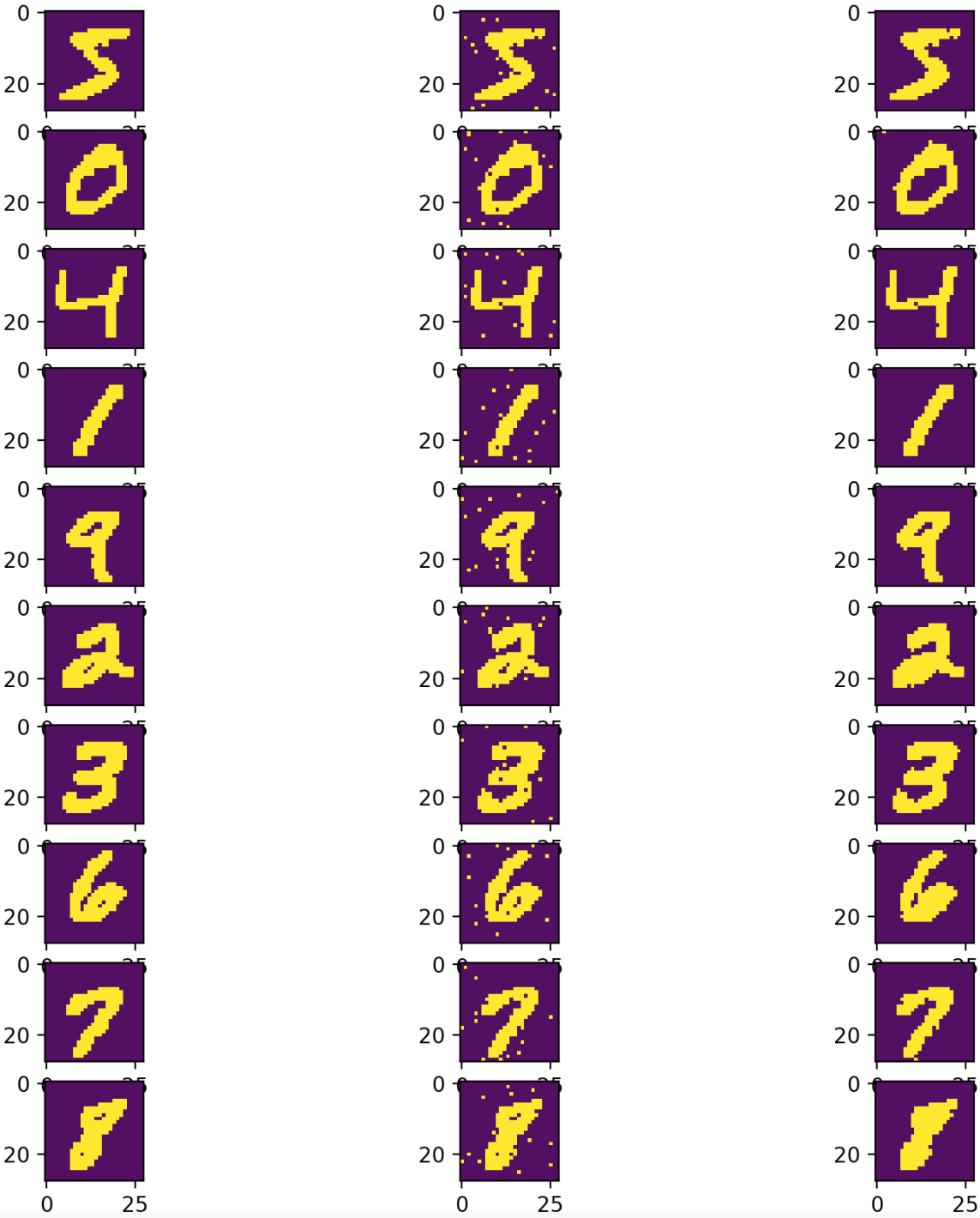
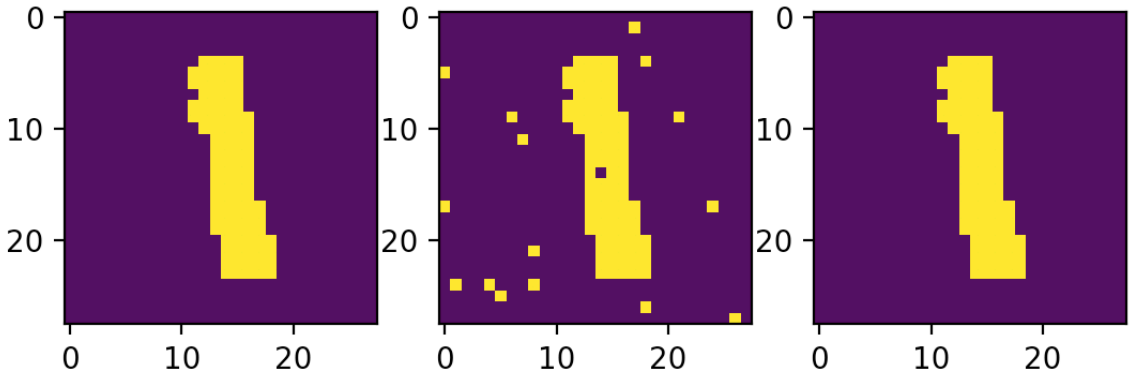
1. Overall Accuracy : 0.9946020408163212

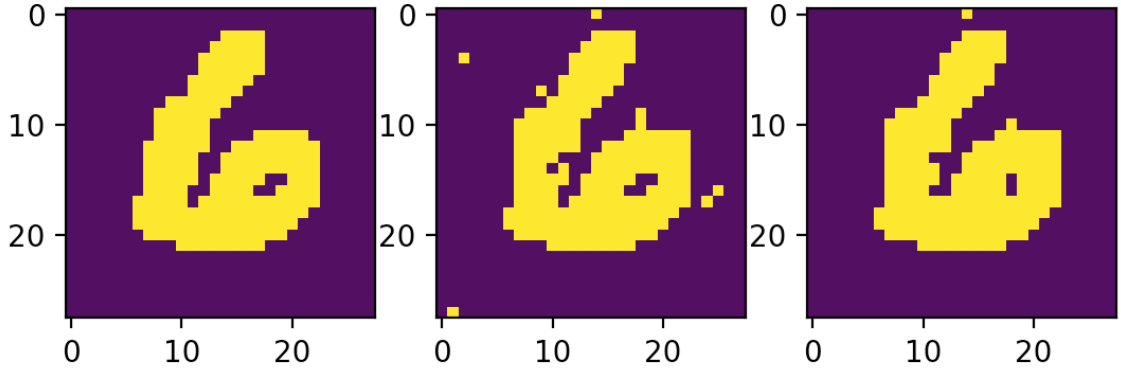
2.



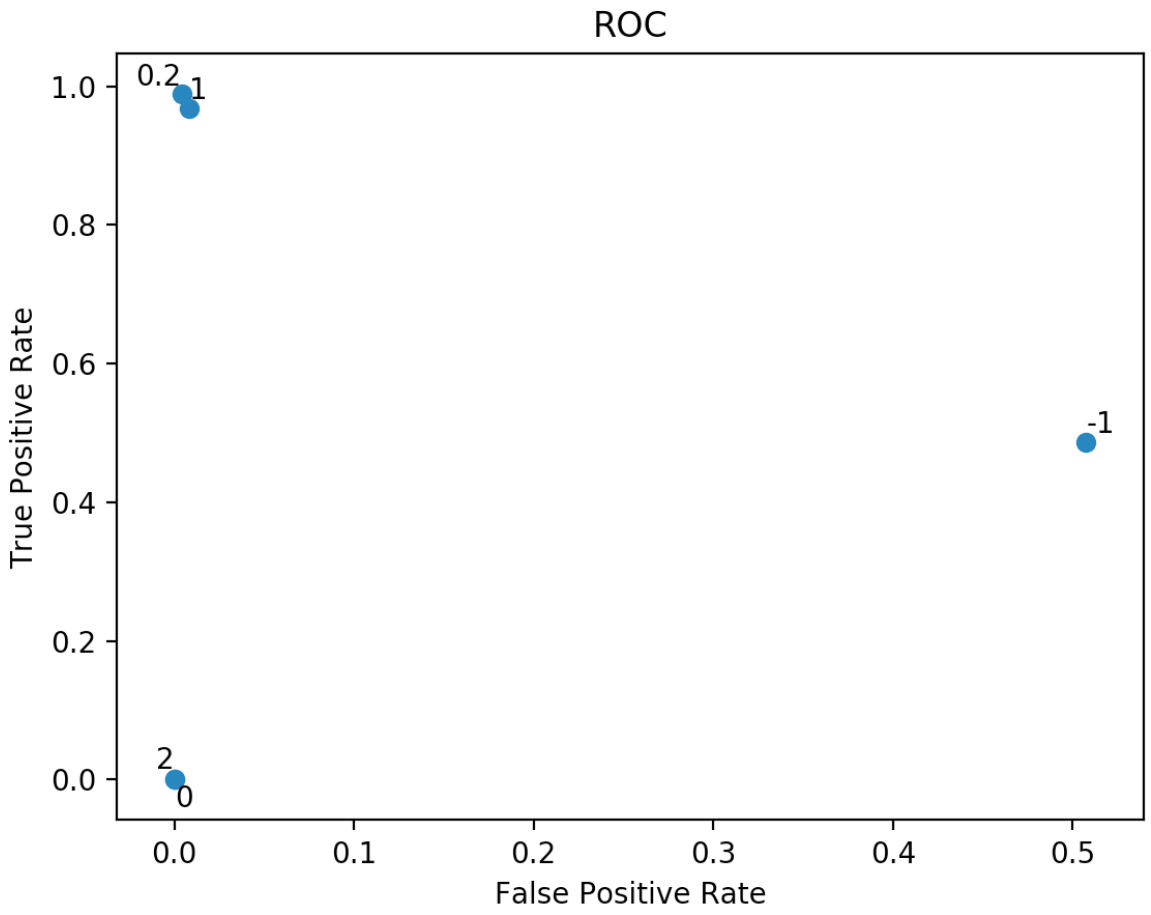
3. Best reconstruction



4. Worst reconstruction



5. Roc Curve


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

6. Code

```
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)

            power_sum2 = theta * 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.

```
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))
```