

# Class 03: CNN 模型

---

林義隆  
義守大學 資訊工程系 副教授

# **Python 程式設計-資料型態**

---

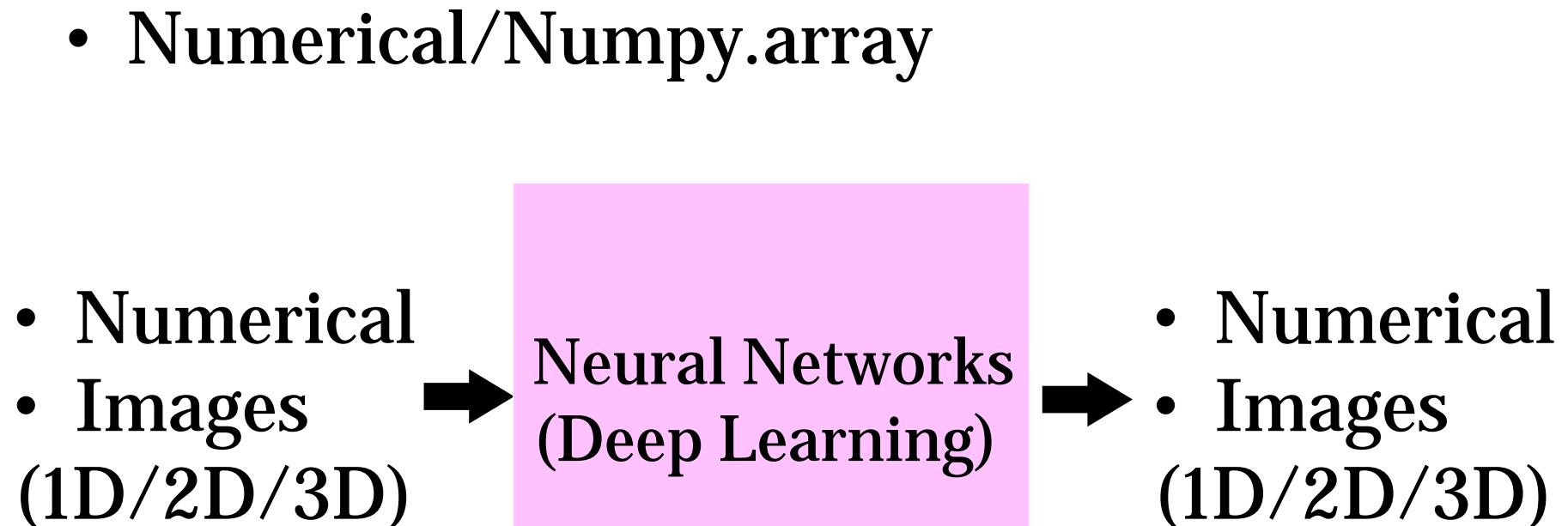
# 課程回顧

---

- 開發環境安裝與基礎 Open CV套件
  - VS Code/Pycharm
  - Jupyter Notebook
  - Virtualenv
  - Python
    - Tuple/Set/List /Dictionary/Numpy
    - Package/Module
  - Open CV
    - Read/Save
  - Image Filters

# Data Structure

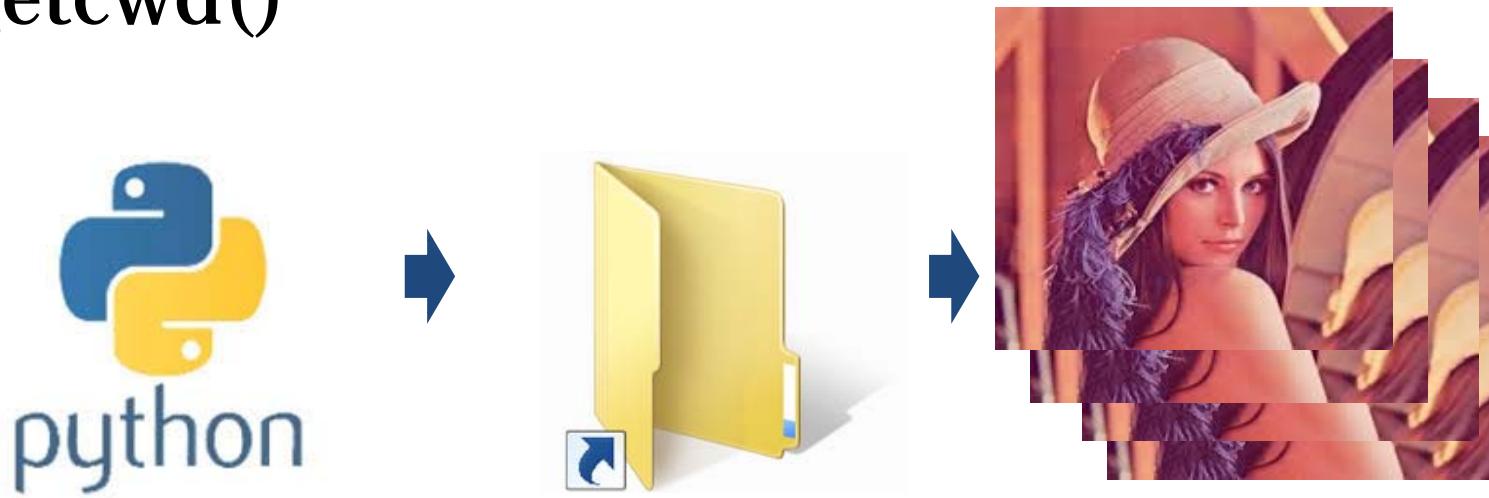
---



# Path

---

- `.listdir()`
- `.chdir(path)`
- `.getcwd()`



# Open Image

---

- matplotlib: Pylab.imread
- Python Image Libaray: PIL.Image
- open computer version: cv2.imread

# Container

---

- tuple(,) #unchanged and order
- List[,] #changed and order
  - add/update/del
  - len()/index()/append()/insert()/remove()
  - del()/pop()/sort()/+
- dict{**key:value**} #search value from key
  - update()/copy()/get()/del
- set() or {} #find existing data
  - | / & / - / ^
- numpy.ndarray #image

# Question

---

- a=()
- b=(5)
- c=(5,)
- d=[]
- e= {}
- e= {1}
- e= {a:2}
- array([1,2,3])

# Answer

---

- a=() #tuple
- b=(5) #int
- c=(5,) #tuple
- d=[] #list
- e= {} #dictionary
- e= {1} #set
- e= {a:2} #dictionary

# Question

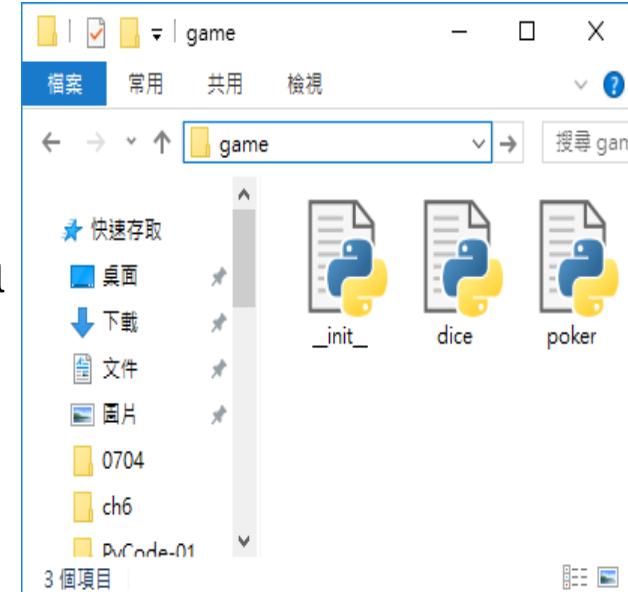
---

- import numpy as np
- a=np.array([1,2])
- a=np.array([[1,2]])
- a=np.array([[[1,2]]])
- a=np.array([[1],[2]])
- a=np.array([[[1],[2]]])
- a=np.array([[[1]],[[2]]])

# Module & Package

---

- module (file)
  - import module as alias
  - from module import function
- Package
  - multi-module
    - add “`__init__.py`” file
  - from package import module1, module2, ...



# 特徵映射與濾波器

---

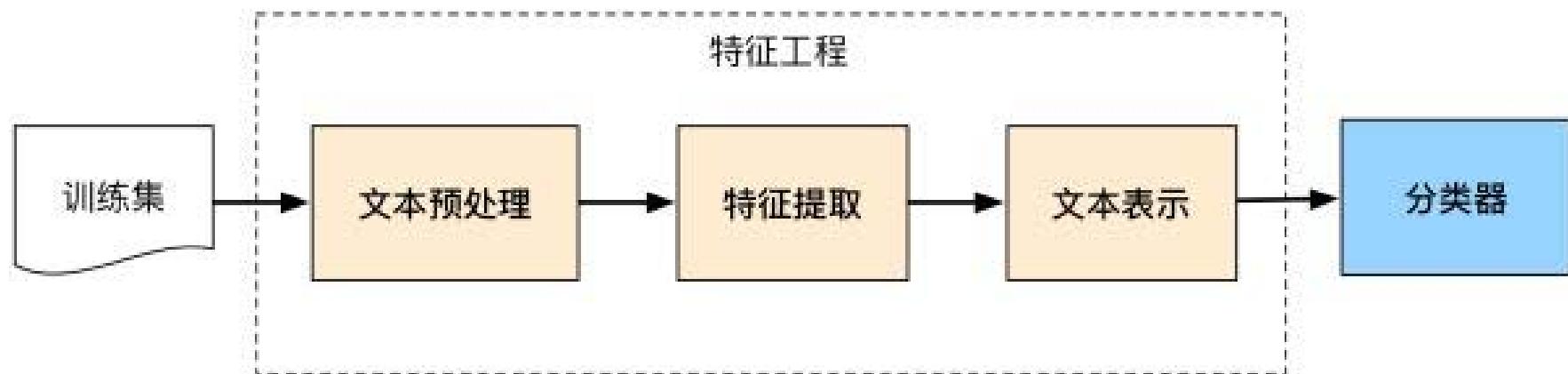
# 特徵映射

---

- 文字:
  - Open file
  - Str/List/Dictionary/Set
- 影像:
  - Numpy
- 視訊:
  - Numpy
- 語音:
  - Numpy

# 文字模型

- 傳統文本分類方法



<https://zhuanlan.zhihu.com/p/76003775>

# Kaggle Text Classification

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [4]:  
    data['category'].value_counts()  
  
Out[4]:  
    sport      511  
    business   510  
    politics   417  
    tech       401  
    entertainment 386  
    Name: category, dtype: int64
```

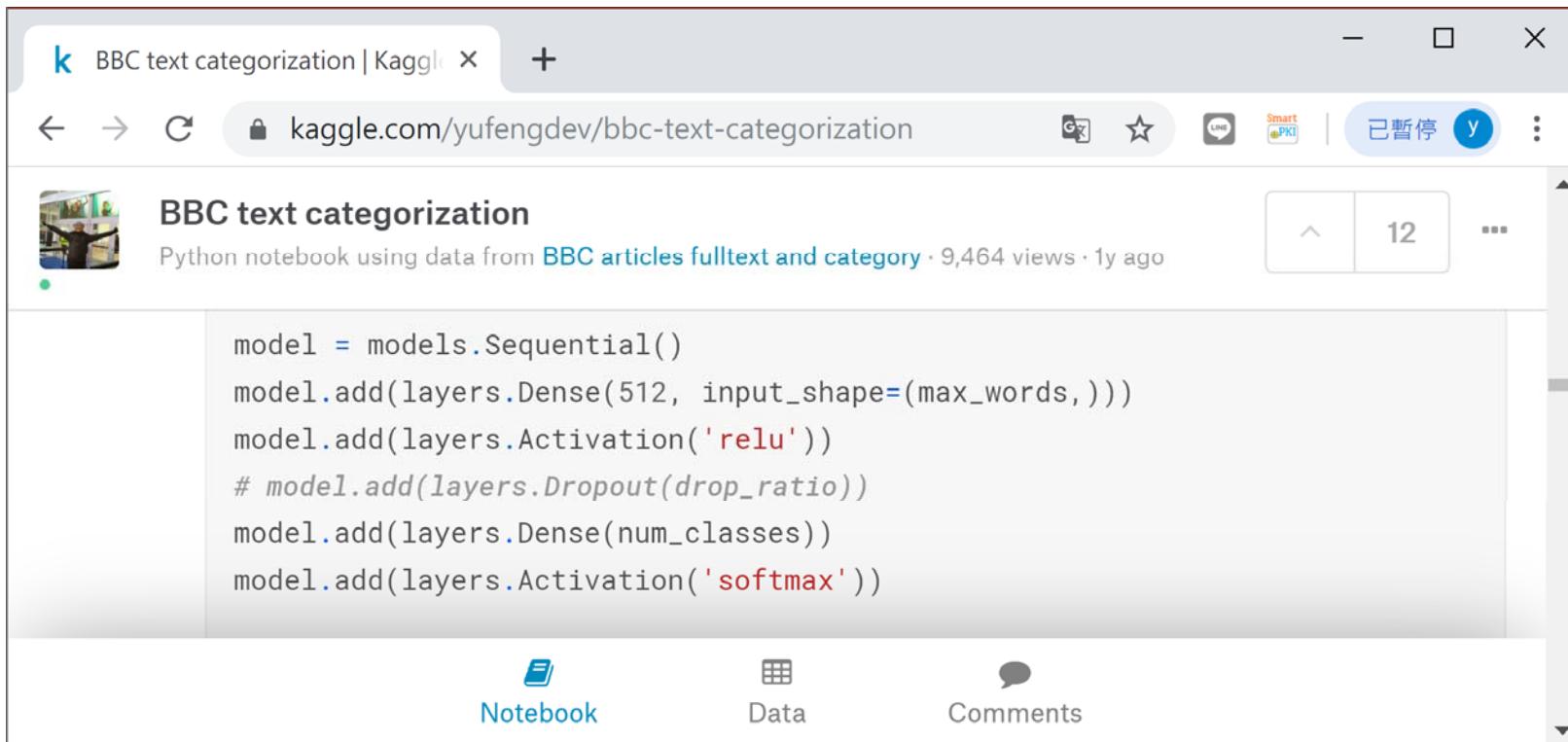
```
In [5]:  
    train_size = int(len(data) * .8)  
    print ("Train size: %d" % train_size)
```

Below the notebook area, there are three navigation buttons: Notebook, Data, and Comments.

<https://www.kaggle.com/yufengdev/bbc-text-categorization>

# Kaggle Text Classification

---



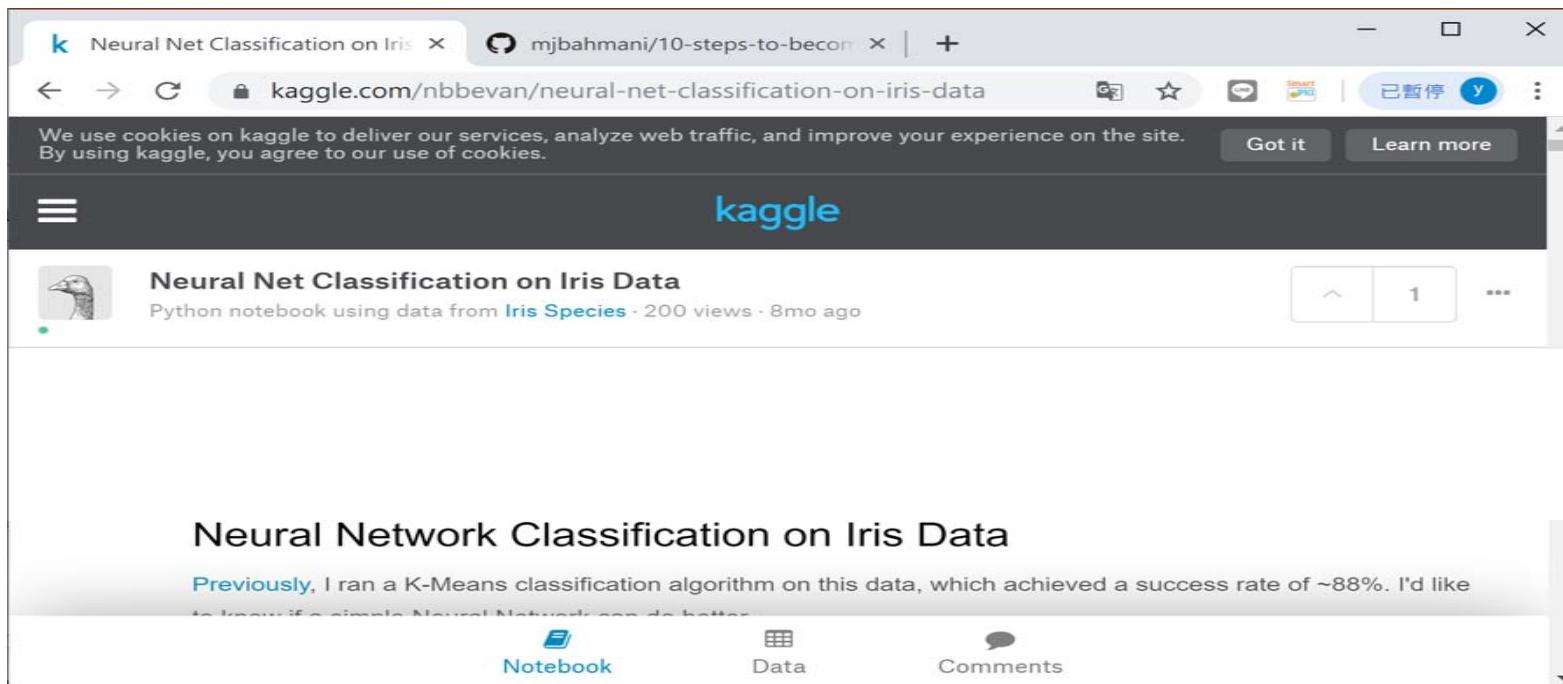
The screenshot shows a Kaggle notebook titled "BBC text categorization". The notebook is a Python script for text classification using a sequential model. The code includes importing TensorFlow, defining the model architecture (Sequential with Dense layers and Activation functions), and specifying the number of classes. Below the code, there are tabs for "Notebook", "Data", and "Comments".

```
model = models.Sequential()
model.add(layers.Dense(512, input_shape=(max_words,)))
model.add(layers.Activation('relu'))
# model.add(layers.Dropout(drop_ratio))
model.add(layers.Dense(num_classes))
model.add(layers.Activation('softmax'))
```

<https://www.kaggle.com/yufengdev/bbc-text-categorization>

# IRIS Classification

- Neural Network



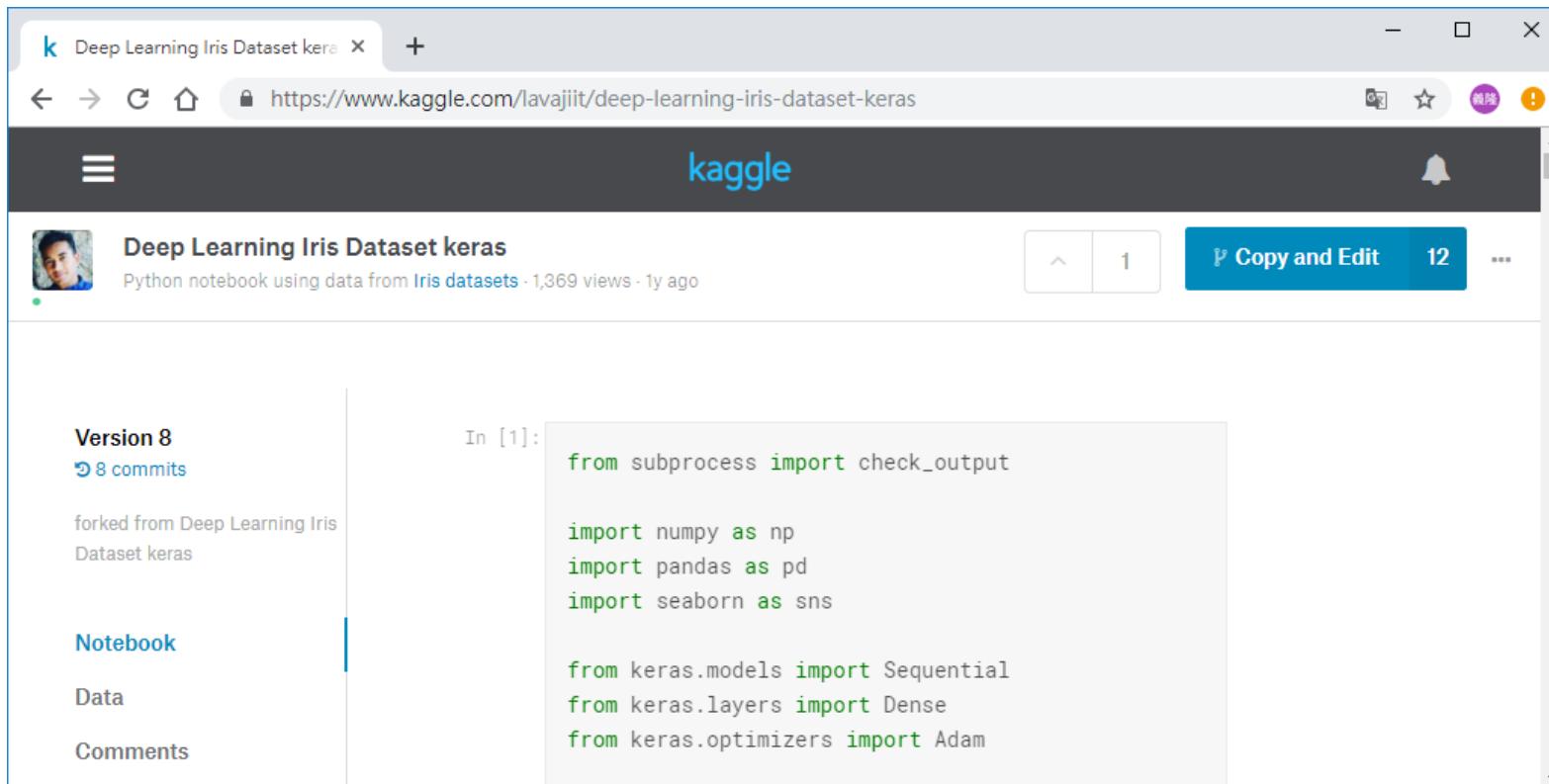
<https://www.kaggle.com/anthonyhills/classifying-species-of-iris-flowers>

<https://www.kaggle.com/mjbahmani/20-ml-algorithms-15-plot-for-beginners>

<https://www.kaggle.com/nbbevan/neural-net-classification-on-iris-data>

# IRIS Classification

- Neural Network



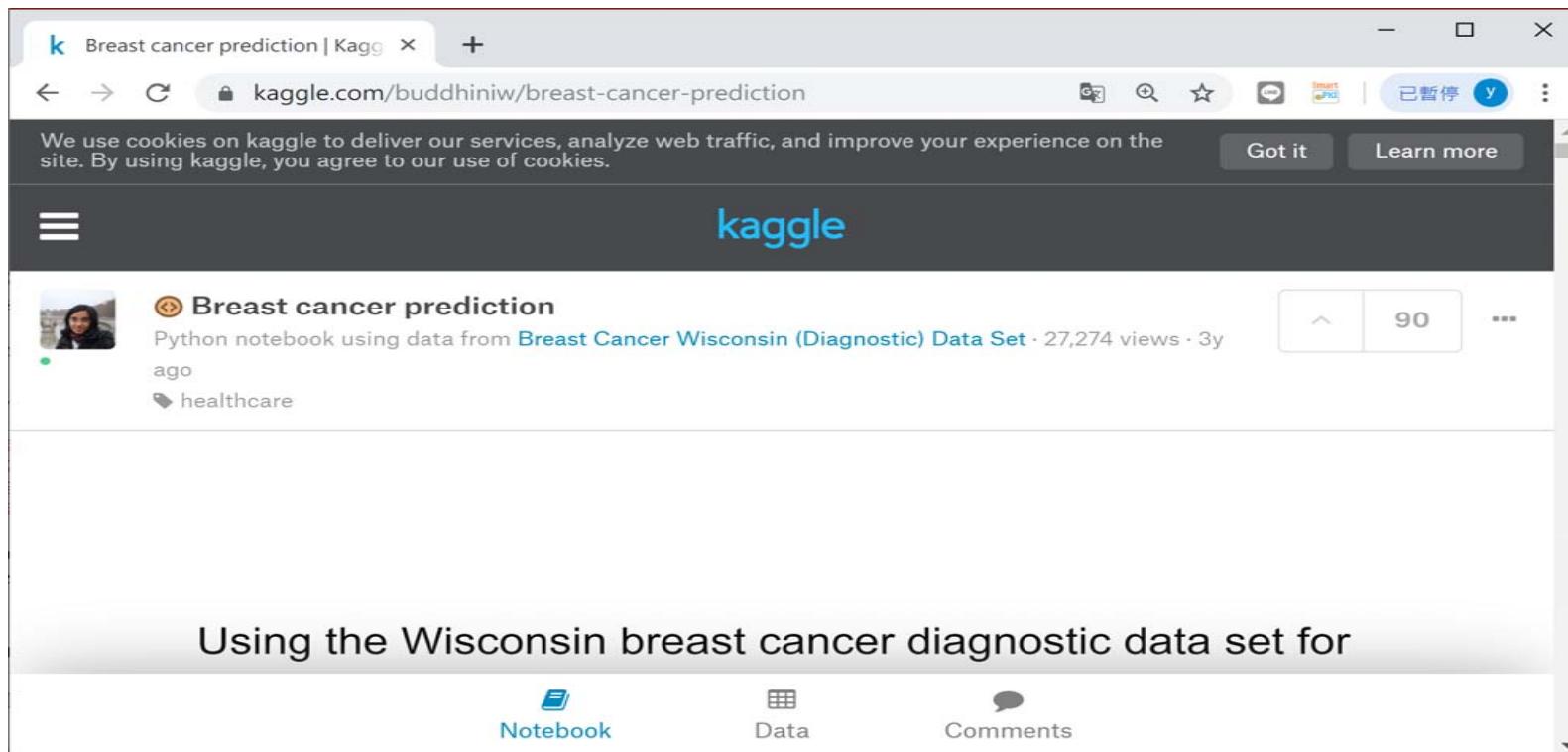
The screenshot shows a Kaggle notebook interface. The title bar says "Deep Learning Iris Dataset keras". The main content area displays a Python notebook with the following code:

```
from subprocess import check_output  
  
import numpy as np  
import pandas as pd  
import seaborn as sns  
  
from keras.models import Sequential  
from keras.layers import Dense  
from keras.optimizers import Adam
```

<https://www.kaggle.com/lavajiit/deep-learning-iris-dataset-keras>

# Breast Cancer Dataset

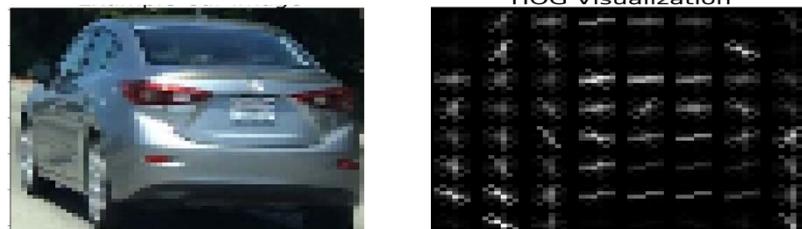
---



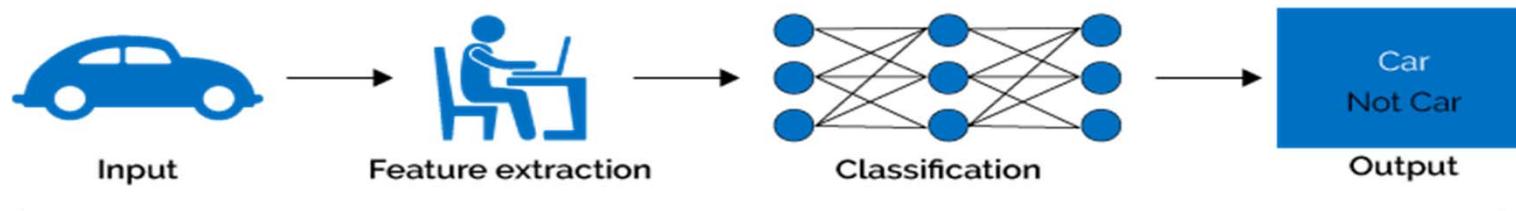
<https://www.kaggle.com/buddhiniw/breast-cancer-prediction>

# 影像處理模型

- 傳統影像處理



Machine Learning

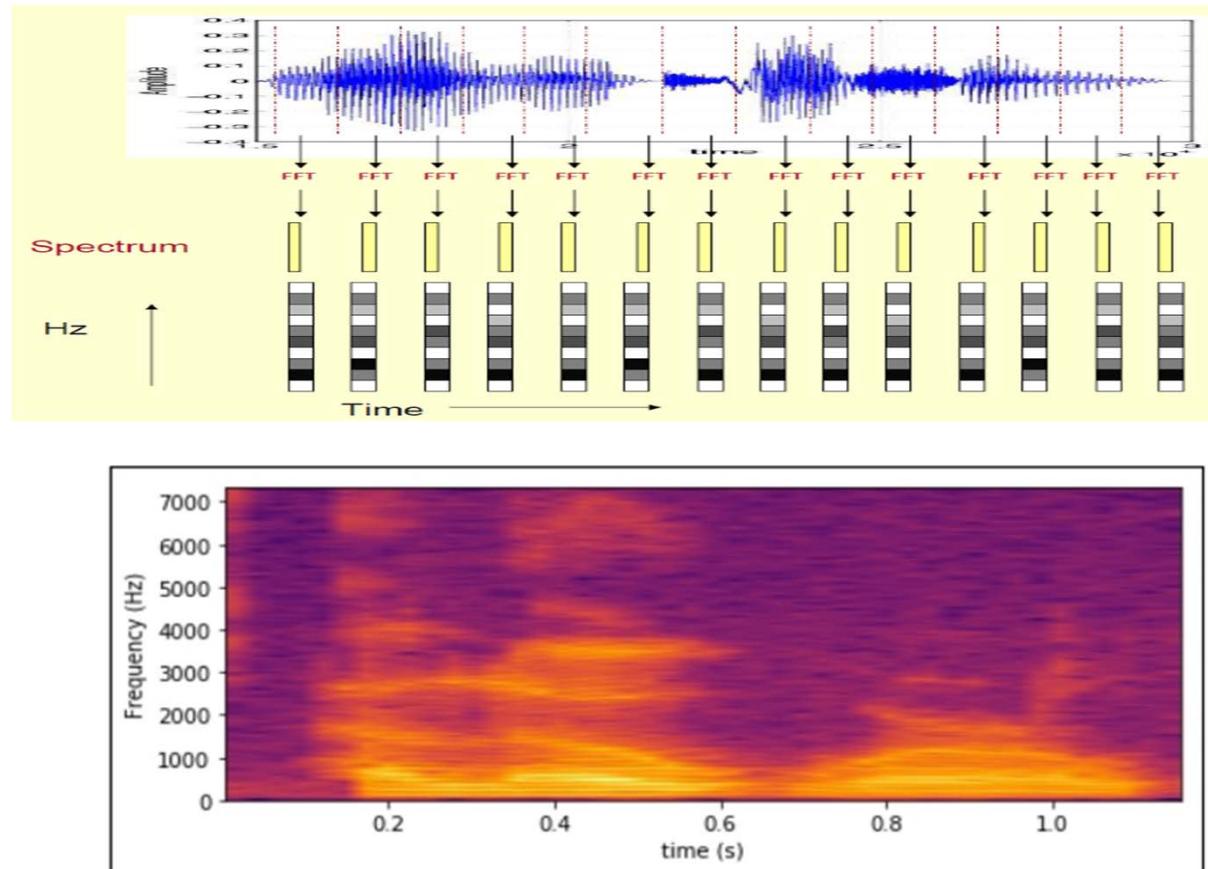


Deep Learning



圖片來源：<https://semiengineering.com/deep-learning-spreads/>

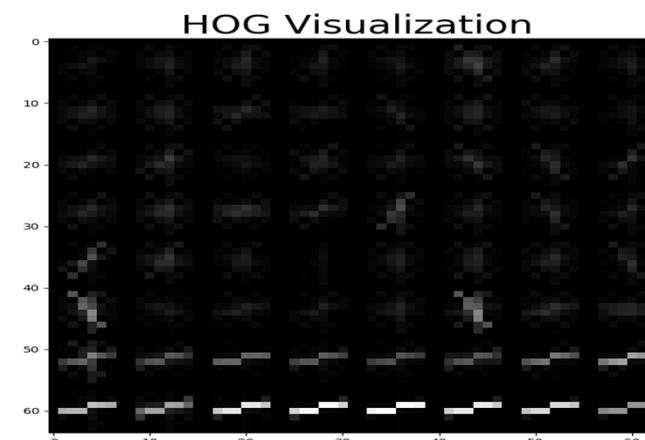
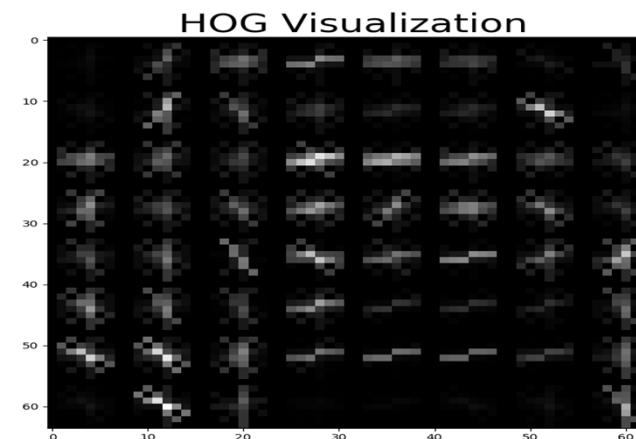
# 語言處理模型



Mel spectrum of speech signal from the previous figure

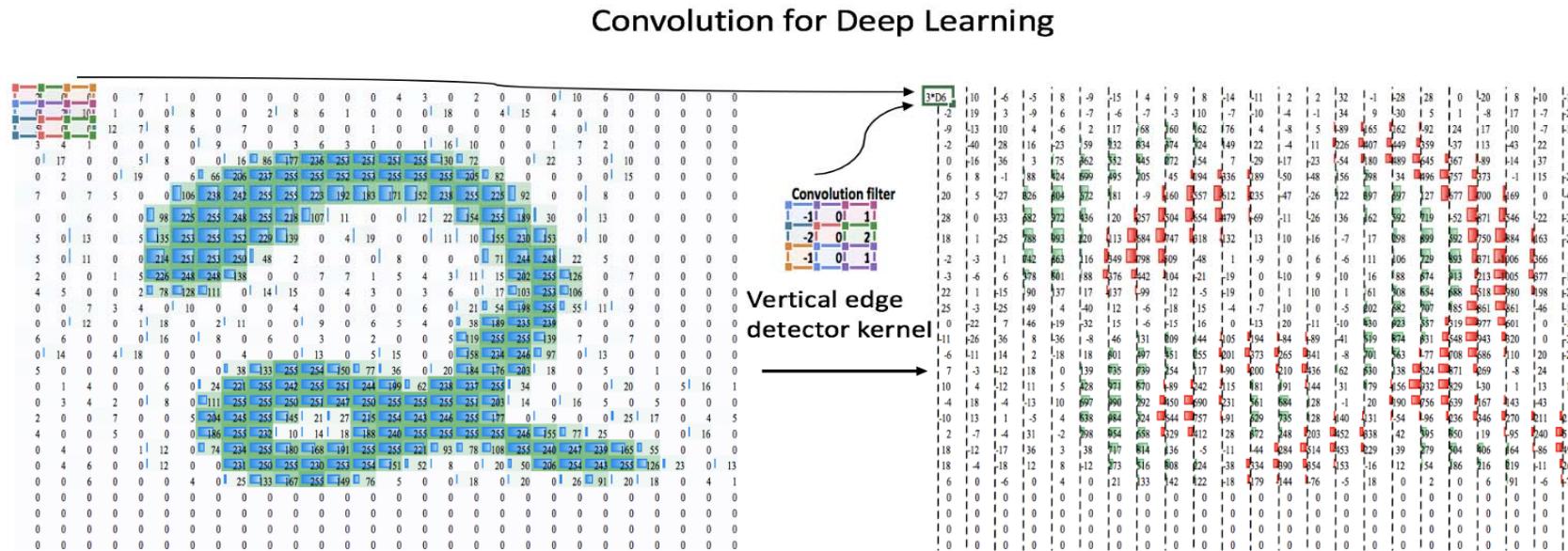
# Feature Extraction

---



# Convolution

- Relative pixel densities (magnitude of numbers)



# Convolution Layers

- Shares weights between local regions

$$\begin{array}{|c|c|c|} \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array} \quad \times \quad \begin{array}{|c|c|c|} \hline 1 & -1 & -1 \\ \hline -1 & 1 & -1 \\ \hline -1 & -1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 3 & -1 & -3 & -1 \\ \hline -3 & 1 & 0 & -3 \\ \hline -3 & -3 & 0 & 1 \\ \hline 3 & 2 & -2 & -1 \\ \hline \end{array}$$

The diagram illustrates a convolution operation. On the left is a 6x6 input matrix with values 1, 0, 0 in the top-left corner, and 1, 0, 1 in the third row. A 3x3 kernel matrix with values 1, -1, -1; -1, 1, -1; and -1, -1, 1 is shown next. A circled 'x' symbol indicates the multiplication. An equals sign leads to the output matrix, where the first value is 3 (from 1\*1+0\*-1+0\*-1), followed by -1, -3, -1, -3, 0, -3, -3, -3, 0, 1, 3, 2, -2, -1.

# Convolution Layers

- Shares weights between local regions

$$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 1 & -1 & -1 \\ \hline -1 & 1 & -1 \\ \hline -1 & -1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 3 & -1 & -3 & -1 \\ \hline -3 & 1 & 0 & -3 \\ \hline -3 & -3 & 0 & 1 \\ \hline 3 & 2 & -2 & -1 \\ \hline \end{array}$$

# Convolution Layers

- Shares weights between local regions

$$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 1 & -1 & -1 \\ \hline -1 & 1 & -1 \\ \hline -1 & -1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 3 & -1 & -3 & -1 \\ \hline -3 & 1 & 0 & -3 \\ \hline -3 & -3 & 0 & 1 \\ \hline 3 & 2 & -2 & -1 \\ \hline \end{array}$$

# Convolution Layers

- Shares weights between local regions

$$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 1 & -1 & -1 \\ \hline -1 & 1 & -1 \\ \hline -1 & -1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 3 & -1 & -3 & -1 \\ \hline -3 & 1 & 0 & -3 \\ \hline -3 & -3 & 0 & 1 \\ \hline 3 & 2 & -2 & -1 \\ \hline \end{array}$$

# Convolution Layers

- Shares weights between local regions

$$\begin{matrix} 1 & 0 & 0 & 0 & 0 & 1 \\ \boxed{0} & \color{red}{1} & \color{red}{0} & 0 & 1 & 0 \\ \color{red}{0} & \color{red}{0} & \color{red}{1} & 1 & 0 & 0 \\ \color{red}{1} & \color{red}{0} & \color{red}{0} & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{matrix} \times \begin{matrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{matrix} = \begin{matrix} 3 & -1 & -3 & -1 \\ \color{red}{-3} & 1 & 0 & -3 \\ -3 & -3 & 0 & 1 \\ 3 & 2 & -2 & -1 \end{matrix}$$

# Convolution Layers

- Shares weights between local regions

$$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 1 & -1 & -1 \\ \hline -1 & 1 & -1 \\ \hline -1 & -1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 3 & -1 & -3 & -1 \\ \hline -3 & 1 & 0 & -3 \\ \hline -3 & -3 & 0 & 1 \\ \hline 3 & 2 & -2 & -1 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|c|c|c|} \hline 3_0 & 3_1 & 2_2 & 1 & 0 \\ \hline 0_2 & 0_2 & 1_0 & 3 & 1 \\ \hline 3_0 & 1_1 & 2_2 & 2 & 3 \\ \hline 2 & 0 & 0 & 2 & 2 \\ \hline 2 & 0 & 0 & 0 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline 12.0 & 12.0 & 17.0 \\ \hline 10.0 & 17.0 & 19.0 \\ \hline 9.0 & 6.0 & 14.0 \\ \hline \end{array}$$

# Convolution Layer

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

0	-1	0
-1	5	-1
0	-1	0

Kernel Matrix

320					

Output Matrix

$$\begin{aligned} & 0 * 0 + 0 * -1 + 0 * 0 \\ & + 0 * -1 + 105 * 5 + 102 * -1 \\ & + 0 * 0 + 103 * -1 + 99 * 0 = 320 \end{aligned}$$

Convolution with horizontal and  
vertical strides = 1

[http://machinelearningguru.com/computer\\_vision/basics/convolution/convolution\\_layer.html](http://machinelearningguru.com/computer_vision/basics/convolution/convolution_layer.html)

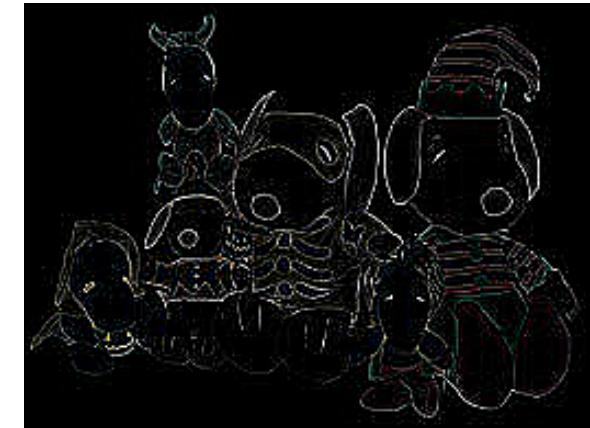
# Filter

---

- Filters (Mask)

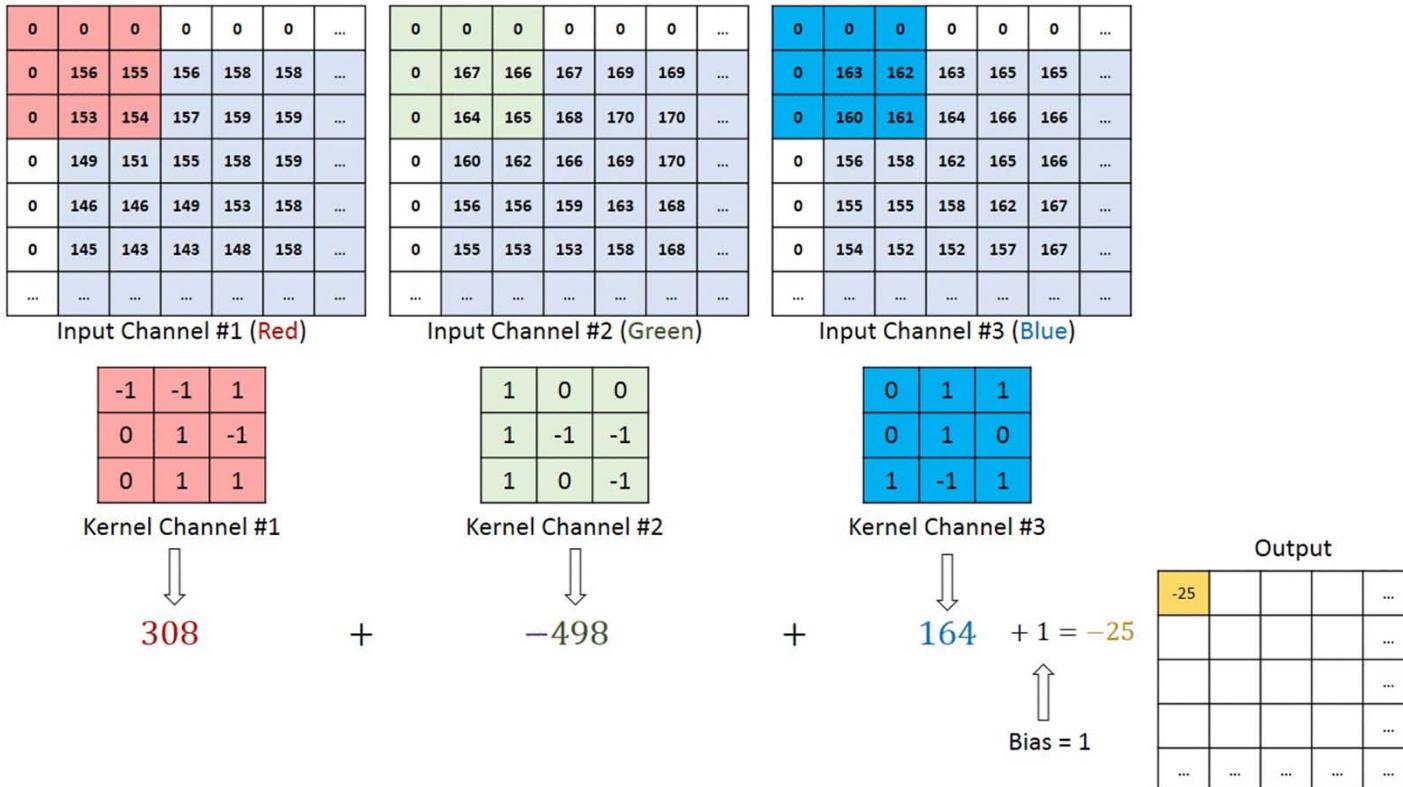


Prewitt/Sobel



Laplacian

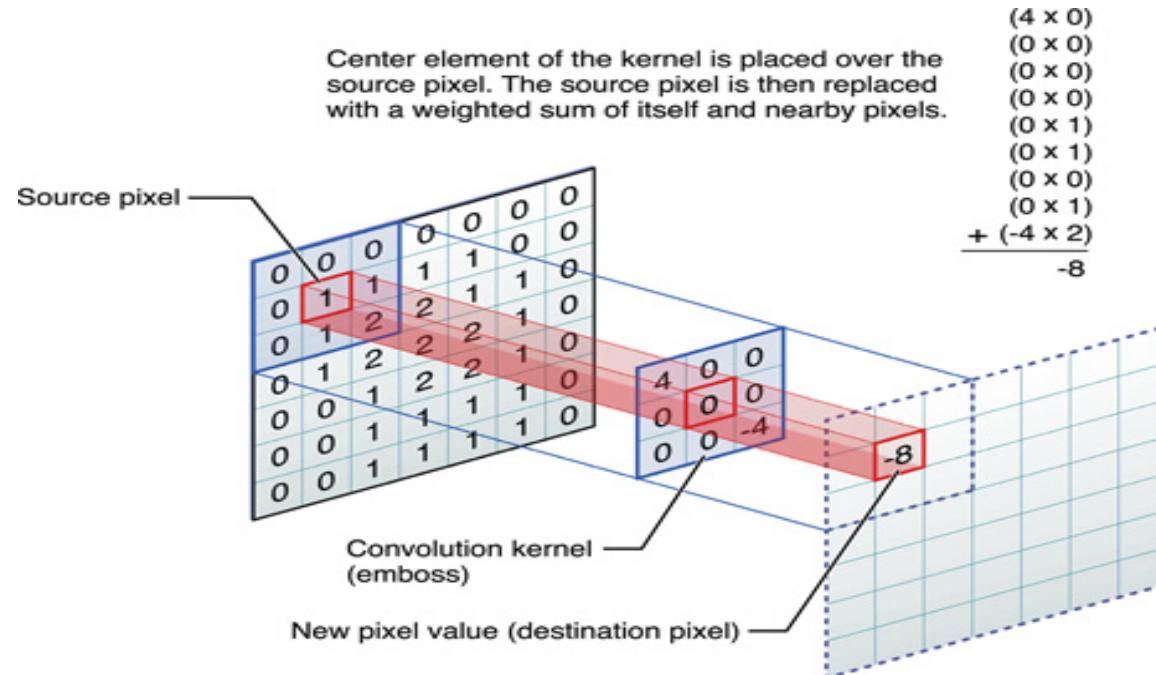
# Convolution



[http://machinelearningguru.com/computer\\_vision/basics/convolution/convolution\\_layer.html](http://machinelearningguru.com/computer_vision/basics/convolution/convolution_layer.html)

# Smoothing

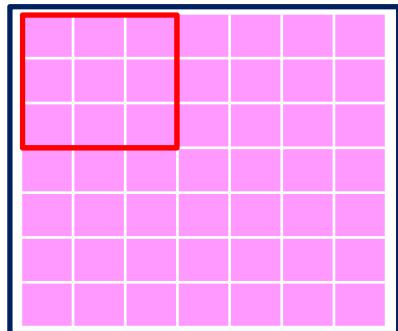
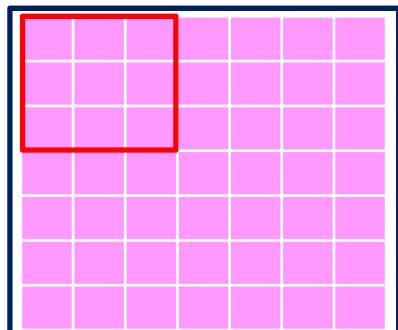
- Denoise & Low Contrast
- Methods
  - Filter
  - Erode
  - Dilation



<http://goo.gl/dck6tD>

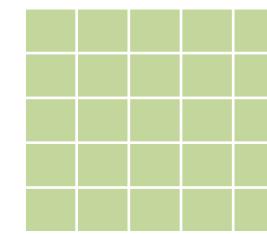
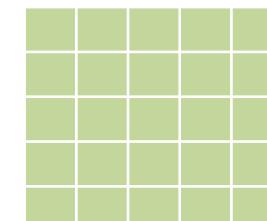
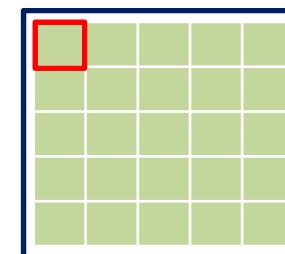
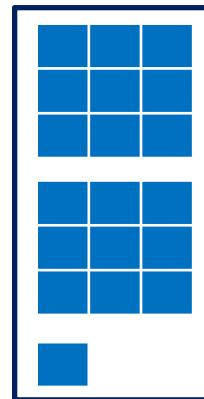
# Convolution Layers

Fully connected:  $(7 \times 7 \times 2 + 1) \times (5 \times 5 \times 3) = 7425$



$7 \times 7 \times 2$

Layer(i-1)



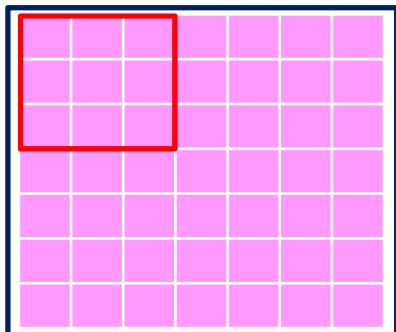
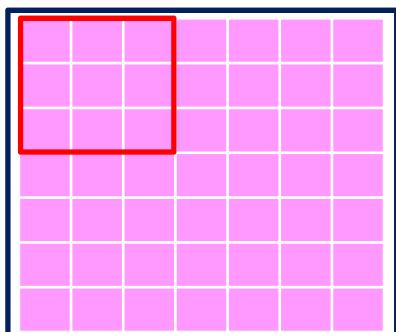
$5 \times 5 \times 3$

Layer(i)



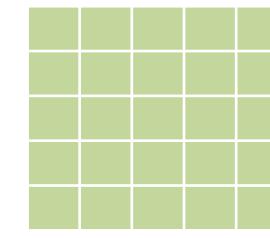
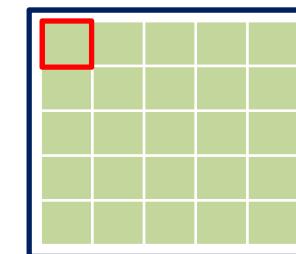
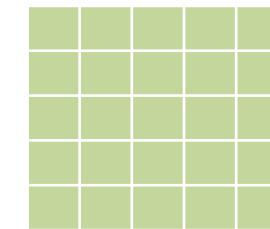
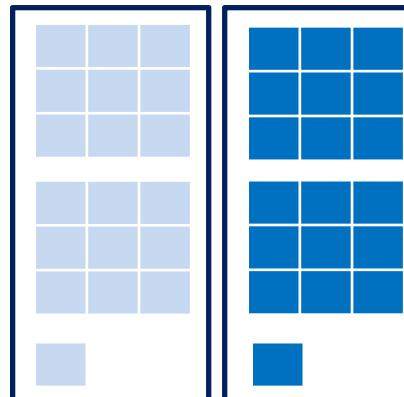
# Convolution Layers

Fully connected:  $(7 \times 7 \times 2) \times (5 \times 5 \times 3) = 7350$



$7 \times 7 \times 2$

Layer(i-1)



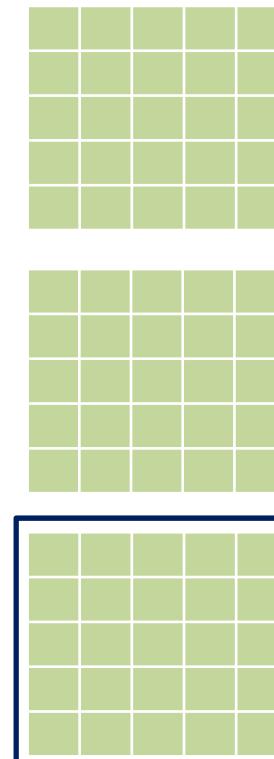
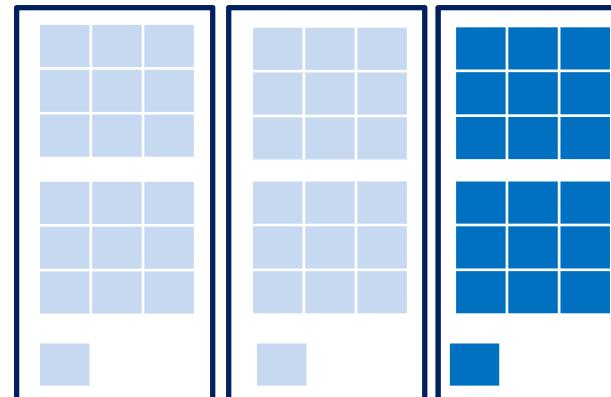
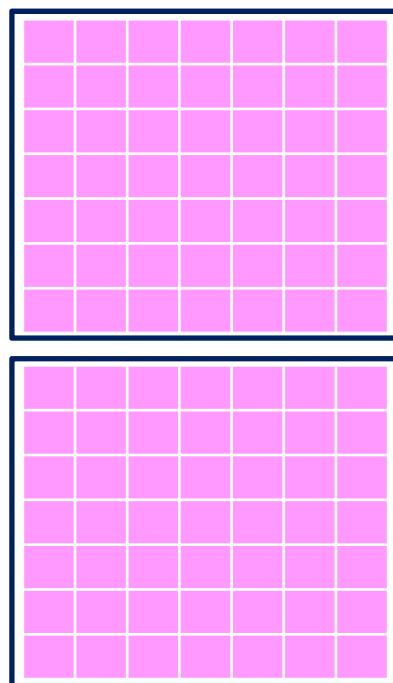
$5 \times 5 \times 3$

Layer(i)

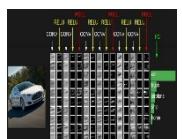


# Convolution Layers

Fully connected:  $(7 \times 7 \times 2) \times (5 \times 5 \times 3) = 7350$

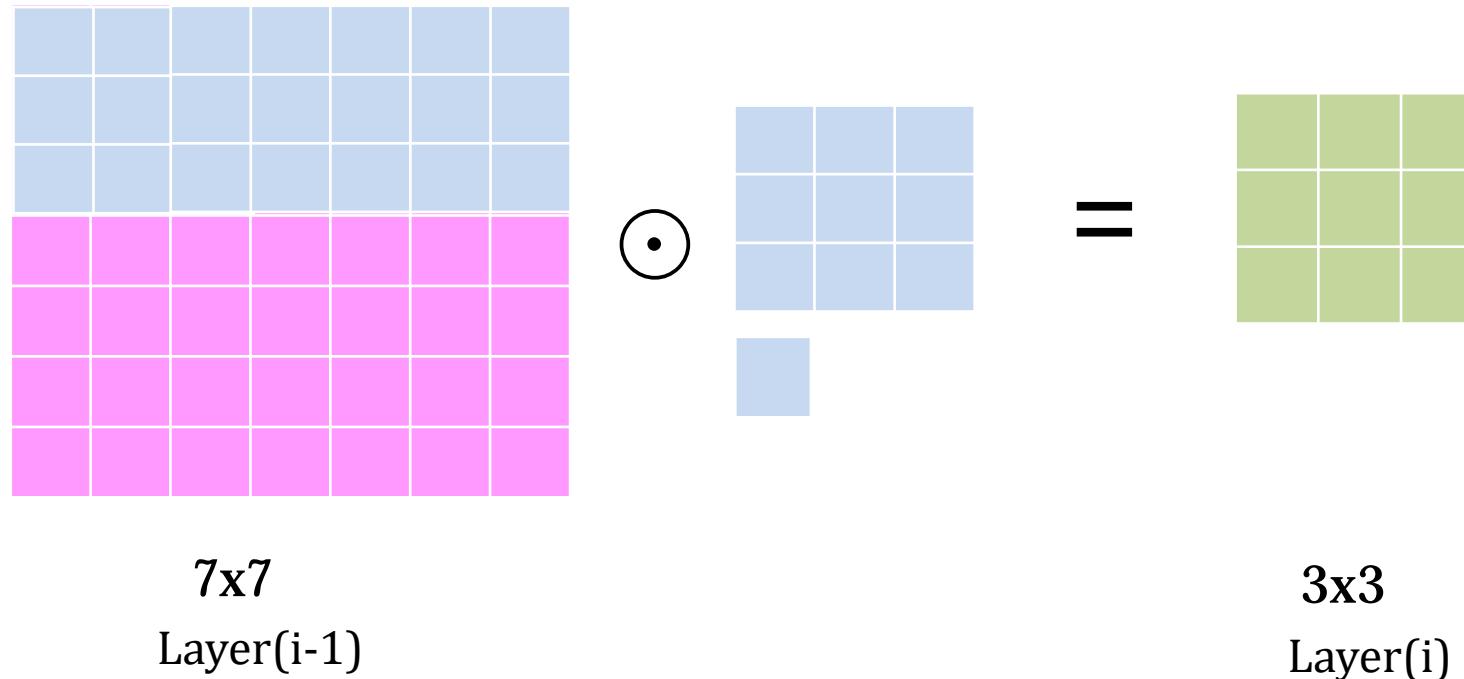


Locally connected:  $3 \times (3 \times 3 + 3 \times 3 + 1) = 57$



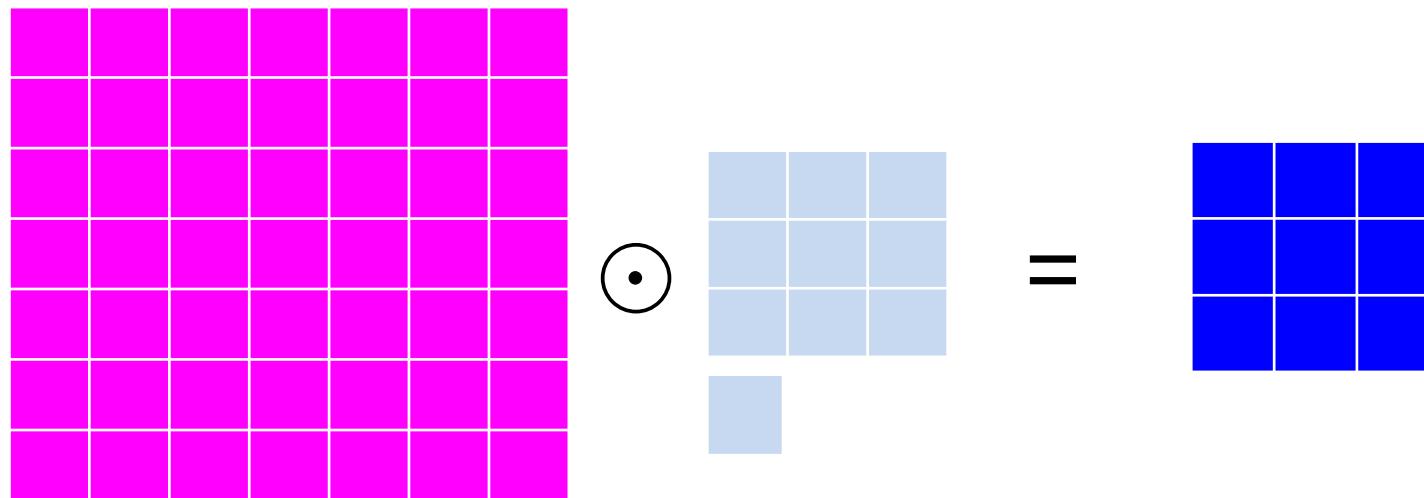
# Convolution Layers

---



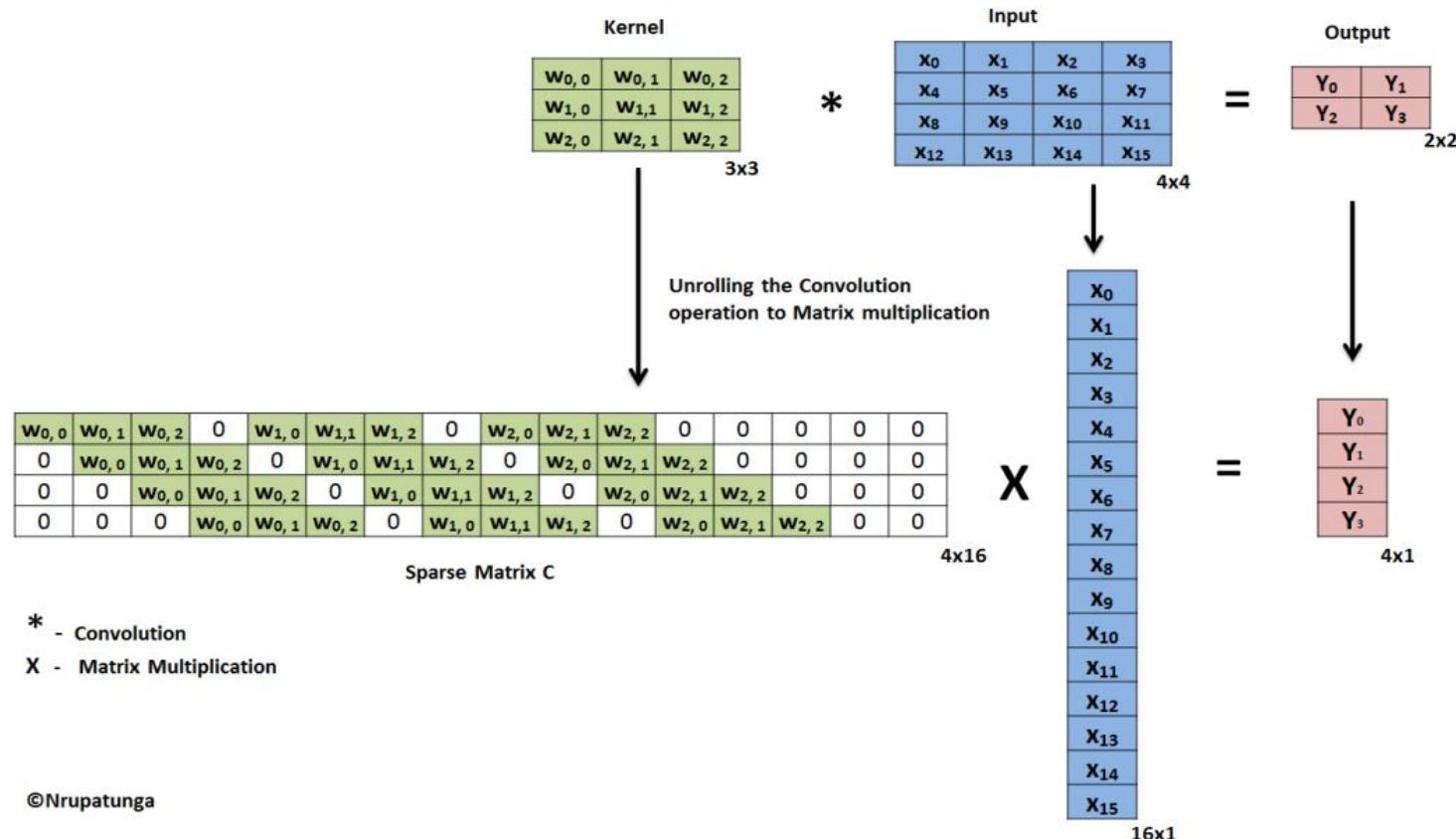
# Convolution Layers

---



```
Conv2D(1, kernel_size=3, strides=(2, 2), padding='valid',  
       input_shape=(7,7,1))  
(image_size-(kernel_size-((kernel_size-1)/2))/strides
```

# Convolution Layers



<https://nrupatunga.github.io/convolution-2/>

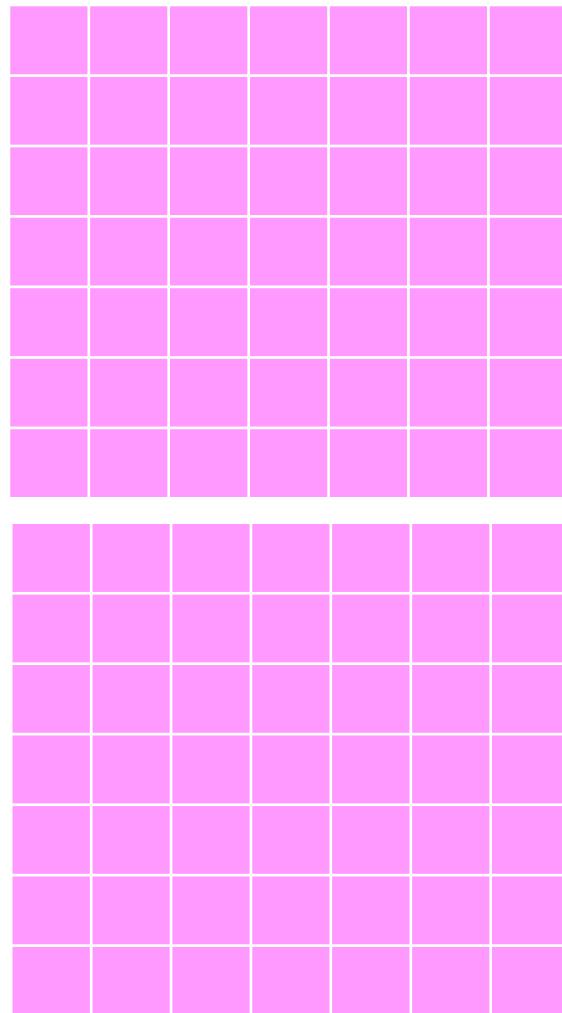
# Exercise

---

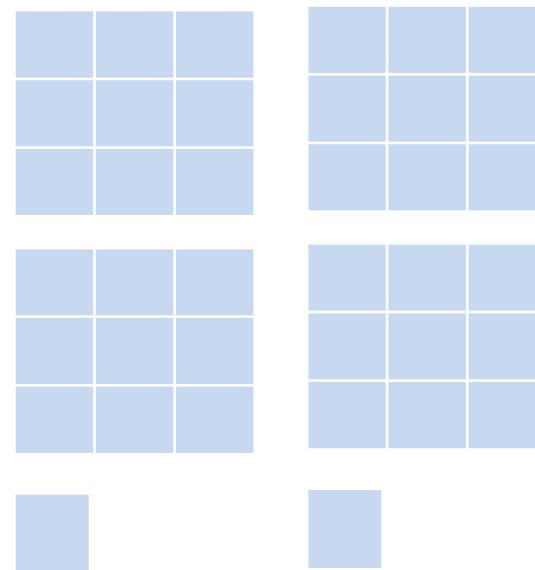
$$\begin{matrix} & \bullet & \\ \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} & \times & \begin{matrix} & & \\ & & \\ & & \\ & & \\ & & \end{matrix} & = & \begin{matrix} & & \\ & & \\ & & \\ & & \\ & & \end{matrix} \end{matrix}$$

# Exercise

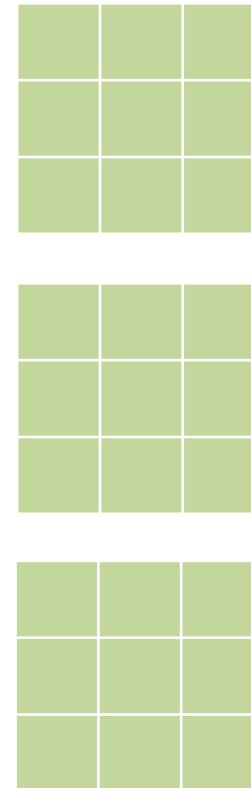
---



•

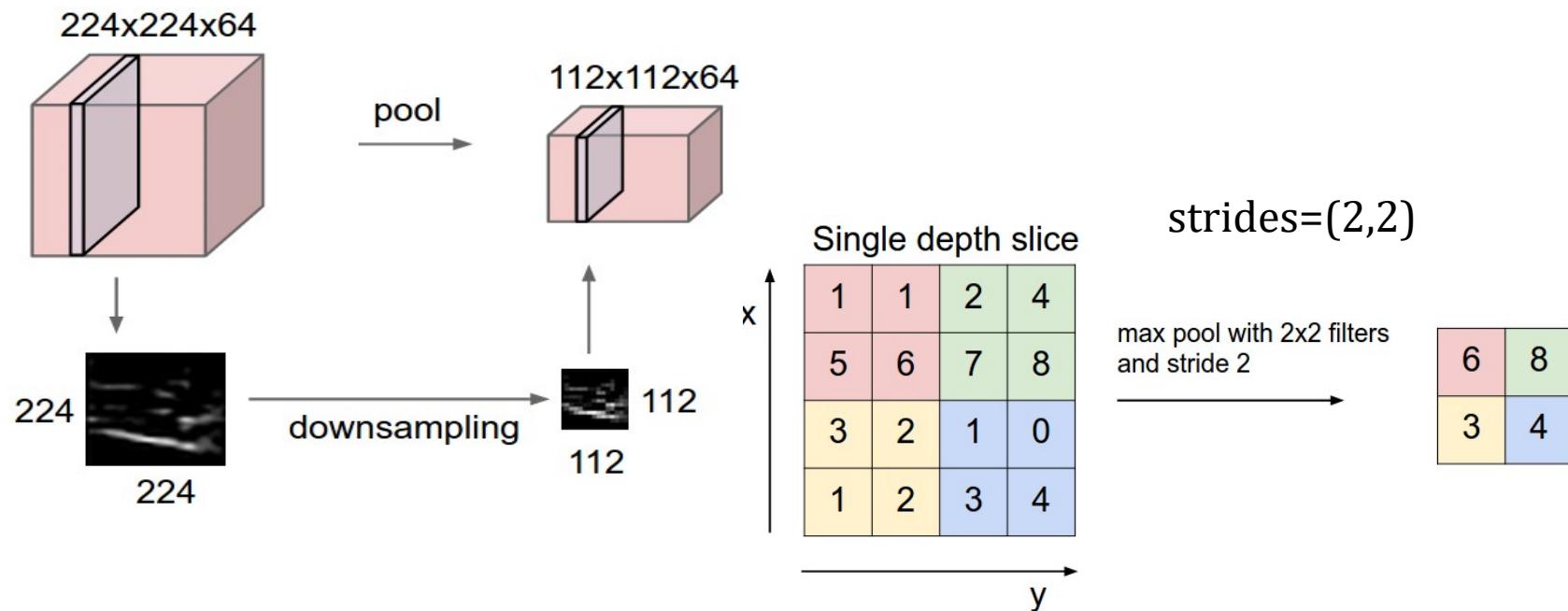


=



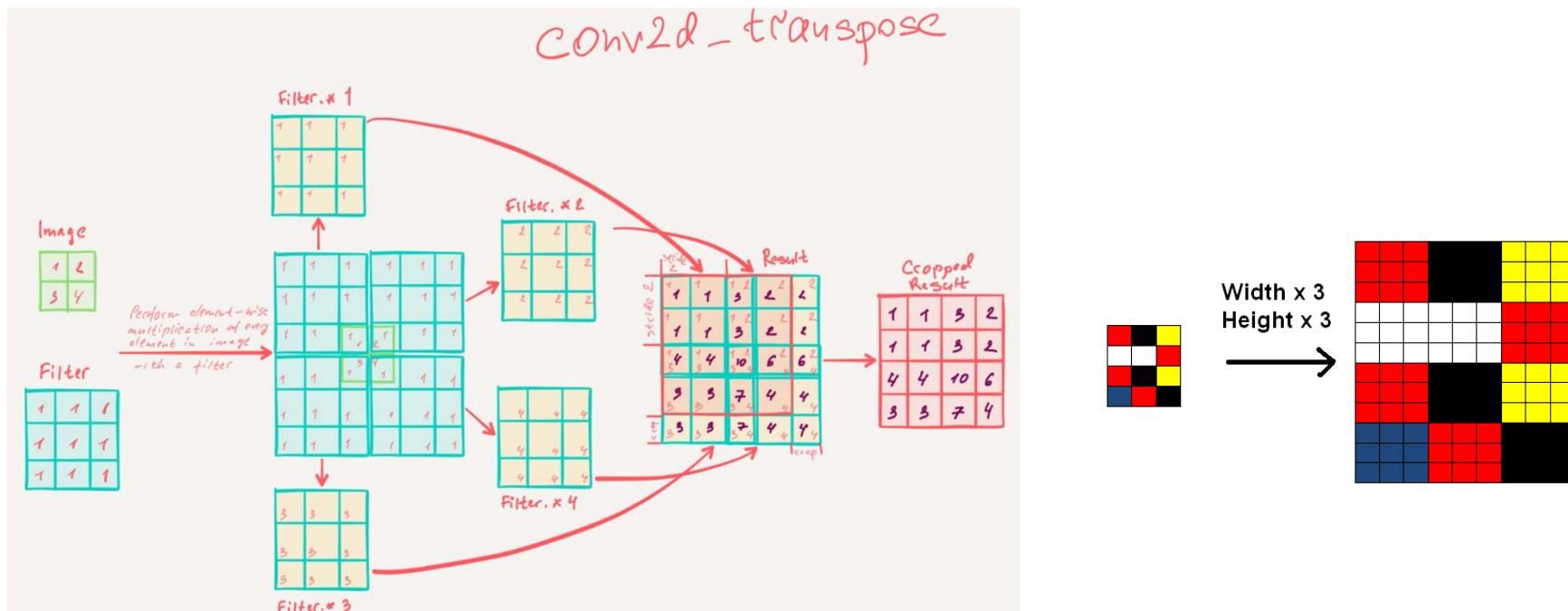
# Max Pooling

- Max Pooling
- Convolution
- Downsampling



# Conv2DTranspose

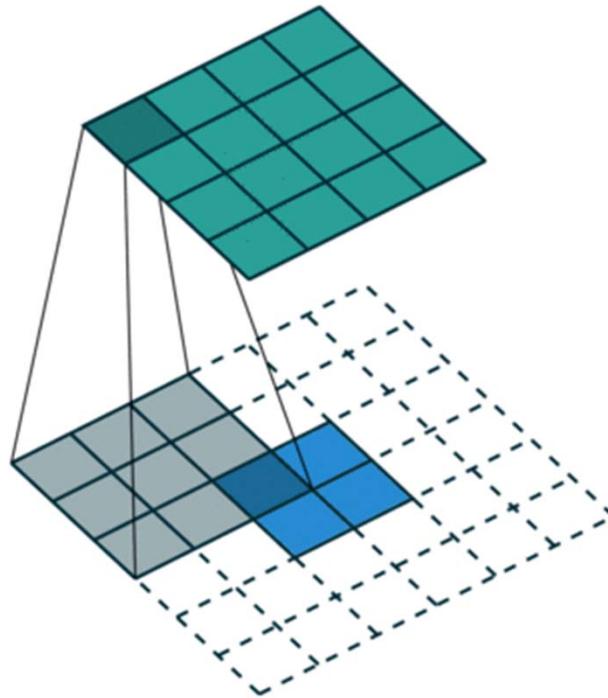
- Multiply + Add



<https://medium.com/@vaibhavshukla182/why-do-we-need-conv2d-transpose-2534cd2a4d98>

# Conv2DTranspose

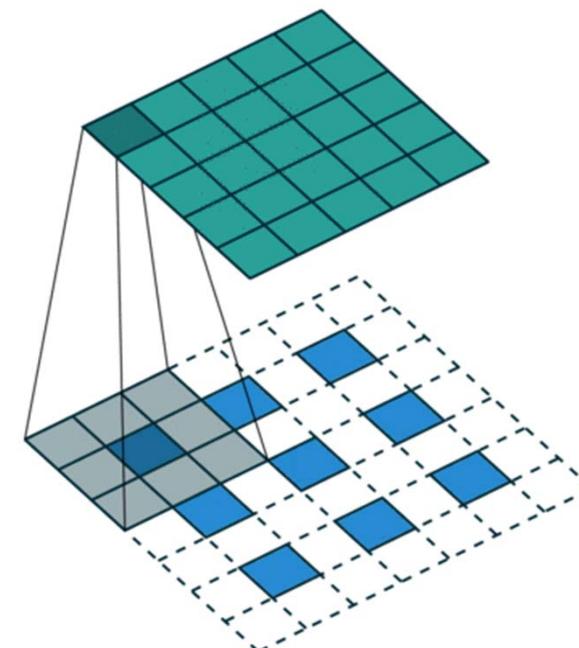
---



stride one and no padding

[https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)

<https://nrupatunga.github.io/convolution-2/>



stride two and padding

# Conv2DTranspose

$$\begin{matrix} \text{Sparse Matrix } C^T \\ \begin{array}{|c|c|c|c|} \hline w_{0,0} & 0 & 0 & 0 \\ \hline w_{0,1} & w_{0,0} & 0 & 0 \\ \hline w_{0,2} & w_{0,1} & w_{0,0} & 0 \\ \hline 0 & w_{0,2} & w_{0,1} & w_{0,0} \\ \hline w_{1,0} & 0 & w_{0,2} & w_{0,1} \\ \hline w_{1,1} & w_{1,0} & 0 & w_{0,2} \\ \hline w_{1,2} & w_{1,1} & w_{1,0} & 0 \\ \hline 0 & w_{1,2} & w_{1,1} & w_{1,0} \\ \hline w_{2,0} & 0 & w_{1,2} & w_{1,1} \\ \hline w_{2,1} & w_{2,0} & 0 & w_{1,2} \\ \hline w_{2,2} & w_{2,1} & w_{2,0} & 0 \\ \hline 0 & w_{2,2} & w_{2,1} & w_{2,0} \\ \hline 0 & 0 & w_{2,2} & w_{2,1} \\ \hline 0 & 0 & 0 & w_{2,2} \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \end{matrix} \quad 16 \times 4$$

$\mathbf{X}$       =       $\begin{matrix} \mathbf{Y}_0 \\ \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \mathbf{Y}_3 \end{matrix}$        $4 \times 1$

$$\begin{matrix} \mathbf{X} \\ \begin{array}{|c|} \hline x_0 \\ \hline x_1 \\ \hline x_2 \\ \hline x_3 \\ \hline x_4 \\ \hline x_5 \\ \hline x_6 \\ \hline x_7 \\ \hline x_8 \\ \hline x_9 \\ \hline x_{10} \\ \hline x_{11} \\ \hline x_{12} \\ \hline x_{13} \\ \hline x_{14} \\ \hline x_{15} \\ \hline \end{array} \end{matrix} \quad 16 \times 1$$

©Nrupatunga

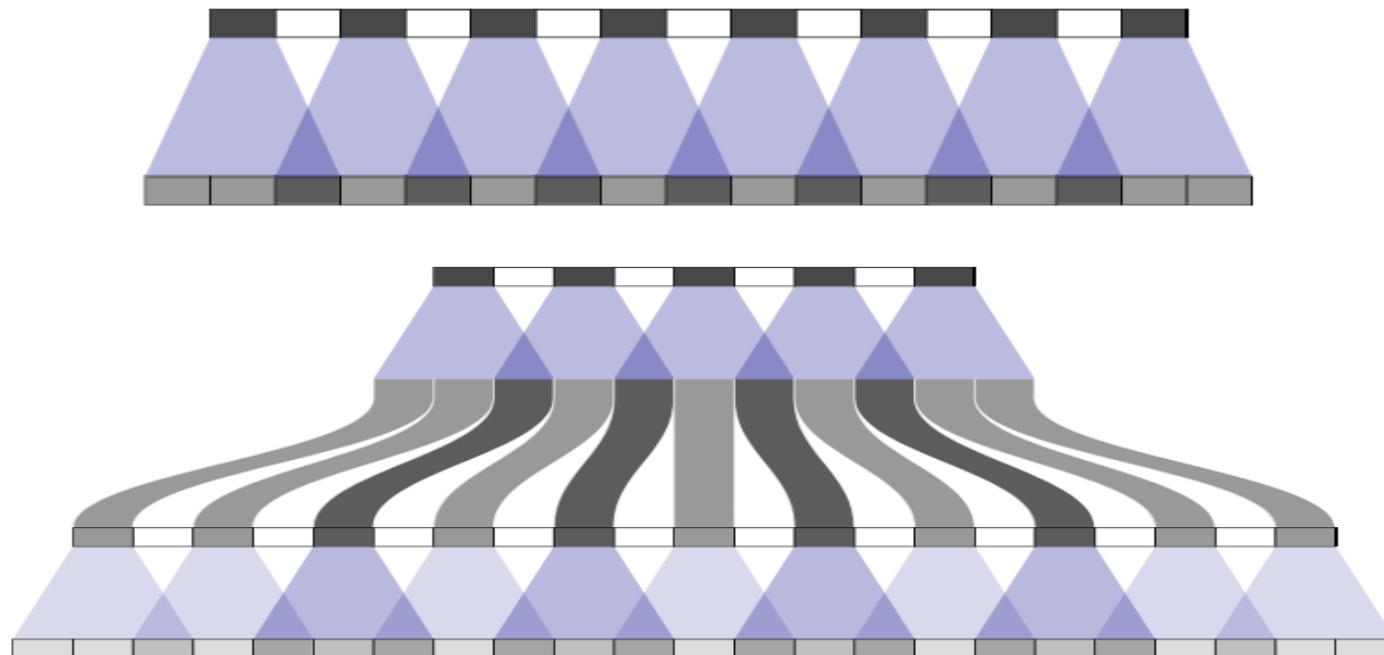
<https://nrupatunga.github.io/convolution-2/>

# Upsampling + Convolution

---

$m = \text{Upsampling2D}(2)(m)$

$m = \text{Conv2D}(16, 3, \text{padding}=\text{'same'})(m)$



<https://distill.pub/2016/deconv-checkerboard/>

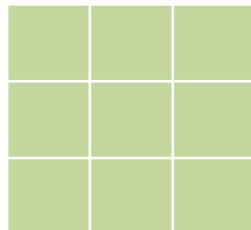
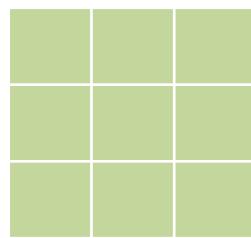
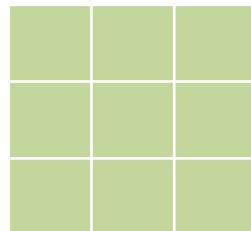
# Exercise

---

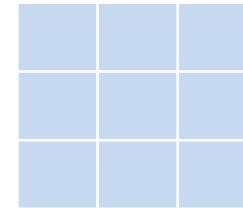
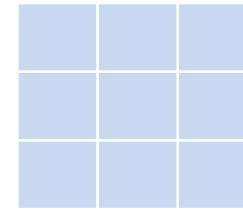
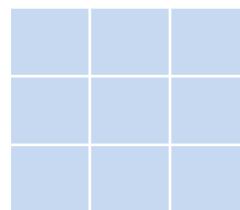
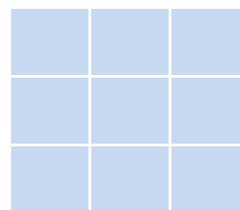
$$\begin{matrix} \text{green} & \otimes & \begin{matrix} \text{blue} \\ \text{blue} \end{matrix} & = & \begin{matrix} \text{pink} & \text{pink} & \text{pink} & \text{pink} & \text{pink} \\ \text{pink} & \text{pink} & \text{pink} & \text{pink} & \text{pink} \\ \text{pink} & \text{pink} & \text{pink} & \text{pink} & \text{pink} \\ \text{pink} & \text{pink} & \text{pink} & \text{pink} & \text{pink} \\ \text{pink} & \text{pink} & \text{pink} & \text{pink} & \text{pink} \end{matrix} \end{matrix}$$

# Exercise

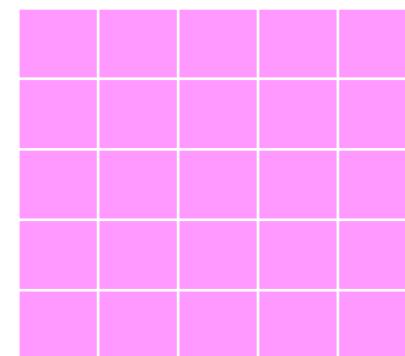
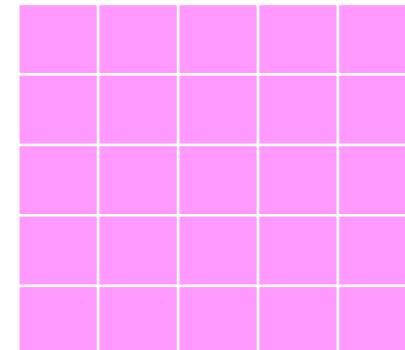
---



•



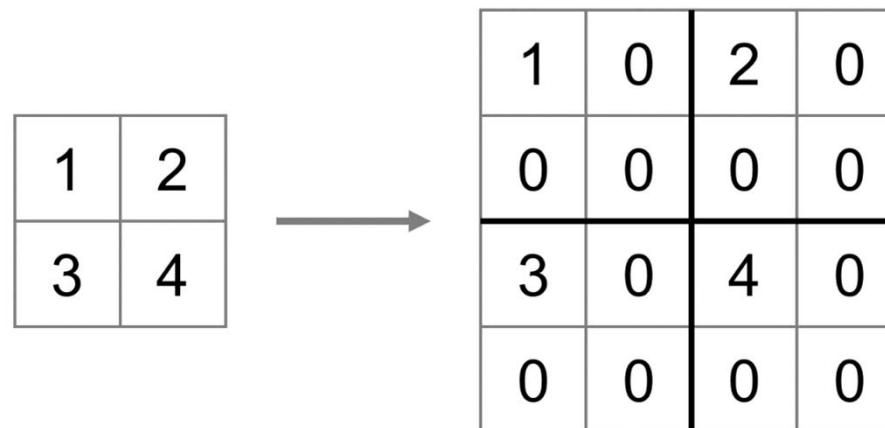
=



# Upsampling

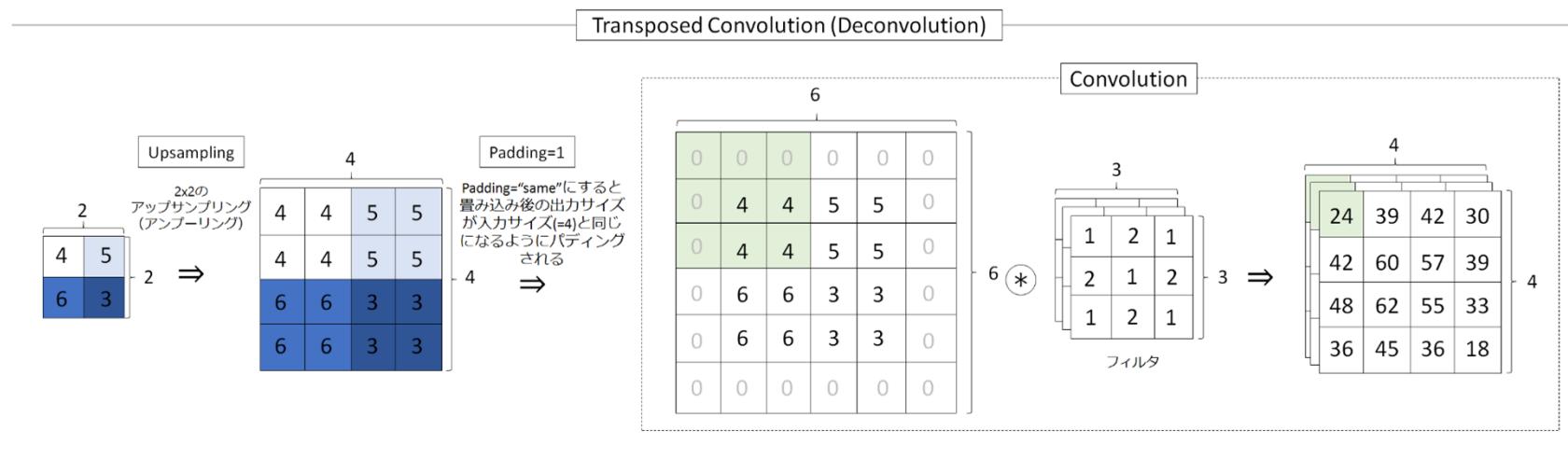
---

- **Un(max)pooling**

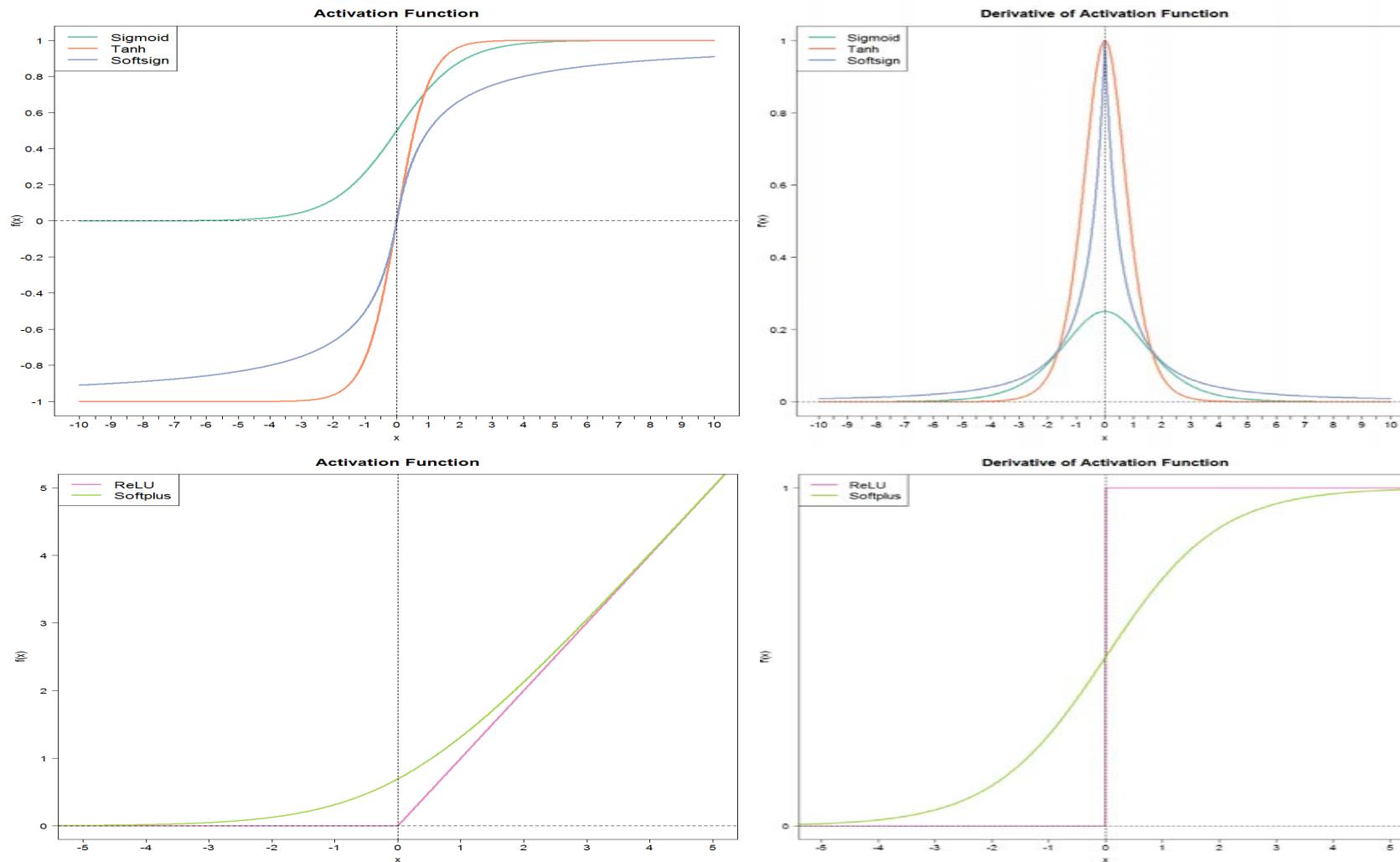


<https://www.oreilly.com/library/view/deep-learning-for/9781788295628/467cf02b-dc52-49c5-9289-b2721f6758da.xhtml>

# Conv2DTranspose



# Activation Function



# Demo



Ulead  
PhotoImpact 10

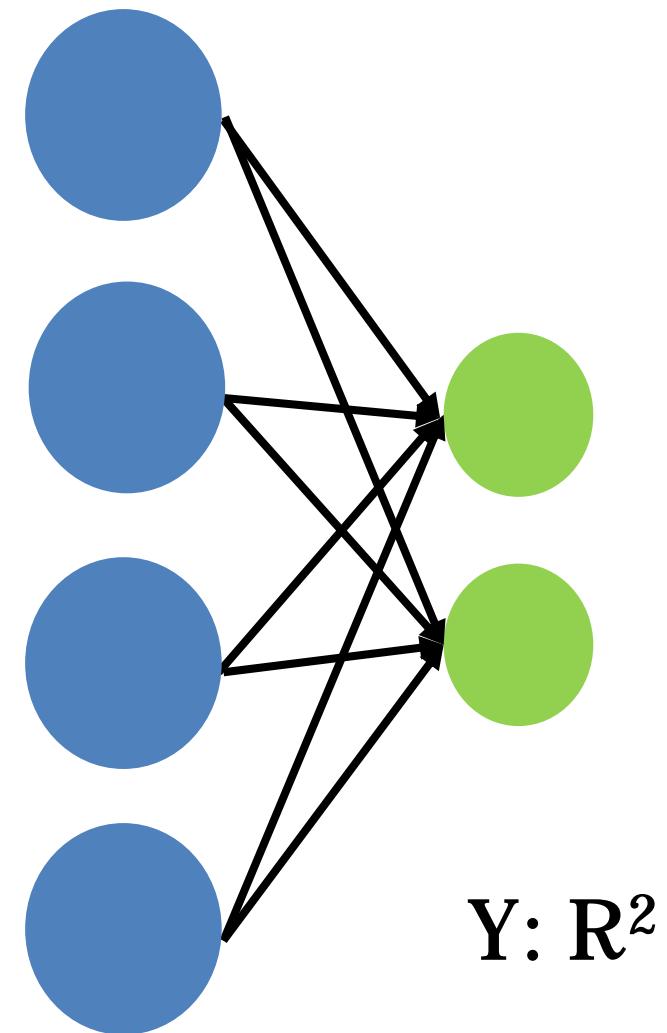
# Feature Map

---

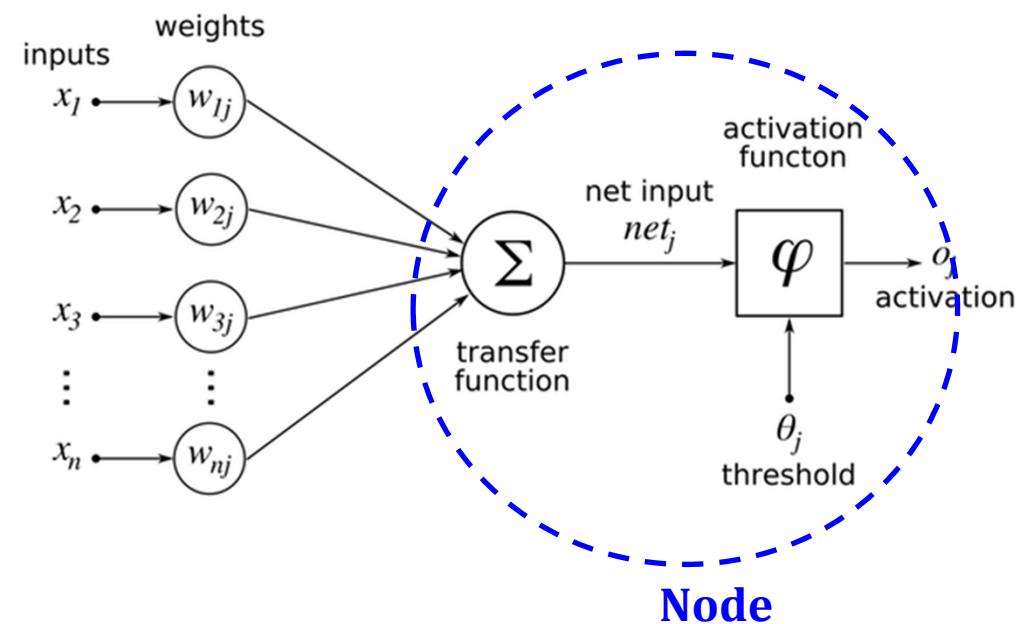
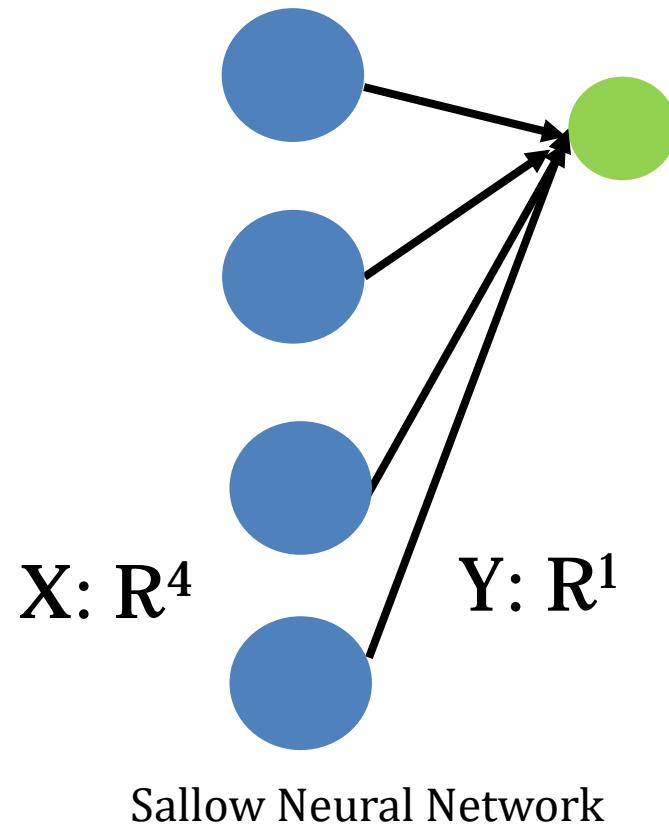
- Matrix
- Mapping
- Nonlinear
- Connected
- Dropout
- Normalization
- ...

$$Y = AX$$

$X: \mathbb{R}^4$

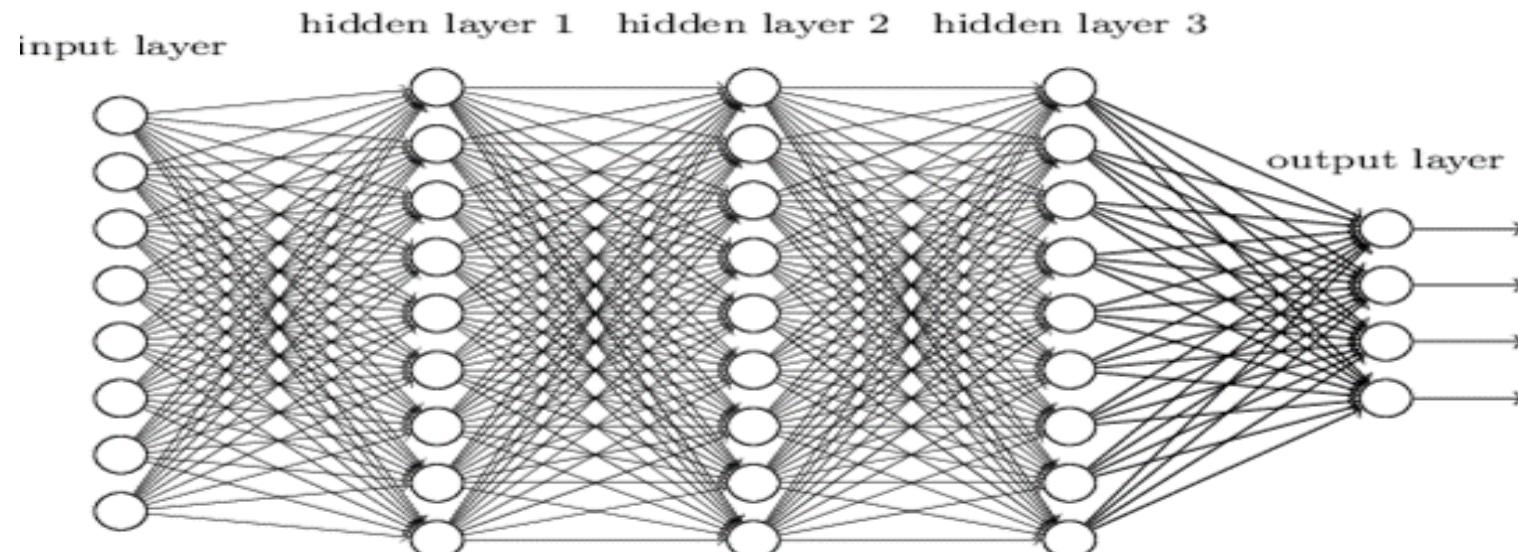


# Feature Map



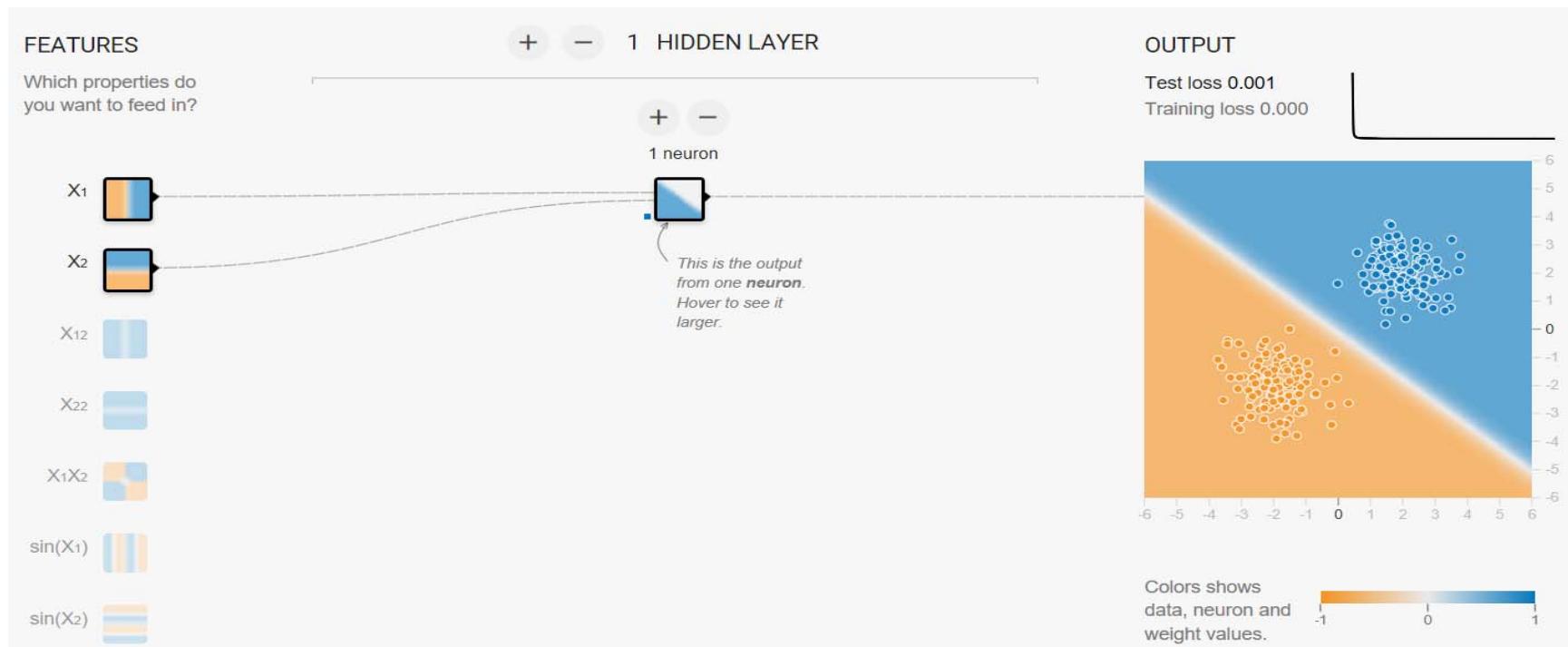
# Deep Feature Map

- 深度學習模型 (Model)



# Neural Network

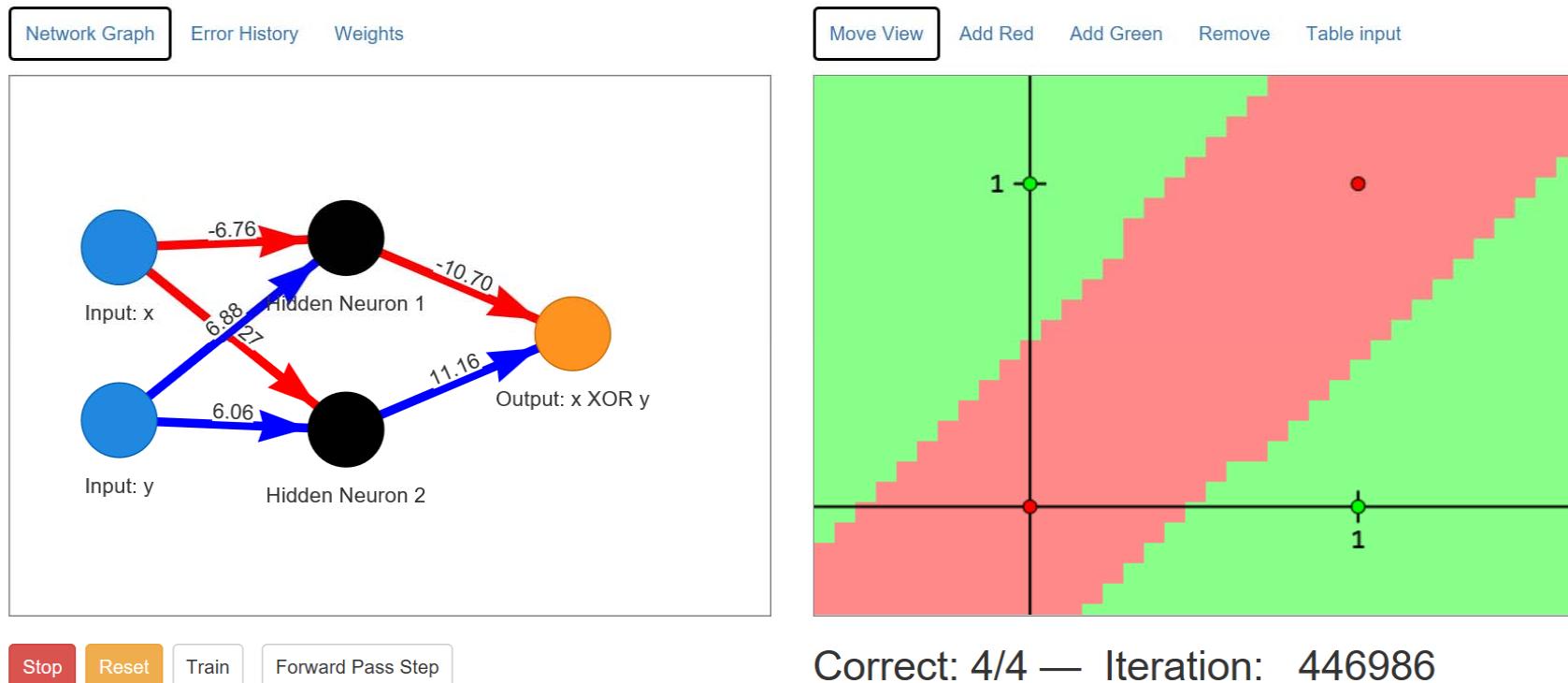
## • Linearly Problem



<https://playground.tensorflow.org/>

# Neural Network demo

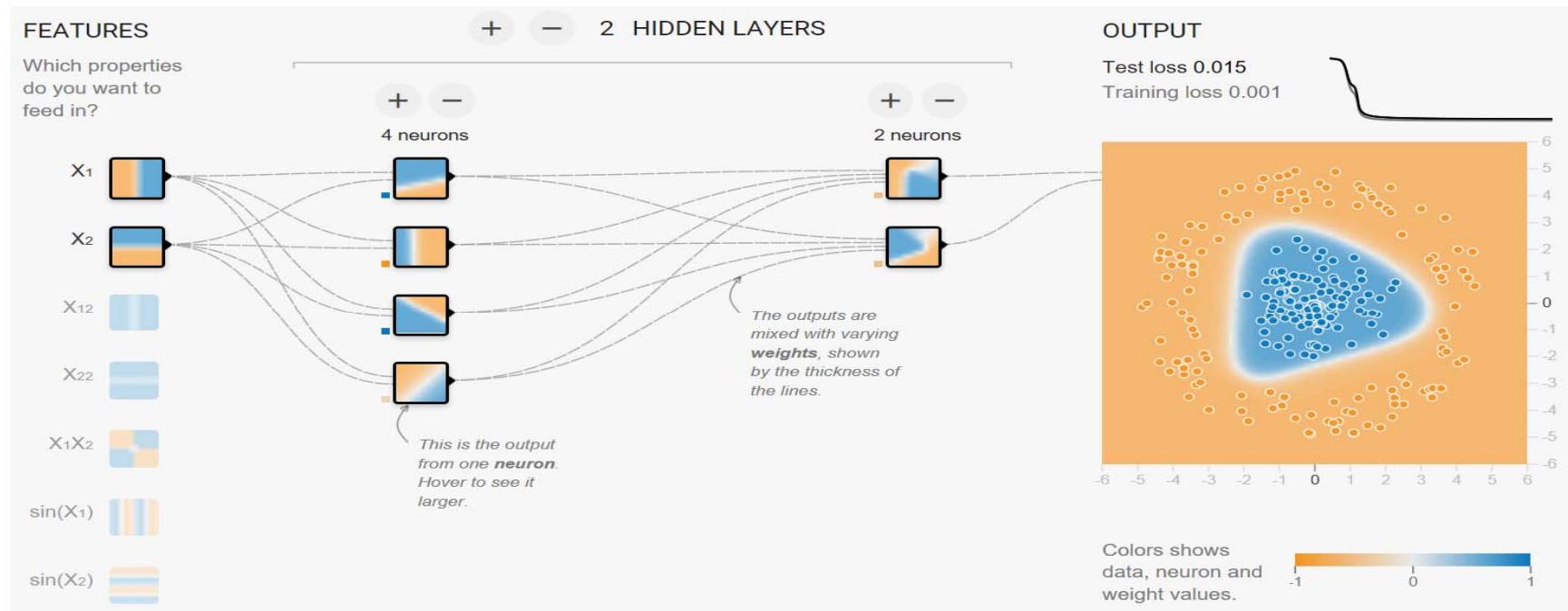
- Nonlinear model



<https://lecture-demo.ira.uka.de/neural-network-demo/>

# Neural Network

- Nonlinear model



# Keras.Layer

---

## **Regular dense layer:**

- keras.layers.core.Dense()

## **Recurrent neural network layer:**

- keras.layers.recurrent.Recurrent()
- keras.layers.recurrent.SimpleRNN()
- keras.layers.recurrent.GRU()
- keras.layers.recurrent.LSTM()

# Keras.Layer

---

## **Convolutional and Pooling layers:**

- keras.layers.convolutional.Conv1D()
- keras.layers.convolutional.Conv2D()
- keras.layers.pooling.MaxPooling1D()
- keras.layers.pooling.MaxPooling2D()

## **Regularization layer:**

- keras.layers.core.Dropout()

# Keras.Layer

---

## Regulization parameter:

- kernel\_regularizer (weight matrix)
- bias\_regularizer (bias vector)
- activity\_regularizer (output of activation)

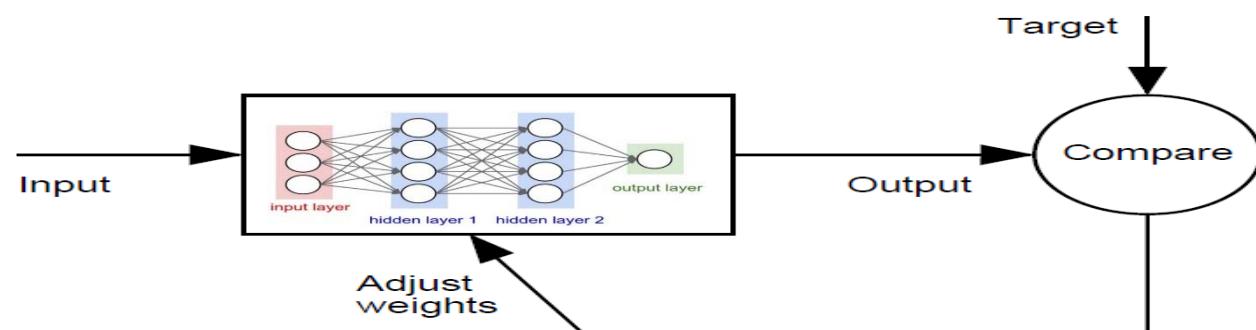
## Performance layer:

- keras.layers.normalization.BatchNormalization()

# Parametrs Setting

---

- **Metrics**
  - <https://keras.io/metrics/>
- **Loss Function**
  - <https://keras.io/losses/>
- **Optimizers**
  - <https://keras.io/optimizers/>



# Compilation

- for a multi-class classification problem  
model.compile(optimizer='rmsprop',  
              loss='categorical\_crossentropy', metrics=['accuracy'])
- for a binary classification problem  
model.compile(optimizer='rmsprop', loss='binary\_crossentropy',  
              metrics=['accuracy'])
- for a mean squared error regression problem  
model.compile(optimizer='rmsprop', loss='mse')
- for custom metrics  
import keras.backend as K  

```
def mean_pred(y_true, y_pred):  
    return K.mean(y_pred)
```

model.compile(optimizer='rmsprop', loss='binary\_crossentropy',  
              metrics=['accuracy', mean\_pred])



# API

---

- Loading and saving models and weights
- Early stopping
- History saving
- Checkpointing

# Save Model & Weight

---

- Save Model

```
from keras.models import load_model  
model.save('my_model.h5')  
del model  
model = load_model('my_model.h5')
```

- Save weight

```
keras.layer.get_weights()  
model.save_weights('my_model_weights.h5')  
model.load_weights('my_model_weights.h5')
```

# Model for keras

---

```
model = Sequential()  
model.add(Conv2D())  
model.add(Activation())  
model.add(Flatten())  
model.add(Dense())  
model.add(Dropout())  
model.compile()  
model.fit(X,Y)  
model.evaluate(X,Y): loss values & metrics values  
model.predict(X)
```

# Composing models in Keras

---

- Sequential composition
  - Keras Model Lift-Cycle
- Functional composition
  - Keras Functional Models



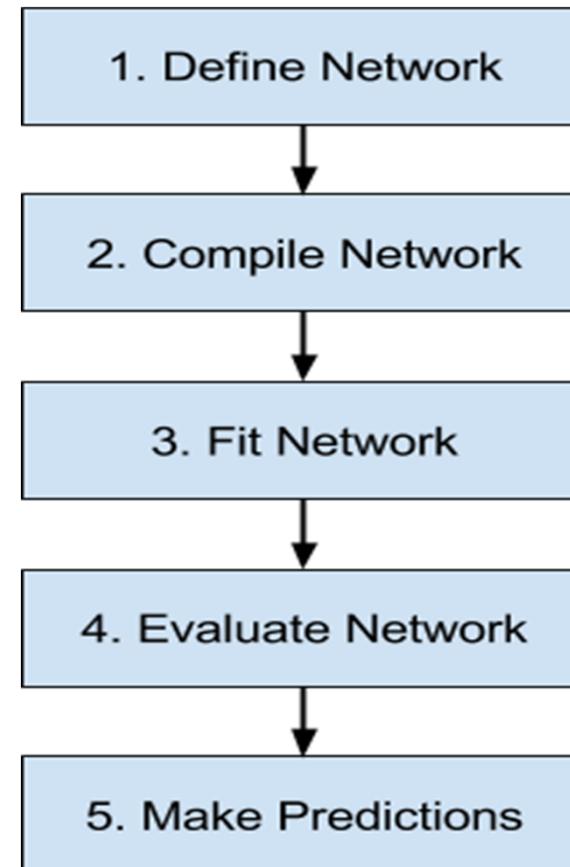
# 學習模型問題

---

# Keras Model Life-Cycle

---

- Define Network
- Compile Network
- Fit Network
- Evaluate Network
- Make Predictions



# Model Structure

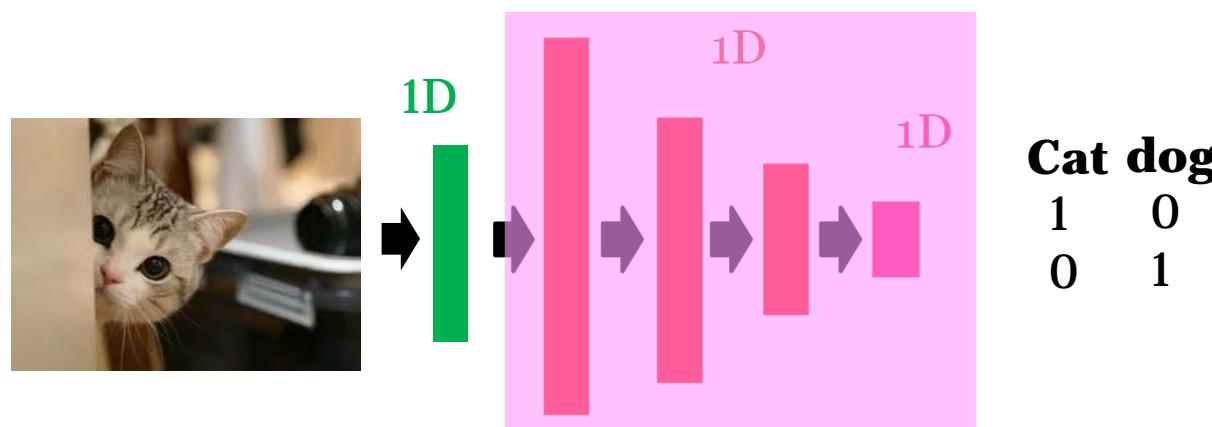
---

- Deep Neural Network
  - Text
  - Numerical →
  - Images
- Neural Networks  
(Deep Learning)
- Text
  - Numerical
  - Images

# Model = Function

---

- reshape
- mapping (fully connected)

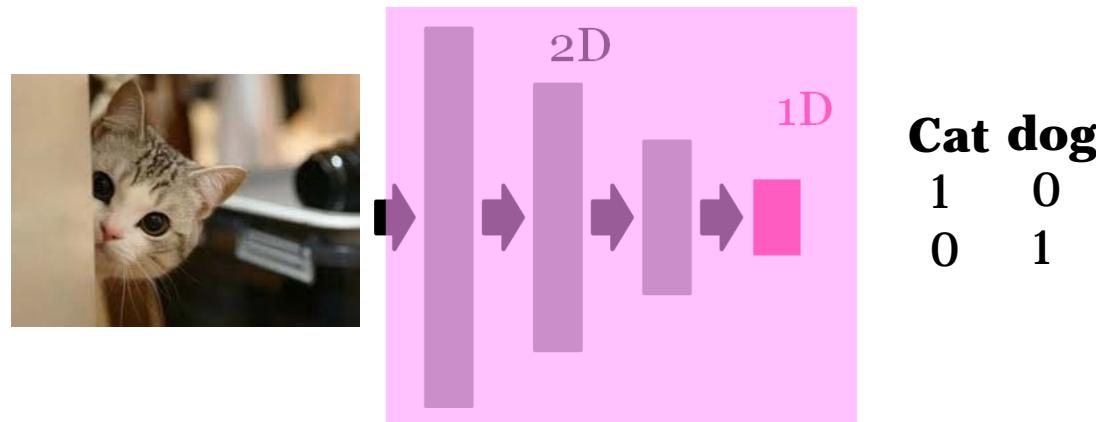


**A critical Neural Network (CNN): Classification/regression**

# Model = Function

---

- reshape
- feature extraction (convolution)
- max pooling/downsampling

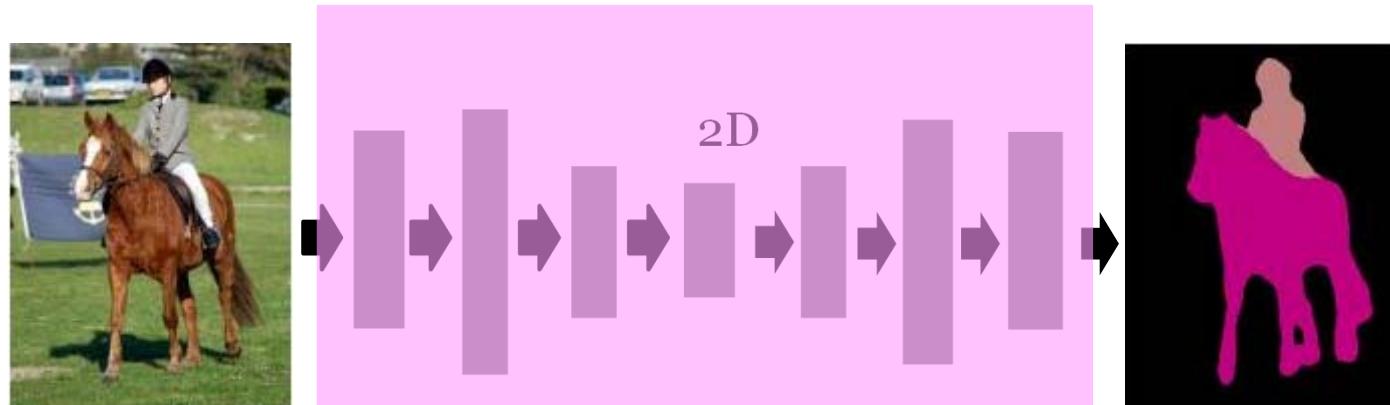


**Convolutional Neural Network (CNN): Classification/regression**

# Model = Function

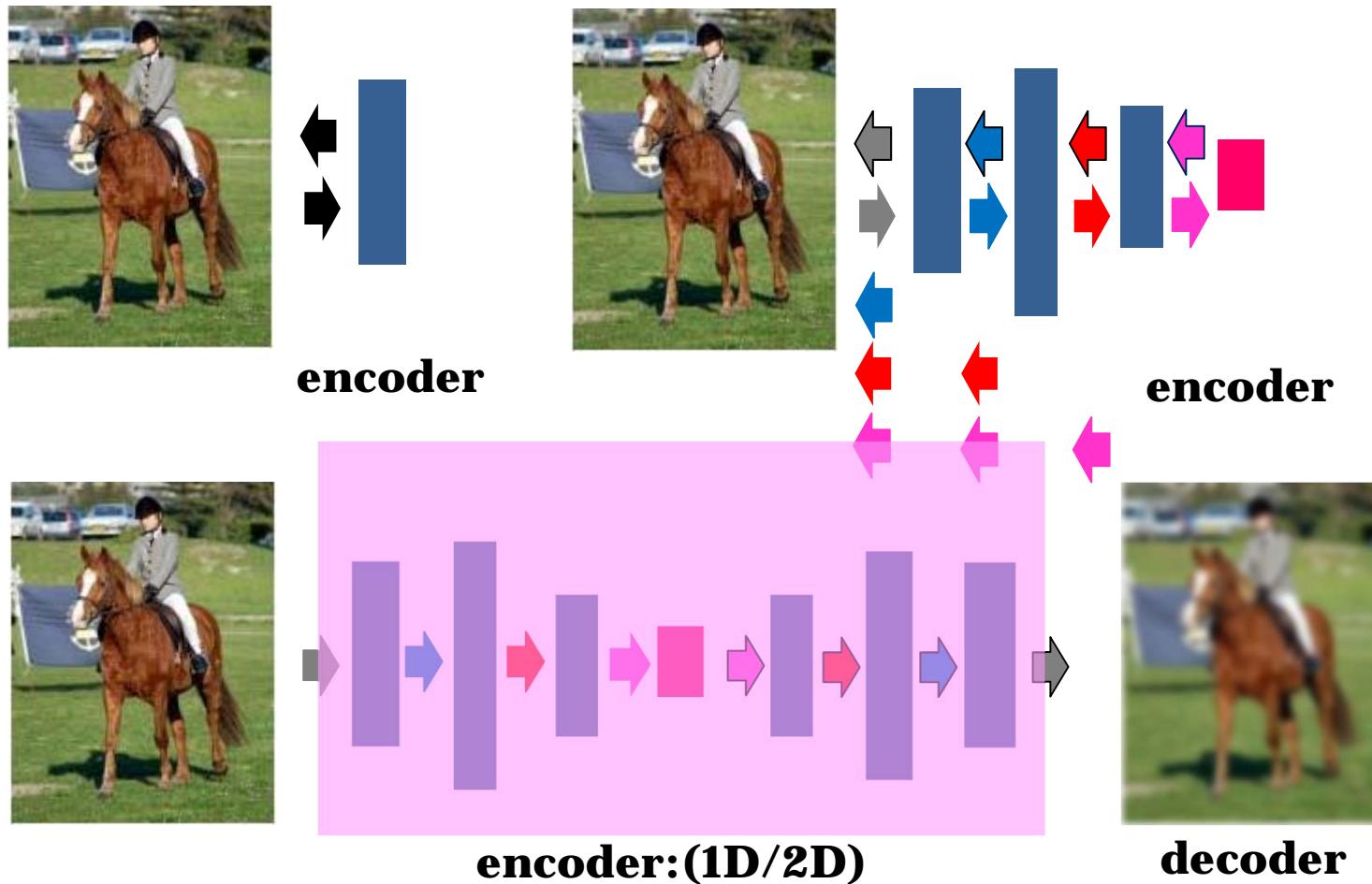
---

- reshape
- feature extraction (convolution)
- max pooling/downsampling/upsampling



**Fully Convolutional Network (FNN): Clustering**

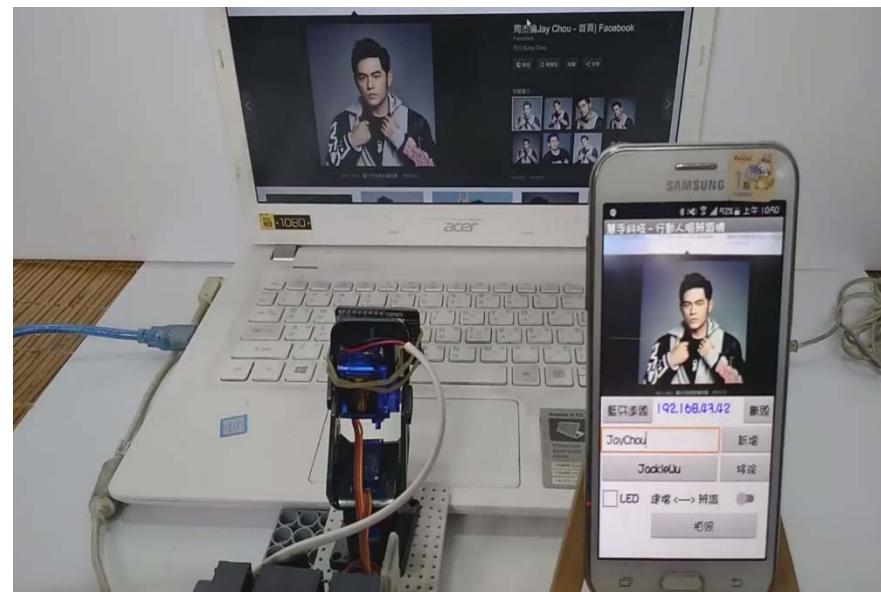
# Model = Function



# DEMO

---

- Edge/Cloud Computing

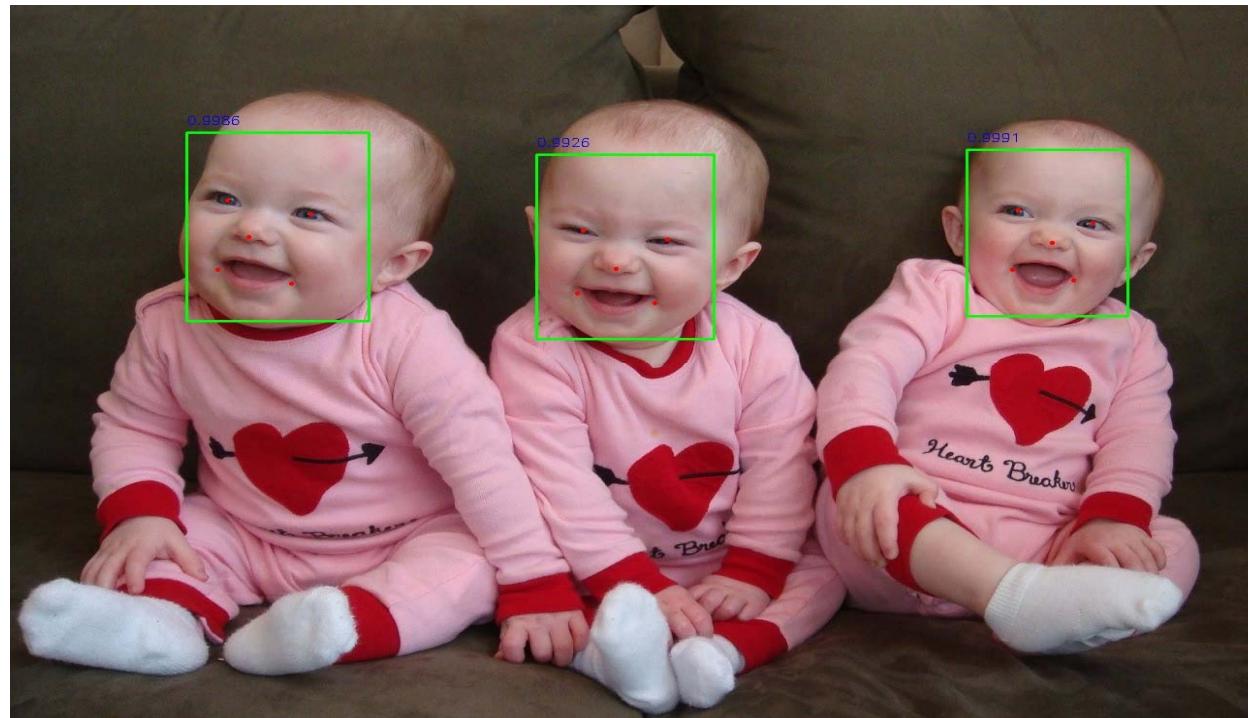


<https://www.facebook.com/MarkeFactory/videos/938065716527342/>  
**義守大學 I-SHOU UNIVERSITY**

# 課後練習

---

- 視覺應用偵測: position? (left or right)



# 補充資料

---

- <http://evexdb.org/pmresources/vec-space-models/>
- <https://github.com/cambridgeltl/BioNLP-2016>
- <http://bioasq.org/news/bioasq-releases-continuous-space-word-vectors-obtained-applying-word2vec-pubmed-abstracts>

# 補充資料

---

- <https://www.kaggle.com/yufengdev/bbc-text-categorization>
- <https://www.kaggle.com/anucool007/multi-class-text-classification-bag-of-words>
- <https://www.kaggle.com/carlosaguayo/deep-learning-for-text-classification>
- <https://www.kaggle.com/shaz13/feature-engineering-for-nlp-classification>
- <https://www.kaggle.com/ngyptr/multi-classification-with-lstm>
- <https://realpython.com/python-keras-text-classification/>

# 專題實作一

---

- Kaggle BBC text classification
- Text Classification using CNN