PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Machine Learning Analysis
Feature Extraction

GyeongYeong, Kim

ML/DL Lab 2nd Day

PIAI Research Department

가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Keras Model Analysis
  - Callback
  - History Confirmation
  - Wrong Sample Confirmation
  - Confusion Matrix
- Feature Extraction
  - CNN / Pooling Layer
  - Batch Normalization / DropOut
- Pytorch Feature Extraction
  - CNN / Pooling Layer
  - Batch Normalization / DropOut
  - Best Model Saving
  - History Confirmation
  - Confusion Matrix
  - Wrong Sample Confirmation
  - Early Stopping

Keras Callbacks

PIAI Research Department

- Keras Callbacks : 학습 이슈을 분석하거나 조정할 때 쓰이는 학습 보조기능

ModelCheckpoint : 학습 도중 모델 저장

TerminateOnNaN : NaN 손실이 발생했을 때 학습을 종료

RemoteMonitor : 이벤트를 서버에 스트림

LearningRateScheduler : 학습속도에 대해 스케쥴을 짬

EarlyStopping : 학습 양상을 보고 학습을 조기에 종료(Overfitting 방지)

Tensorboard : 텐서보드로 그래프를 확인할 수 있도록 학습 양상을 기록(동적 그래프나 활성화 히스토그램 시각화)

ReduceLROnPlateau : 측정 항목이 향상되지 않는 경우 학습속도를 줄임

구체적 내용 : https://keras.io/api/callbacks/

Model Checkpoint

PIAI Research Department

Model Checkpoint : 케라스의 모델(혹은 weight)를 규칙에 따라 저장하는 콜백

- file_path : 모델 저장 경로
- monitor(metric) /save_best_only(True/False) : 모니터할 metric 설정 및 이에 따른 높은 성능의 모델을 저장
- mode (auto/min/max) : 모니터할 metric의 최대나 최소 중 선택(acc는 max, loss는 min)
- Initial_value_threshold : 모니터할 최소/최대 기준값
- save_freq(int) : 저장할 주기(epoch 기준으로 설정)
- verbose(0/1/2) : 콜백이 적용될 시 표출할 모드
- (save_weight_only : full model의 저장 대신 weight만 저장)

```
checkpointer = ModelCheckpoint(filepath=model_name_path,
                               monitor='val_accuracy',
                               verbose=0,
                               save_best_only=True)
```

Early Stopping

PIAI Research Department

Keras Callbacks : 학습 이슈을 분석하거나 조정할 때 쓰이는 학습 보조기능

- monitor(metric) : 모니터할 기준
- patience(int) : 학습 개선이 없어 학습을 멈출 최소 epoch 기준
- verbose(0/1/2) : 콜백이 적용될 시 표출할 모드
- mode(auto/min/max) : monitor metric을 결정하는 기준
- baseline : 모니터하는 데에 있어서의 베이스라인 값
- restore_best_weights(True/False) : 가장 좋은 성능으로 저장할지 안할지 설정

```
earlystopper = EarlyStopping(monitor='val_accuracy',
                             patience=5,
                             verbose=0,
                             mode='auto')
```

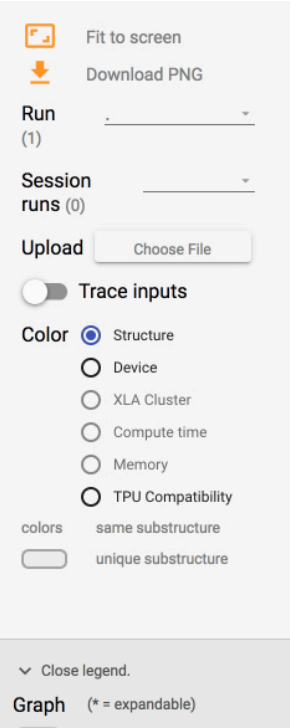
TensorBoard

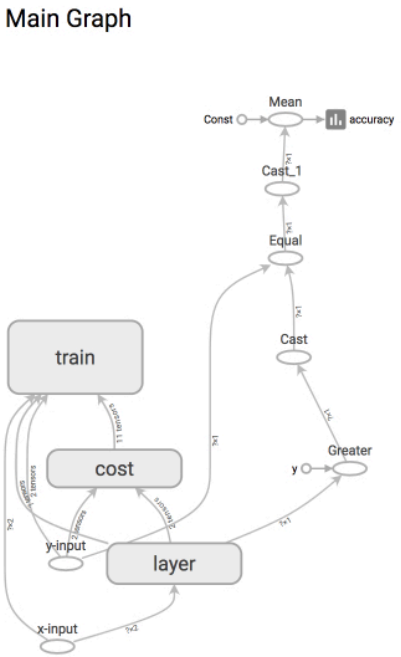
PIAI Research Department

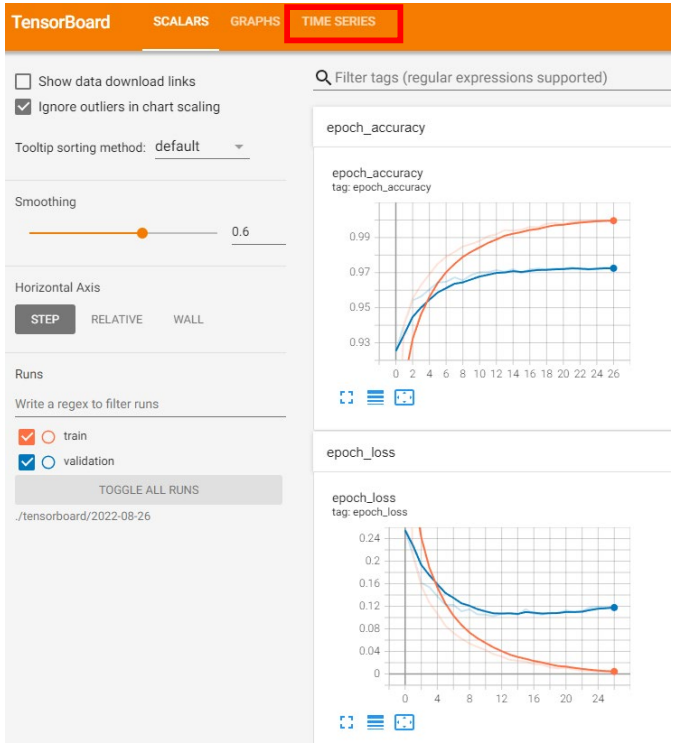
- Keras Callbacks 중 하나 : 시각화 도구로 활용됨

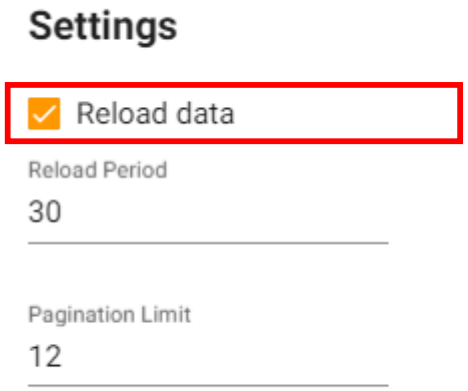
```
tb_saver = TensorBoard(log_dir=tensorboard_path,
                       write_graph=True)
```

- 실시간 업데이트 확인 가능 : Time Series 탭 내 Settings로 확인



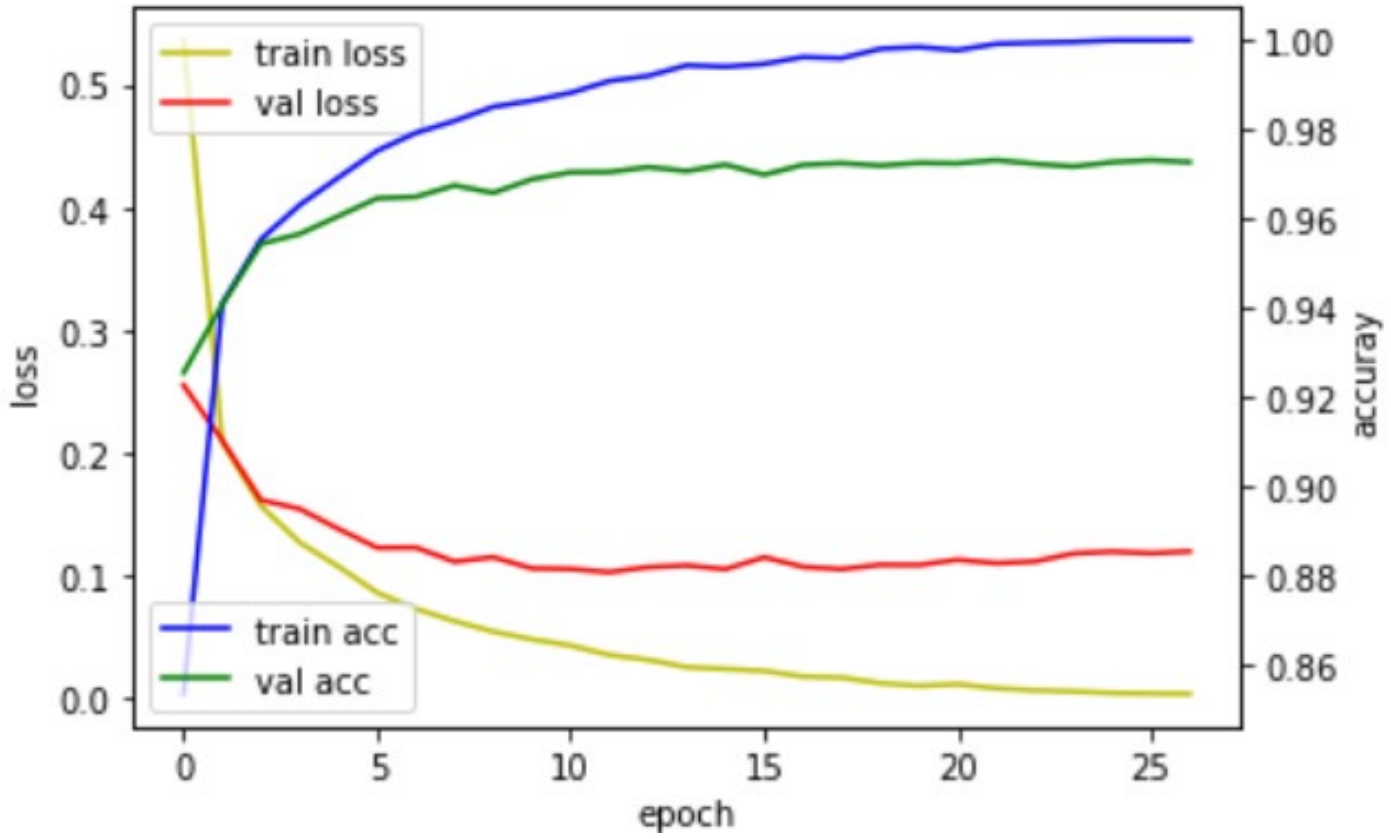






History 확인

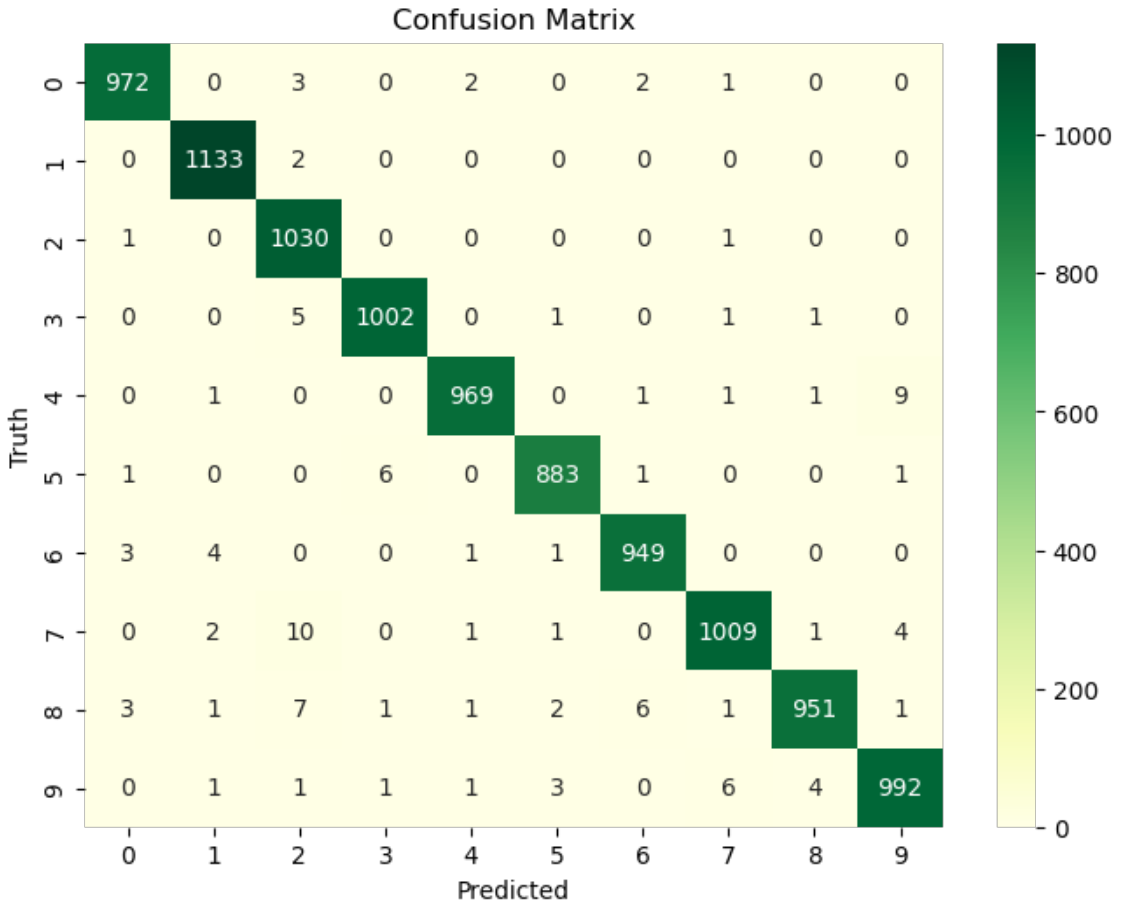
- Epoch 마다의 학습 추이를 확인
- Loss나 Accuracy의 추이를 통해 추가 학습의 필요 여부, Underfitting/ Overfitting 확인



