LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHR- UND FORSCHUNGSEINHEIT FÜR PROGRAMMIER- UND MODELLIERUNGSSPRACHEN
Institut für Informatik

Style Recognition in Paintings using Deep Learning

Luis Wankmüller

Bachelorarbeit

Beginn der Arbeit:	06.02.2017
Abgabe der Arbeit:	04.05.2017
Betreuer:	Prof. Dr. François Bry
	Clemens Schefels

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

München, den 5. Mai 2017 Luis Wankmüller

Zusammenfassung

In jüngster Zeit hat maschinelles Lernen und insbesondere Deep Learning viel Aufmerksamkeit erregt und signifikante Fortschritte erzielt. Diese Arbeit konzentriert sich auf dessen praktischen Anwendungen. Ziel ist es Bilder automatisch in ihre zugehörigen Stilrichtungen einzuordnen. Im ersten Teil werden mehre Datensätze von Bildern und ihren Stilrichtungen erstellt. Dies geschieht mithilfe der Webseite ARTigo, auf der Nutzer kollaborativ Bilder mit Schlagworten versehen. Es werden zwei verschiedene Ansätze zur automatischen Stilerkennung verglichen, in denen Algorithmen auf die verschiedenen Datensätze trainiert werden. Der erste Ansatz verwendet ein Neuronales Netz, um Informationen aus den Datensätzen zu extrahieren. Mithilfe dieser Informationen wird dann ein traditioneller Algorithmus des Maschinellen Lernen trainiert. Der zweite Ansatz verwendet nur Deep Learning und trainiert ein Neuronales Netz um, das zuvor auf Objekterkennung trainiert wurde. Die Methode mit den besten Ergebnissen wird dann verwendet, um Bilder der ARTigo Plattform mit zusätzlichen Schlagwörtern zu versehen.

Abstract

Machine learning and especially deep learning, has gained a lot of attention in the last few years and made significant progress. This thesis focuses on a practical application of deep learning. The goal is to automatically classify images depending on their styles. In its first parts, multiple datasets, consisting of images labeled with their style are constructed. The tags collected on the ARTigo social image tagging platform are used for this task. Two different approaches to automatically detect image styles are evaluated by training various machine learning algorithms on the different datasets. The first approach used a Neural Network for feature extraction and then uses traditional machine learning methods to train a detection algorithm. The second one only uses Deep Learning and retrains a Neural Network that has been trained on the task of object recognition before. The best performing classification method is then used to automatically tag pictures for the ARTigo database.

Acknowledgements

I want to thank Prof. Dr. François Bry and Dr. Clemens Schefels for constantly providing support and feedback. I also want to thank all the people who put so much work into providing machine learning tutorials and publishing them on the internet. Finally, a thank you to everyone who supported me and reassured me when I had doubts.

Content

1. Introduction **7**

2. Related Work **8**
2.1. Feature Extraction . . . 8
2.2. Fine-tuning Pretrained Artificial Neural Networks . . . 9

3. Theoretical Framework **10**
3.1. Supervised Machine Learning . . . 10
3.1.1. Gradient Descent . . . 10
3.1.2. Loss Function . . . 11
3.1.3. Regularisation . . . 12
3.1.4. Hyperparameters and Cross-Validation . . . 12
3.2. Artificial Neural Networks . . . 12
3.2.1. Convolutional Neural Networks . . . 14
3.2.2. Backpropagation . . . 16

4. Conception **18**
4.1. Datasets . . . 18
4.2. Fine-tuning an existing CNN . . . 20
4.3. Using a CNN for feature extraction . . . 23

5. Implementation **24**
5.1. Training Tools . . . 24
5.2. Fine-tuning . . . 25
5.3. Feature Extraction and Shallow Learning . . . 26
5.4. Novelty Detection and Practical Application of the Classifiers . . . 29
5.5. Evaluation . . . 30
5.5.1. Scoring Metrics . . . 30
5.5.2. Shallow Learning with Support Vector Machines . . . 31
5.5.3. Fine-tuning the Inception Network . . . 34
5.5.4. Performance . . . 36
5.5.5. Classification Examples . . . 37
5.5.6. Comparison with Other Works . . . 38

6. Summary and Outlook **43**
6.1. Summary . . . 43
6.2. Outlook . . . 43

A. Appendix **45**
A.1. Style- and Similarity Tags . . . 45
A.2. Class Distribution for the Datasets . . . 47
A.3. Scoring Metrics for the Datasets . . . 51
A.4. Misclassified Images For The *wikiartSel* Dataset . . . 54

Literature **59**

1. Introduction

Machine learning has produced many practical applications in recent years. There are systems that surpass us at speech recognition[CJLV16], image processing and lip reading[ASWdF16]. They are even able to compose music[KSL15] or generate art[GEB15]. There is little surprise that artificial intelligence is mentioned by the media on a regular bases. So much in fact, that it has become a buzzword, that is used for nearly almost everything. Even toothbrushes now contain "proprietary artificial intelligence technology"[1]. On the other hand, there are voices, scared that artificial intelligence might take away jobs or hurt them in any other way. Just a few years back in 2014, the company SwiftKey unveiled a new communications system for Steven Hawkings. At the same day, potentially impressed by the new system, he gave an interview to the BBC, claiming that "the development of full artificial intelligence could spell the end of the human race"[BBC14].

In this thesis, the peaceful applications of machine learning will be demonstrated on the ARTigo[2] project. The ARTigo project aims to produce descriptions for visual artworks. It is not an easy task for most people to predict the style of an image, however, the people working together on the ARTigo project have already tagged more then 4,000 images with their respective style names to their best efforts. With the recent attention and improvements around machine learning one might think, that this work can be replaced or assisted by the use of a machine learning algorithm. This thesis will start by giving an overview about related work that has been done in the task of style recognition (Section 2). Continuing it will give an overview about the basics of machine learning and neural networks (Section 3). Then, it will describe, how the ARTigo database is used to collect images for each of the available style classes (Section 4.1). Next, the thesis will focus one a practical application of machine learning. Following the attention deep learning has received in the last years it will incorporate some of the new techniques that came with it. As an alternative to handcrafted methods[3] (engineered by humans), a trained neural network is used to extract relevant information on the extracted data (relevant features are automatically selected). Then, a classical algorithm is trained on this data (Section 4.3). As the second approach, a neural network will be retrained on the raw images (Section 4.2). Finally, the two approaches are evaluated (Section 5.5.1). The best performing approach is then used to create a program that is able automatically create style tags for new images. In the final section the work done is summarised and an outlook is given. It includes ideas for improvement and lessons learned (Section 6).

[1] Ara, the first toothbrush with Artificial Intelligence. `https://www.kolibree.com/en/ara/`

[2] `http://www.artigo.org/`

[3] e.g. Scale-invariant feature transform

2. Related Work

This section will give an overview about the related works with focus on the used techniques. A more in-depth explanation about the discussed methods and algorithms will follow in the theoretical overview in Section 3. There some existing work focusing on the task of style recognition. Since there is little training data available, none of them train an Artificial Neural Network (ANN) from scratch but instead use a pre-trained model. Some of them use *fine-tuning* of an existing ANN, some use an ANN for feature extraction and others do not use a ANN at all.

2.1. Feature Extraction

One of the first works on recognising image style is "Recognizing Image Syle" by Karayev et. al. [KTH+13]. Published in 2013, their work is still very relevant and one of the most referenced papers on the subject [Sch17].

In their work, they create two datasets focused on image styles. The first one contains about 80,000 images downloaded from *Flickr*[4]. Each class consists of around 4,000 features and is labeled with one of 20 visual styles[5]. The other dataset is assembled using the *Wikiart*[6] collection of labeled images. It consists of 25 styles[7], with at least 1,000 features per class, for a total of 85,000 images. As the second dataset is closer to the problem considered in this thesis, the remainder will focus on it.

Karayev et al. use linear classifiers, relying on sophisticated features rather then sophisticated learning algorithms. This means that most of the work is put into extracting the features and training a somewhat simple classifier on the extracted features. Stochastic gradient descent with an initial learning rate of 0.5 is used to minimise a custom training function. When evaluating the different features the data was split in 20% for testing, 20% for validation and 60% as training set. The performance is measured using average precision evaluation(see Section 5.5.1).

A performance of 0.356 is scored using only features extracted from the sixth layer of Caffe ($DeCAF_6$). The best performance of 0.473 is achieved by a combination of multiple classifiers and $DeCAF_6$, called *Fusion x Content*. This feature is generated by first using $DeCAF_6$ on the PASCAL VOC dataset (containing object classes) and training a classifier on the 20 classes it provides. The classes are then aggregated to train four new classifiers for bigger overshadowing classes ("animals", "vehicles", "indoor objects" and "people"). The outputs (confidences) of these new classifiers is then combined with a feature channel using the outer product. The *Fusion x Content* classifier is then trained on the resulting data. According to Karayev at el., the performance increase confirms a correlation between the content and the style of an image.

The more recent article "Large-scale Classification of Fine-Art Paintings: Learning The Right Metric on the Right Feature" by Saleh and Elgammal [SE15] goes in depth on style

[4] `http://www.flickr.com`
[5] e.g. HDR, Macro, Noir, Vintage, Minimal, Hazy, Long Exposure
[6] `https://www.wikiart.org/`
[7] e.g. impressionism, expressionism, realism

and genre recognition. The used dataset is also downloaded from *Wikiart*[3]. It contains 27 style classes with at least 25,000 features each for a total of around 80,000 images. What makes their approach unique is the use of metrics on-top of extracted features and only then training a linear classifier on the data. The best performing single feature has been extracted using the *Classemes* classifier in combination with *Boost metric learning*, scoring an accuracy of 31.77%. The best performance using a Convolutional Neural Network (CNN) scored only 16.83% accuracy. However, their use of feature extraction using a CNN differs from above examples by using the output of the last layer (which contains object confidences). By using feature fusion, they managed an overall best of 45.97% accuracy.

Siddharth et. al. compare multiple handcrafted feature extraction methods in "Genre and Style based Painting Classification" [AKPP15]. A Support Vector Machine (SVM) using the χ^2 kernel is used for classification. As dataset a subset of the *Wikiart*[3] database was used, consisting of only 300 paintings per style for a total of 3,000 images. It was split 80%, 10% and 10% for training, testing and validation. The best accuracy of 62.37% is measured using 10-fold cross validation and achieved using feature fusion. The best performing single feature is *SIFT* with an accuracy of 59.2%.

The paper "Classification of Artistic Styles using Binarized Features Derived from a Deep Neural Network" by Bar et. al. [BLW14] gives an extensive performance overview over many different methods. Also a method of binarising real numbered features to boost performance is proposed. Methods discussed reach from several low-level descriptors to more sophisticated features like *PiCoDes*. They also discuss features extracted using the *Decaf* CNN. As dataset *Wikiart*[3] with 27 classes and 40,000 features is used. For classification, multiple classifiers including SVM are tested but in the end a k-nearest Neighbors (kNN) classifier was chosen. Three-fold cross validation was used for all results. The overall best performing classifier with an accuracy of 0.43 uses late feature fusion between $PD - 1024$ (PiCoDes with 1024-dimensionality), $PD - 2048$, $Decaf_5$ and $Decaf_6$. The work also compared early-fusion (concatenate descriptors) with late-fusion, in which separate classifiers were trained on each descriptor and then each classifier votes on the most likely class. Generally, late fusion, as used by the top performing classifier, performed much better then early fusion.

2.2. Fine-tuning Pretrained Artificial Neural Networks

Yingui Xia, in the paper "Fine-tuning for Image Style Recognition" [Xia15], is the only publication that could be found to use fine-tuning of a CNN for style recognition. He uses the same *Flickr* dataset mentioned above, splitting it into 80% training set and 20% test set. To speed up the learning progress *batch normalization* was used. Furthermore, some preprocessing was done to boost the classification of grayscale images. The best accuracy of 39.16% is achieved by setting a learning rate of 0.001, using the *Leaky-ReLU* activation function and data augmentation (random cropping and flipping).

3. Theoretical Framework

In the following, we will lay a foundation regarding machine learning to help understand the methods used in the following work. Usually, machine learning literature distinguishes between supervised learning, unsupervised learning and reinforcement learning [RN03]. Our topic, style classification of paintings, is as the name suggests a classification problem. For this reason, in the following, we will focus on supervised learning, in particular classification.

3.1. Supervised Machine Learning

Generally, all supervised machine learning (ML) methods work quite similarly. One trains a so called classifier which is able to predict classes for new data. In practice, the classifier generates a vector with one dimension for each possible class and its binary decision[8]. A subclass of classification is *probabilistic classification* [HTF01a]. Here, the output for each class is not binary, but a confidence[9]. In some cases this provides benefits like avoiding error propagation or more accurate scoring. Both, the currently popular *deep learning* and *shallow learning*[10], build upon a loss function (sometimes called cost function) that has to be minimised. While minimising the parameters of the loss function, the weights and biases are changed accordingly. In the following, if not stated differently, x will denote the input data and y the according class. We will also use the term "deep learning" (big data) for training with Artificial Neural Networks(ANNs) and the term "shallow learning" (not so big data) for the more classical ML approach.

3.1.1. Gradient Descent

Gradient Descent is a method for minimising loss functions. For a loss function $J(\theta)$ we want to choose θ so that $J(\theta)$ gets minimised. Gradient descent does this by first setting an initial θ and then updating the function:

$$\theta_j^{(k+1)} := \theta_j^{(k)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(k)}) \tag{1}$$

This update has to happen for all parameters j simultaneously. The α represents the learning rate or step size of the update. Choosing the right learning rate can be a challenge as a value too small can lead to very slow convergence or getting stuck in a local minima, a value too big might not converge at all by skipping over the minimum. Figure 1 shows a basic version of gradient descent. In practice, this version of gradient descent, known as *batch gradient descent*, is almost never used. It has to calculate the gradient for every training example before it performs **one** update (α) in the direction of the gradient. As an alternative *stochastic gradient descent* was introduced. In contrast to batch gradient descent it performs a parameter update for each training example $x^{(i)}$ and label $y^{(i)}$. This helps θ converge faster. The most used version of gradient descent is the so called *mini-batch*

[8] e.g. for three classes an output could be $\vec{a}^T = (1, 0, 0)$

[9] e.g. for three classes a possible output could be $\vec{a}^T = (0.74, 0.16, 0.1)$

[10] As contrast to deep learning, meaning the classical approaches to ML.

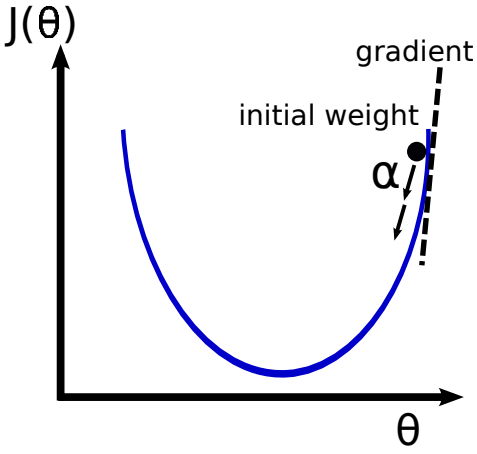

Figure 1: A simplified illustration of gradient descent, with α denoting the step size, θ the parameters and $J(\theta)$ the cost function.

gradient descent. It aims to combine the best of both worlds by splitting the training data in batches and then performing an update for every mini-batch of n training examples. There are many more variants of gradient descent which incorporate ideas like a variable learning rate α. An extensive overview and comparison is available by Sebastian Ruder [Rud16].

3.1.2. Loss Function

The loss function is one of the most important parts of any machine learning algorithm. It represents the accuracy the classifier currently has, while regulating the parameters to prevent too specific representation of the data (*overfitting*). As the gradient descent tries to minimise it, a bad loss function cannot lead to good results. Note that sometimes loss functions are also called score functions or error functions.

As an example for a widely used loss function we will take a look at a Support Vector Machine (SVM), in particular the *Multiclass Support Vector Machine loss*. Its first part, the *hinge-loss* is defined as follows:

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta) \tag{2}$$

It sums up all the incorrect classes and subtracts the correct class score. While y_i denotes the correct class and s_j the score at j. A hyperparameter Δ is added and each sub-term is thresholded to zero. As an example, lets pretend the defined parameters of our classifier generate the scores $\vec{s}^T = (-15, 2, 9)$. The correct class $y_i = 2$ and $\Delta = 5$. Hinge-loss will now calculate $L_i = max(0, -15 - 9 + 5) + max(0, 2 - 9 + 5) = 0$. Basically it aims for the score of the correct class to be larger then the scores for an incorrect class by at least Δ. If this is not the case loss is accumulated. [cs2]

3.1.3. Regularisation

When analysing Equation 2 it becomes clear that the weights (following the scores) are not limited in size and create the same output as smaller weights[11]. Big weights are generally not desired as they lead to *overfitting*. That is where regularisation comes into play. When talking about *Multiclass Support Vector Machine loss* we mentioned its two parts. The first one, hinge-loss, was described in Equation 2. The second one is regularisation. Specifically the *L2* norm, one of the most used regularisation penalties [cs2].

$$R(W) = \sum_k \sum_l W_{k,l}^2 \tag{3}$$

The L2 penalty function takes a weight matrix, iterates over its values, squares them and sums them together. Together with Equation 2 the L2 norms forms the complete *Multiclass Support Vector Machine loss*:

$$L = \frac{1}{N} \sum_i L_i + \alpha R(W) \tag{4}$$

where α is a hyperparameter and N the number of training examples.

Although hinge-loss is one of the most used loss functions for classification there are some other alternatives. L. Rosasco et. al. provide a comparison over other possible candidates [RDC+04].

3.1.4. Hyperparameters and Cross-Validation

Hyperparameters are parameters that cannot be set by training the ML model (like normal parameters) and can not be set universally, so everytime a model is trained one has to find new optimal hyperparameters. It is, however, possible to set them and then evaluate the results. *Cross-validation* is used to measure performance for smaller datasets and in combination with an algorithm like *grid search*[BB12], can be used to find the right hyperparameters[12]. When using grid search, a set of possible hyperparameters is supplied. For each set of hyperparameters a model is trained and its performance is evaluated using cross-validation. Other alternatives to grid search are *random search* [BB12] and *Bayesian optimization* [SLA12].

K-fold cross-validation, one of the most used versions of cross-validation works by splitting the available data into k equal parts. Next the algorithm iterates k times though the data. With every iteration, one of the k parts of data is chosen as testing set and the rest of the data as training set. The final prediction error is the average on an all k folds [HTF01b].

3.2. Artificial Neural Networks

Artificial Neural Networks (ANN or just NN) are networks that are loosely modelled after the human brain. There are many different classes and types of ANNs[13]. However our appli-

[11] *e.g.*, $\vec{s}^T = (-30, 4, 18)$ results in the same hinge loss as the example above

[12] Generally this is only applicable for shallow learning. Deep learning usually does not use cross validation as it requires a lot of data anyway.

[13] e.g., recurrent neural networks, probabilistic neural networks, self-organizing maps

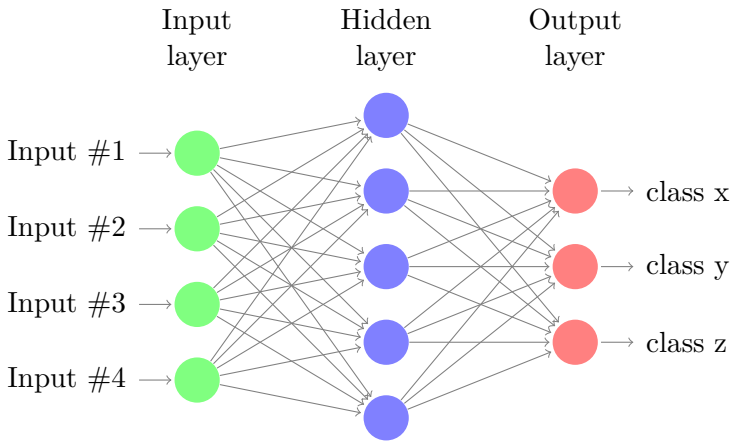

Figure 2: An example for a small neural network. It accepts four inputs, has one hidden layer and three possible class outputs.

cation of ANNs uses *feedforward neural networks* which are generally used for classification. For this reason from now on when talking about ANNs we are referring to feedforward neural networks.

The processing units inside an ANN are called *neurons*. Just like in most other ML algorithms, there is a set of weights which act as parameter to the algorithm. In ANNs, each connection between neurons has a weight assigned that affects the data flowing through the neurons. ANNs are organised in layers. The first layer, called the input layer, has as many neurons as inputs. After the input layer, n *hidden layers* with a variable amount of neurons follow. The last layer, called the *output layer*, maps the data to the classes the ANN was trained on. All layers are fully connected, meaning that each neuron of a layer is connected to all neurons of the next layer. Figure 2 shows a small ANN with one hidden layer four inputs and three possible output classes.

The calculations that lead to the final output all happen inside the neurons. Each neuron has a set of functions to handle the input and generate an output. Firstly, the *propagation function* which propagates the various inputs of a neuron. Secondly, the *activation function* which activates the neuron if a certain condition (often simply a threshold) based on the inputs is met. Lastly, the *output function* which transforms the activation into an outputs for other neurons[14] [Kri07]. Most of the time each neuron has a *bias* which is taken into account when calculating the activation function. The bias is an additional parameter of each neuron (in contrast to the other parameters, which are outputs from other neurons). Mathematically a bias shifts the activation function in positive or negative direction along the x-axis. There are many different functions which are suitable as an activation function[15] and often more than one activation function is used for a ANN.

A wildly used example for an activation function is the *ReLU* function (Rectified linear

[14] Often, the output function is just the identity function, meaning the output of the activation function gets passed along.

[15] Desirable properties among others are: nonlinearity and continuous differentiability.

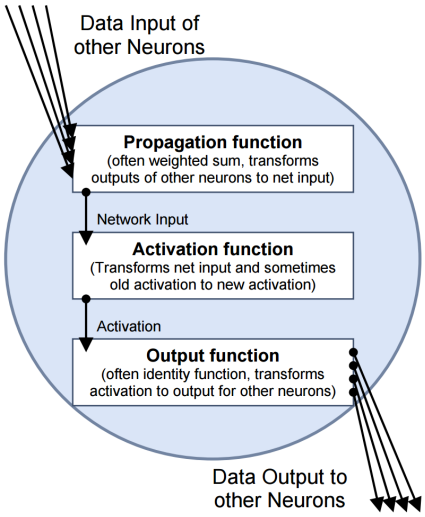

Figure 3: A neurons cross section, showing the different functions inside [Kri07, page 35]

unit). Equation 5 shows the ReLu function.

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \tag{5}$$

3.2.1. Convolutional Neural Networks

When working with large input data like pictures, using an ANN becomes a lot harder. M. Nielsen started his book by training an ANN on the MNIST Dataset[16] [Nie, Chapter 1] with some success. But even on the black and white MNIST images with only 28×28 Pixels the results where not very good. A bigger picture with for example $400 \times 400 \times 3$ (three for RGB) would require 480,000 input neurons. Furthermore one neuron in the first (fully connected) hidden layer would require 480,000 weights. It becomes clear that it is not easy to train such a network.

For this reason, it is common to use a Convolutional Neural Network (CNN) for image processing. A CNN works similarly to an ANN, it has a loss function and the last layer is still a fully connected layer of neurons which output the classification classes. The difference lays in the other layers. Generally, there are three kinds of layers: *Convolutional Layers*, *Pooling Layers* and, already mentioned, *Fully-Connected Layer*. [cs2]

Each of the convolutional layers has a set of filters. One pictures each of the layers as a three dimensional cuboid with a height, a width and a depth. A filter has a fixed, small size[17] when looking at its width and height, but has the full depth of the input volume (three for the color channels in the first layer). While passing the data through the layer we slide (convolve) this filter across the width and height of the input volume and calculate the dot product of the filter weight at the values at the current filter position. We mentioned

[16] `http://yann.lecun.com/exdb/mnist/`
[17] In the first layer the size of the filter refers to the pixel of the input image

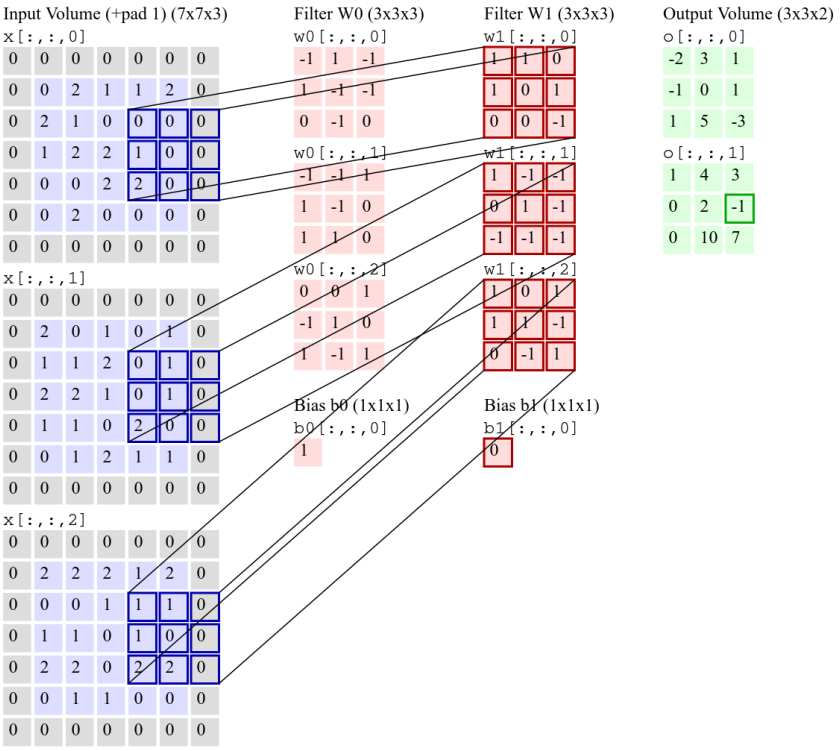

Figure 4: Two filters (i.e. depth of two) processing an input volume with a padding of one. A step size of two is used. The result (green) is offset by the bias. Each row represents one depth dimension. The calculation performed in the current step is $(1) + (-1 + 1 - 2) + (1 + 1 - 2) + 0 = -1$. [cs2, animation about convolutional networks]

before that fully connected layers are not practical because they create to many parameters. Instead, each neuron is only connected to a local region of the input volume. The size of this region is called *receptive field size*. As an example suppose an input volume of size $20 \times 20 \times 3$ and a receptive field size of 5×5. Every filter would now have a size of $5 \times 5 \times 3$. Note that the receptive field size does not have an effect on the depth. This filter is then shifted over the input values, calculating the dot product. Together with a bias, the result gets passed though the activation function (often *ReLu* like in ANNs). Figure 4 shows the process of convolution without the activation function. A convolutional layer also has a set of hyperparameters: the size of the steps the filter makes (stride), the receptive field size F and if a padding is added to the borders of the field[18]. It is possible to stack multiple filters on top of the same region. How many filters are used is also a hyperparameter, called *depth*.

Through backpropagation (see Section 3.2.2) each filter is trained to recognise a specific attribute of the image (e.g., edges). When Figure 5 shows what effect a convolution filter can have on an image.

Pooling layers do not have any parameters to be trained. They remove information that is

[18]e.g., for a stride of one it is common to use zero padding (padding with zeros) with a size of $P = (F-1)/2$.

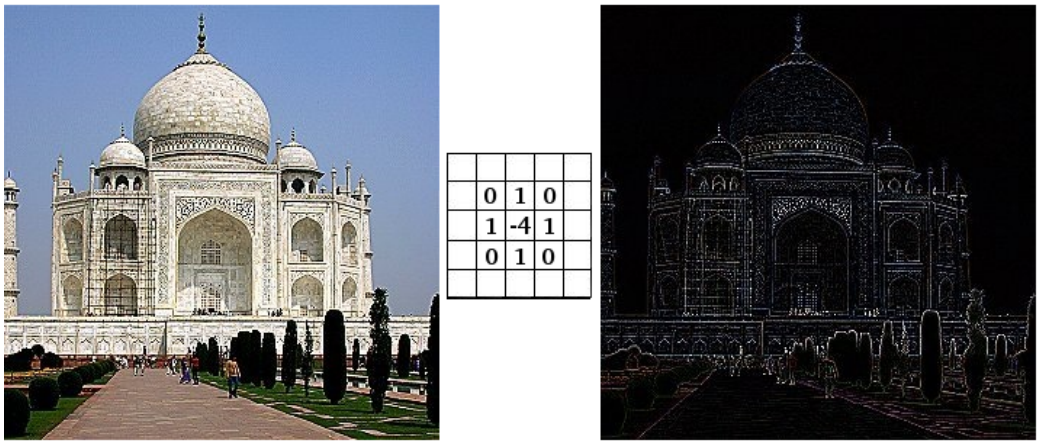

Figure 5: The effect a filter has when it is convolved over all image channels. Taken from the gimp tutorial section. [Gim17]

not needed or too specific and pool the information provided. This speeds up calculations and prevents overfitting. Just like activation functions, there are many possible pooling strategies, the most common version is *max pooling* [SMB10]. Typically in max pooling, a filter with a 2×2 matrix and a step size of two is used. The filter then moves over the convolved feature and takes the maximum value in the filter area. The last layer is a fully connected layer. Its design is identical to the last layer of most ANNs and maps its inputs to the classes. Often the *softmax* function is used[19] as an output function [LW], which produces values from 0 to 1 that add up to one. The results can then be interpreted as confidences that can be handy when evaluating results.

3.2.2. Backpropagation

When training an ANN the parameters (weights and biases) are tweaked using *backpropagation*, a method which uses gradient descent (see Section 3.1.1) to reduce the loss function. When looking at ANNs mathematically, they are actually just compositions of functions. Backpropagation calculates the gradients of expressions through the recursive application of the chain rule (method of differentiating equations that are chained together). The final output should be a vector of partial derivatives for each parameter so they can be changed accordingly. First, a forward pass is done, by inputting the parameters and saving the results. Then backpropagation (backward pass) starts at the end and recursively applies the chain rule to compute the gradients all the way to the inputs of the circuit. Figure 6 shows a simple backpropagation example for the function $f(x, y, z) = (x + y)z$ resulting in $\frac{\partial f}{\partial z} = -4$, $\frac{\partial f}{\partial y} = -4$, $\frac{\partial f}{\partial z} = 3$.

[19] most of the time with *cross-entropy loss* as a cost function

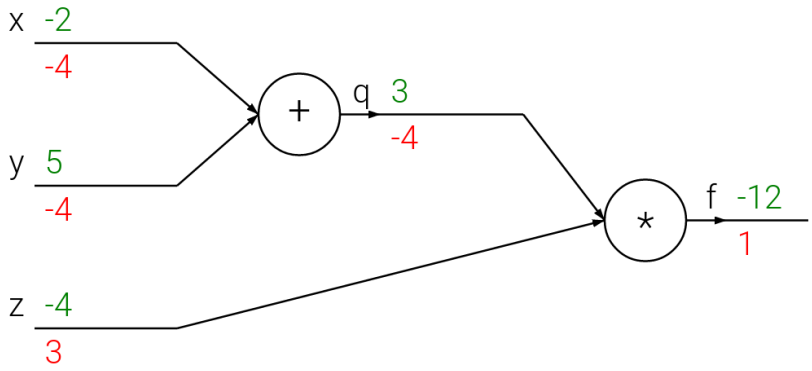

Figure 6: Backpropagation for the function $f(x, y, z) = (x + y)z$ and inputs $x = -2, y = 5, z = -4$. Green shows the values during forward pass, red during the backward pass. The resulting partial derivatives are $\frac{\partial f}{\partial z} = -4$, $\frac{\partial f}{\partial y} = -4$, $\frac{\partial f}{\partial z} = 3$.[cs2]

4. Conception

Originally the topic of this thesis was supposed to be on object recognition in paintings. The plan was to train an Artificial Neural Network (ANN) on the data gathered by the social image tagging project ARTigo. The ARTigo games collect tags for images with the help of humans. A player gets awarded points for naming tags to an image. The amount of points is based on the existing tags. The problem is that pictures that have never been played cannot provide feedback to the player. With a trained ANN it would be possible to generate tags for new paintings and complement the existing data. After some research it turned out that it is not easy to train an ANN nor is the data supplied from the ARTigo detailed enough for object recognition. In particular, bounding boxes for objects that identify the location of an object are missing. The new topic, style recognition in paintings, solves one of those problems as bounding boxes are not needed. Although the new data is not as plentiful, it is still possible to complement the existing tag data with style tags.

