

Tensorflow的使用方法

该课程主要为大家讲授如下的内容：

- Tensorflow的一般 workflow
- 使用tensorflow完成数据预处理工作
- 使用Keras构建模型
- 模型导出和导入

1. tensorflow的一般 workflow

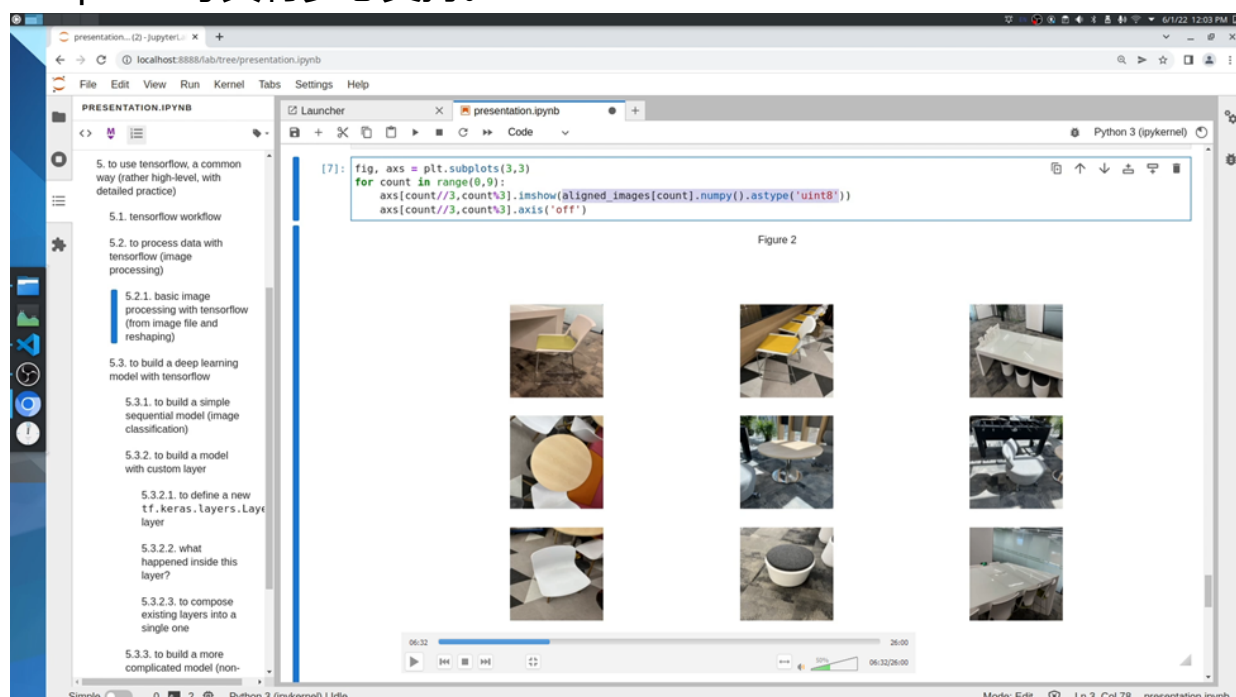
使用Tensorflow可以大致分为三个步骤，分别是：

1. 加载并处理数据
2. 构建，训练和复用模型
3. 部署模型

Tensorflow为这三个步骤分别提供了（1）数据预处理，（2）keras构建模型，和（3）导入导出模型的功能。

2. tensorflow的数据预处理

Tensorflow能够对图形、序列等类型的数据进行对齐、分割、大小变换等操作，且处理后的数据以tensorflow中的数据类型表示，对numpy、matplotlib等具有多态支持。



The screenshot shows a JupyterLab environment with a presentation.ipynb file open. The left sidebar contains a table of contents for the presentation, listing sections from 5.1 to 5.3.2.3. The main area displays the code cell for section 5.2.2, which imports tensorflow_datasets as tfds and lists the available builders. A red error message is visible in the output area, indicating a failure to retrieve a Google authentication token. The bottom status bar shows the current mode is 'Simple', with 0 lines and 2 cells, running on a Python 3 (ipykernel) kernel. The cursor is at line 1, column 21.

Section	Description
5.1	tensorflow workflow
5.2	to process data with tensorflow (image processing)
5.2.1	basic image processing with tensorflow (from image file and reshaping)
5.2.2	load data from tfds
5.3	to build a deep learning model with tensorflow
5.3.1	to build a simple sequential model (image classification)
5.3.2	to build a model with custom layer
5.3.2.1	to define a new tf.keras.layers.Layer
5.3.2.2	what happened inside this layer?
5.3.2.3	to compose existing layers into a single one

```
[8]: import tensorflow_datasets as tfds

[9]: tfds.list_builders()

2022-06-01 12:18:53.960888: W tensorflow/core/platform/cloud/google_auth_provider.cc:184] All attempts to get a Google authentication bearer token failed, returning an empty token. Retrieving token from files failed with "Not found: Could not locate the credentials file.". Retrieving token from GCE failed with "Failed precondition: Error executing an HTTP request: libcurl code 6 meaning 'Could not resolve host name', error details: Could not resolve host: metadata".

[9]: ['abstract_reasoning',
      'accentdb',
      'aestlc',
      'ar1w2k3d',
      'ag_news_subset',
      'ai2_arc',
      'ai2_arc_with_ir',
      'amazon_us_reviews',
      'anli',
      'arc',
      'asset',
      'assin2',
      'bair_robot_pushing_small',
      'bccd',
      ...]
```

5.3. to build a deep learning model with tensorflow

It is a matter of fact that you could define all the components of a neural network, like layers and gradient, from scratch with the reusable classes in tensorflow. You could learn the ways to make it on tensorflow guide. Yet you are not going to do that, as you are going to focus on a rather high-level programming, faster and more

1. 使用Keras构建模型

presentation.ipynb

localhost:8888/lab/tree/presentation.ipynb#to-migrate-model-with-tensorflow

File Edit View Run Kernel Tabs Settings Help

PRESENTATION.IPYNB

presentation.ipynb

Python 3 (ipykernel)

5.3. to build a deep learning model with tensorflow

It is a matter of fact that you could define all the components of a neural network, like layers and gradient, from scratch with the reusable classes in tensorflow. You could learn the way to make it on [tensorflow guide](#). Yet we are not going to do that, as we are going to focus on a rather high-level programming, faster and more efficiently. To make it we would learn about 'tf.keras', in a practical way. Here I would not introduce Keras as a separated project since we are going to make use of it integrated within tensorflow. You may wanna learn more about Keras at [its page](#).

```
[20]: # ever since this session displayed large scales of computing
      # set the debugging log not to print

      tf.debugging.set_log_device_placement(False)
      tf.compat.v1.disable_eager_execution()
```

5.3.1. to build a simple sequential model (image classification)

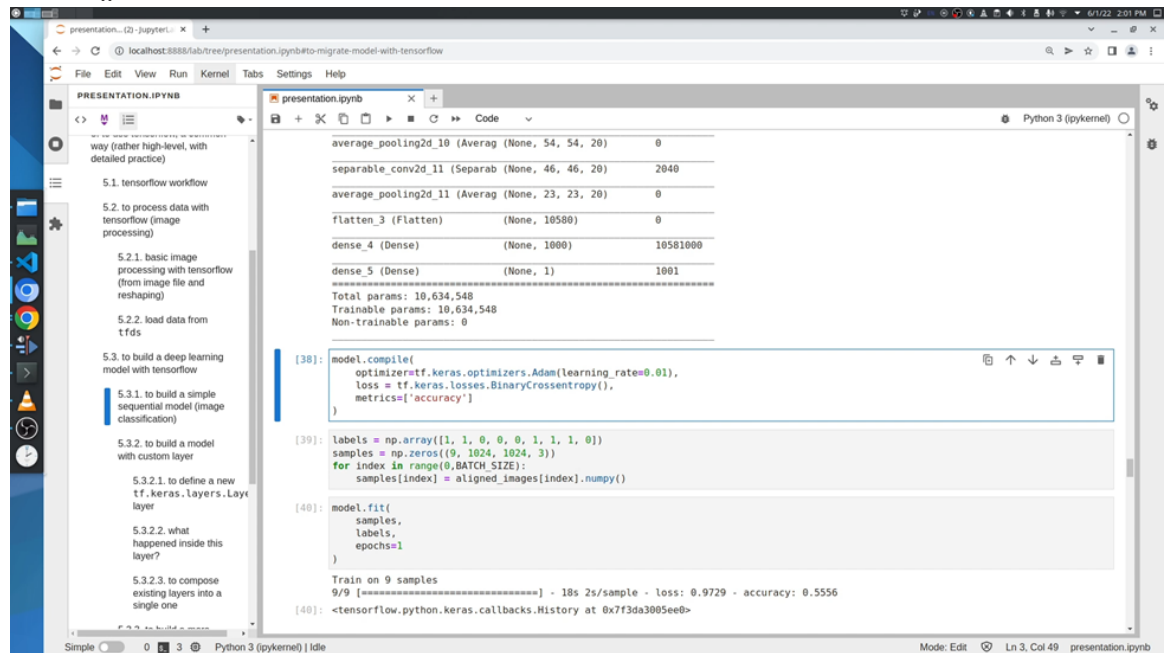
```
[37]: model = tf.keras.Sequential([
      tf.keras.layers.SeparableConv2D(
          filters=20,
          kernel_size=(127,127),
          activation='relu',
          input_shape=(1024,1024,3)
      ),
      tf.keras.layers.AvgPool2D(
          pool_size=(4, 4),
      ),
      tf.keras.layers.SeparableConv2D(filters=20,kernel_size = (9,9),activation='relu'),
      tf.keras.layers.AvgPool2D(4,4),
      tf.keras.layers.SeparableConv2D(filters=20,kernel_size = (9,9),activation='relu'),
      tf.keras.layers.AvgPool2D(2,2),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(units=1000,activation='relu'),
      tf.keras.layers.Dense(units=1,activation='softmax')
  ])

  model.summary()
```

Simple 0 1 3 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 presentation.ipynb

Tensorflow中将深度学习模型拆分为大的组件并高度封装，只需要调用模型对象的.compile()方法就可以为其添加。然后，通

过.fit()方法即可进行训练。



The screenshot shows a Jupyter Notebook with a sidebar on the left containing a table of contents. The main area displays a Keras model summary and training code. The model summary is as follows:

Layer	Output Shape	Param Count
average_pooling2d_10 (Averag	(None, 54, 54, 20)	0
separable_conv2d_11 (Separab	(None, 46, 46, 20)	2040
average_pooling2d_11 (Averag	(None, 23, 23, 20)	0
flatten_3 (Flatten)	(None, 10580)	0
dense_4 (Dense)	(None, 1000)	10581000
dense_5 (Dense)	(None, 1)	1001

Total params: 10,634,548
Trainable params: 10,634,548
Non-trainable params: 0

The code in the notebook is as follows:

```
[38]: model.compile(
      optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
      loss=tf.keras.losses.BinaryCrossentropy(),
      metrics=['accuracy'])

[39]: labels = np.array([1, 1, 0, 0, 0, 1, 1, 1, 0])
      samples = np.zeros((9, 1024, 1024, 3))
      for index in range(0, BATCH_SIZE):
          samples[index] = aligned_images[index].numpy()

[40]: model.fit(
      samples,
      labels,
      epochs=1)

Train on 9 samples
9/9 [=====] - 18s 2s/sample - loss: 0.9729 - accuracy: 0.5556
[40]: <tensorflow.python.keras.callbacks.History at 0x7f3da3005ee6>
```

3. 自定义layer

继承tf.keras.layer.Layer父类或设定现成tf.keras.layer命名空间下的layer类顺序输入和输出，可以得到自定义的layer类，像其它现成的layer类一样使用。

The first screenshot shows the Jupyter Notebook interface with the file explorer on the left. The notebook is titled 'presentation.ipynb'. The code cell [27] defines a custom layer `MyDenseLayer` that inherits from `tf.keras.layers.Layer`. It includes methods for `__init__`, `build`, and `call`. The code cell [34] creates a `custom_model` using `tf.keras.Sequential` with a `tf.keras.layers.SeparableConv2D` layer.

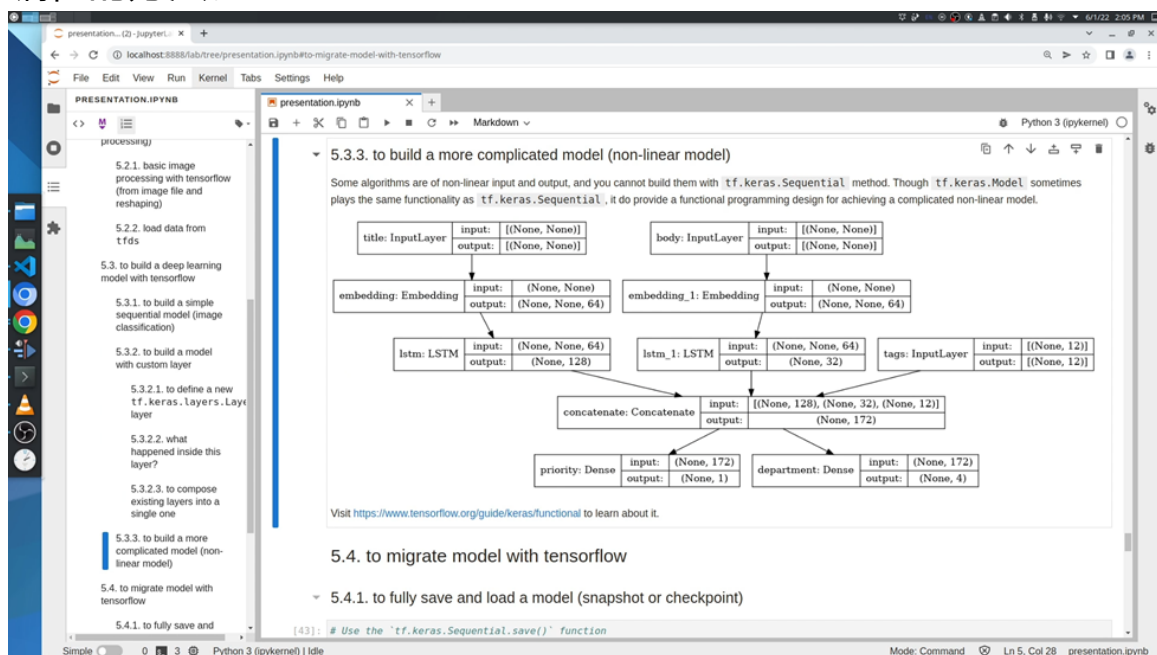
The second screenshot shows the Jupyter Notebook interface with the file explorer on the left. The notebook is titled 'presentation.ipynb'. The code cell [31] defines a custom layer `DoubleDenseLayer` that inherits from `tf.keras.Model`. It includes methods for `__init__`, `build`, and `call`. The code cell [33] creates a `custom_model` using `tf.keras.Sequential` with a `tf.keras.layers.SeparableConv2D` layer.

The third screenshot shows a diagram of a neural network architecture. The diagram illustrates a complex non-linear model with multiple layers and inputs. The layers are: `title: InputLayer`, `body: InputLayer`, `embedding: Embedding`, `embedding_1: Embedding`, `lstm: LSTM`, `lstm_1: LSTM`, and `tags: InputLayer`. The diagram shows the flow of data from the inputs through the layers, with the final output being `input: [(None, 128), (None, 32), (None, 12)]`.

4. 通过函数式编程来解决非线性输入/输出的模型

面对图形分割或看图理解等任务时，编码器和解码器中可能存在非线性的输入或输出。（也就是梯度不相关的layer结构）此时，之前介绍的OOP编程已经不能解决这种设计，而需要采用函数式

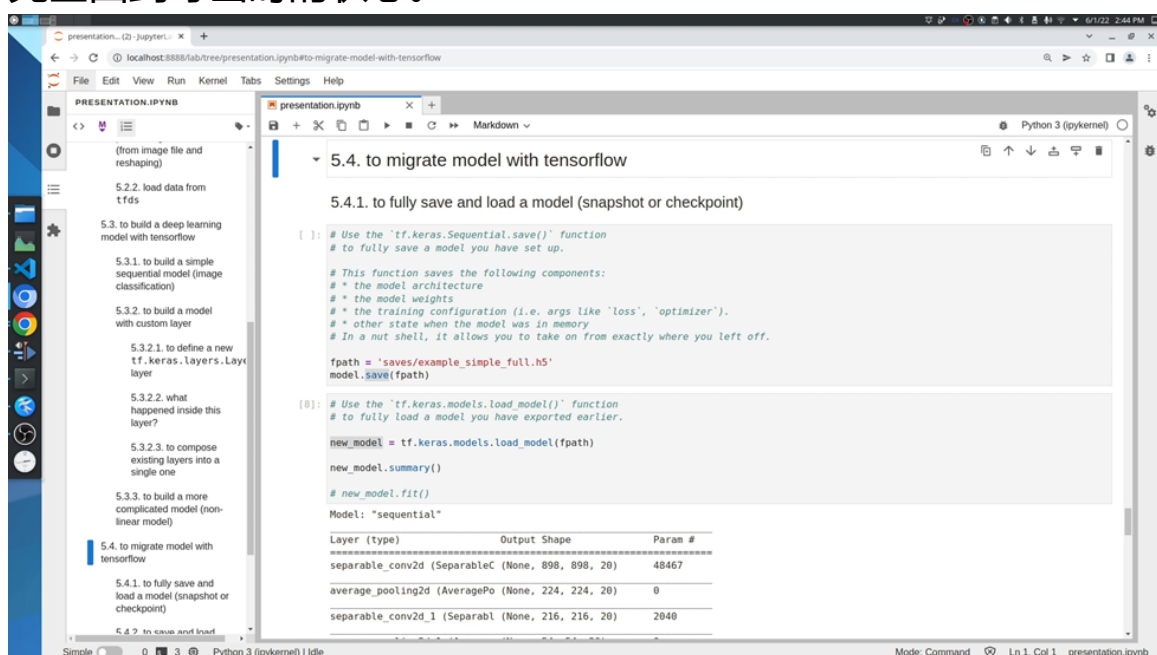
编程的方法。



4. 导入/导出tensorflow模型

1. 完全导出（快照）

Sequential或Model类对象可以通过.save()方法将当前的模型状态完全导出，相当于为模型保存了一份快照。在导入时，模型能够完全回到导出时的状态。



2. 导出结构

Tensorflow支持单独导出模型结构；模型结构以JSON格式保存。相比完全导出，导出结构的文件占用小、并且具有高度序列化的格式，可以被迁移使用于模型可视化、或同训练好的模型参数结

合成为预训练模型。

The screenshot shows a Jupyter Notebook with two tabs. The left tab, 'presentation.ipynb', contains a table of contents and a list of topics. The right tab, 'example_simple.json', displays the JSON representation of a Keras model architecture. The JSON file is a dictionary with keys for 'class_name', 'config', and 'layers'. The 'config' key contains a dictionary with 'name', 'layers', and 'input_shape'. The 'layers' key contains a list of layer configurations, including 'InputLayer', 'Conv2D', and 'MaxPooling2D'. The 'input_shape' is [28, 28, 1, 1]. The 'layers' list includes an 'InputLayer' with 'input_shape' [28, 28, 1, 1], a 'Conv2D' layer with 'filters' [128, 128, 1, 1], and a 'MaxPooling2D' layer with 'strides' [2, 2].

```
[ ]: # Export the architecture of the model to JSON
# with the function
# tf.keras.Sequential.to_json()

fpath = 'saves/example_simple.json'

fp = open(fpath, 'w')
fp.write(model.to_json())
fp.close()

[ ]: # Import the model architecture from earlier
# with the function
# tf.keras.models.model_from_json()

if os.isfile(fpath) is True:
    fp = open(fpath, 'r')

    new_model_from_json = tf.keras.models.model_from_json(
        fp.read())
    fp.close()

    print(new_model_from_json.summary())

    new_model_from_json.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy'])

    new_model_from_json.fit()
```

3. 导出参数

Tensorflow还支持单独导出模型中的所有参数。这往往是训练完成的模型需要部署到预训练的模型上，或发布离线模型等。

The screenshot shows a Jupyter Notebook with two tabs. The left tab, 'presentation.ipynb', contains a table of contents and a list of topics. The right tab, 'example_simple.json', displays the JSON representation of a Keras model architecture. The JSON file is a dictionary with keys for 'class_name', 'config', and 'layers'. The 'config' key contains a dictionary with 'name', 'layers', and 'input_shape'. The 'layers' key contains a list of layer configurations, including 'InputLayer', 'Conv2D', and 'MaxPooling2D'. The 'input_shape' is [28, 28, 1, 1]. The 'layers' list includes an 'InputLayer' with 'input_shape' [28, 28, 1, 1], a 'Conv2D' layer with 'filters' [128, 128, 1, 1], and a 'MaxPooling2D' layer with 'strides' [2, 2].

```
[ ]: # Use the 'tf.keras.Sequential.save_weights()' function
# to only save the weights of a model you have set up.

fpath = 'saves/example_simple_weights.h5'
model.save_weights(fpath)

[ ]: # Use the 'tf.keras.models.load_model()' function
# to fully load a model you have exported earlier.

new_model_from_json.load_weights(fpath)
```

6. examples

6.1. A simple LSTM neural network for manipulating IMDB samples

```
[ ]: VOCAB_SIZE = 88584
MAXLEN = 250
BATCH_SIZE = 64

(train_data, train_labels), (test_data, test_labels) = tf.keras.datasets.imdb.load_data(num_words = VOCAB_SIZE)

[ ]: print(len(train_data[0]))
print(len(train_data[2]))

[ ]: # As the samples are of varied length,
# we have to make them in the same length,
# else we will get an error.

# The full sequence is padded with zeros to the maximum length we set
```