Creating a Basic Speech Recognition System

Phillip Spratling

The Problem

- Creating a neural network to classify speech input as one of 10 command words, silence, or unknown
- Using TensorFlow Speech Commands Datasets – includes 65,000 one-second long utterances of 30 short words by thousands of different people

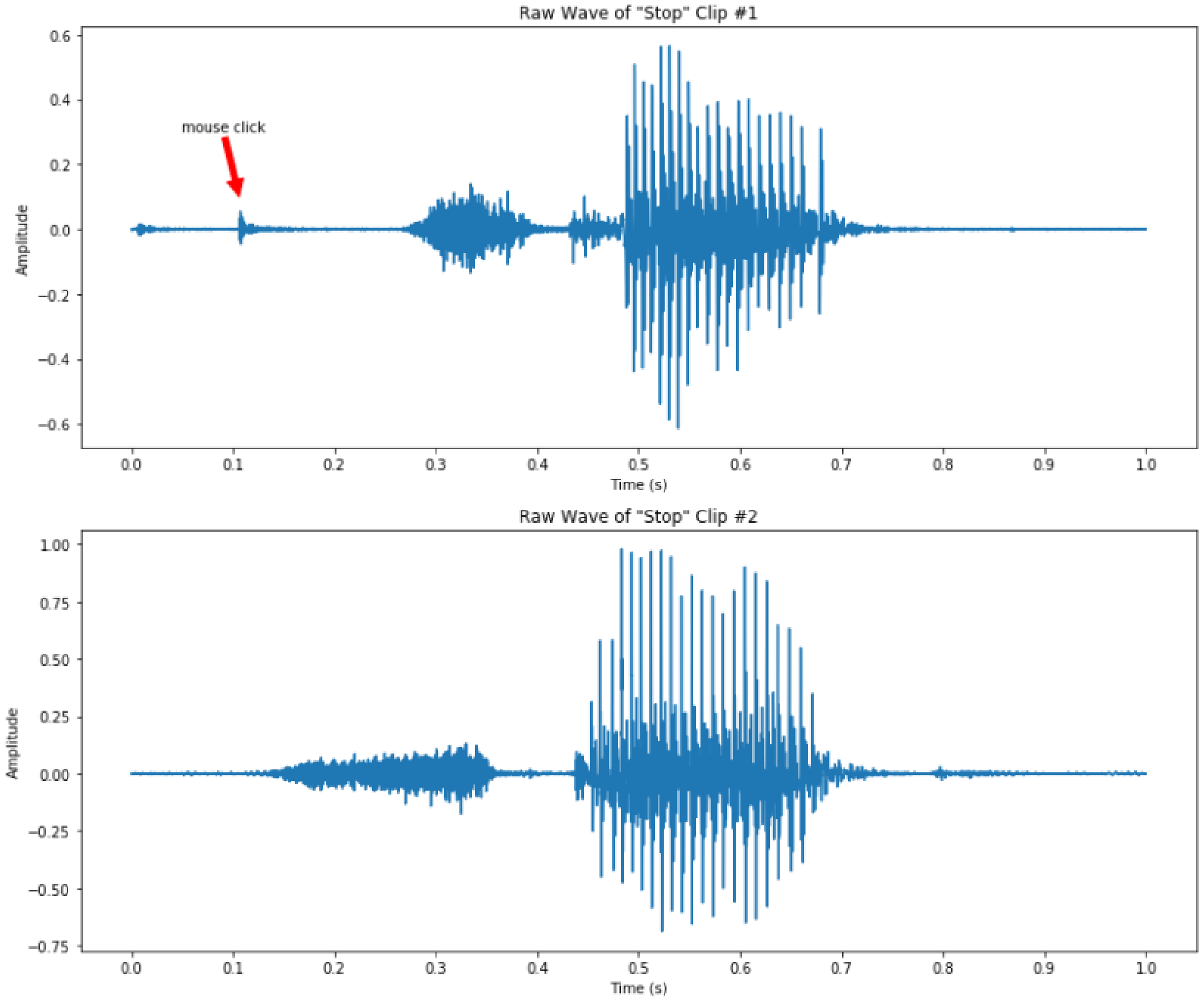
EDA

-Data comes in WAV files and needs to be preprocessed before feeding into neural networks

