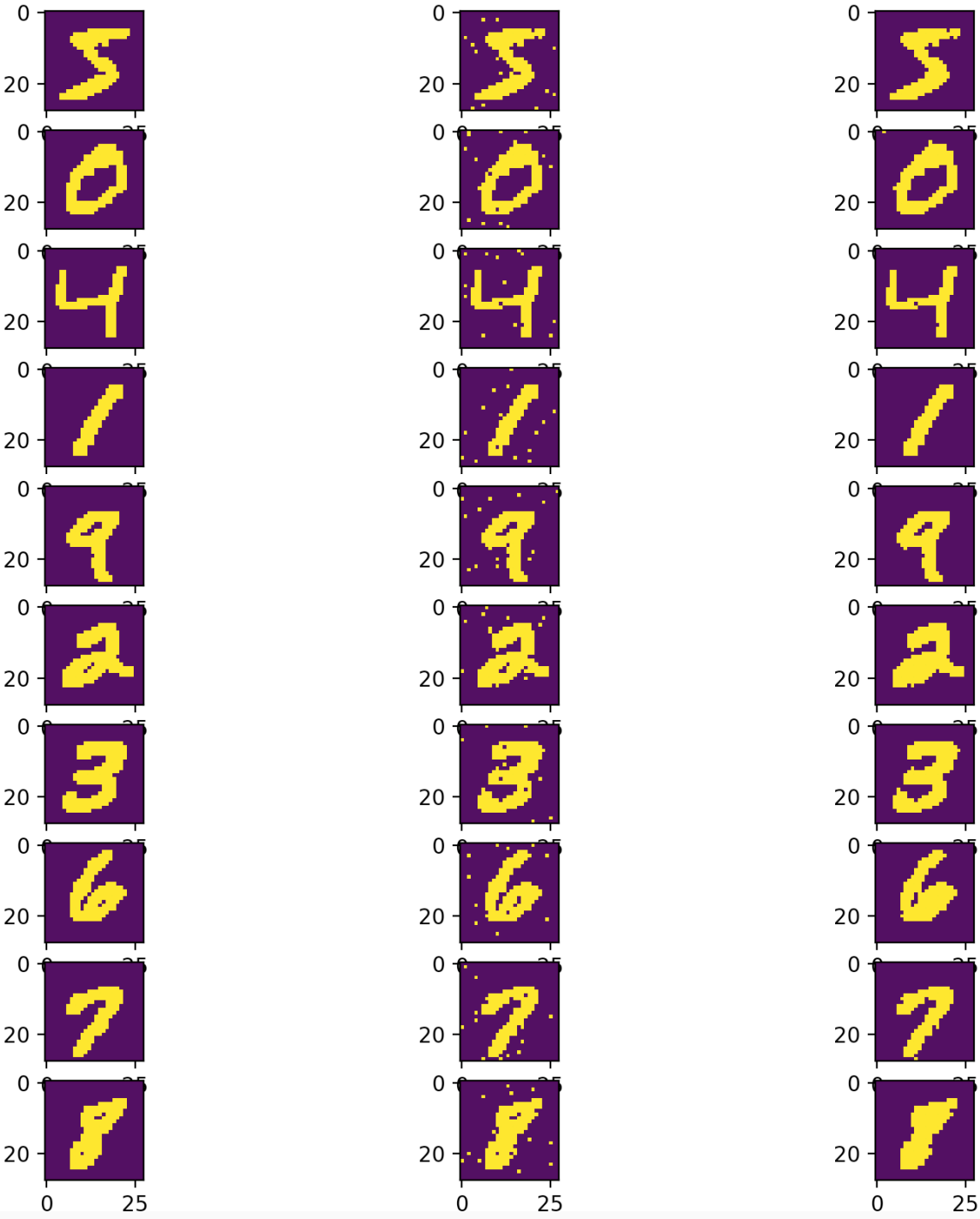
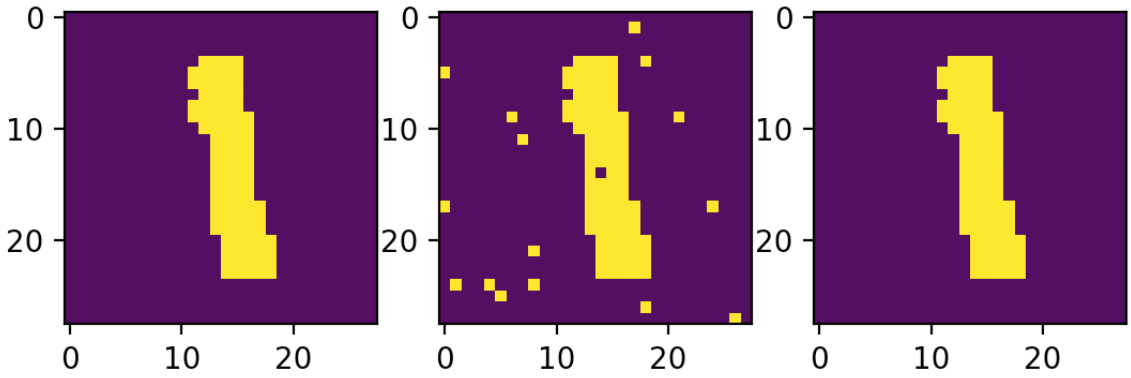
1. Overall Accuracy : 0.9946020408163212