But even after changing the topic of this thesis the data problem is not completely solved, as the extracted data quantity is not large enough to train an ANN from scratch. It is very hard to give a general answer on the amount of data needed to train an ANN, but the general consent is that at least a few thousand examples per class are needed. Especially training that involves high dimensional data, like images, generally need a lot more training data. Even the MNIST challenge, which consists of relatively simple images (grayscale and low resolution) has about 60,000 images[20]. Another concern is that training and designing a Convolutional Neural Network (CNN) costs a lot of time and computer power [cs2].

After evaluation the existing work we decided on two different approaches. In the first approach a CNN trained on the task of object recognition is used for fine-tuning. The second takes the same CNN but uses the output of an earlier layer to train a new classifier.

4.1. Datasets

The ARTigo database does not provide a classification of artworks after their artistic styles. Instead user generated tags are available. It is a well known saying in the machine learning (ML) community that gathering and understanding data is one most important parts of writing a ML application. The main challenge is to get reliable data and still keep the amount of data high.

To classify the artworks' images after the artworks' styles we start by getting the 33 most used styles from the *Wikiart*[21] database. This non-profit project started as a group of experts classifying images by hand and is now run in the same collaborative fashion as Wikipedia[22]. It is now wildly accepted as the biggest database of tagged artworks. While the *ARTigo* database is available in three languages, including English. German has the biggest amount of data and therefore is used. A first approach is to query the database for style names gathered from Wikiart.

[20] `http://yann.lecun.com/exdb/mnist/`
[21] `https://www.wikiart.org/de/paintings-by-style/`
[22] `https://www.wikipedia.org/`

As an attempt to boost the available images the *ARTigo Analytics-Center*[23] neighbourhood analysis is used to get tags which have a close distance to our style tags. They will be referred to as *similarity tags*. For the similarity tag selection all tags that have a cosine distance smaller then 0.7 are chosen, as this seems to be a good balance between very specific tags and more general tags. The style names and the extracted similarity tags can be seen in Section A.1.

On early versions of our tag list, we had "Art Nouveau" as a style but had little success with it. After experimenting it turned out that the alternative name "Jugendstil" is much more successful and is used instead. Although "Art Nouveau" and "Jugendstil" shared some similarity tags the actual style names did not appear during the similarity tag extraction. However "Jugendstil" had "symbolism" as a similarity tag and vice versa. There are some more references to style classes inside the similarity tags, like "rokoko" under "baroque". After evaluating we removed the cross reverences for classes that are big enough to make it into the dataset. For others they have been left in to create bigger classes[24]

Next, the styles and the gathered tags are put into a text file (see Section A.1 and the *ARTigo* database is queried for pictures matching the tags. As there is a different amount of similar tags per style, we experiment by selecting a variable amount of similar tags for each style (e.g., tag x might have five similar tags so we use at least two of those to classify a picture to this style, with ten we might use a combination of four). To make sure we have some reliability in our tagged artworks, but still keep the dataset as big as possible, we set the minimum tag usage per image to 2. A bigger number would have produced less overlap between the different classes(styles), but also severely decimate the data and would not provide enough for the representation of some classes. Next, classes with less then 200 pictures are removed, to give the ML classifiers sufficient data. Table 1 gives an overview about selected datasets and the strategies used to create them. Only datasets who use similar tags have an entry inside the "min. similar tags combination" row. *WikiartSel* is a special case, as it was not extracted from the ARTigo database, but downloaded from the Wikiart database. *ExtraAll* is only mentioned as additional information and is not used to train any classifiers (more later).

To measure the performance of our classifiers and the meaningfulness of our feature extraction methods in the future, a subset of the *Wikiart* database matching the classes extracted from the *Artigo* database is created. *Extra2* with its eight classes is the biggest usable dataset extracted. To get an accurate comparison, *wikiartSel* dataset contains the same style classes as *extra2*. When composing the *wikiartSel* dataset, the downloaded paintings are first scaled down to a size of 800×600 pixels using the same procedure that has been used on the ARTigo paintings. Then a random selection of exactly 2000 pictures for each class is selected. We do this trying to eliminate any potential problems that might occur with imbalanced datasets.

Table 2 shows the different class sizes and exclusivity of features per class. When analysing exclusivity it becomes clear that datasets like *extraAll* are not usable because of the big overlap some classes have. For example, the class *symbolism* only contains 5.34% pictures

[23] `http://analytics.pms.ifi.lmu.de/`

[24] Baroque has rokoko as a similarity tag. As Rokoko has too little pictures on its own, we leave it as a similarity tag for baroque.

dataset name	number of classes	use similar tags	min. similar tags combination
rawStyles	7	$false$	-
extraAll	12	$true$	-
extra2	8	$true$	2
extraVar	7	$true$	$\|\text{sim. tags}\| > 6 = 4$ $\|\text{sim. tags}\| > 4 = 3$ $\|\text{sim. tags}\| < 4 = 2$
wikiartSel	8	-	-

Table 1: An overview for the used datasets. Dataset name is the name we gave the dataset to reference it, number of classes shows the number of classes the dataset possesses and min. similar tags combination shows how many similar tags are used additionally to the main style tags.

that are not shared with other classes. This probably could have been avoided by choosing a smaller similarity distance when selecting tags from the *ARTigo Analytics Center*, though it would have limited the size of our other datasets.

To get a better understanding of the selected datasets we also created a Chord Diagram for each of them. They have been created using a modified D3[25] script by Mike Bostock [Bos]. Each colour represents a class of styles. The marks on the side denote the number of images per class. The connections between the different classes represent the shared pictures between them. Figure 7 shows the generated diagram for the rawStyles dataset. The diagrames for the remaining datasets are available in Section A.1.

4.2. Fine-tuning an existing CNN

Although we could not find many related works on fine-tuning for style recognition we still wanted to retrain a CNN. Since the extracted datasets have a decent size but are still very different from the data most CNNs are trained on it should be possible to finetune an existing CNN [cs2].

When fine-tuning (sometimes just called transfer learning) we take a fully trained CNN and remove the last layer of the network. We then place a new layer on top, matching our new output classes. The ANN is then trained starting with the pre-initialized weights. This approach relies on the hypothesis that filters trained on a specific task can be tweaked without much effort to fit another task. Yosinski et. al demonstrated in *How transferable are features in deep neural networks?* [YCBL14] that fine-tuning a CNN can lead to very good classification results, even if the data is vastly different. Training of the new CNN happens much faster, needs less computational power and data. A downside is one still has to train a CNN. A task that can be difficult without pre-existing knowledge and experience.

[25] `https://d3js.org/`

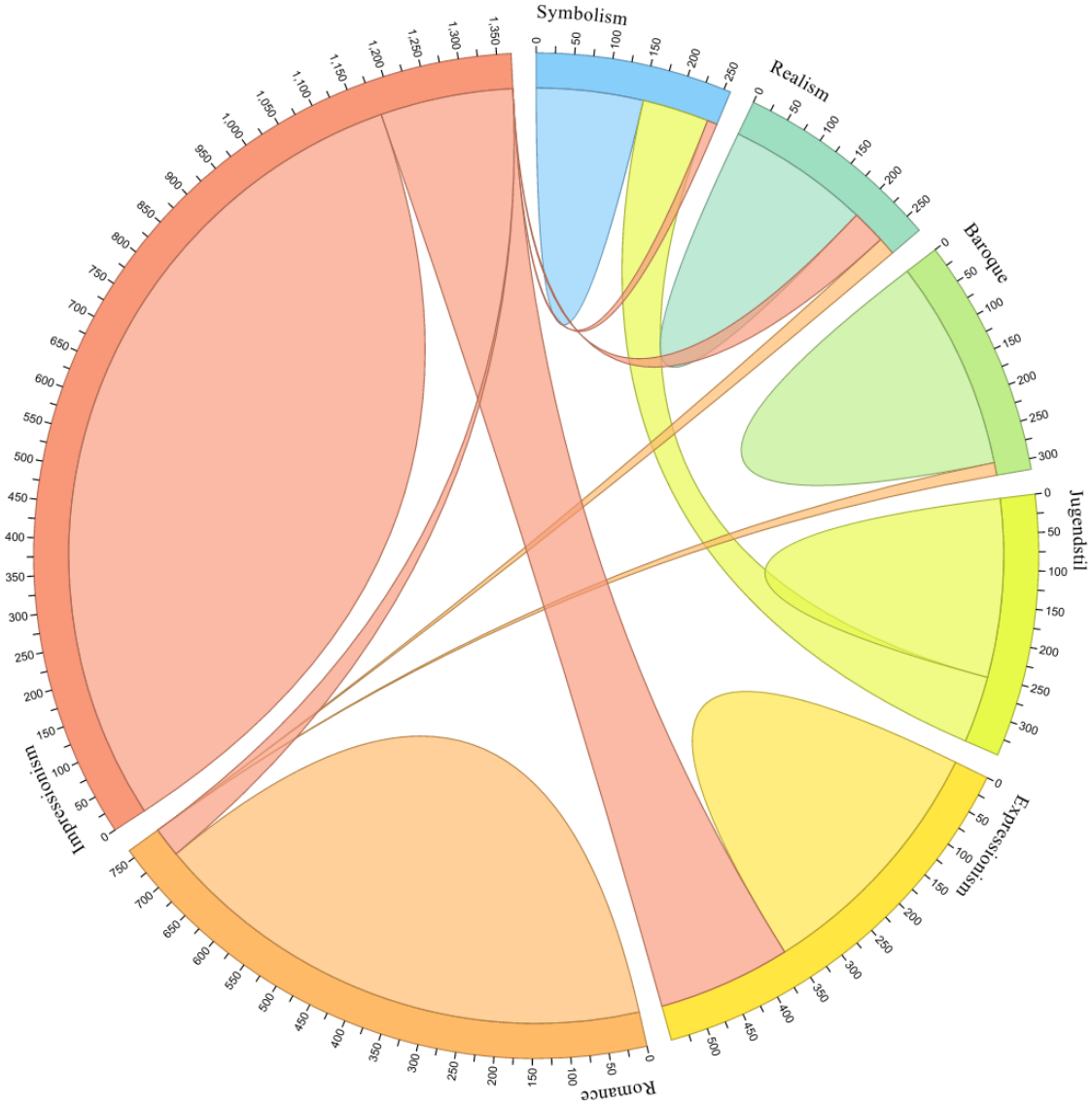

Figure 7: A chord diagram for the *rawStyles* dataset.

dataset	class name	features	exclusive to class
rawStyles	Symbolism	266	58%
	Realism	279	72%
	Baroque	332	90%
	Jugendstil	376	67%
	Expressionism	558	65%
	Romance	824	88%
	Impressionism	1476	80%
extraAll	Symbolism	299	7%
	Minimalism	277	13%
	Realism	8360	9%
	Naive Art	347	15%
	Art Nouveau	364	31%
	Cubism	678	6%
	Jugendstil	1281	19%
	Abstract Art	536	1%
	Baroque	482	14%
	Expressionism	1307	4%
	Romance	10067	27%
	Impressionism	3094	1%
extra2	Symbolism	266	14%
	Realism	5031	24%
	Cubism	293	33%
	Baroque	335	48%
	Expressionism	618	16%
	Romance	6350	41%
	Impressionism	1598	10%
	Jugendstil	472	50%
extraVar	Symbolism	266	40%
	Realism	2406	43%
	Jugendstil	376	64%
	Baroque	335	79%
	Expressionism	618	48%
	Romance	2626	54%
	Impressionism	1506	37%

Table 2: Details for the extracted datasets. Above table shows the class names and images for each class. Exclusive to class denotes the percentage of pictures that are only available inside this specific class. Features the number of images for this class. I.e. inside the extraAll dataset "Abstract Art" only has 1% exclusive images for the 514 images inside its class (≈ 5).

4.3. Using a CNN for feature extraction

Similar but still different is the approach of using a CNN for feature extraction. Again a trained CNN is used. This time we take the output of the m-th layer for every input image put into the CNN. It has been shown that often it is the last layer hidden layer that produces the best output (see Section 2.1). We then take a shallow learning approach and train a classifier on the data. This approach relies on the hypothesis that CNNs trained on a specific classification extract features that can be reused for other tasks. There has been good success using a Support Vector Machine (SVM) with a simple linear kernel. The advantage of using this approach is that it is much faster and easier to design. It is also not too hard to add additional inputs from other classifiers. A disadvantage is that using the wrong method for feature extraction (the wrong CNN) can get you stuck. Even choosing the right algorithms will not help if the used feature extraction methods are not good enough.

5. Implementation

After presenting the four different datasets, we will now go into detail of the actual implementation. First, we will fine-tune a CNN on the various datasets shown in Table 1. We will then use the same (but not fine-tuned) CNN to extract features from the image datasets and train a classifier on the extracted features. The main focus will lie on using classical ML algorithms on the extracted features, as it is easier to implement and most of the related work do the same.

5.1. Training Tools

When starting to get into the topic of deep learning we felt that it is important to spend some time on choosing a good ML framework. With progression of the thesis the focus moved from deep learning into the direction of classical ML. There are various frameworks that can be used to create your own deep learning applications. Some big names that come to mind are *TensorFlow*[26], *Theaono*[27], *Torch*[28] and *Caffe*[29]. We decided at the start that we want to use Python as a programming language so the list gets a lot more manageable. Tran provides a very nice overview for deep learning frameworks [Tra16].

Caffe often get criticized for its poor modelling capabilities and seems a bit outdated. We were still planning on using Caffe due to its extensive collection of models and good documentation, but soon discovered that there is a project for converting Caffe models to Tensorflow. Considering the overview Tran provided and the fact there is already a distributed Tensorflow architecture deployed at our professorial chair[30] we decided to use Tensorflow as our deep learning framework of choice.

There are some works like [BRSS15] that benchmark deep learning frameworks including TensorFlow where it did not perform well. However, one must consider that TensorFlow is still very young and its performance increased greatly since then.

As both of our approaches rely on a trained network we had to decide on a architecture. Using TensorFlow, the *Inception* model is an obvious choice. Inception is a CNN written in Tensorflow trained on the *ImageNet*[31] object recognition challange. *Inception-v3*, the most recent version of Inception [SVI+15], reaches a top 5-error rate of 3.46%, which is better then some humans [kar]. Figure 8 shows the inception architecture. One must not get confused by the second softmax layer at the bottom. It is an *auxiliary classifier* used to speed up convergence and help with the *Vanishing gradient problem*[PMB13]. For more information about the inception-v3 network refer to the original paper "Rethinking the Inception Architecture for Computer Vision" by Szegedy et. al. [SVI+15]. As it has been shown that a CNN trained on the task of object recognition is suitable for style classification we use the trained *Inception-v3* model for our tasks.

[26] `https://www.tensorflow.org/`
[27] `http://deeplearning.net/software/theano/`
[28] `http://torch.ch/`
[29] `http://caffe.berkeleyvision.org/`
[30] by Yingding Wang and Josef Birkner
[31] `http://www.image-net.org/`

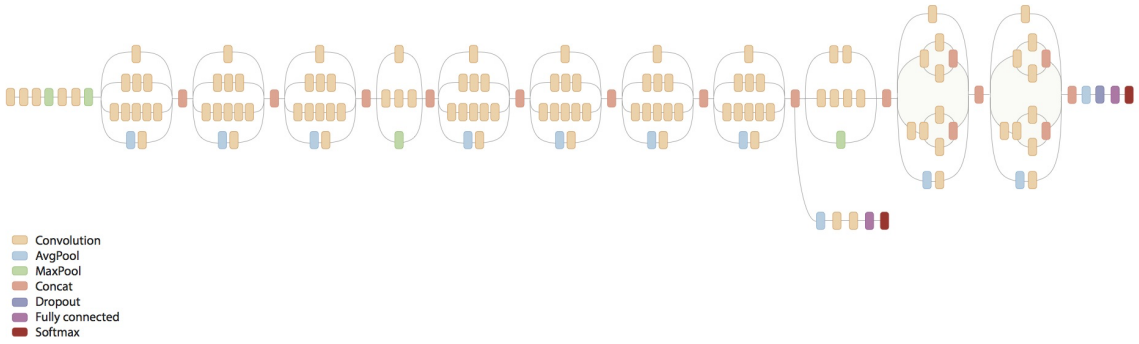

Figure 8: The inception v3 architecture[Ten17]. Details to its implementation are available in the original paper [SVI+15]

TensorFlow focuses on deep learning problems and does not provide functionality for the shallow learning tasks we require. Originally started as a project as part of Googles "summer of code", *scikit-learn*[32] has established itself as one of the biggest libraries for ML in python. Based on libraries like *NumPy*[33] and *SciPy*[34] they provide fast all around functionality and have a extensive documentation. For visualisation *matplotlib*[35] and *plotly*[36] were used. Most of the code was written as part of a *jupyter*[37] notebook, which is very convenient when working with visualization and data analysis. Many of above mentioned libraries are bundled inside the *anaconda*[38] data science platform. It provides a package manager that ships with many libraries.

5.2. Fine-tuning

When fine-tuning the Inception architecture, a modified version of the available retraining script is used[39]. It uses a "vanilla" version of gradient descent (see Section 3.1.1) with a fixed learning rate. There is another, seemingly newer, version of the network and training script available (implementing rate decay among other things) on the official git repository[40]. Unfortunately due to constant changes to the TensorFlow API it is not usable at the time this thesis is written. After some research we came across open bug reports confirming the issue.

We start by training our *wikiartSel* dataset to get a feeling for the hyperparameters, without having to worry about class imbalances or wrong truth labels. It also provides a benchmark

[32] `http://scikit-learn.org/`
[33] `http://www.numpy.org/`
[34] `https://scipy.org/`
[35] `http://matplotlib.org/`
[36] `https://plot.ly/`
[37] `http://jupyter.org/`
[38] `https://www.continuum.io/`
[39] `https://www.tensorflow.org/tutorials/image_retraining`
[40] `https://github.com/tensorflow/models/tree/master/inception`

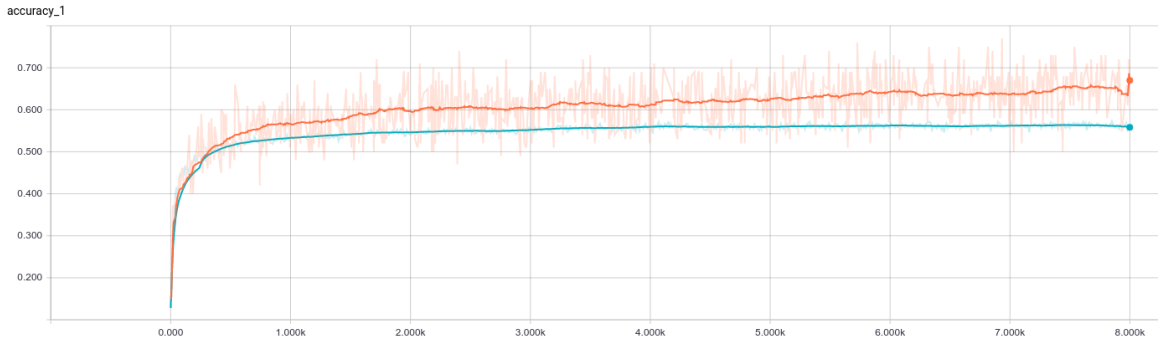

Figure 9: The accuracy graph while fining the inception-v3 architecture on the *wikiartSel* dataset. Blue represents the validation accuracy and orange training accuracy.

for other datasets. To measure our performance we choose accuracy. We will use other metrics in Section 5.5.1.

The data is split in 90% training data and 10% testing data. Starting with a learning rate of 0.1 and 4000 training steps we achieved an accuracy of around 51% ($\pm$0.1). Lowering the learning rate to 0.01 and doubling the training steps to 8000 iterations, we were able to improve the validation accuracy to 57%. Lowering the learning rate to 0.005 or even 0.001 did not yield better results, even when further raising the training steps. The reason might be that gradient descent got stuck in a local minima. Adding random brightness to images or rotating images is a common measure to improve generalisation and boost the training examples. However, it did not further improve the accuracy. Figure 9 shows the accuracy during the most successful training of the *wikiartSel* dataset. No big drop off in accuracy is seen at the end of the training, which would occur in case of overfitting. Note that it seems like the training accuracy is improving during the end of the training. This is just a side effect caused by applied curve smoothing. One might think that improving the training steps further might yield better results but all that it accomplished was causing the model to overfit. Tough, after further testing we were not able to further improve the performance. We then trained models on our remaining datasets. All of them had the most success with using a learning rate of 0.01 and 4000 training iterations. The results are available in Section 5.5.1.

5.3. Feature Extraction and Shallow Learning

Following the work mentioned in Section 2.1 we use the last hidden layer[41] of the Inception network for feature extraction. The Inception architecture calls this layer "pool_3:0"[42]. This reduces the data dimensions from a maximum of $800 \times 600 \times 3$ to a vector with 2048×1 dimensions.

To visualize these high dimensional features we use *t-Distributed Stochastic Neighbor Embedding* (t-SNE) [vdMH08] to reduce the dimensionality to first two and then three dimensions. Note that t-SNE is completely unsupervised, meaning the algorithm does not know the label

[41] ignoring the dropout layer which is used to avoid overfitting [SHK+14]
[42] light blue, far right, shortly before the classification layer in figure 8

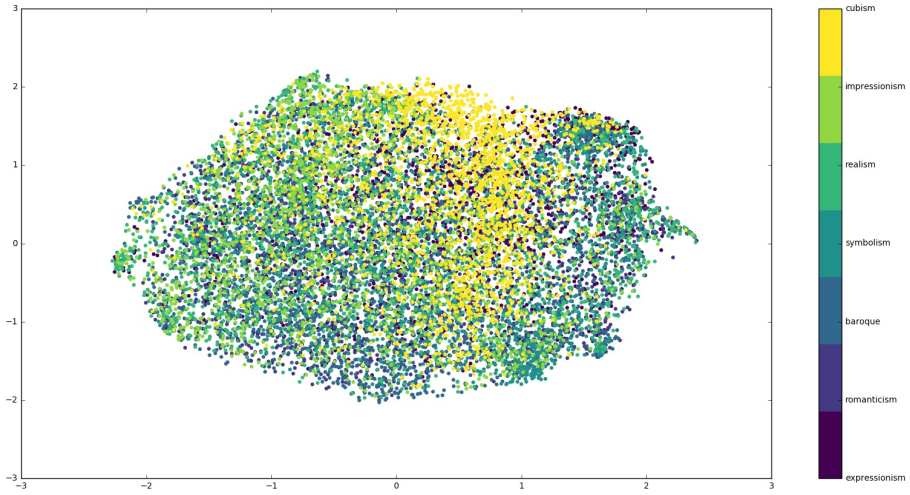

Figure 10: t-Distributed Stochastic Neighbor Embedding (t-sne) used to visualise the extracted *wikiartSel* features generated using matplotlib.

of the data. Without going into to many details t-SNE calculates the similarity between pairs of high dimensional objects[43]. Based on the similarity a probability distribution is constructed. Another probability distribution is constructed for the target dimensional space. The difference between the distributions is then minimized using gradient descent.

t-SNE also has some hyperparameters like learning rate or *perplexity*. Again, we used our *wikiartSel* dataset as a benchmark. The best results were achieved by setting a learning rate of 200, a perplexity of 40, 5000 learning iterations and using the euclidean distance. As recommended by the scikit-learn documentation [TSN], principal component analysis (PCA) was used to reduce the dimensionality before running t-SNE. Figure 10 shows one 2D visualisation of the extracted features. Each color represents a class of image styles. The visualisation shows that t-SNE has descent success clustering the styles cubism, art nouveau and baroque. Additionally a 3D visualisation of the same features was created. Again t-SNE had good success in clustering symbolism, baroque and art nouveau. Visualisations for all the other datasets are created, using above hyperparameters. T-SNE still has success clustering cubism and baroque, but the results seem less clear than for the wikiartSel dataset. This might be a sign of too fuzzy selection criteria. In summary, it seems like even without training, a CNN trained on object recognition is able to extract features relevant for the task of style recognition.

As a next step we split our new data (the extracted features). The same train/test split of 90%/10%, as used when retraining the CNN, is applied. Then 5-fold cross validation is used to train a SVM classifier on the training set. SVMs are designed for binary classification, but can be applied to multi-class classification. The scikit-learn library handles this according

[43] using the euclidean distance as default

dataset	hyperparameter	best performing value
wikiartSel	Cost	10
	Gamma	0.001
	Kernel	rbf
rawStyles	Cost	10
	Gamma	0.0001
	Kernel	rbf
extraVar	Cost	10
	Gamma	0.0001
	Kernel	rbf
extra2	Cost	10
	Gamma	0.0001
	Kernel	rbf

Table 3: The used datasets and their best performing hyperparameters.

to a *one-vs-one* scheme, in which binary classifiers are trained on each class pair. When predicting a new sample, the different classifiers then vote and the class with the most votes is selected. To begin, the *wikiartSel* dataset is used. While using a linear kernel and the default hyperparameter[44] the scikit-learn library provides, an accuracy of 0.48% was achieved. As a next step, following some of the related work, other non linear SVM kernels are evalualted. More precisely the radial basis function (rbf) kernel[SS02] and the chi-squared (χ^2) kernel[SS02] are tested. We were not able to repeat the success Siddharth et. al. had with the χ^2 kernel, scoring only 0.12% accuracy. As our benchmark testing set has 8 classes this corresponds to a random guess[45]. The rbf kernel performed best with an accuracy of 54% (standard deviation of 0.2%) and is used for the rest of our testing.

Often accuracy alone is not a good metric to measure performance. Especially when an imbalanced dataset is used, it helps to get a more in depth information. For this reason and future reference we, calculate the *F1-score* and some other metrics (more in Section 5.5.1). It ranges from zero (worst) to one(best) and is calculated as the harmonic mean of precision and recall [Pow11].

A hyperparameter grid is generated containing possible hyperparameters. The grid contains cost values from 1 to 1000 and experiments with different gamma values. The gamma value represents the influence of the different training samples. A low value improves generalisation and promotes a "smoother" model. A high value tries to fit more features. The gamma hyperparameter is only applicable when using a non linear kernel. The cost hyperparameter assigns a cost to each misclassified training example. For large values, the optimization will choose a smaller-margin hyperplane if it classifies all the training points correctly. Grid search with 5-fold cross validation then traverses this grid.

Next, a classifier is trained on the *rawStyles* dataset. Again, the rbf kernel outperformed the linear kernel. It is noteworthy that during hyperpameter optimisation the hyperparameters stayed similar, but the best performing gamma value decreased to 0.0001. Because of the

[44] cost penalty of 1.0
[45] $1/8 \approx 0.12$

imbalanced dataset, we predicted one-sided predictions, known as the *accuracy paradox* (or class imbalance problem)[Chi13]. Performance metrics like the F1-score did not hint towards any problems. If this would have been the case, it is possible to provide weights for the under represented classes, giving the SVM more "incentive" to choose them.

We repeated the classifier training and hyperparameter tuning for each of our datasets and save the best performing classifiers. Some additional classifiers are then trained on subsets of the *wikiartSel* dataset that have one or more classes removed. This is done to help the comparison to the other datasets with less classes in Section 5.5.1. Table 3 lists the datasets used and the best performing hyperparameters.

5.4. Novelty Detection and Practical Application of the Classifiers

It was only in the final stages of this work that we started thinking about the practical applications of the trained classifiers. As mentioned the motivation to train a classifier was to predict new tags for the ARTigo database. However, our training and testing data does only contain images that belong to a certain class. So, when running the classifier on new data, it will always predict a possible class, even if the image does not belong to any of them.

A few possible approaches were evaluated to solve this problem, which is known as *novelty detection*. Novelty detection is very closely related to *outlier detection*. The difference is that novelty detection presumes a clean training dataset and predicts if new data belongs to this old data or not. Outlier detection presumes a "dirty" dataset and aims to identify those outliers.

The evaluation of such algorithms can be challenging, especially if one can not be sure if a data point is in fact novelty data. The ARTigo dataset of street-art images was used to evaluate the novelty detection techniques, as they promise to only contain novelty images.

One of the most used novelty detection algorithms is a *One Class SVM*[SWS$^+$00]. Another possibilty is to use *RandomForest* classifiers[ZZN$^+$15]. It is even possible to use an ANN for novelty detection[MS03]. In our first approach, we trained a One Class SVM on our training data. But even after evaluating a variety of different hyperparameters, this approach was discarded as unsuccessful. Next, a One Class SVM was trained on each of the classes, hoping that this would improve performance. Unfortunately almost no improvement could be seen and the results were not usable for practical applications. Lastly a RandomForest classifier was trained. Again, even after evaluating multiple hyperparameters, it had little to none success in performing usable novelty detection.

As a last effort, to practically use the classifiers, the top performing SVM classifier is retrained to probabilistic output. SVMs are normally not able to output probabilities, but by using *Platt Scaling*[P$^+$99] it is possible. Table 4 shows the mean probabilities we got when classifying our test set. It displays the mean probabilities for the correctly classified images, the misclassified images and the mean of all classification probabilities. Although a smaller dataset and very different images the mean score for the street-art images scored a mean of 0.427 with a standard deviation of 0.141. This indicates, that setting a high enough threshold might be sufficient to separate the data and still preserve some of the

	mean	standard deviation
all classifications	0.552	0.203
correctly classified	0.634	0.201
misclassified	0.443	0.148

Table 4: The mean probabilities and their standard deviations for the classification results of the *wikiartSel* test set.

classification power of the classifier. When setting the threshold, one has to think about the potential applications of the classifier. In our case, we want to tag images of the ARTigo database that have not been tagged yet. With this new data, the players are able to receive feedback for their guesses (new tags). As the players are available to confirm or deny the classification results, we propose a smaller threshold, allowing more images to be tagged. The threshold is set to 0.5, reducing the classification accuracy of the best performing *wikiartSel* classifier from 0.5675 to 0.3875. However, it also leads to 70% of the steet-art test data to be recognised as novelty data.

5.5. Evaluation

In the following section we will take a closer look at the performance of each classifier, using not only accuracy but more detailed scoring functions and visualisations. At the start of the section as an introduction a short overview of the available scoring method will be given. At later stages the the best performing classifiers will be used on the other datasets there have not been trained on. Finally we will take a look of some actual images of each class.

5.5.1. Scoring Metrics

When measuring classification performance most of the related work only use accuracy as the only measurement. However, as the already mentioned accuracy paradox[Chi13] shows, accuracy alone can, give false impressions, as it is not class dependent. An imbalanced dataset like those at hand is in need of class dependent metrics.

We will first define some terms: true positives, true negatives, false positives, and false negatives compare the results of a (binary) classifier with the actual data labels. E.g., a classifier predicts a picture to be of class expressionism and its class really is expressionism (truth). This example would be a true positive (tp). A positive truth label with a negative prediction would be a true negative (tn), a positive truth label and a negative prediction a false negative (fn) and lastly a negative truth label and a positive prediction is a false positive(fp). Table 5 gives a easy to read overview over the described naming conventions. Using these definitions, accuracy is the sum of all TP and TN divided by the testing data. Furthermore, *precision* is defined as the number of correct positives devided by all positives. *Recall* is defined as the correct positives divided by positives that should have been predicted. Finally, the *F1-score* is defined as the harmonic mean of precision and recall. It is noteworthy that for a balanced dataset those four metrics can be quite similar. They can be formalised

		prediction	
		prediction positive	prediction negative
Truth	positive label	True Positive (TP)	False Negative (FN)
	negative label	False Positive (FP)	True Negative (TN)

Table 5: naming conventions for the different kinds of classification results

dataset	accuracy	standard deviation
rawStyles(7)	0.56	±0.016
extra2(8)	0.56	±0.017
wikiartSel(8)	0.54	±0.006
extraVar(7)	0.41	±0.014

Table 6: The accuracies for the different datasets, calculated using the mean of 5-fold cross-validation. The brackets denote the classes per dataset.

as follows:

$$\begin{aligned} \text{Accuracy} &= \frac{\sum (tp + tn)}{\sum (tp + tn + fp + fn)} \\ \text{Precision} &= \frac{tp}{tp + fp} \\ \text{Recall} &= \frac{tp}{tp + fn} \\ F_1 &= 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} \end{aligned} \tag{6}$$

In addition to precision and recall, confusion matrices are used. They give information about the classified images of the test set and make it easy to spot if classes might get confused with others.

5.5.2. Shallow Learning with Support Vector Machines

When simply measuring accuracy (shown in Table 6), one might be inclined to crown the *rawStyles* dataset as the best performing one. But especially when comparing datasets with a different amount of classes accuracy is very misleading. The only thing one might (carefully) note is that boosting the number of images using a variable amount of tags does not seem to help.

To get more understanding of the classification results, we will take a look at the calculated precision and recall values. Table 7 shows those values for our benchmark set, *wikiartSel*. It also gives an overview for the conception of the test set as it shows the number of test samples under the support row. It seems like the classifier does best when classifying images of the cubism and baroque class. It is easy to see that this is the case for cubism but a bit surprising in the case of baroque. As a visualisation, a confusion matrix is generated, which is available in Figure 11. Its x-axis shows the predicted labels and the y-axis shows the actual truth labels. The numbers show the proportion of total images that got classified

	precision	recall	f1-score	support
Expressionism	0.45	0.46	0.46	208
Romance	0.47	0.52	0.49	198
Baroque	0.70	0.76	0.73	202
Art Nouveau	0.59	0.59	0.59	212
Symbolism	0.49	0.48	0.49	188
Realism	0.45	0.40	0.42	203
Impressionism	0.59	0.57	0.58	196
Cubism	0.78	0.76	0.77	193
avg / total	0.57	0.57	0.57	1600

Table 7: Scoring metrics for the *wikiartSel* dataset. The support row shows the amount of images in the testing set. The classifier was trained using a SVM.

into another class. E.g. 46% of pictures from the expressionism class have been classified as expressionism. Figure 11 shows a diagonal line, indicating that the majority of images are classified into the right class. The biggest miss classification is 18% of pictures of the realism class that are classified as romantic. This jumps out, as realism started as a counter-movement to romance, criticising its unrealistic depiction of life.

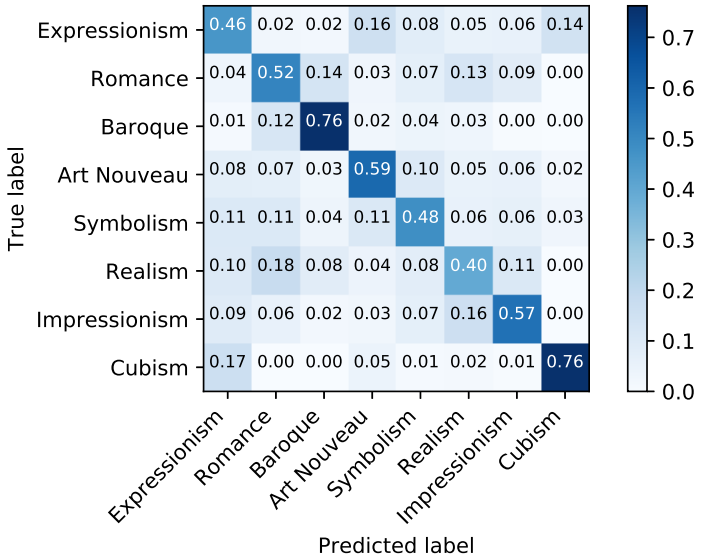

Figure 11: A confusion matrix generated using the classifier trained on the *wikiartSel* dataset.

It is generally more intuitive looking at confusion matrices then the raw scoring metrics. For this reason we will use them to evaluate the classification results. However, in depth tables for all available datasets are available in Section A.3. The provide scoring metrics and information about the test set sizes.

Figure 12 shows a confusion matrix for the rawStyles dataset. In contrast to the classifier

trained on the *wikiartSel*, no diagonal line is visible. The classifier seems to do a fine job on baroque and romance. It confuses symbolism and art nouveau which does not seem too surprising as those classes are quite similar and even had references onto each other when using the ARTigo analytic center. However, it seems to heavily favour impressionism as a class. Our hypothesis is that it seems to be a problem with class imbalances as the impressionism is the biggest class in the dataset. To confirm this, the *wikiartSel* classifier is run on the same testing data.

Figure 14 shows result for the *wikiartSel* classifier, run on the *rawStyles* dataset. The results seem to confirm the hypothesis as most of the data is classified into the right classes. There is some confusion between expressionism and impressionism. Furthermore, the classifier seems to classify a big potion of test data as romance. It particular, it confuses baroque with romance. This might indicate, that players playing the ARTigo games, often confuse baroque and romance. It could also indicate that the players have problems identifying romantic images. Although there are some classification problems, it also shows that there is some accuracy in the generated datasets (and therefore user generated tags).

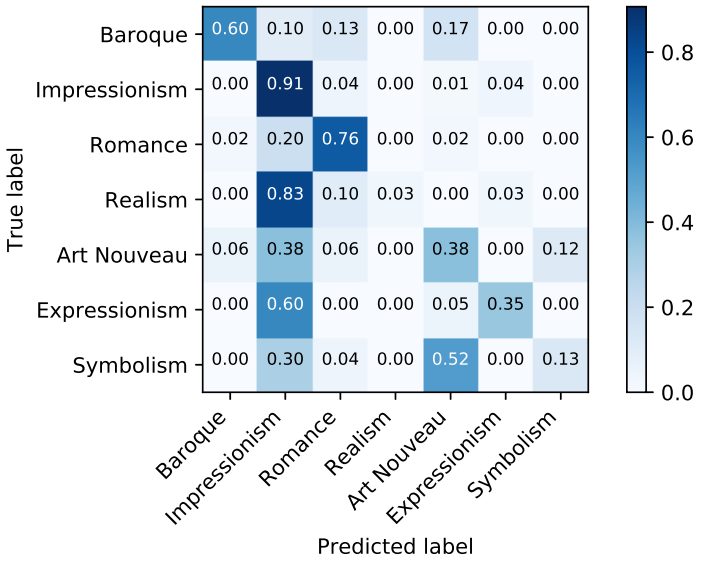

Figure 12: A confusion matrix generated using the classifier trained on the *rawStyles* dataset. This classifier is ran on the same dataset.

Unfortunately the classifiers trained on the remaining datasets show a similar tendency. The classifier trained on the *extra2* dataset favours the realism and romance class. The classifier trained on the *extraVar* dataset also favours the realism and romance class. Again, the *wikiartSel* classifier is ran on the above datasets. The classifier performed worse on the *extraVar* dataset then on the *rawStyles* dataset in every class. However, the *extra2* dataset beats the *rawStyles* dataset in some cases. Figure 13 shows the results on the *extra2* dataset by running the *wikiartSel* classifier on it. The *wikiartSel* classifier showed descent performance and even beat the classification accuracy of Cubism and Art Nouveau compared to the *rawStyles* dataset. This could indicate the efficiency of our image collection

methods in those classes.

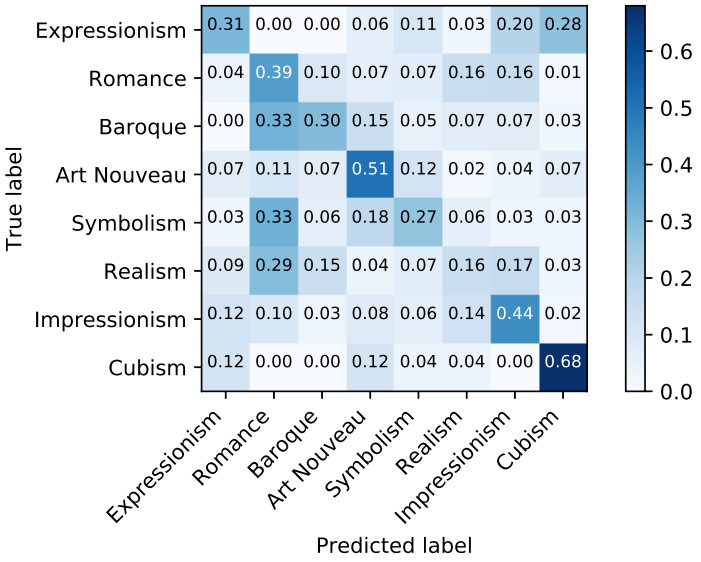

Figure 13: A confusion matrix generated using the classifier trained on the *wikiartSel* dataset. This classifier is then ran on the *extra2* test data.

5.5.3. Fine-tuning the Inception Network

As mentioned, most of the time invested went into training a SVM. However even with the comparably little work invested in the retraining of a CNN it performed well. Table 8 gives an overview over the accuracies for the different datasets. No standard deviation is listed, as no cross validation was used. This is generally not done when training a CNN as the datasets are very big anyway. However, the datasets used in our case relatively small. This makes the measured accuracies and generally the evaluation not as constant as possible. This could have been avoided by using less training data and more testing data.

When comparing the results in Table 6, the accuracies are similar on the *rawStyles* and the *wikiartSel* datasets. However, the accuracies on *extra2* and *extraVar* dataset are much worse. This might be caused by the accuracy paradox. To investigate and get a better understanding of the trained classifiers we again generate a confusion matrix for the trained datasets.

Figure 15 shows the confusion matrix on the *wikiartSel* dataset. Most of the classes have a very similar accuracy, but the CNN outperforms the SVM in a few classes like cubism and realism.

Figure 16 shows the confusion matrix for the *rawStyles* dataset. Unlike the classifier trained using a SVM the classifier is not affected by the class imbalances. It still confuses symbolism and art nouveau which seems to confirm the assumption (mentioned in Section 4.1) that people have problems differentiating between the two classes.

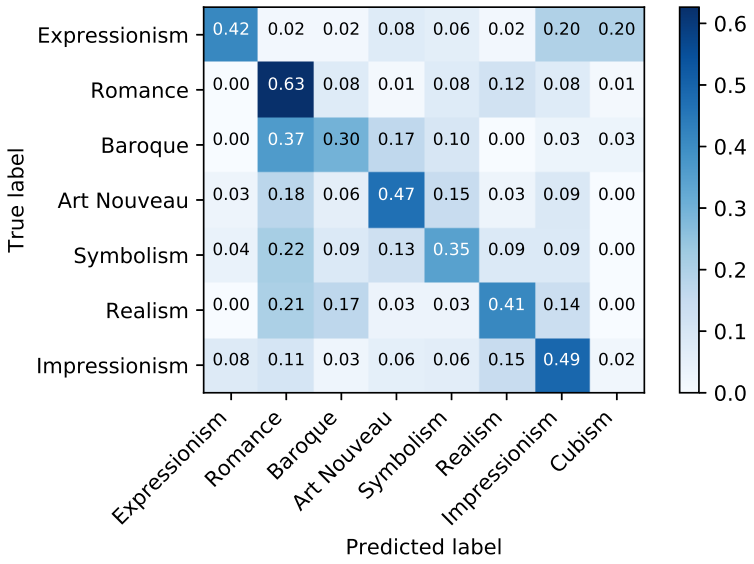

Figure 14: A confusion matrix generated using the classifier trained on the *wikiartSel* dataset. This classifier is then ran on the *rawStyles* test data.

dataset	accuracy
wikiartSel(8)	0.571
rawStyles(7)	0.543
extra2(8)	0.374
extraVar(7)	0.429

Table 8: The accuracies for the different datasets using finetuning of a CNN. The brackets denote the classes per dataset.

Even more prominent as then shown in Figure 14 the classifier in Figure 16 seems to confuse realism and romance. This is odd, as realism tries to represent the subject matter truthfully without any added artificiality. This is almost the opposite of romantic paintings, which are characterized by its emphasis on emotion. It seems that the extracted features are not able to represent this. Furthermore there seems to be some confusion between impressionism and expressionism. Overall, the classification results seem to be a lot less scattered and more concentrated then the results in Figure 14.

Figure 17 shows the confusion matrix on the *extra2* dataset. It is clearly visible that the classification got a lot more fuzzy. Classification on symbolism, realism and cubism dropped. Its noteworthy that cubism dropped by such a big margin, as the *extra2* dataset seems to perform descent in Figure 13.

Displayed in Figure 18 is the confusion matrix for the *extraVar* dataset. Baroque and romance have gained accuracy and the generally the accuracies are higher. This might be a side effect of the less classes in the dataset tough.

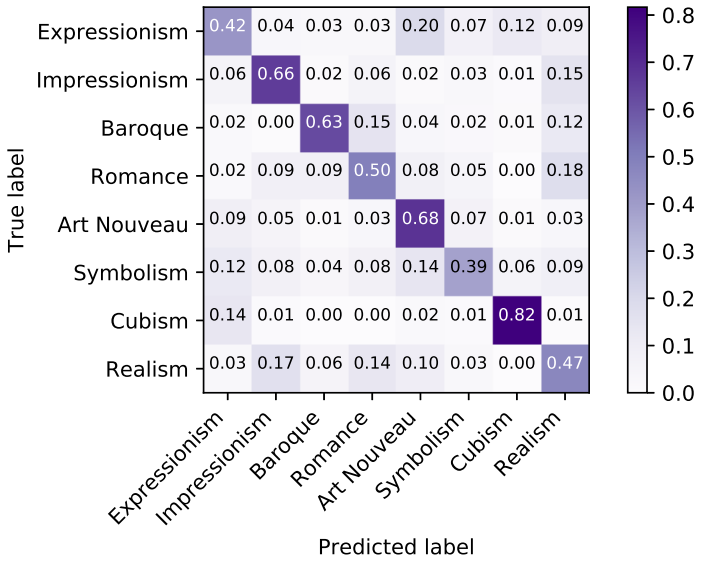

Figure 15: The CNNs classification result on the *wikiartSel* dataset

5.5.4. Performance

When trying to compare the performance of the two approaches we first look at our benchmark set, *wikiartSel*. The performance is very similar for all classes, with both approaches scoring an average F_1-score of 0.57. It is not an easy task to compare the rest of the classifiers, as the SVM classifiers show a bias for the biggest class. Also, to maximise training data, only 10% test data were chosen. Since, the training scores might be misleading and we try not to over-analyse the scores at hand. Overall the training on the *wikiartSel* dataset seems most successful. The trained CNN and SVM both had great success classifying baroque and cubism. They also showed good success in classifying art nouveau and impressionism. The second best performing classifier is the CNNs', trained on the *rawStyles* dataset. It has good success classifying images from the classes romance and impressionism. All in depth scores can bee seen in Section A.3.

Its underwhelming that the imbalanced datasets produced one-sided results when using a SVM. Although SVMs might just not perform well for our data, it is also possible that there is a weight distribution for the SVMs that would have produced better results. Other alternatives would have been over-, or under-sampling of data (e.g. SMOTE[CBHK02]) or training an outlier detection ML algorithm on the data. However, if the same class distribution is present on the data we want to predict on, the unbalanced training data is a good representative of the data, and hence, the unbalance is desired.

In conclusion, the SVM classifiers trained on the extracted ARTigo datasets did not show great success, even tough the data seems accurate as shown by running the *wikiartSel* classier on the data. Another dataset extraction method, only selectively adding more samples to underrepresented classes might be promising. An additional approach would be to train a outlier detection performs a little better, but in the end its its not too surprising as both

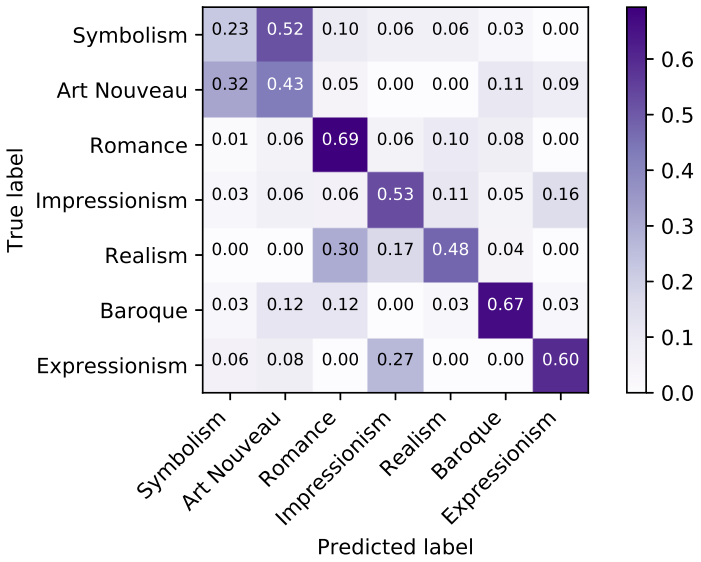

Figure 16: The CNNs classification result on the *rawStyles* dataset

approaches start of with the same features. For the CNN to be able to recognize features that represent a style better, more data is probably needed when retraining.

5.5.5. Classification Examples

Following some examples that the top performing classifier, trained on the *wikiartSel* dataset, classified. First, a list of misclassified images will be presented. It will be limited on the best performing classification classes: "cubism", "baroque" and "art-nouveau". Then we will hypothesise about the reasons for the misclassifications. All of the shown images were selected at random. Section A.4 includes a full list of the misclassified images and their predicted class.

The author of this thesis does not have a background in arts and one can only hypothesise about features the CNN extracted from the images.

Figure 19 shows misclassified from the class cubism. Typical cubistic images appear fragmented and abstracted. This is not the case for the images in Figure 19 and it seems logical that the classifier was not able to correctly identify those samples.

Figure 20 shows images of the art-nouveau class. Its misclassification as cubism could be caused by its straight lines. Figure 20c, also an image of the art-nouveau class, got classified as romantic. A reason might be the colour choice.

Figure 21 shows three baroque images. Figure 21a and Figure 21b show a colour scheme that is often seen in baroque style. Correctly classified images from this style often show people in the foreground, that is not the case for those two. Figure 21c has a different colour scheme and does not contain people. This might be a reason for the misclassification in this case.

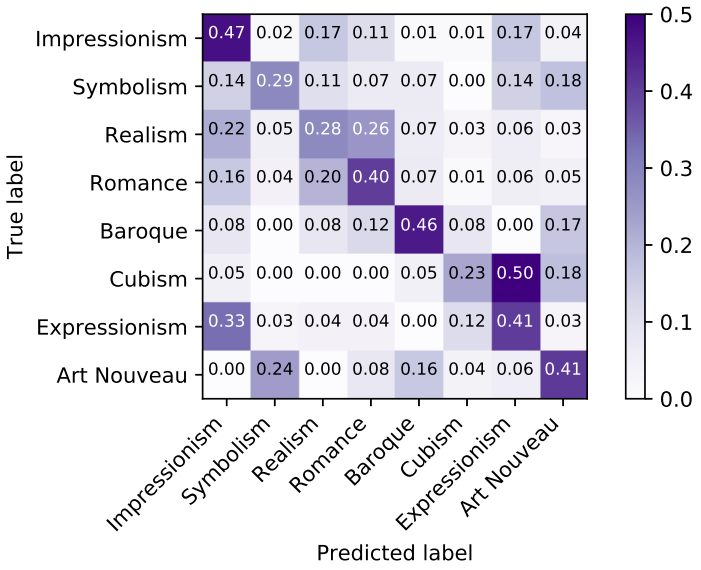

Figure 17: The CNNs classification result on the *extra2* dataset

5.5.6. Comparison with Other Works

Almost all of the works focusing on the task of style recognition, used an existing dataset. Creating a new dataset might not seem like a big task, but takes a lot of time and its performance is very hard to measure. Furthermore, the resulting datasets used in this thesis are a lot smaller than the ones used in the related work, leaving less classes and also less images per class. This gives less training data for each class and might result in a worse performance. However it also raises the general performance, as there are less classes to choose from. For this reason and the limited experience that is available to us, the classification techniques are rather simple compared to related works. We did not use any combination of different classifiers or use boosting metrics, but instead kept a broader scope. Although, ANNs are mentioned all the time, almost none of the related works use them. Hence why we wanted to try using them. In fact, none of the works we could find retrained a CNN and trained a classical ML classifier. When using an ANN for feature extraction all of the related works either use *Decaf* or its successor *Caffe*. We chose a newer framework, TensorFlow, including a more recent network architecture.

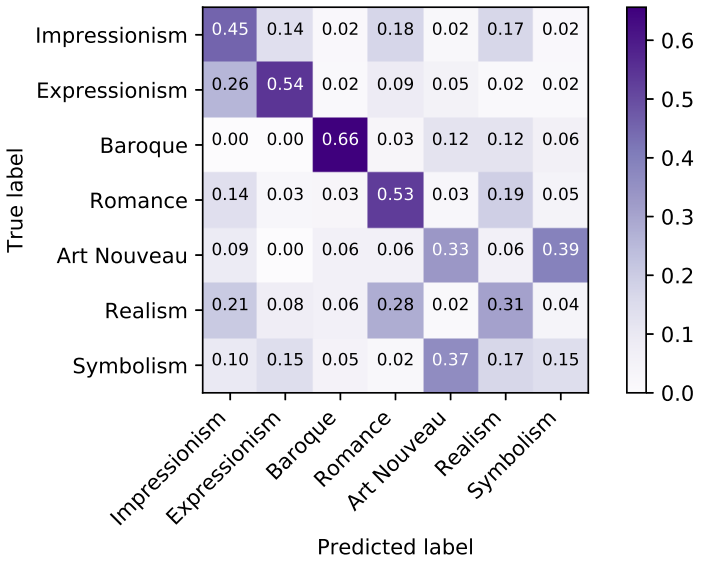

Figure 18: The CNNs classification result on the *extraVar* dataset

(a) A cubism image, misclassified as realism. “Window at Vers” by André Derain (1912).

(b) A cubism image, misclassified as art-nouveau. “Fruit And Flowers” by Maurice de Vlaminck (1910).

(c) A cubism image, misclassified as expressionism. “Jews In Snow” by M. H. Maxy (1943).

Figure 19: Three randomly selected cubism images that got misclassified by the *wikiartSel* classifier.

(a) An art-nouveau image, misclassified as romantic. “Capa "ABC - Revista Portugueza", Nº 162, Ano IV De 23 De Agosto” by Emmerico Nunes (1923).

(b) An art-nouveau image, misclassified as impressionism. “Fontanka” by Boris Kustodiev (1916).

(c) An art-nouveau image, misclassified as romantic. “Merchant Wedding” by Boris Kustodiev (1917).

Figure 20: Three randomly selected art-nouveau images that got misclassified by the *wikiart-Sel* classifier.

(a) A baroque image, misclassified as romance. “Christus Am Ölberg” by Heinrich Schonfeld.

(b) A baroque image, misclassified as symbolism. “Peeckelhaering” by Frans Hals.

(c) A baroque image, misclassified as impressionism. “View of the Groote Kerk in Dordrecht from the River Maas” by Aelbert Cuyp (1647–48).

Figure 21: Three randomly selected baroque images that got misclassified by the *wikiartSel* classifier.

6. Summary and Outlook

6.1. Summary

The first part of the thesis gave an overview over the basics of machine learning and deep learning. We extracted multiple datasets from the ARTigo image tagging website and then evaluated different learning approaches on the data, both based on deep learning. In the first approach, we used a CNN for feature extraction on the generated datasets. Then, a SVM was trained on the data. The second approach took the same trained CNN and retrained it to the new classes.

When first confronted with ML, we were wondering why most of the related work focused on shallow learning to train a classifier and used sophisticated techniques to generate features from the datasets. Now, after some time working with the subject, it seems like its just hard to work with deep learning. Especially creating a custom neural network architecture needs a lot of experience. It takes a lot more time to train a network and sometimes it is not easy to predict the outcome. It is easier to combine multiple classifiers and more promising when working with smaller datasets. We also feel that working with traditional ML algorithms is a lot better documented and provides an easier learning experience. In the end, the best performing classifier was the one trained on the *wikiartSel* dataset. It was used for a small application, which predicts and outputs styles for a list of images. This application can then be run on the ARTigo database. It is unfortunate that the extracted datasets did not perform as well as the Wikiart database. But, when comparing the datasets and keeping in mind that one was mostly annotated by professionals and the other by volunteers, the performance is very good. However, when aiming for a high performing classifier, with as many classes as possible, the best approach would probably be to use the whole Wikiart data.

6.2. Outlook

Like some before us, we showed that features extracted using a CNNs trained on object recognition are usable for tasks beside object recognition. Tough it would be interesting if CNNs trained on other tasks perform similar or even better. When retraining the Inception architecture, we used the "vanilla" version of gradient descent. There are many more variations that might be provide better results. Tensorflow was used as a deep learning library. It performed well and provided sufficient features for our tasks. However, due to its rapid growth sometimes had problems with its API. Also, it seems focused on research and is therefore very low-level. An alternative would be the library *Karas*[46], witch takes a more practical approach. Among others, it provides Tensorflow as a backend. Other than the chosen library, there is much space for improvement to the work done in this thesis.

Most importantly, a better method for dataset creation that creates a clearer separation between classes will probably create better results. Section 5.5.4 proposes a few possible approaches. As mentioned in Section 6.1 the best performing classifier for real life applications, would probably be one that was trained on the full Wikiart database. It might

[46] `http://keras.io`

be interesting to see how a CNN would perform when given so much more data. When handling the data imbalance problem, the classical ML approach did not produce satisfying results and should be further investigated. Either by experimenting with different weight distributions or by choosing a different classification technique (e.g. random forest). Also promising would be the extraction of multiple features and using feature fusion to create better features. Additionally, it is possible to train multiple classifiers and use boosting or other ranking techniques to combine their output (see Section 2.1). Lastly, the hyper parameter optimisation could be optimised. There are some approaches like random grid search that might produce interesting results.

A. Appendix

All of the following data refers to one of the extracted datasets. All of the datasets, the classification application and further files that are related to this thesis are available on the DVD attached to the thesis. The classification application, which uses the trained classifier takes a folder as an argument and outputs a csv file with the results. Because it is not a big application, there is no section discussing it. However, the application does contain some help when running it.

A.1. Style- and Similarity Tags

Style Name	Similarity Tags
impressionism	MONET, IMPRESSIONISTISCH, PASTOS, DUKTUS, ÖL
realism	BRAUN, NATURALISMUS, ÖL, GEMÄLDE, GENRE, MALEREI
romance	ROMANTISCH, NATUR, LANDSCHAFT, HIMMEL, SONNE, LICHT, SONNENUNTERGANG
expressionism	BLAUER REITER, MARC, FRANZ MARC, BUNT
Post-impressionism	-
Surrealism	-
Jugendstil	KLIMT , SPHINX , ORNAMENT , ORNAMENTIK , MORGEN, RECHTECK , FLORAL , GÖTTIN , DURCHSICHTIG
baroque	ROKOKO, ADEL
symbolism	ANHÄNGER
Abstrakter expressionism	-
Naive Kunst	ADAM, PARADIES, EVA, TIGER, APFEL, PFAU, SCHLANGE
Primitivismus	GAUGIN, SÜDSEE, GAUGUIN
Neoklassizistismus	-
Rokkoko	ROKOKO, ROCAILLE
Nördliche Renaissance	-
cubism	KUBISTISCH, FORMEN, FUTURISMUS, QUADRAT, ABSTRAKT, ECKIG, ECKEN, FORM, GEOMETRIE, GEOMETRISCH
Minimalismus	BEUTEL, REGENSCHIRM, SCHUH
Pop Art	-
Informel	EXPRESSIV
Abstrakte Kunst	DURCHEINANDER, ABSTRAKT, FLÄCHEN, CHAOS
Farbfeldmalerei	-
Ukiyo-e	-
Manierismus	-
Spätrenaissance	-
Frührenaissance	-
Hochrenaissance	-
Konzeptuelle Kunst	-
Magischer realism	-
Op art	-
Neoexpressionism	-
Lyrische Abstraktion	-
Akademische Kunst	-

Table 9: The 33 style names and similarity tags used to query the ARTigo database for potential images. The styles and the similarity tags are not translated, but an exact copy of the file used for the data gathering. The styles names have been collected using the Wikiart Database. Similarity tags are extracted using the ARTigo Analytics Center's neighbourhood analysis.

A.2. Class Distribution for the Datasets

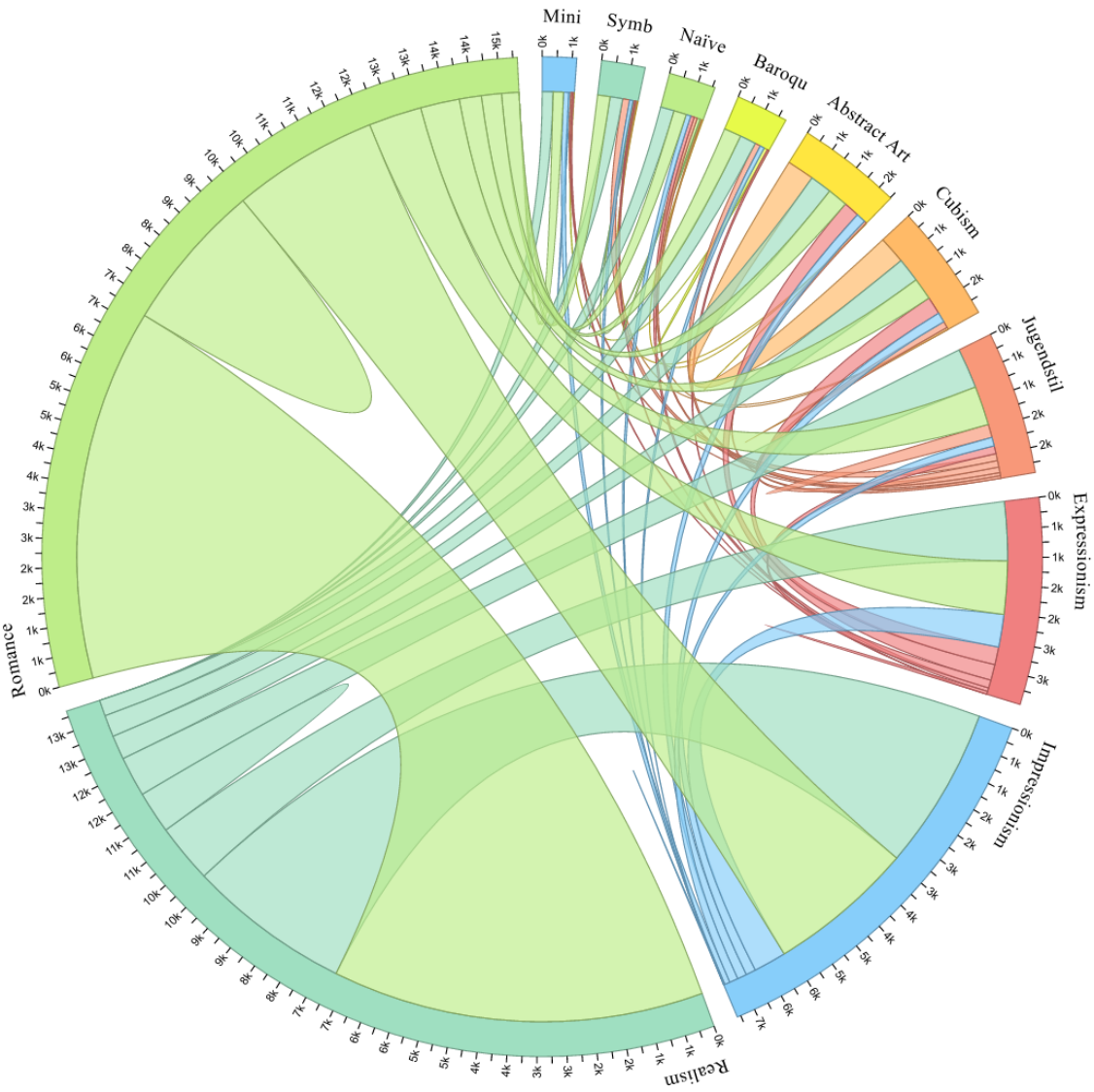

Figure 22: A chord diagram for the *extraAll* dataset.

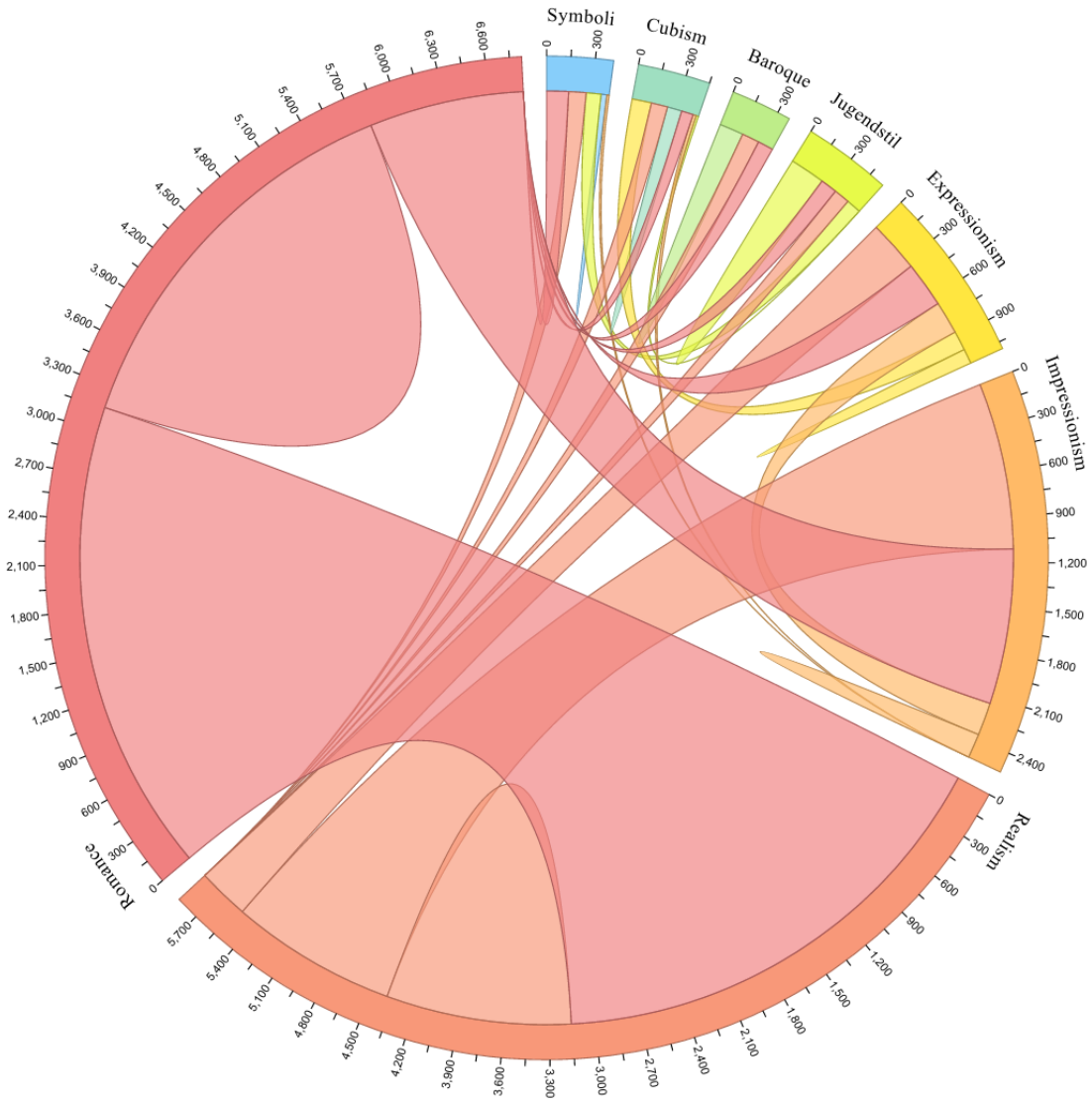

Figure 23: A chord diagram for the *extra2* dataset.

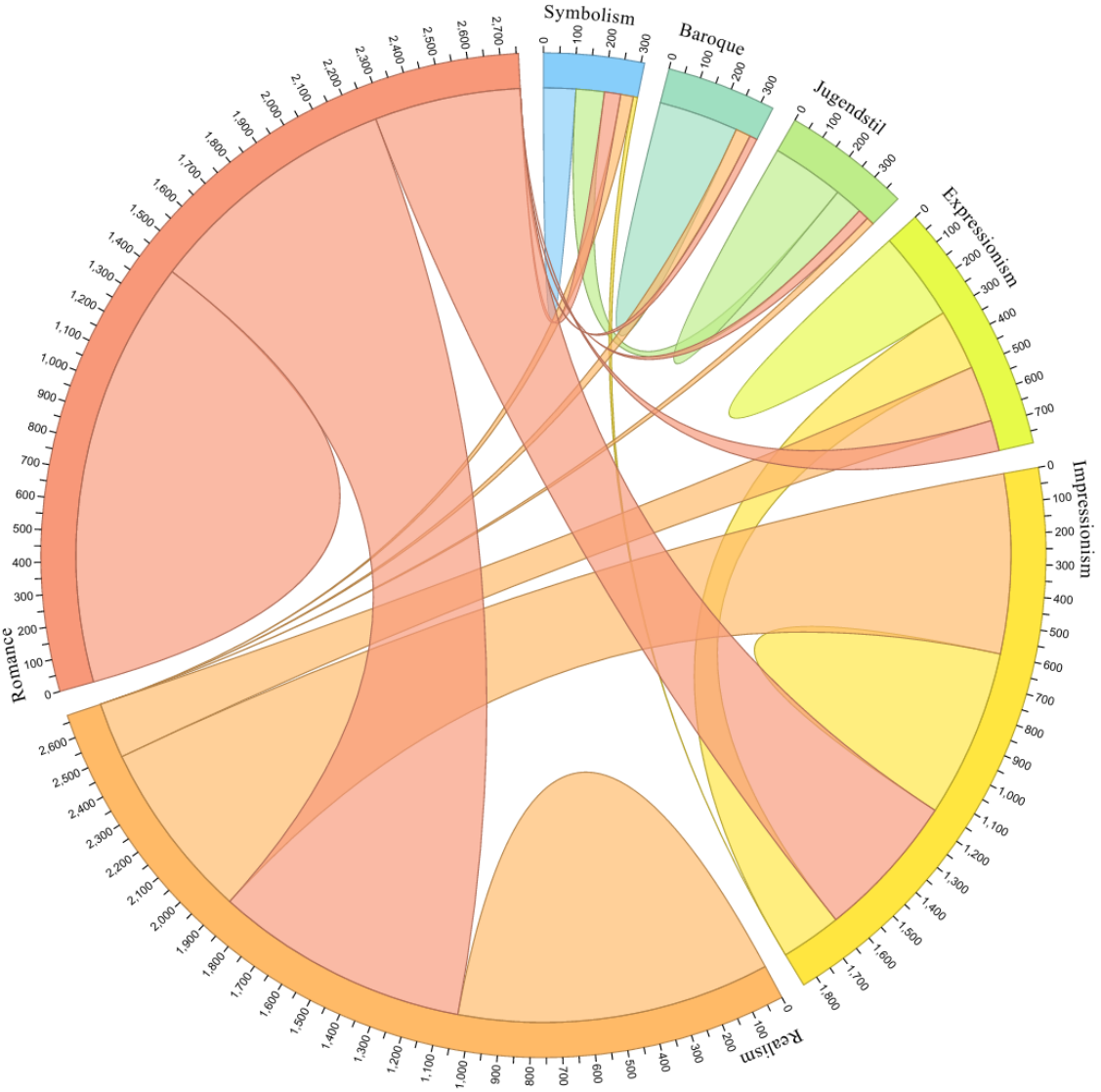

Figure 24: A chord diagram for the *extraVar* dataset.

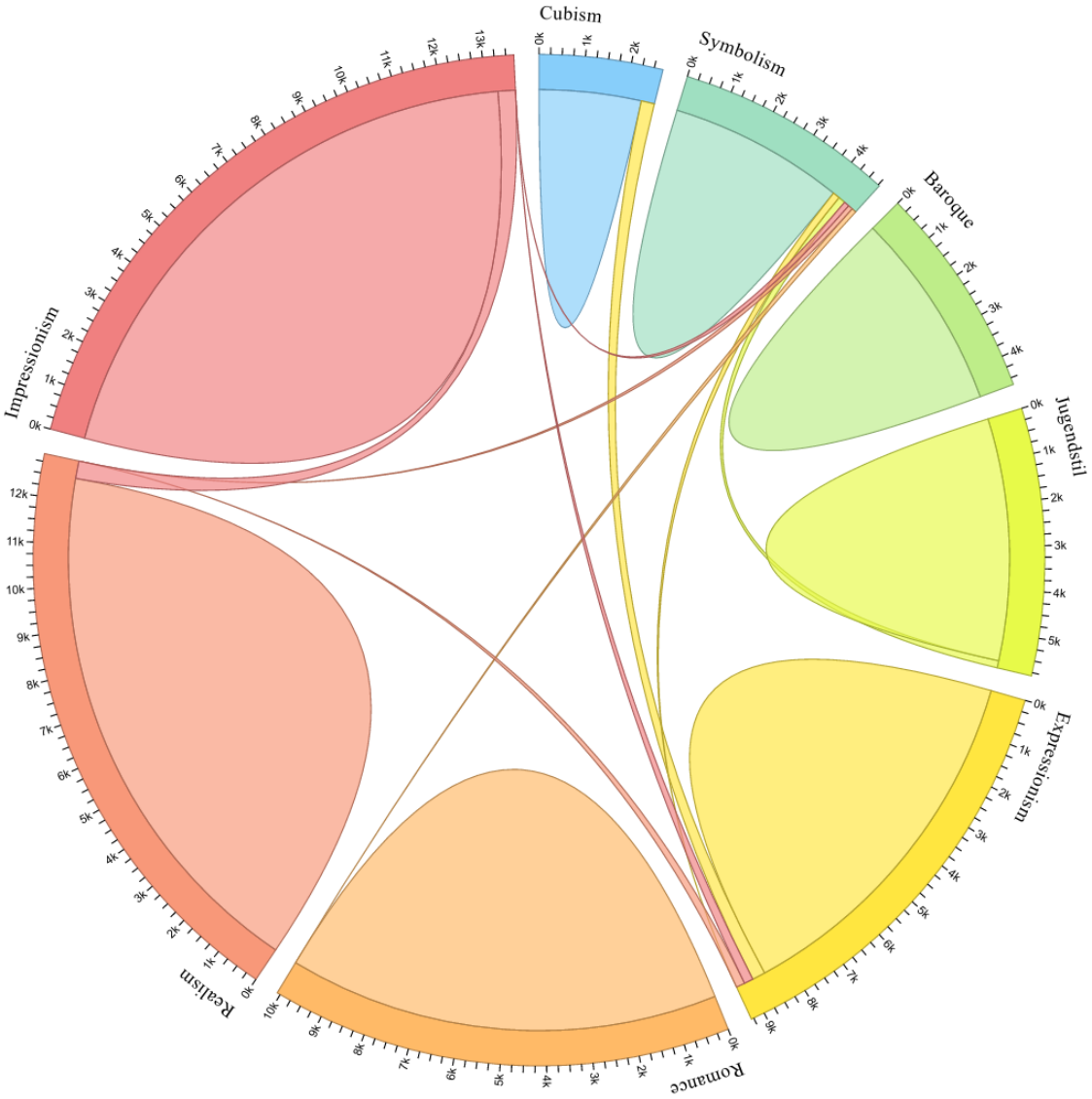

Figure 25: A chord diagram for the *wikiart* dataset.

A.3. Scoring Metrics for the Datasets

	Precision	Recall	F1-score	Support
Baroque	0.82	0.60	0.69	30
Impressionism	0.55	0.91	0.68	140
Romance	0.81	0.76	0.78	91
Realism	1.00	0.03	0.07	29
Art Nouveau	0.35	0.38	0.37	34
Expressionism	0.79	0.35	0.49	65
Symbolism	0.43	0.13	0.20	23
avg / total	0.67	0.62	0.58	412

Table 10: Scoring metrics for the *rawStyles* dataset. The classifier was trained using a SVM.

	Precision	Recall	F1-score	Support
Baroque	0.62	0.42	0.50	24
Impressionism	0.39	0.34	0.36	164
Romance	0.51	0.75	0.61	257
Realism	0.48	0.39	0.43	262
Art Nouveau	0.43	0.29	0.34	35
Expressionism	0.57	0.43	0.49	54
Symbolism	0.00	0.00	0.00	18
avg / total	0.47	0.49	0.47	814

Table 11: Scoring metrics for the *extraVar* dataset. The classifier was trained using a SVM.

	Precision	Recall	F1-score	Support
Baroque	0.80	0.10	0.18	40
Impressionism	0.50	0.01	0.01	154
Cubism	0.33	0.24	0.28	25
Romance	0.53	0.86	0.65	610
Realism	0.50	0.43	0.46	514
Art Nouveau	0.82	0.16	0.26	57
Expressionism	0.39	0.11	0.17	64
Symbolism	0.00	0.00	0.00	33
avg / total	0.51	0.52	0.45	1497

Table 12: Scoring metrics for the *extra2* dataset. The classifier was trained using a SVM.

	Precision	Recall	F1-score	Support
Impressionism	0.25	0.47	0.33	175
Symbolism	0.10	0.29	0.15	28
Realism	0.47	0.28	0.35	515
Romance	0.61	0.40	0.49	641
Baroque	0.10	0.46	0.17	24
Cubism	0.12	0.23	0.16	22
Expressionism	0.19	0.41	0.26	69
Art Nouveau	0.24	0.41	0.30	49
avg / total	0.47	0.37	0.39	1523

Table 13: Scoring metrics for the *extra2* dataset. The classifier was trained using retraining of the Inception network.

	Precision	Recall	F1-score	Support
Impressionism	0.41	0.45	0.43	165
Expressionism	0.36	0.54	0.43	57
Baroque	0.40	0.66	0.50	32
Romance	0.55	0.53	0.54	252
Art Nouveau	0.23	0.33	0.27	33
Realism	0.46	0.31	0.37	248
Symbolism	0.13	0.15	0.14	41
avg / total	0.44	0.43	0.43	828

Table 14: Scoring metrics for the *extraVar* dataset. The classifier was trained using retraining of the Inception network.

	Precision	Recall	F1-score	Support
Symbolism	0.23	0.23	0.23	31
Art Nouveau	0.33	0.43	0.38	44
Romance	0.72	0.69	0.71	88
Impressionism	0.75	0.53	0.62	140
Realism	0.28	0.48	0.35	23
Baroque	0.51	0.67	0.58	33
Expressionism	0.53	0.60	0.56	52
avg / total	0.59	0.55	0.56	411

Table 15: Scoring metrics for the *rawStyles* dataset. The classifier was trained using retraining of the Inception network.

	Precision	Recall	F1-score	Support
Expressionism	0.49	0.42	0.46	217
Impressionism	0.59	0.66	0.62	189
Baroque	0.74	0.63	0.68	211
Romance	0.49	0.50	0.49	186
Art Nouveau	0.53	0.68	0.60	203
Symbolism	0.58	0.39	0.47	203
Cubism	0.77	0.82	0.79	191
Realism	0.40	0.47	0.44	192
avg / total	0.58	0.57	0.57	1592

Table 16: Scoring metrics for the *wikiArtSel* dataset. The classifier was trained using re-training of the Inception network.

A.4. Misclassified Images For The *wikiartSel* Dataset

True class	Predicted Class	Filename
cubism	expressionism	washtable-1926.jpg
cubism	expressionism	two-friends-1959.jpg
cubism	expressionism	nude-1919(1).jpg
cubism	realism	still-life-with-jug-1919.jpg
cubism	expressionism	lady-in-pink-artist-s-sister-anna-rozanova.jpg
cubism	art nouveau	rajasthani-women-1960.jpg
cubism	expressionism	figs.jpg
cubism	expressionism	tomatoes.jpg
cubism	expressionism	spotty-house-1936.jpg
cubism	expressionism	nude-female-figure-shown-from-the-back.jpg
cubism	art nouveau	self-portrait-1960.jpg
cubism	art nouveau	the-dining-table-1912.jpg
cubism	expressionism	imaginative-composition-the-tent-1923.jpg
cubism	expressionism	the-wounded-bird.jpg
cubism	expressionism	the-face-face-and-hands.jpg
cubism	expressionism	moving-away-1985.jpg
cubism	expressionism	the-dancer-1919.jpg
cubism	expressionism	mannequin-barcelona-mannequin-1927.jpg
cubism	expressionism	la-raffinerie-de-p-trole-1961.jpg
cubism	art nouveau	the-hunt-design-for-fabric.jpg
cubism	expressionism	still-life-composition-1948.jpg
cubism	expressionism	pasiphae-2.jpg
cubism	expressionism	two-naked-figures.jpg
cubism	expressionism	jews-in-snow-1943.jpg
cubism	expressionism	composition-1.jpg
cubism	art nouveau	window-at-vers-1912.jpg
cubism	expressionism	fishing-design-for-fabric.jpg
cubism	symbolism	flight-to-the-egypt-1911.jpg
cubism	symbolism	landscape-at-toledo-1913.jpg
cubism	expressionism	mulatas-1928.jpg
cubism	expressionism	man-s-head.jpg
cubism	expressionism	akt-mit-stuhl-und-rotem-tuch-1930.jpg
cubism	expressionism	two-women-seated-by-a-window-1914.jpg
cubism	expressionism	snow-covered-church-1927.jpg
cubism	impressionism	kuntsevo-yellow-cottage-1919.jpg
cubism	expressionism	female-bust.jpg
cubism	expressionism	a-landscape-drawn-into-squares.jpg
cubism	expressionism	untitled-portrait-of-ibn-zainab-1979(1).jpg
cubism	realism	bella-with-white-collar-1917.jpg
cubism	art nouveau	shrine-homage-to-c-zanne-1963.jpg
cubism	expressionism	two-figures.jpg
cubism	art nouveau	a-dream-1932.jpg

cubism	expressionism	bull-plate-iv-1945.jpg
cubism	art nouveau	hommage-picasso-1973.jpg
cubism	realism	fruit-and-flowers.jpg
cubism	art nouveau	portrait-of-a-spanish-dancer.jpg
cubism	expressionism	still-life-with-pear-n-1-1926.jpg

Table 17: A list of all misclassified images of the cubism class class. Filenames refer to the *wikiartSel* dataset.

True Class	**Predicted Class**	**Filename**
baroque	romance	francis-xavier-1670.jpg
baroque	realism	bridge-1645.jpg
baroque	romance	chancelor-g-i-golovkin.jpg
baroque	romance	mattheus-van-den-broucke-councillor-of-the-indies-1678.jpg
baroque	art nouveau	democritus-1660.jpg
baroque	romance	piet-1625.jpg
baroque	romance	town-with-four-towers.jpg
baroque	art nouveau	icon-savior-nerukotvornyi-saviour-not-made-by-hands-from-the-zhovkv iconostasis-1699.jpg
baroque	expressionism	old-woman-with-a-hen.jpg
baroque	art nouveau	portrait-of-catherine-ii-1.jpg
baroque	romance	christus-am-lberg.jpg
baroque	romance	arthur-blayney.jpg
baroque	romance	pan-and-syrinx-1619.jpg
baroque	romance	prayer-before-the-meal.jpg
baroque	symbolism	head-of-woman-1.jpg
baroque	realism	portrait-of-a-condottiero-1622.jpg
baroque	art nouveau	kitchen-scene.jpg
baroque	symbolism	peeckelhaering-detail-1643.jpg
baroque	romance	view-of-bracciano-1620.jpg
baroque	romance	pasture-with-two-sleeping-shepherdesses.jpg
baroque	romance	a-merry-company-in-an-arbor.jpg
baroque	symbolism	maurice-1567-1625-prince-of-orange-lying-in-state.jpg
baroque	art nouveau	icon-the-descent-of-the-holy-spirit-from-the-bilostok-monastery-iconostasis-early-18th-century.jpg
baroque	realism	portrait-of-annibale-ludovico-and-agostino-carracci.jpg
baroque	romance	adoration-of-the-shepherds-1660.jpg
baroque	realism	saint-didacus-of-alcalá-in-ecstasy-before-the-cross-1646.jpg
baroque	romance	charitable-samaritan-also-known-as-the-good-samaritan-1638.jpg
baroque	realism	old-soldier.jpg
baroque	romance	boy-playing-a-violin.jpg
baroque	realism	self-portrait-in-a-soft-hat-and-embroidered-cloak-1631.jpg
baroque	expressionism	icon-of-archangel-gabriel-1699.jpg
baroque	realism	a-woman-with-a-baby-in-her-lap-and-a-small-child-1658.jpg
baroque	symbolism	music-making-angels.jpg

baroque	romance	a-dutch-family-group-1650.jpg
baroque	romance	antony-and-cleopatra.jpg
baroque	impressionism	view-of-the-groote-kerk-in-dordrecht-from-the-river-maas.jpg
baroque	romance	the-abduction-of-ganymede-1635.jpg
baroque	romance	dutch-ships-ramming-spanish-galleys-off-the-flemish-coast-in-october-1602-1617.jpg
baroque	symbolism	the-glory-of-st-dominic-1613.jpg
baroque	romance	mountainous-landscape-with-traveller.jpg
baroque	symbolism	the-triumphal-car-of-kallo-sketch.jpg
baroque	romance	david-and-jonathan-1642.jpg
baroque	symbolism	a-pelican-and-other-birds-near-a-pool-the-floating-feather-1680.jpg
baroque	romance	winter-landscape-with-farmhouse-1624.jpg
baroque	romance	portrait-of-peter-the-great.jpg
baroque	symbolism	portrait-of-old-woman-1630.jpg
baroque	romance	an-extensive-landscape.jpg
baroque	romance	the-age-of-iron-1641.jpg

Table 18: A list of all misclassified images of the baroque class. Filenames refer to the *wikiartSel* dataset.

True Class	**Predicted Class**	**Filename**
art nouveau	impressionism	agony-in-the-garden.jpg
art nouveau	symbolism	at-that-moment-she-was-changed-by-magic-to-a-wonderful-little-fairy-1907.jpg
art nouveau	symbolism	entry-of-tent-of-ivan-the-terrible.jpg
art nouveau	romance	portrait-of-arman-frantsevich-aziber-and-his-son-1915.jpg
art nouveau	romance	portrait-of-a-rich-peasant.jpg
art nouveau	expressionism	girls-with-purple-surrounds-1900-6.jpg
art nouveau	baroque	la-maternelle-1920.jpg
art nouveau	symbolism	north-shore-lake-superior-1926.jpg
art nouveau	cubism	capa-abc-revista-portugueza-n-162-ano-iv-de-23-de-agosto-1923.jpg
art nouveau	realism	sketch-of-costumes-for-tale-of-tsar-saltan-1919-1.jpg
art nouveau	romance	summer-morning-1.jpg
art nouveau	expressionism	portrait-of-a-nurse-1907.jpg
art nouveau	impressionism	houses-with-laundry-seeburg-1914.jpg
art nouveau	expressionism	cacao-lhara-1890.jpg
art nouveau	baroque	odin-and-sleipnir-1911.jpg
art nouveau	symbolism	the-beothic-at-bache-post-ellesmere-island-1928.jpg
art nouveau	romance	portrait-of-olga-konstantinovna-lancere-1910.jpg
art nouveau	symbolism	putivl-1920.jpg
art nouveau	baroque	when-her-majesty-wants-to-know-the-time.jpg
art nouveau	impressionism	chruch-in-cassone.jpg
art nouveau	realism	student-appealing-his-tunic-1909.jpg
art nouveau	symbolism	music.jpg
art nouveau	realism	saluons-les-1899-1.jpg

art nouveau	impressionism	haymaking.jpg
art nouveau	symbolism	judith-and-holofernes-1916.jpg
art nouveau	romance	christ-and-the-samaritan-woman.jpg
art nouveau	romance	polish-hector.jpg
art nouveau	symbolism	mountaisn-and-lake-1929.jpg
art nouveau	realism	chair-design-1.jpg
art nouveau	cubism	they-pulled-up-the-princesses.jpg
art nouveau	symbolism	view-of-the-alcazar-segovia.jpg
art nouveau	symbolism	design-for-a-theater-curtain.jpg
art nouveau	romance	unknown.jpg
art nouveau	realism	boyarin-and-boyarynia-1921.jpg
art nouveau	baroque	it-s-i-your-uncle-scrooge-i-have-come-to-dinner-will-you-let-me-in-fred.jpg
art nouveau	romance	so-the-four-brothers-took-their-sticks-in-their-hands-bade-their-father-good-bye-and-passed-out.jpg
art nouveau	expressionism	smaragda-berg-1906.jpg
art nouveau	cubism	savage-tribes-no-3.jpg
art nouveau	romance	autumn-festivities-1922.jpg
art nouveau	symbolism	bob-cratchit-went-down-a-slide-on-cornhill.jpg
art nouveau	symbolism	portrait-of-nadezhda-zabela-vrubel-1904.jpg
art nouveau	symbolism	el-quetzal-1916.jpg
art nouveau	realism	farmhouse-in-upper-austria-1912.jpg
art nouveau	realism	hearth-1902.jpg
art nouveau	impressionism	untitled.jpg
art nouveau	symbolism	shadows.jpg
art nouveau	romance	merchant-wedding-1917.jpg
art nouveau	expressionism	sweet-smoke-1913.jpg
art nouveau	realism	portrait-of-georges-navazza-1916.jpg
art nouveau	impressionism	the-golden-age-1916.jpg
art nouveau	impressionism	st-isaac-s-cathedral-on-a-foggy-day.jpg
art nouveau	expressionism	cabin-1912-1.jpg
art nouveau	symbolism	french-embankment.jpg
art nouveau	symbolism	portrait-of-henryka-cohn-1908.jpg
art nouveau	romance	old-town-1902.jpg
art nouveau	impressionism	fontanka-1916.jpg
art nouveau	symbolism	the-fairy-circus-1931-9.jpg
art nouveau	expressionism	my-friends-the-carpenter-and-the-painter-1909(1).jpg
art nouveau	impressionism	twilight-baie-saint-paul-1924.jpg
art nouveau	symbolism	viking-s-tomb-1908.jpg
art nouveau	symbolism	ondine.jpg
art nouveau	expressionism	boys-in-sailor-s-striped-vests-1919.jpg
art nouveau	symbolism	joukahainen-s-revenge.jpg
art nouveau	baroque	une-soiree-au-chat-noir.jpg
art nouveau	symbolism	the-year-s-at-the-spring-1920-16.jpg
art nouveau	impressionism	kovno-old-church-1903.jpg
art nouveau	expressionism	study-to-conquest-of-kazan.jpg

art nouveau	realism	die-s-ngerin-oberl-nder-spring-1907-1907.jpg
art nouveau	romance	triton-1914(1).jpg
art nouveau	expressionism	peasant-interior-in-winter-1890(1).jpg
art nouveau	romance	portrait-of-a-ballerina-a-d-danilova-in-costume-for-the-ballet-armida-s-pavilion-1922.jpg
art nouveau	impressionism	portrait-of-maria-akimova-1908.jpg
art nouveau	expressionism	selma-lagerl-f-1908(1).jpg
art nouveau	realism	in-front-of-the-swing-mirror.jpg
art nouveau	expressionism	miss-loie-fuller.jpg
art nouveau	romance	triple-in-uglich-1913.jpg
art nouveau	symbolism	bland-tomtar-och-troll-1913.jpg
art nouveau	expressionism	the-slippers-of-cinderella-1894.jpg
art nouveau	realism	boy-fellah-1923(1).jpg
art nouveau	cubism	operatic-costume-designs-1911.jpg
art nouveau	expressionism	bath-1913.jpg
art nouveau	expressionism	the-portal-n-vitoria-fifties.jpg
art nouveau	baroque	svipdag-speaks-with-thokk-1911.jpg
art nouveau	expressionism	windmill.jpg
art nouveau	impressionism	self-portrait-1907.jpg
art nouveau	romance	garden-landscape-and-fountain-1915.jpg

Table 19: A list of all misclassified images of the art nouveau class. Filenames refer to the *wikiartSel* dataset.

References

[AKPP15] Siddharth Agarwal, Harish Karnick, Nirmal Pant, and Urvesh Patel. Genre and style based painting classification. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 588–594. IEEE, 2015.

[ASWdF16] Yannis M. Assael, Brendan Shillingford, Shimon Whiteson, and Nando de Freitas. LipNet: Sentence-level Lipreading. *CoRR*, abs/1611.01599, 2016.

[BB12] James Bergstra and Yoshua Bengio. Random Search for Hyper-parameter Optimization. *J. Mach. Learn. Res.*, 13:281–305, February 2012.

[BBC14] BBC. Stephen Hawking warns artificial intelligence could end mankind. `http://www.bbc.com/news/technology-30290540`, 2014. Accessed: 2017-03-17.

[BLW14] Yaniv Bar, Noga Levy, and Lior Wolf. Classification of artistic styles using binarized features derived from a deep neural network. In *Workshop at the European Conference on Computer Vision*, pages 71–84. Springer, 2014.

[Bos] Mike Bostock. A Chord Diagram implementation in D3. `https://bl.ocks.org/mbostock/4062006`. Accessed: 2017-03-03.

[BRSS15] Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, and Mohak Shah. Comparative Study of Caffe, Neon, Theano, and Torch for Deep Learning. *CoRR*, abs/1511.06435, 2015.

[CBHK02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[Chi13] Eric Chio. Class Imbalance Problem. `http://www.chioka.in/class-imbalance-problem/`, 2013. Accessed: 2017-04-07.

[CJLV16] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE, 2016.

[cs2] Stanford Computer Sience Class Convolutional Neural Networks for Visual Recognition. `http://cs231n.github.io/`. Accessed: 2017-03-03.

[GEB15] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A Neural Algorithm of Artistic Style. *CoRR*, abs/1508.06576, 2015.

[Gim17] Gimp. Gimp Documentation Convolution Matrix. `https://docs.gimp.org/en/plug-in-convmatrix.html`, 2017. Accessed: 2017-03-15.

[HTF01a] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning : Data mining, inference, and prediction.* Springer, New York, 2001.

[HTF01b] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of*

Statistical Learning. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[kar] What I learned from competing against a ConvNet on ImageNet . http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/. Accessed: 2017-03-03.

[Kri07] David Kriesel. *A Brief Introduction to Neural Networks*. 2007.

[KSL15] Corey Kereliuk, Bob L Sturm, and Jan Larsen. Deep learning and music adversaries. *IEEE Transactions on Multimedia*, 17(11):2059–2071, 2015.

[KTH+13] Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. Recognizing image style. *arXiv preprint arXiv:1311.3715*, 2013.

[LW] Weiyang Liu and Yandong Wen. Large-Margin Softmax Loss for Convolutional Neural Networks.

[MS03] Markos Markou and Sameer Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[Nie] Michael Nielsen. *Neural Networks and Deep Learning*. Accessed: 2017-03-15.

[P+99] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. 1999.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[Pow11] David Martin Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.

[RDC+04] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are Loss Functions All the Same? *Neural Comput.*, 16(5):1063–1076, May 2004.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[Sch17] Google Scholar. Recognizing image style Citations. https://scholar.google.de/scholar?cites=1897197234583669199, 2017. Accessed: 17.02.2017.

[SE15] Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505.00855*, 2015.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In F. Pereira, C. J. C. Burges,

L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.

[SMB10] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks*, pages 92–101. Springer, 2010.

[SS02] B. Schölkopf and AJ. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, December 2002.

[SVI+15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. *CoRR*, abs/1512.00567, 2015.

[SWS+00] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[Ten17] TensorFlow. Inception in TensorFlow Github repository. `https://github.com/tensorflow/models/tree/master/inception`, 2017. Accessed: 2017-03-15.

[Tra16] Kenneth Tran. Evaluation of Deep Learning Toolkits. `EvaluationofDeepLearningToolkits`, 2016. [Online; accessed 15-December-2016].

[TSN] scikit-learn t-distributed Stochastic Neighbor Embedding documentation. `http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`. Accessed: 2017-03-03.

[vdMH08] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[Xia15] Yinghui Xia. Fine-tuning for Image Style Recognition, 2015.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[ZZN+15] Qi-Feng Zhou, Hao Zhou, Yong-Peng Ning, Fan Yang, and Tao Li. Two approaches for novelty detection using random forest. *Expert Systems with Applications*, 42(10):4840–4850, 2015.