

Assignment 1

Pokemon Type Classification by Images

[21/22 WS] Introduction to Deep Learning
Hyemin Ahn

Human-centered Assistive Robotics
Technische Universität München



www.hcr.ei.tum.de



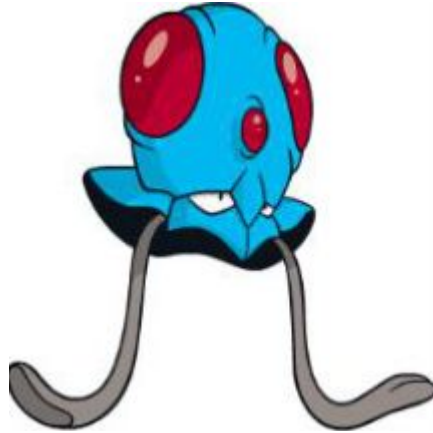
Assignment Description

To Do : Pokemon Type Classification based on the Images



Flying
Bug

...



Water
Poison

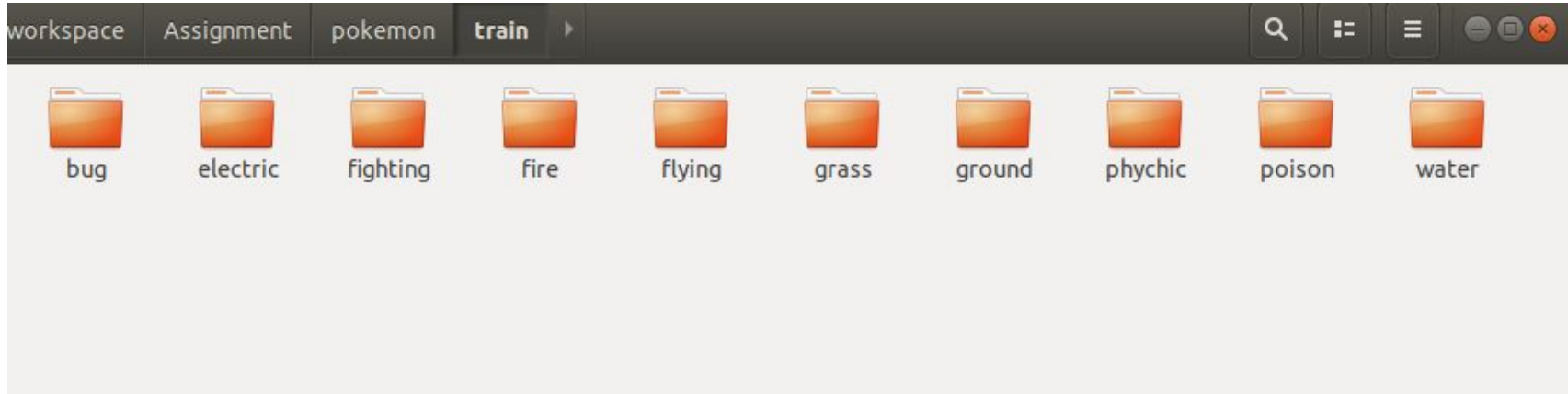
...



Assignment Description

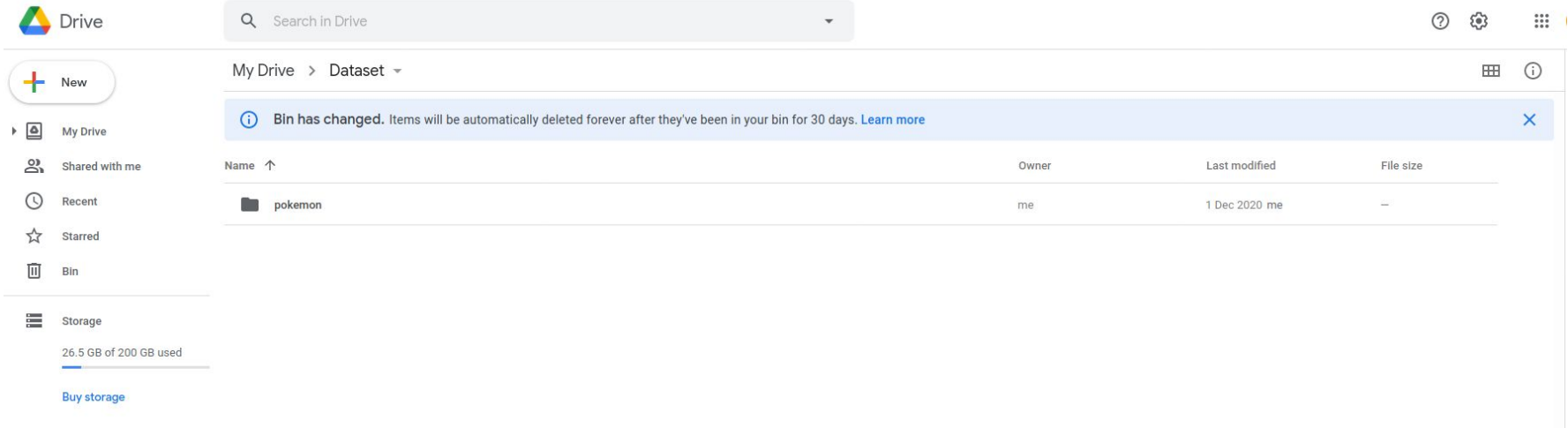
Dataset for Train/Test : Will be distributed to Moodle

- 10 Classes as below



Things you must check before start (in Colab)

1. Put the dataset in your Google Drive!



The screenshot shows the Google Drive web interface. On the left sidebar, the 'New' button is highlighted. Below it, the 'My Drive' section is expanded, showing a list of items: 'Shared with me', 'Recent', 'Starred', 'Bin', and 'Storage'. The 'Storage' section shows a progress bar indicating '26.5 GB of 200 GB used' and a link to 'Buy storage'. The main content area displays the 'My Drive' directory, with a subdirectory 'Dataset' selected. A notification banner at the top of the main area states: 'Bin has changed. Items will be automatically deleted forever after they've been in your bin for 30 days. [Learn more](#)'. Below the notification, a table lists the contents of the 'Dataset' directory. The table has four columns: 'Name', 'Owner', 'Last modified', and 'File size'. There is one entry: a folder named 'pokemon' owned by 'me', last modified on '1 Dec 2020', with a file size of '—'.

Name	Owner	Last modified	File size
pokemon	me	1 Dec 2020	—

Things you must check before start (in Colab)

2. Load your Google Drive to your Colab session
3. Define the path of your Dataset to your notebook code

Let's define some path, and our PokeMon dataset

- Put the "pokemon" folder to somewhere of your Google Drive, and define the train/test path to "train_path" and "test_path"
- To 'model_dir', put the drive's directory path that you want to save your model

```
▶ train_path = './drive/MyDrive/Dataset/pokemon/train'  
test_path = './drive/MyDrive/Dataset/pokemon/test'  
model_dir = './drive/MyDrive/Codes/models' #./drive/MyDrive/Path/To/Save/Your/Model  
classes = ['bug', 'electric', 'fighting', 'fire', 'flying', 'grass', 'ground', 'psychic', 'poison', 'water']
```

Things you must check before start (in Colab)

4. Give the “train_path” or “test_path” as an input argument when creating an instance of **PokemonDataset**.

```
class PokemonDataset(Dataset):
    def __init__(self, data_path, classes):
        self.data_path = data_path
        self.classes = classes

        # organize path information for __len__ and __getitem__
        self.img_path_label = list()
        for c in self.classes:
            img_list = os.listdir(os.path.join(self.data_path, c))
            for fp in img_list:
                full_fp = os.path.join(self.data_path, c, fp)
                self.img_path_label.append((full_fp, c, self.classes.index(c)))
```

```
batch_size = 64

train_dataset = PokemonDataset(train_path, classes)
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_dataset = PokemonDataset([test_path, classes])
test_dataloader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

num_classes = len(classes)
```

Things you must do! ***Super Important***

1. Define your “Model” class!

- Please name your class as “Model”, to make grading easier!!

```
class Model(nn.Module):  
    def __init__(self, feat_dim = 2048, output_dim = num_classes):  
        super(Model, self).__init__()  
  
        self.feat_dim = feat_dim  
        self.output_dim = output_dim  
  
        self.backbone = torchvision.models.resnet50(pretrained=True)  
  
        self.backbone.fc = nn.Linear(feat_dim, output_dim)  
  
    def forward(self, img):  
        out = self.backbone(img)  
        return out
```

