In [1]:
```
! pip3 install Pillow
```

```
Requirement already satisfied: Pillow in /Library/Frameworks/Python.framework/Versions/3.6/lib/
python3.6/site-packages (8.0.1)
WARNING: You are using pip version 19.1.1, however version 20.2.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [2]:
```
import csv
import os
from PIL import Image
import numpy as np
from sklearn.metrics import accuracy_score
import random
import matplotlib.pyplot as plt
```

In [3]:
```
test_down_labels = []
with open('downgesture_test.list.txt') as f:
    file = csv.reader(f)
    for line in file:
        test_down_labels.append(line[0])
len(test_down_labels)
```

Out[3]: 83

In [4]:
```
train_down_labels = []
with open('downgesture_train.list.txt') as f:
    file = csv.reader(f)
    for line in file:
        train_down_labels.append(line[0])
len(train_down_labels)
```

Out[4]: 184

In [5]:
```
data_path = '/Users/leo/Desktop/Analytics/2020 Fall/DSCI552/HW5_1028due/gestures'
```

read data

In [6]:
```
def read_imgs(data_path, labels):
    all_files = os.listdir(data_path)
    imgs_data = []
    imgs_label = []
    for folder in all_files:
        folder_path = os.path.join(data_path, folder)
        # check if it is foler
        if os.path.isdir(folder_path):
            all_imgs = os.listdir(folder_path)
            for img_name in all_imgs:
                name = 'gestures/'+folder+'/'+img_name
                if name in labels:
                    img_path = os.path.join(folder_path,img_name)
                    img  = Image.open(img_path)
                    img_bytes = np.array(img)
                    img_gray_scale = img_bytes/255
                    imgs_data.append(img_gray_scale.flatten())
                    if 'down' in name:
                        imgs_label.append(1)
                    else:
                        imgs_label.append(0)
    return np.array(imgs_data), np.array(imgs_label)
```

In [7]:
```
training_imgs_data, training_imgs_label = read_imgs(data_path, train_down_labels)
ytrain = training_imgs_label
Xtrain = np.concatenate((np.ones((len(training_imgs_data),1)),
                         training_imgs_data), axis=1)
```

Forward Propagation

In [8]:
```
# Initialize a network
def initialize_NN(n_inputs, n_hidden, n_outputs):
    network = []
    np.random.seed(0)
    hidden_layer = np.random.randint(-100,100, size=(n_inputs+1,n_hidden))/10000
    network.append(hidden_layer)
    np.random.seed(0)
    output_layer = np.random.randint(-100,100, size=(n_hidden+1,n_outputs))/10000
    network.append(output_layer)
    return network
```

In [9]:
```
def calculate_activation(w, inputs):
    s = inputs.dot(w)
    return s
```

In [10]:
```
def sigmoid(s):
    return 1/(1+np.exp(-s))
```

In [11]:
```
# Forward propagate input to a network output
def forward_propagate(network, row):
    inputs = row
    output_list = []
    for layer in network:
        s = inputs.dot(layer)
        new_inputs = sigmoid(s)
        output_list.append(new_inputs)
        inputs = np.insert(new_inputs,0,1)
    return new_inputs, output_list
```

Back Propagate

In [16]:
```
def backpropagate(yhat, output_list, xi, yi, NN):
    delta = 2*(yhat-yi)*(yhat*(1-yhat))
    deltaj = (yhat*(1-yhat))*NN[1]*delta
    deltaj = np.delete(deltaj,0)
    NN[0] = NN[0] - 0.1*xi.reshape(-1,1)*deltaj
    xj = np.insert(output_list[0],0,1)
    NN[1] = NN[1] - 0.1*delta*xj.reshape(-1,1)
    return NN
```

Model train

In [17]:
```
def train(X ,y, epochs):
    NN = initialize_NN(960,100,1)
    for epoch in range(epochs):
        for xi, yi in zip(X, y):
            yhat, output_list = forward_propagate(NN, xi)
            NN = backpropagate(yhat, output_list, xi, yi, NN)
            if epoch == epochs-1:
                print(yhat, yi)
    return NN
```

In [18]:
```
NN = train(Xtrain ,ytrain, 1000)
```

...

Predict test data

In [19]:
```
def predict(NN, Xtest, ytest, threshold):
    y_pred_list = []
    for xi, yi in zip(Xtest, ytest):
        yhat, output_list = forward_propagate(NN, xi)
        if yhat > threshold:
            y_pred_list.append(1)
        else:
            y_pred_list.append(0)
    acc = accuracy_score(ytest, y_pred_list)
    return y_pred_list, acc
```

In [20]:
```
test_imgs_data, test_imgs_label = read_imgs(data_path, test_down_labels)
ytest = test_imgs_label
Xtest = np.concatenate((np.ones((len(test_imgs_data),1)), test_imgs_data), axis=1)
y_pred_list, acc = predict(NN, Xtest, ytest, 0.5)
np.array(y_pred_list), acc
```

Out[20]:
```
(array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
        0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
        0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0]), 0.8148148148148148)
```

In [ ]: