Open in Colab
(https://colab.research.google.com/github/oliverfoster27/Practical-Machine-Learning/blob/master/Week%207/C7_Exercises.ipynb)

In [0]:
```
from keras.layers import Dropout


import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt

from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

from keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout, BatchNorm
alization
from keras.utils import to_categorical
import keras.backend as K
from keras.callbacks import EarlyStopping
from tensorflow.keras.datasets import mnist
from keras.optimizers import Adam
```

In [8]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data('/tmp/mnist.npz')
y_train_cat = to_categorical(y_train)
y_test_cat = to_categorical(y_test)

X_train = X_train.reshape(-1, 28*28)
X_test = X_test.reshape(-1, 28*28)
X_train.shape, X_test.shape, y_train, y_test
```

Out[8]:
```
((60000, 784),
 (10000, 784),
 array([5, 0, 4, ..., 5, 6, 8], dtype=uint8),
 array([7, 2, 1, ..., 4, 5, 6], dtype=uint8))
```

In [0]:
```
X_train = X_train / 255
X_test = X_test / 255
```

Exercise

Make a dense neural network with 95%+ accuracy on Mnist that has the smallest number of neurons possible by experimenting with Dropout, and Batch Norm

Strategy:

- Decide on a basic layer architecutre (one hidden dense layer with dropout and batch normalization)
- Iterate starting with 10 neurons to 100 on the hidden layer and find the optimal learning rate & p value for dropout
- If early stopping happens during training it indicates that the network may not be powerful enough. At this point increase the size of the dense layer
- When validation accuracy reaches 95% return that network's architecture

In [0]:
```
from keras.models import Sequential

# Grids to iterate through
learning_grid = [10e-6, 10e-5, 10e-4]
input_dense_grid = np.linspace(10, 100, 19).astype(int)
p_grid = [0.2, 0.4, 0.6]

def find_minimal_network(features, output, val_thresh=0.95):

  data = []

  for input_dense in input_dense_grid:

    stopped_early = False

    for learning_rate in learning_grid:

      if stopped_early:
        break

      for p in p_grid:

        K.clear_session()

        model = Sequential()

        # Baseline architecture
        model.add(Dense(input_dense, activation='relu', input_shape=(784,)))
        model.add(Dropout(p))
        model.add(BatchNormalization())
        model.add(Dense(10, activation='softmax'))

        callback_list = [EarlyStopping(monitor='val_acc', mode='max',
                                       verbose=0, patience=5)]

        optimizer = Adam(lr=learning_rate)
        model.compile(optimizer=optimizer,
                      loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])

        print("\nNumber of Dense Nodes: {}".format(input_dense))
        print("Learning Rate: {}".format(learning_rate))
        print("Dropout P: {}".format(p))

        h = model.fit(features, output, epochs=100, validation_split=0.3,
                      callbacks=callback_list, verbose=0)

        # If we stop early our learning rate doesn't need to increase
        # so once we're done iterating through p increase the layer size
        stopping_interval = callback_list[0].stopped_epoch
        if stopping_interval > 0:
          stopped_early = True

        print("Early Stopping: {}".format(stopping_interval))
        print("Trained Validation Accuracy: {}".format(h.history['val_acc'][-1
]))
```

```
        if h.history['val_acc'][-1] >= val_thresh:
          res={
            'Input Dense Layer Size': input_dense,
            'Learning Rate': learning_rate,
            'Dropout Proportion': p,
            'Validation Accuracy': h.history['acc'][-1]
          }
          # If we've satisfied 95% accuracy leave the learning rate grid
          return res
```

In [5]:
```
optimal_network = find_minimal_network(X_train, y_train)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/pyt
hon/framework/op_def_library.py:263: colocate_with (from tensorflow.python.fr
amework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/
tensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_op
s) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - k
eep_prob`.

Number of Dense Nodes: 10
Learning Rate: 1e-05
Dropout P: 0.2
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/pyt
hon/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is d
eprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Early Stopping: 0
Trained Validation Accuracy: 0.8922777777777777

Number of Dense Nodes: 10
Learning Rate: 1e-05
Dropout P: 0.4
Early Stopping: 0
Trained Validation Accuracy: 0.8823888888888889

Number of Dense Nodes: 10
Learning Rate: 1e-05
Dropout P: 0.6
Early Stopping: 0
Trained Validation Accuracy: 0.8424444444444444

Number of Dense Nodes: 10
Learning Rate: 0.0001
Dropout P: 0.2
Early Stopping: 30
Trained Validation Accuracy: 0.9103888888888889

Number of Dense Nodes: 10
Learning Rate: 0.0001
Dropout P: 0.4
Early Stopping: 30
Trained Validation Accuracy: 0.8892222222222222

