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

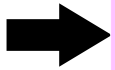
- 開發環境安裝與基礎 Open CV套件
  - VS Code/Pycharm
  - Jupyter Notebook
  - Virtualenv
  - Python
    - Tuple/Set/List /Dictionary/Numpy
    - Package/Module
  - Open CV
    - Read/Save
  - Image Filters

Data Structure

- Numerical/Numpy.array

- Numerical
- Images

(1D/2D/3D)



Neural Networks
(Deep Learning)

- Numerical
- Images

(1D/2D/3D)

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

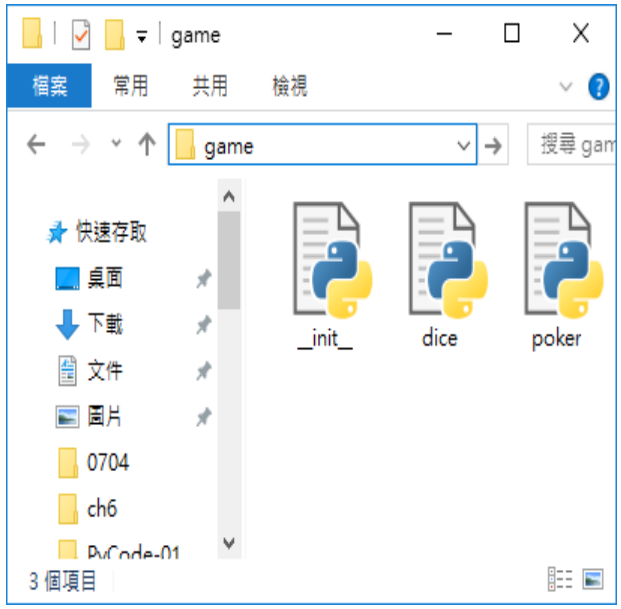
- a=() #tuple
- b=(5) #int
- c=(5,) #tuple
- d=[] #list
- e= {} #dictionary
- e= {1} #set
- e= {a:2} #dictionary

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, ...



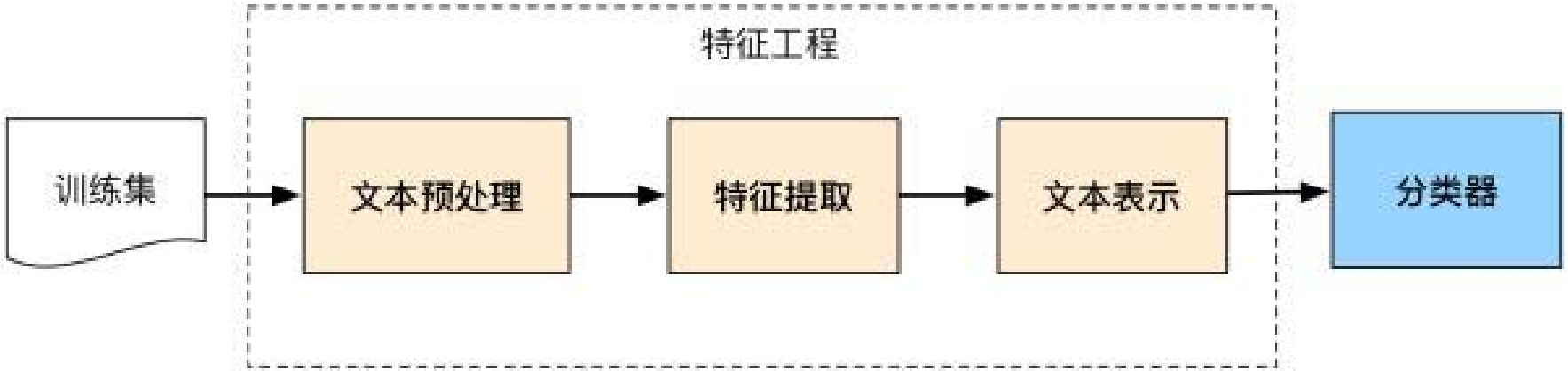
特徵映射與濾波器

特徵映射

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

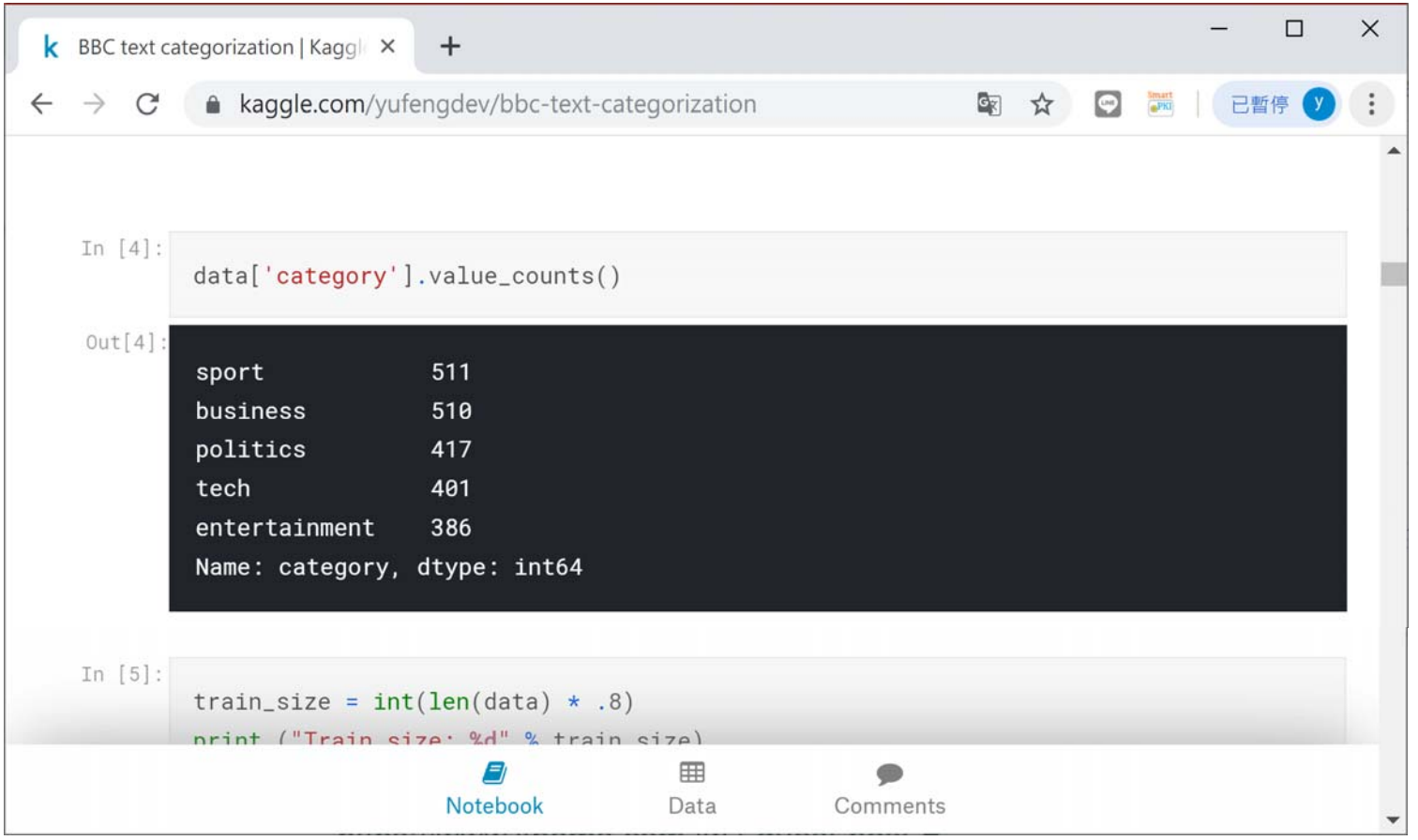
文字模型

- 傳統文本分類方法



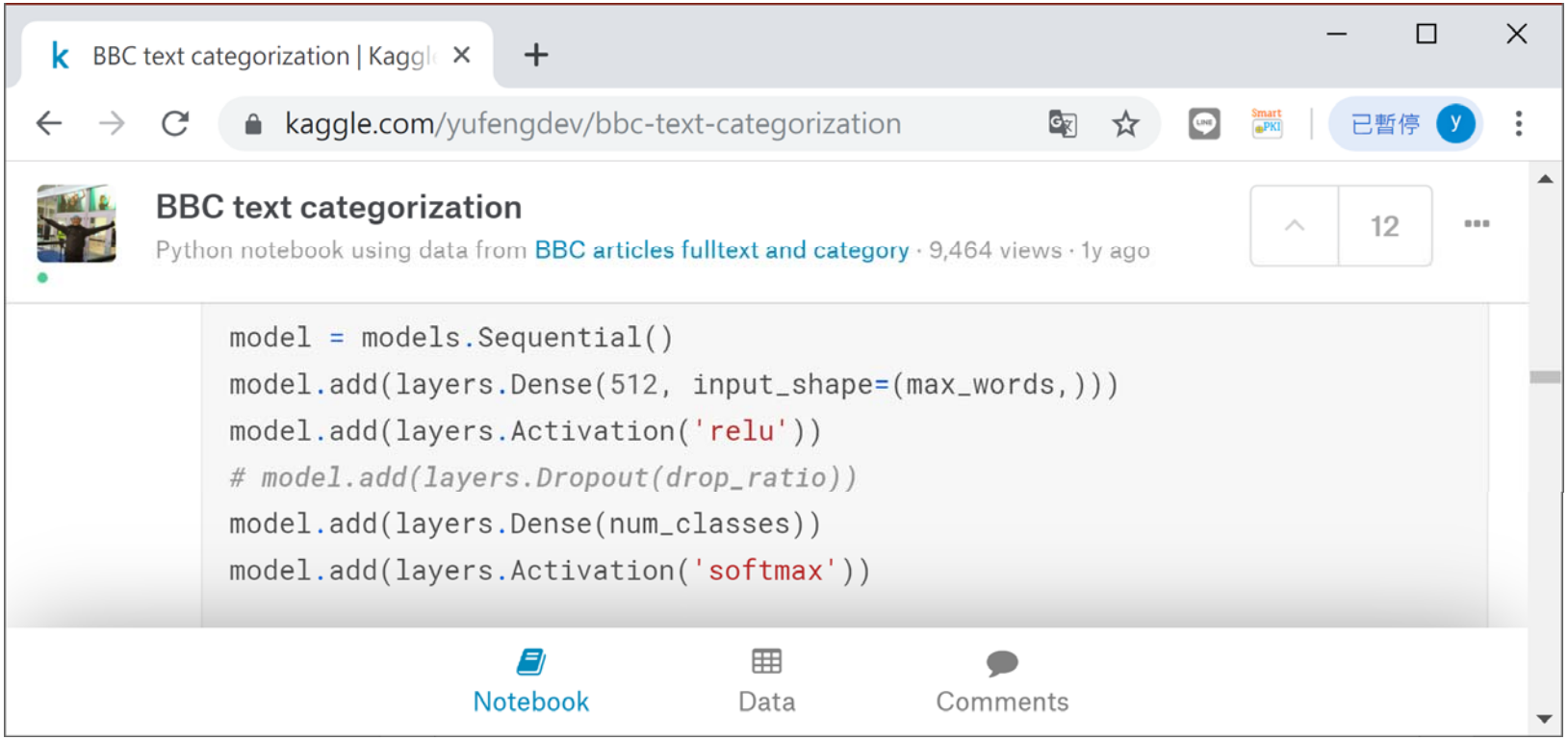
https://zhuanlan.zhihu.com/p/76003775

Kaggle Text Classification



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

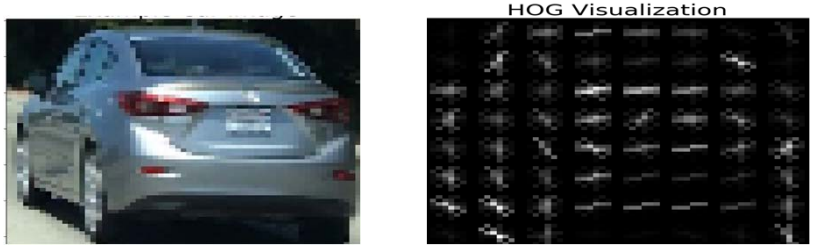
Kaggle Text Classification

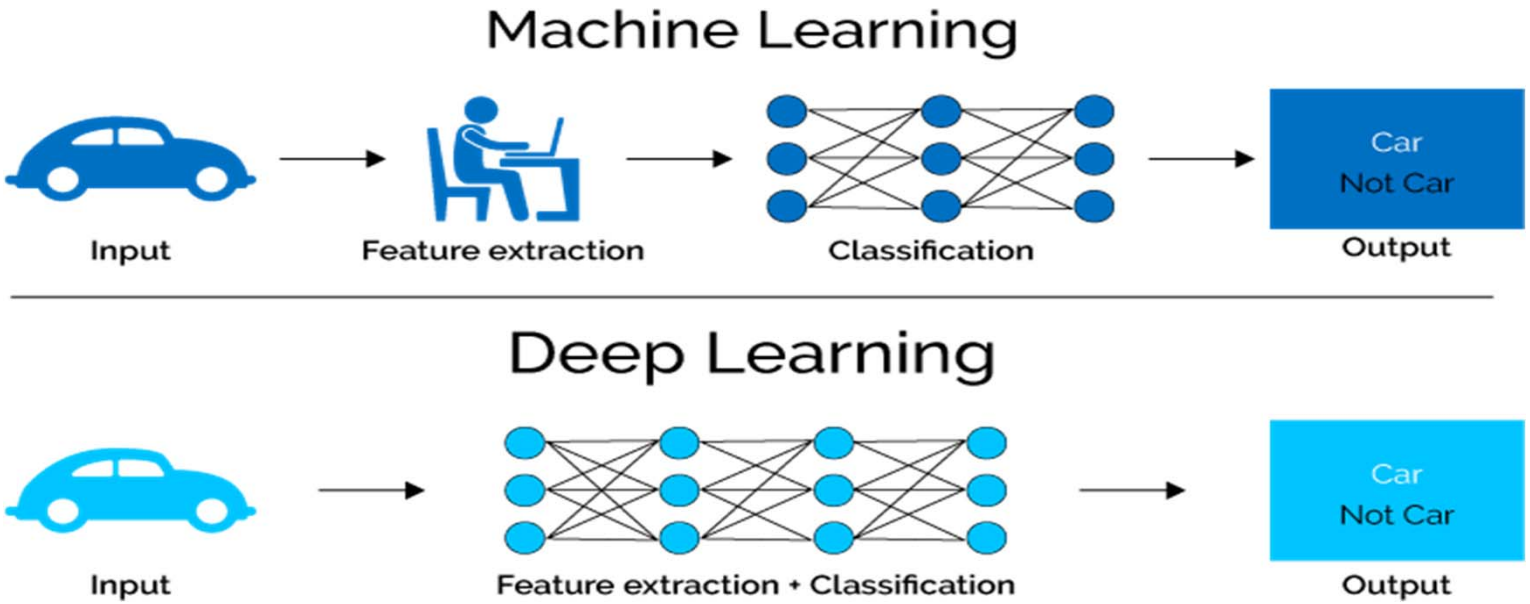


https://www.kaggle.com/yufengdev/bbc-text-categorization

影像處理模型

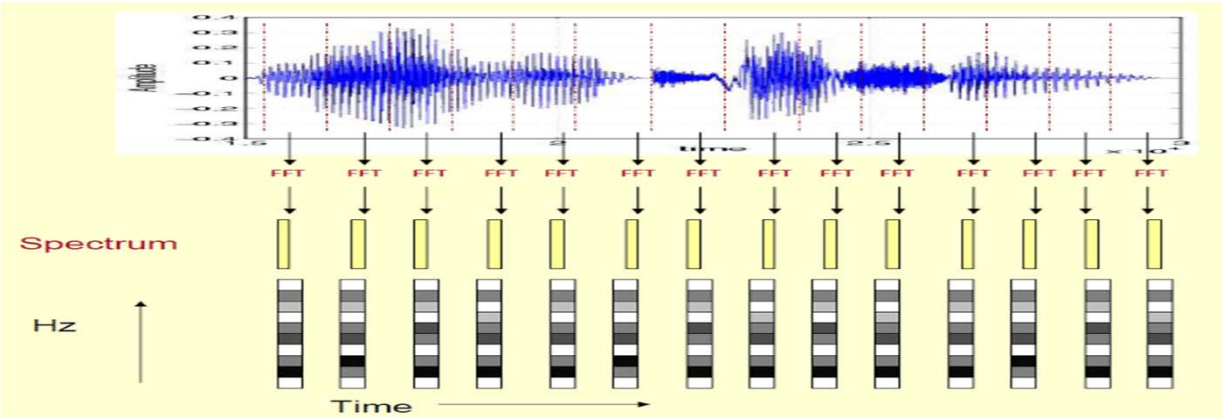
- 傳統影像處理

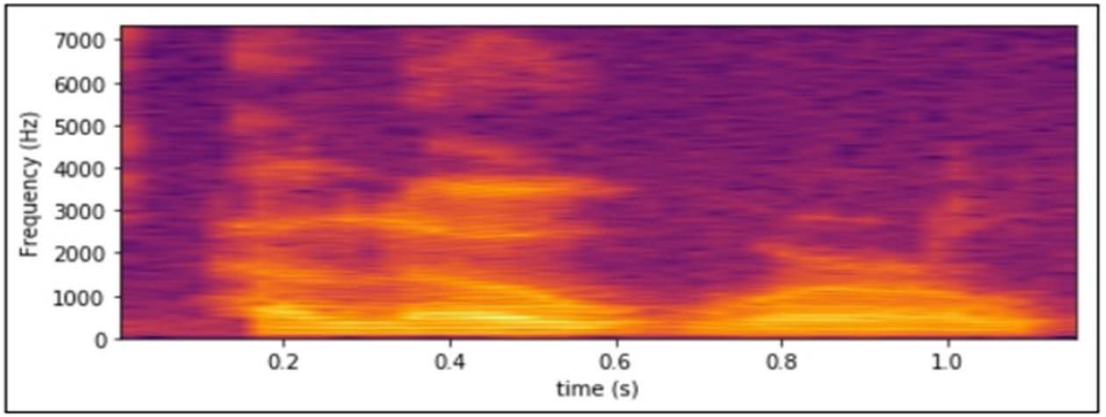




圖片來源：https://semiengineering.com/deep-learning-spreads/

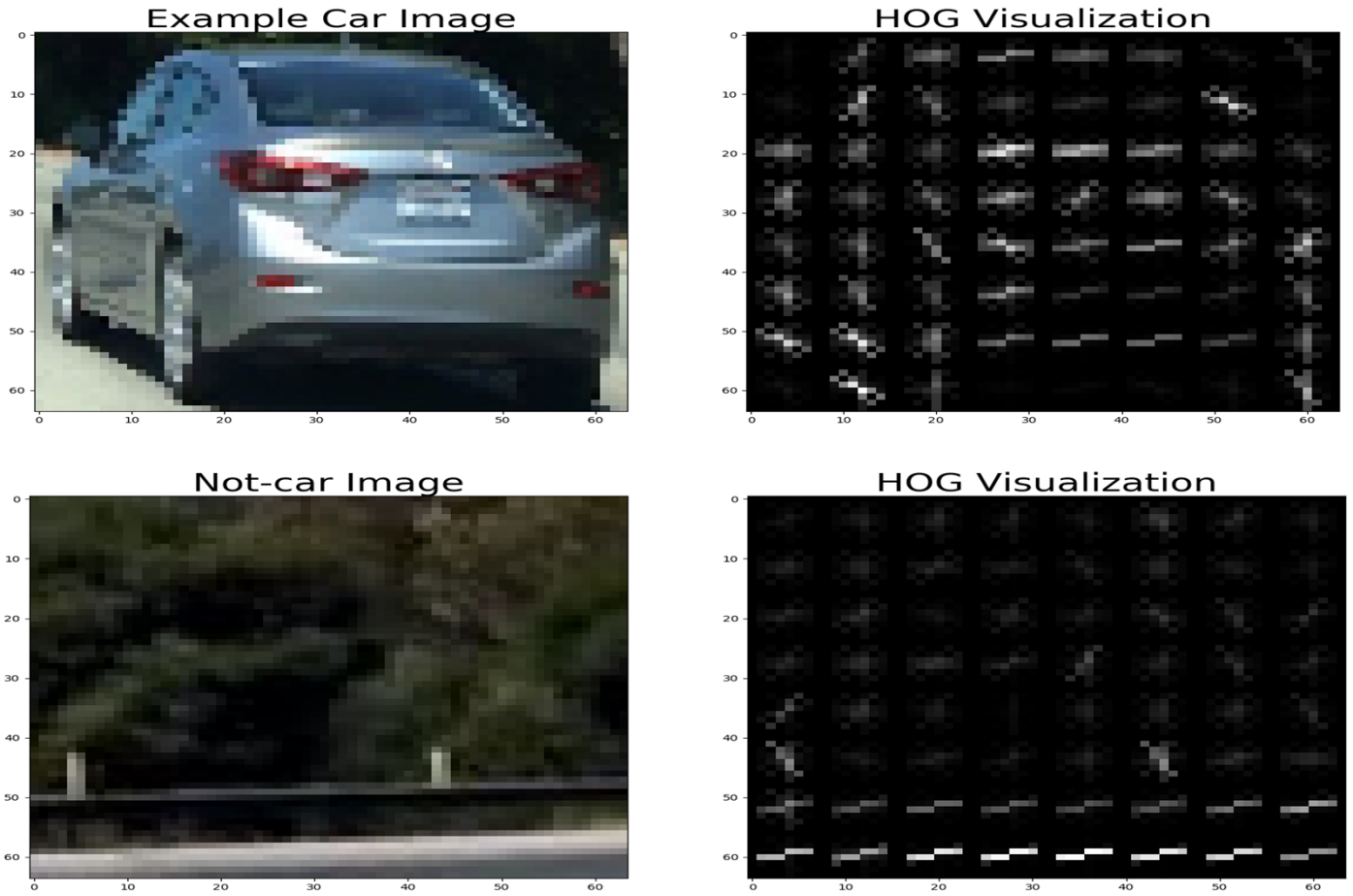
語音處理模型





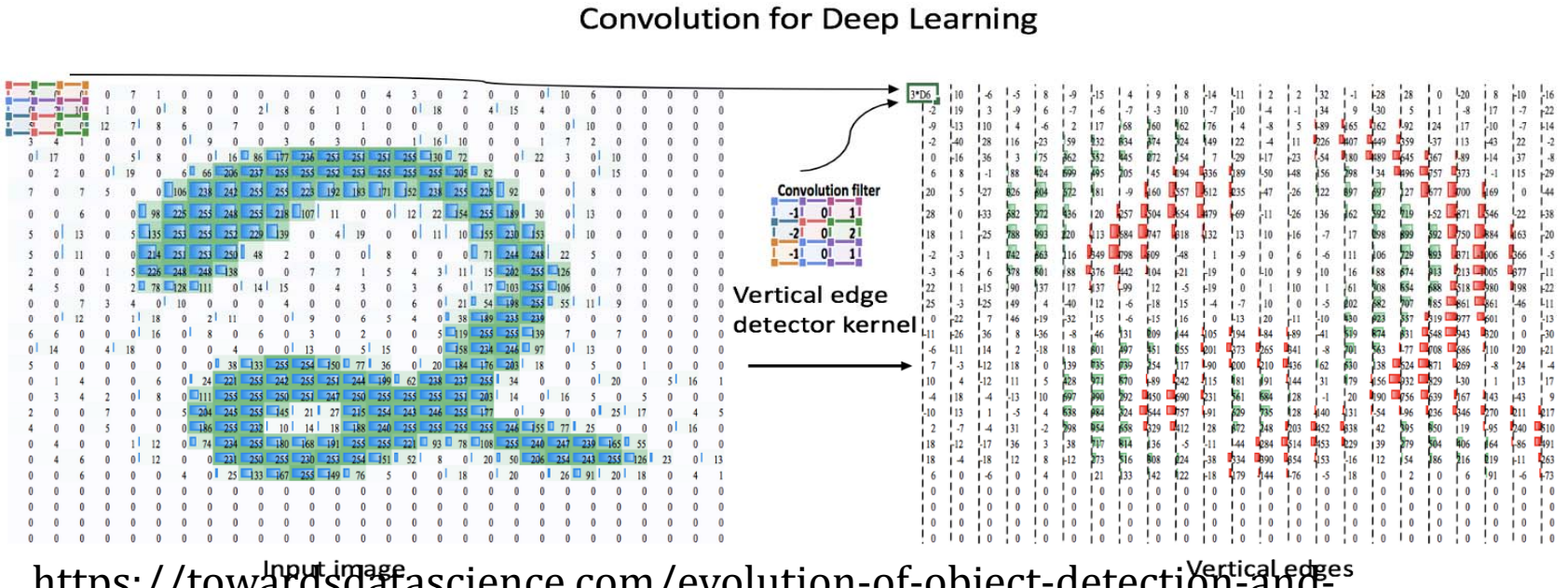
Mel spectrum of speech signal from the previous figure

Feature Extraction



Convolution

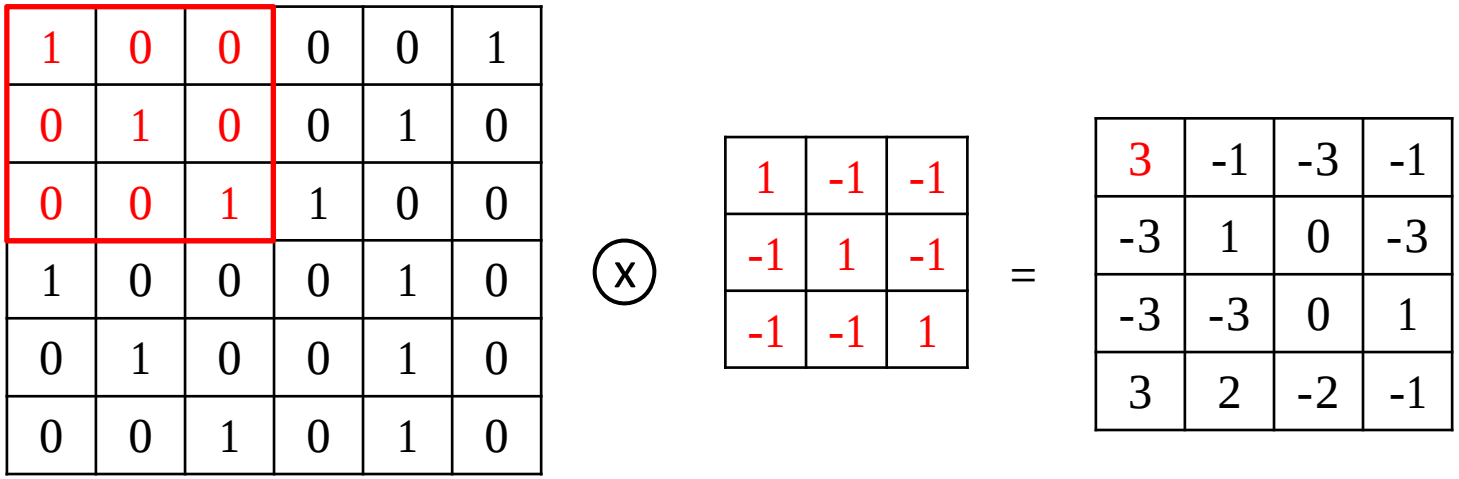
- Relative pixel densities (magnitude of numbers)



https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

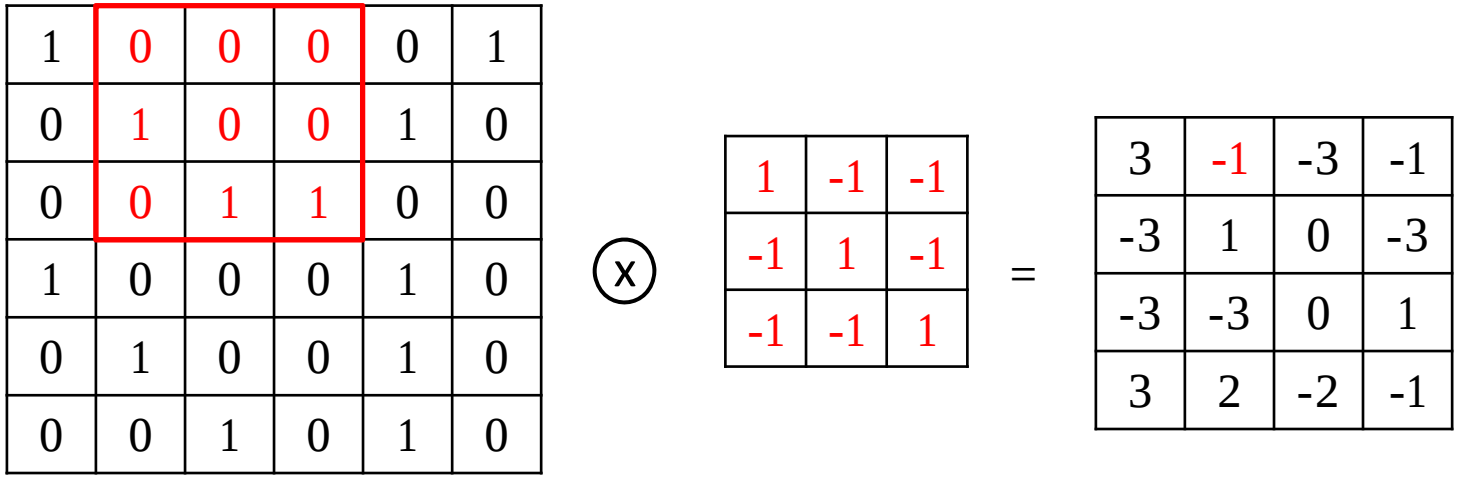
Convolution Layers

- Shares weights between local regions



Convolution Layers

- Shares weights between local regions



Convolution Layers

- Shares weights between local regions

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

⊗

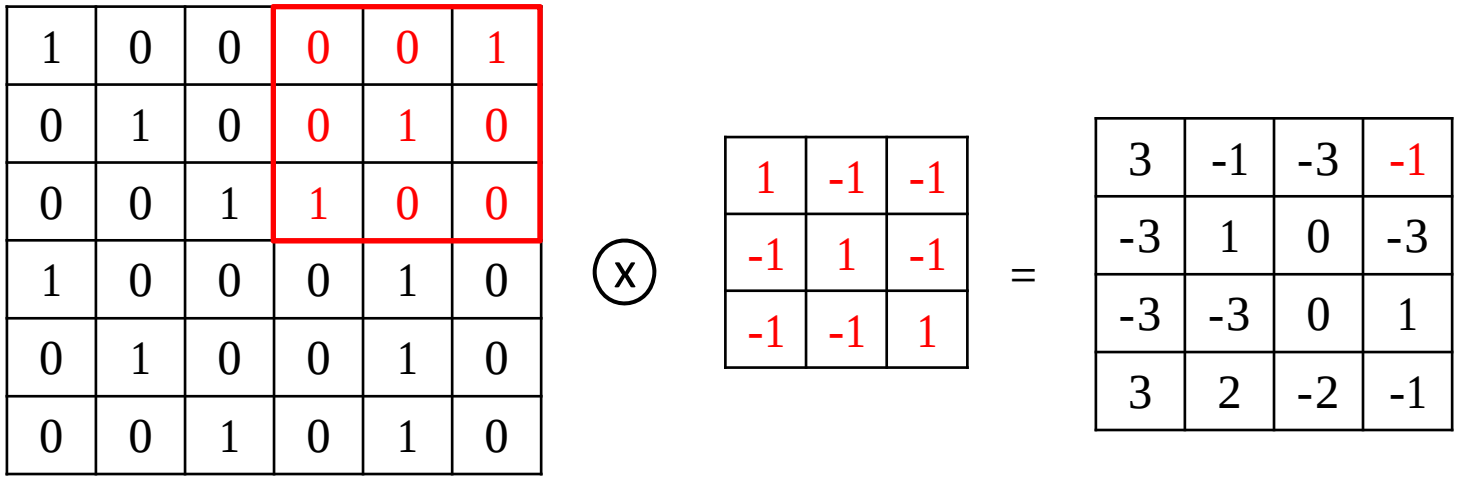
1	-1	-1
-1	1	-1
-1	-1	1

=

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	2	-2	-1

Convolution Layers

- Shares weights between local regions



Convolution Layers

- Shares weights between local regions

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

⊗

1	-1	-1
-1	1	-1
-1	-1	1

=

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	2	-2	-1

Convolution Layers

- Shares weights between local regions

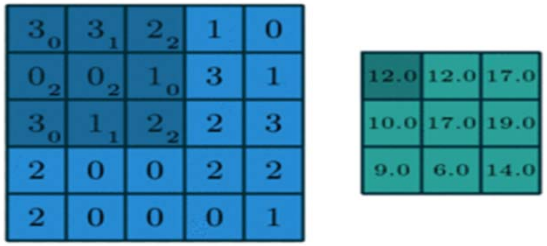
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

⊗

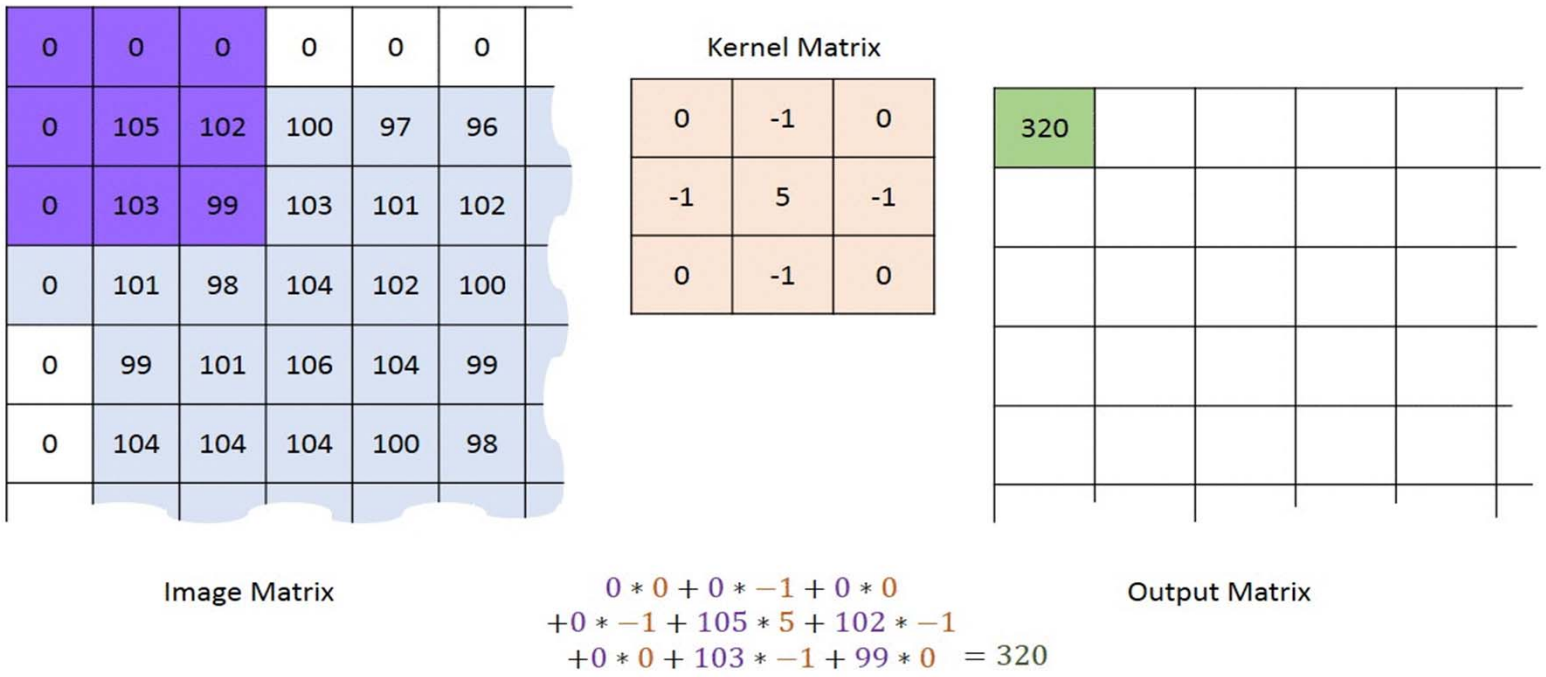
1	-1	-1
-1	1	-1
-1	-1	1

=

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	2	-2	-1



Convolution Layer

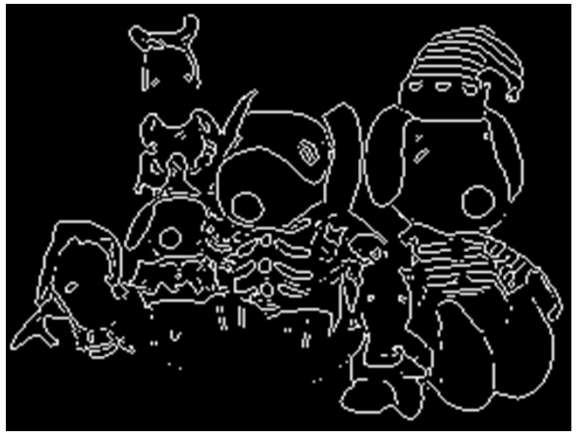


Convolution with horizontal and vertical strides = 1

http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

Filter

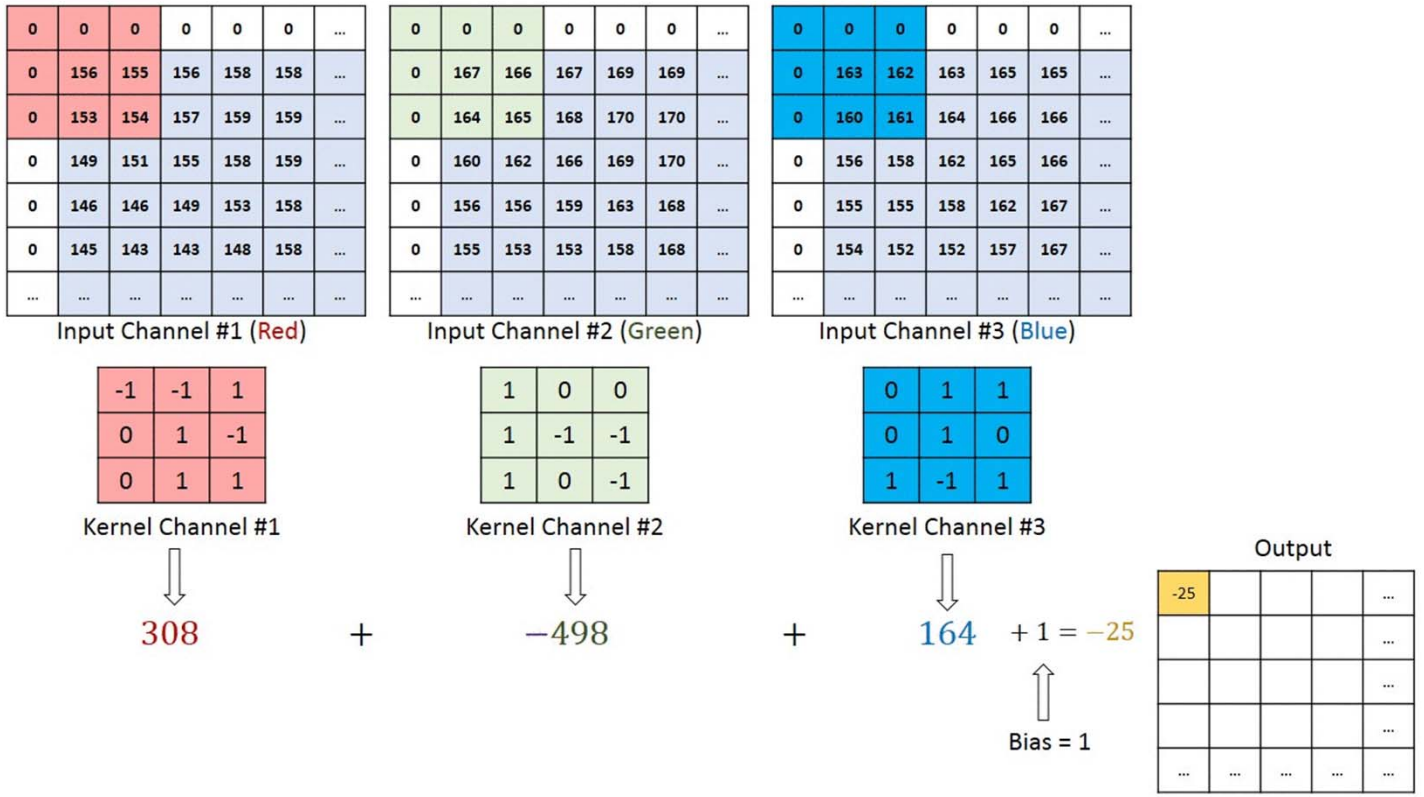
- Filters (Mask)



Prewitt/Sobel

Laplacian

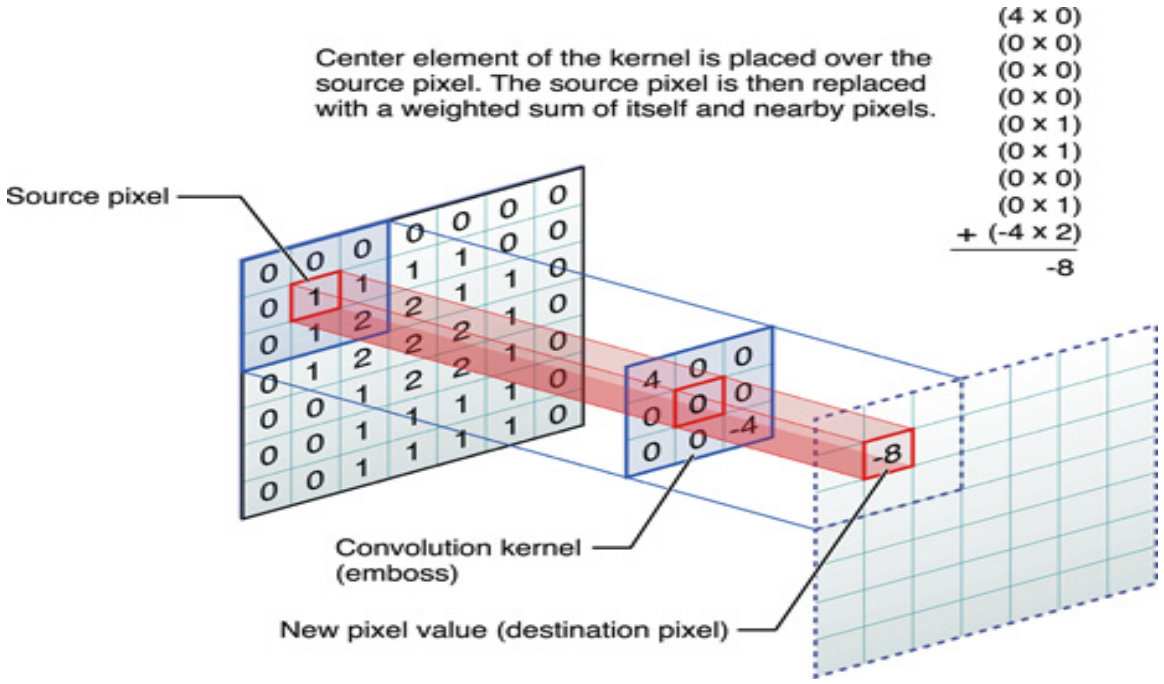
Convolution



http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html

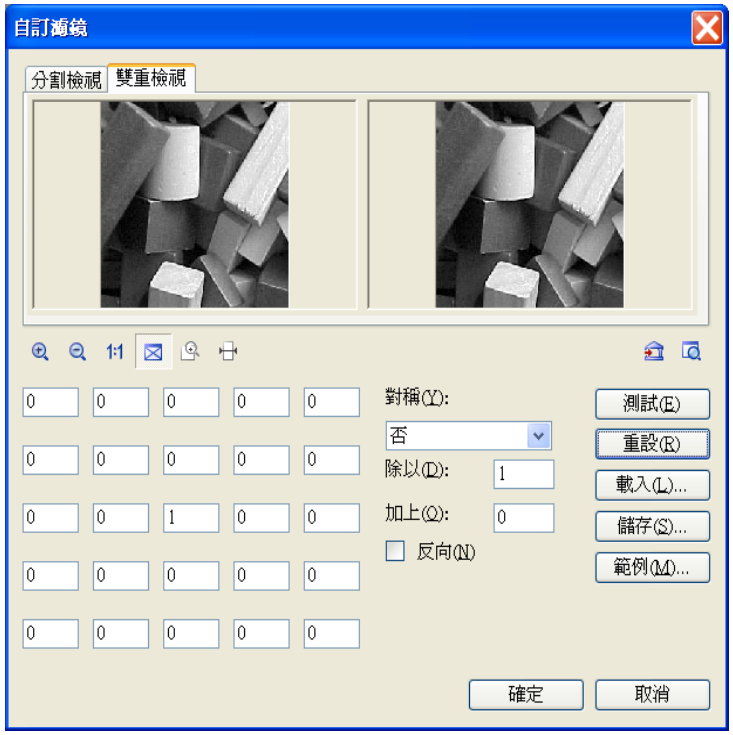
Smoothing

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



http://goo.gl/dck6tD

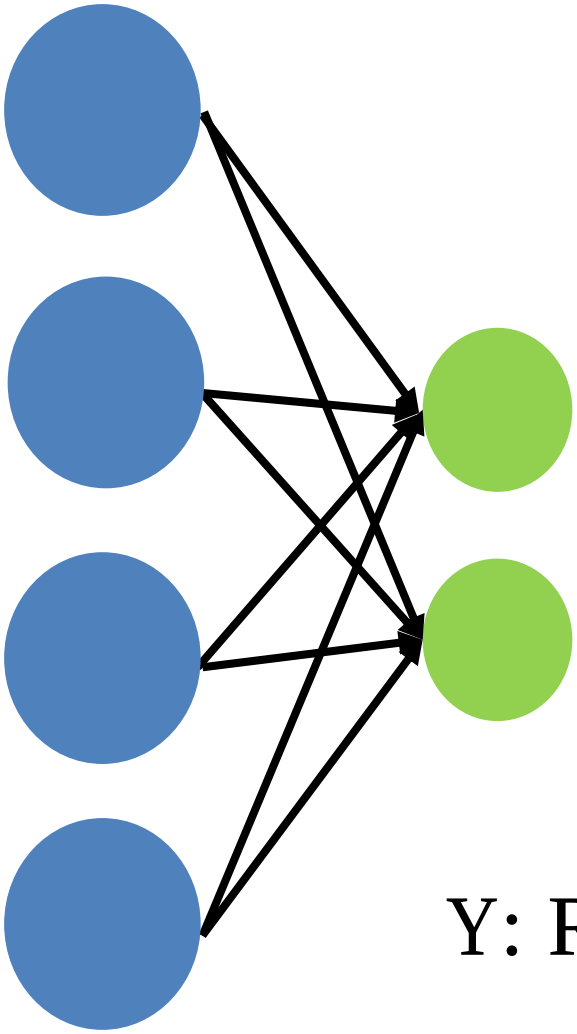
Demo



Feature Map

- Matrix
- Mapping
- Nonlinear
- Connected
- Dropout
- Normalization
- …

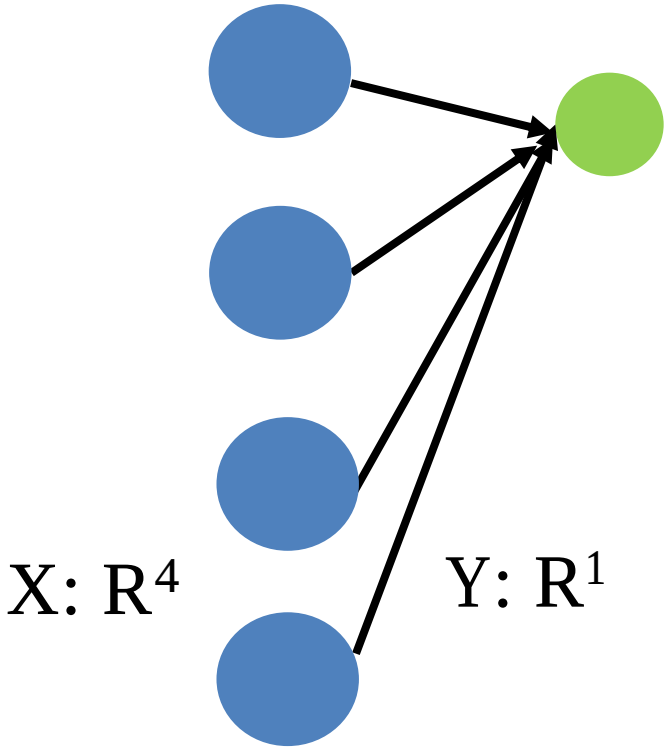
$Y=AX$



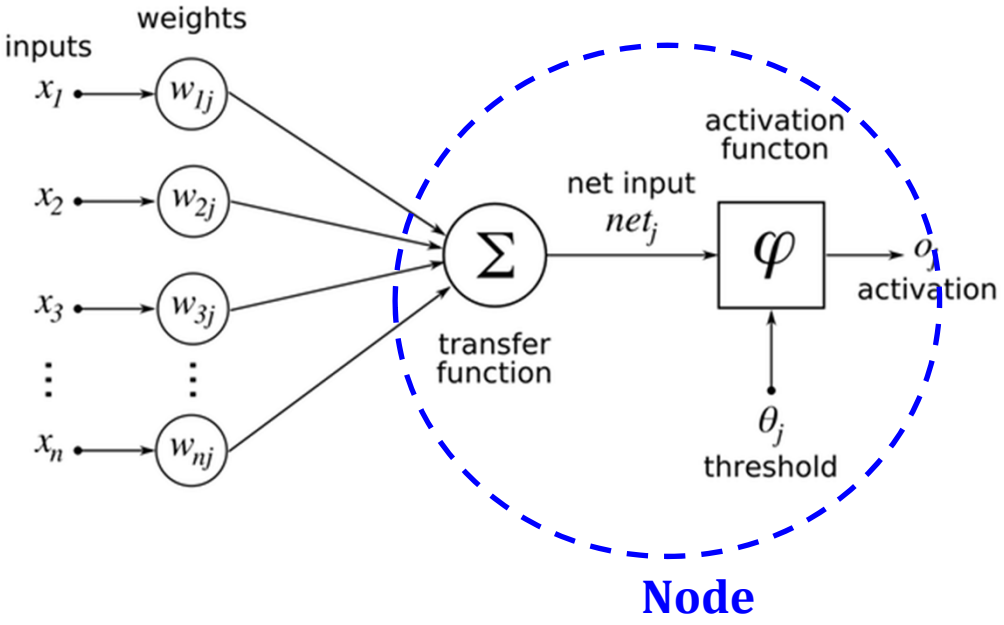
$X: R^4$

$Y: R^2$

Feature Map

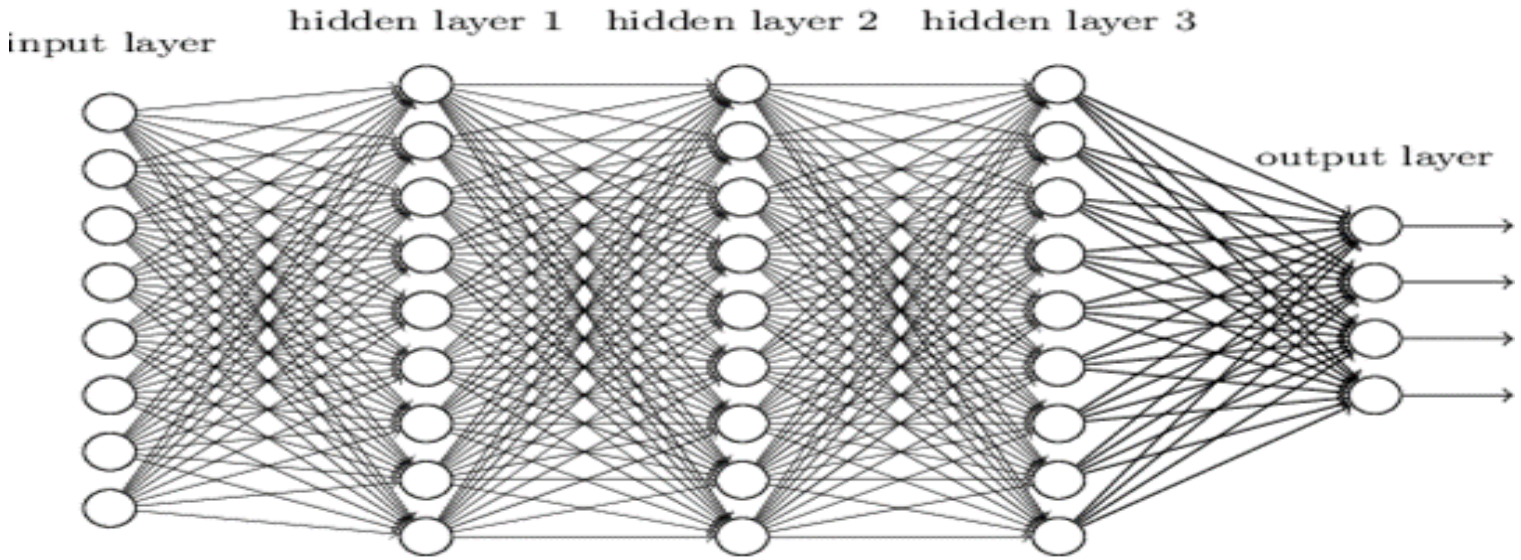


Sallow Neural Network



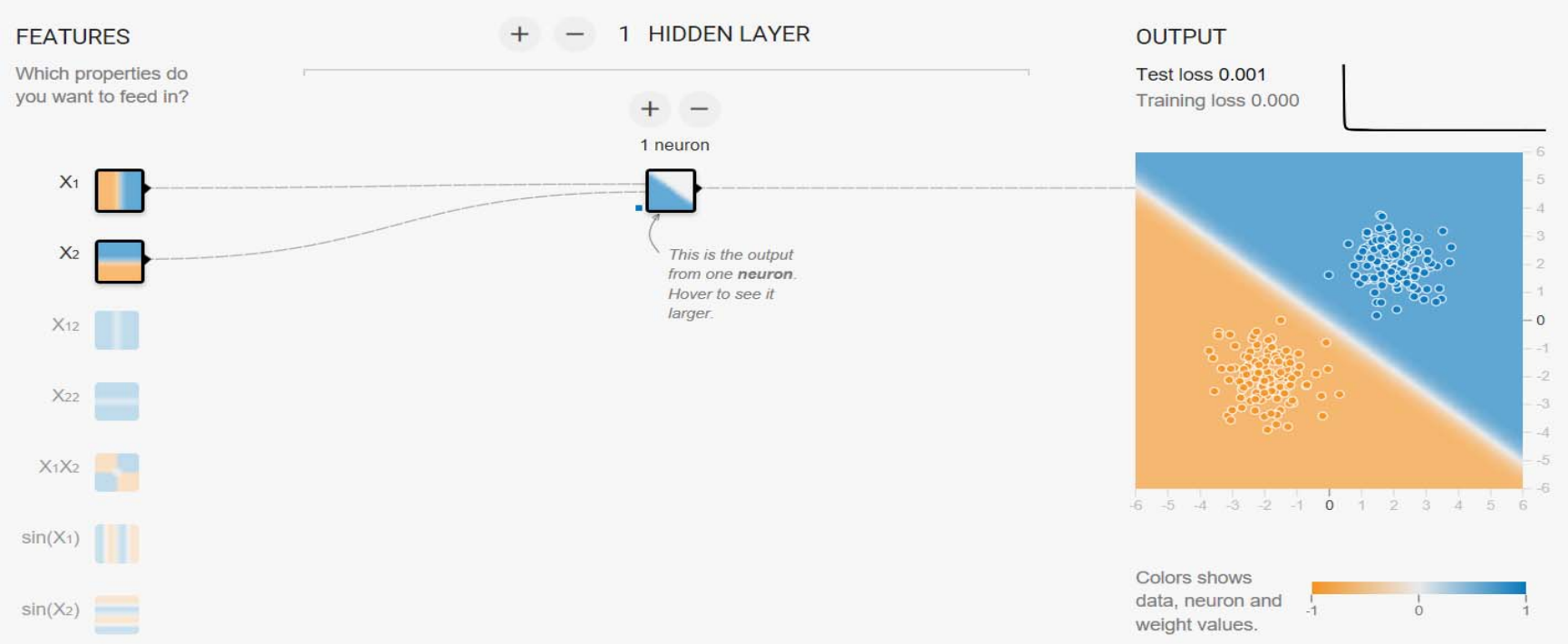
Deep Feature Map

- 深度學習模型 (Model)



Neural Network

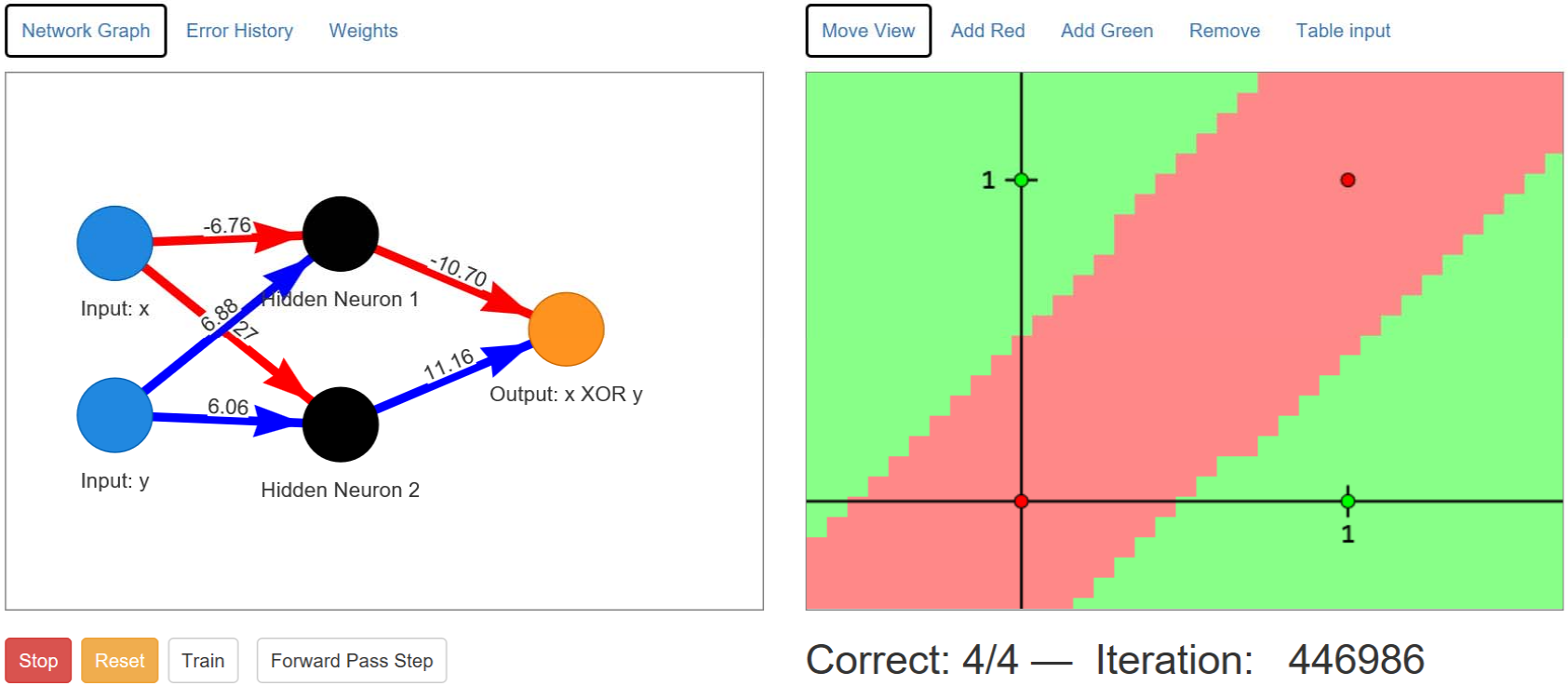
- **Linearly Problem**



https://playground.tensorflow.org/

Neural Network demo

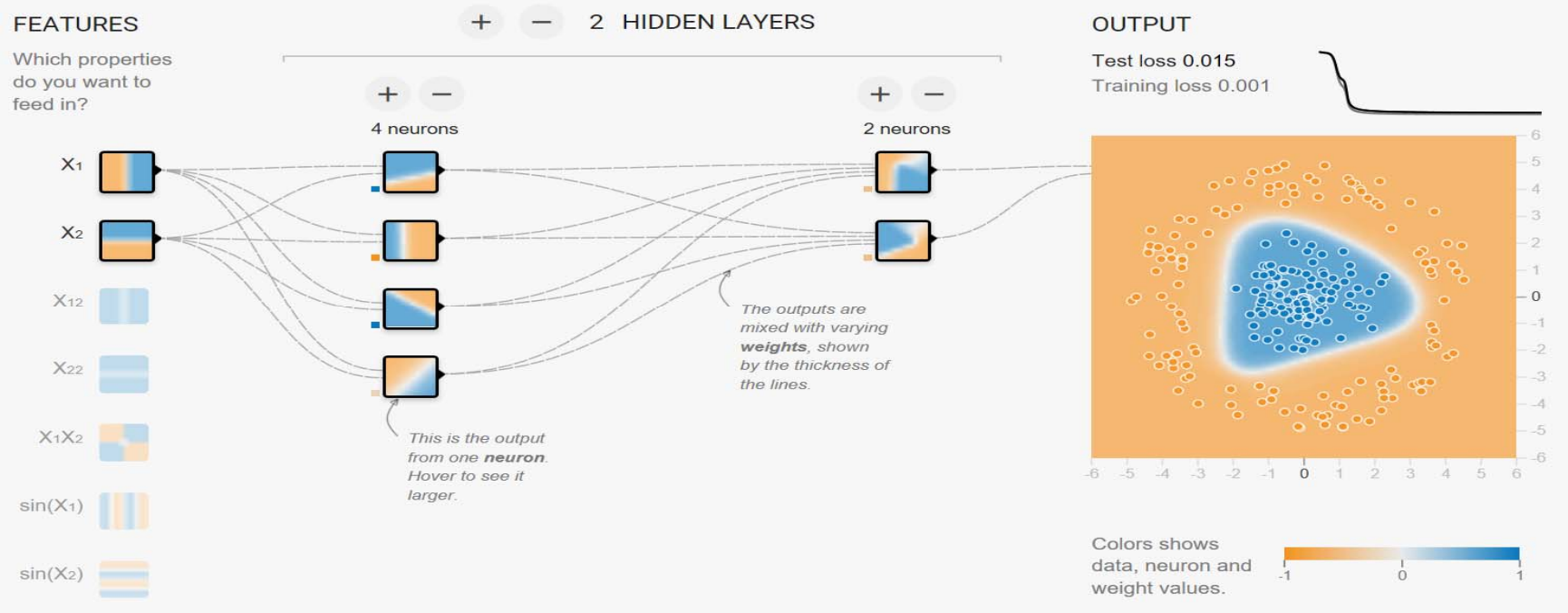
- **Nonlinear model**



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:

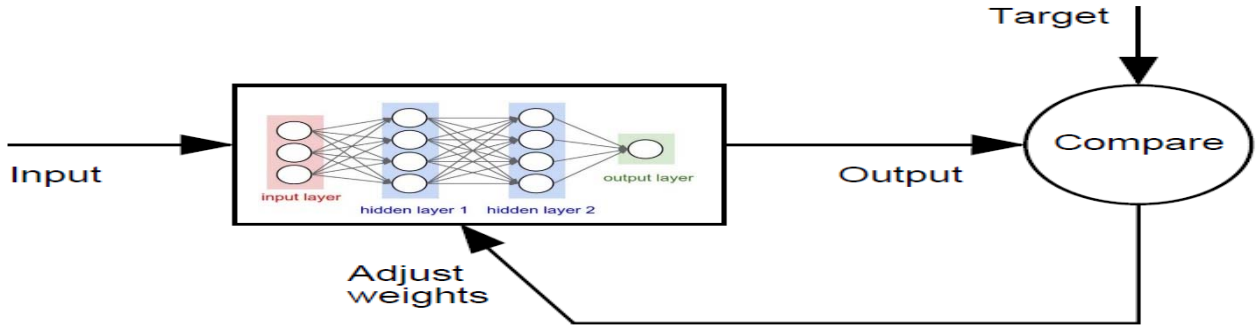
- kernel_regularizer (weight matrix)
- bias_regularizer (bias vector)
- activity_regularizer (output of activation)

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**()

modal.**add**(**Conv2D**())

modal.**add**(**Activation**())

modal.**add**(**Flatten**())

modal.**add**(**Dense**())

modal.**add**(**Dropout**())

modal.**complie**()

modal.**fit**(X,Y)

model.**evaluate**(X,Y): loss values & metrics values

model.**predict**(X)

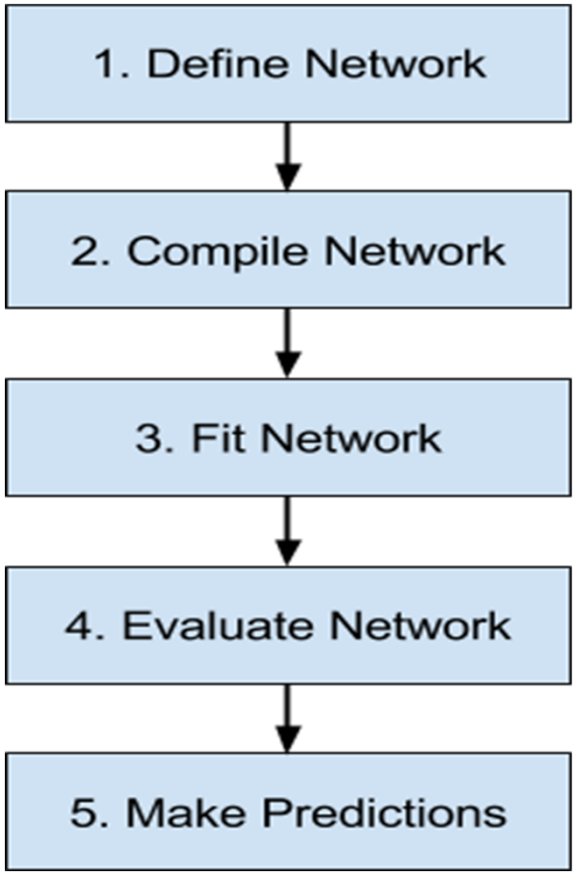
Composing models in Keras

- 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

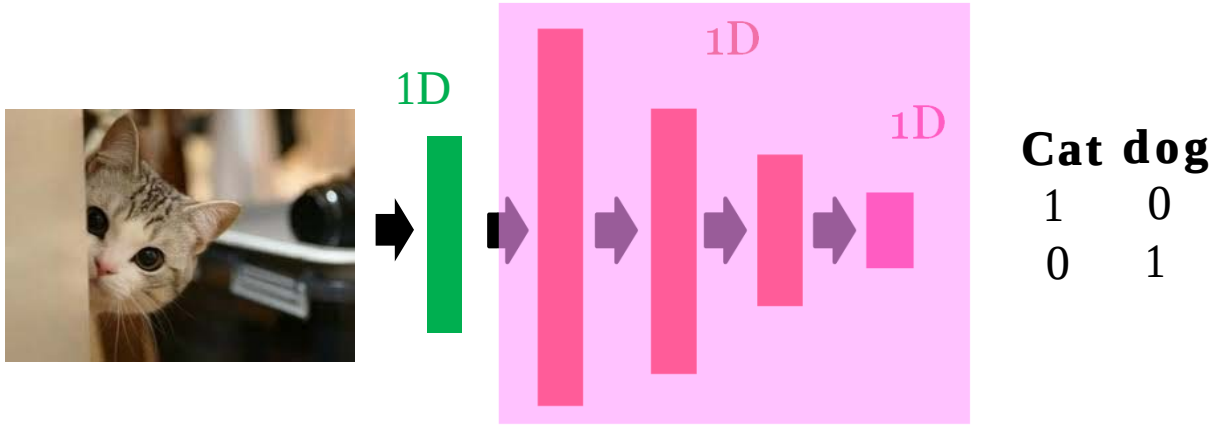
- Text
- Numerical
- Images

Neural Networks
(Deep Learning)

- Text
- Numerical
- Images

Model = Function

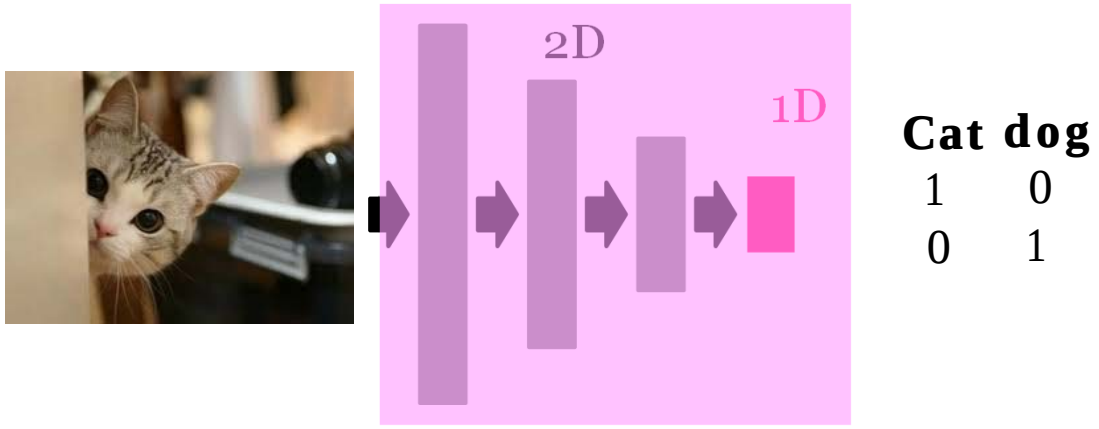
- reshape
- mapping (fully connected)



Acritical Neural Network (CNN): Classification/regression

Model = Function

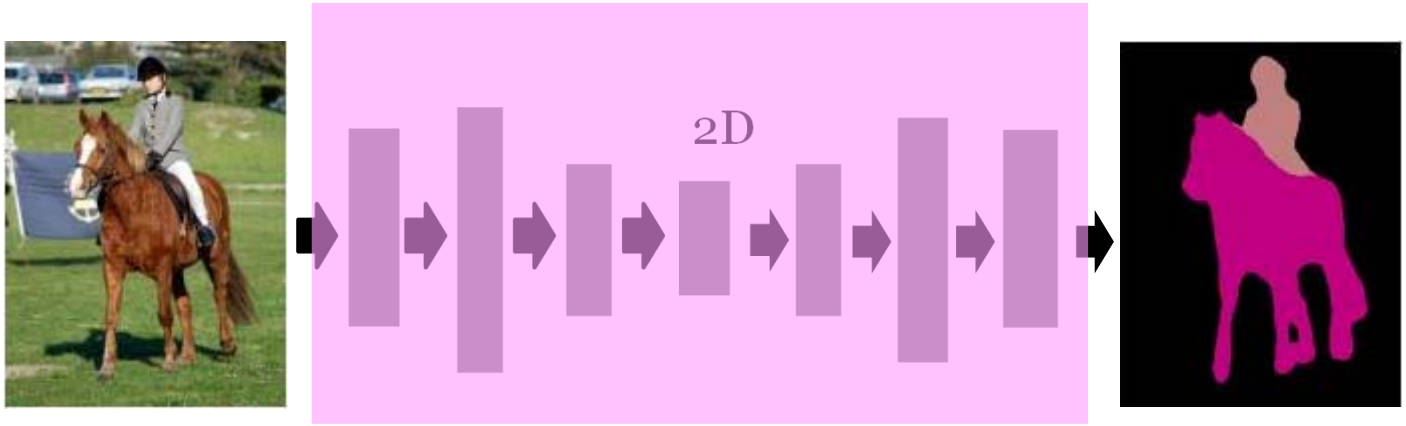
- reshape
- feature extraction (convolution)
- max pooling/downsamping



Convolutional Neural Network (CNN): Classification/regression

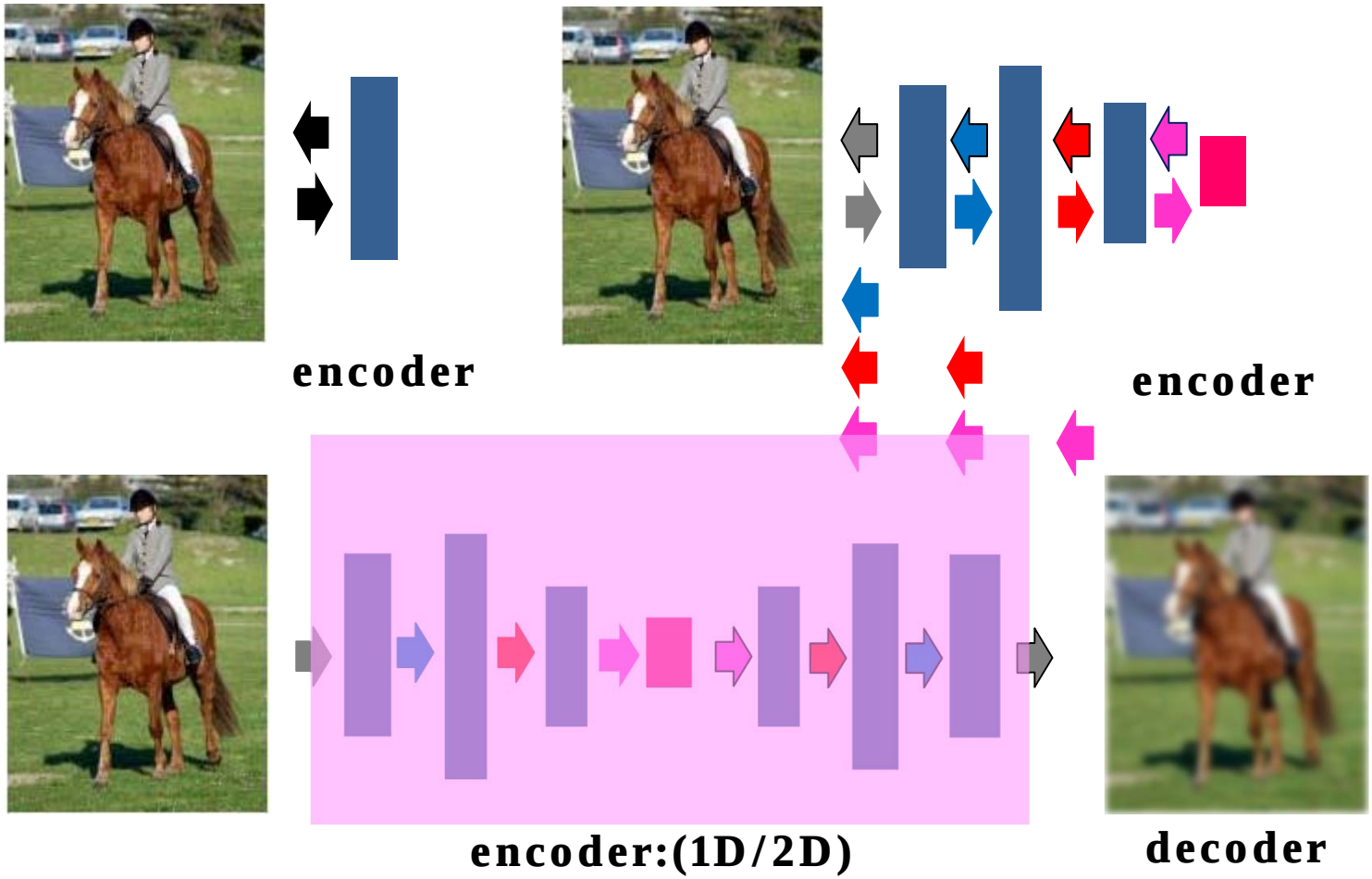
Model = Function

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



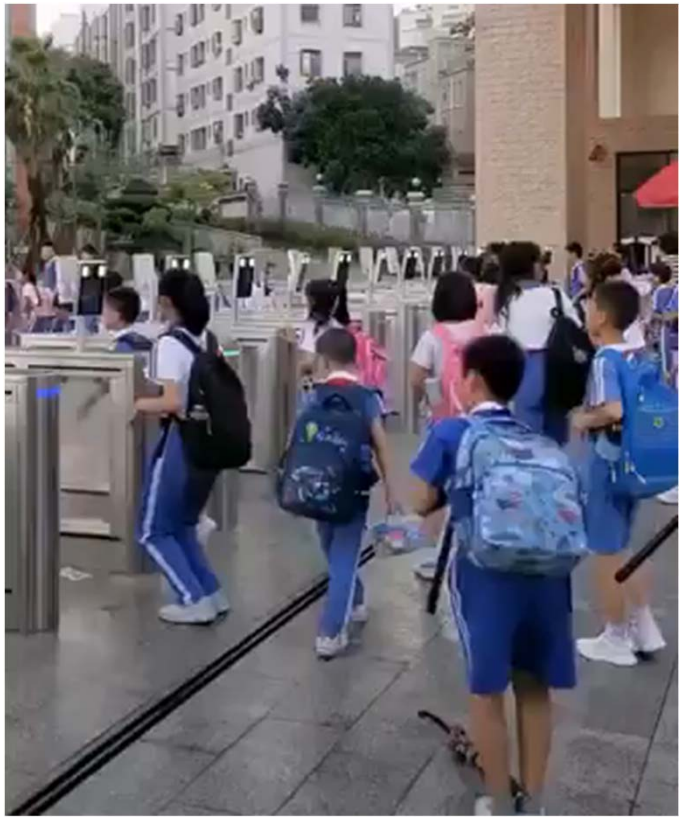
Fully Convolutional Network (FNN): Clustering

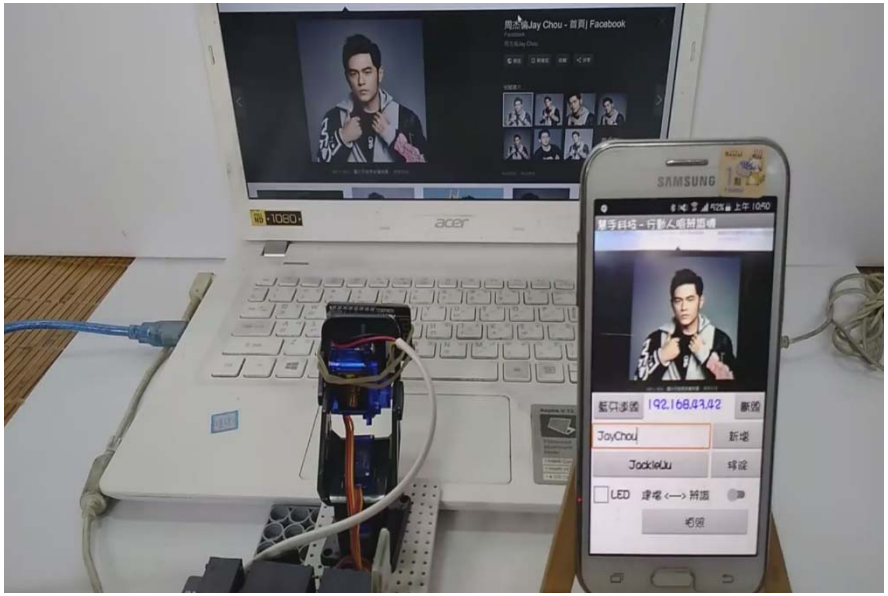
Model = Function



DEMO

- Edge/Cloud Computing

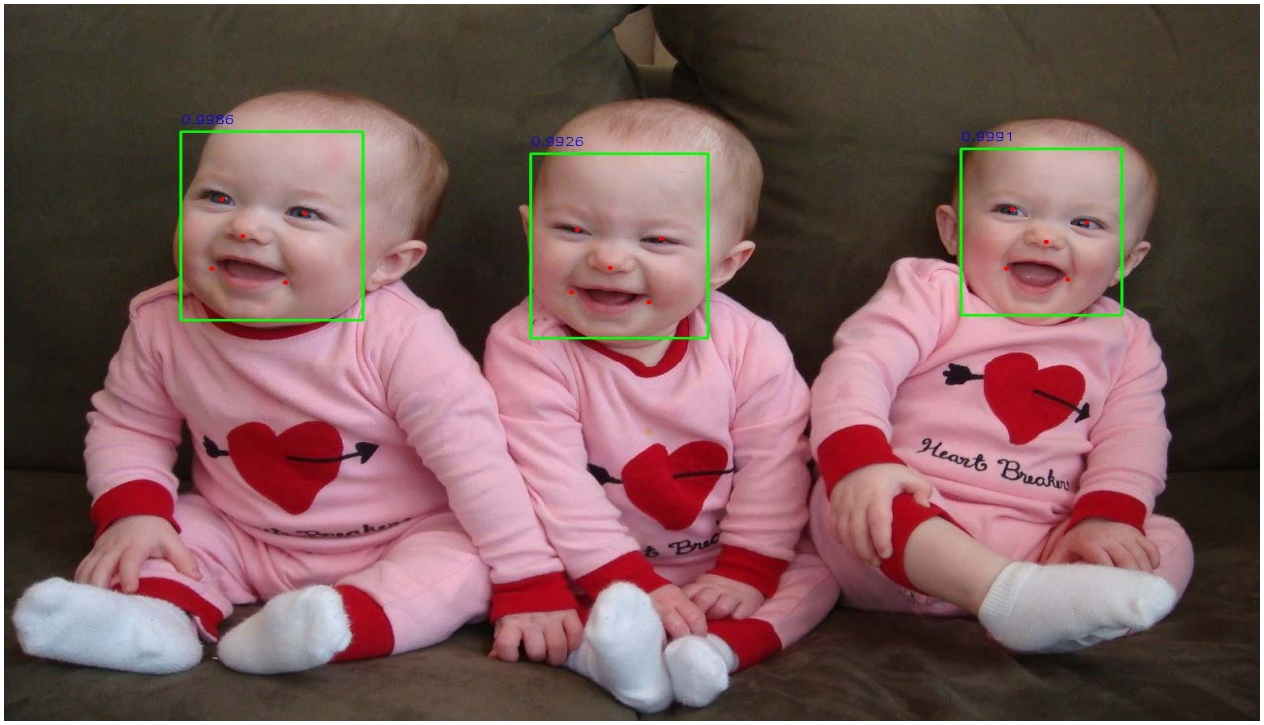




https://www.facebook.com/MarkeFactory/videos/938065716527342/

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-class-classification-with-lstm
- https://realpython.com/python-keras-text-classification/

專題實作一

- 修改 Kaggle BBC text classification