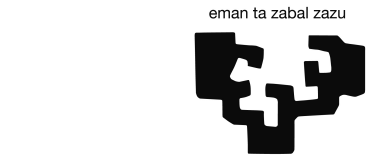

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Bachelor Degree in Computer Engineering

Computer Science

Thesis

Painting Style Classification using Convolutional Neural Networks

---

Author

Andrea López Rodríguez

2020

Abstract

The main objective of this project was to classify paintings of diverse styles by the artistic style using convolutional neural networks. These types of networks are deep learning models that have been widely used in image classification tasks. In this project, three different models were trained and tested with a dataset containing 16 different artistic styles: a simple network, the VGG-16 network and the ResNet-50 model. Before starting the multiclass classification experiment, a binary classification problem that aimed to determine how well those networks differentiated two artistic styles (Baroque and Impressionism) was evaluated with those convolutional models.

Besides, the three types of networks mentioned have also been used to solve another image classification task that consisted of distinguishing photographs from paintings. The three models were retrained with a dataset containing paintings of various styles and photographs ranged from amateur level to professional level.

Another task that has been solved in this project was to study and analyze the neural style transfer algorithm. Along with that, the original convolutional model for which the style transfer was proposed was replaced with one of the networks that had been retrained in this project to investigate how different the results would be with that modified model.

Finally, a graphical user interface was built to test the three different experiments in a more user-friendly way. It is intended for facilitating the application of the models without modifying the code.

Contents

List of Figures

Table index

CHAPTER 1

Introduction

1.1 Art Style Classification

Classifying the art style of a painting is a very important task, since the style of a work of art can be a great source of relevant information about the epoch in which an artwork was made. The information obtained from the style classification can also be used for solving problems such as grouping painters and paintings from the same epoch, relating recent works with more ancient ones, or obtaining information about the trends, lifestyle, customs and religion from the age when a certain painting was created. In summary, the painting styles of the artworks can be really useful information sources for historical researches.

There are many factors to take into account when it comes to determining the style of a painting: the media and paints used, the application of the paint and use of the colors, abstract, the theme and genre, among others. All those details need to be kept in mind in order to correctly classify an artwork into a certain style. Being able to identify all those characteristics in a fine art piece requires having a great education in many art concepts that include understanding color theory and picture composition, recognizing and interpreting many types of symbolism in graphic arts and historical knowledge for identifying the situation of the environment in which the painting was created, and a lot of time is needed to learn that information and understanding it. If all that time could be drastically reduced by using machine learning techniques, apart from saving many time

and resources, it would make a big difference when it comes to classifying art pieces, which is the main motivation to accomplish this project.

In addition to that, another related task that is related to painting classification is to estimate the resemblance between paintings and photographs, because on the one hand, some painting styles such as hyperrealism may be difficult to differentiate from an actual photograph, and on the other hand some photographs can have a great resemblance with paintings due to the composition within the picture or the color palette, so analyzing if a neural network is able to discern between those type of images is a relevant question.

Furthermore, a task that involves paintings, photographs and neural networks is style transfer. This consists of reconstructing a photograph with the style of a given painting while trying to keep the key elements of the image to be converted and the main features of the style to be applied in the final combined picture. Studying and executing the style transfer algorithm would be important to reveal how the networks read and interpret the two images and how the output images change depending on the layers used.

Finally, with those three tasks in mind, building a simple yet effective graphical user interface will be key in order to evaluate the machine learning models in a more visual and comfortable way, without any code skills needed. Along with that, with the interface available no code skills would be required to apply the models, therefore it would be available for everyone to test them.

1.2 Neural Networks as Image Classifiers

An artificial neural network (ANN) or simply a neural network [3] is a computational model inspired by the biological neural networks. Deep learning, part of machine learning algorithms, is based on the use of neural networks to solve all sorts of tasks. Those structures have been widely used in all kinds of knowledge fields for solving a wide variety of tasks such as language translation [4][5], computer vision tasks like pattern recognition [6] and autonomous driving cars [7], for example. The main topic of this project is focused on the second group of tasks since the data that is used is a collection of different images, and the goal is to classify them according to the painting style. Many kinds of different neural networks have been adapted to work on various computer vision questions that include pattern recognition [8] or object detection and classification [9][10]. For the latter, a competition in image classification called ILSVRC (or ImageNet Challenge [11]) has been held since 2010, where different deep learning algorithms have been evaluated

with a large image dataset called ImageNet. The algorithms that have produced the least error rate have been those that implemented convolutional neural networks (CNN), such as the AlexNet [12], ResNet [13] and VGG [2] networks. The accuracy obtained is related with the compatibility between the image structures and the CNN's convolutional layer's behavior, making that network type the perfect match for tasks that involve image processing. Other known applications involve handwritten text recognition such as numbers [14], being the MNIST dataset of handwritten digits [15] one of the most popular when trying to solve tasks that had text as input, consisting of 28x28 images of handwritten images. In this case too, convolutional networks have been those producing the lowest error rates compared to other neural network structures.

Besides CNNs, other neural network types have also been used in computer vision tasks, but with lower accuracy percentages. For example, on the one hand, Recurrent Neural Networks [16] have been used for handwritten text recognition, being able to maintain text coherence due to the sequence learning that those networks implement [17]; on the other, simpler networks such as Multilayer Perceptrons have also been tried in image analysis involved in robot controlling tasks [18].

1.3 Objectives of the project

The main goal of this project is to build a deep neural network that determines the artistic style of an unknown painting; therefore, this problem can be categorized as an image classification problem. To achieve this goal, three architectures with different complexity levels will be tested with two different datasets: a reduced one created for a binary classification problem between two painting styles (Baroque and Impressionism) and a general one containing images from nineteen different classes, which is meant to solve a multiclass classification problem. The classes that will be taken into account in the latter task are the following: Abstract Expressionism, Art Nouveau, Baroque, Cubism, Early Renaissance, Expressionism, Impressionism, Mannerism, Naïve Art, Northern Renaissance, Post-Impressionism, Realism, Rococo, Romanticism, Surrealism and Symbolism.

Along with that, two other problems involving deep neural networks are developed in this project. The main objective of one of the tasks is to solve the classification problem of differentiating between paintings and photographs. To solve this problem, again three neural networks with different architectures are designed and trained with a dataset built with two image categories, which are paintings and photographs.

The other task is to study and test the neural style transfer algorithm with networks that have been trained with different datasets. The network used is the VGG-16 [2], and the weights used for testing the algorithm will be the original ImageNet ones, the ones obtained in the Baroque and Impressionism experiment, and the weights obtained from the photograph and painting classification experiment. With those configurations, different tests have been made to investigate the effect of weights learned from different types of classification tasks in the images produced by the style transfer algorithm.

Lastly, a graphical user interface is developed to test the three developed tasks in a simple approach that does not involve programming skills. This has been built using the R programming language in the RStudio environment and its Shiny library for web application development. This interface contains three different tabs, one per machine learning task. Each of the tabs has the required buttons to load the images, set up the desired parameters and execute the algorithms, along with a main panel that is mainly used for displaying the results and the loaded images.

CHAPTER 2

State of the art

Paintings have been used in deep neural network and machine learning related works for various classification purposes. Most of them focus on determining painters rather than guessing the artistic style. For example, in [19], the authors use Weka [20] to try various machine learning algorithms that include multilayer perceptrons (MLP) [21], sequential minimal optimization (SMO) [22] for support vector machines and classifiers like Naïve Bayes [21], random forests [23] and AdaBoost [24]. In their procedure, some relevant image features like color, intensity and texture features were extracted from a dataset containing up to 20 artists. Their results, measured using the F1 score, showed that the MLP and the AdaBoost classifiers obtained the highest accuracy classifying painters taking into account all of the three features, obtaining a 75% of true positive rate.

Other works like [1] try to classify and relate painters that share similar artistic styles in an unsupervised way. The dataset used in [1] contains 994 works that represent 34 painters, each of them belonging to different art schools. The authors make use of an image analysis method based on the WND-CHARM [25] scheme, which was originally meant to be used in biological and medical images. The experiments made include determining general similarities between all the artists in the dataset used, and starting from there other tests like showing the similarities of artists that only use oil paintings were done. The results obtained were shown in the form of a phylogeny, where the relationships and similarities between artists and art schools can be exposed visually. An example of a phylogeny is shown in Figure 2.1.

Focusing on papers that target painting style classification, in [26] artworks are classified

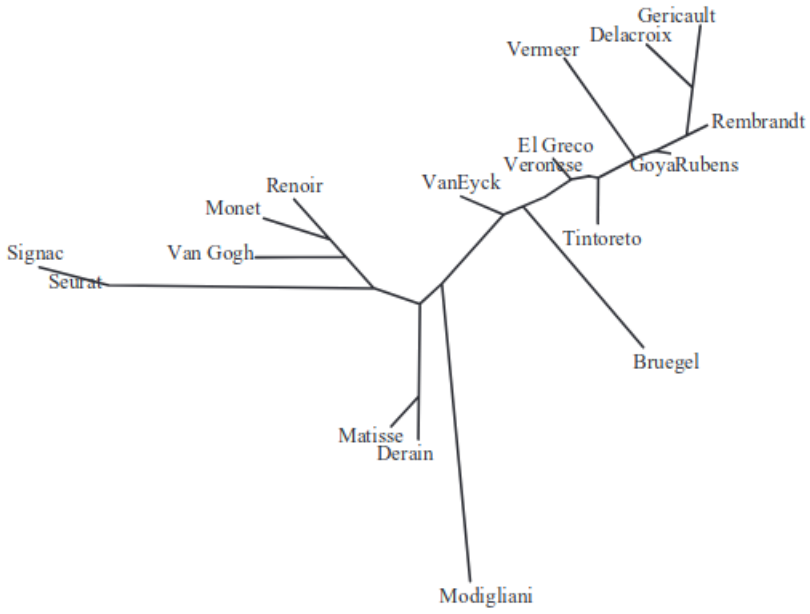

Figure 2.1: Phylogeny of painters that use oil paints. Figure taken from [1].

according to genre and style by using image descriptors such as SIFT (Scale-Invariant Feature Transform) [27], Gist image descriptors [28], HoG (Histogram of Oriented Gradients) and LBP (Local Binary Pattern) [29], GLCM (Gray-Level Co-Occurrence Matrices) [30] and the CIELAB color space [31]. Each of them was tested separately and also combined using different weights in order to form an ensemble, and finally, the accuracy was measured using the libsvm library [32] with χ^2 kernel. For the genre classification experiment, the highest accuracy was obtained by the feature ensemble, reaching an 84.56% of accuracy and followed closely by the SIFT feature that achieved an 82.53% accuracy. The top two classifiers in the style classification problem were the same as in the other task, where the first classifier obtained a 62.53% hit rate and the second a 59.20%.

Other works like [33] present a comparative study of different automated classification algorithms where the objective is to classify paintings between seven styles, having 70 pictures each. The approaches tested are discriminative models using both Bag-of-Words with CSIFT (Colored SIFT) [34] and OSIFT (Opponent SIFT) [35] as local features, and semantic level features and a generative model using only the Bag-of-Words approach. On the one hand, among the discriminative models, the best results were obtained by the one using the CSIFT local features, achieving an 87.5% accuracy when classifying Baroque style; on the other hand, the generative model that got the highest accuracy was the one using the same local features as the best discriminative model, with an accuracy of 86.6% and also with Baroque style.

The article [36] describes an attempt to use deep learning techniques for recognizing artistic styles, which is the main goal of this project. The amount of styles to be classified goes up to 25, having an average of 2662 images per class, and the networks used are AlexNet, ResNet34 and ResNet50 networks, pre-trained on the ImageNet dataset, an image dataset prepared for object recognition problems. In order to increase the accuracy and stability of the classifiers, some training and testing improvements such as bagging and dataset augmentation are conducted. Top-K accuracy is used to measure the accuracy of the results with K = 1, 3 and 5. On the one hand, the overall results showed that the best results were obtained when implementing bagging, obtaining a 93.6% accuracy using the top-5 accuracy measure. On the other hand, doing a per-class analysis of the results, it can be seen in [36] that better results are obtained for classifying styles with a more distinctive appearance such as *Ukiyo-e* or *Color Field Painting* rather than others that have a more similar style like *Baroque* and *Mannerism*.

In other papers like [37] the task of distinguishing photographs from illustrations is solved rather than classifying paintings by style. In that work, different models are benchmarked in order to see which of them obtained the best accuracy for solving the proposed problem. The used models had different approaches, such as using image features (outline detection and color intensity variance) and color histograms for classifying the images, or using the bag of words method combined with an SVM as a classifier. Deep learning architectures were also tested, including customized CNNs with different numbers of layers (from one layer up to five) and a fine-tuned AlexNet with various configurations. Two different datasets were used to test the mentioned model: one with real world photographs and illustrations, and the other with photographs of people wearing fictional character costumes and anime illustrations. After doing the tests, the fine-tuned deep convolutional neural network that succeeded achieving a 96.8% of accuracy with both datasets, followed by the custom CNNs with the highest amount of layers, with a 93% of accuracy. The other models that involved image feature detection were those that obtained the lowest accuracy.

Focusing on the neural style transfer algorithm, works like [38] have tried to solve this task using the different layers within a VGG convolutional neural network to obtain the relevant features of the base image and style reference painting to generate an output that preserves both the content of the base image and the key features of the style reference image. To achieve this objective, the weighted addition of two different loss functions was minimized and back-propagated through the network: the style loss and the content loss. Both loss values represent how similar the output image is to the base image or the style reference image. The weight assigned to each of the loss values will affect how well the

content or the style are represented in the output image. This process is done for a fixed number of iterations. Different layer configurations were tried, and in the end, using the first convolutional layer of each of the convolutional blocks to obtain the style features and the second layer from the next to the last layer to get the base image content achieved the best results.

Finally, the article [39] focuses on the painting style transfer problem on head portraits in order to avoid the facial deformations that often happen when applying neural style transfer algorithms in those types of photographs. Using the VGG convolutional neural network, the approach followed in this paper was to modify the base portrait's feature maps by transferring the painting image's color distribution onto the base photograph using gain maps. This way, the style reference image's features are maintained along with the facial features, keeping the input portrait and the painting reference recognizable.

CHAPTER 3

Project Management

This section covers the planning process followed during the development of the project. This is based on an initial planning that may have had some changes during the development process.

3.1 Planning

Before starting the development, some tasks have been identified in order to complete the project. Those can be divided into two groups: implementation and documentation.

While the documentation tasks are related to the thesis writing, the implementation group includes the following tasks:

1. Organizing and preparing the needed datasets
2. Researching models that have been used for the main goal of this project
3. Adapting the models and trying them in binary and multiclass experiments
4. Studying and analyzing the results
5. Studying and applying the neural style transfer algorithm

To plan and organize the work expected for those two groups of tasks, a WBS diagram and a GANTT diagram are presented in figures 3.1 and 3.2. Those diagrams include all

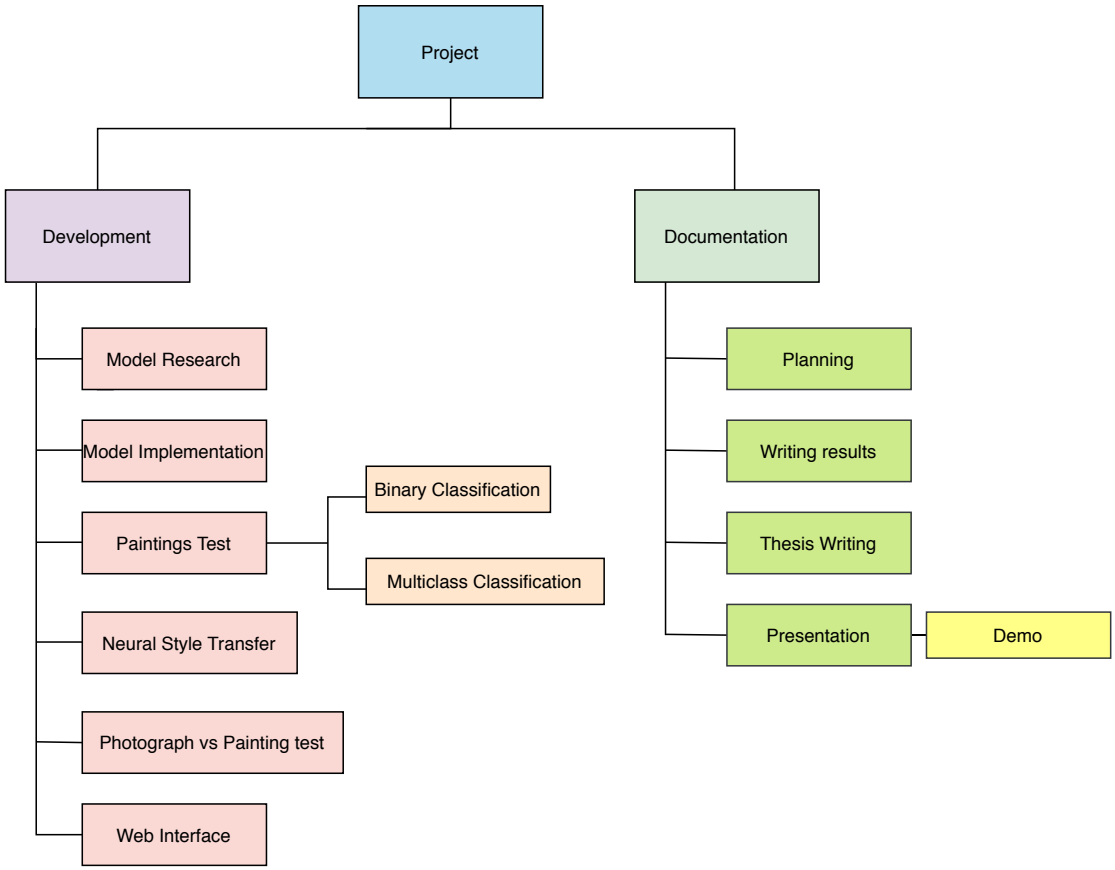

Figure 3.1: Work Breakdown Structure diagram

the basic tasks to be done in order to complete the initial objectives for the PFG with enough time to add corrections if necessary or include extra tasks to make this project more complete.

3.2 Risk prevention

To prevent future problems during the development of the project a risk management task has been done in order to avoid them. The main hazard for this project is data loss: this problem could be critical for the development of the project. To prevent it, the dataset is stored online and available for download in the Kaggle platform; also, it is stored in other external devices to avoid downloading it several times. The project thesis is written in an online platform (Overleaf), and copies of the written documents are stored in a Dropbox repository.

There are other risks such as bad results and delays: the first problem can be prevented

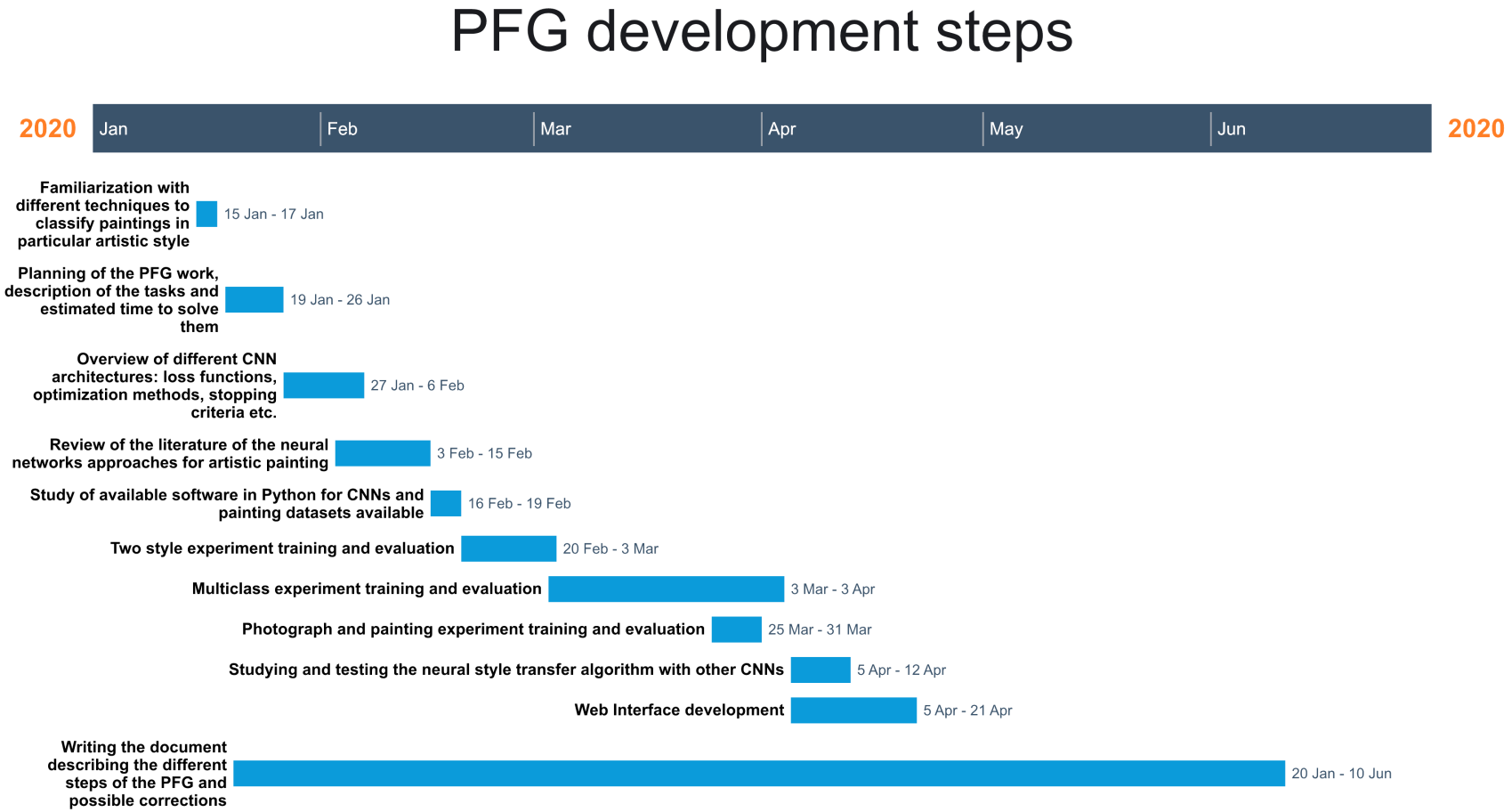

Figure 3.2: GANTT diagram showing the approximate dates for each step.

by carefully studying the code and spotting and correcting possible mistakes. The second one can be avoided by preparing and following a work and delivery schedule.

Last but not least, it has to be taken into account that the computational cost and hardware resources that this project requires to be developed are high since the CNNs used are complex. If those needs are not properly satisfied the time needed to develop can be too long and cause delays in the project. In order to avoid that, powerful GPUs and CPUs have been used during the training of those networks to dodge waiting too long for the results of the experiments.

CHAPTER 4

Introduction to CNNs

A Convolutional Neural Network (CNN or ConvNet [40]) is a type of Deep Neural Network mostly used for image analysis, in tasks such as image recognition or classification. Taking into account the tasks mentioned, the usual input for these ANNs is an image dataset, where each of the images is taken as a matrix of H rows and W columns and three (RGB color channels) or a single channel (depending on whether the image is polychrome or monochrome) [41]. For this particular project, the input images will be colored paintings that will mostly have three RGB channels.

4.1 CNN Architecture

This type of network has the usual layers of a DNN: an input layer, an output layer and the group of hidden layers, where the number of hidden layers may vary depending on the architecture implemented.

The hidden layers of this network are divided into three groups: convolutional layers, pooling layers and dense (fully connected) layers. The layers are usually distributed in the following way: first the input layer, then several blocks of convolutional and pooling layers, and finally one or more dense layers followed by the output layer. The structure may vary from architecture to architecture.

The function of the group of interleaved convolutional and pooling layers is to perform

feature learning, and the last dense layers (including the flatten one) perform the classification task.

4.2 Convolutional Layer

The purpose of this layer is to extract features from the input [42]. To accomplish this goal, a series of kernels is spread through all the dimensions of the image (width, height and depth) and feature maps are generated. A feature map (or activation map) is the output of a convolution between a kernel and an image (or a feature map from the previous layer), which contains the mapping of the features found in that image. A kernel is a matrix of a smaller dimension that is used to transform a bigger matrix. It contains values (weights) that determine which kind of features will be detected on the image. Unlike dense layers, the neurons in this layer are not fully connected to the surrounding layers but to a section.

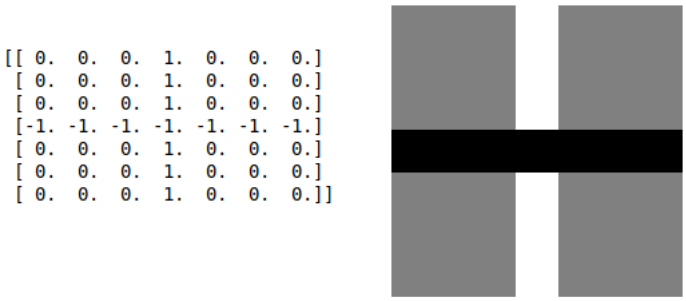

Figure 4.1: 7x7 kernel used for horizontal line detection.

The convolution between the two matrices is performed by doing the dot product of the elements on the kernel and the matrix and adding the results. This operation is repeated through all the initial matrix by sliding the kernel from left to right and from top to bottom.

When sliding the kernels across the input, there are two parameters to have into account: the stride and the padding. The stride defines the number of positions the kernel will move in each step, while the padding prevents the original input from shrinking and losing information from the edges by adding extra layers of pixels to the border of the images. There are different types of padding, but the most used one is the zero padding, which adds layers of zeros to the borders of the input.

4.2.1 Pooling/Subsampling Layers

Pooling layers are used to reduce the computational complexity of the model by reducing the size of the obtained feature map. [43]. The pooling process is similar to the convolution process since a kernel is used, but in this case, the kernel slides through the input avoiding overlapping regions. There are different pooling methods, but the most used one is max-pooling, which consists of down-sampling the feature map to the pixel with the highest value in the sub-sample analyzed by the kernel at the moment.

After down-sampling the feature maps, usually an activation function is applied to them. One of the most used activation functions is ReLU due to its low computational cost.

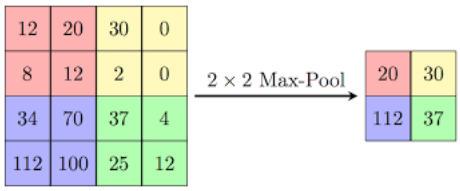

Figure 4.2: Max-pooling process with a 2x2 kernel.

4.2.2 Dense Layers

The neurons in these layers are fully connected to the corresponding input and output layers. These layers perform the task that the CNN is supposed to do (classification, regression...). The inner structure of fully connected layers is similar to the MLP's structure.

Among the most used activation functions in these layers are Softmax and Sigmoid functions. Softmax is used in multiclass classification problems, while Sigmoid is used in binary classification problems.

4.3 Pre-trained Networks

As mentioned in the introduction, this project will make use of two already existing network architectures: the VGG-16 [2] network and the ResNet-50 [12] [13] network, which are available in the Keras library.

A VGG neural network is a deep convolutional neural network based on the AlexNet [12], designed for object recognition. Its name comes from Oxford's Visual Geometry Group,

who trained and developed these networks and obtained a great performance when testing the ImageNet dataset in the ImageNet Challenge in 2014. The input for these networks was a 224x224 pixel RGB image, and the output obtained was a vector of probabilities containing the values for each of the 1000 classes used in the ImageNet dataset.

These networks are known for improving their accuracy by increasing the depth of the network. That could be achieved using small receptive fields, making it possible to stack up many layers. The number of stacked layers is what distinguishes the different types of VGGs. In this project, the VGG-16 network is used, which has a total of 16 layers stacked. Figure 4.3 shows the different architectures of networks depending on the number of layers stacked.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4.3: VGG configurations. Each column (A-E) describes the architecture of the developed VGG models. Bold type layers point out the changes that differentiate each convolutional network. Figure taken from [2].

The residual networks (ResNet) are very deep neural networks used for image recognition tasks that combine residual learning with convolutional blocks [44]. There are different models of this kind, which are ResNet-18, ResNet-34, ResNet-50 (the one used in this project), ResNet-101 and ResNet-152. The difference between those networks is the number of layers that are stacked.

Taking into account that in convolutional networks the features of an input are learned at the end of a layer, residual learning consists of learning the residuals obtained instead of the actual features of the given input, being the residuals the subtraction of a feature learned from that input layer. This is done by creating shortcut connections between layers, where for example, an n^{th} layer is connected to the $(n+1)^{th}$ and also to the (n+x) layer, being x the number of layers skipped. In the end, the values of the regular path and the shortcut connections are added with an addition layer. Figure 4.4 shows the basic building block of a residual network and shows the shortcut connection mentioned. In some versions, batch normalization and ReLU layers may also be included inside the residual block.

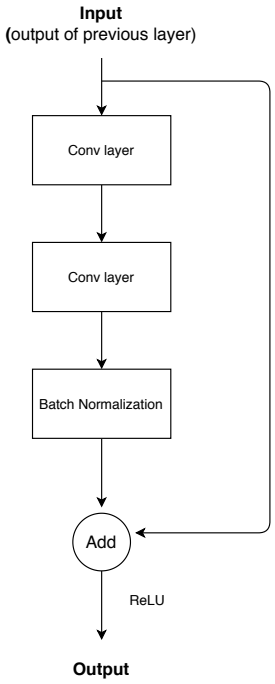

Figure 4.4: Basic residual block example.

The goal of introducing residual learning is to solve the training accuracy degradation problem that may occur in networks that have several layers stacked up, where the accuracy obtained can end up saturating and thus deteriorating.

