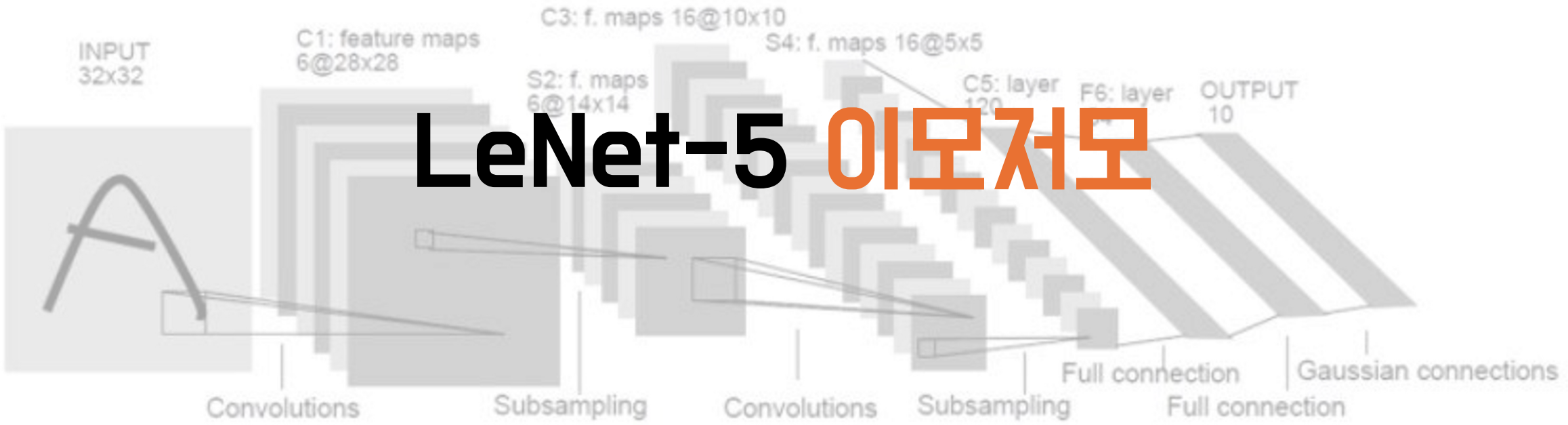
LeNet-5 이모저모


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

기본 모델 : 환경

```
!nvidia-smi
```

```
Wed Jul 24 13:19:12 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.90.07              Driver Version: 550.90.07      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   64C    P0             30W /   70W |     291MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  Tesla T4                       Off |   00000000:00:05.0 Off |                    0 |
| N/A   49C    P0             26W /   70W |     273MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
+-----------------------------------------------------------------------------------------+
```

기본 모델 : 모델 정의

하이퍼파라미터, 모델 구조, 모델 요약

```python
# 하이퍼파라미터 설정
RANDOM_SEED = 4242
BATCH_SIZE = 64
EPOCHS = 20
IMG_SIZE = 32
NUM_CLASSES = 10

# 하이퍼파리미터-학습률 스케줄링 설정
def lr_schedule(epoch):
    if epoch ≤ 2:
        lr = 5e-4
    elif epoch ≤ 5:
        lr = 2e-4
    elif epoch ≤ 9:
        lr = 5e-5
    else:
        lr = 1e-5
    return lr
```

```python
# LeNet5 모델 정의
class LeNet5(nn.Module):

    def __init__(self, num_classes):
        super(LeNet5, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5, stride=1)
        self.pool2 = nn.AvgPool2d(kernel_size=2, stride=2)
        self.conv3 = nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5, stride=1)
        self.pool4 = nn.AvgPool2d(kernel_size=2, stride=2)
        self.conv5 = nn.Conv2d(in_channels=16, out_channels=120, kernel_size=5, stride=1)
        self.fc6 = nn.Linear(120, 84)
        self.fc7 = nn.Linear(84, num_classes)

    def forward(self, x):
        x = F.tanh(self.conv1(x))
        x = self.pool2(x)
        x = F.tanh(self.conv3(x))
        x = self.pool4(x)
        x = F.tanh(self.conv5(x))
        x = x.view(x.size(0), -1)
        x = self.fc6(x)
        x = F.tanh(x)

        logits = self.fc7(x)

        return logits
torchinfo.summary(LeNet5(NUM_CLASSES), input_size=(1, 1, 32, 32))
```

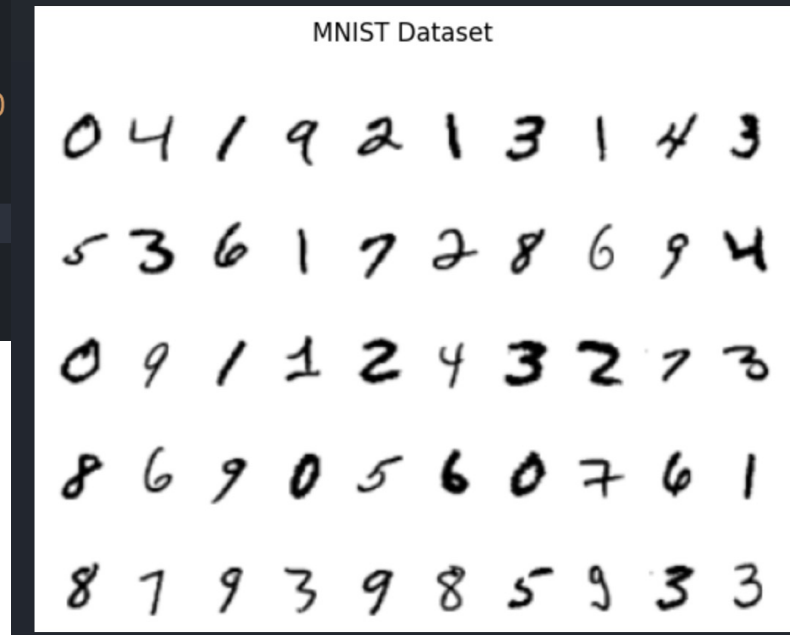
```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
LeNet5                                   [1, 10]                   --
├─Conv2d: 1-1                            [1, 6, 28, 28]            156
├─AvgPool2d: 1-2                         [1, 6, 14, 14]            --
├─Conv2d: 1-3                            [1, 16, 10, 10]           2,416
├─AvgPool2d: 1-4                         [1, 16, 5, 5]             --
├─Conv2d: 1-5                            [1, 120, 1, 1]            48,120
├─Linear: 1-6                            [1, 84]                   10,164
├─Linear: 1-7                            [1, 10]                   850
==========================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
Total mult-adds (M): 0.42
==========================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.05
Params size (MB): 0.25
Estimated Total Size (MB): 0.30
==========================================================================
```

기본 모델 : 데이터 정의

70,000장의 데이터를 모두 모으고 70:15:15 비율로 무작위 분리

```python
# transforms 정의하기
transform = transforms.Compose([transforms.Resize((32, 32)), transforms.ToTensor()])

# 데이터셋 다운로드 및 생성

full_dataset = datasets.MNIST(root="mnist_data", train=True, transform=transform, download=True)
full_dataset = ConcatDataset([full_dataset, datasets.MNIST(root="mnist_data", train=False, transform=transform)])

total_size = len(full_dataset)

train_size = int(0.7 * total_size)
valid_size = int(0.15 * total_size)
test_size = total_size - train_size - valid_size

train_dataset, valid_dataset, test_dataset = random_split(full_dataset, [train_size, valid_size, test_size])

# 데이터 로더 정의
train_loader = DataLoader(dataset=train_dataset, batch_size=BATCH_SIZE, shuffle=True)
valid_loader = DataLoader(dataset=valid_dataset, batch_size=BATCH_SIZE, shuffle=False)
test_loader = DataLoader(dataset=test_dataset, batch_size=BATCH_SIZE, shuffle=False)
```

```
train dataset size:  49000
valid dataset size:  10500
test dataset size:  10500
```

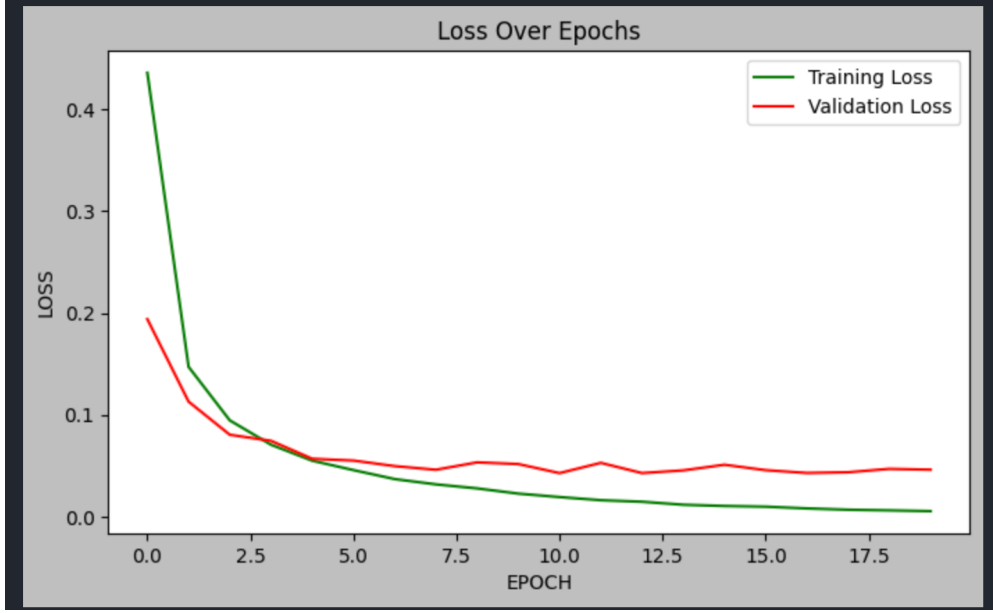


기본 모델 : 학습

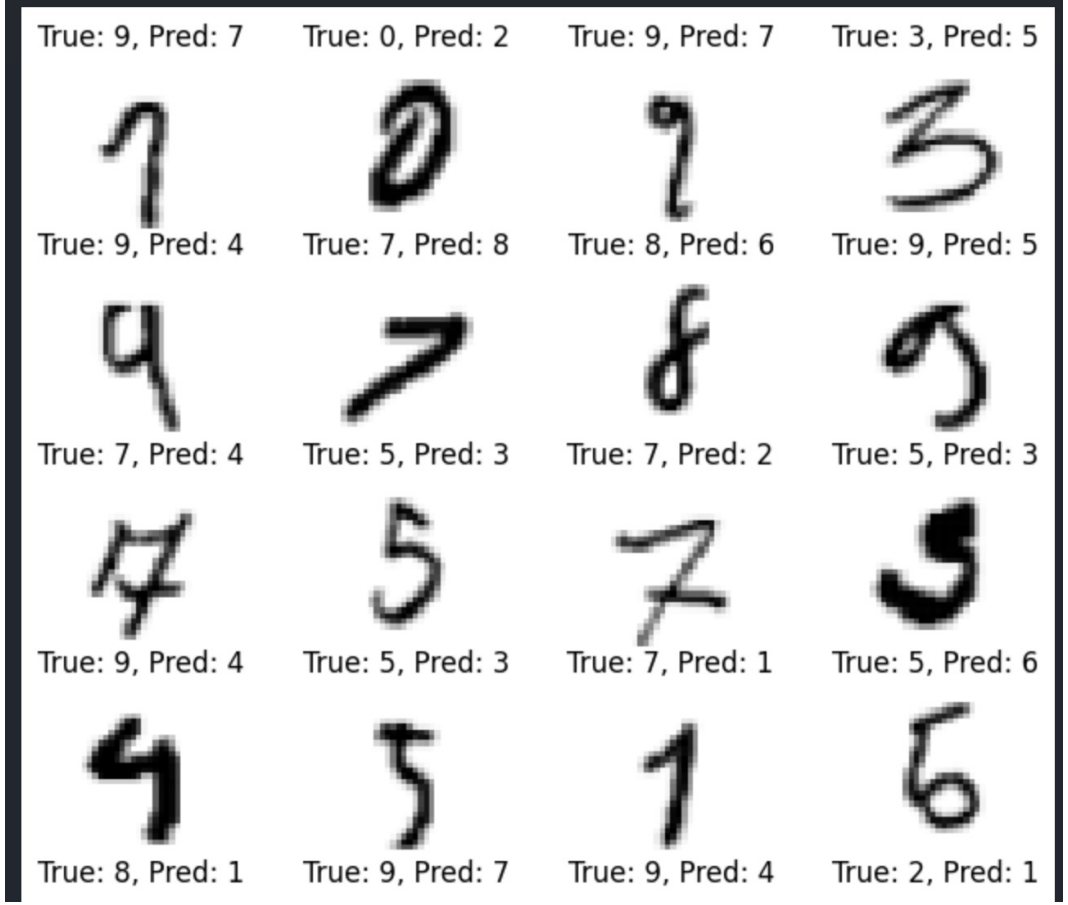
20 에포크 학습, 학습 후 테스트 셋을 이용한 성능 확인

```python
torch.manual_seed(RANDOM_SEED)

_model = LeNet5(NUM_CLASSES).cuda()
model = nn.DataParallel(_model).to(DEVICE)
optimizer = torch.optim.Adam(model.parameters(), lr=lr_schedule(0))
criterion = nn.CrossEntropyLoss()

model, optimizer, _ = training_loop(
    model, criterion, optimizer, train_loader, valid_loader, EPOCHS, DEVICE
)
```

```
22:33:32 ---  Epoch: 19 Train loss: 0.0062      Valid loss: 0.0469      Train accuracy: 99.86   Valid accuracy: 98.85

22:34:11 ---  Epoch: 20 Train loss: 0.0054      Valid loss: 0.0461      Train accuracy: 99.84   Valid accuracy: 98.80
```



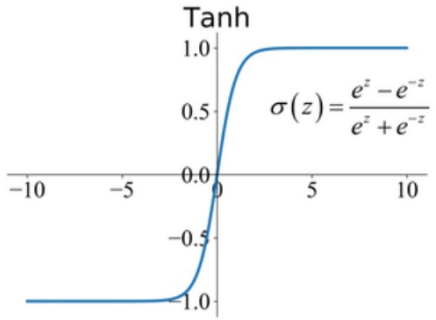

```
Test Loss: 0.0522, Test Accuracy: 0.9860
```



Activation Function 변형

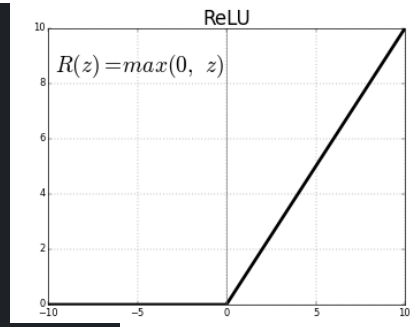
tanh

```python
def forward(self, x):
    x = F.tanh(self.conv1(x))
    x = self.pool2(x)
    x = F.tanh(self.conv3(x))
    x = self.pool4(x)
    x = F.tanh(self.conv5(x))
    x = x.view(x.size(0), -1)
    x = self.fc6(x)
    x = F.tanh(x)

    logits = self.fc7(x)

    return logits
```



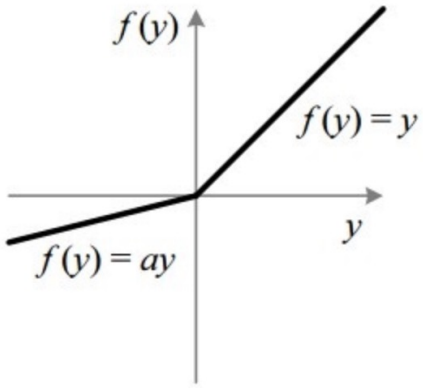
relu

```python
def forward(self, x):
    x = self.conv1(x)
    x = F.relu(x)

    x = self.pool2(x)

    x = self.conv3(x)
    x = F.relu(x)

    x = self.pool4(x)

    x = self.conv5(x)
    x = F.relu(x)

    x = x.view(x.size(0), -1)
    x = self.fc6(x)
    x = F.relu(x)

    logits = self.fc7(x)

    return logits
```



leaky-relu

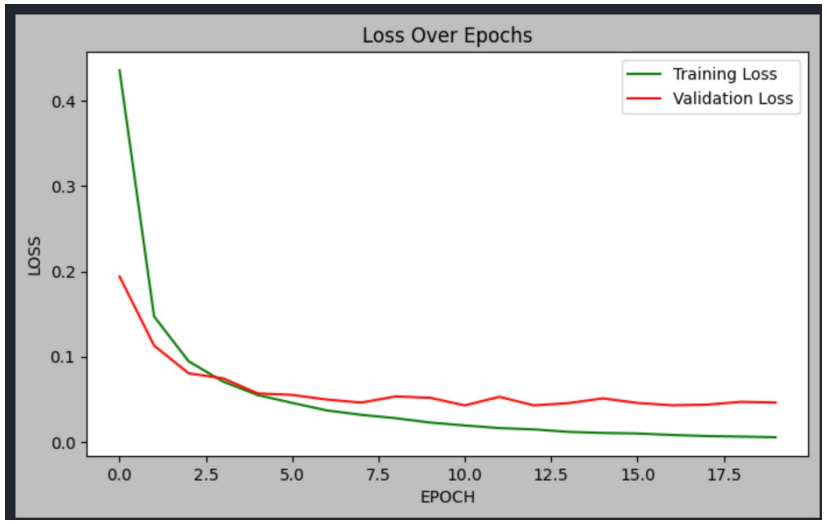
```python
def forward(self, x):
    x = self.conv1(x)
    x = F.leaky_relu(x)

    x = self.pool2(x)

    x = self.conv3(x)
    x = F.leaky_relu(x)

    x = self.pool4(x)

    x = self.conv5(x)
    x = F.leaky_relu(x)

    x = x.view(x.size(0), -1)
    x = self.fc6(x)
    x = F.leaky_relu(x)

    logits = self.fc7(x)

    return logits
```



Activation Function 변형

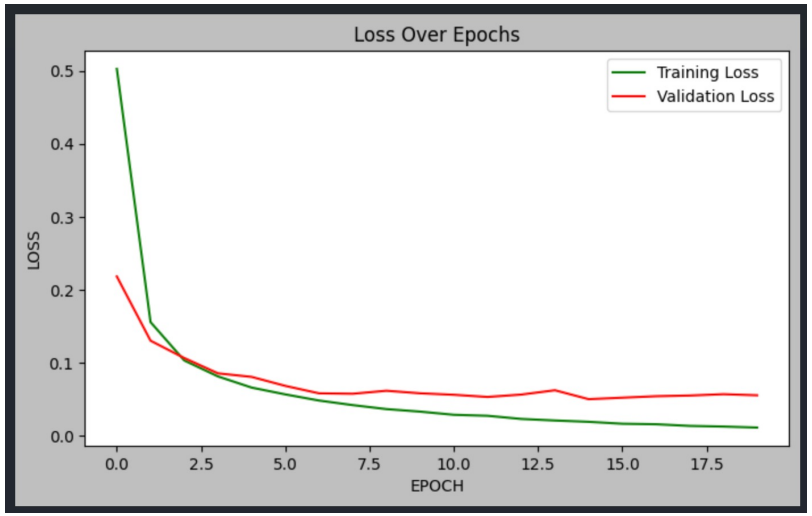
tanh

	loss	accuracy(%)
epoch20, train	0.0054	99.84
epoch20, valid	0.0461	98.80
test set	0.0522	98.60



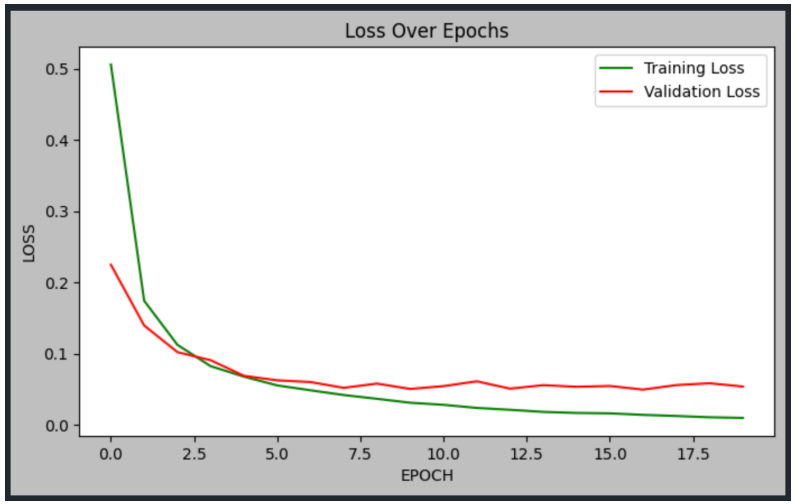
relu

	loss	accuracy(%)
epoch20, train	0.0110	99.74
epoch20, valid	0.0553	98.53
test set	**0.0499**	98.74



leaky-relu

	loss	accuracy(%)
epoch20, train	0.0097	99.85
epoch20, valid	0.0538	98.83
test set	0.0520	**98.76**



Pooling Layer 변형

AvgPool

```
def __init__(self, num_classes):
    super(LeNet5, self).__init__()
    self.conv1 = nn.Conv2d(in_channels=1, out_channels=
    self.pool2 = nn.AvgPool2d(kernel_size=2, stride=2)
    self.conv3 = nn.Conv2d(in_channels=6, out_channels=
    self.pool4 = nn.AvgPool2d(kernel_size=2, stride=2)
    self.conv5 = nn.Conv2d(in_channels=16, out_channels
    self.fc6 = nn.Linear(120, 84)
    self.fc7 = nn.Linear(84, num_classes)
```

MaxPool

```
def __init__(self, num_classes):
    super(LeNet5, self).__init__()
    self.conv1 = nn.Conv2d(in_channels=1, out_channels=
    self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
    self.conv3 = nn.Conv2d(in_channels=6, out_channels=
    self.pool4 = nn.MaxPool2d(kernel_size=2, stride=2)
    self.conv5 = nn.Conv2d(in_channels=16, out_channels
    self.fc6 = nn.Linear(120, 84)
    self.fc7 = nn.Linear(84, num_classes)
```

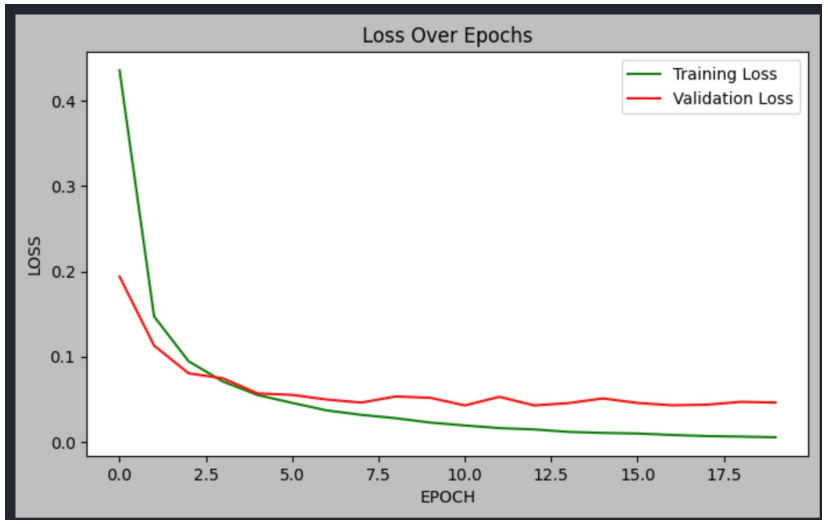
MinPool

```
def __init__(self, num_classes):
    super(LeNet5, self).__init__()
    self.conv1 = nn.Conv2d(in_channels=1,
    self.pool2 = nn.MaxPool2d(kernel_size=
    self.conv3 = nn.Conv2d(in_channels=6,
    self.pool4 = nn.MaxPool2d(kernel_size=
    self.conv5 = nn.Conv2d(in_channels=16,
    self.fc6 = nn.Linear(120, 84)
    self.fc7 = nn.Linear(84, num_classes)

def forward(self, x):
    x = self.conv1(x)
    x = F.tanh(x)

    x = -self.pool2(-x)

    x = self.conv3(x)
    x = F.tanh(x)

    x = -self.pool4(-x)

    x = self.conv5(x)
    x = F.tanh(x)

    x = x.view(x.size(0), -1)
    x = self.fc6(x)
    x = F.tanh(x)

    logits = self.fc7(x)

    return logits
```

Pooling Layer 변형

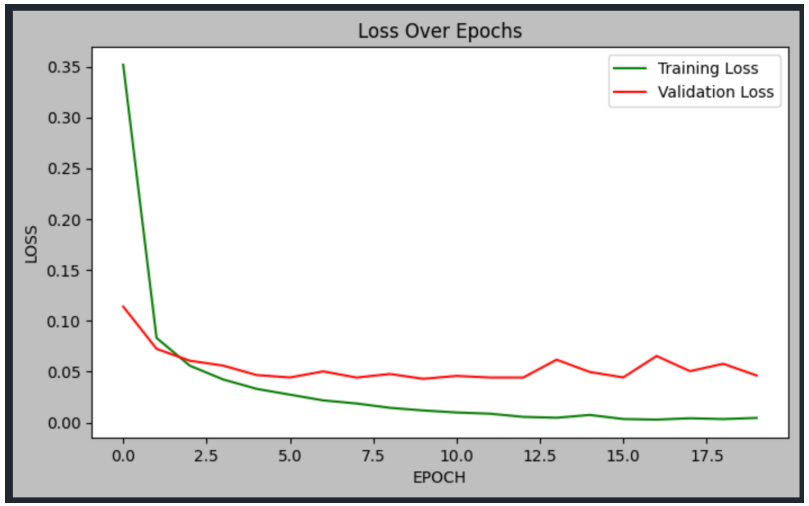
AvgPool

	loss	accuracy(%)
epoch20, train	0.0054	99.84
epoch20, valid	0.0461	98.80
test set	0.0522	98.60



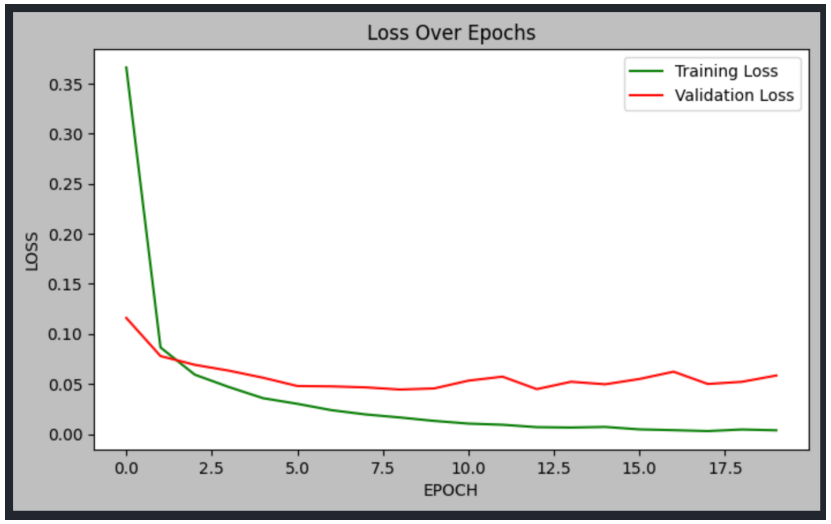
MaxPool

	loss	accuracy(%)
epoch20, train	0.0045	99.98
epoch20, valid	0.0462	98.85
test set	**0.0489**	**98.93**



MinPool

	loss	accuracy(%)
epoch20, train	0.0037	99.92
epoch20, valid	0.0584	98.57
test set	0.0533	98.89



Optimizer 변형

Adam

```
torch.manual_seed(RANDOM_SEED)

_model = LeNet5(NUM_CLASSES).cuda()
model = nn.DataParallel(_model).to(DEVICE)
optimizer = torch.optim.Adam(model.parameters(), lr=lr_schedule(0))
criterion = nn.CrossEntropyLoss()

model, optimizer, _ = training_loop(
    model, criterion, optimizer, train_loader, valid_loader, EPOCHS, DEVICE
)
```

AdaGrad

```
torch.manual_seed(RANDOM_SEED)

model = LeNet5(NUM_CLASSES).to(DEVICE)
optimizer = torch.optim.Adagrad(model.parameters())
criterion = nn.CrossEntropyLoss()

model, optimizer, _ = training_loop(
    model, criterion, optimizer, train_loader, valid
)
```

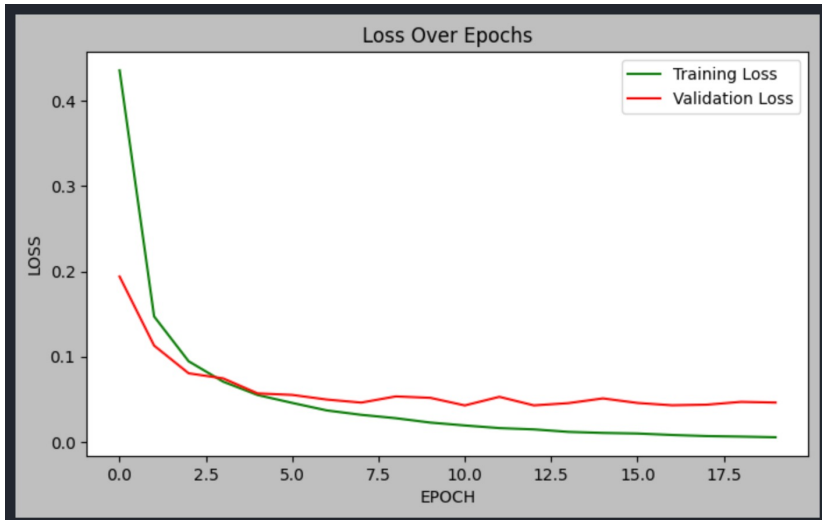
SGD

```
torch.manual_seed(RANDOM_SEED)

_model = LeNet5(NUM_CLASSES).cuda()
model = nn.DataParallel(_model).to(DEVICE)
optimizer = torch.optim.SGD(model.parameters())
criterion = nn.CrossEntropyLoss()

model, optimizer, _ = training_loop(
    model, criterion, optimizer, train_loader, v
)
```

Optimizer 변형

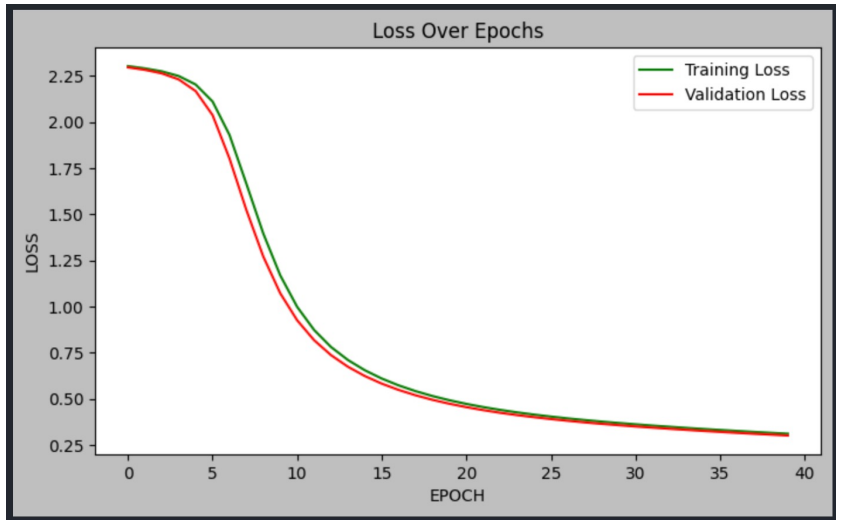
Adam

	loss	accuracy(%)
epoch20, train	0.0054	99.84
epoch20, valid	0.0461	98.80
test set	**0.0522**	**98.60**



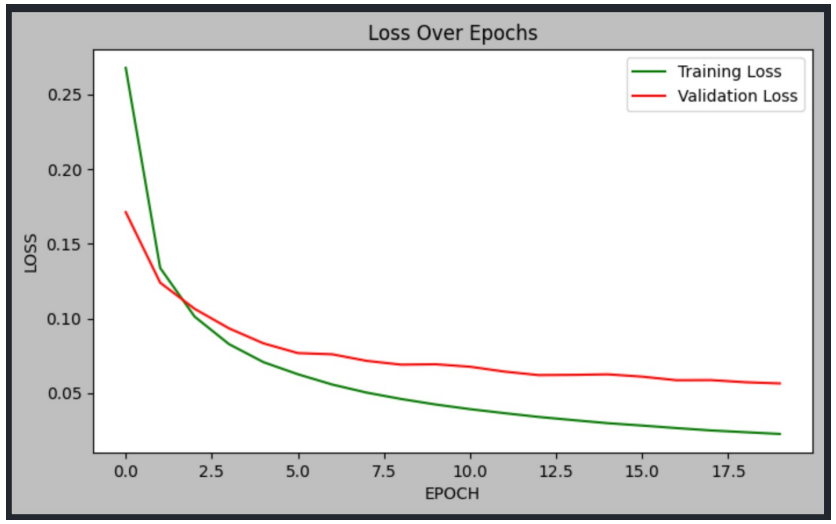
SGD

	loss	accuracy(%)
epoch40, train	0.3103	91.03
epoch40, valid	0.3006	91.22
test set	0.3186	90.70



AdaGrad

	loss	accuracy(%)
epoch20, train	0.0227	99.55
epoch20, valid	0.0565	98.29
test set	0.0562	98.47



전체 확인

https://github.com/minseok128/DeepLearning-for-VisionSystems/tree/main/1.%20LeNet-5(1998)/for_study

minseok128 Update 6-adagrad.ipynb : typo

Name	Last commit message
..	
0.ipynb	Update 0.ipynb : 모델 정리
1-relu.ipynb	Added 1-relu.ipynb : relu 함수 사용
2-leakyRelu.ipynb	Update 2-leakyRelu.ipynb typo
3-maxpool.ipynb	Added 3-maxpool.ipynb : maxpool 사용
4-minpool.ipynb	Added 4-minpool.ipynb : minpool 적용
5-SGD.ipynb	Added 5-SGD.ipynb : optimizer
6-adagrad.ipynb	Update 6-adagrad.ipynb : typo

kaggle

CO