Motivation and Introduction to CNN

Quang-Vinh Dinh Ph.D. in Computer Science

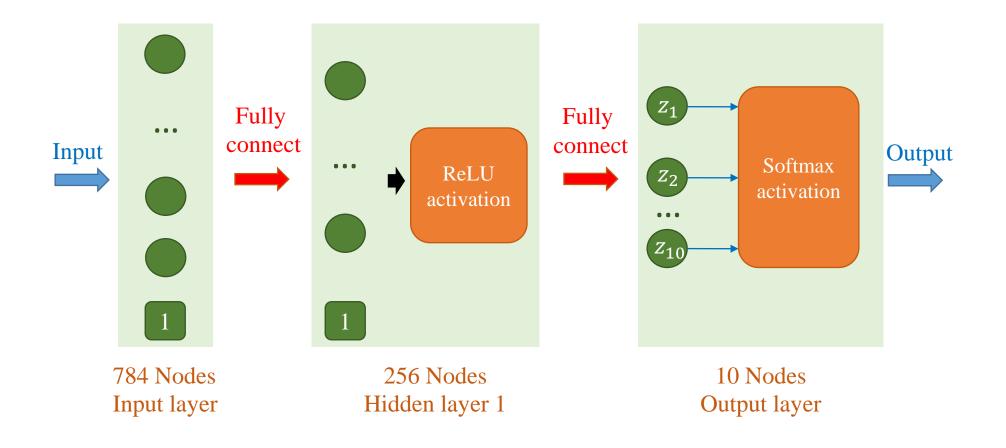
Outline

- > MLP Limitation
- > From MLP to CNN
- Feature Map Down-sampling
- > Some Examples
- > Application to Cifar10

	T-shirt	
	Trouser	
Fashion-MNIST dataset	Pullover	
	Dress	
Grayscale images	Coat	
Resolution=28x28	Sandal	JA
Training set: 60000 samples	Shirt	
Testing set: 10000 samples	Sneaker	
	Bag	
	Ankle Boot	鱼鱼人人人人

MLP for Fashion-MNIST

Case 1



MLP for Fashion-MNIST

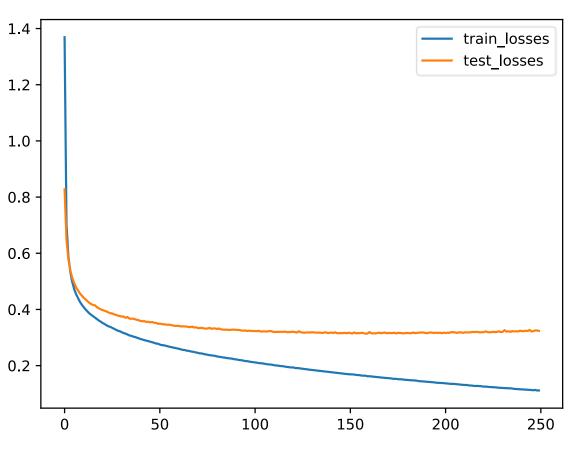
Case 1

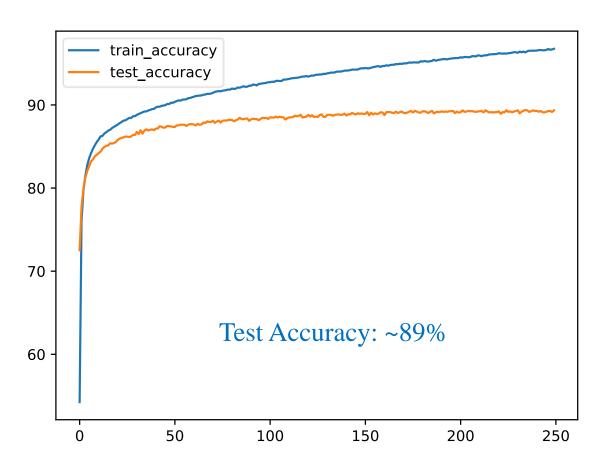
```
# model
model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(784, 256),
    nn.ReLU(),
    nn.Linear(256, 10)
)
```

```
# 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

Case 1





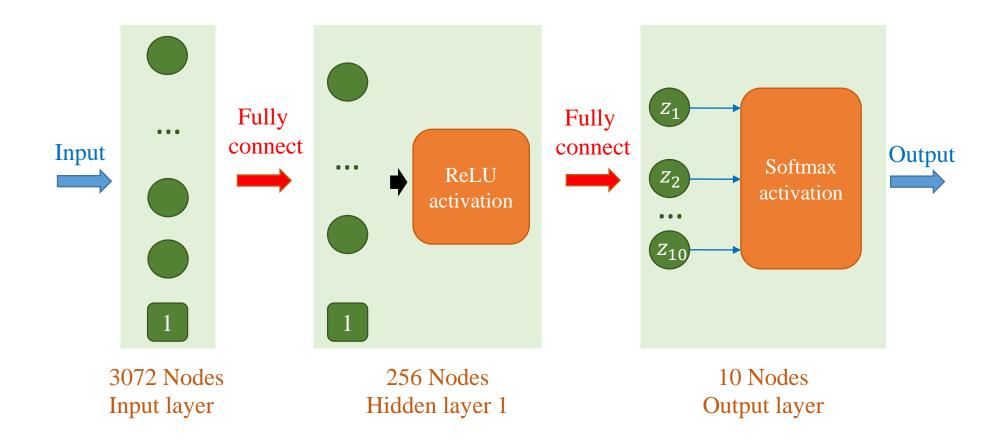
Adam with learning rate of 1e-4

Perform reasonably

	airplane	*	*		1	*	-	-		-	東
a	utomobile				· 法						1
	bird	4		and the same	K	1	49	4	-	2	3
Cifar-10 dataset	cat		500	*	A		(A	4	700		4
	deer			30	ME TO SE	- L	m	P	·v		
Color images	dog	A.		3	A.	-			2.	1	9
Resolution=32x32						COS		efficie.			100 S
Training set: 50000 samples	frog	100		1			1	64		V	1
Testing set: 10000 samples	horse	KA	2	4	Vi n	P	庆	1	H	15	
	ship	E	-	-	逝	2	4.45	41			
	truck						2		Name of Street		

MLP for Cifar-10

Case 2

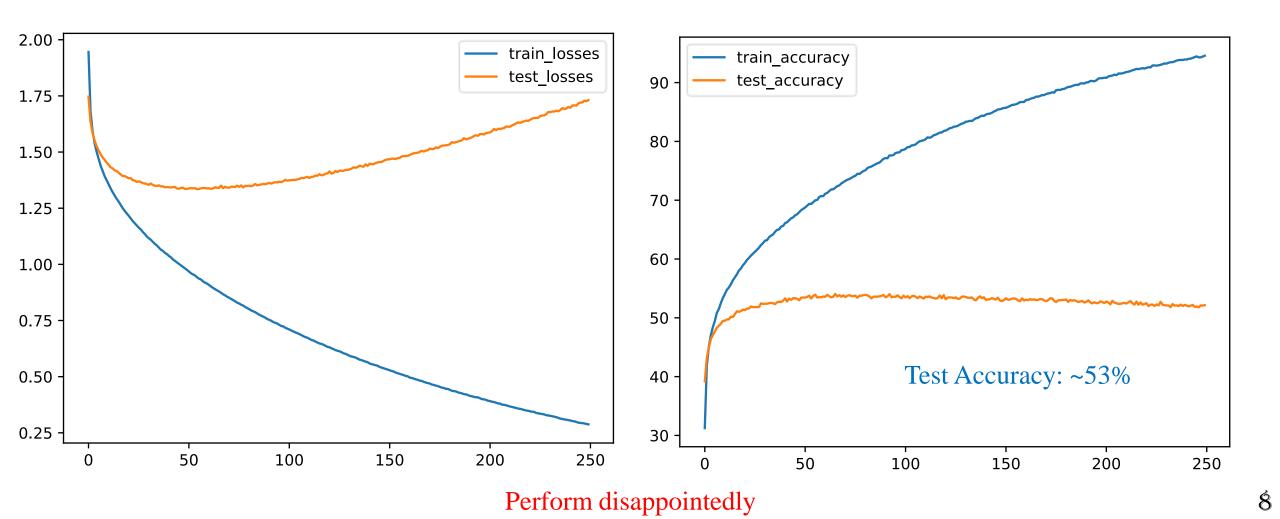