Confusion Matrix

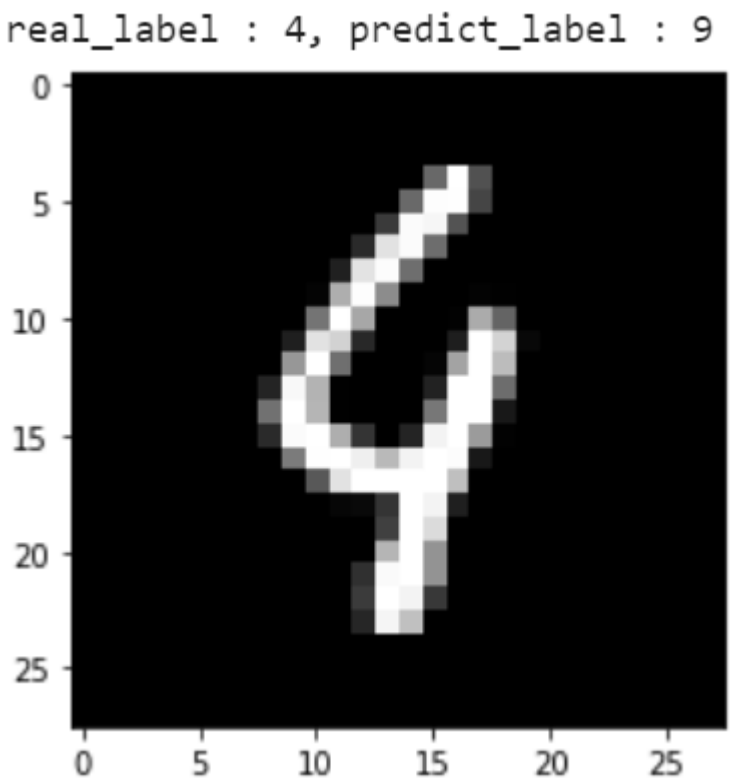
PIAI Research Department

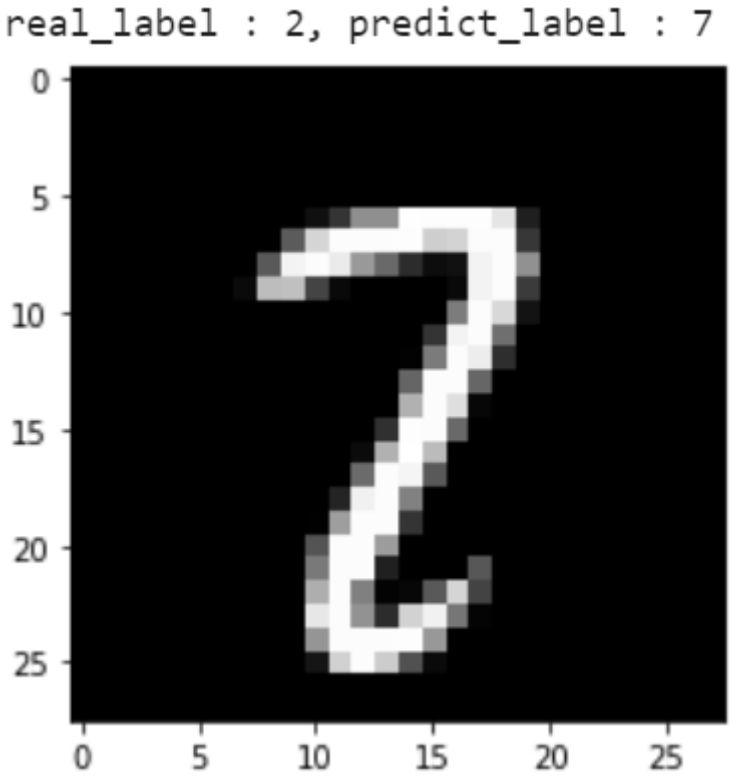
- 모델결과에 대한 분석이 용이함
- 클래스가 많으면 하기에 어려울 수 있음
- 모델의 구체적인 문제 사례를 확인하기엔 한계가 있음 → 특정 사례에 한해 직접 확인!



Wrong Sample 확인

- Confusion Matrix : 거시적 분석 / Wrong Sample : 미시적 분석
- 학습을 통한 개선이 가능한지의 여부를 확인 가능 (항상 네트워크 학습으로 해결할 수 있는 것은 아니다..!)





가치창출대학
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

Feature Extraction

Image classification

PIAI Research Department

```
def classify_image(image):
    # Some magic here?
    return class_label
```

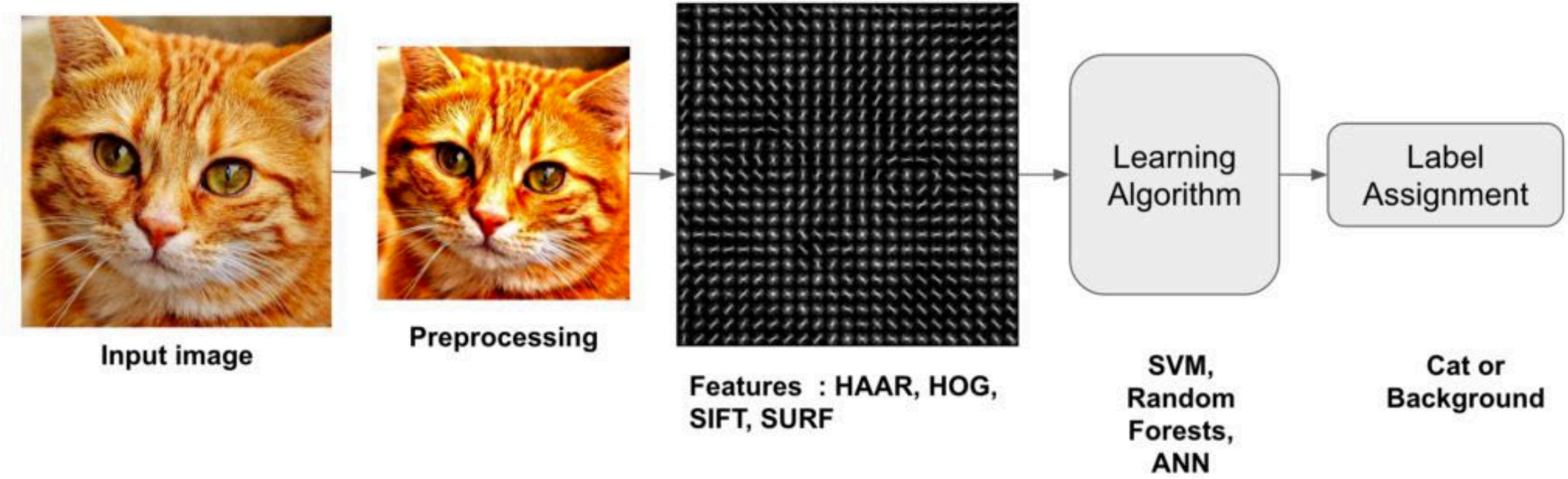
Cat

어떻게 하면 컴퓨터가 '고양이'를 인식하도록 코딩할 수 있을까?

Traditional Image classification

PIAI Research Department

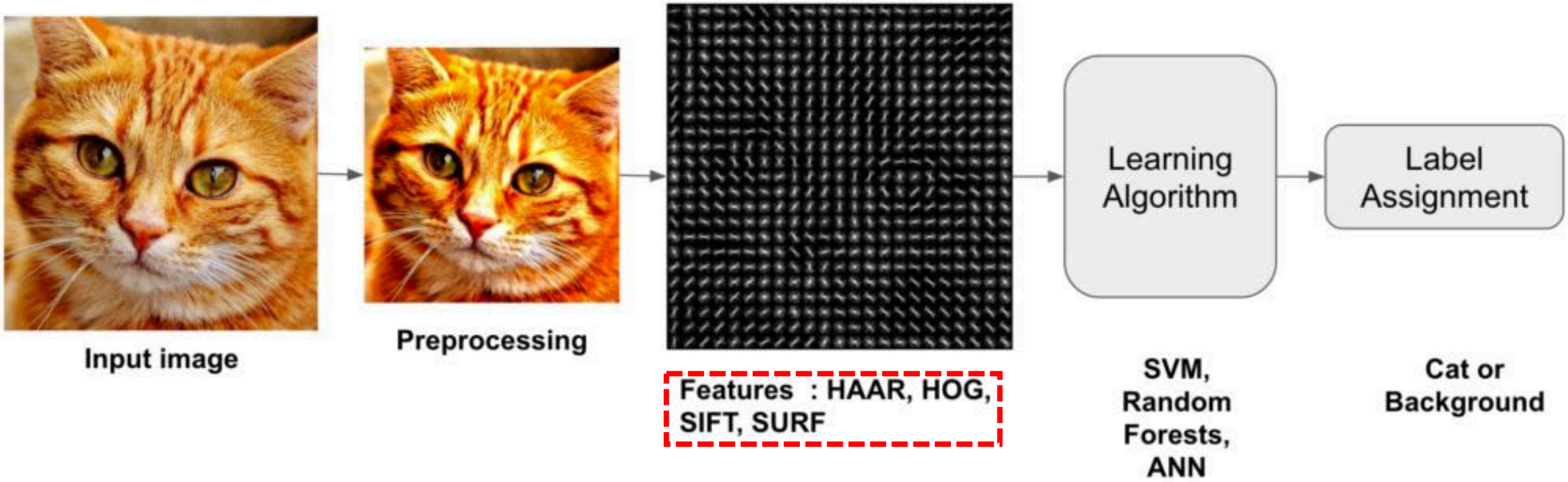
- Traditional Image Classification



Data → Feature extraction → Classification

Image classification

PIAI Research Department



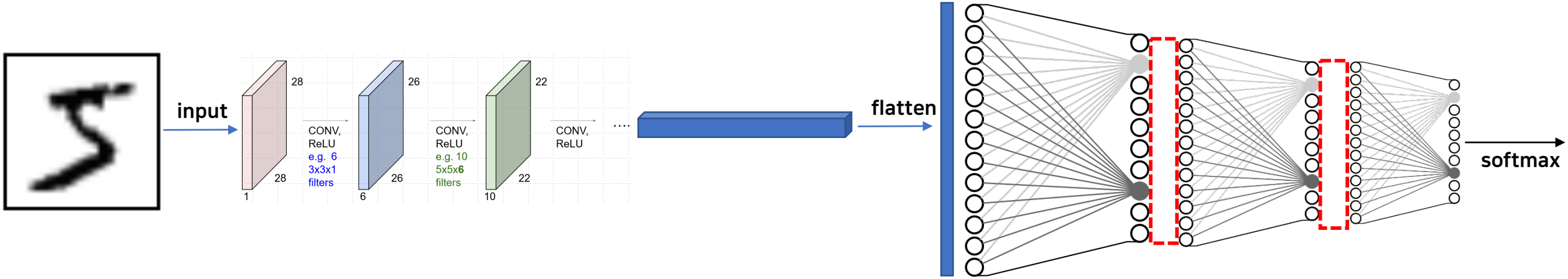
Data → Feature extraction → Classification

Convolution, max-pooling…

Overall Flow of Deep Learning

PIAI Research Department

- CNN

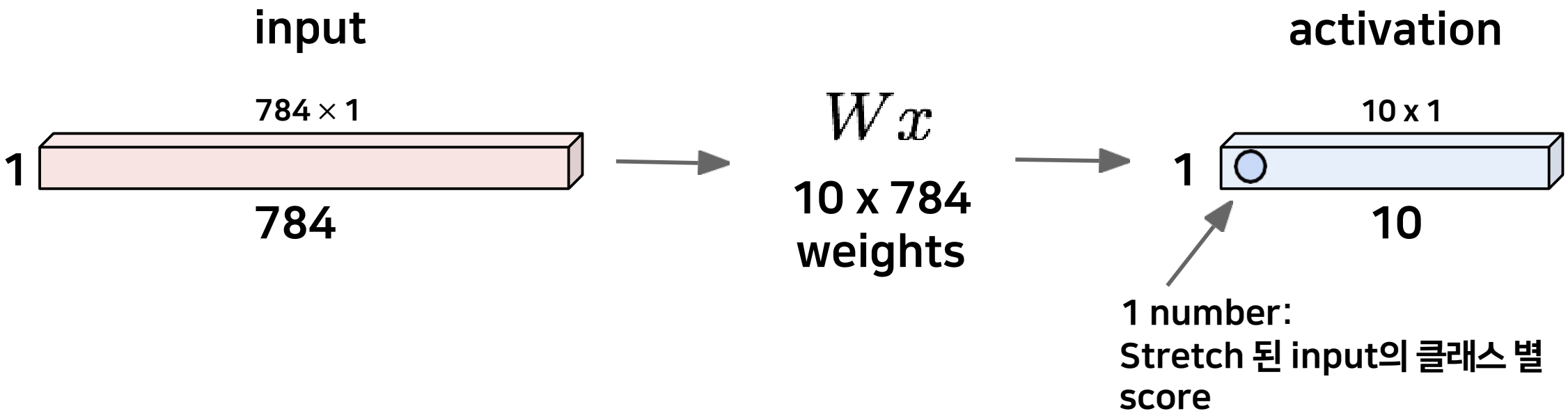


Feature extraction
(Convolution, Pooling layer ….)

DNN
(Fully Connected, Activation layer …)

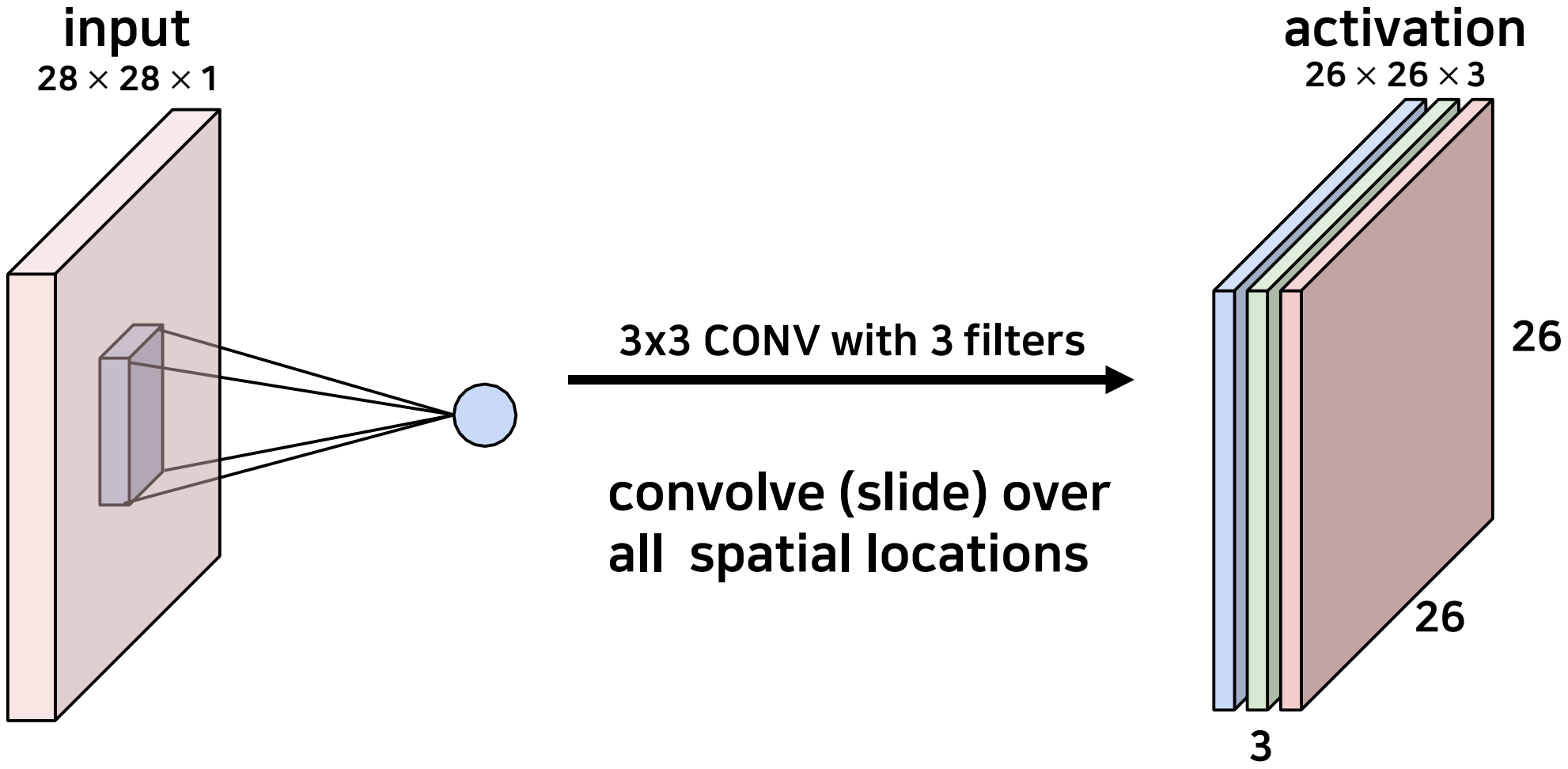
Fully Connected Layer

PIAI Research Department



Convolution Layer

PIAI Research Department

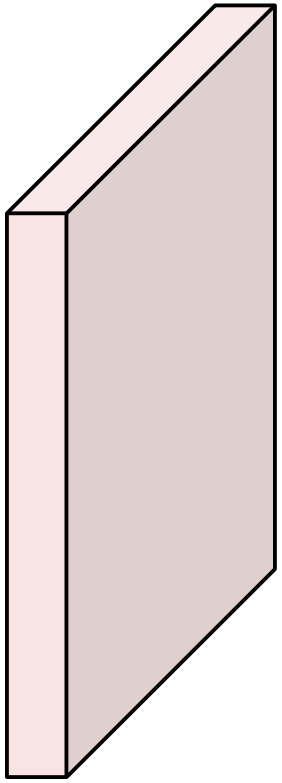


Convolution Layer

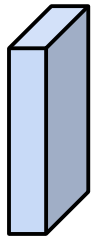
PIAI Research Department

Filter는 input data와
같은 depth를 가짐

28x28x1 image

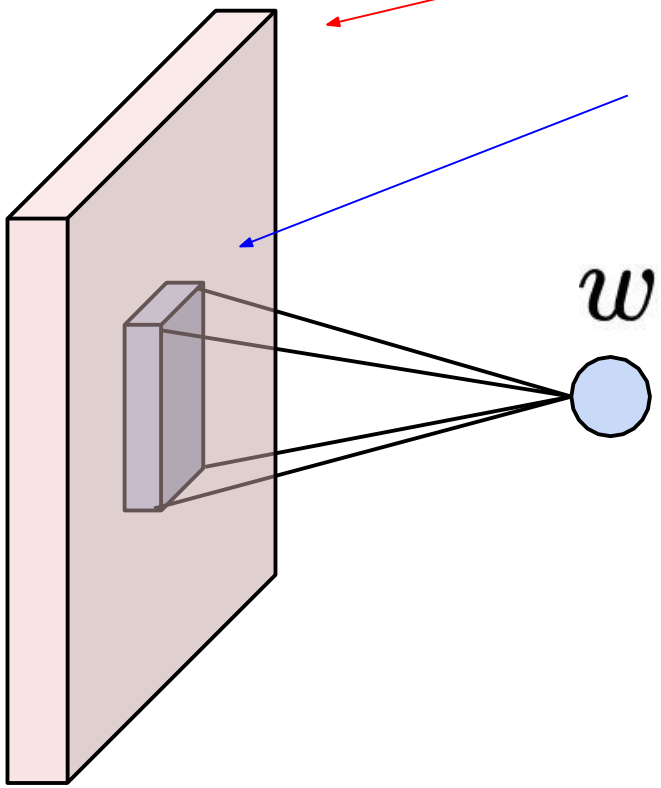


3x3x1 filter



Convolution Layer

PIAI Research Department



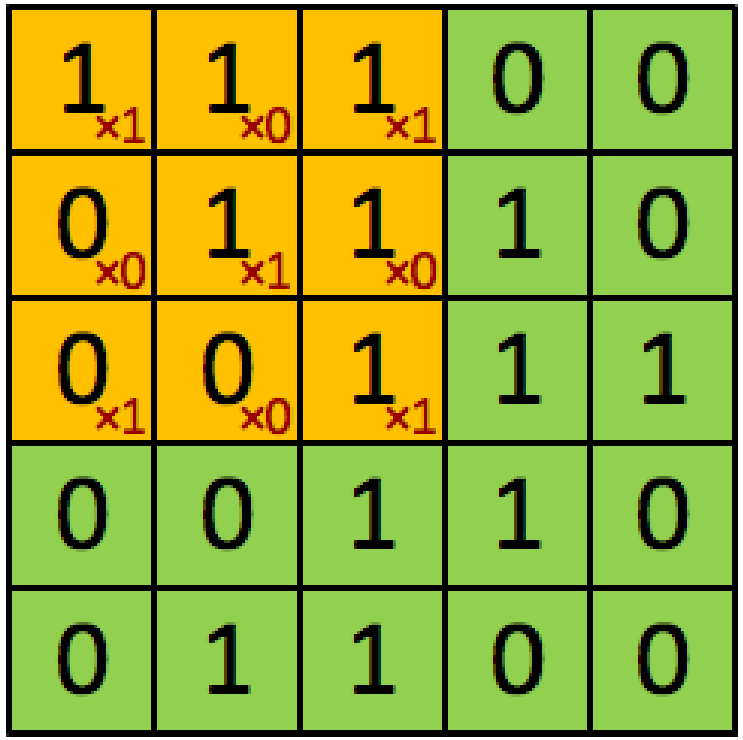
28x28x1 image
3x3x1 filter

1 number:
Image의 일부분과 filter를 element wise한 하나의 값이
activation map의 한 원소가 됨

$$w^T x + b$$

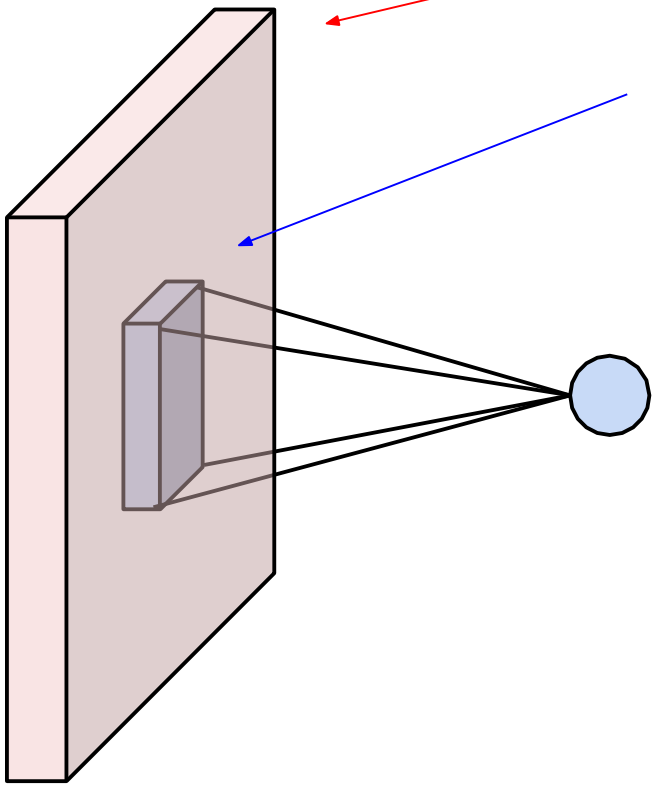
Convolution Layer

PIAI Research Department



Convolution Layer

PIAI Research Department



28x28x1 image
3x3x1 filter

convolve (slide) over
all spatial locations

activation map

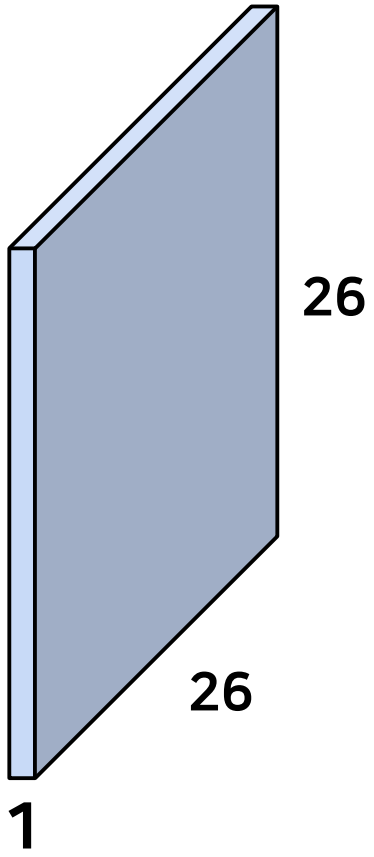
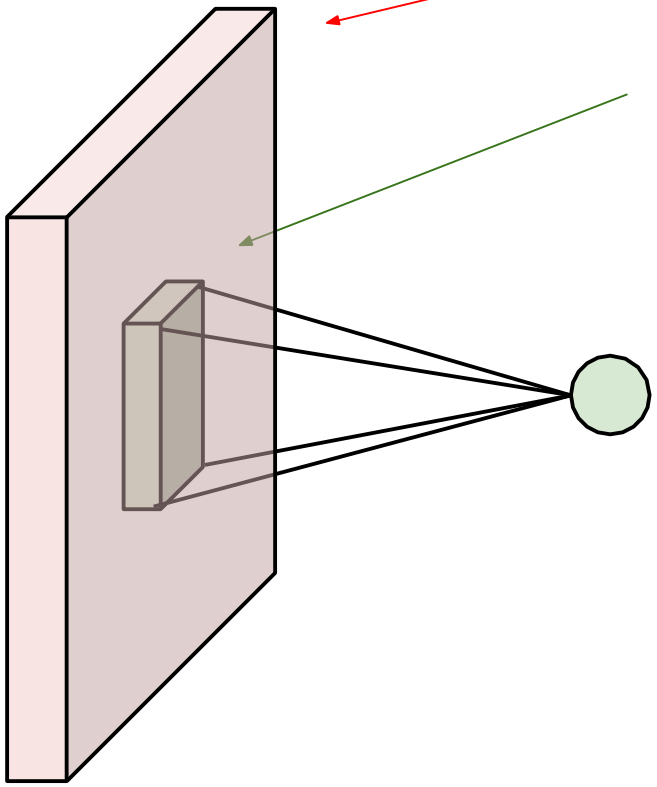


Image classification

PIAI Research Department

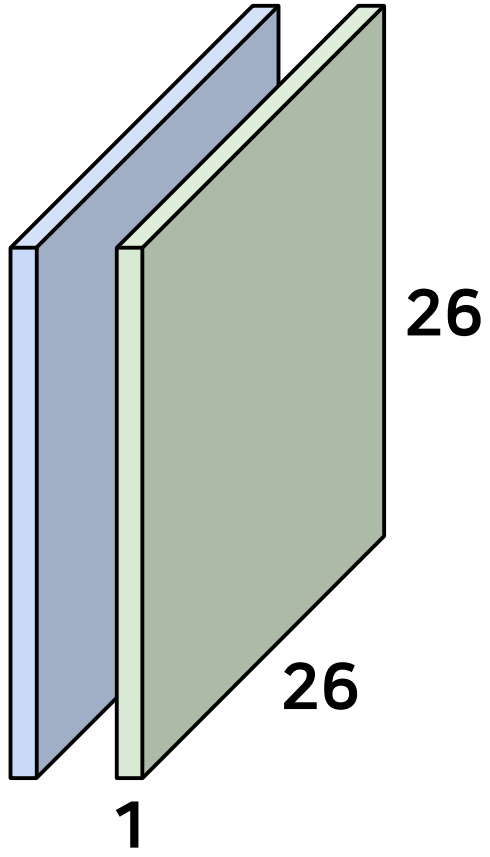


28x28x1 image

3x3x1 filter (second filter)

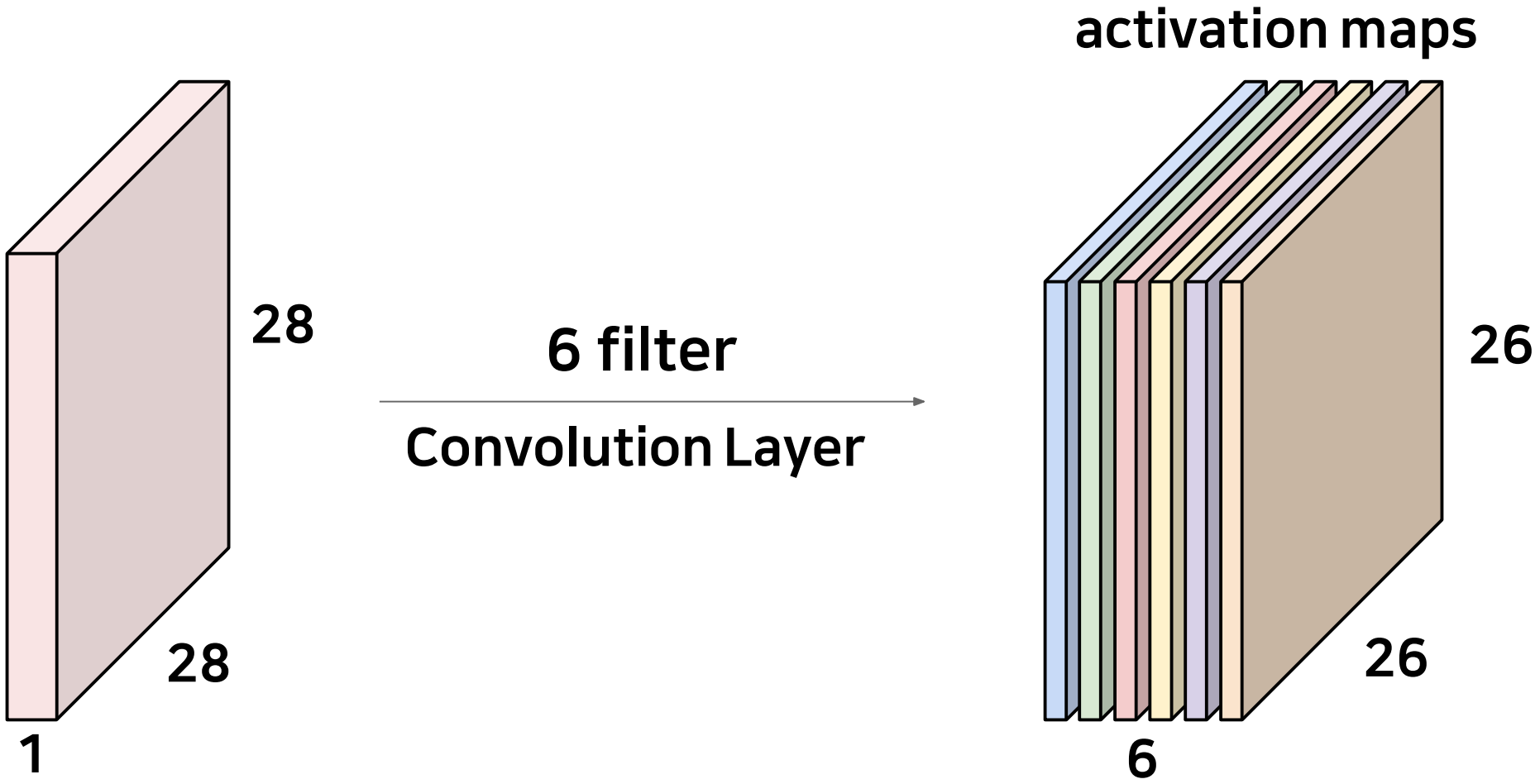
convolve (slide) over
all spatial locations

activation maps



Convolution Layer

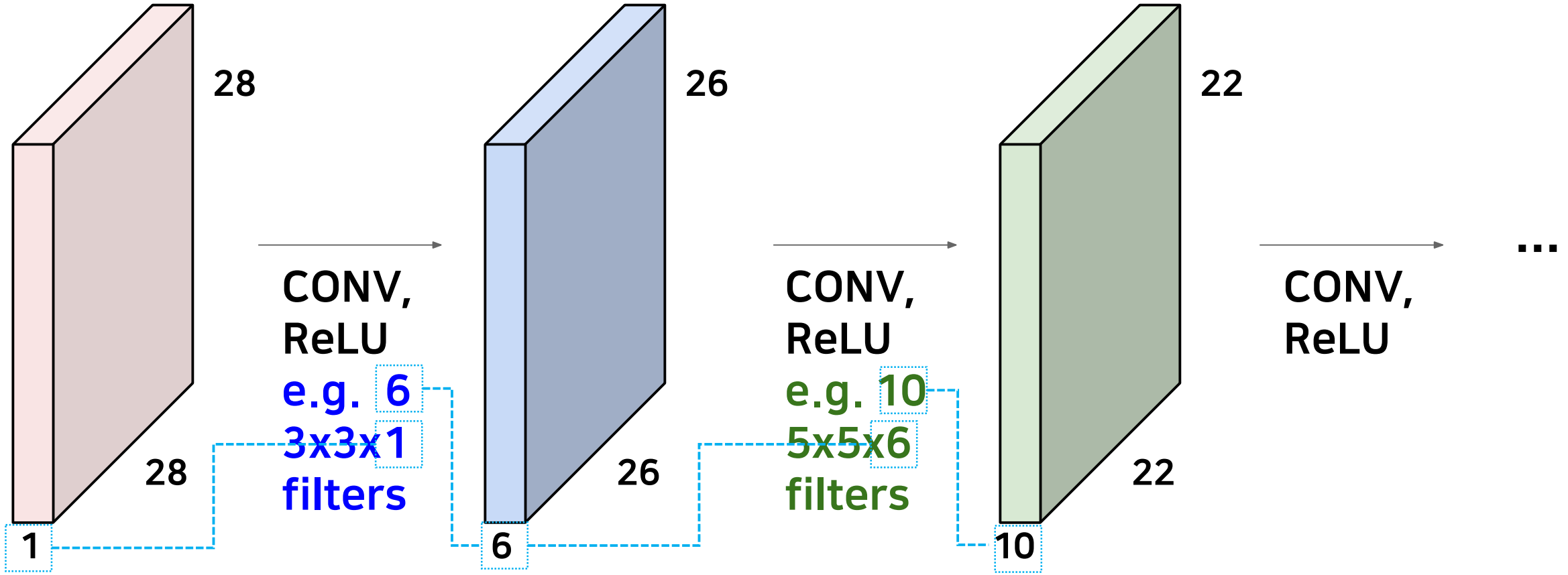
PIAI Research Department



26x26x6의 새로운 이미지는 다음 layer로 전달

Convolution Layer

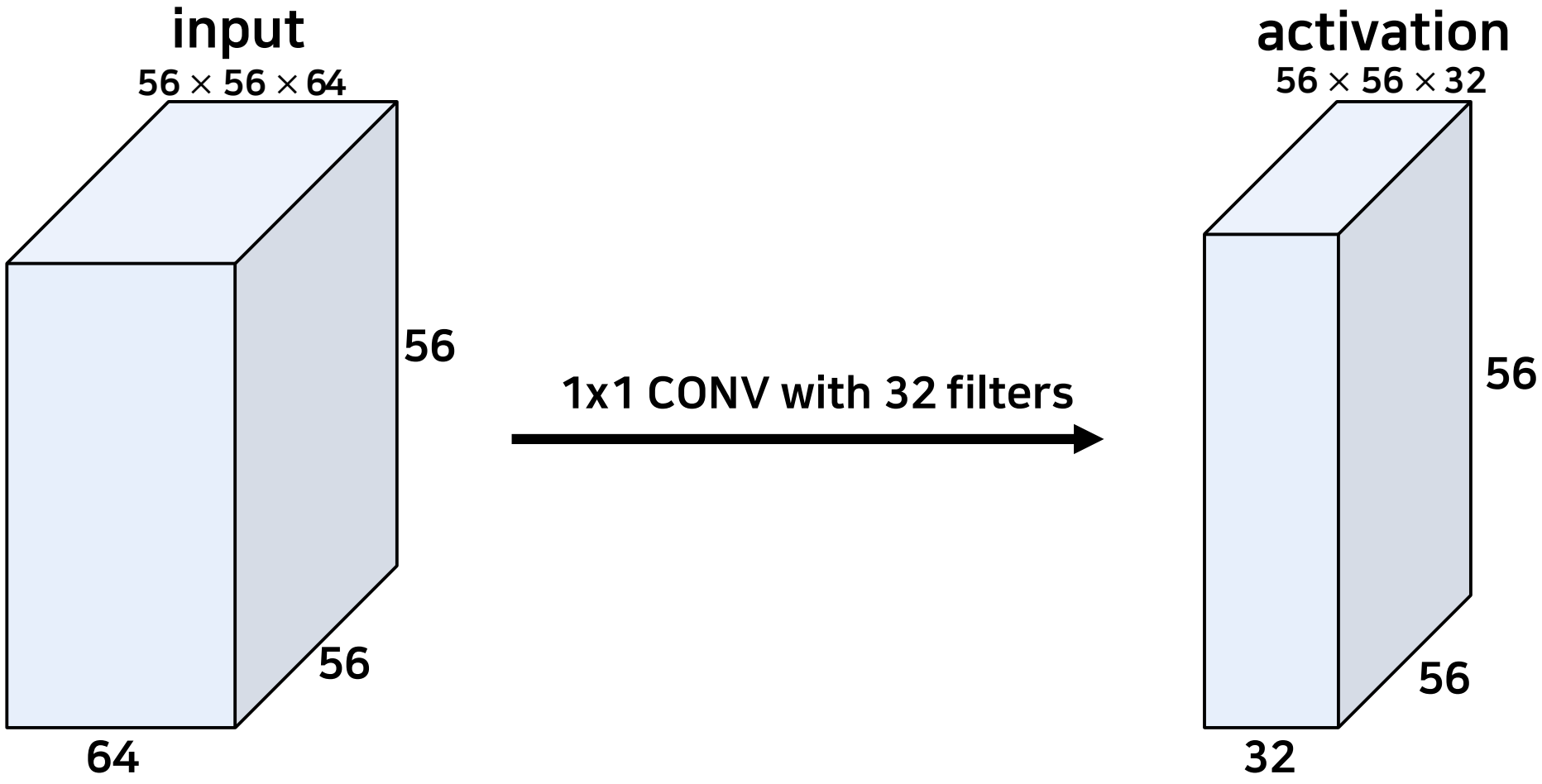
PIAI Research Department



각각의 필터는 input image만큼의 depth를 가져야한다.
그리고 필터를 거친 activation map은 필터의 개수만큼 만들어 진다.
Padding을 적용하지 않으면 이미지의 크기가 점점 줄어들어 convolution을 진행할 수 없다. (28 -> 26 -> 22 ...)

1 ×1 Convolution Layer

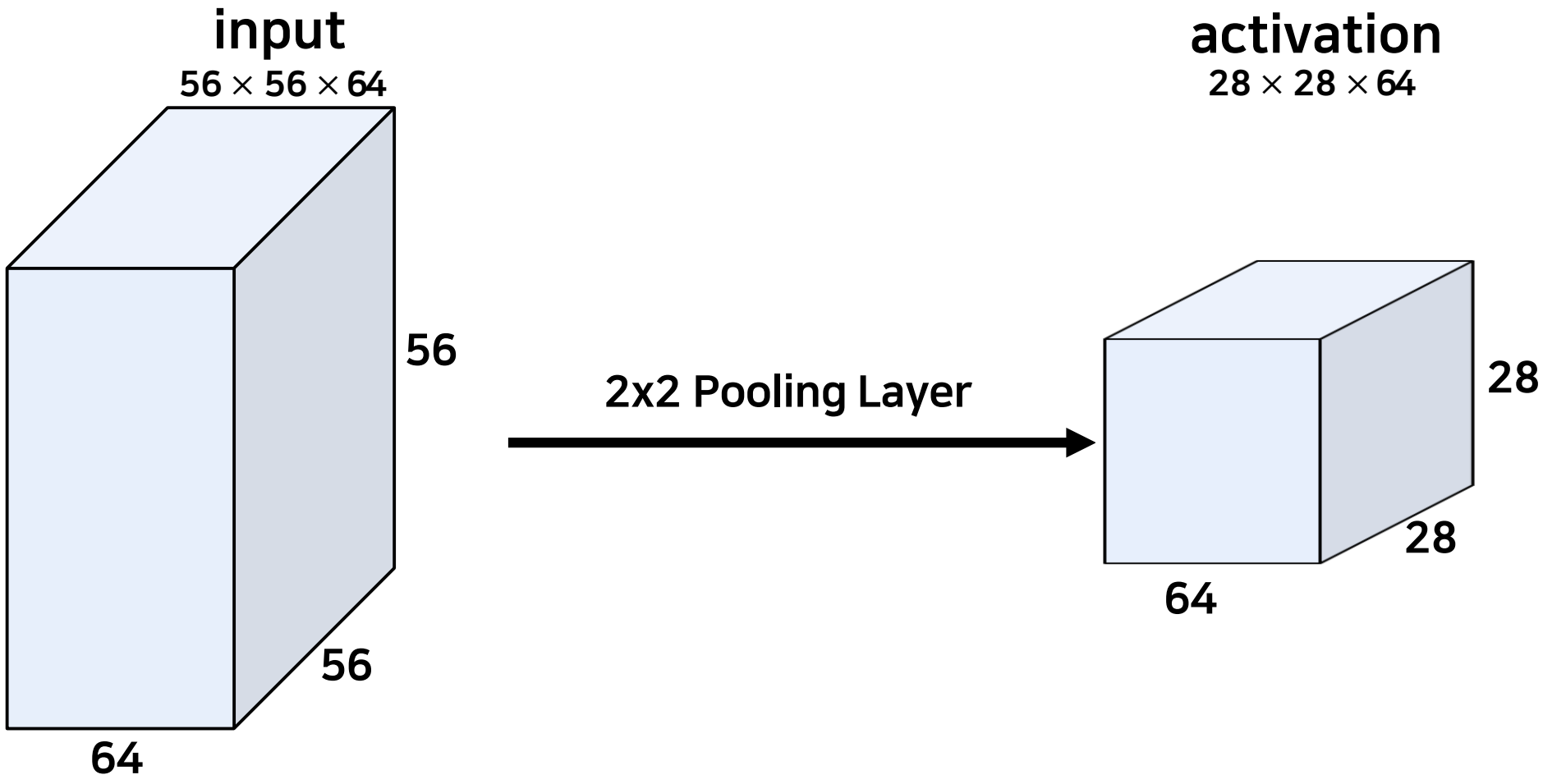
PIAI Research Department



1x1 convolution layers : 이미지 크기는 유지, depth를 줄임

Pooling Layer

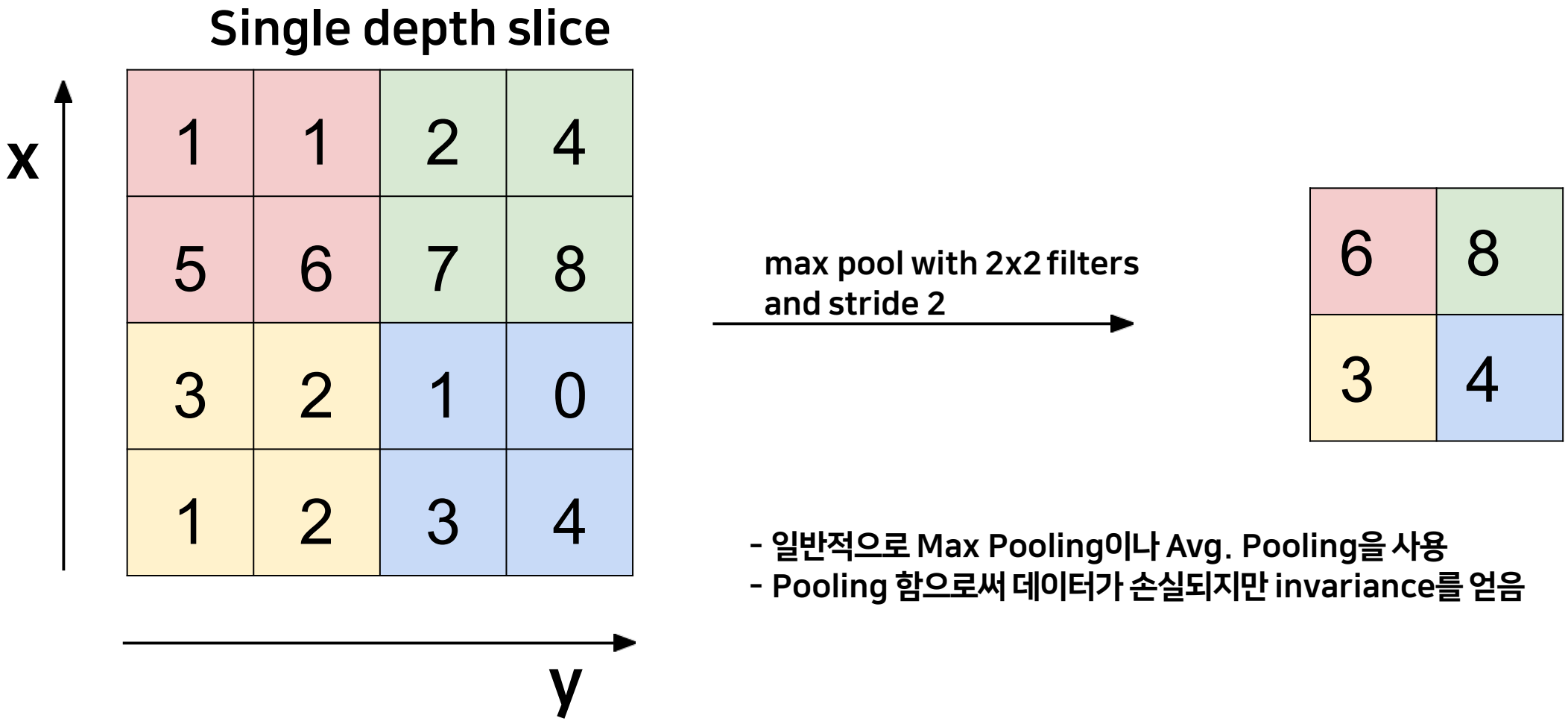
PIAI Research Department



Pooling layers : 이미지 크기를 줄이고, depth를 유지

Pooling Layer

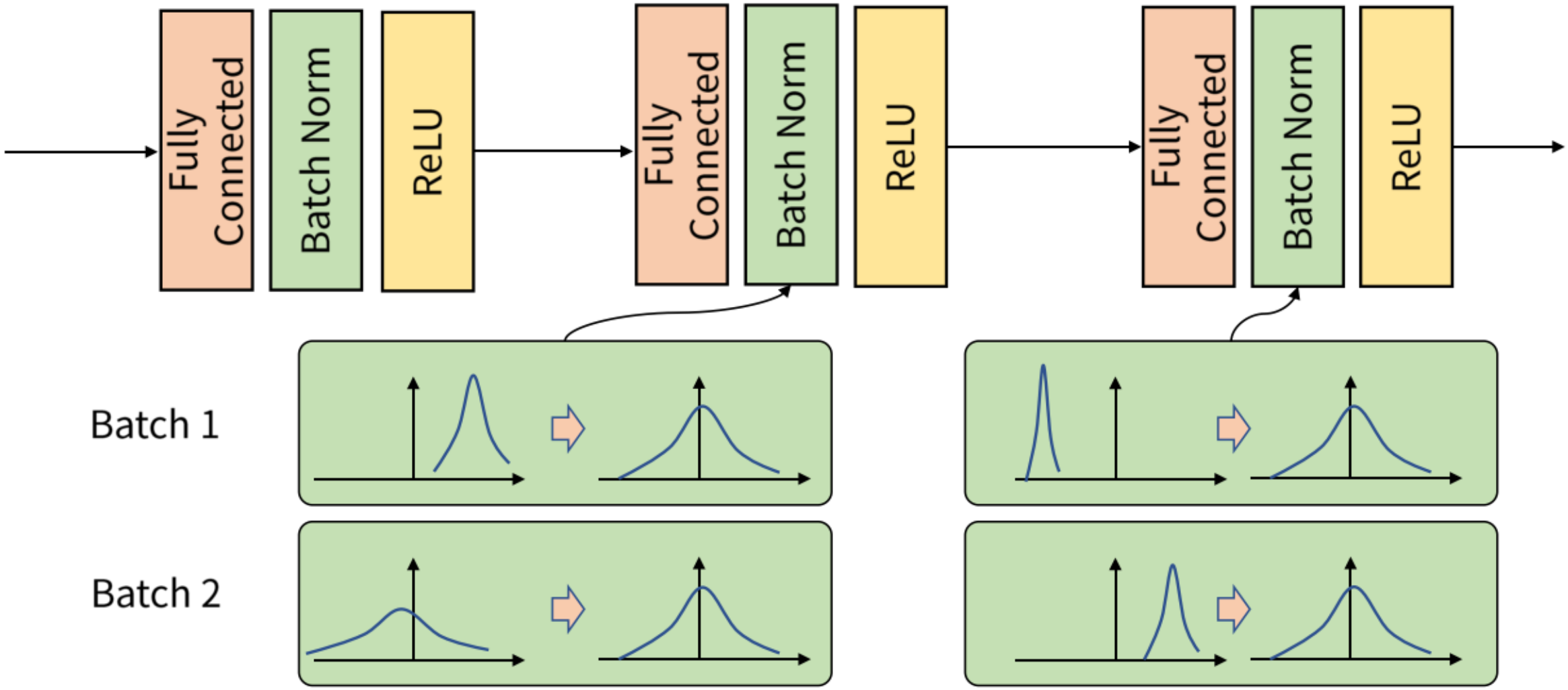
PIAI Research Department



- 일반적으로 Max Pooling이나 Avg. Pooling을 사용
- Pooling 함으로써 데이터가 손실되지만 invariance를 얻음

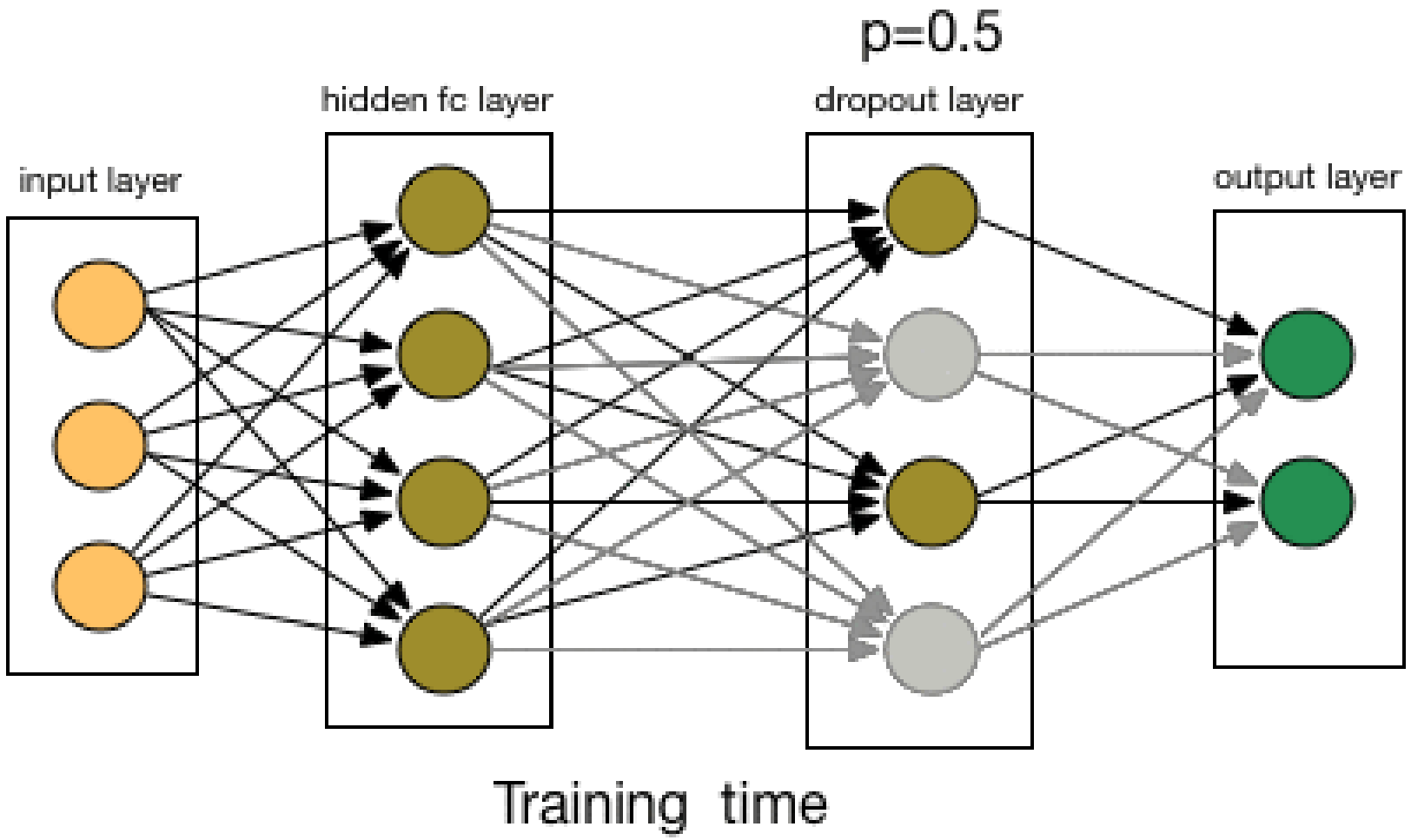
Batch Normalization

PIAI Research Department



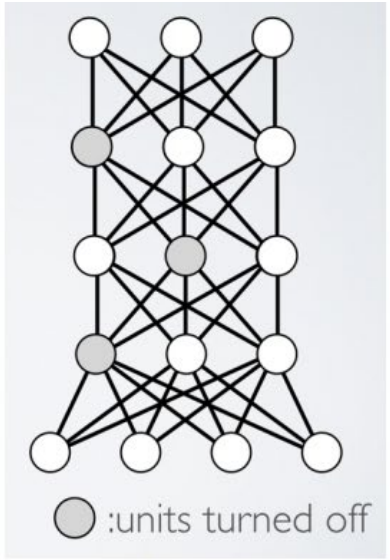
Dropout

PIAI Research Department

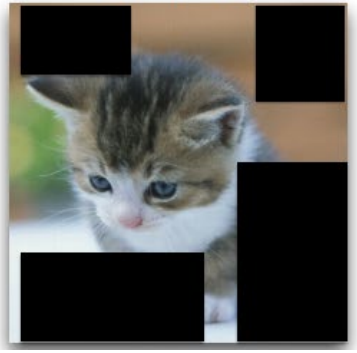


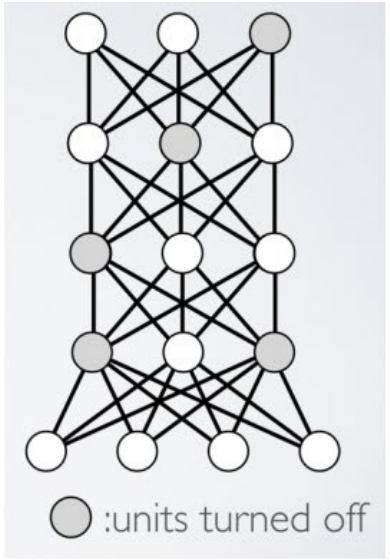
Dropout

PIAI Research Department

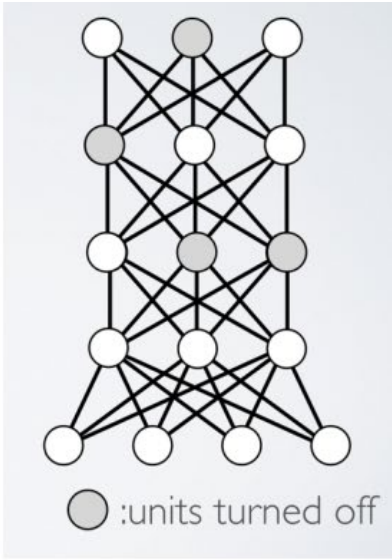


얼굴위주





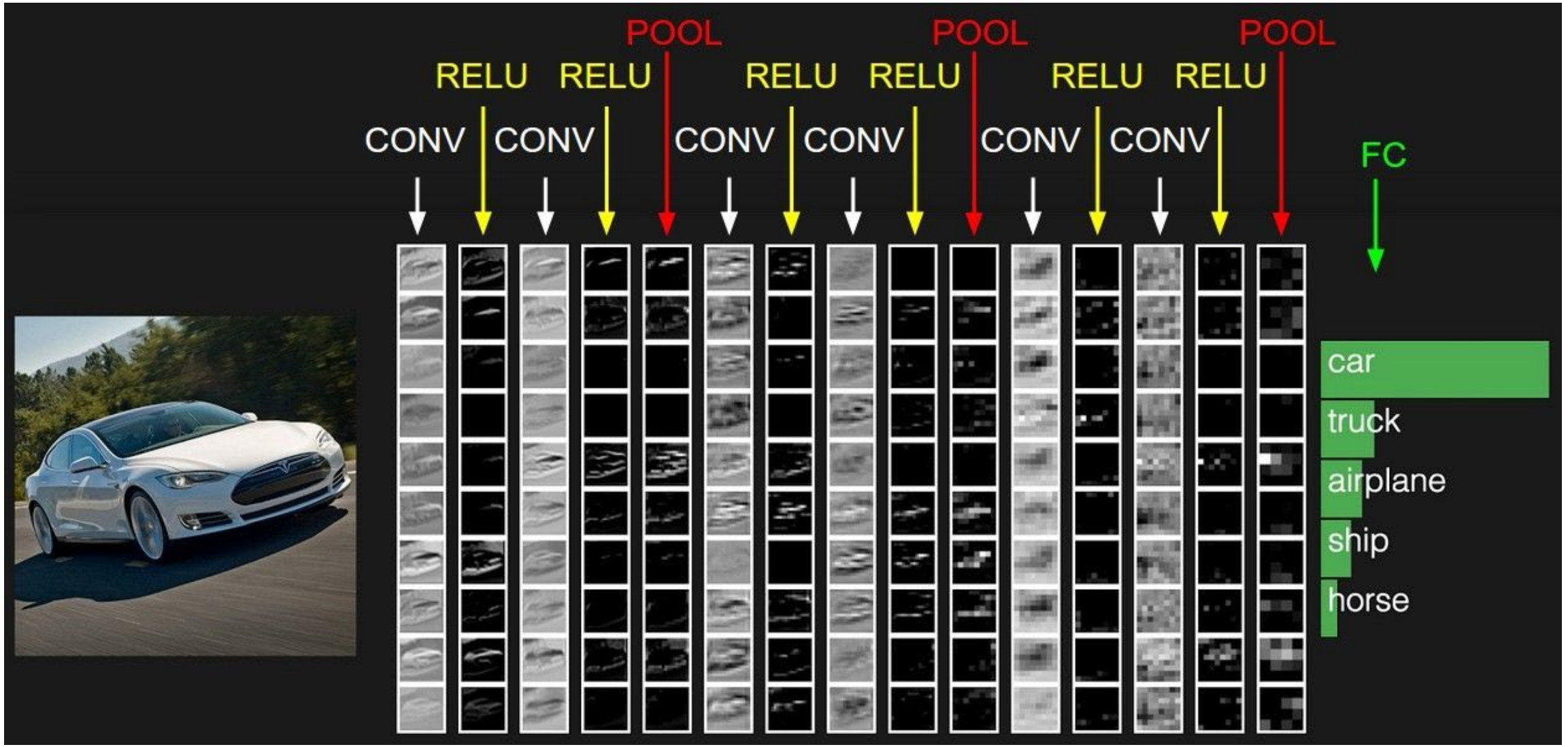
색지우고



귀 빼고

Image classification

PIAI Research Department



Code Running

2_KerasFeature Extraction.ipynb

PIAI
인공지능연구원
POSTECH Institute of Artificial Intelligence

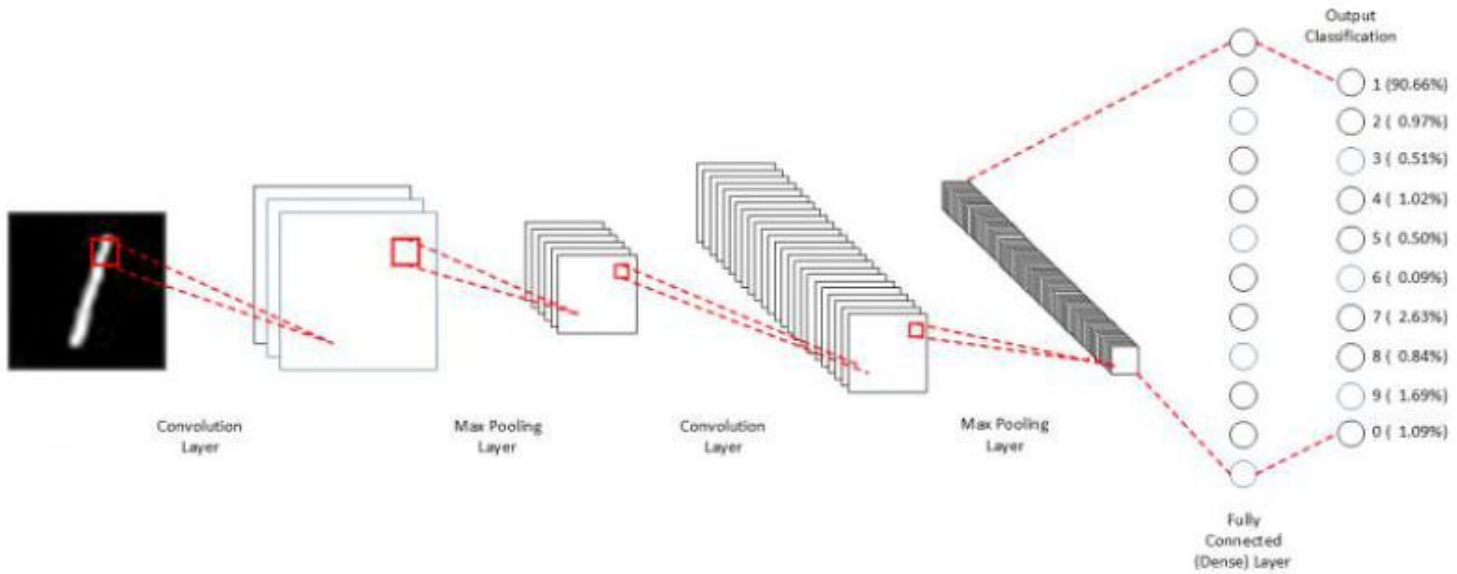
PyTorch Feature Extraction

I-Eon, Na

CNN Feature Extraction

PIAI Research Department

- **CNN – with MNIST**



- 특징을 추출하기 특화
- 기존 FC와 다르게 모든 뉴런들이 연결되어 있지 않음
- MLP에 비해 학습이 효율적이고, 모델의 성능도 더 좋음
- 여러가지 학습 기법 적용 예정 (Batch Nor., EarlySt., Dropout, Check Point Save 기법 등)

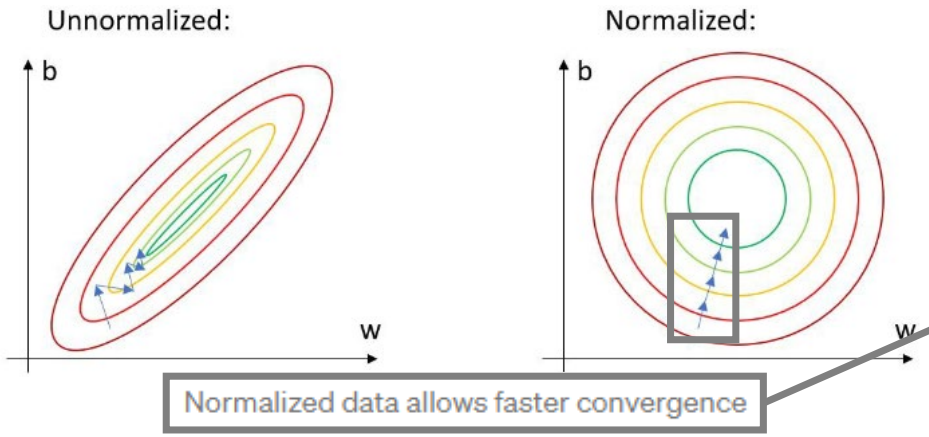
이미지 출처: https://minimin2.tistory.com/36

CNN – 데이터 전처리

PIAI Research Department

- **데이터 전처리 – Dataset Normalization**

```
transforms.Compose([ToTensor(),
                    Normalize(mean=(0.5,), std=(0.5, ))
```



데이터셋의 평균(mean)과 표준편차(std)를 계산하여 적용시

- 학습할 이미지 데이터셋이 일반적인 조도, 각도, 배경, 사물체가 아닌 경우는 직접 평균/표준편차를 계산하여 적용할 수 있습니다.

아래 함수는 이미지 데이터셋에 대하여 평균, 표준편차를 산출해 주는 함수 입니다.

```
def calculate_norm(dataset):
    # dataset의 axis=1, 2에 대한 평균 산출
    mean_ = np.array([np.mean(x.numpy(), axis=(1, 2)) for x, _ in dataset])
    # r, g, b 채널에 대한 각각의 평균 산출
    mean_r = mean_[:, 0].mean()
    mean_g = mean_[:, 1].mean()
    mean_b = mean_[:, 2].mean()
```

- MNIST 데이터 정규화 작업
(임의의 평균 & 표준편차 적용)
- 데이터 Normalize를 통하여
Global Minimum으로 빠르게 수렴
- 데이터의 평균과 표준편차를 직접 계산하여 적용 가능

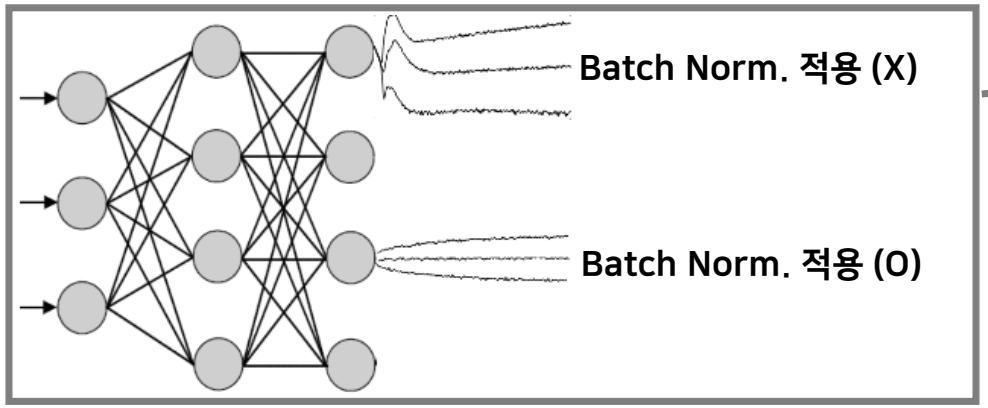
(링크: https://teddylee777.github.io/pytorch/torchvision-transform)

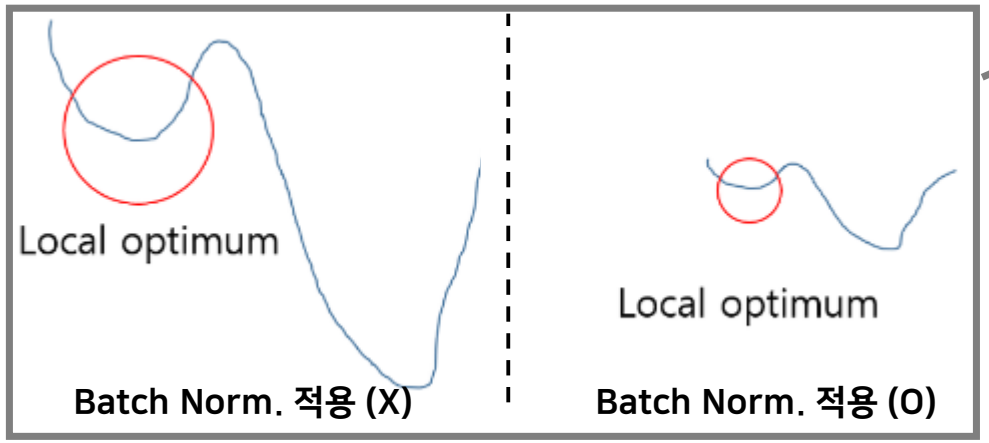
이미지 출처: https://towardsdatascience.com/how-to-calculate-the-mean-and-standard-deviation-normalizing-datasets-in-pytorch-704bd7d05f4c

CNN – 기능

- **기능 - Batch Normalization**

```
self.layer1 = torch.nn.Sequential(
    nn.Conv2d(in_channels=1, out_channels=32, kernel_
    nn.BatchNorm2d(32), #Batch Normalization을 수행
```





