COSC 4337

All_forms_of RNN

#Recurrent Neural Networks

##Introduction to Recurrent Neural Networks ###Simple RNNs

[1]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, Embedding

model = Sequential()
model.add(SimpleRNN(10, input_shape=(5, 2)))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, 10)                130
=================================================================
Total params: 130
Trainable params: 130
Non-trainable params: 0
_________________________________________________________________
```

###Embedding Layers

[2]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, Embedding

vocab_size = 30
embddng_dim = 10
seqnc_lngth = 5

model = Sequential()
model.add(Embedding(vocab_size, embddng_dim, input_length=seqnc_lngth))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 5, 10)             300
=================================================================
Total params: 300
Trainable params: 300
Non-trainable params: 0
_________________________________________________________________
```

###Word Embedding + RNN on IMDB

[3]:
```
from keras.datasets import imdb
from keras.preprocessing import sequence

inpt_dim =  128
index_from = 3

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=10000,
                                                      start_char=1,
                                                      oov_char=2,
                                                      index_from=index_from,
                                                      skip_top=20)
x_train = sequence.pad_sequences(x_train, maxlen=inpt_dim)
x_test = sequence.pad_sequences(x_test, maxlen=inpt_dim)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

print(' '.join(str(int(id)) for id in x_train[7]))

word_to_id = imdb.get_word_index()
word_to_id = {k:(v+index_from) for k,v in word_to_id.items()}
word_to_id["<PAD>"] = 0
word_to_id["<START>"] = 1
word_to_id["<UNK>"] = 2
word_to_id["<UNUSED>"] = 3

id_to_word = {value:key for key,value in word_to_id.items()}
print(' '.join(id_to_word[id] for id in x_train[7]))
```

```
Using TensorFlow backend.
C:\Users\Nouhad\.conda\envs\cosc4337\lib\site-
packages\keras\datasets\imdb.py:101: 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
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
C:\Users\Nouhad\.conda\envs\cosc4337\lib\site-
packages\keras\datasets\imdb.py:102: 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
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])

x_train shape: (25000, 128)
x_test shape: (25000, 128)
55 655 707 6371 956 225 1456 841 42 1310 225 2 2493 1467 7722 2828 21 2 2 2 364
23 2 2228 2407 225 24 76 133 2 2 189 2293 2 2 814 2 2 2 2642 2 47 2 682 364 352
168 44 2 45 24 913 93 21 247 2441 2 116 34 35 1859 2 72 177 2 164 2 901 344 44 2
191 135 2 126 421 233 2 259 2 2 2 2 6847 2 2 3074 2 112 199 753 357 39 63 2 115
2 763 2 2 35 3282 1523 65 57 599 2 1916 277 1730 37 25 92 202 2 8848 44 25 28 2
22 2 122 24 4171 72 33 32
very middle class suburban setting there's zero atmosphere or mood there's <UNK>
lesbian suggest incestuous kiss but <UNK> <UNK> <UNK> low on <UNK> exploitation
scale there's not much here <UNK> <UNK> horror crowd <UNK> <UNK> filmed <UNK>
<UNK> <UNK> california <UNK> has <UNK> modern low budget look about <UNK> it's
not badly made but rather forgettable <UNK> acting by an unknown <UNK> me cast
<UNK> nothing <UNK> write home about <UNK> can't say <UNK> ever felt anything
<UNK> anyone <UNK> <UNK> <UNK> <UNK> commits <UNK> <UNK> sin <UNK> being both
dull boring from which <UNK> never <UNK> add <UNK> <UNK> an ultra thin story no
gore <UNK> rubbish ending character's who you don't give <UNK> toss about you
have <UNK> film <UNK> did not impress me at all
```

[4]:
```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import SimpleRNN, Embedding, BatchNormalization
from tensorflow.keras.layers import Dense, Activation, Input, Dropout
from keras.datasets import imdb
from keras.preprocessing import sequence
import numpy as np

seqnc_lngth =  128
embddng_dim = 64
vocab_size = 10000

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size,␣
 ↪skip_top=20)
x_train = sequence.pad_sequences(x_train, maxlen=seqnc_lngth)
x_test = sequence.pad_sequences(x_test, maxlen=seqnc_lngth)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

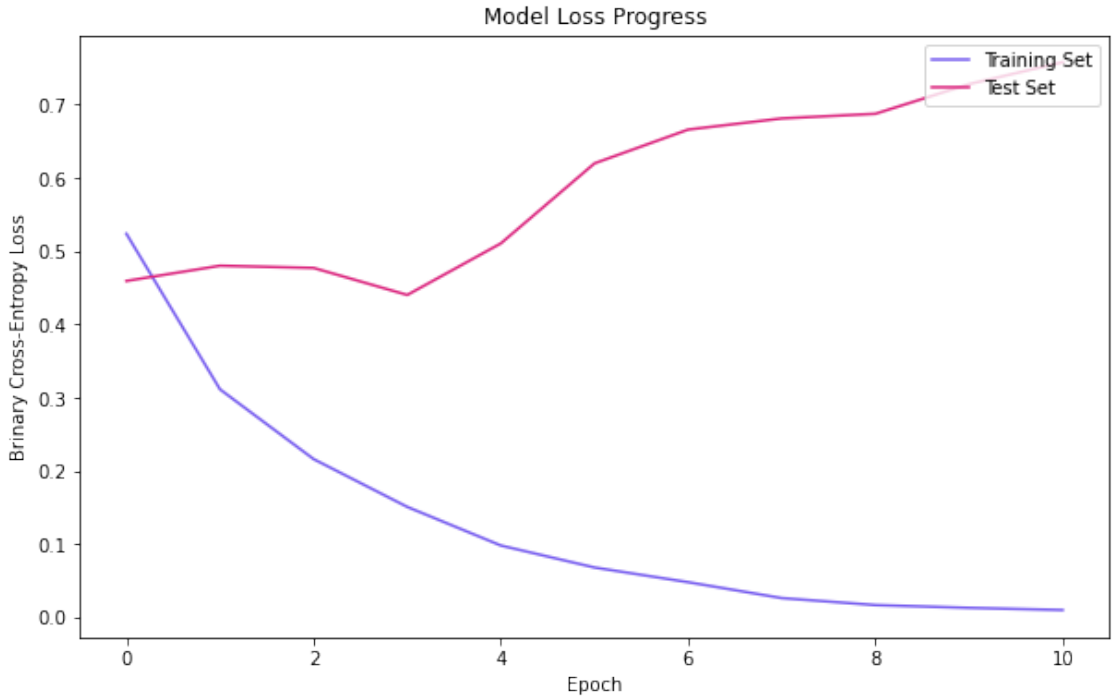
```
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# now with batch norm
inpt_vec = Input(shape=(seqnc_lngth,))
l1 = Embedding(vocab_size, embddng_dim, input_length=seqnc_lngth)(inpt_vec)
l2 = Dropout(0.3)(l1)
l3 = SimpleRNN(32)(l2)
l4 = BatchNormalization()(l3)
l5 = Dropout(0.2)(l4)
output = Dense(1, activation='sigmoid')(l5)

# model that takes input and encodes it into the latent space
rnn = Model(inpt_vec, output)

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

```
x_train shape: (25000, 128)
x_test shape: (25000, 128)
25000 train samples
25000 test samples
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128)]             0
_________________________________________________________________
embedding_1 (Embedding)      (None, 128, 64)           640000
_________________________________________________________________
dropout (Dropout)            (None, 128, 64)           0
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 32)                3104
_________________________________________________________________
batch_normalization (BatchNo (None, 32)                128
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 33
=================================================================
Total params: 643,265
Trainable params: 643,201
Non-trainable params: 64
_________________________________________________________________
```

[5]:
```
# Fitting the RNN to the data

from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3,
                              min_delta=1e-4, mode='min', verbose=1)

stop_alg = EarlyStopping(monitor='val_loss', patience=7,
                         restore_best_weights=True, verbose=1)

hist = rnn.fit(x_train, y_train, batch_size=100, epochs=1000,
                   callbacks=[stop_alg, reduce_lr], shuffle=True,
                   validation_data=(x_test, y_test))

rnn.save_weights("rnn.hdf5")

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,6))
plt.plot(hist.history['loss'], color='#785ef0')
plt.plot(hist.history['val_loss'], color='#dc267f')
plt.title('Model Loss Progress')
plt.ylabel('Brinary Cross-Entropy Loss')
plt.xlabel('Epoch')
plt.legend(['Training Set', 'Test Set'], loc='upper right')
plt.savefig('ch.13.rnn.imdb.loss.png', dpi=350, bbox_inches='tight')
plt.show()
```

```
Epoch 1/1000
250/250 [==============================] - 13s 53ms/step - loss: 0.5238 -
accuracy: 0.7327 - val_loss: 0.4593 - val_accuracy: 0.7764
Epoch 2/1000
250/250 [==============================] - 14s 56ms/step - loss: 0.3113 -
accuracy: 0.8723 - val_loss: 0.4801 - val_accuracy: 0.7932
Epoch 3/1000
250/250 [==============================] - 15s 59ms/step - loss: 0.2161 -
accuracy: 0.9156 - val_loss: 0.4771 - val_accuracy: 0.8304
Epoch 4/1000
250/250 [==============================] - 14s 57ms/step - loss: 0.1508 -
accuracy: 0.9442 - val_loss: 0.4402 - val_accuracy: 0.8453
Epoch 5/1000
250/250 [==============================] - 13s 53ms/step - loss: 0.0981 -
accuracy: 0.9672 - val_loss: 0.5106 - val_accuracy: 0.8350
Epoch 6/1000
250/250 [==============================] - 13s 53ms/step - loss: 0.0681 -
accuracy: 0.9760 - val_loss: 0.6199 - val_accuracy: 0.8176
Epoch 7/1000
```

```
249/250 [============================>.] - ETA: 0s - loss: 0.0479 - accuracy:
0.9837
Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
250/250 [==============================] - 13s 52ms/step - loss: 0.0480 -
accuracy: 0.9836 - val_loss: 0.6660 - val_accuracy: 0.8390
Epoch 8/1000
250/250 [==============================] - 13s 53ms/step - loss: 0.0263 -
accuracy: 0.9918 - val_loss: 0.6812 - val_accuracy: 0.8372
Epoch 9/1000
250/250 [==============================] - 14s 58ms/step - loss: 0.0168 -
accuracy: 0.9953 - val_loss: 0.6876 - val_accuracy: 0.8292
Epoch 10/1000
249/250 [============================>.] - ETA: 0s - loss: 0.0130 - accuracy:
0.9965
Epoch 00010: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
250/250 [==============================] - 13s 53ms/step - loss: 0.0130 -
accuracy: 0.9966 - val_loss: 0.7280 - val_accuracy: 0.8339
Epoch 11/1000
250/250 [==============================] - ETA: 0s - loss: 0.0100 - accuracy:
0.9974Restoring model weights from the end of the best epoch.
250/250 [==============================] - 15s 59ms/step - loss: 0.0100 -
accuracy: 0.9974 - val_loss: 0.7580 - val_accuracy: 0.8351
Epoch 00011: early stopping
```

