Lab 4: CNN & Image Processing

<삼성 AI 전문가 교육과정> 실습

서울대학교 바이오지능 연구실 (장병탁 교수)

최원석, 김윤성

2022.06.09

Biointelligence Laboratory

Dept. of Computer Science and Engineering

Seoul National University

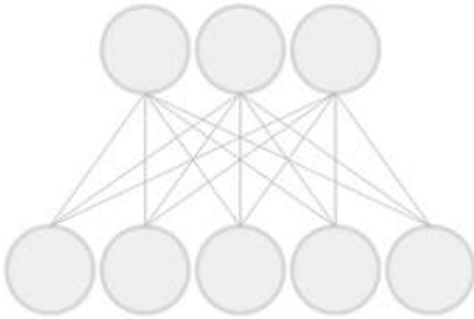
BI

BIOINTELLIGENCE

CNN implementation

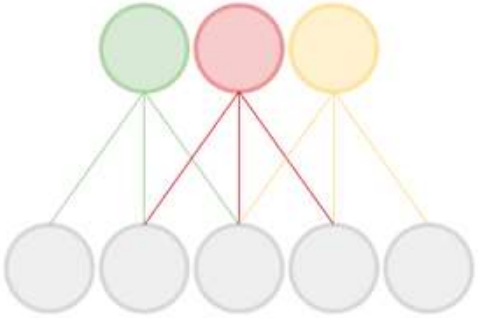
background

- Linear layer
  - 입력의 개수가 n, 출력의 개수가 m
    - $\boldsymbol{m \times n}$**가중치 행렬**, 길이 m bias vector
    - 하나의 출력 노드가 모든 입력값을 참조



Fully connected layer

- CNN
  - 이미지 형태의 입력은 보통 거리가 가까운 픽셀에 중요한 정보가 많이 담겨있음 (지리적 지역성)
  - 의존도가 적은(거리가 먼) parameter를 제거
    - 단일 층의 표현력을 최대한 유지하면서
    - Cost를 크게 줄여 더 높은 층을 쌓을 수 있도록 함

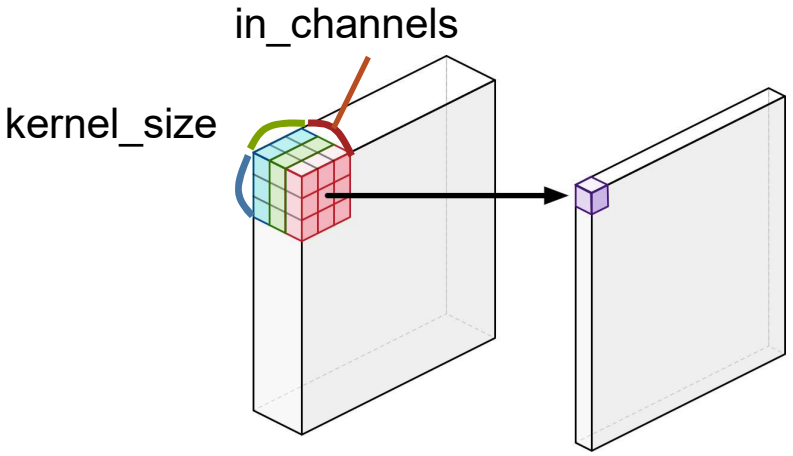


Convolutional layer

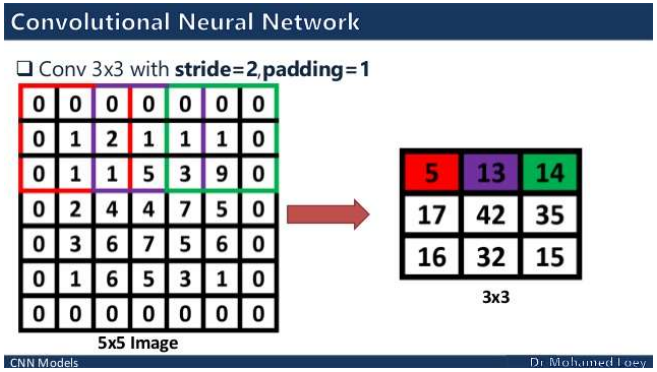
출처:https://www.quora.com/What-is-the-difference-between-a-convolutional-neural-network-and-a-multilayer-perceptron

CNN model

- nn.Conv2d(*in_channels*, *out_channels*, *kernel_size*, *stride*, *padding*)
  - out_channels: 사용할 커널의 개수
  - stride: 한번에 넘길 픽셀 개수 (default: 1)
  - padding: 입력의 가장자리에 padding만큼의 빈값을 채워넣음(default: 0)
  - https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md



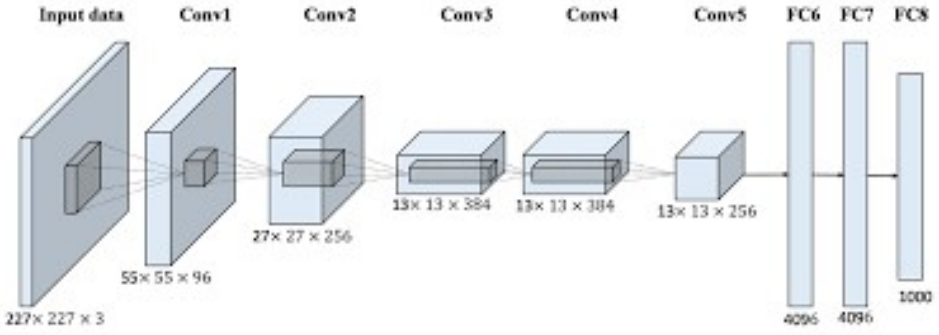
출처:https://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/



출처:https://www.slideshare.net/mohamedloey/convolutional-neural-network-models-deep-learning

CNN model

- Stride vs Pooling
  - Compression을 위한 2가지 method
  - 최근 추세는 stride 방법을 선호
    - 계산량이 더 적음
    - https://stats.stackexchange.com/questions/387482/pooling-vs-stride-for-downsampling



- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

CNN classifier implementation

- CNN classifier model
  - nn.Conv2d 레이어를 사용
    - 2d input의 batch norm을 위해 nn.BatchNorm2d 함수 사용
    - $[1, 28, 28] \xrightarrow[\text{conv1}]{} [8,12,12]$

      $\xrightarrow[\text{conv2}]{} [8,4,4] \xrightarrow[\text{conv3}]{} [8,2,2]$
  - 컨볼루션 레이어 사용 이후에 마지막으로 fully connected layer로 최종 결과 반환
    - $[8,2,2] \xrightarrow[\text{view}]{} [32] \xrightarrow[\text{fc}]{} [10]$

```
class MNISTCNN(nn.Module):
    def __init__(self,IMG_SIZE=28):
        super(MNISTCNN,self).__init__()
        self.conv1 = nn.Conv2d(1,8,5,stride=2)
        self.BN1 = torch.nn.BatchNorm2d(8)
        self.conv2 = nn.Conv2d(8,8,5,stride=2)
        self.BN2 = torch.nn.BatchNorm2d(8)
        self.conv3 = nn.Conv2d(8,8,3,stride=1)
        self.fc = nn.Linear(8*2*2,10)

    def forward(self,x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.BN1(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = self.BN2(x)
        x = self.conv3(x)
        x = F.relu(x)
        x = x.view(-1,8*2*2)
        x = self.fc(x)
        x = torch.softmax(x,dim=-1)
        return x
```

Training results

- 학습 결과
  - 약 98% 정도의 accuracy
  - 학습 파라메터: **2762**개
    - Layer 개수: 2 → 4
    - 파라메터 수: 25514 → 2762
    - 정확도: 97.2% → 98.0%

```
Epoch 26, Loss(train) : 346.68723475933075
Acc(test) : 0.98828125
Epoch 27, Loss(train) : 346.658863902092
Epoch 28, Loss(train) : 346.5927355289459
Acc(test) : 0.98828125
Epoch 29, Loss(train) : 346.5126872062683
Epoch 30, Loss(train) : 346.40869534015656
Acc(test) : 0.9765625
Acc(test) : 0.98046875
```

Codes

- 실습 코드 Github 주소
  - https://github.com/yskim5892/AI_Expert_2022

Lab 4.5: Image Processing

<삼성 AI전문가 교육과정> 실습
서울대학교 바이오지능 연구실 (장병탁 교수)
이현도, 최원석

2021.06.22

Biointelligence Laboratory
Dept. of Computer Science and Engineering
Seoul National University

BI
BIOINTELLIGENCE

Contents

- Image processing
  - Torchvision datasets
  - Using custom datasets
  - Loading pretrained models
  - Object detection models

Image processing

Datasets

- Dataset class
  - data, label pair로 구성
  - DataLoader의 인자로 넣어, iteration마다 dataset batch를 반환

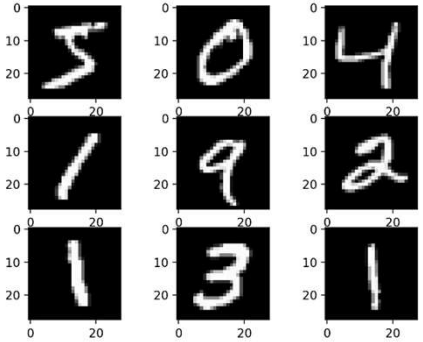
```
trainset = MNIST('root',train=True,transform=transforms,download=True)
train_loader = DataLoader(trainset, num_workers=1,batch_size=BATCH_SIZE,shuffle=True)
for x,y in train_loader:
    ...
```

- Torchvision datasets
  - 간략한 소개

- Custom datasets
  - 오픈 데이터셋
  - 구성 방법
  - 전처리 과정

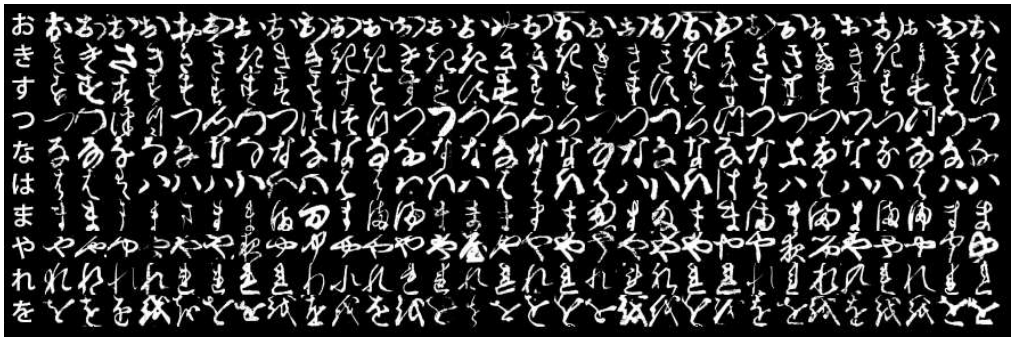
Torchvision datasets

- MNIST 계열
  - MNIST
    - Handwritten digits



  - Fashion-MNIST
    - Grayscale cloth images
    - 난이도 높은 MNIST

  - KMNIST
    - 일본어 MNIST

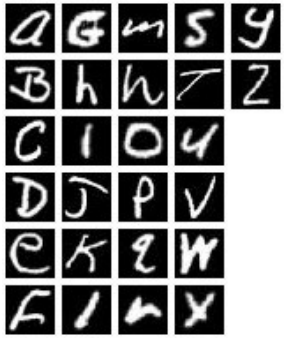


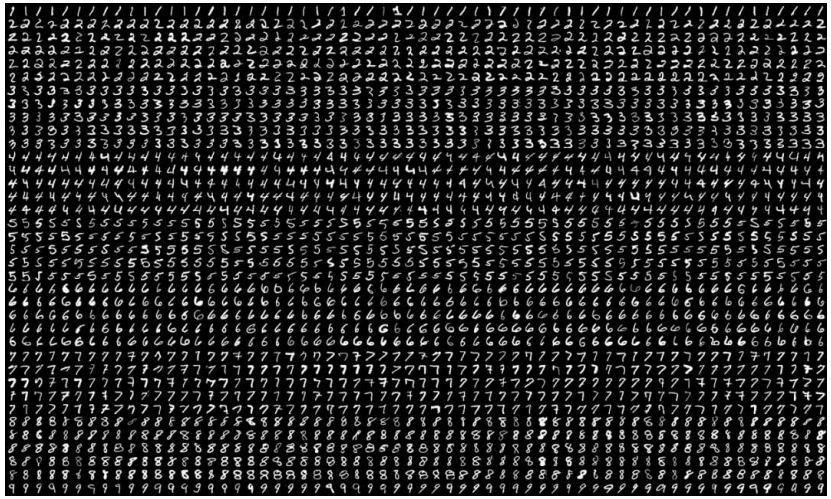
Datasets

- MNIST
- Fashion-MNIST
- KMNIST
- EMNIST
- QMNIST
- FakeData
- COCO
  - Captions
  - Detection
- LSUN
- ImageFolder
- DatasetFolder
- ImageNet
- CIFAR
- STL10
- SVHN
- PhotoTour
- SBU
- Flickr
- VOC
- Cityscapes
- SBD
- USPS
- Kinetics-400
- HMDB51
- UCF101

Torchvision datasets

- MNIST 계열
  - EMNIST
    - Letters + digit
  - QMNIST
    - 데이터 수 확장
- FakeData
  - 랜덤 생성된 이미지





Datasets

- MNIST
- Fashion-MNIST
- KMNIST
- EMNIST
- QMNIST
- FakeData
- COCO
  - Captions
  - Detection
- LSUN
- ImageFolder
- DatasetFolder
- ImageNet
- CIFAR
- STL10
- SVHN
- PhotoTour
- SBU
- Flickr
- VOC
- Cityscapes
- SBD
- USPS
- Kinetics-400
- HMDB51
- UCF101

Torchvision datasets

- Real images
  - CocoDetection
    - Segmented images

  - CocoCaption
    - Classification 대신 caption(설명)

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

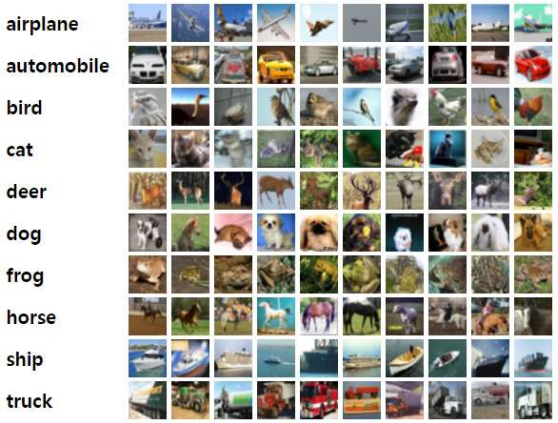
  - LSUN
    - 약 100만개의 scene
    - 10 scene, 20 object categories

Datasets

- MNIST
- Fashion-MNIST
- KMNIST
- EMNIST
- QMNIST
- FakeData
- COCO
  - Captions
  - Detection
- LSUN
- ImageFolder
- DatasetFolder
- ImageNet
- CIFAR
- STL10
- SVHN
- PhotoTour
- SBU
- Flickr
- VOC
- Cityscapes
- SBD
- USPS
- Kinetics-400
- HMDB51
- UCF101

Torchvision datasets

- Real images
  - ImageNet
    - 2만개 이상의 class, 1400만장의 image
  - CIFAR10, CIFAR100
    - 클래스가 각각 10, 100개
    - 총 6만장의 가벼운 데이터셋
  - STL10
    - 작은 labeled dataset과 10만개의 unlabeled image
    - Semi-supervised learning

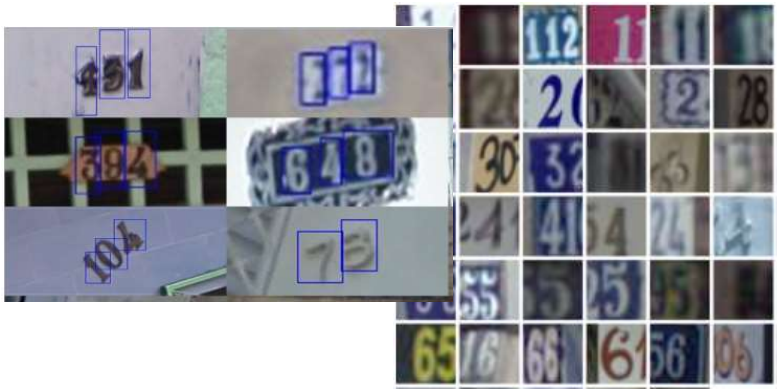


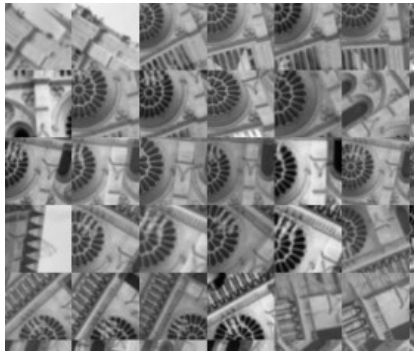
Datasets

- MNIST
- Fashion-MNIST
- KMNIST
- EMNIST
- QMNIST
- FakeData
- COCO
  - Captions
  - Detection
- LSUN
- ImageFolder
- DatasetFolder
- ImageNet
- CIFAR
- STL10
- SVHN
- PhotoTour
- SBU
- Flickr
- VOC
- Cityscapes
- SBD
- USPS
- Kinetics-400
- HMDB51
- UCF101

Torchvision datasets

- Real images
  - SVHN
    - 길거리의 house number 데이터
    - Cropped, uncropped
  - PhotoTour
    - 여러 방향으로 찍은 데이터
  - SBU
    - Image captioning dataset



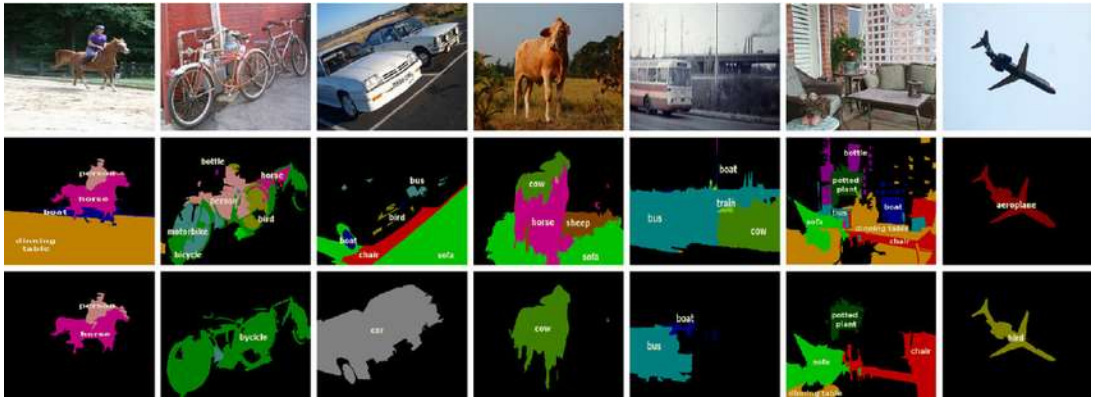


Datasets

- MNIST
- Fashion-MNIST
- KMNIST
- EMNIST
- QMNIST
- FakeData
- COCO
  - Captions
  - Detection
- LSUN
- ImageFolder
- DatasetFolder
- ImageNet
- CIFAR
- STL10
- SVHN
- PhotoTour
- SBU
- Flickr
- VOC
- Cityscapes
- SBD
- USPS
- Kinetics-400
- HMDB51
- UCF101

Torchvision datasets

- Real images
  - Flickr8k, Flickr30k
    - Flickr 에서 모은 8,000장의 이미지에 캡션이 포함됨
    - 각 캡션마다 점수가 포함
  - VOCDetection, VOCSegmentation
    - PASCAL VOC challenge의 detection, segmentation 데이터셋

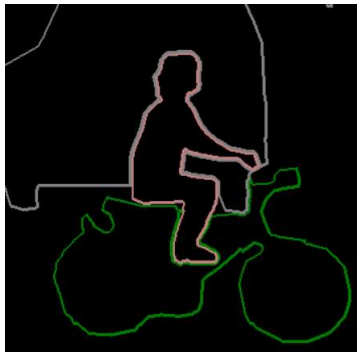


Datasets

- MNIST
- Fashion-MNIST
- KMNIST
- EMNIST
- QMNIST
- FakeData
- COCO
  - Captions
  - Detection
- LSUN
- ImageFolder
- DatasetFolder
- ImageNet
- CIFAR
- STL10
- SVHN
- PhotoTour
- SBU
- Flickr
- VOC
- Cityscapes
- SBD
- USPS
- Kinetics-400
- HMDB51
- UCF101

Torchvision datasets

- Real images
  - Cityscapes
    - 거리 사진의 segmented image
  - SBD
    - Semantic Boundary Dataset



- Video, motion dataset
  - Kinetics-400
    - action recognition video dataset
  - HMDB51
    - Human Motion dataset
  - UCF101
    - action recognition video dataset

Datasets

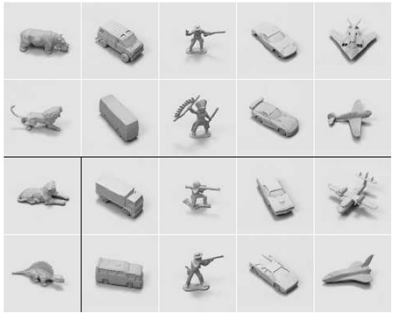
- MNIST
- Fashion-MNIST
- KMNIST
- EMNIST
- QMNIST
- FakeData
- COCO
  - Captions
  - Detection
- LSUN
- ImageFolder
- DatasetFolder
- ImageNet
- CIFAR
- STL10
- SVHN
- PhotoTour
- SBU
- Flickr
- VOC
- Cityscapes
- SBD
- USPS
- Kinetics-400
- HMDB51
- UCF101

Custom datasets

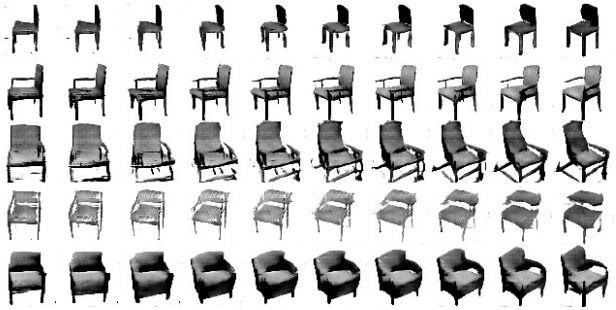
- 이미지를 가지고 직접 데이터셋을 생성
  - Open dataset을 다운받아서 생성, 혹은
  - 직접 데이터를 모아서 생성 (사내 수집 데이터 등)

- Open datasets
  - 풀고자 하는 task에 맞춰서 검색
    - 비슷한 논문을 검색해보면 어떤 데이터셋을 사용했는지 잘 나와있음
  - 참고할 만한 링크
    - https://deepestdocs.readthedocs.io/en/latest/003_image_processing/0031/
    - https://blog.cambridgespark.com/50-free-machine-learning-datasets-image-datasets-241852b03b49

Open datasets



NORB



3D chairs

CelebA

DeepFashion

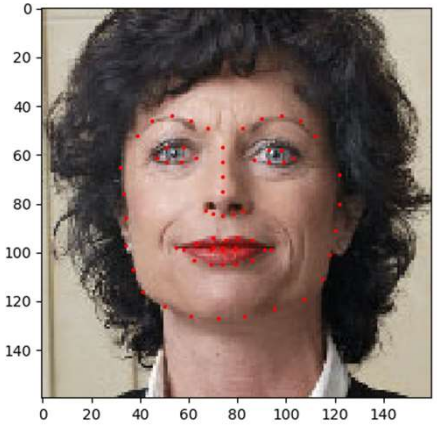
Writing custom datasets

- torchvision.datasets.ImageFolder
  - 각 class label을 이름으로 갖는 folder 내부에 데이터가 존재
  - Root directory를 지정해주는 것 만으로 자동으로 label이 포함된 dataset 생성

```
root/dog/xxx.png
root/dog/xxy.png
root/dog/xxz.png

root/cat/123.png
root/cat/nsdf3.png
root/cat/asd932_.png
```

- Inheriting torch.utils.data.Dataset
  - 복잡한 label을 가지거나, 기타 customize가 필요한 경우
  - .csv 파일에 복잡한 label 정보를 저장
  - *__getitem__* method 내부에 이미지와, csv 파일을 읽어 해당 label을 추출하는 과정 필요
  - https://pytorch.org/tutorials/beginner/data_loading_tutorial.html



preprocessing

- torchvision.transforms
  - 이미지 전처리를 위한 기본 함수들
    - 이미지 크기 조정
    - Bounding box를 기준으로 cropping
    - Data augmentation : random cropping, color jittering, grayscale, …
    - 이미지 회전 및 뒤집기
  - https://pytorch.org/docs/stable/torchvision/transforms.html
  - Compose 함수
    - 기본 함수들을 모아서, 하나의 함수인 것처럼 합쳐주는 함수

preprocessing

- torchvision.transforms
  - Ex) dataset input format

흑백 이미지
64 x 64
-1 ~ 1 tensor

```
datasets.ImageFolder(root='datasets/3dfaces/',
                     transform=transforms.Compose([
                         transforms.Grayscale(),
                         transforms.Resize((img_size, img_size)),
                         transforms.ToTensor(),
                         transforms.Normalize(mean=(0.5,), std=(0.5,))
                     ]))
```

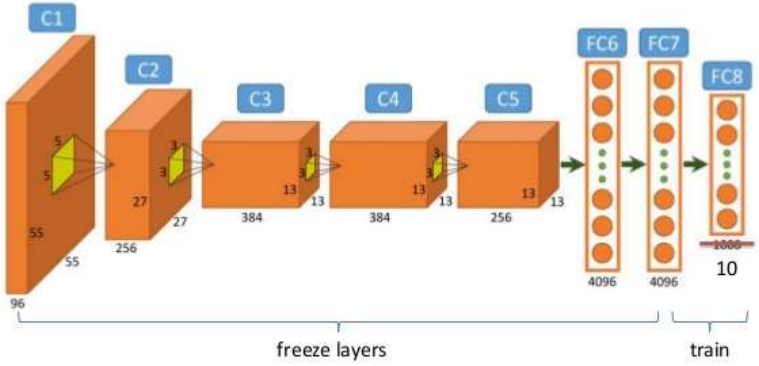
Pretrained image classifier models

- torchvision.models
  - https://pytorch.org/docs/stable/torchvision/models.html
  - 유명한 image classifier를 미리 구현한 모델을 import 할 수 있게끔 지원
    - ImageNet 데이터셋에 대해서 pretrained된 모델 로드
    - 로드 후 다른 데이터에 대해 학습 가능 (fine-tuning)
  - 다른 데이터셋에서 동작하는 모델을 만들고 싶을 경우
    - transfer learning (전이학습)

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNet v2
- ResNeXt
- Wide ResNet
- MNASNet

Transfer learning

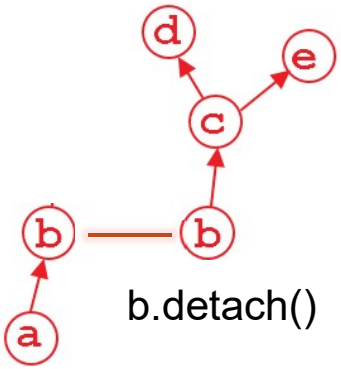
- 2 cases
  - Pre-trained된 모델을 feature extractor로 두는 경우
    - 마지막 layer를 제외한 weight를 freeze시킨 채, 마지막 레이어만 주어진 데이터셋에 맞게 학습
  - Fine-tuning
    - 마지막 layer뿐만이 아닌, 하위 layer들도 조금씩 학습시키는 경우
    - 모델의 초기화를 pre-trained 모델의 weight로 두어 학습



Transfer learning

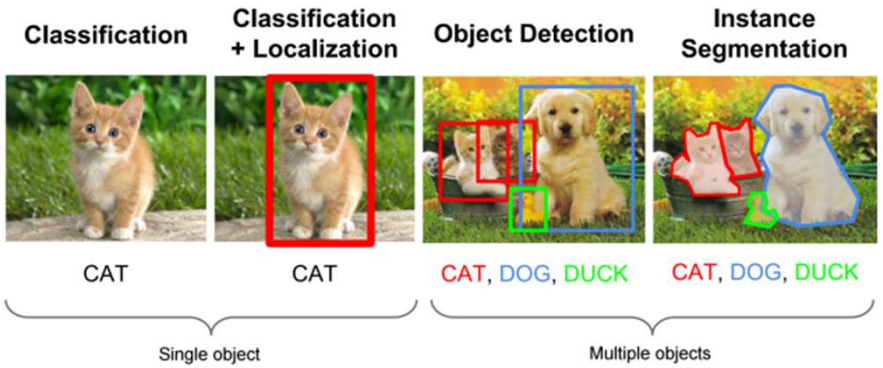
- Tutorials
  - https://tutorials.pytorch.kr/beginner/transfer_learning_tutorial.html
  - https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

- 일부 parameter를 freeze시키는 방법
  - 학습 대상으로 지정하지 않는 방법
    - Optimizer 초기화 시 parameter에 pre-trained model을 넣지 않는다
  - Gradient를 차단하는 방법
    - .detach() 함수 활용
    - param.requires_grad를 False로 지정



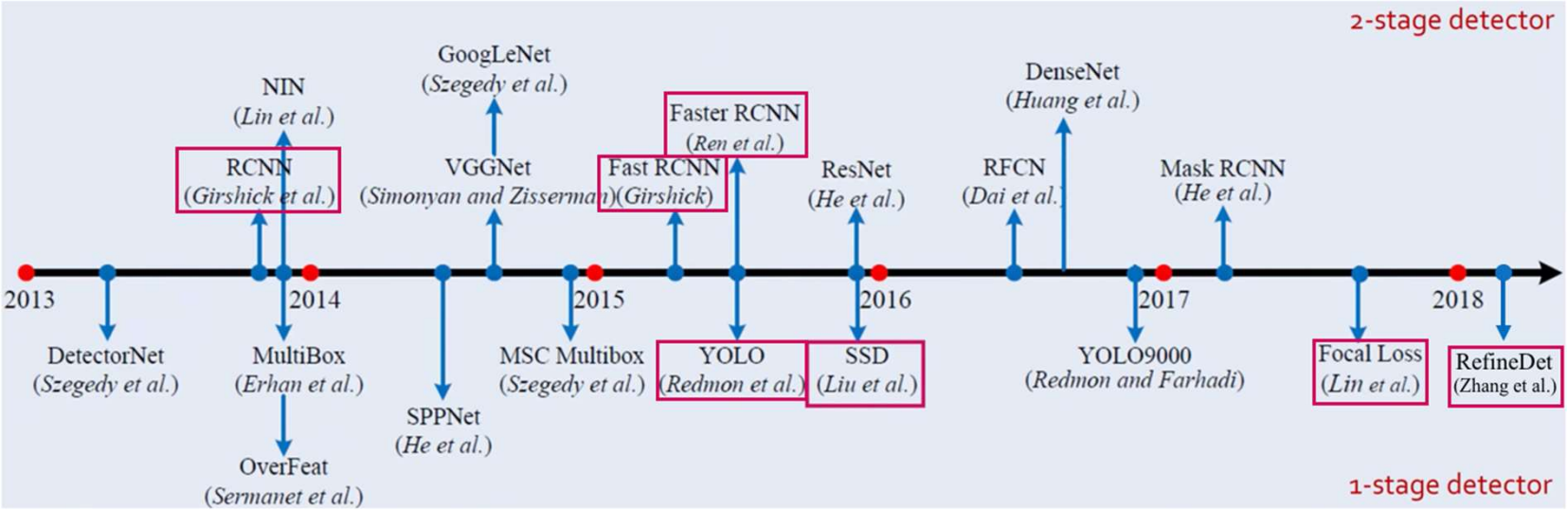
Object detection

- Image classification
  - (일반적으로) 하나의 오브젝트를 분류
  - 대부분의 이미지는 다수의 오브젝트 존재



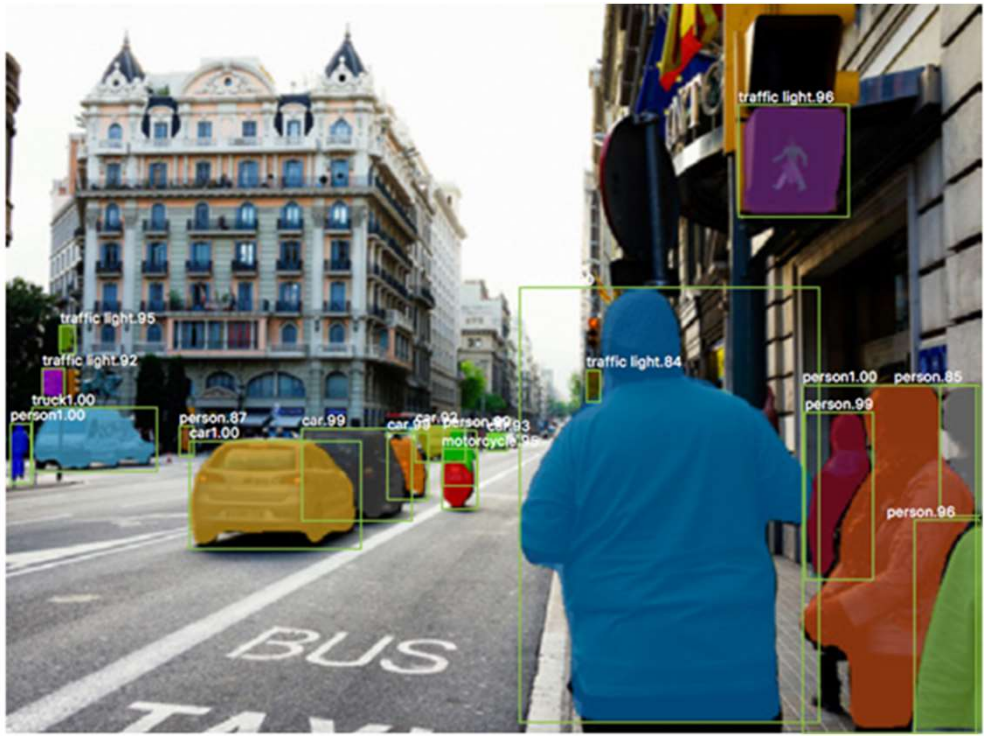
- Object detection, segmentation
  - 한 화면에서 다수의 object를 감지하고 물체의 영역까지 추정
    - Object detection – 물체의 bounding box 위치를 추정
      - Top-left, bottom-right 좌표 2가지만 추정하면 됨
    - Instance segmentation – 물체의 정확한 경계를 추정
      - 픽셀 단위로 어떤 물체에 포함되어 있는지를 결정

Object detection

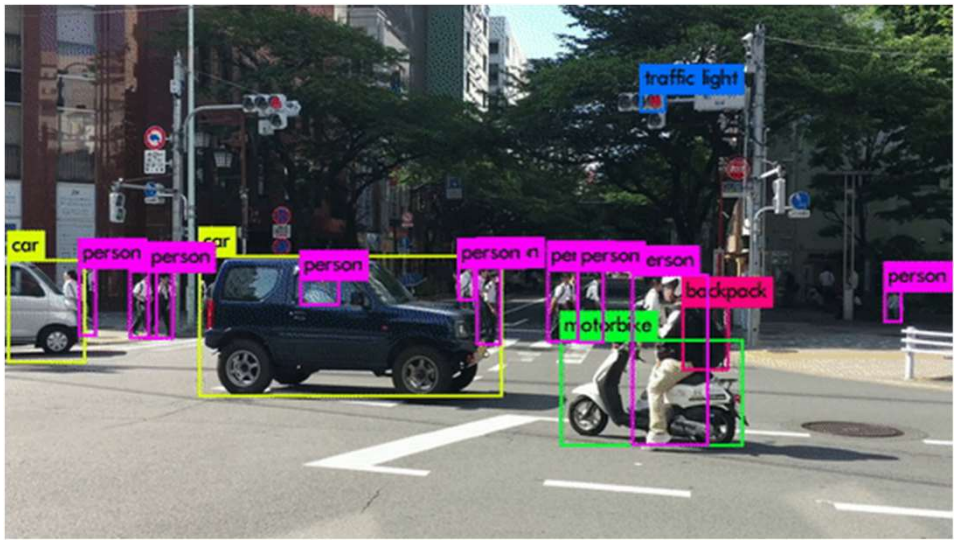


- 2가지 계열
  - 2-stage detector: localization, classification 문제를 순차적으로 해결
    - RCNN 계열 / 정확도
  - 1-stage detector: localization, classification 문제를 동시에 해결
    - YOLO / 속도

Object detection



Mask RCNN

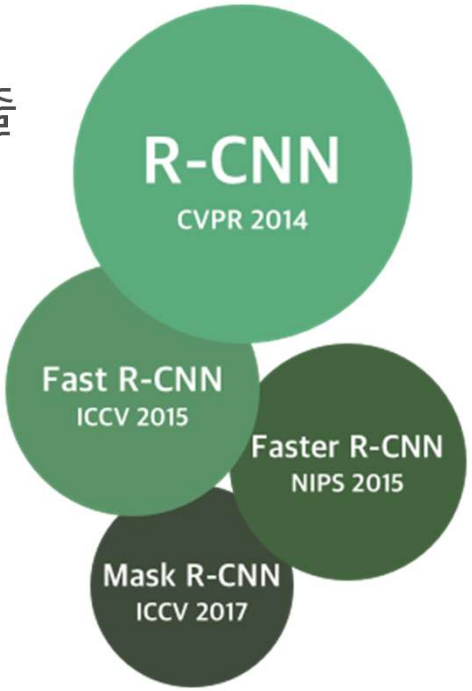


YOLOv3

[Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR 2014]

2-stage detector

- RCNN
  - Region Proposal - 물체의 영역을 찾는 모듈
  - CNN - 각각의 영역으로부터 고정된 크기의 Feature Vector 추출
  - SVM - Classification 을 위한 선형 지도학습 모델



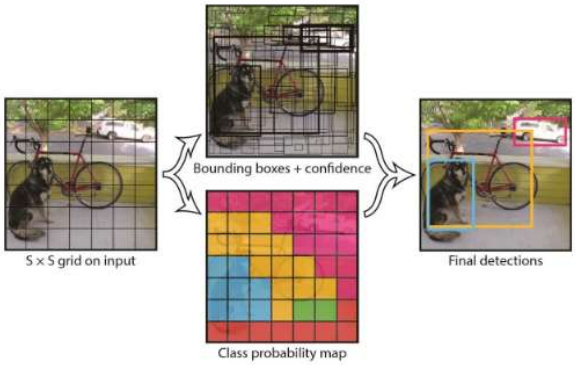
- Fast, Faster RCNN
  - CNN을 한 번만 수행하여 연산
  - Region Proposal 과정에 selective search 대신 CNN 사용

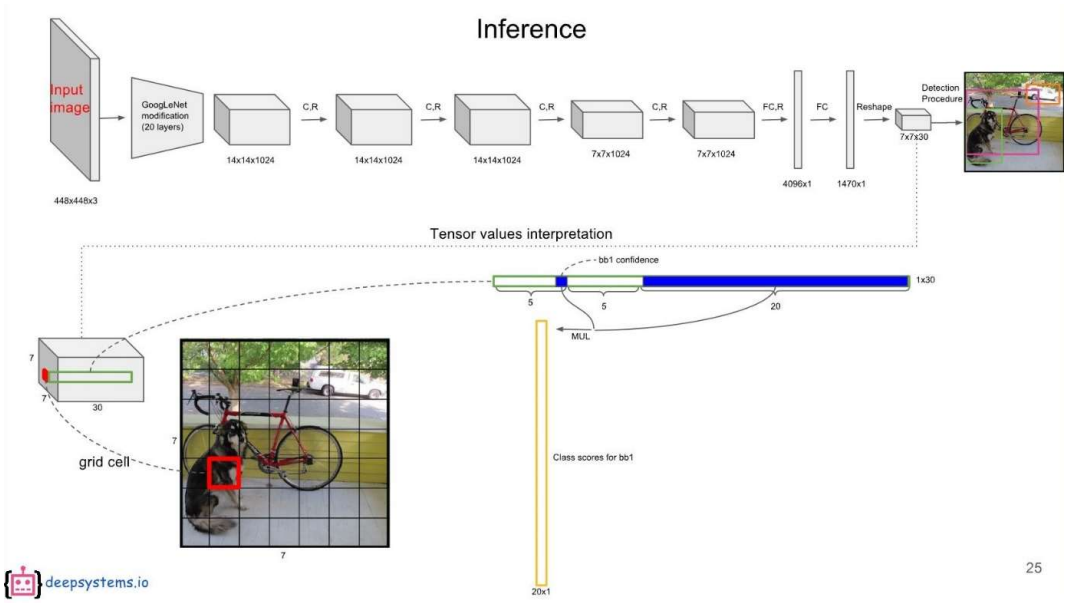
- Mask RCNN
  - 각 픽셀이 객체에 해당하는 것인지 아닌지를 masking하는 CNN을 추가

[You Only Look Once: Unified, Real-Time Object Detection , CVPR 2016]

1-stage detector

- YOLO(You Only Look Once)
  - Input image를 S X S grid로 나눈다.
  - 각 grid cell은 bounding box 위치와 confidence score를 예측
  - 동시에 각 cell에서 모든 class에 대한 conditional class probability 예측
  - 두 값을 곱하여 bounding box의 class-specific confidence score를 구한다.





YOLO implementation

- Pytorch 구현
  - https://github.com/eriklindernoren/PyTorch-YOLOv3
- 상세 구현 + 구현 설명
  - https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

Codes

- 실습 코드 Github 주소
  - https://github.com/yskim5892/AI_Expert_2022