```
#@title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Licensed under the Apache License, Version 2.0 (the "License");

Multiple Layer GRU

```
[ ] from __future__ import absolute_import, division, print_function, unicode_literals


    import tensorflow_datasets as tfds
    import tensorflow as tf
    print(tf.__version__)
```

```
[ ] import tensorflow_datasets as tfds
    import tensorflow as tf
    print(tf.__version__)
```

```
[ ] # Get the data
    dataset, info = tfds.load('imdb_reviews/subwords8k', with_info=True, as_supervised=True)
    train_dataset, test_dataset = dataset['train'], dataset['test']
```

```
[ ] tokenizer = info.features['text'].encoder
```

```
[ ] BUFFER_SIZE = 10000
    BATCH_SIZE = 64

    train_dataset = train_dataset.shuffle(BUFFER_SIZE)
    train_dataset = train_dataset.padded_batch(BATCH_SIZE, train_dataset.output_shapes)
    test_dataset = test_dataset.padded_batch(BATCH_SIZE, test_dataset.output_shapes)
```

```
[ ] model = tf.keras.Sequential([
        tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
        tf.keras.layers.Conv1D(128, 5, activation='relu'),
        tf.keras.layers.GlobalAveragePooling1D(),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])
```

```
[ ] model.summary()
```

```
[ ] model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
[ ] NUM_EPOCHS = 10
    history = model.fit(train_dataset, epochs=NUM_EPOCHS, validation_data=test_dataset)
```

```
[ ] import matplotlib.pyplot as plt


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

```
[ ] plot_graphs(history, 'accuracy')
```

```
[ ] plot_graphs(history, 'loss')
```