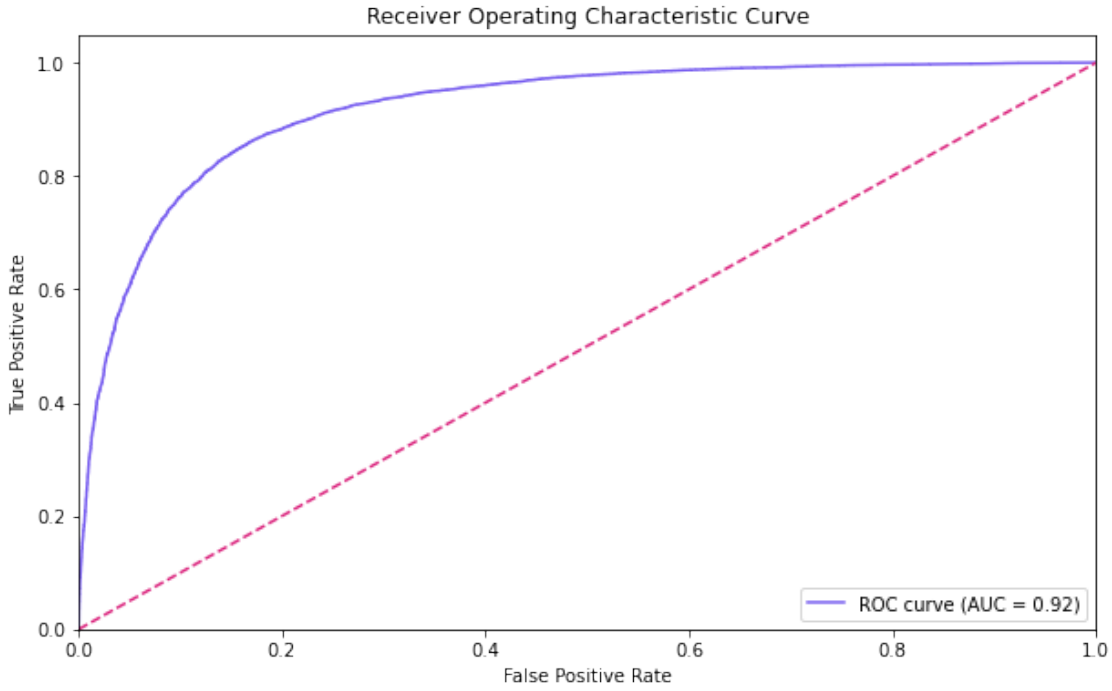


[6]:
```
# Generate predictions
predictions = rnn.predict(x_train[0:8])
print(predictions)

for i in range(2):
  INDEX_FROM=3   # word index offset
  word_to_id = imdb.get_word_index()
  word_to_id = {k:(v+INDEX_FROM) for k,v in word_to_id.items()}
  word_to_id["<PAD>"] = 0
  word_to_id["<START>"] = 1
  word_to_id["<UNK>"] = 2
  word_to_id["<UNUSED>"] = 3

  id_to_word = {value:key for key,value in word_to_id.items()}
  print('==================================================')
  print(f'Sample = {i} | Length = {len(x_test[i])}')
  print('==================================================')
  print(' '.join(id_to_word[id] for id in x_test[i] ))
```

```
[[0.99616086]
 [0.05577141]
 [0.00283012]
 [0.99280834]
 [0.01235956]
 [0.04756054]
 [0.996265  ]
 [0.00696427]]
==================================================
Sample = 0 | Length = 128
==================================================
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <UNK> please give <UNK> one
<UNK> miss <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> rest <UNK> <UNK> cast rendered
terrible performances <UNK> show <UNK> flat flat flat <UNK> <UNK> <UNK> don't
know how michael madison could have allowed <UNK> one on his plate he almost
seemed <UNK> know <UNK> wasn't going <UNK> work out <UNK> his performance <UNK>
quite <UNK> so all you madison fans give <UNK> <UNK> miss
==================================================
Sample = 1 | Length = 128
==================================================
young man regular altman player michael murphy has <UNK> small part <UNK> <UNK>
moody set fits <UNK> content <UNK> <UNK> story very well <UNK> short <UNK> movie
<UNK> <UNK> powerful study <UNK> loneliness sexual <UNK> <UNK> desperation be
patient <UNK> up <UNK> atmosphere <UNK> pay attention <UNK> <UNK> wonderfully
```

```
written script <UNK> <UNK> <UNK> praise robert altman <UNK> <UNK> one <UNK> his
many films <UNK> deals <UNK> unconventional fascinating subject matter <UNK>
film <UNK> disturbing but it's sincere <UNK> it's sure <UNK> <UNK> <UNK> strong
emotional response from <UNK> viewer if you want <UNK> see an unusual film some
might even say bizarre <UNK> <UNK> worth <UNK> time <UNK> <UNK> unfortunately
it's very difficult <UNK> find <UNK> video stores you may have <UNK> buy <UNK>
off <UNK> internet
```

[7]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import numpy as np

y_hat = rnn.predict(x_test)

# Compute ROC curve and ROC area for each class
fpr, tpr, thresholds = roc_curve(y_test, y_hat)
roc_auc = auc(fpr, tpr)

fig = plt.figure(figsize=(10,6))
plt.plot(fpr, tpr, color='#785ef0', label='ROC curve (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='#dc267f', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic Curve')
plt.legend(loc="lower right")
plt.savefig('ch.13.rnn.imdb.roc.png', dpi=350, bbox_inches='tight')
plt.show()

optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Threshold value is:", optimal_threshold)
y_pred = np.where(y_hat>=optimal_threshold, 1, 0)
print(balanced_accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```




```
Threshold value is: 0.53164583
0.84596
[[10401  2099]
 [ 1752 10748]]
```

##Long Short-Term Memory Models

[8]:
```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Embedding, BatchNormalization
from tensorflow.keras.layers import Dense, Activation, Input, Dropout
from keras.datasets import imdb
from keras.preprocessing import sequence
import numpy as np

seqnc_lngth =  128
embddng_dim = 64
vocab_size = 10000

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size,␣
 ↪skip_top=20)
x_train = sequence.pad_sequences(x_train, maxlen=seqnc_lngth)
x_test = sequence.pad_sequences(x_test, maxlen=seqnc_lngth)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

```
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# now with batch norm
inpt_vec = Input(shape=(seqnc_lngth,))
l1 = Embedding(vocab_size, embddng_dim, input_length=seqnc_lngth)(inpt_vec)
l2 = Dropout(0.3)(l1)
l3 = LSTM(32)(l2)
l4 = BatchNormalization()(l3)
l5 = Dropout(0.2)(l4)
output = Dense(1, activation='sigmoid')(l5)

# model that takes input and encodes it into the latent space
lstm = Model(inpt_vec, output)

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

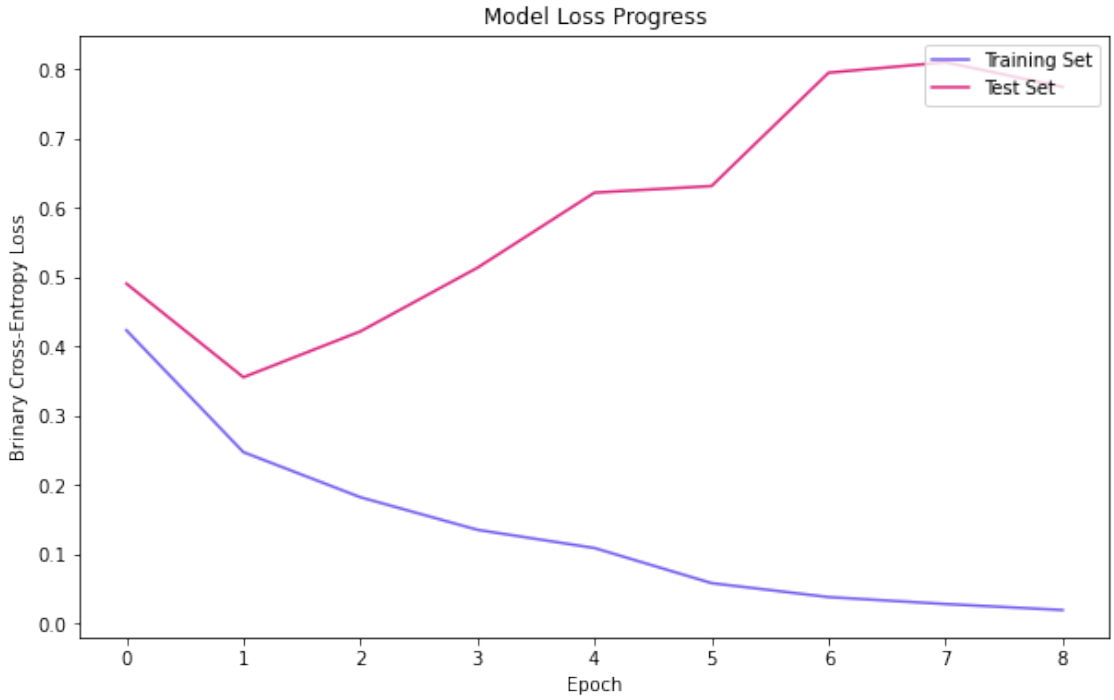
```
C:\Users\Nouhad\.conda\envs\cosc4337\lib\site-
packages\keras\datasets\imdb.py:101: VisibleDeprecationWarning: Creating an
ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-
tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
C:\Users\Nouhad\.conda\envs\cosc4337\lib\site-
packages\keras\datasets\imdb.py:102: 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
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])

x_train shape: (25000, 128)
x_test shape: (25000, 128)
25000 train samples
25000 test samples
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 128)]             0
_________________________________________________________________
embedding_2 (Embedding)      (None, 128, 64)           640000
_________________________________________________________________
dropout_2 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
```

```
lstm (LSTM)                  (None, 32)                12416
_________________________________________________________________
batch_normalization_1 (Batch (None, 32)                128
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 652,577
Trainable params: 652,513
Non-trainable params: 64
_________________________________________________________________
```

