michaelhipes

pytorch transfer learning

1: get data

1: get data (mine.py)

2: create datasets and dataloaders

```
train_dataloader, test_dataloader, class_names = data_setup.create_dataloader(...)
```

3: getting a pretrained model

```
weights = torchvision.models.EfficientNet_B0_Weights.DEFAULT
model = torchvision.models.efficientnet_b0(weights=weights).to(device)
```

import model

```
for param in model.features.parameters():
    param.requires_grad = False
```

freeze weights

```
torch.manual_seed(42) ... torch.cuda.manual_seed(42)
output_shape = len(class_names)
model.classifier = torch.nn.Sequential(torch.nn.Dropout(p=0.2, inplace=True), torch.nn.Linear(in_features=1280,
                   out_features=output_shape, bias=True)).to(device)
```

rewrite classifier layer

torchinfo

```
summary()
```

```
Sequential(features)
adaptiveavgpool2d(avgpool)
Sequential(classifier)
```

4: train model

```
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
torch.manual_seed(42) ... torch.cuda.manual_seed(42)
results = engine.train(...)
```

5: plot loss curves

6: make predictions on random image

11: make predictions on random images