- Multiple ways of accomplishing this

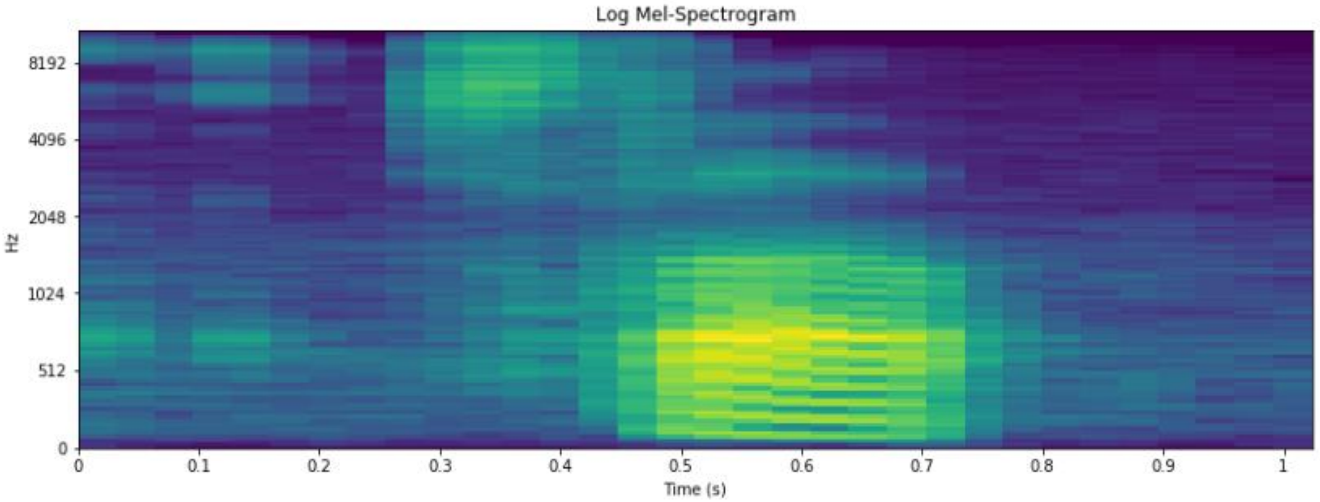
Raw Wave Form

- One sample is a one-dimensional vector with each data point indicating an amplitude at a certain point in time
- Amplitude is unitless when extracted by Librosa



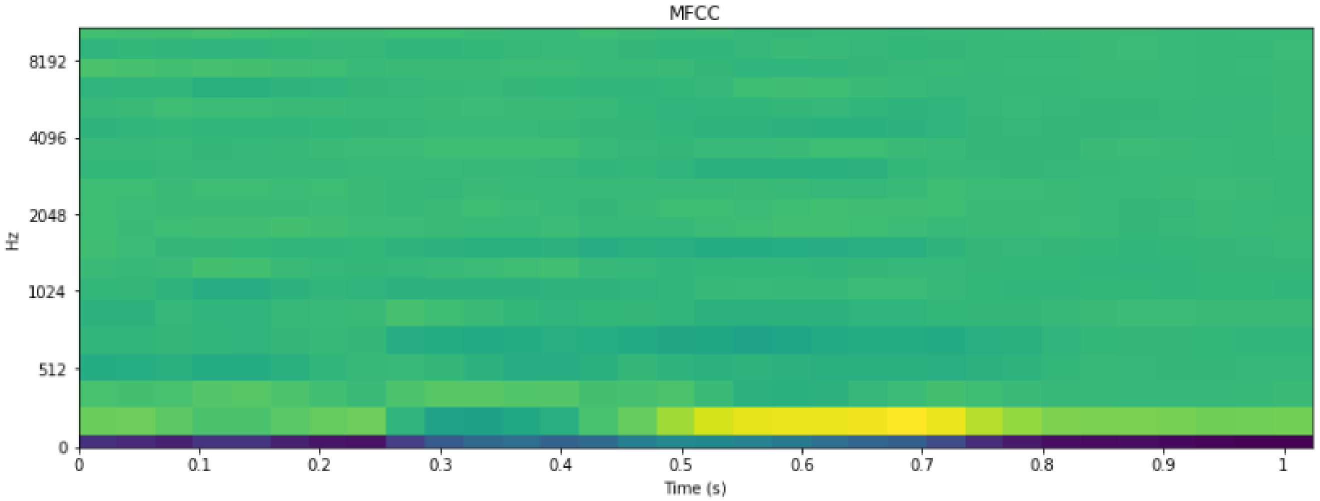
Log-Mel Spectrogram

- adds a dimension of complexity to the raw wave forms
- 2 dimensional array of amplitudes at different frequencies at different points of time



