In [32]:
```
# !pip install tensorflow as tf
```

In [33]:
```
# @Secure_Voice_Channel
def generic_vns_function(input_dim, number_dense_layers, classes, units):

    # adjusted to effectively import the libraries
    from keras.models import Sequential
    from keras.layers import Dense
    model = Sequential()

    # adjusted to leverage the correct functions
    for i in range(number_dense_layers):
        model.add(Dense(units=units, input_dim=input_dim,
                              kernel_initializer='normal',
                              activation='relu'))

    model.add(Dense(classes, kernel_initializer='normal',
                              activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',
                  metrics=['accuracy'])
    return model

# adjusted to ingest varying number of epochs, batch sizes and units
def train_model(number_dense_layers, X_train, y_train, X_test, y_test,
                epochs, batch_size, units):
    # adjusted to leverage the generic_vns_function function
    model = generic_vns_function(X_train.shape[1], number_dense_layers,
                                 y_train.shape[1], units)
    model.fit(X_train, y_train, validation_data=(X_test, y_test),
              epochs=epochs, batch_size=batch_size, verbose=2)
    scores = model.evaluate(X_test, y_test, verbose=2)
    print("Baseline Error: %.2f%%" % (100-scores[1]*100))
    return model
```

Computer vision

In [26]:
```
# load data
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

In [27]:
```
# generic_vns_function(input_dim=num_pixels, number_dense_layers=1,
# @ classes=classes, units=1000)
```

In [28]:
```
# reshaping the dataset from 28x28 to 784
num_pixels = X_test.shape[1]*X_test.shape[2] # 28x28 pixels = 784
X_test = X_test.reshape(X_test.shape[0], num_pixels).astype('float32')
X_train = X_train.reshape(X_train.shape[0], num_pixels).astype('float32')
```

In [29]:

```
# normalize inputs
X_train = X_train / 255
X_test = X_test / 255
```

In [30]:

```
# one hot encode outputs
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
classes = y_test.shape[1] # Number of possible classes, = 10.
```

In [81]:

```
train_model(number_dense_layers=1, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=10, batch_size=200,
 units=1000)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
 - 3s - loss: 0.2571 - accuracy: 0.9264 - val_loss: 0.1220 - val_accurac
y: 0.9632
Epoch 2/10
 - 3s - loss: 0.1006 - accuracy: 0.9710 - val_loss: 0.0889 - val_accurac
y: 0.9729
Epoch 3/10
 - 3s - loss: 0.0642 - accuracy: 0.9812 - val_loss: 0.0764 - val_accurac
y: 0.9767
Epoch 4/10
 - 3s - loss: 0.0429 - accuracy: 0.9882 - val_loss: 0.0645 - val_accurac
y: 0.9795
Epoch 5/10
 - 3s - loss: 0.0297 - accuracy: 0.9920 - val_loss: 0.0680 - val_accurac
y: 0.9791
Epoch 6/10
 - 3s - loss: 0.0228 - accuracy: 0.9937 - val_loss: 0.0636 - val_accurac
y: 0.9791
Epoch 7/10
 - 3s - loss: 0.0155 - accuracy: 0.9963 - val_loss: 0.0584 - val_accurac
y: 0.9813
Epoch 8/10
 - 3s - loss: 0.0113 - accuracy: 0.9974 - val_loss: 0.0696 - val_accurac
y: 0.9789
Epoch 9/10
 - 3s - loss: 0.0093 - accuracy: 0.9979 - val_loss: 0.0613 - val_accurac
y: 0.9821
Epoch 10/10
 - 3s - loss: 0.0064 - accuracy: 0.9990 - val_loss: 0.0584 - val_accurac
y: 0.9826
Baseline Error: 1.74%
```

Out[81]:

```
<keras.engine.sequential.Sequential at 0x7f859d5ecbd0>
```

Vision Observations from default model

- Each epoch loss decreases, and accuracy increases on the training dataset
- There seems to be a bit of warning sign around epoch 8: accuracy on the training set went up, but accuracy on the validaiton set went down.

In [97]:

```
# adding an additional dense layer
train_model(number_dense_layers=2, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=10, batch_size=200,
 units=1000)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
 - 6s - loss: 0.1973 - accuracy: 0.9403 - val_loss: 0.1067 - val_accurac
y: 0.9649
Epoch 2/10
 - 6s - loss: 0.0693 - accuracy: 0.9786 - val_loss: 0.0746 - val_accurac
y: 0.9769
Epoch 3/10
 - 6s - loss: 0.0436 - accuracy: 0.9865 - val_loss: 0.0637 - val_accurac
y: 0.9789
Epoch 4/10
 - 6s - loss: 0.0290 - accuracy: 0.9905 - val_loss: 0.0696 - val_accurac
y: 0.9800
Epoch 5/10
 - 6s - loss: 0.0206 - accuracy: 0.9938 - val_loss: 0.0805 - val_accurac
y: 0.9767
Epoch 6/10
 - 6s - loss: 0.0189 - accuracy: 0.9938 - val_loss: 0.0768 - val_accurac
y: 0.9782
Epoch 7/10
 - 6s - loss: 0.0151 - accuracy: 0.9951 - val_loss: 0.0946 - val_accurac
y: 0.9762
Epoch 8/10
 - 6s - loss: 0.0143 - accuracy: 0.9953 - val_loss: 0.0883 - val_accurac
y: 0.9783
Epoch 9/10
 - 6s - loss: 0.0155 - accuracy: 0.9950 - val_loss: 0.0689 - val_accurac
y: 0.9827
Epoch 10/10
 - 6s - loss: 0.0113 - accuracy: 0.9965 - val_loss: 0.0858 - val_accurac
y: 0.9794
Baseline Error: 2.06%
```

Out[97]:

```
<keras.engine.sequential.Sequential at 0x7f7a4bc8ab50>
```

```
Conclusion: adding an additional layer did not increase accuracy for the
vision model
```

In [98]:

```
# increasing epochs
train_model(number_dense_layers=1, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=20, batch_size=200,
 units=1000)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
 - 3s - loss: 0.2592 - accuracy: 0.9256 - val_loss: 0.1316 - val_accurac
y: 0.9592
Epoch 2/20
 - 3s - loss: 0.1007 - accuracy: 0.9703 - val_loss: 0.0898 - val_accurac
y: 0.9733
Epoch 3/20
 - 3s - loss: 0.0633 - accuracy: 0.9816 - val_loss: 0.0735 - val_accurac
y: 0.9774
Epoch 4/20
 - 3s - loss: 0.0433 - accuracy: 0.9877 - val_loss: 0.0618 - val_accurac
y: 0.9793
Epoch 5/20
 - 3s - loss: 0.0310 - accuracy: 0.9915 - val_loss: 0.0619 - val_accurac
y: 0.9792
Epoch 6/20
 - 3s - loss: 0.0220 - accuracy: 0.9944 - val_loss: 0.0654 - val_accurac
y: 0.9794
Epoch 7/20
 - 3s - loss: 0.0163 - accuracy: 0.9959 - val_loss: 0.0614 - val_accurac
y: 0.9816
Epoch 8/20
 - 3s - loss: 0.0121 - accuracy: 0.9974 - val_loss: 0.0588 - val_accurac
y: 0.9820
Epoch 9/20
 - 3s - loss: 0.0087 - accuracy: 0.9984 - val_loss: 0.0592 - val_accurac
y: 0.9807
Epoch 10/20
 - 3s - loss: 0.0058 - accuracy: 0.9991 - val_loss: 0.0577 - val_accurac
y: 0.9826
Epoch 11/20
 - 3s - loss: 0.0068 - accuracy: 0.9984 - val_loss: 0.0789 - val_accurac
y: 0.9772
Epoch 12/20
 - 3s - loss: 0.0081 - accuracy: 0.9979 - val_loss: 0.0662 - val_accurac
y: 0.9815
Epoch 13/20
 - 4s - loss: 0.0036 - accuracy: 0.9994 - val_loss: 0.0605 - val_accurac
y: 0.9827
Epoch 14/20
 - 3s - loss: 0.0023 - accuracy: 0.9998 - val_loss: 0.0643 - val_accurac
y: 0.9820
Epoch 15/20
 - 3s - loss: 0.0021 - accuracy: 0.9997 - val_loss: 0.0650 - val_accurac
y: 0.9817
Epoch 16/20
 - 3s - loss: 0.0019 - accuracy: 0.9997 - val_loss: 0.0812 - val_accurac
y: 0.9800
Epoch 17/20
 - 3s - loss: 0.0168 - accuracy: 0.9944 - val_loss: 0.0753 - val_accurac
y: 0.9799
```

