深度學習Deep Learning (4)
112-1

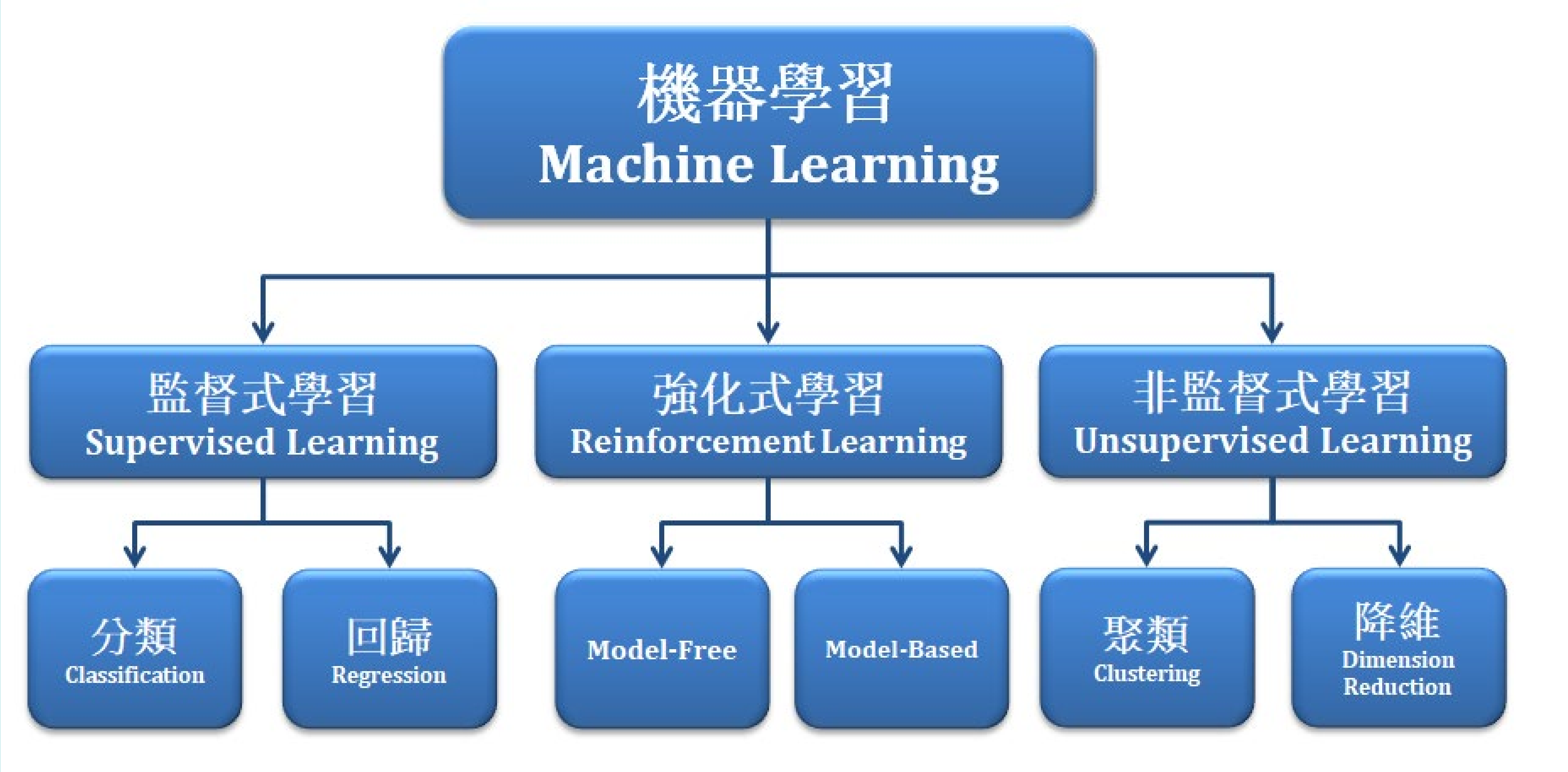
朱學亭老師

課程大綱

- W1-課程介紹/Introduction
- W2-Python/Colab and TensorFlow
- W3-神經網路/Numpy/Pandas
- W4-機器學習/Sklearn/PyTorch
- W5-CNN/Encoder–Decoder /GAN
- W6-RNN
- W7-Transformer
- W8-Computer Vision
- W9-Midterm presentation
- W10-Seq2Seq/Word2Vec
- W11-BERT
- W12-LLM
- W13-NLP1
- W14-NLP2
- W15-Audio Analysis
- W16-AICUP 1
- W17-AICUP 2
- W18-Final presentation

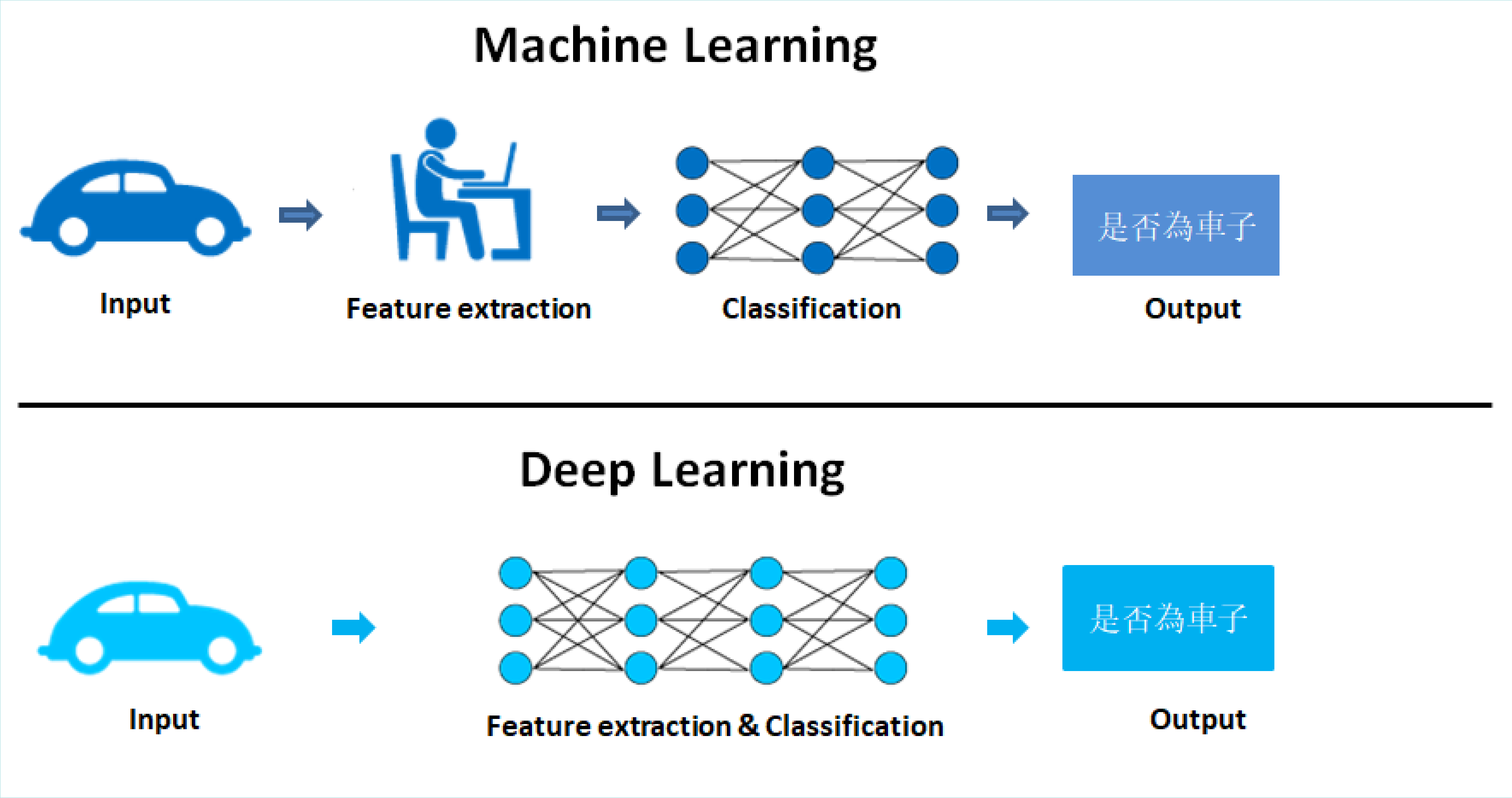
(1) 機器學習

機器學習



機器學習
Machine Learning
監督式學習
Supervised Learning
強化式學習
Reinforcement Learning
非監督式學習
Unsupervised Learning
分類
Classification
回歸
Regression
Model-Free
Model-Based
聚類
Clustering
降維
Dimension Reduction

ML VS DL


Machine Learning
Input
Feature extraction
Classification
是否為車子
Output
Deep Learning
Input
Feature extraction & Classification
是否為車子
Output

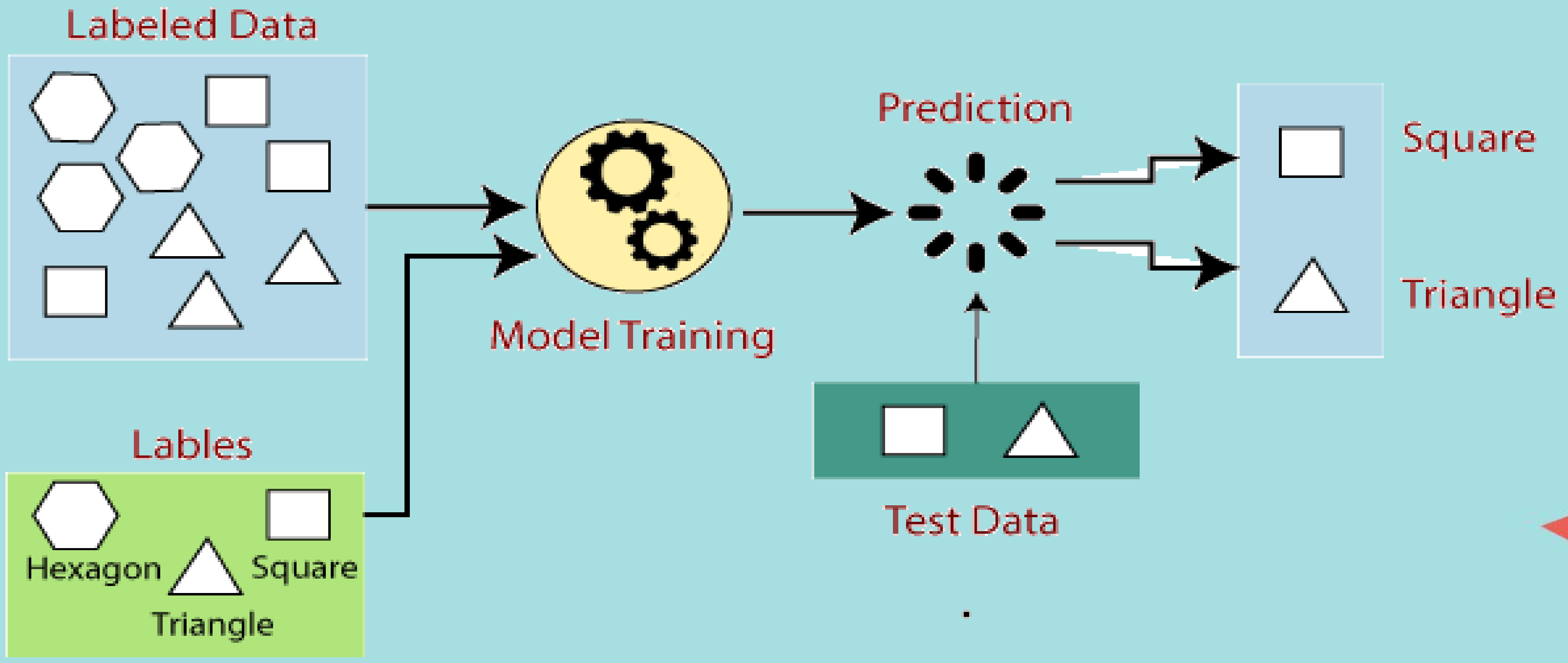
問題構建 (Framing)：機器學習主要術語

- 什麼是（監督式）機器學習？簡單來說，它的定義如下：
  - 機器學習系統通過學習如何組合輸入資訊來對從未見過的資料做出有用的預測。
- 機器學習的基本術語
  - **標籤 (Labels)**
  - **特徵 (Features)**
  - **樣本 (Examples)**
  - **模型 (Models)**
  - 回歸與分類 (Regression vs. classification)

回歸與分類 (Regression vs. classification)

- A **regression** model (回歸) predicts continuous values. For example, regression models make predictions that answer questions like the following:
  - What is the value of a house in California?
  - What is the probability that a user will click on this ad?
- A **classification** model (分類) predicts discrete values. For example, classification models make predictions that answer questions like the following:
  - Is a given email message spam or not spam?
  - Is this an image of a dog, a cat, or a hamster?

Features and Labels
特徵和標籤



K-NN (K Nearest Neighbor)

- 在模式識別領域中，最近鄰居法是一種用於分類和回歸的無母數統計方法。在這兩種情況下，輸入包含特徵空間中的k個最接近的訓練樣本。

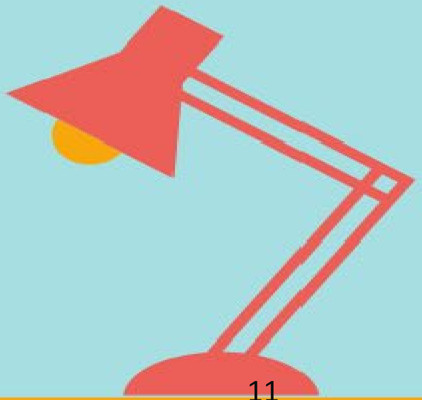
- 在k-NN分類中，輸出是一個分類族群。一個物件的分類是由其鄰居的「多數表決」確定的，k個最近鄰居（k為正整數，通常較小）中最常見的分類決定了賦予該物件的類別。若k = 1，則該物件的類別直接由最近的一個節點賦予。
- 在k-NN回歸中，輸出是該物件的屬性值。該值是其k個最近鄰居的值的平均值。

K-Means(k-means clustering)

- 「K means」是一種聚類(Cluster) 的方式. 聚類基本上就是依照著「物以類聚」的方式在進行. (或許我們也可能想成，相似的東西有著相似的特徵).
- 給予一組資料，將之分為k類 (k由使用者設定) 就是「K means」的用處.

隨機森林(Random forest)

- **決策樹**（Decision tree）
- **分類樹**分析
- **回歸樹**分析
- CART分析



支持向量機
(Support vector machine;SVM)

- 監督式學習演算法。
- 分類與迴歸分析
- 線性分類
- 非線性分類

基因演算法
(Genetic Algorithm;GA)

- 基因演算法是計算數學中用於解決最佳化的搜索算法，是擬生物算法的一種。
- 進化算法最初是借鑑了進化生物學中的一些現象而發展起來的，這些現象包括遺傳、突變、自然選擇以及雜交等。

啟發式演算法

- 基因演算法(Genetic Algorithm，GA)
- 模擬退火演算法(Simulated Annealing，SA)
- 禁忌搜尋法(Tabu Search，TS)
- 蟻群最佳化演算法(Ants Colony Optimization，ACO)
- 粒子群最佳化演算法(Particle Swarm Optimization，PSO)
- 細菌覓食最佳化演算法(Bacterial Foraging Optimization，BFO)、
- 蜜蜂演算法(Artificial Bee Colony Algorithm，ABCA)

主成分分析（PCA）

- 數據集 X 的主成分 w_1 可以被定義為：

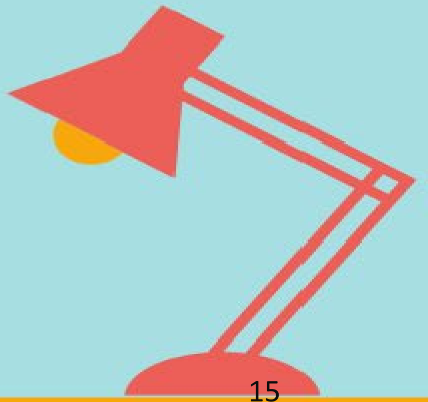
$$\mathbf{w}_1 = \underset{\Vert \mathbf{w} \Vert = 1}{\arg\max}\ \mathrm{Var}\{\mathbf{w}^\top \mathbf{X}\} = \underset{\Vert \mathbf{w} \Vert = 1}{\arg\max}\ E\left\{\left(\mathbf{w}^\top \mathbf{X}\right)^2\right\}$$

為了得到第 k 個主成分，必須先從 $\mathbf{X}$ 中減去前面的 $k-1$ 個主成分：

$$\hat{\mathbf{X}}_{k-1} = \mathbf{X} - \sum_{i=1}^{k-1} \mathbf{w}_i \mathbf{w}_i^\top \mathbf{X}$$

然後把求得的第 k 個主成分帶入數據集，得到新的數據集，繼續尋找主成分。

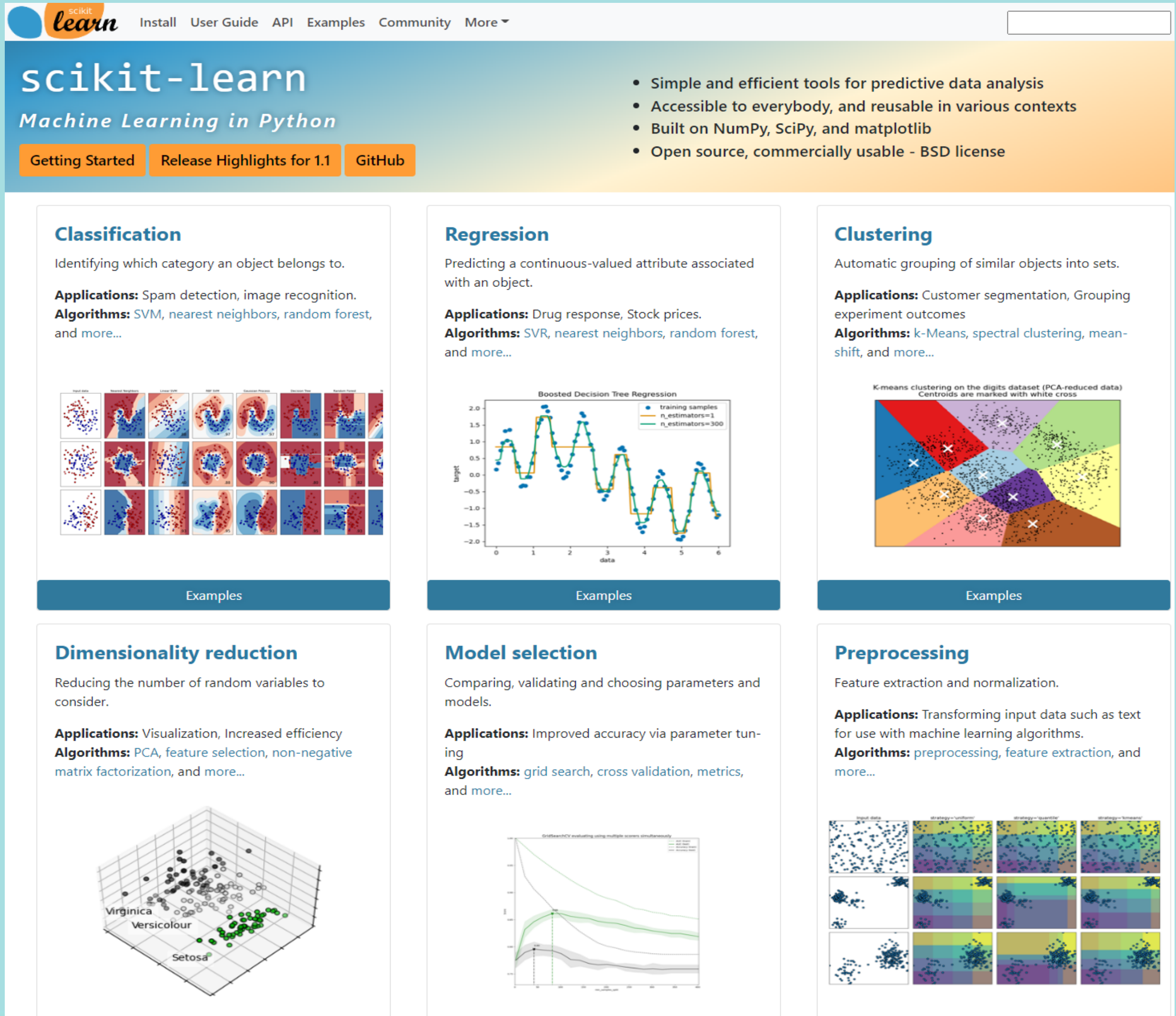
$$\mathbf{w}_k = \underset{\Vert \mathbf{w} \Vert = 1}{\arg\max}\ E\left\{\left(\mathbf{w}^\top \hat{\mathbf{X}}_{k-1}\right)^2\right\}.$$



(2) SKLEARN: SCIKIT-LEARN

scikit-learn

Machine Learning in Python



https://scikit-learn.org/

鳶尾花卉數據集*Iris dataset*

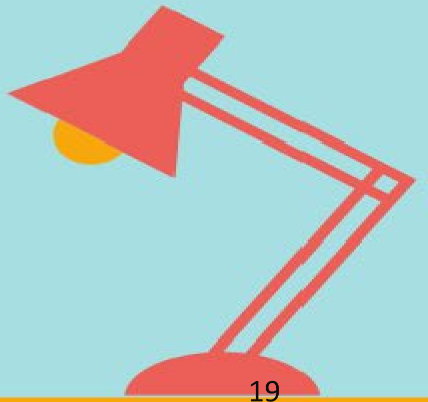
- 山鳶尾*Iris setosa*

- 變色鳶尾Iris versicolor

- 維吉尼亞鳶尾Iris virginica

四個數值特徵值

- 鳶尾花的「萼片長」、「萼片寬」、「花瓣長」、「花瓣寬」
- 資料共有150筆。(每個類別各50筆)
- epal_length,sepal_width,petal_length,petal_width,class name
- 5.1,3.5,1.4,0.2,Iris-setosa
- 4.9,3.0,1.4,0.2,Iris-setosa
- 6.7,3.1,4.4,1.4,Iris-versicolor
- 5.6,3.0,4.5,1.5,Iris-versicolor
- 6.1,2.6,5.6,1.4,Iris-virginica
- 7.7,3.0,6.1,2.3,Iris-virginica



http://www.ics.uci.edu/~mlearn/MLRepository.html

(3) TENSORFLOW

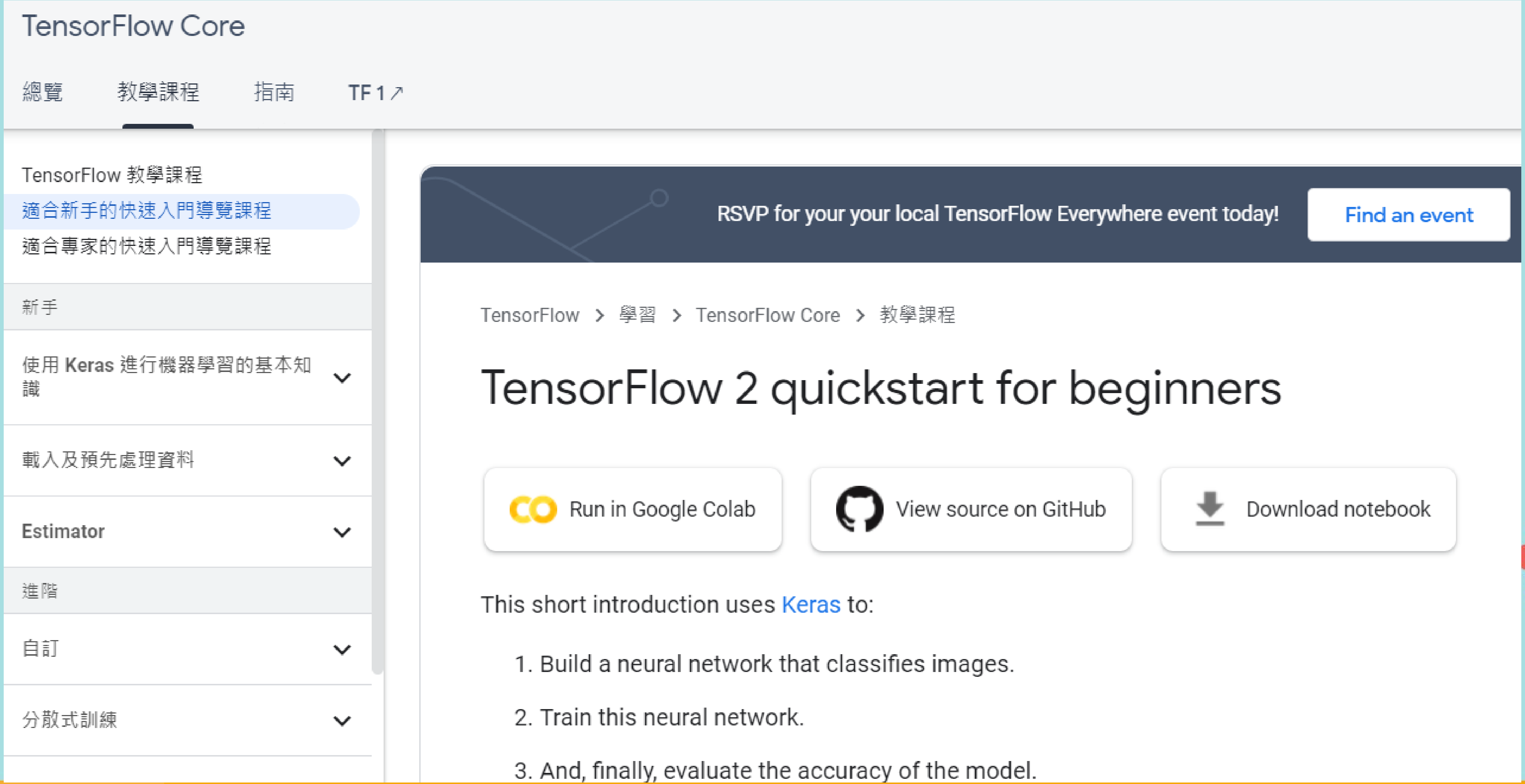
TensorFlow 2.0 three model APIs

(1) Sequential API

(2) Subclassing API

(3) Functional API

Sequential model



Model subclassing

TensorFlow Core

總覽 教學課程 指南 TF 1 ↗

TensorFlow 教學課程
適合新手的快速入門導覽課程
適合專家的快速入門導覽課程

新手

使用 Keras 進行機器學習的基本知識

載入及預先處理資料

Estimator

進階

自訂

分散式訓練

圖片

RSVP for your your local TensorFlow Everywhere event today! Find an event

TensorFlow > 學習 > TensorFlow Core > 教學課程

TensorFlow 2 quickstart for experts

Run in Google Colab | View source on GitHub | Download notebook

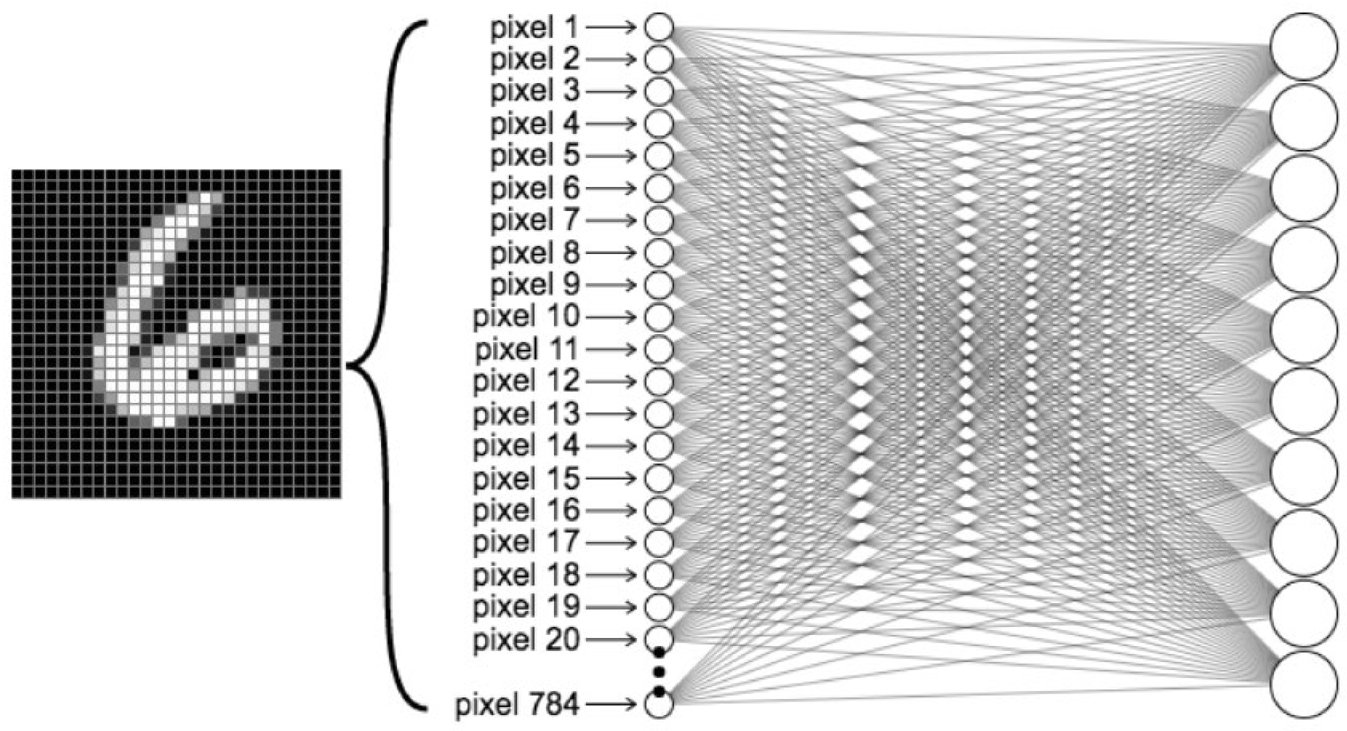
This is a Google Colaboratory notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To follow this tutorial, run the notebook in Google Colab by clicking the button at the top of this page.

1. In Colab, connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. Run all the notebook code cells: Select *Runtime* > *Run all*.

Download and install TensorFlow 2. Import TensorFlow into your program:

MNIST handwritten digit dataset

Feed Forward Neural Networks



NN with input and output layer, hidden layer is hidden :)

```
model = keras.models.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(128, activation='relu'),
  keras.layers.Dropout(0.2),
  keras.layers.Dense(10, activation='softmax')
])

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

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test, y_test)
```

IPO (1)-Hello Word

```
[2] mnist = tf.keras.datasets.mnist
    (x_train, y_train), (x_test, y_test) = mnist.load_data()
    x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
[3] print(x_train.shape)
    print(y_train.shape)
```

```
(60000, 28, 28)
(60000,)
```

Input

Output

```
[4] model = tf.keras.models.Sequential([
      tf.keras.layers.Flatten(input_shape=(28, 28)),
      tf.keras.layers.Dense(128, activation='relu'),
      tf.keras.layers.Dropout(0.2),
      tf.keras.layers.Dense(10)
    ])
```

```
[5] loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
    model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy'])
```

Build a model

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

Training & Inference

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

model.fit(x_train, y_train, epochs=5)

model.predict(test_img)
model.evaluate(x_test, y_test)
```

Test a single image

```
[ ] from PIL import Image
    from IPython.display import display
```

```
[ ] img = Image.open( "Digit4.bmp" )
    print(img.format, img.size, img.mode)
```

```
[ ] display(img)
```

```
[ ] import tensorflow as tf
```

```
[ ] model = tf.keras.models.load_model('my_model.h5')
    model.summary()
```

```
[ ] import numpy as np
    img = np.resize(img, (28,28))
    im2arr = np.array(img)
    im2arr = im2arr.reshape(1, 28,28)
    y_pred = new_model.predict_classes(im2arr)
    print(y_pred)
```

```
[ ] img = Image.open( "DigitX.bmp" )
    print(img.format, img.size, img.mode)
    display(img)
```

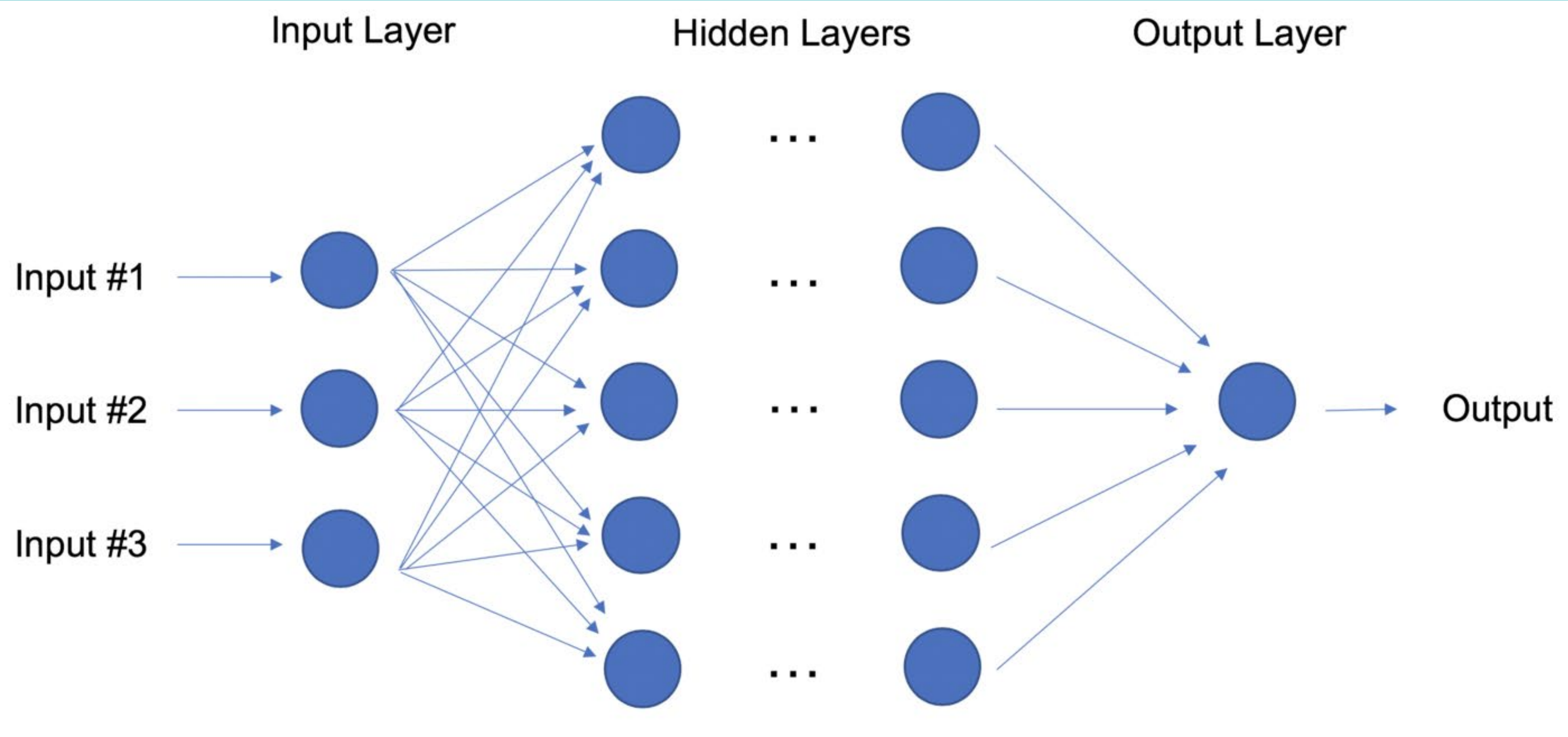
```
img = np.resize(img, (28,28))
im2arr = np.array(img)
im2arr = im2arr.reshape(1, 28,28)
y_pred = new_model.predict_classes(im2arr)
print(y_pred)
```

(4) PYTORCH

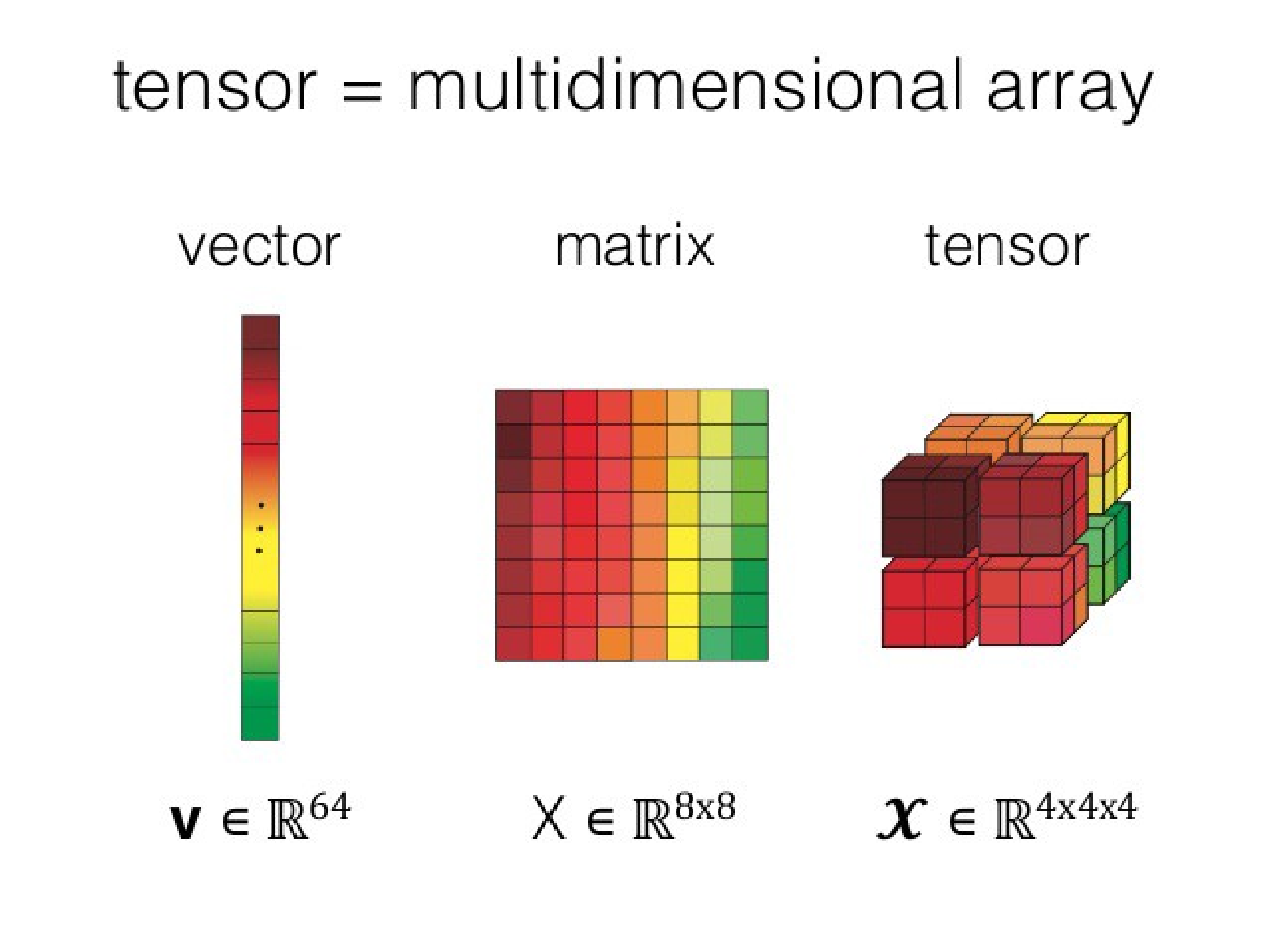
PyTorch overview

- Neural Network using PyTorch
- PyTorch Tensor
- torch.nn
- torch.autograd
- torch.no_grad()

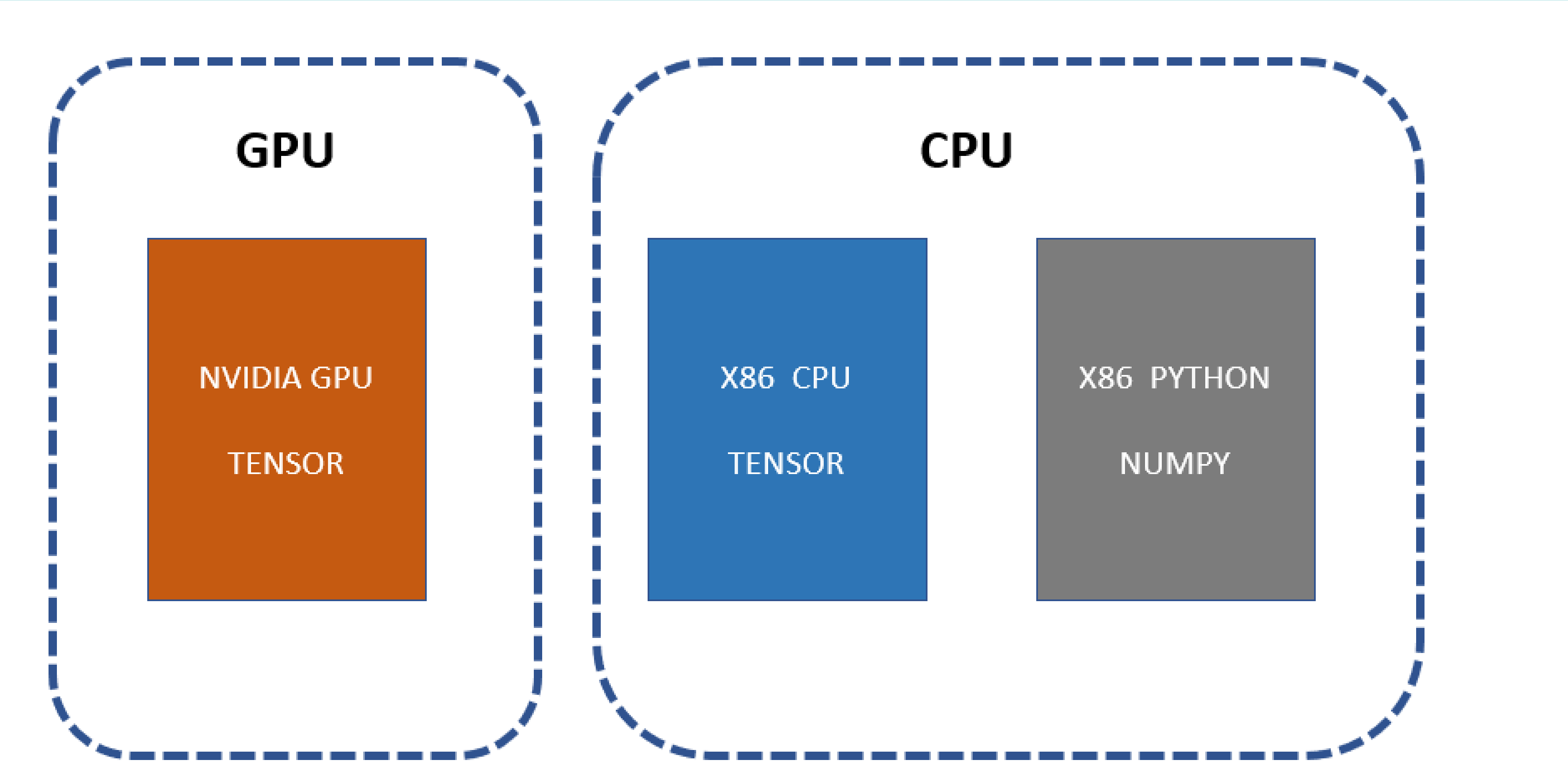
About deep learning (Neural Network)



PyTorch Tensor



tensor.cuda()<-->tensor.cpu()



Example of a PyTorch Network model

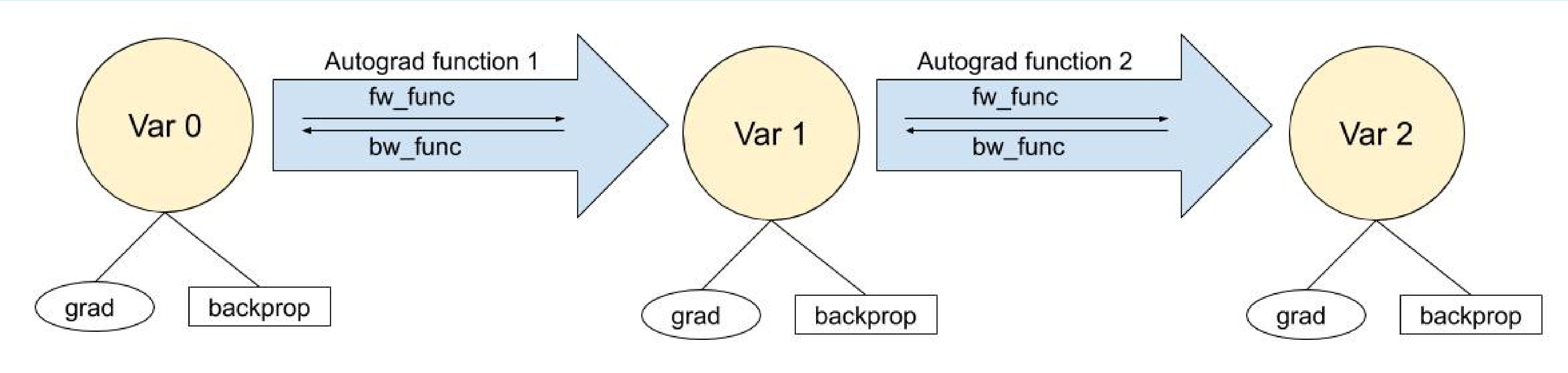
```
class Sample_Network(nn.Module):
    def __init__(self):
        super(Sample_Network, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Define the layers of model

Forward function

Autograd



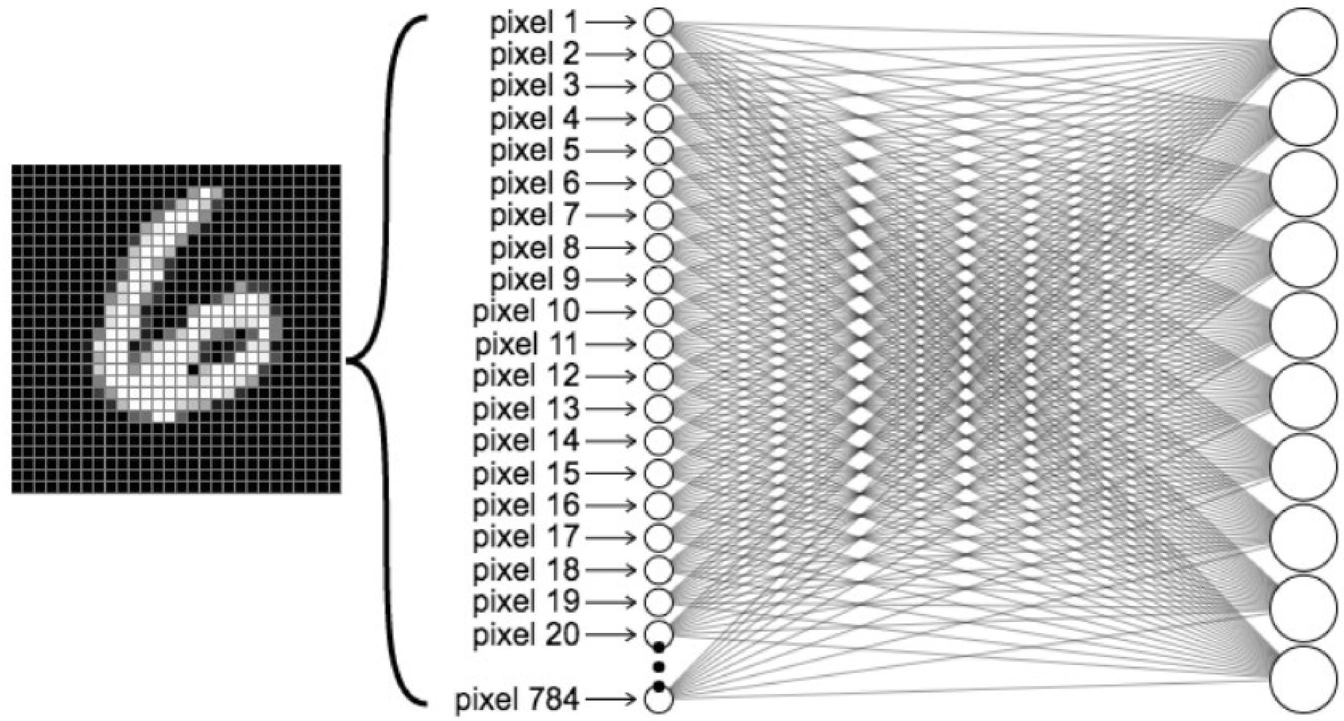
The term Autograd, or Automatic Differentiation, does not essentially mean calculating the gradients; that should instead be referred to as symbolic differentiation or numerical differentiation. A more precision definition of Autograd should be “automatically chaining the gradients”.

https://medium.com/@ralphmao95/simple-autograd-implementation-understand-automatic-differentiation-hand-by-hand-9e86f6d703ab

no_grad()

```
# backward + optimize only if in training phase
if phase == 'train':
    loss.backward()
      optimizer.step()
    with torch.no_grad():
        for p in base_model.parameters():
            p.sub_(learning_rate* p.grad)
            print(p.grad)
            p.grad.zero_()
            p = p.clone()
```

Feed Forward Neural Networks (PyTorch)



NN with input and output layer, hidden layer is hidden :)

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

model = NeuralNetwork().to(device)
print(model)
```

Thanks!

Q&A