LECTURE 14

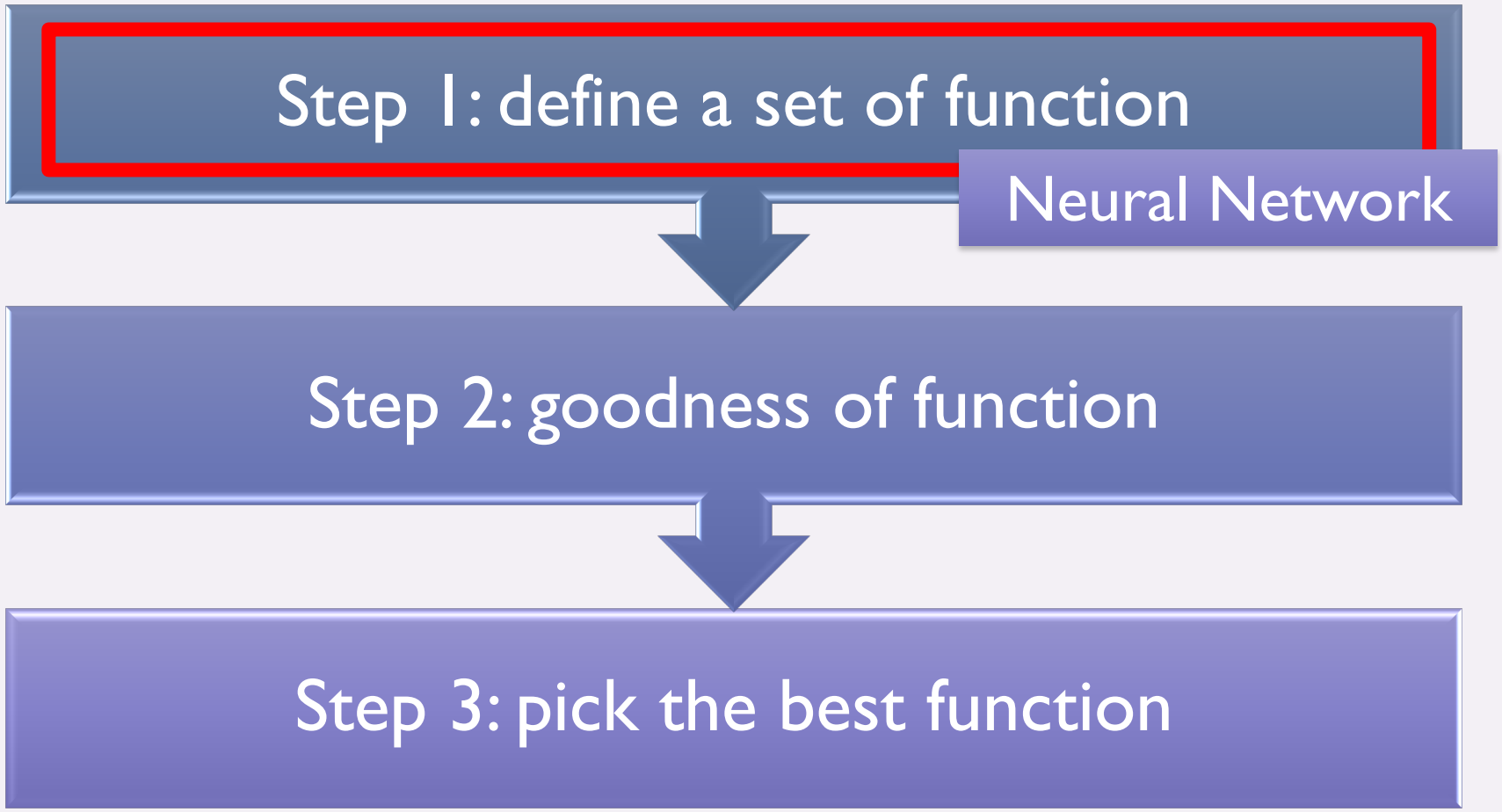
SPRING 2021
APPLIED MACHINE LEARNING
CIHANG XIE

SLIDE CREDIT:
NARGES NOROUZI

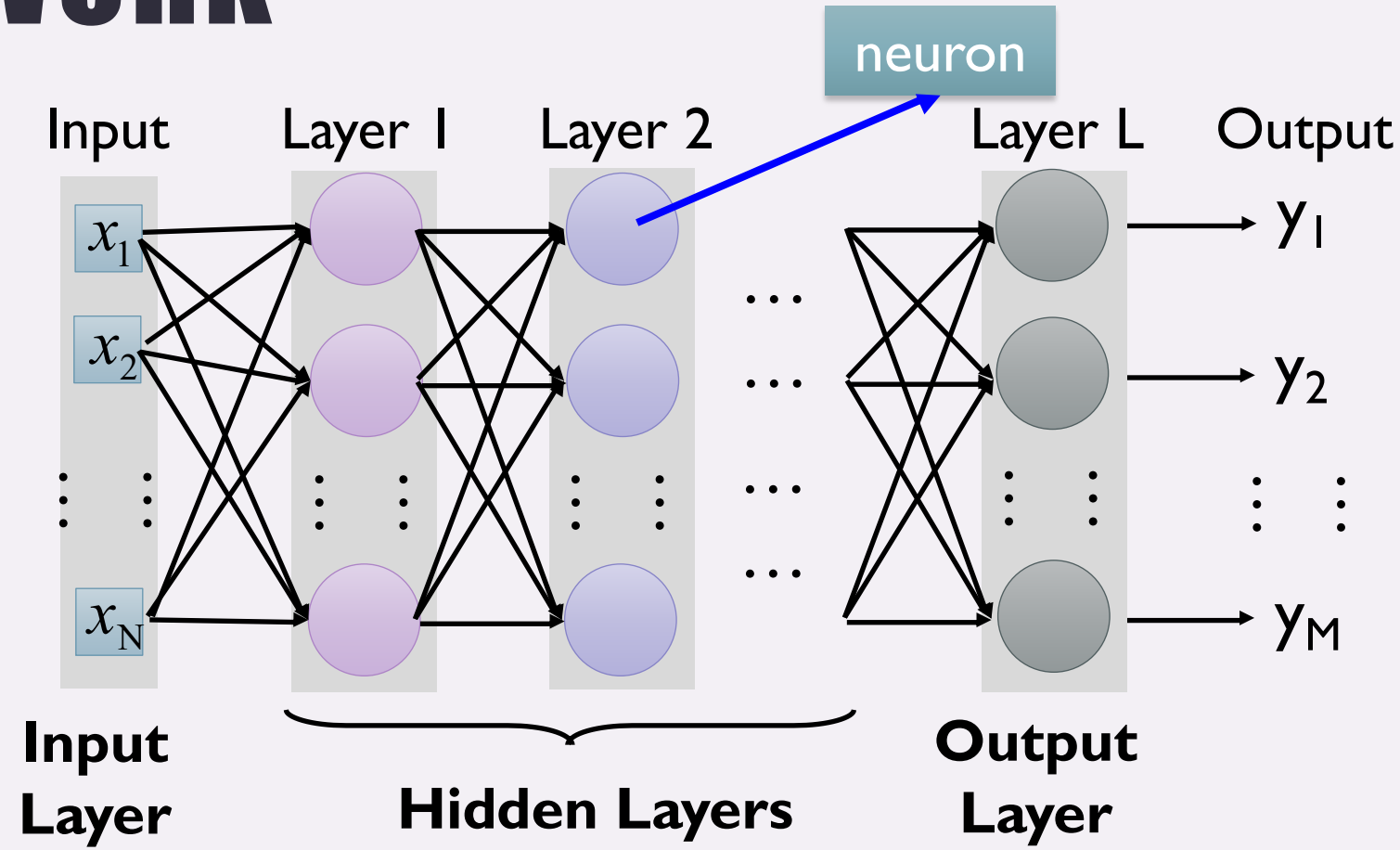
TODAY

- Introduction to Deep Neural Networks
  - TensorFlow implementation of neural networks
  - Using proper loss function
  - Mini-batch optimization

THREE STEPS FOR DEEP LEARNING



FULLY CONNECT FEEDFORWARD NETWORK



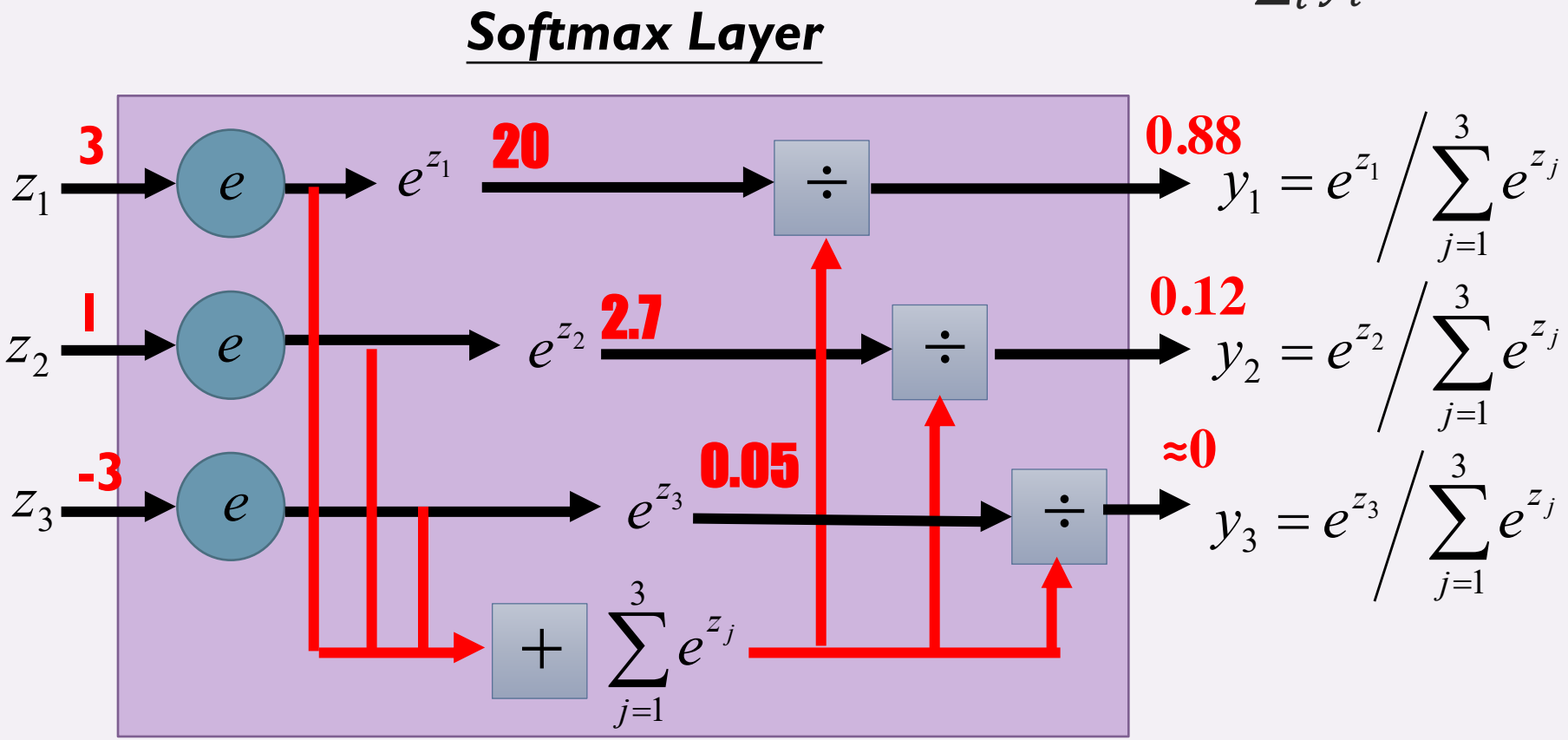
Deep means many hidden layers

SOFTMAX LAYER

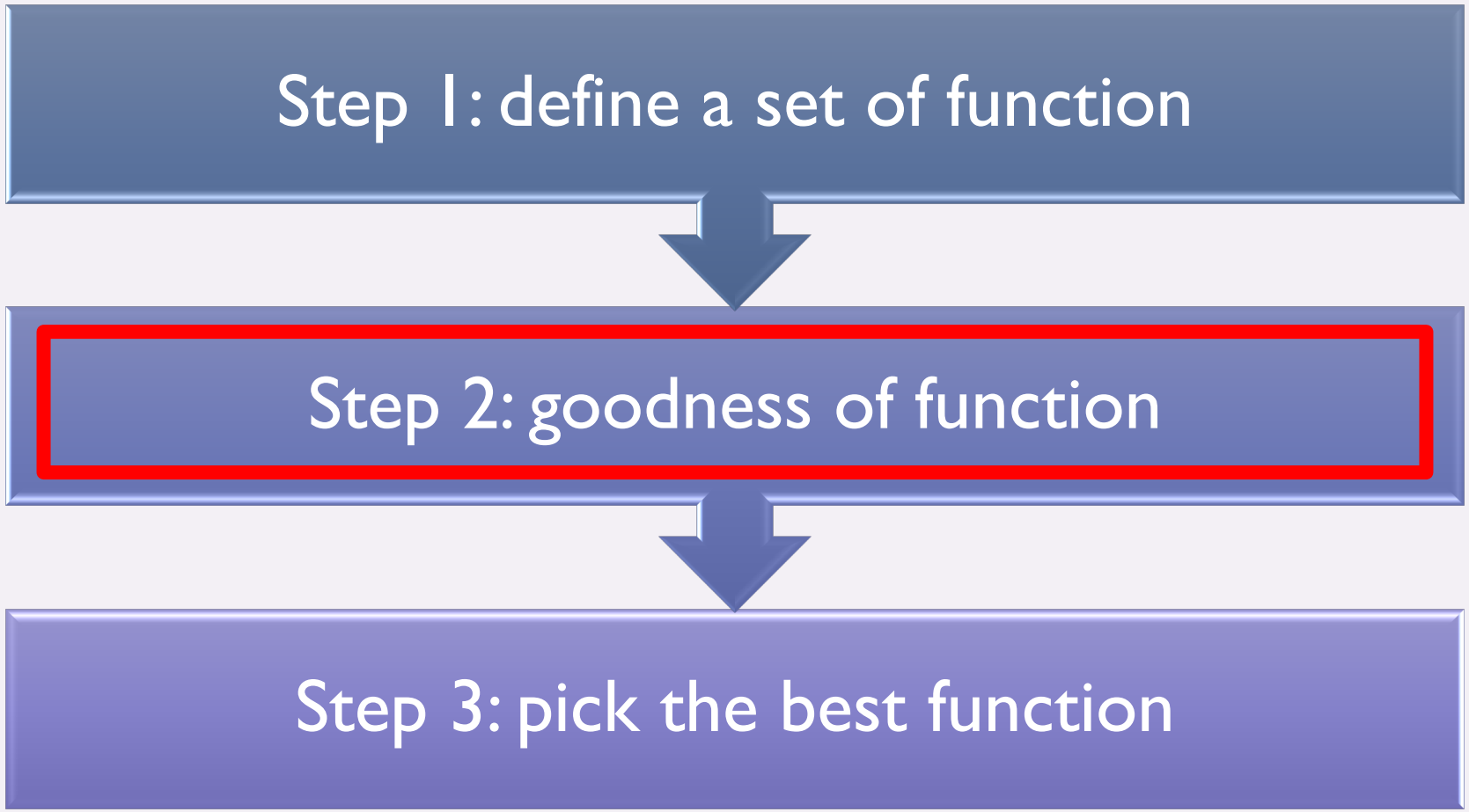
- Softmax layer as the output layer

Probability:

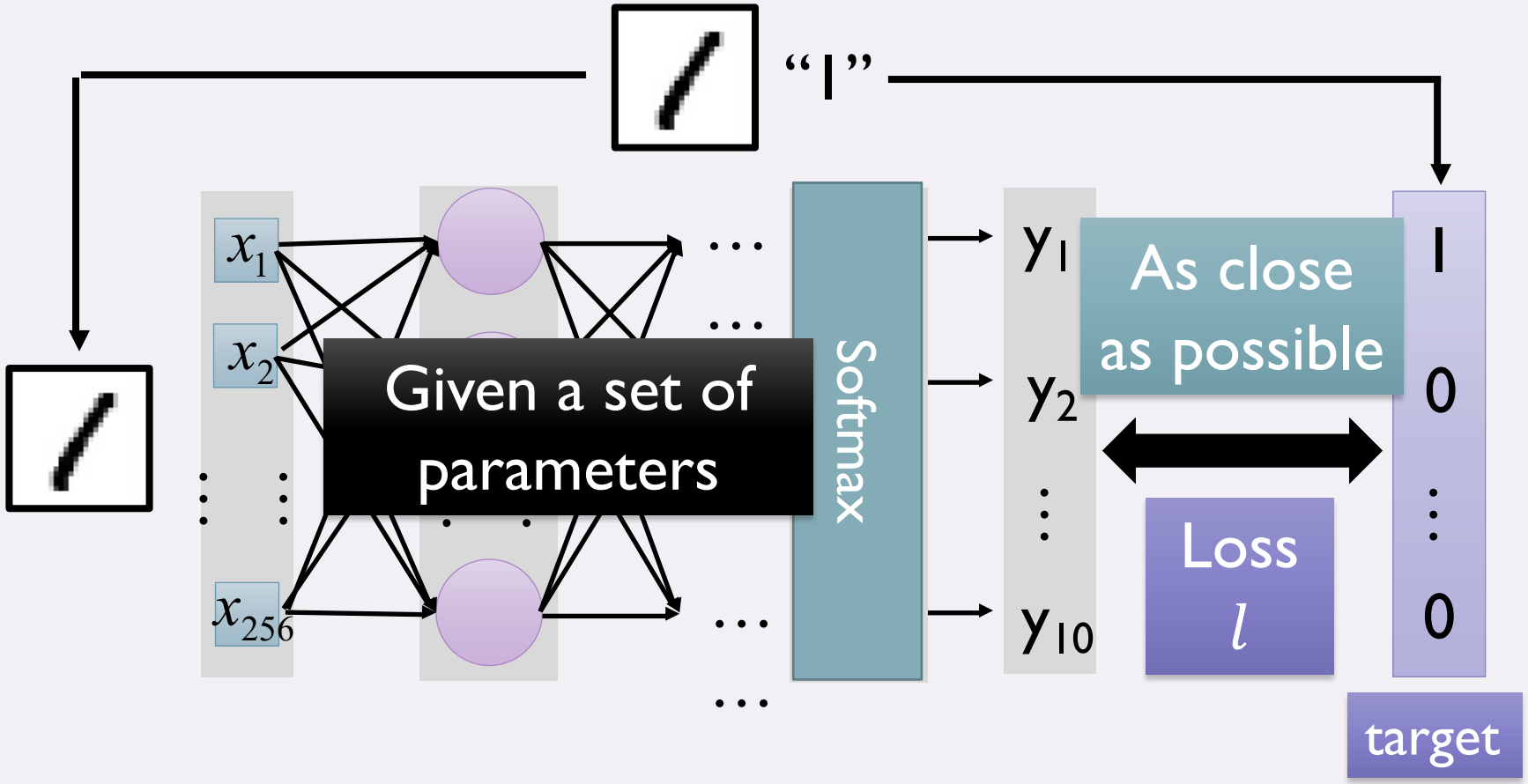
- $1 > y_i > 0$
- $\sum_i y_i = 1$



THREE STEPS FOR DEEP LEARNING

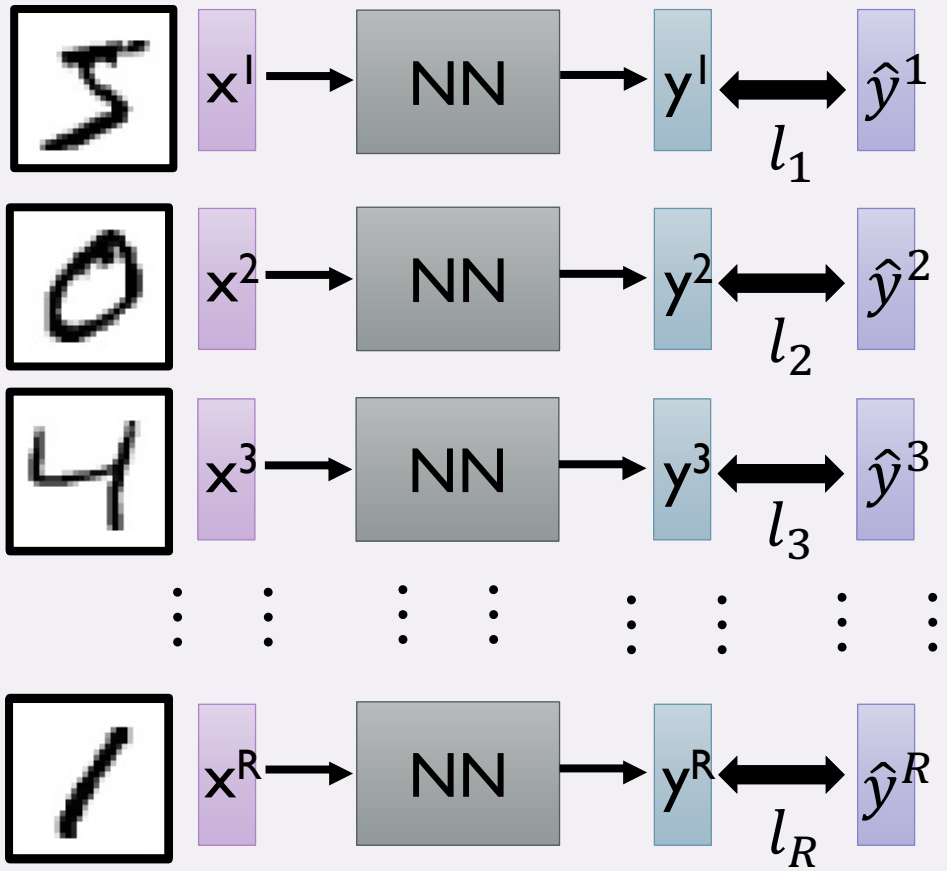


LOSS


"1"
x_1
x_2
x_{256}
Given a set of parameters
Softmax
y_1
y_2
y_{10}
As close as possible
Loss
l
1
0
0
target

TOTAL LOSS

For all training data …



Total Loss:

$$L = \sum_{r=1}^{R} l_r$$

As small as possible

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

