Neural Networks

January 20, 2021

1 Neural Networks and Their Power to Process Images

To process and comprehend the world around us, we utilize our powerful minds to help us distinguish different objects from each other. We do this inherently and trained our distinguishing ability throughout the years of our lives. How did we do this? From childhood, we gather data and information about the things around us. Through this accumulated information, we establish clear ideas of what things actually are in reality.

With a lack of information, a basketball would be no different from a soccer ball in our eyes. But with experience and information accumulation, we determine different traits about those objects like shape,color,design to help us distinguish the fact that they are actually different things.

We can use this same human principle and apply it towards computers to have them recognize specific things in an image for us. In this case, we can utilize this in order to help the computer recognize a computer drawn image of an eye and to find the iris in the image. We can accomplish this by "training" the computer with a set of data that is considered "correct" towards our purpose of being able to self recognize future data sets. The sets of data in this case would be the images of computer drawn eyes with the "correct" data being where the irises are in said images. Once the computer has "experienced" a data set, subsequent accumulated experience would help the computer be more accurate just like how we as humans become more accurate with more experience.

The specific name for the method that we will be using to actually accomplish these goals is a convolutional neural network. Which is a form of artificial intelligence that involves taking an image and assigning weights and values to properties of that image and matching other images based on those weights and values based on our own criteria.

2 Establishing the Model

To begin, we first need to create the model that we will be using. We will use the Tensorflow module that is a collection of libraries that help facilitate the creation of a machine learning neural network. It works by utilizing a data representation form known as a tensor.

[9]:
```
#code provded by Dr. John Ringland
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Dropout
```

```python
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model

# build the network

def create_cnn(tot,width, height, depth, nfilters=(16, 32, 64), regress=False):
    # initialize the input shape and channel dimension, assuming
    # TensorFlow/channels-last ordering
    inputShape = (height, width, depth)
    chanDim = -1

    # define the model input
    inputs = Input(shape=inputShape)

    # loop over the number of filters
    for (i, nf) in enumerate(nfilters):
        # if this is the first CONV layer then set the input
        # appropriately
        if i == 0:
            x = inputs

        # CONV => RELU => BN => POOL
        x = Conv2D(nf, (3, 3), padding="same")(x)
        x = Activation("relu")(x)
        x = BatchNormalization(axis=chanDim)(x)
        x = MaxPooling2D(pool_size=(2, 2))(x)

    # flatten the volume, then FC => RELU => BN => DROPOUT
    x = Flatten()(x)
    x = Dense(16)(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chanDim)(x)
    x = Dropout(0.5)(x)

    # apply another FC layer: JR not sure what the role of this is
    x = Dense(4)(x)
    x = Activation("relu")(x)

    x = Dense(tot, activation="linear")(x)    ################  2D output

    # construct the CNN
    model = Model(inputs, x)

    # return the CNN
    return model
```

This code creates the convulutional network that we will be using on our set of images.

A tensor is similar to an array of data but can span many dimensions. The data that we will be using to distinguish features that we will apply our network to will be in the form of tensors.

To elaborate on how a convolutional network works, it takes in an input images of a certain dimension. Once it has the image, it will apply filters to the images. A filter is a method of evaluating the relevancy of a pixel in the image by applying a set of rules on that pixel and the neighborhood surrounding that pixel. Those set of rules will determine how important or relevant that pixel is towards our objective and gives it a weight based on that result. The more the filters we use in our network, the more likely we are able to find an accurate result to accomplish our objective. Complex neural networks will often have many layers of filters upon filters to handle the image.

Specifically in this use case, we are trying to find the (x,y) coordinates of the center of the iris inside a picture of an eye. By training our network through giving it images with the corresponding correct set of (x,y) coordinates, it is able to see just how far it's results are from the actual correct coordinates and make adjustments accordingly to take steps towards obtaining the correct set of coordinates for the next case.

[48]:
```
#code provided by Dr. John Ringland
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
import numpy as np
import os
from os.path import join
from glob import glob
import cv2
import pandas as pd

def setup(datafolder,columns_to_use):

    datafile   = join(datafolder, datafolder + ".txt")
    # construct the path to the input .txt file that contains information
    # on each image in the dataset and then load the dataset
    #print("Loading attributes...")

    df = pd.read_csv(datafile, sep =" ")[columns_to_use].copy()

    # load the images and then scale the pixel intensities to the
    # range [0, 1]
    print("Loading images...")
    imagefiles = sorted(glob(join(datafolder,"*.png")))
    images = np.array([cv2.imread(imagefile)[:,:,:3] for imagefile in␣
 ↪imagefiles]) #slice off opacity layer
    images = images / 255.0

    # partition the data into training and testing splits using 75% of
    # the data for training and the remaining 25% for testing
    split = train_test_split(df, images, test_size=0.25, random_state=42)
    (trainAttrX, testAttrX, trainImagesX, testImagesX) = split
    #to recale trainY and testY
    minAttr = trainAttrX.min(axis = 0)
    maxAttr = trainAttrX.max(axis = 0)
    rangeAttr = maxAttr - minAttr

    trainAttrX -= minAttr
    trainAttrX = (trainAttrX/rangeAttr).values

    testAttrX -= minAttr
    testAttrX = (testAttrX/rangeAttr).values

    return␣
↪trainAttrX,testAttrX,trainImagesX,testImagesX,minAttr,maxAttr,rangeAttr
```

With this code, we are now able to begin training and testing our images for accuracy. In the setup for the model, we loaded in our image set, scaled them, and split the data so that we are training our model on 75% of the images and testing that trained model on the remaining 25% of our images. The image set in question will be a pregenerated image set of 1000 images. This function itself will return the data for us to use in the model.

Now we can finally run the model and have it predict the position of the (x,y) coordinates of the eyes. We'll start off with a relatively quick training period of two epochs. An epoch is the number of times you pass the entire dataset into the model to be trained from.

3 Training and Testing the Model

[10]:
```
tra,testa,tri,testi,mina,maxa,rangea = setup("one_eye",["x","y"])
#data from the setup function

#creating the model
model = create_cnn(2,90,48, 3, regress=True)
opt = Adam(lr=1e-3, decay=1e-3 / 200)
model.compile(loss="mean_squared_error", optimizer=opt)

# training the model
print("Training model...")
model.fit(tri, tra, validation_data=(testi, testa),
        epochs=2, batch_size=8)

model.save("model.h5")

preds = model.predict(testi)

print("Done")
```

```
Loading images...
Training model...
```

```
Train on 750 samples, validate on 250 samples
Epoch 1/2
750/750 [==============================] - 10s 14ms/sample - loss: 0.4846 -
val_loss: 0.0958
Epoch 2/2
750/750 [==============================] - 8s 11ms/sample - loss: 0.2424 -
val_loss: 0.7367
Done
```

Now that we have trained and tested on our dataset, it is time to see how well it predicted the results. We can use the mean squared error formula to see just how much of an error we had when predicting the coordinates. The closer to 0 the error is, the closer the prediction is to the actual coordinate.