Number of Dense Nodes: 10
Learning Rate: 0.0001
Dropout P: 0.6
Early Stopping: 31
Trained Validation Accuracy: 0.8757777777777778

Number of Dense Nodes: 15
Learning Rate: 1e-05
Dropout P: 0.2
Early Stopping: 78
```

```
Trained Validation Accuracy: 0.9126111111111112

Number of Dense Nodes: 15
Learning Rate: 1e-05
Dropout P: 0.4
Early Stopping: 0
Trained Validation Accuracy: 0.9080555555555555

Number of Dense Nodes: 15
Learning Rate: 1e-05
Dropout P: 0.6
Early Stopping: 0
Trained Validation Accuracy: 0.8847222222222222

Number of Dense Nodes: 20
Learning Rate: 1e-05
Dropout P: 0.2
Early Stopping: 88
Trained Validation Accuracy: 0.9285555555555556

Number of Dense Nodes: 20
Learning Rate: 1e-05
Dropout P: 0.4
Early Stopping: 0
Trained Validation Accuracy: 0.9137222222222222

Number of Dense Nodes: 20
Learning Rate: 1e-05
Dropout P: 0.6
Early Stopping: 0
Trained Validation Accuracy: 0.8990555555555556

Number of Dense Nodes: 25
Learning Rate: 1e-05
Dropout P: 0.2
Early Stopping: 0
Trained Validation Accuracy: 0.9355555555555556

Number of Dense Nodes: 25
Learning Rate: 1e-05
Dropout P: 0.4
Early Stopping: 0
Trained Validation Accuracy: 0.9193888888888889

Number of Dense Nodes: 25
Learning Rate: 1e-05
Dropout P: 0.6
Early Stopping: 0
Trained Validation Accuracy: 0.9048888888888889

Number of Dense Nodes: 25
Learning Rate: 0.0001
Dropout P: 0.2
Early Stopping: 59
Trained Validation Accuracy: 0.9515
```

Train network on minimum architecture

In [25]:
```
from keras.models import Sequential

input_dense = optimal_network['Input Dense Layer Size']
learning_rate = optimal_network['Learning Rate']
p = optimal_network['Dropout Proportion']

print("\nNumber of Dense Nodes: {}".format(input_dense))
print("Learning Rate: {}".format(learning_rate))
print("Dropout P: {}\n".format(p))

K.clear_session()

model = Sequential()

model.add(Dense(input_dense, activation='relu', input_shape=(784,)))
model.add(Dropout(p))
model.add(BatchNormalization())
model.add(Dense(10, activation='softmax'))

callback_list = [EarlyStopping(monitor='val_acc', mode='max',
                               verbose=0, patience=5)]

optimizer = Adam(lr=learning_rate)
model.compile(optimizer=optimizer,
             loss='sparse_categorical_crossentropy',
             metrics=['accuracy'])

model.summary()

h = model.fit(X_train, y_train, epochs=100, validation_split=0.3,
             callbacks=callback_list, verbose=1)
```

```
Number of Dense Nodes: 25
Learning Rate: 0.0001
Dropout P: 0.2

_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 25)                19625     
_________________________________________________________________
dropout_1 (Dropout)          (None, 25)                0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 25)                100       
_________________________________________________________________
dense_2 (Dense)              (None, 10)                260       
=================================================================
Total params: 19,985
Trainable params: 19,935
Non-trainable params: 50
_________________________________________________________________
Train on 42000 samples, validate on 18000 samples
Epoch 1/100
42000/42000 [==============================] - 7s 161us/step - loss: 1.3284 - 
acc: 0.5979 - val_loss: 0.7249 - val_acc: 0.8461
Epoch 2/100
42000/42000 [==============================] - 7s 161us/step - loss: 0.7479 - 
acc: 0.8036 - val_loss: 0.4941 - val_acc: 0.8882
Epoch 3/100
42000/42000 [==============================] - 7s 155us/step - loss: 0.5929 - 
acc: 0.8377 - val_loss: 0.4051 - val_acc: 0.9016
Epoch 4/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.5049 - 
acc: 0.8593 - val_loss: 0.3496 - val_acc: 0.9085
Epoch 5/100
42000/42000 [==============================] - 6s 152us/step - loss: 0.4636 - 
acc: 0.8674 - val_loss: 0.3149 - val_acc: 0.9147
Epoch 6/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.4289 - 
acc: 0.8758 - val_loss: 0.2948 - val_acc: 0.9199
Epoch 7/100
42000/42000 [==============================] - 6s 148us/step - loss: 0.4036 - 
acc: 0.8849 - val_loss: 0.2824 - val_acc: 0.9231
Epoch 8/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.3875 - 
acc: 0.8862 - val_loss: 0.2693 - val_acc: 0.9252
Epoch 9/100
42000/42000 [==============================] - 7s 163us/step - loss: 0.3661 - 
acc: 0.8935 - val_loss: 0.2606 - val_acc: 0.9284
Epoch 10/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.3585 - 
acc: 0.8940 - val_loss: 0.2524 - val_acc: 0.9291
Epoch 11/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.3498 - 
acc: 0.8963 - val_loss: 0.2446 - val_acc: 0.9332
Epoch 12/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.3389 - 
acc: 0.8998 - val_loss: 0.2383 - val_acc: 0.9342
Epoch 13/100
```

