```
pip install adversarial-robustness-toolbox
```

```
Requirement already satisfied: adversarial-robustness-toolbox in /usr/local/lib/python3.10/dist-packages (Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from adversarial-Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from adversarial-requirement already satisfied: scikit-learn<1.2.0,>=0.22.2 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness Requirement already satisfied: scikit-learn/local/lib/python3.10/dist-packages (from adversarial-robustness Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn scik
```

Pre-trained model

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import to_categorical
# Load pre-trained model
model = load model('/content/drive/MyDrive/ColabNotebooks/mnist model.h5') # Update with the path to your mode
# Load MNIST data
(x_train, y_train), (x_test, y_test) = mnist.load_data()
# Preprocess the data
# Reshape the data to fit the model
x_{train} = x_{train.reshape}(60000, 28, 28, 1).astype('float32') / 255
x_{\text{test}} = x_{\text{test.reshape}}(10000, 28, 28, 1).astype('float32') / 255
# One-hot encode the labels
y_train = to_categorical(y_train, 10)
y_test_categorical = to_categorical(y_test, 10)
# Get the model's predictions on the test data
predictions = model.predict(x_test)
predicted_labels = np.argmax(predictions, axis=1)
nb_correct_pred = np.sum(predicted_labels == y_test)
print("Original test data:")
print("Correctly classified: {}".format(nb_correct_pred))
print("Incorrectly classified: {}".format(len(x_test)-nb_correct_pred))
# Evaluate the model on the test data
loss, accuracy = model.evaluate(x_test, y_test_categorical, verbose=0)
print(f'Test accuracy: {accuracy:.3f}')
# Save only the examples that the model identifies correctly
correct_indices = predicted_labels == y_test
correct_examples = x_test[correct_indices]
correct_labels = y_test[correct_indices]
# Save the correct examples and their labels
np.save('/content/drive/MyDrive/ColabNotebooks/correct_examples.npy', correct_examples)
np.save('/content/drive/MyDrive/ColabNotebooks/correct_labels.npy', correct_labels)
     /usr/local/lib/python3.10/dist-packages/keras/src/engine/training_v1.py:2359: UserWarning: `Model.state_up
      updates=self.state_updates,
     Original test data:
    Correctly classified: 9834
    Incorrectly classified: 166
     /usr/local/lib/python3.10/dist-packages/keras/src/engine/training_v1.py:2335: UserWarning: `Model.state_up
      updates = self.state_updates
    Test accuracy: 0.983
 # For clean examples
y_pred_clean = np.argmax(model.predict(correct_examples), axis=1)
```

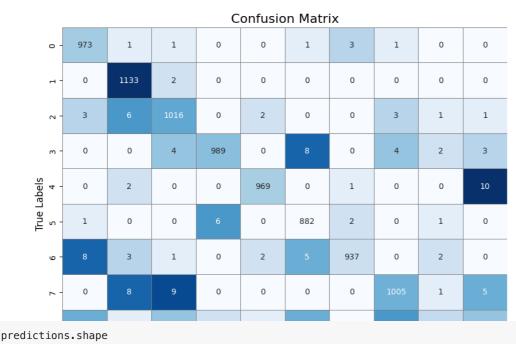
accuracy_clean = np.sum(y_pred_clean == correct_labels) / len(correct_labels)

ax.set xticklabels(label names) ax.set yticklabels(label names)

plt.show()

plt.savefig('model.png', bbox_inches='tight')

```
10/11/2023, 15:55
                                                     CounterExamplesV5.ipynb - Colaboratory
   print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
        Accuracy on clean data: 100.00%
   # Count the occurrences of each label in the test set
   unique, counts = np.unique(correct_labels, return_counts=True)
   correct_labels_counts = dict(zip(unique, counts))
   print("correct_labels_counts:", correct_labels_counts)
   for label, count in correct_labels_counts.items():
       print(f"Label {label}: {count}")
        correct_labels_counts: {0: 973, 1: 1133, 2: 1016, 3: 989, 4: 969, 5: 882, 6: 937, 7: 1005, 8: 946, 9: 984}
        Label 0: 973
        Label 1: 1133
        Label 2: 1016
        Label 3: 989
        Label 4: 969
        Label 5: 882
        Label 6: 937
        Label 7: 1005
        Label 8: 946
        Label 9: 984
   from sklearn.metrics import confusion_matrix
   import seaborn as sns
   import matplotlib.pyplot as plt
   import matplotlib.pyplot as plt
   # Calculate the confusion matrix
   cm = confusion_matrix(y_test, predicted_labels,labels=range(10))
   # Create a mask for the diagonal elements
   mask = np.eye(len(cm), dtype=bool)
   # Set up the matplotlib figure
   fig, ax = plt.subplots(figsize=(10, 8))
   # Plot the heatmap for off-diagonal elements using the mask
   # Use a professional color palette like 'Blues'
   sns.heatmap(cm, mask=mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, linecolor='grey
   # Plot the heatmap for diagonal elements using the inverse of the mask
   # Use the same color palette for consistency
   sns.heatmap(cm, mask=~mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, linecolor='gre
   # Labels, title and ticks
   label names = [f'{i}' for i in range(10)]
   ax.set_xlabel('Predicted Labels', fontsize=12)
   ax.set_ylabel('True Labels', fontsize=12)
   ax.set_title('Confusion Matrix', fontsize=16)
```



```
(10000, 10)
```

```
# Count the occurrences of each label in the correct labels
unique, counts = np.unique(predicted_labels, return_counts=True)
predicted_labels_counts = dict(zip(unique, counts))
print("correct_predicted set label counts:", predicted_labels_counts)
for label, count in predicted_labels_counts.items():
              print(f"Label {label}: {count}")
total_predicted_labels_count = len(predicted_labels)
# Display the total count of all labels
print(f"Total count of all total_predicted_labels_count combined in the MNIST dataset: {total_predicted_labels_count combined in the MNIST dataset: {total_
                correct_predicted set label counts: {0: 992, 1: 1157, 2: 1037, 3: 999, 4: 983, 5: 904, 6: 943, 7: 1024, 8:
                Label 0: 992
                Label 1: 1157
                Label 2: 1037
                Label 3: 999
                Label 4: 983
                Label 5: 904
               Label 6: 943
                Label 7: 1024
                Label 8: 954
                Label 9: 1007
```

FastGradientMethod

Load the saved examples Choose attack FGM Apply without target, with target, muktiple attacks without target and with target

Total count of all total predicted labels count combined in the MNIST dataset: 10000

```
import numpy as np
import tensorflow as tf
from keras.models import load_model
from art.attacks.evasion import FastGradientMethod
from art.estimators.classification import KerasClassifier
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Disable eager execution (necessary for ART with TensorFlow v1)
tf.compat.v1.disable_eager_execution()

# Load your trained model
```

```
model = load_model('/content/drive/MyDrive/ColabNotebooks/mnist_model.h5')

# Load the saved correct examples and their labels
correct_examples = np.load('/content/drive/MyDrive/ColabNotebooks/correct_examples.npy')
correct_labels = np.load('/content/drive/MyDrive/ColabNotebooks/correct_labels.npy')

# Preprocess the examples
#correct_examples = correct_examples.astype('float32') / 255

# Wrap the model with ART KerasClassifier
classifier = KerasClassifier(model=model, clip_values=(0, 1))

# For clean examples
y_pred_clean = np.argmax(classifier.predict(correct_examples), axis=1)
accuracy_clean = np.sum(y_pred_clean == correct_labels) / len(correct_labels)
print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
```

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training_v1.py:2359: UserWarning: `Model.state_up updates=self.state_updates,
Accuracy on clean data: 100.00%

```
import numpy as np
import tensorflow as tf
from keras.models import load model
from art.attacks.evasion import FastGradientMethod
from art.estimators.classification import KerasClassifier
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
# Disable eager execution (necessary for ART with TensorFlow v1)
tf.compat.v1.disable_eager_execution()
# Load your trained model
model = load_model('/content/drive/MyDrive/ColabNotebooks/mnist_model.h5')
# Load the saved correct examples and their labels
correct_examples = np.load('/content/drive/MyDrive/ColabNotebooks/correct_examples.npy')
correct_labels = np.load('/content/drive/MyDrive/ColabNotebooks/correct_labels.npy')
# Preprocess the examples
# correct_examples = correct_examples.astype('float32') / 255
# Wrap the model with ART KerasClassifier
classifier = KerasClassifier(model=model, clip_values=(0, 1))
# For clean examples
y_pred_clean = np.argmax(classifier.predict(correct_examples), axis=1)
accuracy_clean = np.sum(y_pred_clean == correct_labels) / len(correct_labels)
print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
# Define the range of eps values
eps_range = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
# Initialize a DataFrame to store results
results_df = pd.DataFrame(columns=['eps', 'total_correct', 'total_adv', 'correct_adv_counts'])
# Loop over the eps values
for eps in eps_range:
    # Define the attack with the current eps
    attack = FastGradientMethod(classifier, eps=eps)
    # Apply the attack to generate adversarial examples
    x_adv = attack.generate(x=correct_examples)
    # Predict the labels of the adversarial examples
    y_adv = np.argmax(classifier.predict(x_adv), axis=1)
    nb_correct_adv_pred = np.sum(y_adv == correct_labels)
    print(f"Adversarial test data: eps:{eps}")
```

```
print("Correctly classified: {}".format(nb_correct_adv_pred))
    print("Incorrectly classified: {}".format(len(correct_examples)-nb_correct_adv_pred))
    # For clean examples
    y_pred_clean = np.argmax(classifier.predict(correct_examples), axis=1)
    accuracy_clean = np.sum(y_pred_clean == correct_labels) / len(correct_labels)
    print(f"accuracy_clean:{accuracy_clean}")
    # Inside the loop for each eps
    # You have already calculated this for adversarial examples:
    # nb_correct_adv_pred = np.sum(y_adv == correct_labels)
    # Now calculate the accuracy for adversarial examples
    accuracy_adv = nb_correct_adv_pred / len(correct_labels)
    print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
    print(f"Adversarial test data: eps:{eps}")
    print(f"Accuracy on adversarial examples: {accuracy_adv * 100:.2f}%")
    # Count the occurrences of each label in the adversarial predictions
    unique_adv, counts_adv = np.unique(y_adv, return_counts=True)
    adv_counts = dict(zip(unique_adv, counts_adv))
    # Calculate the confusion matrix
    cm = confusion_matrix(correct_labels, y_adv, labels=range(10))
    # Draw and save the confusion matrix
    fig, ax = plt.subplots(figsize=(10, 8))
    # Create a mask for the diagonal elements
    mask = np.eye(len(cm), dtype=bool)
    # Plot the heatmap for off-diagonal elements using the mask
    # Use a professional color palette like 'Blues'
    sns.heatmap(cm, mask=mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, linecolor='e
    # Plot the heatmap for diagonal elements using the inverse of the mask
    # Use the same color palette for consistency
    sns.heatmap(cm, mask=~mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, linecolor=
    # Labels, title and ticks
    label_names = [f'{i}' for i in range(10)]
    ax.set_xlabel('Predicted Labels', fontsize=12)
    ax.set_ylabel('True Labels', fontsize=12)
    ax.set_title(f'Confusion Matrix for eps={eps}', fontsize=16)
    ax.set_xticklabels(label_names)
    ax.set_yticklabels(label_names)
    image_filename = f'confusion_matrix_eps_{eps}.png'
    plt.savefig(image_filename, bbox_inches='tight')
    plt.show()
    # Save the results in the DataFrame
    results_df = results_df.append({
        'eps': eps,
        'total_correct': len(correct_labels),
        'total_adv': len(y_adv),
        'correct_adv_counts': adv_counts
    }, ignore_index=True)
# Save the results to a CSV file
results_df.to_csv('/content/drive/MyDrive/ColabNotebooks/withouttruelabel_adv_results.csv', index=False)
# Print the DataFrame
print(results_df)
```

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01 Correctly classified: 9774 Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01

			C	omasic	iii iiiaci	IX 101 C	p3-0.0	_		
0	- 971	0	0	0	0	1	0	0	1	0
1	- 0	1132	0	0	0	0	1	0	0	0
2	- 0	2	1009	1	0	0	0	2	1	1
т	- 0	1	1	986	0	0	0	0	0	1
abels 4	- 0	0	2	0	962	0	1	0	0	4
True Labels 5 4	- 0	0	1	1	0	880	0	0	0	0
9	- 0	0	0	1	2	1	933	0	0	0
7	- 0	3	1	0	0	1	0	1000	0	0
œ	- 0	2		1	4	2	2	1	926	5
6	- 0	1	0	0	3	0	0	5	0	975
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remove. results_df = results_df.append({

Adversarial test data: eps:0.02 Correctly classified: 9692 Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

							P 0.0	_		
0 -	968	0	1	0	0	2	0	0	2	0
- 1	0	1132	0	0	0	0	1	0	0	0
2	- 1	8	999	1	1	1	0	3	1	1
m -	0	1	2	979	0	4	0	2	0	1
Labels 4	0	2	3	0	956	0	2	0	0	6
True L	1	1	1	1	0	877	1	0	0	0



<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be removed.

results_df = results_df.append({
Adversarial test data: eps:0.03
Correctly classified: 9550
Incorrectly classified: 284

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03

			C	omusic	iii Mati	1X 101 E	ps-0.0	J		
0 -	966	0	2	0	1	2	0	0	2	0
1	0	1131	0	0	0	1	1	0	0	0
2 -	. 3	13	980	7	1	1	1	5	4	1
m -	0	1	5	971	0	6	0	4	1	1
True Labels	1	4	3	0	943	0	3	1	1	13
True L	- 1	3	1	2	1	871	2	0	1	0
9 -	- 2	1	1	1	10	8	912	1	1	0
7	0	10	6	2	0	2	0	975	1	9
ω -	7	10	12	9	9	13	5	2	866	13
6 -		4	1	4	6	5	0	21	4	935
	0	í	2	3	4 Predicte	5ٰ d Labels	6	7	8	9

<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remove

results_df = results_df.append({
Adversarial test data: eps:0.04
Correctly classified: 9394
Incorrectly classified: 440

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

							•			
0 -	962	0	3	0	1	2	1	0	3	1

1	0	1129	0	0	0	1	1	1	1	0
7 -	- 5	16	966	9	3	1	1	9	4	2
m -	0	2	10	952	0	12	0	5	3	5
abels 4	1	6	5	0	932	0	4	1	2	18
True Labels	- 2	3	1	7	1	863	3	0	1	1
9 -	- 5	3	2	1	11	14	899	1	1	0
7	0	14	10	2	4	2	0	958	2	13
ω -	7	14	25	15	11	20	7	4	824	19
ი -		4	2	6	14	11	0	30	4	909
	ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remove results of append()

results_df = results_df.append({
Adversarial test data: eps:0.05
Correctly classified: 9132
Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

				C	oniusio	m Matr	ix ioi e	ps=0.0	5		
c	o -	955	0	4	0	1	5	3	0	3	2
,	- -	0	1126	1	1	0	1	2	1	1	0
ď	7 -	7	30	941	13	4	1	1	13	4	2
(m -	0	3	16	926	0	25	0	5	4	10
True Labels	4 -	2	12	6	0	918	0	4	2	3	22
True L	ი -	2	4	1	12	1	850	6	0	3	3
	- م	7	4	4	1	13	25	880	1	2	0
1		0	20	17	4	7	2	0	935	2	18
(ω -	8	22	37	27	16	37	9	9	753	28
,	~ _	5	6	3	14	40	17	n	41	10	848



<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remover sults_df = results_df.append({

Adversarial test data: eps:0.1 Correctly classified: 6834 Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

Confusion Matrix for eps=0.1

				Joinusi	on Mac	11/2 101 (-p3-0	ь		
0	847	1	26	2	7	36	24	6	11	13
. 1	- 0	1097	13	4	2	2	9	3	3	0
8	- 17	137	675	88	13	1	4	52	24	5
ω.	- 1	10	62	700	0	129	1	19	32	35
True Labels	7	51	20	0	738	0	11	26	7	109
True L	- 7	6	1	78	2	725	21	1	23	18
9 -	- 29	17	20	2	58	158	642	2	8	1
7	- 2	56	81	17	23	6	1	737	4	78
œ ·	- 17	65	175	151	35	121	20	32	252	78
ο.		12	8	51	192	55	1	195	43	421
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remove results_df = results_df.append({

Adversarial test data: eps:0.2 Correctly classified: 2300 Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

0 -	250	5	239	4	19	164	171	35	25	61
1 -	3	564	276	56	10	8	22	121	67	6
2 -	27	343	213	196	26	3	8	118	77	5
m -	0	24	143	238	0	364	1	32	74	113
1										

abel:	20	108	46	0	287	34	19	128	38	289
True Label	- 12	15	4	278	5	327	47	1	121	72
9 -	- 66	33	63	4	240	361	145	6	17	2
7	- 5	137	220	88	45	22	1	260	8	219
∞ -	20	96	258	192	38	179	21	45	10	87
ი -	7	22	19	141	282	94	1	323	89	6
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9

<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remoresults_df = results_df.append({

Adversarial test data: eps:0.3 Correctly classified: 793 Incorrectly classified: 9041 accuracy_clean:1.0 Accuracy on clean data: 100.00%

Adversarial test data: eps:0.3 Accuracy on adversarial examples: 8.06%

Confusion Matrix for eps=0.3

				zoniusi	on Mati	IX TOLE	ps=0.3	3		
0 -	85	7	291	5	17	287	161	50	22	48
1	- 2	120	418	362	36	13	32	48	101	1
2 -	- 28	400	95	239	30	7	7	123	83	4
m -	0	35	182	103	0	476	0	37	79	77
True Labels	16	138	80	7	97	147	19	192	55	218
True L	- 10	20	24	376	3	185	42	4	151	67
9 -	- 59	50	91	5	266	401	36	8	20	1
7	- 5	149	266	148	47	120	1	66	9	194
ω -	- 15	115	259	196	39	203	12	47	6	54
ი -		29	44	148	255	98	1	333	70	0
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9

<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remoresults_df = results_df.append({

Adversarial test data: eps:0.4 Correctly classified: 347 Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%

Confusion Matrix for eps=0.4

					on Pac		- 0.	<u> </u>		
0 -	12	7	297	8	17	390	137	58	19	28
1	1	5	396	608	11	14	12	2	84	0
7 -	- 26	422	44	263	27	19	7	120	85	3
m -	0	35	204	63	0	573	0	31	67	16
True Labels	15	151	116	23	38	255	15	183	63	110
True L	7	17	73	434	1	143	37	3	144	23
9 -	40	64	126	6	233	427	11	8	22	0
7	2	152	272	207	47	248	1	28	13	35
ω -	12	142	262	193	32	240	2	44	3	16
6 -		63	100	159	212	105	0	294	46	0
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9

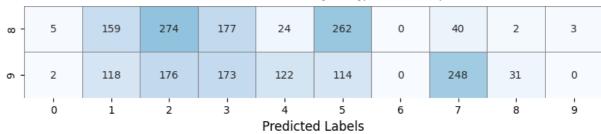
<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remove results_df = results_df.append({

Adversarial test data: eps:0.5 Correctly classified: 211 Incorrectly classified: 9623

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5 Accuracy on adversarial examples: 2.15%

			•	Joinusi	orr i-laci	17 101 0	.ps 0.5	,		
0 -	1	6	295	15	12	471	94	53	16	10
ч -	1	1	358	672	1	30	4	0	66	0
7 -	- 24	428	26	275	21	50	7	109	75	1
m -	- 0	36	194	30	0	643	0	30	53	3
abels 4	- 6	149	157	32	16	325	6	161	58	59
True Labels	- 2	17	129	467	1	119	25	3	116	3
9 -	- 27	86	157	8	158	463	3	11	24	0
7	- 1	150	277	225	36	284	0	13	13	6



<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remove results_df = results_df.append({

Adversarial test data: eps:0.6 Correctly classified: 161 Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

Confusion Matrix for eps=0.6

							-			
0 -	0	3	303	24	8	515	58	45	15	2
1	0	0	328	703	1	44	2	0	55	0
2 -	- 11	431	20	283	12	101	4	93	61	0
m -	0	32	184	18	0	688	0	24	43	0
abels 4	4	150	186	47	4	374	6	131	52	15
True Labels	0	16	169	480	0	112	13	2	90	0
9 -	14	82	185	9	89	520	3	14	21	0
۲ -	0	142	283	252	26	287	1	3	9	2
ω -	- 2	152	288	162	15	290	0	35	1	1
თ -		143	267	184	54	134	0	183	18	0
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9

eps total_correct total_adv 0 0.01 9834 9834 1 0.02 9834 9834 2 9834 9834 0.03 3 0.04 9834 9834 4 9834 9834 0.05 5 0.10 9834 9834 6 9834 9834 0.20 7 0.30 9834 9834 8 9834 9834 0.40 9 0.50 9834 9834 10 0.60 9834 9834

correct_adv_counts

```
0 {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
1 {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
2 {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
3 {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
4 {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5: ...
5 {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
6 {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5: ...
```

```
{0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
        {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
    9 {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
10 {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
    <ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is deprecated and will be remove
# Define the range of eps values
eps_range = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
# Initialize a DataFrame to store results
results_df = pd.DataFrame(columns=['eps', 'total_correct', 'total_adv', 'correct_adv_counts'])
# Loop over the eps values
for eps in eps_range:
    # Define the attack with the current eps
    attack = FastGradientMethod(classifier, eps=eps)
    # Apply the attack to generate adversarial examples
    x_adv = attack.generate(x=correct_examples ,y=correct_labels)
    # Predict the labels of the adversarial examples
    y_adv = np.argmax(classifier.predict(x_adv), axis=1)
    nb_correct_adv_pred = np.sum(y_adv == correct_labels)
    print(f"Adversarial test data: eps:{eps}")
    print("Correctly classified: {}".format(nb_correct_adv_pred))
    print("Incorrectly classified: {}".format(len(correct_examples)-nb_correct_adv_pred))
    # For clean examples
    y_pred_clean = np.argmax(classifier.predict(correct_examples), axis=1)
    accuracy_clean = np.sum(y_pred_clean == correct_labels) / len(correct_labels)
    print(f"accuracy_clean:{accuracy_clean}")
    # Inside the loop for each eps
    # You have already calculated this for adversarial examples:
    # nb_correct_adv_pred = np.sum(y_adv == correct_labels)
    # Now calculate the accuracy for adversarial examples
    accuracy_adv = nb_correct_adv_pred / len(correct_labels)
    print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
    print(f"Adversarial test data: eps:{eps}")
    print(f"Accuracy on adversarial examples: {accuracy_adv * 100:.2f}%")
    # Count the occurrences of each label in the adversarial predictions
    unique_adv, counts_adv = np.unique(y_adv, return_counts=True)
    adv_counts = dict(zip(unique_adv, counts_adv))
    # Calculate the confusion matrix
    cm = confusion matrix(correct labels, y adv, labels=range(10))
   # Draw and save the confusion matrix
    fig, ax = plt.subplots(figsize=(10, 8))
    # Create a mask for the diagonal elements
    mask = np.eye(len(cm), dtype=bool)
    # Plot the heatmap for off-diagonal elements using the mask
    # Use a professional color palette like 'Blues'
    sns.heatmap(cm, mask=mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, linecolor='e
    # Plot the heatmap for diagonal elements using the inverse of the mask
    # Use the same color palette for consistency
    sns.heatmap(cm, mask=~mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, linecolor=
    # Labels, title and ticks
    label_names = [f'{i}' for i in range(10)]
    ax.set_xlabel('Predicted Labels', fontsize=12)
    ax.set_ylabel('True Labels', fontsize=12)
    ax.set_title(f'Confusion Matrix for eps={eps}', fontsize=16)
    ax.set_xticklabels(label_names)
    ax.set_yticklabels(label_names)
    image_filename = f'_correct_labels_confusion_matrix_eps_{eps}.png'
    plt.savefig(image_filename, bbox_inches='tight')
    plt.show()
```

```
# Save the results in the DataFrame
results_df = results_df.append({
    'eps': eps,
    'total_correct': len(correct_labels),
    'total_adv': len(y_adv),
    'correct_adv_counts': adv_counts
}, ignore_index=True)

# Save the results to a CSV file
results_df.to_csv('/content/drive/MyDrive/ColabNotebooks/with_truelabel_adv_results_labels.csv', index=False)

# Print the DataFrame
print(results_df)
```

Adversarial test data: eps:0.01 Correctly classified: 9774 Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01 True Labels 5 4 ó í Predicted Labels

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.02 Correctly classified: 9692 Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

	Confusion Matrix for eps=0.02											
0 -	968	0	1	0	0	2	0	0	2	0		
н.	- 0	1132	0	0	0	0	1	0	0	0		
2	- 1	8	999	1	1	1	0	3	1	1		
m -	- 0	1	2	979	0	4	0	2	0	1		
True Labels	- 0	2	3	0	956	0	2	0	0	6		
True L	- 1	1	1	1	0	877	1	0	0	0		
۰ و	- 0	0	1	1		2	928	0	0	0		
۲.	- 0			1	0	1	0	987	0	5		
œ -	- 4	5	9	6		3	3	1	903	6		
ი -		2	0	1	4	1	0	10	2	963		
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is results_df = results_df.append({

Adversarial test data: eps:0.03 Correctly classified: 9550 Incorrectly classified: 284 accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03

	Confusion Matrix for eps=0.03										
0 -	966	0	2	0	1	2	0	0	2	0	
н -	0	1131	0	0	0	1	1	0	0	0	
2 -	- 3	13	980	7	1	1	1	5	4	1	
m -	0	1	5	971	0	6	0	4	1	1	
True Labels	1	4	3	0	943	0	3	1	1	13	
True L	1	3	1	2	1	871	2	0	1	0	
9 -	2	1	1	1	10	8	912	1	1	0	
7	0	10	6	2	0	2	0	975	1	9	
ω -	. 7	10	12	9	9	13	5	2	866	13	
ი -	- 4	4	1	4	6	5	0	21	4	935	
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is

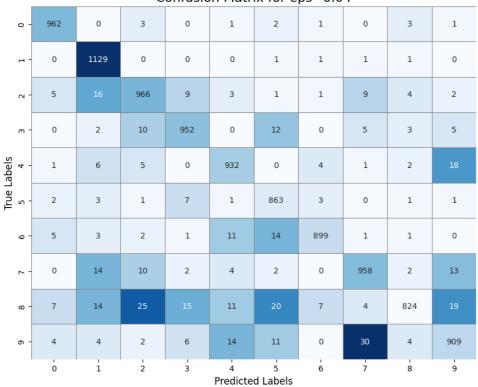
results_df = results_df.append({
Adversarial test data: eps:0.04
Correctly classified: 9394
Incorrectly classified: 440

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04



<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is
 results_df = results_df.append({
 Adversarial test data: ens:0 05

Muversariar rest data. epsivivs Correctly classified: 9132 Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

ACCUI	acy on	auversa		onfusio		ix for e	ps=0.0	5		
0	- 955	0	4	0	1	5	3	0	3	2
1	- 0	1126	1	1	0	1	2	1	1	0
2	- 7	30	941	13	4	1	1	13	4	2
м	- 0	3	16	926	0	25	0	5	4	10
True Labels 5 4	- 2	12	6	0	918	0	4	2	3	22
True L	- 2	4	1	12	1	850	6	0	3	3
9	- 7	4	4	1	13	25	880	1	2	0
7	- 0	20	17	4	7	2	0		2	18
80	- 8	22	37	27	16	37	9	9	753	28
6	- 5	6	3	14	40	17	0	41	10	848
	0	i	2	3	4	5	6	7	8	9

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({ Adversarial test data: eps:0.1 Correctly classified: 6834 Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

ACCUI	acy on a	auversa			on Mat	rix for e	eps=0.1	L		
0 -	847	1	26	2	7	36	24	6	11	13
	0	1097	13	4	2	2	9	3	3	0
5	- 17	137	675	88	13	1	4	52	24	5
m -	1	10	62		0	129	1	19	32	35
True Labels	7	51	20	0	738	0	11	26	7	109
True L	7	6	1	78	2	725	21	1	23	18
9 -	- 29	17	20	2	58	158	642	2	8	1
7	- 2	56	81	17	23	6	1	737	4	78
∞ -	- 17	65	175	151	35	121	20	32	252	78
ი -	- 6	12	8	51	192	55	1	195	43	421
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is results_df = results_df.append({

Adversarial test data: eps:0.2

Correctly classified: 2300 Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

Confusion Matrix for eps=0.2

	Confusion Matrix for eps=0.2											
0 -	- 250	5	239	4	19	164	171	35	25	61		
1	- 3	564	276	56	10	8	22	121	67	6		
2 -	- 27	343	213		26	3	8	118	77	5		
m -	0	24	143	238	0	364	1	32	74	113		
True Labels	- 20	108	46	0	287	34	19	128	38	289		
True L	12	15	4	278	5	327	47	1	121	72		
9 -	- 66	33	63	4	240	361	145	6	17	2		
7	- 5	137	220	88	45	22	1	260	8	219		
ω -	20	96	258	192	38	179	21	45	10	87		
6 -		22	19	141	282	94	1	323	89	6		
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

 $\verb|-ipython-input-65-1f2968efe6e5>: 72: Future Warning: The frame.append method is |$

results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793
Incorrectly classified: 9041

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%

Confusion Matrix for eps=0.3

	Confident Matrix for eps—0.5											
0 -	85	7	291	5	17	287	161	50	22	48		
г -	2	120	418	362	36	13	32	48	101	1		
2 -	- 28	400			30	7	7	123	83	4		
m -	0	35	182		0	476	0	37	79	77		
True Labels	16	138	80	7		147	19	192	55	218		
True L	- 10	20	24	376	3	185	42	4	151	67		
φ-	- 59	50	91	5		401	36	8	20	1		
۲-	- 5	149		148	47	120	1	66	9	194		
ω -	- 15	115	259	196	39	203	12	47	6	54		
ი -		29	44	148	255	98	1	333	70	0		
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is
 results df = results df.append({

Adversarial test data: eps:0.4

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%

Confusion Matrix for eps=0.4

	Confusion Matrix for eps=0.4												
0 -	- 12	7		8	17	390	137	58	19	28			
. 1	1	5	396	608	11	14	12	2	84	0			
7 -	- 26	422	44	263	27	19	7	120	85	3			
m -	0	35	204	63	0	573	0	31	67	16			
True Labels	15	151	116	23	38	255	15	183	63	110			
True L	7	17	73	434	1	143	37	3	144	23			
9 -	40	64	126	6	233	427	11	8	22	0			
۲ -	- 2	152	272	207	47	248	1	28	13	35			
ω -	- 12	142	262	193	32	240	2	44	3	16			
ი -		63	100	159	212	105	0		46	0			
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.5
Correctly classified: 211
Incorrectly classified: 9623

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

Confusion Matrix for eps=0.5

	Confusion Matrix for eps=0.5											
0 -	1	6	295	15	12	471	94	53	16	10		
н -	1	1		672	1	30	4	0	66	0		
- 2	- 24	428	26	275	21	50	7	109	75	1		
m -	0	36	194	30	0	643	0	30	53	3		
True Labels	- 6	149	157	32	16		6	161	58	59		
True L	- 2	17	129	467	1	119	25	3	116	3		
9 -	- 27	86	157	8	158	463	3	11	24	0		
7 -	1	150	277	225	36	284	0	13	13	6		
∞ -	- 5	159	274	177	24	262	0	40	2	3		
6 -		118	176	173	122	114	0	248	31	0		
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.6 Correctly classified: 161

0.60

Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

Confusion Matrix for eps=0.6 **True Labels** Predicted Labels eps total_correct total_adv 0.01 0.02 0.03 0.04 0.05 0.10 0.20 0.30 0.40 0.50

<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is
 results_df = results_df.append({

```
# Define the range of eps values
eps_range = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
# Initialize a DataFrame to store results
results_df = pd.DataFrame()
# Loop over the eps values
for eps in eps_range:
    for attack_num in range(1, 6): # Perform attack 5 times
       # Define the attack with the current eps
       attack = FastGradientMethod(classifier, eps=eps)
       # Apply the attack to generate adversarial examples
        x_adv = attack.generate(x=correct_examples)
       # Predict the labels of the adversarial examples
       y_adv = np.argmax(classifier.predict(x_adv), axis=1)
       nb_correct_adv_pred = np.sum(y_adv == correct_labels)
        print(f"Adversarial test data: eps:{eps}")
        print("Correctly classified: {}".format(nb correct adv pred))
        print("Incorrectly classified: {}".format(len(correct_examples)-nb_correct_adv_pred))
       # For clean examples
       y pred clean = np.argmax(classifier.predict(correct examples), axis=1)
       accuracy clean = np.sum(y pred clean == correct labels) / len(correct labels)
        print(f"accuracy_clean:{accuracy_clean}")
        # Inside the loop for each eps
        # You have already calculated this for adversarial examples:
        # nb_correct_adv_pred = np.sum(y_adv == correct_labels)
        # Now calculate the accuracy for adversarial examples
        accuracy_adv = nb_correct_adv_pred / len(correct_labels)
        print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
        print(f"Adversarial test data: eps:{eps}")
        print(f"Accuracy on adversarial examples: {accuracy_adv * 100:.2f}%")
       # Calculate the confusion matrix
        cm = confusion_matrix(correct_labels, y_adv, labels=range(10))
        # Draw and save the confusion matrix
        fig, ax = plt.subplots(figsize=(10, 8))
       # Create a mask for the diagonal elements
        mask = np.eye(len(cm), dtype=bool)
```

```
# Plot the heatmap for off-diagonal elements using the mask
       # Use a professional color palette like 'Blues'
       sns.heatmap(cm, mask=mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, linecolo
       # Plot the heatmap for diagonal elements using the inverse of the mask
       # Use the same color palette for consistency
        sns.heatmap(cm, mask=~mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, lineco
        # Labels, title and ticks
        label_names = [f'{i}' for i in range(10)]
        ax.set_xlabel('Predicted Labels', fontsize=12)
       ax.set_ylabel('True Labels', fontsize=12)
       ax.set title(f'Confusion Matrix for eps={eps}', fontsize=16)
       ax.set xticklabels(label names)
        ax.set_yticklabels(label_names)
        image_filename = f'confusion_matrix_eps_{eps}_attack_{attack_num}.png'
        plt.savefig(image_filename, bbox_inches='tight')
        plt.show() # Close the figure to avoid displaying it in the notebook
        # Save the results in the DataFrame
        results_df = results_df.append({
            'eps': eps,
            'attack_num': attack_num,
            'total_correct': len(correct_labels),
            'total_adv': len(y_adv),
            'correct_adv_counts': dict(zip(*np.unique(y_adv, return_counts=True)))
        }, ignore_index=True)
# Save the results to a CSV file
results_df.to_csv('/content/drive/MyDrive/ColabNotebooks/5attack_withouttruelabel_adv_results.csv', index=False
# Print the DataFrame
print(results_df)
```

Adversarial test data: eps:0.01 Correctly classified: 9774 Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01 True Labels 5 4 ó í Predicted Labels

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.01 Correctly classified: 9774 Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

	Confusion Matrix for eps=0.01											
0	971	0	0	0	0	1	0	0	1	0		
1	- 0	1132	0	0	0	0	1	0	0	0		
2	- 0	2		1	0	0	0	2	1	1		
m	- 0	1	1	986	0	0	0	0	0	1		
True Labels	- 0	0	2	0	962	0	1	0	0	4		
True L	- 0	0	1	1	0	880	0	0	0	0		
9	- 0	0	0	1	2	1	933	0	0	0		
7	- 0		1	0	0	1	0	1000	0	0		
00	- 0	2	3	1	4	2	2	1	926	5		
6		1	0	0		0	0	5	0	975		
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is results_df = results_df.append({

Adversarial test data: eps:0.01 Correctly classified: 9774 Incorrectly classified: 60 accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01 m **True Labels** œ

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.01
Correctly classified: 9774
Incorrectly classified: 60

accuracy_clean:1.0

ò

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

	Confusion Matrix for eps=0.01												
0 -	971	0	0	0	0	1	0	0	1	0			
1	0	1132	0	0	0	0	1	0	0	0			
7 -	0	2		1	0	0	0	2	1	1			
m -	- 0	1	1	986	0	0	0	0	0	1			
True Labels	0	0	2	0	962	0	1	0	0	4			
True l	0	0	1	1	0	880	0	0	0	0			
φ-	0	0	0	1	2	1	933	0	0	0			
^ -	0		1	0	0	1	0	1000	0	0			
ω -	0	2	3	1	4	2	2	1	926	5			
ი -	- 0	1	0	0		0	0	5	0	975			
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({
 Adversarial test data: ens:0 01

Muversariar rest darar ebs.n.ni Correctly classified: 9774 Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

	Confusion Matrix for eps=0.01											
0 -	971	0	0	0	0	1	0	0	1	0		
	0	1132	0	0	0	0	1	0	0	0		
2 -	- 0	2		1	0	0	0	2	1	1		
m -	0	1	1	986	0	0	0	0	0	1		
True Labels	0	0	2	0	962	0	1	0	0	4		
True L	0	0	1	1	0	880	0	0	0	0		
9 -	0	0	0	1	2	1	933	0	0	0		
۲.	0		1	0	0	1	0	1000	0	0		
œ -	0	2	3	1	4	2	2	1	926	5		
ი -	0	1	0	0	3	0	0	5	0	975		
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.02 Correctly classified: 9692 Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Accui	acy on	auversa			n Matr	ix for e	ps=0.0	2		
0	- 968	0	1	0	0	2	0	0	2	0
1	- 0	1132	0	0	0	0	1	0	0	0
2	- 1	8	999	1	1	1	0	3	1	1
м	- 0	1	2	979	0	4	0	2	0	1
True Labels	- 0	2	3	0	956	0	2	0	0	6
True L	- 1	1	1	1	0	877	1	0	0	0
9	- 0	0	1	1		2	928	0	0	0
7	- 0			1	0	1	0	987	0	5
ω	- 4		9			3	3	1	903	6
ი	- 1	2	0	1	4	1	0	10	2	963
	Ó	í	2	3	4 Predicte	5ٰ d Labels	6	7	8	9

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is results_df = results_df.append({ Adversarial test data: eps:0.02

Correctly classified: 9692 Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02

			C	omusic	iii Mati	IX IUI E	ps-0.0	_		
0 -	968	0	1	0	0	2	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
7 -	- 1	8	999	1	1	1	0	3	1	1
m -	0	1	2	979	0	4	0	2	0	1
True Labels	0	2	3	0	956	0	2	0	0	6
True L	1	1	1	1	0	877	1	0	0	0
9 -	0	0	1	1		2	928	0	0	0
7	. 0			1	0	1	0	987	0	5
ω -	4		9			3	3	1	903	6
6 -		2	0	1	4	1	0	10	2	963
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

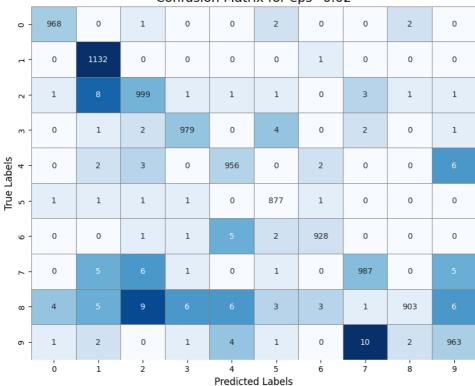
results_df = results_df.append({
Adversarial test data: eps:0.02
Correctly classified: 9692
Incorrectly classified: 142
accuracy classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02



<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
 results df = results df.append({

Adversarial test data: eps:0.02

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02

	Confusion Matrix for eps=0.02												
0 -	- 968	0	1	0	0	2	0	0	2	0			
п-	- 0	1132	0	0	0	0	1	0	0	0			
2 -	- 1	8	999	1	1	1	0	3	1	1			
m -	0	1	2	979	0	4	0	2	0	1			
True Labels	0	2	3	0	956	0	2	0	0	6			
True L	- 1	1	1	1	0	877	1	0	0	0			
9 -	0	0	1	1		2	928	0	0	0			
7	0			1	0	1	0	987	0	5			
ω -	4		9			3	3	1	903	6			
ი -		2	0	1	4	1	0	10	2	963			
	ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

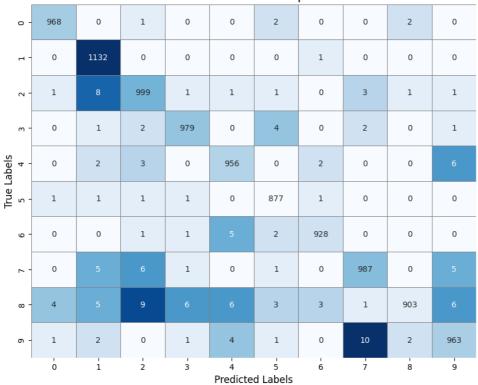
results_df = results_df.append({
Adversarial test data: eps:0.02
Correctly classified: 9692
Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02



<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
results_df = results_df.append({

Adversarial test data: eps:0.03 Correctly classified: 9550

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03 True Labels 5 4

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.03
Correctly classified: 9550
Incorrectly classified: 284

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

			С	onfusic	n Matr	ix for e	ps=0.0	3		
0 -	966	0	2	0	1	2	0	0	2	0
	- 0	1131	0	0	0	1	1	0	0	0
2 -	- 3	13	980	7	1	1	1	5	4	1
m -	0	1	5	971	0	6	0	4	1	1
True Labels	1	4	3	0	943	0	3	1	1	13
True L	1	3	1	2	1	871	2	0	1	0
9 -	- 2	1	1	1	10	8	912	1	1	0
۲ -	0	10	6	2	0	2	0	975	1	9
ω -	7	10	12	9	9	13	5	2	866	13
ი -	- 4	4	1	4	6	5	0	21	4	935
	ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
results_df = results_df.append({

Adversarial test data: eps:0.03 Correctly classified: 9550

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03

	Confusion Matrix for eps=0.03												
0 -	966	0	2	0	1	2	0	0	2	0			
1	0	1131	0	0	0	1	1	0	0	0			
7 -	- 3	13	980	7	1	1	1	5	4	1			
m -	0	1	5	971	0	6	0	4	1	1			
True Labels	1	4	3	0	943	0	3	1	1	13			
True L	1	3	1	2	1	871	2	0	1	0			
9 -	- 2	1	1	1	10	8	912	1	1	0			
7 -	0	10	6	2	0	2	0	975	1	9			
ω -	. 7	10	12	9	9	13	5	2	866	13			
6 -	4	4	1	4	6	5	0	21	4	935			
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.03 Correctly classified: 9550 Incorrectly classified: 284 accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03

	Confusion Matrix for eps=0.03												
0 -	966	0	2	0	1	2	0	0	2	0			
- 17	0	1131	0	0	0	1	1	0	0	0			
- 2	- 3	13	980	7	1	1	1	5	4	1			
m -	0	1	5	971	0	6	0	4	1	1			
abels 4	1	4	3	0	943	0	3	1	1	13			
True Labels	1	3	1	2	1	871	2	0	1	0			
9 -	- 2	1	1	1	10	8	912	1	1	0			
7 -	0	10	6	2	0	2	0	975	1	9			
∞ -	- 7	10	12	9	9	13	5	2	866	13			
6 -		4	1	4	6	5	0	21	4	935			
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.03
Correctly classified: 9550
Incorrectly classified: 284

accuracy_clean:1.0 Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

		ucy 011 ·		C		n Matr	ix for e	ps=0.0	3		
	0 -	966	0	2	0	1	2	0	0	2	0
	- n	0	1131	0	0	0	1	1	0	0	0
	7 -	3	13	980	7	1	1	1	5	4	1
	m -	0	1	5	971	0	6	0	4	1	1
True Labels	4 -	1	4	3	0	943	0	3	1	1	13
True L	ი -	1	3	1	2	1	871	2	0	1	0
	- و	2	1	1	1	10	8	912	1	1	0
	_	0	10	6	2	0	2	0	975	1	9
	ω -	7	10	12	9	9	13	5	2	866	13
	ი -	4	4	1	4	6	5	0	21	4	935
		Ó	'n	2	3	4	5	6	7	8	9

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({ Adversarial test data: eps:0.04 Correctly classified: 9394 Incorrectly classified: 440

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

			С	onfusio	n Matr	ix for e	ps=0.0	4		
0 -	962	0	3	0	1	2	1	0	3	1
1	0	1129	0	0	0	1	1	1	1	0
2	- 5		966	9	3	1	1	9	4	2
m -	0	2	10	952	0	12	0	5	3	5
True Labels	1	6	5	0	932	0	4	1	2	18
True L	- 2	3	1	7	1	863	3	0	1	1
9 -	- 5	3	2	1	11	14	899	1	1	0
۲.	0	14	10	2	4	2	0	958	2	13
ω -	7	14	25		11		7	4	824	19
ი -	- 4	4	2	6	14	11	0	30	4	909
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.04 Correctly classified: 9394 Incorrectly classified: 440

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04

	Confusion Matrix for eps=0.04											
0 -	962	0	3	0	1	2	1	0	3	1		
п-	- 0	1129	0	0	0	1	1	1	1	0		
7 -	- 5		966	9	3	1	1	9	4	2		
m -	0	2	10	952	0	12	0	5	3	5		
True Labels	1	6	5	0	932	0	4	1	2	18		
True L	- 2	3	1	7	1	863	3	0	1	1		
9 -	- 5	3	2	1	11	14	899	1	1	0		
۲.	0	14	10	2	4	2	0	958	2	13		
ω -	7	14	25		11		7	4	824	19		
ი -		4	2	6	14	11	0	30	4	909		
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

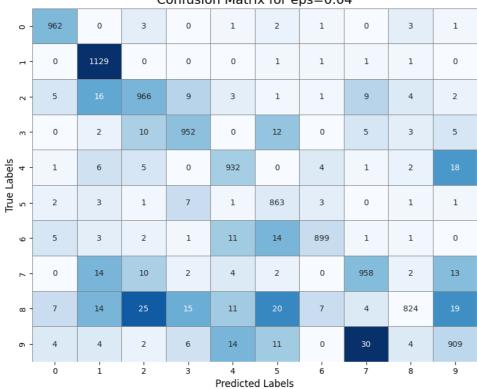
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.04
Correctly classified: 9394
Incorrectly classified: 440
accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04



<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.04
Correctly classified: 9394
Incorrectly classified: 440

accuracv clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04

	Confusion Matrix for eps=0.04											
0 -	962	0	3	0	1	2	1	0	3	1		
1	0	1129	0	0	0	1	1	1	1	0		
7 -	- 5		966	9	3	1	1	9	4	2		
m -	0	2	10	952	0	12	0	5	3	5		
True Labels	1	6	5	0	932	0	4	1	2	18		
True L	- 2	3	1	7	1	863	3	0	1	1		
φ-	- 5	3	2	1	11	14	899	1	1	0		
7	0	14	10	2	4	2	0	958	2	13		
ω -	7	14	25		11		7	4	824	19		
ი -		4	2	6	14	11	0	30	4	909		
	ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.04
Correctly classified: 9394
Incorrectly classified: 440

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04

	Confusion Matrix for eps=0.04											
0 -	962	0	3	0	1	2	1	0	3	1		
г.	- 0	1129	0	0	0	1	1	1	1	0		
2 -	- 5		966	9	3	1	1	9	4	2		
m -	- 0	2	10	952	0	12	0	5	3	5		
True Labels	1	6	5	0	932	0	4	1	2	18		
True L	- 2	3	1	7	1	863	3	0	1	1		
φ-	- 5	3	2	1	11	14	899	1	1	0		
۲.	0	14	10	2	4	2	0	958	2	13		
ω -	7	14	25		11		7	4	824	19		
ი -		4	2	6	14	11	0	30	4	909		
	Ó	í	2	3	4 Predicte	5 d Labels	6	7	8	9		

 $\verb|-ipython-input-56-9fea085cbc61>:66: Future Warning: The frame.append method is |$

results_df = results_df.append({
Adversarial test data: eps:0.05
Correctly classified: 9132
Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05

	Confusion Matrix for eps=0.05											
0 -		0	4	0	1	5	3	0	3	2		
н -	0	1126	1	1	0	1	2	1	1	0		
2 -	. 7	30	941	13	4	1	1	13	4	2		
m -	0	3	16	926	0	25	0	5	4	10		
True Labels	- 2	12	6	0	918	0	4	2	3	22		
True L	- 2	4	1	12	1	850	6	0	3	3		
φ-	7	4	4	1	13	25	880	1	2	0		
7	0		17	4	7	2	0		2	18		
ω -	- 8	22	37	27	16	37	9	9	753	28		
ი -		6	3	14	40	17	0	41	10	848		
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

 $\verb| <ipython-input-56-9fea085cbc61>:66: Future Warning: The frame.append method is | \\$

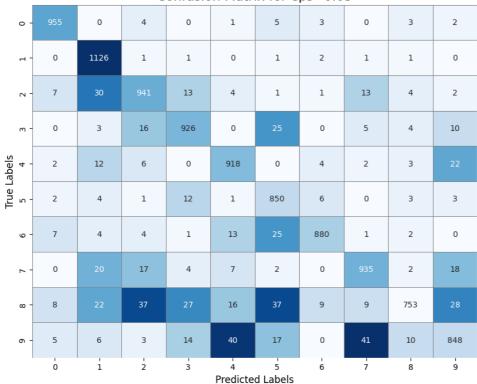
results_df = results_df.append({
Adversarial test data: eps:0.05
Correctly classified: 9132
Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05



<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.05
Correctly classified: 9132
Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00%

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05											
0 -	955	0	4	0	1	5	3	0	3	2	
. 1	- 0	1126	1	1	0	1	2	1	1	0	
2 -	7	30	941	13	4	1	1	13	4	2	
m -	0	3	16	926	0	25	0	5	4	10	
True Labels	- 2	12	6	0	918	0	4	2	3	22	
True L	- 2	4	1	12	1	850	6	0	3	3	
9 -	7	4	4	1	13	25	880	1	2	0	
7	0		17	4	7	2	0		2	18	
ω -	- 8	22	37	27	16	37	9	9	753	28	
ი -		6	3	14	40	17	0	41	10	848	
	Ö	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132

Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05 **True Labels** Predicted Labels

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is results_df = results_df.append({

Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00%

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05										
0 -	955	0	4	0	1	5	3	0	3	2
1	0	1126	1	1	0	1	2	1	1	0
2 -	. 7	30	941	13	4	1	1	13	4	2
m -	0	3	16	926	0	25	0	5	4	10
True Labels	- 2	12	6	0	918	0	4	2	3	22
True L	- 2	4	1	12	1	850	6	0	3	3
9 -	7	4	4	1	13	25	880	1	2	0
7	0		17	4	7	2	0		2	18
ω -	- 8	22	37	27	16	37	9	9	753	28
ი -		6	3	14	40	17	0	41	10	848
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.1 Correctly classified: 6834 Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

Confusion Matrix for eps=0.1

Confusion Matrix for eps=0.1										
0 -	847	1	26	2	7	36	24	6	11	13
- 11	0	1097	13	4	2	2	9	3	3	0
- 2	17	137	675	88	13	1	4	52	24	5
m -	1	10	62		0	129	1	19	32	35
abels 4	7	51	20	0	738	0	11	26	7	109
True Labels	7	6	1	78	2	725	21	1	23	18
	29	17	20	2	58	158	642	2	8	1
7 -	2	56	81	17	23	6	1	737	4	78
∞ -	17	65	175	151	35	121	20	32	252	78
6 -		12	8	51	192	55	1	195	43	421
,	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.1 Correctly classified: 6834

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: ens:0.1

Incorrectly classified: 3000

Marci sai tac cosc aaca: opsioit

Accuracy on adversarial examples: 69.49%

Confusion Matrix for eps=0.1											
0 -	847	1	26	2	7	36	24	6	11	13	
1	0	1097	13	4	2	2	9	3	3	0	
2 -	17	137	675	88	13	1	4	52	24	5	
m -	1	10	62		0	129	1	19	32	35	
True Labels	7	51	20	0	738	0	11	26	7	109	
True L	7	6	1	78	2	725	21	1	23	18	
φ-	- 29	17	20	2	58	158	642	2	8	1	
7	- 2	56	81	17	23	6	1	737	4	78	
ω -	17	65	175	151	35	121	20	32	252	78	
ი -	- 6	12	8	51	192	55	1	195	43	421	
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.1 Correctly classified: 6834 Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

Confusion Matrix for eps=0.1 m **True Labels** ω Predicted Labels

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({
 Advancedial took datasets.append 1

Adversarial test data: eps:0.1 Correctly classified: 6834 Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

421

Accuracy on adversarial examples: 69.49%

	Confusion Matrix for eps=0.1											
0 -	847	1	26	2	7	36	24	6	11	13		
1	0	1097	13	4	2	2	9	3	3	0		
7 -	17	137	675	88	13	1	4	52	24	5		
m -	- 1	10	62		0	129	1	19	32	35		
True Labels	7	51	20	0	738	0	11	26	7	109		
True L	7	6	1	78	2	725	21	1	23	18		
9 -	- 29	17	20	2	58	158	642	2	8	1		
7	- 2	56	81	17	23	6	1	737	4	78		
ω -	17	65	175	151	35	121	20	32	252	78		

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.1
Correctly classified: 6834
Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

	Confusion Matrix for eps=0.1												
0 -	847	1	26	2	7	36	24	6	11	13			
- 11	0	1097	13	4	2	2	9	3	3	0			
2 -	17	137	675	88	13	1	4	52	24	5			
m -	1	10	62		0	129	1	19	32	35			
True Labels	7	51	20	0	738	0	11	26	7	109			
True L	7	6	1	78	2	725	21	1	23	18			
9 -	- 29	17	20	2	58	158	642	2	8	1			
7 -	- 2	56	81	17	23	6	1	737	4	78			
∞ -	17	65	175	151	35	121	20	32	252	78			
6 -	- 6	12	8	51	192	55	1	195	43	421			
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.2 Correctly classified: 2300 Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial evamples: 23 30%

Accuracy on auversariac examples. 23:350

	Confusion Matrix for eps=0.2												
0 -	- 250	5	239	4	19	164	171	35	25	61			
1	3	564	276	56	10	8	22	121	67	6			
2 -	27	343	213		26	3	8	118	77	5			
m -	0	24	143	238	0	364	1	32	74	113			
True Labels	- 20	108	46	0	287	34	19	128	38	289			
True L	- 12	15	4	278	5	327	47	1	121	72			
9 -	- 66	33	63	4	240	361	145	6	17	2			
7	- 5	137	220	88	45	22	1	260	8	219			
ω -	- 20	96	258	192	38	179	21	45	10	87			
თ -		22	19	141	282	94	1	323	89	6			
	ó	i	2	3	4	5	6	7	8	9			

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300
Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

Confusion Matrix for eps=0.2

	Confusion Matrix for eps—0.2												
0 -	250	5	239	4	19	164	171	35	25	61			
- 11	3	564	276	56	10	8	22	121	67	6			
2 -	27	343	213		26	3	8	118	77	5			
m -	0	24	143	238	0	364	1	32	74	113			
True Labels	20	108	46	0	287	34	19	128	38	289			
True L	12	15	4	278	5	327	47	1	121	72			
9 -	66	33	63	4	240	361	145	6	17	2			
7 -	5	137	220	88	45	22	1	260	8	219			
ω -	20	96	258	192	38	179	21	45	10	87			
თ -		22	19	141	282	94	1	323	89	6			
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300

Incorrectly classified: 7534
accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

Confusion Matrix for eps=0.2 m **True Labels** ω Ó

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300
Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

	Confusion Matrix for eps=0.2												
0 -	- 250	5	239	4	19	164	171	35	25	61			
п -	- 3	564	276	56	10	8	22	121	67	6			
7 -	27	343	213		26	3	8	118	77	5			
m -	0	24	143	238	0	364	1	32	74	113			
True Labels	- 20	108	46	0	287	34	19	128	38	289			
True L	- 12	15	4	278	5	327	47	1	121	72			
9 -	- 66	33	63	4	240	361	145	6	17	2			
~ -	- 5	137	220	88	45	22	1	260	8	219			
ω -	- 20	96	258	192	38	179	21	45	10	87			
6 -	7	22	19	141	282	94	1	323	89	6			
,	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

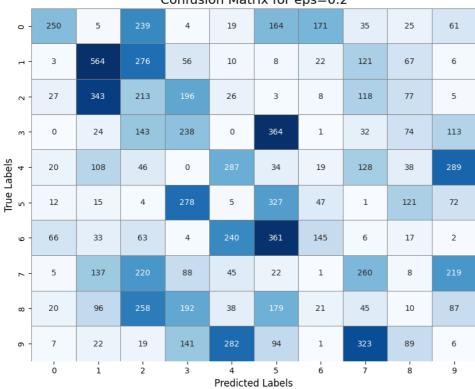
results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300
Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%





results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793
Incorrectly classified: 9041
accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%

Confusion Matrix for eps=0.3

				Jonnusi	on Mati	IX IOI 6	eps=0.5)		
0 -	85	7	291	5	17	287	161	50	22	48
. 1	- 2	120	418	362	36	13	32	48	101	1
2 -	- 28	400		239	30	7	7	123	83	4
m -	0	35	182		0	476	0	37	79	77
abels	16	138	80	7		147	19	192	55	218
True Labels	10	20	24	376	3	185	42	4	151	67
φ-	- 59	50	91	5		401	36	8	20	1
۲ -	- 5	149		148	47	120	1	66	9	194
ω -	15	115	259	196	39	203	12	47	6	54
თ -		29	44	148	255	98	1	333	70	0
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793
Incorrectly classified: 9041

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3 Accuracy on adversarial examples: 8.06%

caracy on adversariat examples. 0.000

	CONTRACTOR TO CPS—0.5												
0 -	- 85	7	291	5	17	287	161	50	22	48			
п.	- 2	120	418	362	36	13	32	48	101	1			
2 -	- 28	400			30	7	7	123	83	4			
m -	0	35	182		0	476	0	37	79	77			
True Labels	- 16	138	80	7		147	19	192	55	218			
True L	- 10	20	24	376	3	185	42	4	151	67			
9 -	- 59	50	91	5		401	36	8	20	1			
7	- 5	149		148	47	120	1	66	9	194			
ω -	- 15	115	259	196	39	203	12	47	6	54			
ი -		29	44	148	255	98	1	333	70	0			
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

 $<\!\!\text{ipython-input-}56-9 fea 085 cbc 61>:66\colon Future \textit{Warning: The frame.append method is}\\$

results_df = results_df.append({
Adversarial test data: eps:0.3

Correctly classified: 793 Incorrectly classified: 9041

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%

Confusion Matrix for eps=0.3 m **True Labels** œ Predicted Labels

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

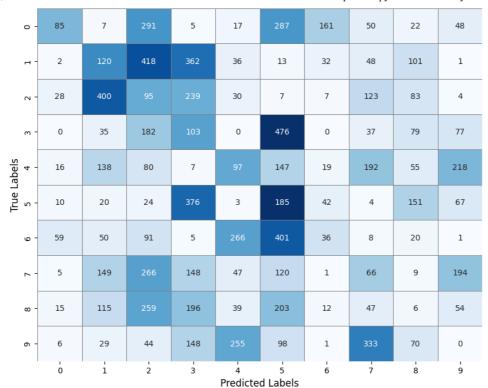
results_df = results_df.append({
Adversarial test data: eps:0.3

Correctly classified: 793 Incorrectly classified: 9041

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%



 $\verb|-ipython-input-56-9fea085cbc61|| : 66: Future \textit{Warning: The frame.append method is}| \\$

Confusion Matrix for eps=0.3

results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793

Incorrectly classified: 9041

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%

Frue Labels

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347

Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%

							1	1 3		,
0 -	12	7	297	8	17	390	137	58	19	28
1	1	5	396	608	11	14	12	2	84	0
2 -	- 26	422	44	263	27	19	7	120	85	3
m -	0	35	204	63	0	573	0	31	67	16
True Labels	- 15	151	116	23	38	255	15	183	63	110
True L	7	17	73	434	1	143	37	3	144	23
9 -	40	64	126	6	233	427	11	8	22	0
^ -	2	152	272	207	47	248	1	28	13	35
œ -	12	142	262	193	32	240	2	44	3	16
ი -		63	100	159	212	105	0	294	46	0
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9

results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Incorrectly classified: 9487

Accuracy on adversarial examples: 3.53%

Confusion Matrix for eps=0.4

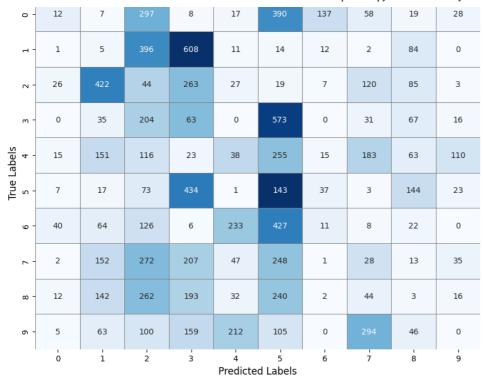
	Contactor Matrix for Cps—0.4											
0 -	12	7		8	17	390	137	58	19	28		
- -	1	5	396	608	11	14	12	2	84	0		
- 2	26	422	44	263	27	19	7	120	85	3		
m -	0	35	204	63	0	573	0	31	67	16		
abels 4	15	151	116	23	38	255	15	183	63	110		
True Labels	7	17	73	434	1	143	37	3	144	23		
9 -	40	64	126	6	233	427	11	8	22	0		
۲ -	2	152	272	207	47	248	1	28	13	35		
∞ -	12	142	262	193	32	240	2	44	3	16		
6 -	- 5	63	100	159	212	105	0		46	0		
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.4 Correctly classified: 347 Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4 Accuracy on adversarial examples: 3.53%



Confusion Matrix for eps=0.4

results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347
Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%

True Labels

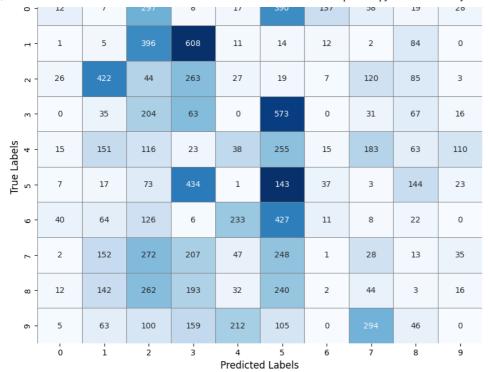
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

4 5 Predicted Labels results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347
Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%



results_df = results_df.append({
Adversarial test data: eps:0.5
Correctly classified: 211
Incorrectly classified: 9623

accuracy_clean:1.0 Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

Confusion Matrix for eps=0.5 **True Labels** í ż Predicted Labels

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.5
Correctly classified: 211
Incorrectly classified: 9623

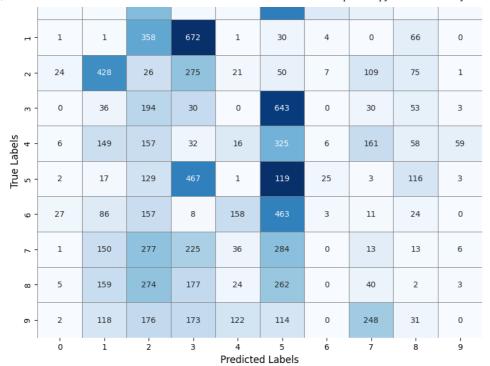
accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

Confusion Matrix for eps=0.5

0 - 1 6 295 15 12 471 94 53 16 10



results_df = results_df.append({
Adversarial test data: eps:0.5

Correctly classified: 211 Incorrectly classified: 9623

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

Confusion Matrix for eps=0.5 m **True Labels** œ ò ż Predicted Labels

<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

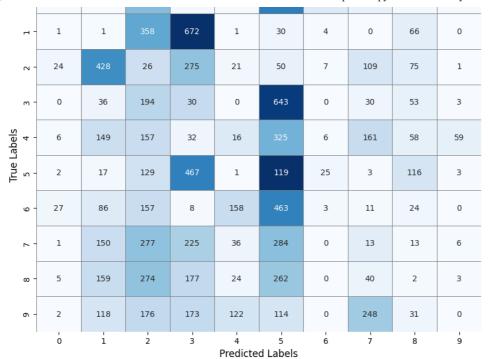
results_df = results_df.append({
Adversarial test data: eps:0.5

Correctly classified: 211 Incorrectly classified: 9623

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%



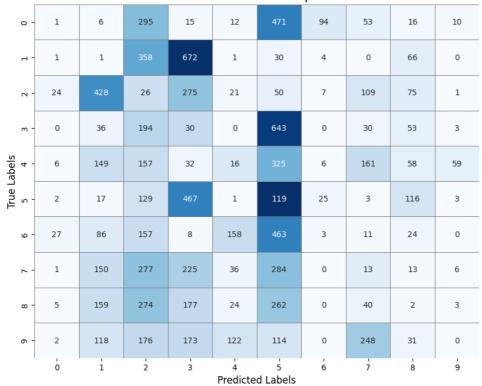
results_df = results_df.append({
Adversarial test data: eps:0.5
Correctly classified: 211
Incorrectly classified: 9623

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

Confusion Matrix for eps=0.5



<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.6

Correctly classified: 161
Incorrectly classified: 9673

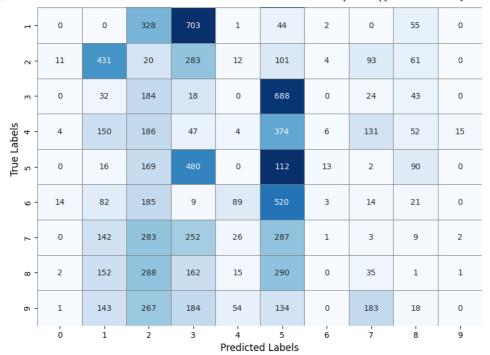
accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

Confusion Matrix for eps=0.6

0 - 0 3 303 24 8 515 58 45 15 2



results_df = results_df.append({
Adversarial test data: eps:0.6
Correctly classified: 161
Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

Confusion Matrix for eps=0.6 **True Labels** ∞ ò i Predicted Labels

 $\verb|-ipython-input-56-9fea085cbc61|| : 66: Future Warning: The frame.append method is \\$

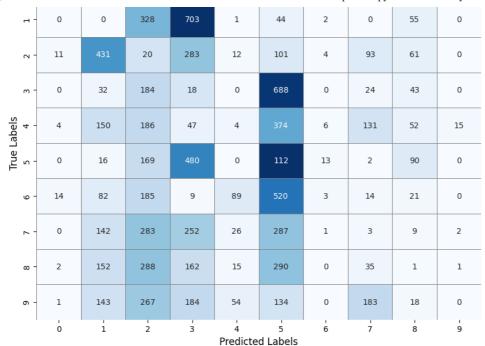
results_df = results_df.append({
Adversarial test data: eps:0.6
Correctly classified: 161
Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

0 -	0	3	303	24	8	515	58	45	15	2



results_df = results_df.append({
Adversarial test data: eps:0.6

Correctly classified: 161 Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

Confusion Matrix for eps=0.6 **True Labels** œ Predicted Labels

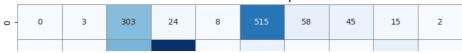
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.6
Correctly classified: 161
Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%



- 1	0	0	328	703	1	44	2	0	55	0
7 -	11	431	20	283	12	101	4	93	61	0
m -	0	32	184	18	0	688	0	24	43	0
True Labels	4	150	186	47	4	374	6	131	52	15
True L	0	16	169	480	0	112	13	2	90	0
9 -	14	82	185	9	89	520	3	14	21	0
7	0	142	283	252	26	287	1	3	9	2
ω -	2	152	288	162	15	290	0	35	1	1
ი -		143	267	184	54	134	0	183	18	0
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

	0	1	2 3	4 5
				Predicted Labels
	eps	attack_num	total correc	t total_adv
0	0.01	1	983	
		2		
1	0.01	2	983	
2	0.01	3	983	
3	0.01	4	983	
4	0.01	5	983	4 9834
5	0.02	1	983	4 9834
6	0.02	2	983	4 9834
7	0.02	3	983	4 9834
8	0.02	4	983	
9	0.02	5	983	
10	0.03	1	983	
11		2		
	0.03	2	983	
12	0.03	3	983	
13	0.03	4	983	
14	0.03	5	983	
15	0.04	1	983	4 9834
16	0.04	2 3	983	4 9834
17	0.04	3	983	4 9834
18	0.04	4	983	4 9834
19	0.04	5	983	
20	0.05	1	983	
21	0.05	2	983	
22	0.05	3	983	
23	0.05	4	983	
24	0.05	5	983	
25	0.10	1	983	
26	0.10	2	983	
27	0.10	3	983	4 9834
28	0.10	4	983	4 9834
29	0.10	5	983	4 9834
30	0.20	1	983	4 9834
31	0.20	2	983	
32	0.20	3	983	
33	0.20	4	983	
34	0.20	5	983	
35	0.30	1		
			983	
36	0.30	2	983	
37	0.30	3	983	
38	0.30	4	983	
39	0.30	5	983	
40	0.40	1	983	4 9834
41	0.40	2	983	4 9834
42	0.40	3	983	4 9834
43	0.40	4	983	4 9834
44	0.40	5	983	
45	0.50	1	983	
46	0.50	2	983	
47	0.50	3	983	
48	0.50	4	983	
49	0.50	5	983	
50	0.60	1	983	
51	0.60	2	983	
52	0.60	3	983	
53	0.60	4	983	4 9834

54 0.60

```
correct_adv_counts
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971,
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
{0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
     {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
     {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
     {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
{0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
10
     {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
     {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
{0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
11
12
     {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
13
14
     {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
     {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
{0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
15
16
     {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
17
18
     {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
     {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
{0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
19
20
     {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
     {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
{0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
22
23
24
     {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
25
     {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
     {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
{0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
26
27
     {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
28
29
     {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
     {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
{0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
30
31
     {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
32
     {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
{0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
{0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
33
35
     {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
     {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
{0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
37
38
39
     {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
40
     {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
          120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
41
     {0:
42
     {0:
     {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
43
     {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
44
     {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
45
     {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
46
     {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
47
          69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
48
     {0:
49
     {0:
     {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
          32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ... 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
51
     {0:
     {0:
     {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209,
                                                              5: ...
     {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is
```

results_df = results_df.append({

9834

```
# Apply the attack to generate adversarial examples
        x_adv = attack.generate(x=correct_examples,y=correct_labels)
        # Predict the labels of the adversarial examples
       y_adv = np.argmax(classifier.predict(x_adv), axis=1)
        nb_correct_adv_pred = np.sum(y_adv == correct_labels)
        print(f"Adversarial test data: eps:{eps}")
        print("Correctly classified: {}".format(nb_correct_adv_pred))
       print("Incorrectly classified: {}".format(len(correct_examples)-nb_correct_adv_pred))
        # For clean examples
        y_pred_clean = np.argmax(classifier.predict(correct_examples), axis=1)
        accuracy_clean = np.sum(y_pred_clean == correct_labels) / len(correct_labels)
        print(f"accuracy_clean:{accuracy_clean}")
        # Inside the loop for each eps
       # You have already calculated this for adversarial examples:
       # nb_correct_adv_pred = np.sum(y_adv == correct_labels)
        # Now calculate the accuracy for adversarial examples
        accuracy_adv = nb_correct_adv_pred / len(correct_labels)
        print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
        print(f"Adversarial test data: eps:{eps}")
        print(f"Accuracy on adversarial examples: {accuracy_adv * 100:.2f}%")
       # Calculate the confusion matrix
       cm = confusion_matrix(correct_labels, y_adv, labels=range(10))
        # Draw and save the confusion matrix
        fig, ax = plt.subplots(figsize=(10, 8))
       # Create a mask for the diagonal elements
       mask = np.eye(len(cm), dtype=bool)
       # Plot the heatmap for off-diagonal elements using the mask
        # Use a professional color palette like 'Blues'
       sns.heatmap(cm, mask=mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, linecol
       # Plot the heatmap for diagonal elements using the inverse of the mask
       # Use the same color palette for consistency
       sns.heatmap(cm, mask=~mask, annot=True, fmt='d', cmap='Blues', ax=ax, cbar=False, linewidths=.5, lineco
        # Labels, title and ticks
        label_names = [f'{i}' for i in range(10)]
        ax.set_xlabel('Predicted Labels', fontsize=12)
        ax.set_ylabel('True Labels', fontsize=12)
        ax.set_title(f'Confusion Matrix for eps={eps}', fontsize=16)
        ax.set_xticklabels(label_names)
       ax.set_yticklabels(label_names)
        image_filename = f'target_confusion_matrix_eps_{eps}_attack_{attack_num}.png'
        plt.savefig(image_filename, bbox_inches='tight')
        plt.show() # Close the figure to avoid displaying it in the notebook
        # Save the results in the DataFrame
        results_df = results_df.append({
            'eps': eps,
            'attack_num': attack_num,
            'total_correct': len(correct_labels),
            'total_adv': len(y_adv),
            'correct_adv_counts': dict(zip(*np.unique(y_adv, return_counts=True)))
        }, ignore_index=True)
# Save the results to a CSV file
results_df.to_csv('/content/drive/MyDrive/ColabNotebooks/5attack_withtruelabel_adv_results.csv', index=False)
# Print the DataFrame
print(results_df)
```

Adversarial test data: eps:0.01 Correctly classified: 9774 Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01 True Labels 5 4 ó í Predicted Labels

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.01 Correctly classified: 9774 Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01											
0 -	971	0	0	0	0	1	0	0	1	0	
ч.	- 0	1132	0	0	0	0	1	0	0	0	
7 -	0	2		1	0	0	0	2	1	1	
m -	0	1	1	986	0	0	0	0	0	1	
True Labels	0	0	2	0	962	0	1	0	0	4	
True L	- 0	0	1	1	0	880	0	0	0	0	
φ-	- 0	0	0	1	2	1	933	0	0	0	
7	0		1	0	0	1	0	1000	0	0	
ω -	0	2	3	1	4	2	2	1	926	5	
o -	0	1	0	0	3	0	0	5	0	975	
	Ó	í	2	3	4 Predicte	5ٰ d Labels	6	7	8	9	

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is results_df = results_df.append({

Adversarial test data: eps:0.01 Correctly classified: 9774 Incorrectly classified: 60 accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01 m **True Labels** œ ò

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.01
Correctly classified: 9774
Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01											
0 -	971	0	0	0	0	1	0	0	1	0	
г.	- 0	1132	0	0	0	0	1	0	0	0	
2 -	- 0	2		1	0	0	0	2	1	1	
m -	- 0	1	1	986	0	0	0	0	0	1	
True Labels	- 0	0	2	0	962	0	1	0	0	4	
True L	- 0	0	1	1	0	880	0	0	0	0	
9 -	- 0	0	0	1	2	1	933	0	0	0	
۲.	- 0		1	0	0	1	0	1000	0	0	
ω -	- 0	2	3	1	4	2	2	1	926	5	
o -		1	0	0		0	0	5	0	975	
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({
 Adversarial test data: ens:0 01

Muversariar rest darar ebs.n.ni Correctly classified: 9774 Incorrectly classified: 60

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01												
0 -	971	0	0	0	0	1	0	0	1	0		
1	- 0	1132	0	0	0	0	1	0	0	0		
2 -	0	2		1	0	0	0	2	1	1		
m -	0	1	1	986	0	0	0	0	0	1		
True Labels	0	0	2	0	962	0	1	0	0	4		
True L	0	0	1	1	0	880	0	0	0	0		
9 -	0	0	0	1	2	1	933	0	0	0		
7	0		1	0	0	1	0	1000	0	0		
ω -	0	2		1	4	2	2	1	926	5		
ი -	0	1	0	0		0	0	5	0	975		
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.02 Correctly classified: 9692 Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02											
0	- 968	0	1	0	0	2	0	0	2	0	
-	- 0	1132	0	0	0	0	1	0	0	0	
^	- 1	8	999	1	1	1	0	3	1	1	
m	- 0	1	2	979	0	4	0	2	0	1	
True Labels	- 0	2	3	0	956	0	2	0	0	6	
True L	- 1	1	1	1	0	877	1	0	0	0	
9	- 0	0	1	1		2	928	0	0	0	
_	- 0	5		1	0	1	0	987	0	5	
00	- 4		9			3	3	1	903	6	
6		2	0	1	4	1	0	10	2	963	
	0 1 2 3 4 5 6 7 8 9 Predicted Labels										

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is results_df = results_df.append({

Adversarial test data: eps:0.02

Correctly classified: 9692 Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02

	Confusion Matrix for eps=0.02											
0 -	968	0	1	0	0	2	0	0	2	0		
- 1	0	1132	0	0	0	0	1	0	0	0		
2 -	1	8	999	1	1	1	0	3	1	1		
m -	0	1	2	979	0	4	0	2	0	1		
True Labels	0	2	3	0	956	0	2	0	0	6		
True L	1	1	1	1	0	877	1	0	0	0		
9 -	0	0	1	1		2	928	0	0	0		
7 -	0			1	0	1	0	987	0	5		
ω -	4		9			3	3	1	903	6		
6 -	1	2	0	1	4	1	0	10	2	963		
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

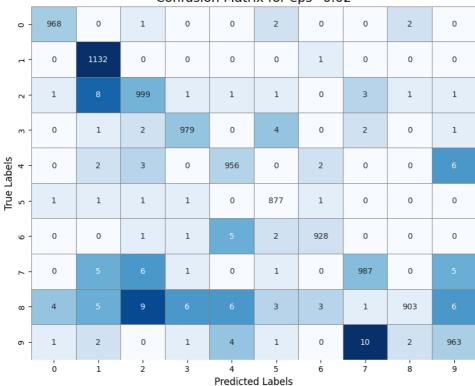
results_df = results_df.append({
Adversarial test data: eps:0.02
Correctly classified: 9692
Incorrectly classified: 142
accuracy classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02



<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.02

Correctly classified: 0602

COLLECTTÀ CTOPPETTEM POST Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02											
0 -	- 968	0	1	0	0	2	0	0	2	0	
г.	- 0	1132	0	0	0	0	1	0	0	0	
۸.	1	8	999	1	1	1	0	3	1	1	
m ·	0	1	2	979	0	4	0	2	0	1	
True Labels	- 0	2	3	0	956	0	2	0	0	6	
True L	- 1	1	1	1	0	877	1	0	0	0	
9 -	- 0	0	1	1		2	928	0	0	0	
7	- 0			1	0	1	0	987	0	5	
ω ·	- 4		9			3	3	1	903	6	
σ.		2	0	1	4	1	0	10	2	963	
	Ó	i	2	3	4 Predicte	် d Labels	6	7	8	9	

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

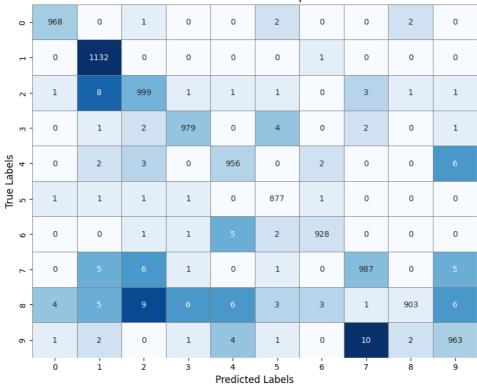
results_df = results_df.append({ Adversarial test data: eps:0.02 Correctly classified: 9692 Incorrectly classified: 142

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.02

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02



<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is results_df = results_df.append({

Adversarial test data: eps:0.03 Correctly classified: 9550

Incorrectly classified: 284

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03 True Labels 5 4

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.03
Correctly classified: 9550
Incorrectly classified: 284

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03												
0 -	966	0	2	0	1	2	0	0	2	0		
	- 0	1131	0	0	0	1	1	0	0	0		
2	- 3	13	980	7	1	1	1	5	4	1		
m -	- 0	1	5	971	0	6	0	4	1	1		
True Labels	- 1	4	3	0	943	0	3	1	1	13		
True L	1	3	1	2	1	871	2	0	1	0		
9 -	2	1	1	1	10	8	912	1	1	0		
۲.	- 0	10	6	2	0	2	0	975	1	9		
ω -	7	10	12	9	9	13	5	2	866	13		
ი -	- 4	4	1	4	6	5	0	21	4	935		
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
results df = results df append()

results_df = results_df.append({
Adversarial test data: eps:0.03
Correctly classified: 9550

incorrectly classified: 284

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03

Confusion Matrix for eps=0.03											
0 -	966	0	2	0	1	2	0	0	2	0	
1	0	1131	0	0	0	1	1	0	0	0	
7 -	- 3	13	980	7	1	1	1	5	4	1	
m -	0	1	5	971	0	6	0	4	1	1	
True Labels	1	4	3	0	943	0	3	1	1	13	
True L	1	3	1	2	1	871	2	0	1	0	
9 -	- 2	1	1	1	10	8	912	1	1	0	
7 -	0	10	6	2	0	2	0	975	1	9	
ω -	. 7	10	12	9	9	13	5	2	866	13	
6 -	4	4	1	4	6	5	0	21	4	935	
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
results_df = results_df.append({

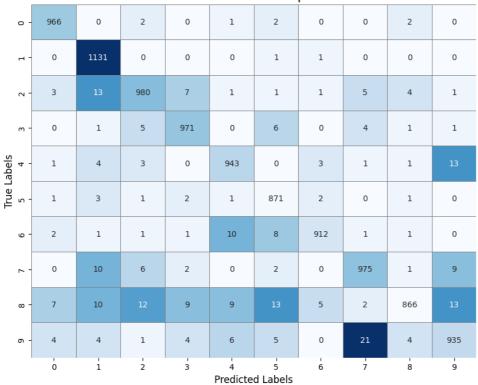
results_df = results_df.append({ Adversarial test data: eps:0.03 Correctly classified: 9550 Incorrectly classified: 284

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03



<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.03 Correctly classified: 9550 Incorrectly classified: 284

accuracy_clean:1.0 Accuracy on clean data: 100.00% Adversarial test data: eps:0.03

Accuracy on adversarial examples: 97.11%

Confusion Matrix for eps=0.03											
0	- 966	0	2	0	1	2	0	0	2	0	
1	- 0	1131	0	0	0	1	1	0	0	0	
7	- 3	13	980	7	1	1	1	5	4	1	
м	- 0	1	5	971	0	6	0	4	1	1	
abels 4	- 1	4	3	0	943	0	3	1	1	13	
True Labels	- 1	3	1	2	1	871	2	0	1	0	
9	- 2	1	1	1	10	8	912	1	1	0	
7	- 0	10	6	2	0	2	0	975	1	9	
00	- 7	10	12	9	9	13	5	2	866	13	
6	- 4	4	1	4	6	5	0	21	4	935	
	Ó	'n	2	3	4	5	6	7	8	9	

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({ Adversarial test data: eps:0.04 Correctly classified: 9394 Incorrectly classified: 440

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04											
0	- 962	0	3	0	1	2	1	0	3	1	
-	- 0	1129	0	0	0	1	1	1	1	0	
2	- 5	16	966	9	3	1	1	9	4	2	
m	- 0	2	10	952	0	12	0	5	3	5	
True Labels	- 1	6	5	0	932	0	4	1	2	18	
True L	- 2	3	1	7	1	863	3	0	1	1	
9	- 5	3	2	1	11	14	899	1	1	0	
7	- 0	14	10	2	4	2	0	958	2	13	
00	- 7	14	25		11		7	4	824	19	
თ		4	2	6	14	11	0	30	4	909	
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.04 Correctly classified: 9394 Incorrectly classified: 440

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04

	Confusion Matrix for eps=0.04											
0 -	962	0	3	0	1	2	1	0	3	1		
п-	- 0	1129	0	0	0	1	1	1	1	0		
7 -	- 5	16	966	9	3	1	1	9	4	2		
m -	0	2	10	952	0	12	0	5	3	5		
True Labels	1	6	5	0	932	0	4	1	2	18		
True L	- 2	3	1	7	1	863	3	0	1	1		
9 -	- 5	3	2	1	11	14	899	1	1	0		
7	. 0	14	10	2	4	2	0	958	2	13		
ω -	7	14	25		11		7	4	824	19		
6 -		4	2	6	14	11	0	30	4	909		
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

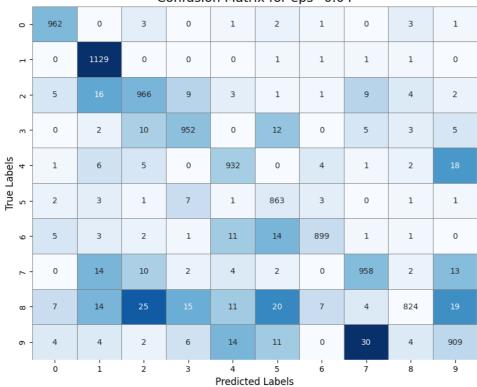
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.04
Correctly classified: 9394
Incorrectly classified: 440
accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04



<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.04
Correctly classified: 9394
Incorrectly classified: 440

accuracv clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

	Confusion Matrix for eps=0.04													
0 -	962	0	3	0	1	2	1	0	3	1				
	- 0	1129	0	0	0	1	1	1	1	0				
2 -	- 5	16	966	9	3	1	1	9	4	2				
m -	- 0	2	10	952	0	12	0	5	3	5				
True Labels	1	6	5	0	932	0	4	1	2	18				
True L	- 2	3	1	7	1	863	3	0	1	1				
φ-	- 5	3	2	1	11	14	899	1	1	0				
7	0	14	10	2	4	2	0	958	2	13				
ω -	7	14	25		11		7	4	824	19				
ი -		4	2	6	14	11	0	30	4	909				
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9				

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.04 Correctly classified: 9394 Incorrectly classified: 440

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04

	Confusion Matrix for eps=0.04												
0 -	962	0	3	0	1	2	1	0	3	1			
г.	- 0	1129	0	0	0	1	1	1	1	0			
2 -	- 5		966	9	3	1	1	9	4	2			
m -	- 0	2	10	952	0	12	0	5	3	5			
True Labels	1	6	5	0	932	0	4	1	2	18			
True L	- 2	3	1	7	1	863	3	0	1	1			
φ-	- 5	3	2	1	11	14	899	1	1	0			
۲.	0	14	10	2	4	2	0	958	2	13			
ω -	7	14	25		11		7	4	824	19			
ი -		4	2	6	14	11	0	30	4	909			
	Ó	í	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05

	Confusion Matrix for eps=0.05												
0 -	955	0	4	0	1	5	3	0	3	2			
г.	0	1126	1	1	0	1	2	1	1	0			
2 -	. 7	30	941	13	4	1	1	13	4	2			
m -	0	3	16	926	0	25	0	5	4	10			
True Labels	- 2	12	6	0	918	0	4	2	3	22			
True L	- 2	4	1	12	1	850	6	0	3	3			
9 -	7	4	4	1	13	25	880	1	2	0			
7	0		17	4	7	2	0		2	18			
ω -	- 8	22	37	27	16	37	9	9	753	28			
ი -		6	3	14	40	17	0	41	10	848			
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05

	Confusion Matrix for eps=0.05												
0 -		0	4	0	1	5	3	0	3	2			
г -	0	1126	1	1	0	1	2	1	1	0			
- 2	7	30	941	13	4	1	1	13	4	2			
m -	0	3	16	926	0	25	0	5	4	10			
abels 4	2	12	6	0	918	0	4	2	3	22			
True Labels	2	4	1	12	1	850	6	0	3	3			
9 -	7	4	4	1	13	25	880	1	2	0			
۲ -	0		17	4	7	2	0		2	18			
ω -	8	22	37	27	16	37	9	9	753	28			
ი -		6	3	14	40	17	0	41	10	848			
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00%

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

	Confusion Matrix for eps=0.05													
0 -	955	0	4	0	1	5	3	0	3	2				
н.	- 0	1126	1	1	0	1	2	1	1	0				
2	- 7	30	941	13	4	1	1	13	4	2				
m -	- 0	3	16	926	0	25	0	5	4	10				
True Labels	- 2	12	6	0	918	0	4	2	3	22				
True L	- 2	4	1	12	1	850	6	0	3	3				
9 -	7	4	4	1	13	25	880	1	2	0				
۲.	- 0		17	4	7	2	0		2	18				
ω -	- 8	22	37	27	16	37	9	9	753	28				
ი -	- 5	6	3	14	40	17	0	41	10	848				
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9				

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05 **True Labels** Predicted Labels

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is results_df = results_df.append({

Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702

accuracy_clean:1.0

Accuracy on clean data: 100.00%

Adversarial test data: eps:0.05 $\,$

Accuracy on adversarial examples: 92.86%

Confusion Matrix for eps=0.05

	Confusion Matrix for eps=0.05												
0 -		0	4	0	1	5	3	0	3	2			
п-	0	1126	1	1	0	1	2	1	1	0			
7 -	. 7	30	941	13	4	1	1	13	4	2			
m -	0	3	16	926	0	25	0	5	4	10			
True Labels	- 2	12	6	0	918	0	4	2	3	22			
True L	- 2	4	1	12	1	850	6	0	3	3			
9 -	. 7	4	4	1	13	25	880	1	2	0			
7	- 0		17	4	7	2	0		2	18			
ω -	- 8	22	37	27	16	37	9	9	753	28			
ი -		6	3	14	40	17	0	41	10	848			
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.1
Correctly classified: 6834
Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

Confusion Matrix for eps=0.1

	Confusion Matrix for eps=0.1												
0 -	847	1	26	2	7	36	24	6	11	13			
1	0	1097	13	4	2	2	9	3	3	0			
2 -	17	137	675	88	13	1	4	52	24	5			
m -	1	10	62		0	129	1	19	32	35			
True Labels	7	51	20	0	738	0	11	26	7	109			
True L	7	6	1	78	2	725	21	1	23	18			
9 -	29	17	20	2	58	158	642	2	8	1			
7 -	2	56	81	17	23	6	1	737	4	78			
ω -	17	65	175	151	35	121	20	32	252	78			
ი -		12	8	51	192	55	1	195	43	421			
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.1
Correctly classified: 6834

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: ens:0.1

Incorrectly classified: 3000

Accuracy on adversarial examples: 69.49%

	Confusion Matrix for eps=0.1													
0 -	847	1	26	2	7	36	24	6	11	13				
г.	- 0	1097	13	4	2	2	9	3	3	0				
2	- 17	137	675	88	13	1	4	52	24	5				
m -	- 1	10	62		0	129	1	19	32	35				
True Labels	7	51	20	0	738	0	11	26	7	109				
True L	7	6	1	78	2	725	21	1	23	18				
9 -	29	17	20	2	58	158	642	2	8	1				
۲.	- 2	56	81	17	23	6	1	737	4	78				
œ -	- 17	65	175	151	35	121	20	32	252	78				
o -		12	8	51	192	55	1	195	43	421				
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9				

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.1 Correctly classified: 6834 Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

Confusion Matrix for eps=0.1 m **True Labels** ω Predicted Labels

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.1 Correctly classified: 6834 Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

	Confusion Matrix for eps=0.1													
0 -	847	1	26	2	7	36	24	6	11	13				
1	- 0	1097	13	4	2	2	9	3	3	0				
2 -	- 17	137	675	88	13	1	4	52	24	5				
m -	- 1	10	62		0	129	1	19	32	35				
True Labels	7	51	20	0	738	0	11	26	7	109				
True L	7	6	1	78	2	725	21	1	23	18				
9 -	- 29	17	20	2	58	158	642	2	8	1				
7	- 2	56	81	17	23	6	1	737	4	78				
ω -	17	65	175	151	35	121	20	32	252	78				
ი -	- 6	12	8	51	192	55	1	195	43	421				

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.1
Correctly classified: 6834
Incorrectly classified: 3000

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.1

Accuracy on adversarial examples: 69.49%

Confusion Matrix for eps=0.1													
0 -	847	1	26	2	7	36	24	6	11	13			
1	0	1097	13	4	2	2	9	3	3	0			
2 -	17	137	675	88	13	1	4	52	24	5			
m -	1	10	62		0	129	1	19	32	35			
True Labels	7	51	20	0	738	0	11	26	7	109			
True L	7	6	1	78	2	725	21	1	23	18			
9 -	- 29	17	20	2	58	158	642	2	8	1			
7	2	56	81	17	23	6	1	737	4	78			
ω -	17	65	175	151	35	121	20	32	252	78			
ი -		12	8	51	192	55	1	195	43	421			
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.2 Correctly classified: 2300 Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial evamples: 23 30%

Accuracy on auversariac examples. 23:350

	Confusion Matrix for eps=0.2												
0 -	250	5	239	4	19	164	171	35	25	61			
1	3	564	276	56	10	8	22	121	67	6			
2 -	27	343	213		26	3	8	118	77	5			
m -	0	24	143	238	0	364	1	32	74	113			
True Labels	- 20	108	46	0	287	34	19	128	38	289			
True L	- 12	15	4	278	5	327	47	1	121	72			
9 -	- 66	33	63	4	240	361	145	6	17	2			
7 -	- 5	137	220	88	45	22	1	260	8	219			
œ -	20	96	258	192	38	179	21	45	10	87			
6 -	7	22	19	141	282	94	1	323	89	6			
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300
Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

Confusion Matrix for eps=0.2

	Comusion Fluctive for Cp3=0.2												
0 -	250	5	239	4	19	164	171	35	25	61			
1 -	3	564	276	56	10	8	22	121	67	6			
2 -	27	343	213		26	3	8	118	77	5			
m -	0	24	143	238	0	364	1	32	74	113			
True Labels	20	108	46	0	287	34	19	128	38	289			
True L	12	15	4	278	5	327	47	1	121	72			
9 -	66	33	63	4	240	361	145	6	17	2			
7	5	137	220	88	45	22	1	260	8	219			
ω -	20	96	258	192	38	179	21	45	10	87			
ი -		22	19	141	282	94	1	323	89	6			
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300
Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

Confusion Matrix for eps=0.2 m **True Labels** ω Ó Predicted Labels

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300
Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

	Confusion Matrix for eps=0.2													
0 -	- 250	5	239	4	19	164	171	35	25	61				
- 13	3	564	276	56	10	8	22	121	67	6				
7 -	27	343	213		26	3	8	118	77	5				
m -	0	24	143	238	0	364	1	32	74	113				
True Labels	20	108	46	0	287	34	19	128	38	289				
True L	12	15	4	278	5	327	47	1	121	72				
9 -	66	33	63	4	240	361	145	6	17	2				
۲ -	5	137	220	88	45	22	1	260	8	219				
∞ -	20	96	258	192	38	179	21	45	10	87				
6 -	7	22	19	141	282	94	1	323	89	6				
,	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9				

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

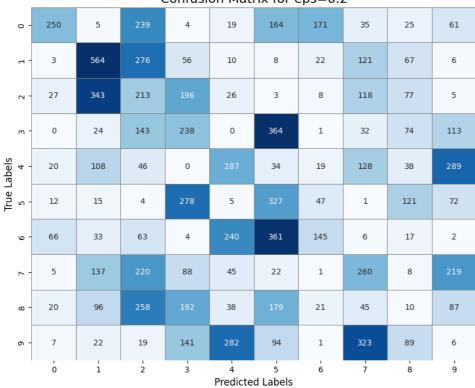
results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300
Incorrectly classified: 7534

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.2

Accuracy on adversarial examples: 23.39%

Confusion Matrix for eps=0.2



<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793
Incorrectly classified: 9041
accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%

Confusion Matrix for eps=0.3

	Confusion Matrix for eps—0.5													
0 -	85	7	291	5	17	287	161	50	22	48				
- -	2	120	418	362	36	13	32	48	101	1				
- 2	28	400		239	30	7	7	123	83	4				
m -	0	35	182		0	476	0	37	79	77				
True Labels	16	138	80	7		147	19	192	55	218				
True L	10	20	24	376	3	185	42	4	151	67				
9 -	59	50	91	5		401	36	8	20	1				
۲ -	5	149		148	47	120	1	66	9	194				
ω -	15	115	259	196	39	203	12	47	6	54				
o -		29	44	148	255	98	1	333	70	0				
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9				

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793
Incorrectly classified: 9041

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%

	Contrasion matrix for cps-0.5													
0 -	- 85	7	291	5	17	287	161	50	22	48				
2 .	- 2	120	418	362	36	13	32	48	101	1				
	- 28	400			30	7	7	123	83	4				
m -	0	35	182		0	476	0	37	79	77				
True Labels	- 16	138	80	7		147	19	192	55	218				
True L	- 10	20	24	376	3	185	42	4	151	67				
9 -	- 59	50	91	5		401	36	8	20	1				
7	- 5	149		148	47	120	1	66	9	194				
ω -	- 15	115	259	196	39	203	12	47	6	54				
ი -		29	44	148	255	98	1	333	70	0				
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9				

results_df = results_df.append({
dversarial test data: ens:0.3

Adversarial test data: eps:0.3 Correctly classified: 793 Incorrectly classified: 9041

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%

Confusion Matrix for eps=0.3 m **True Labels** œ Predicted Labels

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793

Incorrectly classified: 9041

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Accuracy on adversarial examples: 8.06%



results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.3

Incorrectly classified: 9041

Accuracy on adversarial examples: 8.06%

Confusion Matrix for eps=0.3 **Frue Labels**

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

Predicted Labels

results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347

Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%

, 10.00	countries (Fig. 1)										
0 -	12	7	297	8	17	390	137	58	19	28	
. 1	1	5	396	608	11	14	12	2	84	0	
2 -	- 26	422	44	263	27	19	7	120	85	3	
m -	0	35	204	63	0	573	0	31	67	16	
True Labels	15	151	116	23	38	255	15	183	63	110	
True L	7	17	73	434	1	143	37	3	144	23	
φ-	40	64	126	6	233	427	11	8	22	0	
۲ -	- 2	152	272	207	47	248	1	28	13	35	
ω -	12	142	262	193	32	240	2	44	3	16	
ი -		63	100	159	212	105	0	294	46	0	
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347
Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%

Confusion Matrix for eps=0.4

	Comusion Matrix for eps—0.4												
0 -	12	7		8	17	390	137	58	19	28			
- 13	1	5	396	608	11	14	12	2	84	0			
7 -	26	422	44	263	27	19	7	120	85	3			
m -	0	35	204	63	0	573	0	31	67	16			
abels 4	15	151	116	23	38	255	15	183	63	110			
True Labels	7	17	73	434	1	143	37	3	144	23			
9 -	40	64	126	6	233	427	11	8	22	0			
۲ -	2	152	272	207	47	248	1	28	13	35			
∞ -	12	142	262	193	32	240	2	44	3	16			
6 -	- 5	63	100	159	212	105	0		46	0			
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

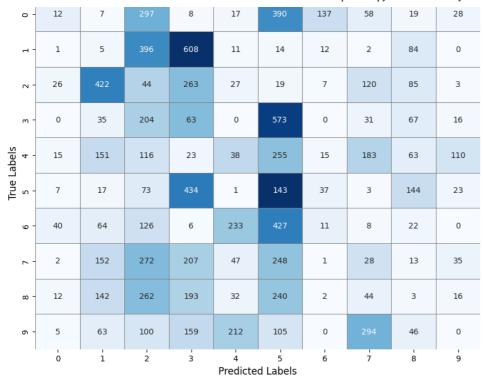
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
 results_df = results_df.append({

Adversarial test data: eps:0.4 Correctly classified: 347 Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%



results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347
Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%

Confusion Matrix for eps=0.4 **True Labels Predicted Labels**

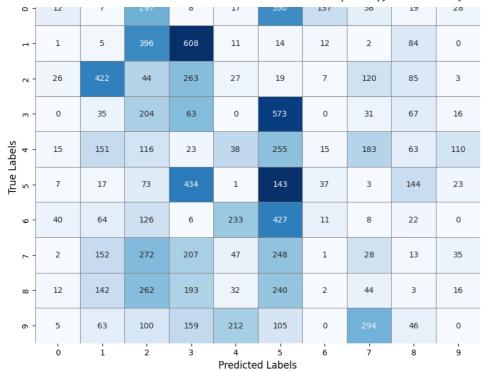
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347
Incorrectly classified: 9487

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%



results_df = results_df.append({
Adversarial test data: eps:0.5
Correctly classified: 211
Incorrectly classified: 9623

accuracy_clean:1.0 Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

Confusion Matrix for eps=0.5 **True Labels** í Predicted Labels

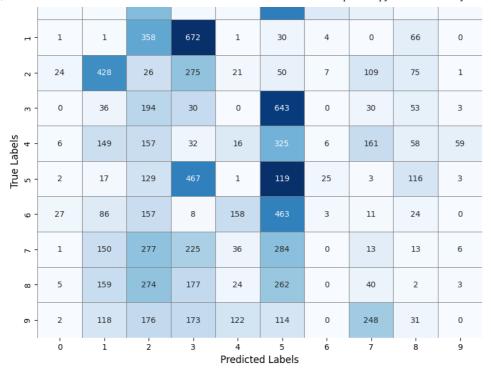
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.5
Correctly classified: 211
Incorrectly classified: 9623
accuracy class:1 0

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%



results_df = results_df.append({
Adversarial test data: eps:0.5

Correctly classified: 211 Incorrectly classified: 9623

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

Confusion Matrix for eps=0.5 m **True Labels** œ ò ż Predicted Labels

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.5

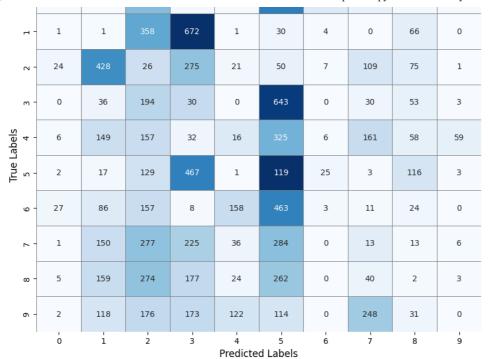
Correctly classified: 211 Incorrectly classified: 9623

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

0 -	1	6	295	15	12	471	94	53	16	10
_	_	_								



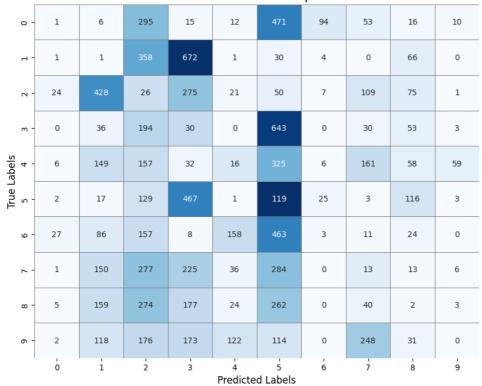
results_df = results_df.append({
Adversarial test data: eps:0.5
Correctly classified: 211
Incorrectly classified: 9623

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.5

Accuracy on adversarial examples: 2.15%

Confusion Matrix for eps=0.5



<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.6

Correctly classified: 161 Incorrectly classified: 9673

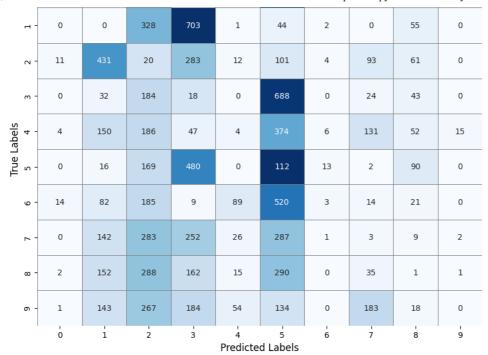
accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

Confusion Matrix for eps=0.6

0 - 0 3 303 24 8 515 58 45 15 2



results_df = results_df.append({
Adversarial test data: eps:0.6
Correctly classified: 161
Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

Confusion Matrix for eps=0.6 **True Labels** ∞ ò i Predicted Labels

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

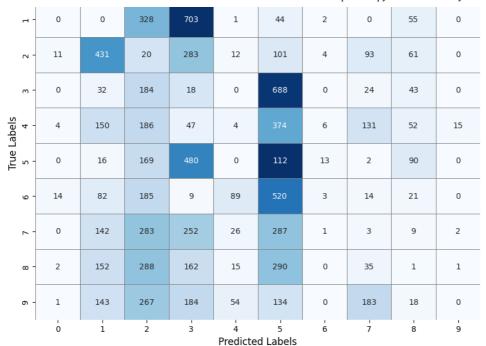
results_df = results_df.append({
Adversarial test data: eps:0.6
Correctly classified: 161
Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

0 -	0	3	303	24	8	515	58	45	15	2
							I			



results_df = results_df.append({
Adversarial test data: eps:0.6

Correctly classified: 161 Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%

Confusion Matrix for eps=0.6 **True Labels** œ Predicted Labels

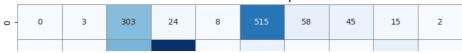
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.6
Correctly classified: 161
Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6

Accuracy on adversarial examples: 1.64%



г -	0	0	328	703	1	44	2	0	55	0
2 -	- 11	431	20	283	12	101	4	93	61	0
m -	0	32	184	18	0	688	0	24	43	0
True Labels	4	150	186	47	4	374	6	131	52	15
True L	0	16	169	480	0	112	13	2	90	0
9 -	14	82	185	9	89	520	3	14	21	0
7	0	142	283	252	26	287	1	3	9	2
ω -	2	152	288	162	15	290	0	35	1	1
ი -		143	267	184	54	134	0	183	18	0
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9

0.60

```
54 0.60
                                         9834
                                               correct_adv_counts
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
      {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971,
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
     {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
{0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
     {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
     {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
     {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972,
                                             990, 4: 972, 5: ...
10
     {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
     {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
{0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
11
12
     {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
13
14
     {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
     {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
{0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
15
16
     {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
17
18
     {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
     {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
{0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
19
20
21
     {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
     {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
{0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
22
23
24
     {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
25
     {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
     {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
{0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
26
27
     {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
28
29
     {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
     {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
{0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
30
31
     {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
32
     {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
{0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
{0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
33
35
     {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
     {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
{0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
37
38
39
     {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
40
     {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
          120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
41
     {0:
42
     {0:
     {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
43
     {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
44
     {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
45
     {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
46
     {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
47
          69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
48
     {0:
49
     {0:
     {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
          32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ... 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
51
     {0:
     {0:
     {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209,
                                                              5: ...
     {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is
   results_df = results_df.append({
```