Gheorghe Asachi Technical University of Iaşi
Faculty of Automatic Control and Computer Engineering
Department of Automatic Control and Applied Informatics

MACHINE LEARNING PROJECT

Classification of Artistic Styles Using the VGG-16 Convolutional Neural Network

Authors
Denisa-Gabriela MUSTEAȚĂ

Iași, 2025

Contents

1 Abstract

This project explores the application of deep learning techniques in the field of art style classification. Utilizing the pre-trained VGG16 convolutional neural network, the study aims to classify images of artwork based on their artistic styles.The VGG16 model, known for its exceptional performance in image recognition tasks, was adapted by leveraging transfer learning to fine-tune it for a specialized dataset of artworks. The dataset comprised images categorized by styles such as Impressionism, Cubism, Baroque, Abstract Expressionism, Ukiyo e and Realism. The model was trained and validated to ensure accurate prediction and effective generalization. A robust preprocessing pipeline was implemented, including reducing the dataset, resizing and normalization to enhance model performance and mitigate overfitting. Performance metrics, including accuracy, precision, recall, and F1-score, were employed to evaluate the classification results. The findings demonstrate that VGG16 can successfully classify artwork styles with high accuracy. The method combines the high-resolution images from the WikiArt dataset with the high performance of the VGG16 network, which led to the achievement of encouraging results, suggesting that, with further adjustments, the results can be significantly improved, offering valuable insights into the intersection of deep learning and artistic style analysis.

2 Introduction

VGG16, proposed by Simonyan and Zisserman in 2014, is a convolutional neural network model known for its deep and uniform architecture. The model uses convolutional layers with fixed 3x3 filters and a ReLU activation function to introduce nonlinearity, contributing to the stability and performance of the learning process. Additionally, max pooling is employed to reduce spatial dimensions while preserving essential information, and the fully connected layers at the end are used for image classification. VGG16 achieved remarkable performance in the ILSVRC-2014 competition, becoming a reference point for subsequent neural network architectures. The classification of artistic styles using machine learning represents a challenging yet fascinating task in the field of computer vision and artificial intelligence. This project explores the use of a convolutional neural network (CNN) based on the VGG-16 architecture to classify artworks into distinct artistic styles. The choice of VGG-16 is motivated by its robustness and proven effectiveness in handling large-scale image classification tasks.

The dataset employed in this study consists of high-quality images, meticulously curated to ensure the representation of six distinct artistic styles: Baroque, Realism, Impressionism, Abstract Expressionism, Cubism, and Ukiyo-e. The goal of this project is to leverage transfer learning techniques to train a custom model capable of accurately differentiating between these styles, even when significant visual similarities exist among them.

This work involves several stages, including data preprocessing, architectural modifications to the pre-trained VGG-16 model, and multiple rounds of training and evaluation. Special attention is given to analyzing the network's performance through confusion matrices and performance metrics such as accuracy on training, validation, and testing datasets.

The results obtained demonstrate the challenges associated with classifying visually similar artistic styles and highlight the potential of advanced deep learning techniques in addressing these challenges. Additionally, a two-level classification strategy is proposed to improve the network's ability to handle confusion among specific classes, such as Baroque, Realism, and Impressionism.

The structure of the paper is as follows:

1. **Chapter 1:** Abstract
2. **Chapter 2:** Introduction
3. **Chapter 3:** Theoretical Foundations
4. **Chapter 4:** Classification Using the VGG-16 Algorithm
5. **Chapter 5:** Experimental Results
6. **Chapter 6:** Conclusions

3 Theoretical Foundations

Machine learning emerged with the desire to replicate a series of natural phenomena and human cognitive processes, most of which occur at the level of nerve cells in the human brain. These include mechanisms such as: thinking patterns, learning methods, perception of the surrounding environment, problem-solving, analytical thinking, solution identification, and more. The study of these biological mechanisms, followed by their abstraction and foundation in mathematical theories, led to the creation of various information processing methods.

Machine learning (ML) encompasses a wide range of methods and algorithms designed to enable computers to learn from data and make decisions or predictions without explicit programming. These methods can be broadly categorized into supervised, unsupervised, and reinforcement learning. Further, each of the categories will be detailed in a succinct manner.

Supervised learning algorithms, such as linear regression, decision trees, and support vector machines, rely on labeled training data to build a model that can predict outputs for unseen data. These methods are particularly effective when the relationship between input features and target labels is well-defined and structured. For example, classification tasks, where the goal is to assign input data to predefined categories, and regression tasks, where the objective is to predict continuous values, are commonly addressed through supervised learning.

Unsupervised learning algorithms, including clustering techniques like k-means and dimensionality reduction methods such as principal component analysis, operate on unlabeled data. These algorithms aim to discover hidden patterns or structures within the data without the need for explicit target labels. Unsupervised learning is particularly valuable in exploratory data analysis, anomaly detection, and feature extraction.

Reinforcement learning represents a class of algorithms where an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on its actions, and through trial and error, it strives to maximize cumulative rewards. RL has shown remarkable success in complex decision-making tasks such as game playing, robotics, and autonomous systems.

Each machine learning method and algorithm has its strengths and limitations, and the choice of the appropriate approach depends on the nature of the data, the task at hand, and the computational resources available. Moreover, the effectiveness of machine learning models often hinges on the quality of the data, feature engineering, and the ability to fine-tune hyperparameters to achieve optimal performance.

3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models that have demonstrated remarkable performance in various computer vision tasks, such as image classification, object detection, and segmentation. Inspired by the biological visual processing system, CNNs leverage local connectivity and shared weights to capture spatial hierarchies in data (LeCun et al., 1998). The core idea behind CNNs is to apply convolution operations to input data, which allows the model to automatically and adaptively learn spatial features at multiple levels of abstraction.

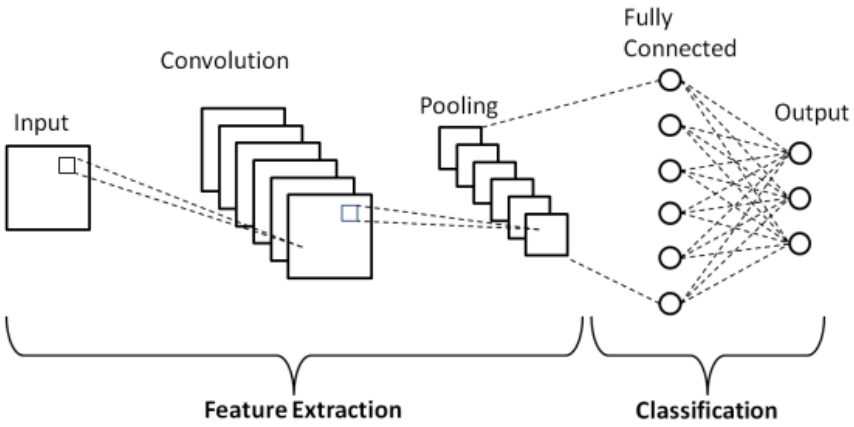


Figure 3.1: Diagram of a classic convolutional neural network structure – architecture.

In figure 3.1 is illustrated the structure of a Convolutional Neural Network (CNN), showcasing the key stages involved in the process of feature extraction and classification. The architecture can be divided into two primary components:

1. **Feature Extraction**

   - **Input Layer:** The input layer is the starting point of a CNN, receiving raw data (e.g., images) and preparing it for further processing by the network.
   - **Convolutional Layer:** The convolutional layer is the core of a CNN, applying filters to the input to extract local features such as edges, textures, and patterns.
   - **Pooling Layer:** The pooling layer reduces the spatial dimensions of feature maps, retaining important information while improving computational efficiency and robustness.

2. **Classification**

   - **Fully Connected Layer:** The fully connected layer connects all neurons, combining extracted features to make final predictions or classifications.
   - **Output Layer:** The output layer in a CNN provides the final prediction by processing features extracted from previous layers, often using activation functions like softmax or sigmoid.

The following will briefly detail the stages through which the data pass from their entry into the system to the provision of the identified classes at the output:

Feature extraction follows a few simple steps. The input data is illustrated on the left side of the figure. It is passed to the convolutional layers for processing. The convolutional layers apply filters, also known as kernels, to the input data to extract features. Each convolutional layer produces a feature map that highlights specific characteristics of the image.

After passing through the convolutional layers, the data is sent to the pooling layers. While essential information is retained, pooling reduces the spatial dimensions of the feature maps and also lowers computational complexity. Max pooling is an operation that occurs within the pooling layers where the maximum value from each region of the map is selected and evaluated.

Classification is the second stage in the evaluation process of a convolutional neural network and takes place in the fully connected layers and the output layer. After the first stage, where features have been extracted and reduced dimensionally, the information is passed to the fully connected layers. In

these layers, each individual neuron is connected to every neuron in the previous layer, thus combining the extracted features for the classification process.

The task of making the final prediction is assigned to the last layer.

3.2 VGG-16 Network

The VGG16 (Visual Geometry Group 16) network is a deep convolutional neural network architecture proposed by the Visual Geometry Group at the University of Oxford. It gained significant attention and popularity for its simplicity and effectiveness in various computer vision tasks, especially image classification.

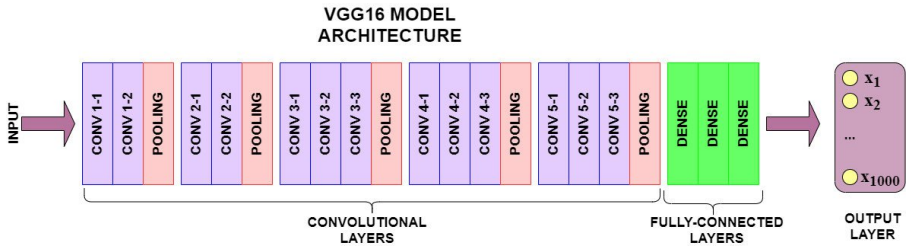


Figure 3.2: VGG-16 Network Architecture.

VGG16 is characterized by its uniform architecture comprising 16 layers, including 13 convolutional layers and 3 fully connected layers. The architecture uses small 3x3 convolution filters throughout the convolutional layers, which allows it to learn rich feature representations. Each convolutional layer uses small 3x3 filters and is followed by a ReLU (Rectified Linear Unit) activation function, which introduces nonlinearity into the model.

The ReLU activation function is mathematically defined as:

$$f(x) = \max(0, x) \tag{3.1}$$

It is widely used in neural networks due to its ability to accelerate convergence during training and to avoid the vanishing gradient problem. In VGG16, ReLU is applied after each convolutional and fully connected layer, contributing to improved performance and stability in the learning process.

The VGG16 architecture consists of alternating convolutional and max-pooling layers, followed by three fully connected layers and a softmax classifier. The Softmax function is mathematically defined as:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3.2}$$

where z_i represents the output value for class i and $\sum_j e^{z_j}$ is the sum of the exponentials of all the output values. Softmax transforms the network's output values into a range between 0 and 1, so that the sum of all output values equals 1, interpreting them as probabilities. This property makes Softmax extremely useful in tasks like image classification, where each class is associated with a probability reflecting the network's confidence in assigning an image to that class.

This feature makes it extremely useful in tasks such as image classification, where each class is associated with a probability that reflects the network's confidence in an image belonging to that class.

VGG-16 was pre-trained on the ImageNet dataset, which consists of over 14 million images belonging to 1000 classes. During training, the network parameters were optimized using stochastic gradient descent (SGD) with mini-batch size of 256, momentum of 0.9, and weight decay of 0.0005. The network achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014, significantly advancing the accuracy of image classification tasks.

Due to its robust architecture and pretrained weights on large datasets like ImageNet, VGG16 has become a popular choice for transfer learning in various domains. Researchers and developers often leverage its learned features by fine-tuning the network on smaller datasets or adapting it to specific tasks such as object detection, image segmentation, and even medical image analysis. This ability to transfer knowledge from pretrained models has accelerated progress in computer vision applications.

VGG offers several advantages, such as its simplicity, which makes it an excellent baseline for experiments and a popular choice for transfer learning. Its architecture has demonstrated impressive results across a variety of computer vision tasks, proving its robustness and reliability. However, the large number of parameters in VGG makes it prone to overfitting, particularly when the available dataset is not sufficiently large. Additionally, VGG is less efficient in terms of memory usage and computational time compared to more modern architectures. It also lacks advanced techniques like skip connections, which have contributed to improved performance and efficiency in newer models.

Training VGG models, particularly deeper versions like VGG-19, is computationally expensive due to the large number of parameters, approximately 138 million. This results in the need for more memory and extended training times. In terms of inference speed, VGG models are slower when compared to optimized architectures such as MobileNet or EfficientNet, which are designed to maximize efficiency. As a result, VGG is less suited for real-time applications where faster processing is critical. Despite these inefficiencies, VGG excels in tasks such as image classification and object detection, especially when fine-tuned on large datasets like ImageNet.

While VGG16 is noted for its accuracy and simplicity, its main drawbacks stem from its deep architecture, which requires significant computational resources. The model's large parameter count makes it more resource-intensive than other architectures like AlexNet or ResNet. Despite these challenges, VGG16 remains a foundational model in the development of convolutional neural networks, influencing the design and improvement of subsequent models in deep learning.

3.3 WikiArt Dataset

WikiArt is an online visual art dataset that offers access to a vast collection of artworks from various time periods, styles, and cultures. It serves as a platform for art enthusiasts, researchers, and historians to explore and discover thousands of works of art, spanning from classical to contemporary pieces. WikiArt provides detailed information about artists, their biographies, and the contexts in which their artworks were created, making it a valuable resource for both education and personal exploration. The platform also allows users to browse artworks by genre, school, or medium, offering a rich and diverse view of the global art landscape. As an open and accessible resource, WikiArt contributes to the democratization of art appreciation and historical knowledge.

4 Classification Using the VGG-16 Algorithm

In this section, we focus on image classification using the VGG16 algorithm, applied to the WikiArt dataset. By leveraging the pre-trained VGG16 model, we aim to classify artwork images based on their artistic style or other relevant categories.

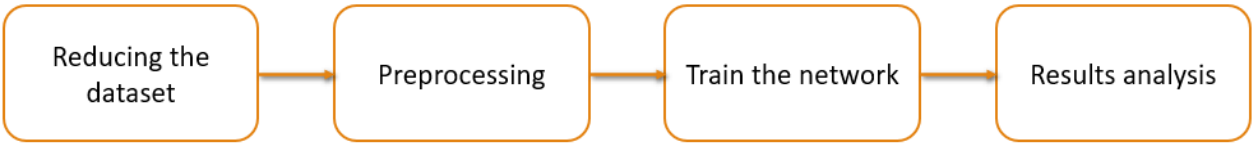


Figure 4.1: Steps followed during the development of the project.

A preliminary step in the development of this project was the reduction of the dataset. WikiArt contains 27 classes and over 81,000 images, which was too large a volume for the limited resources of the laptop used. Therefore, in this stage, I analyzed the dataset and, to simplify the classification process, selected 6 classes (approximately 2,000 images) that were not highly similar to each other.

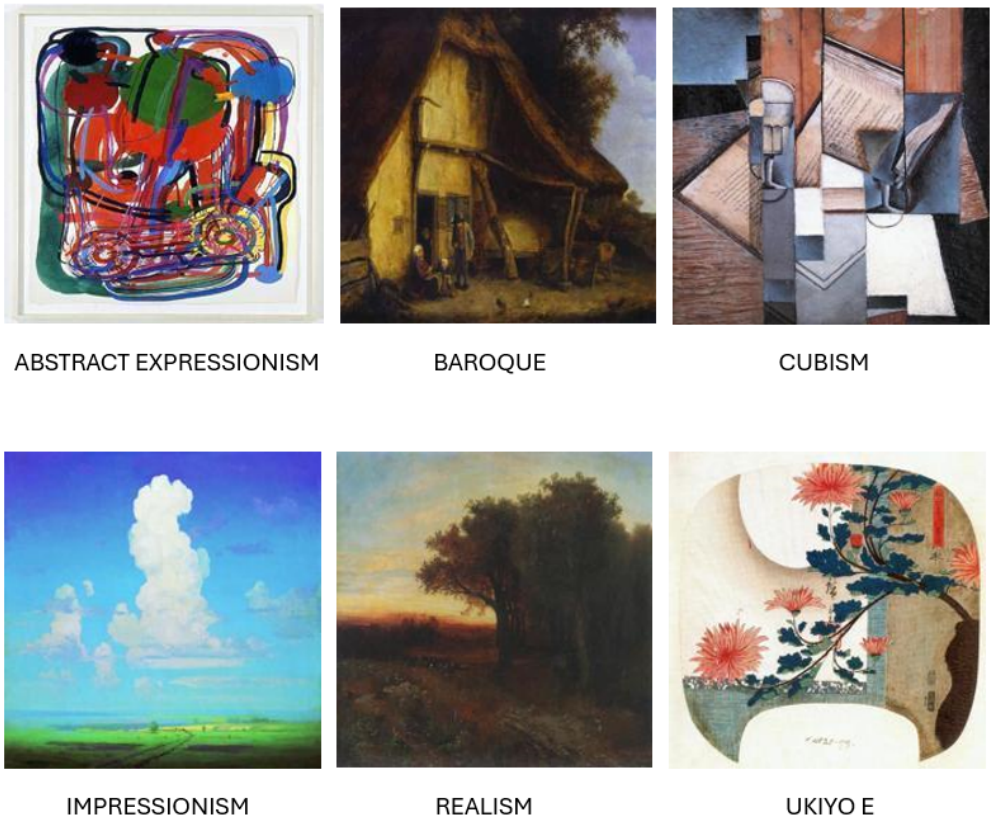


Figure 4.2: Samples from each class.

In Figure 4, I have illustrated one example from each class, providing a direct visualization of the differences between artistic styles. As can be observed, certain styles share similarities, which is why in Chapter 5, I analyzed the results obtained and drew various conclusions. The images used are

of high quality and resolution, so in the early training stages, I did not consider the need to apply specific filters to highlight details between classes. VGG-16 is a robust architecture, and given the high-quality dataset, I assumed that the results obtained would not be significantly affected.

However, a slight preprocessing of the dataset was necessary before the actual training of the model. This involved resizing the images to 224x224x3 to make them compatible with the input layer of the neural network, as well as splitting the dataset into training, testing, and validation subsets. For this step, 70% of the initial dataset was kept for training, with the remaining portion equally distributed to the validation and testing subsets.

Then the training process of a convolutional neural network (CNN) using the VGG16 architecture for image classification was implemented.

The pre-trained VGG16 model was used, and learning was transferred from the network's last layers, keeping only the base layers to leverage the knowledge gained from large datasets like ImageNet. A new fully connected layer was added, adjusted to the number of classes in the dataset.

For training, the Adam optimizer was employed with predefined parameters such as mini-batch size, number of epochs, and learning rate. During training, the validation dataset was used to monitor the network's performance. After the network was trained, the model's accuracy was evaluated on the training, validation, and testing datasets, with the results indicating its performance in correctly classifying the images.

The modifications made in the code are as follows:

line 14: inputSize = [224 224 3];

line 71: options = trainingOptions('adam', ...

lines 106-108: figure; confusionchart(imdsValidation.Labels, YPredValidation); title('Confusion Matrix - Validation Dataset');

During the training process, several configurations of the dataset were analyzed based on the results obtained from the confusion matrices. The first round of training was performed using all six classes. After evaluating the performance, significant confusion was observed between three different artistic styles (Baroque, Realism, and Impressionism) in the validation set. Consequently, a decision was made to eliminate the "Baroque" class initially, followed by the removal of the other two classes (Realism and Impressionism), with successive trainings performed on smaller subsets of the dataset. Thus, the second round of training was conducted using only five classes, and the other two trainings were carried out with four classes.

However, the goal was to achieve satisfactory results across all six classes. The methodological steps implemented in this case were as follows: two of the more confusing classes (Baroque and Realism) were combined for the first round of training. After obtaining initial results, these two classes were separated and a second training was conducted specifically focused on these two artistic styles. The results obtained from each training stage are detailed and discussed in Chapter 5 of the paper.

4.1 Two-Level Classification using VGG-16

During the development of this part of the code, the following were implemented: loading the pre-trained network, defining parameters, preparing the data, defining the network architecture, freezing the weights of the base layers, training the CNN, verifying the results. The following steps will be explained in detail, step by step.

1. **Loading the Pre-trained Network:** The pre-trained network is loaded from a file named rezCNN.mat using the load function. The network is saved in a variable called netTransfer. This network is a pre-existing model that will serve as the base for transfer learning, where it will be adapted to the specific dataset.

    - **line 113:** load('rezCNN.mat', 'netTransfer');

2. **Defining Parameters:** The training parameters are defined as follows: the input image size is set to 224x224x3, which aligns with the input requirements of the VGG16 network. The mini-batch size (MBS) and the number of epochs are also set to 32 and 10, respectively. These features were not modified compared to the original code.

3. **Preparing the Data:** The same steps as in the original code were followed.

4. **Defining the Network Architecture:** A pre-trained VGG16 model is used, and the last three layers are removed. A new architecture is created by adding custom layers: a fully connected layer, with the number of output units corresponding to the number of classes in the dataset, a softmax layer to calculate the probabilities for each class, and a classification layer to compare the predicted label with the true label. No additional modifications were made to the initial code for this section.

5. **Freezing the Weights of the Base Layers:** To accelerate the training process, the weights and biases of the transferred layers are frozen, meaning they will not be updated during training. This includes the convolutional and fully connected layers, whose weights are frozen, allowing the focus to be on adjusting the newly added layers.

    - **line 150:** for i = 1:numel(layersTransfer)
    - **line 151:** if isa(layersTransfer(i), 'nnet.cnn.layer.Convolution2DLayer') || ...
    - **line 152:** isa(layersTransfer(i), 'nnet.cnn.layer.FullyConnectedLayer')
    - **line 153:** layersTransfer(i).WeightLearnRateFactor = 0;
    - **line 154:** layersTransfer(i).BiasLearnRateFactor = 0;
    - **line 155:** end
    - **line 156:** end

6. **Training the CNN:** The network is trained using the following settings: the Adam optimization algorithm is employed, with parameters such as mini-batch size, number of epochs, learning rate, and validation options. The network is trained on the training image set using the defined architecture. Once the training is complete, the trained model is saved in a file (rezCNN2.mat) for future use.

7. **Verifying the Results:** After training, the network's performance is evaluated on the validation, testing, and training sets. The validation accuracy is determined by comparing the predicted labels with the true labels. Accuracy is also calculated for the training and testing sets. Additionally, a confusion matrix is generated for the validation set to visualize classification errors and assess the overall performance. Here, as well, no additional modifications were made to the initial code for this section. A detailed explication of this section can be read in the Chapter 5.

5 Experimental Results

Nr. crt.	**Nr. Classes**	**[NEP, MBS]**	**Train (%)**	**Test (%)**	**Validation (%)**	**Observations**
1	6	[10, 32]	90.04	59.56	60.13	Dataset with all six classes
2	5	[10, 32]	95.37	78.57	78.23	Dataset without the Baroque class
3	4	[10, 32]	98.37	91.70	86.27	Dataset without Baroque and Realism classes

Table 5.1: Experimental results for the first three trainings with different configurations of the dataset

The table 5.1 presents the results of three experiments conducted on datasets with different numbers of classes, with the goal of evaluating the performance of a machine learning model. In the first experiment, all 6 classes are used, while in the next two experiments, the Baroque class and, respectively, both the Baroque and Realism classes are excluded, resulting in more restricted datasets. These changes in the data suggest a progressive testing approach on increasingly simplified subsets.

As the number of classes decreases, the model's performance appears to improve, with better test and validation percentages. This suggests that the model performed more efficiently on simpler datasets, with a better ability to learn and generalize on smaller subsets of information. This trend may indicate that data complexity affects model performance, and reducing the number of classes can lead to a faster and more stable learning process.

The data distribution shows a tendency for more data to be allocated for testing and validation as the experiment progresses. This suggests a more rigorous approach to evaluating the model's performance, aiming to assess its behavior on unseen data (test and validation) in a more detailed manner. The percentages of data for each purpose (training, testing, validation) are adjusted based on the dataset configuration, offering greater flexibility in analyzing how the model responds to various combinations of data.

5.1 Confusion Matrix Analysis

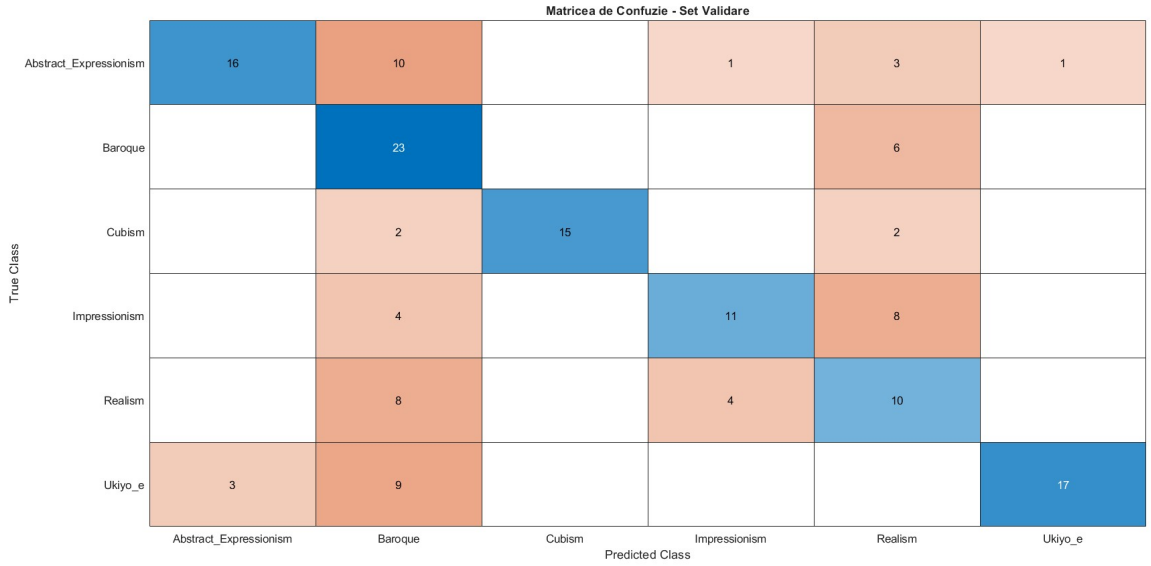


Figure 5.1: Confusion matrix with all 6 classes.

In the first confusion matrix, the model performs well for the **Baroque** class, with **23** correct predictions, and for the **Ukiyo-e** class, with **17** correct predictions, but it struggles to differentiate other classes. The most significant confusion is between **Baroque and Realism**, where **8** instances of the **Realism** class were incorrectly classified as **Baroque**. Another notable confusion is between **Abstract Expressionism and Baroque**, with **10** instances of the former incorrectly predicted as the latter. Additionally, **Impressionism** is often confused with **Realism**, with **8** instances misclassified. These issues may stem from visual similarities between styles, a lack of diversity in the training data, or limitations in the model's architecture.

Following the analysis of the confusion matrix, it was observed that the Baroque class significantly contributes to increased classification confusion. The majority of images from the other three classes were incorrectly classified as belonging to the Baroque class. At the same time, the images that genuinely belong to the Baroque class were correctly classified, which accounts for the high number of correct predictions for this category. However, this situation negatively impacted the Realism and Impressionism classes, whose images were frequently misclassified.

To assess how the absence of this class affects the model's performance, a new training phase was conducted without including the Baroque class. This experiment aimed to analyze whether the classification results improve in terms of accuracy and the distribution of predictions among the remaining classes.

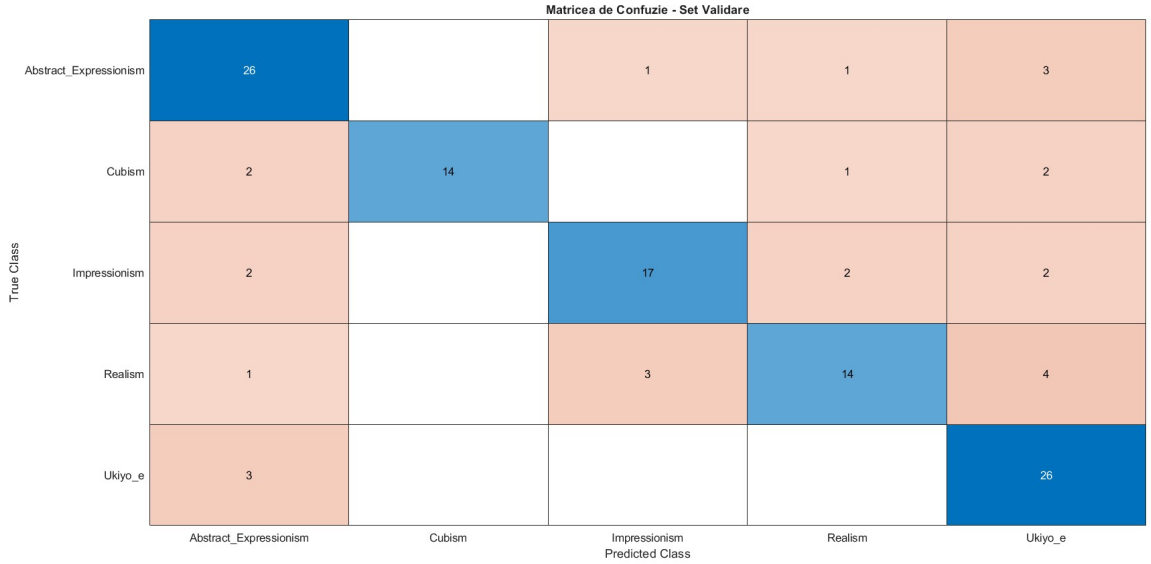


Figure 5.2: Confusion matrix without Baroque class.

The second confusion matrix (Figure 5.2) provides valuable insights into the performance of the classification model on the validation set. The model demonstrates high accuracy for classes such as **Abstract Expressionism** and **Ukiyo-e**, with **26** correctly classified images for each, indicating that these styles have distinctive features easily recognized by the model. However, the model struggles with classes like **Cubism**, **Impressionism**, and **Realism**, where significant misclassifications are observed. For example, **Cubism** has only **14** correct predictions, with several images misclassified as **Abstract Expressionism** or **Ukiyo-e**. Similarly, **Realism** often overlaps with **Impressionism** and **Ukiyo-e**, suggesting difficulties in distinguishing subtle visual features. The confusion matrix shows that Realism frequently overlaps with Ukiyo-e and Impressionism, impacting model performance and underscoring the need for improved class distinction.

These results highlight the need for further improvements. Therefore, I decided to conduct training without the Realism class to assess its influence on the overall results.

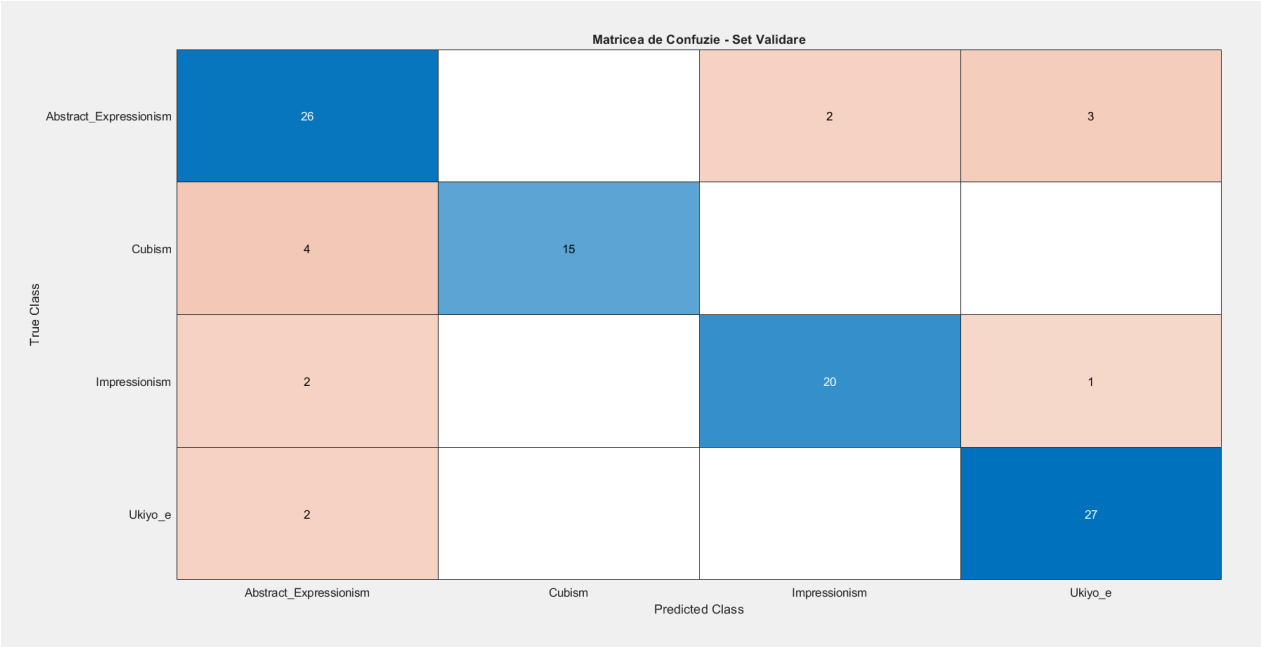


Figure 5.3: Confusion matrix without Baroque and Realism class.

The confusion matrix highlights the classification performance of the model across four artistic styles: **Abstract Expressionism**, **Cubism**, **Impressionism**, and **Ukiyo-e**. The model shows high accuracy for Abstract Expressionism and **Ukiyo-e**, with **26** and **27** correct classifications, respectively, demonstrating that these styles have distinct features that the model recognizes effectively. However, the model shows slightly lower performance for **Cubism** and **Impressionism** compared to other classes. For instance, **Cubism** achieves **15** correct predictions, with some images being classified as **Abstract Expressionism**. Similarly, **Impressionism** has **20** correct classifications, with a minor overlap observed with **Ukiyo-e**. These results indicate that while the model captures distinctive patterns for some classes, it struggles with subtle feature differences in others.

Although the latest training provided the best results, we cannot ignore the fact that achieving this required the removal of certain classes. Subsequently, a method was approached to achieve good classification performance with all six initial classes we started with.

5.2 Analysis of the Two-Level Training using VGG-16

Nr. crt.	**Nr. Classes**	**[NEP, MBS]**	**Train (%)**	**Test (%)**	**Validation (%)**	**Observations**
1	4	[10, 32]	85.15	96.97	82.70	Dataset with Impressionism, Realism and Baroque being a single class

Table 5.2: Experimental results with Impressionism, Realism and Baroque being a single class

Table 5.2 presents the results obtained after training the network using a dataset consisting of four classes, one of which included images from the three classes between which classification confusion occurred. As expected, the results were excellent, as the model no longer needed to distinguish between the three classes. The next step involved further training the network to enable it to correctly differentiate between the three classes.

Nr. crt.	**Nr. Classes**	**[NEP, MBS]**	**Train (%)**	**Test (%)**	**Validation (%)**	**Observations**
1	3	[10, 32]	72.50	95.48	68.13	Dataset with only Impressionism, Realism and Baroque classes

Table 5.3: Experimental results with dataset formed of Impressionism, Realism and Baroque

Table 5.3 illustrates the results obtained after further training the previous network on a dataset consisting solely of the three classes between which confusion was occurring. The results are quite good for the validation dataset, but the performance is significantly weaker for the testing dataset.

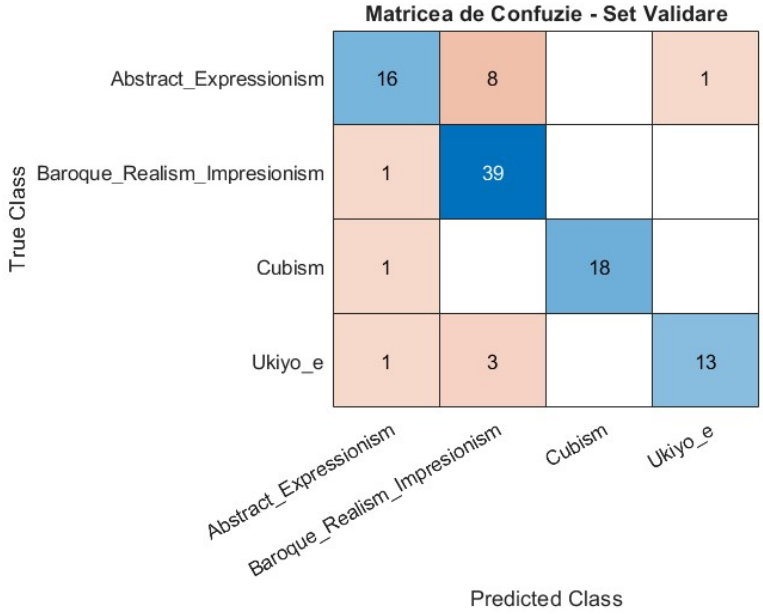


Figure 5.4: Confusion matrix for the first train stage of the CNN.

Figure 5.4 illustrates the confusion matrix for the initial training of the network, where the three classes (Impressionism, Realism, and Baroque) were combined into a single class. It is evident that the network has learned the distinctions between the four classes more effectively, making the differentiation process much easier.

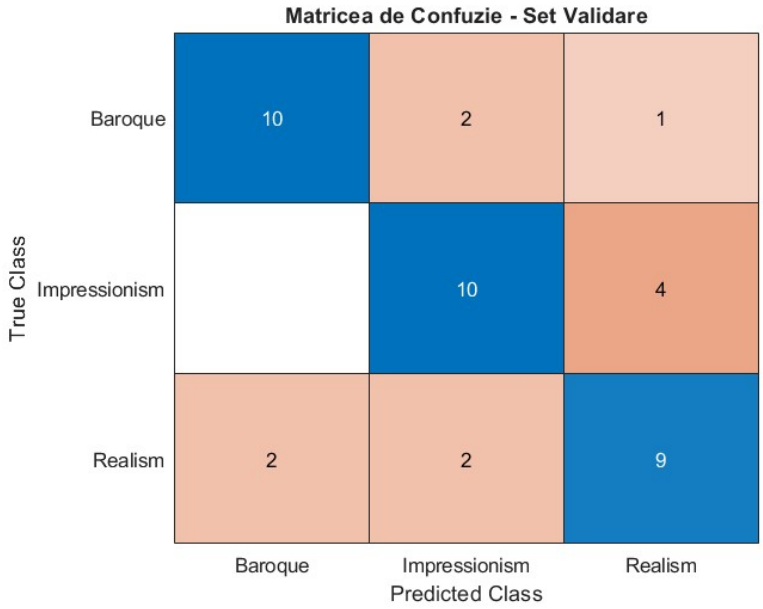


Figure 5.5: Confusion matrix for the second train stage of the CNN.

Figure 5.4 illustrates the confusion matrix for the initial training of the network, where the three classes (Impressionism, Realism, and Baroque) were combined into a single class. It is evident that the network has learned the distinctions between the four classes more effectively, making the differentiation process much easier.

To gain a better understanding of these results, a detailed analysis will be performed on images from the six classes used, providing a clearer view of which images create confusion for the convolutional neural network model.

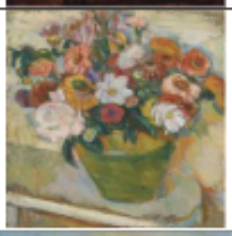
Painting	Old Network Prediction	New Network Prediction	Correct prediction
	Abstract Expressionism	Abstract Expressionism	Abstract Expressionism
	Impressionism	Baroque	Baroque
	Impressionism	Impressionism	Impressionism
	Impressionism	Realism	Realism
	Cubism	Cubism	Cubism
	Ukiyo e	Ukiyo e	Ukiyo e

Figure 5.6: Confusion matrix for the second train stage of the CNN.

Figure 5.4 presents the results obtained after testing the first network and the second network on a new set of images. As can be observed, the second network classified the images more accurately compared to the first trained network. This result is satisfactory, although, for certain images, the network still struggles to correctly identify the class.

6 Conclusions

The use of the VGG-16 algorithm has proven to be extremely effective in addressing the challenge of image classification based on artistic styles. The WikiArt dataset, which comprises high-resolution images, provided an ideal foundation for this classification task, as its quality minimized potential issues related to dataset limitations.

The results obtained are promising and provide a strong basis for future training or data preprocessing endeavors. While the initial stages of the project faced challenges, particularly in distinguishing between certain classes, subsequent refinements in the training process yielded significantly improved outcomes. Specifically, the VGG-16 algorithm accurately identified three out of the six classes without notable confusion. For the remaining three classes, additional training was implemented to enhance the network's ability to distinguish between all six artistic styles.

After the supplementary training phase, the network successfully classified all six new images provided for testing. This achievement not only highlights the exceptional performance of the network and the quality of the dataset but also underscores the potential of applying machine learning techniques to the domain of art. Such results demonstrate that art can seamlessly integrate with automated learning systems to facilitate innovative solutions in classification and beyond.

The chosen classification problem posed a few challenges, yet the promising results indicate room for further exploration. Enhanced pre-processing techniques could lead to even greater performance. For instance, using a larger dataset or applying advanced image filters, such as contrast adjustment or noise reduction, may significantly improve the model's accuracy and generalization.

Bibliography

[1] WikiArt Dataset. https://www.wikiart.org

[2] K. Simonyan, A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv preprint arXiv:1409.1556, 2014.

[3] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. *Gradient-Based Learning Applied to Document Recognition*. Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] J. Deng, W. Dong, R. Socher, L. Li, K. Li, L. Fei-Fei. *ImageNet: A Large-Scale Hierarchical Image Database*. CVPR 2009.

[5] Gatys, L. A., Ecker, A. S., Bethge, M. *Neural Algorithm of Artistic Style*. arXiv:1508.06576, 2015.

[6] Krizhevsky, A., Sutskever, I., Hinton, G. E.. *ImageNet Classification with Deep Convolutional Neural Networks*. Advances in Neural Information Processing Systems, 2012.

[7] Bianco, S., Cadene, R., Celona, L., Napoletano, P. *Benchmark Analysis of Representative Deep Neural Network Architectures*. IEEE Access, 2018.. https://doi.org/10.1109/ACCESS.2018.2877890

[8] Saleh, B., Elgammal, A. *Large-scale Classification of Fine-Art Paintings: Learning The Right Metric on The Right Feature*. International Journal for Digital Art History, 2015. https://arxiv.org/abs/1505.00855