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

CROSS ENTROPY VS. MSE

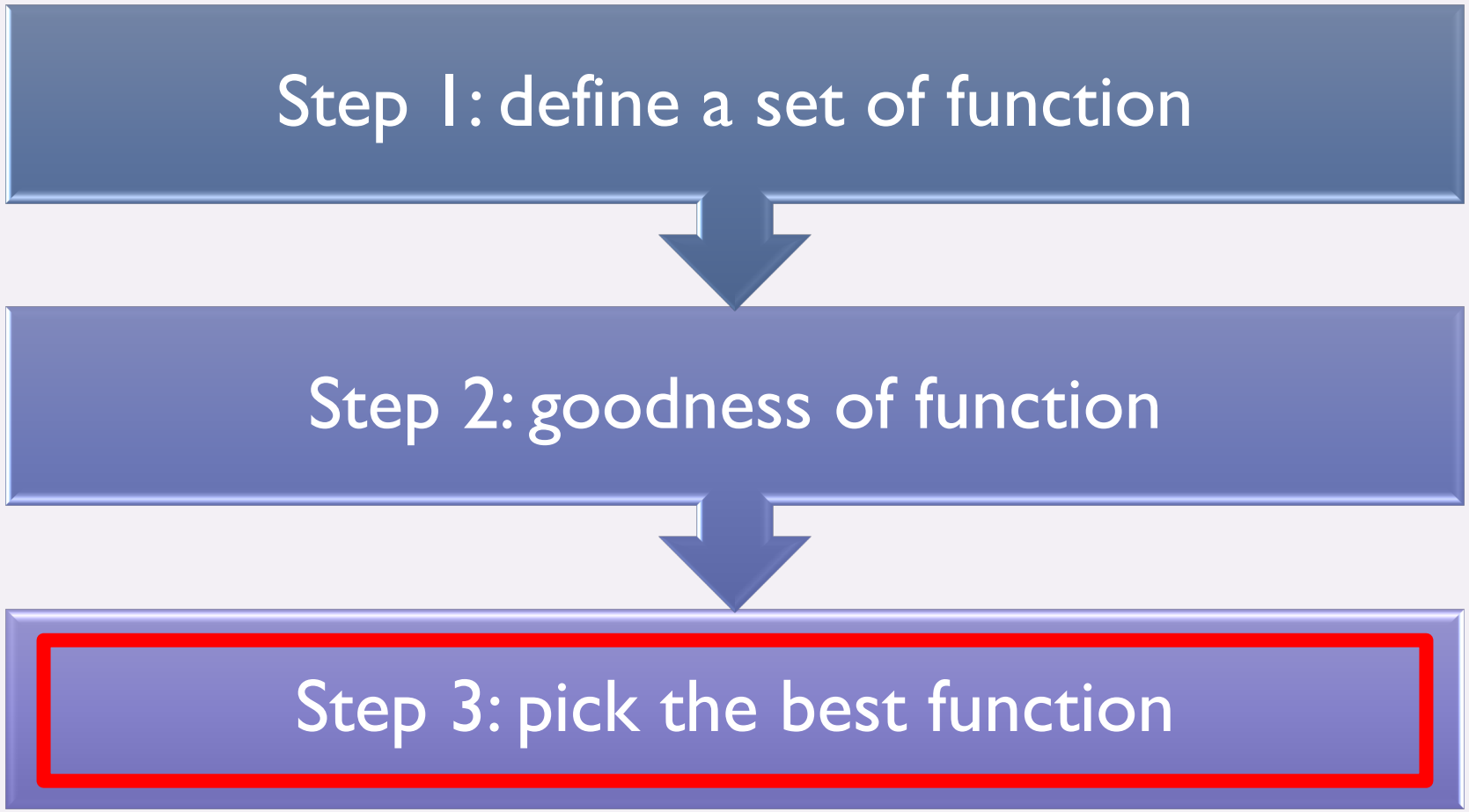
Assume, True probabilities = [1, 0, 0]

Case 1: Predicted probabilities = [0.8, 0.1, 0.1]
Case 2: Predicted probabilities = [0.8, 0.2, 0.0]

Case 3: Predicted probabilities = [0.81, 0.19, 0.0]

For logistic regression, MSE → non-convex optimization
Cross Entropy → still convex

THREE STEPS FOR DEEP LEARNING



GRADIENT DESCENT

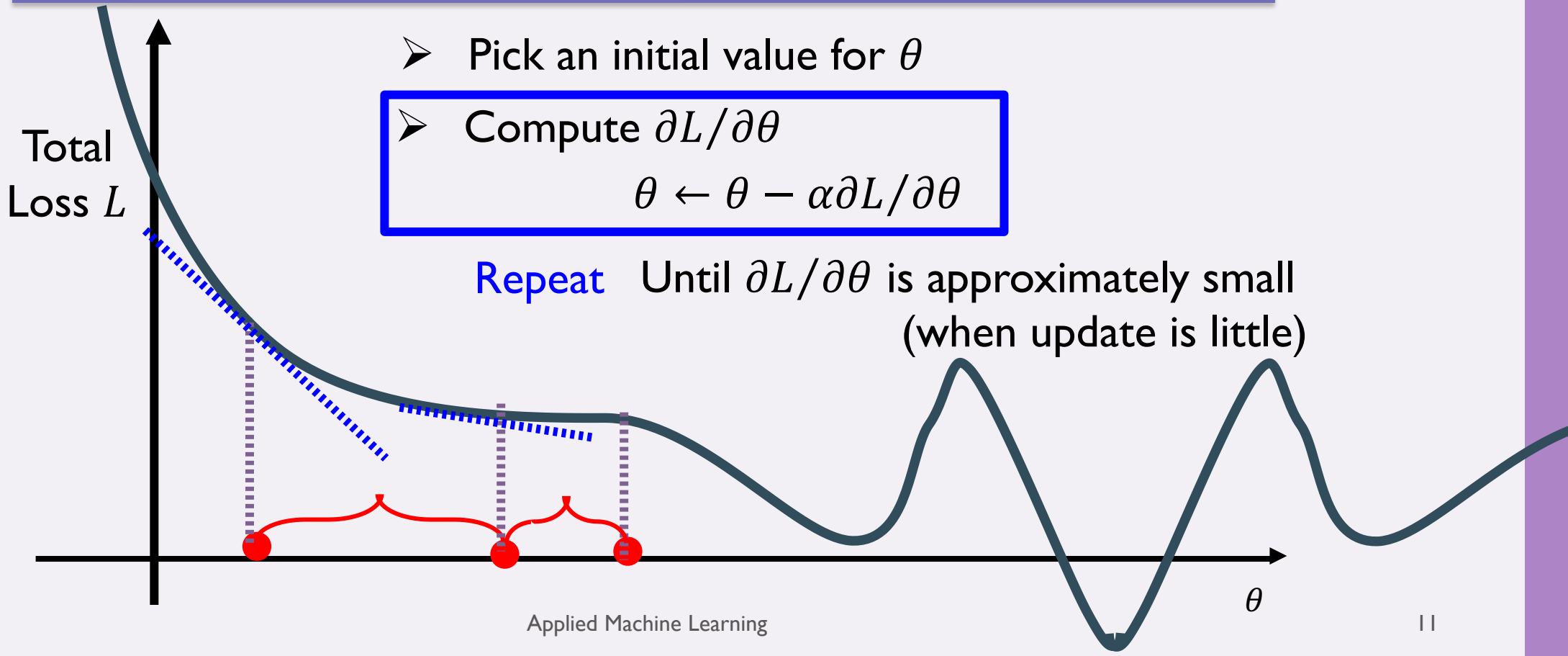
Network parameters $\theta = \{\theta_1, \theta_2, \cdots\}$

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

- Pick an initial value for θ
- Compute $\partial L/\partial\theta$

$$\theta \leftarrow \theta - \alpha\partial L/\partial\theta$$

Repeat Until $\partial L/\partial\theta$ is approximately small (when update is little)



OUTLINE

Introduction of Deep Learning

"Hello World" for Deep Learning

Tips for Deep Learning

RECIPE FOR DEEP LEARNING

YES

NO

Good Results on Testing Data?

Overfitting!

YES

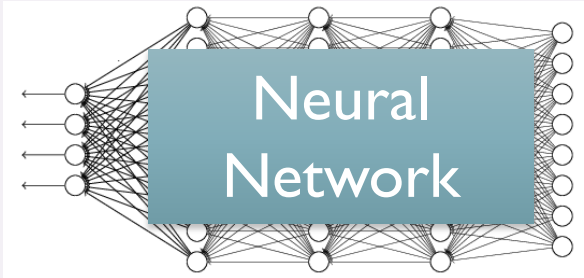
NO

Good Results on Training Data?

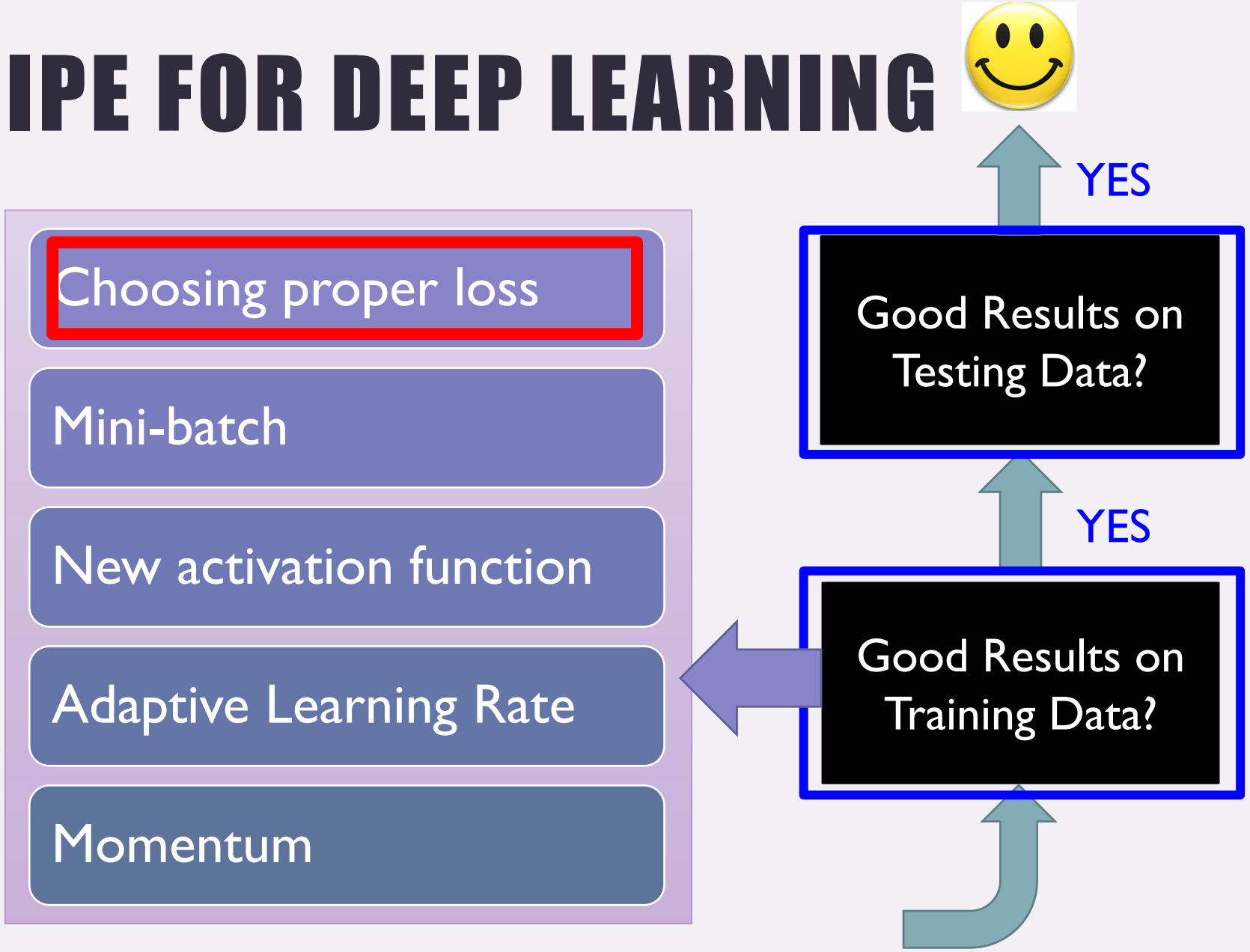
Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function



RECIPE FOR DEEP LEARNING



DEMO

Square Error

```
model.compile(loss=keras.losses.mean_squared_error,
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

Binary Cross Entropy

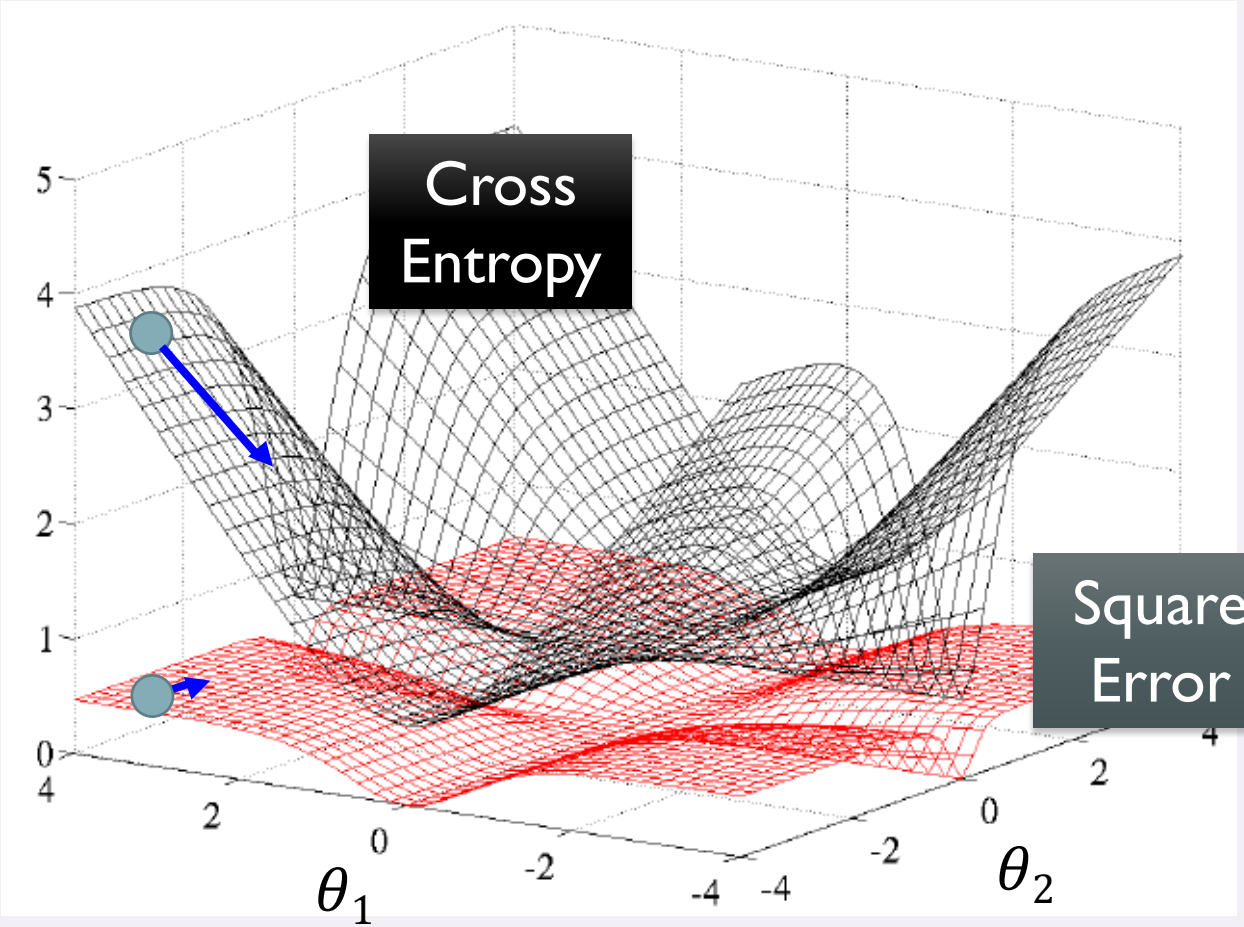
```
model.compile(loss=keras.losses.binary_crossentropy,
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

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

CHOOSING PROPER LOSS

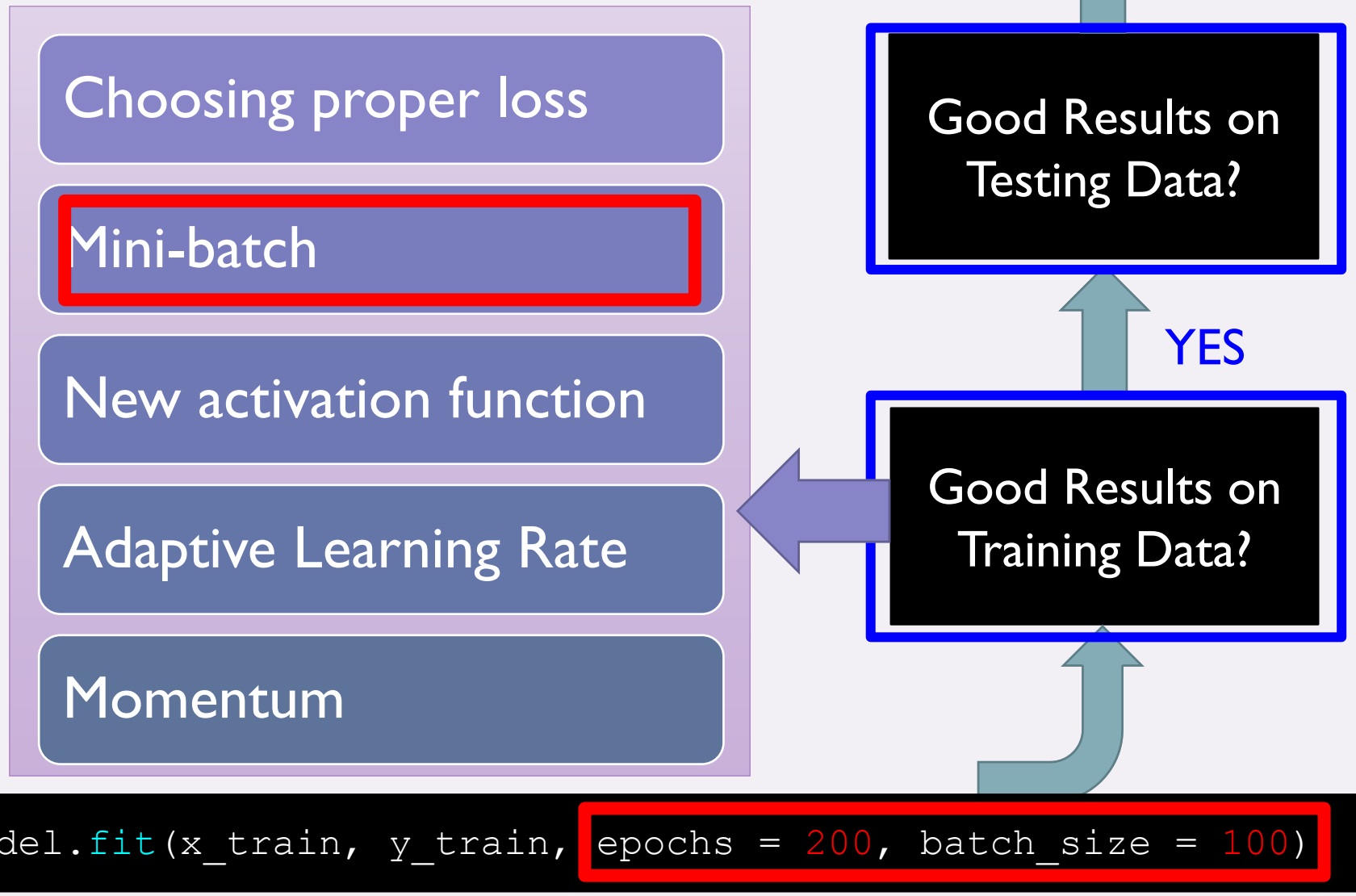
When using softmax output layer, choose cross entropy

Total Loss



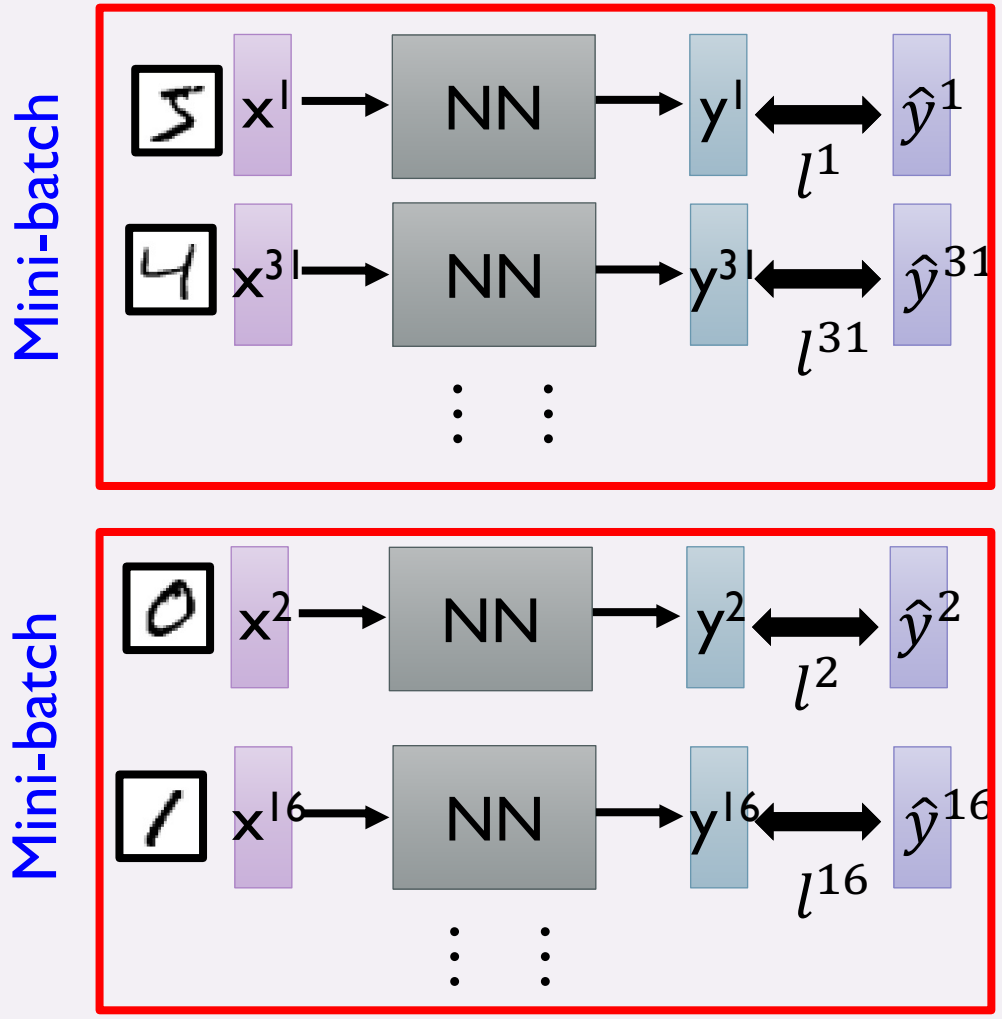
http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

RECIPE FOR DEEP LEARNING



MINI-BATCH

We do not really minimize total loss!



- 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

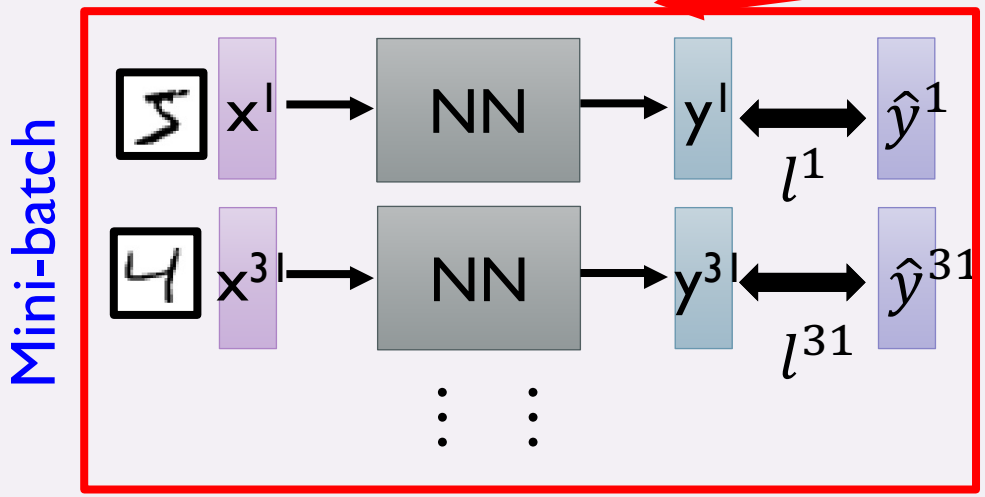
  ⋮
- Until all mini-batches have been picked

one epoch

Repeat the above process

MINI-BATCH

```
model.fit(x_train, y_train, epochs = 200, batch_size = 100)
```



100 examples in a mini-batch

- 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

Repeat 200 times

one epoch

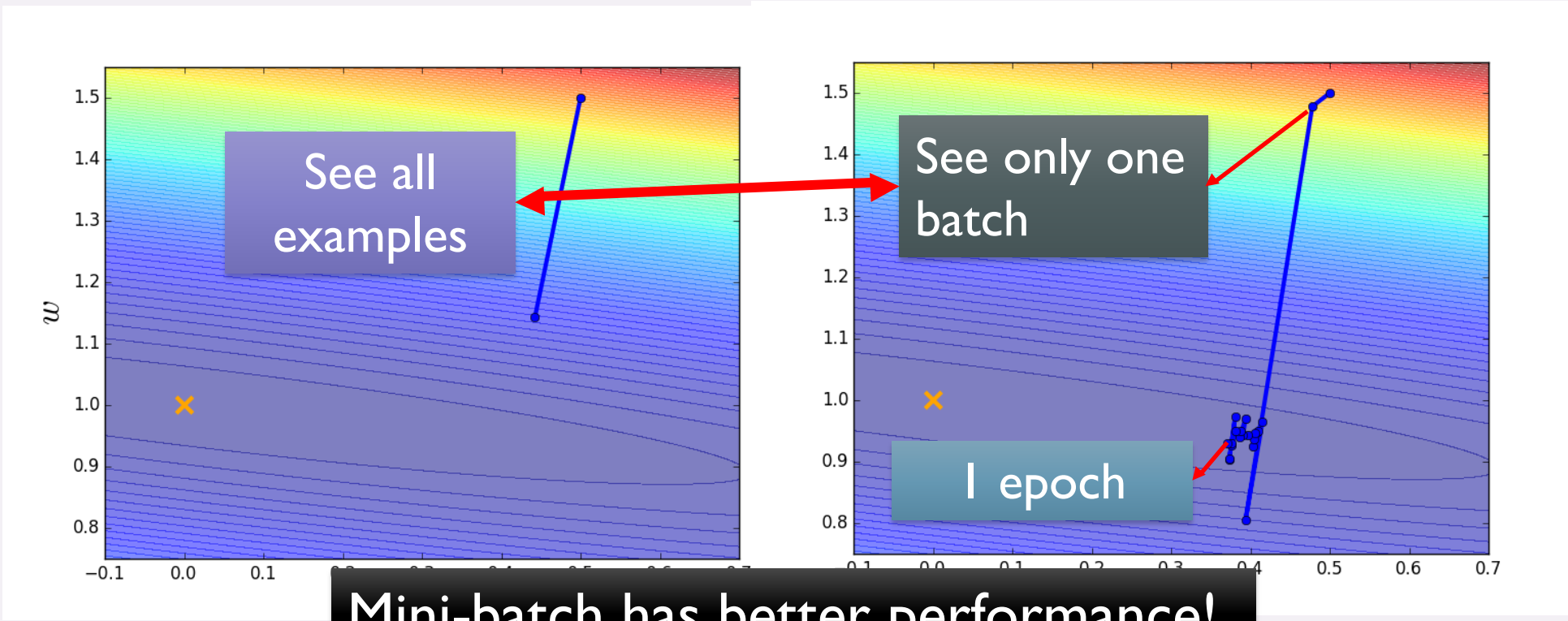
MINI-BATCH IS FASTER

Original Gradient Descent

Update after seeing all examples

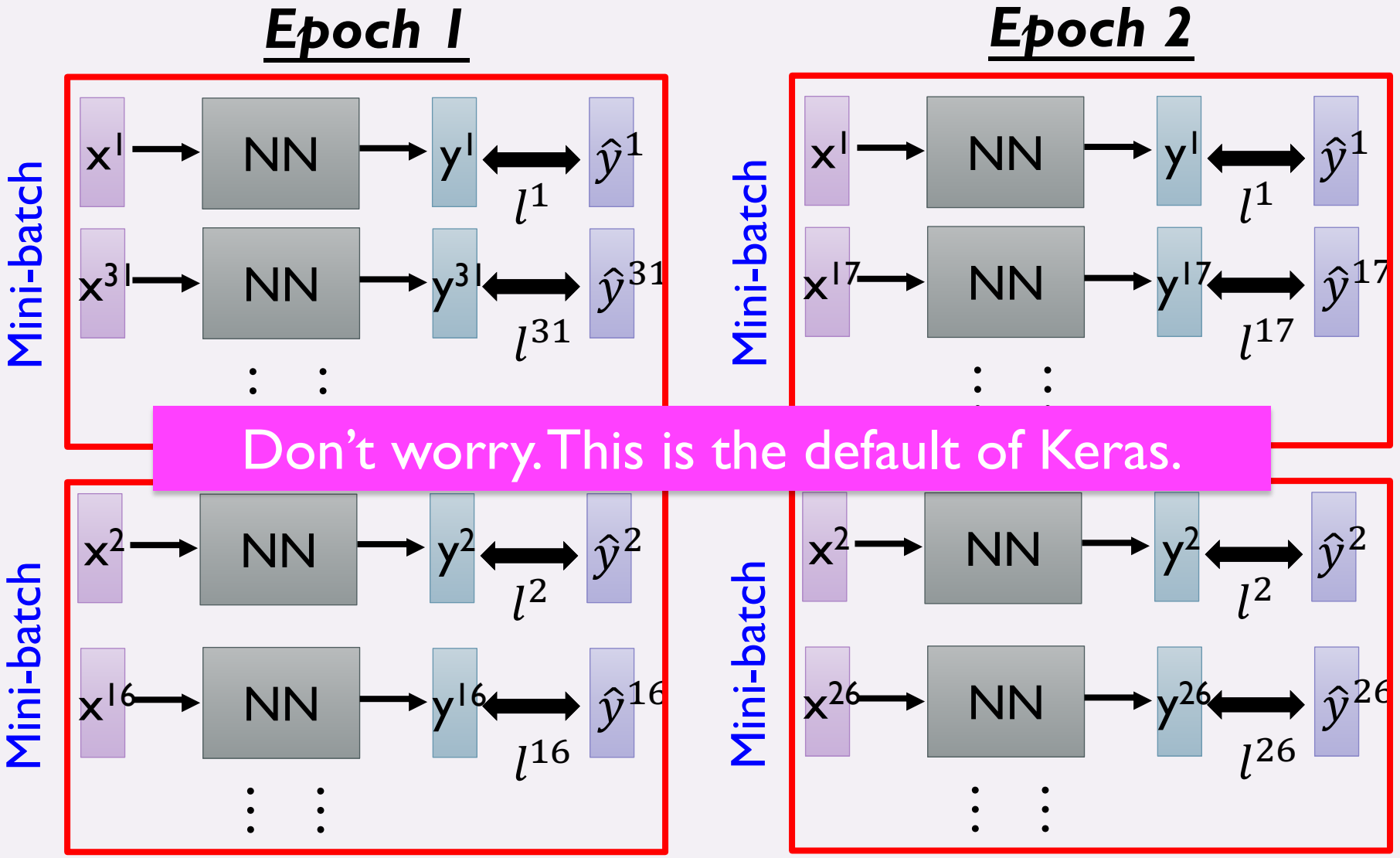
With Mini-batch

If there are 20 batches, update 20 times in one epoch.



Mini-batch has better performance!

SHUFFLE EXAMPLES FOR EACH EPOCH



EXERCISE

HTTPS://BIT.LY/3EK0SPH

QUESTIONS?