[9]:
```
# Fitting the LSTM to the data

from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3,
                              min_delta=1e-4, mode='min', verbose=1)

stop_alg = EarlyStopping(monitor='val_loss', patience=7,
                         restore_best_weights=True, verbose=1)

hist = lstm.fit(x_train, y_train, batch_size=100, epochs=1000,
                   callbacks=[stop_alg, reduce_lr], shuffle=True,
                   validation_data=(x_test, y_test))

lstm.save_weights("lstm.hdf5")

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,6))
plt.plot(hist.history['loss'], color='#785ef0')
plt.plot(hist.history['val_loss'], color='#dc267f')
plt.title('Model Loss Progress')
plt.ylabel('Brinary Cross-Entropy Loss')
plt.xlabel('Epoch')
plt.legend(['Training Set', 'Test Set'], loc='upper right')
plt.savefig('ch.13.lstm.imdb.loss.png', dpi=350, bbox_inches='tight')
plt.show()
```

```
Epoch 1/1000
250/250 [==============================] - 27s 107ms/step - loss: 0.4233 -
accuracy: 0.7954 - val_loss: 0.4906 - val_accuracy: 0.8360
Epoch 2/1000
250/250 [==============================] - 26s 105ms/step - loss: 0.2473 -
accuracy: 0.8997 - val_loss: 0.3555 - val_accuracy: 0.8472
Epoch 3/1000
```

```
250/250 [==============================] - 24s 98ms/step - loss: 0.1822 -
accuracy: 0.9289 - val_loss: 0.4217 - val_accuracy: 0.8455
Epoch 4/1000
250/250 [==============================] - 26s 104ms/step - loss: 0.1352 -
accuracy: 0.9496 - val_loss: 0.5136 - val_accuracy: 0.8419
Epoch 5/1000
250/250 [==============================] - ETA: 0s - loss: 0.1089 - accuracy:
0.9600
Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
250/250 [==============================] - 26s 103ms/step - loss: 0.1089 -
accuracy: 0.9600 - val_loss: 0.6220 - val_accuracy: 0.8188
Epoch 6/1000
250/250 [==============================] - 25s 99ms/step - loss: 0.0582 -
accuracy: 0.9796 - val_loss: 0.6315 - val_accuracy: 0.8345
Epoch 7/1000
250/250 [==============================] - 26s 105ms/step - loss: 0.0381 -
accuracy: 0.9868 - val_loss: 0.7952 - val_accuracy: 0.8212
Epoch 8/1000
250/250 [==============================] - ETA: 0s - loss: 0.0280 - accuracy:
0.9917
Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
250/250 [==============================] - 32s 126ms/step - loss: 0.0280 -
accuracy: 0.9917 - val_loss: 0.8103 - val_accuracy: 0.8299
Epoch 9/1000
250/250 [==============================] - ETA: 0s - loss: 0.0194 - accuracy:
0.9946Restoring model weights from the end of the best epoch.
250/250 [==============================] - 32s 128ms/step - loss: 0.0194 -
accuracy: 0.9946 - val_loss: 0.7747 - val_accuracy: 0.8308
Epoch 00009: early stopping
```



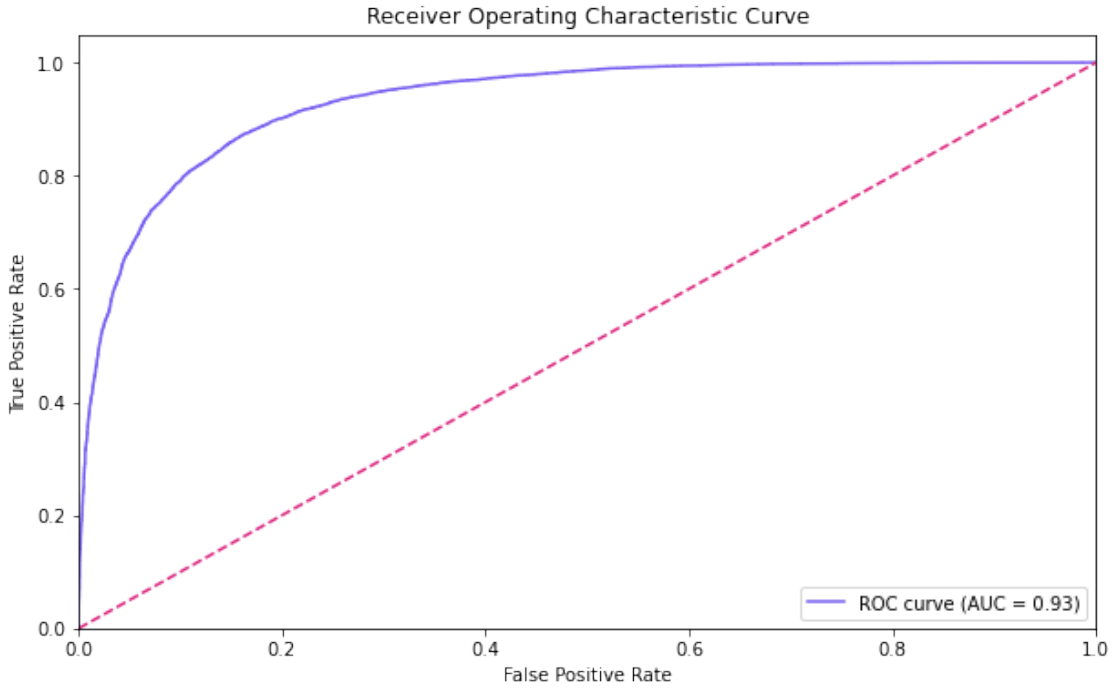
[10]:
```
# Generate predictions
predictions = lstm.predict(x_train[0:8])
print(predictions)

for i in range(2):
  INDEX_FROM=3   # word index offset
  word_to_id = imdb.get_word_index()
  word_to_id = {k:(v+INDEX_FROM) for k,v in word_to_id.items()}
  word_to_id["<PAD>"] = 0
  word_to_id["<START>"] = 1
  word_to_id["<UNK>"] = 2
  word_to_id["<UNUSED>"] = 3

  id_to_word = {value:key for key,value in word_to_id.items()}
  print('==================================================')
  print(f'Sample = {i} | Length = {len(x_test[i])}')
  print('==================================================')
  print(' '.join(id_to_word[id] for id in x_test[i] ))
```

```
[[0.9082912 ]
 [0.07670042]
 [0.06041279]
 [0.6580002 ]
 [0.03603771]
 [0.25771946]
```

```
 [0.671107  ]
 [0.02367324]]
================================================
Sample = 0 | Length = 128
================================================
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <UNK> please give <UNK> one
<UNK> miss <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> rest <UNK> <UNK> cast rendered
terrible performances <UNK> show <UNK> flat flat flat <UNK> <UNK> <UNK> don't
know how michael madison could have allowed <UNK> one on his plate he almost
seemed <UNK> know <UNK> wasn't going <UNK> work out <UNK> his performance <UNK>
quite <UNK> so all you madison fans give <UNK> <UNK> miss
================================================
Sample = 1 | Length = 128
================================================
young man regular altman player michael murphy has <UNK> small part <UNK> <UNK>
moody set fits <UNK> content <UNK> <UNK> story very well <UNK> short <UNK> movie
<UNK> <UNK> powerful study <UNK> loneliness sexual <UNK> <UNK> desperation be
patient <UNK> up <UNK> atmosphere <UNK> pay attention <UNK> <UNK> wonderfully
written script <UNK> <UNK> <UNK> praise robert altman <UNK> <UNK> one <UNK> his
many films <UNK> deals <UNK> unconventional fascinating subject matter <UNK>
film <UNK> disturbing but it's sincere <UNK> it's sure <UNK> <UNK> <UNK> strong
emotional response from <UNK> viewer if you want <UNK> see an unusual film some
might even say bizarre <UNK> <UNK> worth <UNK> time <UNK> <UNK> unfortunately
it's very difficult <UNK> find <UNK> video stores you may have <UNK> buy <UNK>
off <UNK> internet
```

[11]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
import numpy as np

y_hat = lstm.predict(x_test)

# Compute ROC curve and ROC area for each class
fpr, tpr, thresholds = roc_curve(y_test, y_hat)
roc_auc = auc(fpr, tpr)

fig = plt.figure(figsize=(10,6))
plt.plot(fpr, tpr, color='#785ef0', label='ROC curve (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='#dc267f', linestyle='--')
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic Curve')
plt.legend(loc="lower right")
plt.savefig('lstm.imdb.roc.png', dpi=350, bbox_inches='tight')
plt.show()

optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Threshold value is:", optimal_threshold)
y_pred = np.where(y_hat>=optimal_threshold, 1, 0)
print(balanced_accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```




```
Threshold value is: 0.3844713
0.85564
[[10471  2029]
 [ 1580 10920]]
```

