Approaches for the Improvement of the Multilabel Multiclass Classification with a huge Number of Classes

Martha Tatusch
Institute of Computer Science
Heinrich Heine University Düsseldorf
D-40225 Düsseldorf, Germany
tatusch@cs.uni-duesseldorf.de

ABSTRACT

In the field of data analysis, the multilabel multiclass classification is still a major problem in case of a large number of classes.

With the help of deep learning methods, impressive information can be extracted from a wide variety of data. For example, people can be recognized on images and in videos or fonts can be imitated. Nevertheless, these algorithms also encounter limitations. One of these limits when classifying objects is the treatment of multiple classes. For example, if an image is supposed to be described with the help of a dictionary in a few keywords, there are countless words that can be selected, but only very few that apply to the object. Another aggravating fact is that the number of words per image is not fixed.

This paper presents two basic approaches to improve the classification accuracy with neural networks compared to a common approach. One strategy describes a parallel model that requires clustered label sets. For this purpose, different distributions are considered. In the second approach, the effects of different loss functions are investigated.

It is shown that the presented approaches obtain a very significant improvement of the results compared to the basic model. Both approaches show an improvement of at least 400%. The parallel architecture even achieves 31 times better results than the basic model. We also show under which conditions the individual approaches can achieve the most effective enhancement of quality.

Categories and Subject Descriptors

I.2.8 [**Artifical Intelligence**]: Problem Solving, Control Methods, and Search; H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.4.m [**IMAGE PROCESSING AND COMPUTER VISION**]: Miscellaneous—*Image Classification*

Keywords

Neural Networks, Image Processing, Artificial Intelligence, Classification, Information Retrieval

30^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 22.05.2018 - 25.05.2018, Wuppertal, Germany.

1. INTRODUCTION

Today, Deep Learning and Artificial Neural Network are widespreaded terms especially in the fields of information technology and data science. A few years ago, these methods were launched and immediately met with great enthusiasm. They stand for a specific concept of machine learning (ML), in which a machine can learn by itself and opens up new knowledge only on the basis of training data that does not necessarily have to be preprocessed. This discovery was a major breakthrough in the field of data science because there finally was a way to avoid the difficulties of the feature selection that would otherwise be required in ML.

Although there already was a wide variety of classifiers[10] that could learn from training data, the developer always had to manually explore which features of the objects were meaningful and extract them beforehand, so that the human being still had a great influence on the quality of the results.

In Deep Learning, the relevant features are automatically determined and processed. The used construct is an artificial neural network with multiple layers between the input and the output. With these networks it is possible to find correlations between data that cannot be readily grasped by the human mind. In addition, problems that seem simple for humans but are difficult on a programmatic level, such as the artificial generation of realistic images or fonts, and the generation of meaningful answers to freely formulated questions, can be solved.

But these models also have their weaknesses. When classifying objects, large amounts of training data are required so that each class – also called label – can be learned with a moderate number of representatives. If we now want to label a collection of different images – for example, patient images of a hospital – thousands of different words are possible. The number of possible words can of course be limited, for example, by choosing a subject area, but the number of possibilities will still be large. This means that the number of images per class on average is very low. The use of classifiers that require a previous feature selection is not possible, since there are no recognizable consistent properties of relevance that can be extracted. This only leaves the possibility of using deep neural networks. Due to the large number of classes, however, this task also represents a great challenge in Deep Learning, which is dealt with in this paper.

