

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



or theano

Very flexible

Need some effort to learn

Interface of
TensorFlow or
Theano



keras

Easy to learn and use

(still have some flexibility)

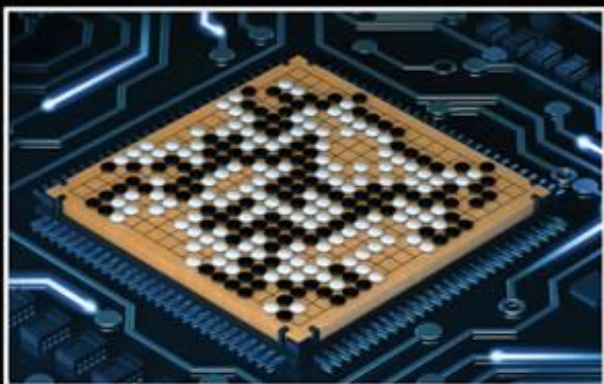
You can modify it if you can write
TensorFlow or Theano

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



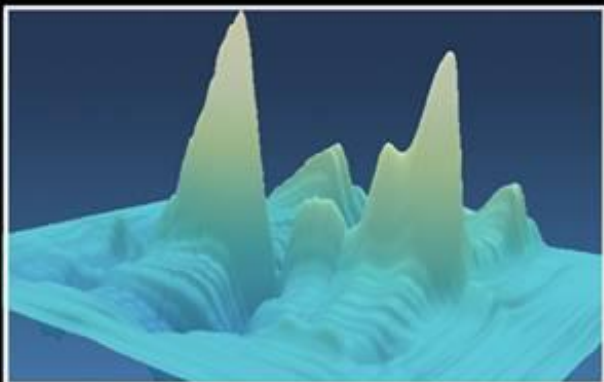
朋友覺得我在



我媽覺得我在



大眾覺得我在



指導教授覺得我在



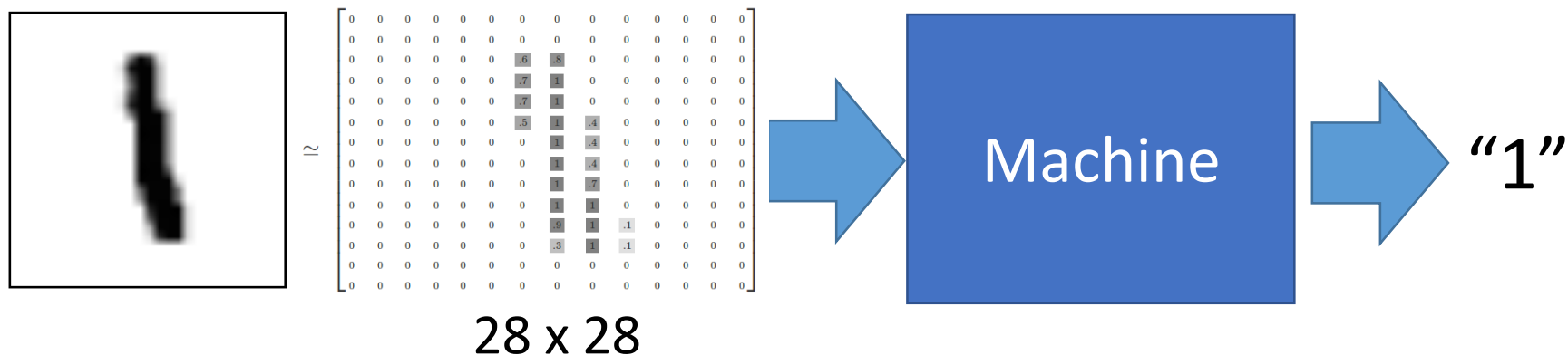
我以為我在



事實上我在

“Hello world”

