ДОДАТОК А

Метрики навчання

Таблиця А.1 – Точність класифікаторів під час атаки

Epsilon	**MaxPool**	**LPPool**	**GLPPool**	***Difference***
1,00E-04	83,94	83,21	**84,31**	*0,37*
2,87E-03	60,71	62,58	**66,35**	*5,64*
5,64E-03	39	42,24	**46,31**	*7,31*
8,40E-03	19,51	23,53	**26,36**	*6,85*
0,01117	7,963	10,42	**13,11**	*5,147*
0,01393	2,885	4,157	**5,99**	*3,105*
0,0167	0,9415	1,552	**2,594**	*1,6525*
0,01947	0,37	0,5909	**1,332**	*0,962*
0,02223	0,2	0,3	**0,6611**	*0,4611*
0,025	0,08	0,13	**0,3**	*0,22*

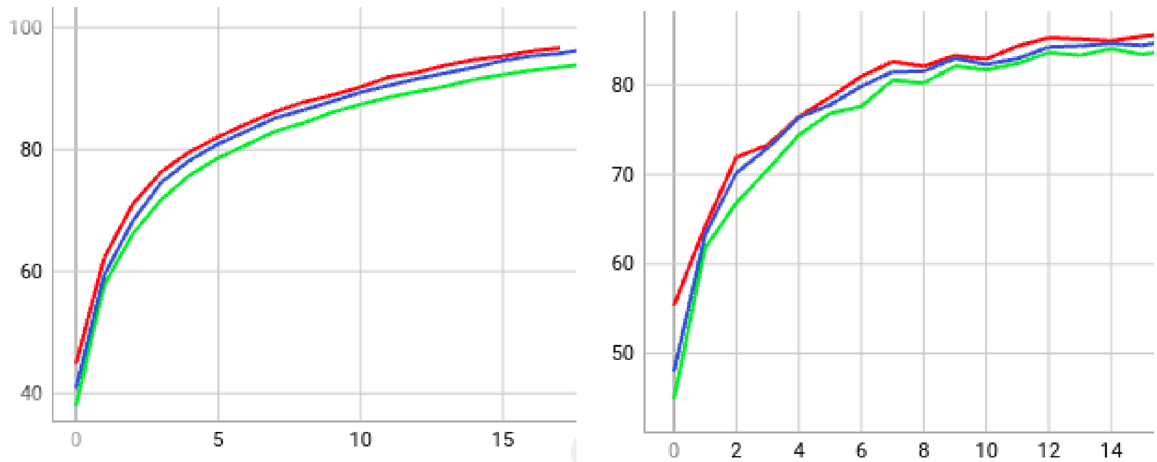

Рисунок А.1 – Графік точності класифікаторів на чистих даних

Синя лінія позначає класифікатор з використанням MaxPooling, зелена лінія – модель із застосуванням LPPooling, червона лінія – модель з Generalized Lehmer Pooling.

Зліва наведено графік точності під час тренування, права частина – валідація.

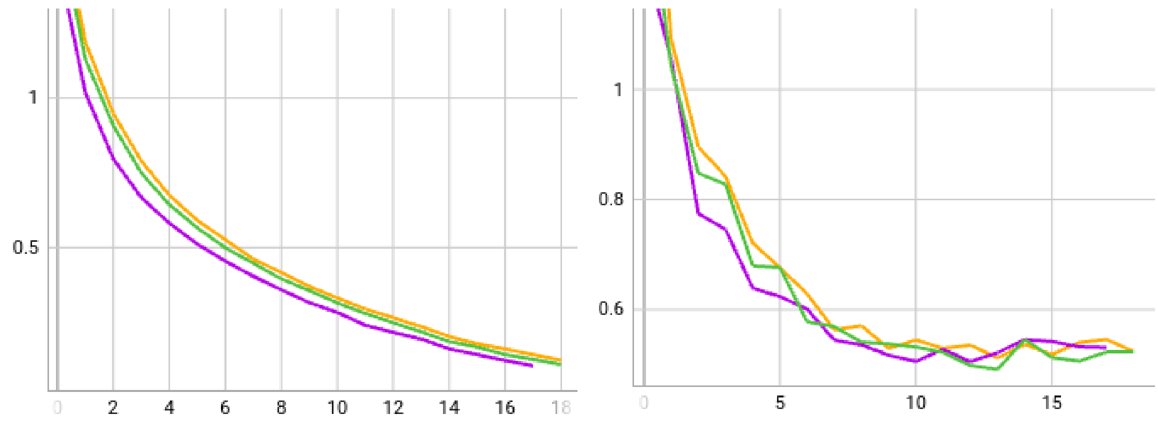

Рисунок А.2 – Графіки функції втрат для моделей з одним GLP

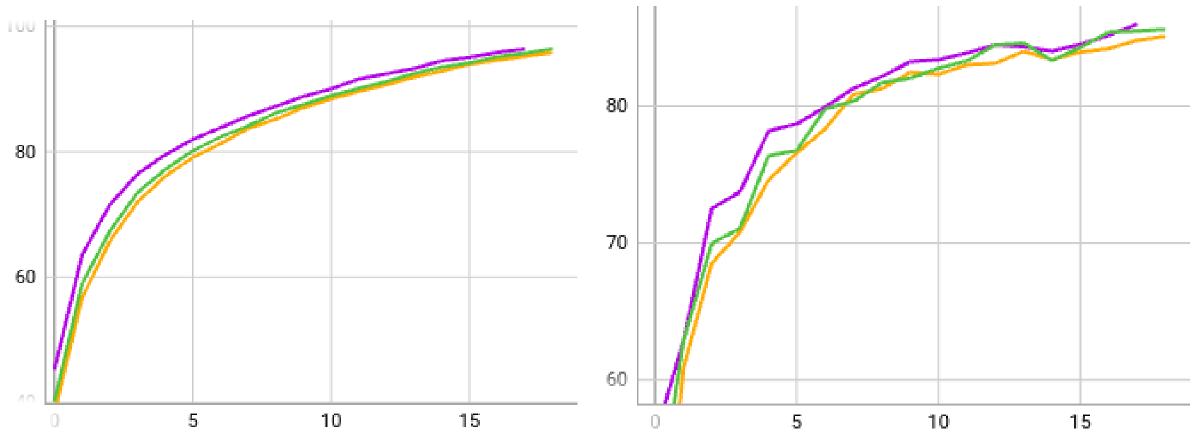

Рисунок А.3 – Графіки точності для моделей з одним GLP

На рисунках А.2 та А.3 зображено функції втрат та точності моделей відповідно. Правий графік відповідає метриці на тренувальному датасеті, лівий – на валідаційному. Фіолетова лінія позначає модель з прошарком субдискретизації у останньому блоці, зелена – у середньому блоці, жовта – у першому блоці.

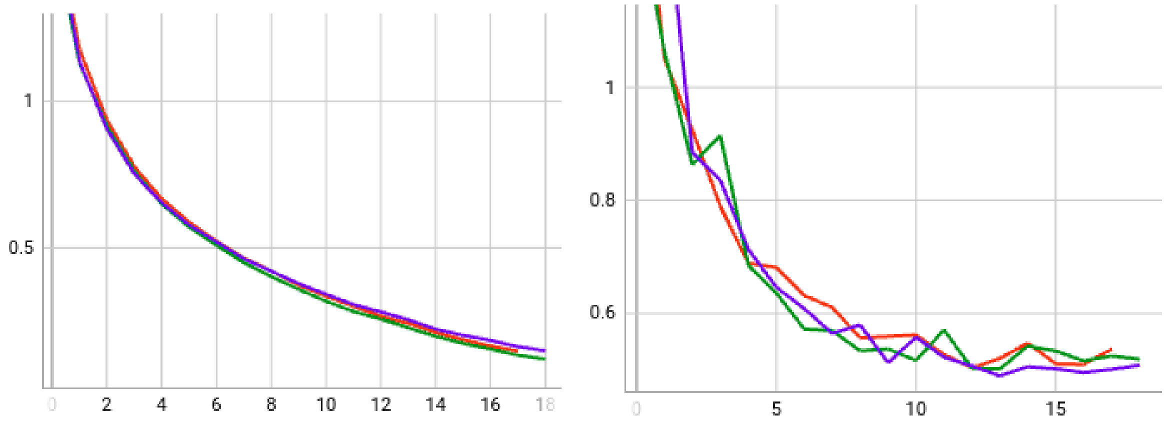

Рисунок А.4 – Графіки функції втрат для моделей з одним LPPool

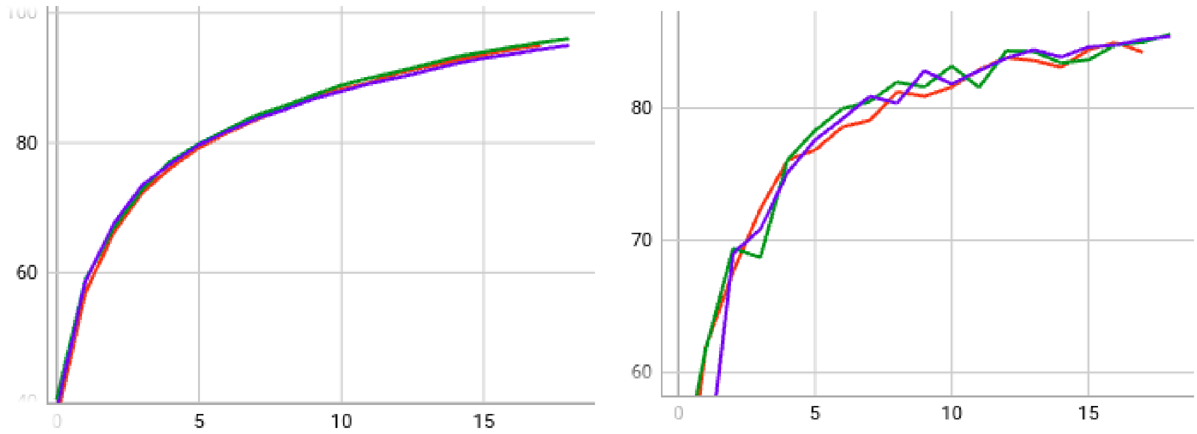

Рисунок А.5 – Графіки точності для моделей з одним LPPool

На рисунках А.4 та А.5 зображено функції втрат та точності моделей відповідно. Правий графік відповідає метриці на тренувальному датасеті, лівий – на валідаційному. Фіолетова лінія позначає модель з прошарком субдискретизації у останньому блоці, зелена – у середньому блоці, помаранчева – у першому блоці.

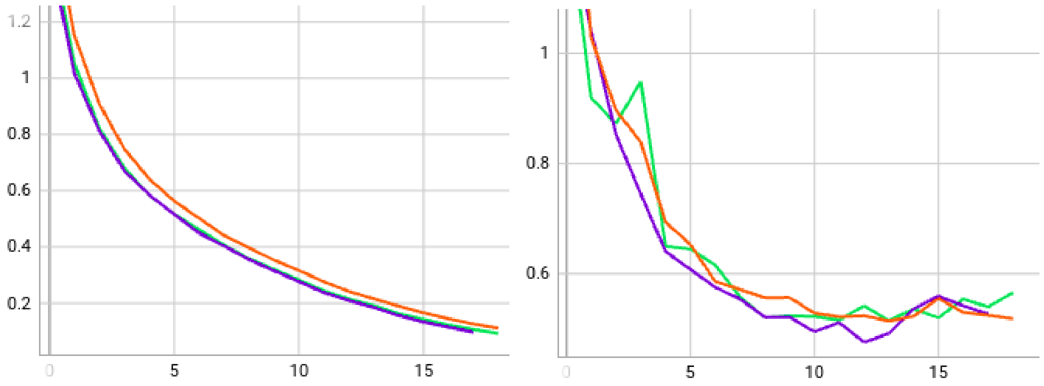

Рисунок А.6 – Графіки функції втрат для моделей з двома GLP

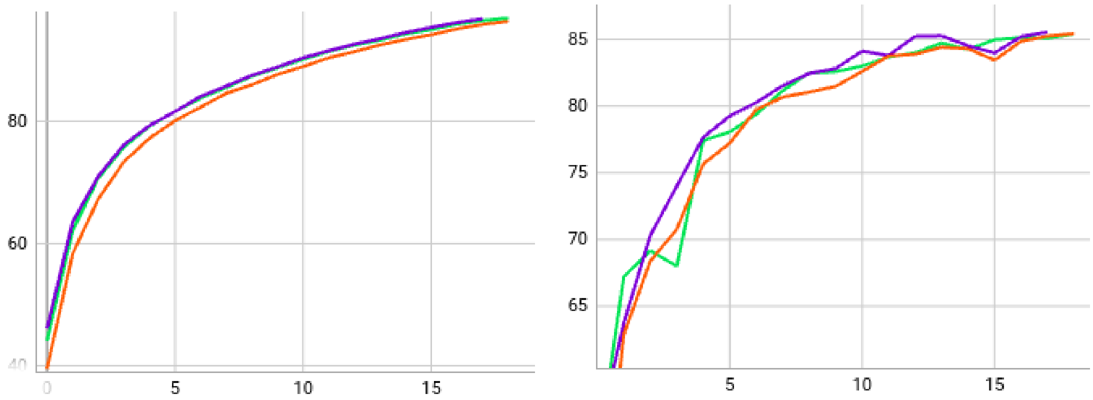

Рисунок А.7 – Графіки точності для моделей двома GLP

На рисунках А.6 та А.7 зображено функції втрат та точності моделей відповідно. Правий графік відповідає метриці на тренувальному датасеті, лівий – на валідаційному. Фіолетова лінія позначає модель з прошарком субдискретизації у блоках 2 та 3, зелена – у блоках 1 та 3, помаранчева – у блоках 1 та 2.

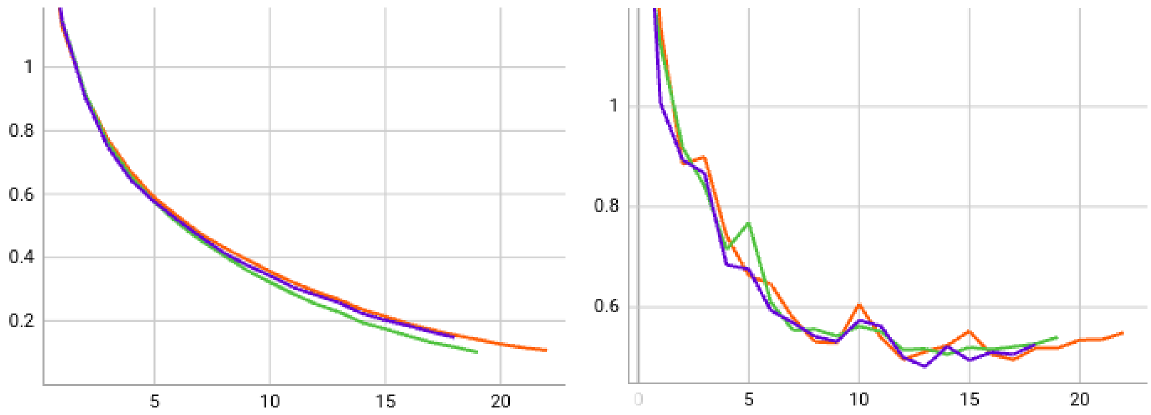

Рисунок А.8 – Графіки функції втрат для моделей з двома LPPool

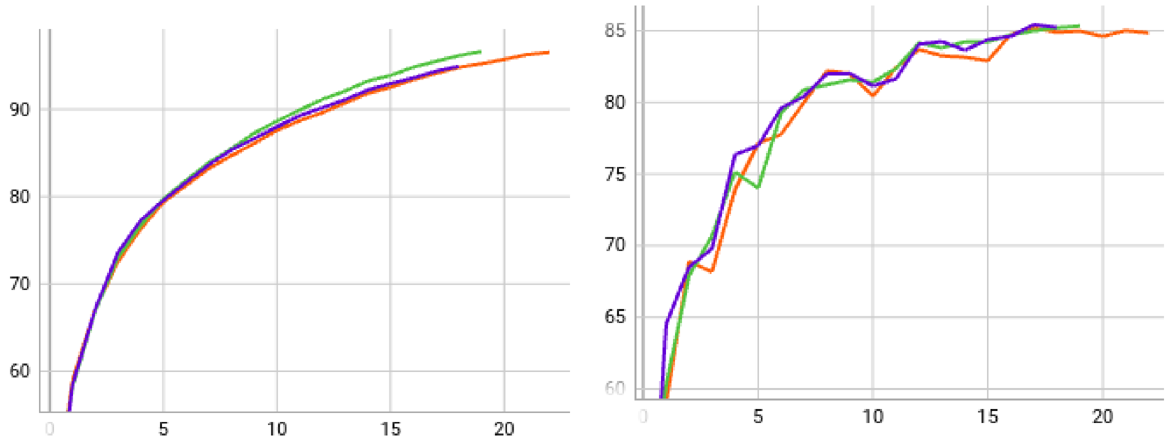

Рисунок А.9 – Графіки точності для моделей двома GLP

На рисунках А.8 та А.9 зображено функції втрат та точності моделей відповідно. Правий графік відповідає метриці на тренувальному датасеті, лівий – на валідаційному. Фіолетова лінія позначає модель з прошарком субдискретизації у блоках 1 та 3, зелена – у блоках 1 та 2, помаранчева – у блоках 2 та 3.

ДОДАТОК Б

Код програми

Лістинг Б.1 – Програмна реалізація класифікатору зображень

```
class Net(nn.Module):
    def __init__(self, pool: List):
        super(Net, self).__init__()
        self.pool1 = pool[0](**pool[1])
        self.pool2 = pool[0](**pool[1])
        self.pool3 = pool[0](**pool[1])

        self.drop2 = nn.Dropout2d(p=0.15)

        self.block1 = nn.Sequential(
            nn.Conv2d(in_channels=3, out_channels=32,
                      kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(in_channels=32, out_channels=64,
                      kernel_size=3, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
        )
        self.block2 = nn.Sequential(
            nn.Conv2d(in_channels=64, out_channels=128,
                      kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(in_channels=128, out_channels=128,
                      kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
        )
        self.block3 = nn.Sequential(
            nn.Conv2d(in_channels=128, out_channels=256,
                      kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(in_channels=256, out_channels=256,
                      kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
        )
        self.fc_layer = nn.Sequential(
            nn.Linear(4096, 1024),
            nn.ReLU(),
            nn.Linear(1024, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.pool1(self.block1(x))
        x = self.drop2(self.pool2(self.block2(x)))
        x = self.pool3(self.block3(x))
        x = x.view(x.size(0), -1)
        x = self.fc_layer(x)
        return x
```

Лістинг Б.2 – Програмна реалізація Generalized Lehmer Pooling

```
class GeneralizedLehmerPool2d(nn.Module):
    def __init__(self, alpha: float, beta: float, kernel_size, stride,
padding=0, dilation=1):
        super().__init__()
        self.kernel_size = _pair(kernel_size)
        self.stride = _pair(stride)
        self.padding = padding
        self.dilation = _pair(dilation)
        self.alpha = Parameter(torch.tensor(
            alpha, dtype=torch.float64, requires_grad=True))
        self.beta = Parameter(torch.tensor(
            beta, dtype=torch.float64, requires_grad=True))

    def forward(self, input: Tensor) -> Tensor:
        return generalized_lehmer_pooling(input, self.alpha, self.beta,
                                          self.kernel_size,self.stride,
                                          self.padding, self.dilation)
```

Лістинг Б.3 – Програмна реалізація LP Pooling

```
class _LPPool2d(nn.Module):
    def __init__(self, norm_type: float, kernel_size, stride=None,
                 ceil_mode: bool = False) -> None:
        super(_LPPoolNd, self).__init__()
        self.norm_type = Parameter(torch.tensor(
            norm_type, dtype=torch.float64, requires_grad=True))
        self.kernel_size = kernel_size
        self.stride = stride
        self.ceil_mode = ceil_mode

    def forward(self, input: Tensor) -> Tensor:
        return F.lp_pool2d(input, self.norm_type, self.kernel_size,
                           self.stride, self.ceil_mode)
```

Лістинг Б.4 – Код функції clip_pooling для обробки обмежень параметрів

```
def clip_poolings(model: nn.Module):
    module_types = {key: type(module) for key, module in
model.named_modules()}
    for name, p in model.named_parameters():
        if (module_types[name.split('.')[0]] is
GeneralizedLehmerPool2d):
            if "alpha" in name:
                p.data = clip_data(p, 1.00001, 2.71828)
            if "beta" in name:
                p.data = clip_data(p, -2.5, 1.5)

        if (module_types[name.split('.')[0]] is
GeneralizedPowerMeanPool2d):
            if "gamma" in name:
                p.data = clip_data(p, 1.00001, 2.71828)
            if "delta" in name:
                p.data = clip_data(p, -2.5, 1.5)
```

```
        if (module_types[name.split('.')[0]] is LPPool2d):
            if "norm_type" in name:
                p.data = clip_data(p, 1.00001, 4)
```

Лістинг Б.5 – Код для роботи з даними та моделлю

```
    train_loader, val_loader, test_loader = get_dataloaders(args)

    pool = pools.get(args.pooling_type, [nn.MaxPool2d, {
                     'kernel_size': 2, 'stride': 2}])
    model = Net(pool)
    model.to(device)
    loss_func = torch.nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=args.lr)
    for epoch in range(start_epoch, start_epoch+args.epochs):
        train_losses, train_accs = train(epoch, model, train_loader,
optimizer,
                                         None, loss_func, writer)
        val_losses, val_accs = validate(
            epoch, model, val_loader, loss_func, writer)
```

Лістинг Б.6 – Приклад .sh файлу для запуску серії експериментів

```
python3 main.py --run_id ex_1 --pooling_type max_pool2d --lr 1e-3 --
epochs 40 --batch_size 256

python3 main.py --run_id ex_2 --pooling_type generalized_lehmer_pool --
alpha 2.5 --beta 1.3 --lr 1e-3 --epochs 70 --batch_size 256

python3 main.py --run_id ex_3 --pooling_type generalized_lehmer_pool --
alpha 1.5 --beta 0.1 --lr 1e-3 --epochs 70 --batch_size 256
```

ДОДАТОК В

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	77 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Малик Д. Г.			Гнучкість нейронних мереж як метод змагального захисту	Шифр групи	Код напр./спец.
Перевірив		Терзіяк В. Я.				СШІм-20-3	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					