TransferLearningResnet18ImageNet_Cifar10_LastLayer

February 23, 2025

1 Cài đặt thư viện

[ ]:
```
!pip install lightning
```

2 shared_utilities.py

[ ]:
```
import lightning as L
import matplotlib.pyplot as plt
import pandas as pd
import torch
import torch.nn.functional as F
import torchmetrics
from torch.utils.data import DataLoader
from torch.utils.data.dataset import random_split
from torchvision import datasets, transforms


class LightningModel(L.LightningModule):
    def __init__(self, model, learning_rate):
        super().__init__()

        self.learning_rate = learning_rate
        self.model = model

        self.save_hyperparameters(ignore=["model"])

        self.train_acc = torchmetrics.Accuracy(task="multiclass",␣
 ↪num_classes=10)
        self.val_acc = torchmetrics.Accuracy(task="multiclass", num_classes=10)
        self.test_acc = torchmetrics.Accuracy(task="multiclass", num_classes=10)

    def forward(self, x):
        return self.model(x)

    def _shared_step(self, batch):
        features, true_labels = batch
```

```
        logits = self(features)

        loss = F.cross_entropy(logits, true_labels)
        predicted_labels = torch.argmax(logits, dim=1)
        return loss, true_labels, predicted_labels

    def training_step(self, batch, batch_idx):
        loss, true_labels, predicted_labels = self._shared_step(batch)

        self.log("train_loss", loss)
        self.train_acc(predicted_labels, true_labels)
        self.log(
            "train_acc", self.train_acc, prog_bar=True, on_epoch=True,␣
 ↪on_step=False
        )
        return loss

    def validation_step(self, batch, batch_idx):
        loss, true_labels, predicted_labels = self._shared_step(batch)

        self.log("val_loss", loss, prog_bar=True)
        self.val_acc(predicted_labels, true_labels)
        self.log("val_acc", self.val_acc, prog_bar=True)

    def test_step(self, batch, batch_idx):
        loss, true_labels, predicted_labels = self._shared_step(batch)
        self.test_acc(predicted_labels, true_labels)
        self.log("test_acc", self.test_acc)

    def configure_optimizers(self):
        optimizer = torch.optim.SGD(self.parameters(), lr=self.learning_rate)
        return optimizer

class Cifar10DataModule(L.LightningDataModule):
    def __init__(
        self,
        data_path="./",
        batch_size=64,
        height_width=None,
        num_workers=0,
        train_transform=None,
        test_transform=None,
    ):
        super().__init__()
        self.batch_size = batch_size
        self.data_path = data_path
        self.num_workers = num_workers
```

```
        self.train_transform = train_transform
        self.test_transform = test_transform
        self.height_width = height_width

    def prepare_data(self):
        datasets.CIFAR10(root=self.data_path, download=True)

        if self.height_width is None:
            self.height_width = (32, 32)

        if self.train_transform is None:
            self.train_transform = transforms.Compose(
                [
                    transforms.Resize(self.height_width),
                    transforms.ToTensor(),
                ]
            )

        if self.test_transform is None:
            self.test_transform = transforms.Compose(
                [
                    transforms.Resize(self.height_width),
                    transforms.ToTensor(),
                ]
            )

        return

    def setup(self, stage=None):
        train = datasets.CIFAR10(
            root=self.data_path,
            train=True,
            transform=self.train_transform,
            download=False,
        )

        self.test = datasets.CIFAR10(
            root=self.data_path,
            train=False,
            transform=self.test_transform,
            download=False,
        )

        self.train, self.valid = random_split(train, lengths=[45000, 5000])

    def train_dataloader(self):
        train_loader = DataLoader(
```

```
            dataset=self.train,
            batch_size=self.batch_size,
            drop_last=True,
            shuffle=True,
            num_workers=self.num_workers,
        )
        return train_loader

    def val_dataloader(self):
        valid_loader = DataLoader(
            dataset=self.valid,
            batch_size=self.batch_size,
            drop_last=False,
            shuffle=False,
            num_workers=self.num_workers,
        )
        return valid_loader

    def test_dataloader(self):
        test_loader = DataLoader(
            dataset=self.test,
            batch_size=self.batch_size,
            drop_last=False,
            shuffle=False,
            num_workers=self.num_workers,
        )
        return test_loader


def plot_loss_and_acc(
    log_dir, loss_ylim=(0.0, 0.9), acc_ylim=(0.7, 1.0), save_loss=None,␣
 ↪save_acc=None
):

    metrics = pd.read_csv(f"{log_dir}/metrics.csv")

    aggreg_metrics = []
    agg_col = "epoch"
    for i, dfg in metrics.groupby(agg_col):
        agg = dict(dfg.mean())
        agg[agg_col] = i
        aggreg_metrics.append(agg)

    df_metrics = pd.DataFrame(aggreg_metrics)
    df_metrics[["train_loss", "val_loss"]].plot(
        grid=True, legend=True, xlabel="Epoch", ylabel="Loss"
    )
```

```
    plt.ylim(loss_ylim)
    if save_loss is not None:
        plt.savefig(save_loss)

    df_metrics[["train_acc", "val_acc"]].plot(
        grid=True, legend=True, xlabel="Epoch", ylabel="ACC"
    )

    plt.ylim(acc_ylim)
    if save_acc is not None:
        plt.savefig(save_acc)
```

3 Import thư viện

[ ]:
```
import torch
import torchvision
import torch.nn as nn
import torch.nn.functional as F
import torchmetrics
from lightning.pytorch.loggers import CSVLogger
import matplotlib.pyplot as plt
import numpy as np
```

4 Tải mô hình/trọng số Resnet18 từ PyTorch Hub.

[ ]:
```
entrypoints = torch.hub.list('pytorch/vision:v0.13.0', force_reload=True)
for e in entrypoints:
    if "resnet" in e:
        print(e)
pytorch_model = torch.hub.load('pytorch/vision:v0.13.0', 'resnet18',␣
 ↪weights='IMAGENET1K_V1')
```

[ ]:
```
pytorch_model
```

5 Fine-tune lớp cuối cùng và đóng băng các lớp trước

[ ]:
```
for param in pytorch_model.parameters():
    param.requires_grad = False

pytorch_model.fc = torch.nn.Linear(512, 10)
```

6 Sử dụng lại đúng các `transform` đã được sử dụng để huấn luyện mô hình ResNet18 trên tập ImageNet

[ ]:
```
from torchvision.models import resnet18, ResNet18_Weights

weights = ResNet18_Weights.IMAGENET1K_V1
preprocess_transform = weights.transforms()
preprocess_transform
```

7 Khởi tạo trainer core

[ ]:
```
%%capture --no-display

L.pytorch.seed_everything(123)

dm = Cifar10DataModule(batch_size=64, num_workers=4,
                       train_transform=preprocess_transform,
                       test_transform=preprocess_transform)

lightning_model = LightningModel(model=pytorch_model, learning_rate=0.1)

trainer = L.Trainer(
    max_epochs=50,
    accelerator="auto",
    devices="auto",
    logger=CSVLogger(save_dir="logs/", name="TransferLearning_LastLayer-model"),
    deterministic=True,
)
```

8 Huấn luyện mô hình

[ ]:
```
trainer.fit(model=lightning_model, datamodule=dm)
```

9 Vẽ loss và accuracy

[ ]:
```
plot_loss_and_acc(trainer.logger.log_dir, loss_ylim=(0.0, 2.0))
```

10 Kết qủa dự đoán trên tập test

[ ]:
```
trainer.test(model=lightning_model, datamodule=dm)
```