Assign 6_final

December 6, 2017

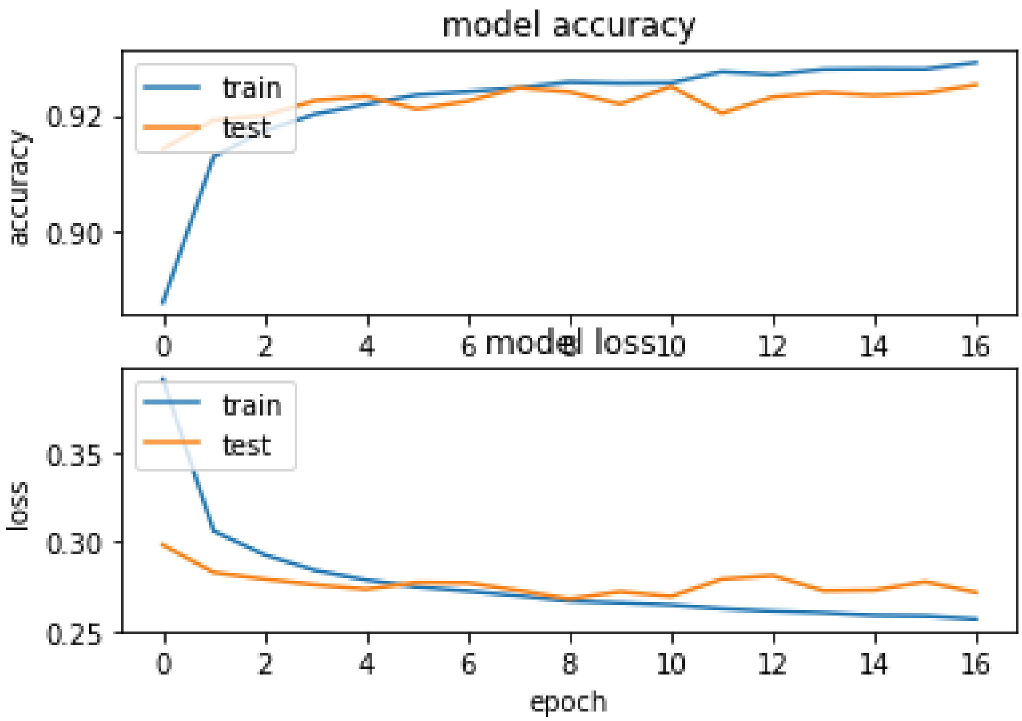
```
In [50]: from tensorflow.examples.tutorials.mnist import input_data

         import tensorflow as tf

         from keras.models import Sequential
         from keras.layers import Dense
         from keras import initializers
         import numpy as np
         from numpy import newaxis
         import matplotlib.pyplot as plt
         import matplotlib.mlab as math
         import keras as keras

         import pandas as pd
```

```
In [61]: # 1A
         from keras.optimizers import SGD
         from keras.initializers import Zeros
         from keras.datasets import mnist
         (train_images, train_labels), (test_images, test_labels) = mnist.load_data()

         ############################
         print('Training data shape : ', train_images.shape, train_labels.shape)

         print('Testing data shape : ', test_images.shape, test_labels.shape)

         # Find the unique numbers from the train labels
         classes = np.unique(train_labels)
         nClasses = len(classes)
         print('Total number of outputs : ', nClasses)
         print('Output classes : ', classes)

         # Change from matrix to array of dimension 28x28 to array of dimention 784
         dimData = np.prod(train_images.shape[1:])
         train_data = train_images.reshape(train_images.shape[0], dimData)
         test_data = test_images.reshape(test_images.shape[0], dimData)
```

```python
# Change to float datatype
train_data = train_data.astype('float32')
test_data = test_data.astype('float32')

# Scale the data to lie between 0 to 1
train_data /= 255
test_data /= 255

# Change the labels from integer to categorical data
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

model = Sequential()
model.add(Dense(10, activation='softmax',use_bias=True, kernel_initializer=Zeros(),
                bias_initializer=Zeros() ,input_shape=(784,)))
model.summary()
model.compile(loss='categorical_crossentropy',
 optimizer=SGD(0.5),
 metrics=['accuracy'])

history = model.fit(train_data, train_labels_one_hot, batch_size=100, epochs=17, verbo
                   validation_data=(test_data, test_labels_one_hot),)

plt.gcf().clear()
plt.figure(1)

# summarize history for accuracy

plt.subplot(211)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')

# summarize history for loss

plt.subplot(212)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
Training data shape :  (60000, 28, 28) (60000,)
```

```
Testing data shape :  (10000, 28, 28) (10000,)
Total number of outputs :  10
Output classes :  [0 1 2 3 4 5 6 7 8 9]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_105 (Dense)            (None, 10)                7850
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
```

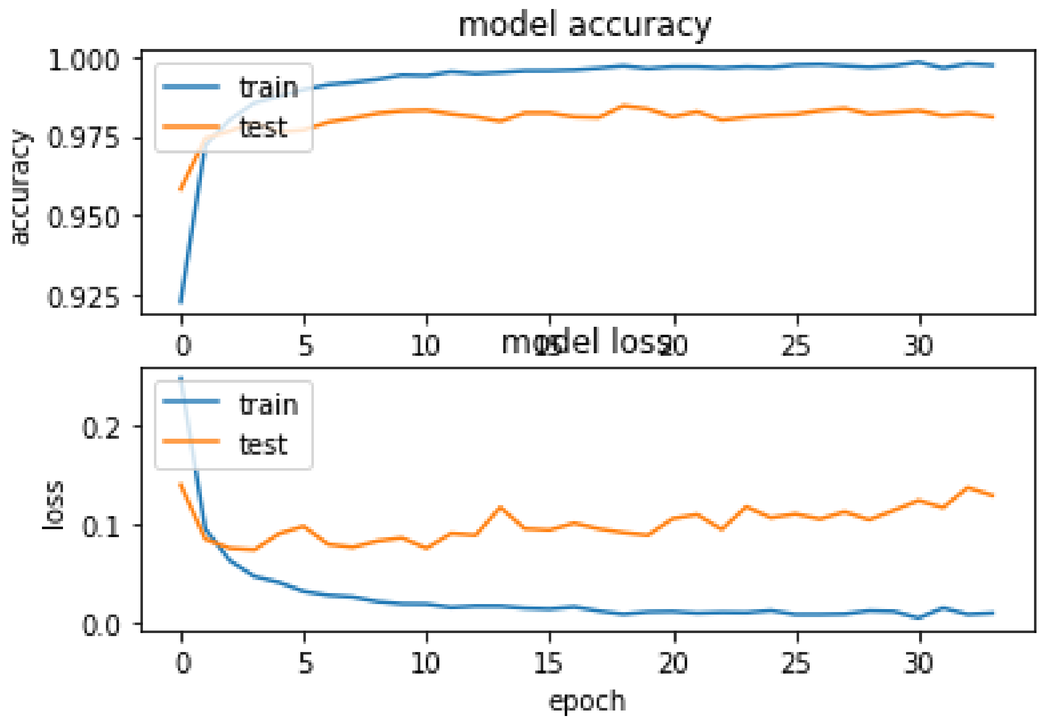



```
In [52]: # 1B
         # now in Keras (since it's easier to use
         # https://www.learnopencv.com/image-classification-using-feedforward-neural-network-i

         from keras.datasets import mnist
         (train_images, train_labels), (test_images, test_labels) = mnist.load_data()

         from keras.utils import to_categorical

         # Find the unique numbers from the train labels
```

```
classes = np.unique(train_labels)
nClasses = len(classes)


# Change from matrix to array of dimension 28x28 to array of dimention 784
dimData = np.prod(train_images.shape[1:])
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

# Change to float datatype
train_data = train_data.astype('float32')
test_data = test_data.astype('float32')

# Scale the data to lie between 0 to 1
train_data /= 255
test_data /= 255

# Change the labels from integer to categorical data
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)


#-----------------------------------------------------------------------------------
#weights
weight1 = TruncatedNormal(mean=0.0, stddev=0.01)
weight1((784,1500))
weight2 = TruncatedNormal(mean=0.0, stddev=0.01)
weight2((1500,1500))
weight3 = TruncatedNormal(mean=0.0, stddev=0.01)
weight3((1500, 1500))
weight4 = TruncatedNormal(mean=0.0, stddev=0.01)
weight4((1500, 10))

#Network Creation
model = Sequential()
model.add(Dense(1500, activation='relu',use_bias=True, input_shape=(dimData,), kernel_
model.add(Dense(1500, activation='relu',use_bias=True, kernel_initializer=weight2, bia
model.add(Dense(1500, activation='relu',use_bias=True, kernel_initializer = weight3, b
model.add(Dense(nClasses, activation='softmax',use_bias=True, kernel_initializer=weigh
model.summary()

#implementing adam optimizer
adam = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1*10**(-8), d

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

history = model.fit(train_data, train_labels_one_hot, batch_size=100, epochs=34, verbo
                   validation_data=(test_data, test_labels_one_hot))
```

```python
[test_loss, test_acc] = model.evaluate(test_data, test_labels_one_hot)

print(history.history.keys())
plt.gcf().clear()
plt.figure(1)

# summarize history for accuracy
plt.subplot(211)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')

# summarize history for loss

plt.subplot(212)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_88 (Dense)             (None, 1500)              1177500
_________________________________________________________________
dense_89 (Dense)             (None, 1500)              2251500
_________________________________________________________________
dense_90 (Dense)             (None, 1500)              2251500
_________________________________________________________________
dense_91 (Dense)             (None, 10)                15010
=================================================================
Total params: 5,695,510
Trainable params: 5,695,510
Non-trainable params: 0
_________________________________________________________________
10000/10000 [==============================] - 3s 261us/step
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```




```
In [55]: # 1C
         # Dropout is the only real addition
         # Our model will contain the dropout() statement

         from keras.layers import Dropout
         from keras.datasets import mnist


         (train_images, train_labels), (test_images, test_labels) = mnist.load_data()

         from keras.utils import to_categorical

         print('Training data shape : ', train_images.shape, train_labels.shape)

         print('Testing data shape : ', test_images.shape, test_labels.shape)

         # Find the unique numbers from the train labels
         classes = np.unique(train_labels)
         nClasses = len(classes)
         print('Total number of outputs : ', nClasses)
         print('Output classes : ', classes)

         # Change from matrix to array of dimension 28x28 to array of dimention 784
         dimData = np.prod(train_images.shape[1:])
```

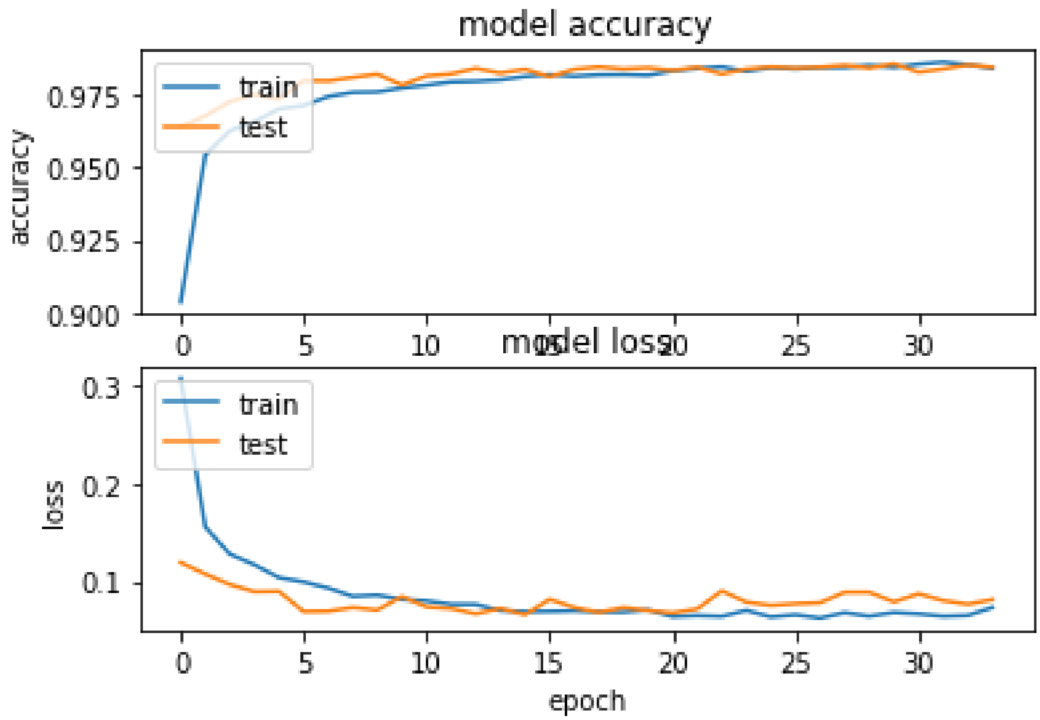
```
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

# Change to float datatype
train_data = train_data.astype('float32')
test_data = test_data.astype('float32')


# Scale the data to lie between 0 to 1
train_data /= 255
test_data /= 255

# Change the labels from integer to categorical data
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

# Display the change for category label using one-hot encoding
print('Original label 0 : ', train_labels[0])
print('After conversion to categorical ( one-hot ) : ', train_labels_one_hot[0])

#-----------------------------------------------------------------------------------
weight1 = TruncatedNormal(mean=0.0, stddev=0.01)
weight1((784,1500))
weight2 = TruncatedNormal(mean=0.0, stddev=0.01)
weight2((1500,1500))
weight3 = TruncatedNormal(mean=0.0, stddev=0.01)
weight3((1500, 1500))
weight4 = TruncatedNormal(mean=0.0, stddev=0.01)
weight4((1500, 10))

model = Sequential()
model.add(Dense(1500, activation='relu',use_bias=True, input_shape=(dimData,), kernel_
model.add(Dropout(0.5))
model.add(Dense(1500, activation='relu',use_bias=True, kernel_initializer=weight2, bia
model.add(Dropout(0.5))
model.add(Dense(1500, activation='relu',use_bias=True, kernel_initializer = weight3, b
model.add(Dropout(0.5))
model.add(Dense(nClasses, activation='softmax',use_bias=True, kernel_initializer=weigh

model.summary()

#implementing adam optimizer
adam = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1*10**(-8), d

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


history = model.fit(train_data, train_labels_one_hot, batch_size=100, epochs=34, verbo
```

```
                        validation_data=(test_data, test_labels_one_hot),)

[test_loss, test_acc] = model.evaluate(test_data, test_labels_one_hot)
#print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss,
plt.gcf().clear()
plt.figure(1)

# summarize history for accuracy

plt.subplot(211)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')

# summarize history for loss

plt.subplot(212)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
Training data shape :  (60000, 28, 28) (60000,)
Testing data shape :  (10000, 28, 28) (10000,)
Total number of outputs :  10
Output classes :  [0 1 2 3 4 5 6 7 8 9]
Original label 0 :  5
After conversion to categorical ( one-hot ) :  [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_96 (Dense)             (None, 1500)              1177500
_________________________________________________________________
dropout_46 (Dropout)         (None, 1500)              0
_________________________________________________________________
dense_97 (Dense)             (None, 1500)              2251500
_________________________________________________________________
dropout_47 (Dropout)         (None, 1500)              0
_________________________________________________________________
dense_98 (Dense)             (None, 1500)              2251500
_________________________________________________________________
```

```
dropout_48 (Dropout)         (None, 1500)             0
_________________________________________________________________
dense_99 (Dense)             (None, 10)               15010
=================================================================
Total params: 5,695,510
Trainable params: 5,695,510
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/34
60000/60000 [==============================] - 62s 1ms/step - loss: 0.3061 - acc: 0.9038 - val_
Epoch 2/34
60000/60000 [==============================] - 59s 990us/step - loss: 0.1556 - acc: 0.9543 - va
Epoch 3/34
60000/60000 [==============================] - 60s 1ms/step - loss: 0.1280 - acc: 0.9627 - val_
Epoch 4/34
60000/60000 [==============================] - 59s 985us/step - loss: 0.1172 - acc: 0.9658 - va
Epoch 5/34
60000/60000 [==============================] - 59s 985us/step - loss: 0.1041 - acc: 0.9702 - va
Epoch 6/34
60000/60000 [==============================] - 59s 986us/step - loss: 0.0997 - acc: 0.9714 - va
Epoch 7/34
60000/60000 [==============================] - 59s 984us/step - loss: 0.0935 - acc: 0.9745 - va
Epoch 8/34
60000/60000 [==============================] - 59s 989us/step - loss: 0.0851 - acc: 0.9759 - va
Epoch 9/34
60000/60000 [==============================] - 59s 982us/step - loss: 0.0863 - acc: 0.9761 - va
Epoch 10/34
60000/60000 [==============================] - 59s 981us/step - loss: 0.0820 - acc: 0.9774 - va
Epoch 11/34
60000/60000 [==============================] - 59s 980us/step - loss: 0.0805 - acc: 0.9783 - va
Epoch 12/34
60000/60000 [==============================] - 59s 979us/step - loss: 0.0770 - acc: 0.9795 - va
Epoch 13/34
60000/60000 [==============================] - 59s 980us/step - loss: 0.0769 - acc: 0.9797 - va
Epoch 14/34
60000/60000 [==============================] - 59s 979us/step - loss: 0.0707 - acc: 0.9802 - va
Epoch 15/34
60000/60000 [==============================] - 59s 978us/step - loss: 0.0695 - acc: 0.9813 - va
Epoch 16/34
60000/60000 [==============================] - 59s 978us/step - loss: 0.0698 - acc: 0.9820 - va
Epoch 17/34
60000/60000 [==============================] - 59s 977us/step - loss: 0.0704 - acc: 0.9814 - va
Epoch 18/34
60000/60000 [==============================] - 59s 983us/step - loss: 0.0694 - acc: 0.9819 - va
Epoch 19/34
60000/60000 [==============================] - 60s 994us/step - loss: 0.0691 - acc: 0.9820 - va
Epoch 20/34
```

