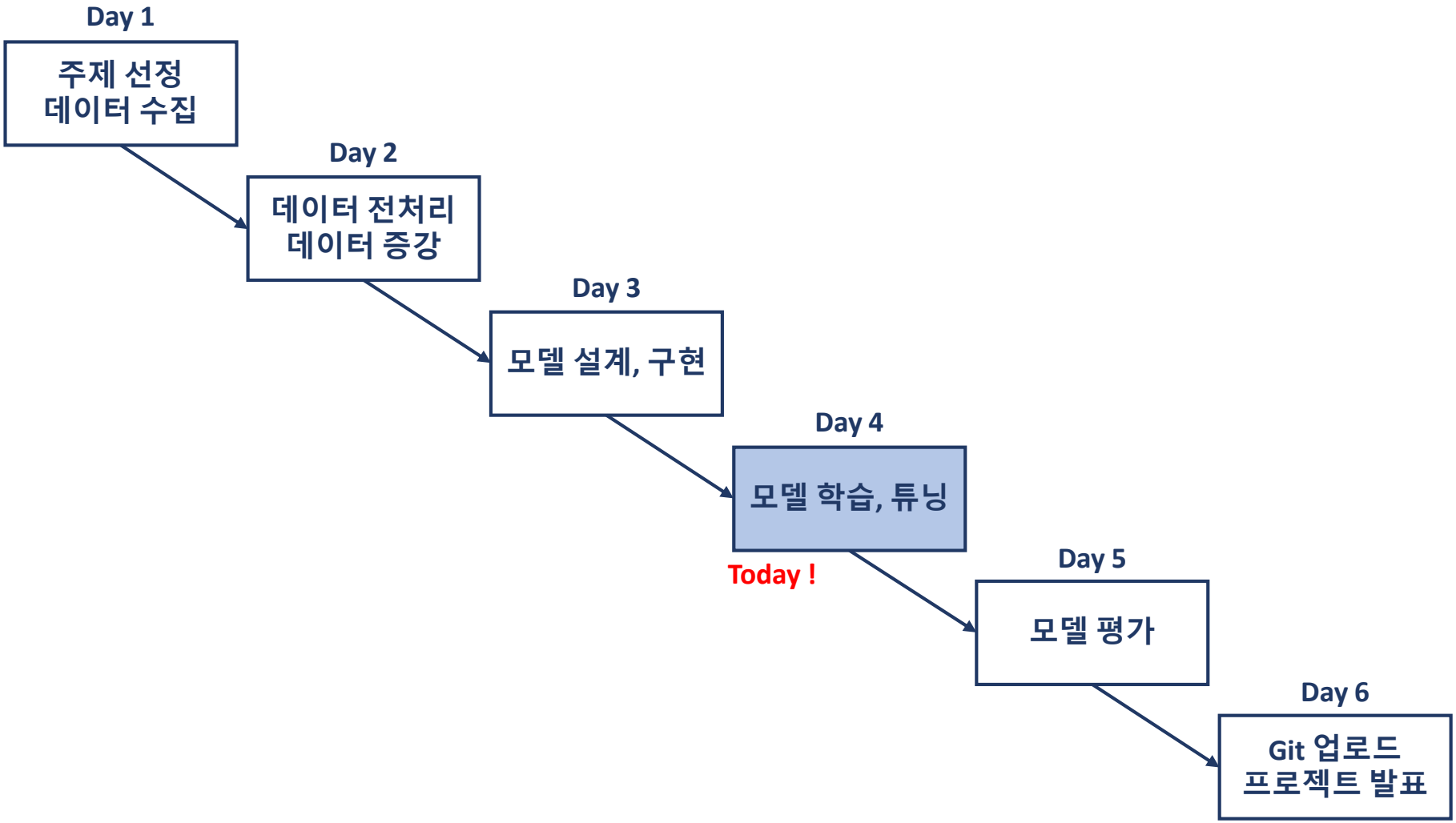
AI 기반 영상 데이터 분석 실습
- Day 4 -

index


Day 1
주제 선정
데이터 수집
Day 2
데이터 전처리
데이터 증강
Day 3
모델 설계, 구현
Day 4
모델 학습, 튜닝
Today !
Day 5
모델 평가
Day 6
Git 업로드
프로젝트 발표

Afternoon Lab session (실습)

Day 4 : Course overview

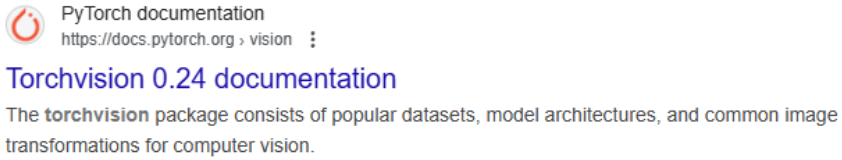
- Torch model 소개
- Architecture 선택 및 구현(Resnet + @)
- model 학습 및 hyper-parameter tunning

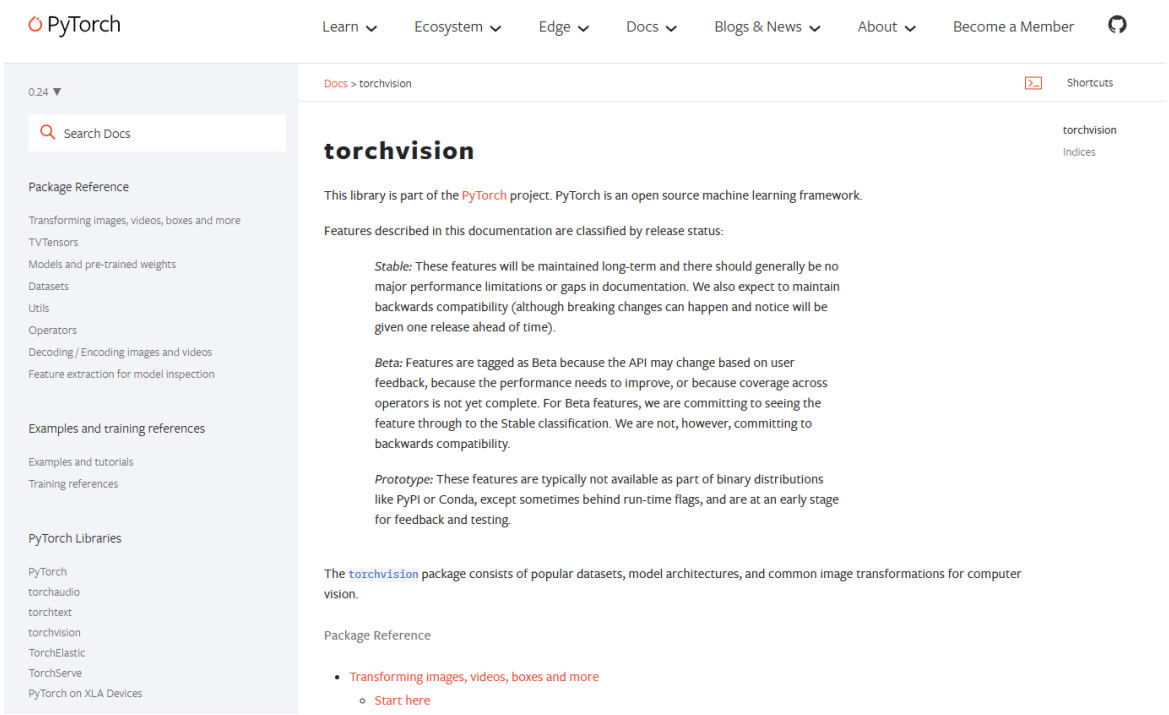
} **실습**

Torch model 소개

PyTorch homepage :

- https://docs.pytorch.org/vision

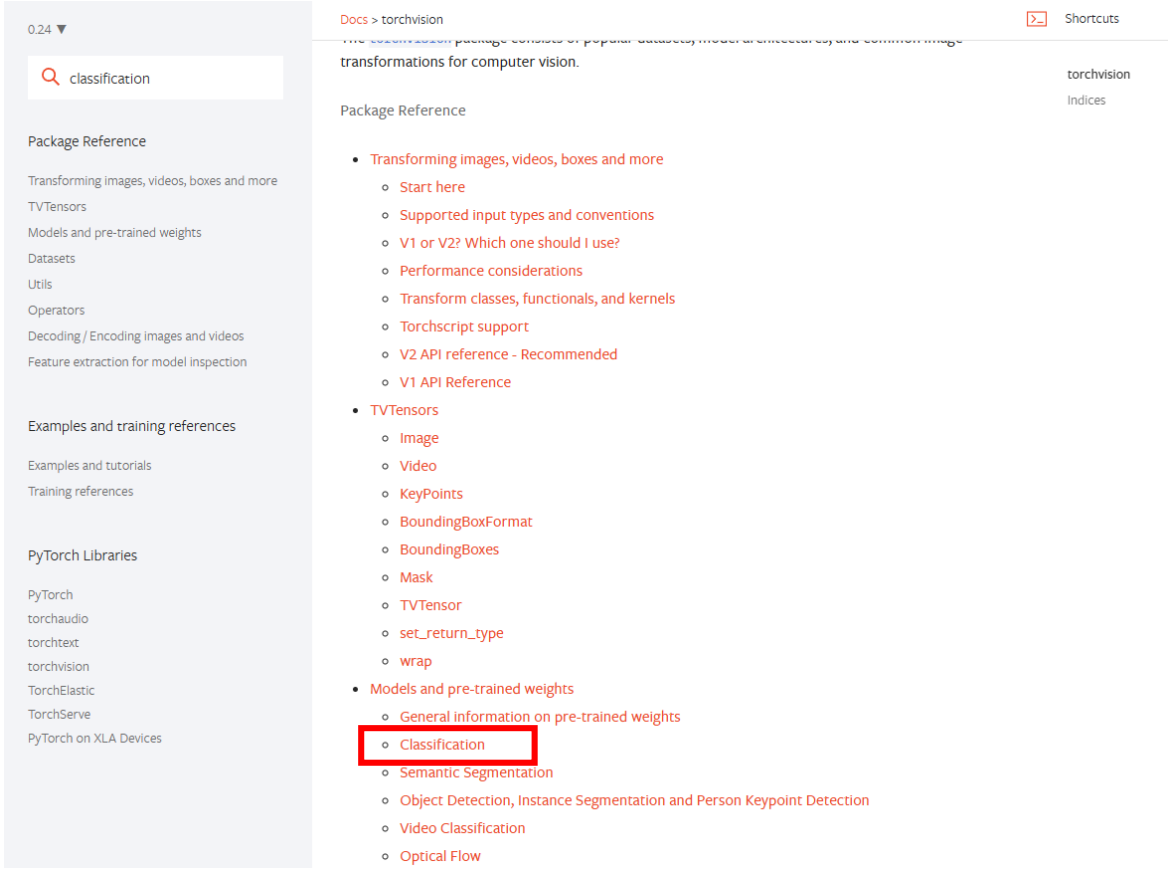




Torch model 소개

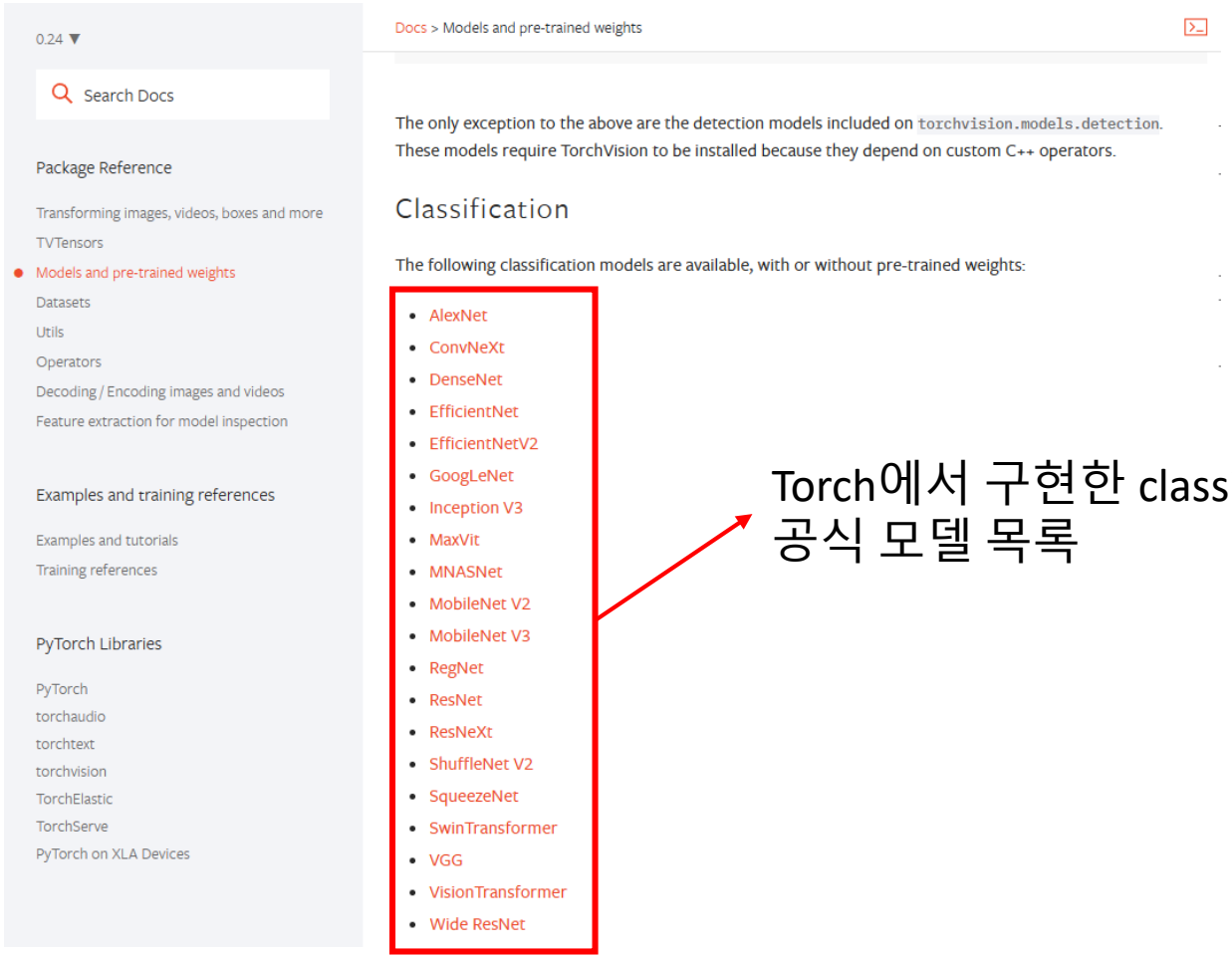
Torch vision -> Models and pre-trained weights

- **classification**



Torch model 소개

Pytorch 공식 구현 코드 목록 확인 (classification 용)

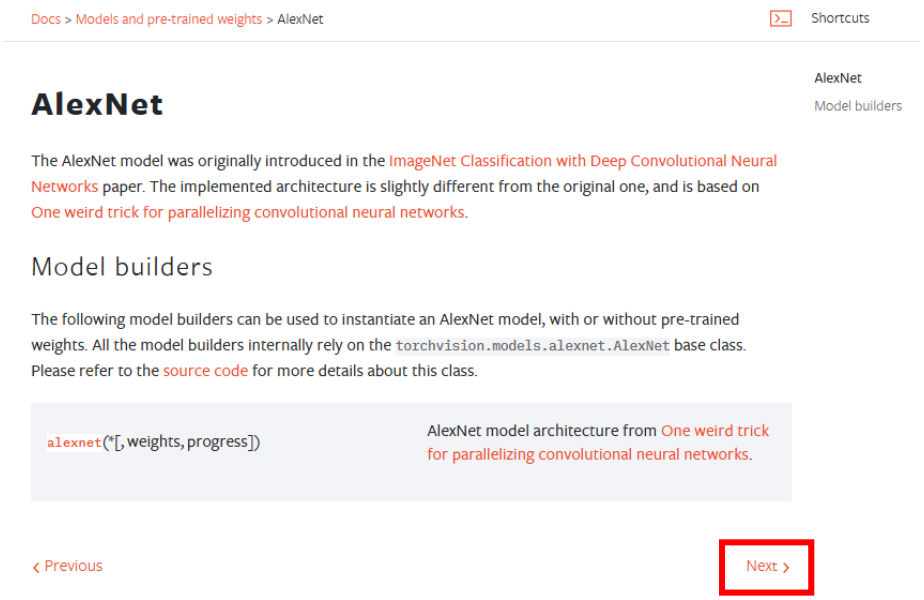


Torch에서 구현한 classification 용 공식 모델 목록

Torch model 소개

구현 코드 열람 및 사용

- Example : AlexNet



Docs > Models and pre-trained weights > AlexNet

Shortcuts

AlexNet

Model builders

AlexNet

The AlexNet model was originally introduced in the ImageNet Classification with Deep Convolutional Neural Networks paper. The implemented architecture is slightly different from the original one, and is based on One weird trick for parallelizing convolutional neural networks.

Model builders

The following model builders can be used to instantiate an AlexNet model, with or without pre-trained weights. All the model builders internally rely on the `torchvision.models.alexnet.AlexNet` base class. Please refer to the source code for more details about this class.

`alexnet`(*[, weights, progress])	AlexNet model architecture from One weird trick for parallelizing convolutional neural networks.

‹ Previous

Next ›

alexnet

torchvision.models.alexnet(*, *weights: Optional[AlexNet_Weights] = None*, *progress: bool = True*, ***kwargs: Any*) → AlexNet [SOURCE]

AlexNet model architecture from One weird trick for parallelizing convolutional neural networks.

• NOTE

AlexNet was originally introduced in the ImageNet Classification with Deep Convolutional Neural Networks paper. Our implementation is based instead on the "One weird trick" paper above.

Parameters:

- **weights** (`AlexNet_Weights`, optional) – The pretrained weights to use. See `AlexNet_Weights` below for more details, and possible values. By default, no pre-trained weights are used.
- **progress** (*bool*, *optional*) – If True, displays a progress bar of the download to stderr. Default is True.
- ****kwargs** – parameters passed to the `torchvision.models.squeezenet.AlexNet` base class. Please refer to the source code for more details about this class.

CLASS vision.models.AlexNet_Weights(*value*) [SOURCE]

The model builder above accepts the following values as the `weights` parameter. `AlexNet_Weights.DEFAULT` is equivalent to `AlexNet_Weights.IMAGENET1K_V1`. You can also use strings, e.g. `weights='DEFAULT'` or `weights='IMAGENET1K_V1'`.

Torch model 소개

코드 복제 및 사용 가능

- Example : AlexNet

Source code for torchvision.models.alexnet

```python
from functools import partial
from typing import Any, Optional

import torch
import torch.nn as nn

from ..transforms._presets import ImageClassification
from ..utils import _log_api_usage_once
from ._api import register_model, Weights, WeightsEnum
from ._meta import _IMAGENET_CATEGORIES
from ._utils import _ovewrite_named_param, handle_legacy_interface


__all__ = ["AlexNet", "AlexNet_Weights", "alexnet"]


class AlexNet(nn.Module):
    def __init__(self, num_classes: int = 1000, dropout: float = 0.5) -> None:
        super().__init__()
        _log_api_usage_once(self)
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
        self.classifier = nn.Sequential(
            nn.Dropout(p=dropout),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(p=dropout),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.features(x)
        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)
        return x
```

Torch model 소개

코드 사용 시 참고

Weights : 사전학습 가중치(체크포인트) 옵션들을 표준화해서 묶어둔 목록

Register_model : 함수/모델을 이름으로 찾고 문서화/자동화 가능하게 함

```
[docs]@register_model()
@handle_legacy_interface(weights=("pretrained", AlexNet_Weights.IMAGENET1K_V1))
def alexnet(*, weights: Optional[AlexNet_Weights] = None, progress: bool =
True, **kwargs: Any) -> AlexNet:
```

```
[docs]class AlexNet_Weights(WeightsEnum):
    IMAGENET1K_V1 = Weights(
        url="https://download.pytorch.org/models/alexnet-owt-7be5be79.pth",
        transforms=partial(ImageClassification, crop_size=224),
        meta={
            "num_params": 61100840,
            "min_size": (63, 63),
            "categories": _IMAGENET_CATEGORIES,
            "recipe":
"https://github.com/pytorch/vision/tree/main/references/classification#alexnet-
and-vgg",
            "_metrics": {
                "ImageNet-1K": {
                    "acc@1": 56.522,
                    "acc@5": 79.066,
                }
            },
            "_ops": 0.714,
            "_file_size": 233.087,
            "_docs": """
                These weights reproduce closely the results of the paper using
a simplified training recipe.
            """,
        },
    )
    DEFAULT = IMAGENET1K_V1
```

모델링 파이프라인


데이터 증강 정의
데이터셋 로드 정의 (2일차)
데이터셋 로더 구성
모델 & loss 등 정의 (3일차)
모니터링 & log용 기록 구성
Training 학습 루프 구성
Validation 학습 루프 구성
Best model 저장

Thank you