```
42000/42000 [==============================] - 6s 150us/step - loss: 0.3332 -
acc: 0.9011 - val_loss: 0.2298 - val_acc: 0.9372
Epoch 14/100
42000/42000 [==============================] - 7s 157us/step - loss: 0.3254 -
acc: 0.9039 - val_loss: 0.2284 - val_acc: 0.9373
Epoch 15/100
42000/42000 [==============================] - 7s 160us/step - loss: 0.3165 -
acc: 0.9041 - val_loss: 0.2240 - val_acc: 0.9387
Epoch 16/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.3131 -
acc: 0.9065 - val_loss: 0.2226 - val_acc: 0.9388
Epoch 17/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.3094 -
acc: 0.9074 - val_loss: 0.2195 - val_acc: 0.9390
Epoch 18/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.3035 -
acc: 0.9082 - val_loss: 0.2157 - val_acc: 0.9407
Epoch 19/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.3002 -
acc: 0.9093 - val_loss: 0.2136 - val_acc: 0.9408
Epoch 20/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2998 -
acc: 0.9091 - val_loss: 0.2105 - val_acc: 0.9426
Epoch 21/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2912 -
acc: 0.9112 - val_loss: 0.2092 - val_acc: 0.9422
Epoch 22/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2884 -
acc: 0.9126 - val_loss: 0.2087 - val_acc: 0.9430
Epoch 23/100
42000/42000 [==============================] - 6s 148us/step - loss: 0.2851 -
acc: 0.9141 - val_loss: 0.2035 - val_acc: 0.9442
Epoch 24/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2798 -
acc: 0.9156 - val_loss: 0.2020 - val_acc: 0.9450
Epoch 25/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2775 -
acc: 0.9157 - val_loss: 0.2046 - val_acc: 0.9436
Epoch 26/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2704 -
acc: 0.9171 - val_loss: 0.2000 - val_acc: 0.9457
Epoch 27/100
42000/42000 [==============================] - 7s 160us/step - loss: 0.2707 -
acc: 0.9175 - val_loss: 0.1968 - val_acc: 0.9457
Epoch 28/100
42000/42000 [==============================] - 7s 156us/step - loss: 0.2664 -
acc: 0.9186 - val_loss: 0.1981 - val_acc: 0.9456
Epoch 29/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2658 -
acc: 0.9172 - val_loss: 0.1945 - val_acc: 0.9468
Epoch 30/100
42000/42000 [==============================] - 6s 148us/step - loss: 0.2631 -
acc: 0.9194 - val_loss: 0.1936 - val_acc: 0.9475
Epoch 31/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2646 -
acc: 0.9206 - val_loss: 0.1959 - val_acc: 0.9467
Epoch 32/100
```

```
42000/42000 [==============================] - 7s 170us/step - loss: 0.2598 -
acc: 0.9205 - val_loss: 0.1926 - val_acc: 0.9481
Epoch 33/100
42000/42000 [==============================] - 7s 161us/step - loss: 0.2553 -
acc: 0.9229 - val_loss: 0.1945 - val_acc: 0.9476
Epoch 34/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2570 -
acc: 0.9218 - val_loss: 0.1942 - val_acc: 0.9468
Epoch 35/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2499 -
acc: 0.9224 - val_loss: 0.1909 - val_acc: 0.9476
Epoch 36/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2548 -
acc: 0.9210 - val_loss: 0.1905 - val_acc: 0.9483
Epoch 37/100
42000/42000 [==============================] - 6s 148us/step - loss: 0.2490 -
acc: 0.9228 - val_loss: 0.1913 - val_acc: 0.9475
Epoch 38/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.2485 -
acc: 0.9235 - val_loss: 0.1889 - val_acc: 0.9481
Epoch 39/100
42000/42000 [==============================] - 7s 155us/step - loss: 0.2502 -
acc: 0.9224 - val_loss: 0.1872 - val_acc: 0.9478
Epoch 40/100
42000/42000 [==============================] - 7s 162us/step - loss: 0.2457 -
acc: 0.9241 - val_loss: 0.1876 - val_acc: 0.9474
Epoch 41/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.2446 -
acc: 0.9251 - val_loss: 0.1864 - val_acc: 0.9490
Epoch 42/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2414 -
acc: 0.9271 - val_loss: 0.1868 - val_acc: 0.9485
Epoch 43/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2415 -
acc: 0.9256 - val_loss: 0.1857 - val_acc: 0.9485
Epoch 44/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.2386 -
acc: 0.9264 - val_loss: 0.1848 - val_acc: 0.9482
Epoch 45/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2363 -
acc: 0.9271 - val_loss: 0.1838 - val_acc: 0.9491
Epoch 46/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2331 -
acc: 0.9284 - val_loss: 0.1849 - val_acc: 0.9481
Epoch 47/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2382 -
acc: 0.9275 - val_loss: 0.1866 - val_acc: 0.9484
Epoch 48/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.2350 -
acc: 0.9282 - val_loss: 0.1858 - val_acc: 0.9480
Epoch 49/100
42000/42000 [==============================] - 6s 151us/step - loss: 0.2357 -
acc: 0.9254 - val_loss: 0.1861 - val_acc: 0.9490
Epoch 50/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2352 -
acc: 0.9275 - val_loss: 0.1836 - val_acc: 0.9497
Epoch 51/100
```