```
# model
model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(32*32*3, 256),
    nn.ReLU(),
    nn.Linear(256, 10)
)
```

```
# 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)
```

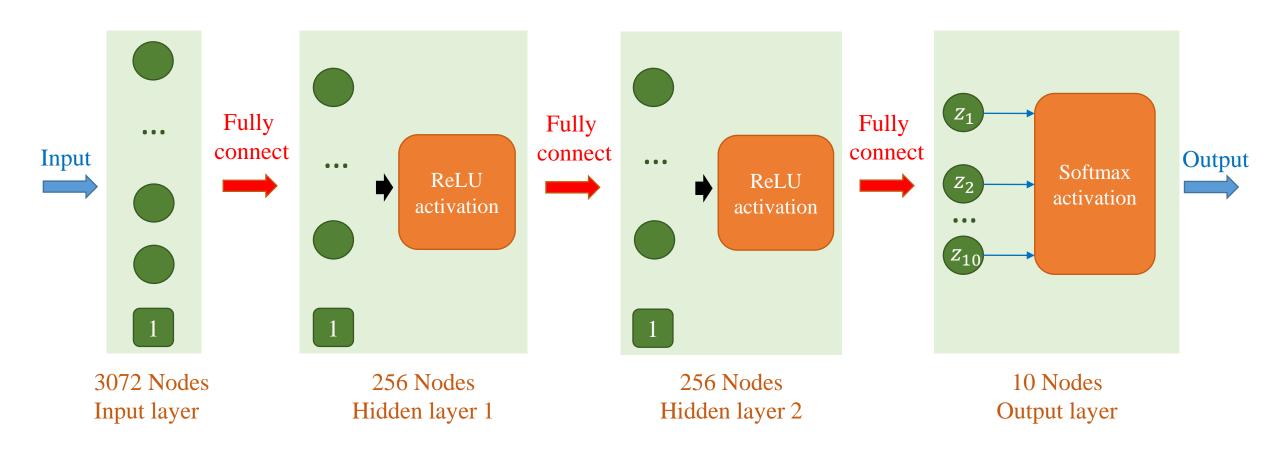
MLP for Cifar-10

Case 2



MLP for Cifar-10

* ReLU, He and Adam: add more layers



* ReLU, He and Adam

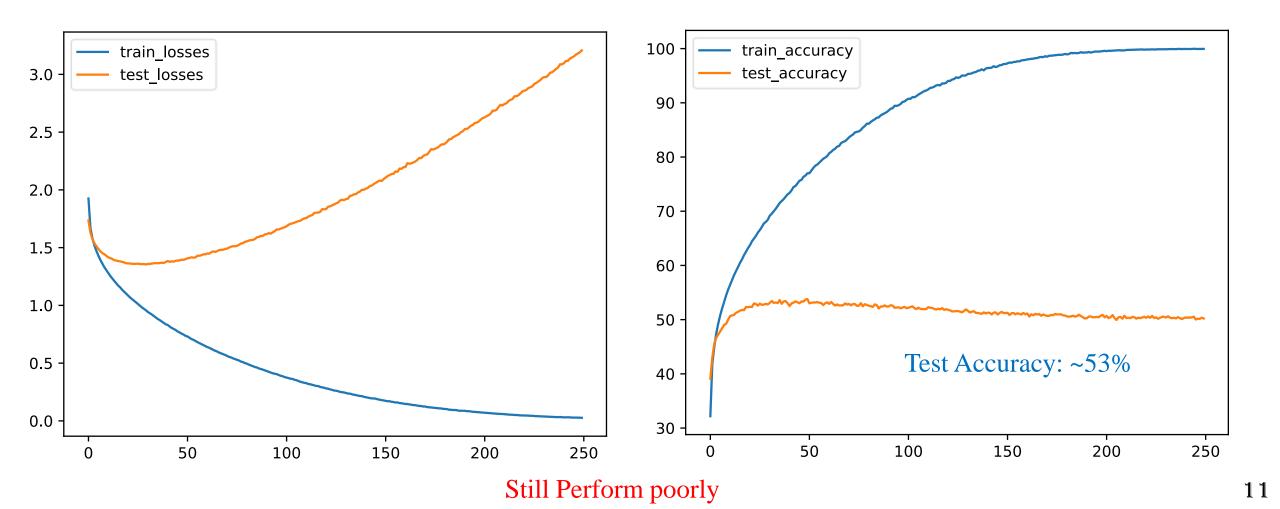
criterion = nn.CrossEntropyLoss()

optimizer = optim.Adam(model.parameters(),

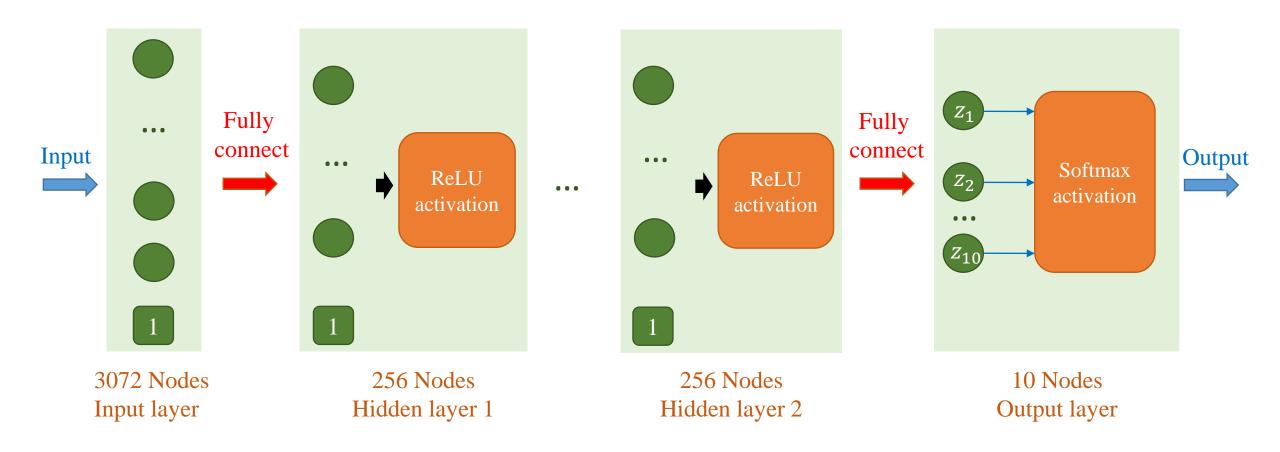
lr=0.001)

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

```
# 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



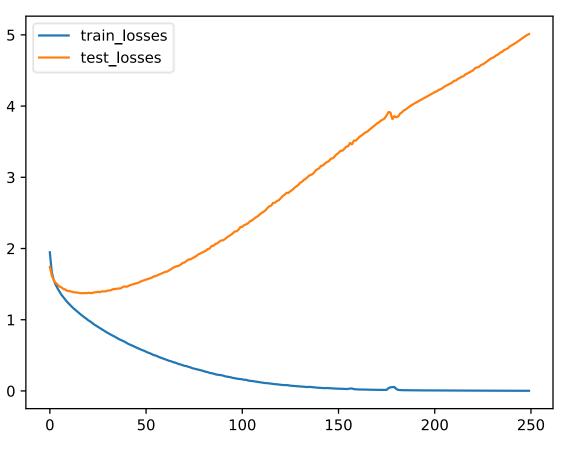
MLP for Cifar-10

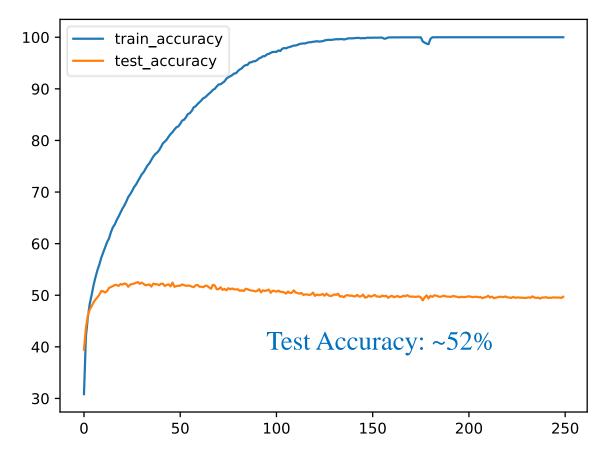
Case 4

```
# 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)
)
```

```
# 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)
```

ReLU, He and Adam: Using 3 hidden layers

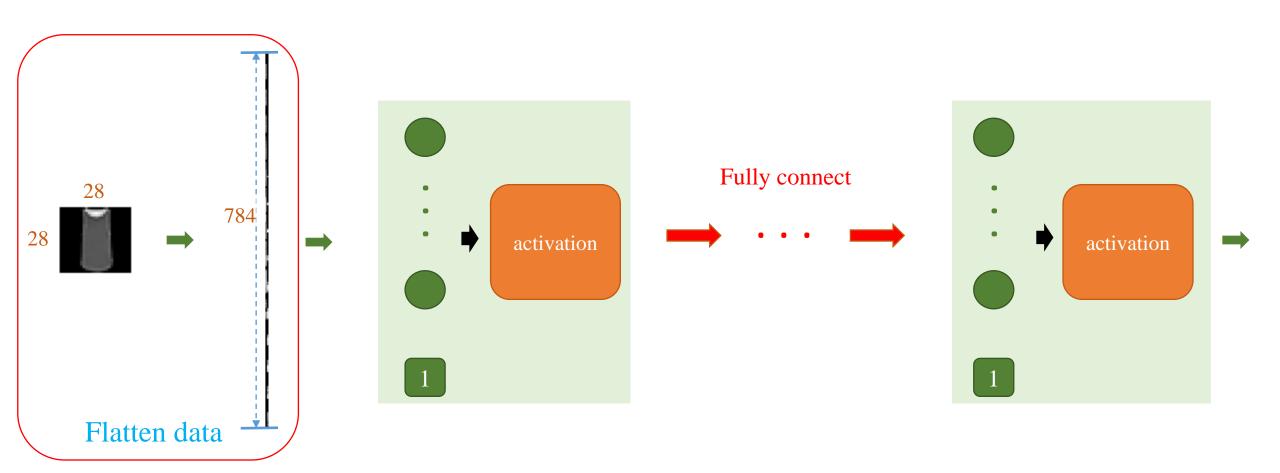




Outline

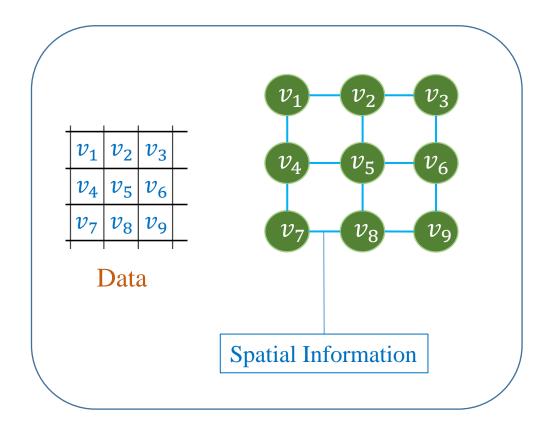
- > MLP Limitation
- > From MLP to CNN
- Feature Map Down-sampling
- > Some Examples
- > Application to Cifar10

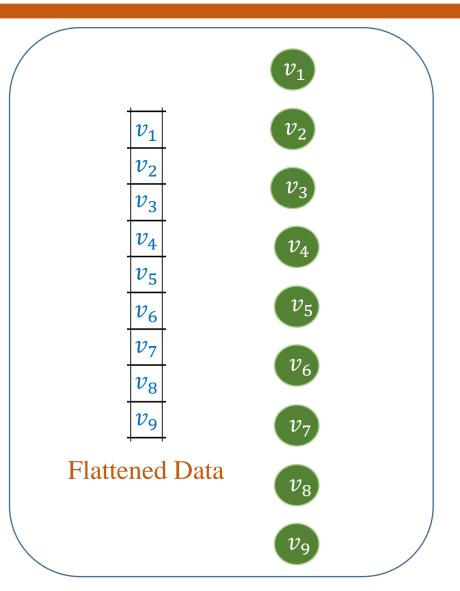
***** Multi-layer Perceptron

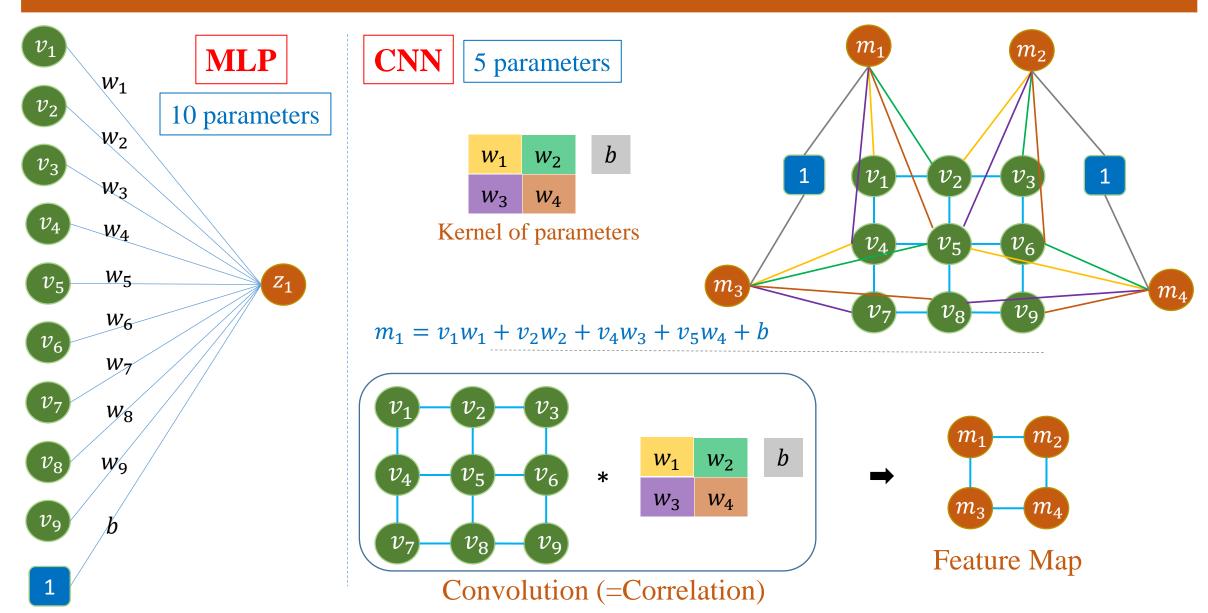


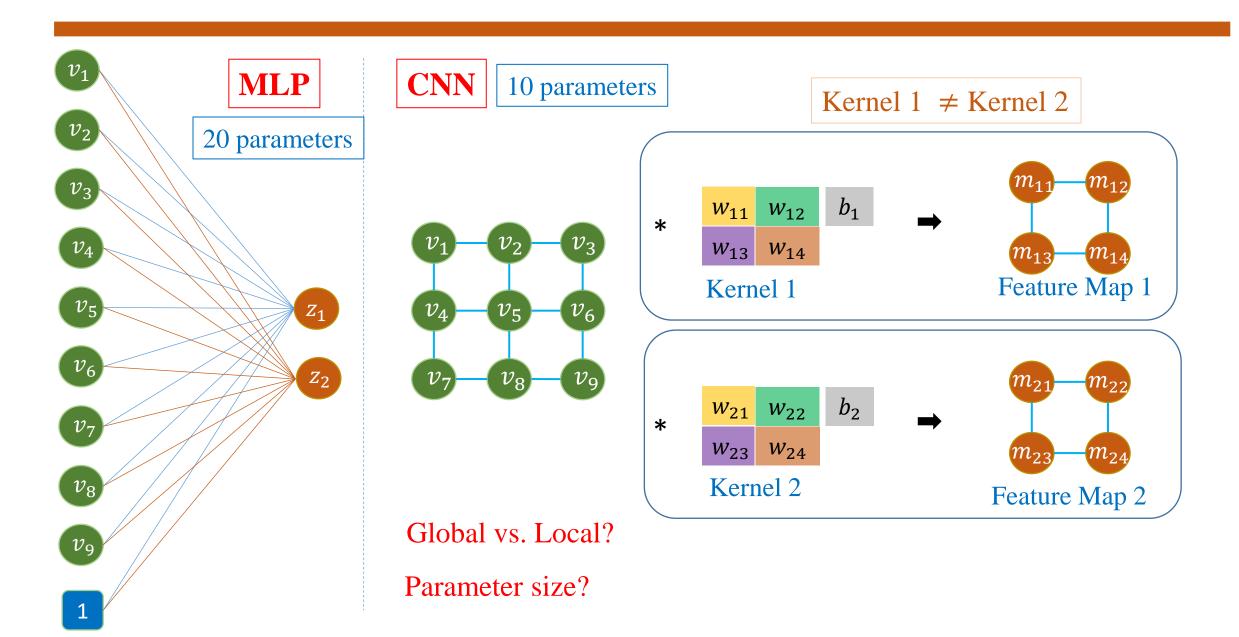
Problem: Remove spatial information of the data
Inefficiently have a large amount of parameters

Problem of flattening data



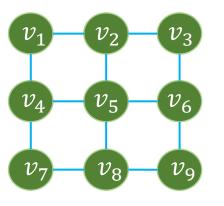






b

Understand convolution









 W_1

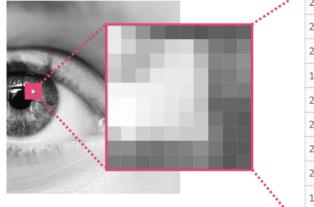
 W_3

 W_2

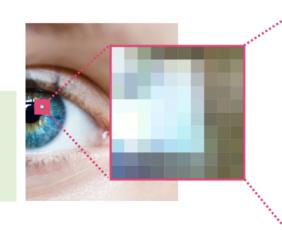
 W_4

Shape=(1,2,2)

#parameters (+bias) = 5



 230	194	147	108	90	98	84	96	91	101
237	206	188	195	207	213	163	123	116	128
210	183	180	205	224	234	188	122	134	147
198	189	201	227	229	232	200	125	127	135
249	241	237	244	232	226	202	116	125	126
251	254	241	239	230	217	196	102	103	99
243	255	240	231	227	214	203	116	95	91
204	231	208	200	207	201	200	121	95	95
144	140	120	115	125	127	143	118	92	91
121	121	108	109	122	121	134	106	86	97



			233	188	137	96	90	95	63	73	73	82
		237	202	159	120	105	110	88	107	112	121	109
•••	226	191	147	110	101	112	98	123	110	119	142	131
	221	191	176	182	203	214	169	144	133	145	155	122
	185	160	161	184	205	223	186	137	147	161	140	115
	181	174	189	207	206	215	194	136	142	151	133	87
	246	237	237	231	208	206	192	122	143	144	111	74
	254	254	241	224	199	192	181	99	122	117	107	74
	239	248	232	207	187	182	184	110	114	110	113	74
	193	215	193	167	158	164	181	114	112	111	105	82
	113	119	110	111	113	123	135	120	108	106	113	
٠,	93	97	91	103	107	111	122	112	104	114		

***** How many cases?

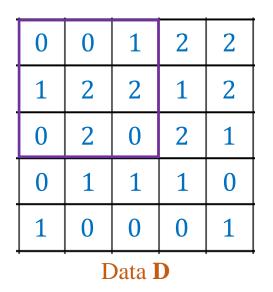
0	0	1	2	2
1	2	2	1	2
0	2	0	2	1
0	1	1	1	0
1	0	0	0	1

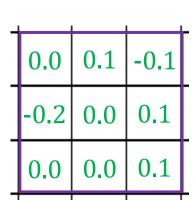
Data **D**

Bias b = 0.0

	0	0	1	2	2	0	0	1	2	2	0	0	1	2	2	
	1	2	2	1	2	1	2	2	1	2	1	2	2	1	2	
	0	2	0	2	1	0	2	0	2	1	0	2	0	2	1	
•	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	
	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	
•	0	0	1	2	2	0	0	1	2	2	0	0	1	2	2	
	1	2	2	1	2	1	2	2	1	2	1	2	2	1	2	
	0	2	0	2	1	0	2	0	2	1	0	2	0	2	1	
	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	
	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	
•	0	0	1	2	2	0	0	1	2	2	0	0	1	2	2	$ar{lack}$
	1	2	2	1	2	1	2	2	1	2	1	2	2	1	2	
	0	2	0	2	1	0	2	0	2	1	0	2	0	2	1	
	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	
	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	

