Dacon Computer Vision Contest
Private 4th

https://github.com/newave986

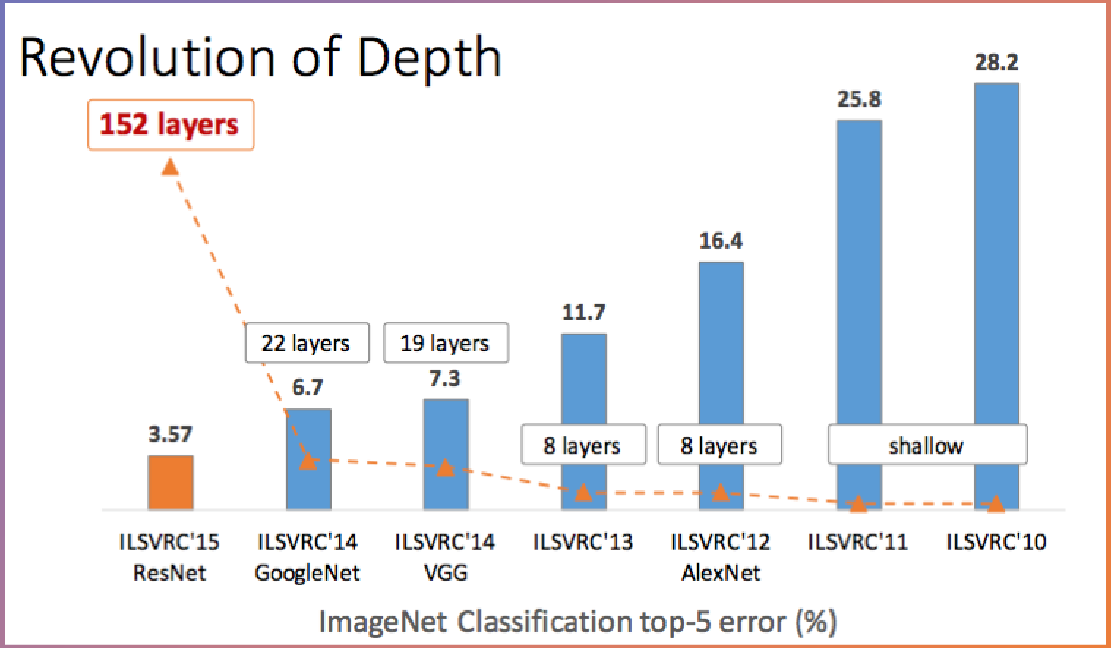
Main Algorithm

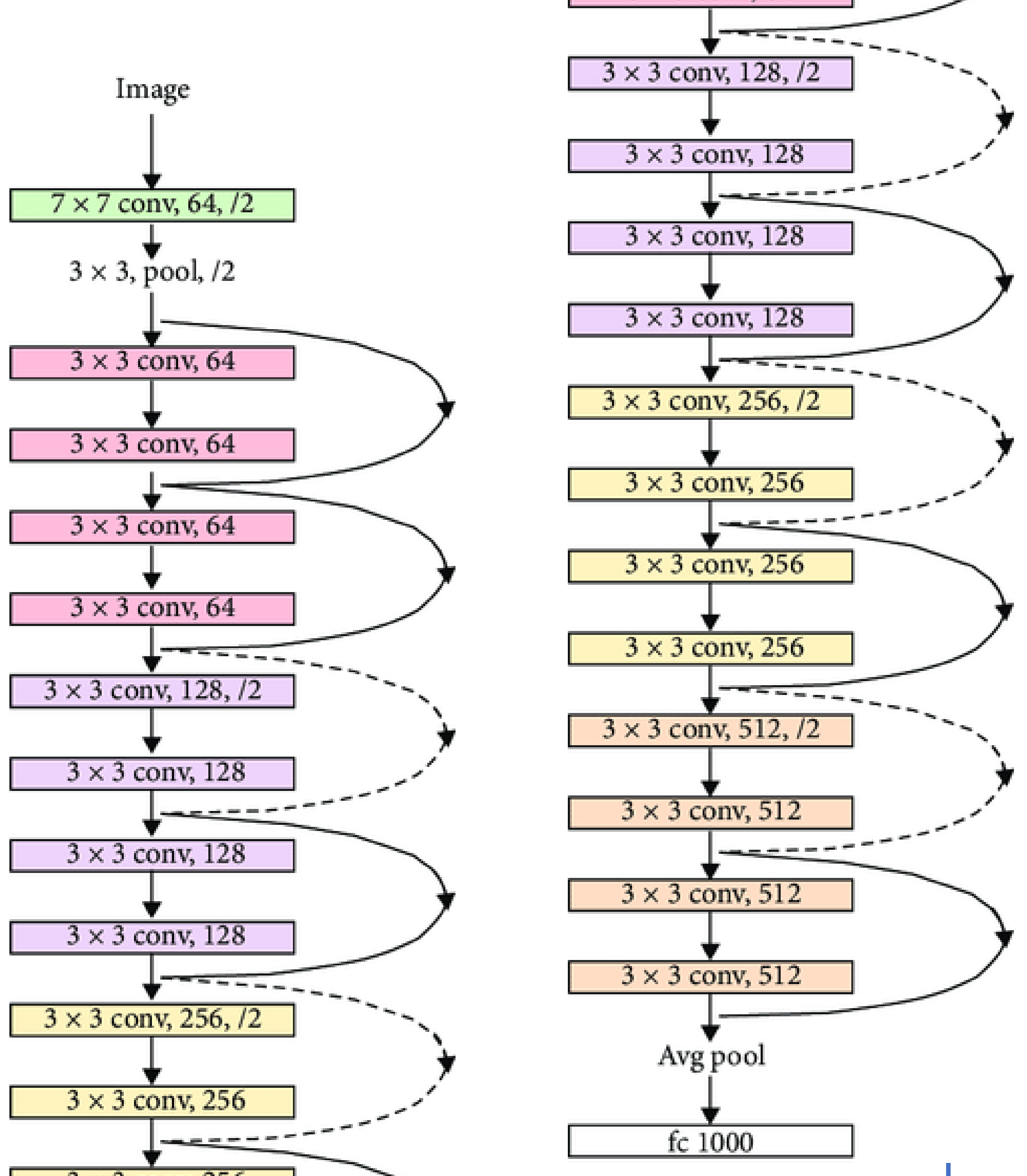
ResNet
CutMix
SWA
Label Smoothing

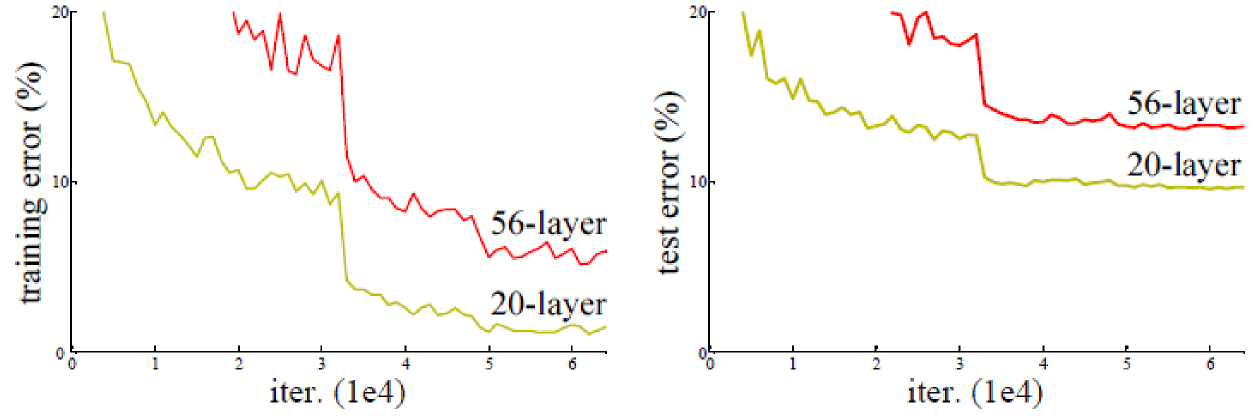
"CV 대회에서 순위권에 들기 위해서는
Ensemble하기 위해
다양한 Model을 제작하는 것이 중요함."

