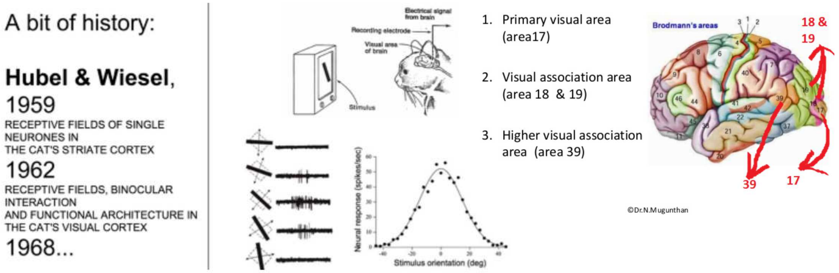
Convolutional Neural Network

PART-II

Convolutional Neural Network (CNN)



A bit of history:

Hubel & Wiesel,

1959

RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

1968...

1. Primary visual area (area17)
2. Visual association area (area 18 & 19)
3. Higher visual association area (area 39)

그림 출처: Fei-Fei Lee et al. 강의자료 (2016)

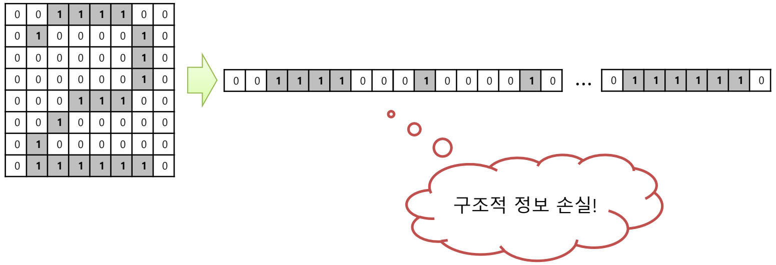
[실험]
고양이에게 시각자극을 제시하고 피질의 각기 다른 층에 있는 개별 신경세포의 활동을 기록

[목적]
각각의 신경세포가 무엇을 탐지하도록 전문화되어 있는 것인지를 찾아내려는 것

[발견]
고양이의 피질이 **위계적 세부 특징 탐색 회로망**으로 작동
피질의 하위층에 있는 신경세포는 단순한 세부 특징을 탐지하는 반면, 계속해서 상위층으로 올라갈수록 신경세포들은 보다 복잡한 세부 특징을 탐지함

Data Representation Problem

- Feed-Forward Neural Network (FFNN)
  - n차원 벡터를 1차원으로 변환하여 입력



0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0
0	0	0	1	1	1	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	1	1	1	0

0 0 1 1 1 1 0 0 0 1 0 0 0 0 1 0 … 0 1 1 1 1 1 1 0

구조적 정보 손실!

