Digital Humanities Hackathon 2023: Detecting Similar Images

Aleksi Suuronen

This report documents the work and research made during the Digital Humanities Hackathon 2023 at University of Helsinki. I was assigned the task to come up with a solution for detecting similar images in our dataset. The dataset consisted of tens of thousands of scientific illustrations, which were scraped using image segmentation from a collection of 18th century books. The different methods used for this task were Image Hashing, and a machine learning approach which used transfer learning and convolutional neural networks. In the next sections, I will present and compare these methods in more detail.

1 Image Hashing

In Image hashing the compared images are first converted to hashes. Similar looking images have very similar hashes. These hashes are then compared, and if they differ little enough by some determined threshold, the images can be classified as similar. Image hashes are different compared to cryptographic hashes by nature. If some cryptographic hashes would be used in this case, even a slight difference in the images would create a completely different hash and make the comparison impossible.

In the project, the conversion from an image to a hash happens through a specifically designed library for image hashing, called *ImageHash*[1]. *ImageHash* library contains multiple kinds of different hashing algorithms, such as *average hashing*, *perceptual hashing* and *wavelet hashing* to mention a few. In the hackathon prototype *average hashing* algorithm was used, and it showed some decent results. As future work, other hashing algorithms could be also tested to see how they compare to the *average hash* and what kind of results they give.

Image hashing is a good technique for spotting almost identical, duplicate images. It can also classify some images correctly as similar, which have a minor difference, for example in image quality and lighting, some times surprisingly

Correspondence aleksi.suuronen@helsinki.fi
Draft June 7, 2023

accurately. However, most of the time it is not as accurate as the machine learning approach and lacks a certain kind of "intelligence" what the machine learning model brings. Example of a research, where this kind of image hashing is combined with machine learning methods and string processing algorithms to detect image reuse in the same dataset can be found in [2].

2 Transfer Learning with ResNet-18

For the machine learning approach, transfer learning and PyTorch's ready made and pretrained convolutional neural networks were used. In the hackathon prototype, a residual network with 18 layers called *ResNet-18* was selected due to being recommended by a tutorial found on the internet. Theory behind the residual networks with multiple layers can be found in [3] and more practical documentation how to use the models in your code in [4].

First the data was preprocessed, which included scaling and normalization of the images. After that feature vectors were generated from each of the image. Using these feature vectors is a very common technique when doing image processing or image recognition with a computer. The similarity between the images is calculated by using cosine similarities, which is a technique based on linear algebra, were the inner product of the vectors is taken and then normalized. This cosine similarity now determines a *"similarity score"*, a value between 0 and 1. These images were then compared so that one *"query image"* was selected randomly from the images, which was then compared to the other images. Larger similarity score means that the image is more likely to be more similar to the query image.

Due to the vast amount of images and data in the project, a cutoff value was determined and used to get the very similar looking images. In the hackathon prototype code, there is some room for improvement to make the code run faster. The biggest bottlenecks are resizing the images and creating the feature vectors. With large datasets, a high-performance computing environment is suggested to be used. During the hackathon we used *Puhti* supercomputer [5], which was provided by CSC.

In my opinion, this machine learning approach was a better option compared to image hashing as it was more accurate most of the time. As future work, refactoring the code to be more performant and trying models with more layers, for example *ResNet-34, ResNet-50, ResNet-101* or *ResNet-152* could show more interesting results.

References

[1] ImageHash source code. Bucher, Johannes (2022). https://github.com/JohannesBuchner/imagehash

[2] Copyright and the Early English Book Market: An Algorithmic Study. Duhaime, Douglas (2019). https://www.earlybookmarket.com/press-piracy.html

[3] Deep Residual Learning for Image Recognition, He et. al (2015). https://arxiv.org/abs/1512.03385

[4] PyTorch documentation: Models and pre-trained weights, Torch contributors (2017). https://pytorch.org/vision/main/models.html

[5] CSC User Guide, (2023). https://docs.csc.fi/computing/systems-puhti/

A Code

The code used for the hackathon prototypes can be found from my GitHub