ImageClassification

Team name: "What could go wrong?"

Team members: Szakszon Mátyás - DXR372, ~~Pete Dávid - GVJ529, Hajszter Donát - QOZ2I7~~

Pete Dávid és Hajszer Donát neve, azért van áthúzva, mert november 16-án úgy döntöttek, hogy abbahagyják az Msc képzést, úgyhogy egyedül Szakszon Mátyás készítette a feladatot.

Project description:

The goal of this project is to compare and demonstrate the advantages of using pretrained neural networks vs randomly initialized ones for image classification. Build an image classification pipeline for a smaller dataset (e.g. CIFAR-10), and train both a randomly initialized and an ImageNet-pretrained network (e.g. ResNet) for the task. Compare their performance.

Functions of the files in the repository:

- **build/Dockerfile** : Describes the process of the containerization of the solution.
- **build/requirements.txt** : Contains the installed requirements of the project with fixed version numbers.
- **build/resnet.ipynb** : Contains the implementation and evaluation of a `PreTrained` and a `Not PreTrained` resnet model. It also contains a small `Gradio` implementation that shows the difference between the two model. We can upload any `Image` we would like to test the two different models with.

How to run it:

1. Navigate from the Project's root directory to the build directory.
2. Run this command: `docker compose up -d`

Introduction

For the task i choosed the Resnet-18 model because my task is to test the difference between a pretraiend and not pretrained model. The Resnet-18 model is perfect for that porpuse because you can load a pretrained version of the network trained on more than a million images from the ImageNet database or just use the not pretrained version and train it myself.

The ResNet-18 is a convolutional neural network that is 18 layers deep. It has around 11 million trainable parameters. It consists of CONV layers with filters of size 3x3

Dataset

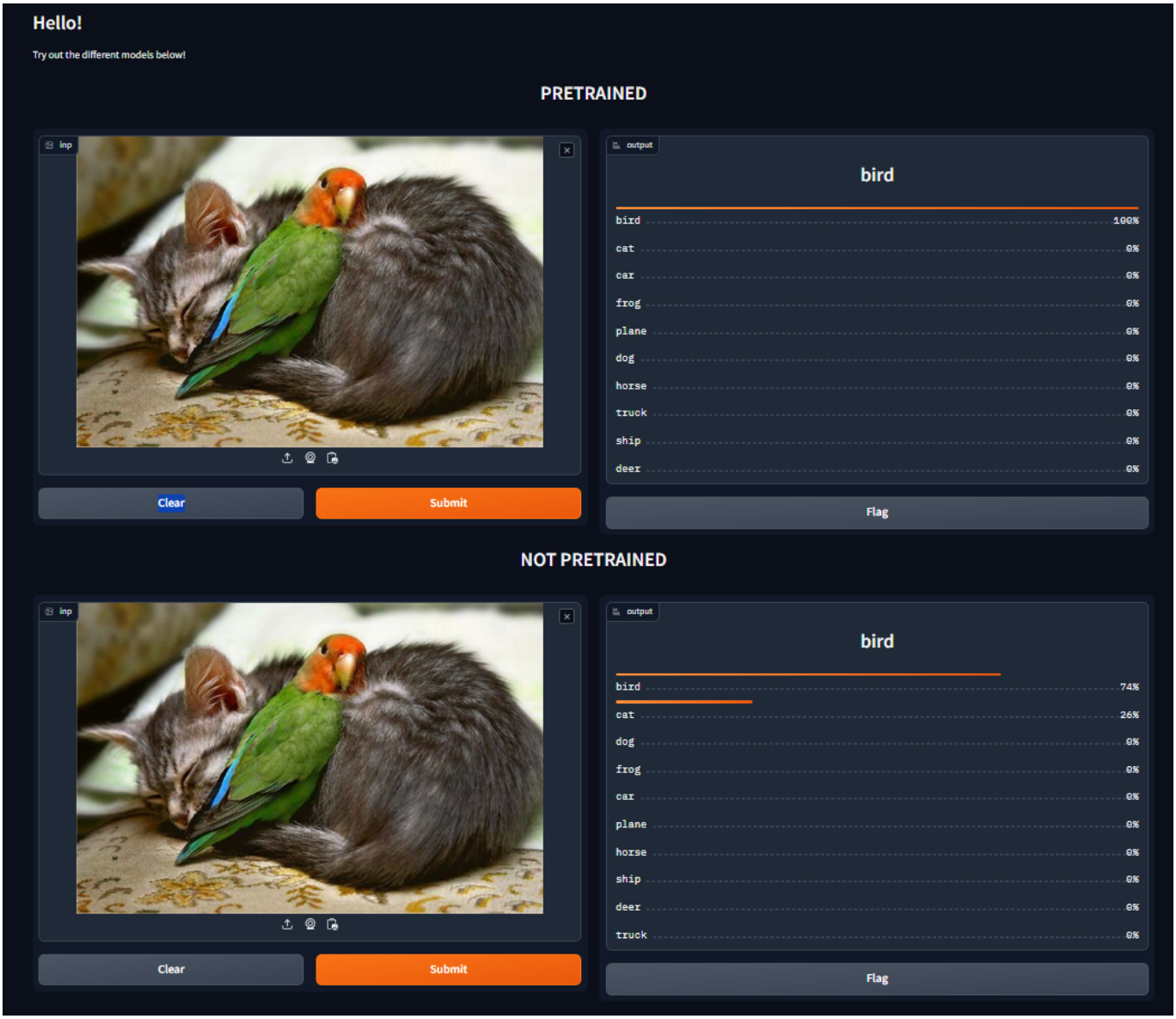
I choose the CIFAR10 Dataset which contains around 50000 labeled images for 10 different classes.

