Neural Network-baseimage classifier report

Paolo Messina
Student Number: 18037634
Department of Computer Science and Creative Technologies
University of the West of England
Coldharbour Lane
Bristol, UK
Paolo2.Messina@live.uwe.ac.uk

Introduction

Descript The purpose of this report is to classify an image based on its context/content and identify which category or class the object belongs to (Image Classification or Object Categorization). The goal is to recognize a particular object, for example a specific model of a planet, but to check if there is an object belonging to the class in the image.

Being able to identify an object belonging to a certain class it is not a simple task.

It is necessary to generalize and abstract from the single image, identifying salient parts of the object that belong to the whole class. Through the project, will also try to identify the ideal image classification for this type of problem, evaluating the performance and level of accuracy (Udacity, 2017).

After reflecting on the choice for the creative task for this project, was decided to base it on the classification between astronomical objects, more specifically between planets and galaxies.

Background & Research

With The first step for the realization of the image classifier is to understand how Artificial neural networking works.

Artificial neural networks are mathematical models that represent the interconnection between elements, for the simulation of a network of biological neurons (human nervous system), termed artificial neurons. These models can be used to solve artificial intelligence and other issues that arise in different technological fields.

Neural Networking refer to a connected model of computation, in which the computation is carried out utilizing the interaction between elementary units (neurons) that perform, individually, very simple operations and exchange information utilizing of the links that join them.

The weights and threshold values can be determined through a learning process starting from a training set consisting of input-output pairs, in which the correct classification is associated with the input. The neuron trained using the samples of the set T can be used later to classify new inputs x not belonging to T. The weighted inputs are added together and passed through a function to get the output (Freecodecamp.org, 2020).

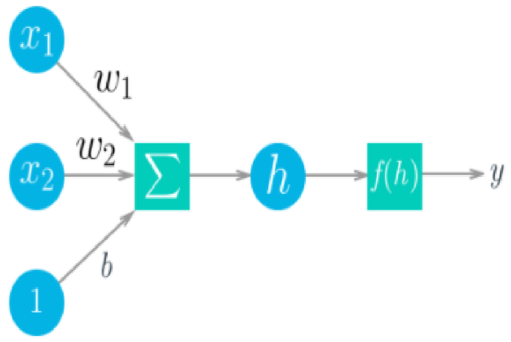


Figure 1 from Udacity 2019

The computation of the neural network is a series of linear algebra operations on tensors. Tensors are a generalization of matrices, a vector is a 1-dimensional tensor, a matrix is a 2-dimensional tensor, an array with three indices is a 3-dimensional tensor.

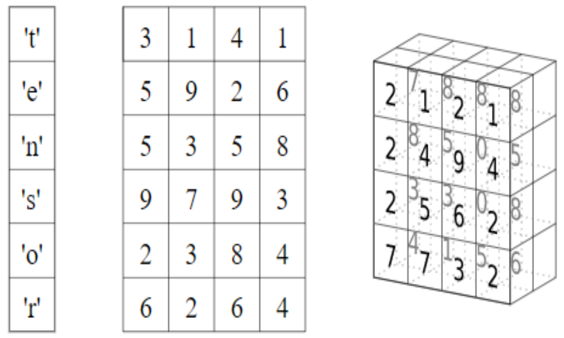


Figure 2 from Udacity 2019

The fundamental data structure used for neural networks are tensors and PyTorch, which are built around tensors (*Udacity, 2019*).

Our task will be to train a convolutional neural network using transfer learning. You can read more about the transfer learning. Which is used the tutorial on PyTorch (Sasank Chilamkurthy, 2017) to learn how to use it whit the ResNet18.

Implementations

For the classification task, was used Google Colab for the deep learning model training.

The choice falls on the fact that Google Colab is a rather simple tool to use and to work on neural networking projects with PyTorch.

The notebook on Google Colab can be saved on Google Drive or GitHub, and in addition, it can be downloaded to be run locally.

To biggin, the idea of the type of creative task as mentioned above falls on astronomical objects, specifically objects such as planets, stars, and galaxies.

However, during the realization of the deep learning model, was decided to change the choice of classes, so was decided to divide it between planets and galaxies.

The choice of an AI-based tool that can distinguish and catechize the stars can be useful to astronomers and beginners who are interfacing first time with astronomy and can help to identify the position of the stars with the telescope.

To group the dataset was used image downloader which is a plugin for google chrome that allows you to download images and the type of format easily.