[11]:
```python
def mse(p,t):
    predstest = p.copy()
    testatest = t.copy()
    result = testatest - predstest
    squared = result ** 2
    summed = np.sum(squared,axis = 0)
    ans = summed/len(squared)
    print("Error in X: " + str(ans[0]))
    print("Error in Y: " + str(ans[1]))
```

[12]:
```python
mse(preds,testa)
```

```
Error in X: 0.8818159522986927
Error in Y: 0.5915493678561144
```

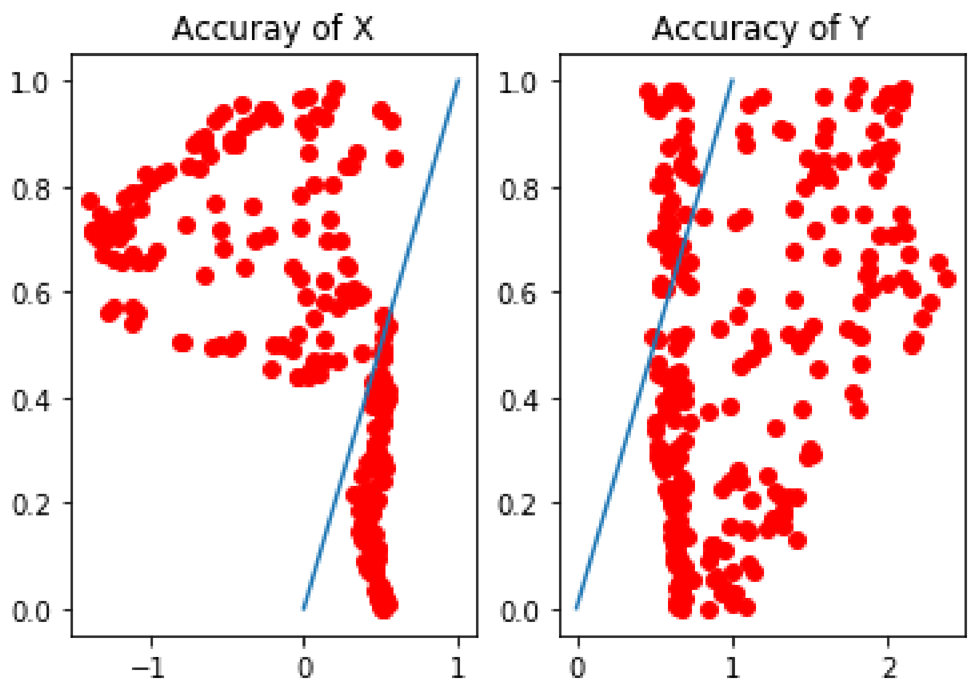
We can also create a visual graphical component in order to showcase how accurate the prediction of the model is as well.

[15]:
```python
import matplotlib.pyplot as plt
def graphrep(p,t):
    preds = p.copy()
    xpreds = preds[:,0] #take the x value of predicted results
    ypreds = preds[:,1] #take the y value of predicted results
    testa = t.copy()
    xa = testa[:,0] #take the x value of actual data
    ya = testa[:,1] #take y value of actual data

    fig, (ax1, ax2) = plt.subplots(1,2)
    x = np.linspace(0,1,1000)
    ax1.plot(xpreds,xa,"ro")
    ax1.plot(x,x)
    ax2.plot(ypreds,ya,"ro")
    ax2.plot(x,x)
    ax1.title.set_text("Accuray of X")
    ax2.title.set_text("Accuracy of Y")
```

In our graph we will be plotting the predicted values against the actual values. This graph will also include the line y = x which represents the total accuracy line. This means the closer the points on the graph are to the line y = x, the more accurate the predictions of the model.

[16]:
```
graphrep(preds,testa)
```



Here we can see that the accuracy of the model is all over the place with the clusters not even close to resembling a correct prediction of (x,y) values.

4 Can we do better?

How can we make the predictions of our model more accurate? One of the ways is to make alterations to how the model trains. In our first scenario, the model trained for 2 epochs. We can try to increase the epochs in an effort to increase the accuracy of the model.

[18]:
```
#code provided by Dr. John Ringland
tra,testa,tri,testi,mina,maxa,rangea = setup("one_eye",["x","y"])
#data from the setup function

#creating the model
model = create_cnn(2,90,48, 3, regress=True)
opt = Adam(lr=1e-3, decay=1e-3 / 200)
model.compile(loss="mean_squared_error", optimizer=opt)

# training the model
print("Training model...")
```

```
model.fit(tri, tra, validation_data=(testi, testa),
        epochs=10, batch_size=8)

model.save("model.h5")

preds = model.predict(testi)

print("Done")
```

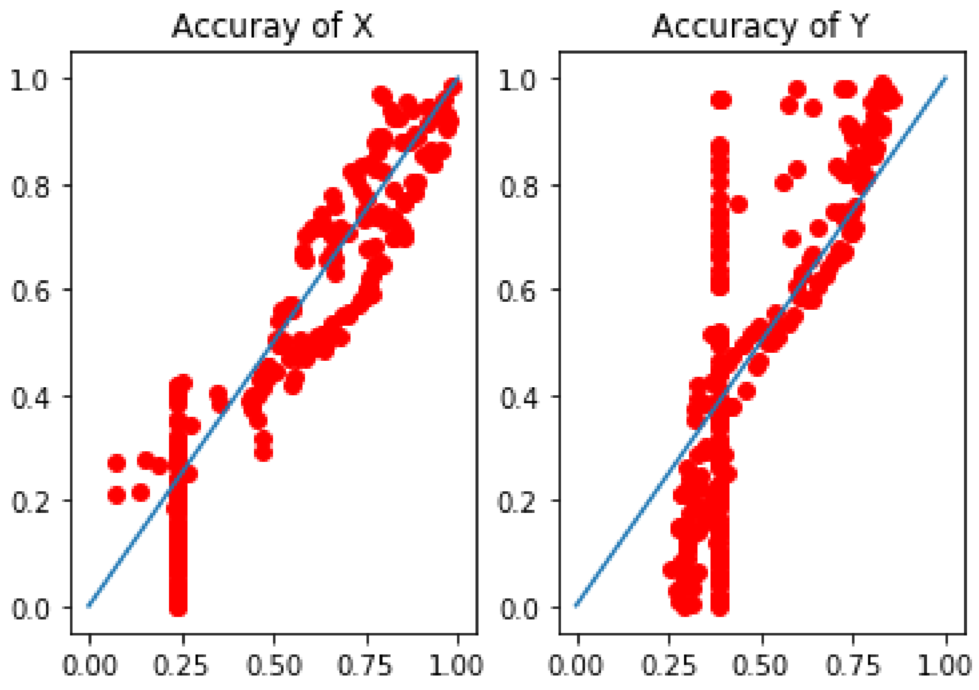
```
Loading images...
Training model...
Train on 750 samples, validate on 250 samples
Epoch 1/10
750/750 [==============================] - 10s 13ms/sample - loss: 0.5118 -
val_loss: 0.2815
Epoch 2/10
750/750 [==============================] - 9s 11ms/sample - loss: 0.2689 -
val_loss: 0.2451
Epoch 3/10
750/750 [==============================] - 8s 11ms/sample - loss: 0.1857 -
val_loss: 0.2147
Epoch 4/10
750/750 [==============================] - 8s 11ms/sample - loss: 0.1478 -
val_loss: 0.1874
Epoch 5/10
750/750 [==============================] - 8s 11ms/sample - loss: 0.1139 -
val_loss: 0.1639
Epoch 6/10
750/750 [==============================] - 8s 10ms/sample - loss: 0.0970 -
val_loss: 0.1450
Epoch 7/10
750/750 [==============================] - 7s 10ms/sample - loss: 0.0782 -
val_loss: 0.0706
Epoch 8/10
750/750 [==============================] - 7s 10ms/sample - loss: 0.0708 -
val_loss: 0.0434
Epoch 9/10
750/750 [==============================] - 8s 10ms/sample - loss: 0.0656 -
val_loss: 0.0410
Epoch 10/10
750/750 [==============================] - 8s 11ms/sample - loss: 0.0663 -
val_loss: 0.0387
Done
```

[125]:
```
mse(preds,testa)
```

```
Error in X: 0.012689644970984074
Error in Y: 0.03797099675822403
```

Notice that after raising the number of epochs to 10, the error for each of the coordinates has gone down significantly.

[126]:
```
graphrep(preds,testa)
```



The same pattern seems to be the case for the graphs. The points seem to be trending closer and closer to the line y = x, signaling an improvement in accuracy but we still have a considerable amount of clusters straying from the line y = x

Lets try an even higher number of epochs and see how the accuracy changes.

[86]:
```
tra,testa,tri,testi,mina,maxa,rangea = setup("one_eye",["x","y"])

model = create_cnn(2,90,48, 3, regress=True)
opt = Adam(lr=1e-3, decay=1e-3 / 200)
model.compile(loss="mean_squared_error", optimizer=opt)

print("Training model...")
model.fit(tri, tra, validation_data=(testi, testa),
        epochs=100, batch_size=8)

model.save("model.h5")
preds = model.predict(testi)
print("Done")
```

```
Loading images...
Training model...
```

```
Train on 750 samples, validate on 250 samples
Epoch 1/100
750/750 [==============================] - 9s 13ms/sample - loss: 0.6256 -
val_loss: 0.1923
Epoch 2/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.2476 -
val_loss: 0.3385
Epoch 3/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.1954 -
val_loss: 0.2707
Epoch 4/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.1422 -
val_loss: 0.1382
Epoch 5/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.1021 -
val_loss: 0.1268
Epoch 6/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0934 -
val_loss: 0.0928
Epoch 7/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0770 -
val_loss: 0.0472
Epoch 8/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0671 -
val_loss: 0.0381
Epoch 9/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0552 -
val_loss: 0.0270
Epoch 10/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0519 -
val_loss: 0.0207
Epoch 11/100
750/750 [==============================] - 8s 11ms/sample - loss: 0.0494 -
val_loss: 0.0225
Epoch 12/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0459 -
val_loss: 0.0167
Epoch 13/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0407 -
val_loss: 0.0160
Epoch 14/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0409 -
val_loss: 0.0235
Epoch 15/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0459 -
val_loss: 0.0162
Epoch 16/100
750/750 [==============================] - 8s 11ms/sample - loss: 0.0374 -
```

```
val_loss: 0.0145
Epoch 17/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0381 -
val_loss: 0.0141
Epoch 18/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0365 -
val_loss: 0.0155
Epoch 19/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0364 -
val_loss: 0.0119
Epoch 20/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0345 -
val_loss: 0.0140
Epoch 21/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0349 -
val_loss: 0.0121
Epoch 22/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0329 -
val_loss: 0.0111
Epoch 23/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0321 -
val_loss: 0.0089
Epoch 24/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0341 -
val_loss: 0.0083
Epoch 25/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0316 -
val_loss: 0.0106
Epoch 26/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0335 -
val_loss: 0.0110
Epoch 27/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0334 -
val_loss: 0.0108
Epoch 28/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0295 -
val_loss: 0.0110
Epoch 29/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0308 -
val_loss: 0.0103
Epoch 30/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0288 -
val_loss: 0.0096
Epoch 31/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0313 -
val_loss: 0.0091
Epoch 32/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0291 -
```

```
val_loss: 0.0074
Epoch 33/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0294 -
val_loss: 0.0086
Epoch 34/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0300 -
val_loss: 0.0086
Epoch 35/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0293 -
val_loss: 0.0071
Epoch 36/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0294 -
val_loss: 0.0070
Epoch 37/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0302 -
val_loss: 0.0086
Epoch 38/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0276 -
val_loss: 0.0067
Epoch 39/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0293 -
val_loss: 0.0065
Epoch 40/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0271 -
val_loss: 0.0069
Epoch 41/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0296 -
val_loss: 0.0062
Epoch 42/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0303 -
val_loss: 0.0071
Epoch 43/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0271 -
val_loss: 0.0073
Epoch 44/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0278 -
val_loss: 0.0066
Epoch 45/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0299 -
val_loss: 0.0067
Epoch 46/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0284 -
val_loss: 0.0074
Epoch 47/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0293 -
val_loss: 0.0099
Epoch 48/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0263 -
```

```
val_loss: 0.0059
Epoch 49/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0271 -
val_loss: 0.0064
Epoch 50/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0268 -
val_loss: 0.0079
Epoch 51/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0280 -
val_loss: 0.0060
Epoch 52/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0295 -
val_loss: 0.0062
Epoch 53/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0271 -
val_loss: 0.0063
Epoch 54/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0275 -
val_loss: 0.0056
Epoch 55/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0285 -
val_loss: 0.0058
Epoch 56/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0271 -
val_loss: 0.0063
Epoch 57/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0269 -
val_loss: 0.0060
Epoch 58/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0269 -
val_loss: 0.0052
Epoch 59/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0253 -
val_loss: 0.0064
Epoch 60/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0268 -
val_loss: 0.0070
Epoch 61/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0262 -
val_loss: 0.0068
Epoch 62/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0270 -
val_loss: 0.0087
Epoch 63/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0270 -
val_loss: 0.0056
Epoch 64/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0289 -
```

```
val_loss: 0.0048
Epoch 65/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0295 -
val_loss: 0.0051
Epoch 66/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0276 -
val_loss: 0.0048
Epoch 67/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0264 -
val_loss: 0.0063
Epoch 68/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0272 -
val_loss: 0.0052
Epoch 69/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0250 -
val_loss: 0.0049
Epoch 70/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0264 -
val_loss: 0.0053
Epoch 71/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0287 -
val_loss: 0.0048
Epoch 72/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0280 -
val_loss: 0.0047
Epoch 73/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0267 -
val_loss: 0.0044
Epoch 74/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0274 -
val_loss: 0.0077
Epoch 75/100
750/750 [==============================] - 8s 10ms/sample - loss: 0.0256 -
val_loss: 0.0054
Epoch 76/100
750/750 [==============================] - 8s 10ms/sample - loss: 0.0237 -
val_loss: 0.0045
Epoch 77/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0249 -
val_loss: 0.0058
Epoch 78/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0275 -
val_loss: 0.0058
Epoch 79/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0281 -
val_loss: 0.0050
Epoch 80/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0247 -
```

