Задание на практику № 2. АЗСИИ.

1. Скопировать проект по ссылке в локальную среду выполнения Jupyter (Google Colab) https://github.com/ewatson2/EEL6812_DeepFool_Project

2. Сменить директорию исполнения на вновь созданную папку "EEL6812_DeepFool_Project" проекта.

3. Выполнить импорт библиотек: numpy as np import json, torch from torch.utils.data import DataLoader, random_split from torchvision import datasets, models from torchvision.transforms import transforms

4. Выполнить импорт вспомогательных библиотек из локальных файлов проекта:

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

5. Установить случайное рандомное значение в виде переменной rand_seed={"Порядковый номер ученика группы в Гугл-таблице"}
6. Установить указанное значение для np.random.seed и torch.manual_seed
7. Использовать в качестсве устройства видеокарту (Среды выполнения--> Сменить среду выполнения --> T4 GPU)
8. Загрузить датасет MNIST с параметрами mnist_mean = 0.5, mnist_std = 0.5, mnist_dim = 28

mnist_min, mnist_max = get_clip_bounds(mnist_mean, mnist_std, mnist_dim) mnist_min = mnist_min.to(device) mnist_max = mnist_max.to(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)

9. Загрузить датасет CIFAR-10 с параметрами cifar_mean = [0.491, 0.482, 0.447] cifar_std = [0.202, 0.199, 0.201] cifar_dim = 32

cifar_min, cifar_max = get_clip_bounds(cifar_mean, cifar_std, cifar_dim) cifar_min = cifar_min.to(device) cifar_max = cifar_max.to(device)

cifar_tf = transforms.Compose([transforms.ToTensor(), transforms.Normalize(mean=cifar_mean, std=cifar_std)])

cifar_tf_train = transforms.Compose([transforms.RandomCrop(size=cifar_dim, padding=4), transforms.RandomHorizontalFlip(), transforms.ToTensor(), transforms.Normalize(mean=cifar_mean, std=cifar_std)])

cifar_tf_inv = transforms.Compose([transforms.Normalize(mean=[0.0, 0.0, 0.0], std=np.divide(1.0, cifar_std)), transforms.Normalize(mean=np.multiply(-1.0, cifar_mean), std=[1.0, 1.0, 1.0])])

cifar_temp = datasets.CIFAR10(root='datasets/cifar-10', train=True, download=True, transform=cifar_tf_train) cifar_train, cifar_val = random_split(cifar_temp, [40000, 10000])

cifar_test = datasets.CIFAR10(root='datasets/cifar-10', train=False, download=True, transform=cifar_tf)

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

10. Выполнить настройку и загрузку DataLoader batch_size = 64 workers = 4

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)

11. Загрузить и оценить стойкость модели LeNet к FGSM и DeepFool атакам

fgsm_eps = 0.6 model = LeNet_MNIST().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth', map_location=torch.device('cpu')))

evaluate_attack('mnist_lenet_fgsm.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True) print('') evaluate_attack('mnist_lenet_deepfool.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, deep_args, is_fgsm=False)

if device.type == 'cuda': torch.cuda.empty_cache()

12. Загрузить и оценить стойкость модели FC к FGSM и DeepFool атакам

fgsm_eps = 0.2 model = FC_500_150().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth', map_location=torch.device('cpu')))

evaluate_attack('mnist_fc_fgsm.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True) print('') evaluate_attack('mnist_fc_deepfool.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, deep_args, is_fgsm=False)

if device.type == 'cuda': torch.cuda.empty_cache()