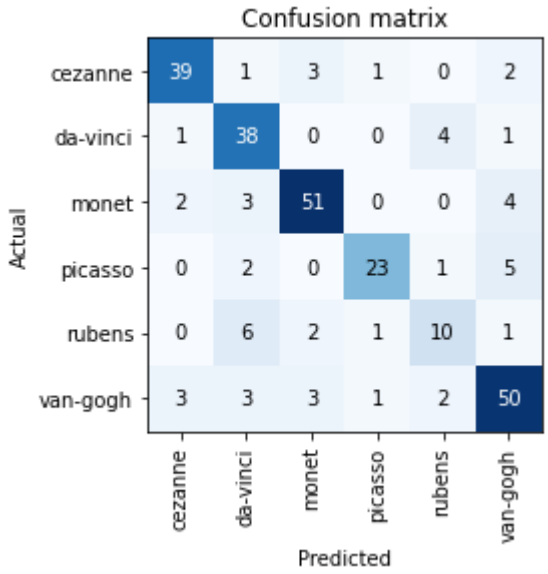
Computer vision

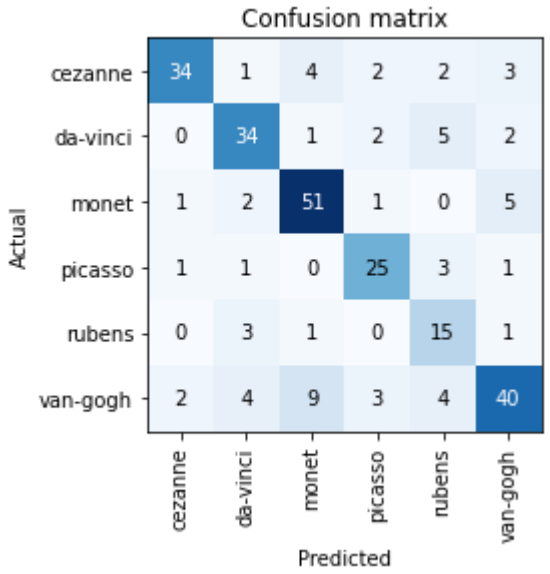
Classifier

We chose to classify the paintings of different painters. First, we started with collecting data. To do this we used the Bing Image Search API from MS. We used this API because we were already familiar with it through the FastAI book. We also tried the Google Image Search API, but we had worse results, so we went back to the Bing API.

When we first put our paintings trough a resnet34 we got an accuracy of 80% which isn't bad, but we can do better so we improved our model. This is when we started using weighted classes because there isn't an equal number of paintings of every painter. So, the model was biased towards painters with a lot of paintings. Weighted classes are used to change the weight of each class. Classes (painters in our case) with less data get a higher weight and classes with a lot of data get a lower weight so the model isn't as biased.



Then we put our data through a resnet50. This got out accuracy up to 85%. We can't blame our model too much for some wrong classifications because sometimes the paintings of 2 painters are almost identical.



Neural Style Transfer

We chose to make a NST as an extension on our project. The principle behind NST is simple, you have two images. One which content you want to show and one which style you want to use. Then we define two distances, one for the content (Dc) and one for the style (Ds). Dc measures how different the content is between the two images while Ds measures how different the style is between the two images. Then we take a third image, the input, and transform it to minimize both its content-distance with the content-image and the style distance with the style-image.

Most of the code comes from a tutorial from PyTorch (https://pytorch.org/tutorials/advanced/neural_style_tutorial.html). We changed how the input images get loaded because this model expected that every image was the same size, this isn't the case when you get 'random' data from the internet.

Our code for the NST and the classifier with a couple of examples of the NST can be found here: https://github.com/YoriVerbist/Bot-Ross

Deploying the model

GCP

First, we tried deploying our model on GCP trough the app engine. Every time we deployed our app, we got an error that it ran out of memory. We spend two days trying to fix this (we tried changing the memory from 256Mb to 1Gb to 2Gb), but we kept getting the same error. This is when we decided to deploy it in an VM on DigitalOcean

Digital Ocean

We deployed a VM on DigitalOcean and installed Nginx with uwsgi.

You can try the classifier here: http://128.199.63.207:5000/