Lab2: Deep Residual Learning

Lab Objective:

In this lab, you will be asked to build the state of the art convolutional neural network architecture: Residual Network (ResNet) [1] and train it on the cifar-10 dataset. Moreover, you need to use data augmentation and learning rate schedule during training.

Important Date:

1. Experiment Report Submission Deadline: 8/28 (Wed) 12:00(中午)
2. Demo date: 8/28 (Wed)

Turn in:

1. Experiment Report (.pdf)
2. Source code

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」

Requirements:

1. Implement **ResNet 18/50** architecture and train on **cifar-10** dataset [1]
2. Training ResNet with **data augmentation** and **learning rate scheduling**
3. Compare to **vanilla CNNs** (without skip connection) with same architecture 18/50 and plotting the **comparison figure (loss curve)**

Demo:

1. Show your **code** and explain briefly
2. Show your **testing results (accuracy)**
3. Show your **comparison figure**
4. (optimal) Show your **confusion matrix**, if you have completed bonus part
5. We will ask some simple questions

Implementation Details:

1. Prepare data

- **Cifar-10 dataset**

  The cifar-10 dataset consists of 60000 32×32 color images (RGB) in 10 classes, shown as below figure. There are 50000 images for training and 10000 images for testing.

- **Load data**

  You can use "**torchvision**" package to read CIFAR-10 data. The "torchvision" package consists of popular datasets, model architectures, and common image transformations for computer vision. The more details can be found in the official documents.

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

- **Data preprocessing**

  A. Color normalization

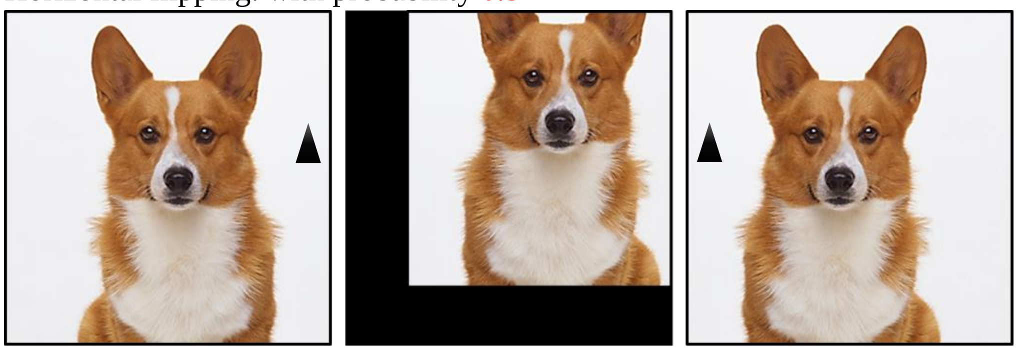
  Normalize each color channel (compute from entire CIFAR10 training set)

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

  B. Data augmentation

  Translation: **pad 4 zeros** in each side and random cropping back to 32x32 size

  Horizontal flipping: with probability **0.5**



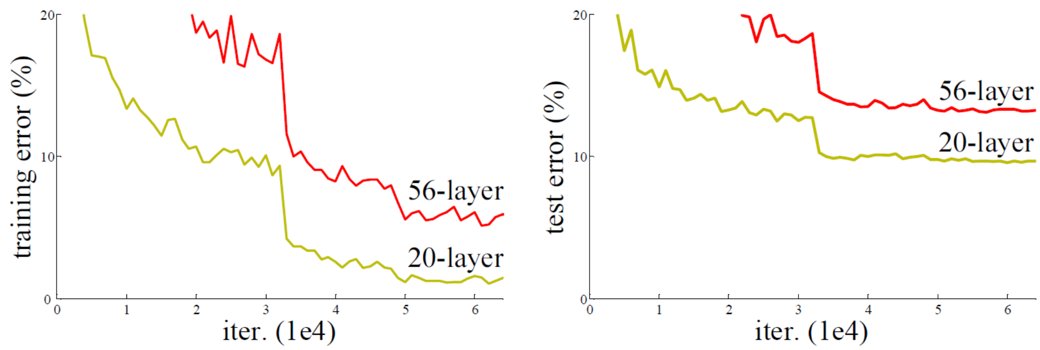
Original **Translation** **Horizontal flipping**

2. **Deep residual network**

ResNet (Residual Network) is the Winner of ILSVRC 2015 in image classification, detection, and localization, as well as Winner of MS COCO 2015 detection.