```
Epoch 18/20
 - 3s - loss: 0.0067 - accuracy: 0.9981 - val_loss: 0.0638 - val_accurac
y: 0.9834
Epoch 19/20
 - 3s - loss: 0.0018 - accuracy: 0.9997 - val_loss: 0.0599 - val_accurac
y: 0.9845
Epoch 20/20
 - 3s - loss: 5.1167e-04 - accuracy: 1.0000 - val_loss: 0.0604 - val_accu
racy: 0.9850
Baseline Error: 1.50%
```

```
Out[98]: <keras.engine.sequential.Sequential at 0x7f7a4b3b7650>
```

Conclusion: increasing the number of epochs does increase accuracy for the vision model

In [99]:

```
# decreasing batch size
train_model(number_dense_layers=1, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=20, batch_size=100,
 units=1000)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
 - 5s - loss: 0.2231 - accuracy: 0.9347 - val_loss: 0.1150 - val_accurac
y: 0.9665
Epoch 2/20
 - 5s - loss: 0.0854 - accuracy: 0.9747 - val_loss: 0.0791 - val_accurac
y: 0.9753
Epoch 3/20
 - 5s - loss: 0.0536 - accuracy: 0.9835 - val_loss: 0.0673 - val_accurac
y: 0.9795
Epoch 4/20
 - 5s - loss: 0.0344 - accuracy: 0.9898 - val_loss: 0.0714 - val_accurac
y: 0.9776
Epoch 5/20
 - 5s - loss: 0.0247 - accuracy: 0.9930 - val_loss: 0.0616 - val_accurac
y: 0.9800
Epoch 6/20
 - 5s - loss: 0.0180 - accuracy: 0.9948 - val_loss: 0.0618 - val_accurac
y: 0.9812
Epoch 7/20
 - 5s - loss: 0.0137 - accuracy: 0.9959 - val_loss: 0.0679 - val_accurac
y: 0.9807
Epoch 8/20
 - 5s - loss: 0.0104 - accuracy: 0.9970 - val_loss: 0.0662 - val_accurac
y: 0.9819
Epoch 9/20
 - 5s - loss: 0.0105 - accuracy: 0.9969 - val_loss: 0.0617 - val_accurac
y: 0.9825
Epoch 10/20
 - 5s - loss: 0.0075 - accuracy: 0.9981 - val_loss: 0.0654 - val_accurac
y: 0.9818
Epoch 11/20
 - 5s - loss: 0.0123 - accuracy: 0.9961 - val_loss: 0.0671 - val_accurac
y: 0.9819
Epoch 12/20
 - 4s - loss: 0.0062 - accuracy: 0.9981 - val_loss: 0.0686 - val_accurac
y: 0.9832
Epoch 13/20
 - 5s - loss: 0.0023 - accuracy: 0.9995 - val_loss: 0.0662 - val_accurac
y: 0.9847
Epoch 14/20
 - 5s - loss: 0.0092 - accuracy: 0.9969 - val_loss: 0.0764 - val_accurac
y: 0.9812
Epoch 15/20
 - 5s - loss: 0.0069 - accuracy: 0.9980 - val_loss: 0.0706 - val_accurac
y: 0.9831
Epoch 16/20
 - 5s - loss: 0.0034 - accuracy: 0.9989 - val_loss: 0.0672 - val_accurac
y: 0.9839
Epoch 17/20
 - 5s - loss: 0.0030 - accuracy: 0.9991 - val_loss: 0.0793 - val_accurac
y: 0.9822
```

```
Epoch 18/20
 - 5s - loss: 0.0073 - accuracy: 0.9976 - val_loss: 0.0853 - val_accurac
y: 0.9814
Epoch 19/20
 - 5s - loss: 0.0046 - accuracy: 0.9984 - val_loss: 0.0943 - val_accurac
y: 0.9813
Epoch 20/20
 - 5s - loss: 0.0043 - accuracy: 0.9987 - val_loss: 0.0773 - val_accurac
y: 0.9844
Baseline Error: 1.56%
```

```
Out[99]: <keras.engine.sequential.Sequential at 0x7f7a4b8d4f90>
```

```
Conclusion: decreasing batch size did not increase accuracy for the vision
model
```

In [31]:

```
# increasing epochs
train_model(number_dense_layers=1, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=20, batch_size=200,
 units=2000)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
 - 6s - loss: 0.2255 - accuracy: 0.9326 - val_loss: 0.1198 - val_accurac
y: 0.9647
Epoch 2/20
 - 6s - loss: 0.0833 - accuracy: 0.9757 - val_loss: 0.0786 - val_accurac
y: 0.9756
Epoch 3/20
 - 7s - loss: 0.0521 - accuracy: 0.9845 - val_loss: 0.0726 - val_accurac
y: 0.9765
Epoch 4/20
 - 6s - loss: 0.0334 - accuracy: 0.9901 - val_loss: 0.0663 - val_accurac
y: 0.9792
Epoch 5/20
 - 6s - loss: 0.0211 - accuracy: 0.9944 - val_loss: 0.0709 - val_accurac
y: 0.9778
Epoch 6/20
 - 7s - loss: 0.0141 - accuracy: 0.9965 - val_loss: 0.0624 - val_accurac
y: 0.9811
Epoch 7/20
 - 7s - loss: 0.0104 - accuracy: 0.9976 - val_loss: 0.0602 - val_accurac
y: 0.9826
Epoch 8/20
 - 7s - loss: 0.0064 - accuracy: 0.9987 - val_loss: 0.0666 - val_accurac
y: 0.9806
Epoch 9/20
 - 7s - loss: 0.0082 - accuracy: 0.9980 - val_loss: 0.0661 - val_accurac
y: 0.9818
Epoch 10/20
 - 7s - loss: 0.0089 - accuracy: 0.9974 - val_loss: 0.0684 - val_accurac
y: 0.9810
Epoch 11/20
 - 7s - loss: 0.0080 - accuracy: 0.9978 - val_loss: 0.0745 - val_accurac
y: 0.9809
Epoch 12/20
 - 7s - loss: 0.0116 - accuracy: 0.9960 - val_loss: 0.0759 - val_accurac
y: 0.9793
Epoch 13/20
 - 7s - loss: 0.0068 - accuracy: 0.9980 - val_loss: 0.0677 - val_accurac
y: 0.9825
Epoch 14/20
 - 7s - loss: 0.0038 - accuracy: 0.9991 - val_loss: 0.0712 - val_accurac
y: 0.9817
Epoch 15/20
 - 7s - loss: 0.0023 - accuracy: 0.9995 - val_loss: 0.0657 - val_accurac
y: 0.9825
Epoch 16/20
 - 7s - loss: 0.0031 - accuracy: 0.9991 - val_loss: 0.0774 - val_accurac
y: 0.9828
Epoch 17/20
 - 7s - loss: 0.0044 - accuracy: 0.9988 - val_loss: 0.0859 - val_accurac
y: 0.9782
```

```
Epoch 18/20
 - 7s - loss: 0.0146 - accuracy: 0.9954 - val_loss: 0.0807 - val_accurac
y: 0.9806
Epoch 19/20
 - 6s - loss: 0.0069 - accuracy: 0.9977 - val_loss: 0.0810 - val_accurac
y: 0.9822
Epoch 20/20
 - 6s - loss: 0.0027 - accuracy: 0.9993 - val_loss: 0.0702 - val_accurac
y: 0.9836
Baseline Error: 1.64%
```

Out[31]:
```
<keras.engine.sequential.Sequential at 0x7f980f2a1410>
```

Conclusion:

- the best model turned to be a model with one dense layer and 20 epochs
- there doesn't seem to be an intuitive reason why neither increasing batch size or units can't improve the result. Setting hyperparameters so far seems more than an art than a science.

Speech recognition

In [7]:
```
pip install python_speech_features
```

```
Requirement already satisfied: python_speech_features in /opt/anaconda3/e
nvs/deep37/lib/python3.7/site-packages (0.6)
Note: you may need to restart the kernel to use updated packages.
```

In [13]:
```
pip install pandas
```

```
Collecting pandas
  Downloading pandas-1.1.2-cp37-cp37m-macosx_10_9_x86_64.whl (10.4 MB)
     |████████████████████████████████| 10.4 MB 2.4 MB/s eta 0:00:01
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/anaconda3/e
nvs/deep37/lib/python3.7/site-packages (from pandas) (2.8.1)
Collecting pytz>=2017.2
  Using cached pytz-2020.1-py2.py3-none-any.whl (510 kB)
Requirement already satisfied: numpy>=1.15.4 in /opt/anaconda3/envs/deep3
7/lib/python3.7/site-packages (from pandas) (1.18.1)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/envs/deep37/li
b/python3.7/site-packages (from python-dateutil>=2.7.3->pandas) (1.14.0)
Installing collected packages: pytz, pandas
Successfully installed pandas-1.1.2 pytz-2020.1
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
#import python_speech_features
# mfcc = python_speech_features.mfcc(file, rate)
```

In [11]:
```
# import spoken digit dataset (get_speech_dataset)
from db_utils import get_speech_dataset
(X_train, y_train), (X_test, y_test), (X_val, y_val) = get_speech_dataset()
# (X_train, y_train), (X_test, y_test) = get_speech_dataset()
```

In [12]:

```
# flatten the data
num_pixels = X_test.shape[1]*X_test.shape[2]
X_test = X_test.reshape(X_test.shape[0], num_pixels).astype('float32')
X_train = X_train.reshape(X_train.shape[0], num_pixels).astype('float32')
```

In [13]:

```
# standardize the inputs
import numpy as np
mean = np.mean(X_train)
std = np.std(X_train)
X_train = (X_train-std)/mean
X_test = (X_test-std)/mean
```

In [14]:

```
# one hot encode the outputs to avoid squeueing due to proximity of values
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
classes = y_test.shape[1] # Number of possible classes, = 10.
```

In [86]:

```
train_model(number_dense_layers=1, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=10, batch_size=200,
 units=1000)
```

```
Train on 18620 samples, validate on 2552 samples
Epoch 1/10
 - 3s - loss: 26.8836 - accuracy: 0.3362 - val_loss: 3.2326 - val_accurac
y: 0.4189
Epoch 2/10
 - 3s - loss: 2.0473 - accuracy: 0.4948 - val_loss: 1.9372 - val_accurac
y: 0.5106
Epoch 3/10
 - 3s - loss: 1.3198 - accuracy: 0.5996 - val_loss: 1.5285 - val_accurac
y: 0.5803
Epoch 4/10
 - 3s - loss: 1.0028 - accuracy: 0.6780 - val_loss: 1.2918 - val_accurac
y: 0.6364
Epoch 5/10
 - 3s - loss: 0.8033 - accuracy: 0.7401 - val_loss: 1.1430 - val_accurac
y: 0.6642
Epoch 6/10
 - 3s - loss: 0.6661 - accuracy: 0.7844 - val_loss: 1.0823 - val_accurac
y: 0.6947
Epoch 7/10
 - 3s - loss: 0.5638 - accuracy: 0.8145 - val_loss: 1.0386 - val_accurac
y: 0.7092
Epoch 8/10
 - 3s - loss: 0.4798 - accuracy: 0.8439 - val_loss: 0.9570 - val_accurac
y: 0.7386
Epoch 9/10
 - 3s - loss: 0.4087 - accuracy: 0.8653 - val_loss: 0.9334 - val_accurac
y: 0.7496
Epoch 10/10
 - 3s - loss: 0.3387 - accuracy: 0.8905 - val_loss: 0.9283 - val_accurac
y: 0.7645
Baseline Error: 23.55%
```

Out[86]:

```
<keras.engine.sequential.Sequential at 0x7f859de676d0>
```

Speech - Conclusions on default model

- The model we developed generalizes between computer vision and speech recognition
- In contrast to computer vision, the baseline error remains 23.55%.
- It is reasonable to assume running more epochs could help as both train and validation error are still decreasing
- It also seems reasonable to assume that other model specs could further improve the model.

In [104]:

```
# adding an additional dense layer
train_model(number_dense_layers=2, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=10, batch_size=200,
 units=1000)
```

```
Train on 18620 samples, validate on 2552 samples
Epoch 1/10
 - 4s - loss: 22.9293 - accuracy: 0.3347 - val_loss: 1.9929 - val_accurac
y: 0.4412
Epoch 2/10
 - 4s - loss: 1.4581 - accuracy: 0.5432 - val_loss: 1.5184 - val_accurac
y: 0.5517
Epoch 3/10
 - 4s - loss: 1.0155 - accuracy: 0.6739 - val_loss: 1.3201 - val_accurac
y: 0.6148
Epoch 4/10
 - 4s - loss: 0.7683 - accuracy: 0.7487 - val_loss: 1.1635 - val_accurac
y: 0.6646
Epoch 5/10
 - 4s - loss: 0.5859 - accuracy: 0.8044 - val_loss: 1.0080 - val_accurac
y: 0.7104
Epoch 6/10
 - 4s - loss: 0.4675 - accuracy: 0.8420 - val_loss: 1.0629 - val_accurac
y: 0.7132
Epoch 7/10
 - 4s - loss: 0.3709 - accuracy: 0.8748 - val_loss: 0.9579 - val_accurac
y: 0.7382
Epoch 8/10
 - 4s - loss: 0.2747 - accuracy: 0.9086 - val_loss: 0.9272 - val_accurac
y: 0.7582
Epoch 9/10
 - 4s - loss: 0.2252 - accuracy: 0.9231 - val_loss: 0.9172 - val_accurac
y: 0.7680
Epoch 10/10
 - 4s - loss: 0.1838 - accuracy: 0.9376 - val_loss: 0.9212 - val_accurac
y: 0.7751
Baseline Error: 22.49%
```

Out[104]:

```
<keras.engine.sequential.Sequential at 0x7f7a4d20bc10>
```

```
Conclusion: adding a layer improves the speech model
```

In [105]:

```
# increasing epochs while using one layer
train_model(number_dense_layers=1, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=20, batch_size=200,
 units=1000)
```

```
Train on 18620 samples, validate on 2552 samples
Epoch 1/20
 - 3s - loss: 28.4327 - accuracy: 0.3272 - val_loss: 3.2682 - val_accurac
y: 0.4122
Epoch 2/20
 - 3s - loss: 2.0190 - accuracy: 0.4762 - val_loss: 1.8003 - val_accurac
y: 0.4765
Epoch 3/20
 - 3s - loss: 1.3007 - accuracy: 0.5833 - val_loss: 1.4661 - val_accurac
y: 0.5748
Epoch 4/20
 - 3s - loss: 1.0065 - accuracy: 0.6689 - val_loss: 1.3271 - val_accurac
y: 0.6148
Epoch 5/20
 - 3s - loss: 0.8277 - accuracy: 0.7251 - val_loss: 1.2889 - val_accurac
y: 0.6399
Epoch 6/20
 - 3s - loss: 0.7078 - accuracy: 0.7653 - val_loss: 1.0619 - val_accurac
y: 0.6842
Epoch 7/20
 - 3s - loss: 0.5941 - accuracy: 0.8014 - val_loss: 1.0487 - val_accurac
y: 0.6920
Epoch 8/20
 - 3s - loss: 0.4959 - accuracy: 0.8363 - val_loss: 0.9693 - val_accurac
y: 0.7257
Epoch 9/20
 - 3s - loss: 0.4335 - accuracy: 0.8550 - val_loss: 0.9776 - val_accurac
y: 0.7237
Epoch 10/20
 - 3s - loss: 0.3765 - accuracy: 0.8758 - val_loss: 0.9247 - val_accurac
y: 0.7363
Epoch 11/20
 - 3s - loss: 0.3342 - accuracy: 0.8898 - val_loss: 0.9370 - val_accurac
y: 0.7469
Epoch 12/20
 - 3s - loss: 0.2957 - accuracy: 0.9043 - val_loss: 0.9256 - val_accurac
y: 0.7598
Epoch 13/20
 - 3s - loss: 0.2594 - accuracy: 0.9139 - val_loss: 0.9339 - val_accurac
y: 0.7629
Epoch 14/20
 - 3s - loss: 0.2288 - accuracy: 0.9248 - val_loss: 0.9318 - val_accurac
y: 0.7598
Epoch 15/20
 - 3s - loss: 0.2086 - accuracy: 0.9332 - val_loss: 0.9046 - val_accurac
y: 0.7759
Epoch 16/20
 - 3s - loss: 0.1777 - accuracy: 0.9430 - val_loss: 0.9177 - val_accurac
y: 0.7806
Epoch 17/20
 - 3s - loss: 0.1540 - accuracy: 0.9526 - val_loss: 0.9221 - val_accurac
y: 0.7884
```

```
Epoch 18/20
 - 3s - loss: 0.1391 - accuracy: 0.9573 - val_loss: 0.9266 - val_accurac
y: 0.7915
Epoch 19/20
 - 3s - loss: 0.1323 - accuracy: 0.9595 - val_loss: 0.9683 - val_accurac
y: 0.7845
Epoch 20/20
 - 3s - loss: 0.1216 - accuracy: 0.9633 - val_loss: 0.9257 - val_accurac
y: 0.7911
Baseline Error: 20.89%
```

```
Out[105]: <keras.engine.sequential.Sequential at 0x7f7a4eb1c710>
```

Conclusion: increasing epochs (without increasing the model) increases accuracy. The model also doesn't seem to be done learning, so we'll increase the number of epochs in the next model. We'll also combine more epochs with two layers.

In [7]:

```
# two layers; increased epochs
train_model(number_dense_layers=2, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=30, batch_size=200,
 units=1000)
```

```
Using TensorFlow backend.

