CNN Architectures

PHYS591000 2022.04.20

Warming up

- As usual, take 3 mins to introduce yourself to your teammate for this week!
  – “It’s midterm period! How are you doing?”
  – “Gee you’ve got 3 midterm exams this week? Let’s work together to make life easier!”

Outline

- CNN is one of the most widely used NN in real applications nowadays. Hence there are many popular models of CNN.

- Today we're going to give a brief introduction on

    - AlexNet
    - VGG
    - GoogLeNet
    - ResNet

Ref: Lecture 9 of CS231(2017) at Stanford (link to youtube).

LeNet-5 (Yann LeCun et al. 1989)

- First 'real' implementation of CNN for MNIST classification
    - 5x5 filter; Sigmoid (activation); Average Pooling; 7 layers

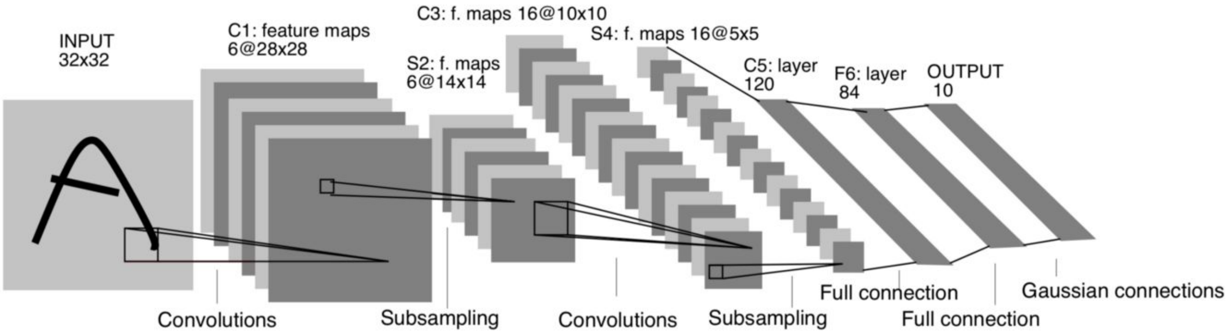
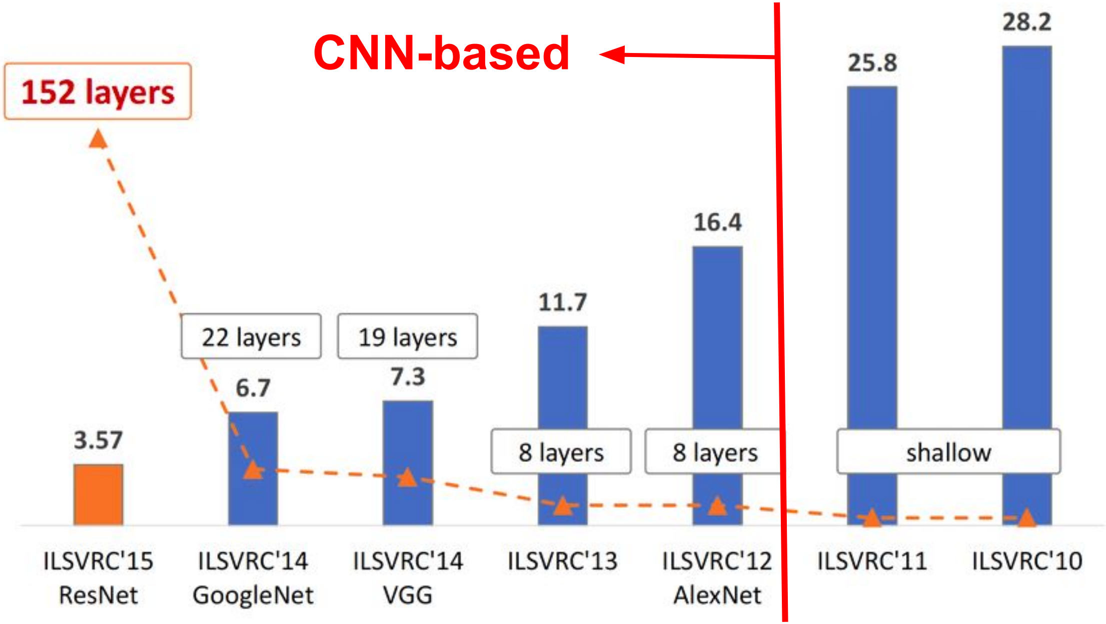


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Evolution of ILSVRC Winners

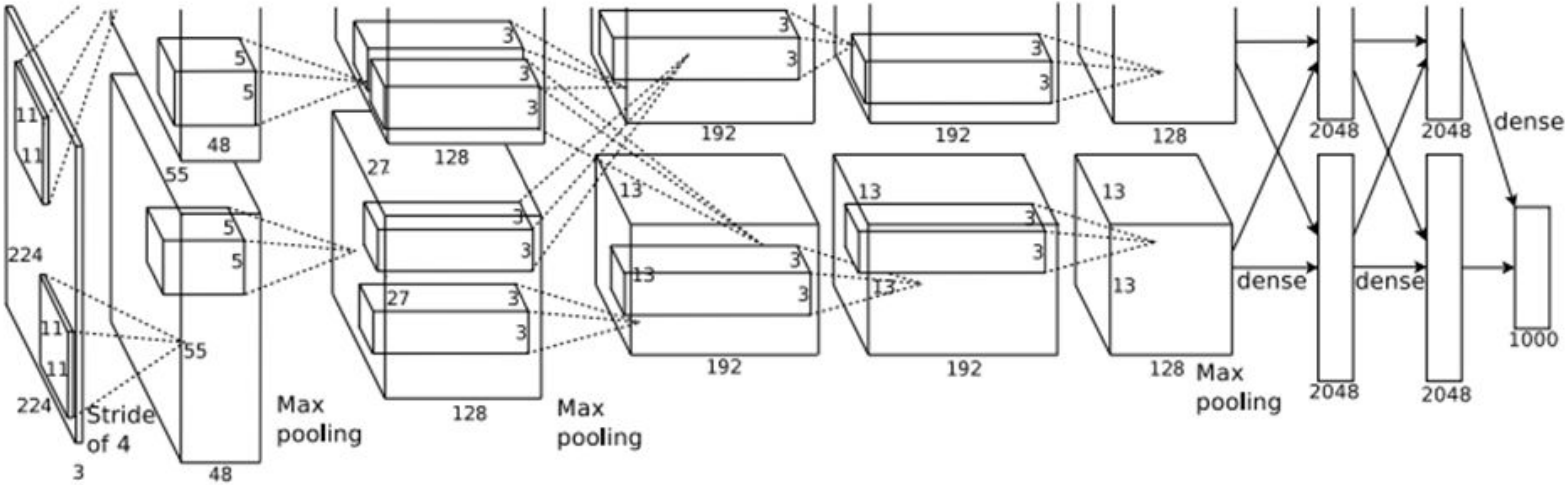
ImageNet Large Scale Visual Recognition Challenge

