pytorch model deployment

0: getting setup

0: getting setup - pytorch paper replicating

10: creating foodvision Bia

```
def create_effnetb2_model(num_classes:int=3, seed:int=42):
    weights = torchvision.models.EfficientNet_B2_Weights.DEFAULT
    transforms = weights.transforms()
    model = torchvision.models.efficientnet_b2(weights=weights)
    for param in model.parameters(): param.requires_grad = False
    torch.manual_seed(seed)
    model.classifier = nn.Sequential(nn.Dropout(p=0.3, inplace=True), nn.Linear(in_features=1408, out_features=
    return model, transforms

effnetb2_food101, effnetb2_transforms = create_effnetb2_model(num_classes=101)
food101_train_transforms = torchvision.transforms.Compose([torchvision.transforms.TrivialAugmentWide(), effnetb2_transforms,])
from torchvision import datasets ... from pathlib import Path
data_dir = Path("data")
train_data = datasets.Food101(root=data_dir, split="train", transform=food101_train_transforms, download=True)
test_data = "" "" "test" "" ""
def split_dataset(dataset:torchvision.datasets, split_size:float=0.2, seed:int=42):
    length_1 = int(len(dataset) * split_size) ... length_2 = len(dataset) - length_1
    random_split_1, random_split_2 = torch.utils.data.random_split(dataset, lengths=[length_1, length_2], generator=torch.manual_seed(s
    return random_split_1, random_split_2
train_data_food101_20_percent, _ = split_dataset(dataset=train_data, split_size=0.2)
test_data_ "" "" "" test "" ""
import os ... import torch ... batch_size=32 ... num_workers=1 if os.cpu_count()<=4 else 4
train_dataloader_food101_20_percent = torch.utils.data.DataLoader(train_data_food101_20_percent, batch_size=batch_size, shuffle=True, num_workers=num_
test "" "" "" test "" "" False "
from going_modular.going_modular import engine
optimizer = torch.optim.Adam(params=effnetb2_food101.parameters(), lr=1e-3)
loss_fn = torch.nn.CrossEntropyLoss(label_smoothing=0.1)
effnetb2_food101_results = engine.train(model=effnetb2_food101, train_dataloader=train_dataloader_food101_20_perc
    test_dataloader=test_dataloader_food101_20_percent, loss_fn=loss_fn, epochs=5, device=device)
```

```
from going_modular.going_modular import utils

effnetb2_food101_model_path = "09_pretrained_effnetb2_feature_extractor_food101_20_percent.pth"

utils.save_model(model=effnetb2_food101, target_dir="models", model_name=effnetb2_food101_model_path)
```

③

```
loaded_effnetb2_food101, effnetb2_transforms = create_effnetb2_model(num_classes=101)

loaded_effnetb2_food101.load_state_dict(torch.load("models/09_pretrained_effnetb2_feature_extractor_food101_20_percent.pth"))
```

```
foodvision_big_demo_path = Path("demos/foodvision_big/")

foodvision_big_demo_path.mkdir(parents=True, exist_ok=True)

(foodvision_big_demo_path / "examples").mkdir(parents=True, exist_ok=True)

food101_class_names = train_data.classes

foodvision_big_class_names_path = foodvision_big_demo_path / "class_names.txt"

with open(foodvision_big_class_names_path, "w") as f:
    f.write("\n".join(food101_class_names))
```

②

```
with open(foodvision_big_class_names_path, "r") as f:
    food101_class_names_loaded = [food.strip() for food in f.readlines()]
```

```
%%writefile demos/foodvision_big/model.py

import torch... import torchvision ... from torch import nn
```

①

```
%%writefile demos/foodvision_big/app.py

import gradio as gr ... import os ... import torch ... from model import create_effnetb2_model

from timeit import default_timer as timer ... from typing import Tuple, Dict
```

② ③

```
def predict(img) -> Tuple[Dict, float]:

    start_time = timer()

    img = effnetb2_transforms(img).unsqueeze(0)

    effnetb2.eval()

    with torch.inference_mode():

        pred_probs = torch.softmax(effnetb2(img), dim=1)

    pred_labels_and_probs = {class_names[i]: float(pred_probs[0][i]) for i in range(len(class_names))}

    pred_time = round(timer()-start_time, 5)

    return pred_labels_and_probs, pred_time
```

pytorch model deployment continued...

```
## gradio app

title = "FoodVision Big :)"

description = "EfficientNetB2 feature extractor computer vision model to classify images of food into 101 different classes."

example_list = [["examples/" + example] for example in os.listdir("examples")]

demo = gr.Interface(fn=predict, inputs=gr.Image(type="pil"), outputs=[gr.Label(num_top_classes=5,
                    label="Predictions"), gr.Number(label="Prediction time (s)"), ], examples=example_list,
                    title=title, description=description)

demo.launch()

%%writefile demos/foodvision_big/requirements.txt
torch==1.12.0 ... torchvision==0.13.0 ... gradio==3.1.4

!cd demos/foodvision_big && zip -r ../foodvision_big.zip * -x "*.pyc" "*.ipynb" "*__pycache__*" "*ipynb_checkpoints*"

try:
    from google.colab import files
    files.download("demos/foodvision_big.zip")
except:
    print("Not running in Google Colab, can't use google.colab.files.download()")

from IPython.display import IFrame
IFrame(src="https://hf.space/embed/mrdbourke/foodvision_big/+", width=900, height=750)
```