The classes: `plane, car, bird, cat, deer, dog, frog, horse, ship, truck`

Interactive UI usage

I implemented an interactive UI with Gradio that allow us to upload any `Image` we would like and we can test the models with it. There are two `input` and `output` sections first one for the pretrained and the second one is for the not pretrained. If we upload any image in the input fields and submit it will auto select that image for the other input as well just to make it a little easier to test the two models with the same input. The ideal solution would be to use only one input field and two output field but it did not seem possible with Gradio but it is still pretty good to test the models.

After we uploaded and sumbitted the image we are able to see the results in the output field. It shows the model's probabilities for the 10 different classes (sum 100%).



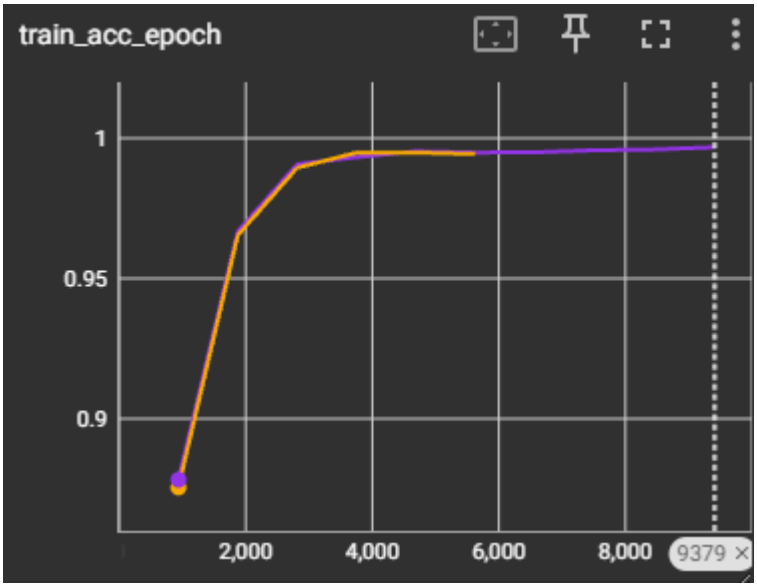
Methods of training

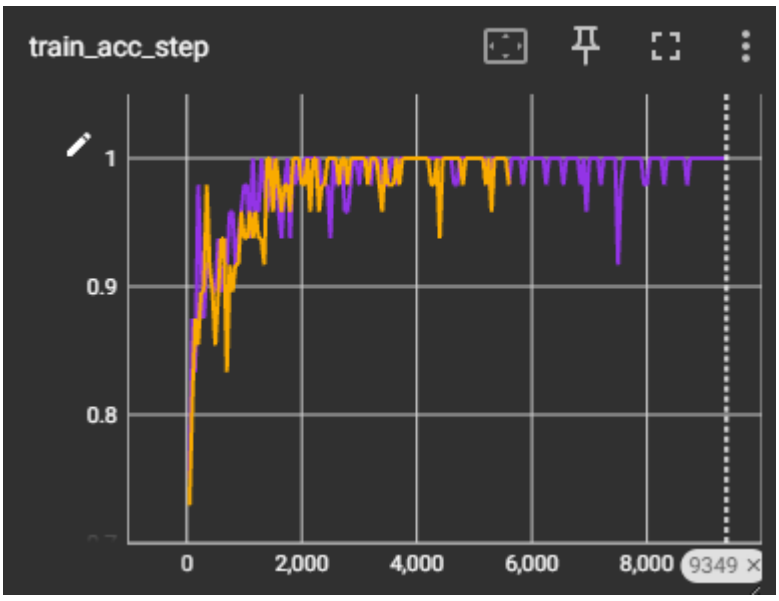
The dataset was resized to 224x224, and the images were normalized. Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])the mean and standard deviation values used here. The values [0.485, 0.456, 0.406] represent the mean values for the red, green, and blue channels, respectively. The values [0.229, 0.224, 0.225] represent the standard deviation for the red, green, and blue channels.