CHAPTER 5

Dataset Description

5.1 Painting Classification Dataset

The dataset used to perform the training, validation and testing tasks is the one provided in Kaggle's Painter by Numbers competition [1]. The pictures used in the dataset are mostly taken from the WikiArt painting collection. There are a total of 103250 pictures in the entire dataset, where 79433 (76,9327%) are used for the training process and 23817 (23,0673%) for testing. The images that the dataset contains are from different artists and generations, and the total number of different art styles goes up to 136. Part of the training dataset is also used for the testing process, as some images are tagged as *train_and_test* in the file containing the dataset information. Figure 5.1 shows an example of the art style diversity within the whole database:

The images contained within the dataset are mostly RGB or grayscale and of different dimensions, and feature various painting genres such as still life, portraits, landscapes and animal paintings. As the dataset is analyzed, it can be noted that the classes in the dataset are not balanced at all, since there are classes with less than 100 artworks, whereas other classes have more than 5000 images. With such a small amount of training pictures, it is unlikely that data augmentation could help to solve the imbalance problem. Therefore, the way chosen to address this issue is to reduce the number of classes for this project, saving only those that contain more than 1000 images and creating new pictures with data

[1] Competition link: https://www.kaggle.com/c/painter-by-numbers

Figure 5.1: Different styles in the dataset. From left to right, top to bottom: Color Field Painting, Realism, Post-Impressionism and Baroque

augmentation in those classes that have fewer pictures. The new number of pictures per category in the training set is described in Figure 5.2.

It can be seen in the bar chart that even after reducing the number of classes that the neural network will learn from 136 to 16 the data amount is still unbalanced, but in this case data augmentation is far more viable than before.

After setting the final number of classes of our modified classification problem, the images have been organized in folders according to the class that they belong to. The same procedure has been followed for the validation set.

The training set has been built with images that were tagged as *train_only* and *train_and_test*,

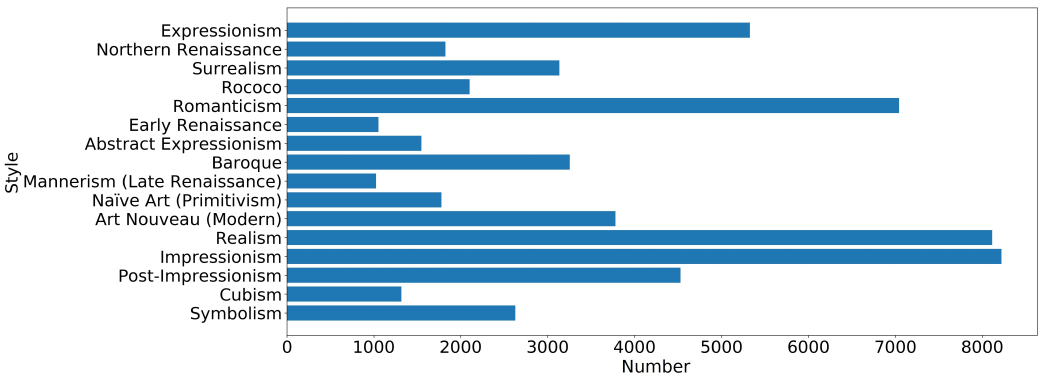

Figure 5.2: number of pictures per art style in the training set.

leaving the remaining ones that were labeled as *test_only* just for the testing purpose (2936 pictures in total). From those pictures, the ones that were not from the classes that this project uses have been deleted, having a final amount of 2022 testing pictures. For the training of the network, the number of images for each of the classes had to be balanced since the difference of the number of images among all the categories is too large to leave it as it is, and if left like that, the CNN would end over-fitting. To avoid that, the images have been re-scaled to a certain dimension and rotated in different ways, creating new instances and reaching a more even number of images for each class, ending with an average of 4446 pictures in each folder, and a total number of 71144 images.

5.2 *Photograph vs Painting* dataset

The dataset chosen to prepare a neural network for the task of separating real images from paintings is the flickr30k [1] image dataset, available in the Kaggle platform. This dataset consists of real life images and was used for image description related works [45].

Figure 5.3: Images from the flickr30 dataset.

The number of images in this dataset goes up to 31783 pictures, and includes several types of photography, such as landscapes, portraits and daily-life situation pictures, mostly in a beginner photograph quality. Figure 5.3 shows some examples of the images in this dataset.

[1] https://www.kaggle.com/hsankesara/flickr-image-dataset

CHAPTER 6

Data Preprocessing

Data preprocessing has been an essential part of this project since it helped to increase the number of images of those classes that had few pictures comparing to others, making the results of the final neural network more balanced. For this process, Python's PIL (*Python Imaging Library*) has been used for performing different transformations and changes to the images. Those changes include organizing and cleaning the dataset, re-scaling and rotating the images, and this process has been applied in the whole training set.

The steps followed during the data-preprocessing procedure are the following:

1. Remove unnecessary artworks.
2. Re-scale the pictures
3. Split into train and validation sets.
4. Data-augmentation.

6.1 Paintings dataset

The first step of this process has been organizing and deleting images that were not needed, as there were pictures that belonged to classes that would not be learned by the neural network. To do so, the pictures that wouldn't be used have been deleted from the

original dataset with a script. After that, the remaining images were organized in folders according to the category they belonged to. The categories have been retrieved from the file *train_info.csv* has been analyzed. That file contains information about each of the images, including the author, title and the style, which is the information used to create the final folder organization. After deleting unnecessary pictures, the number of images obtained for training is 59857.

With the images organized, the next step has been to re-scale all the pictures to the same size in order to normalize the input and decrease the input size. The chosen value has been 224 x 224 so that the paintings are still recognizable and the complexity of the neural network is reduced. The resizing step has been accomplished by using a script with PIL's *resize* function, and overwriting the original file after the transformation. Proportions of the resulting image were most likely to appear distorted due to the transformation that forces the picture to achieve those concrete dimensions.

Figure 6.1: *La Plume* magazine cover art by Alphonse Mucha. Proportions of the resulting image are clearly distorted in comparison to the original.

Before applying the data-augmentation step, all the images were distributed into the training and validation sets. For the validation set 200 pictures of each class were taken, having a total of 3200 images for the validation set. The distribution of the paintings for each of the sets is 95% (56657 images) for the training set and 5% for the validation set.

The last step for the preprocessing has been to create more data from the original one with the purpose of balancing the number of pictures of each class. The transformations applied to each class vary depending on the original quantity of pictures in that category. The goal was to achieve an average amount of images of around 4500 pictures per class. In order to achieve this goal, each of the categories has been augmented with different amounts of pictures, applying various transformations in each case, which consist of horizontal and

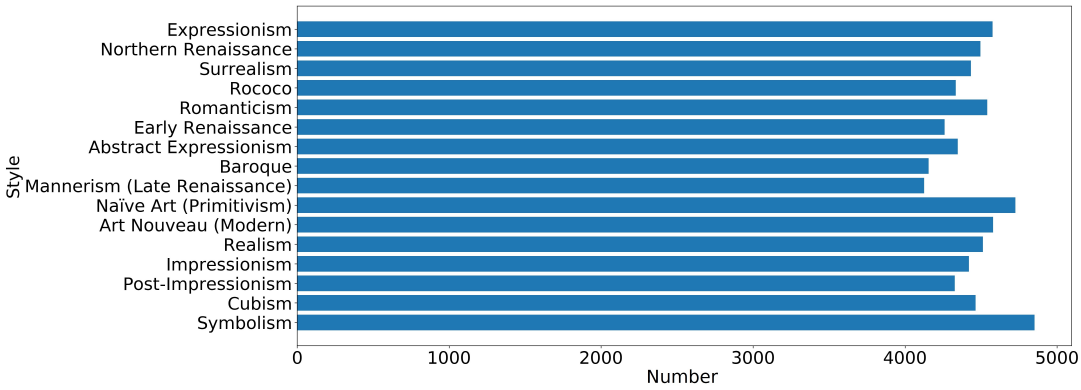

Figure 6.2: Number of pictures per art style in the training set after the completion of the data augmentation step.

vertical flips along with rotations in two directions. An example of those transformations is presented in Figure 6.3. Classes with less than 1500 pictures have been augmented using up to four transformations; in those where the amount of pictures goes up to 3000 have had no more than three different conversions applied, and those that were near to the desired average picture amount have not been augmented in more than two ways. For classes with more than 5000 images, some images were removed to reduce unbalancedness. In the end, the average amount of pictures per class has been set to 4470 pictures per style, and the total number of images has been increased to 71144 pictures, which means an increment of 25.5%. The new number of images per class is described in Figure 6.2.

6.2 Preprocessing photographs

The photographs for this dataset were taken from the previously mentioned flickr30k dataset containing 31783 pictures. These pictures were RGB pictures with different height and width values, so in order to fit the input size for the architectures of the neural networks, the images were resized to 224x224 pixels with the same method that was used for the paintings dataset.

Data augmentation was not performed since the number of photographs was more than enough to train a network for binary classification. To build the training, validation and test set, the dataset was divided in the following way: 80% of the images (25426 pictures) were used in the first set, and the remaining 20% (6357 pictures) was saved for testing. From the training set, an amount of 2546 (around 10% of the training pictures) were used for validation. In the end, 22880 pictures were put in the training set, 2546 in the

validation set and 6357 in the testing set.

Figure 6.3: *Tranquility* by John William Godward. Original image and the generated data with four different transformations.

CHAPTER 7

Architectures

Three neural models with different architectures and parameters will be tested with three different image classification problems in order to study how the complexity of a network can affect its performance with different datasets. In this project three types of models are considered:

- Simple Network: A simple CNN made from scratch
- VGG-16 Network
- ResNet-50 Network

Each of the networks is described in the sections below. The more complex networks have been chosen according to the results obtained in other related works, and those networks are the VGG-16 [2] and the ResNet-50 [13]. Since every network has a different configuration and this may have changed during the experimentation phase, the final number of parameters of each of the networks in the three experiments is summarized in Table 7.1.

Table 7.1: Number of parameters of each model in the different experiments.

	Simple CNN	**VGG-16**	**ResNet-50**
Binary Experiments	3,320,321	15,009,729	24,767,489
Multiclass Experiment	38,777,296	14,722,896	30,011,344

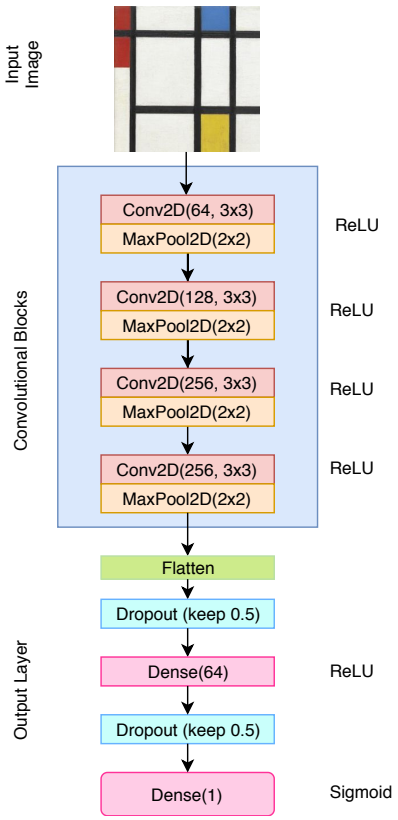

Figure 7.1: Simple network built for the binary classification experiments, as the output layer only has one unit.

7.0.1 Simple Network

The simplest network tested is a convolutional neural network made from scratch, built with a total of 13 layers, where four layers are convolutional layers with different sizes, the other four layers are maxpooling layers, and the remaining five layers consist of a flatten layer, two dropout layers and two dense layers. The activation functions chosen were ReLU (*Rectified Linear Unit*) [46] for the convolutional layers and the first dense layer, and Sigmoid for the last dense layer, which is the output. For the binary experiment, the output layer has only one unit, returning 1 or 0 depending on the class: 0 representing Baroque art style and 1 for Impressionism. For the multiclass experiment, the last layer has 16 units, one per artistic style. Along with that, another layer has been added to prevent overfitting from happening: a dense layer with 1024 units. In Figure 7.1 the architecture used and its organization are shown.

7.0.2 VGG-16 Network

As mentioned in Chapter 4, this network is a convolutional neural network built using a total of 16 weight layers.

The architecture of this network has been modified using a pre-trained VGG-16 model available in the Keras library. The base network has already been trained using the ImageNet challenge dataset. From that model, the top layer in which the output of the original model is returned is not added, and only the last five layers were trained with the binary dataset experiment, leaving the rest frozen and retaining their previous weights. To that architecture another five layers were added: a MaxPooling 2x2 layer, a flatten layer, a dense layer (with 64 units), a dropout layer and a final dense layer, with a single unit or 16 units depending on the dataset used. In the two dense layers, the activation functions chosen have been ReLU and Sigmoid respectively. For the multiclass experiment, all the layers were set not to be trainable, in order to take advantage of the whole network to classify the different 16 styles. To those layers, only two layers were added apart from the 16 unit dense layer, since this was the configuration that retrieved the best results solving the multiclass experiment: a MaxPooling 2x2 layer and a dropout layer. Figure 7.2 shows the configuration used for the binary classification experiments.

7.0.3 ResNet-50 Network

The procedure used for the pre-trained ResNet-50 model is similar to the one used for the VGG-16 network. The model has been loaded from the Keras library with the ImageNet challenge weights, and the top layer has been discarded. This time, instead of just freezing every layer instead of the last five ones, several tests have been made with different frozen layer proportions. In the end, for the binary classification problems the layers of the network were left trainable except for the first 20 ones. On top of those, the same five layers that were added to the VGG-16 network were included in this model as well. The model is represented in Figure 7.3.

For the multiclass classification task, this model has gone through a lot of modifications due to the low validation accuracy obtained. Those changes go from modifying the ratio of frozen and trainable layers, adding more dense layers and increasing the amount of dropout in those layers. In the end, the resulting model using the ResNet5-50 network was built by leaving trainable the last residual block of that pre-trained model and the batch normalization layers. Finally, a flatten layer, a pooling layer, and two dense layers

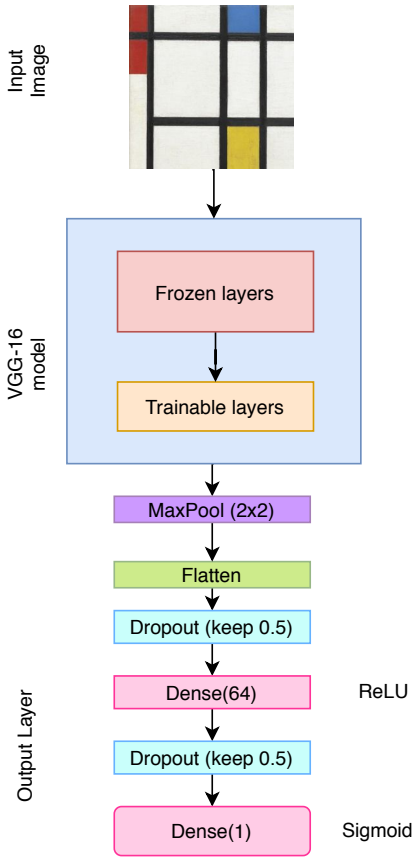

Figure 7.2: VGG-16 based model used in the binary classification experiments, as the output layer has a single unit.

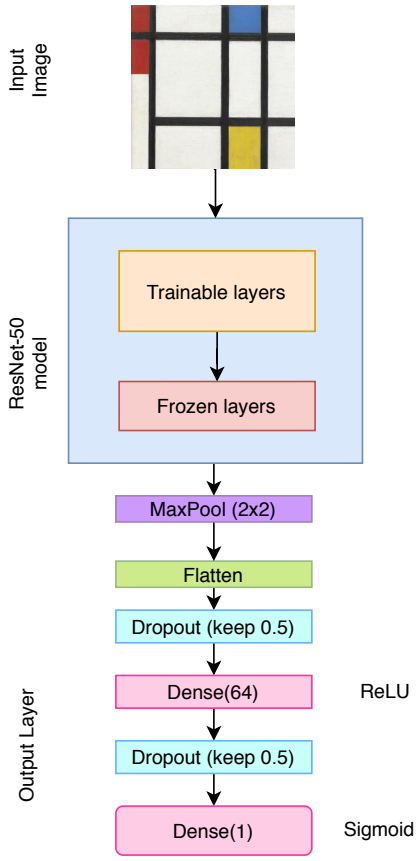

Figure 7.3: ResNet-50 based model used in the binary classification experiments, as the output layer has a single unit.

with a dropout layer in between were added. The first dense layer is built with 1024 units, and the second one with 16, to retrieve the probability of a picture belonging to each of the styles.

CHAPTER 8

Experiments

The objectives of the experiments are evaluating and testing the capacity to discern between painting styles and distinguishing paintings from photographs for both new and preexisting models. The first experiment consists of evaluating the models for a binary classification problem, in which the two classes to differentiate are Baroque and Impressionism. An example of both classes can be seen in Figure 8.1. The second experiment will evaluate the models for a multiclass classification problem with 16 different painting styles. Finally, in the third experiment, we will evaluate the models for another binary classification problem where the aim is to classify between paintings and photographs.

The experimentation in this project has been developed in two stages: the first stage uses a rather smaller dataset for solving a binary classification problem, while the second one

Figure 8.1: *San José* by Guido Reni, Baroque style, and *Pietà* by Van Gogh, Impressionist

is the multiclass classification problem that this project aims to solve. The images used for the binary classification are taken from the original dataset, using the pictures tagged as Impressionist and Baroque styles.

The programming language chosen for the setup and implementation of the experiments has been Python in a jupyter notebook styled environment provided by the Kaggle[1] website. The libraries used have been Keras with TensorFlow backend for the neural network implementation and Scikit-Learn Machine Learning library for the result analysis and statistics. The hardware provided includes an NVIDIA Tesla P100 GPU, 16 GB of RAM memory and an Intel Xeon CPU with two cores. From all of those resources, the GPU has been the most valuable resource in this project due to the reduction of the training time it provided, achieving an average of 39 seconds per epoch.

8.1 Image preparation

The images have not been really manipulated since when it comes to artistic style factors such as color hues and image composition are really important. The only changes that have been made affect the dimensions of the pictures that have been set to 224x224 in order to have uniform dimensions in all the dataset and, in those cases where data augmentation is needed, the images have been rotated in different angles and flipped horizontally and vertically.

To fit all the image sets (training, validation and test), Python's `ImageDataGenerator` and `flow_from_directory` functions were used. To match those methods when fitting and testing new data, functions `fit_generator` and `predict_generator` were needed.

8.2 Training

The neural networks' training has been done differently depending on the experiment. For the first binary classification task, the training process has been done in two steps, both with a maximum of 100 epochs:

1. Training with an initial configuration
2. Tuning and finding the most optimal configuration for each network

[1] https://www.kaggle.com/

The training process for the other two image classification experiments completed in just one step, using as a base the configurations obtained in the second step of the first binary classification experiment.

To speed up the training process and avoid overfitting, the function `EarlyStopping` from the Keras library was used. This function monitors how a given metric that describes the performance of the network changes during the training, and stops the process when the metric reaches its maximum or minimum possible value. In this case, the goal is to optimize the validation loss.

In the first step, the initial training was done with the following settings: firstly, the patience and min_delta for the `EarlyStopping` function from the Keras library were used. The patience parameter sets the maximum number of steps to wait before stopping the training once a minimum validation loss has been found, while the min_delta will set the minimum value to consider as an improvement in the monitored metric. With those parameters set the `EarlyStopping` function will stop the training when the patience epochs are met or the total training epochs have ended.

These parameters have been set to 10 and 0 initially. Along with that, the batch size has been set to 32 for all the networks. Finally, the initial optimizers chosen for this first training were the following: Stochastic Gradient Descent [47], with a learning rate of 0.001 for the simple one, and RMSProp with a learning rate of 0.0001 for the VGG and the ResNet models.

After performing the training with that configuration, the second step started, where each of the networks has been trained with different parameters in order to obtain the best model. The parameters, optimizers and batch size have been manually adjusted so that the results of the previous training processes were improved.

The loss functions used in the three experiments were binary cross-entropy for the binary classification problems and categorical cross-entropy for the multiclass classification task.

8.3 Results

This section wraps up the results of the experiments for all the architectures with the two datasets.

8.3.1 Binary Classification Results

The following experiments were performed with three networks introduced in the previous sections, using an output layer of a single unit to get the classification results. The dataset used for training has only two classes: Baroque painting style and Impressionism.

For this experiment, the training set comprises 5106 pictures, a total of 2553 pictures per class, and a total of 1400 images for validation, 700 for each category. Finally, a total of 414 images were used for validating the network, 207 for each target class.

Simple Network

During the first training, the network used 21 epochs and obtained a training accuracy of 85.61% and a validation accuracy of 87.57%, which is not really bad for a network built from scratch. In Figure 8.2, it can be seen that both validation and training values end up converging, but the validation loss shows some spikes that suggest that the learning rate of the function is too high. After the training was completed, the test set was used to try out the network, and the final accuracy obtained was 78.74%. The obtained confusion matrix is shown in Table 8.1. It can be seen that the network is clearly biased towards the Baroque class. An example of misclassified images can be found in Figure 8.3.

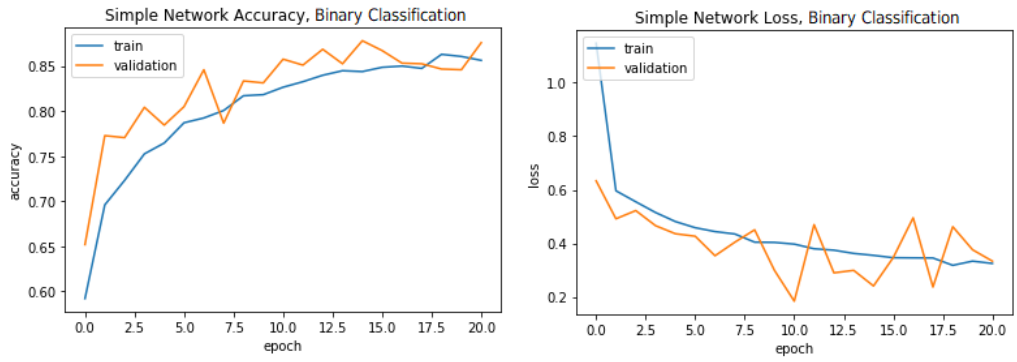

Figure 8.2: Loss and accuracy of the simple network with the initial configuration.

With the initial results in mind, some tuning procedures were made in order to improve the accuracy of the model until a more balanced version was obtained. The activation function for the last layer was kept, while the optimizer was changed to Adam due to positive results after some preliminary tests. The learning rate was reduced to 0.00001 to

Table 8.1: Confusion matrix for the binary classification experiment with the simple network, first experiment.

	Baroque	**Impressionism**	**Test Samples**
Baroque	185	22	207
Impressionism	66	141	207

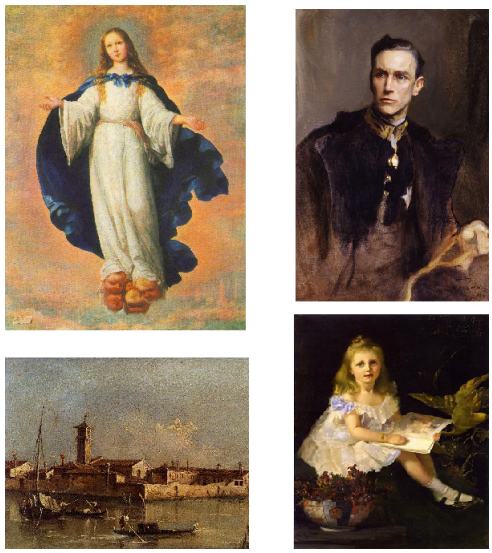

Figure 8.3: Misclassified samples by the simple network. On the left column, two Baroque paintings classified as impressionist: *Inmaculate Conception* by Francisco de Zurbarán and *View of the Island of San Michele near Murano, Venice* by Francesco Guardi; on the right, two misclassified impressionist paintings: *John Loader Maffey, 1st Baron Rugby* by Philip de László and *Louise, Daughter of the Hon. L. I. Smith* by Tom Roberts.

ensure that the loss function would not diverge. The batch size was also reduced to a half, having now batches of size 16.

Figure 8.4 shows the accuracy and loss function charts for the new configurations, which now are more stable in comparison to the first ones. The accuracy values obtained this time were 83.02% for the training and 83.09% for the validation accuracy, using a total of 13 epochs. The values in the new confusion matrix in Table 8.2 indicate that the network is now more balanced even though the accuracy for the test set has been reduced to 73.19%, since the overfitting produced in the first training has decreased and the network is more capable of distinguishing both styles more equally.

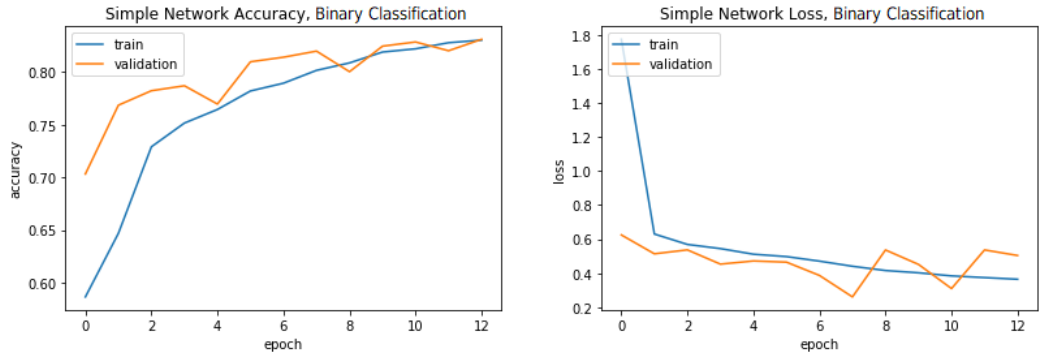

Figure 8.4: Loss and accuracy of the simple network with the final configuration.

Table 8.2: Final confusion matrix for the binary classification experiment with the simple network.

	Baroque	**Impressionism**	**Test Samples**
Baroque	156	51	207
Impressionism	60	147	207

VGG-16 Network

In the first experiments, the training process for this model used a total of 40 epochs out of a maximum of 100. The total accuracy achieved was 100% with the training set and 94.23% with the validation set, which was to be expected for a network of this kind. However, analyzing the results for the test set in Figure 8.3 and the accuracy of 87.43% obtained with that set, it can be seen that again the results are to some extent unbalanced, classifying more images as Baroque rather than as Impressionist. Looking at the accuracy and loss graphs in Figure 8.5, it can be seen that the loss varies considerably and has several spikes during the training process. Again, as it happened in the simple model, this can be due to a rather small learning rate for the optimizer. Having a training process that lasted for too long can also make the loss prone to spiking and producing overfit results.

Table 8.3: Confusion matrix for the binary classification experiment with the VGG-16 network, first experiment.

	Baroque	**Impressionism**	**Test Samples**
Baroque	185	22	207
Impressionism	30	177	207

Attempting to reduce spikiness and the overfitting produced, and making the network

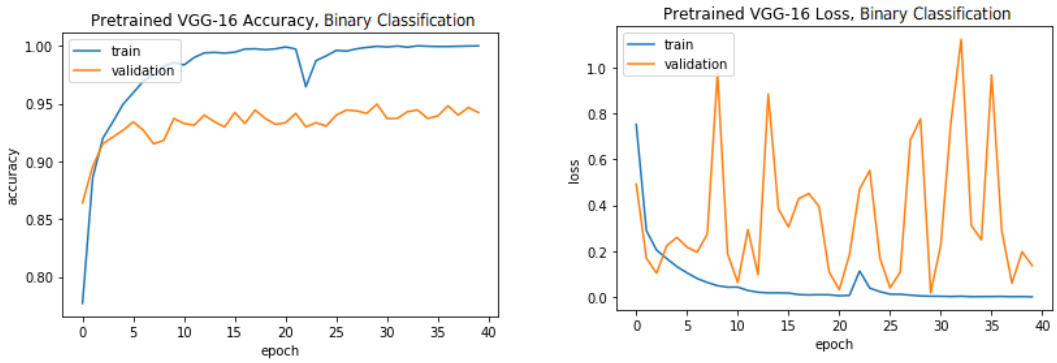

Figure 8.5: Loss and accuracy of the VGG-16 network with the initial configuration.

Table 8.4: Confusion matrix computed from the predictions of the VGG-16 network on the test set of the binary classification problem.

	Baroque	**Impressionism**	**Test Samples**
Baroque	167	40	207
Impressionism	42	165	207

more able to distinguish both styles more evenly, the network's optimizer and the learning rate were changed along with the batch size and the patience which were reduced to 16 and 5 respectively. After trying with different optimizers, the one that seemed to produce the best results was SGD with a learning rate of 0.0001. The improvement can be seen in Figure 8.6, which presents more even graphs for both accuracy and loss. The accuracy obtained in the training set with this last configuration stays similar to the initial one going up to 86.54% while the validation accuracy drops to 90.43% using a total of 6 epochs this time. The test accuracy obtained in this test also decreases to 80.19%. Although this result could be interpreted as if the performance of the model has deteriorated, an inspection of the confusion matrix in Table 8.4 indicates the model is able to recognize both kinds of pictures in a more precise way with only 6 epochs of training, which were 34 fewer epochs compared to the initial experiment.

ResNet-50 Network

For the last network's initial results, this network reached 99.8% of accuracy for the training set and a 96.2% accuracy for the validation set after 24 epochs. The testing accuracy goes up to 89.85%, which is the highest of the three networks with the initial configura-

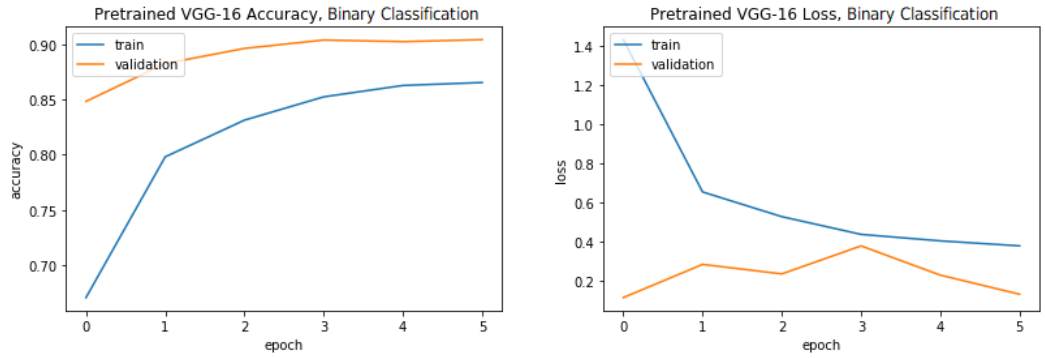

Figure 8.6: Loss and accuracy of the VGG-16 network with the final configuration.

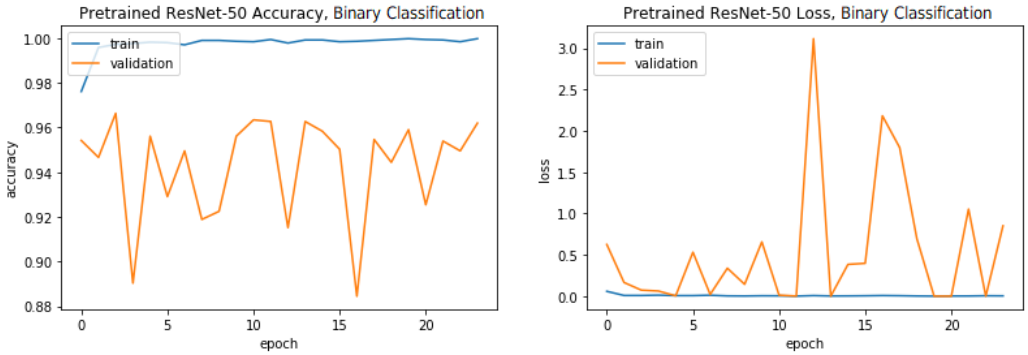

Figure 8.7: Loss and accuracy of the ResNet-50 network with the initial configuration.

tion. Table 8.5 shows the classification results and Figure 8.7 illustrates the variation of both loss and accuracy during the training process. The results of the confusion matrix are really good, even though this network also seems to classify better Baroque styled paintings. Also, the loss and accuracy plots are again spiking significantly during the 24 epochs.

Table 8.5: Confusion matrix for the binary experiment with the ResNet-50 network, first try.

	Baroque	**Impressionism**	**Test Samples**
Baroque	192	15	207
Impressionism	27	180	207

For the sake of trying to reduce the irregularities in the loss and accuracy during the training of the network, the tuning process included again reducing the batch size and the patience for the training, and changing the optimizer to Adam. The first one was again set

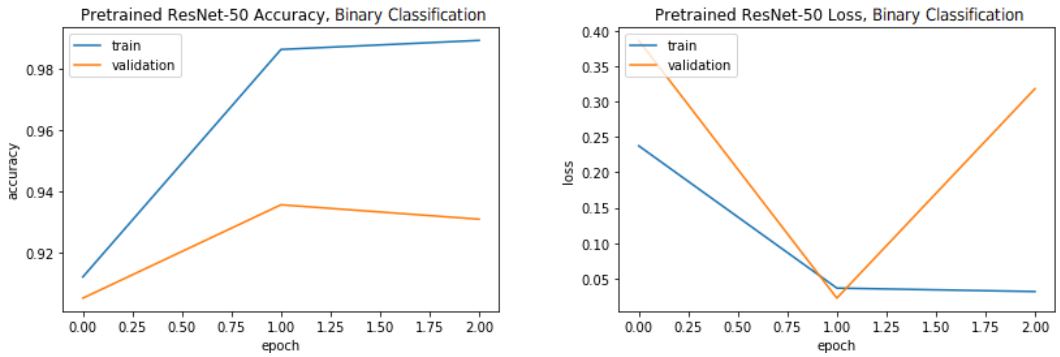

Figure 8.8: Loss and accuracy of the ResNet-50 network with the final configuration.

to 16, while the second one was reduced just to one since after several training processes a higher patience value made the network overfit. The final accuracy values for training and validation were 98.93% and 93.1% after only 3 epochs, and the test accuracy dropped to 85.26%. Confusion matrix in Table 8.6 shows that the results actually deteriorated. Although the accuracy and loss plot in Figure 8.8 looks more even, it has to be taken into account the small number of epochs used for training. After only three epochs the loss is already spiking, which could mean that if more epochs were used the spikiness would still be there. The only positive outcome obtained during this final experiment is that the epochs needed by the network to effectively classify the images have been reduced to less than 5. Therefore, in conclusion, tuning, at least as conducted for this network, may not be worth it.

Table 8.6: Confusion matrix computed from the predictions of the ResNet-50 network on the test set of the binary classification problem.

	Baroque	**Impressionism**	**Test Samples**
Baroque	183	24	207
Impressionism	37	170	207

Overall, the results obtained for this experiment were satisfactory. The high accuracy of the classifier is mostly due to the differences between the two styles: while the Baroque style is characterized by being bolder and using strong color contrasts to separate light from shadow, Impressionism can be seen as a more calm and subtle style, using complementary colors to emphasize the main ones and rarely using black. However, that does not apply to all of the paintings in each style, and that might be one of the main causes of the wrongly classified images. In this sense, the images for which the classifier fails can be

interesting because either they combine features of the two styles or they depart from the most characteristic features of the corresponding style. Figure 8.9 shows two examples of paintings that were not correctly classified probably due to the color palette confusion.

Figure 8.9: On the left, *San Lorenzo* by Francisco de Zurbarán, a Baroque painting classified as impressionist. On the right, *Self-portrait* by Giovanni Fattori, impressionist painting classified as Baroque. Misclassified paintings by VGG-16 network.

8.3.2 Multiclass Classification Results

These experiments were performed with the same three types of networks used before including changes such as the last output layer, which now has 15 more units, having 16 in total, in order to obtain the probability of a painting belonging to each of the 16 styles in each neuron. Along with that, the activation function has been changed to Softmax.

For these next tests, the training set was built with 71144 images, having an average of 4446 pictures per class; the validation set contains a total of 3200 images, 200 paintings per style, and the test set has 2022 images, with a varying amount of images per style. The accuracy was measured using two metrics: accuracy and top 5 accuracy. The last one has been added due to the number of classes found in the training set and the similarities between those, which may produce poor results in the top 1 classification accuracy.

The results for the following tests will include a classification report along with the confusion matrix due to the unbalanced amount of pictures for the test dataset. This report contains the precision, recall and F1-score calculation for the 16 classes, giving more information about the performances of the models with the multiclass dataset.

Simple Network

To train this network, the `EarlyStopping` function was disabled for this model due to the severe value changes in the validation loss metric, which may have triggered the mentioned function too early. To help regulate the learning rate a callback function called `ReduceLROnPlateau` has been added to the training, which will be triggered when the validation accuracy has not improved after a certain number of epochs. For this network, the number of epochs before activating the learning rate reduction has been set to 6 with this network, and the learning rate was reduced by 0.1.

Due to Kaggle's CPU use time restrictions, the network was trained for a total of 60 epochs with a batch size of 32, reaching a top 5 validation accuracy of 75.28% and a validation accuracy of 30.94%; the training accuracy obtained was similar to those values, having a top 5 training accuracy of 74.22% and an accuracy of 29.16%. The final test accuracy obtained reaches 24.82% and 67.15% in the top 1 and top 5 accuracy metrics respectively. Confusion matrix in Figure 8.10 shows how many images of the test set were classified correctly for each class, and Table 8.7 displays the evaluation of the performance of the model and how well each of the classes has been learned.

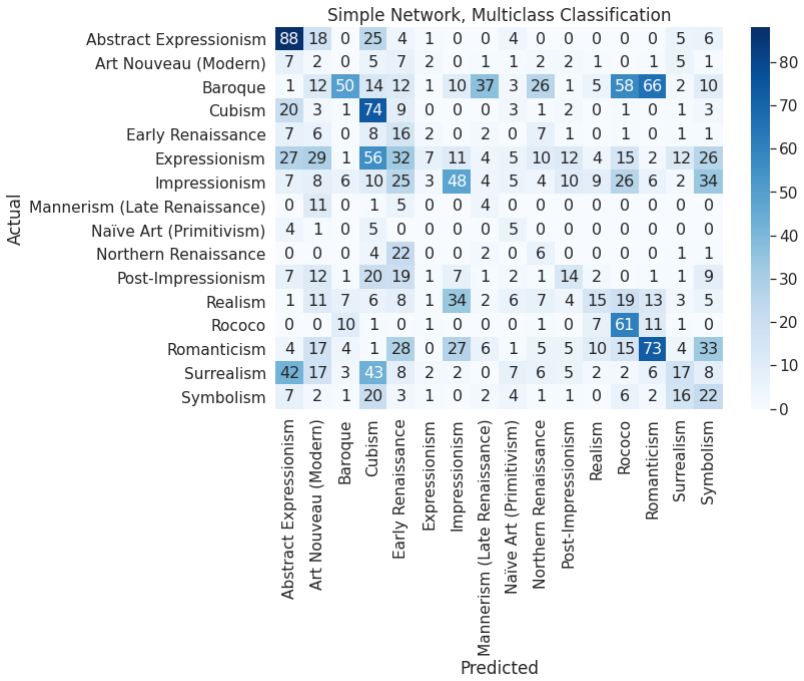

Figure 8.10: Confusion matrix for the simple network on the multiclass experiment. It can be seen that the styles that are related or happened almost at the same epoch such as Northern Renaissance and Early Renaissance are confused due to the similarities between them.

The results obtained with this network were not the best by far, but taking into account the number of classes to be learned and the simple architecture within this model using only four convolutional layers, they were not really bad either. The confusion matrix shows how each of the 2022 images has been classified. It can be noted that some styles were learned better than others, such as Abstract Expressionism, Cubism and Rococo, which also have the highest recall values. However, there are many styles that have been misclassified: 22 paintings out of 36 that belong to Northern Renaissance were assigned to the Early Renaissance class, and from 308 Baroque pictures, 58 were classified as Rococo, 66 as Romanticism style and 37 as Mannerist. The first misclassification can be due to the likeness between the two styles, since Northern Renaissance is a derivation of the Renaissance style that developed in the north of the Alps in Europe. Rococo, Romanticism and Baroque styles are related in a similar way, were Rococo can be considered as a more exaggerated and opulent evolution of Baroque and Romanticism is described as the "resurrection" of Baroque, neglecting the neoclassical painting schemes and utilizing similar color contrasts that Baroque paintings display to show emotions. The same happens with Baroque and Mannerism, which the first one came after the other, taking elements and painting styles from the latter one. Those reasons could explain why the network confused the mentioned styles.

Similarly, more modern styles such as Surrealism, Symbolism and Expressionism have been confused with each other or with other correctly classified styles like Abstract Expressionism. This result can be due to the almost simultaneous development of those styles, which could have lead to similar color palettes, image compositions and painting techniques.

The loss and accuracy functions for the training set improved steadily, while the validation set shows severe spikes during all the training in the three metrics even though the learning rate regulating function was activated several times. The charts in Figure 8.11 describe the situation mentioned.

During the last epochs of the training, the network seemed to be stuck around the loss and accuracy values mentioned before. However, this behavior may have been caused by the times the learning rate had been reduced, reaching a value so low that the training may have been slowed. Taking that into account, if the network had more epochs to train then there may have been room for improvement.

Table 8.7: Precision, Recall and F1-score for the simple network test classification results. The precision and recall values along with the confusion matrix show that the best learned classes are Cubism, Abstract Expressionism and Rococo, while the worst learned ones are Art Nouveau, Expressionism and Surrealism.

Style	Precision	Recall	f1-score	N. of paintings
Abstract Expressionism	0,3963	0,5827	0,4718	151
Art Nouveau(Modern)	0,0134	0,0541	0,0215	37
Baroque	0,5952	0,1623	0,2551	308
Cubism	0,2525	0,6271	0,3601	118
Early Renaissance	0,0808	0,3076	0,1280	52
Expressionism	0,3181	0,0276	0,0509	253
Impressionism	0,3453	0,2318	0,2774	207
Mannerism(Late Renaissance)	0,0615	0,1904	0,0930	21
Naïve Art (Primitivism)	0,1086	0,3333	0,1639	15
Northern Renaissance	0,0779	0,1666	0,1061	36
Post-Impressionism	0,2456	0,1428	0,1806	98
Realism	0,2727	0,1056	0,1522	142
Rococo	0,2990	0,6559	0,4107	93
Romanticism	0,4033	0,3133	0,3526	233
Surrealism	0,2394	0,1000	0,1410	170
Symbolism	0,1383	0,2500	0,1781	88

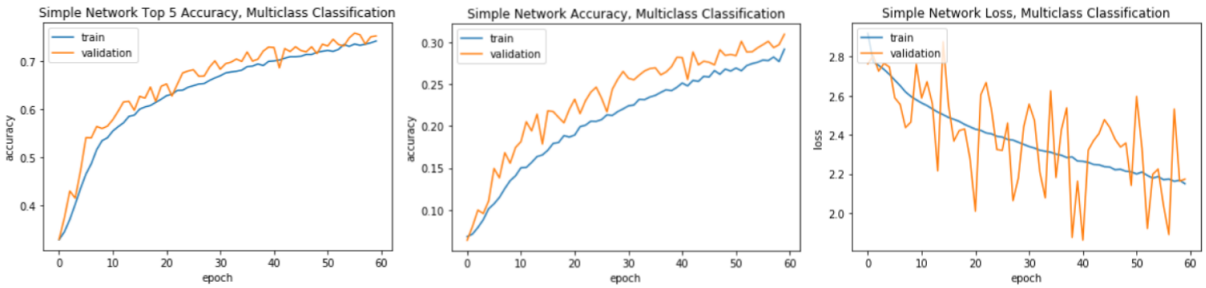

Figure 8.11: Accuracy and loss along the 60 epochs of training. The metrics computed from the training set show that learning behaved as expected. However, metrics computed on the validation set show instabilities and steep spikes specially in the validation loss.

VGG-16 Network

In this experiment, the layers of the pre-trained VGG-16 network were left frozen and not trained with the new dataset. The reason for leaving the base model not trainable was that after doing some tests with different numbers of layers left trainable, this last configuration obtained the best results and was the least overfit of all of them.

The network trained for a total of 12 epochs using the mentioned training stopping and learning rate reducing functions and using a batch size of 32. The training process for this network was really stable, with little to no spikes at all in the loss metric for both validation and training set as it can be seen in Figure 8.12, which means that the network was properly trained and the possible overfitting is small. The obtained top 5 and top 1 accuracies for the validation set scored a 79.62% and a 38.37%, while the training set values obtained a 74.33% in the top 5 accuracy and 29.9% in the top 1 accuracy. Finally, the test set values stayed between the validation and training accuracy values: the top 5 accuracy obtained was 76.3% and top 1 accuracy was 34.6%. Confusion matrix in Figure 8.13 shows how the 2022 pictures were classified.

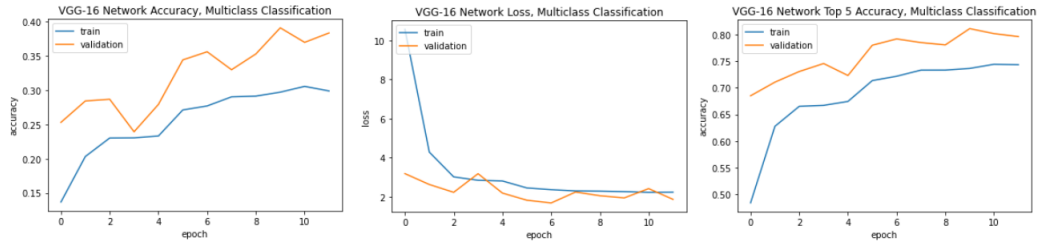

Figure 8.12: Loss and accuracy metrics' progress during training. The loss for the training set has a high initial value, but it decreases after one epoch and continues reducing its value. Validation accuracy for the two accuracy measures stays higher than the training accuracy metrics, which may be a sign of using too much dropout in the network.

The confusion matrix and the classification report in Table 8.8 show that this model has had a better performance than the previous simpler model. Again, the best learned styles are Abstract Expressionism and Cubism, but the Rococo style is not recognized as well as before this time. Along with those classes, the number of correctly classified images for classes Impressionism and Baroque also increased, as well as Surrealism styled images.

However, there have been misclassified images, which happened in a similar way as described in the analysis of the results for the previous model. In this case, even though more Baroque styled images have been correctly classified, there have been many images that were classified as Romanticist, Impressionist and Northern Renaissance. Most of these

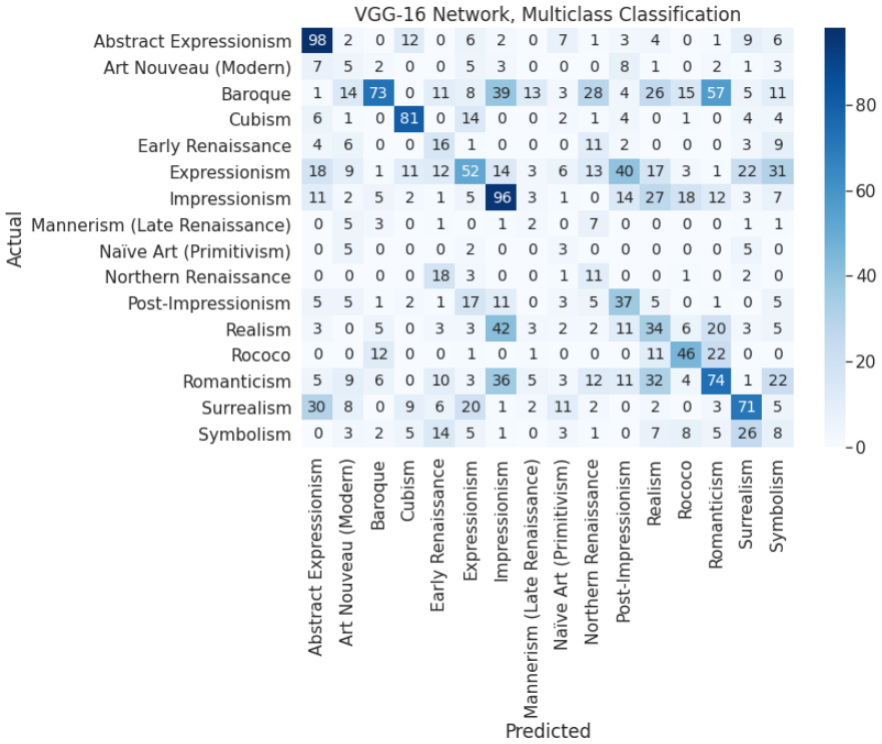

Figure 8.13: Confusion matrix for the VGG-16 network on the multiclass experiment. The number of correctly categorized has improved in comparison to the results of the previous network. However, the number of correctly classified images has decreased in classes such as Rococo, Post-Impressionism and Symbolism.

misclassifications may have happened due to the similarities between Romanticism and Northern Renaissance classes, but Baroque and Impressionist paintings are usually not so alike. However, there are Baroque paintings that have light and softer color palettes that could resemble those used in Impressionist paintings, explaining the misclassification.

Besides, the mixed classifications between modern styles seem to have been stabilized with this model since those styles have had better classification results. The only one that seems to be most difficult to correctly classify is Realism, having 42 paintings classified as Impressionist and 20 as Romanticist. Again, this could be due to the color and composition likeness, since most of the painting styles except for the more abstract ones have realist elements in their paintings. Impressionist paintings included realist landscape and portrait paintings with a more harmonic color palette because of the development of color theory in that epoch; Romanticist paintings also included very detailed and realist landscape paintings that match with a more contrasting palette. These characteristics may have misled the network into inferring wrongly most of the Realist paintings due to the resemblance they have with the paintings from other classes.

In the end, this model was the one that obtained the best results compared to the other two networks. It makes sense that the accuracy values obtained by this network were better than the ones given by the simple network since this model uses an already trained VGG-16 model as its base, which was designed to learn and differentiate 1000 different classes.

Table 8.8: Precision, Recall and F1-Score calculated with the VGG-16 classification results. An overall improvement can be seen in most of the classes.

Style	Precision	Recall	f1-score	N. of paintings
Abstract Expressionism	0,5213	0,6490	0,5782	151
Art Nouveau(Modern)	0,0676	0,1351	0,0901	37
Baroque	0,6636	0.2370	0,3493	308
Cubism	0,6639	0,6864	0,6750	118
Early Renaissance	0,1720	0,3077	0,2207	52
Expressionism	0,3586	0,2055	0,2613	253
Impressionism	0,3902	0,4638	0,4238	207
Mannerism(Late Renaissance)	0,0625	0,0952	0,0755	21
Naïve Art (Primitivism)	0,0667	0,2000	0,1000	15
Northern Renaissance	0,1170	0,3056	0,1692	36
Post-Impressionism	0,2761	0,3776	0,3190	98
Realism	0,2048	0,2394	0,2208	142
Rococo	0,4510	0,4946	0,4718	93
Romanticism	0,3737	0,3176	0,3434	233
Surrealism	0,4551	0,4176	0,4356	170
Symbolism	0,0684	0,0909	0,7800	88

ResNet-50 Network

In this experiment, this network had many modifications done in order to figure out the best configuration for this classification problem. The main changes added to the network to avoid that overfitting started by adding dense layers with different numbers of units and then increasing the dropout rate, since dropout is known to help reducing overfitting. After that, the number of frozen layers of the base model was changed, starting by freezing just 20 layers and increasing the number of untrainable layers from that starting point. The number was increased because when reducing the number of trainable layers the effective capacity of the network also reduces, and due to that the chances of overfitting are also reduced. Nevertheless, none of those mentioned changes seemed to work, and the network kept overfitting with the validation set.

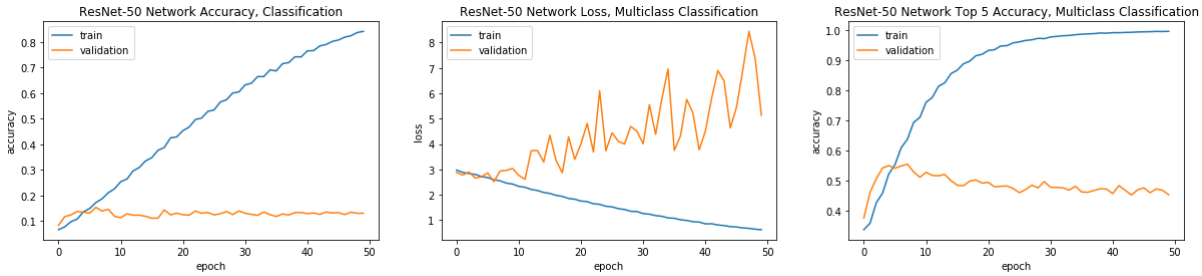

Figure 8.14: Accuracy and loss charts for the ResNet-50 model training. The validation set loss does not decrease at all during the 50 epochs. Even though the learning rate reducing function is active and decreasing that parameter, the loss value is still growing.

After researching more possible solutions, a possible answer [2] to this problem was found. The overfitting may have been due to an unexpected behavior in the batch normalization layers found in the ResNet models that caused the network to overfit when trained with datasets different to the ImageNet ones. When frozen, the batch normalization layers maintained the mean/variance of the original dataset, and when testing the network with new data that previous mean/variance stored in those batch normalization layers would affect the obtained result. One of the proposed solutions to change this behavior was to unfreeze only the batch normalization layers and leave the rest frozen. But the implementation of this solution was not installed in the available version of Keras found in Kaggle, and the setting the batch normalization layers as trainable by hand did not seem to work either since the model still overfitted massively, so this network could not be successful with this task.

However, the network was still trained and the process lasted 50 epochs, with only the learning rate reducing function was used. On the one hand, the training accuracy obtained after the 50 epochs reached an 84.2% for the top 1 accuracy metric and 99.7% for the top 5 accuracy metric. On the other hand, the validation accuracy value for the top 5 metric did not even reach a 50% accuracy and got stuck at 45.31%, staying at 13.9% for the top 1 accuracy value. The top 1 accuracy obtained for the test set was 11.4% and the top 5 accuracy barely reached 39%. The accuracy and loss charts in Figure 8.14 show that although the training set loss and accuracy metrics are performed as expected, the validation set metrics are not really improving during the 50 epochs of training, and the loss values would not stop increasing and spiking even with the learning rate being reduced several times due to the `ReduceLROnPlateau` function being triggered.

The confusion matrix in Figure 8.15 and result evaluation in Table 8.9 prove that the

[2] GitHub PR with the proposed solution: https://github.com/keras-team/keras/pull/9965

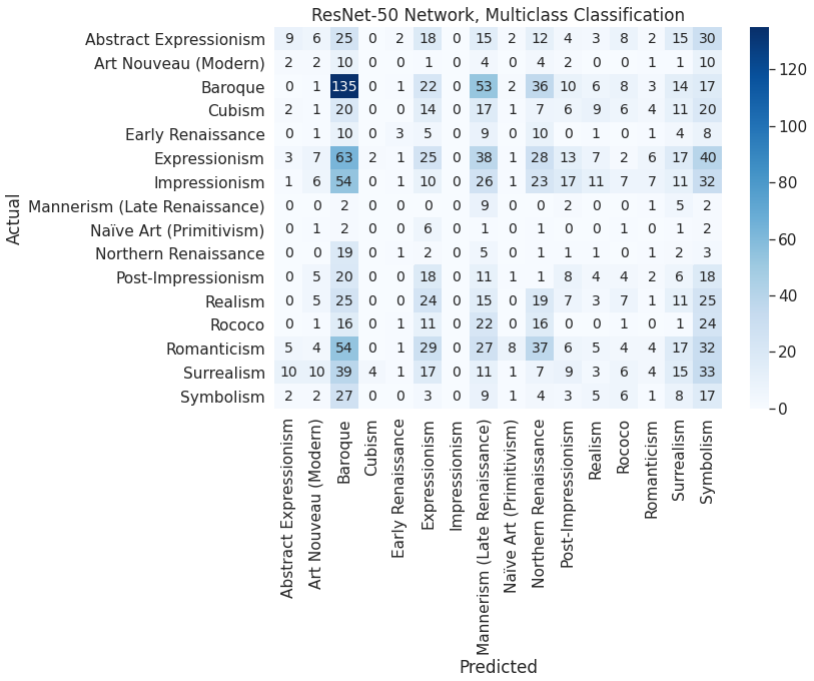

Figure 8.15: Confusion matrix for the ResNet-50 network when applied to the multiclass problem. The network is significantly biased towards the Baroque style, followed by Expressionism. But the rest of the pictures seem to have been classified in a random fashion, most of them classified as Baroque or Symbolist style.

model is overfitted since the only class that this network has somehow learned is Baroque, with 135 out of 308 images correctly classified. The rest of the classes seemed to have been mainly classified as Baroque, Expressionism, Mannerism, Northern Renaissance or Symbolist. That could mean that the network simply classified most of the pictures as the classes that it inferred that were more alike to, without really learning anything. The precision obtained for most of the classes except for Abstract Expressionism and Baroque barely reaches the 0.1 out of 1, and three styles, Cubism, Impressionism and Naïve Art, did not have any of the pictures correctly classified. The charts in Figure 8.14 show the computation of the loss and accuracy metrics for the training and validation sets. It can be seen that the validation accuracy does not improve in the 50 epochs of training, and the validation set loss even increases during the training while the training set metrics behave as expected, indicating that there is not any learning happening in the network in the training process.

In conclusion, the results obtained by this last model were surprisingly bad. Taking into account this model was initially trained for the same task as the VGG-16 and got the best scores in one of the ImageNet challenges, where it had to learn a 1000 different categories,

Table 8.9: Precision, Recall and F1-score calculated with the ResNet-50 classification results. Overall, the values show that this network did not really learn any class except for the Baroque style.

Style	Precision	Recall	f1-score	N. of paintings
Abstract Expressionism	0,2647	0,0596	0,0973	151
Art Nouveau(Modern)	0,0385	0,0541	0,0449	37
Baroque	0,2591	0,4383	0,3257	308
Cubism	0,0000	0,0000	0,0000	118
Early Renaissance	0,2500	0,0577	0,0938	52
Expressionism	0,1220	0,0988	0,1092	253
Impressionism	0,0000	0,0000	0,0000	207
Mannerism(Late Renaissance)	0,0331	0,4286	0,0614	21
Naïve Art (Primitivism)	0,0000	0,0000	0,0000	15
Northern Renaissance	0,0049	0,0278	0,0083	36
Post-Impressionism	0,0909	0,0816	0,0860	98
Realism	0,0517	0,0211	0,0300	142
Rococo	0,0167	0,0108	0,0131	93
Romanticism	0,1053	0,0172	0,0295	233
Surrealism	0,1079	0,0882	0,0971	170
Symbolism	0,0543	0,1932	0,0848	88

the outcome of this model should improve or at least match the model with the VGG-16 base. However, after many different experiments trying different configurations, all the tests ended up having a really good accuracy for the training set, but a poor accuracy for the validation set.

8.3.3 Photograph and Painting Classification Results

This problem consisted in training a neural network so that it can become able to distinguish a painting from a photograph, rating how *real* a painting can be, or how *artistic* a photograph is depending on the probability value obtained. To address this task, the neural networks used will be the same as for the art painting binary classification problem, but using the configurations that obtained the best results for that problem. Therefore, the experiments will only be performed with those network settings. The architecture of the three networks is the same that was used for the first binary classification problem. The `EarlyStopping` function has also been included for these tests.

The paintings including in the training set are the same as the ones used in the previous task, while the photos have been taken from the flickr30k dataset. The dimensions of the images from this dataset have been set to 224x224 to fit the input size of the three networks.

This time, the training set contains 45760 pictures, 22880 images for each category; the validation set includes 5092 images, having 2546 per class. Finally, the test set comprises 12714 pictures, where 6357 are photographs and the other half are paintings.

Simple Network

This network took 13 epochs out of 100 to complete the training that stopped when the `EarlyStopping` function was triggered. The obtained accuracy for the training was 98.8% and for the validation set 93.51%. The test set accuracy reached 95.6%. Charts in Figure 8.16 show the steady growth of the accuracy for the validation and training sets, but the validation loss shows a few spikes, even though that in the end it reaches a value similar to the training loss. Overall, the values of those metrics were similar to the ones obtained in the first binary art painting classification problem using this network, but with a significant improvement in the accuracy values. However, this could be due to the size differences of the datasets, having a rather larger dataset for this experiment.

The confusion matrix in Table 8.10 shows the results obtained for the test set. The classification results are mostly even for both classes, but it can be seen that some paintings have been classified as photographs, showing that there might be a slight bias towards that category. Some examples of misclassified paintings are shown in Figure 8.17.

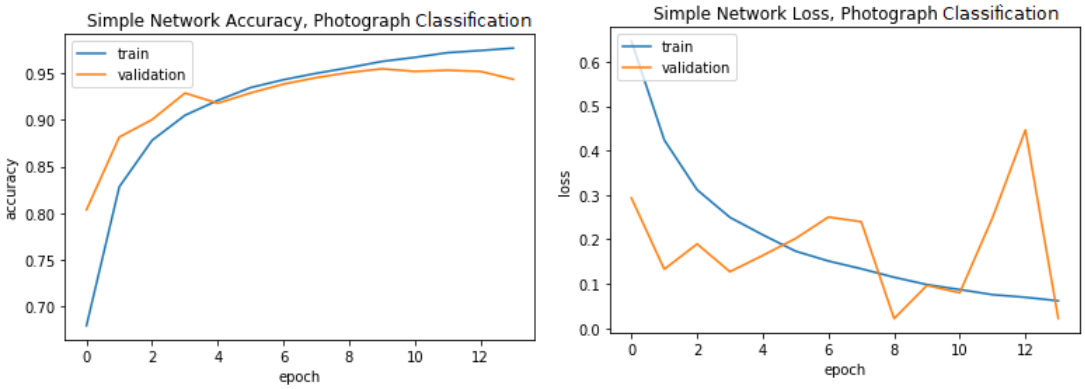

Figure 8.16: Loss and Accuracy charts for the Simple Network for the Photograph classification problem. Accuracy curve grows steadily for both validation and train sets, but the loss function for the first set shows spikes that in the end converge to a value lower than the train loss.

Table 8.10: Confusion matrix computed from the predictions of the simple network on the test set of the photograph and painting classification problem. The results show a slight bias towards the photograph class.

	Painting	**Photograph**	**Test Samples**
Painting	6044	313	6357
Photograph	246	6111	6357

VGG-16 Network

The epochs needed to train this pre-trained network were 13, the same as the amount needed for the simple network. However, even though the same amount of epochs were needed, the accuracy at the beginning of the training was much higher having a starting value of around 90% for the training set and above that value for the validation set. As Figure 8.18 depicts, the accuracy chart shows that the learning was even for both validation and training sets until the latter epochs, where the accuracy value for the validation set starts to decrease. The loss values for both sets decreased as well, but the validation loss shows some steep spikes especially at the end, just where the accuracy for that set starts to deteriorate. This behavior could indicate that the network has been trained for more epochs than needed.

The confusion matrix obtained for this experiment shows that the number of wrongly classified images for both classes is almost the same, just as it happened in the first binary classification problem using this network as represented in Table 8.4. The accuracy reached for the validation set was 98.04%, the value for the training set was 98.7%, and

Figure 8.17: Paintings misclassified as photographs by the simple network. From left to right, top to bottom: *Crystals*, by George Saru; *A Fair*, by Borís Mijáilovich Kustódiev; *St. Sergius of Radonezh*, by Mikhail Nesterov; *Clearing Sunset*, by Frederick Childe Hassam

Table 8.11: Confusion matrix computed from the predictions of the VGG-16 network on the test set of the photograph and painting classification problem.

	Painting	**Photograph**	**Test Samples**
Painting	6239	118	6357
Photograph	111	6246	6357

finally the accuracy obtained for the test set was 98.19%, really close to the previous values, which indicates that the network was properly trained and can effectively distinguish photographs from paintings to some extent. Some examples of correctly and wrongly classified images are illustrated in Figures 8.19 and 8.20.

ResNet-50 Network

This is the network that needed the least amount of epochs to finish the training, reducing the number from 13 epochs to 4. The accuracy reached for the training and validation test was 98.68% and 96.8% respectively, and the score for the test set was 96.36%. Again, the training increased steadily for both training and validation scores, but the loss chart that can be seen in Figure 8.21 shows otherwise for the validation loss, since as it happened

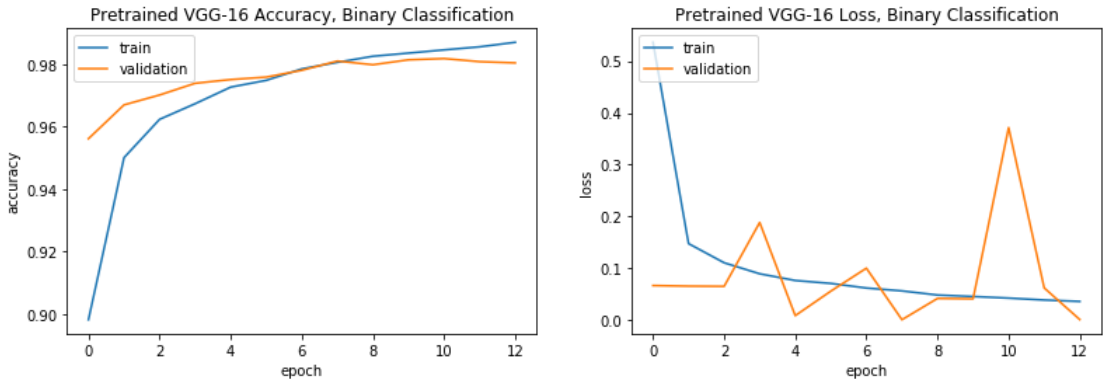

Figure 8.18: Loss and Accuracy charts for the VGG-16 network with the Photograph experiment.

Figure 8.19: Example of two correctly classified pictures using the VGG-16 network: on the left, a family picture; on the right, *Nessus and Deianeira*, a Symbolist painting by Arnold Böcklin.

with the previous experiments with this network spikes again within only 4 epochs. It can also be noticed that the training accuracy grows faster and reaches a higher value than the validation accuracy, meaning that this network can be overfitted or is less effective when generalizing with unseen paintings. This fact can be verified by checking Table 8.12, where it is shown that more paintings than photographs are assigned the wrong class, an outcome similar to the results produced by the simple network. However, the number of misclassified images is lower in this experiment. Figure 8.22 shows four misclassified examples.

After checking the three confusion matrices obtained, it can be noted that the three networks have misclassified more paintings as photographs than the other way around. This

Figure 8.20: On the left, *Portrait of Artist's Daughters Alexandra and Felisata* by Alexey Venetsianov, a realist painting misclassified as a photograph; On the right, a person surrounded by snow, classified as a painting by VGG-16 network.

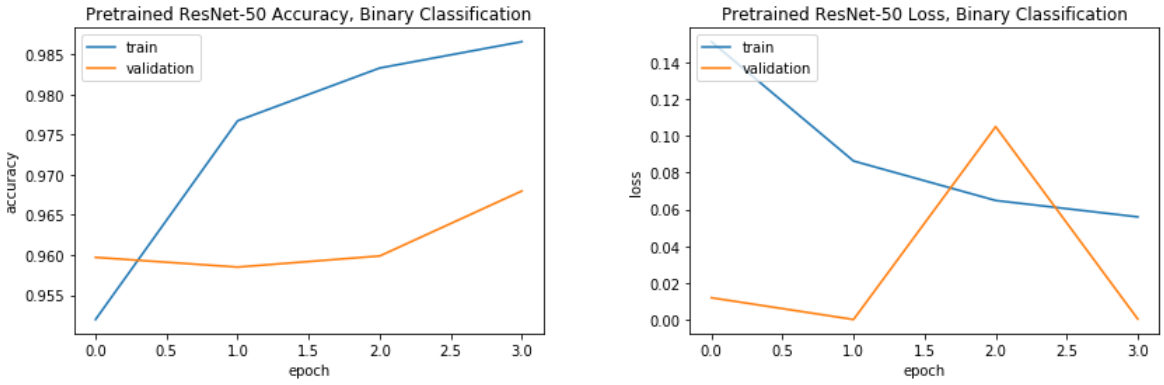

Figure 8.21: Loss and Accuracy charts for the VGG-16 network with the Photograph experiment.

may have happened due to the number of realistic looking portraits and landscape paintings that could have a strong resemblance with actual real images. The photographs that were classified as paintings, on the other hand, may be those that had blurry sections or strong color variations similar to some painting styles.

Figure 8.22: Four misclassified images obtained with the ResNet-50 network. On the first row, two paintings classified as photographs: *The Medicine Man No. 2* by Charles M. Russell and *Peaceable Kingdom* by Edward Hicks. On the second row, two photographs classified as paintings.

Table 8.12: Confusion matrix computed from the predictions of the ResNet-50 network on the test set of the photograph and painting classification problem.

	Painting	**Photograph**	**Test Samples**
Painting	6104	253	6357
Photograph	209	6146	6357

CHAPTER 9

Neural Style Transfer

The neural style transfer algorithm [38] is a method that, given a base image to transform and a style reference picture, combines both images in such a way that the content of the base image is represented with the style of the reference image.

The goals of studying this algorithm were to investigate the ability of the neural networks covered and tested in this thesis for implementing the neural transfer, to use the art painting database with this purpose, and to incorporate to the graphical user interface an additional functionality that implements style-transfer.

The original neural network used for this task is the VGG-16 network trained with the ImageNet weights. For this task, the network to be used will be the VGG-16 trained for the painting and photograph task, which has been trained with a great number of paintings compared to the VGG-16 used for the Baroque and Impressionism classification problem.

To obtain the combined image, the algorithm tries to minimize two different loss functions at the same time: the content loss and the style loss. Those loss values are calculated using different layer combinations. The first loss function is represented in Equation 9.1,

$$\mathcal{L}_{content}(\vec{p},\vec{x},l) = \frac{1}{2}\sum_{i,j}(F^{l}{}_{\mathrm{ij}} - P^{l}{}_{\mathrm{ij}})^2 \tag{9.1}$$

where $\vec{p}$ and $\vec{x}$ are the original and the generated image with their respective feature representation P^l and F^l in layer l, and the representation of the second loss function is found in Equation 9.3.

$$E_l = \frac{1}{4N^2{}_l M^2{}_l} \sum_{i,j} (G^l{}_{ij} - A^l{}_{ij}) \tag{9.2}$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \tag{9.3}$$

being $\vec{a}$ and $\vec{x}$ the original image and the generated image, A^l and G^l their respective style representation in layer l. w_l represents the weighting factors of the contribution of each layer to the total loss, and E_l is the contribution of layer l to the total loss, which is calculated in Equation 9.2. Finally, the N^l and M^l represent the total number of feature maps in layer l and the height times the width of that feature map.

The content loss measures how similar the input base image's features and the combined image's features are, so the lower that value is, the more preserved are the original image's details. To obtain that information, the deepest layers of the network are preferred since those layers are the ones that learn most of the high level feature details and overall content information. In this case, the layer `block4_conv2` is used, because if the last block's convolutional layer were used the obtained results would be completely abstract and the original image's content would be lost. An example can be seen in Figure 9.1.

The style loss is similar to the previous loss because it also compares the differences between the generated image and the style reference image feature maps. However, those comparisons are done differently. A Gram matrix is calculated with each of the feature maps, and with that the correlation between each feature map is calculated, obtaining which is the tendency of the features for every map with the style reference image. This is calculated with the most shallow convolutional layer in each of the convolutional blocks contained within the network, which for this network are `block1_conv1`, `block2_conv1`, `block3_conv1`, `block4_conv1` and `block5_conv1`. That last layer and overall the last block has been trained differently in the VGG-16 networks used in this project, but the rest of the layers still retain the ImageNet dataset weights, so the output image may not be so different from the output of the VGG-16 with the original weights.

With those two losses, a third loss will be calculated, the general loss, which is the weighted sum of the previous ones. The weight for each of the losses is arbitrary, and it affects how the content of the original image or the style of the reference picture is conserved in the combined output. This metric is the loss to be optimized, and its mathematical representation is found in Equation 9.4, where the α and β symbols represent the weights assigned for the content and style loss respectively. In each iteration the algo-

(a) Original image

(b) Style reference

(c) Layer block3_conv2

(d) Layer block4_conv2

(e) Layer block5_conv2

Figure 9.1: Results obtained using different layers to capture the original image's features after 15 iterations. It can be seen that when using the layer that is nearest to the output layer the original image's content is completely lost (**e**). However, using the layer from a block that less deep as block 3, the content of the original picture is more conserved, but the details of the painting are less visible (**c**). The layer that combines both images in a more balanced way is the second layer of the fourth block, obtaining both the style reference and the base image's content recognizable (**d**). The style reference painting is *The Starry Night* by Vincent van Gogh.

rithm executes, the content and style loss will be computed and used to calculate the total loss, which will be back-propagated and minimized with an optimization method, which in this case is the L-BFGS [48] algorithm.

$$\mathcal{L}_{total}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{content}(\vec{p},\vec{x}) + \beta\mathcal{L}_{style}(\vec{a},\vec{x}) \tag{9.4}$$

Figure 9.2 shows the results obtained with the VGG-16 fully trained with the ImageNet weights, the VGG-16 trained for the Baroque and Impressionism classification task and finally photograph and classification task. The outputs look really similar since the only affected layer for this task is the `block5_conv1` layer. The results may have varied more if

the proportion of the retrained layers were higher. Finally, Figure 9.3 presents an example of the results that can be obtained with this algorithm using paintings of different styles.

(a) Original image

(b) Style reference

(c) VGG-16 with ImageNet dataset

(d) VGG-16 with binary painting dataset.

(e) VGG-16 with photograph and painting dataset.

Figure 9.2: Different outputs obtained with the three differently trained VGG-16 networks. The results are really similar since only the `block5_conv1` layer has been changed. The style reference painting is *The Swing* by Jean-Honoré Fragonard.

Original image

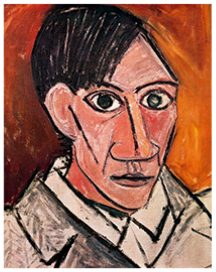

Figure 9.3: Results obtained with different paintings. From top to bottom, *The Milkmaid* by Johannes Vermeer, *Self Portrait* by Pablo Picasso, *The Scream* by Edvard Munch and *Meules, milieu du jour* by Claude Monet.

CHAPTER 10

Graphical User Interface

In order to test the networks trained in an easier way, a graphical user interface has been developed so that coding skills or changing code lines are not essential to experiment with new images. That way, anyone can use and try the neural networks effortlessly.

The GUI has been developed using R language and the Shiny [1] library for web interface and interactive applications development with the Shinythemes package for aesthetic purpose. The coding environment has been R Studio. With that in mind, having the software and the resources mentioned installed is necessary to use the application. The code created as well as the weights and parameters used can be found in the GitHub repository [2] linked at the end of the page.

The application is divided in three main tabs: the first one handles the style classification tasks, the second one is in charge of the photograph and painting classification tests and the last one performs the style transfer. A picture of the main interface is shown in Figure 10.1.

The three tabs are built in a similar way: on the left side there is a side menu which contains elements to load the images, select the network to use and a button to start the desired task; the rest of the space is used for displaying the image shown and presenting the obtained results beneath that image. The following sections describe those tabs and how the results are displayed in a more precise way.

[1] RStudio Shiny web page: https://rstudio.com/products/shiny/
[2] Github repository: https://github.com/koishus/TFGInterfaz

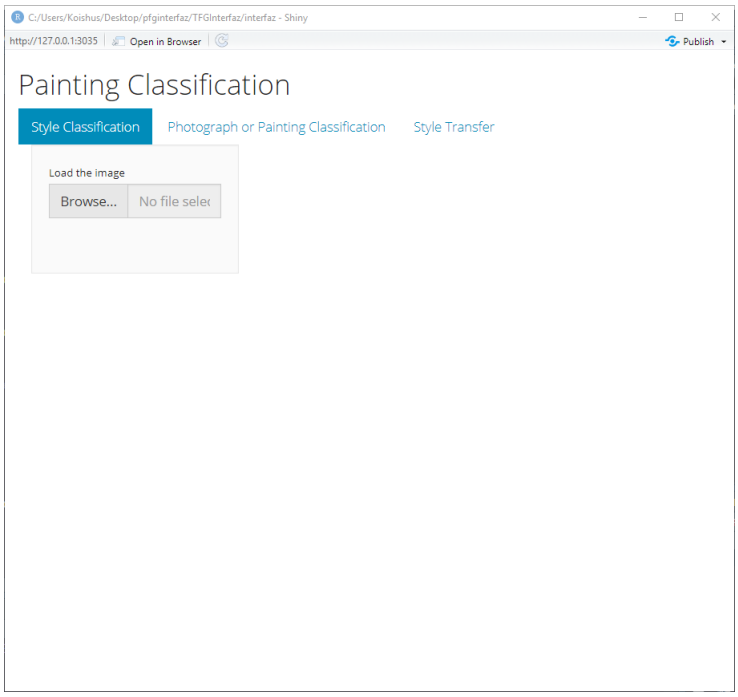

Figure 10.1: Main window.

10.1 First tab: Painting Style Classification

This tab performs the painting style classification of the loaded image. That picture will be shown in the center of the main space of the tab, and beneath that, a bullet-style list will show the top 5 possible styles inferred along with the obtained probability value for the given image. Figure 10.2 presents an example of a classified painting.

10.2 Second tab: Photograph and Painting Classification

This tab works similarly to the previous one. Along with the obtained classification label, a value that represents the probability value obtained has been added. The closer that value is to 1, the more likely it is that the loaded image is a photograph, and the other way around if the value is closer to 0. If the image were classified as a painting, the list of the styles to which the image could belong to is also shown. An example of those results can be seen in Figure 10.3.

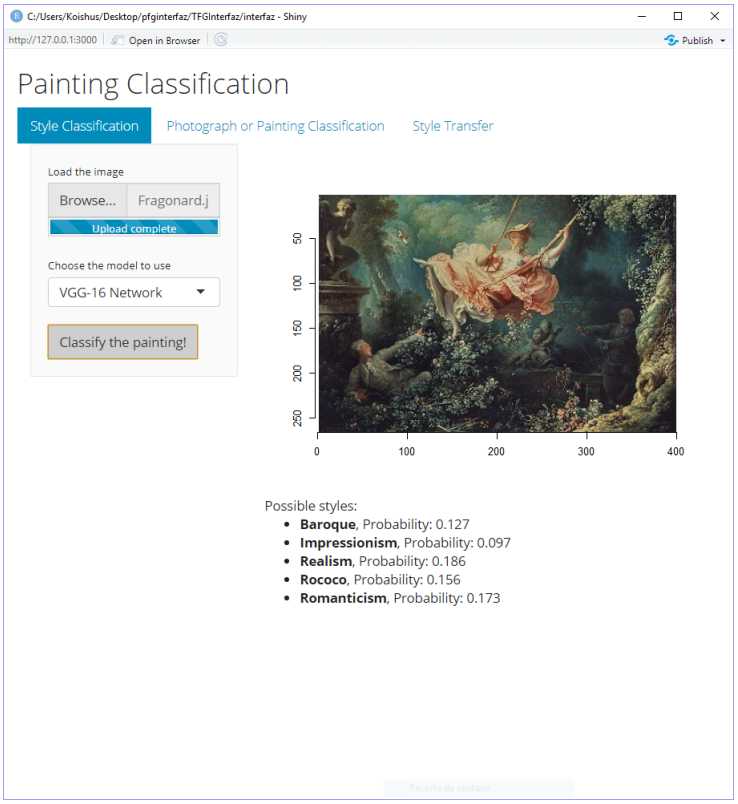

Figure 10.2: Style classification window, with the results displayed.

10.3 Third tab: Style Transfer

This last tab is the most different of the three. The sidebar menu has two file loading buttons, and also includes a numeric input to specify how many iterations of the algorithm are going to be executed. The main panel contains three images: the first two belong to the loaded images, and the third, located under the others, shows the final combination of both images. Since the process can be long, a progress bar has been added to indicate which of the iterations is executing at the moment. This tab's appearance is presented in Figure 10.4.

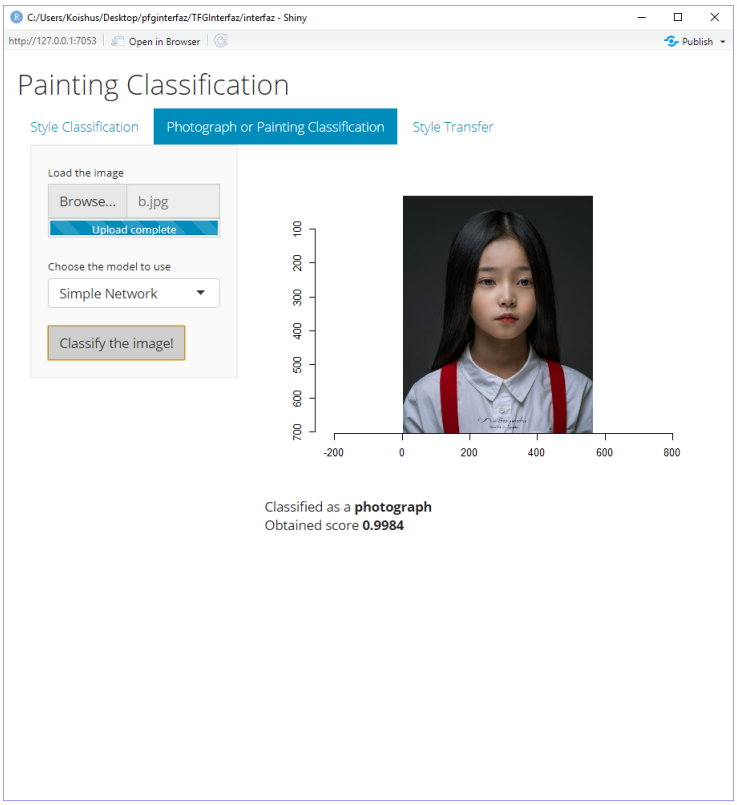

Figure 10.3: Photograph of painting classification window, with the results and the probability value.

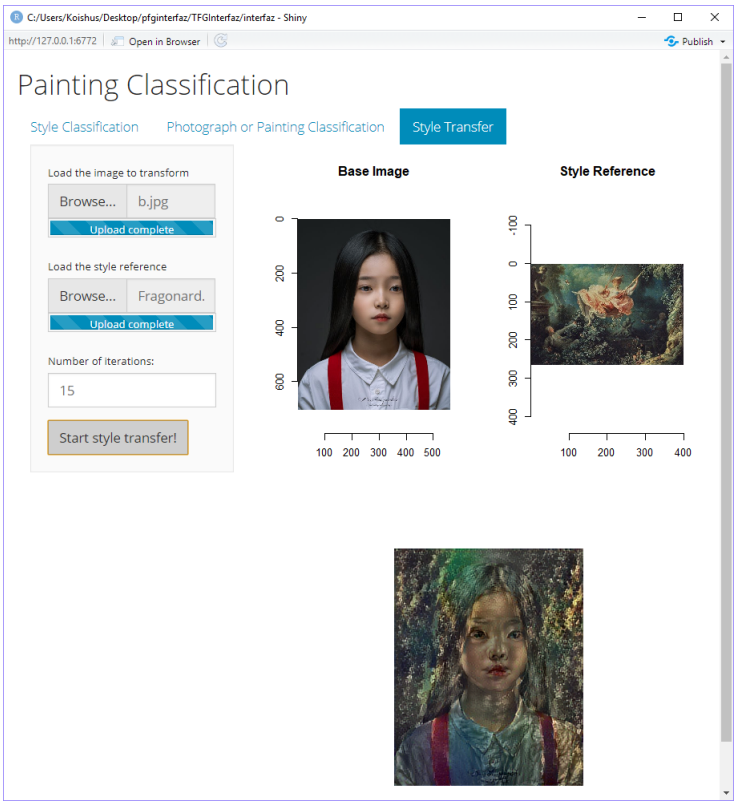

Figure 10.4: Style transfer window, with the combined image displayed below the base image and the style reference image.

CHAPTER 11

Conclusions

As mentioned in the introduction, the main goal of this project was to build a deep neural network to determine the style of a painting. That objective was achieved and tested with three different networks with different levels of complexity, and three experiments were done with each of them to see which one produced the best performance.

After finishing all the classification experiments, it can be seen that the one network that performed the best in both binary classification tests, and in the multiclass classification experiment was the VGG-16 based model. This network used the same number or less of epochs to train and obtained really good results. However, the simple network performed better than expected, and obtained high quality results with the multiclass classification experiments even though it had a really simple architecture. Nevertheless, those results may have been improved with better hardware equipment and more training epochs. Finally, the ResNet-50 could have obtained a great accuracy with the multiclass experiment too if the malfunctions mentioned would not have happened, but still, it had a remarkable performance with the other two image classification problems.

To improve the results obtained, on the one hand having better hardware equipment that is not dependent of use restrictions unlike the Kaggle platform would be recommended, since the time limitations have affected in training parameters such as batch size and epochs used, along with the time of the day in which the network training had to be started, which some times took a really long time.

On the other hand, the number of paintings of each style was not really balanced even after

discarding those categories with a small number of paintings. Some of the classes had to go through data augmentation processes, and in the end an average of 4500 pictures per style was obtained. However, having 16 different categories, the final number of images may have not been enough to reach the main goal. If more painting data was available, probably the results would have been improved.

Besides, the machine learning task for which the best results were obtained was the photograph and painting classification problem, since the average accuracy obtained for the test set was 96.71%. However, the high accuracy values obtained with the three models may have been caused by the number of images that composed the dataset, where each of the classes contained more than 20000 training pictures, which is almost five times the average number of images per class in the dataset used in the multiclass experiment.

Finally, the style transfer tests with one of the tested networks was successful, but the overall performance was not the best since the execution times were too long and the image size was restricted to rather smaller image sizes. Nevertheless, those performance issues could be solved using more powerful hardware and GPU.

However, in spite of these obstacles and limitations, the overall results obtained were better than expected, thus it can be said that the main goal and the other two machine learning tasks addressed were successful.

Bibliography

[1] L. Shamir and J. A. Tarakhovsky, "Computer analysis of art," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 5, no. 2, p. 7, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[4] D. Dong, H. Wu, W. He, D. Yu, and H. Wang, "Multi-task learning for multiple language translation," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1723–1732, 2015.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[6] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[7] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 163–168, IEEE, 2011.

[8] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks*, vol. 1, no. 2, pp. 119–130, 1988.

[9] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, IEEE, 2015.

[10] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *Advances in neural information processing systems*, pp. 656–664, 2012.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[14] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, pp. 396–404, 1990.

[15] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[16] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *arXiv preprint arXiv:1808.03314*, 2018.

[17] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," pp. 545–552, 2009.

[18] M. G. Quiles and R. A. F. Romero, "A computer vision system based on multi-layer perceptrons for controlling mobile robots," vol. 2, pp. 661–668, 2005.

[19] E. Cetinic and S. Grgic, "Automated painter recognition based on image feature extraction," in *Proceedings ELMAR-2013*, pp. 19–22, IEEE, 2013.

[20] G. Holmes, A. Donkin, and I. H. Witten, "Weka: A machine learning workbench," in *Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference*, pp. 357–361, IEEE, 1994.

[21] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.

[22] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[23] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[24] R. E. Schapire, "A brief introduction to boosting," in *Ijcai*, vol. 99, pp. 1401–1406, 1999.

[25] N. Orlov, L. Shamir, T. Macura, J. Johnston, D. M. Eckley, and I. G. Goldberg, "Wnd-charm: Multi-purpose image classification using compound image transforms," *Pattern recognition letters*, vol. 29, no. 11, pp. 1684–1693, 2008.

[26] S. Agarwal, H. Karnick, N. Pant, and U. Patel, "Genre and style based painting classification," in *2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 588–594, IEEE, 2015.

[27] T. Lindeberg, "Scale invariant feature transform," 2012.

[28] I. Sikirić, K. Brkić, and S. Šegvić, "Classifying traffic scenes using the gist image descriptor," *arXiv preprint arXiv:1310.0316*, 2013.

[29] X. Wang, T. X. Han, and S. Yan, "An hog-lbp human detector with partial occlusion handling," in *2009 IEEE 12th international conference on computer vision*, pp. 32–39, IEEE, 2009.

[30] V. Sebastian, A. Unnikrishnan, K. Balakrishnan, *et al.*, "Gray level co-occurrence matrices: generalisation and some new features," *arXiv preprint arXiv:1205.4831*, 2012.

[31] X. Zhang, B. A. Wandell, *et al.*, "A spatial extension of cielab for digital color image reproduction," in *SID international symposium digest of technical papers*, vol. 27, pp. 731–734, Citeseer, 1996.

[32] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[33] R. S. Arora and A. Elgammal, "Towards automated classification of fine-art painting style: A comparative study," pp. 3541–3544, 2012.

[34] A. E. Abdel-Hakim and A. A. Farag, "Csift: A sift descriptor with color invariant characteristics," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1978–1983, Ieee, 2006.

[35] K. Van De Sande, T. Gevers, and C. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1582–1596, 2009.

[36] A. Lecoutre, B. Negrevergne, and F. Yger, "Recognizing art style automatically in painting with deep learning," in *Asian conference on machine learning*, pp. 327–342, 2017.

[37] G. Gando, T. Yamada, H. Sato, S. Oyama, and M. Kurihara, "Fine-tuning deep convolutional neural networks for distinguishing illustrations from photographs," *Expert Systems with Applications*, vol. 66, pp. 295–301, 2016.

[38] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *CoRR*, vol. abs/1508.06576, 2015.

[39] A. Selim, M. Elgharib, and L. Doyle, "Painting style transfer for head portraits using convolutional neural networks," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 4, pp. 1–18, 2016.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[41] J. Wu, "Introduction to convolutional neural networks," *National Key Lab for Novel Software Technology. Nanjing University. China*, 2017.

[42] S. Hijazi, R. Kumar, and C. Rowen, "Using convolutional neural networks for image recognition," *Cadence Design Systems Inc.: San Jose, CA, USA*, 2015.

[43] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[45] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," in *Proceedings of the IEEE international conference on computer vision*, pp. 2641–2649, 2015.

[46] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2015.

[47] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.

[48] J. L. Morales, "A numerical study of limited memory bfgs methods," *Applied Mathematics Letters*, vol. 15, no. 4, pp. 481–487, 2002.