

# MobileNet-CNN

---

ここでは、torchvisionを利用してMobileNet-CNNをファインチューニングし、推論を実行します。

## データセットの定義

torchvisionには画像分類用のデータセットが多数実装されています。今回は、MNISTを利用します。データセットのリファレンスは以下にあります。

<https://pytorch.org/docs/stable/torchvision/datasets.html>

リファレンスに従ってデータセットを生成します。今回はCIFAR10データセットを使用しました。

```
import torch
import torchvision

# 学習データをダウンロード
transform = torchvision.transforms.Compose([
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=4, shuffle=True,
num_workers=2)

# ラベル一覧取得
classes = trainset.classes
num_classes = len(classes)
print(num_classes)
```

## ネットワーク定義

torchvisionには画像分類用のネットワークが多数実装されています。今回はMobileNetV2を利用します。モデルのリファレンスは以下にあります。

<https://pytorch.org/docs/stable/torchvision/models.html#classification>

リファレンスに従い、以下のようにmodelを生成します。今回はImageNetによる事前学習済みのモデルをダウンロードします。学習データとしてCIFAR10を使用するので、出力属性数は10としました。

```
model = torchvision.models.mobilenet_v2(pretrained=True, progress=True,
num_classes=10)
```

## 学習

学習スクリプトを記述します。

```
# 学習パラメータ定義
criterion = nn.CrossEntropyLoss() # クロスエントロピー
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9) # 確率的勾配降下法
num_epoch = 1
out_model_path = './mobilenet.pth'

# 学習
for epoch in range(num_epoch):
    running_loss = 0.0
    for i, data in enumerate(trainloader, 1):
        inputs, labels = data
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        if not i % 10:
            print('epoch: {}/{}'.format(epoch + 1, num_epoch), i, len(trainloader), running_loss / 2000)
            running_loss = 0.0

# 学習済みモデル出力
torch.save(model.state_dict(), out_model_path)
print('Finished Training: {}'.format(out_model_path))
```

## 推論

推論スクリプトを記述します。

```
import torch
import torch.nn as nn
import torchvision

# テストデータをダウンロード
transform = torchvision.transforms.Compose([
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=4, shuffle=True, num_workers=2)

# ラベル一覧取得
classes = testset.classes
num_classes = len(classes)
```

```
print('num_classes: {}'.format(num_classes))
print('classes: {}'.format(classes))

# モデル生成
model = torchvision.models.mobilenet_v2(pretrained=True, progress=True)
model.classifier[1] = nn.Linear(model.classifier[1].in_features, num_classes)

# 読み込み
model_path = './mobilenet.pth'
model.load_state_dict(torch.load(model_path))

# 推論
dataiter = iter(testloader)
images, labels = dataiter.next()
outputs = model(images)
_, predicted = torch.max(outputs, 1)
print('ground_truth: {}'.format([classes[labels[i]] for i in range(4)]))
print('detected_label: {}'.format([classes[predicted[i]] for i in range(4)]))

# ground_truth: ['frog', 'airplane', 'horse', 'deer']
# detected_label: ['frog', 'airplane', 'horse', 'deer']
```

スクリプト全体は以下にあります。

[train.py](#)

[test.py](#)

[目次へ戻る](#)