Mel-Frequency Ceptsral Coefficients (MFCCs)

- designed to extract information about linguistic content from audio while ignoring background noise
- Used in most speech recognition systems
- We’ll use these to get the best results



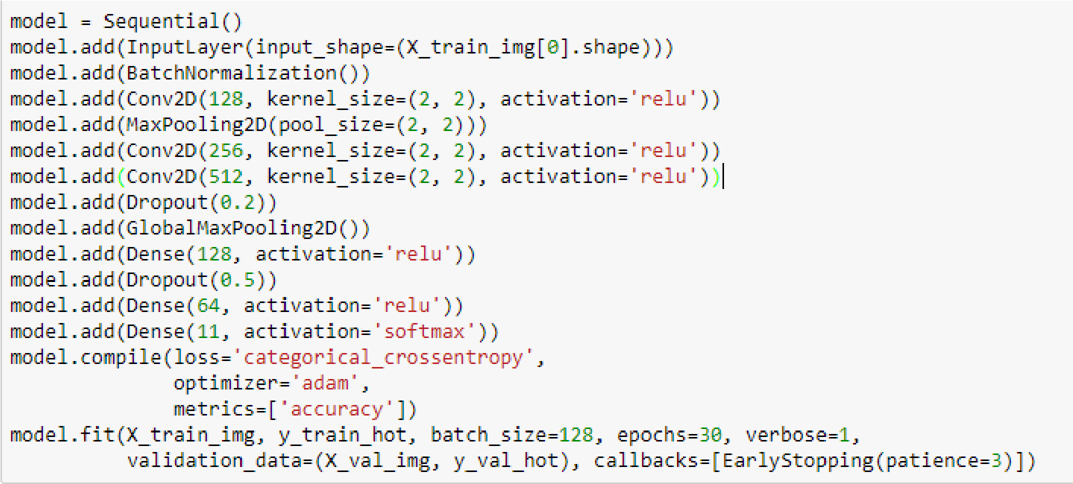
Designing the Neural Network

- MFCCs allow us to treat each data sample as an image

- Convolutional Neural Networks (CNN) tend to work best for image data

Convolutional Neural Network

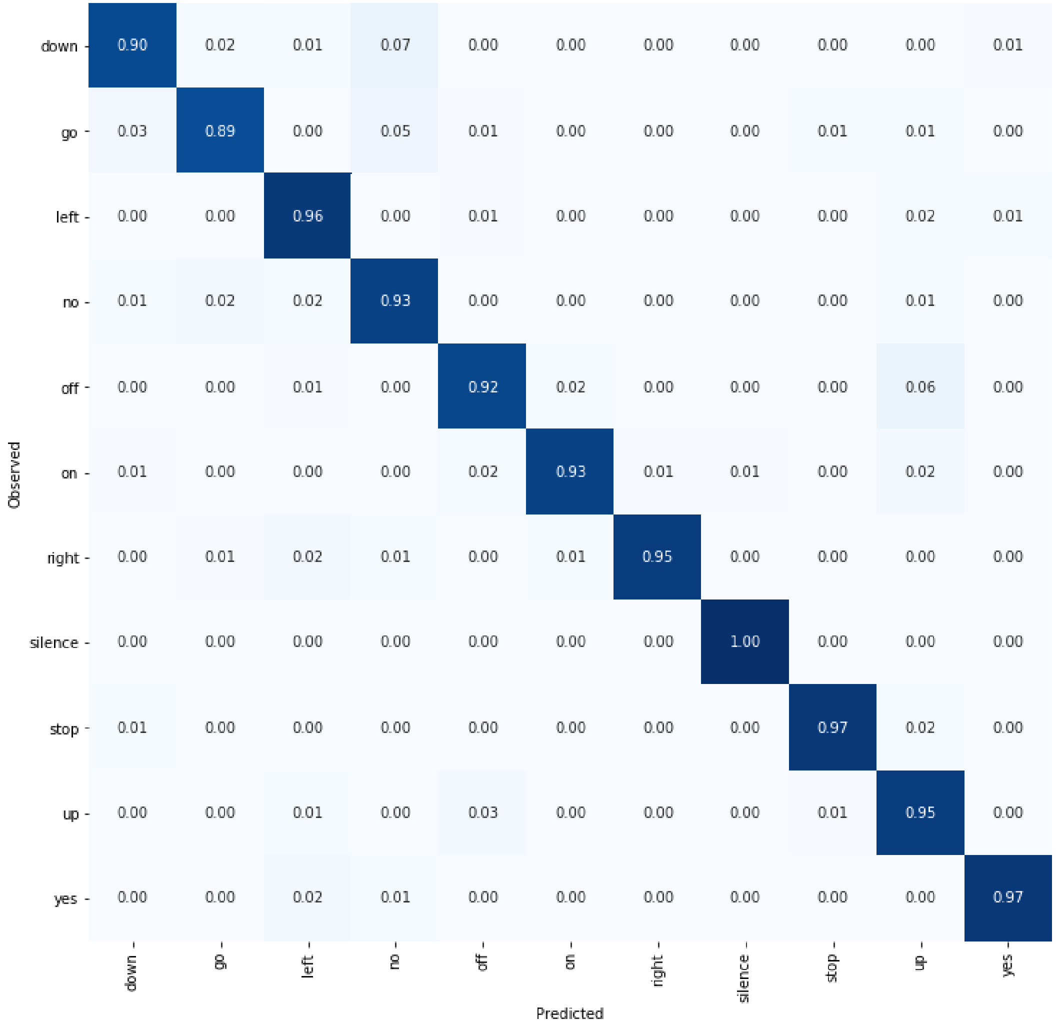
- A convolutional neural network was trained with the 10 command words and extracted silence
- Obtained accuracy of about 94%
- Worked best out of any other models trained



```
model = Sequential()
model.add(InputLayer(input_shape=(X_train_img[0].shape)))
model.add(BatchNormalization())
model.add(Conv2D(128, kernel_size=(2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(256, kernel_size=(2, 2), activation='relu'))
model.add(Conv2D(512, kernel_size=(2, 2), activation='relu'))
model.add(Dropout(0.2))
model.add(GlobalMaxPooling2D())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dense(11, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.fit(X_train_img, y_train_hot, batch_size=128, epochs=30, verbose=1,
          validation_data=(X_val_img, y_val_hot), callbacks=[EarlyStopping(patience=3)])
```

Confusion Matrix

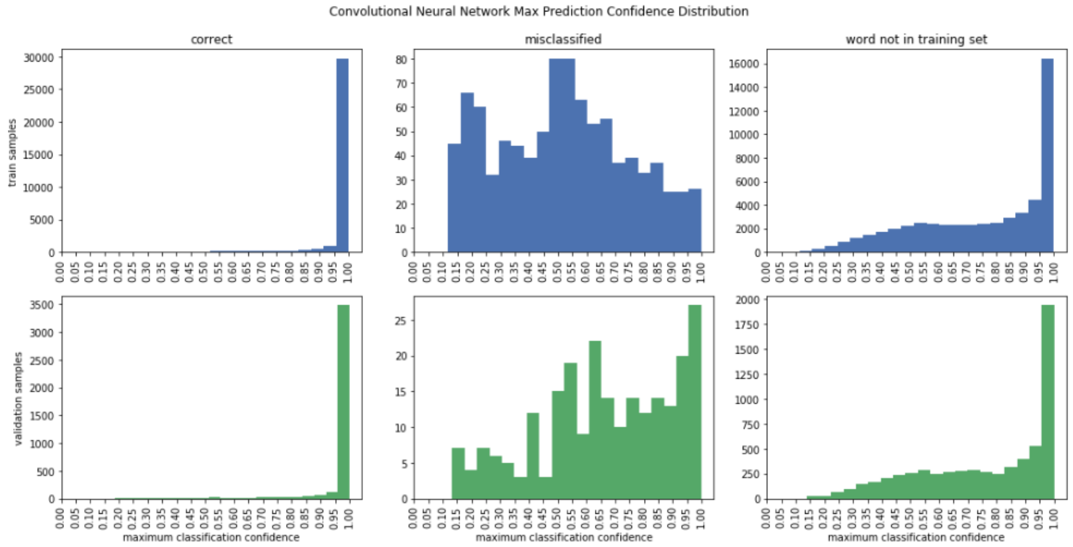
- Model still tends to confuse similar words
- Overall performs well

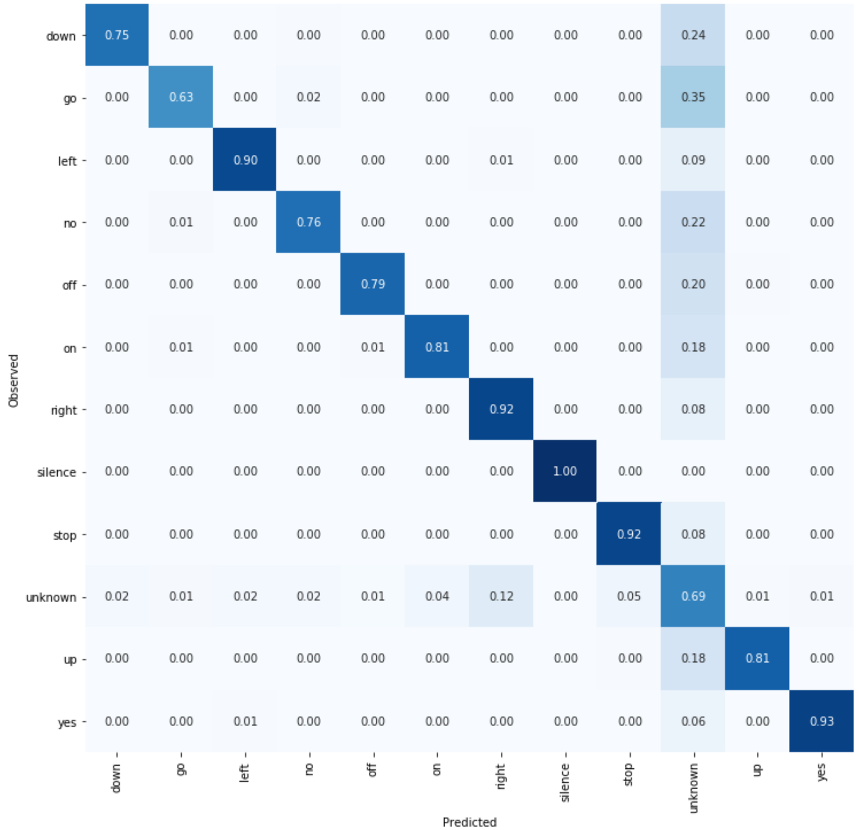


Observed \ Predicted	down	go	left	no	off	on	right	silence	stop	up	yes
down	0.90	0.02	0.01	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.01
go	0.03	0.89	0.00	0.05	0.01	0.00	0.00	0.00	0.01	0.01	0.00
left	0.00	0.00	0.96	0.00	0.01	0.00	0.00	0.00	0.00	0.02	0.01
no	0.01	0.02	0.02	0.93	0.00	0.00	0.00	0.00	0.00	0.01	0.00
off	0.00	0.00	0.01	0.00	0.92	0.02	0.00	0.00	0.00	0.06	0.00
on	0.01	0.00	0.00	0.00	0.02	0.93	0.01	0.01	0.00	0.02	0.00
right	0.00	0.01	0.02	0.01	0.00	0.01	0.95	0.00	0.00	0.00	0.00
silence	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
stop	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.02	0.00
up	0.00	0.00	0.01	0.00	0.03	0.00	0.00	0.00	0.01	0.95	0.00
yes	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.97

Adding Unknown Detection

- Maximum confidences evaluated for entire training and validation sets
- Threshold of .95 established to classify anything below as “unknown”
- Overall accuracy drops to ~74% due to 10x as many unknown words
- Overall good results on individual words while minimizing false positives (initial goal)





Observed \ Predicted	down	go	left	no	off	on	right	silence	stop	unknown	up	yes
down	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.00	0.00
go	0.00	0.63	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.35	0.00	0.00
left	0.00	0.00	0.90	0.00	0.00	0.00	0.01	0.00	0.00	0.09	0.00	0.00
no	0.00	0.01	0.00	0.76	0.00	0.00	0.00	0.00	0.00	0.22	0.00	0.00
off	0.00	0.00	0.00	0.00	0.79	0.00	0.00	0.00	0.00	0.20	0.00	0.00
on	0.00	0.01	0.00	0.00	0.01	0.81	0.00	0.00	0.00	0.18	0.00	0.00
right	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.00	0.00	0.08	0.00	0.00
silence	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
stop	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.08	0.00	0.00
unknown	0.02	0.01	0.02	0.02	0.01	0.04	0.12	0.00	0.05	0.69	0.01	0.01
up	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18	0.81	0.00
yes	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.93

Fine Tuning Model Performance

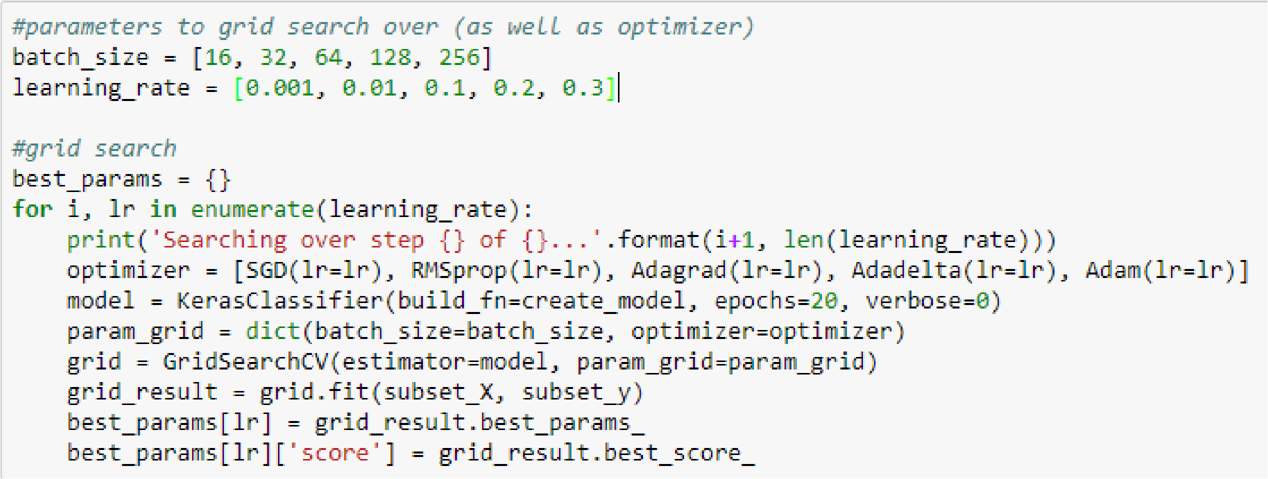
- Now that we have a working model, we can improve it to get the best possible results

Adding Augmented Data

- Added samples to training set of randomly sampled clips with background noise added in
- Focused on words the model had trouble predicting

Grid Search

- Selected random subset (~20%) of training data to perform grid search on to save computing time
- Performed a grid search in two stages:
  - Searched over optimizer, batch size, and learning rate
  - Searched over dropout percentages for each dropout layer in network

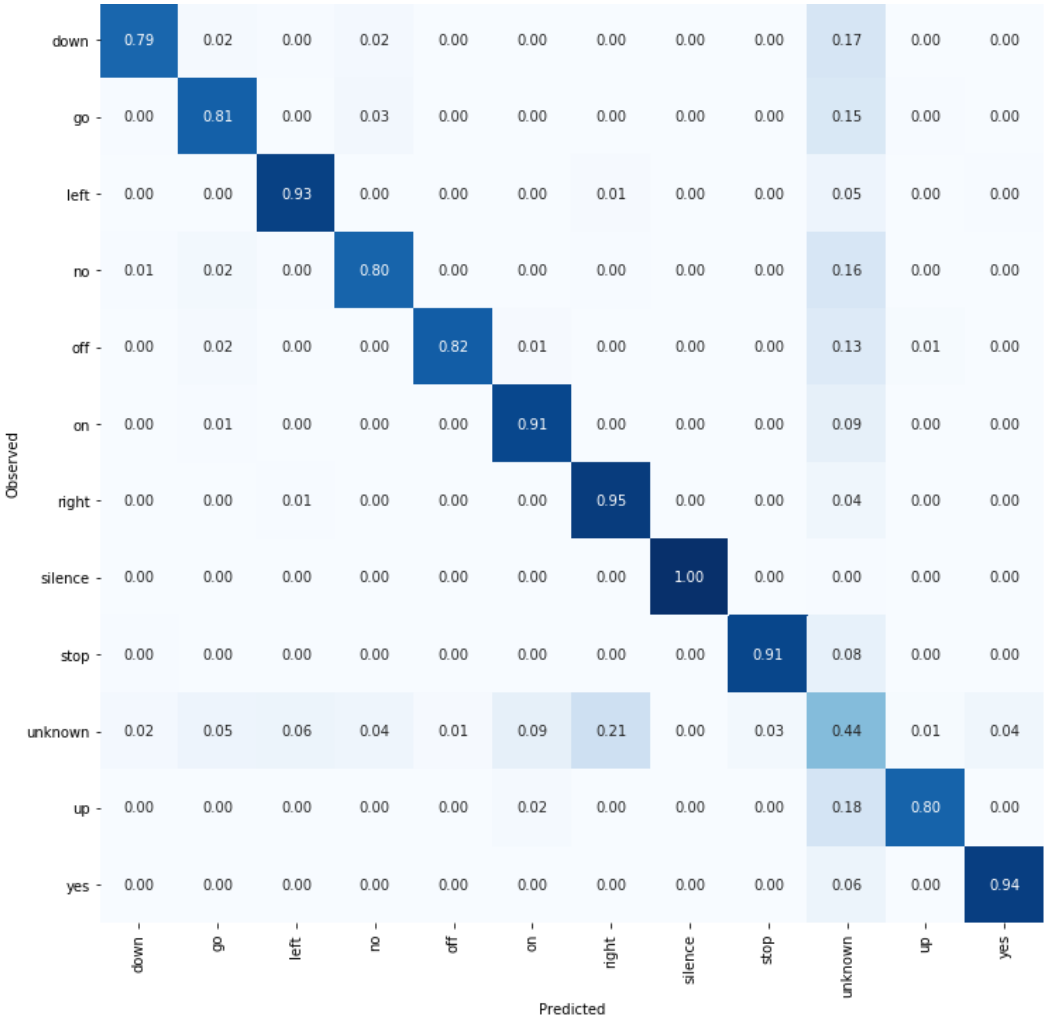


```
#parameters to grid search over (as well as optimizer)
batch_size = [16, 32, 64, 128, 256]
learning_rate = [0.001, 0.01, 0.1, 0.2, 0.3]

#grid search
best_params = {}
for i, lr in enumerate(learning_rate):
    print('Searching over step {} of {}...'.format(i+1, len(learning_rate)))
    optimizer = [SGD(lr=lr), RMSprop(lr=lr), Adagrad(lr=lr), Adadelta(lr=lr), Adam(lr=lr)]
    model = KerasClassifier(build_fn=create_model, epochs=20, verbose=0)
    param_grid = dict(batch_size=batch_size, optimizer=optimizer)
    grid = GridSearchCV(estimator=model, param_grid=param_grid)
    grid_result = grid.fit(subset_X, subset_y)
    best_params[lr] = grid_result.best_params_
    best_params[lr]['score'] = grid_result.best_score_
```

Results

- Took best hyperparameters from grid search and trained on entire dataset, including augmented data
- Accuracy dropped to 64%
- Likely due to only being able to train on a subset of the training data



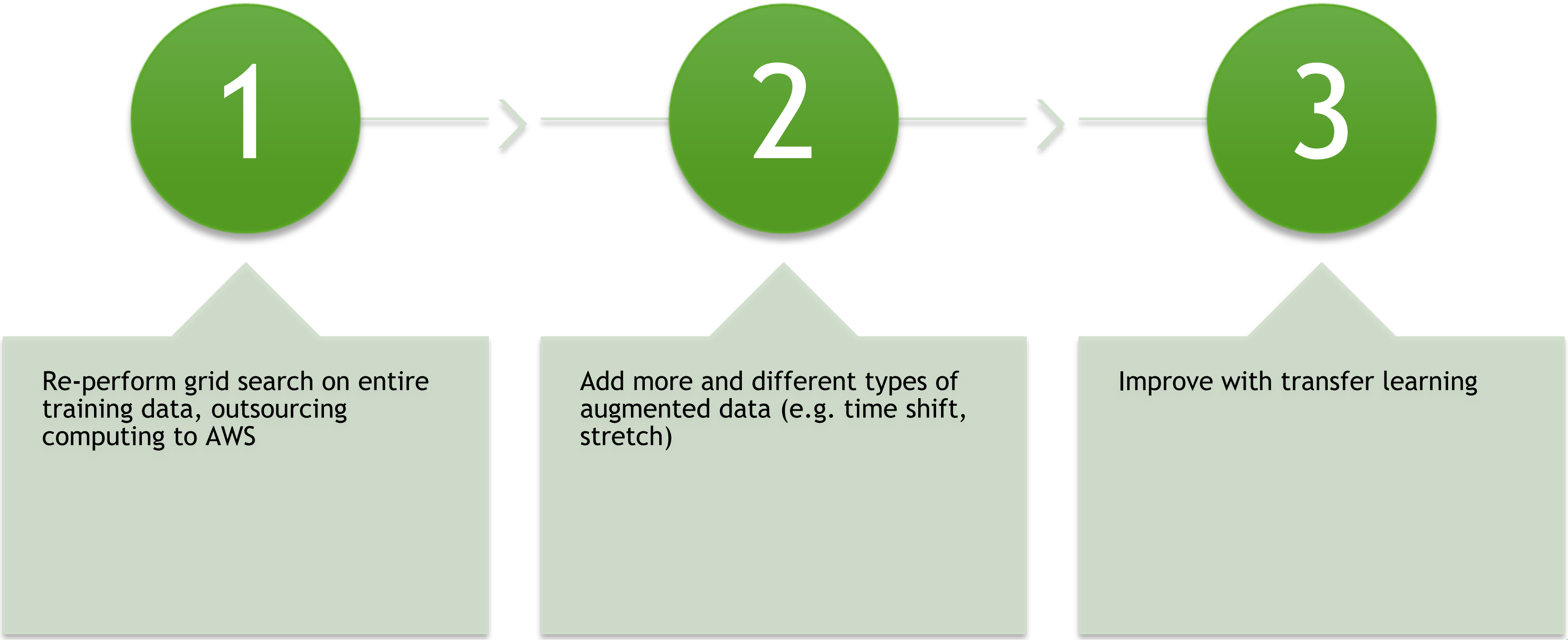
Observed \ Predicted	down	go	left	no	off	on	right	silence	stop	unknown	up	yes
down	0.79	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.00
go	0.00	0.81	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.00
left	0.00	0.00	0.93	0.00	0.00	0.00	0.01	0.00	0.00	0.05	0.00	0.00
no	0.01	0.02	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.00
off	0.00	0.02	0.00	0.00	0.82	0.01	0.00	0.00	0.00	0.13	0.01	0.00
on	0.00	0.01	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.09	0.00	0.00
right	0.00	0.00	0.01	0.00	0.00	0.00	0.95	0.00	0.00	0.04	0.00	0.00
silence	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
stop	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.91	0.08	0.00	0.00
unknown	0.02	0.05	0.06	0.04	0.01	0.09	0.21	0.00	0.03	0.44	0.01	0.04
up	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.18	0.80	0.00
yes	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.94

Conclusion

We have built a functional speech recognition system for a simple speech interface

Could be used as the foundation of a more sophisticated speech system

Next Steps



1

Re-perform grid search on entire training data, outsourcing computing to AWS

2

Add more and different types of augmented data (e.g. time shift, stretch)

3

Improve with transfer learning