Deep convolutional neural network ensembles for art classification

Thijs Bertram
STUDENT NUMBER: u183015

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE OR DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

Prof. dr. E. O. Postma
J. S. Olier

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
May 2019

Preface

This thesis "Deep convolutional neural network ensembles for art classification" has been written to fulfill the graduation requirements of the Communication and Information Sciences program, master track Cognitive Science & Artificial Intelligence, at Tilburg University. I engaged in writing this thesis from February 2019 untill May 2019.

I would like to thank my sister for all the interesting conversations we have about art, creativity and life in general on a daily basis. I am not sure if I would have written my thesis on the subject of artwork analysis if it were not for you. Secondly, I would like to thank my close friends (you know who you are), mother and father for all the support. Prefaces often say something along the lines of: I could not have done it without you. I would say instead, that I probably could have, but you made it so much more enjoyable and bearable. Thank you.

Lastly, I want to thank my supervisor Prof. dr. E.O. Postma for pressing me to keep it simple. I have the tendency to be overly ambitious so I really needed someone to (repeatedly) tell me to 'be practical'.

Oh, and thank god for Google Colaboratory

Table of contents

Deep CNN ensembles for art classification

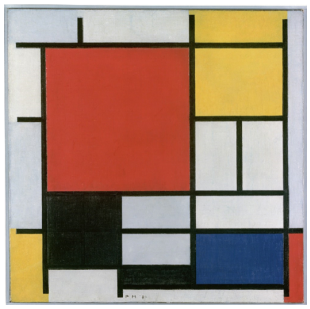
Thijs Bertram

Automatic author attribution of artworks is an emerging sub-field of artificial intelligence and is considered to be one of the harder computer vision task. Large inter- and intra-variation of classes, as well as differences in nature of data, hamper the development of a robust automated classification system. This paper aims to tackle above mentioned issues by taking an ensemble based approach. Ensembles are collections of models that are able to decrease the part of an error that is caused by variance, resulting in greater generalizability. Ten ensemble approaches are compared against a single classifier baseline, which they all manage to outperform. Boosted ensembles perform best, followed by models trained with the DisturbLabel algorithm. The performance of bagged ensembles is somewhat weaker and depends heavily on the aggregation algorithm that is being used: more sophisticated aggregation algorithms achieve better results. Boosted ensembles seem to have the most potential, especially when combined with state-of-the-art CNN architectures. DisturbLabel is a great cost-effective alternative when training time and model flexibility are imperative.

1. Introduction

In recent years, artificial neural networks have gained in popularity immensely. Terms like 'AI' and 'Neural network' are heard all around us and their implementations play a big role in shaping today's society. The sudden rise of this field has been described by some as a mere hype and while there may be some truth to this, it is hard not to get excited by the latest developments. Deep learning specifically (a type of neural network containing a lot of hidden layers) has proven to be a very promising technique. Theory on these types of networks has existed for decades already, but it is the recent surge in computing power and available data that has lead to their dominance as state-of-the-art models. Deep forward neural networks are the go-to architecture for building recommendation systems (Covington, Adams, and Sargin 2016; Deng et al. 2017), deep encoder-decoder (Chiu et al. 2018) and deep LSTM models (Graves, Jaitly, and Mohamed 2013) produce the best results on natural language tasks (Graves, Mohamed, and Hinton 2013), while deep convolutional neural networks reign supreme in the field of computer vision.

One of the more challenging computer vision tasks seems to be the automatic classification of artwork style and authorship (Huang 2018). It has been receiving an increasing amount of academic attention (Puthenputhussery, Liu, and Liu 2016), partly due to the advent of datasets containing high-resolution digital reproductions of artworks (Van Noord, Hendriks, and Postma 2015). A lot of previous studies have produced promising results, but the academic community is far from creating a robust and generalizable industry-grade model. This is a shame, because automatic artwork classification could lead to valuable applications for the art sector such as detection of forgery, artwork recommendation and automated expert-grade artwork analysis.



(a) "Compositie met groot rood vlak, geel, zwart, grijs en blauw" by Mondriaan

(b) "Boerderij met waslijn" by Mondriaan

(c) "Zwaardveger" by Jan Luyken

(d) "Boer" by Caspar Luyken

Figure 1: A representation of inter- (a and b) and intra-variation (c and d) of classes

There are multiple issues that stand in the way of creating such a powerful model. One of which is the large inter- and intra-variations of classes, inherent to the problem of art classification (Jangtjik, Yeh, and Hua 2016). In simpler terms; one artist is not bound to just one style (e.g. the difference between pre- and post-WW1 Mondriaan), while styles of different artists may overlap significantly (e.g. student/teacher relationships), see figure **??** for an example.

Another important factor is the (in)availability of true representative data. No benchmark dataset exists, and new releases of datasets are few and far between (Khan et al. 2014). The datasets that are available differ a lot in terms of: (1) heterogeneity vs. homogeneity of classes, (2) high- vs. low-loss digitization processes and (3) the number of available types of artworks. This high degree of variance in data and classes makes it difficult to build a reliable and generalizable classifier. With this thesis I will aim to tackle the above mentioned issue by exploring the viability of using CNN ensembles for artist classification. Ensemble learning is a well documented machine learning technique that seems to be able to increase overall performance, while reducing a models' variance at the same time. The research question of this paper shall therefore be formulated as:

To what extent do ensemble models influence the performance of automatic artist classification?

2. Related Work

I start off this chapter with an introduction into the history of art analysis, which will help to understand the scientific context and relevance of this paper. This will be followed by a high level description of computer vision algorithms, specifically convolutional neural networks. Lastly, I will present an overview of literature on ensemble models and cover their relevance to the task of art classification.

2.1 Art analysis

Art has been subject to analysis for at least as long as written history exists (Chalmers 2019). Greek philosophers already theorized about the value, origin and meaning of art (Neill and Ridley 1995) and it still remains a popular research topic among philosophers, historians, sociologists and psychologists today. Theories on what it is that makes art so valuable are almost as manifold as there are theorists, but there seems to be some consensus regarding the historical and cultural value of art.

Mattern (1999) believes art is a potent form of communication through which community is developed and political action is taken. Carney (1994) argues that art has aesthetic, functional and historical value. He describes artworks as a frame of their time: giving insight into culture, traditions and societal norms. Wolfe (1969) even believes art to be the creation of forms symbolic of human feeling; assigning a psychological dimension to artworks. It is safe to say that analyzing art can give us insights into all sorts of facets of humanity. This underlines the relevance of being able to correctly identify artworks: without knowing the author of a painting we are unable to derive cultural and historical meaning from it.

Classification of paintings has historically been a task for art historians, experts who classify paintings based on a set of attributes that represent "painting style". Those attributes range from low level stylistic characteristics like lines, brush strokes, color and texture to more high level, abstract concepts like shape, proportion, pattern and subject matter (Fichner-Rathus 2011). Detecting these attributes requires a huge body of knowledge, gained by studying vast amounts of annotated artworks over long periods of time. Because this knowledge is often tacit, it was hard to develop a systematic approach for art analysis and classification (Van Noord, Hendriks, and Postma 2015).

This would change thanks to technological breakthroughs during the 20th and 21st century. The invention of X-rays in 1895 allowed experts to view under-drawings of paintings and would be the first of many techniques that enabled experts to analyze paintings in a more 'objective' manner. Later, infra-red photography, reflectography, multipsectra, fluoroesence and ultra-violet imaging were added to the tool-belt of art historians (Stork 2009). They can be considered as an extension of experts' knowledge; their results still need to be interpreted to reveal any useful meaning. It marked the beginning of experts using technological systems complementary to their tacit knowledge.

Cooperation between experts and technology gained another boost during the last couple of decades. Advances in image processing and computer storage led museums to digitally store huge parts of their collection by scanning and uploading artworks. This was mainly done for preservation, documentation and dissemination purposes, but it opened up doors for digitally analyzing and classifying artworks(Cornelis et al. 2011). Experts took full advantage of this development by constructing all kinds of quantification tools for art analysis, like pigmentation analysis and statistical representation of brushstrokes (Maitre, Schmitt, and Lahanier 2001). These tools can be used for building automated classification systems like the Bag of Words approach (Arora and Elgammal

2012), but the experts themselves are still leading in this process: quantifiable representations of style attributes result from translating their knowledge into the domains of statistics and computer science. The result has been the immersion of a new cross-disciplinary field of engineering, mathematics and culture (Cornelis et al. 2011).

The most recent progress in computer vision has sparked another revolution in the field of artwork analysis that makes the past developments seem trivial. Instead of relying on expert knowledge to create automatic systems, we can now create completely autonomous systems that are able to outperform (systems built by) human experts. The technique that makes this possible is called Deep Learning and allows for an expert-independent classification method where algorithms build an internal representation of style attributes, without any interference of experts. The goal is not to replace art historians, but to enrich their knowledge and teach them about style attributes and features that they may have previously been unaware of.

2.2 Computer vision with Convolutional Neural Networks

Computer vision is a subfield of A.I that aims towards creating systems capable of automating (parts of) the human visual system. Automatic vision systems were originally dependent on hand-crafted features, until Yann LeCun popularized Convolutional Neural Networks (CNNs) in 1989 (LeCun et al. 1989). The architecture of CNNs is based on the visual cortex of humans, where layers of simple and complex neurons are stacked on top of each other (Hubel and Wiesel 1962). These stacks of biological neurons are represented by modules of convolutional and pooling layers.

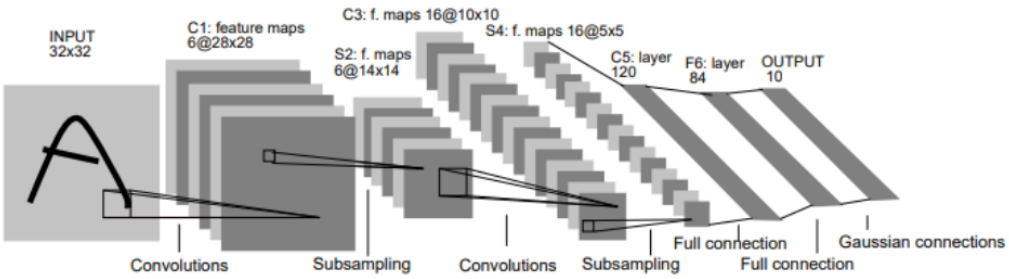


Figure 2: Architecture of LeNet-5 (LeCun et al., 1998), image taken from Rawat and Wang (2017)

A graphical representation of convolution architectures can be seen in figure **??**, where every square represents a feature map and every collection of squares represents a layer. Convolutional layers apply convolutions to an input by 'scanning' the input for certain patterns. These convolved features are than sub-sampled by a pooling layer, basically compressing the information that is outputted by convolutional layers. This compression reduces the spatial resolution of feature maps, achieving the spatial invariance of CNNs (LeCun et al. 1989). Modules of convolutional layers followed by pooling layers are stacked on top of each other, resulting in smaller (read: more abstract) representations as the depth of the network increases. The tail is a collection of dense layers followed by a Gaussian layer. The weights between dense layers represent the relationship between features, whereas the Gaussian layer applies a softmax function to

map these relationships to class probabilities. I wont go into further detail on CNNs so I suggest confused readers to catch up by reading this paper by Zeiler and Fergus (2014).

The strength of CNNs is that they, like regular neural networks, are universal function approximators. They can theoretically compute any function that describes a relationship between input images and output labels. The best part is that feature maps are automatically learned during training. The feature maps in lower convolutional layers of a CNN contain information on low-level features, like lines and shades, while the high-level features like shapes, balance and symmetry are represented in the higher layers. Note that these features align with the style attributes that are used for art analysis as described in section 2.1. The key point is that CNNs can extract features needed for specific tasks, in this case artist classification, and thus learn patters that are representative of (the style of) certain artists.

Scholars have used the above mentioned characteristics of CNNs to build powerful artist and/or style classification systems. David and Netanyahu (2016) achieved state-of-the-art performance on the Webmuseum dataset by using a deep convolutional autoencoder to extract features, which were than fed in to a CNN for supervised artist classification. Van Noord, Hendriks, and Postma (2015) developed a deep CNN called PigeoNET that is able to discover dual authorship. They even used occlusion sensitivity to visualize artist-characteristic regions in the form a heatmap. Van Noord and Postma (2017) also used Gaussian pyramids to create multi-scale representations of artworks. The resulting collection of multi-scale CNNs was combined into an ensemble that is capable of capturing both scale-variant, as well as scale-invariant representations. Their usage of ensembles inspired me to write my thesis on the subject of ensemble learning. The next section will be dedicated to explaining the basic concepts of ensemble learning for those who are unfamiliar with it.

2.3 Ensemble learning

Ensemble learning is a technique that has existed long before the rise of the deep learning paradigm (Dasarathy and Sheela 1979). It has been extensively researched, predominantly in combination with classic machine learning algorithms like decision trees (Kotsiantis 2013). The theory behind it is actually quite simple and is based on the well known notion of 'wisdom of crowds': instead of relying on just one model for prediction, an ensemble of multiple (pre-trained) models is created. The individual outputs of those models are than combined into one final classification by constructing some kind of weighted vote (Dietterich 2000).

The original idea was developed to reduce the variance of classifiers (Dietterich 2000), hoping to achieve a better overall performance. Indeed, lots of prior research has proven that the performance of an ensemble model is often as good as or better than its individual components (Qiu et al. 2014; Sollich and Krogh 1996; Wang et al. 2011). It works because ensembles inherently perform regularization by smoothing in functional space, which helps to avoid over-fitting (Perrone and Cooper 1992). Let us make sure that this idea is clear by putting it into simpler terms. Every classifier is imperfect and has an error with non-zero variability. This error consists of two parts: a bias that tells us something about the accuracy of the model and a variance, that represents the generalizability of the model. It is well known in the field of machine learning that there exists a trade-off between these two types of error: decreasing bias almost always leads to an increase in variance and vice versa (Geman, Bienenstock, and Doursat 1992). Variance can be seen as a result of noise within the dataset, which can be averaged out when it is independent for each sample (Zhang and Ma 2012). This is

precisely what an ensemble does: it reduces variance by averaging it out over multiple models with similar bias. It is essentially a way to combat over-fitting, a phenomenon that has been considered to be a big issue for neural networks.

Neural networks over-fit when they get too attuned to the training data, leading to a drop in performance on other data. It hurts the ability of a model to perform consistently and basically hurts its generalizability (Cogswell et al. 2015). Deep learning models are known to be especially vulnerable to over-fitting when they trained on relatively small datasets (Cogswell et al. 2015), which, like mentioned in the previous chapter, is one of the issues that is standing in the way of creating a robust model for art classification. Academic dedication to 'solving' over-fitting for deep neural networks has led to the development of some successful techniques like L2-regularization (Pereyra et al. 2017), weight decay (Blundell et al. 2015) and drop-out (Srivastava et al. 2014), but research on the usefulness of ensembles is lagging behind. It is only recently that researchers have started implementing ensembles of neural networks on a larger scale. (Zhang and Ma 2012).

For example, Huang et al. (2016) created an ensemble of CNNs that outperforms single models on the CIFAR-10 dataset, Antipov, Berrani, and Dugelay (2016) achieved state of the art performance (97.31%) on facial image gender classification by constructing a simple CNN ensemble and the ensemble constructed by Guo and Gould (2015) outperforms previous methods for object detection on the PASCAL VOC challenge. As far as I am aware, the paper by Van Noord and Postma (2017) is the only implementation of ensembles when it comes to classifying art authorship. This paper will further explore the use of ensembles for art classification, hopefully bridging an academic gap on the viability of ensembles for analyzing artworks.

2.4 Ensemble algorithms

There exist a lot of different procedures for ensemble building, all of which can be categorized into one of the following three dimensions: (1) Data sampling and selection, (2) Training members and (3) Combination of individual classifiers. These dimensions are also called 'the 3 pillars of ensemble systems' since they reflect the different strategies that can be chosen when building ensembles (Zhang and Ma 2012). The following segment entails description of the strategies that will be used in this paper, by clarifying the algorithms and placing them along the three above mentioned axis.

2.4.1 Bagging. Bagging (short for Bootstrap Aggregating) is one of the oldest, yet most powerful ensemble techniques. This algorithm developed by Breiman (Breiman 1996) concerns the first pillar of ensemble building; that of data sampling and selection. Data sampling is a key step in ensemble building because it ensures diversity of individual classifiers. This diversity of models has proven to be of vital importance to the performance of ensembles (Brown 2004). After all, we want the errors of individual models to be as different as possible, otherwise there is little to be gained from combining them.

Bagging creates a bootstrapped subset of data S_t for every classifier by randomly drawing N samples (with replacement) from dataset S with size N. The result is n subsets for n ensembles, where samples can occur multiple times or be absent altogether in every subset S_t. With a large enough dataset S, every subset S_t is expected to contain $1 - 1/\ln (\approx 63.2\%)$ of the unique samples of S (Zhang and Ma 2012).

Classifiers can be trained serially on their respective datasets after subsetting, and will have different weights and errors due to the difference in training data. This technique apparently works best when training on relatively small datasets, especially

for 'unstable' classification algorithms like neural networks and decision trees (Bauer and Kohavi 1999). Following this, I expect the performance of bagged ensembles to be superior to that of individual classifiers.

2.4.2 Boosting. Boosting is another algorithm that concerns the subsampling of training data. The original boosting method was developed by Schapire (1990) for binary classification tasks, but many variations have been developed ever since. AdaBoost.M1 and AdaBoost.M2 (Freund, Schapire et al. 1996) are the most popular algorithms for multiclass classification and, just like bagging, subsample training data to ensure model diversity. The difference with bagging is that boosting algorithms do not sample according to an equal probability distribution. Instead, they 'sample' from a probability distribution D_t by using sample weights, as can be seen in line 2 of algorithm 1. This probability distribution is updated based on the performance of previous models. The sampling probabilities for miss classified examples are increased, so that the next subset $S_t + 1$ contains more samples that are difficult to classify (Zhang and Ma 2012) (see line 7 of algorithm 1). Boosting thus 'boosts' the model by oversampling particularly hard to predict samples.

Algorithm 1 AdaBoost.M1

Input: $Data = \{(x_n, y_n)\}_{n=1}^{N}$, ensemble size T
Initialization: sample distribution D_0: $w_i = \frac{1}{N}$, $i = 1, ..., N$
procedure: Training
1: **for** $t = 1, 2, ..., T$ **do**
2: Train classifier M_t on the weighted dataset D_t
3: Compute error e_t on weighted dataset: $e_t = \sum_{n:y_n \neq f_t(x_n)} w_n$
4: **if** $e_t = 0$ or $e_t > 0.5$ **then**
Terminate model generation
5: compute $\beta_t = \log \frac{1-e_t}{e_t}$
6: **for** $n \leftarrow 1, N$ **do**
7: $w_n \leftarrow w_n e^{II(y_n \neq f_t(x_n)))\alpha_t}$
8: **end for**
9: Renormalize weights so that $\Sigma_n w_n = 1$
10: **end for**

procedure: Prediction of new example x_i
11: **for** $t \leftarrow 1, T$ **do**
12: Predict class with current model $[c, x_i] = f_t(x_i)$
13: $\mu_c \leftarrow \mu_c \beta_t$
14: **end for**
15: The output is the class with maximum probability

This research will compare two boosted ensembles, one trained according to the SAMME algorithm and another one that is trained according to ConfAdaBoost.M1. Both these algorithms are an improvement on the classic AdaBoost.M1 algorithm. SAMME finds a workaround for the condition that can be seen in line 4 of algorithm 1. This condition asserts that the accuracy of individual classifiers is bigger than 0.5. This is fine when tasks have a low number of classes, but becomes very harsh for a task like this, which has as much as 34 possible classes. The SAMME algorithm adds the term $log(K - 1)$ before computing β (line 5 of algorithm 1) and changes the condition in

lines 6 and 7 so that β has to be positive. The effect of this adaption is that individual classifiers only need to be better than random guessing, instead of needing an accuracy of 0.5 or higher.

One of the newer boosting algorithms is called 'ConfAdaboost.M1' (Reiss, Hendeby, and Stricker 2013). This method incorporates, like the name suggests, the confidence of predictions into the process of updating sample weights. The more confident a classifier is, the larger the increase or decrease in weight of that specific sample (see algorithm 2 line 8). Line 14 of the pseudocode for ConfAdaBoost.M1 shows that probabilities are also utilized in the incorporated aggregation stage: confidence and classifier error (based on the weights of incorrectly classified samples) are combined into classifier weights.

Algorithm 2 ConfAdaBoost.M1

Input: $Data = \{(x_n, y_n)\}_{n=1}^{N}$, ensemble size T
Initialization: sample distribution D_0: $w_i = \frac{1}{N}$, $i = 1, ..., N$
procedure: Training
1: **for** $t = 1, 2, ..., T$ **do**
2: Train classifier M_t on the weighted dataset D_t
3: Compute the confidence of the prediction that sample x_n belongs to predicted class: p_{tn}, $n = 1, ..., N$
4: Compute error e_t on weighted dataset: $e_t = \sum_{n:y_n \neq f_t(x_n)} p_{tn} w_n$
5: **if** $e_t = 0$ or $e_t > 0.5$ **then**
Terminate model generation
6: compute $\beta_t = \frac{1}{2} \log \frac{1-e_t}{e_t}$
7: **for** $n \leftarrow 1, N$ **do**
8: $w_n \leftarrow w_n e^{(\frac{1}{2} - II(y_n \neq f_t(x_n)))p_{ti}\alpha_t}$
9: **end for**
10: Renormalize weights so that $\Sigma_n w_n = 1$
11: **end for**

procedure: Prediction of new example x_i
12: **for** $t \leftarrow 1, T$ **do**
13: Predict class with current model $[c, p_t(x_i)] = f_t(x_i)$, where $p_t(x_i)$ is the confidence of the prediction
14: $\mu_c \leftarrow \mu_c + p_t(x_i)\beta_t$
15: **end for**
16: The output is the class with maximum probability

Some academics have been skeptical towards boosting neural networks. Sollich and Krogh (1996) argue that putting emphasize on samples that are hard to classify may lead to overfitting. Difficulty of classifying specific samples could be the result of noisy data, in which case boosting actually encourages fitting noise. In line with this, Maclin and Opitz (1997) found that the risk of overfitting is greater when data is less reliable, so the success of boosting neural networks relies a lot on the dataset. Since the Rijksmuseum Challenge dataset has been carefully annotated by experts, I expect ConfAdaBoost.M1 to outperform the baseline.

2.4.3 Aggregation algorithms. The final step of an ensemble systems is related to the third pillar of ensemble building. It concerns the method of combining outputs from individual classifiers into aggregated predictions. The amount of possible ways to aggregate outputs depends on the type of classifier algorithm that is being used: classifiers that output discrete labels are limited in aggregation options, but neural networks output a continues, class-specific output which opens up a wide range of possible aggregation methods (Zhang and Ma 2012) [1].

Perhaps the most simple and intuitive method is that of **majority voting**. Every individual classifier outputs a single class, which represents that classifier's vote. All votes are added up and the class with the majority of votes will be outputted as the final prediction. Additional requirements like unanimity or plurality of votes can be set, but these will not be covered in this research. I suggest reading 'Neural Network Ensembles' by Hansen and Salamon (1990) for anyone who is interested in the mathematical proof of the majority voting approach.

Classifiers with continuous output can also make use of various algebraic combination methods like mean rule, trimmed mean rule and weighted average rule. **Mean rule** simply averages the class probabilities over all classifiers, which is formally described in formula (1). Here μ_c represents the probability for class c given input x and $d_{t,c}(x)$ represents the probability of classifier t predicting class c given input x. The class with the highest probability after averaging is outputted as the final aggregated prediction.

$$\mu_c(x) = \frac{1}{T} \sum_{t=1}^{T} d_{t,c}(x) \tag{1}$$

The **trimmed mean rule** avoids the problem that occurs when individual classifiers erroneously output unusually high or low support for a particular class. It does so by dropping all decisions of classifiers with the highest and lowest probability before averaging. The downside to this approach is that the ensemble size is essentially decreased by 2; potentially decreasing model variety if the highest and lowest probabilities are not erroneous at all.

Finally, the **average weighted rule** weighs individual predictions before averaging them. These weights are based on an estimated generalization performance of classifiers and can be obtained by testing performance on a validation set during model generation. As can be seen in formula (2), the only difference between this method and the mean rule method is the inclusion of w_t that represents the weight of classifier t. I will compare two different methods of constructing weights, both based on validation set performance. The first method uses the validation accuracy as an absolute weight, whereas the second method constructs a uniform distribution of weights based on a validation accuracy ranking. Creating weights based on a ranking of validation scores could be superior when validation scores do not differ a lot among classifiers (Rokach 2010)

1 Note that this section predominantly applies to bagged ensembles, since most boosting methods use their own, specific way of aggregating

$$\mu_c(x) = \frac{1}{T}\sum_{t=1}^{T} w_t d_{t,c}(x) \tag{2}$$

Note that more complex aggregation methods exist where an extra training step is used to learn the best way of aggregating. Methods like GASEN (Zhou, Wu, and Tang 2002) apply an evolutionary learning algorithm to learn the best weights for the average weighted aggregation rule. These methods will unfortunately be outside of the scope of this thesis, since they require extra training time that was unavailable given the time constraints of this work.

2.4.4 DisturbLabel. A seperate section is needed to cover DisturbLabel, a relatively new algorithm that was developed as a regularization technique by Xie et al. (2016). The key concept is to add noise to the loss (softmax) layer to prevent overfitting. This is done by changing correct training labels into incorrect training labels upon training, or, in other words, disturbing them.

The pseudocode of the Disturblabel algorithm shows how simple this technique actually is. We input a dataset D of size N and a noise rate α. We than loop over all samples x_n of minibatch D_t to choose a random label $\tilde{y}$ with probability α. After disturbing approximately α of the samples the model M is trained on the mini-batch and weights are updated as usual. Note that the disturbed label is not limited to other classes: a label can be disturbed without it being changed. This will happen to a proportion of $\alpha * \frac{1}{34} \approx 0.00029$ for a task with 34 classes.

Algorithm 3 DisturbLabel

Input: $D = \{(x_n, y_n)\}_{n=1}^{N}$, noise rate α
Initialization: a model $M : f(x; \theta_0) \in \mathbb{R}^C$;
1: **for** each mini-batch $D_t = \{(x_m, y_m)\}_{m=1}^{M}$ **do**
2: **for** each sample (x_m, y_m) **do**
3: Generate a disturbed label $\tilde{y}$ with probability α
4: **end for**
5: Update weights of model M
6: **end for**
7: **Output:** trained model M

Xie et al. (2016) explain in their paper that models trained with the DisturbLabel algorithm share similarities with ensembles. Iterations in the DisturbLabel training process are similar to bagging, in the sense that every DisturbLabel iteration can be seen as training on a different noisy dataset $\tilde{D}$. Thus, the authors argue, DisturbLabel can be regarded as a way of combining exponentially many neural networks that were trained on different data, but share the same architecture. They propose a value of 0.1 or 0.2 for the noise rate α which they found to be superior to other values after rigorous testing. According to the authors it should be possible to combine DisturbLabel with other regularization techniques like Dropout. It will be interesting to compare the DisturbLabel algorithm alongside ensembles, because they are built upon the same theoretical grounds.

3. Method

A detailed description of the research methodology is presented to enable successful reproduction of the experiment results. The first part of this chapter describes the dataset and pre-processing procedures, which is followed by a specification of the architecture of individual classifiers. The chapter is concluded with a section on the experimental pipeline and environment.

3.1 Data and pre-processing

This thesis can be considered as an attempt on the Rijksmuseum challenge, introduced by Mensink and Van Gemert (2014). They made a collection of 110,039 digital photographic reproductions of artworks publicly available to be used for participating in four classification challenges; artwork authorship, type, material and creation year. The dataset contains 1,824 different types of artworks from 6,626 unique authors, dating from ancient times to the late 19th century. All artworks have been displayed in the Rijksmuseum of Amsterdam and were carefully digitized under a controlled setting. The dataset is unique in its size, breadth and annotation and can be considered the only dataset that is truly representative of a museum collection.

I followed the approach of Van Noord, Hendriks, and Postma (2015) who subsetted a part of the original dataset for the challenge of artist classification. They divide the original dataset into four different subsets that vary in terms of (1) heterogeneity vs homogeneity of artwork types, (2) number of artists and (3) number of artworks per artist, excluding any artworks for which there is no clear attribution (labeled 'Anonomyous' or 'Unknown'). They motivate their decision by the fact that the original dataset contains lots of artists from whom only a few artworks are available, as well as artists who created lots of different types of artworks. These variations can have a non-trivial influence on model performance and are best controlled for by subsetting the data.

The dataset that is used for this thesis is one of those subsets created by Van Noord, Hendriks, and Postma (2015). It is heterogeneous with regards to artwork type and contains at least 256 works for every of the 34 artists with the largest available pools of work. As can be seen in Table 1, the majority of the work consists of prints. The 'other' category contains a variety of artwork types, including but not limited to 35 paintings and over 1000 pieces of porcelain. Finally, all digital images are reshaped into squares of 256 by 256 pixels.

3.1.1 Pre-processing. I split the dataset into a train, test and validation set according to standard machine learning practice. Comparable to the research by Van Noord, Hendriks, and Postma (2015), the train set contains 17.026 images (70% of total images) while the test and validation set contain 4.850 and 2.436 images respectively (proportion of 20% and 10% of total images). The validation set is solely used for tuning hyperparameters during training, while the test set is kept apart to evaluate model performance.

The next pre-processing step consists of assuring the correct input shape of images. I will be using the VGG19 model (Simonyan and Zisserman 2014) which was originally trained on images of size 224 by 224 pixels. Adhering to this input size has two benefits: it maximizes model performance while, at the same time, acting as a data augmentation technique. Random crops of 224 by 224 pixels are taken by a generator object that also applies horizontal flips with a probability of 0.5. These data augmentation techniques should help to minimize overfitting. Lastly, the resulting images are pre-processed by

Table 1: List of the 34 artists with at least 256 artworks and the distribution of artworks over main types (prints, drawings and other)

#	Name	Prints	Drawings	Other
1	Heinrich Aldegrever	347	27	
2	Ernst Willem Jan Bagelaar	400	27	
3	BoÃńtius Adamsz Bolswert	592		
4	Schelte Adamsz Bolswert	398		
5	Anthonie Van Den Bos	531	3	
6	Nicolaes De Bruyn	515	2	
7	Jacques Callot	1008	4	1
8	Adriaen Collaert	648	1	
9	Albrecht DÃijrer	480	9	2
10	Simon Fokke	1177	90	
11	Jacob Folkema	437	4	3
12	Simon Frisius	396		
13	Cornelis Galle (i)	421		
14	Philips Galle	838		
15	Jacob De Gheyn (ii)	808	75	10
16	Hendrick Goltzius	763	43	4
17	Frans Hogenberg	636		4
18	Romeyn De Hooghe	1109	5	5
19	Jacob Houbraken	1105	42	1
20	Pieter De Jode (ii)	409	1	
21	Jean Lepautre	559		1
22	Caspar Luyken	359	18	
23	Jan Luyken	1895	33	
24	Jacob Ernst Marcus	372	23	2
25	Jacob Matham	546	4	
26	Meissener Porzellan Manufaktur			1003
27	Pieter Nolpe	344	2	
28	Crispijn Van De Passe (i)	841	15	
29	Jan Caspar Philips	401	17	
30	Bernard Picart	1369	132	3
31	Marcantonio Raimondi	448	2	
32	Rembrandt Harmensz. Van Rijn	1236	119	29
33	Johann Sadeler (i)	578	1	
34	Reinier Vinkeles	573	50	

Note: taken from Van Noord, Hendriks, and Postma (2015)

Keras' specific VGG19 'preprocess_input' function. This function subtracts the mean RGB values of the ImageNet training images from the input images (103.939 for red, 116.779 for green and 123.68 for blue values).

3.2 Model architecture

The architecture of individual CNN classifiers is based on the VGG19 model (Simonyan and Zisserman 2014). The VGG19 model contains 19 weight layers and comes with

weights that are pre-trained on the ImageNet dataset. I replace the existing top with a global average pooling layer, a dense layer (1024 nodes), a Dropout layer (0.75) and a Softmax layer (34 nodes). This architecture is inspired by the successful implantation of transfer learning for artist attribution by Teeuwen (Teeuwen 2018)[2]. The top part is trained separately once, after which it functions as a starting point for training individual classifiers. To speed up training, weights up until layer 13 are frozen when training.

3.3 Experimental procedure

All code is written in the programming language Python and can be found on my Github here. The experimental pipeline is split up into two parts. First all the individual CNN classifiers are built and trained using the Keras libarry. This libary runs on the Tensorflow backend and provides an easy, high-level framework for implementing neural architectures. The script for training individual classifiers is executed on the Google Colaboratory platform, an online tool that grants access to a Tesla T4 GPU for training. After a model is finished training its validation accuracy is saved, as well as a numpy array of test set prediction probabilities. In total 10 models are trained according to the Bagging algorithm (section 2.4.1) 4 are trained according to the Disturblabel algorithm (section 2.4.4) and ??? are trained using Boosting algorithms. The hyper-parameters learning rate, momentum, batch size and number of epochs were set to 5 * 10^{-4}, 0.9, 136 and 100 after testing for the best settings on validation data. An early stopping callback with a patience of 10 and best weight restoration was used to ensure optimal weights of classifiers.

The second part of the experimental pipeline consists of aggregating probabilities into an ensemble output. Some custom aggregation functions are created using the matrix multiplication library Numpy. Mean class accuracy is used as an evaluation metric like in most mutliclass classification tasks. It is calculated using the 'classification_error' function of scikit-learn, a scientific computation library. The performance of a single VGG19 classifier will act as baseline for comparison of ensemble architectures. The architecture of the baseline model is the same as that of individual ensemble classifiers and has a MCA of $80,4\%$. Finally, the visualization library Seaborn is used for constructing confusion matrices of the results.

2 The high Dropout rate is a questionable design choice. I followed the approach of Teeuwen (2018), who added a Dropout layer of 0.75. By the time I had realized the rate should be lowered, I had already trained a significant amount of classifiers, so, unfortunately I had to stick to this architecture. If I had the opportunity to train all models again, I would definitely choose a smaller rate

4. Results

The results of all the experiments can be found in table 2. They seem to conform this thesis' hypothesis that ensemble techniques positively influence artist attribution performance: all ensembles perform better than the baseline of $80,4\%$. There does seem to be significant variation between ensembles, so it may be interesting to look at performance of different architectures in more detail before we draw any conclusions.

Table 2: MCA of various models evaluated on the test set

Ensemble technique	Details	Aggregation	MCA
Baseline	1 classifier	N.A	0.804
Bagging	10 classifiers	Majority	0.809
Bagging	10 classifiers	Mean	0.817
Bagging	10 classifiers	Trimmed mean	0.816
Bagging	10 classifiers	Absolute weights	0.818
Bagging	10 classifiers	Ranked weights	0.823
DisturbLabel	$\alpha = 0.1$, Dropout rate $= 0.75$	N.A	0.828
DisturbLabel	$\alpha = 0.2$, Dropout rate $= 0.75$	N.A	0.818
Boosting	10 classifiers, ConfAdaBoost.M1	Confidence $* \beta$	0.847
Boosting	**3 classifiers, SAMME**	$\boldsymbol{\beta}$	**0.849**

4.1 Bagging

Let us start by taking a closer look at bagged ensembles. As we can see in Table 2, their performance ranges from $80,9\%$ up to $82,3\%$. This difference is caused solely by the aggregation algorithms and it appears that, in general, the more sophisticated the algorithm the better its performance.

Ensembles that aggregate using **majority voting** perform worst. The reason for this could be the level of stochasticity that is involved in majority voting. Whenever a vote is tied, the final class is chosen at random. For this experiment, such a tied vote occurs 186 times, accounting for approximately $3,84\%$ of the total test cases. Another experiment is ran to gain some insight into the influence of this stochasticity. Aggregation by means of majority vote has been repeated 1000 times and resulted in a mean accuracy of $80,9\%$, a minimum accuracy of $80,6\%$, a maximum accuracy of $81,3\%$ and a standard deviation of $0,12\%$. So while the stochasticity of majority voting is non-trivial, we can safely say that it does not influence the ranking of bagged majority vote ensembles: even in the best case scenario, it is still the worst performing aggregation method. Another possible explanation for the relatively bad performance of majority voting is the fact that it does not utilize information on class probabilities, while the other methods do.

Mean voting and **trimmed mean** voting methods perform slightly better, achieving MCA scores of $81,7\%$ and $81,6\%$. Performance seems to be negatively influenced by trimming off predictions from the two models with the highest and lowest predictions, suggesting that confident predictions are not likely to be erroneous.

Weighted voting reigns supreme to all other methods, achieving the highest MCA. Weights based on a **ranking** of validation accuracy result in $82,7\%$ MCA, which is slightly better than the MCA of $82,3\%$ resulting from **absolute weights**. A possible reason for this small (but non-negligible) difference may be the fact that ranked weights significantly decrease the influence of weaker models. For both methods, the weights

of individual classifiers are based on their validation accuracy. So, the greater the variation of a weight distribution, the more decisive strong classifiers become. In fact, the variation of ranked weights is nearly 16 times as big as the variation of absolute weights (standard deviation of 0.2872 and 0.0175). At the same time, a stronger emphasis on classifier performance will be beneficial, only when there is a large variation in performance of said classifiers. I decided to conduct another experiment to further examine this relationship between ensemble composition and performance of weighted aggregation methods. This performance is evaluated on 10 ensembles of different sizes, which are created by step-wise removing of the weakest individual classifiers. Thus, the smaller the ensemble size, the smaller the variation in performance of its individual classifiers.

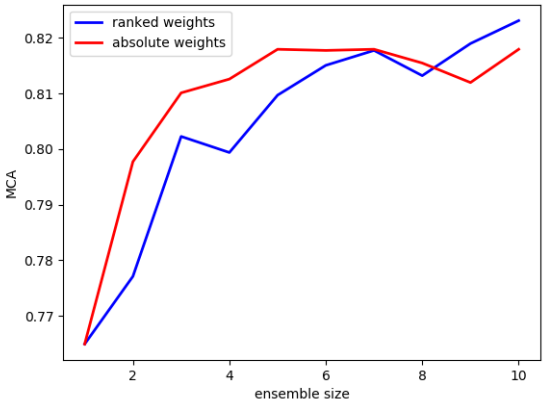


Figure 3: Ranked weights outperform absolute weights *only* when variance in classifier performance is high

Results are presented in figure **??**, which shows us that absolute weights perform better in most cases. When ensemble size (read: variation in performance of individual classifiers) gets bigger, aggregation based on ranked weights actually becomes the superior method [3]. It may be interesting to see how this relationship holds up for ensembles that apply some sort of model selection, like GASEN. Model selection could 'filter out' individual classifiers that negatively impact ensemble predictions, potentially making absolute weights superior to ranked weights in every instance. This point will be further addressed in section 5.

We can conclude that bagged ensembles outperform single classifiers, regardless of their aggregation technique. Performances of aggregation techniques seem to depend on the amount of information they process. Mean voting performs better than majority voting because it incorporates class probabilities, while weighted voting performs better than majority voting because it also adds information on estimated classifier performance. In short, The more sophisticated the aggregation algorithm (read: the more diverse the information on which aggregation is based), the better its performance.

3 Note: the lack of confidence intervals prohibits us from drawing any hard conclusions. Instead, the graph can only give us a rough idea on the relationship between diversity in classifier performance and performance of aggregation techniques

4.2 DisturbLabel

Models trained with the DisturbLabel algorithm have the potential to outperform bagged ensembles. Their success rate seems to depend heavily on the noise rate parameter α; the model trained with $\alpha = 0.1$ performs better than all bagged ensembles (0.828), while the model trained with $\alpha = 0.2$ performs significantly worse. This confirms the findings from Xie et al. (2016) that the value of noise rates should be kept low when DisturbLabel is combined with Dropout layers. They argue that a combination of dropout and high alpha values is viable only if the task is simple enough and/or the data set is very noisy. All in all, DisturbLabel seems to be a powerful regularization algorithm that can provide a computationally cheap alternative to bagged ensembles.

4.3 Boosting

Boosted ensembles outperform all other ensemble architectures. Training according to the ConfAdaBoost.M1 algorithm results in a MCA of 0.847, which is over 4% better than the baseline. The SAMME algorithm performs even better, achieving a MCA of 0.849. To examine whether the performance of boosted ensembles is actually increased due to their superior ability to classify hard examples, we can take a closer look at the confusion matrices. Plots of all confusion matrices can be found in the appendix, but they will not be able to provide us with sufficient information. Instead, precision, recall and F1 scores are computed for two specifically hard to classify classes: Jan Luyken and Casper Luyken. Father and son share a very similar style and are the two classes that our classifiers confuse the most [4]. I will compare the F1, recall and precision scores of the SAMME ensemble to that of the best DisturbLabel model.

Table 3: scores on various performance metrics on the binary classification of Casper Luyken (vs Jan Luyken) for SAMME and DisturbLabel models

Model	Metric	Score
SAMME	Precision	0.854
	Recall	0.594
	F1	0.701
DisturbLabel	Precision	0.596
	Recall	0.736
	F1	0.658

Table 3 presents the results: SAMME boosted ensembles perform better (F1 of 0.701 vs. 0.658) because the score on Precision rises more than the score on Recall drops. DisturbLabel ascribes a lot of works to Jan Luyken that are actually made by Casper. Van Noord, Hendriks, and Postma (2015) also noticed this tendency and argued that this has to do with disproportionate sample sizes. Since the works of Jan and Casper are very similar, classifiers tend to ascribe the work to the artist for which more samples are present. The SAMME ensemble is boosted on these hard to classify samples, so has less of a problem with differentiating between Caspar and Jan Luyken. This is indicated by

4 The confusion between these two classes is probably emphasized by the high Dropout rate of individual classifiers. See section 3.2 for more info

the difference in precision: the SAMME ensemble is better at correctly ascribing a work to Caspar, which implies that it is less likely to simply predict the majority class when faced with hard to classify samples. The theoretical arguments against boosting neural networks (see section2.4.2) do not seem to hold for this experiment, implying that the Rijksmuseum Challenge dataset labels are not very noisy.

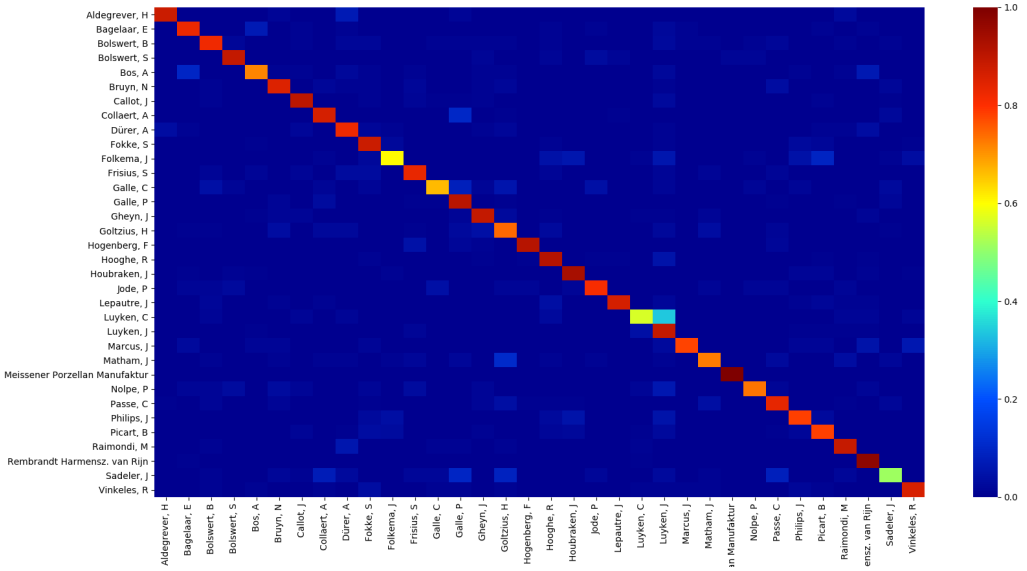


Figure 4: Plotted confusion matrix of ensemble trained with the SAMME method

5. Discussion and future work

In this work I explored the viability of using CNN ensembles for automatic artist classification. The idea behind the use of ensembles is that they reduce variance, leading to better performance on unseen data. As I mention in the introduction, variance caused by imperfect data is one of the main hurdles that needs to be overcome if academics ever want to create a robust artist classification system. I therefore conducted several experiments, evaluating the performance of 6 bagged and 2 boosted ensembles, as well as 2 models trained with the DisturbLabel algorithm.

The results seem to confirm the theoretical case for the usage of CNN ensembles. Every ensemble architecture manages to outperform the baseline, so using ensembles instead of individual classifiers positively impacts artist classification performance. The next section will consider these results from a more practical point of view, addressing the implications of choosing ensemble approaches. I will also touch upon some imperfections of the experimental design, which will lead to recommendations for future work. Van Noord, Hendriks, and Postma (2015) already attend to some limitations of the dataset, mainly regarding differences in digitization processes, as well as the nature of its class distribution. I consider these limitations of the Rijksmuseum Challenge dataset to be known, so I will focus on limitations that are specific to this study instead.

The most obvious downside to ensemble learning is the increased time it takes to train and predict. All individual classifiers need to be trained, so the relationship between training time and ensemble size is of linear nature. It is questionable whether ensemble approaches are feasible in the real world. Picture a scenario in which a model needs to be extended to a bigger number of classes. Re-training all the individual classifiers would be a necessary evil, taking up considerable time and resources. The increased prediction time could prove to be impractical as well, especially when applications demand near-instantaneous classifications (think of a mobile art analysis application).

Some of the major limitations of this study are a direct result of the time complexity involved in training ensembles. I had to freeze, for example, half of the VGG19 layers. This restricts the model to use its ImageNet weights for low-level features, limiting the extent to which the VGG model is being transferred (in the context of transfer learning) to the domain of art classification. I also applied some kind of two-phase learning to cut back on convergence time. The top layers were trained once at the beginning of the experiment, after which their weights were used as a starting point for training all other classifiers. By taking this approach I basically chose to sacrifice ensemble diversity for the sake of speeding up convergence. Lastly, as can be seen in table 2, the SAMME ensemble contains less classifiers compared to other ensembles. Boosted ensembles take especially long to convergence due to the shrinking of sample weights in later iterations. The remaining SAMME classifiers are still being trained as I am writing this section. Their results will be added in the future. Comparing ensembles of different sizes is not completely useless though; SAMME ensembles of smaller sizes already outperform bagged ensembles and DisturbLabel models, which, if anything, illustrates the superiority of this technique.

5.1 Future work

The absence of error decomposition analysis can be seen as another major limitation. In section 2.3 I argue that ensembles should theoretically reduce the part of the error that is caused by variance. The results seem to indicate that this is indeed the case,

but a quantitative method that actually tests this assumption regarding bias-variance decomposition would have been a valuable addition to this study. Perhaps scholars could examine this bias-variance decomposition for CNN ensembles in future work. Another possible avenue of research could be the implementation and evaluation of other, more modern ensemble algorithms. There are still tons of boosting algorithms (ABCBoost, AdaBoost.HM, AdaBoost.BCH, etcetera) that have the ability to outperform ConfAdaBoost.M1 and SAMME. At least as promising, is the implementation of model selection algorithms like GASEN. In their famous paper, Zhou, Wu, and Tang (2002) prove that selecting a subset of classifiers is almost always superior to using all classifiers. Since time constraints left me no other choice but to use all classifiers, any form of model selection in future work would most likely lead to improved ensemble performance.

A final recommendation for future work concerns the architecture of CNN classifiers. The VGG19 network, which stems from the 2014 ImageNet competition, is relatively outdated and could be replaced by more sophisticated networks. Zeren (2019) compares the performance of several state-of-the-art convolutional networks for artist classification on the exact same dataset that is used in this research. Her findings are promosing: the Xception network manages to get a MCA of 89.9. A combination of the Xception network and ensemble based methods (be it boosted ensembles or a DisturbLabel implementation) would probably achieve unprecedented MCA scores on this dataset.

6. Conclusion

This thesis explored the viability of using ensemble approaches for automatic artist attribution. All nine ensembles outperform the single classifier baseline, so it is safe to say that ensemble approaches have the potential to contribute towards creating a robust automated artist classification system.

Bagged ensembles perform only marginally better than the baseline, with their performance depending heavily on the way that individual classifiers are aggregated. In general, the more sophisticated the way of aggregating, the better the ensemble's performance. Boosted ensembles that are trained with the SAMME algorithm perform best, achieving a MCA score that is 4.5 higher than the baseline (84.9 vs. $80,4$). The reason behind the increased performance of boosted methods is their increased ability to correctly predict the author of hard to classify samples.

Unfortunately, there is no such thing as a free lunch: ensembles come with their own set of limitations. Training individual classifiers takes a lot of time, so expanding models to greater sets of classes can be problematic. Increased prediction time also hampers the use of ensemble approaches for real-time applications that require near instantaneous results. Above mentioned problems can be circumvented with the help of the DisturbLabel algorithm; a technique that applies regularization to the loss layer. This algorithm acts somewhat like a bagged ensemble, achieving even better results (0.828 MCA) while taking only a fraction of the time to train.

Concluding, ensemble techniques positively influence performance on an automatic artist attribution task. Boosting approaches result in the highest performance increase, and therefore have the highest potential to contribute to a robust automatic artist classification system. The DisturbLabel algorithm can be seen as a resource efficient alternative, which I would suggest if model flexibility or resource availability are imperative.

References

Antipov, Grigory, Sid-Ahmed Berrani, and Jean-Luc Dugelay. 2016. Minimalistic cnn-based ensemble model for gender prediction from face images. *Pattern recognition letters*, 70:59–65.

Arora, Ravneet Singh and Ahmed Elgammal. 2012. Towards automated classification of fine-art painting style: A comparative study. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3541–3544, IEEE.

Bauer, Eric and Ron Kohavi. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1-2):105–139.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Breiman, Leo. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Brown, Gavin. 2004. *Diversity in neural network ensembles*. Ph.D. thesis, Citeseer.

Carney, James D. 1994. A historical theory of art criticism. *Journal of Aesthetic Education*, 28(1):13–29.

Chalmers, F Graeme. 2019. The study of art in a cultural context. In *Art, Culture, and Pedagogy*. Brill Sense, pages 95–105.

Chiu, Chung-Cheng, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778, IEEE.

Cogswell, Michael, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. 2015. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*.

Cornelis, Bruno, Ann Dooms, Jan Cornelis, Frederik Leen, and Peter Schelkens. 2011. Digital painting analysis, at the cross section of engineering, mathematics and culture. In *2011 19th European Signal Processing Conference*, pages 1254–1258, IEEE.

Covington, Paul, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, ACM.

Dasarathy, B.v. and B.v. Sheela. 1979. A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE*, 67(5):708âĂŞ713.

David, Omid E and Nathan S Netanyahu. 2016. Deeppainter: Painter classification using deep convolutional autoencoders. In *International conference on artificial neural networks*, pages 20–28, Springer.

Deng, Shuiguang, Longtao Huang, Guandong Xu, Xindong Wu, and Zhaohui Wu. 2017. On deep learning for trust-aware recommendations in social networks. *IEEE transactions on neural networks and learning systems*, 28(5):1164–1177.

Dietterich, Thomas G. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15, Springer.

Fichner-Rathus, Lois. 2011. *Foundations of Art and Design: An Enhanced Media Edition*. Cengage Learning.

Freund, Yoav, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156, Citeseer.

Geman, Stuart, Elie Bienenstock, and René Doursat. 1992. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

Graves, Alex, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278, IEEE.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649, IEEE.

Guo, Jian and Stephen Gould. 2015. Deep cnn ensemble with data augmentation for object detection. *arXiv preprint arXiv:1506.07224*.

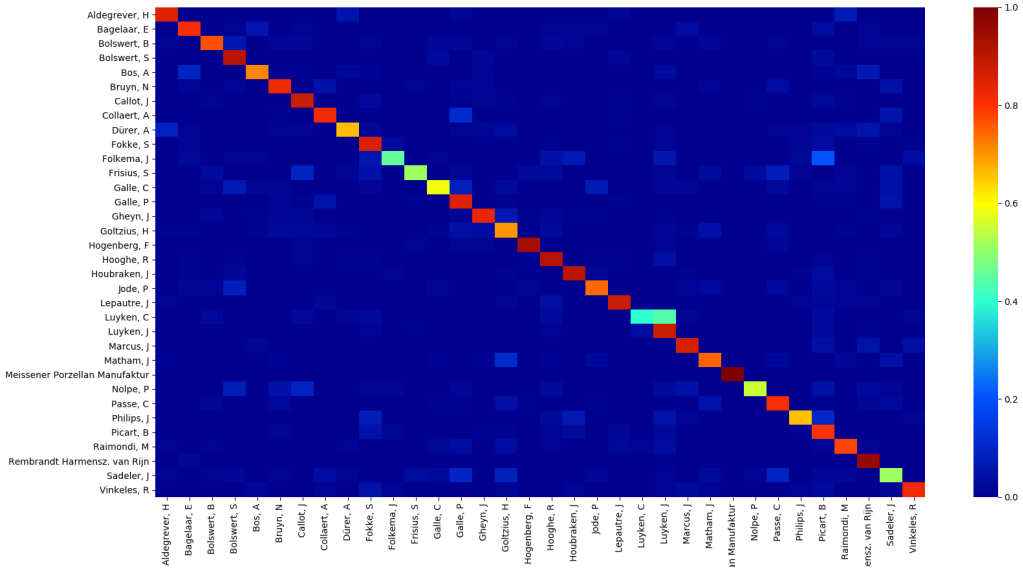
Hansen, Lars Kai and Peter Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):993–1001.

Huang, Hsin-Kai, Chien-Fang Chiu, Chien-Hao Kuo, Yu-Chi Wu, Narisa NY Chu, and Pao-Chi Chang. 2016. Mixture of deep cnn-based ensemble model for image retrieval. In *2016 IEEE 5th Global Conference on Consumer Electronics*, pages 1–2, IEEE.

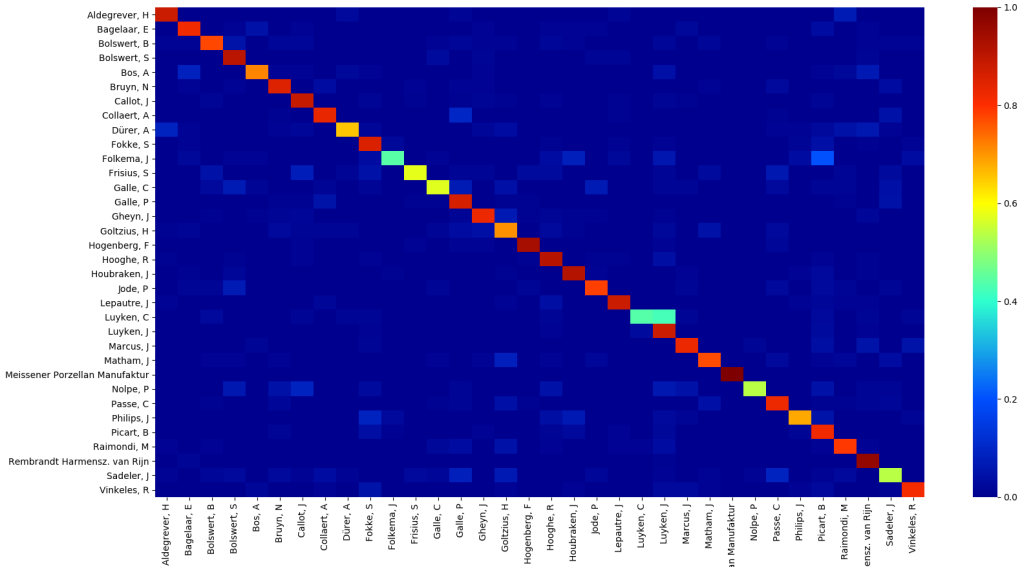
Huang, Jing. 2018. *Fine-grained artworks classification*. Ph.D. thesis.
Hubel, David H and Torsten N Wiesel. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.
Jangtjik, Kevin Alfianto, Mei-Chen Yeh, and Kai-Lung Hua. 2016. Artist-based classification via deep learning with multi-scale weighted pooling. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 635–639, ACM.
Khan, Fahad Shahbaz, Shida Beigpour, Joost Van de Weijer, and Michael Felsberg. 2014. Painting-91: a large scale database for computational painting categorization. *Machine vision and applications*, 25(6):1385–1397.
Kotsiantis, Sotiris B. 2013. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283.
LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.
Maclin, Richard and David Opitz. 1997. An empirical evaluation of bagging and boosting. *AAAI/IAAI*, 1997:546–551.
Maitre, Henri, Francis Schmitt, and Christian Lahanier. 2001. 15 years of image processing and the fine arts. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 1, pages 557–561, IEEE.
Mattern, Mark. 1999. John dewey, art and public life. *The Journal of Politics*, 61(1):54–75.
Mensink, Thomas and Jan Van Gemert. 2014. The rijksmuseum challenge: Museum-centered visual recognition. In *Proceedings of International Conference on Multimedia Retrieval*, page 451, ACM.
Neill, Alex and Aaron Ridley. 1995. *The philosophy of art: readings ancient and modern*. McGraw-Hill.
Pereyra, Gabriel, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.
Perrone, Michael P and Leon N Cooper. 1992. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, BROWN UNIV PROVIDENCE RI INST FOR BRAIN AND NEURAL SYSTEMS.
Puthenputhussery, Ajit, Qingfeng Liu, and Chengjun Liu. 2016. Color multi-fusion fisher vector feature for fine art painting categorization and influence analysis. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9, IEEE.
Qiu, Xueheng, Le Zhang, Ye Ren, Ponnuthurai N Suganthan, and Gehan Amaratunga. 2014. Ensemble deep learning for regression and time series forecasting. In *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pages 1–6, IEEE.
Rawat, Waseem and Zenghui Wang. 2017. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449.
Reiss, Attila, Gustaf Hendeby, and Didier Stricker. 2013. A competitive approach for human activity recognition on smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2013), 24-26 April, Bruges, Belgium*, pages 455–460, ESANN.
Rokach, Lior. 2010. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.
Schapire, Robert E. 1990. The strength of weak learnability. *Machine learning*, 5(2):197–227.
Simonyan, Karen and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
Sollich, Peter and Anders Krogh. 1996. Learning with ensembles: How overfitting can be useful. In *Advances in neural information processing systems*, pages 190–196.
Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
Stork, David G. 2009. Computer vision and computer graphics analysis of paintings and drawings: An introduction to the literature. In *International Conference on Computer Analysis of Images and Patterns*, pages 9–24, Springer.
Teeuwen, Margot. 2018. Learning to automatically recognize artists by their artworks. tilburg university.
Van Noord, Nanne, Ella Hendriks, and Eric Postma. 2015. Toward discovery of the artist's style: Learning to recognize artists by their artworks. *IEEE Signal Processing Magazine*, 32(4):46–54.

Van Noord, Nanne and Eric Postma. 2017. Learning scale-variant and scale-invariant features for deep image classification. *Pattern Recognition*, 61:583–592.
Wang, Gang, Jinxing Hao, Jian Ma, and Hongbing Jiang. 2011. A comparative assessment of ensemble learning for credit scoring. *Expert systems with applications*, 38(1):223–230.
Wolfe, Alvin W. 1969. Social structural bases of art. *Current Anthropology*, 10(1):3–44.
Xie, Lingxi, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. 2016. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762.
Zeiler, Matthew D and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833, Springer.
Zeren, Veda. 2019. Automatic artist attribution task with current convolutional neural networks.
Zhang, Cha and Yunqian Ma. 2012. *Ensemble machine learning: methods and applications*. Springer.
Zhou, Zhi-Hua, Jianxin Wu, and Wei Tang. 2002. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263.

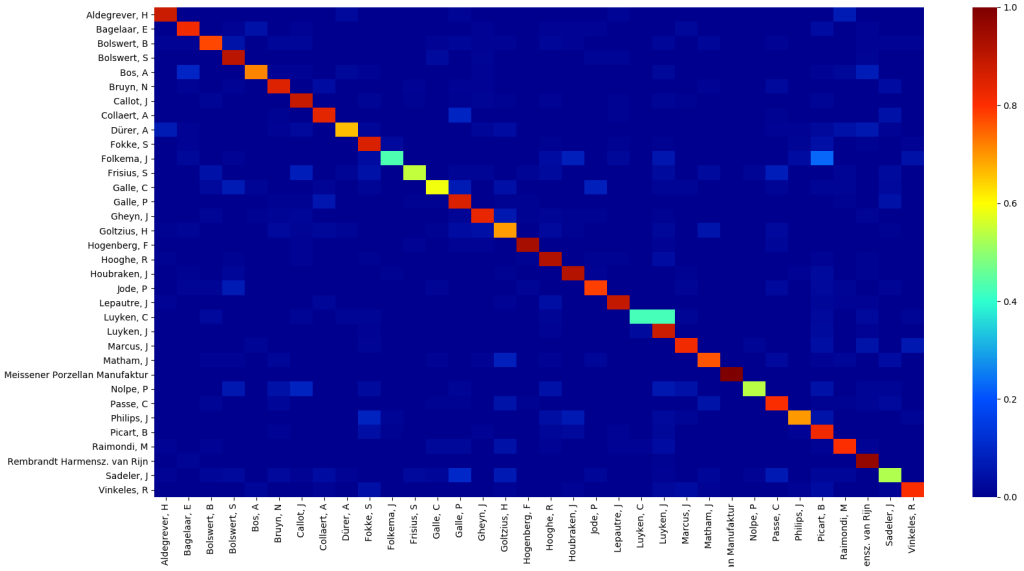
Appendix A: confusion matrix Bagged majority vote



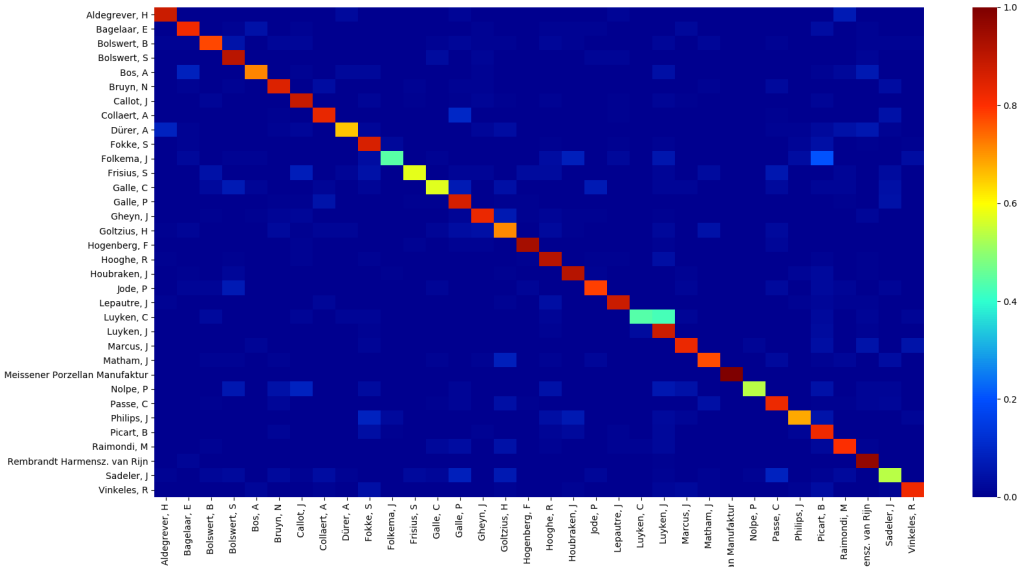
Appendix B: confusion matrix Bagged mean vote



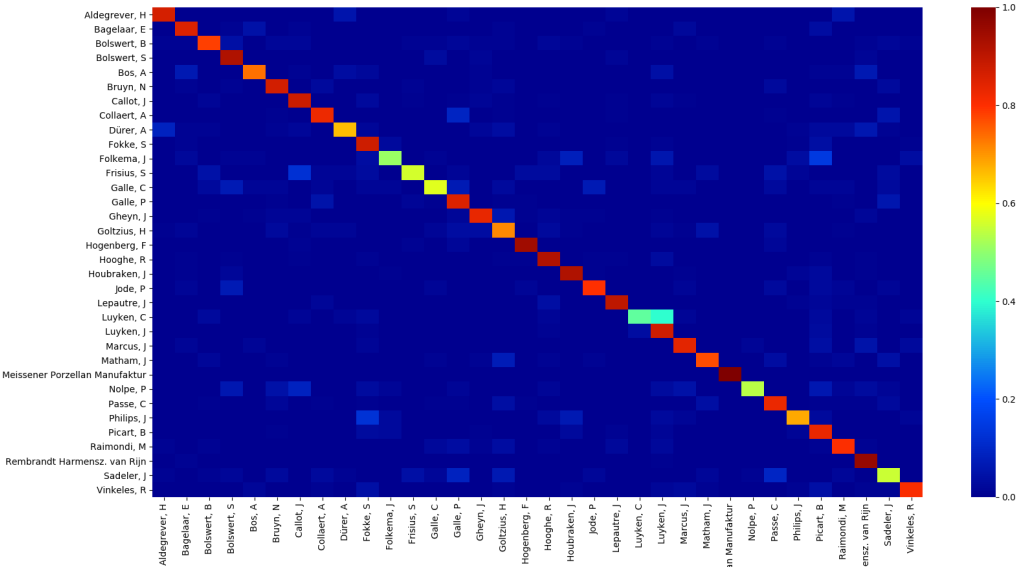
Appendix C: confusion matrix Bagged trimmed mean vote



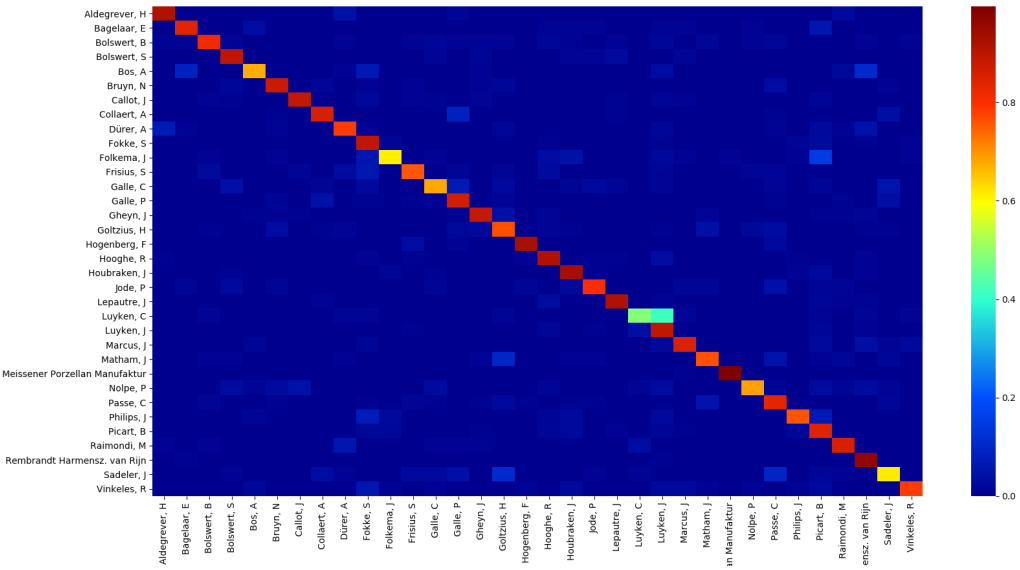
Appendix D: confusion matrix Bagged weighted vote (absolute)



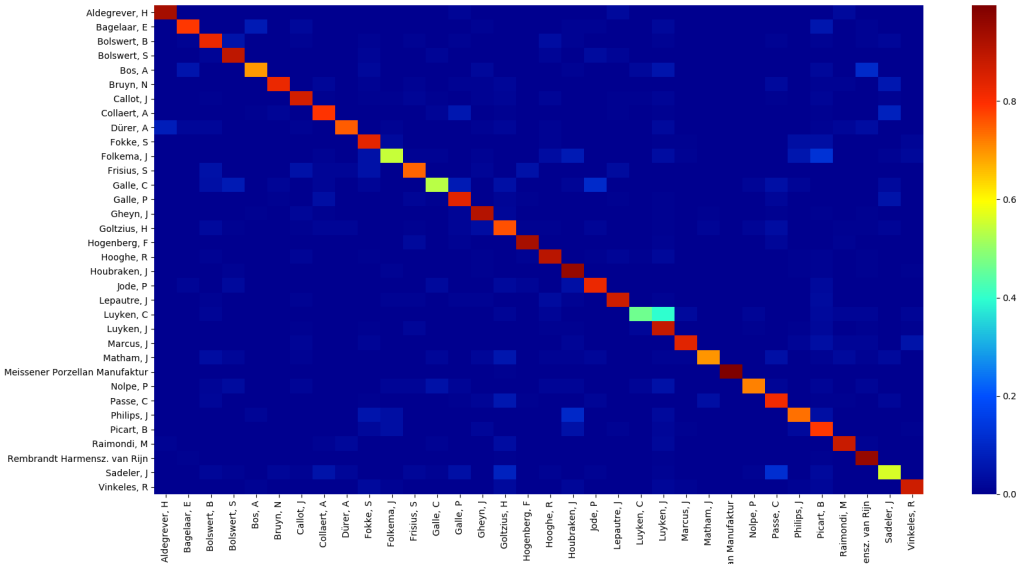
Appendix E: confusion matrix Bagged weighted vote (ranked)



Appendix F: confusion matrix ConfAdaBoost.M1



Appendix G: confusion matrix DisturbLabel ($\alpha = 0.1$, Dropout $= 0.75$)



Appendix H: confusion matrix DisturbLabel ($\alpha = 0.2$, Dropout $= 0.75$)