МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Отчёт по лабораторной работе № 1
По дисциплине

«Анализ защищенности систем искусственного интеллекта»

Студент

Лысак Ярослав Денисович

Группа

ББМО-01-22

Работу проверил

Спирин А.А.

Москва, 2023

Цель лабораторной работы

Цель данной лабораторной работы – провести эксперимент по оценки стойкости модели и LeNet к FGSM и DeepFool атакам на основе набора данных CIFAR-10. В процессе работы необходимо провести вычислительный эксперимент в среде Python, следуя инструкции, затем предоставить оценку и вывод по проведенной работе.

Этапы проведения эксперимента

1. Установка зависимостей и набора данных для проведения эксперимента.

```
!git clone https://github.com/ewatson2/EEL6812_DeepFool_Project
!cd EEL6812_DeepFool_Project

!pip install numpy torch torchvision matplotlib pandas

import numpy as np
import json, torch
from torch.utils.data import DataLoader, random_split
from torchvision import datasets, models
from torchvision.transforms import transforms

from models.project_models import FC_500_150, LeNet_CIFAR, LeNet_MNIST, Net
from utils.project_utils import get_clip_bounds, evaluate_attack,
display_attack
```

2. Установка параметров для проведения эксперимента.

```
mnist_mean, mnist_std, mnist_dim = 0.5, 0.5, 28

mnist_min, mnist_max = get_clip_bounds(mnist_mean, mnist_std, mnist_dim)

mnist_min = mnist_min.to(use_device)
mnist_max = mnist_max.to(use_device)

mnist_tf = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(
        mean=mnist_mean,
        std=mnist_std)])

mnist_tf_train = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=mnist_mean,
        std=mnist_std)])

mnist_tf_inv = transforms.Compose([
    transforms.Normalize(
        mean=0.0,
        std=np.divide(1.0, mnist_std)),
    transforms.Normalize(
        mean=np.multiply(-1.0, mnist_std),
        std=1.0)])

mnist_temp = datasets.MNIST(root='datasets/mnist', train=True, download=True,
transform=mnist_tf_train)
mnist_train, mnist_val = random_split(mnist_temp, [50000, 10000])

mnist_test = datasets.MNIST(root='datasets/mnist', train=False, download=True,
transform=mnist_tf)
```

3. Запуск нескольких экспериментов и проверка результатов.

```
cifar_classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',
'frog', 'horse', 'ship', 'truck']

batch_size = 64
workers = 8
fgsm_eps = 0.6

deep_args = [5, len(cifar_classes), 0.05, 50]

mnist_loader_train = DataLoader(mnist_train, batch_size=batch_size,
shuffle=True, num_workers=workers)
mnist_loader_val = DataLoader(mnist_val, batch_size=batch_size, shuffle=False,
num_workers=workers)
mnist_loader_test = DataLoader(mnist_test, batch_size=batch_size,
shuffle=False, num_workers=workers)
cifar_loader_train = DataLoader(cifar_train, batch_size=batch_size,
shuffle=True, num_workers=workers)
cifar_loader_val = DataLoader(cifar_val, batch_size=batch_size, shuffle=False,
num_workers=workers)
cifar_loader_test = DataLoader(cifar_test, batch_size=batch_size,
shuffle=False, num_workers=workers)

model = LeNet_MNIST().to(use_device)

model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth',
map_location=use_device))

evaluate_attack('mnist_lenet_fgsm.csv', 'results', use_device, model,
mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)

print('-------------------------------------------------')

evaluate_attack('mnist_lenet_deepfool.csv', 'results', use_device, model,
mnist_loader_test, mnist_min, mnist_max, deep_args, is_fgsm=False)
```

4. Получение результатов:

```
model = LeNet_MNIST().to(use_device)
# model = FC_500_150().to(use_device)

model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth',
map_location=use_device))

evaluate_attack('mnist_lenet_fgsm.csv', 'results', use_device, model,
mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)

print('-------------------------------------------------')

evaluate_attack('mnist_lenet_deepfool.csv', 'results', use_device, model,
mnist_loader_test, mnist_min, mnist_max, deep_args, is_fgsm=False
```

Результаты эксперимента

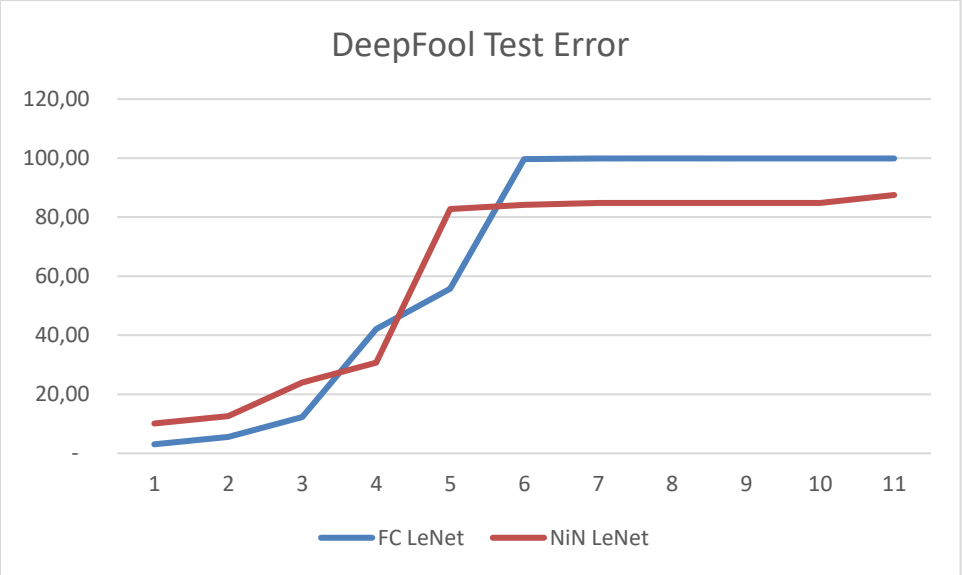
В результате проведенных экспериментов были получены следующие результаты:

Таблица 1.

#	fgsm_eps	DeepFool Test Error	
		FC LeNet	NiN LeNet
1	0,000	3,08	10,12
2	0,002	5,54	12,63
3	0,020	12,31	23,98
4	0,200	42,10	30,76
5	0,500	55,78	82,76
6	0,750	99,62	84,21
7	0,900	99,87	84,76
8	1,000	99,89	84,76
9	2,000	99,84	84,79
10	5,000	99,88	84,78
11	10,000	99,87	87,50

Также полученные данные отображены на рисунке 1.

Рисунок 1.


DeepFool Test Error
120,00
100,00
80,00
60,00
40,00
20,00
-
1
2
3
4
5
6
7
8
9
10
11
FC LeNet
NiN LeNet

Выводы

В результате выполнения лабораторной работы и проведения нескольких вычислительных экспериментов было продемонстрировано , что значения fgsm_eps влияют на стойкость сетей к атакам.

На примере из таблицы 1 показано, что при увеличении значения fgsm_eps сети становятся более уязвимыми к атакам и допускают большее количство ошибок классификации.

Используя набор данных CIFAR для сети LeNet FC и NIN наблюдается увеличения числа ошибок при увеличении значения параметра fgsm_eps. Стойкость сети падает значительно.