AI VIETNAM
All-in-One Course

Convolutional Neural Network (Draft)

Quang-Vinh Dinh
Ph.D. in Computer Science

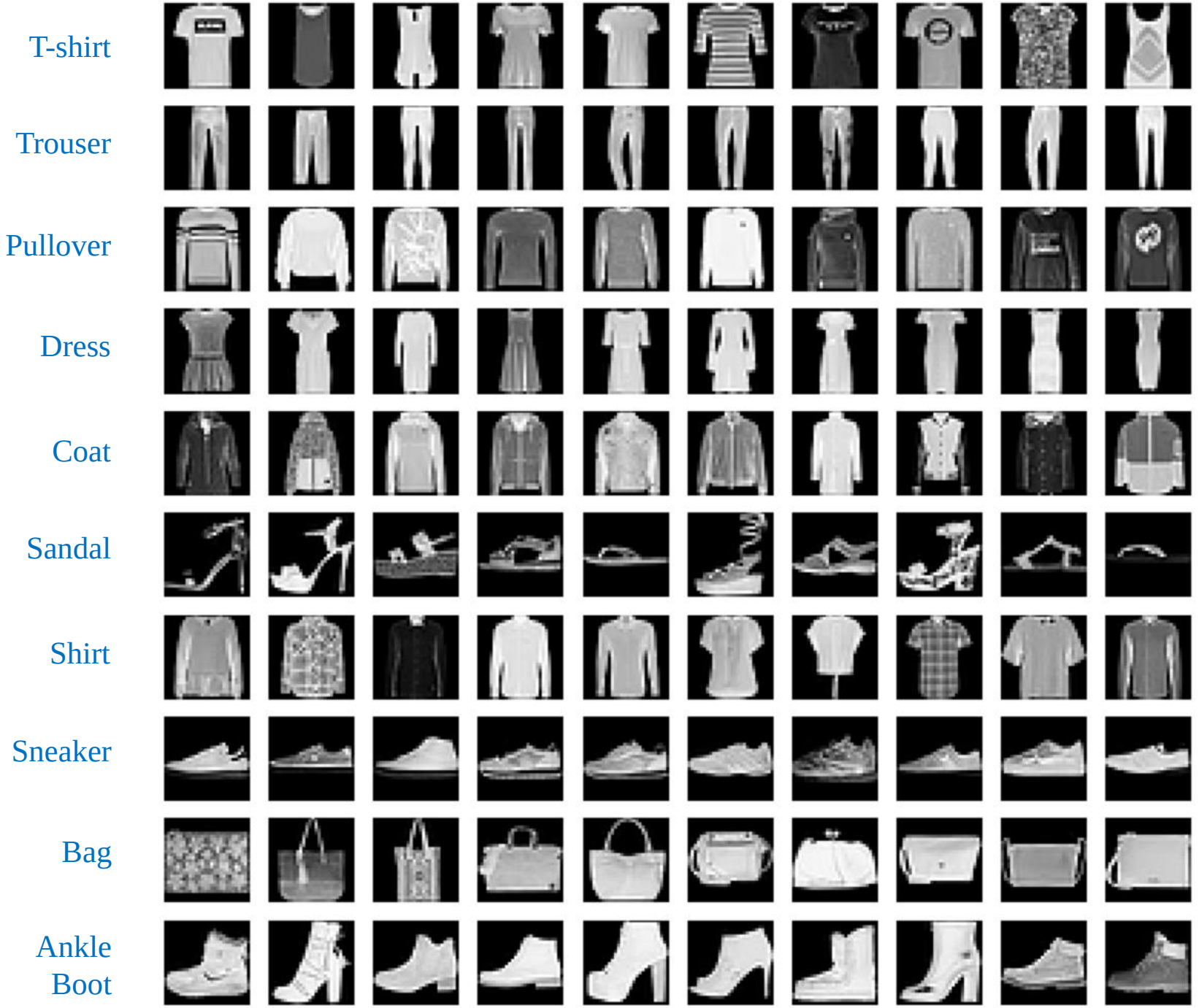
Year 2023

Outline

- MLP Warm-up
- From MLP to CNN
- Feature Map Down-sampling
- Examples

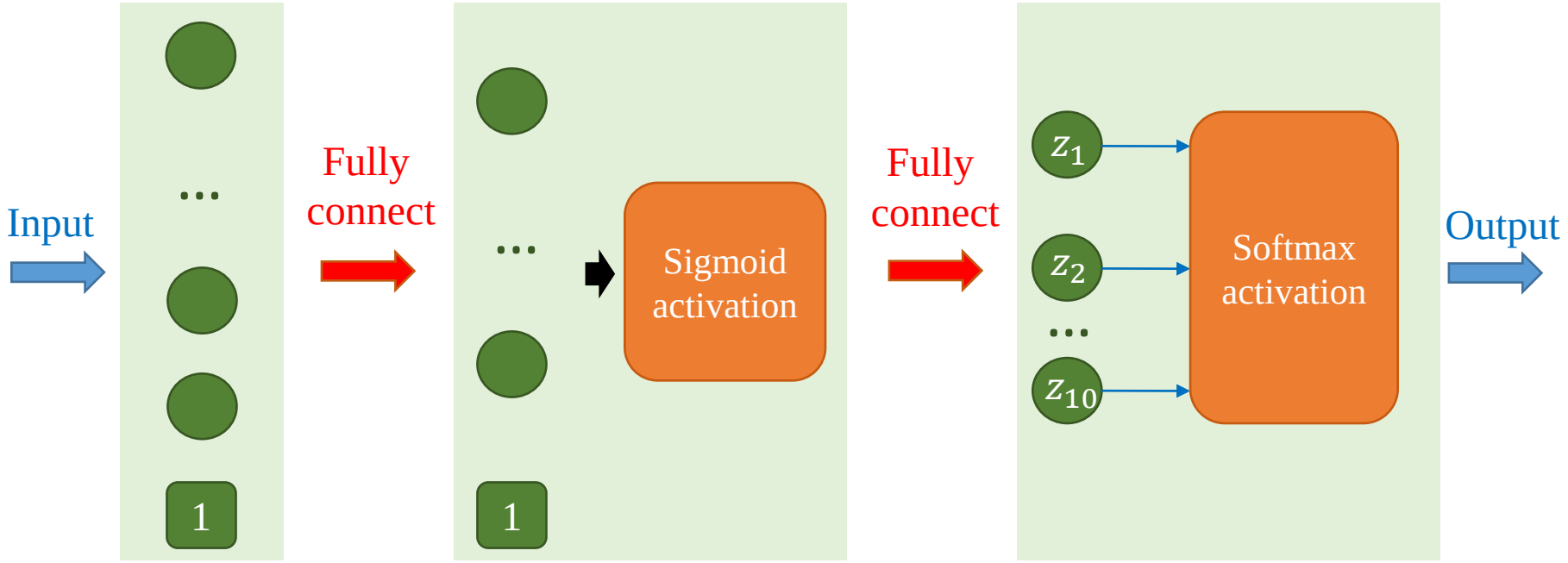
Fashion-MNIST dataset

- Grayscale images
- Resolution=28x28
- Training set: 60000 samples
- Testing set: 10000 samples



MLP for Fashion-MNIST

❖ Sigmoid, Xavier and Adam



MLP for Fashion-MNIST

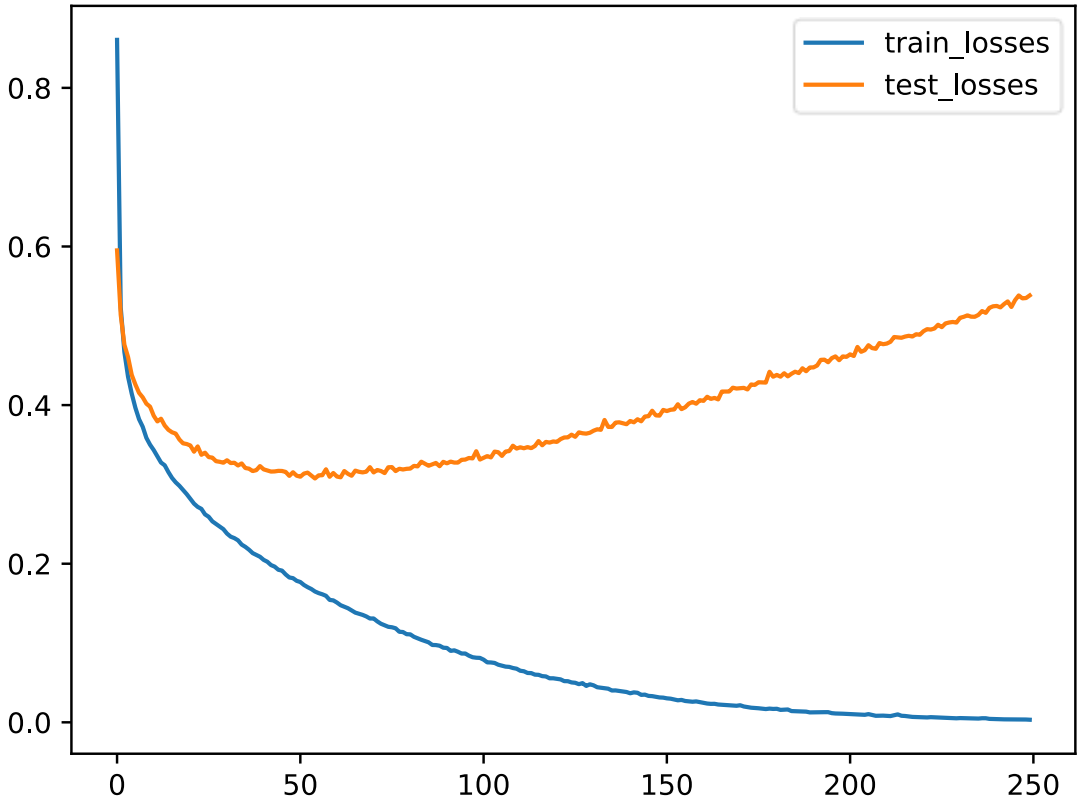
❖ Sigmoid, Xavier and Adam

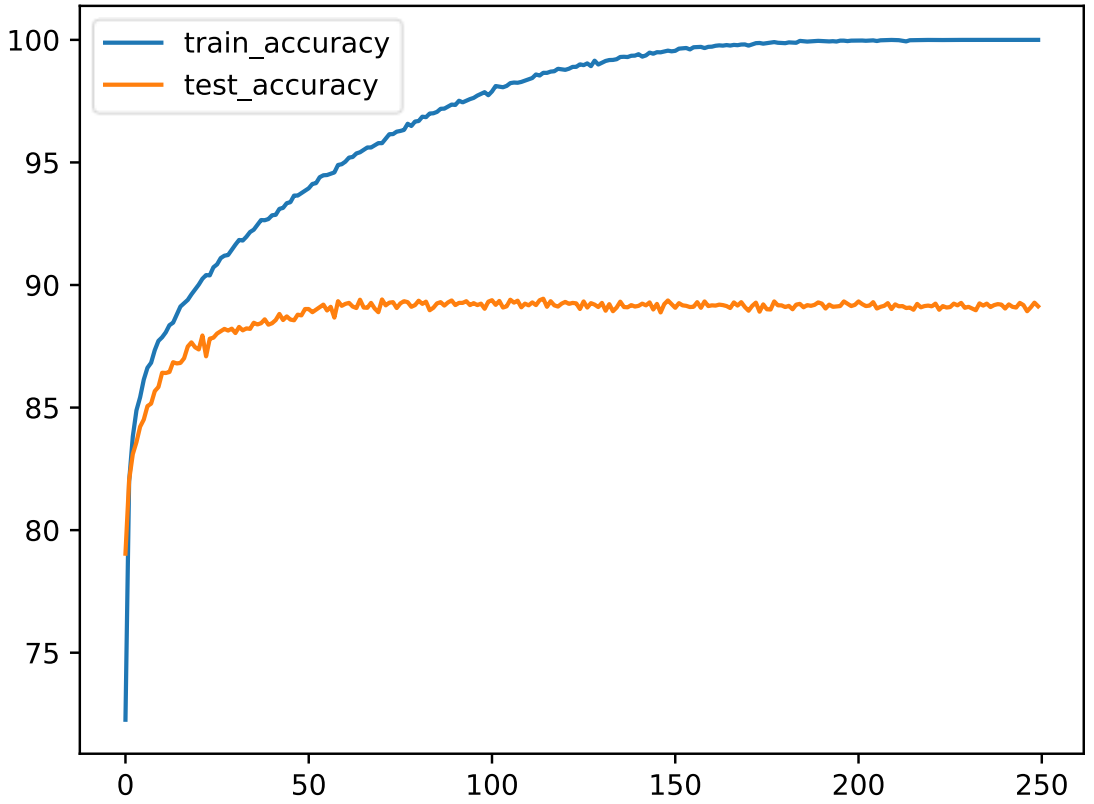
```
# model
model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(784, 256),
    nn.Sigmoid(),
    nn.Linear(256, 10)
)

# Xavier Glorot initialization
for layer in model:
    if isinstance(layer, nn.Linear):
        init.xavier_uniform_(layer.weight)
        if layer.bias is not None:
            layer.bias.data.fill_(0)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
                       lr=0.001)
```

```
# Load CFashionMNIST dataset
transform = Compose([transforms.ToTensor(),
                     transforms.Normalize((0.5,),
                                          (0.5,))])

trainset = FashionMNIST(root='data',
                        train=True,
                        download=True,
                        transform=transform)
trainloader = DataLoader(trainset,
                         batch_size=1024,
                         num_workers=10,
                         shuffle=True,
                         drop_last=True)

testset = FashionMNIST(root='data',
                       train=False,
                       download=True,
                       transform=transform)
testloader = DataLoader(testset,
                        batch_size=1024,
                        num_workers=10,
                        shuffle=False)
```

