

## 1. 用于深度学习的高级 API

### 1.1 Sequential API



Sequential API 是定义神经网络模型最常用的方法。它对应于我们构建深度学习神经网络时的直观印象：堆叠的网络层次序列。

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

# 加载数据集
mnist = datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# 构建一个神经网络模型
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(512, activation=tf.nn.relu))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(10, activation=tf.nn.softmax))
model.compile(optimizer='adam', \
              loss='sparse_categorical_crossentropy', \
              metrics=['accuracy'])

# 训练和评估模型
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

### 1.2 Functional API

Functional API (<https://keras.io/getting-started/functional-api-guide/>) 使工程师能够定义复杂拓扑，包括多输入和多输出模型，以及具有共享层的高级模型和具有残差连接的模型。

```
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.models import Model

# 加载数据集
inputs = tf.keras.Input(shape=(28, 28))
x = Flatten()(inputs)
x = Dense(512, activation='relu')(x)
x = Dropout(0.2)(x)
predictions = Dense(10, activation='softmax')(x)
model = Model(inputs=inputs, outputs=predictions)

# 编译、训练和评估模型
```

# Show Me AI

如果数据集与原始数据集没有显著分布差异，则迁移学习和微调预训练模型 (<https://keras.io/applications/>) 可以节省时间。

在执行给定的迁移学习代码后，可以使 MobileNet V2 层训练，并对结果模型进行微调，以获得更好的结果。

```
import tensorflow as tf
import tensorflow_datasets as tfds

dataset = tfds.load(name='tf_flowers', as_supervised=True)
NUMBER_OF_CLASSES_IN_DATASET = 5
IMG_SIZE = 160

def preprocess_example(image, label):
    image = tf.cast(image, tf.float32)
    image = (image / 127.5) - 1
    image = tf.image.resize(image, (IMG_SIZE, IMG_SIZE))
    return image, label

DATASET_SIZE = 3670
BATCH_SIZE = 32
train = dataset['train'].map(preprocess_example)
train_batches = train.shuffle(DATASET_SIZE).batch(BATCH_SIZE)

# 加载 MobileNetV2 预训练模型
model = tf.keras.applications.MobileNetV2(
    input_shape=(IMG_SIZE, IMG_SIZE, 3), \
    include_top=False, weights='imagenet', \
    pooling='avg')
model.trainable = False

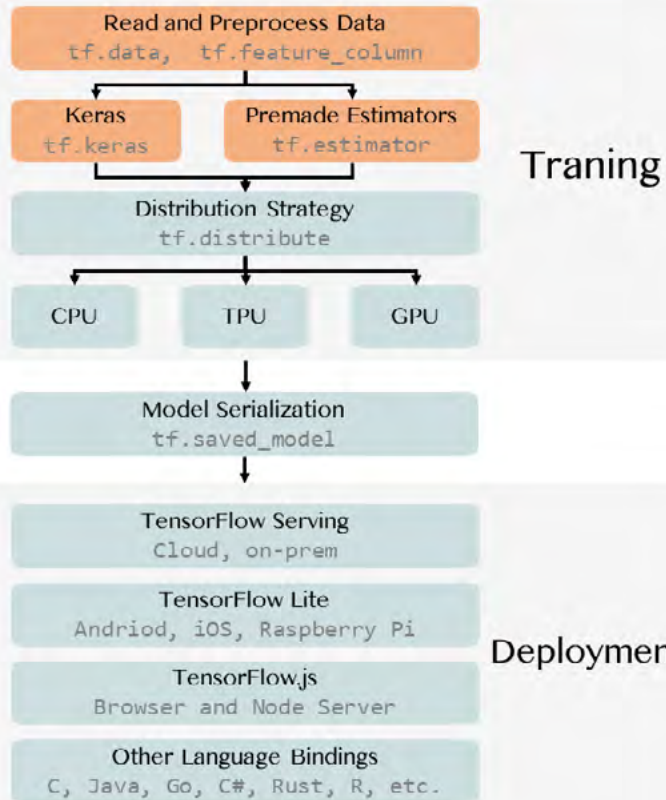
# 添加新层次以用于多分类
new_output = tf.keras.layers.Dense(NUMBER_OF_CLASSES_IN_DATASET, \
    activation='softmax')
new_model = tf.keras.Sequential([model, new_output])
new_model.compile(loss=tf.keras.losses.categorical_crossentropy, \
    optimizer=tf.keras.optimizers.RMSprop(lr=1e-3), \
    metrics=['accuracy'])

# 训练分类器
new_model.fit(train_batches.repeat(), epochs=10, \
    steps_per_epoch=DATASET_SIZE // BATCH_SIZE)
```

### 1.3 预训练模型

## 3. 机器学习 workflow 示例

## Machine Learning Workflow



- 01 使用 `tf.data` 创建的管道 ([https://www.tensorflow.org/alpha/guide/data\\_performance](https://www.tensorflow.org/alpha/guide/data_performance)) 加载训练数据。输入可以使用内存中数据 (NumPy)、本地文件或远程数据。
- 02 使用 `tf.keras` 搭建、训练和验证模型，或使用预制估计器。
- 03 快速运行和调试，然后使用 `tf.function` 获得计算图的优势。
- 04 对于大型 ML 训练任务，使用 Distribution Strategy API ([https://www.tensorflow.org/guide/distribute\\_strategy](https://www.tensorflow.org/guide/distribute_strategy)) 在本地或云环境中的 Kubernetes 集群上部署训练。
- 05 导出到 SavedModel ([https://www.tensorflow.org/alpha/guide/saved\\_model](https://www.tensorflow.org/alpha/guide/saved_model)) —— TensorFlow 服务、TensorFlow Lite、TensorFlow.js 等的交换格式。

## 3.1 tf.data

`tf.data` API (<https://www.tensorflow.org/guide/datasets>) 允许从简单的片段构建复杂的输入管道。管道聚合来自分布式文件系统的数据，对每个对象应用转换，并将乱序 (shuffle) 后的样本合并到训练批 (batch) 中。

`tf.data.Dataset` 表示一系列数据元素，每个元素包含一个或多个张量对象。这可以用一对表示图像的张量和相应的类标签来说明。

```
import tensorflow as tf
```

```
DATASET_URL = "https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.data.gz"
```

```
DATASET_SIZE = 387698
```

```
dataset_path = tf.keras.utils.get_file(fname=DATASET_URL.split('/')[-1], origin=DATASET_URL)
```

```
COLUMN_NAMES = ['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology', 'Vertical_Distance_To_Hydrology',  
                'Horizontal_Distance_To_Roadways', 'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',  
                'Horizontal_Distance_To_Fire_Points', 'Soil_Type', 'Cover_Type']
```

```
def _parse_line(line):
```

```
    fields = tf.io.decode_csv(records=line, record_defaults=[0.0] * 54 + [0]) # 对 csv 行进行解码
```

```
    features = dict(zip(COLUMN_NAMES, fields[:10] + [tf.stack(fields[14:54])] + [fields[-1]])) # 打包成字典格式
```

```
    class_label = tf.argmax(fields[10:14], axis=0) # 抽取 one-hot 编码后的类别中对应的类别标签
```

```
    return features, class_label
```

```
def csv_input_fn(csv_path, test=False, batch_size=DATASET_SIZE // 1000):
```

```
    dataset = tf.data.TextLineDataset(filename=csv_path, compression_type='GZIP') # 构建包含 csv 行的数据集
```

```
    dataset = dataset.map(_parse_line) # 解析每一行
```

```
    dataset = dataset.shuffle(buffer_size=DATASET_SIZE, seed=42) # 对训练集和测试集进行乱序、分拆成 batch
```

```
TEST_SIZE = DATASET_SIZE // 10
```

```
return dataset.take(TEST_SIZE).batch(TEST_SIZE) if test else dataset.skip(TEST_SIZE).repeat().batch(batch_size)
```



扫码回复“工具库”  
下载最新全套资料

## TensorFlow2 速查表 (Keras &amp; Estimator)

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

参考 | ALTOROS



## 3.2 tf.feature\_column

`tf.feature_column` ([https://www.tensorflow.org/guide/feature\\_columns](https://www.tensorflow.org/guide/feature_columns)) 命名空间中的函数用于将原始数据放入 TensorFlow 数据集中。