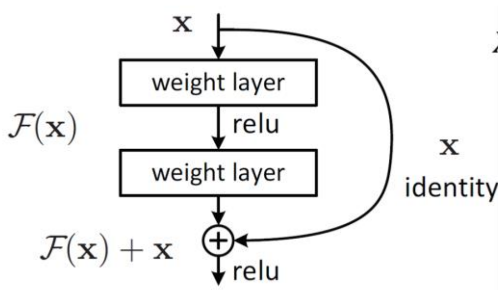
- **Degradation problem**

  The network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Not overfitting, it's the vanishing/exploding gradients problem.



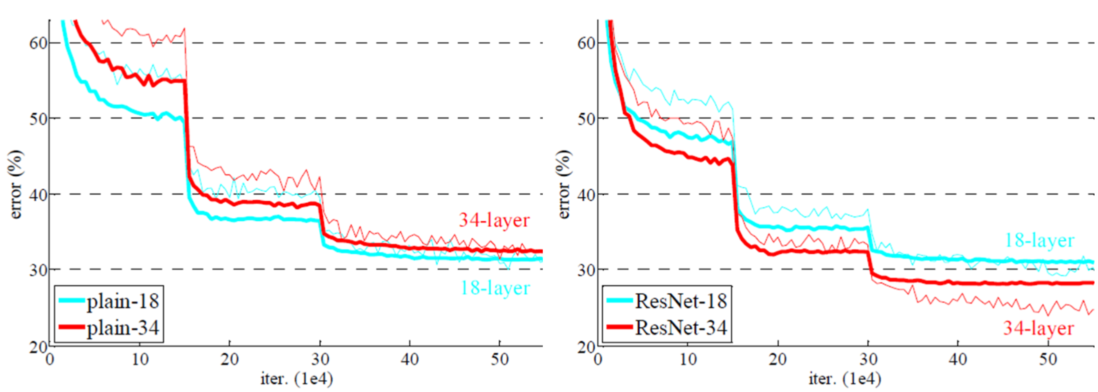
- **Skip/Short connection**

  To solve 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.

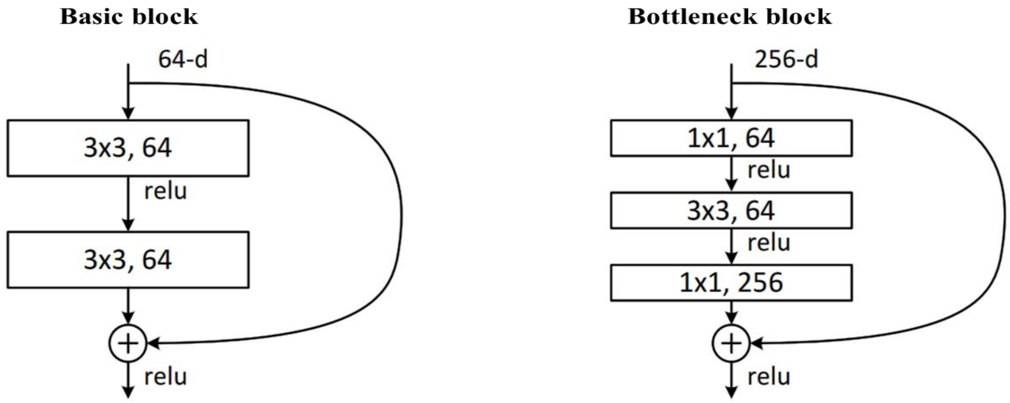


$$X_l = F(X_{l-1}) + X_{l-1}$$

- **Learning better networks as easy as stacking more layer**



- **Building residual block**



- **Network architecture**

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

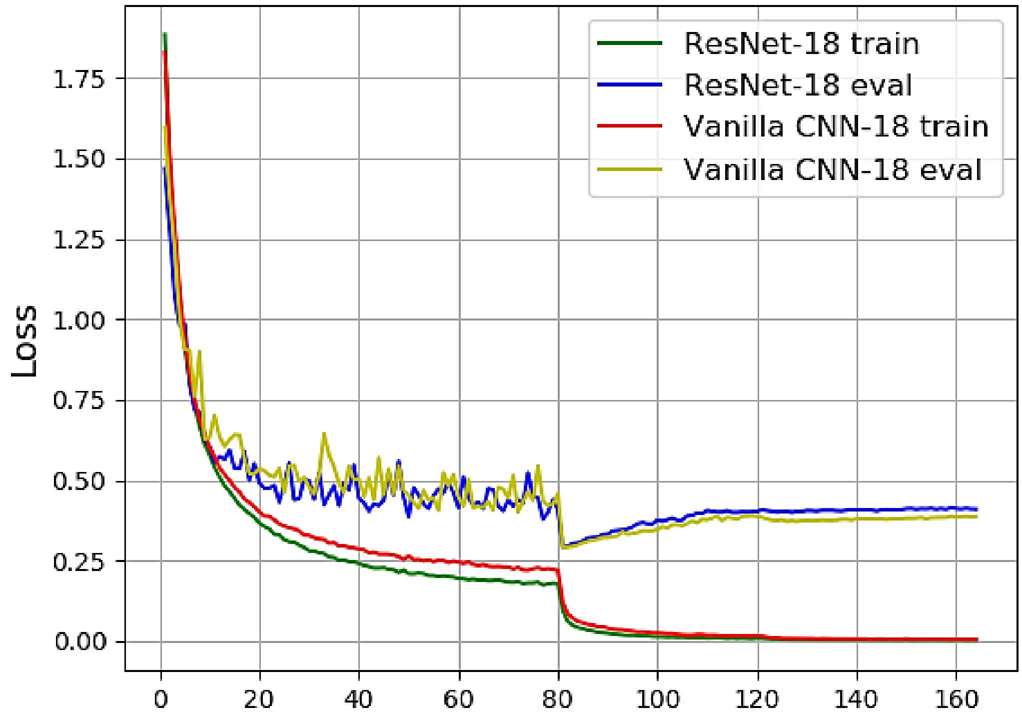
- **Sample code**
  URL: https://github.com/kuangliu/pytorch-cifar

3. Hyper-parameters

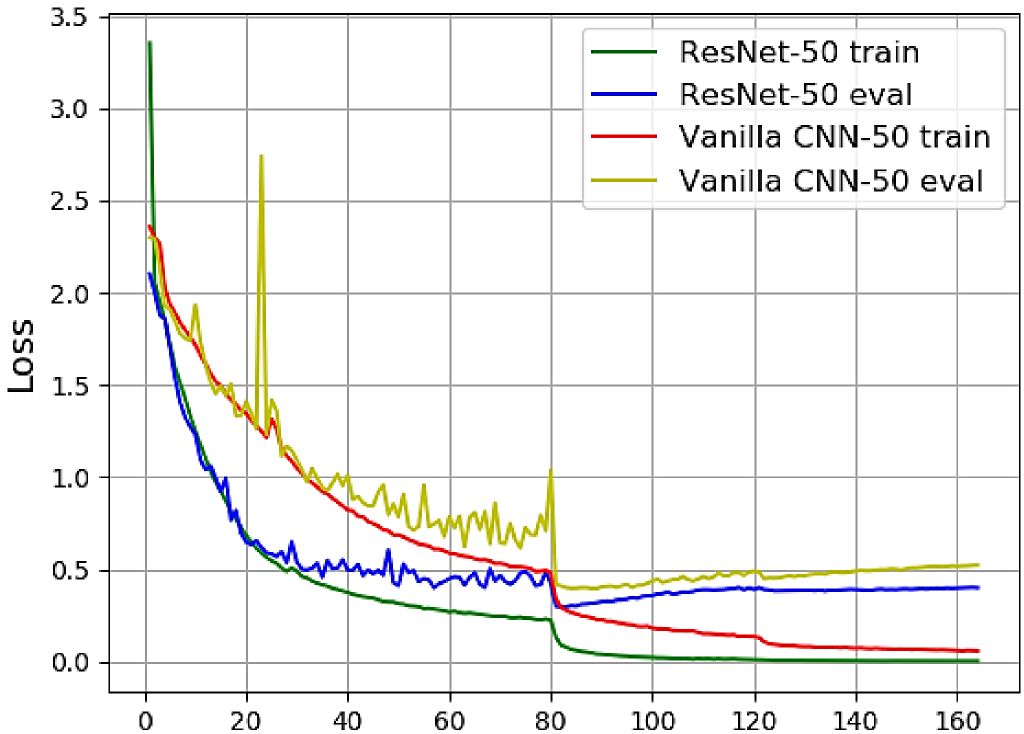
- Optimizer: **SGD** with mometum
- Mini batch size: **128**
- Total epochs: **164**, momentum **0.9**
- Initial learning rate: **0.1**, divide by 10 at **81, 122** epoch
- Weight decay = **0.0001**
- Weight initialization: torch.nn.init.kaiming_normal
- Loss function: cross entropy

4. Comparison figure

- Compare to **vanilla CNN-18**



- Compare to **vanilla CNN-50**



5. **Methodology**
   - ResNet 18 got 92.37% accuracy, Time: 0.58 hr
   - ResNet 50 got 93.53% accuracy, Time: 1.48 hr
   - On single Titan X (Maxwell)

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)

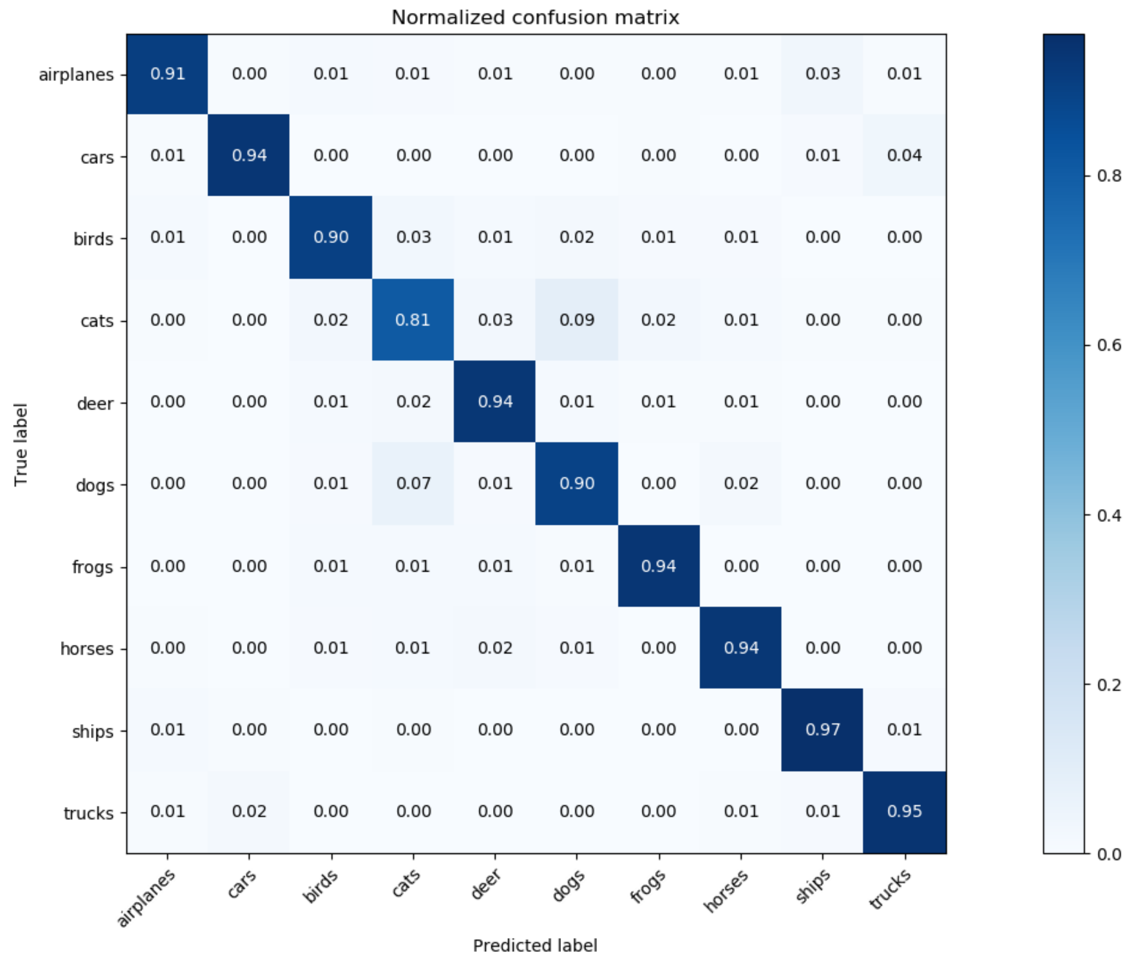
Criterion of result

Accuracy	> 91%	= 100%
Accuracy	: 91% ~ 90%	= 90%
Accuracy	: 90% ~ 87%	= 80%
Accuracy	: 87% ~ 10%	= 70%
Accuracy	: 10%	= 0%

評分標準：40%*實驗結果 + 60%*(報告+DEMO)

Bonus (+5) – Confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC curve.



- **Sample code:**
  You can use the following example and library to conduct this section.
  https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py

Reference:

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