##Sequence to Vector Models ###Unsupervised Model

[12]:
```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Activation, Input
from tensorflow.keras.layers import BatchNormalization, Dropout
```

```
from tensorflow.keras.layers import Embedding, LSTM
from tensorflow.keras.layers import RepeatVector, TimeDistributed
from tensorflow.keras.datasets import mnist
import numpy as np
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping

seqnc_lngth =  28
ltnt_dim = 2

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

inpt_vec = Input(shape=(seqnc_lngth, seqnc_lngth,))
l1 = Dropout(0.1)(inpt_vec)
l2 = LSTM(seqnc_lngth, activation='tanh',
          recurrent_activation='sigmoid')(l1)
l3 = BatchNormalization()(l2)
l4 = Dropout(0.1)(l3)
l5 = Dense(ltnt_dim, activation='sigmoid')(l4)

# model that takes input and encodes it into the latent space
encoder = Model(inpt_vec, l5)

l6 = RepeatVector(seqnc_lngth)(l5)
l7 = LSTM(seqnc_lngth, activation='tanh', recurrent_activation='sigmoid',
          return_sequences=True)(l6)
l8 = BatchNormalization()(l7)
l9 = TimeDistributed(Dense(seqnc_lngth, activation='sigmoid'))(l8)

autoencoder = Model(inpt_vec, l9)

autoencoder.compile(loss='binary_crossentropy', optimizer='adam')
autoencoder.summary()

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5,
                              min_delta=1e-4, mode='min', verbose=1)

stop_alg = EarlyStopping(monitor='val_loss', patience=15,
                         restore_best_weights=True, verbose=1)

hist = autoencoder.fit(x_train, x_train, batch_size=100, epochs=1000,
                       callbacks=[stop_alg, reduce_lr], shuffle=True,
```

```
                    validation_data=(x_test, x_test))
```

```
x_train shape: (60000, 28, 28)
x_test shape: (10000, 28, 28)
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 28, 28)]          0
_________________________________________________________________
dropout_4 (Dropout)          (None, 28, 28)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 28)                6384
_________________________________________________________________
batch_normalization_2 (Batch (None, 28)                112
_________________________________________________________________
dropout_5 (Dropout)          (None, 28)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 58
_________________________________________________________________
repeat_vector (RepeatVector) (None, 28, 2)             0
_________________________________________________________________
lstm_2 (LSTM)                (None, 28, 28)            3472
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28)            112
_________________________________________________________________
time_distributed (TimeDistri (None, 28, 28)            812
=================================================================
Total params: 10,950
Trainable params: 10,838
Non-trainable params: 112
_________________________________________________________________
Epoch 1/1000
600/600 [==============================] - 23s 39ms/step - loss: 0.3548 -
val_loss: 0.2431
Epoch 2/1000
600/600 [==============================] - 24s 40ms/step - loss: 0.2293 -
val_loss: 0.2250
Epoch 3/1000
600/600 [==============================] - 20s 34ms/step - loss: 0.2215 -
val_loss: 0.2154
Epoch 4/1000
600/600 [==============================] - 22s 37ms/step - loss: 0.2173 -
val_loss: 0.2123
Epoch 5/1000
600/600 [==============================] - 21s 36ms/step - loss: 0.2146 -
val_loss: 0.2095
```

```
Epoch 6/1000
600/600 [==============================] - 22s 37ms/step - loss: 0.2127 -
val_loss: 0.2075
Epoch 7/1000
600/600 [==============================] - 17s 29ms/step - loss: 0.2112 -
val_loss: 0.2061
Epoch 8/1000
600/600 [==============================] - 18s 30ms/step - loss: 0.2101 -
val_loss: 0.2044
Epoch 9/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.2085 -
val_loss: 0.2033
Epoch 10/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.2076 -
val_loss: 0.2022
Epoch 11/1000
600/600 [==============================] - 18s 30ms/step - loss: 0.2065 -
val_loss: 0.2012
Epoch 12/1000
600/600 [==============================] - 18s 31ms/step - loss: 0.2050 -
val_loss: 0.1998
Epoch 13/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.2042 -
val_loss: 0.1977
Epoch 14/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.2033 -
val_loss: 0.1983
Epoch 15/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.2023 -
val_loss: 0.1965
Epoch 16/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.2010 -
val_loss: 0.1954
Epoch 17/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.2005 -
val_loss: 0.1945
Epoch 18/1000
600/600 [==============================] - 15s 26ms/step - loss: 0.1999 -
val_loss: 0.1939
Epoch 19/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1992 -
val_loss: 0.1929
Epoch 20/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1989 -
val_loss: 0.1929
Epoch 21/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1990 -
val_loss: 0.1934
```

```
Epoch 22/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1984 -
val_loss: 0.1930
Epoch 23/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1980 -
val_loss: 0.1953
Epoch 24/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1974 -
val_loss: 0.1913
Epoch 25/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1972 -
val_loss: 0.1900
Epoch 26/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1970 -
val_loss: 0.1907
Epoch 27/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1966 -
val_loss: 0.1897
Epoch 28/1000
600/600 [==============================] - 18s 31ms/step - loss: 0.1960 -
val_loss: 0.1907
Epoch 29/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1958 -
val_loss: 0.1893
Epoch 30/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1957 -
val_loss: 0.1937
Epoch 31/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1960 -
val_loss: 0.1891
Epoch 32/1000
600/600 [==============================] - 16s 26ms/step - loss: 0.1952 -
val_loss: 0.1897
Epoch 33/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1951 -
val_loss: 0.1897
Epoch 34/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1948 -
val_loss: 0.1875
Epoch 35/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1944 -
val_loss: 0.1887
Epoch 36/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1941 -
val_loss: 0.1886
Epoch 37/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1944 -
val_loss: 0.1877
```

```
Epoch 38/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1939 -
val_loss: 0.1882
Epoch 39/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1939
Epoch 00039: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
600/600 [==============================] - 16s 27ms/step - loss: 0.1939 -
val_loss: 0.1877
Epoch 40/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1928 -
val_loss: 0.1871
Epoch 41/1000
600/600 [==============================] - 17s 29ms/step - loss: 0.1931 -
val_loss: 0.1858
Epoch 42/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1930 -
val_loss: 0.1857
Epoch 43/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1924 -
val_loss: 0.1856
Epoch 44/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1923 -
val_loss: 0.1855
Epoch 45/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1920 -
val_loss: 0.1859
Epoch 46/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1920 -
val_loss: 0.1848
Epoch 47/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1933 -
val_loss: 0.1851
Epoch 48/1000
600/600 [==============================] - 17s 28ms/step - loss: 0.1924 -
val_loss: 0.1856
Epoch 49/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1922 -
val_loss: 0.1873
Epoch 50/1000
600/600 [==============================] - 16s 26ms/step - loss: 0.1921 -
val_loss: 0.1852
Epoch 51/1000
600/600 [==============================] - 16s 26ms/step - loss: 0.1923 -
val_loss: 0.1844
Epoch 52/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1917 -
val_loss: 0.1838
Epoch 53/1000
```

```
600/600 [==============================] - 15s 24ms/step - loss: 0.1916 -
val_loss: 0.1844
Epoch 54/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1919 -
val_loss: 0.1843
Epoch 55/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1916 -
val_loss: 0.1851
Epoch 56/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1920 -
val_loss: 0.1844
Epoch 57/1000
598/600 [============================>.] - ETA: 0s - loss: 0.1913
Epoch 00057: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
600/600 [==============================] - 14s 24ms/step - loss: 0.1913 -
val_loss: 0.1840
Epoch 58/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1911 -
val_loss: 0.1837
Epoch 59/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1909 -
val_loss: 0.1832
Epoch 60/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1907 -
val_loss: 0.1833
Epoch 61/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1905 -
val_loss: 0.1833
Epoch 62/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1903 -
val_loss: 0.1828
Epoch 63/1000
600/600 [==============================] - 15s 24ms/step - loss: 0.1907 -
val_loss: 0.1839
Epoch 64/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1903 -
val_loss: 0.1829
Epoch 65/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1906 -
val_loss: 0.1828
Epoch 66/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1906 -
val_loss: 0.1829
Epoch 67/1000
600/600 [==============================] - ETA: 0s - loss: 0.1903
Epoch 00067: ReduceLROnPlateau reducing learning rate to 0.00012500000059371814.
600/600 [==============================] - 14s 23ms/step - loss: 0.1903 -
val_loss: 0.1831
```

```
Epoch 68/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1899 -
val_loss: 0.1827
Epoch 69/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1902 -
val_loss: 0.1827
Epoch 70/1000
600/600 [==============================] - 15s 24ms/step - loss: 0.1903 -
val_loss: 0.1827
Epoch 71/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1898 -
val_loss: 0.1826
Epoch 72/1000
600/600 [==============================] - 15s 26ms/step - loss: 0.1905 -
val_loss: 0.1825
Epoch 73/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1900 -
val_loss: 0.1826
Epoch 74/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1900 -
val_loss: 0.1823
Epoch 75/1000
600/600 [==============================] - 13s 22ms/step - loss: 0.1901 -
val_loss: 0.1822
Epoch 76/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1898 -
val_loss: 0.1824
Epoch 77/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1898 -
val_loss: 0.1822
Epoch 78/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1899 -
val_loss: 0.1822
Epoch 79/1000
600/600 [==============================] - ETA: 0s - loss: 0.1900
Epoch 00079: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
600/600 [==============================] - 14s 23ms/step - loss: 0.1900 -
val_loss: 0.1822
Epoch 80/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1899 -
val_loss: 0.1821
Epoch 81/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1896 -
val_loss: 0.1821
Epoch 82/1000
600/600 [==============================] - 13s 22ms/step - loss: 0.1897 -
val_loss: 0.1822
Epoch 83/1000
```

```
600/600 [==============================] - 14s 23ms/step - loss: 0.1898 -
val_loss: 0.1820
Epoch 84/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1900 -
val_loss: 0.1821
Epoch 85/1000
600/600 [==============================] - ETA: 0s - loss: 0.1897
Epoch 00085: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
600/600 [==============================] - 14s 24ms/step - loss: 0.1897 -
val_loss: 0.1820
Epoch 86/1000
600/600 [==============================] - 15s 24ms/step - loss: 0.1896 -
val_loss: 0.1820
Epoch 87/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1893 -
val_loss: 0.1820
Epoch 88/1000
600/600 [==============================] - 15s 24ms/step - loss: 0.1897 -
val_loss: 0.1820
Epoch 89/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1895 -
val_loss: 0.1820
Epoch 90/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1895 -
val_loss: 0.1820
Epoch 91/1000
600/600 [==============================] - 18s 30ms/step - loss: 0.1898 -
val_loss: 0.1819
Epoch 92/1000
600/600 [==============================] - 16s 27ms/step - loss: 0.1895 -
val_loss: 0.1820
Epoch 93/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1897 -
val_loss: 0.1820
Epoch 94/1000
600/600 [==============================] - 15s 26ms/step - loss: 0.1896 -
val_loss: 0.1819
Epoch 95/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1893 -
val_loss: 0.1819
Epoch 96/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1896
Epoch 00096: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
600/600 [==============================] - 16s 26ms/step - loss: 0.1896 -
val_loss: 0.1820
Epoch 97/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1896 -
val_loss: 0.1819
```

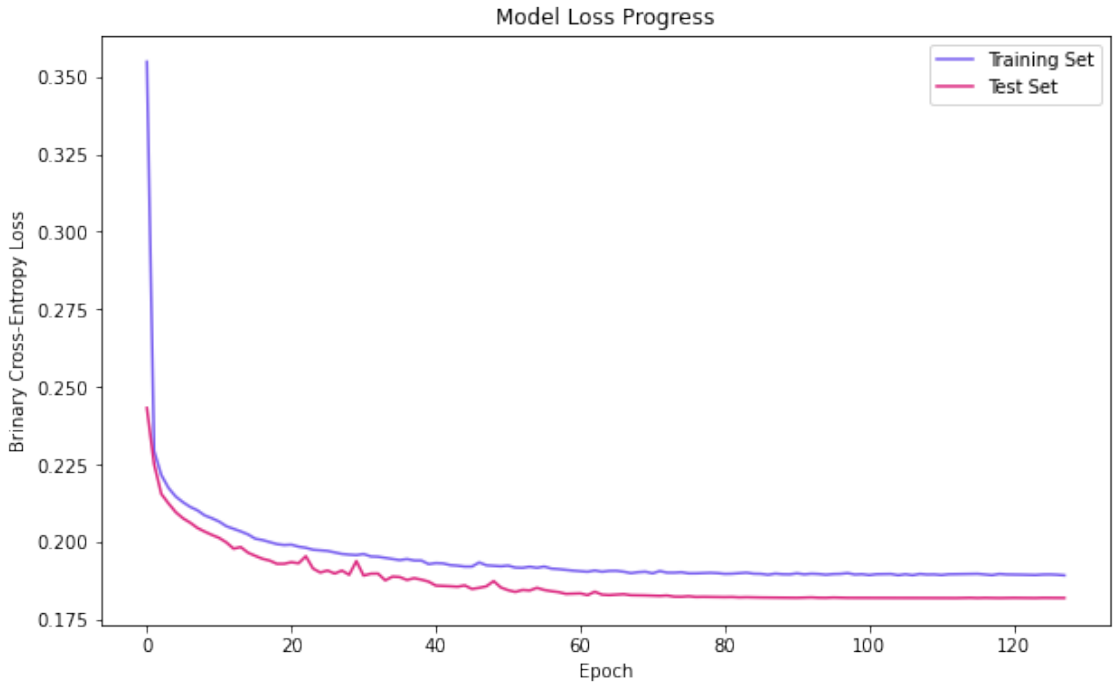
```
Epoch 98/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1899 -
val_loss: 0.1819
Epoch 99/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1894 -
val_loss: 0.1819
Epoch 100/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1895 -
val_loss: 0.1819
Epoch 101/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1892-
Epoch 00101: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
600/600 [==============================] - 14s 24ms/step - loss: 0.1892 -
val_loss: 0.1818
Epoch 102/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1895 -
val_loss: 0.1819
Epoch 103/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1895 -
val_loss: 0.1819
Epoch 104/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1896 -
val_loss: 0.1818
Epoch 105/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1892 -
val_loss: 0.1818
Epoch 106/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1895
Epoch 00106: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
600/600 [==============================] - 14s 24ms/step - loss: 0.1895 -
val_loss: 0.1819
Epoch 107/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1892 -
val_loss: 0.1819
Epoch 108/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1896 -
val_loss: 0.1819
Epoch 109/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1894 -
val_loss: 0.1819
Epoch 110/1000
600/600 [==============================] - 15s 26ms/step - loss: 0.1894 -
val_loss: 0.1818
Epoch 111/1000
598/600 [============================>.] - ETA: 0s - loss: 0.1892
Epoch 00111: ReduceLROnPlateau reducing learning rate to 1.9531250927684596e-06.
600/600 [==============================] - 14s 24ms/step - loss: 0.1892 -
val_loss: 0.1818
```

```
Epoch 112/1000
600/600 [==============================] - 15s 26ms/step - loss: 0.1895 -
val_loss: 0.1818
Epoch 113/1000
600/600 [==============================] - 15s 26ms/step - loss: 0.1895 -
val_loss: 0.1818
Epoch 114/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1896 -
val_loss: 0.1819
Epoch 115/1000
600/600 [==============================] - 15s 24ms/step - loss: 0.1896 -
val_loss: 0.1819
Epoch 116/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1896
Epoch 00116: ReduceLROnPlateau reducing learning rate to 9.765625463842298e-07.
600/600 [==============================] - 15s 25ms/step - loss: 0.1896 -
val_loss: 0.1818
Epoch 117/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1894 -
val_loss: 0.1819
Epoch 118/1000
600/600 [==============================] - 15s 24ms/step - loss: 0.1892 -
val_loss: 0.1818
Epoch 119/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1896 -
val_loss: 0.1818
Epoch 120/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1894 -
val_loss: 0.1819
Epoch 121/1000
600/600 [==============================] - ETA: 0s - loss: 0.1894
Epoch 00121: ReduceLROnPlateau reducing learning rate to 4.882812731921149e-07.
600/600 [==============================] - 15s 25ms/step - loss: 0.1894 -
val_loss: 0.1819
Epoch 122/1000
600/600 [==============================] - 14s 24ms/step - loss: 0.1894 -
val_loss: 0.1818
Epoch 123/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1893 -
val_loss: 0.1818
Epoch 124/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1893 -
val_loss: 0.1818
Epoch 125/1000
600/600 [==============================] - 14s 23ms/step - loss: 0.1894 -
val_loss: 0.1819
Epoch 126/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1894
```

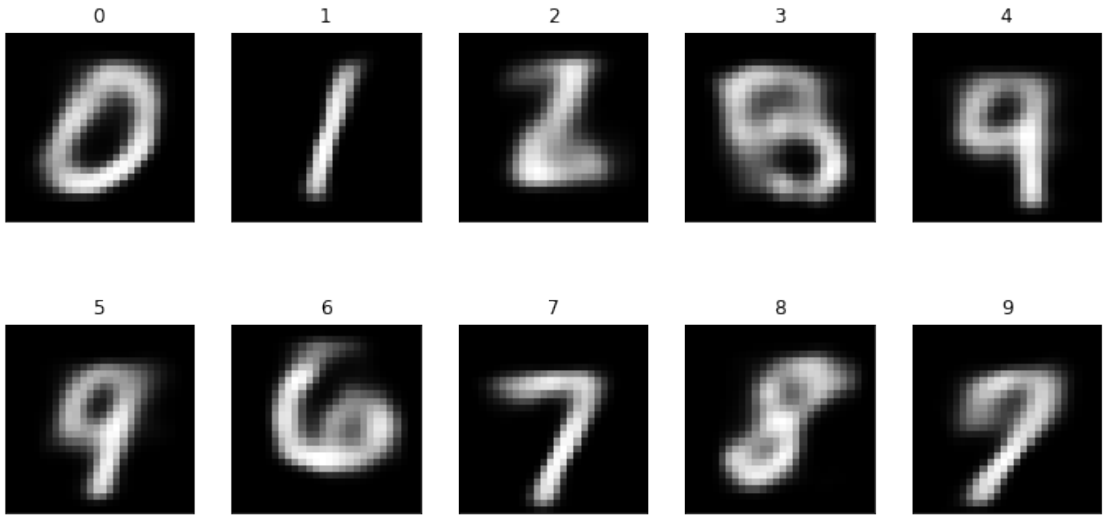
```
Epoch 00126: ReduceLROnPlateau reducing learning rate to 2.4414063659605745e-07.
600/600 [==============================] - 15s 25ms/step - loss: 0.1894 -
val_loss: 0.1819
Epoch 127/1000
600/600 [==============================] - 15s 25ms/step - loss: 0.1893 -
val_loss: 0.1818
Epoch 128/1000
600/600 [==============================] - ETA: 0s - loss: 0.1891Restoring model
weights from the end of the best epoch.
600/600 [==============================] - 14s 24ms/step - loss: 0.1891 -
val_loss: 0.1818
Epoch 00128: early stopping
```

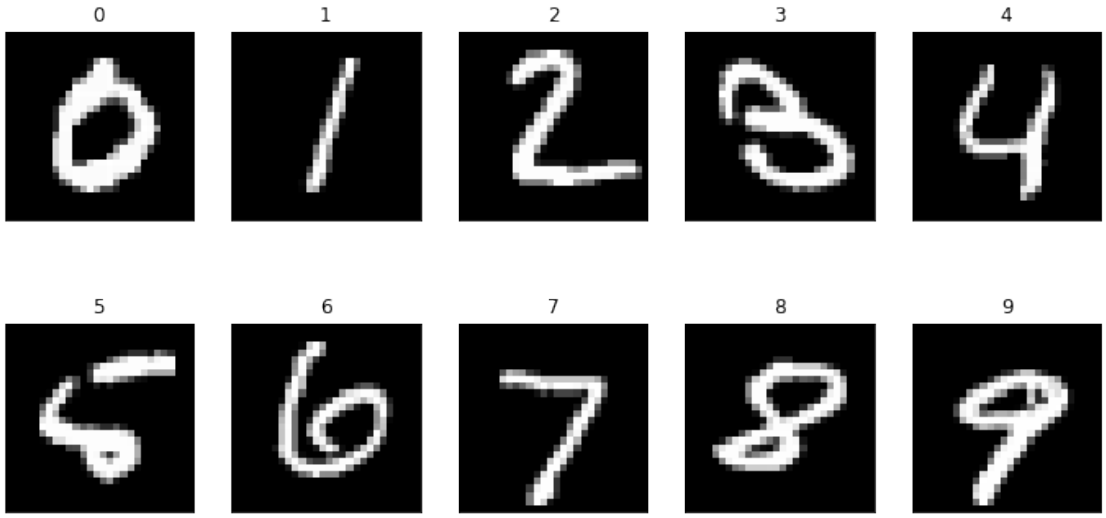
###Results

[13]:
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,6))
plt.plot(hist.history['loss'], color='#785ef0')
plt.plot(hist.history['val_loss'], color='#dc267f')
plt.title('Model Loss Progress')
plt.ylabel('Brinary Cross-Entropy Loss')
plt.xlabel('Epoch')
plt.legend(['Training Set', 'Test Set'], loc='upper right')
plt.savefig('lstm.mnist.loss.png', dpi=350, bbox_inches='tight')
plt.show()
```



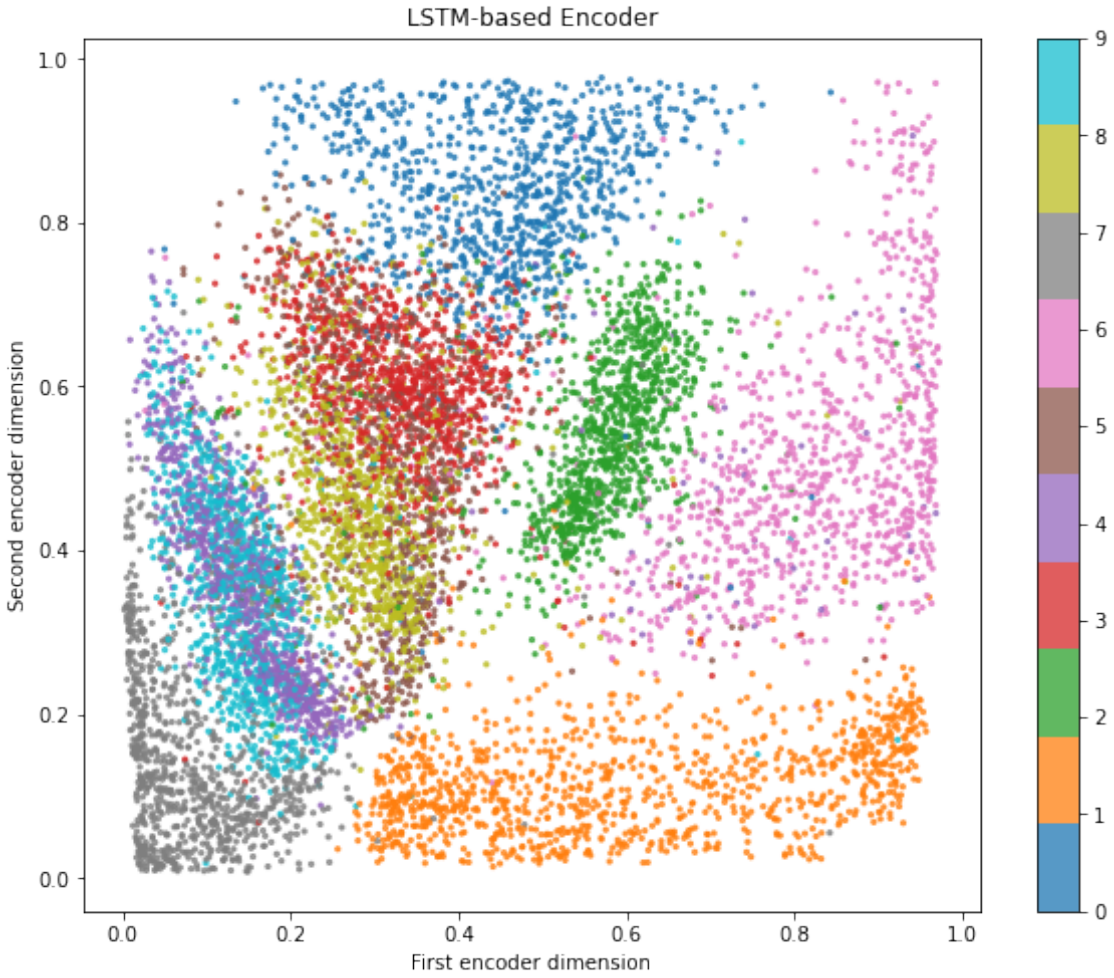
[14]:
```
import matplotlib.pyplot as plt
import numpy as np

encdd = encoder.predict(x_test)
x_hat = autoencoder.predict(x_test)

smp_idx = [3,2,1,18,4,8,11,0,61,9]
plt.figure(figsize=(12,6))
for i, (img, y) in enumerate(zip(x_hat[smp_idx].reshape(10, 28, 28),␣
 ↪y_test[smp_idx])):
  plt.subplot(2,5,i+1)
  plt.imshow(img, cmap='gray')
  plt.xticks([])
  plt.yticks([])
  plt.title(y)
plt.savefig('ch.13.ae.lstm.mnist.reconstructed.png', bbox_inches='tight',␣
 ↪dpi=350)
plt.show()

plt.figure(figsize=(12,6))
for i, (img, y) in enumerate(zip(x_test[smp_idx].reshape(10, 28, 28),␣
 ↪y_test[smp_idx])):
  plt.subplot(2,5,i+1)
  plt.imshow(img, cmap='gray')
  plt.xticks([])
  plt.yticks([])
  plt.title(y)
plt.savefig('lstm.mnist.original.png', bbox_inches='tight', dpi=350)
plt.show()
```





[15]:
```
import matplotlib.pyplot as plt

y_ = list(map(int, y_test))
X_ = encdd

print(X_.shape)

plt.figure(figsize=(10,8))
plt.title('LSTM-based Encoder')
plt.scatter(X_[:,0], X_[:,1], s=5.0, c=y_, alpha=0.75, cmap='tab10')
plt.xlabel('First encoder dimension')
plt.ylabel('Second encoder dimension')
plt.colorbar()
plt.savefig('lstm.png', bbox_inches='tight', dpi=350)
```

```
(10000, 2)
```



##Vector to Sequence Models ###Bidirectional LSTM ###Implementation and Results

[16]:
```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Activation, Input
from tensorflow.keras.layers import BatchNormalization, Dropout
from tensorflow.keras.layers import Bidirectional, LSTM
from tensorflow.keras.layers import RepeatVector, TimeDistributed
from tensorflow.keras.datasets import mnist
import numpy as np
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping

seqnc_lngth =  28
ltnt_dim = 100

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255.
```

```
x_test = x_test.astype('float32') / 255.

print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

inpt_vec = Input(shape=(seqnc_lngth, seqnc_lngth,))
l1 = Dropout(0.5)(inpt_vec)
l2 = Bidirectional(LSTM(seqnc_lngth, activation='tanh',
                        recurrent_activation='sigmoid'))(l1)
l3 = BatchNormalization()(l2)
l4 = Dropout(0.5)(l3)
l5 = Dense(ltnt_dim, activation='sigmoid')(l4)

# model that takes input and encodes it into the latent space
encoder = Model(inpt_vec, l5, name='encoder')
encoder.summary()

ltnt_vec = Input(shape=(ltnt_dim,))
l6 = Dropout(0.1)(ltnt_vec)
l7 = RepeatVector(seqnc_lngth)(l6)
l8 = Bidirectional(LSTM(seqnc_lngth, activation='tanh',␣
 ↪recurrent_activation='sigmoid',
          return_sequences=True))(l7)
l9 = BatchNormalization()(l8)
l10 = TimeDistributed(Dense(seqnc_lngth, activation='sigmoid'))(l9)

decoder = Model(ltnt_vec, l10, name='decoder')
decoder.summary()

recon = decoder(encoder(inpt_vec))
autoencoder = Model(inpt_vec, recon, name='ae')

autoencoder.compile(loss='binary_crossentropy', optimizer='adam')
autoencoder.summary()

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5,
                              min_delta=1e-4, mode='min', verbose=1)

stop_alg = EarlyStopping(monitor='val_loss', patience=15,
                         restore_best_weights=True, verbose=1)

hist = autoencoder.fit(x_train, x_train, batch_size=100, epochs=1000,
                       callbacks=[stop_alg, reduce_lr], shuffle=True,
                       validation_data=(x_test, x_test))
```

```
x_train shape: (60000, 28, 28)
x_test shape: (10000, 28, 28)
```

```
Model: "encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 28, 28)]          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 28, 28)            0
_________________________________________________________________
bidirectional (Bidirectional (None, 56)                12768
_________________________________________________________________
batch_normalization_4 (Batch (None, 56)                224
_________________________________________________________________
dropout_7 (Dropout)          (None, 56)                0
_________________________________________________________________
dense_4 (Dense)              (None, 100)               5700
=================================================================
Total params: 18,692
Trainable params: 18,580
Non-trainable params: 112
_________________________________________________________________
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
dropout_8 (Dropout)          (None, 100)               0
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 28, 100)           0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 28, 56)            28896
_________________________________________________________________
batch_normalization_5 (Batch (None, 28, 56)            224
_________________________________________________________________
time_distributed_1 (TimeDist (None, 28, 28)            1596
=================================================================
Total params: 30,716
Trainable params: 30,604
Non-trainable params: 112
_________________________________________________________________
Model: "ae"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 28, 28)]          0
_________________________________________________________________
encoder (Functional)         (None, 100)               18692
_________________________________________________________________
```

```
decoder (Functional)         (None, 28, 28)            30716
=================================================================
Total params: 49,408
Trainable params: 49,184
Non-trainable params: 224
_________________________________________________________________
Epoch 1/1000
600/600 [==============================] - 24s 40ms/step - loss: 0.3135 -
val_loss: 0.2022
Epoch 2/1000
600/600 [==============================] - 23s 38ms/step - loss: 0.1911 -
val_loss: 0.1652
Epoch 3/1000
600/600 [==============================] - 23s 38ms/step - loss: 0.1774 -
val_loss: 0.1493
Epoch 4/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1706 -
val_loss: 0.1459
Epoch 5/1000
600/600 [==============================] - 24s 39ms/step - loss: 0.1663 -
val_loss: 0.1479
Epoch 6/1000
600/600 [==============================] - 24s 40ms/step - loss: 0.1636 -
val_loss: 0.1417
Epoch 7/1000
600/600 [==============================] - 24s 41ms/step - loss: 0.1612 -
val_loss: 0.1345
Epoch 8/1000
600/600 [==============================] - 24s 41ms/step - loss: 0.1595 -
val_loss: 0.1342
Epoch 9/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1581 -
val_loss: 0.1334
Epoch 10/1000
600/600 [==============================] - 27s 44ms/step - loss: 0.1572 -
val_loss: 0.1301
Epoch 11/1000
600/600 [==============================] - 27s 44ms/step - loss: 0.1560 -
val_loss: 0.1314
Epoch 12/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1551 -
val_loss: 0.1291
Epoch 13/1000
600/600 [==============================] - 25s 42ms/step - loss: 0.1543 -
val_loss: 0.1276
Epoch 14/1000
600/600 [==============================] - 24s 41ms/step - loss: 0.1536 -
val_loss: 0.1269
```

```
Epoch 15/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1530 -
val_loss: 0.1259
Epoch 16/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1521 -
val_loss: 0.1263
Epoch 17/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1517 -
val_loss: 0.1246
Epoch 18/1000
600/600 [==============================] - 25s 42ms/step - loss: 0.1513 -
val_loss: 0.1265
Epoch 19/1000
600/600 [==============================] - 24s 41ms/step - loss: 0.1508 -
val_loss: 0.1238
Epoch 20/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1504 -
val_loss: 0.1241
Epoch 21/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1499 -
val_loss: 0.1242
Epoch 22/1000
600/600 [==============================] - 25s 42ms/step - loss: 0.1497 -
val_loss: 0.1235
Epoch 23/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1491 -
val_loss: 0.1227
Epoch 24/1000
600/600 [==============================] - 30s 50ms/step - loss: 0.1488 -
val_loss: 0.1211
Epoch 25/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1485 -
val_loss: 0.1214
Epoch 26/1000
600/600 [==============================] - 24s 40ms/step - loss: 0.1483 -
val_loss: 0.1218
Epoch 27/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1478 -
val_loss: 0.1227
Epoch 28/1000
600/600 [==============================] - 34s 57ms/step - loss: 0.1473 -
val_loss: 0.1204
Epoch 29/1000
600/600 [==============================] - 27s 46ms/step - loss: 0.1472 -
val_loss: 0.1214
Epoch 30/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1470 -
val_loss: 0.1206
```

```
Epoch 31/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1467 -
val_loss: 0.1204
Epoch 32/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1464 -
val_loss: 0.1187
Epoch 33/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1463 -
val_loss: 0.1189
Epoch 34/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1462 -
val_loss: 0.1195
Epoch 35/1000
600/600 [==============================] - 26s 44ms/step - loss: 0.1456 -
val_loss: 0.1196
Epoch 36/1000
600/600 [==============================] - 33s 54ms/step - loss: 0.1454 -
val_loss: 0.1190
Epoch 37/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1452
Epoch 00037: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
600/600 [==============================] - 26s 44ms/step - loss: 0.1452 -
val_loss: 0.1198
Epoch 38/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1443 -
val_loss: 0.1169
Epoch 39/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1439 -
val_loss: 0.1170
Epoch 40/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1438 -
val_loss: 0.1169
Epoch 41/1000
600/600 [==============================] - 26s 44ms/step - loss: 0.1436 -
val_loss: 0.1167
Epoch 42/1000
600/600 [==============================] - 28s 47ms/step - loss: 0.1436 -
val_loss: 0.1169
Epoch 43/1000
600/600 [==============================] - 30s 50ms/step - loss: 0.1436 -
val_loss: 0.1161
Epoch 44/1000
600/600 [==============================] - 31s 52ms/step - loss: 0.1434 -
val_loss: 0.1159
Epoch 45/1000
600/600 [==============================] - 27s 45ms/step - loss: 0.1432 -
val_loss: 0.1157
Epoch 46/1000
```

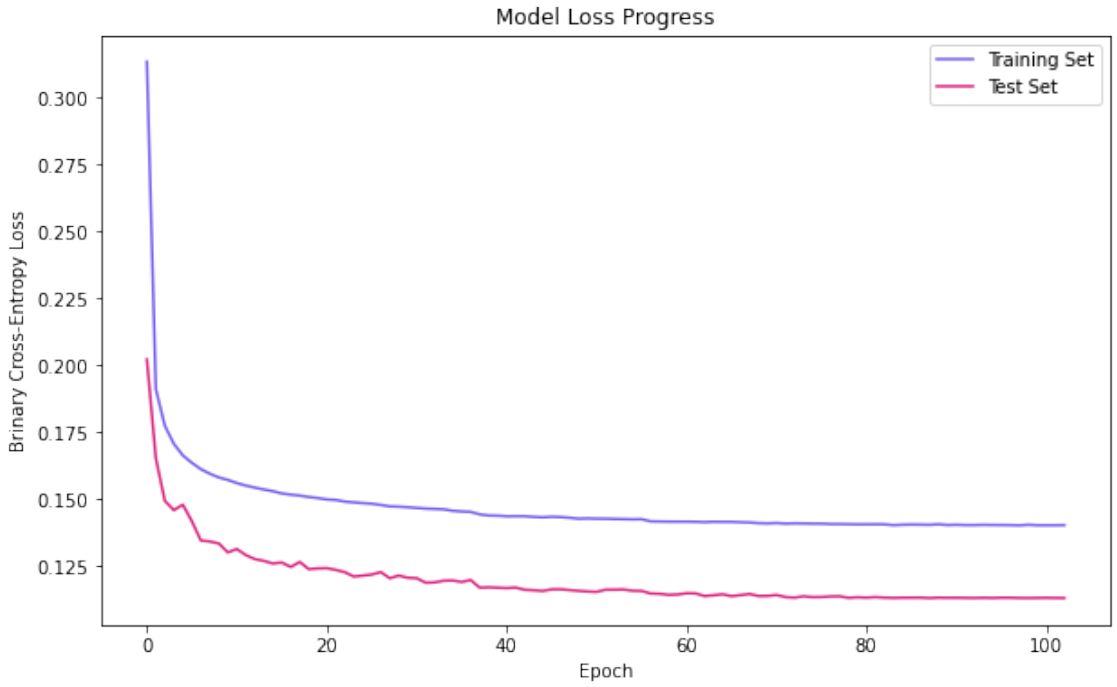
```
600/600 [==============================] - 26s 43ms/step - loss: 0.1434 -
val_loss: 0.1163
Epoch 47/1000
600/600 [==============================] - 26s 43ms/step - loss: 0.1433 -
val_loss: 0.1163
Epoch 48/1000
600/600 [==============================] - 24s 41ms/step - loss: 0.1430 -
val_loss: 0.1160
Epoch 49/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1427 -
val_loss: 0.1157
Epoch 50/1000
600/600 [==============================] - 23s 38ms/step - loss: 0.1428 -
val_loss: 0.1155
Epoch 51/1000
600/600 [==============================] - 23s 38ms/step - loss: 0.1427 -
val_loss: 0.1153
Epoch 52/1000
600/600 [==============================] - 23s 38ms/step - loss: 0.1427 -
val_loss: 0.1161
Epoch 53/1000
600/600 [==============================] - 23s 39ms/step - loss: 0.1426 -
val_loss: 0.1162
Epoch 54/1000
600/600 [==============================] - 25s 41ms/step - loss: 0.1425 -
val_loss: 0.1162
Epoch 55/1000
600/600 [==============================] - 23s 38ms/step - loss: 0.1424 -
val_loss: 0.1157
Epoch 56/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1425
Epoch 00056: ReduceLROnPlateau reducing learning rate to 0.00025000001187436283.
600/600 [==============================] - 22s 37ms/step - loss: 0.1425 -
val_loss: 0.1157
Epoch 57/1000
600/600 [==============================] - 21s 35ms/step - loss: 0.1417 -
val_loss: 0.1147
Epoch 58/1000
600/600 [==============================] - 20s 34ms/step - loss: 0.1416 -
val_loss: 0.1146
Epoch 59/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1416 -
val_loss: 0.1142
Epoch 60/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1416 -
val_loss: 0.1144
Epoch 61/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1416 -
```

```
val_loss: 0.1148
Epoch 62/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1415 -
val_loss: 0.1148
Epoch 63/1000
600/600 [==============================] - 19s 31ms/step - loss: 0.1414 -
val_loss: 0.1138
Epoch 64/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1416 -
val_loss: 0.1141
Epoch 65/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1415 -
val_loss: 0.1144
Epoch 66/1000
600/600 [==============================] - 19s 31ms/step - loss: 0.1415 -
val_loss: 0.1137
Epoch 67/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1414 -
val_loss: 0.1141
Epoch 68/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1413
Epoch 00068: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
600/600 [==============================] - 19s 32ms/step - loss: 0.1413 -
val_loss: 0.1145
Epoch 69/1000
600/600 [==============================] - 19s 31ms/step - loss: 0.1410 -
val_loss: 0.1138
Epoch 70/1000
600/600 [==============================] - 19s 31ms/step - loss: 0.1409 -
val_loss: 0.1138
Epoch 71/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1411 -
val_loss: 0.1141
Epoch 72/1000
600/600 [==============================] - 19s 31ms/step - loss: 0.1408 -
val_loss: 0.1134
Epoch 73/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1409 -
val_loss: 0.1132
Epoch 74/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1409 -
val_loss: 0.1137
Epoch 75/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1408 -
val_loss: 0.1134
Epoch 76/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1408 -
val_loss: 0.1134
```

```
Epoch 77/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1407 -
val_loss: 0.1136
Epoch 78/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1407
Epoch 00078: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
600/600 [==============================] - 20s 33ms/step - loss: 0.1407 -
val_loss: 0.1137
Epoch 79/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1406 -
val_loss: 0.1131
Epoch 80/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1406 -
val_loss: 0.1133
Epoch 81/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1406 -
val_loss: 0.1132
Epoch 82/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1406 -
val_loss: 0.1134
Epoch 83/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1406
Epoch 00083: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
600/600 [==============================] - 20s 33ms/step - loss: 0.1406 -
val_loss: 0.1132
Epoch 84/1000
600/600 [==============================] - 22s 36ms/step - loss: 0.1403 -
val_loss: 0.1131
Epoch 85/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1404 -
val_loss: 0.1132
Epoch 86/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1405 -
val_loss: 0.1132
Epoch 87/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1405 -
val_loss: 0.1132
Epoch 88/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1404 -
val_loss: 0.1130
Epoch 89/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1406
Epoch 00089: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
600/600 [==============================] - 20s 33ms/step - loss: 0.1406 -
val_loss: 0.1132
Epoch 90/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1403 -
val_loss: 0.1131
```

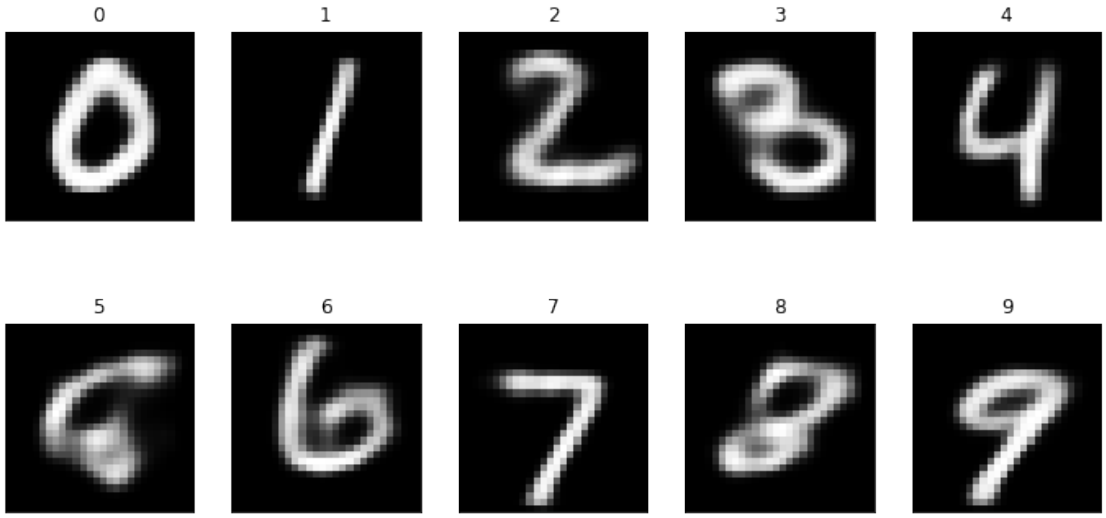
```
Epoch 91/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1404 -
val_loss: 0.1131
Epoch 92/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1403 -
val_loss: 0.1131
Epoch 93/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1403 -
val_loss: 0.1130
Epoch 94/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1404
Epoch 00094: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
600/600 [==============================] - 19s 32ms/step - loss: 0.1404 -
val_loss: 0.1131
Epoch 95/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1403 -
val_loss: 0.1131
Epoch 96/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1403 -
val_loss: 0.1132
Epoch 97/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1403 -
val_loss: 0.1131
Epoch 98/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1402 -
val_loss: 0.1131
Epoch 99/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1404
Epoch 00099: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
600/600 [==============================] - 20s 33ms/step - loss: 0.1404 -
val_loss: 0.1130
Epoch 100/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1402 -
val_loss: 0.1131
Epoch 101/1000
600/600 [==============================] - 19s 32ms/step - loss: 0.1402 -
val_loss: 0.1131
Epoch 102/1000
600/600 [==============================] - 20s 33ms/step - loss: 0.1402 -
val_loss: 0.1131
Epoch 103/1000
599/600 [============================>.] - ETA: 0s - loss: 0.1403Restoring model
weights from the end of the best epoch.
600/600 [==============================] - 20s 33ms/step - loss: 0.1403 -
val_loss: 0.1130
Epoch 00103: early stopping
```

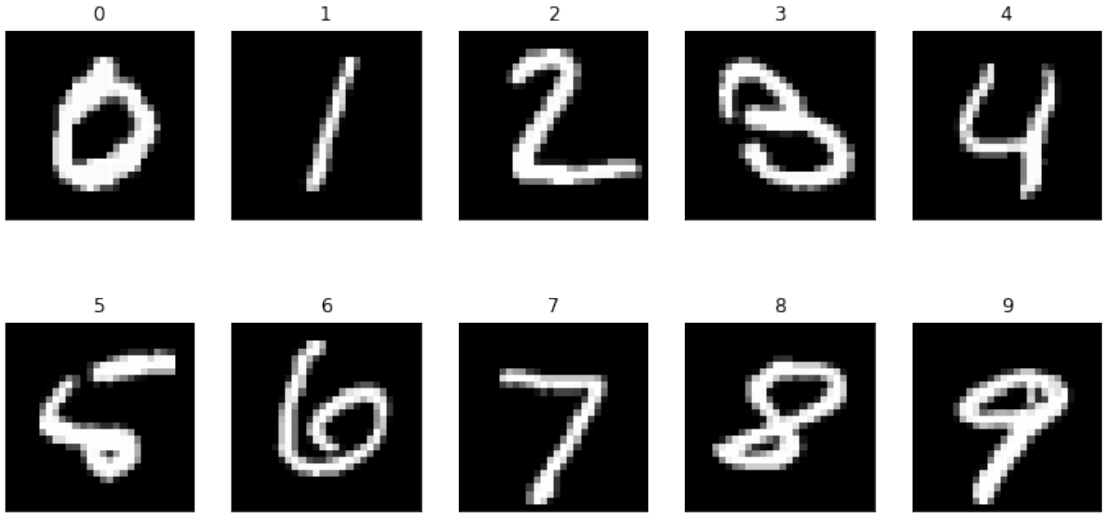
[17]:
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,6))
plt.plot(hist.history['loss'], color='#785ef0')
plt.plot(hist.history['val_loss'], color='#dc267f')
plt.title('Model Loss Progress')
plt.ylabel('Brinary Cross-Entropy Loss')
plt.xlabel('Epoch')
plt.legend(['Training Set', 'Test Set'], loc='upper right')
plt.savefig('ch.13.ae.bilstm.mnist.loss.png', dpi=350, bbox_inches='tight')
plt.show()
```



[18]:
```
import matplotlib.pyplot as plt
import numpy as np

encdd = encoder.predict(x_test)
x_hat = autoencoder.predict(x_test)

smp_idx = [3,2,1,18,4,8,11,0,61,9]
plt.figure(figsize=(12,6))
for i, (img, y) in enumerate(zip(x_hat[smp_idx].reshape(10, 28, 28),␣
 ↪y_test[smp_idx])):
  plt.subplot(2,5,i+1)
  plt.imshow(img, cmap='gray')
```

```
  plt.xticks([])
  plt.yticks([])
  plt.title(y)
plt.savefig('bilstm.mnist.reconstructed.png', bbox_inches='tight', dpi=350)
plt.show()

plt.figure(figsize=(12,6))
for i, (img, y) in enumerate(zip(x_test[smp_idx].reshape(10, 28, 28),␣
 ↪y_test[smp_idx])):
  plt.subplot(2,5,i+1)
  plt.imshow(img, cmap='gray')
  plt.xticks([])
  plt.yticks([])
  plt.title(y)
plt.savefig('bilstm.mnist.original.png', bbox_inches='tight', dpi=350)
plt.show()
```





[20]:
```
!pip install umap
```

```
Collecting umap
  Downloading umap-0.1.1.tar.gz (3.2 kB)
Building wheels for collected packages: umap
  Building wheel for umap (setup.py): started
  Building wheel for umap (setup.py): finished with status 'done'
  Created wheel for umap: filename=umap-0.1.1-py3-none-any.whl size=3565
sha256=b9360416a6acf68f8d8b568ae68c3530319e2ff91226f0f3858ab744cc03d43c
  Stored in directory: c:\users\nouhad\appdata\local\pip\cache\wheels\65\55\85\9
45cfb3d67373767e4dc3e9629300a926edde52633df4f0efe
Successfully built umap
Installing collected packages: umap
Successfully installed umap-0.1.1
```

