In [1]:

```
import os
%matplotlib inline
# Prevent CUDA from using GPU as it does not work well on my pc
#os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
H5_OUPUT_FILENAME = "test_epoch.h5"
IMPORT_DATA_FILENAME = 'data_train.npy'
IMPORT_LABELS_FILENAME = 'labels_train.npy'

# Set Constants of the model
BATCH_SIZE = 64
```

In [2]:

```
# Helper functions
import numpy as np

# Breaks down a list of integer values into a one-hot like format
def one_hot_training(np_array):
    transformed_list = []
    for arr in np_array:
        new_arr = np.zeros(10)
        new_arr[int(arr)] = 1
        transformed_list.append(new_arr)
    return np.array(transformed_list)

# This translates the highest value from the one-hot encoding into the correct sign nam
def one_hot_translator(np_array):
    labels_names = ['Stop','Yield','Red Light','Green Light','Roundabout','Right Turn O
                    'Do Not Enter','Crosswalk','Handicap Parking','No Parking']
    #return labels_names[np.argmax(np_array)]
    return np.argmax(np_array)

# This translates the highest value from the one-hot encoding into the correct sign nam
def one_hot_translator_thres(np_array):
    max_index = np.argmax(np_array)
    max_value = np.max(np_array)
    if max_value <= 0.8071025020177562:
        return -1
    return max_index

# This translates an entire array of one-hot encoded sign predictions
def translate_all(np_array):
    translated_values = []
    for i in np_array:
        translated_values.append(one_hot_translator(i))
    return np.array(translated_values)

# This translates an entire array of one-hot encoded sign predictions
def translate_all_thres(np_array):
    translated_values = []
    for i in np_array:
        translated_values.append(one_hot_translator_thres(i))
    return np.array(translated_values)
```

In [3]:

```
# First import the data
import tensorflow as tf
data_train = np.load(IMPORT_DATA_FILENAME).transpose()
```

```
labels_train = np.load(IMPORT_LABELS_FILENAME)
data_train = np.array([i.reshape(300,300,3) for i in data_train])
data_train = np.array(tf.cast(tf.image.resize(data_train,(150,150)), np.uint8))
```

In [4]:

```
# Process the data so that it is in the expected form for the InceptionV3 model
import tensorflow as tf
processed = tf.keras.applications.inception_v3.preprocess_input(data_train, data_format

# Break down data into training and test sets
from sklearn.model_selection import train_test_split
x_train, x_test, t_train, t_test = train_test_split(processed, one_hot_training(labels_
```

In [5]:

```
# Augment data to reduce overfitting
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(horizontal_flip=True,
                                   vertical_flip=True,
                                   rotation_range=90,
                                   brightness_range=(.75, 1))

train_generator = train_datagen.flow(
    x_train,
    y = t_train,
    batch_size=BATCH_SIZE)
```

In [6]:

```
# Import the InceptionV3 Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
inception = InceptionV3(input_shape=(150,150,3),
                        include_top=False,
                        weights='imagenet')

# Set layers to false to prevent overwriting the existing model
for layer in inception.layers:
    layer.trainable = False

# Create output layers that will be trained
from tensorflow.keras.optimizers import SGD
x = tf.keras.layers.Flatten()(inception.output)
x = tf.keras.layers.Dense(1024, activation="relu")(x)
x = tf.keras.layers.Dropout(0.15)(x)
x = tf.keras.layers.Dense(10, activation='softmax')(x)

# Create Optimizer
Adam = tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=
Nadam = tf.keras.optimizers.Nadam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilo
SGD = SGD(learning_rate=0.01, nesterov=True)
optimizer = SGD

# Finalize and compile the model
model = tf.keras.Model(inception.input, outputs = x)
model.compile(optimizer = optimizer,
             loss = 'categorical_crossentropy',
             metrics = ['categorical_accuracy', 'acc','mean_squared_error'])
```