Train on 18620 samples, validate on 2552 samples
Epoch 1/30
 - 4s - loss: 22.8555 - accuracy: 0.3086 - val_loss: 1.9293 - val_accurac
y: 0.4306
Epoch 2/30
 - 4s - loss: 1.4847 - accuracy: 0.5312 - val_loss: 1.5164 - val_accurac
y: 0.5525
Epoch 3/30
 - 4s - loss: 1.0594 - accuracy: 0.6601 - val_loss: 1.2104 - val_accurac
y: 0.6309
Epoch 4/30
 - 4s - loss: 0.7857 - accuracy: 0.7429 - val_loss: 1.0193 - val_accurac
y: 0.6799
Epoch 5/30
 - 4s - loss: 0.6033 - accuracy: 0.8061 - val_loss: 0.9331 - val_accurac
y: 0.7261
Epoch 6/30
 - 4s - loss: 0.4828 - accuracy: 0.8413 - val_loss: 0.8836 - val_accurac
y: 0.7390
Epoch 7/30
 - 4s - loss: 0.4173 - accuracy: 0.8633 - val_loss: 0.8749 - val_accurac
y: 0.7520
Epoch 8/30
 - 4s - loss: 0.3311 - accuracy: 0.8908 - val_loss: 0.8320 - val_accurac
y: 0.7696
Epoch 9/30
 - 4s - loss: 0.2485 - accuracy: 0.9173 - val_loss: 0.7892 - val_accurac
y: 0.7958
Epoch 10/30
 - 4s - loss: 0.2153 - accuracy: 0.9266 - val_loss: 0.8490 - val_accurac
y: 0.7739
Epoch 11/30
 - 4s - loss: 0.1782 - accuracy: 0.9404 - val_loss: 0.8145 - val_accurac
y: 0.8021
Epoch 12/30
 - 4s - loss: 0.1568 - accuracy: 0.9476 - val_loss: 0.7907 - val_accurac
y: 0.8037
Epoch 13/30
 - 4s - loss: 0.1139 - accuracy: 0.9628 - val_loss: 0.8878 - val_accurac
y: 0.7998
Epoch 14/30
 - 4s - loss: 0.1218 - accuracy: 0.9596 - val_loss: 0.8368 - val_accurac
y: 0.8103
Epoch 15/30
 - 4s - loss: 0.1151 - accuracy: 0.9606 - val_loss: 0.8734 - val_accurac
y: 0.8096
Epoch 16/30
 - 4s - loss: 0.0901 - accuracy: 0.9709 - val_loss: 0.8813 - val_accurac
y: 0.8194
Epoch 17/30
```

```
 - 4s - loss: 0.0700 - accuracy: 0.9769 - val_loss: 0.8639 - val_accurac
y: 0.8190
Epoch 18/30
 - 4s - loss: 0.0633 - accuracy: 0.9789 - val_loss: 0.9201 - val_accurac
y: 0.8143
Epoch 19/30
 - 4s - loss: 0.0847 - accuracy: 0.9713 - val_loss: 1.0225 - val_accurac
y: 0.8107
Epoch 20/30
 - 4s - loss: 0.0532 - accuracy: 0.9832 - val_loss: 0.9768 - val_accurac
y: 0.8186
Epoch 21/30
 - 4s - loss: 0.0450 - accuracy: 0.9857 - val_loss: 1.0767 - val_accurac
y: 0.8009
Epoch 22/30
 - 4s - loss: 0.0497 - accuracy: 0.9840 - val_loss: 1.1091 - val_accurac
y: 0.8068
Epoch 23/30
 - 4s - loss: 0.1143 - accuracy: 0.9627 - val_loss: 1.0087 - val_accurac
y: 0.8139
Epoch 24/30
 - 4s - loss: 0.1522 - accuracy: 0.9505 - val_loss: 1.7775 - val_accurac
y: 0.7484
Epoch 25/30
 - 4s - loss: 0.4334 - accuracy: 0.8849 - val_loss: 1.4541 - val_accurac
y: 0.7567
Epoch 26/30
 - 4s - loss: 0.2606 - accuracy: 0.9229 - val_loss: 0.9987 - val_accurac
y: 0.8088
Epoch 27/30
 - 4s - loss: 0.1932 - accuracy: 0.9393 - val_loss: 0.9455 - val_accurac
y: 0.8264
Epoch 28/30
 - 4s - loss: 0.1248 - accuracy: 0.9584 - val_loss: 0.9811 - val_accurac
y: 0.8268
Epoch 29/30
 - 4s - loss: 0.0709 - accuracy: 0.9759 - val_loss: 0.9585 - val_accurac
y: 0.8327
Epoch 30/30
 - 4s - loss: 0.0266 - accuracy: 0.9905 - val_loss: 0.9036 - val_accurac
y: 0.8413
Baseline Error: 15.87%
```

```
Out[7]: <keras.engine.sequential.Sequential at 0x7fa3b4b18250>
```

```
Conclusion: model error dropped significantly from 20+ to 16+ percent.
Validation continues to go down consistently. We'll increase the epochs.
```

In [15]:

```
# adding more epochs as validation accuracy is still increasing
train_model(number_dense_layers=2, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=40, batch_size=200,
 units=1000)
```

```
Train on 18620 samples, validate on 2552 samples
Epoch 1/40
 - 4s - loss: 28.7320 - accuracy: 0.3277 - val_loss: 2.0146 - val_accurac
y: 0.4224
Epoch 2/40
 - 4s - loss: 1.4722 - accuracy: 0.5302 - val_loss: 1.5860 - val_accurac
y: 0.5278
Epoch 3/40
 - 4s - loss: 1.0978 - accuracy: 0.6434 - val_loss: 1.2976 - val_accurac
y: 0.6101
Epoch 4/40
 - 4s - loss: 0.8248 - accuracy: 0.7290 - val_loss: 1.1451 - val_accurac
y: 0.6681
Epoch 5/40
 - 4s - loss: 0.6575 - accuracy: 0.7840 - val_loss: 1.0502 - val_accurac
y: 0.6967
Epoch 6/40
 - 4s - loss: 0.5269 - accuracy: 0.8267 - val_loss: 0.9702 - val_accurac
y: 0.7183
Epoch 7/40
 - 4s - loss: 0.4284 - accuracy: 0.8612 - val_loss: 0.9807 - val_accurac
y: 0.7226
Epoch 8/40
 - 4s - loss: 0.3497 - accuracy: 0.8838 - val_loss: 0.8753 - val_accurac
y: 0.7712
Epoch 9/40
 - 4s - loss: 0.2888 - accuracy: 0.9050 - val_loss: 0.8916 - val_accurac
y: 0.7641
Epoch 10/40
 - 4s - loss: 0.2471 - accuracy: 0.9156 - val_loss: 0.9434 - val_accurac
y: 0.7641
Epoch 11/40
 - 4s - loss: 0.2001 - accuracy: 0.9337 - val_loss: 0.8954 - val_accurac
y: 0.7778
Epoch 12/40
 - 4s - loss: 0.1903 - accuracy: 0.9348 - val_loss: 0.9532 - val_accurac
y: 0.7853
Epoch 13/40
 - 4s - loss: 0.1306 - accuracy: 0.9578 - val_loss: 0.9695 - val_accurac
y: 0.7798
Epoch 14/40
 - 4s - loss: 0.1105 - accuracy: 0.9649 - val_loss: 0.8898 - val_accurac
y: 0.7970
Epoch 15/40
 - 4s - loss: 0.1095 - accuracy: 0.9640 - val_loss: 0.9157 - val_accurac
y: 0.8025
Epoch 16/40
 - 4s - loss: 0.0869 - accuracy: 0.9718 - val_loss: 0.9532 - val_accurac
y: 0.7955
Epoch 17/40
 - 4s - loss: 0.0701 - accuracy: 0.9782 - val_loss: 0.9095 - val_accurac
y: 0.8115
```