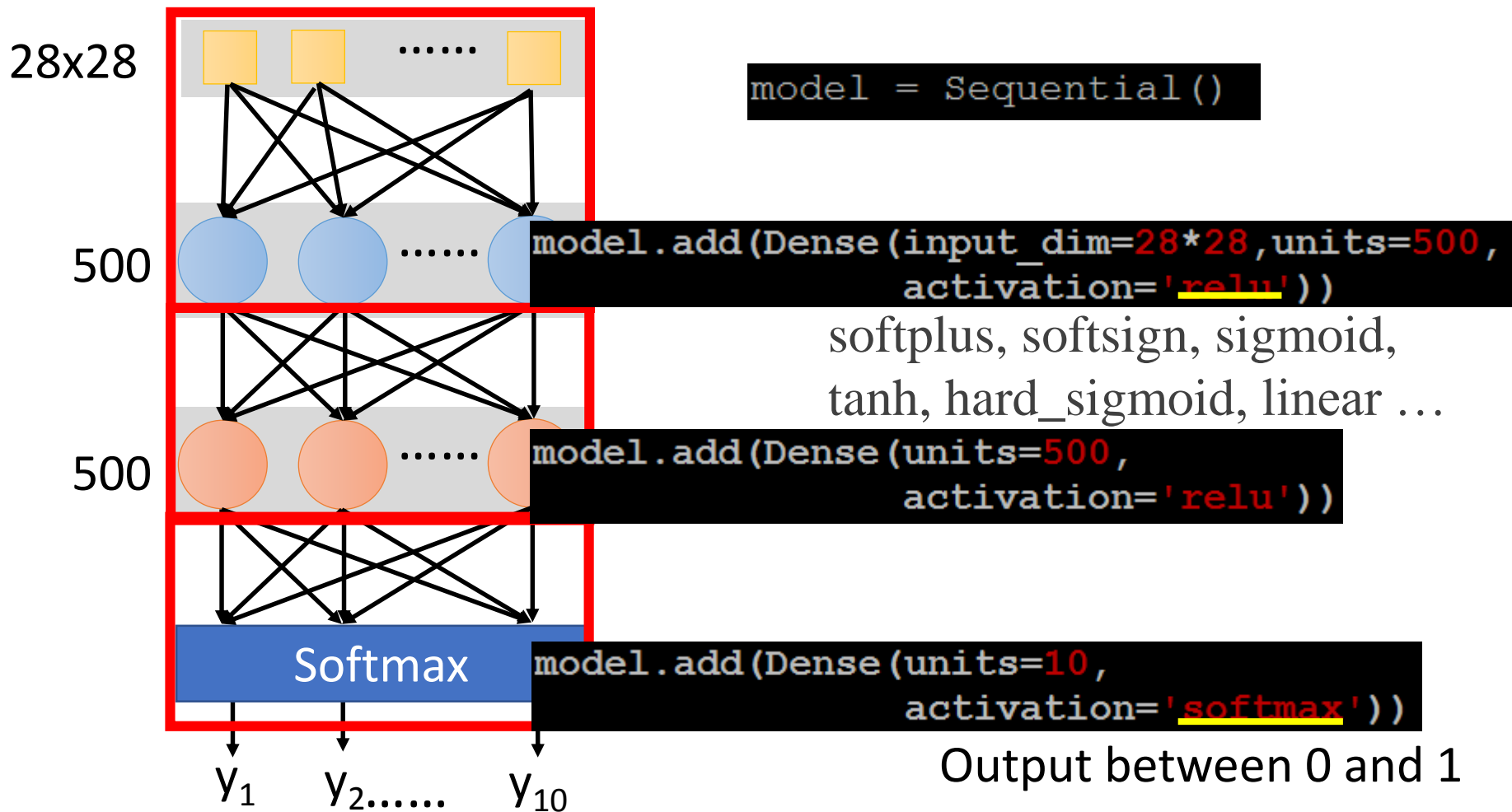
- Handwriting Digit Recognition



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

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

Keras: Building a Network



Configuration

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

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

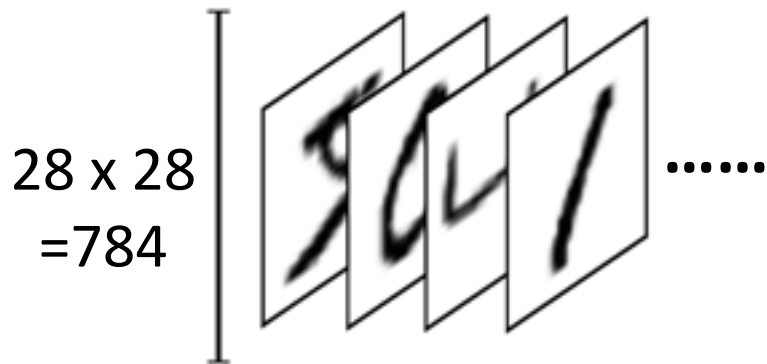
SGD, RMSprop, Adagrad, Adadelata, Adam, Adamax, Nadam

Pick the best function

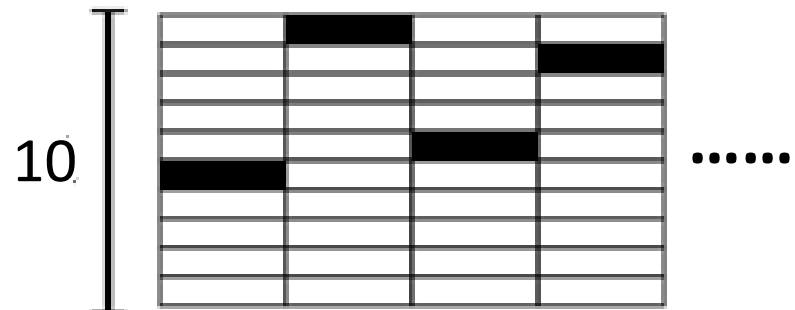
```
model.fit(x_train, y_train, batch_size=100, epochs=20)
```

numpy array

numpy array



Number of training examples

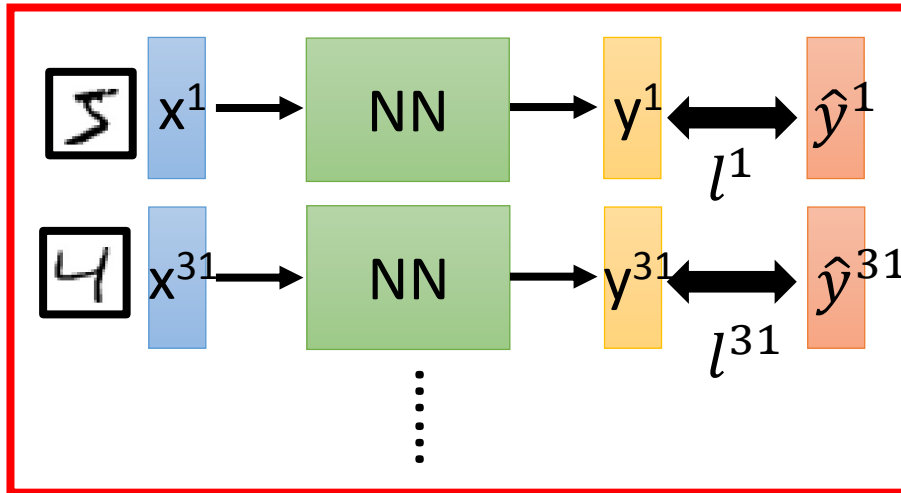


Number of training examples

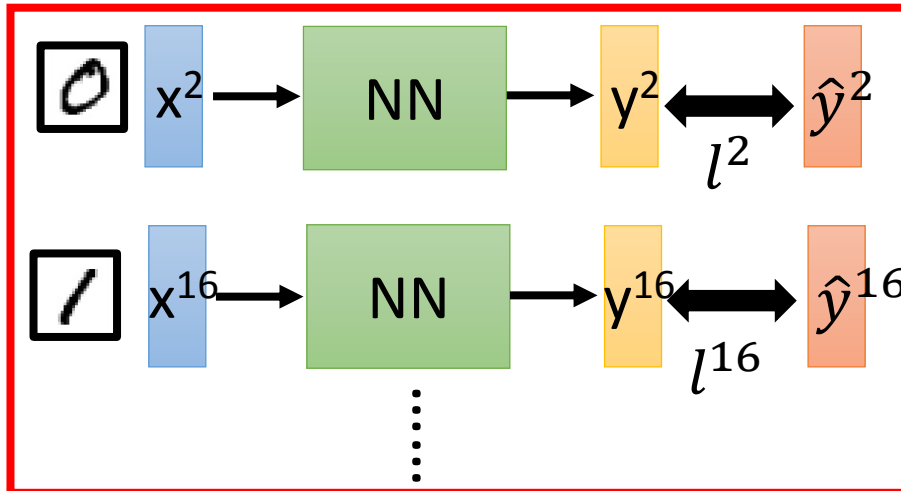
We do not really minimize total loss!

