AI on Chip
Lab1

2022/02/17

TAs: course.aislab@gmail.com

Outline

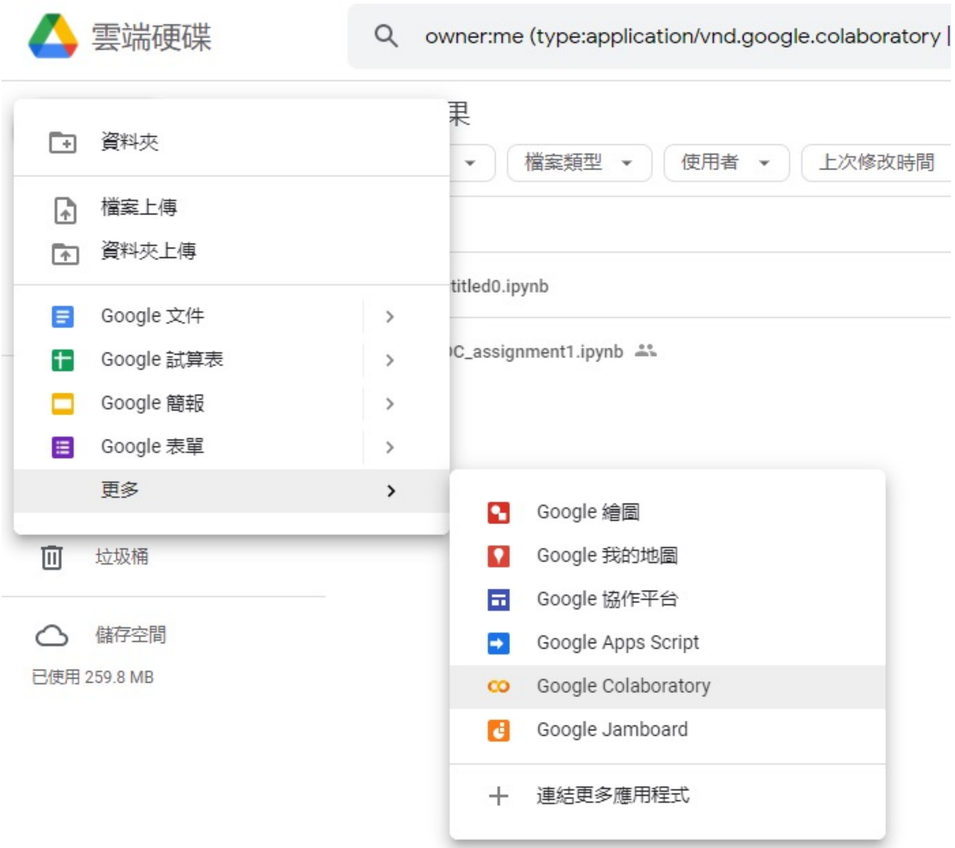
- Google Colab
- What is tensorflow?
- Neural Networks(NN)
- Convolutional Neural Networks(CNN)
- Dataset
- Advanced Topics

Google Colab

- Google offer Jupyter Notebook develop environment
- Free 12GB GPU(Tesla K80)
- 50GB storage
- 12 hours continuous using limited.
- Idling over 90 minutes will be kick out by host. Need to reconnect.
- Google Colab run in Ubuntu Linux

Google Colab

- Goto Google Drive and new Google Colaboratory file

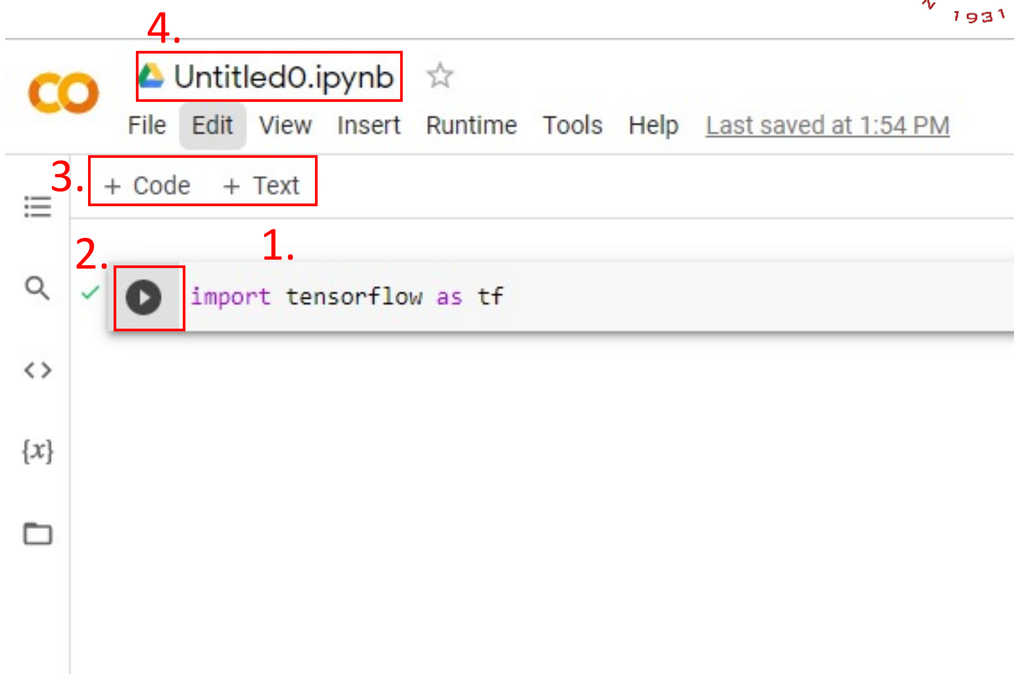


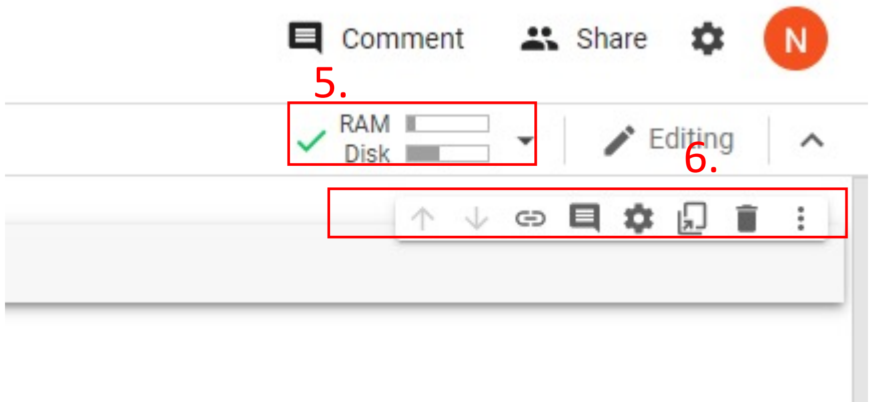
First open Colab file

- It will show Jupyter Notebook like page

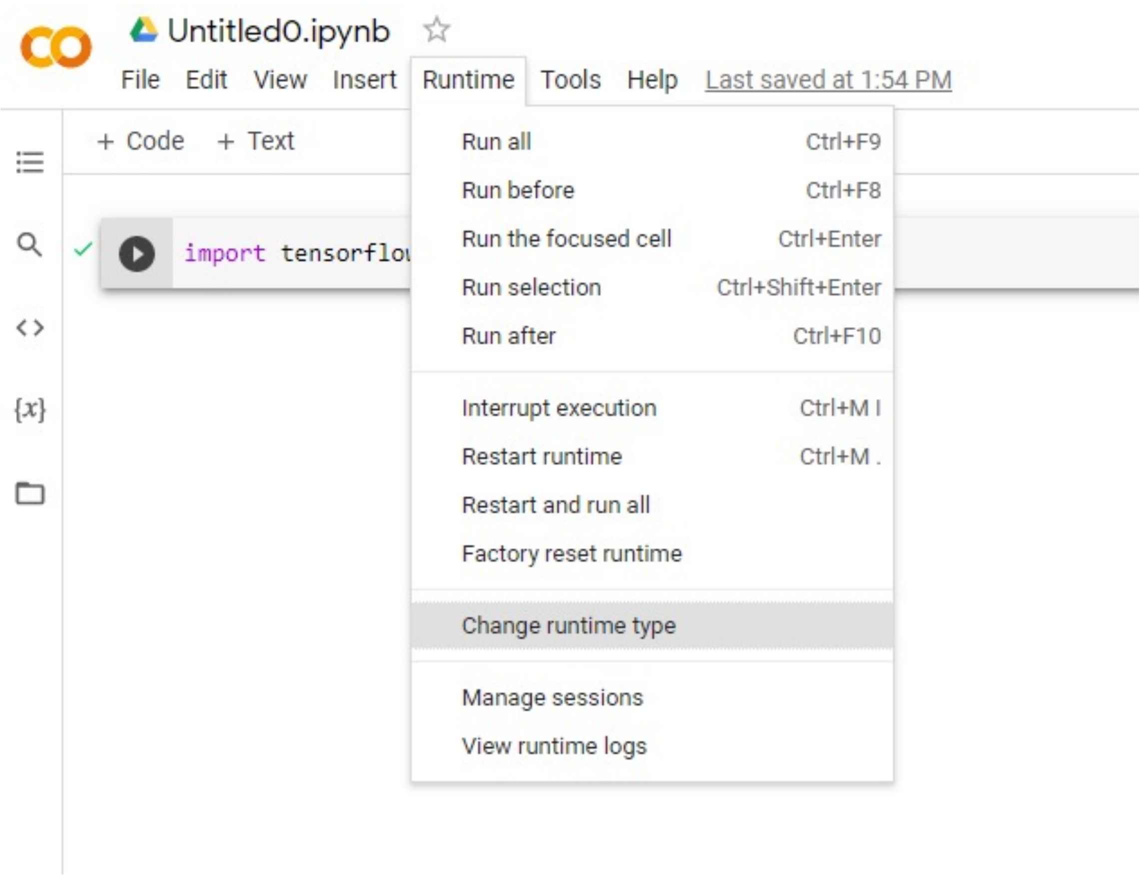
- After creating a file

    1. Code section
    2. Run cell (run your code)
    3. Append Code section or Text section
    4. Rename
    5. Resource usage
    6. Code section operation

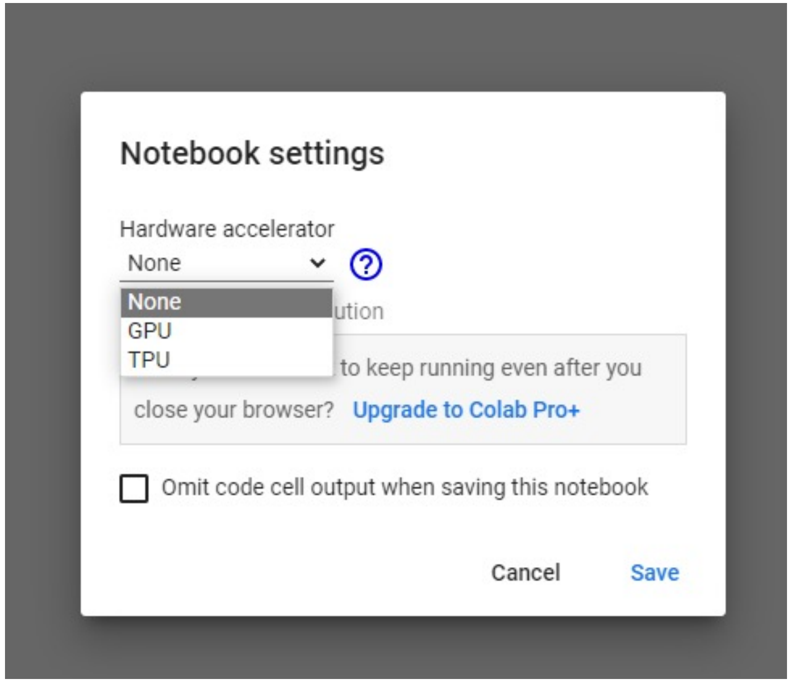




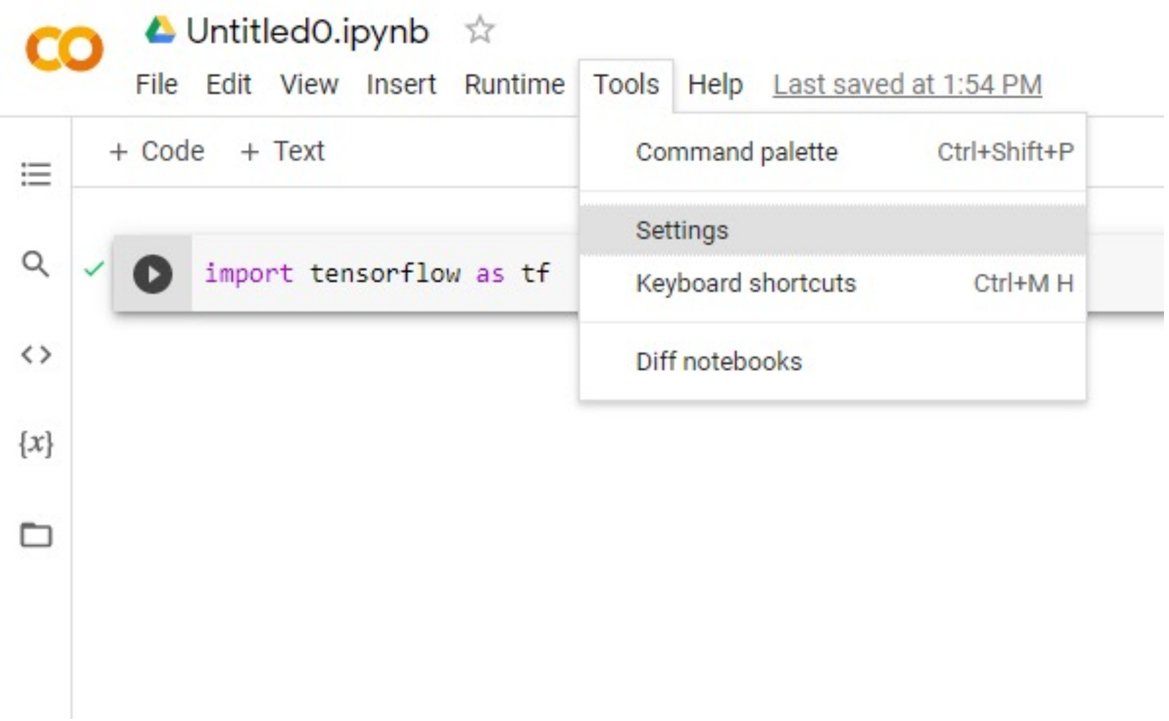
Choose GPU as runtime



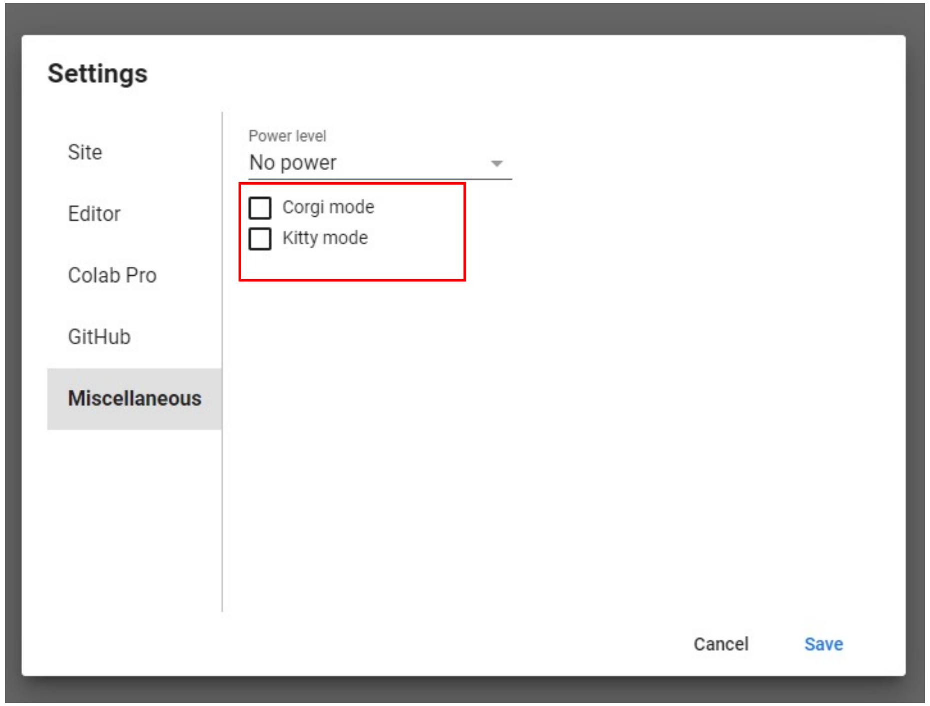
Default runtime: CPU
You can change it as needed
Runtime -> Change runtime type -> GPU



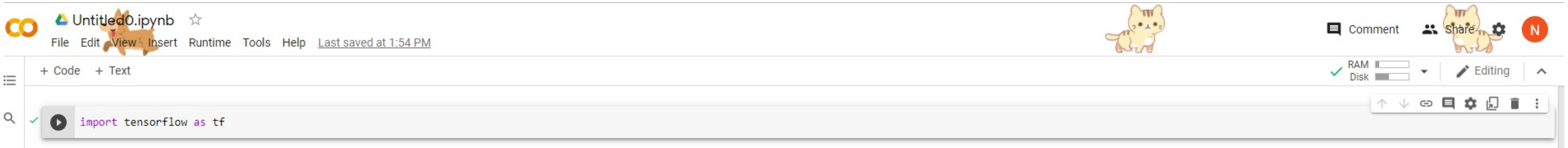
Choose a pet



Untitled0.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:54 PM
+ Code + Text
Command palette Ctrl+Shift+P
Settings
Keyboard shortcuts Ctrl+M H
Diff notebooks
import tensorflow as tf



Settings
Site
Editor
Colab Pro
GitHub
Miscellaneous
Power level
No power
Corgi mode
Kitty mode
Cancel
Save



Untitled0.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:54 PM
+ Code + Text
Comment
Share
RAM
Disk
Editing
import tensorflow as tf

If you want to use shell instruction in Colab

Add ! For most shell instruction

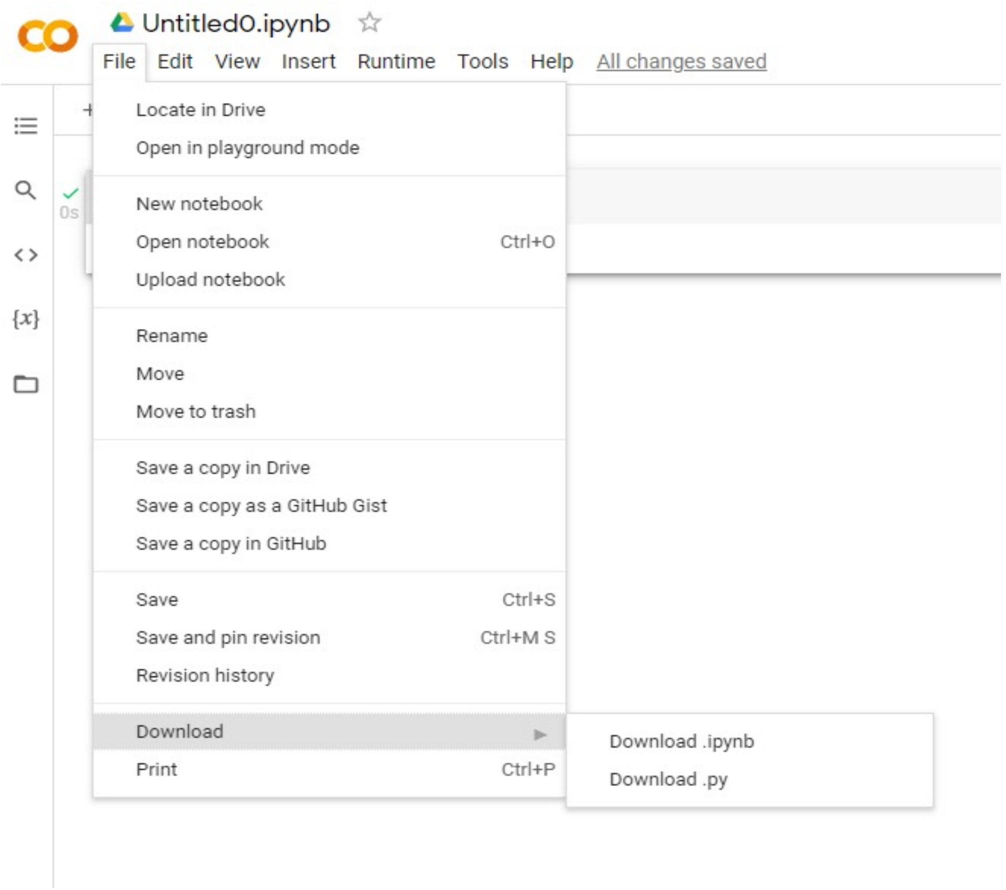
Ex: !ls -a, for list all directories

```
!ls -a
```

```
.  ..  .config  sample_data
```

Google Colab - Save & Export

Save: *.ipynb



Export: export as HTML, LaTex, Markdown, PDF, Python…

What is Tensorflow?

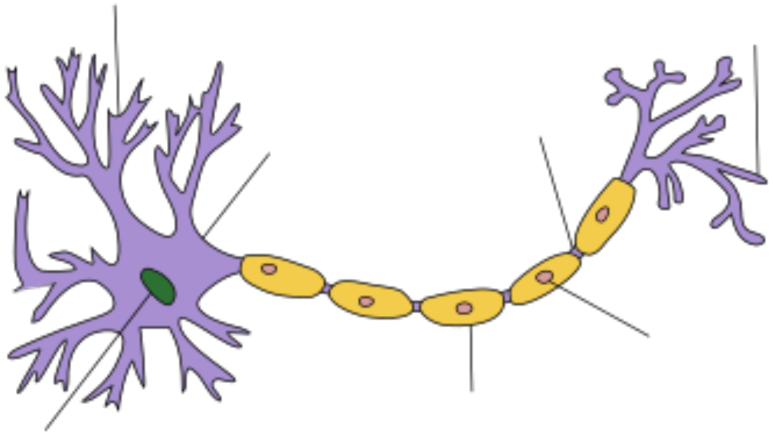
- https://www.tensorflow.org/
- Developed by Google Brain
- The most popular machine learning framework amongst ML developers.
- Other framework: PyTorch, Caffe …

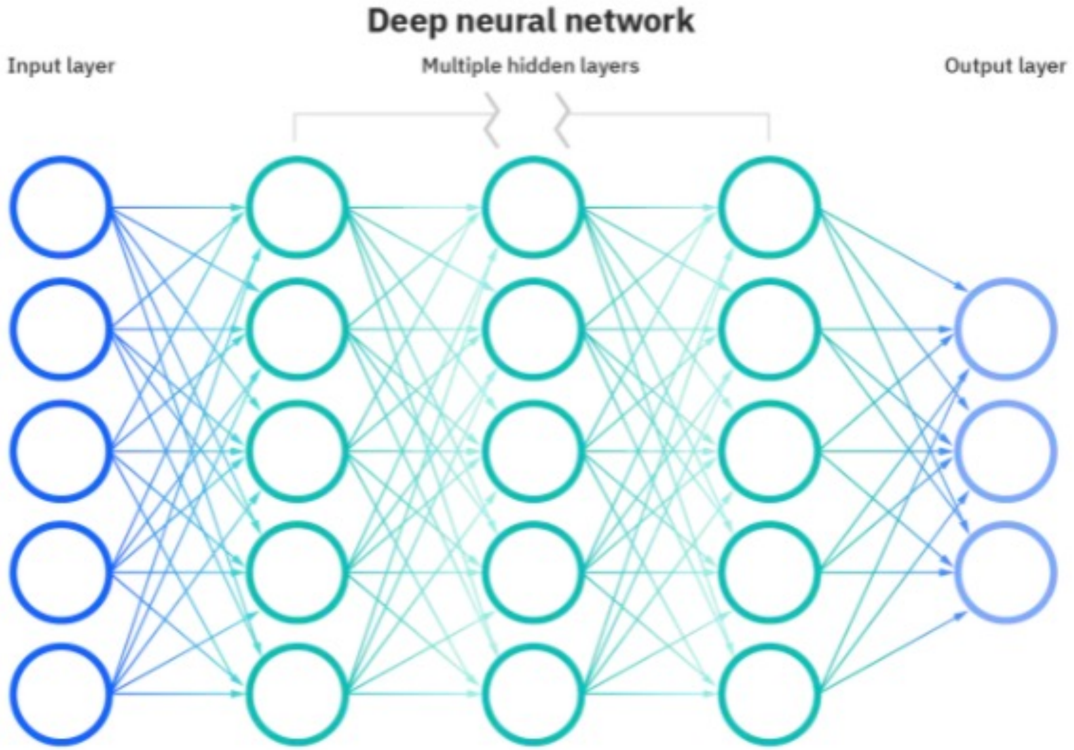
- Framework:
  - A predefined frame that work, also offer implementation of code and tell you how to use it.

API Documentation | TensorFlow Core v2.8.0

Neural Networks(NN)

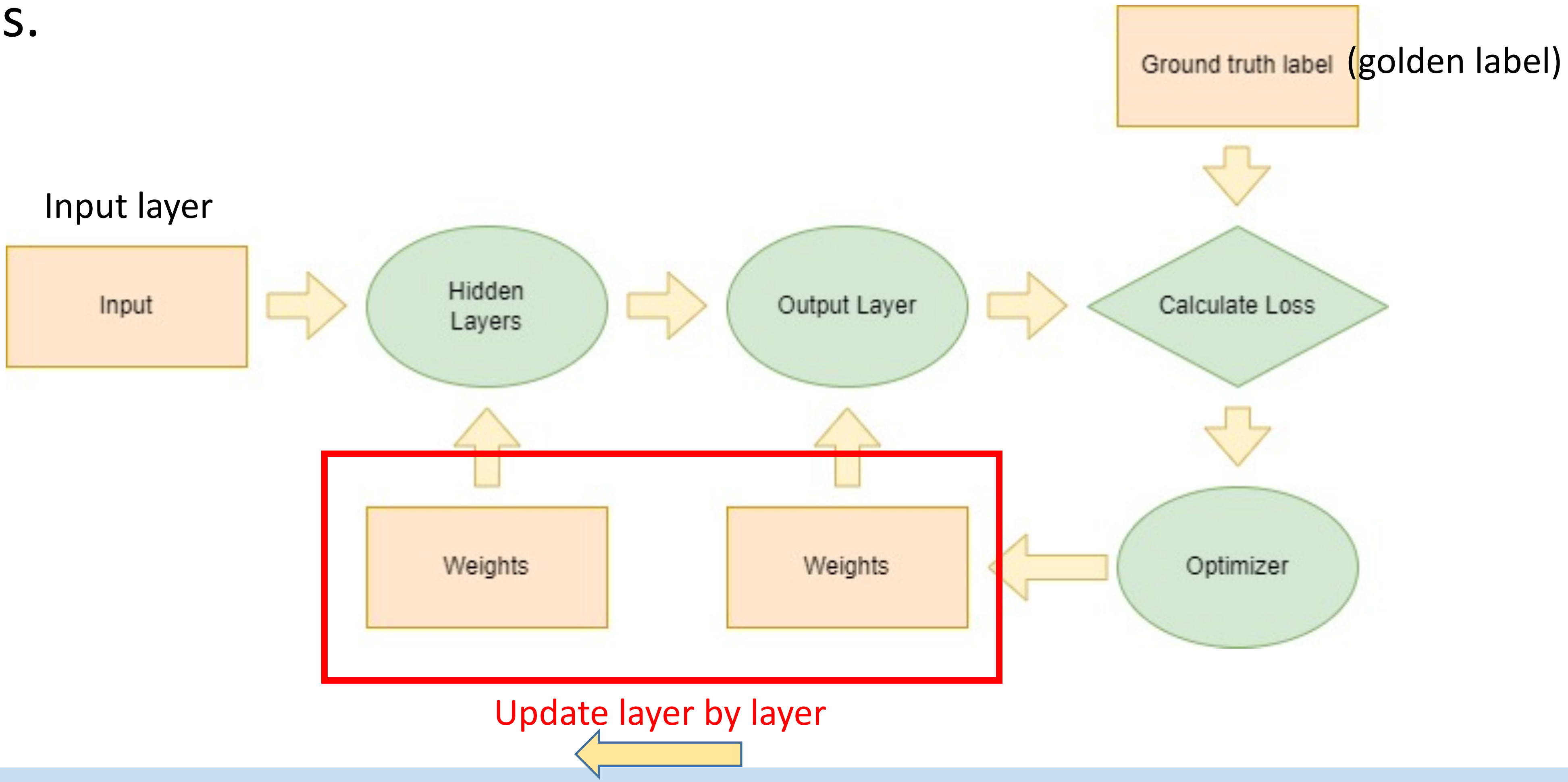
- Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.





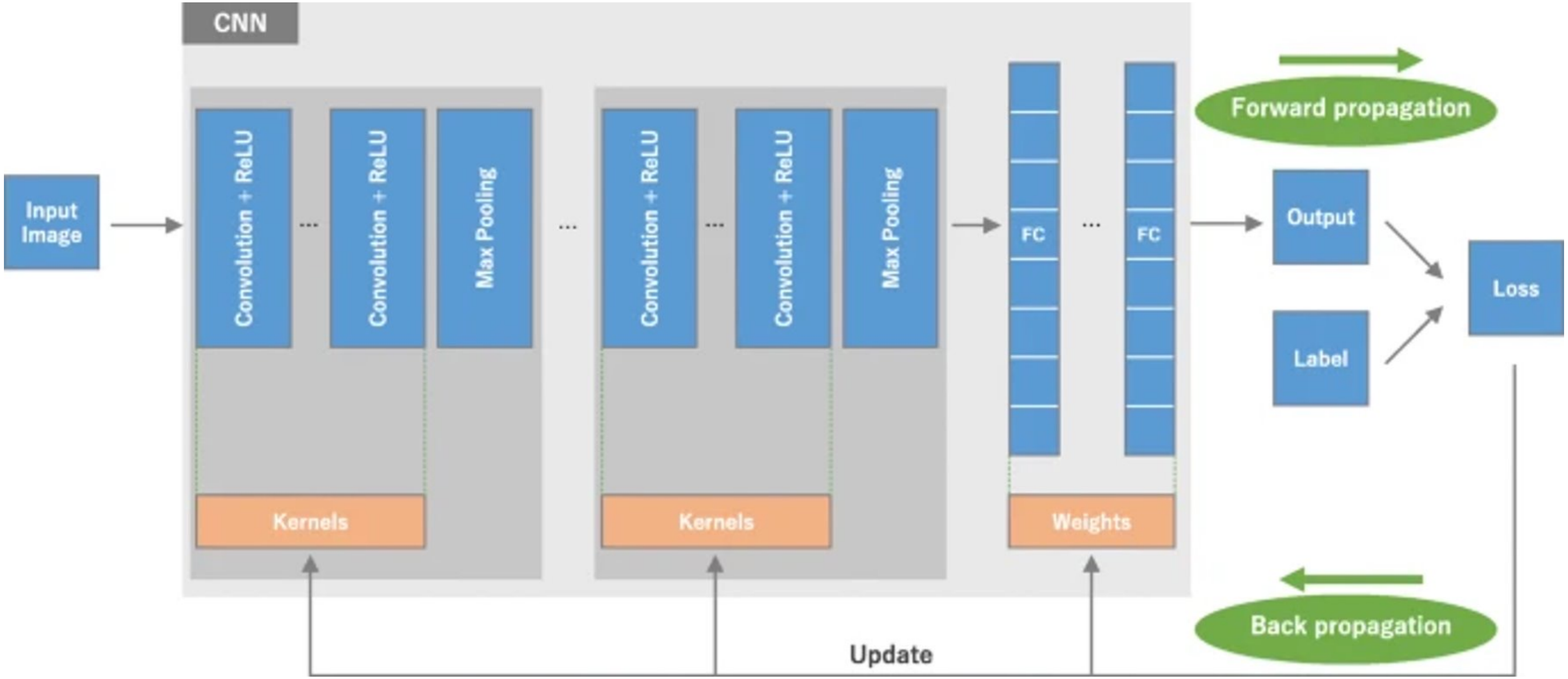
How Neural Networks(NN) work?

- Neural Networks in general are composed of a collection of neurons that are organized in layers, each with their own learnable weights and biases.



What is CNN (Convolutional Neural Network)?

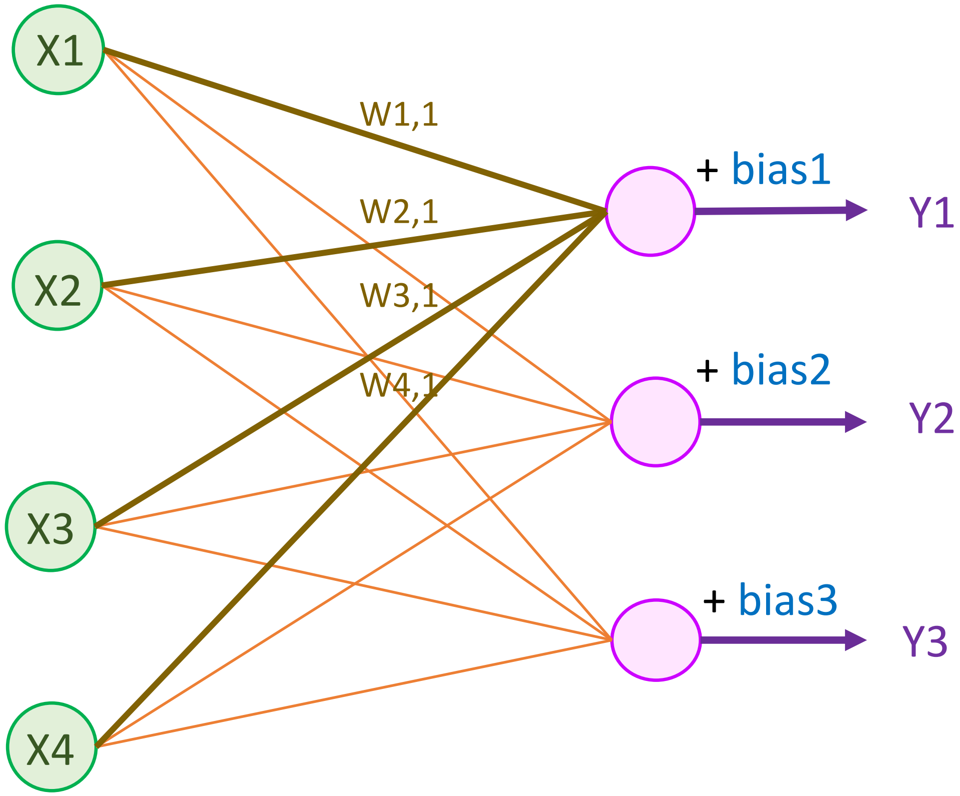
- A CNN is a neural network: An algorithm used to recognize patterns in data.



Convolution, Activation, Pool, Fully connection could repeat many times

Fully Connection

- A fully connected neural network consists of a series of fully connected layers that connect every neuron in one layer to every neuron in the other layer.

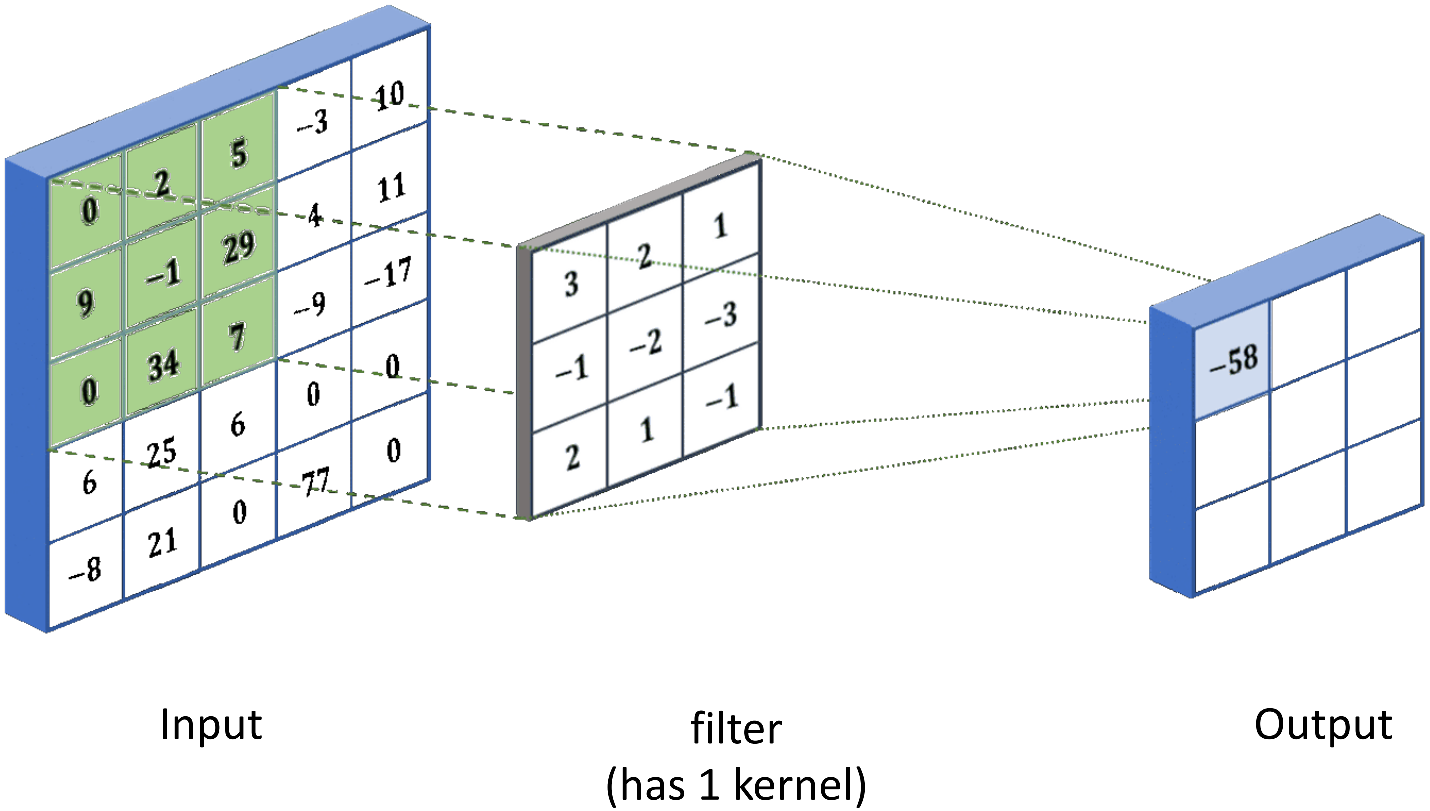


Y1 = X1*W1,1 + X2*W2,1 + X3*W3,1 + X4*W4,1+ bias1

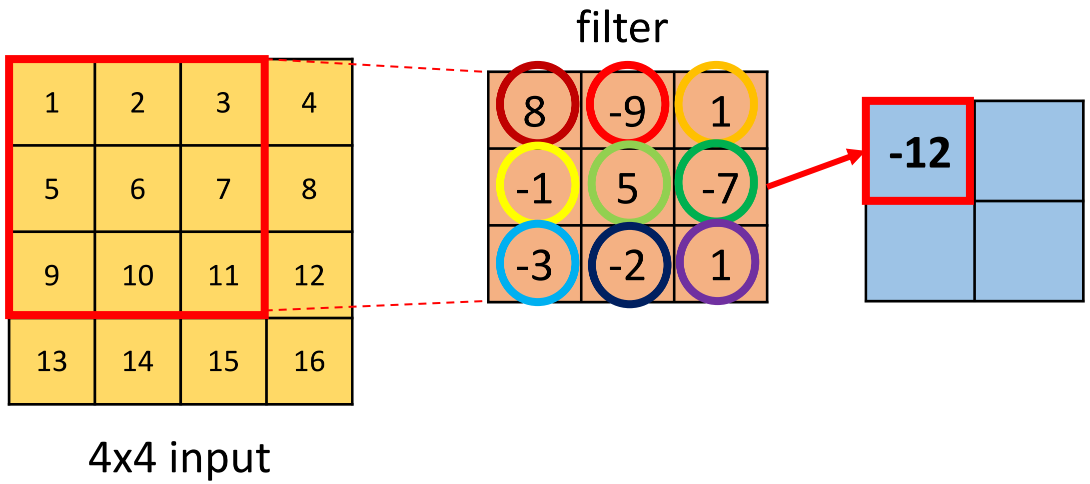
$$\begin{bmatrix} W1{,}1 & \cdots & W4{,}1 \\ \vdots & \ddots & \vdots \\ W1{,}3 & \cdots & W4{,}3 \end{bmatrix} \begin{bmatrix} X1 \\ \dots \\ X4 \end{bmatrix} + \begin{bmatrix} \text{bias1} \\ \dots \\ \text{bias3} \end{bmatrix} = \begin{bmatrix} Y1 \\ \dots \\ Y3 \end{bmatrix}$$

Convolution layer

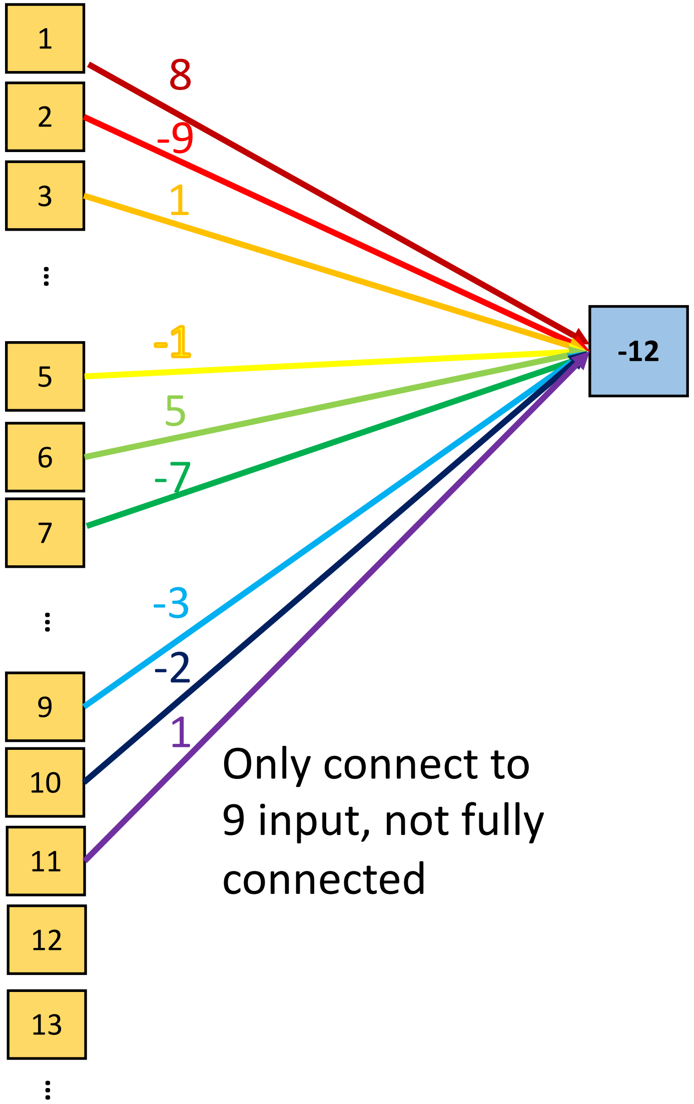
- Basic operation of convolution



What is CNN(Convolutional Neural Network)? - Not fully connected



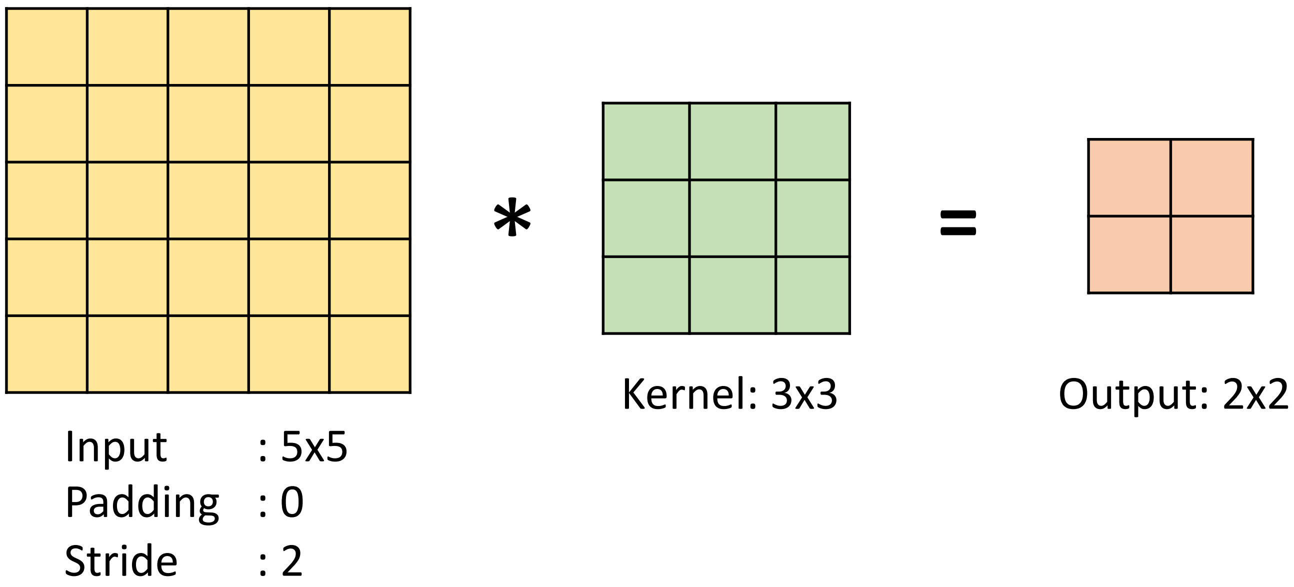
8x1 + (-9)x2 + 1x3 + (-1)x5 + 5x6 + (-7)x7 + (-3)x9 + (-2)x10 +1x11 = -12



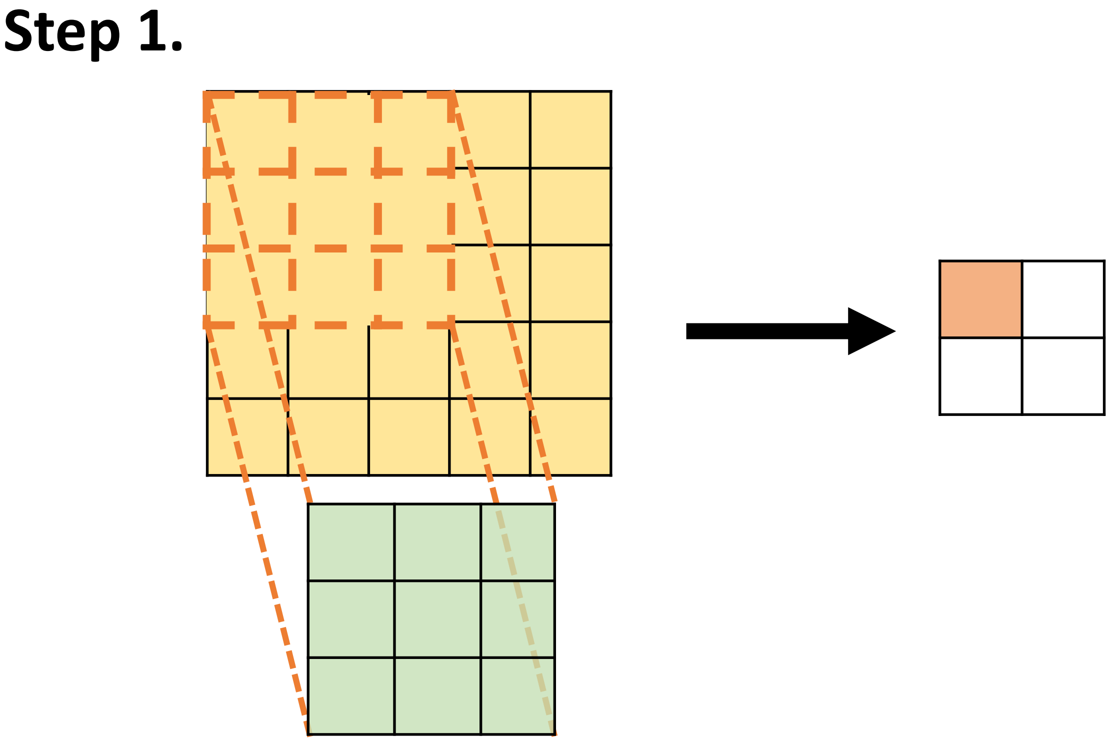
Only connect to 9 input, not fully connected

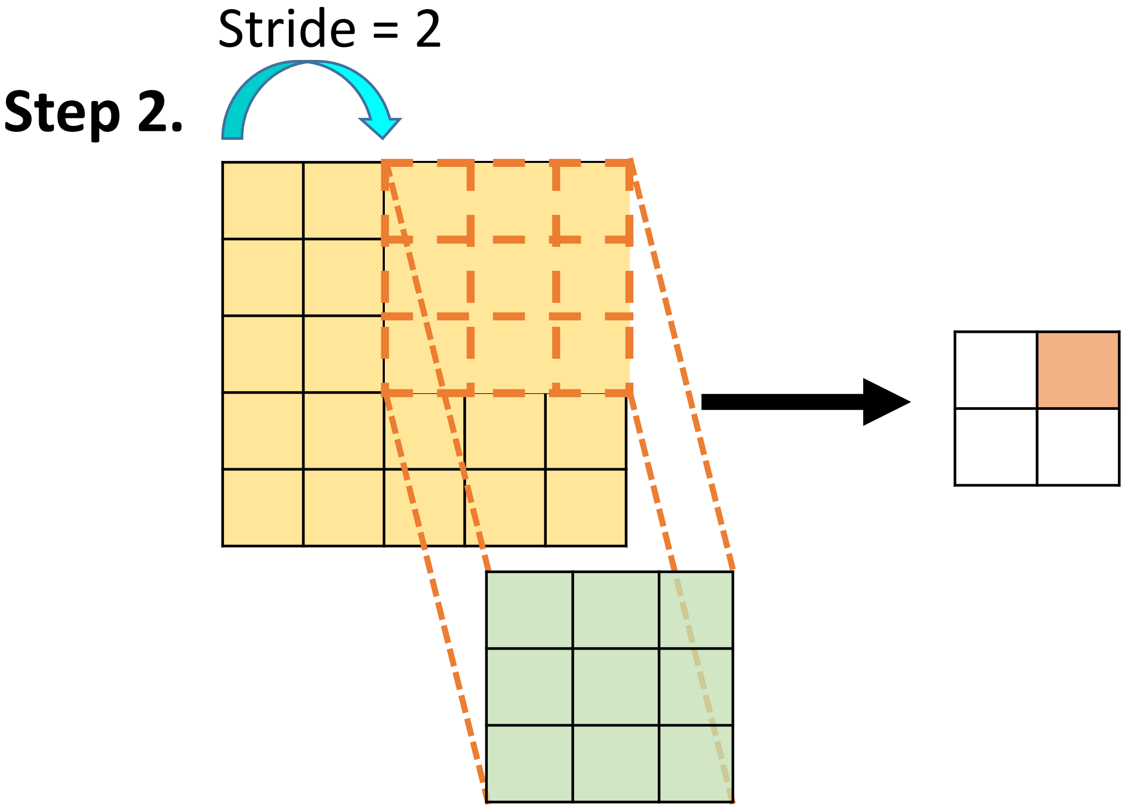
What is CNN(Convolutional Neural Network)? - Stride

Here, set stride to 2. It means kernel slides on the input with step of 2. We will explain how it work in following pages.



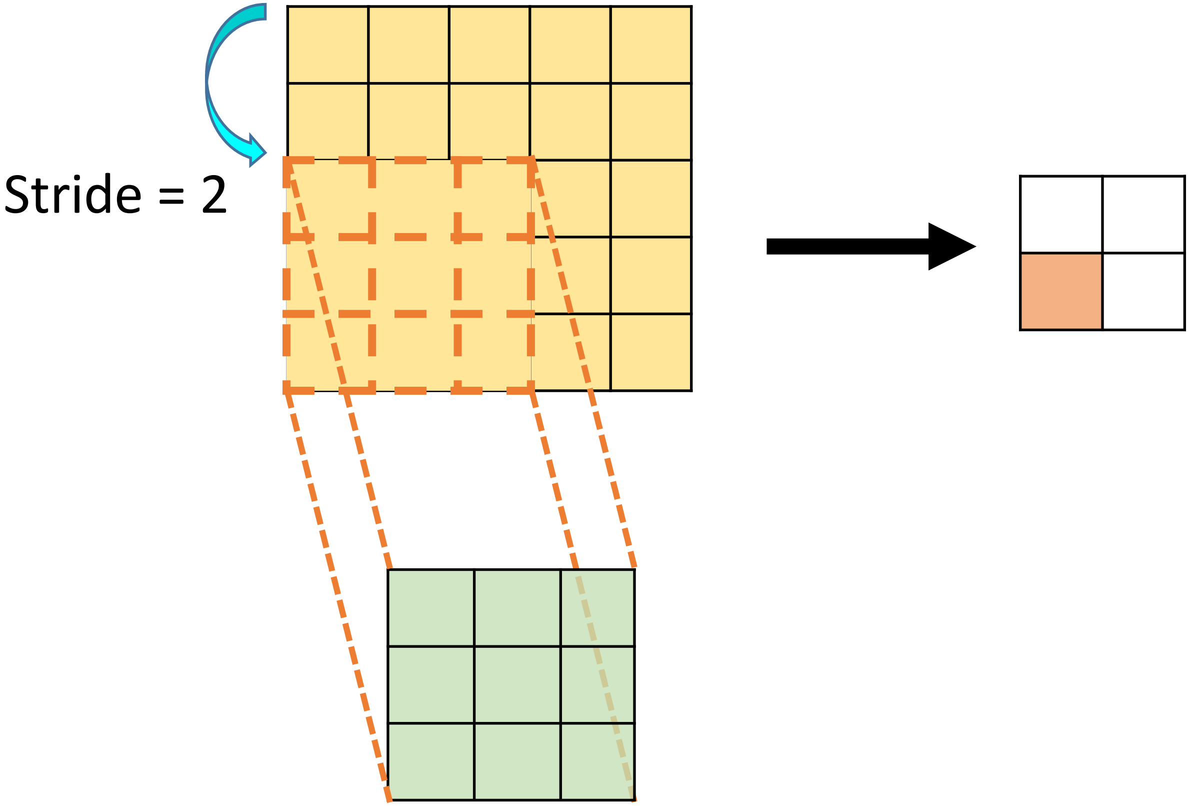
What is CNN(Convolutional Neural Network)? - Stride



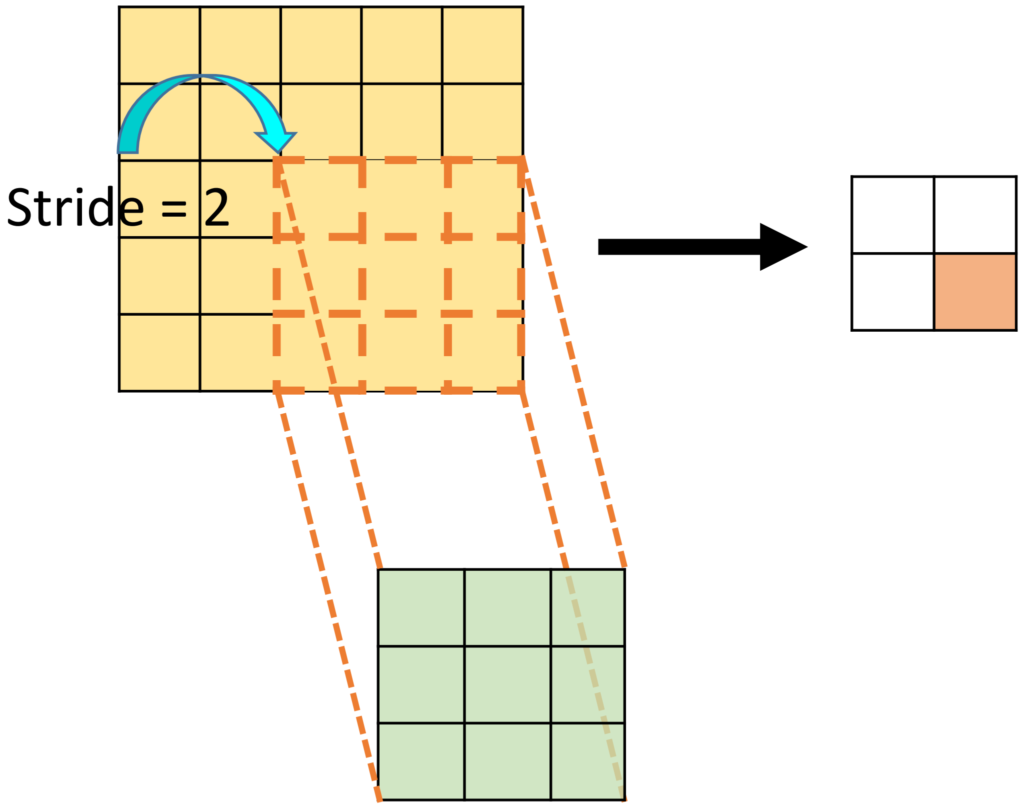


What is CNN(Convolutional Neural Network)? - Stride

Step 3.



Step 4.

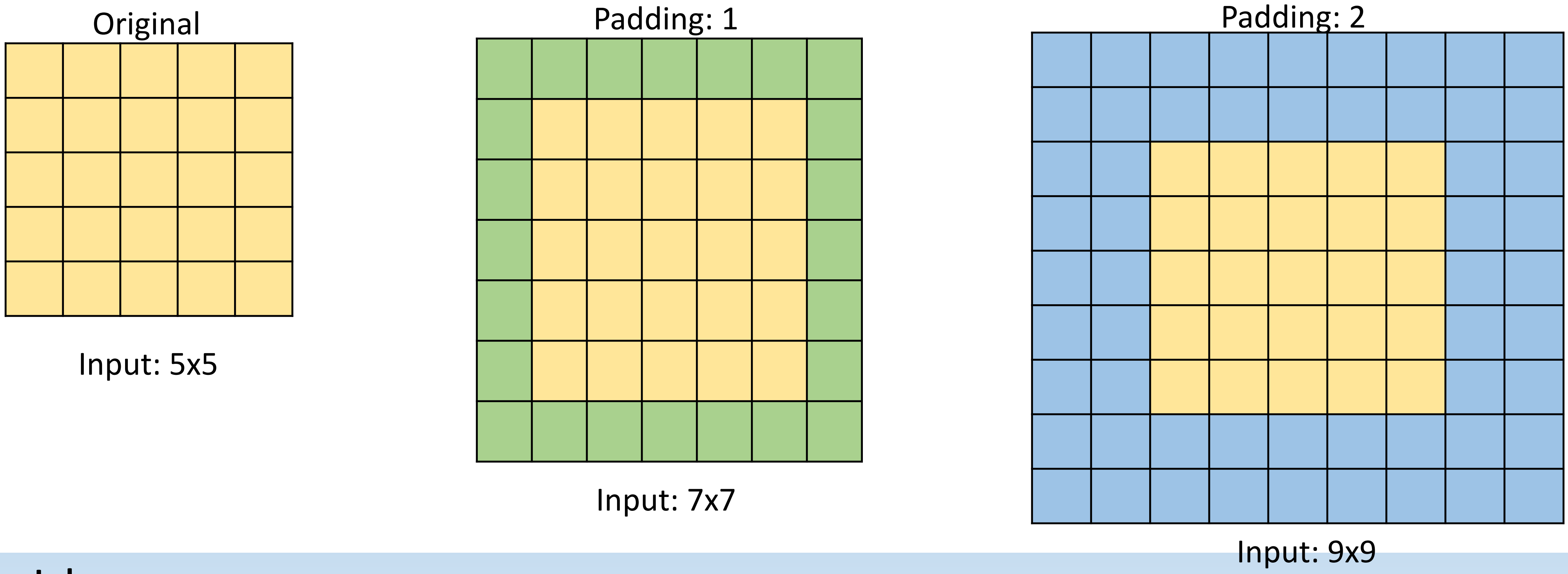


What is CNN(Convolutional Neural Network)? - Padding

Also, we could set padding. It means padding extra pixel surrounding to input.

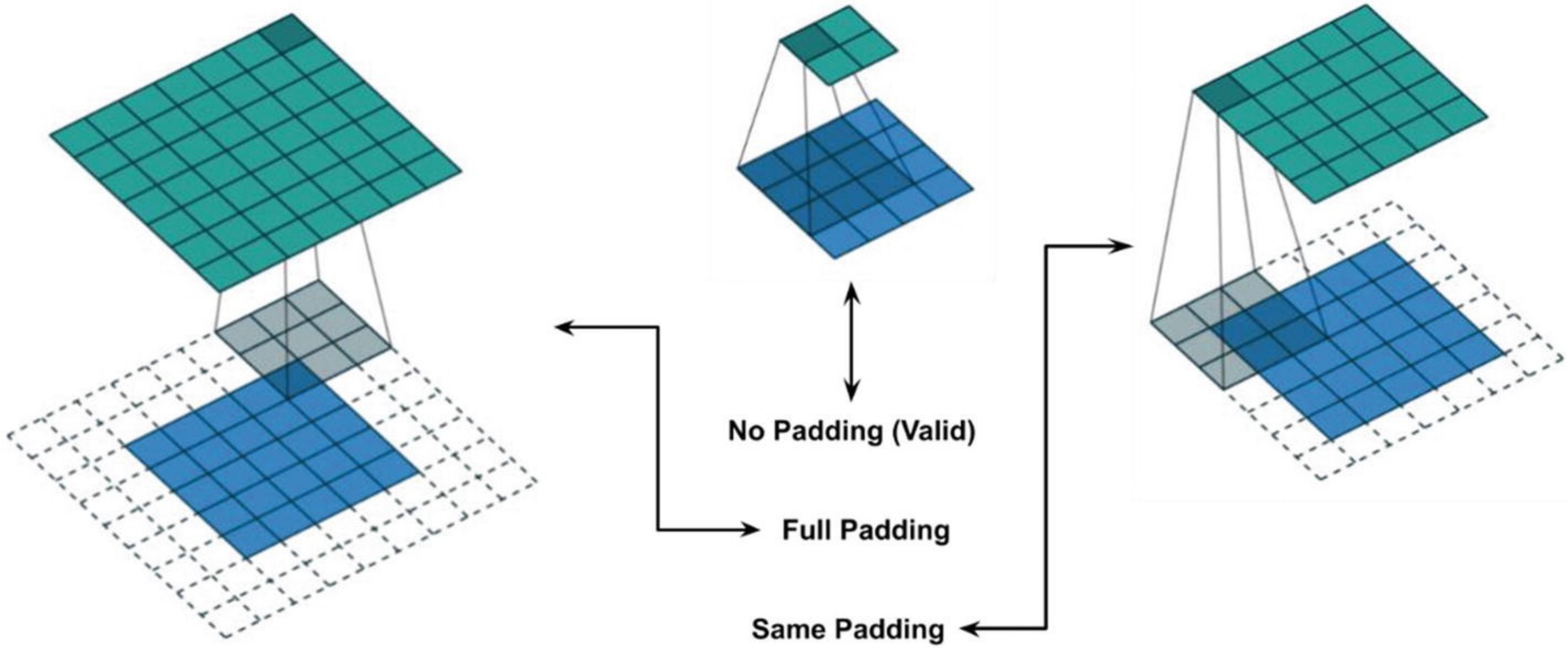
There are two Padding mode {Valid, Same} in Tensorflow, yet you could set custom padding by yourself.

- **Same Padding :** Padding around the input so that the output is the same size as the input (when stride=1)
- **Valid Padding :** No padding !



Padding Example

- **Valid Padding** : No Padding, output size will less than input size
- **Same Padding** : When stride=1, the output is the same size as the input. But when stride > 2, output size is still less than input size
- **Full Padding** : Use to increase output size



Convolution layer – effect of filter

filter

Height = 3
Width = 3
Channel = 1
3*3*1
Count = 5

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Ridge detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Ridge detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen

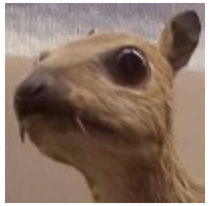
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Box blur

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian blur

Input

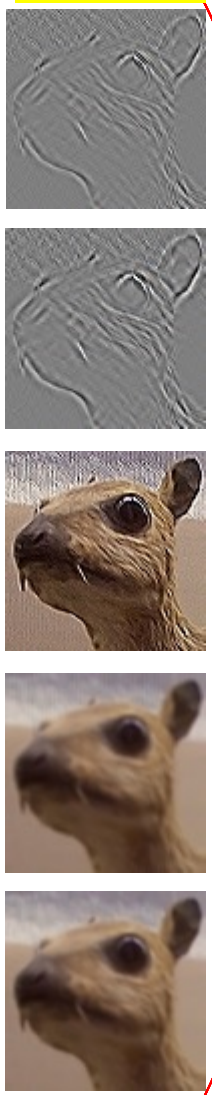


Height = 28
Width = 28
Channel = 1
28*28*1
Count = 1

*

=

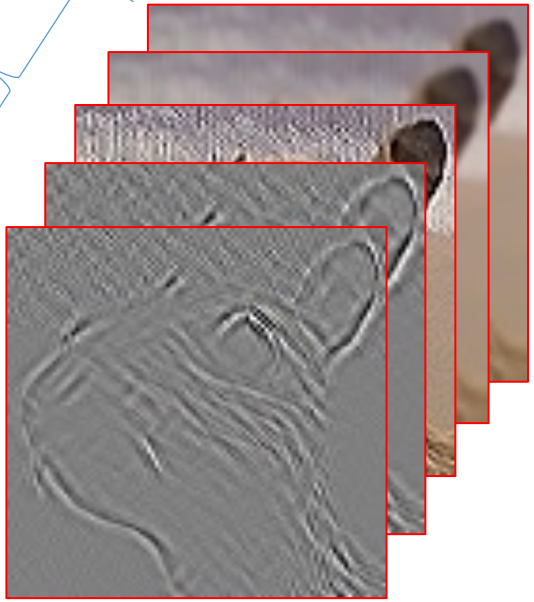
Output



- Use filter to extract object's **feature**.
- We can convolve the input with multiple filters to get <u>a multi-channel</u> output

(Output Channel = Filter Counts)

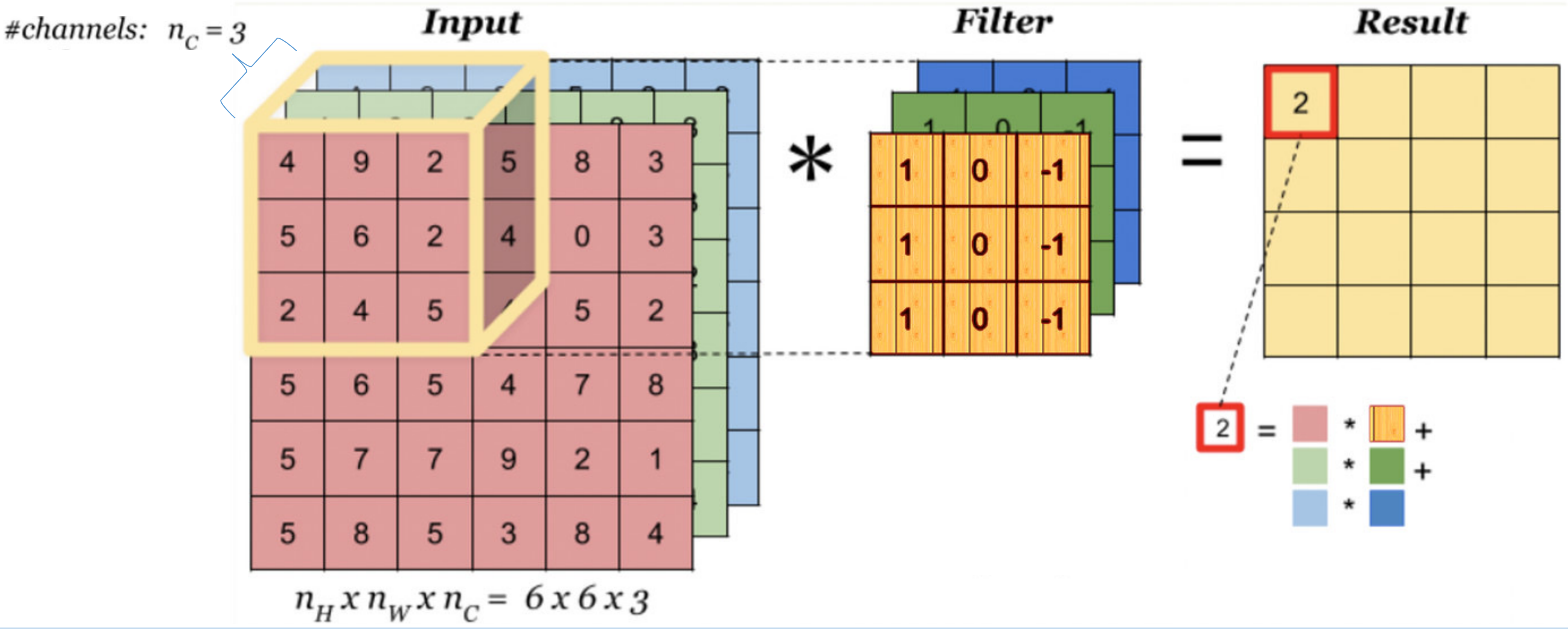
number of
output channel : 5



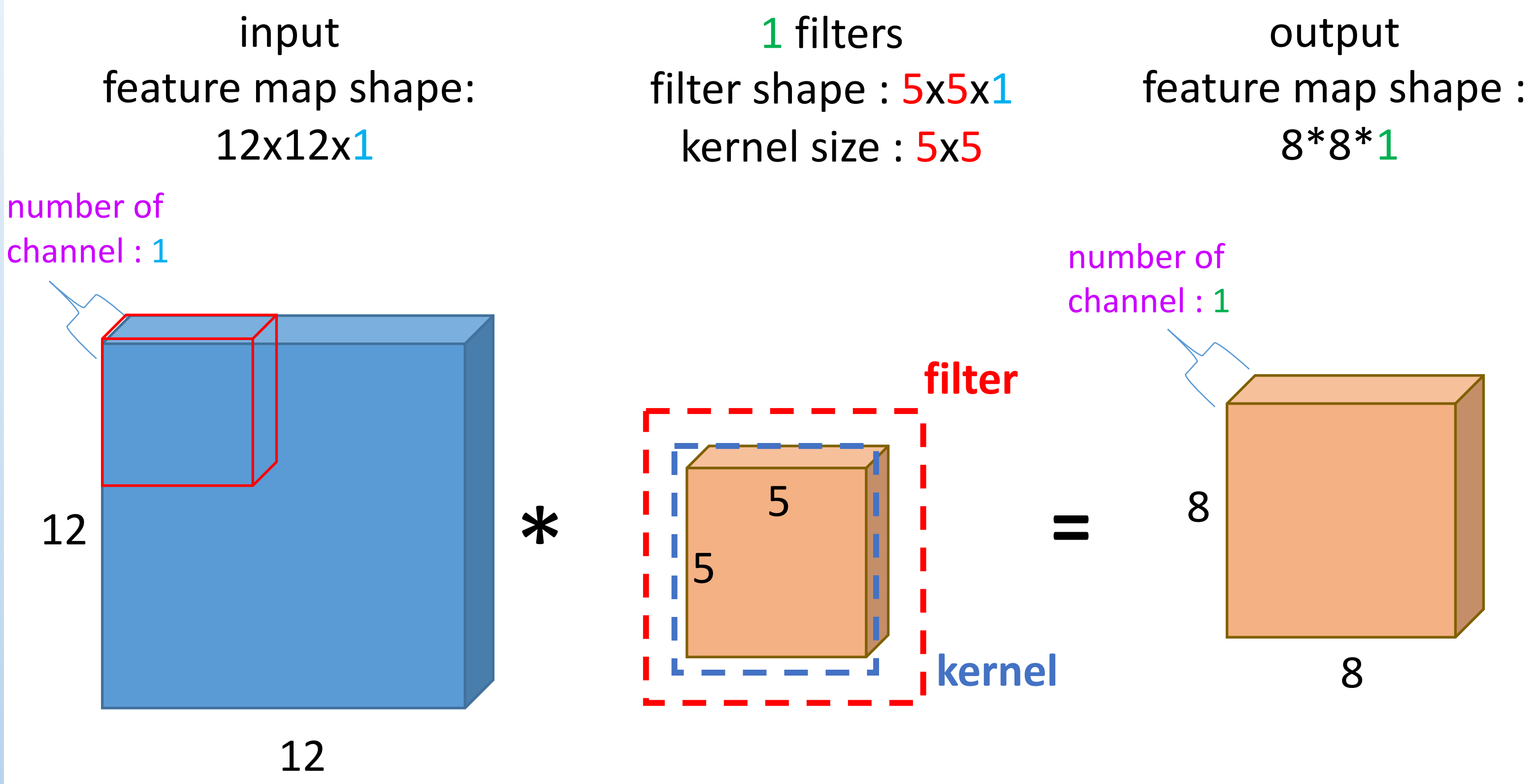
Height = 26
Width = 26
Channel = 5
26*26*5
Count = 1

Convolution layer
– If input has multiple channel

- If input has multiple channel, filter's channel needs to be the same as the input.
- As in the previous ppt, 1 filter generates 1 channel output

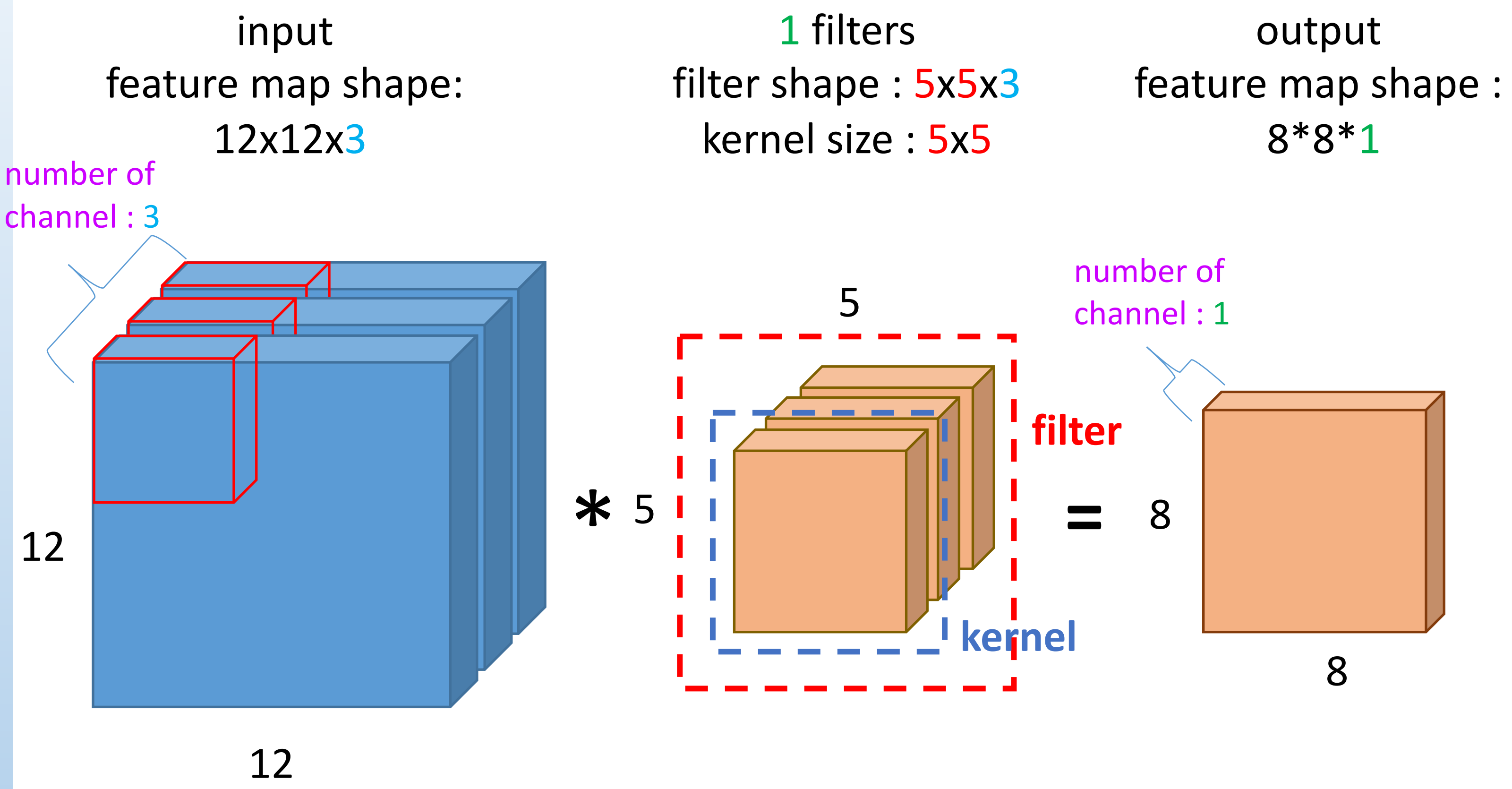


What is CNN(Convolutional Neural Network)?
- Filter, Kernel, Channel



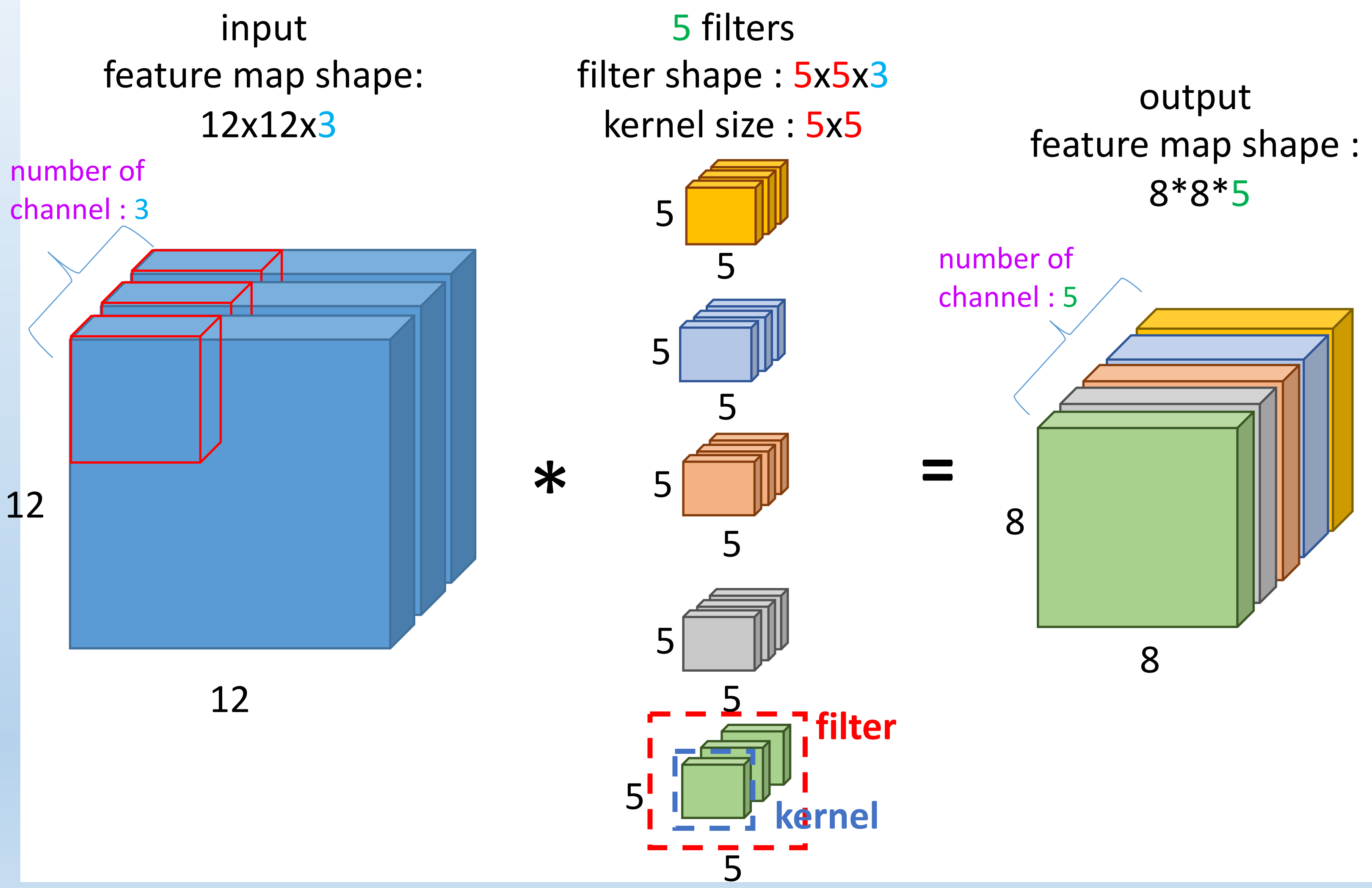
- The example is 1 channel input and 1 filter, so the filter has 1 channel, and output has 1 channel.
- **The number of channel of filter** must be same as **the number of channel of input**
- **The number of channel of output** is same as **the number of filter**

What is CNN(Convolutional Neural Network)? - Filter, Kernel, Channel



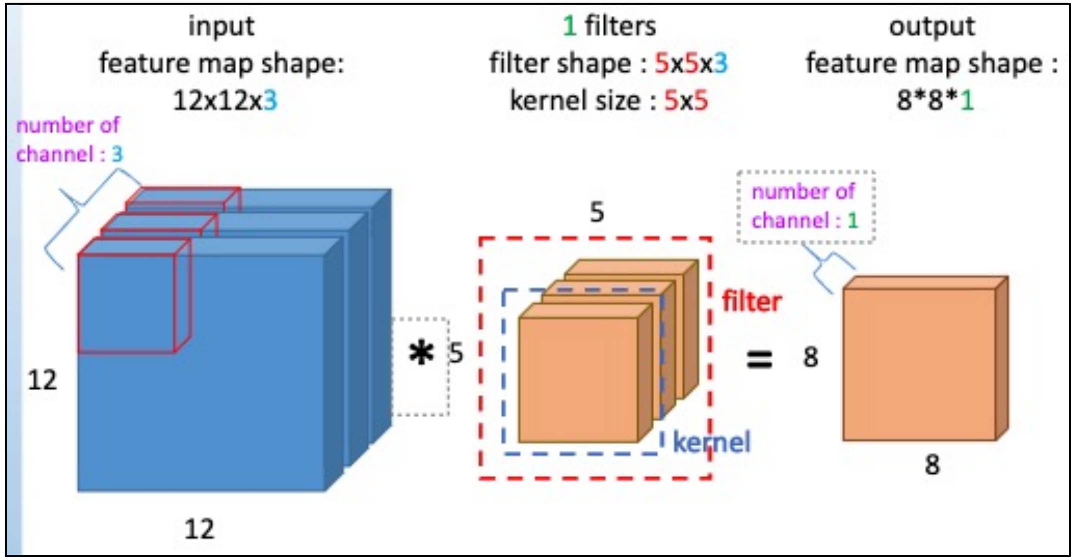
- The example is 3 channel input and 1 filter, so the filter has 3 channel, and output has 1 channel.
- **The number of channel of filter** must be same as **the number of channel of input**
- **The number of channel of output** is same as **the number of filter**

What is CNN(Convolutional Neural Network)? - Filter, Kernel, Channel



- The example is 3 channel input and 5 filter, so each filter has 3 channel, and output has 5 channel.
- **The number of channel of filter** must be same as **the number of channel of input**
- **The number of channel of output** is same as **the number of filter**

What is CNN(Convolutional Neural Network)? – - Output feature map size



FORMULA:

$$\text{Height}_{out} = \text{floor}\left(\frac{\text{Height}_{in} + 2 \text{ x padding} - \text{kernel\_size}}{\text{stride}} + 1\right)$$

$$\text{Width}_{out} = \text{floor}\left(\frac{\text{Width}_{in} + 2 \text{ x padding} - \text{kernel\_size}}{\text{stride}} + 1\right)$$

In this example, we have
Height_{in}=12
Width_{out} =12
padding = 0
kernel_size = 5
stride = 1

Applying formula above, we get
Height_{out}= 8 = floor($\frac{12+2 \text{ x } 0 - 5}{1}$ + 1)
Width_{out}= 8 = floor($\frac{12+2 \text{ x } 0 - 5}{1}$ + 1)

Much more detail [Conv2d — PyTorch 1.10 documentation]

Pooling layer

- Remain feature information and reduce parameters

Max Pooling

Feature map

0	**3**	0	0
0	**1**	1	1
1	0	1	2
1	4	2	1

Pooled Feature map

3	1
4	2

Average Pooling

Feature map

0	**3**	0	0
0	**1**	1	0
1	4	2	0
0	0	1	1

(0 + 3 + 0 + 1) /4 = 1

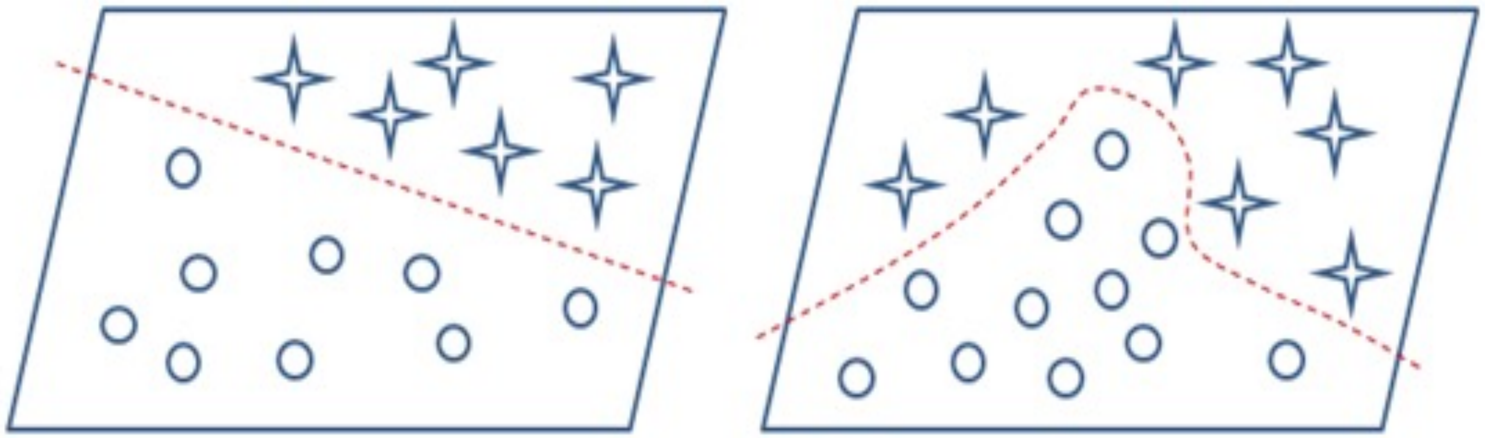
Pooled Feature map

1	0.25
1.5	1

Effect of pooling : https://youtu.be/fApFKmXcp2Y

Activation function

- Main purpose of activation function is offering non-linearity
- Make network possible to capture complex pattern and get SOTA(state-of-the-art) result.

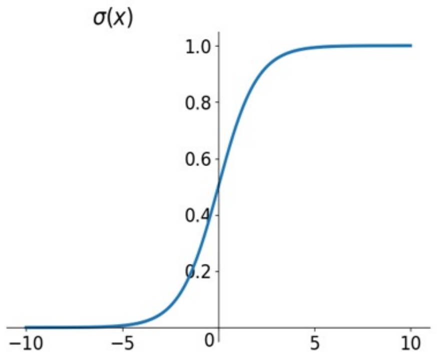


With linearity

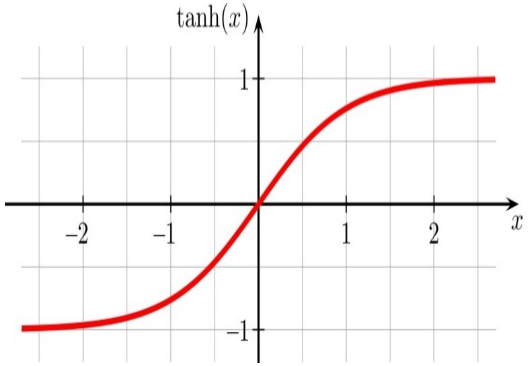
Non-linearity

Activation function

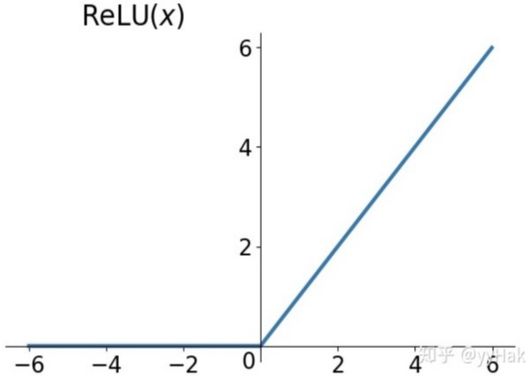
Sigmoid, $\sigma(x) = \frac{1}{1+e^{-x}}$



$\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$



$\text{ReLU}(x) = max(0, x)$



Softmax $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ for $j = 1, \ldots, K$.

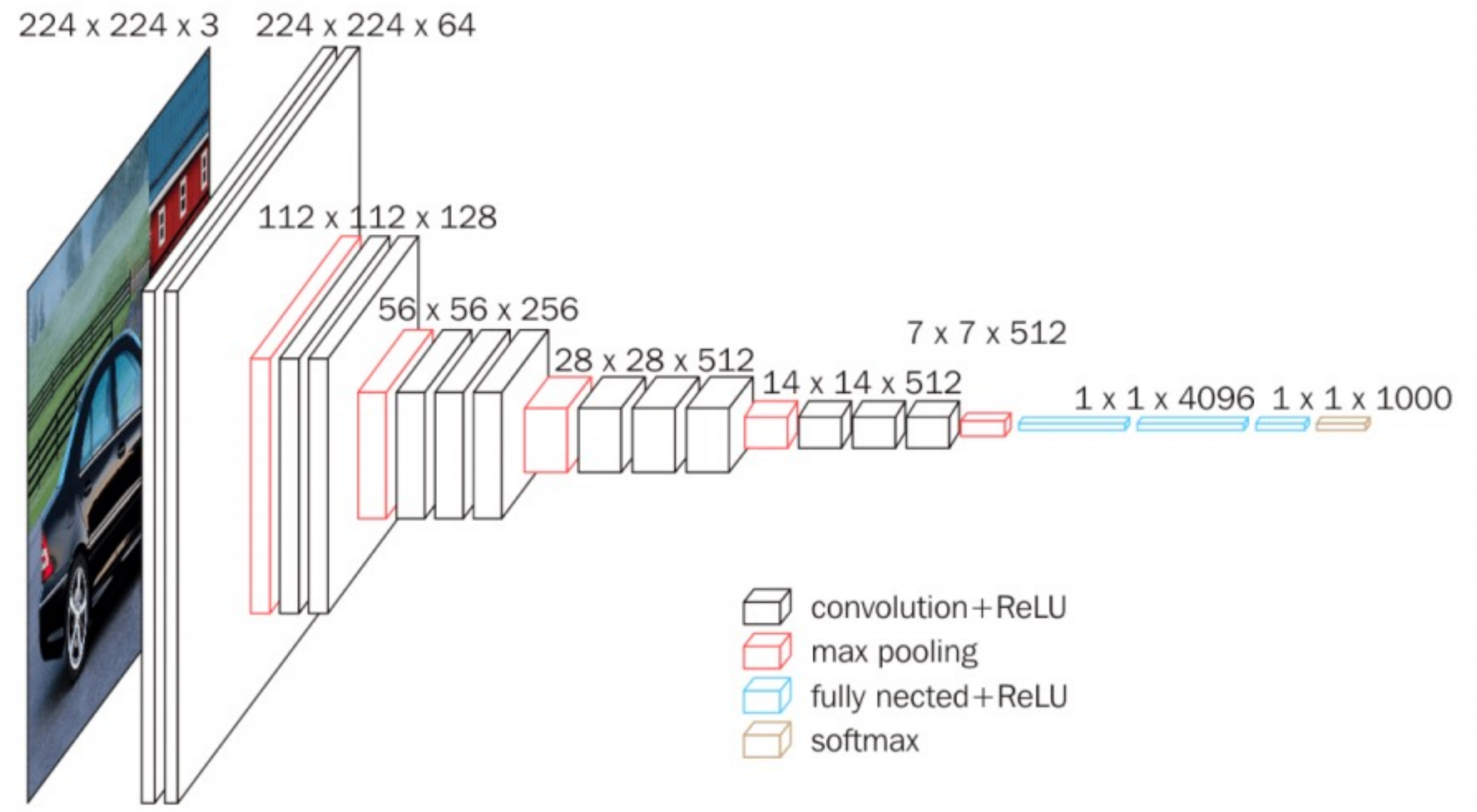
Common CNN network – LeNet5

- LeNet5

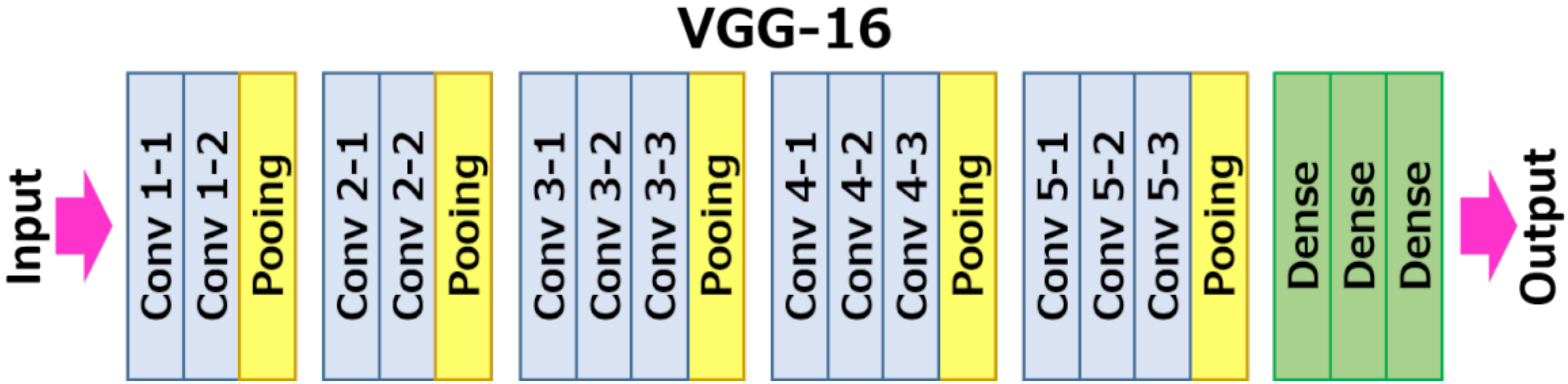
Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Common CNN network – VGG16

- VGG16



Common CNN network – VGG16



- Dense layer means fully connected layer

Common CNN network

- AlexNet
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenetclassification with deep convolutional neural networks. Advances in neural information processing systems, 25.
- ResNet
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition(pp. 770-778).
- DenseNet•Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition(pp. 4700-4708).
- MobileNet
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXivpreprint arXiv:1704.04861.
- ShuffleNet
- Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition(pp. 6848-6856).

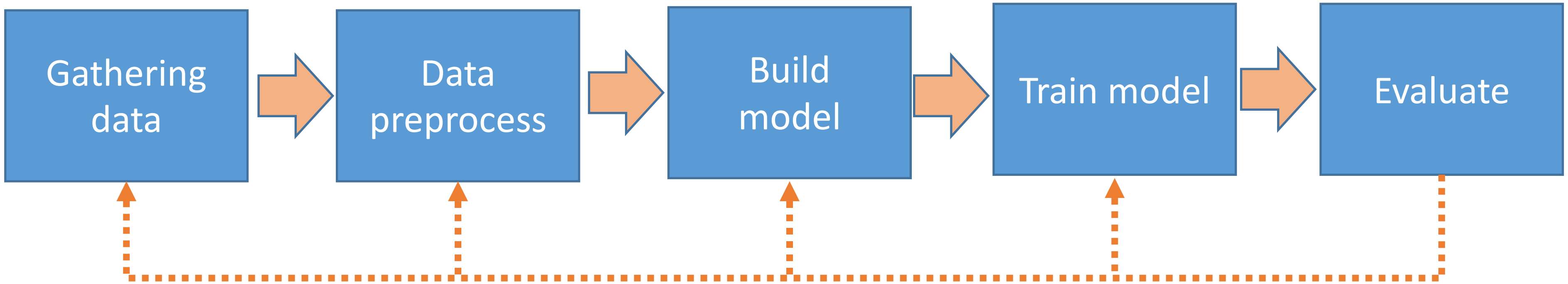
From lecture pdf

Import predefined model in Keras

- You could also import predefined model from Keras.
  - VGG16
  - ResNet50
  - MobileNet
  - EfficientNet
  - DenseNet121
  - …..

Keras predefined model

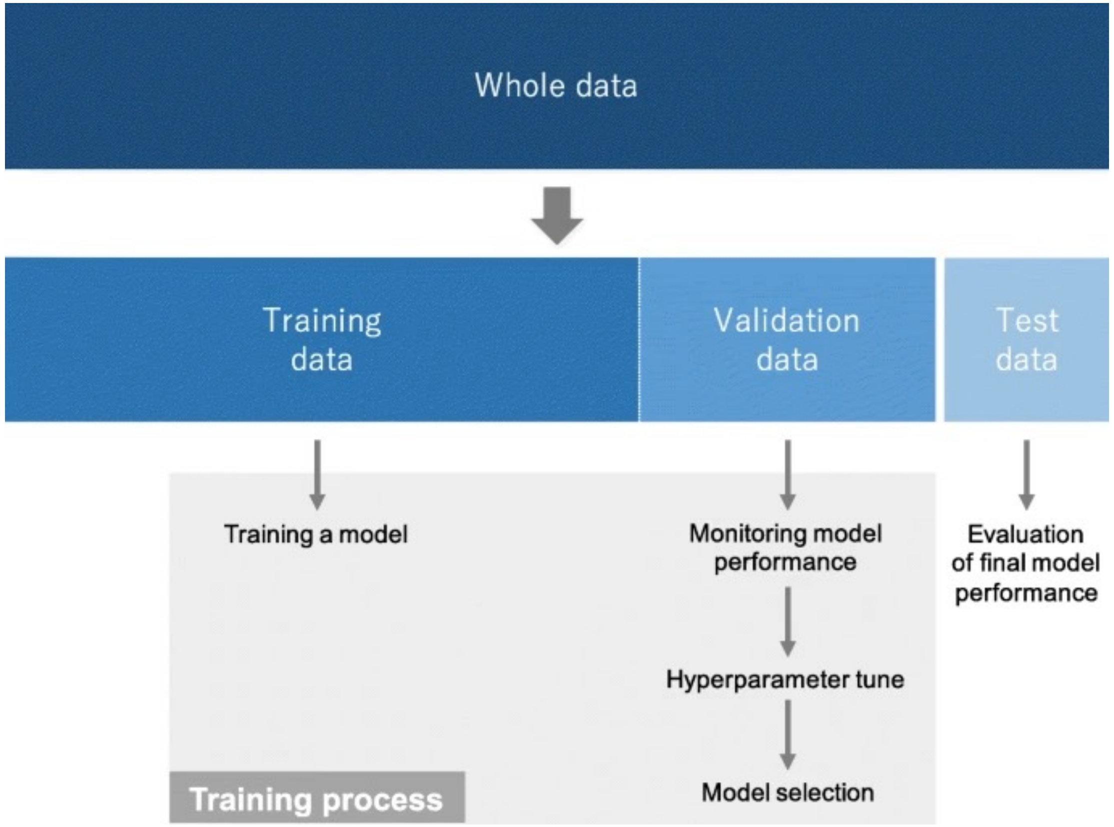
Classic flow for training a model



- **Gathering data** -> From network or gather by yourself
- **Data preprocess** -> Raw data will probably lead to bad classification performances
- **Build model** -> Design a model for predicting data
- **Train model** -> Learn good values for all the parameters from labeled training data
- **Evaluate** -> Evaluate model could give a suitable response from its experience

Dataset

- training set
  - A dataset of examples is used during the learning process and is used to fit the parameters(e.g., weights).
- validation set
  - A validation data set is a dataset of examples used to tune the hyperparameter (i.e. the architecture) of a model.
  - Validation set is not necessary.
- testing set
  - A testing set is used to evaluate the final ability of the model.
  - It should not be used as a basis for parameter adjustment, selection of features.



Assignment Overview

- TAs build the model like LeNet-5 as assignment examples.
- **Link** : Assignment examples in Colab

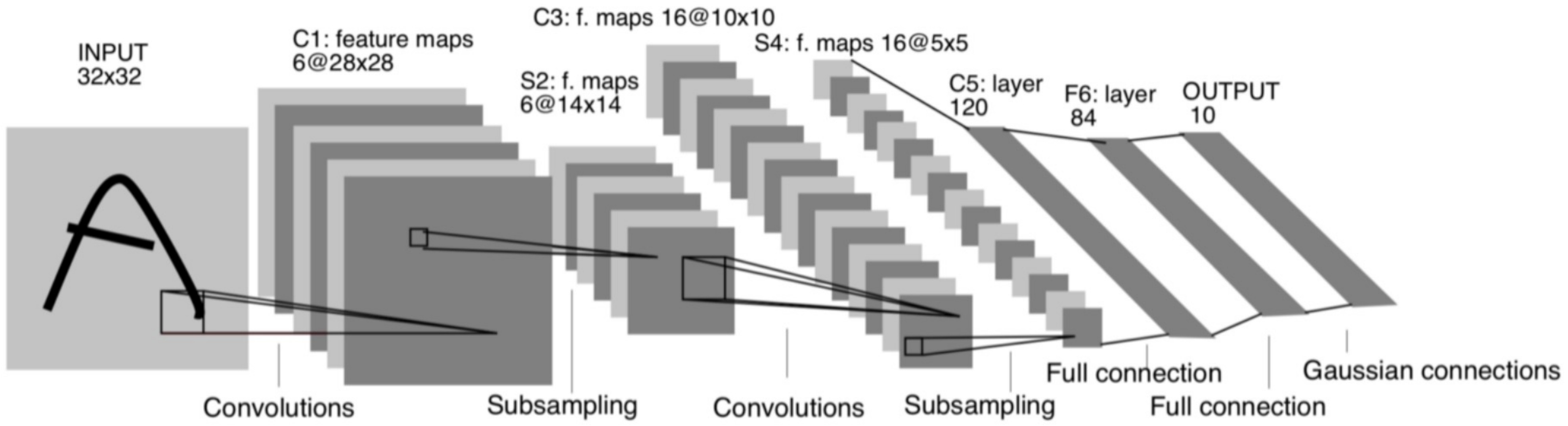
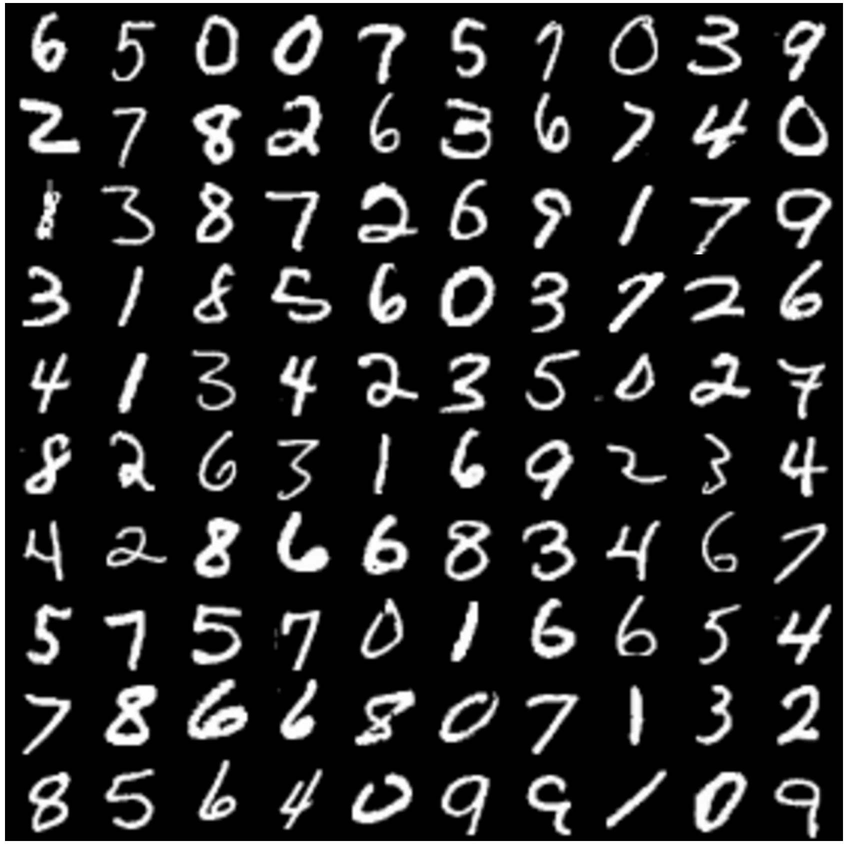


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

- LeNet-5 Architecture

Dataset : MNIST



- Database of handwritten digits

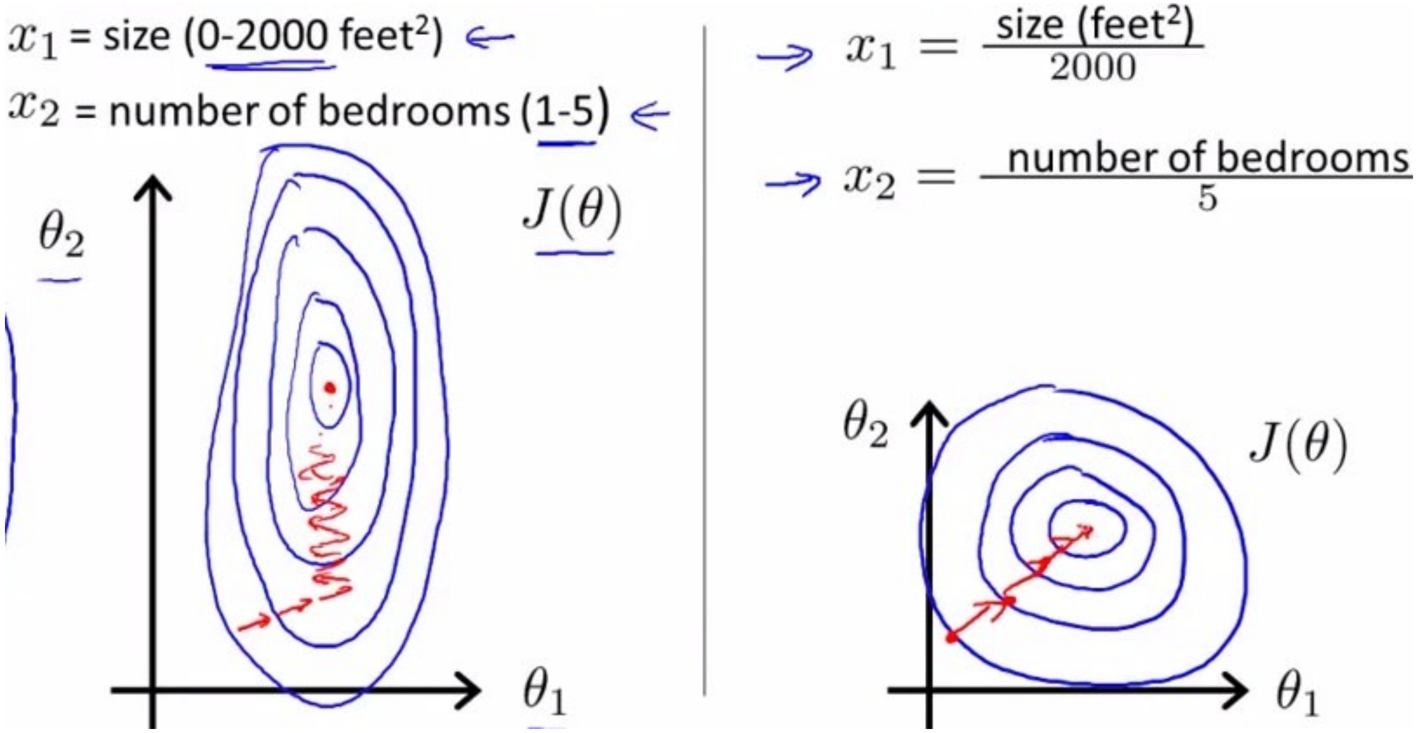
- 60,000 training data
- 10,000 testing data
- 10 categories(0~9).
- Each gray-scale image is 28x28.

```
# Load MNIST dataset
# mnist.load_data() method returns tuple of NumPy arrays.
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

tf.keras.datasets.mnist.load_data | TensorFlow Core v2.8.0

Data Preprocess : Normalization

- Purpose: Optimize gradient descent and get higher accuracy
- Normalization usually rescales features to [0, 1]. That is, $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$




```
6 #  Normalization
7 x_train  =  x_train/255
8 x_test  =  x_test/255
```

Data Preprocess : One-Hot Encoding

original label target → one-hot encoding label target

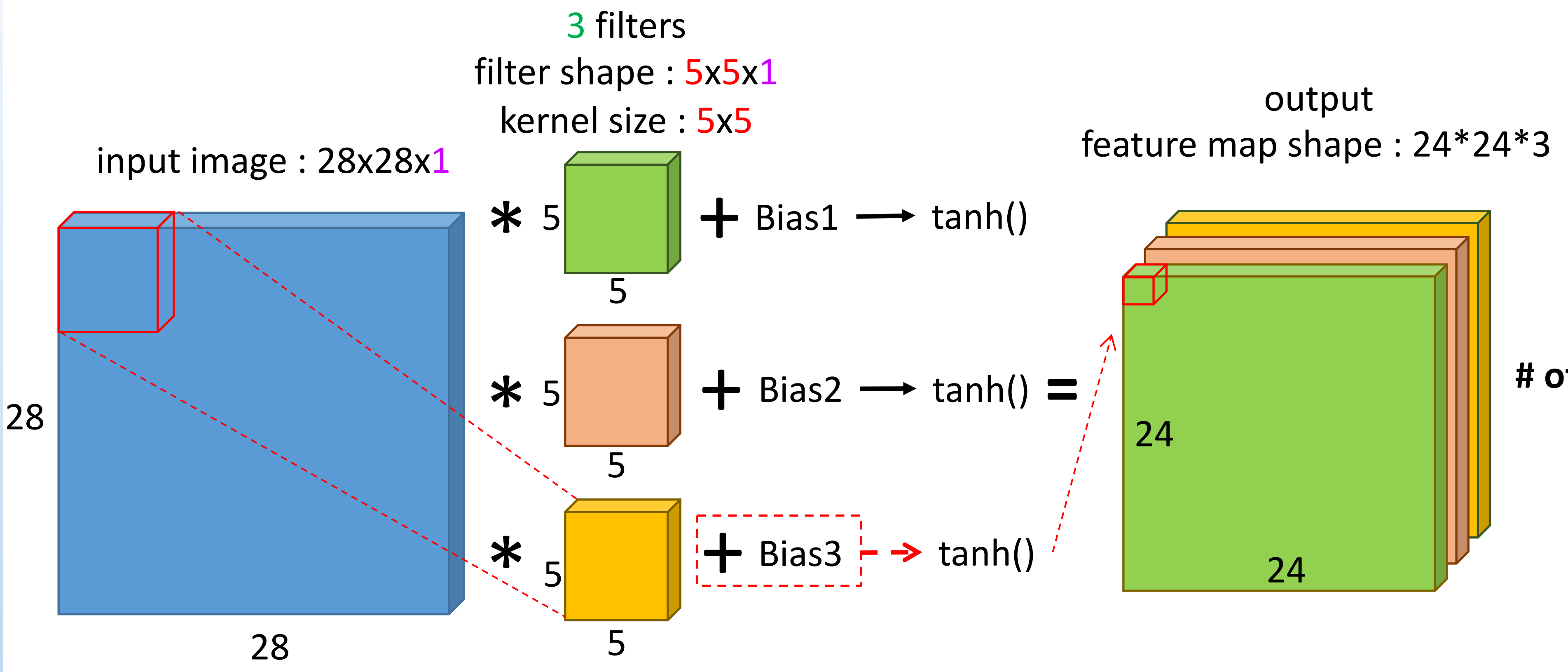
label		label0	label1	label2	label3	label4	label5	label6	label7	label8	label9
0	→	1	0	0	0	0	0	0	0	0	0
1	→	0	1	0	0	0	0	0	0	0	0
2	→	0	0	1	0	0	0	0	0	0	0
3	→	0	0	0	1	0	0	0	0	0	0
4	→	0	0	0	0	1	0	0	0	0	0
5	→	0	0	0	0	0	1	0	0	0	0
6	→	0	0	0	0	0	0	1	0	0	0
7	→	0	0	0	0	0	0	0	1	0	0
8	→	0	0	0	0	0	0	0	0	1	0
9	→	0	0	0	0	0	0	0	0	0	1

- Every image in MNIST has 1 label which is 0 ~ 9
- This is a classification problem.
  We have to transform 1 label into 10 labels.
- A single high (1) bit and all the others low (0).

```
# One-hot encode the labels
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
```

Classification as regression?
[ML Lecture 4: Classification (06:30)]

Layer 1 : Conv2D



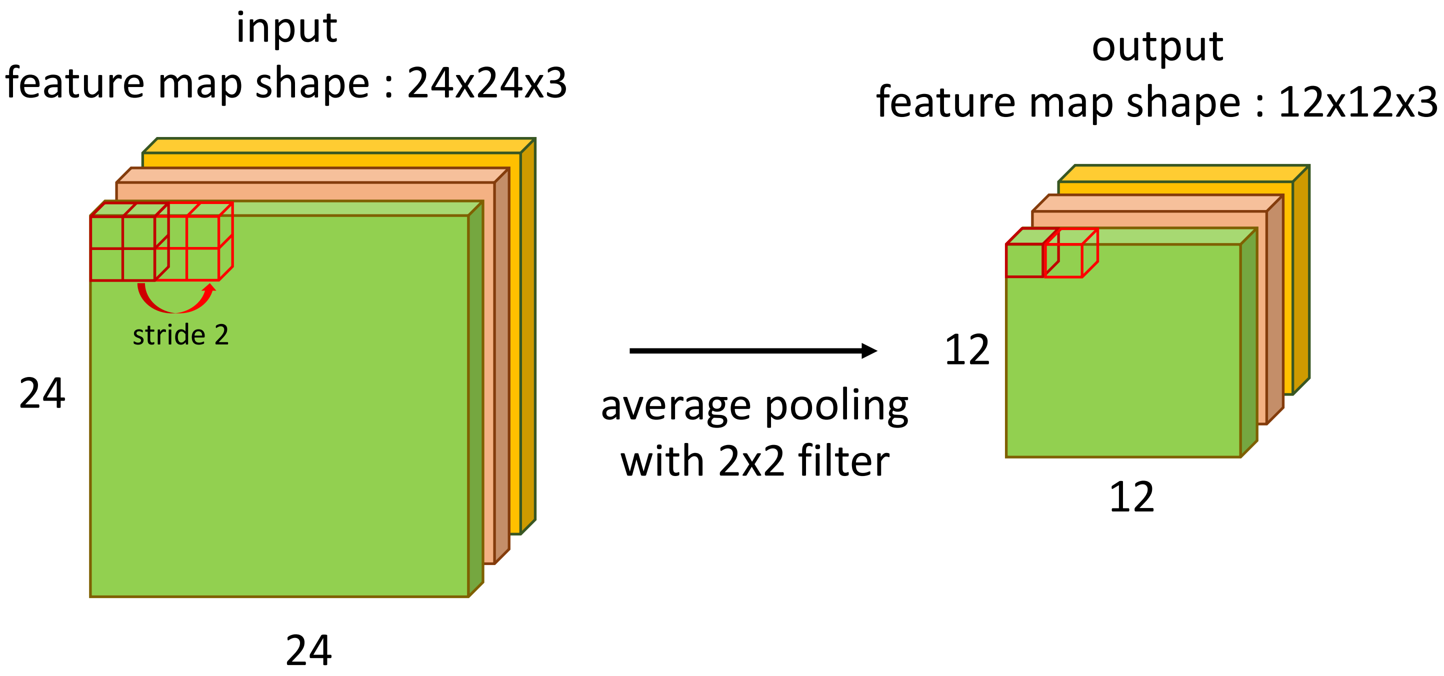
# of parameter : 3x5x5x1 + 3 = 78
# of bias
(=# of filter)

```
model.add(Conv2D(input_shape=(28,28,1), filters=3, kernel_size=(5, 5), activation='tanh'))
```

- When using this layer as the first layer in a model, provide the keyword argument **input_shape.**

[tf.keras.layers.Conv2D | TensorFlow Core v2.8.0]

Layer 2 : Average Pooling




```
model.add(AveragePooling2D(2,2))
```

[tf.keras.layers.AveragePooling2D | TensorFlow Core v2.8.0]

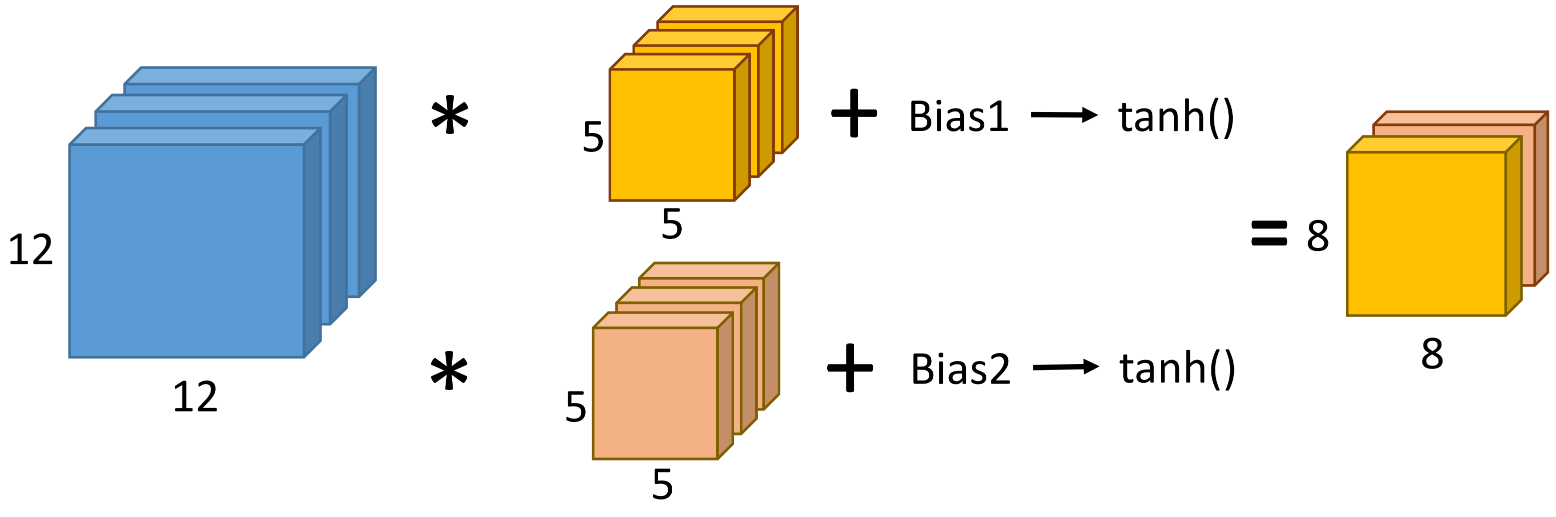
[tf.keras.layers.MaxPool2D | TensorFlow Core v2.8.0]

Layer 3 : Conv2D

input
feature map shape:
12x12x3

2 filters
filter shape : 5x5x3
kernel size : 5x5

output
feature map shape :
8*8*2



# of parameter : 2x5x5x3 + 2 = 152

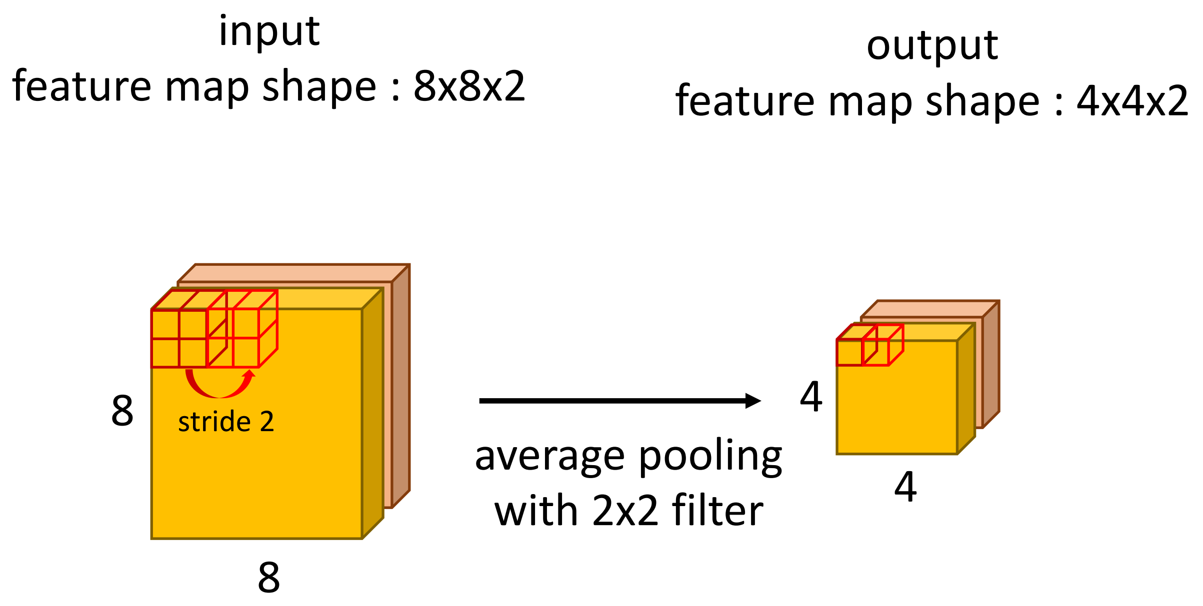
# of bias

(=# of filter)

```
model.add(Conv2D(filters=2, kernel_size=(5, 5), activation='tanh'))
```

tf.keras.layers.Conv2D | TensorFlow Core v2.8.0

Layer 4 : Average Pooling

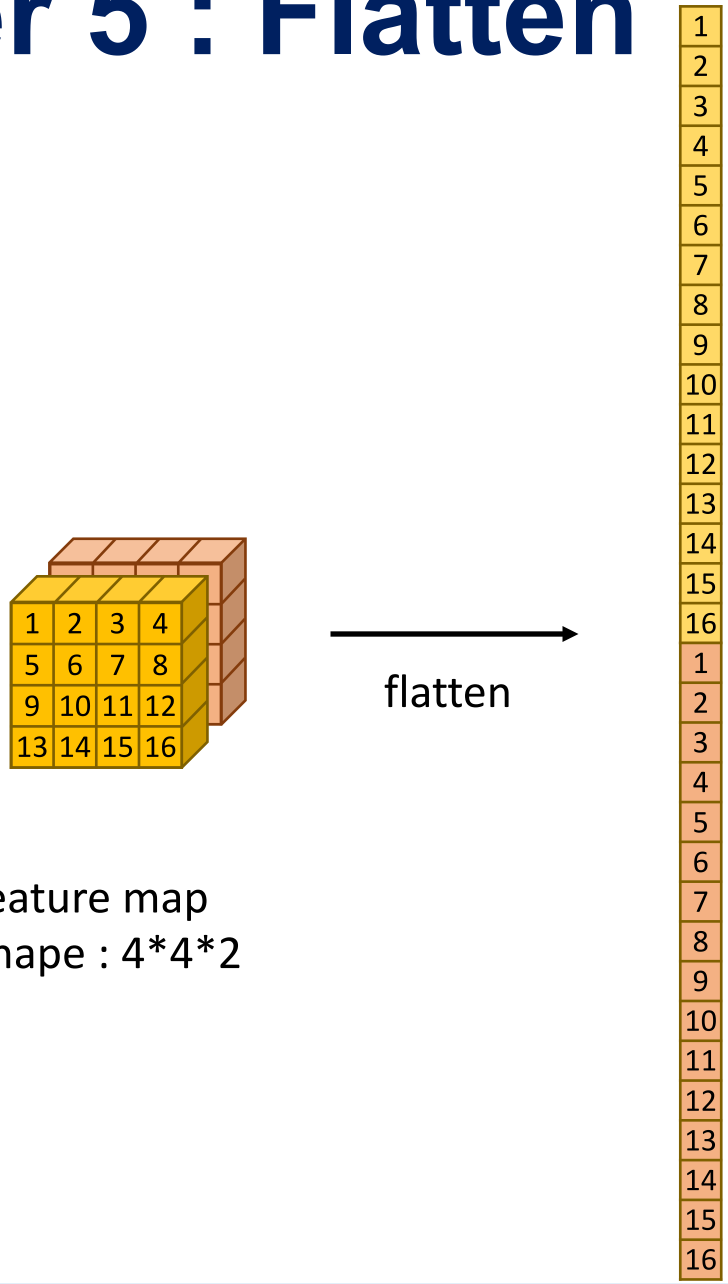



```
model.add(AveragePooling2D(2,2))
```

[tf.keras.layers.AveragePooling2D | TensorFlow Core v2.8.0]

[tf.keras.layers.MaxPool2D | TensorFlow Core v2.8.0]

Layer 5 : Flatten



feature map
shape : 4*4*2

feature map
shape : 32

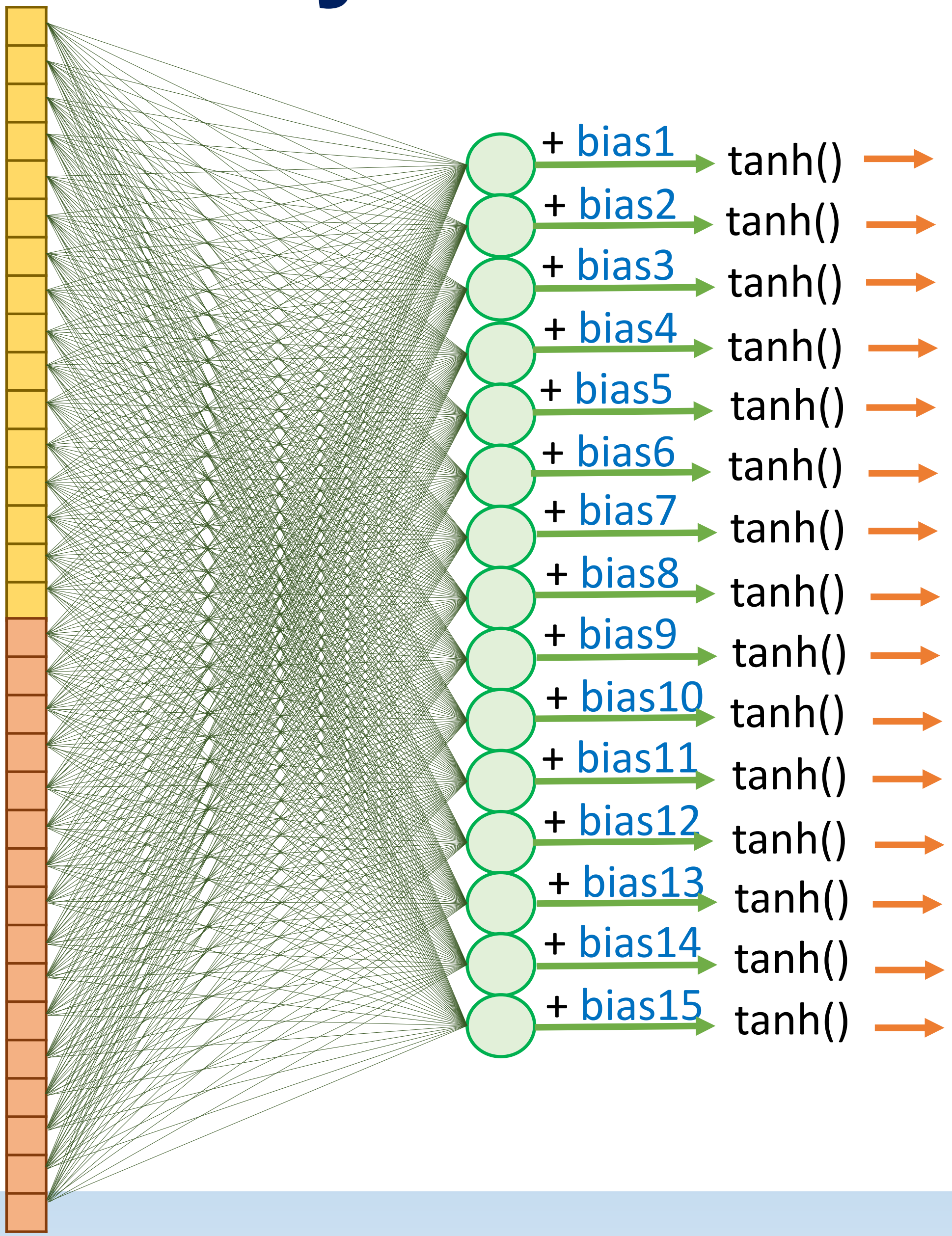
- Flatten layer transforms multi-dimensional into one-dimension
- Commonly used in the transition from convolution layer to fully connected layer

```
model.add(Flatten())
```

[tf.keras.layers.Flatten | TensorFlow Core v2.8.0]

Layer 6 : Fully Connected Layer

input
feature map shape: 32



output
feature map shape: 15

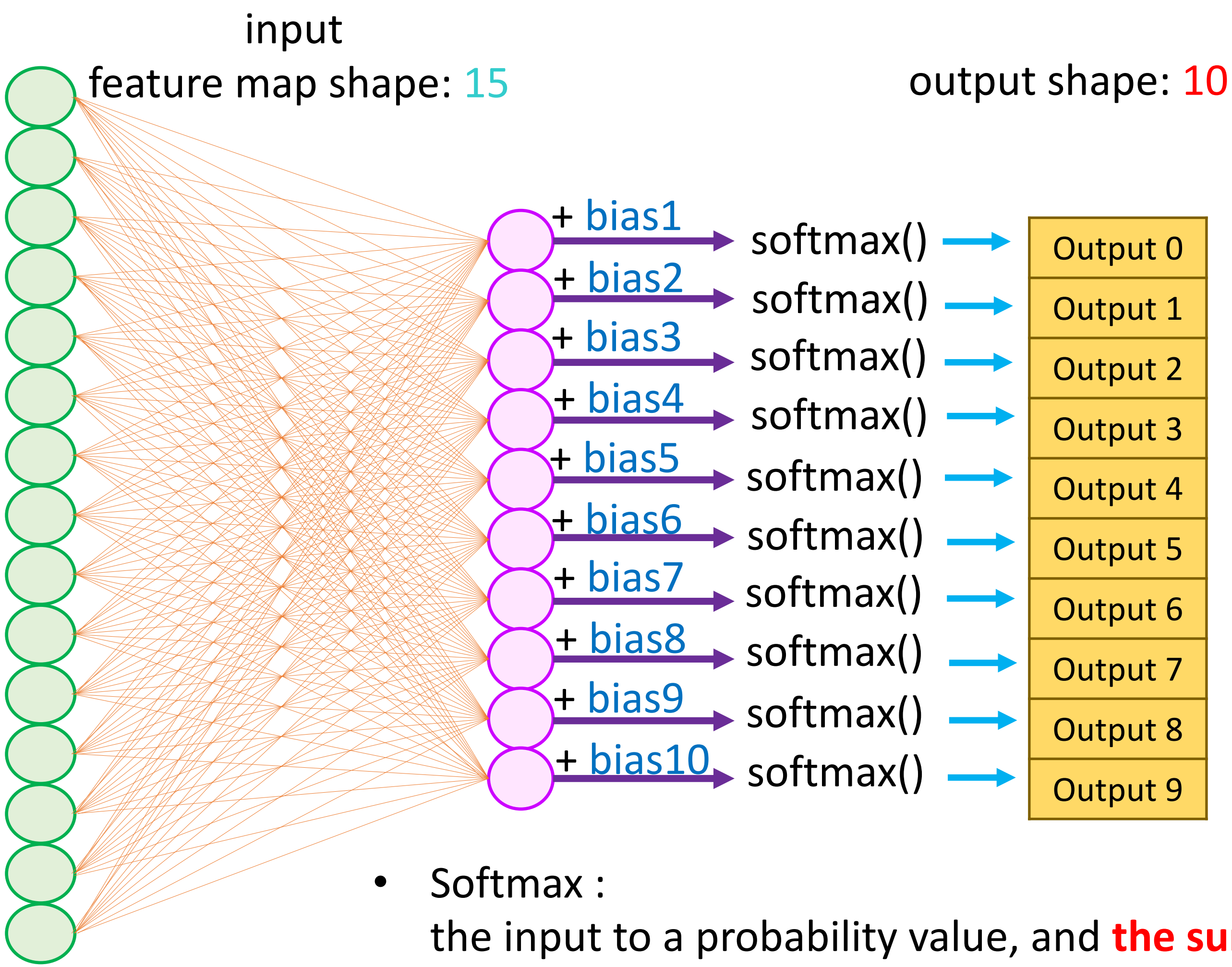
# of parameter : 32 x 15 + 15 = 495
# of bias

```
model.add(Dense(units=15,  activation='tanh'))
```

- dense layer means fully connected layer

[tf.keras.layers.Dense | TensorFlow Core v2.8.0]

Layer 7 : Fully Connected Layer



# of parameter : 15 x 10 + 10 = 160

# of bias

```
model.add(Dense(units=10,  activation="softmax"))
```

- Softmax : Convert the input to a probability value, and **the sum of the probability of all output classes is equal to 1**
- **Output 0 + Output 1 + … Output 9 = 1**
- **Each output means the probability (confidence score) of the corresponding class**

tf.keras.layers.Dense | TensorFlow Core v2.8.0

Total Parameters

```
# Print the model summary
model1.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 24, 24, 3)         78

 average_pooling2d (AverageP  (None, 12, 12, 3)        0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 8, 8, 2)           152

 average_pooling2d_1 (Averag  (None, 4, 4, 2)          0
 ePooling2D)

 flatten (Flatten)           (None, 32)                0

 dense (Dense)               (None, 15)                495

 dense_1 (Dense)             (None, 10)                160

=================================================================
Total params: 885
Trainable params: 885
Non-trainable params: 0
```

number of filter

3x5x5x1 + 3

filter size number of bias (number of filter)

2x5x5x3 + 2

number of bias

32x15 + 15

15x10 + 10

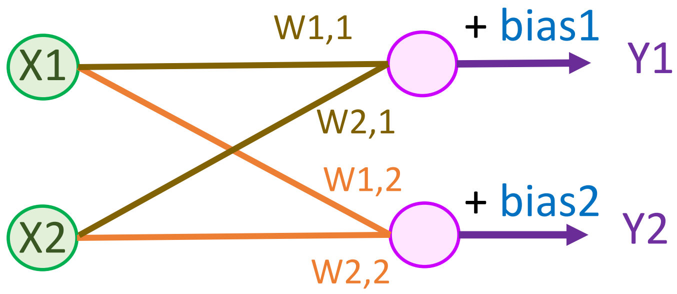
```
model = tf.keras.models.Sequential()

model.add(Conv2D(input_shape=(28,28,1), filters=3,
                 kernel_size=(5, 5), activation='tanh'))
model.add(AveragePooling2D(2,2))
model.add(Conv2D(filters=2,
                 kernel_size=(5, 5), activation='tanh'))
model.add(AveragePooling2D(2,2))
model.add(Flatten())
model.add(Dense(units=15, activation='tanh'))
model.add(Dense(units=10, activation="softmax"))
```

total parameters :
78 + 152 + 495 + 160 = 885

FLOPs (floating point operations)

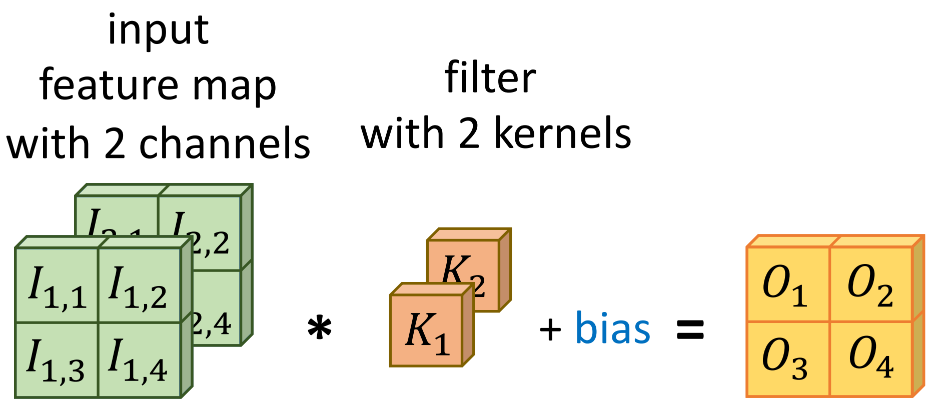
- **Example – Fully connected layer**



Y1 = + X1*W1,1 + X2*W2,1 + bias1
Y2 = + X1*W1,2 + X2*W2,2 + bias2

# of multiplication operations : 4
# of addition operations : 6
FLOPs = 4 + 6 = 10

- **Example – Convolution layer**



O_1 = + $I_{1,1}$*K_1 + $I_{2,1}$*K_2 + bias
O_2 = + $I_{1,2}$*K_1 + $I_{2,2}$*K_2 + bias
O_3 = + $I_{1,3}$*K_1 + $I_{2,3}$*K_2 + bias
O_4 = + $I_{1,4}$*K_1 + $I_{2,4}$*K_2 + bias

# of multiplication operations : 8
# of addition operations : 12
FLOPs = 8 + 12 = 20

FLOPs (floating point operations)

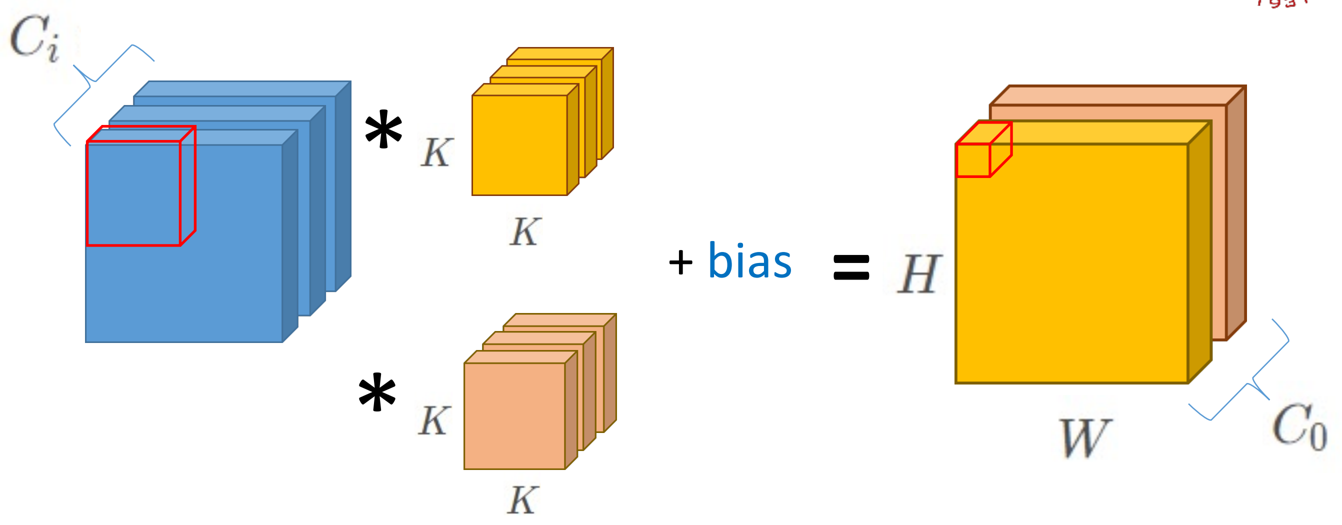
- **Convolution layer**

$$\left[\underbrace{(C_i \cdot K^2)}_{①} + \underbrace{(C_i \cdot K^2 + 1)}_{②}\right] \times H \times W \times C_0$$

$$= (2 \times C_i \times K^2 + 1) \times H \times W \times C_0$$

① : Number of multiplication operations

② : Number of addition operations.

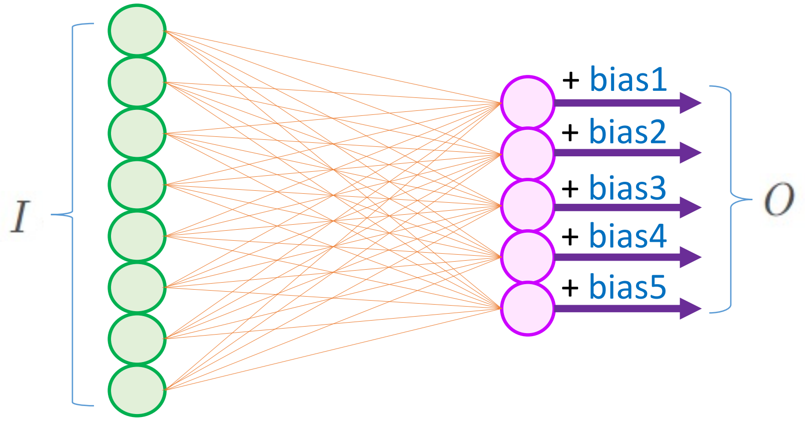


- **Fully connected layer**

$$[\underbrace{I}_{①} + \underbrace{(I + 1)}_{②}] \times O = (2 \times I + 1) \times O$$

① : Number of multiplication operations

② : Number of addition operations.

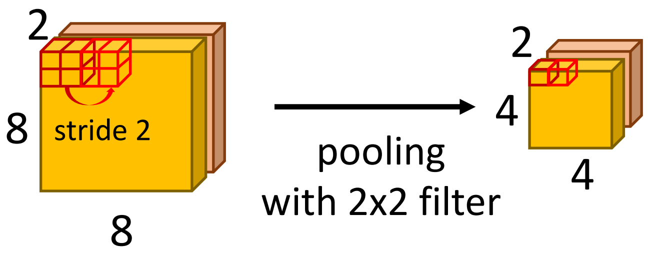


FLOPs (floating point operations)

- We provide the function of FLOPs.
- It return the FLOPs of the model.

```
#Get Model FLOPs function
def get_flops(model):
```

- **In this FLOPs function** :
  - The FLOPs of the **pooling (avg & max)** is the **input size**.
    - Example :



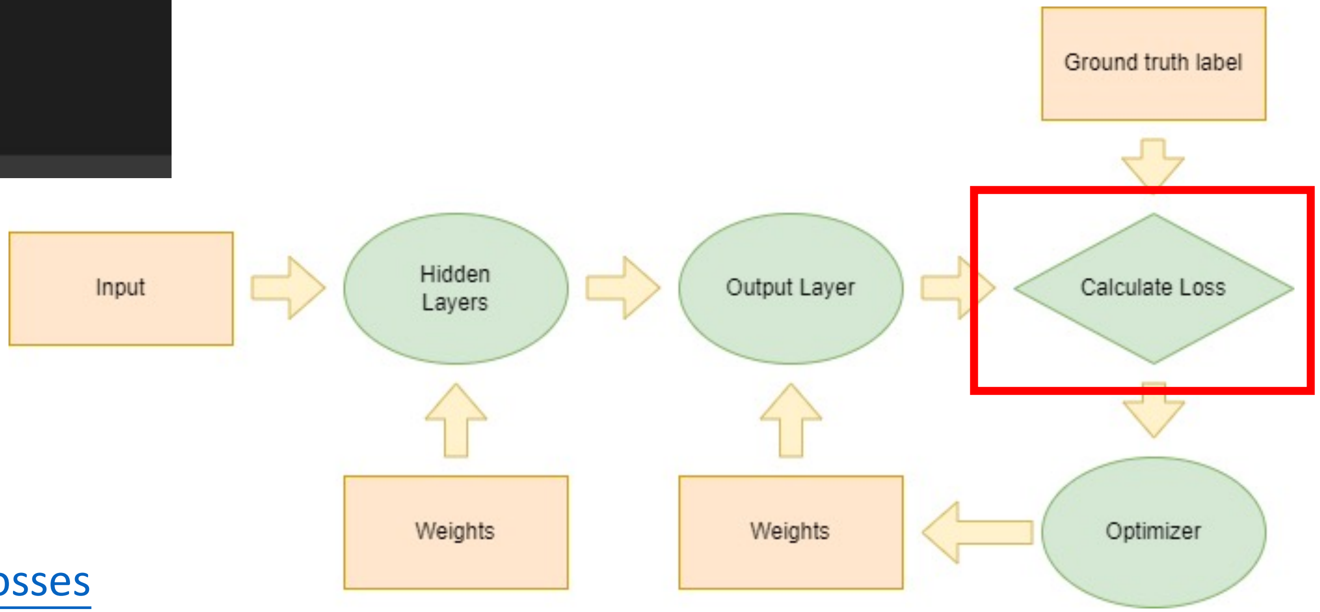
The FLOPs is 8x8x2

  - The activation function **softmax** would increase the FLOPs, and most activation functions wouldn't increase the FLOPs.

Loss function

- To evaluate if model is good/bad.
- Loss means residual between ground truth value and predict value. Thus, we want to minimize residual.
- Choose cross entropy to model our classification problem

```
model.compile(
        loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
```

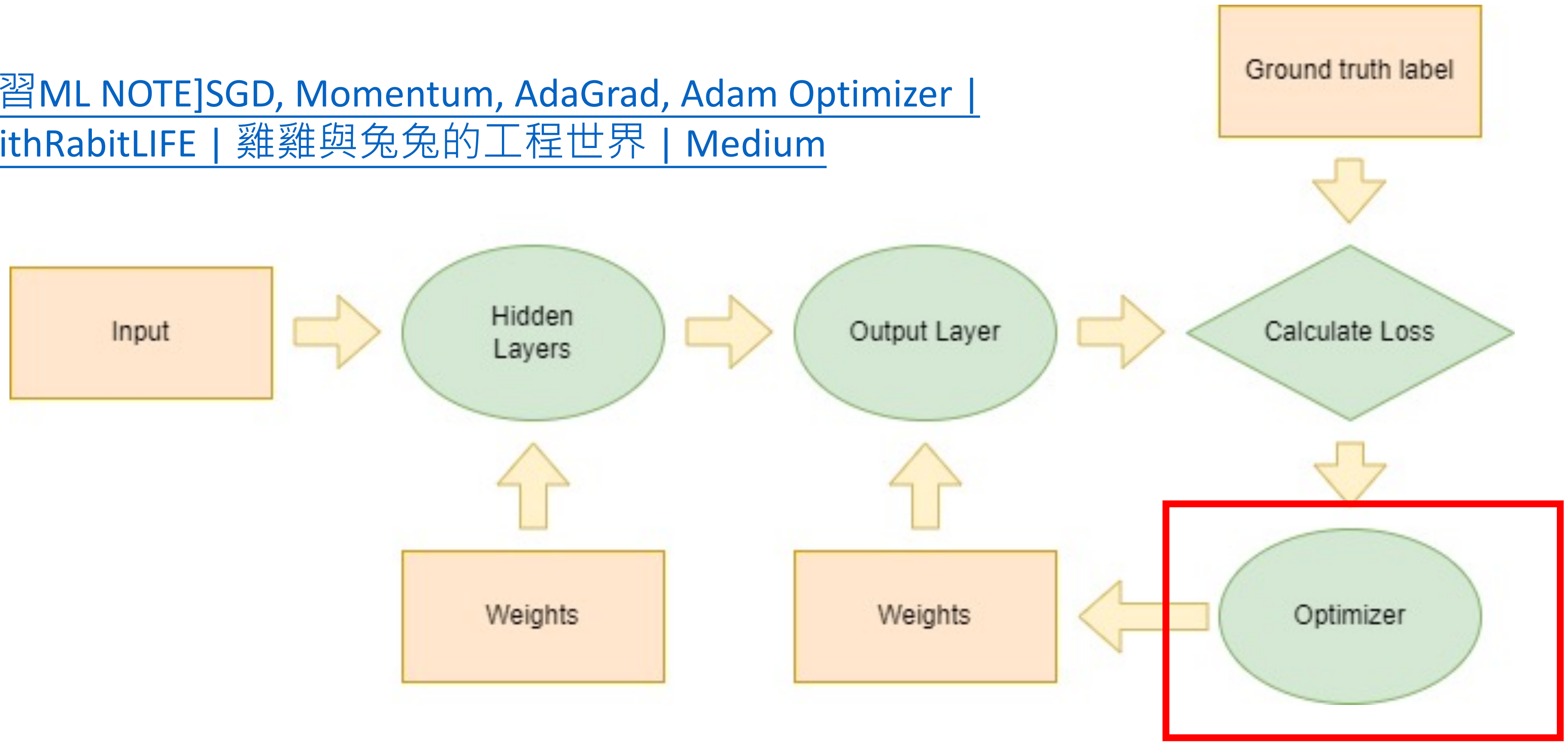


https://www.tensorflow.org/api_docs/python/tf/keras/losses

Optimizer

- **Optimizers** are algorithms or methods used to minimize an error function(*loss function*)or to maximize the efficiency of production.

[機器學習ML NOTE]SGD, Momentum, AdaGrad, Adam Optimizer | by GGWithRabitLIFE | 雞雞與兔兔的工程世界 | Medium



Module: tf.keras.optimizers | TensorFlow Core v2.8.0

Assignment

- Requirement :

1. Need to design 4 CNN models for 4 different datasets listed below.
2. The first 3 models achieve the specified **accuracy** respectively and the **mean average error** of the 4th model need to not exceed 3.
   (You can get the basic grade 70 if you meet the requirement of each dataset)
3. According to the number of **parameter** and **FLOPs**, you will be rated 70~100, the **fewer parameter** and **FLOPs** are the better.

1. Fashion MNIST dataset, an alternative to MNIST **(classification)**
   - **Accuracy > 90%** `metrics=['accuracy']`
2. CIFAR10 small images classification dataset **(classification)**
   - **Accuracy > 65%** `metrics=['accuracy']`
3. CIFAR100 small images classification dataset **(classification)**
   - **Accuracy > 35%** `metrics=['accuracy']`
4. Boston Housing price regression dataset **(regression)**
   - **mean average error < 3** `metrics=['mae']`

You can check this link for datasets Introduction
Module: tf.keras.datasets | TensorFlow Core v2.8.0

Assignment Format

Upload the assignment to Moodle

- File format: (total 4)
  - 1. StudentID_Name_fashion_mnist.ipynb(25%)
  - 2. StudentID_Name_cifar10.ipynb(25%)
  - 3. StudentID_Name_cifar100.ipynb(25%)
  - 4. StudentID_Name_boston_housing.ipynb(25%)

  - ex: N123456789_王大明_cifar100.ipynb

Upload 4 files independently
Don’t need to zip them
Please be attention to the file name

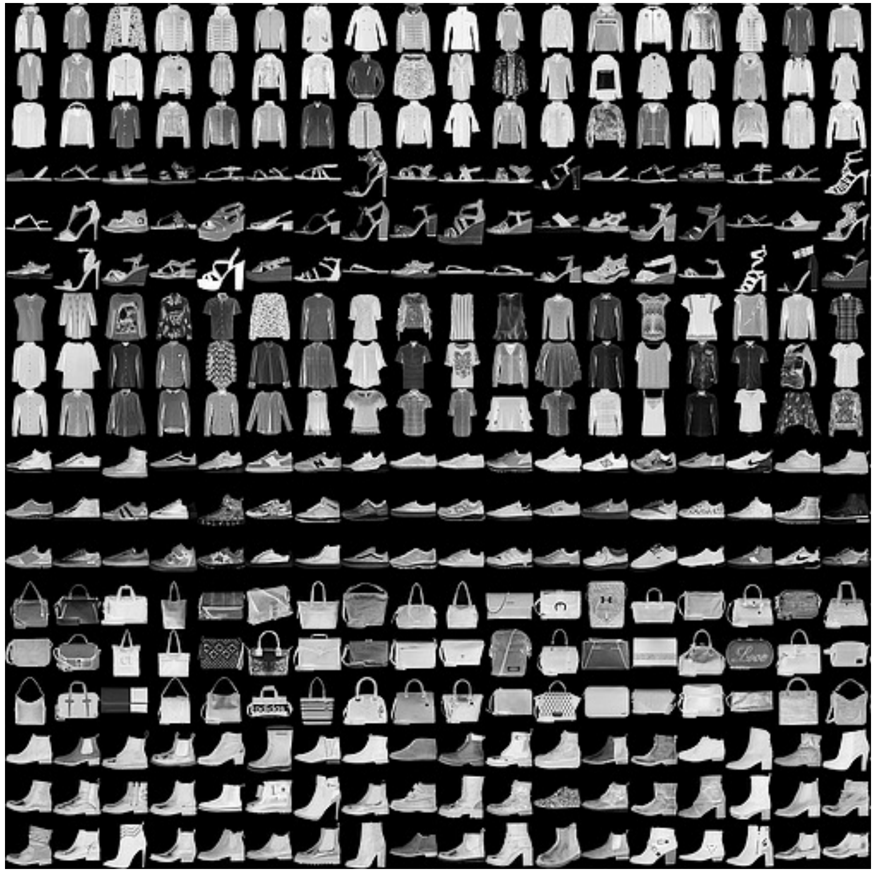
- **Deadline: 3/6(Sun.) 23:59**

Classification & Regression(supervised learning algorithm)

Both the algorithms can be used for forecasting in Machine learning and operate with the labelled datasets.

Problem	Regression problem	Classification problem
Output	Continuous (house price)	Discrete (sells for more or less than the asking price)
Output Layer	Only need one node The output means the prediction quantity value	Need N nodes for N classes Each output means the confidence score of the class
Evaluate (loss function)	mean squared error	cross-entropy
Loss API	loss=tf.keras.losses.MeanSquaredError()	loss=tf.keras.losses.CategoricalCrossentropy()

Dataset : Fashion-MNIST



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Why Fashion-MNIST ?

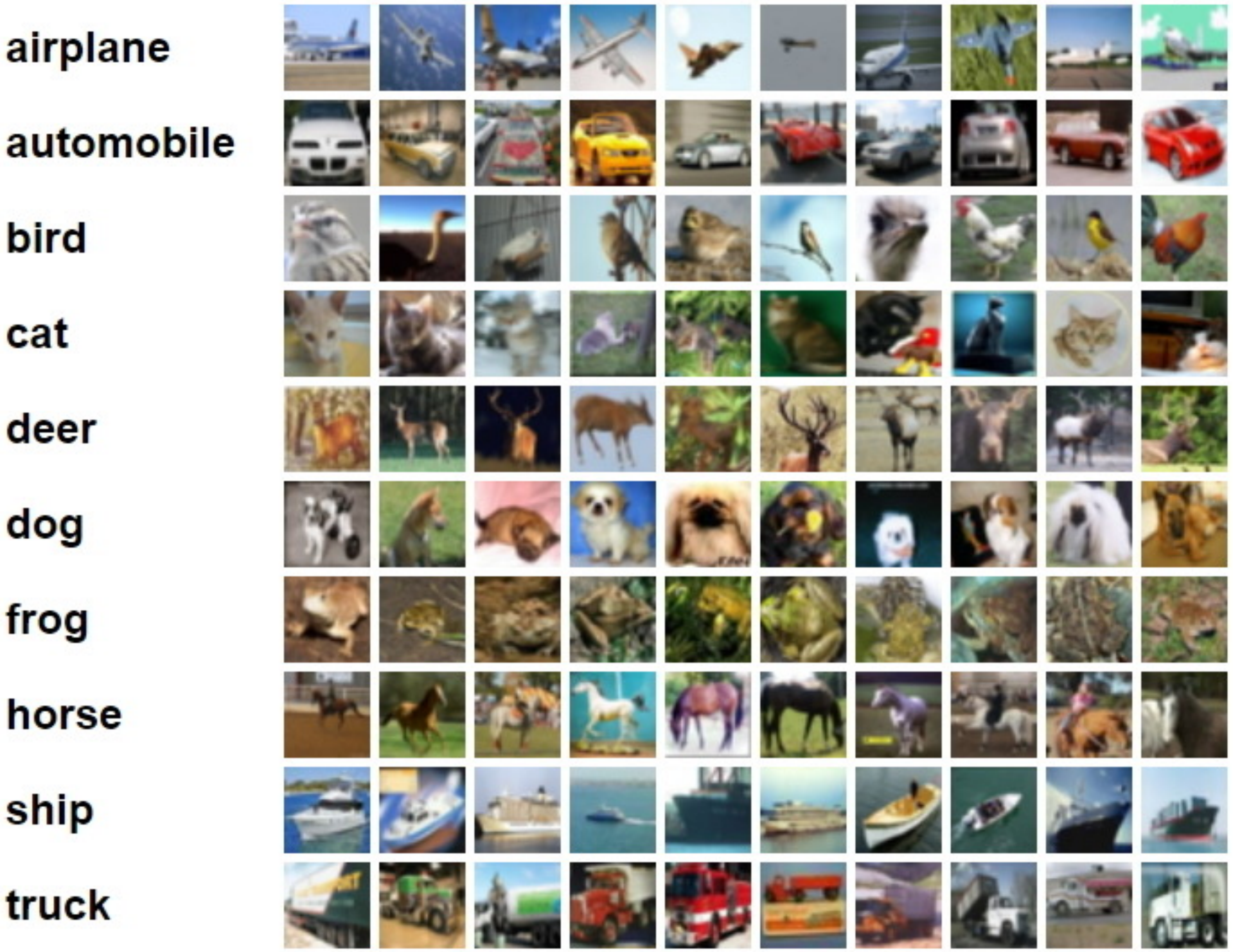
- MNIST is too easy and overused
- MNIST can not represent modern Computer Vision tasks

Fashion-MNIST

- 60,000 training data
- 10,000 testing data
- 10 categories (0~9).
- Each gray-scale image is 28x28.

tf.keras.datasets.fashion_mnist.load_data | TensorFlow Core v2.8.0

Dataset : CIFAR-10

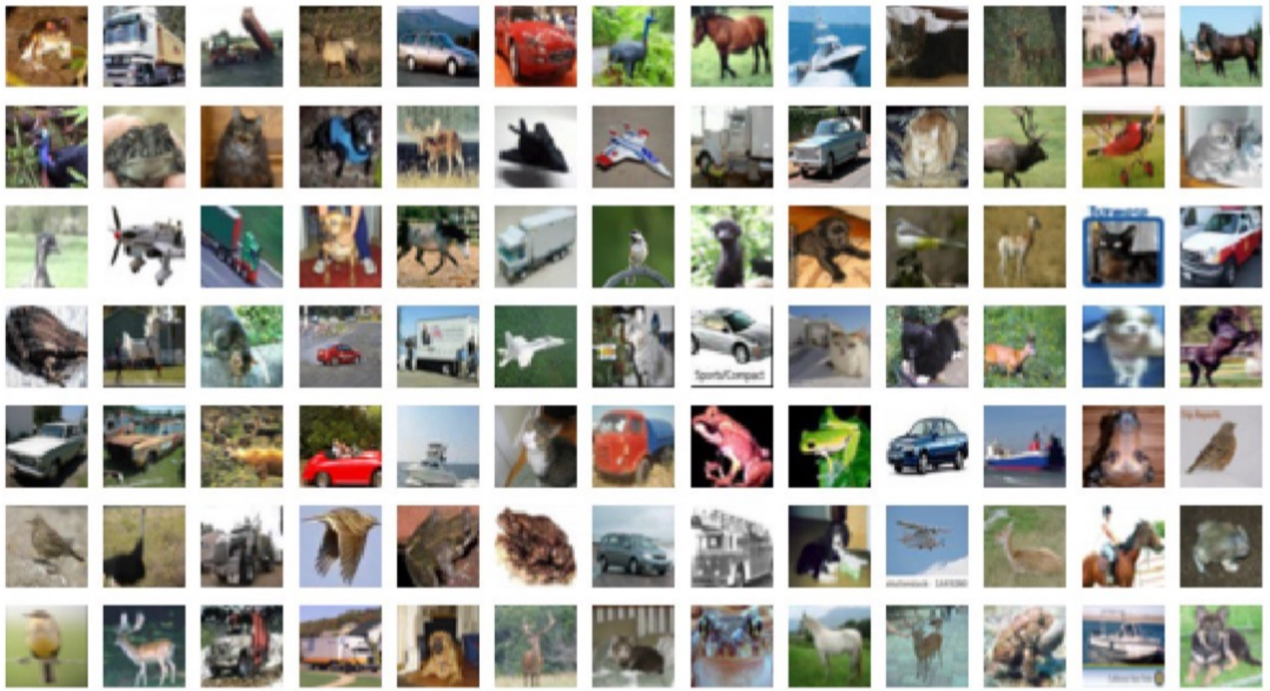


CIFAR-10

- 32x32 color
- 50,000 training images
- 10,000 test images
- labeled over 10 categories.

tf.keras.datasets.cifar100.load_data | TensorFlow Core v2.8.0

Dataset : CIFAR-100



CIFAR-100

- 100 classes containing 600 images each

- 500 training images and 100 testing images per class.
- 100 classes
- Each color image is 32x32.

[tf.keras.datasets.cifar100.load_data | TensorFlow Core v2.8.0]

Dataset : Boston_Housing

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
2	0.00632	18.00	2.310	0	0.5380	6.5750	65.20	4.0900	1	296.0	15.30	396.90	4.98	24.00
3	0.02731	0.00	7.070	0	0.4690	6.4210	78.90	4.9671	2	242.0	17.80	396.90	9.14	21.60
4	0.02729	0.00	7.070	0	0.4690	7.1850	61.10	4.9671	2	242.0	17.80	392.83	4.03	34.70
5	0.03237	0.00	2.180	0	0.4580	6.9980	45.80	6.0622	3	222.0	18.70	394.63	2.94	33.40
6	0.06905	0.00	2.180	0	0.4580	7.1470	54.20	6.0622	3	222.0	18.70	396.90	5.33	36.20
7	0.02985	0.00	2.180	0	0.4580	6.4300	58.70	6.0622	3	222.0	18.70	394.12	5.21	28.70
8	0.08829	12.50	7.870	0	0.5240	6.0120	66.60	5.5605	5	311.0	15.20	395.60	12.43	22.90
9	0.14455	12.50	7.870	0	0.5240	6.1720	96.10	5.9505	5	311.0	15.20	396.90	19.15	27.10
10	0.21124	12.50	7.870	0	0.5240	5.6310	100.00	6.0821	5	311.0	15.20	386.63	29.93	16.50
11	0.17004	12.50	7.870	0	0.5240	6.0040	85.90	6.5921	5	311.0	15.20	386.71	17.10	18.90
12	0.22489	12.50	7.870	0	0.5240	6.3770	94.30	6.3467	5	311.0	15.20	392.52	20.45	15.00
13	0.11747	12.50	7.870	0	0.5240	6.0090	82.90	6.2267	5	311.0	15.20	396.90	13.27	18.90
14	0.09378	12.50	7.870	0	0.5240	5.8890	39.00	5.4509	5	311.0	15.20	390.50	15.71	21.70
15	0.62976	0.00	8.140	0	0.5380	5.9490	61.80	4.7075	4	307.0	21.00	396.90	8.26	20.40
16	0.63796	0.00	8.140	0	0.5380	6.0960	84.50	4.4619	4	307.0	21.00	380.02	10.26	18.20
17	0.62739	0.00	8.140	0	0.5380	5.8340	56.50	4.4986	4	307.0	21.00	395.62	8.47	19.90
18	1.05393	0.00	8.140	0	0.5380	5.9350	29.30	4.4986	4	307.0	21.00	386.85	6.58	23.10
19	0.78420	0.00	8.140	0	0.5380	5.9900	81.70	4.2579	4	307.0	21.00	386.75	14.67	17.50
20	0.80271	0.00	8.140	0	0.5380	5.4560	36.60	3.7965	4	307.0	21.00	288.99	11.69	20.20
21	0.72580	0.00	8.140	0	0.5380	5.7270	69.50	3.7965	4	307.0	21.00	390.95	11.28	18.20
22														

- Samples contain 13 attributes of houses at different locations around the Boston suburbs in the late 1970s.
- Targets are the median values of the houses at a location (in k$).
- 404 training set
- 102 testing set

Attribute :

1.CRIM - per capita crime rate by town
2.ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3.INDUS - proportion of non-retail business acres per town.
4.CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5.NOX - nitric oxides concentration (parts per 10 million)
6.RM - average number of rooms per dwelling
7.AGE - proportion of owner-occupied units built prior to 1940
8.DIS - weighted distances to five Boston employment centres
9.RAD - index of accessibility to radial highways
10.TAX - full-value property-tax rate per $10,000
11.PTRATIO - pupil-teacher ratio by town
12.B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13.LSTAT - % lower status of the population

Label :

MEDV - Median value of owner-occupied homes in $1000's

tf.keras.datasets.boston_housing.load_data | TensorFlow Core v2.8.0

Dataset : Boston_Housing

- This is a regression problem, so we don’t use **cross-entropy** to be the loss function and don’t use **accuracy** to be the evaluation metric .
- We ask **Mean average errore(mae)** to be the **metrics** in the dataset in the assignment.

```
model.compile(
            loss='mean_squared_error',
            optimizer='adam',
            metrics=['mae'])
```

[【Day 20】 Google ML - Lesson 6 - 使用損失函數(Loss Functions)來評估ML模型的好壞吧! MSE, RMSE, Cross Entropy的計算方法與特性]

Advanced Topics

- Online CNN explainer: CNN Explainer (poloclub.github.io)
- Professor Hung-yi Lee machine learning course: https://www.youtube.com/watch?v=Ye018rCVvOo&list=PLJV_el3uVTsMhtt7_Y6sgTHGHp1Vb2P2J

- Supplement:
- https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/classification.ipynb