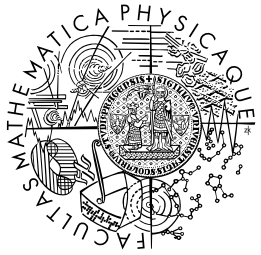


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Yuliia Alpaieva

A Comparative Study of Selected Machine Learning Methods for Art Images Classification

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: doc. RNDr. Elena Šikudová, Ph.D.

Study programme: Artificial Intelligence

Prague 2025

I declare that I carried out this master thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank my supervisor, doc. RNDr. Elena Šikudová, Ph.D., for her guidance, support, feedback, and valuable advice during the preparation of this thesis. I would also like to thank my family and friends for their support.

Title: A Comparative Study of Selected Machine Learning Methods for Art Images Classification

Author: Yuliia Alpaieva

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: doc. RNDr. Elena Šikudová, Ph.D., Department of Software and Computer Science Education

Abstract: In this thesis we compare machine learning techniques for the art classification task. We compare several pre-trained deep learning approaches, including the Residual Neural Network, EfficientNet, Densely Connected Convolutional Network, MobileNet, and Vision Transformer. We implement an approach that extracts features using deep learning methods and combines them with the features extracted by classical computer vision algorithms, then classifies them using machine learning classifiers. We compare the performance of selected classifiers using features extracted from various layers of chosen deep learning methods, combined with hand-engineered features. We apply the best-performing classifiers in combination with the best-performing feature extraction strategy for the genre classification task and the style classification task using cropped facial regions. In addition, we compare the described tasks from a classification perspective and evaluate the performance of applied classifiers.

Keywords: Art Classification, Machine learning, Feature Extraction

Contents

1 Introduction **6**
- 1.1 Automatic Art Classification . . . 6
- 1.2 Existing Methods . . . 7
  - 1.2.1 Feature extraction approaches . . . 7
  - 1.2.2 Deep Learning approaches . . . 8
  - 1.2.3 Transformers . . . 9
- 1.3 Thesis Goal . . . 10

2 Theoretical background **11**
- 2.1 Machine learning techniques . . . 11
  - 2.1.1 K-nearest neighbors . . . 11
  - 2.1.2 Random Forest . . . 11
  - 2.1.3 Gradient Boosting . . . 12
  - 2.1.4 Multilayer perceptron . . . 13
  - 2.1.5 Support vector machine . . . 13
- 2.2 Deep Learning . . . 14
  - 2.2.1 Residual Neural Network . . . 15
  - 2.2.2 MobileNet . . . 16
  - 2.2.3 EfficientNet . . . 17
  - 2.2.4 DenseNet . . . 17
  - 2.2.5 Vision Transformer . . . 18
- 2.3 Low-level features extractors . . . 19
  - 2.3.1 Local binary pattern . . . 19
  - 2.3.2 K-means clustering . . . 20

3 Dataset **21**

4 Experiments **26**
- 4.1 Pretrained Models . . . 26
- 4.2 Feature Extraction . . . 28
  - 4.2.1 Style Classification . . . 31
  - 4.2.2 Genre Classification . . . 41
  - 4.2.3 Style Classification using Cropped Facial Regions . . . 50

Conclusion **57**

Bibliography **59**

List of Figures **61**

List of Tables **63**

1 Introduction

Art classification involves recognizing art style and genre or identifying the author of paintings. Experts primarily handle this task because it requires a deep understanding of historical context, visual art knowledge, and the ability to analyze the painting. To identify art style, experts rely on the following:

- Composition: how elements are arranged and how space is used.
- Color palette: artists prefer some specific color palettes, making movement recognizable.
- Brushwork: the way the art piece was painted.
- Lines and Shapes: the lines that the artist used can help to identify the style. For instance, strong lines are characteristic of Cubism, and softer lines are characteristic of Rococo.
- Historical and Cultural context: experts analyze location, period, and relevant social trends.
- Scientific methods include infrared reflectography, X-ray scanning, and pigment analysis.

1.1 Automatic Art Classification

Art analysis can take months, especially if the collection contains many pieces. Moreover, sometimes evaluation can be subjective and vary among experts. Automatic detection of art styles can be beneficial for several purposes, such as:

- Digitization of art repositories in museums, galleries, or private collectors.
- Art understanding and analysis. Classification can provide useful information about art, such as revealing similarities in art styles, understanding art evolution, and detecting influences among artists and styles.

The problem of art classification has unique challenges compared to standard image classification tasks. Unlike real-object images, which contain well-defined objects and backgrounds, artwork often represents abstract forms, expressive brushstrokes, and stylistic features that can be difficult to quantify. Additionally, art styles can blend, unlike real-object classification, where each class has well-recognized boundaries (for instance, animal classification). Some artists used combined techniques that incorporate several elements from various art movements. Furthermore, another challenge can be the imbalance of the data set, where some art styles are overrepresented while others have fewer examples. This can be caused by several factors, like the popularity of an art style, the lack of digital documentation in art archives, and the variety of artworks that exist only in private collections and have never been digitized. This limits the amount of publicly available training data.

1.2 Existing Methods

Art classification is challenging due to the variability in artistic styles, techniques, and visual elements. A combination of techniques is often needed for sufficient performance. Traditional methods rely on hand-crafted features such as color histograms and texture descriptors, which can provide low-level information. Deep learning approaches excel in capturing complex patterns and relationships in artworks.

1.2.1 Feature extraction approaches

This technique is based on extracting key attributes from images using various classical computer vision and machine learning techniques and passing them into a classification algorithm. The idea is to transform raw images into a meaningful representation for machine learning models. Traditional methods extract features using techniques such as edge detection, color histograms, and texture descriptors. The extracted features play an important role in distinguishing artistic styles, color palettes, and brushstroke patterns.

In [Aga+15] authors use five features, namely SIFT, GIST, HoG combined with LBP, GLCM, and color features. For color feature extraction, they partitioned each image into 16 patches and extracted color features in the CIELAB space by dividing each dimension (L*, a*, b*) into four bins, resulting in a 1×1024 dimensional vector per image. SIFT was applied to extract local features. Then a visual dictionary of 400 words was created by the bag-of-words approach. After this, they constructed a spatial pyramid, dividing the image into 21 subregions, each represented as a histogram. As a result, they achieved an 8400-dimensional feature vector. To extract global scene characteristics, the authors applied GIST, which captures the spatial structure of images by decomposing the painting with 32 filters. After this downsampled output, the magnitudes of the filters were combined into a 512-dimensional vector for each painting. Additionally, the authors combined HoG and LBP features; HoG captures object shapes and edge directions, while LBP enhances texture representation by filtering noisy edges. Next, they applied a bag-of-words approach to combined vector, forming a visual dictionary. Finally, they computed a histogram based on this dictionary for each vector, resulting in a final feature vector of size 4000. To extract texture-related features such as brushstroke patterns, the authors used GLCM, which analyzes the spatial relationship between pixel intensities in four different directions. In this research, authors classify 10 styles and 6 genres. They compare various classifiers such as Random Forest, MLP, and libsvm with different kernels, and libsvm with χ^2 kernel gave the best accuracy.

In [GCM18], the authors used unsupervised methods to extract features. The study focuses on classifying and clustering 8 art styles: Art Nouveau, Baroque, Expressionism, Impressionism, Realism, Romanticism, Renaissance, and Post-Impressionism. Their approach consists of three phases: the first step is to downsampling all images of the paintings to 64×49, the second is feature extraction using Principal Component Analysis (PCA) (100, 300, 500, and 1000 features) or Unsupervised Feature Learning with K-means (UFLK) (500, 1000, 2000, and 3000 features), the third phase is to apply the SVM algorithm for classification or spectral clustering algorithm for clustering on extracted features or raw pixels (without feature extraction). UFLK features provided the best overall style prediction performance for classifying the eight different painting styles. The study compares UFLK with AlexNet and LeNet, and UFLK achieves comparable accuracy.

In [Nun+22], authors combine features for classifying Cubism, Fauvism, Impressionism,

Naive Art, Pointivism, and Realism. They convert images to RGB, CIELab, CIELuv, and HSV. Later, feature extraction is applied using Local Binary patterns to capture texture and Sum and Difference Histograms to capture the relationship between pixels. Since the extracted feature set is high-dimensional, PCA is applied to select the most discriminant features. The final feature vector is reduced from 428 to 168 dimensions. Then, the obtained principal features are submitted to the multilayer perceptron (MLP) algorithm, which contains 168 input neurons, 88 neurons in the hidden layer, and 7 neurons at the output.

In [Nun+18], the authors classify 6 styles such as Abstract Impressionism, Cubism, Pop Art, Renaissance, Romantism, and Muralism. The proposed method considers the extraction of texture features from different perceptual color spaces (CIELab, CIELuv, and HSV). They calculate the mean, standard deviation, and color palette from the different color spaces, combine these values with Local Binary Pattern (histograms of 59, 18, 10 bins) and Sum and Difference Histograms, and apply PCA. They are using an Artificial Neural Network for classification.

As we can see in the analyzed methods, the authors extracted various features to capture the representation of the art style. Brushstroke patterns, color palettes, and scene characteristics are crucial for art classification. In almost every method, authors extract these features.

1.2.2 Deep Learning approaches

Deep learning approaches have significantly improved art classification. Convolutional Neural Networks (CNN) shows high performance in capturing intricate stylistic patterns, brushstroke details, and compositional elements, enabling highly accurate classification of paintings by genre, style, or artist.

In [VS17] the authors compared several CNNs for artist identification; they used 17,000 paintings from 57 different artists, each having 300 paintings. The images are pre-processed by zero centering and normalization, cropping to size 224×224, and data augmentation (random horizontal flips with probability 50% and then taking a crop of a random section of the painting). The study evaluates three different CNN architectures for artist classification: a lightweight convolutional network, ResNet-18 trained from scratch, and pretrained ResNet18. The pretrained network showed significantly better accuracy than other considered methods. CNNs successfully learn artist-specific styles, but challenges remain for artists who have various styles and is influenced by others.

This study [SRM24] introduces an approach using CNN to classify Persian miniatures from five schools: Herat, Tabriz-e Avval, Shiraz-e Avval, Tabriz-e Dovvom, and Qajar. Data preprocessing involves dividing each image into five distinct patches. Each patch is then normalized separately to ensure consistency in input data. The final classification result is derived by combining the classification results of these five sections (fused classification approach). Authors applied Pre-trained CNN models (DenseNet201, DenseNet121, DenseNet169, VGG16, VGG19, InceptionV3, Xception). They compared the performance of the network's individual patches and fused images (combined patch predictions). This research demonstrates the effectiveness of employing a fusing patching strategy for classifying Persian miniatures into their respective schools of art, achieving an average accuracy rate of over 91%.

The authors of [Imr+23] also used patching as a pre-processing technique. The proposed approach is divided into two phases. In the first phase, the input image is

divided into five patches. Then, these patches are classified by the deep neural network. Each patch produces a probability vector that indicates the likelihood of different artistic styles. In the second phase, the probability vectors from the first phase are combined into a single vector, and a shallow network is trained on these vectors. This network makes the final style decision based on the combined information. Instead of working directly with the image data, this phase operates on the probabilities assigned to each artistic style by the first-phase classifier. For the analysis, the authors used six different CNN models with varying architectural complexities (GoogleNet, ResNet50, Inceptionv3, VGG16, AlexNet, and VGG19). In addition, three distinct datasets (WikiArt, Australian Aboriginal art, Pandora 18K) comprising images of fine art paintings were used for the evaluation. The proposed approach was compared to other classification approaches: single CNN to classify whole images, dividing images into five patches and classifying them independently, classifying patches using the voting mechanism, employing average probabilities to determine final labels, averaging probabilities with weights suggested by an optimization algorithm, and the same as a proposed method but as additional input to classifier the entire resized image was passed. The proposed approach significantly improves style classification. The patch-based approach preserves artistic details, avoiding issues related to resizing entire images.

As we can see, Convolution Neural Networks perform well in art-classification tasks. Their ability to capture style patterns, texture, and overall composition allows them to classify art styles. The patching strategy also performs well in this task; it keeps the original texture representation of the image, avoiding resizing. Moreover, CNNs can be applied not only to style classification, but also to the classification school to which the painting belongs.

1.2.3 Transformers

Transformer architecture, which was initially developed for language tasks, has been successfully applied to visual tasks, such as art classification.

In [SPP23], the authors compare CNN-based methods and the ViT transformer for identifying Vincent van Gogh‘s works. As a data source, they utilized 654 images confirmed as Vincent van Gogh paintings and the dataset of negative examples, which contains imitations and proxy artworks (paintings by artists with similar styles). They compared EfficientNetB5, ResNet101, and transformers Swin-Tiny and Swin-Base. For each architecture, they performed 20 experiments and compared the average prediction accuracies for individual patches created from images and for the entire paintings. EfficientNetB5 can better distinguish between van Gogh and his contemporaries than both Swin Transformers do. The Swin-Tiny Transformer is better compared with EfficientNetB5 on the imitation set.

CNNs can be used not only as classifiers, but also as feature extractors. In [Wan+24], the authors proposed the hybrid architecture where CNN and Transformer are combined into one model; the CNN extracts local features, and these features are passed to Transformer, which applies multiple layers of self-attention and finds global dependencies between these local features. For the CNN component, they used Resnet50 and Vision Transformer for the transformer component. They employ datasets of Chinese paintings and oil paintings by Western authors such as Van Gogh, Da Vinci, and Picasso. The proposed fusion model was compared with the ResNet50 and Transformer models. Experimental results in both Chinese datasets show that the fusion model significantly

outperforms the traditional CNN and Transformer models.

Although the Transformer was originally created for language tasks, it has been successfully applied to an art classification task. The transformer showed good performance for authorship classification even when the dataset contained paintings by artists with similar styles. A CNN as feature extraction combined with a Vision Transformer performed better than each model separately.

1.3 Thesis Goal

This thesis aims to classify art images. We will compare several approaches applied to this task and evaluate their performance. Our goals are:

G1: Explore the differences in styles and genres that can influence automatic classification.

G2: Compare existing deep learning methods for classifying paintings according to their styles. Examine how architectural differences in deep learning methods influence classification performance.

G3: Implement an approach where deep learning models are used as feature extractors in combination with computer vision techniques, and the resulting vector is classified using machine learning classifiers. Evaluate the performance of selected classifiers. Examine whether there is a difference in performance between classifiers that use features extracted from different layers of the same model. Examine which layers of Convolutional Neural Networks (CNNs) provide the most representative features for the task considered.

G4: Apply the best-performing feature strategies and machine learning classifiers (from experiment G3) to the genre classification task and evaluate their performance. Analyze the differences between genre and style classification tasks and examine how they influence the performance.

G5: Apply the best-performing feature strategies and machine learning classifiers (from experiment G3) for the style classification task using Cropped Facial Regions. Examine how the usage of cropped paintings influences the performance.

2 Theoretical background

In this chapter, we will discuss the methods used in this study. We provide a description of machine learning methods used for classification, deep learning methods used for classification and feature extraction, and some classical computer vision algorithms used as feature extractors. Firstly, we will provide a brief theoretical background on the main concepts. We will also define the motivation behind our chosen methods and their advantages and disadvantages for art classification tasks.

2.1 Machine learning techniques

This section presents various machine learning methods used in this work. The objective was to select algorithms representing different implementation strategies and underlying mechanisms and apply them to our task. This section briefly describes the theory behind the techniques and highlights their strengths, which benefit our task and potential limitations.

2.1.1 K-nearest neighbors

The K-nearest neighbors algorithm (KNN) is a non-parametric supervised learning method. It can be utilized for classification, where the input is a feature vector and the output indicates class membership. It is based on the principle that close data points will likely belong to the same class. The idea is that class label is assigned based on major class among k closest neighbors. To determine the nearest neighbors, a distance metric is used. For example, it can be Euclidean (2.1) or Manhattan distance (2.2)

Euclidean Distance

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

Manhattan distance

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.2}$$

The algorithm's training phase consists only of storing the vectors and labels of the training sample, making the training process fast. The best choice of k depends on the data, and the search for k is usually performed using parameter search techniques. The main advantages of KNN are that it is intuitive, easy to implement, and can perform well if the decision boundary is irregular. Moreover, it performs well on small and noisy data sets. However, it can suffer from the curse of the dimensionality problem because it relies on distance metrics, which become less representative in higher dimensions. During the parameter search, we will consider several distance metrics, including the cosine distance. This metric measures the angle between the vectors, making the relationship between the features in the sample more important than the dimensionality of the feature vector.

2.1.2 Random Forest

Ensemble learning is a technique that combines multiple models to form a stronger predictive model. The central idea is that combining weak models can yield a more

accurate and robust model. Bagging is an ensemble learning technique that trains multiple independent models on different subsets of data. The final prediction is a combination of the models' predictions; this can be implemented by voting for classification or averaging for regression. Random Forest (RF) is a bagging technique widely used for various classification and regression tasks. It combines the predictions of several decision trees. A decision tree is a structure in which each node represents a predicate, each branch represents the outcome of a predicate, and each leaf represents a class label. Decision trees can develop complex structures to classify irregular patterns, which often leads to overfitting. Random forest trains multiple trees on different subsets of data. A subset of data is formed by bootstrap sampling, which means that each model is trained using randomly selected data points drawn with replacement. Unlike a traditional decision tree, which considers all features at each node to find the best split, a Random Forest selects a random subset of features, so even if trees are trained on the same data, they use different nodes as building blocks. The Random Forest classification result is the majority vote of all trees. The internal implementation of Random Forest makes it a suitable choice for image classification problems. One advantage is that the algorithm handles high-dimensional data relatively well. Random selection of features reduces the chance of overfitting to irrelevant features. Moreover, during the building process, models do not consider all features simultaneously, which reduces the influence of the curse of the dimensionality problem. Random Forest builds many trees and then averages the predictions; this makes the classifier more stable and decreases the risk of overfitting.

2.1.3 Gradient Boosting

Boosting is an ensemble technique in which the models are trained sequentially, the next model focusing on correcting the errors of the previous ones. The idea is to assign higher weights to misclassified instances and, as a result, force the next model to pay more attention to misclassified points. As a Random Forest, Gradient Boosting is an ensemble method that combines the predictions of several weak classifiers. It works by sequentially training models where each new model tries to correct the errors made by its predecessor. At the beginning, the first tree is trained on the original data. The predictions made by this tree are used for the computation of residuals. Residuals are the differences between actual and predicted values. The second tree is then trained on features and residuals from the previous tree. This process continues for all trees. Each tree is trained to predict residuals of the previous one.

$$\hat{y}_{\text{pred}} = y_1 + \eta \cdot r_1 + \eta \cdot r_2 + \cdots + \eta \cdot r_N \tag{2.3}$$

where:

- $\hat{y}_{\text{pred}}$ is the final predicted value.
- y_1 is the initial prediction, often the mean of the target variable in regression.
- η is the learning rate.
- r_i represents the residual predicted by the i-th tree.
- N is the total number of trees.

The learning rate controls the contribution of each tree. A lower learning rate means a smaller contribution of a single tree, which usually means that more estimators will be needed.

eXtreme Gradient Boosting (XGBoost) is an optimized implementation of Gradient Boosting known for its accuracy, efficiency, and scalability. It improves traditional boosting methods by incorporating second-order optimization, regularization, and optimization for better computational efficiency, making it a powerful algorithm. Compared with traditional boosting algorithms, XGBoost provides better updates because it uses first and second-order derivatives. It produces trees in an optimized way, reducing unnecessary splits. Also, it prevents overfitting by discounting complex trees using techniques such as L1 and L2 regularization. XGBoost often performs better than other ensemble methods because of its ability to focus on hard-to-classify data points, which can be crucial for art classification tasks where some styles can be hard to distinguish due to their similarity to other styles. However, it can be more sensitive to noisy data and outliers. Optimizations implemented for this algorithm make its training fast and performance highly effective for various problems.

2.1.4 Multilayer perceptron

The perceptron is the first neural network introduced. The model takes inputs, weights them, and sums them up, then passes the sum to the nonlinear activation function. The function's output is the output of the perceptron. Multilayer Perceptron (MLP) is a feedforward neural network that can be used for classification and regression tasks. It is inspired by the structure and function of the human network of neurons. MLP consists of nodes organized in layers, which are called neurons, with a nonlinear activation function. MLP has three or more layers: one input layer, one output layer, and one or more hidden layers. MLP is fully connected, which means that each node in the layer connects with a weight to a node in the next layer. The weights determine the strength of the connection, and they change during the learning process. The number of hidden layers and the number of neurons in each layer are hyperparameters that can be determined by parameter-searching techniques. The number of output neurons is determined by the task. In binary classification, it can be one neuron or two to represent the probability of belonging to each class. Furthermore, hidden and output layers have bias neurons, which also have their own weight. This neuron allows the network to learn bias in the decision boundary. The learning process contains two phases: during the first, input is propagated forward through the network, and output is computed; the gradient of error is calculated, and using this gradient in the second phase, the weights are updated. The process repeats. MLP is a highly flexible model that can learn complex relationships in data, making it suitable for art classification tasks. However, it requires a large amount of data for training and data regularization.

2.1.5 Support vector machine

Support Vector Machine (SVM) is a robust machine learning algorithm that is widely used for classification tasks. The algorithm's goal is to find a decision boundary that separates classes. Better separation provides a decision boundary with a larger maximum distance to points in classes. The algorithm identifies the hyperplane that separates the data while maximizing the margin. The margin is defined as the distance between the

support vectors and the hyperplane. Support vectors are the data points closest to the hyperplane from both classes, and they are often challenging to classify due to their positioning. The hyperplane:

$$w^T x + b = 0 \tag{2.4}$$

where:

- w is the weight vector (normal to the hyperplane),
- b is the bias (offset from the origin),
- x is the input feature vector.

The optimization goal is to minimize the norm of the weight vector to achieve the maximum margin:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \tag{2.5}$$

In real-world scenarios, achieving perfect separation between classes can be difficult due to noise in the data or overlapping between classes. SVM addresses this challenge by introducing slack variables $\xi_i \geq 0$, allowing misclassifications controlled by parameter C. It provides a trade-off between classification error and maximization of margin. The new optimization problem becomes:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \tag{2.6}$$

SVM is able to handle non-linearly separable data; for this, it uses the kernel trick. The idea is to transform data into a higher dimension where it will be possible to separate it by a hyperplane. Here are several examples of kernel functions:

- Linear kernel: $K(x, x') = x^T x'$
- Polynomial kernel: $K(x, x') = (x^T x' + c)^d$
- Radial Basis Function (RBF) kernel: $K(x, x') = \exp(-\gamma\|x - x'\|^2)$

The kernel trick allows us to find the decision boundary without explicitly computing the transformation. SVM has several advantages that make it suitable for art classification tasks. First, it is effective for high-dimensional data, which is beneficial for our task because the feature vectors we will classify are high-dimensional. Second, it is efficient for small amounts of data, and with a correctly chosen kernel function, it can identify decision boundaries between different art styles.

2.2 Deep Learning

This section describes the deep learning models selected for this study. Firstly, we describe Convolution Neural Networks (CNNs), their architectures, and their suitability for art classification tasks. Next, we will cover the Transformer architecture and its implementation nuances for classification tasks. Additionally, we will analyze the advantages and disadvantages of the selected models.

2.2.1 Residual Neural Network

Residual Neural Network (ResNet) was introduced by [He+15]. Figure 2.1 represents Residual network with 34 parameter layers. The motivation for creating this architecture was the problem of vanishing and exploding gradients and degradation problems in Neural Networks.

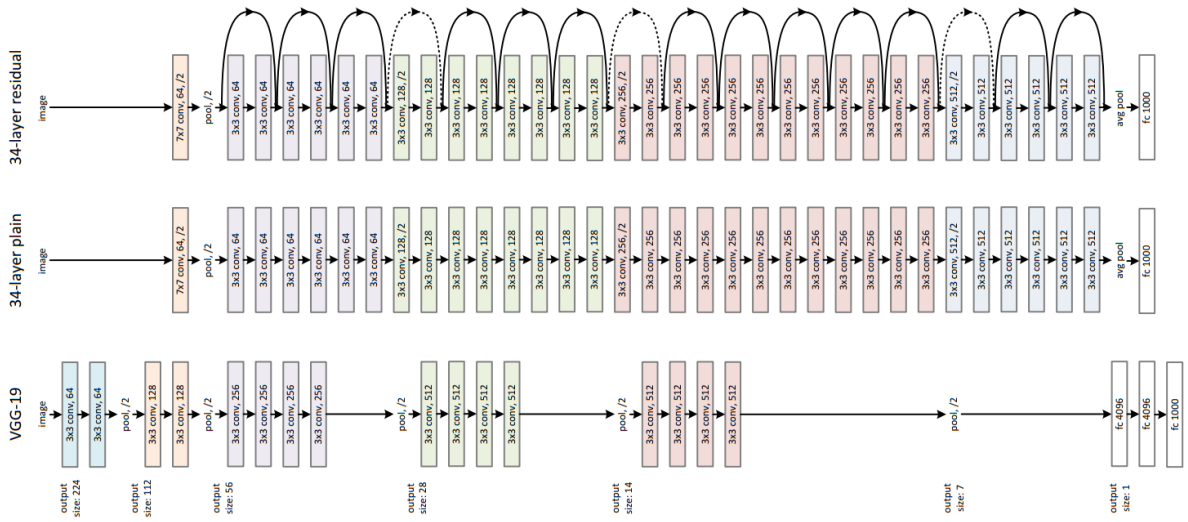


Figure 2.1 VGG-19, a plain network with 34 parameter layers, a residual network with 34 parameter layers. Source:[He+15]

The Vanishing gradients occur when the gradients of the loss function significantly decrease and become extremely small during the backpropagation phase of the learning algorithm. As a result, earlier layers get low-gradient updates, which cause slow learning or even the absence of learning. Degradation of a neural network is a phenomenon where, with increasing network depth, the performance decreases even though the model is starting to be able to extract more abstract features. To solve this problem, the Residual Block was introduced. Let x denote the input to a given subnetwork, and $H(x)$ represent the transformation function learned by this subnetwork. This transformation can be reformulated such that the network learns a residual function $F(x) = H(x) - x$. As a result, the output of the subnetwork can be represented as:

$$y = F(x) + x \tag{2.7}$$

The operation $+x$ is implemented using a *skip connection*, which connects the subnetwork's input to its output. This connection is called *residual connection*. Each subnetwork is *residual block*. A deep residual network (ResNet) is constructed by combining multiple residual blocks sequentially. ResNet has several key advantages that make a considerable difference for image classification tasks, making it one of the most impactful architectures. Firstly, introducing skip connections helps decrease the problem of vanishing gradients, allowing effective learning of deep architectures. This architecture can be scaled to a large number of layers without performance degradation, allowing the network to extract abstract and discriminative features. Secondly, identity mappings using skip connections allow features from earlier layers to be passed to deeper layers without modification. As a

result, we achieve feature reuse, as low-level features that are learned in the early layers are accessible in the later layers. In deep networks, information goes through several layers sequentially, after each layer applies a modification, causing data change; therefore, later layers get modified features, and some details can be lost. For art style recognition, low-level details such as texture, edges, and brushstroke patterns are crucial. Feature reuse allows the model to consider both local and global features at the same time, which is important for the art classification task.

2.2.2 MobileNet

MobileNet is a family of convolutional networks designed for low-latency and low-power computations. Therefore, it is suitable for running on resource-constrained devices. MobileNet V1 was introduced in 2017 ([How+17]). The authors introduce the idea of depthwise separable convolutions, which significantly reduce the computational complexity of convolution layers. One convolution is decomposed into two separate convolutions: a depth-wise convolution that filters each input channel independently and a point-wise convolution that combines the output of the depth-wise convolution.

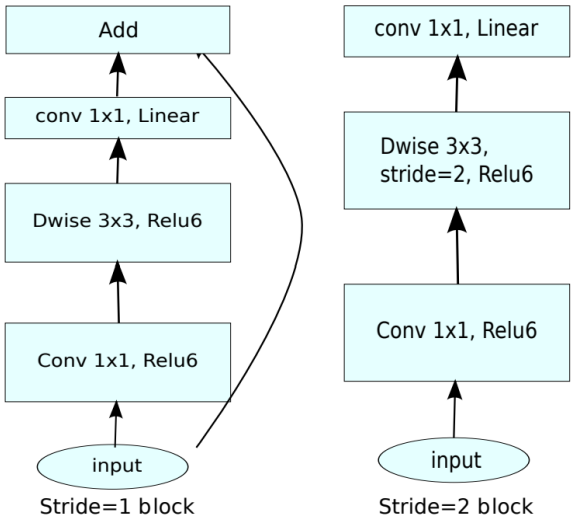


Figure 2.2 Block Mobilenet V2. Source: [How+17]

MobileNet V2 ([San+19]) introduced architectural improvements over MobileNet V1, including inverted residual blocks and linear bottlenecks. Figure 2.2 illustrates the structure of MobileNet V2 blocks. Inverted residual blocks expand the input feature map to a higher-dimensional space, process it via depthwise convolutions, and then project it back to a lower-dimensional output. This is the opposite strategy to classical residual blocks that keep the feature map dimensions roughly the same throughout the block. The main purpose of the linear bottleneck is to reduce dimensionality. The authors experimentally showed that using linear layers in the narrow layers is crucial as it prevents the loss of too much information.

MobileNet V3 ([How+19]) introduced several improvements over its predecessors, such as the use of hard-swish activation functions to improve the model's efficiency, squeeze-and-excitation (SE) modules to enhance feature representation capabilities, and

platform-aware NAS optimizations to optimize the architecture for mobile CPUs. Hard Swish (2.8) is a type of activation function based on the Swish activation function, but in order to decrease computation costs, it replaces the computationally expensive sigmoid with a piecewise linear analogue:

$$\text{hardswish}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6} \tag{2.8}$$

SE blocks enable the network to adaptively recalibrate feature responses across channels, highlighting more useful features while hiding less useful ones. This selection improves efficiency by allowing the model to prioritize informative features and reduce less informative. This is the opposite strategy to the DenseNet architecture approach, which implements feature reuse. These improvements enable MobileNet V3 to maintain a small model size while achieving competitive performance in classification tasks.

2.2.3 EfficientNet

EfficientNet is a family of convolution networks introduced by [TL19]. In contrast to other architectures where depth and width increase arbitrarily, EfficientNet introduces compound scaling. Instead of scaling one dimension of the network at once, such as the number of layers, the number of channels, or the size of the input image, it uses a compound coefficient ϕ to scale all three dimensions in parallel. This leads to significantly improved performance and efficiency across various model sizes. If there is a goal of using 2^N times more computational resources, then it is possible to increase the depth of the network by α^N, width by β^N, and image size by γ^N, where α, β, γ are coefficients determined by a grid search on the original model. The intuition behind this is based on the idea that if the image is bigger, then more layers and channels are needed to increase the receptive field and capture more details. The base EfficientNet-B0 network is based on the inverted bottleneck residual blocks, the same as MobileNet V2 has, and squeeze-and-excitation blocks. EfficientNet combines SE blocks with deeper and wider layers, allowing it to capture complex composition effectively.

2.2.4 DenseNet

Densely Connected Convolutional Network (DenseNet) was proposed by [Hua+18]. The authors of DenseNet proposed a revolutionary approach to connect layers in the network. Unlike other CNN architectures, where each layer is connected only to the next layer, DenseNet establishes direct connections between all layers within a block. This type of connection allows modeling complex relations between features and extracting high-level representations. Each dense block (represented in Figure 2.3) consists of multiple convolution layers, followed by batch normalization and a non-linear activation function. These dense blocks are connected using transition layers, which are used to reduce the number of features and down-sample the spatial dimensions of feature maps. This gives an opportunity to preserve computational efficiency. Transition layers consist of a convolution layer and average pooling. Like ResNet architecture, DenseNet implements feature reuse by allowing each layer to access all previous layers. In contrast to the Resnet architecture, where each layer has access to the previous output and residual, in DenseNet, each layer receives the combined outputs of all previous layers. This allows DenseNet to preserve feature diversity and encourages explicit feature reuse, while ResNet tends to focus on

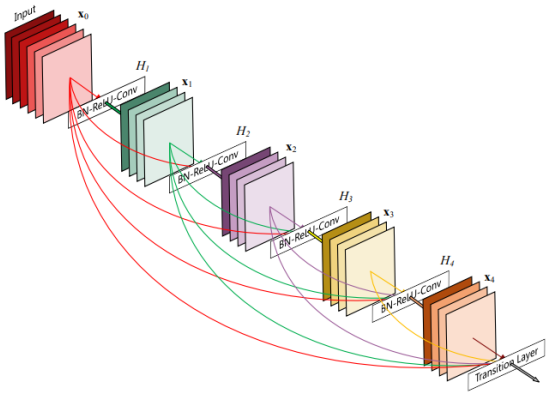


Figure 2.3 A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input. Source:[Hua+18]

feature improvement. The direct connection between layers reduces the risk of vanishing the gradients and improves convergence. Despite the increased number of connections, DenseNet is more parameter-efficient than ResNet or other traditional CNNs. However, dense connections increase memory usage and computation complexity.

2.2.5 Vision Transformer

A Vision Transformer (ViT) introduced in [Dos+21] is a type of Transformer that is designed for various computer vision tasks. It adapts the Transformer architecture, which is primarily designed for natural language processing tasks. ViTs are less data efficient than CNNs but have higher capacity. In contrast to CNN architectures, which extract features using convolution filters, the Vision Transformer operates with images as a sequence of patches, similar to words. The model is represented in Figure 2.4. Each patch is pushed through a linear operator to obtain a patch embedding vector. To prevent positional information lost, the position of the patch is transferred into a position encoding vector. After this, two vectors are concatenated and passed through several Transformer encoders. The attention mechanism in a ViT repeatedly transforms the representation vectors of image patches, incorporating more and more relations between image patches. To perform classification, a special learnable token is added to the sequence, which is called CLS, while the rest of the tokens represent individual patches. This token does not correspond to any patch; it aggregates relevant information from the entire image during the self-attention process. The output of the final encoder layer is passed to a multilayer perceptron head for classification. The MLP typically consists of one or more fully connected layers followed by a softmax activation to output class probabilities. ViT performs well in various computer vision tasks and can overperform CNNs. However, in cases where training data is limited, CNNs performs better. In contrast to CNNs, which focus more on local receptive fields, ViT analyzes the entire image using an attention mechanism, which allows it to understand the overall composition of the image. Building long-range dependencies between pixels can help capture complex structures and patterns of artwork. Moreover, each attention head can concentrate on various features of an image, such as color, shape, texture, and structure. As a result, it allows the model to extract diverse features, which is crucial for capturing the complex semantics of artwork.

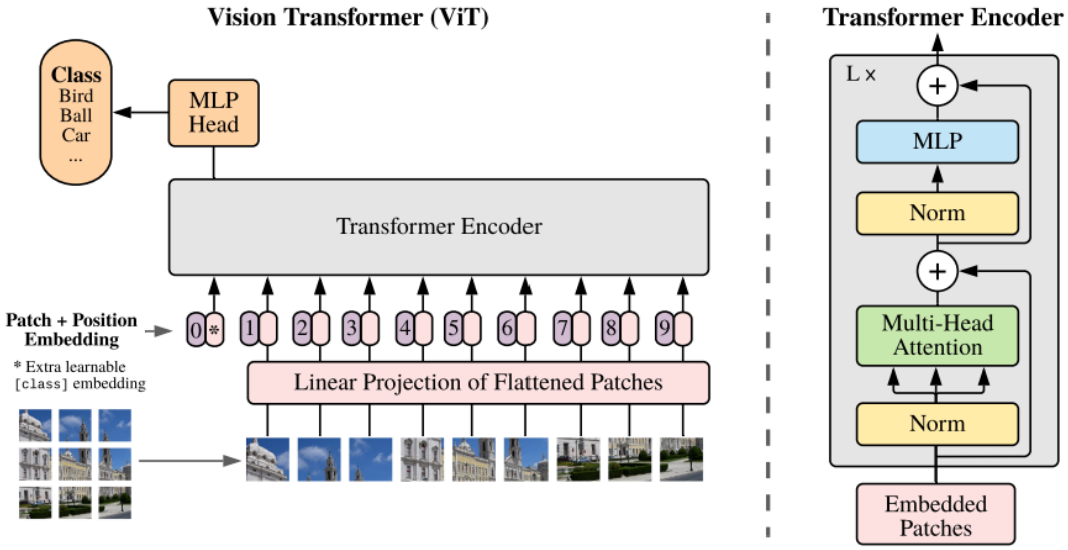


Figure 2.4 Vision Transformer architecture. Source: [Dos+21]

2.3 Low-level features extractors

In this section, we will describe algorithms that we used to extract low-level features. We will extract the color palette using the K-means algorithm and the texture using Local Binary Pattern (LBP). We will concatenate them with a feature vector extracted from deep learning methods.

2.3.1 Local binary pattern

Texture plays a key role in art classification; almost every art style has its own texture pattern, and capturing it can benefit classification. Local binary pattern (LBP), described in [OPH94], is a visual descriptor used for texture detection. Due to its computational simplicity and efficiency, it can be applied to a wide range of problems, including texture classification and biomedical image analysis. In art classification, brushstroke pattern is one of the most important features in determining style. Firstly, the examined window is divided into cells. For each pixel in the cell, compare the pixel to each of the P neighbors, set 0 if the pixel value is greater than the neighbors, and set 1 otherwise. Secondly, compute the histogram of the frequency of each number occurring. Then, histograms of all cells are concatenated to obtain a feature vector for the entire window. This histogram represents the frequency of the occurrence of texture patterns in the image. The Figure 2.5 represent the process of feature extraction.

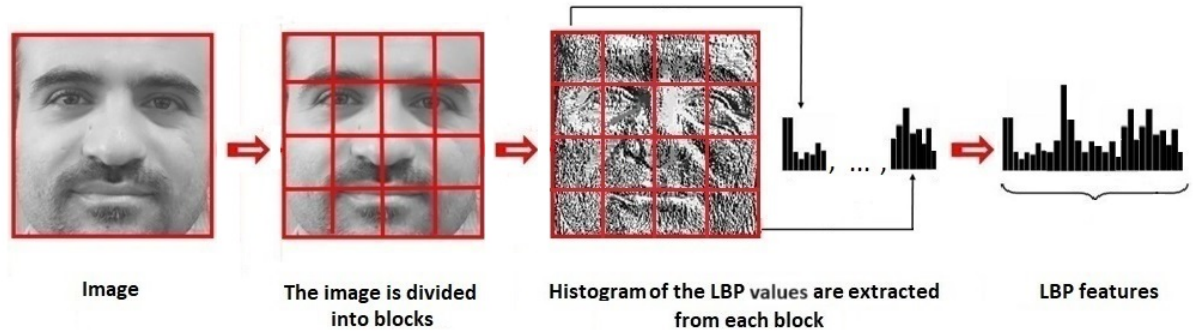


Figure 2.5 LBP feature extraction process using LBP. Source: [SHB24]

The obtained feature vector can be passed to classifiers for classifying images based on their texture properties.

2.3.2 K-means clustering

K-means clustering is an unsupervised learning technique. It can be used to identify a color palette by searching for k clusters that represent the dominant colors. Each painting has its own unique color palette, and authors who belong to one style often use similar colors. This creates a unique style color palette, making it distinct from other styles. Figure 2.6 represents an example of an image and its colors extracted by the k-means algorithm.

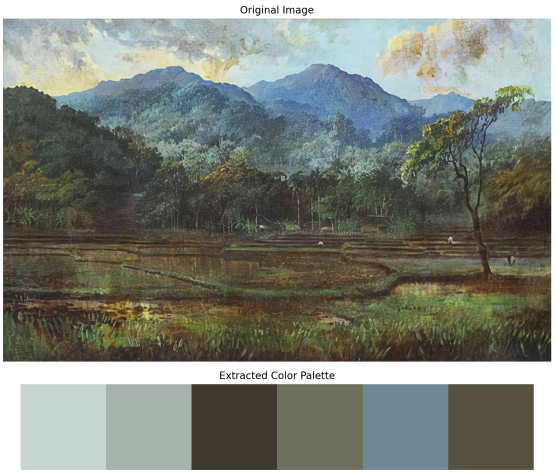


Figure 2.6 Example of painting and its color palette.

Firstly, centroids are randomly initialized in the three-dimensional RGB color space. Each pixel in the image is then assigned to the nearest cluster centroid based on a distance metric. After all pixels have been assigned, the algorithm updates each centroid by computing the mean of all pixel values belonging to that cluster. These steps are repeated iteratively until the algorithm converges. Resulting centroids represent dominant colors of the painting.

3 Dataset

In this work, we use the WikiArt dataset, which contains 81,444 pieces of visual art from various artists, taken from WikiArt.org. Images are categorized by genres, styles, and artists. Our goal is to classify art pieces according to their styles and genres.

Classification based on style

For art style classification, we selected 10 classes: Abstract Expressionism, Baroque, Cubism, Fauvism, Impressionism, Minimalism, Naive Art Primitivism, Pointillism, Rococo and Ukiyo-e [1]. Each style has unique features that distinguish it from other styles. Figure 3.1 represents randomly selected examples of each style.

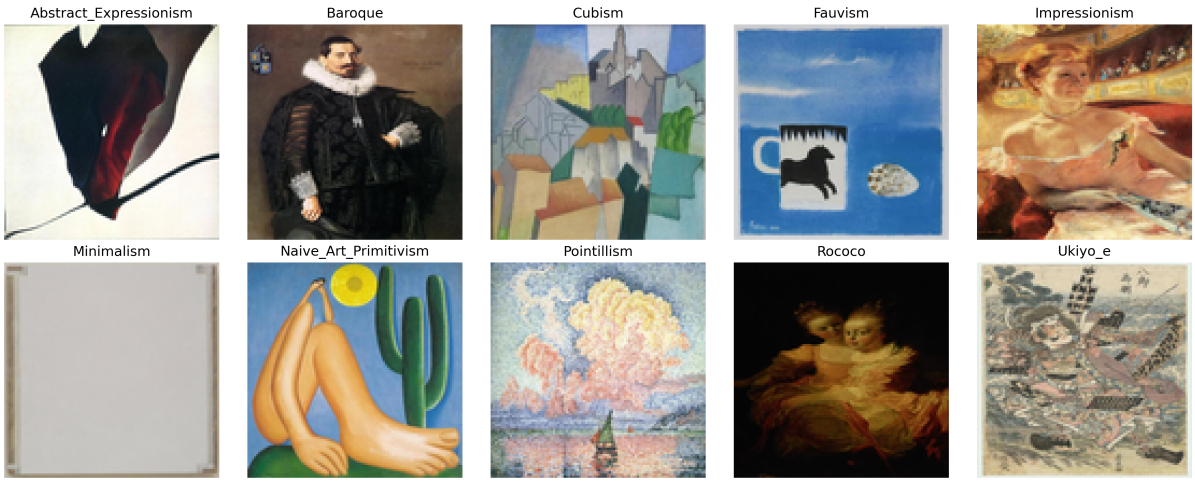


Figure 3.1 Examples of images and their styles

Abstract Expressionism works are large, abstract, and spontaneous. These paintings are often chaotic, prioritizing personal feelings over object representation. The artists used drip painting, color field painting, action painting, layering, and other techniques to create their art. In addition to oil or acrylic paints, they used tools such as sticks, knives, and their hands. Features of this style often include chaotic compositions, irregular textures and shapes, large-scale canvases, and dynamic paint application. Color palettes can be highly varied, often with intense contrast colors. The absence of shapes makes feature extraction complex.

Baroque art portrays historical and religious scenes and realistic figures. It is characterized by deep, rich naturalistic colors with a contrast between light and dark. We can observe the dominance of dark colors with light focal points in the majority of paintings. In addition, it often features strong directional lighting, shading, and perspective.

Cubism made a revolution by breaking down objects into geometric shapes and transforming them into flattened, abstract compositions that showed multiple viewpoints at the same time. In earlier periods, artists used softened, earthy colors such as brown and gray, focusing on structure and form. Later, collage elements were introduced. Cubism can be recognized for its intersecting geometric forms and angular fragmentation, making edge- and shape-based feature extraction necessary.

[1]To read more about the styles and genres described in this work, visit WikiArt.org.

Fauvism is characterized by unnatural colors and simplified forms, discarding light and depth. Art typically features flat areas of bright, contrasting colors with minimal presence of shading or details.

Impressionists capture the moment in their paintings with rapid, visible brushstrokes and a soft, pastel palette without using black color. Key features for recognizing Impressionism are diffused contours, blurred edges, the presence of natural light, and the determined texture of brushstrokes.

Minimalism focuses on simplicity, abstraction, and repetitive patterns. Artworks typically have monochromatic surfaces or basic color fields with smooth, untextured finishes. For machine learning algorithms, this means uniform colors and shapes, low complexity, and an absence of objects.

Naive Art, grouped with Primitivism, refers to artworks created by self-taught artists who lack the formal education and training that professional artists have, for instance, in areas like anatomy, history, and perspective. Therefore, paintings express childlike simplicity, frankness, and straightforward storytelling. Artists avoid realistic perspective, proportions, and shading; in contrast, they use repetitive patterns and symbolism in their work.

Pointillism is a style where the artists construct their paintings using small, distinct dots colored in contrasting colors, avoiding mixing pigments. They rely on optical mixing of the viewer's eye. This style is characterized by finely distributed color gradients and high contrast.

Rococo artists painted romantic scenes, floral motifs, cherubs, and sentimental themes. Therefore, curved lines, light and soft colors, asymmetry, and light brushwork played a fundamental role in the work's composition.

Ukiyo-e is a style of Japanese woodblocks depicting daily life, kabuki actors, and landscapes. These works are characterized by limited color palettes, a lack of perspective, and graphic shapes. It is characterized by line-based structure, symmetry, and block coloring.

As we discussed in previous chapters, data set imbalance is a common problem for the art classification task. Some classes are presented with more examples due to the popularity of art styles. To decrease the influence of this problem, first, down-sampling was performed. The most presented classes were Impressionism and Baroque, their size were 13100 and 4240 files, at that time the smallest class, Pointillism, was presented in 513 examples. We reduce the size of the biggest classes to the size of Abstract Expressionism to 2782 examples. Downsampling was performed by random selection of images. Second, we apply WeightedRandomSampler in PyTorch, which allows the sampling of training instances with replacement, assigning higher selection probabilities to samples from smaller classes. In our setup, the WeightedRandomSampler does not produce a fully balanced dataset; rather, it increases the likelihood of selecting minority class samples, resulting in a more balanced dataset for the following classification.

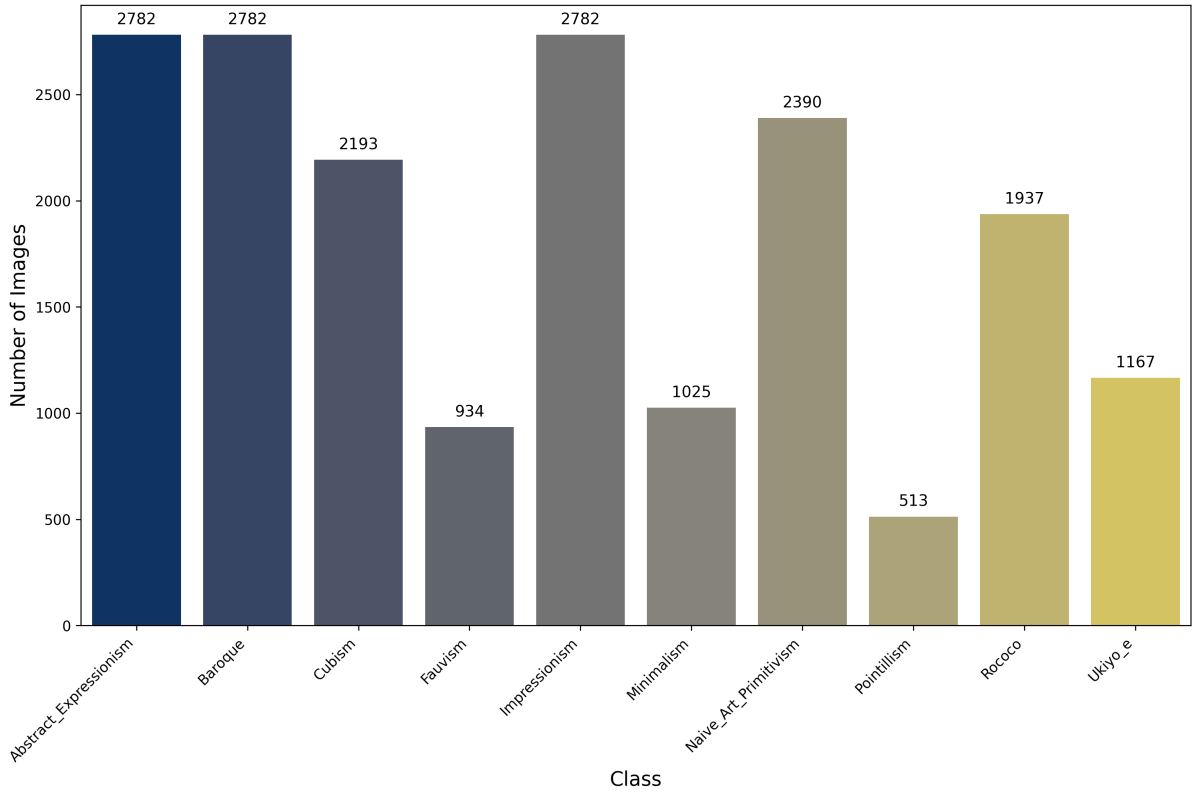


Figure 3.2 Distribution of images according to their styles after downsampling

Classification based on genres

We select 10 classes for genre classification and perform the same preprocessing as in the case of the style dataset. We downsample the biggest classes to the size of the Nude Art class (3000 examples), and discard corrupted files. Figure 3.3 represents randomly selected examples of each genre.

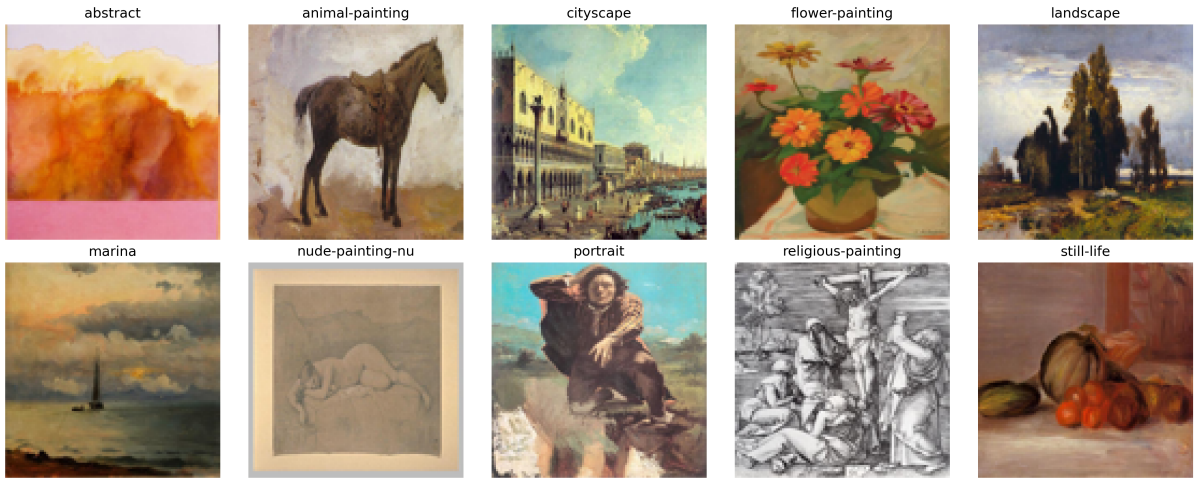


Figure 3.3 Examples of images and their genres

Abstract paintings do not represent recognizable objects or shapes. They focus on forms, color, and texture.

Animal art paintings are paintings where the main subject is domestic or wild animals.

Cityscape paintings focus on the urban environment, like streets, buildings, and other architectural elements.

Floral paintings picture flowers, floral arrangements, or patterns realistically or symbolically. Landscape paintings represent natural views, such as mountains, forests, rivers, or countryside. They often have horizons, perspective, and natural colors.

Marina paintings portray views of the sea, oceans, and harbors, often showing boats, waves, or coastal views. This artwork frequently has water surfaces, ships, and sky reflections.

Nude painting focuses on picturing the unclothed human body, often portrayed in classical or expressive poses. The challenge can be in distinguishing these paintings from portraits.

Portrait paintings focusing on depicting the person's face and upper body can also portray a group of people.

Religious artworks feature sacred themes and figures (saints or angels). They often contain devotional poses, symbolic gestures, halos, and architectural elements, such as altars or temples.

Still life paintings picture inanimate objects such as fruits, vases, books, or household items. Often objects are arranged intentionally, therefore we can observe symmetry and balanced composition in the picture. Figure 3.4 represents the distribution of genres after downsampling.

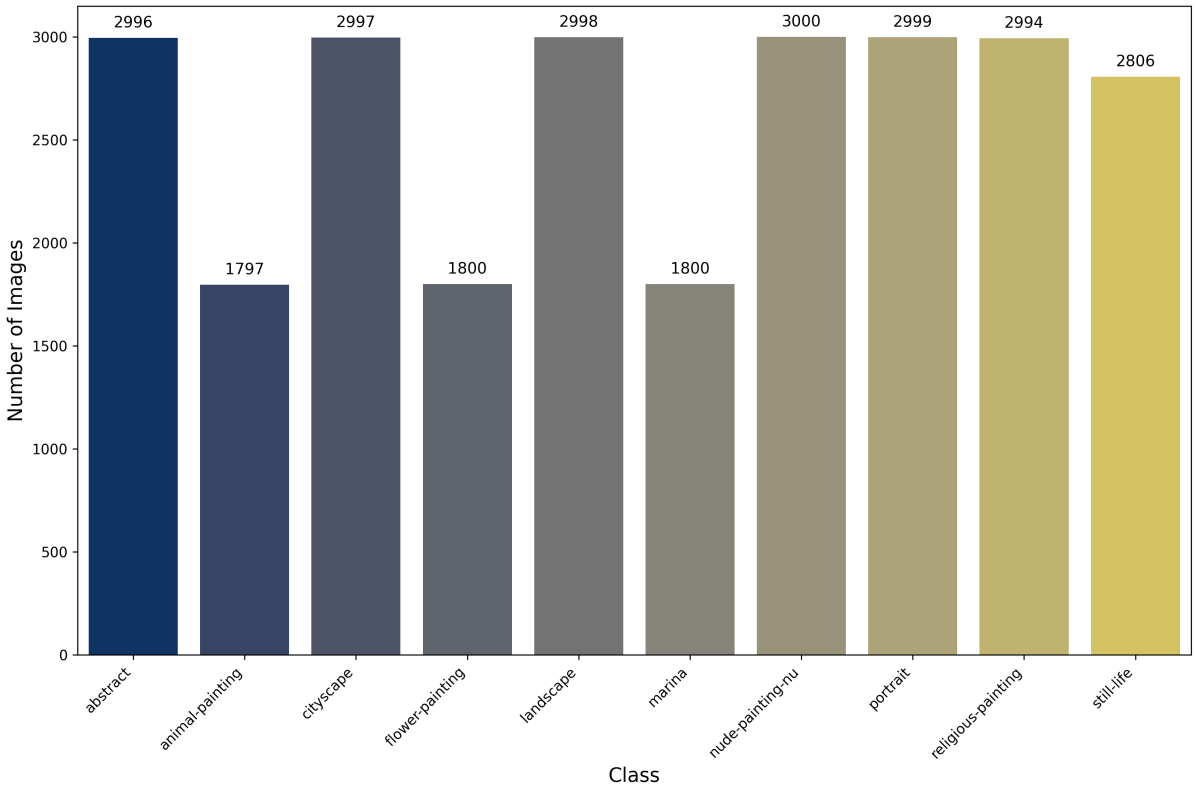


Figure 3.4 Distribution of images according to their genres after downsampling

Classification based on style using Cropped Facial Regions

For art style classification we selected 10 classes as we selected for art style classification using full paintings. While not all of these styles have picture faces for some classes, such as

Minimalism, our dataset does not have any examples. So this class was deleted. Abstract expressionism contain only 14 examples and few of them belongs to other styles, so we deleted this class as well because this number of images is not enough for classification. Figure 3.5 represent the examples of images.

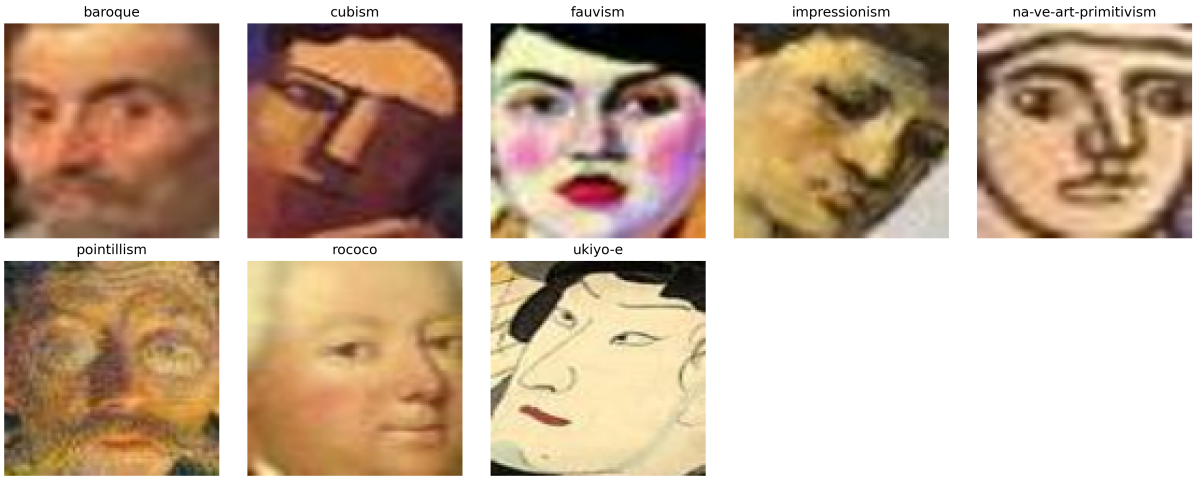


Figure 3.5 Examples of images and their styles

We did not perform downsampling, but applied WeightedRandomSampler in PyTorch and resized images to 224 × 224 size. Figure 3.6 represents the distribution of styles

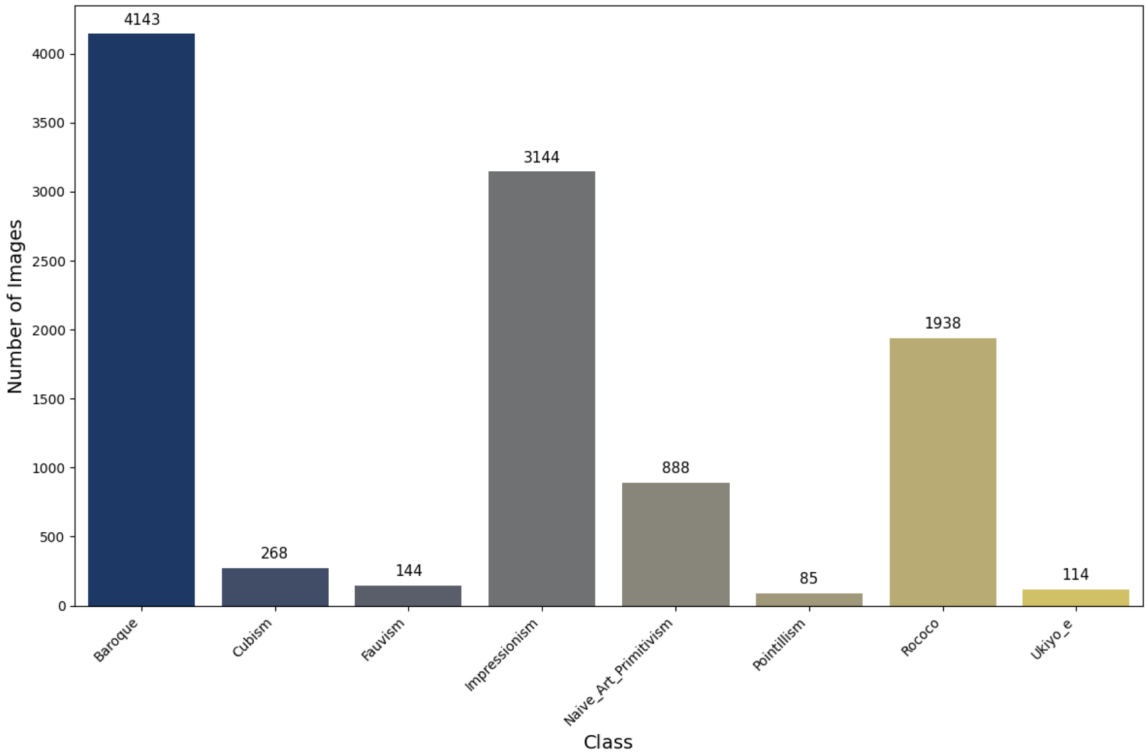


Figure 3.6 Distribution of images according to their styles. Classification based on style using Cropped Facial Regions.

4 Experiments

In this section, we discuss the results of the experiments. The first experiment compares pre-trained models for the style classification task. The main goal of this experiment is to explore how different architectures can perform for this task. The next experiments are dedicated to the combined approach that uses the deep learning method as feature extraction in combination with hand-crafted features extracted by computer vision algorithms and performs classification using a classical machine learning classifiers. In the first part, we examine the performance of these classifiers on different layers of deep learning models for the style classification task. After that, we select the best-performing models and their corresponding layers for each classifier and conduct a parameter search. In the second part, we apply these tuned classifiers in combination with the best-performing model and its layer to the genre classification task. In the third part, we apply the same methods for the style classification of images containing only faces extracted from paintings. All experiments are performed using the dataset described in Chapter 3 and evaluated using weighted F1 score. The following sections describe the implementation details and results achieved.

4.1 Pretrained Models

In this experiment, we compare several pre-trained models described in Chapter 2. All models are pre-trained on the ImageNet dataset. We used PyTorch as a framework for the experiments. To implement training, first, we freeze the network's backbone, which makes it use constant weights that it gets during the training on the ImageNet dataset. Then we remove the original classification head and implement our own classification layer represented in Table 4.1. During the training process, only its weights change. To make all networks comparable, we use the same classification layer across all models so that differences in performance come from the backbone architecture, not the classification head. We performed 25 epochs of training for each model with early stopping based on the value of the weighted F1 score. For training, we use the Adam optimizer and the cross-entropy loss function.

The dataset was divided into a training set of 80% and a test set of 20%.

Layer	**Input → Output**	**Details**
Linear	*in_features* → 256	Fully connected layer
ReLU	256 → 256	Activation function
Dropout	256 → 256	$p = 0.5$
Linear	256 → *num_classes*	Fully connected output layer

Table 4.1 Classifier architecture

Each of these networks has its disadvantages and advantages that can be beneficial for our task. Our goal was to compare networks of different sizes and architectural implementations. In Table 4.2, we can observe that the best result was achieved by MobileNetV3Large, DenseNet121, ResNet50, and EfficientNetB1, which showed a high F1 score. This can be caused by the fact that the backbones have efficient feature

Model	Train F1	Test F1	Test Precision	Test Recall
DenseNet121	77.37	75.35	75.45	75.44
EfficientNetB0	77.09	71.55	71.73	71.79
EfficientNetB1	79.93	76.81	77.1	76.98
ResNet18	71.84	72.7	72.96	72.7
ResNet30	73.88	72.49	72.6	72.57
ResNet50	75.70	75.2	75.33	75.27
MobileNetV2	76.79	73.15	73.44	73.6
MobileNetV3Large	79.22	75.08	75.41	75.47
Vision Transformer	92.52	82.34	82.37	82.49

Table 4.2 Pretrained CNNs for the style classification task.

representation and provide highly discriminative features. In the case of DenseNet121, dense layers provide the opportunity for strong feature reuse, giving the classifier access to diverse features, including early extracted features, which can be crucial for art classification. MobileNetV3Large utilizes lightweight depthwise separable convolutions with squeeze-and-excitation modules, which allow the extraction of only relevant features that can also influence the effectiveness of the backbone for art-style recognition. This can be important because styles share the same visual elements, such as textures or colors; therefore, it can be harder to extract useful features for some classes. Also, we can observe that EfficientNetB0 achieves a lower F1. EfficientNet was designed with parameter efficiency as its top priority in adapting to the ImageNet dataset. This means that the depth, width, and resolution scales were chosen to perform well on ImageNet. However, the classes for art classification differ from ImageNet, causing lower performance for EfficientNetB0. But its scaled version, EfficientNetB1, has more parameters and deeper and wider layers, allowing it to extract richer, more abstract features and achieve a high F1 score.

In contrast, smaller backbones showed lower accuracy. For example, ResNet18, the shallowest selected model with a limited 18-layer depth, utilizes fewer convolutional filters and, as a result, is able to extract less complex feature hierarchies, making it harder for the frozen model to distinguish differences between classes in the WikiArt dataset. The intermediate ResNet30 shows a slightly better F1 score, but it is lower than ResNet50. ResNet50 is deeper than ResNet30 and ResNet18; it allows the extraction of more complex features for classification. In addition, ResNet architecture implements feature reuse, as DenseNet121, but is less strong, which, in combination with depth, shows high results compared with other selected models. MobileNetV2 is lightweight but shallow compared to other models, so it may not extract sufficiently rich features for our classification task.

Vision Transformer showed higher performance compared to CNN models. It can happen because it is able to capture dependencies and global relationships in the picture using self-attention. It helps to detect the overall composition and style of the painting.

From the results we have achieved, we can observe that relative performance depends on architectural strengths. Deeper and more innovative models extract more diverse features, improving generalization in test data and performance. At the same time, shallower models or models with compact architectures provide more limited feature extraction, resulting in lower F1 scores.

4.2 Feature Extraction

In this part, we experiment with combining deep learning methods and classic machine learning approaches for art classification. We aim to experiment with deep learning methods as feature extractors in combination with computer vision algorithms. In addition, we compare the performance of selected machine learning methods for classifying the resulting features. Firstly, we run experiments for style classification. We extract features from deep learning methods, combine them with low-level features extracted by computer vision algorithms, and then pass the resulting vector to the machine learning classifiers. We evaluate the performance of classifiers on features extracted from several layers of deep learning methods. Additionally, we perform a parameter search for the classifiers. Secondly, we apply the best-performing machine learning classifiers and best feature extraction strategies to the genre classification task and style classification task using the cropped face regions.

Algorithm

CNNs have proven to be highly efficient for image classification tasks. They are able to identify low-level patterns, such as edges and textures, as well as high-level representations related to composition. Each CNN layer is designed to extract increasingly complex and abstract features from the input data. As the data progresses through convolutional layers, the network extracts features starting from low-level features such as edges, basic textures, corners, and colors, and as depth increases, the network can capture abstract representations like shapes and parts of the objects. Low-level features can be too generic because they can be shared across several classes, for example, repetitive texture can appear in Minimalism and Naive art, so they can be less efficient in distinguishing between complex and abstract classes, such as different artistic styles. While deeper layers focus on capturing more complex and abstract information, they can overlook fine-grained visual characteristics such as brushstroke textures. Moreover, with increasing network depth, the image is downsampled multiple times. As a result, the resolution is reduced, and information about the exact placement of elements in the image can be lost. In our experiments, we are using pre-trained ImageNet models as feature extractors, the last layers of which can return features that are highly specific for the ImageNet classification task, which can also influence the performance.

In this experiment, we extract features from different layers of selected CNNs and compare their representativeness for the task. The output of the convolution layer is a set of matrices called feature maps. The output of a convolutional layer is a 3D tensor with the shape:

$$(H_{\text{out}}, W_{\text{out}}, C_{\text{out}})$$

where:

- H_{out} and W_{out} are the height and width of the output feature maps.
- C_{out} is the number of output channels, corresponding to the number of filters in the convolution layer.

To construct a vector of features from the output of the layers, we apply the Global Avarage Pooling technique that converts a set of feature maps of size $(H_{\text{out}}, W_{\text{out}}, C_{\text{out}})$ to a 1D vector of shape C_{out} by averaging each channel. This helps to create a compact

representation of feature maps and preserve important patterns without their exact position.

For this experiment, as feature extractors, we select three CNN models: ResNet18, DenseNet121, and MobileNet V3. As we discussed in previous chapters, feature reuse can be extremely beneficial for our task. Both ResNet18 and DenseNet121 implement it. The difference is that in the case of DenseNet121, the reuse of features is stronger because each layer connects all previous layers. The MobileNet V3 adopts a different approach, which can also benefit our task. SE blocks enhance important features and reduce noise, and the model is adapted for highly efficient computations. In ResNet18, we extract the features after each residual layer. In DenseNet121, we perform extraction after each transition layer between dense blocks; these layers reduce the number of channels, making the features more representative. MobileNetV3Large is composed of a sequence of inverted residual blocks, and we extract features after 5^{th}, 11^{th}, and 15^{th} blocks.

In the case of Vision Transformer, we are extracting the CLS token from the 6^{th} block. Our goal is to extract less abstract features that capture mid-level features rather than highly abstract representations. CLS token aggregates relevant information from the entire image, which allows it to represent global and local features, composition, and style information.

In addition, we enhance the vector achieved from the deep learning method with low-level feature extraction. We use the K-means algorithm to extract RGB coordinates of the major color palette with k = 15. A higher value of k can produce a confusing vector because the palette will be more diverse and contain intermediate colors.

Furthermore, we use the LBP to capture fine-grained textures to ensure that the feature vector contains both micro- and macro-details of the painting. LBP is a robust, efficient rotation-invariant texture descriptor, making it ideal for capturing brushwork details and abstract textures. We utilize LBP with $P = 8$ (number of neighbors used to compute LBP) and $R = 1$ (radadius of the circle used to select neighbor pixels), and $P = 24$ and $R = 3$ to build a histogram of 10 and 26 bins.

Furthermore, we enhance our vector with statistical features such as standard deviation and mean. The standard deviation measures the variation in color or intensity, indicating the level of contrast and color diversity. The mean, on the other hand, measures the overall color tone of the image. In Impressionism and Rococo painting, we observe high variance because artists often use dark and light tones. In contrast, Minimalist painting has lower variance because colors are often blended. For styles where the majority of colors are light, the mean is high, and the mean is low if dark colors are used often. Each color space offers a unique perspective on how color is represented and perceived. To achieve this, we transform the image into several color spaces (CIE Lab, YCbCr, and HSV). Mean and standard deviation are computed from the three channels of all color spaces. After this, the resulting vector of mean and standard deviation is concatenated with the common feature vector. CIE Lab offers a perceptually uniform color representation, which means that numerical differences between values align with how humans perceive visual differences. YCbCr separates luminance from chrominance, allowing a clearer analysis of structural and color components. In addition, HSV captures aspects of color such as hue and saturation. Figure 4.1 represents the obtained feature vector.

Figure 4.1 Feature vector where A - features extracted using deep learning model size is represented in Table 4.3, B - LBP, size is 36, C - color palette size is 45, D - statistical features, size is 18

Table 4.3 represents the size of the feature vector passed to classification. Each vector contains features extracted from a deep learning method combined with hand-crafted features. The size of hand-crafted features is 99.

Model	Layer	Size of combined vector
A	1	163
	2	227
	3	355
	4	611
B	5	139
	11	211
	15	259
C	1	227
	2	355
	3	611
D	6(CLS)	867

Table 4.3 The sizes of feature vectors. Models: A - ResNet18. B - MobileNetV3Large, C - DenseNet121, D - Vision Transfor.

Vizualization

To visualize features, we used Uniform Manifold Approximation and Projection (UMAP) dimensionality reduction technique. It constructs a fuzzy topological representation of the data in high dimensions. Then, it optimizes a low-dimensional representation by minimizing the cross-entropy between the high-dimensional and low-dimensional representations; it allows the method to preserve both the local and global structure of data in the low-dimensional embedding.

We used the parameters: $nneighbors = 10$, which determines the size of the local neighborhood; $mindist = 0.05$, which controls how closely UMAP places points together in the low-dimensional space; $metric = cosine$, which defines the distance function used. Before applying UMAP, we standardize features and reduce the dimensionality using Principal Component Analysis (PCA) with $ncomponents = 0.95$ to have 95% of the total variance preserved.

Because UMAP relies on distances, it can be less informative in high-dimensional spaces. PCA helps project data into a space where these distances are more informative. Before applying the PCA, the dimensionalities of vectors were: Resnet18 - 355, MobileNetV3Large - 259 and DenseNet121 - 611, ViT - 867. Figure 4.2 represents a visualization of processed features extracted from ResNet18 layer 4, MobileNetV3Large 15^{th} block, DenseNet121 transition 3, Vision Transformer CLS token from 6^{th} block.

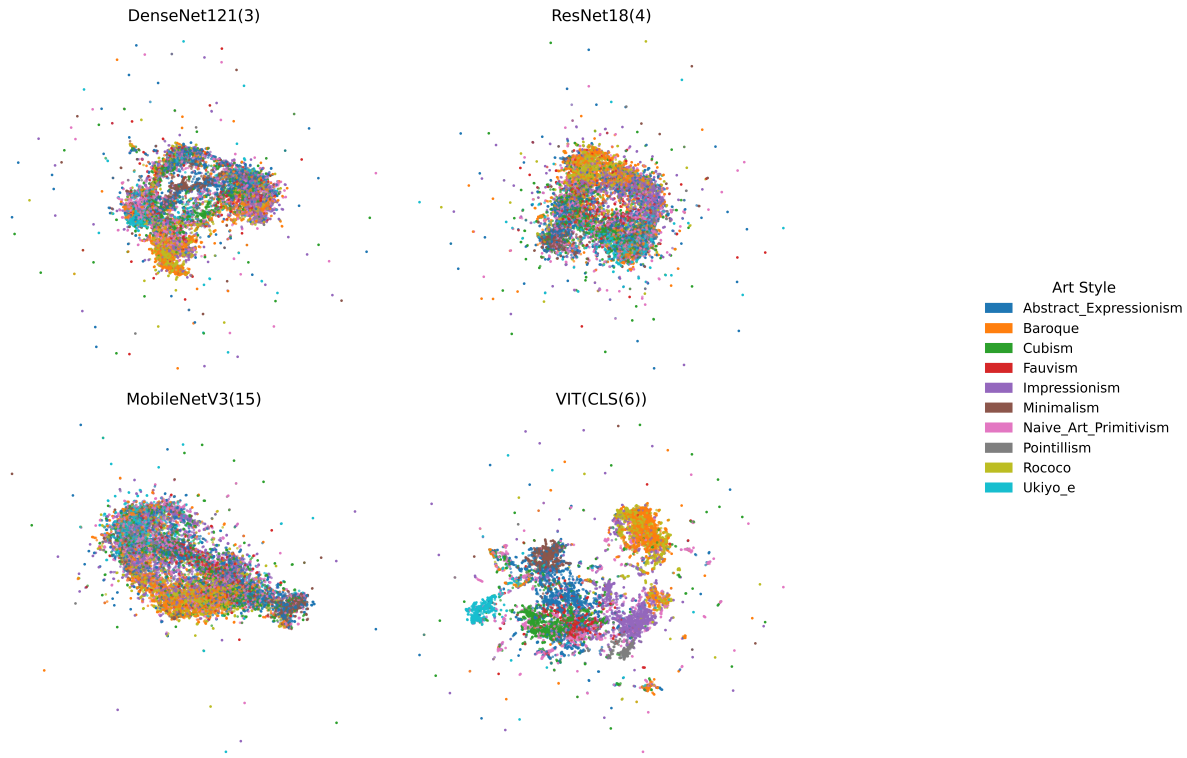


Figure 4.2 Extracted features for Style Classification.The caption of the picture represents the model name and the layer from which features were extracted.

On the graphs represented in Figure 4.2, we can see that some features are separable, but there is an overlap for some classes. For instance, in all represented models, Baroque overlaps with Rococo, but neither of them overlaps with Ukiyo-e or Cubism. Overlapped classes have more common characteristics than nonoverlapping ones, which can cause model confusion in distinguishing these classes. Baroque and Rococo, for instance, share similarities, such as the use of ornaments, working with the contrast of light and dark, and curved lines. In comparison, Ukiyo-e and Cubism have completely different characteristics that set them apart. We can observe that features extracted by CNN tend to overlap more frequently, while features extracted by Transformers show less overlap. It can be caused by architectural differences. As a result, Transformers are able to extract more separable features, even among similar classes. Moreover, features of Transformer represent more compact and separable clusters than those extracted by CNNs. This can influence which classification models will give better performance, linear classifiers like SVM or more advanced techniques like XGBoost. In all visualizations, we can see outliers. These outliers may result from authors employing entirely different and revolutionary techniques, even though they are still categorized under a specific style, which can lead to misclassifications. In the case of ResNet18, we can still see compact clusters with some overlap, especially for classes that do not differ much, while MobileNetV3Large provides widespread data clusters with a high percentage of overlap.

4.2.1 Style Classification

In this part, we experiment with feature extraction for the style classification task. Firstly, we test selected machine learning classifiers across several layers of deep learning models. The model provides vectors from layers of different depths, including the early

layers and the deepest; these vectors are concatenated with the low-level features discussed in the previous section. Achieved feature vectors are used as input to the classifiers. Before applying classifiers, the extracted features were standardized, which is needed because SVM, MLP, and KNN are sensitive to the scale of features. The parameters of the tested classifiers were intuitively selected, without parameter search. Performance was evaluated using stratified 5-fold cross-validation, with a weighted F1 score as the primary metric, because the dataset is unbalanced.

Model	**Layer**	**RF**	**SVM**	**KNN**	**MLP**	**XG Boost**
A	1	82.3 ± 0.7	65.6 ± 1	49.3 ± 1	68.5 ± 1	81.2 ± 0.8
	2	84.3 ± 0.9	77.6 ± 0.7	60.6 ± 0.5	79.7 ± 0.7	83.5 ± 1
	3	81.7 ± 0.8	79.6 ± 0.6	59 ± 0.6	79.7± 1	80.5 ± 0.7
	4	79.4± 0.6	75.2± 0.5	44 ± 0.4	75.3 ± 0.5	79 ± 0.7
B	5	80.1 ± 0.6	59.01± 0.7	42.5± 1	61.2 ± 1	78.4 ± 0.7
	11	80.1± 0.2	65.5 ± 0.5	38.9 ± 0.8	66.1 ± 0.9	78.6± 0.5
	15	79.9 ± 0.5	67.2 ± 0.5	38.7 ± 0.6	68.4 ± 1	78.8 ± 0.7
C	1	82.9 ± 0.6	69.5 ± 0.6	56.1 ± 0.9	73.6 ± 0.7	82.2 ± 0.9
	2	82.7 ± 0.6	76.7 ± 0.4	57.4 ± 0.7	80.6 ± 0.6	82.1± 0.7
	3	81.5± 0.6	78.4 ± 0.9	49.2 ± 0.6	80.3± 0.8	81± 0.7
D	6(CLS)	88.8 ± 0.7	89.9 ± 0.7	75.5 ± 0.7	91.1± 0.8	88.5 ± 0.7

Table 4.4 F1 score of classifiers applied to different features. Models: A - ResNet18. B - MobileNetV3Large, C - DenseNet121, D - Vision Transfor. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting

As we can see in Table 4.4, the performance of classifiers is different. Moreover, it differs across layers for each deep model. As we discussed in the previous part, the complexity of extracted features varies across layers; the shallowest layers extract low-level features that are less informative for class recognition, while deeper layers extract more extensive features. We can observe that on the shallowest layers, all models show lower performance compared with deeper layers. Although the deepest features are not universally optimal, most classifiers show the highest performance in intermediate layers. This can be caused by the fact that intermediate layers encode mid-level patterns offering rich, moderately abstract features. These features are not too compressed or task-specific as in the last layers. This feature diversity is beneficial for classifiers. Our models are pre-trained on the ImageNet dataset, so the last layers are specific to this task. The art classification task differs from the ImageNet classification task and can cause features to be less representative and the F1 score to be lower. Moreover, these features can be too abstract and do not contain low-level information, which is crucial for art classification tasks.

We can observe that the overall performance of ensemble methods is higher among classifiers that use CNN-extracted features. Both Random Forest and XGBoost perform better than other classifiers on different layers of the models. SVM and MLP show intermediate performance for CNN features, while the highest is for ViT. This can be caused by the different nature of the extracted features. On the one hand, CNN extracts localized patterns from the images, and each convolution filter detects specific patterns, such as texture. When we apply Average Pooling, each dimension often has a meaning and may strongly indicate a particular style. As a result, vectors tend to have several

informative values that help to distinguish classes. However, the Vision Transformer, using a self-attention mechanism, generates a global representation of the image through the CLS token, which means that useful information spreads across dimensions. CNN features have meaningful individual dimensions, and, as a result, tree-based models can easily split the data based on them and make predictions. Moreover, these classifiers implement internal feature elimination, so they focus only on relevant features, and they tend to be more robust because they combine predictions of several classifiers. We observe that ViT features showed linear separability in the visualizations from the previous section. Visualized features create well-formed and distinct clusters, making classifiers that rely on separation (such as SVM and MLP) more effective. CNN features require complex, non-linear rules to separate overlapping regions. KNN shows the lowest F1 score for both CNN and Transformer features; this can be caused by the fact that the classifier relying on distance suffers from high dimensionality.

As a next step, we perform the parameter search for the best-performing features for each classifier. We selected from Table 4.4 the layers of the models where the classifiers showed the highest F1 score and performed a parameter search. For this purpose, we use the Optuma library. We utilize the same set of parameters for each classifier independently of the feature set for which it is used. We can see in Table 4.5 that performance changed significantly for some models, such as SVM, KNN, and MLP; this means that the initial configuration of the parameters was not appropriate for these classifiers.

Classifier	Model (Layer)	k-fold F1
RF	DenseNet121 (2)	83.2 ± 0.5
	DenseNet121 (1)	83.1 ± 0.5
	MobileNetV3Large (5)	80.6 ± 0.5
	ResNet18 (2)	84.6 ± 0.8
	MobileNetV3Large (11)	80.6 ± 0.2
	ViT (6(CLS))	89.5 ± 0.7
SVM	DenseNet121 (3)	81.9 ± 0.3
	ResNet18 (3)	83.6 ± 0.2
	MobileNetV3Large (15)	75.8 ± 0.9
	ViT (6(CLS))	92.5 ± 0.4
KNN	ResNet18 (2)	81 ± 0.9
	DenseNet121 (2)	79.2 ± 0.9
	MobileNetV3Large (5)	71.1 ± 0.7
	ViT (6(CLS))	89.4 ± 0.7
MLP	ResNet18 (2)	80.6 ± 0.7
	DenseNet121 (3)	81.6 ± 0.6
	MobileNetV3Large (15)	76 ± 0.2
	DenseNet121 (2)	80.2 ± 2
	ResNet18 (3)	81.5 ± 0.5
	ViT (6(CLS))	91.4 ± 0.6
XGBoost	DenseNet121 (2)	84.9 ± 1
	MobileNetV3Large (5)	81.3 ± 0.4
	MobileNetV3Large (15)	80.7 ± 0.6
	MobileNetV3Large (11)	80.9 ± 0.2
	ResNet18 (2)	85.9 ± 0.7
	DenseNet121 (1)	84.5 ± 0.6
	ViT (6(CLS))	91.8 ± 0.5

Table 4.5 F1 scores from one run and k-fold cross-validation. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting

From Table 4.5 we can observe that the best performance was achieved by XGBoost ResNet18(2) F1 = 85.9 and Random Forest ResNet18(2) F1 = 84.6. Most classifiers showed their best performance for features selected from the ResNet18 model combined with hand-engineered features; the only exception is MLP, which achieved the best performance for features that include features extracted from DenseNet121(3).

As we discussed in previous chapters, both these models implement feature reuse, which is beneficial for the art classification task because it helps to preserve low-level features. For future investigation, we select the best-performing layer and network for each classifier represented in Table 4.5. We evaluate these classifiers by performing one run and calculating confusion matrices and precision-recall curves.

The confusion matrices (Figure 4.3) are normalized per class, where the values in each row represent the percentage of examples from the actual class. This is important for interpreting results, especially for underrepresented classes, because it allows for analyzing performance relative to each class size. The diagonal elements represent the percentage of correctly classified examples for each class, while the off-diagonal elements represent the

percentage of confusion.

From the confusion matrices, we can observe that the smallest class, Pointillism, was correctly classified in 88% of the cases by XGBoost; a similar performance (84%) was shown by Random Forest, and both models showed a high F1 in Table 4.5. However, SVM, which is the next in performance, classified Pointillism only in 79%, while KNN, which has a lower F1 score in Table 4.5, classified it in 82% of the cases.

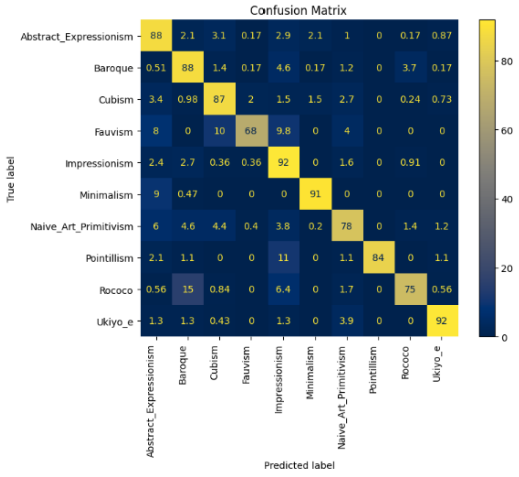
Fauvism is the other small class, and all models show a lower percentage of correct classification for this class; for almost all models, it is the lowest value compared with other classes. The best performance (77%) was shown by MLP DenseNet121(3). Both Pointillism and Fauvism show the highest percentage of confusion with Impressionism. Pointillism evolved directly from Impressionism, while Fauvism was inspired by Impressionism. Impressionism, Fauvism, and Pointillism use bright and clean colors, avoiding the black lines and the usage of visible brushstrokes. Fauvism and Impressionism are similar because both styles focus on expressing feelings rather than objects. As we can see, all three styles share similarities and, as a result, in combination with the low number of examples of Pointillism and Fauvism, confusion of these classes with Impressionism occurs. Although there is a low number of confusions for Impressionism with Pointillism or Fauvism, despite the described similarities, this can happen because Impressionism is one of the biggest classes in the dataset, which allows classifiers to have enough different examples for training.

The other smallest class is Minimalism. But almost all classifiers show outstanding performance for this class; the best is XGBoost ResNet18(2) (92%) and Random Forest ResNet18 (2) - 91%, and the lowest percent is 80%, shown by KNN ResNet18 (2), which is still a good result. This class is distinguished from other selected classes, which can be observed from the confusion matrices. It has a low percentage of confusion with other classes across all classifiers. The only exception is Abstract Expressionism; all confusion matrices show a high number of confusions between Minimalism and this style, but not the other way around. It happens because Abstract Expressionism has more examples than Minimalism, so models can learn to distinguish them better. But confusion can be caused by the fact that some Abstract Expressionist paintings have large fields of color, which we can observe in Minimalism paintings too. Moreover, both styles are abstract, and some works have low feature density; artists often use large, uniform areas of color and minimal shapes, which can also make the extracted features for these styles similar to the classifiers.

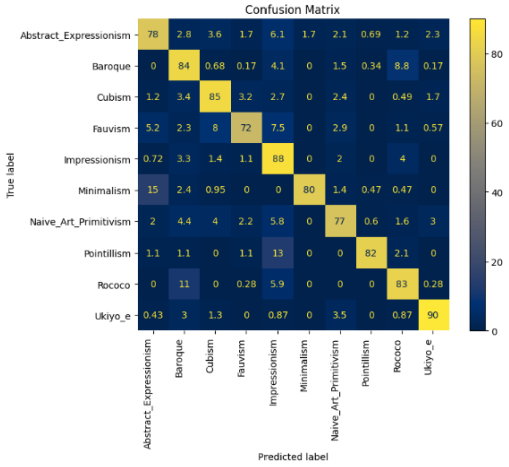
Classifiers distinguish Cubism in most cases, and this class has a low number of confusions with other styles across all classifiers. But Fauvism is often confused with Cubism. It can happen because some Fauvism paintings have simplified shapes and flattened perspectives. Cubism has many more examples than Fauvism in the dataset, which is why Cubism has a low number of confusions with Fauvism. Baroque and Rococo styles share some features. This may happen because Rococo evolved from the late Baroque.

All classifiers distinguish Baroque from Rococo quite well, but classifiers often confuse Rococo with Baroque. The same is true of Naive art. Despite the fact that it can be similar to Fauvism and Cubism, in some cases, it shows a lower level of confusion with these two classes.

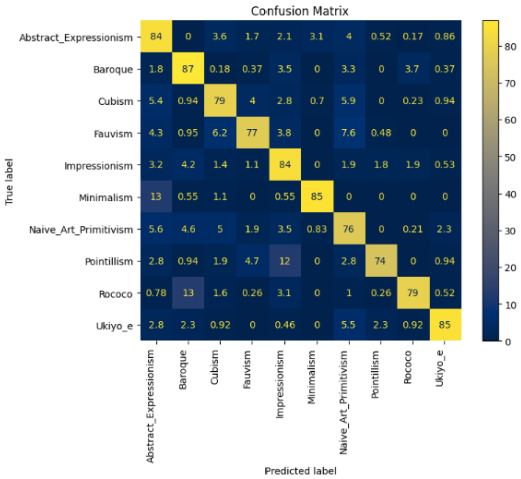
In Figure 4.4, there are confusion matrices of the classifiers that used Vision Transformer features in combination with hand-engineering features. We can see that the best performance is shown by SVM, followed by XGBoost and MLP. All models showed



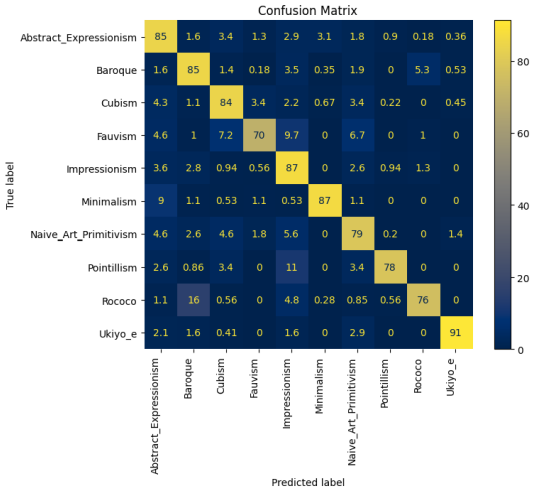
(a) RF ResNet18(2)



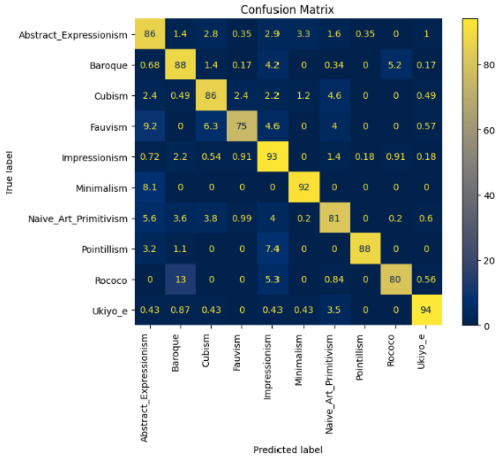
(b) KNN Resnet18(2)



(c) MLP DenseNet121(3)

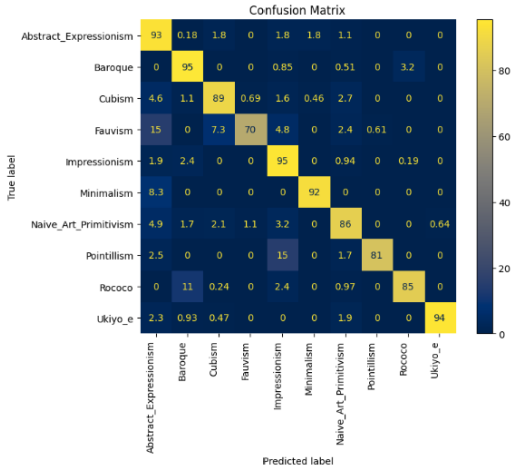


(d) SVM ResNet18(3)

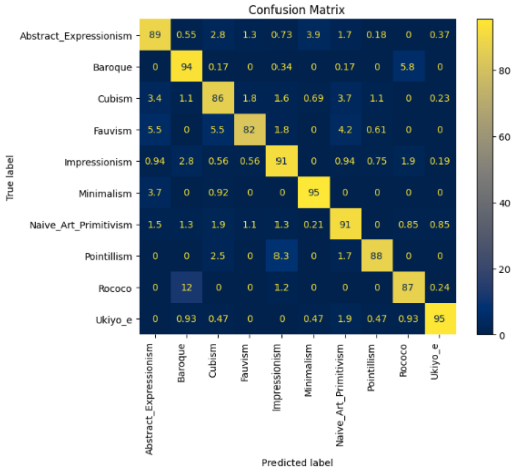


(e) XGBoost ResNet18(2)

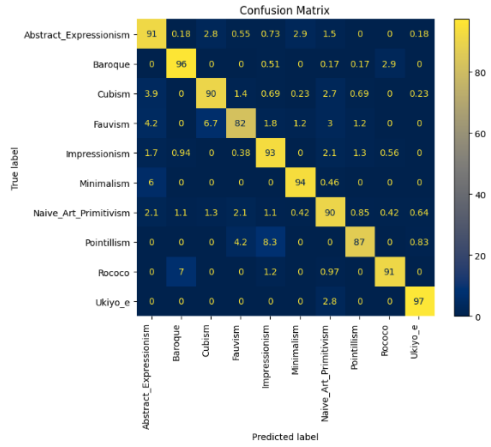
Figure 4.3 Confusion matrices for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.



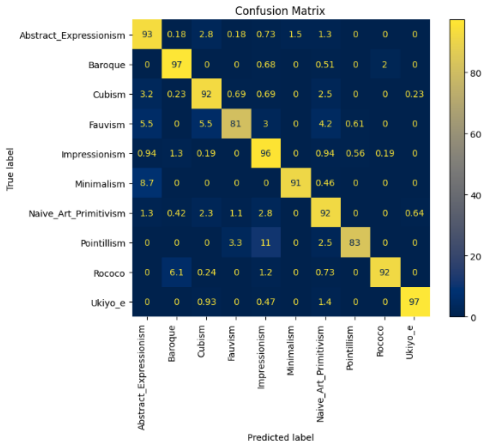
(a) RF ViT (6(CLS))



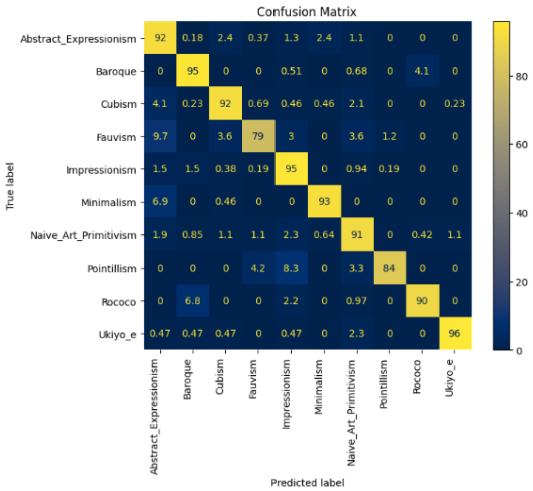
(b) KNN ViT (6(CLS))



(c) MLP ViT (6(CLS))



(d) SVM ViT (6(CLS))



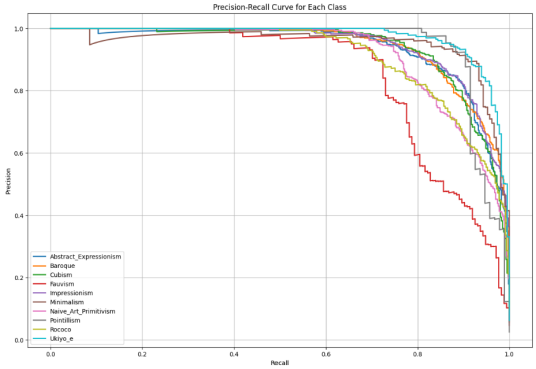
(e) XGBoost ViT (6(CLS))

Figure 4.4 Confusion matrices for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.

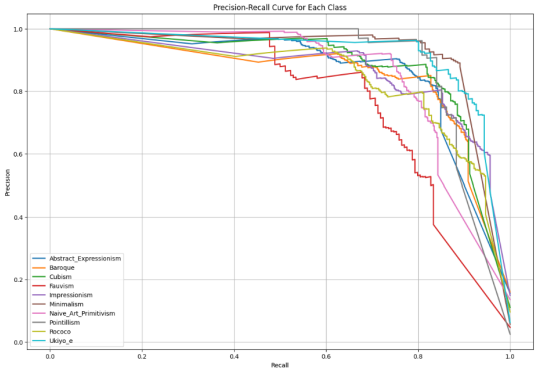
higher performance on features that included features extracted from ViT compared with classifiers with CNNs features. CNN and Transformer analyze images differently. CNN focuses more on local features and patterns, while Transformer focuses on global shapes and forms. This can be the reason for the better performance, but this can also lead to different difficulties in style differentiation for this deep learning method. In general, we can observe that for the most similar classes (like Baroque and Rococo or Pointillism and Impressionism), classifiers with CNN features and ViT features have the same confusion, but in the case of classifiers that use ViT features, a percentage of the confusion is lower. Therefore, we can observe better performance for similar classes. All classifiers that used hand-engineered features combined with CNN features demonstrated confusion between other classes and Impressionism, even if this class shares fewer similarities at first glance. Classifiers utilizing ViT features do not experience this confusion; the high level of confusion associated with Impressionism occurs only for classes that closely resemble it, such as Pointillism.

As a next step, we analyze the precision-recall curves for each class. This metric is very representative for imbalanced datasets, because it shows how well a model can identify positive samples by plotting precision against recall at different threshold levels. Precision measures the proportion of correctly predicted positive samples among all predicted positives, while recall measures the proportion of correctly predicted positives among all actual positives. Figure 4.5 represents Precision-Recall curves for the classifiers that used CNN features combined with hand-engineering features. According to the previous analysis, the best performing classifiers for CNN extracted features combined with hand-engineering features are Random Forest and XGBoost. We can observe that the curves appear higher than the curves of other classifiers, which means the classifiers achieve higher precision for the same level of recall or higher recall for the same level of precision across different thresholds. This shows that these classifiers are more effective in distinguishing the positive class from all other classes. All models show the lowest curve for the Fauvism class. For MLP and SVM, the curve is always lower. At the same time, for KNN, Random Forest, and XGBoost, at first, it is as high as other curves, but after some point, it starts decreasing; this means that initially the model is confident, but after decreasing the threshold, the model adds more false positives. A low curve indicates that the model struggles with this class. The reason may be that Fauvism is underrepresented in our dataset. Moreover, this class has visual similarities to Impressionism, making this class harder to distinguish. In contrast, Pointillism, another underrepresented class, has a high curve for majority classifiers; only MLP and SVM are exceptions. Minimalism, which also has a low number of examples, shows high curves across classifications. As we discussed earlier, it has distinct features that make it easy to distinguish it from other classes. The highest curve across all classifiers shows the Ukyio-e class.

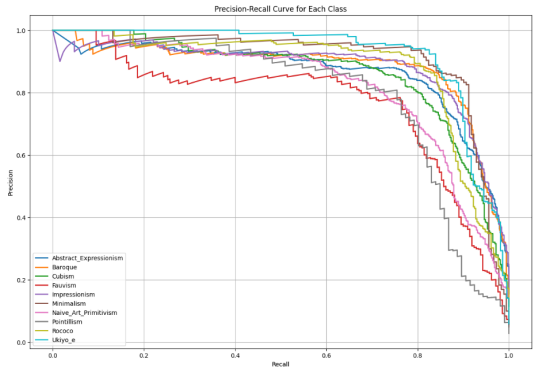
As we observed in the previous analysis, classifiers that used Transformer features combined with hand-engineering features showed better performance. The curves represented in Figure 4.6 are grouped and lie higher than those presented in Figure 4.5. This shows better performance and a high ability to distinguish classes. The performance of Ukyio-e is close to perfect for all classifiers. We can observe high precision across a wide range of recall values, which indicates that the model can correctly classify the majority of examples of this class and keep false positives low. Moreover, these classifiers introduce smoother and less steep curves without sharper drops compared with classifiers that use features that include CNN features. This means that classifiers are more stable and maintain more balanced performance across thresholds and stronger overall distinguishing



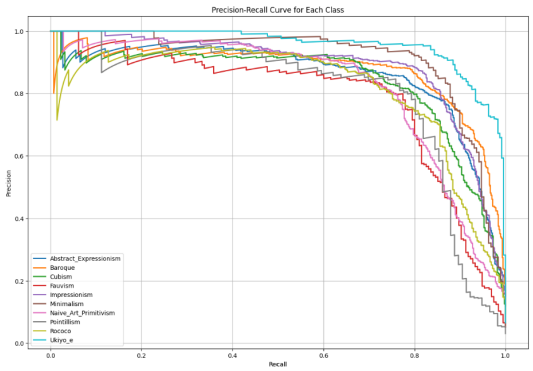
(a) RF ResNet18(2)



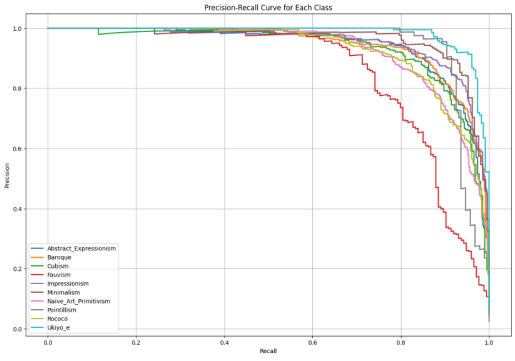
(b) KNN Resnet18(2)



(c) MLP DenseNet121(3)

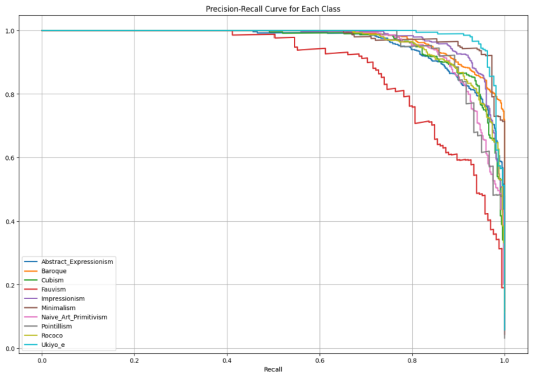


(d) SVM ResNet18(3)

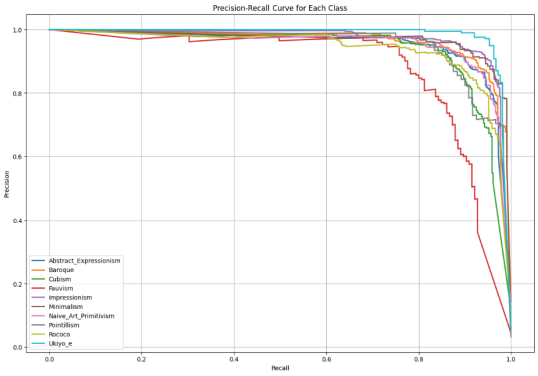


(e) XGBoost ResNet18(2)

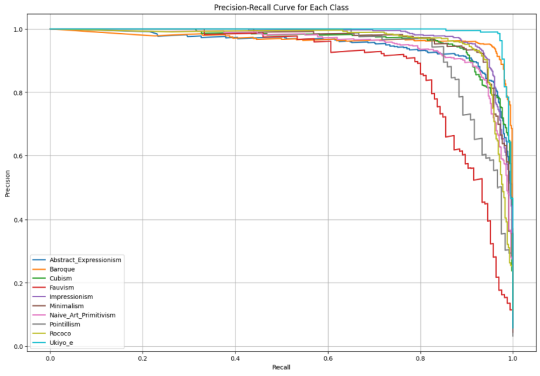
Figure 4.5 Precision-Recall curves for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.



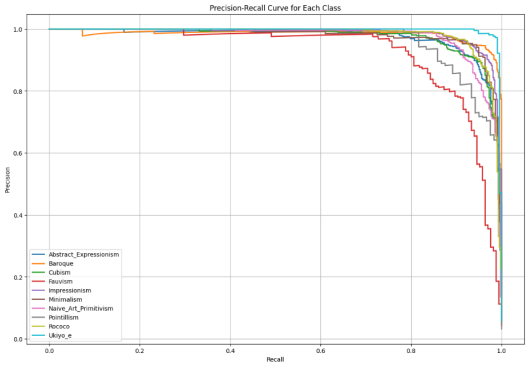
(a) RF ViT (6(CLS))



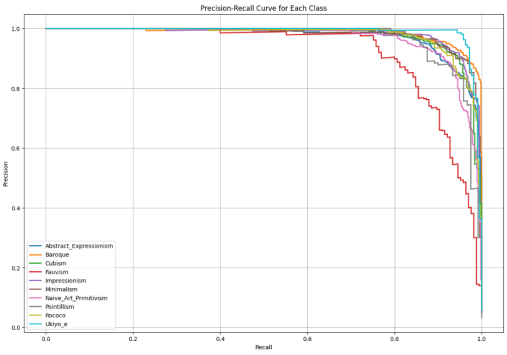
(b) KNN ViT (6(CLS))



(c) MLP ViT (6(CLS))



(d) SVM ViT (6(CLS))



(e) XGBoost ViT (6(CLS))

Figure 4.6 Precision-Recall curves for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM -Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.

among classes, while CNN-based classifiers show higher sensitivity to threshold selection.

As we observed in the previous part of the experiment, the classifiers showed a lower curve for Fauvism, represented in Figure 4.6. SVM and MLP classifiers showed the best F1 score, while they performed worse for Pointillism compared to other classes. In contrast, Random Forest, KNN, and XGBoost performed better for Pointillism. We observed the same behavior for the classifiers that used features that include CNN features in Figure 4.5.

4.2.2 Genre Classification

In this part of the experiments, we apply the classifiers that we found after parameter search in combination with the features that performed the best in the style classification task. Genre and style classification are related but different tasks; the main difference is the features that are important for distinguishing classes. In the case of style classification, color, texture, brushwork pattern, and composition are important. In contrast, genre classification focuses on the central theme and content in the picture. This task depends more on object detection and their arrangement and position in the painting. For instance, in marine painting, the sea is below while the sky is above, while in a portrait, a person or group of people is in the center, and the background is less detailed. Figure 4.6 represents

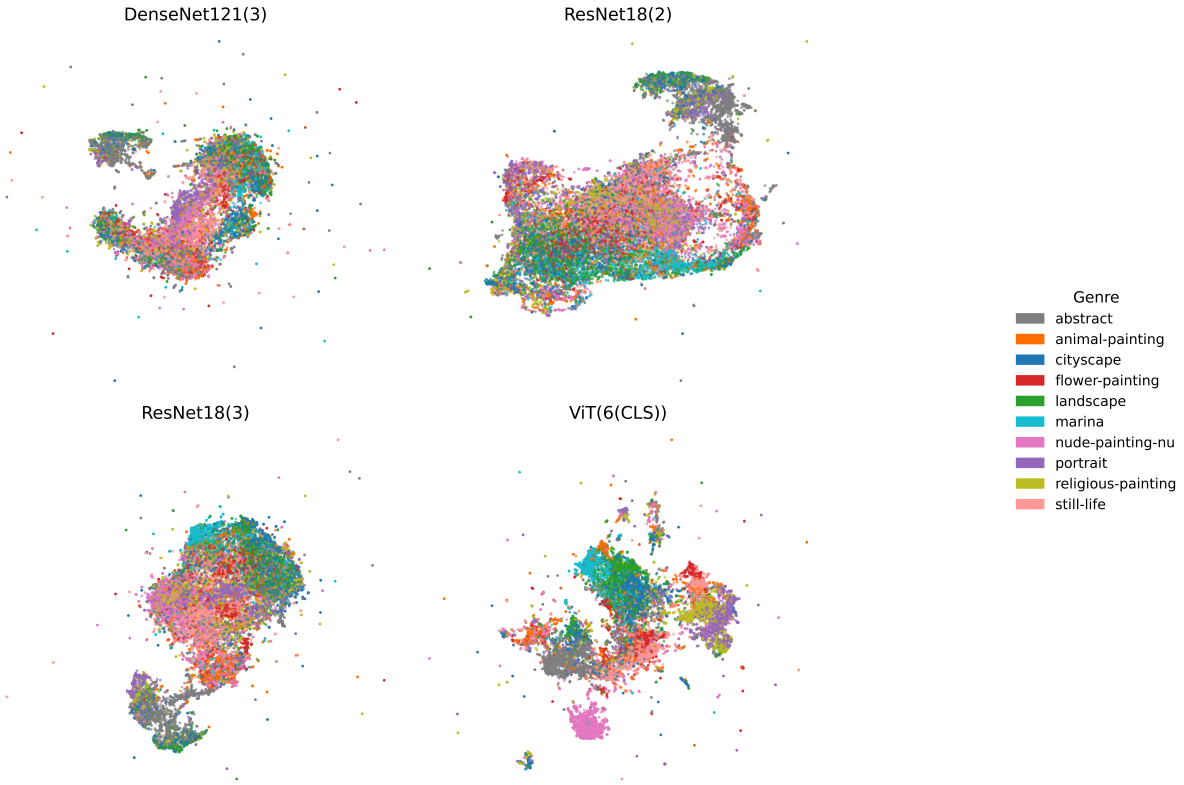


Figure 4.7 Features extracted from the selected layers of models. Genre Classification.

the visualization of features using UMAP, as described in the previous section. We can observe that ViT-based features create well-separated dense classes, while ResNet18-based features create more spread clusters. DenseNet121(3) features are also dense but more spread out compared to ViT-based feature clusters. This can influence which models will perform better for these features. For ViT-base features, linear classifiers can show strong

performance, while for CNN-based features, more advanced methods may be needed. We can observe that classes overlap with each other. Similar classes such as Nude art, Portrait, Landscape, Marina, Still life, and Flower painting are located close to each other. For CNN-based feature visualization, we can observe that the Abstract Art cluster is located far away from the others. The location of the features can influence which classes the classifier confuses and which it does not.

As in previous sections, we test our classifiers using stratified 5-fold cross-validation;

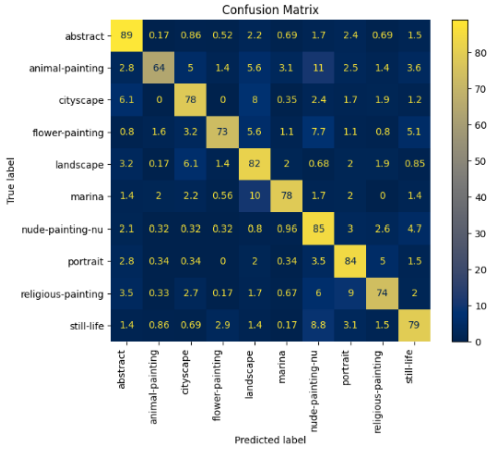
Classifier	**Model (Layer)**	**k-fold F1**
RF	ResNet18 (2)	80.3 ± 0.8
	ViT (6(CLS))	88.9 ± 0.6
SVM	ResNet18 (3)	83.4 ± 0.7
	ViT (6(CLS))	91.9 ± 0.3
KNN	ResNet18 (2)	75.4 ± 0.4
	ViT (6(CLS))	87.8 ± 0.7
MLP	DenseNet121 (3)	81.2 ± 0.8
	ViT (6(CLS))	91.6 ± 0.1
XGBoost	ResNet18 (2)	82.8 ± 0.8
	ViT (6(CLS))	91.8 ± 0.5

Table 4.6 F1 scores from one run and k-fold cross-validation. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting

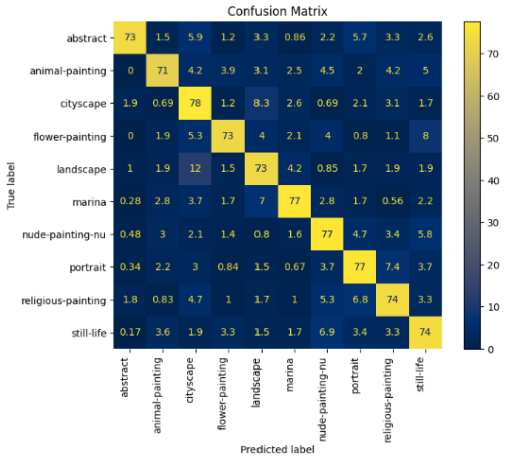
As in previous experiments, we evaluated classifiers using a weighted F1 score due to data set imbalance. Table 4.6 represents the F1 achieved during the experiment. For some classifiers, performance slightly decreased; for some, it decreased more. This happens because the examples differ more within the class than in the style classification task. For example, in the case of animal painting, the size of the pictured animals, their location, and their background can be different. In addition, the painting style can influence the accuracy of the classification. Animals represented in Primitivism differ from animals represented in more realistic styles. Also, we applied the best classifiers from the style classification experiment with parameters found for this task, so there is a possibility of achieving better performance if we run a parameter search and check the performance of classifiers across different layers of CNNs.

The best performance among classifiers that use CNN features was achieved by SVM ResNet18 (3), and the next two classifiers are MLP DenseNet121 (3) and XGBoost ResNet18 (2). It differs from the results we achieved for style classification, where the best performing classifiers were Random Forest ResNet18(2) and XGboost ResNet18(2). This can happen because MLP and SVM use features from deeper layers of CNN that are more representative of this task. XGBoost assembly technique where each next tree focuses on the misclassification of the previous one. This can be beneficial for this task because each class can overlap or be a subset of another one, for example, still life and flower painting or cityscape and landscape. As a result, simpler models cannot easily separate them.

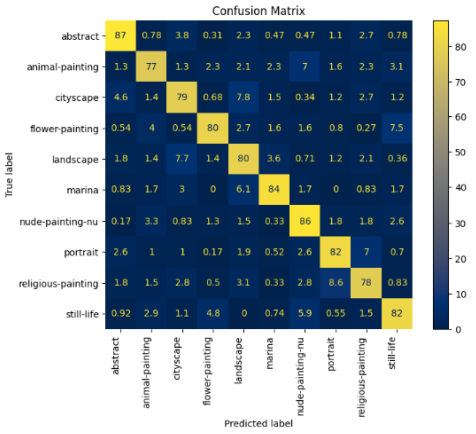
Among the classifiers that use ViT features, the best-performing are SVM and MLP, and XGBoost is the next. Moreover, classifiers that used ViT-based features overperformed classifiers that used CNN-based features. The same was observed for the style classification task. As a next step, we evaluate these classifiers by performing one run and calculating confusion matrices and precision-recall curves.



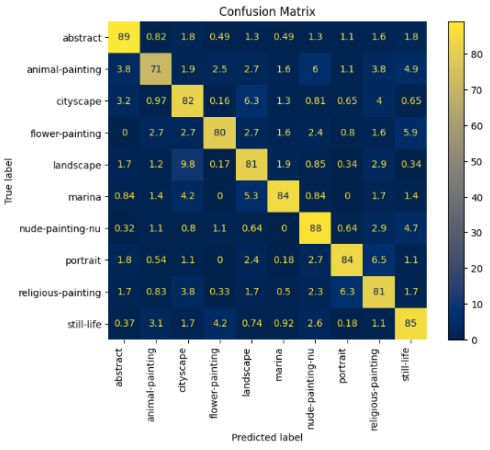
(a) RF ResNet18(2)



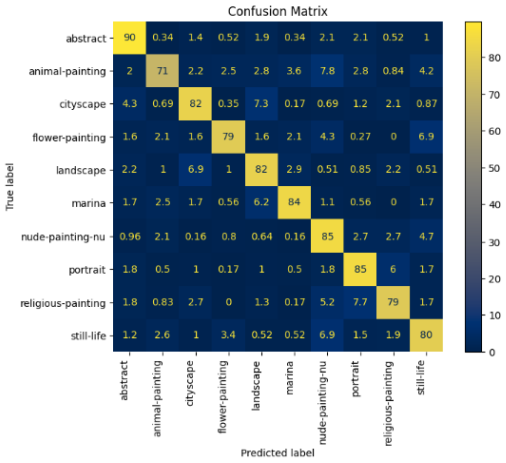
(b) KNN ResNet18(2)



(c) MLP DenseNet121 (3)



(d) SVM ResNet18(3)



(e) XGBoost ResNet18(2)

Figure 4.8 Confusion matrices for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.

Figure 4.8 represents normalized confusion matrices for classifiers that used CNN features. The cityscape and landscape show a high percentage of confusion with each other. These two genres are close and sometimes considered the same genre. Therefore, they may share certain features that can confuse classifiers, such as the horizon line and the sky gradient. In addition, some cityscapes can include a representation of nature. However, cityscapes can be confused with marina art because sometimes cityscapes can picture port cities, considered classifiers, and show a low number of confusion with this class. Some classifiers showed confusion with abstract art; the maximum number of this confusion showed Random Forest (6.1%). This can happen because some unconventional cityscapes with distorted forms could visually resemble an abstract composition, making them harder for some classifiers to recognize.

Flower painting is often confused with still life. Sometimes, flower painting can be considered a subset of still life because still life painting can picture vases with flowers or flowers themselves as part of the composition. Moreover, still life and flower painting can picture the same objects, for example, tables, tablecloths, plates, and fruits.

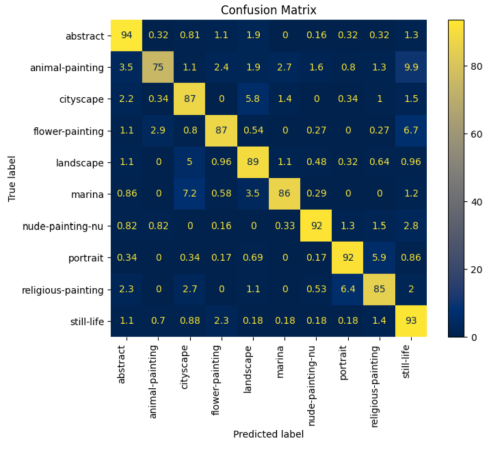
Marina art is often confused with landscapes. Marina art can be considered as part of the landscape, while both represent nature. In addition, some classifications distinguish seascapes as part of the landscape. Moreover, both genres can represent the same objects, such as the sky and the horizon.

Some classifiers sometimes confuse landscapes with marina art, but the percentage of confusion is not high (the maximum percentage of confusion was 4% for KNN). It can happen because some landscapes can picture a lake or sea, which can be misclassified as marina art.

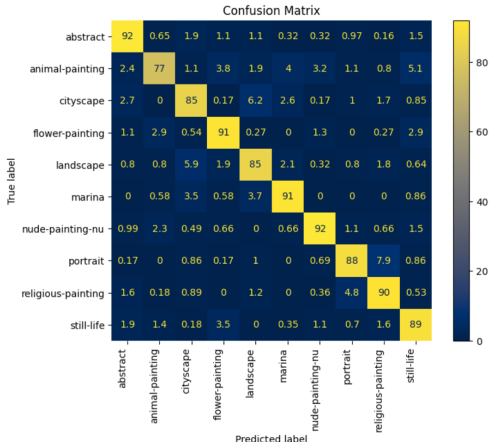
We expect that nude art will be confused with portraits, while both genres portray people's faces and gaze. Moreover, portraits, where a person is pictured with bare shoulders, can also be similar to nude for the classifiers. However, nude art is one of the best performing classes, and the biggest number of confusion with portrait is only 4.7% shown by KNN. The portrait also has a low percentage of confusion with nude art. But it is often confused with religious painting and the other way around. This happened because religious paintings often include central figures, such as saints, Christ, or the Virgin Mary, and essentially these paintings are portraits in composition.

Figure 4.9 represents normalized confusion matrices for classifiers that used the Vision Transformer CLS token as part of the feature vector. We can observe that the classifiers showed better performance compared to the classifiers that used CNN features from the previous part of the experiment. The percentage of correct classification for each class is higher than it was in the matrices in Figure 4.8. The most similar classes are still confused, but the percentage of confusion is lower. The overall number of confusions with classes that are not similar is lower as well. For example, Random Forest ResNet18 (2) showed confusion between animal paintings and nude art (7%). However, Random Forest ViT (CLS(6)) showed only 1.6% of confusion. Moreover, in the case of classifiers that used CNN features, cityscapes were confused with abstract art, the maximum value was shown by Random Forest ResNet18(2) 6.1%. In contrast, classifiers that used ViT features showed a low number of confusions for this class; the maximum value was shown by KNN (2.7%).

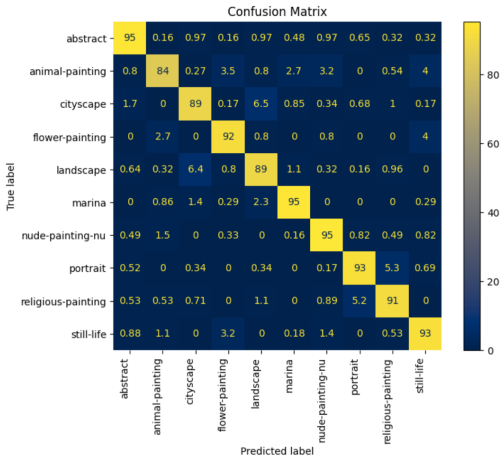
However, for animal painting, some classifiers that used Transformer features (KNN, Random Forest, and SVM) showed confusion with still life; the maximum confusion was shown by Random Forest (9.9%). Classifiers that used CNN features showed a low percentage of confusion for these classes. Visually, these two styles are not similar. But



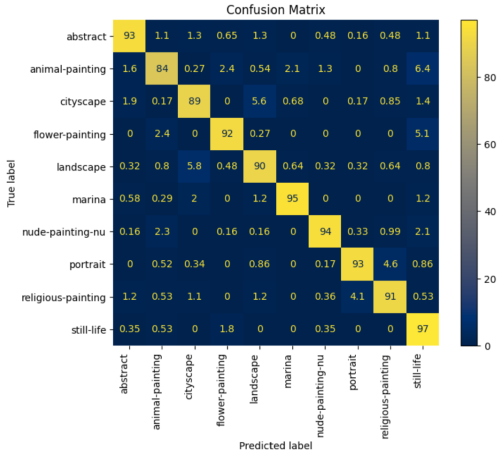
(a) RF ViT (6(CLS))



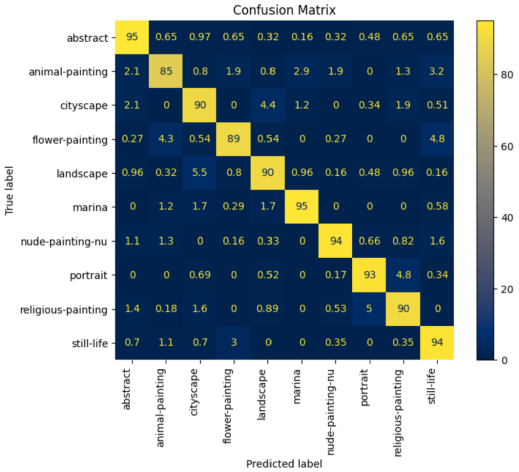
(b) KNN ViT (6(CLS))



(c) MLP ViT (6(CLS))



(d) SVM ViT (6(CLS))

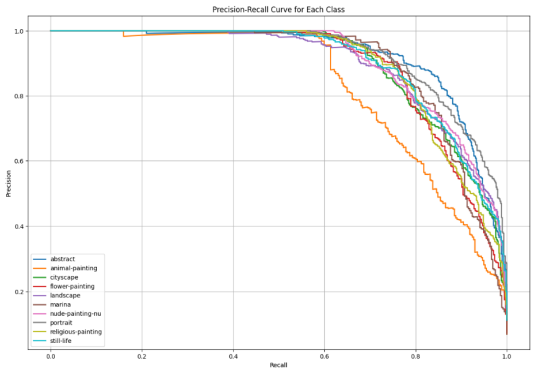


(e) XGBoost ViT (6(CLS))

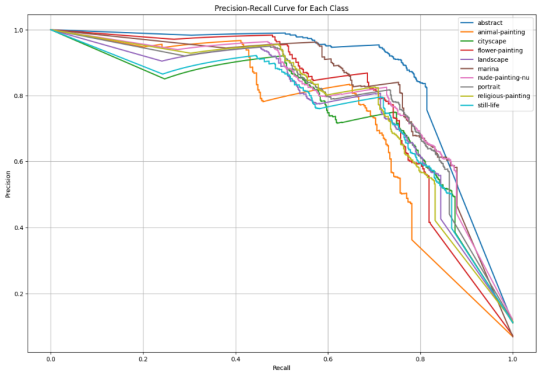
Figure 4.9 Confusion matrices for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.

still life can picture dead animals, and these paintings can create clusters of paintings that are harder to distinguish from animal art. As we discussed in the previous section, the way CNN processes images and Transformers is different. CNN focuses more on fine local details, while Transformer processes the whole image at once and summarizes it at once. Therefore, the Transformer focuses more on global shapes and forms and can miss small details like eye or fur texture. As a result, the achieved feature vector contains different information about the image, and this can lead to different confusing classes for classifiers that use CNN features and Transformer features.

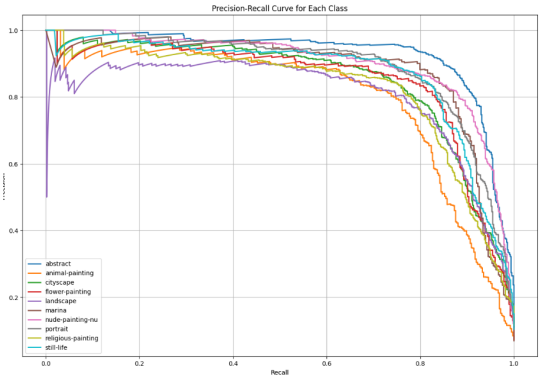
As a next step, we analyze the precision-recall curves of the classifiers that used CNN features combined with hand-engineered features, represented in Figure 4.10.



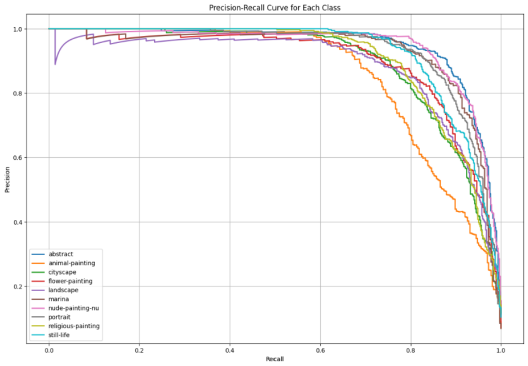
(a) RF ResNet18 (2)



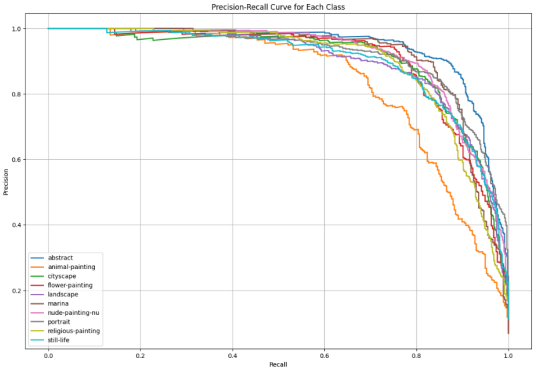
(b) KNN ResNet18 (2)



(c) MLP DenseNet121 (3)



(d) SVM ResNet18 (3)



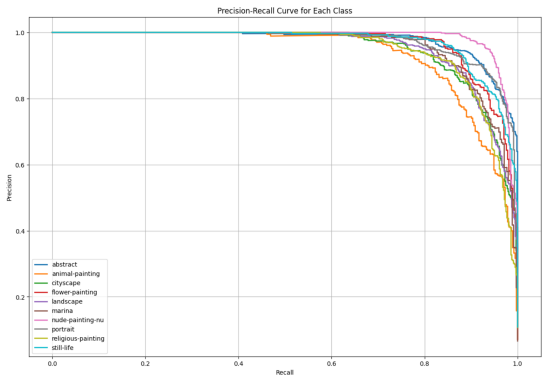
(e) XGBoost ResNet18 (2)

Figure 4.10 Precision-Recall curves for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.

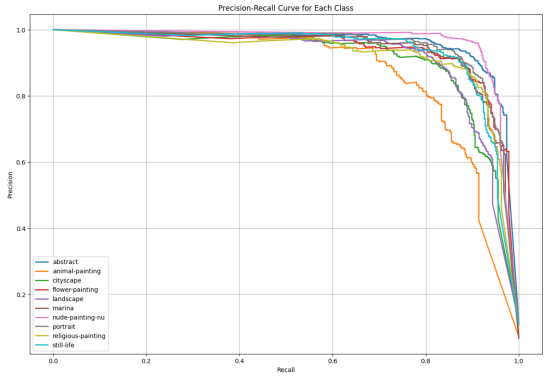
We can see that the curves of the best-performing classifiers (SVM, MLP, and XGBoost) are grouped and located higher compared to KNN and Random Forest. This indicates that classifiers are successful in distinguishing classes from other classes. The curves are close to smooth, indicating that the classifiers can maintain stable performance across different thresholds. The KNN curves are steep with sharp drops; even classes that are located higher, such as abstract art, exhibit this pattern. This means that classifiers can perform well for some points, but because their performance depends on the selected threshold, it can quickly drop with a threshold change. This can happen because there is

high-class overlapping among genres. Moreover, our data set is imbalanced. Therefore, it can be difficult for some models to separate classes. Random Forest showed better performance compared to KNN. The random forest curves are also smooth, but are located lower compared to the best-performing classifier curves. This means that Random Forest struggles with distinguishing classes, but it does not have sharp drops in performance. The MLP classifier showed a sudden drop at the beginning for the landscape class. The classifier generates low-confidence predictions for that class, which leads to an increase the false positives without an increase in true positives in its predictions. This can indicate that the model struggled to separate this class from the others at certain thresholds. The lowest curve has animal art, and the highest has abstract art across all classifiers.

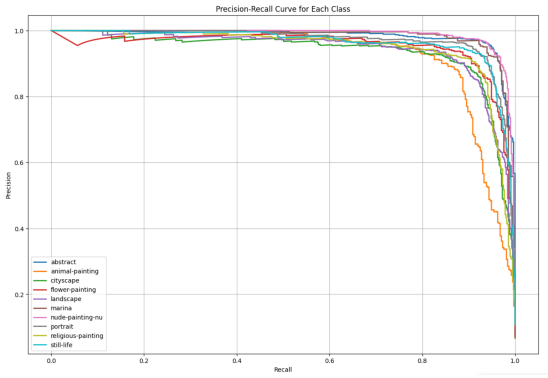
As a next step, we analyze the precision-recall curves of the classifiers that used the Vision Transformer features combined with the hand-engineered features represented in Figure 4.11.



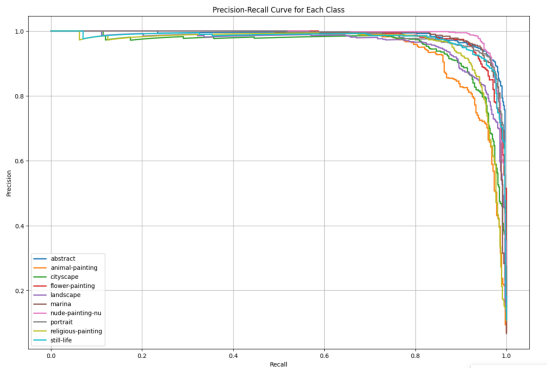
(a) RF ViT (6(CLS))



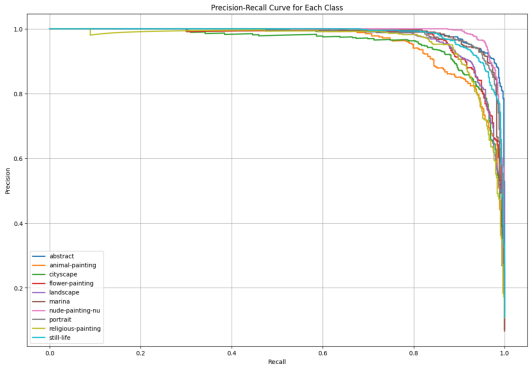
(b) KNN ViT (6(CLS))



(c) MLP ViT (6(CLS))



(d) SVM ViT (6(CLS))



(e) XGBoost ViT (6(CLS))

Figure 4.11 Precision-Recall curves for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.

The curves shown in Figure 4.11 are smooth and positioned higher than those in Figure 4.10, indicating that the classifiers demonstrated better performance and stability across all classes. The same was observed in style classification: the curves of classifiers that used Vision Transformer features were higher than the curves of classifiers that used CNN features. The lowest curves corresponded to the animal painting class, as it was in the previous part of the experiment, where we compared classifiers that used CNN-extracted features. However, the best-performing class, according to the curves, is nude painting, while the abstract class curve is low across all classifiers. Both XGBoost and SVM

achieved the highest curves, even for the animal painting class, which demonstrated lower performance compared to other classes. For MLP and SVM, we can observe drops in the beginning, but they are not significant.

4.2.3 Style Classification using Cropped Facial Regions

In this part of the experiment, we apply the classifiers found in the style classification experiment to the data, where each style is represented by faces extracted from the paintings. Full paintings contain more information, including background details, composition, colors, and brushstroke patterns. While faces keep texture and a limited number of colors, cropped images include only part of the paintings, therefore leading to the loss of many stylistic elements. In contrast, considering only facial regions can help reduce the number of unrelated elements, such as complex backgrounds, objects, or varying compositions, that may distract the model. Faces are typically placed in a consistent way across paintings, as a result, the model gets standardized inputs. Table 4.7 represents weighed F1 scores for selected classifiers tested using stratified 5-fold cross-validation.

Classifier	**Model (Layer)**	**k-fold F1**
RF	ResNet18 (2)	87.3 ± 1
	ViT (6(CLS))	89.5 ± 0.07
SVM	ResNet18 (3)	87.9 ± 0.6
	ViT (6(CLS))	91.5 ± 0.4
KNN	ResNet18 (2)	85.6 ± 1
	ViT (6(CLS))	88.6 ± 0.3
MLP	DenseNet121 (3)	86.3 ± 1
	ViT (6(CLS))	90.9 ± 0.4
XGBoost	ResNet18 (2)	88.9 ± 0.7
	ViT (6(CLS))	91.2 ± 0.4

Table 4.7 Wighed F1 scores from one run and k-fold cross-validation. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting

Table 4.7 shows that for CNN-based features, the best-performing classifier is XGBoost ResNet18(2). The next are RF ResNet18(2) and SVM ResNet18(3). While for ViT-based features, the highest F1 score classifier is SVM, as it was in previous experiments. In addition, we can observe that, as was the case in previous experiments, classifiers that used ViT-based features overperform classifiers that used CNN-based features. But in this experiment, the difference is not as big as it was in previous experiments. While ViT is more focused on global relations because of the attention mechanism, CNNs are good at capturing local features through their convolutional filters. The ability of ViT modeling global dependencies is limited for this task, while the background and composition are absent, and the most representative features in this case are texture and brushwork. All classifiers showed a high overall F1 score. In this and previous experiments for evaluation, we use a weighted F1 score to address the class imbalance problem. However, in this case, we have a greater class imbalance compared to previous experiments. As a result, larger classes can increase the overall weighted F1 score, masking the poor performance of smaller classes despite weighting. We evaluate classifiers by performing one run and

calculating confusion matrices and precision-recall curves. Figure 4.12 represents confusion matrices for classifiers that used CNN-based features.

In Figure 4.12 we can observe that for the smallest class, Pointillism, the best performance was shown by MLP DenseNet121(3) 82%. While the next performing XGBoost showed 77% of correctly classified. As in the experiment for art classification using full paintings, this class had the biggest confusion with Impressionism. For KNN and RF (both showed 9.1%) there is a confusion of Pointillism and Baroque, while MLP and SVM have no confusion for this class at all.

Another small class is Fauvism, and in this case, the best performance for this class showed SVM (86%). As it was in style classification, the biggest amount of confusion was shown with Impressionism. In addition, KNN shows confusion between Fauvism and Baroque (9.1%)

Ukiyo-e is also a class represented by a low number of examples. However, MLP presented a high performance for this class (93%), showing low confusion with other classes. The next is KNN (90%). The best-performing XGBoost and SVM achieved 87% correct classification. Some classifiers showed confusion with Baroque, while KNN and MLP had no confusion between these classes.

Baroque is the most represented class in our dataset. We can observe that the highest percentage of correctly classified examples for this class showed Random Forest ResNet18(2) (95%), and this is the highest number achieved among all classes and classifiers. The next is XGBoost, which showed 93% of correct examples. Although many previously described classes showed high confusion with Baroque, Baroque is not confused with these classes. This can be explained by the fact that Baroque has enough examples, giving the classifiers the opportunity to train.

As in the style classification experiment, Rococo showed a high confusion with Baroque. While Baroque is not often confused with Rococo, the maximum value showed SVM (5.3%) and the minimum was by Random Forest ResNet18 (0.72%)

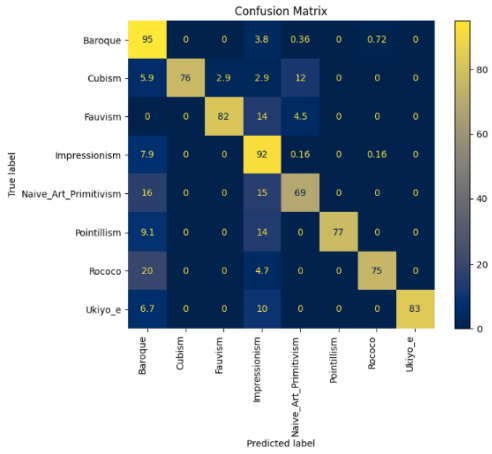
Figure 4.13 represents confusion matrices of classifiers that used ViT-based features. We can see that even though Ukiyo-e is also one of the smallest classes, classifiers that used ViT-based features showed correct classification in 100%, the only exception is Random Forest, which still showed high performance, 95%

However, CNN-based classifiers often confused Fauvism with Impressionism. ViT-based classifiers, such as MLP, XGBoost, and KNN, showed the highest confusion with Primitive art.

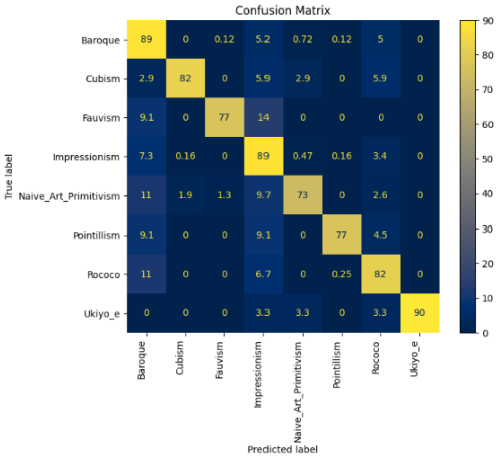
As in the case of ViT-based classifiers, Pointillism is confused with Baroque by KNN, RF, and XGBoost (all have 13%). In contrast, MLP and SVM have no confusion for this class. In addition, Pointillism shows a high number of confusion with Impressionism, the maximum being 20%, shown by Random Forest, SVM, and XGBoost.

KNN that used CNN-based features showed better performance for Cubism (82%) than KNN that used ViT features (72%). All classifiers that used CNN-based features showed a higher percentage of correct classifier examples for the Pointillism class: CNN-based classifier showed 77% of correct classification, the only exception was Random Forest 82%, while ViT-based MLP, XGBoost, and Random Forest showed 67%, MLP and KNN - 73%, which is lower comparing to classifiers that used CNN based features. This is explained by the fact that Pointillism is composed of small, distinct dots of color, and convolution filters are powerful in the recognition of such structures, while ViT, due to its focus on overall structure, which in this case is not very representative, can overlook these details.

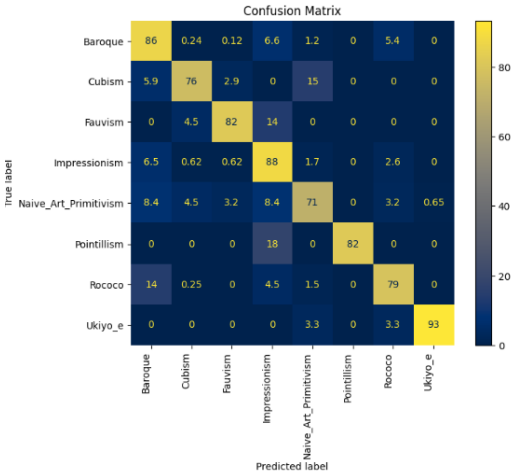
Figures 4.14 and 4.15 represent Precision Recall curves for classifiers that used CNN-



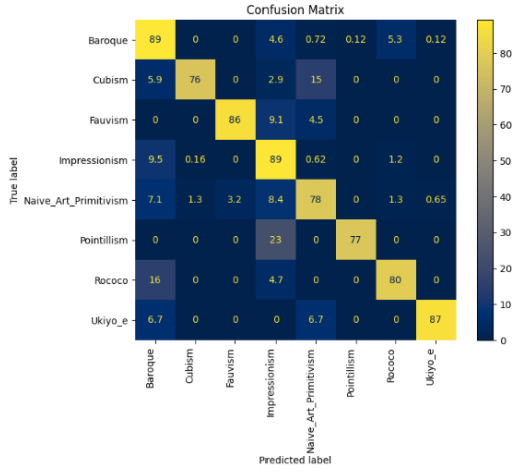
(a) RF ResNet18 (2)



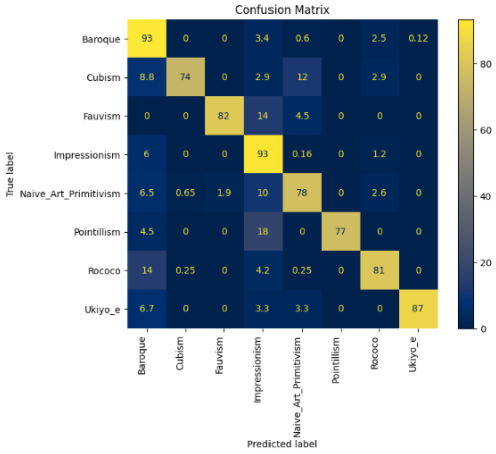
(b) KNN ResNet18 (2)



(c) MLP DenseNet121 (3)

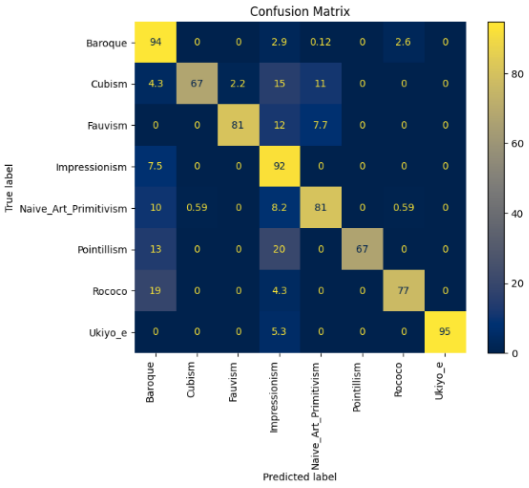


(d) SVM ResNet18 (3)

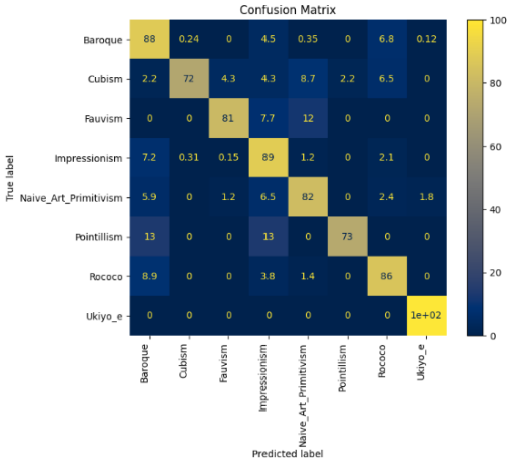


(e) XGBoost ResNet18 (2)

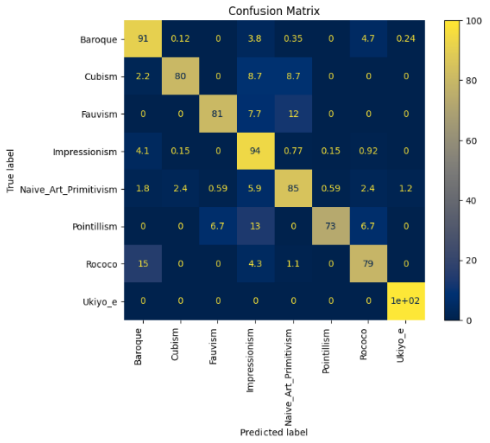
Figure 4.12 Confusion matrices for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.



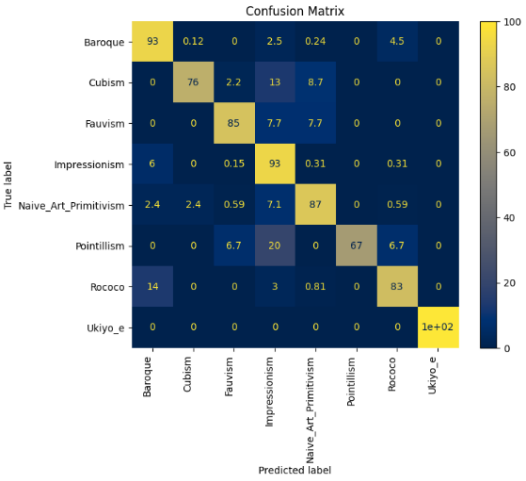
(a) RF ViT (6(CLS))



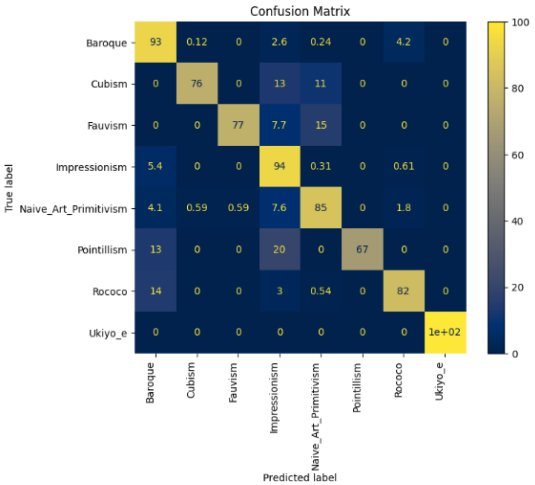
(b) KNN ViT (6(CLS))



(c) MLP ViT (6(CLS))



(d) SVM ViT (6(CLS))



(e) XGBoost ViT (6(CLS))

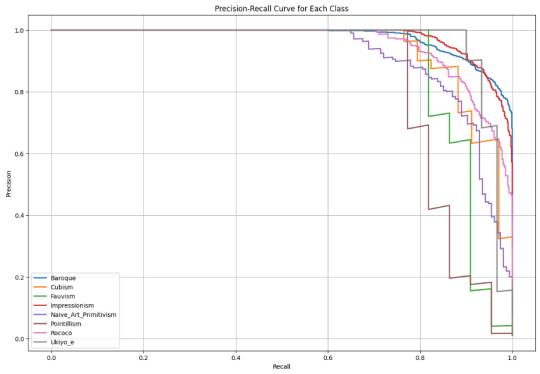
Figure 4.13 Confusion matrices for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.

based features and ViT-based features.

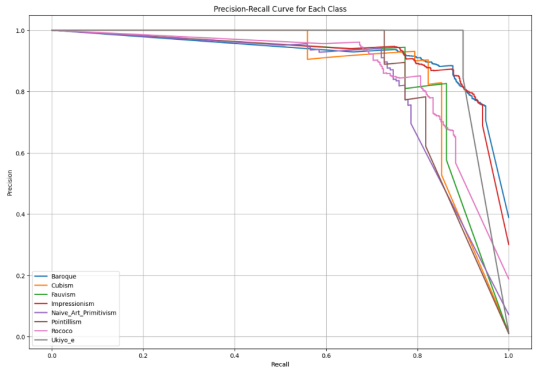
According to Precision-Recall curves, both classifiers using CNN-based features and ViT-based features show steep, sharp curves for Pointillism, Cubism, and Fauvism. Moreover, these three classes have a low number of examples in our data. The curve's form indicates that classifiers are not confident in their predictions and are highly sensitive to threshold changes due to the limited number of positive examples. At the same time, classes represented in many examples have smoother curves across all classifiers. Naive art has a relatively average number of examples, but its curves are also smooth. For ViT-based classifiers, their curve is located higher compared with CNN-based classifiers, which indicates a higher ability to detect positive examples.

For ViT-based classifiers, Ukiyo-e has perfectly located curves among all classifiers except MLP and KNN, but even for these classifiers, the curve is close to the ideal location, except they have one gap.

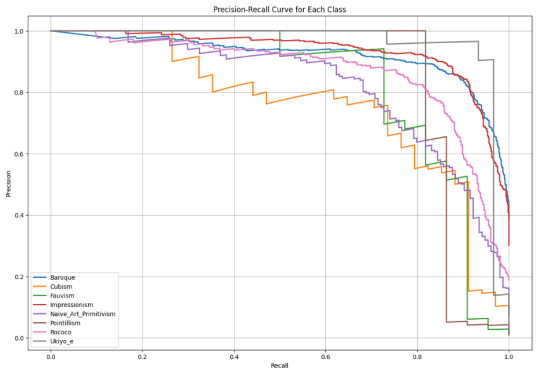
SVM ResNet18(3) showed a huge gap at the beginning for Rococo style. This indicates that the classifier at the beginning makes many incorrect positive predictions for Rococo, which causes the precision to be very low at low thresholds. And further, after increasing the threshold, does the model start to correctly classify true positives, and as a result, precision improves.



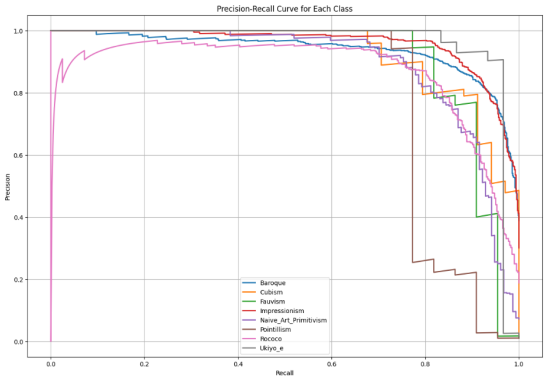
(a) RF ResNet18(2)



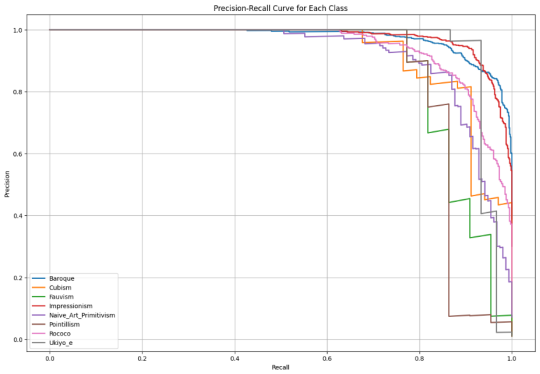
(b) KNN ResNet18(2)



(c) MLP DenseNet121 (3)

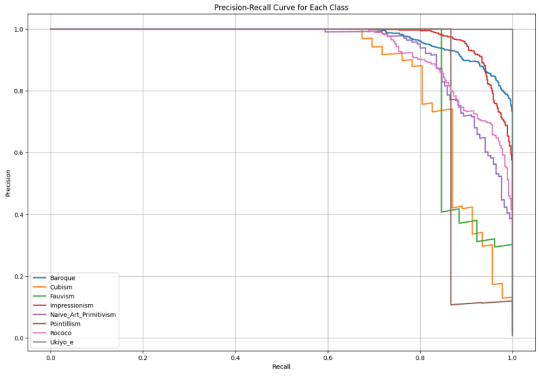


(d) SVM ResNet18(3)

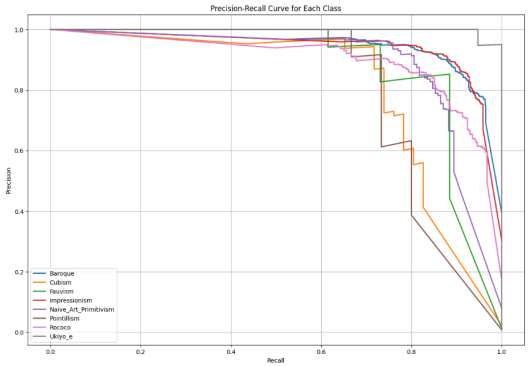


(e) XGBoost ResNet18(2)

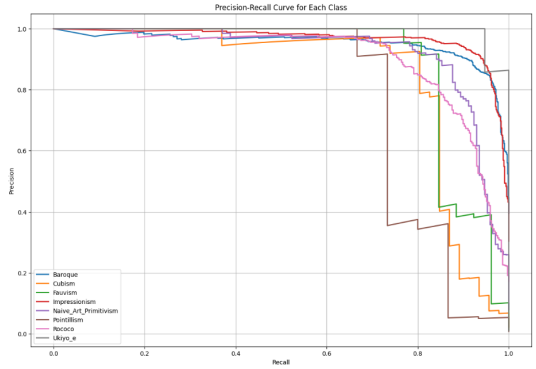
Figure 4.14 Precision-Recall curves for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.



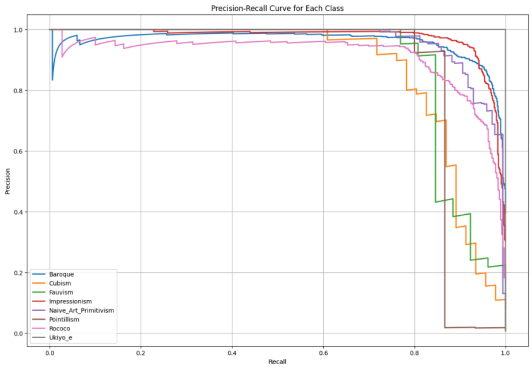
(a) RF ViT (6(CLS))



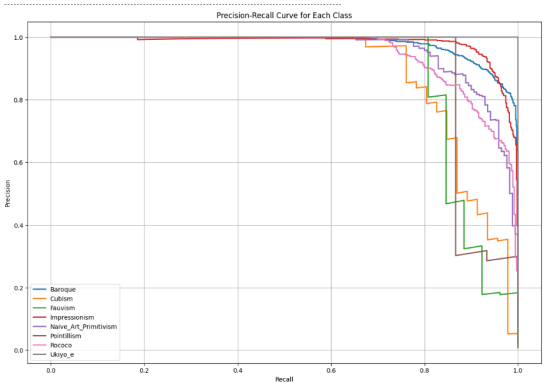
(b) KNN ViT (6(CLS))



(c) MLP ViT (6(CLS))



(d) SVM ViT (6(CLS))



(e) XGBoost ViT (6(CLS))

Figure 4.15 Precision-Recall curves for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting.

Conclusion

This thesis presented a comprehensive analysis of various machine learning techniques for the art classification task. Our goal was to evaluate the performance using the Wiki Art dataset, which is one of the biggest collections of digitalized art. Firstly, we analyze selected styles and genres and their visual characteristics that are important for automated classification; we explore their differences and similarities to understand where methods can have difficulties in classification.

It was demonstrated through experimentation that deeper and more innovative models extract more diverse features, showing better performance compared to shallow models with compact architecture. We showed that among the selected pre-trained CNNs, EfficientNetB1, DenseNet121, and ResNet50 achieved better F1 scores due of their ability to capture complex and diverse features. Although MobileNetV3Large is lighter and shallower, it also showed the same high F1 score. Although CNNs demonstrated strong performance, the Vision Transformer (ViT) outperformed all CNN-based models, showing the highest F1 score in this experiment.

In addition, we focus on deep learning-based feature extraction combined with classical machine learning techniques. We use a deep learning method as a feature extractor, combine it with handcrafted features, and pass the resulting vector to machine learning classifiers. We demonstrated that ViT-extracted features combined with handcrafted features create well-separated dense clusters, while clusters are more spread for CNN-based features. We found that for CNN-based features, the best performing classifiers were tree-based ensemble techniques such as XGBoost and Random Forest. While with ViT-based features, the best F1 score was shown by SVM. All classifiers achieved the best performance on features that included features from intermediate layers of the CNN. The majority of classifiers showed the best performance for features of ResNet18(2), ResNet18(3) and DenseNet121(3). Through experiments, we showed that intermediate features can be more representative of our task. In addition, this and previous experiments showed that feature reuse is beneficial for art classification tasks, while it allows the preservation of low-level features that are significant in style classification. CNNs that utilize feature reuse showed high F1 scores in both experiments. Through experimentation, we demonstrate that classifiers that use ViT-based features outperform classifiers that use CNN-based features.

We applied the best-performing feature strategies and machine learning classifiers from art styles classification experiment to the genre classification task. We demonstrate that the selected methods show promising performance despite the various differences between tasks. We showed that our methods are effective for genre classification, and with the necessary parameter search and layer selection, stronger performance can be achieved.

We demonstrated that selected feature strategies and machine learning classifiers are effective for style classification, where cropped facial regions represent paintings. We demonstrated that, in general, classifiers using VIT-extracted features outperformed those using CNN-extracted features, as was the case in previous experiments. But for this task, there are classes where classifiers that used CNN-based features performed better.

Transfer learning, described in the first experiment, and feature extraction approach, described in further experiments, are completely different paradigms, and they are not directly comparable. Through experimentation, we demonstrate that the combined feature extraction approach can show competitive results.

As the next step in this research, we can explore feature extraction from different layers of the ViT to identify which features are most representative for all described tasks: the style and genre classification, and style classification using cropped facial regions. Additionally, we can experiment with the CNNs selected for the style classification task and their layers for genre classification and style classification using cropped facial regions to investigate which features are the most representative for these tasks. After this, we can perform a parameter search for machine learning classifiers. Furthermore, we can experiment with other architectures and computer vision algorithms for feature extraction and evaluate their performance. Additionally, we can experiment with various data augmentation techniques to create a more diverse dataset. Moreover, we can apply the investigated methods to the classification of other art directions and styles, for example, Chinese or Indian art.

Bibliography

[Aga+15] Siddharth Agarwal et al. "Genre and Style Based Painting Classifications". In: *2015 IEEE Winter Conference on Applications of Computer Vision* 1.1 (2015), pp. 588–594.

[Dos+21] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[GCM18] Eren Gultepea, Thomas Edward. Conturob, and Masoud Makrehchia. "Predicting and grouping digitized paintings by style using unsupervised feature learning". In: *Journal of Cultural Heritage* 31 (2018), pp. 13–23.

[He+15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[How+17] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: https://arxiv.org/abs/1704.04861.

[How+19] Andrew Howard et al. *Searching for MobileNetV3*. 2019. arXiv: 1905.02244 [cs.CV]. URL: https://arxiv.org/abs/1905.02244.

[Hua+18] Gao Huang et al. *Densely Connected Convolutional Networks*. 2018. arXiv: 1608.06993 [cs.CV]. URL: https://arxiv.org/abs/1608.06993.

[Imr+23] Saqib Imran et al. "Artistic Style Recognition: Combining Deep and Shallow Neural Networks for Painting Classification". In: *Mathematics* 11.22 (2023). ISSN: 2227-7390. DOI: 10.3390/math11224564. URL: https://www.mdpi.com/2227-7390/11/22/4564.

[Nun+18] Ivan Nunez-Garcia et al. "Classification of Paintings by Artistic Genre Integrating Color and Texture Descriptors". In: *In International Conference on Artificial Intelligence and Pattern Recognition* (2018), pp. 66–70.

[Nun+22] Ivan Nunez-Garcia et al. "Classification of Paintings by Artistic Style Using Color and Texture Features". In: *Computación y Sistemas* 26.4 (2022), pp. 1503–1514.

[OPH94] T. Ojala, M. Pietikainen, and D. Harwood. "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions". In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 1. 1994, 582–585 vol.1. DOI: 10.1109/ICPR.1994.576366.

[San+19] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV]. URL: https://arxiv.org/abs/1801.04381.

[SHB24] Zeinab Sedaghatjoo, Hossein Hosseinzadeh, and Bahram Sadeghi Bigham. *Local Binary Pattern(LBP) Optimization for Feature Extraction*. 2024. arXiv: 2407.18665 [cs.CV]. URL: https://arxiv.org/abs/2407.18665.

[SPP23] Ludovica Schaerf, Eric Postma, and Carina Popovici. "Art authentication with vision transformers". In: *Neural Computing and Applications* 36 (Aug. 2023), pp. 1–10. DOI: 10.1007/s00521-023-08864-8.

[SRM24] Mojtaba Shahi, Roozbeh Rajabi, and Farnaz Masoumzadeh. “CNN-Based Classification of Persian Miniature Paintings from Five Renowned Schools”. In: (2024). arXiv: 2411.10330 [cs.CV]. URL: https://arxiv.org/abs/2411.10330.

[TL19] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *CoRR* abs/1905.11946 (2019). URL: http://arxiv.org/abs/1905.11946.

[VS17] Nitin Viswanathan and Stanford. “Artist Identification with Convolutional Neural Networks”. In: (2017). URL: https://api.semanticscholar.org/CorpusID:198974841.

[Wan+24] Junyan Wang et al. “RepCFT: A Fusion Model of Reparameterized Convolutional Neural Network and Visual Transformer for Wood Species Recognition”. In: (May 2024), pp. 343–349. DOI: 10.1145/3652628.3652685.

List of Figures

2.1 VGG-19, a plain network with 34 parameter layers, a residual network with 34 parameter layers. Source:[He+15] 15
2.2 Block Mobilenet V2. Source: [How+17] 16
2.3 A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input. Source:[Hua+18] 18
2.4 Vision Transformer architecture. Source: [Dos+21] 19
2.5 LBP feature extraction process using LBP. Source: [SHB24] 20
2.6 Example of painting and its color palette. 20

3.1 Examples of images and their styles 21
3.2 Distribution of images according to their styles after downsampling . . . 23
3.3 Examples of images and their genres 23
3.4 Distribution of images according to their genres after downsampling . . . 24
3.5 Examples of images and their styles 25
3.6 Distribution of images according to their styles. Classification based on style using Cropped Facial Regions. 25

4.1 Feature vector where A - features extracted using deep learning model size is represented in Table 4.3, B - LBP, size is 36, C - color palette size is 45, D - statistical features, size is 18 30
4.2 Extracted features for Style Classification.The caption of the picture represents the model name and the layer from which features were extracted. . 31
4.3 Confusion matrices for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 36
4.4 Confusion matrices for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 37
4.5 Precision-Recall curves for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 39
4.6 Precision-Recall curves for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM -Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 40
4.7 Features extracted from the selected layers of models. Genre Classification. 41

4.8 Confusion matrices for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 43
4.9 Confusion matrices for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 45
4.10 Precision-Recall curves for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 47
4.11 Precision-Recall curves for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 49
4.12 Confusion matrices for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 52
4.13 Confusion matrices for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 53
4.14 Precision-Recall curves for the classifiers that used CNN features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 55
4.15 Precision-Recall curves for the classifiers that used Vision Transformer features combined with hand-engineering features. The caption of the picture represents the model name and the layer from which features were extracted. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting. 56

List of Tables

4.1 Classifier architecture 26
4.2 Pretrained CNNs for the style classification task. 27
4.3 The sizes of feature vectors. Models: A - ResNet18. B - MobileNetV3Large, C - DenseNet121, D - Vision Transfor. 30
4.4 F1 score of classifiers applied to different features. Models: A - ResNet18. B - MobileNetV3Large, C - DenseNet121, D - Vision Transfor. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting 32
4.5 F1 scores from one run and k-fold cross-validation. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting 34
4.6 F1 scores from one run and k-fold cross-validation. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting 42
4.7 Wighed F1 scores from one run and k-fold cross-validation. Classifiers: RF - Random Forest, SVM - Support Vector Machines, MLP - Multilayer Perceptron, XGBoost - eXtreme Gradient Boosting 50