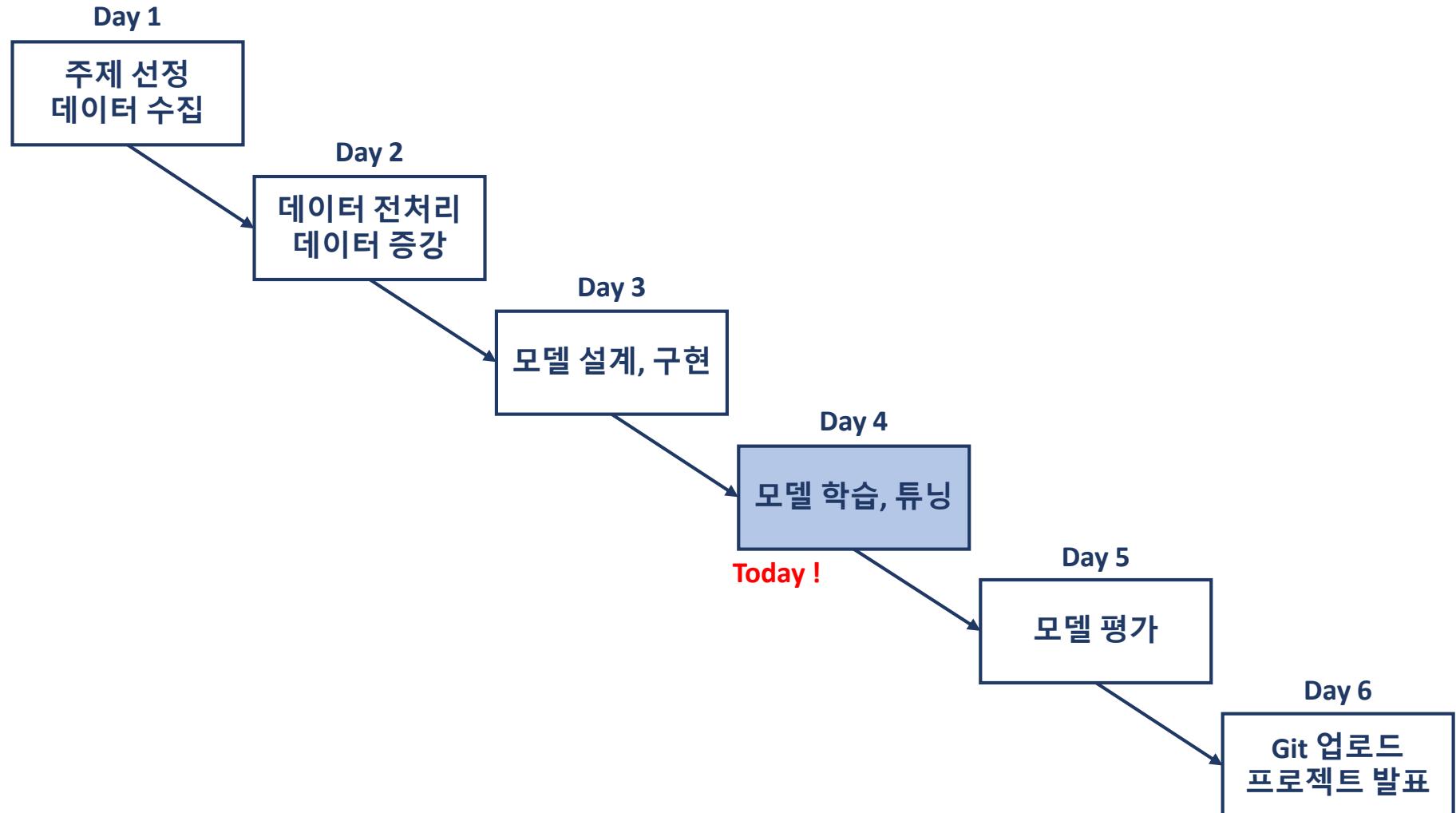


AI 기반 영상 데이터 분석 실습

- Day 4 -

index



Afternoon Lab session (실습)

Day 4 : Course overview

- Torch model 소개
 - Architecture 선택 및 구현(Resnet + @)
 - model 학습 및 hyper-parameter tuning
- }] 실습

Torch model 소개

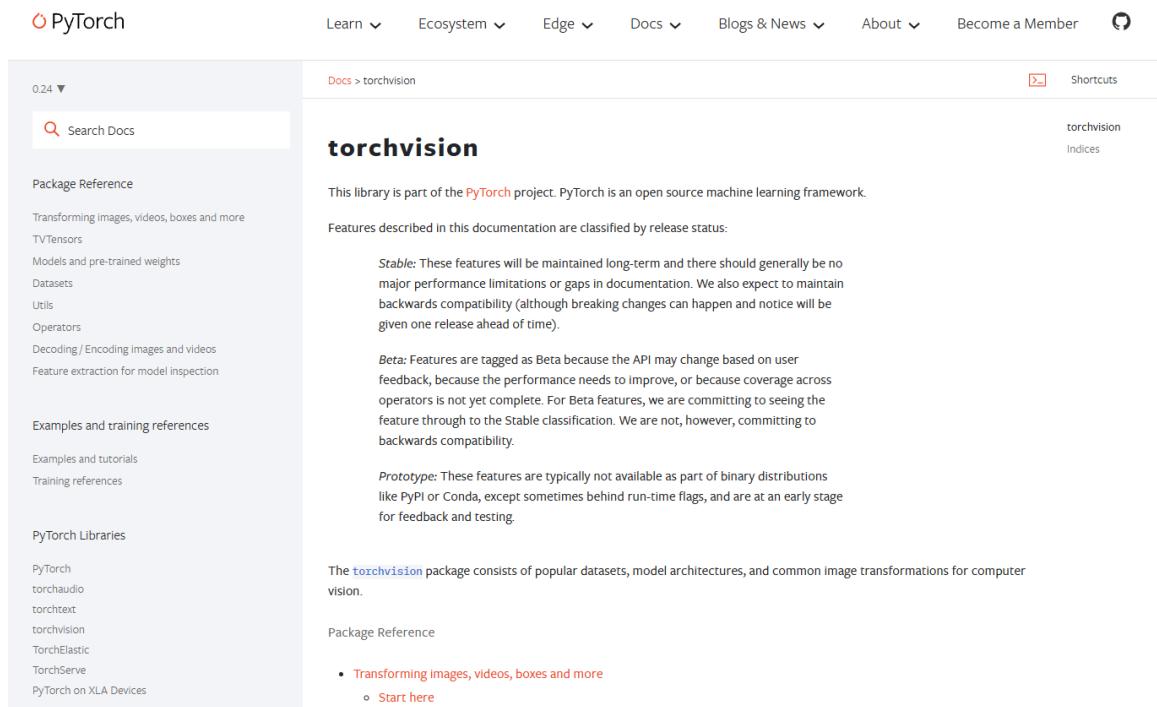
PyTorch homepage :

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

 PyTorch documentation
<https://docs.pytorch.org> :

Torchvision 0.24 documentation

The `torchvision` package consists of popular datasets, model architectures, and common image transformations for computer vision.



The screenshot shows the PyTorch documentation homepage. At the top, there's a navigation bar with links for Learn, Ecosystem, Edge, Docs, Blogs & News, About, and Become a Member. On the far right is a user profile icon. Below the navigation bar, the main content area has a header "torchvision". To the left is a sidebar with sections like Package Reference, Examples and training references, and PyTorch Libraries. The main content area contains information about the `torchvision` library, its release status (Stable, Beta, Prototype), and a note about its purpose. There are also links to Package Reference and specific documentation pages like "Transforming images, videos, boxes and more".

Torch model 소개

Torch vision -> Models and pre-trained weights

- classification

The screenshot shows the PyTorch torchvision documentation page. At the top, there is a search bar with the query "classification". Below the search bar, there are three main sections: "Package Reference", "Examples and training references", and "PyTorch Libraries". The "Package Reference" section is expanded, showing sub-sections like "Transforming images, videos, boxes and more", "TVTensors", "Models and pre-trained weights", etc. The "Classification" section under "Models and pre-trained weights" is highlighted with a red box. The right side of the page has a sidebar with "Shortcuts" (torchvision, Indices) and a navigation bar with "Docs > torchvision".

Docs > torchvision
The `torchvision` package consists of popular datasets, model architectures, and common image transformations for computer vision.

Package Reference

- Transforming images, videos, boxes and more
 - Start here
 - Supported input types and conventions
 - V1 or V2? Which one should I use?
 - Performance considerations
 - Transform classes, functionals, and kernels
 - Torchscript support
 - V2 API reference - Recommended
 - V1 API Reference
- TVTensors
 - Image
 - Video
 - Keypoints
 - BoundingBoxFormat
 - BoundingBoxes
 - Mask
 - TVTensor
 - set_return_type
 - wrap
- Models and pre-trained weights
 - General information on pre-trained weights
 - Classification Classification
 - Semantic Segmentation
 - Object Detection, Instance Segmentation and Person Keypoint Detection
 - Video Classification
 - Optical Flow

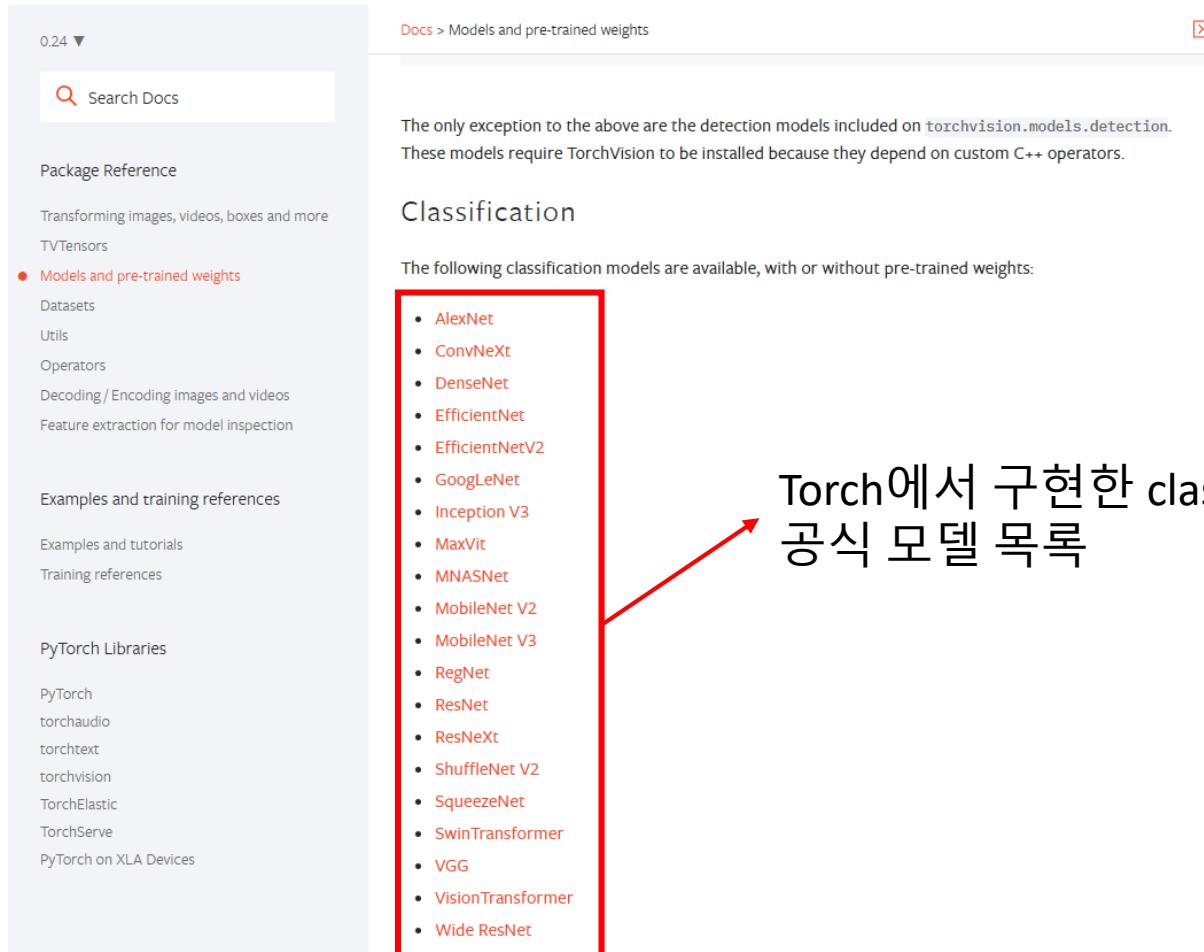
Shortcuts

torchvision

Indices

Torch model 소개

Pytorch 공식 구현 코드 목록 확인 (classification 용)



The screenshot shows a browser window displaying the PyTorch documentation. The URL in the address bar is `Docs > Models and pre-trained weights`. The page content discusses detection models and then lists classification models. A red box highlights the list of classification models, which includes:

- AlexNet
- ConvNeXt
- DenseNet
- EfficientNet
- EfficientNetV2
- GoogLeNet
- Inception V3
- MaxVit
- MNASNet
- MobileNet V2
- MobileNet V3
- RegNet
- ResNet
- ResNeXt
- ShuffleNet V2
- SqueezeNet
- SwinTransformer
- VGG
- VisionTransformer
- Wide ResNet

A red arrow points from the text "Torch에서 구현한 classification 용 공식 모델 목록" to the highlighted list.

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

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 pretrained 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

```
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

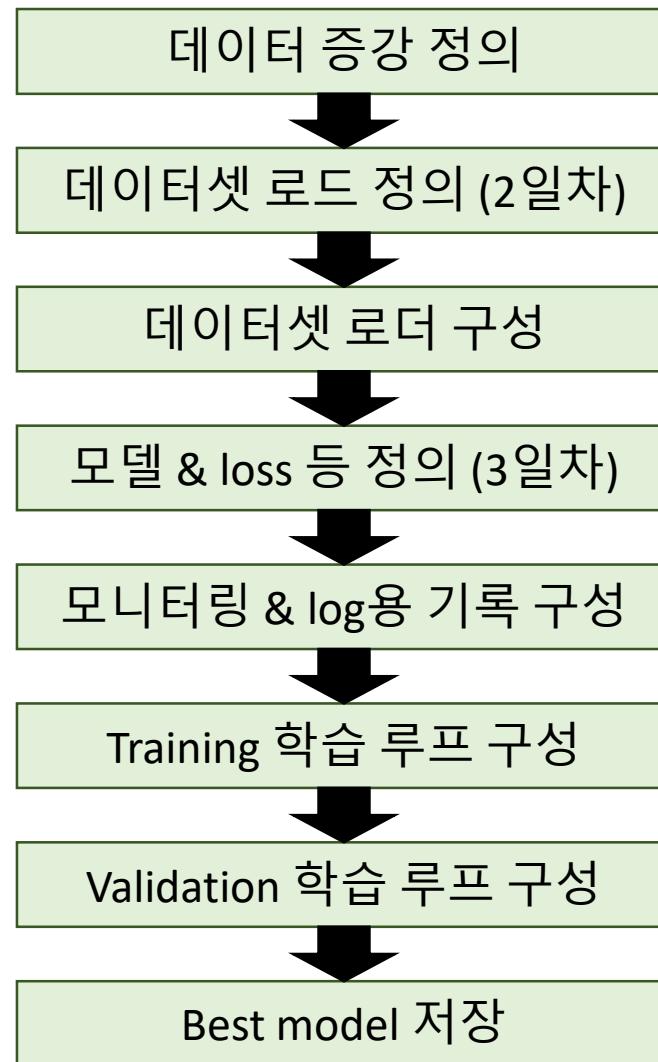
Weights : 사전학습 가중치(체크포인트) 옵션들을 표준화해서 묶어둔 목록

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



Thank you