```
!wget --quiet
https://raw.githubusercontent.com/tensorflow/models/master/official/
nlp/bert/tokenization.py

import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
import tensorflow_hub as hub
from sklearn.model_selection import train_test_split

import tokenization


FMS = pd.read_csv("/content/hateful_memes_original.csv")
FMS.head()
```

```
         img  label                                                text
\
0  13894.png      0                            putting bows on your pet

1  37408.png      0  i love everything and everybody! except for sq...

2  82403.png      0  everybody loves chocolate chip cookies, even h...

3  16952.png      0           go sports! do the thing! win the points!

4  02973.png      0        how long can i run? till the chain tightens


                                             caption
0  little girl is playing with her hands on the g...
1  black dog with blue collar is shaking off the ...
2  man with black hair and white shirt is standin...
3  two women are playing with their arms around e...
4                  dog running through the grass .
```

```
FMS['textNdesc'] = 'In the picture '+ FMS.caption + ' And the text
says: ' + FMS.text
print(FMS.textNdesc[0], '\n\n')
FMS.head()
```

```
In the picture little girl is playing with her hands on the ground .
And the text says: putting bows on your pet
```

```
         img  label                                                text
\
0  13894.png      0                            putting bows on your pet

1  37408.png      0  i love everything and everybody! except for sq...

2  82403.png      0  everybody loves chocolate chip cookies, even h...

3  16952.png      0           go sports! do the thing! win the points!

4  02973.png      0        how long can i run? till the chain tightens


                                                 caption  \
0  little girl is playing with her hands on the g...
1  black dog with blue collar is shaking off the ...
2  man with black hair and white shirt is standin...
3  two women are playing with their arms around e...
4                    dog running through the grass .

                                               textNdesc
0  In the picture little girl is playing with her...
1  In the picture black dog with blue collar is s...
2  In the picture man with black hair and white s...
3  In the picture two women are playing with thei...
4  In the picture dog running through the grass ....
```

```python
def bert_encode(texts, tokenizer, max_len=512):
    all_tokens = []
    all_masks = []
    all_segments = []

    for text in texts:
        text = tokenizer.tokenize(text)

        text = text[:max_len-2]
        input_sequence = ["[CLS]"] + text + ["[SEP]"]
        pad_len = max_len - len(input_sequence)

        tokens = tokenizer.convert_tokens_to_ids(input_sequence)
        tokens += [0] * pad_len
        pad_masks = [1] * len(input_sequence) + [0] * pad_len
        segment_ids = [0] * max_len

        all_tokens.append(tokens)
        all_masks.append(pad_masks)
        all_segments.append(segment_ids)

    return np.array(all_tokens), np.array(all_masks),
np.array(all_segments)
```

```
def build_model(bert_layer, max_len=512):
    input_word_ids = Input(shape=(max_len,), dtype=tf.int32,
name="input_word_ids")
    input_mask = Input(shape=(max_len,), dtype=tf.int32,
name="input_mask")
    segment_ids = Input(shape=(max_len,), dtype=tf.int32,
name="segment_ids")

    _, sequence_output = bert_layer([input_word_ids, input_mask,
segment_ids])
    clf_output = sequence_output[:, 0, :]
    out = Dense(1, activation='sigmoid')(clf_output)

    model = Model(inputs=[input_word_ids, input_mask, segment_ids],
outputs=out)
    model.compile(Adam(lr=1e-5), loss='binary_crossentropy',
metrics=['accuracy', 'AUC'])

    return model

%%time
module_url = "https://tfhub.dev/tensorflow/bert_en_uncased_L-24_H-
1024_A-16/1"
bert_layer = hub.KerasLayer(module_url, trainable=True)

CPU times: user 22.6 s, sys: 4.99 s, total: 27.5 s
Wall time: 29.9 s

X_train, X_test, y_train, y_test = train_test_split(FMS.textNdesc,
FMS.label, test_size=0.2, random_state=42)

vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
tokenizer = tokenization.FullTokenizer(vocab_file, do_lower_case)

train_input = bert_encode(X_train.values, tokenizer, max_len=160)
test_input = bert_encode(X_test.values, tokenizer, max_len=160)
train_labels = y_train.values

print(len(X_train), len(X_test), len(y_train), len(y_test))

2719 680 2719 680

model = build_model(bert_layer, max_len=160)
model.summary()

Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to
==================================================================
```

```
==============================
input_word_ids (InputLayer)     [(None, 160)]        0

_____________________________________________________________________
_____________________________
input_mask (InputLayer)         [(None, 160)]        0

_____________________________________________________________________
_____________________________
segment_ids (InputLayer)        [(None, 160)]        0

_____________________________________________________________________
_____________________________
keras_layer (KerasLayer)        [(None, 1024), (None 335141889
input_word_ids[0][0]

input_mask[0][0]

segment_ids[0][0]
_____________________________________________________________________
_____________________________
tf.__operators__.getitem (Slici (None, 1024)         0
keras_layer[0][1]
_____________________________________________________________________
_____________________________
dense (Dense)                   (None, 1)            1025
tf.__operators__.getitem[0][0]
==========================================================================
==============================
Total params: 335,142,914
Trainable params: 335,142,913
Non-trainable params: 1
_____________________________________________________________________
_____________________________
```

```
from keras import backend as K
K.clear_session()

checkpoint = ModelCheckpoint('model.h5', monitor='val_loss',
save_best_only=True)

train_history = model.fit(
    train_input, train_labels,
    validation_split=0.3,
    epochs=30,
    callbacks=[checkpoint],
    batch_size=8,
    steps_per_epoch=12
)
```

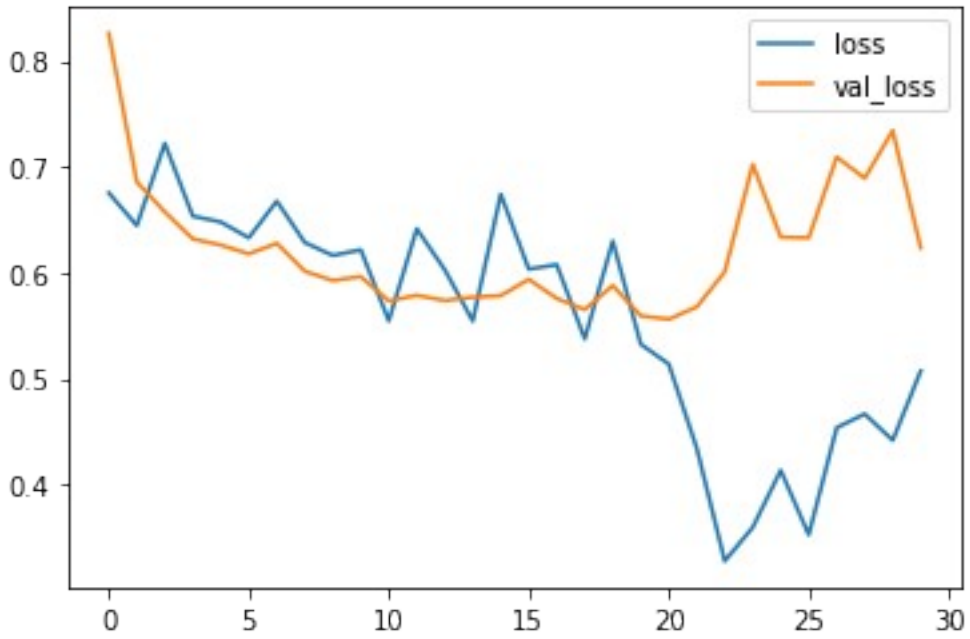
```
Epoch 1/30
12/12 [==============================] - 59s 2s/step - loss: 0.6537 -
accuracy: 0.6650 - auc: 0.5033 - val_loss: 0.8260 - val_accuracy:
0.3615 - val_auc: 0.5998
Epoch 2/30
12/12 [==============================] - 22s 2s/step - loss: 0.7052 -
accuracy: 0.5185 - auc: 0.5671 - val_loss: 0.6857 - val_accuracy:
0.6471 - val_auc: 0.6000
Epoch 3/30
12/12 [==============================] - 22s 2s/step - loss: 0.7849 -
accuracy: 0.5367 - auc: 0.6318 - val_loss: 0.6573 - val_accuracy:
0.6752 - val_auc: 0.6131
Epoch 4/30
12/12 [==============================] - 22s 2s/step - loss: 0.6635 -
accuracy: 0.5818 - auc: 0.5891 - val_loss: 0.6318 - val_accuracy:
0.6471 - val_auc: 0.6507
Epoch 5/30
12/12 [==============================] - 22s 2s/step - loss: 0.6474 -
accuracy: 0.6624 - auc: 0.4979 - val_loss: 0.6260 - val_accuracy:
0.6520 - val_auc: 0.6602
Epoch 6/30
12/12 [==============================] - 22s 2s/step - loss: 0.6434 -
accuracy: 0.6201 - auc: 0.6193 - val_loss: 0.6177 - val_accuracy:
0.6532 - val_auc: 0.6708
Epoch 7/30
12/12 [==============================] - 22s 2s/step - loss: 0.6685 -
accuracy: 0.5819 - auc: 0.6922 - val_loss: 0.6276 - val_accuracy:
0.7071 - val_auc: 0.6946
Epoch 8/30
12/12 [==============================] - 22s 2s/step - loss: 0.6396 -
accuracy: 0.7161 - auc: 0.6738 - val_loss: 0.6012 - val_accuracy:
0.7083 - val_auc: 0.7023
Epoch 9/30
12/12 [==============================] - 22s 2s/step - loss: 0.6347 -
accuracy: 0.6889 - auc: 0.5433 - val_loss: 0.5925 - val_accuracy:
0.7034 - val_auc: 0.7021
Epoch 10/30
12/12 [==============================] - 22s 2s/step - loss: 0.6628 -
accuracy: 0.6240 - auc: 0.6246 - val_loss: 0.5962 - val_accuracy:
0.6716 - val_auc: 0.7219
Epoch 11/30
12/12 [==============================] - 22s 2s/step - loss: 0.5249 -
accuracy: 0.7288 - auc: 0.8927 - val_loss: 0.5732 - val_accuracy:
0.7194 - val_auc: 0.7286
Epoch 12/30
12/12 [==============================] - 22s 2s/step - loss: 0.6802 -
accuracy: 0.6654 - auc: 0.5736 - val_loss: 0.5785 - val_accuracy:
0.7083 - val_auc: 0.7322
Epoch 13/30
12/12 [==============================] - 22s 2s/step - loss: 0.5253 -
```

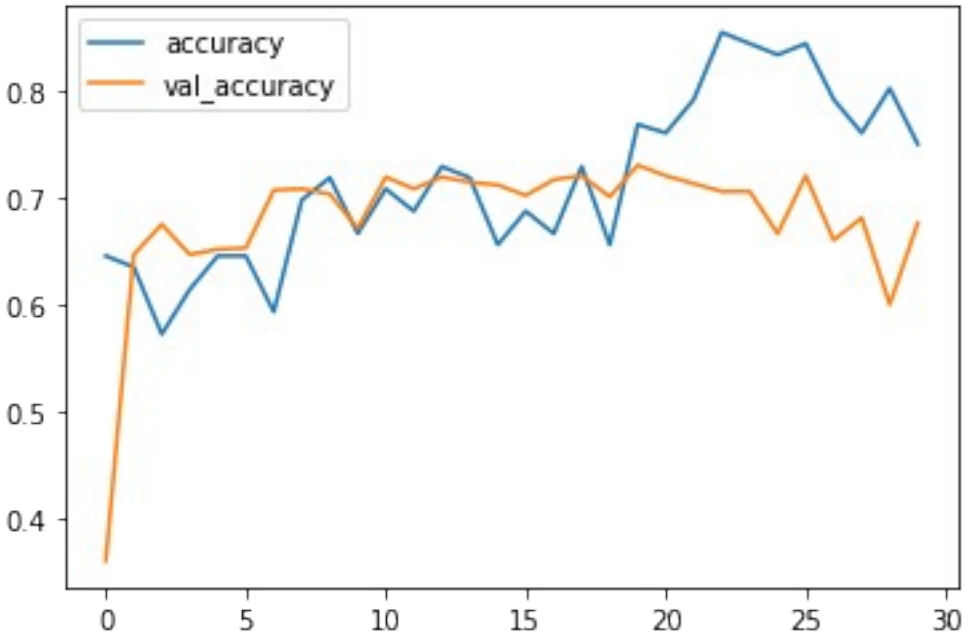
```
accuracy: 0.7844 - auc: 0.7663 - val_loss: 0.5735 - val_accuracy:
0.7194 - val_auc: 0.7339
Epoch 14/30
12/12 [==============================] - 22s 2s/step - loss: 0.5641 -
accuracy: 0.7254 - auc: 0.7311 - val_loss: 0.5768 - val_accuracy:
0.7145 - val_auc: 0.7331
Epoch 15/30
12/12 [==============================] - 22s 2s/step - loss: 0.6028 -
accuracy: 0.7031 - auc: 0.7037 - val_loss: 0.5782 - val_accuracy:
0.7120 - val_auc: 0.7349
Epoch 16/30
12/12 [==============================] - 22s 2s/step - loss: 0.5678 -
accuracy: 0.7195 - auc: 0.7695 - val_loss: 0.5936 - val_accuracy:
0.7022 - val_auc: 0.7348
Epoch 17/30
12/12 [==============================] - 22s 2s/step - loss: 0.6036 -
accuracy: 0.6358 - auc: 0.7074 - val_loss: 0.5756 - val_accuracy:
0.7169 - val_auc: 0.7334
Epoch 18/30
12/12 [==============================] - 22s 2s/step - loss: 0.5703 -
accuracy: 0.6957 - auc: 0.7962 - val_loss: 0.5650 - val_accuracy:
0.7206 - val_auc: 0.7361
Epoch 19/30
12/12 [==============================] - 22s 2s/step - loss: 0.5900 -
accuracy: 0.6700 - auc: 0.7785 - val_loss: 0.5877 - val_accuracy:
0.7010 - val_auc: 0.7391
Epoch 20/30
12/12 [==============================] - 22s 2s/step - loss: 0.5703 -
accuracy: 0.7388 - auc: 0.7537 - val_loss: 0.5591 - val_accuracy:
0.7304 - val_auc: 0.7419
Epoch 21/30
12/12 [==============================] - 22s 2s/step - loss: 0.5245 -
accuracy: 0.7193 - auc: 0.8243 - val_loss: 0.5560 - val_accuracy:
0.7206 - val_auc: 0.7477
Epoch 22/30
12/12 [==============================] - 22s 2s/step - loss: 0.4042 -
accuracy: 0.8298 - auc: 0.9111 - val_loss: 0.5675 - val_accuracy:
0.7132 - val_auc: 0.7513
Epoch 23/30
12/12 [==============================] - 22s 2s/step - loss: 0.3593 -
accuracy: 0.8284 - auc: 0.8210 - val_loss: 0.6006 - val_accuracy:
0.7059 - val_auc: 0.7418
Epoch 24/30
12/12 [==============================] - 22s 2s/step - loss: 0.3532 -
accuracy: 0.8435 - auc: 0.9249 - val_loss: 0.7022 - val_accuracy:
0.7059 - val_auc: 0.7408
Epoch 25/30
12/12 [==============================] - 22s 2s/step - loss: 0.5514 -
accuracy: 0.7521 - auc: 0.8147 - val_loss: 0.6335 - val_accuracy:
0.6667 - val_auc: 0.7323
```

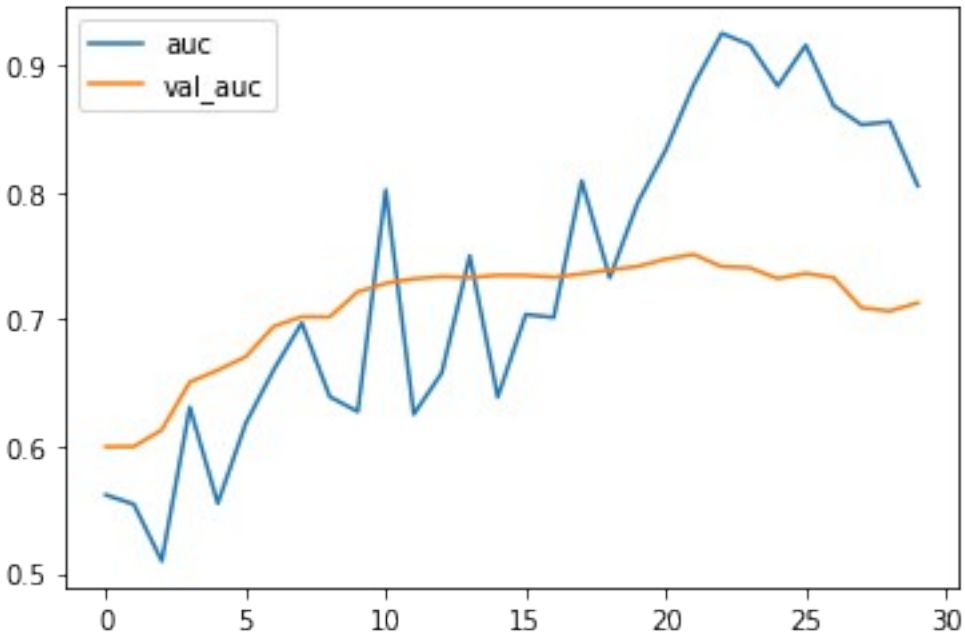
```
Epoch 26/30
12/12 [==============================] - 22s 2s/step - loss: 0.3674 -
accuracy: 0.8445 - auc: 0.9172 - val_loss: 0.6329 - val_accuracy:
0.7206 - val_auc: 0.7365
Epoch 27/30
12/12 [==============================] - 22s 2s/step - loss: 0.4863 -
accuracy: 0.8063 - auc: 0.8437 - val_loss: 0.7091 - val_accuracy:
0.6605 - val_auc: 0.7330
Epoch 28/30
12/12 [==============================] - 22s 2s/step - loss: 0.3361 -
accuracy: 0.8259 - auc: 0.9233 - val_loss: 0.6891 - val_accuracy:
0.6814 - val_auc: 0.7093
Epoch 29/30
12/12 [==============================] - 22s 2s/step - loss: 0.3786 -
accuracy: 0.8565 - auc: 0.9154 - val_loss: 0.7341 - val_accuracy:
0.6005 - val_auc: 0.7068
Epoch 30/30
12/12 [==============================] - 22s 2s/step - loss: 0.5064 -
accuracy: 0.7188 - auc: 0.8537 - val_loss: 0.6233 - val_accuracy:
0.6765 - val_auc: 0.7132
```

```
history_frame = pd.DataFrame(train_history.history)
history_frame.loc[:, ['loss', 'val_loss']].plot()
history_frame.loc[:, ['accuracy', 'val_accuracy']].plot()
history_frame.loc[:, ['auc', 'val_auc']].plot()
```

```
<AxesSubplot:>
```








```
model.load_weights('model.h5')
test_pred = model.predict(test_input)

from sklearn.metrics import roc_auc_score, balanced_accuracy_score

y_actual = list(y_test)
y_prob = list(test_pred.reshape(len(test_pred), ))

print("AUC: ", roc_auc_score(y_actual, y_prob))
```

```
AUC:  0.7705123561697267
```

```
def threshold(i, th):
    if i > th:
        return 1
    else:
        return 0

def select_threshold(y_actual, y_prob):
    acc = 0.5
    selected_th = 0
    for th in np.arange(0, 1, 0.005):
        y_predicted = [threshold(i, th) for i in y_prob]
        if balanced_accuracy_score(y_actual, y_predicted) > acc:
            acc = balanced_accuracy_score(y_actual, y_predicted)
            selected_th = th
    return selected_th

th = select_threshold(y_actual, y_prob)
y_predicted = [threshold(i, th) for i in y_prob]

print("Accuracy: ", balanced_accuracy_score(y_actual, y_predicted))
```

```
Accuracy:  0.7112853945523268
```