# Adversarial Robustness-toolbox
# Generate Counter Examples

Fast Gradient Method, Basic Iterative Method, Deep Fool

November 9, 2023

## 1. Label Counts

```
Basic Information:
Total number of images: 70000
Size of each image: (28, 28)
Number of unique labels: 10

Distribution of Labels:
Label 0: 6903
Label 1: 7877
Label 2: 6990
Label 3: 7141
Label 4: 6824
Label 5: 6313
Label 6: 6876
Label 7: 7293
Label 8: 6825
Label 9: 6958

Pixel Intensity Statistics:
Minimum pixel value: 0
Maximum pixel value: 255
Mean pixel value: 33.39
Standard deviation of pixel values: 78.65

Balance Check:
The dataset is not balanced across different labels.
```

Figure 1: Basic Information of Dataset

**Algorithm 1** Evaluation of a Pre-trained Model on MNIST Dataset

**Require:** Pre-trained model path $\mathcal{P}_{\text{model}}$, MNIST dataset
**Ensure:** Test accuracy, Correct examples and labels saved

**function** LOADMODEL($\mathcal{P}_{\text{model}}$)
    $\mathcal{M} \leftarrow$ load model from $\mathcal{P}_{\text{model}}$
    **return** $\mathcal{M}$
**function** LOADMNISTDATA
    $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}}), (\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}}) \leftarrow$ MNIST data
    **return** $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}}), (\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}})$
**function** PREPROCESSDATA($\mathcal{X}$)
    $\mathcal{X} \leftarrow$ reshape and normalize $\mathcal{X}$
    **return** $\mathcal{X}$
**function** ONEHOTENCODE($\mathcal{Y}$)
    $\mathcal{Y}_{\text{encoded}} \leftarrow$ one-hot encode $\mathcal{Y}$
    **return** $\mathcal{Y}_{\text{encoded}}$
**function** EVALUATEMODEL($\mathcal{M}$, $\mathcal{X}_{\text{test}}$, $\mathcal{Y}_{\text{test}}$)
    $\mathcal{P} \leftarrow$ model predictions for $\mathcal{X}_{\text{test}}$
    $\mathcal{L}, \mathcal{A} \leftarrow$ evaluate $\mathcal{M}$ on $\mathcal{X}_{\text{test}}$ and $\mathcal{Y}_{\text{test}}$
    **return** $\mathcal{P}$, $\mathcal{L}$, $\mathcal{A}$
**function** SAVECORRECT($\mathcal{X}_{\text{test}}$, $\mathcal{Y}_{\text{test}}$, $\mathcal{P}$)
    $\mathcal{I}_{\text{correct}} \leftarrow$ indices where $\mathcal{P}$ equals $\mathcal{Y}_{\text{test}}$
    $\mathcal{X}_{\text{correct}} \leftarrow$ examples from $\mathcal{X}_{\text{test}}$ at $\mathcal{I}_{\text{correct}}$
    $\mathcal{Y}_{\text{correct}} \leftarrow$ labels from $\mathcal{Y}_{\text{test}}$ at $\mathcal{I}_{\text{correct}}$
    Save $\mathcal{X}_{\text{correct}}$ and $\mathcal{Y}_{\text{correct}}$ to disk
$\mathcal{M} \leftarrow$ LOADMODEL($\mathcal{P}_{\text{model}}$)
$(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}}), (\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}}) \leftarrow$ LOADMNISTDATA
$\mathcal{X}_{\text{train}} \leftarrow$ PREPROCESSDATA($\mathcal{X}_{\text{train}}$)
$\mathcal{X}_{\text{test}} \leftarrow$ PREPROCESSDATA($\mathcal{X}_{\text{test}}$)
$\mathcal{Y}_{\text{train}} \leftarrow$ ONEHOTENCODE($\mathcal{Y}_{\text{train}}$)
$\mathcal{Y}_{\text{test}} \leftarrow$ ONEHOTENCODE($\mathcal{Y}_{\text{test}}$)
$\mathcal{P}$, $\mathcal{L}$, $\mathcal{A} \leftarrow$ EVALUATEMODEL($\mathcal{M}$, $\mathcal{X}_{\text{test}}$, $\mathcal{Y}_{\text{test}}$)
Print $\mathcal{A}$ as test accuracy
SAVECORRECT($\mathcal{X}_{\text{test}}$, $\mathcal{Y}_{\text{test}}$, $\mathcal{P}$)