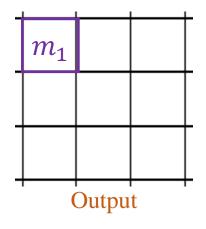
Example





Kernel **K**

Bias b = 0.0



Data size =
$$5 \times 5$$

Kernel size = 3×3

$$Stride = 1$$

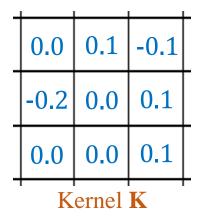
$$m_1 = 0 \times 0.0 + 0 \times 0.1 + 1 \times -0.1 +$$
 $1 \times -0.2 + 2 \times 0.0 + 2 \times 0.1 +$
 $0 \times 0.0 + 2 \times 0.0 + 0 \times 0.1$
 $m_1 = -0.1$

Example

$$S_o = \left\lfloor \frac{S_D - K}{S} \right\rfloor + 1$$

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
	Γ	oata I)	

Bias
$$b = 0.0$$

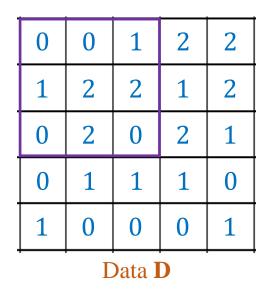


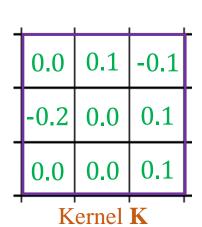
Data size =
$$5 \times 5$$

Kernel size =
$$3 \times 3$$

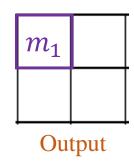
$$Stride = 1$$

Example





Bias b = 0.0



Data size =
$$5 \times 5$$

Kernel size =
$$3 \times 3$$

$$Stride = 2$$

$$m_1 = 0 \times 0.0 + 0 \times 0.1 + 1 \times -0.1 +$$

$$1 \times -0.2 + 2 \times 0.0 + 2 \times 0.1 +$$

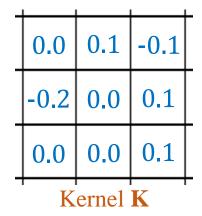
$$0 \times 0.0 + 2 \times 0.0 + 0 \times 0.1$$
 $m_1 = -0.1$

Example

$$S_o = \left\lfloor \frac{S_D - K}{S} \right\rfloor + 1$$

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
	Γ	oata I	D	•

Bias
$$b = 0.0$$

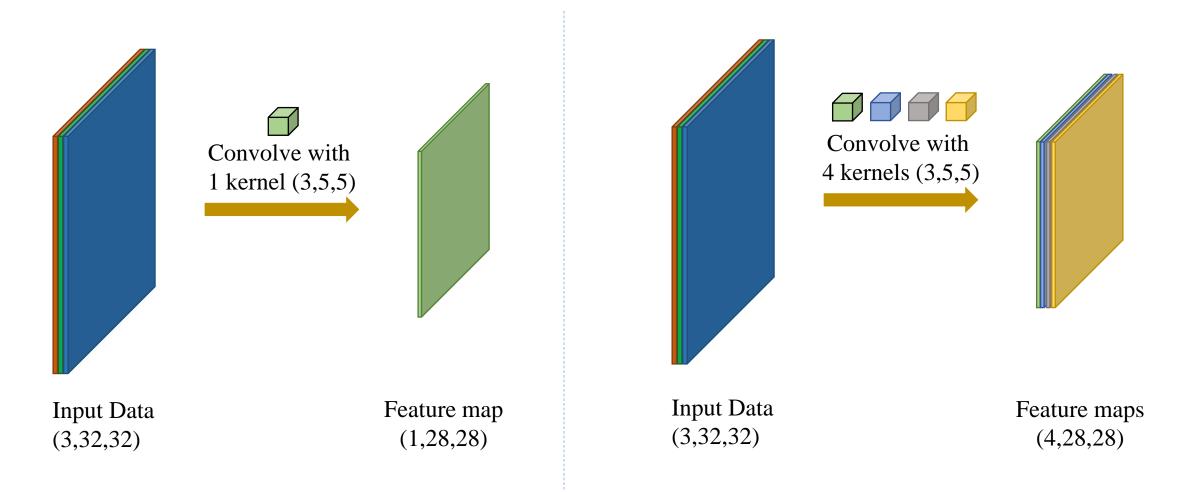


Data size =
$$5 \times 5$$

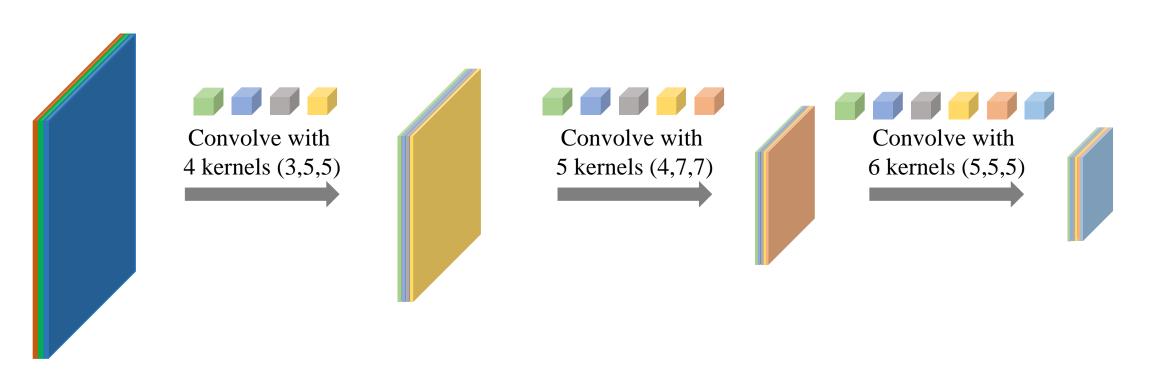
Kernel size =
$$3 \times 3$$

$$Stride = 2$$

Understand convolution



A stack of convolutions



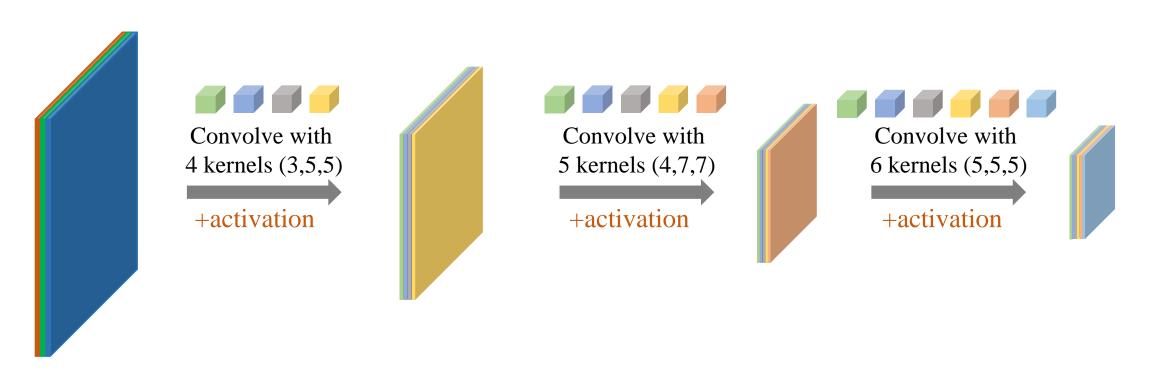
Input Data (3,32,32)

Feature maps (4,28,28)

Feature maps (5,22,22)

Feature maps (6,18,18)

A stack of pairs of convolution+activation



Input Data (3,32,32)

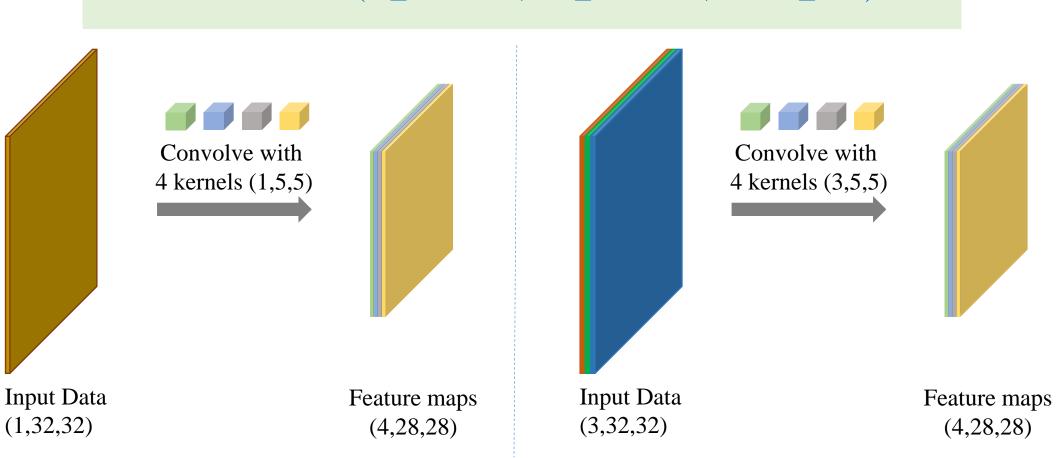
Feature maps (4,28,28)

Feature maps (5,22,22)

Feature maps (6,18,18)

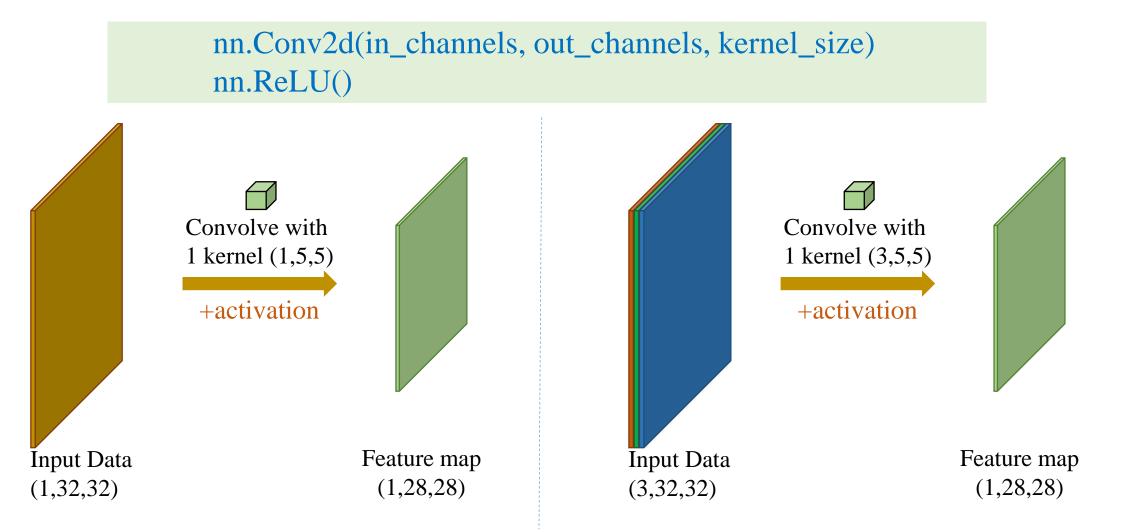
Convolution layer in PyTorch

nn.Conv2d(in_channels, out_channels, kernel_size)



Convolution layer in PyTorch

demo



Trouser

T-shirt





















Fashion-MNIST dataset

Pullover





















Resolution=28x28

Training set: 60000 samples

Testing set: 10000 samples

Coat

Sandal

Dress









































































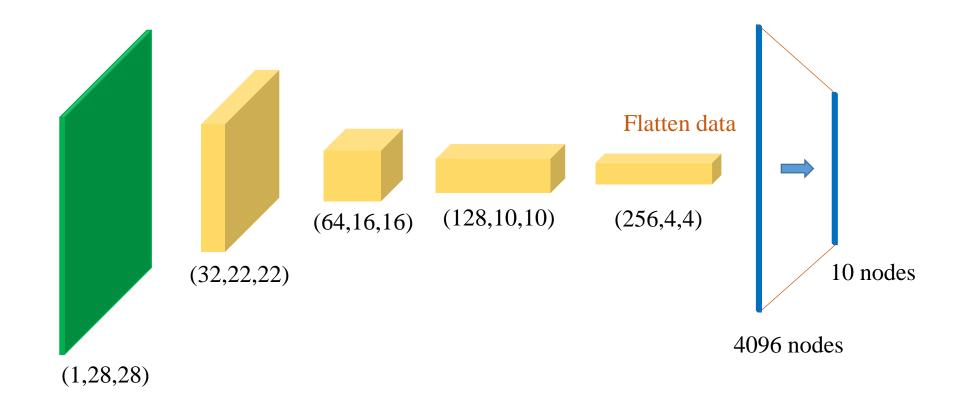






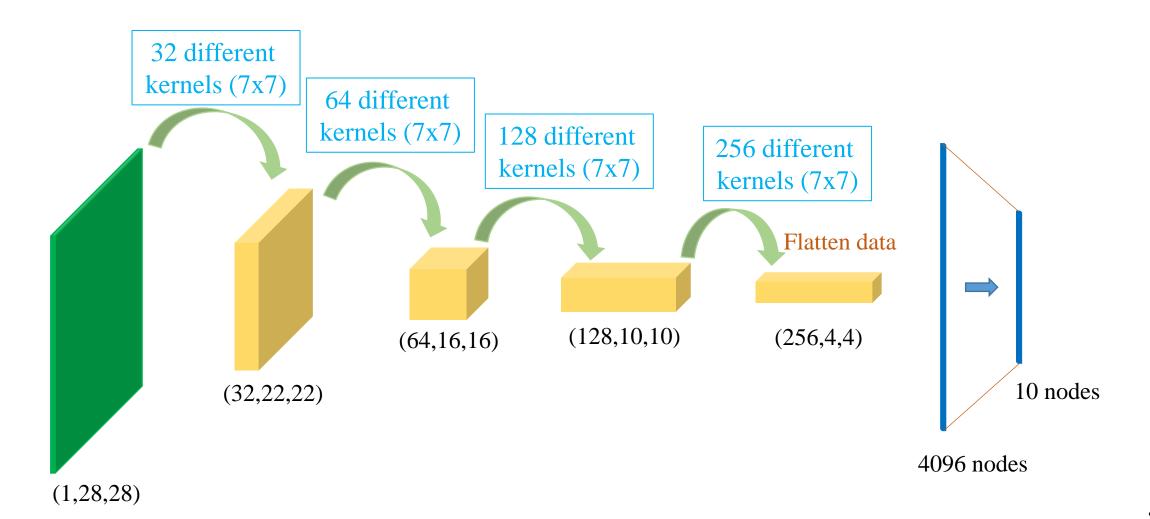


Apply for Fashion-MNIST dataset



Apply for Fashion-MNIST dataset

demo



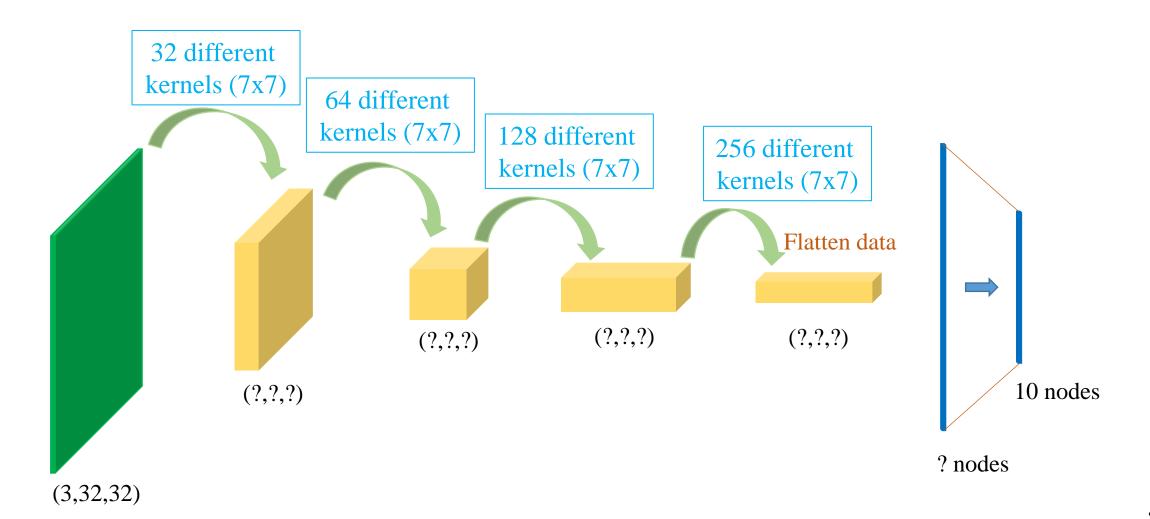
Simple Convolutional Neural Network

```
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
                                                           1,600
                               [-1, 32, 22, 22]
              ReLU-2
                               [-1, 32, 22, 22]
            Conv2d-3
                               [-1, 64, 16, 16]
                                                         100,416
                               [-1, 64, 16, 16]
              ReLU-4
            Conv2d-5
                              [-1, 128, 10, 10]
                                                         401,536
              ReLU-6
                              [-1, 128, 10, 10]
                                [-1, 256, 4, 4]
            Conv2d-7
                                                       1,605,888
              ReLU-8
                                [-1, 256, 4, 4]
           Flatten-9
                                     [-1, 4096]
           Linear-10
                                      [-1, 128]
                                                         524,416
             ReLU-11
                                      [-1, 128]
           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
```

Apply for Cifar-10 dataset

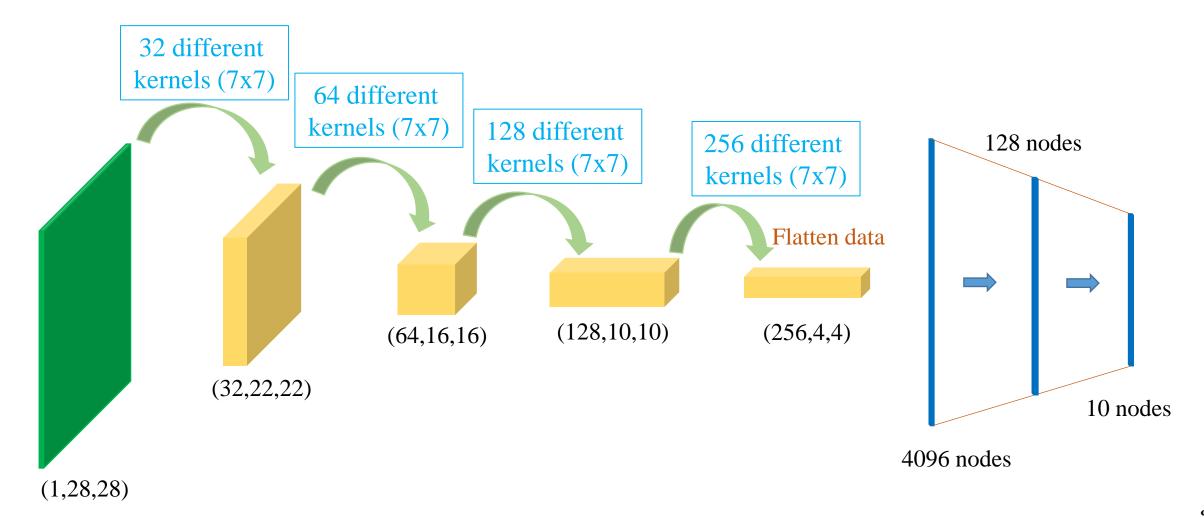
demo



Outline

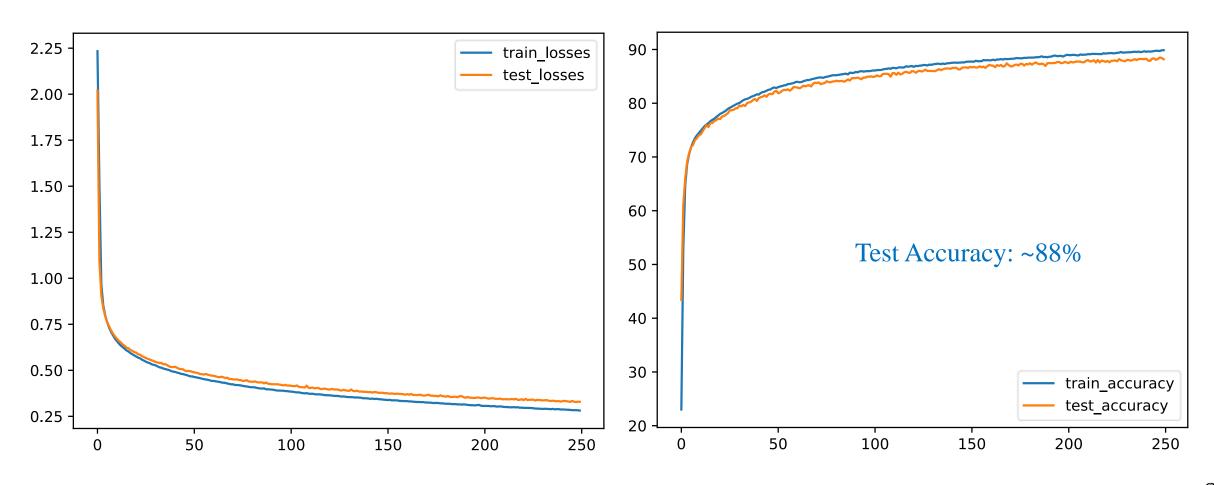
- > MLP Limitation
- > From MLP to CNN
- Feature Map Down-sampling
- > Some Examples
- > Application to Cifar10

Apply for Fashion-MNIST dataset: case 1

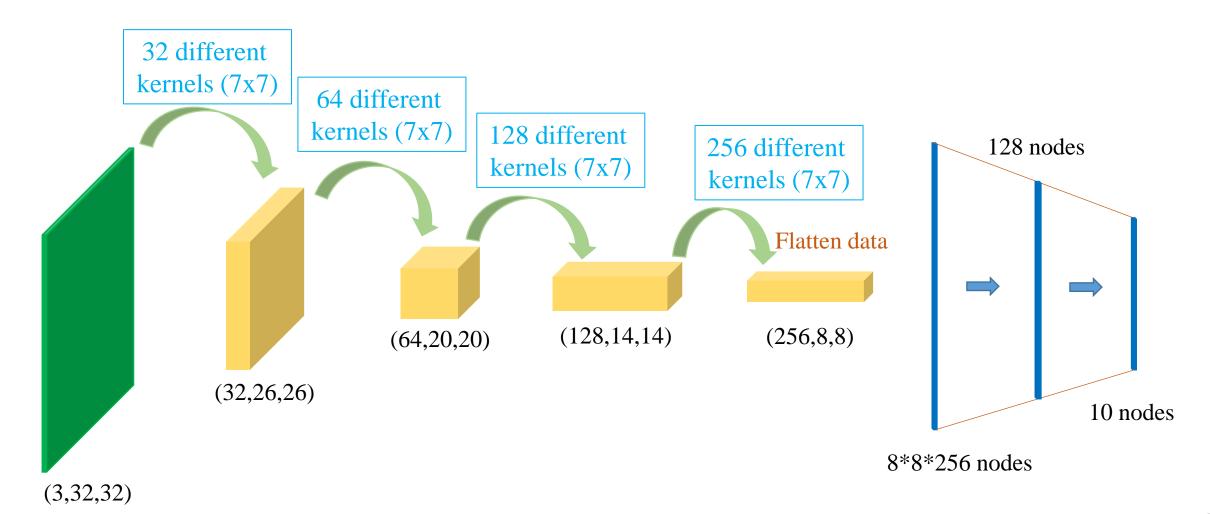


```
class CustomModel(nn.Module):
                                                             # Load FashionMNIST dataset
                                                             transform = Compose([ToTensor(),
    def init (self):
                                                                                 Normalize((0.5,),
        super(CustomModel, self). init ()
                                                                                          (0.5,))])
        self.conv1 = nn.Conv2d(1, 32, kernel size=7)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=7)
                                                             trainset = FashionMNIST(root='data',
        self.conv3 = nn.Conv2d(64, 128, kernel size=7)
                                                                                   train=True,
        self.conv4 = nn.Conv2d(128, 256, kernel_size=7)
                                                                                   download=True,
        self.flatten = nn.Flatten()
                                                                                   transform=transform)
        self.dense1 = nn.Linear(4*4*256, 128)
                                                             trainloader = DataLoader(trainset,
        self.dense2 = nn.Linear(128, 10)
                                                                                    batch_size=1024,
                                                                                    num workers=10,
        self.relu = nn.ReLU()
                                                                                     shuffle=True,
                                                                                     drop last=True)
    def forward(self, x):
        x = self.relu(self.conv1(x))
                                                             testset = FashionMNIST(root='data',
        x = self.relu(self.conv2(x))
                                                                                   train=False,
        x = self.relu(self.conv3(x))
                                                                                   download=True,
        x = self.relu(self.conv4(x))
                                                                                   transform=transform)
                                                             testloader = DataLoader(testset,
        x = self.flatten(x)
                                                                                   batch_size=1024,
        x = self.relu(self.dense1(x))
                                                                                   num workers=10,
        x = self.dense2(x)
                                                                                   shuffle=False)
        return x
                                                             # loss and optimizer
model = CustomModel()
                                                             criterion = nn.CrossEntropyLoss()
model = model.to(device)
                                                             optimizer = optim.Adam(model.parameters(), lr=1e-5)
```

Apply for Fashion-MNIST dataset: case 1

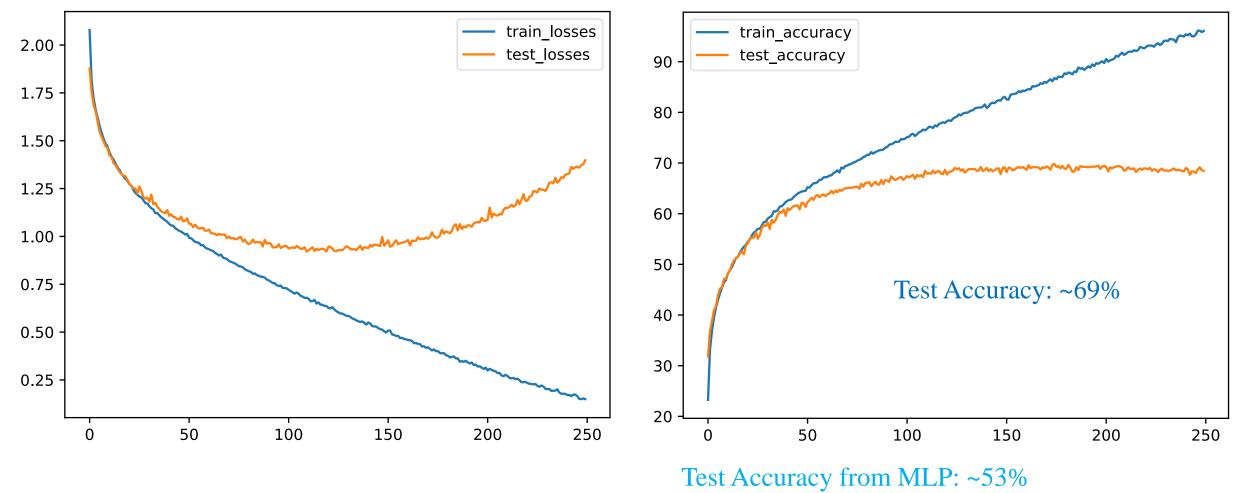


Apply for Cifar-10 dataset: case 2



```
class CustomModel(nn.Module):
                                                             # Load CIFAR10 dataset
                                                             transform = Compose([ToTensor(),
    def __init__(self):
                                                                                 Normalize((0.5,0.5, 0.5),
        super(CustomModel, self).__init__()
                                                                                          (0.5, 0.5, 0.5)))
        self.conv1 = nn.Conv2d(3, 32, kernel_size=7)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=7)
                                                             trainset = CIFAR10(root='data',
        self.conv3 = nn.Conv2d(64, 128, kernel_size=7)
                                                                               train=True,
        self.conv4 = nn.Conv2d(128, 256, kernel_size=7)
                                                                               download=True,
        self.flatten = nn.Flatten()
                                                                               transform=transform)
        self.dense1 = nn.Linear(8*8*256, 128)
                                                             trainloader = DataLoader(trainset,
        self.dense2 = nn.Linear(128, 10)
                                                                                     batch size=1024,
                                                                                     num workers=10,
        self.relu = nn.ReLU()
                                                                                     shuffle=True,
                                                                                     drop last=True)
    def forward(self, x):
        x = self.relu(self.conv1(x))
                                                             testset = CIFAR10(root='data',
        x = self.relu(self.conv2(x))
                                                                              train=False,
        x = self.relu(self.conv3(x))
                                                                              download=True,
        x = self.relu(self.conv4(x))
                                                                              transform=transform)
        x = self.flatten(x)
                                                             testloader = DataLoader(testset,
                                                                                    batch size=1024,
        x = self.relu(self.dense1(x))
                                                                                    num workers=10,
        x = self.dense2(x)
                                                                                    shuffle=False)
        return x
                                                             # loss and optimizer
model = CustomModel()
                                                             criterion = nn.CrossEntropyLoss()
model = model.to(device)
                                                             optimizer = optim.Adam(model.parameters(), lr=1e-5)
```

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

