Master’s in advanced Analytics & Data Science

NOVA IMS

Subject	
Deep Learning	
Project	
Finding Patterns in Art Paintings	
Group	7
Team Members	
António Pinto	m20200659
Davide Farinati	m20201080
Tomás de Sá	m20200630

Background

Paintings have been present in human history since the Neolithic period (10,000–3,000 BC). Over time several types of artists began to appear and with them, their painting styles. A painting specialist can identify a painting and its author by looking at it but how? By the patterns found in it. There are a lot of different patterns that can be distinguished in a painting as the use of a repeated palette of colours, the shapes present in a geometrical (mosaics or tessellation) or in a natural way (floral patterns for instance),

The proposed project is training a model in recognizing the style of painting and then using it as a photo style changer based on pre-existing painting styles. The system itself receives a normal photograph (it can be anything from mountains to a portrait) we want to transform and a paint with the style we would like to recreate, to finally output the same first photo with the painting style from the second one.

Once the user introduces its photo and a photo/paint with the style to be recreated, our model is going to be trained based on that painting style inserted by classifying it and then applying the found characteristics on our first uploaded photo.

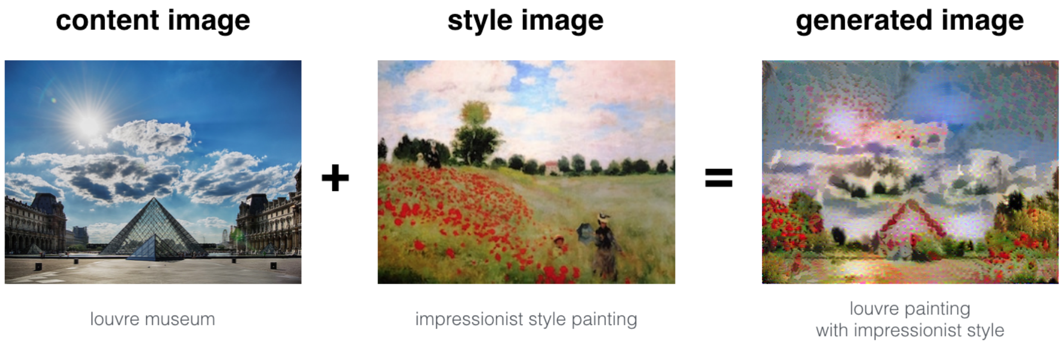


Figure 1 - Proposed project

Materials & Methodologies

Environment

For this project, Google Colab was used as the execution environment. This product allows anybody to write and execute python code through the browser and the added benefit is that Google provides an instance with a GPU that results in slower execution times. In our first tests, Colab managed to train a CNN in two-thirds of the time compared to our local environment, which was using only the CPU power for the execution and so we decided to adopt this platform.

Dataset

The dataset we used in this project had two different options to be chosen:

1. Pandora 18K: 18 classes of paintings with a total of 18038 images.
2. Pandora 7K: 10 classes of paintings with a total of 7740 images.

In the beginning, we decided to use the 18K dataset because it looked more complete in terms of the diversity of paintings but after training the data with different models, we concluded that our system was facing some classification problems. Due to a large number of classes and the similarity between some of them, the system was returning low accuracy results and so we decided to develop our final model based on the 7K dataset.

Some of the models we trained took a lot of time to finish due to the number of classes the dataset had to be trained. We also created a test dataset with 34 images, but it was worthless once it didn't give us any feedback with statistical significance that we could use.

Dataset Partitioning

⇒ Train - 5314 images;
⇒ Validation - 1269 images;
⇒ Test - 34 images.

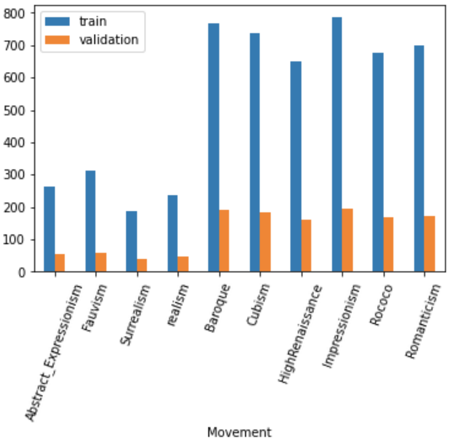


Figure 2 - Dataset Distribution

Model

Once the project was defined and the data was collected and pre-processed, it was time to discuss how we would build the model that would correctly classify the art movement of any given painting that belonged to the classes present in our dataset. The most used class of deep neural networks for image classification is Convolutional Neural Network (CNN).

