

# 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]) → 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。有关该类的更多详细信息, 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

```
行 Existing network models ×
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

The screenshot shows a PyCharm code editor with a file named `model.py`. The code imports `torchvision.models.vgg16` and creates a `model` object. A comment indicates that the original `vgg16` has 1000 output classes, while CIFAR10 has 10, so the classifier needs to be modified. A blue box highlights the following code additions:

```
model = torchvision.models.vgg16(weights=VGG16_Weights.DEFAULT)
#CIFAR10 是10个输出类, 但vgg16足足有1000个, 所以要对vgg16进行修改
#添加
# model.add_module('add_linear',nn.Linear(1000, 10))
#在指定层添加
# model.classifier.add_module('add_linear',nn.Linear(in_features=4096, out_features=10))
#修改指定内容
model.classifier[6]=nn.Linear(in_features=4096, out_features=10)
print(model)
```

Below the code, a dropdown menu titled "Existing network models" lists three options:

- (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)          先搞个框架出来，在把参数加载进去
model2.load_state_dict(torch.load('vgg16_1.pth'))
print(model2)
```