Mini-batch

Mini-batch



Mini-batch



➤ Randomly initialize network parameters

- Pick the 1st batch
 $L' = l^1 + l^{31} + \dots$
Update parameters once
- Pick the 2nd batch
 $L'' = l^2 + l^{16} + \dots$
Update parameters once
- ⋮
- Until all mini-batches have been picked

one epoch

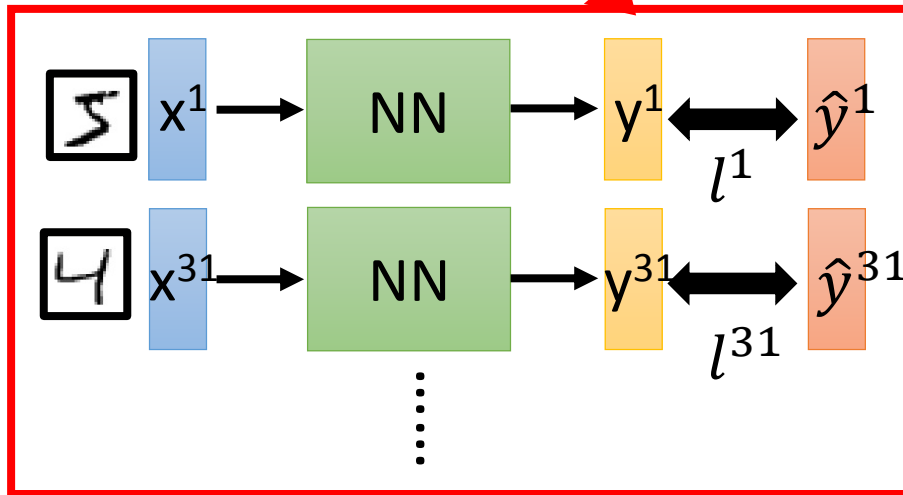
Repeat the above process

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)
```

Mini-batch



100 examples in a mini-batch

Batch size = 1 ➡

Stochastic gradient descent

- Pick the 1st batch
 $L' = l^1 + l^{31} + \dots$
Update parameters once
- Pick the 2nd batch
 $L'' = l^2 + l^{16} + \dots$
Update parameters once
- ⋮
- Until all mini-batches have been picked

Repeat 20 times

one epoch

Speed

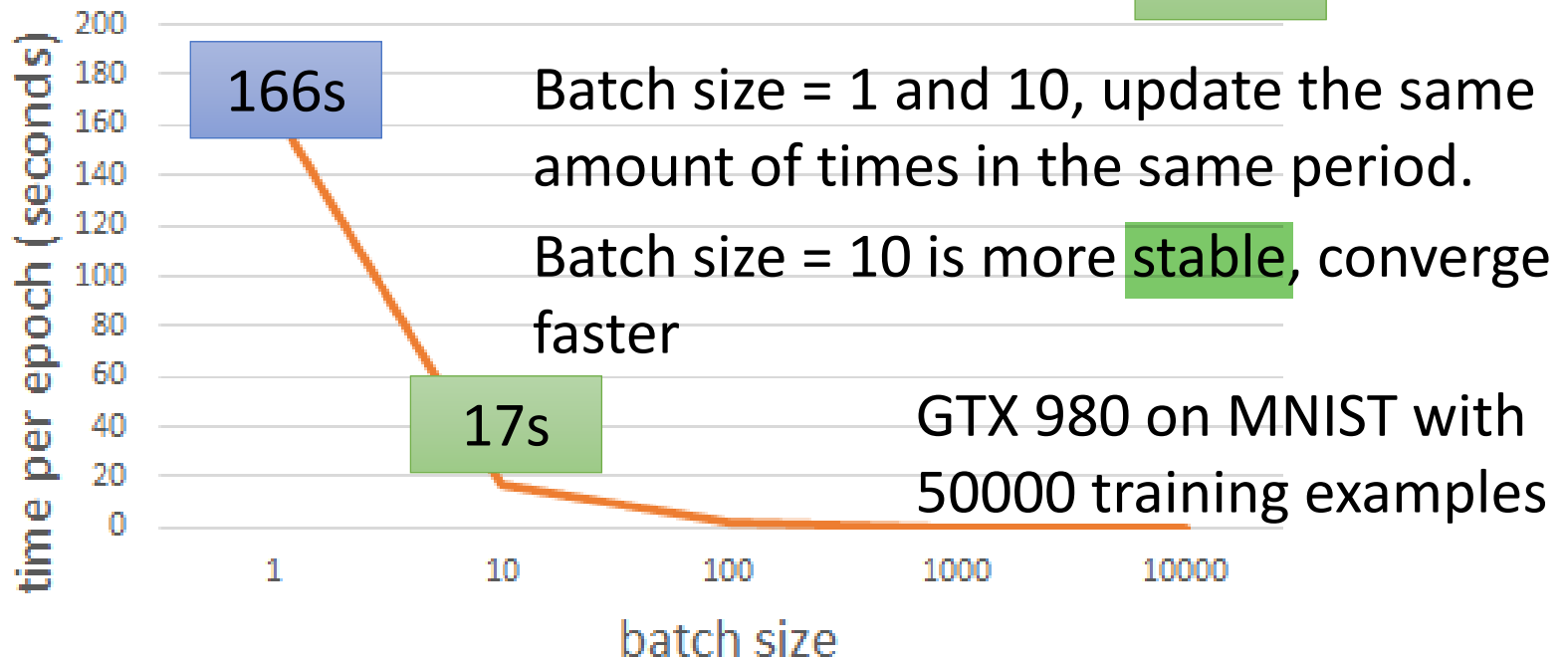
Very large batch size can yield worse performance

- Smaller batch size means more updates in one epoch

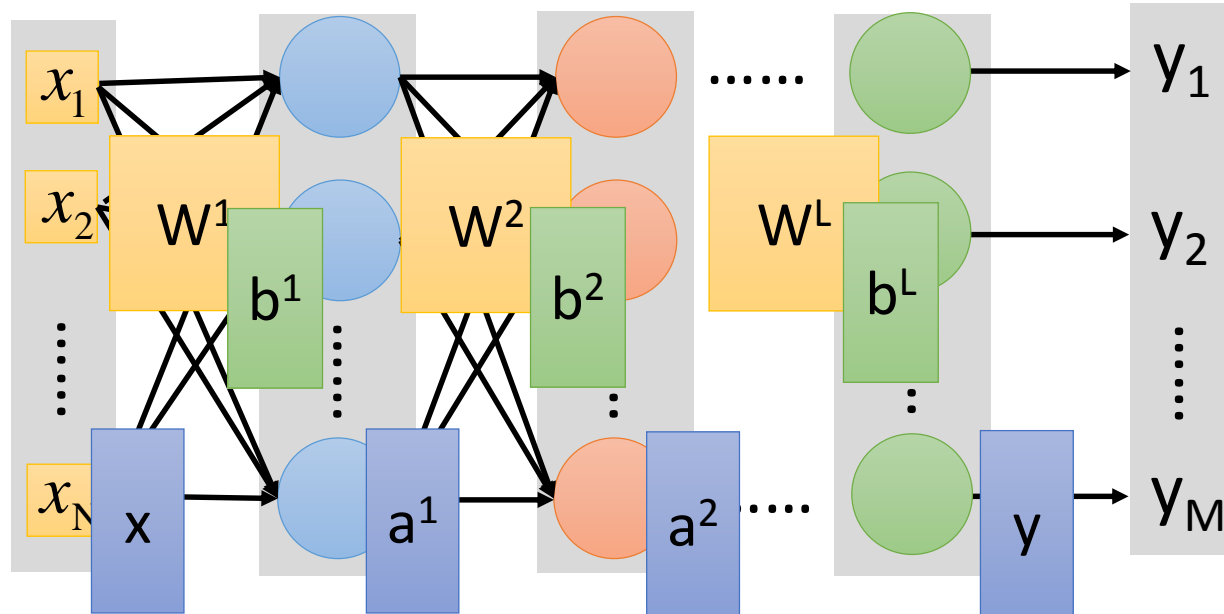