All our models' executions include the Keras Callback Early Stopping, which stops the execution of the model once the loss in the validation set does not reduce for a specific number of epochs. This method impacts our search for the best configuration possible but is also a big-time saver, and this trade-off is something that we agreed on taking, keeping always in mind that for a future work with a bigger time window this patience can be increased or even discarded.

In our approach, we started by implementing a simple CNN with few layers, and a small number of neurons both in the convolutional part as well as in the fully connected one. Our goal was to evaluate whether the project was feasible both in terms of achieving a suitable solution to our challenge and if the given timeframe was enough. The latter seemed at a first glance an issue but was soon solved by switching our environment to Colab and realizing that the times were acceptable to develop a CNN that would fit our purposes. Furthermore, for this first network, comparing it with later versions, the patience is half the epochs. Each epoch took around 115s to run and a total of 13 epochs for the model to stop running due to the early stopping call back we imposed. The final accuracy on the validation set was 35%. This number was considered good since on a naive approach, of randomly selecting a class we would guess it correctly with a probability of 10% and bearing in mind that we still must do some model finetuning, resulting in higher accuracy.

Following the first model, several network architectures were tested, including a grid search that for time constraints could not take much longer to run hence testing 4 possible combinations (appendices 1 and 2). Even though, the group would prefer to extend the research field by including more possibilities and parameters, like changing the optimizer to use Adam instead of RMSprop (the former gave worst results when tested in other models), the results that came out from the grid search gave some guidelines on what parameters might be best to tune for achieving the best possible. All the models used in the CNN finetuning, that are not mentioned in the report, are not included in the notebook to display only what is strictly relevant for the project.

Soon we started noticing how some parameters positively impact the network, like having more neurons on the fully connected part and having lower values for the dropout rate, always mitigating the overfitting. After several tests, we got to what is now our best model (appendix 3), with an accuracy of 48% on validation. The model was trained with a higher number (12) for the epochs patience in the early stopping call back which also contributed to being the longest model, in terms of training epochs, we have trained. However, the number of epochs retained by our model's configuration was 20. This parameter was obtained by specifying that we would like to keep the weights of the epoch correspondent to the minimum validation loss.

After achieving what was considered a good solution for a CNN, we decided to perform some trials in our approach to improve the network, but this time, the changes would be on the dataset. As seen before, there are some discrepancies in terms of images per class in our training dataset and so we decided to level these numbers, by having the same number of images per class to assess if this would improve our model's performance. The results were conclusive and showed that this approach worsened our model. Other trials like applying a XGBoost model on top of the CNN were conducted, but again, it did not output any promising results and so we decided to discard this approach.

Finally, to compare our model's performance with a well-known model that proved to be state of the art in classifying the images, we decided to use the pre-trained VGG19 model, with the weights of the ImageNet dataset classification problem, adding the fully connected part. The results obtained were good, with an accuracy of 42% on validation, but still below our best model.

Neural Style Transfer

Neural Style Transfer is a known and commonly used technique that exploits the structure of Convolutional Neural Networks to take a target image and a reference image and produce an output that retains both the content of the target image along with the style of the reference image. To perform so, it uses a loss function that is minimized using gradient descent. The loss function is itself composed of three different losses, the content loss, the style loss and the total variation loss. The first one is defined simply as the L2 distance between the two images representations in a prefixed layer, usually a deeper one, where the Network recognizes more complex patterns of the image, therefore the content. For the style loss, we use the L2 distance between the gram matrices (multiplication of the image matrix and its transpose that approximate the ‘style’ of an image) of the style image representation in prefixed layers and the output image. Lower layers are normally used for the style loss because it is where the Network recognizes more simple patterns, therefore lines and colours, that are what define the style of a painting. The total variation loss helps to distribute changes in the entire image, and not only where the curve and points concentrate, and it does so by penalizing larger gradients during the transfer process.