Data Abstraction

center one

10	2	8
2	15	3
5	1	5

→ 15

average

10	2	8
2	15	3
5	1	5

→ 5.6

median

10	2	8
2	15	3
5	1	5

→ 5

그림 출처: 충남대 정상근 교수님 강의자료

Data Abstraction

Weighted Sum

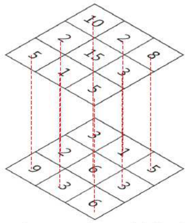
10	2	8
2	15	3
5	1	5

Value

3	1	5
2	6	3
9	3	6

Weight

→ 28.1



Element-wise multiplication

Weighted Average

10	2	8
2	15	3
5	1	5

Value

3/9	1/9	5/9
2/9	6/9	3/9
9/9	3/9	6/9

Weight

→ 6.65

그림 출처: 충남대 정상근 교수님 강의자료

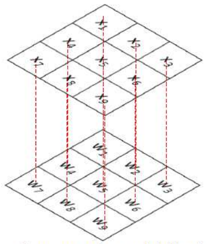
Data Abstraction

Weighted Sum

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

→ ~



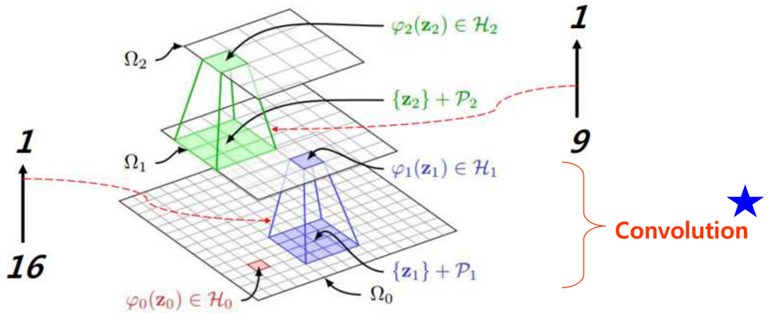
Element-wise multiplication

$$v = x_1 * w_1 + x_2 * w_2 + \cdots + x_9 * w_9$$

➡ [1 x 9] matrix x [9x1] matrix = [1x1] matrix $\sum_i^9 x_i * w_i$

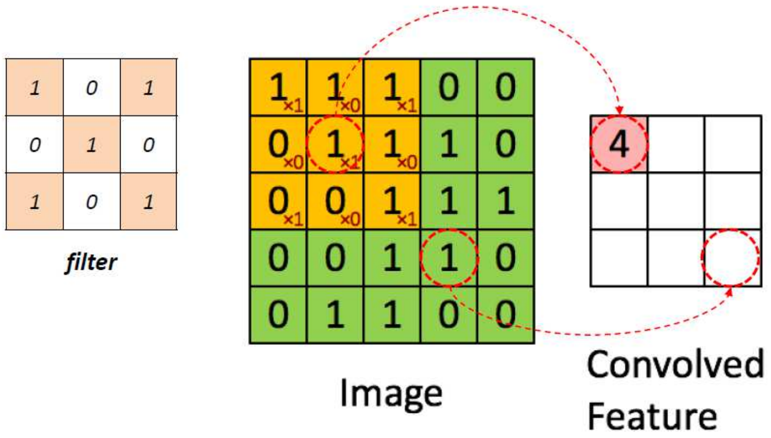
그림 출처: 충남대 정상근 교수님 강의자료

Architecture of CNN


$\varphi_2(\mathbf{z}_2) \in \mathcal{H}_2$
Ω_2
$\{\mathbf{z}_2\} + \mathcal{P}_2$
Ω_1
$\varphi_1(\mathbf{z}_1) \in \mathcal{H}_1$
$\{\mathbf{z}_1\} + \mathcal{P}_1$
Ω_0
$\varphi_0(\mathbf{z}_0) \in \mathcal{H}_0$
1
16
1
9
Convolution

- Multi-layer feed-forward ANN
- Combinations of *convolutional* and fully connected layers
- Convolutional layers with *local* connectivity
- *Shared* weights across spatial positions
- Local or global pooling layers

What is Convolution?



1	0	1
0	1	0
1	0	1

filter

1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

그림 출처: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Convolution Step

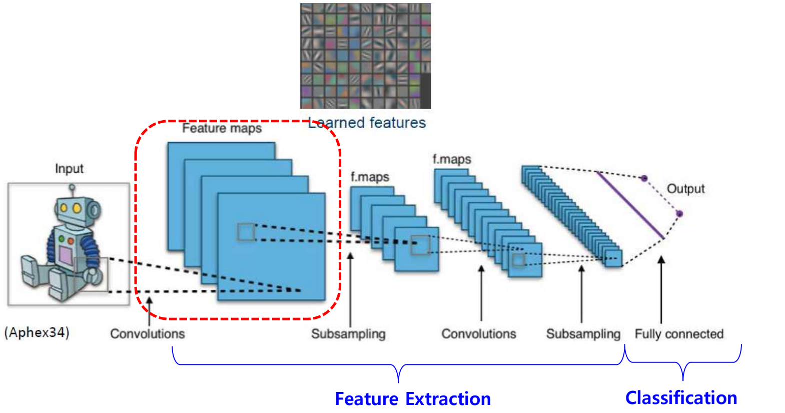


그림 출처: Nelson, Daniel. "What Are Convolutional Neural Networks?" Unite.AI, May 24, 2020. https://www.unite.ai/what-are-convolutional-neural-networks/.

Channel

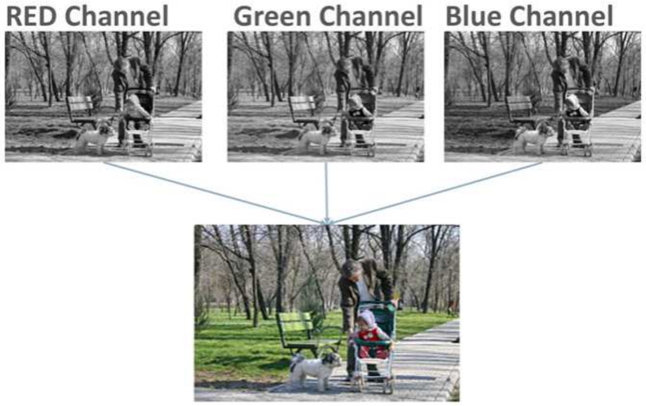
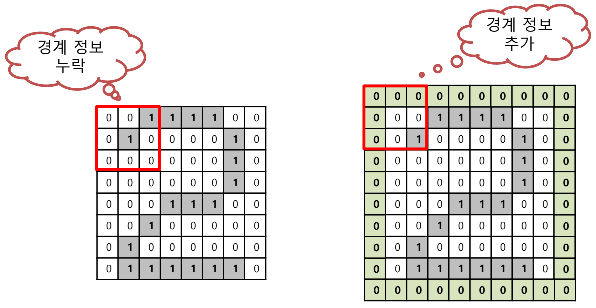


그림 출처: https://en.Wikipedia.org/wiki/Channel_(digital_image)

Padding

- 패딩
  - 경계(edge, boundary)에 대한 정보를 누락하지 않기 위해서 벡터 외부에 특정 정보를 추가하는 것



0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0
0	0	0	1	1	1	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	1	1	1	0

패딩 정보 추가 전

0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0

패딩 정보 추가 후

Filter (Kernel)

Stride = 1
Filter Size= 2*2

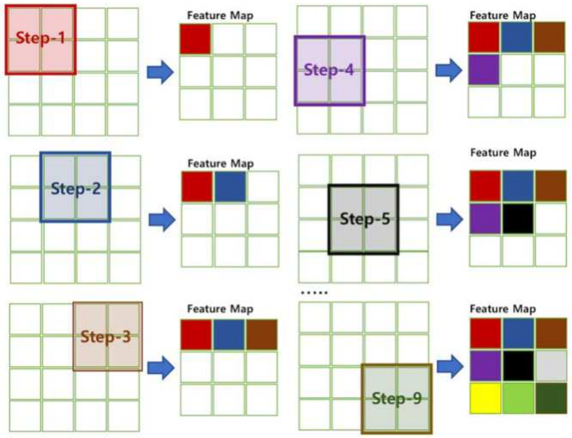
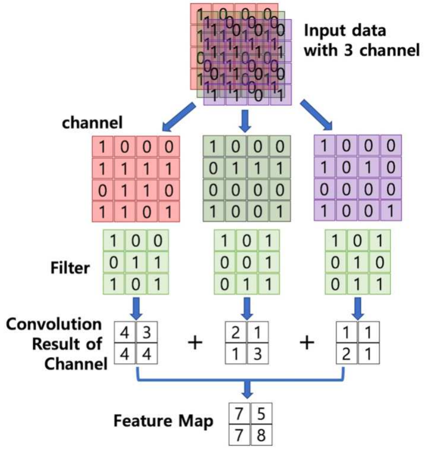


그림 출처: TAEWAN.KIM 블로그(http://taewan.kim/post/cnn/)

Feature Map



channel

1	0	0	0
1	1	1	1
0	1	1	0
1	1	0	1

1	0	0	0
0	1	1	1
0	0	0	0
1	0	0	1

1	0	0	0
1	0	1	0
0	0	0	0
1	0	0	1

Filter

1	0	0
0	1	1
1	0	1

1	0	1
0	0	1
0	1	1

1	0	1
0	1	0
0	1	1

Convolution Result of Channel

4	3
4	4

+

2	1
1	3

+

1	1
2	1

Feature Map

7	5
7	8

그림 출처: TAEWAN.KIM 블로그(http://taewan.kim/post/cnn/)

Subsampling (Pooling) Step

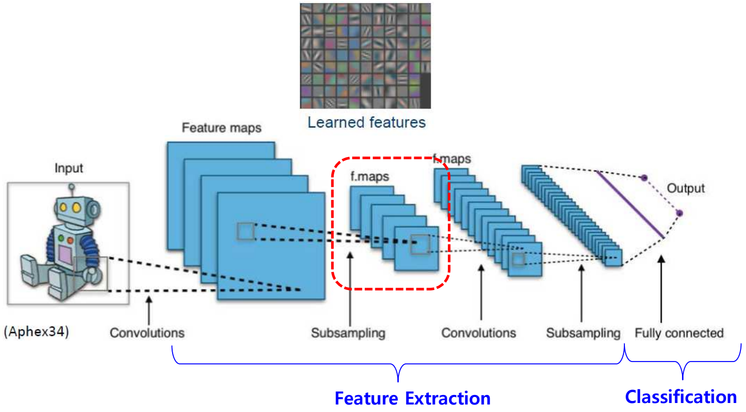
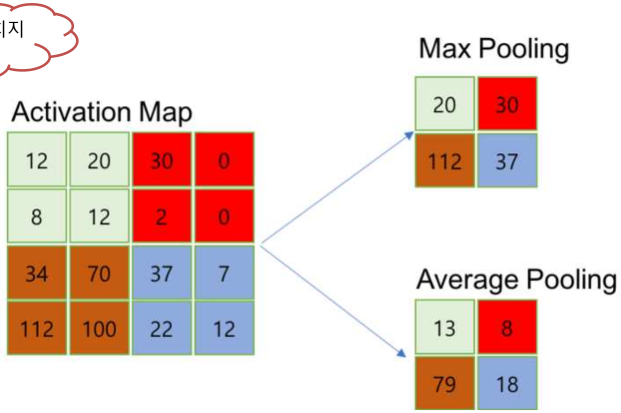


그림 출처: Nelson, Daniel. "What Are Convolutional Neural Networks?" Unite.AI, May 24, 2020. https://www.unite.ai/what-are-convolutional-neural-networks/.

Subsampling (Pooling)

필터가 겹치지 않음



Activation Map

12	20	30	0
8	12	2	0
34	70	37	7
112	100	22	12

Max Pooling

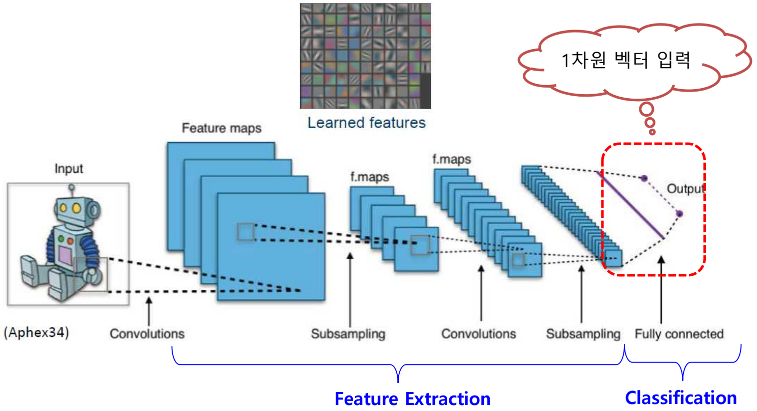
20	30
112	37

Average Pooling

13	8
79	18

참고: TAEWAN.KIM 블로그(http://taewan.kim/post/cnn/)

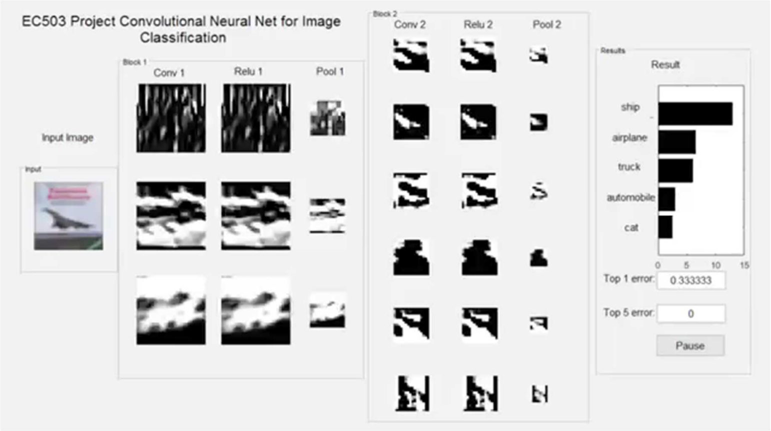
FNN Step



1차원 벡터 입력

그림 출처: Nelson, Daniel. "What Are Convolutional Neural Networks?" Unite.AI, May 24, 2020. https://www.unite.ai/what-are-convolutional-neural-networks/.

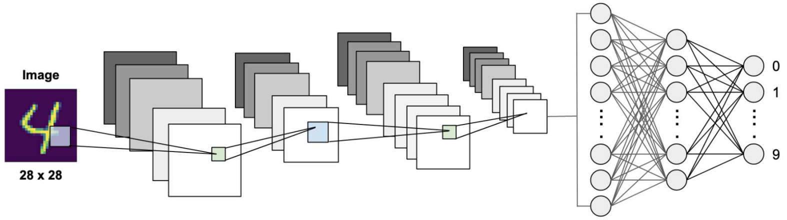
CNN 시연 영상



영상 출처: https://www.youtube.com/watch?v=bEzS-kFSi5k

MNIST by CNN

실습 코드 다운로드:
https://github.com
/KUNLP/Lecture



Convolution
Channel: 32
Filter: (3,3)
Stride: (1,1)
Padding: (1,1)
+
ReLU

Max Pooling
Filter: (2,2)
Stride: (2,2)

Convolution
Channel: 64
Filter: (3,3)
Stride: (1,1)
Padding: (1,1)
+
ReLU

Max Pooling
Filter: (2,2)
Stride: (2,2)

Fully-Connected NN
Input: 7*7*64
Output: 10

Shape

(28,28,1) → (28,28,32) → (14,14,32) → (14,14,64) → (7,7,64) → (10)

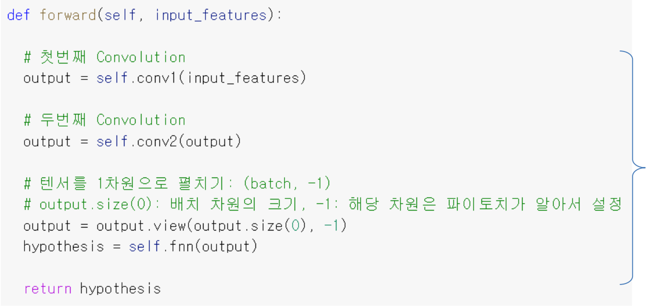
MNIST by CNN

```
class MNIST_CNN(nn.Module):

  def __init__(self, config):

    super(MNIST_CNN, self).__init__()

    # 첫번째 층 설계: Convolutional NN
    # (batch, 28, 28, 1) -> (batch, 28, 28, 32) -> (batch, 14, 14, 32)
    self.conv1 = nn.Sequential()
    self.conv1.add_module("conv1", nn.Conv2d(1,32,kernel_size=(3,3), stride=(1,1), padding=(1,1)))
    self.conv1.add_module("relu1", nn.ReLU())
    self.conv1.add_module("maxpool1", nn.MaxPool2d(kernel_size=(2,2), stride=(2,2)))

    # 두번째 층 설계: Convolutional NN
    # (batch, 14, 14, 32) -> (batch, 14, 14, 64) -> (batch, 7, 7, 64)
                              ?

    # 세번째 층 설계: Fully-Connected NN
    # (batch, 7, 7, 64) -> (batch, 10)
                              ?
    # FNN 가중치 초기화
    nn.init.xavier_uniform_(self.fnn.weight)
```

다양한 초기화 함수 존재

$$W \sim U\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, +\sqrt{\frac{6}{n_{in}+n_{out}}}\right)$$

MNIST by CNN

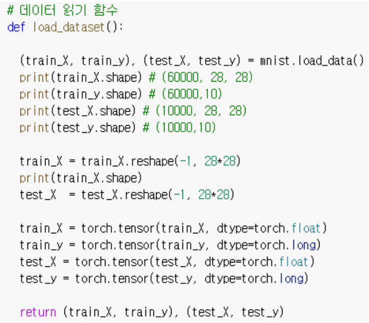


```python
def forward(self, input_features):

  # 첫번째 Convolution
  output = self.conv1(input_features)

  # 두번째 Convolution
  output = self.conv2(output)

  # 텐서를 1차원으로 펼치기: (batch, -1)
  # output.size(0): 배치 차원의 크기, -1: 해당 차원은 파이토치가 알아서 설정
  output = output.view(output.size(0), -1)
  hypothesis = self.fnn(output)

  return hypothesis
```

Hypothesis 만들기

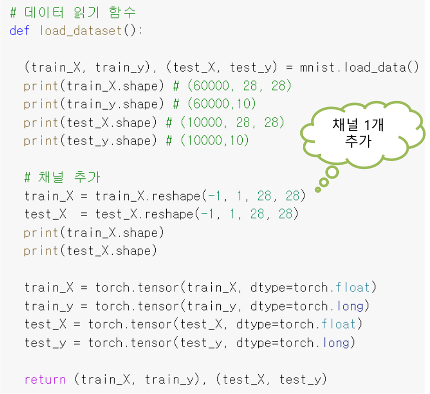
MNIST by CNN

MLP 1차원 입력 (batch, 28*28)



```python
# 데이터 읽기 함수
def load_dataset():

  (train_X, train_y), (test_X, test_y) = mnist.load_data()
  print(train_X.shape) # (60000, 28, 28)
  print(train_y.shape) # (60000,10)
  print(test_X.shape) # (10000, 28, 28)
  print(test_y.shape) # (10000,10)

  train_X = train_X.reshape(-1, 28*28)
  print(train_X.shape)
  test_X  = test_X.reshape(-1, 28*28)

  train_X = torch.tensor(train_X, dtype=torch.float)
  train_y = torch.tensor(train_y, dtype=torch.long)
  test_X = torch.tensor(test_X, dtype=torch.float)
  test_y = torch.tensor(test_y, dtype=torch.long)

  return (train_X, train_y), (test_X, test_y)
```

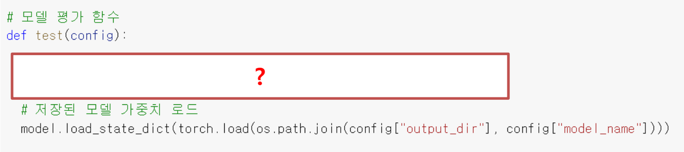
CNN 3차원 입력 (batch, 1, 28, 28)



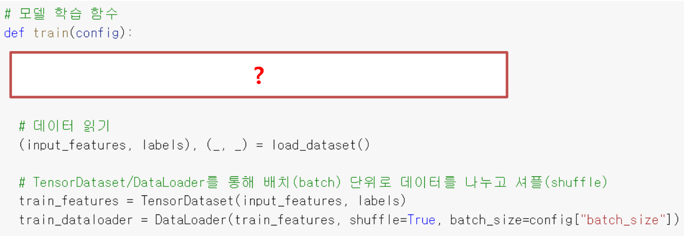
```python
# 데이터 읽기 함수
def load_dataset():

  (train_X, train_y), (test_X, test_y) = mnist.load_data()
  print(train_X.shape) # (60000, 28, 28)
  print(train_y.shape) # (60000,10)
  print(test_X.shape) # (10000, 28, 28)
  print(test_y.shape) # (10000,10)

  # 채널 추가
  train_X = train_X.reshape(-1, 1, 28, 28)
  test_X  = test_X.reshape(-1, 1, 28, 28)
  print(train_X.shape)
  print(test_X.shape)

  train_X = torch.tensor(train_X, dtype=torch.float)
  train_y = torch.tensor(train_y, dtype=torch.long)
  test_X = torch.tensor(test_X, dtype=torch.float)
  test_y = torch.tensor(test_y, dtype=torch.long)

  return (train_X, train_y), (test_X, test_y)
```

채널 1개
추가

MNIST by CNN

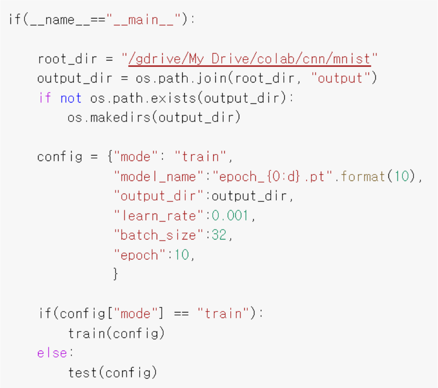


```python
# 모델 평가 함수
def test(config):

                              ?

  # 저장된 모델 가중치 로드
  model.load_state_dict(torch.load(os.path.join(config["output_dir"], config["model_name"])))
```



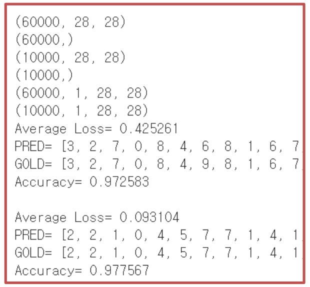
```python
# 모델 학습 함수
def train(config):

                              ?

  # 데이터 읽기
  (input_features, labels), (_, _) = load_dataset()

  # TensorDataset/DataLoader를 통해 배치(batch) 단위로 데이터를 나누고 셔플(shuffle)
  train_features = TensorDataset(input_features, labels)
  train_dataloader = DataLoader(train_features, shuffle=True, batch_size=config["batch_size"])
```

MNIST by CNN



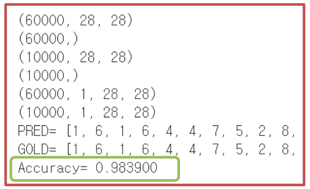
```python
if(__name__=="__main__"):

    root_dir = "/gdrive/My Drive/colab/cnn/mnist"
    output_dir = os.path.join(root_dir, "output")
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    config = {"mode": "train",
              "model_name":"epoch_{0:d}.pt".format(10),
              "output_dir":output_dir,
              "learn_rate":0.001,
              "batch_size":32,
              "epoch":10,
              }

    if(config["mode"] == "train"):
        train(config)
    else:
        test(config)
```

학습



```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
(60000, 1, 28, 28)
(10000, 1, 28, 28)
Average Loss= 0.425261
PRED= [3, 2, 7, 0, 8, 4, 6, 8, 1, 6, 7
GOLD= [3, 2, 7, 0, 8, 4, 9, 8, 1, 6, 7
Accuracy= 0.972583

Average Loss= 0.093104
PRED= [2, 2, 1, 0, 4, 5, 7, 7, 1, 4, 1
GOLD= [2, 2, 1, 0, 4, 5, 7, 7, 1, 4, 1
Accuracy= 0.977567
```

평가



```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
(60000, 1, 28, 28)
(10000, 1, 28, 28)
PRED= [1, 6, 1, 6, 4, 4, 7, 5, 2, 8,
GOLD= [1, 6, 1, 6, 4, 4, 7, 5, 2, 8,
Accuracy= 0.983900
```

질의응답

Q&A

Homepage: http://nlp.konkuk.ac.kr
E-mail: nlpdrkim@konkuk.ac.kr