```
Epoch 18/40
 - 4s - loss: 0.0469 - accuracy: 0.9867 - val_loss: 1.0180 - val_accurac
y: 0.8002
Epoch 19/40
 - 4s - loss: 0.0510 - accuracy: 0.9832 - val_loss: 0.9875 - val_accurac
y: 0.8115
Epoch 20/40
 - 4s - loss: 0.0440 - accuracy: 0.9873 - val_loss: 0.9434 - val_accurac
y: 0.8217
Epoch 21/40
 - 4s - loss: 0.0304 - accuracy: 0.9917 - val_loss: 1.0081 - val_accurac
y: 0.8178
Epoch 22/40
 - 4s - loss: 0.0689 - accuracy: 0.9774 - val_loss: 1.1835 - val_accurac
y: 0.7861
Epoch 23/40
 - 4s - loss: 0.1720 - accuracy: 0.9408 - val_loss: 1.1325 - val_accurac
y: 0.7978
Epoch 24/40
 - 4s - loss: 0.1920 - accuracy: 0.9396 - val_loss: 1.4916 - val_accurac
y: 0.7551
Epoch 25/40
 - 4s - loss: 0.3465 - accuracy: 0.9020 - val_loss: 1.2406 - val_accurac
y: 0.7900
Epoch 26/40
 - 4s - loss: 0.2753 - accuracy: 0.9190 - val_loss: 1.3598 - val_accurac
y: 0.7813
Epoch 27/40
 - 4s - loss: 0.2280 - accuracy: 0.9298 - val_loss: 1.1924 - val_accurac
y: 0.8072
Epoch 28/40
 - 4s - loss: 0.1195 - accuracy: 0.9577 - val_loss: 1.0743 - val_accurac
y: 0.8135
Epoch 29/40
 - 4s - loss: 0.1211 - accuracy: 0.9608 - val_loss: 1.2234 - val_accurac
y: 0.8013
Epoch 30/40
 - 4s - loss: 0.1247 - accuracy: 0.9599 - val_loss: 1.2023 - val_accurac
y: 0.8135
Epoch 31/40
 - 4s - loss: 0.0876 - accuracy: 0.9708 - val_loss: 1.1661 - val_accurac
y: 0.8131
Epoch 32/40
 - 4s - loss: 0.1036 - accuracy: 0.9663 - val_loss: 1.1898 - val_accurac
y: 0.8217
Epoch 33/40
 - 4s - loss: 0.0635 - accuracy: 0.9779 - val_loss: 1.1359 - val_accurac
y: 0.8284
Epoch 34/40
 - 4s - loss: 0.0534 - accuracy: 0.9825 - val_loss: 1.1293 - val_accurac
y: 0.8225
Epoch 35/40
 - 4s - loss: 0.0780 - accuracy: 0.9752 - val_loss: 1.3012 - val_accurac
y: 0.8096
Epoch 36/40
 - 4s - loss: 0.1010 - accuracy: 0.9688 - val_loss: 1.2635 - val_accurac
y: 0.8197
```

```
Epoch 37/40
 - 4s - loss: 0.0950 - accuracy: 0.9695 - val_loss: 1.1657 - val_accurac
y: 0.8315
Epoch 38/40
 - 4s - loss: 0.1196 - accuracy: 0.9641 - val_loss: 1.3705 - val_accurac
y: 0.8033
Epoch 39/40
 - 4s - loss: 0.2309 - accuracy: 0.9380 - val_loss: 1.5089 - val_accurac
y: 0.7958
Epoch 40/40
 - 4s - loss: 0.2138 - accuracy: 0.9433 - val_loss: 1.6747 - val_accurac
y: 0.8013
Baseline Error: 19.87%
```

```
Out[15]: <keras.engine.sequential.Sequential at 0x7fa3b4b49310>
```

Conclusion: we observe signs of overfitting (decreasing validation errors) as of epoch 35. We'll stop the next model there, and explore batch size next.

In [18]:

```
train_model(number_dense_layers=2, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=35, batch_size=100,
 units=1000)
```

```
Train on 18620 samples, validate on 2552 samples
Epoch 1/35
 - 7s - loss: 14.3605 - accuracy: 0.3959 - val_loss: 1.5848 - val_accurac
y: 0.5223
Epoch 2/35
 - 6s - loss: 1.1489 - accuracy: 0.6321 - val_loss: 1.3096 - val_accurac
y: 0.6105
Epoch 3/35
 - 8s - loss: 0.7934 - accuracy: 0.7478 - val_loss: 1.0109 - val_accurac
y: 0.6928
Epoch 4/35
 - 7s - loss: 0.5902 - accuracy: 0.8072 - val_loss: 0.9551 - val_accurac
y: 0.7198
Epoch 5/35
 - 7s - loss: 0.4674 - accuracy: 0.8468 - val_loss: 0.8564 - val_accurac
y: 0.7582
Epoch 6/35
 - 7s - loss: 0.4090 - accuracy: 0.8645 - val_loss: 0.8032 - val_accurac
y: 0.7782
Epoch 7/35
 - 7s - loss: 0.3383 - accuracy: 0.8861 - val_loss: 0.8636 - val_accurac
y: 0.7766
Epoch 8/35
 - 6s - loss: 0.2861 - accuracy: 0.9038 - val_loss: 0.8317 - val_accurac
y: 0.7947
Epoch 9/35
 - 6s - loss: 0.2448 - accuracy: 0.9194 - val_loss: 0.9595 - val_accurac
y: 0.7829
Epoch 10/35
 - 6s - loss: 0.2617 - accuracy: 0.9142 - val_loss: 1.0185 - val_accurac
y: 0.7731
Epoch 11/35
 - 6s - loss: 0.2535 - accuracy: 0.9192 - val_loss: 1.0130 - val_accurac
y: 0.7853
Epoch 12/35
 - 6s - loss: 0.2712 - accuracy: 0.9129 - val_loss: 1.1181 - val_accurac
y: 0.7798
Epoch 13/35
 - 6s - loss: 0.2252 - accuracy: 0.9300 - val_loss: 0.9736 - val_accurac
y: 0.8127
Epoch 14/35
 - 6s - loss: 0.3075 - accuracy: 0.9088 - val_loss: 1.1095 - val_accurac
y: 0.7915
Epoch 15/35
 - 6s - loss: 0.2317 - accuracy: 0.9299 - val_loss: 1.0414 - val_accurac
y: 0.8111
Epoch 16/35
 - 6s - loss: 0.2065 - accuracy: 0.9332 - val_loss: 0.9959 - val_accurac
y: 0.8170
Epoch 17/35
 - 6s - loss: 0.1958 - accuracy: 0.9394 - val_loss: 1.2746 - val_accurac
y: 0.7958
Epoch 18/35
```

```
 - 6s - loss: 0.2684 - accuracy: 0.9222 - val_loss: 1.0934 - val_accurac
y: 0.7962
Epoch 19/35
 - 6s - loss: 0.2154 - accuracy: 0.9340 - val_loss: 1.0218 - val_accurac
y: 0.8256
Epoch 20/35
 - 6s - loss: 0.2307 - accuracy: 0.9322 - val_loss: 1.1134 - val_accurac
y: 0.8174
Epoch 21/35
 - 6s - loss: 0.2086 - accuracy: 0.9387 - val_loss: 1.1377 - val_accurac
y: 0.8135
Epoch 22/35
 - 6s - loss: 0.1719 - accuracy: 0.9493 - val_loss: 1.2245 - val_accurac
y: 0.8154
Epoch 23/35
 - 6s - loss: 0.2080 - accuracy: 0.9385 - val_loss: 1.1235 - val_accurac
y: 0.8178
Epoch 24/35
 - 7s - loss: 0.1416 - accuracy: 0.9555 - val_loss: 1.1198 - val_accurac
y: 0.8382
Epoch 25/35
 - 6s - loss: 0.1788 - accuracy: 0.9494 - val_loss: 0.9720 - val_accurac
y: 0.8389
Epoch 26/35
 - 6s - loss: 0.1339 - accuracy: 0.9579 - val_loss: 1.2324 - val_accurac
y: 0.8150
Epoch 27/35
 - 6s - loss: 0.1541 - accuracy: 0.9548 - val_loss: 1.1913 - val_accurac
y: 0.8237
Epoch 28/35
 - 6s - loss: 0.1675 - accuracy: 0.9519 - val_loss: 1.0991 - val_accurac
y: 0.8370
Epoch 29/35
 - 6s - loss: 0.1599 - accuracy: 0.9544 - val_loss: 1.2818 - val_accurac
y: 0.8245
Epoch 30/35
 - 6s - loss: 0.1317 - accuracy: 0.9598 - val_loss: 0.9771 - val_accurac
y: 0.8417
Epoch 31/35
 - 6s - loss: 0.1353 - accuracy: 0.9595 - val_loss: 1.2327 - val_accurac
y: 0.8256
Epoch 32/35
 - 6s - loss: 0.1602 - accuracy: 0.9565 - val_loss: 1.0702 - val_accurac
y: 0.8487
Epoch 33/35
 - 6s - loss: 0.1002 - accuracy: 0.9690 - val_loss: 1.2223 - val_accurac
y: 0.8370
Epoch 34/35
 - 6s - loss: 0.1457 - accuracy: 0.9579 - val_loss: 1.1952 - val_accurac
y: 0.8484
Epoch 35/35
 - 6s - loss: 0.1211 - accuracy: 0.9647 - val_loss: 1.1628 - val_accurac
y: 0.8460
Baseline Error: 15.40%
```

```
Out[18]: <keras.engine.sequential.Sequential at 0x7f9b76b17ed0>
```

```
Conclusion: lowering batch size seems to improve the model mildly.
```

In [20]:

```
train_model(number_dense_layers=2, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=35, batch_size=100,
 units=2000)
```

```
Train on 18620 samples, validate on 2552 samples
Epoch 1/35
 - 18s - loss: 51.2224 - accuracy: 0.3471 - val_loss: 1.8722 - val_accura
cy: 0.4165
Epoch 2/35
 - 18s - loss: 1.4065 - accuracy: 0.5387 - val_loss: 1.5753 - val_accurac
y: 0.5255
Epoch 3/35
 - 18s - loss: 1.0712 - accuracy: 0.6488 - val_loss: 1.2885 - val_accurac
y: 0.6164
Epoch 4/35
 - 18s - loss: 0.8431 - accuracy: 0.7248 - val_loss: 1.2451 - val_accurac
y: 0.6528
Epoch 5/35
 - 18s - loss: 0.7023 - accuracy: 0.7705 - val_loss: 1.1592 - val_accurac
y: 0.6814
Epoch 6/35
 - 18s - loss: 0.5708 - accuracy: 0.8150 - val_loss: 1.1567 - val_accurac
y: 0.7081
Epoch 7/35
 - 18s - loss: 0.5881 - accuracy: 0.8125 - val_loss: 1.1787 - val_accurac
y: 0.7116
Epoch 8/35
 - 18s - loss: 0.5060 - accuracy: 0.8424 - val_loss: 1.1888 - val_accurac
y: 0.7339
Epoch 9/35
 - 18s - loss: 0.5177 - accuracy: 0.8416 - val_loss: 1.1389 - val_accurac
y: 0.7394
Epoch 10/35
 - 17s - loss: 0.4844 - accuracy: 0.8508 - val_loss: 1.0547 - val_accurac
y: 0.7680
Epoch 11/35
 - 18s - loss: 0.4275 - accuracy: 0.8704 - val_loss: 1.1926 - val_accurac
y: 0.7653
Epoch 12/35
 - 20s - loss: 0.4038 - accuracy: 0.8795 - val_loss: 1.1648 - val_accurac
y: 0.7621
Epoch 13/35
 - 18s - loss: 0.4808 - accuracy: 0.8622 - val_loss: 1.1720 - val_accurac
y: 0.7594
Epoch 14/35
 - 19s - loss: 0.4381 - accuracy: 0.8785 - val_loss: 1.3413 - val_accurac
y: 0.7665
Epoch 15/35
 - 18s - loss: 0.3996 - accuracy: 0.8861 - val_loss: 1.2496 - val_accurac
y: 0.7786
Epoch 16/35
 - 18s - loss: 0.3506 - accuracy: 0.9001 - val_loss: 1.2700 - val_accurac
y: 0.7829
Epoch 17/35
 - 18s - loss: 0.3608 - accuracy: 0.9016 - val_loss: 1.1891 - val_accurac
y: 0.7810
Epoch 18/35
```

```
 - 18s - loss: 0.4199 - accuracy: 0.8914 - val_loss: 1.3686 - val_accurac
y: 0.7708
Epoch 19/35
 - 18s - loss: 0.4007 - accuracy: 0.8959 - val_loss: 1.3264 - val_accurac
y: 0.7782
Epoch 20/35
 - 18s - loss: 0.3722 - accuracy: 0.9017 - val_loss: 1.5191 - val_accurac
y: 0.7606
Epoch 21/35
 - 19s - loss: 0.3267 - accuracy: 0.9122 - val_loss: 1.2879 - val_accurac
y: 0.7915
Epoch 22/35
 - 18s - loss: 0.3093 - accuracy: 0.9150 - val_loss: 1.2363 - val_accurac
y: 0.8025
Epoch 23/35
 - 18s - loss: 0.2410 - accuracy: 0.9297 - val_loss: 1.1787 - val_accurac
y: 0.8056
Epoch 24/35
 - 18s - loss: 0.2377 - accuracy: 0.9308 - val_loss: 1.2409 - val_accurac
y: 0.8072
Epoch 25/35
 - 18s - loss: 0.2595 - accuracy: 0.9311 - val_loss: 1.3103 - val_accurac
y: 0.8131
Epoch 26/35
 - 19s - loss: 0.2133 - accuracy: 0.9369 - val_loss: 1.2300 - val_accurac
y: 0.8068
Epoch 27/35
 - 19s - loss: 0.2501 - accuracy: 0.9302 - val_loss: 1.5590 - val_accurac
y: 0.7896
Epoch 28/35
 - 21s - loss: 0.2499 - accuracy: 0.9338 - val_loss: 1.3390 - val_accurac
y: 0.8096
Epoch 29/35
 - 18s - loss: 0.1947 - accuracy: 0.9456 - val_loss: 1.2830 - val_accurac
y: 0.8076
Epoch 30/35
 - 19s - loss: 0.2304 - accuracy: 0.9392 - val_loss: 1.4268 - val_accurac
y: 0.8068
Epoch 31/35
 - 19s - loss: 0.1873 - accuracy: 0.9488 - val_loss: 1.3671 - val_accurac
y: 0.7943
Epoch 32/35
 - 20s - loss: 0.1773 - accuracy: 0.9511 - val_loss: 1.1776 - val_accurac
y: 0.8311
Epoch 33/35
 - 18s - loss: 0.1918 - accuracy: 0.9484 - val_loss: 1.3365 - val_accurac
y: 0.8197
Epoch 34/35
 - 19s - loss: 0.2028 - accuracy: 0.9464 - val_loss: 1.2438 - val_accurac
y: 0.8233
Epoch 35/35
 - 18s - loss: 0.1607 - accuracy: 0.9541 - val_loss: 1.1784 - val_accurac
y: 0.8260
Baseline Error: 17.40%
```

```
Out[20]: <keras.engine.sequential.Sequential at 0x7f9808209650>
```

```
Conclusion: increasing units made the model worse. Maybe we should reduce?
```

In [23]:
```
train_model(number_dense_layers=2, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=35, batch_size=100,
 units=750)
```

```
Train on 18620 samples, validate on 2552 samples
Epoch 1/35
 - 5s - loss: 9.4356 - accuracy: 0.4048 - val_loss: 1.5659 - val_accurac
y: 0.5157
Epoch 2/35
 - 5s - loss: 1.1549 - accuracy: 0.6263 - val_loss: 1.2942 - val_accurac
y: 0.6074
Epoch 3/35
 - 5s - loss: 0.7875 - accuracy: 0.7426 - val_loss: 0.9384 - val_accurac
y: 0.7288
Epoch 4/35
 - 5s - loss: 0.5932 - accuracy: 0.8032 - val_loss: 0.8631 - val_accurac
y: 0.7484
Epoch 5/35
 - 5s - loss: 0.4716 - accuracy: 0.8456 - val_loss: 0.8662 - val_accurac
y: 0.7594
Epoch 6/35
 - 5s - loss: 0.3770 - accuracy: 0.8750 - val_loss: 0.9100 - val_accurac
y: 0.7567
Epoch 7/35
 - 5s - loss: 0.3466 - accuracy: 0.8868 - val_loss: 0.7817 - val_accurac
y: 0.7978
Epoch 8/35
 - 5s - loss: 0.2738 - accuracy: 0.9085 - val_loss: 0.8506 - val_accurac
y: 0.7892
Epoch 9/35
 - 5s - loss: 0.2450 - accuracy: 0.9175 - val_loss: 0.8997 - val_accurac
y: 0.7900
Epoch 10/35
 - 5s - loss: 0.2533 - accuracy: 0.9166 - val_loss: 0.8475 - val_accurac
y: 0.8123
Epoch 11/35
 - 5s - loss: 0.2845 - accuracy: 0.9076 - val_loss: 0.9785 - val_accurac
y: 0.7908
Epoch 12/35
 - 5s - loss: 0.3152 - accuracy: 0.9025 - val_loss: 0.9470 - val_accurac
y: 0.8013
Epoch 13/35
 - 5s - loss: 0.2323 - accuracy: 0.9245 - val_loss: 0.9522 - val_accurac
y: 0.8100
Epoch 14/35
 - 5s - loss: 0.2348 - accuracy: 0.9222 - val_loss: 0.9492 - val_accurac
y: 0.8045
Epoch 15/35
 - 5s - loss: 0.2034 - accuracy: 0.9351 - val_loss: 1.0116 - val_accurac
y: 0.8154
Epoch 16/35
 - 5s - loss: 0.2436 - accuracy: 0.9259 - val_loss: 1.0681 - val_accurac
y: 0.8194
Epoch 17/35
 - 5s - loss: 0.1927 - accuracy: 0.9403 - val_loss: 1.1365 - val_accurac
y: 0.8021
Epoch 18/35
```