- One of the most important AI visual recognition competitions



AlexNet (2012)

- First CNN-based ILSVRC winner
    - Input: 227x227x3 images

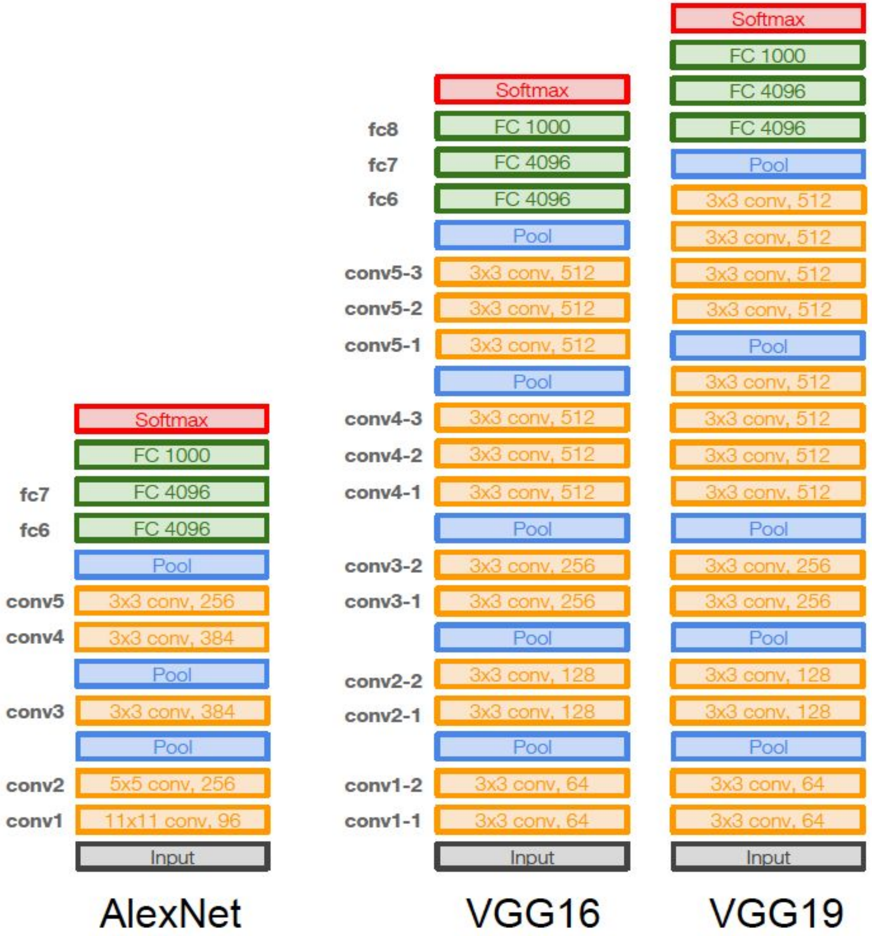


AlexNet (2012)

- First CNN-based ILSVRC winner (trained w/ 2 GPU’s)
  - 8 Layers (5 Conv and 3 Fully-connected (FC)).
  - First use of ReLU (to overcome the ‘vanishing gradient’ problem of sigmoid).
  - Max pooling
  - Dropout (between the FC layers)

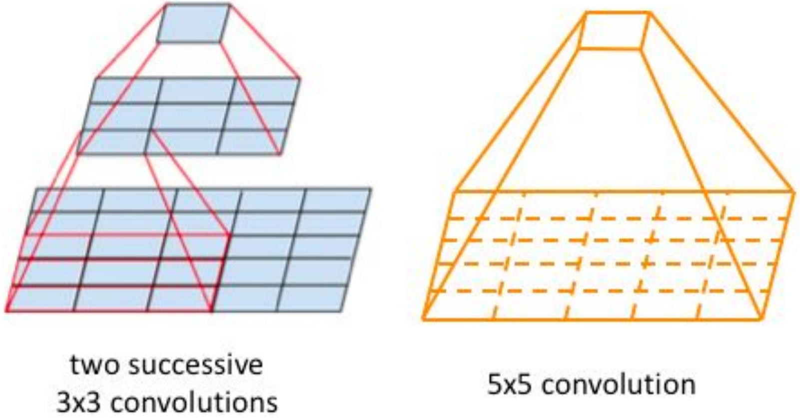
VGG (2014, 2nd place)

- Deeper networks with smaller filters
    - Only use 3x3 filters
    - 2x2 max. pooling
    - Use padding to preserve sizes of feature maps (the same as input)



VGG (2014, 2nd place)

- Stacking smaller filters can have the same effect as a larger filter.
  - Applying two 3x3 filters twice is equivalent to one convolution with a 5x5
  - But with fewer parameters: 3x3x2 < 5x5

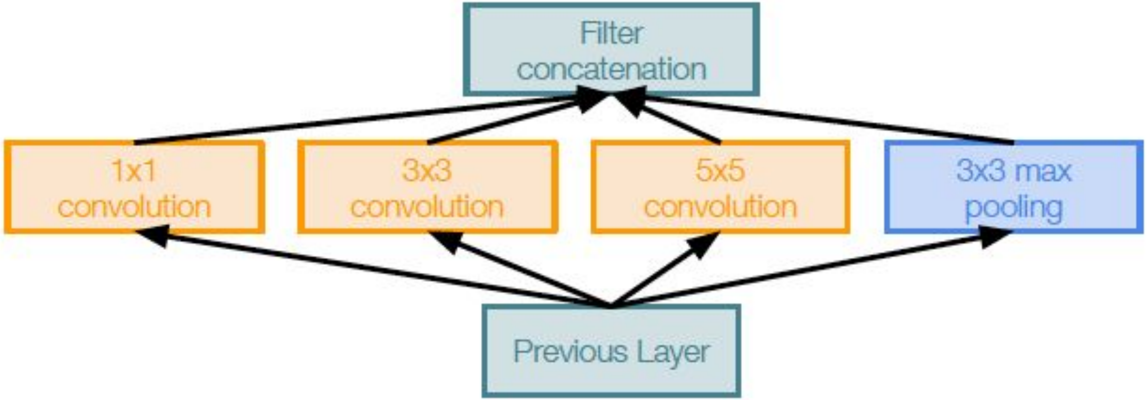


GoogLeNet (2014 winner)

- Deeper (22 layers) with computational efficiency
    - Inception module: an example of ‘Network-in-Network’
    - Dimension reduction with 1x1 conv ‘bottleneck’ layer
    - No FC layer – replaced by Global Average Pooling

GoogLeNet (2014 winner)

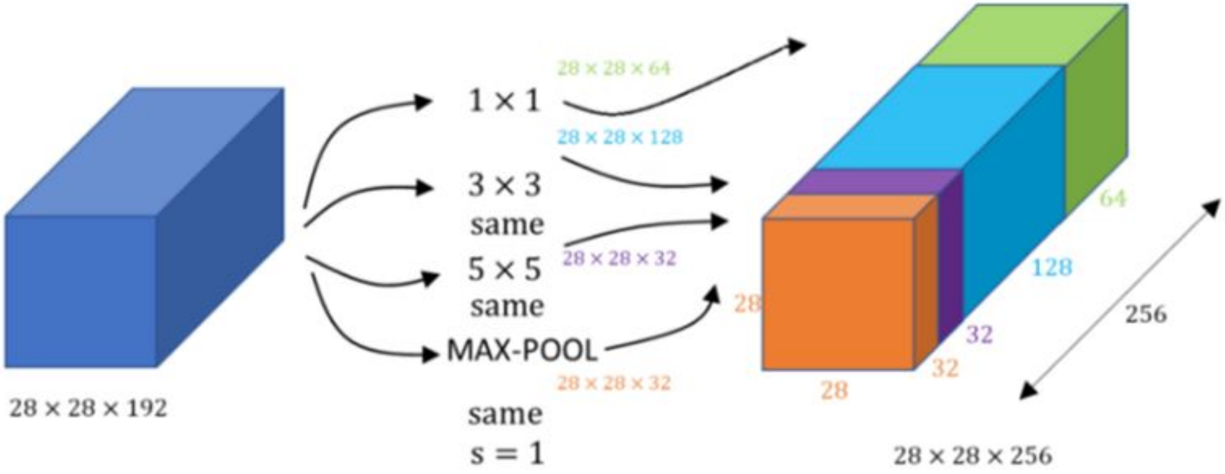
- Inception module: parallel filter operations



Naive Inception module

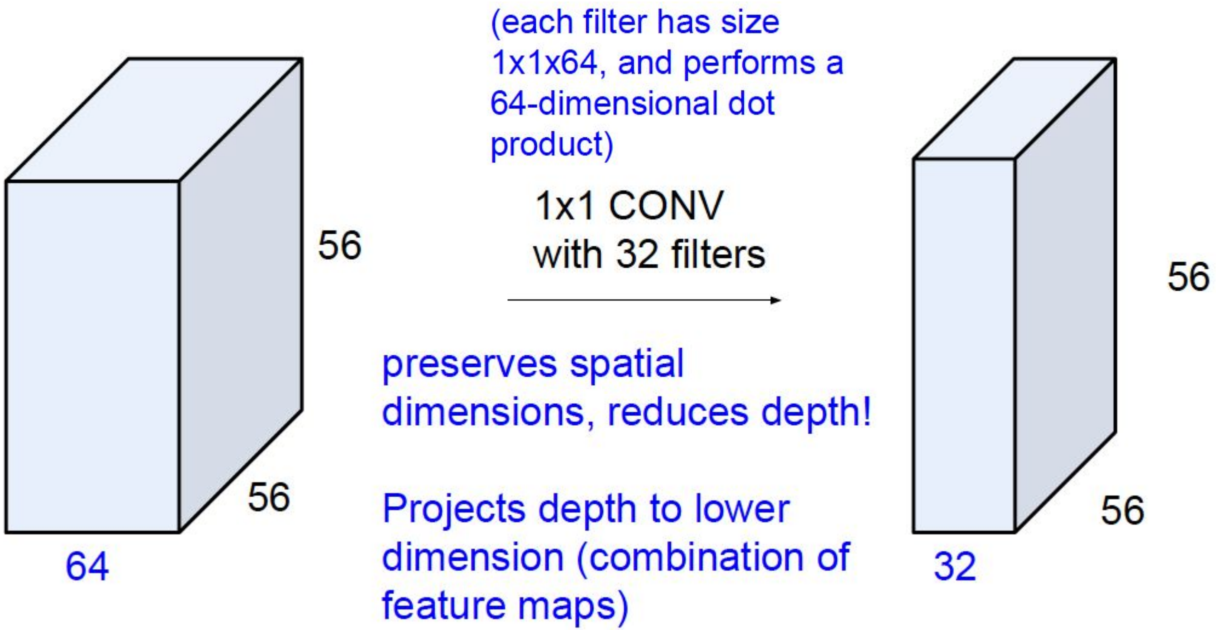
GoogLeNet (2014 winner)

- Problem: Very expensive in computing
  – Contain a lot of parameters
  – Depth grows up after each layer



GoogLeNet (2014 winner)

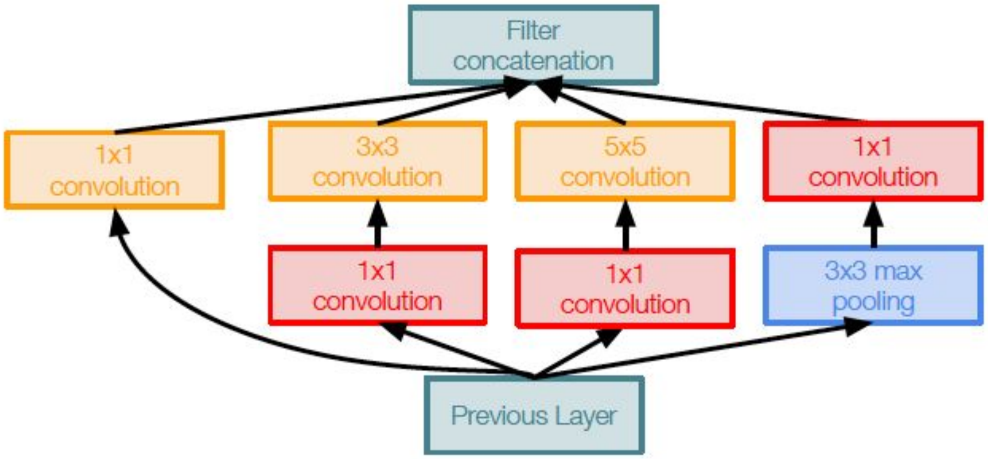
- Solution: 1x1 conv 'bottleneck' layer for dimension reduction



GoogLeNet (2014 winner)

- Inception module with bottleneck layers

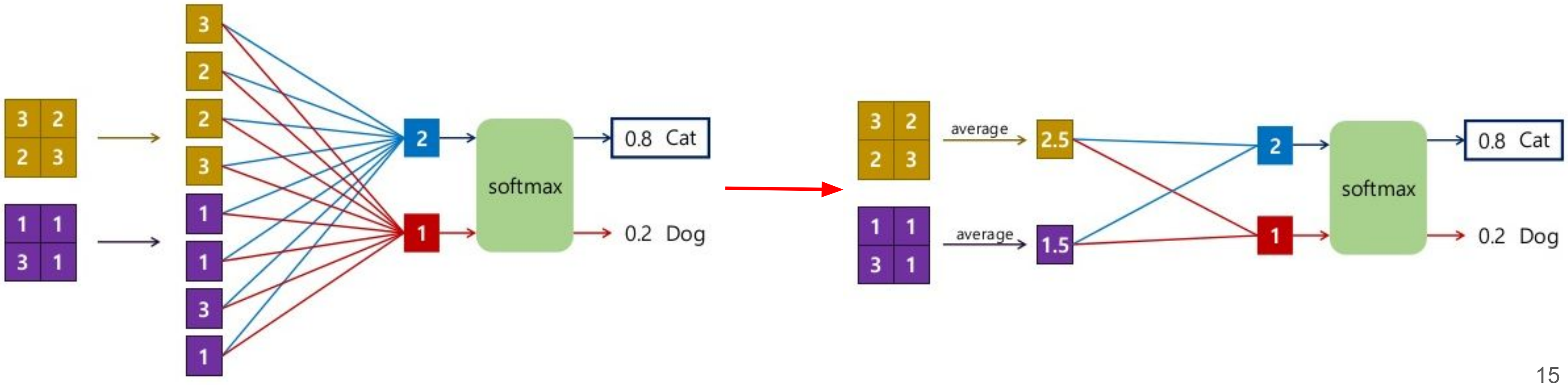
1x1 conv "bottleneck" layers



Inception module with dimension reduction

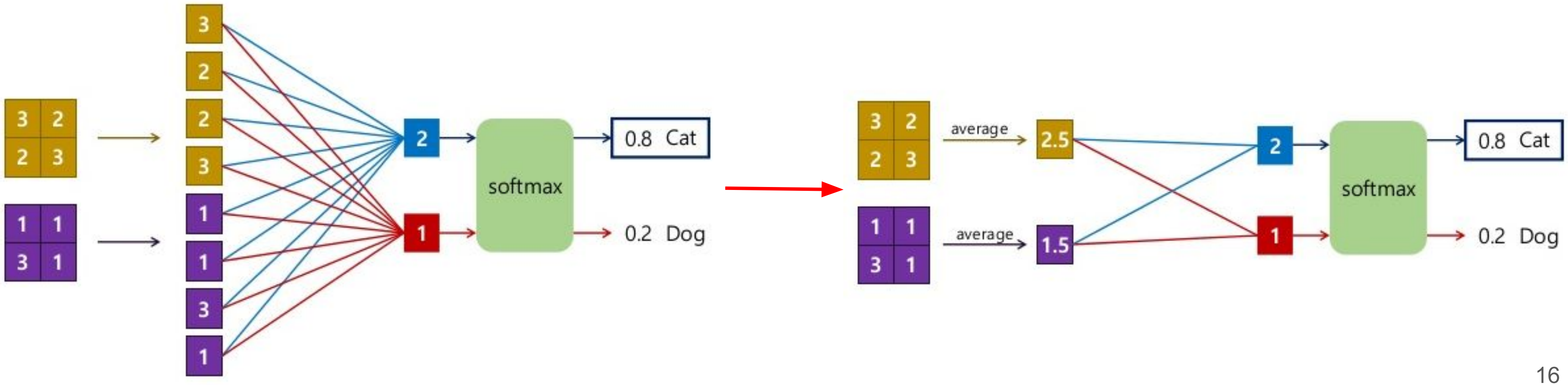
GoogLeNet (2014 winner)

- FC layers are expensive in computing → replace FC layers by Global Average Pooling

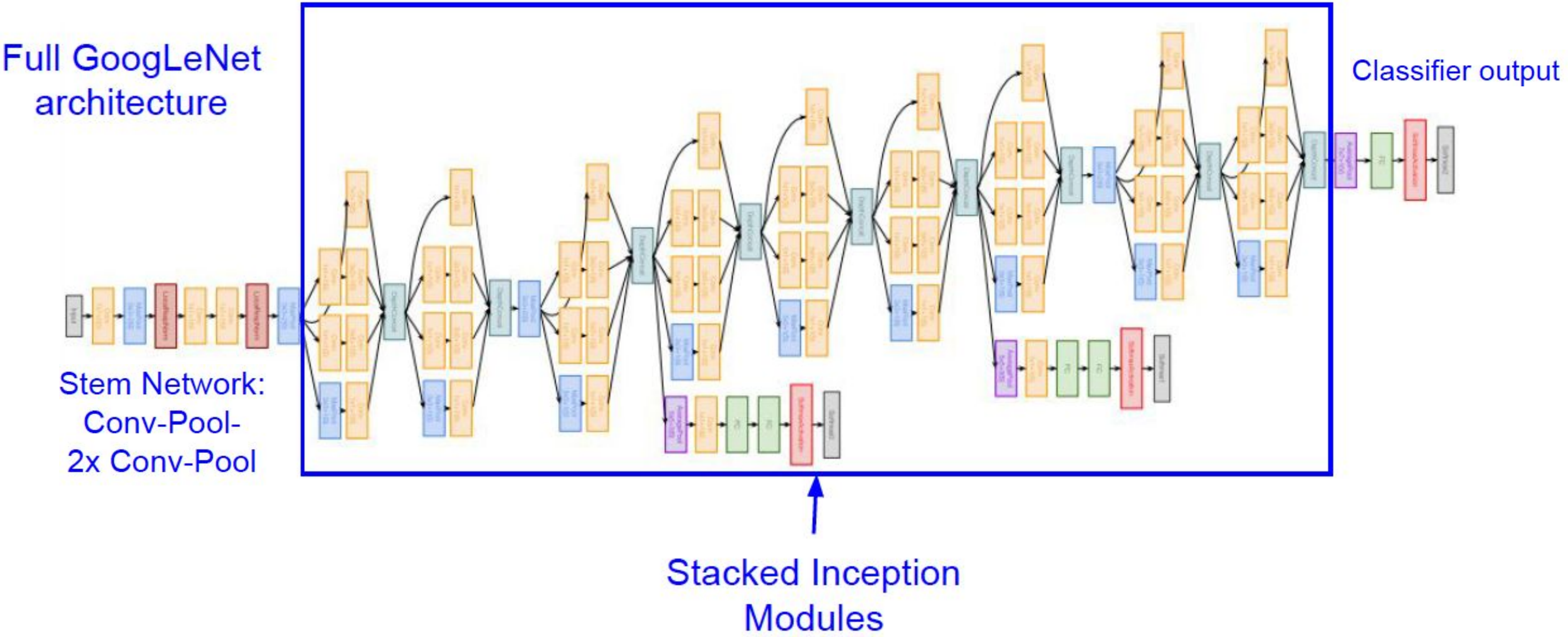


GoogLeNet (2014 winner)

- Global Average Pooling: Average over the whole feature map – reduce parameters/dimension, less prone to overfitting

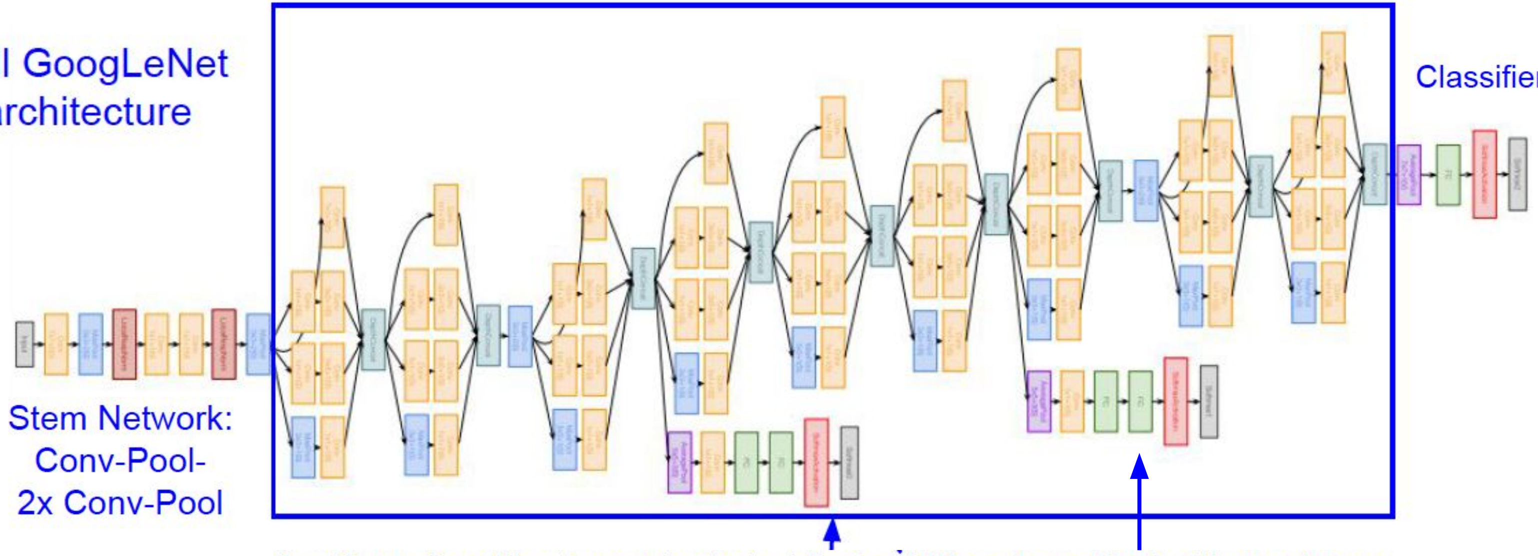


GoogLeNet (2014 winner)



GoogLeNet (2014 winner)

Full GoogLeNet architecture



Classifier output

Stem Network: Conv-Pool- 2x Conv-Pool

Auxiliary classification outputs to inject additional gradient at lower layers

Stacked Inception Modules

ResNet (2015 winner)

- Revolution of Depth: 152 layers!
  (Other versions nowadays: ResNet-50, ResNet-101, etc.)

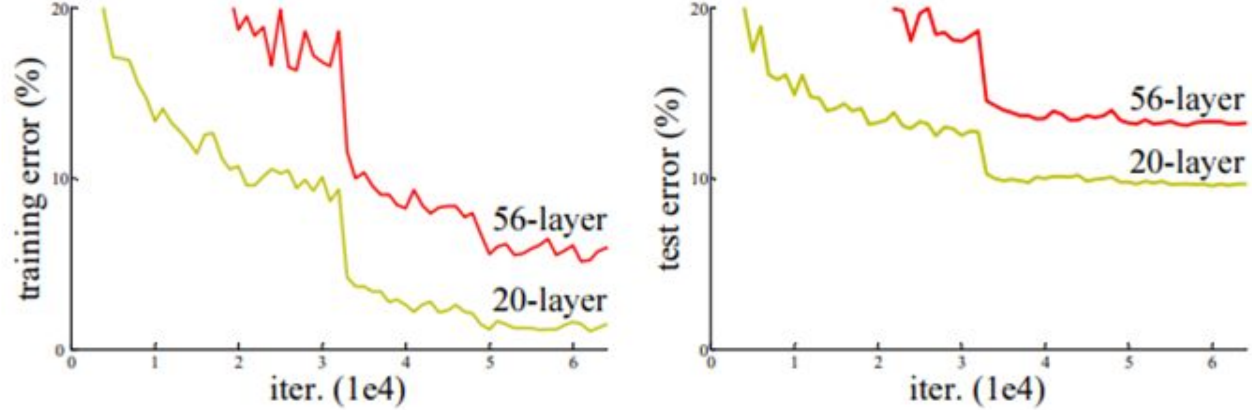
- Error rate ~3.6% (Human: ~5.1%)

- Overcome the vanishing gradient problem of deep(er) NN using *Residual blocks*

ResNet (2015 winner)

- Deeper NN are more difficult to optimize
  - More parameters to be optimized
  - Vanishing gradient problem

→ Result in poorer performances than shallow networks.

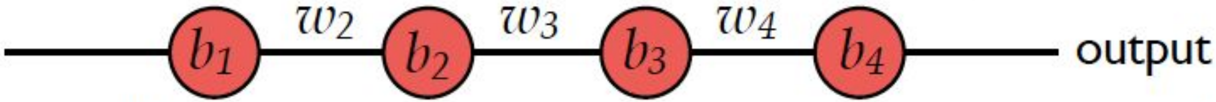


ResNet (2015 winner)

Courtesy of Prof. Kai-Feng Chen (NTU)

- Let's consider a chain of neurons and calculate the gradient according to back propagation:

$$\frac{\partial L}{\partial b_4} = \sigma'(z_4) \cdot \frac{\partial L}{\partial y}$$



$$\frac{\partial L}{\partial b_1} = \sigma'(z_1) \cdot w_2 \cdot \sigma'(z_2) \cdot w_3 \cdot \sigma'(z_3) \cdot w_4 \cdot \sigma'(z_4) \cdot \frac{\partial L}{\partial y}$$

Generally the weights are small (<1) after training, and $\sigma'(z)$ is less then 0.25 by definition, if the sigmoid function is used. This will enforce $\frac{\partial L}{\partial b_1} < 0.0156 \frac{\partial L}{\partial b_4}$

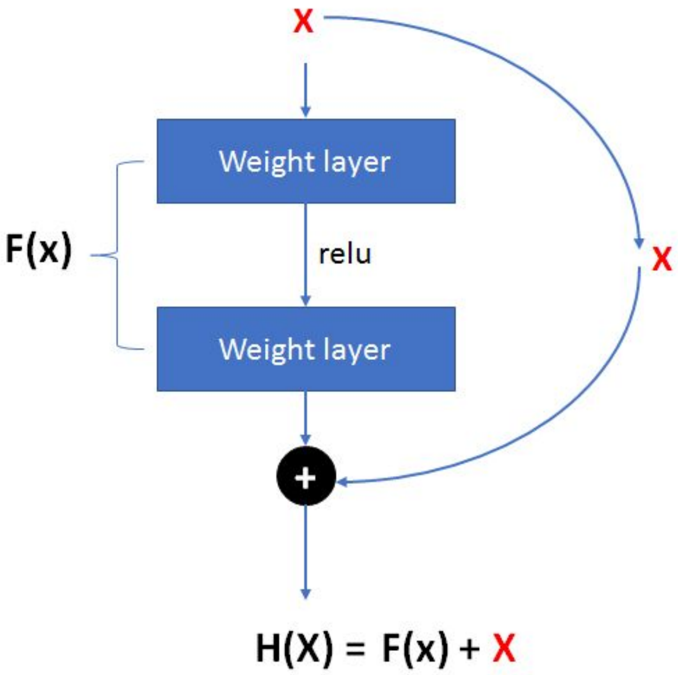
The updating on b_1 will be much slower than b_4.

ResNet (2015 winner)

- Idea: Copying what was learned in previous layers (shallower models).

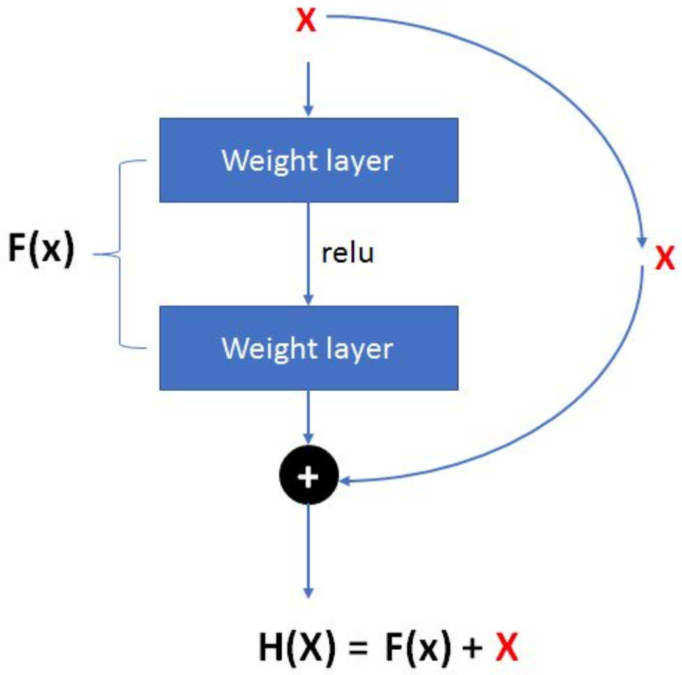
  Only need to learn (fit) the difference (*residual*) between the input (from previous layer) and the output.

  → Residual block



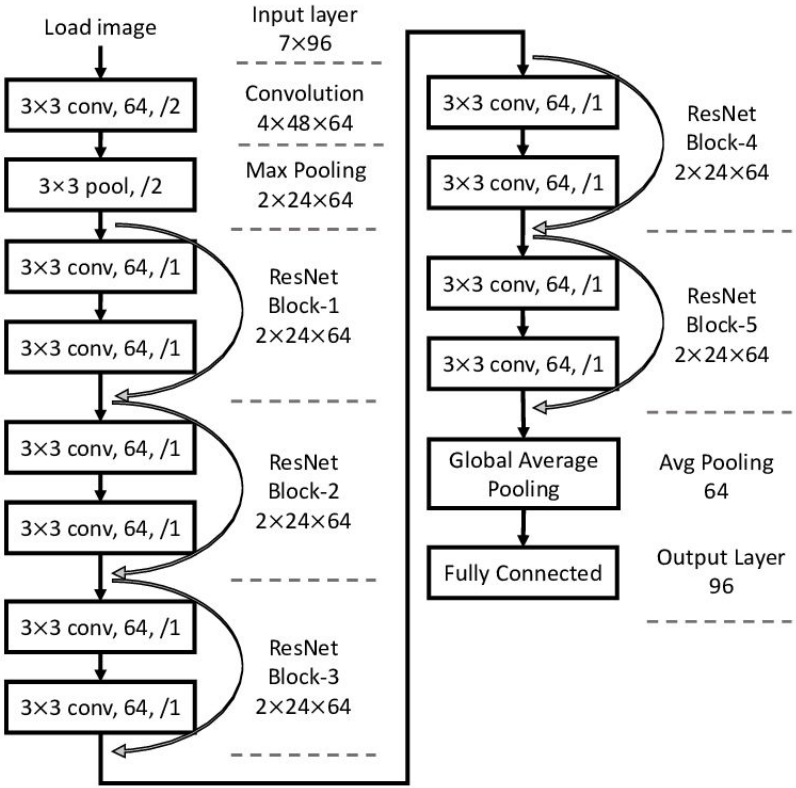
ResNet (2015 winner)

- Residual block: Fit F(x) = H(x) - x (Output - Input)
  - When gradient is vanishing F(x) will be fit to 0
  - The output will thus be the same as previous layer (restore gradients to the values in previous layers).



ResNet (2015 winner)

- ResNet: Stack of residual blocks
    - Also employ 1x1 conv bottleneck layers in residual blocks
    - Global Avg pooling before the output
    - No FC except the output
    - No dropout

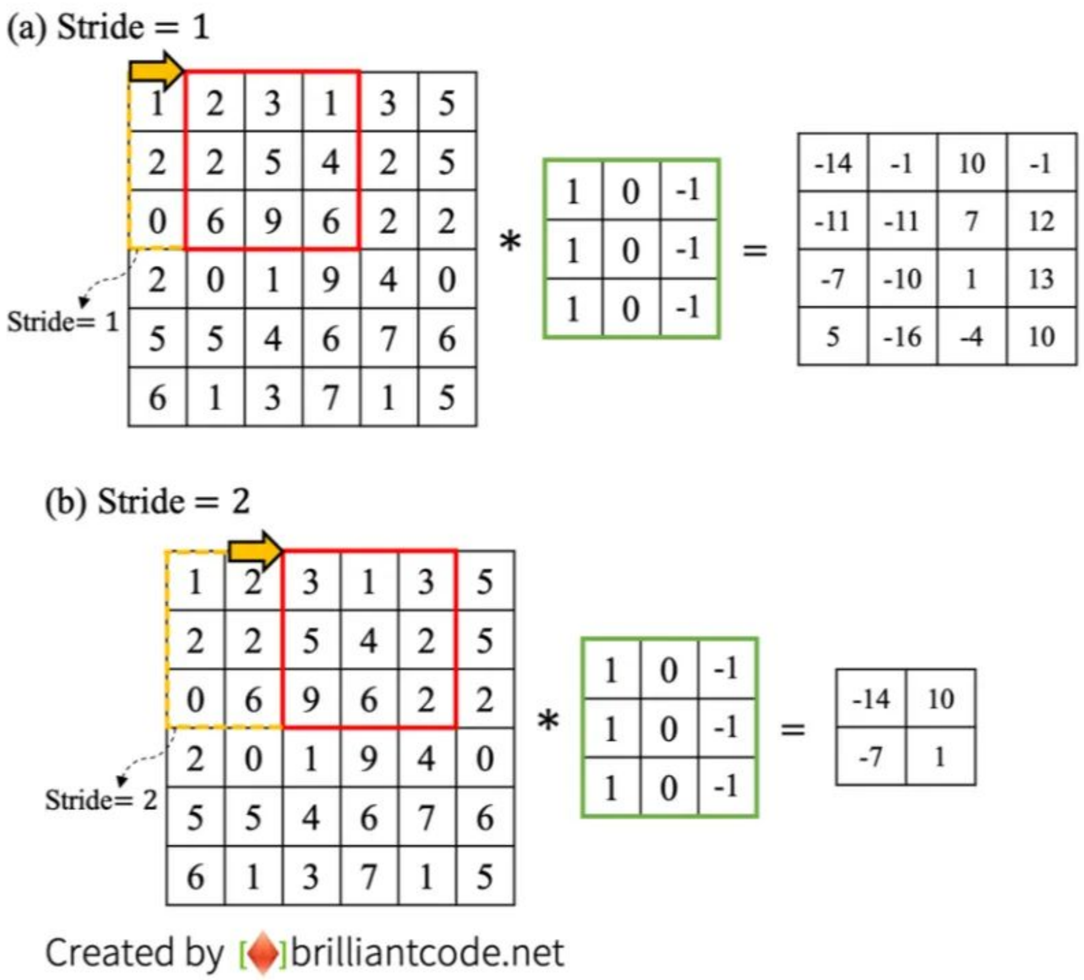


Lab for this week

- No in-class exercise this week
- For the Lab session, you’ll going to implement AlexNet, VGG, GoogLeNet, and ResNet for a classification task with OxFlower17, a dataset with images of 17 different categories of flowers.
- Furthermore, you’ll learn how to increase the amount of training data ‘by hand’ (**data augmentation**).

Backup

CNN Strides


(a) Stride = 1
1 2 3 1 3 5
2 2 5 4 2 5
0 6 9 6 2 2
2 0 1 9 4 0
5 5 4 6 7 6
6 1 3 7 1 5
Stride= 1
*
1 0 -1
1 0 -1
1 0 -1
=
-14 -1 10 -1
-11 -11 7 12
-7 -10 1 13
5 -16 -4 10
(b) Stride = 2
1 2 3 1 3 5
2 2 5 4 2 5
0 6 9 6 2 2
2 0 1 9 4 0
5 5 4 6 7 6
6 1 3 7 1 5
Stride= 2
*
1 0 -1
1 0 -1
1 0 -1
=
-14 10
-7 1
Created by brilliantcode.net

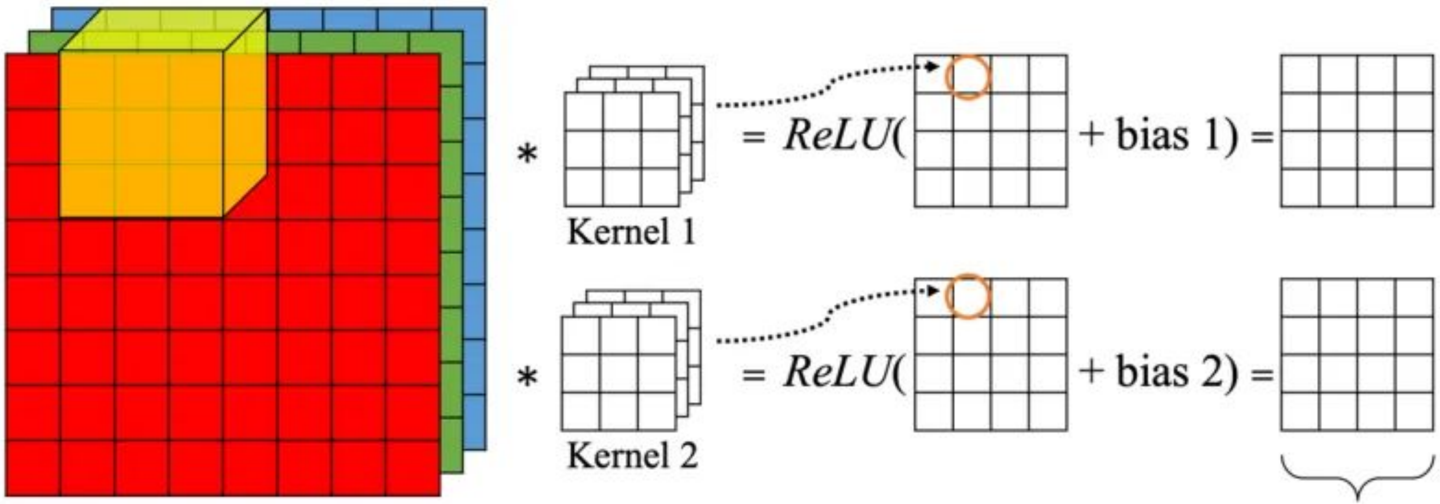
CNN Padding

- We can apply padding to allow more space for the filter to cover the image and preserve the size of feature maps.

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 3 & 1 & 3 & 5 & 0 \\
0 & 2 & 2 & 5 & 4 & 2 & 5 & 0 \\
0 & 0 & 6 & 9 & 6 & 2 & 2 & 0 \\
0 & 2 & 0 & 1 & 9 & 4 & 0 & 0 \\
0 & 5 & 5 & 4 & 6 & 7 & 6 & 0 \\
0 & 6 & 1 & 3 & 7 & 1 & 5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
1 & 0 & -1 \\
1 & 0 & -1 \\
1 & 0 & -1
\end{bmatrix}
=
\begin{bmatrix}
-4 & -5 & -1 & 3 & -5 & 5 \\
-10 & -14 & -1 & 10 & -1 & 7 \\
-8 & -11 & -11 & 7 & 12 & 8 \\
11 & -7 & -10 & 1 & 13 & 13 \\
-6 & 5 & -16 & -4 & 10 & 12 \\
-6 & 4 & -7 & -1 & 2 & 8
\end{bmatrix}
$$

CNN image channel

- A color image has three channels (R/G/B) and thus needs three layers of kernels.



CNN hyperparameters

- Filter/Kernel size (height and width of the kernel)
- Strides and padding
- Data format/channel numbers
- Ways of pooling
- Other hyperparameter for the (fully-connected) NN, e.g. activation functions.