Задание 1

Загружаем данные

Загрузка данных

```
import kagglehub

path = kagglehub.dataset_download("meowmeowmeowmeowmeow/gtsrb-german-traffic-sign")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?dataset_version_number=1...
100%|██████████| 612M/612M [00:28<00:00, 22.2MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign/versions/1
```

```
data_path = path
train_data_path = os.path.join(data_path, 'Train')
test_data_path = os.path.join(data_path, 'Test')
meta_data_path = os.path.join(data_path, 'Meta')
```

Читаем тренировочный набор

Чтение тренировочного набора данных

```
data = []
labels = []
class_count = 43
for i in range(class_count):
    img_path = os.path.join(train_data_path, str(i))
    for img in os.listdir(img_path):
        img = image.load_img(img_path + '/' + img, target_size=(32, 32))
        img_array = image.img_to_array(img)
        img_array = img_array / 255
        data.append(img_array)
        labels.append(i)
data = np.array(data)
labels = np.array(labels)
labels = to_categorical(labels, 43)
print("data[0]:\n",data[0])
```

```
[ ] data[0]:
 [[[0.07058824 0.07843138 0.07450981]
  [0.07450981 0.08235294 0.07843138]
  [0.09019608 0.10196079 0.10980392]
  ...
  [0.08627451 0.09019608 0.09019608]
  [0.07058824 0.07843138 0.07843138]
  [0.06666667 0.07058824 0.07058824]]

 [[0.08235294 0.08235294 0.07450981]
  [0.14901961 0.1254902  0.10980392]
  [0.22352941 0.17254902 0.1764706 ]
  ...
  [0.12156863 0.10980392 0.11372549]
  [0.11764706 0.10980392 0.10980392]
  [0.10196079 0.09411765 0.09019608]]

 [[0.11372549 0.10588235 0.10588235]
  [0.25490198 0.19215687 0.19607843]
  [0.3647059  0.24705882 0.23921569]
  ...
  [0.19607843 0.16470589 0.16470589]
  [0.1882353  0.16078432 0.17254902]
  [0.1764706  0.15686275 0.18039216]]

 ...

 [[0.33333334 0.30980393 0.30588236]
  [0.5568628  0.48235294 0.47058824]
  [0.78039217 0.654902   0.63529414]
  ...
  [0.4627451  0.47843137 0.47843137]
  [0.5647059  0.5058824  0.49019608]
  [0.40784314 0.36862746 0.3372549 ]]

 [[0.33333334 0.30980393 0.29803923]
  [0.5764706  0.49411765 0.47058824]
  [0.80784315 0.6666667  0.6392157 ]
  ...
  [0.5764706  0.5372549  0.5294118 ]
  [0.5764706  0.47843137 0.45490196]
  [0.43137255 0.37254903 0.33333334]]
```

Разделение данных на тестовый и тренировочный наборы

Разделение данных на тестовый и тренировочный наборы

```
[ ] x_train, x_val, y_train, y_val = train_test_split(data, labels, test_size=0.3, random_state=1)
    print("training shape: ",x_train.shape, y_train.shape)
    print("testing shape: ",x_val.shape, y_val.shape)
    print(y_train[0])
```

```
training shape:  (27446, 32, 32, 3) (27446, 43)
testing shape:  (11763, 32, 32, 3) (11763, 43)
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Модель ResNet50

Обучение модели в числе 5 эпох

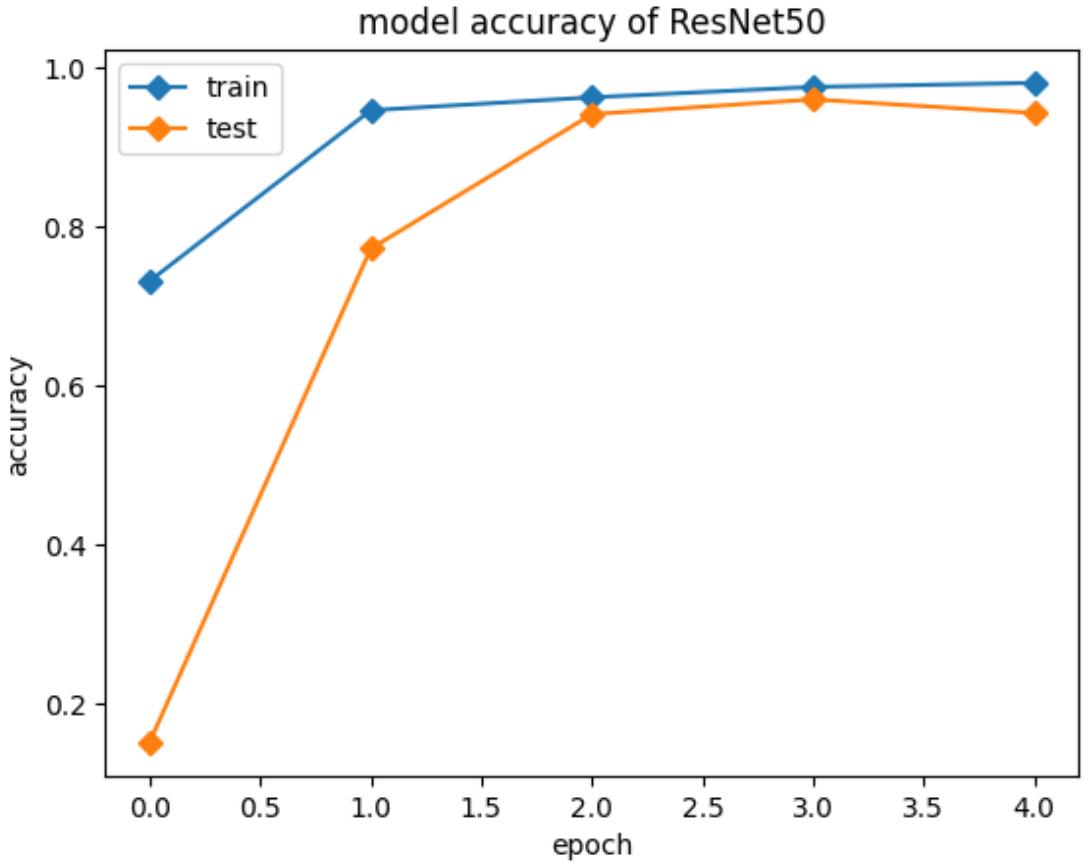
Обучение модели

```
model.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
history = model.fit(x_train, y_train, validation_data =(x_val, y_val), epochs = 5, batch_size = 64)
```

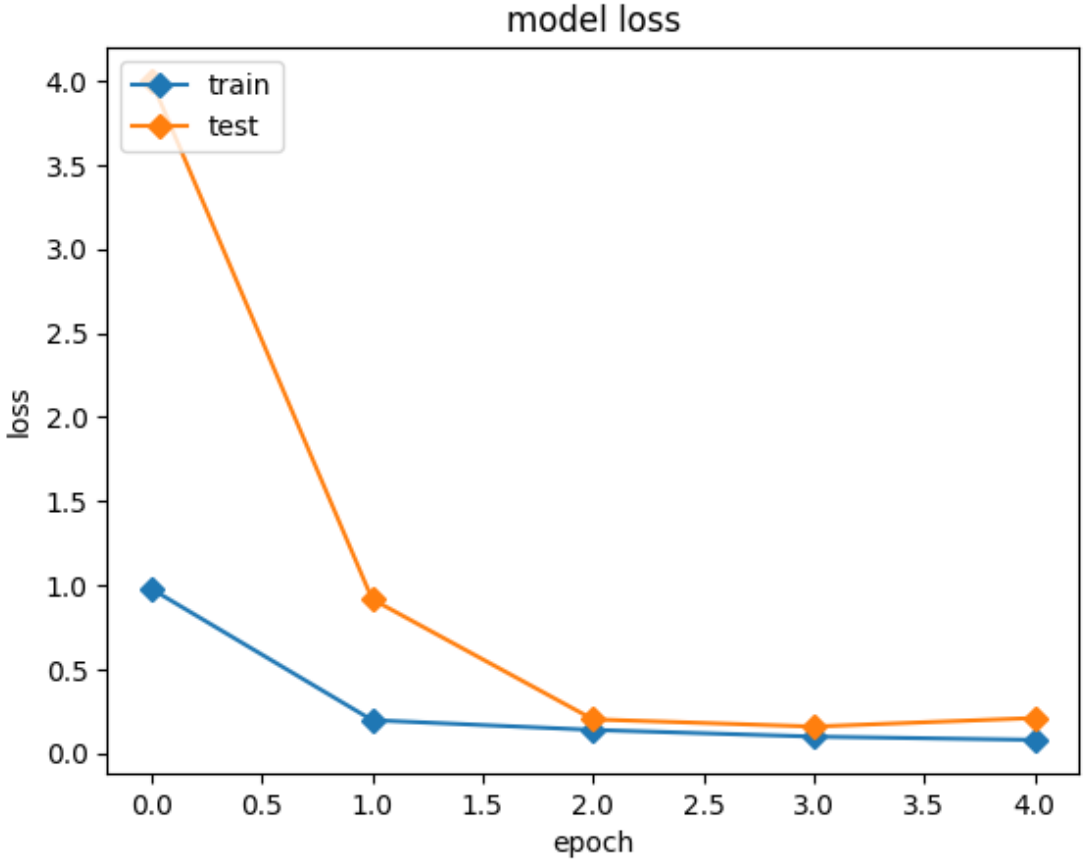
```
Epoch 1/5
429/429 ━━━━━━━━━━━━━━━━━━━━ 132s 138ms/step - accuracy: 0.5007 - loss: 1.9721 - val_accuracy: 0.2901 - val_loss: 2.8279
Epoch 2/5
429/429 ━━━━━━━━━━━━━━━━━━━━ 70s 47ms/step - accuracy: 0.9290 - loss: 0.2634 - val_accuracy: 0.7489 - val_loss: 1.0405
Epoch 3/5
429/429 ━━━━━━━━━━━━━━━━━━━━ 18s 41ms/step - accuracy: 0.9678 - loss: 0.1131 - val_accuracy: 0.9496 - val_loss: 0.1727
Epoch 4/5
429/429 ━━━━━━━━━━━━━━━━━━━━ 19s 45ms/step - accuracy: 0.9739 - loss: 0.1011 - val_accuracy: 0.9028 - val_loss: 1.8695
Epoch 5/5
429/429 ━━━━━━━━━━━━━━━━━━━━ 24s 52ms/step - accuracy: 0.9746 - loss: 0.1003 - val_accuracy: 0.9539 - val_loss: 0.1654
```

Точность модели ResNet50 в виде графиков

[17]


model accuracy of ResNet50
train
test
accuracy
epoch
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

.


model loss
train
test
loss
epoch
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Модель VGG16

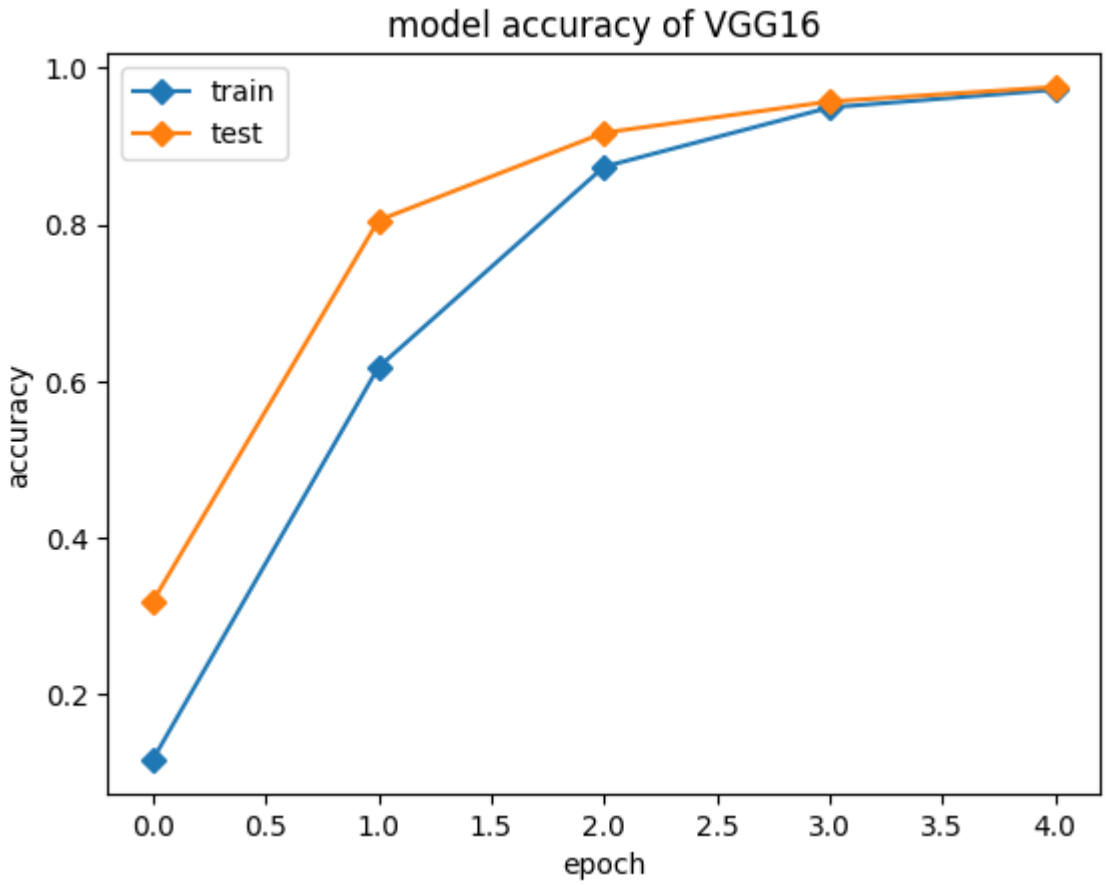
Обучение модели в числе 5 эпох

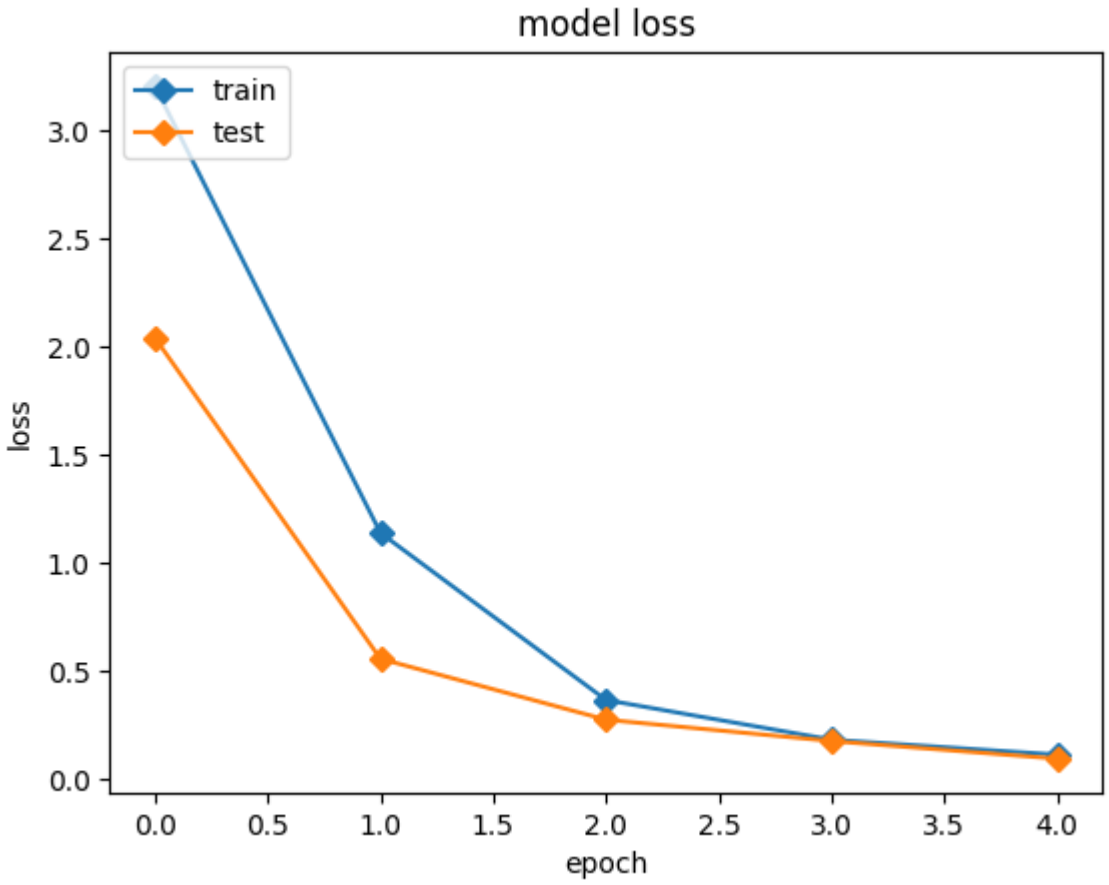
Обучение модели

```
model2.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
history2 = model2.fit(x_train, y_train, validation_data =(x_val, y_val), epochs = 5, batch_size = 64)
```

```
Epoch 1/5
429/429 [==============================] - 28s 49ms/step - loss: 3.2036 - accuracy: 0.1161 - val_loss: 2.0396 - val_accuracy: 0.3183
Epoch 2/5
429/429 [==============================] - 17s 40ms/step - loss: 1.1366 - accuracy: 0.6176 - val_loss: 0.5541 - val_accuracy: 0.8053
Epoch 3/5
429/429 [==============================] - 17s 40ms/step - loss: 0.3636 - accuracy: 0.8740 - val_loss: 0.2725 - val_accuracy: 0.9174
Epoch 4/5
429/429 [==============================] - 19s 43ms/step - loss: 0.1799 - accuracy: 0.9497 - val_loss: 0.1737 - val_accuracy: 0.9572
Epoch 5/5
429/429 [==============================] - 18s 41ms/step - loss: 0.1107 - accuracy: 0.9723 - val_loss: 0.0923 - val_accuracy: 0.9759
```

Точность модели ResNet50 в виде графиков





Model	Training Accuracy	Validation Accuracy	Test Accuracy
Resnet50	97.9961	94.2107	99.1924
VGG16	98.6568	99.4219	98.6568

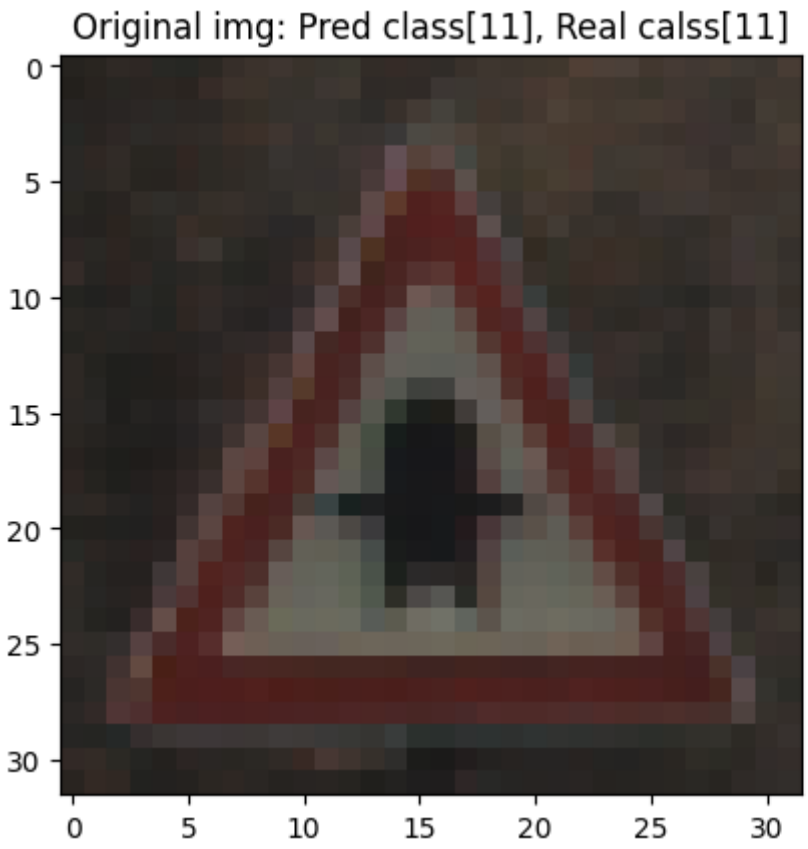
Задание 2

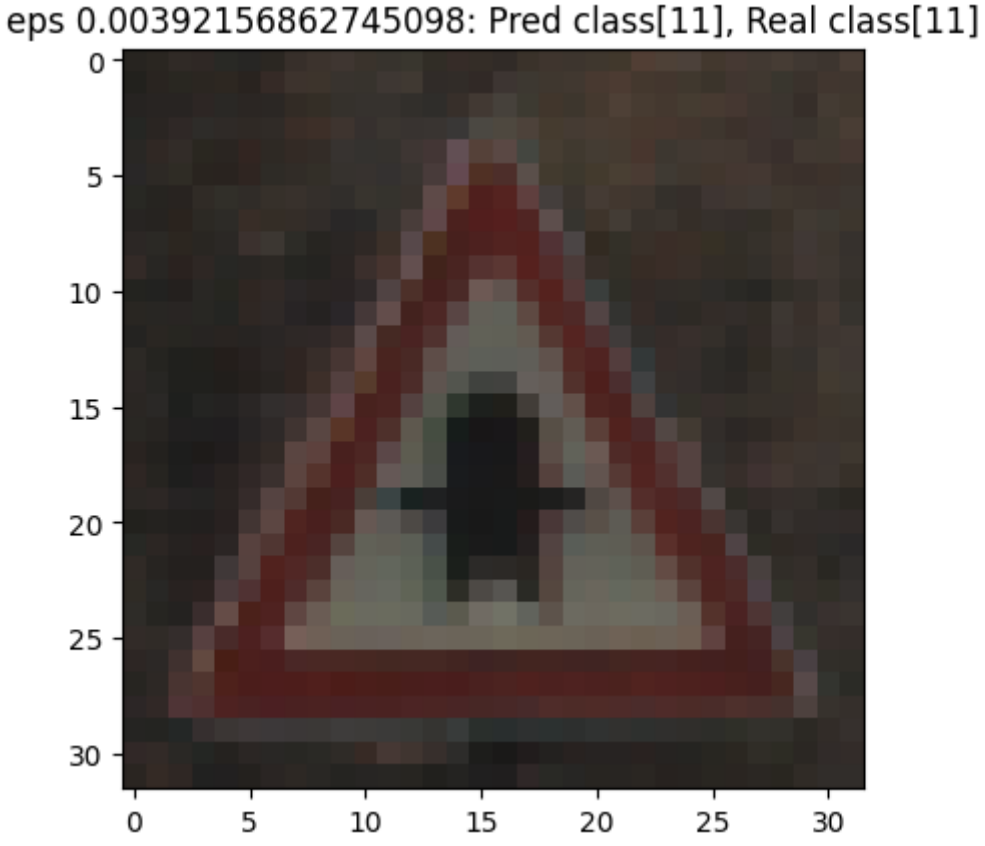
Атаки FGSM и PGD на модель ResNet50

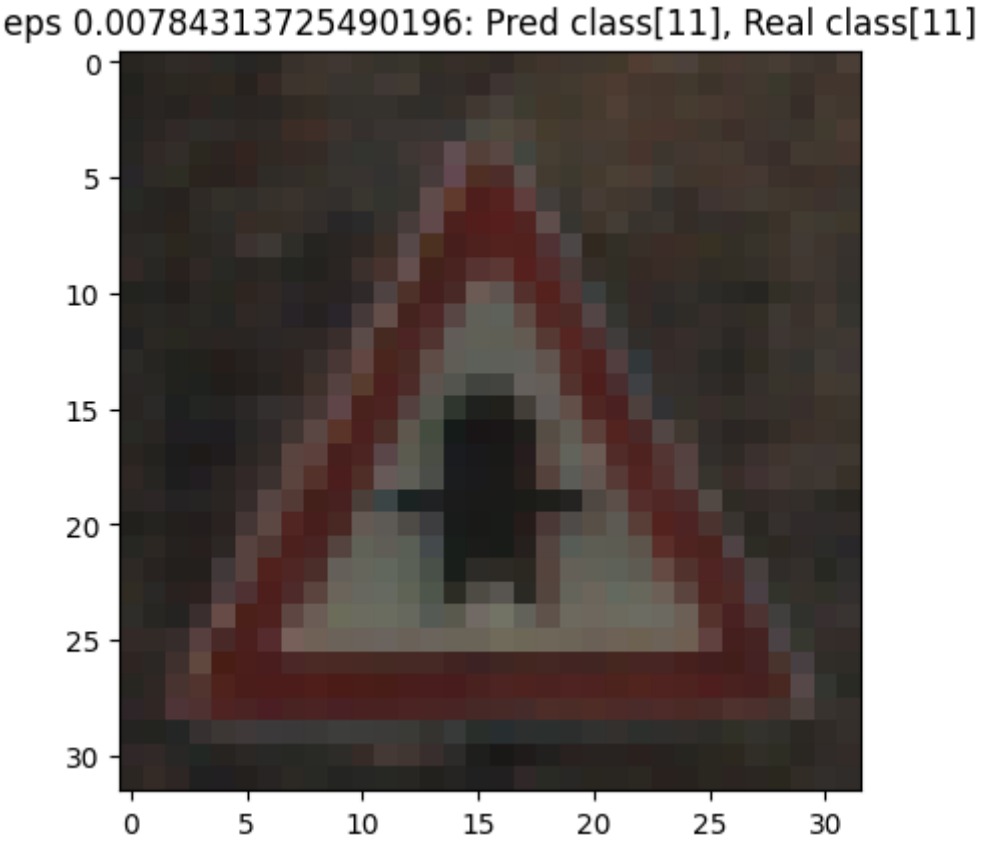
ResNet50 FGSM

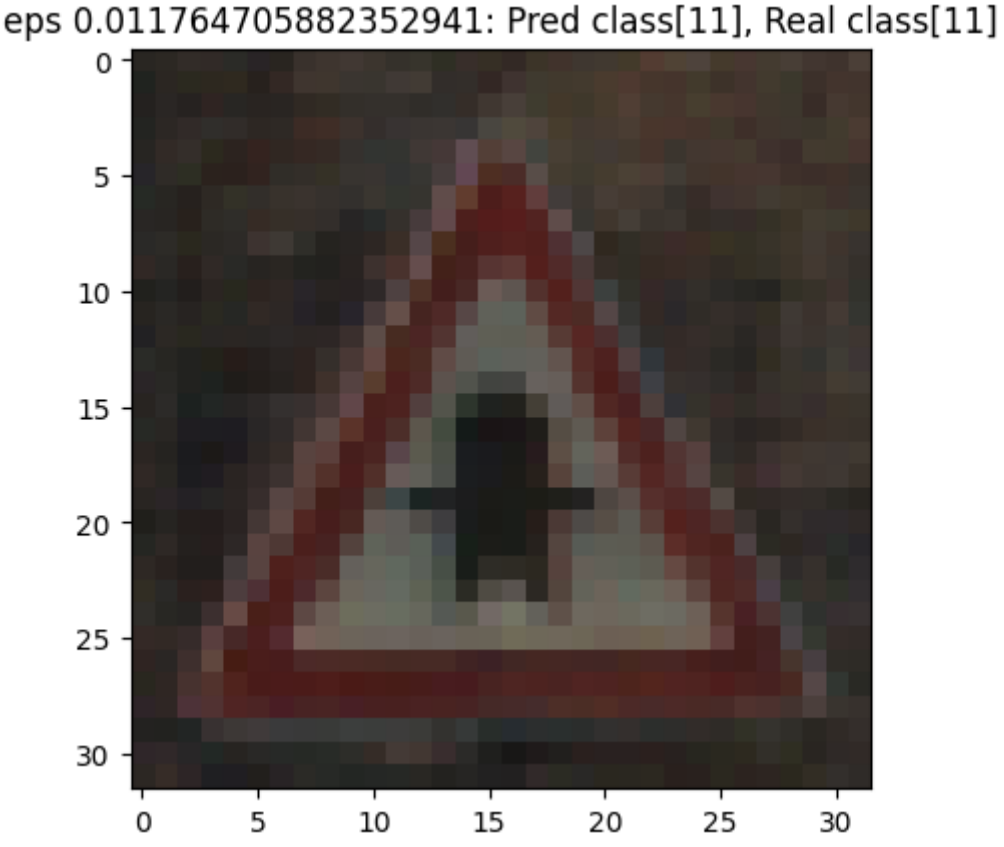
Атака FGSM

```
attack_fgsm = FastGradientMethod(estimator=classifier, eps=0.3)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
true_accuracies = []
adv_accuracises_fgsm = []
true_losses = []
adv_losses_fgsm = []

for eps in eps_range:
    attack_fgsm.set_params(**{'eps': eps})
    print(f"Eps: {eps}")
    x_test_adv = attack_fgsm.generate(x_test, y_test)
    loss, accuracy = model.evaluate(x_test_adv, y_test)
    adv_accuracises_fgsm.append(accuracy)
    adv_losses_fgsm.append(loss)
    print(f"Adv Loss: {loss}")
    print(f"Adv Accuracy: {accuracy}")
    loss, accuracy = model.evaluate(x_test, y_test)
    true_accuracies.append(accuracy)
    true_losses.append(loss)
```

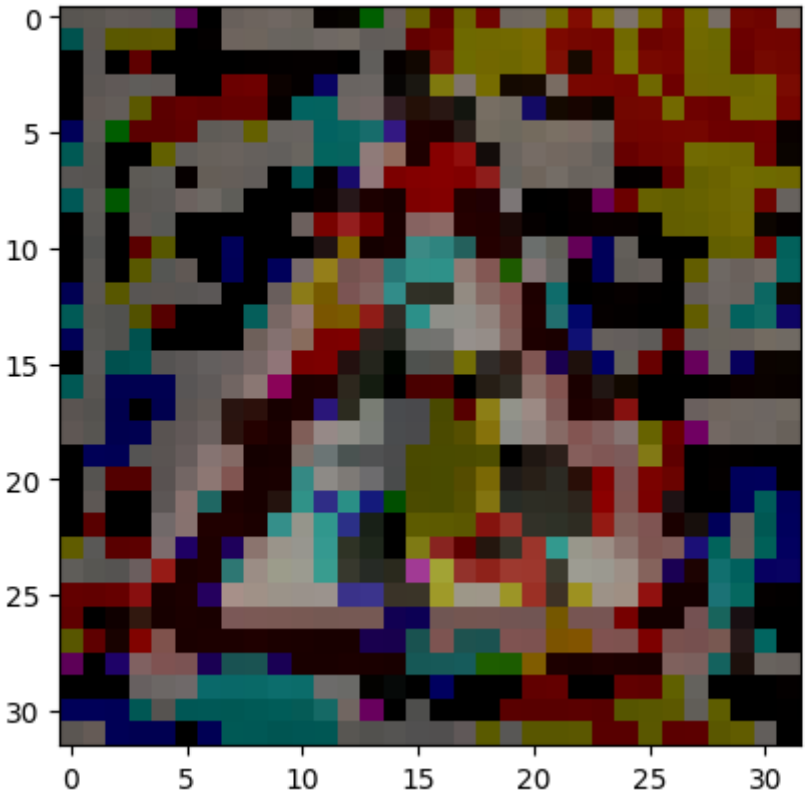

Original img: Pred class[11], Real calss[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


eps 0.00392156862745098: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

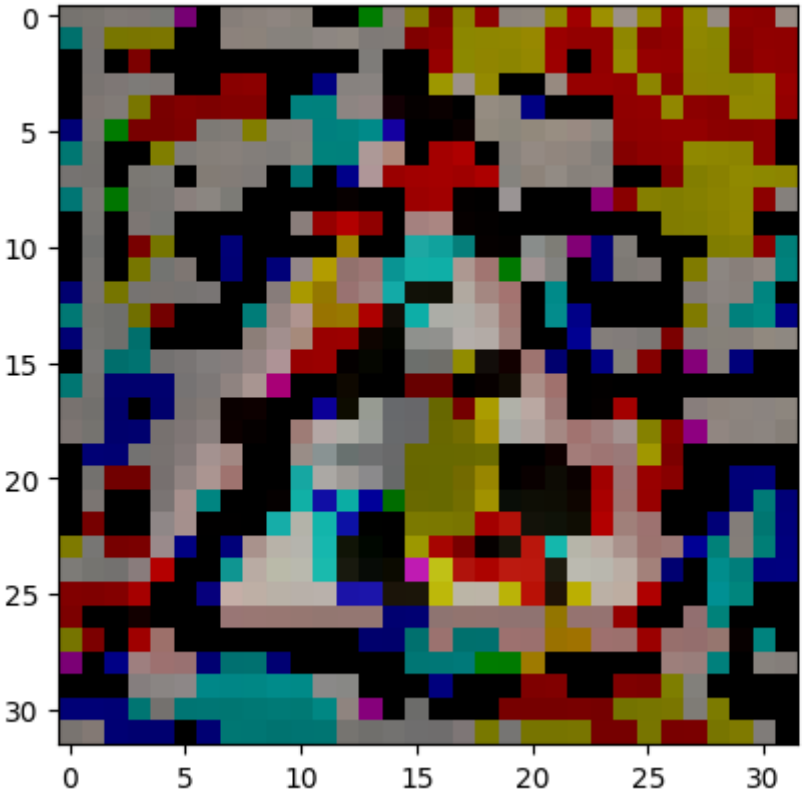

eps 0.00784313725490196: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


eps 0.011764705882352941: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

eps 0.19607843137254902: Pred class[24], Real class[11]



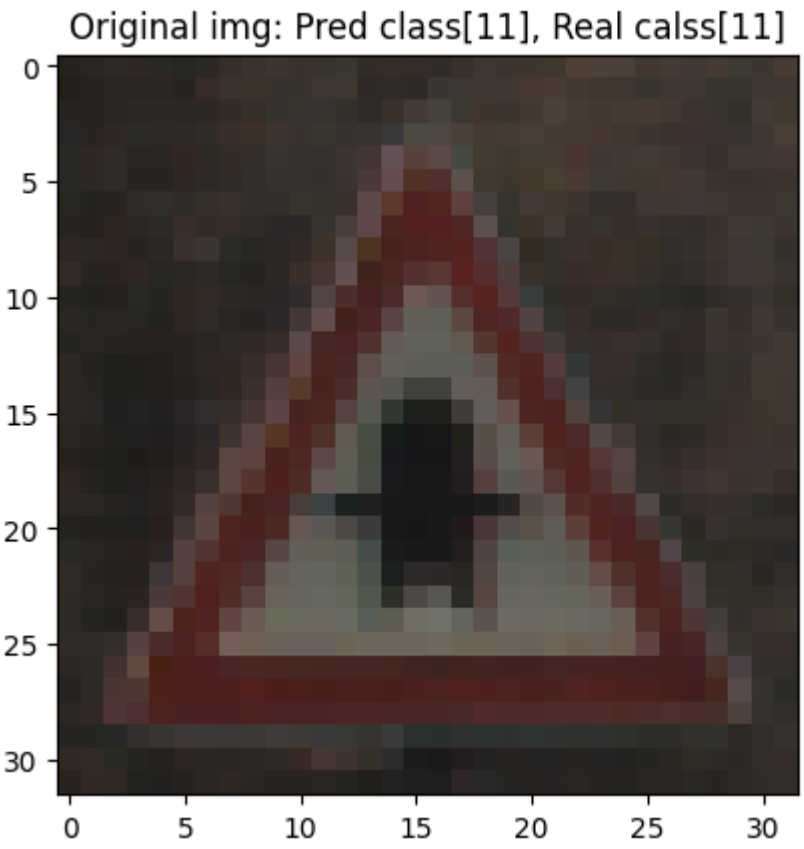
eps 0.3137254901960784: Pred class[24], Real class[11]

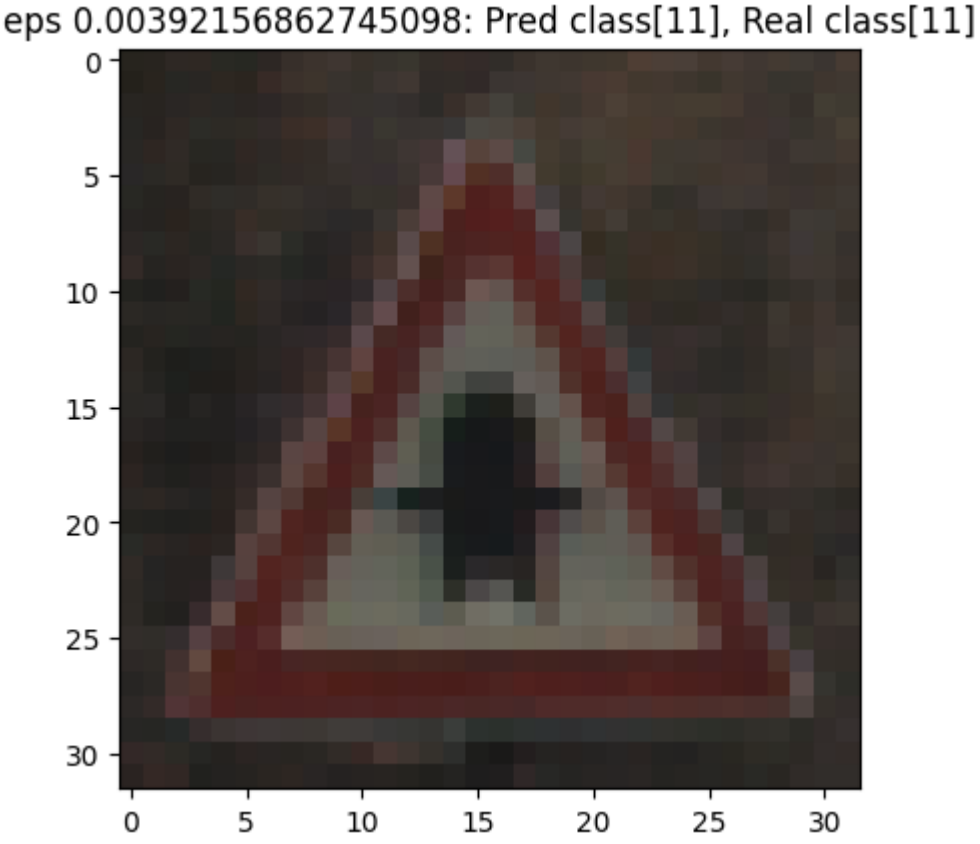


ResNet50 PGD

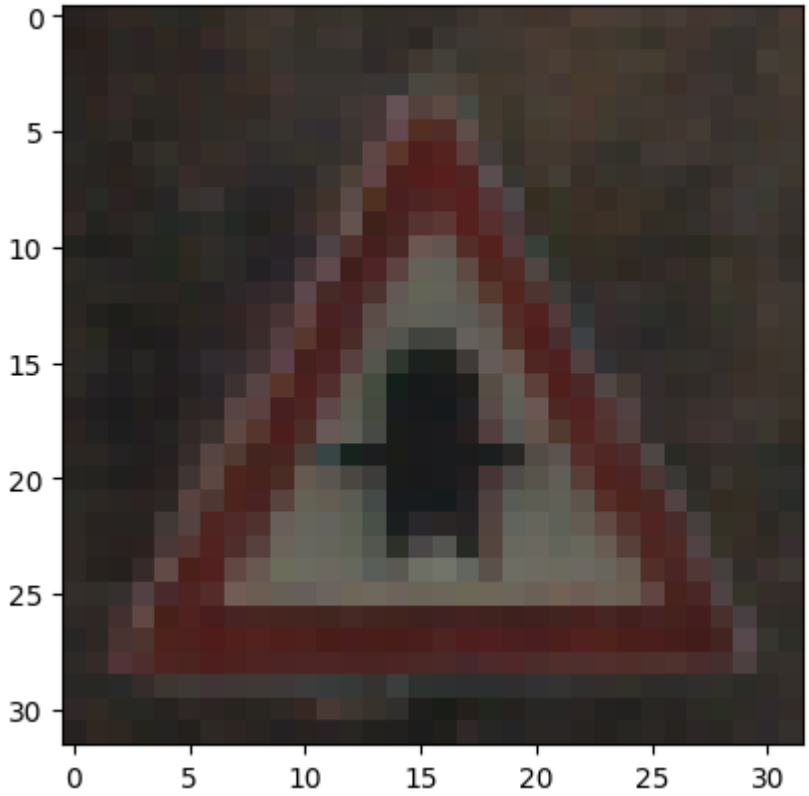
Атака PGD

```
attack_pgd = ProjectedGradientDescent(estimator=classifier, eps=0.3, max_iter=4, verbose=False)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
true_accuracies = []
adv_accuracises_pgd = []
true_losses = []
adv_losses_pgd = []

for eps in eps_range:
    attack_pgd.set_params(**{'eps': eps})
    print(f"Eps: {eps}")
    x_test_adv = attack_pgd.generate(x_test, y_test)
    loss, accuracy = model.evaluate(x_test_adv, y_test)
    adv_accuracises_pgd.append(accuracy)
    adv_losses_pgd.append(loss)
    print(f"Adv Loss: {loss}")
    print(f"Adv Accuracy: {accuracy}")
    loss, accuracy = model.evaluate(x_test, y_test)
    true_accuracies.append(accuracy)
    true_losses.append(loss)
    print(f"True Loss: {loss}")
    print(f"True Accuracy: {accuracy}")
```

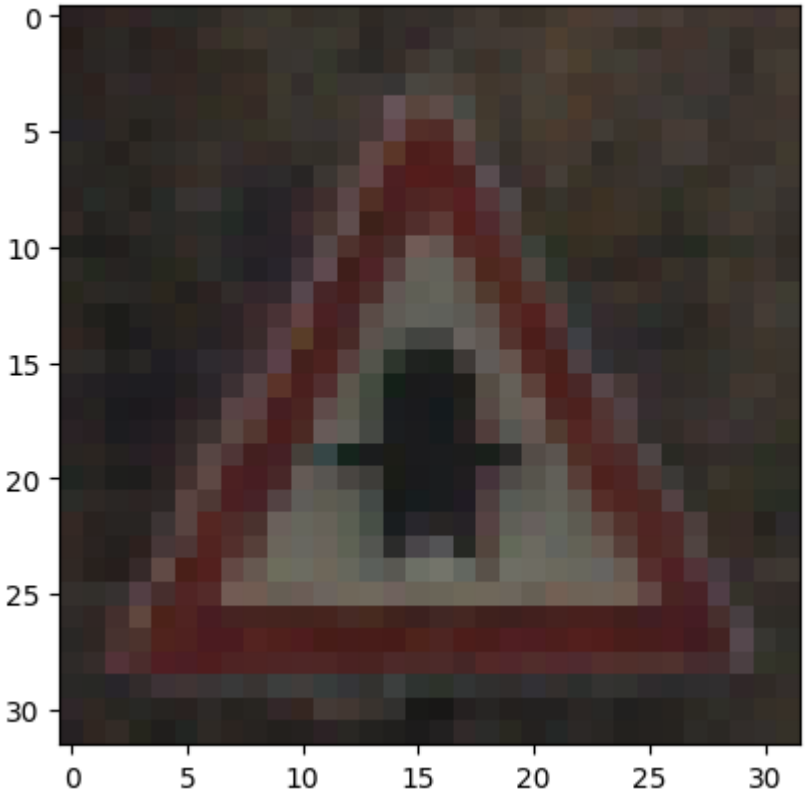

Original img: Pred class[11], Real calss[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

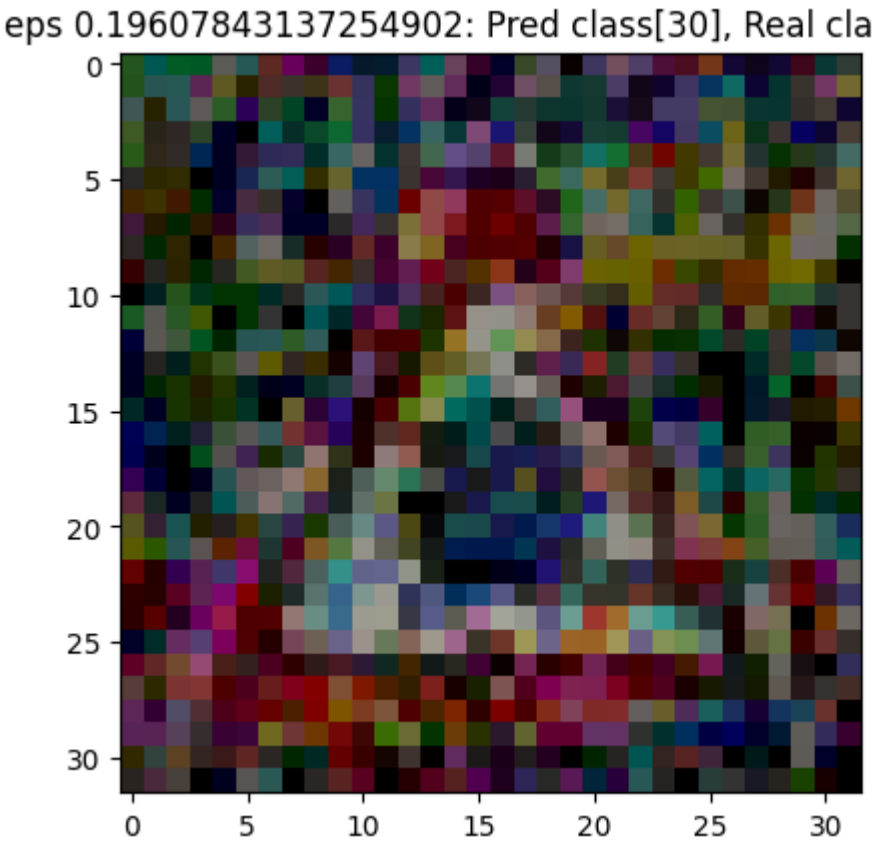

eps 0.00392156862745098: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

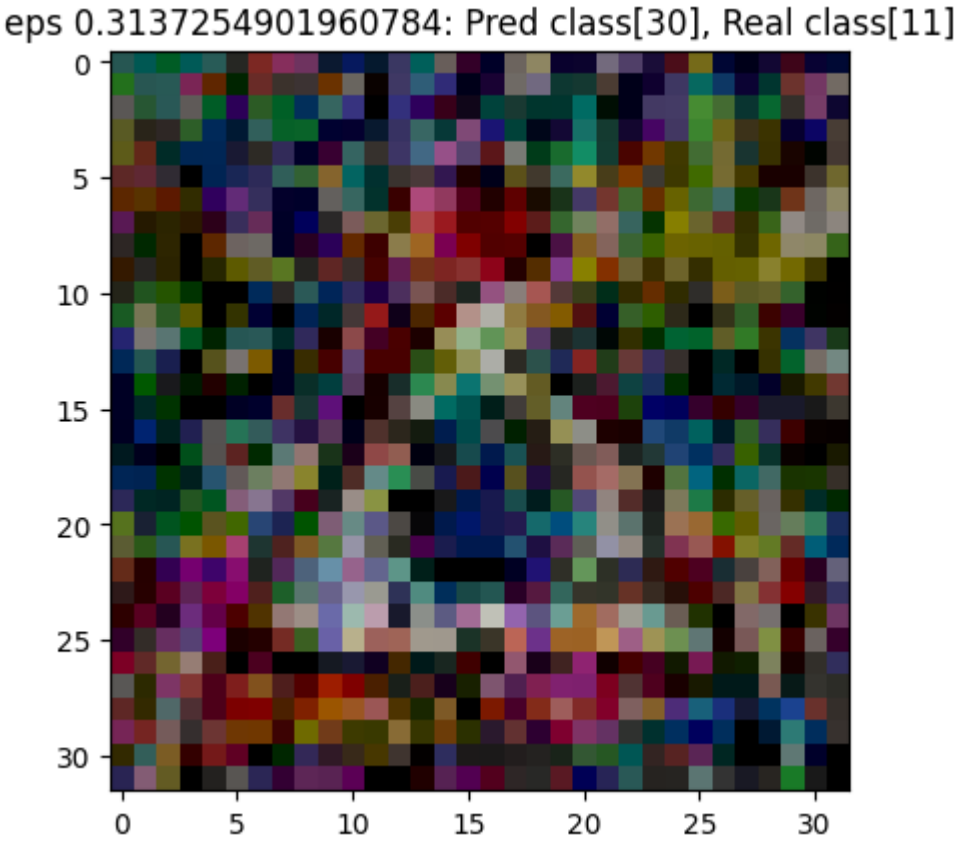
eps 0.00784313725490196: Pred class[30], Real class[11]

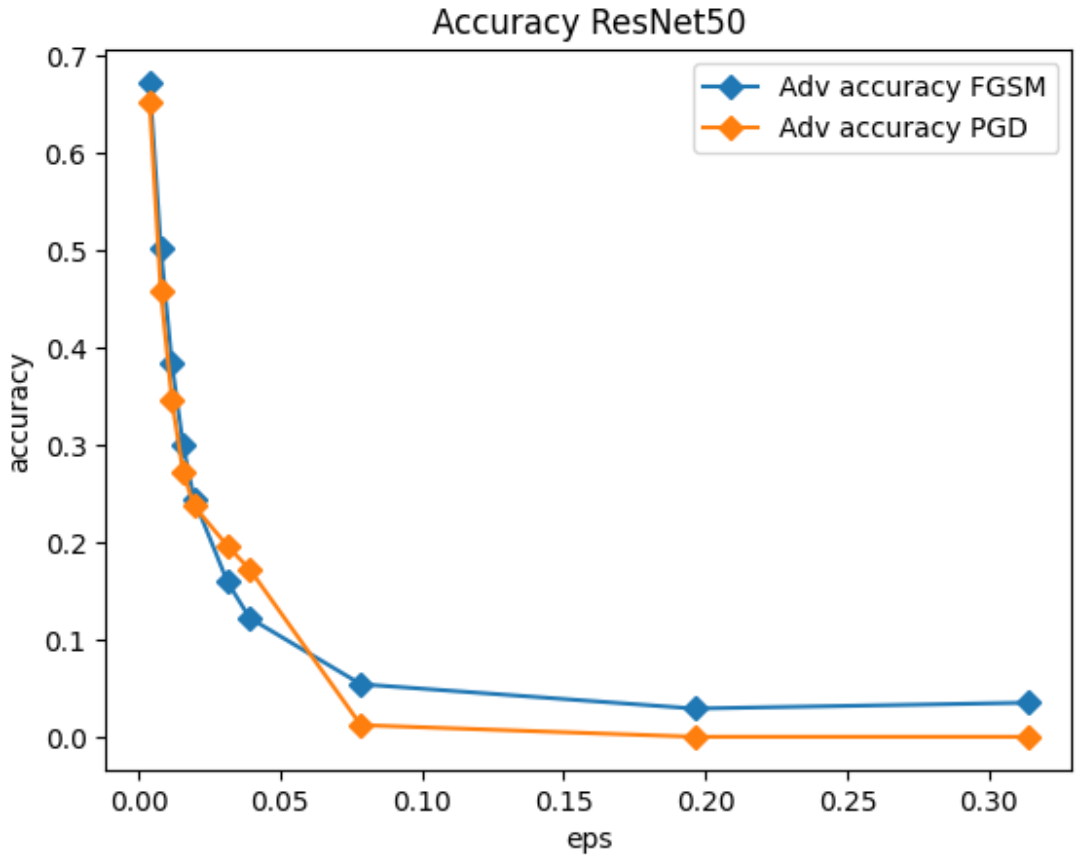


eps 0.011764705882352941: Pred class[30], Real class[11]




eps 0.19607843137254902: Pred class[30], Real class[11]
0
5
10
15
20
25
30

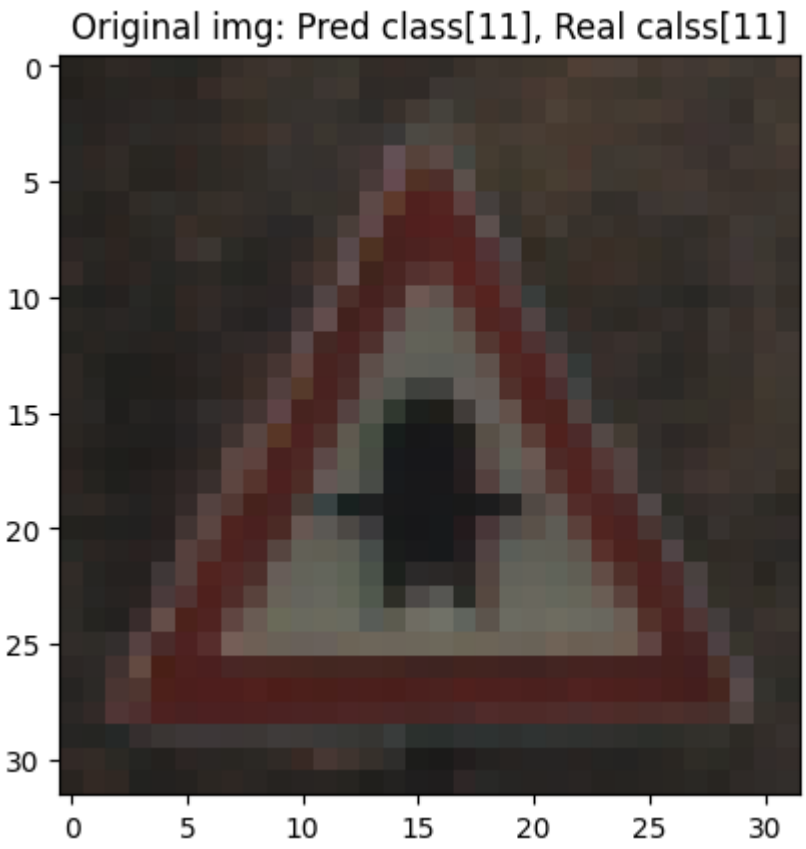

eps 0.3137254901960784: Pred class[30], Real class[11]
0
5
10
15
20
25
30

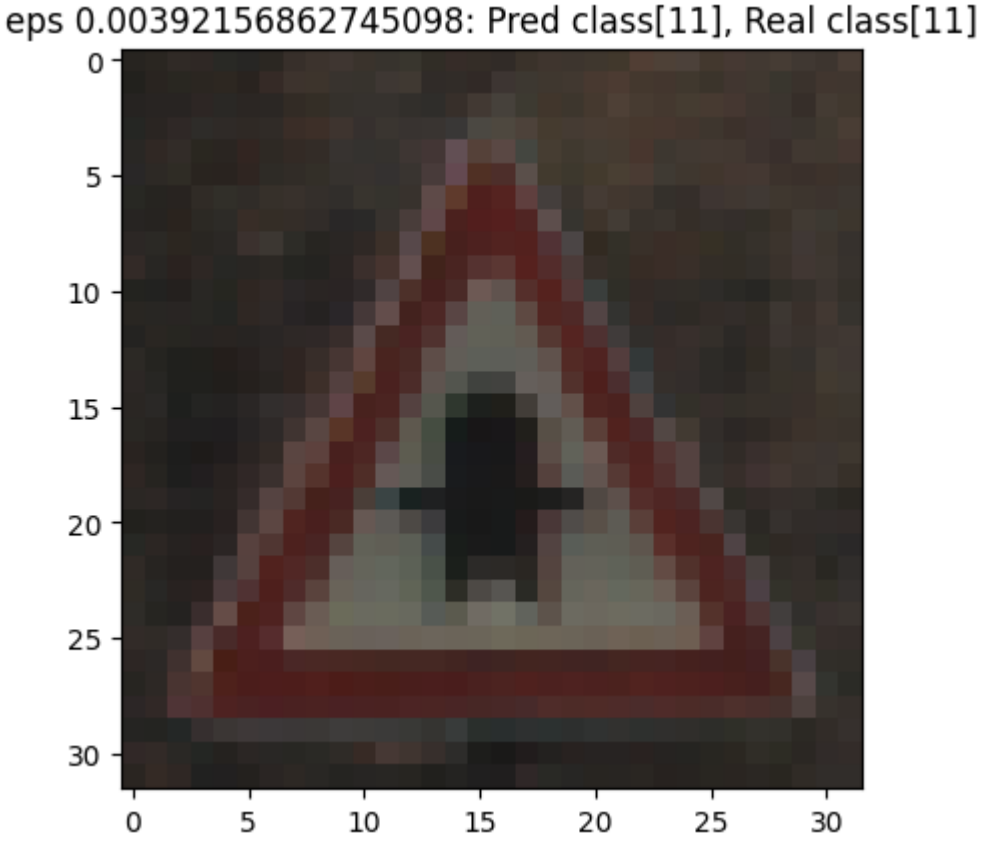


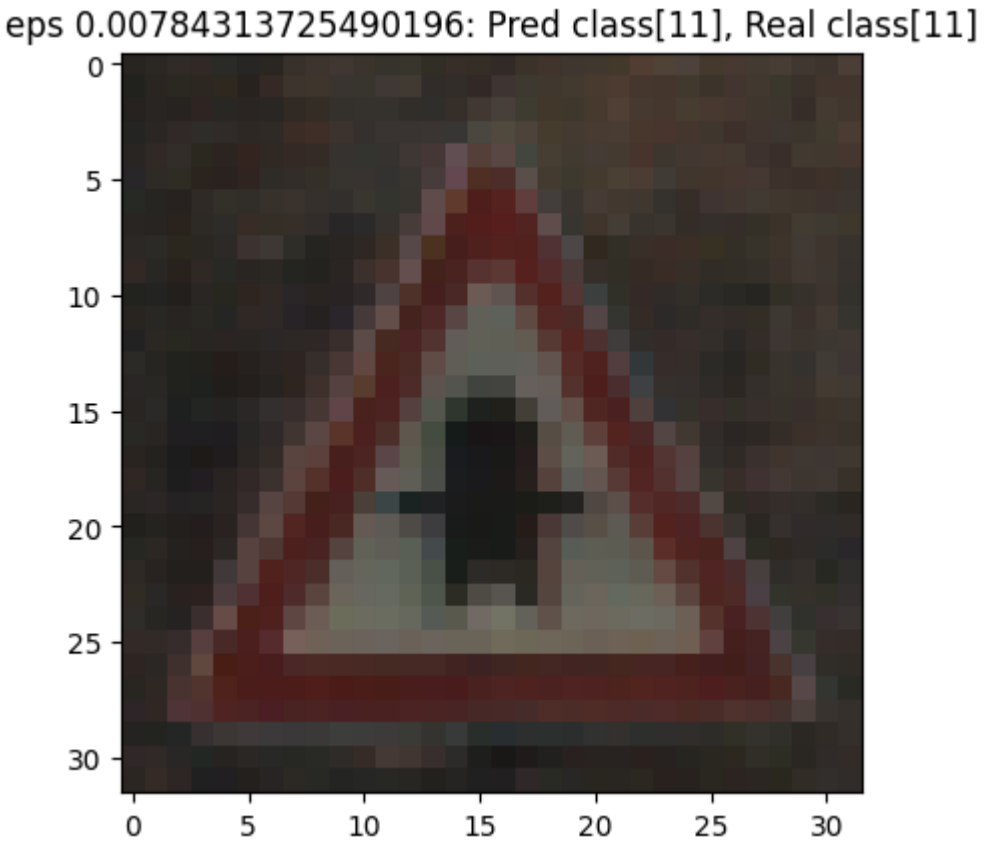
VGG16 FGSM

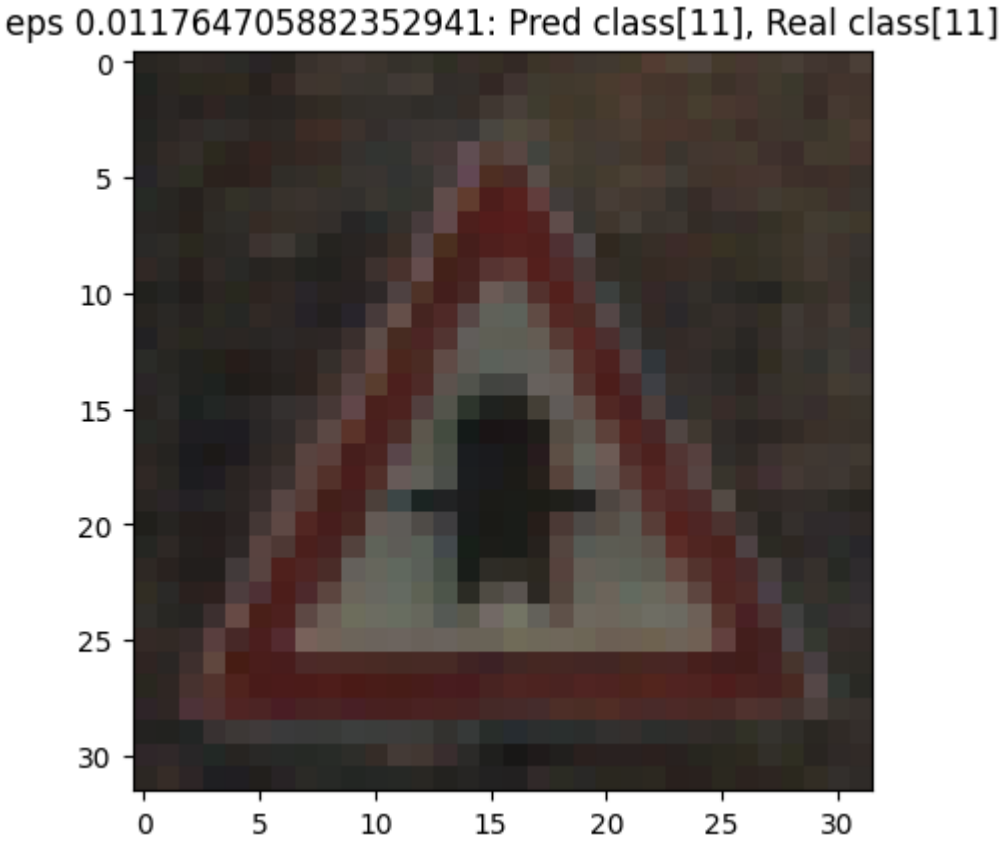
Атака FGSM для VGG16

```
attack_fgsm = FastGradientMethod(estimator=classifier, eps=0.3)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
true_accuracies = []
adv_accuracises_fgsm = []
true_losses = []
adv_losses_fgsm = []

for eps in eps_range:
    attack_fgsm.set_params(**{'eps': eps})
    print(f"Eps: {eps}")
    x_test_adv = attack_fgsm.generate(x_test, y_test)
    loss, accuracy = model.evaluate(x_test_adv, y_test)
    adv_accuracises_fgsm.append(accuracy)
    adv_losses_fgsm.append(loss)
    print(f"Adv Loss: {loss}")
    print(f"Adv Accuracy: {accuracy}")
    loss, accuracy = model.evaluate(x_test, y_test)
    true_accuracies.append(accuracy)
    true_losses.append(loss)
```

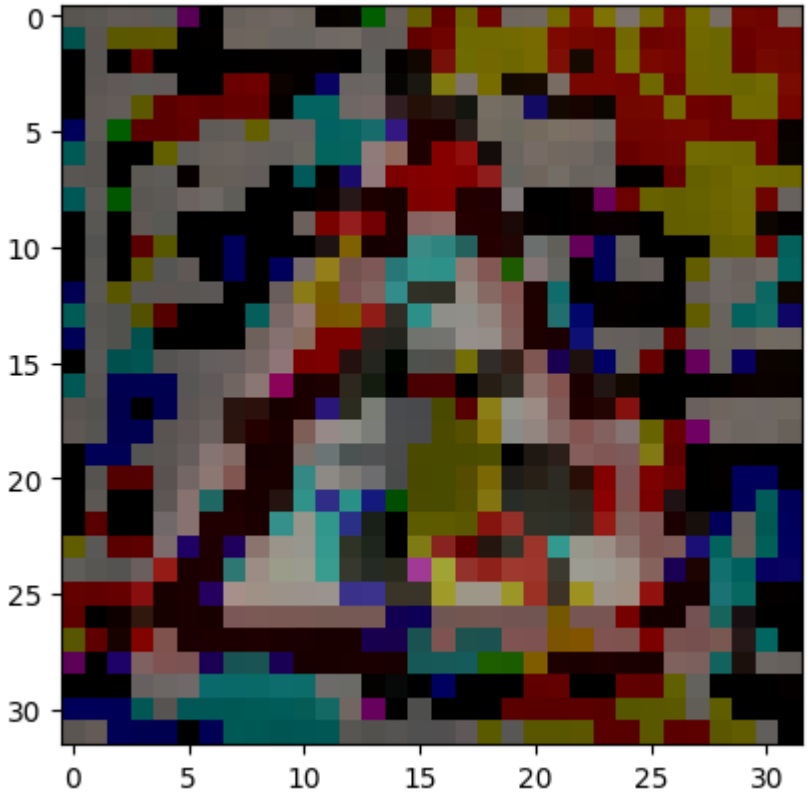

Original img: Pred class[11], Real calss[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


eps 0.00392156862745098: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

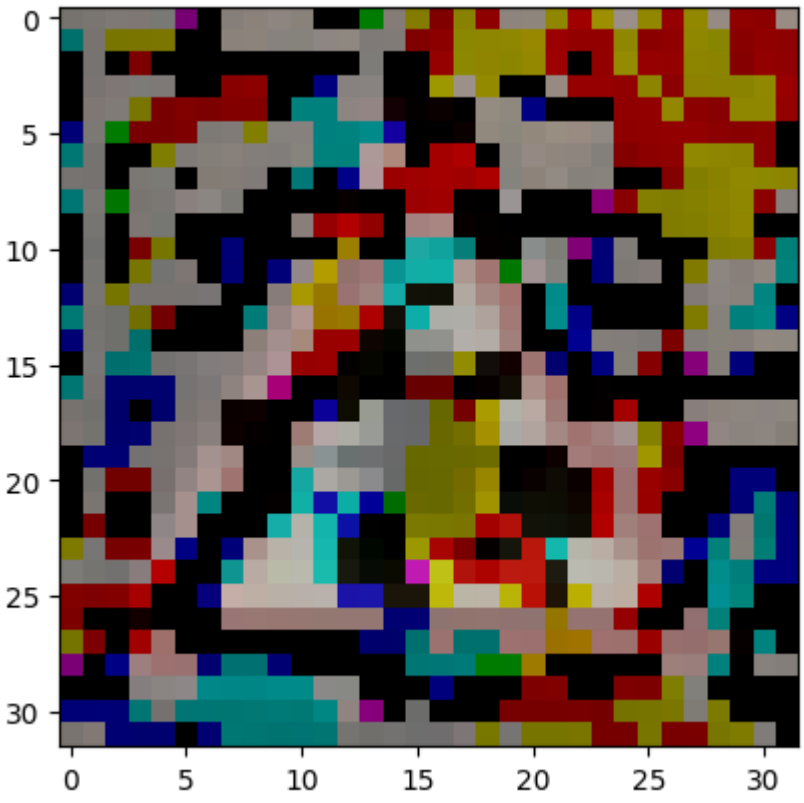

eps 0.00784313725490196: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


eps 0.011764705882352941: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

eps 0.19607843137254902: Pred class[24], Real class[11]



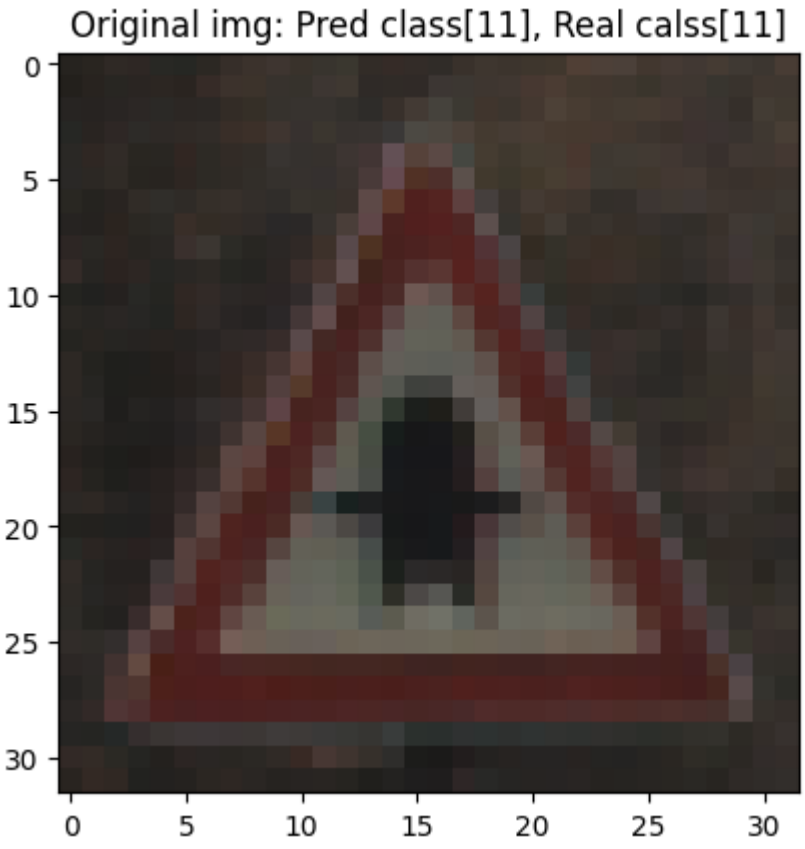
eps 0.3137254901960784: Pred class[24], Real class[11]

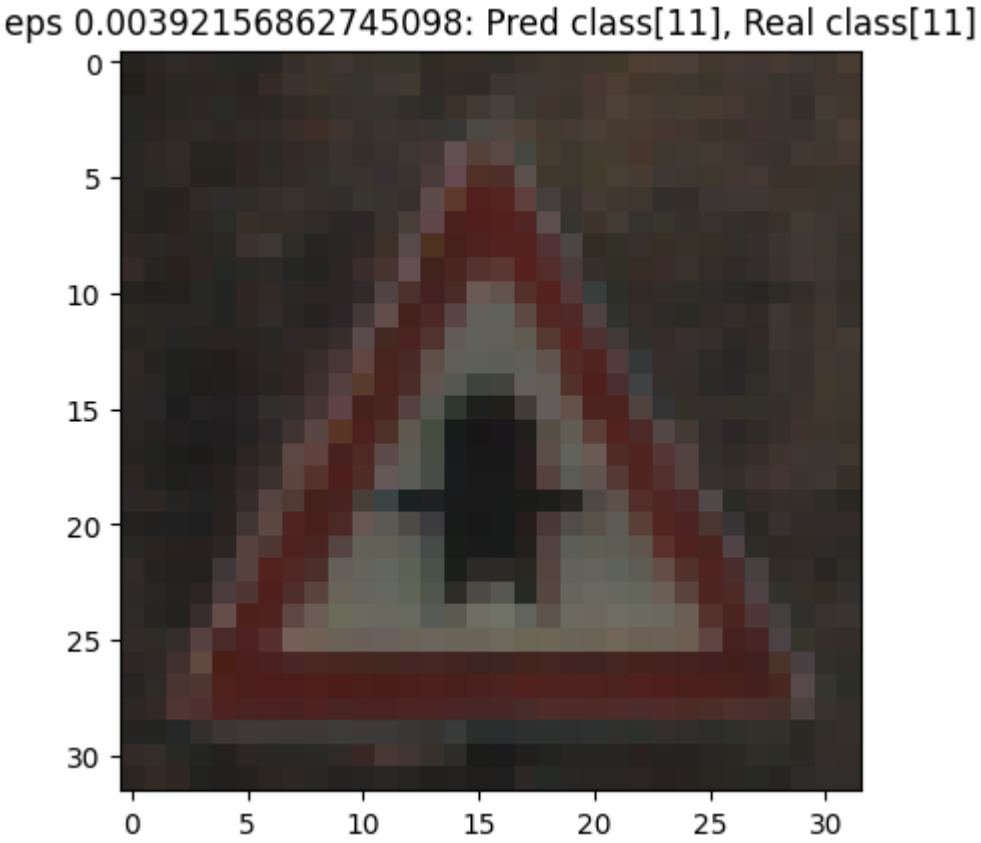


VGG16 PGD

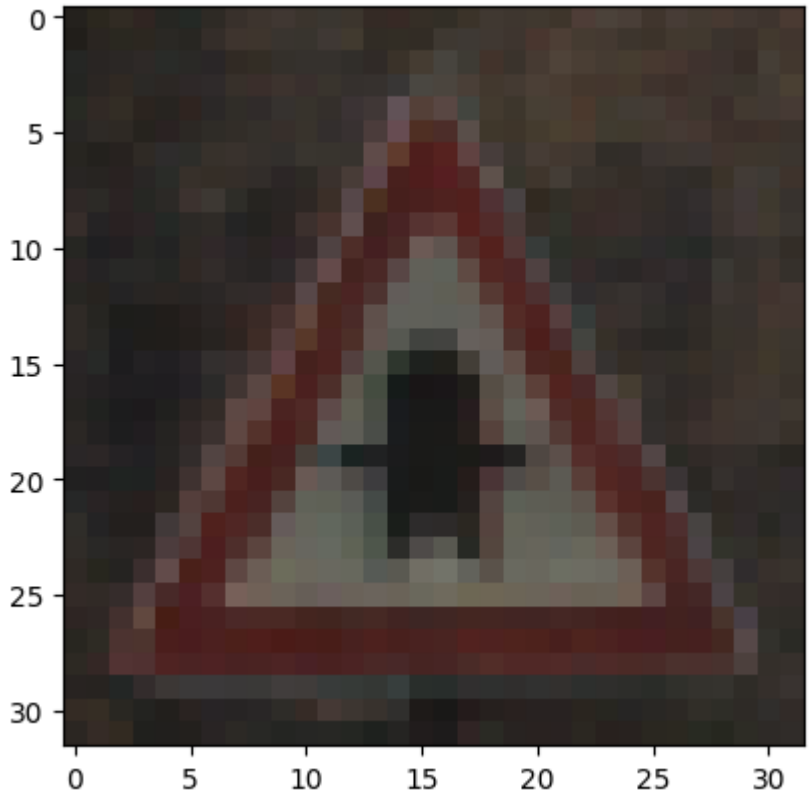
Атака PGD для VGG16

```
attack_pgd = ProjectedGradientDescent(estimator=classifier, eps=0.3, max_iter=4, verbose=False)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
true_accuracies = []
adv_accuracises_pgd = []
true_losses = []
adv_losses_pgd = []

for eps in eps_range:
    attack_pgd.set_params(**{'eps': eps})
    print(f"Eps: {eps}")
    x_test_adv = attack_pgd.generate(x_test, y_test)
    loss, accuracy = model.evaluate(x_test_adv, y_test)
    adv_accuracises_pgd.append(accuracy)
    adv_losses_pgd.append(loss)
    print(f"Adv Loss: {loss}")
    print(f"Adv Accuracy: {accuracy}")
    loss, accuracy = model.evaluate(x_test, y_test)
    true_accuracies.append(accuracy)
    true_losses.append(loss)
```

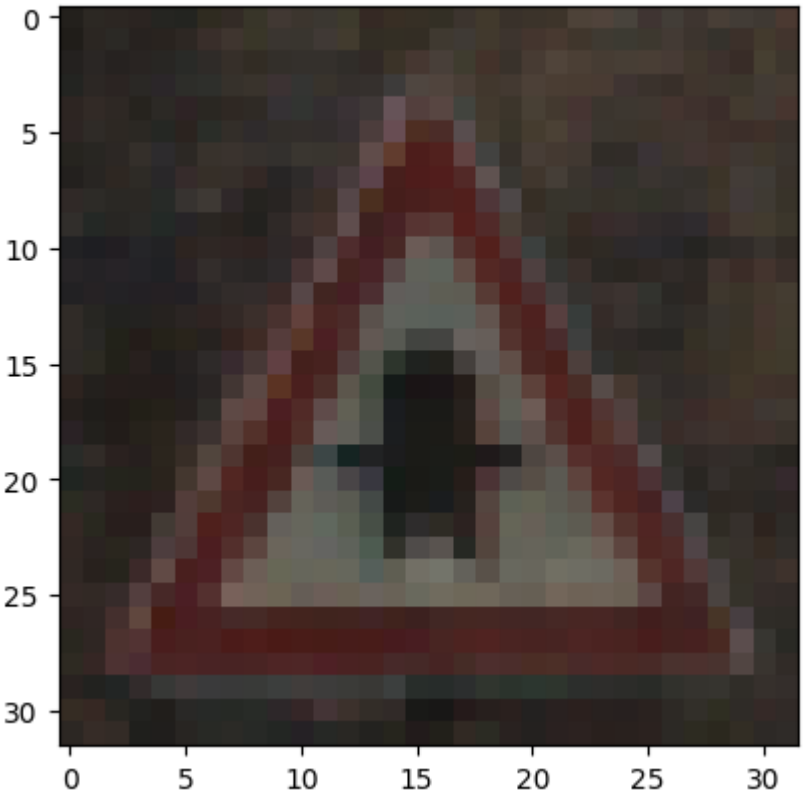

Original img: Pred class[11], Real calss[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

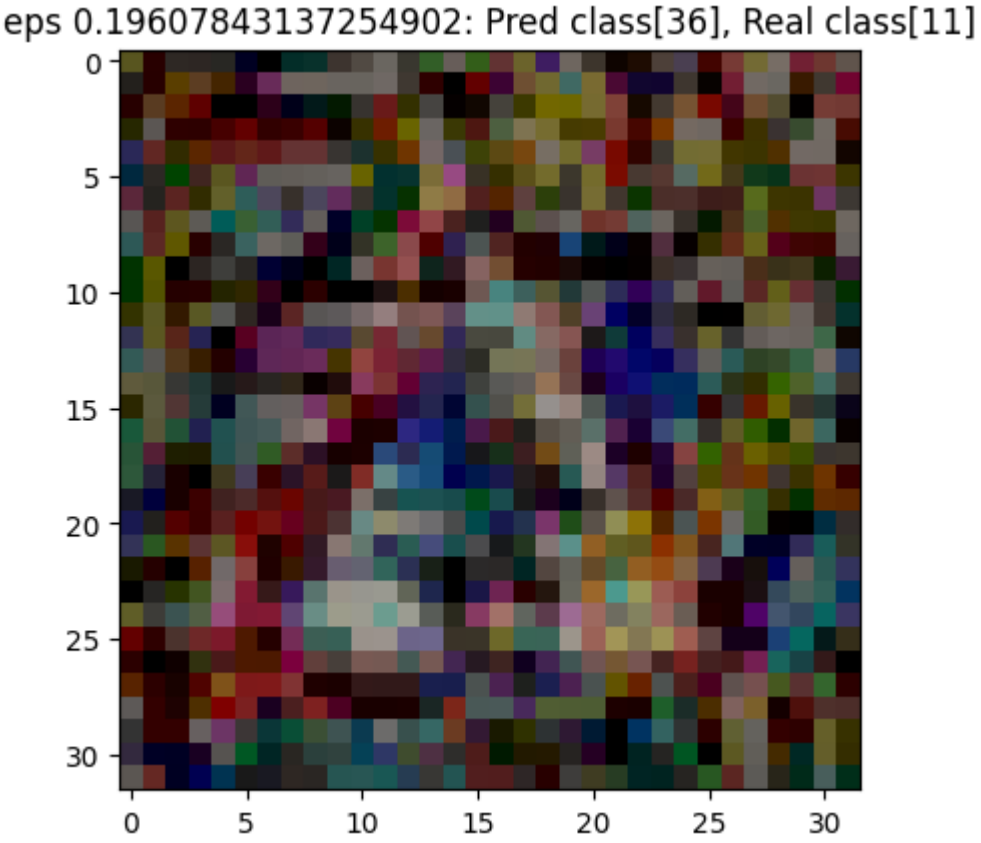

eps 0.00392156862745098: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

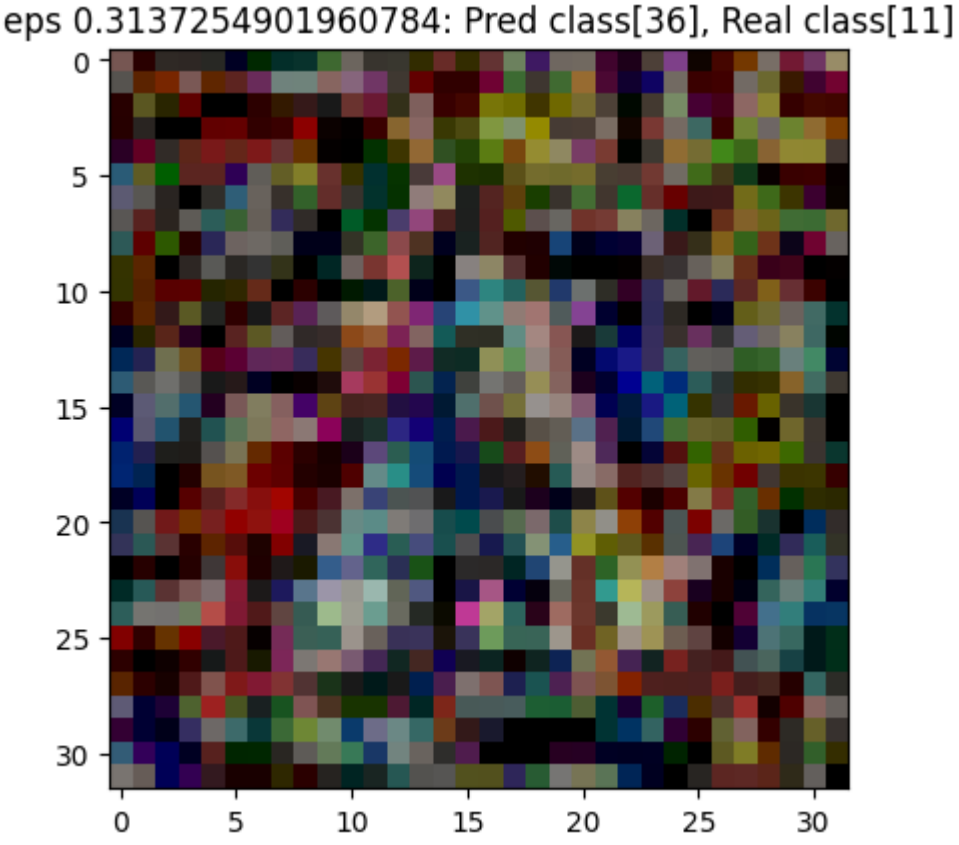
eps 0.007843137254901961: Pred class[11], Real class[11]

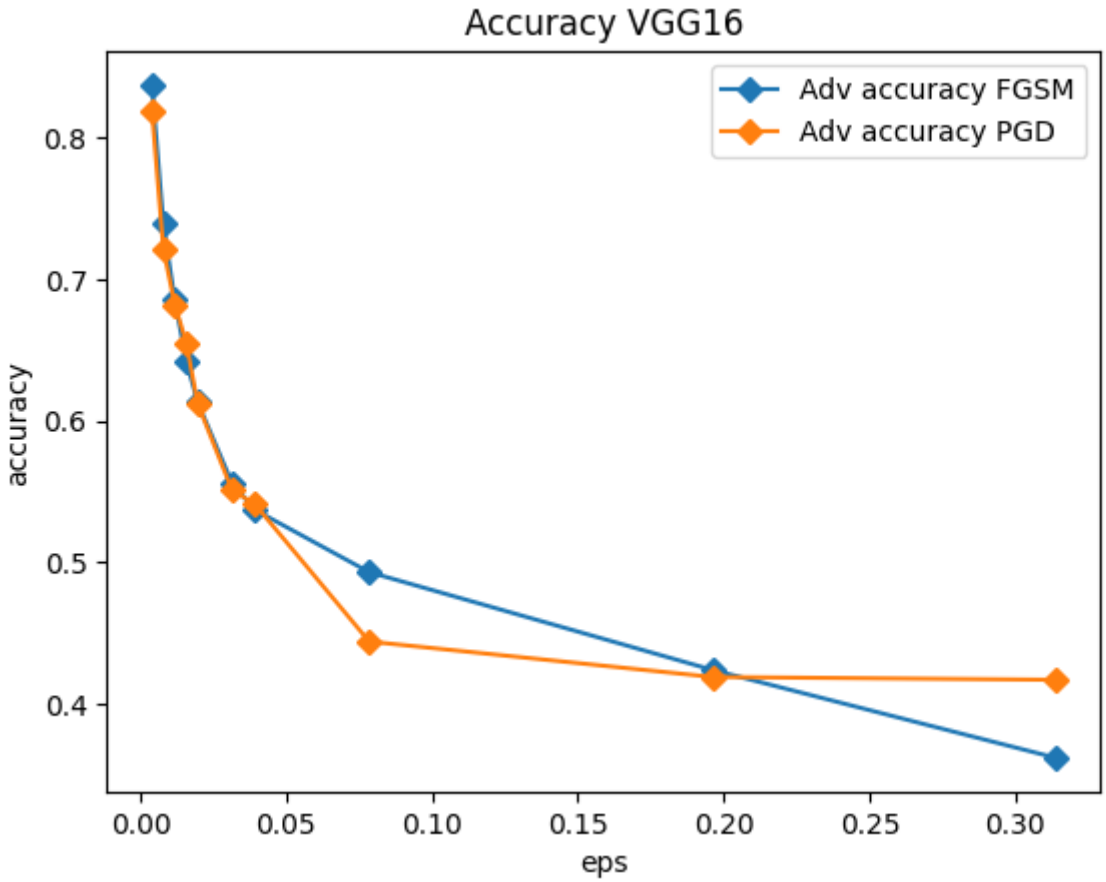


eps 0.011764705882352941: Pred class[11], Real class[11]




eps 0.19607843137254902: Pred class[36], Real class[11]
0
5
10
15
20
25
30


eps 0.3137254901960784: Pred class[36], Real class[11]
0
5
10
15
20
25
30



Model	Original accuracy	eps = 1/255	eps = 2/255	eps = 3/255	eps = 4/255
Resnet50 FGSM	97.9961	67.2	50.2	38.3	30
Resnet50 PGD	97.9961	65.2	45.8	34.6	27.2
VGG16 FGSM	98.6568	83.7	73.9	68.5	64.2
VGG16 PGD	98.6568	81.9	72.1	68.2	65.5

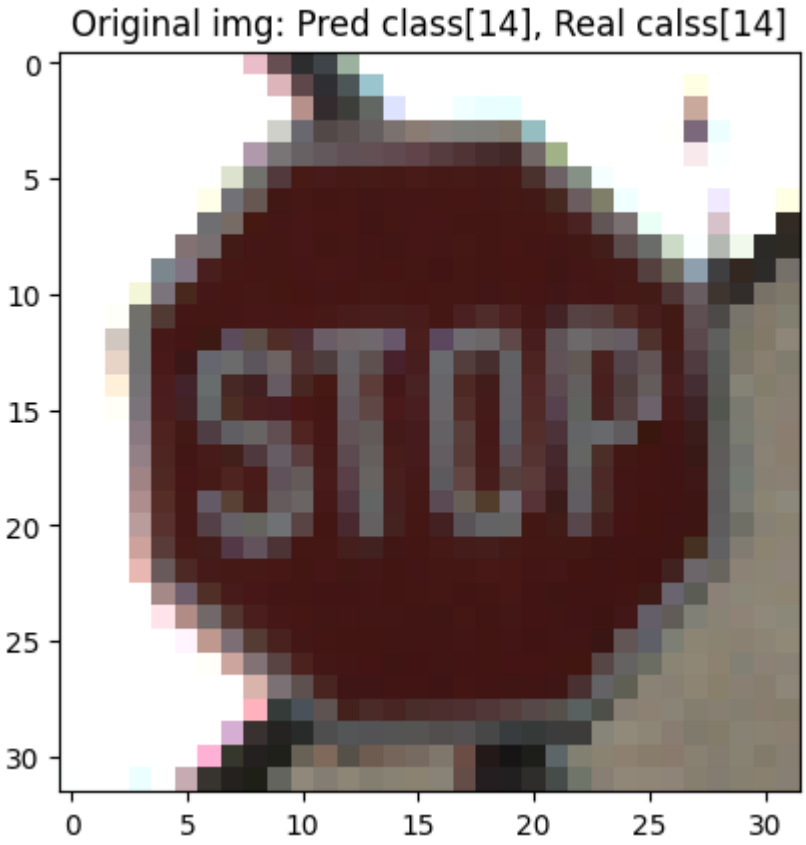
eps = 5/255	eps = 8/255	eps = 10/255	eps = 20/255	eps = 50/255	eps = 80/255
24.4	15.9	12.2	5.4	2.9	3.5
23.7	19.6	17.2	1.2	0	0
61.3	55.6	53.7	49.3	42.4	36.2
61.2	55.1	54.1	44.4	41.9	41.7

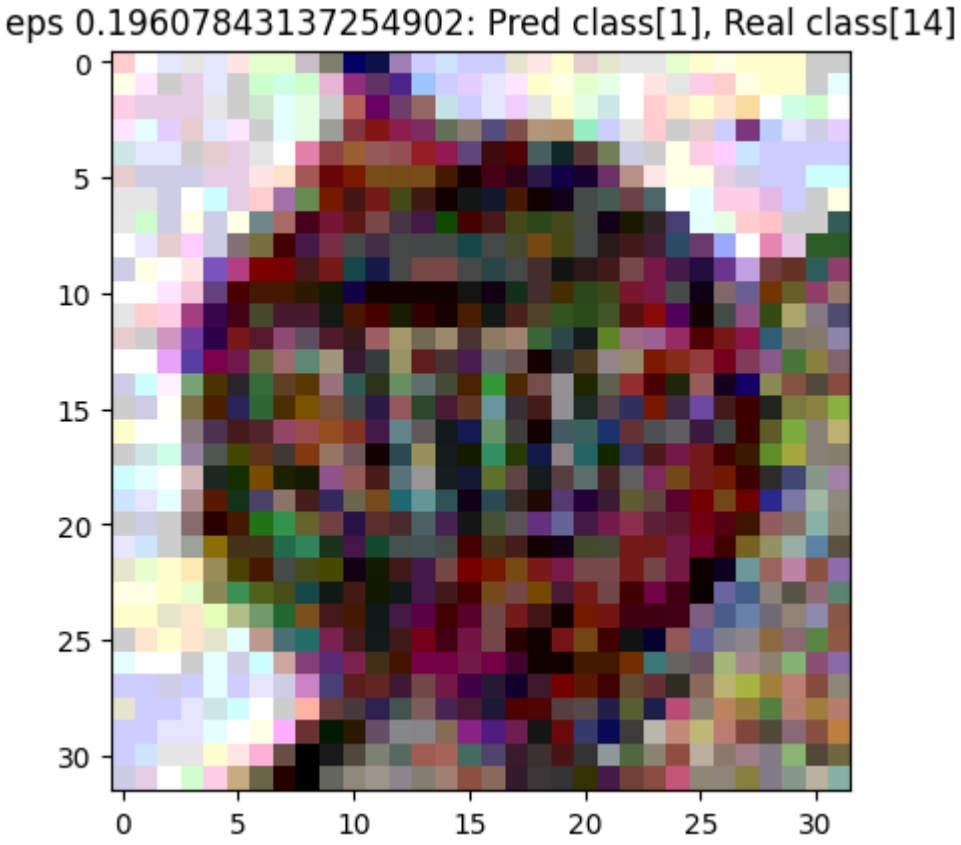
Задание 3

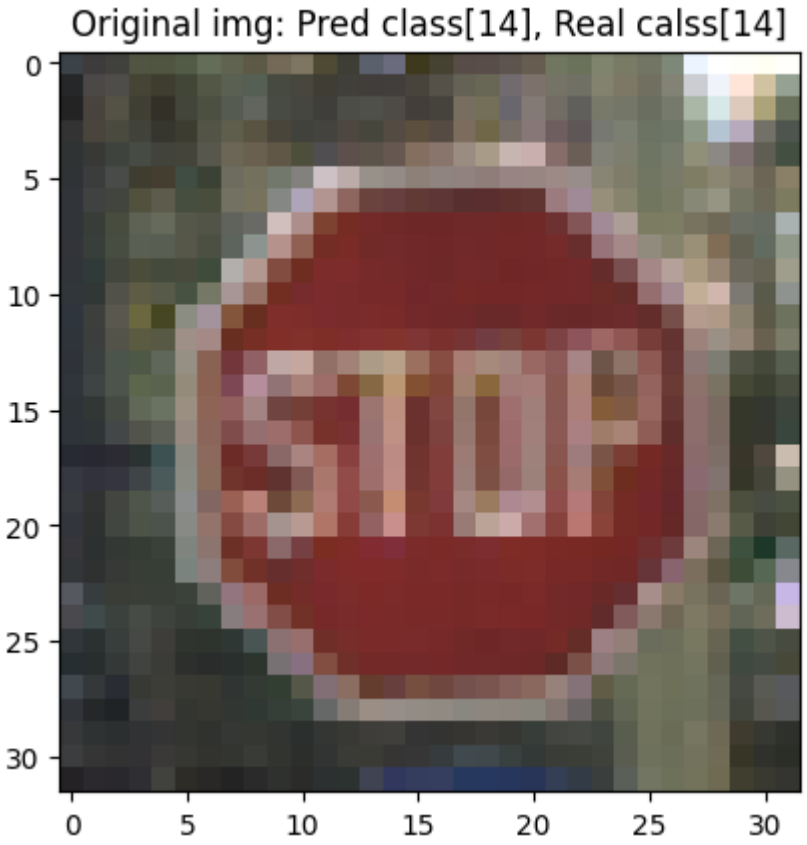
Атака FGSM на знак Стоп

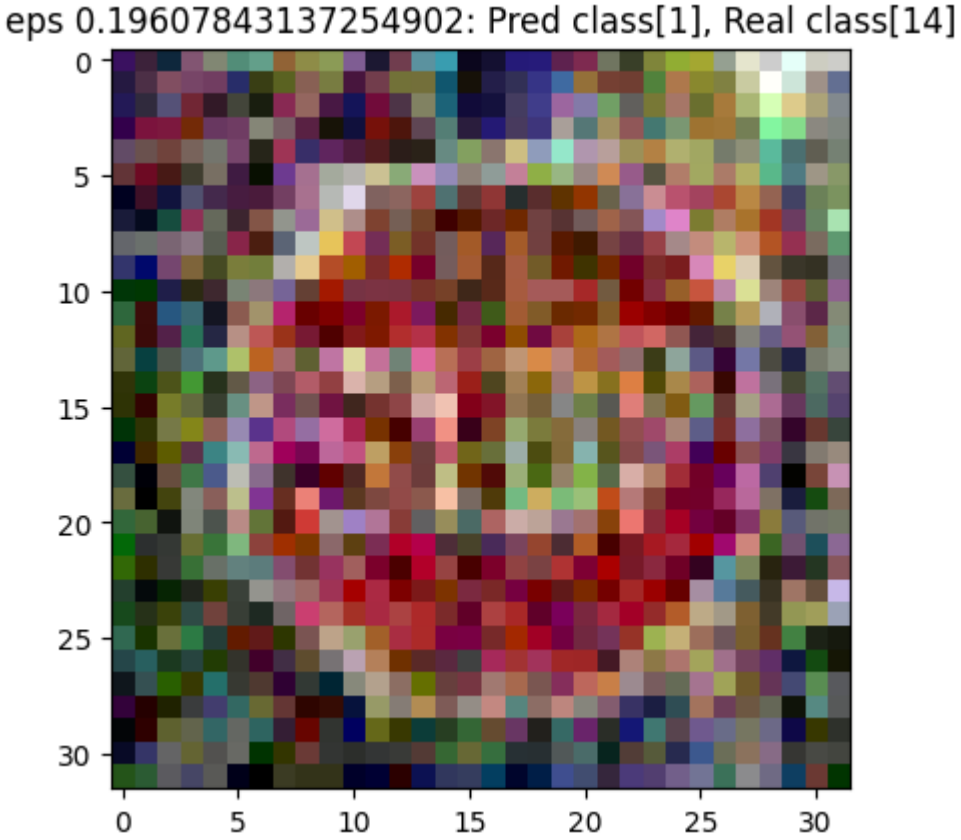
Атака FGSM

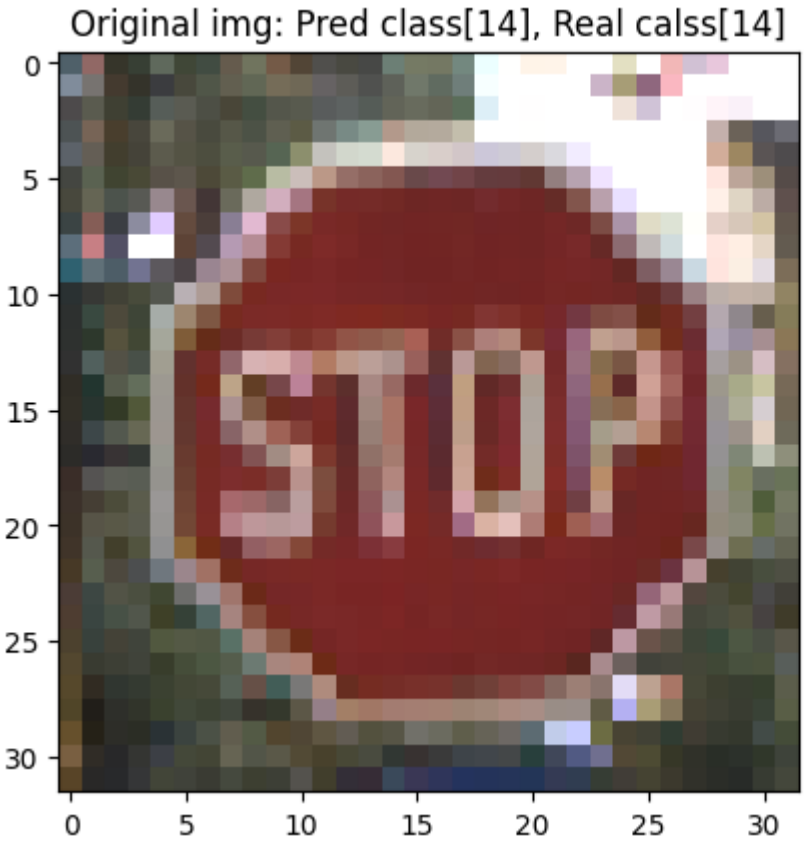
```
model=load_model('ResNet50.h5')
tf.compat.v1.disable_eager_execution()
t_class = 1
t_class = to_categorical(t_class, 43)
t_classes = np.tile(t_class, (270, 1))
x_test = data
classifier = KerasClassifier(model=model, clip_values=(np.min(x_test), np.max(x_test)))
attack_fgsm = FastGradientMethod(estimator=classifier, eps=0.2, targeted=True, batch_size=64)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]

for eps in eps_range:
    attack_fgsm.set_params(**{'eps': eps})
    print(f"Eps: {eps}")
    x_test_adv = attack_fgsm.generate(x_test, t_classes)
    loss, accuracy = model.evaluate(x_test_adv, y_test)
    print(f"Adv Loss: {loss}")
    print(f"Adv Accuracy: {accuracy}")
    loss, accuracy = model.evaluate(x_test, y_test)
    print(f"True Loss: {loss}")
    print(f"True Accuracy: {accuracy}")
```

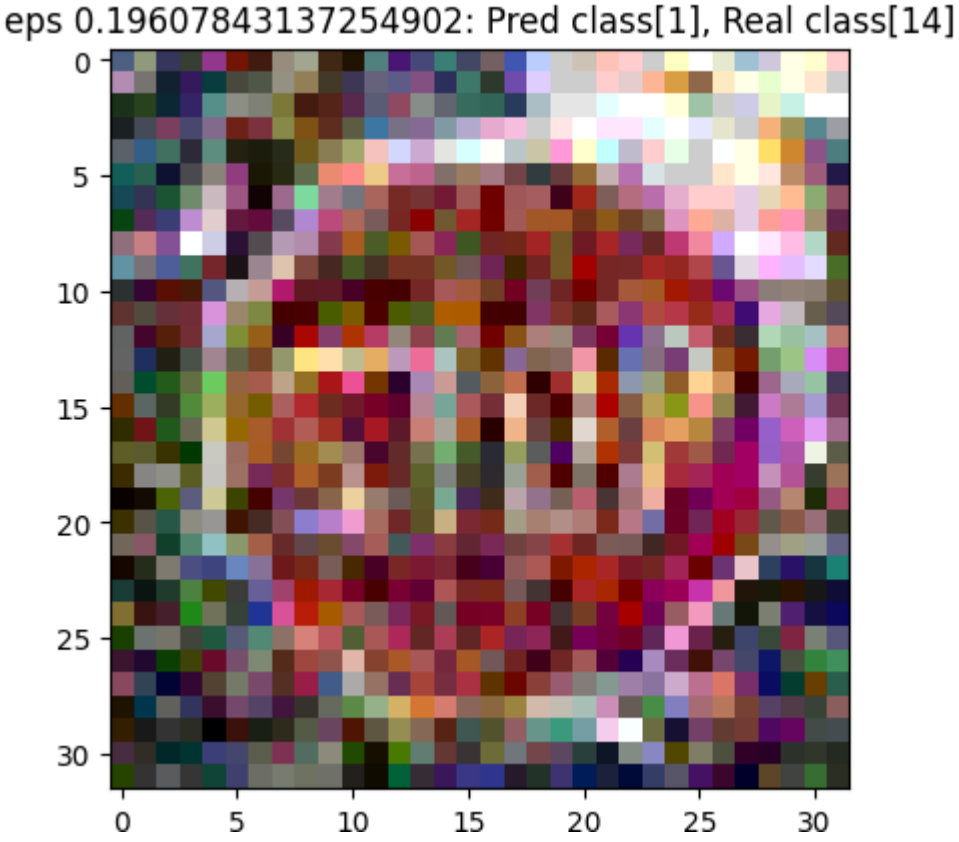

Original img: Pred class[14], Real calss[14]
STOP

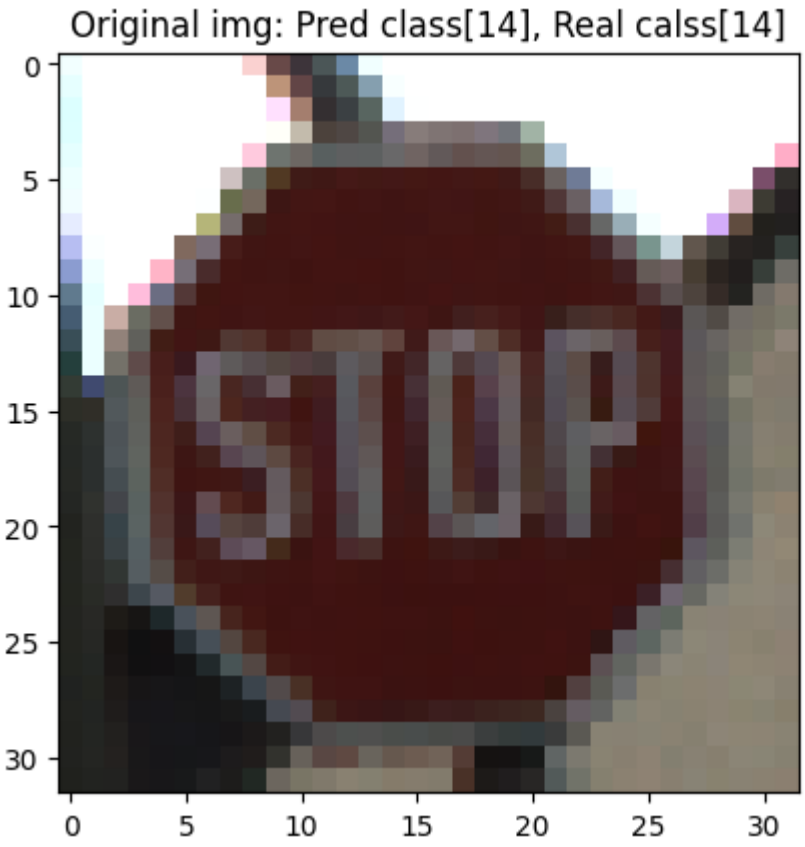

eps 0.19607843137254902: Pred class[1], Real class[14]

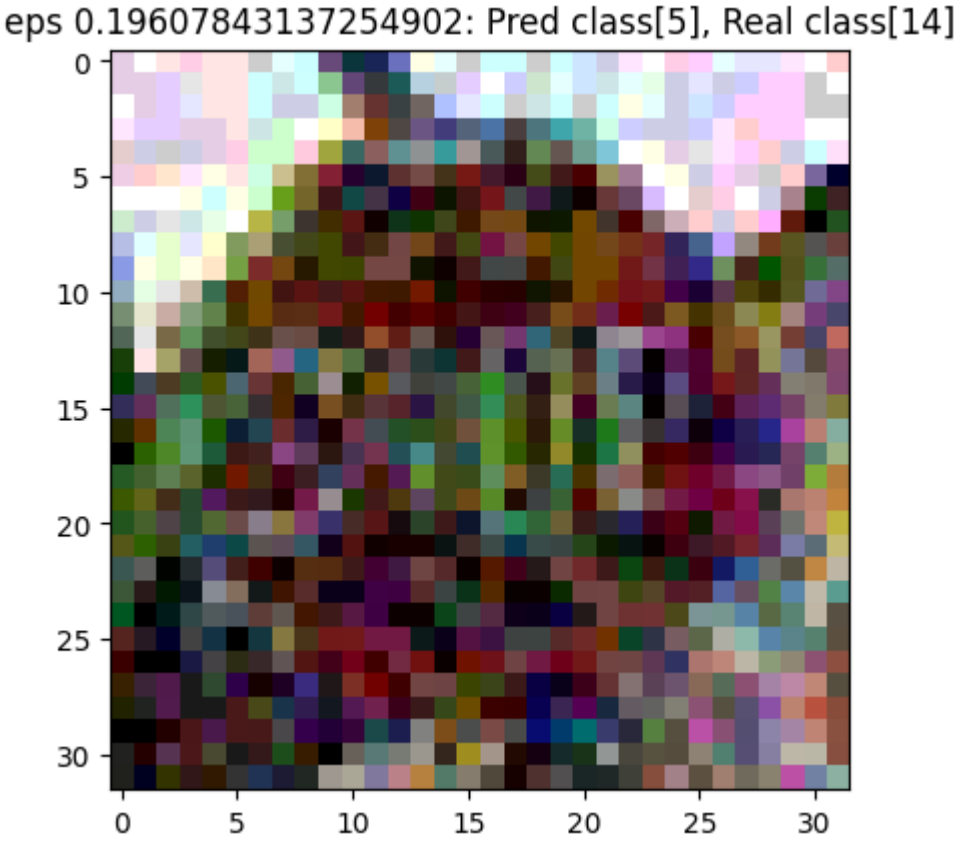

Original img: Pred class[14], Real calss[14]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

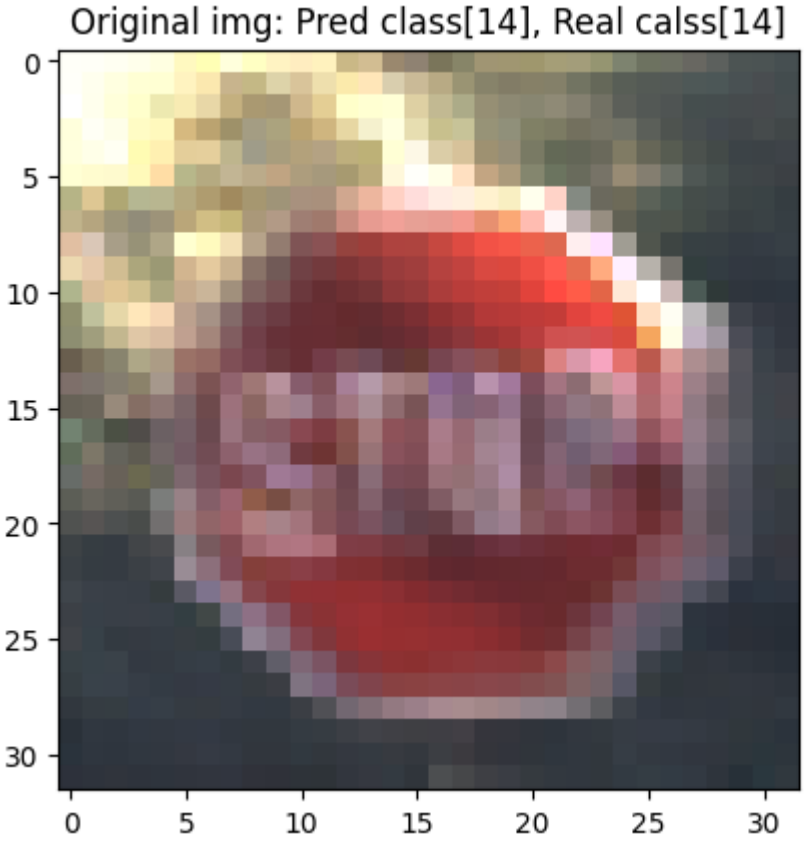

eps 0.19607843137254902: Pred class[1], Real class[14]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

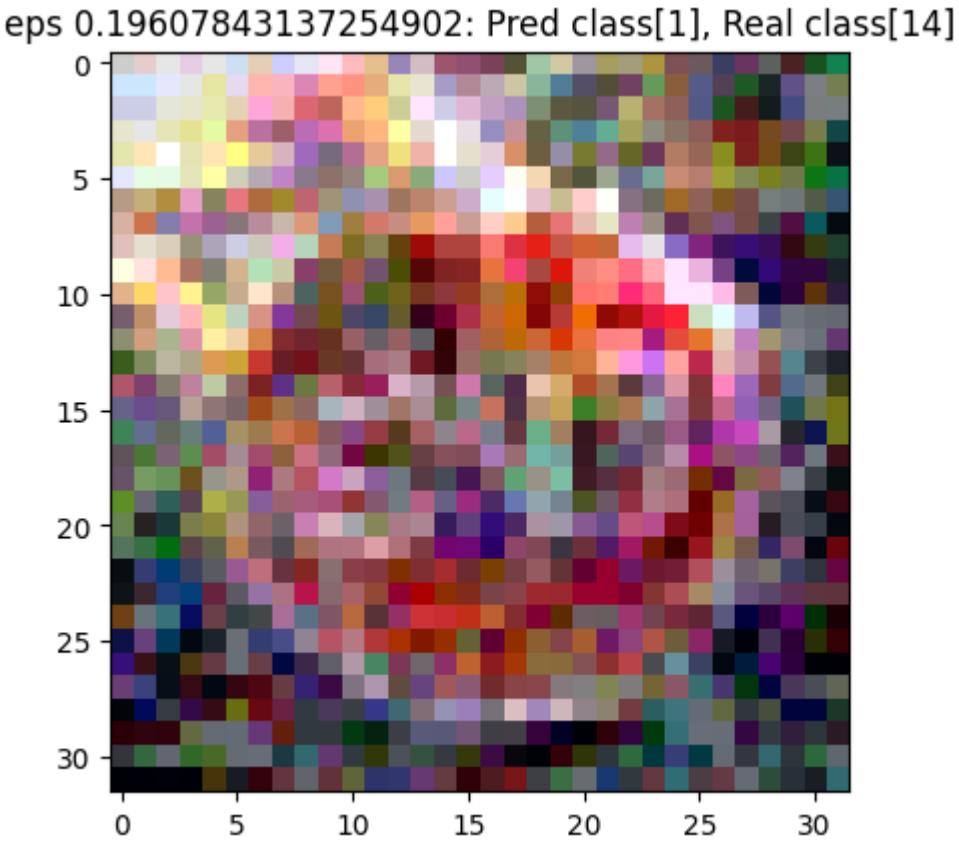

Original img: Pred class[14], Real calss[14]
STOP


eps 0.19607843137254902: Pred class[1], Real class[14]


Original img: Pred class[14], Real calss[14]
STOP


eps 0.19607843137254902: Pred class[5], Real class[14]

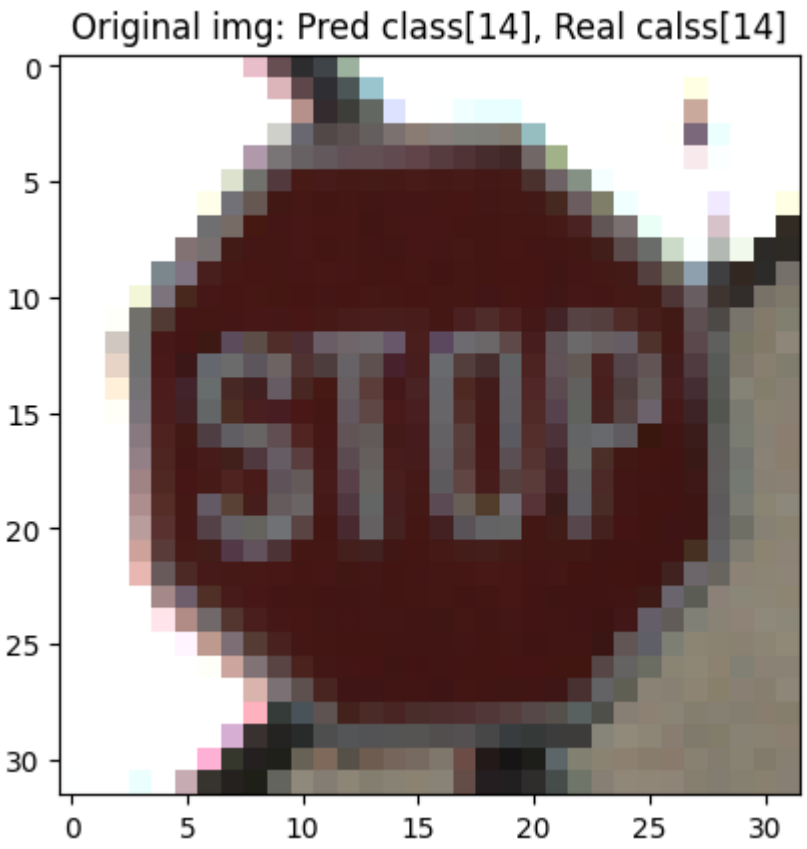

Original img: Pred class[14], Real calss[14]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

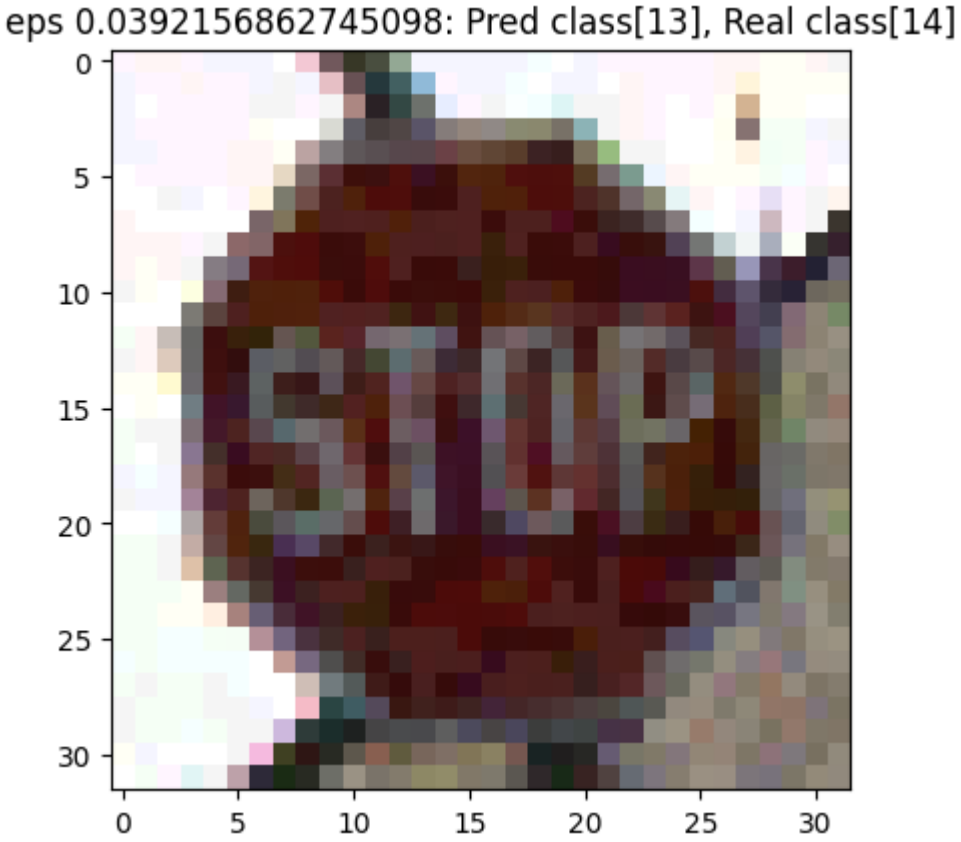

eps 0.19607843137254902: Pred class[1], Real class[14]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

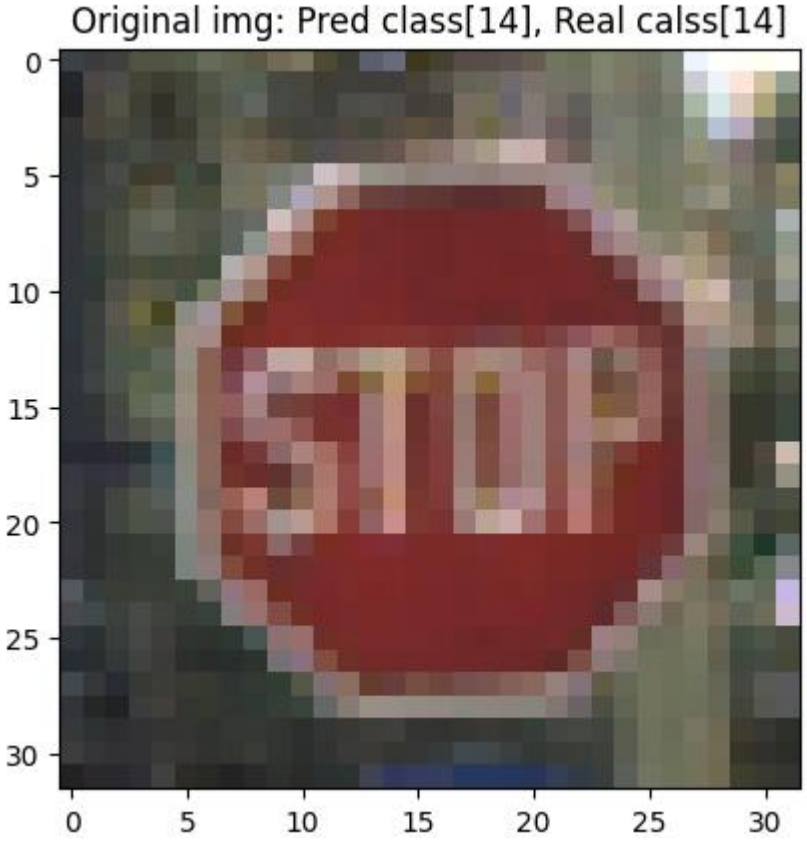
Атака PGD на знак Стоп

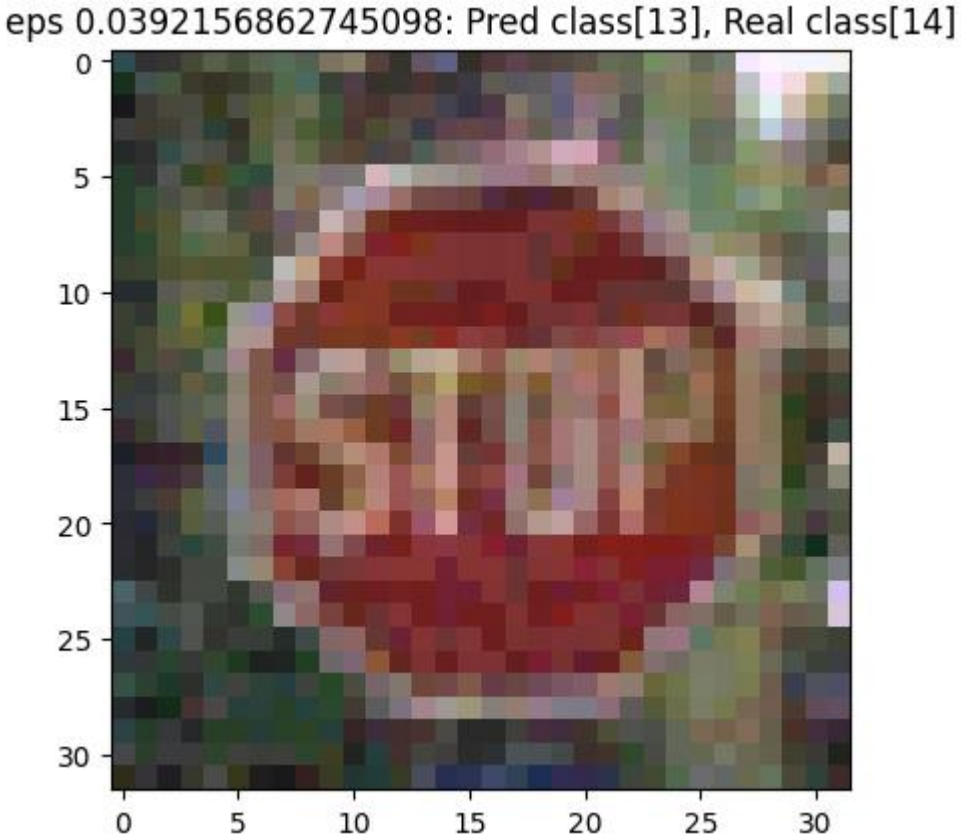
Атака PGD

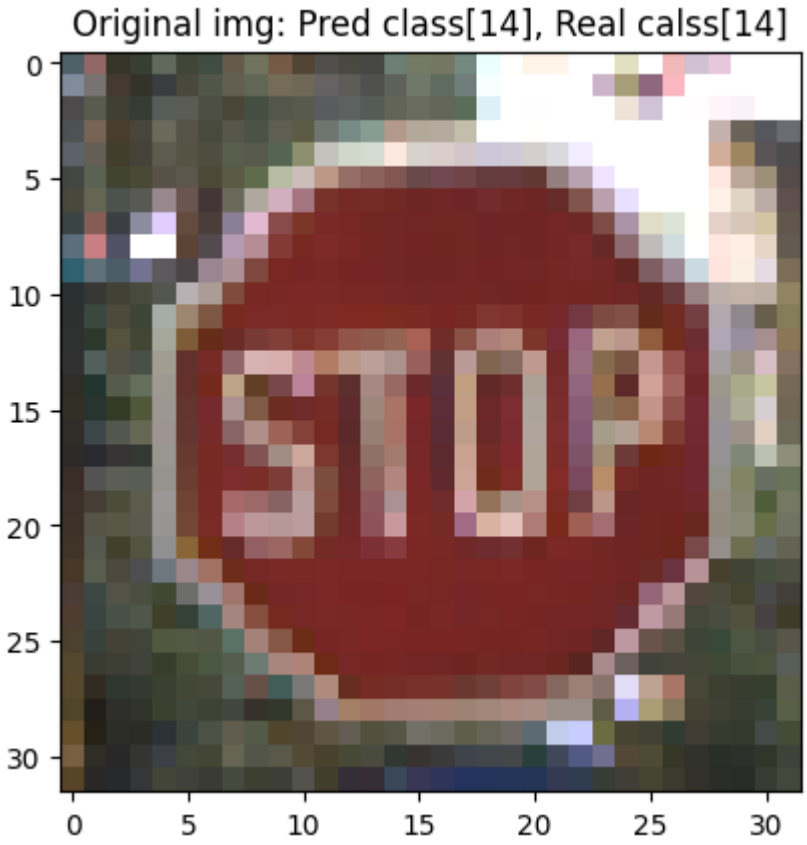
```
model=load_model('ResNet50.h5')
classifier = KerasClassifier(model=model, clip_values=(np.min(x_test), np.max(x_test)))
attack_pgd = ProjectedGradientDescent(estimator=classifier, eps=0.3, max_iter=4, verbose=False, targeted=True)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]

for eps in eps_range:
    attack_pgd.set_params(**{'eps': eps})
    print(f"Eps: {eps}")
    x_test_adv = attack_pgd.generate(x_test, t_classes)
    loss, accuracy = model.evaluate(x_test_adv, y_test)
    print(f"Adv Loss: {loss}")
    print(f"Adv Accuracy: {accuracy}")
    loss, accuracy = model.evaluate(x_test, y_test)
    print(f"True Loss: {loss}")
    print(f"True Accuracy: {accuracy}")
```

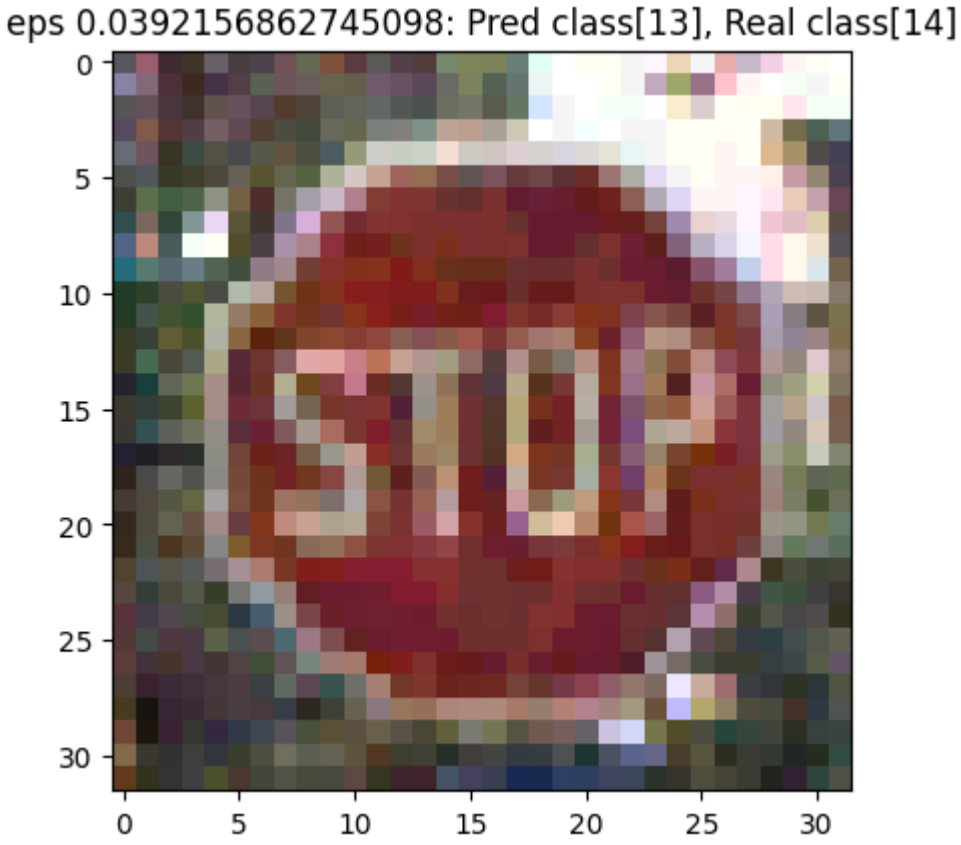

Original img: Pred class[14], Real calss[14]
STOP

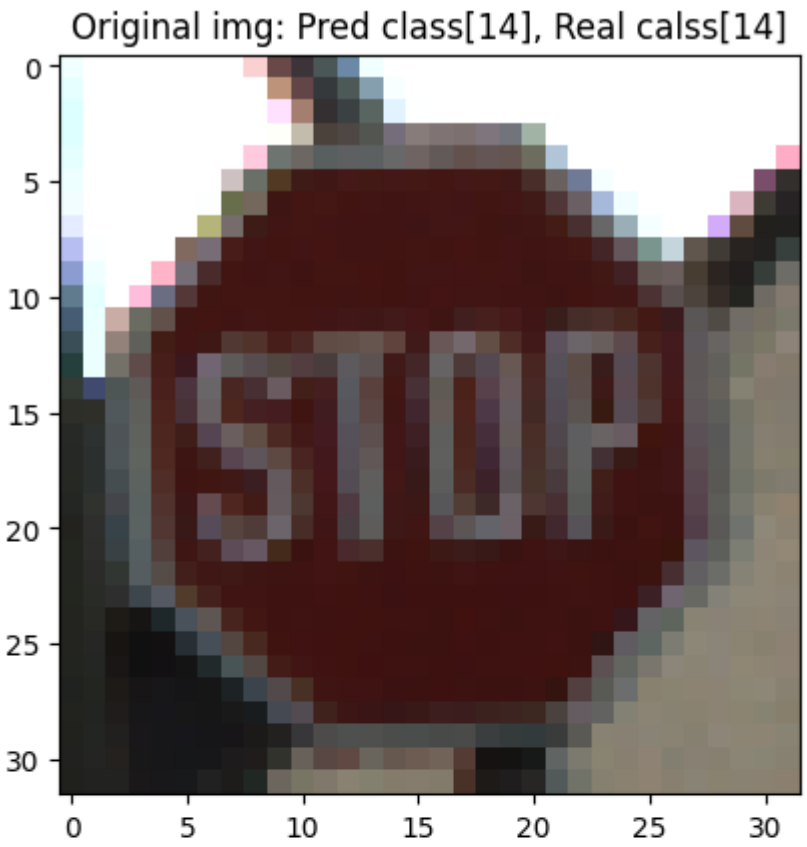

eps 0.0392156862745098: Pred class[13], Real class[14]
STOP

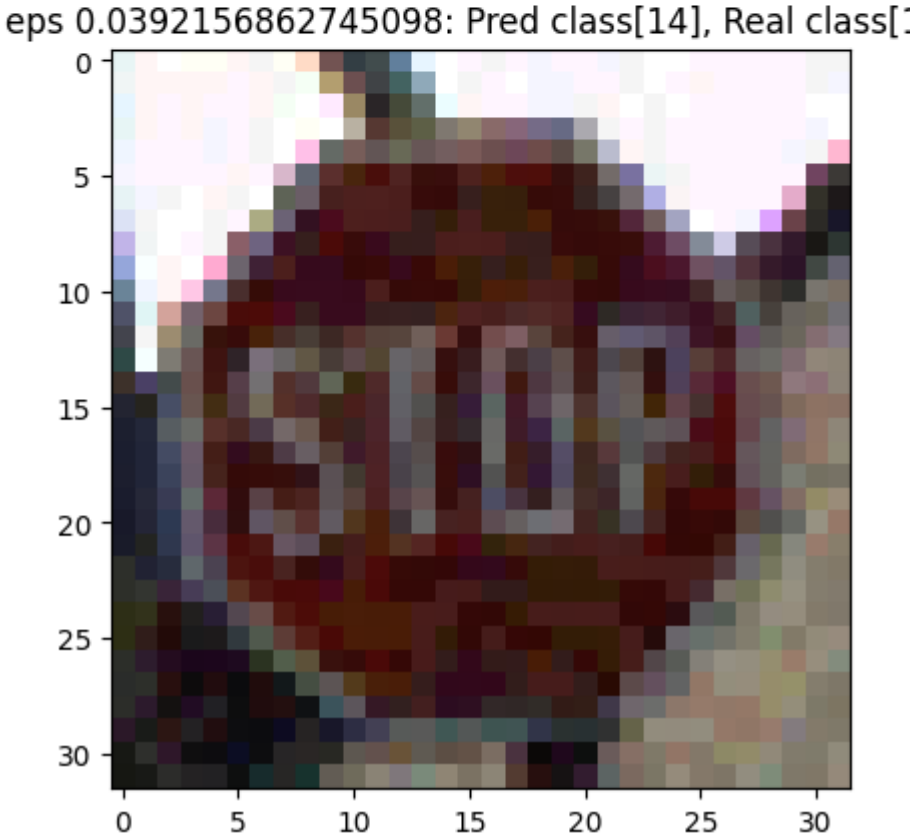

Original img: Pred class[14], Real calss[14]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

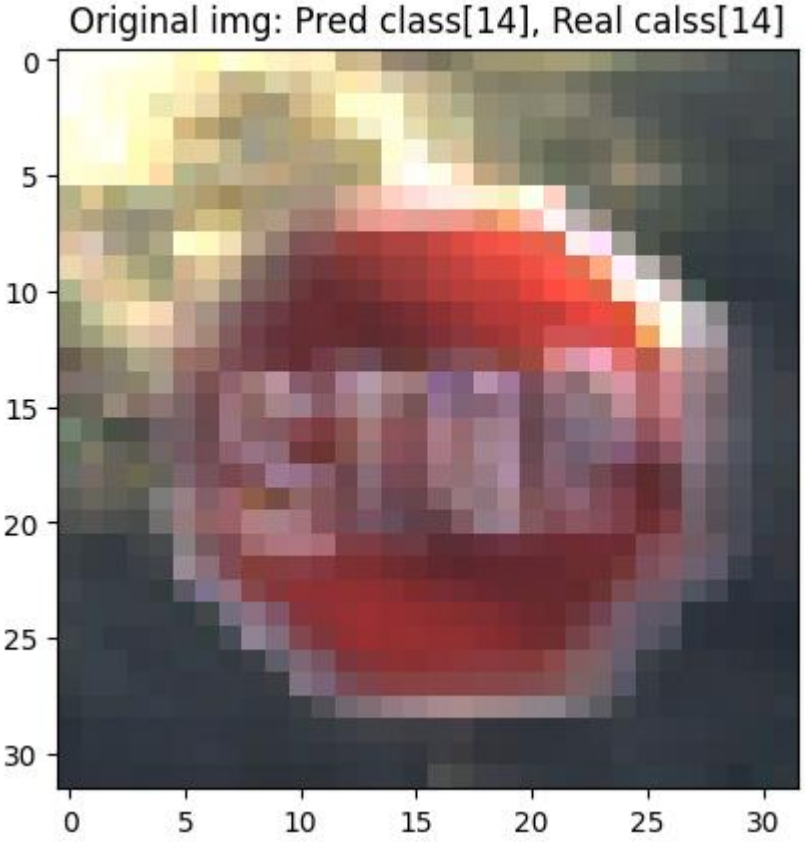

eps 0.0392156862745098: Pred class[13], Real class[14]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

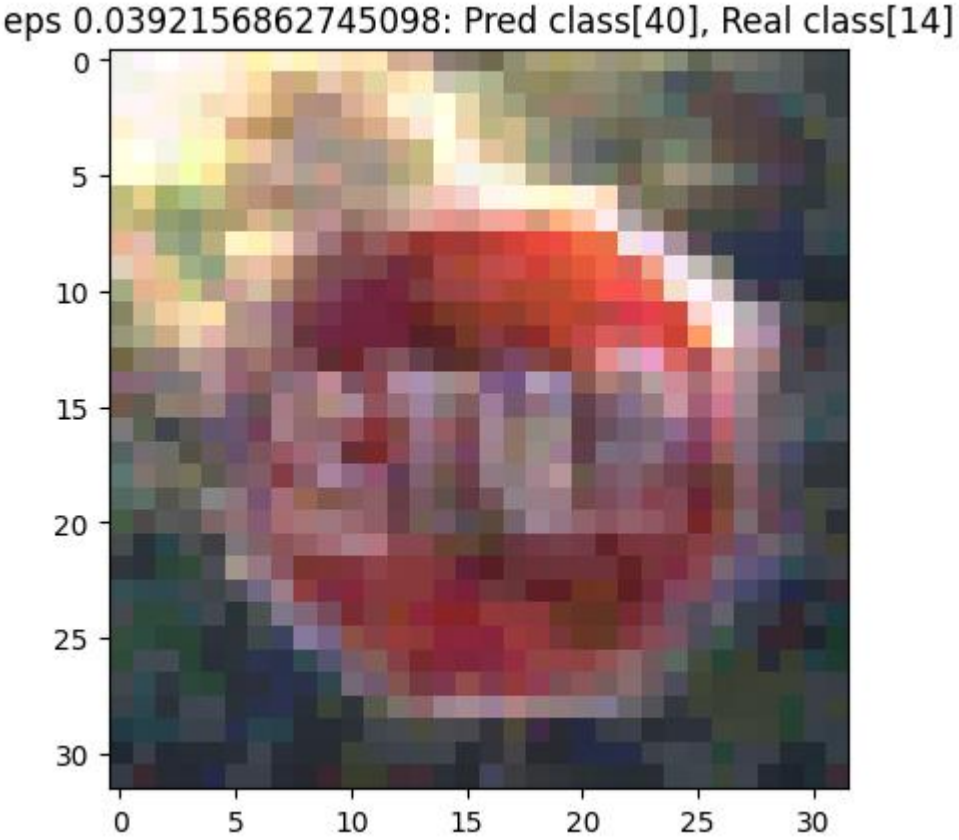

Original img: Pred class[14], Real calss[14]
STOP


eps 0.0392156862745098: Pred class[13], Real class[14]
STOP


Original img: Pred class[14], Real calss[14]
STOP


eps 0.0392156862745098: Pred class[14], Real class[
STOP


Original img: Pred class[14], Real calss[14]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


eps 0.0392156862745098: Pred class[40], Real class[14]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

Искажение	FGSM	PGD
ϵ=1/255	85%	97%
ϵ=2/255	76%	93%
ϵ=3/255	62%	87%
ϵ=4/255	52%	86%
ϵ=5/255	45%	79%
ϵ=8/255	19%	76%
ϵ=10/255	13%	75%
ϵ=20/255	4%	38%
ϵ=50/255	1%	5%
ϵ=80/255	1%	3%