

DDesigner API

가이드북

목차

1. 개요

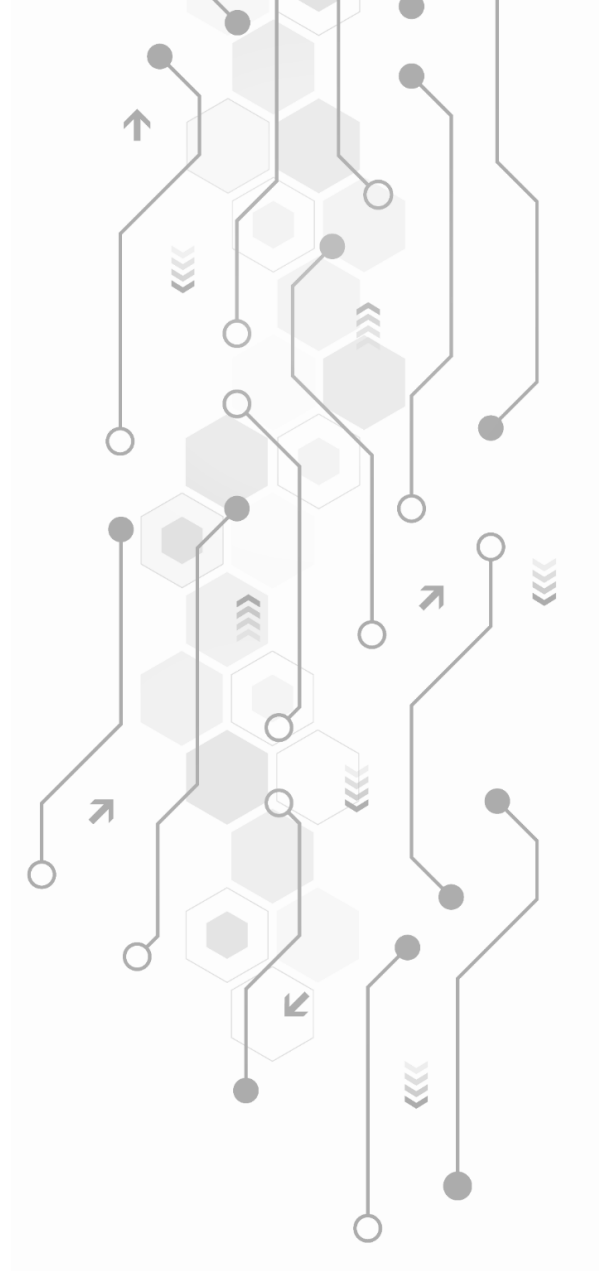
- 1-1) DDesigner API란?
- 1-2) 주요 특징 및 장점

2. 요구사항

- 2-1) 시스템 요구사항
- 2-2) 설치 전 준비 사항

3. 활용 예제 가이드

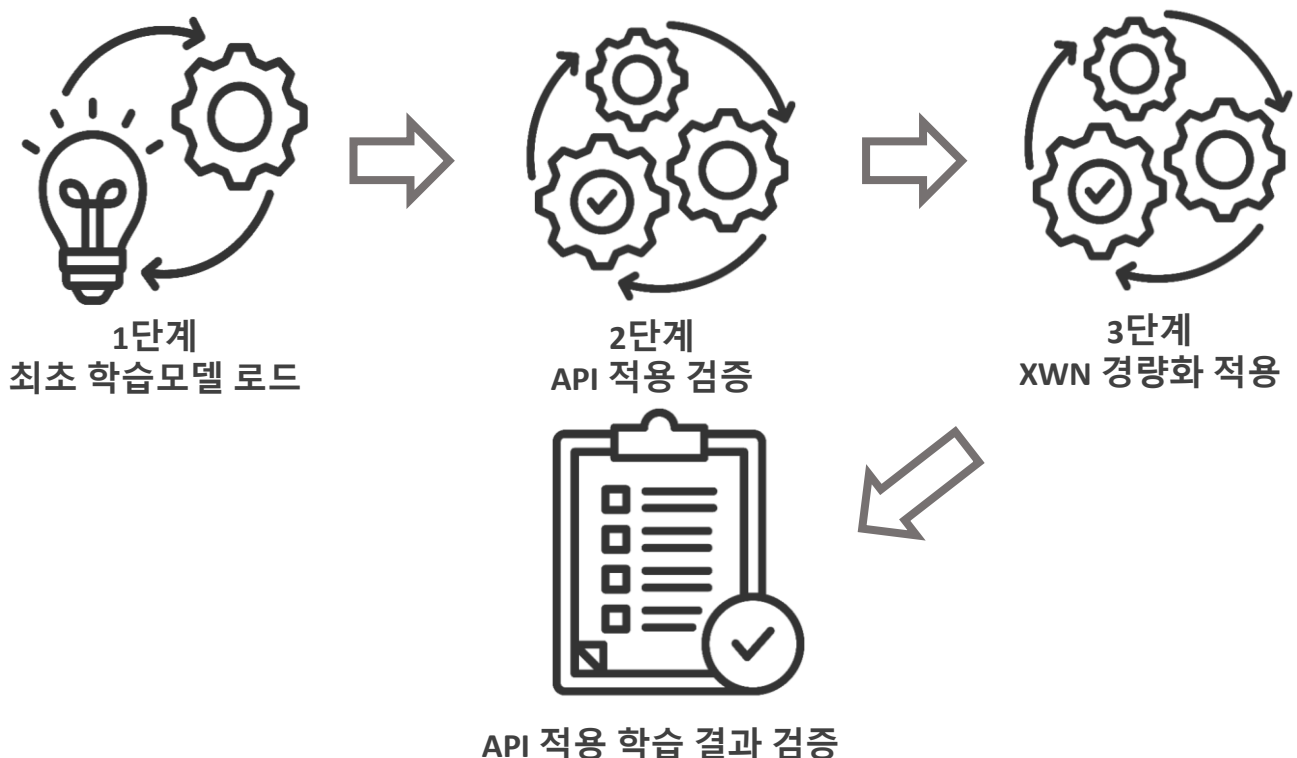
- 3-1) 1단계 – 기본 모델 학습
- 3-2) 2단계 – DDesigner API 연동 검증
- 3-3) 3단계 – XWN(경량화) 적용 학습
- 3-4) 정확도 비교 및 정상 동작 검증 기준



