

## ▾ Intel FoolBox Library

```
!pip install foolbox
```

## ▾ Fast Gradient Sign Method (FGSM)

```
import numpy as np
import tensorflow as tf
from keras.models import load_model
import foolbox as fb
import eagerpy as ep
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Load your trained model
model = load_model('/content/drive/MyDrive/ColabNotebooks/mnist_model.h5')

# Load and preprocess the examples
correct_examples = np.load('/content/drive/MyDrive/ColabNotebooks/correct_examples.npy')
correct_labels = np.load('/content/drive/MyDrive/ColabNotebooks/correct_labels.npy').astype('int32') # Cast labels to int32

# Create a Foolbox model
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))

# Calculate accuracy on clean data
clean_predictions = model.predict(correct_examples).argmax(axis=-1)
accuracy_clean = np.mean(clean_predictions == correct_labels)
print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")

# Convert data to TensorFlow tensors
images = tf.convert_to_tensor(correct_examples)
labels = tf.convert_to_tensor(correct_labels, dtype=tf.int32) # Cast labels to int32

# Create a Foolbox model for TensorFlow
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))

# Apply an FGSM attack
attack = fb.attacks.FGSM()
epsilons = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
raw_advs, clipped_advs, success = attack(fmodel, images, labels, epsilons=epsilons)

# Assuming 'clipped_advs' are the adversarial examples for different epsilons
for eps, advs_ in zip(epsilons, clipped_advs):
    # Predict the labels of the adversarial examples
    y_adv = np.argmax(model.predict(advs_), axis=1)

    # Calculate accuracy on adversarial examples
    accuracy_adv = np.mean(y_adv == correct_labels)
    print(f"\nAdversarial 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
    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
    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)

# Save the plot
image_filename = f'confusion_matrix_eps_{eps}.png'
plt.savefig(image_filename, bbox_inches='tight')
plt.show() # Display the figure in the notebook
```



WARNING:tensorflow:From /usr/local/lib/python3.10/dist-packages/foolbox/model: Instructions for updating:  
 Use `tf.config.list\_physical\_devices('GPU')` instead.  
 308/308 [=====] - 8s 2ms/step  
 Accuracy on clean data: 100.00%  
 308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.01  
 Accuracy on adversarial examples: 99.39%

Confusion Matrix for eps=0.01

True Labels	0	1	2	3	4	5	6	7	8	9
0	971	0	0	0	0	1	0	0	1	0
1	0	1132	0	0	0	0	1	0	0	0
2	0	2	1009	1	0	0	0	2	1	1
3	0	1	1	986	0	0	0	0	0	1
4	0	0	2	0	962	0	1	0	0	4
5	0	0	1	1	0	880	0	0	0	0
6	0	0	0	1	2	1	933	0	0	0
7	0	3	1	0	0	1	0	1000	0	0
8	0	2	3	1	4	2	2	1	926	5
9	0	1	0	0	3	0	0	5	0	975
Predicted Labels	0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.02  
 Accuracy on adversarial examples: 98.56%

Confusion Matrix for eps=0.02

True Labels	0	1	2	3	4	5	6	7	8	9
0	968	0	1	0	0	2	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
2	1	8	999	1	1	1	0	3	1	1
3	0	1	2	979	0	4	0	2	0	1
4	0	2	3	0	956	0	2	0	0	6
5	1	1	1	1	0	877	1	0	0	0
6	0	0	1	1	5	2	928	0	0	0
7	0	5	6	1	0	1	0	987	0	5
8	4	5	9	6	6	3	3	1	903	6
9	1	2	0	1	4	1	0	10	2	963
Predicted Labels	0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 3ms/step

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

Confusion Matrix for eps=0.03

True Labels	0	1	2	3	4	5	6	7	8	9
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

True Labels \ Predicted Labels	0	1	2	3	4	5	6	7	8	9
3	0	1	5	971	0	6	0	4	1	1
4	1	4	3	0	943	0	3	1	1	13
5	1	3	1	2	1	871	2	0	1	0
6	2	1	1	1	10	8	912	1	1	0
7	0	10	6	2	0	2	0	975	1	9
8	7	10	12	9	9	13	5	2	866	13
9	4	4	1	4	6	5	0	21	4	935

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.53%

Confusion Matrix for eps=0.04

True Labels \ Predicted Labels	0	1	2	3	4	5	6	7	8	9
0	962	0	3	0	1	2	1	0	3	1
1	0	1129	0	0	0	1	1	1	1	0
2	5	16	966	9	3	1	1	9	4	2
3	0	2	10	952	0	12	0	5	3	5
4	1	6	5	0	932	0	4	1	2	18
5	2	3	1	7	1	863	3	0	1	1
6	5	3	2	1	11	14	899	1	1	0
7	0	14	10	2	4	2	0	958	2	13
8	7	14	25	15	11	20	7	4	824	19
9	4	4	2	6	14	11	0	30	4	909

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.86%

### Confusion Matrix for $\epsilon=0.05$

	0	1	2	3	4	5	6	7	8	9
0	955	0	4	0	1	5	3	0	3	2
1	0	1126	1	1	0	1	2	1	1	0
2	7	30	941	13	4	1	1	13	4	2
3	0	3	16	926	0	25	0	5	4	10
4	2	12	6	0	918	0	4	2	3	22
5	2	4	1	12	1	850	6	0	3	3
6	7	4	4	1	13	25	880	1	2	0
7	0	20	17	4	7	2	0	935	2	18
8	8	22	37	27	16	37	9	9	753	28
9	5	6	3	14	40	17	0	41	10	848

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.1  
Accuracy on adversarial examples: 69.49%

Confusion Matrix for eps=0.1

True Labels	Predicted Labels									
	0	1	2	3	4	5	6	7	8	9
0	846	1	27	2	7	36	24	6	11	13
1	0	1097	13	4	2	2	9	3	3	0
2	17	137	675	88	13	1	4	52	24	5
3	1	10	61	702	0	128	1	19	32	35
4	7	51	20	0	738	0	11	26	7	109
5	7	6	1	78	2	725	21	1	23	18
6	29	17	19	3	58	158	642	2	8	1
7	2	56	81	17	23	6	1	737	4	78
8	17	65	175	151	35	121	20	32	252	78
9	6	13	8	51	191	55	1	196	43	420

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.2  
Accuracy on adversarial examples: 23.39%

Confusion Matrix for eps=0.2

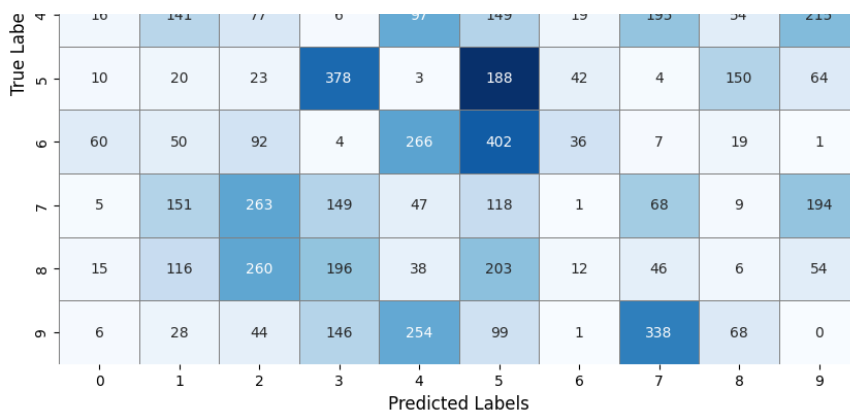
True Labels	Predicted Labels									
	0	1	2	3	4	5	6	7	8	9
0	252	4	233	4	19	168	170	37	25	61
1	3	565	278	56	10	8	22	119	66	6
2	26	343	213	195	26	3	8	119	78	5
3	0	24	144	234	0	366	1	32	73	115
4	21	109	48	0	285	35	20	127	37	287
5	12	15	4	278	5	328	47	1	120	72
6	65	37	62	4	235	358	150	6	18	2
7	5	137	220	89	46	22	1	257	8	220
8	20	95	258	192	38	179	22	45	10	87
9	7	22	18	141	281	95	1	324	89	6

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.3  
Accuracy on adversarial examples: 8.03%

Confusion Matrix for eps=0.3

True Labels	Predicted Labels									
	0	1	2	3	4	5	6	7	8	9
0	85	7	289	6	17	296	155	50	21	47
1	2	113	421	369	33	14	31	46	103	1
2	28	398	95	240	31	8	7	124	81	4
3	0	35	181	102	0	485	0	37	78	71
4	16	141	77	6	97	149	19	195	54	215

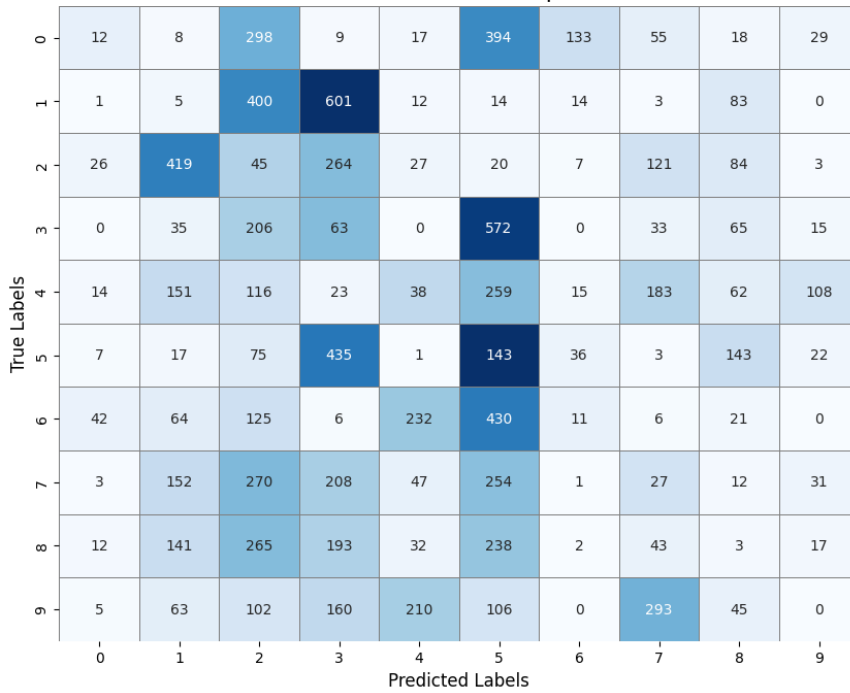


308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.4

Accuracy on adversarial examples: 3.53%

Confusion Matrix for eps=0.4

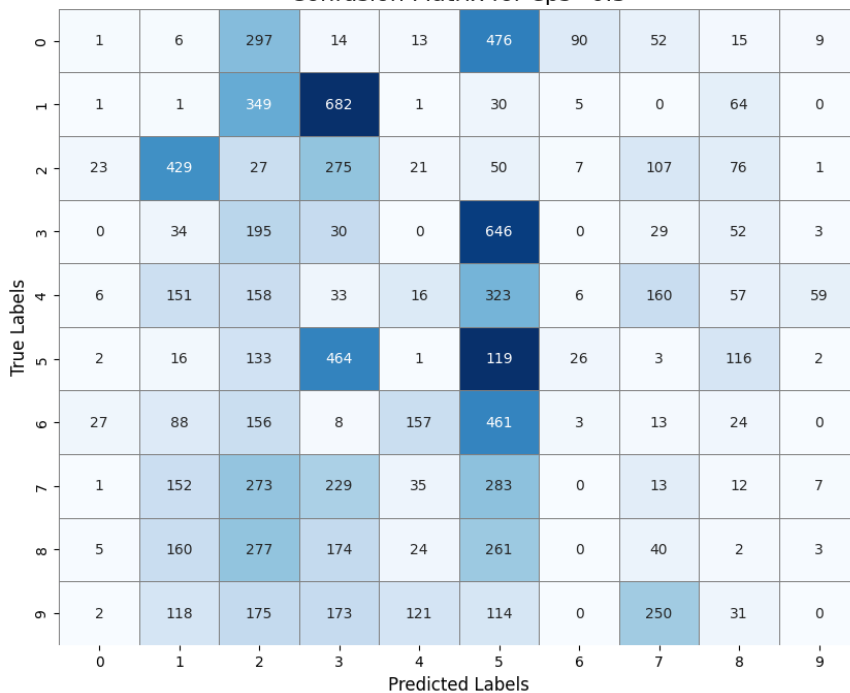


308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.5

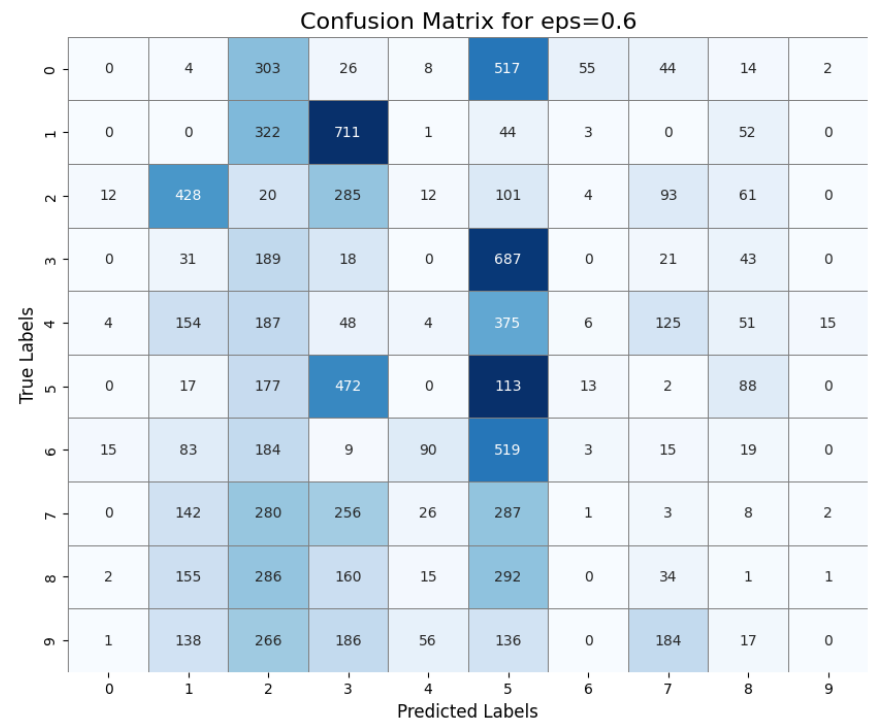
Accuracy on adversarial examples: 2.16%

Confusion Matrix for eps=0.5



308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.6  
 Accuracy on adversarial examples: 1.65%



```
correct_examples.shape
```

```
(9834, 28, 28, 1)
```

```
images.shape
```

```
TensorShape([9834, 28, 28, 1])
```

```
correct_labels.shape
```

```
(9834,)
```

```
labels.shape
```

```
TensorShape([9834])
```

```
robust_accuracy
```

```
<tf.Tensor: shape=(11,), dtype=float32, numpy=
array([0.9938987 , 0.9855603 , 0.9711206 , 0.9552573 , 0.92861503,
        0.6951393 , 0.17937768, 0.02379501, 0.0043726 , 0.00274557,
        0.0025422 ], dtype=float32)>
```

➤ Calculate and report the robust accuracy

```
# Calculate and report the robust accuracy
# Convert 'success' to float, and calculate the mean
robust_accuracy = 1 - tf.reduce_mean(tf.cast(success, tf.float32), axis=-1)
print("robust accuracy for perturbations with")
for eps, acc in zip(epsilons, robust_accuracy):
    print(f" Linf norm ≤ {eps:<6}: {acc.numpy() * 100:4.1f} %")
```

```
robust accuracy for perturbations with
Linf norm ≤ 0.01 : 99.4 %
Linf norm ≤ 0.02 : 98.6 %
Linf norm ≤ 0.03 : 97.1 %
Linf norm ≤ 0.04 : 95.5 %
Linf norm ≤ 0.05 : 92.9 %
Linf norm ≤ 0.1 : 69.5 %
Linf norm ≤ 0.2 : 17.9 %
Linf norm ≤ 0.3 : 2.4 %
Linf norm ≤ 0.4 : 0.4 %
Linf norm ≤ 0.5 : 0.3 %
Linf norm ≤ 0.6 : 0.3 %
```

```
import numpy as np
import tensorflow as tf
```

```
# Assuming 'clipped_advs' and 'images' are available as tensors
for eps, advs_ in zip(epsilons, clipped_advs):
    # Flatten the spatial dimensions of the adversarial and original images
    original_flat = tf.reshape(images, [images.shape[0], -1])
    adversarial_flat = tf.reshape(advs_, [advs_.shape[0], -1])

    # Calculate Linf norm using TensorFlow
    perturbation_sizes = tf.norm(adversarial_flat - original_flat, ord=np.inf, axis=1)
    print(f"Linf norm ≤ {eps:<6}: perturbation sizes:", perturbation_sizes.numpy())
```

```
Linf norm ≤ 0.01 : perturbation sizes: [0.01000001 0.01000001 0.01000001 ... 0.01000001 0.01000002 0.01000002]
Linf norm ≤ 0.02 : perturbation sizes: [0.02000001 0.02000001 0.02000001 ... 0.02000001 0.02000001 0.02000001]
Linf norm ≤ 0.03 : perturbation sizes: [0.03 0.03 0.03 ... 0.03 0.03 0.03]
Linf norm ≤ 0.04 : perturbation sizes: [0.04000002 0.04000002 0.04000002 ... 0.04000002 0.04000002 0.04000002]
Linf norm ≤ 0.05 : perturbation sizes: [0.05000001 0.05000001 0.05000001 ... 0.05000001 0.05000001 0.05000001]
Linf norm ≤ 0.1 : perturbation sizes: [0.10000002 0.10000002 0.10000002 ... 0.10000002 0.10000002 0.10000002]
Linf norm ≤ 0.2 : perturbation sizes: [0.20000002 0.20000002 0.20000002 ... 0.20000002 0.20000002 0.20000002]
Linf norm ≤ 0.3 : perturbation sizes: [0.30000004 0.30000004 0.30000004 ... 0.30000004 0.30000004 0.30000004]
Linf norm ≤ 0.4 : perturbation sizes: [0.40000004 0.40000004 0.40000004 ... 0.40000004 0.40000004 0.40000004]
Linf norm ≤ 0.5 : perturbation sizes: [0.5 0.5 0.5 ... 0.5 0.5 0.5]
Linf norm ≤ 0.6 : perturbation sizes: [0.6 0.6 0.6 ... 0.6 0.6 0.6]
```

## ▼ ProjectedGradientDescentAttack

```
import numpy as np
import tensorflow as tf
from keras.models import load_model
import foolbox as fb
import eagerpy as ep
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

```
# Load your trained model
model = load_model('/content/drive/MyDrive/ColabNotebooks/mnist_model.h5')
```

```
# Load and preprocess the examples
correct_examples = np.load('/content/drive/MyDrive/ColabNotebooks/correct_examples.npy')
correct_labels = np.load('/content/drive/MyDrive/ColabNotebooks/correct_labels.npy').astype('int32') # Cast labels to int32
```

```
# Create a Foolbox model
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))
```

```
# Calculate accuracy on clean data
clean_predictions = model.predict(correct_examples).argmax(axis=-1)
accuracy_clean = np.mean(clean_predictions == correct_labels)
print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")
```

```
# Convert data to TensorFlow tensors
images = tf.convert_to_tensor(correct_examples)
labels = tf.convert_to_tensor(correct_labels, dtype=tf.int32) # Cast labels to int32
```

```
# Create a Foolbox model for TensorFlow
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))
```



```

# Apply an FGSM attack
attack = fb.attacks.LinfPGD()
epsilons = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
raw_advs, clipped_advs, success = attack(fmodel, images, labels, epsilons=epsilons)

# Assuming 'clipped_advs' are the adversarial examples for different epsilons
for eps, advs_ in zip(epsilons, clipped_advs):
    # Predict the labels of the adversarial examples
    y_adv = np.argmax(model.predict(advs_), axis=1)

    # Calculate accuracy on adversarial examples
    accuracy_adv = np.mean(y_adv == correct_labels)
    print(f"\nAdversarial 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
    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
    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)

    # Save the plot
    image_filename = f'confusion_matrix_eps_{eps}.png'
    plt.savefig(image_filename, bbox_inches='tight')
    plt.show() # Display the figure in the notebook

```

```
308/308 [=====] - 1s 3ms/step
```

Accuracy on adversarial examples: 99.20%

	0	1	2	3	4	5	6	7	8	9
0	970	0	0	0	0	1	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
2	1	3	1007	1	0	0	0	2	1	1
3	0	1	1	985	0	1	0	0	0	1
4	0	0	2	0	961	0	2	0	0	4
5	0	0	1	1	0	880	0	0	0	0
6	0	0	0	1	3	1	932	0	0	0
7	0	3	3	0	0	1	0	994	0	4
8	1	3	4	2	4	2	2	1	921	6
9	0	1	0	1	3	0	0	6	0	973

Accuracy on adversarial examples: 97.86%

	0	1	2	3	4	5	6	7	8	9
0	968	0	1	0	0	2	0	0	2	0
1	0	1130	0	0	0	1	1	0	1	0
2	1	8	992	4	1	1	1	4	3	1
3	0	1	3	974	0	6	0	3	1	1
4	0	3	3	0	947	0	3	0	2	11
5	1	2	1	2	1	872	2	0	1	0
6	0	1	1	1	8	4	920	1	1	0
7	0	9	6	2	0	2	0	975	2	9
8	4	8	10	8	6	6	4	1	893	6
9	1	4	1	1	5	3	0	12	4	953

Accuracy on adversarial examples: 95.41%

[illegible]

4	1	6	5	0	930	0	4	2	3	18
5	2	3	1	8	1	859	5	0	1	2
6	4	4	2	1	11	13	900	1	1	0
7	0	15	16	3	5	2	0	945	2	17
8	5	15	25	15	9	19	7	3	832	16
9	4	4	2	5	13	11	0	26	6	913
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 91.10%

Confusion Matrix for eps=0.04

0	951	0	5	0	2	7	3	0	2	3
1	0	1103	7	3	2	3	8	1	5	1
2	7	31	929	17	7	1	2	15	5	2
3	0	4	19	917	0	28	1	5	4	11
4	2	15	7	0	894	1	4	6	4	36
5	3	6	1	22	1	831	9	0	4	5
6	6	6	5	1	15	31	868	1	4	0
7	0	26	28	6	9	3	0	900	2	31
8	6	29	44	31	13	40	10	10	735	28
9	5	7	3	14	48	20	0	45	11	831
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 84.73%

Confusion Matrix for eps=0.05

0	931	0	6	0	2	15	9	2	4	4
1	1	1086	9	4	2	2	9	3	12	5
2	8	51	869	34	8	1	4	24	13	4
3	0	7	29	868	0	48	1	12	8	16
4	3	26	10	0	841	0	7	9	6	67
5	5	6	1	44	1	782	15	0	13	15
6	12	9	9	2	24	58	814	2	6	1
7	0	37	53	12	13	3	1	818	4	64
8	8	44	80	69	19	64	11	15	595	41
9	6	12	5	18	80	28	0	84	23	728
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.1  
Accuracy on adversarial examples: 22.51%

Confusion Matrix for eps=0.1

0	484	8	170	2	17	111	83	28	16	54
1	3	149	316	47	160	11	55	171	193	28
2	26	292	305	160	31	2	6	110	76	8
3	0	28	141	250	0	340	0	29	86	115
4	13	110	43	0	336	13	16	108	27	303
5	16	16	2	274	8	243	43	2	151	127
6	51	27	59	9	237	318	222	2	11	1
7	4	131	222	77	43	12	0	191	8	317
8	17	94	296	192	31	169	20	37	11	79
9	6	21	15	131	311	103	1	305	68	23
	0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.2  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.2

0	0	25	343	1	22	201	192	107	11	71
1	1	0	507	59	179	23	22	287	49	6
2	32	501	0	209	34	7	9	140	81	3
3	1	73	189	0	1	509	0	35	81	100
4	15	214	70	1	0	30	18	251	23	347
5	15	35	4	424	17	0	46	5	184	152
6	50	55	116	4	306	390	0	4	11	1
7	5	247	317	91	46	29	0	0	2	268
8	11	126	343	188	28	181	9	42	0	18
9	4	38	31	114	289	154	0	326	28	0
	0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.3  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.3

0	0	46	420	4	15	232	92	133	5	26
1	0	0	627	96	86	39	8	269	8	0
2	23	598	0	200	28	8	4	116	37	2
3	0	108	231	0	1	564	0	30	36	19
4	9	299	120	2	0	69	14	249	9	198
5	16	66	9	492	23	0	40	14	154	68
	0	1	2	3	4	5	6	7	8	9

Predicted Labels	0	1	2	3	4	5	6	7	8	9
6	31	81	164	1	272	375	0	9	4	0
7	4	339	352	65	46	66	0	0	1	132
8	3	163	391	142	16	189	5	36	0	1
9	4	59	57	105	233	195	1	325	5	0

Adversarial test data: eps:0.4  
Accuracy on adversarial examples: 0.00%

### Confusion Matrix for eps=0.4

True Labels \ Predicted Labels	0	1	2	3	4	5	6	7
0	0	70	461	2	13	255	38	126
1	0	0	724	107	32	57	0	212
2	19	650	0	188	20	18	2	98
3	0	176	255	0	0	512	0	31
4	5	373	175	3	0	136	9	198
5	6	106	31	527	24	0	34	29
6	12	144	231	3	177	359	0	9
7	3	401	369	70	32	97	0	0
8	1	170	441	94	15	188	1	36
9	4	83	119	100	175	230	0	272

Adversarial test data: eps:0.5  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.5

[illegible]

308/308 [=====] - 1s 3ms/step

## ▸ 5 times ProjectedGradientDescentAttack

```
import numpy as np
import tensorflow as tf
from keras.models import load_model
import foolbox as fb
import eagerpy as ep
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Load your trained model
model = load_model('/content/drive/MyDrive/ColabNotebooks/mnist_model.h5')

# Load and preprocess the examples
correct_examples = np.load('/content/drive/MyDrive/ColabNotebooks/correct_examples.npy')
correct_labels = np.load('/content/drive/MyDrive/ColabNotebooks/correct_labels.npy').astype('int32') # Cast labels to int32

# Create a Foolbox model
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))

# Calculate accuracy on clean data
clean_predictions = model.predict(correct_examples).argmax(axis=-1)
accuracy_clean = np.mean(clean_predictions == correct_labels)
print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")

# Convert data to TensorFlow tensors
images = tf.convert_to_tensor(correct_examples)
labels = tf.convert_to_tensor(correct_labels, dtype=tf.int32) # Cast labels to int32

# Create a Foolbox model for TensorFlow
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))

# Apply an FGSM attack
attack = fb.attacks.LinfPGD()
epsilons = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
raw_advs, clipped_advs, success = attack(fmodel, images, labels, epsilons=epsilons)

# Assuming 'clipped_advs' are the adversarial examples for different epsilons
for eps, advs_ in zip(epsilons, clipped_advs):
    for attack_num in range(1, 6):
        # Predict the labels of the adversarial examples
        y_adv = np.argmax(model.predict(advs_), axis=1)

        # Calculate accuracy on adversarial examples
        accuracy_adv = np.mean(y_adv == correct_labels)
        print(f"\nAdversarial 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
    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
    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
```



308/308 [=====] - 1s 2ms/step  
 Accuracy on clean data: 100.00%  
 308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.01  
 Accuracy on adversarial examples: 99.22%

Confusion Matrix for eps=0.01

0	970	0	0	0	0	1	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
2	1	3	1007	1	0	0	0	2	1	1
3	0	1	1	985	0	1	0	0	0	1
4	0	0	2	0	961	0	2	0	0	4
5	0	0	1	1	0	880	0	0	0	0
6	0	0	0	1	3	1	932	0	0	0
7	0	3	3	0	0	1	0	994	0	4
8	0	3	4	2	4	2	2	1	922	6
9	0	1	0	1	3	0	0	5	0	974
	0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.01  
 Accuracy on adversarial examples: 99.22%

Confusion Matrix for eps=0.01

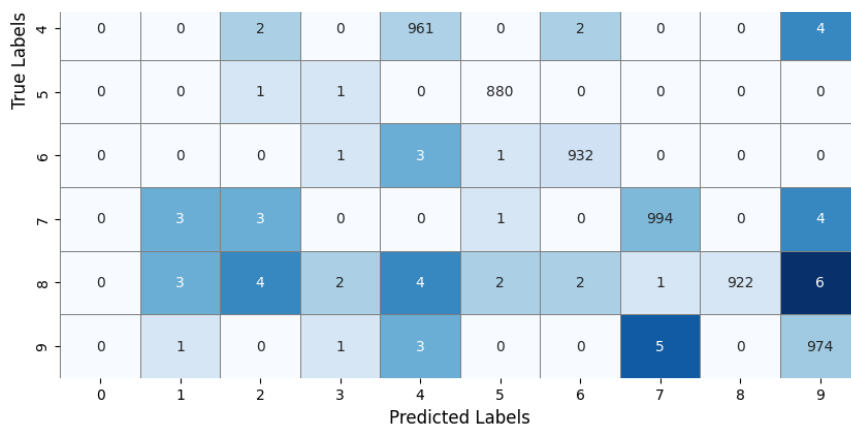
0	970	0	0	0	0	1	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
2	1	3	1007	1	0	0	0	2	1	1
3	0	1	1	985	0	1	0	0	0	1
4	0	0	2	0	961	0	2	0	0	4
5	0	0	1	1	0	880	0	0	0	0
6	0	0	0	1	3	1	932	0	0	0
7	0	3	3	0	0	1	0	994	0	4
8	0	3	4	2	4	2	2	1	922	6
9	0	1	0	1	3	0	0	5	0	974
	0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.01  
 Accuracy on adversarial examples: 99.22%

Confusion Matrix for eps=0.01

0	970	0	0	0	0	1	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
2	1	3	1007	1	0	0	0	2	1	1
3	0	1	1	985	0	1	0	0	0	1

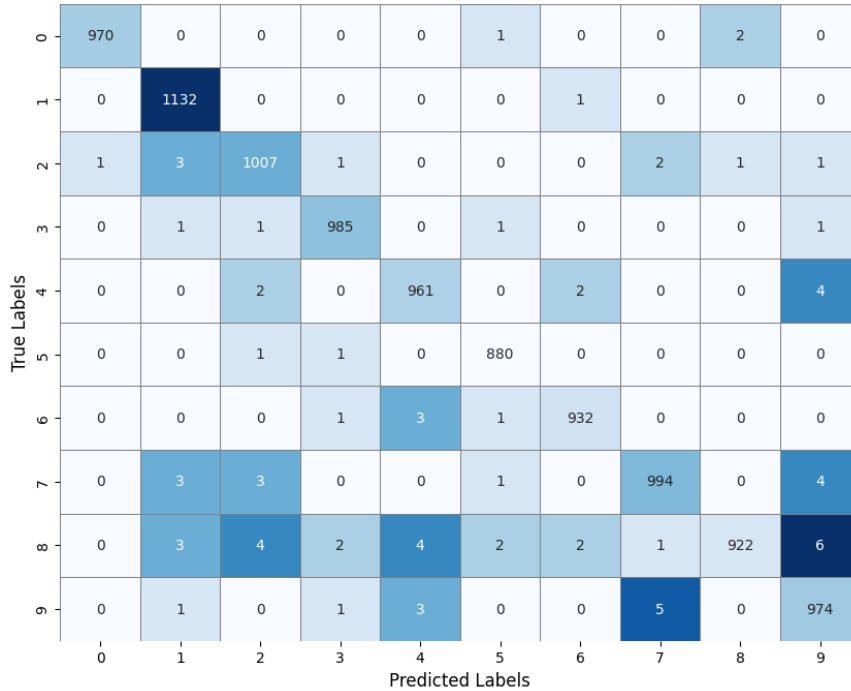


308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.22%

Confusion Matrix for eps=0.01

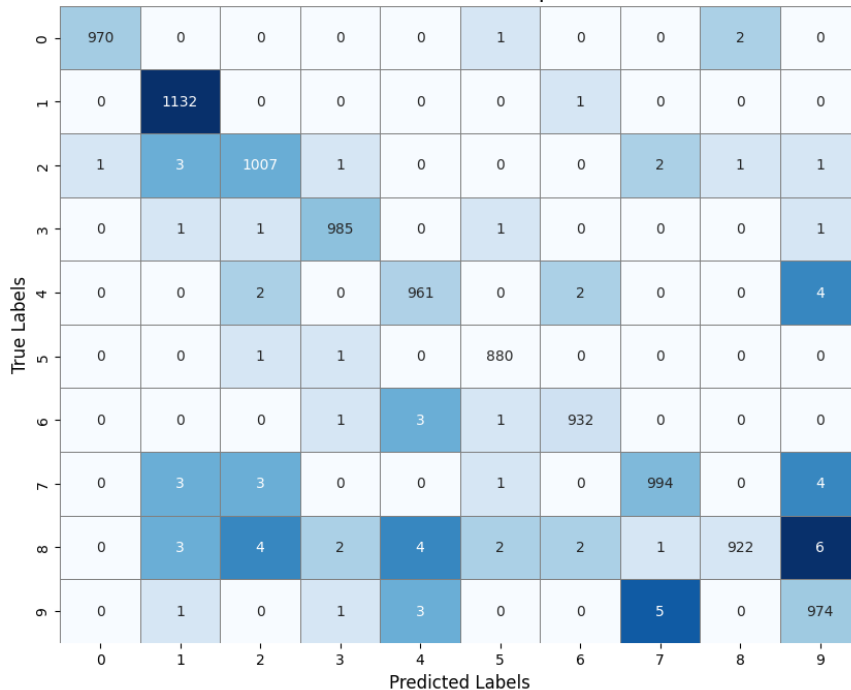


308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.01

Accuracy on adversarial examples: 99.22%

Confusion Matrix for eps=0.01



308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.02  
Accuracy on adversarial examples: 97.83%

Confusion Matrix for eps=0.02

True Labels	0	968	0	1	0	1	2	0	0	1	0
	1	0	1129	0	0	0	1	2	0	1	0
	2	1	8	992	4	1	1	1	4	3	1
	3	0	1	3	974	0	6	0	3	1	1
	4	0	3	3	0	947	0	3	0	2	11
	5	1	2	1	2	1	872	2	0	1	0
	6	0	1	1	1	8	5	919	1	1	0
	7	0	9	6	2	0	2	0	975	2	9
	8	4	7	10	9	6	7	4	1	892	6
	9	1	4	1	1	5	3	0	12	4	953
		Predicted Labels									

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.02  
Accuracy on adversarial examples: 97.83%

Confusion Matrix for eps=0.02

True Labels	0	968	0	1	0	1	2	0	0	1	0
	1	0	1129	0	0	0	1	2	0	1	0
	2	1	8	992	4	1	1	1	4	3	1
	3	0	1	3	974	0	6	0	3	1	1
	4	0	3	3	0	947	0	3	0	2	11
	5	1	2	1	2	1	872	2	0	1	0
	6	0	1	1	1	8	5	919	1	1	0
	7	0	9	6	2	0	2	0	975	2	9
	8	4	7	10	9	6	7	4	1	892	6
	9	1	4	1	1	5	3	0	12	4	953
		Predicted Labels									

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.02  
Accuracy on adversarial examples: 97.83%

Confusion Matrix for eps=0.02

True Labels	0	968	0	1	0	1	2	0	0	1	0
	1	0	1129	0	0	0	1	2	0	1	0
	2	1	8	992	4	1	1	1	4	3	1
	3	0	1	3	974	0	6	0	3	1	1
	4	0	3	3	0	947	0	3	0	2	11
	5	1	2	1	2	1	872	2	0	1	0

True Labels	0	1	2	3	4	5	6	7	8	9
0	0	1	1	1	8	5	919	1	1	0
1	0	9	6	2	0	2	0	975	2	9
2	4	7	10	9	6	7	4	1	892	6
3	1	4	1	1	5	3	0	12	4	953

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.02

Accuracy on adversarial examples: 97.83%

Confusion Matrix for eps=0.02

True Labels	0	1	2	3	4	5	6	7	8	9
0	968	0	1	0	1	2	0	0	1	0
1	0	1129	0	0	0	1	2	0	1	0
2	1	8	992	4	1	1	1	4	3	1
3	0	1	3	974	0	6	0	3	1	1
4	0	3	3	0	947	0	3	0	2	11
5	1	2	1	2	1	872	2	0	1	0
6	0	1	1	1	8	5	919	1	1	0
7	0	9	6	2	0	2	0	975	2	9
8	4	7	10	9	6	7	4	1	892	6
9	1	4	1	1	5	3	0	12	4	953

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.02

Accuracy on adversarial examples: 97.83%

Confusion Matrix for eps=0.02

True Labels	0	1	2	3	4	5	6	7	8	9
0	968	0	1	0	1	2	0	0	1	0
1	0	1129	0	0	0	1	2	0	1	0
2	1	8	992	4	1	1	1	4	3	1
3	0	1	3	974	0	6	0	3	1	1
4	0	3	3	0	947	0	3	0	2	11
5	1	2	1	2	1	872	2	0	1	0
6	0	1	1	1	8	5	919	1	1	0
7	0	9	6	2	0	2	0	975	2	9
8	4	7	10	9	6	7	4	1	892	6
9	1	4	1	1	5	3	0	12	4	953

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.03

Accuracy on adversarial examples: 95.42%

Confusion Matrix for eps=0.03

Confusion Matrix for eps=0.03

0	964	0	2	0	2	2	0	0	2	1
1	0	1120	1	2	0	2	5	0	2	1
2	5	16	964	8	4	1	1	10	6	1
3	0	3	8	955	0	10	0	5	2	6
4	1	6	5	0	930	0	4	2	3	18
5	2	3	1	8	1	859	5	0	1	2
6	3	4	2	1	11	13	901	1	1	0
7	0	15	14	4	5	2	0	946	2	17
8	5	15	25	15	9	19	6	3	832	17
9	4	4	2	5	13	12	0	25	6	913
	0	1	2	3	4	5	6	7	8	9
True Labels	Predicted Labels									

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.03

Accuracy on adversarial examples: 95.42%

Confusion Matrix for eps=0.03

0	964	0	2	0	2	2	0	0	2	1
1	0	1120	1	2	0	2	5	0	2	1
2	5	16	964	8	4	1	1	10	6	1
3	0	3	8	955	0	10	0	5	2	6
4	1	6	5	0	930	0	4	2	3	18
5	2	3	1	8	1	859	5	0	1	2
6	3	4	2	1	11	13	901	1	1	0
7	0	15	14	4	5	2	0	946	2	17
8	5	15	25	15	9	19	6	3	832	17
9	4	4	2	5	13	12	0	25	6	913
	0	1	2	3	4	5	6	7	8	9
True Labels	Predicted Labels									

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.03

Accuracy on adversarial examples: 95.42%

Confusion Matrix for eps=0.03

0	964	0	2	0	2	2	0	0	2	1
1	0	1120	1	2	0	2	5	0	2	1
2	5	16	964	8	4	1	1	10	6	1
3	0	3	8	955	0	10	0	5	2	6
4	1	6	5	0	930	0	4	2	3	18
5	2	3	1	8	1	859	5	0	1	2
6	3	4	2	1	11	13	901	1	1	0

7	0	15	14	4	5	2	0	946	2	17
8	5	15	25	15	9	19	6	3	832	17
9	4	4	2	5	13	12	0	25	6	913
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.03

Accuracy on adversarial examples: 95.42%

Confusion Matrix for eps=0.03

0	964	0	2	0	2	2	0	0	2	1
1	0	1120	1	2	0	2	5	0	2	1
2	5	16	964	8	4	1	1	10	6	1
3	0	3	8	955	0	10	0	5	2	6
4	1	6	5	0	930	0	4	2	3	18
5	2	3	1	8	1	859	5	0	1	2
6	3	4	2	1	11	13	901	1	1	0
7	0	15	14	4	5	2	0	946	2	17
8	5	15	25	15	9	19	6	3	832	17
9	4	4	2	5	13	12	0	25	6	913
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.03

Accuracy on adversarial examples: 95.42%

Confusion Matrix for eps=0.03

0	964	0	2	0	2	2	0	0	2	1
1	0	1120	1	2	0	2	5	0	2	1
2	5	16	964	8	4	1	1	10	6	1
3	0	3	8	955	0	10	0	5	2	6
4	1	6	5	0	930	0	4	2	3	18
5	2	3	1	8	1	859	5	0	1	2
6	3	4	2	1	11	13	901	1	1	0
7	0	15	14	4	5	2	0	946	2	17
8	5	15	25	15	9	19	6	3	832	17
9	4	4	2	5	13	12	0	25	6	913
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

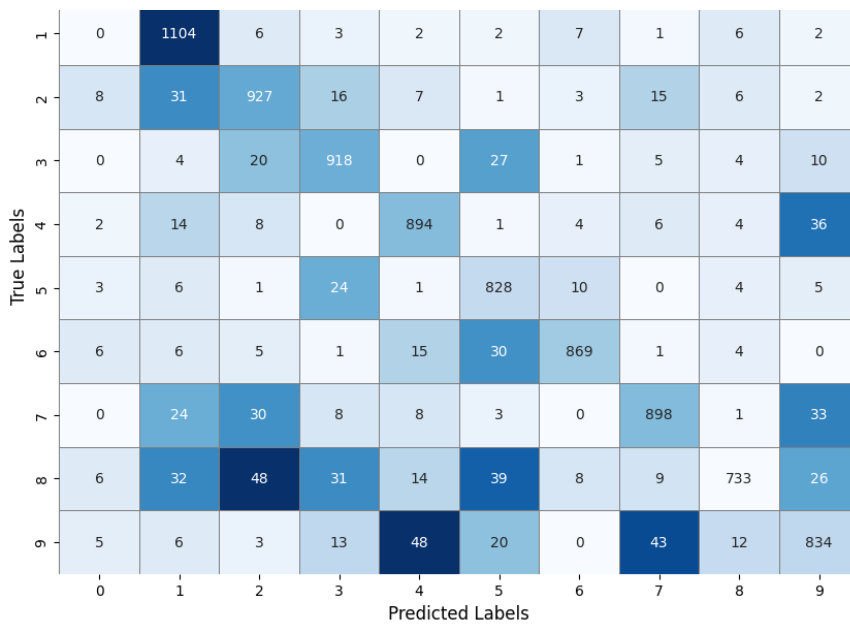
308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 91.07%

Confusion Matrix for eps=0.04

0	951	0	5	0	2	6	4	0	2	3
---	-----	---	---	---	---	---	---	---	---	---

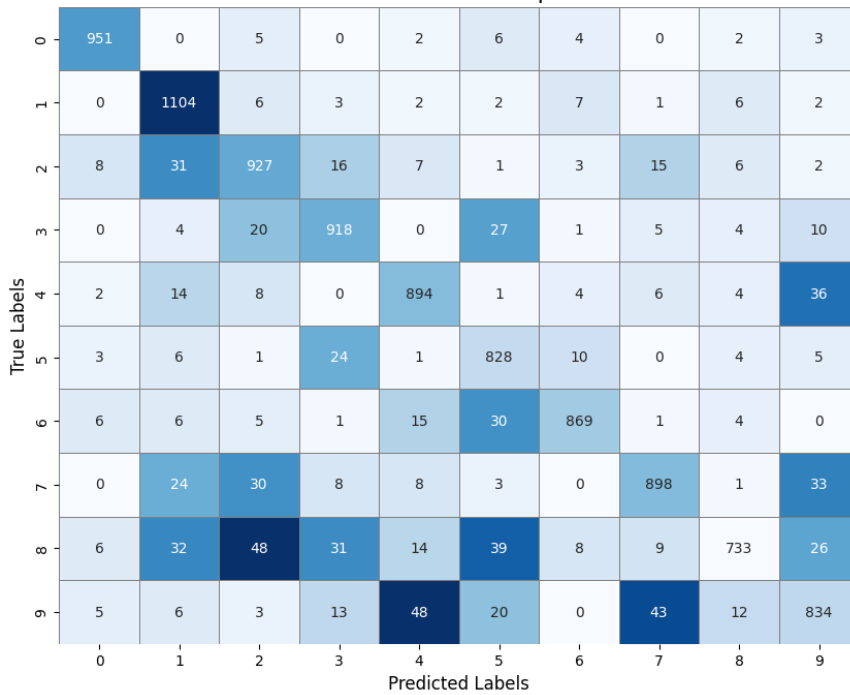


308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 91.07%

Confusion Matrix for eps=0.04

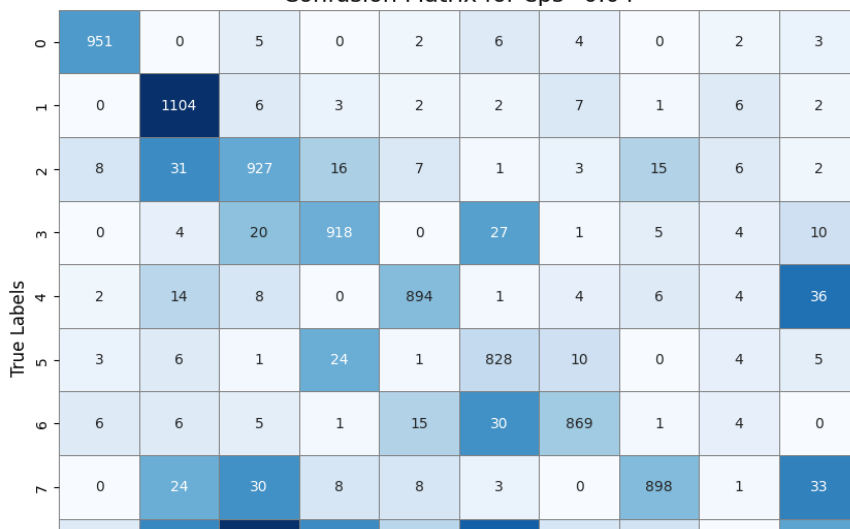


308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 91.07%

Confusion Matrix for eps=0.04



8	6	32	48	31	14	39	8	9	733	26
9	5	6	3	13	48	20	0	43	12	834
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 91.07%

Confusion Matrix for eps=0.04

0	951	0	5	0	2	6	4	0	2	3
1	0	1104	6	3	2	2	7	1	6	2
2	8	31	927	16	7	1	3	15	6	2
3	0	4	20	918	0	27	1	5	4	10
4	2	14	8	0	894	1	4	6	4	36
5	3	6	1	24	1	828	10	0	4	5
6	6	6	5	1	15	30	869	1	4	0
7	0	24	30	8	8	3	0	898	1	33
8	6	32	48	31	14	39	8	9	733	26
9	5	6	3	13	48	20	0	43	12	834
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 91.07%

Confusion Matrix for eps=0.04

0	951	0	5	0	2	6	4	0	2	3
1	0	1104	6	3	2	2	7	1	6	2
2	8	31	927	16	7	1	3	15	6	2
3	0	4	20	918	0	27	1	5	4	10
4	2	14	8	0	894	1	4	6	4	36
5	3	6	1	24	1	828	10	0	4	5
6	6	6	5	1	15	30	869	1	4	0
7	0	24	30	8	8	3	0	898	1	33
8	6	32	48	31	14	39	8	9	733	26
9	5	6	3	13	48	20	0	43	12	834
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 2ms/step

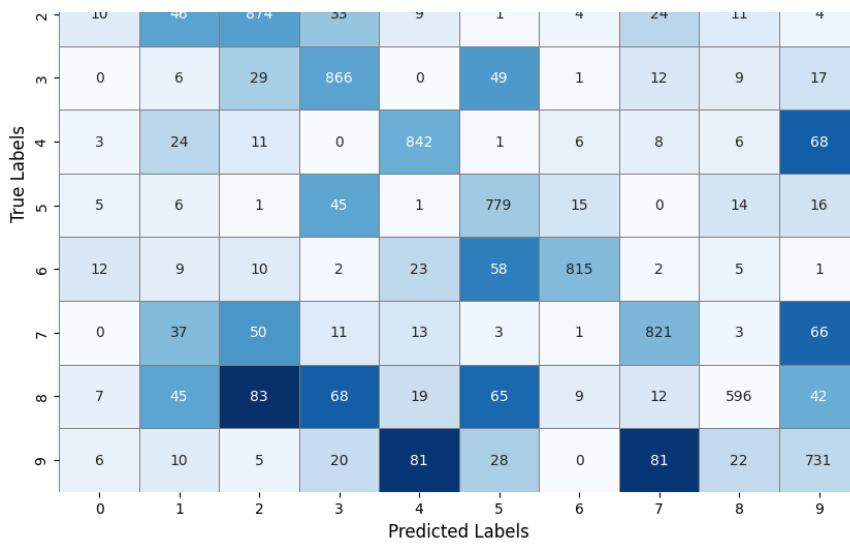
Adversarial test data: eps:0.05

Accuracy on adversarial examples: 84.82%

Confusion Matrix for eps=0.05

0	931	0	7	1	2	16	8	1	4	3
1	1	1086	9	4	2	3	9	2	10	7
2	12	46	874	23	0	1	4	24	11	4



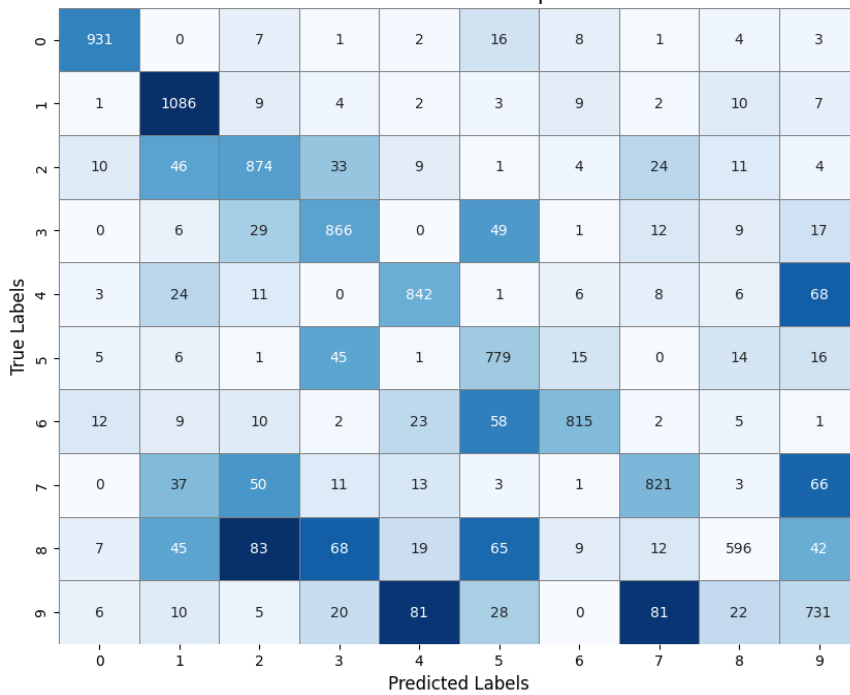


308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 84.82%

Confusion Matrix for eps=0.05

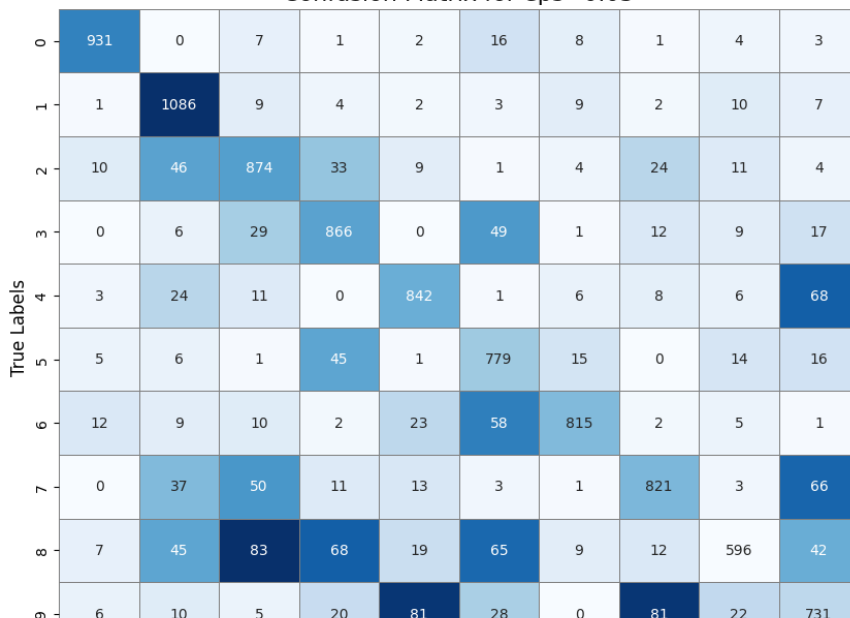


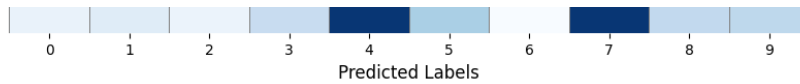
308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 84.82%

Confusion Matrix for eps=0.05





308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.05  
Accuracy on adversarial examples: 84.82%

Confusion Matrix for eps=0.05

True Labels	0	1	2	3	4	5	6	7	8	9
0	931	0	7	1	2	16	8	1	4	3
1	1	1086	9	4	2	3	9	2	10	7
2	10	46	874	33	9	1	4	24	11	4
3	0	6	29	866	0	49	1	12	9	17
4	3	24	11	0	842	1	6	8	6	68
5	5	6	1	45	1	779	15	0	14	16
6	12	9	10	2	23	58	815	2	5	1
7	0	37	50	11	13	3	1	821	3	66
8	7	45	83	68	19	65	9	12	596	42
9	6	10	5	20	81	28	0	81	22	731
Predicted Labels										

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.05  
Accuracy on adversarial examples: 84.82%

Confusion Matrix for eps=0.05

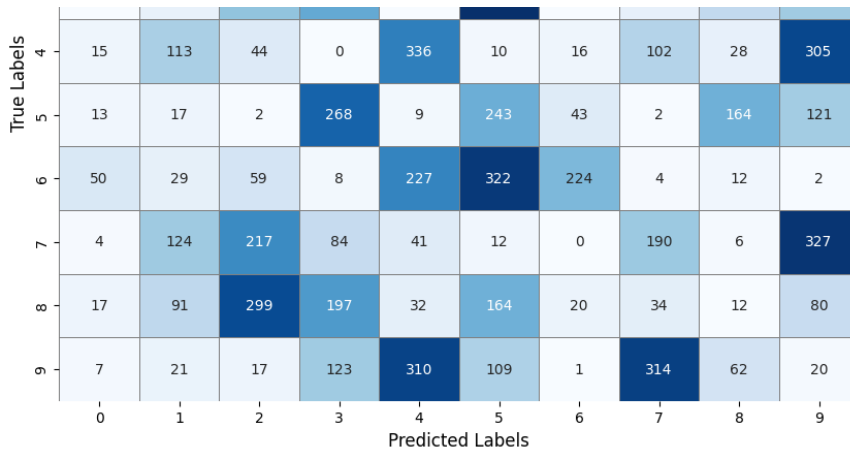
True Labels	0	1	2	3	4	5	6	7	8	9
0	931	0	7	1	2	16	8	1	4	3
1	1	1086	9	4	2	3	9	2	10	7
2	10	46	874	33	9	1	4	24	11	4
3	0	6	29	866	0	49	1	12	9	17
4	3	24	11	0	842	1	6	8	6	68
5	5	6	1	45	1	779	15	0	14	16
6	12	9	10	2	23	58	815	2	5	1
7	0	37	50	11	13	3	1	821	3	66
8	7	45	83	68	19	65	9	12	596	42
9	6	10	5	20	81	28	0	81	22	731
Predicted Labels										

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.1  
Accuracy on adversarial examples: 22.47%

Confusion Matrix for eps=0.1

0	477	9	167	3	18	119	85	24	15	56
1	5	160	320	36	164	9	58	167	187	27
2	28	305	294	159	30	1	7	106	79	7
3	0	26	135	254	1	334	1	28	88	122

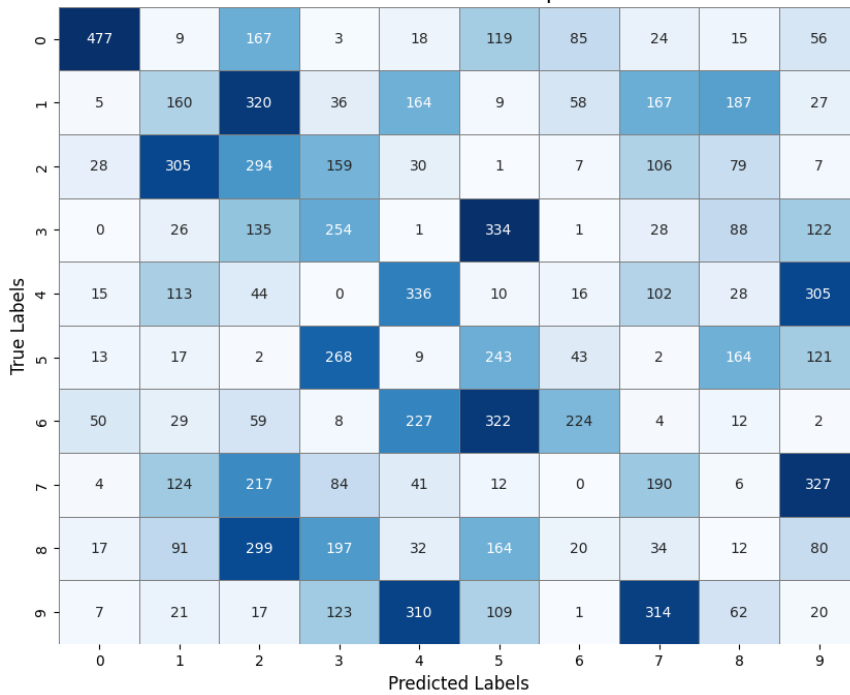


308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.1

Accuracy on adversarial examples: 22.47%

Confusion Matrix for eps=0.1

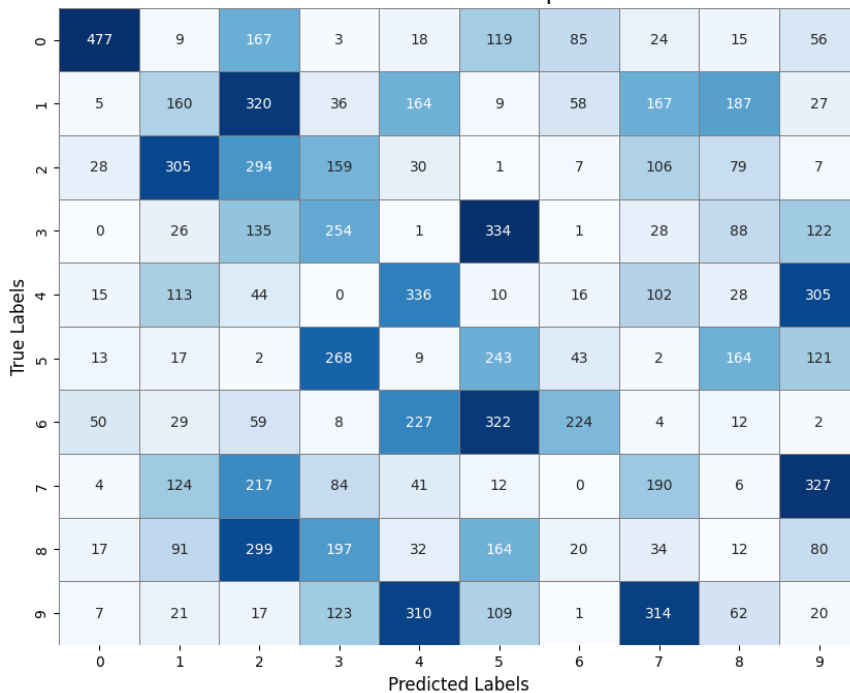


308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.1

Accuracy on adversarial examples: 22.47%

Confusion Matrix for eps=0.1



```
Adversarial test data: eps:0.1
Accuracy on adversarial examples: 22.47%
```

0	477	9	167	3	18	119	85	24	15	56
1	5	160	320	36	164	9	58	167	187	27
2	28	305	294	159	30	1	7	106	79	7
3	0	26	135	254	1	334	1	28	88	122
4	15	113	44	0	336	10	16	102	28	305
5	13	17	2	268	9	243	43	2	164	121
6	50	29	59	8	227	322	224	4	12	2
7	4	124	217	84	41	12	0	190	6	327
8	17	91	299	197	32	164	20	34	12	80
9	7	21	17	123	310	109	1	314	62	20
	0	1	2	3	4	5	6	7	8	9

```
Adversarial test data: eps:0.1
Accuracy on adversarial examples: 22.47%
```

	0	1	2	3	4	5	6	7	8	9
0	477	9	167	3	18	119	85	24	15	56
1	5	160	320	36	164	9	58	167	187	27
2	28	305	294	159	30	1	7	106	79	7
3	0	26	135	254	1	334	1	28	88	122
4	15	113	44	0	336	10	16	102	28	305
5	13	17	2	268	9	243	43	2	164	121
6	50	29	59	8	227	322	224	4	12	2
7	4	124	217	84	41	12	0	190	6	327
8	17	91	299	197	32	164	20	34	12	80
9	7	21	17	123	310	109	1	314	62	20
	0	1	2	3	4	5	6	7	8	9

```
Adversarial test data: eps:0.2
Accuracy on adversarial examples: 0.00%
```

	0	1	2	3	4	5	6	7	8	9
0	0	21	369	2	20	202	177	101	10	71
1	2	0	503	67	176	24	25	283	47	6
2	27	515	0	210	33	4	6	148	69	4
3	0	83	191	0	0	508	1	34	70	102
4	15	211	76	2	0	30	14	245	26	350

True	5	17	32	2	429	17	0	52	6	176	151
	6	47	57	105	4	317	384	0	8	14	1
	7	7	245	320	91	52	32	0	0	2	256
	8	8	128	354	179	25	183	10	41	0	18
	9	7	39	29	119	292	148	0	331	19	0
		0	1	2	3	4	5	6	7	8	9
		Predicted Labels									

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.2

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.2

True Labels	0	0	21	369	2	20	202	177	101	10	71
	1	2	0	503	67	176	24	25	283	47	6
	2	27	515	0	210	33	4	6	148	69	4
	3	0	83	191	0	0	508	1	34	70	102
	4	15	211	76	2	0	30	14	245	26	350
	5	17	32	2	429	17	0	52	6	176	151
	6	47	57	105	4	317	384	0	8	14	1
	7	7	245	320	91	52	32	0	0	2	256
	8	8	128	354	179	25	183	10	41	0	18
	9	7	39	29	119	292	148	0	331	19	0
		0	1	2	3	4	5	6	7	8	9
		Predicted Labels									

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.2

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.2

True Labels	0	0	21	369	2	20	202	177	101	10	71
	1	2	0	503	67	176	24	25	283	47	6
	2	27	515	0	210	33	4	6	148	69	4
	3	0	83	191	0	0	508	1	34	70	102
	4	15	211	76	2	0	30	14	245	26	350
	5	17	32	2	429	17	0	52	6	176	151
	6	47	57	105	4	317	384	0	8	14	1
	7	7	245	320	91	52	32	0	0	2	256
	8	8	128	354	179	25	183	10	41	0	18
	9	7	39	29	119	292	148	0	331	19	0
		0	1	2	3	4	5	6	7	8	9
		Predicted Labels									

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.2

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.2

0	0	21	369	2	20	202	177	101	10	71
1	2	0	503	67	176	24	25	283	47	6
2	27	515	0	210	33	4	6	148	69	4
3	0	83	191	0	0	508	1	34	70	102
4	15	211	76	2	0	30	14	245	26	350
5	17	32	2	429	17	0	52	6	176	151
6	47	57	105	4	317	384	0	8	14	1
7	7	245	320	91	52	32	0	0	2	256
8	8	128	354	179	25	183	10	41	0	18
9	7	39	29	119	292	148	0	331	19	0
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.2

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.2

0	0	21	369	2	20	202	177	101	10	71
1	2	0	503	67	176	24	25	283	47	6
2	27	515	0	210	33	4	6	148	69	4
3	0	83	191	0	0	508	1	34	70	102
4	15	211	76	2	0	30	14	245	26	350
5	17	32	2	429	17	0	52	6	176	151
6	47	57	105	4	317	384	0	8	14	1
7	7	245	320	91	52	32	0	0	2	256
8	8	128	354	179	25	183	10	41	0	18
9	7	39	29	119	292	148	0	331	19	0
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.3

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.3

0	0	54	404	3	17	233	94	139	3	26
1	1	0	634	78	95	42	6	264	13	0
2	23	589	0	188	30	15	3	128	38	2
3	0	130	220	0	1	550	0	36	29	23
4	6	305	127	0	0	76	8	239	6	202
5	10	68	12	495	20	0	45	11	138	83
6	25	100	150	4	248	305	0	10	5	0

True Labels

6	23	100	150	4	248	395	0	10	5	0
7	2	329	361	69	45	69	0	0	0	130
8	5	151	398	141	19	182	3	44	0	3
9	4	71	62	114	237	173	1	314	8	0
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.3  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.3

0	0	54	404	3	17	233	94	139	3	26
1	1	0	634	78	95	42	6	264	13	0
2	23	589	0	188	30	15	3	128	38	2
3	0	130	220	0	1	550	0	36	29	23
4	6	305	127	0	0	76	8	239	6	202
5	10	68	12	495	20	0	45	11	138	83
6	25	100	150	4	248	395	0	10	5	0
7	2	329	361	69	45	69	0	0	0	130
8	5	151	398	141	19	182	3	44	0	3
9	4	71	62	114	237	173	1	314	8	0
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.3  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.3

0	0	54	404	3	17	233	94	139	3	26
1	1	0	634	78	95	42	6	264	13	0
2	23	589	0	188	30	15	3	128	38	2
3	0	130	220	0	1	550	0	36	29	23
4	6	305	127	0	0	76	8	239	6	202
5	10	68	12	495	20	0	45	11	138	83
6	25	100	150	4	248	395	0	10	5	0
7	2	329	361	69	45	69	0	0	0	130
8	5	151	398	141	19	182	3	44	0	3
9	4	71	62	114	237	173	1	314	8	0
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.3  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.3

0	0	54	404	3	17	233	94	139	3	26
---	---	----	-----	---	----	-----	----	-----	---	----

True Labels	1	1	0	634	78	95	42	6	264	13	0
	2	23	589	0	188	30	15	3	128	38	2
	3	0	130	220	0	1	550	0	36	29	23
	4	6	305	127	0	0	76	8	239	6	202
	5	10	68	12	495	20	0	45	11	138	83
	6	25	100	150	4	248	395	0	10	5	0
	7	2	329	361	69	45	69	0	0	0	130
	8	5	151	398	141	19	182	3	44	0	3
	9	4	71	62	114	237	173	1	314	8	0
			0	1	2	3	4	5	6	7	8
Predicted Labels											

Adversarial test data: eps:0.3

Accuracy on adversarial examples: 0.00%

### Confusion Matrix for eps=0.3

True Labels \ Predicted Labels	0	1	2	3	4	5	6	7	8	9
0	0	54	404	3	17	233	94	139	3	26
1	1	0	634	78	95	42	6	264	13	0
2	23	589	0	188	30	15	3	128	38	2
3	0	130	220	0	1	550	0	36	29	23
4	6	305	127	0	0	76	8	239	6	202
5	10	68	12	495	20	0	45	11	138	83
6	25	100	150	4	248	395	0	10	5	0
7	2	329	361	69	45	69	0	0	0	130
8	5	151	398	141	19	182	3	44	0	3
9	4	71	62	114	237	173	1	314	8	0

Adversarial test data: eps:0.4

Accuracy on adversarial examples: 0.00%

### Confusion Matrix for eps=0.4

	0	1	2	3	4	5	6	7		
0	0	53	476	0	13	260	41	122	2	6
1	0	0	746	114	31	56	0	185	1	0
2	11	671	0	169	21	22	4	97	20	1
3	0	180	225	0	1	544	0	26	11	2
4	7	355	171	1	0	141	9	195	6	84
5	10	107	32	519	21	0	30	24	114	25
6	14	135	236	3	169	366	0	10	3	1
7	2	396	355	87	30	94	0	0	0	41



8	2	194	424	96	16	183	1	30	0
9	2	94	102	105	172	230	1	278	0
	0	1	2	3	4	5	6	7	8
Predicted Labels									

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.4

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.4

0	0	53	476	0	13	260	41	122	2	6
1	0	0	746	114	31	56	0	185	1	0
2	11	671	0	169	21	22	4	97	20	1
3	0	180	225	0	1	544	0	26	11	2
4	7	355	171	1	0	141	9	195	6	84
5	10	107	32	519	21	0	30	24	114	25
6	14	135	236	3	169	366	0	10	3	1
7	2	396	355	87	30	94	0	0	0	41
8	2	194	424	96	16	183	1	30	0	0
9	2	94	102	105	172	230	1	278	0	0
	0	1	2	3	4	5	6	7	8	9
True Labels										
Predicted Labels										

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.4

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.4

0	0	53	476	0	13	260	41	122	2	6
1	0	0	746	114	31	56	0	185	1	0
2	11	671	0	169	21	22	4	97	20	1
3	0	180	225	0	1	544	0	26	11	2
4	7	355	171	1	0	141	9	195	6	84
5	10	107	32	519	21	0	30	24	114	25
6	14	135	236	3	169	366	0	10	3	1
7	2	396	355	87	30	94	0	0	0	41
8	2	194	424	96	16	183	1	30	0	0
9	2	94	102	105	172	230	1	278	0	0
	0	1	2	3	4	5	6	7	8	9
True Labels										
Predicted Labels										

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.4

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.4

0	0	53	476	0	13	260	41	122	2	6
1	0	0	746	114	31	56	0	185	1	0

True Labels	2	11	671	0	169	21	22	4	97	20	1
	3	0	180	225	0	1	544	0	26	11	2
	4	7	355	171	1	0	141	9	195	6	84
	5	10	107	32	519	21	0	30	24	114	25
	6	14	135	236	3	169	366	0	10	3	1
	7	2	396	355	87	30	94	0	0	0	41
	8	2	194	424	96	16	183	1	30	0	0
	9	2	94	102	105	172	230	1	278	0	0
		0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.4

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.4

True Labels	0	0	53	476	0	13	260	41	122	2	6
	1	0	0	746	114	31	56	0	185	1	0
	2	11	671	0	169	21	22	4	97	20	1
	3	0	180	225	0	1	544	0	26	11	2
	4	7	355	171	1	0	141	9	195	6	84
	5	10	107	32	519	21	0	30	24	114	25
	6	14	135	236	3	169	366	0	10	3	1
	7	2	396	355	87	30	94	0	0	0	41
	8	2	194	424	96	16	183	1	30	0	0
	9	2	94	102	105	172	230	1	278	0	0
		0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.5

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.5

True Labels	0	0	71	511	0	7	285	12	85	0	2
	1	0	0	818	132	6	59	0	117	1	0
	2	7	712	0	162	15	34	2	76	8	0
	3	0	212	255	0	0	496	0	21	5	0
	4	3	377	234	8	0	189	7	128	0	23
	5	3	157	90	479	14	0	23	32	77	7
	6	3	155	317	5	93	359	0	4	1	0
	7	1	434	358	57	20	125	0	0	0	10
	8	1	186	467	83	6	178	0	25	0	0
		0	1	2	3	4	5	6	7	8	9

Predicted Labels



3	0	212	255	0	0	496	0	21	5	0
4	3	377	234	8	0	189	7	128	0	23
5	3	157	90	479	14	0	23	32	77	7
6	3	155	317	5	93	359	0	4	1	0
7	1	434	358	57	20	125	0	0	0	10
8	1	186	467	83	6	178	0	25	0	0
9	3	131	166	79	104	284	0	217	0	0
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.5

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.5

0	0	71	511	0	7	285	12	85	0	2
1	0	0	818	132	6	59	0	117	1	0
2	7	712	0	162	15	34	2	76	8	0
3	0	212	255	0	0	496	0	21	5	0
4	3	377	234	8	0	189	7	128	0	23
5	3	157	90	479	14	0	23	32	77	7
6	3	155	317	5	93	359	0	4	1	0
7	1	434	358	57	20	125	0	0	0	10
8	1	186	467	83	6	178	0	25	0	0
9	3	131	166	79	104	284	0	217	0	0
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.6

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.6

0	0	102	509	1	2	284	8	66	1	0
1	0	0	854	134	5	68	0	72	0	0
2	3	743	0	152	7	63	2	43	3	0
3	0	249	284	0	0	437	0	18	1	0
4	1	401	258	6	0	218	5	72	0	8
5	4	223	178	377	10	0	12	43	35	0
6	1	181	373	5	45	327	0	5	0	0
7	2	487	311	58	12	133	0	0	0	2
8	0	181	510	51	3	177	2	22	0	0
9	1	169	212	77	51	316	0	158	0	0
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

Predicted Labels  
308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.6  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.6

True Labels	0	1	2	3	4	5	6	7	8	9
	0	102	509	1	2	284	8	66	1	0
	1	0	854	134	5	68	0	72	0	0
	2	3	743	0	152	7	63	2	43	3
	3	0	249	284	0	0	437	0	18	1
	4	1	401	258	6	0	218	5	72	0
	5	4	223	178	377	10	0	12	43	35
	6	1	181	373	5	45	327	0	5	0
	7	2	487	311	58	12	133	0	0	0
	8	0	181	510	51	3	177	2	22	0
	9	1	169	212	77	51	316	0	158	0
Predicted Labels										

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.6  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.6

True Labels	0	1	2	3	4	5	6	7	8	9
	0	102	509	1	2	284	8	66	1	0
	1	0	854	134	5	68	0	72	0	0
	2	3	743	0	152	7	63	2	43	3
	3	0	249	284	0	0	437	0	18	1
	4	1	401	258	6	0	218	5	72	0
	5	4	223	178	377	10	0	12	43	35
	6	1	181	373	5	45	327	0	5	0
	7	2	487	311	58	12	133	0	0	0
	8	0	181	510	51	3	177	2	22	0
	9	1	169	212	77	51	316	0	158	0
Predicted Labels										

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.6  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.6

True Labels	0	1	2	3	4	5	6	7	8	9
	0	102	509	1	2	284	8	66	1	0
	1	0	854	134	5	68	0	72	0	0
	2	3	743	0	152	7	63	2	43	3
	3	0	249	284	0	0	437	0	18	1
	4	1	401	258	6	0	218	5	72	0













## ▼ DeepFoolAttack

```
import numpy as np
import tensorflow as tf
from keras.models import load_model
import foolbox as fb
import eagerpy as ep
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Load your trained model
model = load_model('/content/drive/MyDrive/ColabNotebooks/mnist_model.h5')

# Load and preprocess the examples
correct_examples = np.load('/content/drive/MyDrive/ColabNotebooks/correct_examples.npy')
correct_labels = np.load('/content/drive/MyDrive/ColabNotebooks/correct_labels.npy').astype('int32') # Cast labels to int32

# Create a Foolbox model
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))

# Calculate accuracy on clean data
clean_predictions = model.predict(correct_examples).argmax(axis=-1)
accuracy_clean = np.mean(clean_predictions == correct_labels)
print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")

# Convert data to TensorFlow tensors
images = tf.convert_to_tensor(correct_examples)
labels = tf.convert_to_tensor(correct_labels, dtype=tf.int32) # Cast labels to int32

# Create a Foolbox model for TensorFlow
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))

# Apply an FGSM attack
attack = fb.attacks.LinfDeepFoolAttack()
epsilons = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
raw_advs, clipped_advs, success = attack(fmodel, images, labels, epsilons=epsilons)

# Assuming 'clipped_advs' are the adversarial examples for different epsilons
for eps, advs_ in zip(epsilons, clipped_advs):
    # Predict the labels of the adversarial examples
    y_adv = np.argmax(model.predict(advs_), axis=1)

    # Calculate accuracy on adversarial examples
    accuracy_adv = np.mean(y_adv == correct_labels)
    print(f"\nAdversarial 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
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
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)

# Save the plot
image_filename = f'confusion_matrix_eps_{eps}.png'
plt.savefig(image_filename, bbox_inches='tight')
plt.show() # Display the figure in the notebook
```

308/308 [=====] - 1s 2ms/step  
 Accuracy on clean data: 100.00%  
 308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.01  
 Accuracy on adversarial examples: 99.33%

Confusion Matrix for eps=0.01

0	970	0	0	0	0	1	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
2	0	2	1009	1	0	0	0	2	1	1
3	0	1	1	985	0	1	0	0	0	1
4	0	0	2	0	962	0	1	0	0	4
5	0	0	1	1	0	880	0	0	0	0
6	0	0	0	1	2	1	933	0	0	0
7	0	3	1	0	0	1	0	998	0	2
8	0	2	3	1	4	2	2	1	925	6
9	0	1	0	1	3	0	0	5	0	974
	0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.02  
 Accuracy on adversarial examples: 98.53%

Confusion Matrix for eps=0.02

0	968	0	1	0	0	2	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
2	1	8	998	2	1	1	0	3	1	1
3	0	1	2	979	0	4	0	2	0	1
4	0	2	3	0	955	0	2	0	0	7
5	1	1	1	1	0	877	1	0	0	0
6	0	0	1	1	5	2	928	0	0	0
7	0	6	6	1	0	1	0	986	0	5
8	4	5	9	6	6	3	3	1	903	6
9	1	2	0	1	5	1	0	9	2	963
	0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.03  
 Accuracy on adversarial examples: 97.06%

Confusion Matrix for eps=0.03

0	966	0	2	0	1	2	0	0
1	0	1131	0	0	0	1	1	0
2	3	13	981	6	1	1	1	5

True Labels	3	0	2	5	970	0	6	0	4
	4	1	4	3	0	944	0	3	1
	5	1	3	1	2	1	870	3	0
	6	2	1	1	1	10	8	912	1
	7	0	11	6	2	0	1	0	973
	8	7	10	12	9	9	14	5	2
	9	4	3	1	5	8	6	0	19
		0	1	2	3	4	5	6	7
		Predicted Labels							

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 95.43%

Confusion Matrix for eps=0.04

True Labels	0	962	0	3	0	1	2	1	0	3	1
	1	0	1129	0	0	0	1	1	1	1	0
	2	5	17	965	9	3	1	1	9	5	1
	3	0	3	9	951	0	13	0	5	3	5
	4	1	6	5	0	931	0	4	1	3	18
	5	1	3	1	7	1	863	4	0	1	1
	6	5	3	2	1	11	14	899	1	1	0
	7	0	15	11	2	4	1	0	955	2	15
	8	7	14	26	16	11	20	7	4	821	20
	9	4	3	2	7	15	11	0	27	6	909
		0	1	2	3	4	5	6	7	8	9
		Predicted Labels									

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 92.75%

Confusion Matrix for eps=0.05

True Labels	0	955	0	4	0	1	5	3	0	3	2
	1	0	1125	1	2	0	1	2	1	1	0
	2	7	31	941	13	4	1	1	12	5	1
	3	0	4	15	926	0	25	0	5	4	10
	4	2	13	6	0	917	0	4	2	3	22
	5	1	4	1	12	1	849	8	0	3	3
	6	7	4	4	1	14	26	878	1	2	0
	7	0	20	18	3	8	1	0	932	2	21

8	9	21	39	28	16	37	9	9	750	28
9	5	5	3	15	42	17	0	38	11	848
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.1  
Accuracy on adversarial examples: 69.04%

Confusion Matrix for eps=0.1

0	848	1	27	2	7	33	24	6	11	14
1	1	1098	11	3	2	2	9	3	3	1
2	18	138	673	87	12	1	4	52	27	4
3	0	10	64	695	0	130	0	17	34	39
4	7	52	19	0	735	0	12	26	9	109
5	6	6	1	80	2	725	20	1	23	18
6	31	16	18	2	62	160	635	2	10	1
7	1	57	80	15	24	4	1	735	5	83
8	21	58	183	149	35	122	22	33	234	89
9	6	9	7	59	194	59	1	190	48	411
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.2  
Accuracy on adversarial examples: 21.80%

Confusion Matrix for eps=0.2

0	230	3	233	3	17	145	196	46	30	70
1	4	558	271	41	14	7	21	135	72	10
2	29	356	189	178	26	1	7	127	96	7
3	0	26	137	211	0	361	0	29	89	136
4	22	106	40	0	262	30	22	131	47	309
5	10	16	4	271	7	302	47	2	134	89
6	77	30	49	5	259	360	120	4	30	3
7	5	130	209	89	45	18	1	258	9	241
8	24	80	264	188	41	164	27	40	9	109
9	7	13	12	146	293	93	1	299	115	5
	0	1	2	3	4	5	6	7	8	9

True Labels

Predicted Labels

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.3  
Accuracy on adversarial examples: 7.02%

Confusion Matrix for eps=0.3

0	65	6	274	4	17	250	196	67	29	65
1	2	131	427	331	41	12	36	49	101	3

True Labels	2	32	400	78	214	28	6	8	138	107	5
	3	0	32	169	93	0	470	0	32	97	96
	4	23	130	68	7	85	147	22	191	62	234
	5	10	18	20	373	4	147	48	3	173	86
	6	67	40	74	8	289	399	22	4	32	2
	7	5	146	234	153	46	120	2	65	9	225
	8	21	93	271	189	38	189	21	41	4	79
	9	7	22	26	149	280	95	1	304	100	0
	Predicted Labels										

Adversarial test data: eps:0.4

Accuracy on adversarial examples: 2.64%

Confusion Matrix for eps=0.4

	0	1	2	3	4	5	6	7	8	9
0	8	5	283	13	17	357	160	71	25	34
1	1	5	438	558	13	18	17	5	78	0
2	32	417	32	242	27	15	7	132	109	3
3	0	32	190	57	0	565	0	34	83	28
4	20	136	109	21	34	262	19	175	66	127
5	7	15	73	440	2	94	39	5	170	37
6	49	56	107	8	253	419	8	5	32	0
7	4	149	241	203	48	261	2	20	15	62
8	17	112	280	185	35	233	10	41	2	31
9	6	46	66	151	247	102	1	297	68	0
	0	1	2	3	4	5	6	7	8	9

Adversarial test data: eps:0.5

Accuracy on adversarial examples: 1.52%

Confusion Matrix for eps=0.5

[illegible]



9	5	102	134	170	162	115	0	250	46	0
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.6

Accuracy on adversarial examples: 0.95%

Confusion Matrix for eps=0.6

0	0	5	289	24	10	508	62	51	22	2
1	0	0	373	663	1	37	6	1	52	0
2	20	424	14	267	16	83	5	97	89	1
3	0	31	188	8	0	678	0	23	50	11
4	9	141	181	45	4	360	8	124	58	39
5	1	14	154	507	1	63	25	3	110	4
6	20	84	186	12	95	506	0	9	25	0
7	1	142	257	236	35	289	2	4	9	30
8	8	122	283	173	24	273	8	33	0	22
9	5	125	227	191	76	134	0	201	25	0
	0	1	2	3	4	5	6	7	8	9

Predicted Labels

## ▼ AdamPGD

```
import numpy as np
import tensorflow as tf
from keras.models import load_model
import foolbox as fb
import eagerpy as ep
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Load your trained model
model = load_model('/content/drive/MyDrive/ColabNotebooks/mnist_model.h5')

# Load and preprocess the examples
```

```

correct_examples = np.load('/content/drive/MyDrive/ColabNotebooks/correct_examples.npy')
correct_labels = np.load('/content/drive/MyDrive/ColabNotebooks/correct_labels.npy').astype('int32') # Cast labels to int32

# Create a Foolbox model
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))

# Calculate accuracy on clean data
clean_predictions = model.predict(correct_examples).argmax(axis=-1)
accuracy_clean = np.mean(clean_predictions == correct_labels)
print(f"Accuracy on clean data: {accuracy_clean * 100:.2f}%")

# Convert data to TensorFlow tensors
images = tf.convert_to_tensor(correct_examples)
labels = tf.convert_to_tensor(correct_labels, dtype=tf.int32) # Cast labels to int32

# Create a Foolbox model for TensorFlow
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))

# Apply an FGSM attack
attack = fb.attacks.LinfAdamPGD()
epsilons = [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
raw_advs, clipped_advs, success = attack(fmodel, images, labels, epsilons=epsilons)

# Assuming 'clipped_advs' are the adversarial examples for different epsilons
for eps, advs_ in zip(epsilons, clipped_advs):
    # Predict the labels of the adversarial examples
    y_adv = np.argmax(model.predict(advs_), axis=1)

    # Calculate accuracy on adversarial examples
    accuracy_adv = np.mean(y_adv == correct_labels)
    print(f"\nAdversarial 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
    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
    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)

    # Save the plot
    image_filename = f'confusion_matrix_eps_{eps}.png'
    plt.savefig(image_filename, bbox_inches='tight')
    plt.show() # Display the figure in the notebook

```

308/308 [=====] - 1s 2ms/step

Accuracy on adversarial examples: 99.26%

True Labels \ Predicted Labels	0	1	2	3	4	5	6	7	8	9
0	970	0	0	0	0	1	0	0	2	0
1	0	1132	0	0	0	0	1	0	0	0
2	0	3	1008	1	0	0	0	2	1	1
3	0	1	1	985	0	1	0	0	0	1
4	0	0	2	0	961	0	2	0	0	4
5	0	0	1	1	0	880	0	0	0	0
6	0	0	0	1	2	1	933	0	0	0
7	0	3	3	0	0	1	0	995	0	3
8	0	3	4	1	4	2	2	1	923	6
9	0	1	0	1	2	0	0	6	0	974

Accuracy on adversarial examples: 97.97%

	0	1	2	3	4	5	6	7	8	9
0	968	0	1	0	1	2	0	0	1	0
1	0	1130	0	0	0	1	1	0	1	0
2	1	8	992	4	1	1	1	4	3	1
3	0	1	2	976	0	5	0	3	1	1
4	0	3	3	0	947	0	3	0	2	11
5	1	2	1	1	1	873	2	0	1	0
6	0	1	1	1	8	4	920	1	1	0
7	0	9	6	2	0	2	0	975	2	9
8	4	6	10	7	6	5	4	1	897	6
9	1	4	1	1	5	1	0	11	4	956

Accuracy on adversarial examples: 95.66%

[illegible]

True Labels	4	1	6	4	0	932	0	4	1	3	18
	5	2	3	1	8	1	860	4	0	1	2
	6	3	4	1	1	10	13	903	1	1	0
	7	0	15	12	2	4	2	0	950	2	18
	8	6	14	23	13	9	19	6	3	838	15
	9	4	4	2	5	12	11	0	26	6	914
		Predicted Labels									
		0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.04

Accuracy on adversarial examples: 91.86%

Confusion Matrix for eps=0.04

True Labels	0	953	0	4	0	2	6	3	0	2	3
	1	0	1106	5	3	2	2	8	1	5	1
	2	7	30	936	14	6	1	2	14	4	2
	3	0	4	19	922	0	25	1	5	4	9
	4	2	13	6	0	899	1	5	6	4	33
	5	3	5	1	19	1	835	9	0	4	5
	6	6	5	4	1	13	28	875	1	4	0
	7	0	23	25	7	8	3	0	905	2	32
	8	7	24	43	25	11	39	6	7	760	24
	9	5	6	3	13	42	18	0	43	11	843
		Predicted Labels									
		0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.05

Accuracy on adversarial examples: 86.14%

Confusion Matrix for eps=0.05

True Labels	0	935	0	6	1	2	12	8	1	4	4
	1	1	1090	9	3	2	2	9	3	10	4
	2	9	46	883	30	8	1	3	24	10	2
	3	0	5	28	881	0	41	1	11	7	15
	4	3	23	10	0	852	1	6	8	4	62
	5	5	6	1	38	1	790	14	0	12	15
	6	12	9	7	2	21	52	827	1	5	1
	7	0	35	46	10	12	3	1	834	3	61
	8	7	42	70	56	18	60	10	14	632	37
	9	6	10	5	20	73	27	0	76	20	747
		Predicted Labels									
		0	1	2	3	4	5	6	7	8	9

308/308 [=====] - 1s 3ms/step

Adversarial test data: eps:0.1  
Accuracy on adversarial examples: 27.52%

Confusion Matrix for eps=0.1

True Labels	0	1	2	3	4	5	6	7	8	9
	543	5	141	3	18	112	71	23	13	44
	3	263	299	38	129	9	57	153	156	26
	26	273	351	157	27	1	5	96	72	8
	0	27	131	292	1	322	1	27	78	110
	12	109	41	0	381	9	15	98	26	278
	10	18	2	259	8	289	46	3	141	106
	44	33	54	5	204	296	285	3	12	1
	4	123	210	73	38	10	0	235	4	308
	18	93	290	194	33	167	17	37	19	78
9	6	22	17	114	315	97	1	306	58	48
Predicted Labels										

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.2  
Accuracy on adversarial examples: 0.01%

Confusion Matrix for eps=0.2

True Labels	0	1	2	3	4	5	6	7	8	9
	0	24	347	0	20	208	187	115	6	66
	3	0	496	67	171	16	28	292	56	4
	28	517	1	204	32	5	7	146	70	6
	0	76	192	0	0	506	0	38	70	107
	17	211	66	2	0	26	20	243	25	359
	15	34	5	425	17	0	51	9	177	149
	52	56	104	5	318	384	0	6	11	1
	4	258	313	88	49	33	0	0	3	257
	9	124	355	173	24	187	9	44	0	21
9	7	35	33	126	294	148	1	318	22	0
Predicted Labels										

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.3  
Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.3

True Labels	0	1	2	3	4	5	6	7	8	9
	0	44	418	2	14	229	99	143	3	21
	1	0	646	70	85	45	4	270	13	0
	19	595	0	192	30	6	3	129	40	2
	0	128	225	0	0	554	0	34	23	25
	7	298	127	3	0	64	18	255	2	195
5	14	69	8	489	25	0	37	18	148	74

	0	1	2	3	4	5	6	7	8	9
6	26	90	148	3	269	387	0	12	2	0
7	3	328	354	77	48	57	0	0	1	137
8	4	145	394	137	15	194	4	48	0	5
9	5	60	56	109	253	190	1	303	7	0
	Predicted Labels									

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.4

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.4

0	0	75	446	2	10	266	41	124	3	6
1	0	0	760	104	23	56	1	188	1	0
2	15	663	0	184	26	18	4	90	14	2
3	0	181	228	0	1	547	0	27	5	0
4	7	374	162	1	0	144	9	190	4	78
5	9	115	27	510	18	0	29	25	119	30
6	20	129	219	5	185	370	0	6	3	0
7	3	411	376	62	27	94	0	0	0	32
8	2	175	449	101	10	174	0	35	0	0
9	2	97	100	93	176	242	0	271	3	0
	0	1	2	3	4	5	6	7	8	9
True Labels										

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.5

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.5

0	0	87	492	1	3	285	13	87	1	4
1	0	0	810	118	12	67	0	126	0	0
2	8	722	0	158	19	38	1	66	4	0
3	0	221	254	0	0	490	0	20	4	0
4	0	399	243	4	0	174	7	116	0	26
5	7	180	77	476	17	0	20	32	66	7
6	7	160	298	2	91	371	0	8	0	0
7	4	420	369	55	14	134	0	0	0	9
8	1	175	482	73	4	182	0	29	0	0
9	2	128	189	76	105	276	0	208	0	0
	0	1	2	3	4	5	6	7	8	9
True Labels										

308/308 [=====] - 1s 2ms/step

Adversarial test data: eps:0.6

Accuracy on adversarial examples: 0.00%

Confusion Matrix for eps=0.6



