```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt
```

```
import keras
keras.__version__
```

```
'2.4.3'
```

```
from keras.datasets import imdb

(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

```
<string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/datasets/imdb.py:159: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify '
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/datasets/imdb.py:160: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify '
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])
```

```
max([max(sequence) for sequence in train_data])
```

```
9999
```

```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
    # Create an all-zero matrix of shape (len(sequences), dimension)
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.  # set specific indices of results[i] to 1s
    return results

# Our vectorized training data
x_train = vectorize_sequences(train_data)
# Our vectorized test data
x_test = vectorize_sequences(test_data)
```

```
x_train[0]
```

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

```
# Our vectorized labels
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

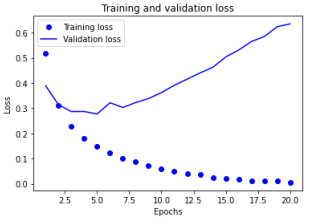
run the model implementing Checkpoint

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
# checkpoint
filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val),
                    callbacks=callbacks_list)
```

```
Epoch 7/20
30/30 [==============================] - 1s 35ms/step - loss: 0.1219 - accuracy: 0.9630 - val_loss: 0.3217 - val_accuracy: 0.8744

Epoch 00006: val_accuracy did not improve from 0.88880
Epoch 7/20
30/30 [==============================] - 1s 35ms/step - loss: 0.1022 - accuracy: 0.9711 - val_loss: 0.3028 - val_accuracy: 0.8847
```

```
Epoch 00007: val_accuracy did not improve from 0.88880
Epoch 8/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0855 - accuracy: 0.9768 - val_loss: 0.3224 - val_accuracy: 0.8801

Epoch 00008: val_accuracy did not improve from 0.88880
Epoch 9/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0684 - accuracy: 0.9833 - val_loss: 0.3384 - val_accuracy: 0.8812

Epoch 00009: val_accuracy did not improve from 0.88880
Epoch 10/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0556 - accuracy: 0.9876 - val_loss: 0.3617 - val_accuracy: 0.8801

Epoch 00010: val_accuracy did not improve from 0.88880
Epoch 11/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0440 - accuracy: 0.9909 - val_loss: 0.3908 - val_accuracy: 0.8764

Epoch 00011: val_accuracy did not improve from 0.88880
Epoch 12/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0357 - accuracy: 0.9933 - val_loss: 0.4152 - val_accuracy: 0.8740

Epoch 00012: val_accuracy did not improve from 0.88880
Epoch 13/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0298 - accuracy: 0.9951 - val_loss: 0.4398 - val_accuracy: 0.8727

Epoch 00013: val_accuracy did not improve from 0.88880
Epoch 14/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0219 - accuracy: 0.9973 - val_loss: 0.4621 - val_accuracy: 0.8737

Epoch 00014: val_accuracy did not improve from 0.88880
Epoch 15/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0210 - accuracy: 0.9969 - val_loss: 0.5033 - val_accuracy: 0.8684

Epoch 00015: val_accuracy did not improve from 0.88880
Epoch 16/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0155 - accuracy: 0.9982 - val_loss: 0.5306 - val_accuracy: 0.8674

Epoch 00016: val_accuracy did not improve from 0.88880
Epoch 17/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0123 - accuracy: 0.9989 - val_loss: 0.5646 - val_accuracy: 0.8650

Epoch 00017: val_accuracy did not improve from 0.88880
Epoch 18/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0117 - accuracy: 0.9983 - val_loss: 0.5841 - val_accuracy: 0.8682

Epoch 00018: val_accuracy did not improve from 0.88880
Epoch 19/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0076 - accuracy: 0.9993 - val_loss: 0.6231 - val_accuracy: 0.8677

Epoch 00019: val_accuracy did not improve from 0.88880
Epoch 20/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0059 - accuracy: 0.9997 - val_loss: 0.6345 - val_accuracy: 0.8675

Epoch 00020: val_accuracy did not improve from 0.88880
```

```
import matplotlib.pyplot as plt
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