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

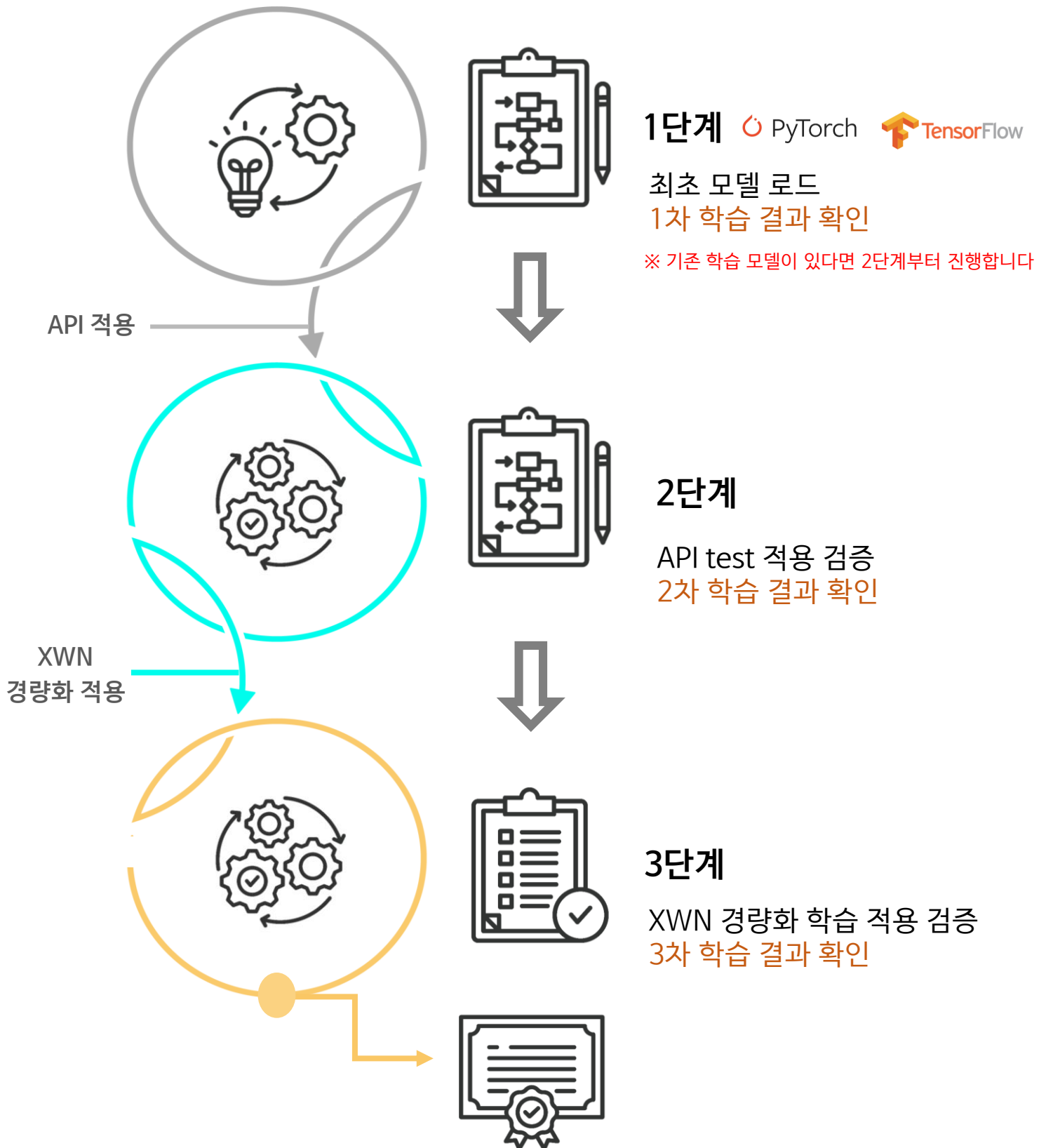
→ python 3.8.5

→ numpy 1.23.5

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

※ pytorch 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 1
1 '''Train CIFAR10 with PyTorch.'''
2 import torch
3 import torch.nn as nn
4 import torch.optim as optim
5 import torch.nn.functional as F
6 import torch.backends.cudnn as cudnn
7
8 import torchvision
9 import torchvision.transforms as transforms
10
11 import os
12 import argparse
13
14 from models import *
15 from utils import progress_bar
16
17
18 parser = argparse.ArgumentParser(description='PyTorch CIFAR10 Training')
19 parser.add_argument('--lr', default=0.1, type=float, help='learning rate')
20 parser.add_argument('--resume', '-r', action='store_true',
21                     help='resume from checkpoint')
22 args = parser.parse_args()
23
24 device = 'cuda' if torch.cuda.is_available() else 'cpu'
25 best_acc = 0 # best test accuracy
26 start_epoch = 0 # start from epoch 0 or last checkpoint epoch
27
```

□ 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

```

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

```

□ 1차 Training 학습 시작 코드 기입

```
python main.py
```

```

$ curl -o /dev/null -H "Host: bost.pytorch-cifar-10-python-main.py" https://bost.pytorch-cifar-10-python-main.py
-- Preparing data...
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
[#####] 79499007/79499007 [00:11:06.10, 14345373.481/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
-- Building model...

Epoch: 0
[#####] 391/391 [#####] Step: 154279s | Tot: 3153260s | Loss: 1.908 | Acc: 29.8648 (14532/50000)
[#####] 100/100 [#####] Step: 20m | Tot: 2x30m | Loss: 1.568 | Acc: 40.2808 (4028/10000)
Waiting...

Epoch: 1
[#####] 391/391 [#####] Step: 53m | Tot: 36x3930m | Loss: 1.425 | Acc: 47.4006 (23715/50000)
[#####] 100/100 [#####] Step: 20m | Tot: 2x30m | Loss: 1.293 | Acc: 53.4908 (5349/10000)
Waiting...

Epoch: 2
[#####] 391/391 [#####] Step: 53m | Tot: 36x579ms | Loss: 1.153 | Acc: 58.1108 (29069/50000)
[#####] 100/100 [#####] Step: 20m | Tot: 2x50m | Loss: 1.171 | Acc: 58.4908 (5849/10000)
Waiting...

Epoch: 3
[#####] 391/391 [#####] Step: 53m | Tot: 36x8179ms | Loss: 0.974 | Acc: 63.3788 (32089/50000)
[#####] 100/100 [#####] Step: 21m | Tot: 2x40m | Loss: 0.936 | Acc: 68.7108 (6871/10000)
Waiting...

Epoch: 4
[#####] 391/391 [#####] Step: 54m | Tot: 31x730ms | Loss: 0.836 | Acc: 70.4408 (35228/50000)
[#####] 100/100 [#####] Step: 21m | Tot: 2x70m | Loss: 1.393 | Acc: 57.9208 (5792/10000)
Waiting...

Epoch: 5
[#####] 391/391 [#####] Step: 53m | Tot: 31x1310ms | Loss: 0.728 | Acc: 74.4408 (37224/50000)
[#####] 100/100 [#####] Step: 20m | Tot: 2x80ms | Loss: 0.745 | Acc: 74.5608 (7456/10000)
Waiting...

Epoch: 6
[#####] 391/391 [#####] Step: 53m | Tot: 31x8370ms | Loss: 0.640 | Acc: 77.9208 (38908/50000)
[#####] 100/100 [#####] Step: 21m | Tot: 1x270m | Loss: 0.723 | Acc: 75.6208 (7562/10000)
Waiting...

Epoch: 7
[#####] 63/63 [#####] Step: 88m | Tot: 6x110ms | Loss: 0.681 | Acc: 79.1708 (8117/10000)

```

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

```
=====
Netrw Directory Listing                                (netrw v171)
/home/alfread/test/pytorch-cifar/models
Sorted by      name
Sort sequence: [\/]$, \<core\%(\\.d\\+\\)=|>, \.h$, \.c$, \.cpp$, \~|=|*$, *, \.o$, \.obj$, \.info$, \.swp$, \.bak$, \~$
Quick Help: <F1>:help  -:go up dir  D:delete  R:rename  s:sort-by  x:special
=====
../
./
__pycache__/
__init__.py
densenet.py
dla.py
dla_simple.py
dpn.py
efficientnet.py
googlenet.py
lenet.py
mobilenet.py
mobilenetv2.py
pnasnet.py
preact_resnet.py
regnet.py
resnet.py
resnext.py
senet.py
shufflenet.py
shufflenetv2.py
vgg.py
.regnet.py.swp
```

resnet.py 파일 선택

```
=====
Netrw Directory Listing                                (netrw v171)
/home/alfread/test/pytorch-cifar/models
Sorted by      name
Sort sequence: [\/]$, \<core\%(\\.d\\+\\)=|>, \.h$, \.c$, \.cpp$, \~|=|*$, *, \.o$, \.obj$, \.info$, \.swp$, \.bak$, \~$
Quick Help: <F1>:help  -:go up dir  D:delete  R:rename  s:sort-by  x:special
=====
../
./
__pycache__/
__init__.py
densenet.py
dla.py
dla_simple.py
dpn.py
efficientnet.py
googlenet.py
lenet.py
mobilenet.py
mobilenetv2.py
pnasnet.py
preact_resnet.py
regnet.py
resnet.py
resnext.py
senet.py
shufflenet.py
shufflenetv2.py
vgg.py
.regnet.py.swp
```


regnet.py 파일 실행 화면

```
'''
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 = 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 ddesigner_api.pytorch.xwn import torch_nn as cnn
```

```
'''
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 = 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

```
==> Building model...

Epoch: 0
[===== 391/391 =====] Step: 1s449ms | Tot: 31s924ms | Loss: 2.814 | Acc: 28.810% (14405/50000)
[===== 100/100 =====] Step: 22ms | Tot: 2s168ms | Loss: 1.660 | Acc: 38.680% (3868/10000)
Saving...

Epoch: 1
[===== 391/391 =====] Step: 54ms | Tot: 30s817ms | Loss: 1.515 | Acc: 44.086% (22043/50000)
[===== 100/100 =====] Step: 23ms | Tot: 2s210ms | Loss: 1.319 | Acc: 50.890% (5089/10000)
Saving...

Epoch: 2
[===== 391/391 =====] Step: 55ms | Tot: 31s85ms | Loss: 1.276 | Acc: 53.822% (26911/50000)
[===== 100/100 =====] Step: 24ms | Tot: 2s222ms | Loss: 1.194 | Acc: 57.510% (5751/10000)
Saving...

Epoch: 3
[===== 391/391 =====] Step: 55ms | Tot: 31s195ms | Loss: 1.084 | Acc: 61.360% (30600/50000)
[===== 100/100 =====] Step: 24ms | Tot: 2s224ms | Loss: 1.142 | Acc: 59.850% (5985/10000)
Saving...

Epoch: 4
[===== 391/391 =====] Step: 55ms | Tot: 31s188ms | Loss: 0.940 | Acc: 66.578% (33289/50000)
[===== 100/100 =====] Step: 23ms | Tot: 2s223ms | Loss: 1.035 | Acc: 64.580% (6458/10000)
Saving...

Epoch: 5
[===== 391/391 =====] Step: 55ms | Tot: 31s208ms | Loss: 0.839 | Acc: 70.602% (35301/50000)
[===== 100/100 =====] Step: 24ms | Tot: 2s214ms | Loss: 0.895 | Acc: 68.490% (6849/10000)
Saving...

Epoch: 6
[===== 391/391 =====] Step: 55ms | Tot: 31s136ms | Loss: 0.736 | Acc: 74.318% (37159/50000)
[===== 100/100 =====] Step: 23ms | Tot: 2s217ms | Loss: 0.681 | Acc: 76.570% (7657/10000)
Saving...

Epoch: 7
[===== 391/391 =====] Step: 55ms | Tot: 31s67ms | Loss: 0.662 | Acc: 76.836% (38418/50000)
[===== 100/100 =====] Step: 23ms | Tot: 2s213ms | Loss: 0.697 | Acc: 76.120% (7612/10000)

Epoch: 8
[===== 391/391 =====] Step: 55ms | Tot: 31s197ms | Loss: 0.613 | Acc: 78.878% (39439/50000)
[===== 100/100 =====] Step: 22ms | Tot: 2s212ms | Loss: 0.932 | Acc: 70.570% (7057/10000)

Epoch: 9
[===== 391/391 =====] Step: 55ms | Tot: 31s211ms | Loss: 0.573 | Acc: 80.232% (40116/50000)
[===== 100/100 =====] Step: 19ms | Tot: 2s227ms | Loss: 0.616 | Acc: 78.750% (7875/10000)
Saving...
```

3) 3단계 – XWN 적용 학습

→ XWN 기반 경량화 적용

vim models 실행

```
=====
" Netrw Directory Listing                               (netrw v171)
" /home/alfread/test/pytorch-cifar/models
" Sorted by      name
" Sort sequence: [V]$, \<core\%(\\.d|\\)|=|>, \.h$, \.c$, \.cpp$, \-|=*$, *, \.o$, \.obj$, \.info$, \.swp$, \.bak$, \-
" Quick Help: <F1>:help  -:go up dir  D:delete  R:rename  S:sort-by  X:special
=====
../
./
__pycache__/
__init__.py
densenet.py
dla.py
dla_simple.py
dnn.py
efficientnet.py
googlenet.py
lenet.py
mobilenet.py
mobilenetv2.py
pnasnet.py
preact_resnet.py
regnet.py
resnet.py
resnext.py
senet.py
shufflenet.py
shufflenetv2.py
vgg.py
regnet.py.swp
```

resnet.py 파일 실행

use transform=True, max_scale=4.0, use pruning=True, prun_weight=0.50, 코드 적용

```
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

```
Epoch: 0
[===== 301/301 =====] Step: 1s47ms | Tot: 40s30ms | Loss: 1.949 | Acc: 31.160% (15580/50000)
[===== 100/100 =====] Step: 34ms | Tot: 3s667ms | Loss: 1.517 | Acc: 41.720% (4172/10000)
Saving..

Epoch: 1
[===== 301/301 =====] Step: 76ms | Tot: 39s710ms | Loss: 1.380 | Acc: 40.610% (2485/50000)
[===== 100/100 =====] Step: 38ms | Tot: 3s711ms | Loss: 1.585 | Acc: 48.850% (4885/10000)
Saving..

Epoch: 2
[===== 301/301 =====] Step: 77ms | Tot: 39s420ms | Loss: 1.058 | Acc: 62.080% (3104/50000)
[===== 100/100 =====] Step: 37ms | Tot: 3s714ms | Loss: 1.083 | Acc: 62.540% (6254/10000)
Saving..

Epoch: 3
[===== 301/301 =====] Step: 77ms | Tot: 39s596ms | Loss: 0.868 | Acc: 69.546% (3477/50000)
[===== 100/100 =====] Step: 37ms | Tot: 3s726ms | Loss: 0.905 | Acc: 69.820% (6982/10000)
Saving..

Epoch: 4
[===== 301/301 =====] Step: 77ms | Tot: 39s643ms | Loss: 0.714 | Acc: 75.812% (3756/50000)
[===== 100/100 =====] Step: 38ms | Tot: 3s721ms | Loss: 0.707 | Acc: 76.860% (7686/10000)
Saving..

Epoch: 5
[===== 301/301 =====] Step: 76ms | Tot: 39s510ms | Loss: 0.617 | Acc: 78.582% (3929/50000)
[===== 100/100 =====] Step: 38ms | Tot: 3s784ms | Loss: 0.671 | Acc: 77.370% (7737/10000)
Saving..

Epoch: 6
[===== 301/301 =====] Step: 77ms | Tot: 39s552ms | Loss: 0.560 | Acc: 80.572% (4026/50000)
[===== 100/100 =====] Step: 35ms | Tot: 3s845ms | Loss: 0.651 | Acc: 78.140% (7814/10000)
Saving..

Epoch: 7
[===== 301/301 =====] Step: 77ms | Tot: 39s865ms | Loss: 0.526 | Acc: 81.824% (4091/50000)
[===== 100/100 =====] Step: 38ms | Tot: 3s724ms | Loss: 0.596 | Acc: 80.000% (8000/10000)
Saving..

Epoch: 8
[===== 301/301 =====] Step: 77ms | Tot: 39s661ms | Loss: 0.584 | Acc: 82.580% (4125/50000)
[===== 100/100 =====] Step: 38ms | Tot: 3s721ms | Loss: 0.727 | Acc: 76.710% (7671/10000)

Epoch: 9
[===== 301/301 =====] Step: 77ms | Tot: 39s670ms | Loss: 0.478 | Acc: 83.738% (4189/50000)
[===== 100/100 =====] Step: 35ms | Tot: 3s730ms | Loss: 0.621 | Acc: 79.210% (7921/10000)
```

각 단계별 학습결과 비교

1단계 학습결과

```
Epoch: 9
[===== 391/391 =====>] Step: 65ms | Tot: 35s102ms | Loss: 0.517 | Acc: 82.278% (41139/50000)
[===== 100/100 =====>] Step: 30ms | Tot: 2s846ms | Loss: 0.688 | Acc: 76.370% (7637/10000)
```

Acc: 82%

2단계 학습결과

```
Epoch: 9
[===== 391/391 =====>] Step: 57ms | Tot: 31s813ms | Loss: 0.510 | Acc: 82.402% (41201/50000)
[===== 100/100 =====>] Step: 19ms | Tot: 2s341ms | Loss: 0.581 | Acc: 80.790% (8079/10000)
```

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

Acc: 82%

3단계 학습결과

```
Epoch: 9
[===== 391/391 =====>] Step: 77ms | Tot: 39s670ms | Loss: 0.478 | Acc: 83.738% (41869/50000)
[===== 100/100 =====>] Step: 35ms | Tot: 3s730ms | Loss: 0.621 | Acc: 79.210% (7921/10000)
```

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

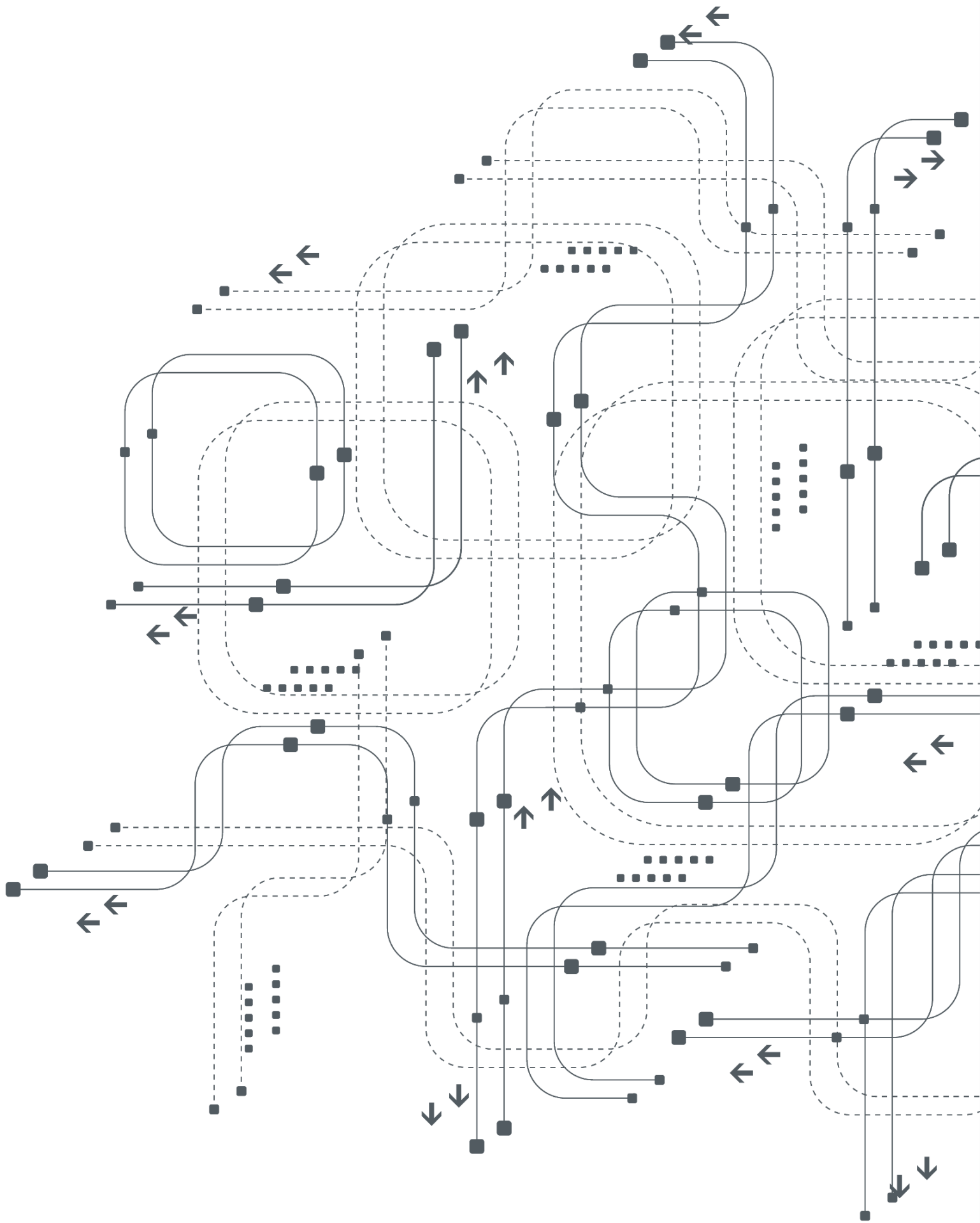
Acc: 83%

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>