[22]:
```
!pip install umap-learn
```

```
Collecting umap-learn
  Downloading umap-learn-0.5.1.tar.gz (80 kB)
Requirement already satisfied: numpy>=1.17 in
c:\users\nouhad\.conda\envs\cosc4337\lib\site-packages (from umap-learn)
(1.19.5)
Requirement already satisfied: scikit-learn>=0.22 in
c:\users\nouhad\.conda\envs\cosc4337\lib\site-packages (from umap-learn)
(0.24.2)
Requirement already satisfied: scipy>=1.0 in
c:\users\nouhad\.conda\envs\cosc4337\lib\site-packages (from umap-learn) (1.7.1)
Collecting numba>=0.49
  Downloading numba-0.54.1-cp37-cp37m-win_amd64.whl (2.3 MB)
Collecting pynndescent>=0.5
```

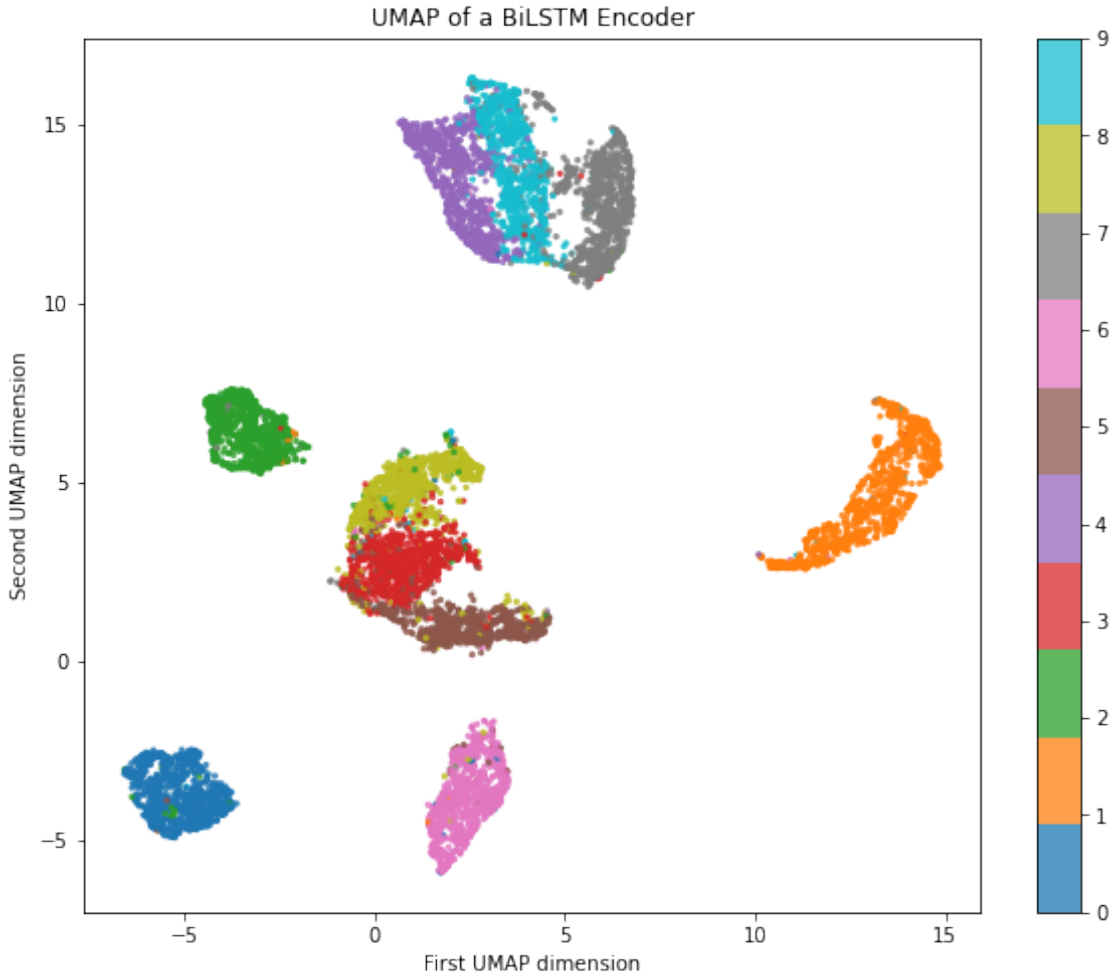
```
  Downloading pynndescent-0.5.4.tar.gz (1.1 MB)
Collecting llvmlite<0.38,>=0.37.0rc1
  Downloading llvmlite-0.37.0-cp37-cp37m-win_amd64.whl (17.0 MB)
Requirement already satisfied: setuptools in
c:\users\nouhad\.conda\envs\cosc4337\lib\site-packages (from numba>=0.49->umap-
learn) (52.0.0.post20210125)
Requirement already satisfied: joblib>=0.11 in
c:\users\nouhad\.conda\envs\cosc4337\lib\site-packages (from
pynndescent>=0.5->umap-learn) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\nouhad\.conda\envs\cosc4337\lib\site-packages (from scikit-
learn>=0.22->umap-learn) (2.2.0)
Building wheels for collected packages: umap-learn, pynndescent
  Building wheel for umap-learn (setup.py): started
  Building wheel for umap-learn (setup.py): finished with status 'done'
  Created wheel for umap-learn: filename=umap_learn-0.5.1-py3-none-any.whl
size=76564
sha256=cc79d16261f0b032112a8a255b9bf14a09ff7ba185ed19c33c488f5221586f7e
  Stored in directory: c:\users\nouhad\appdata\local\pip\cache\wheels\01\e7\bb\3
47dc0e510803d7116a13d592b10cc68262da56a8eec4dd72f
  Building wheel for pynndescent (setup.py): started
  Building wheel for pynndescent (setup.py): finished with status 'done'
  Created wheel for pynndescent: filename=pynndescent-0.5.4-py3-none-any.whl
size=52359
sha256=1fdc4cc3d59e4a1821fbbddd2e99713e800cf542f5f730c86eecf04b31fbcc08
  Stored in directory: c:\users\nouhad\appdata\local\pip\cache\wheels\d0\5b\62\3
401692ddad12324249c774c4b15ccb046946021e2b581c043
Successfully built umap-learn pynndescent
Installing collected packages: llvmlite, numba, pynndescent, umap-learn
Successfully installed llvmlite-0.37.0 numba-0.54.1 pynndescent-0.5.4 umap-
learn-0.5.1
```

[24]:
```
import umap.umap_ as umap
```

[25]:
```
import matplotlib.pyplot as plt
#import umap

y_ = list(map(int, y_test))
X_ = encdd

print(X_.shape)

X_ = umap.UMAP().fit_transform(encdd)
print(X_.shape)


plt.figure(figsize=(10,8))
```

```
plt.title('UMAP of a BiLSTM Encoder')
plt.scatter(X_[:,0], X_[:,1], s=5.0, c=y_, alpha=0.75, cmap='tab10')
plt.xlabel('First UMAP dimension')
plt.ylabel('Second UMAP dimension')
plt.colorbar()
plt.savefig('bilstm.umap.png', bbox_inches='tight', dpi=350)
```

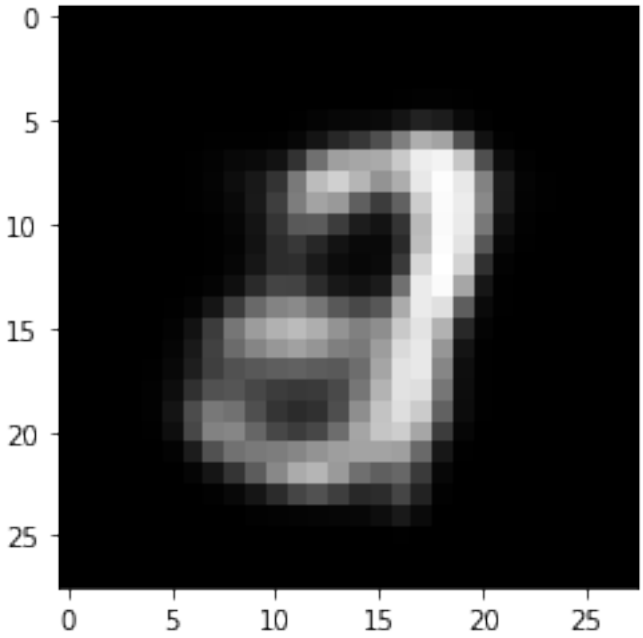
```
(10000, 100)
(10000, 2)
```



[26]:
```
z = np.random.rand(1,100)
x_ = decoder.predict(z)
print(x_.shape)
plt.imshow(x_[0], cmap='gray')
```

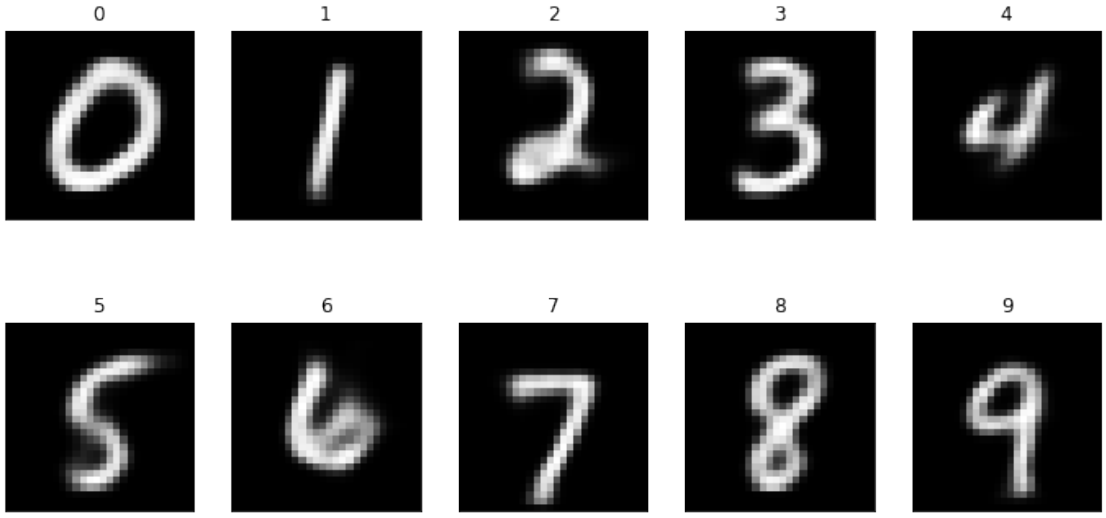
```
(1, 28, 28)
```

[26]: <matplotlib.image.AxesImage at 0x1f85363c748>



##Sequence to Sequence Models

[27]:
```
plt.figure(figsize=(12,6))
for i in range(10):
  plt.subplot(2,5,i+1)
  rnd_vec = np.round(np.mean(x_test[y_test==i],axis=0))
  rnd_vec = np.reshape(rnd_vec, (1,28,28))
  z = encoder.predict(rnd_vec)
  decdd = decoder.predict(z)
  plt.imshow(decdd[0], cmap='gray')
  plt.xticks([])
  plt.yticks([])
  plt.title(i)
plt.savefig('bilstm.mnist.v2s.png', bbox_inches='tight', dpi=350)
plt.show()
```


0
1
2
3
4
5
6
7
8
9

[ ]: