

Apply to Machine Learning

Machine Learning

이선우(Seon Woo Lee)

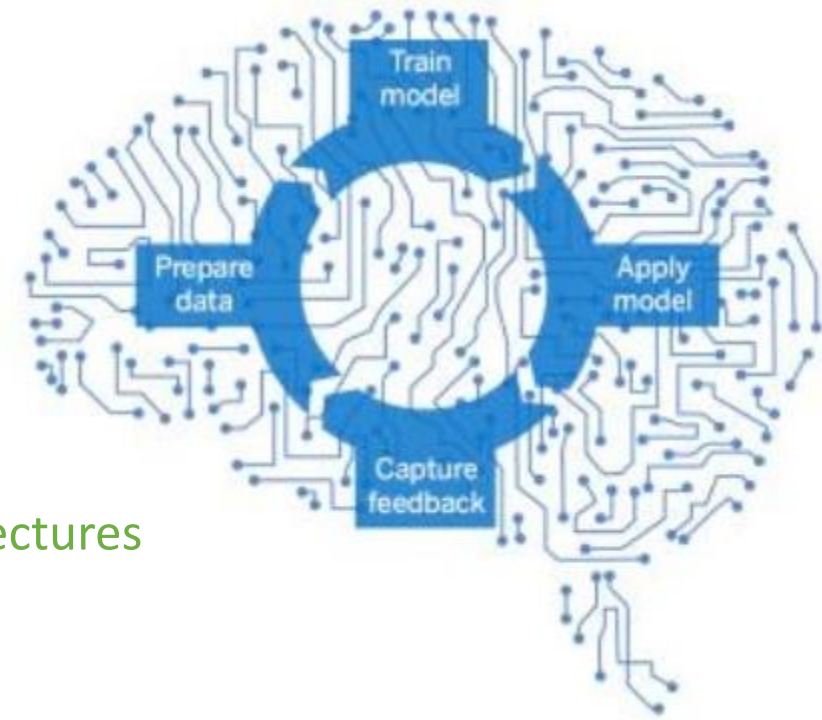


인하대학교



Contents

- 1 Review
- 2 CNN Architectures
- 3 Practice



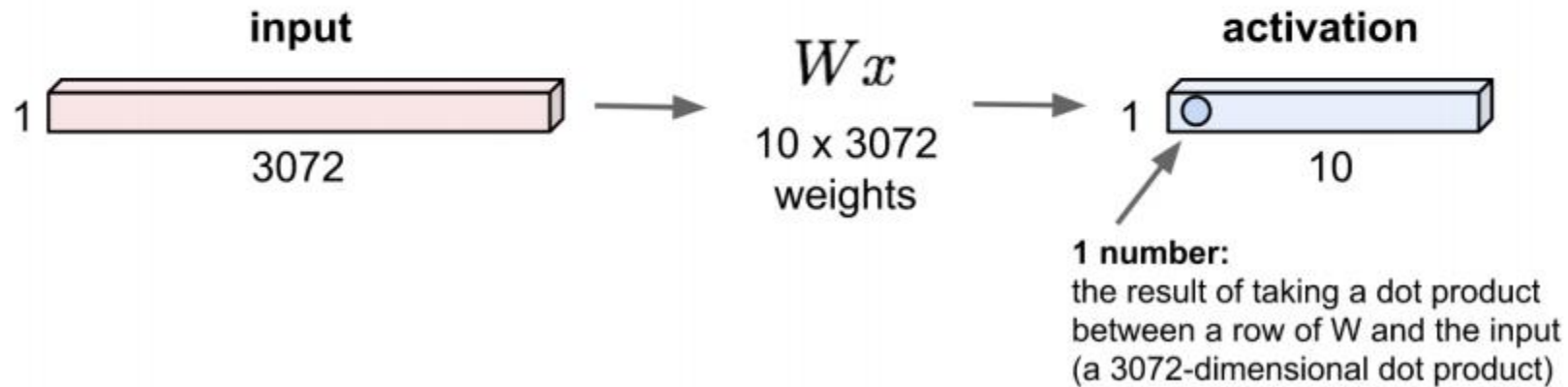
1

Review

CNN(Convolution Neural Network)

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



CNN(Convolution Neural Network)

그림 7-4 합성곱 연산의 계산 순서

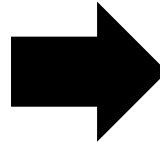
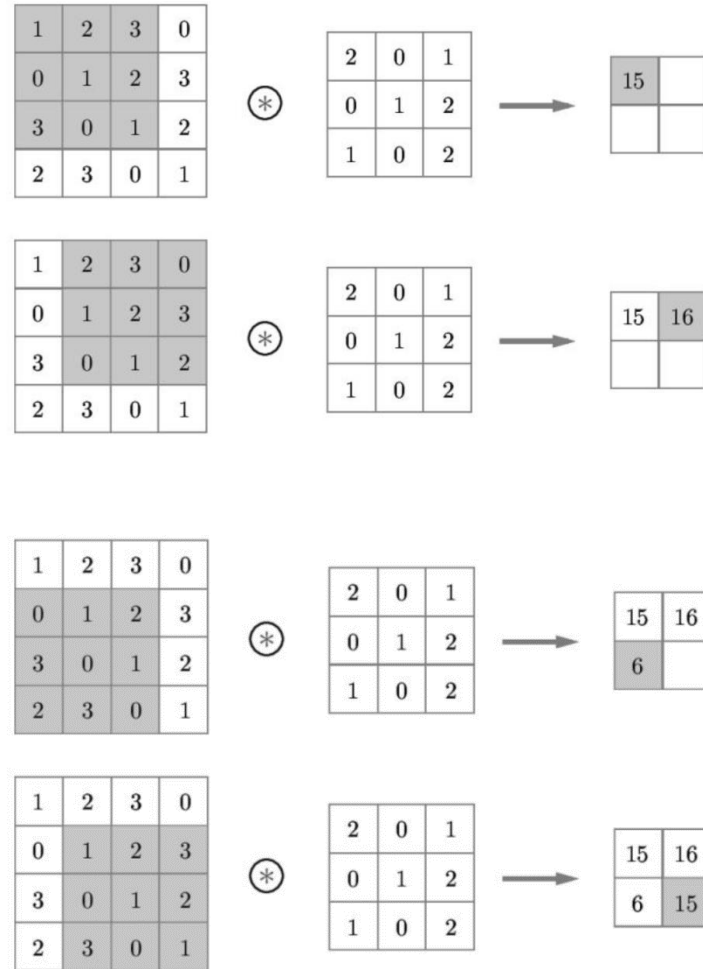
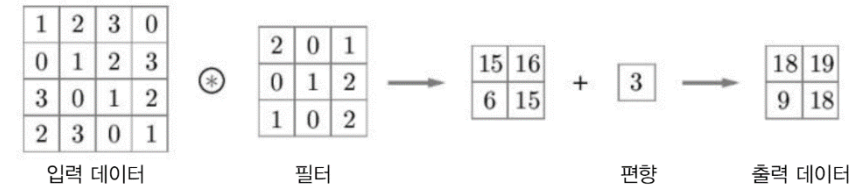


그림 7-5 합성곱 연산의 편향: 필터를 적용한 원소에 고정값(편향)을 더한다.



CNN(Convolution Neural Network)

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

7x7 output!

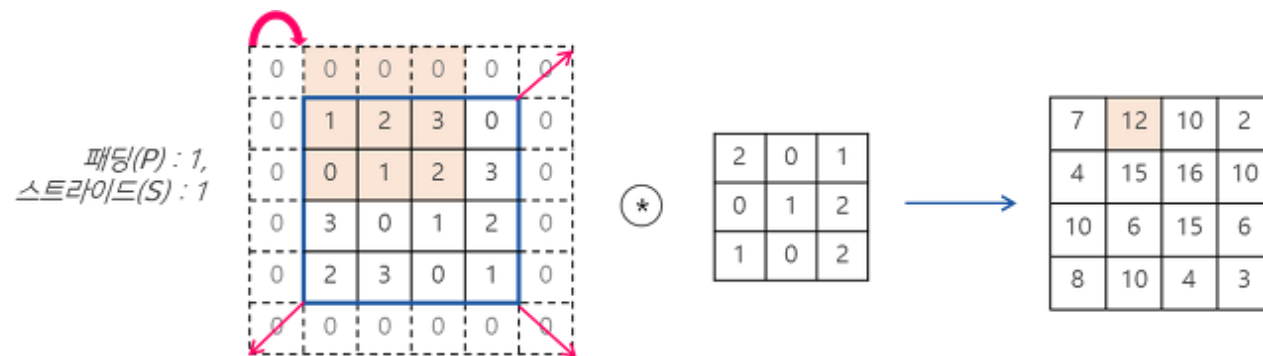
(recall:)

$$(N + 2P - F) / \text{stride} + 1$$

CNN(Convolution Neural Network)

$$(OH, OW) = \left(\frac{H + 2P - FH}{S} + 1, \frac{W + 2P - FW}{S} + 1 \right)$$

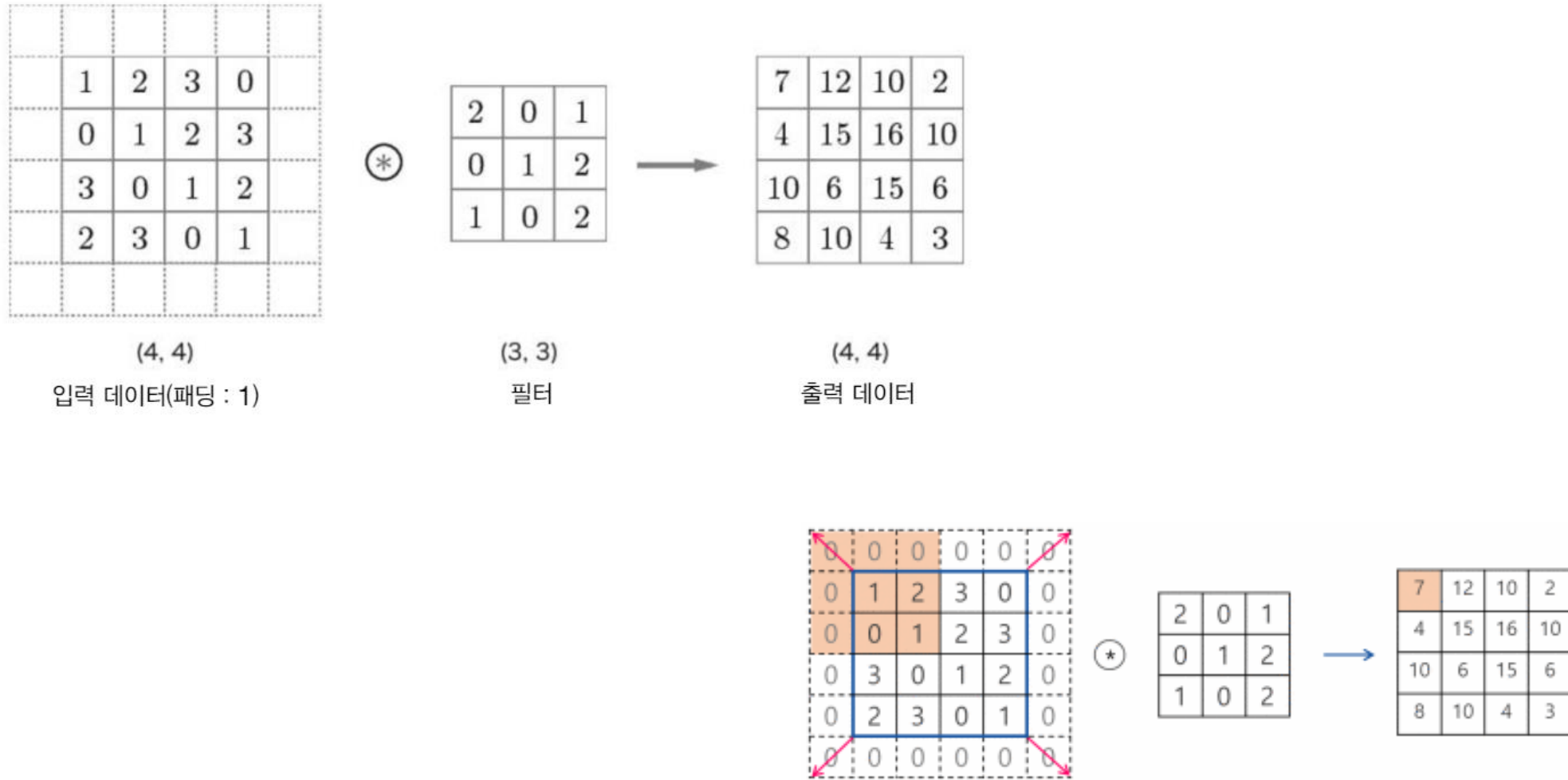
- (H, W) : 입력크기
- (FH, FW) : 필터크기
- (OH, OW) : 출력크기
- P : 패딩
- S : 스트라이드



$$(OH, OW) = \left(\frac{4 + 2 * 1 - 3}{1} + 1, \frac{4 + 2 * 1 - 3}{1} + 1 \right) = (4, 4)$$

CNN(Convolution Neural Network)

그림 7-6 합성곱 연산의 패딩 처리 : 입력 데이터 주위에 0을 채운다(패딩은 점선으로 표시했으며 그 안의 값 '0'은 생략했다).

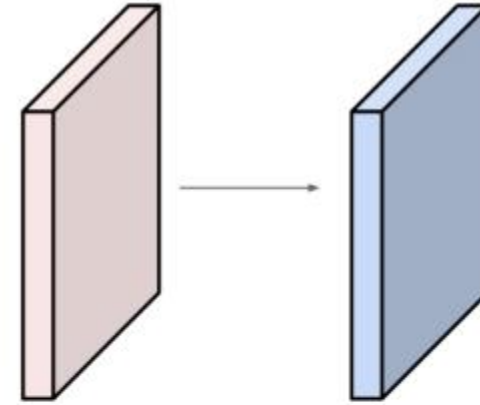


CNN(Convolution Neural Network)

Examples time:

Input volume: **32x32x3**
10 **5x5** filters with stride **1**, pad **2**

Output volume size:
 $(32+2*2-5)/1+1 = 32$ spatially, so
32x32x10

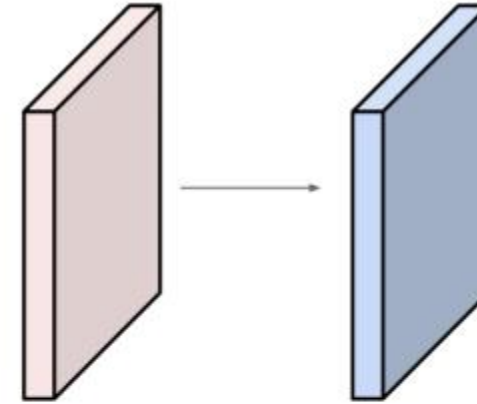


CNN(Convolution Neural Network)

Examples time:

Input volume: **32x32x3**

10 **5x5** filters with stride 1, pad 2



Number of parameters in this layer?

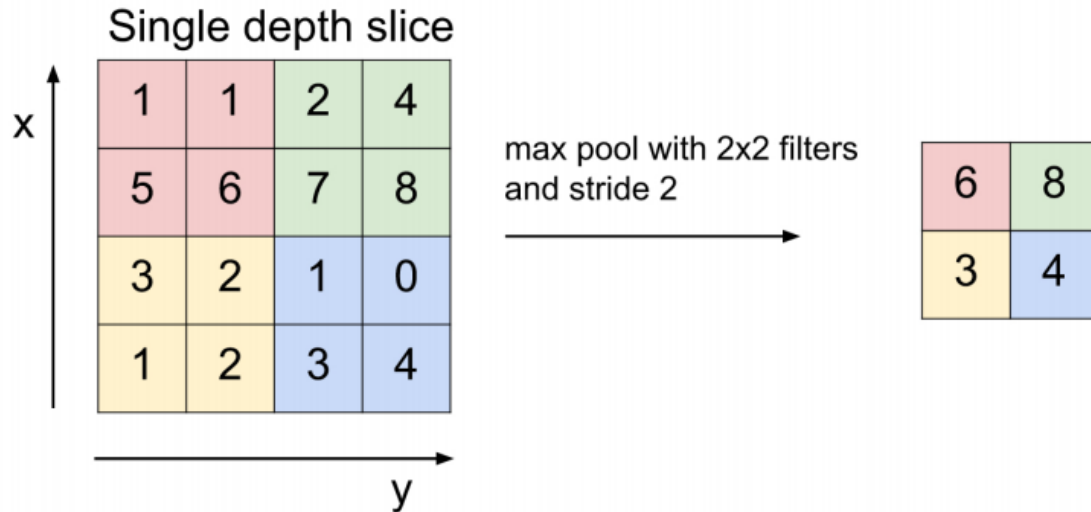
each filter has $5*5*3 + 1 = 76$ params

(+1 for bias)

$\Rightarrow 76*10 = 760$

CNN(Convolution Neural Network)

MAX POOLING



Pooling layer: summary

Let's assume input is $W_1 \times H_1 \times C$

Conv layer needs 2 hyperparameters:

- The spatial extent **F**
- The stride **S**

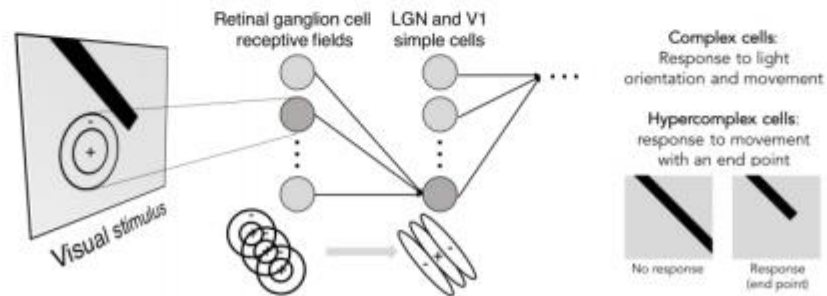
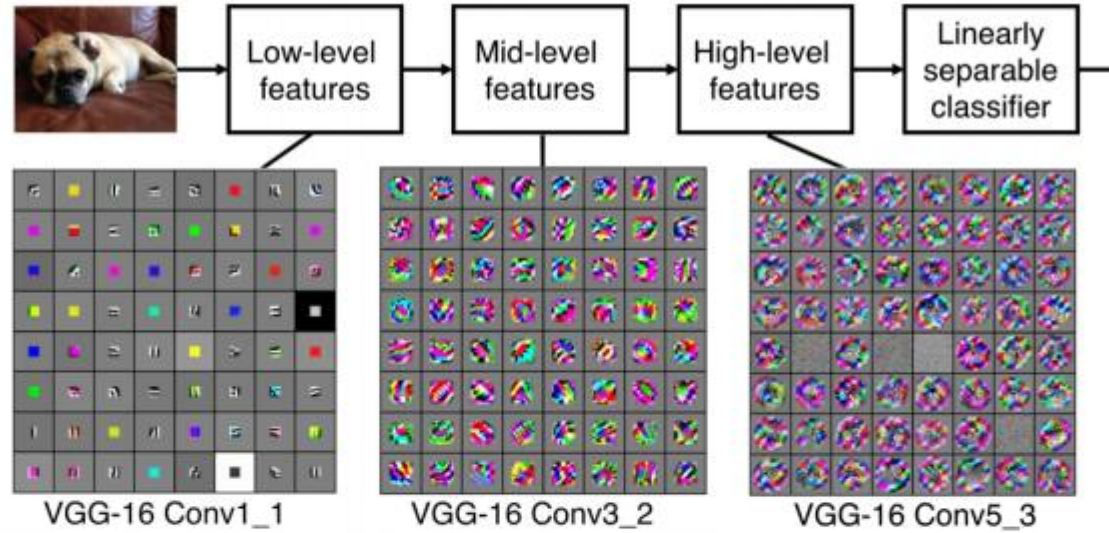
This will produce an output of $W_2 \times H_2 \times C$ where:

- $W_2 = (W_1 - F) / S + 1$
- $H_2 = (H_1 - F) / S + 1$

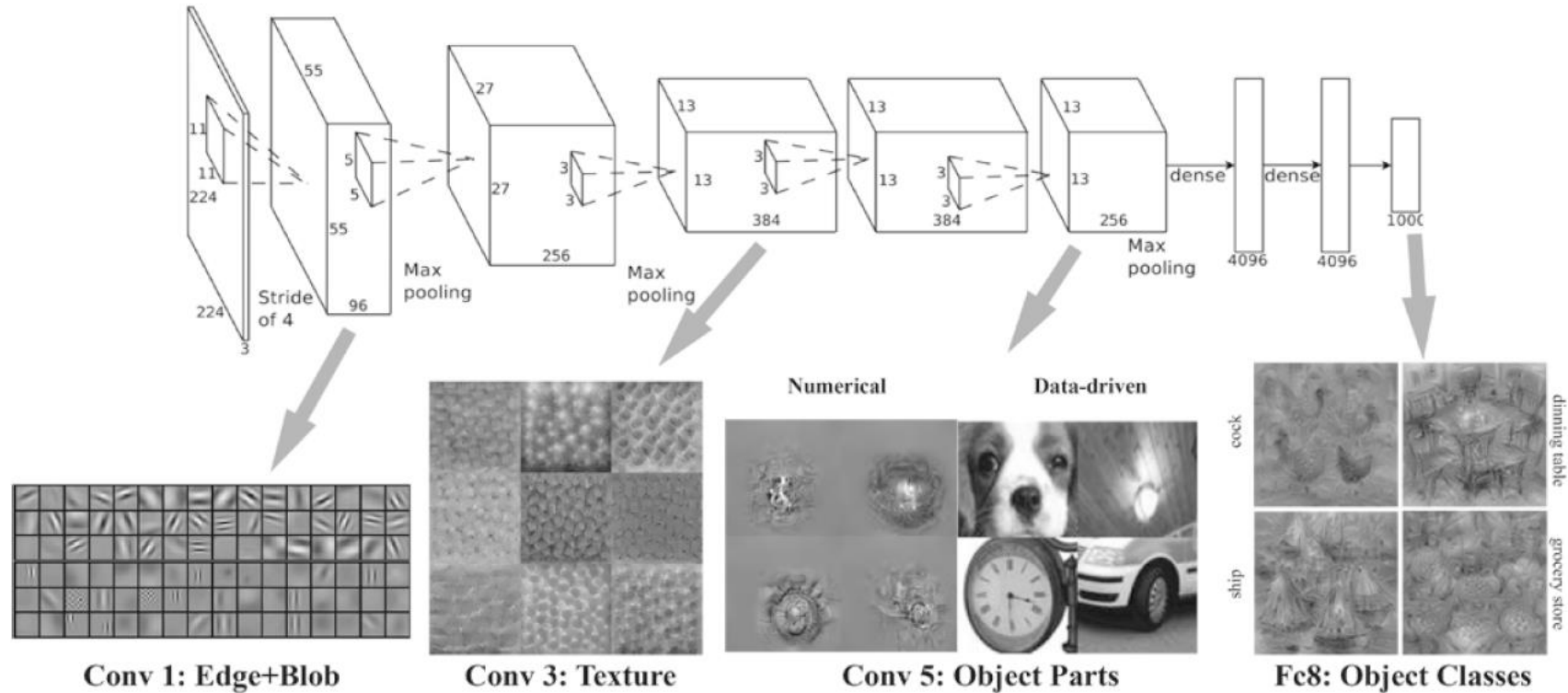
Number of parameters: 0

CNN(Convolution Neural Network)

Preview



CNN(Convolution Neural Network)

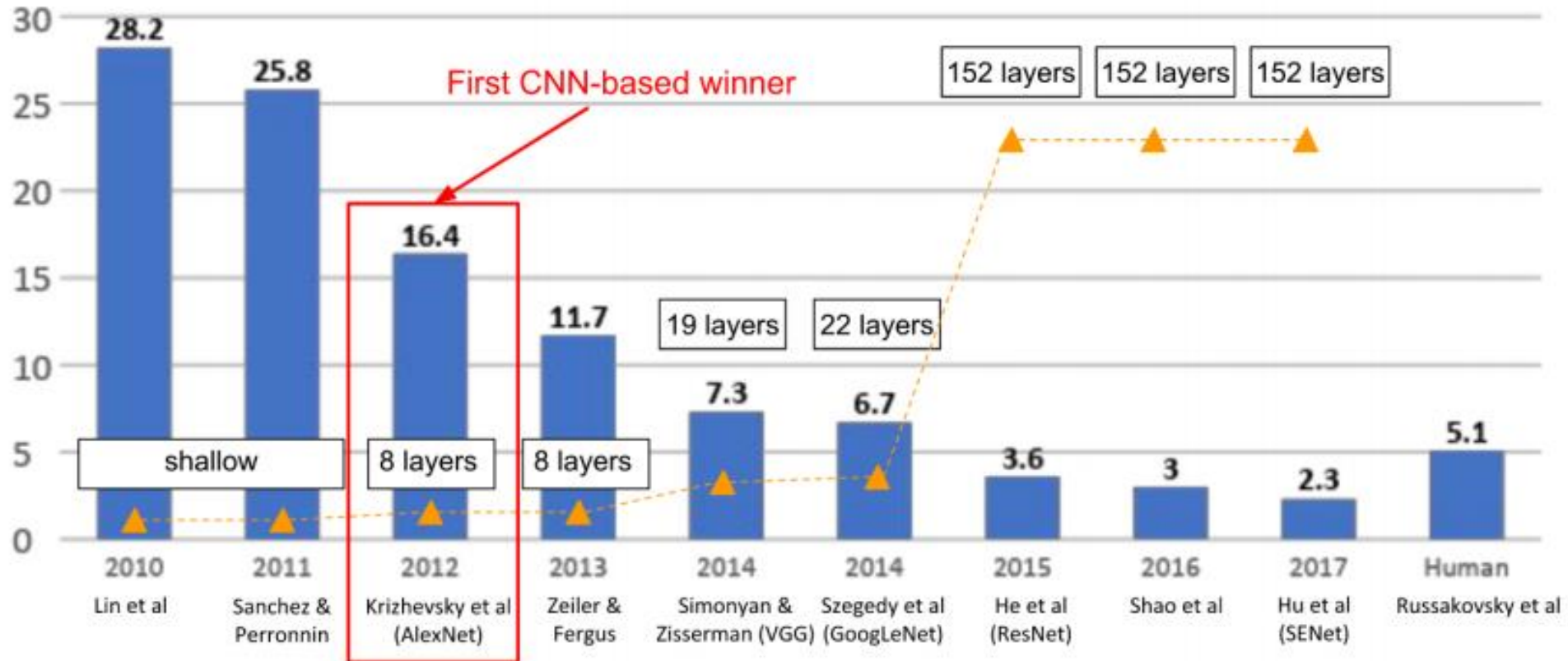


2

CNN Architectures

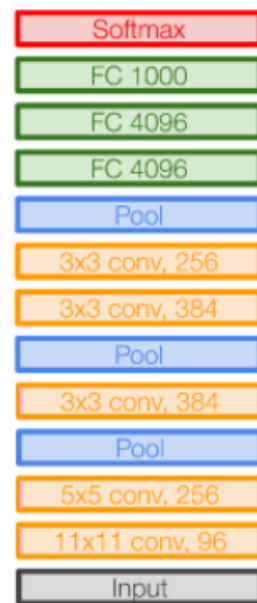
CNN Architectures

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



CNN Architectures

Case Study: AlexNet, VGGNet



AlexNet



VGG16

VGG19

CNN Architectures

INPUT: [224x224x3] memory: $224*224*3=150K$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*3)*64 = 1,728$

CONV3-64: [224x224x64] memory: $224*224*64=3.2M$ params: $(3*3*64)*64 = 36,864$

POOL2: [112x112x64] memory: $112*112*64=800K$ params: 0

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*64)*128 = 73,728$

CONV3-128: [112x112x128] memory: $112*112*128=1.6M$ params: $(3*3*128)*128 = 147,456$

POOL2: [56x56x128] memory: $56*56*128=400K$ params: 0

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*128)*256 = 294,912$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

CONV3-256: [56x56x256] memory: $56*56*256=800K$ params: $(3*3*256)*256 = 589,824$

POOL2: [28x28x256] memory: $28*28*256=200K$ params: 0

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*256)*512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28*28*512=400K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [14x14x512] memory: $14*14*512=100K$ params: 0

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14*14*512=100K$ params: $(3*3*512)*512 = 2,359,296$

POOL2: [7x7x512] memory: $7*7*512=25K$ params: 0

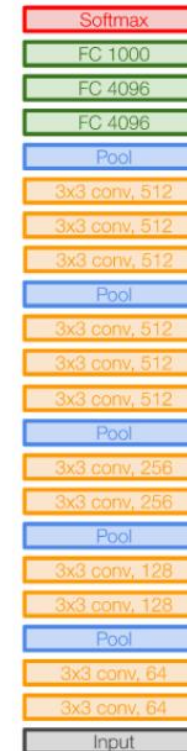
FC: [1x1x4096] memory: 4096 params: $7*7*512*4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096*4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096*1000 = 4,096,000$

TOTAL memory: $24M * 4 \text{ bytes} \sim 96MB$ / image (for a forward pass)

TOTAL params: 138M parameters



VGG16

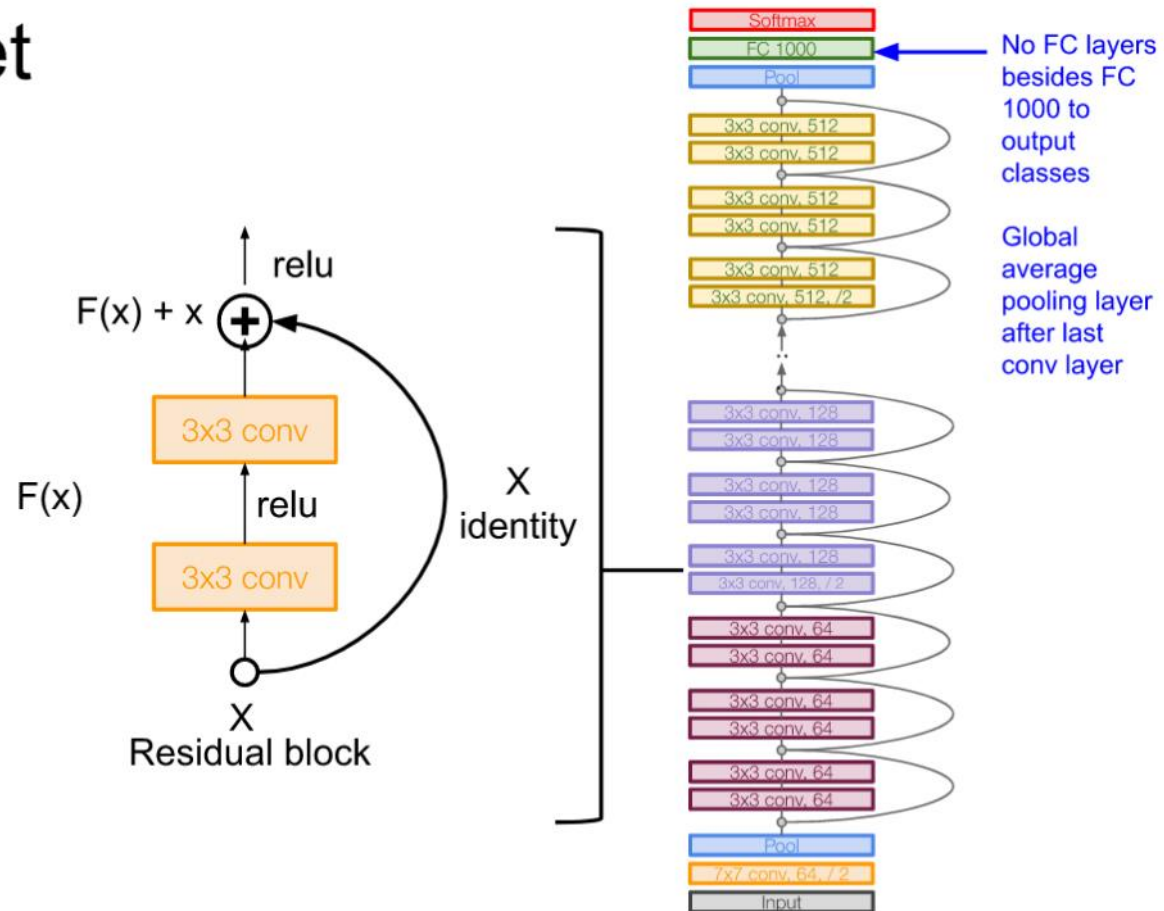
CNN Architectures

Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning (stem)
- No FC layers at the end (only FC 1000 to output classes)
- (In theory, you can train a ResNet with input image of variable sizes)



CNN Architectures

- <https://pytorch.org/docs/stable/torchvision/models.html>

TORCHVISION.MODELS

The models subpackage contains definitions of models for addressing different tasks, including: image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection and video classification.

```
import torchvision.models as models
resnet18 = models.resnet18()
alexnet = models.alexnet()
vgg16 = models.vgg16()
squeezenet = models.squeezenet1_0()
densenet = models.densenet161()
inception = models.inception_v3()
googlenet = models.googlenet()
shufflenet = models.shufflenet_v2_x1_0()
mobilenet = models.mobilenet_v2()
resnext50_32x4d = models.resnext50_32x4d()
wide_resnet50_2 = models.wide_resnet50_2()
mnasnet = models.mnasnet1_0()
```

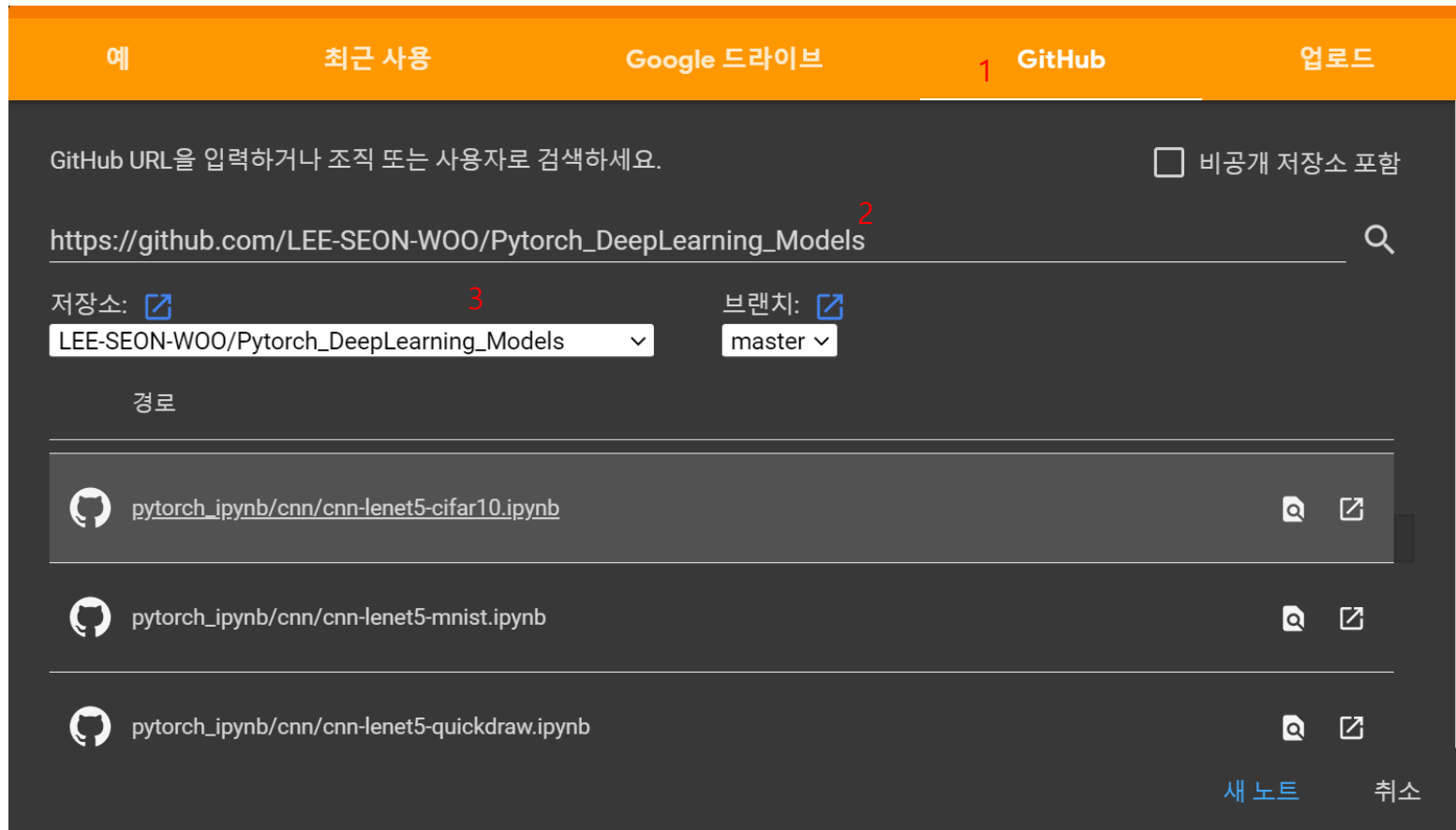
```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet = models.mobilenet_v2(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

3

Practice

CNN Tutorials

<https://colab.research.google.com/>



https://github.com/LEE-SEON-WOO/Pytorch_DeepLearning_Models

CNN Tutorials

<https://colab.research.google.com/>

The screenshot shows the Google Colaboratory web interface. At the top, there's a header with the Colab logo and the text 'Colaboratory에 오신 것을 환영합니다'. Below this is a navigation bar with options like '파일', '수정', '보기', '삽입', '런타임', '도구', '도움말', and a status message '변경사항을 저장할 수 없음'. The main area is divided into a left sidebar with icons for file management and a central workspace. The workspace contains a list of bullet points in Korean, including links to '이미지 분류기 재훈련', '텍스트 분류', '스타일 트랜스퍼', 'Multilingual Universal Sentence Encoder', and '동영상 보간 유형'. A modal dialog box titled '노트 설정' (Note Settings) is open in the center. It has a section '하드웨어 가속기' (Hardware Accelerator) with a dropdown menu set to 'GPU' and a help icon. Below this, there's a paragraph of text in Korean and a link '자세히 알아보기' (Learn more). At the bottom of the dialog, there's a checkbox labeled '이 노트를 저장할 때 코드 셀 출력 생략' (Omit code cell output when saving this notebook) and two buttons: '취소' (Cancel) and '저장' (Save).

Colaboratory에 오신 것을 환영합니다

파일 수정 보기 삽입 런타임 도구 도움말 변경사항을 저장할 수 없음

+ 코드 + 텍스트 드라이브로 복사 RAM 디스크 수정 가능

- [이미지 분류기 재훈련](#): 사전에 훈련된 이미지 분류기를 기반으로 꽃을 분류하기 위한 Keras 모델을 구축합니다.
- [텍스트 분류](#): IMDB 영화 리뷰를 *긍정적인 리뷰* 또는 *부정적인 리뷰*로 분류합니다.
- [스타일 트랜스퍼](#): 딥 러닝을 사용하여 이미지 간에 스타일을 전이시킵니다.
- [Multilingual Universal Sentence Encoder](#): 컴퓨터 세트의 질문에 답변합니다.
- [동영상 보간 유형](#): 동영상에서 첫 프레임과 다음 프레임을 보간합니다.

노트 설정

하드웨어 가속기
GPU ?

Colab를 최대한 활용하려면 필요하지 않은 경우 GPU를 사용하지 않는 것이 좋습니다.
[자세히 알아보기](#)

☐ 이 노트를 저장할 때 코드 셀 출력 생략

취소 저장

Book

주교재

밑바닥부터 시작하는 딥러닝 1, 사이토 고키 지음, 개앞맨시 옮김 .

(Deep Learning from Scratch의 번역서입니다 .)

부교재

Pytorch로 시작하는 딥러닝, 비슈누 수브라마니안 지음 김태완 옮김 .

(Deep Learning with PYTORCH 의 번역서입니다 .)

