Install Library

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

▼ Pre-trained model--Save Correct Examples and Labels

```
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 model
# 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_updates` will be
      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_updates` will be
      updates = self.state_updates
     Test accuracy: 0.983
```

Count the occurrences of each label in the test set

```
# Count the occurrences of each label in the test set
unique, counts = np.unique(y_test, 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: 980, 1: 1135, 2: 1032, 3: 1010, 4: 982, 5: 892, 6: 958, 7: 1028, 8: 974, 9: 1009}
    Label 0: 980
    Label 1: 1135
    Label 2: 1032
    Label 3: 1010
    Label 4: 982
```

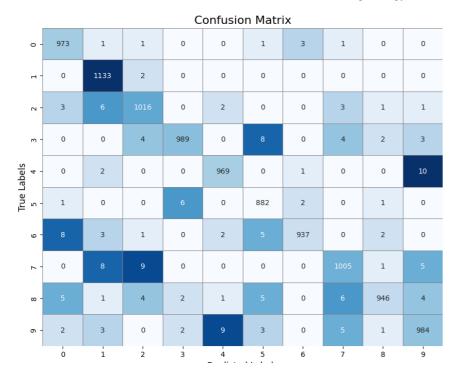
Label 5: 892 Label 6: 958 Label 7: 1028 Label 8: 974 Label 9: 1009

Count the occurrences of each predicted_labels in the set

```
# Count the occurrences of each label in the predicted_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}")
     correct_predicted set label counts: {0: 992, 1: 1157, 2: 1037, 3: 999, 4: 983, 5: 904, 6: 943, 7: 1024, 8: 954, 9: 1007}
     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
     Total count of all total_predicted_labels_count combined in the MNIST dataset: 10000
```

Calculate the confusion matrix of Pretrained Model

```
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='grey')
# 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)
ax.set_xticklabels(label_names)
ax.set_yticklabels(label_names)
plt.savefig('model.png', bbox_inches='tight')
plt.show()
```



Check Correct Examples Accuracy

```
# 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)
print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
```

Accuracy on clean data: 100.00%

Count the occurrences of each correct label in set

```
# 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
predictions.shape
(10000, 10)
```

FastGradientMethod-- eps_range = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6] Without Label

```
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
```

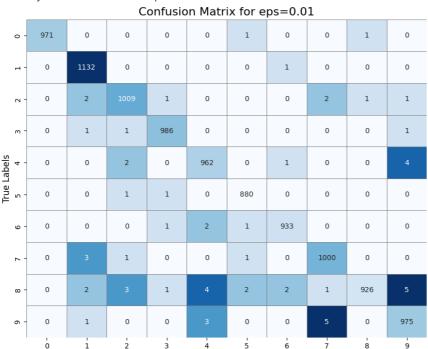
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
# Disable eager execution (necessary for ART with TensorFlow v1)
tf.compat.v1.disable_eager_execution()
# Load your trained model
model = load_model('/content/drive/MyDrive/ColabNotebooks/mnist_model.h5')
# Load the saved correct 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')
# 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='grey')
    # Plot the heatmap for diagonal elements using the inverse of the mask
    # Use the same color palette for consistency
    sns heatman(cm mask=-mask annot=True fmt='d' cman='Rlues' ay=ay char=False linewidths= 5 linecolor='drey')
```

```
STISTICAL CHAPTERS IN HASTER - HASTER AND CONTROL OF THE A CHAPTER DEAD TANADAY CONTROL OF THE C
            # 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%



 $\verb|-ipython-input-64-18fe375a8b4b|| \verb|:107: Future Warning: The frame.append method is a property of the prop$

Predicted Labels

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%

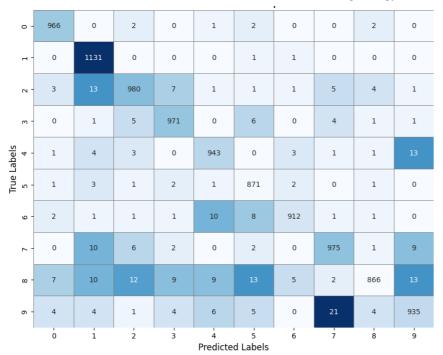
Confusion Matrix for eps=0.02 **True Labels** i ġ Predicted Labels

<ipython-input-64-18fe375a8b4b>:107: 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 or



<ipython-input-64-18fe375a8b4b>:107: 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

	Comusion Flactive to the contractive contr										
0 -	962	0	3	0	1	2	1	0	3	1	
- 17	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	
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	
∞ -	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-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is

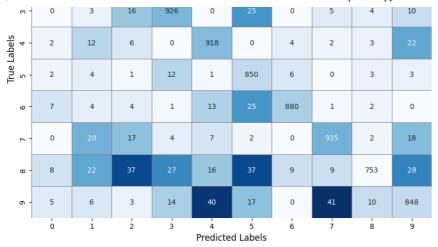
results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702 accuracy_clean:1.0

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

Accuracy on adversarial examples: 92.86%

Confusion	Matrix	for e	ps=0.	05
-----------	--------	-------	-------	----

	Confusion Matrix for eps=0.05												
0 -		0	4	0	1	5	3	0	3	2			
٦ -	0	1126	1	1	0	1	2	1	1	0			
7 -	7			13	4	1	1	13	4	2			



<ipython-input-64-18fe375a8b4b>:107: 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

	Confidence 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		
7 -	17	137		88	13	1	4	52	24	5		
m -	1	10	62		0	129	1	19	32	35		
True Labels	7	51	20	0		0	11	26	7	109		
True L	7	6	1	78	2		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		
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

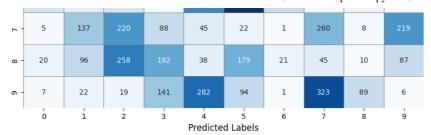
<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method is

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		
. 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		
abels	- 20	108	46	0		34	19	128	38	289		
True Labels	- 12	15	4	278	5		47	1	121	72		
9 -	- 66	33	63	4	240	361	145	6	17	2		



<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method i: 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=0.3												
0 -	85	7		5	17	287	161	50	22	48			
н -	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			
۲ -	5	149		148	47	120	1	66	9	194			
ω -	15	115		196	39	203	12	47	6	54			
6 -	6	29	44	148		98	1	333	70	0			
	Ö	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

<ipython-input-64-18fe375a8b4b>:107: 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										
0 -	- 12	7		8	17	390	137	58	19	28	
н.	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	
7	- 2	152	272	207	47	248	1	28	13	35	
ω ·	- 12	142	262	193	32	240	2	44	3	16	
σ-		63	100	159	212	105	0		46	0	
	Ó	'n	2	3	4	5	6	7	8	9	

Predicted Labels

<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method i:
 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												
			(ontusi	on Mati	rix tor e	eps=u.:)				
0 -	1	6	295	15	12	471	94	53	16	10		
г -	1	1		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		
True Labels	6	149	157	32	16	325	6	161	58	59		
True L	2	17	129	467	1	119	25	3	116	3		
φ-	27	86	157	8	158	463	3	11	24	0		
r -	1	150	277	225	36	284	0	13	13	6		
ω -	- 5	159	274	177	24	262	0	40	2	3		
ი -	2	118	176	173	122	114	0	248	31	0		
	ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

<ipython-input-64-18fe375a8b4b>:107: FutureWarning: The frame.append method i:
 results_df = results_df.append({
 Adversarial test data: eps:0.6

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%											
			ı	(Confusi	on Mati	rix for e	eps=0.6	5		
	0 -	0	3	303	24	8	515	58	45	15	2
	г-	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 -	4	150	186	47	4	374	6	131	52	15
True I	٥ -	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 -	1	143	267	184	54	134	0	183	18	0
		0	i	2	3	4 Predicte	5 d Labels	6	7	8	9
0 1 2	1 0.02 9834 9834										

eps	total_correct	total_adv
0.01	9834	9834
0.02	9834	9834
0.03	9834	9834
0.04	9834	9834
0.05	9834	9834
0.10	9834	9834
0.20	9834	9834
	0.01 0.02 0.03 0.04 0.05 0.10	0.02 9834 0.03 9834 0.04 9834 0.05 9834 0.10 9834

```
9834
                                9834
    0.30
                                9834
8
    0.40
                     9834
                                9834
    0.50
                     9834
10 0.60
                                9834
                                         correct_adv_counts
    {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
    {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
     {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: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
    {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
    {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:...
{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
  results_df = results_df.append({
```

FastGradientMethod-- eps_range = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6] With Label

```
# 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='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='grey')
   # 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											
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		
м	- 0	1	1	986	0	0	0	0	0	1		
True Labels 5 4	- 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		2	2	1	926	5		
6	- 0	1	0	0		0	0		0	975		
	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:0.02

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%

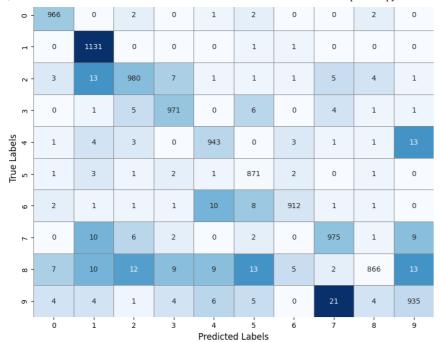
Confusion Matrix for eps=0.02 m **True Labels** ò Predicted Labels

<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%



<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 **True Labels** o Predicted Labels

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

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

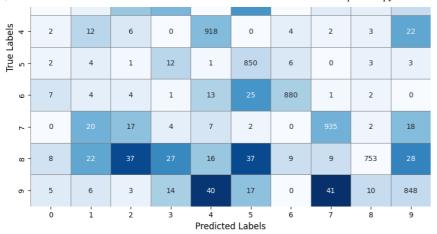
accuracy_clean:1.0

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

Accuracy on adversarial examples: 92.86%

0	4	0	1	5	3	(

٦ -	0	1126	1	1	0	1	2	1	1	0
7 -	7			13	4	1	1	13	4	2
m -	0	3	16	926	0		0	5	4	10



<ipython-input-65-1f2968efe6e5>:72: 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

				zoniusi	on Mati	IIX IOI 6	eps=0	L		
0 -	847	1	26	2	7	36	24	6	11	13
н -	- 0	1097	13	4	2	2	9	3	3	0
7 -	17	137		88	13	1	4	52	24	5
m -	1	10	62		0	129	1	19	32	35
True Labels	7	51	20	0		0	11	26	7	109
True L	7	6	1	78	2		21	1	23	18
9 -	- 29	17	20	2	58	158	642	2	8	1
7	- 2	56	81	17	23	6	1		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-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is

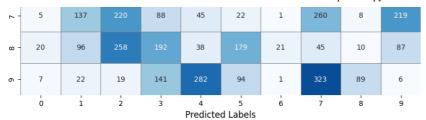
results_df = results_df.append({ Adversarial test data: eps:0.2

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%

			(Confusi	on Mati	rix for e	eps=0.2	2		
0 -	- 250	5	239	4	19	164	171	35	25	61
т-	- 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		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



<ipython-input-65-1f2968efe6e5>:72: 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 Co - 85											
0	- 85	7		5	17	287	161	50	22	48		
1	- 2	120	418	362	36	13	32	48	101	1		
2	- 28	400			30	7	7	123	83	4		
м	- 0	35	182		0	476	0	37	79	77		
abels 4	- 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		
00	- 15	115		196	39	203	12	47	6	54		
6	- 6	29	44	148		98	1	333	70	0		
	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:0.4 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%

Accur	acy on a	adversa								
			(Confusi	on Mat	rix for e	eps=0.4	4		
0	- 12	7		8	17	390	137	58	19	28
. 1	- 1	5	396	608	11	14	12	2	84	0
8	- 26	422	44	263	27	19	7	120	85	3
ω.	- 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
o ·	- 5	63	100	159	212	105	0		46	0
	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:0.5
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												
0 -	- 1	6	295	15	12	471	94	53	16	10			
1	- 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			
۲.	1	150	277	225	36	284	0	13	13	6			
ω -	- 5	159	274	177	24	262	0	40	2	3			
o -	- 2	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 Incorrectly classified: 9673

accuracy_clean:1.0

5

6

0.10

0.20

0.30

9834

9834 9834 9834

9834

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

Acc		acy on a				1.64%					
				(Confusi	on Mat	rix for e	eps=0.6	5		
	0 -	0	3	303	24	8	515	58	45	15	2
	п-	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		6	131	52	15
True Labels	۲۰ -	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
	œ -	- 2	152	288	162	15	290	0	35	1	1
	ი -	1	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	0 0 0	eps tota .01 .02 .03 .04 .05	98 98 98	ect tota 334 334 334 334 334	9834 9834 9834 9834 9834 9834	\					

```
9834
8
       0.40
                                  9834
       0.50
                                  9834
                                                      9834
10
     0.60
                                  9834
                                                      9834
                                                                   correct_adv_counts
       {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
{0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
0
       {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: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
{0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
{0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
6
       {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
{0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
{0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
{0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
<ipython-input-65-1f2968efe6e5>:72: FutureWarning: The frame.append method is
    results_df = results_df.append({
```

→ Attacks run 5 times on each eps Without Label

```
# 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, 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='grey')
        # 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 ģ

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%

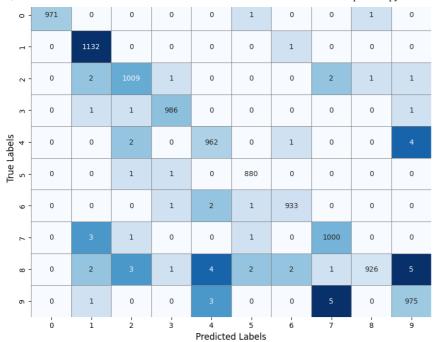
Confusion Matrix for eps=0.01 m **True Labels** ò ġ **Predicted Labels**

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

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%



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 **True Labels** Predicted Labels

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

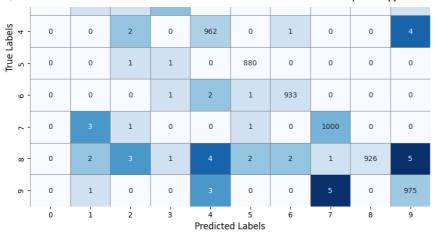
results_df = results_df.append({
Adversarial test data: eps:0.01
Correctly classified: 9774
Incorrectly classified: 60
accuracy_clean:1.0

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

Accuracy on adversarial examples: 99.39%

-	971	0	0	0	0	1	0	0	
	0	1132	0	0	0	0	1	0	

Confusion Matrix for eps=0.01

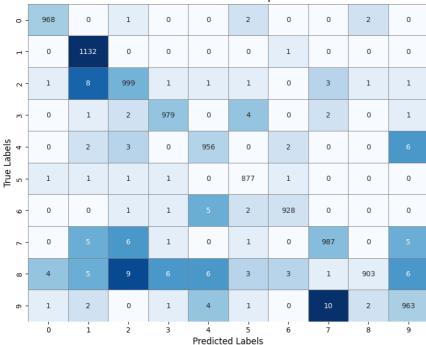


results_df = results_df.append({ Adversarial test data: eps:0.02 Correctly classified: 9692 Incorrectly classified: 142 accuracy_clean:1.0

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:0.02 Correctly classified: 9692 Incorrectly classified: 142

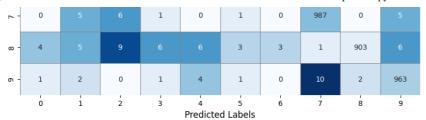
accuracy_clean:1.0

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

Accuracy on adversarial examples: 98.56%

Confusion	Matrix	for	ens=	O	02
Commusion	אווווווא	101	cp3-	v.	U Z

			С	onfusio	n Matr	ix for e	ps=0.0	2		
0 -	968	0	1	0	0	2	0	0	2	0
н -	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	5	2	928	0	0	0



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 **True Labels** Ö

Predicted Labels <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 example

Accur	acy on a	adversai	rial exa	amples:	98.56%					
			C	onfusio	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		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
φ-	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
o -	1	2	0	1	4	1	0	10	2	963
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

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%

			С	onfusio	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		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					3	3	1	903	6
ი -		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:0.03 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		
г.	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		
o -		4	1	4	6	5	0	21	4	935		
	ò	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

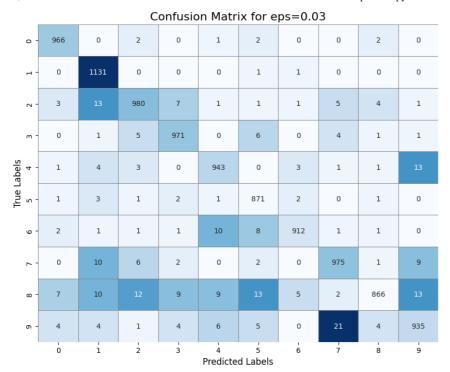
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

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%

 $https://colab.research.google.com/drive/1dSwdmWTF6bA0sDXtVCmY6SdE8JXXbqt8?usp=chrome_ntp\#scrollTo=d9bWa3Clm9Oy\&printMode=true$



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 **Predicted Labels**

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

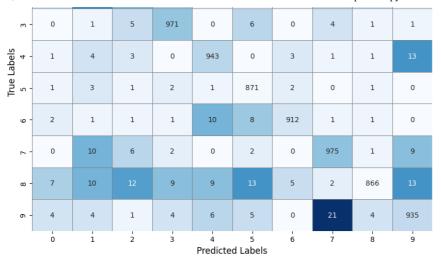
results_df = results_df.append({ Adversarial test data: eps:0.03 Correctly classified: 9550 Incorrectly classified: 284
accuracy_clean:1.0

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

Accuracy on adversarial examples: 97.11%

(Confusion	Matrix	tor	eps=	0.0	J

	Confidence Matrix for eps—0.05												
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			



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 Predicted Labels**

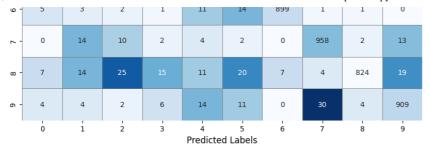
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

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%

4ccur	ccuracy on adversarial examples: 95.53%												
	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			
2	- 5		966	9	3	1	1	9	4	2			
m ·	0	2	10	952	0	12	0	5	3	5			
Labels 4	1	6	5	0	932	0	4	1	2	18			
True L	- 2	3	1	7	1	863	3	0	1	1			
									_				



<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 m **True Labels**

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

à

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%

ò

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

400	Confusion Matrix for eps=0.04												
				С	onfusio	n Matr	ix for e	ps=0.0	4				
(o -	962	0	3	0	1	2	1	0	3	1		
,		0	1129	0	0	0	1	1	1	1	0		
,	- ۲	5	16	966	9	3	1	1	9	4	2		
	m -	0	2	10	952	0	12	0	5	3	5		
True Labels	4 -	1	6	5	0	932	0	4	1	2	18		
True	- 2	2	3	1	7	1	863	3	0	1	1		
	- و	5	3	2	1	11	14	899	1	1	0		
1		0	14	10	2	4	2	0	958	2	13		
,		7	14	25		11		7	4	824	19		
(ი -	4	4	2	6	14	11	0	30	4	909		



<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 **True Labels**

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

results_df = results_df.append({
Adversarial test data: eps:0.04
Correctly classified: 9394
Incorrectly classified: 440
accuracy_clean:1.0
Accuracy on clean data: 100.00%
Adversarial test data: eps:0.04

ò

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04 **True Labels** ω **Predicted Labels**

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

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 -		0	4	0	1	5	3	0	3	2		
	٦ -	0	1126	1	1	0	1	2	1	1	0		
c	7 -	7	30		13	4	1	1	13	4	2		
	m -	0	3	16	926	0		0	5	4	10		
True Labels	4 -	2	12	6	0	918	0	4	2	3	22		
True L	ω -	2	4	1	12	1	850	6	0	3	3		
	φ-	7	4	4	1	13		880	1	2	0		
	7 -	0		17	4	7	2	0		2	18		
	ω -	8		37	27	16	37	9	9	753	28		
	ი-	5	6	3	14	40	17	0	41	10	848		
		0	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

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

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

	contasion matrix for cps—0.05										
0 -		0	4	0	1	5	3	0	3	2	
r-	0	1126	1	1	0	1	2	1	1	0	
- 2	7	30		13	4	1	1	13	4	2	
m -	0	3	16	926	0		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		880	1	2	0	
۲ -	0		17	4	7	2	0		2	18	
- ∞	8		37	27	16	37	9	9	753	28	
ი -	5	6	3	14	40	17	0	41	10	848	
	Ö	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9	

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

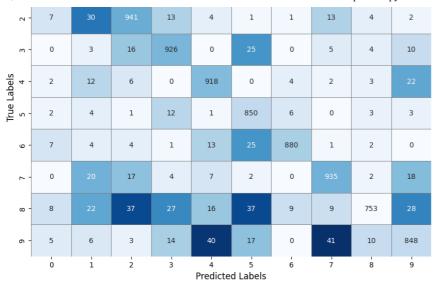
results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702 accuracy_clean:1.0 Accuracy on clean data: 100.00%

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	
- 1	0	1126	1	1	0	1	2	1	1	0	



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 ò **Predicted Labels**

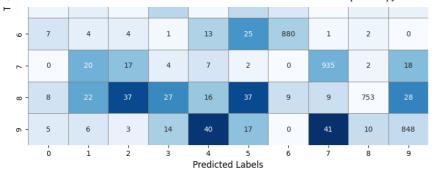
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

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%

	icea, acy on dave, our fact examples of services													
	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				
7 -	7			13	4	1	1	13	4	2				
m -	0	3	16	926	0		0	5	4	10				
Labels 4	- 2	12	6	0	918	0	4	2	3	22				
rue L	- 2	4	1	12	1	850	6	0	3	3				



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

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

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		88	13	1	4	52	24	5			
m -	- 1	10	62		0	129	1	19	32	35			
True Labels	7	51	20	0		0	11	26	7	109			
True L	7	6	1	78	2		21	1	23	18			
φ-	- 29	17	20	2	58	158	642	2	8	1			
7	- 2	56	81	17	23	6	1		4	78			
ω -	17	65	175	151	35	121	20	32	252	78			



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

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	
п-	0	1097	13	4	2	2	9	3	3	0	
7 -	17	137		88	13	1	4	52	24	5	
m -	1	10	62		0	129	1	19	32	35	
True Labels	7	51	20	0		0	11	26	7	109	
True L	7	6	1	78	2		21	1	23	18	
9 -	29	17	20	2	58	158	642	2	8	1	
7	- 2	56	81	17	23	6	1		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-56-9fea085cbc61>:66: FutureWarning: The frame.append method is or applied of a property of a p