Results

Evaluation Measures

For the model construction part, the measure used to assess the model’s performance was the accuracy on the validation set. We also assessed this same metric on a test set, but we decided not to give it much importance since we only had one image per class for most of the categories. Assessing the model’s performance on the validation set while training it and considering the values obtained as an indicator for the models’ success, might be subjected to some biases, as the data had already some contribution to the model development. Also, a possible measurement that the group conducted was to compare the best model with a state-of-the-art CNN (VGG19), which showed to underperform our best model.

In the neural style transfer part, one of our biggest concerns while developing the system was the way we could measure if the system itself was finding the right patterns in the paintings already existing and if those patterns were being applied in the right way on user’s photos. We came up with an idea that, in our vision, could give us good feedback on how the system was behaving by creating two stages:

- Stage 1: from the dataset with known artist paintings, we applied our developed convolutional neural network and see if the model can find the patterns/details in the right way. But how to check if the patterns were right detected? From the accuracy/confusion matrix classification report with f1 score and other insights.
- Stage 2: change our photos to a previously detected style - harder to define the accuracy. The main way to check if the transformation was done correctly was by checking visually if the image was changed correctly.

Models Results

As described before, the group tested several models and methodologies to achieve what is now the best model. The following image demonstrates how the model behaved in the training stage. We can observe that the model is not performing overfitting on the training set, although this was not the case in the earlier stages of the architectures that led to the final one. Overcoming overfitting was possible using dropout layers twice, with a ratio of 0.2, where the output of the neurons in the fully connected part was discarded. After the 20th epoch, the model stopped improving its results, reaching the patience threshold and consequently stopping its execution.

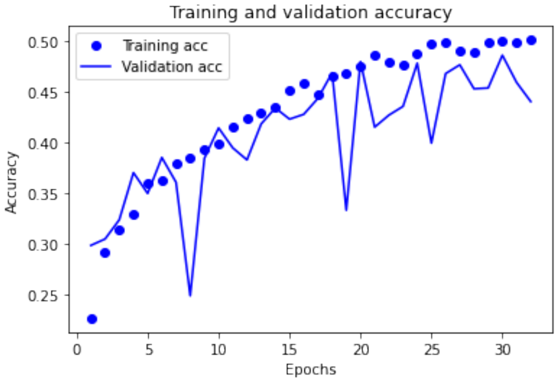


Figure 3

Comparing the accuracies on validation while training the network (figure 4), we can observe that it is only after the 17 epochs that our best model starts to outperform the others, reaching above the 47% accuracy mark several times. The model trained on the balanced dataset is the one that got the worst results, not surpassing the 40% accuracy. The VGG19 is the second-best model, showing a consistent output of results placing a little bit below our best model (figure 5), yet reaching acceptable results. In the figure below, only one of the grid search models is displayed for readability purposes, if desired, the performance of the models that resulted from the grid search can be consulted in the appendix. As aforementioned, the models do not last the same because of the early stop call back implemented, to reduce the running times.

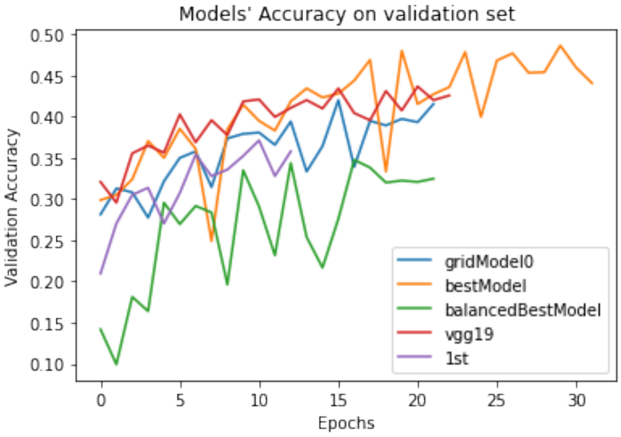


Figure 4

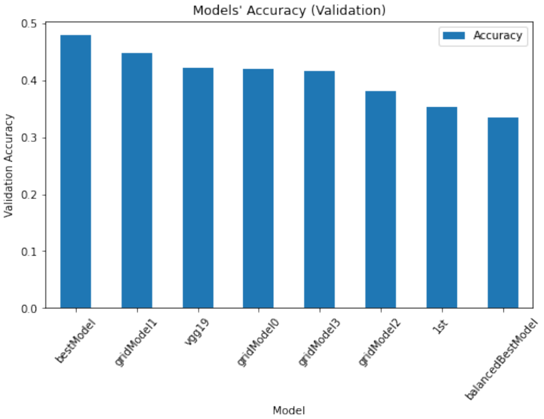


Figure 5

Neural Style Transfer Results

We started the Neural Style Transfer by selecting a content image and a style image. The first one was a normal picture of ourselves, while the second one was a painting from the training dataset of our classification task.

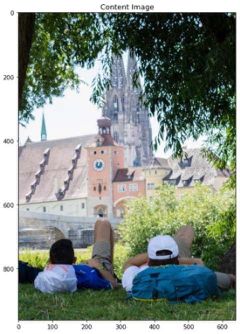


Figure 6

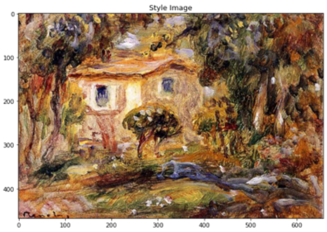


Figure 7

Then, we applied the Neural Style Transfer with the VGG19 model, a pre-trained model, with layers and weights set as shown in the literature.

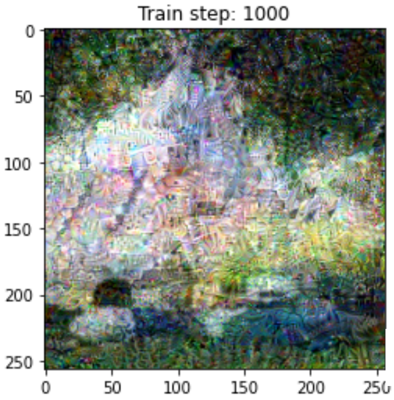


Figure 8

Then we performed again the transfer using this time our trained model on the painting classification task. For the first run, we set the weights as explained in the literature and with the VGG19 model, therefore with earlier layers as the style layers and deeper layers as content layers. The first results were disappointing (figure 9) since also the content of the style image was transferred to the output image. Then we tried different combinations of layers and weights and the results improved (figures 10 and 11).

