機器學習於材料資訊的應用
Machine Learning on Material Informatics

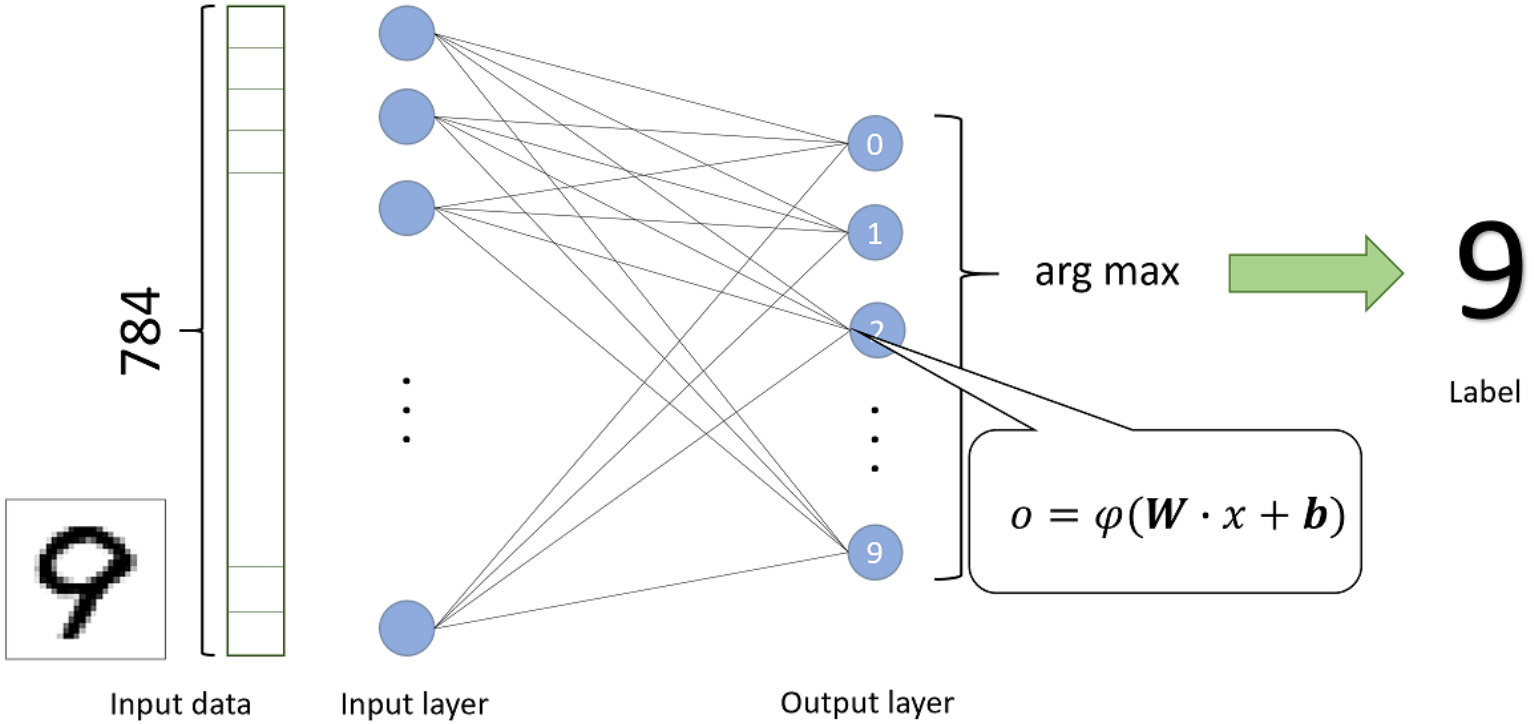
陳南佑(NAN-YOW CHEN)

nanyow@narlabs.org.tw

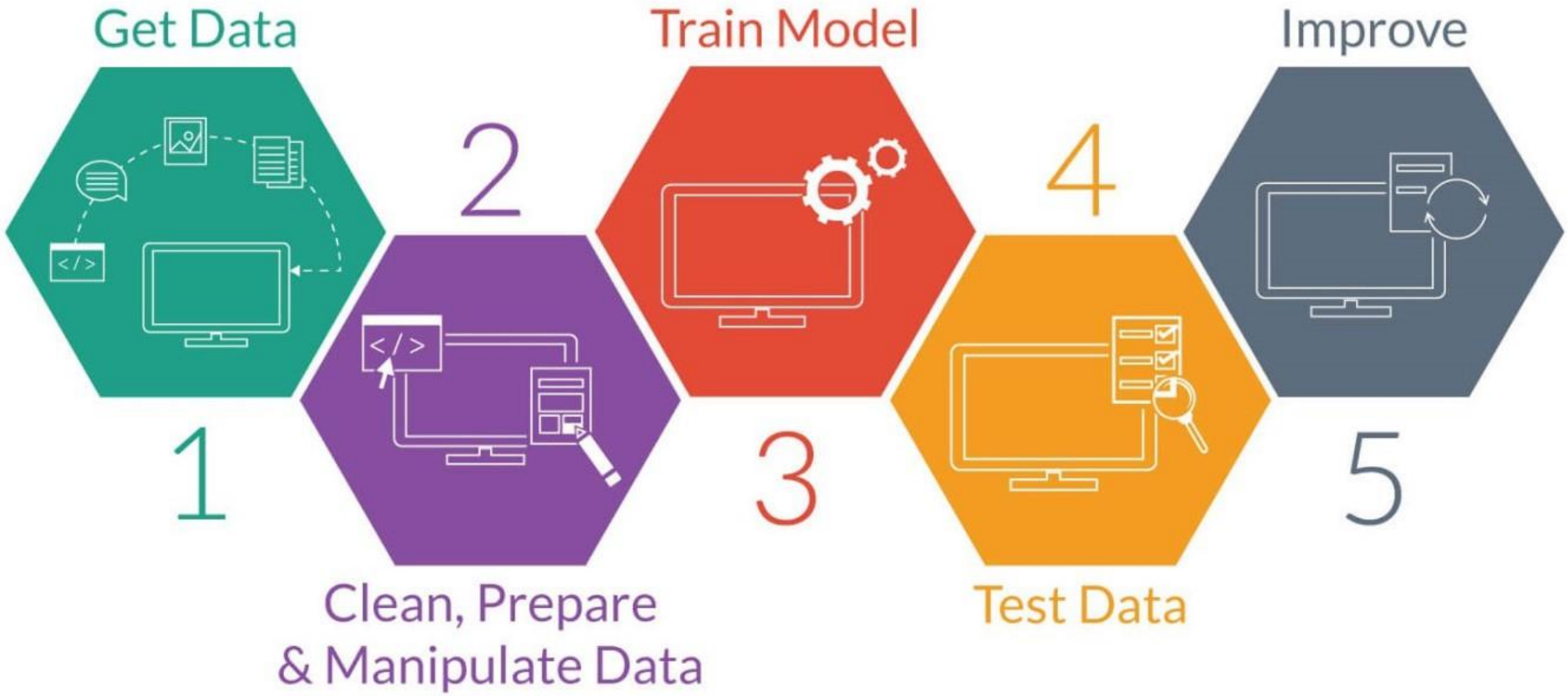
楊安正(AN-CHENG YANG)

acyang@narlabs.org.tw

Handwritten Digits classification by ANN



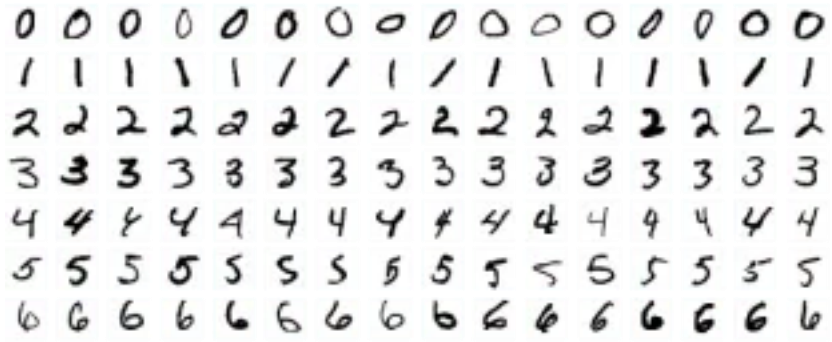
https://www.kaggle.com/fazilbtopal/a-detailed-cnn-with-tensorflow


Get Data
1
2
Clean, Prepare
& Manipulate Data
Train Model
3
4
Test Data
Improve
5

Get Data

THE MNIST DATABASE

of handwritten digits



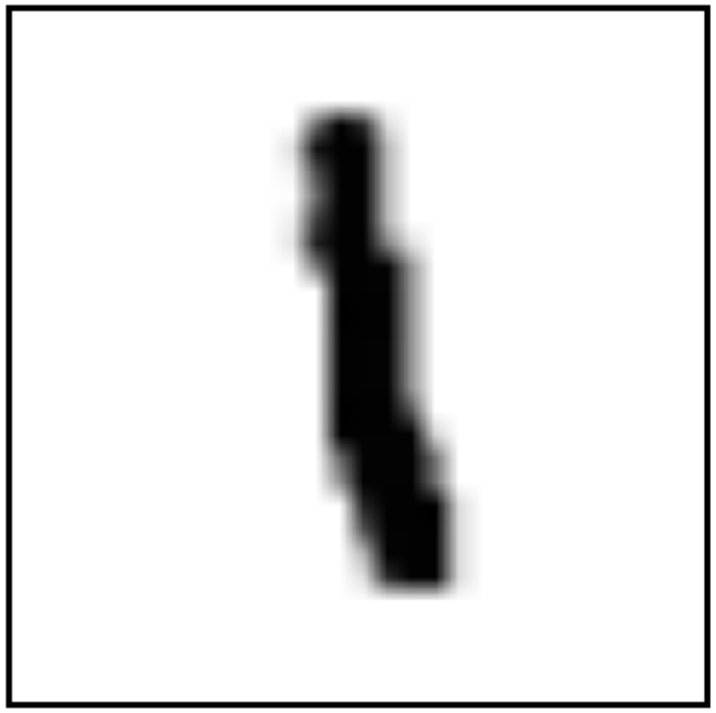
MNIST database 由兩種資料來源組成NIST‘s Special Database 3(SD-3)和 Special Database 1(SD-1)。
SD-3 品質比SD-1更乾淨更容易分類。

http://yann.lecun.com/exdb/mnist/

1. 手動下載 wget curl
2. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/mnist/input_data.py (即將廢棄)
3. https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data
4. …

Clean, Prepare, Manipulate Data

Input data



≈

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	.6	.8	0	0	0	0	0	0
0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	.5	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	1	.7	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	.9	1	.1	0	0	0	0
0	0	0	0	0	0	0	.3	1	.1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

每筆資料都是一張28×28的圖，左圖例子是
14×14縮小的例子

Training Dataset
SD-3:27500
SD-1:27500
Test Dataset
SD-3:5000
SD-1:5000
Validation data
SD-3:2500
SD-1:2500

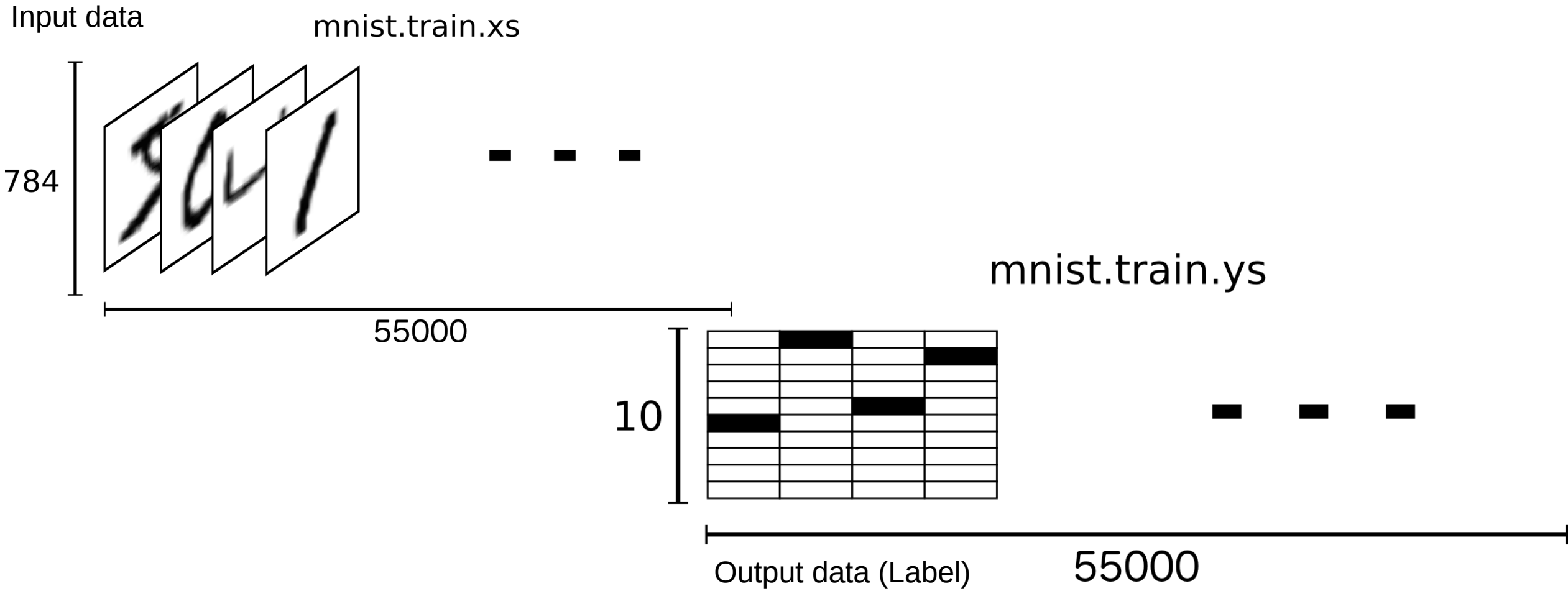
Clean, Prepare, Manipulate Data

Output data (Label)

- 每張圖的答案(Label)就是數字本身，例如0,1,2,…,9，不做特別處理的編碼稱為自然狀態碼。
- One-Hot Encoding又稱一位有效編碼，其方法是使用N位狀態暫存器來對N個狀態進行編碼，每個狀態都有它獨立的暫存器位，並且在任意時候，其中只有一位有效。例如：

自然狀態碼	一位有效編碼
0	[1,0,0,0,0,0,0,0,0,0]
1	[0,1,0,0,0,0,0,0,0,0]
…	…
9	[0,0,0,0,0,0,0,0,0,1]

Clean, Prepare, Manipulate Data



Clean, Prepare, Manipulate Data

Import Data, One-Hot Encoding

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("./data/", one_hot=True)
```

取用train set

```
mnist.train.images
mnist.train.labels
```

取用test set

```
mnist.test.images
mnist.test.labels
```

取用validation set

```
mnist.validation.images
mnist.validation.labels
```

Train Model

Setting training parameter

```
# Parameters
learning_rate = 0.1
num_steps = 500
batch_size = 128
display_step = 100
```

Setting network parameter

```
# Network Parameters
n_hidden_1 = 256 # 1st layer number of neurons
n_hidden_2 = 256 # 2nd layer number of neurons
num_input = 784  # MNIST data input (img shape: 28*28)
num_classes = 10 # MNIST total classes (0-9 digits)
```

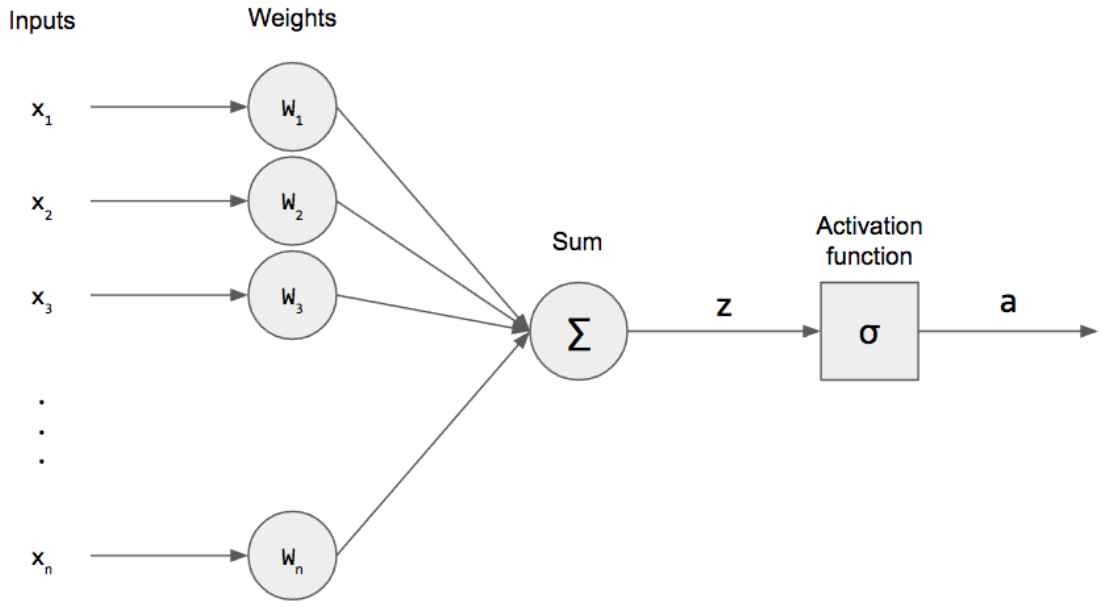
Define input & output

```
# https://www.tensorflow.org/api_docs/python/tf/placeholder
tf_x = tf.placeholder(tf.float32, [None, num_input])       # input
tf_y = tf.placeholder(tf.float32, [None, num_classes])     # label
```

Train Model

Define Graph

```
# Create model
def neural_net(x):
    # Hidden fully connected layer with 256 neurons
    layer_1 = tf.add(tf.matmul(x, weights['h1']), biases['b1'])
    # Hidden fully connected layer with 256 neurons
    layer_2 = tf.add(tf.matmul(layer_1, weights['h2']), biases['b2'])
    # Output fully connected layer with a neuron for each class
    out_layer = tf.matmul(layer_2, weights['out']) + biases['out']
    return out_layer
```



Train Model

```
# Construct model
logits = neural_net(X)
# Use Softmax Regression
prediction = tf.nn.softmax(logits)
```

```
# Define loss and optimizer

#https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits

loss_op = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=logits,
labels=Y))

optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)

train_op = optimizer.minimize(loss_op)
```

Train Model

Evaluate model

```
# https://www.tensorflow.org/api_docs/python/tf/math/argmax
correct_pred = tf.equal(tf.argmax(prediction, 1), tf.argmax(Y, 1))
# https://www.tensorflow.org/api_docs/python/tf/dtypes/cast
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
```

`tf.argmax:` 找出最大值的位置
tf.equal : 比較是否相等，回傳A Tensor of type bool with the same size as that of x or y.
tf.cast : 將bool轉成tf.float32
tf.reduce_mean : Computes the mean of elements

Train Model

```
# Initialize the variables (i.e. assign their default value)
init = tf.global_variables_initializer()


# 'Saver' op to save and restore all the variables
saver = tf.train.Saver()
```

你訓練完一個網路，儲存這個網路成為”模型”，可以拿來使用或直接用於其他平台的 deploy。
tensorflow 模型包括：已訓練並優化的權重引數，網路結構和 graph。存在兩個部分meta graph 和checkpoint file

checkpoint
$file.meta
$file.data-00000-of-00001
$file.index

Train Model

```
batch_x, batch_y = mnist.train.next_batch(batch_size)
# Run optimization op (backprop)
sess.run(train_op, feed_dict={X: batch_x, Y: batch_y})
```

- Batch_Size 是機器學習中一個重要的參數，Batch 的選擇會決定梯度下降的方向。
- 如果dataset比較小，那麼可以採用full dataset的方式。優點就是full dataset的方向更能代表母體，可以準確地找到極值方向。
- 另外一個極端是一次只載入一個數據。每個樣本的修正方向以各自樣本的梯度方向修正，批次愈小，對於方向的估計愈不準確。
- 找一個適中的 Batch_Size 值就很重要。
- 如果dataset夠多，那麼用一半的data算出來的梯度與用full dataset幾乎一樣的。在合理範圍內，增大 Batch_Size ，(類似convergence test)。

Train Model

```
with tf.Session() as sess:


……
# Save model weights to disk
    save_path = saver.save(sess, "model_old")
    print("Model saved in file: %s" % save_path)
# Load model weights from disk
    saver.restore(sess, "model_old")
    print("Model restored from file: %s" % save_path)
```

Use Model

```
# Running a test dataset by loading the model saved earlier
with tf.Session() as sess:
    # Run the initializer
    sess.run(init)

    saver.restore(sess, "model_old")
    print("Model restored from file: %s" % save_path)

    # Calculate the answer for the image
    print("Answer:", sess.run(ans, feed_dict={X: mnist.validation.images[0:1]}))
```

使用tensorflow的keras api來讀資料和網路

Get Data

```
# https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape)
```

注意這裡的x_train…都是numpy array，keras會幫你處理產生tensor的部分。

Train Model

```
# https://www.tensorflow.org/api_docs/python/tf/keras/models/Sequential
  model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),                    # Flattens the
input.
  tf.keras.layers.Dense(128, activation=tf.nn.relu),
#https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
  tf.keras.layers.Dense(64, activation=tf.nn.relu),
# https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

Train Model

- tf.keras.Sequential
  - Linear stack of layers.
- 支援兩種建立方式
  - 把layer用list方式填入Sequential
    - 例如model = Sequential([layer1,layer2,…])
  - 用add方法加入model
    - model = Sequential()
    - model.add()

Train Model

Configures the model for training

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

Trains the model for a fixed number of epochs

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

Returns the loss value & metrics values for the model in test mode.

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

Save / Load Model

Saves / Load the model to Tensorflow SavedModel or a single HDF5 file.

```
tf.keras.models.save_model( model, ".\model" )

tf.keras.models.load_model(".\model")
```