Lab 2: Deep Residual Learning

Department of Computer Science, NCTU

TA Yu-Chuan Chuang (莊祐銓)

Paper: He, Kaiming, et al. "Deep residual learning for image recognition." CVPR, 2016.

Important Rules

Important Date :

- Report Submission Deadline: 8/28 (Wed) 12:00(中午)
- Demo date: 8/28 (Wed)

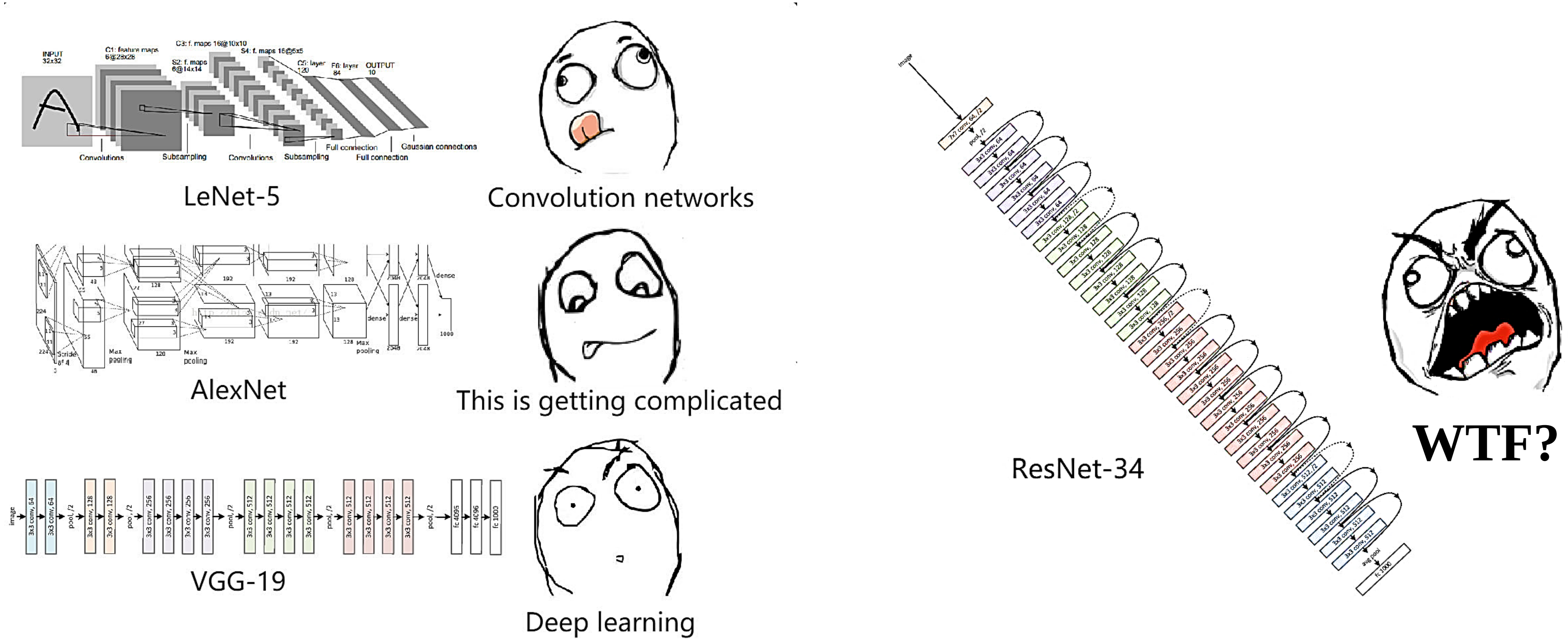
Turn in :

- Experiment Report (.pdf)
- Source code (.py)

Notice: zip all files in one file and name it like「DLP_LAB2_your studentID_name.zip」, ex: 「DLP_LAB2_0656608_莊祐銓.zip」

Lab Objective

- In this lab, you will be asked to build the SOTA convolutional neural network:
  - ➢ Implement ResNet and train on **cifar-10 dataset**
  - ➢ Training ResNet with **data augmentation** and **learning rate schedule**



Source: https://linkinpark213.com/2018/04/22/resnet/

Requirements

- Implement ResNet[1] 18/50 architecture and train on cifar-10 dataset
- Training ResNet with data augmentation and learning rate schedule
- Compare to vanilla CNNs(no shortcut) and plotting comparison figure

Dataset - Cifar-10

- Cifar-10 dataset consists of 60000 images (32×32-RGB)
- Training: 50000 images, Testing: 10000 images
- Class: 10 (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks)

reference: https://www.cs.toronto.edu/~kriz/cifar.html

Prepare Data

- Data loader: **torchvision.datasets.CIFAR10**
  - Input: [Batch size, 3, 32, 32]
  - Ground truth: [Batch size]

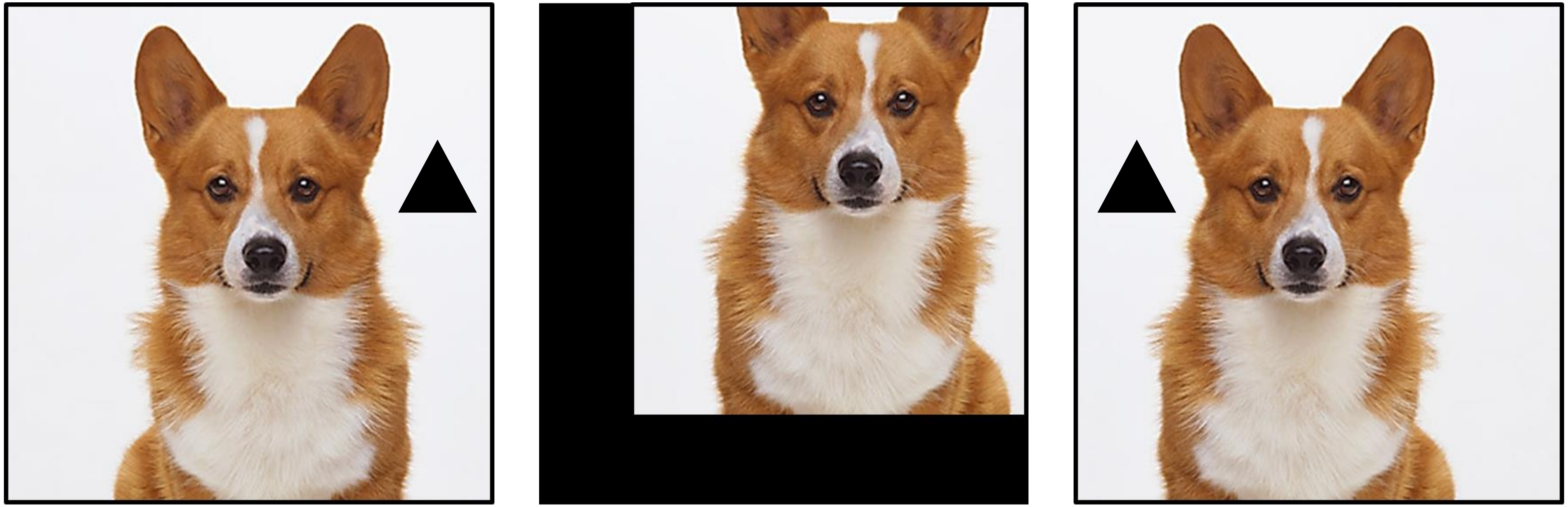
- Color normalization: **torchvision.transforms**