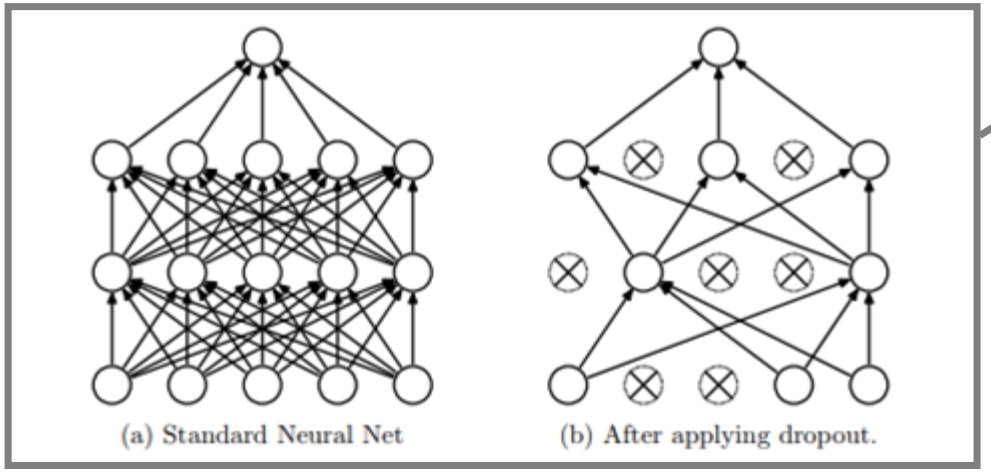
- 레이어 중간에 Batch Norm. 수행
- 이전 레이어의 파라미터 변화로 현재 레이어의 입력 분포가 바뀌는 현상 방지
- Local Minimum에 빠짐을 방지
- 수식 참조

(https://eehoeskrap.tistory.com/430#google_vignette)

CNN – 기능

- **기능 – Dropout**

```
self.layer1 = torch.nn.Sequential(
    nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, st
    nn.BatchNorm2d(32),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.MaxPool2d(kernel_size = 2, stride = 2),
)
```



- Dropout 기법을 사용하여 과적합을 방지
- 랜덤한 확률로 Layer 내에 있는 뉴런들을 제거 후 학습하는 기법
- 매 학습 마다 새롭게 제거된 뉴런들의 조합으로 인하여 앙상블 러닝 효과 발생
- 일반적으로 0 ~ 0.5 사이로 사용

출처: https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=isu112600&logNo=221578533182

CNN – 기능

- **기능 - Early Stopping 기법**

```python
class EarlyStopping:
    """주어진 patience 이후로 validation loss가 개선되지 않으면 학습을 조기 중지"""
    def __init__(self, patience=7, verbose=False, delta=0, path='checkpoint.pt'):
        self.patience = patience
        self.verbose = verbose
# Training Iteration
while epoch_not_finished():
    start_time = time.time()
    tloss, tacc = epoch(train_loader_es)
    end_time = time.time()
    time_taken = end_time - start_time
    record_train_log(tloss, tacc, time_taken)
    with torch.no_grad():
        vloss, vacc = epoch(valid_loader_es)
        record_valid_log(vloss, vacc)
    print_log()

    # early_stopping는 validation loss가 감소하였는지 확인이 필요하며,
    # 만약 감소하였을경우 현제 모델을 checkpoint로 만든다.
    early_stopping(vloss, net)
    if early_stopping.early_stop:
        print("Early stopping")
        break
```

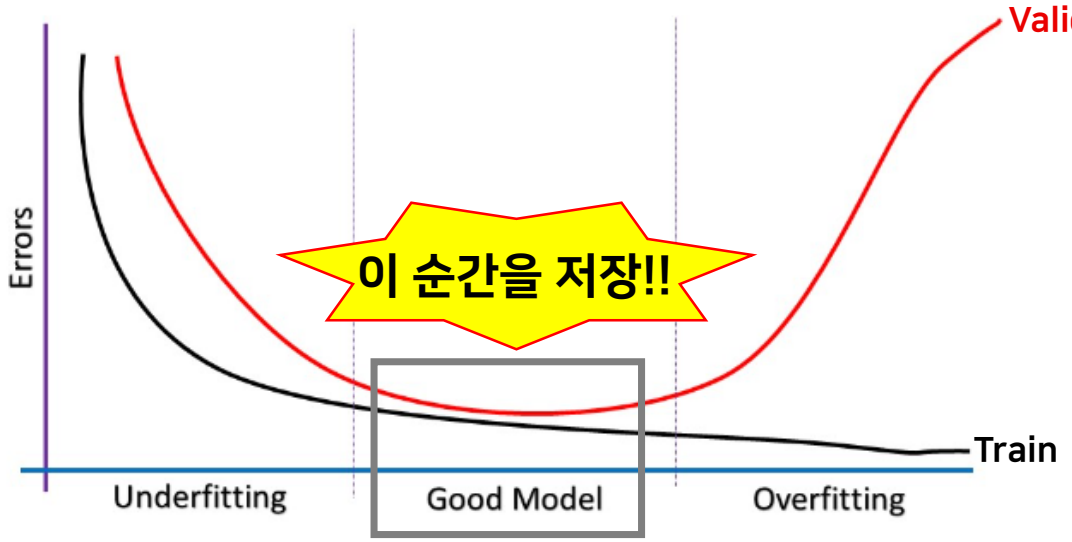
```
EarlyStopping counter: 10 out of 10
Early stopping

 Training completed!

====================== Test Result ======================

Test accuracy = 0.8904272151898734
Test loss = 0.4271463234496268
```

- 학습중 Over Fitting 판단시, 중간에 학습을 종료하는 기법
- [Patience]
  Validation Loss에 개선이 없으면 몇 번 기다렸다 학습을 중지 할지
- 학습 알고리즘에 Early Stopping 모듈을 추가하여 Over Fitting 감지 시, while문에서 break
- Early Stopping 작동 확인

CNN – 기능

- **기능 – Check Point Save 기법**

```
def __call__(self, val_loss, model):

    score = -val_loss

    if self.best_score is None:
        self.best_score = score
        self.save_checkpoint(val_loss, model)
def save_checkpoint(self, val_loss, model):
    # validation loss가 감소하면 모델을 저장한다.
    if self.verbose:
        print('Validation loss decresed ({:5} --> {:5}). Saving
    torch.save(model.state_dict(), self.path)
    self.val_loss_min = val_loss
```



- 학습 중 모델이 최고 점수를 기록하는 순간을 지속적으로 Epoch마다 저장
- EarlyStopping 이랑 비슷한 목적
- 학습 결과 Overfitting이 되었다고 하더라도, Best Model 보관 가능

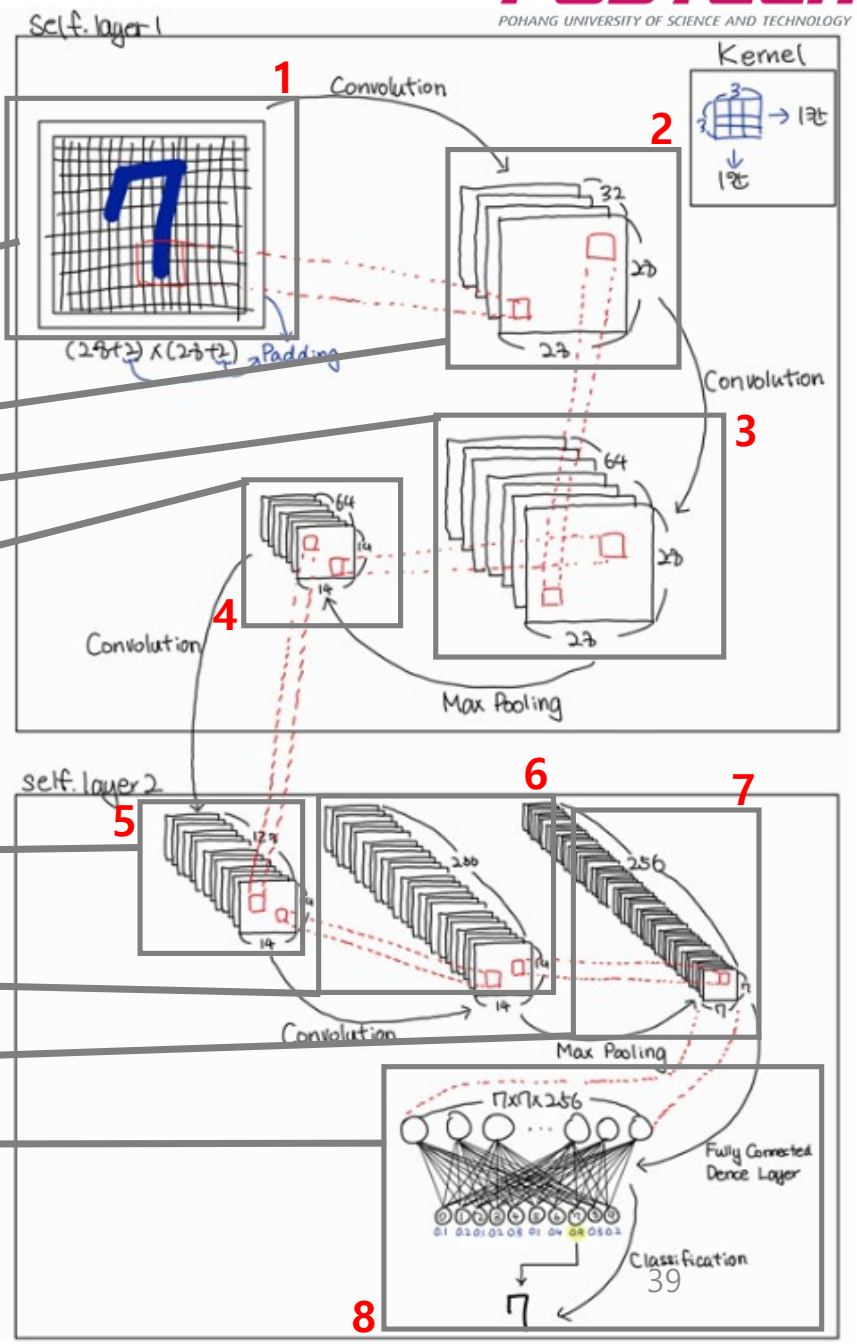
출처: https://meditationsonbianddatascience.com/2017/05/11/overfitting-underfitting-how-well-does-your-model-fit/

CNN – 모델 설계

PIAI Research Department

- **CNN(Convolutional Neural Network) 모델 설계**

```
class CNN_model(nn.Module):
    def __init__(self):
        super(CNN_model, self).__init__()

        # 1번 레이어 생성
        self.layer1 = torch.nn.Sequential(
            nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, stride=1, padding=1),

            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.MaxPool2d(kernel_size = 2, stride = 2))

        # 2번 레이어 생성
        self.layer2 = torch.nn.Sequential(
            nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1),

            nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = 2, stride = 2))

        #마지막 0 ~ 9 classification을 위한 fully connected layer 생성
        self.fc1 = nn.Linear(7 * 7 * 256, 10, bias = True)

        #가중치 초기화
        nn.init.xavier_uniform_(self.fc1.weight)
```



CNN – 모델 설계

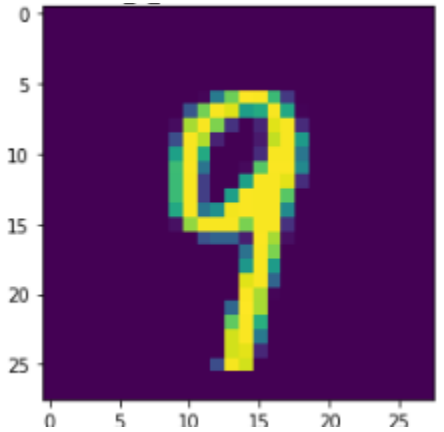
- **설계된 CNN 모델의 구조 확인**

```
43 # Model structure check
44 Summary(CNN_model().to(device), (1, 28, 28))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 28, 28]             320
            Conv2d-2           [-1, 64, 28, 28]          18,496
       BatchNorm2d-3           [-1, 64, 28, 28]             128
              ReLU-4           [-1, 64, 28, 28]               0
           Dropout-5           [-1, 64, 28, 28]               0
         MaxPool2d-6           [-1, 64, 14, 14]               0
            Conv2d-7          [-1, 128, 14, 14]          73,856
            Conv2d-8          [-1, 256, 14, 14]         295,168
       BatchNorm2d-9          [-1, 256, 14, 14]             512
             ReLU-10          [-1, 256, 14, 14]               0
        MaxPool2d-11            [-1, 256, 7, 7]               0
           Linear-12                  [-1, 10]         125,450
================================================================
Total params: 513,930
Trainable params: 513,930
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 3.25
Params size (MB): 1.96
Estimated Total Size (MB): 5.22
----------------------------------------------------------------
```

- Summary 기능으로 설계된 모델의 구조 확인
- 입력 데이터 사이즈 확인
  (1, 28, 28)

  1: Gray Scale (1 channel 이미지)
  28 * 28: 사진의 화소



CNN – 모델 설계

- **CNN 모델 학습 알고리즘**

```python
# 학습 알고리즘
def epoch(data_loader, mode = 'train'):
    global epoch_cnt

    # 사용되는 변수 초기화
    iter_loss, iter_acc, last_grad_performed, last_out, last_label = [],

    for _data, _label in data_loader:
        data, label = _data.to(device), _label.to(device)

        # 1. Feed-forward
        if mode == 'train':
            net.train()
        else:
            net.eval()
        result, _ = net(data)
        _, out = torch.max(result, 1)
```

```python
        # 5. 컨퓨전 매트릭스 출력을 위해 기록
        last_out = out.cpu().detach()
        last_label = _label
```

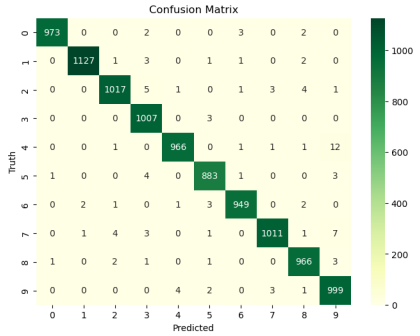
- Day1 알고리즘에서 **2개** 기능 추가

1) Train/Valid 모드에 따라 다르게 동작

- net.train() :
  dropout, batch nor. -> 사용

- net.eval() :
  dropout, batch nor. -> 미사용

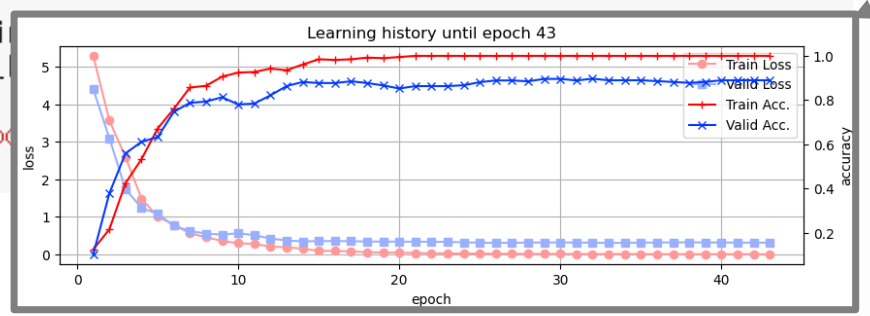
2) Confusion Matrix 기록 로직 추가



CNN – Verbose Setting(Graph)

- **Verbose – 실시간 학습 추이 그래프 출력**

```
# 학습 추이 그래프 출력
hist_fig, loss_axis = plt.subplots(figsize=(10, 3), dpi=99) # 그래프 사이즈 설정
hist_fig.patch.set_facecolor('white') # 그래프 배경색 설정

# Loss Line 구성
loss_t_line = plt.plot(iter_log, tloss_log, label='Train Loss', color='red', marker='o')
loss_v_line = plt.plot(iter_log, vloss_log, label='Valid Loss', color='blue', marker='s')
loss_axis.set_xlabel('epoch')
loss_axis.set_ylabel('loss')

# Acc. Line 구성
acc_axis = loss_axis.twinx()
acc_t_line = acc_axis.plot(iter_log, tacc_log, label='Train Acc.', color='red', marker='+')
acc_v_line = acc_axis.plot(iter_log, vacc_log, label='Valid Acc.', color='blue', marker='x')
acc_axis.set_ylabel('accuracy')

# 그래프 출력
hist_lines = loss_t_line + loss_v_li
loss_axis.legend(hist_lines, [l.get_l
loss_axis.grid() # 격자 설정
plt.title('Learning history until epo
plt.draw()
```



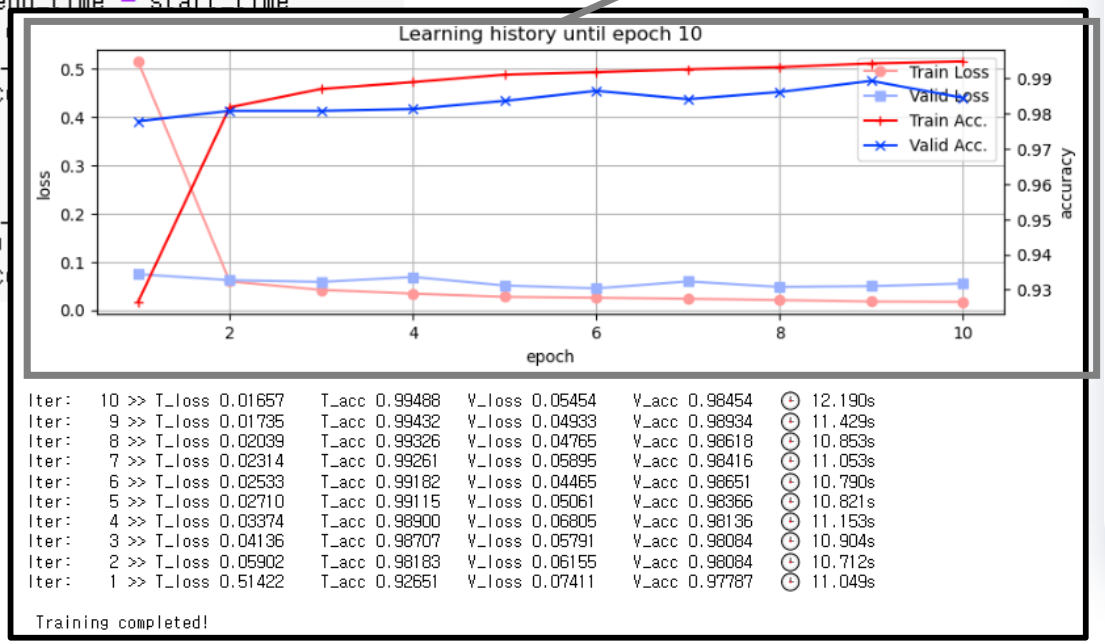
- 학습 추이를 그래프를 통하여 실시간으로 모니터링
- Train과 Validation에 대하여 Loss와 Acc.를 실시간으로 시각화

CNN – 모델 학습

PIAI Research Department

- **CNN 모델 학습**

```
# Training Initialization
init_model()
init_epoch()
init_log()
plt.rc('font', size = 10)

best_test_acc, best_epoch = 0, 0
# Training Iteration
while epoch_not_finished():
    start_time = time.time()
    tloss, tacc = epoch(train_loader)
    end_time = time.time()
    time_taken = end_time - start_time
    record_train_l
    with torch.no_
        vloss, vac
        record_val
    print_log()

    with torch.no_
        net.eval()
        _, test_ac
```




```
Iter:   10 >> T_loss 0.01657   T_acc 0.99488   V_loss 0.05454   V_acc 0.98454   12.190s
Iter:    9 >> T_loss 0.01735   T_acc 0.99432   V_loss 0.04933   V_acc 0.98934   11.429s
Iter:    8 >> T_loss 0.02039   T_acc 0.99326   V_loss 0.04765   V_acc 0.98618   10.853s
Iter:    7 >> T_loss 0.02314   T_acc 0.99261   V_loss 0.05895   V_acc 0.98416   11.053s
Iter:    6 >> T_loss 0.02533   T_acc 0.99182   V_loss 0.04465   V_acc 0.98651   10.790s
Iter:    5 >> T_loss 0.02710   T_acc 0.99115   V_loss 0.05061   V_acc 0.98366   10.821s
Iter:    4 >> T_loss 0.03374   T_acc 0.98900   V_loss 0.06805   V_acc 0.98136   11.153s
Iter:    3 >> T_loss 0.04136   T_acc 0.98707   V_loss 0.05791   V_acc 0.98084   10.904s
Iter:    2 >> T_loss 0.05902   T_acc 0.98183   V_loss 0.06155   V_acc 0.98084   10.712s
Iter:    1 >> T_loss 0.51422   T_acc 0.92651   V_loss 0.07411   V_acc 0.97787   11.049s

Training completed!
```

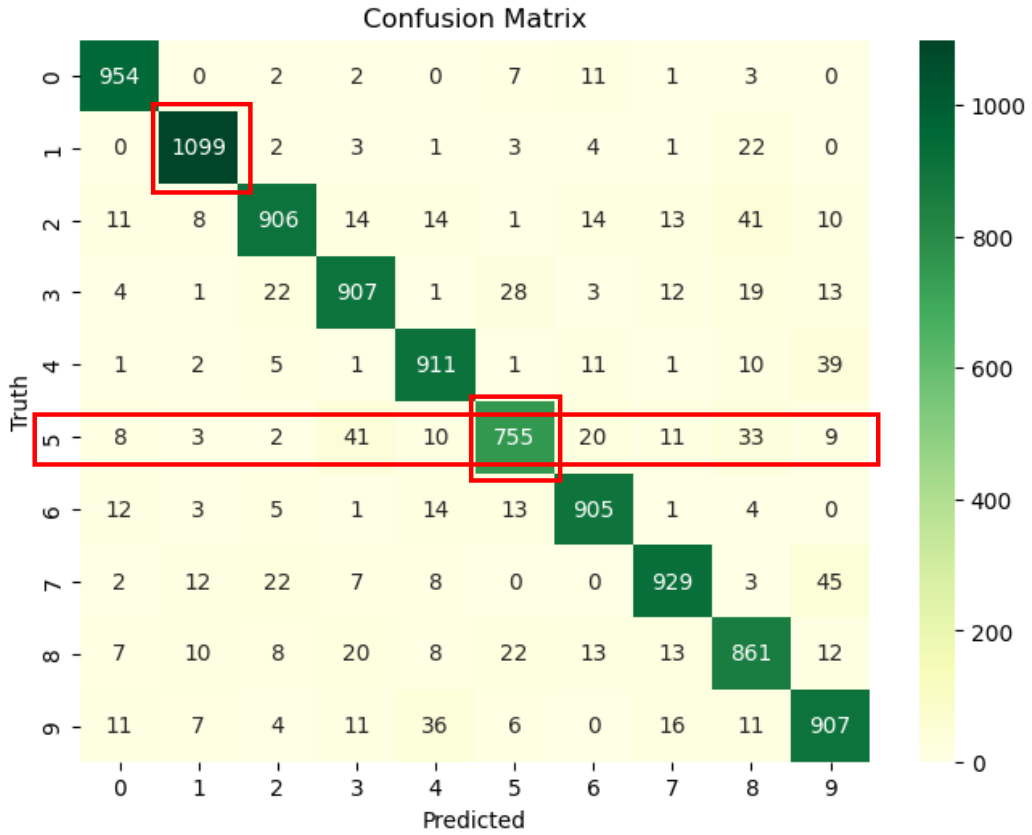
- CNN 모델 학습
- 실시간 그래프로 학습 추이 확인
- 모델의 정확도 확인

```
Test Acc.: 0.9898
Test Loss: 0.0321
```

CNN – 결과 분석

- **Confusion Matrix**

```
# Confusion matrix
our_cmatrix = confusion_matrix(test_label, test_out)
plt.figure(figsize=(8, 6), dpi=99)
sns.heatmap(our_cmatrix, annot=True, fmt='g', cmap='YlGn').set(xlabel='Predicted', ylabel='Truth')
plt.title('Confusion Matrix')
plt.show()
```



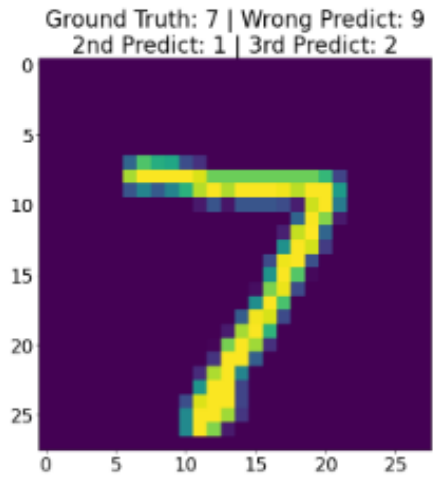
- 혼동 행렬을 통하여 모델이 예측하는 10개의 Class에 대한 성능을 한눈에 가시화
- 모든 Class에 대하여 모델이 분류한 결과 확인

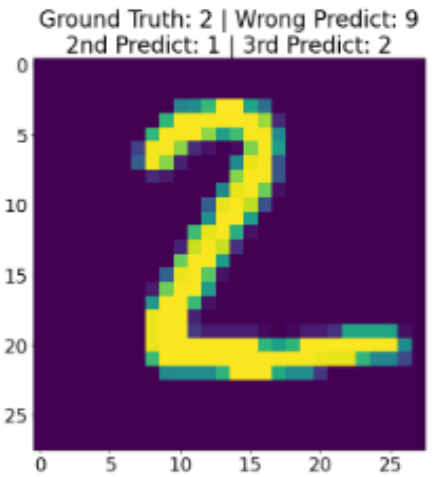
CNN – 결과 분석

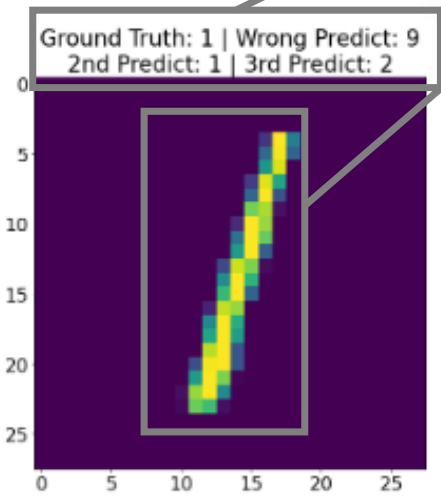
- **모델의 오 분류 사례 분석**

```
# 예측 결과 1, 2, 3순위를 만드는 과정
max_pred = max(pred_info)
second_pred = pred_info[1]
third_pred = pred_info[2]
pred_label = pred_info.index(max_pred)
second_label = pred_info.index(second_pred)
thirt_label = pred_info.index(third_pred)
```

```
# 정답과 예측 결과가 다른 경우(모델이 잘못 분류한 경우)
if gt_label != pred_label:
    # plt로 4개를 한번에 출력하기 때문에 4번동안 리스트에 누적시킨다
    show_cnt += 1
    wrong_pic, _ = test_data[cnt]
    plt_pic_list.append(wrong_pic)
    gt_label_list.append(gt_label)
    pred_label_list.append(pred_label)
    second_pred_label_list.append(second_label)
    third_pred_label_list.append(thirt_label)
    plt_cnt += 1
```







- 학습된 모델이 잘못 분류한 케이스 분석
- 모델의 Prediction 결과가 Ground Truth랑 다른 경우를 예측 확률 순위로 1, 2, 3위 출력
  (정답지가 1인 사진을 9로 잘못 예측했으며, 두번째 세번째 후보는 1과 2)
- 모델이 잘못했는지, 데이터가 잘못되었는지를 판단

Code Running

2. Pytorch CNN – Feature Extraction.ipynb