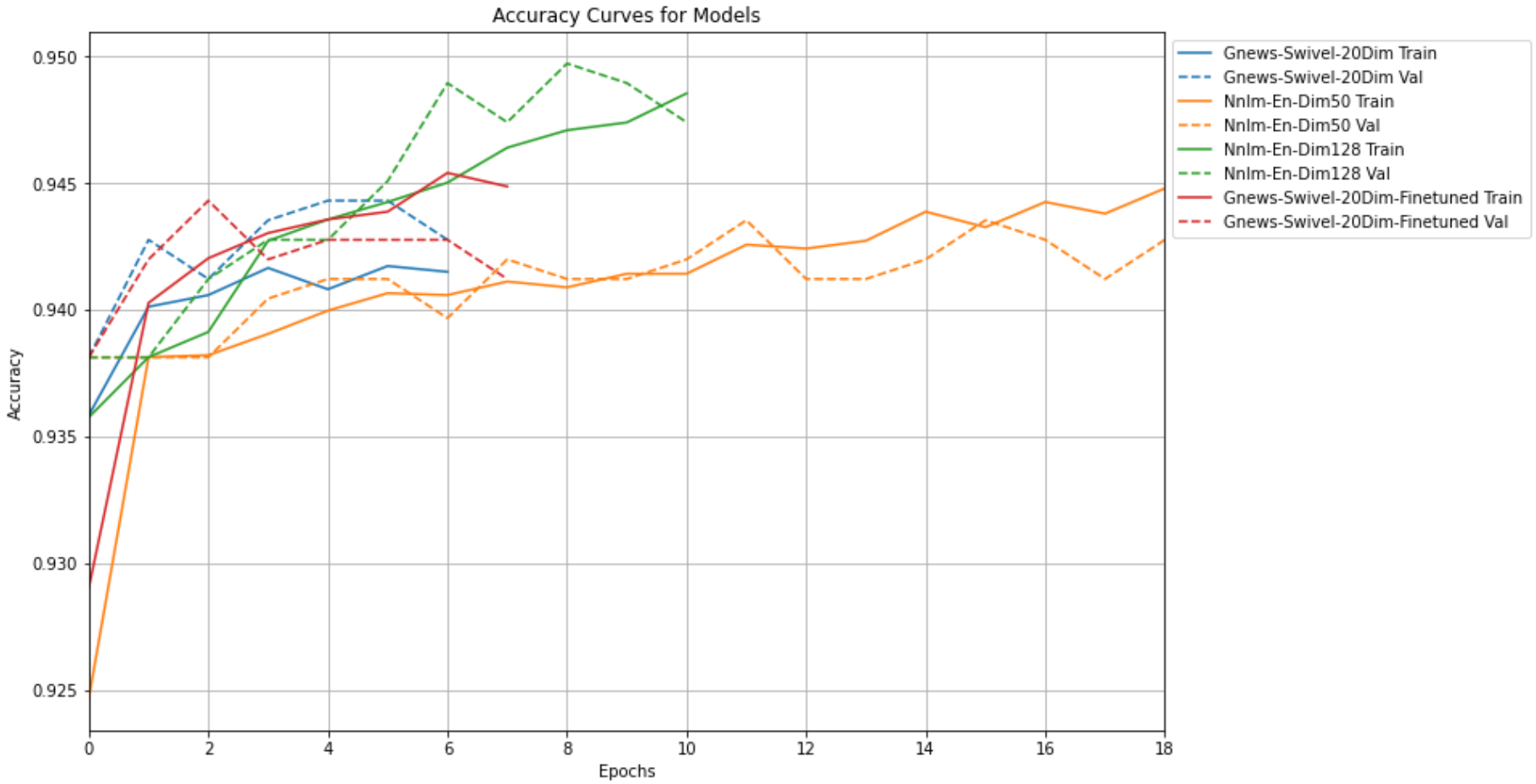
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_3 (KerasLayer)  (None, 20)                400020

 dense_9 (Dense)             (None, 256)               5376

 dense_10 (Dense)            (None, 64)                16448

 dense_11 (Dense)            (None, 1)                 65

=================================================================
Total params: 421,909
Trainable params: 421,909
Non-trainable params: 0
_________________________________________________________________

