Project 8 Image Processing for explain pintors by their time

Pau Vilanova - 1638223
Ricard Tuneu - 1634796
Aksel Serret - 1639282
Joel Bautista - 1605507

Universitat Autònoma de Barcelona
Grau en Enginyeria de Dades
Deep Learning 2023-2024

INDEX

1. Introduction: **2**

2. Objectives **2**

3. Datasets **2**

3. 2. Data Distribution 3

4. Artists Classifier **4**

5. 1. ResNet-18 5

5. 2. ResNet-50 7

5. 3. VGG-16 8

5. 4. Classifiers comparison 9

5. 5. Process of reducing overfitting of the best model 9

5. 6. Test the best model found with more data 10

5. 7. Pipeline predictions 14

5. Date Sorter **18**

6. 1. Date Classifier 18

6. 1. 1. Top 5 Artists with 100 18

6. 1. 2. Top 100 Artists with all images 20

6. 1. 3. Specific network for Vincent Van Gogh 23

6. 4. Pipeline 24

6. Metric Learning **28**

7. 1. TOP_15_DATES & Random Batchs 28

7. 2. Van Gogh Images & Customized Batchs 29

7. 3. Clustering exploration via embeddings visualization 29

7. 4. Development of the best model 31

7. 5. Pipeline 31

7. Conclusions and Enhancements **33**

Bibliography **33**

Contributors **34**

1. Introduction:

This project aims to utilize advanced deep learning techniques to delve into the world of art history and analyze the evolution of artists over time.

Our main objective is to trace the chronological careers of several artists by examining their works, not only to identify their individual pieces but also to understand how their styles and techniques evolved throughout their careers sorting the paintings chronologically.

To achieve this, we will first need to identify which artist each painting corresponds to, essentially solving a multiclass classification problem by classifying the paintings by artist, and finally, sorting these paintings by date once we have recognized the author of the painting.

As we need to first solve the classification task, our starting point is based on a multiclass classification code that has been adapted and modified to accomplish our task [1].

We understand that each artist is unique, and their artistic trajectory is marked by a series of influences, experiments, and evolutions over time. Therefore, in representing the chronological career of an artist through their works, we hope to capture not only their distinctive style but also moments of change and development in their art.

2. Objectives

Our main objective is to trace the chronological trajectory of paintings by several artists. To achieve this, we need 2 subgoals:

- Build a multiclass classifier capable of classifying paintings by artists.
- Develop a system that enables the sorting of paintings by each artist chronologically.

3. Datasets

Our dataset, sourced from Kaggle[2], comprises 103.250 paintings created by 2.319 artists. Our data cleaning process involves removing images that do not have a recorded creation date, as they are not useful for our chronological classification purposes. After this cleaning process, we are left with 76.887 images and 2.109 artists.

To facilitate testing and agile code development, we have created different datasets formed by the top N artists that appear most frequently in the original dataset (those with the most paintings), selecting a specific number of paintings from each artist. Each of these datasets has been divided into TRAIN and TEST sets, following an 80/20 split, to train the different models created in the practice and evaluate them. Hitherto, the datasets created have been:

- Top 5 artists with 100 images (500 images)
- Top 100 artists with ALL images (28.865)
- Top 15 datas with 100 images (1500 images)
- Images of Vincent Van gogh

3. 2. Data Distribution

In this section, a brief analysis of the distribution of our data will be explained. Considering the provided data, we observe that the distribution of artists (Image 1) appears to be more balanced compared to the distribution of dates (Image 2). There are a total of 2319 distinct artists and 1690 distinct dates. Notably, there are 12 artists that appear exactly 500 times.

Regarding biases in the data, it's evident that the amount of paintings is not equal for each artist or year. Some artists may have a higher number of paintings compared to others, and similarly, some years may have more paintings represented than others. This suggests potential biases in the dataset, indicating that certain artists or time periods might be overrepresented or underrepresented compared to others.

However, in our case, we will always work with the *top n* artists that appear most frequently in the dataset, making the bias towards artists less significant initially.

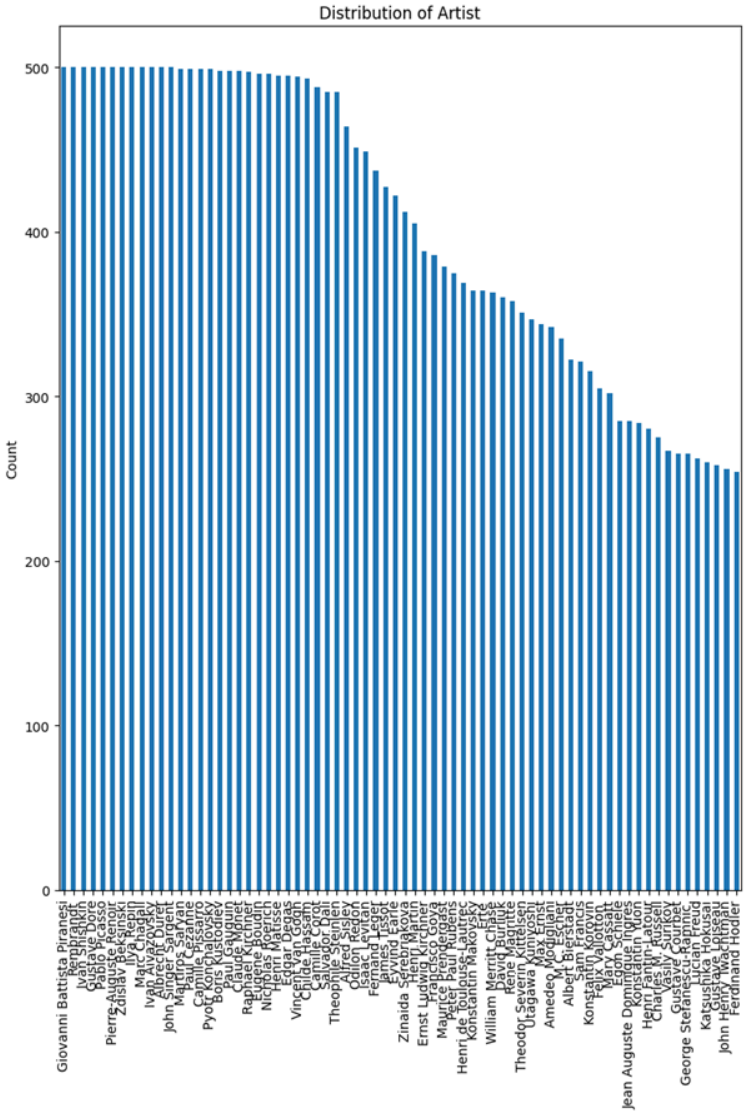

Image 1. Artists Distribution

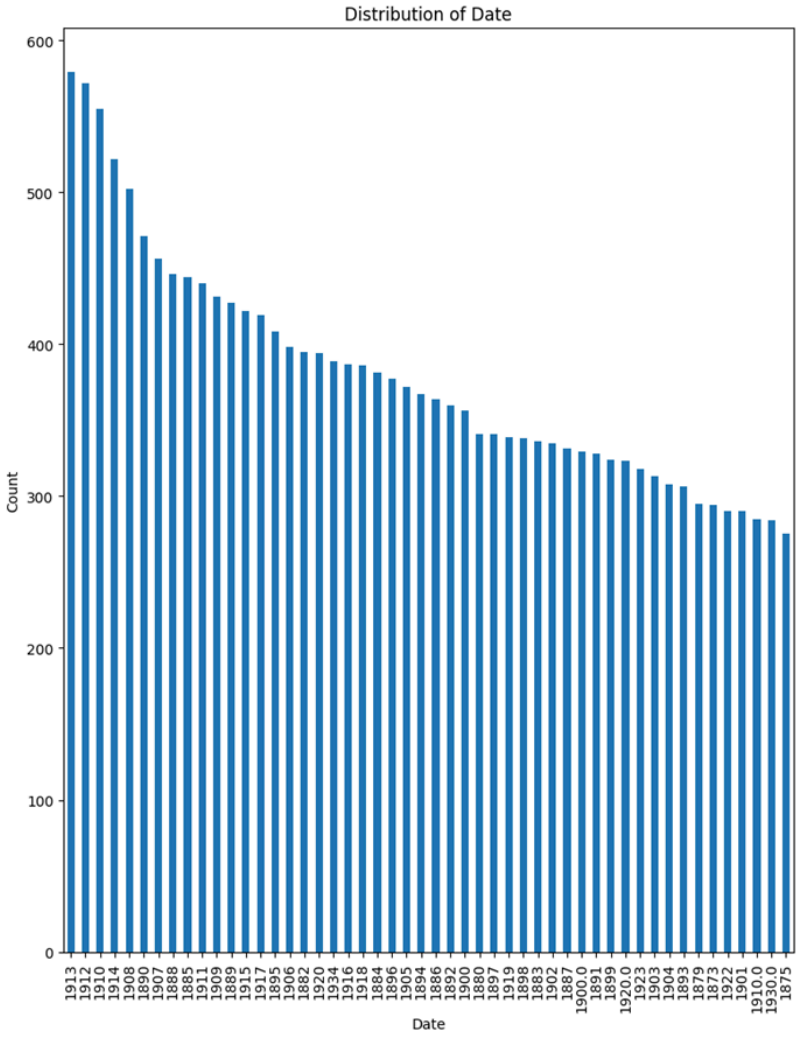

Image 2. Date Distribution

4. Artists Classifier

In this section, we will explain and compare the different models implemented to achieve the objective of classifying paintings by artists. Our intention in this section is to test different classifiers with various combinations of optimizers, learning rate values, and learning rate schedulers. With this approach, we aim to evaluate and compare different models to identify the best one for classifying the images in our dataset by artist. The models used are *ResNet18*, *ResNet50*, and *VGG16*.

Initially, two versions of each model were run: one trained from scratch and one pretrained (parameter pretrained = True). Setting the parameter pretrained = True means that the model uses weights from a network that has been previously trained on a large dataset, typically *ImageNet*. It was clearly observed that the pretrained models yielded better results. From this point onward, we worked exclusively with the pretrained models.

Subsequently, each of these models was first executed with the same learning rate and learning rate scheduler, varying only the optimizers. This allowed us to determine the best optimizer for each model. The optimizers used were:

- **Adam:** Combines the advantages of two other extensions of stochastic gradient descent, AdaGrad and RMSProp. It works well in practice and is robust to noise.
- **AdaGrad:** Adapts the learning rate to the parameters, performing larger updates for infrequent parameters and smaller updates for frequent ones. It is well-suited for sparse data.
- **SGD (Stochastic Gradient Descent):** Updates the weights in the direction of the gradient of the loss function. It's simple and effective, but can be slow to converge.
- **RMSprop:** Maintains a moving average of the square of gradients to adapt the learning rate for each parameter. It helps to stabilize the training process.

Once the best optimizer was identified, we conducted different runs using this optimizer but varied the learning rate scheduler. The different learning rate schedulers used were:

- **StepLR:** Decays the learning rate by a factor every few epochs, which helps to gradually reduce the learning rate.
- **MultiStepLR:** Allows more flexibility by specifying epochs at which to decay the learning rate by a factor.
- **ExponentialLR:** Decays the learning rate exponentially based on the epoch number, providing a smooth transition.
- **CosineAnnealingLR:** Adjusts the learning rate following a cosine curve, starting with a high learning rate and gradually reducing it, then increasing it again.

This methodology allowed us to identify the most effective combination of optimizer and learning rate scheduler for each model. Once this was done, we compared the best version of each model to finally determine the optimal version for classifying images by artist according to our data. It is important to note that all these executions were performed using the *Top 5 artists with 100 images* dataset.

5. 1. ResNet-18

The first model we have used is Resnet18, which is a residual network architecture that was mainly created for image recognition, so in this sense it fits our problem.

As the name suggests, this model has a total of 18 convolution layers and although it is not a very deep neural network, it provides a good balance between performance and computational efficiency.

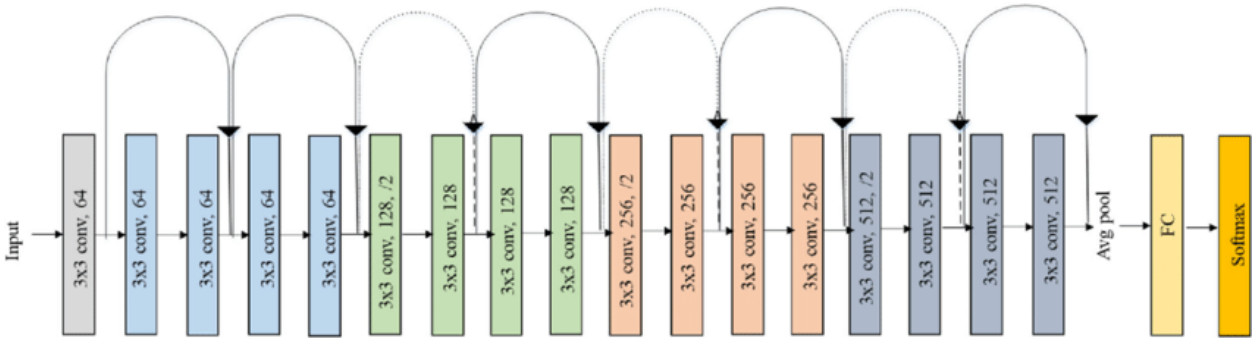

Image 3. ResNet-18 architecture simple schema

With this model, we conducted several tests as we mentioned earlier. All the different executions carried out are in the file *Starting_Points/Classifier/ResNet18/ResNet18_DifferentExecutions.ipynb*. However, the best result was achieved using a learning rate of 1e-4, the *SGD* optimizer and the *MultiStepLR* lr scheduler.
The following image shows the result obtained from training this model:

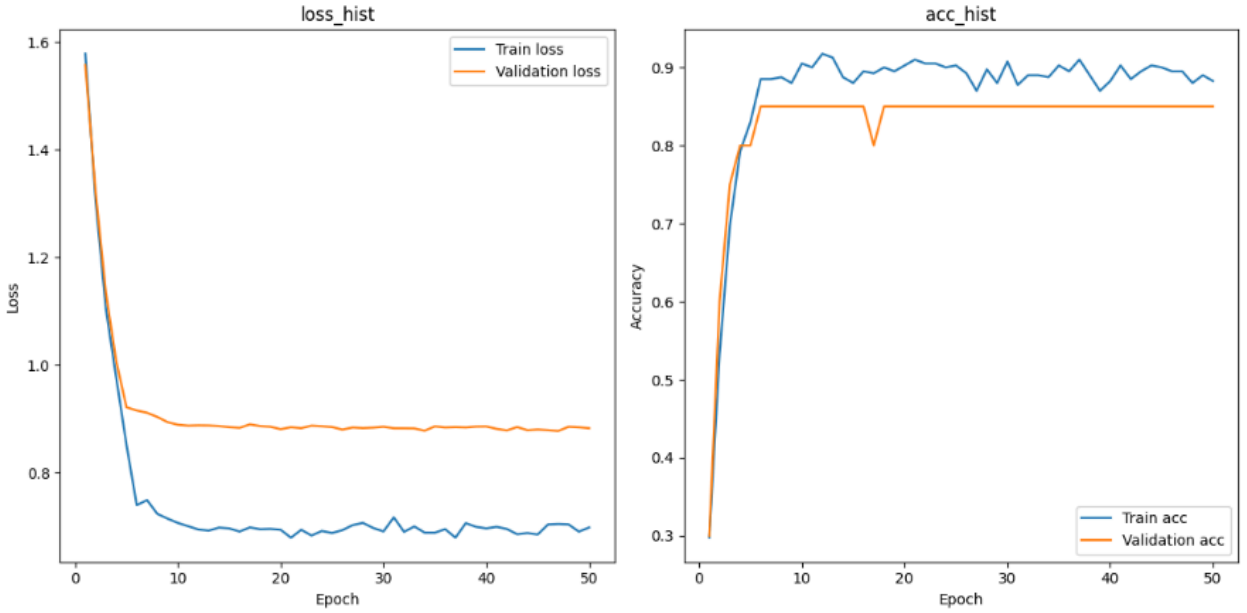

Image 4. Results of ResNet18 best Model with the dataset "Top 5 artists with 100 images"

In the loss graph we can see how the validation curve stops progressing before the training curve, which may indicate that there is some overfitting. For now, we will not take this into account since we are only comparing the performance of the models. It is worth noting that overfitting was also present in all other executions.

In terms of accuracy, validation and training are closer although it is also a little better in training. The difference is small, so it indicates that the model generalises quite well, although it could still be a little better.

5. 2. ResNet-50

The second model we have used is ResNet50, which is another residual network architecture primarily designed for image recognition. This model has a total of 50 convolution layers, offering a deeper neural network that provides a good balance between performance and computational efficiency.

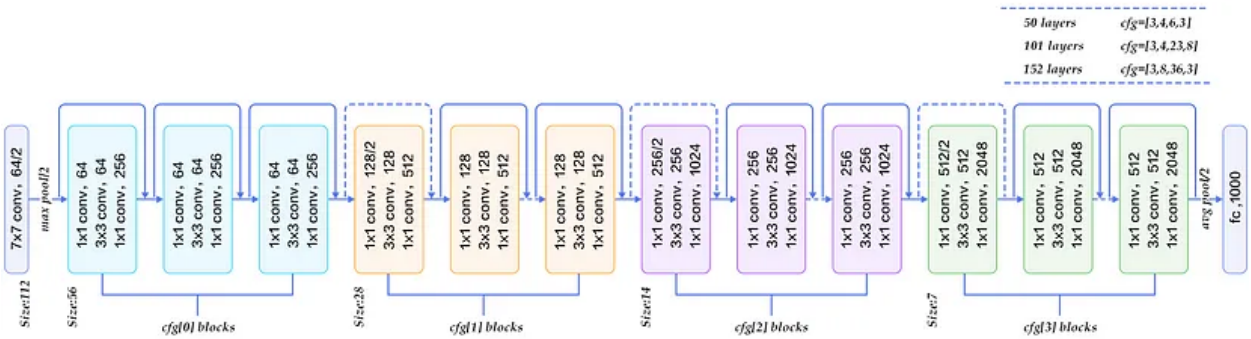

Image 5. ResNet-50 architecture simple schema

All the different executions carried out are in the file *Starting_Points/Classifier/ResNet50/ResNet50_DifferentExecutions.ipynb*. However, the best result was achieved using a learning rate of 1e-4, the *AdaGrad* optimizer, and the *ExponentialLR* learning rate scheduler.

The following image shows the result obtained from training this model:

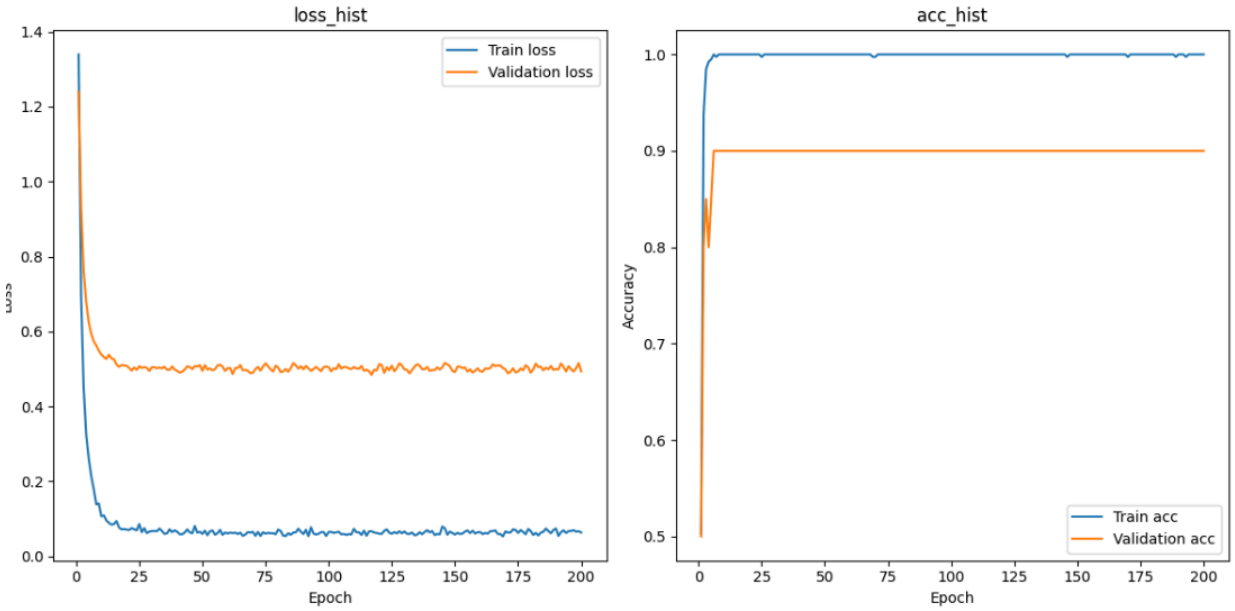

Image 6. Results of ResNet50 best Model with the dataset "Top 5 artists with 100 images"

Similar to ResNet18, overfitting is also observed with ResNet50, not only in this execution but also in those using other optimizers and schedulers. Notably, the model now shows an accuracy of up to 90% on the validation set.

5. 3. VGG-16

The last model we used is VGG-16, which is quite similar to both above in the sense of how we make it perform to classify, but with some own characteristics. The architecture has a 16 in his name refered to 16 layers that have weights, which thirteen are convolutinonal layers and the remaining three hare Dense layers. Instead of using a variety of hyper-parameters, it consistently employs 3x3 convolutional filters with stride 1 and same padding, along with 2x2 max-pooling layers with stride 2.

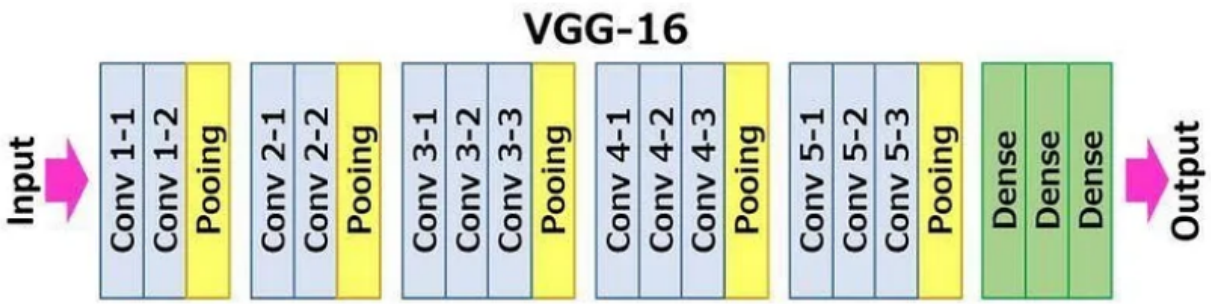

Image 7. VGG-16 Architecture simple schema

We have conducted several execution tests by varying parameters. The results of these tests can be observed at the following file:
Starting_Points/Classifier/VGG16/Vgg16_DifferentExecutions.ipynb.

One of the executuion tests using *Adam* optimizer with *ExponentialLR* scheduler and an initial learning rate equal to 1e-4, the model showed an unusual behavior where the validation loss increased over time. On the other hand, accuracy remained stable over time showing large overfitting.

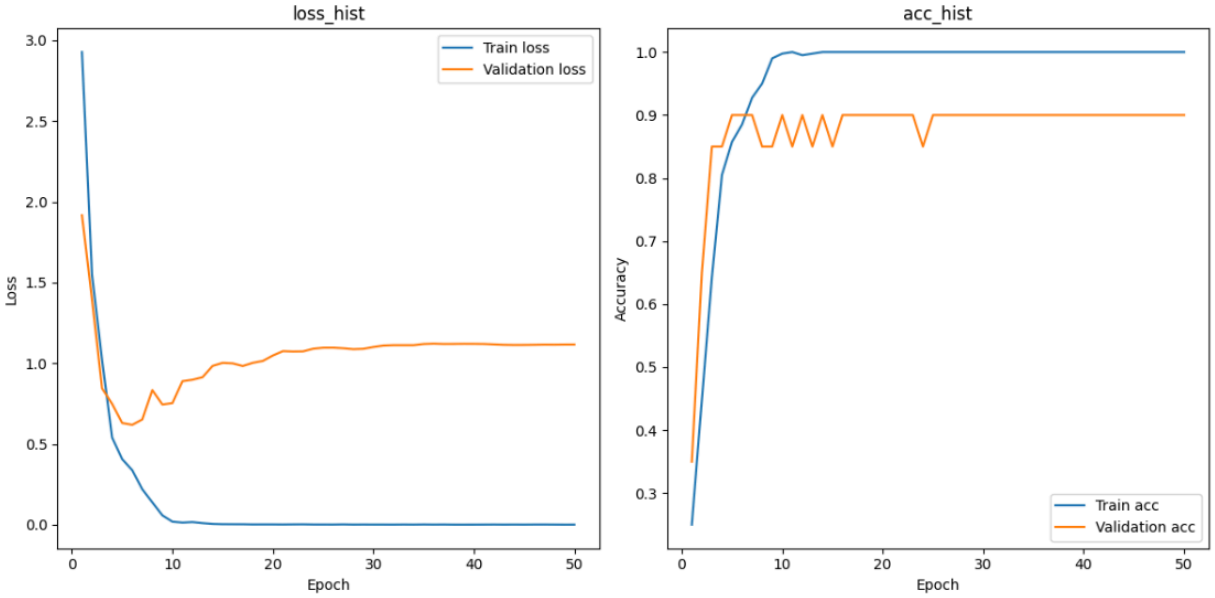

Image 8. Results of Vgg16 best Model 1 with the dataset "Top 5 artists with 100 images"

However, the model keeping *Adam* optimizer and same initial learning rate as above and replacing the

lr_scheduler by a *MultipleLR* showed better performance. This improvement can be seen because the validation loss and accuracy stabilized at good levels, as can be seen in the following image:

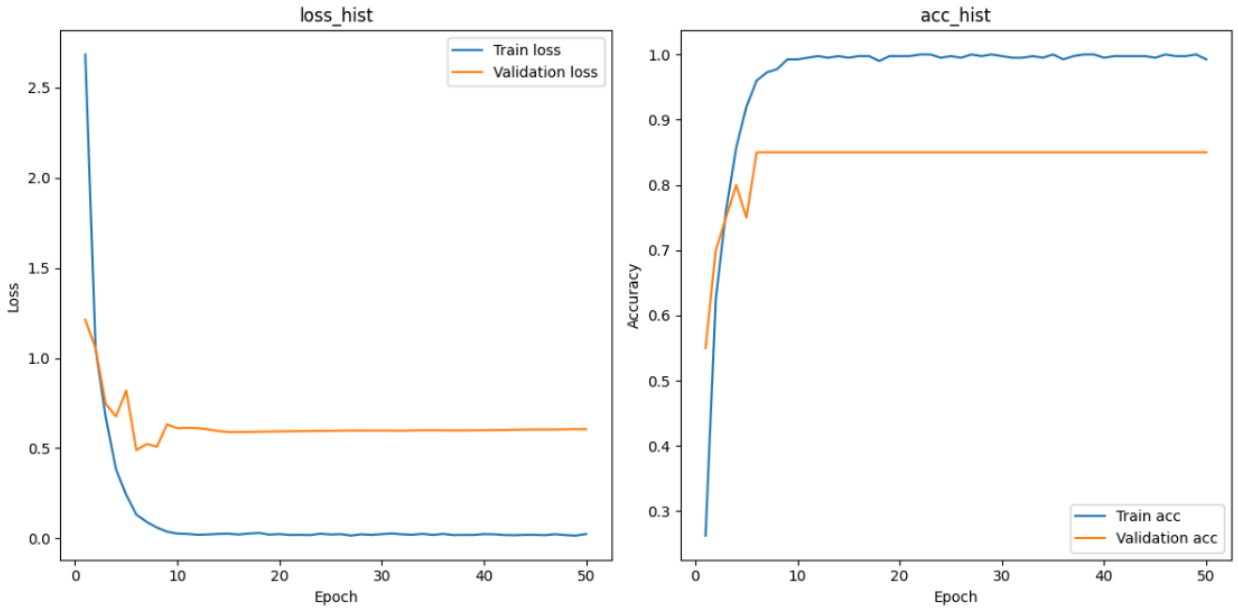

Image 9. Results of Vgg16 best Model 2 with the dataset "Top 5 artists with 100 images"

5. 4. Classifiers comparison

It is clear that all three models exhibit overfitting, with ResNet-18 showing the least. Despite this, ResNet-18 also has the lowest validation accuracy value (85%) compared to the other models. ResNet-50 exhibits the largest difference in loss between training and validation but shows the highest accuracy values. Notably, after epoch 20, no differences are seen in the graphs, indicating that the model no longer learns. VGG-16 appears to be intermediate between the two in terms of accuracy, achieving 90%, but the fact that the loss increased in the first model of this network led us to discard it. The second VGG-16 model performed better but gave a maximum accuracy of 85%, which is lower than that obtained with ResNet-50.

Therefore, considering this comparison, it has been decided that the best model among those tested with the "Top 5 artists with 100 images" dataset is ResNet-50 with a learning rate of *1e-4*, optimizer *AdaGrad*, and scheduler *ExponentialLR*. This model will be used as a base to improve it by applying techniques to prevent overfitting, such as data augmentation, regularization, dropout, or unfreezing layers of the model.

5. 5. Process of reducing overfitting of the best model

Working with the best classification model identified for the dataset of "The Top 5 Artists with 100 Images Each," various techniques were employed with the aim of minimizing the observed overfitting. Through multiple tests, significant reduction in overfitting was achieved using the following strategies:

- Layer Freezing:
  Initially, the entire pretrained ResNet50 neural network is frozen to preserve the general features previously learned. Subsequently, selective thawing of deeper layers (layer3, layer4) along with

the fully connected layer (fc) allows for adaptation to the specific characteristics of the new dataset.

- Dropout:
  A dropout layer with a rate of 50% is introduced into the fully connected layer. This method randomly deactivates half of the neurons during each training iteration, preventing the model from relying too heavily on any single neuron and enhancing its ability to generalize.

- Data Augmentation:
  During training, random transformations are applied to the images, such as random crops, horizontal flips, and color adjustments. This technique generates different versions of the training images, aiding the model in better understanding image variations and improving its generalization capability.

- Weight Decay:
  The Adagrad optimizer is configured with a weight decay parameter. This approach penalizes large weights during training, reducing the model's complexity and mitigating the risk of overfitting.

The results obtained applying these techniques with the dataset of "The Top 5 Artists with 100 Images" have been as follows:

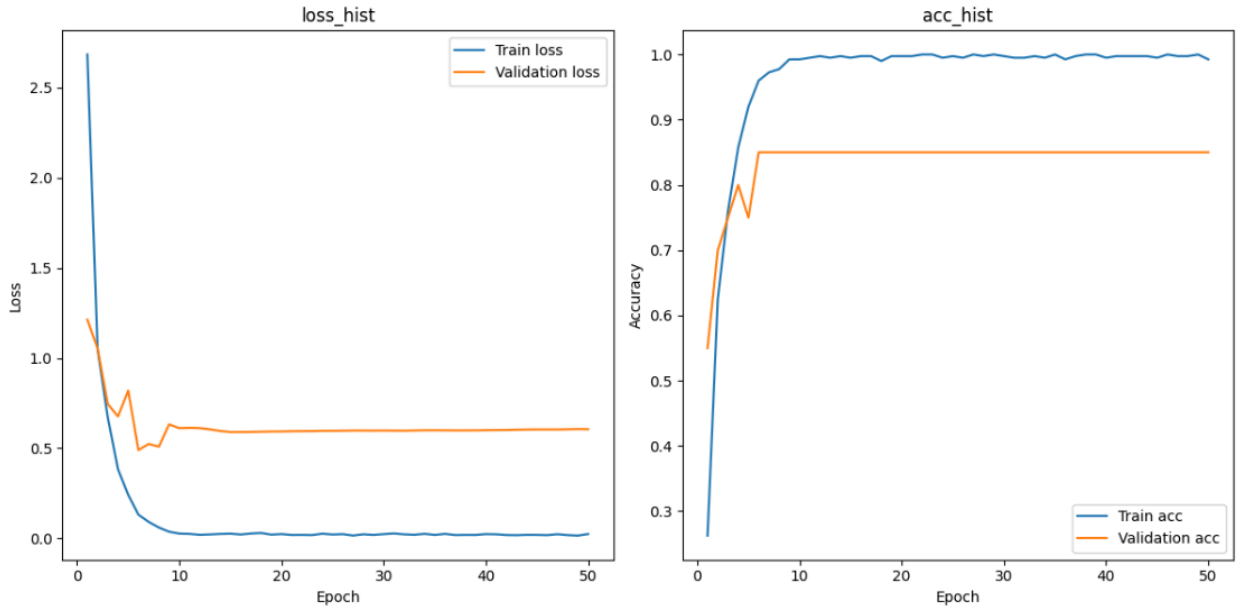

Image 10. Results of the Best Model (ResNet50) Reducing Overfitting with the Dataset "Top 5 artists with 100 images"

The next step is to test this same model with more artists.

5. 6. Test the best model found with more data

Once we've found the best model for classifying images by artist and reduced its overfitting to the maximum extent possible, the next task is to test this same model with a larger volume of data. For this

purpose, we have created the dataset "TOP100_ARTISTS_WITH_ALL_PICTURES," which contains **28.865** images corresponding to the 100 artists with the most images within the entire dataset.

Now, if we feed this dataset into our artist classification network, we see that the results are extremely poor:

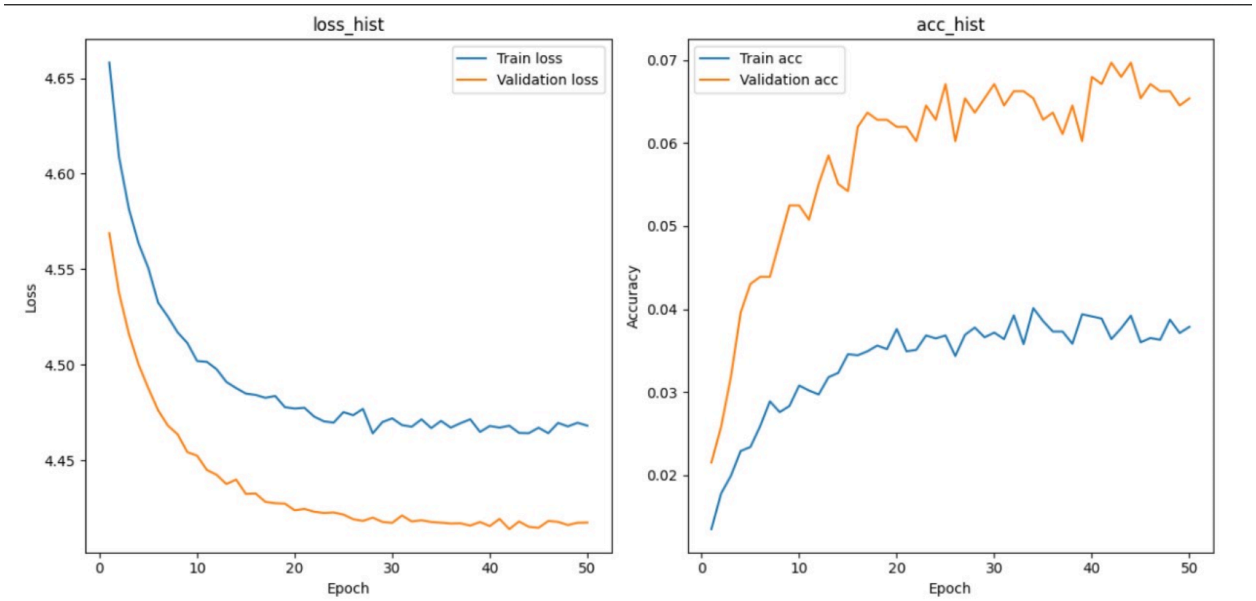

Image 11. Results of the Best Model (ResNet50) Reducing Overfitting with the Dataset "Top 100 artists with ALL images"

The reason why the model performed so poorly when executed with the "Top 100 artists with ALL images" dataset is because it was initially trained with only 500 images (from the "Top 5 artists with 100 images" set). Now, with the use of this new dataset containing more artists, the volume of images has increased by **57.73x** times. This significantly outpaces the capabilities of the previously constructed model, which was too simplistic for the current task. To address this issue, we need to experiment with more complex classification models, as DenseNet networks. Below are some of the results from executions and tests carried out with the new "Top 100 artists with ALL images" dataset using these advanced network models.

The architecture we found to work best for our problem was DenseNet169, in addition to the other configurations in the model that include finetuning or dropout.
The first results we obtained with this architecture were through unfreezing the last layer of the network so that it would only train with these weights. In this last layer we also applied a dropout of 0.5 to make the model more general. And on the other hand we applied a dynamic learning rate in which every 3 epochs if the validation loss did not improve we reduced the lr, which is known as reduce on plateau.

In this way, we have obtained the following results by training for 200 epochs:

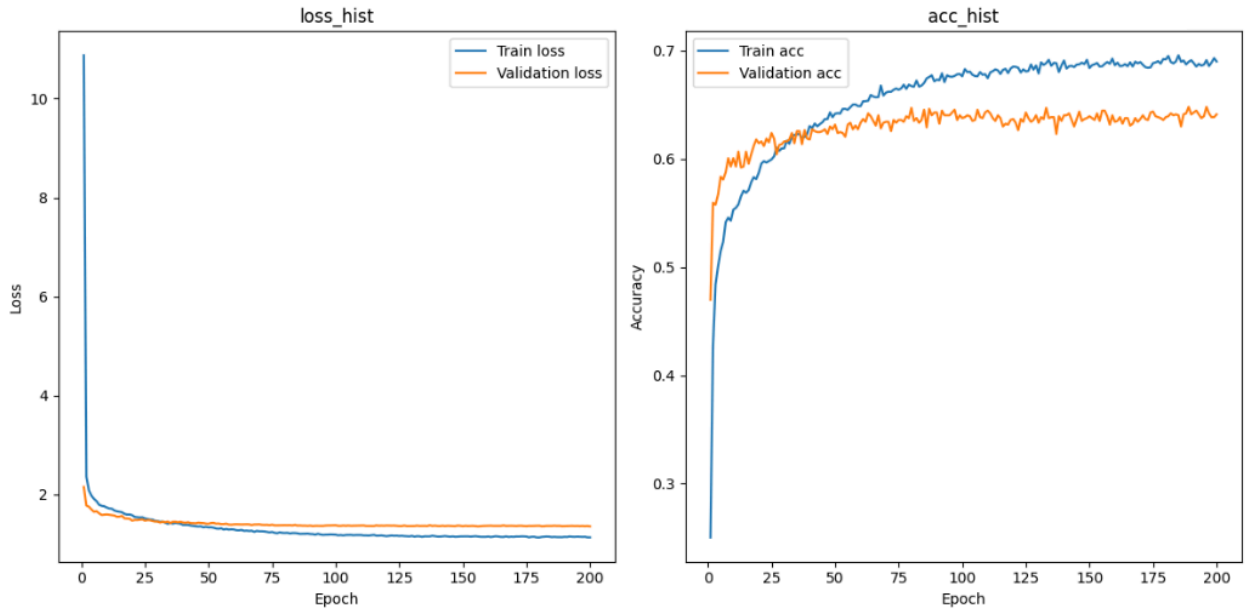

Image 12. Results of the DenseNet169 architecture unfreezing the last layer with the Dataset "Top 100 artists with ALL images"

As we can see we don't obtain the best results in terms of accuracy but we get a model without overfitting with 100 artists dataset. We also see that probably we need to train more weights to reach a better accuracy in the train dataframe, so the next execution we have done is the same as we done but this time with the last dense layer unfreezed (layer4) so the model could train more weights and adapt more to the training dataset.

The results we get are the next ones:

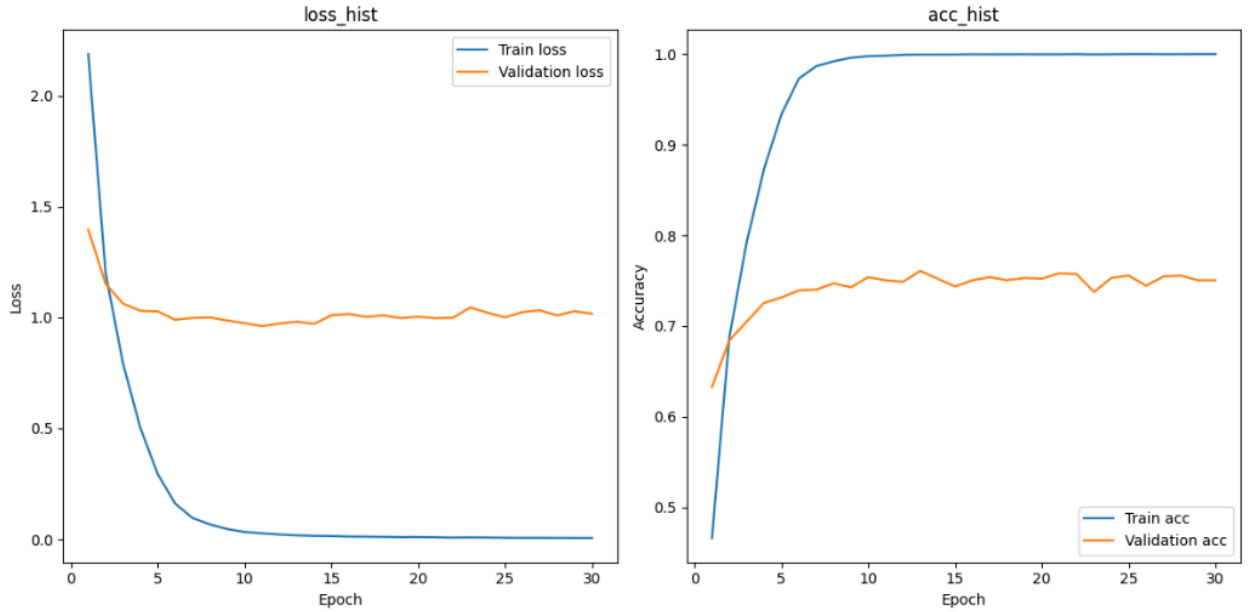

Image 13. Results of the DenseNet169 architecture unfreezing the connected and the last dense layers

with the Dataset "Top 100 artists with ALL images"

If we analyse the results, we get what we expect. Now we have a top accuracy with the train and the test accuracy also improved but it is true that now it has appeared some overfitting.

We tried with a short number of epochs, many other architectures and configurations, but this one is the best we get in general terms, so with this one, and the one without overfitting, we saved the weights that give the results we obtained. Testing both networks, we found that the one with the best results had 74% accuracy, even though there was a bit of overfitting. Plus, at the end of the training, we wanted to see which artists in the validation set the network struggled with the most, meaning it showed lower accuracy for them. Here are those artists for each network.

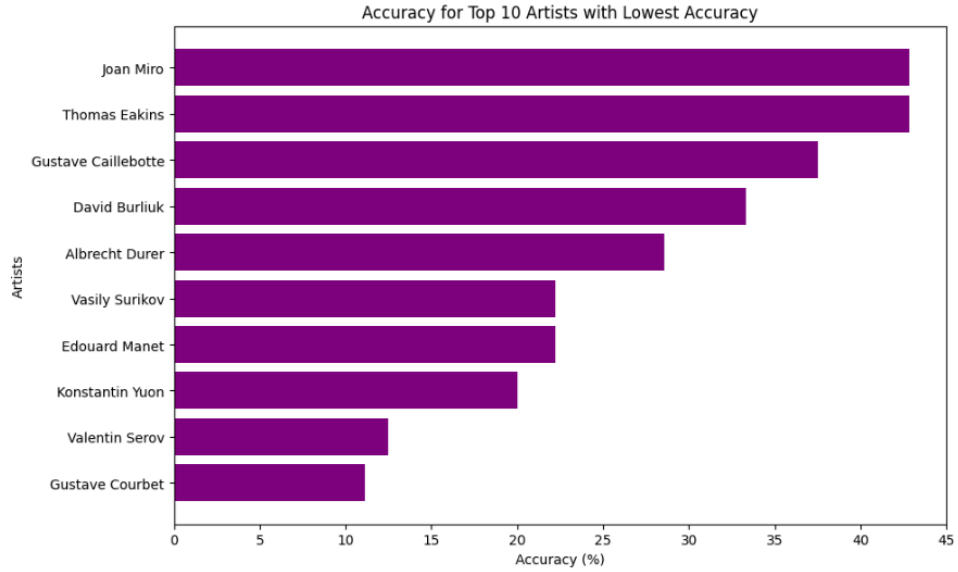

Image 14. Top 10 artists from the validation test set of the "Top 100 artists with ALL images" dataset with the lowest accuracy from the network without overfitting (65% accuracy)

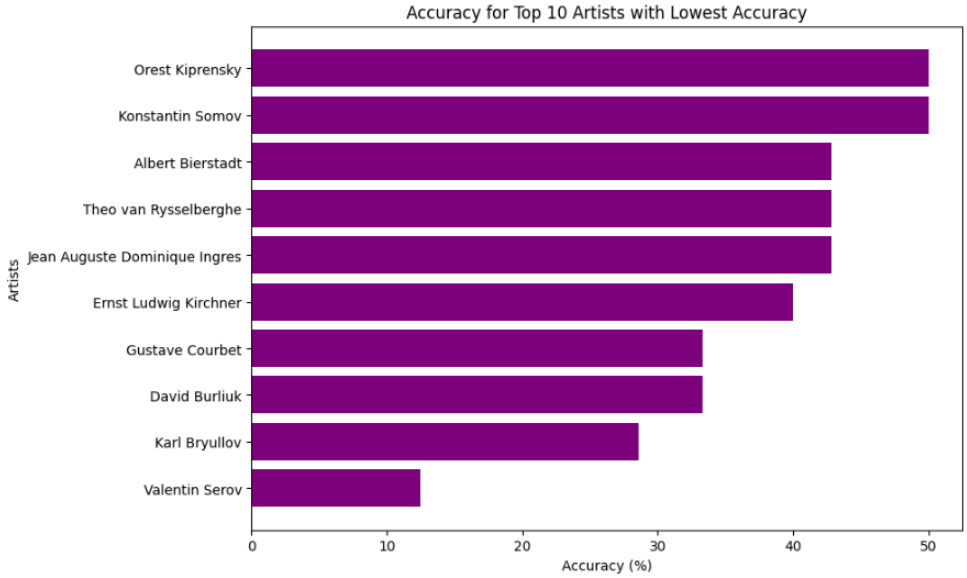

Image 15. Top 10 artists from the validation test set of the "Top 100 artists with ALL images" dataset

with the lowest accuracy from the network with a bit of overfitting (74% accuracy)

Comparing images 14 and 15, we can say that the network with 74% accuracy, despite having a bit of overfitting, has better accuracy per artist. The top 10 artists with the worst accuracy have a higher accuracy compared to the other network.

5. 7. Pipeline predictions

After training the model, it is now time to generate a pipeline that allows us to pass images to the model and classify them according to the weights learned from the last model we have seen (74% accuracy).

It's true that a more realistic scenario would be to feed the network a set of images where the artist is unknown and get the artist for each piece of art. For testing our pipeline, we didn't do this because our goal is, once the artist is classified, to sort their images chronologically. Since the network doesn't have 100% accuracy, there will always be some margin of error, meaning it won't always get the artist right. So, if we fed our network a set of images to be classified by artist, then grouped and sorted them chronologically by artist, it wouldn't be accurate. We'd end up with images assigned to the wrong artist, and those would be used in the sorting, leading to an incorrect final order.

That's why our tests were done with images from the same artist. Each image was classified and the artist assigned to this set was the one that appeared the most. Once the artist is identified, the same set of images will be used for sorting.

Below is an image of the pipeline so far, showing only the artist classifier part.

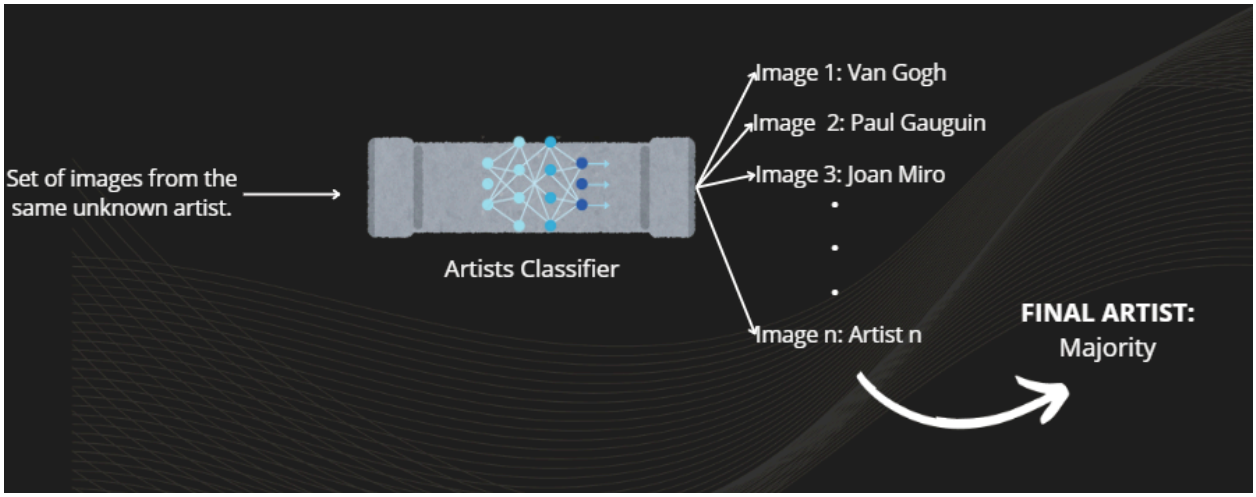

Image 16. Pipeline so far

To test that pipeline, what we have done is to choose 3 painters, which are the top 1 with the highest number of paintings, the top 50 and the top 100. And for each of them we have generated 2 datasets of images, one with the training images and another with the test images (images that the model has not seen for training).

That is to say, we have passed these 6 groups of images through the pipeline and the results obtained are shown below:

TOP 1 Paul Gauguin:

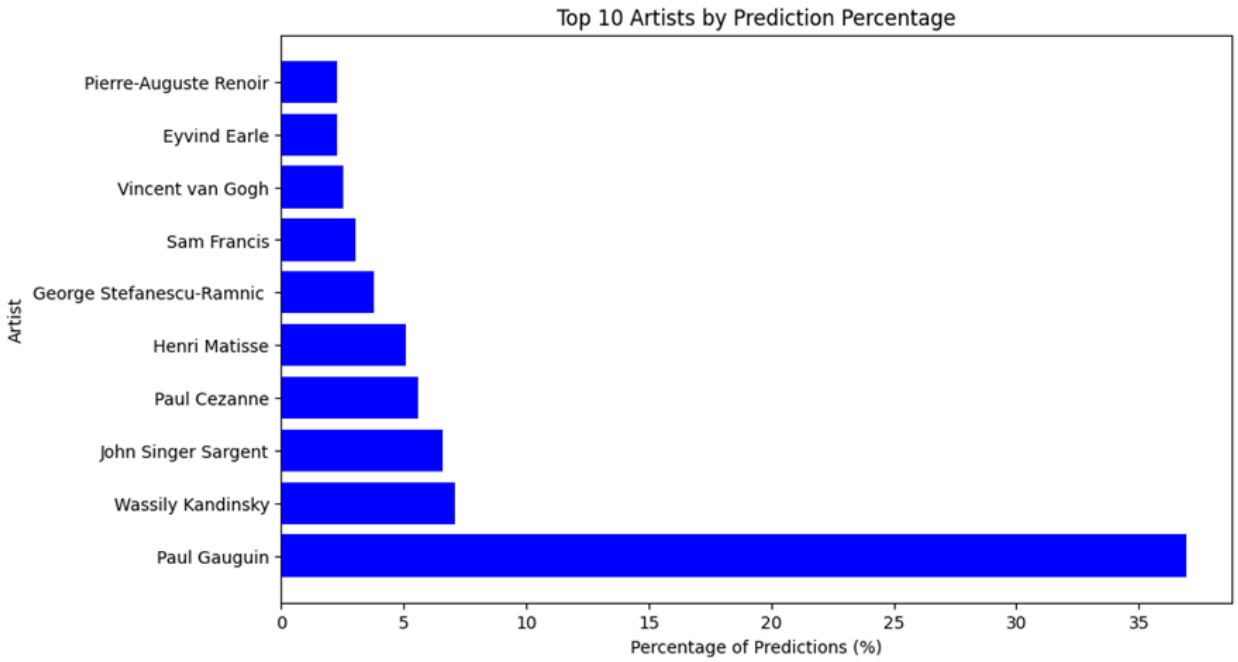

Image 17. Results of the predictions of Paul Gauguin train dataset (396 images)

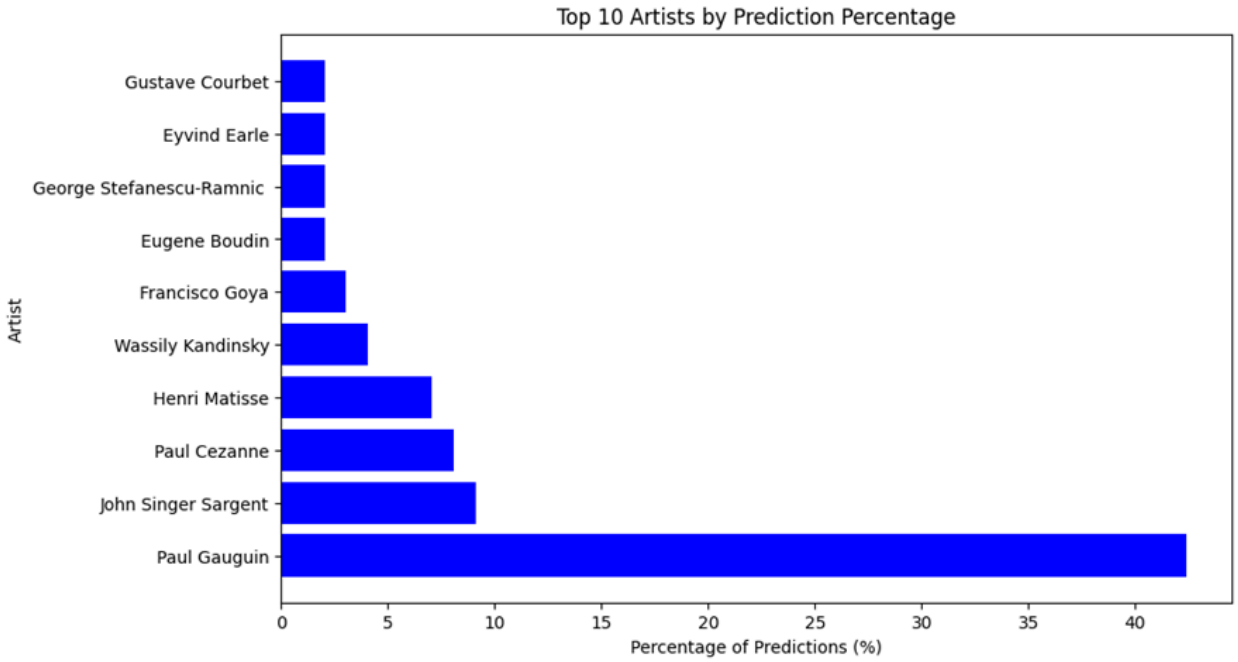

Image 18. Results of the predictions of Paul Gauguin test dataset (100 images)

TOP 50 Raphael Kirchner:

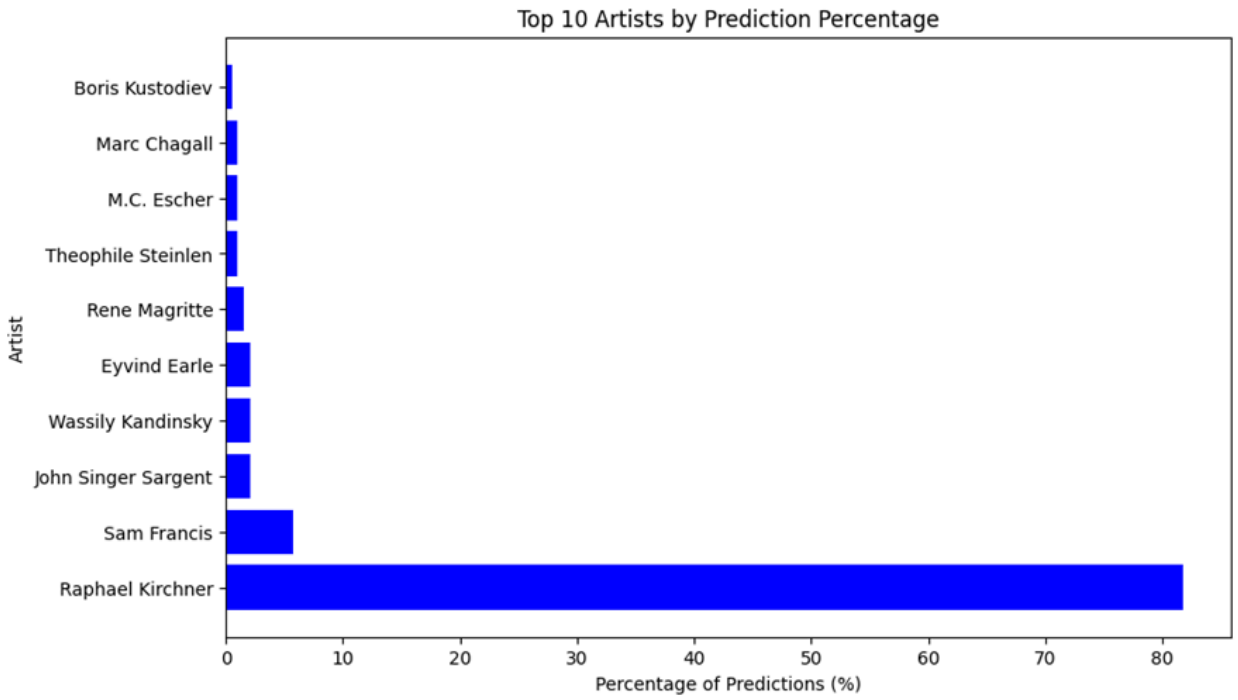

Image 19. Results of the predictions of Raphael Kirchner train dataset (193 images)

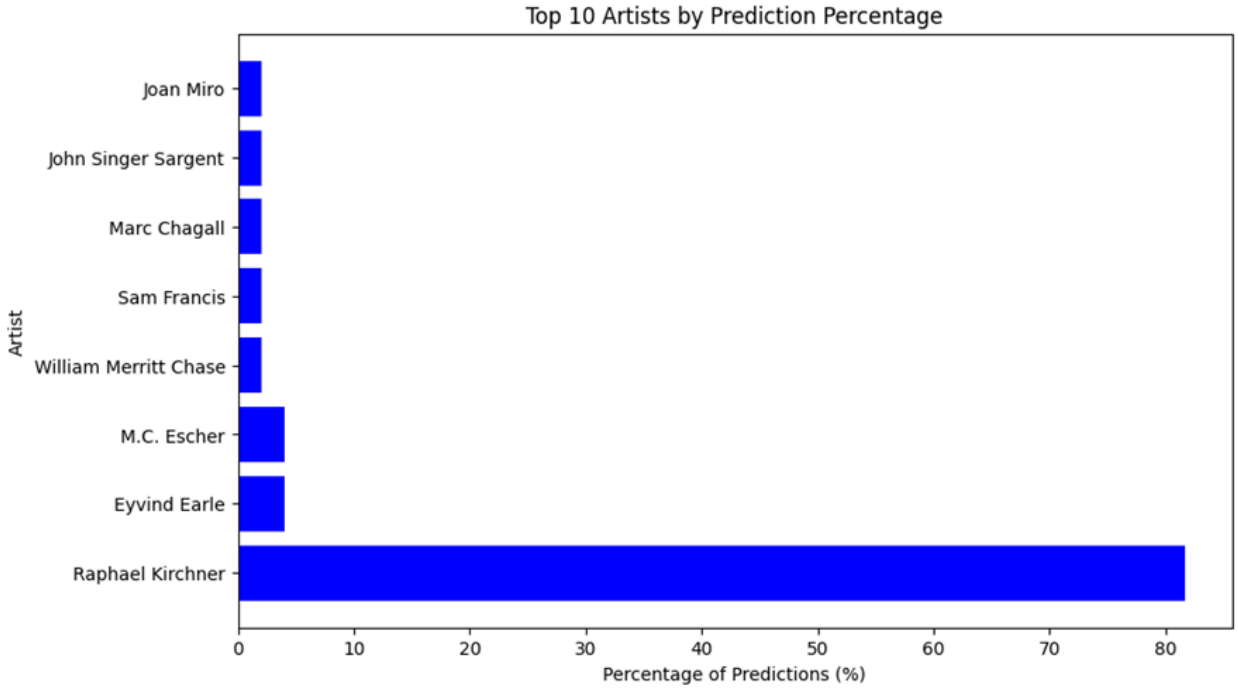

Image 20. Results of the predictions of Raphael Kirchner test dataset (50 images)

TOP 100 Dante Gabriel Rossetti:

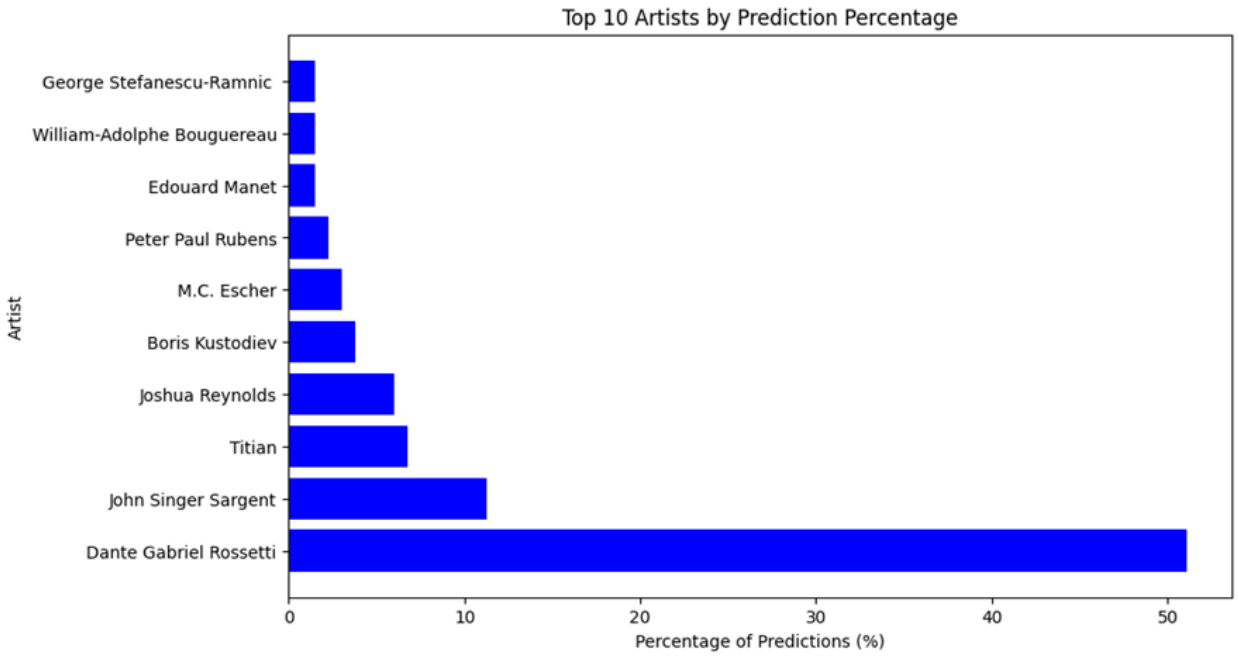

Image 21. Results of the predictions of Dante Gabriel Rossetti train dataset (134 images)

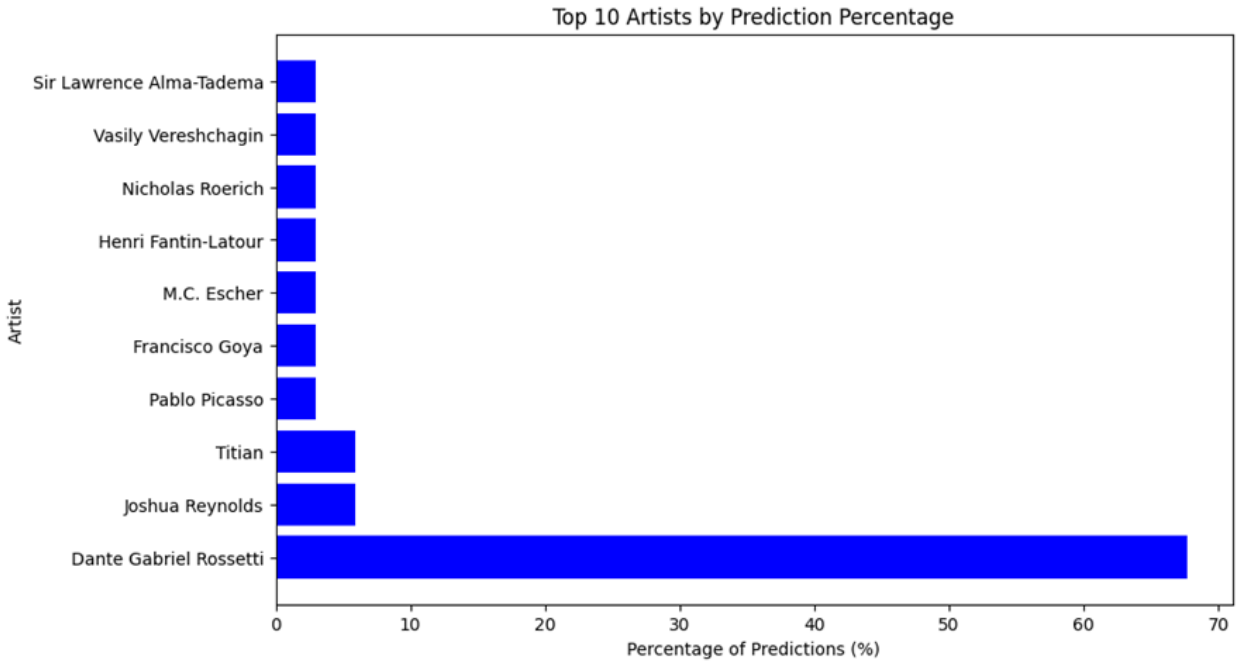

Image 22. Results of the predictions of Dante Gabriel Rossetti test dataset (35 images)

To start analysing the results, we can realise something that may seem a little strange, but it makes a lot of sense.

What we see is that although we have more images of an artist does not mean that we get a higher percentage of correct predictions, although it will obviously have a relationship, but it is not everything.

It is necessary to think that each artist will have a style and there will be those who have a very defined style with similar characteristics between paintings and there will be others who will paint in different ways in their paintings with what this can make the predictions difficult. This is why we see that Raphael Kirchner, for example, despite having fewer images than Paul Gauguin, has a higher percentage of correct predictions.

5. Date Sorter

Once we've tackled the first task on artist classification, the next task on the list is to carry out chronological sorting. To do this, we've wanted to implement 2 types of solutions to compare and draw conclusions. These have been:

- Date Classifier
- Metric Learning

6. 1. Date Classifier

Before jumping into explaining the date classifier, it's crucial to grasp the concept of classification. The implemented network will receive pairs of images to which it can assign 2 labels, 0 or 1 (binary classification). 1 will indicate that the date of the first image is greater than that of the second and 0 if it's less than or equal. For example, the pair (2001, 1990) will be labeled as 1. With this classification, subsequently, an ordering algorithm will be required to achieve the final chronological order.

$$\begin{cases} date_{img1} > date_{img2} & then \quad 1 \\ date_{img1} \leq date_{img2} & then \quad 0 \end{cases}$$

Image 23. Classification function

6. 1. 1. Top 5 Artists with 100

First off, we experimented with different networks and parameters to build the network just like we did with the artists classifier. Additionally, to set up this initial network, we were working with the TOP 5 Artists with 100 Images dataset.

The first model worth noting was a Vgg16, which showed significant overfitting, as can be seen in the validation loss curve. The network had a test accuracy of 57%.:

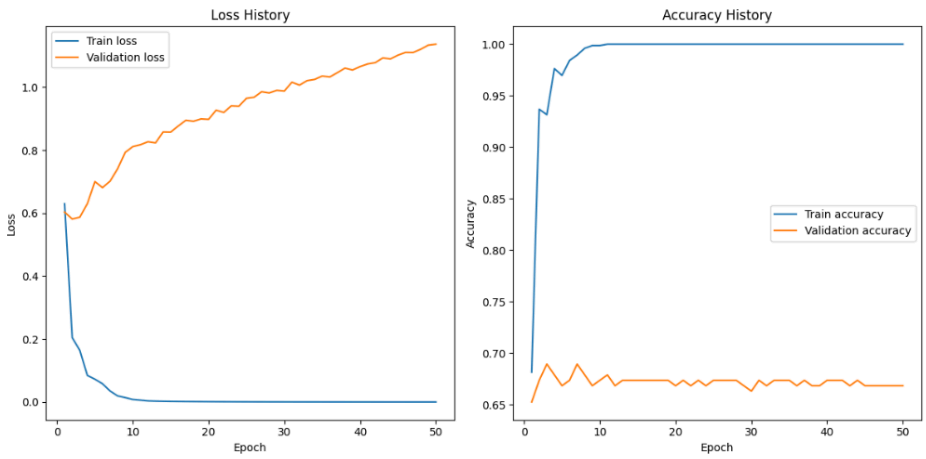

Image 24. Results Vgg16 date classifier first model

To reduce overfitting, various techniques were tested, such as fine-tuning (progressively unfreezing layers), in addition to applying different regularizations and dropouts. Ultimately, the best model achieved a test accuracy of 66% By using a combination of unfreezing the last four layers, L2 regularization, applying dropout of 60% after the eighth and tenth layer, and early stopping with 10 of patience.

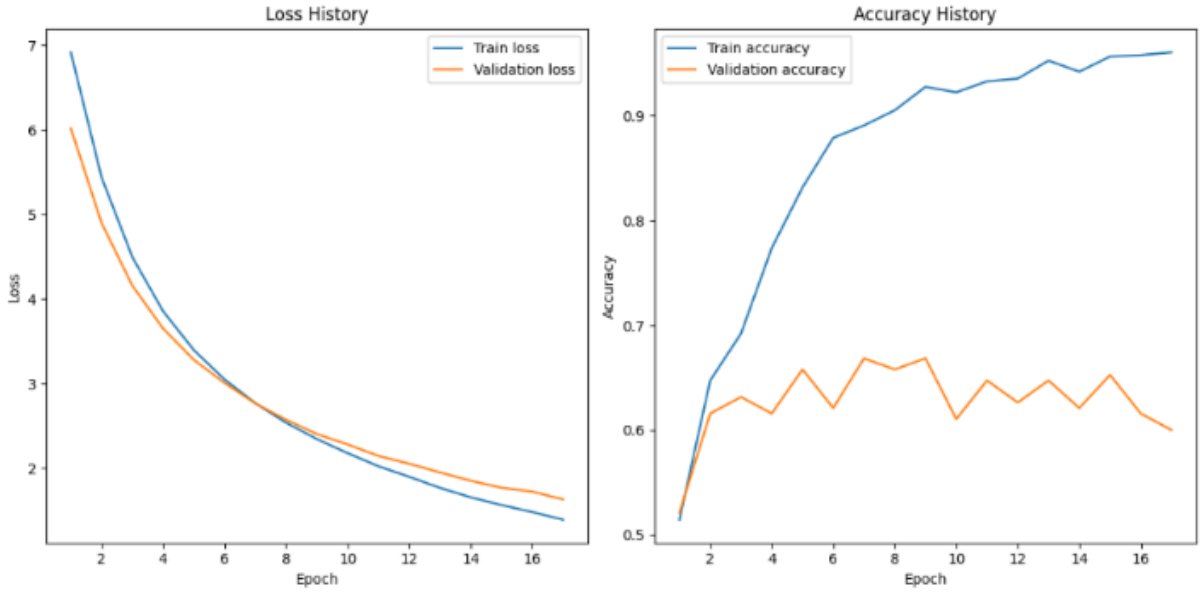

Image 25. Results Vgg16 date classifier with top 5 artists Unfreezing last 4 layers + regularitzation l2 + dropout + early stopping

Also, ResNet50 was tried, but no execution was achieved that improved the previous one. About this execution (the best one, which gives 66% on the test), we can see how even though the loss is very similar in both train and test, there's a clear separation in the accuracy lines of train and test. This could mean two things:

```
- The network is too complex for the volume of data we have.
- The network has been trained with too few data.
```

Therefore, since these results are about the TOP5 artists, we are going to test it with the dataset of the TOP100 artists to see which of the two mentioned scenarios is actually happening.

6. 1. 2. Top 100 Artists with all images

Before running the network with the dataset of the TOP 100 artists, we wanted to check the distribution of dates and the date differences for each pair between the train and test sets. This way, we ensure that the model is being trained on aspects that will be evaluated later on.

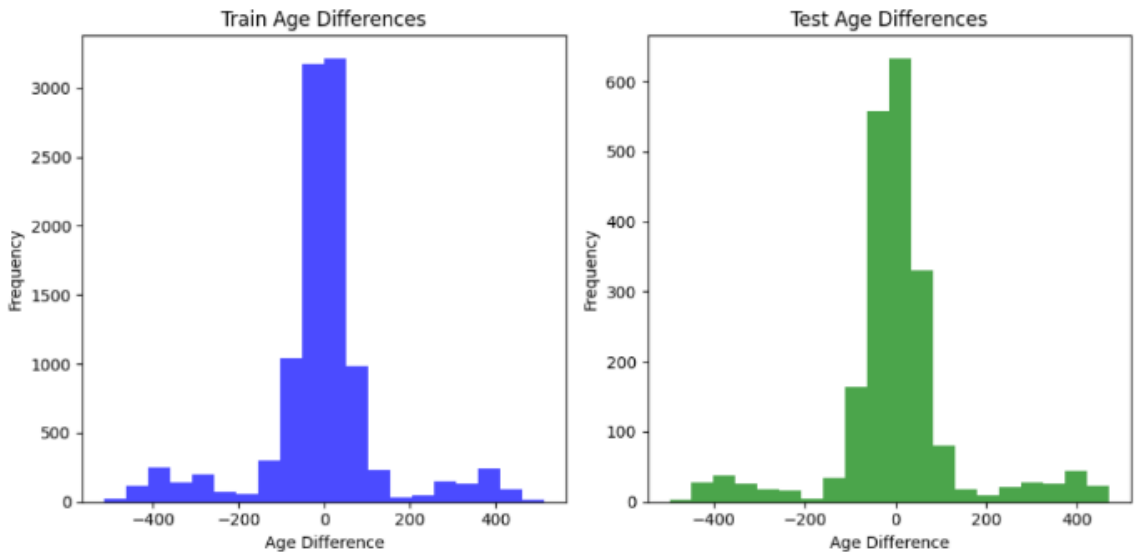

Image 26. Age difference histogram

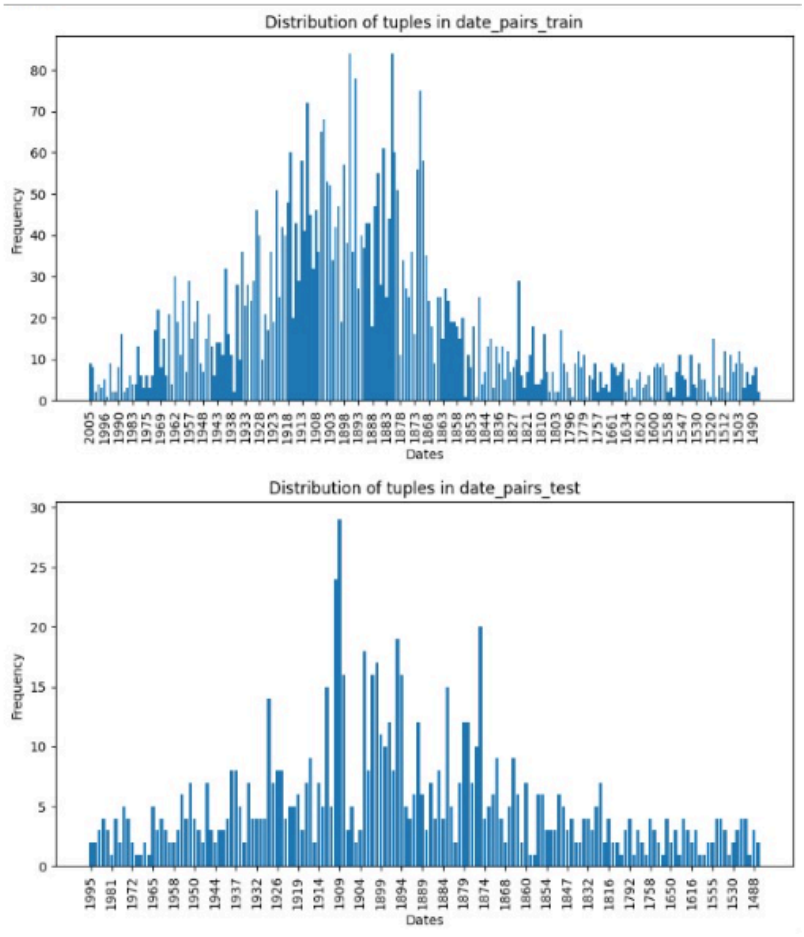

Image 27. Age pairs histogram

As we can see in images 26 and 27, the difference in dates between train and test for the pairs is practically identical. Additionally, both sets have the most frequent dates in common.

Now that we know the distribution, we proceed to train the model with the Top 100 artists dataset. Here are the results:

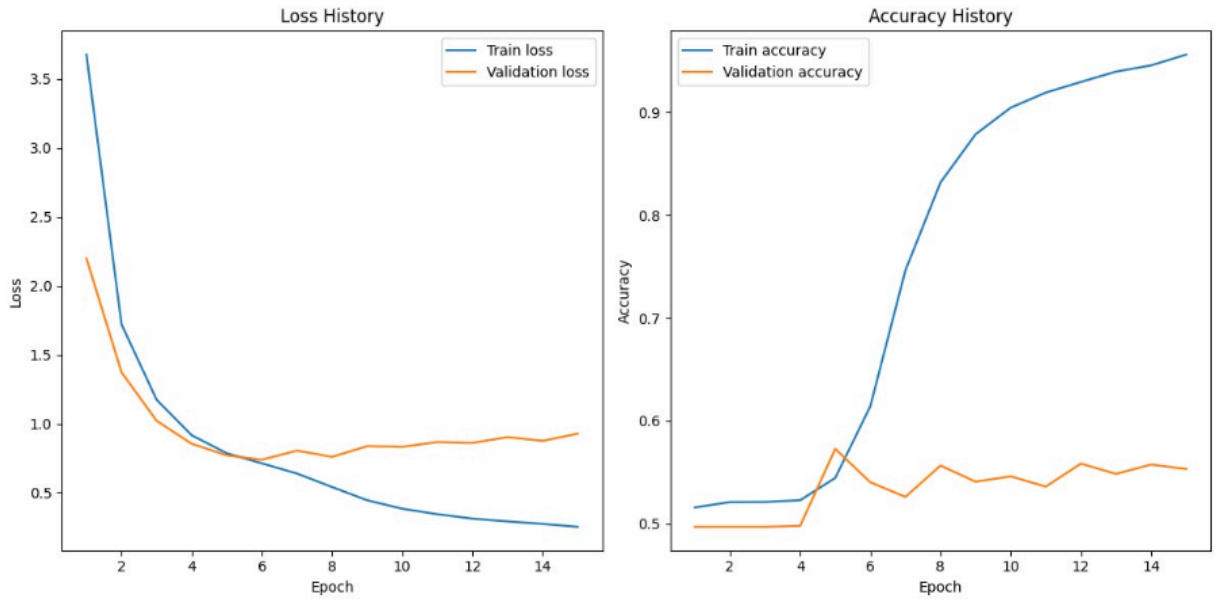

Image 28. Results Vgg16 date classifier with top 100 artists Unfreezing last 4 layers + regularitzation l2 + dropout + early stopping

As we can see in the results shown in figure 28, when launching the network for the Top 100 artists dataset, the model didn't improve. This led us to think that it could be because the network is unable to recognize the different styles followed by the artists present in this set. The pairs of images in Train and Test have been built with images from the same artist. The network fails to recognize the individual style of each artist during evaluation. For example, the network does not detect that an artist painted a piece in 2024 in a 1940s style, treating it incorrectly as from 1940 when it is actually from 2024.

To verify this, the same network was executed but with pairs constructed without grouping by artist; that is, the pairs were formed randomly between images of different artists:

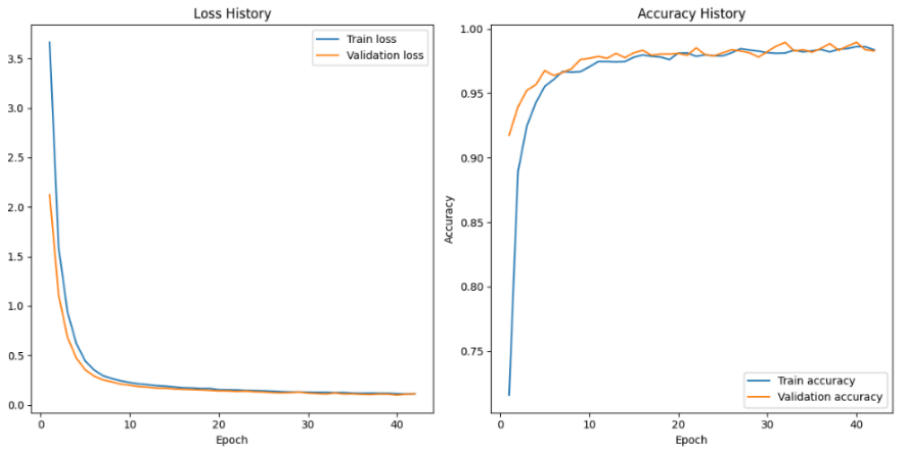

Image 29. esults Vgg16 *diferent artist pairs* date classifier with top 100 artists Unfreezing last 4 layers + regularitzation l2 + dropout + early stopping

The training result was better, but the test, which continued to be constructed with pairs of images from the same artist (since that's how the network will be used in the future), yielded an accuracy of 62%. This gives us the understanding that indeed the network is unable to detect the style of each artist because when making pairs without grouping by artists, the accuracy of the validation set reached almost 100%, which was not the case in the test set.

Seeing that the network still struggled to detect the styles of each artist, we thought of a new possible solution: training a network with images from a single artist. This means creating a specific network for a particular artist. In our project, considering that sorting happens after artist detection, this seems to be a good practice, as we already know the artist during the sorting stage. However, if this solution were implemented, it would be necessary to build a date classification network for each of the artists present in our dataset, which would mean building a total of 100 different networks. This solution is not the most optimal, but despite this, we wanted to run a test by creating a specific network for an artist, taking **Vincent Van Gogh** as an example.

6. 1. 3. Specific network for Vincent Van Gogh

The same model we've been using so far has been trained and tested with specific images of Van Gogh (383 images for train, 103 for test).

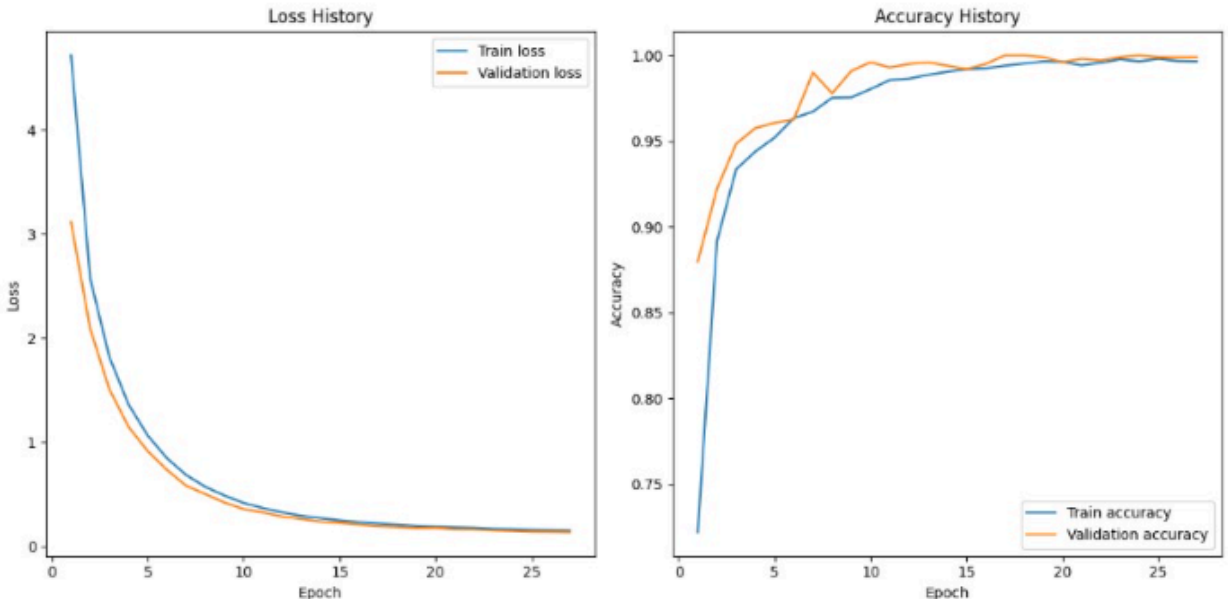

Image 30. Predicction metrics with Vincent Van Gogh

The result achieved in training was very successful, as it showed an accuracy of almost 100%. Despite this, the training accuracy remained comparatively low, with a value of 66.67%. Nevertheless, it has been the best accuracy obtained in the test when compared to the two previous models trained with all the artists (creating pairs randomly or by artists).

6. 4. Pipeline

So far, three different models have been trained:
- **Model 1:** With all the artists, creating pairs of images by artist
- **Model 2:** With all the artists, creating pairs of images randomly
- **Model 3:** Specific model for the artist (in our case example for Van Gogh)

The weights of these three models have been saved and added to the pipeline built in the artist classifier. Therefore, in the new pipeline, once the artist is classified, we have two options: sort the images using one of the two models trained with all the artists or sort the images based on the specific model of the classified artist.

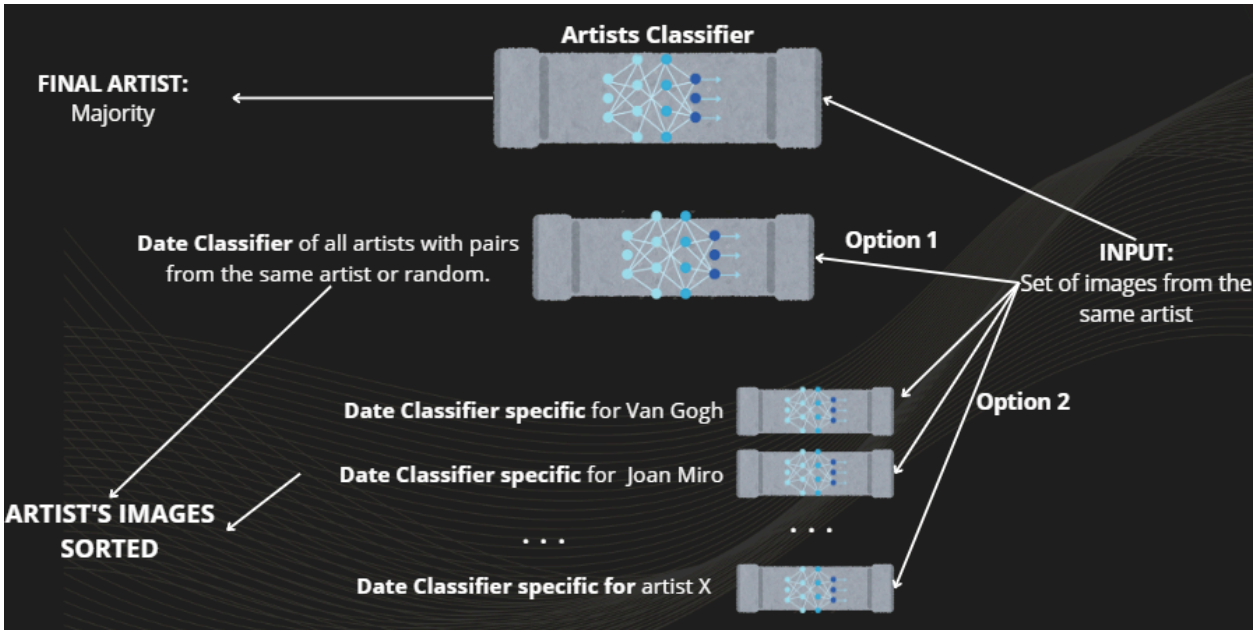

Image 31. Pipeline so far

With the pipeline built, we moved on to run different executions to understand the results of each of the 3 models so we could compare them. To do this, executions were carried out for each model on 3 different datasets. They are as follows:
- VanGogh_4_Images_TEST
- VanGogh_4_Images_TRAIN
- Vincent_van_Gogh_20_IMAGES (random images)

Highlight that to select 4 images for Train and Test sets, we picked 4 common images from the different Train and Test sets used in the 3 models, since the Train and Test sets are different for each model employed. All the images used were from the artists Vincent Van Gogh, as the specific network trained with images from a single artist was about this artist. These 3 test sets will allow us to see how each model performs against images used in training and entirely new images.

For sorting, we used an algorithm that pairs each image once with the rest of the images, without repeated pairs, and makes predictions for each pair. For each prediction, it accumulates the value of the prediction (between 0 and 1) for the image that the model predicts is older for each pair processed. That the image is older means that the date is the smaller.

During the process of making predictions and accumulating the value for the older image, the result obtained for each pair has been saved. This way, the predicted result has been compared with the real ones, and the corresponding accuracy has been calculated for the total pairs of each set with each model. The following table shows this result.

	VanGogh_4_Imatges_TEST	VanGogh_4_Imatges_TRAIN	20 Imatges Random
Number of pairs	6	6	190
Random pairs	83.33%	83.33%	61.58%

Specific Network for Van Gogh	83.33%	100%	80%
Pairs from the same artist	50%	16.67%	38.42%

We can see how the network that gives the best classifications is the one trained exclusively with images of Van Gogh **Specific Network for Van Gogh**, which leads us to understand that building a network for each of the artists, despite being an inefficient solution, would be the way to get the best results.

Below are the results of the obtained orderings with each model tested on the different test sets.

Image 32. Results for each model on "VanGogh_4_Imatges_TEST" set

Image 33. Results for each model on "VanGogh_4_Imatges_TRAIN" set

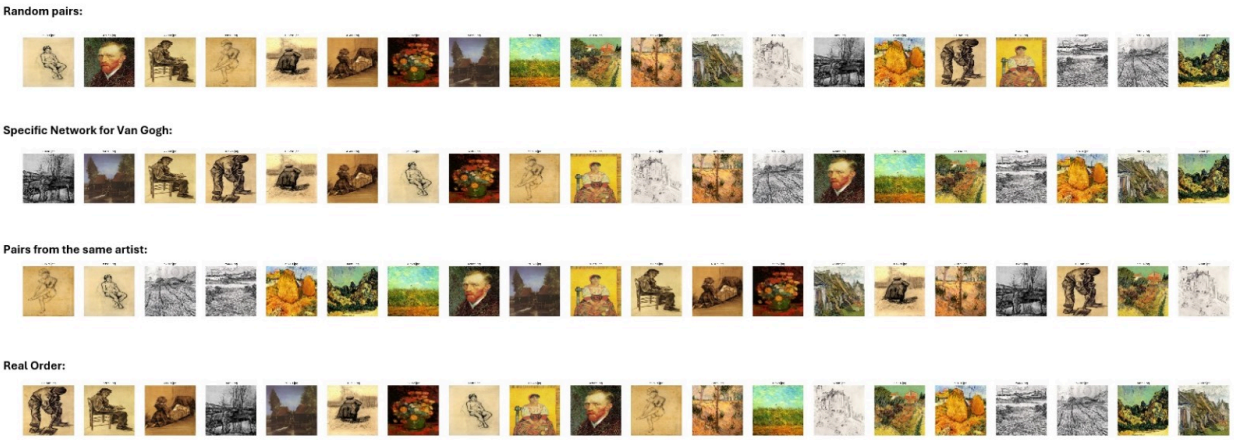

Image 34. Results for each model on "20 Imatges Random" set

Comparing the images, we see that even though the accuracy results on the pairs of each model are high, the result of the final ordering can still be erroneous. This is because if the network predicts a pair incorrectly, both images in this pair will already be incorrectly ordered, resulting in an incorrect final ordering. This is exemplified by the specific model of Van Gogh on the **20 Random Images** set, where, in the accuracy table, we can see how the value is 80%, but the final ordering result is highly incorrect.

6. Metric Learning

In addition to chronologically ordering through the binary classification explained above, we also aimed to achieve this ordering using metric learning techniques. These techniques should theoretically allow us to achieve our goal by clustering each date (different year) in a metric space. Thus, the way we implemented all versions of our metric learning is through the PyTorch library (*PyTorch Metric Learning*). From the beginning, we decided to base our metric learning models and their training on the creation of triplets, which basically consist of sets of 3 images that are passed consecutively to each batch. These triplets are formed by: an anchor image with a specific date (e.g., 1980), a positive image that is very close to the anchor (e.g., 1981), and finally, a negative image, which is quite distant from the anchor (e.g., 1995).

7. 1. TOP_15_DATES & Random Batchs

In our initial versions, what we did was create triplets of all the images with all the images without customizing the batches in any way, and therefore, without customizing the training itself. The objective of this was to see what our starting point with metric learning was since we were not entirely sure how it would work. It is important to note that these initial tests were conducted on the TOP_15_DATES dataset.

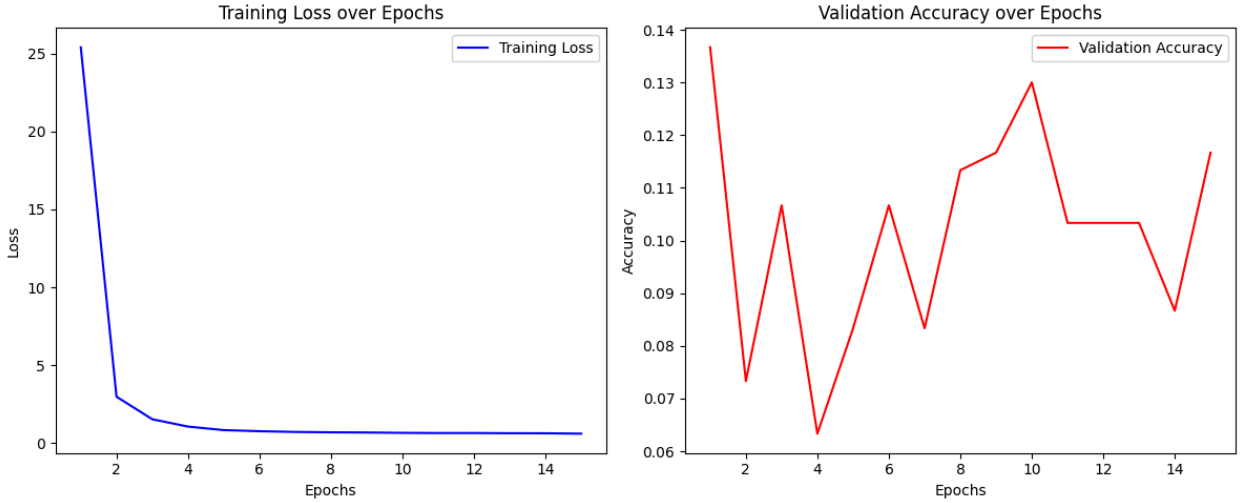

Image 35. Results for metric learning initial model

As can be seen in the image, the validation accuracy results were very poor, as it was not able to surpass 14% accuracy at any point throughout the epochs. From this, we concluded that, just as happened with the date classifier, the network was not able to detect the artists' styles when it received any triplet from different artists in each batch. This is why we decided to try customizing the training in such a way that each batch contained paintings from only one artist at a time. At this point, we also decided to use the

dataset with images only of Van Gogh, as it was again the most logical choice when considering how the image sets would be fed to this metric learning model.

7. 2. Van Gogh Images & Customized Batchs

With all this in mind and with the intention of testing a network based on a ResNet50 with pre-trained weights, we launched the execution of the metric learning with customized batches from a single artist. The results were as follows:

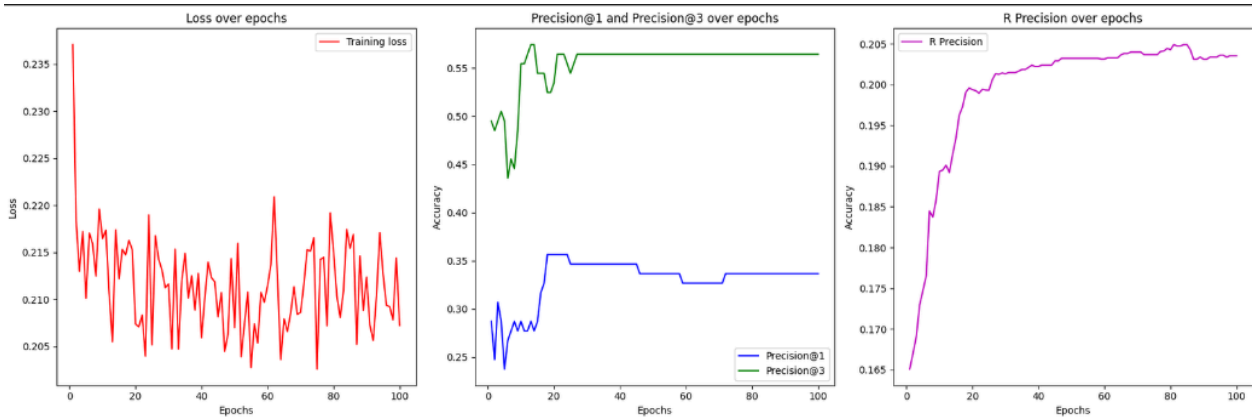

Image 36. Results for metric learning with customized batchs

From these graphs, we can extract three main ideas: the first is that the model still performs very poorly. This can be observed both in the loss, which instead of being controlled and decreasing, has very distinct peaks both high and low, and in the precision_at_1 and r_precision, which do not exceed 0.40 and 0.2 respectively. On the other hand, a positive takeaway from this is that compared to the previous test, there has been a significant improvement in the model's performance, indicating that we believe customized batches are key to further improvement.

7. 3. Clustering exploration via embeddings visualization

Even with all this, we still did not really understand what was happening internally during the training that could provide a coherent reason for the poor loss and resulting metrics. It was then that we decided to try visualizing how the embeddings of images from different dates group together and form (or do not form) clusters. Thus, we decided to plot the embeddings in 2D using the UMAP tool every 50 epochs for both the training and test sets, identifying each label (date in years) with a color scale.

