Front Matter

I want the Notebook to be as informative as possible, but model creating and training process follows some standard procedure that I do not want to repeat. Therefore, if you can, spend time reading the `PROLOGUE/Routine.ipynb` Notebook first.

Paper Implementation - VGG16

Hello, this is my first milestone project - implementation of the VGG16 architecture from the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The paper explored the effect of increasing layers on a model based on the filter size of $3 * 3$ that we previously explored in the `TinyCNN` notebook (that is also the reason why that architecture is called TinyVGG). The architecture was the runner-up in the ImageNet 2014 Challenge for classification.

16 in VGG16 stands for 16 layers, where they are based on two basic units: convolution with filter size $3 * 3$, stride 1, padding 1 and max-pooling with window size $2 * 2$, stride 2. The table shown below, taken from the paper abovem, is the architecture for each of the VGG configuration. In this notebook, we will implement the VGG16-D one.

- 10.48550_arxiv.1409.1556.pdf.png)

In this first notebook, we will focus on getting and transforming the data first.

Downloading and extracting data

The task for our model will be classification, using a bigger dataset called Food101. This is a built-in PyTorch dataset, so the processing can be fairly straightforward. However, it is not fun, so let's take the Kaggle version and process it to what we want.

First, downloading data from Kaggle. The easy way: you can download the zip file (~6 GB), upload it to Google Drive, and then mount Google Drive to Colab. . The slightly harder: you will need to sign up and obtain a Kaggle token, and then use the `kaggle` module to download the data. Let's do that.

In [ ]:

```
!pip install torchmetrics
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
s/public/simple/
Collecting torchmetrics
  Downloading torchmetrics-0.11.0-py3-none-any.whl (512 kB)
     |████████████████████████████████| 512 kB 4.3 MB/s
Requirement already satisfied: kaggle in /usr/local/lib/python3.8/dist-packages
(1.5.12)
Requirement already satisfied: numpy>=1.17.2 in /usr/local/lib/python3.8/dist-pack
ages (from torchmetrics) (1.21.6)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.8/dist-
packages (from torchmetrics) (4.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages
```

```
(from torchmetrics) (21.3)
Requirement already satisfied: torch>=1.8.1 in /usr/local/lib/python3.8/dist-packa
ges (from torchmetrics) (1.13.0+cu116)
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages
(from kaggle) (2022.12.7)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (fro
m kaggle) (4.64.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.8/dist-pac
kages (from kaggle) (7.0.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.8/dist-packages
(from kaggle) (1.15.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.8/dist-packages
(from kaggle) (1.24.3)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages
(from kaggle) (2.23.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.8/dist-pa
ckages (from kaggle) (2.8.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.
8/dist-packages (from packaging->torchmetrics) (3.0.9)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.8/dis
t-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-
packages (from requests->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packa
ges (from requests->kaggle) (2.10)
Installing collected packages: torchmetrics
```

In [ ]:

```
!pip install --upgrade mlxtend kaggle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
s/public/simple/
Requirement already satisfied: mlxtend in /usr/local/lib/python3.8/dist-packages
(0.14.0)
Collecting mlxtend
  Downloading mlxtend-0.21.0-py2.py3-none-any.whl (1.3 MB)
     |████████████████████████████████| 1.3 MB 4.1 MB/s
Requirement already satisfied: kaggle in /usr/local/lib/python3.8/dist-packages
(1.5.12)
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.8/dist-packa
ges (from mlxtend) (1.7.3)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.8/dis
t-packages (from mlxtend) (1.0.2)
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.8/dist-pack
ages (from mlxtend) (1.21.6)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.8/dist-
packages (from mlxtend) (3.2.2)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.8/dist-pac
kages (from mlxtend) (1.3.5)
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.8/dist-pac
kages (from mlxtend) (1.2.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-package
s (from mlxtend) (57.4.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/di
st-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/lo
cal/lib/python3.8/dist-packages (from matplotlib>=3.0.0->mlxtend) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-
packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packa
ges (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packa
ges (from pandas>=0.24.2->mlxtend) (2022.6)
```

```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages
(from python-dateutil>=2.1->matplotlib>=3.0.0->mlxtend) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/di
st-packages (from scikit-learn>=1.0.2->mlxtend) (3.1.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.8/dist-packages
(from kaggle) (1.24.3)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages
(from kaggle) (2.23.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages
(from kaggle) (2022.12.7)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (fro
m kaggle) (4.64.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.8/dist-pac
kages (from kaggle) (7.0.0)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.8/dis
t-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packa
ges (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-
packages (from requests->kaggle) (3.0.4)
Installing collected packages: mlxtend
  Attempting uninstall: mlxtend
    Found existing installation: mlxtend 0.14.0
    Uninstalling mlxtend-0.14.0:
      Successfully uninstalled mlxtend-0.14.0
Successfully installed mlxtend-0.21.0
```

First, we need a variable to keep track of the environment we are in as it is different to run the notebook right on Kaggle and run it anywhere else (from the teaching of a Kaggle Grandmaster)

In [ ]:

```python
# Import modules
import os
from pathlib import Path

# Keep an environment variable
iskaggle = os.environ.get('KAGGLE_KERNEL_RUN_TYPE', '')
```

Next, based on the docs, we will need to create a `/.kaggle/kaggle.json` . You can go to File Explorer and create a folder in your machine, or we can code that. I will code.

In [ ]:

```python
# Paste your API here (I have run and then deleted mine)
creds = ''
```

In [ ]:

```python
cred_path = Path('~/.kaggle/kaggle.json').expanduser()
if not cred_path.exists():
    cred_path.parent.mkdir(exist_ok=True)
    cred_path.write_text(creds)
    cred_path.chmod(0o600)
```

Next, let's use the method `dataset_download_cli` to download and unzip data files.

In [ ]:

```python
# Sanity check
!kaggle datasets list
```

```
ref                                                             title
size  lastUpdated          downloadCount  voteCount  usabilityRating
--------------------------------------------------------------
```

```
---------------------------------------------  -----  -------------------
-------------  ---------  ---------------
meirnizri/covid19-dataset                                     COVID-19 Dataset
5MB  2022-11-13 15:47:17          11624        344  1.0
michals22/coffee-dataset                                      Coffee dataset
24KB  2022-12-15 20:02:12           2316         63  1.0
thedevastator/jobs-dataset-from-glassdoor                     Salary Prediction
3MB  2022-11-16 13:52:31           7233        155  1.0
thedevastator/unlock-profits-with-e-commerce-sales-data       E-Commerce Sales D
ataset                           6MB  2022-12-03 09:27:17       1694          4
8  1.0
ahmettalhabektas/argentina-car-prices                         Argentina car pric
es                               8KB  2022-12-05 09:05:21        745          3
4  1.0
mvieira101/global-cost-of-living                              Global Cost of Liv
ing                              1MB  2022-12-03 16:37:53       3260          6
9  0.9705882
thedevastator/uncovering-wage-disparities-in-pennsylvania-s-hi  Higher Education W
ages                           223KB  2022-12-04 15:42:36       1172          3
6  1.0
danela/fatal-alligator-attacks-us                             Fatal Alligator At
tacks US                         5KB  2022-12-15 16:37:57        350          2
6  1.0
kabhishm/best-selling-music-artists-of-all-time               Best Selling Music
Artists of All Time             3KB  2022-12-09 07:04:29        920         39
0.9411765
swaptr/fifa-world-cup-2022-statistics                         FIFA World Cup 202
2 Team Data                     15KB  2022-12-19 00:29:15       2478          5
8  0.9705882
whenamancodes/predict-diabities                               Predict Diabetes
9KB  2022-11-09 12:18:49           7549        119  1.0
die9origephit/fifa-world-cup-2022-complete-dataset            Fifa World Cup 202
2: Complete Dataset              7KB  2022-12-18 22:51:11       1991          8
3  0.9411765
mattop/alcohol-consumption-per-capita-2016                    Alcohol Consumptio
n Per Capita 2016                4KB  2022-12-09 00:03:11       1186          4
3  1.0
swaptr/fifa-world-cup-2022-match-data                         FIFA World Cup 202
2 Match Data                     7KB  2022-12-19 00:30:28       1201          3
3  1.0
laibaanwer/superstore-sales-dataset                           SuperStore Sales D
ataset                           2MB  2022-12-07 08:53:32       1339          3
6  1.0
thedevastator/discovering-hidden-trends-in-global-video-games   Discovering Hidden
Trends in Global Video Games   56KB  2022-12-03 11:21:47        727         39
1.0
thedevastator/the-ultimate-netflix-tv-shows-and-movies-dataset  Netflix TV Shows a
nd Movies (2022 Updated)         2MB  2022-11-27 20:41:41       2272          3
9  1.0
catherinerasgaitis/mxmh-survey-results                        Music & Mental Hea
lth Survey Results              22KB  2022-11-21 10:03:12       2991          6
8  1.0
kabhishm/imdb-100-movie-titles                                IMDB 100 Movies
9KB  2022-12-07 11:36:06            655         32  0.9411765
tirendazacademy/fifa-world-cup-2022-tweets                    FIFA World Cup 202
2 Tweets                         1MB  2022-12-08 19:43:37        958          3
4  1.0
```

In [ ]:

```
path = Path('kmader/food41')
```

In [ ]:

```
if not iskaggle and not path.exists():
    import kaggle
    kaggle.api.dataset_download_cli(str(path))
```

```
Downloading food41.zip to /content
100%|██████████| 5.30G/5.30G [02:58<00:00, 31.9MB/s]
```

We have the data in the zip file. Now all we need to do is to extract them out.

In [ ]:

```
folder_path = Path('food41')
os.mkdir(folder_path)
```

In [ ]:

```
import zipfile
zipfile.ZipFile(f'{folder_path}.zip').extractall(folder_path)
```

Food-101

The dataset is introduced in this paper, consisting of 101 classes, each with 750 training and 250 testing examples, totalling 1000 images each. The dataset comes with a metadata folder, giving information about which image should go into which subset, which is great! The data was already split into traing and testing examples, but we also need a *validation set*. The testing examples were manually selected to contain noise and challenge the model, so we will not touch that, but we will split the training set further to create a validation set. Now, creating a good validation set is an art, but here we will jsut use good ol' random splitting.

First, we will need to format the in the `images` folder into `train` and `test` folders. Next, we will load the data. The process is quite the same, what's new this time is we will random split the training data into training set and validation set, as well as applying more transformation.

In [ ]:

```
# Generic torch process
from torch import nn
import torch
from torch.utils.data import DataLoader

# Specifically for computer vision
import torchvision
from torchvision import datasets, transforms

# Other module(s)
import matplotlib.pyplot as plt
import gc
import json
import shutil
import itertools
```

In [ ]:

```
with open('/content/food41/meta/meta/train.json', 'r') as fp:
    train_dict = json.load(fp)
with open('/content/food41/meta/meta/test.json', 'r') as fp:
    test_dict = json.load(fp)
print(len(train_dict['apple_pie']), train_dict['apple_pie'][-10:])
print(len(test_dict['apple_pie']), test_dict['apple_pie'][-10:])
```

```
750 ['apple_pie/960233', 'apple_pie/960669', 'apple_pie/962315', 'apple_pie/966595
', 'apple_pie/973088', 'apple_pie/973428', 'apple_pie/98352', 'apple_pie/98449', '
apple_pie/987860', 'apple_pie/997124']
250 ['apple_pie/885848', 'apple_pie/886793', 'apple_pie/904832', 'apple_pie/908367
', 'apple_pie/963140', 'apple_pie/981895', 'apple_pie/984571', 'apple_pie/986844',
'apple_pie/99556', 'apple_pie/997950']
```

The list value of a dictionary key contains the strings that are the file paths of the images without the extension. We will use this to copy the images to proper folders.

In [ ]:

```
os.mkdir('data')
```

In [ ]:

```
new_data_path = Path('data')
original_data_path = Path('food41/images')
new_folders = ['train', 'test']
for folder in new_folders:
    if folder == 'train':
        for key, value in train_dict.items():
            value_set = set(value)
            if not os.path.exists(new_data_path/folder/key):
                os.mkdir(new_data_path/folder/key)
            for image in os.listdir(original_data_path/key):
                image_path = key + '/' + image
                image_path = image_path.split('.')[0]
                if image_path in value_set:
                    shutil.copy(original_data_path/key/image, new_data_path/folder
    else:
        for key, value in test_dict.items():
            value_set = set(value)
            if not os.path.exists(new_data_path/folder/key):
                os.mkdir(new_data_path/folder/key)
            for image in os.listdir(original_data_path/key):
                image_path = key + '/' + image
                image_path = image_path.split('.')[0]
                if image_path in value_set:
                    shutil.copy(original_data_path/key/image, new_data_path/folder
```

And we are done! Now we can load data as we like!

But first, let's write some transformations for the images to perform data augmentation.

Data Augmentation

This is a technique to generate more training data by performing operations on the original data (such as flipping, shearing, rotating for images). The artificial data should generate the same output as the original one, but they are different, so hopefully the model is encouraged to learn the general pattern of the data instead of overfitting. Data augmentation is usually seen in training, but there is a technique called "test-time augmentation" that has been passed down among the Kaggle Grandmaster and implemented in the library fastai.

For the transformations, first we have the staples: `ToTensor()` , which turns images to `torch.tensor` objects. You may notice the `Normalize()` with some arbitrary parameters (the first is a list of means for each color channel and the second is a list of standard deviations for each color channel). These are parameters for the normalization of ImageNet dataset and are required by all PyTorch pre-trained models. This may not necessarily be true for our data, but it can be used. PyTorch also recommends having images of size $224 * 224$ pixels, so we use resize to that. The other transformations do what it is called for, with parameters for angle, probability, etc. (Explore more transformations on PyTorch docs.)

Finally, we call Compose to stack these transformations together

In [ ]:

```
train_transforms = transforms.Compose([transforms.RandomResizedCrop(224),
                                       transforms.RandomRotation(35),
                                       transforms.RandomVerticalFlip(0.27),
                                       transforms.RandomHorizontalFlip(0.27),
                                       transforms.ToTensor(),
                                       transforms.Normalize([0.485, 0.456, 0.406],

valid_n_test_transforms = transforms.Compose([transforms.Resize(224),
                                        transforms.ToTensor(),
                                        transforms.Normalize([0.485, 0.456, 0.406]
```

In [ ]:

```
data_dir = Path('data')
train_dir = data_dir/'train'
test_dir = data_dir/'test'
```

In [ ]:

```
train_dataset = datasets.ImageFolder(train_dir, transform = train_transforms)
```