Convolutional Neural Networks

Project: Write an Algorithm for Landmark Classification

A simple app

In this notebook we build a very simple app that uses our exported model.

> ? Note how we are not importing anything from our source code (we do not use any module from the `src` directory). This is because the exported model, differently from the model weights, is a standalone serialization of our model and therefore it does not need anything else. You can ship that file to anybody, and as long as they can import `torch`, they will be able to use your model. This is very important for releasing pytorch models to production.

Test your app

Go to a search engine for images (like Google Images) and search for images of some of the landmarks, like the Eiffel Tower, the Golden Gate Bridge, Machu Picchu and so on. Save a few examples locally, then upload them to your app to see how your model behaves!

The app will show the top 5 classes that the model think are most relevant for the picture you have uploaded

```
#!pip install -r requirements.txt | grep -v "already satisfied"
!pip install livelossplot torch==1.11.0 torchvision==0.12.0
```

```
Requirement already satisfied: livelossplot in /usr/local/lib/python3.10/dist-packages (0.5.5)
Requirement already satisfied: torch==1.11.0 in /usr/local/lib/python3.10/dist-packages (1.11.0)
Requirement already satisfied: torchvision==0.12.0 in /usr/local/lib/python3.10/dist-packages (0.12.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch==1.11.0) (4.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision==0.12.0) (1.23.5)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from torchvision==0.12.0) (2.31.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision==0.12.0) (
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from livelossplot) (3.7.1)
Requirement already satisfied: bokeh in /usr/local/lib/python3.10/dist-packages (from livelossplot) (3.2.2)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (3.1.2)
Requirement already satisfied: contourpy>=1 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (1.1.0)
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (23.1)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (1.5.3)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (6.0.1)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot) (6.3.2)
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/python3.10/dist-packages (from bokeh->livelossplot)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot) (0.11
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplot) (
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->livelossplo
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->torchvisi
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->torchvision==0.12.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->torchvision==0.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->torchvision==0.
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2>=2.9->bokeh->liveloss
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->bokeh->livelossplo
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->l
```

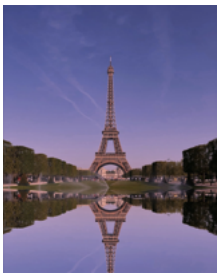
```
from ipywidgets import VBox, Button, FileUpload, Output, Label
from PIL import Image
from IPython.display import display
import io
import numpy as np
import torchvision
import torchvision.transforms as T
import torch

# Decide which model you want to use among the ones exported

jit_model_path = "checkpoints/transfer_exported.pt"
learn_inf = torch.jit.load(jit_model_path)

def on_click_classify(change):

    # Load image that has been uploaded
    fn = io.BytesIO(btn_upload.data[-1])

    img = Image.open(fn)
```

```
    img.load()

    # Let's clear the previous output (if any)
    out_pl.clear_output()

    # Display the image
    with out_pl:

        ratio = img.size[0] / img.size[1]
        c = img.copy()
        c.thumbnail([ratio * 200, 200])
        display(c)

    # Transform to tensor
    timg = T.ToTensor()(img).unsqueeze_(0)

    # Calling the model
    softmax = learn_inf(timg).data.cpu().numpy().squeeze()

    # Get the indexes of the classes ordered by softmax
    # (larger first)
    idxs = np.argsort(softmax)[::-1]

    # Loop over the classes with the largest softmax
    for i in range(5):
        # Get softmax value
        p = softmax[idxs[i]]

        # Get class name
        landmark_name = learn_inf.class_names[idxs[i]]

        labels[i].value = f"{landmark_name} (prob: {p:.2f})"


# Putting back btn_upload to a widget for next cell
btn_upload = FileUpload()

btn_run = Button(description="Classify")
btn_run.on_click(on_click_classify)

labels = []
for _ in range(5):
    labels.append(Label())

out_pl = Output()
out_pl.clear_output()

wgs = [Label("Please upload a picture of a landmark"), btn_upload, btn_run, out_pl]
wgs.extend(labels)

VBox(wgs)
```

Please upload a picture of a landmark

Upload (1)

Classify



16.Eiffel_Tower (prob: 0.85)

19.Vienna_City_Hall (prob: 0.06)

14.Terminal_Tower (prob: 0.03)

31.Washington_Monument (prob: 0.01)

15.Central_Park (prob: 0.01)

(optional) Standalone app or web app

You can run this notebook as a standalone app on your computer by following these steps:

1. Download this notebook in a directory on your machine
2. Download the model export (for example, `checkpoints/transfer_exported.pt`) in a subdirectory called `checkpoints` within the directory where you save the app.ipynb notebook
3. Install voila if you don't have it already (`pip install voila`)
4. Run your app: `voila app.ipynb --show_tracebacks=True`
5. Customize your notebook to make your app prettier and rerun voila

You can also deploy this app as a website using Binder: https://voila.readthedocs.io/en/stable/deploy.html#deployment-on-binder

Create your submission archive

Now that you are done with your project, please run the following cell. It will generate a file containing all the code you have written, as well as the notebooks. Please submit that file to complete your project

```
!python src/create_submit_pkg.py
```

✓ 1s completed at 7:27 PM