In [8]:

```
# Fit the model to the dataset
```

```
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=H5_OUPUT_FILENAME,monitor='los
es = tf.keras.callbacks.EarlyStopping(monitor='loss', mode='min', verbose=1, patience=1
history = model.fit(train_generator, epochs=200, callbacks=[es, checkpoint])
#model.save(H5_OUPUT_FILENAME)
```

```
Epoch 1/200
78/78 [==============================] - 30s 293ms/step - loss: 1.8864 - categorical_acc
uracy: 0.7070 - acc: 0.7070 - mean_squared_error: 0.0442
Epoch 2/200
78/78 [==============================] - 23s 288ms/step - loss: 0.4418 - categorical_acc
uracy: 0.8697 - acc: 0.8697 - mean_squared_error: 0.0193
Epoch 3/200
78/78 [==============================] - 22s 277ms/step - loss: 0.3857 - categorical_acc
uracy: 0.8884 - acc: 0.8884 - mean_squared_error: 0.0168
Epoch 4/200
78/78 [==============================] - 23s 291ms/step - loss: 0.3428 - categorical_acc
uracy: 0.9003 - acc: 0.9003 - mean_squared_error: 0.0151
Epoch 5/200
78/78 [==============================] - 23s 294ms/step - loss: 0.2979 - categorical_acc
uracy: 0.9173 - acc: 0.9173 - mean_squared_error: 0.0128
Epoch 6/200
78/78 [==============================] - 22s 281ms/step - loss: 0.2935 - categorical_acc
uracy: 0.9118 - acc: 0.9118 - mean_squared_error: 0.0130
Epoch 7/200
78/78 [==============================] - 22s 285ms/step - loss: 0.2700 - categorical_acc
uracy: 0.9229 - acc: 0.9229 - mean_squared_error: 0.0115
Epoch 8/200
78/78 [==============================] - 22s 287ms/step - loss: 0.2537 - categorical_acc
uracy: 0.9255 - acc: 0.9255 - mean_squared_error: 0.0112
Epoch 9/200
78/78 [==============================] - 22s 279ms/step - loss: 0.2439 - categorical_acc
uracy: 0.9306 - acc: 0.9306 - mean_squared_error: 0.0107
Epoch 10/200
78/78 [==============================] - 22s 278ms/step - loss: 0.2329 - categorical_acc
uracy: 0.9360 - acc: 0.9360 - mean_squared_error: 0.0099
Epoch 11/200
78/78 [==============================] - 22s 280ms/step - loss: 0.2257 - categorical_acc
uracy: 0.9362 - acc: 0.9362 - mean_squared_error: 0.0099
Epoch 12/200
78/78 [==============================] - 22s 276ms/step - loss: 0.2283 - categorical_acc
uracy: 0.9372 - acc: 0.9372 - mean_squared_error: 0.0097
Epoch 13/200
78/78 [==============================] - 22s 279ms/step - loss: 0.2091 - categorical_acc
uracy: 0.9429 - acc: 0.9429 - mean_squared_error: 0.0091
Epoch 14/200
78/78 [==============================] - 22s 280ms/step - loss: 0.2014 - categorical_acc
uracy: 0.9445 - acc: 0.9445 - mean_squared_error: 0.0087
Epoch 15/200
78/78 [==============================] - 22s 286ms/step - loss: 0.1989 - categorical_acc
uracy: 0.9415 - acc: 0.9415 - mean_squared_error: 0.0087
Epoch 16/200
78/78 [==============================] - 22s 276ms/step - loss: 0.1949 - categorical_acc
uracy: 0.9471 - acc: 0.9471 - mean_squared_error: 0.0085
Epoch 17/200
78/78 [==============================] - 22s 284ms/step - loss: 0.1879 - categorical_acc
uracy: 0.9453 - acc: 0.9453 - mean_squared_error: 0.0083
Epoch 18/200
78/78 [==============================] - 22s 278ms/step - loss: 0.1913 - categorical_acc
uracy: 0.9445 - acc: 0.9445 - mean_squared_error: 0.0085
Epoch 19/200
```

```
78/78 [==============================] - 22s 275ms/step - loss: 0.1752 - categorical_acc
uracy: 0.9512 - acc: 0.9512 - mean_squared_error: 0.0077
Epoch 20/200
78/78 [==============================] - 22s 277ms/step - loss: 0.1741 - categorical_acc
uracy: 0.9530 - acc: 0.9530 - mean_squared_error: 0.0077
Epoch 21/200
78/78 [==============================] - 22s 278ms/step - loss: 0.1592 - categorical_acc
uracy: 0.9536 - acc: 0.9536 - mean_squared_error: 0.0072
Epoch 22/200
78/78 [==============================] - 22s 278ms/step - loss: 0.1689 - categorical_acc
uracy: 0.9520 - acc: 0.9520 - mean_squared_error: 0.0075
Epoch 23/200
78/78 [==============================] - 21s 273ms/step - loss: 0.1625 - categorical_acc
uracy: 0.9530 - acc: 0.9530 - mean_squared_error: 0.0073
Epoch 24/200
78/78 [==============================] - 22s 275ms/step - loss: 0.1572 - categorical_acc
uracy: 0.9562 - acc: 0.9562 - mean_squared_error: 0.0072
Epoch 25/200
78/78 [==============================] - 22s 275ms/step - loss: 0.1519 - categorical_acc
uracy: 0.9558 - acc: 0.9558 - mean_squared_error: 0.0069
Epoch 26/200
78/78 [==============================] - 22s 275ms/step - loss: 0.1424 - categorical_acc
uracy: 0.9586 - acc: 0.9586 - mean_squared_error: 0.0064
Epoch 27/200
78/78 [==============================] - 22s 275ms/step - loss: 0.1372 - categorical_acc
uracy: 0.9625 - acc: 0.9625 - mean_squared_error: 0.0062
Epoch 28/200
78/78 [==============================] - 22s 275ms/step - loss: 0.1498 - categorical_acc
uracy: 0.9534 - acc: 0.9534 - mean_squared_error: 0.0068
Epoch 29/200
78/78 [==============================] - 21s 273ms/step - loss: 0.1244 - categorical_acc
uracy: 0.9639 - acc: 0.9639 - mean_squared_error: 0.0056
Epoch 30/200
78/78 [==============================] - 21s 273ms/step - loss: 0.1435 - categorical_acc
uracy: 0.9588 - acc: 0.9588 - mean_squared_error: 0.0066
Epoch 31/200
78/78 [==============================] - 21s 274ms/step - loss: 0.1362 - categorical_acc
uracy: 0.9609 - acc: 0.9609 - mean_squared_error: 0.0061
Epoch 32/200
78/78 [==============================] - 22s 275ms/step - loss: 0.1287 - categorical_acc
uracy: 0.9655 - acc: 0.9655 - mean_squared_error: 0.0058
Epoch 33/200
78/78 [==============================] - 22s 278ms/step - loss: 0.1241 - categorical_acc
uracy: 0.9641 - acc: 0.9641 - mean_squared_error: 0.0057
Epoch 34/200
78/78 [==============================] - 22s 279ms/step - loss: 0.1260 - categorical_acc
uracy: 0.9613 - acc: 0.9613 - mean_squared_error: 0.0058
Epoch 35/200
78/78 [==============================] - 22s 283ms/step - loss: 0.1355 - categorical_acc
uracy: 0.9568 - acc: 0.9568 - mean_squared_error: 0.0064
Epoch 36/200
78/78 [==============================] - 22s 278ms/step - loss: 0.1231 - categorical_acc
uracy: 0.9641 - acc: 0.9641 - mean_squared_error: 0.0057
Epoch 37/200
78/78 [==============================] - 22s 281ms/step - loss: 0.1230 - categorical_acc
uracy: 0.9637 - acc: 0.9637 - mean_squared_error: 0.0056
Epoch 38/200
78/78 [==============================] - 22s 281ms/step - loss: 0.1240 - categorical_acc
uracy: 0.9639 - acc: 0.9639 - mean_squared_error: 0.0058
Epoch 39/200
```

```
78/78 [==============================] - 21s 273ms/step - loss: 0.1125 - categorical_acc
uracy: 0.9653 - acc: 0.9653 - mean_squared_error: 0.0053
Epoch 40/200
78/78 [==============================] - 22s 275ms/step - loss: 0.1084 - categorical_acc
uracy: 0.9665 - acc: 0.9665 - mean_squared_error: 0.0051
Epoch 41/200
78/78 [==============================] - 21s 274ms/step - loss: 0.1165 - categorical_acc
uracy: 0.9673 - acc: 0.9673 - mean_squared_error: 0.0054
Epoch 42/200
78/78 [==============================] - 22s 284ms/step - loss: 0.1083 - categorical_acc
uracy: 0.9705 - acc: 0.9705 - mean_squared_error: 0.0049
Epoch 43/200
78/78 [==============================] - 22s 278ms/step - loss: 0.1093 - categorical_acc
uracy: 0.9647 - acc: 0.9647 - mean_squared_error: 0.0052
Epoch 44/200
78/78 [==============================] - 22s 279ms/step - loss: 0.1076 - categorical_acc
uracy: 0.9673 - acc: 0.9673 - mean_squared_error: 0.0049
Epoch 45/200
78/78 [==============================] - 22s 276ms/step - loss: 0.1014 - categorical_acc
uracy: 0.9695 - acc: 0.9695 - mean_squared_error: 0.0048
Epoch 46/200
78/78 [==============================] - 22s 276ms/step - loss: 0.1006 - categorical_acc
uracy: 0.9715 - acc: 0.9715 - mean_squared_error: 0.0047
Epoch 47/200
78/78 [==============================] - 22s 277ms/step - loss: 0.1024 - categorical_acc
uracy: 0.9681 - acc: 0.9681 - mean_squared_error: 0.0048
Epoch 48/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0899 - categorical_acc
uracy: 0.9720 - acc: 0.9720 - mean_squared_error: 0.0042
Epoch 49/200
78/78 [==============================] - 22s 284ms/step - loss: 0.0999 - categorical_acc
uracy: 0.9693 - acc: 0.9693 - mean_squared_error: 0.0047
Epoch 50/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0990 - categorical_acc
uracy: 0.9685 - acc: 0.9685 - mean_squared_error: 0.0047
Epoch 51/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0878 - categorical_acc
uracy: 0.9732 - acc: 0.9732 - mean_squared_error: 0.0042
Epoch 52/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0885 - categorical_acc
uracy: 0.9736 - acc: 0.9736 - mean_squared_error: 0.0042
Epoch 53/200
78/78 [==============================] - 22s 280ms/step - loss: 0.0907 - categorical_acc
uracy: 0.9726 - acc: 0.9726 - mean_squared_error: 0.0043
Epoch 54/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0882 - categorical_acc
uracy: 0.9738 - acc: 0.9738 - mean_squared_error: 0.0042
Epoch 55/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0796 - categorical_acc
uracy: 0.9768 - acc: 0.9768 - mean_squared_error: 0.0037
Epoch 56/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0839 - categorical_acc
uracy: 0.9738 - acc: 0.9738 - mean_squared_error: 0.0040
Epoch 57/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0861 - categorical_acc
uracy: 0.9722 - acc: 0.9722 - mean_squared_error: 0.0042
Epoch 58/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0918 - categorical_acc
uracy: 0.9734 - acc: 0.9734 - mean_squared_error: 0.0043
Epoch 59/200
```

```
78/78 [==============================] - 22s 277ms/step - loss: 0.0746 - categorical_acc
uracy: 0.9778 - acc: 0.9778 - mean_squared_error: 0.0036
Epoch 60/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0782 - categorical_acc
uracy: 0.9770 - acc: 0.9770 - mean_squared_error: 0.0037
Epoch 61/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0892 - categorical_acc
uracy: 0.9673 - acc: 0.9673 - mean_squared_error: 0.0046
Epoch 62/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0804 - categorical_acc
uracy: 0.9772 - acc: 0.9772 - mean_squared_error: 0.0035
Epoch 63/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0810 - categorical_acc
uracy: 0.9770 - acc: 0.9770 - mean_squared_error: 0.0038
Epoch 64/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0852 - categorical_acc
uracy: 0.9744 - acc: 0.9744 - mean_squared_error: 0.0042
Epoch 65/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0898 - categorical_acc
uracy: 0.9740 - acc: 0.9740 - mean_squared_error: 0.0043
Epoch 66/200
78/78 [==============================] - 22s 280ms/step - loss: 0.0751 - categorical_acc
uracy: 0.9780 - acc: 0.9780 - mean_squared_error: 0.0035
Epoch 67/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0722 - categorical_acc
uracy: 0.9798 - acc: 0.9798 - mean_squared_error: 0.0034
Epoch 68/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0807 - categorical_acc
uracy: 0.9746 - acc: 0.9746 - mean_squared_error: 0.0038
Epoch 69/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0744 - categorical_acc
uracy: 0.9790 - acc: 0.9790 - mean_squared_error: 0.0035
Epoch 70/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0724 - categorical_acc
uracy: 0.9770 - acc: 0.9770 - mean_squared_error: 0.0036
Epoch 71/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0743 - categorical_acc
uracy: 0.9780 - acc: 0.9780 - mean_squared_error: 0.0036
Epoch 72/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0648 - categorical_acc
uracy: 0.9810 - acc: 0.9810 - mean_squared_error: 0.0030
Epoch 73/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0696 - categorical_acc
uracy: 0.9798 - acc: 0.9798 - mean_squared_error: 0.0033
Epoch 74/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0650 - categorical_acc
uracy: 0.9802 - acc: 0.9802 - mean_squared_error: 0.0031
Epoch 75/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0613 - categorical_acc
uracy: 0.9824 - acc: 0.9824 - mean_squared_error: 0.0029
Epoch 76/200
78/78 [==============================] - 22s 280ms/step - loss: 0.0608 - categorical_acc
uracy: 0.9806 - acc: 0.9806 - mean_squared_error: 0.0029
Epoch 77/200
78/78 [==============================] - 22s 281ms/step - loss: 0.0640 - categorical_acc
uracy: 0.9790 - acc: 0.9790 - mean_squared_error: 0.0032
Epoch 78/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0619 - categorical_acc
uracy: 0.9800 - acc: 0.9800 - mean_squared_error: 0.0031
Epoch 79/200
```

