Deep Learning for Computer Vision in the Art Domain: Proceedings of the Master Seminar on Practical Introduction to Deep Learning for Computer Vision, HPI WS 20/21

Christian Bartz, Ralf Krestel

Technische Berichte Nr. 139

des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam | 139

Christian Bartz | Ralf Krestel

Deep Learning for Computer Vision in the Art Domain

Proceedings of the Master Seminar on Practical Introduction to
Deep Learning for Computer Vision, HPI WS 20/21

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

Universitätsverlag Potsdam 2021
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 2533 / Fax: 2292
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Online veröffentlicht auf dem Publikationsserver der Universität Potsdam
https://doi.org/10.25932/publishup-51290
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-512906

Zugleich gedruckt erschienen im Universitätsverlag Potsdam:
ISBN 978-3-86956-514-9

Preface

This technical report summarizes the results of a Master seminar on *Practical Introduction to Deep Learning for Computer Vision* at the Hasso Plattner Institute in the winter term 2020/2021. The seminar was organized by Christian Bartz from the Multimedia and Machine Learning Group of the Internet Technologies and Systems chair and Dr. Ralf Krestel, Head of the Web Science Group. Both groups are involved in a joint research project analyzing art-historic data in collaboration with the Wildenstein Plattner Institute.[1]

The goal of the seminar was to give an introduction to current computer vision approaches with a focus on deep learning. We wanted to make the seminar as applied as possible and therefore decided to define three tasks that the students had to solve in the course of the winter term. The tasks were major computer vision tasks: classification, object detection, and image retrieval. As an application domain, we decided to look into fine art paintings with a collection of images provided by the Getty Research Institute.[2] Students were divided into three teams with three students each and each team had to solve each task. For each task, the teams had around one month to come up with a good solution. Each task was then automatically evaluated by us using a hold-out test dataset or manual annotations. The winning team of each task won a small award.

This report consists of the description of the approaches and results achieved by the three teams for each of the three tasks. The first chapter was written by us as an introduction to the tasks and data. We also summarize the results for the three tasks. Finally, we want to thank the participating students for the results and the Getty Research Institute for providing the image collection.

Potsdam, June 16th, 2021

Christian Bartz,
Dr. Ralf Krestel

[1] `https://wpi.art/` (last accessed June 16, 2021).

[2] `https://www.getty.edu/research/` (last accessed June 16, 2021).

Contents

Introducing the Seminar Topics

Christian Bartz and Ralf Krestel

Hasso Plattner Institute for Digital Engineering, University of Potsdam
{christian.bartz,ralf.krestel}@hpi.de

In recent years, computer vision algorithms have seen a rapid development. One of the most important factors is the availability of annotated training data. However, in the art domain annotated data is a scarce resource. The Getty Research Institute provided us with a challenging evaluation dataset, which we used throughout the seminar to evaluate the solutions of our students. In this chapter, we introduce this dataset, discuss the tasks of the students tackled during the seminar, and summarize the results.

1 Introduction

Computer vision algorithms have seen a rapid development in recent years [4, 6, 7]. Much of this development is based on the availability of computing hardware that executes calculations in a massive parallel way, e.g., GPUs. Another reason for this development is the availabilty of large-scale annotated datasets, such as the ImageNet dataset [2]. Research has mainly focused on the analysis of semantics in photographs and videos because no expert knowledge is required for the annotation of everyday objects in images and videos, thus it can be performed cheaply by clickworkers.

While a huge amount of annotations is available for the analysis of everyday objects in photographs, only a small amount of annotations is available for the analysis of works of art. The reasons for this are rooted in two aspects. First, annotating data is a too costly endeavor for most non-profit organizations dealing with digitized copies of works of art. Thus organizations such as the Getty Research Institute[1] are interested in research on possible applications of computer vision algorithms on their scarcely annotated digitized data.

To go forward in this direction, we offered an introductory master seminar that dealt with the application of computer vision methods on a dataset supplied to us by the Getty Research Institute.

Based on a thorough analysis of the available data we identified three possible areas of different difficulty that would set a challenging objective for our students and also benefit the research endeavors of the Getty. During the course of the seminar, the students dealt with the following tasks: (1) Image Classification, i.e., classifying whether a work of art is a drawing or painting and also classifying

[1] https://www.getty.edu/research/ (last accessed June 16, 2021).

the genre of a work of art (section 3). (2) Object Detection, i.e., detecting objects depicted in a work of art (section 4). (3) Image Retrieval, i.e., given a query work of art, return a pre-defined number of similar works of art (section 5). During our work on each of the tasks, we used the resources of the Future SOC lab to train a wide range of deep neural networks on the multi-GPU machines of the Future SOC lab.

2 Dataset

The Getty Research Institute is undertaking a large digitization project in the course of their PhotoTech project.[2] Since they are digitizing a large collection of approximately 700,000 items they are looking for the possibility of automated analysis of the photographed works of art. The goal of the Getty is to find ways that support the researchers in fast metadata generation for faster categorization and search of digitized data. To achieve this they provided us with an excerpt of the PhotoTech collection. The dataset provided by the Getty consists of 7872 scans of works of art. Besides the images, some manually created metadata for the training of machine learning methods is also available. For each image three different kinds of information are available: 1) A categorization whether the image is a painting or a drawing. 2) A categorization which artistic genre the photographed work of art belongs to. For our work, we followed the definition of artistic genres by Cetinic et al. [1] and mapped the provided categories to their corresponding artistic genres. 3) The provided metadata also contains an enumeration of depicted subjects in each image. However, this field does not contain standardized identifiers. Therefore, we could not use it in our analysis.

For each of the tasks, we split the dataset into a dedicated train and test set. For the task of classification (see section 3), no additional annotation from our side for the test set was necessary. For the task of object detection(see section 4), we manually annotated 156 images by adding bounding boxes for objects of each class we wanted to identify. For the task of image retrieval (see section 5), we did not provide any annotated data for testing. Instead, we held back a small number of images and evaluated the performance of each developed retrieval method using a consensus based approach.

3 Image Classification

The first task we tackled was the classification of the provided Photo Tech data into different classes. On the one hand, we developed a set of neural networks for the classification of the *form* (painting vs. drawing) of the work of art. On the

[2] `https://www.getty.edu/research/scholars/research_projects/phototech.html` (last accessed June 16, 2021).

Table 1: Classes of Objects to be Detected in the Images

Persons	Animals	Produce	Structures	Landforms	Faces
Angel	Cow	Flower	Ruins	Water Bodies	Others
Child Jesus	Cat	Fish	Ship	Hills	
Crucifixion	Cock	Meat	Building	Mountains	
Mary	Dog	Fruit	Others	Others	
Nudity	Horse	Others			
Saint Sebastian	Duck				
Others	Others				

other hand, we developed a set of neural networks for the classification of the *genre*, where we follow the definition introduced by Cetinic et al. [1].

Results For our experiments we compared a multitude of network configurations and architectures. We evaluated our models by calculating the average F1-Score for all classes.

For the task of form classification our best model (based on a DenseNet-201) achieves a near perfect classification result with an average F1-Score of 0.9846. Further experiments showed that even using a network with a considerable lower amount of layers (ResNet-50) also achieves comparable results (average F1-Score 0.9844), which hints at the fact that the task itself is not very complex.

For the task of genre classification our best model (based on a DenseNet-201) achieves an average F1-Score of 0.924. The score is considerably lower compared to the form classification task. However, the task of genre classification involves more classes (5 vs. 2) and the dataset is also highly imbalanced across the available images.

4 Object Detection

The second task we tackled was the localization and classification of objects depicted in the artworks. Here, we defined a range of object categories that should be detected in the images. We provide an overview of all categories in Table 1. The categories "persons", "animals", "produce", "structures" and "landforms" serve as high-level categories while the classes belonging to them are finer sub-divisions. The classes "others" are meant as a default fallback: Whenever an object does not fit into any of the subcategories, it is classified as the corresponding supercategory. This can be seen as a two-level classification approach. We distinguish between a *fine* classification (e. g., "cow", "cat", "cock" etc.) and a *coarse* classification (e. g., "animals").

The dataset provided by the Getty does not include any annotations for object locations, hence we manually annotated 156 images for evaluation using the classes defined in Table 1.

Table 2: Average Precision on the Manually Annotated Test Set for Object Detection

Ensemble	Fine Labels		Coarse Labels	
	AP@0.5	AP@0.75	AP@0.5	AP@0.75
YOLO	11.5%	**4.6%**	18%	4.5%
ResNe(X)t and WBF	**12.9%**	3.8%	**22.8%**	**5.9%**
IconArt/pre-trained	4.3%	1.1%	8.5%	1.8%

Results We performed experiments with three different model ensembles, which were mostly trained on non-art data due to the lack of sufficient annotated training data from the art domain. First, we built an ensemble using pre-trained object detection models with one model specifically trained on the IconArt [3] dataset. We denote this ensemble as "IconArt/pre-trained". Second, we built an ensemble consisting of a YOLO v5[3] object detector for detection of models from each category, accompanied by a face detection model introduced by Zhang et al. [8]. We denote this ensemble as "YOLO". Our last ensemble consists of several object detection networks from the Detectron 2[4] toolkit that were adapted to the domain of art analysis by performing art transformations on real photographs and fine-tuning of the models on the adapted images. We denote this ensemble as "ResNe(X)t and WBF".

We evaluated our models by computing the average precision (AP) of the predictions on our fine set of labels and our coarse set of labels, as shown in Table 2. We provide results using detection thresholds of 0.5 and 0.75. These thresholds denote that a prediction is counted as correct if the intersection over union of the ground truth bounding box and the predicted bounding box is larger or equal to the defined threshold. Our results show that it is in general possible to identify objects within works of art by training on non-art images, namely photo collections. However, the performance of the models is very low and hints at the fact that the domain gap from photographs to works of art is too large. We conclude that current methods cannot directly be applied to object detection on works of art, without improved training data or better transfer learning approaches.

5 Image Retrieval

In image retrieval, we investigated the task of returning a set of similar images given a query image. Here, the given Getty dataset also does not include any annotations that could be used for the development or evaluation of an image retrieval system. This is why we did not use an automated approach for the evaluation of this task. We rather specifically selected a number of works of art that we used as

[3] `https://pytorch.org/hub/ultralytics_yolov5/` (last accessed June 16, 2021).
[4] `https://github.com/facebookresearch/detectron2` (last accessed June 16, 2021).

Figure 1: Exemplary result of our best performing image retrieval model. The left-most image is the query image and the following images are the top 3 results in ascending order.

query images and did a manual evaluation of the results returned by the different approaches. We then determined the best approach by calculating the normalized discounted cumulative gain [5] based on the ranking created by our group.

Results After evaluating all developed approaches, we found that a simple approach based on a ResNet-50 trained on the genre classification task (see section 3) performs best. We show an exemplary result obtained by using our best model on an image in Figure 1. In this approach the features before the classification layer are extracted and the distance of the query image to all images in the database is calculated. We also found that adding another training objective that also predicts the mean color of the input image helps to improve the results of the model.

6 Conclusion

In this report, we briefly summarized our experiments on the analysis of works of art, based on an excerpt of the PhotoTech project by the Getty Research Institute. We performed experiments for three different tasks of computer vision, namely image classification, object detection, and image retrieval. While we achieved very good results for the classification tasks using current state of the art methods, this was not the case for the object detection. The main reason was the lack of suitable annotated training data, which current deep neural networks rely on. The good results for image retrieval are based again on the models trained on the classification tasks where sufficiently large training data was available.

References

[1] E. Cetinic and S. Grgic. "Genre classification of paintings". In: *2016 International Symposium ELMAR*. ISSN: 1334-2630. Sept. 2016, pages 201–204. DOI: 10.1109/ELMAR.2016.7731786.

[2] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer*

Vision and Pattern Recognition. June 2009, pages 248–255. DOI: 10.1109/CVPR.2009.5206848.

[3] N. Gonthier, Y. Gousseau, S. Ladjal, and O. Bonfait. "Weakly Supervised Object Detection in Artworks". In: *Proceedings of the European Conference on Computer Vision (ECCV) 2018*.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pages 2672–2680.

[5] K. Järvelin and J. Kekäläinen. "Cumulated gain-based evaluation of IR techniques". In: *ACM Transactions on Information Systems (TOIS)* 20.4 (2002), pages 422–446.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pages 1097–1105.

[7] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28*. Edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pages 91–99.

[8] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE Signal Processing Letters* 23.10 (2016), pages 1499–1503.

Image Classification

Josafat-Mattias Burmeister, Konstantin Dobler, and Nataniel Müller

Hasso Plattner Institute for Digital Engineering, University of Potsdam
{josafat-mattias.burmeister,konstantin.dobler,nataniel.mueller}
@student.hpi.de

Digitizing art collections is a major challenge for many museums and art galleries. To facilitate and accelerate cataloging photos of artworks, we tackle two classification tasks from the art domain using deep learning: type classification and genre classification of artworks. To train our models, we use the popular transfer learning approach. Since our training dataset is highly imbalanced, our work focuses on coping with imbalanced training data. Our results show that the transfer learning approach can produce very good results even for small and highly imbalanced training datasets. We observed that acquiring or generating additional training data as well as certain data augmentation methods can slightly improve training results. Over- and undersampling techniques, on the other hand, do not seem to be necessary and did not provide a substantial benefit.
To optimize performance in both classification tasks, we experiment with multiple training methods and model architectures. In this way, we obtain good results in both tasks: In the type classification task, we achieve an accuracy of over 99% and an F_1-score above 97% for both the minority and majority class. In the genre classification task, we achieve an accuracy of over 96% and F_1-scores ranging from 88% to 99% for the respective classes.

1 Introduction

In recent years, many museums and art galleries have made great efforts to digitize their collections. For example, Amsterdam's Rijksmuseum [36, 44], London's National Gallery [33], Oslo's National Museum [8, 43] and the J. Paul Getty Museum [14] have made significant parts of their art collections available online. Recently, the COVID-19 pandemic has prompted further institutions to digitize their collections [24, 4]. Digitization involves capturing high-resolution images of the artworks [2, 35] and linking them to related information in digital repositories [31]. These repositories can not only facilitate researchers' access to information [42], but can also be used to provide digital content to museum visitors [1]. In addition, digital collections play an important role in marketing and in digital selling of prints and merchandise [1, 44].

However, the creation of digital collections is a major challenge for many institutions. Many museums and galleries own large collections but have limited human and financial resources for digitization [1, 24]. In particular, cataloging

photos and updating existing records is expensive and laborious [1, 31]. During cataloging, photos of artworks are linked to metadata such as author, style, genre, and type of artwork. Automated acquisition of these metadata using computer vision techniques could greatly facilitate and accelerate the digitization of museum collections [29, 6, 32]. A wide range of research has already addressed this issue. Some works rely on the extraction of image features and classify them using shallow machine learning models such as SVMs or kNN classifiers [48, 38, 12]. The image features used can be either low-level features such as color histograms and edge maps [48], or feature maps extracted from convolutional neural networks [38, 6].

In contrast, another part of the existing work uses end-to-end deep learning models [41, 23, 37]. Training deep neural networks to classify artworks is challenging because the available datasets are comparatively small [37] and in some cases unbalanced [47]. Labeling of additional training data is often impractical as it requires expert knowledge from the art domain [38, 7]. Therefore, the transfer learning approach became popular in the art domain [7, 37, 23, 41]. In transfer learning, the deep learning models are first pre-trained on large photo datasets such as ImageNet with a different classification task. Subsequently, some layers of the models are re-trained on the actual training set from the art domain to solve a classification task for artworks [37].

This work applies the transfer learning approach to two classification tasks from the art domain: In the first classification task, artworks are to be classified in terms of their type as a painting or a drawing. To our knowledge, similar classification tasks have been studied only by Sabatelli et al. [37] and Mensink et al. [29] so far. The second task is a genre classification of artworks. Most existing work on genre classification uses only the popular Wikiart dataset [38, 41], although additional datasets with genre annotations are available [13, 16, 27, 34, 9]. In this paper, we make use of several other datasets for training.

For both classification tasks, we systematically evaluate different architectures and training methods. We investigate how different techniques for data augmentation and different approaches for coping with imbalanced data affect model performance. In addition, we study whether model performance can be improved by ensemble learning. Through model optimization, we achieve human-like performance in type classification, with F_1-scores above 97 % for both classes on the test set. In genre classification, we also achieve high performance, with F_1-scores above 88 % for all classes on the test set.

2 Related Work

Since the majority of available fine-art datasets contain metadata on style, artist, genre, technique and material, research efforts have been predominantly focused on style, genre, and artist classification [7]. These classification problems have been addressed through two major methodological categories namely classical approaches and deep learning-based techniques [39]. In classical approaches, im-

age features are extracted and classified using shallow machine learning models. Feature extraction approaches are divided into feature engineering and feature learning methods [32].

Feature engineering approaches. In feature engineering approaches, domain-specific knowledge is used for transforming "low-level" feature sets such as brush strokes and color into meaningful image features [38]: Florea et al. [12] use local and global features in combination with shallow machine learning models like SVM, Random Forest Models and k-Nearest Neighbor to classify artworks in terms of the artistic movement. Other works use texture feature extractors that take into account global color features and composition features to classify artworks based on their genre [48].

Feature learning approaches. With the advancements of CNN-based feature learning approaches, Cetinic et al. investigate the use of features derived from pre-trained CNN layers [6]. Results indicate higher accuracies for "high-level" CNN-based feature sets given the problem of genre classification compared to "low-level" feature sets derived by Scale-Invariant Feature Transform (SIFT) [26] and Histogram of Oriented Gradients (HOG) [10].

Deep learning-based approaches. Lecoutre et al. demonstrate the performance of a residual neural network (ResNet50) and a pre-trained AlexNet for genre classification on the Wikiart paintings dataset, achieving an overall accuracy of more than 62% over 25 classes [23]. Emphasizing the importance of brush stroke in fine-art classification, Huang et al. implement a two-channel deep residual network consisting of a RGB channel and a brush stroke information channel. They achieve a test accuracy of 68.96 % using a pre-trained ResNet50 [18]. Sabatelli et al. compare the performance of transfer learning approaches that solely re-train the decision layer of pre-trained CNNs with approaches that also retrain the convolutional layers [37]. While retraining the convolutional layers is computationally more demanding, it also yields better results.

Tan et al. compare different fine-tuning methods on the Wikiart paintings dataset for style, genre and artist classification. They show that pre-trained CNNs with an additional softmax layer (genre accuracy: 74.14 %) outperform a pre-trained CNN with a 1000 dimensional feature extraction layer compressed by PCA and a SVM trained on top [41]. Zhao et al. tackle the same genre classification problem on the Wikiart paintings dataset with pre-training on ImageNet and random initialization in the last fully connected layer and achieve an accuracy of 78.03% [47]. Cetinica et al. [7] evaluate five different fine-tuning scenarios on a range of datasets including the Wikiart paintings on a CaffeNet architecture [19]: They observe that retraining all except the first two convolutional layers yields the best results with a test accuracy of 77.7 %.

Recently, Mohammadi et al. proposed a hierarchical classification approach, based on clustering the Wikiart paintings dataset styles into 7 super-style parent classes P each containing image style children C [30]. Then a hierarchical ensem-

ble of P+1 parallel CNNs predict parent as well as the child class, improving the average F_1-score of a DenseNet121 compared to a hierarchical DenseNet121 by more than 3%. Focusing on the highly imbalanced class distribution of the Wikiart paintings dataset, Joshi et al.[20] train a semi-supervised Ensemble of Auto-Encoding Transformations (EnAET) model [45]: Instead of pre-training the model, autoencoding transformations are used to train the classifiers in a four-block wide ResNet-28-2. When comparing to ResNet50 models with/without data augmentation and fine-tuned over all layers this approach yields a test accuracy of 82.61% compared to the ResNet50 baseline of 50.1% [20].

3 Datasets and Classification Tasks

In this paper, we aim to solve two distinct classification tasks for artworks. Both tasks are to be solved using deep learning models that receive images of the artworks as input. The first task is to determine the type of an artwork. We distinguish two types of artworks, drawings and paintings. Examples from both classes are shown in Figure 1. In the following, we refer to this binary classification task as "type classification". The second task is a multi-class classification task, where artworks are to be classified in terms of their genre. The term "genre" is used in different meanings in existing works. In this paper, we follow the definition from Cetinic et al. [6] and distinguish between the five classical genres of art: genre painting, history painting, landscape painting, portrait, and still life. Example images for each genre are shown in Figure 2. In the following, we refer to this task as "genre classification".

(a) Drawings

(b) Paintings

Figure 1: Example images from the classes of the type classification task.

Table 1: Datasets Used for the Type Classification Task (not all datasets were used in all experiments)

Dataset	Number of suitable training images		
	Drawings	Paintings	Total
Bing[1]	662	0	662
Brill[2]	185	20	205
Getty	746	4425	5171
Kaggle[3,4]	1,231	2,270	3,501
Metropolitan[5]	1,000	0	1,000
Rijksmuseum[6]	14,223	3,593	17,816
Wikiart[7]	3,943	0	3,943
All datasets	21,990	10,308	32,298

[1] `https://www.microsoft.com/en-us/bing/apis/bing-image-search-api` (last accessed June 16, 2021).

[2] `https://labs.brill.com/ictestset` (last accessed June 16, 2021).

[3] `https://www.kaggle.com/thedownhill/art-images-drawings-painting-sculpture-engraving` (last accessed June 16, 2021).

[4] `https://www.kaggle.com/ikarus777/best-artworks-of-all-time` (last accessed June 16, 2021).

[5] `https://metmuseum.github.io` (last accessed June 16, 2021).

[6] `https://doi.org/10.21942/uva.5660617` (last accessed June 16, 2021).

[7] `https://github.com/cs-chan/ArtGAN/tree/master/data` (last accessed June 16, 2021).

3.1 Training Set

The main training datasets for both classification tasks consist of images provided by the Getty Research Institute ("Getty dataset"). All images shown in Figure 1 and Figure 2 belong to this Getty dataset. The label distribution of the Getty dataset is very unbalanced: Paintings occur much more frequently than drawings (Table 1). The most represented genre are landscape paintings with 1156 samples, the least represented genre are portraits with only 35 samples (Table 2).

To enlarge the training datasets and to address their imbalance, we used data from additional sources in some of our experiments. For the type classification, a subset of the Brill Iconclass AI Test Set ("Brill dataset") [34], two datasets from Kaggle ("Kaggle datasets"), a dataset from the Rijksmuseum Amsterdam ("Rijksmuseum dataset") [29], and a subset of the Wikiart paintings dataset ("Wikiart dataset") [41] were used. Table 1 provides an overview of these datasets. Because the datasets differ in quality, in some of our experiments only a subset of the datasets was used. While the Kaggle datasets and the Rijksmuseum dataset included type labels that were suitable for our type classification task, the other

(a) Genre painting **(b)** History painting **(c)** Landscape

(d) Portrait **(e)** Still life

Figure 2: Example images from the classes of the genre classification task.

Table 2: Datasets Used for the Genre Classification Task (not all datasets were used in all experiments)

Dataset	Number of suitable training images					
	Genre	History	Land-scape	Portrait	Still life	Total
Art500k[1]	14,752	8,394	18,632	19,593	3,467	64,838
Europeana[2]	11	0	4,368	6,213	1,068	11,660
Getty	763	898	1556	35	421	3673
SemArt[3]	1,813	8,931	2,779	3,650	1,029	18,202
WGA[4]	2,897	1,4507	4,474	5,184	1,435	28,497
All datasets	20,236	45,010	42,287	35,984	7,648	151,165

[1] `https://deepart.ust.hk/ART500K/art500k.html` (last accessed June 16, 2021).

[2] `https://pro.europeana.eu/page/search` (last accessed June 16, 2021).

[3] `http://noagarciad.com/SemArt/` (last accessed June 16, 2021).

[4] `https://www.wga.hu/index1.html` (last accessed June 16, 2021).

datasets required preprocessing of the labels: Images labeled as "sketches" in the Wikiart dataset were assigned to our drawings class. Images from the Brill dataset with the Iconclass codes "48(+354) / art (+ drawing)" or "48C52 / drawing" were assigned to our drawings class and images with the iconclass code "48(+351) / art (+ painting)" to our paintings class. Some images from the Brill dataset were discarded in a manual filtering step because they differed strongly from the rest of the training data. In addition to the previously mentioned datasets, we also downloaded training data from the Collection API of New York's Metropolitan Museum of Art ("Metropolitan dataset") [40] and from the Bing Image Search API ("Bing dataset"). In the case of the Metropolitan Museum API, we retrieved all images from the "Drawings and Prints" department whose "objectName" property was "drawing". For the Bing Image Search API, we used the search terms "anatomical drawing" and "court sketch" to retrieve images of drawings. To ensure high data quality, the query results were filtered manually in both cases.

For the genre classification task, in addition to the Getty dataset, we used a subset of the Art500k dataset ("Art500k dataset") [27], the SemArt dataset ("SemArt dataset") [13], and data from the Web Gallery of Art ("WGA dataset") [16]. In the Art500k, SemArt, and WGA datasets, there is no "history paintings" class, but a "religious paintings" class. Images from this class were classed as history paintings for our genre classification task. Additional training data for the genre classification task were obtained from the Europeana Search API ("Europeana dataset") [9]. An overview of all datasets applicable for the genre classification task is provided in Table 2. Since the listed datasets differ in quality, not all available training data were used in all of our experiments.

3.2 Validation Set and Test Set

For both tasks, a split of the Getty dataset was used for model validation ("Getty validation set"). The validation set for the type classification consists of 1,438 images. Similar to the Getty training set for this task, it is also very unbalanced: 1,271 of the images represent paintings, while only 212 images show drawings. The validation set for the genre classification consists of 1,049 images and is also unbalanced: It includes 445 landscape paintings, 276 history paintings, 208 genre paintings, 109 still lifes and 11 portraits.

To evaluate the models, another split of the Getty dataset was used, which was not used to train or validate the models ("Getty test set"). The test set for the type classification task consists of 750 images. 636 of these are paintings and 114 are drawings. The test set for the genre classification task consists of 156 images. It includes 47 landscape paintings, 37 history paintings, 19 genre paintings, 50 still lifes, and 3 portraits. For the genre classification task, we also used the SemArt dataset described in subsection 3.1 to evaluate some of our models. This dataset was not used for training the models in these cases.

4 Methods

We have two distinct problem statements: a binary classification whether an image is a painting or a drawing ("type classification") and a multi-class classification of an image's genre ("genre classification"). Our approaches to these two share many similarities; however, we will also highlight the differences.

4.1 Evaluation Metrics for Imbalanced Datasets

For the Getty dataset, we can trivially construct a classifier with 85% validation accuracy in the type classification challenge by classifying each image as painting. When dealing with imbalanced datasets in general, accuracy is not a good performance metric [3]. Therefore, we use two other metrics that are commonly recommended for such scenarios: the F_1-score and Matthew's Correlation Coefficient. Both metrics account for the frequency distribution of the classes.

A hypothetical classification of 24 images, with TP=18, TN=1, FP=3, and FN=2 would yield an F_1-score of 88%. This would indicate a good classifier, however closer examination reveals the following facts: Only one in four drawings is classified correctly; also, two out of three predicted drawings are actually paintings. One of the weaknesses of the F_1-score is that it is not symmetric; the positive class is given more "weight" and this choice therefore affects the result. Also, the number of true negatives (TN) has no impact on the F_1-score as it does not influence precision or recall. If we were to flip positive and negative classes in our example, the F_1-score would drop to 29%. Matthew's Correlation Coefficient (MCC) is a symmetric performance measure that is also well-suited for imbalanced datasets. It measures the correlation between predicted and actual values (obviously, a high correlation is desirable). It is given by the formula:

$$MCC = \frac{TP \times TN + FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{1}$$

Because the F_1-score was one of the primary metrics by which we were evaluated in the challenges, we still use it as our primary metric. We also use the MCC as a "sanity check"; when the MCC and F_1-score are too far apart, it indicates that our model is not performing well across all classes.

4.2 Type Classification

We use the popular DenseNet and ResNet architectures [15, 17]. Motivated by similar research [23, 37] and initial experiments, we make heavy use of transfer learning. In particular, we use the ResNet18, ResNet34, ResNet50 and DenseNet201 architectures with weights pre-trained on ImageNet [11], which are provided by Torchvision [28]. Because of the imbalances in our primary dataset (the Getty dataset), we pay special attention the the F_1-score computed with "drawings" as the positive class.

Data preprocessing. Our dataset contains images that often exceed a dimension of 1000×1000 pixel. Because of hardware limitations, we are forced to re-scale our inputs to a smaller size. Additionally, the images are not square. Therefore, we also apply a quadratic crop before feeding the images to our networks. Although researchers often re-scale their inputs to 224×224, following a convention from AlexNet [22], we also experiment with input dimensions up to 400×400. We expect that larger images contain more information and thus improve the performance of our models.

In order to further speed up the training process, we re-scaled the entire training set so that the smaller dimension (width or height) is equal to 400. This way, we can still apply various cropping techniques but significantly reduce disk I/O, which was a bottleneck in our initial experiments. In particular, we apply either a centered or a random quadratic crop. As we observed that some paintings and drawings are framed or do not fill the entire input image, we also implemented a "random borderless crop": before applying the random crop, we truncate 20 pixels from every side of the input image. In most cases, this was sufficient to remove frames from a picture.

We also used a multitude of different augmentation techniques on our training images. We experimented with random horizontal flipping, perspective transformations, color jittering, grayscale transformations, Gaussian blurs and random rotations, shearing or re-scaling with varying probabilities.

We normalized all images with the channel means and standard deviations from ImageNet. It should be noted that this is suboptimal and could be improved by calculating the means and standard deviations of our actual training set; however, we did achieve satisfactory results.

Model architecture and training details. As mentioned above, we conducted experiments with ResNet50 and DenseNet201. We chose these as the deepest representatives of their respective classes that we could feasibly train on our hardware. However, we also ran experiments with smaller versions of ResNet, such as ResNet18 and ResNet34, which have the benefit of reduced training times. We use transfer learning: the model weights are pre-loaded with weights pre-trained on ImageNet. As we want to adapt to a new problem space, we replace the fully-connected decision layers. Whereas the original architectures use only one fully-connected layer to produce the one thousand outputs of ImageNet, we also experimented with using 4 consecutive fully-connected layers with the ReLU activation function. The motivation is two-fold: firstly, our problem space requires only one output instead ImageNet's thousand and we theorized that this additional reduction warrants additional layers. Secondly, we believe our problem space to be more complex and additional layers might be able to better capture this complexity. Our best model for the type classification task uses four such fully-connected layers.

There are two main variants of transfer learning: feature extraction and fine-tuning. During feature extraction, only the decision layer is trained on the new dataset, while the rest of the weights are frozen. Therefore, the pre-trained part of the network extracts useful features that the new decision layer then receives

as inputs. During fine-tuning, all weights of the network are re-trained to adapt the entire network to the new problem space. Of course, one can also re-train only selected layers such as the last convolutional layer during fine-tuning.

During our experiments, we try feature extraction as well as differing degrees of fine-tuning. In all cases, we do not freeze the weights of batch normalization layers. These layers capture statistics of the underlying dataset, which is ImageNet in the cases of our pre-trained weights. As we use a different dataset, we want these statistics to be updated according to our new dataset. We found that fine-tuning leads to superior results compared to feature extraction. Since our dataset consists of paintings and drawings, which are not part of ImageNet, we expected to achieve better results when adapting larger parts of our model to the new dataset.

We use the AdaDelta and Adam optimizers [21, 46] and a binary cross-entropy loss. We run experiments with cosine learning rate scheduling [25], step-wise learning rate decay, and constant learning rates. We employ a version of *early-stopping*: after every epoch, we evaluate our model on the validation set and store the result as well as the model weights. At the end of the training, we can then choose the model weights that yielded the best results.

Combating data imbalances. A significant challenge was the imbalance of our primary dataset. In the following, we detail several different strategies that we implemented to address this problem.

If we train our models only on the Getty dataset, they will see many more paintings than drawings. This can be counteracted by sampling drawings at a higher rate than paintings, so that the model is presented an even distribution between the two classes. Since we have few original drawings, this should be coupled with strong augmentation techniques. Otherwise the model might overfit to the drawings that it is repeatedly shown. There are two "flavours" of this sampling technique: oversampling and undersampling. Oversampling works as described by sampling the minority class at a higher rate, while undersampling works by sampling the majority class at a lower rate to make the two classes balanced. While oversampling has the drawback of potential overfitting, undersampling can cause loss of information as majority-class samples are randomly excluded from the training set each epoch.

Another solution is to use additional datasets that contain drawings to achieve balance. As can be seen in section 3, we experimented with many additional datasets. Because they also did not have an even distribution of paintings and drawings, we combined the additional datasets with oversampling or undersampling. In some cases, depending on the additional datasets that were used, paintings became the minority class and had to be oversampled (or drawings undersampled). Another approach we have taken is to generate artificial training data. We perform a rudimentary style transformation of paintings to drawing-like images. Paintings are first converted to greyscale, then a gaussian filter with a σ of 10 is applied. The inverted image and the gaussian filter are then overlayed. After the style transformation, a random resize crop to 224 x 224 pixels, a random horizontal flip and a normalization is applied. We evaluate transformation probabilities of $\frac{1}{3}$ and $\frac{1}{2}$ that regulate the proportion of transformed paintings.

The methods described above operate at the *data level*: they modify the data so that the model is not "aware" of the imbalance. We also implemented methods at the *algorithm level*; specifically, we use two modifications of the loss function. One is a weighted loss, where the influence of wrongly classified minority-class samples on the loss can be increased by a weight factor. The imbalance in the dataset is then offset by the fact that the minority class has more impact on the loss and therefore the training process.

Another modification is to use a different loss that incorporates a balancing mechanism: the F_1-score. This has the added benefit of a closer alignment between the loss that is used to train the model and the primary metric that we want to optimize. Unfortunately, the F_1-score is not guaranteed to be differentiable as there is the possibility of a division by zero. This problem can be overcome by adding a small ϵ = 1e-10 to the denominator wherever this possibility exists. As we want to maximise the F_1-score, we minimize the F_1-score subtracted from one as our loss.

Other improvements. We employ *test-time augmentation*. Using PyTorch's FiveCrop and TenCrop modules,[1] we feed our model five versions of the same image at test-time: one crop from every corner of the image as well as the center crop. In the case of TenCrop, the model is also fed a horizontally flipped version of every crop. The final prediction is then set to the most common prediction over all crops. We also experiment with an ensemble of three different ResNet models. As the ResNet50 model has more parameters than the smaller ResNets, it is more prone to overfitting, which is why we also included a ResNet34 and ResNet18.

4.3 Genre Classification

In the genre classification challenge, we implemented two different approaches: an ensemble-based approach consisting of five one-versus-the-rest classifiers for the five genres and one single-model approach.

The data preprocessing and techniques to combat data imbalances are analogous to the type classification described in subsection 4.2. However, we use an additional sampling technique: a combination of over- and undersampling. The open-source ImbalancedDatasetSampler[2] aims to achieve a balance between over- and undersampling to alleviate their respective drawbacks.

Ensemble model. To classify the five different genres, we train one model for each genre that classifies whether an image belongs to that genre or not ("one-versus-the-rest"). These models then build an ensemble. The final predicted genre is decided by the network that yielded a positive result with the highest confidence. For the individual models of the ensemble we used the ResNet50 architecture. We train each ResNet50 model with different combinations of augmentation methods

[1] `https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.[Five|Ten]Crop` (last accessed June 16, 2021).

[2] `https://github.com/ufoym/imbalanced-dataset-sampler` (last accessed June 16, 2021).

and learning rate scheduling. We did not do a systematic search for the best combinations but rather made choices based on intuition. Our first ensemble consisted of the best-performing classifiers for each genre. However, we found that this choice was not optimal. Through manual experimentation, we found an ensemble where the individual classifiers were sub-optimal but together, they outperformed the previous ensemble.

Single model. In contrast to the ensemble approach, we also trained a single model capable of classifying all genres. We experiment with the same model architectures mentioned in subsection 4.2. Because there are five different genres, our models' final decision layer now has five outputs instead of just one, which are fed into a softmax layer. Instead of binary cross-entropy loss, categorical cross-entropy loss is used.

5 Results and Discussion

In the following, we describe our results for both classification tasks. For the type classification task, we explored multiple training methods. In the genre classification task, we built on these results and used the best training approaches from the type classification task.

6 Type Classification Results

For the type classification task, we first describe the main findings we obtained from the exploration of different training methods. Subsequently, we describe the model with the best overall performance and compare it to some of our other models.

6.1 Impact of Data Augmentation

Since the Getty dataset is comparatively small, we experimented with various augmentation methods to prevent model overfitting. To identify the most suitable augmentation methods, we trained ResNet50 models on the Getty dataset with one augmentation method each and compared them to a baseline model trained without data augmentation. To account for different model initializations, we ran three trainings with 30 epochs for each augmentation method. We determined the best model (best F_1-score for the drawings class on the Getty validation set) from each training and report the averaged F_1-scores of the best models in Table 3. Details on the training setting are also provided in Table 3. As shown in Figure 3, converting the training images to grayscale images significantly degrades the model performance on the validation set compared to the baseline model. Random resizing and cropping of the training images, on the other hand, slightly improves the

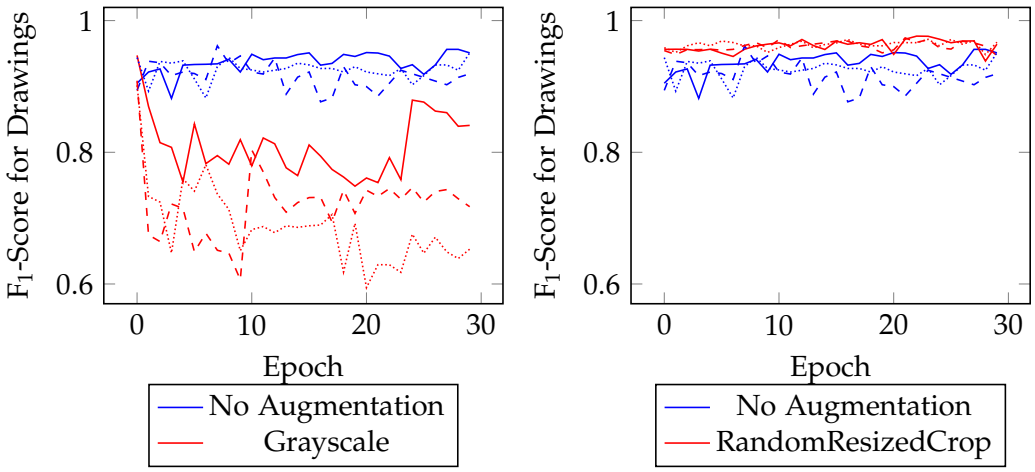


Figure 3: Performance of ResNet50 models trained with different data augmentation methods on the unbalanced Getty dataset. In both plots, we show the F_1-score for the drawings class on the Getty validation set.

performance on the validation set. For all other augmentation methods we tested, the performance of the models on the Getty validation set is very similar to that of models trained without data augmentation.
The fact that converting training images to grayscale images significantly degrades the training results suggests that the models strongly account for color information in discriminating between drawings and paintings. This is surprising given that only a subset of the training set consists of color images. However, among the colored training images, paintings are indeed usually colorful, while drawings use a limited color palette.
One possible explanation for the performance increase achieved by random resizing and cropping is that it prevents overfitting to certain image features. However, it is unclear why random cropping does not have a similar effect on model performance. Our custom "border cropping" technique that removes image frames led to a slight decrease in performance. It could be that the image frames, which we initially thought of as clutter or noise, actually contain information that our model can learn from.

6.2 Coping With Unbalanced Training Data

As described in subsection 4.2, we implemented several approaches to address the imbalance in the Getty training set. Below, we describe how these approaches impact model performance.

Oversampling and undersampling. To examine how oversampling of the minority class (drawings) or undersampling of the majority class (paintings) affects model performance, we trained three ResNet50 models for 30 epochs with each sampling method. To determine the baseline performance, we trained three Resnet50 models

Table 3: Performance of ResNet50 Models Trained on the Getty Dataset

Augmentation Method	Validation Set		Test Set	
	F_1-Score for Drawings	F_1-Score for Paintings	F_1-Score for Drawings	F_1-Score for Paintings
RandomResized-Crop	0,974 ± 0,002	0,996 ± 0,000	0,971 ± 0,005	0,995 ± 0,001
No Augmentation	0,956 ± 0,004	0,993 ± 0,001	0,967 ± 0,009	0,994 ± 0,002
ColorJitter	0,959 ± 0,004	0,993 ± 0,001	0,965 ± 0,008	0,994 ± 0,001
Border Cropping	0,946 ± 0,008	0,991 ± 0,001	0,965 ± 0,002	0,994 ± 0,000
RandomPerspective	0,961 ± 0,002	0,994 ± 0,000	0,964 ± 0,000	0,994 ± 0,000
RandomHorizontal-Flip	0,959 ± 0,005	0,991 ± 0,003	0,963 ± 0,006	0,993 ± 0,001
RandomRotation	0,958 ± 0,004	0,993 ± 0,001	0,962 ± 0,008	0,993 ± 0,001
RandomAffine	0,948 ± 0,004	0,992 ± 0,001	0,961 ± 0,006	0,993 ± 0,001
RandomCrop	0,955 ± 0,005	0,993 ± 0,001	0,954 ± 0,014	0,992 ± 0,002
Grayscale	0,919 ± 0,020	0,986 ± 0,004	0,920 ± 0,022	0,985 ± 0,005

Models were pre-trained on ImageNet; batch norm, conv. 4, conv. 5 and fc. layers were fine-tuned for 30 epochs on the unbalanced Getty dataset. Training images were resized and cropped to 224 x 224 pixels. A batch size of 100 was used. An Adam optimizer and a cosine annealing learning rate scheduler were used (initial learning rate: 10^{-4}, η_{min} = 0). For all augmentation techniques except border cropping, the implementations from the torchvision package were used.[1] For border cropping, a custom implementation was used as described in subsection 4.2. For the augmentation methods from the torchvision package, the default parameters were used unless otherwise specified below:

ColorJitter: brightness=0.5, contrast=0.5, saturation=0.5, hue=0.1

RandomAffine: degrees=(0, 40), translate=(0.0, 0.4), scale=(0.6, 1.4), shear=0.2, resample=BICUBIC

RandomCrop: size=224, images were resized to 224 pixels (longer edge) in advance

RandomResizedCrop: size=224, images were resized to 300 pixels (longer edge) in advance

RandomRotation: degrees=360

[1] `https://pytorch.org/vision/stable/transforms.html` (last accessed June 16, 2021).

on the unbalanced Getty dataset. We determined the best model (best F_1-score for the drawings class on the Getty validation set) from each training and report the averaged F_1-scores of the best models in Table 4.
We expected that undersampling the majority class would degrade the model performance because information is lost by randomly discarding training images. On the other hand, we expected that oversampling would improve the results by counteracting a model bias in favor of the majority class. However, as our results in Table 4 show, this did not prove true. All three ResNet50 models achieve very similar performance on the Getty test set. Similar observations were made when training models with other architectures, e.g. ResNet18. This demonstrates that transfer learning yields satisfactory results even on highly imbalanced datasets and that over- or undersampling techniques are not necessarily needed.

Table 4: Performance of ResNet50 Models Trained on the Getty Dataset Using Different Sampling Methods

Dataset	Validation Set		Test Set	
	F_1-Score for Drawings	F_1-Score for Paintings	F_1-Score for Drawings	F_1-Score for Paintings
Getty, unbalanced	0,956 ± 0,004	0,993 ± 0,001	0,967 ± 0,009	0,994 ± 0,002
Getty, random undersampling	0,960 ± 0,004	0,994 ± 0,001	0,966 ± 0,008	0,994 ± 0,001
Getty, random oversampling	0,955 ± 0,006	0,993 ± 0,001	0,965 ± 0,005	0,994 ± 0,001

Models were pre-trained on ImageNet; batch norm, conv. 4, conv. 5 and fc. layers were fine-tuned for 30 epochs. Training images were resized and cropped to 224 x 224 pixels. A batch size of 100 was used. An Adam optimizer and a cosine annealing learning rate scheduler were used (initial learning rate: 10^{-4}, $\eta_{min} = 0$).

Additional training data. We expected that models trained on larger training sets would perform better and overfit less. To verify this assumption, we trained ResNet50 models on the Getty dataset and one additional dataset each. To exclude differences caused by different distributions of the datasets, we balanced the training sets by random undersampling. Table 5 shows for each dataset combination the metrics of the model that achieved the highest F_1-score for the drawings class on the Getty validation set. Despite our assumption, not all datasets improve performance compared to the model trained only on the Getty dataset. Possibly this is because some datasets differ too much from the Getty dataset in terms of style and era of the artworks. In particular, the images in the Bing and Brill datasets differ significantly from those in the Getty dataset. In contrast, the Rijksmuseum dataset, which is stylistically most similar to the Getty dataset and represents the

largest dataset, yields the model with the best performance. This indicates that additional, high quality datasets can improve the models. However, the transfer learning approach allows to obtain satisfactory results even with small datasets.

Table 5: Performance of ResNet50 Models Trained on the Getty Dataset and One Additional Dataset Each

Dataset	Validation Set		Test Set	
	F_1-Score for Drawings	F_1-Score for Paintings	F_1-Score for Drawings	F_1-Score for Paintings
Getty + Rijksmuseum	0.9644	0.9941	0.9735	0.9953
Getty + Wikiart	0.9573	0.9929	0.96	0.9929
Getty + Metropolitan	0.9567	0.993	0.96	0.9929
Getty	0.954	0.9926	0.96	0.9929
Getty + Bing	0.9571	0.9929	0.9469	0.9906
Getty + Brill	0.9592	0.9933	0.9432	0.9898
Getty + Kaggle	0.9471	0.9914	0.9417	0.9898

The models were pre-trained on ImageNet; batch norm, conv. 4, conv. 5 and fc. layers were fine-tuned for 30 epochs. Training images were resized and cropped to 224 x 224 pixels. The training sets were balanced by random undersampling and a batch size of 100 was used. An Adam optimizer and a cosine annealing learning rate scheduler were used (initial learning rate: 10^{-4}, $\eta_{min} = 0$).

Artificial training data. Besides acquiring additional training data, we also experimented with algorithmically converting paintings into drawing-like images. The results of these experiments are listed in Table 6. For each experiment, we report the highest micro-averaged F_1-scor achieved on the Getty validation set. Converting a portion of the paintings to drawing-like images slightly improved results over the baseline model. This indicates that the models consider color and edge information when discriminating between drawings and paintings, as our style transformation mainly changes colors and edges of the images. This is consistent with our observation from the experiments with data augmentation, which showed that color information is very important for the type classification task.

Adapted loss functions. As described in subsection 4.2, we also tested two custom loss functions. In the first loss function, the cross-entropy loss of the minority class was weighted higher by a factor. Intuitively, this factor should be chosen according to the relation of drawings to paintings in the training set. This did not prove correct in our initial experiments and we had to manually fine-tune this factor to give drawings even more weight in order to achieve satisfactory results. Because of this difficulty and the introduction of yet another hyper-parameter in form of the weight factor, we discarded this strategy in our further experiments. Using the

F_1-score as loss function also did not improve the results. Therefore, for simplicity, we used only the binary cross-entropy loss in our other experiments.

6.3 Model With the Best Overall Performance

The overall best performance in the type classification task was achieved with a DenseNet201 pre-trained on the ImageNet dataset. We fine-tuned all layers of the model using the Getty dataset, the Kaggle datasets, the Metropolitan dataset and the Wikiart dataset. Images were resized to 400×400. We apply random horizontal flipping and random affine augmentations.[3] During training, the drawings class was randomly oversampled. The model optimization was done using the binary cross-entropy loss, a constant learning rate of 0.005 and the AdaDelta optimizer. When generating predictions, we used FiveCrop for test-time augmentation, as described in subsection 4.2. With this configuration, we achieve an F_1-score of 0.9825 for the drawings class on the Getty validation set. On the Getty test set, we achieve an F_1-score of 0.9739 for the drawings class and an F_1-score of 0.9953 for the paintings class (Table 7). Overall, this is our best result, but we have obtained similar results with other model architectures and configurations. For example, the ResNet50 listed in Table 7 that was trained on the Getty and the Rijskmusuem dataset, yields almost the same performance. However, the ensemble model listed in Table 7 did not improve the results. Considering the comparatively low complexity of the type classification task, satisfactory results can also be obtained with flatter models, e.g. ResNet18 models. Due to the transfer learning approach, good results are usually achieved after only a few training epochs, allowing to train models for type classification even with limited computational resources.

Table 6: Performance of ResNet18 Models Trained on the Getty Dataset

Dataset	Micro F_1-Score
Getty	0.9256
Getty, 1/3 of paintings converted to drawings	0.9357
Getty, 1/2 of paintings converted to drawings	0.9365

The models were pre-trained on ImageNet; batch norm, conv. 4, conv. 5 and fc. layers were fine-tuned for 30 epochs. Training images were resized and cropped to 224 x 224 pixels and a batch size of 32 was used. An Adam optimizer and a constant learning rate of 10^{-3} were used.

[3] `https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.RandomAffine` (last accessed June 16, 2021).

Table 7: Performance of Selected Models in the Type Classification Task

Model	Validation Set		Test Set	
	F_1-Score for Drawings	F_1-Score for Paintings	F_1-Score for Drawings	F_1-Score for Paintings
DenseNet201	0.9749	0.9959	0.9739	0.9953
ResNet50	0.9644	0.9941	0.9735	0.9953
Ensemble of Resnet18, Resnet34 and Resnet50	0.969	0.9949	0.96	0.9929
ResNet34	0.9592	0.9933	0.9553	0.9922
ResNet18	0.9596	0.9933	0.9391	0.9890

The DenseNet201 was trained on the Getty dataset, the Kaggle datasets, the Metropolitan dataset and the Wikiart dataset. The ResNet50 was trained on the Getty dataset and the Rijksmuseum dataset. The ResNet34 and the ResNet18 were trained on the Brill dataset, the Getty dataset, the Kaggle datasets and the Rijksmuseum dataset. In the ensemble model, the predictions of the ResNet18, the ResNet34 and the ResNet50 were combined by a majority vote.

7 Genre Classification Results

Building upon the results of the binary type classification task, we approach the multi-class genre classification with single model setups of different architectures, as well as ensemble-based approaches.

7.1 Single Model Approaches

For the single model approach, we used two different architectures, ResNet50 and DenseNet201. Table 8 compares the results obtained with both architectures. There, we report the highest micro-averaged F_1-score achieved on the Getty validation set for each model.

To determine the baseline performance for the DenseNet201 architecture, we fine-tuned a model on the Getty and Art500K datasets. In this baseline configuration, all layers were fine-tuned and the training set was balanced by random oversampling. In Table 8, this baseline model is compared to a model that was trained with cosine annealing learning rate scheduling [25]. We observe that the model without learning rate scheduling performs slightly better and achieves a 0.3% higher micro F_1-score. We observed only small differences between the FiveCrop and TenCrop test-time image augmentations. FiveCrop performed better in our final model but we do not believe that this observation is generalizable. The overall best model of the DenseNet201 architecture achieves a micro F_1-score of 0.924. For this model, we resized and cropped the images to 400 x 400 pixels and applied random horizontal flipping and random affine transformations for data augmentation.

For the ResNet50 architecture, we use a model as baseline with only the softmax classifier fine-tuned. Compared to the baseline model, a model trained with the Pytorch "reduce learning rate on plateau" scheduler[4] achieves a 0.3% higher micro F_1-score. Using the ImbalancedDatasetSampler[5] we improve the baseline micro F_1-score by 3.4%. For type classification, we observed that unfreezing the last convolutional layers and the classifier results in further improvements. We therefore evaluate the impact of retraining layers in combination with the previously explained balancing technique for the genre classification. We note that retraining three layers outperforms retraining five layers and results in an overall best run of the ResNet50 with a micro F_1-score of 0.931.

Table 8: Results for Single Models of DenseNet201 and ResNet50 Architecture in the Genre Classification Task

Experiment	Configuration	Micro F_1-Score
DenseNet201		
Baseline	Baseline configuration[1]	0.924
Learning rate	Cosine annealing scheduling	0.921
ResNet50		
Baseline	Baseline configuration[2]	0.838
Learning rate	Reduce on plateau scheduling	0.841
Balancing	Imbalanced sampler	0.872
Retraining	3 layers retrained[3]	0.928
	5 layers retrained[4]	0.881

[1] The DenseNet201 models were pre-trained on ImageNet; all layers were fine-tuned for 30 epochs using the Art500k dataset and the Getty dataset. The training set was balanced by random oversampling and a batch size of 32 was used. A constant learning rate of 10^{-3} was used in the baseline configuration.

[2] The ResNet50 models were pre-trained on ImageNet; the classifier was fine-tuned for 30 epochs using the Getty dataset. A batch size of 32 was used. A constant learning rate of 10^{-3} was used in the baseline configuration. Both retraining configurations used the ImbalancedDatasetSampler for balancing.

[3] Retraining of conv. 4, conv. 5 and fc.

[4] Retraining of conv. 2, conv. 3, conv. 4, conv. 5 and fc.

[4] `https://pytorch.org/docs/stable/optim.html#torch.optim.lr_scheduler.ReduceLROnPlateau` (last accessed June 16, 2021).

[5] `https://github.com/ufoym/imbalanced-dataset-sampler` (last accessed June 16, 2021).

Table 9: Performance of ResNet50 Models Trained to Detect One Genre Each (one-versus-the-rest)

Hyperparameter	Class					
	Portrait, ImageNet	Portrait, VGGFace2	Genre	History	Still life	Landscape
Without data augmentation						
Stepwise	0.474	0.545	0.728	0.899	0.53	0.9
Cosine annealing	0.439	0.390	0.705	0.9	0.545	0.918
With data augmentation						
Stepwise	0.428	-	0.572	0.841	0.472	0.762
Cosine annealing	0.342	0.5	0.592	0.702	0.395	0.734

The models were pre-trained on ImageNet, for the portrait class an additional model pre-trained on VGGFace2 was evaluated. Batch norm, conv. 4, conv. 5 and fc. layers were fine-tuned for 30 epochs on the Art500k, Europeana, Getty and WGA datasets. Training images were resized and cropped to 224 x 224 pixels. The training sets were balanced by random undersampling and a batch size of 115 was used. An Adam optimizer was used. For stepwise learning rate decay (denoted as *stepwise*), the initial learning rate was set to 10^{-3} and the step size to 4. For cosine annealing learning rate scheduling (denoted as *cosine annealing*), the initial learning rate was set to 10^{-4} and η_{min} to 0. The transformations RandomGrayscale, RandomPerspective and RandomHorizontalFlip from the torchvision package were used for data augmentation.[1]

[1] `https://pytorch.org/vision/stable/transforms.html` (last accessed June 16, 2021).

Table 10: Performance of Two Ensemble Models for Genre Classification

Genre	Ensemble Model 1		Ensemble Model 2	
	F_1-Score on Getty	F_1-Score on SemArt	F_1-Score on Getty	F_1-Score on SemArt
Genre	0.757	0.5101	0.7646	0.5237
History	0.763	0.8542	0.7467	0.8631
Landscape	0.9154	0.8574	0.9145	0.8658
Portrait	0.8	0.6949	0.8	0.7736
Still life	0.9507	0.7434	0.9507	0.7711

Ensemble 1 is a combination of the best individual models from the experiments presented in Table 9. Ensemble 2 was created by exploring various single model combinations. Both Ensemble models consist of one individual ResNet50 classifier per genre category pre-trained on ImageNet. Batch norm, conv. 4, conv. 5 and fc. layers of each ResNet50 were fine-tuned for 30 epochs on the Art500k, Europeana, Getty and WGA datasets. For details on the training setting, see Table 9. In both ensemble models, the prediction of the classifier with the highest confidence was used as final prediction.

7.2 Ensemble model approach

Besides single models, we also experimented with ensemble models in the genre classification task. As described in subsection 4.3, our ensemble models consist of five one-versus-the-rest ResNet50 classifiers where each classifier is trained to detect one genre class. For each of the five classifiers, we evaluate the impact of data augmentation and learning rate scheduling. For all genres, we use ResNet50 models pre-trained on the ImageNet dataset. For the portrait class, we additionally evaluate an InceptionResNet pre-trained on the VGGFace2 dataset [5] for face recognition. Table 9 shows the performance of the best single models for these different experimental settings. For each model, we report the highest F_1-score that was achieved on the Getty validation set. For landscape paintings we observe a baseline F_1-score of 0.9 which is the highest baseline score over all classes. Data augmentation reduces the F_1-score by 14%. This decrease caused by the use of augmentation is observed throughout all single models of the ensemble with varying margins. Fine-tuning with cosine annealing learning rate scheduling improved the F_1-scores of the landscape, history and still life classifiers up to 1% compared to the stepwise learning rate decay. For the classification of the genre paintings class the use of stepwise learning rate decay outperforms the cosine scheduling by 2.3%. A similar effect is noticeable for the InceptionResNet classifier for the portrait class where the the F_1-score drops by 15% when using the cosine annealing learning rate scheduling. Comparing the ResNet50 and the InceptionResNet for classifying the portrait class, we notice that they both perform better without image augmentation but the InceptionResNet shows slightly higher confidence.

When selecting the individual models for the ensemble, we compared a combination of the best individual models (Ensemble 1) with a trial-and-error ensemble (Ensemble 2) whose composition was guided by intuition. Table 10 shows that both ensemble models perform similarly on the Getty validation set. When validated on the SemArt dataset, Ensemble 2 performs slightly better. In particular, Ensemble 2 achieves a higher F_1-score for the history paintings class on the Semart dataset. We therefore conclude that Ensemble 2 shows increased robustness compared to the best-of-model.

8 Conclusion

In this work, we successfully applied deep learning models to two image classification tasks from the art domain. In the type classification of artworks, our models achieve human-like accuracy. As expected, the genre classification of artworks turned out to be more a complex problem. Building on our results, future work should aim to further improve model performance for this classification task.
Overall, our results demonstrate that the transfer learning approach can produce very good results even on small, highly unbalanced training datasets. By using additional training data and employing random resizing for data augmentation, we were able to further improve the performance of our models. In contrast, most data

augmentation methods, over- and undersampling techniques, and adjustments to the loss functions did not yield any benefits. Ensemble learning also did not improve the results compared to the single-model approach. Since most of our experiments rely on a comparatively small number of training runs, future work should strive for additional statistical evidence. In particular, the impact of data augmentation and over- or undersampling techniques likely depends on the training datasets and should therefore be further investigated.

References

[1] S. Baker. "Finding A Way To Make Digitizing Art Collections Profitable". In: *Forbes* (June 1, 2019).

[2] M. Barni, A. Pelagotti, and A. Piva. "Image processing for the analysis and conservation of Paintings: Opportunities and challenges". In: *IEEE Signal Processing Magazine* 22.5 (Sept. 2005), pages 141–144. ISSN: 1558-0792. DOI: 10.1109/MSP.2005.1511835.

[3] P. Branco, L. Torgo, and R. Ribeiro. "A Survey of Predictive Modelling under Imbalanced Distributions". In: *arXiv* 1505.01658 (May 13, 2015).

[4] V. Burke, D. Jørgensen, and F. A. Jørgensen. "Museums at Home: Digital Initiatives in Response to COVID-19". In: *Norsk museumstidsskrift* 6.2 (Dec. 10, 2020), pages 117–123. ISSN: 2464-2525. DOI: 10.18261/issn.2464-2525-2020-02-05.

[5] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. *VGGFace2: A dataset for recognising faces across pose and age*. May 13, 2018.

[6] E. Cetinic and S. Grgic. "Genre classification of paintings". In: *Proceedings of the International Symposium on Electronics in Marine* (Zadar, Croatia). Edited by M. Muštra, D. Tralić, and B. Zovko-Cihlar. Zadar, Croatia: ELMAR, 2016, pages 201–204. ISBN: 978-953-184-221-1. DOI: 10.1109/ELMAR.2016.7731786.

[7] E. Cetinica, T. Lipica, and S. Grgic. "Fine-tuning Convolutional Neural Networks for fine art classification". In: *Expert Systems With Applications* 114 (Dec. 30, 2018), pages 107–118. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.07.026.

[8] *Collection*. The National Museum of Art, Architecture and Design Oslo. URL: https://www.nasjonalmuseet.no/en/collection (visited on 2021-03-16).

[9] C. Concordia, S. Gradmann, and S. Siebinga. "Not just another portal, not just another digital library: A portrait of Europeana as an application program interface". In: *IFLA Journal* 36.1 (2010), pages 61–69. ISSN: 1745-2651. DOI: 10.1177/0340035209360764.

[10] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (San Diego, USA). Volume 1. IEEE, 2005, pages 886–893. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.177.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (San Francisco, USA). IEEE, 2009, pages 248–255. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848.

[12] C. Florea, R. Condorovici, C. Vertan, R. Boia, L. Florea, and R. Vrânceanu. "Pandora: Description of a Painting Database for Art Movement Recognition with Baselines and Perspectives". In: *Proceedings of the 24th European Signal Processing Conference* (Budapest, Hungary). IEEE, 2016, pages 918–922. ISBN: 978-0-9928-6266-4. DOI: 10.1109/EUSIPCO.2016.7760382.

[13] N. Garcia and G. Vogiatzis. "How to Read Paintings: Semantic Art Understanding with Multi-modal Retrieval". In: *Computer Vision – ECCV 2018 Workshops. Part II* (Munich, Germany). Edited by L. Leal-Taixé and S. Roth. Lecture Notes in Computer Science. Springer, Cham, Jan. 29, 2019, pages 676–691. ISBN: 978-3-030-11012-3. DOI: 10.1007/978-3-030-11012-3_52.

[14] *Getty Search Gateway*. The J. Paul Getty Trust. URL: https://search.getty.edu/gateway (visited on 2021-03-16).

[15] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *arXiv* 1512.03385 (Dec. 10, 2015).

[16] J. M. Hollands. "Web Gallery of Art". In: *Reference Reviews* 15.6 (2001), pages 32–32. ISSN: 0950-4125. DOI: 10.1108/rr.2001.15.6.32.335.

[17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. "Densely Connected Convolutional Networks". In: *arXiv* 1608.06993 (Jan. 18, 2018).

[18] X. Huang, S.-h. Zhong, and Z. Xiao. "Fine-art painting classification via two-channel deep residual network". In: *Advances in Multimedia Information Processing – PCM 2017* (Harbin, China). Edited by B. Zeng, Q. Huang, A. El-Saddik, H. Li, S. Jiang, and X. Fan. Volume 10736. Lecture Notes in Computer Science. Springer, Cham, May 10, 2018, pages 79–88. ISBN: 978-3-319-77383-4. DOI: 10.1007/978-3-319-77383-4_8.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the 22nd ACM International Conference on Multimedia* (Orlando, USA). New York, USA: ACM, 2014, pages 675–678. ISBN: 9781450330633. DOI: 10.1145/2647868.2654889.

[20] A. Joshi, A. Agrawal, and S. Nair. "Art Style Classification with Self-Trained Ensemble of AutoEncoding Transformations". In: *arXiv* 2012.03377 (Dec. 6, 2020).

[21] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv* 1412.6980 (Jan. 17, 2017).

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems* (Lake Tahoe, USA). Edited by P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Red Hook, USA: Curran Associates Inc., 2012, pages 1097–1105.

[23] A. Lecoutre, B. Negrevergne, and F. Yger. "Recognizing Art Style Automatically in painting with deep learning". In: *Proceedings of the Ninth Asian Conference on Machine Learning* (Seoul, Korea). Edited by M.-L. Zhang and Y.-K. Noh. Proceedings of Machine Learning Research. PMLR, pages 327–342.

[24] E. Leventaki. "Art Professionals and the Frenzy of Digitisation". In: *ICOM Voices* (July 8, 2020).

[25] I. Loshchilov and F. Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *arXiv* 1608.03983 (May 3, 2017).

[26] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pages 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94.

[27] H. Mao, M. Cheung, and J. She. "DeepArt: Learning Joint Representations of Visual Arts". In: *Proceedings of the 25th ACM international conference on Multimedia* (Mountain View, USA). New York, USA: ACM, Oct. 2017, pages 1183–1191. ISBN: 9781450349062. DOI: 10.1145/3123266.3123405.

[28] S. Marcel and Y. Rodriguez. "Torchvision the Machine-Vision Package of Torch". In: *Proceedings of the 18th ACM International Conference on Multimedia* (Firenze, Italy). New York, USA: Association for Computing Machinery, 2010, pages 1485–1488. ISBN: 978-1-60558-933-6. DOI: 10.1145/1873951.1874254.

[29] T. Mensink and J. van Gemert. "The Rijksmuseum Challenge: Museum-Centered Visual Recognition". In: *Proceedings of the International Conference on Multimedia Retrieval* (Glasgow, UK). New York, USA: ACM, 2014, pages 451–454. ISBN: 9781450327824. DOI: 10.1145/2578726.2578791.

[30] M. R. Mohammadi and F. Rustaee. "Hierarchical classification of fine-art paintings using deep neural networks". In: *Iran Journal of Computer Science* 4.1 (Sept. 30, 2020), pages 59–66. ISSN: 2520-8446. DOI: 10.1007/s42044-020-00072-0.

[31] T. Navarrete. "Digitization in Museums". In: *Teaching in Cultural Economics*. Edward Elgar, 2020, pages 204–213. ISBN: 978 1 78897 073 0.

[32] E. Oomen. "Classification of painting style with transfer learning". Master's thesis. Tilburg: Tilburg University, July 30, 2018.

[33] *Paintings*. The National Gallery London. 2021. URL: https://www.nationalgallery.org.uk/paintings (visited on 2021-03-16).

[34] E. Posthumus. *Brill Iconclass AI Test Set*. Feb. 21, 2020. URL: https://labs.brill.com/ictestset (visited on 2021-03-17).

[35] T. Rieger. *Technical Guidelines for Digitizing Cultural Heritage Materials: Creation of Raster Image Files*. Federal Agencies Digitization Guidelines Initiative, Aug. 2016.

[36] *Rijksstudio*. Rijksmuseum Amsterdam. URL: https://www.rijksmuseum.nl/en/rijksstudio (visited on 2021-03-16).

[37] M. Sabatelli, M. Kestemont, W. Daelemans, and P. Geurts. "Deep Transfer Learning for Art Classification Problems". In: *Computer Vision – ECCV 2018 Workshops. Part II* (Munich, Germany). Edited by L. Leal-Taixé and S. Roth. Lecture Notes in Computer Science. Springer, Cham, Jan. 29, 2019, pages 631–646. ISBN: 978-3-030-11012-3. DOI: 10.1007/978-3-030-11012-3_48.

[38] B. Saleh and A. Elgammal. "Large-scale classification of fine-art paintings: Learning the right metric on the right feature". In: *International Journal for Digital Art History* 2 (Oct. 2016), pages 70–93. ISSN: 2363-5401. DOI: 10.11588/dah.2016.2.23376.

[39] C. Sandoval, E. Pirogova, and M. Lech. "Two-stage deep learning approach to the classification of fine-art paintings". In: *IEEE Access* 7 (Mar. 28, 2019), pages 41770–41781. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2907986.

[40] L. Tallon. *Scaling the Mission: The Met Collection API*. The Metropolitan Museum of Art. Oct. 25, 2018. URL: https://www.metmuseum.org/blogs/now-at-the-met/2018/met-collection-api (visited on 2021-03-20).

[41] W. R. Tan, C. S. Chan, H. E. Aguirre, and K. Tanaka. "Ceci n'est pas une pipe: A Deep Convolutional Network for Fine-art Paintings Classification". In: *Proceedings of the IEEE International Conference on Image Processing* (Phoenix, USA). IEEE, 2016, pages 3703–3707. ISBN: 978-1-4673-9961-6. DOI: 10.1109/ICIP.2016.7533051.

[42] M. Terras. "Opening Access to collections: the making and using of open digitised cultural content". In: *Online Information Review* 39.5 (Sept. 2015), pages 733–752. ISSN: 1468-4527. DOI: 10.1108/OIR-06-2015-0193.

[43] *The National Museum in 2018*. The National Museum of Art, Architecture and Design Oslo. 2018. URL: https://www.nasjonalmuseet.no/contentassets/98adac84980c4555ae99de8a5ed00e80/rsmelding2018_eng.pdf (visited on 2021-03-17).

[44] L. Volkers. "Image First: Opening Up the Rijksmuseum With Rijksstudio". In: *The Digital in Cultural Spaces* (Singapore). Edited by T. Karthigesu, T. C. Hua, and C.-A. L. M. Gek. Singapore: Culture Academy Singapore, 2017, pages 15–22. ISBN: 978-981-11-5764-6.

[45] X. Wang, D. Kihara, J. Luo, and G.-J. Qi. "Enaet: Self-trained ensemble autoencoding transformations for semi-supervised learning". In: *arXiv* 1911.09265 (Feb. 1, 2021).

[46] M. D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *arXiv* 1212.5701 (Dec. 22, 2012).

[47] W. Zhao, D. Zhou, X. Qiu, and W. Jiang. "Compare the performance of the models in art classification". In: *PLOS One* 16.3 (Mar. 12, 2021), pages 1–16. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0248414.

[48] J. Zujovic, L. Gandy, S. Friedman, B. Pardo, and T. N. Pappas. "Classifying Paintings by Artistic Genre: An Analysis of Features & Classifiers". In: *Proceedings of the IEEE International Workshop on Multimedia Signal Processing* (Rio de Janeiro). IEEE, Nov. 2009, pages 1–5. ISBN: 978-1-4244-4463-2. DOI: 10.1109/MMSP.2009.5293271.

Object Detection

Alexander Junger, Emanuel Metzenthin, and Paul Wullenweber

Hasso Plattner Institute for Digital Engineering, University of Potsdam
{alexander.junger,emanuel.metzenthin,paul.wullenweber}@student.hpi.de

As more and more art collections are being digitized, new opportunities for automatic analysis open up, object detection being one of such tasks. However, annotated collections of artworks are still rare and models pre-trained on photos generally do not transfer well onto works of art. We therefore evaluated what can be achieved, relying only on available datasets and pre-trained convolutional neural networks (CNNs) in a complex object detection task.

We present three different ensemble training approaches for building a cross-domain object detector on art data. These models can predict 22 classes divided into six categories. While average precision metrics are still quite low, qualitative analysis shows promising results for many classes. Due to a lack of data, some classes performed better than others.

1 Introduction

Nowadays, many museums are digitizing their art collections to enable worldwide public web access (e. g. the New York Public Library Digital Collection [1]) and through the digitization the artworks remain preserved. Filtering functions based on artwork metadata (e. g. epoch, genre, painter etc.) and the types of objects depicted in images can be very helpful when it comes to exploring these digital collections. This is where automatic object detection can be applied. An exemplary use case for a user who wants to study religious paintings, could be to filter all art works for ones that contains only Mary and Jesus.

Various machine learning methods have already been developed for detecting objects in images [32]. They include the localization of objects by predicting bounding boxes and the classification of these objects into a set of classes. Most often, convolutional neural networks (CNNs) are used for this purpose as they have proved to be an extremely powerful tool in the area of image recognition in recent years [9]. Their ability to take images as input and encode semantics in their hidden layers makes them very efficient algorithms for image recognition with good generalization qualities.

However, CNNs usually require very large amounts of labeled training data in order to generalize well on unseen images. Object detection tasks in particular need very detailed labeling of individual objects with bounding boxes in the training

[1] https://digitalcollections.nypl.org (last accessed June 16, 2021).

images (so called instance-level annotation). This annotation process is both time-intensive and expensive, resulting in only a few available data sources. Especially in the art domain, only a handful of datasets are available and even fewer contain annotations. We take a closer look into this in section 3.

To tackle this issue, a commonly used method is to rely on pre-trained CNNs, which are trained on large, general photo collections and encapsulate low-level image understanding. They are then fine-tuned on the target domain (retrained on the target dataset while initializing layers with the pre-trained weights) in a separate training step, requiring less data and time. In section 4, we present three distinct approaches which rely on such pre-trained CNNs.

Still, the question of cross-domain transfer remains. Photos and artworks are known to have largely different data distributions [3] rendering a transfer learning from the photo domain difficult. In order to assess how well existing object detection models can be transferred to the art domain we set up a complex and very broad detection task covering 22 object classes in six categories, which were given as the project task. In section 5, we evaluate the effectiveness of our implementations of such transfer learning, before discussing the results in section 6, concluding and giving an overview of possible future work in section 7.

2 Related Work

In the following section we provide an overview of common object detection algorithms and introduce related work in the domain of detecting objects in the art domain.

2.1 Object Detection Algorithms

The first big leap in the development of detection algorithms was made through the introduction of CNNs, namely the region-based CNN (R-CNN) by Girshick et al. [7]. Previously mainly hand-crafted image features were used in combination with classification algorithms [33]. R-CNNs are based on the selective search algorithm [27], which proposes hundreds of regions per image, which potentially contain an object. These regions are then passed through a pre-trained CNN to extract features and are then classified using linear classifiers [33].

Further improvements were made with Fast R-CNN [6] and later Faster R-CNN [23], which made the region proposal step part of the CNN itself in the form of a region proposal network (RPN). This allowed for much faster processing as the original region proposal stage could be skipped.

An even faster model was presented by Redmon et al. [21] with You Only Look Once (YOLO). Their CNN processes an image once and predicts all bounding boxes across all classes for the image simultaneously with only a single network evaluation.

Predictions are made taking into account the whole picture, which means that YOLO partially encodes contextual information as well. This distinguishes YOLO

from existing detection methods, such as Fast R-CNN, which cannot detect wider contexts as it is restricted to region proposals. Furthermore, YOLO allows for real-time prediction because the image is processed only once by the CNN, and it is also more than twice as accurate as other real-time detection systems and generates fewer false positives [21].

2.2 Object Detection in Artworks

Style Transfer and Style Prediction. Smirnov et al. [24] propose a transfer learning approach to object detection in artworks. They stylize photos from the PASCAL VOC dataset [5] using artistic reference images and fine-tune a pre-trained object detection model using these painting-like images. Additionally, they use a style prediction network and fuse the high-level features of both models into a final classifier. Using this technique the authors report a mean average precision (mAP) on the transformed PASCAL VOC of 0.58.

Detecting Persons in Artwork. Westlake et al. [28] tried to detect people in paintings. They achieve an mAP of 0.58 on their dataset, called 'People-Art', by fine-tuning a VGG-16, which was pre-trained on ImageNet.

Region Proposals and MI-max. Gonthier et al. [8] present a weakly supervised approach using transfer learning with the help of residual networks (ResNets) [11] to detect objects in artworks. Their algorithm is able to only work with image-level annotations marking whether an object of a certain class is present in the image or not without knowing its exact location.

They use a Faster R-CNN, whose region proposal network serves as a feature extractor, from which they store 300 proposals (bounding boxes) with the highest objectness scores of an image and their feature maps. A linear classifier is then trained on these feature maps, which decides for each of the bounding boxes whether it belongs to the class annotated for the image at the image level. For training this classifier, Gonthier et al. [8] use a multiple instance learning loss. In this way, they want the model to learn to distinguish the bounding boxes with the highest scores of positively labeled images from the bounding boxes with the highest scores of negatively labeled images. They call this method *MI-max*.

3 Problem Statement

In this section we present the project task and the training data we relied on to solve it.

3.1 Prediction Target

Our goal is to train an object detection model which can correctly detect the classes listed in Table 1 in artworks.

Table 1: Classes of Objects to be Detected in the Images

Persons	*Animals*	*Produce*	*Structures*	*Landforms*	*Faces*
Angel	*Cow*	*Flower*	*Ruins*	*Water Bodies*	
Child Jesus	*Cat*	*Fish*	*Ship*	*Hills*	
Crucifixion	*Cock*	*Meat*	*Building*	*Mountains*	
Mary	*Dog*	*Fruit*	*Others*	*Others*	
Nudity	*Horse*	*Others*			
Saint Sebastian	*Duck*				
Others	*Others*				

The categories *Persons*, *Animals*, *Produce*, *Structures* and *Landforms* serve as high-level categories while the classes belonging to them are finer sub-divisions, so we distinguish between a *coarse* classification (e. g. *Animals*) and a *fine* classification (e. g. *Cow*, *Cat* or *Cock*). The classes *Others* are meant as a default fallback: Whenever an object does not fit into any of the classes, it is classified as the corresponding category.

3.2 Data Description

The dataset we are working with contains 4722 images provided by the Getty Research Institute.[2] The dataset is not annotated for object detection, which means that we have to rely on additional data for model training. For evaluation we manually constructed a test dataset from these images consisting of 156 images.

Since the most applications of machine learning are often based on photo-realistic images, finding adequate data sources in the artwork domain can be difficult. According to Milani and Fraternali [19], some prominent painting-centric examples include the Art500K [17], Rijksmuseum [18], Multitask Painting 100K [1], WikiArt [26] and IconArt [8] datasets. Of these, only the IconArt dataset provides instance-level bounding box annotations for a small part of its images, while most other available datasets only offer image-level annotations, which are not as helpful concerning the goal of object detection. Furthermore, the IconArt dataset only covers some of our target classes, namely the *Persons* category. For the training of these classes we partially used the IconArt dataset.

[2] `https://www.getty.edu/research` (last accessed June 16, 2021).

We give a complete overview of the annotated datasets we used for training in Table 2. As annotated art data is so rare except for IconArt all of them are photographic datasets.

The Google Open Images (GOI) dataset [13] is a large open source collection of over 9M images for 600 classes including bounding box annotations. However, the number of images per class varies greatly between classes.

A second dataset containing annotated photographs is the COCO 2014 dataset [16], which also provides some usable class images.

One further source of instance-level annotations is the ImageNet dataset [4]. It contains some examples of classes that are hard to find elsewhere, for example geographical features such as *Water Bodies*, *Hills* or *Mountains*.

The intersection of classes contained in the different annotated datasets and those that should be detected in the artworks is depicted in Table 2.

While some classes, such as those of the different animals, transfer well from recently-taken photographs, classes such as *Ship* or *Building* do not only contain the sailing boats or castles and cottages contained in the artworks, but also modern container ships and contemporary architecture. This data quality constraint should be kept in mind when training models on these classes later on.

4 Approaches and Methods

In the following, we present three different approaches for detecting objects in artworks. All these approaches rely on ensembles but use different methods to generate and merge their predictions.

4.1 ResNe(X)t and WBF Ensemble

The first approach is to combine the predictions of multiple different network architectures. We used Detectron2, a computer vision framework developed by Wu et al. [29], which is based on PyTorch [20] and is commonly used for object detection and image segmentation tasks. It provides a simple training pipeline that is designed for transfer learning based on a variety of network architectures. The available baseline weights have been trained on the COCO dataset [16] and include Faster R-CNN [23], RetinaNet [15], as well as RPN and Fast R-CNN [6] and ResNeXt [30] architectures.

"Artifying" Training Augmentations. Given that the majority of the classes to be detected in the artworks is only found in photograph-based annotated datasets, we tried to augment these photorealistic source images to make them more similar to the artworks of the prediction domain. In order to achieve this, we supplied custom functions to the augmentation wrappers provided by Detectron2 during the training phase.

The OpenCV library [2] offers four different functions that were used for this kind of style transfer: `edgePreservingFilter`, `pencilSketch`, `stylization` and

Table 2: Dataset Sources for the Prediction Targets

Category	Classes	GOI	COCO 2014	IconArt	ImageNet
Animals	*Dog*	7714	4562	×	×
	Cat	6348	4298	×	×
	Duck	4737	×	×	×
	Horse	6312	3069	×	×
	Cow	1147	2055	×	×
	Cock	×	×	×	474
	Others	6211	9483	×	×
Produce	*Fruit*	5762	5792	×	×
	Meat	×	×	×	×
	Fish	5790	×	×	×
	Flower	8336	4624	×	×
	Others	7709	6823	×	×
Persons	*Angel*	×	×	261	×
	Child Jesus	×	×	313	×
	Crucifixion	×	×	107	×
	Mary	×	×	446	×
	Nudity	×	×	403	×
	Saint Sebastian	×	×	82	×
	Others	×	66808	×	×
Structures	*Ruins*	×	×	114	×
	Ship	6567	3146	×	×
	Building	6640	×	×	×
	Others	×	×	×	×
Landforms	*Water Bodies*	×	×	×	1384
	Hills	×	×	×	87
	Mountains	×	×	×	133
	Others	×	×	×	×
Sum		73282	110660	1726	2078

`xphoto.oilPainting`. All of these transform the input image in a way that makes them appear more like some form of artwork and thus "artify" an input photograph. The last two transforms make photographs appear like water colour and oil paintings respectively, and thus are most likely to match the target images. These two were therefore given a higher probability to be applied through weighting, as shown in Table 3.

Given that a large number of the artworks in the Getty dataset are greyscale images, we also reduced saturation to zero with a high probability (80%).

We found that in most training runs, the "artification" did not lead to a large increase in detection accuracy, but that it could provide a small boost if applied sparingly. This lead us to apply this type of augmentation with a probability of only 20%.

Table 3: Augmentations Applied to "Artify" Training Images

Augmentation	Weight	Transform	Subweight
"Artify"	0.2	`edgePreservingFilter`	0.05
		`pencilSketch`	0.05
		`stylization`	0.2
		`xphoto.oilPainting`	0.7
Greyscale	0.8	Reduce saturation to 0	1.0

Repeat Factor Sampling. Working with imbalanced training data is a common challenge in machine learning. The niche status of some of the prediction targets, such as particular saints in religious images means that they are underrepresented in the training data. We employed is repeat factor sampling (RFS) as proposed by Gupta et al. [10] to address the problem. This resampling strategy increases the number of infrequently occurring classes shown to the neural network by oversampling them to a certain degree. While they apply it in a case where the most infrequent classes make up less than 0.1% of all instances, our least frequent classes occur more often relative to the most frequent ones. We used a repeat factor threshold of 5%, which we applied using Detectron2's `RepeatFactorTrainingSampler` class. This ensures that, classes that make up less than 5% of the training data are oversampled to such a degree that they make up 5% of all instances shown to the model during training. The specific values used are shown in Table 4.

Model Training and Selection. Since we did not have any instance-level annotated data to use as a validation set with which to judge the quality of the trained models, we decided to hand-pick a suitable set of the unlabelled Getty images for this purpose. The subset of "interesting" images that we chose covered at least one example of each image genre as well as some edge cases that seemed hard in an object detection sense.

Then, we fine-tuned a variety of pre-trained network architectures found in the Detectron2 model zoo [3] on our training set assembled from the sources listed in Table 2. We decided to select three different architectures with the best performance on the COCO Object Detection baseline [16]. These are all Faster R-CNN approaches with ResNet50, ResNet101 and ResNeXt101 backbones which use a feature pyramid network (FPN) [14] that further improves their performance.

During training, we mainly varied the learning rate, the RFS threshold and the content of the training dataset.

Next, we subjectively judged the performance of each fine-tuned network on our validation set. Here, the goal was to identify models with a good general performance, as well as models with a high accuracy for specific classes. In this step we found that including instances from the ImageNet dataset during training increased the number of false positives during validation dramatically. This is likely due to fact that the *Landforms* classes covered by ImageNet do not transfer well from photographs to artworks, even when applying the "artify" augmentations as mentioned before. We therefore decided to exclude all ImageNet instances from our training set.

For the final ensemble, we chose the qualitatively best-performing models on our validation set. We then also checked if the addition of any model further improved the predictions of the overall ensemble by doing predictions on the same validation set. The final list, shown in Table 4, includes three different model architectures trained with and without our "artify" augmentations as well as with different repeat factor sampling thresholds. All ResNe(X)t models employ FPN and were trained on training data from all sources in Table 2 except the ImageNet instances.

Table 4: Model Architectures and Parameters Used in the Final Ensemble

Model	"Artify"	RFS threshold	Conf. threshold
ResNeXt-101-32x8d		0.06	0.3
ResNet101	✓	0.4	0.15
ResNet101	✓	0.2	0.3
ResNet101		0.2	0.5
ResNet101		0.06	0.2
ResNet50	✓	0.06	0.2
ResNet50		0.06	0.2

Weighted Boxes Fusion. Solovyev et al. [25] present an alternative to non-maximum suppression (NMS) when ensembling bounding box predictions from multiple models. Instead of removing the subset of boxes with low confidence, their ap-

[3] `https://github.com/facebookresearch/detectron2` (last accessed June 16, 2021).

proach uses confidence scores of all proposed boxes in order to construct averaged bounding boxes and their corresponding confidence scores. This method is especially useful if the ensembled models predict slightly inaccurate boxes that still at least partly overlap on a ground truth target. In order to determine how to address overlapping boxes, an intersection over union (IoU) threshold has to be provided. To further improve performance, the models to be ensembled can be weighted by the user, making it possible to increase the influence of a well-performing model in the ensembling process. A general confidence threshold can be used to filter predictions with very low confidence. In the authors' testing, weighted boxes fusion (WBF) outperformed both regular and soft NMS. Judging by our chosen test images, we also found WBF to perform better than either NMS solution in our case. For the given challenge, we used their publicly available implementation [4].

In the context of our predictions, we used WBF to merge the predictions of our selected fine-tuned networks. We did not assign WBF weights to our models, but varied each one's confidence threshold for returning bounding boxes, as shown in Table 4. Generally, we used an IoU threshold for WBF of 0.4, meaning that any two boxes with an IoU ≥ 0.4 would be matched.

Post-Processing. To further improve our solution, we also applied some additional, manual tuning to deal with the peculiarities of our data. For example, we commonly found duplicate predictions of the fine classes on a single object, such as *Nudity* co-occurring with *Child Jesus* on the same detection. We addressed this issue by first mapping each class to its category (e. g. *Child Jesus* and *Nudity* to *Persons*). Then we used the WBF approach to calculate an averaged bounding box. To this new bounding box, we then assigned the label of the most confident prediction that was fed into the WBF step by mapping its coarse category back to its class label. An instance with the two predictions *Child Jesus* and *Nudity* thus received an averaged bounding box labelled as the more confident of the two.

We faced another issue with the classes *Flower*, *Fruit*, *Produce* and *Ship*. For these, our models would in some cases find sub-instances of a class inside a larger detection. While not necessarily wrong, we assumed that a bounding box encompassing the entire plant is semantically more sensible than a multitude of blooms labelled as such. One example are flowers, where smaller blooms or petals inside a larger bounding box were also labelled. In such cases, we simply removed the inner predictions and thus only kept the largest bounding box.

The final flaw that we decided to handle is associated with the *Building* class. Our models labelled entire scenes taking place indoors with this class in some cases. While these predictions do have some semantic validity, we did not view them as particularly meaningful, especially as they tended to cover almost the entire image in such cases. Therefore, this particular problem of whether a depicted scene might be located inside a building could be solved more efficiently with a classifier than in our object detection approach. As a result, we decided to prune these predictions

[4] `https://github.com/ZFTurbo/Weighted-Boxes-Fusion` (last accessed June 16, 2021).

by eliminating any *Building* predictions covering more than 85% of the total image area.

Face Detection Using MTCNN. A common task in computer vision is the detection of faces within images. A popular algorithm in this domain was developed by Zhang et al. [31], which enables face detection in real-time through a cascaded architecture of deep CNNs. In the case of face detection inside the artworks, we relied on Centeno's [5] implementation of a multi-task cascaded convolutional network (MTCNN). As this model is pre-trained mainly on photographs, we suspected that it might not necessarily transfer well to the art domain. However, we found its predictions to be reasonable in most cases and did not resort to fine-tuning of any kind. We also did not compare its performance to other network architectures or implementations. This could be an area of future research.

Given the pruned output of the previous steps, we then simply appended the predicted face bounding boxes without any further modification to yield our final predictions.

4.2 IconArt and Pre-Trained Ensemble

Another approach we pursued was an ensemble approach of self-trained and pre-trained models. For this, we trained some of the models ourselves using supervised training (only for the IconArt classes), but we also made use of pre-trained models (for the other classes). For example, we trained a common Faster R-CNN for almost all IconArt classes (except *Saint Sebastian*). Then we trained one Faster R-CNN each for the classes *Angel*, *Mary* and *Crucifixion*. We used NudeNet [6] for the class *Nudity*, YOLOFace [7] for the category *Faces* and for all other classes and categories, we used a Faster R-CNN pre-trained on the COCO dataset, a Faster R-CNN pre-trained on the Google Open Images dataset, and YOLO9000 [22]. The input image is processed by the individual submodels and possible bounding boxes are cached after a NMS is performed at the class level in each case. The resulting bounding boxes are then merged and filtered using further NMS.

4.3 YOLO Ensemble

In this ensemble approach, we do not aggregate different model architectures, but rather split the data into the categories (see Table 1) and train individual models on each of these separate categories.

The assumption is that object detectors are easier to train on a single category (e. g. *Animals*) rather than on a mixture of all target categories. After training individual models on all categories, the predictions get aggregated, filtering con-

[5] `https://pypi.org/project/mtcnn` (last accessed June 16, 2021).
[6] `https://github.com/notAI-tech/NudeNet` (last accessed June 16, 2021).
[7] `https://github.com/sthanhng/yoloface` (last accessed June 16, 2021).

tradicting labels of overlapping bounding boxes based on predefined rules. This approach also alleviates data imbalance concerns between categories.

Model Training. We used the YOLO v5 [12] model for each category. As mentioned in subsection 2.2 it is one of the state-of-the-art object detectors, which allow end-to-end training and do not rely on region proposals. This makes the model very fast during inference and well suited for ensemble prediction as we can run multiple models sequentially and still expect reasonable runtime performance.

For the classes of the category *Persons*, we trained the model on the annotated image from the IconArt dataset.

In addition to that we relied on a YOLO model pre-trained on the COCO dataset for general person detection (i.e. class *Others* in category *Persons*). Through manual assessment we determined that the IconArt model rarely produced false positives for person predictions, i. e. it did not classify general persons as religious characters. This let us use it as a special case for religious figures, whereas the pre-trained COCO model covered all other persons in an image.

For training on the categories *Animals*, *Structures* and *Produce* we used the data from the Google Open Images dataset. All models were trained for 30 epochs.

For the category *Faces* we relied on MTCNN as described above.

We left out the category *Landforms* entirely as we were not able to gather enough training data to train a whole separate model on it.

Post-Processing. The post-processing of the ensemble prediction involved filtering bounding boxes which overlapped with an IoU by more than a threshold (we chose 0.7 for all experiments) and removing the higher-level class prediction in favor of more specific classes. For example, if *Mary* is simultaneously classified as *Person* we would erase the latter.

The YOLO model output also includes a confidence measure. Our initial approach of using these values to filter low-scoring bounding box predictions did not work as the model was often underconfident on good predictions. We therefore did not perform any confidence filtering.

5 Experiments and Results

In this section we present detection results for all three approaches and compare them against each other in the end. In addition, we compare quantitatively based on our test dataset from the Getty dataset, as well as qualitatively through manual assessment of example images.

5.1 ResNe(X)t and WBF Ensemble

As stated in subsection 4.1, we validated the individual models and the complete ensemble model only qualitatively on a set of hand-picked images. Generally, the predictions were quite accurate. We found that overall prediction quality was better

when leaving out classes representing geographic features, such as water bodies or hills. We only had instances for these classes from a single, photorealistic source, ImageNet. Including these instances in the training process lead to much worse results on our validation set. We therefore did not train on any of the instances taken from ImageNet.

In cases where we found significant and systematic prediction errors that were simple to fix, we applied the post-processing mentioned in subsection 4.1.

Generally, the approach of relying solely on qualitative validation on a very small set of hand-picked images as opposed to a quantitative evaluation on a larger, instance-level annotated validation set did not result in a significantly worse performance on the hold-out test set, as visible in Table 6.

5.2 YOLO Ensemble

For the category-based ensemble of multiple YOLO models, we evaluated the performance of each individual model per category by using a validation dataset extracted from the Google Open Images and IconArt data respectively (50% validation split for Google Open Images, 20% for IconArt). Table 5 shows the mean average precision for an IoU of 0.5 (mAP@0.5) for all classes in a category.

Table 5: Results on Domain Models (mAP)

Category	mAP@0.5
Persons (IconArt)	0.63
Animals	0.82
Structures	0.52
Produce	0.34

The results show a varying performance of the category-specific models. The category *Produce* scored the lowest performance with only 0.34 mAP. Manual examination of the data in that category showed very poor annotation quality for some images. We believe this to be the reason for the low performance.

We are unsure of the reason behind the performance of the *Structures* model. As mentioned before, these results cannot be easily transferred to the art domain, which is why an even lower AP is to be expected on the Getty data.

The model for *Animals* performed quite well. The same can be said for IconArt, even though its AP is quite a bit lower, because the classes in the IconArt data are much more complicated in a semantic sense. It is for instance very difficult to differentiate *Saint Sebastian*, who is always depicted without clothes, from *Nudity*. The same goes for telling apart all of the religious figures from regular persons appearing in the same image. In addition to that, the model also had much less training data available.

5.3 IconArt and Pre-Trained Ensemble

Before we decide to use the ensemble approach as mentioned in subsection 4.2, we pursued the MI-max classifier approach by Gonthier et al. [8]. With this approach, we did not achieve the results as Gonthier et al. (for example results see Figure 1). Objects were often false positive classified and so we decided against this approach.

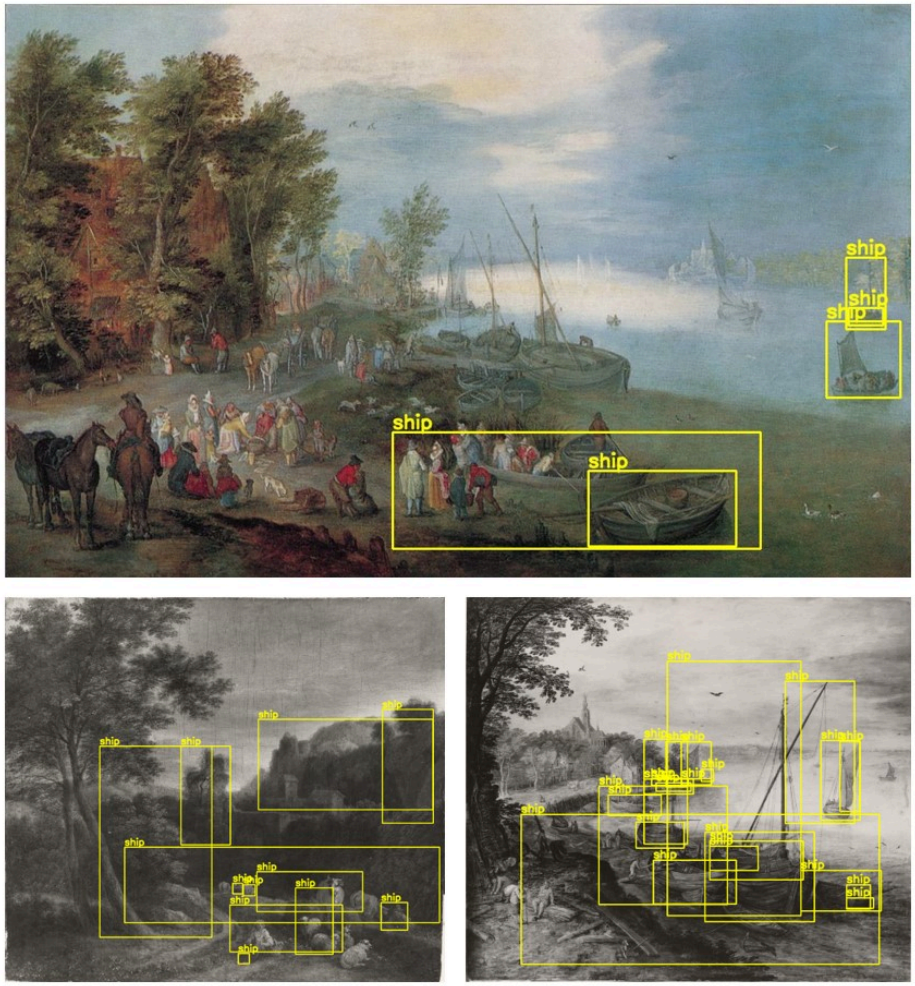


Figure 1: Example images with the results of the predicted bounding boxes using the MI-max classifier for the class *Ship*. On the top, ships are correctly classified, but on the other two images (below) there are a lot of false positive classified ships.

Due to the lack of bounding box labels, evaluation for the IconArt and pre-trained ensemble approach was only possible at image level (except for the IconArt classes). For the IconArt classes, we could evaluate the precision of our approach with IoU. For example, we achieved a mAP@0.5 of 0.25 for the *Crucifixion* class. The precision on the hold-out test set can be seen in Table 6.

5.4 Evaluation on Hold-out Test Set

The images in our test set were manually annotated with bounding boxes for the target detection task.

Table 6 shows the mAP results for all three approaches on this test data. Both the YOLO and ResNe(X)t and WBF methods outperformed the IconArt/pre-trained approach by a large margin (e. g. 12.9% vs. 4.3% for class labels AP@0.5). Out of all models the ResNe(X)t ensemble produced the best results. Worth noting is the large drop in precision between IoU of 0.5 and 0.75. This hints to very inaccurate bounding boxes compared to the ground truth. We assume this inaccuracy to originate in large objects in the images, e. g. buildings or ruins, as these tend to have a much larger play in size and shape. These findings can be confirmed when looking at the example images as well (e. g. Figure 3e).

The effectiveness of the post-processing steps of the ResNe(X)t and WBF approach is evident in Figure 2: while the other two approaches draw too many bounding boxes, only the first approach yields the desired prediction, although it is slightly out of place. However, this method seems to reduce performance in the case of Figure 3f, where some ships are omitted that might otherwise have been detected. Furthermore, the post-processing step of always mapping back to the specific class does not seem to work well in the case of buildings, where a *Ruins* label seems much more likely to be assigned than a generic *Building* label.

The IconArt detection works quite well in most cases, as can be seen for the YOLO and the weighted-box fusion ensemble models in Figure 3a. Persons also work very well, even though in larger crowds (see Figure 3b) not all persons are detected.

In many cases the models do not recall all of the bounding boxes and often also have overlapping predictions. This especially a problem with the IconArt/pre-trained model, which explains its weaker performance.

Some annotated objects are arguably very hard to distinguish, even for humans. In these cases poor model performance is to be expected. An example for this are birds or very small animals, e. g. in Figure 3e.

Table 6: AP Results on the Getty Test Set

Ensemble	Class labels		Category labels	
	AP@0.5	AP@0.75	AP@0.5	AP@0.75
YOLO	11.5%	**4.6%**	18%	4.5%
ResNe(X)t and WBF	**12.9%**	3.8%	**22.8%**	**5.9%**
IconArt/pre-trained	4.3%	1.1%	8.5%	1.8%

Figure 2: Example images of results; top-left: ground truth, top-right: ResNe(X)t and WBF ensemble (1), bottom-left: YOLO ensemble (2), bottom-right: IconArt/pre-trained ensemble (3). Model 1 predicts nearly the same as the ground truth. Model 2 predicts individual flowers, while model 3 predicts both individual flowers as well as many of their petals.

6 Discussion

Taking the complexity of the task (being cross-domain over 22 classes) into account an AP@0.5 of 12.9% for the best model for fine classification is still very satisfactory. The approaches named in related work achieved 58% for single domain use cases, ours is much more difficult.

We attribute the limitations of our approaches mainly to the lack of data. Even though we already made use of transfer learning because of the rarity of artistic data, we still ended up short of training examples for some classes of which not enough suitable photographic material existed. Datasets for some of the categories were simply too small or of too poor quality. Qualitative analysis showed that some categories performed therefore much better than others. These are specifically persons and animals as these are common target classes in computer vision tasks. The off-the-shelf face detector worked also great on our art data. *Produce* and *Landforms* however were a lot harder to predict. Further research and data synthesis in these categories would likely have a great impact on the results.

7 Conclusion and Future Work

After extensive research on existing approaches to object detection on art data, we implemented and adapted some of them. The results were however not satisfactory. Thus, we turned to different pre-trained object detection models and built ensemble predictors by fine-tuning them on different photographic datasets. We covered almost all of the predefined 22 classes and achieved promising results on some of the categories, especially considering the complexity of the task at hand. The results are by far not optimal yet, requiring further work in some categories, like detecting *Landforms* and *Produce*.

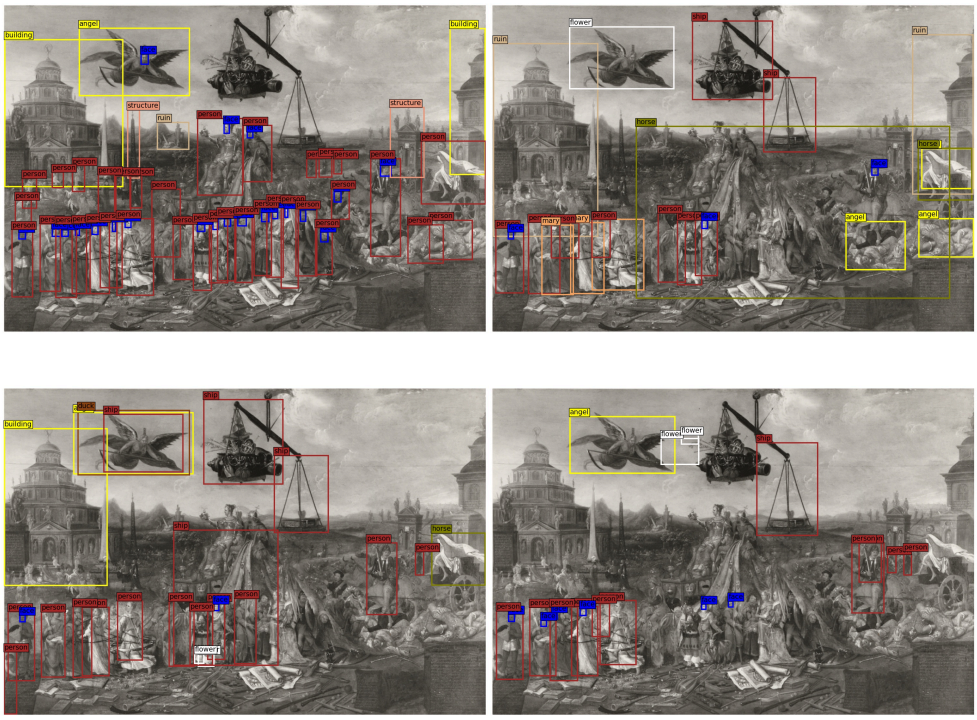
Building more datasets with annotated bounding boxes is very important for greater success in the area of object detection. Photographic datasets are a great start and it can be shown that they can be transferred to art data. However, for the art analysis community more effort should be put into annotating existing art collections.

A Appendix

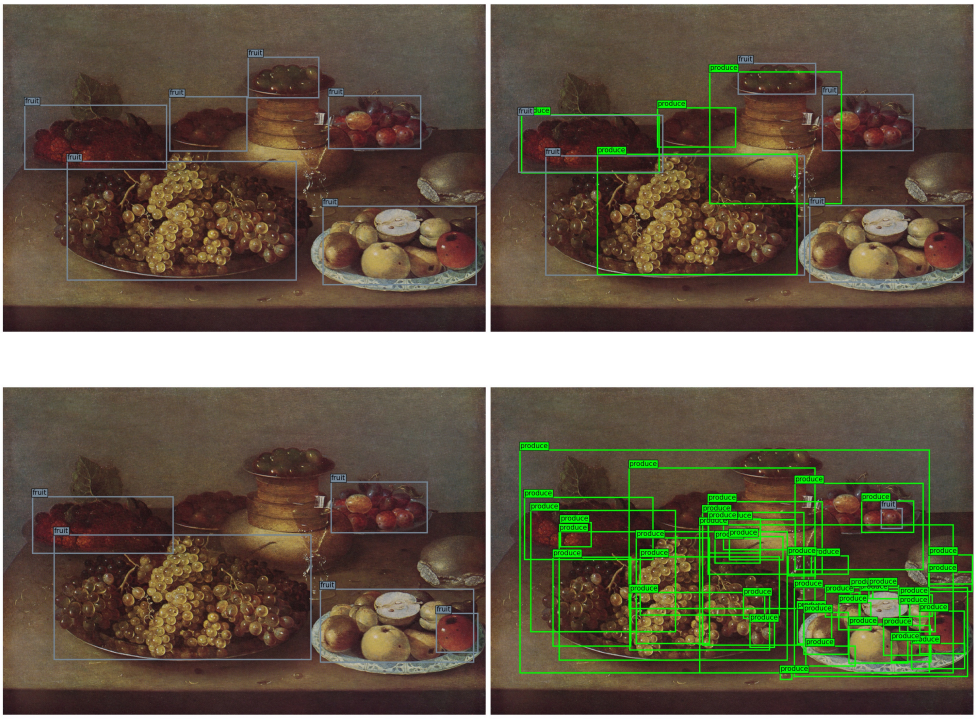
In the appendix, we list additional example images.

Figure 3: Example images of results; respectively top-left: ground truth, top-right: ResNe(X)t and WBF ensemble (1), bottom-left: YOLO ensemble (2), bottom-right: IconArt/pre-trained ensemble (3).

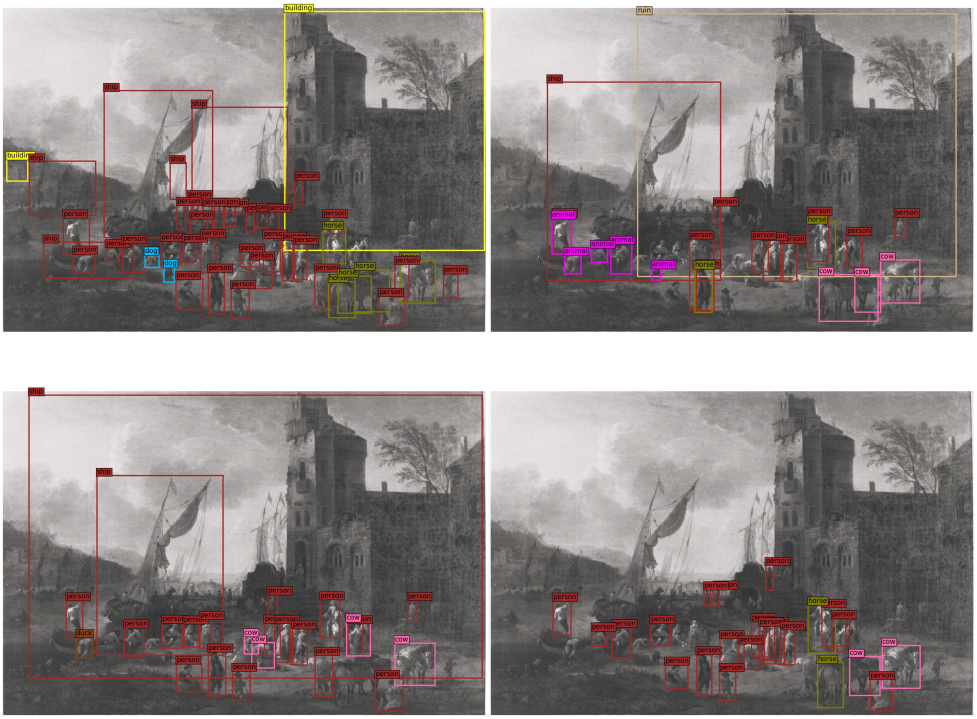
(a) Model 1 summarizes many of the flowers to one bounding box and the other two models detect individual flowers. Each model detects the woman as *Mary*. Models 1 and 2 also detect the baby as *Jesus*, although neither figure is labelled with this class label in the ground truth. Only model 3 detects both faces.



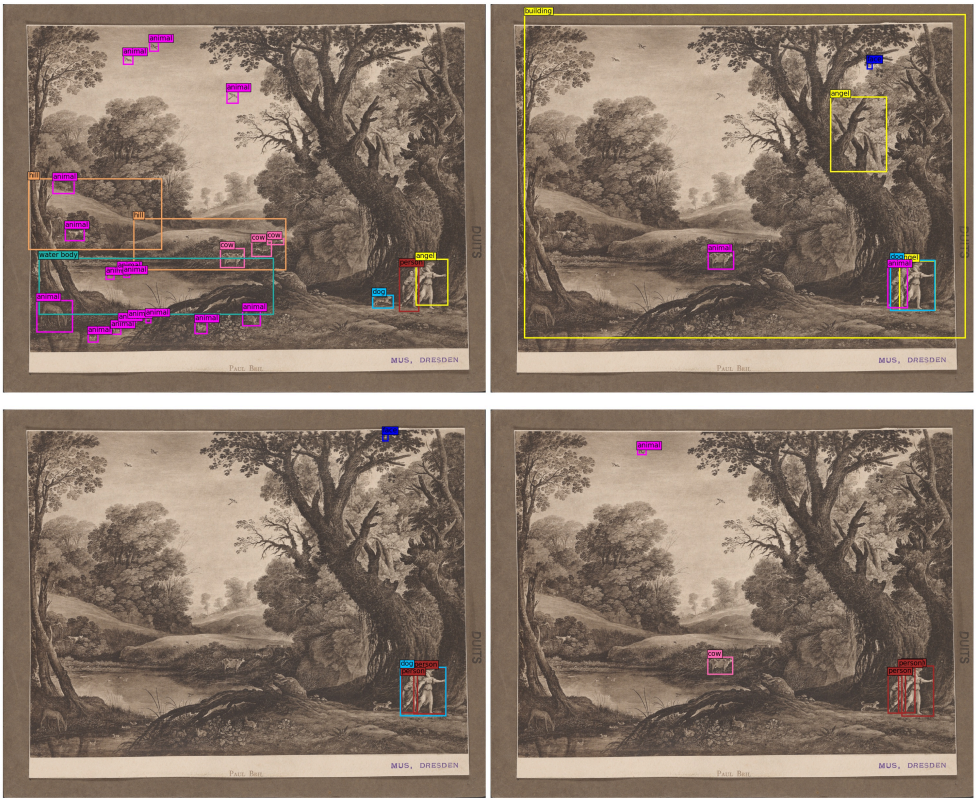
(b) Many persons are annotated. None of the models detect all of them, model 2 detects the most. The same is true for face detection. Model 1 detects the buildings as *Ruins*. The *Angel* in the sky is correctly classified by model 2 and 3, however model 3 also detects it as a *Ship* and model 1 as a *Flower*. The scale is detected as two ships, which it strongly resembles.



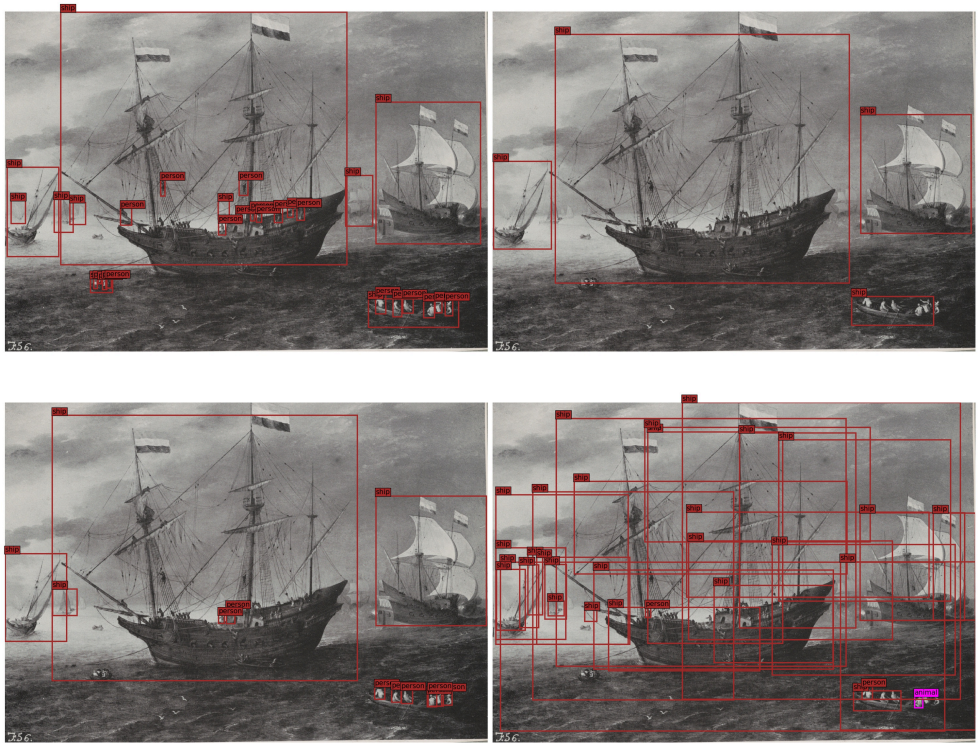
(c) Model 1 has only some overlapping bounding boxes but its predictions are near by the ground truth. Model 2 predicts only *Fruit* but misses some. Model 3 has many overlapping bounding boxes and mostly detects the category *Produce* instead of the correct class *Fruit*.



(d) Models 1 and 2 detect one of the ships and predict some persons as animals. Model 3 detects no ships. All models found some cows in the image but there isn't any in the ground truth and none of the models recognize the buildings or dogs.



(e) Model 1 is the only one to detect the *Angel* correctly, but classifies a large section the artwork as a *Building*. The other two models detect the *Persons* correctly and also recognize the *Angel* as such. There are many animals in the image, but every model detects only very few (not necessarily correct) animals.



(f) Model 1 predicts the big ships correctly but does not detect the small ships or any persons. Model 2 detects some ships and persons but not all. Model 3 manages to detect the small ships, but it has a lot of overlapping bounding boxes and it predicts a person as an animal (so category *Animals*).

References

[1] S. Bianco, D. Mazzini, P. Napoletano, and R. Schettini. *Multitask Painting Categorization by Deep Multibranch Neural Network*. 2018. arXiv: 1812.08052 [cs.CV].

[2] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[3] M. Cornia, M. Stefanini, L. Baraldi, M. Corsini, and R. Cucchiara. "Explaining digital humanities by aligning images and textual descriptions". In: *Pattern Recognition Letters* 129 (2020), pages 166–172.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE. 2009, pages 248–255.

[5] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) challenge". In: *International Journal of Computer Vision* 88 (June 2010), pages 303–338. DOI: 10.1007/s11263-009-0275-4.

[6] R. Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[7] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524.

[8] N. Gonthier, Y. Gousseau, S. Ladjal, and O. Bonfait. "Weakly Supervised Object Detection in Artworks". In: *Computer Vision – ECCV 2018 Workshops* (2019), pages 692–709. DOI: 10.1007/978-3-030-11012-3_53.

[9] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, and G. Wang. "Recent Advances in Convolutional Neural Networks". In: *CoRR* abs/1512.07108 (2015). arXiv: 1512.07108.

[10] A. Gupta, P. Dollár, and R. Girshick. *LVIS: A Dataset for Large Vocabulary Instance Segmentation*. 2019. arXiv: 1908.03195 [cs.CV].

[11] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[12] G. Jocher et al. *ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration*. Version v4.0. Jan. 2021. DOI: 10.5281/zenodo.4418161.

[13] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, and et al. "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale". In: *International Journal of Computer Vision* 128.7 (2020), pages 1956–1981. DOI: 10.1007/s11263-020-01316-z.

[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV].

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[16] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[17] H. Mao, M. Cheung, and J. She. "Deepart: Learning joint representations of visual arts". In: *Proceedings of the 25th ACM international conference on Multimedia*. ACM. 2017, pages 1183–1191.

[18] T. Mensink and J. van Gemert. "The Rijksmuseum Challenge: Museum-Centered Visual Recognition". In: *Proceedings of International Conference on Multimedia Retrieval*. Association for Computing Machinery, 2014, pages 451–454. DOI: 10.1145/2578726.2578791.

[19] F. Milani and P. Fraternali. *A Data Set and a Convolutional Model for Iconography Classification in Paintings*. 2020. arXiv: 2010.11697 [cs.CV].

[20] A. Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[22] J. Redmon and A. Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV].

[23] S. Ren, K. He, R. Girshick, and J. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[24] S. Smirnov and A. Eguizabal. "Deep learning for object detection in fine-art paintings". In: *2018 Metrology for Archaeology and Cultural Heritage*. Oct. 2018. DOI: 10.1109/MetroArchaeo43810.2018.9089828.

[25] R. Solovyev, W. Wang, and T. Gabruseva. "Weighted boxes fusion: Ensembling boxes from different object detection models". In: *Image and Vision Computing* 107 (2021), page 104117. DOI: 10.1016/j.imavis.2021.104117.

[26] W. R. Tan, C. S. Chan, H. E. Aguirre, and K. Tanaka. "Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification". In: *2016 IEEE International Conference on Image Processing*. IEEE. 2016, pages 3703–3707.

[27] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. "Segmentation as selective search for object recognition". In: *2011 International Conference on Computer Vision*. 2011, pages 1879–1886. DOI: 10.1109/ICCV.2011.6126456.

[28] N. Westlake, H. Cai, and P. Hall. "Detecting People in Artwork with CNNs". In: *CoRR* abs/1610.08871 (2016). arXiv: 1610.08871.

[29] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[30] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. arXiv: 1611.05431 [cs.CV].

[31] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks". In: *IEEE Signal Processing Letters* 23.10 (2016), pages 1499–1503. DOI: 10.1109/LSP.2016.2603342.

[32] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu. "Object Detection with Deep Learning: A Review". In: *CoRR* abs/1807.05511 (2018). arXiv: 1807.05511.

[33] Z. Zou, Z. Shi, Y. Guo, and J. Ye. "Object Detection in 20 Years: A Survey". In: *CoRR* abs/1905.05055 (2019). arXiv: 1905.05055.

Image Retrieval

Tobias Bredow, Nicolas Alder, and Martin Büßemeyer

Hasso Plattner Institute for Digital Engineering, University of Potsdam
{tobias.bredow,nicolas.alder,martin.buessemeyer}@student.hpi.de

Over the last years, gigantic art libraries were digitized. This however raises the issue of retrieving certain images from those archives. One solution is content-based image retrieval (CBIR), this allows to search for artworks by providing similar paintings or photos. We propose and evaluate several existing CBIR methods and apply them to a set of photos of artworks that only have genre and category labels. In doing so we observed that bag of visual words and variational autoencoders are outperformed by an approach based on embeddings generated by the popular ResNet deep neural network. We added a co-occurrence filter to the ResNet embedding that had shown promising results in image retrieval. Though our experiments revealed that the addition leads to inferior results. This work tests fundamental approaches in CBIR regarding a dataset without ranking annotations.

1 Introduction

Due to the advancement in technology and the large-scale digitization of artworks, new challenges regarding the accessibility of such large datasets. For example researchers that try to find similar artworks in vast digitized archives mostly need to manually search for each similar image. Therefor it is desirable that computers could automatically search for similar images. However, finding desired images in large-scale collections such as WikiArt[1] can be a challenge. In the past algorithms mostly relied on metadata and tags and not on certain features of artworks, features describe certain characteristics of artworks and can be used to identify similar images. In practice, it is hard for a human to carefully describe the desired features and visual content of the desired artwork or use low-level visual features to describe an artwork's concept and art style. Therefore, it is difficult for search algorithms to find the desired images. One approach to retrieve similar images is content-based image retrieval. Various methods aim to find similar images with the help of machine learning algorithms which we will focus on in this work. In total, we will look at four different approaches variational auto encoders, a ensemble of a certain object detection network, bag of visual words and a model based on the popular ResNet. In order to compare our approaches we will first train them on

[1] https://www.wikiart.org/ (last accessed June 16, 2021).

the same dataset and for evaluation a group of 20 people will rate the images that were retrieved by each network on their similarity.

In Section 2, we give an overview of related work done in the field of content-based image retrieval for artworks. Section 3 gives an overview of the problem statement that we tried to assess with our project. Afterward, Section 4 provides an overview of the different approaches that we examined, while Section 5 provides details of the conducted experiments. In the following, Section 6 describes the results we achieved with each of our approaches. Section 7 discusses the advantages and disadvantages as well as possible reasons for the shortcomings of certain techniques. In the end, Section 8 gives a conclusion and outlook into possible future work.

