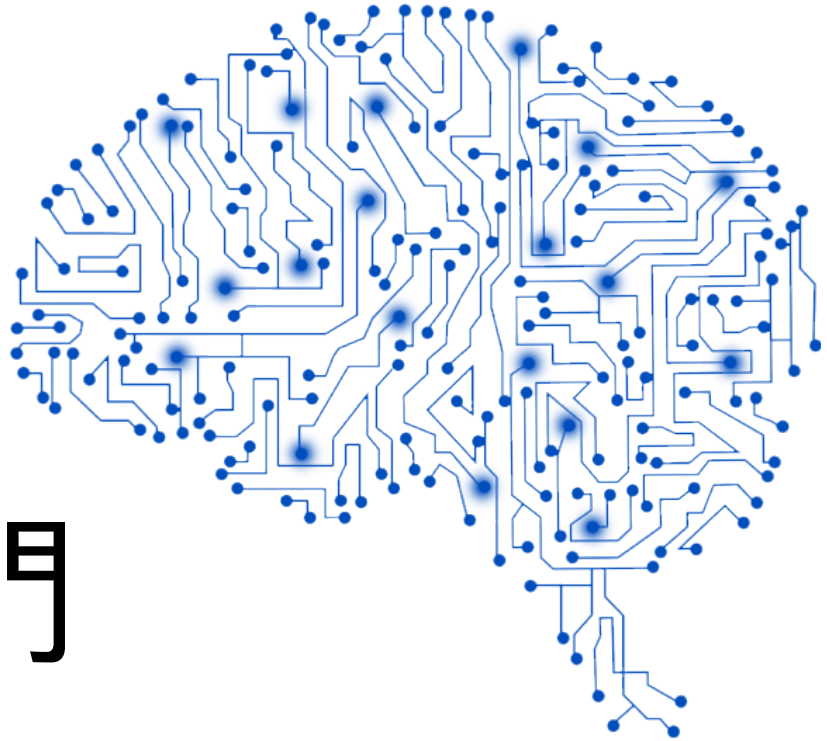
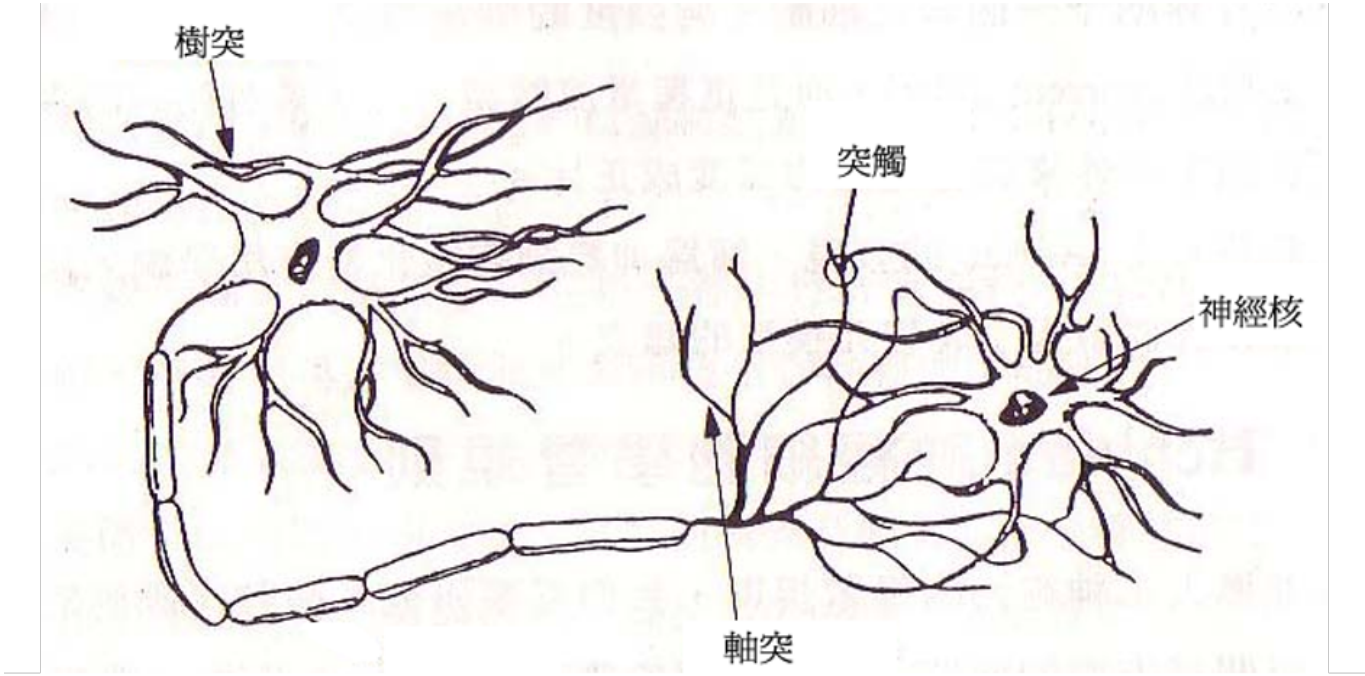


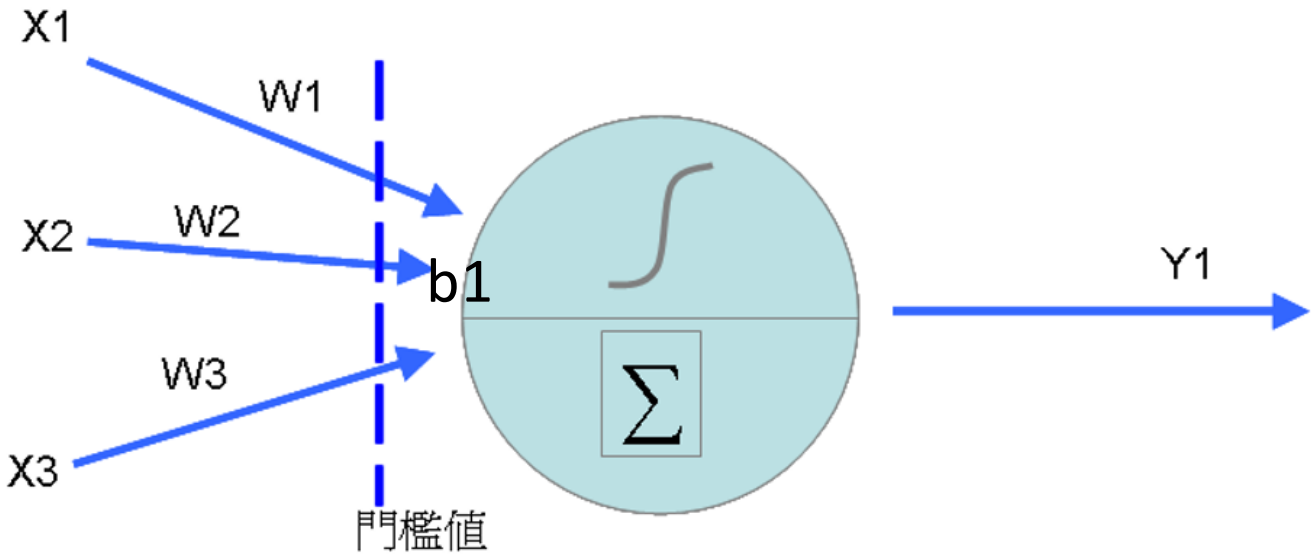
深度學習入門
Keras

DE Sharing
何信賢

深度學習簡介

項目	說明
Input X	模仿輸入神經元
Output Y	模仿輸出神經元
Weight W	模仿軸突
Bias b	模仿突觸
Activation Function	模仿神經傳導的運作方式



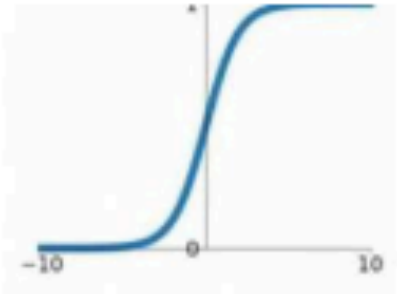


Y1=sigmoid(X1*W1+X2*W2+X3*W3+b1)

激活函數(Activation Function)

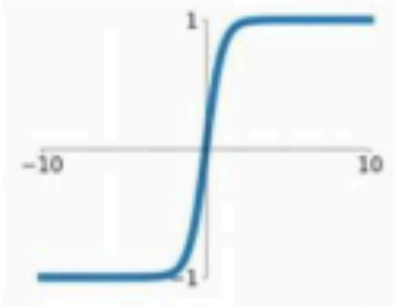
Sigmoid

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



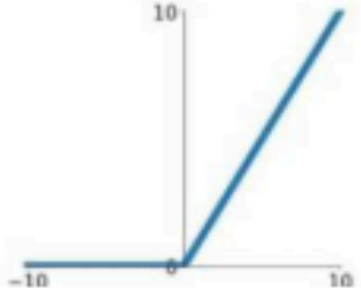
tanh

$$\tanh(x)$$



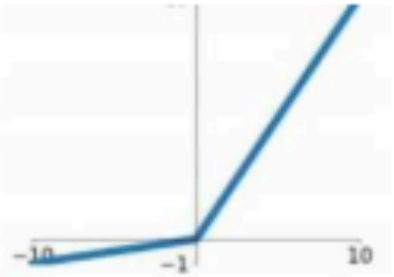
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

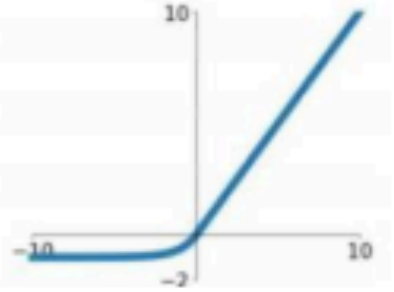


Maxout

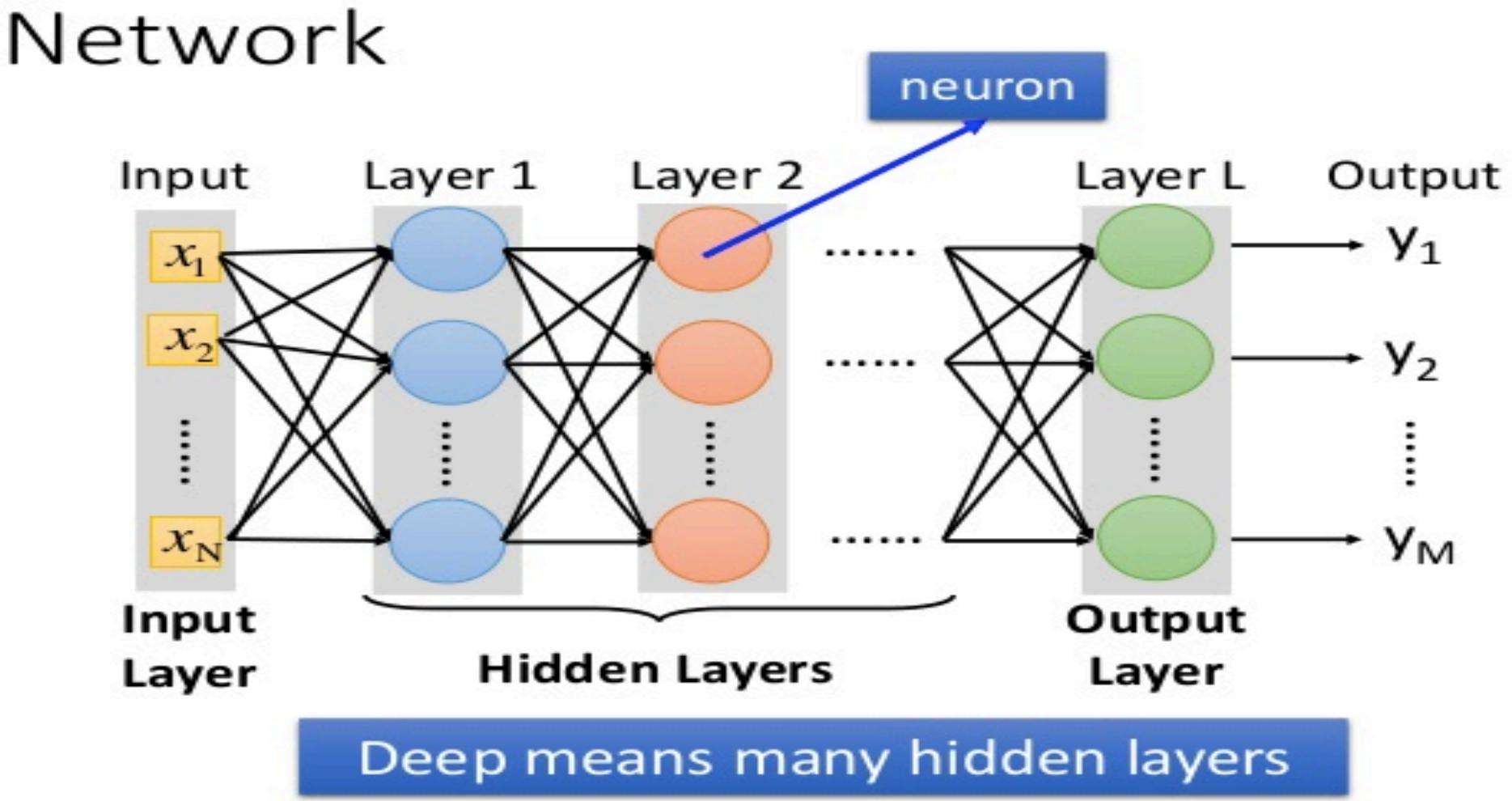
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

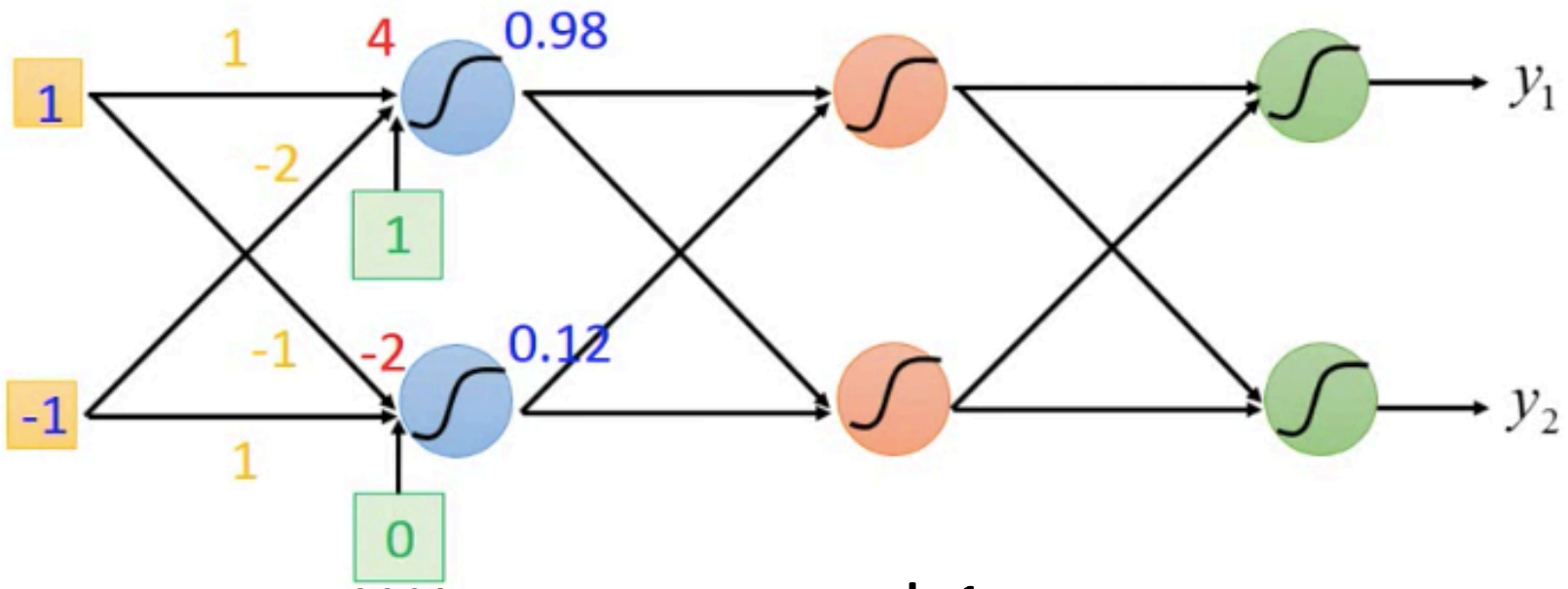
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



多層感知器模型


Network
neuron
Input
Layer 1
Layer 2
Layer L
Output
x_1
x_2
x_N
y_1
y_2
y_M
Input Layer
Hidden Layers
Output Layer
Deep means many hidden layers

Matrix Operation

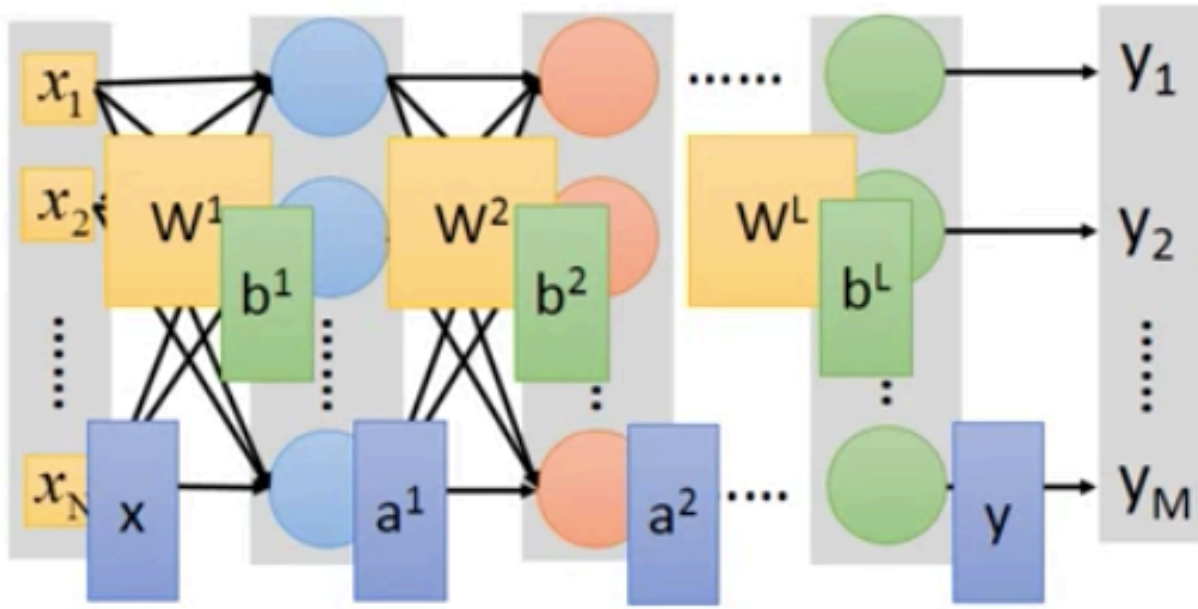


W1 X b1

$$\sigma\left(\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

$$\begin{bmatrix}4\\ -2\end{bmatrix}$$

Neural Network

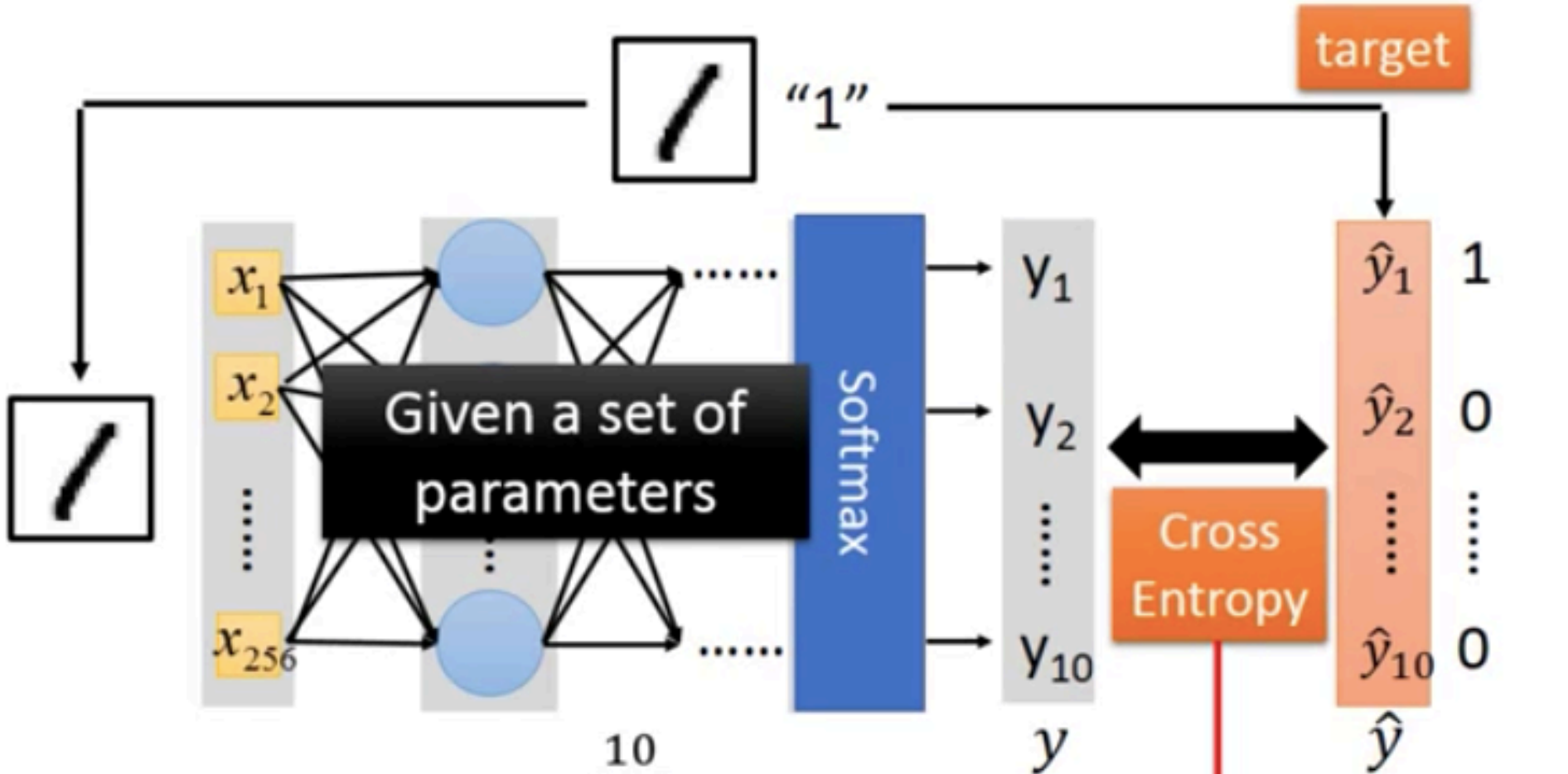


$$y = f(x)$$

Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

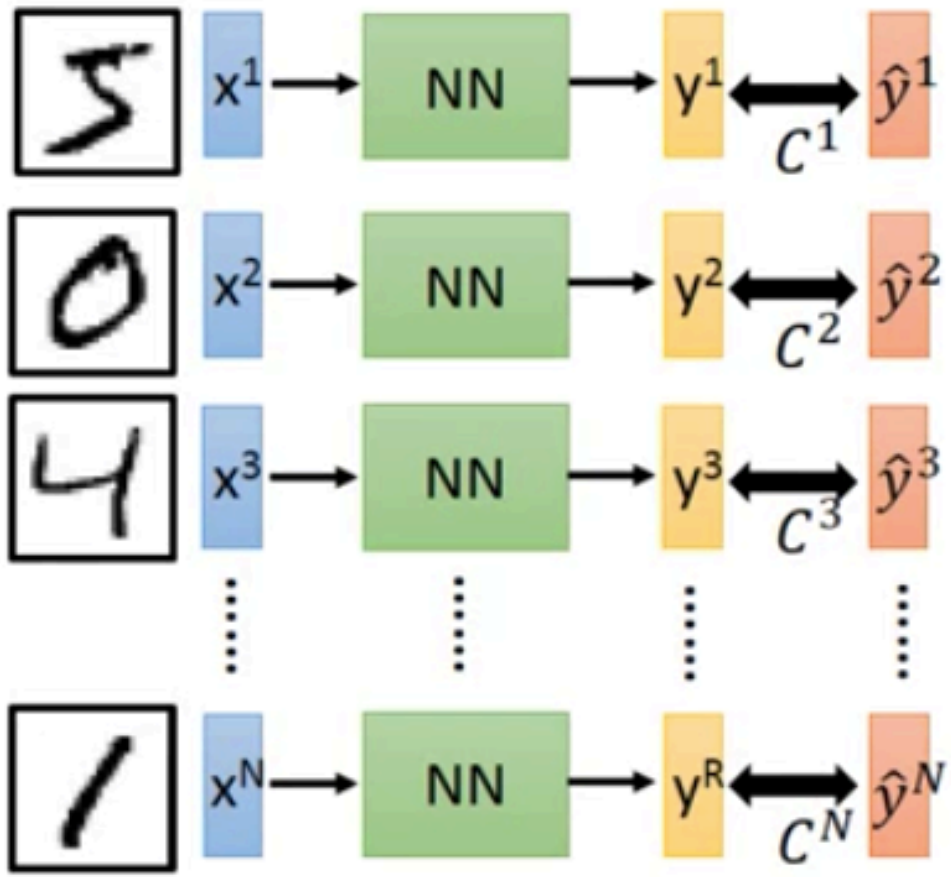
Loss for an Example



$$C(y, \hat{y}) = -\sum_{i=1}^{10} \hat{y}_i \ln y_i$$

Total Loss

For all training data ...



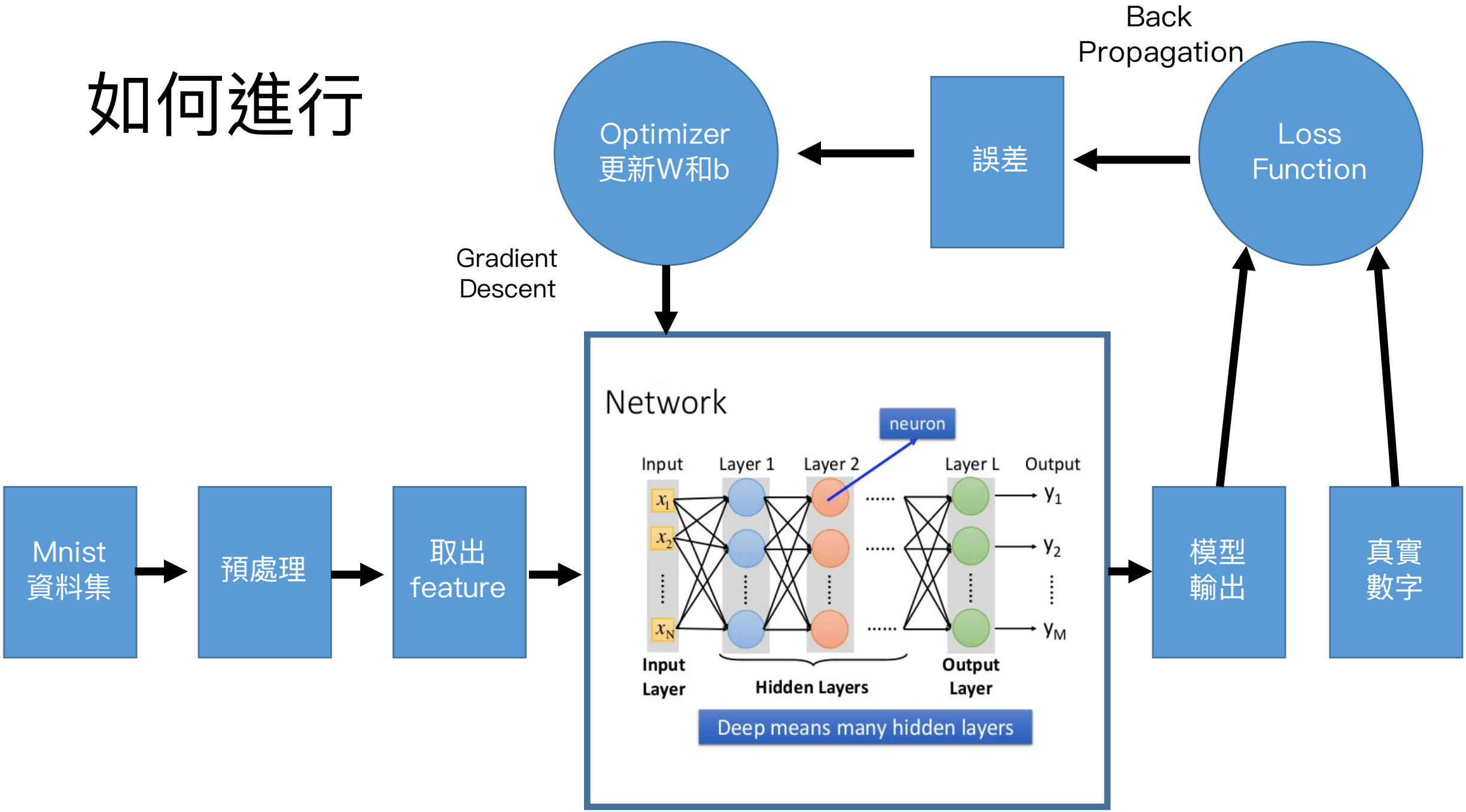
Total Loss:

$$L = \sum_{n=1}^{N} C^n$$

Find ***a function in function set*** that minimizes total loss L

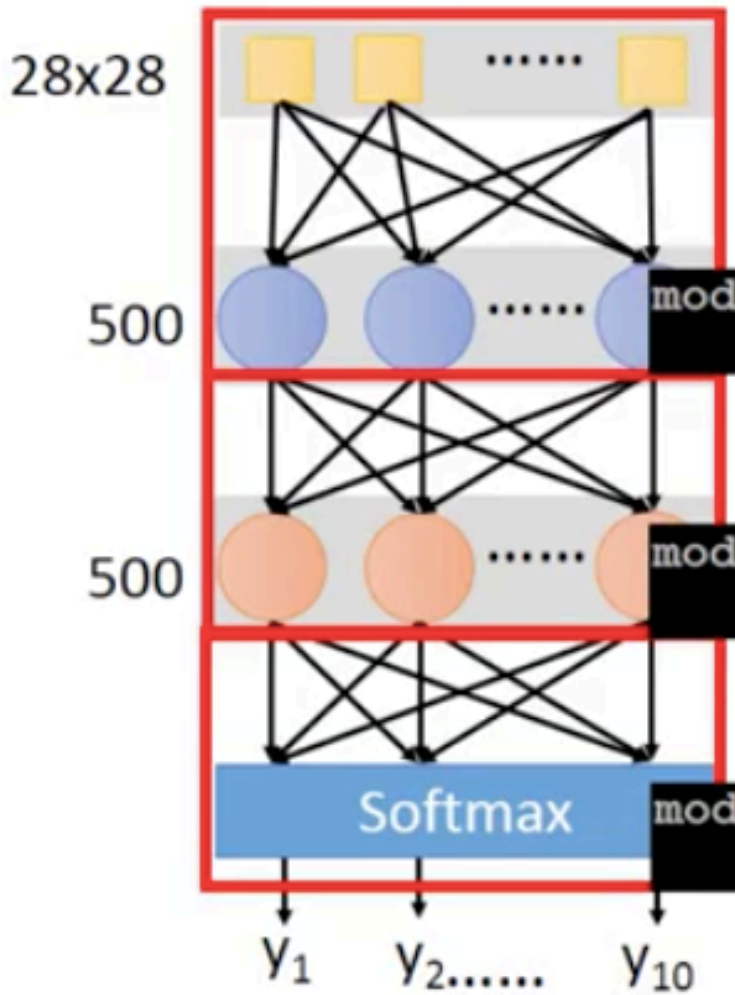
Find ***the network parameters*** θ^* that minimize total loss L

如何進行


Back
Propagation
Optimizer
更新W和b
誤差
Loss
Function
Gradient
Descent
Network
neuron
Input
Layer 1
Layer 2
Layer L
Output
x_1
x_2
x_N
y_1
y_2
y_M
Input
Layer
Hidden Layers
Output
Layer
Deep means many hidden layers
Mnist
資料集
預處理
取出
feature
模型
輸出
真實
數字

Let’s Demo

Keras: Building a Network




```
model = Sequential()
```

```
model.add(Dense(input_dim=28*28,units=500,
                activation='relu'))
```

softplus, softsign, sigmoid,
tanh, hard_sigmoid, linear …

```
model.add(Dense(units=500,
                activation='relu'))
```

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

Configuration

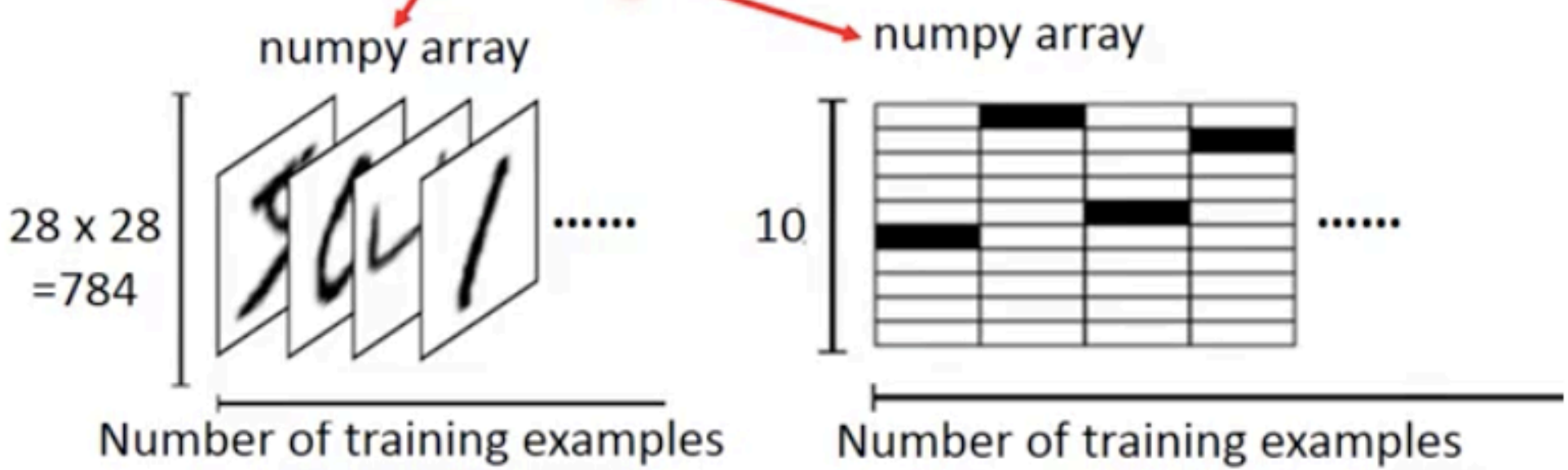
Several alternatives: https://keras.io/objectives/

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

SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam

Pick the best function

```
model.fit(x_train,y_train,batch_size=100,epochs=20)
```



Keras

Save and load models

http://keras.io/getting-started/faq/#how-can-i-save-a-keras-model

How to use the neural network (testing):

case 1:

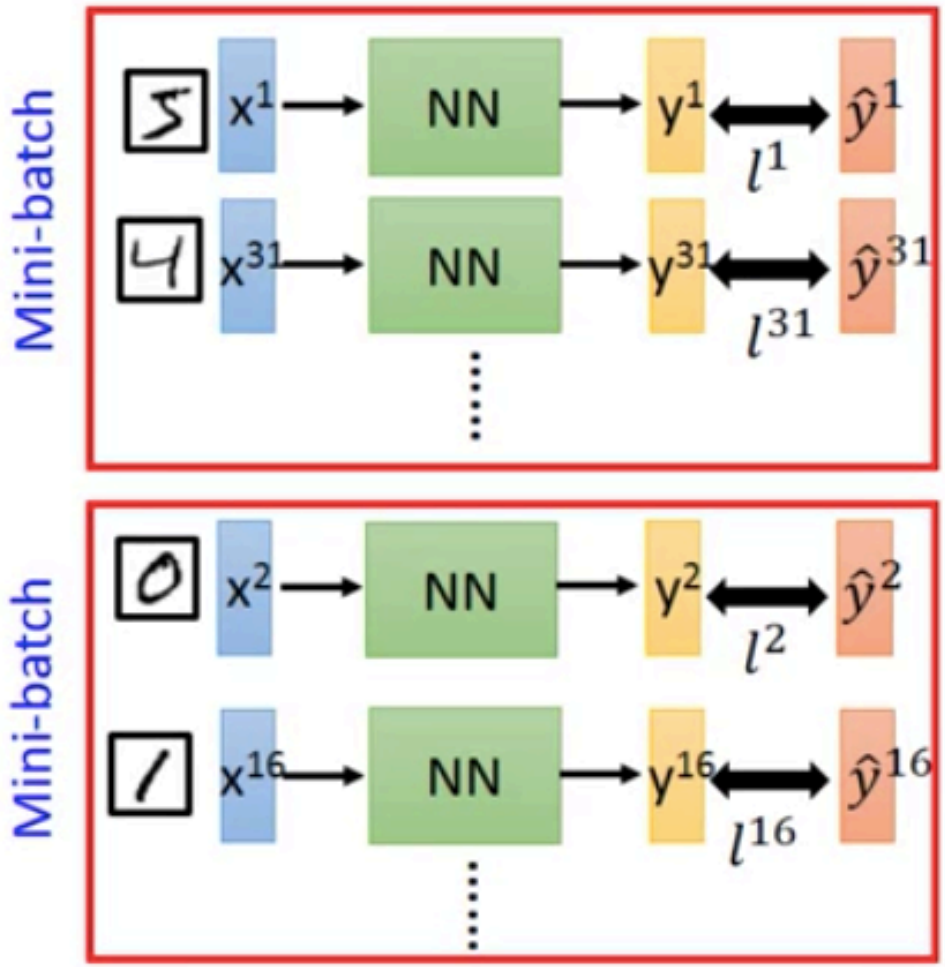
```
score = model.evaluate(x_test,y_test)
print('Total loss on Testing Set:', score[0])
print('Accuracy of Testing Set:', score[1])
```

case 2:

```
result = model.predict(x_test)
```

We do not really minimize total loss!

Mini-batch



- Randomly initialize network parameters
- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  Update parameters once
- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

  Update parameters once

  ⋮
- Until all mini-batches have been picked

one epoch

Repeat the above process

Speed

- Smaller batch size means more updates in one epoch
  - E.g. 50000 examples
  - batch size = 1, 50000 updates in one epoch
  - batch size = 10, 5000 updates in one epoch

166s 1 epoch

17s 10 epoch