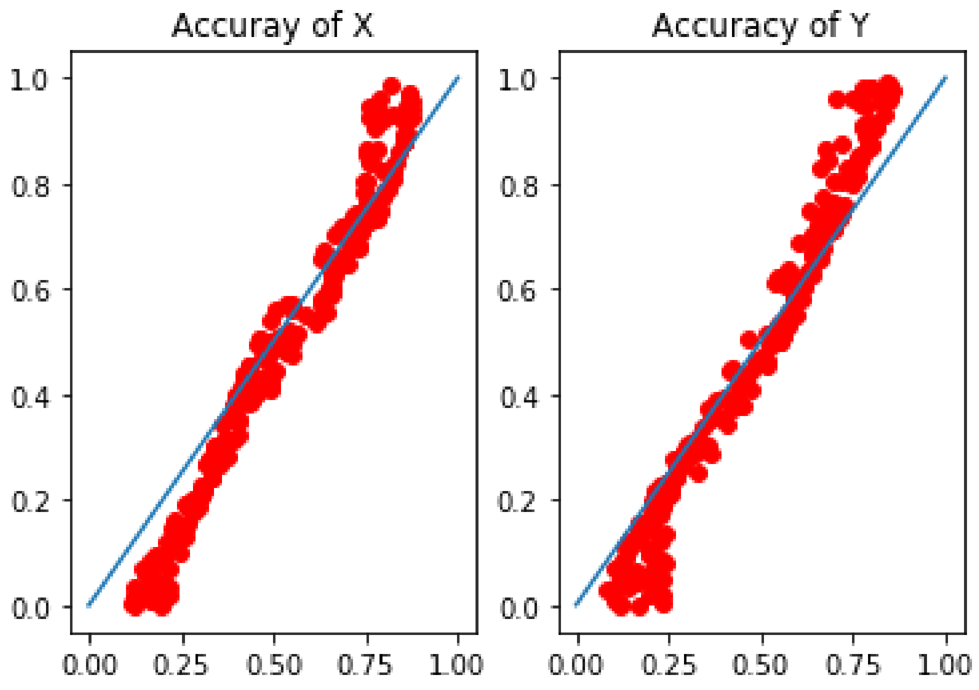
```
val_loss: 0.0052
Epoch 81/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0249 -
val_loss: 0.0052
Epoch 82/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0278 -
val_loss: 0.0045
Epoch 83/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0285 -
val_loss: 0.0057
Epoch 84/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0285 -
val_loss: 0.0066
Epoch 85/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0246 -
val_loss: 0.0046
Epoch 86/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0251 -
val_loss: 0.0051
Epoch 87/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0262 -
val_loss: 0.0045
Epoch 88/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0242 -
val_loss: 0.0047
Epoch 89/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0256 -
val_loss: 0.0083
Epoch 90/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0250 -
val_loss: 0.0081
Epoch 91/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0270 -
val_loss: 0.0050
Epoch 92/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0238 -
val_loss: 0.0049
Epoch 93/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0271 -
val_loss: 0.0056
Epoch 94/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0250 -
val_loss: 0.0062
Epoch 95/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0262 -
val_loss: 0.0070
Epoch 96/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0230 -
```

```
val_loss: 0.0064
Epoch 97/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0227 -
val_loss: 0.0056
Epoch 98/100
750/750 [==============================] - 7s 10ms/sample - loss: 0.0234 -
val_loss: 0.0080
Epoch 99/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0230 -
val_loss: 0.0060
Epoch 100/100
750/750 [==============================] - 7s 9ms/sample - loss: 0.0255 -
val_loss: 0.0063
Done
```

[128]:
```
mse(preds,testa)
```

```
Error in X: 0.006018009619829298
Error in Y: 0.00676164534594826
```

[129]:
```
graphrep(preds,testa)
```



As a final nail in the coffin, we see that after raising the number of epochs to 100, we have an extremely low error value for each coordinate as well as the clusters of points almost hugging the line y = x.

5 Testing With Different Data

We saw how quickly the accuracy of the model's predictions improved when increasing the training time, but we must consider that the test and training sets of data came from the same set of data. This could have a hand in how our model performs. A parallel could be made of a student performing incredibly well on an exam when the exam answers were picked out of a pool of practice questions that the student used to study with. This raises the question of whether the model will have the same level of prediction when testing against a brand new set of data without training with it prior.

We need a brand new set of data with perhaps a subtle change to test the validity of the model's predictions. We can generate a new set of data with the following code.

[58]:
```
#code provide by Dr. John Ringland
svg = '''<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:cc="http://creativecommons.org/ns#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   id="svg1699"
   version="1.1"
   viewBox="0 0 94.872055 51.026783"
   height="48"
   width="90">
  <defs
     id="defs1693">
    <clipPath
       id="clipPath978"
       clipPathUnits="userSpaceOnUse">
      <path
         id="path980"
         d="m -90.424571,169.26703 c -0.826924,-11.9283 30.135244,-42.14526 56.
↪911823,-46.54005 24.3572772,-3.99771 55.9875,12.76809 63.427467,22.47219 -13.
↪037693,22.33642 -37.8766049,33.44776 -53.321611,34.04073 -19.002788,0.72957␣
↪-50.35768,-11.63049 -67.017679,-9.97287 z"
         style="color:#000000;clip-rule:nonzero;display:inline;overflow:visible;
↪visibility:visible;opacity:1;isolation:auto;mix-blend-mode:normal;
↪color-interpolation:sRGB;color-interpolation-filters:linearRGB;solid-color:
↪#000000;solid-opacity:1;vector-effect:none;fill:none;fill-opacity:0.51612902;
↪fill-rule:nonzero;stroke:#0a0a0a;stroke-width:0.70359772;stroke-linecap:butt;
↪stroke-linejoin:miter;stroke-miterlimit:4;stroke-dasharray:none;
↪stroke-dashoffset:0;stroke-opacity:0.5299539;marker:none;color-rendering:
↪auto;image-rendering:auto;shape-rendering:auto;text-rendering:auto;
↪enable-background:accumulate" />
    </clipPath>
  </defs>
  <metadata
```

```
     id="metadata1696">
    <rdf:RDF>
      <cc:Work
         rdf:about="">
        <dc:format>image/svg+xml</dc:format>
        <dc:type
           rdf:resource="http://purl.org/dc/dcmitype/StillImage" />
        <dc:title></dc:title>
      </cc:Work>
    </rdf:RDF>
  </metadata>
  <g
     transform="translate(79.186028,-108.2009)"
     id="layer1">
    <rect
       y="108.2009"
       x="-79.186028"
       height="51.026783"
       width="94.872055"
       id="rect1000"
       style="color:#000000;clip-rule:nonzero;display:inline;overflow:visible;
↪visibility:visible;opacity:1;isolation:auto;mix-blend-mode:normal;
↪color-interpolation:sRGB;color-interpolation-filters:linearRGB;solid-color:
↪#000000;solid-opacity:1;vector-effect:none;fill:#dab6a8;fill-opacity:1;
↪fill-rule:nonzero;stroke:none;stroke-width:0.45317072;stroke-linecap:butt;
↪stroke-linejoin:miter;stroke-miterlimit:4;stroke-dasharray:none;
↪stroke-dashoffset:0;stroke-opacity:0.98617512;marker:none;color-rendering:
↪auto;image-rendering:auto;shape-rendering:auto;text-rendering:auto;
↪enable-background:accumulate" />
    <path
       id="path982"
       d="m 9.7745095,147.90412 c 0.5876285,-8.47665 -21.4151115,-29.94983 -40.
↪4433735,-33.07292 -17.309089,-2.8409 -39.78659,9.07343 -45.07367,15.9695 9.
↪26501,15.873 26.916369,23.76909 37.892119,24.19047 13.504,0.51846 35.
↪7857835,-8.26501 47.6249245,-7.08705 z"
       style="color:#000000;clip-rule:nonzero;display:inline;overflow:visible;
↪visibility:visible;opacity:1;isolation:auto;mix-blend-mode:normal;
↪color-interpolation:sRGB;color-interpolation-filters:linearRGB;solid-color:
↪#000000;solid-opacity:1;vector-effect:none;fill:#ffffff;fill-opacity:0.
↪98617512;fill-rule:nonzero;stroke:#000000;stroke-width:1;stroke-linecap:butt;
↪stroke-linejoin:miter;stroke-miterlimit:4;stroke-dasharray:none;
↪stroke-dashoffset:0;stroke-opacity:0.98617512;marker:none;color-rendering:
↪auto;image-rendering:auto;shape-rendering:auto;text-rendering:auto;
↪enable-background:accumulate" />
    <g
       id="g994"
       transform="matrix(-0.71063337,0,0,0.71063337,-54.509234,27.616976)"
       clip-path="url(#clipPath978)"
       style="opacity:1">
      <g
         id="g992"
         transform="translate(-6.826629,-3.2072343)"
         clip-path="none"
         style="opacity:1">
        <g
           id="g990"
           transform="translate(-2.1381562,-1.6036172)">
          <circle
             r="25.4"
             cy="IRIS_CY"
             cx="IRIS_CX"
             id="circle984"
             style="color:#000000;clip-rule:nonzero;display:inline;overflow:
↪visible;visibility:visible;opacity:0.58899997;isolation:auto;mix-blend-mode:
↪normal;color-interpolation:sRGB;color-interpolation-filters:linearRGB;
↪solid-color:#000000;solid-opacity:1;vector-effect:none;fill:#0e48c4;
↪fill-opacity:1;fill-rule:nonzero;stroke:none;stroke-width:2.00600004;
↪stroke-linecap:butt;stroke-linejoin:miter;stroke-miterlimit:4;
↪stroke-dasharray:none;stroke-dashoffset:0;stroke-opacity:1;marker:none;
↪color-rendering:auto;image-rendering:auto;shape-rendering:auto;
↪text-rendering:auto;enable-background:accumulate" />
          <g
             transform="translate(-39.28862,3.2072343)"
             id="g988"
             style="opacity:1;fill:#000000;fill-opacity:0.78801838">
            <circle
               r="8.5620193"
               cy="PUPIL_CY"
               cx="PUPIL_CX"
               id="circle986"
               style="color:#000000;clip-rule:nonzero;display:inline;overflow:
↪visible;visibility:visible;opacity:1;isolation:auto;mix-blend-mode:normal;
↪color-interpolation:sRGB;color-interpolation-filters:linearRGB;solid-color:
↪#000000;solid-opacity:1;vector-effect:none;fill:#000000;fill-opacity:1;
↪fill-rule:nonzero;stroke:none;stroke-width:0.67619723;stroke-linecap:butt;
↪stroke-linejoin:miter;stroke-miterlimit:4;stroke-dasharray:none;
↪stroke-dashoffset:0;stroke-opacity:1;marker:none;color-rendering:auto;
↪image-rendering:auto;shape-rendering:auto;text-rendering:auto;
↪enable-background:accumulate" />
          </g>
        </g>
      </g>
    </g>
  </g>
```

```
</svg>
'''

xoffset = 5
yoffset = 10
ampx,ampy = 40,15

iris_cy0=144.60001  + yoffset
iris_cx0=-25.4      + xoffset
pupil_cy0=141.39278 + yoffset
pupil_cx0=13.88862  + xoffset

nimages = 100


from os.path import join, exists
from os import makedirs
import os
import numpy as np

def drawsvg(i,x,y,folder):
    if not os.path.exists(folder):
        os.makedirs(folder)
    pngname = join(folder,str(i).zfill(4)+'_one_eye_x' + str(x) + '_y' + str(y)␣
 ↪+ '.png')

    iris_cx = iris_cx0 + x
    iris_cy = iris_cy0 + y
    pupil_cx = pupil_cx0 + x
    pupil_cy = pupil_cy0 + y

    with open('temp.svg','w') as f:
        f.write( svg.replace('IRIS_CX',str(iris_cx)).
 ↪replace('IRIS_CY',str(iris_cy)).replace('PUPIL_CX',str(pupil_cx)).
 ↪replace('PUPIL_CY',str(pupil_cy)) )
    cmd = 'C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-Laptop/
 ↪OneDrive/' + pngname + ' C:/users/Kevin-Laptop/OneDrive/temp.svg'
    print(cmd)
    os.system(cmd)

folder = 'one_eye3'

txt = open( folder + '/one_eye3.txt', 'w' )

for i in range(nimages):
    rx,ry = 2*np.random.rand(2) - 1
    x = round( ampx*rx, 3 )
    y = round( ampy*ry, 3 )
    drawsvg(i, x,y  , folder )
    print( x, y, file=txt )

os.system('del temp.svg')
txt.close()
```

```
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0000_one_eye_x16.57_y11.729.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0001_one_eye_x-23.52_y-3.49.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0002_one_eye_x9.551_y-3.39.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0003_one_eye_x1.41_y13.783.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0004_one_eye_x-29.587_y-5.69.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0005_one_eye_x14.453_y14.578.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0006_one_eye_x-29.351_y3.387.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0007_one_eye_x-20.896_y-14.325.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0008_one_eye_x26.646_y14.697.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0009_one_eye_x37.1_y-7.122.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0010_one_eye_x-18.122_y-1.413.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0011_one_eye_x21.882_y7.324.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0012_one_eye_x4.072_y7.692.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
```

```
Laptop/OneDrive/one_eye3\0013_one_eye_x-10.508_y14.494.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0014_one_eye_x14.255_y1.761.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0015_one_eye_x10.297_y2.361.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0016_one_eye_x16.56_y-14.471.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0017_one_eye_x9.351_y8.869.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0018_one_eye_x14.519_y5.713.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0019_one_eye_x-21.939_y3.866.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0020_one_eye_x6.72_y7.822.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0021_one_eye_x26.207_y13.603.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0022_one_eye_x36.174_y14.104.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0023_one_eye_x-16.758_y13.626.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0024_one_eye_x-10.566_y3.5.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0025_one_eye_x-8.427_y1.232.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0026_one_eye_x-7.186_y14.751.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0027_one_eye_x-36.671_y8.207.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0028_one_eye_x33.206_y-7.59.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
```

```
Laptop/OneDrive/one_eye3\0029_one_eye_x-21.603_y-1.75.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0030_one_eye_x-32.205_y6.135.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0031_one_eye_x32.413_y6.521.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0032_one_eye_x3.887_y-1.474.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0033_one_eye_x-16.67_y12.856.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0034_one_eye_x13.922_y5.603.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0035_one_eye_x-37.375_y-7.332.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0036_one_eye_x20.274_y1.448.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0037_one_eye_x2.156_y-5.48.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0038_one_eye_x-23.681_y-3.017.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0039_one_eye_x-26.706_y-6.689.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0040_one_eye_x-6.91_y7.306.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0041_one_eye_x-11.369_y-3.126.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0042_one_eye_x-37.49_y-11.675.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0043_one_eye_x36.4_y11.243.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0044_one_eye_x17.984_y4.852.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
```

```
Laptop/OneDrive/one_eye3\0045_one_eye_x38.847_y13.084.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0046_one_eye_x-1.24_y13.693.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0047_one_eye_x12.554_y-8.806.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0048_one_eye_x-13.406_y-0.583.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0049_one_eye_x13.809_y12.556.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0050_one_eye_x16.717_y11.079.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0051_one_eye_x33.286_y9.244.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0052_one_eye_x-28.989_y0.012.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0053_one_eye_x-38.419_y-10.41.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0054_one_eye_x23.959_y3.269.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0055_one_eye_x16.375_y7.051.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0056_one_eye_x25.225_y10.039.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0057_one_eye_x34.941_y0.816.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0058_one_eye_x12.412_y-6.711.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0059_one_eye_x28.144_y-14.017.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0060_one_eye_x34.298_y-3.123.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
```

```
Laptop/OneDrive/one_eye3\0061_one_eye_x32.367_y12.406.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0062_one_eye_x-38.851_y-8.382.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0063_one_eye_x-23.589_y-7.238.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0064_one_eye_x29.162_y7.871.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0065_one_eye_x15.571_y7.547.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0066_one_eye_x28.998_y-7.772.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0067_one_eye_x2.704_y-10.052.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0068_one_eye_x-19.326_y14.728.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0069_one_eye_x-22.386_y-0.066.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0070_one_eye_x22.505_y14.029.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0071_one_eye_x-35.01_y-11.032.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0072_one_eye_x20.33_y13.044.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0073_one_eye_x8.201_y5.955.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0074_one_eye_x27.617_y-0.309.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0075_one_eye_x6.44_y4.492.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0076_one_eye_x-29.005_y-2.332.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
```

```
Laptop/OneDrive/one_eye3\0077_one_eye_x-8.421_y12.226.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0078_one_eye_x2.568_y5.012.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0079_one_eye_x39.437_y-8.23.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0080_one_eye_x5.669_y-13.268.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0081_one_eye_x-39.12_y-14.046.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0082_one_eye_x-23.113_y13.09.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0083_one_eye_x28.839_y6.391.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0084_one_eye_x-19.376_y-9.931.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0085_one_eye_x-18.373_y8.152.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0086_one_eye_x22.982_y1.493.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0087_one_eye_x33.782_y7.079.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0088_one_eye_x9.084_y-0.186.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0089_one_eye_x-38.48_y13.648.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0090_one_eye_x-18.114_y-14.98.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0091_one_eye_x36.123_y4.876.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0092_one_eye_x12.778_y9.848.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
```

```
Laptop/OneDrive/one_eye3\0093_one_eye_x-38.247_y-6.941.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0094_one_eye_x-32.803_y14.234.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0095_one_eye_x-18.649_y-9.163.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0096_one_eye_x-9.808_y11.51.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0097_one_eye_x-13.848_y-1.246.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0098_one_eye_x-16.959_y-10.147.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
C:/Users/Kevin-Laptop/OneDrive/Inkscape.lnk -e C:/Users/Kevin-
Laptop/OneDrive/one_eye3\0099_one_eye_x12.18_y-13.694.png C:/users/Kevin-
Laptop/OneDrive/temp.svg
```

With this code, we were able to generate a brand new set of images that were close to the original image set in form but with a few slight changes.

In the original set of images, the eye color was brown but now we've created a whole new set of images of which the eye colors are blue.

To load in the new set of images, we are going to modify our original setup function.

[81]:
```python
def setup2(datafolder,columns_to_use):

    datafile    = join(datafolder, datafolder + ".txt")

    df = pd.read_csv(datafile, sep =" ")[columns_to_use].copy()
    attrix = pd.DataFrame(df).to_numpy()
    print("Loading images...")
    imagefiles = sorted(glob(join(datafolder,"*.png")))
    images = np.array([cv2.imread(imagefile)[:,:,:3] for imagefile in␣
 ↪imagefiles]) #slice off opacity layer
    images = images / 255.0

    return attrix,images
```

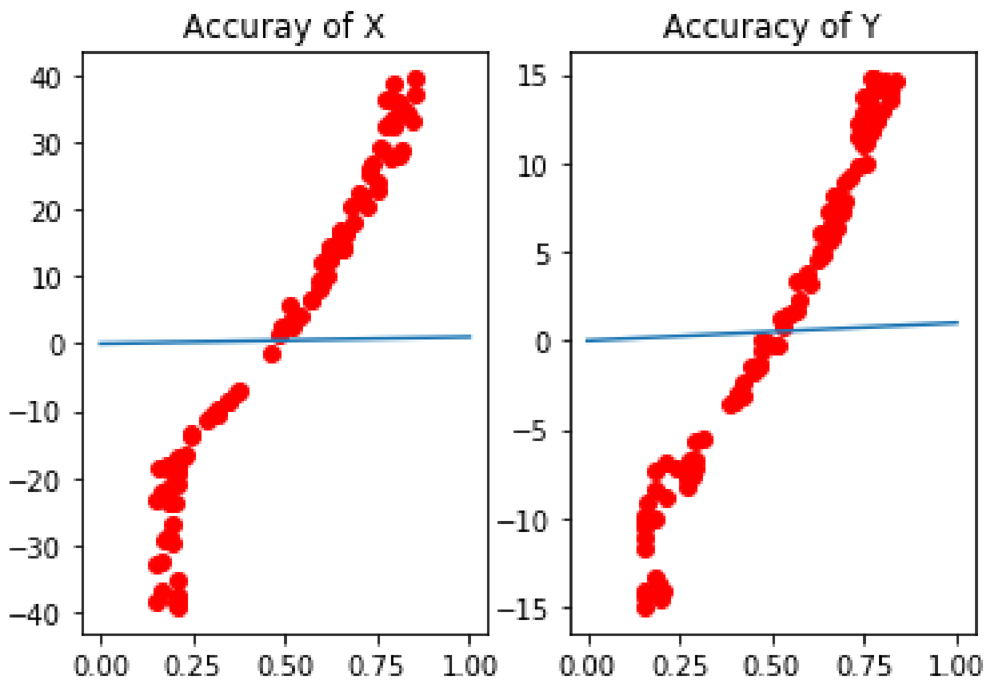
[88]:
```python
testa2,testi2 = setup2("one_eye3",["x","y"])
preds2 = model.predict(testi2)
```

```
Loading images...
```

[89]:
```python
mse(preds2,testa2)
```

```
Error in X: 550.9464378951017
Error in Y: 79.00641457696624
```

[90]:
```
graphrep(preds2,testa2)
```



6 Predicting Image With Two Eyes

Now that we have tested the model's predicting ability with an altered one eye image, we move on to testing the model with two eye images. We will essentially do the same as we have done in the past with the one eye images which is to build the model, load in the images, train the model, and test the images.

[27]:
```
tra3,testa3,tri3,testi3,mina3,maxa3,rangea3 =␣
 ↪setup("two_eyes",["crossx","swivelx","y"])

model3 = create_cnn(3,137,24, 3, regress=True)
opt3 = Adam(lr=1e-3, decay=1e-3 / 200)
model3.compile(loss="mean_squared_error", optimizer=opt)

# training the model
print("Training model...")
model3.fit(tri3, tra3, validation_data=(testi3, testa3),
        epochs=100, batch_size=8)
```

```
model3.save("model.h5")

print("Done")
```

```
Loading images...
Training model...
Train on 1500 samples, validate on 500 samples
Epoch 1/100
1500/1500 [==============================] - 12s 8ms/sample - loss: 0.1111 -
val_loss: 0.1033
Epoch 2/100
1500/1500 [==============================] - 11s 7ms/sample - loss: 0.0632 -
val_loss: 0.0709
Epoch 3/100
1500/1500 [==============================] - 10s 7ms/sample - loss: 0.0577 -
val_loss: 0.0465
Epoch 4/100
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0547 -
val_loss: 0.0458
Epoch 5/100
1500/1500 [==============================] - 10s 7ms/sample - loss: 0.0510 -
val_loss: 0.0273
Epoch 6/100
1500/1500 [==============================] - 12s 8ms/sample - loss: 0.0514 -
val_loss: 0.0431
Epoch 7/100
1500/1500 [==============================] - 11s 7ms/sample - loss: 0.0493 -
val_loss: 0.0259
Epoch 8/100
1500/1500 [==============================] - 12s 8ms/sample - loss: 0.0493 -
val_loss: 0.0240
Epoch 9/100
1500/1500 [==============================] - 8s 6ms/sample - loss: 0.0473 -
val_loss: 0.0935
Epoch 10/100
1500/1500 [==============================] - 10s 7ms/sample - loss: 0.0458 -
val_loss: 0.0314
Epoch 11/100
1500/1500 [==============================] - 11s 7ms/sample - loss: 0.0451 -
val_loss: 0.0201
Epoch 12/100
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0448 -
val_loss: 0.0455
Epoch 13/100
1500/1500 [==============================] - 10s 7ms/sample - loss: 0.0454 -
val_loss: 0.0229
Epoch 14/100
```

```
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0443 -
val_loss: 0.0153
Epoch 15/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0440 -
val_loss: 0.0168
Epoch 16/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0433 -
val_loss: 0.0191
Epoch 17/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0433 -
val_loss: 0.0170
Epoch 18/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0396 -
val_loss: 0.0564
Epoch 19/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0432 -
val_loss: 0.0869
Epoch 20/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0421 -
val_loss: 0.0128
Epoch 21/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0414 -
val_loss: 0.0173
Epoch 22/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0421 -
val_loss: 0.0926
Epoch 23/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0417 -
val_loss: 0.0179
Epoch 24/100
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0408 -
val_loss: 0.0193
Epoch 25/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0396 -
val_loss: 0.0129
Epoch 26/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0404 -
val_loss: 0.0101
Epoch 27/100
1500/1500 [==============================] - 8s 6ms/sample - loss: 0.0402 -
val_loss: 0.0335
Epoch 28/100
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0405 -
val_loss: 0.0144
Epoch 29/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0402 -
val_loss: 0.0215
Epoch 30/100
```

```
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0395 -
val_loss: 0.0143
Epoch 31/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0387 -
val_loss: 0.0226
Epoch 32/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0399 -
val_loss: 0.0128
Epoch 33/100
1500/1500 [==============================] - 10s 6ms/sample - loss: 0.0404 -
val_loss: 0.0133
Epoch 34/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0410 -
val_loss: 0.0327
Epoch 35/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0402 -
val_loss: 0.0100
Epoch 36/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0406 -
val_loss: 0.0145
Epoch 37/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0397 -
val_loss: 0.0127
Epoch 38/100
1500/1500 [==============================] - 8s 6ms/sample - loss: 0.0397 -
val_loss: 0.0187
Epoch 39/100
1500/1500 [==============================] - 8s 6ms/sample - loss: 0.0401 -
val_loss: 0.0136
Epoch 40/100
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0386 -
val_loss: 0.0128
Epoch 41/100
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0398 -
val_loss: 0.0368
Epoch 42/100
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0387 -
val_loss: 0.0185
Epoch 43/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0408 -
val_loss: 0.0135
Epoch 44/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0413 -
val_loss: 0.0106
Epoch 45/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0397 -
val_loss: 0.0154
Epoch 46/100
```

```
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0390 -
val_loss: 0.0092
Epoch 47/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0382 -
val_loss: 0.0222
Epoch 48/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0406 -
val_loss: 0.0154
Epoch 49/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0388 -
val_loss: 0.0226
Epoch 50/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0407 -
val_loss: 0.0360
Epoch 51/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0403 -
val_loss: 0.0407
Epoch 52/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0398 -
val_loss: 0.0118
Epoch 53/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0389 -
val_loss: 0.0161
Epoch 54/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0380 -
val_loss: 0.0092
Epoch 55/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0395 -
val_loss: 0.0395
Epoch 56/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0386 -
val_loss: 0.0094
Epoch 57/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0389 -
val_loss: 0.0255
Epoch 58/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0393 -
val_loss: 0.0171
Epoch 59/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0391 -
val_loss: 0.0219
Epoch 60/100
1500/1500 [==============================] - 11s 8ms/sample - loss: 0.0392 -
val_loss: 0.0091
Epoch 61/100
1500/1500 [==============================] - 11s 7ms/sample - loss: 0.0400 -
val_loss: 0.0957
Epoch 62/100
```

```
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0391 -
val_loss: 0.0112
Epoch 63/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0392 -
val_loss: 0.0114
Epoch 64/100
1500/1500 [==============================] - 9s 6ms/sample - loss: 0.0390 -
val_loss: 0.0158
Epoch 65/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0385 -
val_loss: 0.0097
Epoch 66/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0393 -
val_loss: 0.0305
Epoch 67/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0386 -
val_loss: 0.0505
Epoch 68/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0388 -
val_loss: 0.0182
Epoch 69/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0371 -
val_loss: 0.0120
Epoch 70/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0391 -
val_loss: 0.0167
Epoch 71/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0387 -
val_loss: 0.0188
Epoch 72/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0374 -
val_loss: 0.0127
Epoch 73/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0387 -
val_loss: 0.0525
Epoch 74/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0376 -
val_loss: 0.0094
Epoch 75/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0378 -
val_loss: 0.0244
Epoch 76/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0389 -
val_loss: 0.0151
Epoch 77/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0375 -
val_loss: 0.0175
Epoch 78/100
```

```
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0388 -
val_loss: 0.0344
Epoch 79/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0380 -
val_loss: 0.0130
Epoch 80/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0375 -
val_loss: 0.0113
Epoch 81/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0394 -
val_loss: 0.0090
Epoch 82/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0381 -
val_loss: 0.0163
Epoch 83/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0397 -
val_loss: 0.0273
Epoch 84/100
1500/1500 [==============================] - 8s 6ms/sample - loss: 0.0392 -
val_loss: 0.0205
Epoch 85/100
1500/1500 [==============================] - 10s 6ms/sample - loss: 0.0380 -
val_loss: 0.0175
Epoch 86/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0387 -
val_loss: 0.0116
Epoch 87/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0402 -
val_loss: 0.0087
Epoch 88/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0383 -
val_loss: 0.0454
Epoch 89/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0379 -
val_loss: 0.0364
Epoch 90/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0378 -
val_loss: 0.0111
Epoch 91/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0375 -
val_loss: 0.0309
Epoch 92/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0383 -
val_loss: 0.0119
Epoch 93/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0392 -
val_loss: 0.0169
Epoch 94/100
```

```
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0385 -
val_loss: 0.0132
Epoch 95/100
1500/1500 [==============================] - 7s 4ms/sample - loss: 0.0388 -
val_loss: 0.0133
Epoch 96/100
1500/1500 [==============================] - 7s 4ms/sample - loss: 0.0394 -
val_loss: 0.0107
Epoch 97/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0389 -
val_loss: 0.0294
Epoch 98/100
1500/1500 [==============================] - 8s 5ms/sample - loss: 0.0366 -
val_loss: 0.0152
Epoch 99/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0393 -
val_loss: 0.0106
Epoch 100/100
1500/1500 [==============================] - 7s 5ms/sample - loss: 0.0386 -
val_loss: 0.0086
```

```
       ␣
 ↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
 ↪last)

       <ipython-input-27-13da8148e34e> in <module>
        12 model3.save("model.h5")
        13
    ---> 14 preds3 = model3.predict(testi)
        15
        16 print("Done")


       ␣
 ↪~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training.py␣
 ↪in predict(self, x, batch_size, verbose, steps, callbacks, max_queue_size,␣
 ↪workers, use_multiprocessing)
       907         max_queue_size=max_queue_size,
       908         workers=workers,
    --> 909         use_multiprocessing=use_multiprocessing)
       910
       911   def reset_metrics(self):
```

```
    ␣
↪~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training_v2.
↪py in predict(self, model, x, batch_size, verbose, steps, callbacks, **kwargs)
    460     return self._model_iteration(
    461         model, ModeKeys.PREDICT, x=x, batch_size=batch_size,␣
↪verbose=verbose,
--> 462         steps=steps, callbacks=callbacks, **kwargs)
    463 
    464 


    ␣
↪~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training_v2.
↪py in _model_iteration(self, model, mode, x, y, batch_size, verbose,␣
↪sample_weight, steps, callbacks, **kwargs)
    394           sample_weights=sample_weight,
    395           steps=steps,
--> 396           distribution_strategy=strategy)
    397       total_samples = _get_total_number_of_samples(adapter)
    398       use_sample = total_samples is not None


    ␣
↪~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training_v2.
↪py in _process_inputs(model, x, y, batch_size, epochs, sample_weights,␣
↪class_weights, shuffle, steps, distribution_strategy, max_queue_size, workers,␣
↪use_multiprocessing)
    592         batch_size=batch_size,
    593         check_steps=False,
--> 594         steps=steps)
    595   adapter = adapter_cls(
    596       x,


    ␣
↪~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training.py␣
↪in _standardize_user_data(self, x, y, sample_weight, class_weight, batch_size,␣
↪check_steps, steps_name, steps, validation_split, shuffle,␣
↪extract_tensors_from_dataset)
   2470           feed_input_shapes,
   2471           check_batch_axis=False,  # Don't enforce the batch size.
-> 2472           exception_prefix='input')
   2473 
   2474     # Get typespecs for the input data and sanitize it if necessary.
```