ResNet

https://arxiv.org/abs/1512.03385







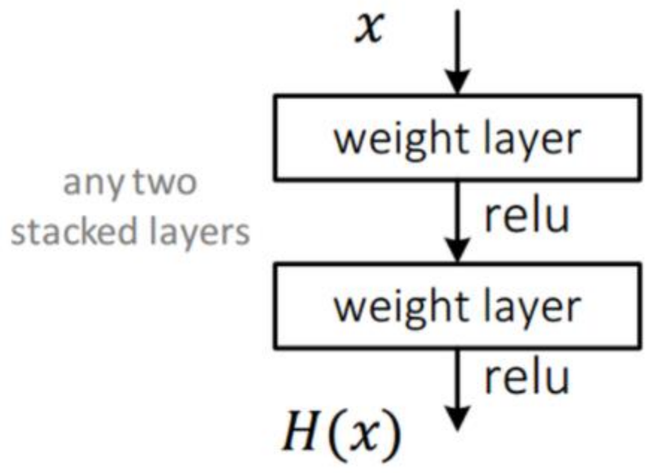
Deep한 모델이 무조건 성능이 좋은가?

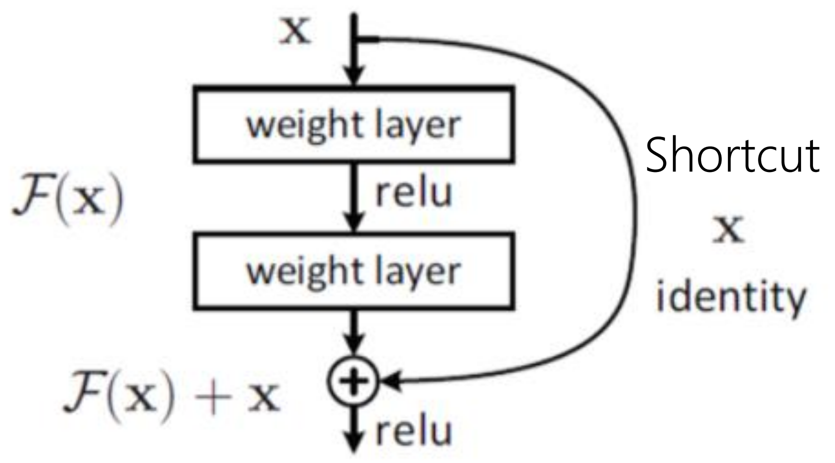
- No.

(1) Vanishing/Exploding Gradient

(2) Parameter 수↑ 에러 ↑

"왜 많이 깊어지는데 오류가 증가할까?"라는 문제의식에서 출발.




x
weight layer
relu
weight layer
$\mathcal{F}(\mathrm{x})$
$\mathcal{F}(\mathrm{x}) + \mathrm{x}$
relu
Shortcut
x
identity

Residual Block

Minimize H(x) = F(x)+x

X is fixed value, so Minimize F(x)

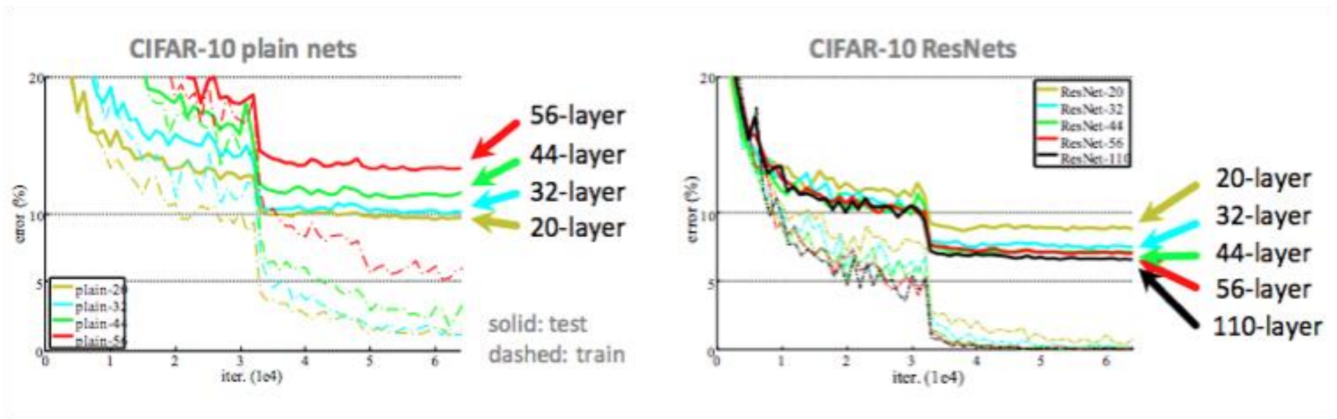
F(x) = H(x) – x, so Minimize H(x)-x and it is **Residual**.