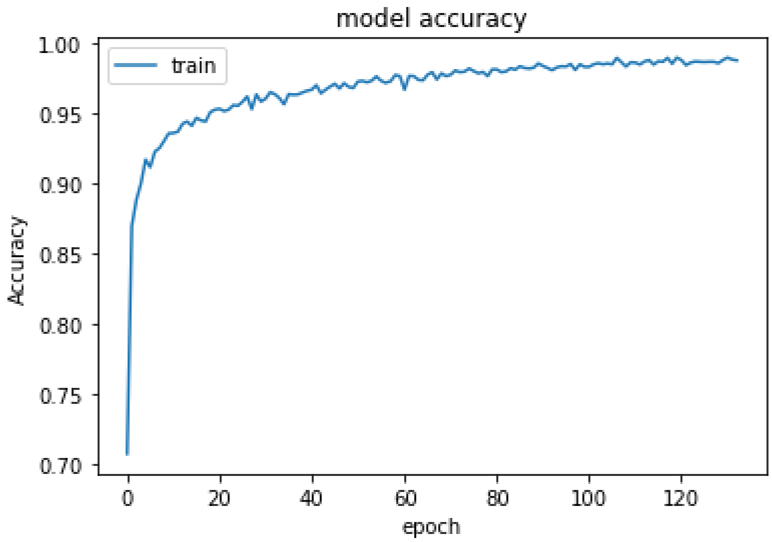
```
78/78 [==============================] - 22s 279ms/step - loss: 0.0699 - categorical_acc
uracy: 0.9770 - acc: 0.9770 - mean_squared_error: 0.0035
Epoch 80/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0641 - categorical_acc
uracy: 0.9818 - acc: 0.9818 - mean_squared_error: 0.0030
Epoch 81/200
78/78 [==============================] - 22s 279ms/step - loss: 0.0642 - categorical_acc
uracy: 0.9818 - acc: 0.9818 - mean_squared_error: 0.0030
Epoch 82/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0638 - categorical_acc
uracy: 0.9798 - acc: 0.9798 - mean_squared_error: 0.0032
Epoch 83/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0647 - categorical_acc
uracy: 0.9804 - acc: 0.9804 - mean_squared_error: 0.0032
Epoch 84/200
78/78 [==============================] - 22s 282ms/step - loss: 0.0564 - categorical_acc
uracy: 0.9826 - acc: 0.9826 - mean_squared_error: 0.0027
Epoch 85/200
78/78 [==============================] - 22s 282ms/step - loss: 0.0630 - categorical_acc
uracy: 0.9816 - acc: 0.9816 - mean_squared_error: 0.0029
Epoch 86/200
78/78 [==============================] - 22s 275ms/step - loss: 0.0545 - categorical_acc
uracy: 0.9841 - acc: 0.9841 - mean_squared_error: 0.0026
Epoch 87/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0561 - categorical_acc
uracy: 0.9828 - acc: 0.9828 - mean_squared_error: 0.0027
Epoch 88/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0571 - categorical_acc
uracy: 0.9824 - acc: 0.9824 - mean_squared_error: 0.0028
Epoch 89/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0530 - categorical_acc
uracy: 0.9831 - acc: 0.9831 - mean_squared_error: 0.0026
Epoch 90/200
78/78 [==============================] - 22s 275ms/step - loss: 0.0493 - categorical_acc
uracy: 0.9861 - acc: 0.9861 - mean_squared_error: 0.0024
Epoch 91/200
78/78 [==============================] - 22s 284ms/step - loss: 0.0506 - categorical_acc
uracy: 0.9843 - acc: 0.9843 - mean_squared_error: 0.0024
Epoch 92/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0538 - categorical_acc
uracy: 0.9828 - acc: 0.9828 - mean_squared_error: 0.0027
Epoch 93/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0582 - categorical_acc
uracy: 0.9812 - acc: 0.9812 - mean_squared_error: 0.0029
Epoch 94/200
78/78 [==============================] - 22s 280ms/step - loss: 0.0526 - categorical_acc
uracy: 0.9833 - acc: 0.9833 - mean_squared_error: 0.0025
Epoch 95/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0543 - categorical_acc
uracy: 0.9841 - acc: 0.9841 - mean_squared_error: 0.0026
Epoch 96/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0502 - categorical_acc
uracy: 0.9839 - acc: 0.9839 - mean_squared_error: 0.0025
Epoch 97/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0518 - categorical_acc
uracy: 0.9857 - acc: 0.9857 - mean_squared_error: 0.0024
Epoch 98/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0559 - categorical_acc
uracy: 0.9814 - acc: 0.9814 - mean_squared_error: 0.0028
Epoch 99/200
```

```
78/78 [==============================] - 22s 279ms/step - loss: 0.0471 - categorical_acc
uracy: 0.9855 - acc: 0.9855 - mean_squared_error: 0.0023
Epoch 100/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0478 - categorical_acc
uracy: 0.9837 - acc: 0.9837 - mean_squared_error: 0.0024
Epoch 101/200
78/78 [==============================] - 22s 285ms/step - loss: 0.0520 - categorical_acc
uracy: 0.9837 - acc: 0.9837 - mean_squared_error: 0.0026
Epoch 102/200
78/78 [==============================] - 22s 279ms/step - loss: 0.0444 - categorical_acc
uracy: 0.9855 - acc: 0.9855 - mean_squared_error: 0.0022
Epoch 103/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0455 - categorical_acc
uracy: 0.9863 - acc: 0.9863 - mean_squared_error: 0.0022
Epoch 104/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0489 - categorical_acc
uracy: 0.9855 - acc: 0.9855 - mean_squared_error: 0.0024
Epoch 105/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0510 - categorical_acc
uracy: 0.9861 - acc: 0.9861 - mean_squared_error: 0.0024
Epoch 106/200
78/78 [==============================] - 22s 280ms/step - loss: 0.0524 - categorical_acc
uracy: 0.9855 - acc: 0.9855 - mean_squared_error: 0.0024
Epoch 107/200
78/78 [==============================] - 22s 280ms/step - loss: 0.0389 - categorical_acc
uracy: 0.9901 - acc: 0.9901 - mean_squared_error: 0.0017
Epoch 108/200
78/78 [==============================] - 22s 277ms/step - loss: 0.0443 - categorical_acc
uracy: 0.9871 - acc: 0.9871 - mean_squared_error: 0.0021
Epoch 109/200
78/78 [==============================] - 22s 281ms/step - loss: 0.0492 - categorical_acc
uracy: 0.9839 - acc: 0.9839 - mean_squared_error: 0.0024
Epoch 110/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0439 - categorical_acc
uracy: 0.9869 - acc: 0.9869 - mean_squared_error: 0.0022
Epoch 111/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0445 - categorical_acc
uracy: 0.9867 - acc: 0.9867 - mean_squared_error: 0.0021
Epoch 112/200
78/78 [==============================] - 22s 281ms/step - loss: 0.0466 - categorical_acc
uracy: 0.9853 - acc: 0.9853 - mean_squared_error: 0.0023
Epoch 113/200
78/78 [==============================] - 22s 281ms/step - loss: 0.0436 - categorical_acc
uracy: 0.9875 - acc: 0.9875 - mean_squared_error: 0.0021
Epoch 114/200
78/78 [==============================] - 21s 273ms/step - loss: 0.0380 - categorical_acc
uracy: 0.9885 - acc: 0.9885 - mean_squared_error: 0.0018
Epoch 115/200
78/78 [==============================] - 22s 275ms/step - loss: 0.0453 - categorical_acc
uracy: 0.9853 - acc: 0.9853 - mean_squared_error: 0.0023
Epoch 116/200
78/78 [==============================] - 21s 273ms/step - loss: 0.0401 - categorical_acc
uracy: 0.9877 - acc: 0.9877 - mean_squared_error: 0.0020
Epoch 117/200
78/78 [==============================] - 22s 275ms/step - loss: 0.0395 - categorical_acc
uracy: 0.9873 - acc: 0.9873 - mean_squared_error: 0.0019
Epoch 118/200
78/78 [==============================] - 23s 295ms/step - loss: 0.0341 - categorical_acc
uracy: 0.9901 - acc: 0.9901 - mean_squared_error: 0.0016
Epoch 119/200
```

```
78/78 [==============================] - 23s 288ms/step - loss: 0.0446 - categorical_acc
uracy: 0.9859 - acc: 0.9859 - mean_squared_error: 0.0022
Epoch 120/200
78/78 [==============================] - 24s 306ms/step - loss: 0.0379 - categorical_acc
uracy: 0.9903 - acc: 0.9903 - mean_squared_error: 0.0018
Epoch 121/200
78/78 [==============================] - 23s 297ms/step - loss: 0.0359 - categorical_acc
uracy: 0.9885 - acc: 0.9885 - mean_squared_error: 0.0018
Epoch 122/200
78/78 [==============================] - 23s 289ms/step - loss: 0.0411 - categorical_acc
uracy: 0.9849 - acc: 0.9849 - mean_squared_error: 0.0021
Epoch 123/200
78/78 [==============================] - 22s 278ms/step - loss: 0.0454 - categorical_acc
uracy: 0.9867 - acc: 0.9867 - mean_squared_error: 0.0023
Epoch 124/200
78/78 [==============================] - 22s 279ms/step - loss: 0.0412 - categorical_acc
uracy: 0.9875 - acc: 0.9875 - mean_squared_error: 0.0019
Epoch 125/200
78/78 [==============================] - 22s 276ms/step - loss: 0.0380 - categorical_acc
uracy: 0.9873 - acc: 0.9873 - mean_squared_error: 0.0019
Epoch 126/200
78/78 [==============================] - 22s 286ms/step - loss: 0.0408 - categorical_acc
uracy: 0.9871 - acc: 0.9871 - mean_squared_error: 0.0020
Epoch 127/200
78/78 [==============================] - 24s 301ms/step - loss: 0.0445 - categorical_acc
uracy: 0.9873 - acc: 0.9873 - mean_squared_error: 0.0021
Epoch 128/200
78/78 [==============================] - 22s 285ms/step - loss: 0.0451 - categorical_acc
uracy: 0.9873 - acc: 0.9873 - mean_squared_error: 0.0020
Epoch 129/200
78/78 [==============================] - 22s 283ms/step - loss: 0.0428 - categorical_acc
uracy: 0.9863 - acc: 0.9863 - mean_squared_error: 0.0021
Epoch 130/200
78/78 [==============================] - 22s 279ms/step - loss: 0.0375 - categorical_acc
uracy: 0.9885 - acc: 0.9885 - mean_squared_error: 0.0018
Epoch 131/200
78/78 [==============================] - 22s 282ms/step - loss: 0.0353 - categorical_acc
uracy: 0.9903 - acc: 0.9903 - mean_squared_error: 0.0017
Epoch 132/200
78/78 [==============================] - 22s 281ms/step - loss: 0.0353 - categorical_acc
uracy: 0.9889 - acc: 0.9889 - mean_squared_error: 0.0017
Epoch 133/200
78/78 [==============================] - 22s 284ms/step - loss: 0.0356 - categorical_acc
uracy: 0.9883 - acc: 0.9883 - mean_squared_error: 0.0018
Epoch 133: early stopping
```

In [9]:

```python
# Plot the progression of the acccuracy through the epochs
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.title('model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [10]:

```
# Demo to randomly pick a sign and demonstrate that it is predicted correctly
from random import randint
test_image = randint(0,len(data_train))
test = np.expand_dims(data_train[test_image], axis=0)
test = tf.keras.applications.inception_v3.preprocess_input(
    test, data_format=None
)
print(one_hot_translator(model.predict(test)))
plt.imshow(data_train[test_image])
```

```
1/1 [==============================] - 2s 2s/step
7
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x27bad781dc0>
```

In [11]:

```
# Calculate the predictions for the test values
predictions = model.predict(x_test)
```

```
39/39 [==============================] - 3s 52ms/step
```

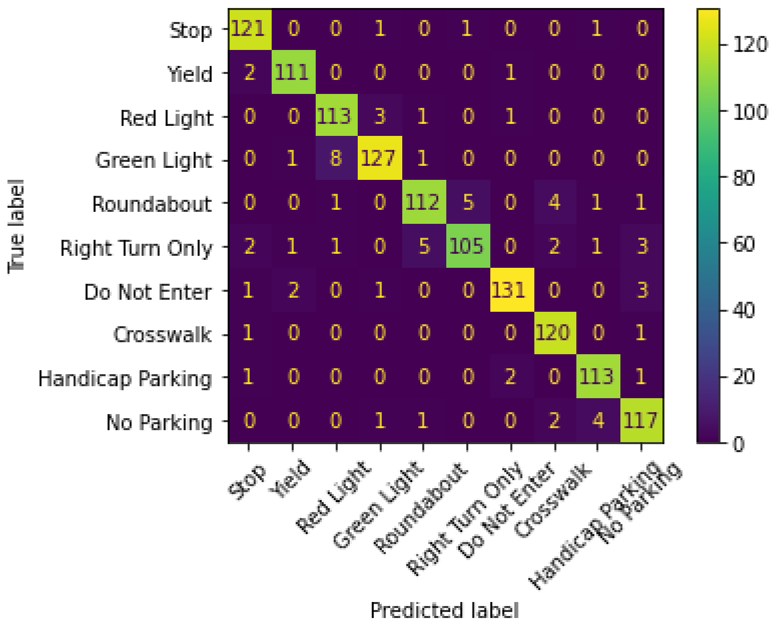
In [12]:

```
evaluation = model.evaluate(x_test, t_test)
print("Test run accuracy is {}".format(evaluation[-2]))
```

```
39/39 [==============================] - 3s 35ms/step - loss: 0.3274 - categorical_accur
```

```
acy: 0.9443 - acc: 0.9443 - mean_squared_error: 0.0095
Test run accuracy is 0.9443099498748779
```

In [26]:
```python
# Create a Confusion Matrix to show the weakness in the model
predicted_values = translate_all(predictions)
real_values = translate_all(t_test)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cfm = confusion_matrix(real_values, predicted_values)
disp = ConfusionMatrixDisplay(confusion_matrix=cfm, display_labels=['Stop','Yield','Red
disp.plot(xticks_rotation=45)
plt.show()
```



In [14]:
```python
# Test Run Data

# Testing optimizers
# NADAM=                                     times = [275,272,286], loss after 3 runs = [
# ADAM=                                      times = [190,186,190], loss after 3 runs = [
# SGD(.001, Nesterov=No, Momentum = No),     times = [182,180,181], loss after 3 runs = [
# SGD(.01, Nesterov=No, Momentum = No),      times = [180,180,179], loss after 3 runs = [
# SGD(.01, Nesterov=Yes, Momentum = 0.25), times = [183,183,183], loss after 3 runs = [
# SGD(.01, Nesterov=Yes, Momentum = 0.5),   times = [189,183,183], loss after 3 runs = [
# SGD(.01, Nesterov=Yes, Momentum = 0.75), times = [185,185,184], loss after 3 runs = [
```

In [15]:
```python
# This method is used to show an example of the post processed test data
from random import randint
import matplotlib.pyplot as plt
test_image = randint(0,len(x_train))
plt.imshow(x_train[test_image])
```

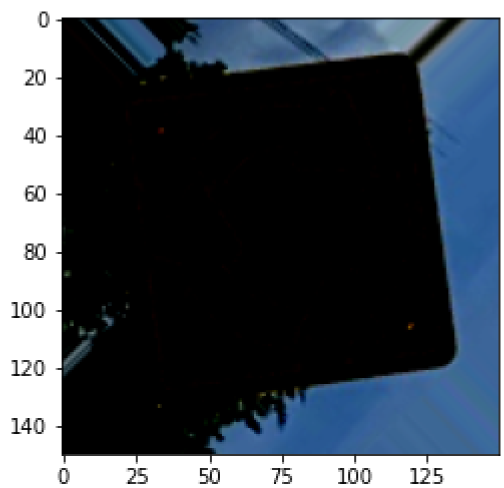
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x27badb2d940>
```

In [16]:

```
# This method is used to show an example of the augmented test data
from random import randint
x,y = next(train_generator)
plt.imshow(x[randint(0,63)])
plt.show()
```

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



In [17]:

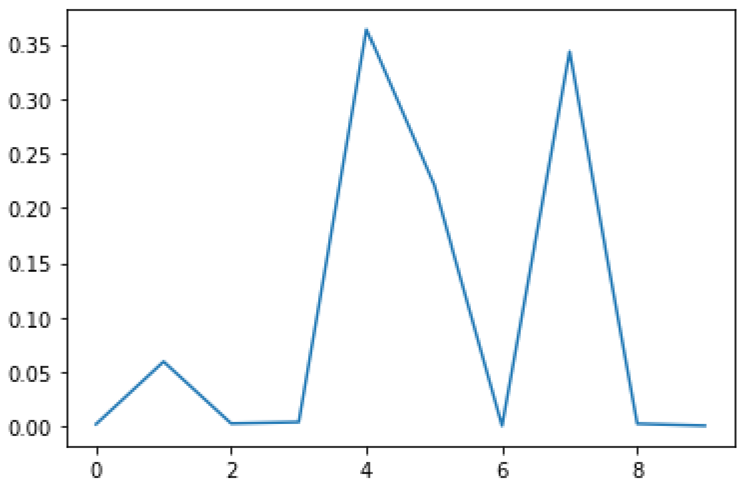
```
# Load an arbitray image and display its predicted value
try:
    Load_image = tf.keras.preprocessing.image.load_img('yield.jpg')
    og_image = Load_image.copy()
    Load_image = np.array(tf.cast(tf.image.resize(Load_image,(150,150)), np.uint8))
    print(Load_image.shape)
    Load_image = tf.keras.applications.inception_v3.preprocess_input(
        Load_image, data_format=None
    )
    Load_image = tf.expand_dims(Load_image,0)
    prediction = model.predict(Load_image)
    print(prediction)
    print(one_hot_translator_thres(prediction))

    plt.plot([0,1,2,3,4,5,6,7,8,9],prediction[0])
    plt.show()
```

```
    except:
        print("No alternate images")
```

```
No alternate images
```

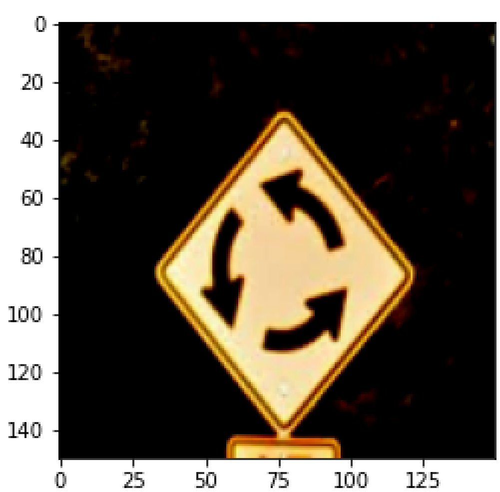
In [86]:
```
#print(x)
#print(np.argmax(x))
#print(predictions)
rand_image = randint(0,len(predictions))
plt.plot(list(range(0,len(predictions[rand_image]))),predictions[rand_image])
```

Out[86]:
```
[<matplotlib.lines.Line2D at 0x27be2ceddf0>]
```



In [87]:
```
plt.imshow(x_test[rand_image])
print(one_hot_translator(t_test[rand_image]))
print(one_hot_translator(predictions[rand_image]))
print(sum(predictions[rand_image]))
```

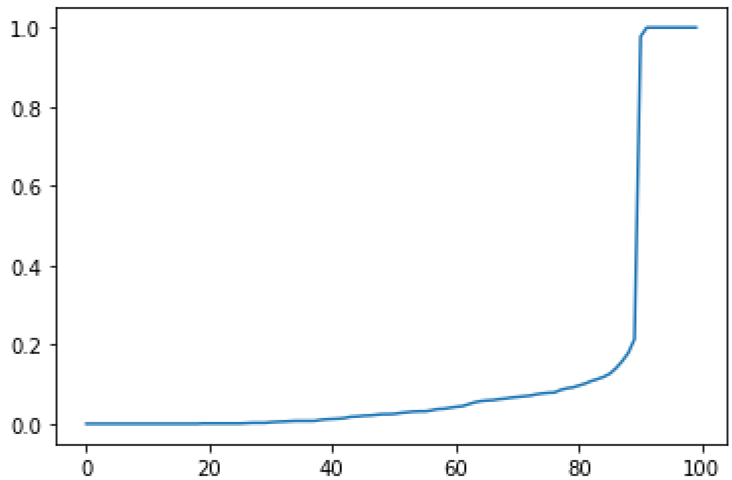
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
4
4
0.9999999767751433
```



In [ ]:

In [20]:

```
# Find ideal cutoff threshold to facilitate weeding out of Unknown items
max_prediction_value = np.array([i[np.argmax(i)] for i in predictions])
total_len = len(max_prediction_value)
threshold_list = np.arange(.1,1.1,.01)
target_acc = .9
plotter = []
for threshold in threshold_list:
    plotter.append(len(np.where(max_prediction_value < threshold)[0])/total_len)
plt.plot(plotter)
plt.show()
plotter = np.sort(plotter)
print(len(np.where(plotter>target_acc)[0])/total_len)
# Insert vertical line at the 90 percintile cutoff
# Include labels and titles
```




```
0.008071025020177562
```

In [88]:

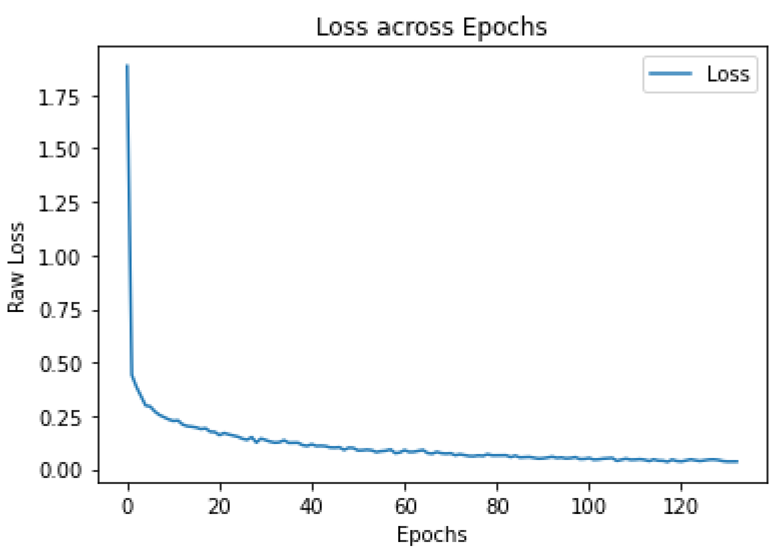
```
# Evaluate performance of grading models with the threshold
original_predicted_labels = translate_all(predictions)
threshold_predicted_labels = translate_all_thres(predictions)

preserved_values = 0
for index, i in enumerate(original_predicted_labels):
    if i == real_values[index]:
        preserved_values = preserved_values + 1
print(preserved_values / len(original_predicted_labels))