2 Related Work

The problem of content-based image retrieval has already been researched for the last two decades. The first approaches were based on hand-crafted descriptor-based image retrieval [30] which lacks the generalization ability, this means it performs poorly on data that is different from the original training data. This refers to the capability of achieving almost equal performance on unseen data. In the last decade, a shift in content-based image retrieval happened due to the rise of deep learning. Razavian et al. [21] showed that convolutional neural network (CNN) representations can outperform the state-of-the-art hand-crafted image features for retrieval. CNNs can deal with rotation and translation invariance off-the-shelf [9] that are necessary capabilities for the given problem. Thus, CNNs started to replace other established methods for representing image data [6].

A sub-problem of image retrieval is the detection of instance-level features where reappearing parts of images should be detected. Gordo et al. [13] created an architecture that can generate a global descriptor in a single forward pass with excellent performance. A self-supervised learning pipeline for automatic reappearing pattern recognition in art collections has been developed by fine-tuning the network based on the spatial consistency between neighboring features [23].

Different categories of approaches emerged over the years. A category consists of using variational and non-variational autoencoders [17] that are trained to compress and reconstruct an image. The compressed latent vector is used as the image descriptor. Another idea is to generate hash codes from images and use these hash codes to compute the similarity between two images [16]. A further approach generates embeddings from CNN architectures like ResNet [14] or VGG [24] that were invented for classification tasks. The idea is to take a feature map of the last convolutional layer and use it as a high-level image representation. Castellano et al. [4] used transfer learning on a VGG16 by taking that higher-level features. Afterwords, they compressed the features with Principle Component Analysis (PCA) [7] to obtain a compact feature vector. We combine the feature extraction and compression by simply taking the output of the global average pooling layer of a ResNet. Forcen et al. [8] suggested a strategy in which they use a special co-occurrence

filter to enrich the embedding which surpassed state-of-the-art performance on multiple benchmark datasets. They directly trained their network for the retrieval problem using e.g. the Oxford dataset [20]. We want to explore if the co-occurrence approach can achieve good results when the network is trained on a classification task instead.

Image retrieval for art can use the research results of general photorealistic image retrieval. Though there are some important differences. Using a pre-trained network without fine-tuning will lead to inferior results because of the difference in photorealism and art. The quality suffers due to a lack of annotated data that could be used for training. Therefore, fine-tuning and transfer learning are promising techniques. Fine-tuning means the usage of a pre-trained network on a different dataset and adapt the learned features to the given dataset by only training a few epochs on the given dataset and optionally freezing the first few layers of the network. This can lead to good results when only a few annotated data is available. Seguin et al. [22] propose the automatic construction of a graph of visual links between art paintings which is a specialization of our topic. They achieved the best performance by fine-tuning a VGG16 that was trained on the Imagenet dataset and using the feature map of the last convolutional block and applying a sum-pool followed by PCA. They also observe inferior results when taking the output of a fully connected layer as the feature map. García et al. [10] improved the CNN embedding approach by adding a context embedding from a knowledge graph for art-specific knowledge to raise the performance of image retrieval for photos of artworks. Since we do not have such a knowledge graph, we could not evaluate the gain of such an extended embedding. Therefore, we will focus on using the output of convolutional layers as well.

3 Problem Statement

Content-based image retrieval is the problem of finding images that resemble a described content, provided through text or similar images, from a image corpus for a search request or the most related images based on a keyword. In this work we will only address the first aspect. It is supposed to be performed exclusively based on the visual information within the images. CBIR involves two fundamental difficulties. The first difficulty is the semantic gap. Smeulders et al. [26] define it as *"the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation."* The presented methods can only address this problem. However, necessary contextual knowledge for the similarity must be a visual part of the image to be regarded. For example, the period of creation of a work of art cannot be included directly as a similarity measure. Only an image's style might indirectly give away the period. The second problem is called the intention gap and refers to users' inability to formulate a precise search query. Imprecision might be caused by the ambiguity of language or a lack of notions of what one intuitively considers similar. Since neither is a universally binding definition of similarity nor can perceptions differ from person

to person. The art domain in which we operate is rich in complex and ambiguous concepts that can be used for similarity assessment. Since our image corpus consists exclusively of historical paintings and drawings, domain-specific aspects such as artist, style, content, period, or iconography are suitable as similarity aspects.

To address the intention gap in the best possible way, we use a query by example approach. For each method, we return five similar images, that resemble the query image in style and content, for a given query image that are then manually evaluated for visual similarity. The query images for testing are each randomly chosen from our image corpus. By manual inspection we ensured that the test data contains a diverse set of images concerning the evaluation aspects. We evaluate the similarity of similar images, based on subjective guidelines and majority voting in a group of students, that are returned by our methods concerning similar style (painting, drawing), similar colors, depicted people, animals or objects, the art historical classification in history, portrait, genre, landscape and still lifes and possible individual peculiarities, such as unique cuts of images. The evaluation takes place through a visual inspection by ourselves. Since we do not have a background in art history and our idea of similarity may differ from domain experts' ideas, this is a possible limitation of our evaluation. For the training and test of our methods, we use exclusively the Getty data set as image corpus and no external data. We extracted some images as test images and used the rest for training and building the image database. Therefore, a test is done by taking a test image and retrieve the result image from all non-test images.

4 Approach

Since our approach is based on finding the nearest neighbors in a high dimensional feature space, by feeding the images to some form of an encoder that represents the images in the mentioned feature space. This section is focused on the different types of encoders we tested. In total there were 4 different methods testes an approach based on variational auto encoders, one based of bag of visual words, an ensemble of YOLO object detection models and encoding generated from a ResNet.

4.1 Variational Autoencoder Feature Extraction

Autoencoders are characterized by two components, an encoder and a decoder. The encoder component's objective is to transform a high-dimensional input into a low-dimensional representation. This representation is referred to as latent space. The decoder's objective is to reconstruct the original input from the low-dimensional representation. The loss function utilizes the reconstruction error, reducing the difference between the reconstructed image and its original input. Autoencoders are employed for image retrieval [3, 18] but also for tasks such as image denoising [12], anomaly detection [1], image generation[29] or deepfake detection [5].

The variational autoencoder (VAE) was introduced in 2013 by Kingma and Welling [17]. While a traditional autoencoder maps an input directly to a fixed

point within latent space, a variational autoencoder maps to a normal distribution and incorporates random noise. A term is included in the loss function, usually the Kullback-Leibler divergence, to penalize too large deviation from the normal distribution. This promotes the low-dimensional space being symmetric through normal distributed mapping as well as nearby regions representing similar images through random noise employment, which might be beneficial for image retrieval. To utilize a variational autoencoder for image retrieval, we train the network on an image corpus. To retrieve similar images for a query image, we first extract all latent space features for our corpus. Subsequently, we compare all latent space feature maps, which means the outputs of the different filters of each layer, to the query image feature map for a given query image. We assess the five images closest to the query image in the latent space, using a k-nearest-neighbor approach, and return them.

4.2 Bag of Visual Words (BOVW)

A bag of visual words [25] represents prominent image features and can be utilized to find similar images. The approach consists of two steps. First, each image's interesting features from the image corpus are manually identified (e.g., humans, animals, trees, houses), counted, and saved within a histogram. One interesting feature is called a visual word, and the histogram is referred to as a bag of visual words. Second, if we want to retrieve similar images for a given input, we compare all bags of visual words and return the most similar ones.

4.3 YOLO Ensemble

Another approach to solving the task at hand is to use a multitude of YOLOv5 Models [11]. The YOLOv5 architecture is a CNN-based model that is designed to detect objects in images as well as assign a bounding box for each of the detected entities. It accomplishes this task with a CNN based feature extractor that extracts objects from the given images. These objects are then fed into a prediction module which try to predict their class and draws a bounding box that surrounds the given object. By training a multitude of these models on a group of objects the goal is to cover a wide variety of possible variations of objects in artworks. In order to create a single embedding that represents the result of all the different models, we concatenate all the embedding vectors that were extracted from each YOLO model. Therefor the resulting embedding should represent all objects that were detected by each of the different models. Since images with similar objects are also similar to each other, retrieval can be done by comparing these embeddings to each other using cosine similarity.

4.4 ResNet Feature Extraction

One challenge in content-based image retrieval is to gain a higher-level understanding of each image and select the images that are the closest to the given query

image. We use a Convolutional Neural Network (CNN) for the generation of embedding vectors of images. CNNs can learn to recognize features at different levels of image abstraction.

As Zeiler et al. [28] have shown, CNNs tend to learn low-level features such as recognizing edges or color gradients in the first layers of the architecture. The deeper a layer lies in the chosen architecture, the more high-level are the learned features, such as from edge to animal detection.This concludes that the output of deep layers of a CNN, called feature maps, can be used as a high-level image representation, which we can use for our image retrieval problem. Other researchers already adopted this idea [8, 10, 27].

We decided to use the ResNet architecture as our feature generator since it is a model that generally performs well in classification tasks and we therefor assume that it is able to learn a good representation of certain features in the given images. We tried two different approaches to extract the higher-level representation. One takes the output of the average pooling layer and the other takes the output of a convolutional layer and applies further processing in the shape of a co-occurrence filter.

Co-Occurrence Filter. Applying a co-occurrence filter to improve the results of content-based image retrieval was already done by Forcen et al. [8]. The co-occurrence filter idea is based on the understanding that different features are encoded in each feature map and that simultaneous activations along different feature maps in a local area can yield valuable information for image retrieval. It works by passing the image through the CNN and taking the activations of the last layer as a tensor (A). These are passed through the co-occurence filer resulting in C, A and C are then combined as shown in Equation 1. In the end the L_2 norm followed by PCA and another L_2 is applied.

We based our pipeline on the approach of Forcen et al. but adjusted the approach to better fit to our problem setting. Because the ResNet architecture's residual connections make training deep neural networks easier, we decided to use the ResNet18 architecture and not the VGG16. In the end we decided to use the bilinear pooling suggested by Forcen et al. [8] and skip other post-processing because we can have a large image descriptor. The bilinear pooling B is calculated by:

$$B(A, C) = \sum_{i}^{M} \sum_{j}^{N} A_{ij} \cdot (C_{ij})^T \tag{1}$$

Where C and $A \in \mathbb{R}^{M \times N \times D}$ resulting in a large tensor of $B \in \mathbb{R}^{D \times D}$.

The image database is created by calculating the pooled image descriptor for every image and storing them. For each query image, the descriptor Q is calculated in the same way by using Equation 1. Hereafter, we calculate the similarity S of an image descriptor I by the squared Frobenius norm which sums the squared difference of each feature of Q and I:

$$S = \sum_{i}^{D} \sum_{j}^{D} (Q_{ij} - I_{ij})^2. \tag{2}$$

Then we create a ranking R of all database images by sorting all S values descending. Finally, we return the first k images of the ranking R as the query results as the kNN of the query image.

5 Experiments

After introducing our approaches, we describe the experimental setups.

5.1 Variational Autoencoder Feature Extraction

While we tried various configurations due to hyperparameter tuning in the preliminary experiments, the following three setups for variational autoencoder (VAE) turned out promising and were incorporated for subsequent latent space feature extraction.

(i) as traditional autoencoder that samples to low-dimensional latent space.
(ii) maintains a high-dimensional latent space and focus on reconstruction.
(iii) maintains a high-dimensional latent space and focus on its symmetry.

Experiment (i) employs a traditional variational autoencoder aiming to distill all relevant properties of the high-dimensional input images into a low-dimensional latent space. We mapped an input image of size 64x64 (4 096) to a lower-dimensional latent space of size 1 024. After resizing and center cropping the image, we applied several data augmentations.[2] The objective of (ii) and (iii) is to examine if image retrieval profits from maintaining the input vector's high-dimensional structure in the latent space. While (ii) focuses on structuring the latent space with a larger weight on the reconstruction loss, (iii) concentrates on a latent space that is heavier centered towards the normal distribution. Both are mapping an input image of size 280x280 (78 400) to a latent space with 72 900 dimensions.

The basic architecture of the variational autoencoder is defined for all experiments consistently. Let x be the input image vector, $z = Encoder(x) \sim q(z|x)$ is the latent space representation of x, and $\hat{x} = Decoder(z) \sim p(x|z)$ is the reconstructed image of x. For all experiments, the Encoder and Decoder utilize five sequential components. As the dimensionality of the latent space should remain the same as the input image's dimensionality in (i), we changed the filter output size of the last component of the encoder (first of the decoder) back to the initial filter size

[2] Data Augmentation refers to the transformations as implemented in PyTorch `https://pytorch.org/vision/stable/transforms.html` (last accessed June 16, 2021).

Table 1: Experimental Setup Variational Autoencoder Feature Extraction

	VAE (i)	VAE (ii)	VAE (iii)
input size	64x64	280x280	280x280
latent space size	1 024	72 900	72 900
kernel size	3	3	3
padding	0	1	1
stride	1	2	2
$\mathcal{L}_{reconstruction}$	MSE	BCE	BCE
$\mathcal{L}_{weight}$	10	10 000	1
data augmentation	yes	yes	no

of 16. The output of the *Encoder* is mapped to a normal distribution $N(0, 1)$ with mean 0 and variance 1. We add random gaussian noise to the mapping to ensure similar images being mapped to similar regions. All further specifications for the architecture can be taken from Table 1. We define the loss function to minimize as follows:

$$\mathcal{L}_{VAE} = \mathcal{L}_{\alpha} \cdot \mathcal{L}_{reconstruction} + \mathcal{L}_{Kullback-Leibler} \tag{3}$$

The $\mathcal{L}_{\alpha}$ enables balancing reconstruction and Kullback-Leiber loss. While a larger weight improves reconstruction, a small weight shifts the focus to a normal distributed latent space. The latter fosters both a more symmetric as well as dense latent space. Preliminary experiments exhibited problems with binary cross-entropy for (i) as reconstruction loss. Therefore, we used the mean squared error as $\mathcal{L}_{reconstruction}$ (i), while we applied the binary cross-entropy as $\mathcal{L}_{reconstruction}$ in (ii) and (iii). All experiments were implemented in PyTorch[3] and are using its implemented mean squared error and binary cross-entropy loss.

5.2 Bag of Visual Words (BOVW)

We utilized an implementation by Chenyang Meng.[4] He first extracted for each image keypoint detectors using a Hessian affine region detector [19] and obtained local invariant descriptors by using RootSIFT [2]. Regions and keypoints are represented by feature vectors that he subsequently clusters by k-means into a vocabulary. This vocabulary consists of the visual words combined into a histogram, i.e., the bag of visual words, for each image. We adjusted the number of clusters, which is the vocabulary size, to 1, 500. Meng subsequently employed Redis[5] as a database for querying the bag of visual words. For each input image's bag of visual words, the five most relevant images are returned. According to their scoring, a

[3] `https://pytorch.org/` (last accessed June 16, 2021).
[4] `https://github.com/meng1994412/CBIR` (last accessed June 16, 2021).
[5] `https://redis.io/` (last accessed June 16, 2021).

relevance ranking is determined by comparing the histograms by chi square as distance measure.

5.3 Yolo Ensemble

We used four different YOLO models for our experiments, each trained for a specific type of object: animals, structures, produce, and a model trained on the classes of IconArt as explained in the object detection task. From these models, we then tried to extract an encoding by getting the output of each model's specific layer. Afterward, these vectors can be combined and formed into an embedding. We used cosine similarity to calculate similarity scores to extract similar images. In the following, we tried multiple different layer approaches from where we extract the encodings.

5.4 ResNet Feature Extraction

As one approach, we used a ResNet classifier that was trained for the genre classification task (cf. Chapter Image Classification in Deep Learning for Computer Vision in the Art Domain) and used the average pooling layer to extract embeddings. We used different similarity measures for calculating the scores for each embedding and different sizes of the ResNet model. Additionally, we introduced the possibility of using a color loss. This component enables a model to learn the RGB channels' average values from an image to include these values in the embedding vector for each image.

We did three different experiments with a ResNet50, which is the biggest model we used in our tests. The first experiment was done to see if taking a larger model would result in better performance compared to a Resnet18 and have a baseline for the implementation of the color loss variant of the same model. The following two experiments both tested the impact of the color loss based on a different weighting of 1% and 5%. The goal of these two was, to evaluate if the addition of the color component significantly impacted the result. Furthermore, we needed to see which weighting we would need to choose to put some emphasis on color but not disregard the context that was depicted in the image.

Co-Occurrence Filter. We experimented with different variants of the co-occurrence filter with a ResNet18. In each experiment, we decided to overfit our model on the classification task on purpose. That way the network can memorize all images of the dataset and learn a better representation for them which can be helpful for the present task but is probably not useful in the general case. We experimented with untrained and trained co-occurrence filters and the layer of the network which is used for the feature extraction. Moreover, we explored the influence of data augmentation. Finally, we unified multiple models in an ensemble.

6 Results

In order to be able to compare results between our different approaches we used the normalized discounted cumulative gain (nDCG) [15]. As depicted in Equation 4 the nDCG is calculated by dividing the discounted cumulative gain(DCG) by the ideal DCG (iDCG). The DCG is a score which sums the relevance scores of each entry in a returned list of items, in our case images, and penalizes high relevance images at the end of such a list. This means it would rank methods better if they return all the high relevance images first instead of later on. For our evaluation the DCG had a maximum score of 15 and the nDCG always has values between 0 and 1, in both cases the retrieval results are better the higher the score is. Since we did not have any rankings included in our given data we used an approach to infer the ranking after retrieving the images. First all models were trained and given a specific set of images as the catalogue from which to choose the similar images. Afterwards each model was given the same reference images and returned the five images it would consider as similar to the reference, in descending order. In the following our seminar group looked at all returned images for a given example image and ranked each image, without knowing from which model at which rank it was returned, in one of three categories good match, okayish match and no match. The results can then be used as relevance rating with different scores a good match has a value of three, an okay match has two and no match has a score of one, therefor we can calculate the DCG and also the iDCG. After calculating both we can then use them to calculate the nDCG and rate our models based on these results.

$$nDCG = \frac{DCG}{iDCG} \tag{4}$$

In the following we will present the results obtained from our different approaches and give some example images for each model. The variational autoencoders performed well on images depicting landscapes but only the first experimental setup performed well on other categories of images like genre or portrait. One aspect in that both experiment (i) and (ii) proved to be good at, is distinguishing certain styles of images and matching the colors of a given example. As can be seen in Figure 1 the models can find similar looking images that match the color and the style of the original image. However there are also some errors present, were images are returned even though they have no visible connection to the query image.

Using bag of visual words only provided good result regarding images depicting landscapes. For other images there were some similar images returned but also often images that had no similarity to the input image made their way into the returned results. In addition, when given an image depicting a still life of a flower pot the models had a tendency to return images depicting trees and branch structures. This can also be seen inFigure 2 in result four, the model only managed to retrieve two similar images in result 1 and 3, every other image had no visible connection to the query image.

(a) Query image **(b)** Result 1 **(c)** Result 2

(d) Result 3 **(e)** Result 4 **(f)** Result 5

Figure 1: VAE retrieval results.

The YOLO ensemble had very poor results regardless of the category of a certain image. As we can see in Figure 3 the returned results show no visible similarity to the given query image. The only result were one could argue that there might be some similarity is result one but even that would be extremely lenient. Also, it seems like the model does not value color at all regarding similarity since none of the returned images have a color palette that even vaguely represents the query image given to the model.
The ResNet models that used the cosine similarity as a distance measure for images and ResNet50 as the embedding network had an overall good performance. They performed well on landscape images, correctly identifying the presence of people or structures and returning similar landscape images with those properties. However, we noticed the same tendency to pick images with windmills as the BOVW approach for marine motives. This approach also yielded very good results regarding Still Life images and handles flower bouquets especially well. As shown in Figure 4 the model only returned images that also depict flowers. The only image that is not a good fit thematically is result five, since it is not a still life of a flower pot on a table. It can also be seen that the model put emphasis on similar colors in the images without disregarding the content. Result two is the only black

(a) Query image **(b)** Result 1 **(c)** Result 2

(d) Result 3 **(e)** Result 4 **(f)** Result 5

Figure 2: BOVW retrieval results.

and white image while the remaining images have a somewhat similar color pallet to the query image. Additionally, it only happens very sparsely that images with no similarity to the input image are mixed in with the results. One weakness of this approach was that it did not emphasize the color patterns in the images and would, for example, return a black and white image of a flower bouquet as the best match to a color image of a similar bouquet. This error could be reduced by using the color loss in addition to the normal classifier to include an emphasis on color in the embedding. After some experimentation, a five percent color loss was chosen to match images based on their color and include more images of similar colors in the returned results. Increasing the importance of color too much, results in a drastic loss of quality in the returned images as images of the same color are returned with little to no regard for the image depicted in it.

When using the co-occurrence filter we could observe a clear performance decrease when data augmentation is added during the training time. As it can be seen in Figure 5 the quality of the results takes a heavy hit and there is not a clear connection between the query image and the results anymore. When testing the quality of the results regarding the used layer, we could confirm the initial speculation that the last layer has a bias on the genre and category of the image. That bias is weaker in the second last layer though this did not lead to a visible improve-

(a) Query image

(b) Result 1

(c) Result 2

(d) Result 3

(e) Result 4

(f) Result 5

Figure 3: YOLO ensemble image retrieval.

ment of the results. We compared the performance of an untrained and a trained co-occurrence filter. The performance is slightly better when the filter is trained during the classification training. Finally, we combined different good-performing models in an ensemble. The models vary in the layer used for the feature extraction and some use a non-trained co-occurrence filter. Yet, it did not increase the quality of the results and can lead to inferior results compared to a single last layer with a trained co-occurrence filter. Thus, we do not recommend using an ensemble. In the end the addition of a co-occurrence filter did not improve the retrieval results as we can see if we compare Figure 4 and Figure 6. There is no visible improvement in the retrieved images on the contrary the similarity between the query images and the result seems to worsen after adding a co-occurence filter to the previous ResNet approach.

In the end the ResNet that included a 1% color loss performed the best on 20 selected reference images with a mean nDCG score of 0.55 and an accumulated score of 9.38.

(a) Query image **(b)** Result 1 **(c)** Result 2

(d) Result 3 **(e)** Result 4 **(f)** Result 5

Figure 4: ResNet image retrieval.

7 Discussion

This section discusses our experimental results and reasons about limitations.

7.1 Variational Autoencoder Feature Extraction

Notably, the variational autoencoders of experiments (i) and (ii) deliver mediocre to good performance, while (iii) is convincing only in broad perspectives and otherwise achieves consistently poor results. The superiority of (i) over (ii) can possibly be attributed to the smaller number of dimensions in the latent space. Although the reconstruction of (i) can be considered significantly worse than that of (ii), this does not also lead to worse results in finding similar images. Thus, it can be stated that a smaller number of dimensions may capture more concise features in the latent space. The reconstruction performance itself is not essential for finding good results, rather attention has to be paid to the resulting structuring of the latent space. Thereby, a stronger symmetry or higher density of the latent space, as focused in the experiment (iii), seems to harm the ability to find similar images.

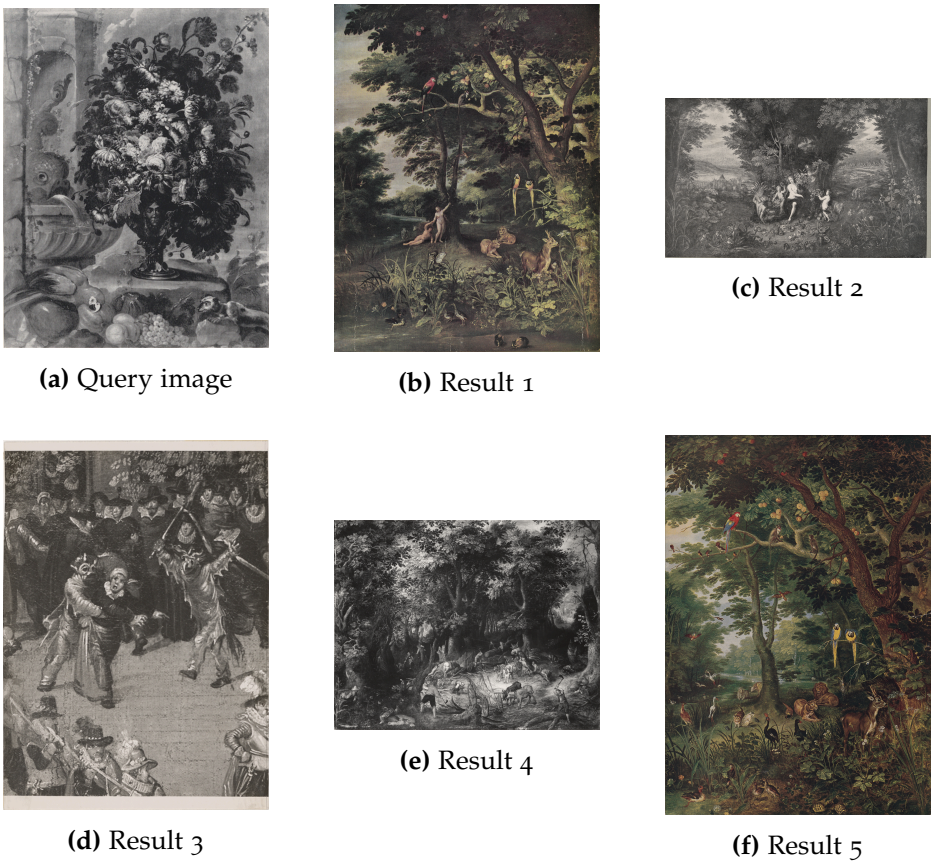
(a) Query image **(b)** Result 1 **(c)** Result 2

(d) Result 3 **(e)** Result 4 **(f)** Result 5

Figure 5: ResNet image retrieval with augmentation.

This also confirms the observation that a good reconstruction of shapes alone, as observed in Figure 7 in (iii), is not crucial for good results. This suggests further focusing on the role of the number of dimensions for a good performance in future experiments. While the stochastic property of variational autoencoders to map randomly to the normal distribution is beneficial to the latent space's similarity, the symmetry and density of the normal distribution itself do not seem to matter or might even have a negative impact. The influence of these properties needs further investigation.

The general strengths of autoencoders (i) and (ii) concerning the style and color of images should be emphasized. This observation is consistent with a more in-depth analysis of latent space that we conducted. Figure 8 shows an input image (still life as a bouquet of flowers) on the left. Starting from the feature maps of the image in the latent space of experiment (ii), we increased all dimensions stepwise and reconstructed them into an image using the decoder. The changes thus give an impression of which features a particular axis represents in the latent space. Figure 8 shows on the left the bouquet as a painting and in dark colors. The further we move along the dimensional axis, the brighter the colors become, and the style itself gradually changes from a painting to a drawing. Based on this observation, a systematic study of latent space and its dimensions is recommended. Possibly, the number of dimensions can still be drastically reduced, on the other hand, further mapped properties can be discovered on other axes, which have not been part of our evaluation so far. In general, these strengths in style and color should be explicitly incorporated into possible future ensemble models. The poor results of



(a) Query image

(b) Result 1

(c) Result 2

(d) Result 3

(e) Result 4

(f) Result 5

Figure 6: ResNet with co-occurence filter.

(iii) are interesting because basic shapes are captured in the reconstruction and thus must also be part of the latent space. In particular, the foreground is captured well, while the background is ignored in a clear separation. This nevertheless does not seem to represent abstract features that sufficiently condition a similarity. We also investigated whether, due to the almost constant number of dimensions compared to the original image in the experiment (ii), the network may have learned the identity function. To do this, we compared the results from (ii) with those that a direct comparison of the images would have obtained. In doing so, we can rule out the possibility that (ii) learned the identity function. Although a comparison among original images obtained good results for broad landscape perspectives (but not the same as in (ii)), the results were not satisfactory for all other comparison images. Therefore, we can conclude that (ii) does not map the identity function.

7.2 Bag of Visual Words

The bag of visual words approach finds acceptable results, but it only finds consistently similar results for landscapes. This might be caused by either a too small or too large vocabulary size. Furthermore, possibly different visual words are grouped

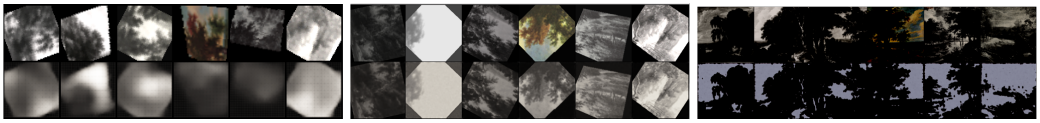


Figure 7: Input and Reconstruction of Images for Experimental Setups VAE (i), VAE (ii), VAE (iii).

Figure 8: Traversal of Latent Space of VAE (ii).

in the wrong clusters. The latter is suggested by observing that bouquets of flowers and tree structures were often classified as similar, or windmills and ships were often grouped. A manual exploratory examination of visual words supports the latter hypothesis. Figure 9 contains three examples of the 1500 visual word clusters. In the cluster on the far left, flowers occasionally appear together with tree structures. The one on the left contains ship masts but no (obvious) windmill blades. However, three-building roofs can be seen. Then, the specific roofs of windmills may contribute to the co-occurrence rather than the windmill-specific blades. On the right, we see an example of a cluster where only tree leaves are included correctly. A systematic analysis of these clusters and how they change with a smaller or larger chosen vocabulary size may help improve the approach. In principle, we consider the clustering step as a possibility for optimization.

7.3 YOLO Ensemble

The YOLO Ensemble did not yield any striking results. This might result from the low accuracy of the object detection models. If objects in specific images are not correctly identified or the detection has low accuracy in general, the resulting embeddings are likely inaccurate. Those inaccurate embeddings subsequently cause poorly captured abstract object features and therefore do not find similar images. However, a likely performance boost from a better object detection model remains an open question. In theory, domain-specific art historic similarities might prevent finding similar images from objects, e.g., humans in an image might not automatically indicate similarity. Practically, the variety and magnitude of depicted objects in images might be unmanageable to cover by object detectors.

7.4 ResNet Feature Extraction

The approach using a ResNet model outperformed every other method. It yielded good results on every category of artwork we tested while being largely independent of the perspective. This could be due to the fact, that the ResNet architecture is

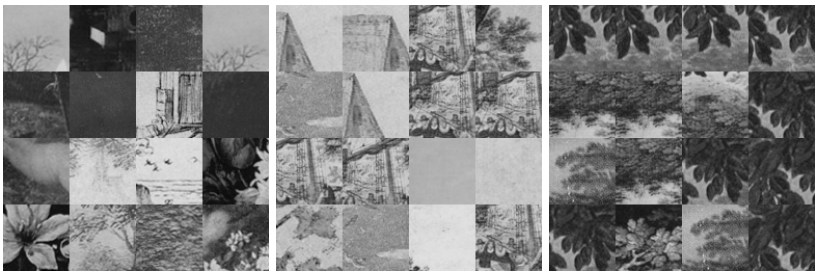


Figure 9: Visual words from the BOVW approach.

well tested and proven to work well for image classification. Therefore, we assume that this method powerfully addresses the semantic gap through an embedding vector containing a well-captured representation of important abstract features. The high abstraction ability of the ResNet experiments is also supported by explorative preliminary investigations in which we trained ResNets only on single art historical categories. These did not yield meaningful results overall, nor on their category, indicating that abstract features could not be learned. Since the models used for image classification show very good performance in classifying the different genres of the artworks, we assume that well-defined features are learned and represented in the embedding and positively impact the retrieval results (cf. Chapter Image Classification in Deep Learning for Computer Vision in the Art Domain). By including the additional color features it is possible to also nudge a model to pay closer attention to colors and therefore representing them in the resulting embedding vector. This approach also improved the result since it put artworks closer together that not only have the same context but a similar color palette as well.

Co-Occurrence Filter. The co-occurrence filter was able to outperform state-of-the-art results for photorealistic image retrieval [8]. Yet, we only managed to achieve mediocre results with the changes we made to the approach. Thus, the reason for the performance difference lies in the differences between ours and the original approach. The most likely causes are the training on a classification task and not a ranking task and the very large image descriptor used for the comparison between images.

8 Conclusion

We can conclude that very similar images for a given query image can be found in our art-historical data corpus. The use of variational autoencoders gives very good results for all categories and is mediocre only for still lifes. A low-dimensional latent space provides the best results. An adequately chosen size of the dimensions of the latent space and necessary analysis of it remains an important task in the future. The bag of visual words approach performs mediocre in all categories, only showing strength in landscapes. A systematic evaluation of the vocabulary

clusters and an optimized clustering as well as the associated correct vocabulary size should be part of future improvements. The use of a YOLO ensemble resulted in very poor results. Since this is likely because the object detection had a low accuracy, future work would include trying to improve the result of the object detection and see if that would lead to better retrieval results as well. We achieved the best results by using a ResNet50 model which was trained to classify the genres of each image as well as put some emphasis on the color depicted in each image. This combination provides very good results regarding the context in each image while also maintaining some form of awareness for images with similar colors. The co-occurrence filter leads to inferior results compared to the results of the average pooling of the ResNet50. This contradicts the research of Forcen et al. [8]. Since we modified the pipeline, further research should be made to reveal what causes that gap. We expect that the co-occurrence approach might outperform the other approaches when the cause of that gap can be removed.

References

[1] J. An and S. Cho. "Variational autoencoder based anomaly detection using reconstruction probability". In: *Special Lecture on IE* 2.1 (2015), pages 1–18.

[2] R. Arandjelović and A. Zisserman. "Three things everyone should know to improve object retrieval". In: *Proceedings of Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pages 2911–2918.

[3] Z. Camlica, H. R. Tizhoosh, and F. Khalvati. "Autoencoding the retrieval relevance of medical images". In: *2015 International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE. 2015, pages 550–555.

[4] G. Castellano and G. Vessio. "Towards a Tool for Visual Link Retrieval and Knowledge Discovery in Painting Datasets". In: *ArXiv* abs/2003.08476 (2020).

[5] M. Du, S. Pentyala, Y. Li, and X. Hu. "Towards Generalizable Deepfake Detection with Locality-Aware AutoEncoder". In: *Proceedings of the 29th International Conference on Information & Knowledge Management*. Virtual Event, Ireland: ACM, 2020, pages 325–334. DOI: 10.1145/3340531.3411892.

[6] S. Dubey. "A Decade Survey of Content Based Image Retrieval using Deep Learning". In: *ArXiv* abs/2012.00641 (2020).

[7] K. P. F.R.S. "LIII. On lines and planes of closest fit to systems of points in space". In: *Philosophical Magazine Series 1* 2 (2010), pages 559–572.

[8] J. I. Forcen, M. Pagola, E. B. Tartas, and H. Bustince. "Co-occurrence of deep convolutional features for image search". In: *Image Vis. Comput.* 97 (2020), page 103909.

[9] K. Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological Cybernetics* 36 (2004), pages 193–202.

[10] N. García, B. Renoust, and Y. Nakashima. "Context-Aware Embeddings for Automatic Art Analysis". In: *Proceedings of the International Conference on Multimedia Retrieval* (2019), pages 25–33.

[11] Glenn Jocher and Alex Stoken and Jirka Borovec and NanoCode012 and ChristopherSTAN and Liu Changyu and Laughing and tkianai and yxNONG and Adam Hogan and lorenzomammana and AlexWang1900 and Ayush Chaurasia and Laurentiu Diaconu and Marc and wanghaoyang0106 and ml5ah and Doug and Durgesh and Francisco Ingham and Frederik and Guilhen and Adrien Colmagro and Hu Ye and Jacobsolawetz and Jake Poznanski and Jiacong Fang and Junghoon Kim and Khiem Doan and Lijun Yu. *ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration*. Version v4.0. Jan. 2021. DOI: 10.5281/zenodo.4418161.

[12] L. Gondara. "Medical image denoising using convolutional denoising autoencoders". In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2016, pages 241–246.

[13] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. "Deep Image Retrieval: Learning Global Representations for Image Search". In: *European Conference on Computer Vision*. 2016, pages 241–257.

[14] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pages 770–778.

[15] K. Järvelin and J. Kekäläinen. "Cumulated Gain-Based Evaluation of IR Techniques". In: *ACM Trans. Inf. Syst.* 20.4 (Oct. 2002), pages 422–446. ISSN: 1046-8188. DOI: 10.1145/582415.582418.

[16] Y. Kang, S. Kim, and S. Choi. "Deep Learning to Hash with Multiple Representations". In: *2012 IEEE 12th International Conference on Data Mining* (2012), pages 930–935.

[17] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[18] A. Krizhevsky and G. E. Hinton. "Using very deep autoencoders for content-based image retrieval." In: *ESANN*. Volume 1. Citeseer. 2011, page 2.

[19] K. Mikolajczyk and C. Schmid. "An affine invariant interest point detector". In: *European conference on computer vision*. Springer. 2002, pages 128–142.

[20] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. "Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pages 5706–5715.

[21] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2014), pages 512–519.

[22] B. Seguin, C. Striolo, I. diLenardo, and F. Kaplan. "Visual Link Retrieval in a Database of Paintings". In: *ECCV Workshops*. 2016.

[23] X. Shen, A. A. Efros, and M. Aubry. "Discovering Visual Patterns in Art Collections With Spatially-Consistent Feature Learning". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pages 9270–9279.

[24] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2015).

[25] Sivic and Zisserman. "Video Google: a text retrieval approach to object matching in videos". In: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003, 1470–1477 vol.2. DOI: 10.1109/ICCV.2003.1238663.

[26] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. "Content-based image retrieval at the end of the early years". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.12 (2000), pages 1349–1380.

[27] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. "Supervised Hashing for Image Retrieval via Image Representation Learning". In: *AAAI*. 2014.

[28] M. D. Zeiler and R. Fergus. "Visualizing and Understanding Convolutional Networks". In: *ECCV*. 2014.

[29] Y. Zhao, B. Deng, J. Huang, H. Lu, and X.-S. Hua. "Stylized Adversarial AutoEncoder for Image Generation". In: *Proceedings of the 25th ACM International Conference on Multimedia*. MM '17. Mountain View, California, USA: Association for Computing Machinery, 2017, pages 244–251. ISBN: 9781450349062. DOI: 10.1145/3123266.3123450.

[30] W. Zhou, H. Li, and Q. Tian. "Recent Advance in Content-based Image Retrieval: A Literature Survey". In: *ArXiv* abs/1706.06064 (2017).

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
138	978-3-86956-513-2	**Proceedings of the HPI research school on service-oriented systems engineering 2020 Fall Retreat**	Christoph Meinel, Jürgen Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Naumann, Holger Giese, Patrick Baudisch, Tobias Friedrich, Erwin Böttinger, Christoph Lippert, Christian Dörr, Anja Lehmann, Bernhard Renard, Tilmann Rabl, Falk Uebernickel, Bert Arnrich, Katharina Hölzle
137	978-3-86956-505-7	**Language and tool support for 3D crochet patterns : virtual crochet with a graph structure**	Klara Seitz, Jens Lincke, Patrick Rein, Robert Hirschfeld
136	978-3-86956-504-0	**An individual-centered approach to visualize people's opinions and demographic information**	Wanda Baltzer, Theresa Hradilak, Lara Pfennigschmidt, Luc Maurice Prestin, Moritz Spranger, Simon Stadlinger, Leo Wendt, Jens Lincke, Patrick Rein, Luke Church, Robert Hirschfeld
135	978-3-86956-503-3	**Fast packrat parsing in a live programming environment : improving left-recursion in parsing expression grammars**	Friedrich Schöne, Patrick Rein, Robert Hirschfeld
134	978-3-86956-502-6	**Interval probabilistic timed graph transformation systems**	Maria Maximova, Sven Schneider, Holger Giese
133	978-3-86956-501-9	**Compositional analysis of probabilistic timed graph transformation systems**	Maria Maximova, Sven Schneider, Holger Giese
132	978-3-86956-482-1	**SandBlocks : Integration visueller und textueller Programmelemente in Live-Programmiersysteme**	Leon Bein, Tom Braun, Björn Daase, Elina Emsbach, Leon Matthes, Maximilian Stiede, Marcel Taeumel, Toni Mattis, Stefan Ramson, Patrick Rein, Robert Hirschfeld, Jens Mönig
131	978-3-86956-481-4	**Was macht das Hasso-Plattner-Institut für Digital Engineering zu einer Besonderheit?**	August-Wilhelm Scheer

ISBN 978-3-86956-514-9
ISSN 1613-5652