



Chapter 3:

Neural networks



Outlines

1. Anatomy of neural networks
2. Introduction to Keras
3. Keras in practice



Keras in practice

3. Keras environment setting

- It's highly recommended to run deep-learning code on a modern NVIDIA GPU.
 - ✓ Some applications will be excruciatingly slow on a CPU, even a fast multicore CPU.
 - ✓ Speed increases by a factor of 5 or 10 by using a modern GPU compared to using a CPU.
- If you don't want to install a GPU on your machine, you can run your experiments on Google Cloud Platform.
- In order to use Keras, it is necessary to install TensorFlow, CNTK, Theano, or all of those.

Keras in practice

3. Keras environment setting

- Google Colab.



<https://colab.research.google.com/#>

- Jupyter notebooks.



<https://jupyter.org/>

- Server
- Worstaion
- Personal computer



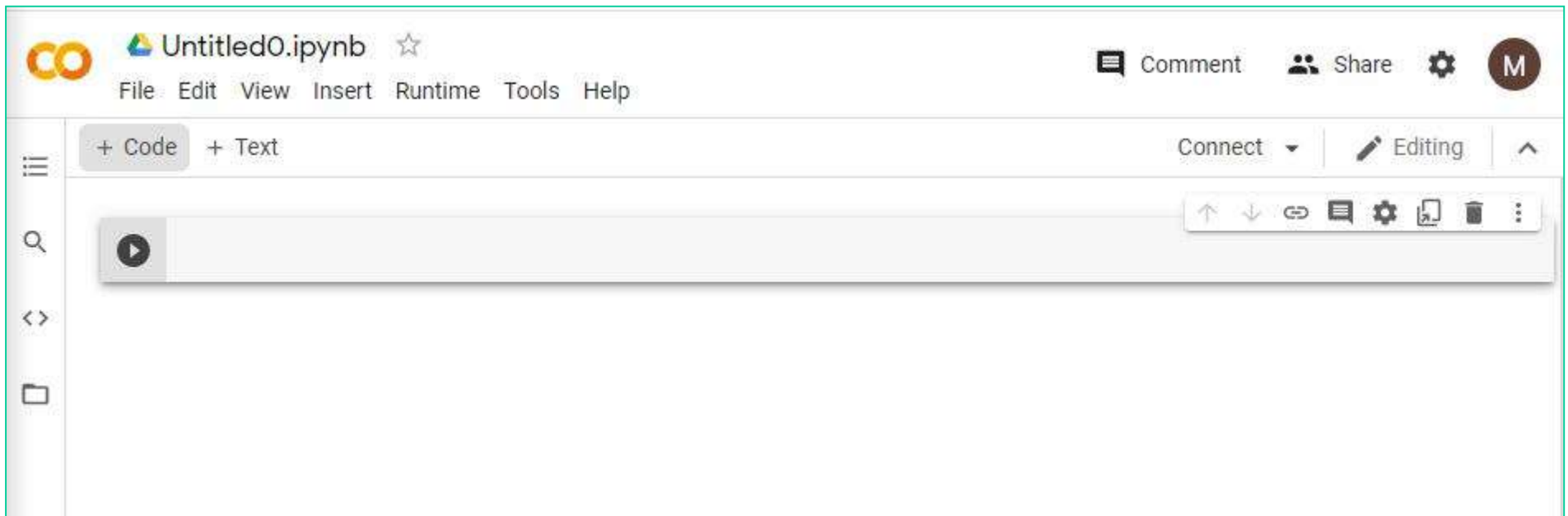
Keras in practice

❖ Google colab:

- [Google colab](#) is a google utility, a cloud service, free.
- [Colab](#) doesn't require complex setup. The notebooks we create can be edited simultaneously by team members - much like you would edit a document in Google Docs.
- The biggest advantage is that [Colab](#) supports most popular machine learning libraries and can be simple when installing a new library.

Keras in practice

❖ Google colab:





Keras in practice

- ❖ Google colab:
- ❖ Connect Colab to Google Drive

```
[ ] from google.colab import drive  
    drive.mount('/content/gdrive')
```

➞ Go to this URL in a browser: <https://accounts.google.com/o/>

Enter your authorization code:

.....

Mounted at /content/gdrive



Keras in practice

❖ Google colab:

- Write and execute code in [Python](#)
- Create/Upload/Share notebook files
- Import/save notebooks to [Google Drive](#)
- Import notebooks from [GitHub](#)
- [PyTorch](#), [TensorFlow](#), [Keras](#), [OpenCV](#) built-in
- Free cloud service with free [GPUs](#), GPU upgradeable using paid GPUs.



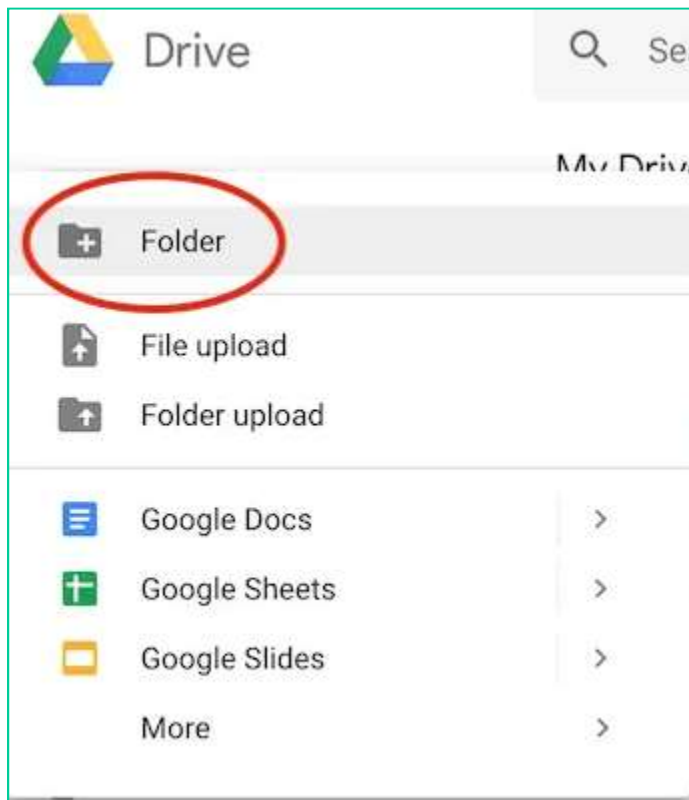
Keras in practice

❖ Google colab:

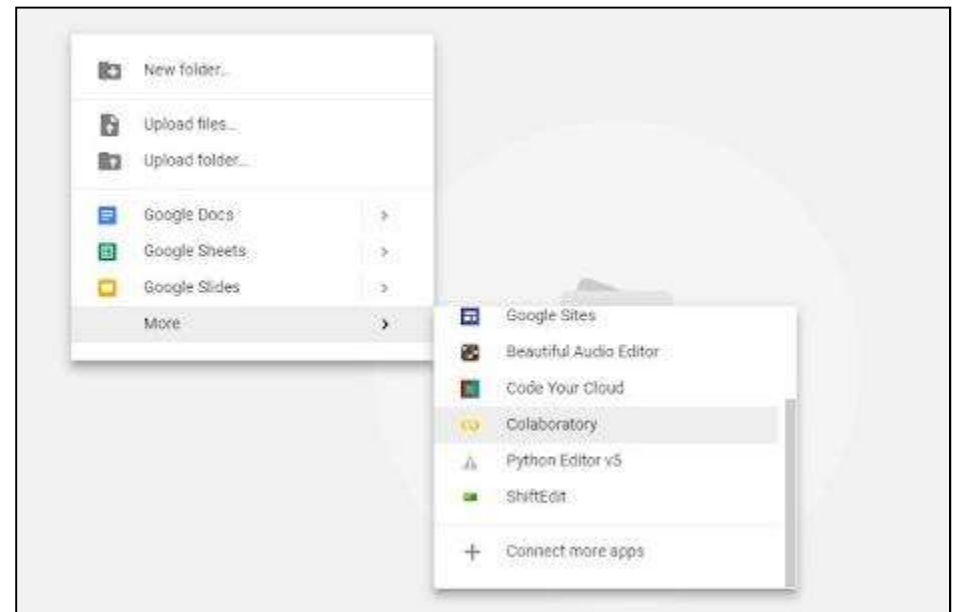
- To use [Colaboratory](#), you must have a Google account and then access [Colaboratory](#) with that account. Otherwise, most of [Colaboratory's](#) features will not work.
- To create a google colab file, first we access the folder on google drive where we want to create the colab file, then right-click and select the "[Colaboratory](#)" option.

Keras in practice

❖ Google colab:



Create a folder on Google Drive



Tạo Notebook trên Colab

Keras in practice

- ❖ Google colab:
 - Select GPU:
 - ✓ Select “runtime”.
 - ✓ Select “change runtime type”.
 - ✓ Select “GPU” from “hardware accelerator”

Notebook settings

Hardware accelerator
GPU ⌵ ?

To get the most out of Colab, avoid using a GPU unless you need one. [Learn more](#)

☐ Background execution

Want your notebook to keep running even after you close your browser? [Upgrade to Colab Pro+](#)

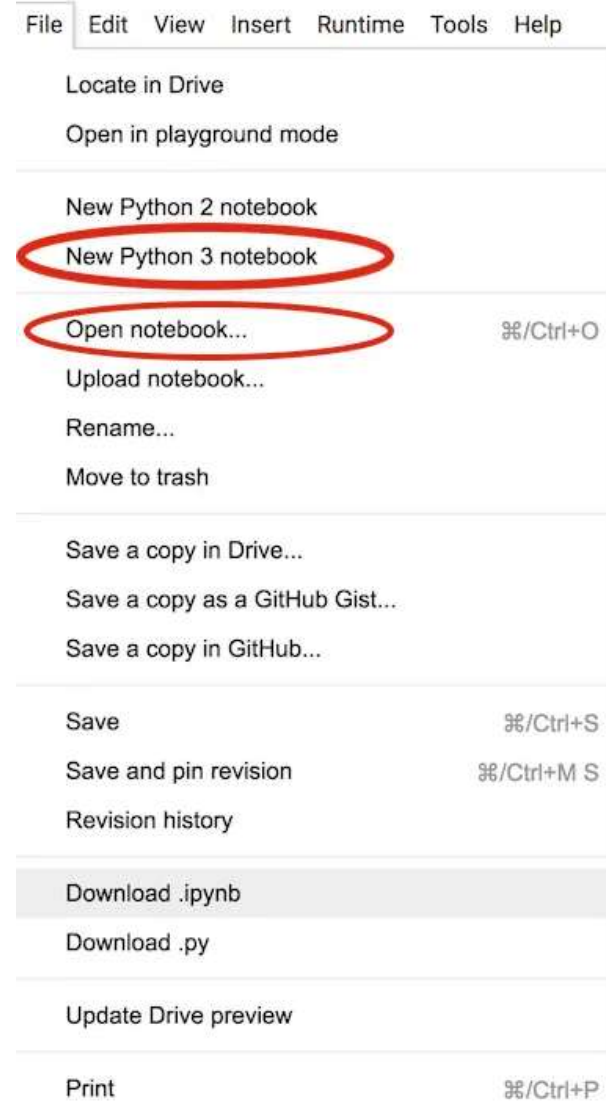
☐ Omit code cell output when saving this notebook

Cancel [Save](#)

Keras in practice

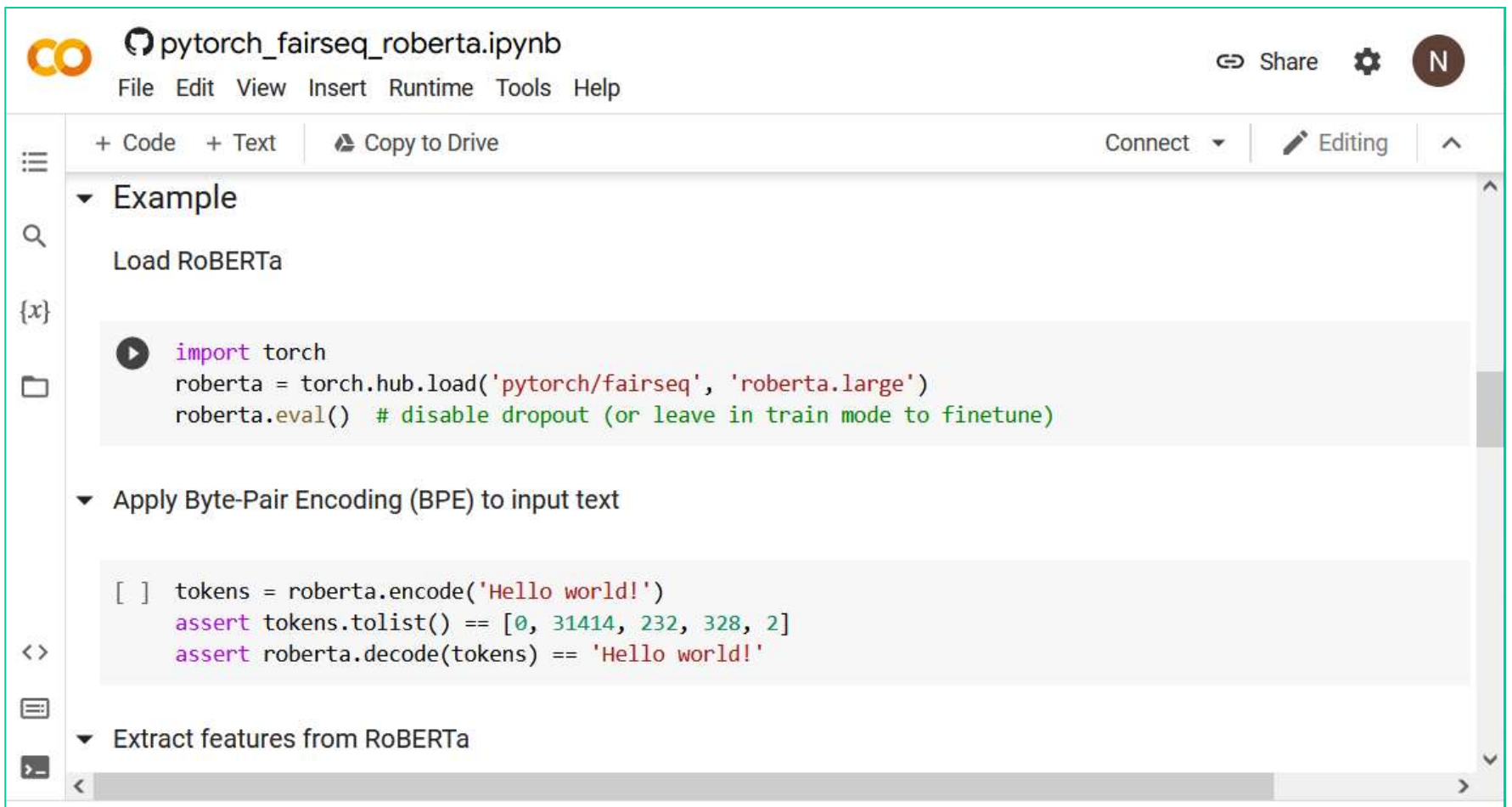
❖ Google colab:

- Reopen or create a new notebook.



Keras in practice

❖ Google colab:



The screenshot shows a Google Colab notebook interface. The title bar indicates the notebook is named 'pytorch_fairseq_roberta.ipynb'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right, there are 'Share', 'Settings', and a user profile icon labeled 'N'. The left sidebar contains icons for file management and search. The main area displays the notebook content, which is organized into sections: 'Example', 'Load RoBERTa', 'Apply Byte-Pair Encoding (BPE) to input text', and 'Extract features from RoBERTa'. The 'Load RoBERTa' section contains a code cell with the following Python code:

```
import torch
roberta = torch.hub.load('pytorch/fairseq', 'roberta.large')
roberta.eval() # disable dropout (or leave in train mode to finetune)
```

The 'Apply Byte-Pair Encoding (BPE) to input text' section contains a code cell with the following Python code:

```
[ ] tokens = roberta.encode('Hello world!')
assert tokens.tolist() == [0, 31414, 232, 328, 2]
assert roberta.decode(tokens) == 'Hello world!'
```

The 'Extract features from RoBERTa' section is currently empty.



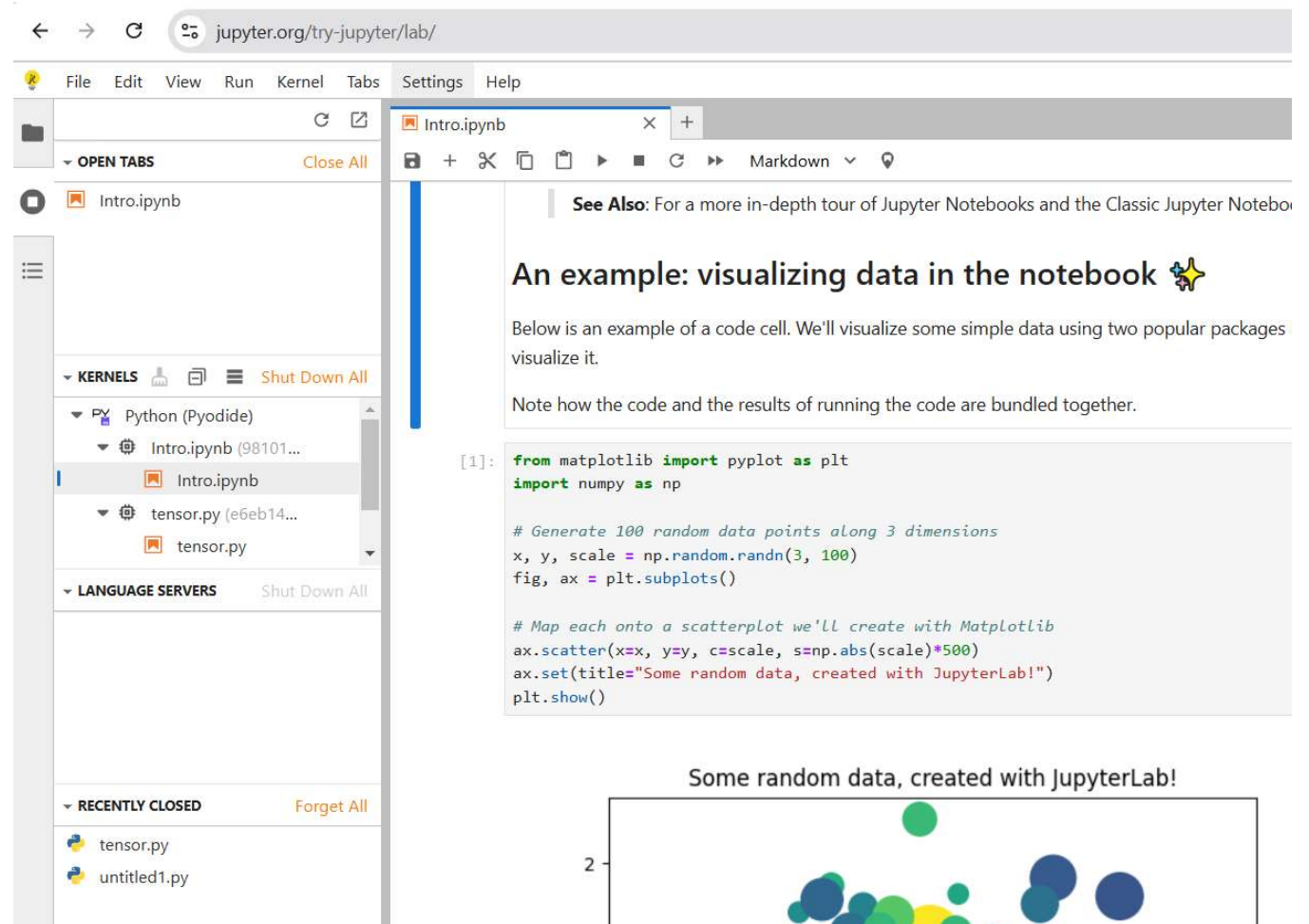
Keras in practice

❖ Jupyter notebooks:

- Jupyter notebooks are a great way to run deep-learning experiments.
- Jupyter notebook, formerly known as IPython (or Interactive Python), is a flexible and powerful open source research tool.
- It is widely used in the data-science and machine-learning communities.
- The name Jupyter is an acronym of the three core languages it was designed for: JULia, PYThon, and R.
- Project Jupyter supports interactive data science and scientific computing across more than 40 programming languages.

Keras in practice

❖ Jupyter notebooks:



The screenshot displays the JupyterLab web interface. The top navigation bar includes menus for File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The left sidebar shows the 'OPEN TABS' section with 'Intro.ipynb' selected. Below this, the 'KERNELS' section lists 'Python (Pydide)' with 'Intro.ipynb (98101...)' and 'tensor.py (e6eb14...)'. The 'LANGUAGE SERVERS' section is also visible. The main area shows the 'Intro.ipynb' notebook with a code cell containing the following Python code:

```
[1]: from matplotlib import pyplot as plt
import numpy as np

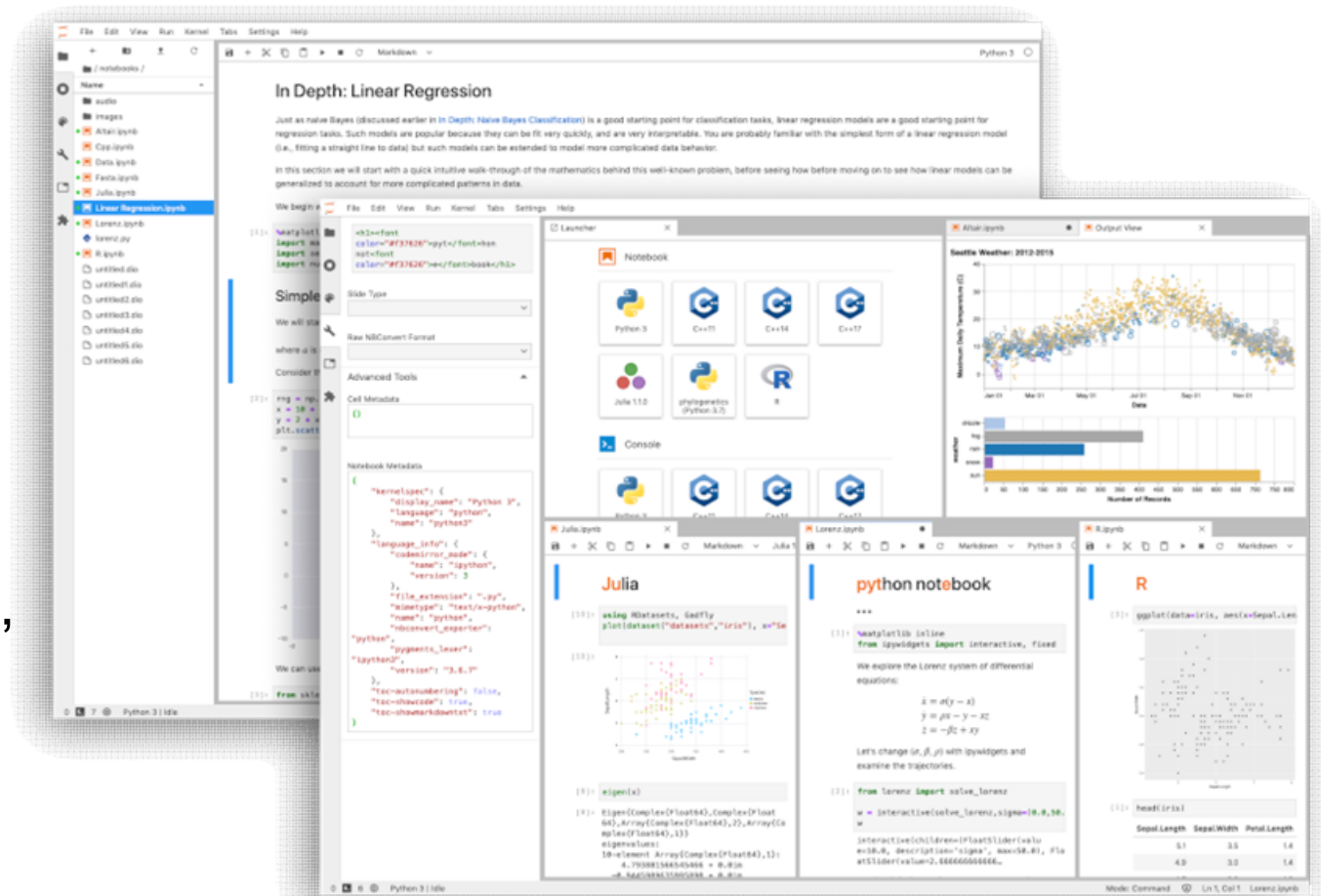
# Generate 100 random data points along 3 dimensions
x, y, scale = np.random.randn(3, 100)
fig, ax = plt.subplots()

# Map each onto a scatterplot we'll create with Matplotlib
ax.scatter(x=x, y=y, c=scale, s=np.abs(scale)*500)
ax.set(title="Some random data, created with JupyterLab!")
plt.show()
```

Below the code cell, a scatter plot titled "Some random data, created with JupyterLab!" is displayed. The plot shows a 2D scatter of data points with varying colors and sizes, representing the 3D data generated by the code. The y-axis is labeled with the number 2.

Keras in practice

- ❖ Jupyter notebooks:
- JupyterLab is the latest web-based interactive development environment for notebooks, code, and data





Keras in practice

4. Keras examples 1: Classifying movie reviews

- This project classifies movie reviews as positive or negative based on the text content of the reviews.
- The dataset used for this project is **IMDB** (Internet Movie Database).
- The **IMDB** dataset has 25,000 reviews for training and 25,000 reviews for testing, each set consisting of 50% negative and 50% positive reviews.
- The **IMDB** dataset comes packaged with Keras.
- The reviews (sequences of words) have been turned into sequences of integers, each integer stands for a specific word in a dictionary.



Keras in practice

4. Keras examples 1: Classifying movie reviews

- Load the dataset:

```
2 from keras.datasets import imdb
3 (train_data, train_labels), (test_data, test_labels) = imdb.load_data(
4     num_words=10000)
5 print(train_data[0])
```

- The argument `num_words=10000` means we'll only keep the top 10,000 most frequently occurring words in the training data.
- Each integer stands for a specific word in a dictionary.

```
C:\Users\Study\AppData\Local\Programs\Python\Python38\python.exe C:/Projects/Python4AI/iMDB.py
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100,
```



Keras in practice

4. Keras examples 1: Classifying movie reviews

- We can quickly decode one of these reviews back to English words:

```
word_index = imdb.get_word_index()
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
decoded_review = ' '.join(
    [reverse_word_index.get(i - 3, '?') for i in train_data[0]])
```

word_index is a dictionary mapping words to an integer index.

Reverses it, mapping integer indices to words

Decodes the review. Note that the indices are offset by 3 because 0, 1, and 2 are reserved indices for “padding,” “start of sequence,” and “unknown.”



Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Preparing the data

- ✓ We can't feed lists of integers into a neural network. We have to turn your lists into **tensors**.
- ✓ There are two ways to do that:
 - **Padding**: Lists are padded to the same length and converted into an integer tensor, which is then processed using an Embedding layer.
 - **One-hot encoding**: Lists are transformed into high-dimensional binary vectors, where only specific indices are set to 1. These vectors can be fed into a Dense layer.



Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Preparing the data

- ✓ Encoding the integer sequences into a binary matrix

```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

Creates an all-zero matrix
of shape (len(sequences),
dimension)

Sets specific indices
of results[i] to 1s

Vectorized training data

Vectorized test data



Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Preparing the data

- ✓ We should also vectorize your labels, which is straightforward:

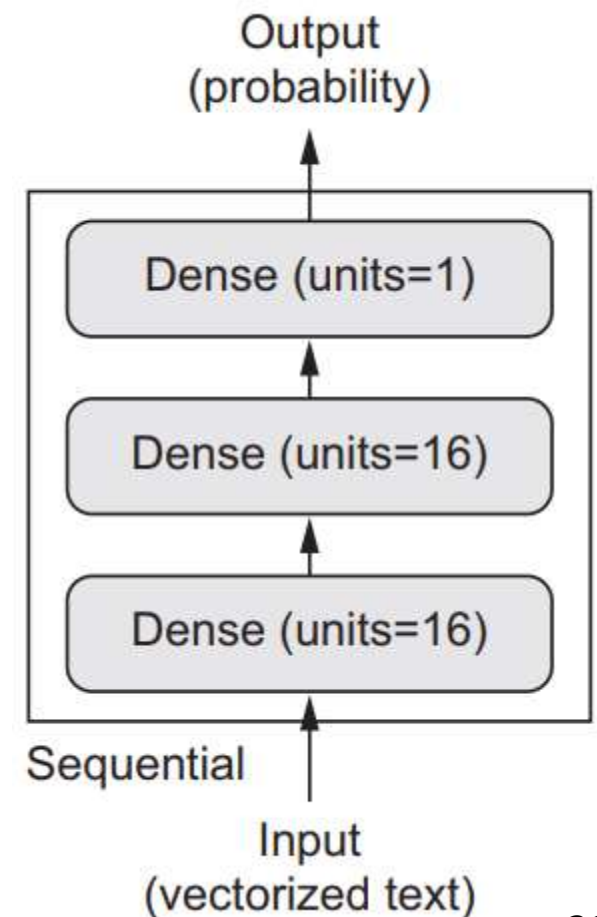
```
y_train = np.asarray(train_labels).astype('float32')  
y_test = np.asarray(test_labels).astype('float32')
```

Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Building the network:

- ✓ The three-layer network
- ✓ Having 16 hidden units means the weight matrix W will have shape `(input_dimension, 16)`: the dot product with W will project the input data onto a 16-dimensional representation space.
- ✓ Having more hidden units (a higher-dimensional representation space) allows the network to learn more-complex representations.

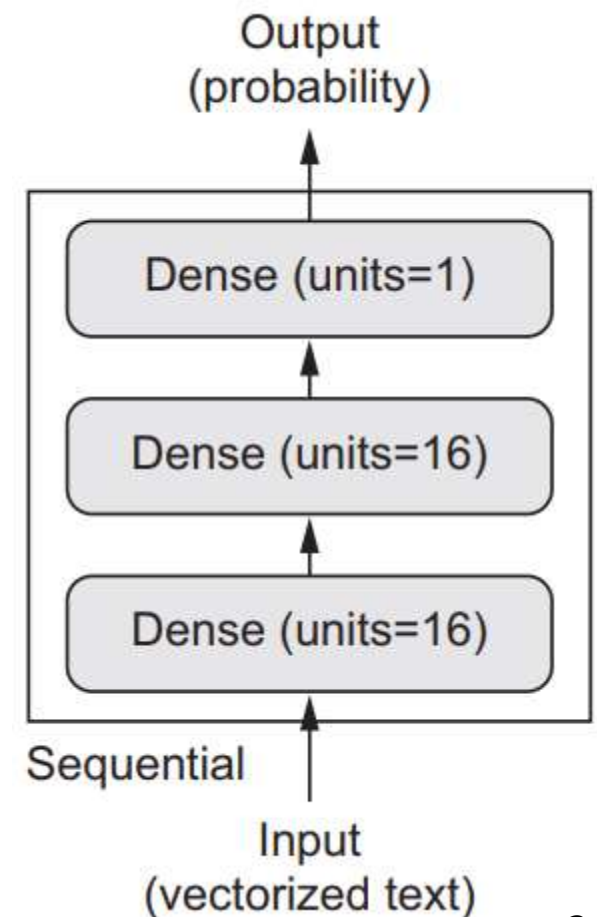


Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Building the network:

- ✓ Two intermediate layers with 16 hidden units each.
- ✓ A third layer that will output the scalar prediction regarding the sentiment of the current review.





Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Building the network:

```
from keras import models
from keras import layers

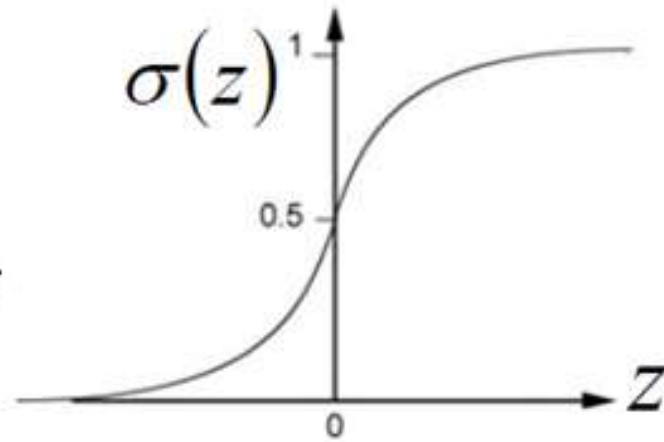
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

- ✓ The ReLU (rectified linear unit) is a function meant to zero out negative values.
- ✓ The sigmoid function “squashes” arbitrary values into the [0, 1] interval, outputting something that can be interpreted as a probability.

Keras in practice

❖ Sigmoid function

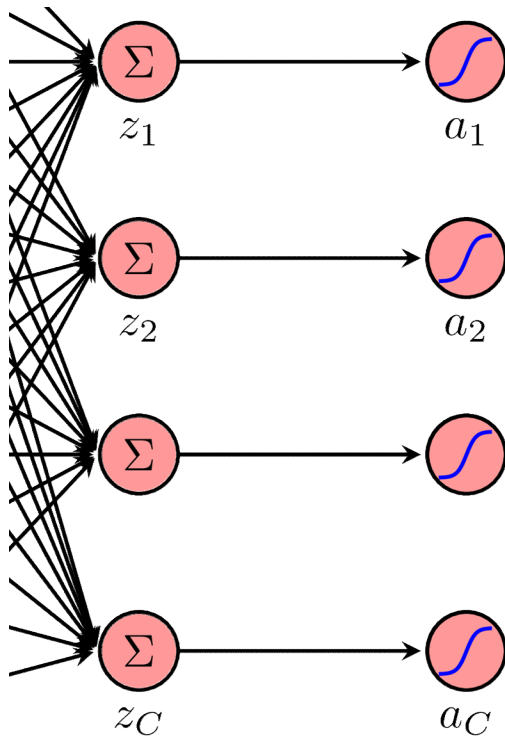
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Học sâu (deep learning)

❖ Softmax function

- Often used in the output layer to calculate the probability for each label.



- ✓ Create a relationship for the a_i values to determine the labels at the output:

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)}, \quad \forall i = 1, 2, \dots, C$$

$$\Rightarrow \sum_1^i a_i = 1$$

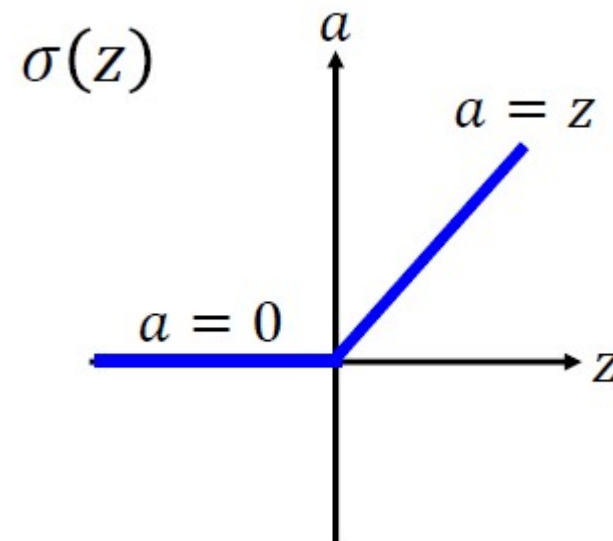
$$\mathbf{a} = \text{softmax}(\mathbf{z}) \in \mathbb{R}^C$$

- ✓ Calculate the probability for each a_i

Keras in practice

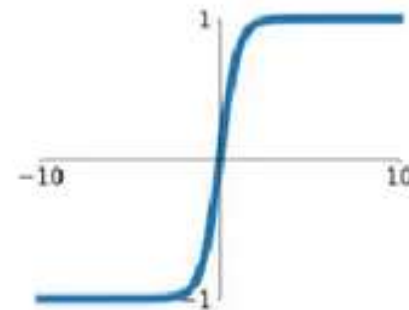
- ❖ ReLU (Rectified Linear Unit)
 - Quick calculation

$$f(z) = \max(0, z)$$



- ❖ \tanh

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





Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Compiling the model:

```
model.compile(optimizer='rmsprop',  
              loss='binary_crossentropy',  
              metrics=['accuracy'])
```

- ✓ **Crossentropy** is usually the best choice when we're dealing with models that output probabilities.
- ✓ **Crossentropy** is a quantity from the field of Information Theory that measures the distance between probability distributions or, in this case, between the ground-truth distribution and our predictions.



Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Training model:

```
47 # Validation data
48 x_val = x_train[:10000]
49 partial_x_train = x_train[10000:]
50 y_val = y_train[:10000]
51 partial_y_train = y_train[10000:]
52
53 # Training your model
54 history = model.fit(partial_x_train,
55                    partial_y_train,
56                    epochs=20,
57                    batch_size=512,
58                    validation_data=(x_val, y_val))
```



Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Training model:

- ✓ The call to `model.fit()` returns a History object. This object has a member `history`, which is a dictionary containing data about everything that happened during training.

```
history_dict = history.history  
print(history_dict.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Plotting the training and validation loss:

```
import matplotlib.pyplot as plt

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

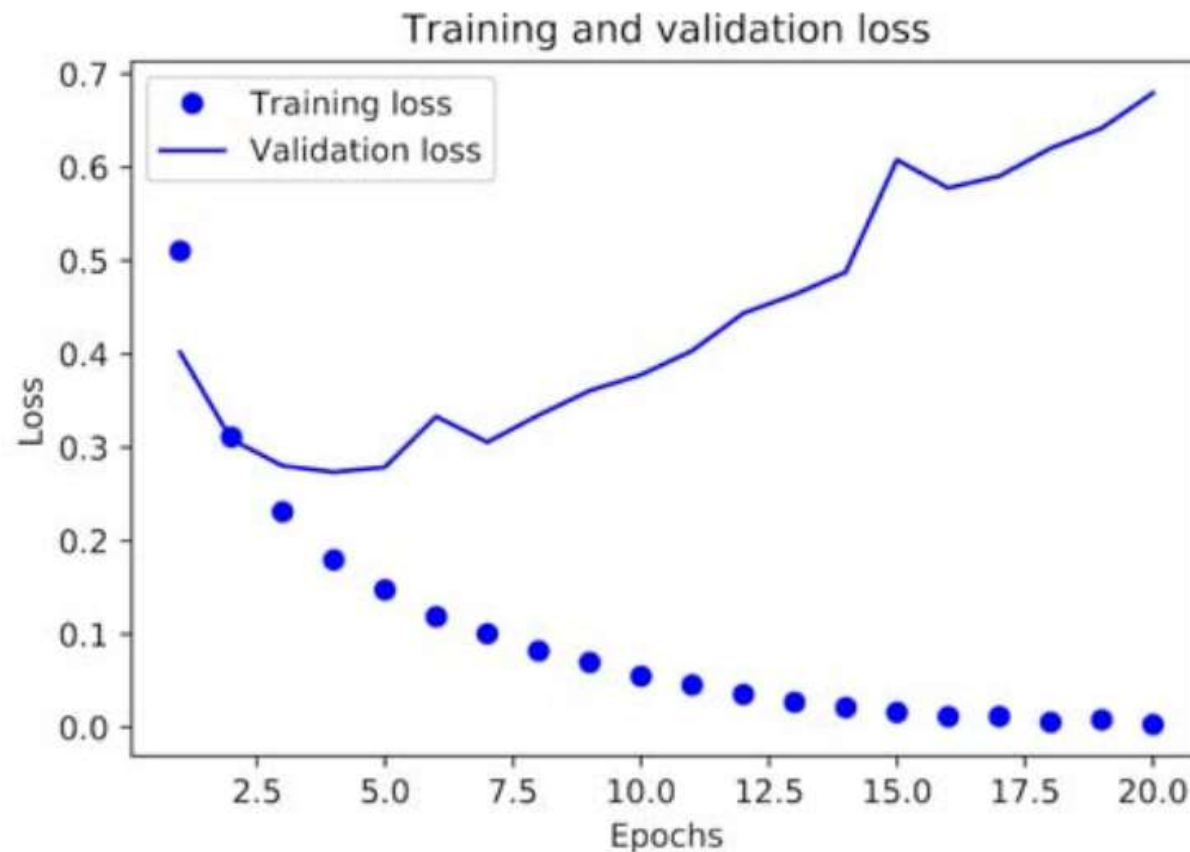
plt.show()
```

“bo” is for
“blue dot.”

“b” is for “solid
blue line.”

Keras in practice

4. Keras examples 1: Classifying movie reviews
 - Plotting the training and validation loss:





Keras in practice

4. Keras examples 1: Classifying movie reviews

- Plotting the training and validation accuracy:

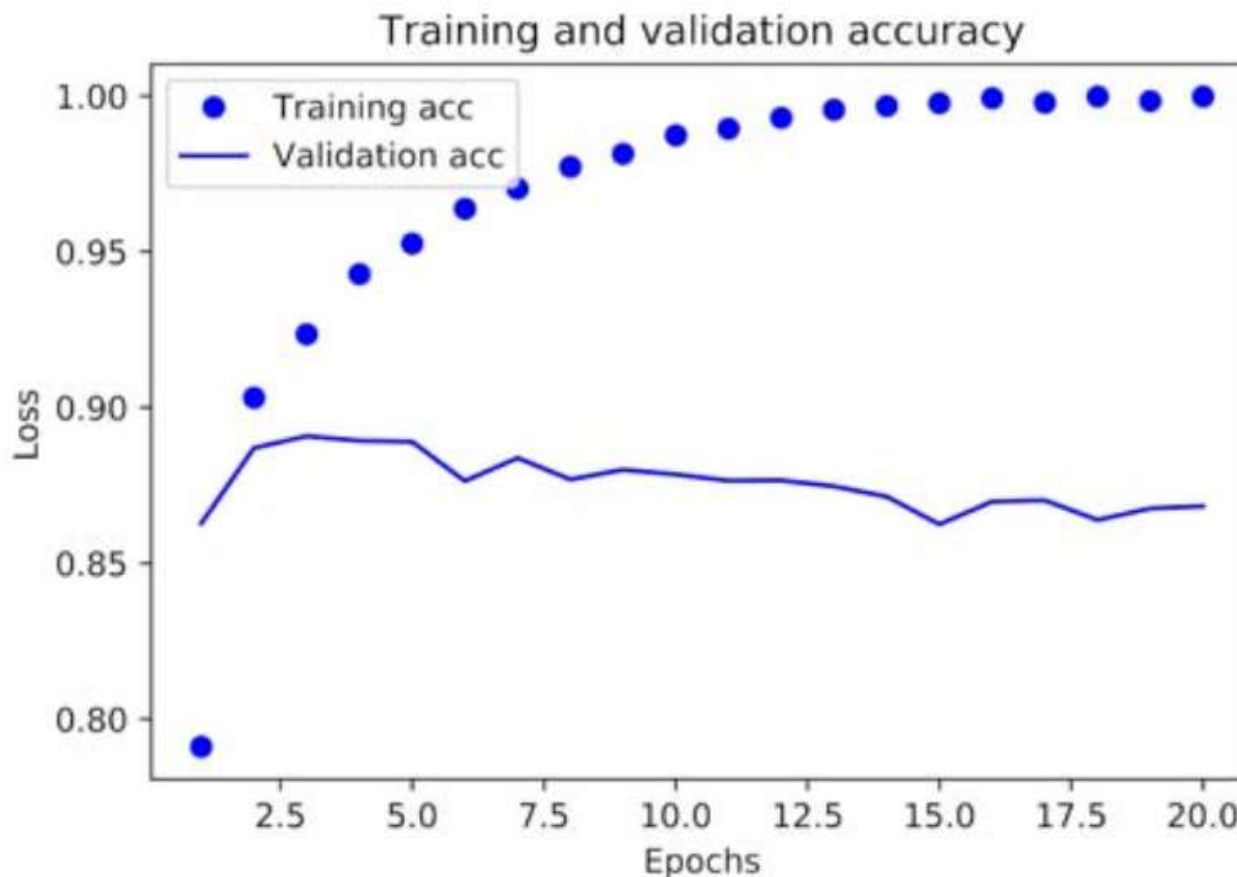
```
plt.clf()           ← Clears the figure
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

Keras in practice

4. Keras examples 1: Classifying movie reviews
 - Plotting the training and validation accuracy:





Keras in practice

4. Keras examples 1: Classifying movie reviews

➤ Test:

```
results = model.evaluate(x_test, y_test)
print(results)
```

```
[0.2929924130630493, 0.88327999999999995]
```