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	
r-	0	1097	13	4	2	2	9	3	3	0	
- 2	17	137		88	13	1	4	52	24	5	
m -	1	10	62		0	129	1	19	32	35	
True Labels	7	51	20	0		0	11	26	7	109	
True L	7	6	1	78	2		21	1	23	18	
- و	29	17	20	2	58	158	642	2	8	1	
۲ -	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-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											
0 -	847	1	26	2	7	36	24	6	11	13	
г-	0	1097	13	4	2	2	9	3	3	0	
7 -	- 17	137		88	13	1	4	52	24	5	
m -	- 1	10	62		0	129	1	19	32	35	
True Labels	7	51	20	0		0	11	26	7	109	
True L	7	6	1	78	2		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	
	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%

Confusion Matrix for eps=0.2

Confusion Matrix for eps=0.2												
0 -	250	5	239	4	19	164	171	35	25	61		
н -	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		34	19	128	38	289		
True L	12	15	4	278	5		47	1	121	72		
φ-	66	33	63	4	240	361	145	6	17	2		
۲ -	5	137	220	88	45	22	1	260	8	219		
œ -	20	96	258		38		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

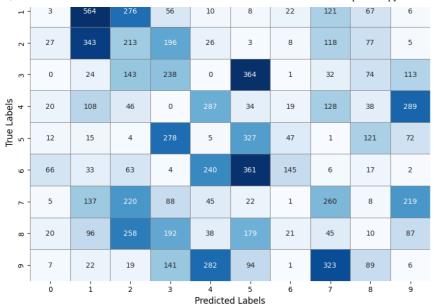
results_df = results_df.append({
Adversarial test data: eps:0.2
Correctly classified: 2300
Incorrectly classified: 7534

accuracy_clean:1.0

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

Accuracy on adversarial examples: 23.39%

	Comasion Matrix for Cps 6.2											
0 -	250	5	239	4	19	164	171	35	25	61		



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		
н -	- 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		34	19	128	38	289		
True L	12	15	4	278	5		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		38		21	45	10	87		
თ -	7	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 ι

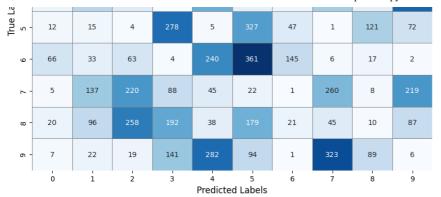
results_df = results_df.append({ Adversarial test data: eps:0.2 Correctly classified: 2300 Incorrectly classified: 7534 accuracy_clean:1.0

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

Accuracy on adversarial examples: 23.39%

Confusion	Matrix	for	ens=0.2
Commusion	אומנווא	101	CP3-0.2

	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			
bels 4	- 20	108	46	0		34	19	128	38	289			



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

	Contasion Matrix for eps—6.2											
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		34	19	128	38	289		
True L	- 12	15	4	278	5		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		38		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-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

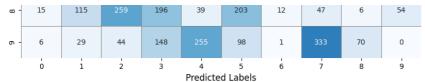
results_df = results_df.append({
Adversarial test data: eps:0.3
Correctly classified: 793 Incorrectly classified: 9041

accuracy_clean:1.0

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

Accuracy on adversarial examples: 8.06%

	Confusion Matrix for eps=0.3												
0 -	85	7		5	17	287	161	50	22	48			
т-	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			
7	- 5	149		148	47	120	1	66	9	194			



<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%

	Confusion Matrix for eps=0.3												
0 -	85	7		5	17	287	161	50	22	48			
1	- 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		196	39	203	12	47	6	54			
ი -		29	44	148		98	1	333	70	0			
	Ó	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9			

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

Confusion Matrix for eps=0.3

17

161

50

22

0

70

8

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

accuracy_clean:1.0

85

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

29

1	- 2	120	418	362	36	13	32	48	101	1
2	- 28	400			30	7	7	123	83	4
м	- 0	35	182		0	476	0	37	79	77
abels 4	- 16	138	80	7		147	19	192	55	218
True Labels	- 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
80	- 15	115		196	39	203	12	47	6	54

 $\verb|-ipython-input-56-9fea085cbc61|| : 66: Future \textit{Warning: The frame.append method is of the following of the control of the following of the control of t$ results_df = results_df.append({
Advarcarial test data: enc:0 3

Predicted Labels

4

5

148

3

1

6

ישיכו שמו במו בכשנ ממנמ. 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**

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

Ó

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 True Labels **Predicted Labels**

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

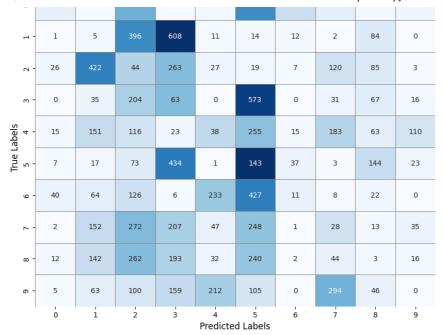
results_df = results_df.append({ Adversarial test data: eps:0.4 Correctly classified: 347

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%

(Lonfusi	on Mati	rix for e	eps=0.4	4		
297	8	17	390	137	58	19	28



<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												
0 -	- 12	7		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			
7	- 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			
	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												
0 -	12	7		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			



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

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												
0 -	- 12	7		8	17	390	137	58	19	28			
н -	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			
ი -		63	100	159	212	105	0		46	0			
	ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

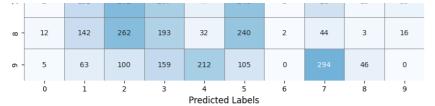
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is \circ

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													
Confusion Matrix for eps=0.4													
0 -	12	7		8	17	390	137	58	19	28			
г -	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			



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

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				
1	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				
φ-	- 27	86	157	8	158	463	3	11	24	0				
۲-	1	150	277	225	36	284	0	13	13	6				
ω -	- 5	159	274	177	24	262	0	40	2	3				
o -		118	176	173	122	114	0	248	31	0				
	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:0.5 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				
т -	- 1	1		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				
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				
ω -	- 5	159	274	177	24	262	0	40	2	3				
თ -	- 2	118	176	173	122	114	0	248	31	0				
	Ó	i	2	3	4 Predicte	s d Labels	6	7	8	9				

distribution for Officershield .co. FutureMannian. The force control mathed in

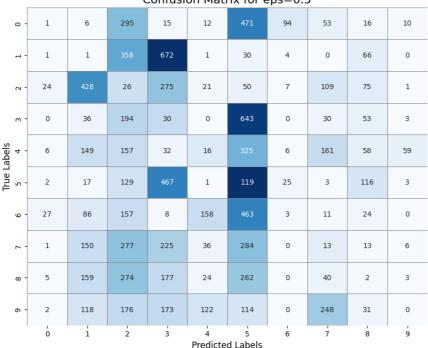
sipytnon-input-co-yreaשמטכטכסי: ruturewarning: The trame.append method is יו

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:0.5 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			
п-	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			
φ-	27	86	157	8	158	463	3	11	24	0			
7	1	150	277	225	36	284	0	13	13	6			
ω -	- 5	159	274	177	24	262	0	40	2	3			
ი -	2	118	176	173	122	114	0	248	31	0			
	0	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

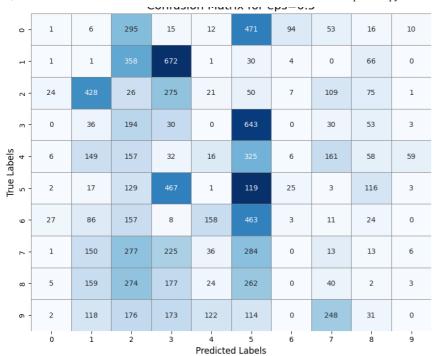
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is α

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%



<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
н -	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
۲.	0	142	283	252	26	287	1	3	9	2
ю -	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	5ٰ d Labels	6	7	8	9

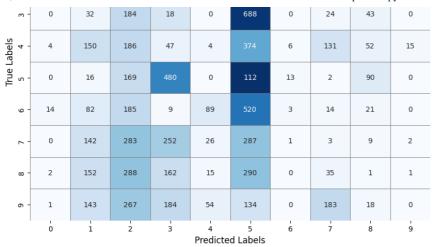
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is

results_df = results_df.append({
Adversarial test data: eps:0.6 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			
п-	0	0	328	703	1	44	2	0	55	0			
- 2	11	431	20	283	12	101	4	93	61	0			



<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

	Confusion Matrix for eps—0.0											
0 -	0	3	303	24	8	515	58	45	15	2		
	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		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		
۲.	0	142	283	252	26	287	1	3	9	2		
ω -	- 2	152	288	162	15	290	0	35	1	1		
o -		143	267	184	54	134	0	183	18	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

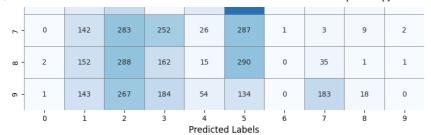
results_df = results_df.append({ Adversarial test data: eps:0.6 Correctly classified: 161 Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6 Accuracy on adversarial examples: 1.64%

Confusion N	Мatrix	for	eps=0.6
-------------	--------	-----	---------

	Confusion Matrix for eps=0.6												
0 -	- 0	3	303	24	8	515	58	45	15	2			
г.	- 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			
abels	4	150	186	47	4		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			



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

Accur	Confusion Matrix for eps=0.6									
			(ontusi	on Mati	rix tor e	eps=0.)		
0 -	- 0	3	303	24	8	515	58	45	15	2
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		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
o -		143	267	184	54	134	0	183	18	0
	ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

	Predicted Lab						
	eps	attack_num	total_correct	total_adv			
ð	0.01	_ 1	9834	9834			
1	0.01	2	9834	9834			
2	0.01	3	9834	9834			
3	0.01	4	9834	9834			
4	0.01	5	9834	9834			
5	0.02	1	9834	9834			
5	0.02	2	9834	9834			
7	0.02	3	9834	9834			
3	0.02	4	9834	9834			
9	0.02	5	9834	9834			
10	0.03	1	9834	9834			
11	0.03	2	9834	9834			
12	0.03	3	9834	9834			
13	0.03	4	9834	9834			
14	0.03	5	9834	9834			
15	0.04	1	9834	9834			
16	0.04	2	9834	9834			
17	0.04	3	9834	9834			
18	0.04	4	9834	9834			
19	0.04	5	9834	9834			
20	0.05	1	9834	9834			
21	0.05	2	9834	9834			
22	0.05	3	9834	9834			
23	0.05	4	9834	9834			
24	0.05	5	9834	9834			
25	0.10	1	9834	9834			
26	0.10	2	9834	9834			
27	0.10	3	9834	9834			
28	0.10	4	9834	9834			
29	0.10	5	9834	9834			
30	0.20	1	9834	9834			
31	0.20	2	9834	9834			
32	0.20	3	9834	9834			
33	0.20	4	9834	9834			
34	0.20	5	9834	9834			
35	0.30	1	9834	9834			
36	0.30	2	9834	9834			
37	0.30	3	9834	9834			

```
9834
                                            9834
38
   0.30
                   4
                   5
                                9834
                                            9834
39
   0.30
                                9834
                                            9834
40
   0.40
                   1
41
   0.40
                   2
                                9834
                                            9834
42
   0.40
                   3
                                9834
                                            9834
43
   0.40
                   4
                                9834
                                            9834
44
   0.40
                                9834
                                            9834
45
                                9834
                                            9834
   0.50
                   1
46
   0.50
                                9834
                                            9834
47
   0.50
                   3
                                9834
                                            9834
48
   0.50
                   4
                                9834
                                            9834
19
   0.50
                   5
                                9834
                                            9834
50
                                9834
   0.60
                   1
                                            9834
   0.60
                                9834
51
                                            9834
52
   0.60
                   3
                                9834
                                            9834
53
   0.60
                   4
                                9834
                                            9834
54
   0.60
                                9834
                                            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: ...
3
    {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: ...
3
    {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
    {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
10
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
11
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
12
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
13
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
14
    {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
15
16
    {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977,
17
    {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
18
    {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
19
    {0:
       986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
    {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
20
21
    {0:
       986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
    {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
22
    {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
23
24
    {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
    {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
25
    {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
    {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
29
    {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
30
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
31
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
32
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
33
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
{0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
34
35
    {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
36
37
    {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
38
    {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
39
    {0:
       226, 1: 1063, 2:
                         1750, 3: 1589, 4: 790, 5:...
40
    {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
41
    {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
    {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
42
43
    {0:
       120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
       120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
    {0:
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
45
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
46
47
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
48
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
49
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
50
    {0:
       32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
51
    {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
    {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
53
    {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
       32, 1: 1151, 2: 2213, 3: 2162, 4: 209,
<ipython-input-56-9fea085cbc61>:66: FutureWarning: The frame.append method is 
  results_df = results_df.append({
```

 $https://colab.research.google.com/drive/1dSwdmWTF6bA0sDXtVCmY6SdE8JXXbqt8?usp=chrome_ntp\#scrollTo=d9bWa3Clm9Oy\&printMode=true$

Attacks run 5 times on each eps With Label

```
# 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,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, 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='grey')
        # 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 ģ

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%

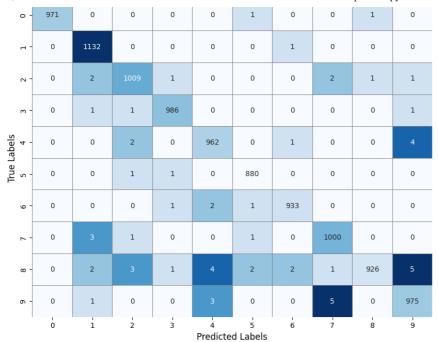
Confusion Matrix for eps=0.01 m **True Labels** ò ġ **Predicted Labels**

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

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



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 **True Labels** ω Predicted Labels

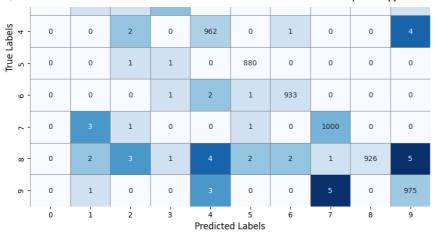
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is ι

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		
п-	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		

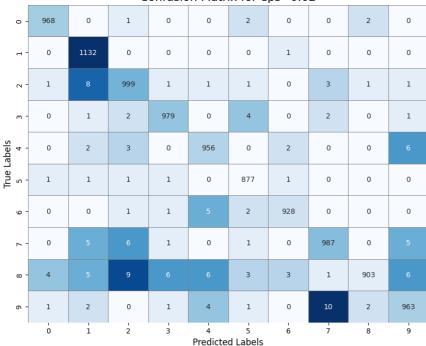


results_df = results_df.append({ Adversarial test data: eps:0.02 Correctly classified: 9692 Incorrectly classified: 142 accuracy_clean:1.0

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

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02



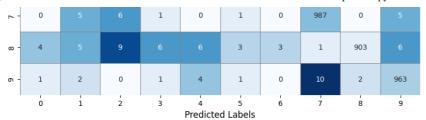
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ Adversarial test data: eps:0.02 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			
۷ .	- 1	8	999	1	1	1	0	3	1	1			
ω.	- 0	1	2	979	0	4	0	2	0	1			
True Labels	- 0	2	3	0	956	0	2	0	0	6			
True I	- 1	1	1	1	0	877	1	0	0	0			
φ.	- 0	0	1	1	5	2	928	0	0	0			



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%

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

Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02 **True Labels** Ö **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%

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

Accuracy on adversarial examples: 98.56%

Accuracy on adversarial examples: 98.56%											
	Confusion Matrix for eps=0.02										
	0 -	968	0	1	0	0	2	0	0	2	0
	- ب	0	1132	0	0	0	0	1	0	0	0
	- 2	1		999	1	1	1	0	3	1	1
	m -	0	1	2	979	0	4	0	2	0	1
True Labels	4 -	0	2	3	0	956	0	2	0	0	6
True L	ω -	1	1	1	1	0	877	1	0	0	0
	9 -	0	0	1	1		2	928	0	0	0
	۲-	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	i	2	3	4 Predicte	sٰ d Labels	6	7	8	9

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 True Labels

 $\label{limits} Predicted\ Labels $$ < ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is $(1.5) \times (1.5) \times$

Confusion Matrix for eps=0.03

ģ

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%

True Labels

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

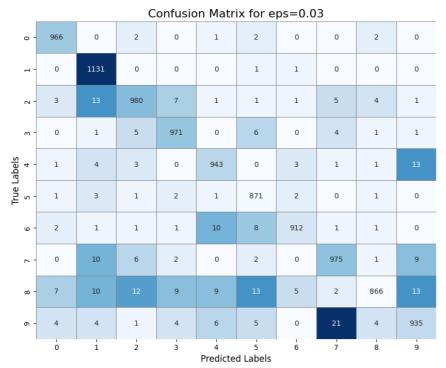
results_df = results_df.append({
Adversarial test data: eps:0.03
Correctly classified: 9550
Incorrectly classified: 284

i

accuracy_clean:1.0

ò

Accuracy on clean data: 100.00% Adversarial test data: eps:0.03 Accuracy on adversarial examples: 97.11%



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 Predicted Labels

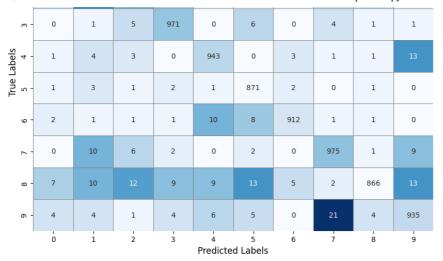
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is α

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		
٦-	0	1131	0	0	0	1	1	0	0	0		
7 -	3	13	980	7	1	1	1	5	4	1		



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	ni Mati	ix ioi e	ps-0.0	5		
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
7	0	10	6	2	0	2	0	975	1	9
ω ·	7	10	12	9	9	13	5	2	866	13
σ.		4	1	4	6	5	0	21	4	935
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

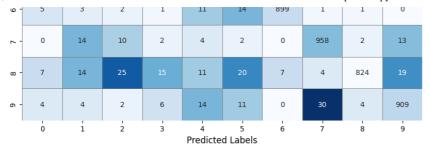
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is

results_df = results_df.append({ 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
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
Labels 4	1	6	5	0	932	0	4	1	2	18
True L	2	3	1	7	1	863	3	0	1	1



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 m **True Labels** ò à Predicted Labels

<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

Accur	Accuracy on adversarial examples: 95.53%									
			C	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
ю	- 0	2	10	952	0	12	0	5	3	5
True Labels 5 4	- 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
6	- 4	4	2	6	14	11	0	30	4	909



<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is \circ 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 **True Labels**

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

results_df = results_df.append({
Adversarial test data: eps:0.04 Correctly classified: 9394 Incorrectly classified: 440 accuracy_clean:1.0 Accuracy on clean data: 100.00%

ò

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
1	- 0	1129	0	0	0	1	1	1	1	0
7	- 5		966	9	3	1	1	9	4	2
м	- 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
б	- 4	4	2	6	14	11	0	30	4	909
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9

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 -		0	4	0	1	5	3	0	3	2
	٦ -	0	1126	1	1	0	1	2	1	1	0
	7 -	7	30		13	4	1	1	13	4	2
	m -	0	3	16	926	0		0	5	4	10
True Labels	4 -	2	12	6	0	918	0	4	2	3	22
True L	ω -	2	4	1	12	1	850	6	0	3	3
	φ-	7	4	4	1	13		880	1	2	0
	7 -	0		17	4	7	2	0		2	18
	ω -	8		37	27	16	37	9	9	753	28
	ი-	5	6	3	14	40	17	0	41	10	848
		0	i	2	3	4 Predicte	5 d Labels	6	7	8	9

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

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

	Contactor Matrix for cps—0.05									
0 -	955	0	4	0	1	5	3	0	3	2
п-	0	1126	1	1	0	1	2	1	1	0
7 -	- 7			13	4	1	1	13	4	2
m -	0	3	16	926	0		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		880	1	2	0
7	0		17	4	7	2	0		2	18
ω -	- 8		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 \circ

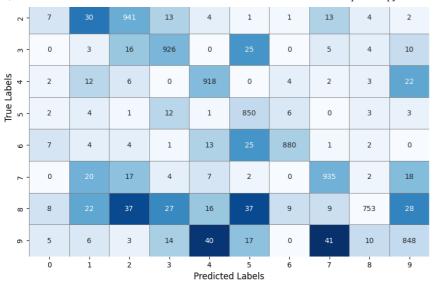
results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702 accuracy_clean:1.0 Accuracy on clean data: 100.00%

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



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 ò **Predicted Labels**

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

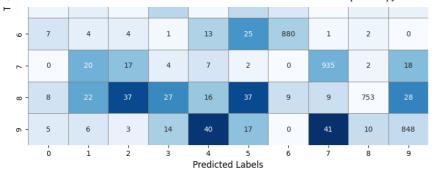
results_df = results_df.append({ Adversarial test data: eps:0.05 Correctly classified: 9132 Incorrectly classified: 702 accuracy_clean:1.0 Accuracy on clean data: 100.00%

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86% Confusion Matrix for eps=0.05

0 -		0	4	0	1	5	3	0
п-	0	1126	1	1	0	1	2	1
7 -	7			13	4	1	1	13

т-	0	1126	1	1	0	1	2	1	1	0
2 -	7			13	4	1	1	13	4	2
m -	0	3	16	926	0		0	5	4	10
Labels 4	- 2	12	6	0	918	0	4	2	3	22
rue L	- 2	4	1	12	1	850	6	0	3	3



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

	Comusion Matrix for eps—0.1									
0 -	847	1	26	2	7	36	24	6	11	13
н-	0	1097	13	4	2	2	9	3	3	0
7 -	17	137		88	13	1	4	52	24	5
m -	1	10	62		0	129	1	19	32	35
True Labels	7	51	20	0		0	11	26	7	109
True L	7	6	1	78	2		21	1	23	18
9 -	29	17	20	2	58	158	642	2	8	1
7	2	56	81	17	23	6	1		4	78
ω -	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-66-119194b068fb>:66: FutureWarning: The frame.append method is \circ

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	ens=0.1
Commusion	אומנווא	101	Ch2-0.1

			(Lonfusi	on Mati	rix for e	eps=0.1	L		
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		88	13	1	4	52	24	5
m -	- 1	10	62		0	129	1	19	32	35
True Labels	7	51	20	0		0	11	26	7	109
True L	7	6	1	78	2		21	1	23	18
φ-	- 29	17	20	2	58	158	642	2	8	1
7	- 2	56	81	17	23	6	1		4	78
ω -	17	65	175	151	35	121	20	32	252	78



<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:0.1 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		
п-	0	1097	13	4	2	2	9	3	3	0		
7 -	17	137		88	13	1	4	52	24	5		
m -	1	10	62		0	129	1	19	32	35		
True Labels	7	51	20	0		0	11	26	7	109		
True L	7	6	1	78	2		21	1	23	18		
9 -	29	17	20	2	58	158	642	2	8	1		
7	- 2	56	81	17	23	6	1		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 or analytic defendance of the property of t

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

	Community for Cps of 1									
0 -	847	1	26	2	7	36	24	6	11	13
- 13	0	1097	13	4	2	2	9	3	3	0
7 -	17	137		88	13	1	4	52	24	5
m -	1	10	62		0	129	1	19	32	35
abels	7	51	20	0		0	11	26	7	109
True Labels	7	6	1	78	2		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
6 -		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
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		
г-	0	1097	13	4	2	2	9	3	3	0		
2 -	17	137		88	13	1	4	52	24	5		
m -	1	10	62		0	129	1	19	32	35		
True Labels	7	51	20	0		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-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 ens=0.2

	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		34	19	128	38	289	
True L	12	15	4	278	5		47	1	121	72	
φ-	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		38		21	45	10	87	
ი -	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

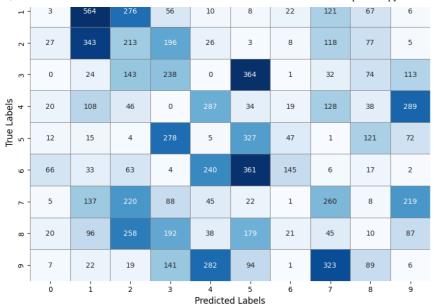
accuracy_clean:1.0

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

Accuracy on adversarial examples: 23.39%

Confusion Matrix for eps=0.2

	CONTROL NATION CASE OF CASE											
0 -	250	5	239	4	19	164	171	35	25	61		



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	
H -	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		34	19	128	38	289	
True L	12	15	4	278	5		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		38		21	45	10	87	
თ -	7	22	19	141	282	94	1	323	89	6	
1	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

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

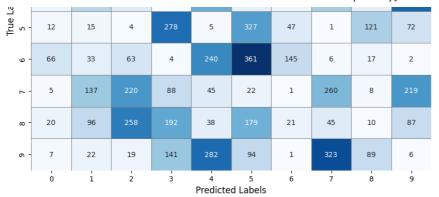
results_df = results_df.append({ Adversarial test data: eps:0.2 Correctly classified: 2300 Incorrectly classified: 7534 accuracy_clean:1.0

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

Accuracy on adversarial examples: 23.39%

Confusion	Matrix	for	ens=0.2
Commusion	אומנווא	101	CP3-0.2

	Confidence for the confidence of the confidence									
0 -	250	5	239	4	19	164	171	35	25	61
- 13	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
bels	20	108	46	0		34	19	128	38	289



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		34	19	128	38	289
True L	- 12	15	4	278	5		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		38		21	45	10	87
თ -	7	22	19	141	282	94	1	323	89	6
	Ö	i	2	3	4 Predicte	5 d Labels	6	7	8	9

<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is or analytic defendant of the property of th

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

Incorrectly classified: 9041

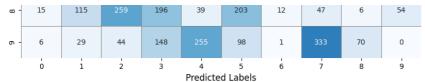
accuracy_clean:1.0

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

Accuracy on adversarial examples: 8.06%

Confusion	Matrix	for	eps=	0.3
-----------	--------	-----	------	-----

			(Confusi	on Mati	rix for e	eps=0	3		
0 -	85	7		5	17	287	161	50	22	48
- 1	2	120	418	362	36	13	32	48	101	1
7 -	- 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
7	- 5	149		148	47	120	1	66	9	194



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											
0 -	85	7		5	17	287	161	50	22	48	
г-	- 2	120	418	362	36	13	32	48	101	1	
7 -	- 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		196	39	203	12	47	6	54	
თ -	- 6	29	44	148		98	1	333	70	0	
	o	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

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

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%

0 -	85	7		5	17	287	161	50	22	48
н -	2	120	418	362	36	13	32	48	101	1
7 -	- 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
۲ -	- 5	149		148	47	120	1	66	9	194
ω -	15	115		196	39	203	12	47	6	54
	_						_			_

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

Predicted Labels

ישיכו שמו במו בכשנ ממנמ. 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

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

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%

Ó

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

Confusion Matrix for eps=0.3 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

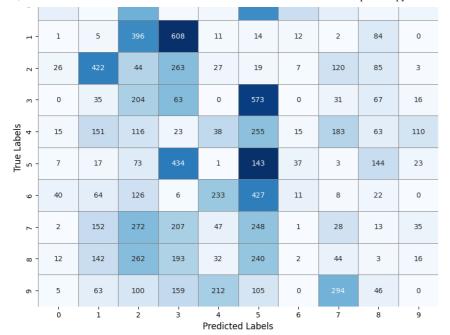
Incorrectly classified: 9487

accuracy_clean:1.0

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

Confusion Matrix for eps=0.4

	Comusion Matrix for Cps—0.4											
0 -	12	7	297	8	17	390	137	58	19	28		



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										
0 -	- 12	7		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
7	- 2	152	272	207	47	248	1	28	13	35
ω -	- 12	142	262	193	32	240	2	44	3	16
ი -	- 5	63	100	159	212	105	0		46	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 ϵ

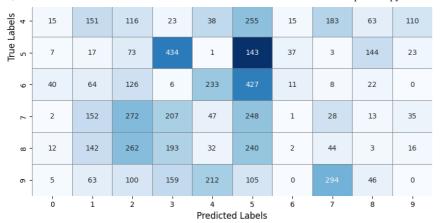
results_df = results_df.append({
Adversarial test data: eps:0.4
Correctly classified: 347
Incorrectly classified: 9487

accuracy_clean:1.0

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

Accuracy on adversarial examples: 3.53%

	Collidsion 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	
2 -	26	422	44	263	27	19	7	120	85	3	
m -	0	35	204	63	0	573	0	31	67	16	



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

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%

-ccui	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		
2	- 26	422	44	263	27	19	7	120	85	3		
м	- 0	35	204	63	0	573	0	31	67	16		
True Labels 5 4	- 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		
00	- 12	142	262	193	32	240	2	44	3	16		
6		63	100	159	212	105	0		46	0		
	o	i	2	3	4 Predicte	5 d Labels	6	7	8	9		

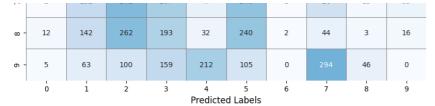
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is ι

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											
0 -	- 12	7		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	
,	. 2	152	272	207	47	248	1	28	13	35	



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

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	
1	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	
φ-	- 27	86	157	8	158	463	3	11	24	0	
۲-	1	150	277	225	36	284	0	13	13	6	
ω -	- 5	159	274	177	24	262	0	40	2	3	
o -		118	176	173	122	114	0	248	31	0	
	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:0.5 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	
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	
۲ -	1	150	277	225	36	284	0	13	13	6	
ω -	5	159	274	177	24	262	0	40	2	3	
ი -	2	118	176	173	122	114	0	248	31	0	
,	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

sipytnon-input-oo-iigisadooti>:oo: ruturewarning: ine trame.append method is י

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**

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

Predicted Labels

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

i

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		
- 1	1	1		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		
Frue 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		
ω -	- 5	159	274	177	24	262	0	40	2	3		
6 -	- 2	118	176	173	122	114	0	248	31	0		
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

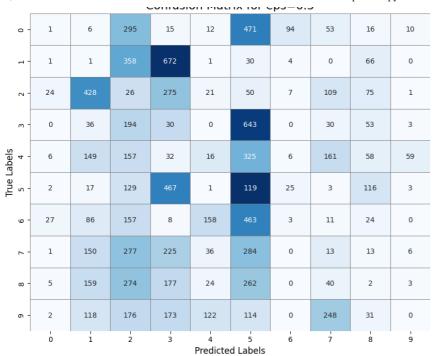
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is α

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 ens=0.5



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

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		
н -	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		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		
თ -	1	143	267	184	54	134	0	183	18	0		
	0	i	2	3	4 Predicte	5ٰ d Labels	6	7	8	9		

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

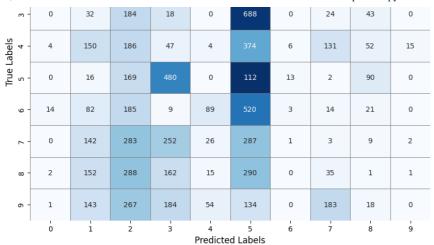
results_df = results_df.append({
Adversarial test data: eps:0.6 Correctly classified: 161 Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6 Accuracy on adversarial examples: 1.64%

Confusi	on	Matri	x fo	or e	ps=	0.6	5

	Confusion Matrix for eps=0.6												
0 -	0	3	303	24	8	515	58	45	15	2			
н-	0	0	328	703	1	44	2	0	55	0			
- 2	- 11	431	20	283	12	101	4	93	61	0			



<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 ens=0.6

	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			
м	- 0	32	184	18	0	688	0	24	43	0			
True Labels 5 4	- 4	150	186	47	4		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			
œ	- 2	152	288	162	15	290	0	35	1	1			
6		143	267	184	54	134	0	183	18	0			
	Ö	i	2	3	4 Predicte	5 d Labels	6	7	8	9			

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

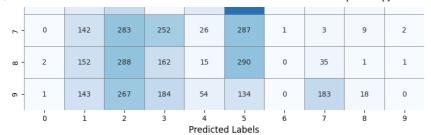
results_df = results_df.append({ Adversarial test data: eps:0.6 Correctly classified: 161 Incorrectly classified: 9673

accuracy_clean:1.0

Accuracy on clean data: 100.00% Adversarial test data: eps:0.6 Accuracy on adversarial examples: 1.64%

Confusion I	Matrix	for	eps=0.6
-------------	--------	-----	---------

	Confusion Matrix for eps=0.6											
0 -	0	3	303	24	8	515	58	45	15	2		
п-	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		
abels	4	150	186	47	4		6	131	52	15		
True Labels	0	16	169	480	0	112	13	2	90	0		
φ-	14	82	185	9	89	520	3	14	21	0		



<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is or 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	
г.	- 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		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	
σ.	- 1	143	267	184	54	134	0	183	18	0	
	Ó	i	2	3	4 Predicte	5 d Labels	6	7	8	9	

			Pr	Predicted Labels	
	eps	attack_num	total_correct	total adv	
ð	0.01	_ 1	9834	9834	
1	0.01	2	9834	9834	
2	0.01	3	9834	9834	
3	0.01	4	9834	9834	
4	0.01	5	9834	9834	
5	0.02	1	9834	9834	
õ	0.02	2	9834	9834	
7	0.02	3	9834	9834	
3	0.02	4	9834	9834	
9	0.02	5	9834	9834	
10	0.03	1	9834	9834	
11	0.03		9834	9834	
12	0.03	2	9834	9834	
13	0.03	4	9834	9834	
14	0.03	5	9834	9834	
15	0.04	1	9834	9834	
16	0.04	2	9834	9834	
17	0.04	3	9834	9834	
18	0.04	4	9834	9834	
19	0.04	5	9834	9834	
20	0.05	1	9834	9834	
21	0.05	2	9834	9834	
22	0.05		9834	9834	
23	0.05	4	9834	9834	
24	0.05	5	9834	9834	
25	0.10	1	9834	9834	
26	0.10	2	9834	9834	
27	0.10	3	9834	9834	
28	0.10	4	9834	9834	
29	0.10	5	9834	9834	
30	0.20	1	9834	9834	
31	0.20	2	9834	9834	
32	0.20	3	9834	9834	
33	0.20	4	9834	9834	
34	0.20	5	9834	9834	
35	0.30	1	9834	9834	
36	0.30	2	9834	9834	
37	0.30	3	9834	9834	

```
9834
                                            9834
38
    0.30
                    4
                   5
                                 9834
                                            9834
39
    0.30
                                 9834
                                            9834
40
   0.40
                   1
41
    0.40
                    2
                                9834
                                            9834
42
    0.40
                    3
                                9834
                                            9834
43
    0.40
                    4
                                 9834
                                            9834
44
    0.40
                                 9834
                                            9834
45
                                 9834
                                            9834
    0.50
                    1
46
    0.50
                                9834
                                            9834
47
    0.50
                    3
                                 9834
                                            9834
48
    0.50
                    4
                                9834
                                            9834
19
    0.50
                   5
                                9834
                                            9834
50
                                9834
                                            9834
    0.60
                    1
51
   0.60
                    2
                                9834
                                            9834
52
    0.60
                    3
                                9834
                                            9834
53
    0.60
                    4
                                 9834
                                            9834
54
    0.60
                                9834
                                            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, 5: ...
{0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
    {0: 971, 1: 1141, 2: 1017, 3: 990, 4: 971, 5: ...
3
    {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: ...
3
    {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
    {0: 975, 1: 1156, 2: 1022, 3: 990, 4: 972, 5: ...
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
10
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
11
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
12
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
13
    {0: 984, 1: 1177, 2: 1011, 3: 996, 4: 971, 5: ...
14
    {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
15
    {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977,
16
17
    {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
18
    {0: 986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
19
    {0:
        986, 1: 1191, 2: 1024, 3: 992, 4: 977, 5: ...
    {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
20
21
    {0:
        986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
    {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
22
    {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
23
24
    {0: 986, 1: 1227, 2: 1030, 3: 998, 4: 1000, 5:...
    {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
25
    {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
    {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
29
    {0: 933, 1: 1452, 2: 1081, 3: 1093, 4: 1070, 5...
30
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
31
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
32
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
33
    {0: 410, 1: 1347, 2: 1481, 3: 1197, 4: 952, 5:...
{0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
34
35
    {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
36
37
    {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
38
    {0: 226, 1: 1063, 2: 1750, 3: 1589, 4: 790, 5:...
39
    {0:
        226, 1: 1063, 2:
                          1750, 3: 1589, 4: 790, 5:...
40
    {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
41
    {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
    {0: 120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
42
43
    {0:
        120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
       120, 1: 1058, 2: 1890, 3: 1964, 4: 618, 5:...
    {0:
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
45
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
46
47
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
48
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
49
    {0: 69, 1: 1150, 2: 2043, 3: 2074, 4: 391, 5: ...
50
    {0:
        32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
51
    {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
    {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
53
    {0: 32, 1: 1151, 2: 2213, 3: 2162, 4: 209, 5: ...
                      2: 2213, 3: 2162, 4: 209,
        32, 1: 1151,
<ipython-input-66-119194b068fb>:66: FutureWarning: The frame.append method is 
  results_df = results_df.append({
```