Problem 1

In [2]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import pandas as pd
import matplotlib.pyplot as plt
#initialization code required to make tensorflow work on my systemabs
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.compat.v1.Session(config=config)
#disabling eager execution
tf.compat.v1.disable_eager_execution()
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
print("Tensorflow version: ",tf.__version__)
```

```
Num GPUs Available:  1
Tensorflow version:  2.4.0
```

In [10]:
```
from art.utils import load_mnist
from art.estimators.classification import KerasClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.optimizers import Adam
from art.attacks.evasion import FastGradientMethod
from art.attacks.evasion import CarliniL2Method
from utils import preprocess_mnist
```

As per recommendation on piazza: defining the same architecture as original trained model but with linear activation instead of softmax

In [4]:
```
#define
model = Sequential()
model.add(Conv2D(filters=4, kernel_size=(5, 5), strides=1, activation="relu", input_sha
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=10, kernel_size=(5, 5), strides=1, activation="relu", input_sh
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(100, activation="relu"))
model.add(Dense(10, activation="linear"))
#compile: same as original
model.compile(loss=categorical_crossentropy, optimizer=Adam(learning_rate=0.01), metric
```

In [5]:
```
#Load trained weights from original model
model.load_weights(r'mnist_model_weights.h5')
```

In [6]:
```
#save full model for use with Carlini Attacks: no softmax layer only linear with traine
model.save(r'model')
```

```
INFO:tensorflow:Assets written to: model\assets
```

In [11]:
```
#apply ART wrapper: min and max pixel value were 0 and 1 when loaded in using load_mnis
classifier = KerasClassifier(model=model, clip_values=(0.0, 1.0), use_logits=True)
```

Load and preprocessed benign test samples

In [13]:
```
x_attack_test = np.load(r'x_attack_test.npy')
y_attack_test = np.load(r'y_attack_test.npy')
```

In [14]:
```
#define some constants
num_test_samples = x_attack_test.shape[0]
#define target class
t = 7
```

In [15]:
```
num_test_samples
```

Out[15]:
```
100
```

In [16]:
```
#print out accuracy on benign samples
predictions = classifier.predict(x_attack_test)
accuracy = np.sum(np.argmax(predictions, axis=1) == np.argmax(y_attack_test, axis=1)) /
print(f"Accuracy on benign test examples: {accuracy*100}%")
```

```
C:\Users\apra\Anaconda3\envs\cy_hw2\lib\site-packages\tensorflow\python\keras\engine\tra
ining.py:2325: UserWarning: `Model.state_updates` will be removed in a future version. T
his property should not be used in TensorFlow 2.0, as `updates` are applied automaticall
y.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
Accuracy on benign test examples: 100.0%
```

In [17]:
```
#print number of target class predictions on benign samples (should be close to 0)
num_targ_class = np.sum(np.argmax(predictions, axis=1) == t) / num_test_samples
```

In [18]:
```
print(f"Percentage predicted to be target class ({t}) on benign test examples: {num_tar
```

```
Percentage predicted to be target class (7) on benign test examples: 0.0%
```

In [19]:
```
x_attack_test.shape
```

Out[19]:
```
(100, 28, 28, 1)
```

In [20]:
```
y_attack_test.shape
```

Out[20]:
```
(100, 10)
```

1a: FSGM attack : we use the softmax model

TARGETTED ATTACKS

In [24]:
```
target_class_vector = t*np.ones(num_test_samples)
```

In [15]:
```
epsilons = [0.1,0.2,0.3,0.4,0.5]
```

In [16]:
```
softmax_model = keras.models.load_model(r'softmax_model')
softmax_classifier = KerasClassifier(model=softmax_model, clip_values=(0.0, 1.0), use_l
```

In [17]:
```
#store best attacks in a list of dictionaries
targeted_fsgm_best_attacks = []
for e in epsilons:
    fsgm_targeted = FastGradientMethod(estimator=softmax_classifier, eps=e,targeted=Tru
    #norm is l_infinity by default
    x_test_fsgm_targeted = fsgm_targeted.generate(x=x_attack_test,y=target_class_vector
    fsgm_targeted_predictions = softmax_classifier.predict(x_test_fsgm_targeted)
    #in the targeted case, attack success rate is simply the percent of test samples cl
    #as target class (since this class is not present in the test set)
    success_mask = np.argmax(fsgm_targeted_predictions, axis=1) == t
    attack_success_rate = np.sum(success_mask) / num_test_samples
    print(f"Attack Success Rate on targeted fsgm attack examples for epsilon = {e} : {r



    #identify and store adversarial inputs for each epsilon with highest target classif
    confidences = pd.Series(fsgm_targeted_predictions[:,7],index=range(num_test_samples

    #get highest successful attacks with highest confidence
    conf_idx = pd.Series.idxmax(confidences[success_mask])

    best_attacks = {'e':e, 'original_image':x_attack_test[conf_idx],
                    'attack_image':x_test_fsgm_targeted[conf_idx],
                    'test_idx' : conf_idx,
                    'perturbation' : (x_attack_test[conf_idx] - x_test_fsgm_targeted[co
    targeted_fsgm_best_attacks.append(best_attacks)
```

```
Attack Success Rate on targeted fsgm attack examples for epsilon = 0.1 : 8.0%
Attack Success Rate on targeted fsgm attack examples for epsilon = 0.2 : 14.0%
Attack Success Rate on targeted fsgm attack examples for epsilon = 0.3 : 24.0%
Attack Success Rate on targeted fsgm attack examples for epsilon = 0.4 : 27.0%
Attack Success Rate on targeted fsgm attack examples for epsilon = 0.5 : 26.0%
```

UNTARGETTED ATTACKS

In [18]:
```
for e in epsilons:
    fsgm_untargeted = FastGradientMethod(estimator=softmax_classifier, eps=e)
    #norm is l_infinity by default
    x_test_fsgm_untargeted = fsgm_untargeted.generate(x=x_attack_test)
    predictions = softmax_classifier.predict(x_test_fsgm_untargeted)
    accuracy_mask = np.argmax(predictions, axis=1) == np.argmax(y_attack_test, axis=1)

    #in untargeted case, attack success rate is the DROP in accuracy i.e, benign_accura
    #this accurately tells us how many previously correctly classified samples were mis
    attack_success_rate = accuracy - np.sum(accuracy_mask) / num_test_samples
    print(f"Attack Success Rate on untargeted fsgm attack examples for epsilon = {e} :
```

```
Attack Success Rate on untargeted fsgm attack examples for epsilon = 0.1 : 34.0%
Attack Success Rate on untargeted fsgm attack examples for epsilon = 0.2 : 53.0%
Attack Success Rate on untargeted fsgm attack examples for epsilon = 0.3 : 63.0%
Attack Success Rate on untargeted fsgm attack examples for epsilon = 0.4 : 69.0%
Attack Success Rate on untargeted fsgm attack examples for epsilon = 0.5 : 75.0%
```

1b Targeted Carlini Wagner in L2: FIXED C

In [65]:
```
constants = [0.5,1,5,10]
```

In [66]:
```
#store attack metrics for various c's in a list of dictionaries
l2_attack_metrics = []
for c in constants:
    #binary search steps is 0 so we can fix the constant c
    carlini = CarliniL2Method(classifier=classifier,targeted=True,initial_const=c,batch
    carlini_attacks = carlini.generate(x_attack_test,target_class_vector)

    #predictions
    carlini_predict = model.predict(carlini_attacks)

    #calculate success rate
    success_mask = np.argmax(carlini_predict, axis=1) == t
    attack_success_rate = sum(success_mask)/num_test_samples

    #calculate l2 norm
    l2dist = np.sqrt(np.sum(np.square(x_attack_test - carlini_attacks).reshape(num_test
    #create series out of l2 dist to pull out min,max,median
    l2dist = pd.Series(l2dist,index=range(num_test_samples))

