“Hello world”
# of deep learning

Keras

If you want to learn theano:

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Theano%20DNN.ecm.mp4/index.html

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RNN%20training%20(v6).ecm.mp4/index.html

TensorFlow or theano

Very flexible

Need some effort to learn

Interface of TensorFlow or Theano

keras

Easy to learn and use

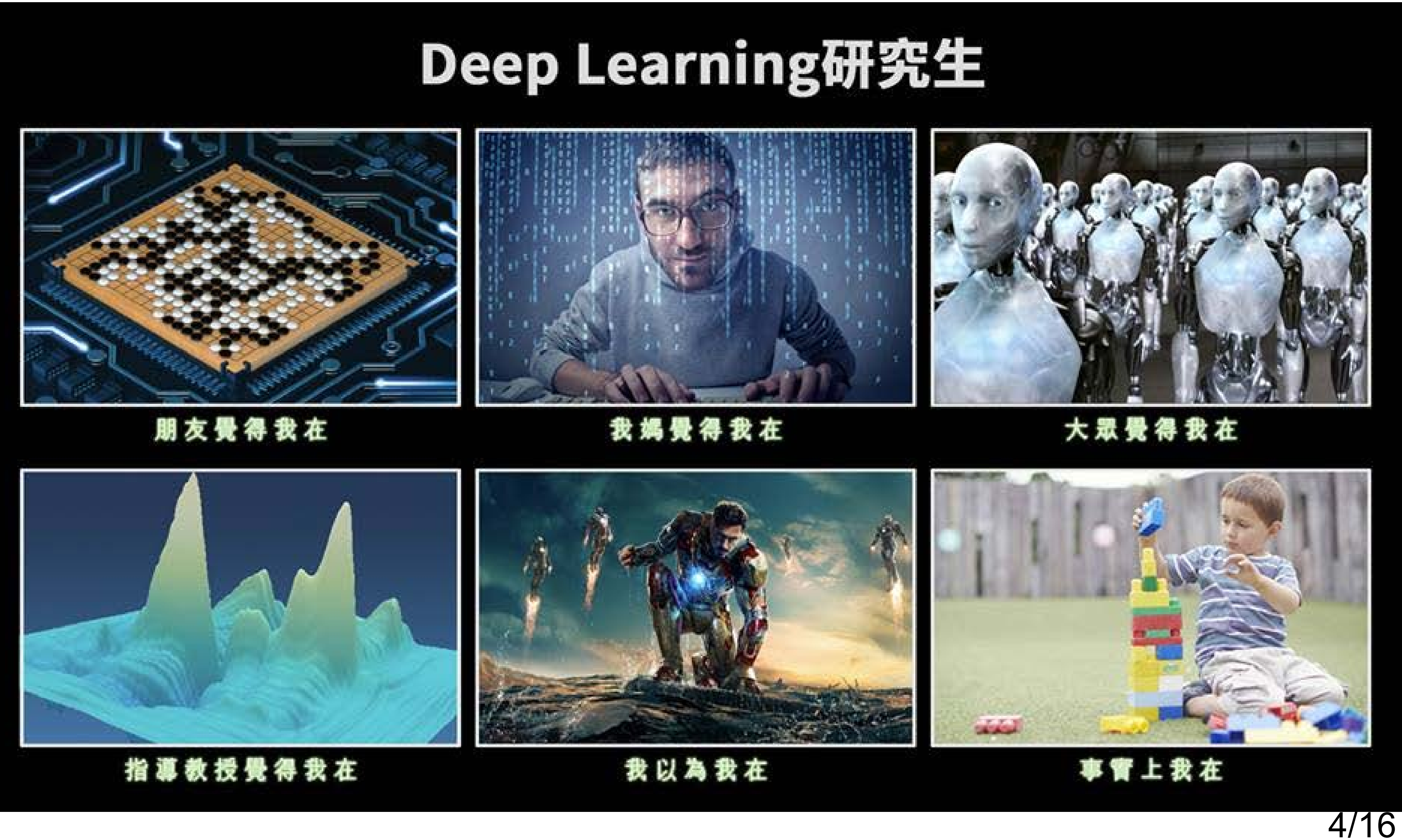
(still have some flexibility)

You can modify it if you can write TensorFlow or Theano

Keras

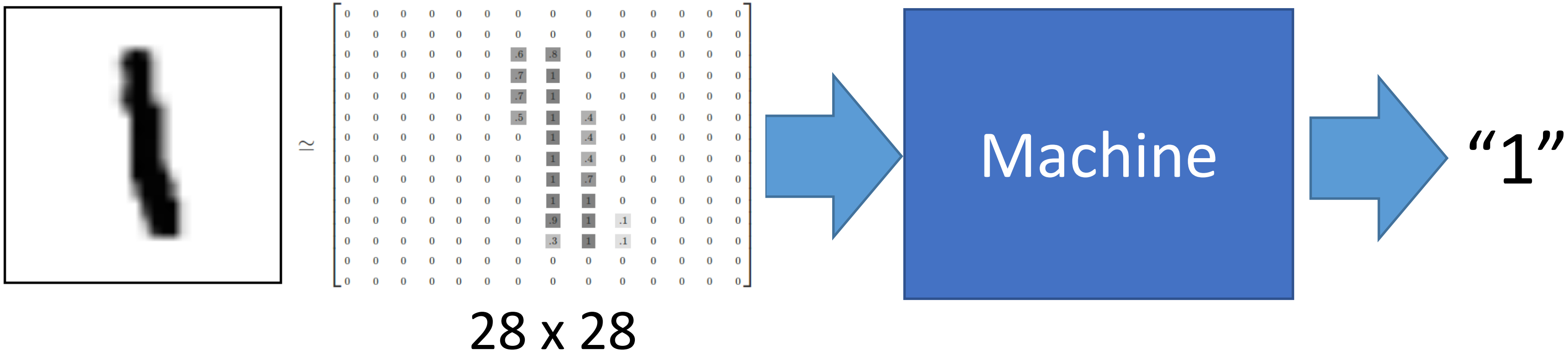
- François Chollet is the author of Keras.
  - He currently works for Google as a deep learning engineer and researcher.
- Keras means *horn* in Greek
- Documentation: http://keras.io/
- Example: https://github.com/fchollet/keras/tree/master/examples

使用 Keras 心得


Deep Learning研究生
朋友覺得我在
我媽覺得我在
大眾覺得我在
指導教授覺得我在
我以為我在
事實上我在

"Hello world"

- Handwriting Digit Recognition



$$\simeq \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&.6&.8&0&0&0&0&0&0 \\ 0&0&0&0&0&0&.7&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&.7&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&.5&1&.4&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&.4&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&.4&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&.7&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&.9&1&.1&0&0&0&0 \\ 0&0&0&0&0&0&0&.3&1&.1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

28 x 28

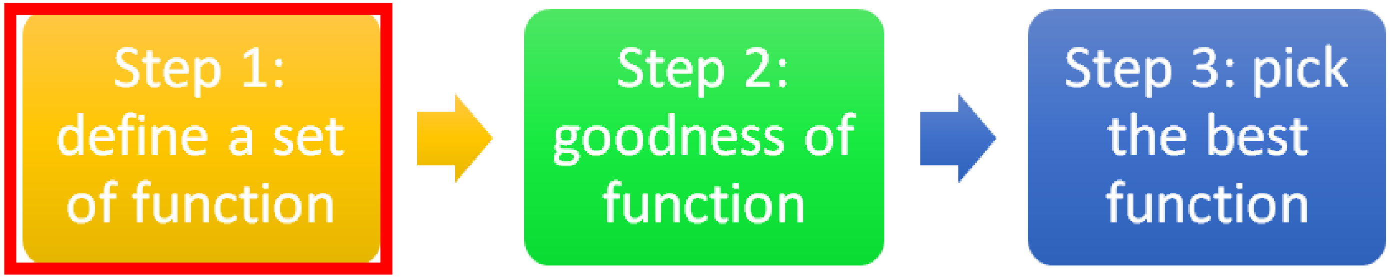
Machine

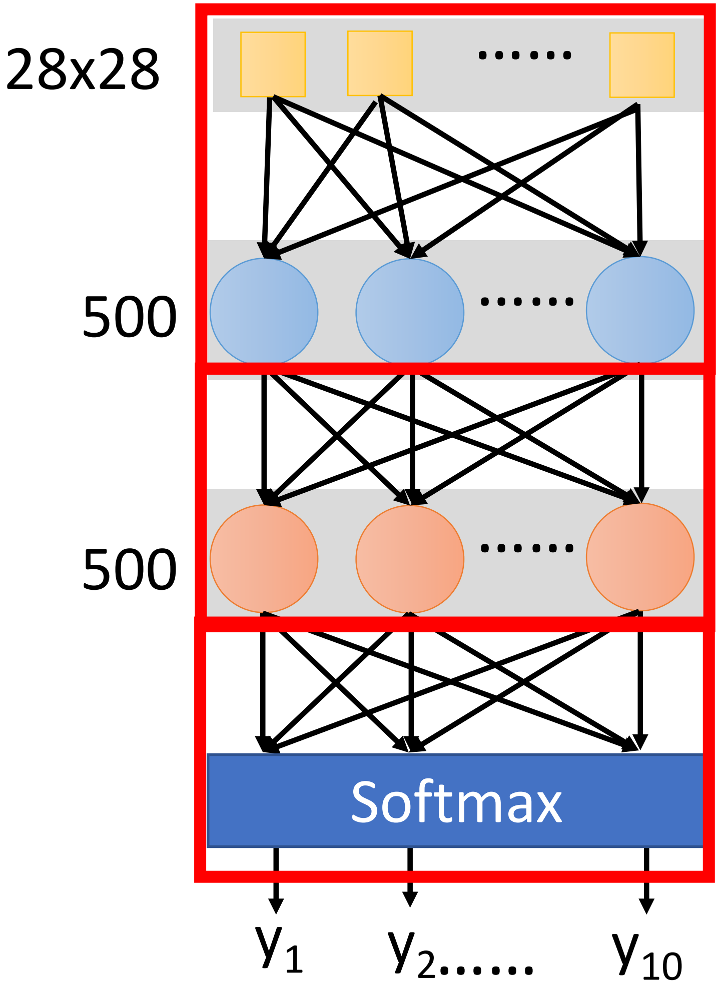
"1"

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

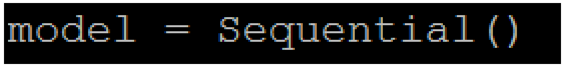
Keras provides data sets loading function: http://keras.io/datasets/

Keras


Step 1: define a set of function
Step 2: goodness of function
Step 3: pick the best function

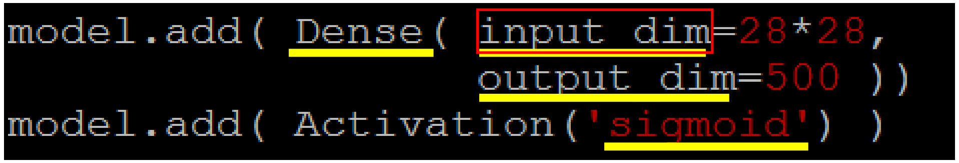


(Because after the first layer, the layer's input dimension is equal to the pervious layer's output dimension.)



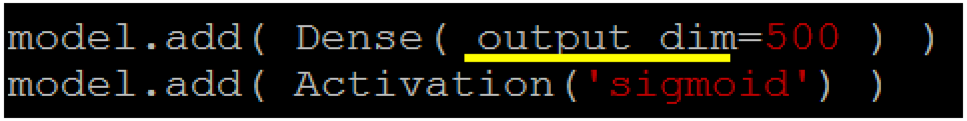
```
model = Sequential()
```

Only the first layer need to set the dimension of input.

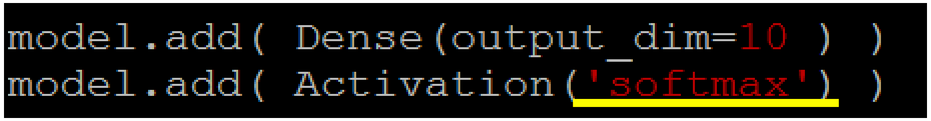


```
model.add( Dense( input_dim=28*28,
                  output_dim=500 ))
model.add( Activation('sigmoid') )
```

softplus, softsign, relu, tanh, hard_sigmoid, linear



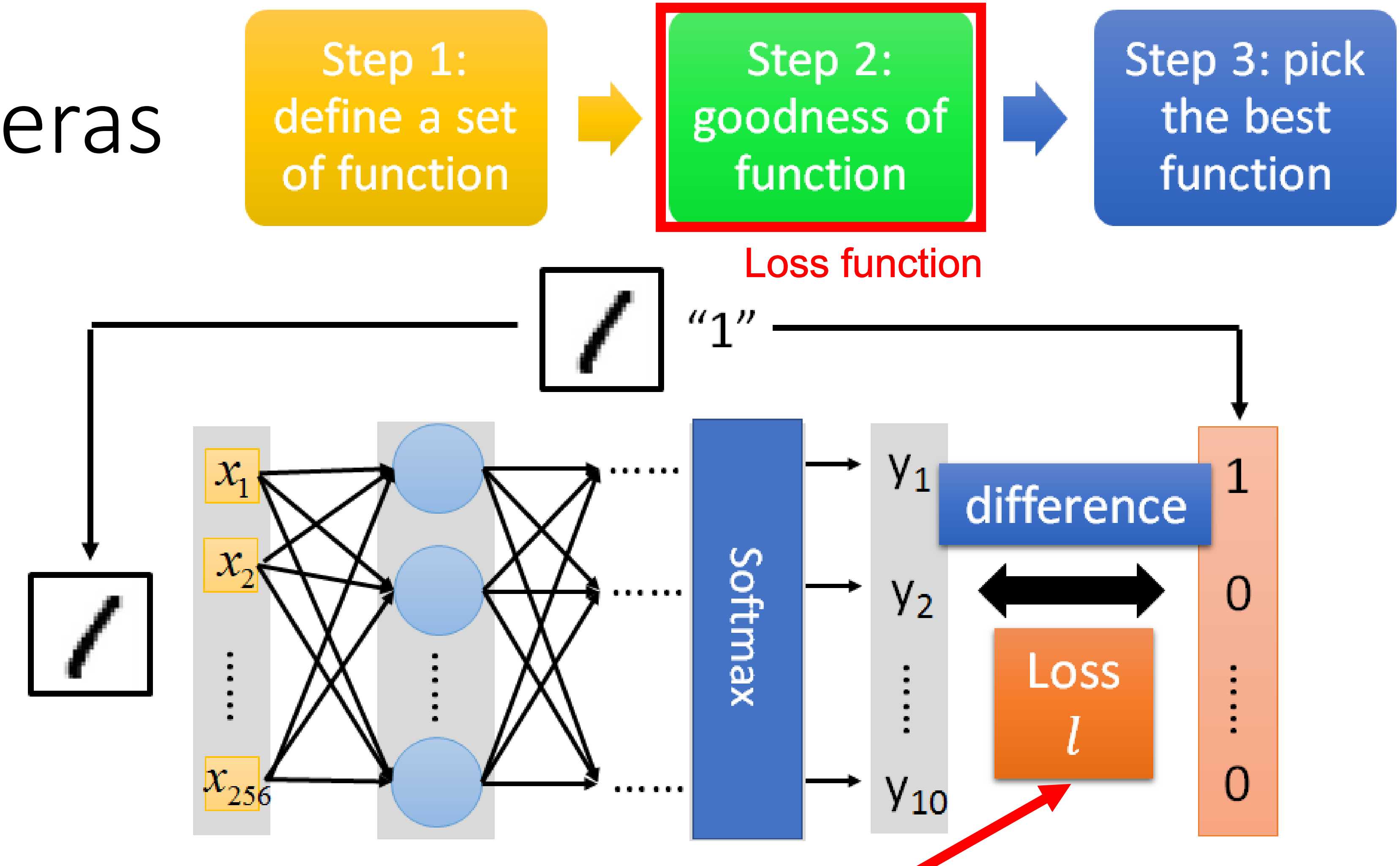
```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```



```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

Normally, the last layer's activation function is softmax.

Keras

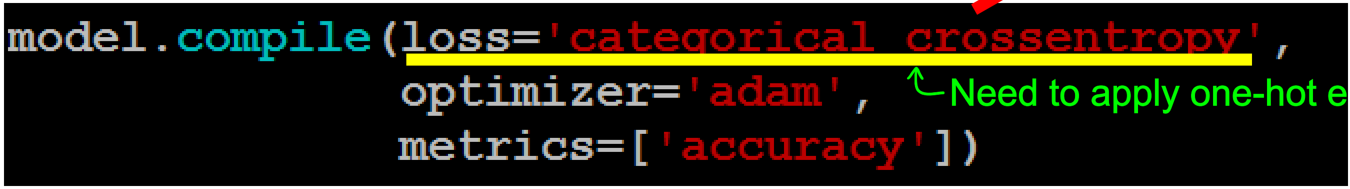


Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Loss function

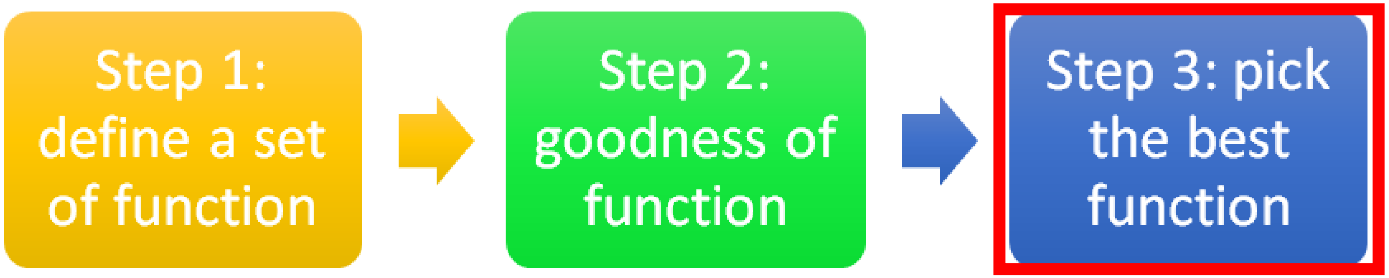


```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

↰Need to apply one-hot encoding on y.

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

Keras



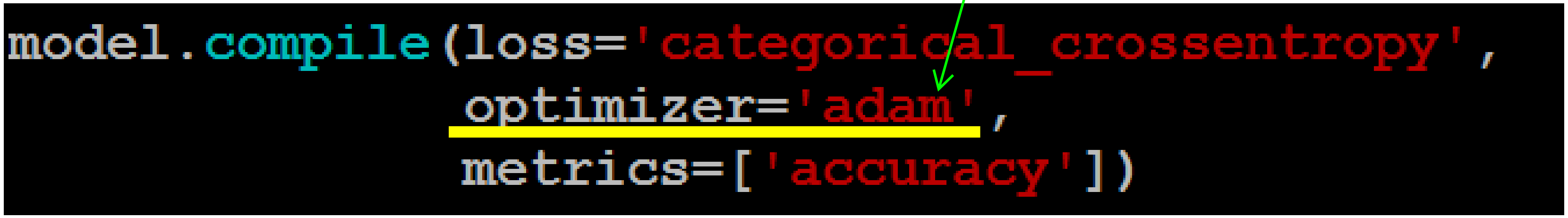
Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Set the method of how we update the function.

Step 3.1: Configuration

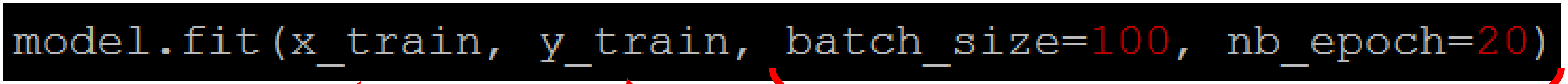


```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

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

Step 3.2: Find the optimal network parameters

Train the model



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

Training data (Images)

Labels (digits)

In the following slides

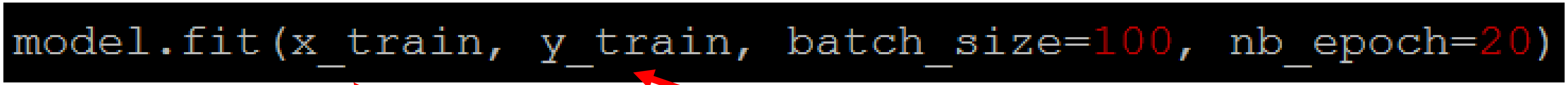
Keras

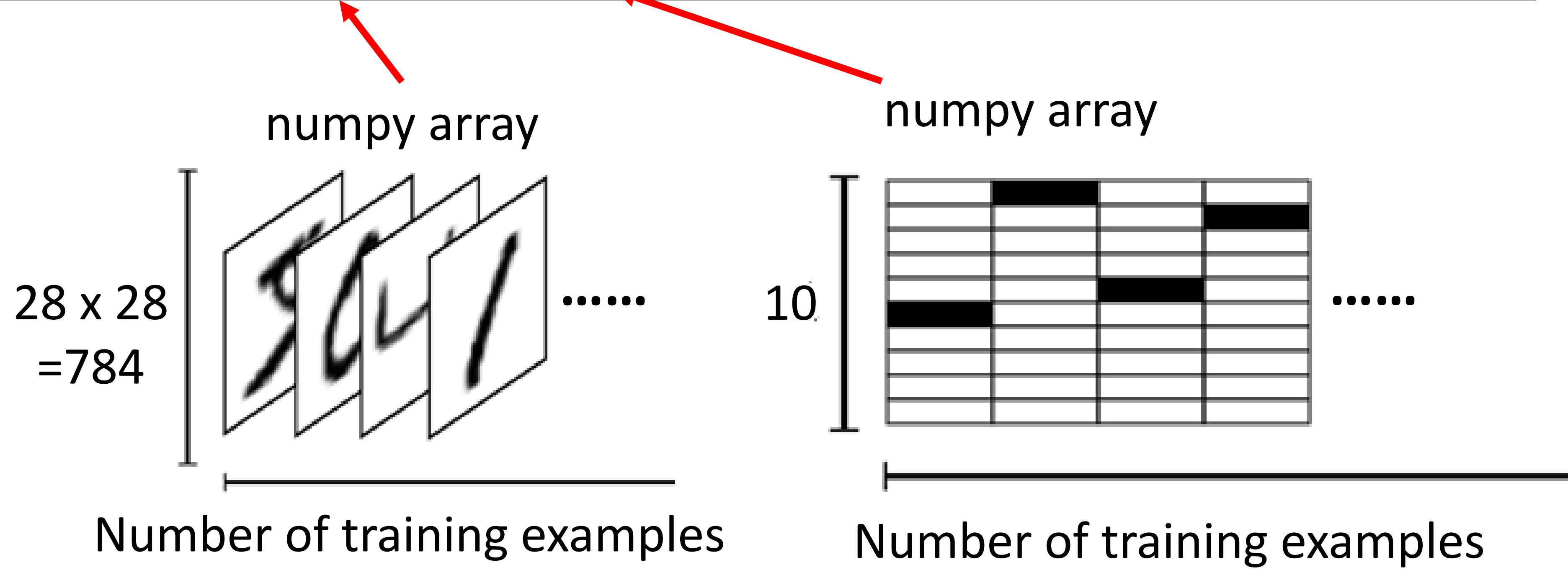
Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Step 3.2: Find the optimal network parameters


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

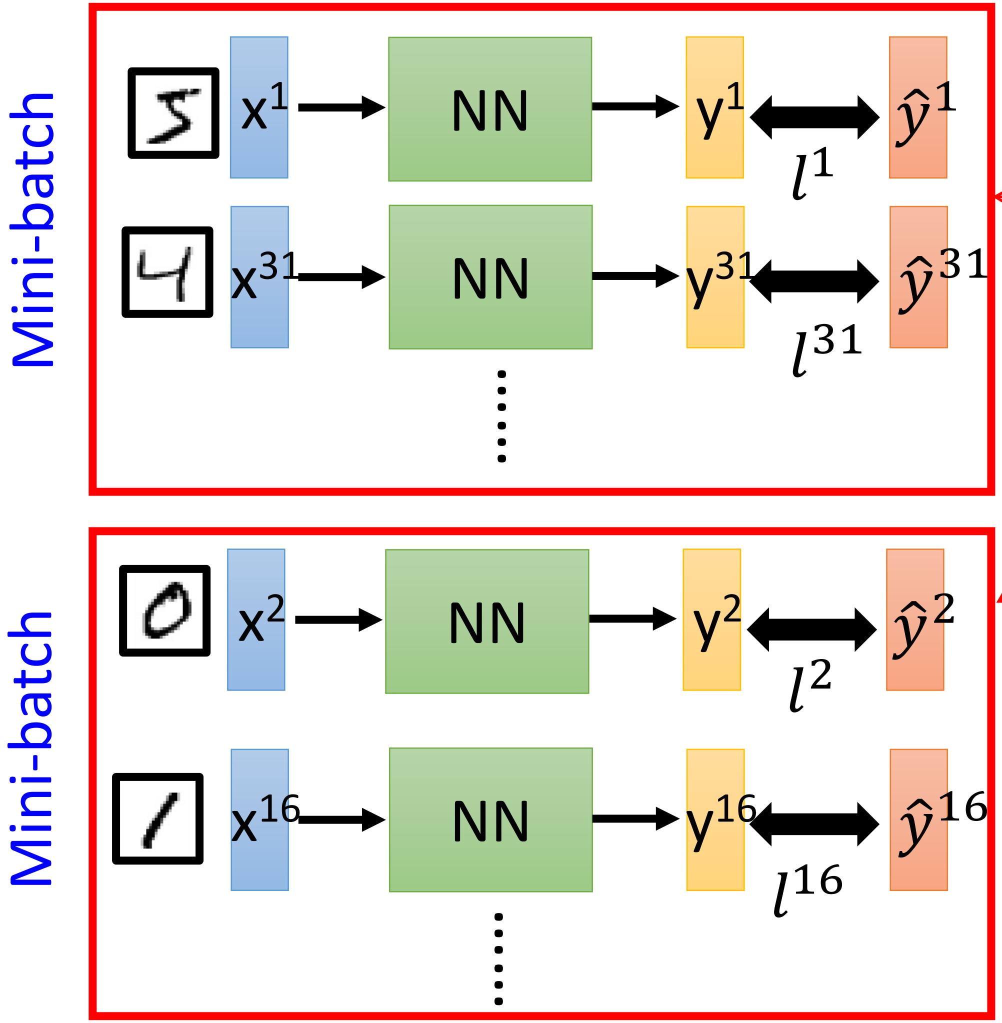


https://www.tensorflow.org/versions/r0.8/tutorials/mnist/beginners/index.html

We do not really minimize total loss!

Mini-batch

Update the model per "batch" instead of per "all data".



- 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

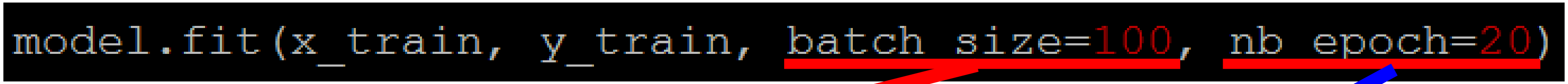
Have processed all data

Repeat the above process

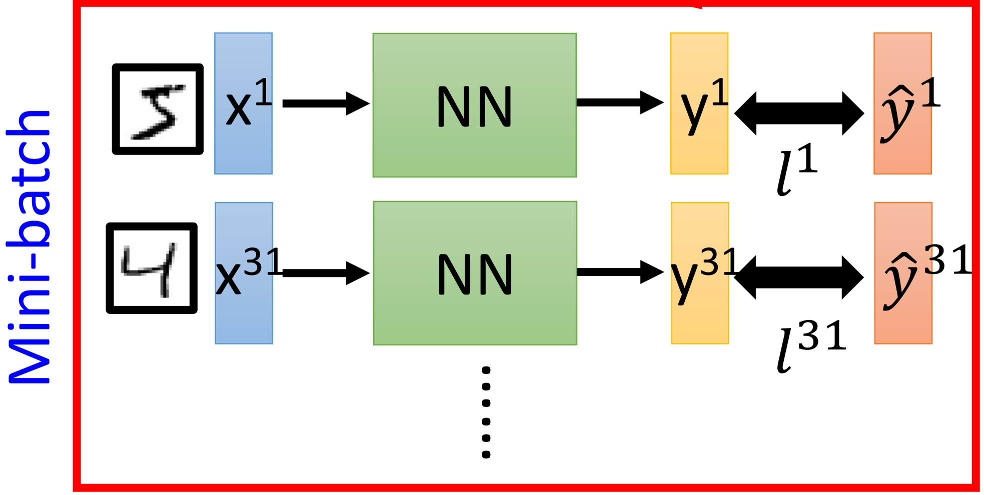
For each epoch, we will update the model (#examples/batch size) times.

Mini-batch

Batch size influences both ***speed*** and ***performance***. You have to tune it.



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



100 examples in a mini-batch

Batch size = 1 ➡

Stochastic gradient descent

- 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

  ⋮

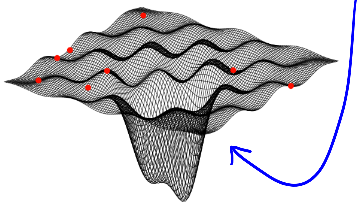
- Until all mini-batches have been picked

one epoch

Repeat 20 times

Two reasons that we shouldn't set the batch size too big:
1. The hardware limitations of GPU. (If the batch size is too big, the computation time for one batch will no longer stay the same.)
2. For DL, if we use full-batch, we will stuck at some local minima in the first few updates. (We need the randomness from mini-batch.)

Speed

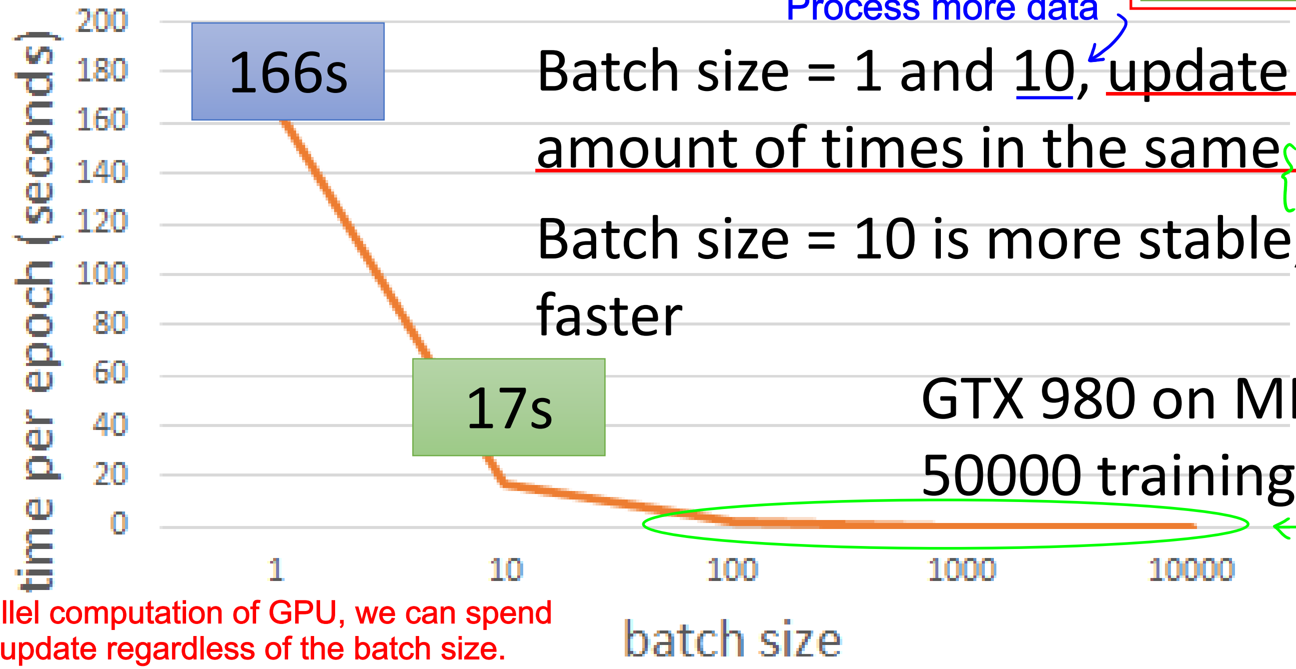


∵ Error surface looks like this.

Very large batch size can yield worse performance

- 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

For one epoch	In the same amount of time
166s	1 epoch
17s	10 epoch



Process more data

Batch size = 1 and 10, update the same amount of times in the same period.

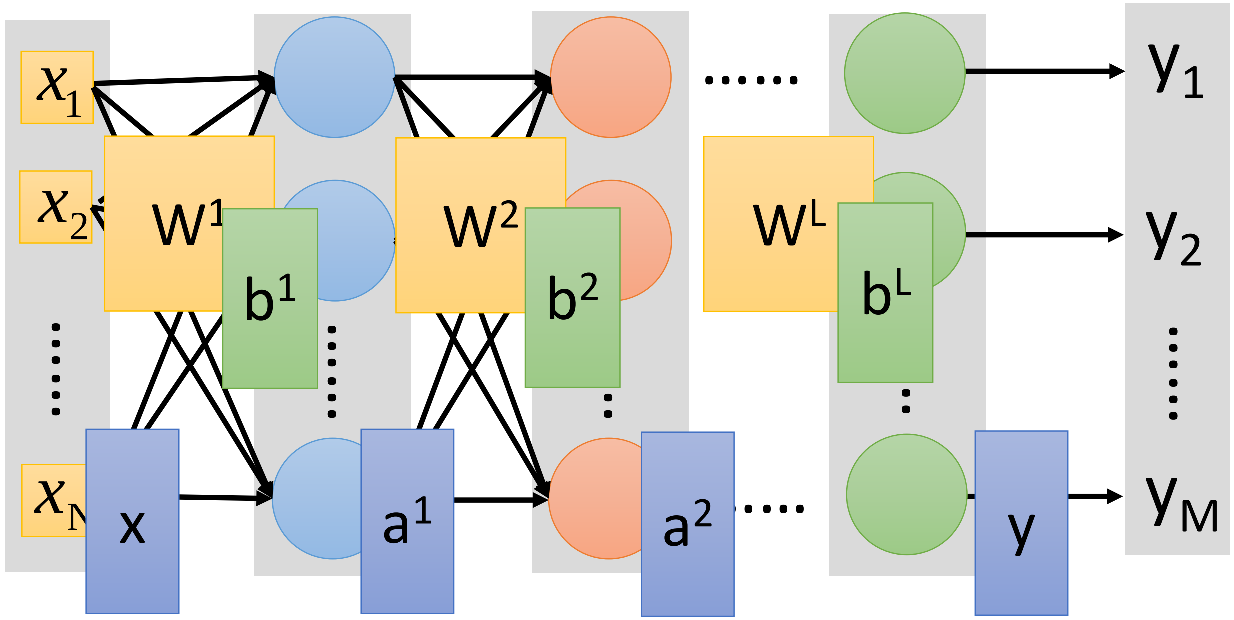
time

Batch size = 10 is more stable, converge faster

GTX 980 on MNIST with 50000 training examples

Thanks to the parallel computation of GPU, we can spend the same time per update regardless of the batch size.
(If batch size is not too big.) (Details on the next few pages.)

Speed - Matrix Operation



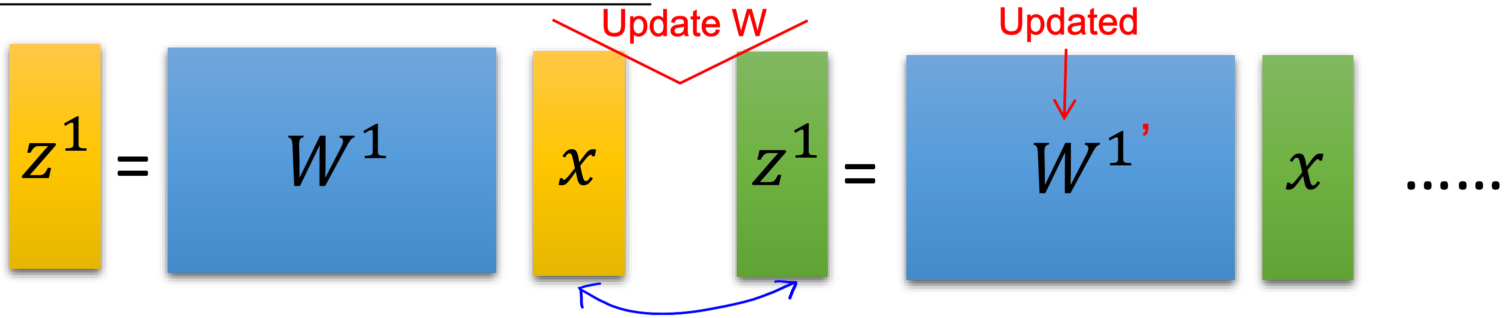
$$y = f(x)$$ Forward pass (Backward pass is similar)

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

Speed - Matrix Operation

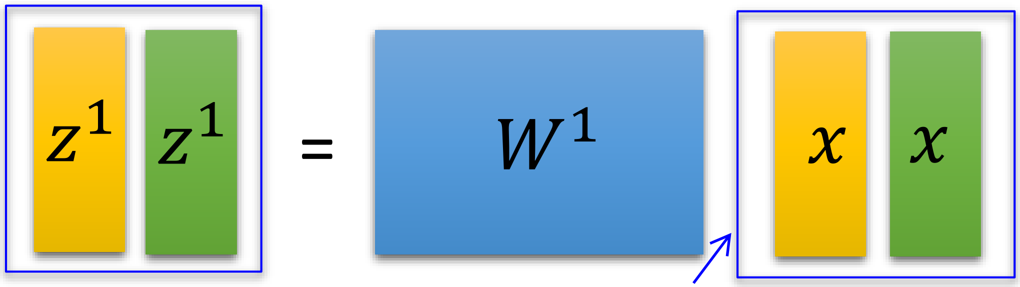
- Why mini-batch is faster than stochastic gradient descent? For stochastic gradient descent, we must wait for the previous update before calculating the next training example.

Stochastic Gradient Descent



Couldn't combine into one single operation.

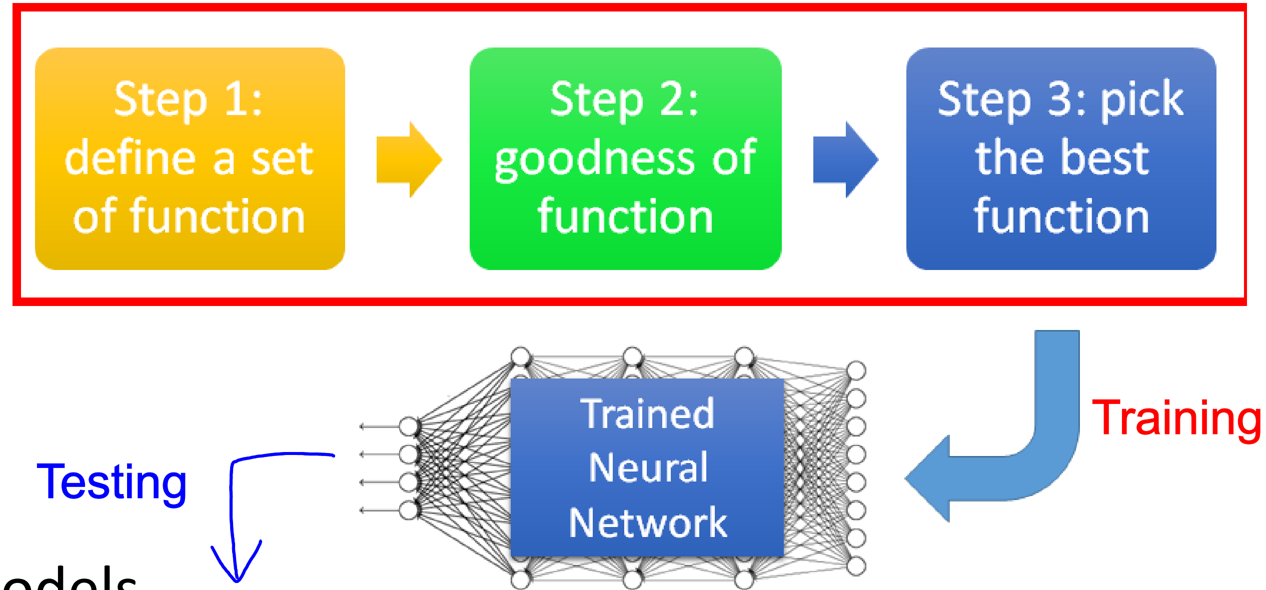
Mini-batch



Practically, which one is faster? Mini-batch

No matter how many elements in this matrix, we can use the same amount of time to solve it.
(If it doesn't exceed the hardware limitation of GPU.)

Keras


Step 1: define a set of function
Step 2: goodness of function
Step 3: pick the best function
Training
Trained Neural Network
Testing

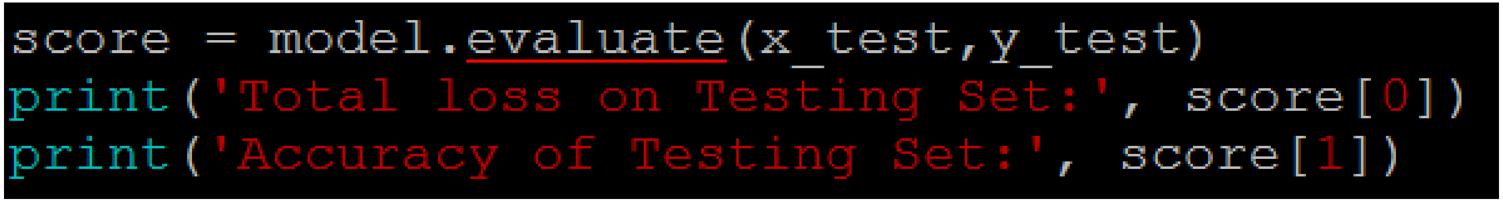
Save and load models

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

How to use the neural network (testing):

We have the label of the testing data: (predict + calculate the loss)

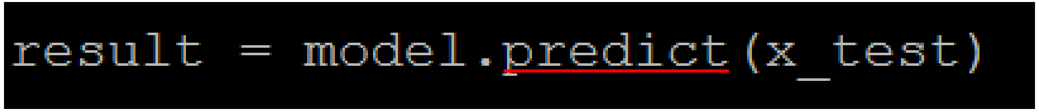
case 1:



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

We don't have the label of the testing data: (predict)

case 2:



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

Keras

- Using GPU to speed training
  - Way 1
    - THEANO_FLAGS=device=gpu0 python YourCode.py
  - Way 2 (in your code)
    - import os
    - os.environ["THEANO_FLAGS"] = "device=gpu0"