Things you must do! ***Super Important***

2. Conduct your Training Phase
3. Save your model in “Checkpoint” (file_name.pth)

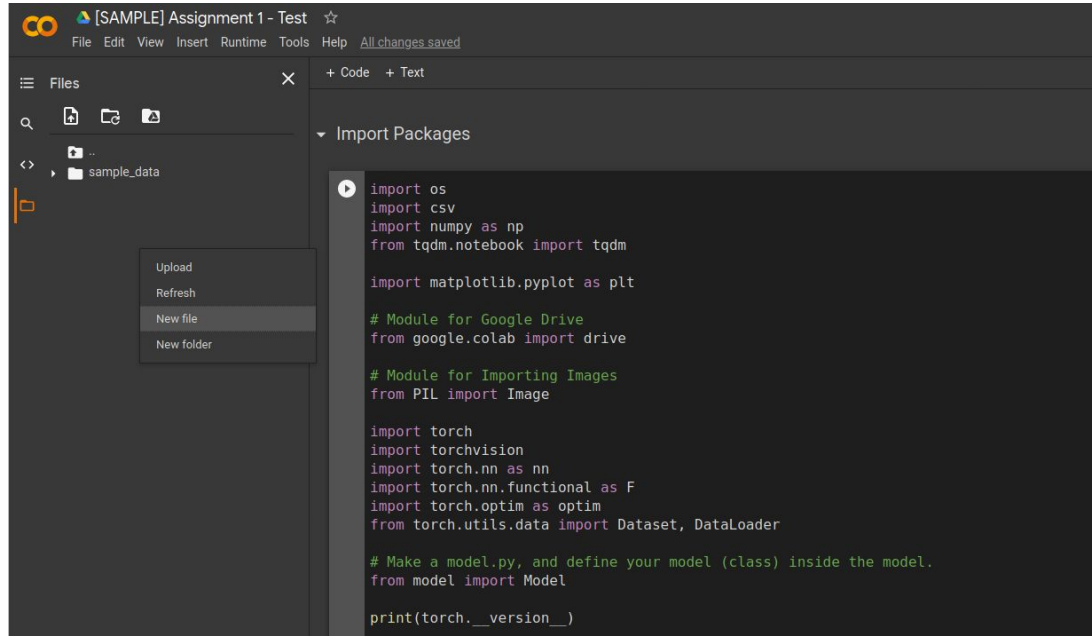
```
# save the model and optimizer's information before the evaluation
checkpoint = {
    'model' : Model(),
    'model_state_dict': model.state_dict(),
    'optimizer_state_dict': optimizer.state_dict(),
}

# Save the checkpoint - you can try to save the "best" model with the validation accuracy/loss
torch.save(checkpoint, tmp_path)
if (epoch+1) % save_stride == 0:
    torch.save(checkpoint, os.path.join(model_dir, 'pokemon_{}.pth'.format(epoch+1)))
torch.save(checkpoint, os.path.join(model_dir, 'pokemon_recent.pth'))
```

```
model_dir = './drive/MyDrive/Codes/models'
```


Things you must do! ***Super Important***

4. Check if your saved model works well on sample test code.



```
[SAMPLE] Assignment 1 - Test ☆
File Edit View Insert Runtime Tools Help All changes saved

Files
├── ..
└── sample_data

Upload
Refresh
New file
New folder

+ Code + Text
▼ Import Packages

import os
import csv
import numpy as np
from tqdm.notebook import tqdm

import matplotlib.pyplot as plt

# Module for Google Drive
from google.colab import drive

# Module for Importing Images
from PIL import Image

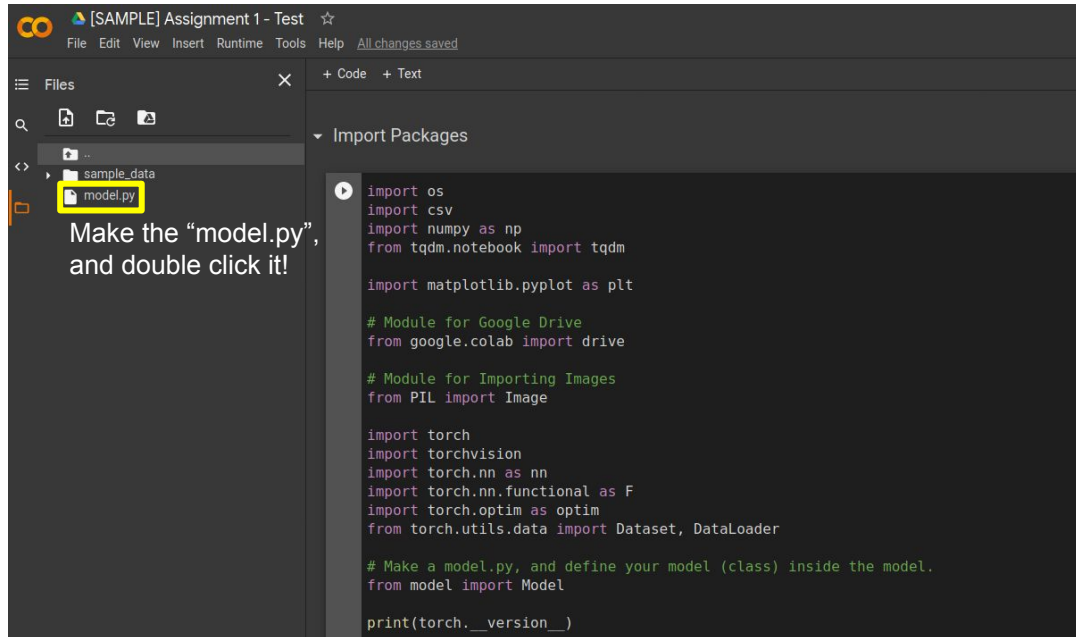
import torch
import torchvision
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader

# Make a model.py, and define your model (class) inside the model.
from model import Model

print(torch.__version__)
```

Things you must do! ***Super Important***

4. Check if your saved model works well on sample test code.



Make the “model.py”,
and double click it!

```
import os
import csv
import numpy as np
from tqdm.notebook import tqdm

import matplotlib.pyplot as plt

# Module for Google Drive
from google.colab import drive

# Module for Importing Images
from PIL import Image

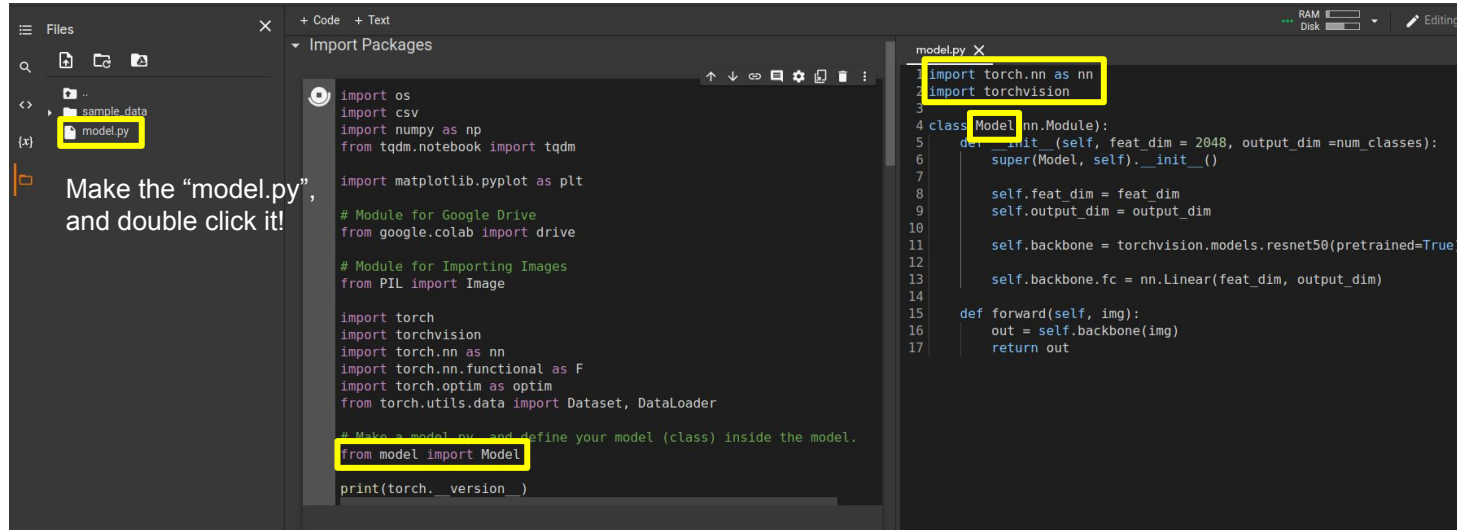
import torch
import torchvision
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader

# Make a model.py, and define your model (class) inside the model.
from model import Model

print(torch.__version__)
```

Things you must do! ***Super Important***

4. Check if your saved model works well on sample test code.



Make the “model.py”, and double click it!

```
import os
import csv
import numpy as np
from tqdm.notebook import tqdm

import matplotlib.pyplot as plt

# Module for Google Drive
from google.colab import drive

# Module for Importing Images
from PIL import Image

import torch
import torchvision
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader

# Make a model.py, and define your model (class) inside the model.
from model import Model

print(torch.__version__)
```

```
import torch.nn as nn
import torchvision

class Model(nn.Module):
    def __init__(self, feat_dim = 2048, output_dim = num_classes):
        super(Model, self).__init__()

        self.feat_dim = feat_dim
        self.output_dim = output_dim

        self.backbone = torchvision.models.resnet50(pretrained=True)
        self.backbone.fc = nn.Linear(feat_dim, output_dim)

    def forward(self, img):
        out = self.backbone(img)
        return out
```

(1) Add your “Model” class’s definition to “model.py”

(2) In there, also import dependent packages.
(i.e., nn, torchvision...)

(3) In .ipynb, check if “from model import Model” works.

Things you must do! ***Super Important***

5. **Zip your codes (train.ipynb, model.py) + saved “only one best” checkpoint (.pth)**
6. **Submit <firstname_lastname>.zip file to hyemin.ahn@tum.de**

```
# save the model and optimizer's information before the evaluation
checkpoint = {
    'model' : Model(),
    'model_state_dict': model.state_dict(),
    'optimizer_state_dict': optimizer.state_dict(),
}

# Save the checkpoint - you can try to save the "best" model with the validation accuracy/loss
torch.save(checkpoint, tmp_path)
if (epoch+1) % save_stride == 0:
    torch.save(checkpoint, os.path.join(model_dir, 'pokemon_r50_{}.pth'.format(epoch+1)))
torch.save(checkpoint, os.path.join(model_dir, 'pokemon_r50_recent.pth'))
```

```
model_dir = './drive/MyDrive/Codes/models'
```

Evaluation

1. **TOP1 accuracy based on the test dataset will be used for grading.**
 - a. Tutor has her own test dataset. Distributed test dataset is only part of it.
2. **This is not a team project!**
 - a. Every student need to submit one's own code and trained model.
3. **Deadline : 17th of December.**
4. **Only THE FIRST SINGLE SUBMISSION IS ACCEPTED!**
 - a. Do not submit your .zip file multiple times.

Some Hints

- Try to make your own “validation” dataset by separating (100% training data) into (80% training data) + (20% validation data).
 - When separating data, try to make (pokemon1) is not included in both (80% training data) and (20% validation data).
 - (Various validation) -> (check test result) -> (various validation)....
 - Try k-fold cross validation. Submit the most robust model.
- Try to add data augmentation tricks.
- Try to fix/freeze the initial layers of ResNet or its variants.
 - Or you can suggest new CNN-based model.

Some Tips for Colab Usage

<https://stackoverflow.com/questions/61254168/prevent-a-google-colab-process-from-being-disconnected>

Criteria for Grading

Accuracy (%)	0	60	65	70	75	80	85	90	95	100
Score	0	20	30	40	50	60	70	80	90	100

Score will be linearly distributed based on this criteria (i.e., accuracy 72.5% → 45)

Experiment Results from Tutor's side (based on Google Colab, best among 10 epoch)

- Resnet 18 : (Accu. of provided test data) : 75.0% (Accu. of tutor's own test data) : 75.7%
- Resnet 34 : (Accu. of provided test data) : 68.3% (Accu. of tutor's own test data) : 71.8%
- Resnet 50 : (Accu. of provided test data) : 78.3% (Accu. of tutor's own test data) : 81.2%

****Running time can be different from account to account.**