The result for the last epoch of training and testing was as follows:

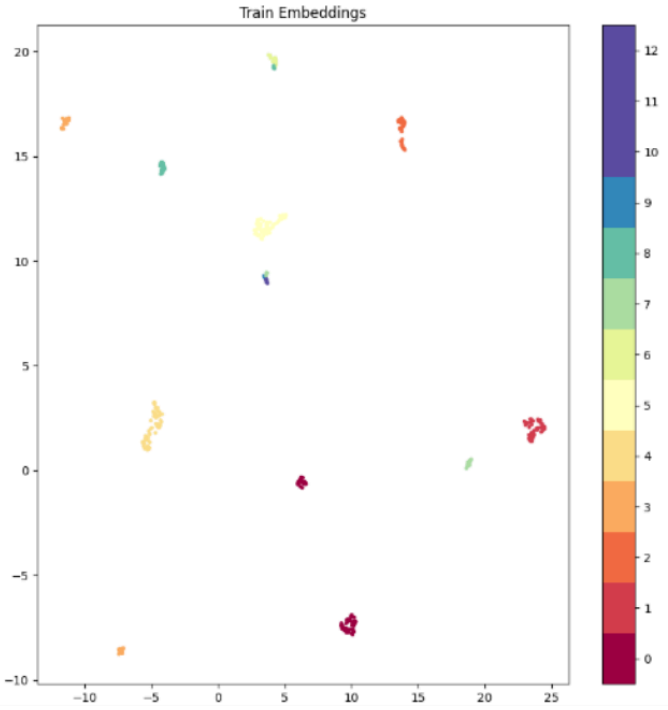

Image 37. Plot of the embeddings in the last epoch of training

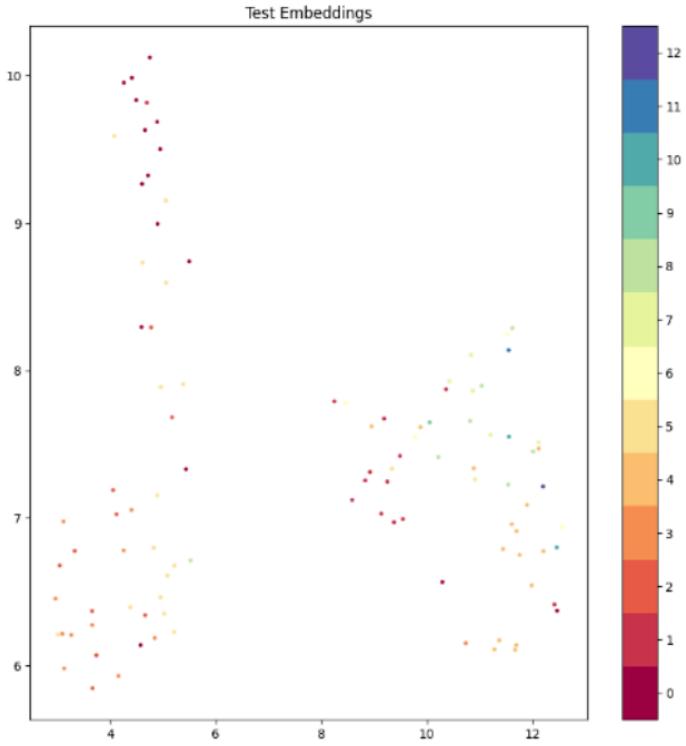

Image 38. Plot of the embeddings in the last epoch of test

As can be seen from these two images, in the training section of the model, the clusters are forming remarkably well. It appears that there are almost the same number of clusters as there are different classes in Van Gogh's paintings. However, the opposite is true in the test section, where no clear clusters can be observed. The information provided by this comparison is very valuable, and it allows us to determine that up to that point, the overfitting of our network was very high.

7. 4. Development of the best model

Knowing that our main problem is overfitting, what we did was employ techniques to reduce it, aiming to significantly improve the results that previously indicated that we did not exceed 40% precision_at_1. Briefly summarizing, the techniques employed involved freezing all layers and then unfreezing only the last layer, to which we also added dropout. We also experimented with different types of triplets (ALL, hard, and semi-hard), and the best results were obtained using semi-hard triplets.

With all of this, we launched the execution again using the dataset of Van Gogh images, and the result was:

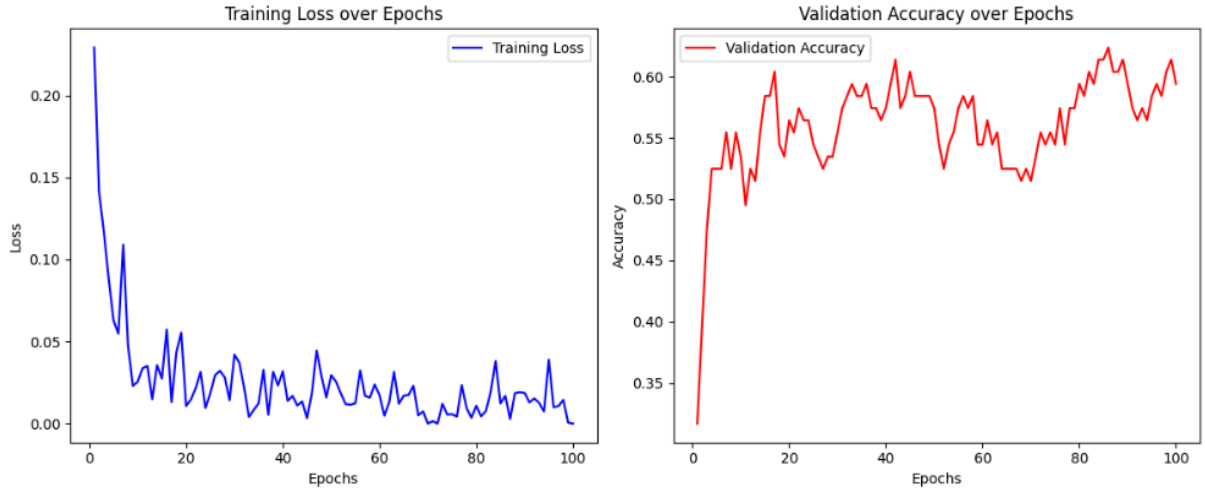

Image 39. Results of the best metric learning model

It can be clearly seen that compared to the last version of the network, there have been very significant improvements: firstly, in the loss curve, which shows a version that avoids the clear peaks observed before and follows a very clear decreasing trend. On the other hand, the results of metrics such as the validation and test accuracy are much better, reaching a value of 62% accuracy in the test set. Lastly, in the r precision, we went from barely reaching 0.2 to achieving a value of 0.5.

With this model as the best version of our metric learning and, therefore, saving its weights, the next step is to check how it would fit into our pipeline if it were to replace the binary date classifier.

7. 5. Pipeline

In the same way that we have done the pipeline with the date classifier, we will now also show how the pipeline would look like to chronologically order the artworks through metric learning.

For this case we have also seen 3 different models as reflected in the following image, on the one hand we have the realization of a network training with batches of random artists or custom batches only with

the images of an artist. And on the other hand, we have the model that has worked best for us, which is training the network with only the artist whose artwork we want to order.

To perform the ordering with the metric learning network, unlike with the date classifier, we don't need a specific sorting algorithm. Instead, using metric learning solution, we just assign a different label to each image, and then the corresponding date is assigned to each image based on these labels. Once this is done, all the images are sorted from oldest to newest according to its assigned date.

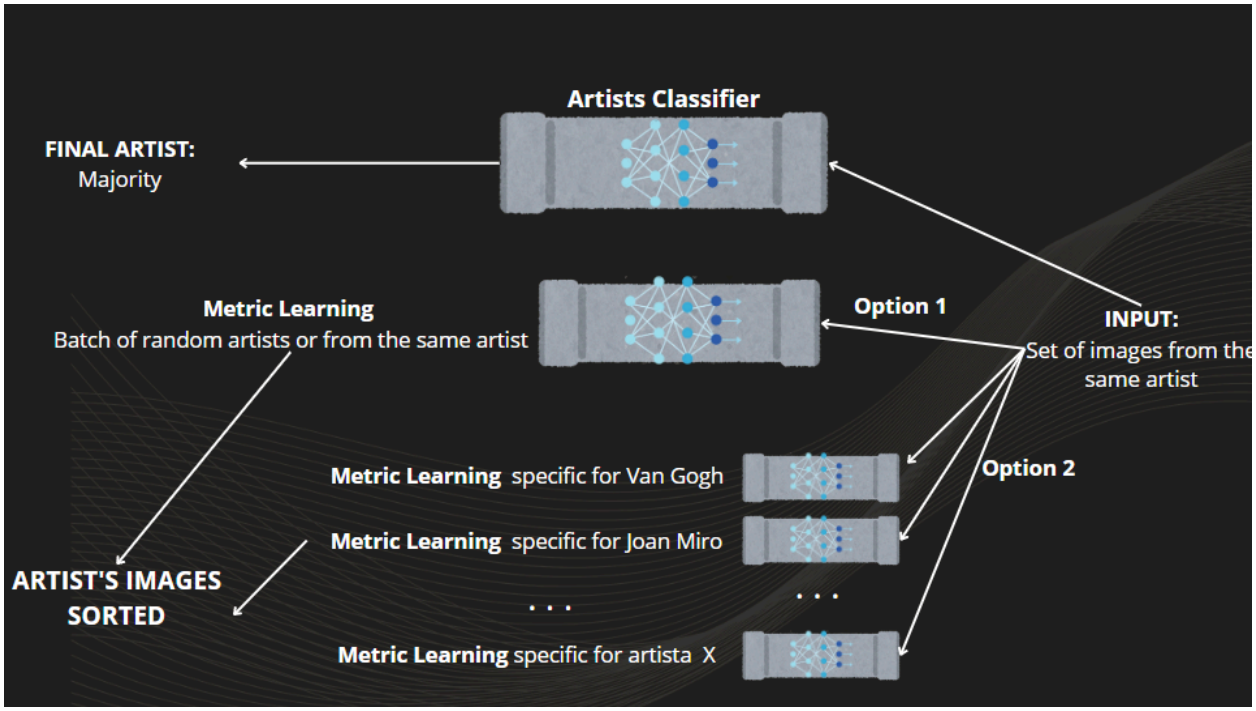

Image 40. Metric Pipeline VanGogh_4_Images_TRAIN set

Once the pipeline has been built, we have tested again with images of Van Gogh to test how the sorting works through metric learning. Here's an example of the execution for the Van Gogh specific metric learning model, being the one with the best results of the 3 different metric learning models. The execution was run with the images from the **VanGogh_4_Images_TRAIN** set. This allowed us to compare the best date classifier obtained (Van Gogh specific date classifier) with the best metric learning obtained (Van Gogh specific metric learning). The Van Gogh specific date classifier, with this same set, gave a 100% accuracy on the pairs, resulting in a perfect final ordering.

Image 41. Real sort VanGogh_4_Images_TRAIN set

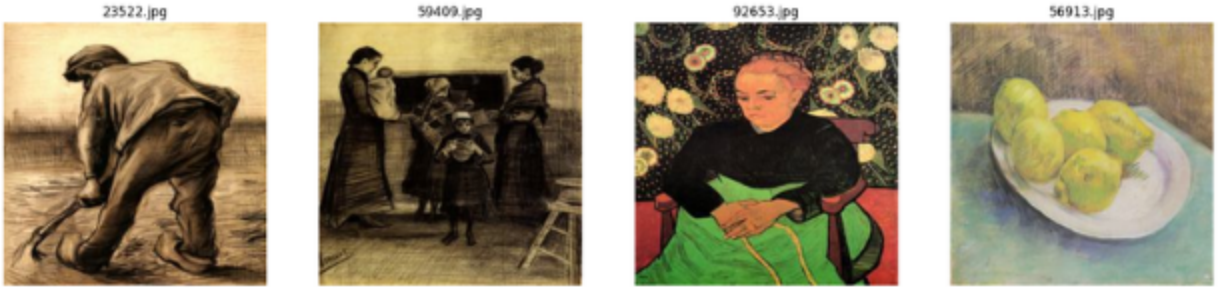

Image 42. Predicted sort VanGogh_4_Images_TRAIN set

Comparing images 41 and 42, we can see that unlike the Van Gogh specific date classifier, the order is not correct. This means that the Van Gogh specific metric learning, despite showing a test accuracy (see image 39) similar to the Van Gogh specific date classifier (66.67% on test set), performs worse when it comes to ordering.

Upon closer analysis of the ordering, the mean absolute error (MAE) of the metric learning net was calculated considering 100 random images of Van Gogh, giving a value of 4.46. This means that, on average, our metric learning net is wrong on each date by a margin of 4.46 years. Considering that the Van Gogh images the network was trained on span 13 different dates over 17 years (1973-1990), the obtained MAE is considered high. Therefore, it seems that to achieve correct chronological ordering, the date classifier would be the better option.

7. Conclusions and Enhancements

As conclusions from the work, we've successfully built an artist classifier that, given a set of images, can classify the artist they belong to individually or collectively. It's worth noting that the complexity of the network for this task will always depend on the volume of data being handled, as we've seen when launching the network trained with the top 5 artists on the dataset of the top 100 artists.

Regarding the sorting process, we've e been able to work firsthand with metric learning solution, which has been compared with the constructed date classifier. In both cases, we've observed that the best sorting for each artist would be obtained from networks specifically trained with images of a single artist. However, this wouldn't be efficient, as it would require as many networks as there are artists. Although the two solutions function differently (metric learning classifies by classes and the date classifier determines which image is older given two different images), both solutions require a sorting process. In other words, in neither of the two solutions is sorting obtained once the network is executed.

If we talk about improvements that could be made to this project, a primary focus would be to continue experimenting with our dataset to further reduce the overfitting of our models. Despite employing various techniques, there is still room for improvement. Additionally, we could explore other training techniques for our networks, taking into account the style of the paintings. This consideration could be crucial in distinguishing artists and achieving more precise sorting of the paintings.

Bibliography

- [1] Andrey Shitrauss, 2023, PyTorch | Multiclass Image Classification: https://www.kaggle.com/code/shtrausslearning/pytorch-multiclass-image-classification

- [2] Michael Fekadu, 2018, Painters-Train part x: https://www.kaggle.com/datasets/mfekadu/painters-train-part-1
- [3] Kevin Musgrave, 2019, PyTorch Metric Learning: https://kevinmusgrave.github.io/pytorch-metric-learning/

Contributors

- Pau Vilanova Castellana: 1638223@uab.cat
- Ricard Tuneu Font: 1634796@uab.cat
- Aksel Serret Llopis: 1639282@uab.cat
- Joel Bautista Rodríguez: 1605507@uab.cat