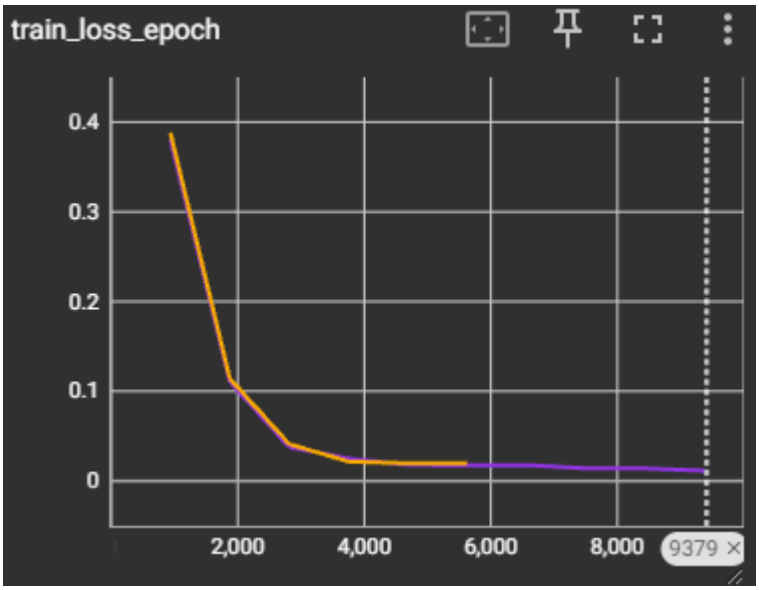
I used a train-validation split with 90-10 ratio. The test set is completely separate and is not involved in the training/validation split. It is created using the CIFAR-10 test dataset, which is distinct from the training

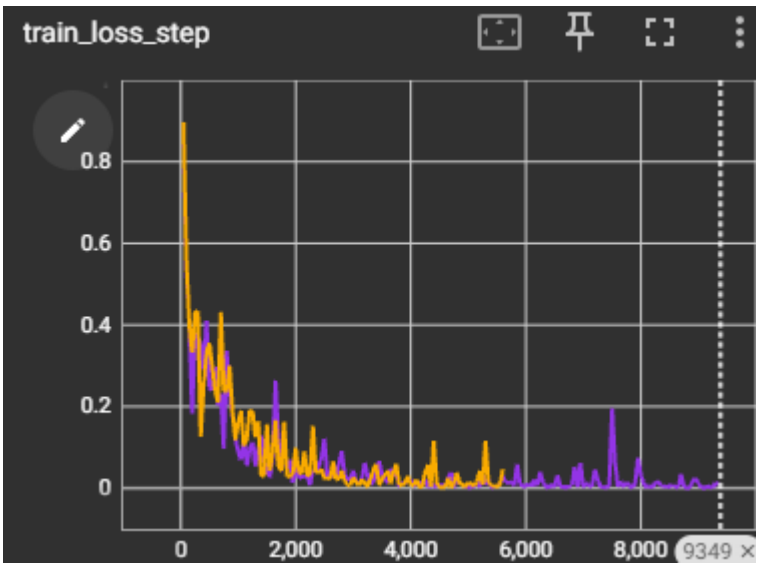
dataset. The dataeset contains 10 different classes that we can classify using our models.

There is a visual representation of the Training of the two models. The Purple is the PreTrained and the yellow is the Not PreTrained.









I trained on my own PC with a RTX 3070 GPU. The training takes about 5-15 minits for each model so the whole training is about 20 minits.

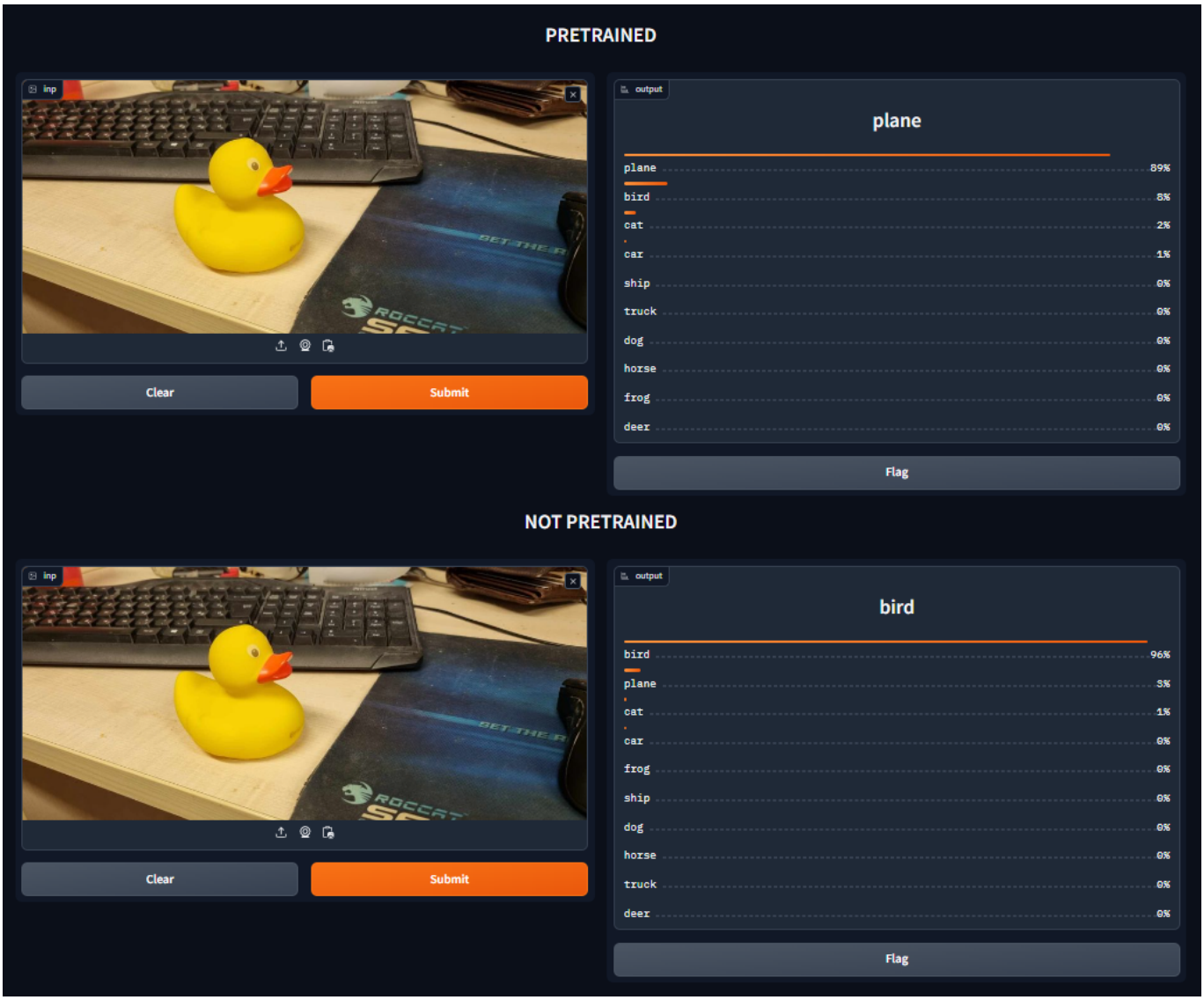
Results

Model	Epochs	Test Loss	Test Accuracy
PreTrained	10	0.2228	0.9369
Not PreTrained	10	0.2316	0.9405

It's pretty surprising but the `Not PreTrained` model is basicly as good as the `PreTrained` version of it. On basic pictures from the internet both of the models give the same results and they are very accurate however i tested the models with selfmade pictures and there i could find very different guesses from the models which is pretty intresting.

As we can see in the picture above that the PreTrained model is way more self-confident because it usually guesses the first class with a higher value than the Not Pretrained. Which is instresting because it's not clear if it's a good or bad property of the model.

My favourite test:



As we can see it's the `Not PreTrained` model that guessed this one correctly.

Conlusions

The final conlusions is that in numbers there is almost no difference between the two models but they give very different percentages compare to that.

I most enjoyed the project is visualizing the results with Gradio it was pretty intresting. However i would have preffered to do it with an actual team so we can try out more intresting and cool stuff together.