Mean $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{pmatrix} 0.4914 \\ 0.4824 \\ 0.4467 \end{pmatrix}$ **Standard deviation** $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{pmatrix} 0.2471 \\ 0.2435 \\ 0.2616 \end{pmatrix}$

reference: https://pytorch.org/docs/stable/torchvision/datasets.html#cifar

Prepare Data

- Data augmentation: **torchvision.transforms**
  - **Translation**: pad 4 zeros in each side and random cropping to **32x32** size
  - **Horizontal flipping**: probability **0.5**

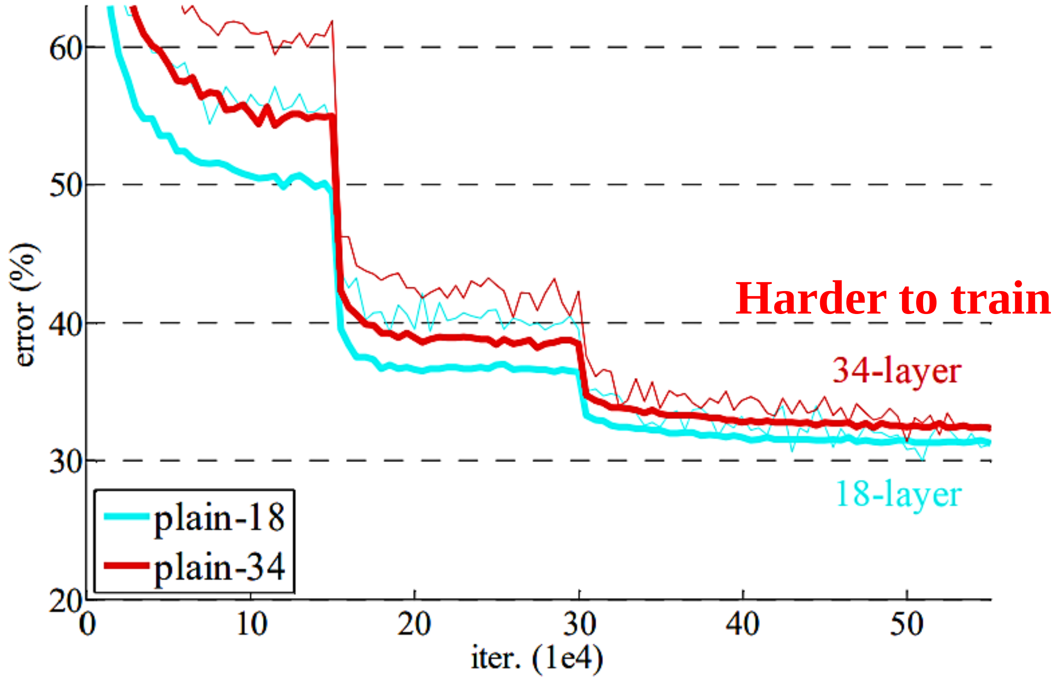


Original **Translation** **Horizontal flipping**

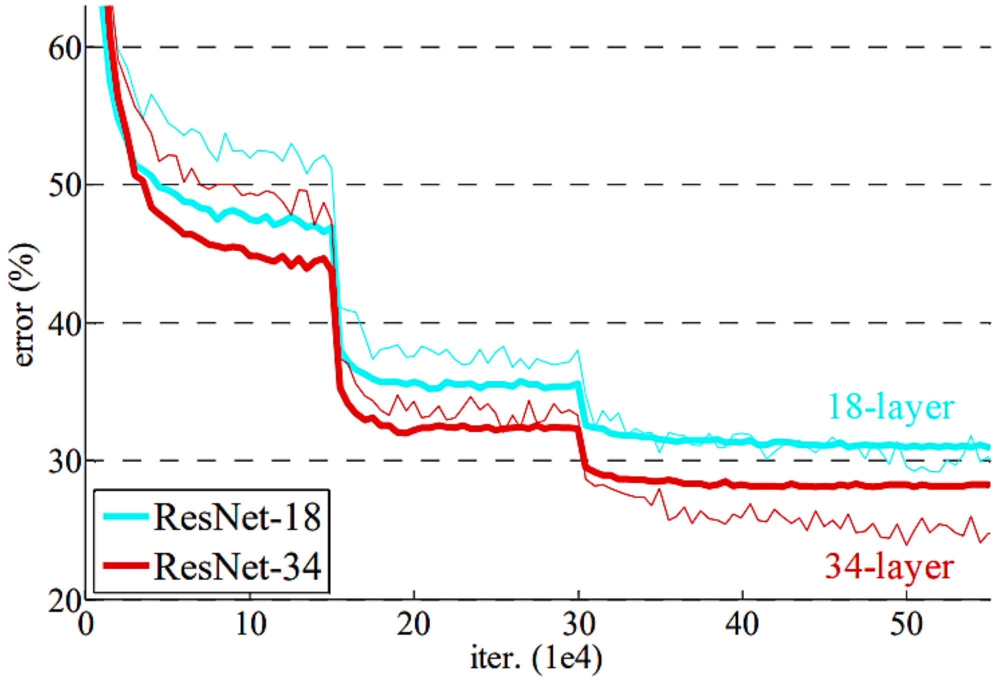
reference: https://pytorch.org/docs/stable/torchvision/datasets.html#cifar

ResNet

- Winner of ILSVRC 2015 in image classification, detection, and localization
- Winner of MS COCO 2015 detection.



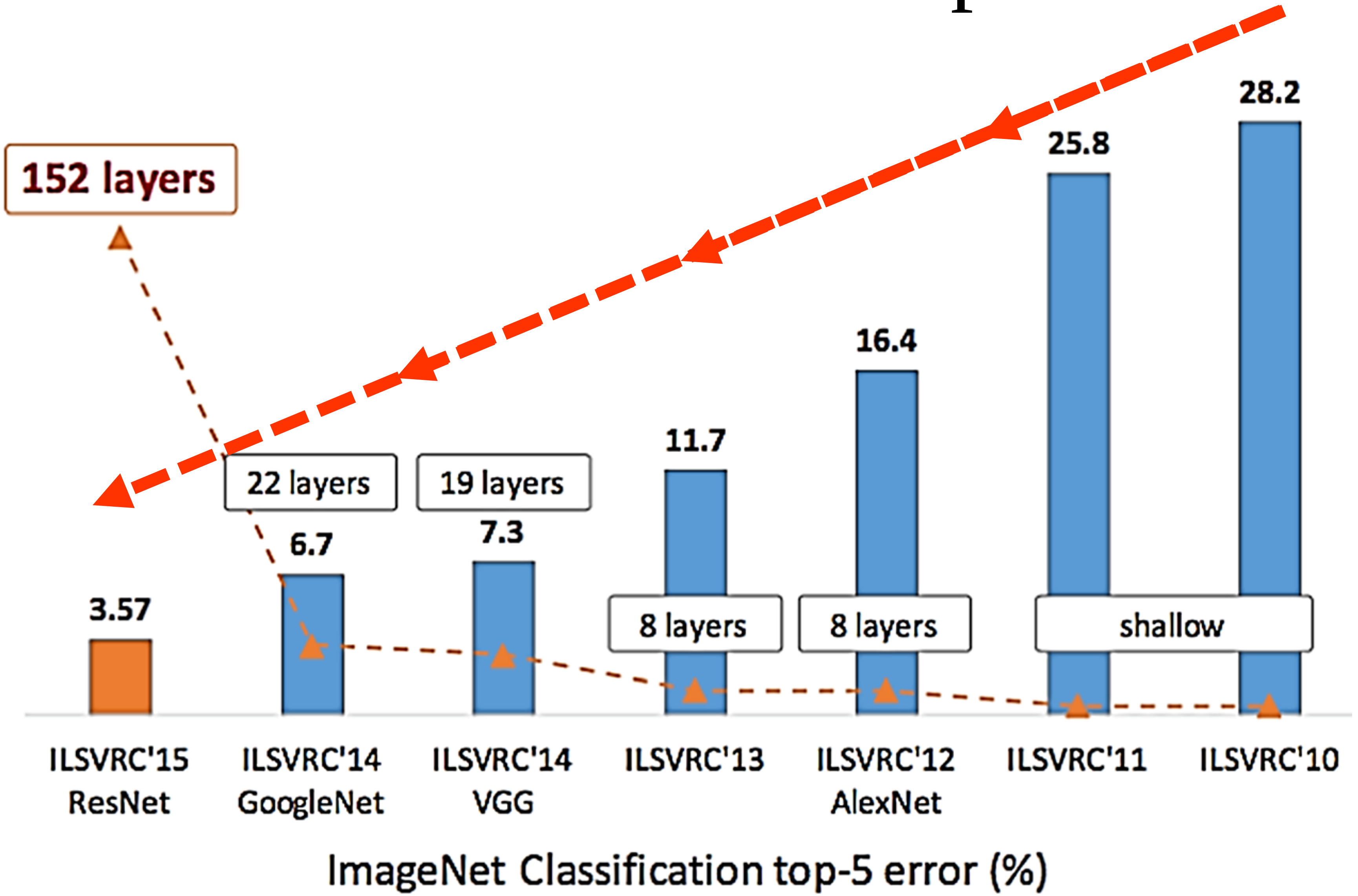
Vanilla CNNs



ResNet

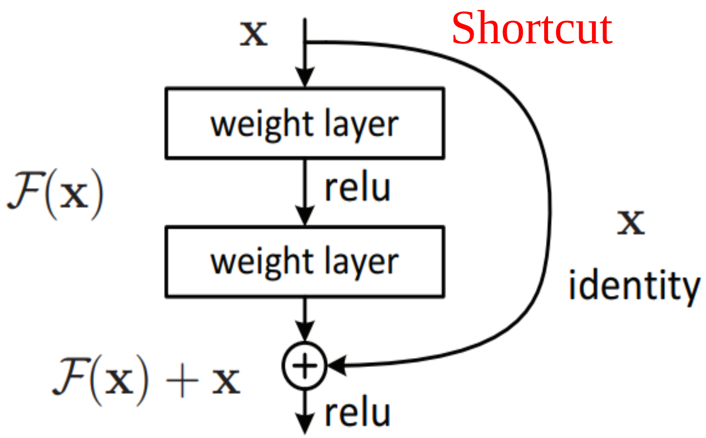
Revolution of Depth

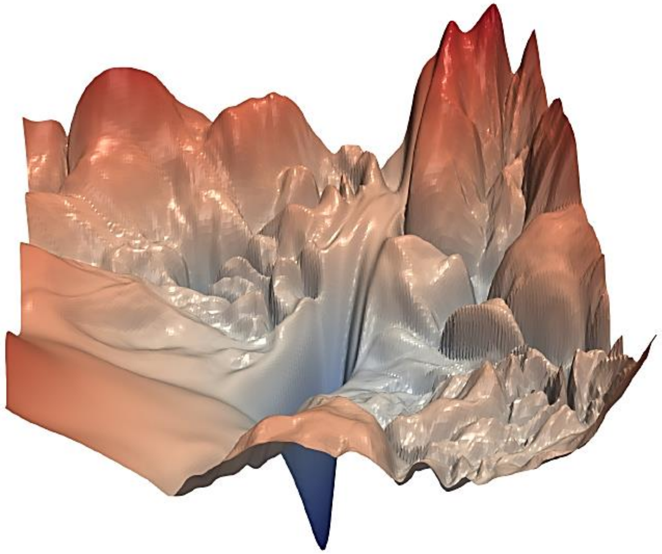
WTF?



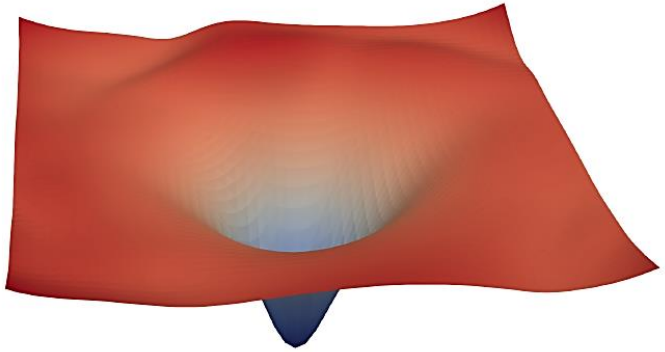
Shortcut Connection

- To tackle the problem of vanishing/exploding gradients, a skip / shortcut connection is added to add the input x to the output after few weight layers as below





(a) without skip connections

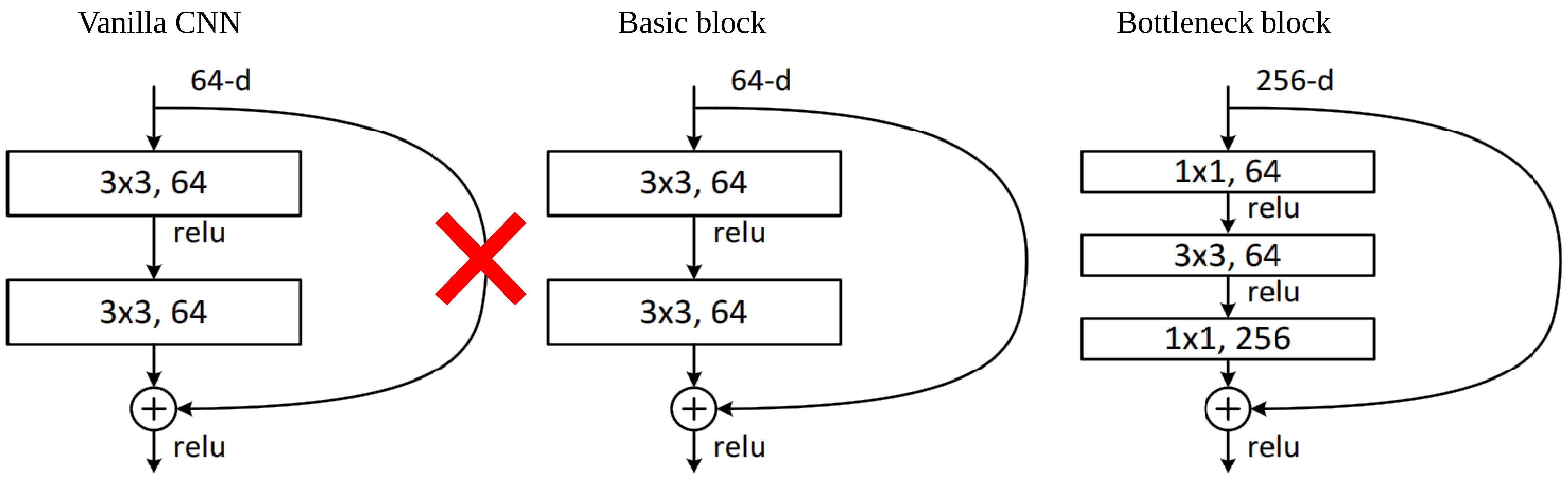


(b) with skip connections

Source: Li, Hao, et al. "Visualizing the loss landscape of neural nets." Advances in Neural Information Processing Systems. 2018.

ResNet Block

- ResNe18(Basic block), ResNet50(Bottleneck block)



ResNet Architecture

- ResNe18(Basic block), ResNet50(Bottleneck block)

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

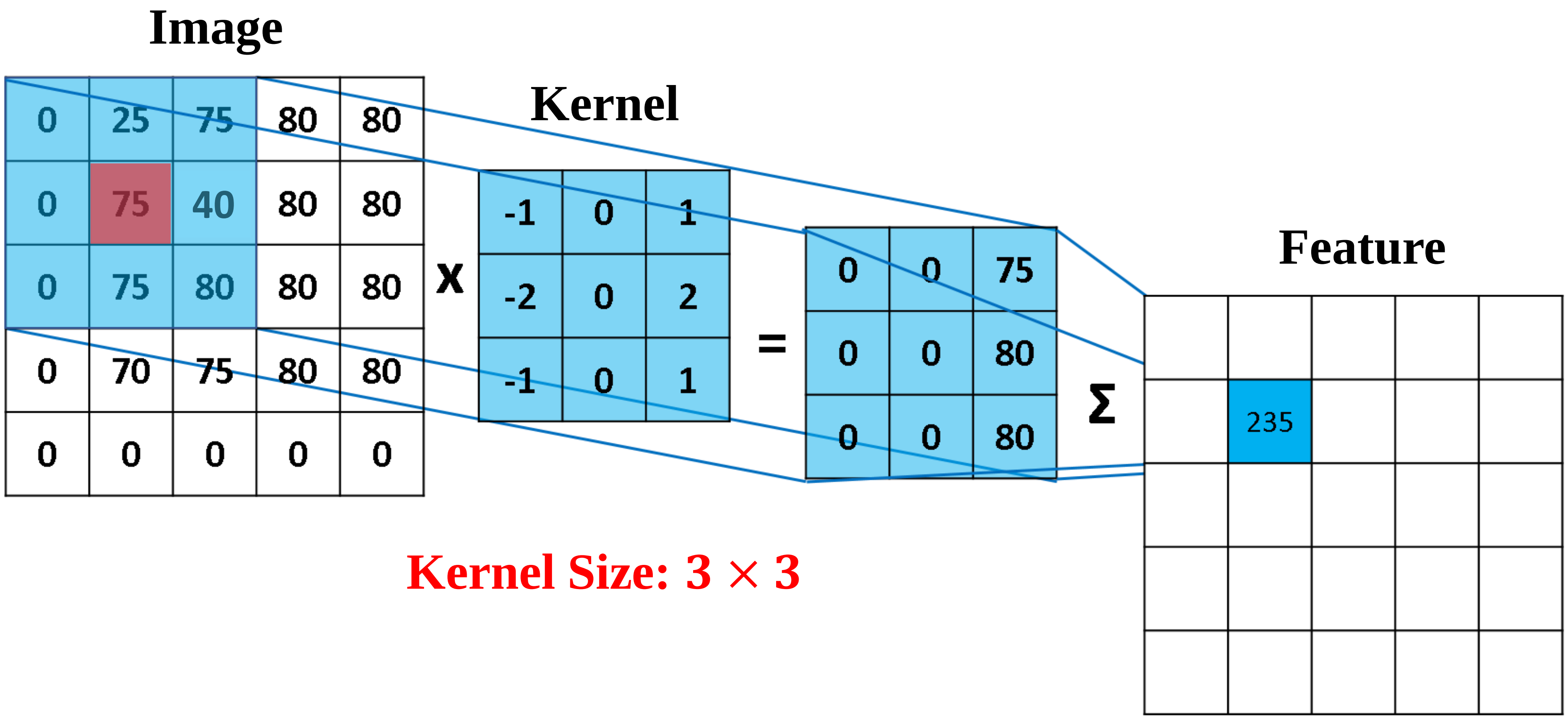
Implement Details

- **Conv:** torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding)
- **Batch Normalization:** torch.nn.BatchNorm2d (in_channels)
- **ReLU:** torch.nn.functional.relu()

- 2D feature: [batch size, channel, height, width]

- Important parameters:
  - Channel
  - Kernel size
  - Stride
  - Padding

Convolution



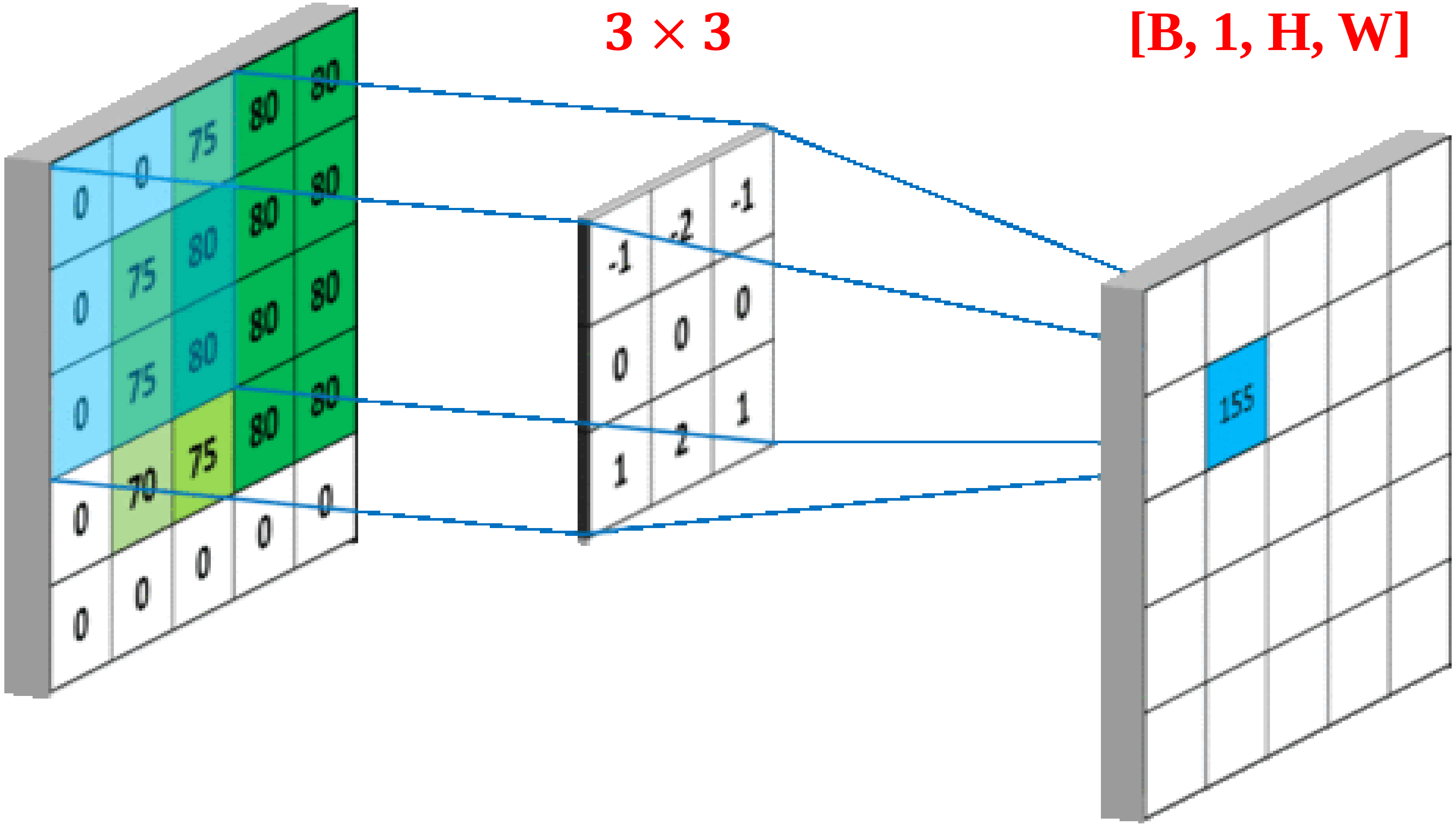
Kernel Size: 3 × 3

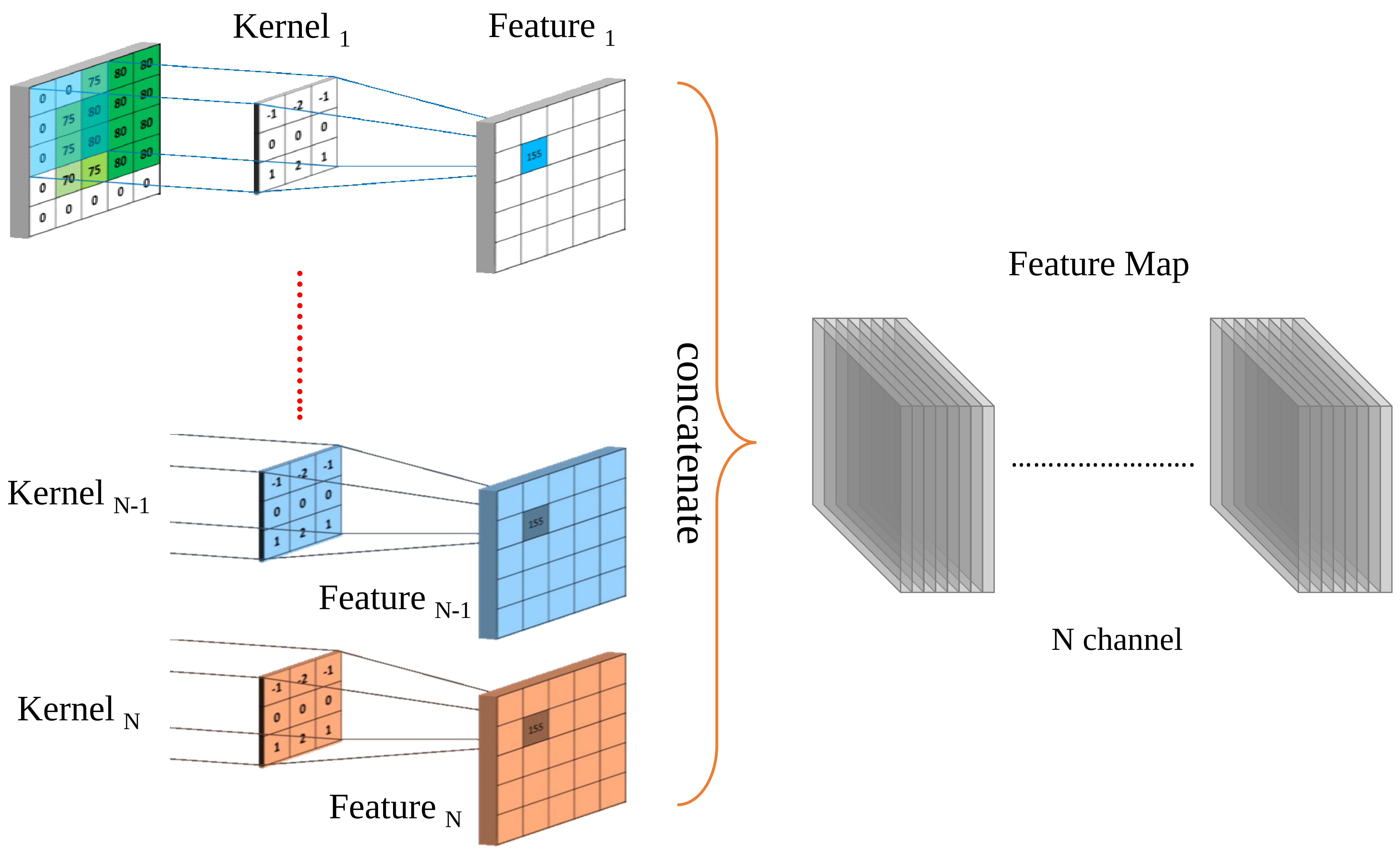
Convolution

Image
[B, 1, H, W]

Kernel
3 × 3

Feature
[B, 1, H, W]




Kernel 1
Feature 1
Kernel N-1
Feature N-1
Kernel N
Feature N
concatenate
Feature Map
N channel

Channel

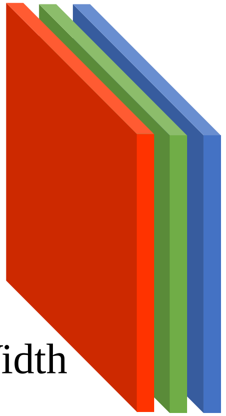
Width

Height

height × width × 3

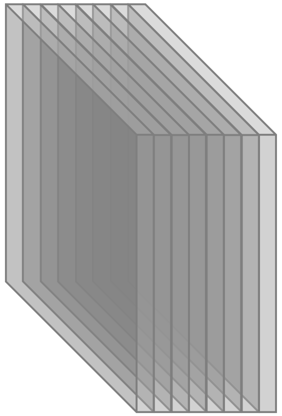
Height

Width

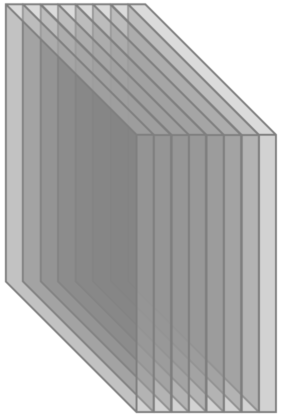


Channel

Feature Map



....

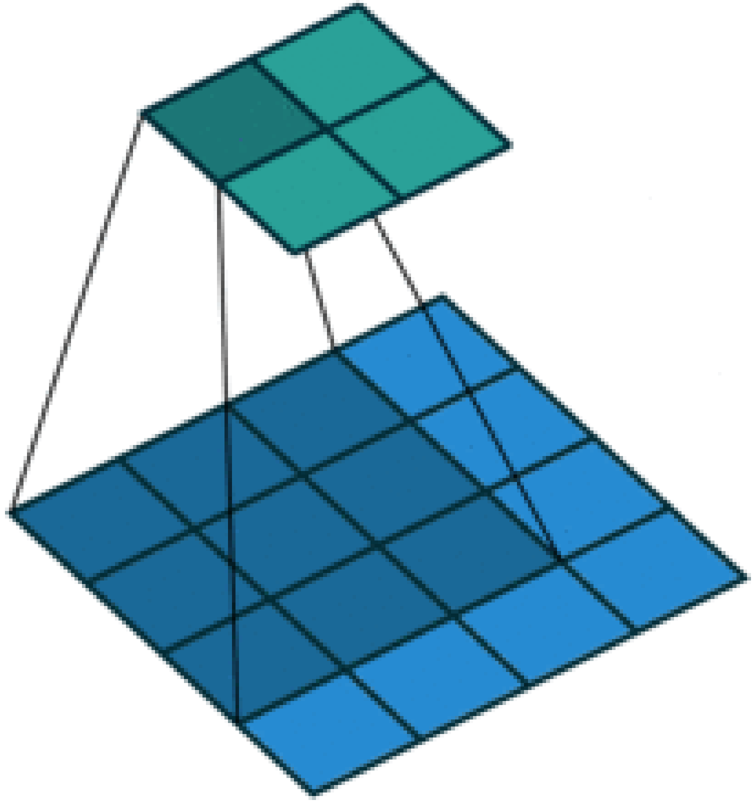


channel × height × width

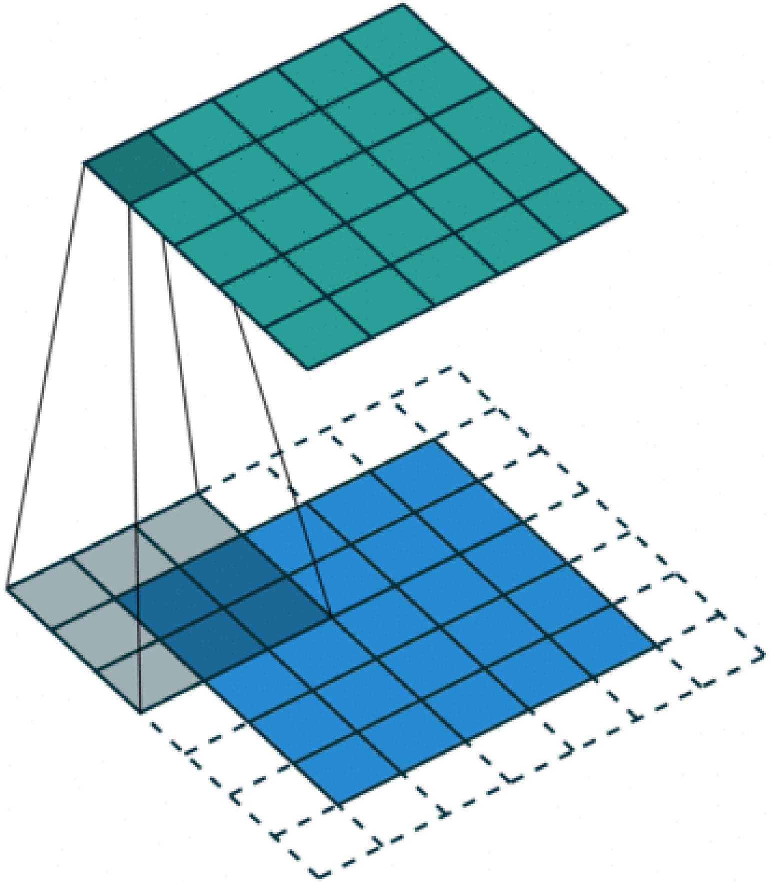
- Each channel has its own filter(weight) and feature

URL: https://*weburbanist.com/2011/09/21/incredible-rgb-art-shifts-as-lighting-colors-change/*

Padding



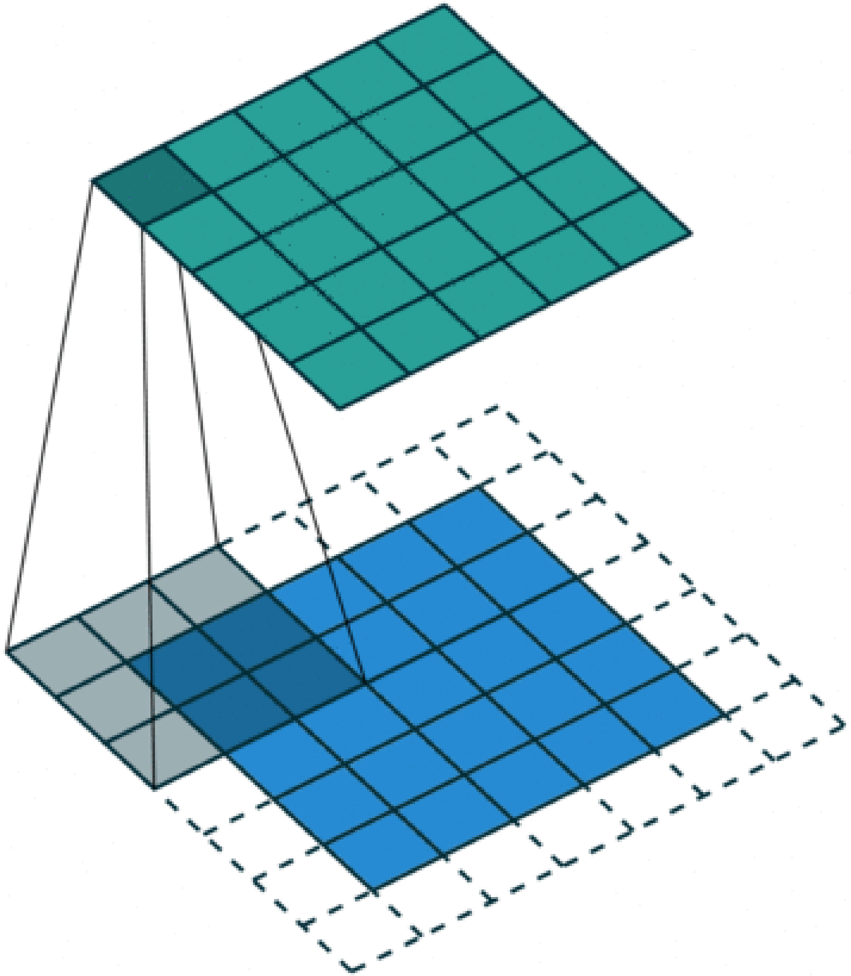
No padding



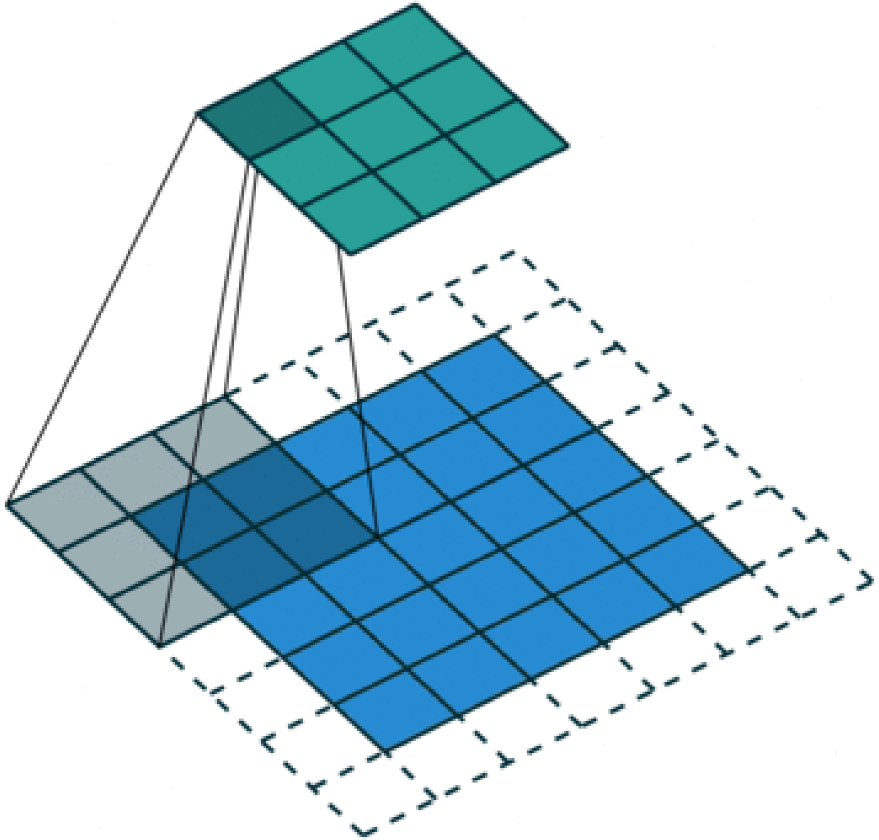
With padding

URL:https://github.com/vdumoulin/conv_arithmetic

Stride



With padding
Stride=1



With padding
Stride=2

URL:https://github.com/vdumoulin/conv_arithmetic

ResNe18 - Basic block

layer name	output size	18-layer
conv1	112×112	
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$

Conv2d(in_channels, out_channels, kernel_size, stride, padding)

64 **128** **3 × 3** **2** **1**

Hyper Parameters

- Optimizer: SGD, momentum 0.9, weight decay = 0.0001
- Mini batch size: 128
- Total epochs: 164
- Initial learning rate: 0.1, divide by 10 at 81, 122 epoch
- Weight initialization: torch.nn.init.kaiming_normal
- Loss function: nn.CrossEntropyLoss()

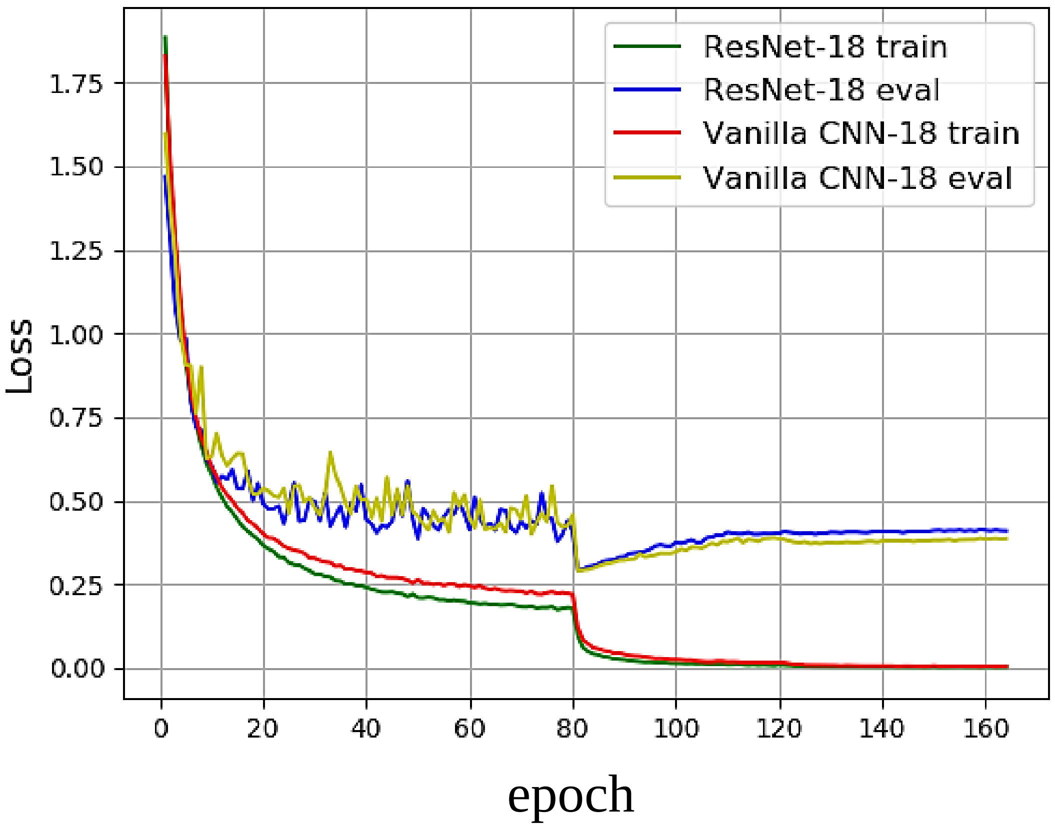
- **You can adjust the hyper-parameters according to your own ideas.**

- If you use "nn.CrossEntropyLoss", don't add softmax after final fc layer because this criterion combines LogSoftMax and NLLLoss in one single class.

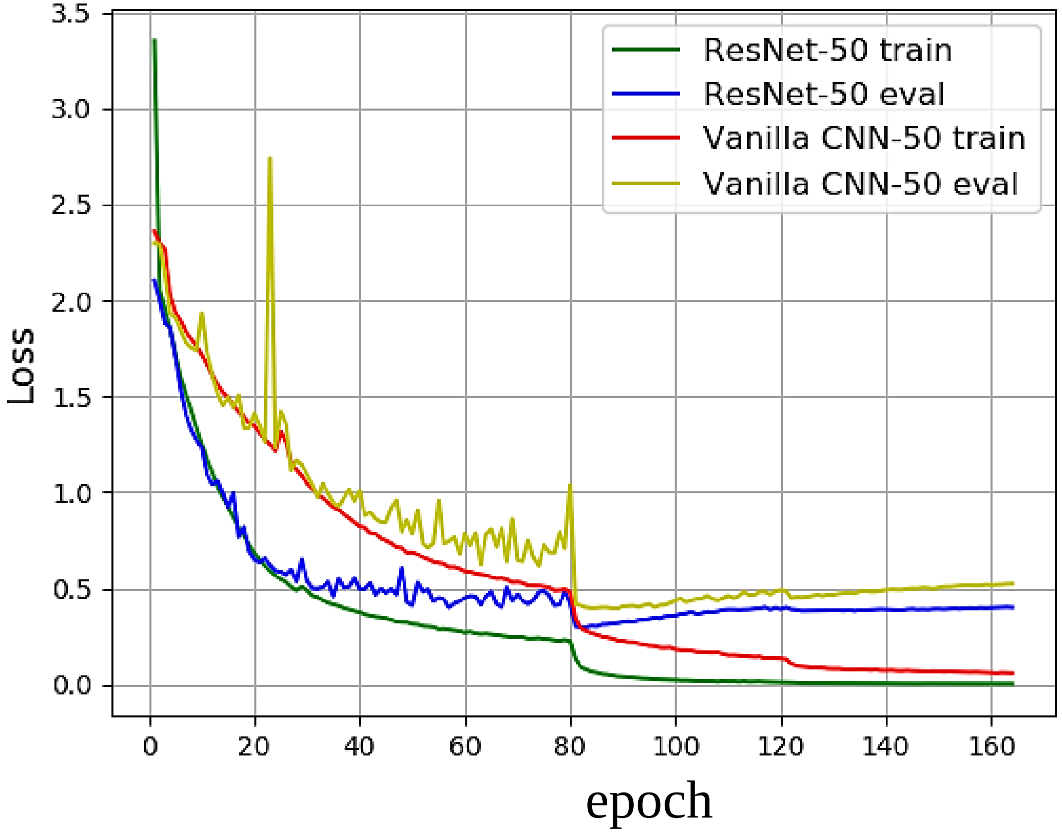
Result Comparison (loss curve)

- Compare to vanilla CNNs (no shortcut)

18 layers

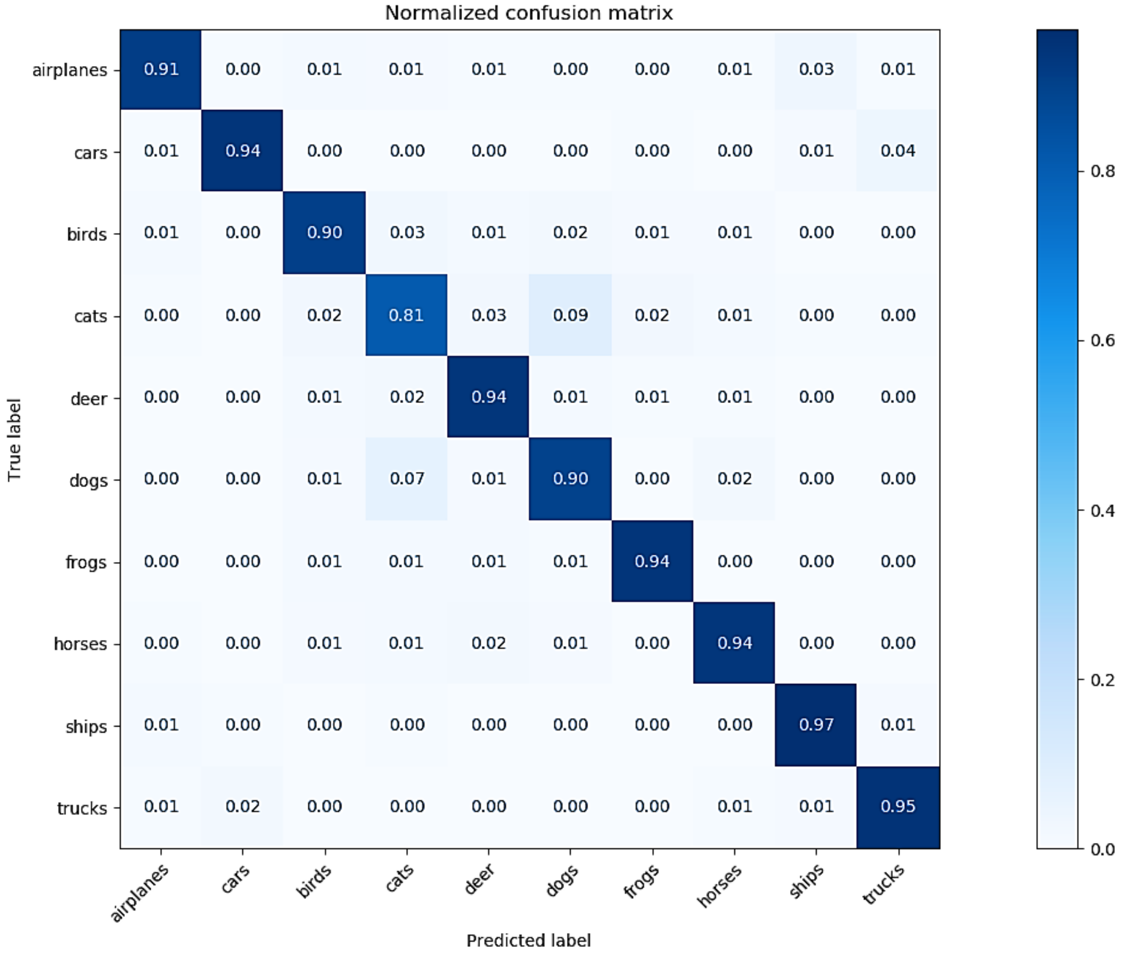


50 layers



Bonus (+5) – Confusion Matrix

- Confusion matrix is often used to describe the performance of a classification model
- Useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC curve



Report Spec

1. Introduction (20%)
   (簡單說明這次 LAB 的目標以及 ResNet 如何避免 Degradation problem)

2. Experiment setup (20%)
   A. The detail of your model
      (簡述 code 以及 model 實作細節)
   B. Report all your training hyper parameters

3. Result (40%)
   A. The comparison between ResNet and vanilla CNNs
   B. Final Test error (screen shot)

4. Discussion (20%)
   (任何你想討論的都可寫在這，包含抱怨也可以 XD)

Criterion of Result

Accuracy	$> 91\%$	$= 100\%$
Accuracy	$: 91\% \sim 90\%$	$= 90\%$
Accuracy	$: 90\% \sim 87\%$	$= 80\%$
Accuracy	$: 87\% \sim 10\%$	$= 70\%$
Accuracy	$: 10\%$	$= 0\%$

Score: 40% experimental results + 60% (report+ demo score)

Reference

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.