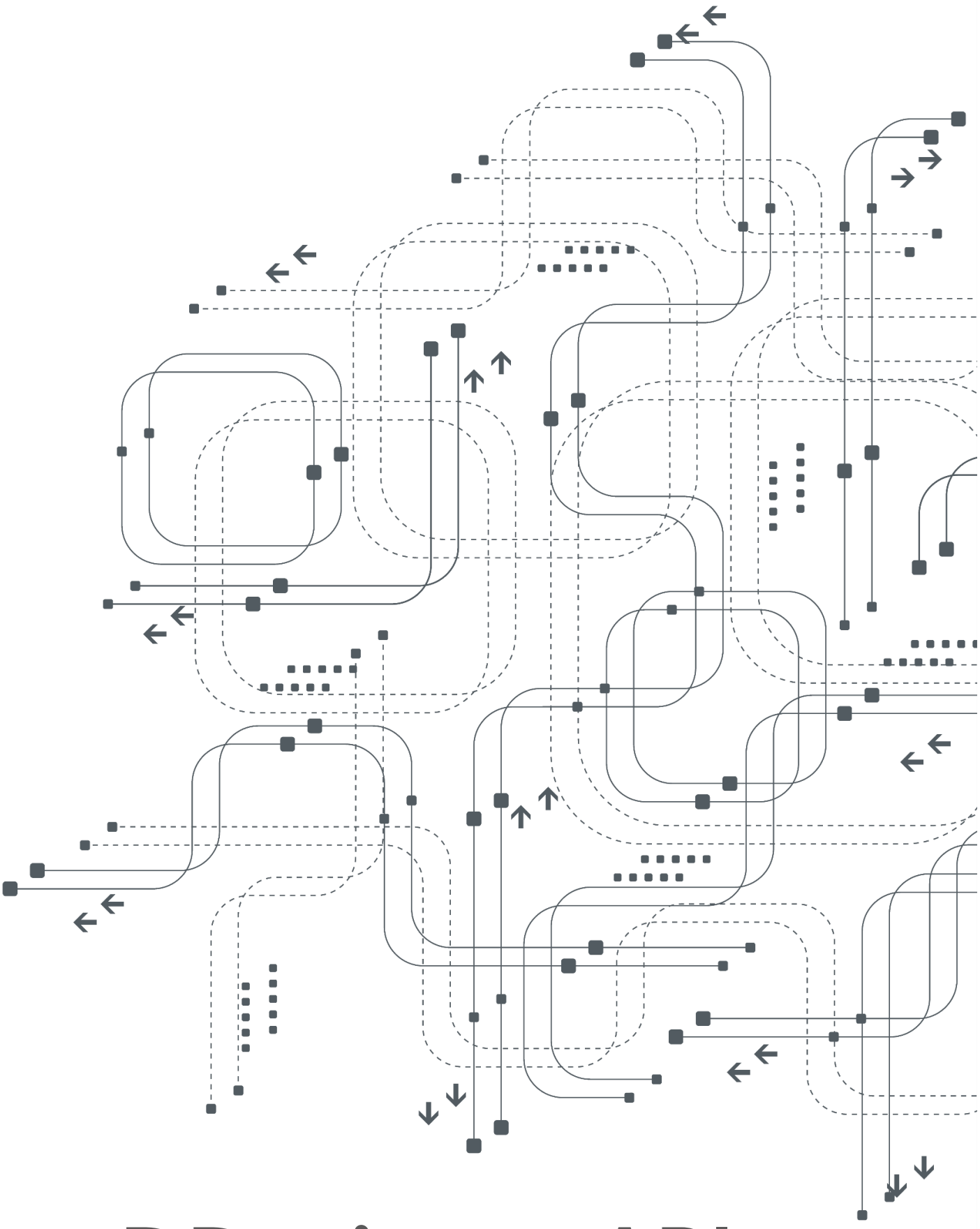


DDesigner API

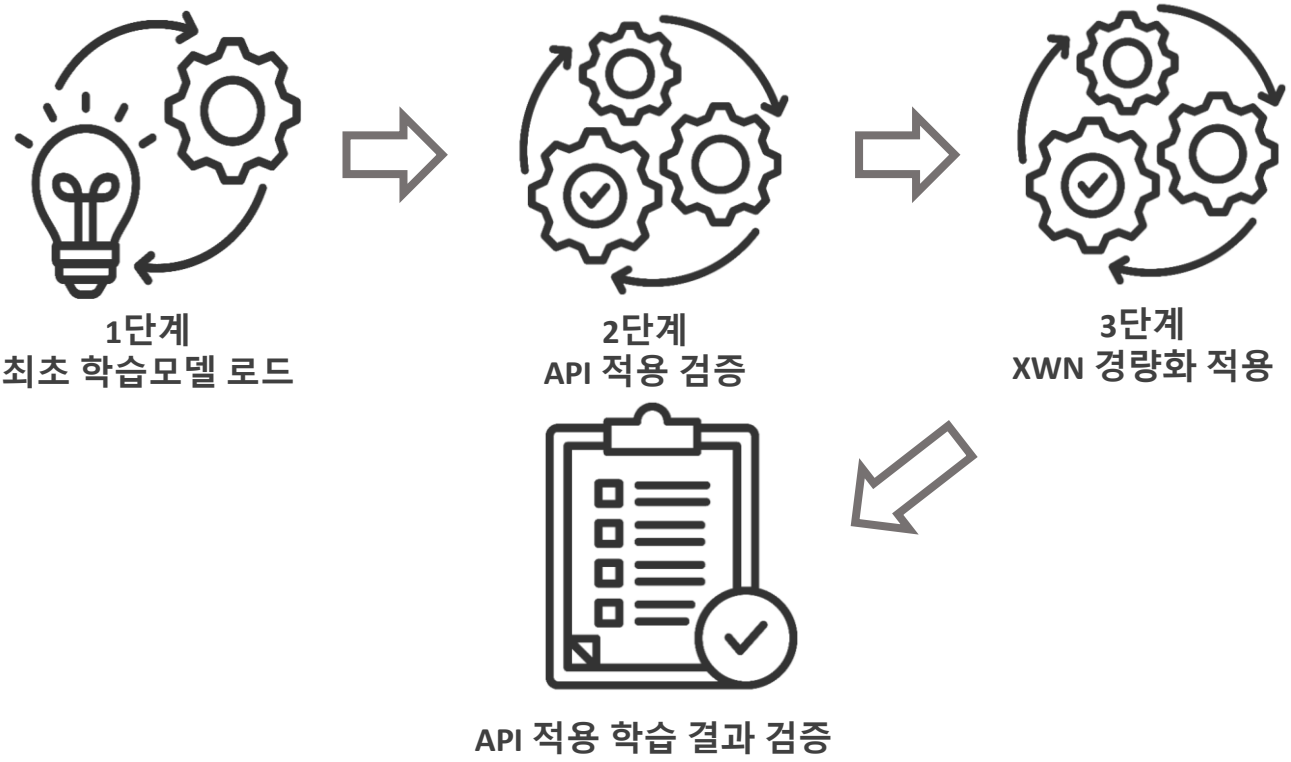
가이드북

목차

1. 개요

DDesigner API를 활용하여 학습모델 학습 시 XWN 기반 경량화 적용 프로세스를 설명합니다. 학습은 총 3단계로 나뉘며, 각각의 단계는 API 연동 및 정확도 비교를 통해 적용의 성공 여부를 검증하는 과정을 포함합니다.

또한 실무 환경에서 API를 연동해 효율적인 학습 및 최적화를 수행할 수 있도록 돕는 것을 목적으로 합니다.



1-1) DDesigner API란?

DDesigner API는 학습된 모델을 Deeper-I NPU 보드에 적합한 형태로 변환 및 최적화하는 API입니다. PyTorch와 TensorFlow 기반 학습 모델을 입력으로 받아 XWN 기반으로 구조적 경량화를 수행합니다.

1-2) 주요 특징 및 장점

→ 경량화(XWN) 지원으로 연산량 감소
→ 기존 학습 파이프라인과의 자연스러운 통합
→ 다양한 CNN 구조 지원
→ 빠른 학습 성능 유지

2. 요구사항

2-1) 시스템 요구사항

지원 모델: PyTorch, TensorFlow

입력 포맷: *.pth, *.pt, *.ckpt

2-2) 설치 전 준비 사항

→ torch 1.13.1

→ python 3.8.5

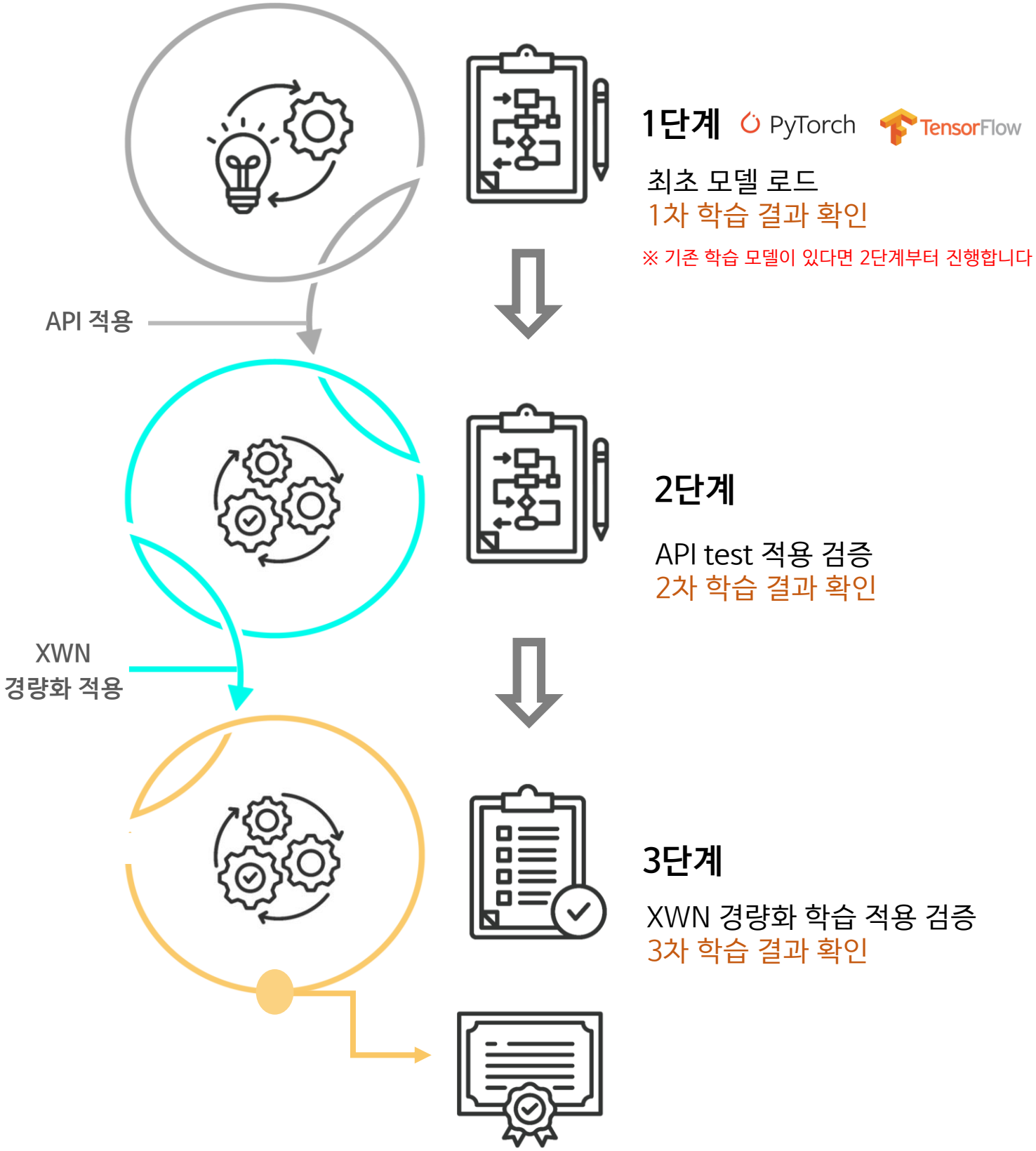
→ numpy 1.23.5

※ 본 가이드는 vim 환경을 기반한 가이드입니다.

3. 활용 예제 가이드

본 가이드는 Ddesigner API 를 활용하여 모델 학습 시 XWN 기반 경량화 적용 프로세스를 설명합니다.

DDesigner API 적용을 위한 Training은 총 3단계로 이루어집니다



각각의 단계는 API 연동 및 정확도 비교를 통해 적용의 성공 여부를 검증합니다

DDesigner API 환경 설정

```
pip install DDesignerAPI
```

1) 1단계 – 기본 모델 학습

※ 기존 학습 모델이 있다면 2단계부터 진행합니다.

→ 최초 모델 로드 및 학습 결과 확인

☐ git 예제 클론

```
git clone https://github.com/kuangliu/pytorch-cifar.git
```

※ 설치 파일 'pytorch-cifar' 확인

※ pytoch 용 API 예제 활용 링크 – 예제 파일 'resnet.py'

https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py

☐ 디렉토리 경로 설정

```
cd pytorch-cifar
```

※ 설치 후, main.py 유무 확인

☐ vim main.py 실행

```
vim main.py
```

vim main.py 실행화면

main.py

```python
'''Train CIFAR10 with PyTorch.'''
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import torch.backends.cudnn as cudnn

import torchvision
import torchvision.transforms as transforms

import os
import argparse

from models import *
from utils import progress_bar


parser = argparse.ArgumentParser(description='PyTorch CIFAR10 Training')
parser.add_argument('--lr', default=0.1, type=float, help='learning rate')
parser.add_argument('--resume', '-r', action='store_true',
                    help='resume from checkpoint')
args = parser.parse_args()

device = 'cuda' if torch.cuda.is_available() else 'cpu'
best_acc = 0  # best test accuracy
start_epoch = 0  # start from epoch 0 or last checkpoint epoch

```

□ vim main.py 수정 리스트

#net = ResNet18() -〉 net =ResNet18() - '#' 제거해서 모델 활성화

```
classes = ('plane', 'car', 'bird', 'cat', 'deer',
           'dog', 'frog', 'horse', 'ship', 'truck')

# Model
print('==> Building model..')
# net = VGG('VGG19')
# net = ResNet18()
# net = PreActResNet18()
# net = GoogLeNet()
# net = DenseNet121()
# net = ResNeXt29_2x64d()
# net = MobileNet()
# net = MobileNetV2()
# net = DPN92()
# net = ShuffleNetG2()
# net = SENet18()
# net = ShuffleNetV2(1)
# net = EfficientNetB0()
# net = RegNetX_200MF()
net = SimpleDLA()
net = net.to(device)
if device == 'cuda':
    net = torch.nn.DataParallel(net)
    cudnn.benchmark = True
```

```
classes = ('plane', 'car', 'bird', 'cat', 'deer',
           'dog', 'frog', 'horse', 'ship', 'truck')

# Model
print('==> Building model..')
# net = VGG('VGG19')
net = ResNet18()
# net = PreActResNet18()
# net = GoogLeNet()
# net = DenseNet121()
# net = ResNeXt29_2x64d()
# net = MobileNet()
# net = MobileNetV2()
# net = DPN92()
# net = ShuffleNetG2()
# net = SENet18()
# net = ShuffleNetV2(1)
# net = EfficientNetB0()
# net = RegNetX_200MF()
net = SimpleDLA()
net = net.to(device)
if device == 'cuda':
    net = torch.nn.DataParallel(net)
    cudnn.benchmark = True
```

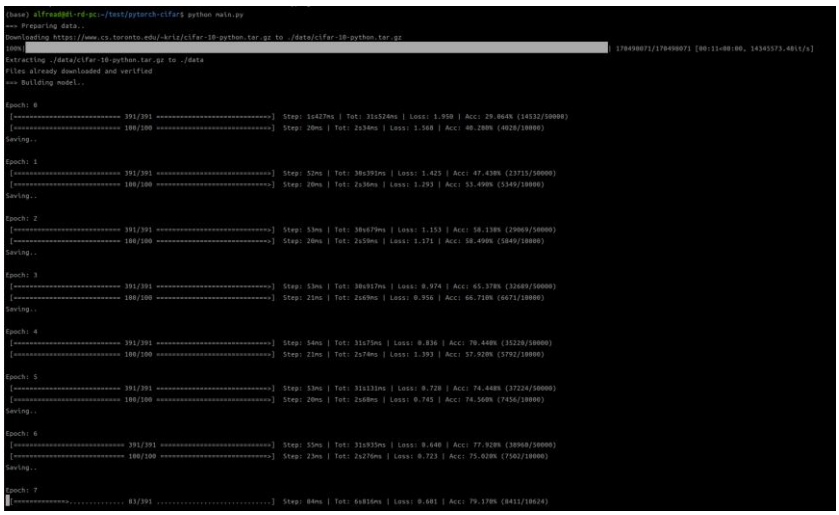
net = net.to(device) -〉 # net = net.to(device) - '#' 추가하여 모델 비활성화

```
classes = ('plane', 'car', 'bird', 'cat', 'deer',
           'dog', 'frog', 'horse', 'ship', 'truck')

# Model
print('==> Building model..')
# net = VGG('VGG19')
net = ResNet18()
# net = PreActResNet18()
# net = GoogLeNet()
# net = DenseNet121()
# net = ResNeXt29_2x64d()
# net = MobileNet()
# net = MobileNetV2()
# net = DPN92()
# net = ShuffleNetG2()
# net = SENet18()
# net = ShuffleNetV2(1)
# net = EfficientNetB0()
# net = RegNetX_200MF()
net = SimpleDLA()
net = net.to(device)
if device == 'cuda':
    net = torch.nn.DataParallel(net)
    cudnn.benchmark = True
```

```
classes = ('plane', 'car', 'bird', 'cat', 'deer',
           'dog', 'frog', 'horse', 'ship', 'truck')

# Model
print('==> Building model..')
# net = VGG('VGG19')
net = ResNet18()
# net = PreActResNet18()
# net = GoogLeNet()
# net = DenseNet121()
# net = ResNeXt29_2x64d()
# net = MobileNet()
# net = MobileNetV2()
# net = DPN92()
# net = ShuffleNetG2()
# net = SENet18()
# net = ShuffleNetV2(1)
# net = EfficientNetB0()
# net = RegNetX_200MF()
# net = SimpleDLA()
net = net.to(device)
if device == 'cuda':
    net = torch.nn.DataParallel(net)
    cudnn.benchmark = True
```

□ 1차 Training 학습 시작 코드 기입

```
python main.py
```



학습 완료 확인 화면

2) 2단계 – DDesigner API 연동 검증

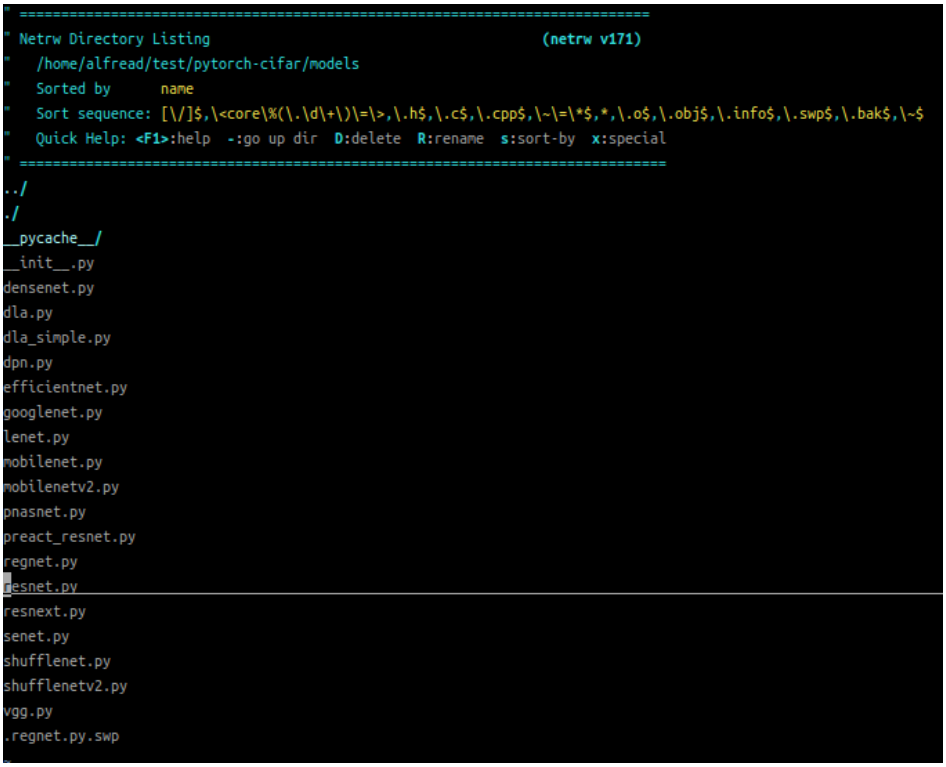
→ API 적용이 잘 되었는지 검증

→ XWN 기능은 OFF 상태

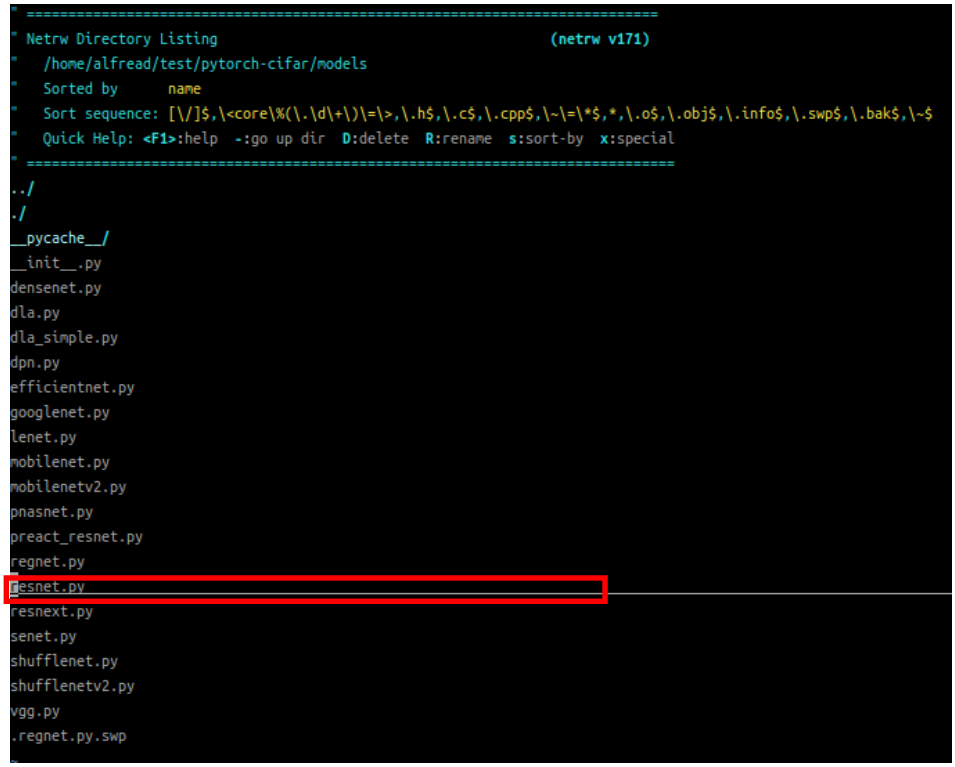
☐ vim models 코드 기입

```
vim models
```

vim models 실행화면



resnet.py 파일 선택



regnet.py 파일 실행 화면

```
'''
import torch
import torch.nn as nn
import torch.nn.functional as F

from ddesigner_api.pytoch.xwn import torch_nn as cnn

class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_planes, planes, stride=1):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(
            in_planes, planes, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = nn.Conv2d(planes, planes, kernel_size=3,
                               stride=1, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_planes, self.expansion*planes,
                          kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(self.expansion*planes)
            )
```

☐ DDesigner API 코드 기입 / 수정

from ddesignr_api.pytorch.xwn import torch_nn as cnn

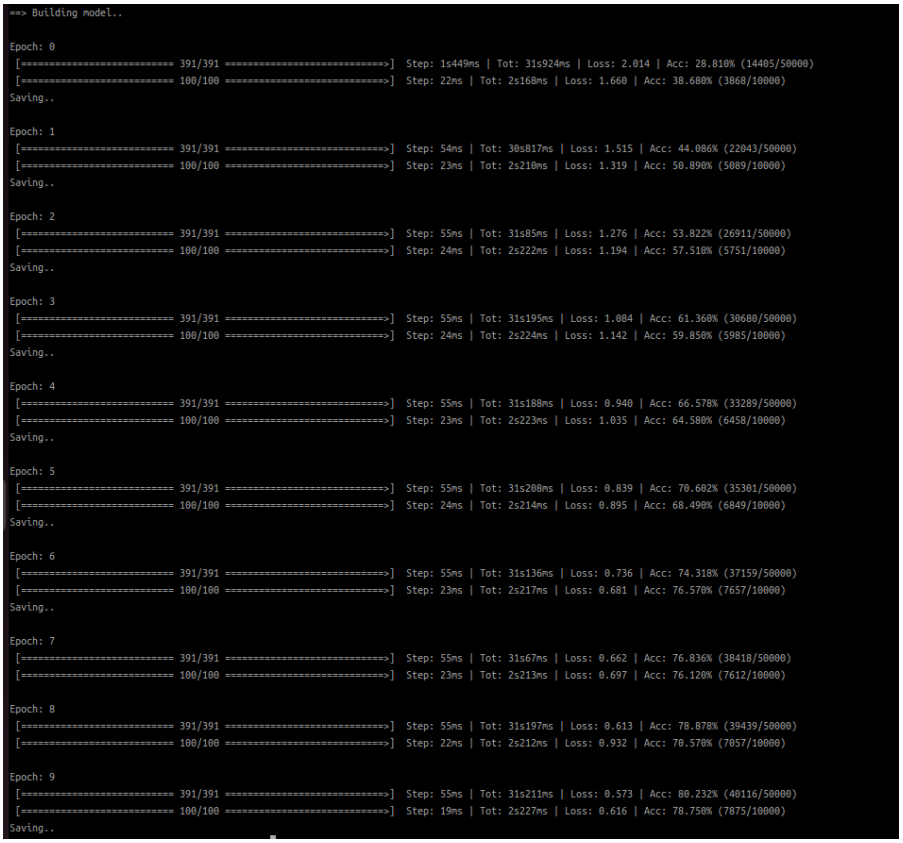
```
'''
import torch
import torch.nn as nn
import torch.nn.functional as F

from ddesigner_api.pytoch.xwn import torch_nn as cnn

class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_planes, planes, stride=1):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(
            in_planes, planes, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = nn.Conv2d(planes, planes, kernel_size=3,
                               stride=1, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_planes, self.expansion*planes,
                          kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(self.expansion*planes)
            )
```

※ 기존 nn.Conv2d 코드를 → cnn.Conv2d 로 바꿔주세요.

☐ python main.py 파일 실행

```
python main.py
```

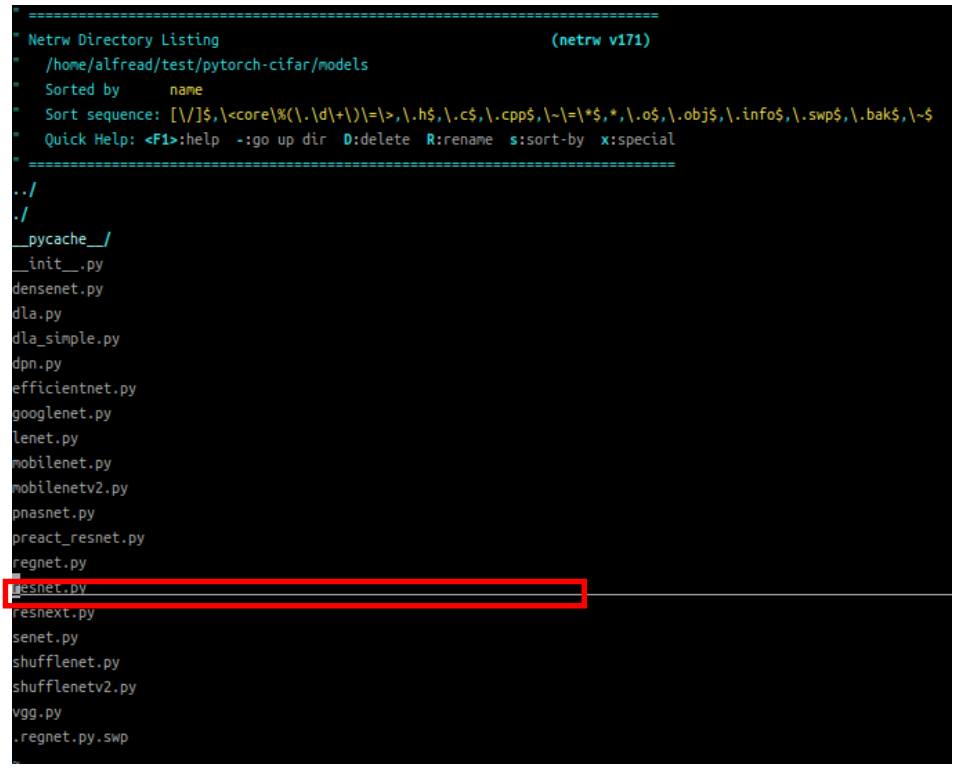
실행 완료 화면



3) 3단계 – XWN 적용 학습

→ XWN 기반 경량화 적용

vim models 실행



resnet.py 파일 실행

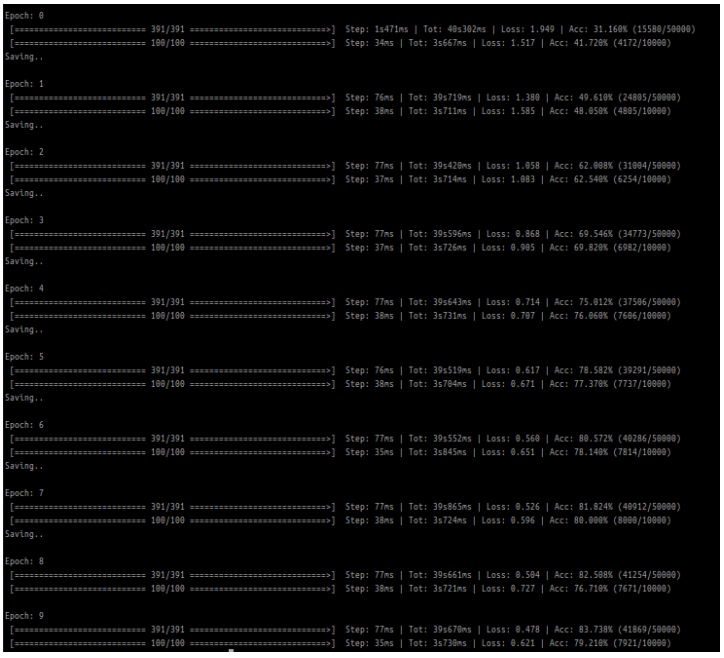
use_transform=True, max_scale=4.0, use_pruning=True, prun_weight=0.50, 코드 적용

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

from ddesigner_api.pytorch.xwn import torch_nn as cnn

class BasicBlock(nn.Module):
    expansion = 1

    def __init__(self, in_planes, planes, stride=1):
        super(BasicBlock, self).__init__()
        self.conv1 = cnn.Conv2d(
            in_planes, planes, kernel_size=3, stride=stride, padding=1, bias=False,
            use_transform=True, max_scale=4.0, use_pruning=True, prun_weight=0.50,
            )
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = cnn.Conv2d(planes, planes, kernel_size=3,
                                stride=1, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                cnn.Conv2d(in_planes, self.expansion*planes,
                           kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(self.expansion*planes)
            )
```

:wq로 저장 후, 나와서

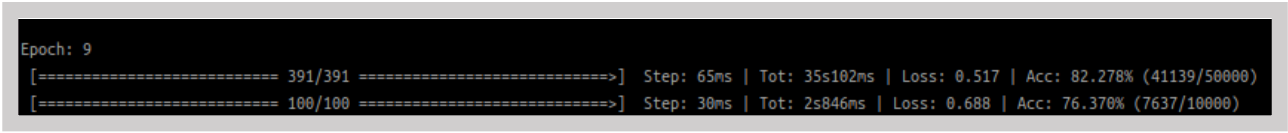
☐ python main.py 실행

```
python main.py
```



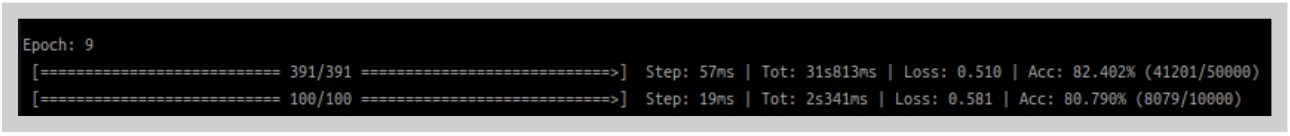
각 단계별 학습결과 비교

1단계 학습결과



Acc: 82%

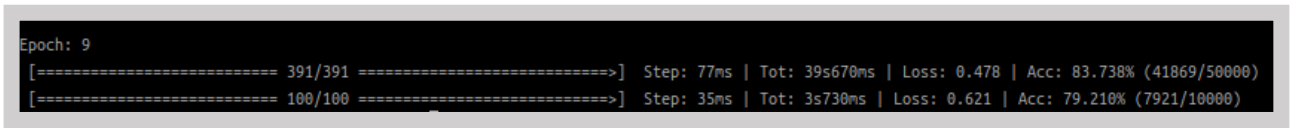
2단계 학습결과



※ 1단계, 2단계 학습결과가 다를 시, DDesigner API 연동 유무 재확인 필요

Acc: 82%

3단계 학습결과



※ 2단계 학습 결과 비교하여, XWN 최적화 학습 결과 확인 필요
(2단계 학습결과와 비슷한 수치가 나와야 함)

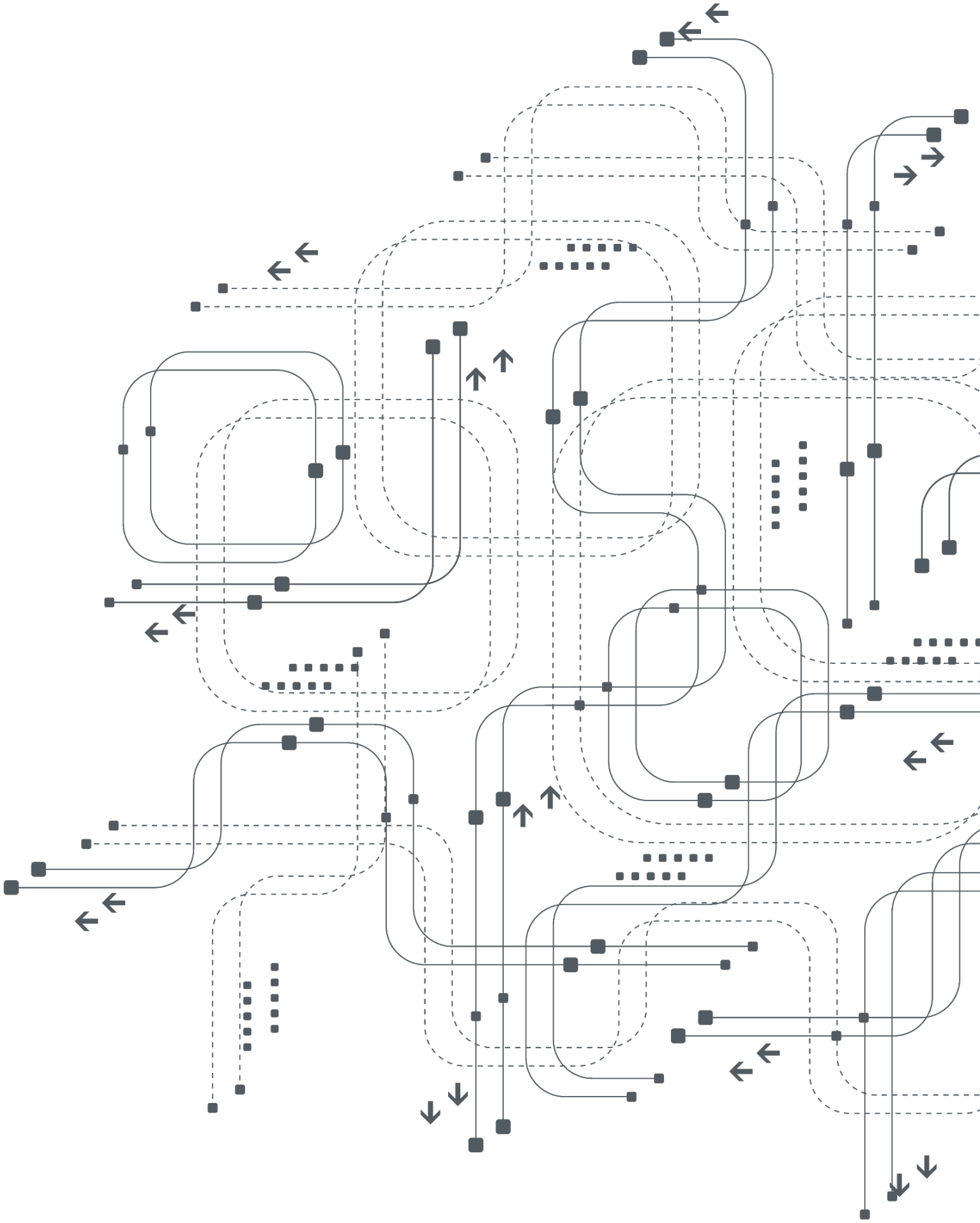
Acc: 83%

4) 정확도 비교 및 정상 동작 검증 기준

단계	정확도 기대치	검증 기준
1단계	ex) 0.82%	최초 검증 기준치
2단계	ex) 0.82%	동일시 정상
3단계	ex) 0.83%	큰 차이 없을 경우 정상

0.82% ↔ 0.83% = 정상

※ 본 가이드의 Epoch 횟수는 임의 9회로 설정함
이에 실전 학습 결과 Acc 수치는 경우에 따라 상이할 수 있음



고객 지원 (Customer Support)

- **문의처**: partner@deeper-i.ai
- **웹사이트**: https://www.deeper-i.ai