컴퓨터비전 프로젝트

최종발표

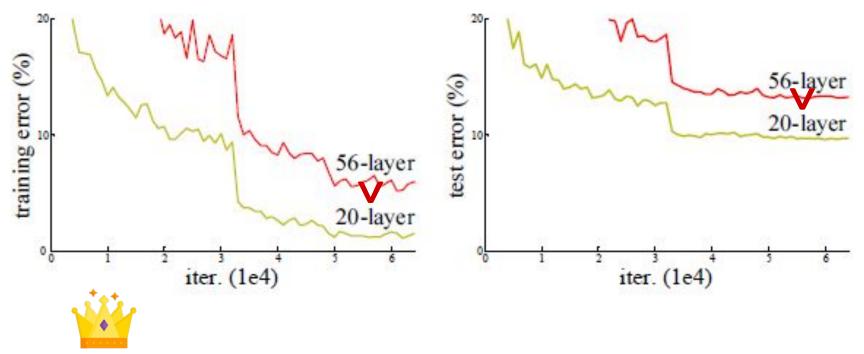
Deep Residual Learning for Image Recognition

2022.6.9

조 편성

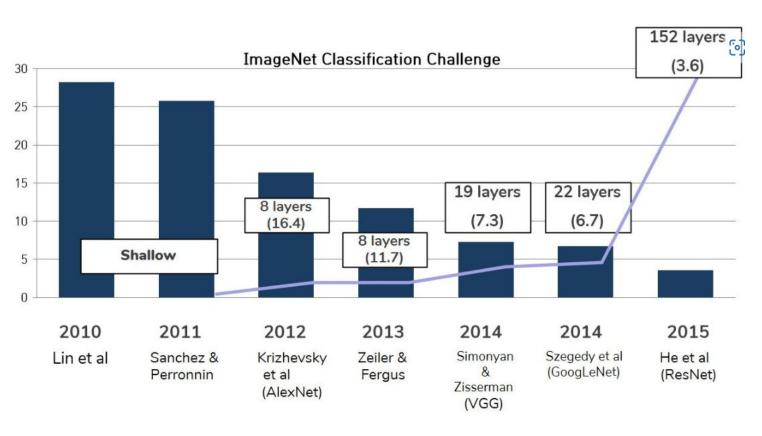
기계공학부 **20175806** 김현우 소프트웨어학부 **20192455** 이주영

1. 문제 제기/필요성



- 1st place on ILSVRC 2015 classification task
- easier to optimize, lower complexity

2. 내용

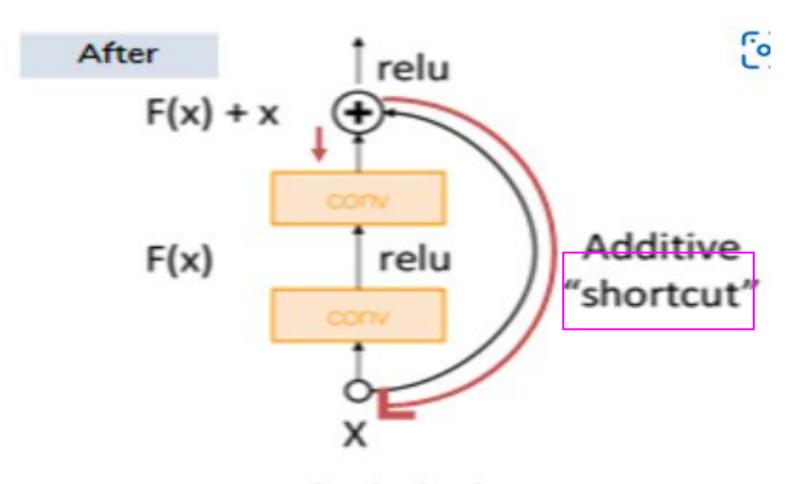


(1) Shortcut Connection

(2) Bottleneck Block

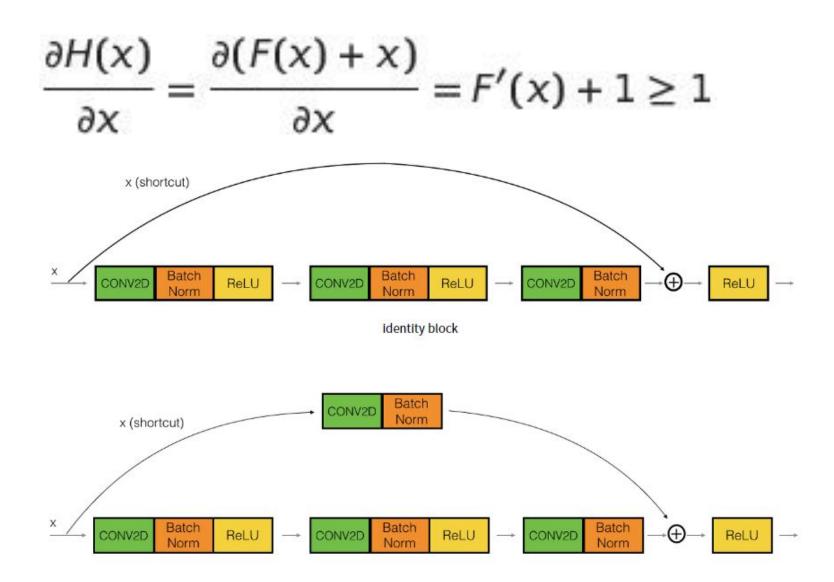
2. 내용 (1) Shortcut Connection

$$H(x) - x = F(x)$$



Residual Block

2. 내용 (1) Shortcut Connection



2. 내용 (1) Shortcut Connection

```
ResNetBasicBlock(
  (blocks): Sequential(
    (0): Sequential(
      (conv): Conv2dAuto(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
    (1): ReLU()
    (2): Sequential(
     (conv): Conv2dAuto(64, 64, kernel size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
  (shortcut): Sequential(
    (conv): Conv2d(32, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                                                   ResNetBottleNeckBlock(
                                                     (blocks): Sequential(
                                                       (0): Sequential(
                                                         (conv): Conv2dAuto(32, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
                                                         (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
                                                       (1): ReLU()
                                                       (2): Sequential(
                                                         (conv): Conv2dAuto(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
                                                         (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
                                                       (3): ReLU()
                                                       (4): Sequential(
                                                         (conv): Conv2dAuto(64, 256, kernel size=(1, 1), stride=(1, 1), bias=False)
                                                         (bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
                                                     (shortcut): Sequential(
                                                       (conv): Conv2d(32, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
                                                       (bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
```

2. 내용 (2) Bottleneck Block



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer	
conv1	112×112	7×7, 64, stride 2					
		3×3 max pool, stride 2					
conv2_x	56×56	$ \begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} $	$\left[\begin{array}{c} 3\times3,64\\ 3\times3,64 \end{array}\right]\times3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$ \begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3 $	$ \begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3 $	
conv3_x	28×28	$\left[\begin{array}{c} 3\times3, 128\\ 3\times3, 128 \end{array}\right] \times 2$	$\left[\begin{array}{c} 3\times3, 128\\ 3\times3, 128 \end{array}\right] \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$	
conv4_x	14×14	$\left[\begin{array}{c} 3\times3,256\\ 3\times3,256 \end{array}\right]\times2$	$\left[\begin{array}{c} 3\times3,256\\ 3\times3,256 \end{array}\right]\times6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$ \begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23 $	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$	
conv5_x	7×7	$\left[\begin{array}{c} 3\times3,512\\ 3\times3,512 \end{array}\right]\times2$	$\left[\begin{array}{c}3\times3,512\\3\times3,512\end{array}\right]\times3$	$ \begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3 $	$ \begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3 $	$ \begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3 $	
	1×1	average pool, 1000-d fc, softmax					
FLO	OPs	1.8×10^{9}	3.6×10^{9}	3.8×10^{9}	7.6×10^9	11.3×10^9	

```
return ResNet(in_channels, n_classes, block=ResNetBasicBlock, deepths=[2, 2, 2, 2])

def resnet34(in_channels, n_classes):
    return ResNet(in_channels, n_classes, block=ResNetBasicBlock, deepths=[3, 4, 6, 3])

def resnet50(in_channels, n_classes):
    return ResNet(in_channels, n_classes, block=ResNetBottleNeckBlock, deepths=[3, 4, 6, 3])

def resnet101(in_channels, n_classes):
    return ResNet(in_channels, n_classes, block=ResNetBottleNeckBlock, deepths=[3, 4, 23, 3])

def resnet152(in_channels, n_classes):
    return ResNet(in_channels, n_classes, block=ResNetBottleNeckBlock, deepths=[3, 8, 36, 3])
```

def resnet18(in_channels, n_classes):

```
def resnet18(in_channels, n_classes):
    return ResNet(in_channels, n_classes, block=ResNetBasicBlock, deepths=[2, 2, 2, 2])
def resnet50(in_channels, n_classes):
    return ResNet (in_channels, n_classes, block=ResNetBottleNeckBlock, deepths=[3, 4, 6, 3])
class ResNet(nn.Module):
   def __init__(self, in_channels, n_classes, *args, **kwargs);
       super().__init__()
       self.encoder = ResNetEncoder(in_channels, *args, **kwargs)
       self.decoder = ResnetDecoder(self.encoder.blocks[-1].blocks[-1].expanded_channels, n_classes)
   def forward(self, x):
       x = self.encoder(x)
       x = self.decoder(x)
       return x
```

```
class ResNetBasicBlock(ResNetResidualBlock):
    expansion = 1
    def init (self, in channels, out channels, activation=nn.ReLU, *args, **kwargs):
        super(). init (in channels, out channels, *args, **kwargs)
       self.blocks = nn.Sequential(
            conv bn(self.in channels, self.out channels, conv=self.conv, bias=False, stride=self.downsampling)
           activation(),
           conv bn(self.out channels, self.expanded channels, conv=self.conv, bias=False),
class ResNetBottleNeckBlock(ResNetResidualBlock):
    expansion = 4
    def init (self, in channels, out channels, activation=nn.ReLU, *args, **kwargs):
        super(). init (in channels, out channels, expansion=4, *args, **kwargs)
        self.blocks = nn.Sequential(
           conv bn(self.in channels, self.out channels, self.conv, kernel size=1)
             activation(),
             conv bn(self.out channels, self.out channels, self.conv, kernel size 3, stride=self.downsampling
             activation(),
             conv_bn(self.out_channels, self.expanded_channels, self.conv, kernel_size=1
```

```
train_dataset = torchvision.datasets.MNIST(root='./data', train=True, download=True, transform=transform_train)
```

train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform_train)

34-layer residual

def resnet34(in_channels, n_classes):
 return ResNet(in_channels, n_classes, block=ResNetBasicBlock, deepths=[3, 4, 6, 3])

34-layer

from torchsummary import summary
model = resnet34(1, 10)
summary(model.cuda(), (1, 28, 28))

$\left[\begin{array}{c}3\times3,64\\3\times3,64\end{array}\right]$	×3
$\left[\begin{array}{c}3\times3,128\\3\times3,128\end{array}\right]$	×4
$\left[\begin{array}{c}3\times3,256\\3\times3,256\end{array}\right]$	×6
$\begin{bmatrix} 3\times3,512\\ 3\times3,512 \end{bmatrix}$	×3

 3.6×10^{9}

ave

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 14, 14]	3,136
BatchNorm2d-2	[-1, 64, 14, 14]	128
ReLU-3	[-1, 64, 14, 14]	0
MaxPool2d-4	[-1, 64, 7, 7]	0
Conv2dAuto-5	[-1, 64, 7, 7]	36,864
BatchNorm2d-6	[-1, 64, 7, 7]	128
ReLU-7	[-1, 64, 7, 7]	0
Conv2dAuto-8	[-1, 64, 7, 7]	36,864
BatchNorm2d-9	[-1, 64, 7, 7]	128
ResNetBasicBlock-10	[-1, 64, 7, 7]	0
Conv2dAuto-11	[-1, 64, 7, 7]	36,864
BatchNorm2d-12	[-1, 64, 7, 7]	128
ReLU-13	[-1, 64, 7, 7]	0
Conv2dAuto-14	[-1, 64, 7, 7]	36,864
BatchNorm2d-15	[-1, 64, 7, 7]	128
ResNetBasicBlock-16	[-1, 64, 7, 7]	0

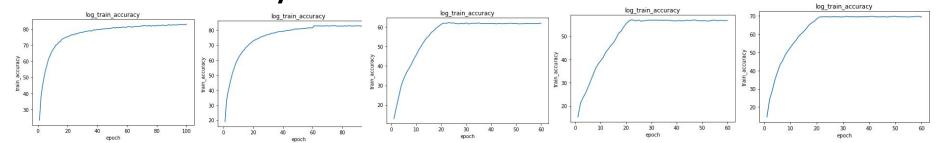
7x7 conv, 64, /2	
pool, /2	
3x3 cpnv, 64	1
3x3 conv, 64	ノ
3x3 conv., 64	-
3x3 conv, 64	_)
3x3 conv, 64	
3x3 conv, 64	
3x3 conv, 128, /2	
3x3 conv, 128	
3x3 conv, 128	
3x3 conv, 128)
3x3 conv, 128	\leq
+	3
3x3 conv, 128	<
3x3 conv, 128	7
3x3 conv, 128	
3x3 conv, 256, /2	
3x3 conv, 256	
3x3 conv, 256	1
3x3 conv, 256	
3x3 conv, 256	-
3x3 cpnv, 256	
3x3 conv, 256	
3x3 conv, 256)
3x3 cpnv, 256	
3x3 conv, 256)
3x3 conv, 256	\leq
3x3 conv, 256)
3x3 conv, 512, /2	
+	3
3x3 conv, 512	
3x3 conv, 512	3
3x3 conv, 512	-
3x3 conv, 512	1
3x3 conv, 512	
avg pool	
fc 1000	

Lavor (type)	Output Bhase	Paran #
Conv2d-1	[-1, 64, 14, 14]	3,188
BatchNorm2d-2 Dat (I=9	[-1, 84, 14, 14] [-1, 84, 14, 14] [-1, 84, 14, 14]	128
Na-Pag 2d-4	[-1, 84, 7, 7]	0
Conv2dhuto-6 BatoHNorm2d-8 Bal.U-7	[-1, 64, 14, 14] [-1, 64, 7, 7] [-1, 64, 7, 7] [-1, 64, 7, 7]	88,884 128
RaLU-7	[-1, 8d, 7, 7] [-1, 8d, 7, 7]	0 000
BatoHlorm2d=8	[-1, 84, 7, 7]	125
Position Bas in Black = 10	(-1, 84, 7, 7) (-1, 84, 7, 7)	98 884
BatchNorm2d-12	[-1, 81, 7, 7]	128
ReLU-18 Conv2dAuto-14	[-1, 84, 7, 7] [-1, 84, 7, 7]	0 98,884
Batchlare2d=16 CoolletBasicStock=18	[-1, 8d, 7, 7] [-5, 8d, 7, 7] [-6, 8d, 7, 7]	128
Conv2dHuto=17	[-1, 81, 7, 7]	38.884
BatohNorm2d-18 RoLU-18	(-1, 84, 7, 7) (-1, 84, 7, 7)	128 0
Conv2dhuto-20	1-1, 84, 7, 71	38.884
Conv2dAuto-20 BatchNorm2d-21 PooNotBas i oB I pok-22	[-1, 84, 7, 7] [-1, 84, 7, 7]	128
Realie tLauer - 28 Conv2d - 24	[-1, 84, 7, 7] [-1, 84, 7, 7] [-1, 128, 4, 4]	0 0 8,192
BatchNorm2d-25		258 78,728
Conv2dAuto-28	[-1, 128, 4, 4]	78,728 258
ReLU-28 Conv2dAuto-29	T-1, 128, 4, 41	.0
Conv2dAuto-28 BatakNorm2d-80	[-1, 128, 4, 4] [-1, 128, 4, 4] [-1, 128, 4, 4] [-1, 128, 4, 4]	147,458 258
RealierBasisBlook=81	[-1, 128, 4, 4] [-1, 128, 4, 4]	147.458
BatchNorm2d-88	[-1, 128, 4, 4]	258
ReLU-84 Deeu/2dhutee/85	[-1, 128, 4, 4] [-1, 128, 4, 4]	147.458
BatchNorm2d=88	[-1, 128, 4, 4] [-1, 128, 4, 4]	258
Conv2d*uto=88	[-1: 128: 4: A]	147,458
BatchNorm2d-88	[-1, 128, 4, 4]	258
Conv2dAutor41	[-1, 128, 4, A]	147,458
BatchNorm2d=42 Cooking Region Laborate		258 0
Conv2dhuto-Ail	[-1, 128, 4, 4] [-1, 128, 4, 4] [-1, 128, 4, 4]	147,458
BatchNorm2d-46 ReLU-48	[-1, 128, 2, 2]	258
Conv2dAuto=47 SatoMore2d=48	[-1, 128, 4, 4] [-1, 128, 4, 4] [-1, 128, 4, 4]	147,458
Goother Bas in Black-49	[-1, 128, \(\Delta \), \(\Delta \)] [-1, 128, \(\Delta \), \(\Delta \)] [-1, 128, \(\Delta \), \(\Delta \)]	0
RealletLaver =50 Conv2d=51	[-1, 128, 4, 4] [-1, 268, 2, 2]	82,765
BatahNarm2d-52	[-1, 258, 2, 2] [-1, 258, 2, 2] [-1, 258, 2, 2]	512
Comy2dAuto=68 BatchNorm2d=64	[-1, 258, 2, 2] [-1, 258, 2, 2]	512
ReLU-55	[-1, 258, 2, 2] [-1, 258, 2, 2]	D 559 524
Batohllarm2d-57 PeolletBasio8 look-58	[-1, 258, 2, 2]	612 0
Conv2dFuto=68	[-1, 258, 2, 2] [-1, 258, 2, 2]	CCD 07/
BatoHNarm2d=80 ReLU=81	[-1, 258, 2, 2] [-1, 258, 2, 2]	612 0
Conv2dAuto-82	[-1, 258, 2, 2] [-1, 258, 2, 2] [-1, 258, 2, 2]	659 924
BatchNorm2d=83 RecitetBasic8 look=84 Cany2dAuto=86	[-1, 258, 2, 2] [-1, 258, 2, 2] [-1, 258, 2, 2]	612 0
Conv2dAuto=86 BatchNore2d=88	[-1, 258, 2, 2] [-1, 258, 2, 2]	659,824 612
ReLU-87	[-1, 258, 2, 2]	
Conv2dhuto-88 BatchNorm2d-88	[-1, 258, 2, 2] [-1, 258, 2, 2]	612
Position Service 21	[-1, 258, 2, 2] [-1, 258, 2, 2]	0 689.824
BatchNorm2d-72 Bat U-78	[-1, 258, 2, 2] [-1, 258, 2, 2]	689,824 612 0
ReLU-78 Conv2dFuto-74	[-1, 258, 2, 2]	688,824
Batahlar #2d-76 Pagli at Basi a Bilagk #78	[-1, 258, 2, 2] [-1, 258, 2, 2]	612 0
Conv2dAuto=77	[-1, 258, 2, 2] [-1, 258, 2, 2] [-1, 258, 2, 2]	688,824
BatohNorm2d-78 ReLU-79	[-1, 258, 2, 2] [-1, 258, 2, 2]	689,824 612 0
Conv2dAuton80 Ratablece2de81	f-1 25A 2 21	
ReelletBasicBlack+82	[-1, 258, 2, 2] [-1, 258, 2, 2]	612 0
Conv2dFutor88 Retablisce2de84		689,824
ReLU-86		612 0 689,824 612 0
Conv2dAuto=68 BatchNorm2d=67		688,824 612
Reallet Bas (of Look-68 Reallet Laver-69	[-1, 288, 2, 2]	0
Conv2d-90	[-1, 512, 1, 1]	
BatchNorm2d=81 Conv2dAuto=82	[-1, 512, 1, 1] [-1, 512, 1, 1]	181,072 1,022 1,179,848
BatohNorm2d-88	[-1, 512, 1, 1] [-1, 512, 1, 1]	1.024
Conv2dAuto=95	[-1, 512, 1, 1] [-1, 512, 1, 1]	2,359,298
SatoHlarm2d-56 SeeNetBasisBisok-87	[-1, 512, 1, 1]	1,024
Conv2dAuto-98	F-1: 512: 1: 11	2,358,298
BatahNorm2d-89 RaLU-100	[-1, 512, 1, 1] [-1, 512, 1, 1]	1,024
Dany2dhuta=101	[-1, 512, 1, 1] [-1, 512, 1, 1] [-1, 512, 1, 1] [-1, 512, 1, 1] [-1, 512, 1, 1]	2,859,298 1,024
ReclietBasisBissk=108		
Canv2dhuta=10d BataHlare2d=106	[-1, 512, 1, 1] [-1, 512, 1, 1] [-1, 512, 1, 1]	2,859,298 1,024
Gel U=108	[-1, 512, 1, 1]	
Conv2dAute=107 BatchNore2d=108 GeoNetBasic8/cok=109	[-1, 512, 1, 1] [-1, 512, 1, 1]	2,859,298 1,024
	[-1, 512, 1, 1] [-1, 512, 1, 1] [-1, 512, 1, 1] [-1, 512, 1, 1] [-1, 512, 1, 1]	0
Resiletteneder-111 Adapt i vehveRee 2d-112 Linear-118 ResnetCeceder-114	[-1, 512, 1, 1] [-1, 512, 1, 1] [-1, 512, 1, 1]	2,359,256 1,024 0 0
Linear=118	(-1, 512, 1, 1) [-1, 10]	5,180 0
ResnetDecader=114	[-1, 10]	0

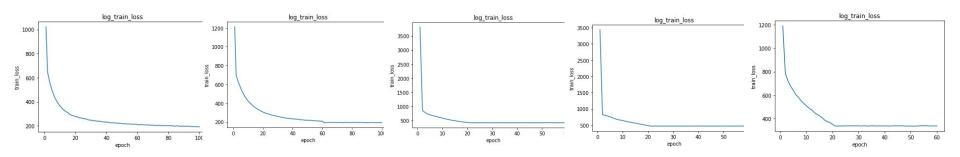
3. 구현 내용 및 결과 분석 - CIFAR-10 dataset

Google Colab

Train Accuracy

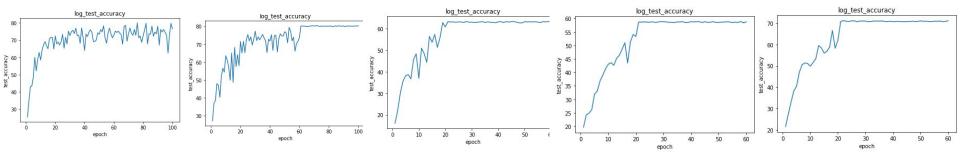


Train Loss

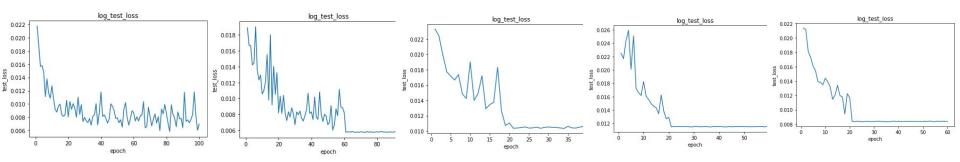


3. 구현 내용 및 결과 분석 - CIFAR-10 dataset

Test Accuracy

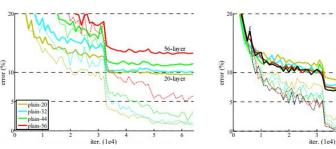


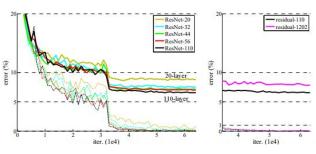
Test Loss



4. 개선 내용 - CIFAR-10 dataset

me	error (%)		
Maxo	9.38		
NIN	8.81		
DSI	8.22		
	# params		
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	$7.54 (7.72 \pm 0.16)$
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	6.43 (6.61±0.16)
ResNet	1202	19.4M	7.93

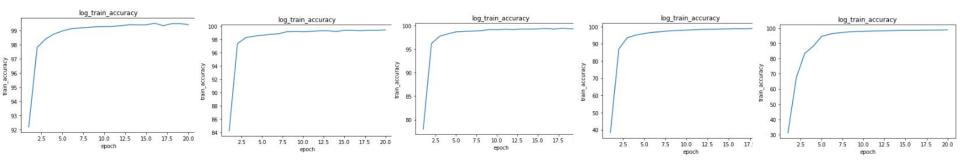




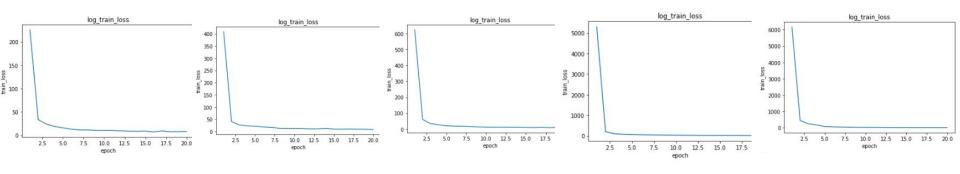
	18	34	50	101	152
train accuracy	83.022	82.722	61.974	56.776	69.376
train loss	193.057136774 0631	194.398092538 11836	416.790073037 1475]	472.024994432 9262	337.761715769 76776
test accuracy	76.69	80.28	63.21	58.76	71.05
test loss	0.00696204485 297203	0.00580733307 0039749	0.01044059162 1398926	0.01148222939 3720627	0.00833706557 1546554

4. 개선 내용 - MNIST dataset

Train Accuracy

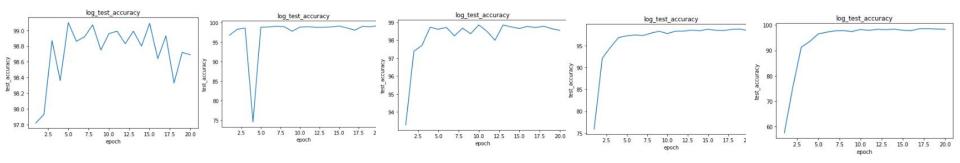


Train Loss

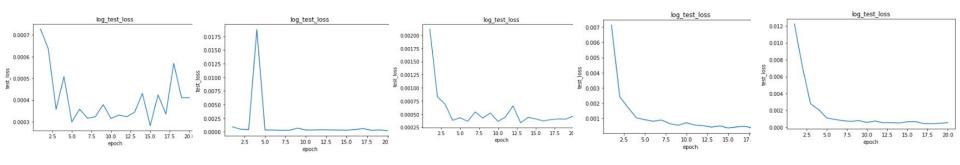


4. 개선 내용 - MNIST dataset

Train Accuracy



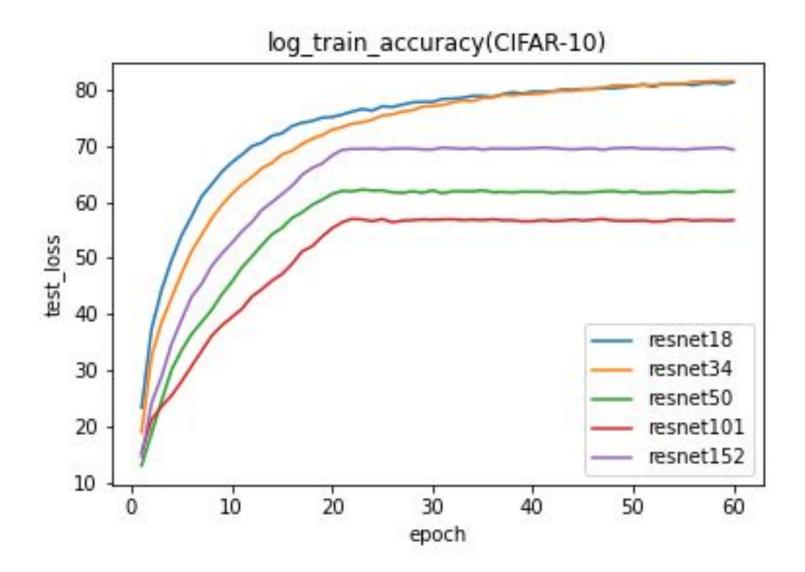
Test Loss



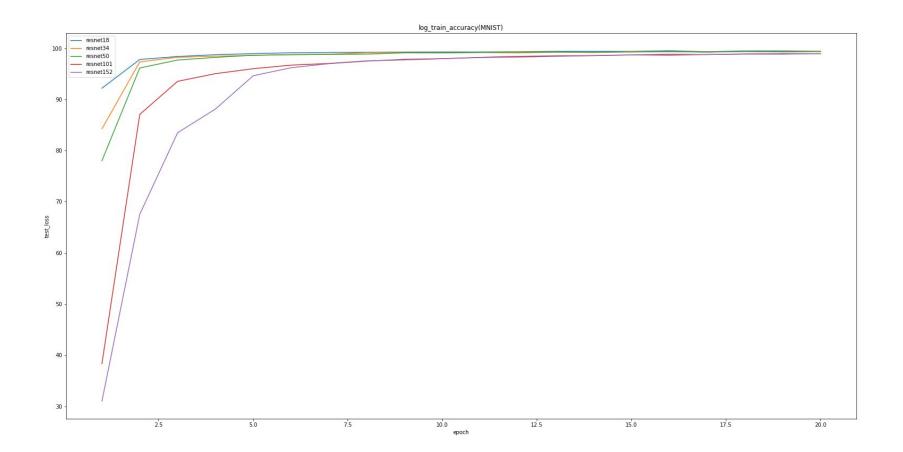
4. 개선 내용 - MNIST dataset

	18	34	50	101	152
train accuracy	99.403333333333 4	99.425	99.363333333333333333333333333333333333	98.945	98.965
train loss	8.20384476095205 2	8.52387726635788 6	9.16429775120923 3	16.1861739578889 7]	15.9279429329326 38
test accuracy	98.69	99.24	98.55	98.37	98.29
test loss	0.00041076599613 87981	0.00023698459719 59876	0.00046397476311 39471	0.00051472630695 20705]	0.00053919836082 78679

4. 개선 내용



4. 개선 내용



5. 결론

- layer 및 block을 정의하고 조합하는 방법 학습
- torchvision.datasets 사용 방법 학습
- Matplotlib 사용 Accuracy & Loss 그래프 작성 방법 학습
- 무작위적 layer 추가가 아닌 ResNet 모델의 깊이별 결과 비교
- 논문 구현 경험
- 논문 결과 검증 경험
- Pytorch 이해 경험

6. 참고문헌

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Microsoft Research
 Deep Residual Learning for Image Recognition
 CVPR 2016
- <u>Deep-Learning-Paper-Review-and-Practice/ResNet18_CIFAR10_Train.ipynb at master · ndb796/Deep-Learning-Paper-Review-and-Practice (github.com)</u>
- <u>Deep-Learning-Paper-Review-and-Practice/ResNet18_MNIST_Train.ipynb at master ndb796/Deep-Learning-Paper-Review-and-Practice (github.com)</u>
- <u>FrancescoSaverioZuppichini/ResNet: Clean, scalable and easy to use ResNet implementation in Pytorch (github.com)</u>
- ResNet에 관한 이해 : 네이버 블로그 (naver.com)
- ResNet에 대하여 (velog.io)