2. APPROACHES

Multilabel multiclass classification describes the task of classifying data into classes, whereby there are a lot of classes and each data point can be assigned to any number of

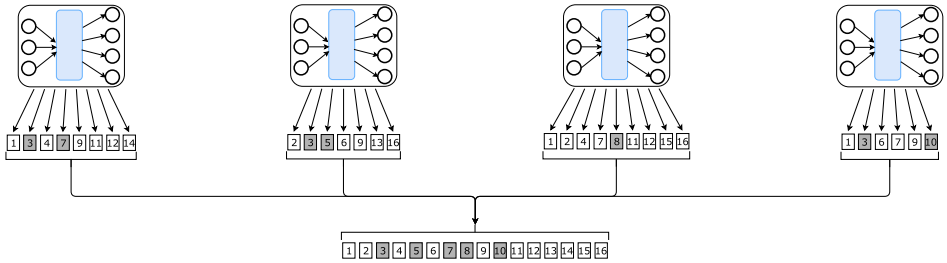


Figure 1: Example of the parallel execution of multiple CNNs on clustered label sets. The final result is calculated by combining the individual results using the indicator method. The rectangles marked in gray represent classes into which the sample object has been classified.

classes. This work deals with the situation in which each input object is assigned to only a fraction of the possible classes. Let $|C|$ be the number of possible classes and $|C_o|$ the number of classes assigned to an object o. A particular difficulty in dealing with this problem using machine learning algorithms is that because of the ratio $\frac{|C_o|}{|C|}$, which is a very small value, the system has difficulties to learn sensibly.

For example, it is possible that the network may adapt itself to assign objects to no class at all. This can be explained by the fact that the hit rate is mainly influenced by the classes which are not assigned to the considered object. The Accuracy Metric is calculated by the formula

$$\frac{number\ of\ correctly\ classified\ classes}{number\ of\ all\ classes}.$$

During the learning process, many systems strive to maximize this value. Suppose there are 1 000 classes and one object is assigned to exactly 5 of these classes. If a classifier does not classify the object to any class, it will have an accuracy of $\frac{1000-5}{1000} = \frac{995}{1000} = 99.5\%$. There is a high probability that this value will deteriorate if the system tries to find the correct classes, which can cause incorrect assignments. Since the accuracy of non-assignment is still very close to 100%, this method proves to be the best option for the network. But for humans, however, this approach does not make sense. The aim is, of course, to make the assignment as accurate as possible, but to assure that an assignment will be made. This means, in this situation, it is much more important to identify the associated classes than to prevent other classes from being incorrectly assigned to an object.

2.1 Parallel Network Architecture

Since the high number of classes is the greatest difficulty, it is very likely that splitting the problem into several smaller sub-problems can improve the results. The division into several easier problems can be achieved by clustering[10] the set of labels. If then, for each cluster, one seperate net is trained, the number of classes gets considerably lower and the ratio $\frac{|C_{avg}|}{|C|}$ increases. The denominator $|C_{avg}| = \frac{1}{N}\sum_{o=0}^{N}|C_o|$ stands for the average number of labels per training object and the counter $|C|$ for the number of all possible classes. Now, the question is how to divide the classes in order to achieve the best possible results.

Since often nothing is known about the labels other than their names, properties must be determined with which the clustering can be performed. One possibility is to look at the independent occurrences of the different classes. Labels that are assigned to a similar number of data objects would then be placed in the same cluster. It is unlikely that only labels with similar occurrences will be assigned to the same object. This means that often multiple clusters contain labels that belong to one data object. This increases the probability of correct assignment. Another possibility is to divide the labels randomly into several groups.

In contrast to classification, which falls under the term *supervised learning*, clustering belongs to the *unsupervised learning*. This means that objects are classified without knowing the classes in advance. Therefore there is no training data, since no information about class affiliation is known. In this work, the clustering by occurrences is done with the KMeans algorithm.

The parameter K represents the number of clusters to be calculated. First, K random data points of the training set are selected as the centers of the individual clusters. These are called *centroids*. All objects are then assigned to the cluster whose centroid is closest to the object. The distance is usually calculated with the euclidean distance. Now the centers are recalculated by computing the mean value of all data points of the respective cluster. All data points are then reassigned and the resulting centroids are calculated. This is repeated until the assignment of the data objects does not show any changes anymore.

With the help of the determined clusters, the classification problem can now be broken down. We consider a construction, in which a CNN is trained separately for each calculated label set. When the model is executed, all CNNs are evaluated in parallel. Here, parallelism does not mean the temporal context, but a symbolism for the fact that all CNNs are used for testing at the same level. All resulting assignments are finally merged and contribute with the same importance to the final result.