`feature` 列是用于获取和表示特性的高级配置抽象。它不包含任何数据，但告诉模型如何转换原始数据，使其符合预期。

要选择的确切要素列取决于要素类型和模型类型。连续特征类型由 `numeric_column` 处理，可直接输入神经网络或线性模型。

## Read Dataset

```
def csv_input_fn(csv_path, ...):
```

```
    """
```

```
    {
        "Elevation": [...],
        "...": [...],
        "Soil_Type": [[...]]
    },
```

```
    [...]
```

batch\_size

feature

labels

## Define Feature Columns

```
feature_column = [
    numeric_column("Elevation"),
    ...
    categorical_column_with_identity("Cover_Type", ...)
    numeric_column("Soil_Type", shape=(40,))
]
```

Match feature names from csv\_input\_fn()

## Build Model

```
Model = LinearClassifier(feature_column, ...)
```

Bridge input to model

具有“`categorical_column`”前缀的函数可以处理分类特征，但它们需要通过 `embedding_column` 或 `indicator_column` 进行包装，然后才能输入到神经网络模型中。对于线性模型，直接传入分类列时，指示符列是一种内部表示形式。

```
feature_columns = [tf.feature_column.numeric_column(name)
                    for name in COLUMN_NAMES[:10]]
feature_columns.append(tf.feature_column.categorical_column_with_identity('Cover_Type', num_buckets=8))
```

# Soil\_type[1-40] is a tensor of length 40

```
feature_columns.append(tf.feature_column.numeric_column('Soil_Type', shape=(40,)))
```

## 3.3 tf.estimator

`estimator` API (<https://www.tensorflow.org/guide/estimators>) 为最佳实践提供了高级封装：模型训练、评估、预测和导出服务。`tf.estimator.Estimator` 子类表示一个完整的模型。

`estimator` 对象用于创建和管理 `tf.Graph` 和 `tf.Session`。

Premade estimators ([https://www.tensorflow.org/guide/premade\\_estimators](https://www.tensorflow.org/guide/premade_estimators)) 包括线性分类器、DNN 分类器和梯度增强树。`BaselineClassifier` 和 `BaselineRegressor` 将帮助建立一个简单的模型，以便在进一步的模型开发过程中进行健全性检查。

# Build, train, and evaluate the estimator

```
model = tf.estimator.LinearClassifier(feature_columns, \
                                     n_classes=4)
model.train(input_fn=lambda: csv_input_fn(dataset_path), \
            steps=10000)
model.evaluate(input_fn=lambda: csv_input_fn(dataset_path, \
                                             test=True))
```



Keras (<https://keras.io/>) 是一个方便的高级 API 标准，用于深度学习模型，广泛用于快速原型和最先进的研究。Keras 最初设计为在不同的低级计算框架上运行，而 TensorFlow 平台完全实现了这一点 (<https://www.tensorflow.org/guide/keras>)。

Jupyter Notebook (<https://jupyter.org/>) 是一个基于网页的交互式计算环境，用于数据科学和科学计算。Google Colaboratory (<https://colab.research.google.com/>) 是一个免费的笔记本电脑环境，不需要安装，完全在云端运行，可以用于机器学习项目的起步阶段。

### 3.4 tf.saved\_model

**SavedModel** ([https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved\\_model/README.md](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md)) 包含一个完整的 TF 程序，不需要运行原始的模式构建代码，这对于部署和共享模型非常有用。

*# Export model to SavedModel*

```
_builder = tf.estimator.export.build_parsing_serving_input_receiver_fn
_spec_maker = tf.feature_column.make_parse_example_spec
serving_input_fn = _builder(_spec_maker(feature_columns))
export_path = model.export_saved_model("/tmp/from_estimator/", serving_input_fn)
```

下面的代码示例演示如何通过 Python 加载和使用保存的模型。

```
imported = tf.saved_model.load(export_path) # 加载 SavedModel 模式存储的模型

def predict(new_object): # 使用加载后的模型用于预估
    example = tf.train.Example()

    for column in COLUMN_NAMES[:-2]: # 常规连续值特征
        val = new_object[column]
        example.features.feature[column].float_list.value.extend([val])

    for val in new_object['Soil_Type']: # 40 列的 One-hot 编码列
        example.features.feature['Soil_Type'].float_list.value.extend([val])

    # Categorical column with ID
    example.features.feature['Soil_Type'].int64_list.value.extend([new_object['Soil_Type']])
    return imported.signatures['predict'](examples=tf.constant([example.SerializeToString()]))

predict({'Elevation': 2296, 'Aspect': 312, 'Slope': 27, 'Horizontal_Distance_To_Hydrology': 256, \
        'Horizontal_Distance_To_Fire_Points': 836, 'Horizontal_Distance_To_Roadways': 1273, \
        'Vertical_Distance_To_Hydrology': 145, 'Hillshade_9am': 136, 'Hillshade_Noon': 208, \
        'Hillshade_3pm': 206, \
        'Soil_Type': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, ], \
        'Cover_Type': 6})
```



## 数据科学工具库速查表



**NumPy** 是 Python 数据科学计算的核心库，提供了高性能多维数组对象及处理数组的工具。使用以下语句导入 NumPy 库：

```
import numpy as np
```



**SciPy** 是基于 NumPy 创建的 Python 科学计算核心库，提供了众多数学算法与函数。



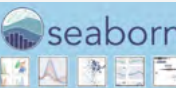
**Pandas** 是基于 NumPy 创建的 Python 库，为 Python 提供了易于使用的的数据结构和数据分析工具。使用以下语句导入：

```
import pandas as pd
```



**Matplotlib** 是 Python 的二维绘图库，用于生成符合出版质量或跨平台交互环境的各类图形。

```
import matplotlib.pyplot as plt
```



**Seaborn** 是基于 matplotlib 开发的高阶 Python 数据可视图库，用于绘制优雅、美观的统计图形。使用下列别名导入该库：

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```



**Bokeh** 是 Python 的交互式可视图库，用于生成在浏览器里显示的大规模数据集高性能可视图。Bokeh 的中间层通用 **bokeh.plotting** 界面主要为两个组件：数据与图示例。

```
from bokeh.plotting import figure
```

```
from bokeh.io import output_file, show
```

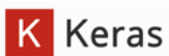


**PySpark** 是 Spark 的 Python API，允许 Python 调用 Spark 编程模型。Spark SQL 是 Apache Spark 处理结构化数据模块。

## AI 垂直领域工具库速查表



**Scikit-learn** 是开源的 Python 库，通过统一的界面实现机器学习、预处理、交叉验证及可视化算法。



**Keras** 是强大、易用的深度学习库，基于 Theano 和 TensorFlow 提供了高阶神经网络 API，用于开发和评估深度学习模型。



“TensorFlow™ is an open source software library for numerical computation using data flow graphs.” **TensorFlow** 是 Google 公司开发的机器学习架构，兼顾灵活性和扩展性，既适合用于工业生产也适合用于科学研究。



**PyTorch** 是 Facebook 团队 2017 年初发布的深度学习框架，有利于研究人员、爱好者、小规模项目等快速搞出原型。**PyTorch** 也是 Python 程序员最容易上手的深度学习框架。



**Hugging Face** 以开源的 NLP 预训练模型库 **Transformers** 而广为人知，目前 GitHub Star 已超过 54000+。**Transformers** 提供 100+ 种语言的 32 种预训练语言模型，简单，强大，高性能，是新手入门的不二选择。



**OpenCV** 是一个跨平台计算机视觉库，由 C 函数 /C++ 类构成，提供了 Python、MATLAB 等语言的接口。**OpenCV** 实现了图像处理和计算机视觉领域的很多通用算法。

## 编程语言速查表



**SQL** 是管理关系数据库的结构化查询语言，包括数据的增删查改等。作为数据分析的必备技能、岗位 JD 的重要关键词，SQL 是技术及相关岗位同学一定要掌握的语言。



**Python** 编程语言简洁快速、入门简单且功能强大，拥有丰富的第三方库，已经成为大数据和人工智能领域的主流编程语言。

More...

## AI 知识技能速查表



**Jupyter Notebook** 交互式计算环境，支持运行 40+ 种编程语言，可以用来编写漂亮的交互式文档。这个教程把常用的基础功能讲解得很清楚，对新手非常友好。



**正则表达式** 非常强大，能匹配很多规则的文本，常用于文本提取和爬虫处理。这也是一门令人难以捉摸的语言，字母、数字和符号堆在一起，像极了“火星文”。

More...



ShowMeAI 速查表 (©2021)

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

数据科学工具库速查表

扫码回复“数据科学”

获取最新全套速查表

AI 垂直领域工具库速查表

扫码回复“工具库”

获取最新全套速查表

编程语言速查表

扫码回复“编程语言”

获取最新全套速查表

AI 知识技能速查表

扫码回复“知识技能”

获取最新全套速查表