```
60000/60000 [==============================] - 59s 988us/step - loss: 0.0714 - acc: 0.9817 - va
Epoch 21/34
60000/60000 [==============================] - 60s 1000us/step - loss: 0.0646 - acc: 0.9833 - v
Epoch 22/34
60000/60000 [==============================] - 60s 998us/step - loss: 0.0656 - acc: 0.9841 - va
Epoch 23/34
60000/60000 [==============================] - 60s 992us/step - loss: 0.0648 - acc: 0.9846 - va
Epoch 24/34
60000/60000 [==============================] - 60s 997us/step - loss: 0.0708 - acc: 0.9831 - va
Epoch 25/34
60000/60000 [==============================] - 60s 998us/step - loss: 0.0642 - acc: 0.9844 - va
Epoch 26/34
60000/60000 [==============================] - 60s 999us/step - loss: 0.0664 - acc: 0.9839 - va
Epoch 27/34
60000/60000 [==============================] - 60s 994us/step - loss: 0.0630 - acc: 0.9844 - va
Epoch 28/34
60000/60000 [==============================] - 60s 996us/step - loss: 0.0685 - acc: 0.9844 - va
Epoch 29/34
60000/60000 [==============================] - 60s 997us/step - loss: 0.0649 - acc: 0.9852 - va
Epoch 30/34
60000/60000 [==============================] - 60s 1ms/step - loss: 0.0687 - acc: 0.9844 - val_
Epoch 31/34
60000/60000 [==============================] - 60s 1ms/step - loss: 0.0673 - acc: 0.9856 - val_
Epoch 32/34
60000/60000 [==============================] - 60s 1ms/step - loss: 0.0649 - acc: 0.9863 - val_
Epoch 33/34
60000/60000 [==============================] - 60s 999us/step - loss: 0.0657 - acc: 0.9854 - va
Epoch 34/34
60000/60000 [==============================] - 60s 999us/step - loss: 0.0737 - acc: 0.9842 - va
10000/10000 [==============================] - 3s 262us/step
```

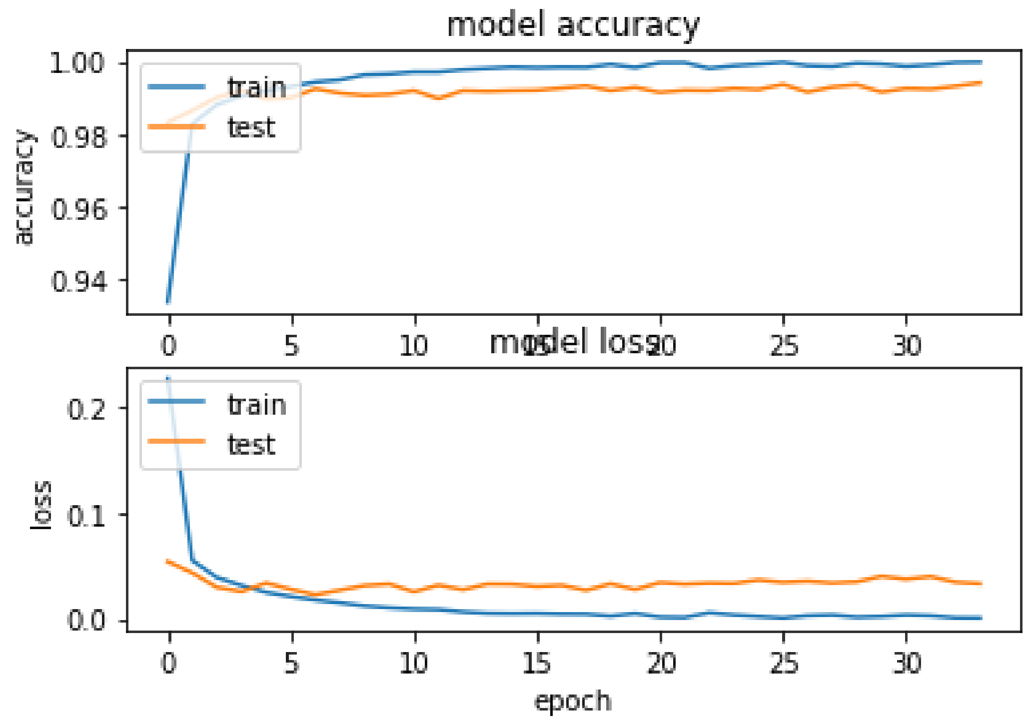


In [58]:
```
# 1D

from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(1)
import keras
import matplotlib.pyplot as plt
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.initializers import TruncatedNormal,Constant,Zeros
from keras.layers import Dense,Dropout,Conv2D,MaxPooling2D,Flatten
from keras.optimizers import SGD,Adam
# setting up of batch, and the number of classes and epochs

batch_size = 100
num_classes = 10
epochs = 34

# input image dimensions
img_x, img_y = 28, 28

# load the MNIST data set, which already splits into train and test sets for us
```

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

# reshape the data into a 4D tensor - (sample_number, x_img_size, y_img_size, num_cha
# because the MNIST is greyscale, we only have a single channel - RGB colour images w
x_train = x_train.reshape(x_train.shape[0], img_x, img_y, 1)
x_test = x_test.reshape(x_test.shape[0], img_x, img_y, 1)
input_shape = (img_x, img_y, 1)

# convert the data to the right type
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices - this is for use in the
# categorical_crossentropy loss below
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

adam = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1*10**(-8), d

model = Sequential()
model.add(Conv2D(32,(5,5),strides=(1, 1),padding='same', activation='relu',use_bias=Tr
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2,2)))
model.add(Conv2D(64,(5,5),strides=(1, 1),padding='same', activation='relu',use_bias=Tr
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2,2)))
model.add(Flatten())
model.add(Dense(num_classes, activation='softmax',use_bias=True, bias_initializer=Cons

model.summary()

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=adam,metrics=['accu


history = model.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=0,val

#[test_loss, test_acc] = model.evaluate(test_data, test_labels_one_hot)
#print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss,
plt.gcf().clear()
plt.figure(1)

 # summarize history for accuracy

plt.subplot(211)
plt.plot(history.history['acc'])
```

```python
        plt.plot(history.history['val_acc'])
        plt.title('model accuracy')
        plt.ylabel('accuracy')
        plt.xlabel('epoch')
        plt.legend(['train', 'test'], loc='upper left')

        # summarize history for loss

        plt.subplot(212)
        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        plt.title('model loss')
        plt.ylabel('loss')
        plt.xlabel('epoch')
        plt.legend(['train', 'test'], loc='upper left')
        plt.show()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_27 (Conv2D)           (None, 28, 28, 32)        832
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 14, 14, 64)        51264
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
flatten_14 (Flatten)         (None, 3136)              0
_________________________________________________________________
dense_102 (Dense)            (None, 10)                31370
=================================================================
Total params: 83,466
Trainable params: 83,466
Non-trainable params: 0
_________________________________________________________________
```



In [ ]: #1e

Plots of the subtasks were shown below of every task.

In [62]:
```
from keras.layers import LSTM

#A
train_size = 8000
# np.random.seed(7)
dataset = np.random.randint(low=0, high=9+1, size=(10000,30))

def get_labels(dataset):
    sum = np.sum(dataset,1)
    sum[sum<100]=0
    sum[sum>=100]=1
    return sum


train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
train_labels = np.reshape(get_labels(train),(-1,1))
test_labels = np.reshape(get_labels(test),(-1,1))
train = train[:,:,newaxis]
test = test[:,:,newaxis]
# print(train.shape)
```

```python
# print(train.ndim)
# print(len(train), len(test))
#B
timesteps=30
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(200, input_shape=(train.shape[1:])))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())

#D change epchos to 60
model.fit(train, train_labels, validation_data=(test, test_labels), epochs=60, batch_s

#E
# Final evaluation of the model
scores = model.evaluate(test, test_labels, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 200)               161600
_________________________________________________________________
dense_106 (Dense)            (None, 1)                 201
=================================================================
Total params: 161,801
Trainable params: 161,801
Non-trainable params: 0
_________________________________________________________________
None
Train on 8000 samples, validate on 2000 samples
Epoch 1/60
8000/8000 [==============================] - 10s 1ms/step - loss: 0.0655 - acc: 0.9856 - val_lo
Epoch 2/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0478 - acc: 0.9866 - val_l
Epoch 3/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0379 - acc: 0.9874 - val_los
Epoch 4/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0343 - acc: 0.9888 - val_los
Epoch 5/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0261 - acc: 0.9914 - val_los
Epoch 6/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0249 - acc: 0.9908 - val_los
Epoch 7/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0190 - acc: 0.9924 - val_los
Epoch 8/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0197 - acc: 0.9934 - val_los
```

```
Epoch 9/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0181 - acc: 0.9931 - val_los
Epoch 10/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0204 - acc: 0.9929 - val_los
Epoch 11/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0163 - acc: 0.9936 - val_los
Epoch 12/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0122 - acc: 0.9959 - val_los
Epoch 13/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0142 - acc: 0.9946 - val_los
Epoch 14/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0127 - acc: 0.9948 - val_los
Epoch 15/60
8000/8000 [==============================] - 8s 994us/step - loss: 0.0156 - acc: 0.9928 - val_l
Epoch 16/60
8000/8000 [==============================] - 8s 995us/step - loss: 0.0130 - acc: 0.9954 - val_l
Epoch 17/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0111 - acc: 0.9960 - val_l
Epoch 18/60
8000/8000 [==============================] - 8s 982us/step - loss: 0.0120 - acc: 0.9958 - val_l
Epoch 19/60
8000/8000 [==============================] - 8s 989us/step - loss: 0.0115 - acc: 0.9954 - val_l
Epoch 20/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0118 - acc: 0.9953 - val_l
Epoch 21/60
8000/8000 [==============================] - 8s 994us/step - loss: 0.0113 - acc: 0.9950 - val_l
Epoch 22/60
8000/8000 [==============================] - 8s 998us/step - loss: 0.0101 - acc: 0.9960 - val_l
Epoch 23/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0110 - acc: 0.9956 - val_los
Epoch 24/60
8000/8000 [==============================] - 8s 989us/step - loss: 0.0122 - acc: 0.9953 - val_l
Epoch 25/60
8000/8000 [==============================] - 8s 996us/step - loss: 0.0084 - acc: 0.9973 - val_l
Epoch 26/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0123 - acc: 0.9951 - val_l
Epoch 27/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0092 - acc: 0.9965 - val_l
Epoch 28/60
8000/8000 [==============================] - 8s 994us/step - loss: 0.0104 - acc: 0.9955 - val_l
Epoch 29/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0111 - acc: 0.9964 - val_l
Epoch 30/60
8000/8000 [==============================] - 8s 987us/step - loss: 0.0096 - acc: 0.9956 - val_l
Epoch 31/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0071 - acc: 0.9975 - val_los
Epoch 32/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0083 - acc: 0.9968 - val_los
```

```
Epoch 33/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0101 - acc: 0.9964 - val_los
Epoch 34/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0070 - acc: 0.9976 - val_los
Epoch 35/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0065 - acc: 0.9975 - val_los
Epoch 36/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0105 - acc: 0.9959 - val_los
Epoch 37/60
8000/8000 [==============================] - 8s 992us/step - loss: 0.0066 - acc: 0.9971 - val_l
Epoch 38/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0083 - acc: 0.9974 - val_los
Epoch 39/60
8000/8000 [==============================] - 8s 992us/step - loss: 0.0080 - acc: 0.9965 - val_l
Epoch 40/60
8000/8000 [==============================] - 8s 993us/step - loss: 0.0087 - acc: 0.9966 - val_l
Epoch 41/60
8000/8000 [==============================] - 8s 993us/step - loss: 0.0061 - acc: 0.9980 - val_l
Epoch 42/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0062 - acc: 0.9980 - val_los
Epoch 43/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0069 - acc: 0.9970 - val_los
Epoch 44/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0085 - acc: 0.9973 - val_los
Epoch 45/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0060 - acc: 0.9976 - val_los
Epoch 46/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0090 - acc: 0.9965 - val_los
Epoch 47/60
8000/8000 [==============================] - 8s 998us/step - loss: 0.0056 - acc: 0.9981 - val_l
Epoch 48/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0064 - acc: 0.9979 - val_l
Epoch 49/60
8000/8000 [==============================] - 8s 993us/step - loss: 0.0065 - acc: 0.9971 - val_l
Epoch 50/60
8000/8000 [==============================] - 8s 995us/step - loss: 0.0047 - acc: 0.9981 - val_l
Epoch 51/60
8000/8000 [==============================] - 8s 995us/step - loss: 0.0061 - acc: 0.9975 - val_l
Epoch 52/60
8000/8000 [==============================] - 8s 996us/step - loss: 0.0100 - acc: 0.9968 - val_l
Epoch 53/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0051 - acc: 0.9981 - val_los
Epoch 54/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0050 - acc: 0.9981 - val_los
Epoch 55/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0059 - acc: 0.9975 - val_los
Epoch 56/60
8000/8000 [==============================] - 9s 1ms/step - loss: 0.0061 - acc: 0.9975 - val_los
```

```
Epoch 57/60
8000/8000 [==============================] - 8s 1ms/step - loss: 0.0040 - acc: 0.9988 - val_los
Epoch 58/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0035 - acc: 0.9991 - val_l
Epoch 59/60
8000/8000 [==============================] - 8s 994us/step - loss: 0.0047 - acc: 0.9983 - val_l
Epoch 60/60
8000/8000 [==============================] - 8s 991us/step - loss: 0.0023 - acc: 0.9993 - val_l
Accuracy: 99.65%
```