Shortcut,
덧셈 연산 추가되는 것 외에 큰 차이 없음

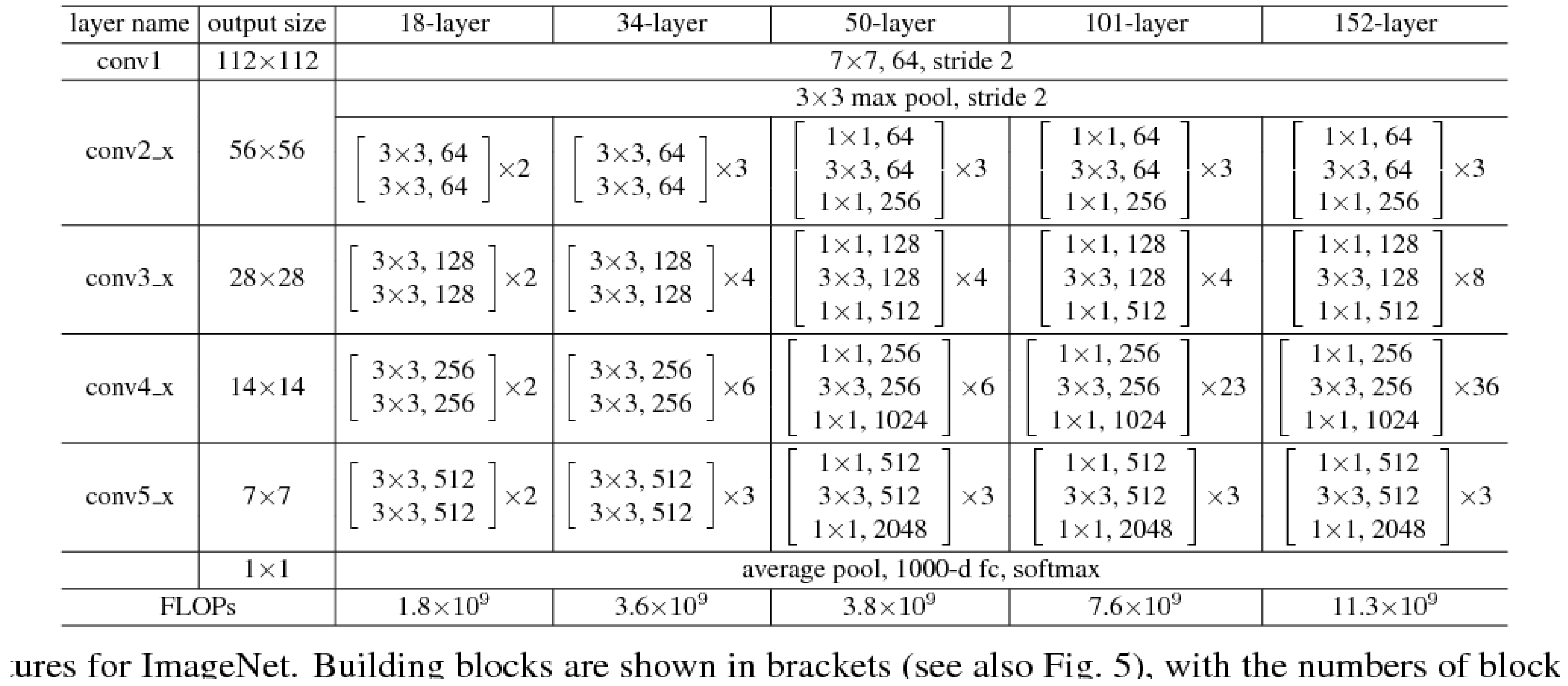
입력과 같은 x가 출력에 연결됨
- Parameter 수에 영향 X

Layer *건너 뛰면서* 입출력 연결
- forward, backward path 단순해짐

깊은 망도 쉽게 최적화 가능



- **ResNet18**
  - PreActBlock
- **ResNet34**
  - BasicBlock
- **ResNet50**
  - Bottleneck

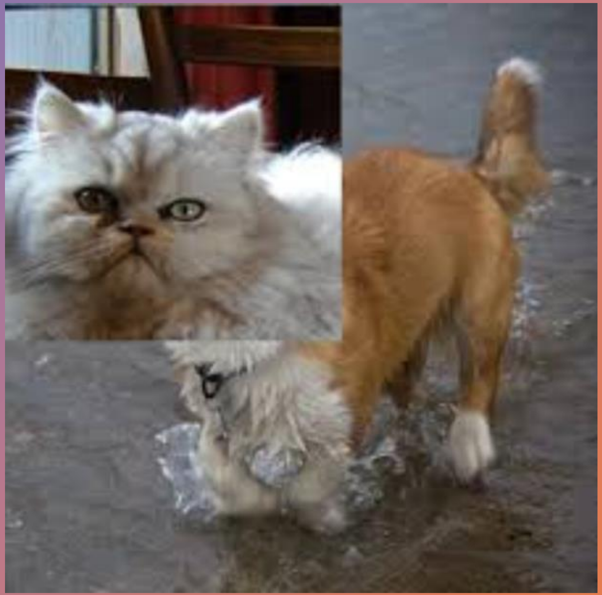


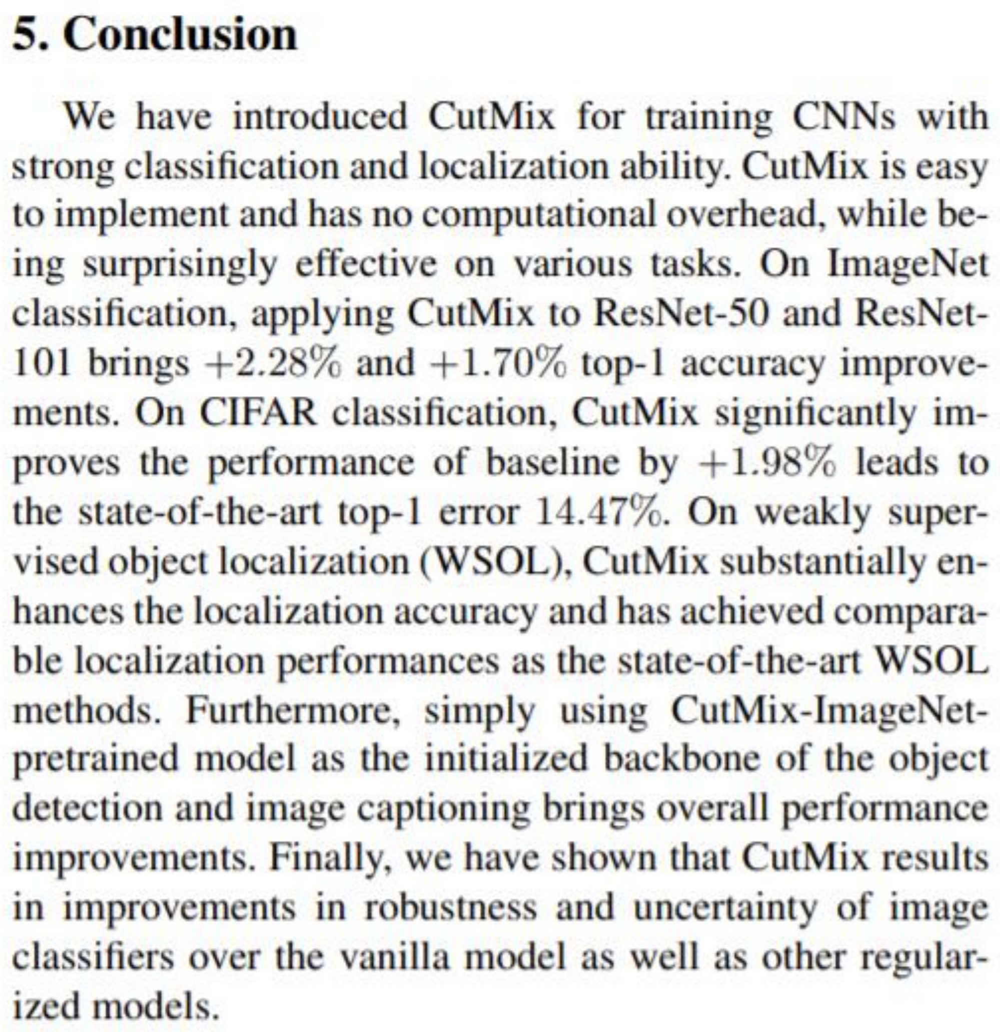
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

CutMix

https://arxiv.org/abs/1905.04899





5. Conclusion

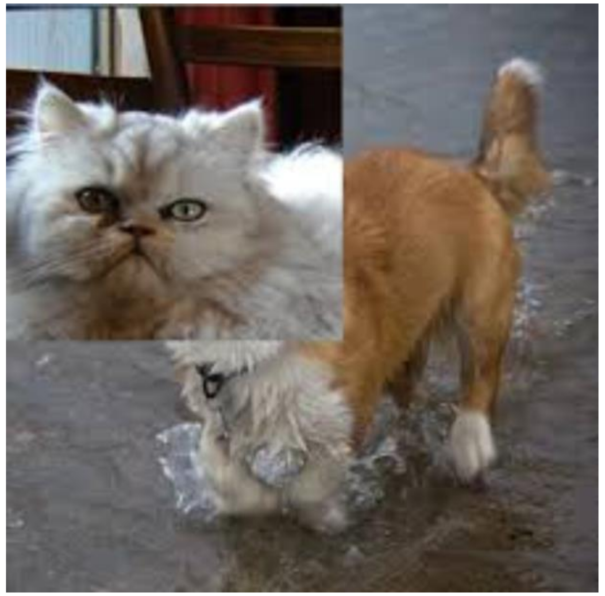
We have introduced CutMix for training CNNs with strong classification and localization ability. CutMix is easy to implement and has no computational overhead, while being surprisingly effective on various tasks. On ImageNet classification, applying CutMix to ResNet-50 and ResNet-101 brings +2.28% and +1.70% top-1 accuracy improvements. On CIFAR classification, CutMix significantly improves the performance of baseline by +1.98% leads to the state-of-the-art top-1 error 14.47%. On weakly supervised object localization (WSOL), CutMix substantially enhances the localization accuracy and has achieved comparable localization performances as the state-of-the-art WSOL methods. Furthermore, simply using CutMix-ImageNet-pretrained model as the initialized backbone of the object detection and image captioning brings overall performance improvements. Finally, we have shown that CutMix results in improvements in robustness and uncertainty of image classifiers over the vanilla model as well as other regularized models.

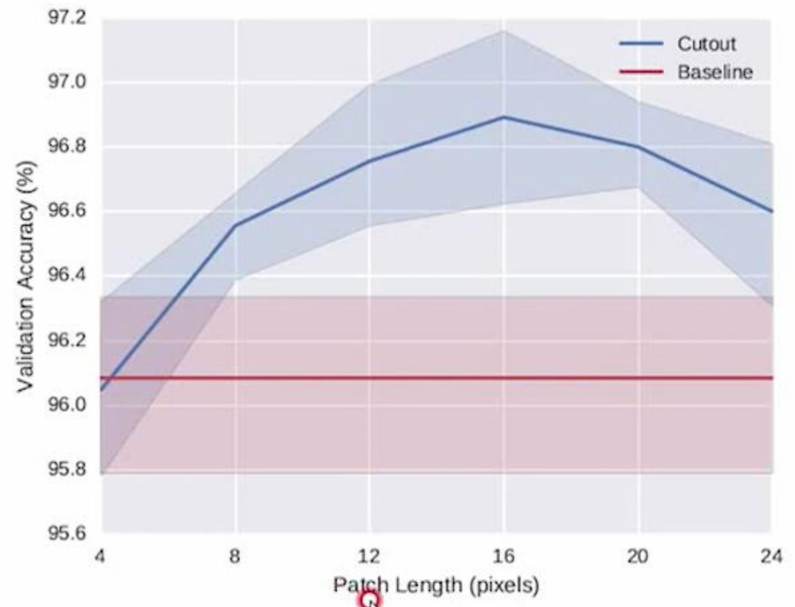
CutMix

- **중요한 피처**(연속된 영역) 제거
  Discriminative Parts ↓
  전체적 이미지의 활용도 ↑
    - **전체적 문맥**의 활용 ↑
    - Overfitting 방지 효과

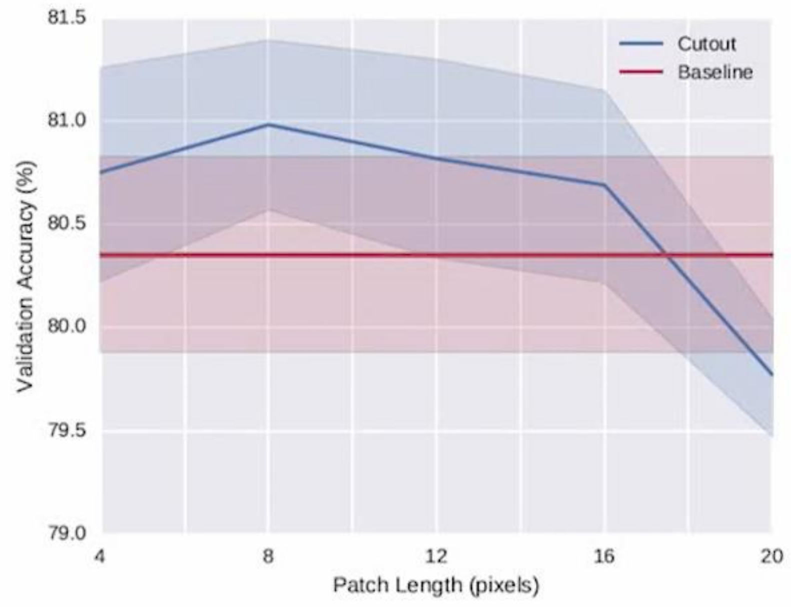
- 전체 Context가 고려될 수 있도록 이미지 전체 중점으로

- Dropout과 비슷한 방식이지만 차이 존재
  1) Cutmix: Input layer에서만 피처 제거
     Dropout보다 noise 발생 적음. Argumentation에 더 가까움.
  2) 무작위 영역 제거가 아닌 연속적 영역 제거





(a) CIFAR-10



(b) CIFAR-100

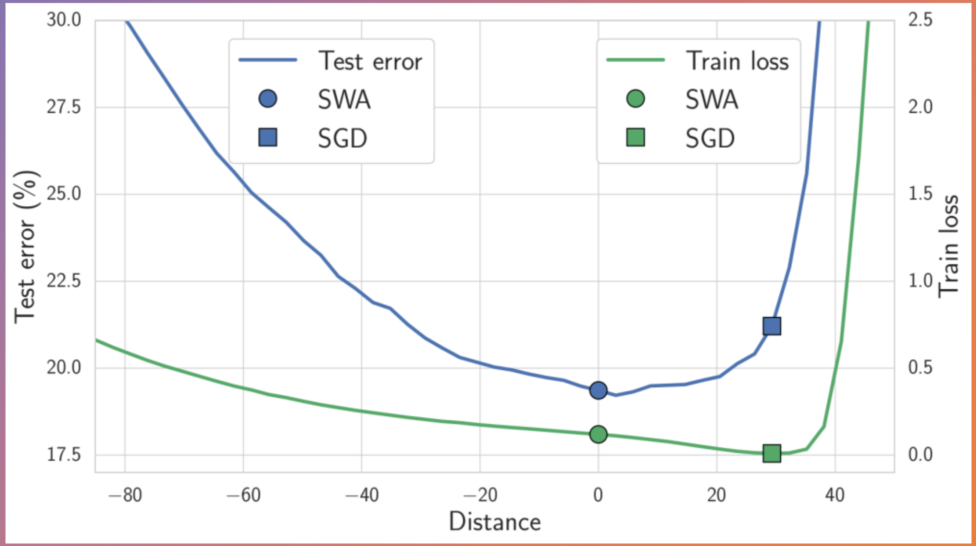
Figure 3: Cutout patch length with respect to validation accuracy with 95% confidence intervals (average of five runs). Tests run on CIFAR-10 and CIFAR-100 datasets using WRN-28-10 and standard data augmentation. Baseline indicates a model trained with no cutout.

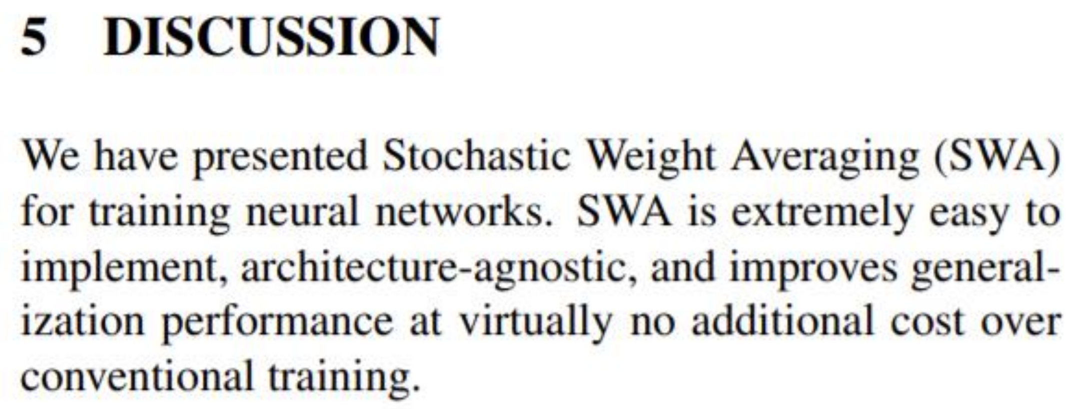
Length에 따라 Cutout의 성능이 어떻게 변하는지 알 수 있음

성능은 shape보다 **size**에 의해 더욱 많이 좌우됨.
따라서, size가 더 신경을 써야 할 Hyperparameter
정사각형의 batch만을 이용.
이때, batch는 이미지 외부에서 가지고 옴

Stochastic Weight Averaging

https://arxiv.org/abs/1803.05407





5 DISCUSSION

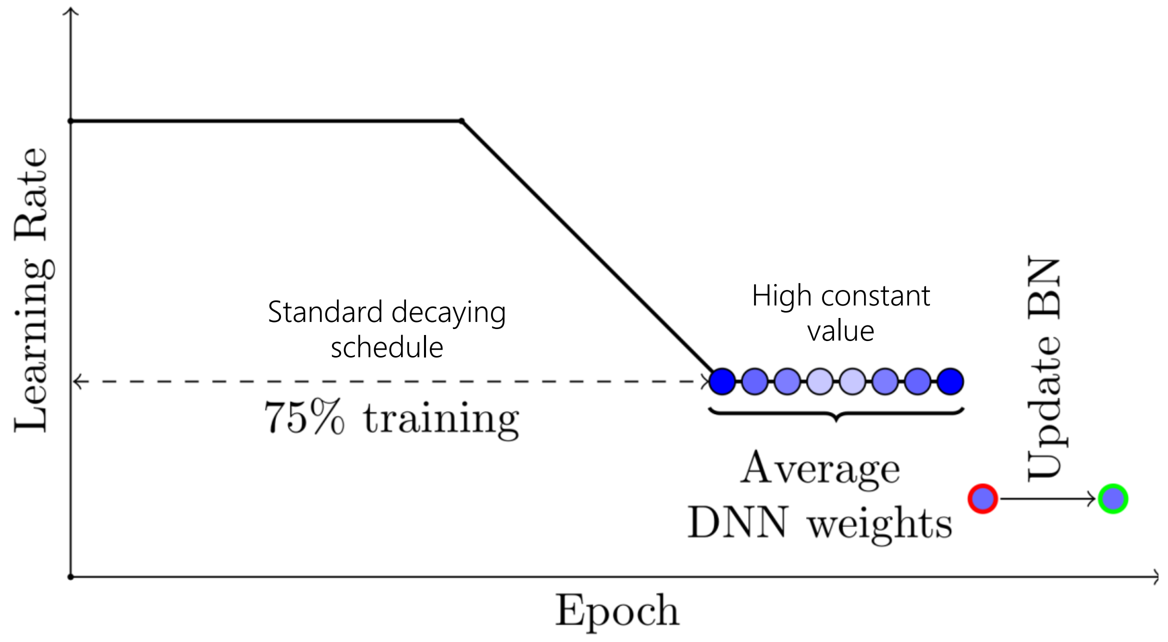
We have presented Stochastic Weight Averaging (SWA) for training neural networks. SWA is extremely easy to implement, architecture-agnostic, and improves generalization performance at virtually no additional cost over conventional training.

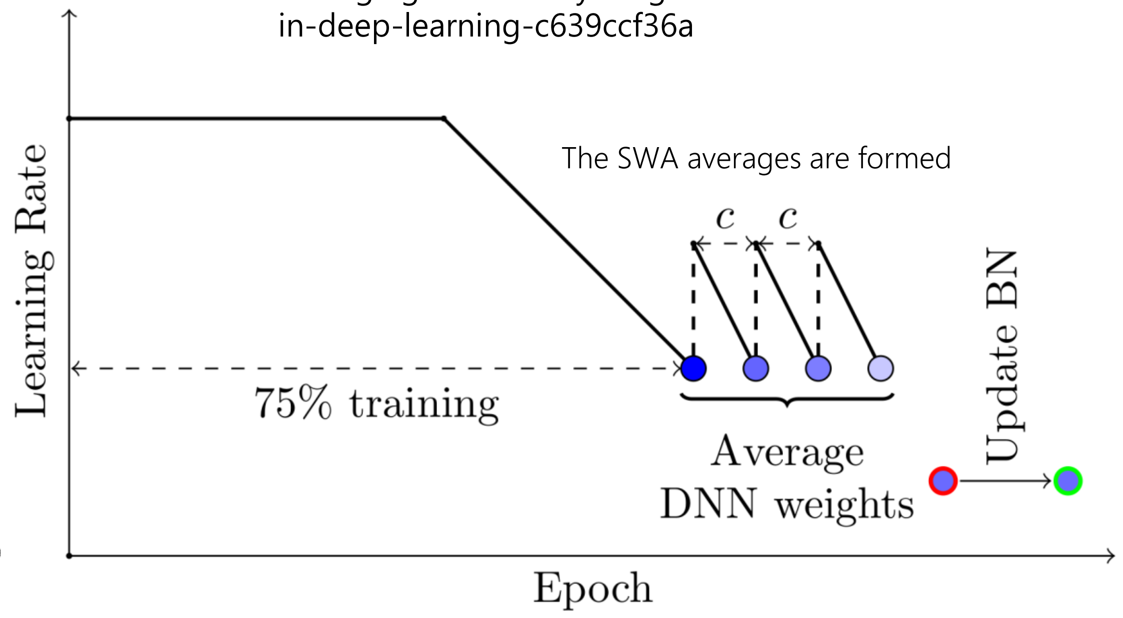
SWA is a simple procedure that **improves generalization in deep learning over Stochastic Gradient Descent (SGD) at no additional cost,** and can be used as a drop-in replacement for any other optimizer in PyTorch.

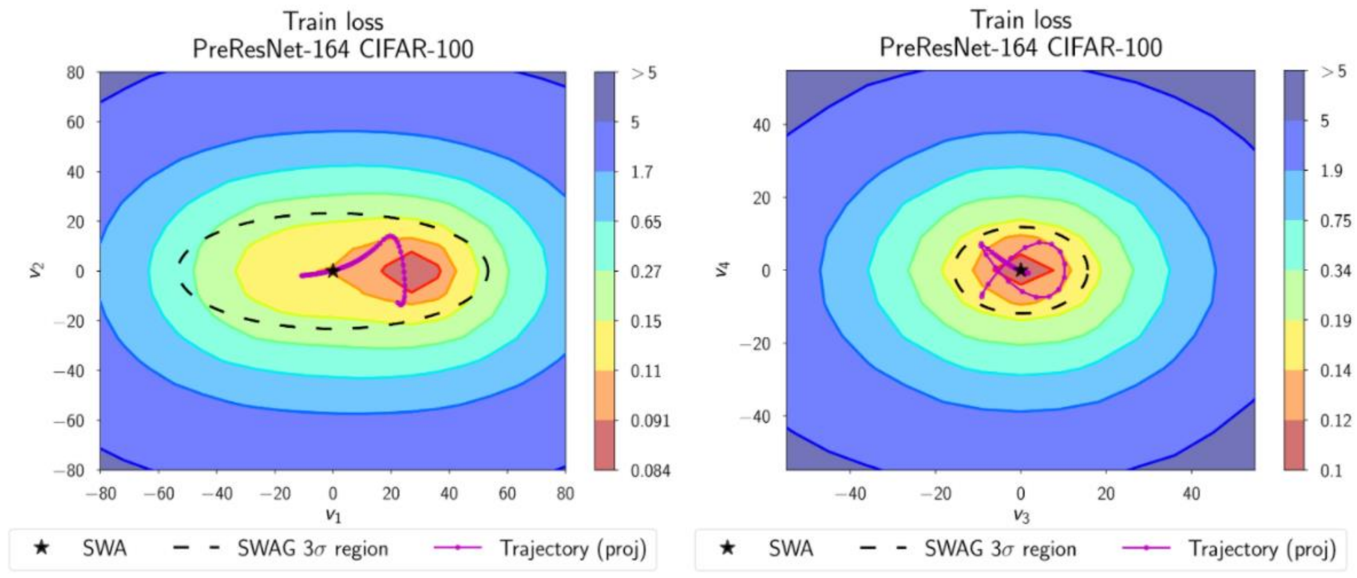
1. SWA has been shown to significantly improve generalization in computer vision tasks, including VGG, ResNets, Wide ResNets and DenseNets on ImageNet and CIFAR benchmarks [1, 2].
2. SWA provides state-of-the-art performance on key benchmarks in semi-supervised learning and domain adaptation [2].
3. SWA is shown to improve the stability of training as well as the final average rewards of policy-gradient methods in deep reinforcement learning [3].
4. An extension of SWA can obtain efficient Bayesian model averaging, as well as high quality uncertainty estimates and calibration in deep learning [4].
5. SWA for low precision training, SWALP, can match the performance of full-precision SGD even with all numbers quantized down to 8 bits, including gradient accumulators [5].

https://pytorch.org/blog/stochastic-weight-averaging-in-pytorch/

https://towardsdatascience.com/stochastic-weight-averaging-a-new-way-to-get-state-of-the-art-results-in-deep-learning-c639ccf36a





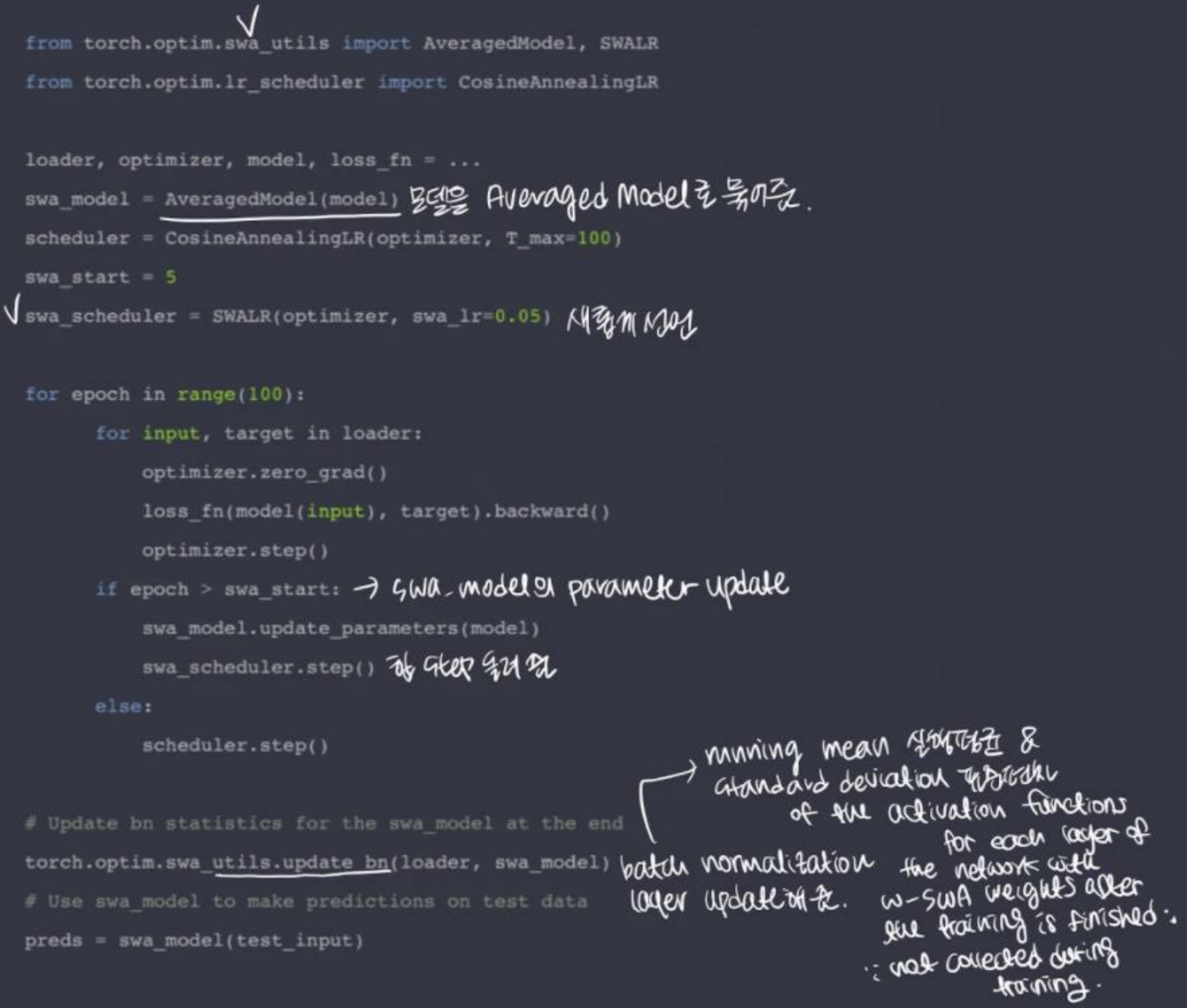


$$w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1},$$

https://github.com/pytorch/pytorch/blob/master/torch/optim/swa_utils.py

여러 파라미터 값의 평균을 사용한다는 점!

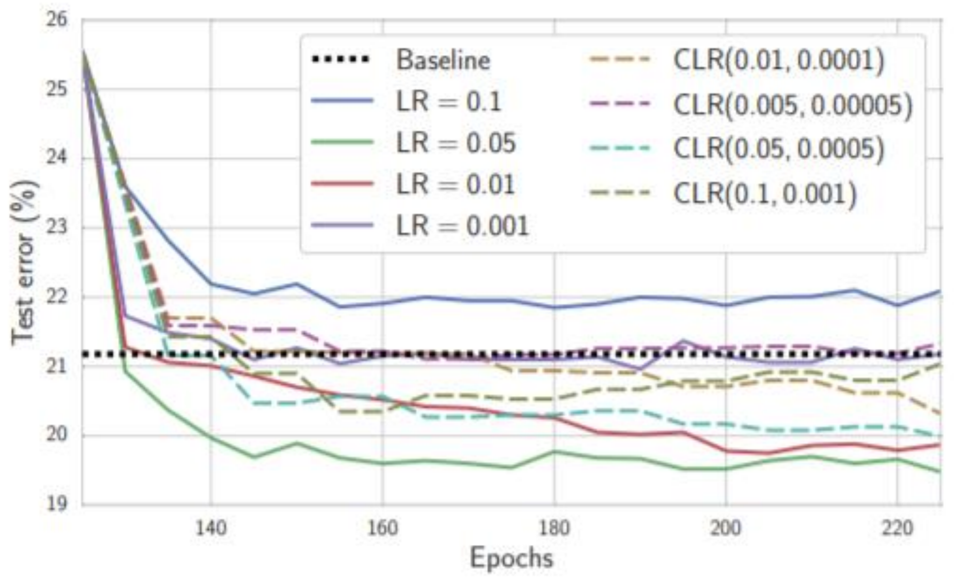
SWA



```
from torch.optim.swa_utils import AveragedModel, SWALR
from torch.optim.lr_scheduler import CosineAnnealingLR

loader, optimizer, model, loss_fn = ...
swa_model = AveragedModel(model) 모델을 Averaged Model로 묶어줌.
scheduler = CosineAnnealingLR(optimizer, T_max=100)
swa_start = 5
√swa_scheduler = SWALR(optimizer, swa_lr=0.05) 새롭게 설정

for epoch in range(100):
      for input, target in loader:
          optimizer.zero_grad()
          loss_fn(model(input), target).backward()
          optimizer.step()
      if epoch > swa_start: → swa_model의 parameter update
          swa_model.update_parameters(model)
          swa_scheduler.step() 한 step 올려줌
      else:
          scheduler.step()

# Update bn statistics for the swa_model at the end
torch.optim.swa_utils.update_bn(loader, swa_model)
# Use swa_model to make predictions on test data
preds = swa_model(test_input)
```

batch normalization layer update해줌. → running mean 실행평균 & standard deviation 계산해서 of the activation functions for each layer of the network with w-SWA weights after the training is finished ∵ not collected during training.

https://hoya012.github.io/blog/Image-Classification-with-Stochastic-Weight-Averaging-Training-PyTorch-Tutorial/



Algorithm	Test Accuracy
Baseline	94.10
SWA_90_0.05	80.53
SWA_90_5e-4	93.87
SWA_90_1e-4	94.20
SWA_90_5e-5	**94.57**
SWA_90_1e-5	94.23
SWA_75_5e-5	94.27
SWA_60_5e-5	94.33

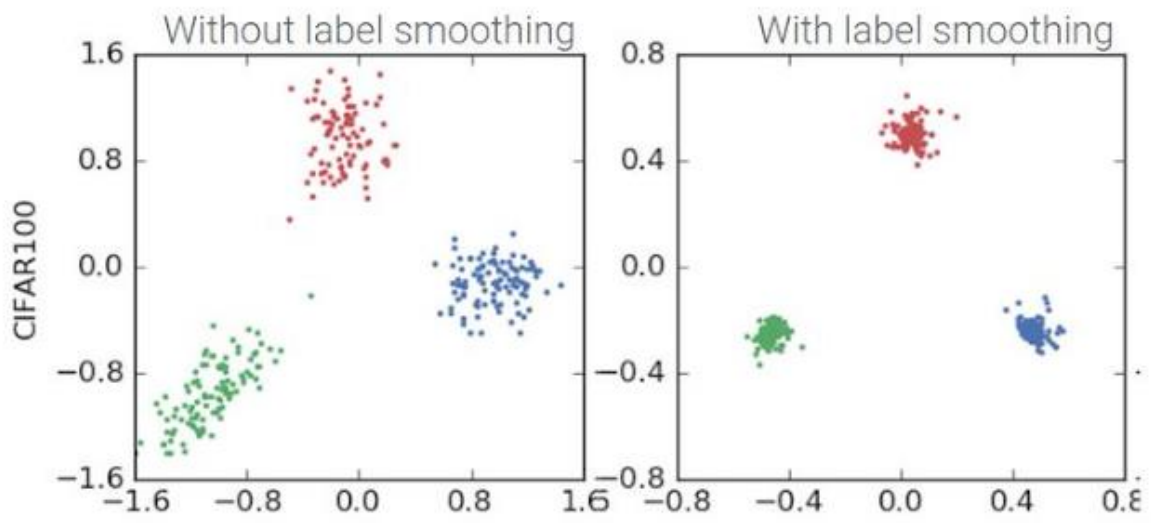
Epoch / Learning Rate의 변화에 대해
Test Accuracy가 변하는 것

- 다음 대회에서는 SWA 사용하고
epoch/lr 바꾸어 가면서 모델
발전시킬 수 있을 듯. -

(Dataset: CIFAR-10,
Intel Image Classification)

Label Smoothing

https://towardsdatascience.com/what-is-label-smoothing-108debd7ef06



$$y_k^{LS} = y_k(1-\alpha) + \frac{\alpha}{K}$$

Label Smoothing:

One-Hot 인코딩 label Vector + Uniform Distribution

K: num of class / aplha: hyperparameter

(1: uniform distribution

0: encoding된 one-hot)

Label Smoothing

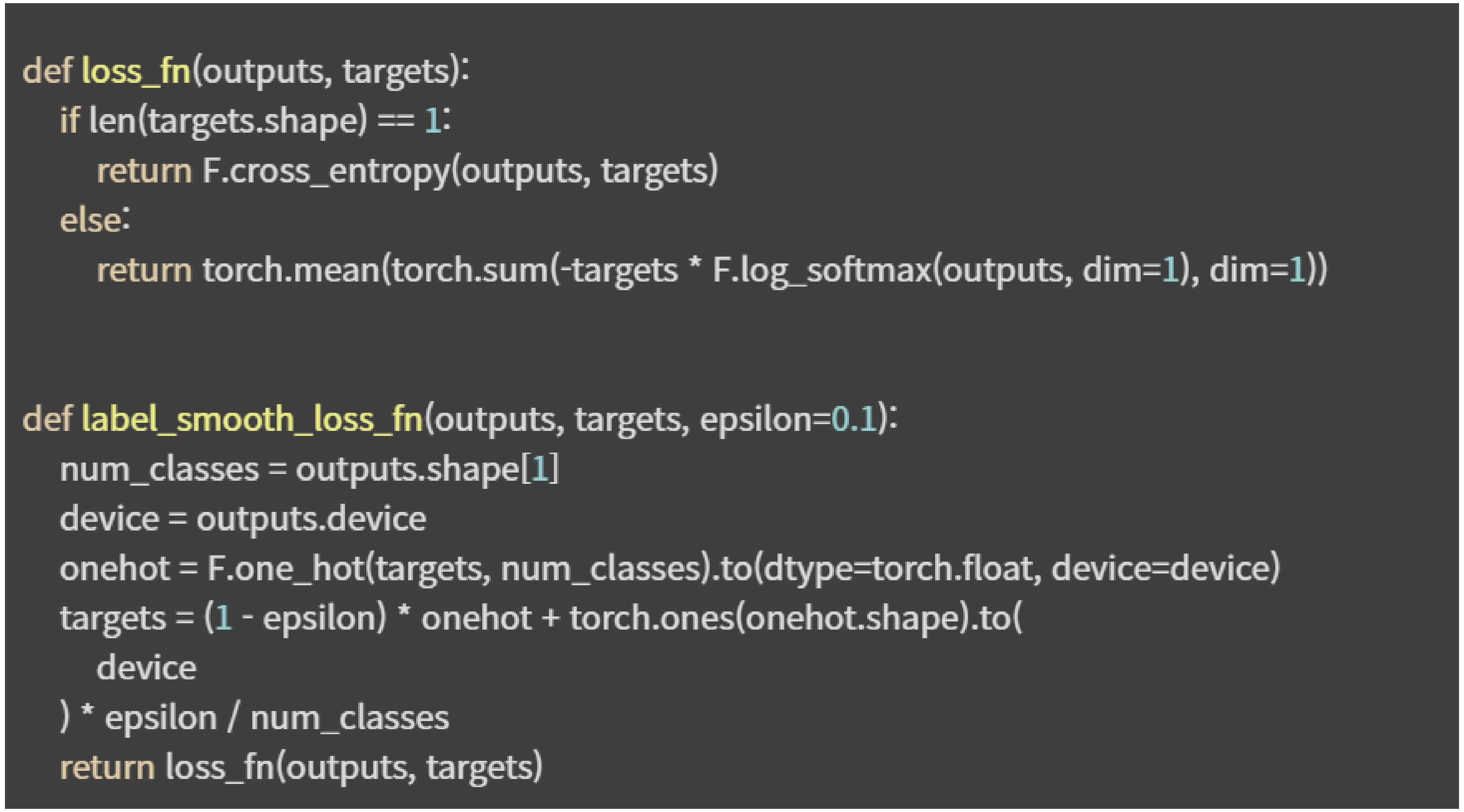
Overfitting/Overconfidence 모두 개선 가능

Introduces noise for the labels.

Loss function의 cross entropy일 때 사용됨

(return값이 loss_fn형태)

Label Smoothing is a regularization technique that **introduces noise for the labels**. This accounts for the fact that *datasets may have mistakes in them*, so maximizing the likelihood of *directly* can be harmful. Assume for a small constant , the training set label is correct with *probability* and incorrect otherwise.
Label smoothing regularizes a model based on a **softmax** with *k* output values by replacing the hard *0* and *1* classification targets with targets of eps / k and 1 - (k - 1) / k * eps, respectively.



```
def loss_fn(outputs, targets):
    if len(targets.shape) == 1:
        return F.cross_entropy(outputs, targets)
    else:
        return torch.mean(torch.sum(-targets * F.log_softmax(outputs, dim=1), dim=1))


def label_smooth_loss_fn(outputs, targets, epsilon=0.1):
    num_classes = outputs.shape[1]
    device = outputs.device
    onehot = F.one_hot(targets, num_classes).to(dtype=torch.float, device=device)
    targets = (1 - epsilon) * onehot + torch.ones(onehot.shape).to(
        device
    ) * epsilon / num_classes
    return loss_fn(outputs, targets)
```

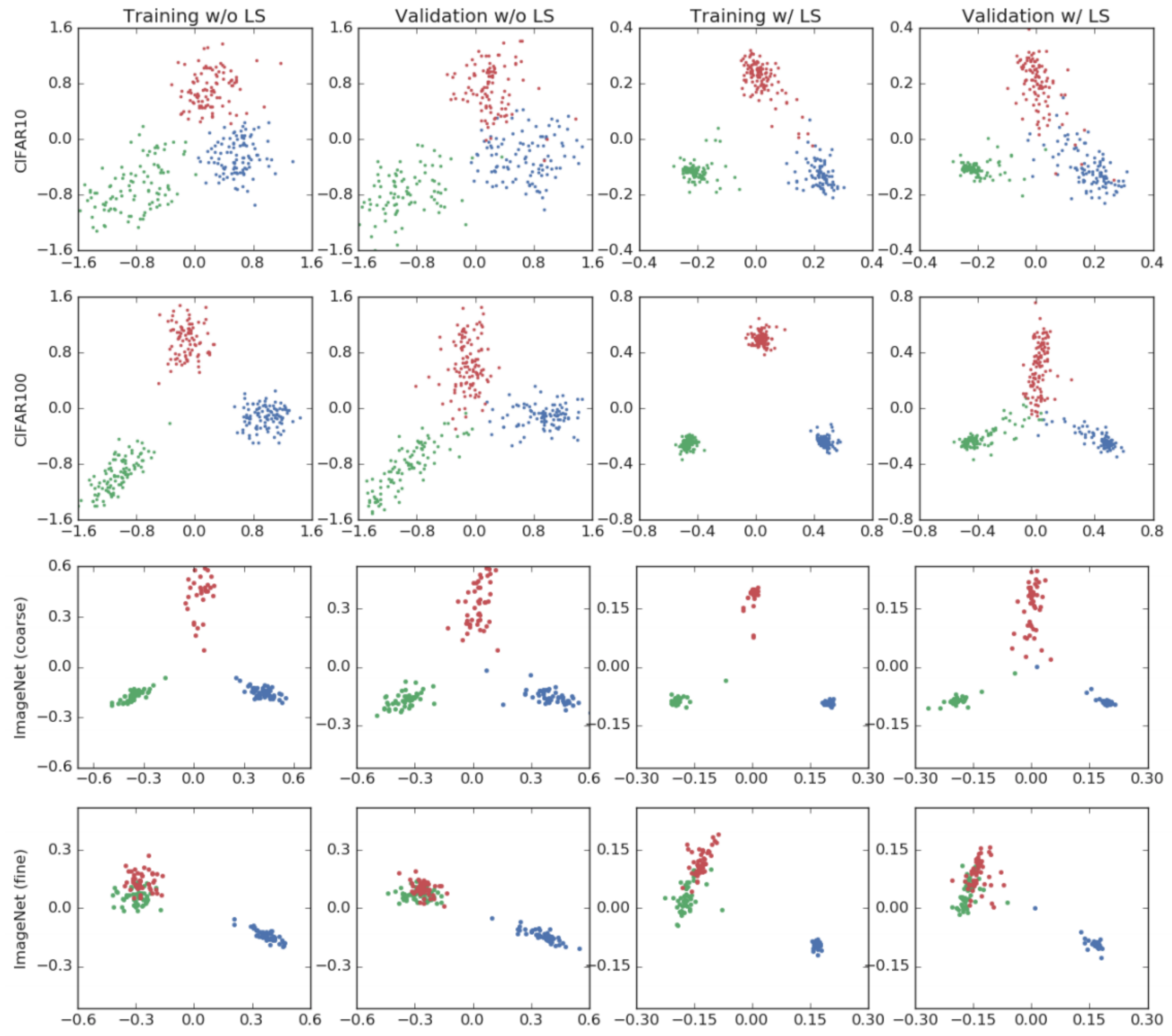


Figure 1: Visualization of penultimate layer's activations of: AlexNet/CIFAR-10 (first row), CIFAR-100/ResNet-56 (second row) and ImageNet/Inception-v4 with three semantically different classes (third row) and two semantically similar classes plus a third one (fourth row).

기존 문제점

1. 올바른 데이터 >>> 오류 데이터
2. 오류 데이터는 올바른 데이터와 차이가 엄청 큼.

해결책

1. (the correct) – (the incorrect) == α * constant
2. Activation of the penultimate layer가 정확한 클래스와 가까워지는 만큼 오류 클래스와 멀어지기

https://medium.com/@nainaakash012/when-does-label-smoothing-help-89654ec75326