Figure 1 illustrates an example of a model that can classify into 16 classes. The individual clusters contain different numbers of labels and some overlaps. In the leftmost cluster, for example, a network is used that can categorize into the classes $1, 3, 4, 7, 9, 11, 12$ and 14. In this example, it has chosen the labels 3 and 7. In the final result, the outcomes of all CNNs are considered equally.

There are several ways to combine the results. For example, all classes selected by at least one CNN can be considered assigned in the final result. It is also possible to use a majority voting system or an average value. In the first case, this means that for each label all results of the clusters containing it are considered. Only if the majority has assigned

the object to this class, it is also selected in the final result. For the calculation of the average values either the binary or unrounded predicted values can be used. The consideration of the decimal values is more suitable, since a prediction of 0.51 for a class in the binary case would already result in a 1, which would flow into the average much more strongly than a 0.51. Finally the average value itself is rounded up, whereby in the binary case a "double rounding" would result, which can falsify the result.

2.2 Propensity Loss Function

A further approach to improving the results of a multilabel multiclass CNN, which has nothing to do with the construction of the model and the clustering of classes, is to adjust the loss function. If it is set up in such a way that, for example, false negatives are strongly and false positives are hardly penalized, then this would already have a great influence on the learning process of the classifier and would prevent objects from not being classified at all.

The learning process of a convolutional neural network requires a loss and an optimization function. Depending on the resulting error value of a run, all weights of the CNN are adjusted during the backpropagation. A frequently used loss function in multilabel classification is the binary cross entropy. It is calculated by $H(p,q) = -\sum_x p(x)\log(q(x))$, where $p(x)$ stands for the actual probability and $q(x)$ for the calculated probability that the considered object belongs to class x. The resulting probabilities are rounded, so that $p(x)$ and $q(x)$ can only have values of 0 or 1. The largest costs are incurred if the network does not classify into the class which the object in question belongs to. If it assigns the data point to a class which it does not belong to, no costs are caused by the object.

In [5] a new type of loss functions is introduced. It is primarily designed for multiclass classification with an enormous number of classes. According to [5], the functions prioritize the assignment to the correct classes and promotes classification to rarely occurring labels. Their special characteristic is the relation to the propensity of the individual labels.

The Hamming Loss is cited as a bad example for a loss function for the multiclass problem. For a model M, it is defined by

$$HL(M) = \frac{1}{N \cdot L} \sum_{i=1}^{N} \sum_{j=1}^{L} (y_{i,j} - \hat{y}_{i,j})^2 \ , \qquad (1)$$

with N the number of data points, L the number of labels, $y_{i,j}$ the actual assignment of an object i to class j, and $\hat{y}_{i,j}$ the predicted assignment of an object i to class j. Because of the squared difference, the model is punished for both false negatives and false positives. In addition, the costs for all individual class assignments are calculated in the same way, as it is usual in most cases. In an unbalanced dataset, however, there may be labels that contain very few representatives but are nevertheless as important as frequently represented labels. These are easily overlooked during the training because the probability of an incorrect assignment is significantly lower than for classes that belong to many data points. Even a correct classification to such minority classes has not much influence on the training result, as this happens so rarely that the relevant weights get hardly changed.

For this reason, a cost function which treats each label individually and calculates weighted costs is desirable. In [5] a type of loss functions is presented, which is based on propensity values that can be calculated with subjective relevance ratings. The developer can assign relevance values to the individual classes, which then are incorporated into the cost function. Since this paper assumes that the relevance ratings of the different labels are not known, another variant is used that was presented in the same paper and is independent of subjective evaluations.

Based on previous observations, the authors have decided that the propensity of a class can be represented by a sigmoid function. For a label l with unknown relevance value the propensity p_l is calculated by

$$p_l = \frac{1}{1 + (log(N-1)) \cdot \sqrt{1.4} \cdot e^{-0.5\ log(N_l + 0.4)}} \ , \qquad (2)$$

where N_l represents the number of data objects that contain the label l and N stands for the number of all training objects. The values for the optimization parameters are the same as those chosen in the paper.

In [5] the integration of propensity scores into different known loss functions was presented. In this work, the decision was made to use an adapted version of the Hamming Loss function:

$$HL(M) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} (\frac{1}{p_j}(2\hat{y}_{ij} - 1)) \cdot (y_{i,j} - \hat{y}_{i,j})^2 \ . \qquad (3)$$

The subterm $(2\hat{y}_{ij} - 1)$ has the function of an indicator that checks whether the object i has been assigned to class j or not. In the binary case, it is 1 if it has been classfied, and -1 if it has not been classified into the observed class. As a result, predictions in which i incorrectly has been assigned *to* class j are punished and those who have wrongly *not* been assigned an object to the class are rewarded. By positioning the propensity in the denominator of the fraction, misclassifications to labels with high propensity are weighted less than those to the rarely occurring ones. Except for this factor, nothing has changed in the original Hamming Loss function.

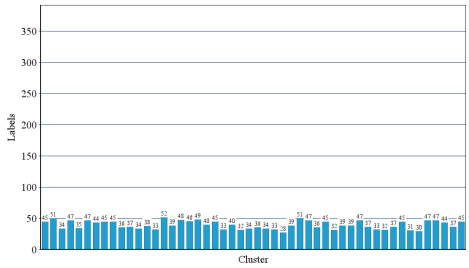
As one of the problems discussed here is that the classifier possibly learns not to classify *at all*, it is not advisable to take the formula from [5] unchanged. The indicator function only punishes false positives and even rewards false negatives. As a result, the likelihood that the network does not make a classification at all rather than misclassifying an object is increased. In this work, it makes more sense to use a loss function that punishes false positives and false negatives equally or possibly prefers false negatives. In any case, however, incorrect allocations must increase the error value. For this reason, the absolute value of the indicator function is used in the following process:

$$HL(M) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} (\frac{1}{p_j}(|2\hat{y}_{ij} - 1|)) \cdot (y_{i,j} - \hat{y}_{i,j})^2 \ . \qquad (4)$$

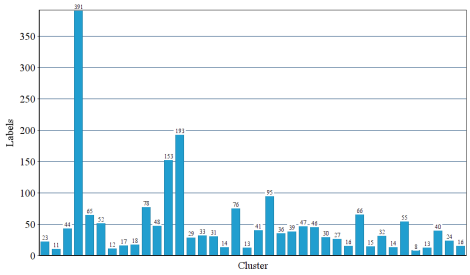
This ensures that all incorrect classifications are treated in the same way.

3. REALISATION

Before creating the model, the input data has to be preprocessed. All images are mapped to the RGB color space. Since the net expects a fixed image size, a squared size of 800×800 pixels has been chosen. If neccessary, the increase of the image size is achieved by adding black borders. This can be



(a) Random



(b) By Occurrences

Figure 2: Distributions of the labels with different clusterings. A dataset of 2 000 labels has been used.

done by adding zero values on the sides. If the image is too big, it is scaled down by means of interpolation until the largest side length is 800 pixels long. The smaller side length is then evenly filled with zero values from both sides.

3.1 Clustering

In order to accomplish the approach of the parallel model, the first step is to cluster the label set. In this work, in any case, 50 clusters has been requested. The determination of random label groups is self-explanatory. The result is a distribution of the classes that is similar to an uniform distribution. This balanced arrangement is illustrated in Figure 2a. The smallest cluster contains 28 and the largest 52 labels. All label groups are therefore relatively small.

Although the used MiniBatchKMeans implementation of Scikit Image[1] receives a desired number of clusters as parameter, it only creates as many clusters as actually make sense. The effect of this is that during clustering by occurrences 39 label groups with 8 to 391 classes are created. Besides a few exceptions, these are again relatively small clusters. Even the largest number of labels is more than four times smaller than the total amount of classes and therefore represents a significant decimation of the label set. Nevertheless, the distribution is very heterogeneous. The differences between the label distributions of the random clusters and the clustering by occurrences become clear in Figure 2, as the Y-axis is same-scaled in both cases.

Both, the random distribution and the clustering by occurrences, generate disjoint label groups. Since it is interesting to see which effect it would have, if the clusters showed overlaps, an additional distribution of the labels has been made. The labels were randomly distributed into 50 clusters, with each label being assigned to a maximum of 5 clusters.

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html`

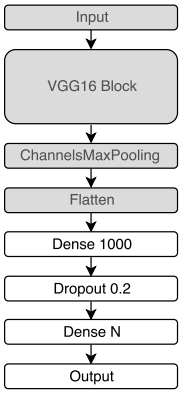


Figure 3: Uniformly used CNN Architectur. The grayed-out part was computed only once.

3.2 Architecture

Due to the problem of determining a suitable Convolutional Neural Network architecture and the usually time-consuming training sessions of a network, it is advisable to use a pre-trained network, which has already achieved convincing results on similar data. In [9] several strongly resemble architectures for Deep Convolutional Neural Networks are presented. They were developed as part of the ImageNet Challenge 2014[8]. The VGG16 net achieved the best results with a depth of 16 trainable layers.

Since both the input data and the required output differ from the original architecture, the model needs to be slightly modified. The entire chosen section of the architecture includes $14,714,688$ pretrained weights. These can be set untrainable so that only the weights which have been added by the own layers are trained. In Figure 3 the final architecture used in all cases is displayed. Since the weights of the VGG16 block were no longer trained, the output of this part of the net could be calculated once and reused to increase efficiency. Since the result of this area still contained 512 channels, the idea arose to pool the result. When using the VGG16 block without pooling, it was noticeable that the output sometimes contained a lot of zeros. For this reason, pooling above the maximum makes sense to reduce the number of zeros. Since it is not the size of the feature maps but the number of channels that should be reduced, we wrote a custom layer. It is named ChannelsMaxPooling Layer and pools one-dimensionally each pixel over a given number of feature maps. It can be found on Github[2]. In this work a filter size of 32 and a step size of 8 pixels were used. This means that the sliding window goes over 32 channels and is moved in 8-pixel steps across all channels. The number of feature maps is reduced from 512 to $\frac{512-32}{8} + 1 = 61$.

The white components in Figure 3 must be trained for each approach. The second dense layer generates a N-dimensional vector, where N stands for the number of CNN classes considered. Unlike all other layers, it does not have ReLu as an activation function, but Sigmoid. By using this activation function, all values of the resulting vector are normalized to the interval $[0, 1]$, which correspond to the probabilities of an assignment to the respective class. Between the two dense layers a dropout layer is applied, which randomly rejects 20% of the tensor values to prevent overfitting.

[2] `http://github.com/tatusch/ChannelsMaxPoolingLayer`

Dataset	Clustering	Loss Function	Precision	Recall	F1-Score
2000-Labels Dataset	None	Binary Cross Entropy	0.000206	0.000900	0.000336
		Propensity Loss	0.001118	**0.461154**	**0.002230**
	Random	Binary Cross Entropy	0.005398	0.282772	**0.010594**
		Propensity Loss	0.002225	0.455054	0.004429
	Random (redundant)	Binary Cross Entropy	0.002082	0.225877	0.004126
		Propensity Loss	0.001842	0.431557	0.003669
	By Occurrences	Binary Cross Entropy	0.000709	0.066093	0.001403
		Propensity Loss	0.000475	0.188481	0.000948
1000-Labels Dataset	None	Binary Cross Entropy	0.002608	0.001677	0.002042
		Propensity Loss	0.005148	0.201733	**0.010040**
	Random	Binary Cross Entropy	0.013830	0.151201	**0.025343**
		Propensity Loss	0.007573	0.209422	0.014618

Table 1: Comparison of the achieved precision, recall and F1 score values with different clusterings and loss functions on the two datasets. The results of the redundant clusters with the Binary Crossentropy were calculated using the indicator and with the Propensity Loss using the average method.

4. EXPERIMENTAL RESULTS

The dataset used for the evaluation was provided during ImageCLEF2017[6]. All images come from the medical field, but can vary greatly in size, color coding and content. For example, there are images of wounds, patients, CT scans and maps of areas where a diseases has been spread. The meanings of the labels can be looked up in the Unified Medical Language System (UMLS)[3]. They, however, were not taken into account in this work.

The dataset contains 164 614 training and 10 000 test images in different formats and a total of 20 812 labels. The test results of the competition listed in [3] clearly show that a precise classification of the objects is a difficult problem.The best achieved average F1 score is only 15.83%. The next 10 places are occupied with values of 12 to 14%. With additional external resources, a maximum F1 score of 17.18% was achieved. All these values are far away from a score, which can be taken for a *precise* classification.

Due to limited technical resources, it was necessary to reduce the total amount of labels to 2 000 labels, which were chosen randomly. Although it is an enormous decimation of the number of classes, the dataset is still large enough to represent the described problem. The decimation of the label set causes, that the reduced dataset contains in total 89 113 training images and 5 451 test images.

Since in this selected dataset a lot of labels with very few representatives were included, another subset of the dataset was chosen for comparison. This time the 1 000 most common labels were selected. All these labels are represented by at least 139 training images. The resulting data set includes 150 339 training and 9 201 test images.

In Table 1 the results of the models with different clusterings and loss functions on the two data sets are displayed. All metrics are calculated by the formulas used in [1]. It becomes clear that all results are anything but good. The basic network with all labels reaches only an F1 score of 0.000336 on the 2000-Labels Dataset and 0.002042 on the 1000-Labels Dataset. In relative terms, however, a strong improvement was achieved by the different approaches. Regarding the F1 score – which describes the most meaningful measure referred to the task – the best results have been obtained with the parallel architecture, the random disjoint clusters and the binary cross entropy. The resulting F1 score is nearly eight times larger than that of the clusters by occurrences. Random redundant clustering achieves the second-best values and is still much better than the worst clustering. Nevertheless, the results are considerably worse than with the other alternative. This fact once again confirms the assumption that a network achieves poorer results the more classes it has to consider.

In order to determine which method is the most suitable, the individual merge strategies have been evaluated on the 2000-Labels Dataset. The results are shown in Table 2. As you can see, for the binary cross entropy the best strategy is given by the indicator method. This outcome was expected as the assignment rate with this loss function is very low and gets supported by the indicator function.

Since the propensity loss already promotes the allocation rate itself, the best results are achieved with the average method. This result can also easily be explained. Since the loss function increases the number of assignments, the false positive rate is increased, as well. To reduce this effect, the most appropriate results must be selected. Since both the average and majority method provide this functionality, both are suitable for the propensity loss. Table 2 shows that both results are very close to each other. In Table 1 the results of the most suitable strategies are displayed.

The usage of the propensity loss function improves the results of the basic network by around 400% on both datasets,

Method	Binary Cross Entropy		Propensity Loss	
	R	F1	R	F1
Indicator	0.225877	**0.004126**	0.642536	0.002345
Average	0.053195	0.003831	0.431557	**0.003669**
Majority	0.056994	0.003796	0.433857	0.003649

Table 2: Comparison of the achieved results with redundant clusters using different merge strategies on the 2000-Labels Dataset.

[3] https://www.nlm.nih.gov/research/umls/

as can be seen in Table 1. Particularly noticeable is the improvement of the recall. On the 2000-Labels Dataset a recall of 0.4612 is achieved. This can be explained by the promotion of classification which causes the decrease of the false negatives rate. In combination with the parallel architecture, however, a significant improvement of the results regarding the basic model is achieved, as well, but it gets clear that in all cases the parallel approach scores better without the propensity loss function.

Furthermore, it can be said, that on both datasets, the results of the basic model using the propensity loss are outperformed by the parallel approach with random clusters (disjoint as well as redundant).

5. RELATED WORK

To the present day, there are a few approaches for multilabel and multiclass classification[7][2][12][13][4], but only few publications work on the combination of the two tasks. In addition, the difficult facts that the number of labels per object is not fixed and the number of classes is very high are usually not taken into account. In [11] a classifier is presented that deals with the multilabel multiclass classification and uses association rules[10] to classify the input objects. The authors achieve very good accuracy results, however, the solution is not suitable for a classification with a large number of classes.

6. CONCLUSION

It has been shown that the division of the original label set into smaller label groups and the parallel execution of multiple CNNs on seperate clusters significantly improves the results regarding a basic network which deals with all labels at a time. Depending on the choice of the clustering, the quality can be even further improved. On the used datasets, the use of clusters after occurrences has not been very advantageous. Significantly better results were achieved with random disjoint clusters. The artificially generated redundancy of the labelsets reduces the quality of the results, as this again leads to an increase in the average cluster size.

The customized propensity loss function reveals a strong improvement of the results, as well. The usage of this function makes particularly sense if the assignment rate is very low. If the structure of the model has already achieved a relatively high assignment rate, it has been shown that the propensity loss can also *reduce* the quality of the results. For this reason, it should not be assumed that the propensity loss always leads to an improvement in quality.

7. FUTURE WORK

Unfortunately the usage of association rules for the clustering of the labels was not possible on the datasets considered here. Due to the diversity of the data, it was not possible to find suitable parameters that covered most of the labels and did *not* produce any rules that only appeared once or twice. Rules that occur so rarely are meaningless and therefore not useful. For future research, it would be interesting to look at a different set of data and examine the effects of such a clustering, especially since in [11] it was shown that association rules can achieve convincing results.

Another interesting aspect would be the development of a hierarchical model, in whose uppermost levels it is decided which cluster the input object can be assigned to. A classification to the concrete labels is only made at the lowest level. It can also be promising to perform backpropagation across all levels so that the first levels can learn from the final results.

Principally, the multilabel multiclass classification with a large number of classes is continually a difficult problem that can be investigated extensively in the future.

8. REFERENCES

[1] F. Chollet. Metrics File in Keras' GitHub Repository. `https://github.com/keras-team/keras/blob/ac1a09c787b3968b277e577a3709cd3b6c931aa5/keras/metrics.py`. Accessed: 2018-04-06.

[2] O. Dekel and O. Shamir. Multiclass-Multilabel Classification with More Classes than Examples. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[3] C. Eickhoff, I. Schwall, A. G. S. de Herrera, and H. Müller. Overview of ImageCLEFcaption 2017 - Image Caption Prediction and Concept Detection for Biomedical Images. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum*, 2017.

[4] D. J. Hsu, S. M. Kakade, J. Langford, and T. Zhang. Multi-Label Prediction via Compressed Sensing. In *Advances in Neural Information Processing Systems 22*. 2009.

[5] H. Jain, Y. Prabhu, and M. Varma. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[6] H. Mller, P. Clough, T. Deselaers, and B. Caputo. *ImageCLEF: Experimental Evaluation in Visual Information Retrieval*. 2010.

[7] M.-E. Nilsback and A. Zisserman. Automated Flower Classification over a Large Number of Classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.

[9] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computing Reasearch Repository*.

[10] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining: Pearson New International Edition*. 2013.

[11] F. A. Thabtah, P. I. Cowling, and Y. Peng. MMAC: A New Multi-Class, Multi-Label Associative Classification Approach. In *ICDM*, 2004.

[12] C. M. Wang, L. and J. Feng. Parallel and Sequential Support Vector Machines for Multi-Label Classification. In *International Journal of Information Technology*, 2005.

[13] T. Zhang. Class-size Independent Generalization Analsysis of Some Discriminative Multi-Category Classification. In *Advances in Neural Information Processing Systems 17*, 2004.