With Check point, vaildation accuracy increased form epochs 1,2,3 and 5

epoch 1 - accuracy increased from 0.7792 to 0.8710

epoch 2 - accuracy increased from 0.8710 to 0.8809

epoch 3 - accuracy increased from 0.8758 to 0.8855

epoch 5 - accuracy increased from 0.8881 to 0.8888

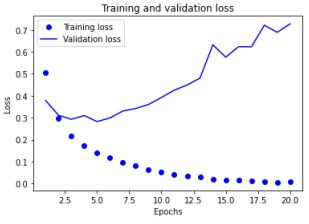
run the model implementing Checkpoint

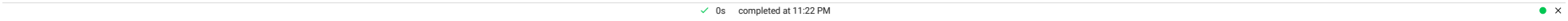
```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
model.add(layers.Dense(1, activation='sigmoid'))
tb_callback = tf.keras.callbacks.TensorBoard(log_dir="C:/AdvML/log", histogram_freq=1)
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val),
                    callbacks=[tb_callback])
```

```
Epoch 1/20
30/30 [==============================] - 2s 56ms/step - loss: 0.5848 - accuracy: 0.7097 - val_loss: 0.3786 - val_accuracy: 0.8727
Epoch 2/20
30/30 [==============================] - 1s 38ms/step - loss: 0.3104 - accuracy: 0.9059 - val_loss: 0.3108 - val_accuracy: 0.8805
Epoch 3/20
30/30 [==============================] - 1s 38ms/step - loss: 0.2240 - accuracy: 0.9251 - val_loss: 0.2926 - val_accuracy: 0.8818
Epoch 4/20
30/30 [==============================] - 1s 37ms/step - loss: 0.1709 - accuracy: 0.9449 - val_loss: 0.3096 - val_accuracy: 0.8765
Epoch 5/20
30/30 [==============================] - 1s 37ms/step - loss: 0.1419 - accuracy: 0.9541 - val_loss: 0.2822 - val_accuracy: 0.8888
Epoch 6/20
30/30 [==============================] - 1s 37ms/step - loss: 0.1112 - accuracy: 0.9686 - val_loss: 0.2986 - val_accuracy: 0.8863
Epoch 7/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0930 - accuracy: 0.9749 - val_loss: 0.3299 - val_accuracy: 0.8808
Epoch 8/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0732 - accuracy: 0.9812 - val_loss: 0.3418 - val_accuracy: 0.8807
Epoch 9/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0611 - accuracy: 0.9853 - val_loss: 0.3598 - val_accuracy: 0.8791
Epoch 10/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0499 - accuracy: 0.9881 - val_loss: 0.3918 - val_accuracy: 0.8759
Epoch 11/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0406 - accuracy: 0.9908 - val_loss: 0.4250 - val_accuracy: 0.8771
Epoch 12/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0330 - accuracy: 0.9935 - val_loss: 0.4488 - val_accuracy: 0.8733
Epoch 13/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0242 - accuracy: 0.9957 - val_loss: 0.4802 - val_accuracy: 0.8715
Epoch 14/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0169 - accuracy: 0.9982 - val_loss: 0.6320 - val_accuracy: 0.8570
Epoch 15/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0188 - accuracy: 0.9970 - val_loss: 0.5756 - val_accuracy: 0.8621
Epoch 16/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0127 - accuracy: 0.9984 - val_loss: 0.6231 - val_accuracy: 0.8694
Epoch 17/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0095 - accuracy: 0.9992 - val_loss: 0.6228 - val_accuracy: 0.8667
Epoch 18/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0062 - accuracy: 0.9997 - val_loss: 0.7211 - val_accuracy: 0.8528
Epoch 19/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0062 - accuracy: 0.9997 - val_loss: 0.6887 - val_accuracy: 0.8658
Epoch 20/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0041 - accuracy: 0.9996 - val_loss: 0.7272 - val_accuracy: 0.8645
```

```
import matplotlib.pyplot as plt
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




0s completed at 11:22 PM