preserved_values = 0
print(real_values)
print(original_predicted_labels)
for index, i in enumerate(threshold_predicted_labels):
    if i == real_values[index]:
        preserved_values = preserved_values + 1
print(preserved_values / len(original_predicted_labels))
```

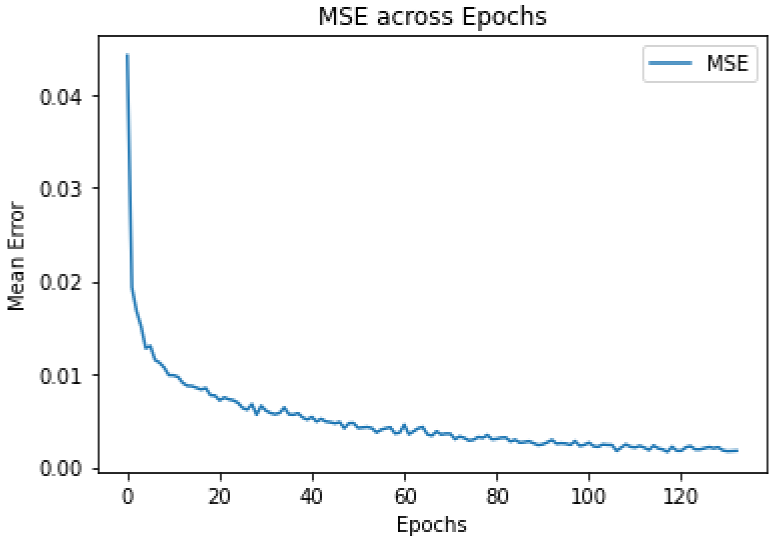
```
0.9443099273607748
[2 7 7 ... 2 6 7]
[2 7 7 ... 2 6 7]
0.9023405972558515
```

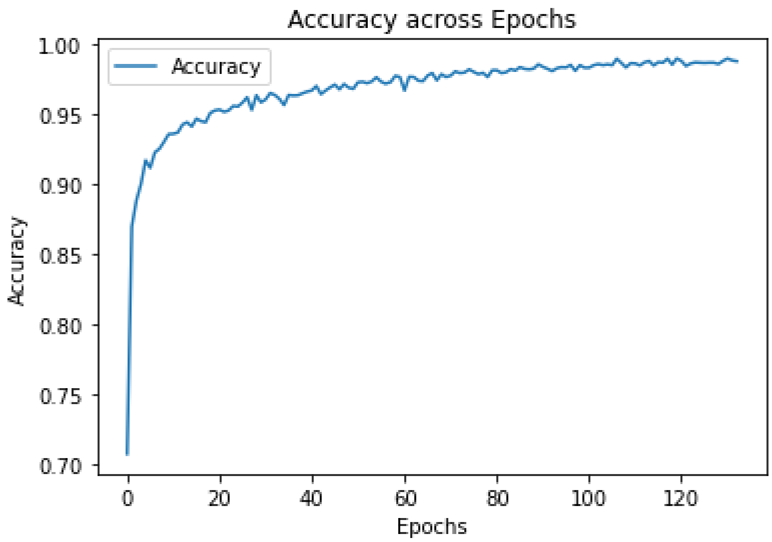
In [162…

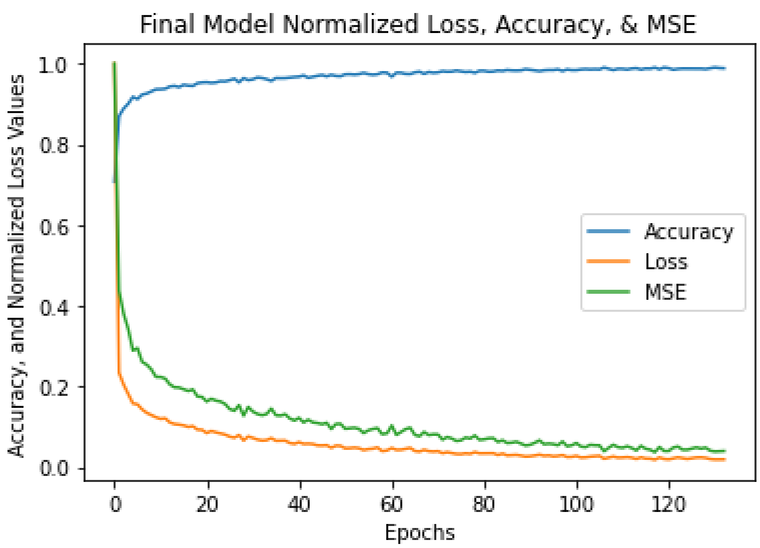
```
from sklearn.preprocessing import normalize
loss = normalize([history.history['loss']], norm='max', axis=1)[0]
```

```
mse = normalize([history.history['mean_squared_error']], norm='max', axis=1)[0]

plt.title("Loss across Epochs")
plt.xlabel("Epochs")
plt.ylabel("Raw Loss")
plt.plot(history.epoch, history.history['loss'], label="Loss")
plt.legend()
plt.show()

plt.title("MSE across Epochs")
plt.xlabel("Epochs")
plt.ylabel("Mean Error")
plt.plot(history.epoch, history.history['mean_squared_error'], label="MSE")
plt.legend()
plt.show()

plt.title("Accuracy across Epochs")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.plot(history.epoch, history.history['acc'], label="Accuracy")
plt.legend()
plt.show()

plt.title("Final Model Normalized Loss, Accuracy, & MSE")
plt.xlabel("Epochs")
plt.ylabel("Accuracy, and Normalized Loss Values")
plt.plot(history.epoch, history.history['acc'], label="Accuracy")
plt.plot(history.epoch, loss, label="Loss")
plt.plot(history.epoch, mse, label="MSE")
plt.legend()
plt.show()
```




MSE across Epochs
MSE
Mean Error
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
120
Epochs


Accuracy across Epochs
Accuracy
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
20
40
60
80
100
120
Epochs


Final Model Normalized Loss, Accuracy, & MSE
Accuracy
Loss
MSE
Accuracy, and Normalized Loss Values
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
120
Epochs

In [ ]:

In [ ]: