



BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
FACULTY OF NATURAL SCIENCES

BSc THESIS

CLASSIFICATION OF BOSCH-RELATED PAINTINGS USING CONVOLUTIONAL NEURAL NETWORKS

ESZTER NAGY

Supervisors:

Marcell Nagy

Ph.D. Student, Department of Stochastics

Budapest University of Technology and Economics

Roland Molontay

Associate professor, Department of Stochastics

Budapest University of Technology and Economics

Contents

Abstract	3
Kivonat	4
1 Introduction	5
1.1 Hieronymus Bosch	5
1.2 The trend of art copying and the confusion	6
1.2.1 Implementing Machine Learning as a Solution for Comprehensive Art Analysis	6
1.3 Related work	8
1.4 The goal	9
2 Convolution neural networks in fine art classification	10
2.1 Artificial neural networks	11
2.2 Convolution neural networks	12
2.3 Other important tools	13
2.4 Transfer learning	14
2.5 Introduction to networks commonly used in transfer learning	15
2.6 EfficientNet	16
3 Experimental Setup and Data	18
3.1 Programming Environment	18
3.2 The classes	19
3.3 The datasets	19
3.4 Data augmentation and dataset ratios	20

4	Training and results	23
4.1	The training process	23
4.2	The significance of image sizes, resolution, and class sizes	25
4.2.1	Small image size	25
4.2.2	Re-cropping to correct size	25
4.2.3	Same class sizes	26
4.3	Adding a layer	27
4.3.1	Finding the best epoch number	28
4.4	Evaluating the results on the miscellaneous set	31
4.5	Examining the Misclassifications in Bosch Paintings	34
5	Conclusion	37
5.1	Future work and outlook	38
	Acknowledgment	39
	Figures	40
	Codes	41
	Tables	41
	Bibliography	44

Abstract

This thesis investigates the application and limitations of pre-trained convolutional neural networks (CNNs) in the classification of artwork, specifically focusing on the works of Dutch painter Hieronymus Bosch and his imitators. As the demand for digitized artwork grows, it is essential to develop reliable, non-human assistance capable of accurately identifying and expediting the classification process. This study utilized CNNs initially trained on general image datasets and further refined using Bosch and follower datasets labeled by art experts. The results demonstrated that deep learning can effectively distinguish between such similar datasets. However, the uncovered limitations indicate that further refinements are required for optimal performance.

Kivonat

Szakdolgozatomban az előretanított konvolúciós neurális hálózatok (CNN) alkalmazását és korlátait vizsgálom a szépművészeti alkotások osztályozásának terén, különös tekintettel a holland festő, Hieronymus Bosch és követőinek műveire. A digitalizált műalkotások iránti növekvő igény miatt elengedhetetlen megbízható, nem emberi segítség kifejlesztése, amely képes pontosan elvégezni és gyorsítani az osztályozási folyamatot. E tanulmányban általános képadatbázisokon előretanított CNN-ek teljesítményét finomítottam szakértők által felcímkézett, Bosch és követőinek munkáit tartalmazó adathalmazzal. Az eredmények azt mutatták, hogy a mélytanulás valóban hatékonyan képes különbséget tenni ennyire hasonló adathalmazok közt. Azonban az is kiderült, hogy további finomításokra van szükség az optimális teljesítmény érdekében.

[...] his pictures are by no means absurdities but rather [...] books of great wisdom and artistic value [...] They are a painted satire on the sins and ravings of man [...] The difference which, to my mind, exists between the pictures of this man and those of all others, is that the others try to paint the man as he appears on the outside, while he alone had the audacity to paint him as he is on the inside.

Fray José de Sigüenza about
Hieronymus Bosch (1605)
translation found in [14]

1

Introduction

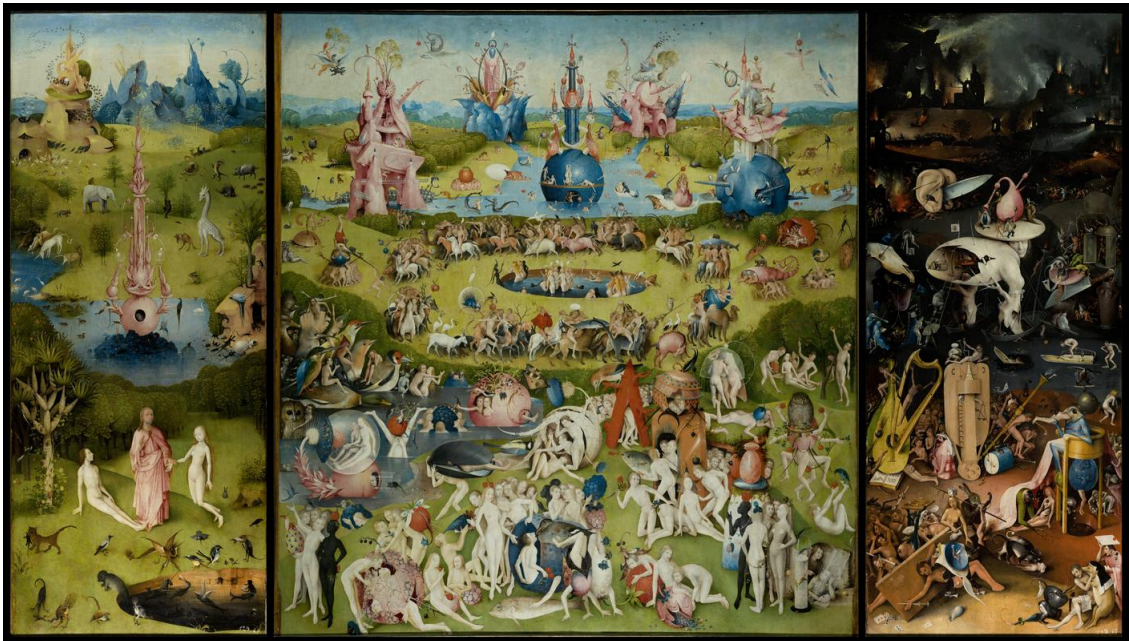


Figure 1.1: *The Garden of Earthly Delights triptych (between 1490 and 1510)*

1.1 Hieronymus Bosch

Hieronymus Bosch (birth name: Jheronimus van Aken) was born in the Dutch city of 's-Hertogenbosch around 1450. We hardly know anything about his personal life; the primarily written documents mentioning him are the account books of the local order of the

Illustrious Brotherhood of Our Blessed Lady, which was an organization dedicated to Virgin Mary. Most of Bosch’s male family members were also members of the Brotherhood, which probably influenced Bosch to stay in close touch with the group during his lifetime. The organization was involved in a lot of charity work and its focus on various religious rituals aiming to combat sins is believed to have inspired the sin-connected allegories in his paintings [3].

Like the majority of artists at the time, he created paintings based on purchase orders, but it did not stop him to thrive in artistic freedom. His aim was to capture religious beliefs with a high focus on sin and damnation, and quickly became known for his otherworldly creatures and hell-like scenes, such as the one in Figure 1.1. It is safe to say that he was an art pioneer of his time; before him, fine art was more subtle in terms of character depiction and scenes of physical suffering were usually less disturbing. Some look at his painting and may say he was crazy; but he was very popular at his time and today he is often considered the precursor of modern surrealism [6].

1.2 The trend of art copying and the confusion

In the 15th century, it was common to honor a talented artist to copy his work or paint in his unique style. They are generally called followers because most of the time their names were not even present in their paintings. Throughout the centuries, this anonymity and the lack of documentation led to some confusion about authorship. Which one is the original and which one is a good imitation? Bosch is no different in this regard: the number of paintings made by his followers is more than that of his own. To make matters worse, the imitation of his art already started during his lifetime [15]. Luckily, there are more and more reliable methods to distinguish between originals and copies — and one of them is machine learning.

1.2.1 Implementing Machine Learning as a Solution for Comprehensive Art Analysis

This section is an overview of machine learning in art, emphasizing its necessity and underscoring the relative ease of utilization of convolution neural networks (CNNs) compared to other methods. Throughout this summary, we follow the comprehensive work of Wang, Kandemir, and Li on the subject [20].

An expert can gain a substantial understanding of the facts and relationships among various stylistic movements and artists. However, it is humanly impossible for one to have sufficient knowledge about all of them, especially considering the immense time required for in-depth analysis of even a smaller collection of artworks. As such, computer vision analysis has emerged as a crucial tool for accelerating and parallelizing art research.

Art historians have traditionally used a variety of tools for fine art analysis, including raking light photography, ultraviolet fluorescence illumination, infrared imaging, cross-section paint and varnish analysis, microscopic analysis, chemical composition analysis, X-ray imaging, multi-spectral imaging, laser 3D scans, and synchrotron induced X-ray fluorescence mapping, among others. Nevertheless, image processing through computer vision offers advantages over these conventional methods.

Computer vision analysis is noninvasive, posing a minimal risk of damaging the artwork. It leverages statistical knowledge from thousands of brushstrokes extracted from numerous artworks, thereby providing researchers with an in-depth understanding of art without needing physical access. Moreover, the number of paintings that can be analyzed using computer vision is virtually limitless.

Machine learning approaches to art analysis can be broadly divided into two categories: “feature engineering-based methods” and “feature learning-based methods”.

Feature engineering-based methods involve the design and programming of visual features by human experts. These features, extracted from paintings, are used as input for statistical classification algorithms. While promising for identifying elements like brushstrokes, color, and texture, the effectiveness of this approach is heavily dependent on the relevance of the selected features to the artwork’s style. A key limitation is the reliance on expert input to identify significant and computable features, which can be time-consuming and potentially biased. Moreover, this method might struggle when dealing with damaged or aged artworks. Among feature engineering-based approaches, stochastic models appear, such as Li and Wang’s method to use local features constructed from wavelet coefficients by spatial stochastic processes, particularly the multiresolution 2D hidden Markov models [11]; there are also methods exploiting the geometric and surface properties, such as the work of Vieira et al., in which classification is based on the features computed from shape, texture complexity and surface curvature [19].

On the other hand, feature learning-based methods utilize statistical learning methods, such as artificial neural networks (ANNs), to determine the relevant features for an analysis task. This approach eliminates the need for handcrafting visual features and allows a deep neural network to extract patterns unique to a group of paintings. These learned patterns can then be used in a classical classification framework or further examined by a neural network. This approach can capture both global and local properties of paintings and has shown superior performance over traditional low-level feature-based methods in style classification. The efficiency and speed of this approach can be further improved by convolution neural networks (CNNs), which focuses more on the local features of an artwork, thus reducing execution time and improving the understanding of subtle details. Despite the interpretability of the extracted features often being a challenge with CNNs and feature-learning approaches in general, they remain one of the most up-to-date and widely adopted methods [20].

1.3 Related work

As discussed in Section 1.2.1, the field encompasses a wide spectrum of work. Art authentication is one of the most crucial and challenging fields in art research. Machine learning appears among the methods of such renowned institutions as the Rijksmuseum [4]. In this paper by Dobbs and Ras, a pre-trained model known as Residual Neural Network (ResNet) was employed to classify art from the Rijksmuseum dataset. The authors addressed the problem of imbalanced classes in the Rijksmuseum dataset to create a more robust data structure for future work. While this thesis employs EfficientNet (EffNet), understanding of its predecessor, the ResNet, is important. The ResNet architecture provides a foundation upon which EffNet, a more modern and state-of-the-art model, was developed. The work conducted by the Rijksmuseum also shows the increasing relevance and necessity of machine-learning projects in prestigious institutions.

As a result of the popularity and efficiency of machine learning in art, numerous art datasets have been created, specifically for the training of convolution neural network (CNN)s. To the best of my knowledge, the first dataset specifically designed to provide a colorful collection of paintings from different artists to be used in classification problems, is Painting-91 [9]. Another, more recent and frequently talked about is the MultitaskPainting100k dataset, originating from the Kaggle competition Painter by Numbers¹. Most of the data in this set derives from WikiArt, and it received its final name as the result of the comprehensive experimental article that used it [2]. These datasets were not used in this thesis, however, they are essential to note when talking about the state-of-the-art in art classification and authentication, as well as serving as good baselines when creating my own dataset.

As the number of open-source pre-trained neural networks increases, there is a growing need to compare their performance to determine their optimal applications. Zhao et al. compare the networks that were the most current and widely applied at the time of this thesis [21]. Their work will be discussed in more detail later since it has served as one of the most important guidelines when choosing EfficientNet as my pre-trained network. Tan and Le discuss the creation of EfficientNet [16]. Menai and Babahenini examine the performance of the different EffNet versions in art classification [12]. Both papers will be discussed in more detail later.

As mentioned in Section 1.2.1, CNNs are not the only method for art classification. The work of Agarwal et al. is an excellent example of the usage of hand-crafted features in other machine learning algorithms [1]. While their results are noteworthy, analyzing their methodology reaffirmed my decision to use convolution neural networks, as the nature of my project differs significantly from theirs.

This study encountered the common challenge of dealing with a small amount of data. Data augmentation, which offers a variety of methods, was employed as a solution to the potential problems arising from this challenge. The authors of [13] offer a comprehensive

¹<https://www.kaggle.com/c/painter-by-numbers> (Accessed 2023.06.02)

look at the different methods used in image augmentation, including methods like cropping and flipping, that were also utilized in this thesis.



(a) *The Last Judgement triptych*
by Hieronymus Bosch

(b) *The Last Judgement triptych*
by an anonymous follower

Figure 1.2: The “same” paintings by different artists

1.4 The goal

The aim of this work is to study the performance, advantages, and limitations of artificial neural networks in the field of fine art classification. The objective is to employ convolution neural networks to address a binary classification problem, where the input is an image of a painting created in the style of Bosch, and the target variable is the category of the painting, which can be either 'Bosch' or 'non-Bosch'. As clear by looking at Figure 1.2, distinguishing such similar works is not a straightforward task. I chose Hieronymus Bosch and his followers as subject because of the uniqueness of Hieronymus's style and the fact that the relations between their paintings are widely studied until this day. I had, of course, personal reasons as well: the way his artworks captivated me was a big motivation.

2

Convolution neural networks in fine art classification

This chapter provides a general overview of the usage of machine learning in painting classification.

The chapter is structured as follows:

- Section 2.1 Artificial neural networks in general; training, validation and testing process; difference between fully-connected networks and convolution neural networks
- Section 2.2 The definition of convolution neural networks and their usage in image processing
- Section 2.3 A non-exhaustive list of other important tools often appearing when using CNNs
- Section 2.4 The definition and importance of transfer learning
- Section 2.5 Networks commonly used in transfer learning based on the article on painting classification by Zhao et al. [21]
- Section 2.6 Description of the network called EfficientNet and its supreme performance both generally and in fine art classification based on the articles by its creators [16, 12]

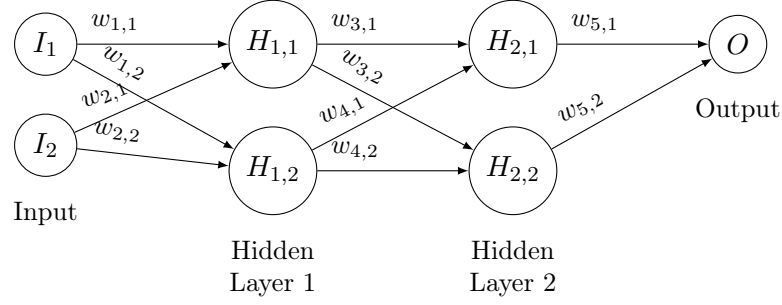


Figure 2.1: A fully-connected ANN with 2 hidden layers

2.1 Artificial neural networks

As its name suggests, an artificial neural network (ANN) is a structure that imitates the operation of biological neural networks. Just like in a real neural network, the smallest component in an ANN is called a neuron. If two neurons are connected, then a signal is transmitted between them. These connections are called edges and have weights, which tell the significance or the strength of the given connection. Typically, neurons are divided into layers, and the neurons within one layer together perform a transformation on their input.

An ANN is made up of the following elements: input, hidden layers, and output, as seen in Figure 2.1. The input enters the network by being connected to the neurons in the first hidden layer. In the hidden layers, the input is forwarded to the next layer after transformation. A transformation made by a neuron is actually a matrix multiplication carried out on the arriving input vector. This weighted sum is then assigned to the activation function of the neuron, determining its final output. The output is then transmitted to the neurons of the next layer, to which the neuron is connected. The last layer is the output layer which returns the result of the whole network. The output layer can have multiple neurons, too.

When used in machine learning, the number of the input and the output is determined by the problem, *e.g.*, in an image classification problem, the input size is the number of pixels, and the output is the number of possible classes. Everything else is freely chosen: the hidden layers, the activation functions, the connections between the neurons *etc.*

The most significant and so far unsurpassed strength of convolution neural networks in the field of image processing is their feature extraction capability: during the training of a network, the initially user-defined weights (parameters) are adjusted by the network itself during transmission, in order to find the relevant features that determine the class of the input. In the training section, the true labels of the input are known to the network, and the weights are adjusted based on the loss calculated using the true labels. During testing, the weights do not change: a so-far not-seen test set is given as an input to the trained network to classify them based on the knowledge it gained from the training. An additional validation set might be added which is classified after the training to check accuracy. The training usually consists of multiple epochs, during each of which the whole training set is

reevaluated, then the performance is checked on the validation set.

ANNs can be divided into groups based on the number of connections between the neurons. In Figure 2.1 we can see a fully-connected network, meaning that all the neurons within a layer are connected to all neurons in the previous and the next layer. (FC neural networks are also called densely connected networks.) However, they are not used in image classification problems, due to the fact that the number of edges would be unmanageably big because of the number of pixels. Instead, convolution neural networks are used which can exploit the local correlation between pixels by having fewer parameters than FC networks, thus focusing on specific areas of an image at a time.

2.2 Convolution neural networks

Convolution neural networks (CNNs) are a type of artificial neural network that consists of three main types of layers:

- convolutional layer
- pooling layer
- fully-connected (FC) layer

The most important building elements are the convolutional layers, which require input data, a filter, and a feature map. The input data in our case is a color image represented by a 3D tensor of pixels (height, width, depth), such that it corresponds to an RGB image. The filter or the kernel is a 2D array of weights. The typical size of such an array is 3×3 , but surely smaller than the size of the input image. Throughout the training of a CNN, the aim is to find the most optimal values of the weights. To compute the output created by the weights, the dot product of the input and the filter is calculated in each layer in the following way: in each step, a different region of the pixels is selected with the same size as the filter, so that the end we get a series of dot products in order. This series is called the feature map.

After each such convolutional operation, an activation function is applied to the feature map. The default function is usually a Rectified Linear Unit (ReLU), which returns the values unchanged if it is nonnegative, otherwise, the return value is zero.

The task of a pooling layer is dimension reduction. It uses a filter as well but with no weights. The two main types of pooling are max and average pooling, which calculates the maximum or the average of clusters of its input.

In a CNN, every layer concentrates on a specific feature of the input. In the case of an image, one layer determines features such as colors, edges, local symmetry, textures, patterns, or gradients. Pixels that are far away from each other are not connected in the hidden

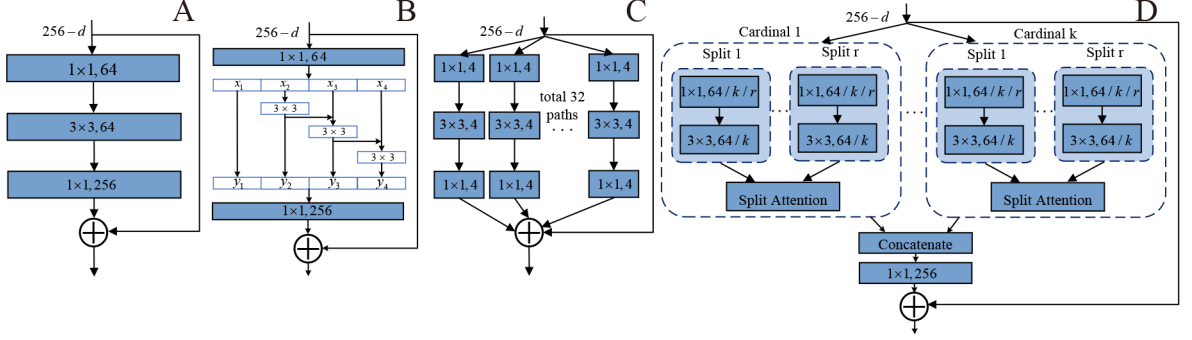


Figure 2.2: The block structure of
A: ResNet, B: Res2Net, C: ResNeXt, D: ResNeSt
source: [21]

layers, since the information about the edges, colors *etc.* are found in pixels close to each other. However, in order to identify more complex objects, the whole picture needs to be understood as one. That is the purpose of the FC layers, which are the last layers. An FC layer connects each pixel value to all the nodes in the output so that all the collected information is assembled at the end.

The convolution neural networks are frequently mentioned in the field of art classification and therefore focused on in this study as well are Residual Neural Network (ResNet) and its variants. The block structure comparison of the versions of ResNet is depicted in Figure 2.2. ResNet is fairly simple: its layers directly follow each other. Res2Net increased the number of scales the output can represent by a hierarchical structure. ResNeXt has multiple parallel chains of convolutions in a block and connects them at the end. ResNeSt takes ResNeXt's approach one step further: it introduces the concept of the split-attention module, as shown. The reason why all these different block structures were created is that it has been realized that a better performance can be achieved by making the network focus on the features respectively. The downside is the dramatic increase in the number of parameters needed.

2.3 Other important tools

The following glossary elucidates the array of tools that were recurrently referred to in the academic literature relevant to this field of study and were also utilized in this project. It is important to note that the realm of tools used in image processing is far more expansive than this selection; nevertheless, these particular tools were deemed relevant for this study.

- Dropout layer: is used to reduce overfitting. It drops out nodes from layers with a dropout probability and leaves the remaining nodes untouched.
- Adam optimizer [10] (named after the term “adaptive moment estimation”): is a replacement and an extension of the stochastic gradient descent. The classical method has a single learning rate, whereas Adam adjusts the learning rate. It computes

individual adaptive learning rates from estimates of the first and second moments of the gradients. It is preferred over the classical method because it treats problems that are non-stationary or have sparse gradients. Also has low memory requirements.

- Softmax activation function: it is the generalization of the logistic function to multiple dimensions. Used in multiclass problems, typically in the last layer.
- Rectified Linear Unit (ReLU): it is a node that implements the often default activation function called Rectified Linear Function. The sigmoid and tanh functions often create the vanishing gradient problem by projecting all values larger than 1 in absolute value to 1. Contrary to this, the function of ReLU puts out the input unchanged if it is positive, otherwise returns 0. Another advantage is the easy calculation of the derivative.
- Batch normalization [7]: a method used to make training faster and more stable through normalization of the layer's inputs by recentering and rescaling.
- Log loss or cross-entropy function: If we have discrete random variables, the logarithmic loss for one data point is defined as

$$L = - \sum_{i=1}^n t_i \log p_i \quad (2.1)$$

where t_i is the true label of the data (0 or 1), p_i is the probability of containment in a class given by the model. In our case, the class number n is 2.

2.4 Transfer learning

Creating a CNN from scratch to deal with a specific problem would be extremely time-consuming. Luckily, when it comes to complex tasks, the basics we need in all projects are quite similar. In order to process an image, you have to first determine the basic shapes, textures *etc.* Whether you have X-ray images or paintings, you have to start in the same way: making the network identify what is depicted on the image, drawing further conclusions comes after that. To shorten the development of a CNN without loss of performance, we apply transfer learning. CNNs that are pre-trained on general datasets are widely available. A pre-trained CNN is able to recognize the outlines, the color *etc.* of any image, thus identifying countless objects with high precision. Their capability of carrying out general feature extraction on any picture can be further specified to fit the needs of a task by training them on another dataset and adding a small number of layers. Further training refines the weights of the already existing layers, which is easier and more efficient than optimizing from random initial values. When adding layer(s) to get an output in the form of our needs, we have to bear in mind that the FC layers need to remain the last ones.

2.5 Introduction to networks commonly used in transfer learning

In a painting classification task, the pre-trained neural networks that are most frequently discussed are ResNet and its variants. There are, of course, others as well (the most well-known is probably AlexNet which became popular after winning the championship ImageNet Large Scale Visual Recognition Challenge in 2012), but they are less often mentioned or utilized in comparison to the ones covered in this work, even though they were considered the best in their respective times.

In the recent paper by Zhao, Zhou, Qiu, and Jiang, a comparative experiment was conducted to ascertain the effectiveness of seven distinct models in the task of art classification - identifying genres, styles, and artists [21]. The models were evaluated under the same experimental conditions on three datasets, Painting-91, WikiArt, and MultitaskPainting100k, and the efficacy of transfer learning was also tested. This study served as a key reference for my thesis, as it facilitated the selection of the optimal model suited to my designated task. Furthermore, it emphasized performance enhancement via supplementary training and the usage of an additional layer within the model structure.

The models compared in [21] are those whose structures are depicted in Figure 2.2, alongside with RegNetX, RegNetY, and EfficientNet. All these networks were pretrained on the ImageNet dataset that contains 15 million hand-labeled high-resolution images representing approximately 22,000 different object categories. It has been found that models with larger network depths, widths, and resolutions achieve better accuracy. However, increasing their size also creates greater complexity, longer run times, and higher storage requirement.

The images of the paintings in [21] went through some data augmentation before entering the models. In this study, I also applied these same steps to augment my data.

1. RandomResizedCrop. A random crop operation with a range of size between 0.08 and 1.0 of the original size and aspect ratio was conducted. The cropped image was finally resized to 224*224 using random interpolation.
2. RandomHorizontalFlip. The given image was randomly flipped horizontally at a pre-set probability of 50%.
3. The Python import library (PIL) image was converted to a tensor and normalized using the mean and standard deviation.

In addition to the image preparation, the last fully-connected layer of the networks was changed to fit the number of classes present in the dataset, and a softmax layer was also added.

The authors of [21] concluded that ResNet and EffNet demonstrated the best performance in terms of style, artist, and genre categorization across all test image datasets. Additionally, EfficientNet outperformed ResNeSt in artist classification in most cases. This achievement of EfficientNet led me to the conclusion that in my artist classification task EfficientNet will be the best choice.

2.6 EfficientNet

EfficientNet differs significantly from the rest of the networks mentioned in the way it is scaled up. The authors of [16] decided to experiment with the way CNNs are scaled up: so far the method to improve the performance of CNNs was to increase only one of the dimensions of the network by making it either wider, deeper or higher in resolution (in Figure 2.2 it can be seen that the versions depicted used this approach as well). The proposed method that led to the creation of EfficientNet was compound scaling that uniformly increases all dimensions by a fixed ratio. “Intuitively, the compound scaling method makes sense because if the image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image[16].” Both theoretical and experimental results show the truth behind this claim.

The equation used to create EfficientNet is the following (excerpt from [16]):

$$\begin{aligned}
 \text{depth: } d &= \alpha^\varphi \\
 \text{width: } w &= \beta^\varphi \\
 \text{resolution: } r &= \gamma^\varphi \\
 \text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\
 \alpha \geq 1, \quad \beta \geq 1, \quad \gamma &\geq 1
 \end{aligned} \tag{2.2}$$

where the compound coefficient φ is a user-specified coefficient.

EfficientNet has versions from B0 up to B7. After finding the best values of α , β , and γ for EffNetB0 (that are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$) and fixing them, we scale up the baseline network with different φ ’s, such obtaining the higher versions.

The supreme performance of EfficientNet models in general:

- they generally use an order of magnitude fewer parameters and FLOPS (floating points operations per second) than other CNNs with similar accuracy;
- faster (*e.g.*, EfficientNetB1 runs 5.7 times faster than ResNet-152);
- great transfer learning results, in particular, they outperform all other types of CNNs commonly used in transfer learning on the dataset ImageNet.

In another recent closely related work [12], Menai, and Babaheini compared the performance of ImageNet-pre-trained EfficientNet models from B0 to B6 in recognizing the style of fine-art paintings, illustrated in Figure 2.3. They created custom models by supplementing the EfficientNet base architectures with additional layers, and these models were then fine-tuned via transfer learning on the Painting-91 dataset under consistent experimental conditions. They did not attempt to solve the classification problem of artists, however, after my choice of EffNet based on the results of [21], their work served as a good basis

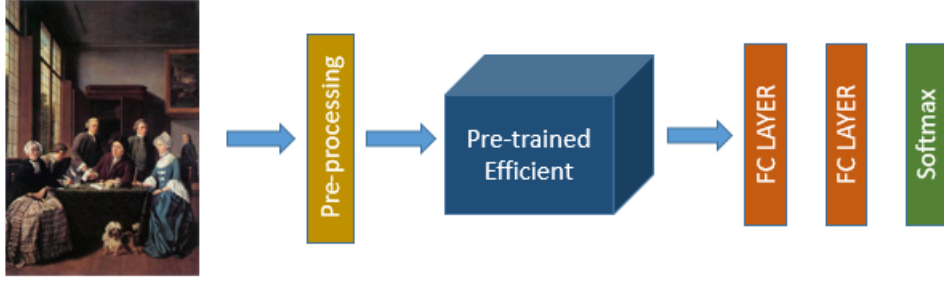


Figure 2.3: The structure of the classification model in [12]

while evaluating my own results, knowing what to expect in terms of the differences between the performance of the versions. They also showed that fine-tuning the pre-trained ImageNet weights instead of reproducing the full training procedure with random weight initialization always resulted in better performance. Therefore, following their approach, in this work, I also started the fine-tuning process with the pre-trained ImageNet weights in order to get better results.

In contrast to the authors of [21], Menai, and Babaheini introduced two additional FC layers instead of one, both had ReLU activation functions. A softmax layer was also applied at the end to fit the output to the number of classes, and before it, a Batch normalization and dropout layers were applied to prevent overfitting.

Table 2.1: Results of EffNet models with different pre-training strategies [12]

Model	Base pre-trained	Custom pre-trained	Custom (8 layers) pre-trained
EfficientNetB0	68.47%	70.31%	71.32%
EfficientNetB1	68.56%	70.68%	72.06%
EfficientNetB2	69.12%	70.86%	72.89%
EfficientNetB3	69.30%	71.42%	73.53%
EfficientNetB4	70.04%	72.52%	73.90%
EfficientNetB5	70.68%	73.13%	74.54%
EfficientNetB6	70.96%	73.47%	75.55%

Table 2.1 shows the performance of the models with different pre-training strategies reported in [12]. The column called “Base pre-trained” presents the performance of the pre-trained models without any additional training on the custom dataset. In contrast, the column called “Custom pre-trained” displays the performance results of these models after they have been further trained on the same dataset. Column “Custom (8 layers) pre-trained” details the outcomes for these models after the inclusion of eight custom layers and further training identical to that of the “Custom pre-trained” column. As seen, custom pre-trained models all showed accuracies no lower than 70%. It was concluded that the accuracy was improved by both deeper networks and higher input resolution.

3

Experimental Setup and Data

3.1 Programming Environment

In this subsection, I will provide an overview of the programming environment, including the software, libraries, and hardware used in the context of this study.

The coding was carried out in Python, the choice of which was driven by its readability and an array of libraries specifically designed for data analysis and machine learning. The Python version used in this study was 3.11.2 (64-bit). The source code of this project and the utilized datasets are available at the following GitHub repository: https://github.com/eszternagy002/bsc_bosch.

The instrumental Python libraries were the following:

- `selenium` and `bs4` were used for web scraping the required data,
- `albumentations` and `cv2` were employed for data manipulation and saving the augmented images,
- `os` was used to create the folder structures of the organized paintings,
- the `datasets.ImageFolder` and `DataLoader` commands of the `torchvision` package were used to create the dataloaders from the folders,
- `tqdm` was utilized to iterate over the dataloaders.
- `torchvision.models` was used to implement the pre-trained models,

- **torch** (officially known as PyTorch, but imported as 'torch' in Python code) was employed for transferring images to tensors, importing optimizers, loss functions, carrying out the actual training and testing, and saving the models,
- **matplotlib** and **seaborn** were used to save plots of accuracies and losses, as well as to create confusion matrices.

The computations were performed on a computer with a high-performance GPU, specifically an NVIDIA GeForce GTX 1050 Ti, which was vital for handling the complex calculations involved in training the deep learning models. Furthermore, to accommodate the large datasets and the intensive processing requirements, the computer was also equipped with 16GB of RAM, ensuring smooth and efficient data handling and model training operations.

3.2 The classes

In this work, I used the following data classes:

- original Bosch paintings,
- works in the style of Bosch made by followers, and
- a miscellaneous or uncertain class.

In the training and validation subsets, I strictly used the first two. The third class has more variety: it consists of pieces of art whose creator is either unsure or that were made by the workshop of Bosch. These latter paintings were all part of the test set.

3.3 The datasets

There are numerous datasets available for painting classification that are pre-processed for training. However, the majority of them are general datasets that are *e.g.*, made for style classification. I could not find any open-source Bosch follower dataset that is made for CNNs, nor one that contains all known Bosch paintings. Therefore, I looked for non-CNN specific websites and decided to use the pictures from the website of the Jheronimus Bosch Art Center (JBAC)¹ and **boschproject.org**². Both sites have high-resolution images labeled by art experts.

I obtained 40 pieces of art from **boschproject.org** that are attributed to Bosch himself and 7 made by his workshop. However, some of these works are ink drawings without color. Combining these ink drawings and the colorful oil paintings would not have resulted in a manageable dataset. After excluding them, there remained 28 different pictures of

¹<https://jheronimusbosch-artcenter.nl/> (Accessed 2023.04.11)

²<http://boschproject.org/> (Accessed 2023.04.11)

original Bosch paintings, 9 workshop paintings, and 2 other artworks, whose authorship is questioned.

Repeating the same selection process with the website of JBAC, a dataset of 746 artworks by followers was obtained. 13 pictures were also added to the third category.

3.4 Data augmentation and dataset ratios

An imbalanced training set refers to a dataset when the amount of data across the classes differs significantly. In our case, the 28 Bosch paintings and 746 follower artworks have a ratio of approximately 0.038. This will inevitably lead to the model having a preference toward classifying as non-Bosch, which results in poor performance in the minority class. In order to avoid this, oversampling was carried out on both sets in the common form of data augmentation. Additionally, oversampling provides a larger dataset in general.

I augmented the data using the same augmentation method described in [21]. For every picture in the Bosch set, I created an additional 124 augmented images using the Python Albumentations package. In the follower set, 7 additional images were added to every existing one. The augmentation process can be seen in Figure 3.1. The original image undergoes a random resized crop, after which there is a 50% chance for a random horizontal flip. Finally, the image is converted to a normalized tensor. At first, RandomResizedCrop was carried out with parameter 224, as seen on the size of the final tensor, but I later repeated the augmentation process with different parameter sizes.

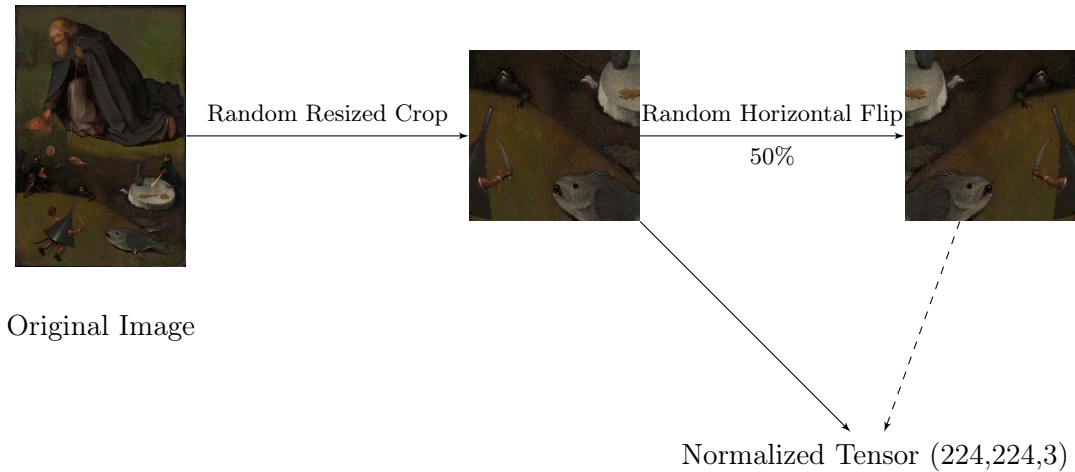


Figure 3.1: Flowchart illustrating the image augmentation process using the painting called *The Temptation of Saint Anthony* by Hieronymus Bosch

After this, the train, validation, and test sets were created, as depicted in Figure 3.2. In order to avoid data leakage between the sets, the selection was based on the individual paintings themselves, not the images. The training set size was chosen to be 70%, the validation set was 10%. To ensure variety, the Bosch set and the follower set were dealt with separately first: 20 Bosch paintings were randomly chosen to be put into the training

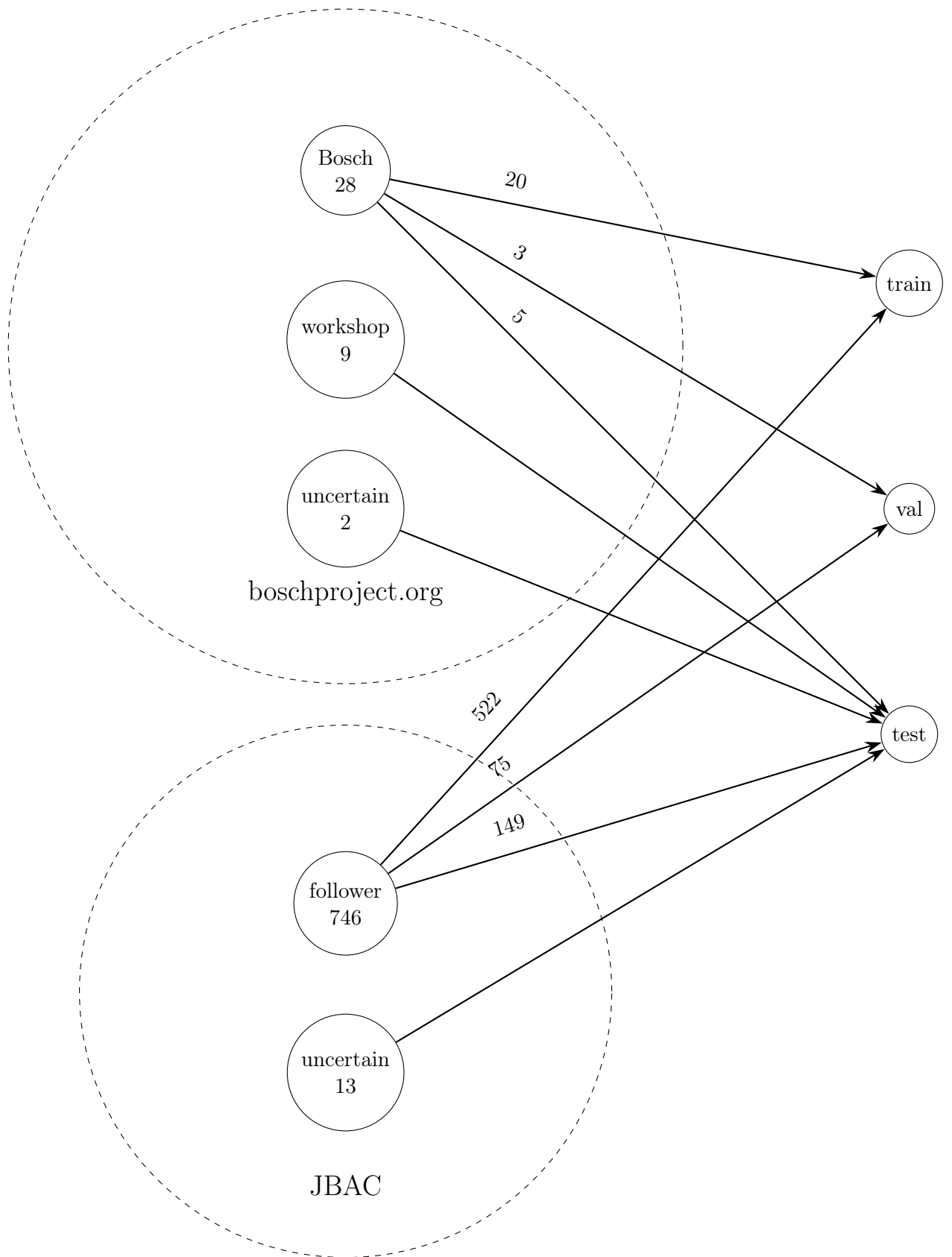


Figure 3.2: *Sorting of the paintings*

set, and 3 into the validation set. The same ratio was applied to the follower set: 522 follower paintings ended up in the train set and 75 in the validation set. The remaining 5 Bosch artworks, 149 follower artworks, which made up 20% of the paintings in each category, and 24 uncertain paintings became the test set. Additionally, the train set was enlarged by a selection of augmented pictures based on the already chosen paintings. Some images were discarded due to the fact that blank places of the pictures were chosen while cropping.

Altogether, a total of 2492 Bosch and 4176 follower pictures were included in the training set after discarding some images, meaning that the size of the Bosch set in the training set was now about half of the follower set, versus the previous 3%. In order to preserve consistency and make the workflow reproducible in case I need to repeat experiments, I set the same random-seed number for every randomized choice on the paintings.

4

Training and results

4.1 The training process

At first, I decided to carry out the training by fine-tuning the weights of the versions of EfficientNet, leaving the structure unchanged. Although to fit the problem at hand, I had to change the output number of the last layer from 1000 to 2, given that the original model was trained on the ImageNet dataset, which contains 1000 distinct classes.

Before entering the networks, the images underwent a process of conversion into normalized tensors, to ensure data consistency. This process includes resizing to match the required input size of the different versions, converting into tensors, and finally normalized, using the torchvision package. The tensors represent the images in three dimensions: the number of channels, the height, and the width of the image. In this case, the channel number is 3, because the pictures are represented in RGB coding. This implies assigning a 3-element float-valued vector to each pixel of the picture that had values in the interval $[0, 255]$, before the tensor conversion. The tensor values are between 0.0 and 1.0, rescaled by the ToTensor command, depending on the content of each red, green, and blue. After the tensor conversion, the normalization was applied by subtracting the mean and dividing by the standard deviation of these three color values. The mean and standard deviation vectors were derived from the training set of paintings, excluding the augmented images.

The required dataset and dataloader structures were created by the PyTorch ImageFolder and DataLoader functions. The latter provides an efficient environment for handling image data through its batching capabilities and memory management. It also allows for seamless integration with multithreading, thus enabling parallelism and speeding up data feeding.


```

#defining the optimizer and loss functions
optimizer = Adam_optimizer()
criterion = CrossEntropyLoss()

#defining the training function for one epoch:
def train(model, trainloader):
    model.train()
    for i in trainloader:
        image, label = i
        outputs = model(image)
        optimizer.zero_grad() #setting all the gradients to zero for fine-tuning
        loss = criterion(outputs, labels)
        train_loss = loss.item()
        loss.backward() #backpropagation based on training loss
        optimizer.step() #optimization step on the weights based on backpropagation

def validate(model, valloader):
    model.eval()
    with torch.no_grad():
        #similar to the training function, but without the optimization steps

for epoch in epochs:
    train(model, trainloader)
    validate(model, valloader)
    save(train_loss, val_loss)
    save(train_acc, val_acc)

save(model)

```

Code 4.1: *Pseudocode of the training code*

Pseudocode 4.1 outlines the training and validation processes. The code includes a training function and a validation function, both executed within a for loop that iterates over multiple epochs.

At the beginning of the training, the default learning rate was 0.0001, which was adjusted by the Adam optimizer in each epoch. The loss function was the log loss or cross-entropy function since the training and validation sets consist of only two classes. During each training, the training and validation accuracies and losses were collected and saved as plots, as well as the result trained models as serialized PyTorch state dictionaries in PTH extension. PTH in Python stands for path and is usually associated with Python path configuration files. However, it is also the conventionally used file extension for saving serialized model weights when preserving the state of a model, as applied here.

The number of epochs was not optimized during the training; it was a user-defined constant, which I set to 15. Initially, I wanted to test the performance with larger epoch sizes. However, I found that a more time-efficient approach was to observe the subtle differences after shorter training periods, making incremental changes. Once the optimal parameters were identified, I ran the final model for a larger number of epochs, such as 20, 40, and 50.

4.2 The significance of image sizes, resolution, and class sizes

During the training, I experimented with different image sizes, number of augmented pictures, and epochs. Many adjustments were necessary, emerging from challenges related to image sizes, class sizes, or finding the optimal number of epochs. However, they highlighted the importance of certain details. The following results were made using the fine-tuned versions without any additional layers. The different approaches utilized were the following:

1. cropping all augmented images to a uniform size of 224×224 , regardless of their version,
2. constructing distinctive augmented sets for each version, corresponding to its unique input requirements,
3. reducing the number of augmented images per follower painting to balance the class sizes.

4.2.1 Small image size

As mentioned above, the first approach was to start the data augmentation process by cropping 224×224 sized sections from each picture. Then, I used all of these cropped sections to train the networks. However, the different networks require different image sizes, the higher version the larger (*e.g.*, EffNetB6 requires 528×528 size), so when using the 224×224 images, they went through an enlargement which made them more and more blurred. This resulted in poor performance on the test data: all pictures were classified as non-Bosch, and this was the case even for versions B1 and B2, which require smaller image sizes. I concluded that this was likely due to the fact that in the Bosch set, the proportion of augmented images was much larger than in the follower set. The higher versions, once trained, could not even run on the test data (either produced an error or ran endlessly), probably due to this issue.

4.2.2 Re-cropping to correct size

Seeing the problems of the first method, a second approach was conducted. Specifically, the recreation of the input data by making different input sets for every version by changing the size parameters of RandomResizedCrop. This way, I received significantly better performances. However, the models seemed to show a preference towards non-Bosch images: the non-Bosch images were classified correctly usually more than 90% of the time, whereas the accuracy of Bosch images was largely fluctuating between 20-40% (but it is needed to be mentioned that the test set contains only 5 Bosch paintings). The confusion matrix shown in Figure 4.1 displays the results for version B4, which exhibited the best performance. (The 40% correctly classified Bosch paintings means 2 out of 5.)

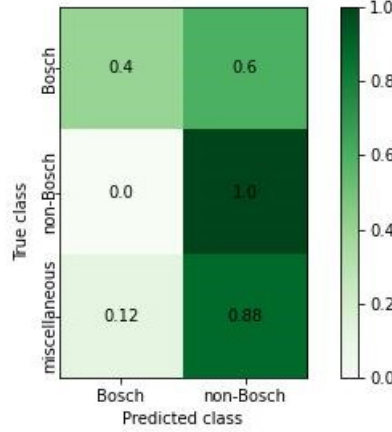


Figure 4.1: *Confusion matrix of the results of EffNetB4, with required input sizes, but non-equal set sizes*

4.2.3 Same class sizes

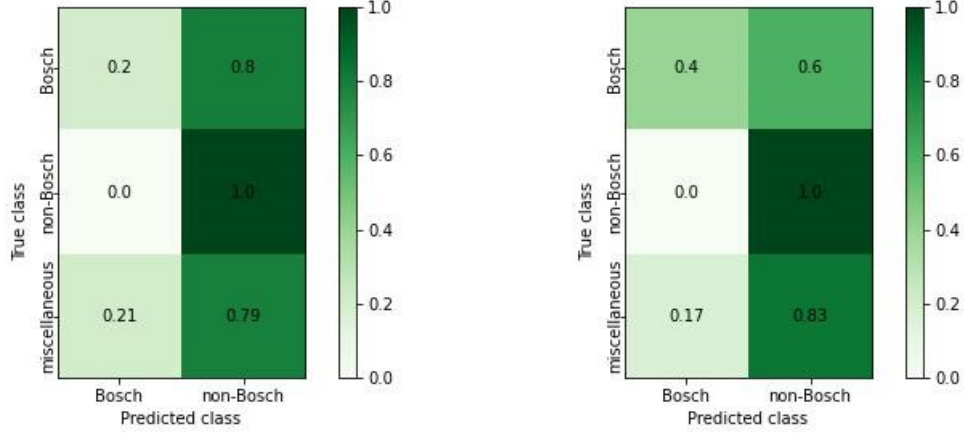
I assumed that the discrepancy described in subsection 4.2.2 could be attributed to the size difference between the Bosch and non-Bosch sets, which, despite the image augmentation, still seemed to have remained significant. Therefore, I decided to change the sampling technique, specifically to reduce the number of augmented images in the follower set to three per painting in order to bring the ratio of the two main sets even closer to 1:1.

I maintained the training size of the Bosch paintings while only reducing the augmented follower pictures. Initially, each painting in the non-Bosch set was represented by seven augmented images. This number was reduced to three by implementing a procedure involving random selection. Specifically, for each of the 746 original paintings in the non-Bosch set, four unique numbers were randomly chosen from a range between 0 and 6. These randomly selected numbers served as indices, corresponding to the augmented images to be removed from each painting's set to ensure consistency. I retrained the models on these sets, then compared the results with the previous ones.

Interestingly, the best performance was given by version B0: 3 out of the 5 Bosch artworks were correctly classified, and 99% of the follower paintings. Bearing in mind that before the environmental changes, the B0 version classified everything in the test set as non-Bosch ones, it is a significant improvement.

Version B5 also showed improvement, as demonstrated in two confusion matrices shown in Figure 4.2. EffNetB5 misclassified 4 out of 5 Bosch paintings at first. After equalizing the class sizes, the model was correct on 2 Bosch. At first glance, it does not seem like a notable difference. However, the proportion of paintings classified as non-Bosch in the miscellaneous category increased from 79% to 83%, suggesting that the model gained a better understanding of the unique features of Bosch's artwork.

The results of EffNetB4, which previously showed the best performance before the class size reduction, remained the same, however, its validation loss decreased from 1.4% to 0.7%,



(a) Performance of EffNetB5 after training on unequal class sizes (b) Performance of EffNetB5 after training on equal-sized classes

Figure 4.2: Improvement of EffNetB5 after set-size change

meaning more confidence in its prediction. The performance of other versions, determined by the number of correctly classified Bosch images, can be seen in Figure 4.3.

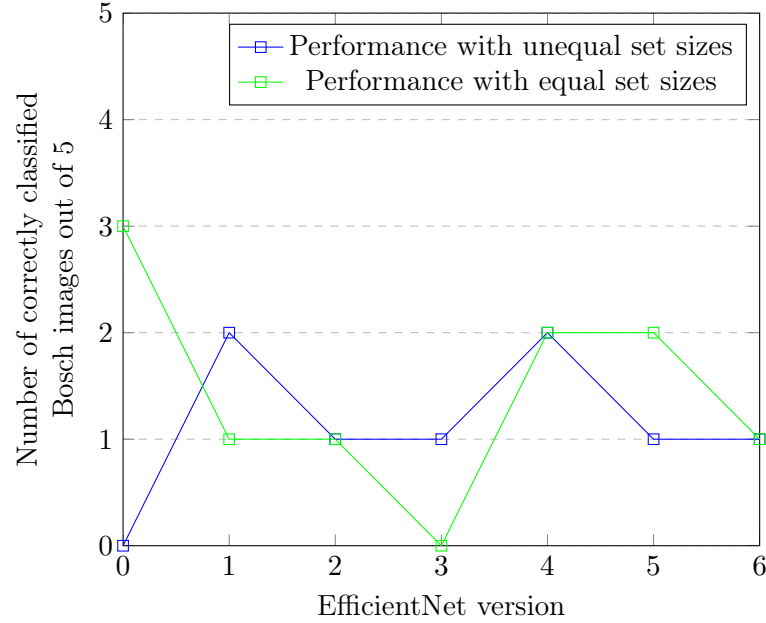


Figure 4.3: Difference in performances with unequal and equal set sizes

4.3 Adding a layer

In the following section, the impact of layer addition on performance is analyzed. Initially, the structure of the layers will be detailed. After that, the optimal number of training epochs will be determined.

The last classification layer of the versions of EffNet have the same structure, but differ in parameters p and `in_features`:

```
(classifier): Sequential(
  (0): Dropout(p, inplace=True)
  (1): Linear(in_features, out_features=2, bias=True))
```

Code 4.2: *Original last classification layer*

In pseudocodes 4.2 and 4.3, the p parameter, which is the dropout probability, increases as the version number increases, as does the `in_features`. The `in_features` parameter depends on the output size of the preceding layer. The latter depends on the output size of the layer before. In experiments previously discussed, `out_features` was modified to match the number of classes, which is 2, for our case. I decided to add another FC layer at the end, whose `out_features` became the number of classes, while the first FC layer's `out_features` were left at the default value.

So the final layer ended up as follows:

```
(classifier): Sequential(
  (0): Dropout2d(p, inplace=True)
  (1): Linear(in_features, out_features=1000, bias=True)
  (2): Linear(in_features=1000, out_features=2, bias=True))
```

Code 4.3: *New last classification layer*

The default value for `out_features` is 1000, which I did not modify. I retrained the models with this structural change, the training set remained the same as in section 4.2.3.

Initially, I decided to run the models for 15 epochs, influenced by the computational complexity of larger epochs. My objective was not to maximize performance at this stage but rather to assess the relative merits of each model for a shorter epoch size. This would provide a basis for determining which models might benefit from longer training.

4.3.1 Finding the best epoch number

In this section, the primary focus was to identify the optimal number of epochs that yield the best performance when distinguishing between the paintings created by Bosch and those created by his followers. Given that all models consistently achieved at least a 90% accuracy rate on the follower set, irrespective of the presence of an additional layer, the decision on which model to proceed with was made based on the performance on the Bosch set.

As seen in Figure 4.4, the biggest increase in performance was observed in version B2. Therefore, I decided to retrain it for 50 epochs.

However, after evaluating the newly trained EffNetB2, its performance declined: only one Bosch painting was correctly classified. Therefore, I examined the validation and loss accuracy plots.

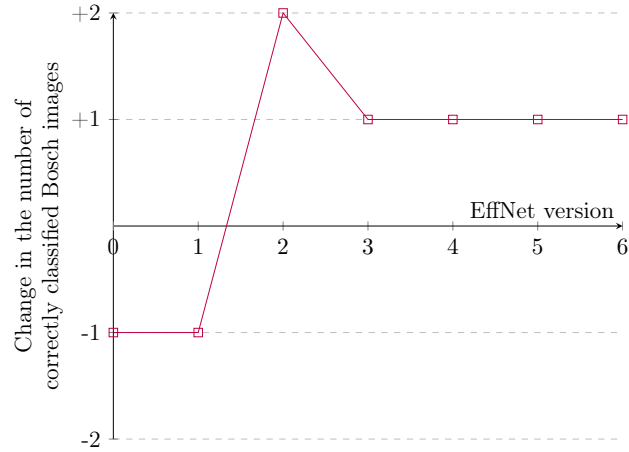


Figure 4.4: *Performance change after adding a layer*

As Figure 4.5 illustrates, the validation loss was fluctuating at first between 4 and 10%, but stabilized around 1-2% after 20 epochs. Then between 35 and 50 epochs, we can see sudden peaks. Since the training loss consistently stayed below 1%, I concluded that the model overfitted the training set, and carried out the training on smaller epoch numbers to find the most optimal model state.

As I was retraining and reevaluating the performance on different epoch numbers, it became clear that the model was overfitting, since the larger the epoch number was, the worse the accuracy became on the test set, as seen in Figure 4.6. Therefore, I concluded that the optimal epoch number must be below 20.

Initially, I retrained the model for 10 epochs, but it resulted in all paintings being classified as non-Bosch. Therefore, I proceeded with 13 and 14 epochs then 16 and 17 epochs (I started all new training from 0, so that the epochs are not added up together), but all were a decline in the performance. This indicates that the accuracy reached its peak at 15

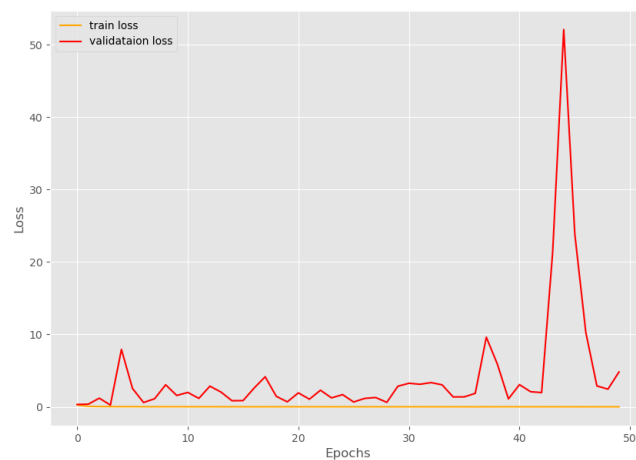


Figure 4.5: *The loss functions of EffNetB2, trained on 50 epochs with new layer*

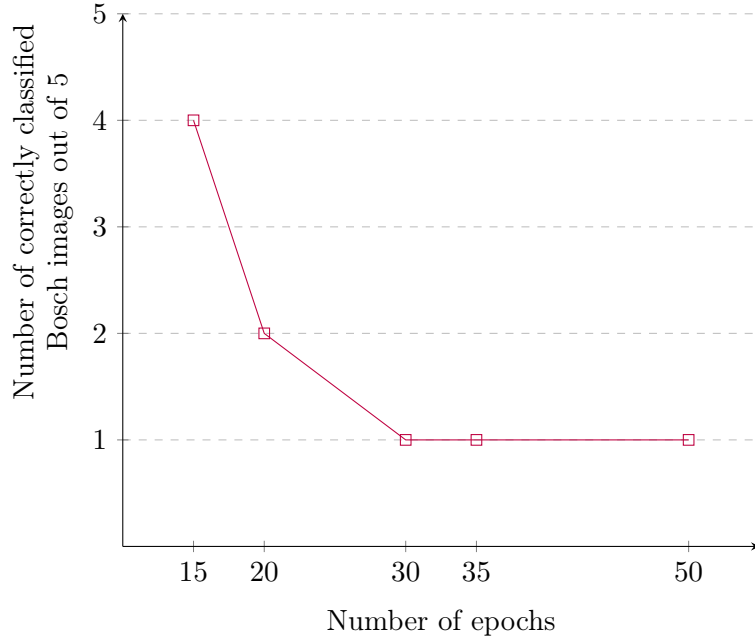
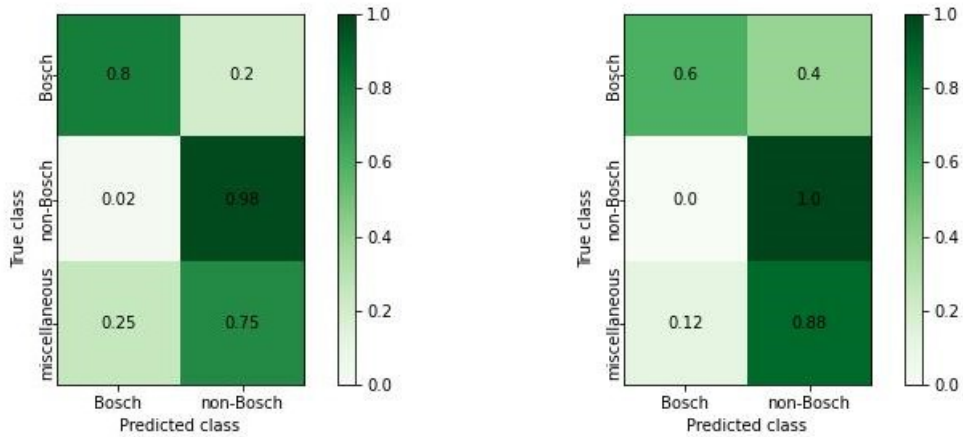


Figure 4.6: *Decline in the performance of EffNetB2 on big epoch numbers*

epochs, so this is the most optimal model state for EffNetB2.

After that, I intended to replicate this experiment on another version, to see whether the optimal epoch number stays the same. Since there is no difference in the performance on versions B3-B6, as seen in Figure 4.4, regarding the classification of Bosch images, I examined the other indicators of performance. As illustrated in Figure 4.3, both B0 and B5 demonstrated significant improvements after the set-size change. Specifically, B0 showed the best improvement after the set-size readjustment in comparison to all other versions. Also, B5 exhibited notable improvement in relation to its own previous performance, as detailed in matrices 4.2. Conventionally, the simpler version, in this case, B0, would be chosen. However, I chose B5 for further experimentation due to the potential benefits that a higher version could offer, especially in light of future work.



(a) *EfficientNetB2, 15 epochs*

(b) *EfficientNetB5, 15 epochs*

Figure 4.8: *Confusion matrices of the most optimal states*

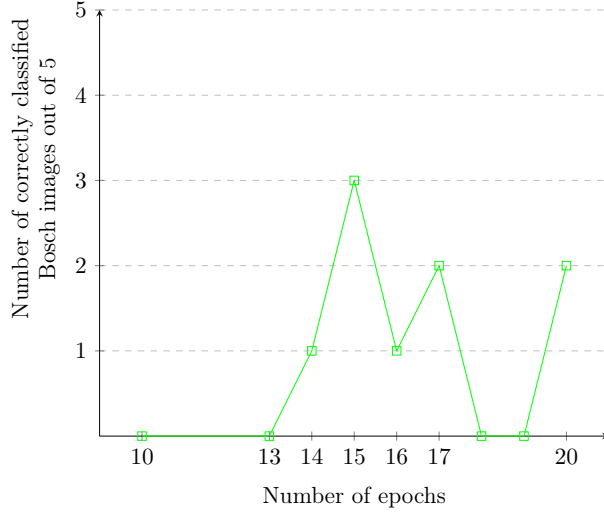


Figure 4.7: *Performance of EffNetB5 on different epoch numbers*

As clearly shown in Figure 4.7, the best epoch number was indeed 15 on version B5 as well. After 20, I retrained it for 25 and 40, and the performance stabilized at 2, implying that the peak was reached at 15 epochs as well, with a total of 3 out of 5 correctly classified Bosch images. In Figure 4.8 can be seen the final results on the test set produced by the most optimal states by the chosen models.

4.4 Evaluating the results on the miscellaneous set

In this section, I further examined the performance outcomes of the finest trained EfficientNet models, versions B2 and B5, each enhanced with an additional layer and fine-tuned for the optimal number of epochs, which was 15 in both cases. The focus is on the miscellaneous set, a subset of the test set that has been discussed less frequently so far in this work. The core objective is to ascertain whether the models exhibit a tendency to classify those paintings as Bosch that bear a substantial connection with Bosch — those that *e.g.*, had physical connections with him — as opposed to those that maintain only a superficial relationship with him.




Table 4.1 presents an overview of the paintings contained in the miscellaneous set. A vital piece of information provided in the table is the supposed creator of each painting, indicated in the Creator column. These creator identities have been extracted directly from the respective sources of the paintings, boschproject.org and the website of JBAC, the authenticity of which is upheld by art experts who originally cataloged or examined these works. It is important to note that these names are replicated as they appear on the source websites, preserving the original naming conventions and attributions of the art experts.

The highlighted words in the Creator column of Table 4.1 refer to the level of relationship between the given artwork and Bosch. The phrase “School of Bosch” is the closest to what we have classified as followers: the artists may or may not be students of Bosch, but they

Table 4.1: *All paintings in the uncertain set*

Title	Creator
The Flood Panels (4 images)	Workshop of Jheronimus Bosch
Job Triptych	Workshop of Jheronimus Bosch
Ecce Homo Triptych	Workshop of Jheronimus Bosch
The Adoration of the Magi	Workshop of Jheronimus Bosch
The Cure of Folly	Workshop <i>or</i> follower of Jheronimus Bosch
The Seven Deadly Sins and the Four Last Things	Workshop <i>or</i> follower of Jheronimus Bosch
The Last Judgment (interior central)	Jheronimus Bosch and Workshop
The Adoration of the Magi (exterior panels)	The Master of Anderlecht - <i>or</i> School <i>or</i> Circle of Jheronimus Bosch
The Temptations of St Anthony	anon. - Circle of Jheronimus Bosch - probably influences from Antwerp
The Temptations of St Anthony	anon. - School of Jheronimus Bosch
Christ's Descent into Limbo	anon. - Circle of Jheronimus Bosch
St Christopher Carrying the Christ Child	anon. - Circle of Jheronimus Bosch
The Garden of Earthly Delights (copy of the left interior panel)	anon. - Workshop of Jheronimus Bosch <i>or</i> Michiel Coxie
The Garden of Eden	anon. - Circle of Jheronimus Bosch
The Garden of Eden	anon. - Circle of Jheronimus Bosch
St Job Triptych	anon. - Circle of Jheronimus Bosch
The Last Judgement	anon. - Circle of Jheronimus Bosch
The Last Judgement (fragmentary wings)	anon. - Circle of Jheronimus Bosch
The Last Judgement (triptych)	anon. - <i>probably</i> (Workshop) Jheronimus Bosch
Two Jewish Priests (?)	anon. - Circle of Jheronimus Bosch - The Master of the Fountain of Life

Table 4.2: *Classification results for Bosch-related images by EffNetB2*

Category	Model Classification	
	Non-Bosch	Bosch
Workshop of Bosch		
	3.5 ¹	3.5
Circle of Bosch		
	8	1
School of Bosch		
	-	1
School/Circle of Bosch	1	-
Workshop/Follower of Bosch	3	-

were part of a large artistic movement that aimed to create in the style of Bosch. The main difference between these types of paintings and the ones that are labeled as made by followers is that the School of Bosch did not want to copy Bosch’s paintings (or fake Bosch’s signature on their works as some followers did), but rather create something that would make the spectator think about Hieronymus Bosch.




“Circle of Bosch” is similar, but it refers to artists that Bosch probably knew personally. The Circle of Bosch was a group of artists who worked closely with Bosch, some even in the same workshop or were friends with him. The physical closeness naturally led to stylistic influence, which made these paintings often come up in conversations regarding Bosch.

The label “Workshop of Bosch” indicates the closest relation. Being a renowned artist in his lifetime, Bosch had a workshop where aspiring artists had the opportunity to become his students and learn from him. However, such a workshop back then did not always aim to motivate the students to find their own styles but rather to imitate the master. More often than not, the master had ideas that he did not want or did not have the time to paint by himself, so he assigned them as tasks to his students - we could even say that the students worked instead of him. Such artworks may have been signed by Bosch himself or even had other brushstrokes by him [18].

The results on the miscellaneous set cannot be considered a main indicator of the performance of the model, however, I was interested in how it would classify unambiguous images and whether in the classification I can see the slight differences described above. I would expect that most paintings classified as Bosch were made by his workshop, a small number of paintings made by his circle and all the others are more likely to be regarded as non-Bosch.

¹The workshop painting B23 consists of 4 panels, all on different images.

Table 4.3: *Classification results for Bosch-related images by EffNetB5*

Category	Model Classification	
	Non-Bosch	Bosch
Workshop of Bosch		
	4.75 ¹	2.25
Circle of Bosch		
	9	-
School of Bosch		
	1	-
School/Circle of Bosch	1	-
Workshop/Follower of Bosch	3	-

In Table 4.2, the categories are represented with a corresponding color bar showing where the category could be placed between Bosch and non-Bosch pictures based on the meaning of the categories. To my delight, the results on the miscellaneous set produced by EffNetB2 corresponded to this expectation. The highest number of paintings classified as Bosch was among those created in his workshop, while paintings associated with the School and Circle of Bosch had much lower numbers. The images that might have been made by followers were all labeled as non-Bosch. This result strengthened my assumption that the model gained a close understanding of what a Bosch artwork is.

Similarly, Table 4.3 summarizes the performance of EffNetB5 on the miscellaneous set, with the same representing color bars. The model had a tendency toward classifying as non-Bosch. This corresponds to the results of the confusion matrix shown in Figure 4.8b, where the rate of being labeled as non-Bosch is considerably higher than on confusion matrix shown in Figure 4.8a. However, we can still witness an understanding of the features of a Bosch painting, since artworks from Bosch’s workshop were labeled as Bosch.

4.5 Examining the Misclassifications in Bosch Paintings

Upon scrutinizing the outputs of the best-performing models, it was at first surprising to find that one of Bosch’s most iconic works, *The Garden of Earthly Delights*, was misclassified by both version B2 and B5. In fact, it was one of the most misidentified paintings by other versions, too. While I anticipated that this distinct representation of Bosch’s style would be easily recognized, a high number of its copies were present in the dataset. Specifically, 7 replicas of the *Earthly Delights* were in the follower set, with 6 incorporated in the training set, two of which are displayed in Figure 4.9. Therefore, I certainly believe that being the most emblematic work of Bosch and thus having the most copies, was the key



Figure 4.9: *Two copies of The Garden of Earthly Delights in the training set*

factor in the misclassification of *The Garden of Earthly Delights*. The one copy of it in the test set was correctly identified as non-Bosch. For the other 4 Bosch paintings in the test set, no copies were part of the follower set.

By EffNetB5, the other incorrectly categorized piece was *Calvary with Donor*, seen in Figure 4.10a. Despite having no copies within the training set, this painting posed more of a challenge, as it does not overtly exhibit Bosch's renowned otherworldly style.



(a) *Calvary with Donor*

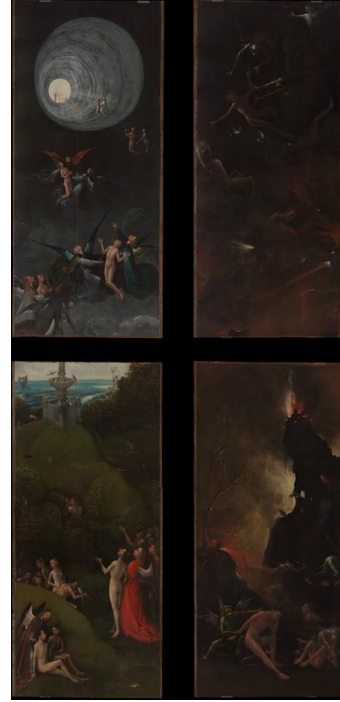


(b) *The Wayfarer*

Figure 4.10: *Paintings in the test set*



(a) *The Death and the Miser*



(b) *Visions of the Hereafter*

Figure 4.11: *Paintings in the test set*

In contrast, three paintings, *Visions of the Hereafter*, *The Wayfarer*, and *Death and the Miser*, displayed in Figures 4.10b and 4.11, were correctly classified. The successful recognition of *Visions of the Hereafter*, a piece that reflects Bosch’s distinctive style, and the detection of *The Wayfarer* and *Death and the Miser*, which are less explicit, underscore the model’s capability to discern varying representations.

5

Conclusion

This study was aimed at constructing a binary classifier using convolution neural networks (CNNs) capable of distinguishing between original Bosch paintings and their imitations. It also provided a broader understanding of the application of CNNs in fine art classification.

The methodology involved using the pre-trained CNNs, EfficientNet models versions B0-B6, with further training on a unique dataset. To ensure the accuracy of the dataset, it was extracted from the website of Jheronimus Bosch Art Center and boschproject.org, with paintings labeled by art experts. To tackle the class imbalance in the training set, common data augmentation techniques were employed for oversampling.

This thesis concluded that the classification problem at hand is resolvable. Additionally, the classifier was even able to give an insight into the miscellaneous set. Replicating steps used in previous art classification studies was a key element in this project, demonstrating how effective the methodology for fine art classification, such as additional training and the inclusion of an extra layer, indeed is. Furthermore, the project offered a unique contribution to the field by classifying Bosch-like paintings solely based on the images and their most basic labels, bypassing the need for additional information often required in this field.

The study also highlighted certain limitations, notably the ones arising from the limited number of Bosch paintings available. This situation inevitably led to complications due to the class imbalance in the training set. Apart from employing data augmentation to address the imbalance in the training set as much as possible, reassurance was gained from the results on the miscellaneous set, verifying the accuracy in the face of the small number of test images. These challenges underscored the constraints inherent in such a

project, given that these methods cannot completely make up for the insufficient amount of available data.

Despite these constraints, this thesis underscores the potential of CNNs in fine art classification, specifically the classification of Bosch-like paintings based on limited information. It sets a precedent for further studies in this direction, refining the classifiers' performance through various, more complex methodologies.

5.1 Future work and outlook

This project was focused on the visual characteristics of paintings that can help distinguish them from each other. However, there are numerous other ways to explore paintings via machine learning, such that brushstroke analysis or X-ray images. Data from the latter is also available as open-source information about Bosch paintings, which may help to refine the results of this project in the future.

Brushstroke analysis is another broad part of painting classification that I have a great interest in. For example, the uniqueness of Van Gogh's brushstrokes can be well-identified [8]. The ratio of the direction of the brushstrokes can also be a main indicator of the authentication process since it determines the handedness of the artist. Indeed, it is believed that a left-handed student in Bosch's workshop is the actual creator of half of the paintings originally attributed to the master [5].

In order to tackle the problem of a small data set and especially a small test set, cross-validation was also considered. However, the attempt proved to be infeasible because of the uniqueness of my dataset. Introducing cross-validation proved to be outside the scope of my thesis, but it is a possible future step to continue this project. Similarly, the exploration of EffNetB7, the most advanced version of EfficientNet, and EfficientNetV2 [17], a new family of convolution neural networks derived from EfficientNet, was also beyond the scope of this work. These areas could provide interesting avenues for future research.

Acknowledgment

I would like to thank my fiancée for encouraging me.

Project no. TKP2021-NVA-02 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

List of Figures

1.1	The Garden of Earthly Delights triptych (between 1490 and 1510)	5
1.2	The “same” paintings by different artists	9
2.1	A fully-connected ANN with 2 hidden layers	11
2.2	The block structure of A: ResNet, B: Res2Net, C: ResNeXt, D: ResNest source: [21]	13
2.3	The structure of the classification model in [12]	17
3.1	Flowchart illustrating the image augmentation process using the painting called <i>The Temptation of Saint Anthony</i> by Hieronymus Bosch	20
3.2	Sorting of the paintings	21
4.1	Confusion matrix of the results of EffNetB4, with required input sizes, but non-equal set sizes	26
4.2	Improvement of EffNetB5 after set-size change	27
4.3	Difference in performances with unequal and equal set sizes	27
4.4	Performance change after adding a layer	29
4.5	The loss functions of EffNetB2, trained on 50 epochs with new layer	29
4.6	Decline in the performance of EffNetB2 on big epoch numbers	30
4.8	Confusion matrices of the most optimal states	30
4.7	Performace of EffNetB5 on different epoch numbers	31
4.9	Two copies of <i>The Garden of Earthly Delights</i> in the training set	35
4.10	Paintings in the test set	35
4.11	Paintings in the test set	36

Codes

4.1	Pseudocode of the training code	24
4.2	Orginal last classification layer	28
4.3	New last classification layer	28

List of Tables

2.1	Results of EffNet models with different pre-training strategies [12]	17
4.1	All paintings in the uncertain set	32
4.2	Classification results for Bosch-related images by EffNetB2	33
4.3	Classification results for Bosch-related images by EffNetB5	34

Glossary

ANN artificial neural network. 1, 7, 9–12, 40

anon. anonymus. 32

CNN convolution neural network. 1, 6–14, 16, 19, 37, 38

EffNet EfficientNet. 1, 8, 10, 15–17, 23, 25–31, 33–35, 37, 38, 40, 41

FC fully-connected. 10–15, 17, 28, 40

ReLU Rectified Linear Unit. 12, 14, 17

ResNet Residual Neural Network. 8, 13, 15, 16, 40

Bibliography

- [1] Siddharth Agarwal, Harish Karnick, Nirmal Pant, and Urvesh Patel. Genre and style based painting classification. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 588–594. IEEE, 2015.
- [2] Simone Bianco, Davide Mazzini, Paolo Napoletano, and Raimondo Schettini. Multitask painting categorization by deep multibranch neural network. *Expert Systems with Applications*, 135:90–101, 2019.
- [3] Walter Bosing. *Bosch: Sämtliche Gemälde*. Taschen, Köln, 2000.
- [4] Todd Dobbs and Zbigniew Ras. On art authentication and the Rijksmuseum challenge: A residual neural network approach. *Expert Systems with Applications*, 200:116933, 2022.
- [5] Matthijs IJl sink and Mark Broch. Hieronymus Bosch: Die Zeichnungen. *Master Drawings*, 51(3):393–407, 2013.
- [6] Matthijs IJl sink and Jos Koldeweij. *Jheronimus Bosch: Visioenen van een genie*. Mercatorfonds, 's-Hertogenbosch, 2016.
- [7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [8] C Richard Johnson, Ella Hendriks, Igor J Berezhnoy, Eugene Brevdo, Shannon M Hughes, Ingrid Daubechies, Jia Li, Eric Postma, and James Z Wang. Image processing for artist identification. *IEEE Signal Processing Magazine*, 25(4):37–48, 2008.
- [9] Fahad Khan, Shida Beigpour, Joost van de Weijer, and Michael Felsberg. Painting-91: A large scale database for computational painting categorization. *Machine Vision and Application (MVAP)*, 25(6):1385–1397, 2014.
- [10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [11] Jia Li and James Ze Wang. Studying digital imagery of ancient paintings by mixtures of stochastic models. *IEEE Transactions on Image Processing*, 13(3):340–353, 2004.

- [12] Baya Lina Menai and Mohamed Chaouki Babahenini. Recognizing the style of a fine-art painting with EfficientNet and Transfer learning. In *2022 7th International Conference on Image and Signal Processing and their Applications (ISPA)*, pages 1–6. IEEE, 2022.
- [13] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122. IEEE, 2018.
- [14] Peter Murray and Linda Murray. *A dictionary of Christian art*. (Oxford paperback reference). Oxford University Press, Oxford, 2001. Bibliogr.: p. [635]–646.
- [15] Laura Ritter. The Making of Bosch: Observations on his artistic reception. In *Jheronimus Bosch-His Life and His Work: 4th International Jheronimus Bosch Conference, April 14-16, 2016, Jheronimus Bosch Art Center, 's-Hertogenbosch, The Netherlands*, 2016.
- [16] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.
- [17] Mingxing Tan and Quoc Le. EfficientNetV2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021.
- [18] Gerd Unverfehrt. *Hieronymus Bosch: die Rezeption seiner Kunst im frühen 16. Jahrhundert*. Mann, 1980.
- [19] Vilson Vieira, Renato Fabbri, David Sbrissa, Luciano da Fontoura Costa, and Gonzalo Travieso. A quantitative approach to painting styles. *Physica A: Statistical Mechanics and its Applications*, 417:110–129, 2015.
- [20] James Z. Wang, Baris Kandemir, and Jia Li. *Computerized Analysis of Paintings*, page 14. Routledge, 1 edition, 2020.
- [21] Wentao Zhao, Dalin Zhou, Xinguo Qiu, and Wei Jiang. Compare the performance of the models in art classification. *PLOS ONE*, 16(3):1–16, 03 2021.