```
 - 5s - loss: 0.2366 - accuracy: 0.9286 - val_loss: 1.0460 - val_accurac
y: 0.8221
Epoch 19/35
 - 5s - loss: 0.1395 - accuracy: 0.9531 - val_loss: 1.2403 - val_accurac
y: 0.8088
Epoch 20/35
 - 5s - loss: 0.1811 - accuracy: 0.9461 - val_loss: 1.1872 - val_accurac
y: 0.8190
Epoch 21/35
 - 5s - loss: 0.1582 - accuracy: 0.9501 - val_loss: 1.1546 - val_accurac
y: 0.8111
Epoch 22/35
 - 5s - loss: 0.1519 - accuracy: 0.9541 - val_loss: 1.1592 - val_accurac
y: 0.8201
Epoch 23/35
 - 5s - loss: 0.1774 - accuracy: 0.9475 - val_loss: 1.2742 - val_accurac
y: 0.8009
Epoch 24/35
 - 5s - loss: 0.2262 - accuracy: 0.9347 - val_loss: 1.2337 - val_accurac
y: 0.8190
Epoch 25/35
 - 5s - loss: 0.1826 - accuracy: 0.9450 - val_loss: 0.9523 - val_accurac
y: 0.8350
Epoch 26/35
 - 5s - loss: 0.1567 - accuracy: 0.9516 - val_loss: 1.0607 - val_accurac
y: 0.8346
Epoch 27/35
 - 5s - loss: 0.1481 - accuracy: 0.9549 - val_loss: 1.0552 - val_accurac
y: 0.8342
Epoch 28/35
 - 5s - loss: 0.1000 - accuracy: 0.9680 - val_loss: 0.9720 - val_accurac
y: 0.8519
Epoch 29/35
 - 5s - loss: 0.1531 - accuracy: 0.9555 - val_loss: 0.9742 - val_accurac
y: 0.8260
Epoch 30/35
 - 5s - loss: 0.1373 - accuracy: 0.9583 - val_loss: 1.1648 - val_accurac
y: 0.8280
Epoch 31/35
 - 5s - loss: 0.1387 - accuracy: 0.9597 - val_loss: 1.0769 - val_accurac
y: 0.8221
Epoch 32/35
 - 5s - loss: 0.1196 - accuracy: 0.9621 - val_loss: 1.0596 - val_accurac
y: 0.8346
Epoch 33/35
 - 5s - loss: 0.0944 - accuracy: 0.9699 - val_loss: 1.0122 - val_accurac
y: 0.8476
Epoch 34/35
 - 5s - loss: 0.1280 - accuracy: 0.9642 - val_loss: 1.3175 - val_accurac
y: 0.8107
Epoch 35/35
 - 5s - loss: 0.1536 - accuracy: 0.9572 - val_loss: 0.9499 - val_accurac
y: 0.8378
Baseline Error: 16.22%
```

```
Out[23]: <keras.engine.sequential.Sequential at 0x7f98097ef0d0>
```

```
The best model remains the one with 2 layers and batch size 100
Could adding more layers help?
```

In [24]:

```
train_model(number_dense_layers=3, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=35, batch_size=100,
 units=2000)
```

```
Train on 18620 samples, validate on 2552 samples
Epoch 1/35
 - 24s - loss: 76.6539 - accuracy: 0.4060 - val_loss: 2.0274 - val_accura
cy: 0.5196
Epoch 2/35
 - 23s - loss: 1.1814 - accuracy: 0.6670 - val_loss: 1.4257 - val_accurac
y: 0.6399
Epoch 3/35
 - 26s - loss: 0.7189 - accuracy: 0.7777 - val_loss: 1.2475 - val_accurac
y: 0.6963
Epoch 4/35
 - 26s - loss: 0.4779 - accuracy: 0.8485 - val_loss: 1.1640 - val_accurac
y: 0.7183
Epoch 5/35
 - 25s - loss: 0.3239 - accuracy: 0.8917 - val_loss: 1.1322 - val_accurac
y: 0.7426
Epoch 6/35
 - 25s - loss: 0.2731 - accuracy: 0.9070 - val_loss: 1.1351 - val_accurac
y: 0.7586
Epoch 7/35
 - 23s - loss: 0.2297 - accuracy: 0.9241 - val_loss: 1.2070 - val_accurac
y: 0.7539
Epoch 8/35
 - 23s - loss: 0.2151 - accuracy: 0.9277 - val_loss: 1.1571 - val_accurac
y: 0.7798
Epoch 9/35
 - 26s - loss: 0.2649 - accuracy: 0.9170 - val_loss: 1.2437 - val_accurac
y: 0.7637
Epoch 10/35
 - 25s - loss: 0.3291 - accuracy: 0.9045 - val_loss: 1.1487 - val_accurac
y: 0.7763
Epoch 11/35
 - 25s - loss: 0.3129 - accuracy: 0.9062 - val_loss: 1.3616 - val_accurac
y: 0.7731
Epoch 12/35
 - 26s - loss: 0.2545 - accuracy: 0.9230 - val_loss: 1.4635 - val_accurac
y: 0.7672
Epoch 13/35
 - 24s - loss: 0.3164 - accuracy: 0.9131 - val_loss: 1.1724 - val_accurac
y: 0.7861
Epoch 14/35
 - 25s - loss: 0.2145 - accuracy: 0.9367 - val_loss: 1.0639 - val_accurac
y: 0.8201
Epoch 15/35
 - 26s - loss: 0.2471 - accuracy: 0.9308 - val_loss: 1.1629 - val_accurac
y: 0.8123
Epoch 16/35
 - 25s - loss: 0.2221 - accuracy: 0.9373 - val_loss: 1.3010 - val_accurac
y: 0.8041
Epoch 17/35
 - 25s - loss: 0.3314 - accuracy: 0.9134 - val_loss: 1.4322 - val_accurac
y: 0.7778
Epoch 18/35
```

```
 - 25s - loss: 0.2783 - accuracy: 0.9234 - val_loss: 1.3126 - val_accurac
y: 0.7876
Epoch 19/35
 - 26s - loss: 0.2070 - accuracy: 0.9412 - val_loss: 1.5889 - val_accurac
y: 0.7810
Epoch 20/35
 - 27s - loss: 0.2516 - accuracy: 0.9318 - val_loss: 1.1914 - val_accurac
y: 0.8111
Epoch 21/35
 - 26s - loss: 0.2074 - accuracy: 0.9424 - val_loss: 1.0575 - val_accurac
y: 0.8252
Epoch 22/35
 - 28s - loss: 0.1938 - accuracy: 0.9466 - val_loss: 1.3285 - val_accurac
y: 0.7990
Epoch 23/35
 - 26s - loss: 0.2396 - accuracy: 0.9359 - val_loss: 1.2530 - val_accurac
y: 0.8147
Epoch 24/35
 - 24s - loss: 0.2447 - accuracy: 0.9334 - val_loss: 1.3976 - val_accurac
y: 0.7923
Epoch 25/35
 - 23s - loss: 0.1591 - accuracy: 0.9532 - val_loss: 1.1413 - val_accurac
y: 0.8197
Epoch 26/35
 - 23s - loss: 0.1945 - accuracy: 0.9471 - val_loss: 1.1266 - val_accurac
y: 0.8186
Epoch 27/35
 - 24s - loss: 0.1773 - accuracy: 0.9492 - val_loss: 1.1547 - val_accurac
y: 0.8111
Epoch 28/35
 - 24s - loss: 0.1773 - accuracy: 0.9503 - val_loss: 1.0609 - val_accurac
y: 0.8245
Epoch 29/35
 - 23s - loss: 0.1776 - accuracy: 0.9528 - val_loss: 1.1838 - val_accurac
y: 0.8241
Epoch 30/35
 - 23s - loss: 0.1860 - accuracy: 0.9483 - val_loss: 1.1191 - val_accurac
y: 0.8229
Epoch 31/35
 - 23s - loss: 0.1738 - accuracy: 0.9507 - val_loss: 1.0110 - val_accurac
y: 0.8248
Epoch 32/35
 - 23s - loss: 0.2311 - accuracy: 0.9374 - val_loss: 1.1553 - val_accurac
y: 0.8237
Epoch 33/35
 - 23s - loss: 0.1761 - accuracy: 0.9505 - val_loss: 1.2413 - val_accurac
y: 0.8241
Epoch 34/35
 - 24s - loss: 0.2208 - accuracy: 0.9416 - val_loss: 0.9756 - val_accurac
y: 0.8346
Epoch 35/35
 - 23s - loss: 0.2263 - accuracy: 0.9402 - val_loss: 1.0766 - val_accurac
y: 0.8225
Baseline Error: 17.75%
```

```
Out[24]: <keras.engine.sequential.Sequential at 0x7f980bb39c90>
```

Speech conclusion

- The best speech model was a 2-layer model with 35 epochs. We probably need to introduce a type of early stopping where the number of epochs is halted after observing one or two decreases in validation accuracy.
- Although 1000 unit seems arbitray, neither decreasing, nor increase the number seemed to produce a more accurate model

Natural Language Processing: IMDb review sentiment analysis

In [8]:

```
from db_utils import get_imdb_dataset
(X_train, y_train), (X_test, y_test) = get_imdb_dataset()
```

In [88]:

```
train_model(number_dense_layers=1, X_train= X_train, y_train=y_train,
 X_test=X_test, y_test= y_test, epochs=10, batch_size=200,
 units=1000)
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/10
 - 239s - loss: 0.3478 - accuracy: 0.8502 - val_loss: 0.2992 - val_accura
cy: 0.8777
Epoch 2/10
 - 285s - loss: 0.0556 - accuracy: 0.9855 - val_loss: 0.3229 - val_accura
cy: 0.8769
Epoch 3/10
 - 235s - loss: 0.0129 - accuracy: 0.9986 - val_loss: 0.3615 - val_accura
cy: 0.8776
Epoch 4/10
 - 246s - loss: 0.0044 - accuracy: 0.9999 - val_loss: 0.3898 - val_accura
cy: 0.8775
Epoch 5/10
 - 256s - loss: 0.0024 - accuracy: 1.0000 - val_loss: 0.4160 - val_accura
cy: 0.8778
Epoch 6/10
 - 277s - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.4308 - val_accura
cy: 0.8780
Epoch 7/10
 - 292s - loss: 8.7237e-04 - accuracy: 1.0000 - val_loss: 0.4467 - val_ac
curacy: 0.8773
Epoch 8/10
 - 292s - loss: 6.2179e-04 - accuracy: 1.0000 - val_loss: 0.4601 - val_ac
curacy: 0.8772
Epoch 9/10
 - 239s - loss: 4.6525e-04 - accuracy: 1.0000 - val_loss: 0.4723 - val_ac
curacy: 0.8767
Epoch 10/10
 - 246s - loss: 3.5998e-04 - accuracy: 1.0000 - val_loss: 0.4832 - val_ac
curacy: 0.8764
Baseline Error: 12.36%
```

Out[88]:

```
<keras.engine.sequential.Sequential at 0x7f859df10410>
```

Conclusions

- Accuracy on the training set is 100% while accuracy on the validatsion set is only 87.7%
- It seems the model is overfitting. As such, longer training with an identical model doesn't seem reasonable.

```
In [9]: # adding an additional dense layer
        train_model(number_dense_layers=2, X_train= X_train, y_train=y_train,
         X_test=X_test, y_test= y_test, epochs=10, batch_size=200,
         units=1000)
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/10
 - 199s - loss: 0.4523 - accuracy: 0.8070 - val_loss: 0.2909 - val_accura
cy: 0.8794
Epoch 2/10
 - 193s - loss: 0.0651 - accuracy: 0.9795 - val_loss: 0.3443 - val_accura
cy: 0.8681
Epoch 3/10
 - 189s - loss: 0.0039 - accuracy: 0.9996 - val_loss: 0.4677 - val_accura
cy: 0.8736
Epoch 4/10
 - 191s - loss: 3.6917e-04 - accuracy: 1.0000 - val_loss: 0.5322 - val_ac
curacy: 0.8720
Epoch 5/10
 - 187s - loss: 7.5469e-04 - accuracy: 1.0000 - val_loss: 0.5352 - val_ac
curacy: 0.8756
Epoch 6/10
 - 178s - loss: 8.8936e-05 - accuracy: 1.0000 - val_loss: 0.5627 - val_ac
curacy: 0.8756
Epoch 7/10
 - 182s - loss: 5.6146e-05 - accuracy: 1.0000 - val_loss: 0.5849 - val_ac
curacy: 0.8756
Epoch 8/10

-------------------------------------------------------------------------
--
KeyboardInterrupt                          Traceback (most recent call las
t)
<ipython-input-9-4c52fc23b648> in <module>
      2 train_model(number_dense_layers=2, X_train= X_train, y_train=y_tr
ain,
      3  X_test=X_test, y_test= y_test, epochs=10, batch_size=200,
----> 4  units=1000)

<ipython-input-2-536cc44a0f99> in train_model(number_dense_layers, X_trai
n, y_train, X_test, y_test, epochs, batch_size, units)
     22                             y_train.shape[1], units)
     23     model.fit(X_train, y_train, validation_data=(X_test, y_test),
---> 24               epochs=epochs, batch_size=batch_size, verbose=2)
     25     scores = model.evaluate(X_test, y_test, verbose=2)
     26     print("Baseline Error: %.2f%%" % (100-scores[1]*100))

/opt/anaconda3/envs/deep37/lib/python3.7/site-packages/keras/engine/train
ing.py in fit(self, x, y, batch_size, epochs, verbose, callbacks, validat
ion_split, validation_data, shuffle, class_weight, sample_weight, initial
_epoch, steps_per_epoch, validation_steps, validation_freq, max_queue_siz
e, workers, use_multiprocessing, **kwargs)
   1237                                         steps_per_epoch=steps_per
_epoch,
   1238                                         validation_steps=validati
on_steps,
-> 1239                                         validation_freq=validatio
```

```
n_freq)
   1240
   1241     def evaluate(self,

/opt/anaconda3/envs/deep37/lib/python3.7/site-packages/keras/engine/train
ing_arrays.py in fit_loop(model, fit_function, fit_inputs, out_labels, ba
tch_size, epochs, verbose, callbacks, val_function, val_inputs, shuffle,
 initial_epoch, steps_per_epoch, validation_steps, validation_freq)
    194                         ins_batch[i] = ins_batch[i].toarray()
    195
--> 196                     outs = fit_function(ins_batch)
    197                     outs = to_list(outs)
    198                     for l, o in zip(out_labels, outs):

/opt/anaconda3/envs/deep37/lib/python3.7/site-packages/tensorflow_core/py
thon/keras/backend.py in __call__(self, inputs)
   3738         value = math_ops.cast(value, tensor.dtype)
   3739       converted_inputs.append(value)
-> 3740     outputs = self._graph_fn(*converted_inputs)
   3741
   3742     # EagerTensor.numpy() will often make a copy to ensure memory
safety.

/opt/anaconda3/envs/deep37/lib/python3.7/site-packages/tensorflow_core/py
thon/eager/function.py in __call__(self, *args, **kwargs)
   1079       TypeError: For invalid positional/keyword argument combinat
ions.
   1080     """
-> 1081     return self._call_impl(args, kwargs)
   1082
   1083   def _call_impl(self, args, kwargs, cancellation_manager=None):

/opt/anaconda3/envs/deep37/lib/python3.7/site-packages/tensorflow_core/py
thon/eager/function.py in _call_impl(self, args, kwargs, cancellation_man
ager)
   1119       raise TypeError("Keyword arguments {} unknown. Expected
 {}.".format(
   1120           list(kwargs.keys()), list(self._arg_keywords)))
-> 1121     return self._call_flat(args, self.captured_inputs, cancellati
on_manager)
   1122
   1123   def _filtered_call(self, args, kwargs):

/opt/anaconda3/envs/deep37/lib/python3.7/site-packages/tensorflow_core/py
thon/eager/function.py in _call_flat(self, args, captured_inputs, cancell
ation_manager)
   1222     if executing_eagerly:
   1223       flat_outputs = forward_function.call(
-> 1224           ctx, args, cancellation_manager=cancellation_manager)
   1225     else:
   1226       gradient_name = self._delayed_rewrite_functions.register()

/opt/anaconda3/envs/deep37/lib/python3.7/site-packages/tensorflow_core/py
thon/eager/function.py in call(self, ctx, args, cancellation_manager)
    509               inputs=args,
    510               attrs=("executor_type", executor_type, "config_prot
o", config),
--> 511                 ctx=ctx)
    512           else:
    513             outputs = execute.execute_with_cancellation(

/opt/anaconda3/envs/deep37/lib/python3.7/site-packages/tensorflow_core/py
thon/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attr
s, ctx, name)
     59     tensors = pywrap_tensorflow.TFE_Py_Execute(ctx._handle, devic
e_name,
     60                                                op_name, inputs, a
ttrs,
---> 61                                                num_outputs)
     62   except core._NotOkStatusException as e:
     63     if name is not None:

KeyboardInterrupt: 
```

Conclusions on Natural Language Processing

- Accuracy on the training set is 100% while accuracy on the validatsion set is only 87.7%
- It seems the model is overfitting. As such, longer training with an identical model doesn't seem reasonable.
- We therefore halted training.

Overall conclusions

- We ran an identical set of functions on a vision, speech and NLP model.
- The fact that a couple dozen lines of code can produce a model across a variety of use cases is pretty amazing
- Optimizing hyperparameters seems trial and error so far. Increasing the number of layers did improve speech but not vision. Increasing epochs seemed to have a positive effect on both vision and speech. But NLP seems to need a different model spec in order to improve performance.
- Training time can go up significantly, even with these small datasets. We tried the vision model on google collab, and training times decreased materially.