- E.g. 50000 examples

- batch size = 1, 50000 updates in one epoch 166s 1 epoch

- batch size = 10. 5000 updates in one epoch 17s 10 epoch



Speed - Matrix Operation



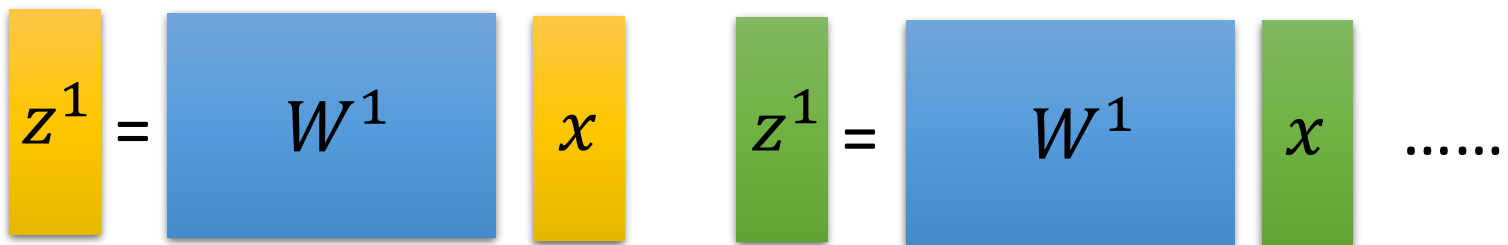
$$y = f(x) \quad \text{Forward pass (Backward pass is similar)}$$

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

Speed - Matrix Operation

- Why mini-batch is faster than stochastic gradient descent?

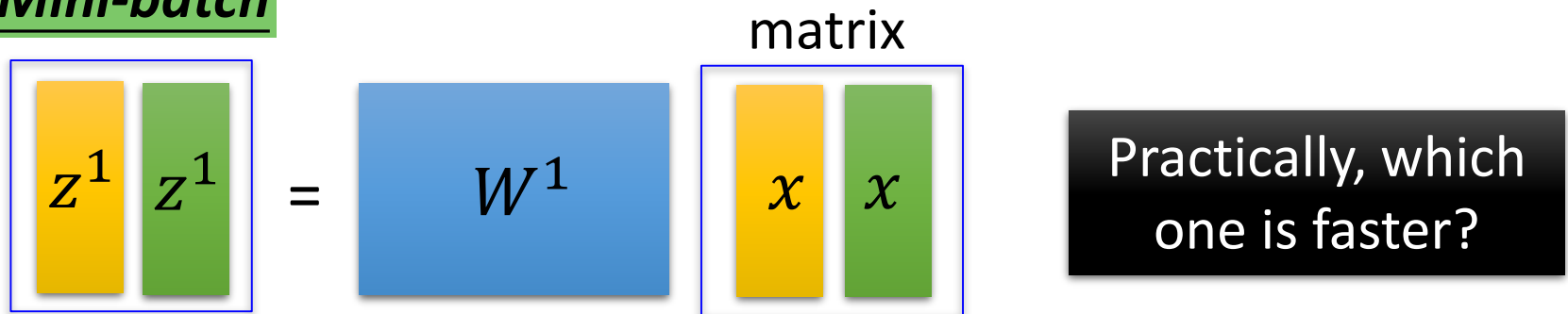
Stochastic Gradient Descent



The diagram illustrates the Stochastic Gradient Descent process. It shows two separate matrix multiplications. In the first, a yellow vertical rectangle labeled z^1 is equal to a blue square labeled W^1 multiplied by a yellow vertical rectangle labeled x . In the second, a green vertical rectangle labeled z^1 is equal to the same blue square W^1 multiplied by a green vertical rectangle labeled x . This is followed by an ellipsis, indicating that these operations are repeated for many different samples.

$$z^1 = W^1 x \quad z^1 = W^1 x \quad \dots$$

Mini-batch

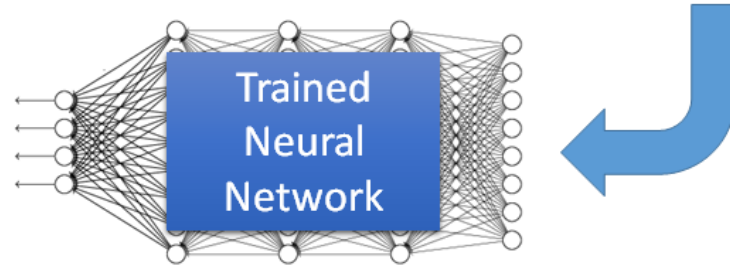


The diagram illustrates the Mini-batch process. On the left, a blue box contains two vertical rectangles, one yellow labeled z^1 and one green labeled z^1 . This is equal to a blue square labeled W^1 multiplied by a blue box on the right. The blue box contains two vertical rectangles, one yellow labeled x and one green labeled x , with the word "matrix" written above it. To the right of this equation is a black box with white text asking, "Practically, which one is faster?"

$$\begin{bmatrix} z^1 \\ z^1 \end{bmatrix} = W^1 \begin{bmatrix} x \\ x \end{bmatrix} \quad \text{matrix}$$

Practically, which one is faster?

Keras



Save and load models

<https://faroit.github.io/keras-docs/2.0.2/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)
```

Live Demo