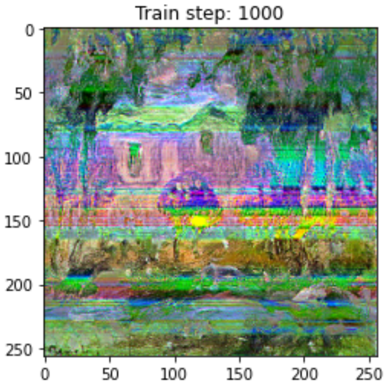


Figure 9

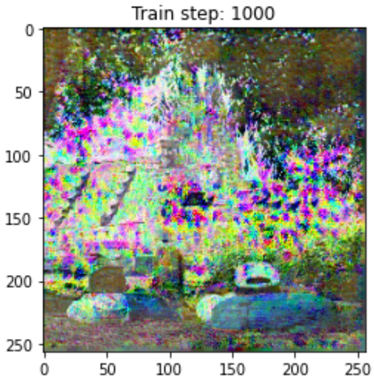


Figure 10

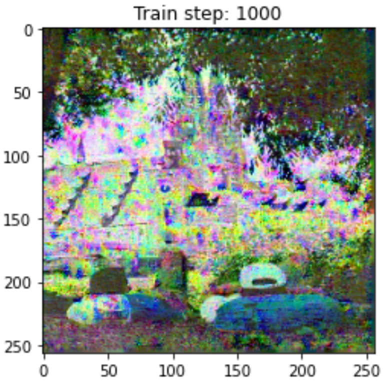


Figure 11

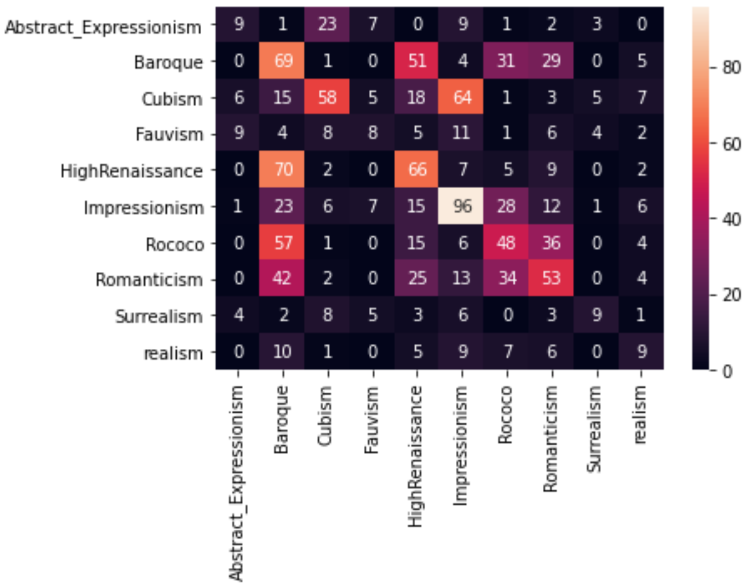
The main difference between the first and the second and third result images was the selection of layers. Indeed, for the first one, we used the layers as explained in the literature, deeper layers for content and earlier layers for style but in the results, it was clear that not only the style but also the content was passed to the output image. Therefore, we tried inverting the layer, earlier layers for content and deeper layers for style, and the results improved by a lot. Comparing our results with the output of the Neural Style Transfer with the VGG19 model, we can see that the only differences are the colours, being clearer in the latter one. Neural Networks are black boxes and therefore almost impossible to interpret, but our model was trained for a classification task based on the genre, and therefore the style, of the painting, and not the content of the image as the VGG19 model. This may be why in our model the style is recognized in feature maps (convolutional layers) that are deeper in the model, and therefore the asset explained in the literature is not a good fit for our model.

Challenges

- ⇒ it was quite hard to decide if the painting style would be associated with the artist or with a specific paint. If an artist only had one painting style, this would be easier to decide but once an artist has more than one painting style, it is a challenge to decide which one to use;
- ⇒ to acquire a good dataset with a variety of artist's paintings.

Classification Problems

After training the model the main resource that we used to evaluate the results on the validation set was the confusion matrix (shown below) together with a classification report, that included important measures such as accuracy, precision, recall and f1 score. As we can see from the confusion matrix, and also the accuracy results presented above, the model misclassified many paintings. The only class in which most of the paintings are classified correctly is Impressions, for all the others it shows much confusion.



	Abstract_Expressionism	Baroque	Cubism	Fauvism	HighRenaissance	Impressionism	Rococo	Romanticism	Surrealism	realism
Abstract_Expressionism	9	1	23	7	0	9	1	2	3	0
Baroque	0	69	1	0	51	4	31	29	0	5
Cubism	6	15	58	5	18	64	1	3	5	7
Fauvism	9	4	8	8	5	11	1	6	4	2
HighRenaissance	0	70	2	0	66	7	5	9	0	2
Impressionism	1	23	6	7	15	96	28	12	1	6
Rococo	0	57	1	0	15	6	48	36	0	4
Romanticism	0	42	2	0	25	13	34	53	0	4
Surrealism	4	2	8	5	3	6	0	3	9	1
realism	0	10	1	0	5	9	7	6	0	9

Figure 12

Conclusion

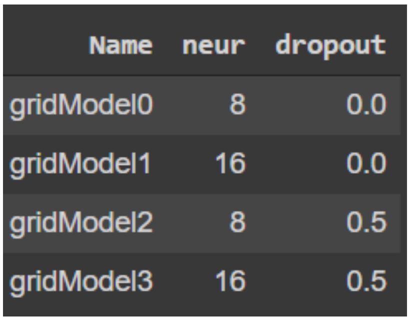
From the beginning, we knew that the project we were proposing to develop was very challenging but interesting at the same time. Looking for a complete dataset with the main styles we would like to reproduce was not easy in the beginning - some were too large and with too many classes and others too incomplete.

After training several models with different approaches and methods, the best model (appendix 3) registered an accuracy of 48% which is a very good result when analysing the time and tools available to develop it. There were implemented very different methods with a variety of approaches that enabled us to reach these results even though the time consumed when training these models due to lack of CPU power was an obstacle.

References

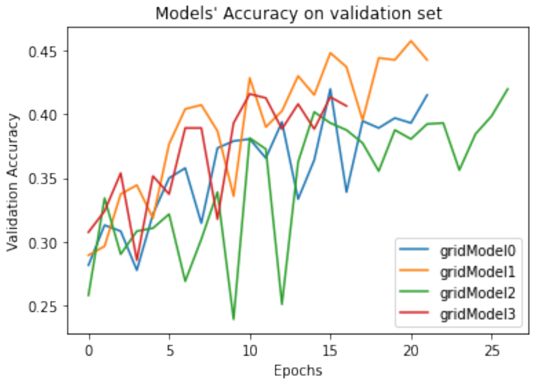
[1] Dataset - Pandora 7k: imag.pub.ro/pandora/pandora_download.html
[2] medium.com/gradientcrescent/neural-art-style-transfer-with-keras-theory-and-implementation-91b7fb08ee81
[3] keras.io/examples/generative/neural_style_transfer/
[4] deepart.io/
[5] iq.opengenus.org/vgg19-architecture/
[6] kaggle.com/ikarus777/best-artworks-of-all-time
[7] medium.com/gradientcrescent/neural-art-style-transfer-with-keras-theory-and-implementation-91b7fb08ee81
[8] arxiv.org/pdf/1508.06576.pdf
[9] kaggle.com/supratimhaldar/deepartist-identify-artist-from-art
[10] arxiv.org/pdf/1602.08855.pdf
[11] reddit.com/r/MachineLearning/comments/3l5qu7/rules_of_thumb_for_cnn_architectures/
[12] vcl.fer.hr/papers_pdf/Fine-tuning%20Convolutional%20Neural%20Networks%20for%20fine%20art%20classification.pdf
[13] towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035
[14] datacamp.com/community/tutorials/implementing-neural-style-transfer-using-tensorflow

Appendix

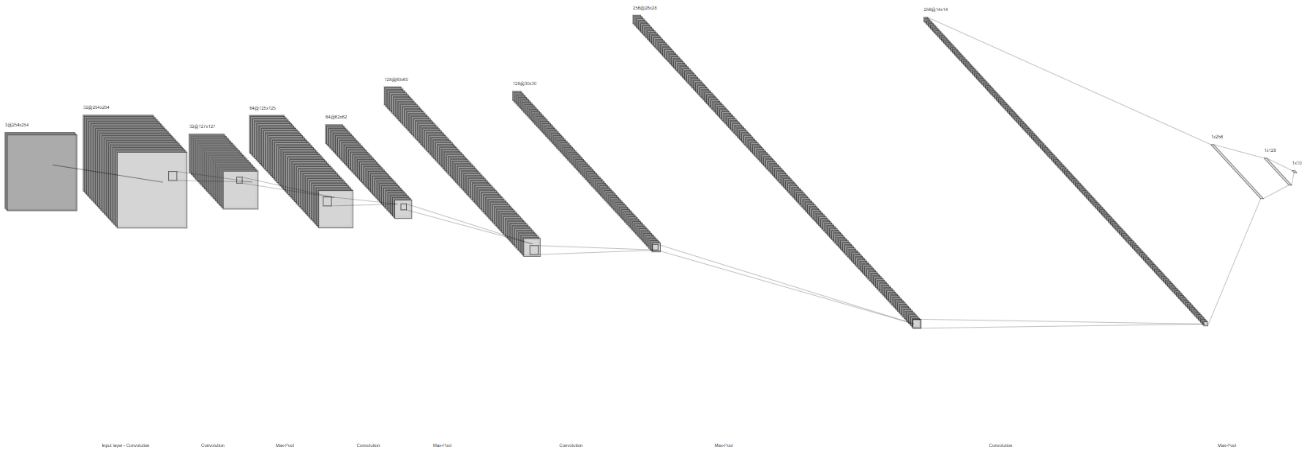


Name	neur	dropout
gridModel0	8	0.0
gridModel1	16	0.0
gridModel2	8	0.5
gridModel3	16	0.5

Appendix 1 - Grid Models Configuration



Appendix 2 - Grid Models Accuracy on Validation



Appendix 3 - Best CNN Architecture