machinelearning

Pytorch going modular — source from custom datasets

0: create a folder for storing Python scripts

```
import os
os.makedirs("going_modular", exist_ok=True)
```

1: get data

1: get data

2: create Datasets and DataLoaders

```
%%writefile going_modular/data_setup.py
def create_dataloaders(train_dir:str, test_dir:str, transform: transforms.Compose, batch_size:int, num_workers: int=NUM_WORKERS):
    4: loading image data using image folder
    return train_dataloader, test_dataloader, class_names
```

3: making a model (TinyVGG)

```
%%writefile going_modular/model_builder.py
pytorch computervision FashionMNISTModelV2
```

4: create trainstep and testStep in train()

```
%%writefile going_modular/engine.py
```

7: model0 (minus) — (just the 3 functions (middle))

6: train, evaluate and save model

```
%%writefile going_modular/train.py
import os ... import torch ... from torchvision import transforms ... import data_setup, engine, model_builder, utils
num_epochs=5 ... batch_size=32 ... hidden_units=10 ... learning_rate=0.001
train_dir="data/pizza_steak_sushi/train"
test_dir=" " " " test
device="cuda" if torch.cuda.is_available() else "cpu"
data_transform=transforms.Compose([transforms.Resize((64,64)), transforms.ToTensor()])
train_dataloader, test_dataloader, class_names=data_setup.create_dataloaders(train_dir=train_dir, test_dir=test_dir,
    transform=data_transform, batch_size=BATCH_SIZE)
```

Goes to create from data_setup.py

final directory:

```
data/
  pizza_steak_sushi/
    train/
      pizza/
        train_image_01.jpeg
        ...
      steak/
      sushi/
    test/
      ...
going_modular/ — train.py ... utils.py
  data_setup.py ... engine.py ... model_builder.py
models/ — saved_model_path
```

num_workers:

help from model_builder.py

```
model = model_builder.TinyVGG(input_shape=3, hidden_units=hidden_units, output_shape=len(class_names)).to(device)

loss_fn = torch.nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

engine.train(model=model, train_dataloader=train_dataloader, test_   , test_   , loss_fn=   , optimizer=   ,
             epochs=num_epochs, device=   )

utils.save_model(model=model, target_dir="models", model_name="05_going_modular_script_mode_tinyvgg_model.pth")
```

5: creating a function to save the model

```
%%writefile going_modular/utils.py

def save_model(model: torch.nn.Module, model_name: str):
    target_dir_path = Path(target_dir)                                  ] create target directory
    target_dir_path.mkdir(parents=True, exist_ok=True)

    assert model_name.endswith(".pth") or model_name.endswith(".pt"), "model_name should end w/ .pt or .pth"   ] create model save path
    model_save_path = target_dir_path / model_name
    torch.save(obj=model.state_dict(), f=model_save_path)   ] save model state_dict()
```

```
! python going_modular/train.py
```

Works :)