```
42000/42000 [==============================] - 6s 151us/step - loss: 0.2301 -
acc: 0.9283 - val_loss: 0.1857 - val_acc: 0.9491
Epoch 52/100
42000/42000 [==============================] - 7s 159us/step - loss: 0.2339 -
acc: 0.9275 - val_loss: 0.1845 - val_acc: 0.9495
Epoch 53/100
42000/42000 [==============================] - 7s 159us/step - loss: 0.2274 -
acc: 0.9282 - val_loss: 0.1813 - val_acc: 0.9491
Epoch 54/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2320 -
acc: 0.9276 - val_loss: 0.1851 - val_acc: 0.9494
Epoch 55/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2287 -
acc: 0.9277 - val_loss: 0.1803 - val_acc: 0.9502
Epoch 56/100
42000/42000 [==============================] - 6s 153us/step - loss: 0.2284 -
acc: 0.9285 - val_loss: 0.1829 - val_acc: 0.9494
Epoch 57/100
42000/42000 [==============================] - 7s 162us/step - loss: 0.2290 -
acc: 0.9285 - val_loss: 0.1816 - val_acc: 0.9501
Epoch 58/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2262 -
acc: 0.9282 - val_loss: 0.1837 - val_acc: 0.9501
Epoch 59/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2281 -
acc: 0.9276 - val_loss: 0.1806 - val_acc: 0.9507
Epoch 60/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2222 -
acc: 0.9318 - val_loss: 0.1803 - val_acc: 0.9500
Epoch 61/100
42000/42000 [==============================] - 6s 148us/step - loss: 0.2198 -
acc: 0.9300 - val_loss: 0.1834 - val_acc: 0.9501
Epoch 62/100
42000/42000 [==============================] - 6s 149us/step - loss: 0.2250 -
acc: 0.9293 - val_loss: 0.1832 - val_acc: 0.9504
Epoch 63/100
42000/42000 [==============================] - 6s 148us/step - loss: 0.2249 -
acc: 0.9300 - val_loss: 0.1805 - val_acc: 0.9499
Epoch 64/100
42000/42000 [==============================] - 6s 150us/step - loss: 0.2233 -
acc: 0.9310 - val_loss: 0.1825 - val_acc: 0.9516
Epoch 65/100
42000/42000 [==============================] - 7s 161us/step - loss: 0.2194 -
acc: 0.9307 - val_loss: 0.1832 - val_acc: 0.9498
Epoch 66/100
42000/42000 [==============================] - 6s 153us/step - loss: 0.2210 -
acc: 0.9309 - val_loss: 0.1824 - val_acc: 0.9498
Epoch 67/100
42000/42000 [==============================] - 6s 148us/step - loss: 0.2151 -
acc: 0.9321 - val_loss: 0.1813 - val_acc: 0.9508
Epoch 68/100
42000/42000 [==============================] - 6s 147us/step - loss: 0.2127 -
acc: 0.9327 - val_loss: 0.1813 - val_acc: 0.9510
Epoch 69/100
42000/42000 [==============================] - 6s 147us/step - loss: 0.2199 -
acc: 0.9298 - val_loss: 0.1796 - val_acc: 0.9516
```

In [26]:
```
plt.plot(h.history['acc'], label='Training Accuracy')
plt.plot(h.history['val_acc'], label='Validation Accuracy')
plt.legend()
```

Out[26]: <matplotlib.legend.Legend at 0x7f942e27a320>