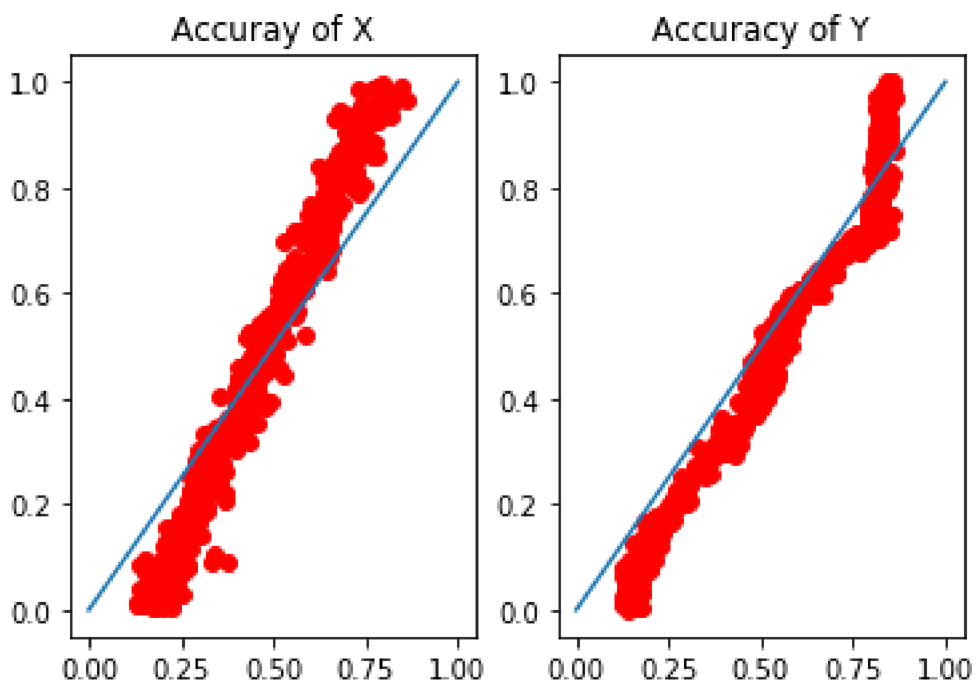
```
        ␣
↪~\Anaconda3\lib\site-packages\tensorflow_core\python\keras\engine\training_utils.
↪py in standardize_input_data(data, names, shapes, check_batch_axis,␣
↪exception_prefix)
        572                              ': expected ' + names[i] + ' to have␣
↪shape ' +
        573                              str(shape) + ' but got array with shape␣
↪' +
    --> 574                              str(data_shape))
        575     return data
        576


        ValueError: Error when checking input: expected input_3 to have shape␣
↪(24, 137, 3) but got array with shape (48, 90, 3)
```

[31]:
```
preds3 = model3.predict(testi3)
mse(preds3,testa3)
```

```
Error in X: 0.012654638523314521
Error in Y: 0.005647550275677117
```

[33]:
```
graphrep(preds3,testa3)
```



We can see by comparing the error and graphical representation of this model's prediction to the prediction of the one_eye model, that with the same number of epochs, the prediction of this model is just slightly less accurate than the original model. This is probably due to the fact that we have an extra parameter for the model to consider. Even though this model's predictions were slightly less accurate, we can still see that with a completely new set of data and parameters to predict on, the model's prediction is still on track for the amount of time trained.

7 Closing

Through our exploration of the convolutional neural network, there are many fascinating takeaways that we can make. We can conclude that we can expect to have a less accurate prediction from the model if we give it a short period of training and when given a longer training period, the model's predictions become increasingly closer to the actual (x,y) coordinates. This really speaks levels on the fact that this is true of humans as well and when we consider that the intention of the neural network/machine learning model is to try to mimic the human brain as best we can, this is a promising outcome.

We also notice that when testing the model on a different set of data than the one it was trained with, the model had difficulty accurately predicting the (x,y) coordinates even though it was trained for 100 epochs. We can then conclude that without a similarity in training data, the model may struggle to perform well.

We notice when testing a convolutional model on a set of data with two eyes, it seems to make no difference as long as we train the model enough to perform the needed task, it will eventually be able to accomplish it to satisfaction.

With the progression of these artificial intelligence tools, we not only getting closer and closer to replicating our brains and its functions, but we also establish many opportunities to have significant change in the world. A world where machines are as intelligent and as capable as us humans. One such reality would at the very least help us lead more comfortable and convenient lives knowing that machines can help accomplish tasks without the disadvantages humans need to deal with. Disadvantages like sleeping, fatigue, hunger, discontent, and susceptibility to entropy. We may be a long ways off from this reality but study and progression into these powerful tools greatly push us towards that hypothetical reality.

8 References

https://en.wikipedia.org/wiki/Mean_squared_error
https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9
https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
https://www.geeksforgeeks.org/confusion-matrix-machine-learning/