    #Report attack success rate and avg l2 norm
    print(f"Attack Success Rate on targeted Carlini Wagner attack examples for c = {c}
    print(f"Avg L2 Norm on targeted Carlini Wagner attack examples for c = {c} : {round


    min_l2_idx = pd.Series.idxmin(l2dist[success_mask])
    max_l2_idx = pd.Series.idxmax(l2dist[success_mask])
    median_l2_idx = int(l2dist[success_mask].sort_values(ignore_index=False).reset_inde

    #append the l2 metrics
    l2_attack_metrics.append(
    {
        'c':c,
        'min_l2_idx':min_l2_idx,
        'max_l2_idx':max_l2_idx,
        'median_l2_idx':median_l2_idx,

        'min_l2':l2dist[min_l2_idx],
        'max_l2':l2dist[max_l2_idx],
        'median_l2':l2dist[median_l2_idx],

        'original_image_min_l2':x_attack_test[min_l2_idx],
        'original_image_max_l2':x_attack_test[max_l2_idx],
        'original_image_median_l2':x_attack_test[median_l2_idx],

        'attack_image_min_l2':carlini_attacks[min_l2_idx],
        'attack_image_max_l2':carlini_attacks[max_l2_idx],
        'attack_image_median_l2':carlini_attacks[median_l2_idx],
```

```
        'perturbation_min_l2':x_attack_test[min_l2_idx]- carlini_attacks[min_l2_idx],
        'perturbation_max_l2':x_attack_test[max_l2_idx] - carlini_attacks[max_l2_idx],
        'perturbation_median_l2':x_attack_test[median_l2_idx] - carlini_attacks[median_
    })
```

```
Attack Success Rate on targeted Carlini Wagner attack examples for c = 0.5 : 14.0%
Avg L2 Norm on targeted Carlini Wagner attack examples for c = 0.5 : 0.22

Attack Success Rate on targeted Carlini Wagner attack examples for c = 1 : 25.0%
Avg L2 Norm on targeted Carlini Wagner attack examples for c = 1 : 0.52

Attack Success Rate on targeted Carlini Wagner attack examples for c = 5 : 60.0%
Avg L2 Norm on targeted Carlini Wagner attack examples for c = 5 : 2.25

Attack Success Rate on targeted Carlini Wagner attack examples for c = 10 : 71.0%
Avg L2 Norm on targeted Carlini Wagner attack examples for c = 10 : 3.14
```

Observations:

1. As we increase c, the attack success rate seems to increase steadily.
2. As we increase c, the avg l2 perturbations also seems to increase. We can also observe that the approximately linear. Doubling c doubles avg l2 and 5x'ing c 5'xs the avg l2 perturbation.

1c Vizualizations

(Successful) Adversarial samples with highest confidence for targeted FSGM

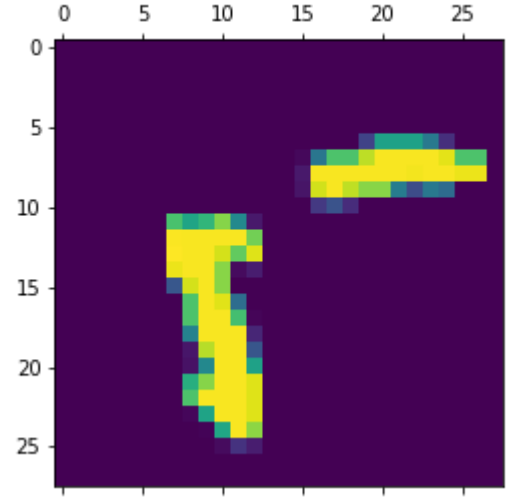
We extracted this information in our attack loop

In [54]:

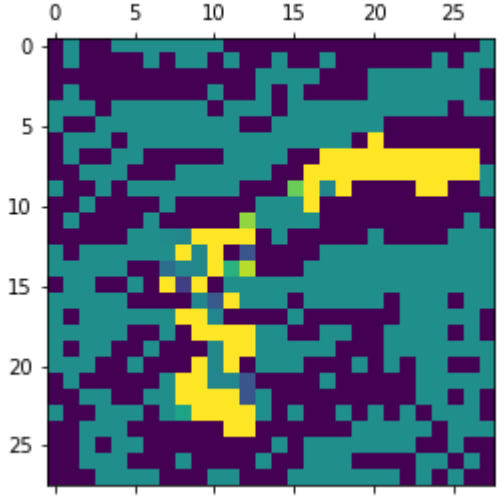
```
for e_idx,e in enumerate(epsilons):
    print(f"==========EPSILON = {e}==============")
    viz_type = "original_image"
    plt.matshow(targeted_fsgm_best_attacks[e_idx][viz_type])
    plt.title(f"{viz_type} for targeted FSGM attack at epsilon = {epsilons[e_idx]}")
    plt.show()
    viz_type = "perturbation"
    plt.matshow(targeted_fsgm_best_attacks[e_idx][viz_type])
    plt.title(f"{viz_type} for targeted FSGM attack at epsilon = {epsilons[e_idx]}")
    plt.show()
    viz_type = "attack_image"
    plt.matshow(targeted_fsgm_best_attacks[e_idx][viz_type])
    plt.title(f"{viz_type} for targeted FSGM attack at epsilon = {epsilons[e_idx]}")
    plt.show()
    print()
```

```
==========EPSILON = 0.1==============
```

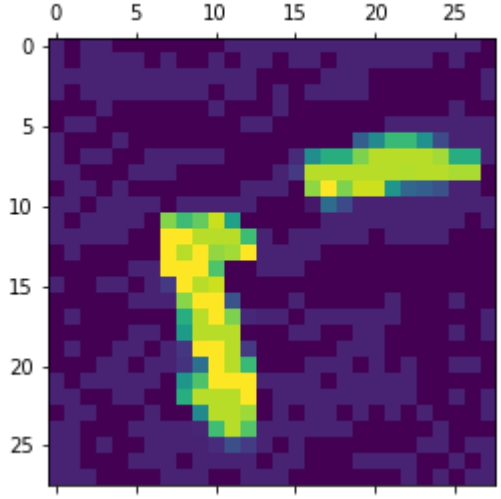
original_image for targeted FSGM attack at epsilon = 0.1



perturbation for targeted FSGM attack at epsilon = 0.1

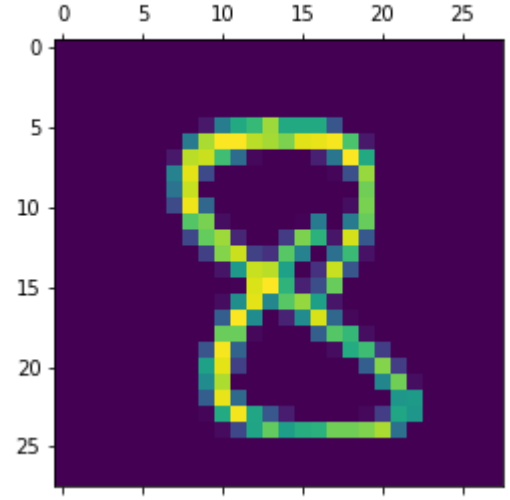


attack_image for targeted FSGM attack at epsilon = 0.1

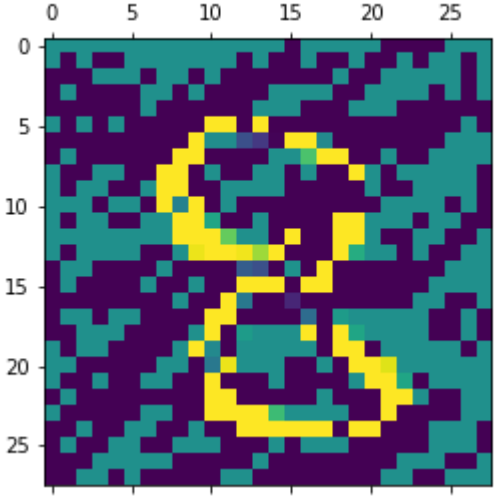


==========EPSILON = 0.2==============

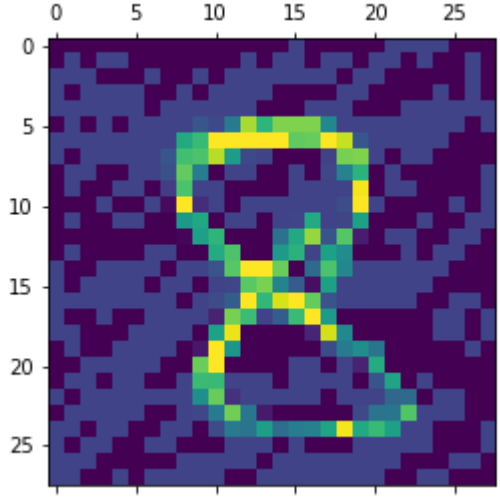
original_image for targeted FSGM attack at epsilon = 0.2



perturbation for targeted FSGM attack at epsilon = 0.2

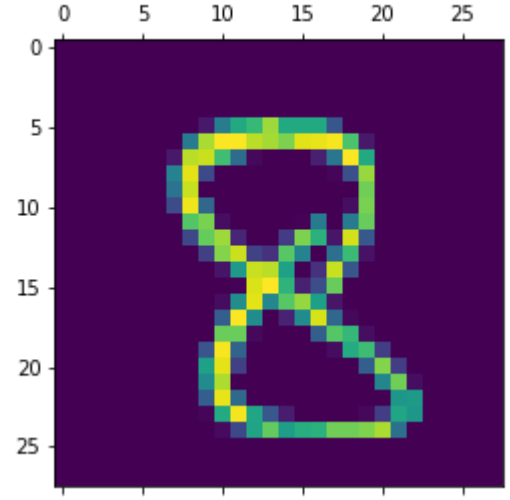


attack_image for targeted FSGM attack at epsilon = 0.2

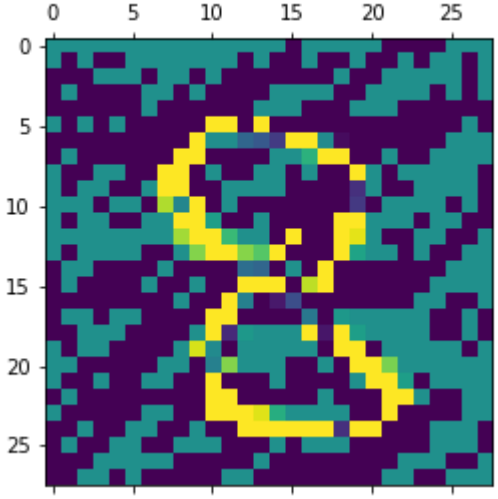


==========EPSILON = 0.3==============

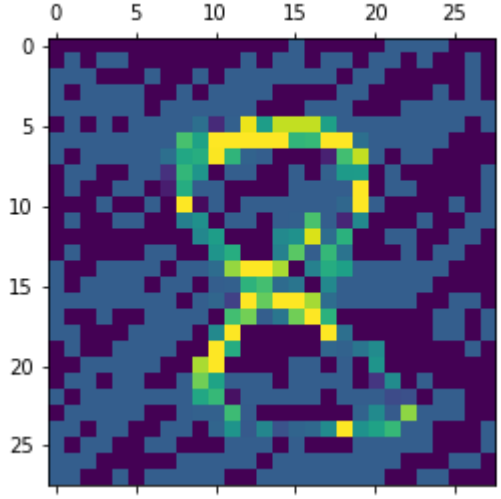
original_image for targeted FSGM attack at epsilon = 0.3



perturbation for targeted FSGM attack at epsilon = 0.3

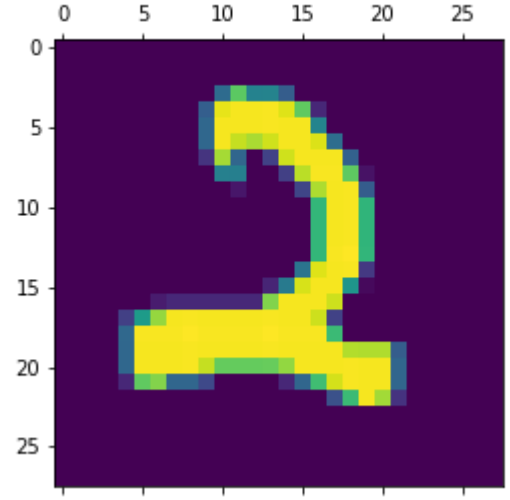


attack_image for targeted FSGM attack at epsilon = 0.3

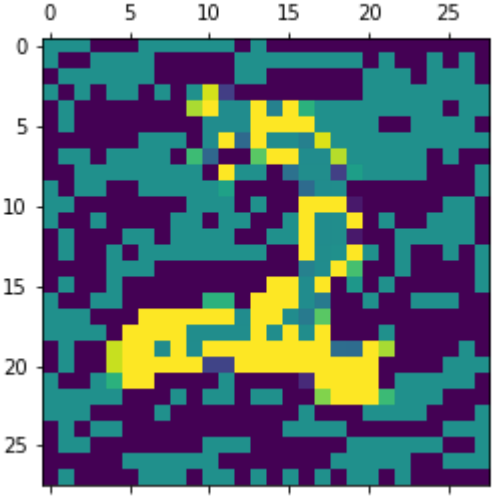



```
==========EPSILON = 0.4==============
```

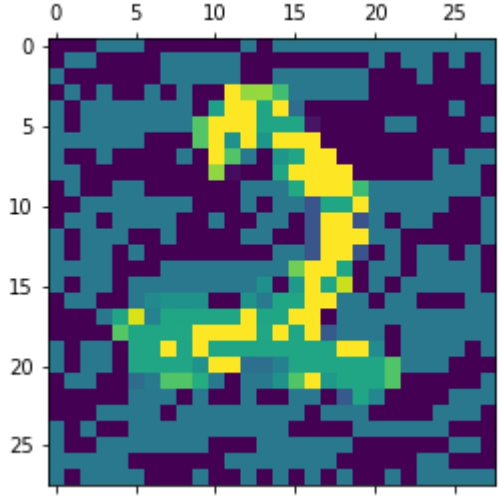
original_image for targeted FSGM attack at epsilon = 0.4



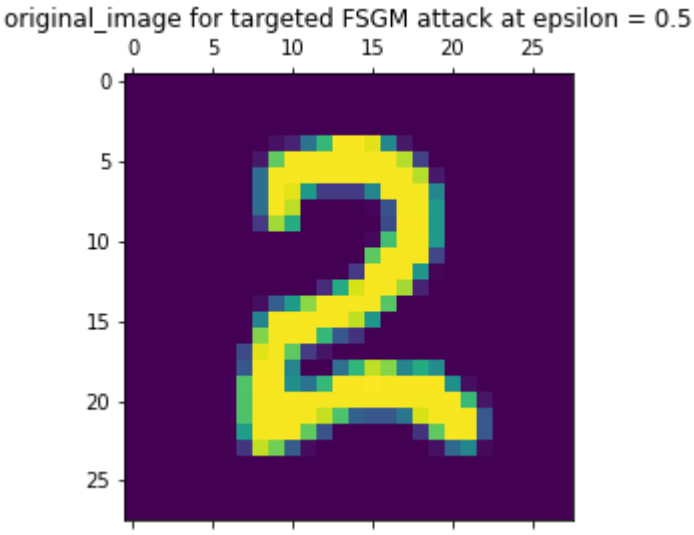
perturbation for targeted FSGM attack at epsilon = 0.4

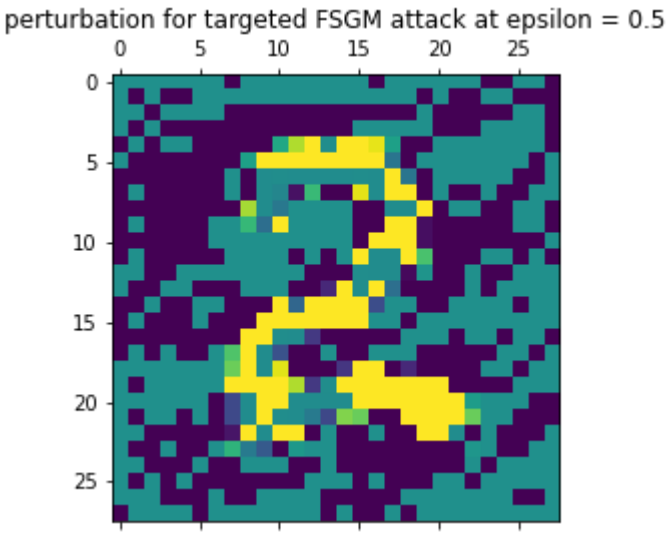


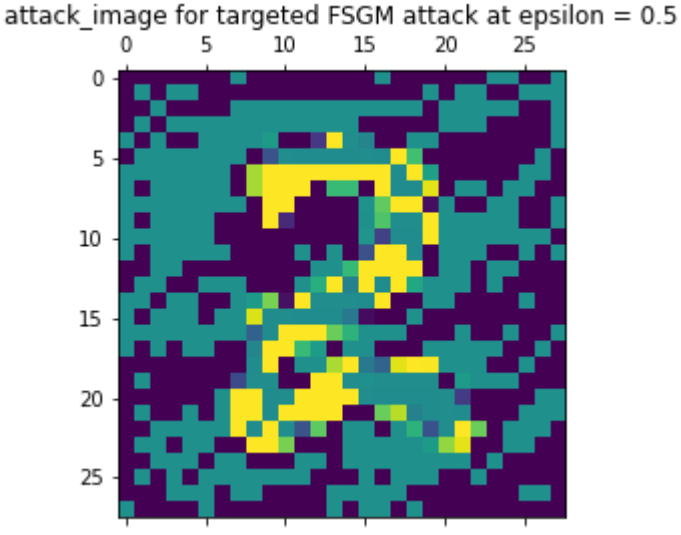
attack_image for targeted FSGM attack at epsilon = 0.4



==========EPSILON = 0.5==============







Min,Max and Median l2 distance images for various values of c: Successful Carlini Wagner attacks

In [67]:
```
l2_attack_metrics_df = pd.DataFrame(l2_attack_metrics)
```

In [68]:
```
l2_attack_metrics_df[['c','min_l2_idx','min_l2','max_l2_idx','max_l2','median_l2_idx','
```

Out[68]:

	c	min_l2_idx	min_l2	max_l2_idx	max_l2	median_l2_idx	median_l2
0	0.5	55	1.170726	73	2.606154	26	1.584509
1	1.0	74	1.261963	98	3.745318	68	2.126723
2	5.0	55	1.706915	62	7.120429	90	3.563080
3	10.0	26	1.858502	18	7.124496	29	4.463604

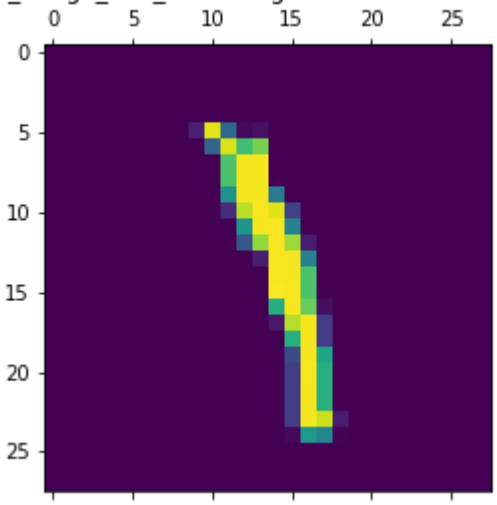
C = 0.5

In [73]:
```
c_i,c = 0,0.5
print(f"===========MIN L2 ================")
l2_type = "min_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

print(f"===========MAX L2 ================")
l2_type = "max_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

print(f"===========MEDIAN L2 ================")
l2_type = "median_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
```

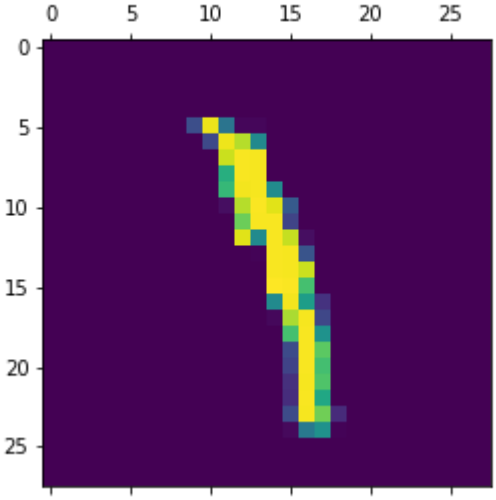
```
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()
```

```
==========MIN L2 ================
```

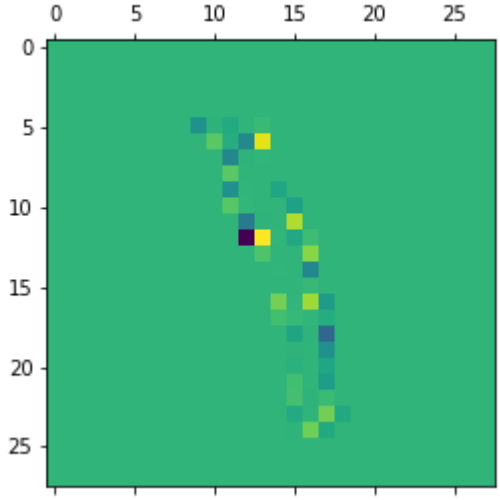
original_image_min_l2 for targeted C&W attack at c = 0.5



attack_image_min_l2 for targeted C&W attack at c = 0.5

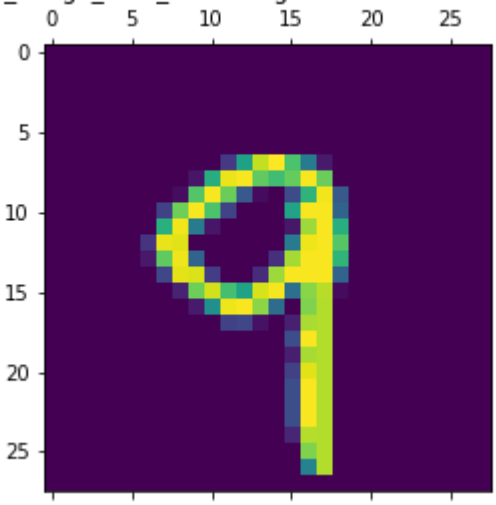


perturbation_min_l2 for targeted C&W attack at c = 0.5

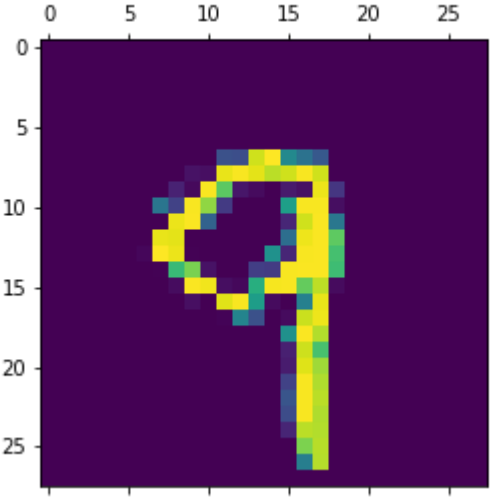


==========MAX L2 ================

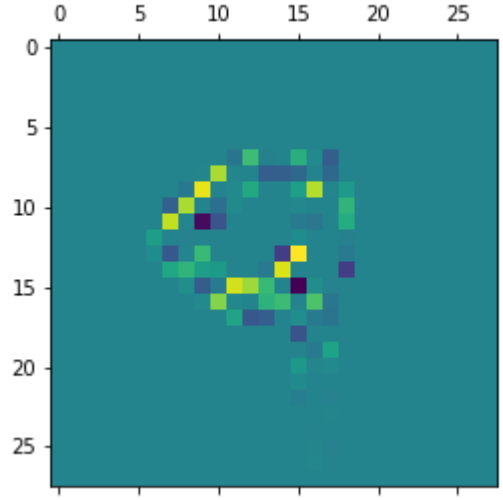
original_image_max_l2 for targeted C&W attack at c = 0.5



attack_image_max_l2 for targeted C&W attack at c = 0.5

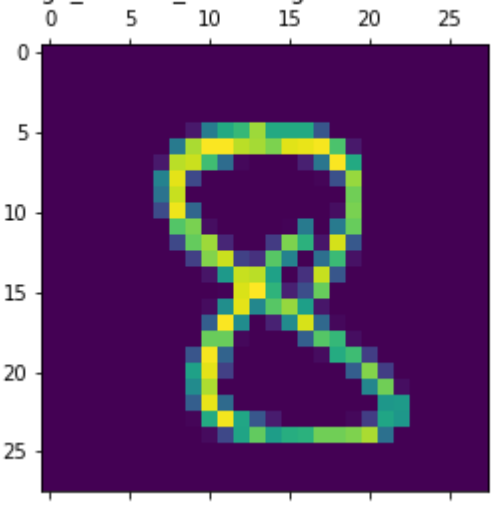


perturbation_max_l2 for targeted C&W attack at c = 0.5

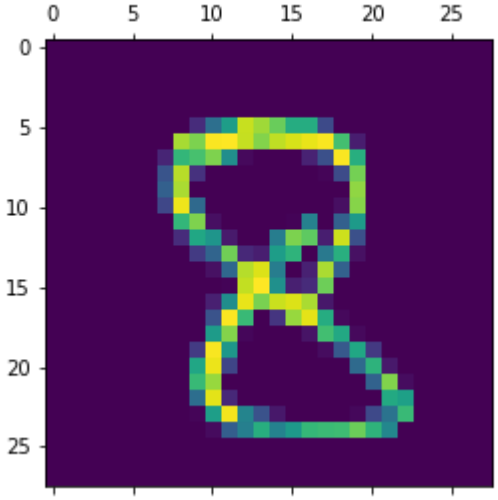


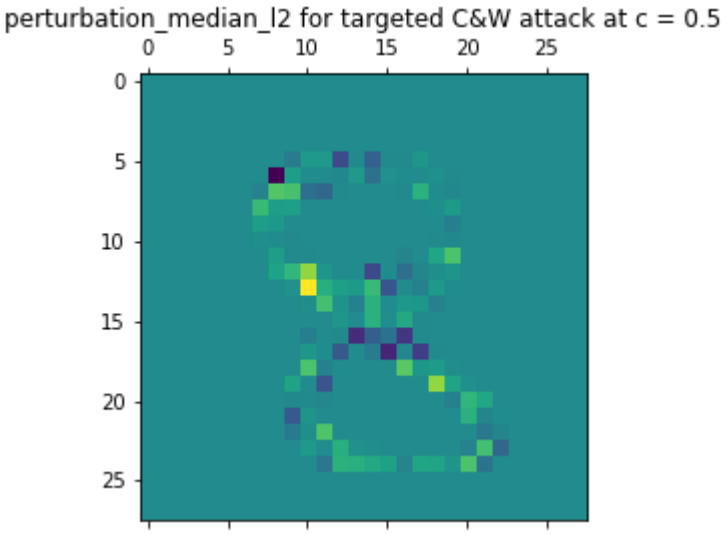
==========MEDIAN L2 ================

original_image_median_l2 for targeted C&W attack at c = 0.5



attack_image_median_l2 for targeted C&W attack at c = 0.5





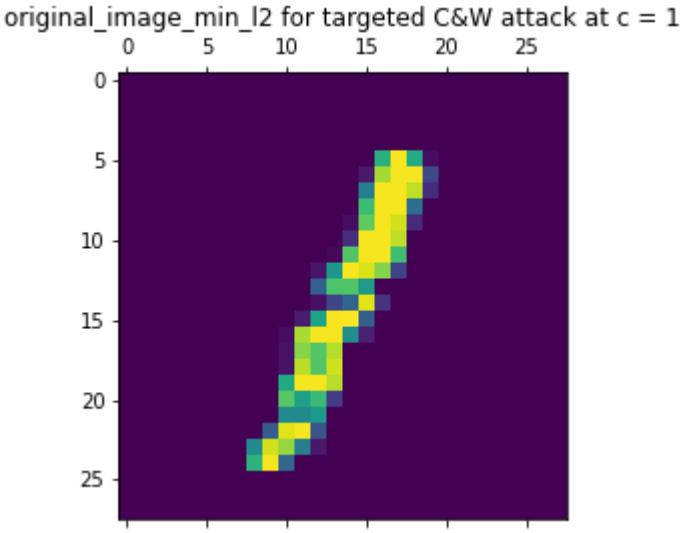
C = 1

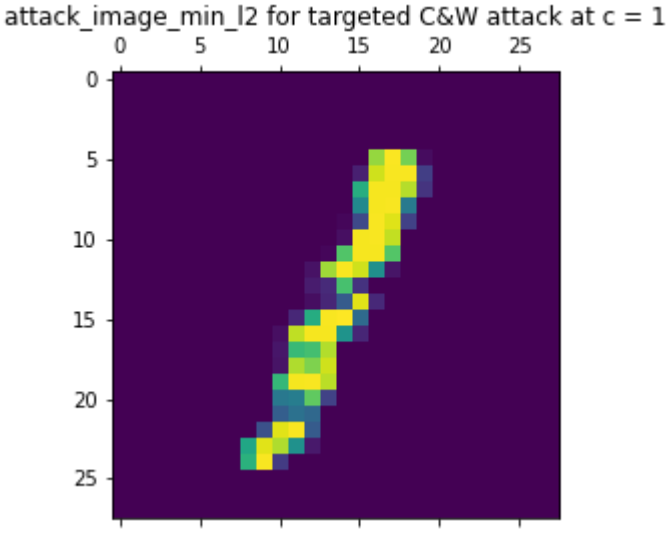
In [74]:

```
c_i,c = 1,1
print(f"===========MIN L2 ================")
l2_type = "min_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

print(f"===========MAX L2 ================")
l2_type = "max_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
```

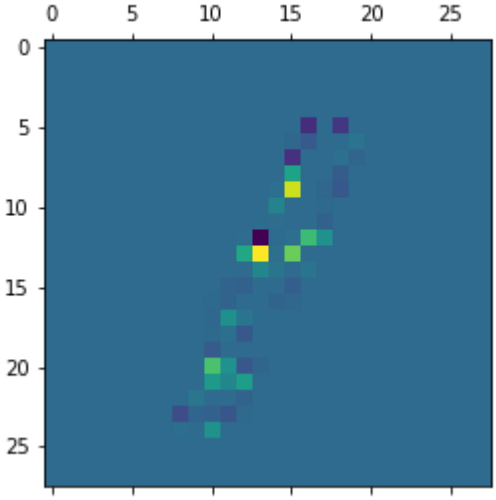
```
plt.show()

print(f"===========MEDIAN L2 ================")
l2_type = "median_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()
```

```
===========MIN L2 ================
```



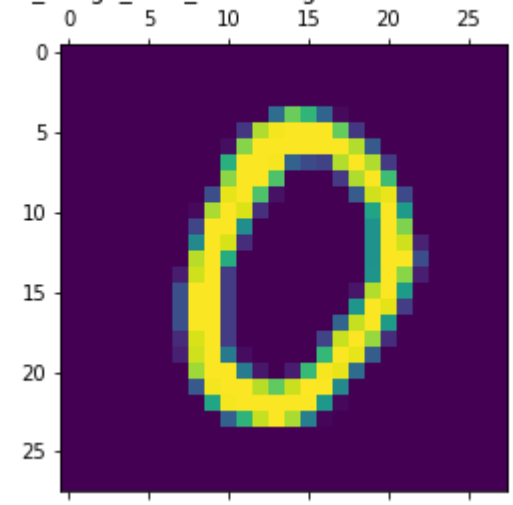


perturbation_min_l2 for targeted C&W attack at c = 1

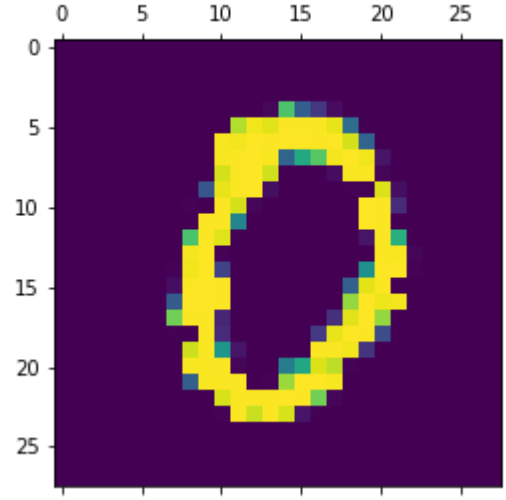


==========MAX L2 ================

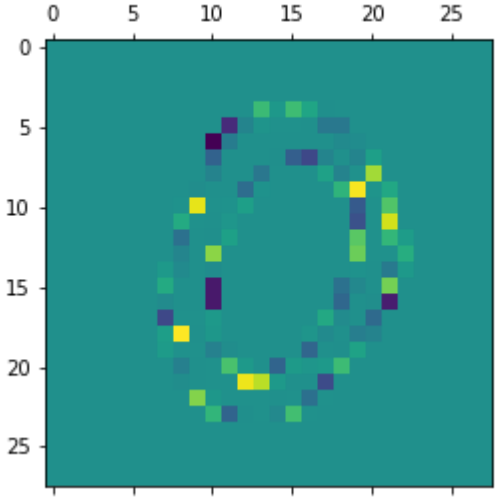
original_image_max_l2 for targeted C&W attack at c = 1



attack_image_max_l2 for targeted C&W attack at c = 1

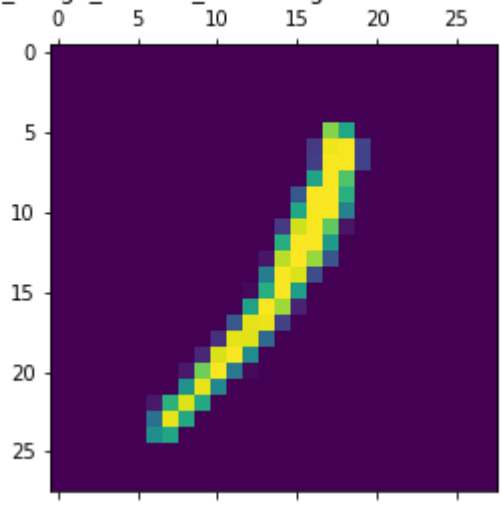


perturbation_max_l2 for targeted C&W attack at c = 1

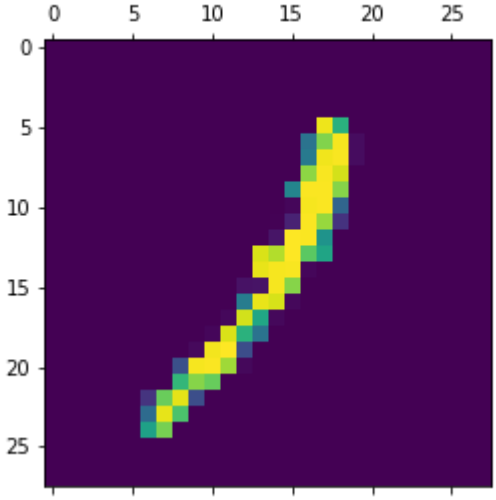


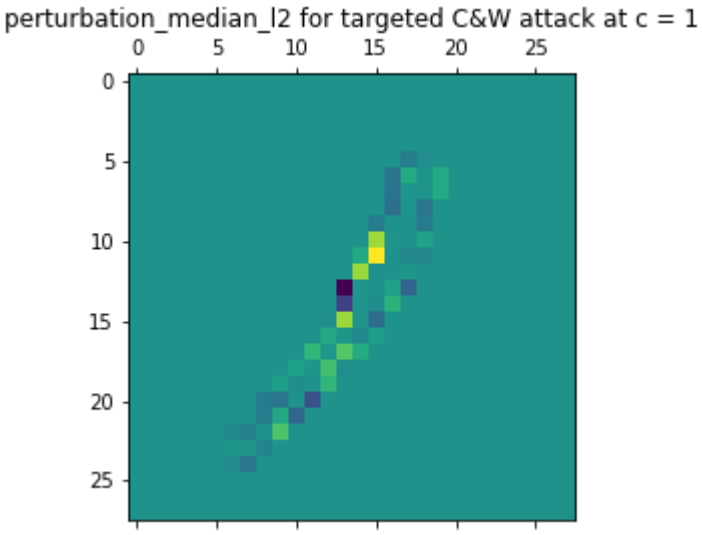
==========MEDIAN L2 ================

original_image_median_l2 for targeted C&W attack at c = 1



attack_image_median_l2 for targeted C&W attack at c = 1





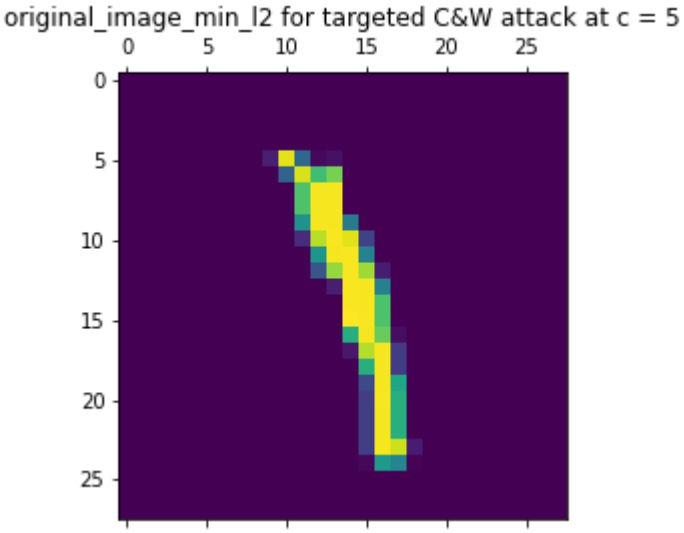
C = 5

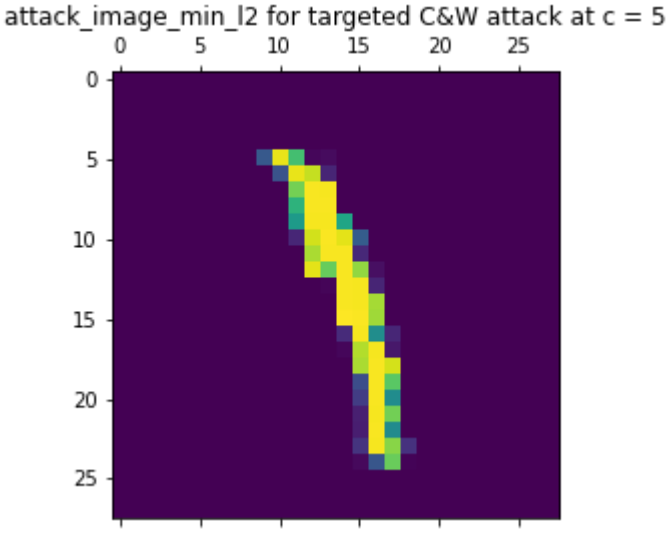
In [75]:

```
c_i,c = 2,5
print(f"===========MIN L2 ================")
l2_type = "min_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

print(f"===========MAX L2 ================")
l2_type = "max_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
```

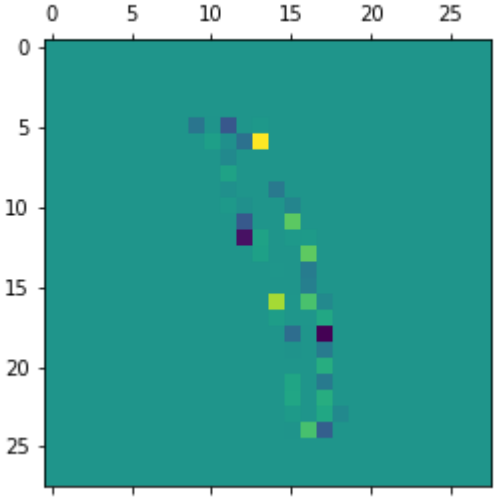
```
plt.show()

print(f"===========MEDIAN L2 ================")
l2_type = "median_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()
```

```
===========MIN L2 ================
```



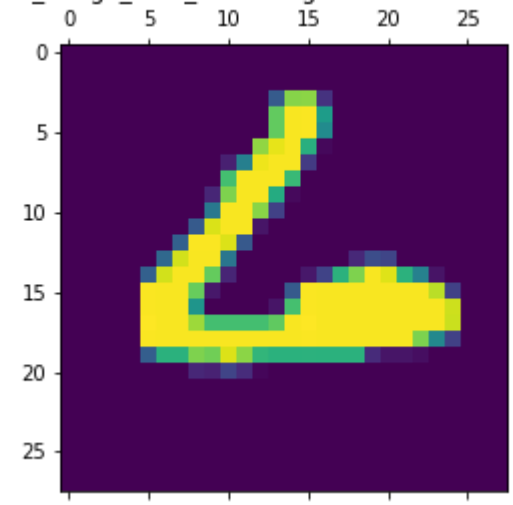


perturbation_min_l2 for targeted C&W attack at c = 5

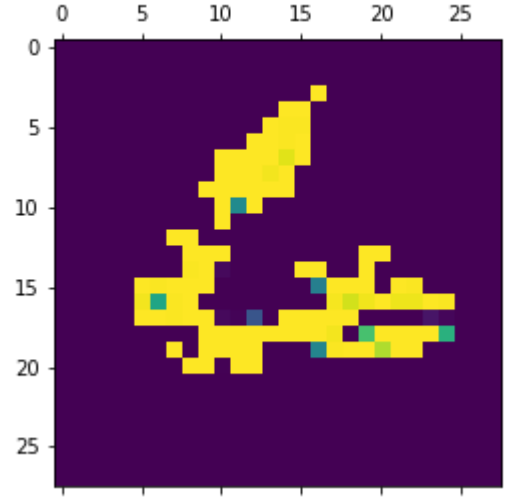


==========MAX L2 ================

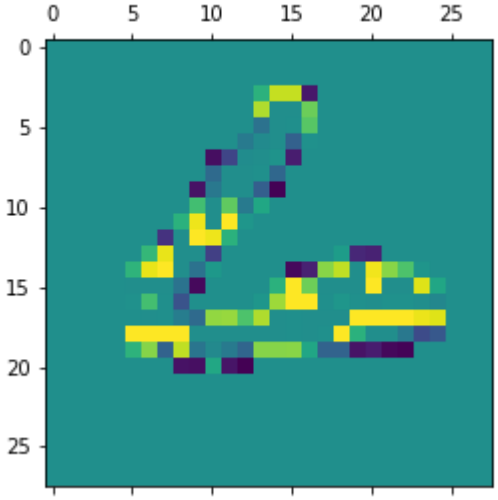
original_image_max_l2 for targeted C&W attack at c = 5



attack_image_max_l2 for targeted C&W attack at c = 5

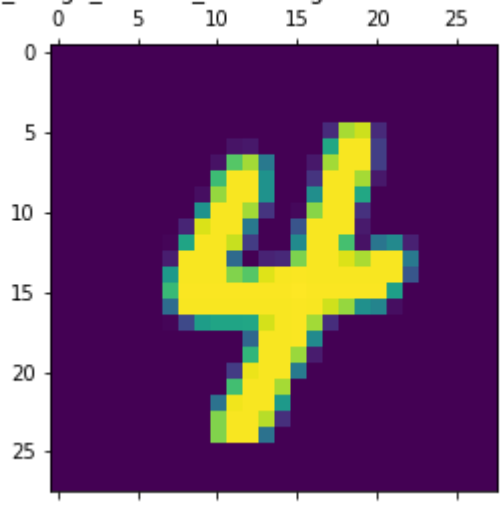


perturbation_max_l2 for targeted C&W attack at c = 5

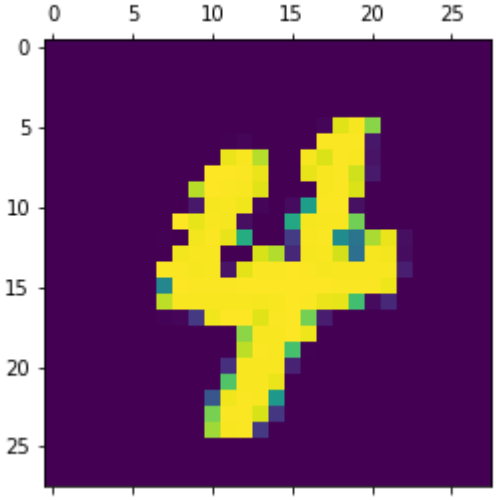


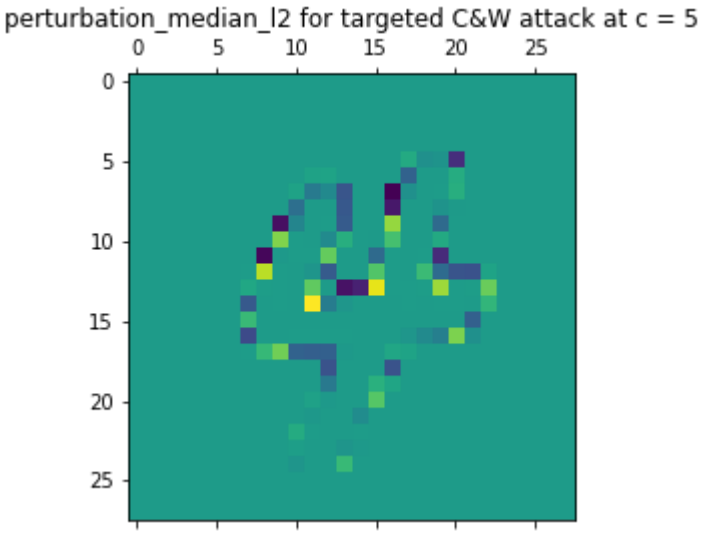
==========MEDIAN L2 ================

original_image_median_l2 for targeted C&W attack at c = 5



attack_image_median_l2 for targeted C&W attack at c = 5





C = 10

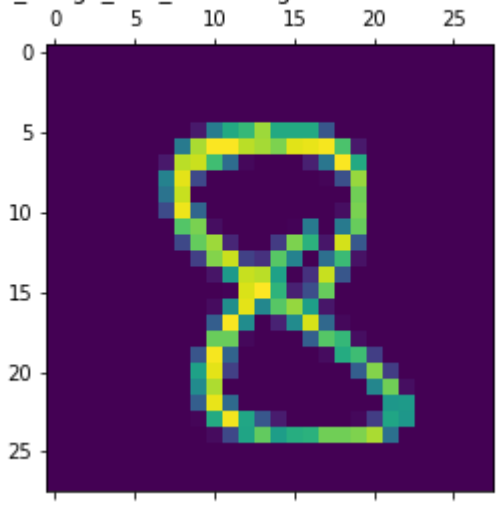
In [76]:

```
c_i,c = 3,10
print(f"===========MIN L2 ================")
l2_type = "min_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

print(f"===========MAX L2 ================")
l2_type = "max_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
```

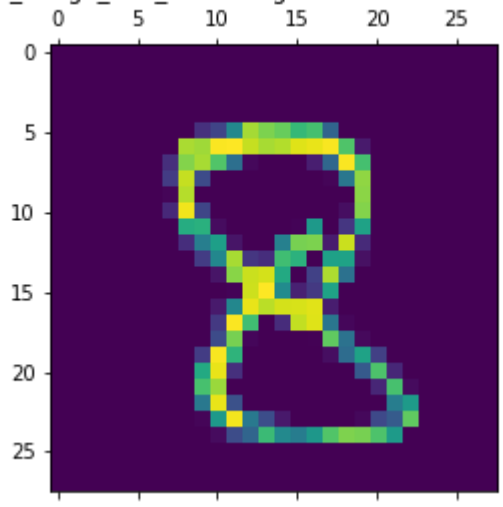
```
plt.show()

print(f"===========MEDIAN L2 ================")
l2_type = "median_l2"
viz_type = "original_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "attack_image_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()

viz_type = "perturbation_"
viz_key = viz_type + l2_type
plt.matshow(l2_attack_metrics[c_i][viz_key])
plt.title(f"{viz_key} for targeted C&W attack at c = {c}")
plt.show()
```

```
===========MIN L2 ================
```

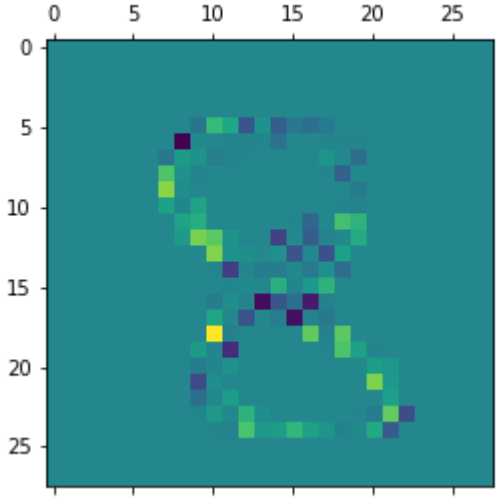
original_image_min_l2 for targeted C&W attack at c = 10



attack_image_min_l2 for targeted C&W attack at c = 10

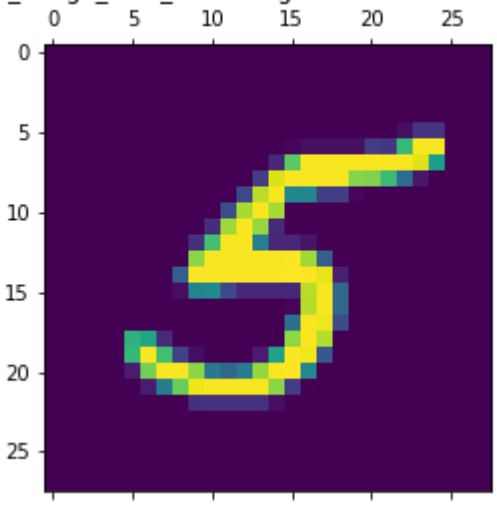


perturbation_min_l2 for targeted C&W attack at c = 10

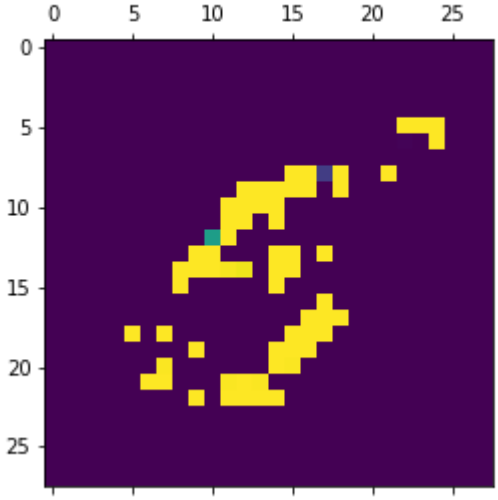


==========MAX L2 ================

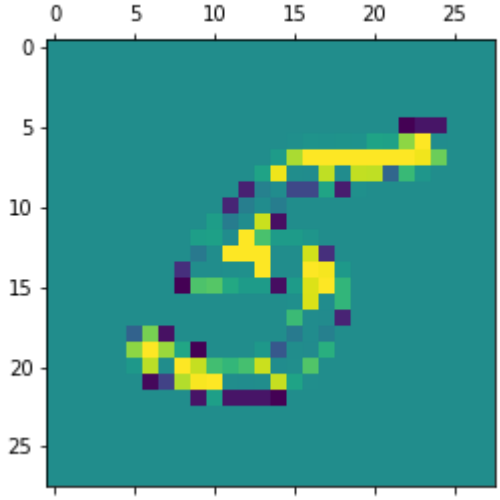
original_image_max_l2 for targeted C&W attack at c = 10



attack_image_max_l2 for targeted C&W attack at c = 10

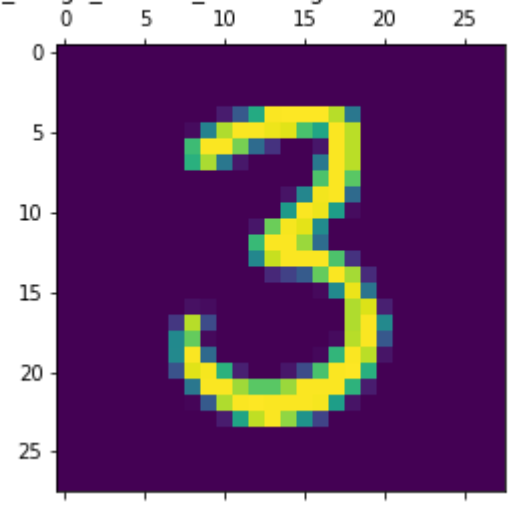


perturbation_max_l2 for targeted C&W attack at c = 10

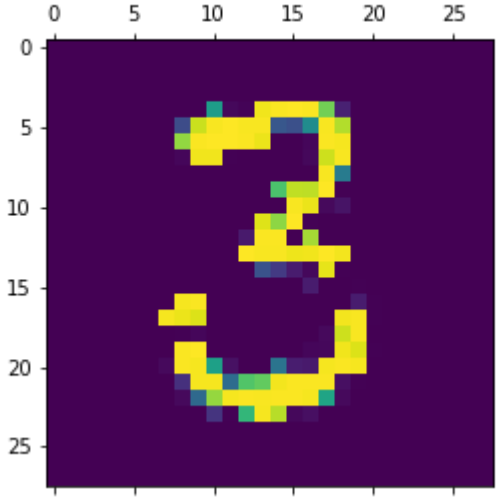


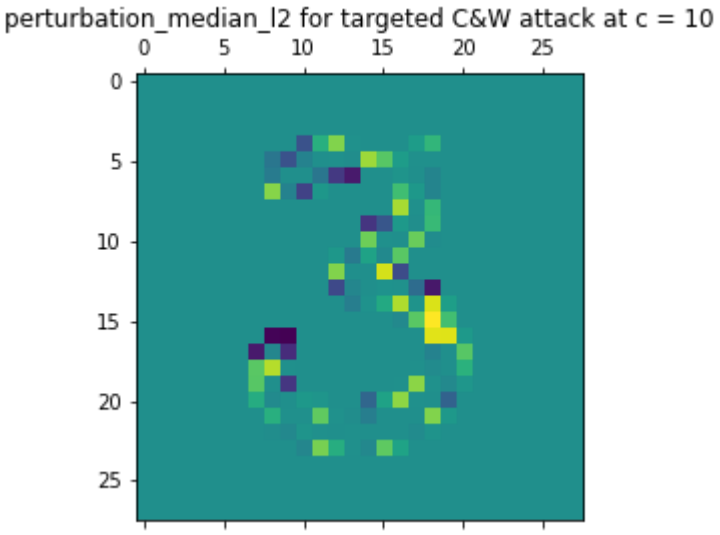
==========MEDIAN L2 ================

original_image_median_l2 for targeted C&W attack at c = 10



attack_image_median_l2 for targeted C&W attack at c = 10





Observations

1. FSGM attacks seem to have much higher perturbation: the difference between original and attacked image is very apparent. Further, the perturbations are not limited to the digit

In [ ]:

In [ ]: