ReadMe for Project_2

Performance Comparison:

	Training	Testing
Logistic Regression	Accuracy: 50.94347826 Time: 2007.46 seconds	Accuracy: 50.291666 Time: 801.09 seconds
Pytorch	Accuracy: 99.9347 Time: 9431.820 seconds	Accuracy: 9431.82 Time: 258.94 seconds

Logistic Regression Classification Model

```
from numpy import log, dot, e
from numpy.random import rand
batch_size = 1000
epochs = 50
height = 50
width = 50
class LogisticRegression:
    def __init__(self,input_size, lr=0.05):
        self.weights = rand(input_size)
        self.lr = lr
    def sigmoid(self, z): return 1 / (1 + e**(-z))

    def fit(self, X, y):
        N = len(X)
        # Predicting with sigmoid function
        y_hat = self.sigmoid(dot(X, self.weights))
        # Updating Weights using Gradient Descent
        self.weights -= self.lr * (dot(X.T,  y_hat - y) / N  )


    def predict(self, X):
        # Predicting with sigmoid function
        z = dot(X, self.weights)
        # Returning binary result
        return [1 if i > 0.5 else 0 for i in self.sigmoid(z)]

```

Training

```
print("Loading training data: ")
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        'Dataset/train',
        target_size=(height,width),
        batch_size=batch_size,
        color_mode="grayscale",
        shuffle=True,
        class_mode='binary')
batch_x,batch_y = train_generator.next()
batch_x = batch_x.reshape(batch_size,-1)
lr = LogisticRegression(batch_x.shape[1])
start_time = time.time()
print("Training model:")
for i in range(epochs):
    accuracy = 0
    for j in range(int(train_generator.samples / batch_size)):
        #print(f"Training Epoch: {i} Batch: {j}")
        lr.fit(batch_x,batch_y)
        batch_x,batch_y = train_generator.next()
        batch_x = batch_x.reshape(batch_size,-1)
        accuracy += accuracy_score(batch_y,lr.predict(batch_x))
    print(f"Epoch {i}/{epochs} Training Accuracy: {(accuracy / int(train_generator.samples / batch_size)) * 100}")
print(f"Training completed in {time.time() - start_time} seconds")
```

```
Epoch 30/50 Training Accuracy: 50.92608695652173
Epoch 31/50 Training Accuracy: 50.978260869565204
Epoch 32/50 Training Accuracy: 51.10869565217392
Epoch 33/50 Training Accuracy: 50.99130434782608
Epoch 34/50 Training Accuracy: 50.83478260869565
Epoch 35/50 Training Accuracy: 51.15652173913045
Epoch 36/50 Training Accuracy: 50.69565217391304
Epoch 37/50 Training Accuracy: 50.534782608695636
Epoch 38/50 Training Accuracy: 50.87826086956523
Epoch 39/50 Training Accuracy: 51.265217391304354
Epoch 40/50 Training Accuracy: 51.23478260869565
Epoch 41/50 Training Accuracy: 51.75652173913045
Epoch 42/50 Training Accuracy: 51.31739130434782
Epoch 43/50 Training Accuracy: 51.060869565217395
Epoch 44/50 Training Accuracy: 50.77391304347827
Epoch 45/50 Training Accuracy: 50.35652173913042
Epoch 46/50 Training Accuracy: 51.35217391304349
Epoch 47/50 Training Accuracy: 50.76956521739131
Epoch 48/50 Training Accuracy: 50.947826086956525
Epoch 49/50 Training Accuracy: 50.94347826086958
Training completed in 2007.4684281349182 seconds
```

Testing

```
start_time = time.time()
print("Loading test data: ")
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
        'Dataset/test',
        target_size=(height,width),
        batch_size=batch_size,
        color_mode="grayscale",
        shuffle=True,
        class_mode='binary')
test_x,test_y = test_generator.next()
test_x = test_x.reshape(batch_size,-1)

print("Testing model:")
test_accuracy = 0
for j in range(int(test_generator.samples / batch_size)):
    #print(f"Testing Batch: {j}")
    test_accuracy += accuracy_score(test_y,lr.predict(test_x))
    test_x,test_y = test_generator.next()
    test_x = test_x.reshape(batch_size,-1)

print(f"Testing accuracy: {(test_accuracy / int(test_generator.samples / batch_size)) * 100}")
print(f"Testing completed in {time.time() - start_time} seconds")
```

```
Loading test data:
Found 12500 images belonging to 2 classes.
Testing model:
Testing accuracy: 50.29166666666666
Testing completed in 801.0907106399536 seconds
```

Pytorch Api Classification Model (NN - CNN)

:

```
import torch.nn as nn
import torch.nn.functional as F
batch_size = 100
epochs = 150
height = 50
width = 50

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.fc1 = nn.Linear(height * width, 1024)
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, 2)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

:

```
import torch.optim as optim

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

Training

```
print("Loading training data: ")
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        'Dataset/train',
        target_size=(height,width),
        batch_size=batch_size,
        color_mode="grayscale",
        shuffle=True,
        class_mode='binary')
batch_x,batch_y = train_generator.next()
batch_x = batch_x.reshape(batch_size,-1)
#batch_y = tf.keras.utils.to_categorical(batch_y, 2)
inputs, labels = batch_x,batch_y
def acc(y_true,y_pred):
    count = 0
    for i in range(len(y_true)):
        if(y_true[i] == np.argmax(y_pred[i])):
            count +=1
    return count

start_time = time.time()
print("Training Model: ")
for epoch in range(epochs):  # loop over the dataset multiple times
    accuracy = 0
    for j in range(int(train_generator.samples / batch_size)):
        # zero the parameter gradients
        optimizer.zero_grad()
        # forward + backward + optimize
        outputs = net(torch.from_numpy(inputs))
        outputs_temp = outputs
        loss = criterion(outputs, torch.from_numpy(labels).long() )
        loss.backward()
        optimizer.step()
        accuracy = accuracy + acc(batch_y,outputs_temp.detach().numpy())
        #print(f"Training Epoch: {i} Batch: {j}")
        batch_x,batch_y = train_generator.next()
        batch_x = batch_x.reshape(batch_size,-1)
        inputs, labels = batch_x,batch_y
    print(f"Epoch {epoch}/{epochs} Training Accuracy: {accuracy / int(train_generator.samples / batch_size)}")
print(f'Training competed in {time.time() - start_time}')
```

```
Epoch 134/150 Training Accuracy: 98.6608695652174
Epoch 135/150 Training Accuracy: 99.14347826086957
Epoch 136/150 Training Accuracy: 99.23478260869565
Epoch 137/150 Training Accuracy: 99.37826086956522
Epoch 138/150 Training Accuracy: 99.52173913043478
Epoch 139/150 Training Accuracy: 99.59565217391304
Epoch 140/150 Training Accuracy: 99.70434782608696
Epoch 141/150 Training Accuracy: 99.58260869565217
Epoch 142/150 Training Accuracy: 99.7695652173913
Epoch 143/150 Training Accuracy: 99.81739130434782
Epoch 144/150 Training Accuracy: 99.80434782608695
Epoch 145/150 Training Accuracy: 99.91739130434783
Epoch 146/150 Training Accuracy: 99.88695652173914
Epoch 147/150 Training Accuracy: 99.89565217391305
Epoch 148/150 Training Accuracy: 99.94347826086957
Epoch 149/150 Training Accuracy: 99.93478260869566
Training competed in 9431.820527076721
```

Testing

```
[58]:
```

```python
print("Loading test data: ")
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
        'Dataset/test',
        target_size=(height,width),
        batch_size=batch_size,
        color_mode="grayscale",
        shuffle=True,
        class_mode='binary')
test_x,test_y = test_generator.next()
test_x = test_x.reshape(batch_size,-1)
start_time = time.time()
print("Testing model:")
test_accuracy = 0
for j in range(int(test_generator.samples / batch_size)):
    #print(f"Testing Batch: {j}")
    optimizer.zero_grad()
    # forward + backward + optimize
    outputs = net(torch.from_numpy(test_x))
    test_accuracy = test_accuracy + acc(test_y,outputs.detach().numpy())
    test_x,test_y = test_generator.next()
    test_x = test_x.reshape(batch_size,-1)

print(f"Testing accuracy: {test_accuracy / int(test_generator.samples / batch_size)}")
print(f'Testing competed in {time.time() - start_time}')
```

```
Loading test data:
Found 12500 images belonging to 2 classes.
Testing model:
Testing accuracy: 99.952
Testing competed in 258.94079184532166
```