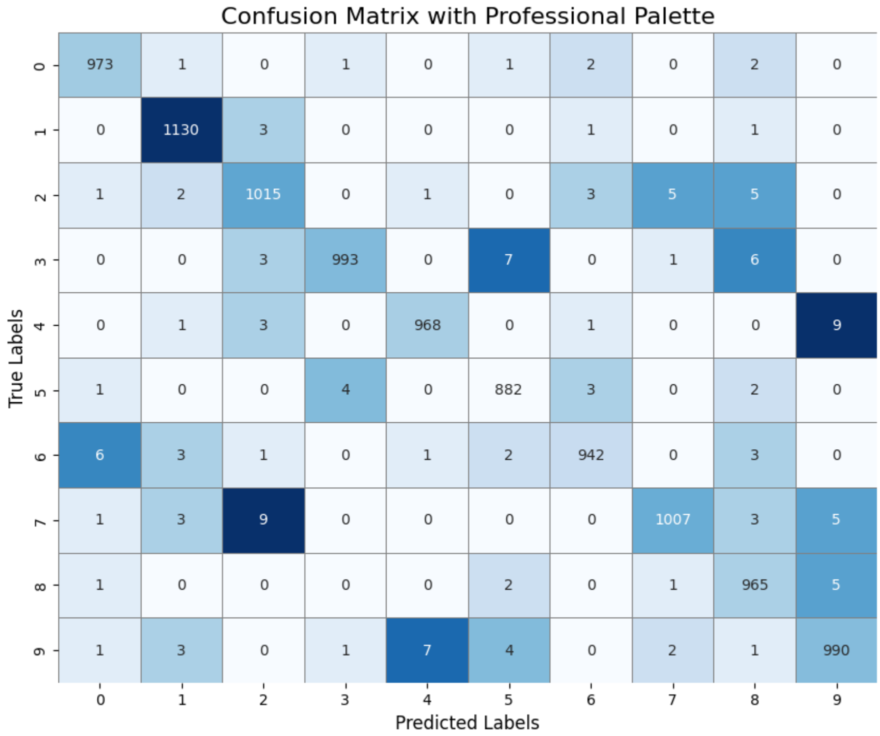



Figure 2: CM FSGM

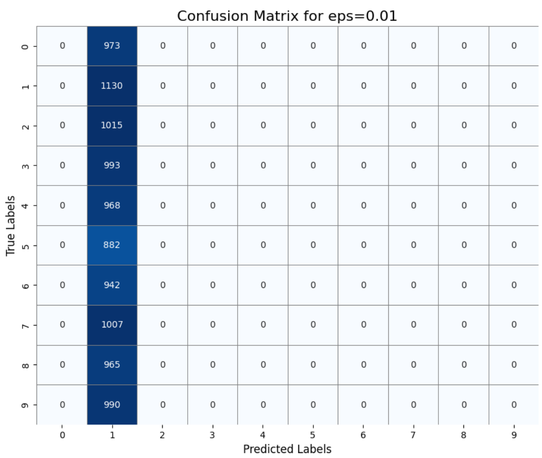
Table 1: Label counts

| | **Total MNIST** | **Training Set** | **Test Set** | **Predict Labels** | **Correct Labels** |
|---|---|---|---|---|---|
| **0** | 6903 | 5923 | 980 | 992 | 973 |
| **1** | 7877 | 6742 | 1135 | 1157 | 1130 |
| **2** | 6990 | 5958 | 1032 | 1037 | 1015 |
| **3** | 7141 | 6131 | 1010 | 999 | 993 |
| **4** | 6824 | 5842 | 982 | 983 | 968 |
| **5** | 6313 | 5421 | 892 | 904 | 882 |
| **6** | 6876 | 5918 | 958 | 943 | 942 |
| **7** | 7293 | 6265 | 1028 | 1024 | 1007 |
| **8** | 6825 | 5851 | 974 | 954 | 965 |
| **9** | 6958 | 5949 | 1009 | 1007 | 990 |
| **Total** | **70000** | **60000** | **10000** | **10000** | **9865** |

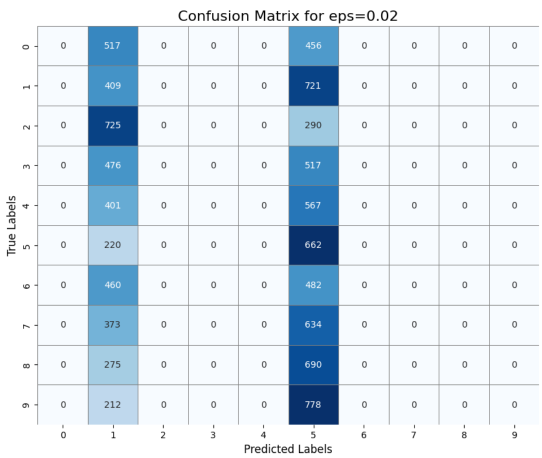
## 2. FSGM without corrected variables

**Algorithm 2** Adversarial Example Generation and Analysis

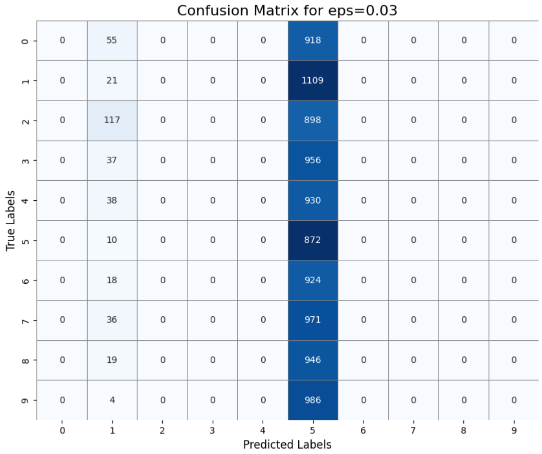
**Input:** classifier, correct_examples, correct_labels
**Output:** results_df, confusion matrices
eps_range ← $[0.01, 0.02, \ldots, 0.6]$
results_df ← DataFrame with columns ['eps', 'total_correct', 'total_adv', 'correct_adv_counts']
**for** eps ∈ eps_range **do**
    attack ← FastGradientMethod(classifier, eps)
    x_adv ← attack.generate(x=correct_examples)
    y_adv ← argmax(classifier.predict(x_adv), axis $= 1$)
    adv_counts ← count_unique(y_adv)
    cm ← confusion_matrix(correct_labels, y_adv)
    Save heatmap of cm to file with filename based on eps
    results_df.append({'eps' : eps, 'total_correct' : length(correct_labels), 'total_adv' : length(y_adv), 'correct_adv_counts' : adv_counts})
results_df.to_csv('/content/drive/MyDrive/ColabNotebooks/adv_results_labels.csv')



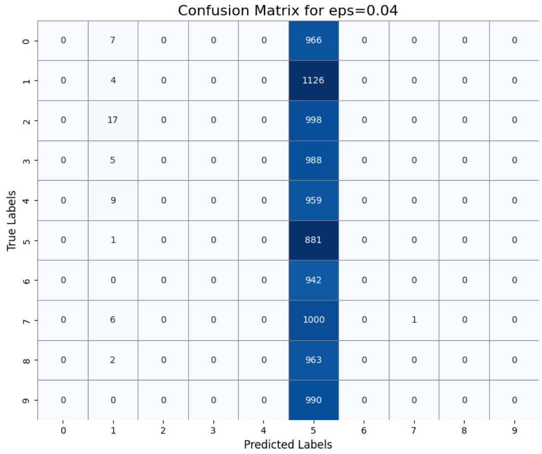

(a) confusion matrix 0.01



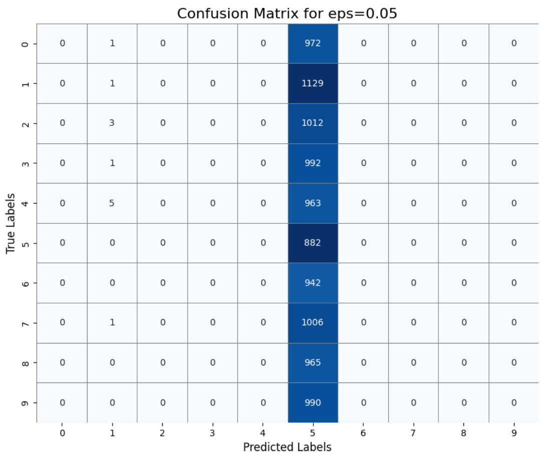

(b) confusion matrix 0.02



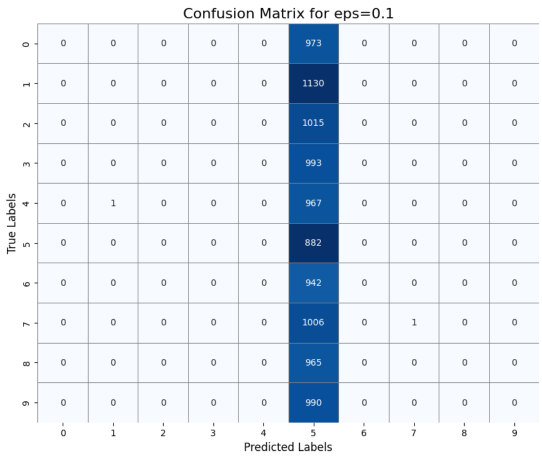

(a) confusion matrix 0.03



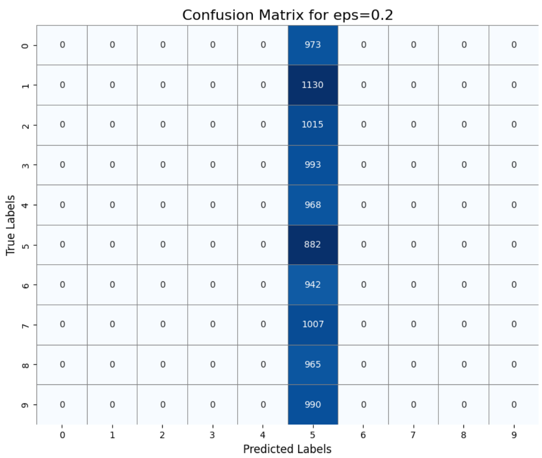

(b) confusion matrix 0.04



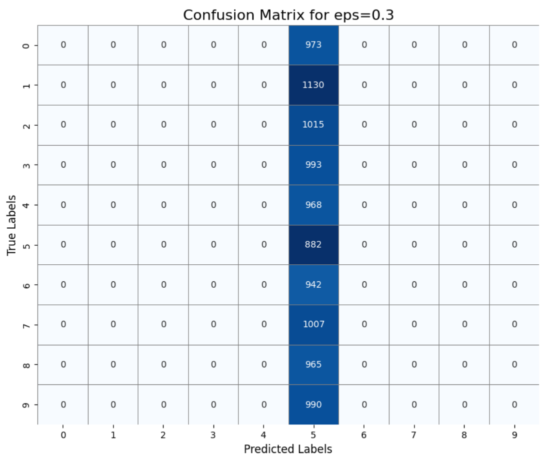

(a) confusion matrix 0.05



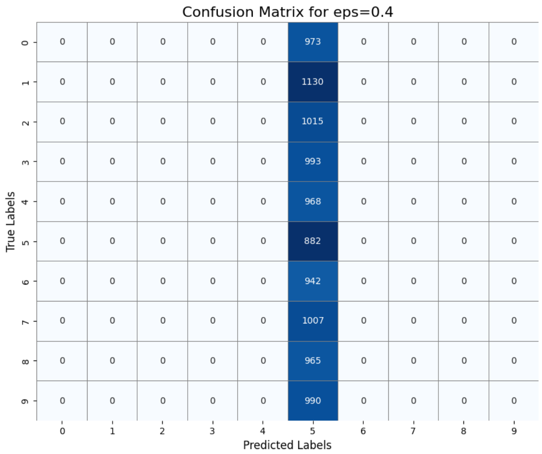

(b) confusion matrix 0.1



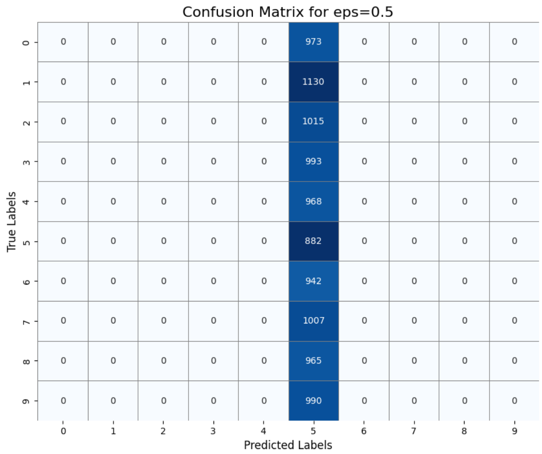

(a) confusion matrix 0.2



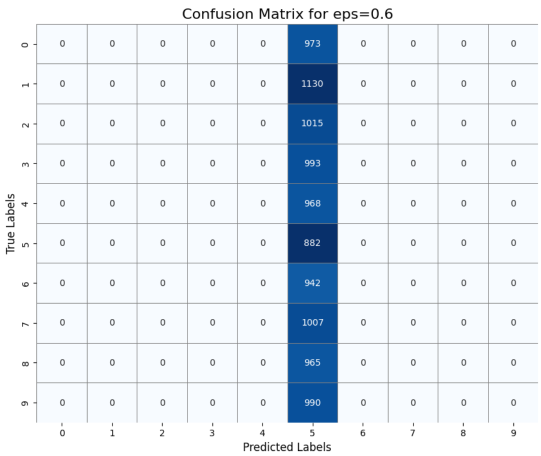

(b) confusion matrix 0.3




(a) confusion matrix 0.4




(b) confusion matrix 0.5

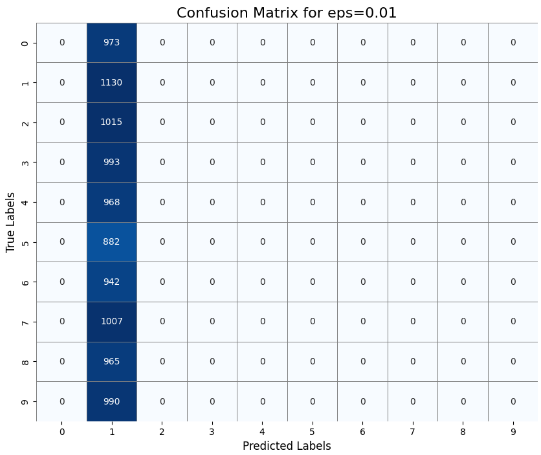



(a) confusion matrix 0.6

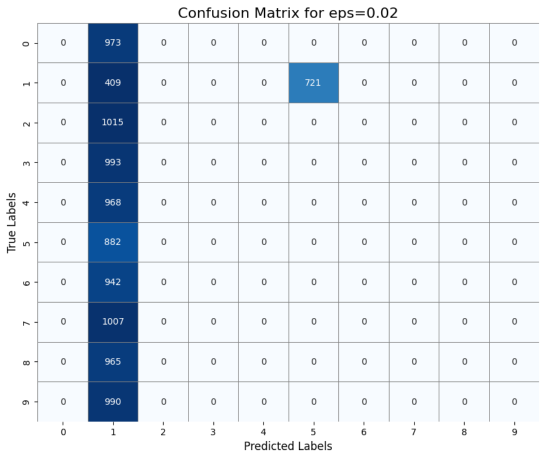
# 3. FSGM with corrected variables

**Algorithm 3** Adversarial Example Generation and Analysis

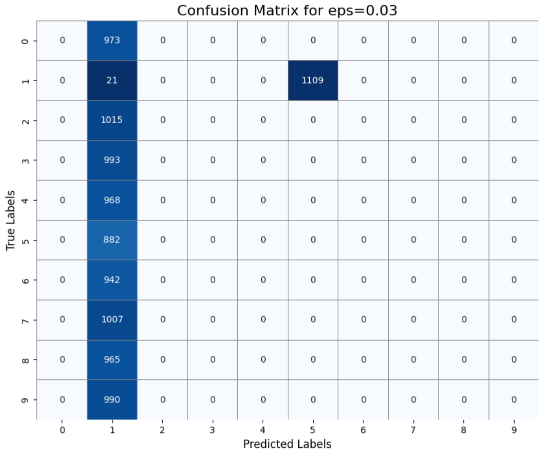
**Input:** classifier, correct_examples, correct_labels
**Output:** results_df, confusion matrices
eps_range ← $[0.01, 0.02, \ldots, 0.6]$
results_df ← DataFrame with columns ['eps', 'total_correct', 'total_adv', 'correct_adv_counts']
**for** eps ∈ eps_range **do**
    attack ← FastGradientMethod(classifier, eps)
    x_adv ← attack.generate(x=correct_examples, y=correct_labels)
    y_adv ← argmax(classifier.predict(x_adv), axis = 1)
    adv_counts ← count_unique(y_adv)
    cm ← confusion_matrix(correct_labels, y_adv)
    Save heatmap of cm to file with filename based on eps
    results_df.append({'eps' : eps, 'total_correct' : length(correct_labels), 'total_adv' : length(y_adv), 'correct_adv_counts' : adv_counts})
results_df.to_csv('/content/drive/MyDrive/ColabNotebooks/adv_results_labels.csv')



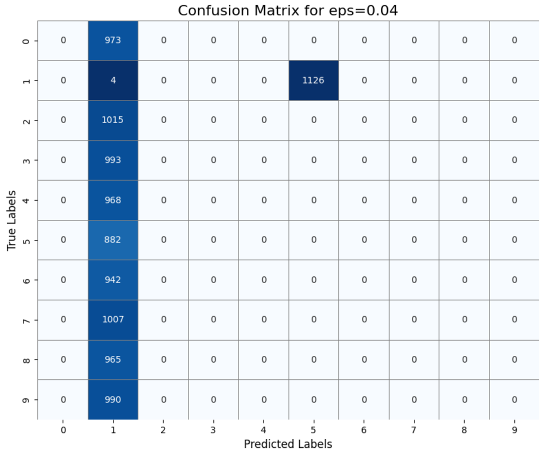

(a) confusion matrix 0.01 (output variable)



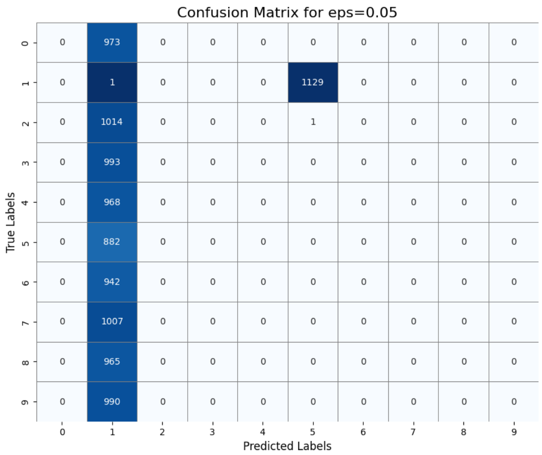

(b) confusion matrix 0.02 (output variable)



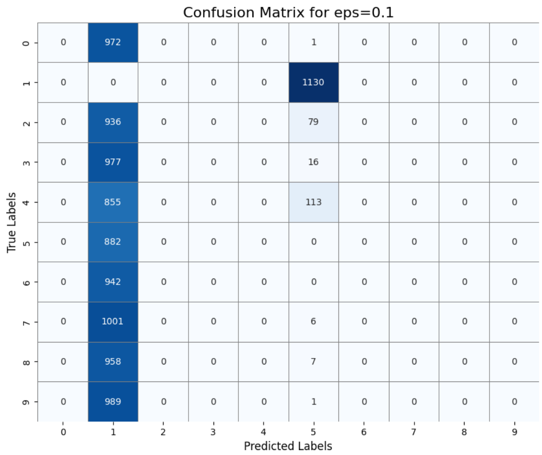

(a) confusion matrix 0.03 (output variable)



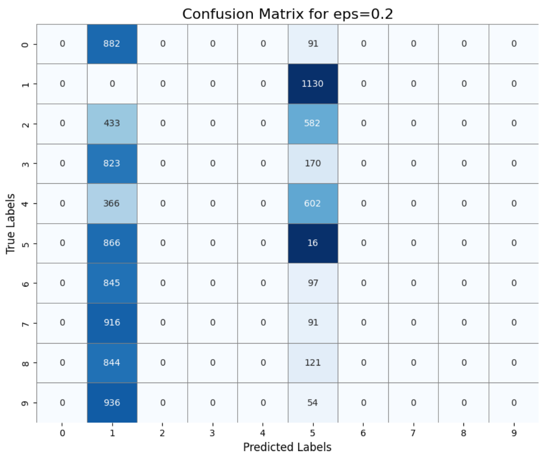

(b) confusion matrix 0.04 (output variable)



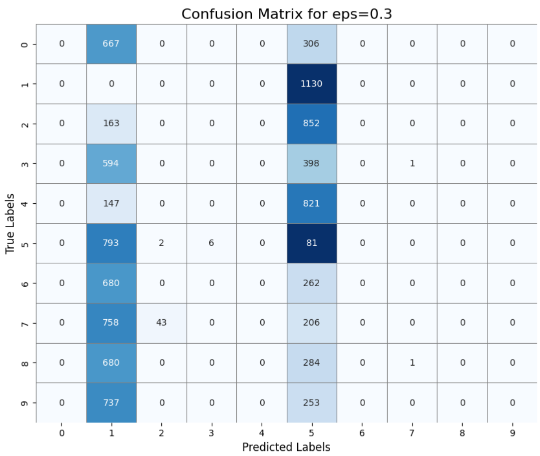

(a) confusion matrix 0.05 (output variable)



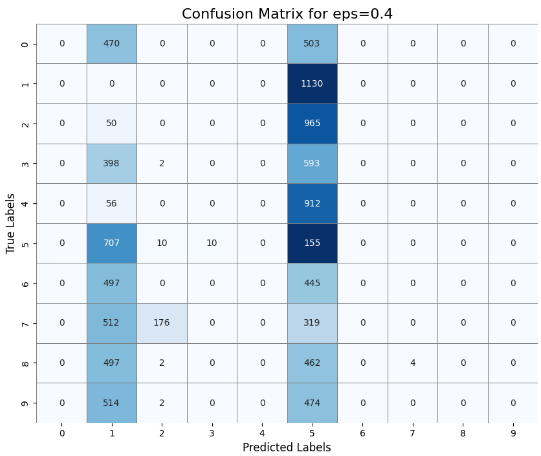

(b) confusion matrix 0.1 (output variable)

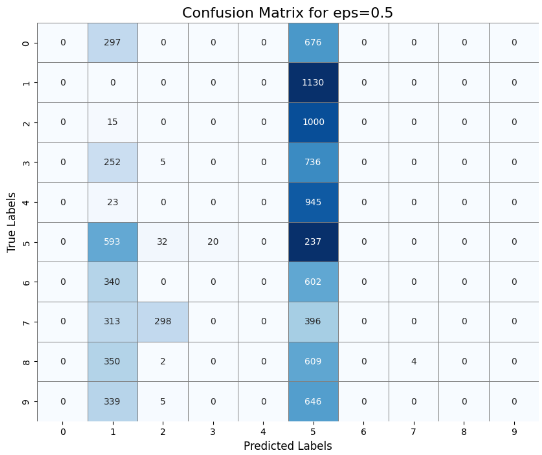



(a) confusion matrix 0.2(output variable)

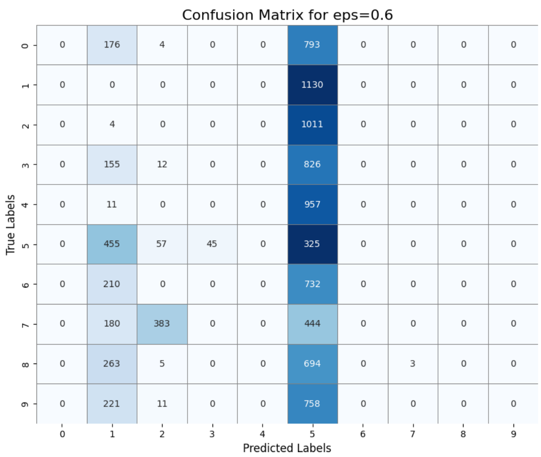



(b) confusion matrix 0.3 (output variable)




(a) confusion matrix 0.5 (output variable)



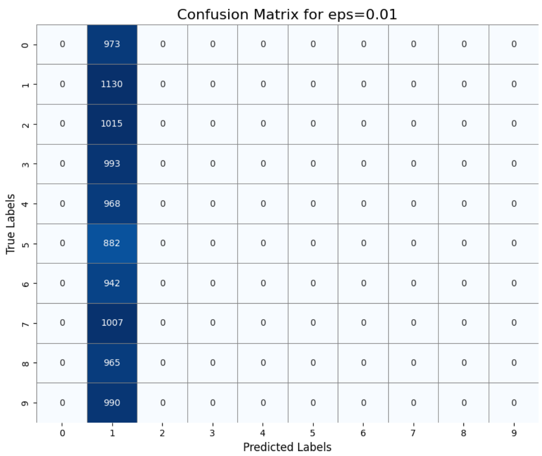

(a) confusion matrix 0.6 (output variable)

| | **eps** | **num** | **with_correct_Labels** | **without_correct_Labels** |
|---|---|---|---|---|
| **0** | 0.01 | 1 | {1: 9865} | {1: 9865} |
| **1** | 0.01 | 2 | {1: 9865} | {1: 9865} |
| **2** | 0.01 | 3 | {1: 9865} | {1: 9865} |
| **3** | 0.01 | 4 | {1: 9865} | {1: 9865} |
| **4** | 0.01 | 5 | {1: 9865} | {1: 9865} |
| **5** | 0.02 | 1 | {1: 9144, 5: 721} | {1: 4068, 5: 5797} |
| **6** | 0.02 | 2 | {1: 9144, 5: 721} | {1: 4068, 5: 5797} |
| **7** | 0.02 | 3 | {1: 9144, 5: 721} | {1: 4068, 5: 5797} |
| **8** | 0.02 | 4 | {1: 9144, 5: 721} | {1: 4068, 5: 5797} |
| **9** | 0.02 | 5 | {1: 9144, 5: 721} | {1: 4068, 5: 5797} |
| **10** | 0.03 | 1 | {1: 8756, 5: 1109} | {1: 355, 5: 9510} |
| **11** | 0.03 | 2 | {1: 8756, 5: 1109} | {1: 355, 5: 9510} |
| **12** | 0.03 | 3 | {1: 8756, 5: 1109} | {1: 355, 5: 9510} |
| **13** | 0.03 | 4 | {1: 8756, 5: 1109} | {1: 355, 5: 9510} |
| **14** | 0.03 | 5 | {1: 8756, 5: 1109} | {1: 355, 5: 9510} |
| **15** | 0.04 | 1 | {1: 8739, 5: 1126} | {1: 51, 5: 9813, 7: 1} |
| **16** | 0.04 | 2 | {1: 8739, 5: 1126} | {1: 51, 5: 9813, 7: 1} |
| **17** | 0.04 | 3 | {1: 8739, 5: 1126} | {1: 51, 5: 9813, 7: 1} |
| **18** | 0.04 | 4 | {1: 8739, 5: 1126} | {1: 51, 5: 9813, 7: 1} |
| **19** | 0.04 | 5 | {1: 8739, 5: 1126} | {1: 51, 5: 9813, 7: 1} |
| **20** | 0.05 | 1 | {1: 8735, 5: 1130} | {1: 12, 5: 9853} |
| **21** | 0.05 | 2 | {1: 8735, 5: 1130} | {1: 12, 5: 9853} |
| **22** | 0.05 | 3 | {1: 8735, 5: 1130} | {1: 12, 5: 9853} |
| **23** | 0.05 | 4 | {1: 8735, 5: 1130} | {1: 12, 5: 9853} |
| **24** | 0.05 | 5 | {1: 8735, 5: 1130} | {1: 12, 5: 9853} |
| **25** | 0.10 | 1 | {1: 8512, 5: 1353} | {1: 1, 5: 9863, 7: 1} |
| **26** | 0.10 | 2 | {1: 8512, 5: 1353} | {1: 1, 5: 9863, 7: 1} |
| **27** | 0.10 | 3 | {1: 8512, 5: 1353} | {1: 1, 5: 9863, 7: 1} |
| **28** | 0.10 | 4 | {1: 8512, 5: 1353} | {1: 1, 5: 9863, 7: 1} |
| **29** | 0.10 | 5 | {1: 8512, 5: 1353} | {1: 1, 5: 9863, 7: 1} |
| **30** | 0.20 | 1 | {1: 6911, 5: 2954} | {5: 9865} |
| **31** | 0.20 | 2 | {1: 6911, 5: 2954} | {5: 9865} |
| **32** | 0.20 | 3 | {1: 6911, 5: 2954} | {5: 9865} |
| **33** | 0.20 | 4 | {1: 6911, 5: 2954} | {5: 9865} |
| **34** | 0.20 | 5 | {1: 6911, 5: 2954} | {5: 9865} |
| **35** | 0.30 | 1 | {1: 5219, 2: 45, 3: 6, 5: 4593, 7: 2} | {5: 9865} |
| **36** | 0.30 | 2 | {1: 5219, 2: 45, 3: 6, 5: 4593, 7: 2} | {5: 9865} |
| **37** | 0.30 | 3 | {1: 5219, 2: 45, 3: 6, 5: 4593, 7: 2} | {5: 9865} |
| **38** | 0.30 | 4 | {1: 5219, 2: 45, 3: 6, 5: 4593, 7: 2} | {5: 9865} |
| **39** | 0.30 | 5 | {1: 5219, 2: 45, 3: 6, 5: 4593, 7: 2} | {5: 9865} |
| **40** | 0.40 | 1 | {1: 3701, 2: 192, 3: 10, 5: 5958, 7: 4} | {5: 9865} |
| **41** | 0.40 | 2 | {1: 3701, 2: 192, 3: 10, 5: 5958, 7: 4} | {5: 9865} |
| **42** | 0.40 | 3 | {1: 3701, 2: 192, 3: 10, 5: 5958, 7: 4} | {5: 9865} |
| **43** | 0.40 | 4 | {1: 3701, 2: 192, 3: 10, 5: 5958, 7: 4} | {5: 9865} |
| **44** | 0.40 | 5 | {1: 3701, 2: 192, 3: 10, 5: 5958, 7: 4} | {5: 9865} |
| **45** | 0.50 | 1 | {1: 2522, 2: 342, 3: 20, 5: 6977, 7: 4} | {5: 9865} |
| **46** | 0.50 | 2 | {1: 2522, 2: 342, 3: 20, 5: 6977, 7: 4} | {5: 9865} |
| **47** | 0.50 | 3 | {1: 2522, 2: 342, 3: 20, 5: 6977, 7: 4} | {5: 9865} |
| **48** | 0.50 | 4 | {1: 2522, 2: 342, 3: 20, 5: 6977, 7: 4} | {5: 9865} |
| **49** | 0.50 | 5 | {1: 2522, 2: 342, 3: 20, 5: 6977, 7: 4} | {5: 9865} |
| **50** | 0.60 | 1 | {1: 1675, 2: 472, 3: 45, 5: 7670, 7: 3} | {5: 9865} |
| **51** | 0.60 | 2 | {1: 1675, 2: 472, 3: 45, 5: 7670, 7: 3} | {5: 9865} |
| **52** | 0.60 | 3 | {1: 1675, 2: 472, 3: 45, 5: 7670, 7: 3} | {5: 9865} |
| **53** | 0.60 | 4 | {1: 1675, 2: 472, 3: 45, 5: 7670, 7: 3} | {5: 9865} |
| **54** | 0.60 | 5 | {1: 1675, 2: 472, 3: 45, 5: 7670, 7: 3} | {5: 9865} |

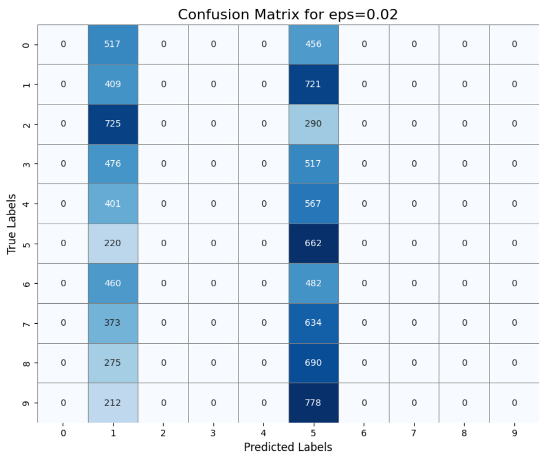
# 4. Multiple Attacks without corrected labels

**Algorithm 4** Adversarial Attack Evaluation

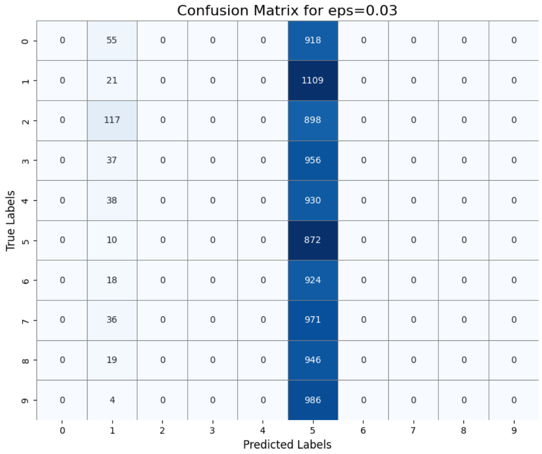
```
eps_range ← [0.01, 0.02, 0.03, ..., 0.6]
results_df ← DataFrame()
for eps ∈ eps_range do
    for attack_num ← 1 to 5 do
        attack ← FastGradientMethod(classifier, eps)
        x_adv ← attack.generate(x=correct_examples)
        y_adv ← classifier.predict(x_adv)
        cm ← confusion_matrix(correct_labels, y_adv)
        Save heatmap of cm to file
        results_df.append({'eps' : eps, 'attack_num' : attack_num, ...})
results_df.to_csv('/content/drive/MyDrive/ColabNotebooks/adv_results.csv')
Print(results_df)
```



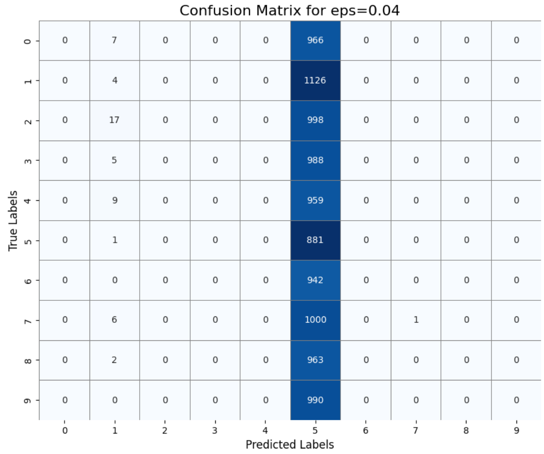

(a) confusion matrix 0.01 attack1



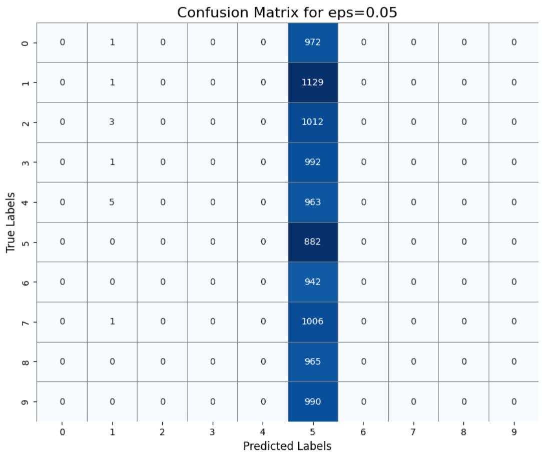

(b) confusion matrix 0.02 attack1



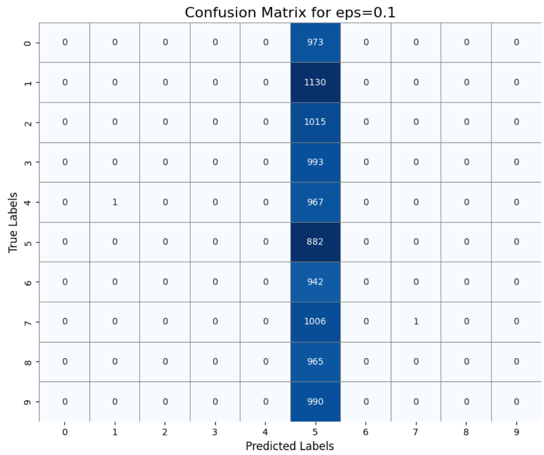

(a) confusion matrix 0.03 attack1



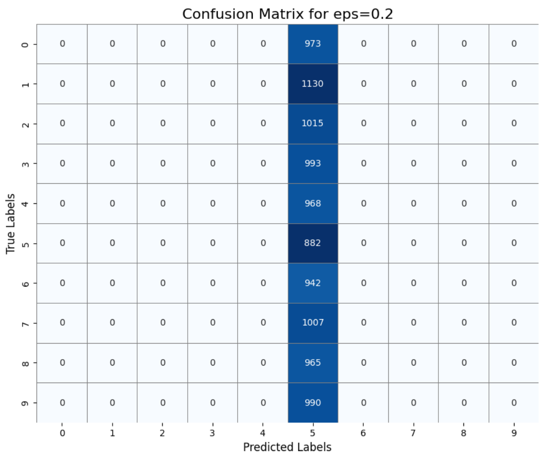

(b) confusion matrix 0.04 attack1



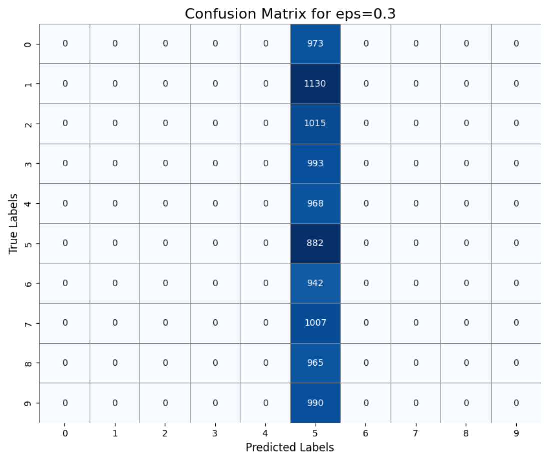

(a) confusion matrix 0.05 attack1



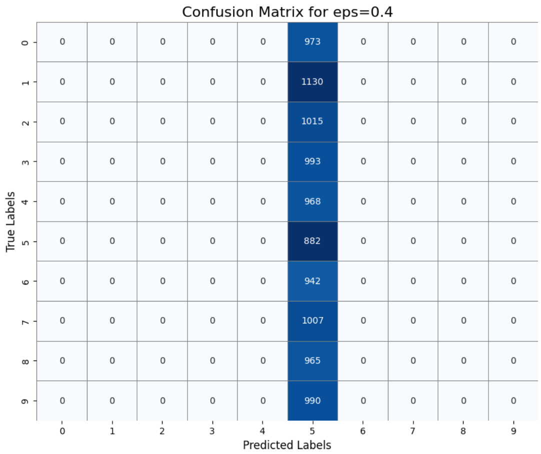

(b) confusion matrix 0.1 attack1



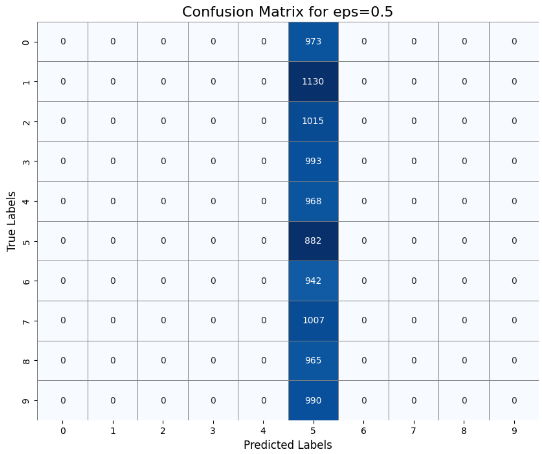

(a) confusion matrix 0.2 attack1



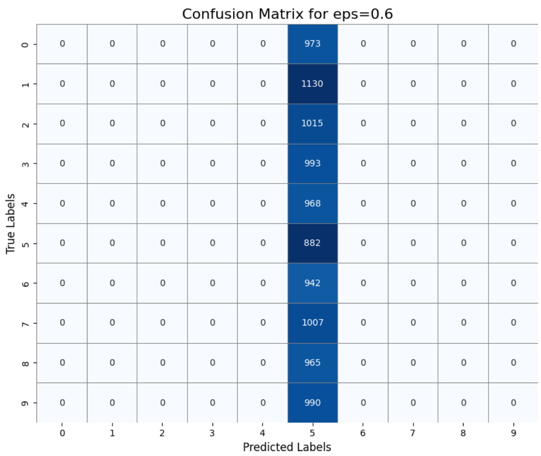

(b) confusion matrix 0.3 attack1




(a) confusion matrix 0.4 attack1




(b) confusion matrix 0.5 attack1

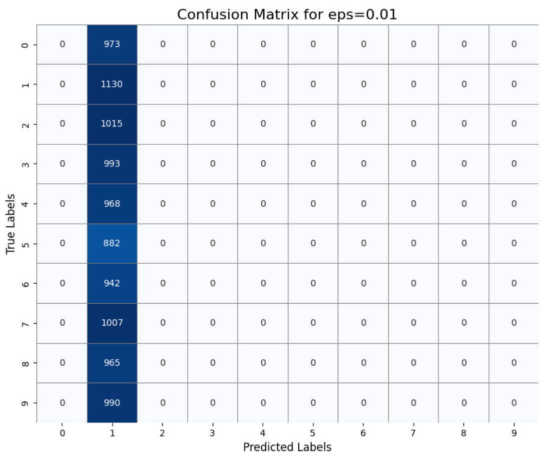



(a) confusion matrix 0.6 attack1

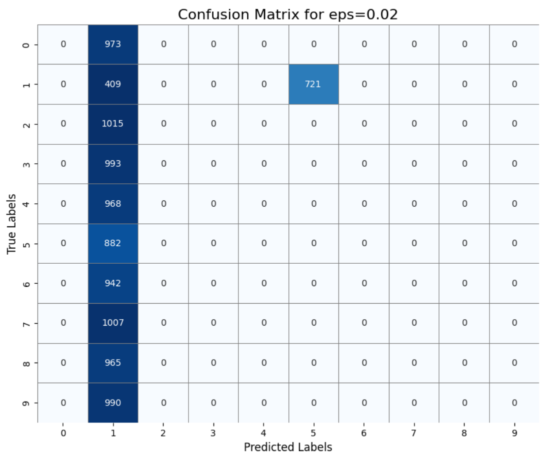
# 5. Multiple Attacks with corrected labels

**Algorithm 5** Adversarial Attack Evaluation

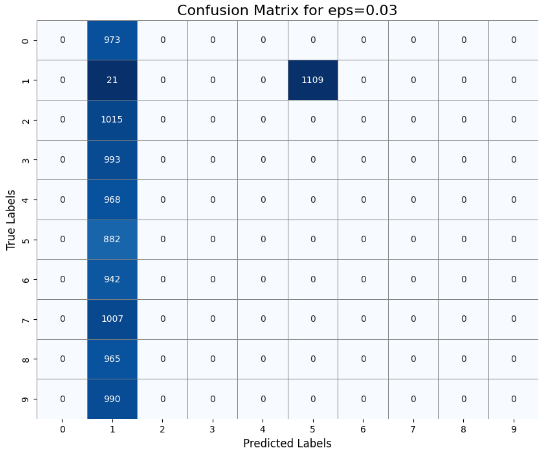
eps_range $\leftarrow [0.01, 0.02, 0.03, \ldots, 0.6]$
results_df $\leftarrow$ DataFrame()
**for** eps $\in$ eps_range **do**
    **for** attack_num $\leftarrow 1$ to $5$ **do**
        attack $\leftarrow$ FastGradientMethod(classifier, eps)
        x_adv $\leftarrow$ attack.generate(x=correct_examples,y=correct_labels)
        y_adv $\leftarrow$ classifier.predict(x_adv)
        cm $\leftarrow$ confusion_matrix(correct_labels, y_adv)
        Save heatmap of cm to file
        results_df.append({'eps' : eps, 'attack_num' : attack_num, ...})
results_df.to_csv('/content/drive/MyDrive/ColabNotebooks/adv_results.csv')
Print(results_df)



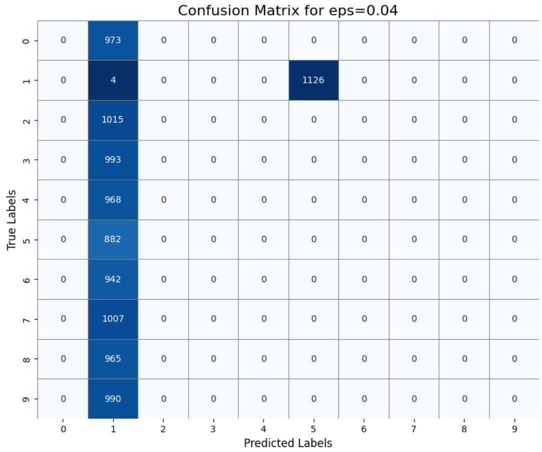

(a) confusion matrix 0.01 attack1 (output variable)



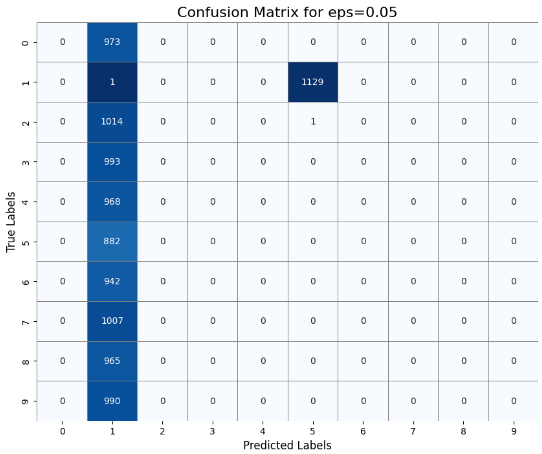

(b) confusion matrix 0.02 attack1 (output variable)



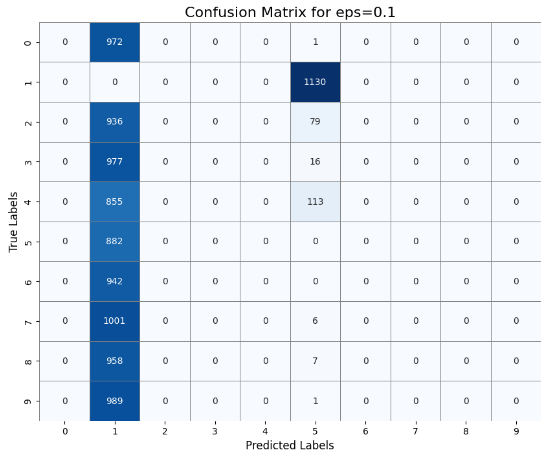

(a) confusion matrix 0.03 attack1 (output variable)



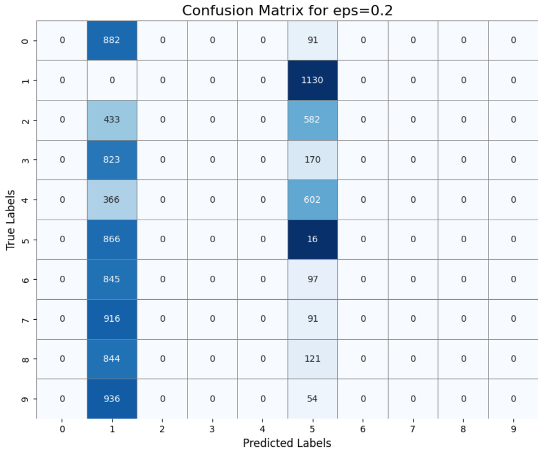

(b) confusion matrix 0.04 attack1 (output variable)



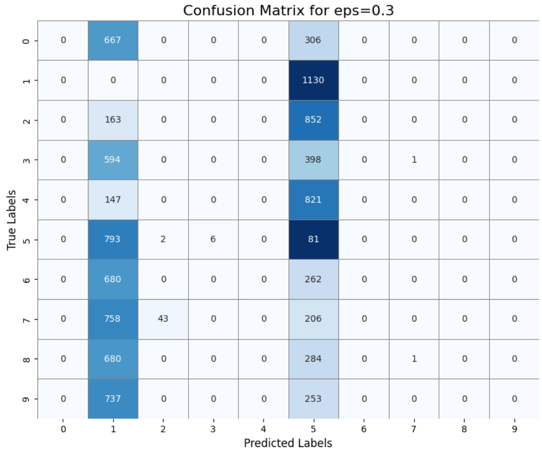

(a) confusion matrix 0.05 attack1 (output variable)



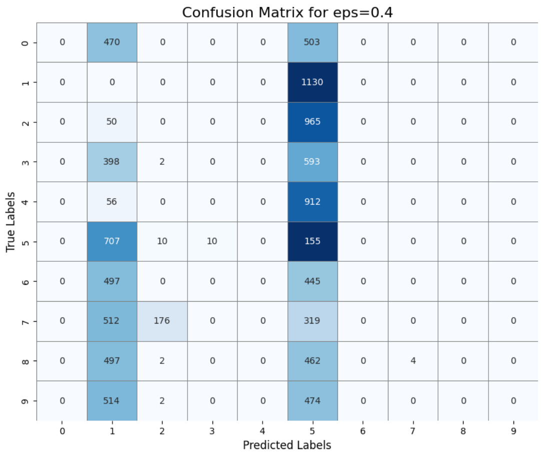

(b) confusion matrix 0.1 attack1 (output variable)



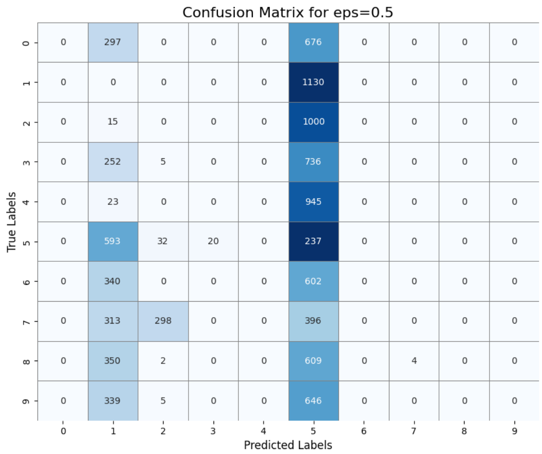

(a) confusion matrix 0.2 attack1 (output variable)



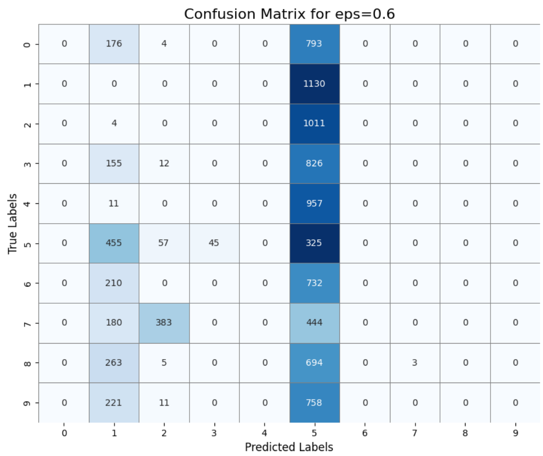

(b) confusion matrix 0.3 attack1 (output variable)




(a) confusion matrix 0.4 attack1 (output variable)




(b) confusion matrix 0.5 attack1 (output variable)




(a) confusion matrix 0.6 attack1 (output variable)