

Tensorflow 2.0 基本运用 - 图片分类

- 写本篇文章的目的是梳理自己在学习 tensorflow 2.0 时的知识点以及编程思路，也为了方便他人在学习 tensorflow 2.0 时能够有一份参考资料，能快速入门 tensorflow 2.0 。
- 本文将通过三节进行讲解：
 - 第一节：简单的介绍 tensorflow
 - 第二节：介绍 tensorflow 2.0 中常见的 API
 - 第三节：通过一个实例，来具体讲解如何使用 tensorflow 2.0 进行编程

一. Tensorflow 介绍

1. Tensorflow 简介

TensorFlow™ 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Nodes）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。它灵活的架构让你可以在多种平台上展开计算，例如台式计算机中的一个或多个 CPU（或 GPU），服务器，移动设备等等。TensorFlow 最初由 Google 大脑小组（隶属于 Google 机器智能研究机构）的研究员和工程师们开发出来，用于机器学习和深度神经网络方面的研究，但这个系统的通用性使其也可广泛用于其他计算领域。

2. Tensorflow 2.0 版本简介

Tensorflow 2.0 版本相比 1.0 版本不是简单得更新，而是一次重大升级。简单地来说，TensorFlow 2.0 默认采用 eager 执行模式，而且重整了很多混乱的模块。毫无疑问，2.0 版本将会逐渐替换 1.0 版本，所以很有必要趁早入手 TensorFlow 2.0。这篇文章将简明扼要地介绍 TensorFlow 2.0，以求快速入门。

二. Tensorflow 2.0 重点 API 介绍

- 本节主要讲解 tensorflow 2.0 使用时，常见且重要的 API，包括 Sequential(), compile(), fit(), evaluate(), save_model(), load_model(), predict()。通过这些 API，我们就可以构建简单的深度学习框架。

1. Sequential()

(1) 作用：

- 用于创建一个 Sequential 模型。
- Sequential 模型是一个线性的层堆栈。

2. compile()

(1) 作用：

- 用于配置训练模型。

(2) 函数原型：

```
compile(  
    optimizer='rmsprop',  
    loss=None,  
    metrics=None,  
    loss_weights=None,  
    sample_weight_mode=None,  
    weighted_metrics=None,  
    target_tensors=None,  
    distribute=None,  
    **kwargs  
)
```

(3) 重要参数说明：

- optimizer：字符串（优化器的名称）或优化器实例。
- loss：目标函数，或称损失函数，是网络中的性能函数，也是编译一个模型必须的两个参数之一。
- metrics：列表，包含评估模型在训练和测试时的性能的指标，典型用法是 metrics=['accuracy']

3. fit()

(1) 作用：

为固定数量的阶段（数据集上的迭代）训练模型。

(2) 函数原型：

```
fit(  
    x=None,  
    y=None,  
    batch_size=None,  
    epochs=1,  
    verbose=1,  
    callbacks=None,  
    validation_split=0.0,  
    validation_data=None,  
    shuffle=True,  
    class_weight=None,  
    sample_weight=None,  
    initial_epoch=0,  
    steps_per_epoch=None,  
    validation_steps=None,  
    validation_freq=1,  
    max_queue_size=10,  
    workers=1,  
    use_multiprocessing=False,  
    **kwargs  
)
```

(3) 重要参数说明:

- x: 输入数据。
- y: 目标数据
- epochs: 数据训练次数

4. evaluate()

(1) 作用:

返回测试模式下模型的损失值和度量值。

(2) 函数原型:

```
evaluate(  
    x=None,  
    y=None,  
    batch_size=None,  
    verbose=1,  
    sample_weight=None,  
    steps=None,  
    callbacks=None,  
    max_queue_size=10,  
    workers=1,  
    use_multiprocessing=False  
)
```

(3) 重要参数说明:

- x: 输入数据。
- y: 目标数据

5. save_model()

(1) 作用:

将模型保存为TensorFlow SavedModel 或 HDF5文件。

(2) 函数原型

```
tf.keras.models.save_model(  
    model,  
    filepath,  
    overwrite=True,  
    include_optimizer=True,  
    save_format=None,  
    signatures=None,  
    options=None  
)
```

(3) 重要参数说明:

- model: 要保存的 Keras 模型实例。
- filepath: 模型保存路径

6. load_model()

(1) 作用:

加载通过 save_model 保存的模型。

(2) 函数原型:

```
tf.keras.models.load_model(  
    filepath,  
    custom_objects=None,  
    compile=True  
)
```

(3) 重要参数说明:

- filepath: save_model 保存路径

7. predict()

(1) 作用:

为输入样本生成输出预测。

(2) 函数模型

```
predict(  
    x,  
    batch_size=None,  
    verbose=0,  
    steps=None,  
    callbacks=None,  
    max_queue_size=10,  
    workers=1,  
    use_multiprocessing=False  
)
```

(3) 重要参数说明：

- x: 输入样本

(4) 返回值：

预测的 Numpy array(s)

这些 API 都是 tensorflow 2.0 中常见的 API，在使用 tensorflow 2.0 时，我们几乎都会用到这些 API。能熟练掌握这些 API 是非常重要的一件事情。下面，我们就通过一个实例来演示这些 API 的运用。

三. 实例演示

- 本节将对一个图片分类的程序进行讲解，目的是让读者了解如何初步使用 Tensorflow 2.0。
- 本节将分两部分进行讲解：
 - 第一部分：主要讲述如何载入数据，构建模型，编译模型，训练模型，获取模型训练数据以及保存模型。
 - 第二部分：主要讲解如何载入保存的模型以及使用载入的模型进行预测。