However, it was not the best choice as most of the dataset downloaded with this tool later turned out to be unsuitable for the dataset as the extension downloaded completely different images from the selected ones.

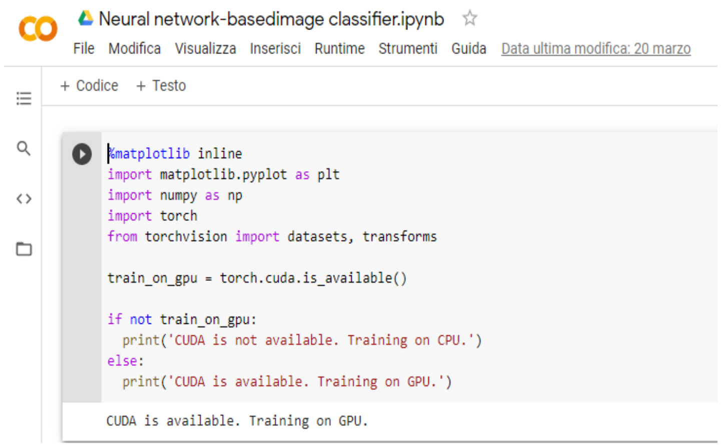
The images downloaded in total are 44 and have been divided into two classes between galaxies and planets, it was planned to extend the dataset to at least more than 100 images per class but this was not possible due to the remaining short time.

Subsequently, the dataset was divided into three sets for training, validation, and testing (70% training, 15% validation and 15% on testing).

The dataset was uploaded to google drive and imported and used on notebooks.

The notebook uses Matplotlib to load the images and then displays them, and has been imported other important libraries NumPy, PyTorch, and Torchvision.

It is used the GPU with Colab in the project, and to test if the GPU is available, a line of code has been implemented where if it is active it uses the GPU through the code.


Neural network-basedimage classifier.ipynb
File Modifica Visualizza Inserisci Runtime Strumenti Guida Data ultima modifica: 20 marzo
+ Codice + Testo

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import torch
from torchvision import datasets, transforms

train_on_gpu = torch.cuda.is_available()

if not train_on_gpu:
  print('CUDA is not available. Training on CPU.')
else:
  print('CUDA is available. Training on GPU.')
```

```
CUDA is available. Training on GPU.
```

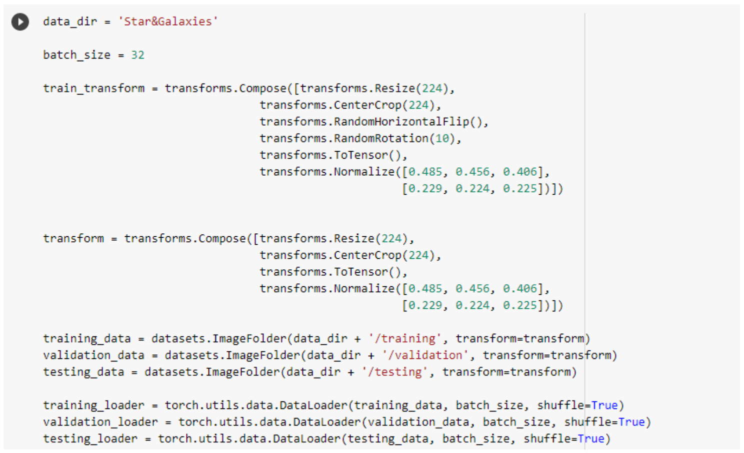
Figure 1 setup GPU for CUDA.

This project is using a convolutional neural network for use of the transfer learning system, and for this notebook was used network architecture, ResNet18 taken from PyTorch.

For displaying images, is used the imShow function from Udacity, and this network requires an image input of 3x224x224 so normalize the mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225], so that these values both can be updated in the imShow function.

The next step is to define the directory which is the path folder that contains the dataset and set the batch size which is going to process 32 images from the dataset.

The transform function set the parameters of the images, and then choose the training, validation and test dataset and prepper them into the data loader.

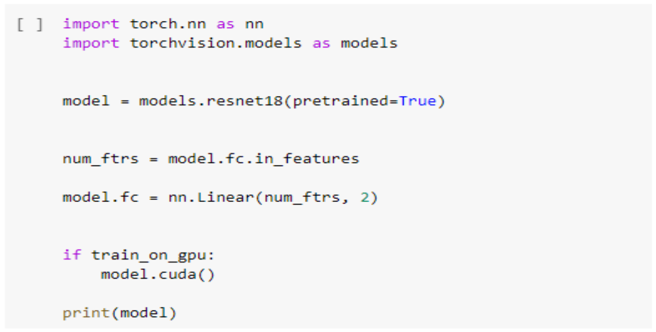


```
data_dir = 'Star&Galaxies'

batch_size = 32

train_transform = transforms.Compose([transforms.Resize(224),
                                      transforms.CenterCrop(224),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.RandomRotation(10),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.485, 0.456, 0.406],
                                                           [0.229, 0.224, 0.225])])


transform = transforms.Compose([transforms.Resize(224),
                                transforms.CenterCrop(224),
                                transforms.ToTensor(),
                                transforms.Normalize([0.485, 0.456, 0.406],
                                                     [0.229, 0.224, 0.225])])

training_data = datasets.ImageFolder(data_dir + '/training', transform=transform)
validation_data = datasets.ImageFolder(data_dir + '/validation', transform=transform)
testing_data = datasets.ImageFolder(data_dir + '/testing', transform=transform)

training_loader = torch.utils.data.DataLoader(training_data, batch_size, shuffle=True)
validation_loader = torch.utils.data.DataLoader(validation_data, batch_size, shuffle=True)
testing_loader = torch.utils.data.DataLoader(testing_data, batch_size, shuffle=True)
```

Figure 2 define directory, image size.

To use ResNet18 network is imported Torch model in the project, and then it loads the ResNet and makes use of the pre-trained weight of the model.

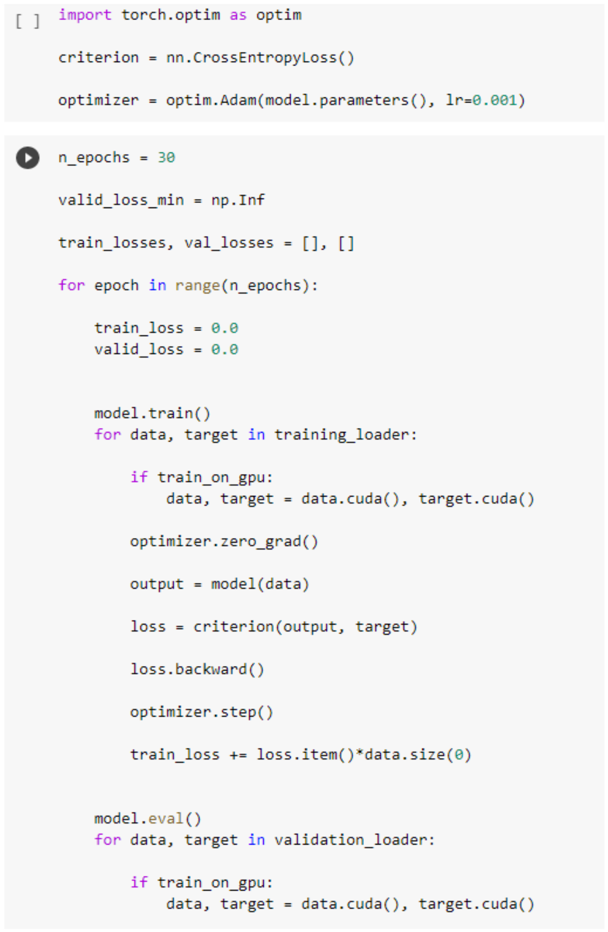
It gets the number of the final layer and then replaces it to output the 2 classes (the galaxies, and the planet classes).



```
[ ] import torch.nn as nn
    import torchvision.models as models


    model = models.resnet18(pretrained=True)


    num_ftrs = model.fc.in_features

    model.fc = nn.Linear(num_ftrs, 2)


    if train_on_gpu:
        model.cuda()

