KAIST Korea Advanced Institute of Science and Technology

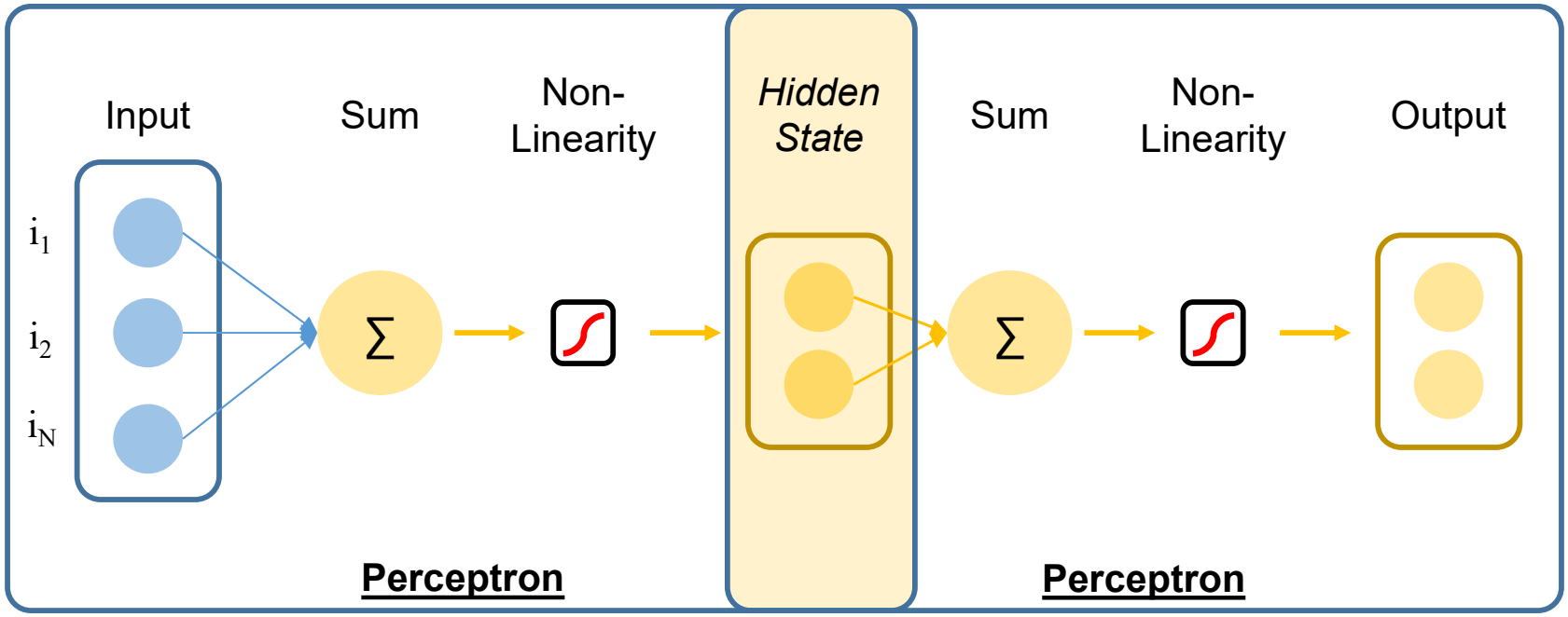
Week 4: MNIST with CNN

KAIST

Mobile Robitics & Intelligence Lab

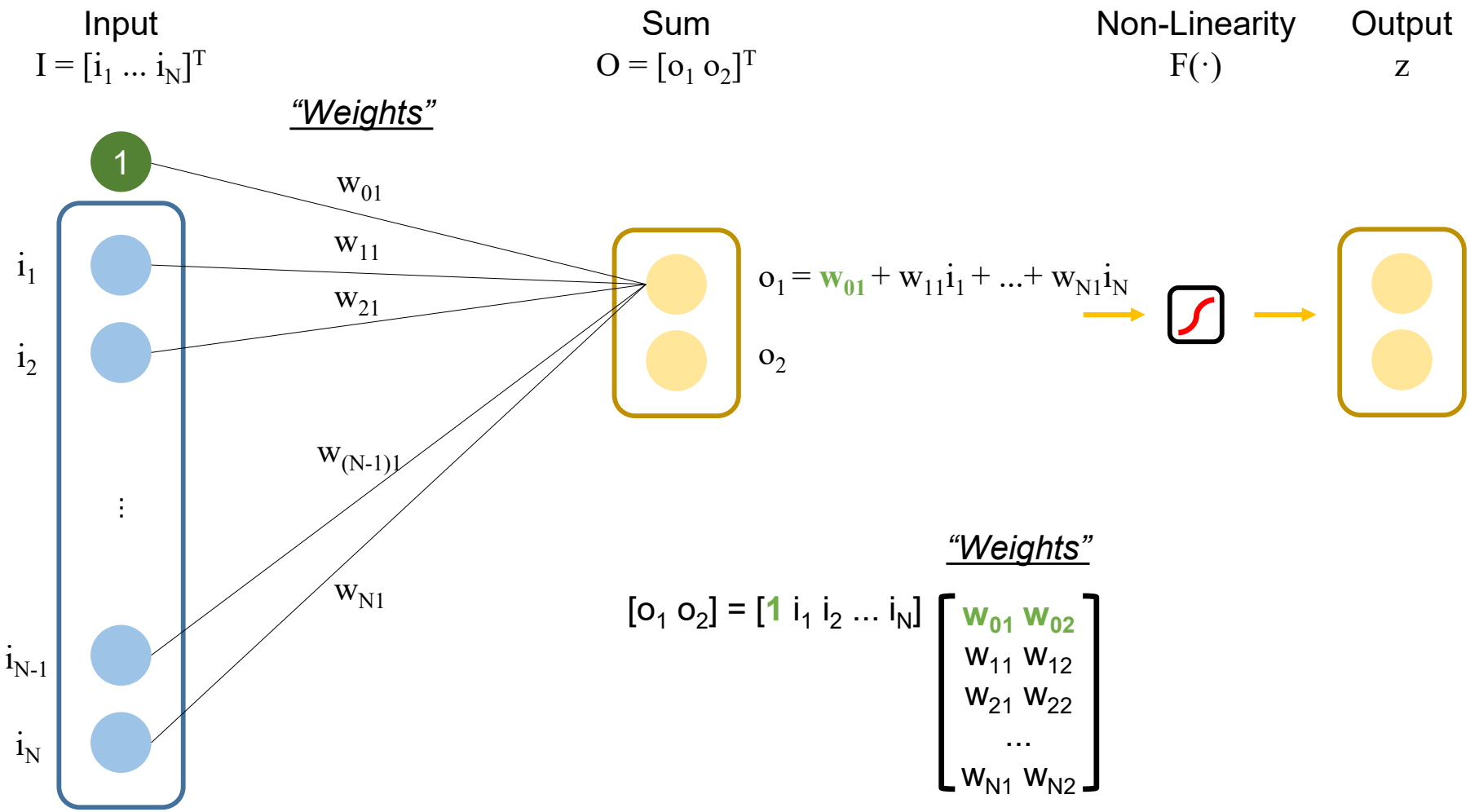
박사과정 김진식

Multi-layer perceptron



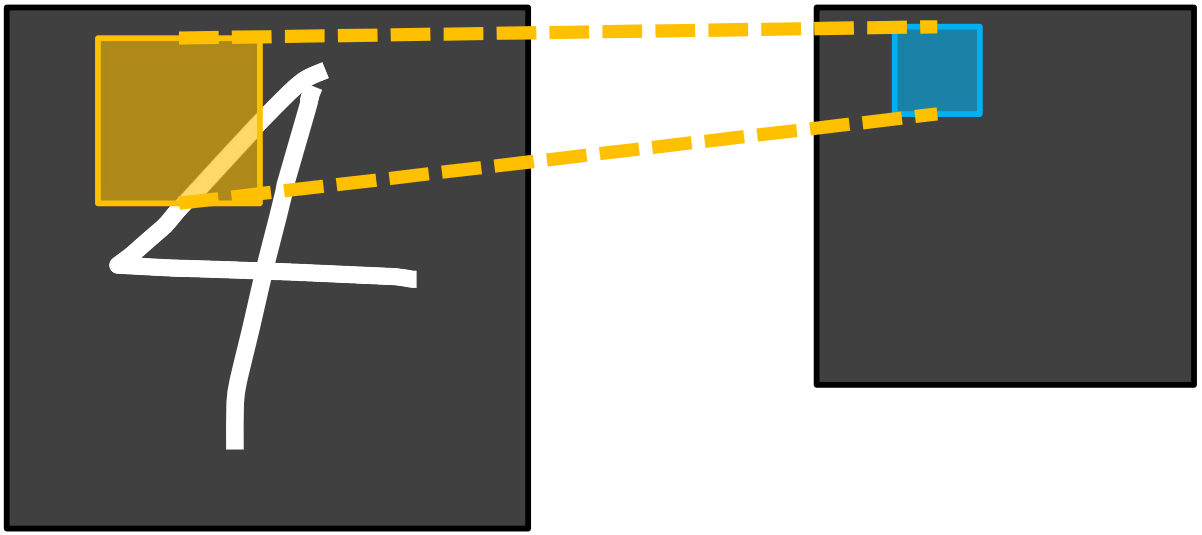
+ Dropout

Perceptrons (with bias)



Introduction

Convolutional Neural Network (CNNs)



"Convolutions" : Convolution of input with fixed size array(**kernels**)

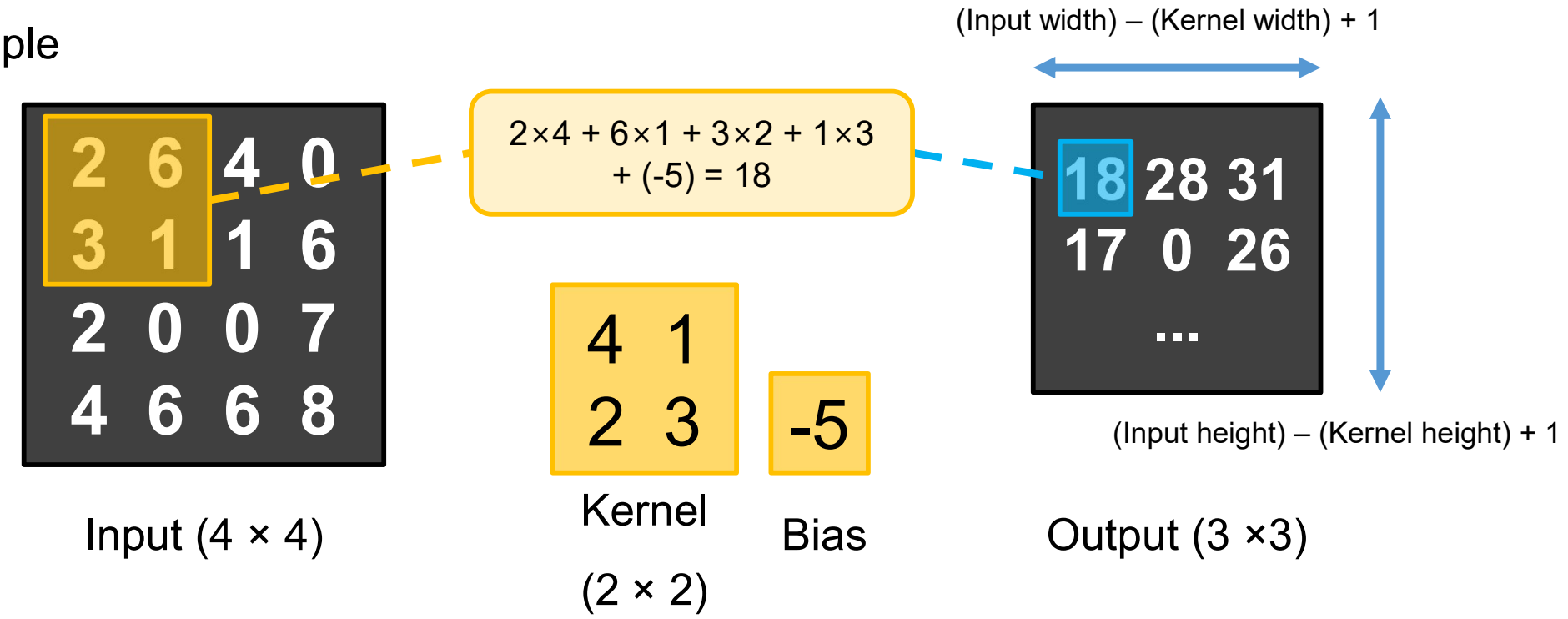
- Reduce dimensions
- Extract locality features
- ...

Convolutional Neural Network (CNNs)

"Kernels"

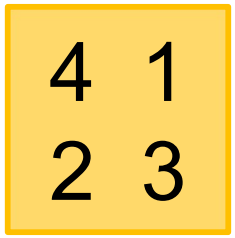
- Fixed size of weights to be **trained**
- Sliding windows
- Can be multiple

Example



Convolutional Neural Network (CNNs)

Number of training parameters



(Kernel size) × (Input Channel) × (Output Channel)

Bias : 1 per output

1 × (Output Channel)

Total Number of Trainable Weights

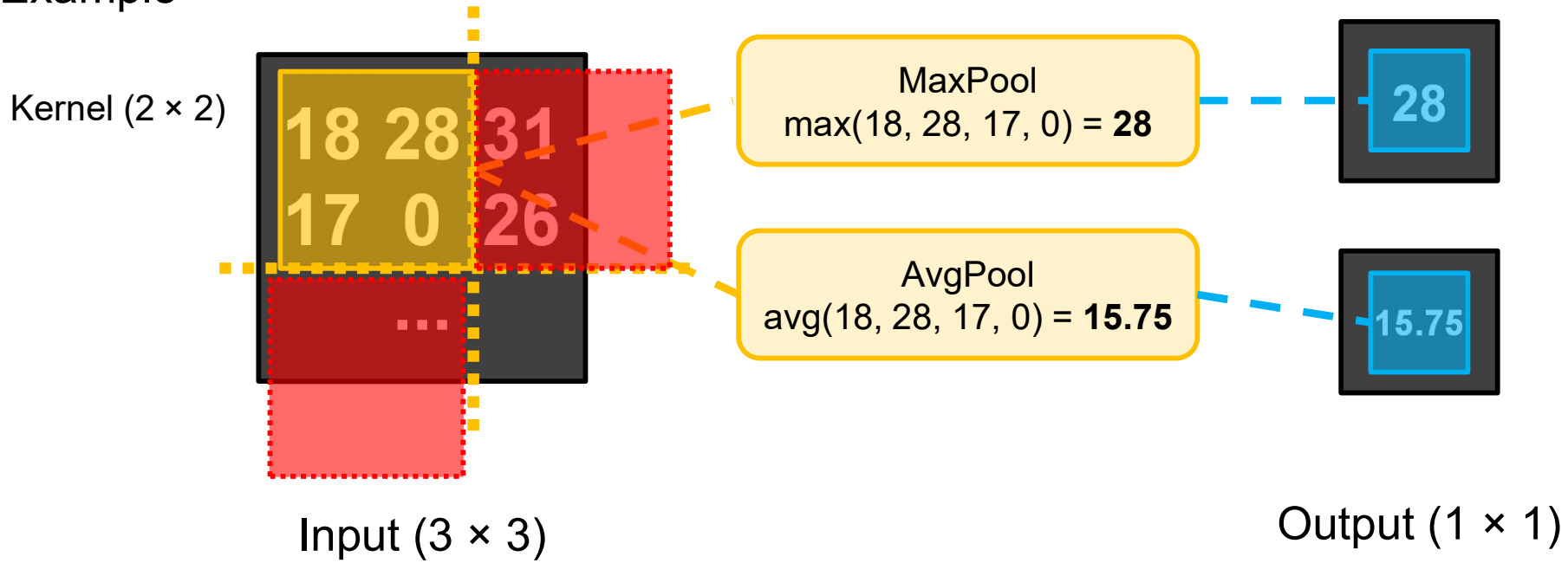
- Bias = True *(by default)* : (kernel size + 1) * input * output
- Bias = False : (kernel) * input * output

Convolutional Neural Network (CNNs)

"Pooling"

- Reduction in size
- Generalization ...
- Average Pooling, **Max Pooling**, ...

Example



MNIST classification with CNN

Input = 5 x 1 x 28 x 28

CNN

Probability

Output = 5 x 10

Example

MLP

z	0	1	2	3	4	**5**	6	7	8	9
P(z)	0.02	0.01	0.01	0.02	0.01	**0.85**	0.03	0.01	0.02	0.02

- Softmax

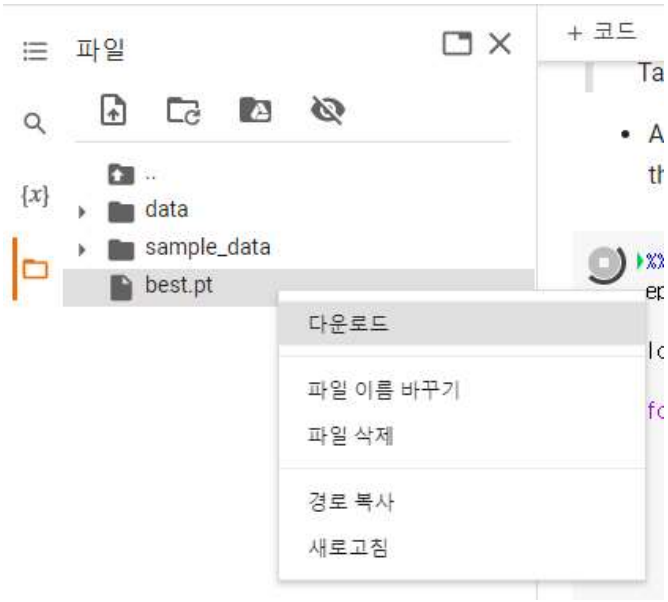
$\mathbf{z} = [z_1, ..., z_n]$

$\text{softmax}(\mathbf{z}) = [e^{z1}/\Sigma(e^{zk}), e^{z2}/\Sigma(e^{zk}), ..., e^{zn}/\Sigma(e^{zk})]$

Checkpointing

- After training a model, we would like to **save** our trained results as a file.
- To continue training, distribute pre-trained model, ...
- We should save our model (weights; nn.Module.state_dict()), as well as optimizer parameters and epochs(to continue training).

```
def save_model(model_, optimizer_, epoch_, save_as="best.pt"):
    best_model = {'model':model_.state_dict(),
                  'optimizer':optimizer_.state_dict(),
                  'epoch':epoch_}

    torch.save(best_model, save_as)

# Usage : save_model(model, optimizer, epoch)
```



➢ **Save our model as “best.pt”**

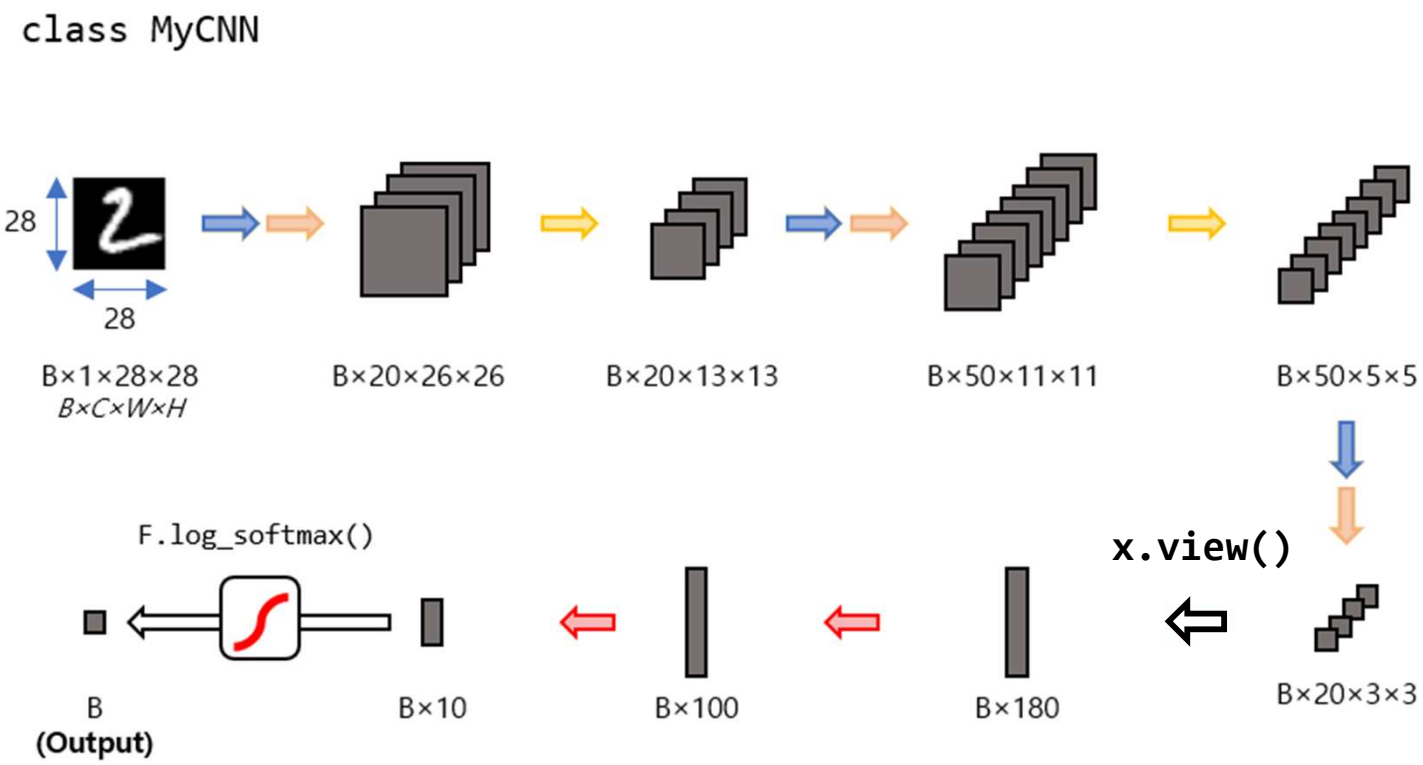
Practice

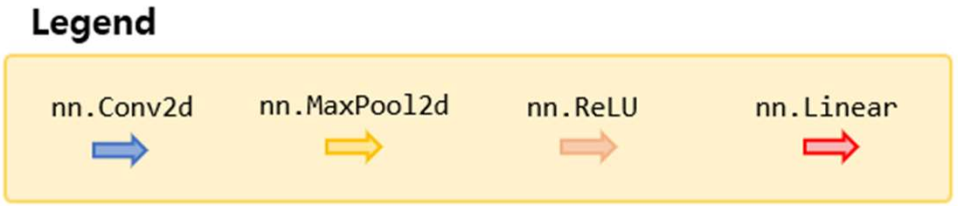
Today, you will be asked to

- Task 1. Implement custom CNN class
- Task 2. Implement checkpointing

Check out colab notebook for details

Implement MyCNN class





Implement Checkpointing

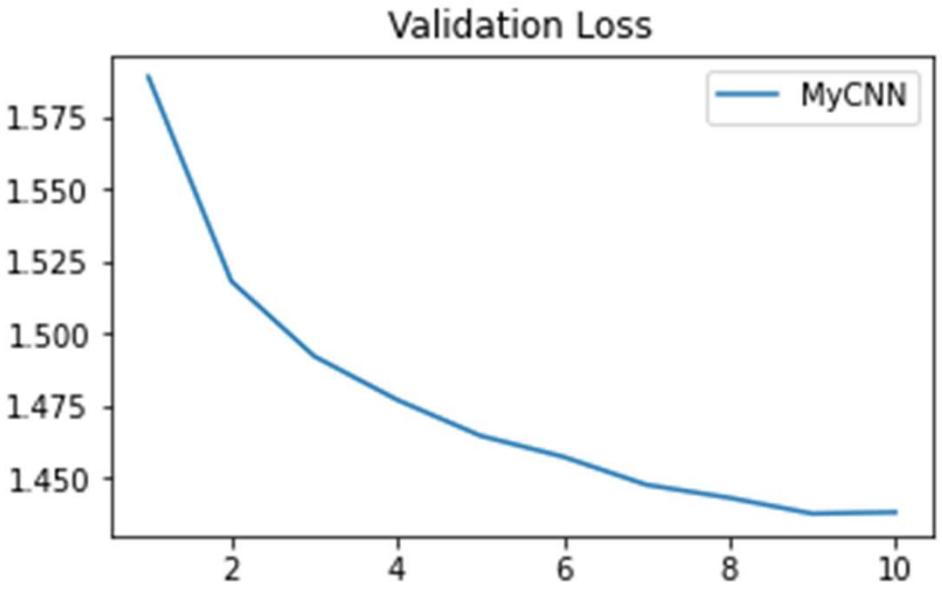
Training loop

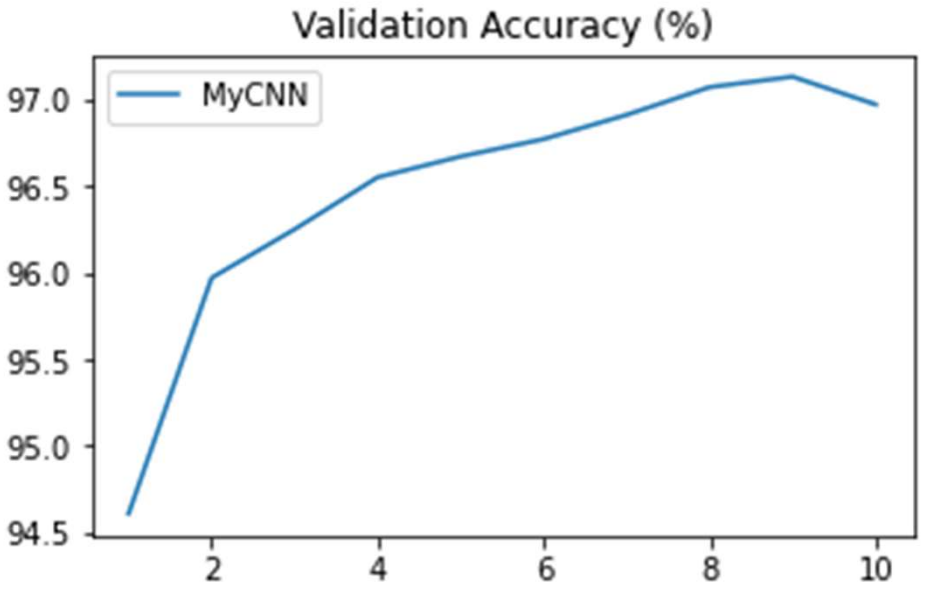
```
%%time
epochs = 10

lossv_CNN, accv_CNN = [], []

for epoch in range(1, epochs + 1):
    train(model, optimizer, epoch)
    validate(lossv_CNN, accv_CNN)

    ## Task 2. Checkpointing ##
    # Save your model only if recent validation loss is minimum

    # Change this line
    save_condition = [redacted]

    if save_condition:
        save_model(model, optimizer, epoch)
```

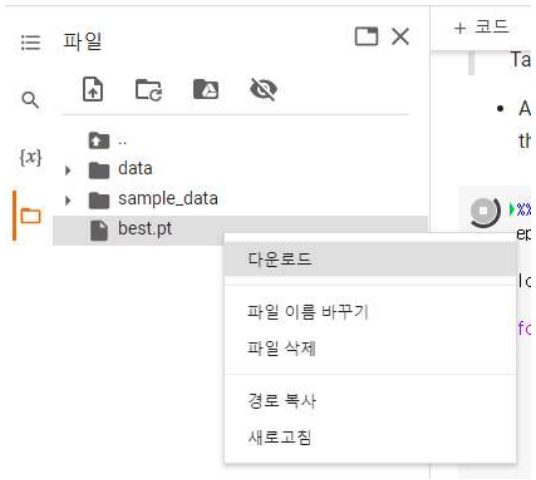
Save only when validation loss is minimum

Quiz #1

➢ Today's quiz #1. Submit your outputs (must run by yourself)







<u>Submissions</u>

- Validation Loss Graph
- Validation Accuracy Graph
- **'best.pt'** : will be cross-checked with your graphs

Practice Session / Q & A

Task 1. Implement custom CNN class
Task 2. Implement Checkpointing

- Quiz will begin at 5:20