Epoch: 0, accuracy:0.9290,  loss:0.2744,  val_accuracy:0.9381,  val_loss:0.1993,
........
```

here we got 421909 trainable parameters which more than the trainable parameters when we set trainable=False

```
#histories['universal-sentence-encoder-large']=train_and_evaluate_model(module_url,embed_size=512,name='universal-sentence-e
```



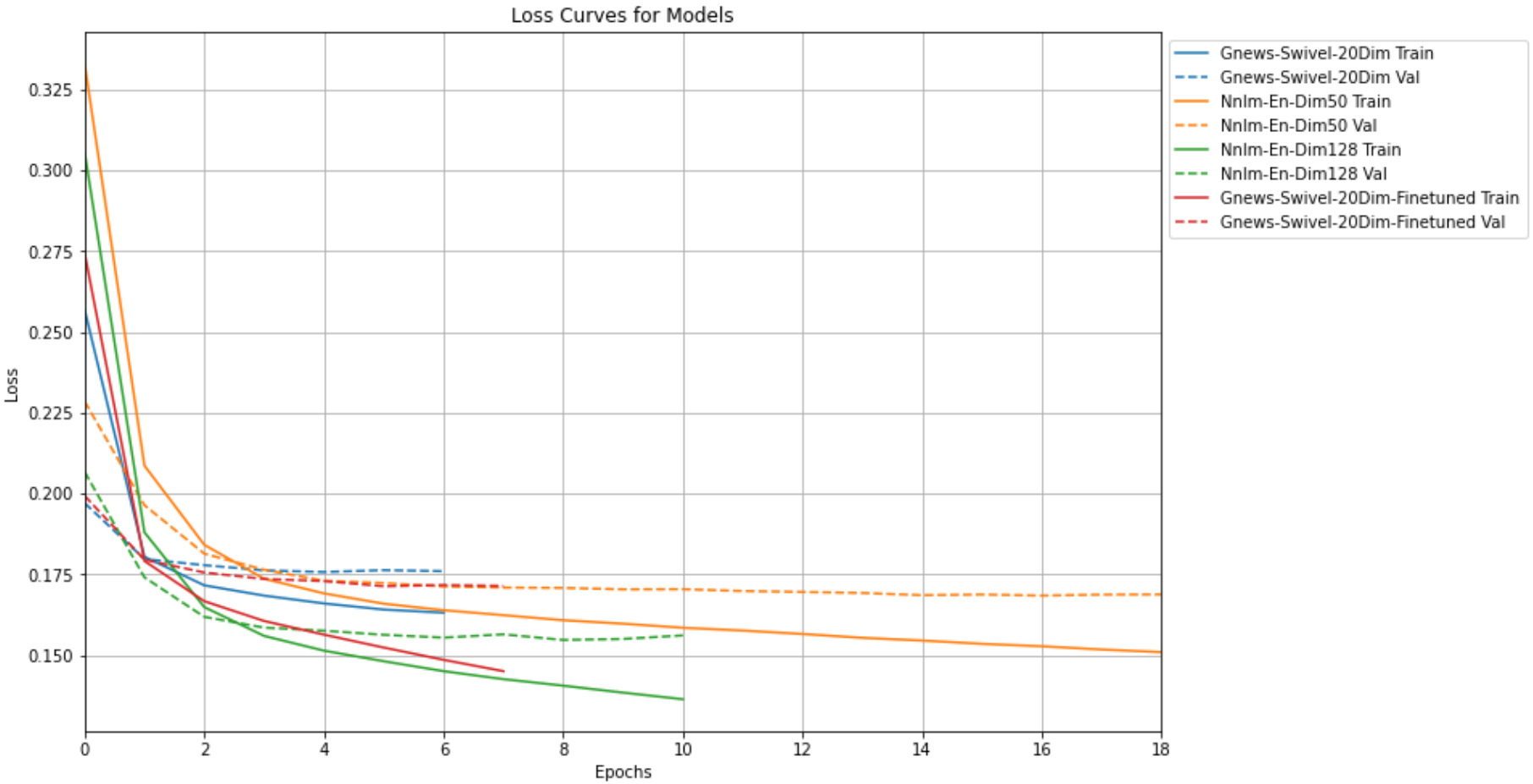
```
                                                    , 8)
                                                    r(metric = 'accuracy')
plotter.plot(histories)
plt.xlabel("Epochs")
plt.legend(bbox_to_anchor=(1.0, 1.0), loc='upper left')
plt.title("Accuracy Curves for Models")
plt.show()
```



from this graph we can understand that the fine tuned gnews-swivel-20dim model performs better than not fine tuned bigger model nnlm-en-dim50.

Saving...

```
r(metric = 'loss')
plotter.plot(histories)
plt.xlabel("Epochs")
plt.legend(bbox_to_anchor=(1.0, 1.0), loc='upper left')
plt.title("Loss Curves for Models")
plt.show()
```



from this graph we can understand that the fine tuned gnews-swivel-20dim model performs better than not fine tuned bigger model nnlm-en-dim50.

Saving...

Task 10: Train Bigger Models and Visualize Metrics with TensorBoard

```
# module_url = "https://tfhub.dev/google/universal-sentence-encoder/4" #@param ["https://tfhub.dev/google/tf2-preview/gnews-s

# histories['universal-sentence-encoder']=train_and_evaluate_model(module_url,embed_size=512,name='universal-sentence-encoder
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_4 (KerasLayer)  (None, 512)               256797824

 dense_12 (Dense)            (None, 256)               131328

 dense_13 (Dense)            (None, 64)                16448

 dense_14 (Dense)            (None, 1)                 65

=================================================================
Total params: 256,945,665
Trainable params: 147,841
Non-trainable params: 256,797,824
_________________________________________________________________

Epoch: 0, accuracy:0.9330,  loss:0.3067,  val_accuracy:0.9381,  val_loss:0.1778,
.............
```

universal-sentence-encoder-4 this model is based on the DEEP AVERAGING neurl network architecture. Where as universal-sentence-encoder-large model is based on the Transformer architecture.

Saving...

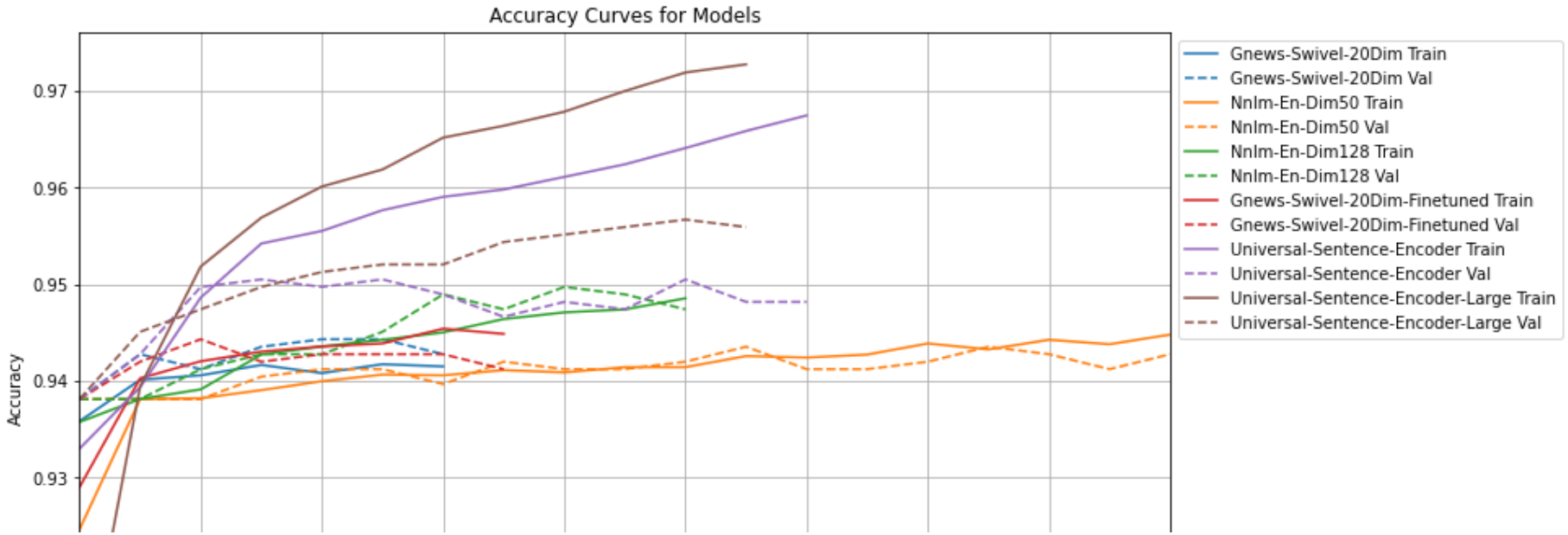
```
module_url = "https://tfhub.dev/google/universal-sentence-encoder-large/5" #@param ["https://tfhub.dev/google/tf2-p  iew/gne
```

module_url: https://tfhub.dev/google/universal-sente

```
histories['universal-sentence-encoder-large']=train_and_evaluate_model(module_url,embed_size=512,name='universal-sentence-en
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 keras_layer_5 (KerasLayer)  (None, 512)               147354880

 dense_15 (Dense)            (None, 256)               131328

 dense_16 (Dense)            (None, 64)                16448

 dense_17 (Dense)            (None, 1)                 65

=================================================================
Total params: 147,502,721
Trainable params: 147,841
Non-trainable params: 147,354,880
_________________________________________________________________

Epoch: 0, accuracy:0.9061,  loss:0.3341,  val_accuracy:0.9381,  val_loss:0.1730,
.............
```

```
plt.rcParams['figure.figsize'] = (12, 8)
plotter = tfdocs.plots.HistoryPlotter(metric = 'accuracy')
plotter.plot(histories)
plt.xlabel("Epochs")
plt.legend(bbox_to_anchor=(1.0, 1.0), loc='upper left')
plt.title("Accuracy Curves for Models")
plt.show()
```



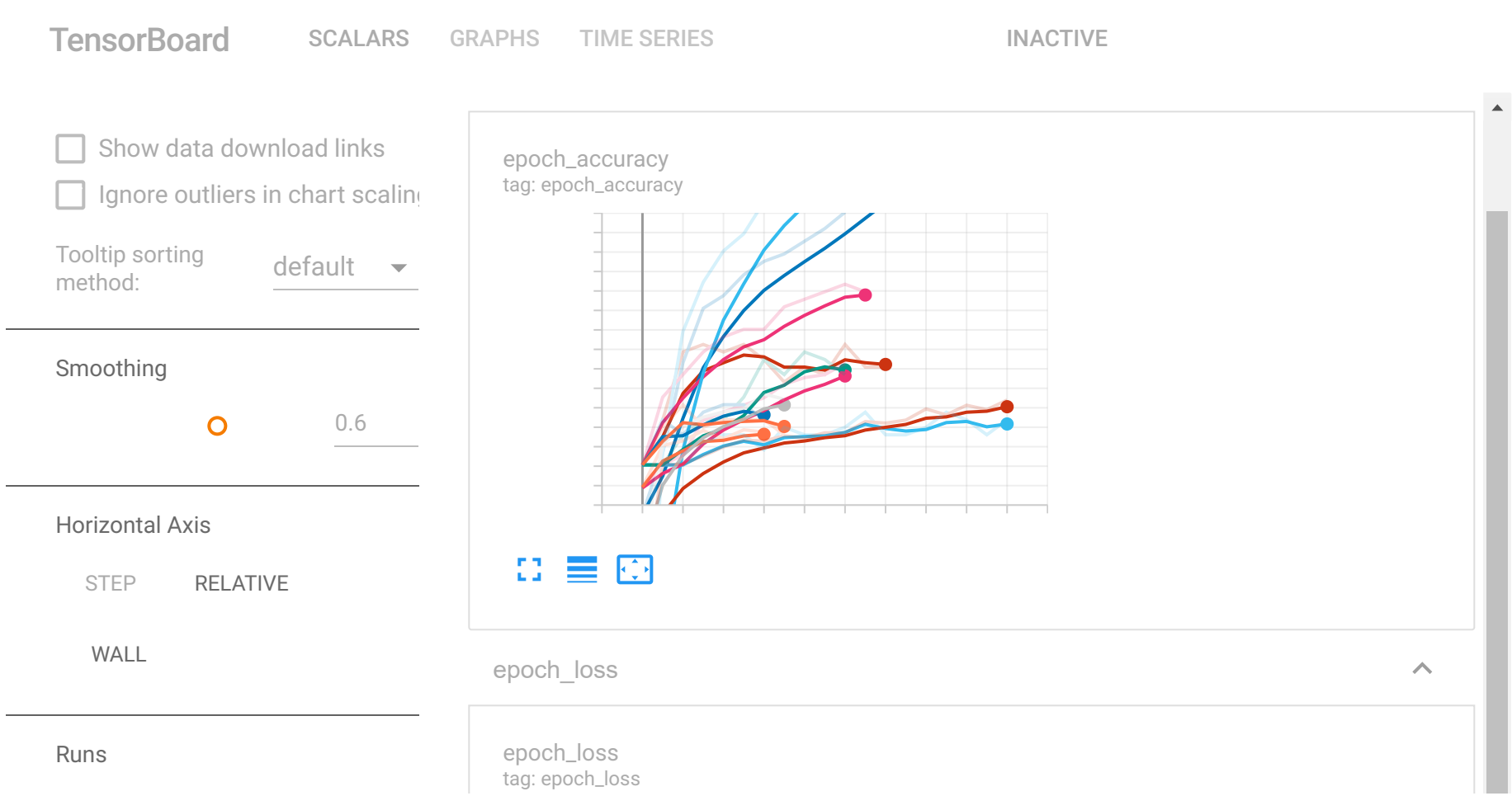

```
plotter = tfdocs.plots.HistoryPlotter(metric = 'loss')
plotter.plot(histories)
plt.xlabel("Epochs")
plt.legend(bbox_to_anchor=(1.0, 1.0), loc='upper left')
plt.title("Loss Curves for Models")
plt.show()
```

Saving...

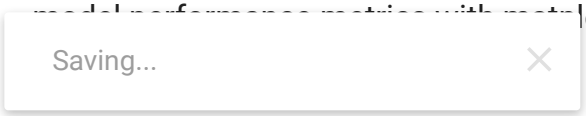
So, clearly our bigger model is winning in terms of validation accuracy and validatiion loss

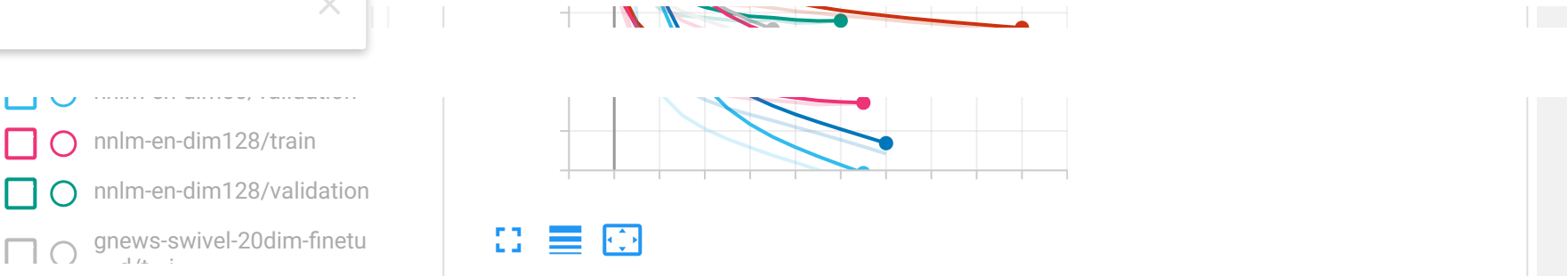
```
%load_ext tensorboard
%tensorboard --logdir {logdir}
```

Saving...



in this project, I have learned how to use various pre trained, NLP text embedding models from TensorFlow and TensorFlow Hub, and I am able to perform transfer, learning and fine tuning on models on a real world text data set. And finally, we were able to visualize, model performance metrics with matplotlib a TensorFlow documentation package.And finally with the Tensorboard viewer.





TOGGLE ALL RUNS

/tmp/tmpwt68oxl_/tensorboard_logs

evaluation_accuracy_vs_iterations

✓ 0s completed at 1:45 AM

Saving...