    print(model)
```

Figure 3 setting model.

Afterwards, was set the optimizer and since it has a low number of datasets to train, the learning parameter was set to be 0.001 on 30 number of epochs.



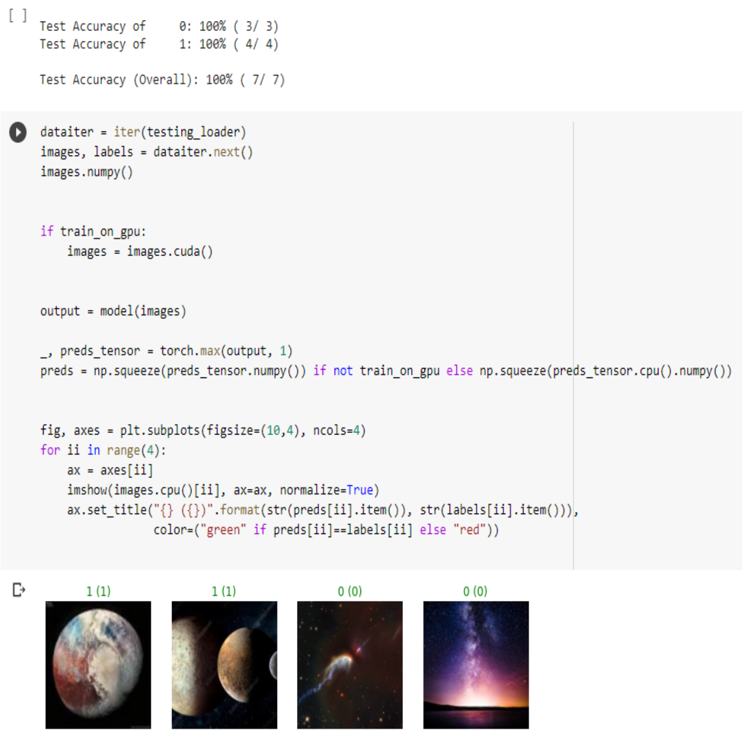
```
import torch.optim as optim

criterion = nn.CrossEntropyLoss()

optimizer = optim.Adam(model.parameters(), lr=0.001)
```

```
n_epochs = 30

valid_loss_min = np.Inf

train_losses, val_losses = [], []

for epoch in range(n_epochs):

    train_loss = 0.0
    valid_loss = 0.0


    model.train()
    for data, target in training_loader:

        if train_on_gpu:
            data, target = data.cuda(), target.cuda()

        optimizer.zero_grad()

        output = model(data)

        loss = criterion(output, target)

        loss.backward()

        optimizer.step()

        train_loss += loss.item()*data.size(0)


    model.eval()
    for data, target in validation_loader:

        if train_on_gpu:
            data, target = data.cuda(), target.cuda()
```

Figure 4 optimizer, training, and validation function.

For each epoch was saved the low values, and then passed on train the model and then validate it for each epoch set. The results give back a model with the percentage of accuracy and the total loss of the training and validation of it.

In the end, was created a function for visualizing the sample of the test result which was set the output 1 for planets and 0 for galaxies.



```
Test Accuracy of     0: 100% ( 3/ 3)
Test Accuracy of     1: 100% ( 4/ 4)

Test Accuracy (Overall): 100% ( 7/ 7)
```

```
dataiter = iter(testing_loader)
images, labels = dataiter.next()
images.numpy()


if train_on_gpu:
    images = images.cuda()


output = model(images)

_, preds_tensor = torch.max(output, 1)
preds = np.squeeze(preds_tensor.numpy()) if not train_on_gpu else np.squeeze(preds_tensor.cpu().numpy())


fig, axes = plt.subplots(figsize=(10,4), ncols=4)
for ii in range(4):
    ax = axes[ii]
    imshow(images.cpu()[ii], ax=ax, normalize=True)
    ax.set_title("{} ({})".format(str(preds[ii].item()), str(labels[ii].item())),
                 color=("green" if preds[ii]==labels[ii] else "red"))
```

Figure 5 test loss, test accuracy and classification.

Conclusion

Based on the results obtained from the model result, test loss (which is 0.027599), and all test accuracy in overall 100% in both classes, it is possible to agree that it has been obtained a good level of results.
However, it would have been necessary to expand the quantity of the dataset size to have more material to train the model.

References.

Abhiojha8, (2019) Deep Learning Pytorch Tutorial. *Udacity* [online]. [Accessed 1 March 2021].https://github.com/udacity/deep-learning-v2-pytorch/tree/master/intro-to-pytorch

Sasank Chilamkurthy, (2017) Transfer Learning for Computer Vision Tutorial. *Pytorch* [online]. [Accessed 19 March 2021].https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

Udacity, (2017) Intro to Deep Learning with Pytorch. *Udacity* [online]. [Accessed 10 March 2021].https://classroom.udacity.com/courses/ud188

Freecodecamp.org, (2020) Pytorch For Deep Learning - Full Course / Tutorial. *YouTube* [online]. [Accessed 30 February 2021].https://www.youtube.com/watch?v=GIsg-ZUy0MY&ab_channel=freeCodeCamp.org