1. 第一部分

(1). 载入数据

```
(train_images, train_labels), (test_images, test_labels) = keras.datasets.mnist.load_data()
```

本程序载入的数据是 MNIST 数据库。



MNIST 数据库是由 Yann 提供的手写数字数据库文件。数据库里的图像都是 28*28 大小的灰度图像，每个像素的是一个八位字节 (0-255)。这个数据库主要包含了 60000 张的训练图像和 10000 张的测试图像。

(2) 构建模型

```
model = keras.Sequential()
model.add(keras.layers.Flatten(input_shape=(28, 28)))
model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dense(10, activation='softmax'))
```

本模型包括三层：



第一层：输入层

- Flatten 层。将输入的二维矩阵图片数据平坦化，转化为一维矩阵。

第二层：隐含层

- Dense 层即全连接层，逻辑上等价于这样一个函数：

$$out = Activation(Wx + bias)$$

- 权重W为m*n的矩阵。
- 输入x为n维向量。
- 激活函数Activation。
- 偏置bias。
- 输出向量out为m维向量。
- 使用 'relu' 作为激活函数。

第三层：输出层

- Dense 层。使用 'softmax' 作为激活函数。
- softmax 激活函数一般用在解决多分类问题网络的最后一层，用于生成对应每个类别的概率分布，这些类别的概率分布之和为 1。设当前的任务为一个 n 分类任务，对应于第 i 个类别的 softmax 值为：

$$S_i = \frac{e^i}{\sum_{j=1}^n e^j}$$

模型构建完成之后可以使用 summary() 查看模型



从上面可以看到模型层次，参数数量等。

(3) 模型编译

- 通过调用 compile() 方法配置该模型的学习流程。

```
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=['acc'])
```

(4) 模型训练

- 使用 `fit()` 函数进行模型训练，训练数据为 mnist 数据集的 `train_images` 和 `train_labels`，总共训练 50 轮。

```
model.fit(train_images, train_labels, epochs=50)
```

(5) 训练效果

- 使用 `evaluate()` 函数可以查看训练模型的 `loss` 值以及准确率。

```
loss, acc = model.evaluate(test_images, test_labels)
print("The acc is ", acc*100, "%")
```

(6) 保存模型

```
keras.models.save_model(model, "save_model")
```

(7) 完整代码 (test_code_part1.py)

- `test_code_part1.py` 在本项目文件的 `src` 文件夹下

```

import tensorflow as tf
from tensorflow import keras
import numpy as np

# 打印 Tensorflow 版本以及是否支持 GPU 加速
print("Tensorflow version : ",tf.__version__)
print(tf.test.is_gpu_available())

if __name__ == "__main__":
    # 导入数据
    (train_images, train_labels), (test_images, test_labels) = keras.datasets.mnist.load_data()

    train_images = train_images / 255      # 图片数据归一化

    # 构建模型
    model = keras.Sequential()
    model.add(keras.layers.Flatten(input_shape=(28, 28)))
    model.add(keras.layers.Dense(128, activation='relu'))
    model.add(keras.layers.Dense(10, activation='softmax'))

    # 查看构建模型
    model.summary()

    # 模型编译
    model.compile(optimizer="adam",
                  loss="sparse_categorical_crossentropy",
                  metrics=['acc'])

    # 模型训练
    model.fit(train_images, train_labels, epochs=50)

    # 得到模型准确率
    loss, acc = model.evaluate(test_images, test_labels)
    print("The acc is ",acc*100, "%")

    # 保存模型
    keras.models.save_model(model, "save_model")

```

2. 第二部分

(1) 读取图片数据并进行处理

- 使用 opencv 对图片数据进行读取，并对图片数据进行归一化处理。

```

image_6 = cv2.imread("test_image/image6.bmp", cv2.IMREAD_GRAYSCALE)

image_6 = image_6 / 255

```

(2) 载入模型


```
mnist_model = keras.models.load_model("./save_model")
```

(3) 使用载入的模型进行预测

- 使用 predict() 函数进行预测，并输出预测结果

```
a = mnist_model.predict(image_6.reshape(1, 28, 28))  
  
print(a)  
print(np.argmax(a, axis=1))
```

(4) 完整代码 (test_code_part2.py)

- test_code_part2.py 在本项目文件的 src 文件夹下

```
import tensorflow as tf  
from tensorflow import keras  
  
import numpy as np  
import cv2  
  
# 读取图片数据  
image_6 = cv2.imread("test_image/image6.bmp", cv2.IMREAD_GRAYSCALE)  
  
# 将图片数据进行归一化处理  
image_6 = image_6 / 255  
  
# 导出模型  
mnist_model = keras.models.load_model("./save_model")  
  
# 使用导出模型，进行测试  
a = mnist_model.predict(image_6.reshape(1, 28, 28))  
  
# 打印预测结果  
print(a)  
print(np.argmax(a, axis=1))
```

本节通过一个简单的图片分类程序，讲解了 tensorflow2.0 的基本使用。在tensorflow 中基本的应用程序框架为：载入数据，构建模型，编译模型，模型训练，模型评估，模型保存。

四. 推荐及参考资料：

- (1) [TensorFlow 官网](#)
- (2) [知乎 Tensorflow2教程](#)
- (3) [TensorFlow 2.0 高效开发指南](#)