个人工作

ZHENGYUNLONG

学习Pytorch中hook函数并分享

1.hook简介

Pytorch中自带hook函数用于提取非叶子节点的梯度，进一步，可利用hook函数提取特征图。

Pytorch中有四类hook函数：

1. torch.Tensor.register_hook
2. torch.nn.Module.register_forward_pre_hook
3. torch.nn.Module.register_forward_hook
4. torch.nn.Module.register_backward_hook

2.torch.Tensor.register_hook()

Tensor类下的hook函数，当注册hook函数后，再次计算某个tensor的梯度会执行注册的hook函数。

3.torch.nn.Module.register_forward_pre_hook()

Module类下的hook函数，当注册hook函数后，模型执行forward前自动调用注册的hook函数。

4.torch.nn.Module.register_forward_hook()

Module类下的hook函数，当注册hook函数后，模型执行forward时自动调用注册的hook函数。

具体执行过程：网络模型被调用执行时，先执行Moule.call函数，该函数内会首先判定当前是否注册了hook函数，若注册，则调用hook函数。基于此，我们可以编写特定hook函数来可视化非叶子节点的梯度。

5.torch.nn.Module.register_backward_hook()

Module类下的hook函数，反向传播时调用，可依此提取特征图梯度。

6.代码示例

MNIST数据集上数字识别，模型结构：俩个卷积层+三个全连接层。

```
from pathlib import Path
import pickle
import gzip
import torch
import struct
import numpy as np
import matplotlib.pyplot as plt
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import TensorDataset, DataLoader

# 加载MNIST数据集
```

```
train_img_path = Path("D:\\Daily\\training\\code\\Dataset\\MNIST\\train-images.idx3-ubyte")
train_label_path = Path("D:\\Daily\\training\\code\\Dataset\\MNIST\\train-labels.idx1-ubyte")
test_img_img = Path("D:\\Daily\\training\\code\\Dataset\\MNIST\\t10k-images.idx3-ubyte")
test_label_path = Path("D:\\Daily\\training\\code\\Dataset\\MNIST\\t10k-labels.idx1-ubyte")

with open(train_img_path, "rb") as f1:
    images_magic, images_num, rows, cols = struct.unpack('>IIII', f1.read(16))
    x_train = np.fromfile(f1, dtype=np.uint8).reshape(images_num, rows * cols)
with open(train_label_path, "rb") as f2:
    labels_magic, labels_num = struct.unpack('>II', f2.read(8))
    y_train = np.fromfile(f2, dtype=np.uint8)
with open(test_img_img, "rb") as f3:
    test_images_magic, test_images_num, test_rows, test_cols =
struct.unpack('>IIII', f3.read(16))
    x_test = np.fromfile(f3, dtype=np.uint8).reshape(test_images_num, test_rows
* test_cols)
with open(test_label_path, "rb") as f4:
    test_labels_magic, test_labels_num = struct.unpack('>II', f4.read(8))
    y_test = np.fromfile(f4, dtype=np.uint8)

# img = x_train[1].reshape(28, 28)
# plt.imshow(img)
# plt.title("the label is : {}".format(y_train[1]))
# plt.show()
#
# print(x_train[0])

print(torch.cuda.is_available())
dev = torch.device("cuda:0")

# 转换成tensor
x_train = torch.tensor(x_train).float().to(dev)
y_train = torch.tensor(y_train).float().to(dev)
x_test = torch.tensor(x_test).float().to(dev)
y_test = torch.tensor(y_test).float().to(dev)

bs = 64

train_ds = TensorDataset(x_train, y_train)
train_dl = DataLoader(dataset=train_ds, batch_size=bs, shuffle=True)

test_ds = TensorDataset(x_test, y_test)
test_dl = DataLoader(dataset=test_ds, batch_size=bs * 2)

# 定义模型


class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=1, out_channels=6, kernel_size=3)
```

```
        self.conv2 = nn.Conv2d(in_channels=6, out_channels=16, kernel_size=4)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = x.view(-1, 1, 28, 28)
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = x.view(-1, self.num_flat_features(x))
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

    def num_flat_features(self, x):
        size = x.size()[1:]
        num_features = 1
        for s in size:
            num_features *= s
        return num_features


cnn_model = Net()
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(cnn_model.parameters(), lr=0.05)
cnn_model.to(dev)
# 训练模型
# epochs = 10
#
# for epoch in range(epochs):
#     for xb, yb in train_dl:
#         pred = cnn_model(xb)
#         loss = criterion(pred, yb.long())
#
#         optimizer.zero_grad()
#         loss.backward()
#         optimizer.step()


def loss_mean(model, valid_dl):
    valid_loss = 0
    for xb, yb in valid_dl:
        valid_loss += criterion(model(xb), yb.long())

    return valid_loss / len(valid_dl)


def accuracy(out, yb):
    preds = torch.argmax(out, dim=1)
    return (preds == yb).float().mean()


def accuracy_rate(model, valid_dl):
    acc = 0
    for xb, yb in valid_dl:
        acc += accuracy(model(xb), yb.long())
```

```python
    return (acc / len(valid_dl))


def hook(module, grad_input, grad_output):
    print("grad_input: ", grad_input)
    # plt.imshow(grad_input.reshape(28, 28))
    # plt.title("the label is : {}".format("grad_input"))
    # plt.show()
    print("grad_output: ", grad_output)
    # plt.imshow(grad_output.reshape(-1, 28, 28))
    # plt.title("the label is : {}".format("grad_output"))
    # plt.show()
    # return grad_input[0]


# with torch.no_grad():
#     print('验证集loss: %.4f' % loss_mean(cnn_model, train_dl).item())
#     print('准确率acc: %.2f%%' % (accuracy_rate(cnn_model, test_dl).item() *
100))

for para in cnn_model.named_parameters():
    print(para[0])

x = x_test[0]
handle = cnn_model.conv1.register_forward_hook(hook)
y = cnn_model(x)
handle.remove()
```

该代码目前能可视化模型在前向传播时非叶子节点的梯度，但由于尺寸问题，暂时未将其显示成图片，以及Grad_cam热力图未绘制，这将是下一阶段工作的主要内容。

XIEMENGYING

- 在图片分类网络任务上验证深度可分卷积效果

数据集

- tfds horses_or_humans
- 大小300*300, training set 1027张图片两二元分类，validation set 256张

网络结构

5层卷积+1全连接

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d_66 (MaxPooling (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 147, 147, 64)      9280
_________________________________________________________________
max_pooling2d_67 (MaxPooling (None, 73, 73, 64)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 71, 71, 64)        36928
_________________________________________________________________
max_pooling2d_68 (MaxPooling (None, 35, 35, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 33, 33, 128)       73856
_________________________________________________________________
max_pooling2d_69 (MaxPooling (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 14, 14, 128)       147584
_________________________________________________________________
max_pooling2d_70 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_12 (Flatten)         (None, 6272)              0
_________________________________________________________________
dense_24 (Dense)             (None, 32)                200736
_________________________________________________________________
dropout_11 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_25 (Dense)             (None, 1)                 33
=================================================================
Total params: 468,865
```

结果

Epoch 1/10 - 5s 578ms/step - loss: 0.6816 - accuracy: 0.5787 - val_loss: 0.6797 - val_accuracy: 0.5000 Epoch 2/10 - 4s 501ms/step - loss: 0.6585 - accuracy: 0.6208 - val_loss: 0.6452 - val_accuracy: 0.5742 Epoch 3/10 - 4s 552ms/step - loss: 0.6117 - accuracy: 0.7574 - val_loss: 0.6545 - val_accuracy: 0.5352 Epoch 4/10 - 4s 547ms/step - loss: 0.5149 - accuracy: 0.8047 - val_loss: 0.6068 - val_accuracy: 0.7031 Epoch 5/10 - 4s 507ms/step - loss: 0.4141 - accuracy: 0.8431 - val_loss: 0.4577 - val_accuracy: 0.8320

将Conv2D替换成SeparableConv2D后没学出来，大概替换两层以上就不行了