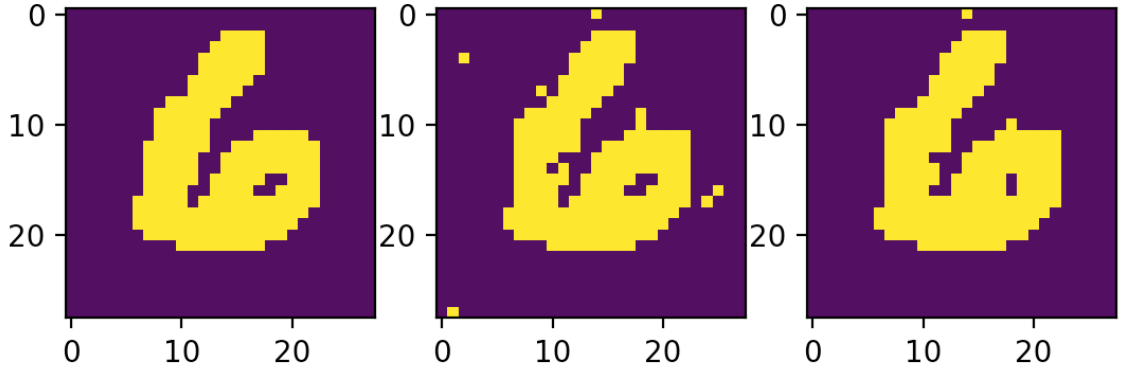
2. 30 sample images



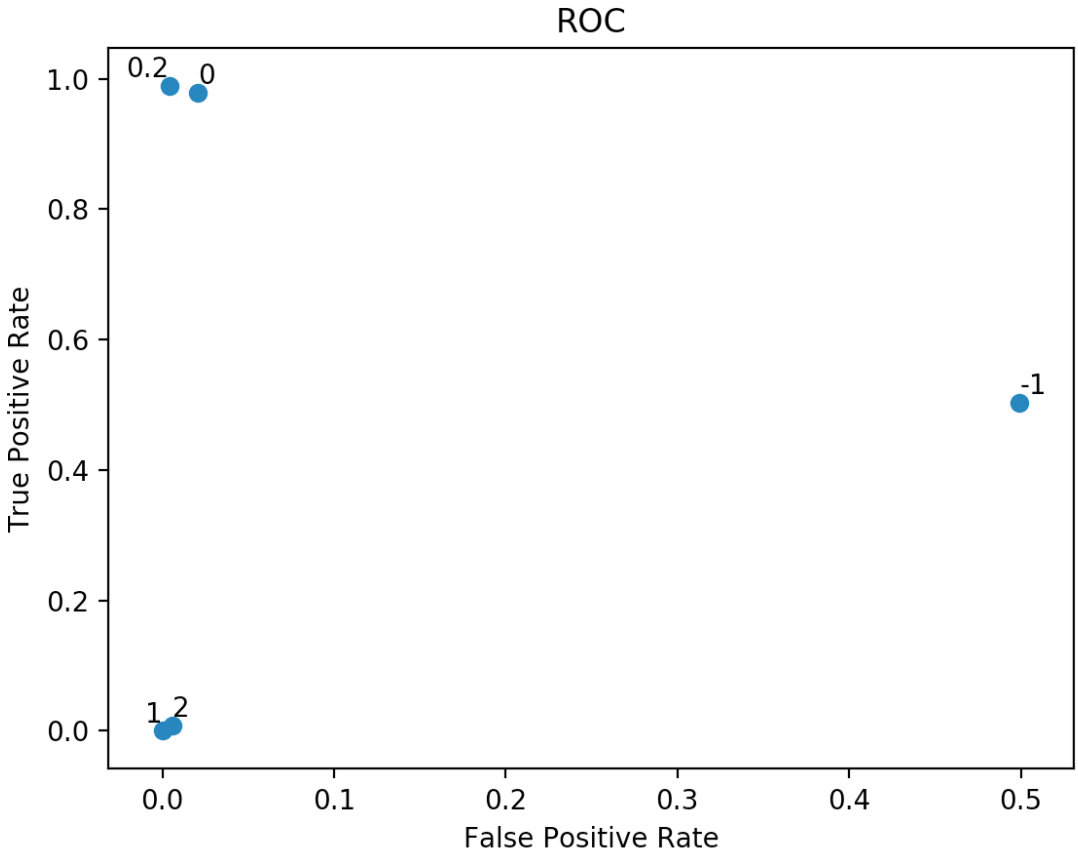
3. Best reconstruction



4. Worst reconstruction



5. Roc Curve


ROC
True Positive Rate
False Positive Rate
0.2
0
-1
1
2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.1
0.2
0.3
0.4
0.5

6. Code

Add noise:

```
binary_train = []
for image in train_images:

    binary = np.zeros(image.shape)
    for i in range(len(image)):
        for j in range(len(image[i])):
            if image[i][j] <= 0.5:
                binary[i][j] = -1
            else:
                binary[i][j] = 1
    binary_train.append(binary)


row_range = len(train_images[0])
col_range = len(train_images[0][0])
size = int(row_range * col_range * 0.02)


noise_train = []

for image in binary_train:
    count = 0
    used = []
    noise = copy.deepcopy(image)
    while count <= size:
        row_idx = np.random.randint(0, row_range)
        col_idx = np.random.randint(0, col_range)
        if (row_idx, col_idx) not in used:
            used.append((row_idx, col_idx))
            count += 1
            if noise[row_idx][col_idx] == 1:
                noise[row_idx][col_idx] = -1
            else:
                noise[row_idx][col_idx] = 1
    noise_train.append(np.array(noise))
```

Denoise:

```
for theta in [-1, 0, 0.2, 1, 2]:

    total_accuracy = 0

    min_accuracy = sys.maxsize
    max_accuracy = -sys.maxsize

    min_index = 0
    max_index = 0

    all_digit = []

    all_30 = []

    result = []

    for idx in range(len(noise_train)):

        hidden = np.zeros(noise_train[0].shape)
        for i in range(len(hidden)):
            for j in range(len(hidden[0])):
                hidden[i][j] = 0.5
```

```
        while True:

            orig = copy.deepcopy(hidden)

            for i in range(len(hidden)):
                for j in range(len(hidden[0])):

                    power_sum1 = 0

                    if i - 1 >= 0:
                        power_sum1 += theta * (2 * hidden[i - 1][j] - 1)
                    if i + 1 <= len(hidden) - 1:
                        power_sum1 += theta * (2 * hidden[i + 1][j] - 1)
                    if j - 1 >= 0:
                        power_sum1 += theta * (2 * hidden[i][j - 1] - 1)
                    if j + 1 <= len(hidden[0]) - 1:
                        power_sum1 += theta * (2 * hidden[i][j + 1] - 1)

                    hx = 0.2

                    power_sum2 = hx * noise_train[idx][i][j]

                    top = (math.e)**(power_sum1 + power_sum2)

                    down = (math.e)**(power_sum1 + power_sum2) + \
                           (math.e)**(-power_sum1 - power_sum2)

                    hidden[i][j] = top / down

            distance = np.sum(np.power(hidden - orig, 2))

            if distance < 0.001:
                break
```

7.

Get accuracy

```
    for i in range(len(hidden)):
        for j in range(len(hidden[0])):
            if hidden[i][j] <= 0.5:
                hidden[i][j] = -1
            else:
                hidden[i][j] = 1

    result.append(hidden)


    correct = 0
    for i in range(len(hidden)):
        for j in range(len(hidden[0])):
            if hidden[i][j] == binary_train[idx][i][j]:
                correct += 1


    accuracy = correct / (len(hidden) * len(hidden[0]))

    if accuracy < min_accuracy:
        min_accuracy = accuracy
        min_index = idx

    if accuracy > max_accuracy:
        max_accuracy = accuracy
        max_index = idx

    total_accuracy += accuracy

    if train_labels[idx] not in all_digit:
        all_digit.append(train_labels[idx])
        all_30.append(binary_train[idx])
        all_30.append(noise_train[idx])
        all_30.append(hidden)
```

```
total_accuracy /= 500
print(total_accuracy)

true_label = []
predicted_label = []
for i in range(500):
    true_image = binary_train[i].tolist()
    for l in true_image:
        true_label += l
    predicted_image = result[i].tolist()
    for l in predicted_image:
        predicted_label += l

fpr, tpr, thresholds = metrics.roc_curve(true_label, predicted_label)
print(fpr)
print(tpr)

if len(fpr) == 3:
    x.append(fpr[1])
    y.append(tpr[1])
if len(fpr) == 2:
    x.append(fpr[0])
    y.append(tpr[0])
```

Plot Images

```
min_list = [binary_train[min_index], noise_train[min_index], result[min_index]]
max_list = [binary_train[max_index], noise_train[max_index], result[max_index]]

fig = plt.figure()
ax1 = fig.add_subplot(231)
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)

ax1.imshow(min_list[0])
ax2.imshow(min_list[1])
ax3.imshow(min_list[2])

ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)

ax4.imshow(max_list[0])
ax5.imshow(max_list[1])
ax6.imshow(max_list[2])

plt.show()

##################################################
fig2 = plt.figure(figsize=(10, 10))
for i in range(1, 31):
    ax = fig2.add_subplot(10, 3, i)
    ax.imshow(all_30[i - 1])
plt.show()
```

Roc curve

```
    ##################################################
    fig2 = plt.figure(figsize=(10, 10))
    for i in range(1, 31):
        ax = fig2.add_subplot(10, 3, i)
        ax.imshow(all_30[i - 1])
    plt.show()


labels = [-1, 0, 0.2, 1, 2]

fig, ax = plt.subplots()
ax.scatter( x, y )
texts = []
for i, txt in enumerate(labels):
    texts.append(ax.text(x[i], y[i], txt))
adjust_text(texts)
plt.title('ROC')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```