ALGORITHMS FOR MASSIVE DATA

Project 4: picture recognizer

Author
Davide Matta

Contents

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work, and including any code produced using generative AI systems. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

1 Introduction

The goal of this project is to use deep learning techniques to build a classifier on the Prado Museum Picture dataset[1]. Among the possible classifications, I chose to classify art pieces based on their author. In order to do so, I implemented Convolutional Neural Networks (CNN). In the following chapters I will talk about the dataset, the preprocessing of data, the algorithm used and the results obtained.

2 Dataset

This project is based on the Prado Museum Picture dataset, available on Kaggle[1] and distributed under the MIT license. This dataset contains pictures of the works exhibited in the museum, along with some metadata about the art pieces. These aadditional column contain information about the authors, materials, age, etc... of the works, and can be used as labels for a classification problems. Among the possible options, I decided to build a classification system based on the author, since it seemed the most interesting one in terms of possible real world applications.

2.1 Pictures selection

The dataset contains exactly 13487 rows, which correspond to 13487 pictures. However, it is impossible to use them all for this task. In fact, the classifier should handle a manageable amount of classes (i.e. authors), and for each classes there should be a reasonable amount of pictures, otherwise the model is not able to generalize sufficiently.

For this reason, I decided to keep the three most frequent artists: Francisco Goya, Francisco Bayeu and Carlos Haes. In the dataset there are 1079 rows related to Goya, 446 to Bayeu and 326 to Haes. Despite ideally we would want to have more images (excluding maybe Goya), these number still allows to build something, especially considering the artistic differences between the authors. However, I do not consider appropriate to add other artists with even less data and possibly similar to the already included ones.

In addition, I have also undersampled Goya to 500 images, in order to avoid the creation of a classifier that is biased toward that class.

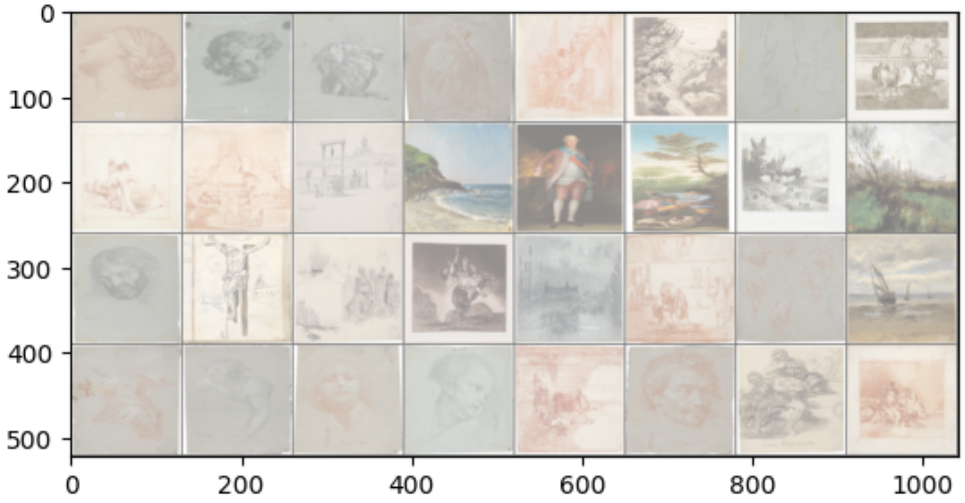

Figure 1: Images

2.2 Data retrieval

As previously stated, only a small portion of the available images will be used. For this reason it is useless to download the whole dataset. Then, rather than downloading the

whole folder from Kaggle, I have only downloaded the csv file, in which links to the images are stored, selected the relevant rows, and then downloaded the images using the request library.

2.3 Data preprocessing

In order to be usable by neural networks, images should be equal sized, so I resized them to 128x128. In Figure 1 are shown some of them.

After this step, it is also necessary to transform the images to tensors, so that they can be read and loaded by deep learning frameworks.

3 Model

The chosen algorithm is Convolutional Neural Networks (CNN). It is a neural network architecture that works particularly well with images, and is then ideal for this task. In particular, other than fully connected layers, CNNs make use of two special types of layers[2]:

- Convolutional layers: they make use of filters to analyze smaller portions of the images in order to spot any significant pattern that can be useful for the classification.
- Pooling layers: they are put between convolutional layers, and we need them to resize the feature map and reduce the number of parameter to estimate.

In this project, I use two convolutional layers, each of the two encapsulated in a ReLU function and a MaxPooling layer. The rest of the network is composed by a fully connected layer with ReLU activation, and finally one last fully connected layer with softmax activation, since this is a multi-class problem.

The loss function is the Cross Entropy, and the optimizer is Adam. The number of epochs was manually tuned to 25, since this seems sufficient for the model to converge to a certain test error.

3.1 Scalability

Neural networks not only perform well with large quantity of data, but they actually tend to improve their performances when trained on larger quantity of data.

Of course, this requires hardware that is capable to handle large quantity of data, something for what Colab (at least in the free version) is not that suitable.

However, we still have a GPU in Colab, which allows a more efficient training of the model. The GPU acceleration effect is more evident when we have more data, thus helps us to scale the project.

In addition, there are some measures that could be taken to make the algorithm more efficient, like for instance pooling layers (that I used).

If the model needs to be used extensively and/or updated frequently, a good solution could be to moove the model to the cloud. For example, Amazon SageMaker is a suitable solution for managing an entire machine learning pipeline such the one developed in this project. SageMaker also offers services for training and tuning models in a more efficient way than what can be done in a colab notebook.

4 Results

The model proved to be effective for this task. In fact, it reached a remarkable 94.09% accuracy on the test set. This is a good result considering the complexity of the task and the small dataset size.

In Figure 2 it is shown how accuracy evolves over epochs. As we can see, the test accuracy reaches a plateau around epoch 13, obtaining only small improvements afterwards. The train accuracy, by contrast, keeps growing, which likely indicates overfitting.

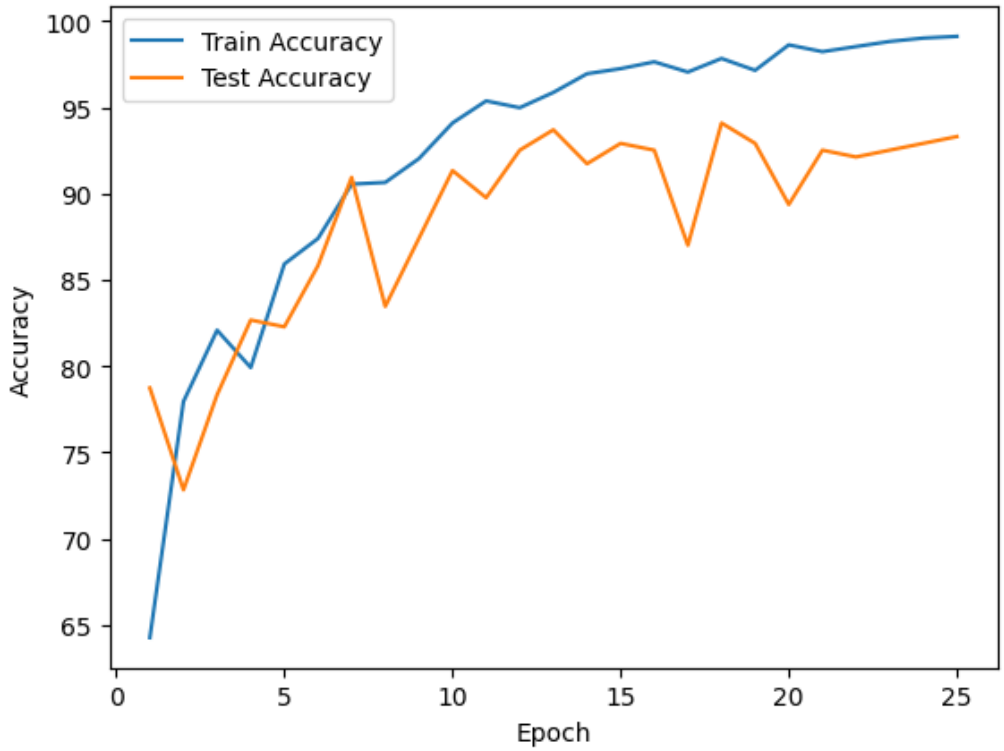

Figure 2: Accuracy

4.1 Reproducibility

The results are fully reproducible thanks to the use of torch.manual_seed or the random_state sklearn parameter every time there was any random component.

5 Possible improvements

- The model could be improved further, for example experimenting with new architectures, not only different CNNs, but also different models, like for instance recurrent neural networks (RNN), that are also good for sequential data like images.
- Regardless of the architecture, the model performances could be improved tuning hyperparameters. However, this is a slow process if we have to rely on the free version of Google Colab, and this is the reason why it is not implemented here.
- Finally, the model is easy to generalize to a n-classes classification, if we are interested in classifying works from other artists (of course we need a reasonable number of pictures for each new author), or to any classification problem applicable to this dataset.

6 Code

The code for this project is available at: GitHub link
The colab version of the notebook is also available: Colab link

References

[1] M. P. Lara, "Prado museum picture dataset." [Online]. Available: https://www.kaggle.com/datasets/maparla/prado-museum-pictures

[2] cs231n, "Convolutional neural networks for visual recognition." [Online]. Available: https://cs231n.github.io/convolutional-networks/