

现有网络模型的调用和修改

2025年11月27日 14:57

vgg16 VG16

```
torchvision.models.vgg16(*, weights: Optional[VGG16_Weights], **kwargs: Any) → VGG [SOURCE]
torchvision.models.vgg16(*, 权重: 可选[VGG16_Weights], **kwargs: Any) → VGG [来源]
```

VGG-16 from [Very Deep Convolutional Network](#)
VGG-16 来自用于大规模图像识别的超深卷积神经网络

Parameters:
参数:

• **weights** (VGG16_Weights, optional)

details, and possible values. By default, no pre-trained weights are used.

权重 (VGG16_Weights, 可选) – 要使用的预训练权重。有关更多详细信息和可能的值, 请参阅下面的 [VGG16_Weights](#)。默认情况下, 不使用预先训练的权重。

• **progress** (bool, optional) – If True, displays a progress bar of the download to stderr. Default is True.

Progress (bool, 可选) – 如果为 True, 则显示下载到 stderr 的进度条。默认为 True。

• ****kwargs** – parameters passed to the `torchvision.models.vgg.VGG` base class. Please refer to the [source code](#) for more details about this class.

****kwargs** – 传递给 `torchvision.models.vgg.VGG` 基类的参数。有关该类的更多信息, 请参阅源代码。

`CLASS`vision.models.VGG16_Weights (value)

类 `torchvision.models.VGG16_Weights(值)`

[\[SOURCE\]](#) [\[来源\]](#)

The model builder above accepts the following values as the weights parameter. `VGG16_Weights.DEFAULT` is equivalent to `VGG16_Weights.IMAGENET1K_V1`. You can also use strings, e.g. `weights='DEFAULT'` or `weights='IMAGENET1K_V1'`.

上面的模型构建器接受以下值作为权重参数。 `VGG16_Weights.DEFAULT` 相当于

2. `weights: Optional[VGG16_Weights] = None`

这是最关键的一个参数, 它决定了你的模型是从头开始学习, 还是 **基于已有的知识**。

• `Optional` : 表示这个参数是可选的。

• `VGG16_Weights` : 一个专门的数据类型, 用于指定要加载哪种权重。

• `= None` (默认值):

• 通俗解释: 如果你忽略这个参数或设为 `None`, 模型参数 (权重) 会随机初始化。

• 后果: 你的模型必须 **从零开始** 在你的数据集上学习, 训练时间会非常长。

• 如果你想加载预训练权重 (Pretrained Weights):

• 你可以传入一个特定的权重枚举值 (比如 `VGG16_Weights.IMAGENET1K_V1`)。

• 前瞻性意义: 这是深度学习中打破常规思维的利器——**迁移学习 (Transfer Learning)**。它允许你加载 VGG-16 在巨大的 ImageNet 数据集上学习到的通用特征 (比如边缘、纹理等), 然后你只需要在你的小数据集上微调, 可以 **大大加快训练速度和提升性能**。

```
Existing network models x
D:\anaconda3\envs\pytorch\python.exe "D:\pycharm_space\PycharmProjects\PythonProject1\nurse_net\Existing network models.py"
Downloading: "https://download.pytorch.org/models/vgg16-397923af.pth" to C:\Users\Jay\.cache\torch\hub\checkpoints\vgg16-397923af.pth
44%|██████████| 230M/528M [01:02<01:23, 3.74MB/s]
```

看最终输出

```
(4): ReLU(inplace=True)
(5): Dropout(p=0.5, inplace=False)
(6): Linear(in_features=4096, out_features=1000, bias=True)
)
```

1000个类, 但实际情况应该都不会与现有的网络模型相符
所以, 要对其进行一个修改

添加和修改

```
loss.py
max_pool.py
nn_model.py

dataset
model

8 model = torchvision.models.vgg16(weights=VG16_Weights.DEFAULT)
9
10 #CIFAR10 是10个输出类，但vgg16足足有1000个，所以要对vgg16进行修改
11
12 #添加
13 # model.add_module('add_linear',nn.Linear(1000, 10))
14 #在指定层添加
15 # model.classifier.add_module('add_linear',nn.Linear(in_features=4096, out_features=10))
16 #修改指定内容
17 model.classifier[6]=nn.Linear(in_features=4096, out_features=10)
18 print(model)
19
```

Existing network models x

```
(4): ReLU(inplace=True)
(5): Dropout(p=0.5, inplace=False)
(6): Linear(in_features=4096, out_features=10, bias=True)
)
```

Model 模型

Module 模块

```
torch.save(model, state_dict(), pickle_module: 'vgg16_1.pth')
```

不是，是.

```
import torch
import torchvision
from torch.distributed.checkpoint import state_dict
from torchvision.models import VGG16_Weights

model = torchvision.models.vgg16(weights=VGG16_Weights.DEFAULT)

#保存方式1  框架+参数
torch.save(model, f'vgg16_0.pth')
#加载
model1 = torch.load('vgg16_0.pth')
print(model1)

#保存方式2  只保存参数（权重）
torch.save(model.state_dict(), f'vgg16_1.pth')
#加载
model2 = torchvision.models.vgg16(pretrained=False)
model2.load_state_dict(torch.load('vgg16_1.pth'))
print(model2)
```

先搞个框架出来，在把参数加载进去