MLP for Fashion-MNIST

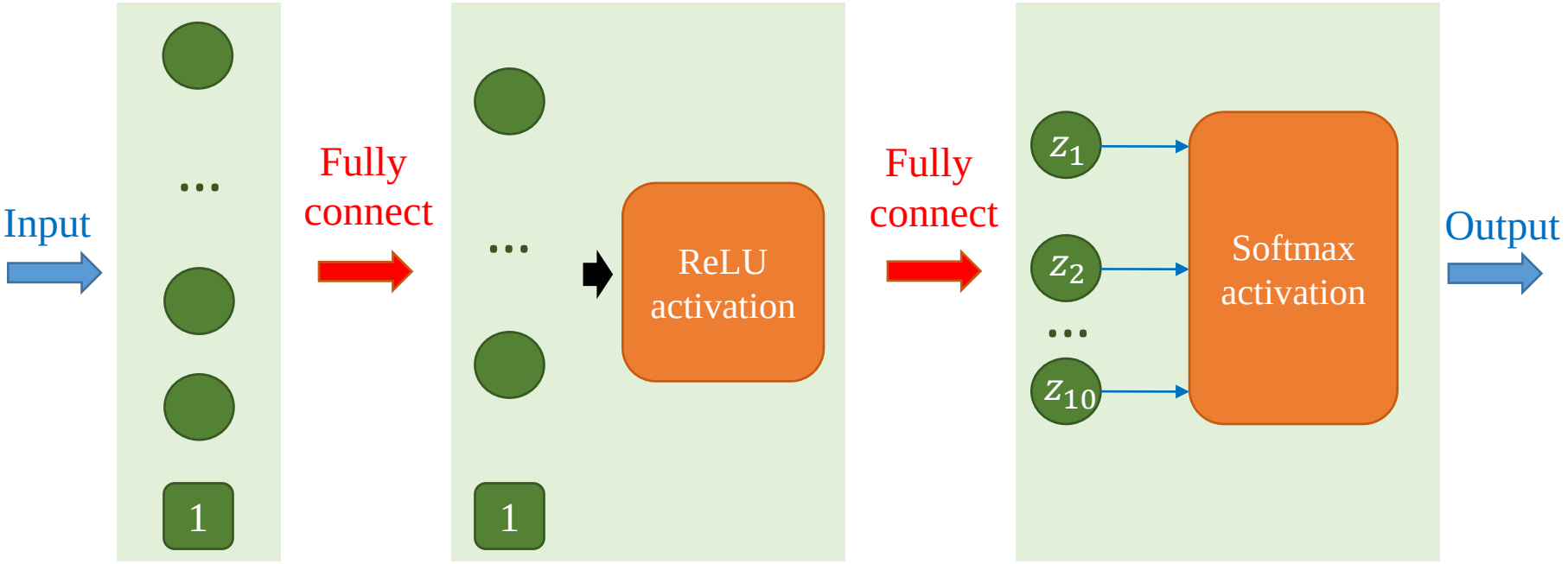
❖ Sigmoid, Xavier and Adam





MLP for Fashion-MNIST

❖ ReLU, He and Adam



MLP for Fashion-MNIST

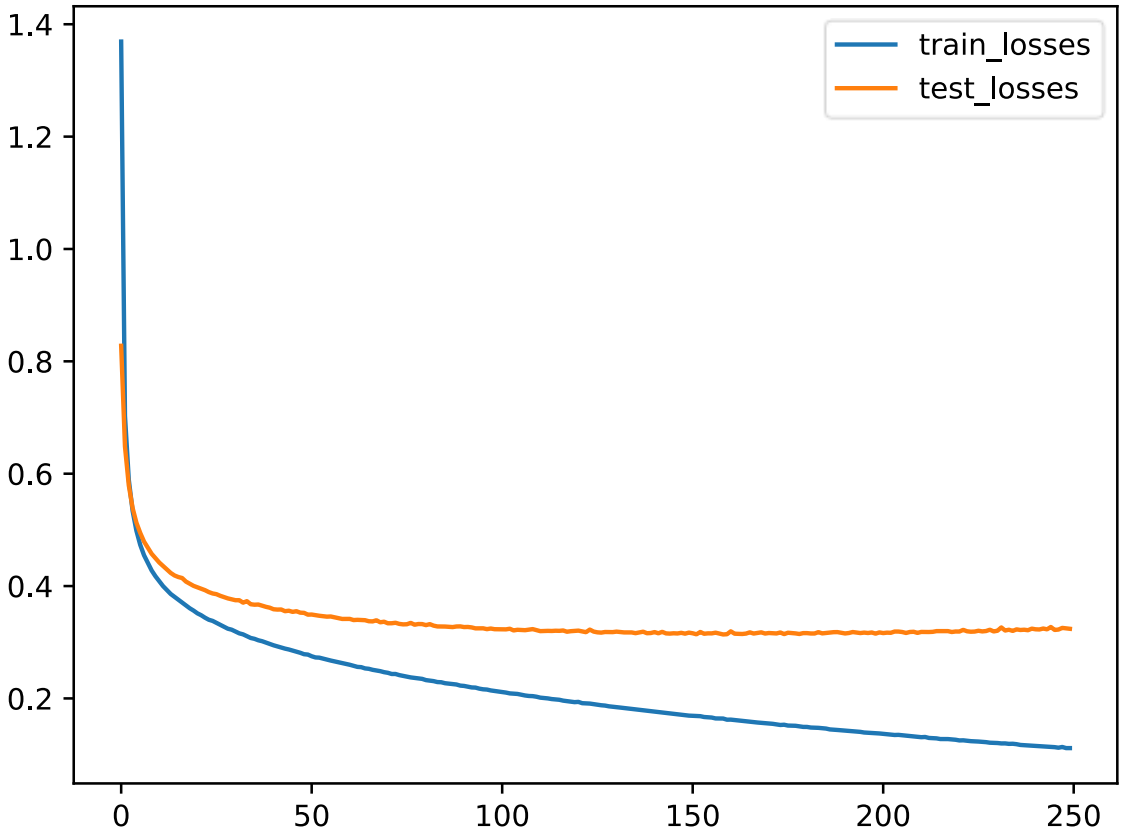
❖ ReLU, He and Adam

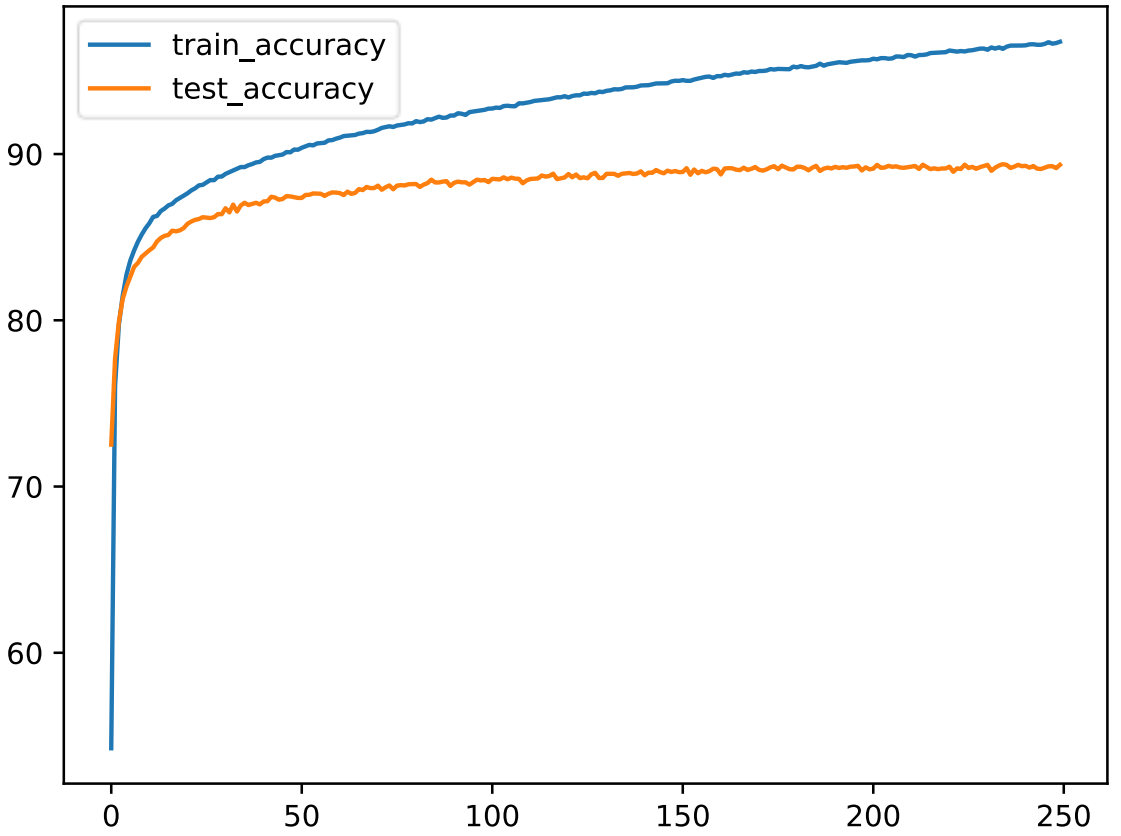
```python
# model
model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(784, 256),
    nn.ReLU(),
    nn.Linear(256, 10)
)

# Initialize the weights
for layer in model:
    if isinstance(layer, nn.Linear):
        init.kaiming_uniform_(layer.weight,
                              nonlinearity='relu')
        if layer.bias is not None:
            layer.bias.data.fill_(0)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
                       lr=0.001)
```

```python
# Load CFashionMNIST dataset
transform = Compose([transforms.ToTensor(),
                     transforms.Normalize((0.5,),
                                          (0.5,))])

trainset = FashionMNIST(root='data',
                        train=True,
                        download=True,
                        transform=transform)
trainloader = DataLoader(trainset,
                         batch_size=1024,
                         num_workers=10,
                         shuffle=True,
                         drop_last=True)

testset = FashionMNIST(root='data',
                       train=False,
                       download=True,
                       transform=transform)
testloader = DataLoader(testset,
                        batch_size=1024,
                        num_workers=10,
                        shuffle=False)
```

MLP for Fashion-MNIST

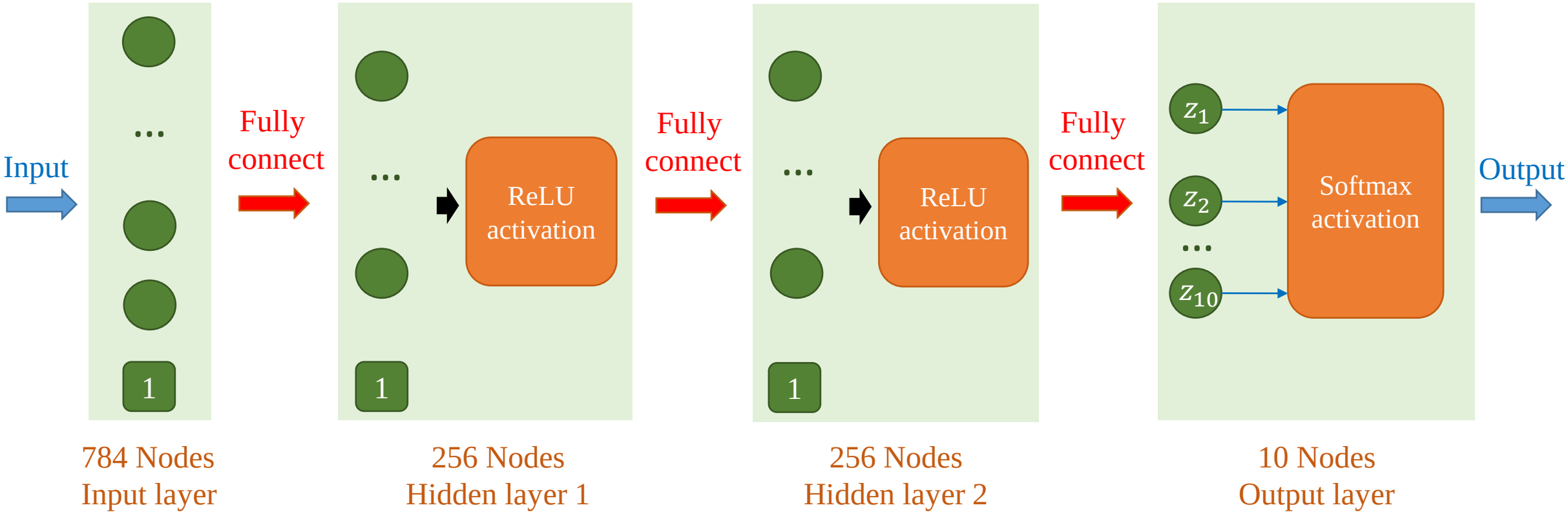
❖ ReLU, He and Adam





MLP for Fashion-MNIST

❖ ReLU, He and Adam



MLP for Fashion-MNIST

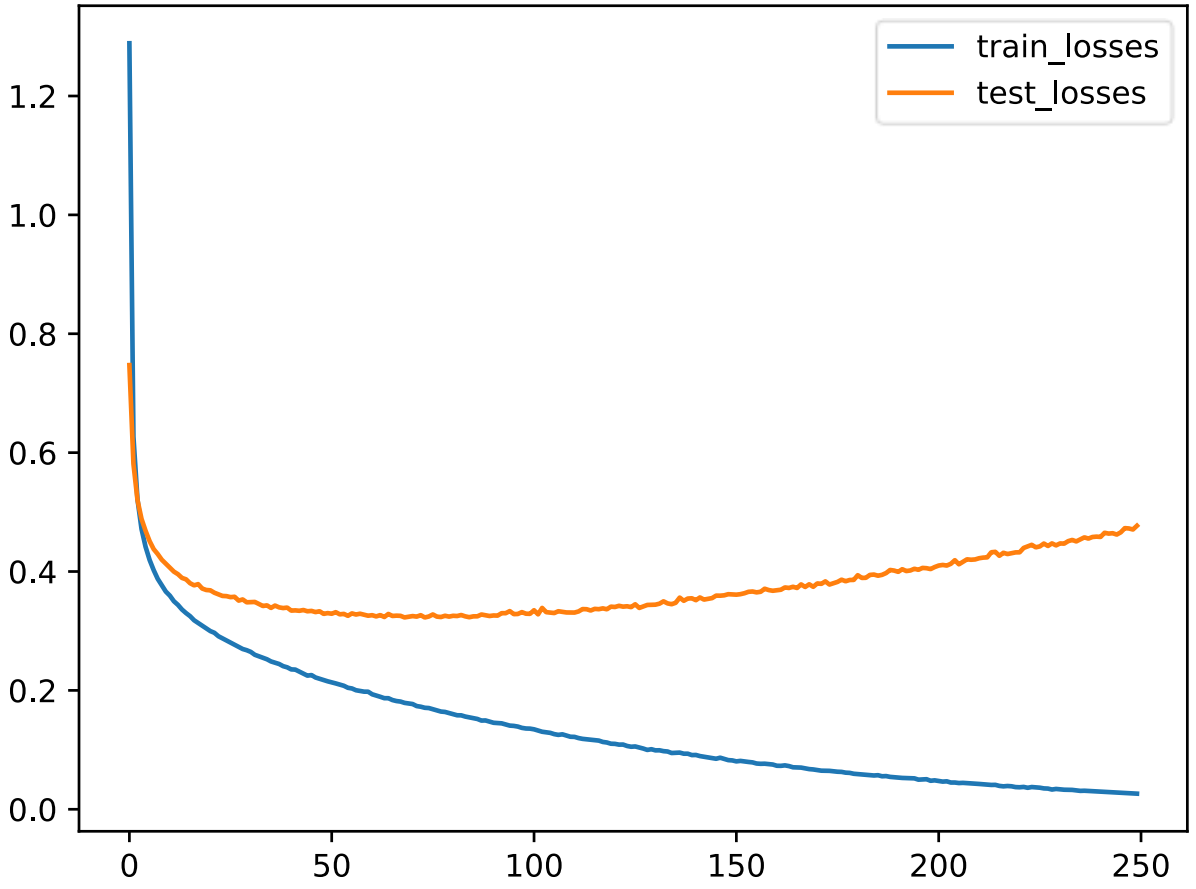
❖ ReLU, He and Adam

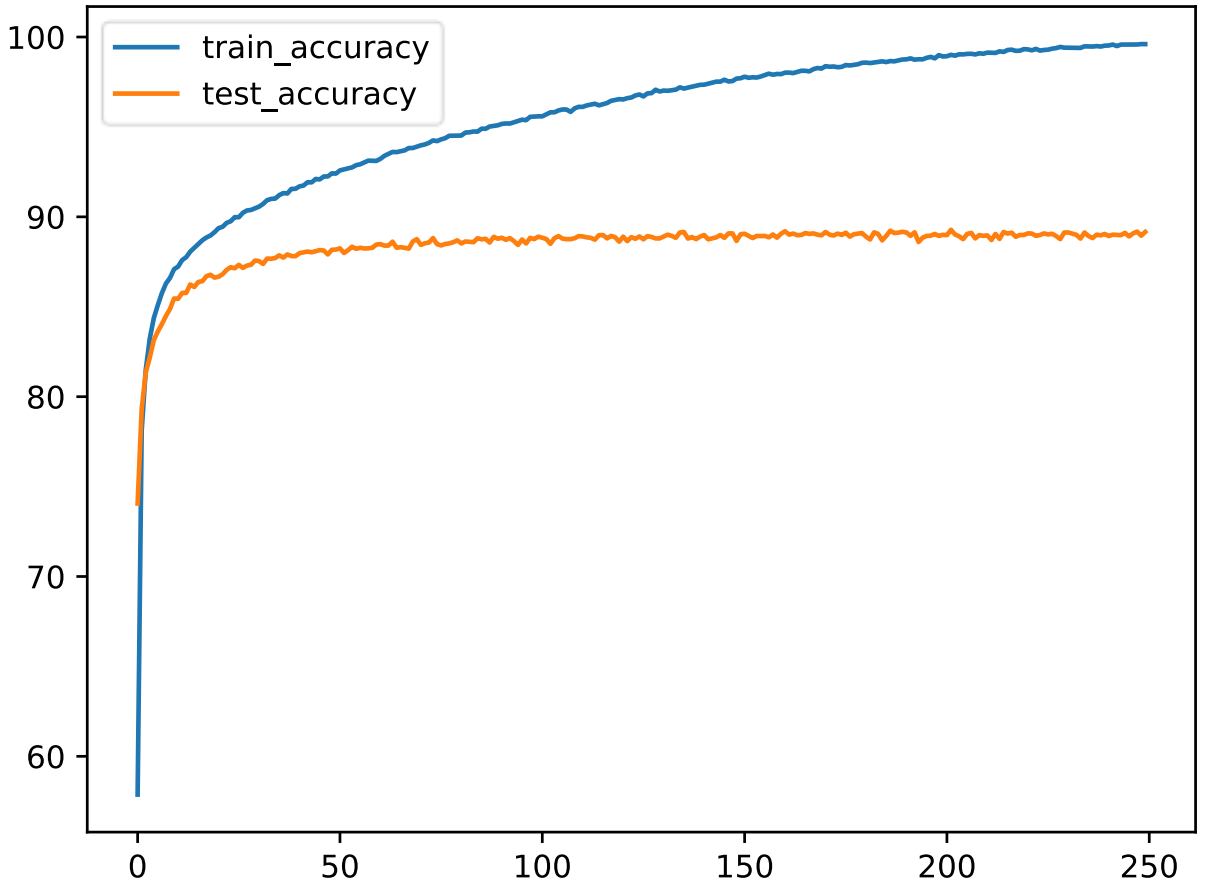
```
# model
model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(784, 256),
    nn.ReLU(),
    nn.Linear(256, 256),
    nn.ReLU(),
    nn.Linear(256, 10)
)

# Initialize the weights
for layer in model:
    if isinstance(layer, nn.Linear):
        init.kaiming_uniform_(layer.weight,
                              nonlinearity='relu')
        if layer.bias is not None:
            layer.bias.data.fill_(0)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
                       lr=0.001)
```

```
# Load CFashionMNIST dataset
transform = Compose([transforms.ToTensor(),
                     transforms.Normalize((0.5,),
                                          (0.5,))])

trainset = FashionMNIST(root='data',
                        train=True,
                        download=True,
                        transform=transform)
trainloader = DataLoader(trainset,
                         batch_size=1024,
                         num_workers=10,
                         shuffle=True,
                         drop_last=True)

testset = FashionMNIST(root='data',
                       train=False,
                       download=True,
                       transform=transform)
testloader = DataLoader(testset,
                        batch_size=1024,
                        num_workers=10,
                        shuffle=False)
```

MLP for Fashion-MNIST

❖ ReLU, He and Adam



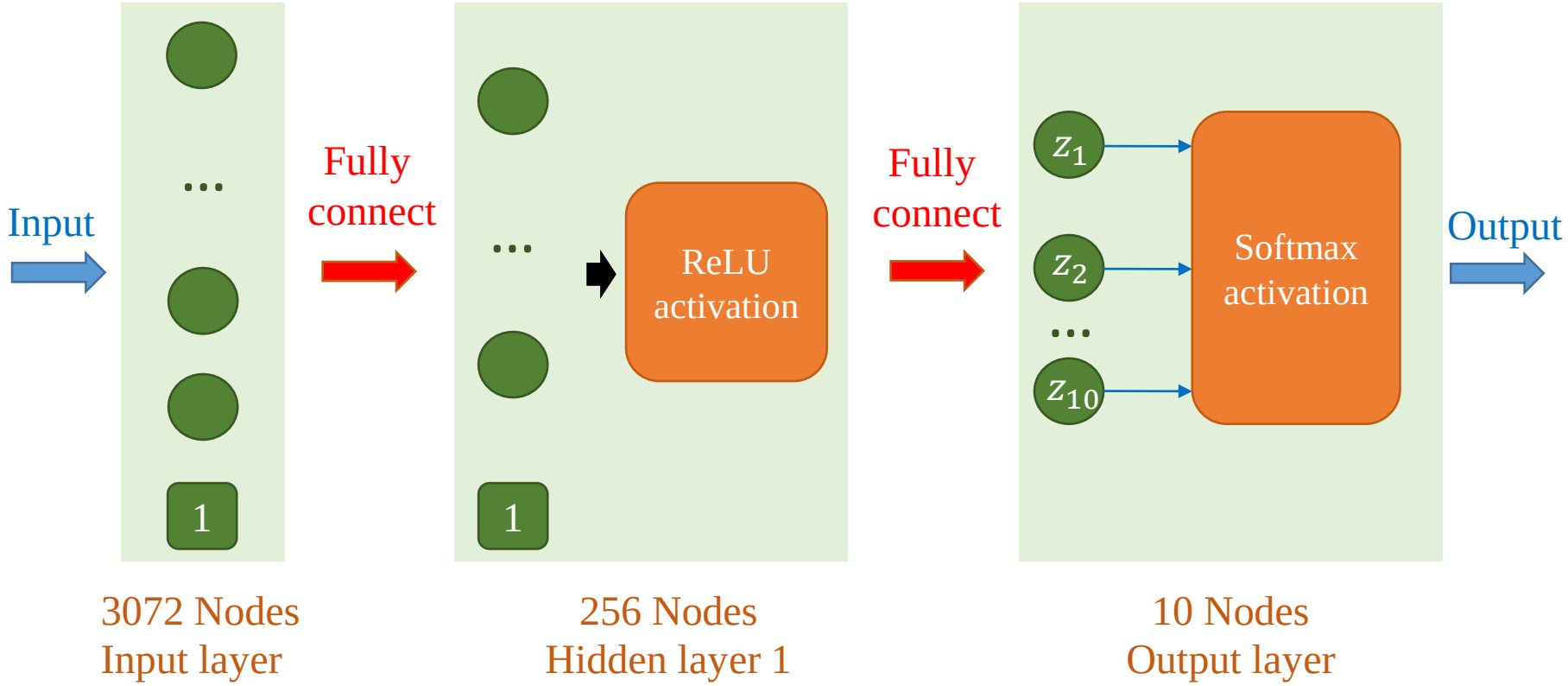


Cifar-10 dataset

- Color images
- Resolution=32x32
- Training set: 50000 samples
- Testing set: 10000 samples

MLP for Cifar-10

❖ ReLU, He and Adam



MLP for Cifar-10

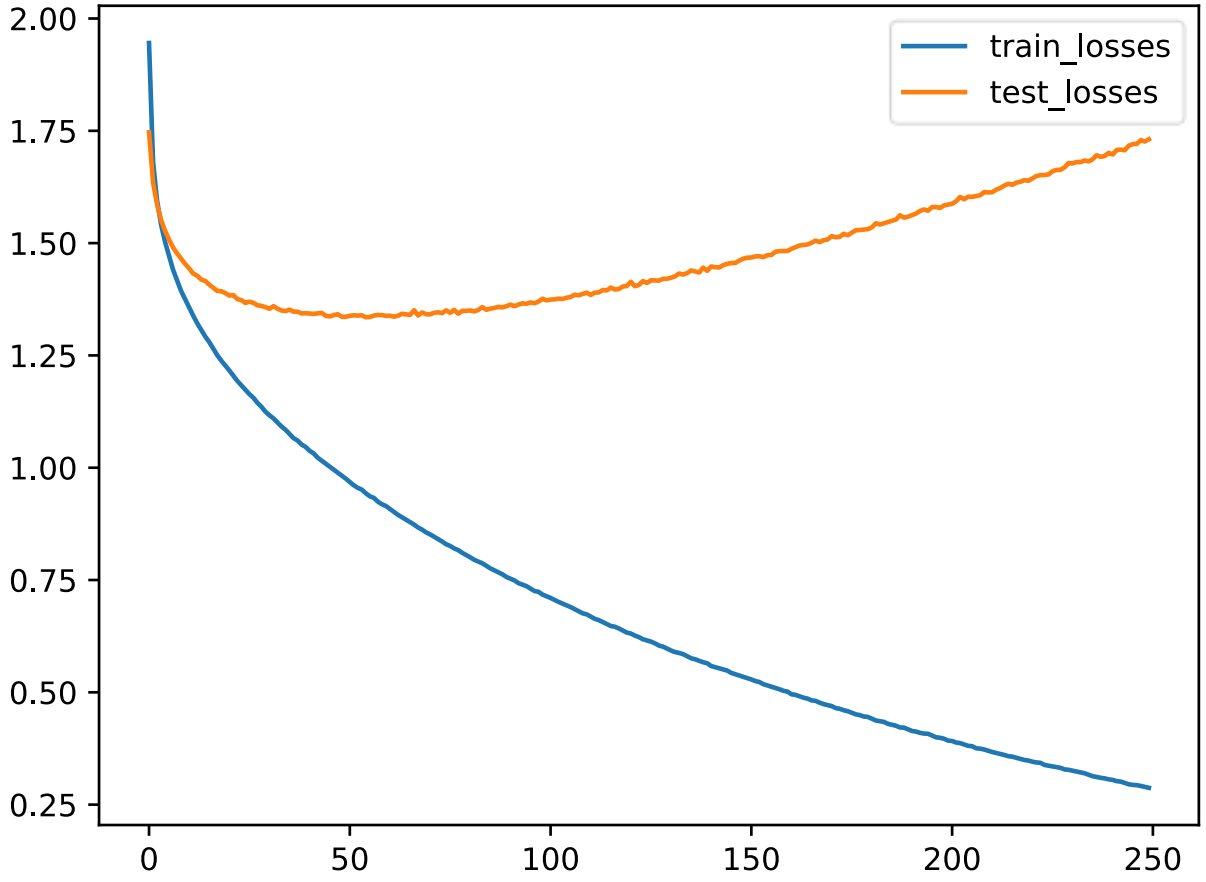
❖ ReLU, He and Adam

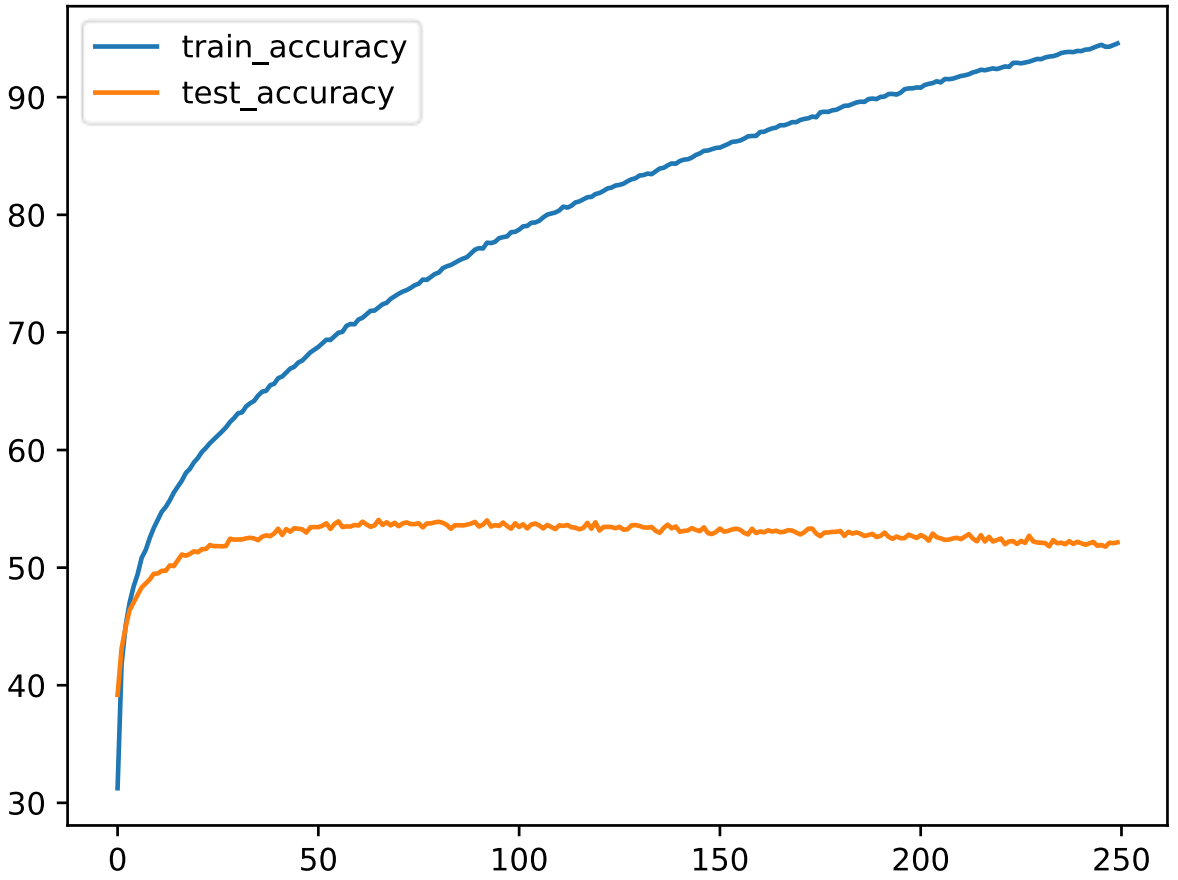
```
# model
model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(32*32*3, 256),
    nn.ReLU(),
    nn.Linear(256, 10)
)

# Initialize the weights
for layer in model:
    if isinstance(layer, nn.Linear):
        init.kaiming_uniform_(layer.weight,
                              nonlinearity='relu')
        if layer.bias is not None:
            layer.bias.data.fill_(0)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
                       lr=0.001)
```

```
# Load CIFAR10 dataset
transform = Compose([ToTensor(),
                     Normalize((0.5,0.5, 0.5),
                               (0.5,0.5, 0.5))])

trainset = CIFAR10(root='data',
                   train=True,
                   download=True,
                   transform=transform)
trainloader = DataLoader(trainset,
                         batch_size=1024,
                         num_workers=10,
                         shuffle=True,
                         drop_last=True)

testset = CIFAR10(root='data',
                  train=False,
                  download=True,
                  transform=transform)
testloader = DataLoader(testset,
                        batch_size=1024,
                        num_workers=10,
                        shuffle=False)
```

MLP for Cifar-10

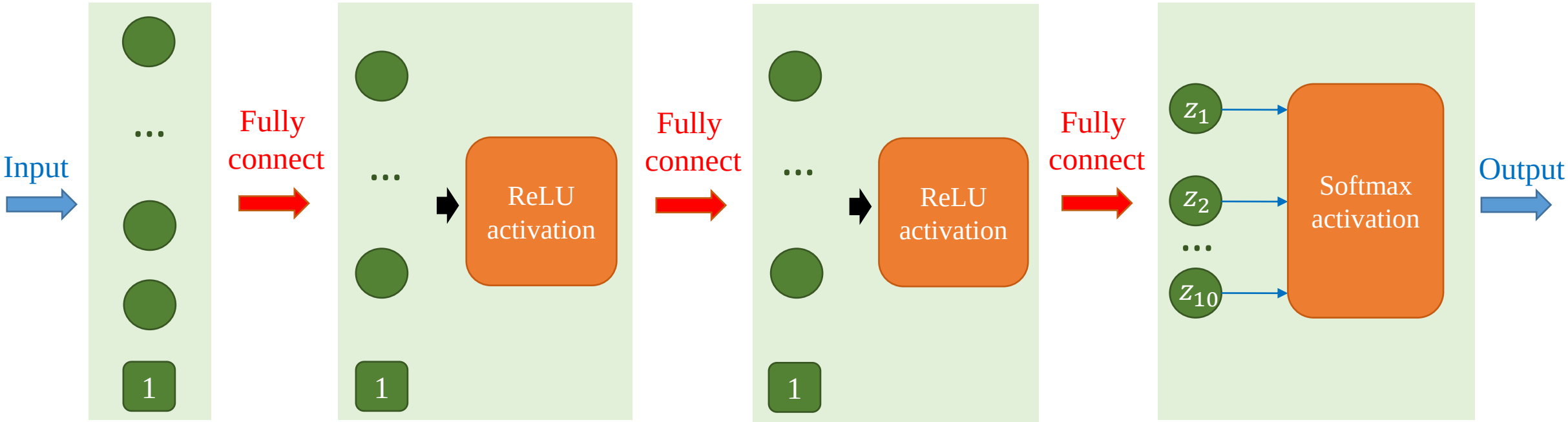
❖ ReLU, He and Adam





MLP for Cifar-10

❖ ReLU, He and Adam



MLP for Cifar-10

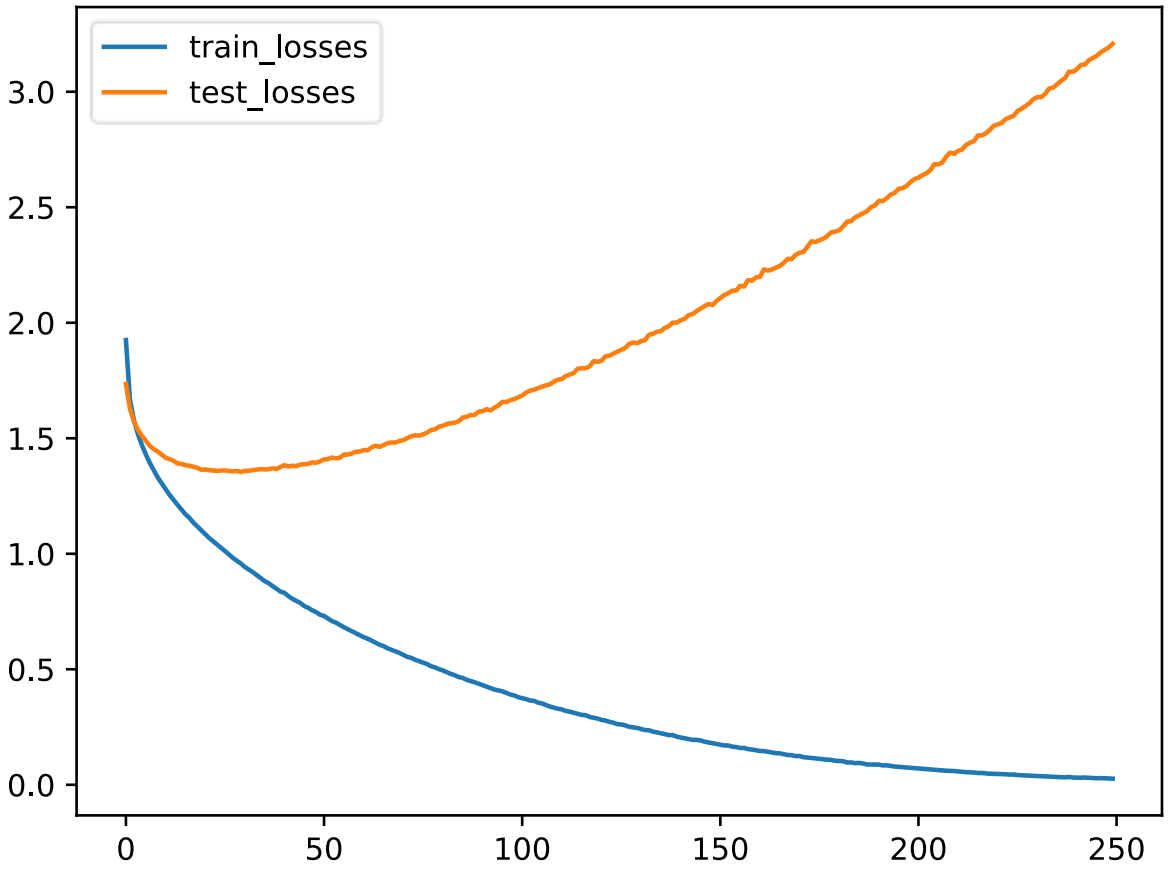
❖ ReLU, He and Adam

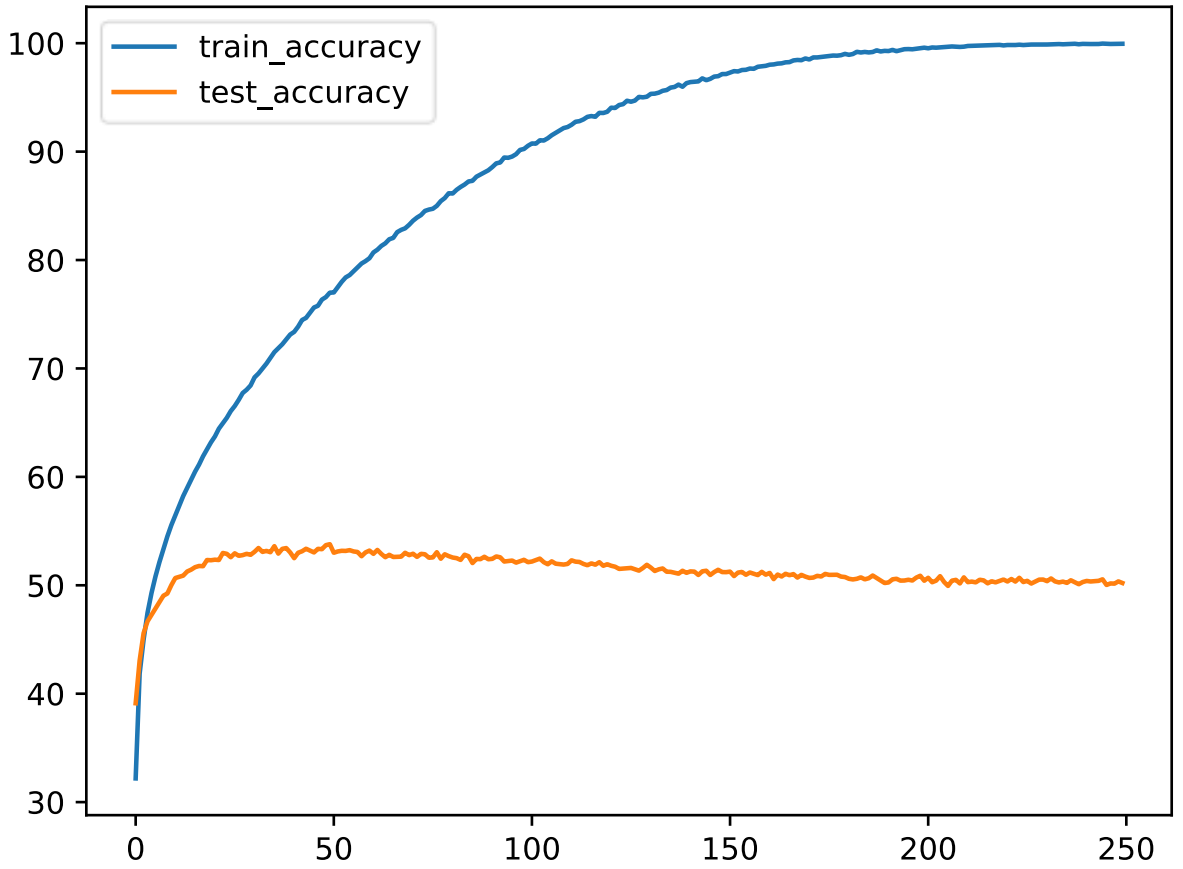
```
# model
model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(32*32*3, 256),
    nn.ReLU(),
    nn.Linear(256, 256),
    nn.ReLU(),
    nn.Linear(256, 10)
)

# Initialize the weights
for layer in model:
    if isinstance(layer, nn.Linear):
        init.kaiming_uniform_(layer.weight,
                              nonlinearity='relu')
        if layer.bias is not None:
            layer.bias.data.fill_(0)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
                       lr=0.001)
```

```
# Load CIFAR10 dataset
transform = Compose([ToTensor(),
                     Normalize((0.5,0.5, 0.5),
                               (0.5,0.5, 0.5))])

trainset = CIFAR10(root='data',
                   train=True,
                   download=True,
                   transform=transform)
trainloader = DataLoader(trainset,
                         batch_size=1024,
                         num_workers=10,
                         shuffle=True,
                         drop_last=True)

testset = CIFAR10(root='data',
                  train=False,
                  download=True,
                  transform=transform)
testloader = DataLoader(testset,
                        batch_size=1024,
                        num_workers=10,
                        shuffle=False)
```

MLP for Cifar-10

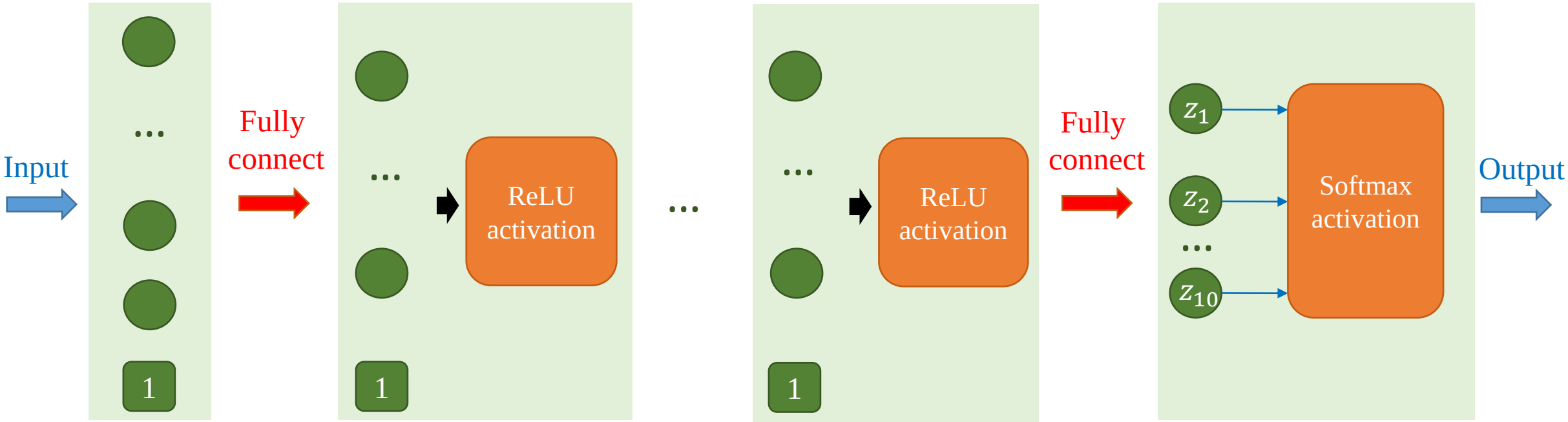
❖ ReLU, He and Adam





MLP for Cifar-10

❖ ReLU, He and Adam



MLP for Cifar-10

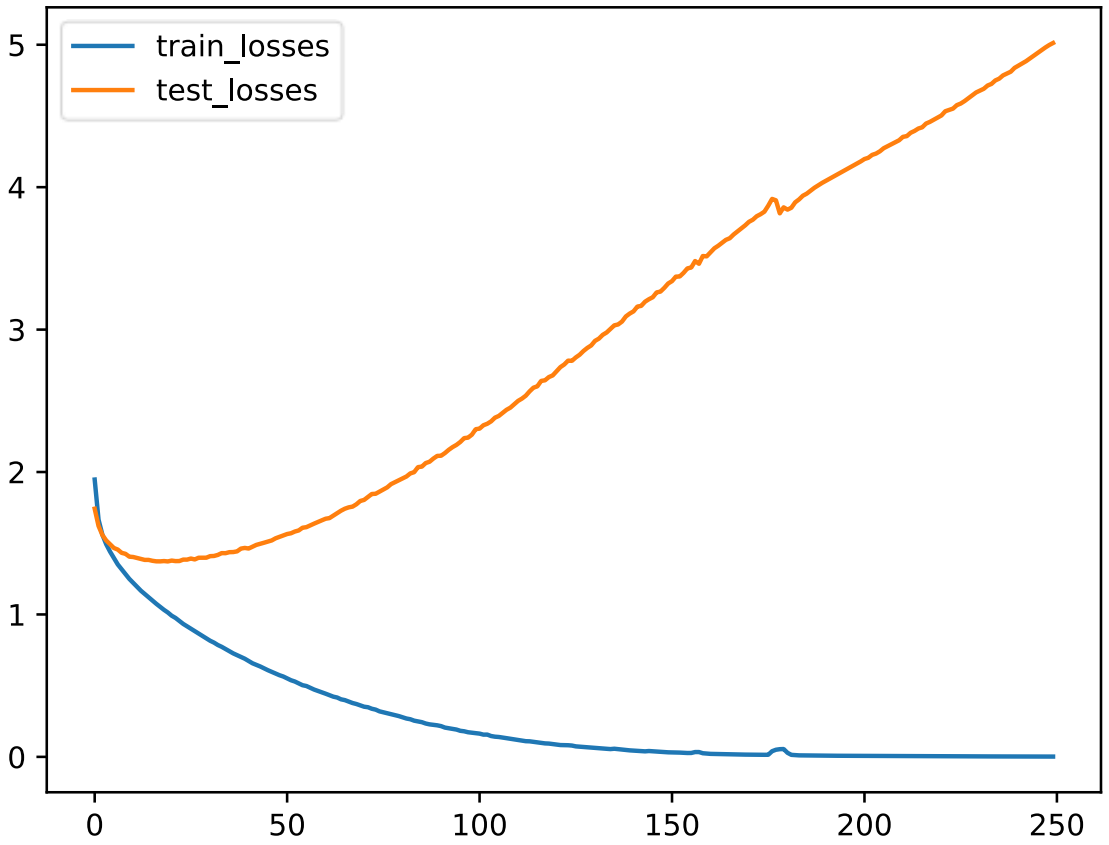
❖ ReLU, He and Adam

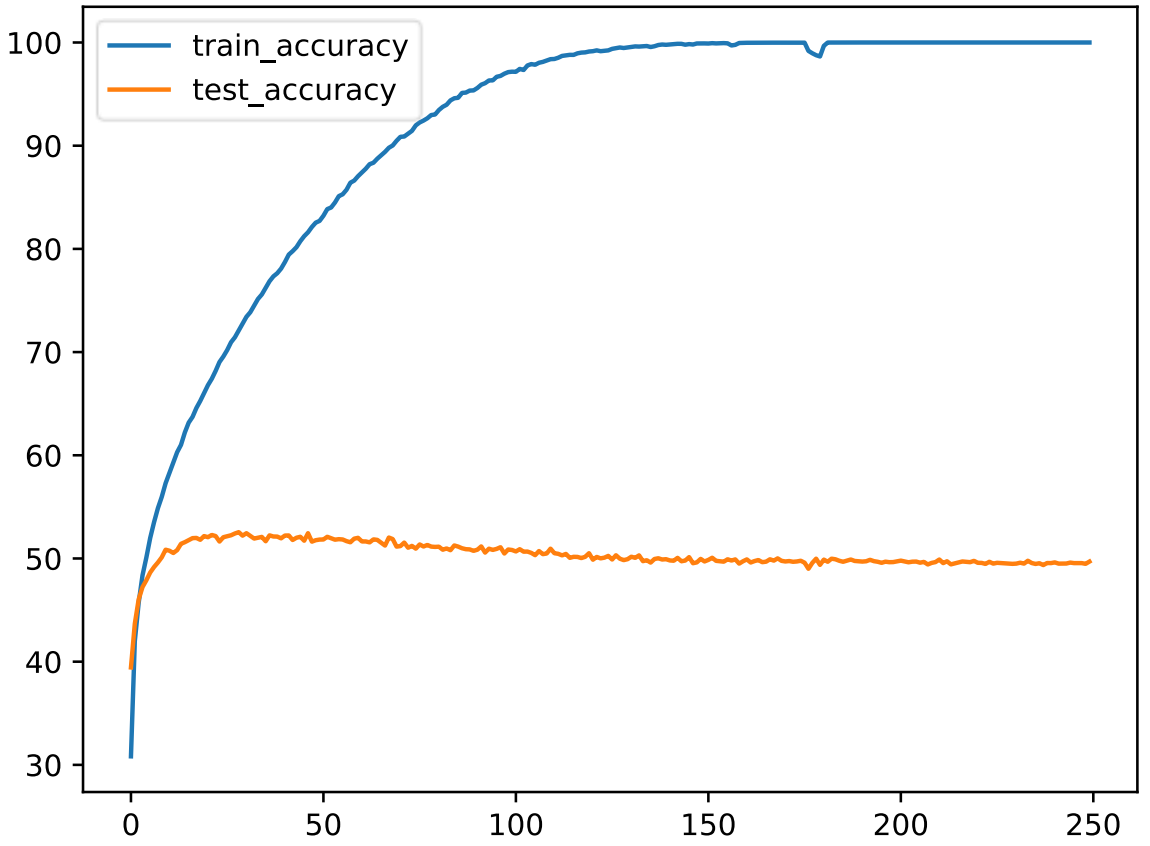
```
# model
model = nn.Sequential(
    nn.Flatten(), nn.Linear(32*32*3, 256),
    nn.ReLU(), nn.Linear(256, 256),
    nn.ReLU(), nn.Linear(256, 256),
    nn.ReLU(), nn.Linear(256, 10)
)

# Initialize the weights
for layer in model:
    if isinstance(layer, nn.Linear):
        init.kaiming_uniform_(layer.weight,
                              nonlinearity='relu')
        if layer.bias is not None:
            layer.bias.data.fill_(0)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
                       lr=0.001)
```

```
# Load CIFAR10 dataset
transform = Compose([ToTensor(),
                     Normalize((0.5,0.5, 0.5),
                               (0.5,0.5, 0.5))])

trainset = CIFAR10(root='data',
                   train=True,
                   download=True,
                   transform=transform)
trainloader = DataLoader(trainset,
                         batch_size=1024,
                         num_workers=10,
                         shuffle=True,
                         drop_last=True)

testset = CIFAR10(root='data',
                  train=False,
                  download=True,
                  transform=transform)
testloader = DataLoader(testset,
                        batch_size=1024,
                        num_workers=10,
                        shuffle=False)
```

MLP for Cifar-10

❖ ReLU, He and Adam



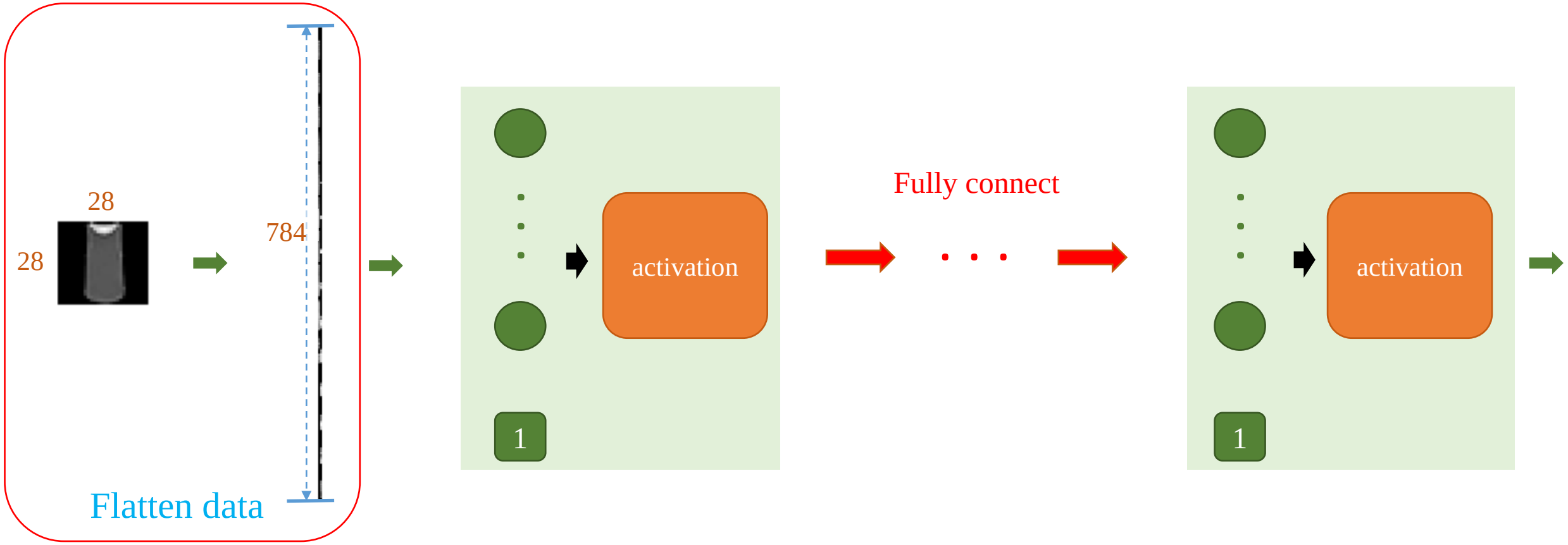


Outline

- MLP Warm-up
- From MLP to CNN
- Feature Map Down-sampling
- Examples

From MLP to CNN

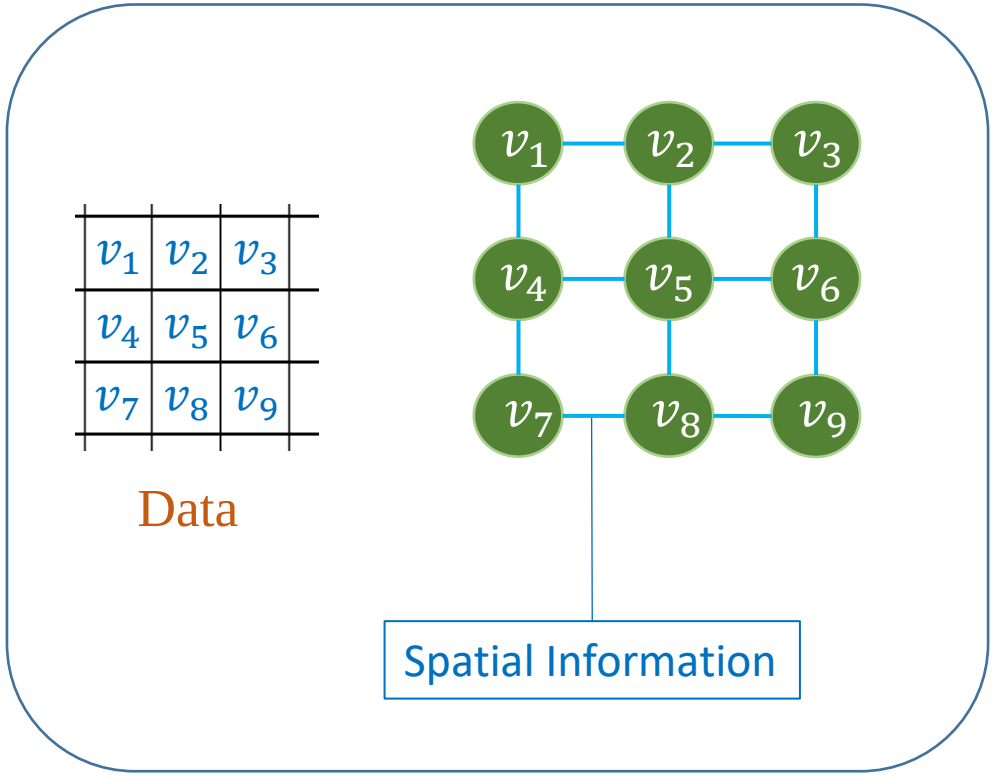
❖ Multi-layer Perceptron

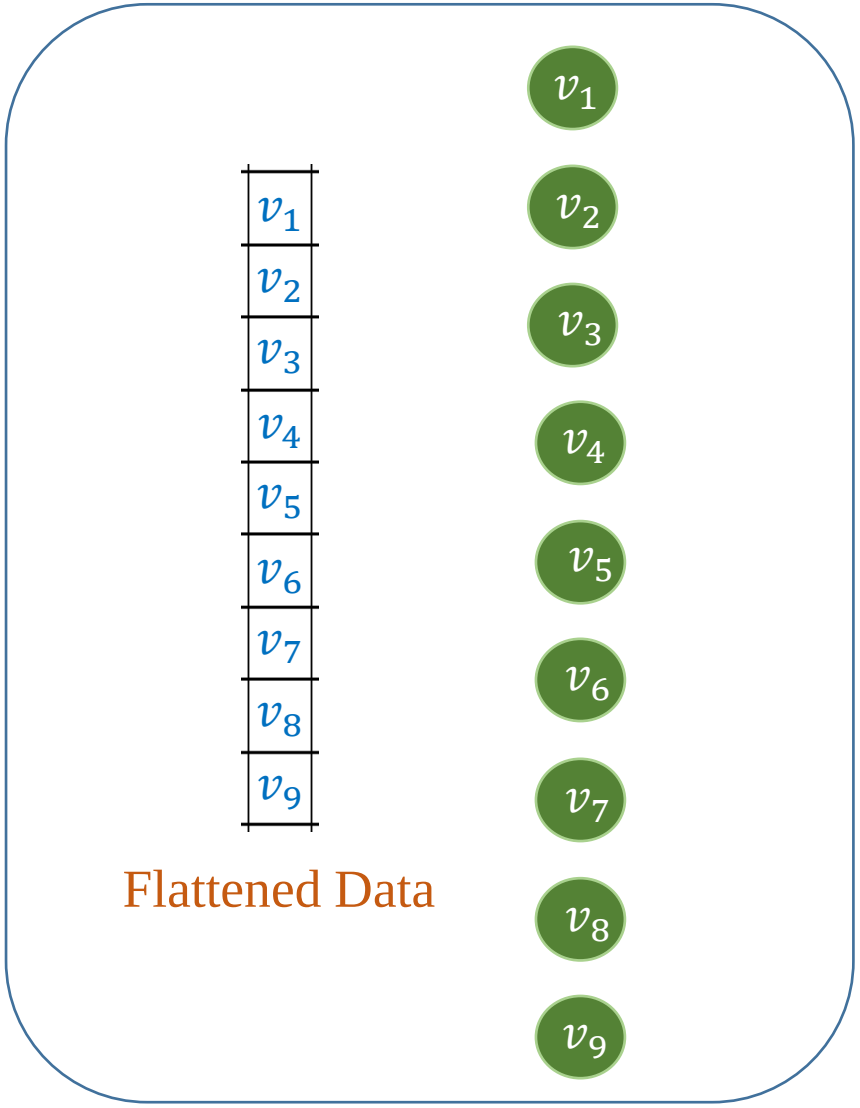


Problem: Remove spatial information of the data
Inefficiently have a large amount of parameters

From MLP to CNN

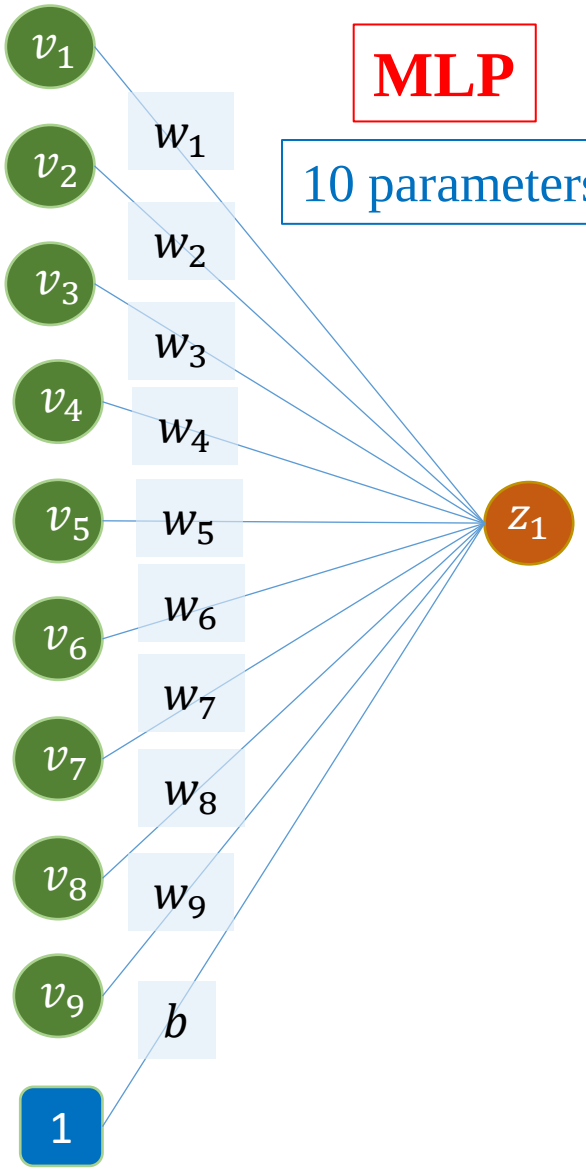
❖ Problem of flattening data





Remove spatial information of the data

From MLP to CNN



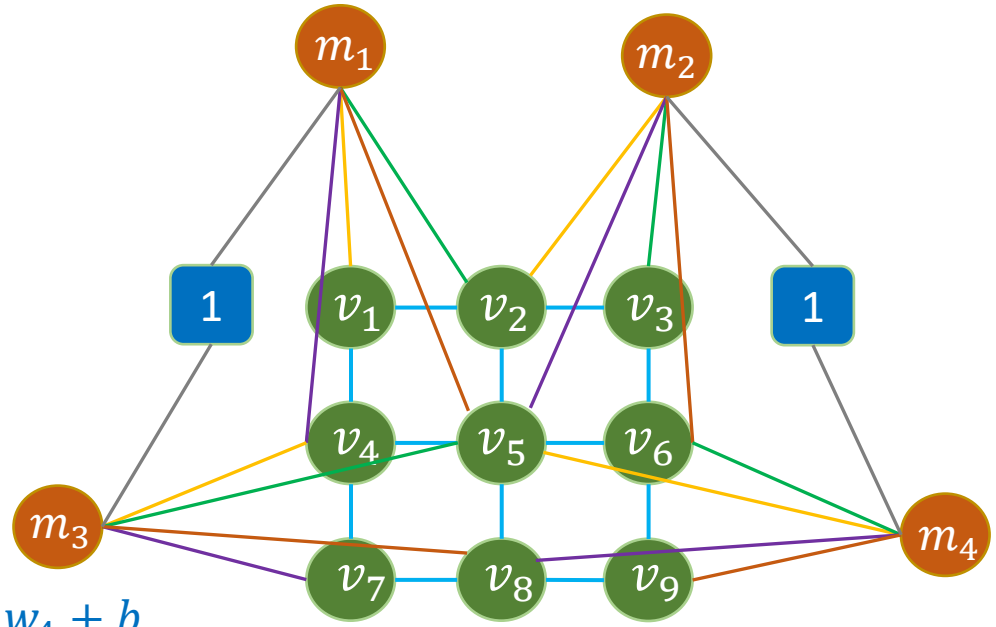
MLP

10 parameters

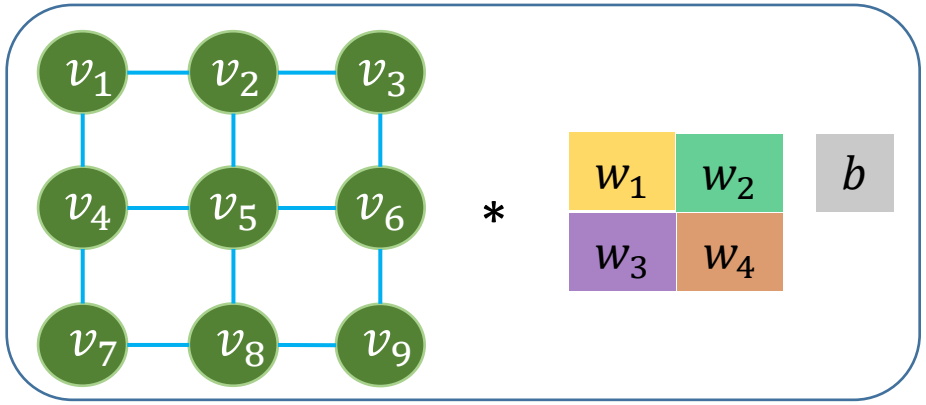
CNN

5 parameters

Kernel of parameters

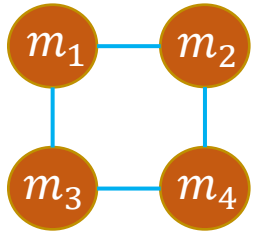


$$m_1 = v_1 w_1 + v_2 w_2 + v_4 w_3 + v_5 w_4 + b$$



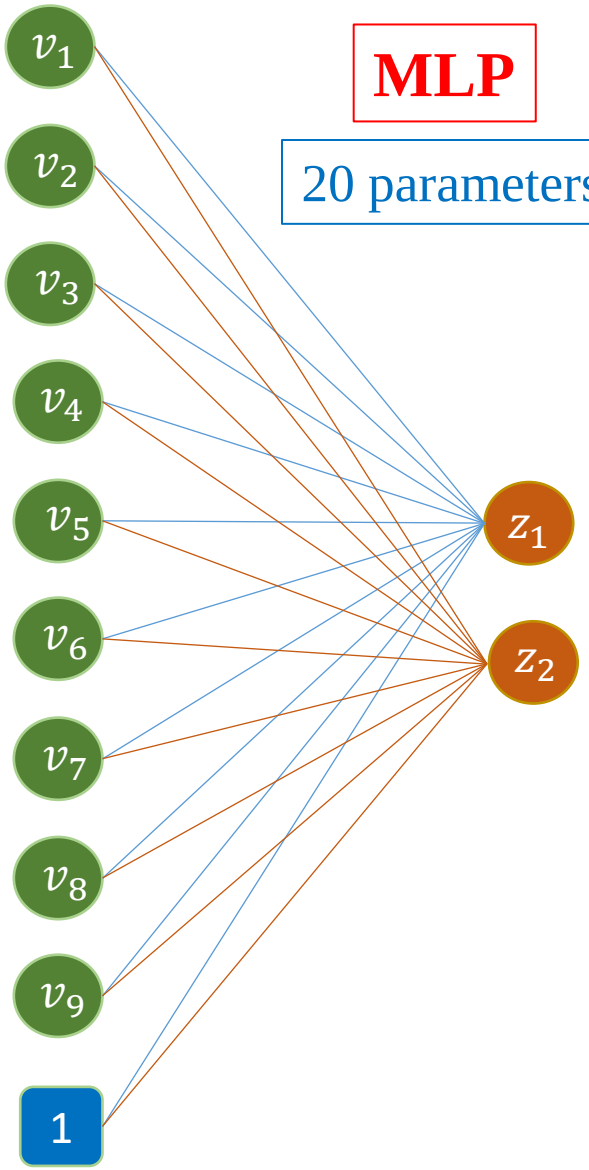
Convolution(~Correlation)

➡



Feature Map

From MLP to CNN

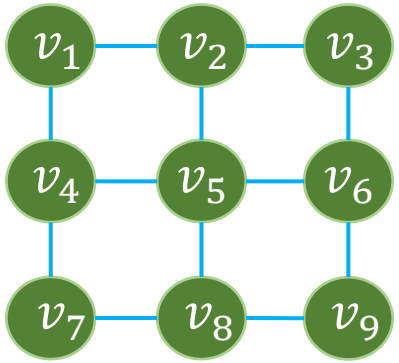


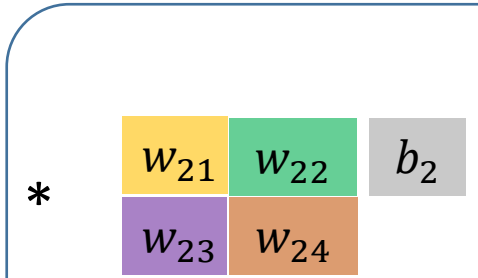
MLP

20 parameters

CNN

10 parameters

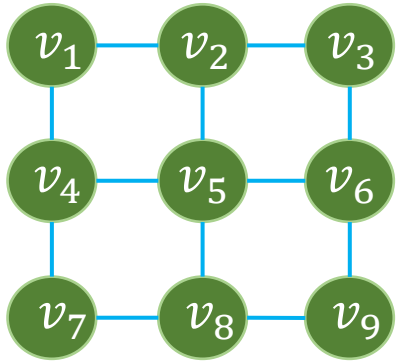




Kernel 1 $\neq$ Kernel 2

From MLP to CNN

❖ Understand convolution



(Height=3, Width=3, Channel=1)

Shape=(3,3,1)

*

Shape=(2,2,1)

#parameters (including bias) = 5

#channels of data must = #channels of kernel

Convolution

❖ Example

0	0	1	2	2
1	2	2	1	2
0	2	0	2	1
0	1	1	1	0
1	0	0	0	1

Data **D**

Bias b = 0.0

0.0	0.1	-0.1
-0.2	0.0	0.1
0.0	0.0	0.1

Kernel **K**

m_1		

Output

Data size = 5×5

Kernel size = 3×3

Stride = 1

$m_1 = 0\times0.0 + 0\times0.1 + 1\times-0.1 +$

$1\times-0.2 + 2\times0.0 + 2\times0.1 +$

$0\times0.0 + 2\times0.0 + 0\times0.1$

➡ $m_1 = -0.1$

Convolution

❖ Example

$$S_o = \frac{S_D - K}{S} + 1$$

0	0	1	2	2
1	2	2	1	2
0	2	0	2	1
0	1	1	1	0
1	0	0	0	1

Data **D**

Bias b = 0.0

0.0	0.1	-0.1
-0.2	0.0	0.1
0.0	0.0	0.1

Kernel **K**

-0.1	-0.1	-0.2
0.3	-0.2	0.1
0.3	-0.3	0.1

Output

Data size = 5×5

Kernel size = 3×3

Stride = 1

Convolution

❖ Example

0	0	1	2	2
1	2	2	1	2
0	2	0	2	1
0	1	1	1	0
1	0	0	0	1

Data **D**

Bias b = 0.0

0.0	0.1	-0.1
-0.2	0.0	0.1
0.0	0.0	0.1

Kernel **K**

m_1	

Output

Data size = 5×5

Kernel size = 3×3

Stride = 2

$m_1 = 0\times0.0 + 0\times0.1 + 1\times-0.1 +$

$1\times-0.2 + 2\times0.0 + 2\times0.1 +$

$0\times0.0 + 2\times0.0 + 0\times0.1$

➡ $m_1 = -0.1$

Convolution

❖ Example

$$S_o = \frac{S_D - K}{S} + 1$$

0	0	1	2	2
1	2	2	1	2
0	2	0	2	1
0	1	1	1	0
1	0	0	0	1

Data **D**

Bias b = 0.0

0.0	0.1	-0.1
-0.2	0.0	0.1
0.0	0.0	0.1

Kernel **K**

-0.1	-0.2
0.3	0.1

Output

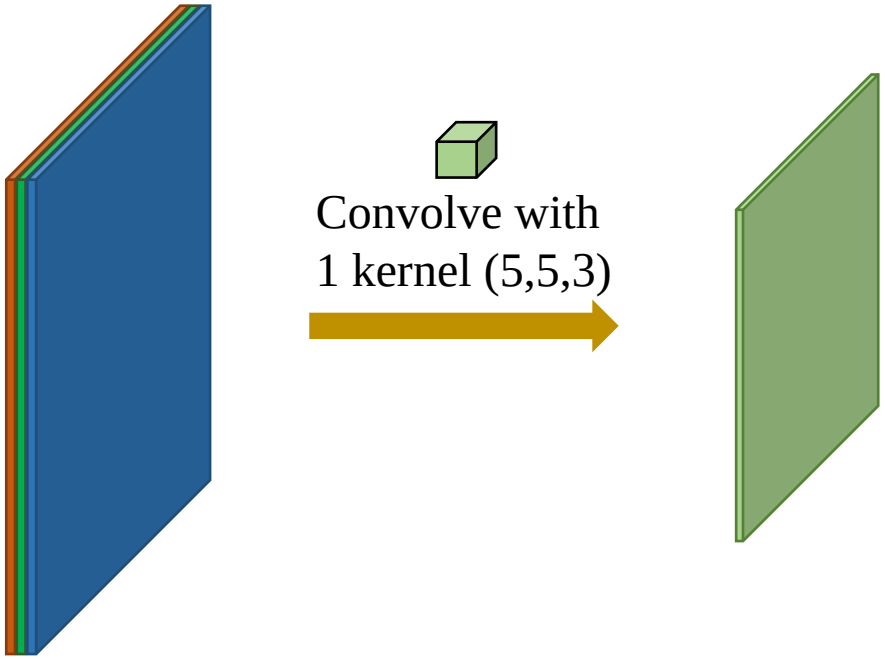
Data size = 5×5

Kernel size = 3×3

Stride = 2

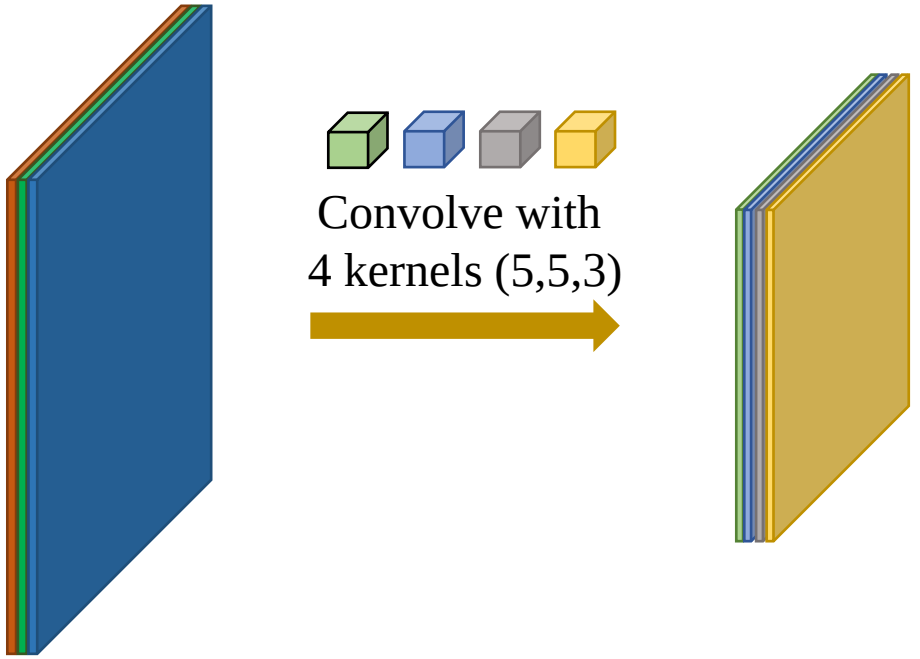
Convolutional Neural Network

❖ Understand convolution



Input Data
(32,32,3)

Feature map
(28,28,1)

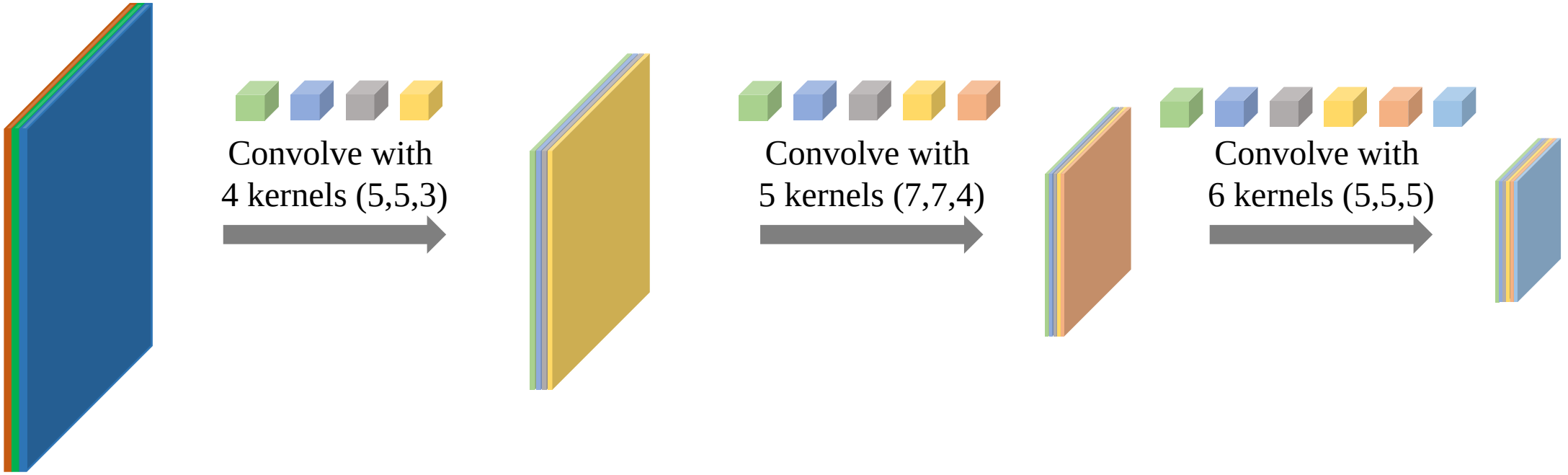


Input Data
(32,32,3)

Feature maps
(28,28,4)

Convolutional Neural Network

❖ A stack of convolutions



Input Data
(32,32,3)

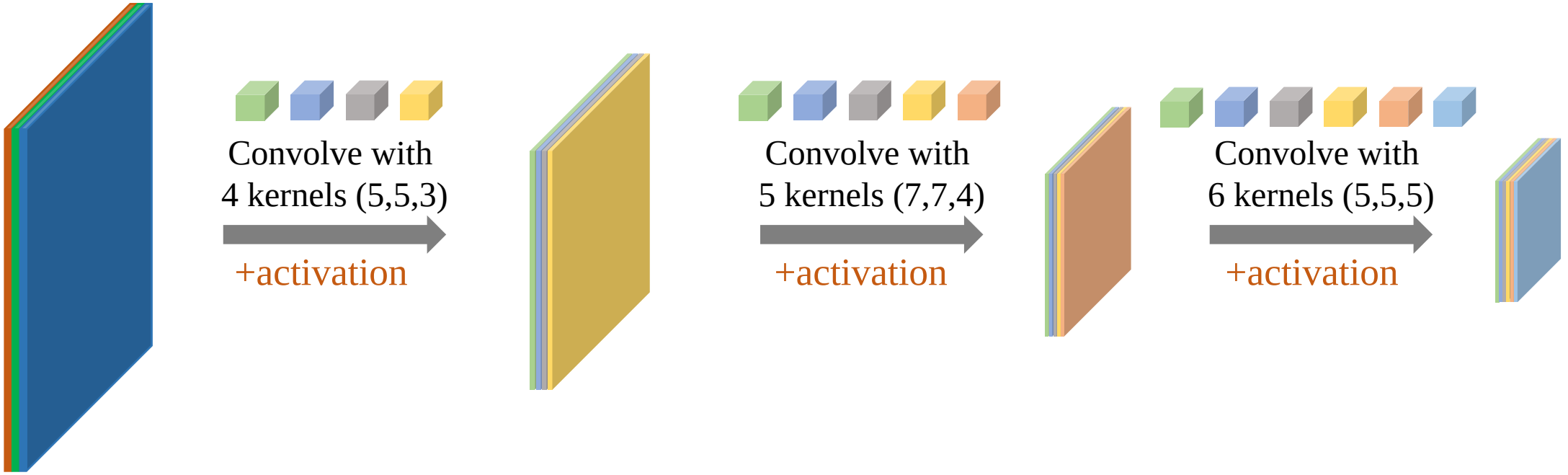
Feature maps
(28,28,4)

Feature maps
(22,22,5)

Feature maps
(18,18,6)

Convolutional Neural Network

❖ A stack of pairs of convolution+activation



Input Data
(32,32,3)

Feature maps
(28,28,4)

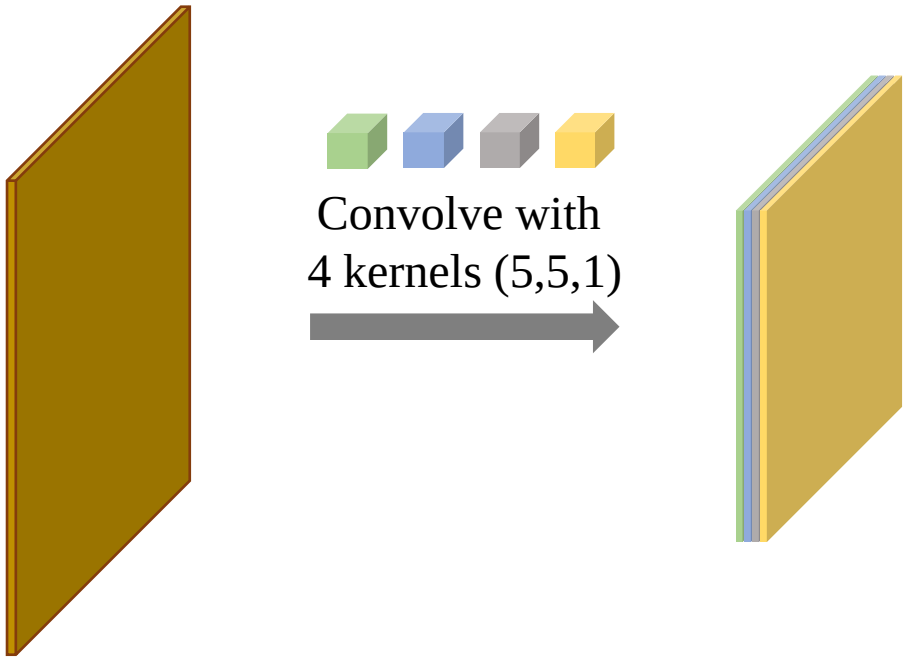
Feature maps
(22,22,5)

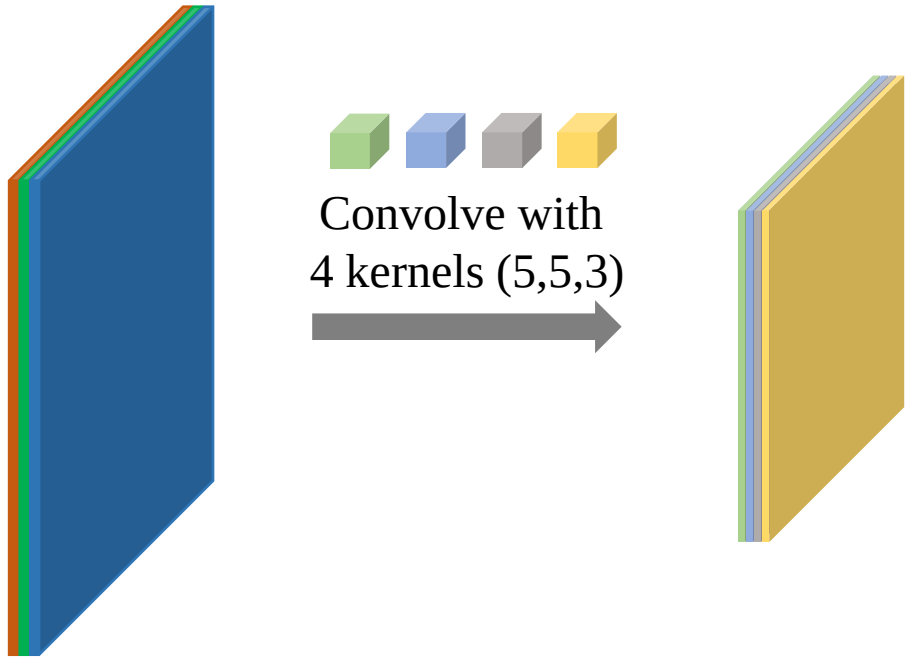
Feature maps
(18,18,6)

Convolutional Neural Network

❖ Convolution layer in Keras

nn.Conv2d(in_channels, out_channels, kernel_size)



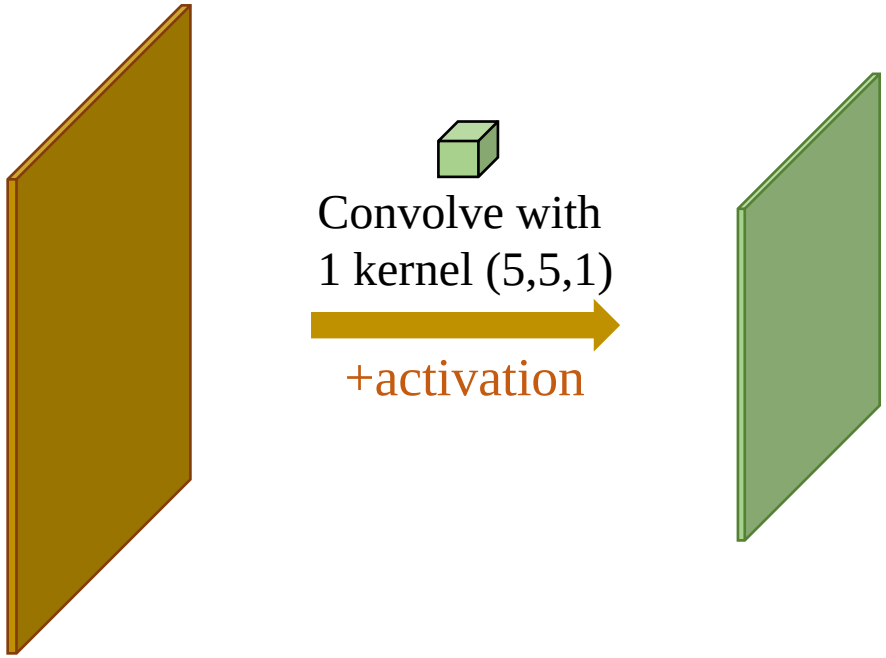


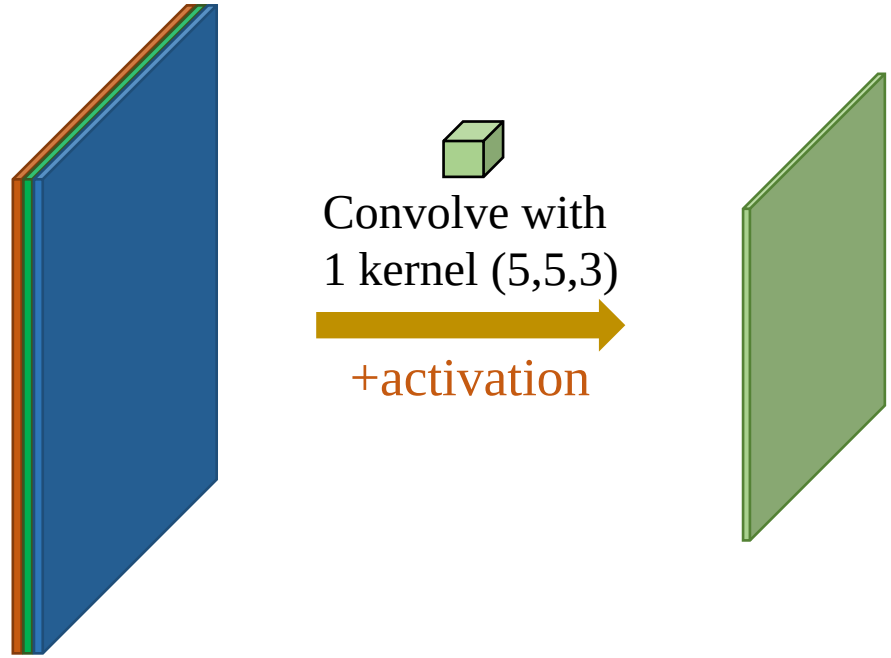
Convolutional Neural Network

❖ Convolution layer in Keras

demo

nn.Conv2d(in_channels, out_channels, kernel_size)





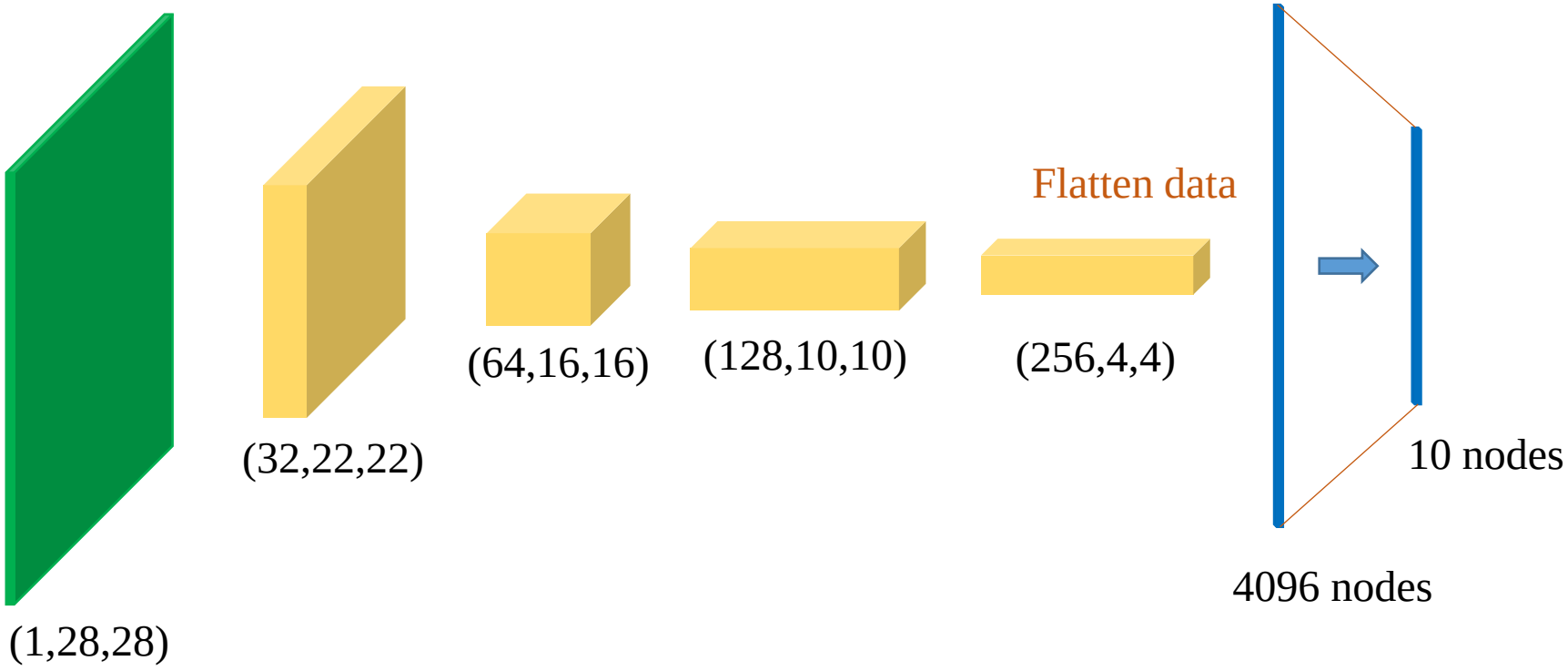
Convolutional Neural Network

Fashion-MNIST dataset

- Grayscale images
- Resolution=28x28
- Training set: 60000 samples
- Testing set: 10000 samples

Convolutional Neural Network

❖ Apply for Fashion-MNIST dataset



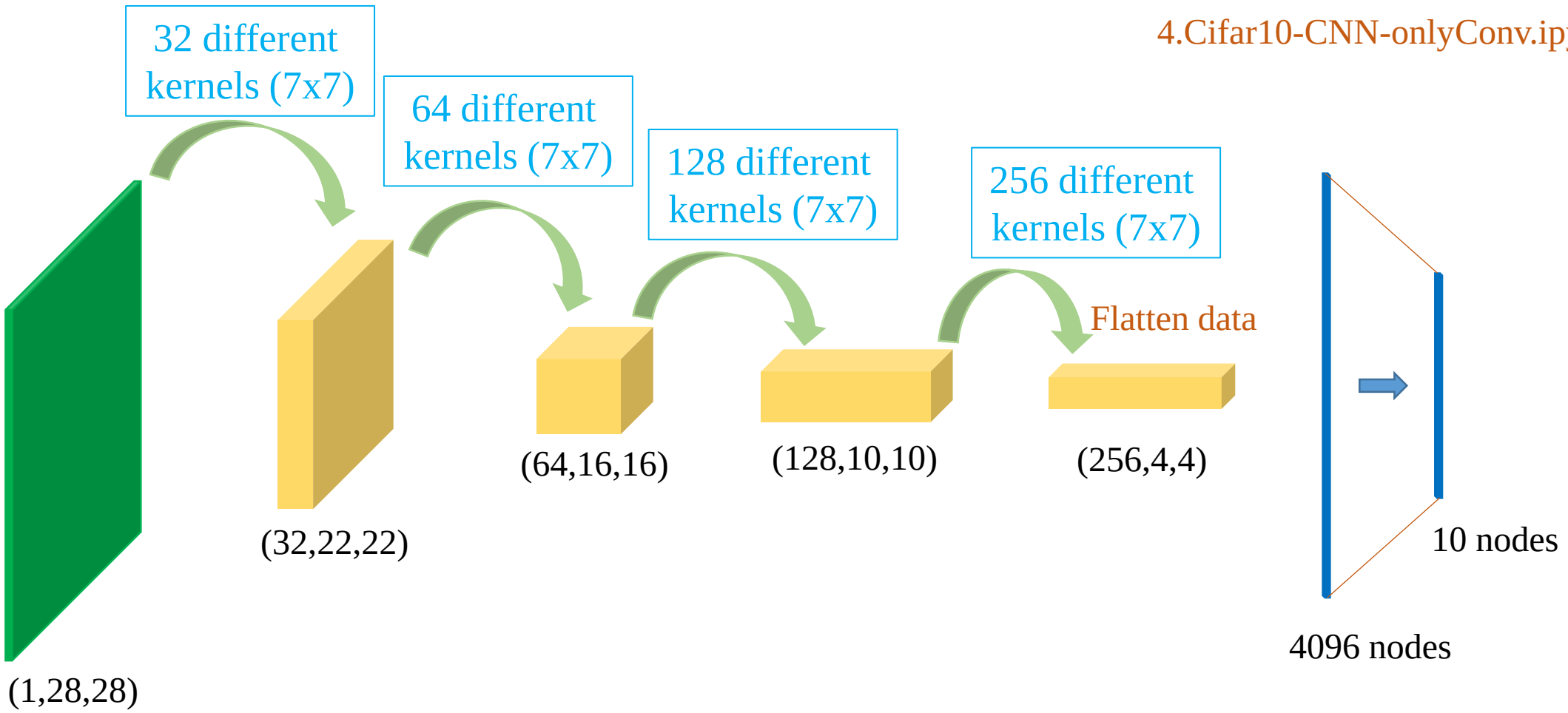
Convolutional Neural Network

❖ Apply for Fashion-MNIST dataset

2.CNN_Models.ipynb

3.FashionMNIST-CNN-onlyConv.ipynb

4.Cifar10-CNN-onlyConv.ipynb



Simple Convolutional Neural Network

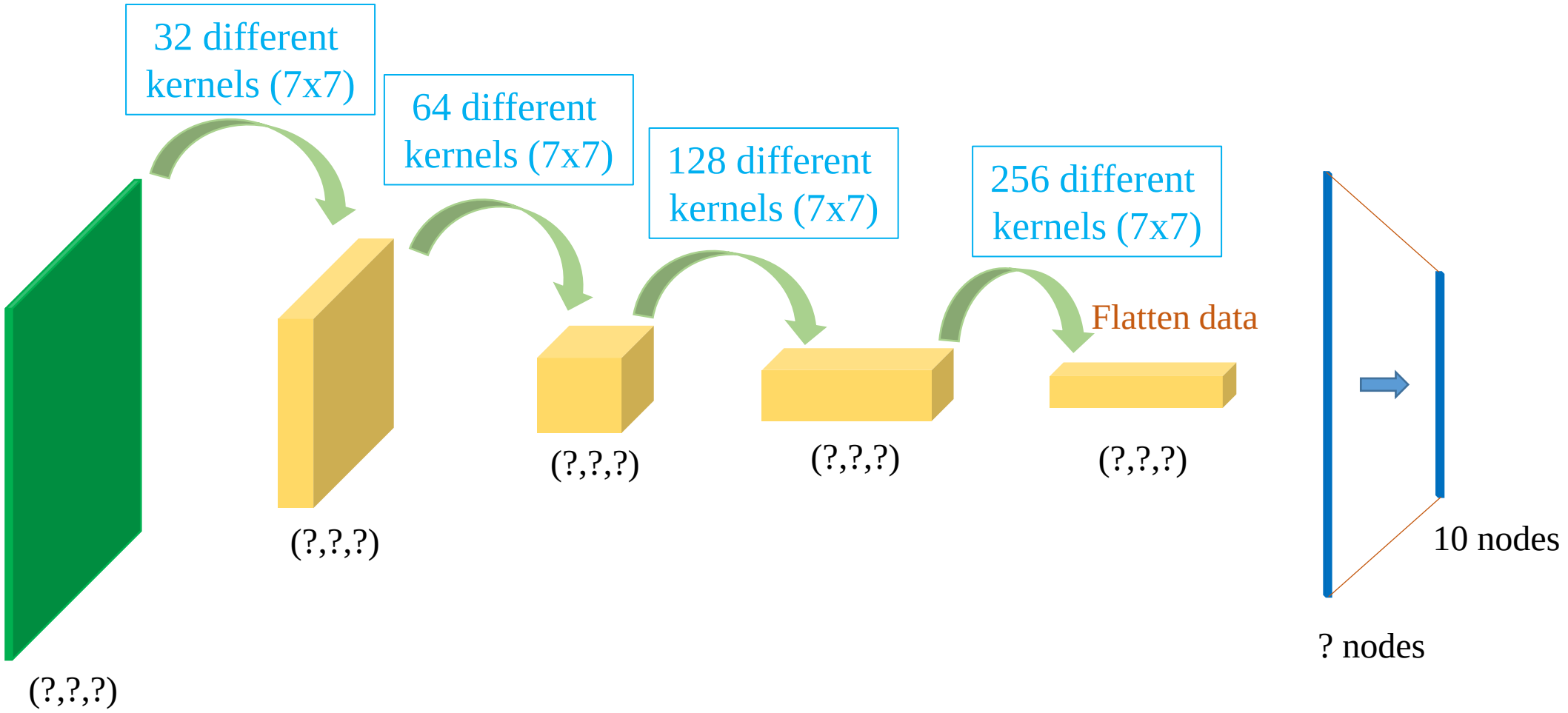
```python
class CustomModel(nn.Module):
    def __init__(self):
        super(CustomModel, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=7)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=7)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=7)
        self.conv4 = nn.Conv2d(128, 256, kernel_size=7)
        self.flatten = nn.Flatten()
        self.dense = nn.Linear(4*4*256, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = self.relu(self.conv3(x))
        x = self.relu(self.conv4(x))
        x = self.flatten(x)
        x = self.dense(x)
        return x

model = CustomModel()
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 22, 22]           1,600
              ReLU-2          [-1, 32, 22, 22]               0
            Conv2d-3          [-1, 64, 16, 16]         100,416
              ReLU-4          [-1, 64, 16, 16]               0
            Conv2d-5         [-1, 128, 10, 10]         401,536
              ReLU-6         [-1, 128, 10, 10]               0
            Conv2d-7           [-1, 256, 4, 4]       1,605,888
              ReLU-8           [-1, 256, 4, 4]               0
           Flatten-9                [-1, 4096]               0
           Linear-10                 [-1, 128]         524,416
             ReLU-11                 [-1, 128]               0
           Linear-12                  [-1, 10]           1,290
================================================================
Total params: 2,635,146
Trainable params: 2,635,146
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.78
Params size (MB): 10.05
Estimated Total Size (MB): 10.83
----------------------------------------------------------------
```

Convolutional Neural Network

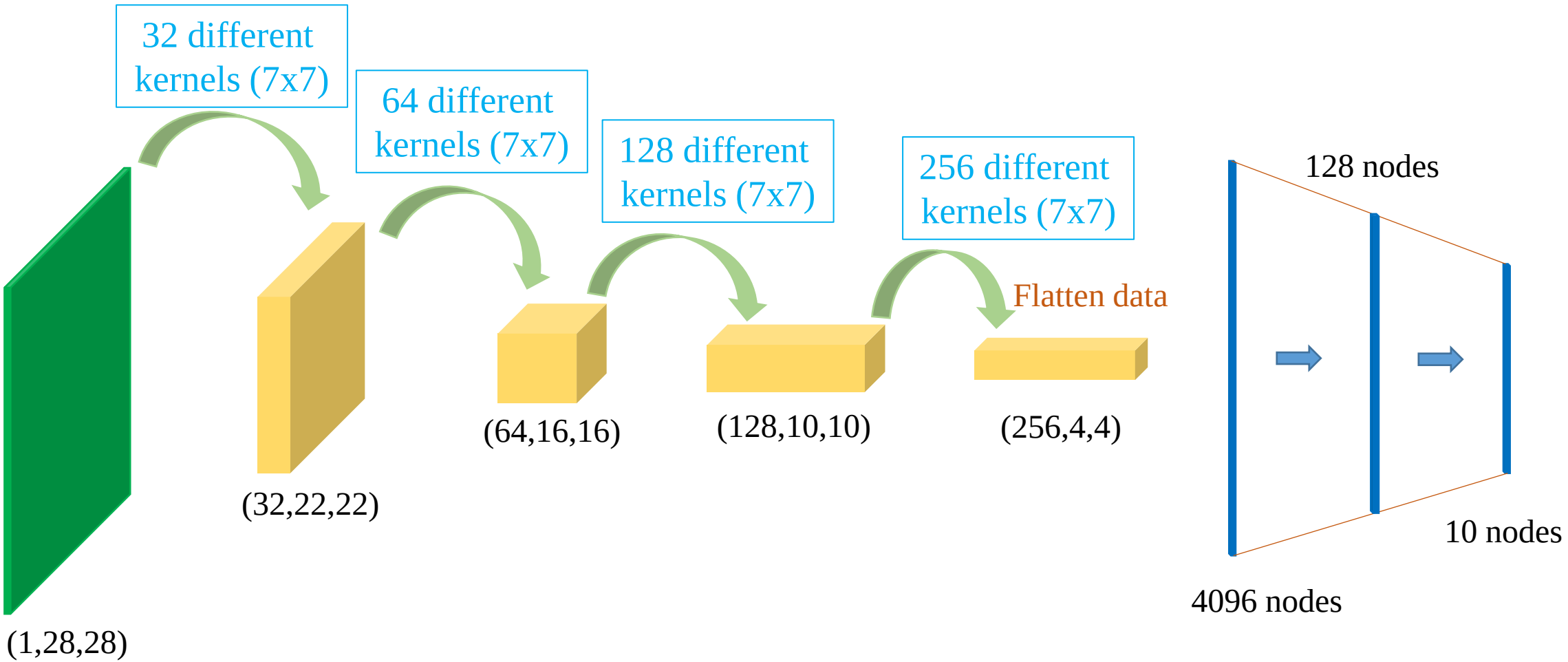
❖ Apply for Cifar-10 dataset

demo



Convolutional Neural Network

❖ Apply for Fashion-MNIST dataset: case 1

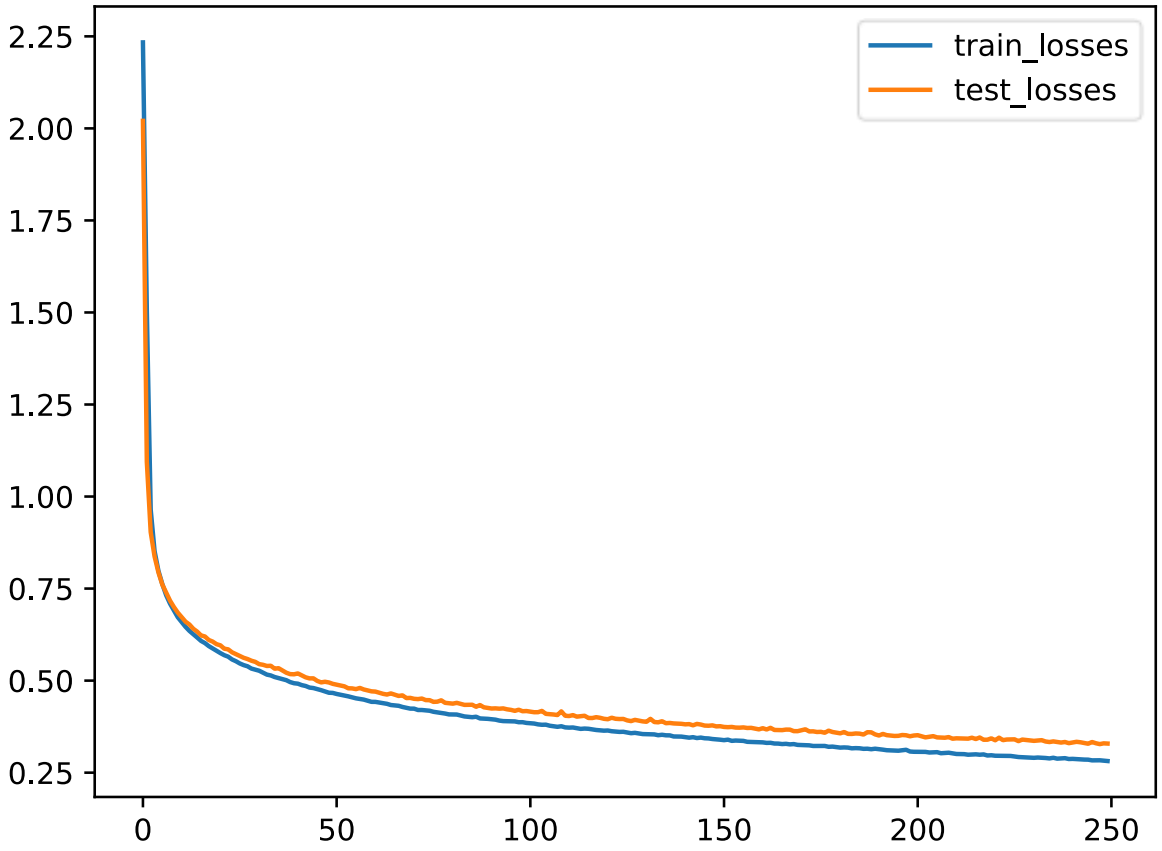


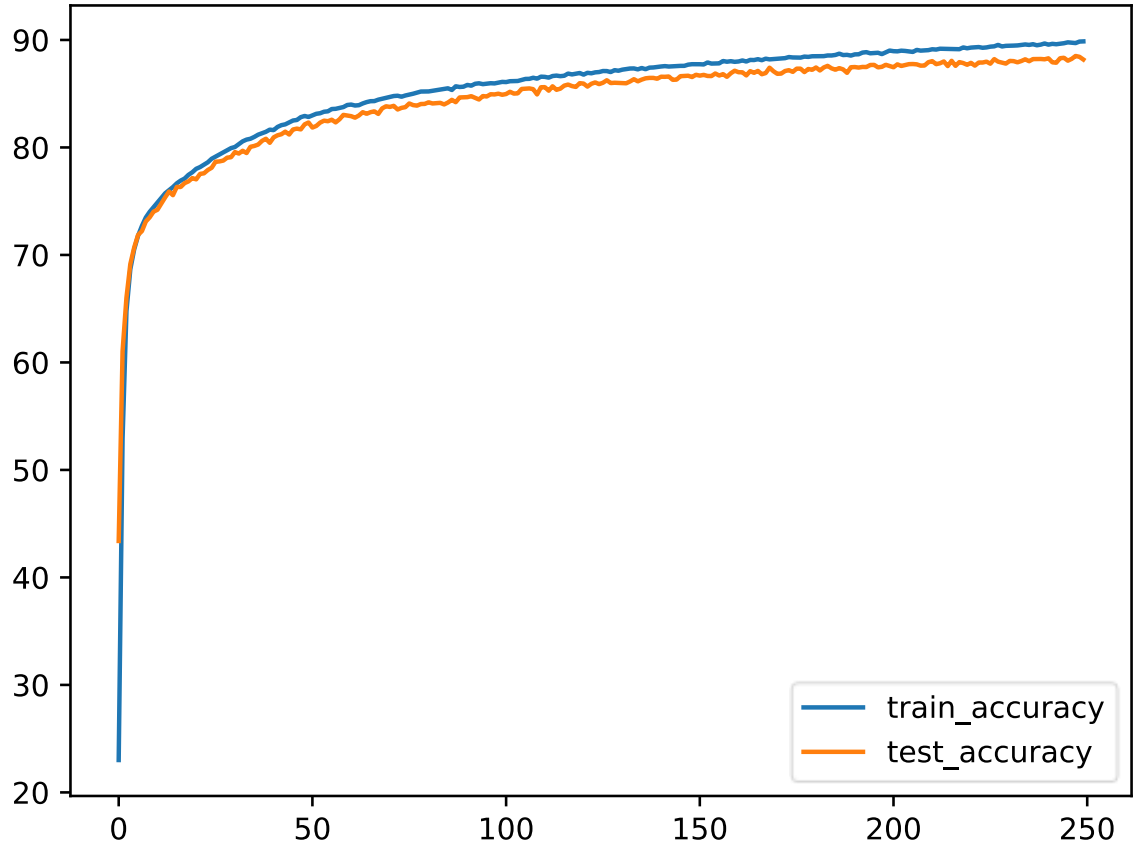
```python
class CustomModel(nn.Module):
    def __init__(self):
        super(CustomModel, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=7)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=7)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=7)
        self.conv4 = nn.Conv2d(128, 256, kernel_size=7)
        self.flatten = nn.Flatten()
        self.dense1 = nn.Linear(4*4*256, 128)
        self.dense2 = nn.Linear(128, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = self.relu(self.conv3(x))
        x = self.relu(self.conv4(x))
        x = self.flatten(x)
        x = self.relu(self.dense1(x))
        x = self.dense2(x)
        return x

model = CustomModel()
model = model.to(device)
```

```python
# Load FashionMNIST dataset
transform = Compose([ToTensor(),
                     Normalize((0.5,),
                               (0.5,))])

trainset = FashionMNIST(root='data',
                        train=True,
                        download=True,
                        transform=transform)
trainloader = DataLoader(trainset,
                         batch_size=1024,
                         num_workers=10,
                         shuffle=True,
                         drop_last=True)

testset = FashionMNIST(root='data',
                       train=False,
                       download=True,
                       transform=transform)
testloader = DataLoader(testset,
                        batch_size=1024,
                        num_workers=10,
                        shuffle=False)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=1e-5)
```

Convolutional Neural Network

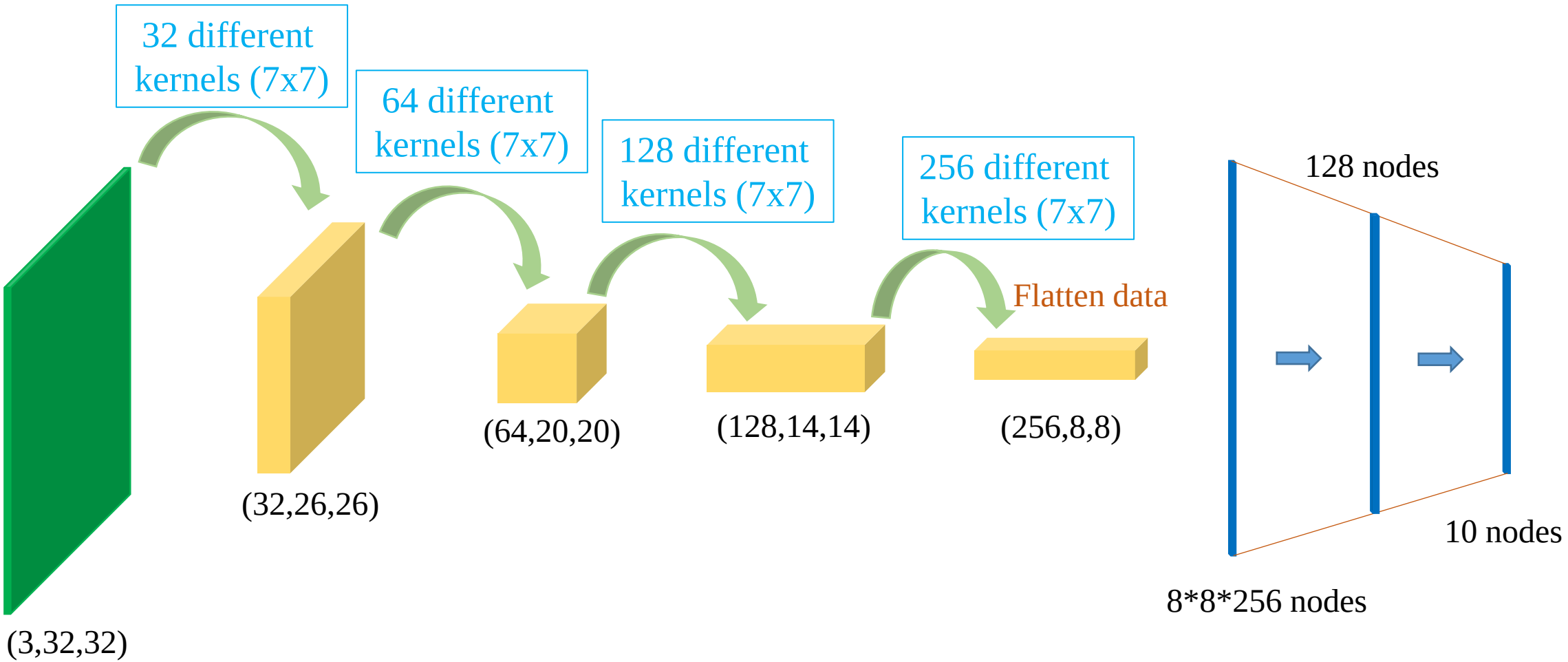
❖ Apply for Fashion-MNIST dataset: case 1





Convolutional Neural Network

❖ Apply for Cifar-10 dataset: case 2

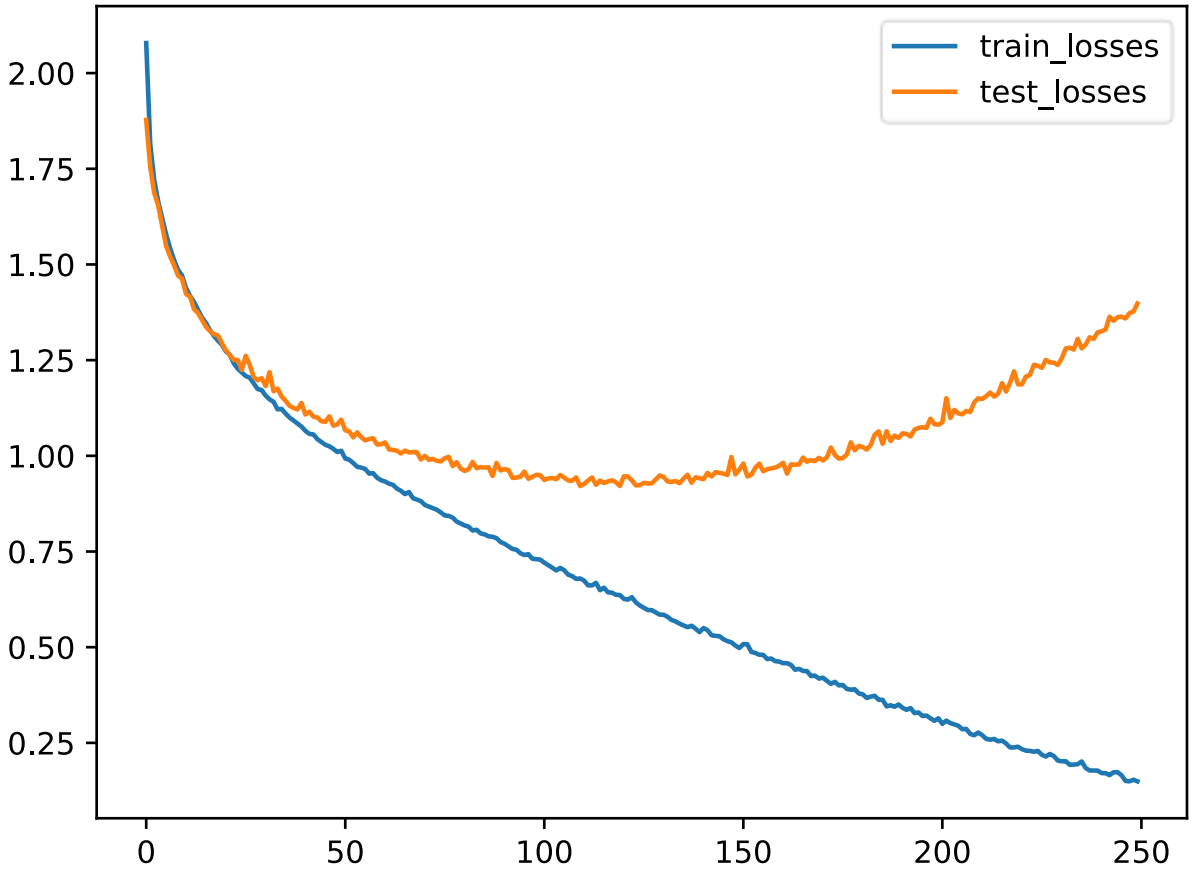


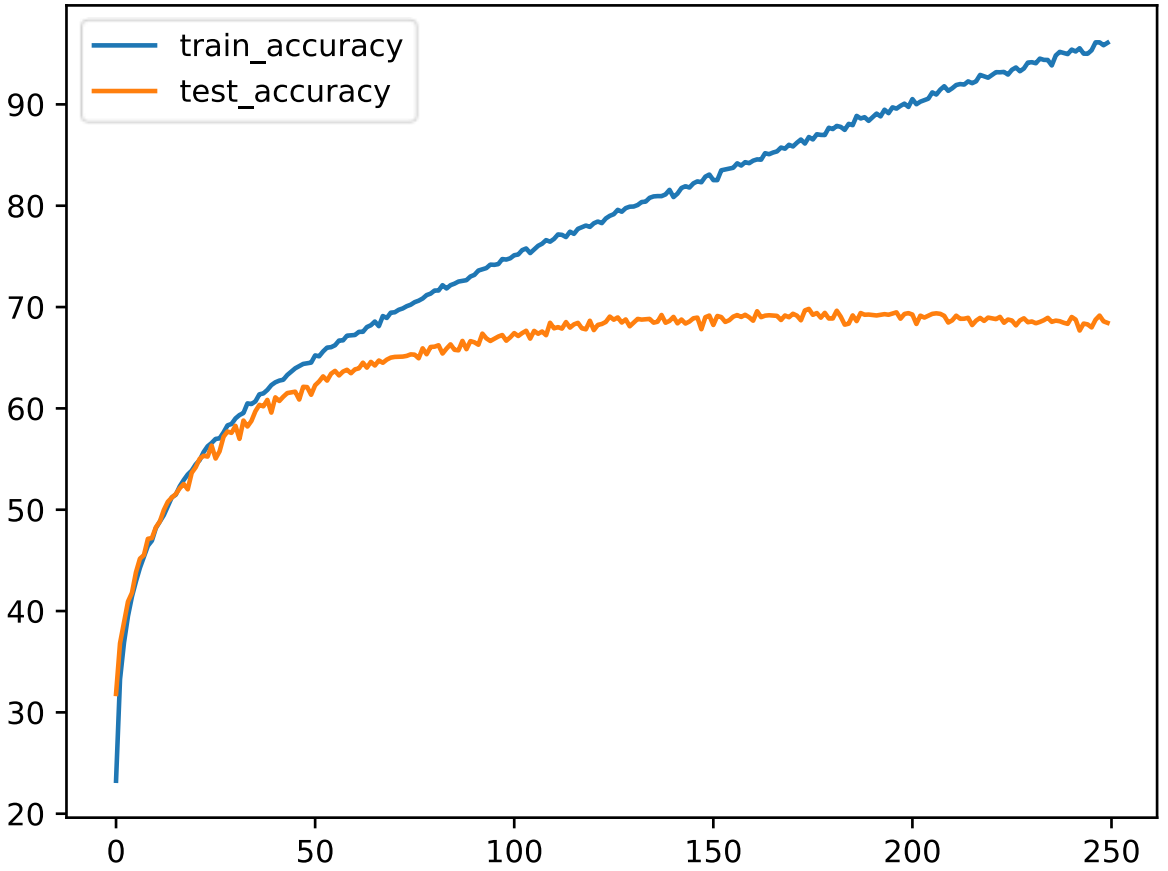
```python
class CustomModel(nn.Module):
    def __init__(self):
        super(CustomModel, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=7)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=7)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=7)
        self.conv4 = nn.Conv2d(128, 256, kernel_size=7)
        self.flatten = nn.Flatten()
        self.dense1 = nn.Linear(8*8*256, 128)
        self.dense2 = nn.Linear(128, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = self.relu(self.conv3(x))
        x = self.relu(self.conv4(x))
        x = self.flatten(x)
        x = self.relu(self.dense1(x))
        x = self.dense2(x)
        return x

model = CustomModel()
model = model.to(device)
```

```python
# Load CIFAR10 dataset
transform = Compose([ToTensor(),
                     Normalize((0.5,0.5, 0.5),
                               (0.5,0.5, 0.5))])

trainset = CIFAR10(root='data',
                   train=True,
                   download=True,
                   transform=transform)
trainloader = DataLoader(trainset,
                         batch_size=1024,
                         num_workers=10,
                         shuffle=True,
                         drop_last=True)

testset = CIFAR10(root='data',
                  train=False,
                  download=True,
                  transform=transform)
testloader = DataLoader(testset,
                        batch_size=1024,
                        num_workers=10,
                        shuffle=False)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=1e-5)
```

Convolutional Neural Network

❖ Apply for Cifar-10 dataset: case 2





Further Reading

❖ Reading

https://cs231n.github.io/convolutional-networks/

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks