Exceptional model mining of convolutional neural networks

van Strien, B.

Award date:
2019

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Data Mining Research Group

Exceptional Model Mining of Convolutional Neural Networks

Master Thesis

Brent van Strien

Supervisor:
W. Duivesteijn

Eindhoven, 2019-03-20

Abstract

Subgroup discovery is a technique to find subsets of a dataset which deviate from the entire dataset with regards to a target variable. Exceptional Model Mining (EMM) has been introduced to employ subgroup discovery for various models with multiple target variables. Recently, neural networks are becoming more popular due to increased computational speed. We introduce an EMM model class that discovers exceptional subgroups in training datasets of convolutional neural networks. For this model class, we propose two distinct quality measures which are used to measure the exceptionality of a subgroup. The first quality measure finds subgroups of training data which learn a different representation by comparing the trained weights of convolutional layers. The second quality measure finds subgroups of training data which have exceptionally many low or high features by checking their convolutional layer outputs. Our experimental results on the MNIST and CIFAR-100 dataset trained for image classification show that the model class finds subgroups of these natures.

Contents

Chapter 1

Introduction

Subgroup discovery aims to find subsets of a dataset for which a chosen target variable has deviating values in comparison with the entire dataset. Such a subset is named a subgroup and is usually obtained by a guard on one or more descriptive variable values. Whereas subgroup discovery is limited to a simple, singular target variable, exceptional model mining (EMM) [9][21] allows for more complex target concepts. These are called model classes in the context of EMM and they allow more than one target. The purpose of EMM is to find the most exceptional subgroups in a dataset with regards to the model class. A quality measure is used to quantify the exceptionality of the model class of a subgroup. It takes the target values of the subgroup and those of either the whole dataset or the complement of the subgroup. The quality measure is then a function which outputs a value when these targets values are given. Numerous model classes have been researched already, including correlation, association, regression, classification [9], Bayesian networks [10] and rank correlation [8].

Problem statement Neural networks are often seen as a black box, an output is obtained by giving an input, and their inner workings are abstracted from. It is hard to gain insight into the precise way it obtained that output due to their enormous number of parameters. Data mining is used to gain insight into datasets, but methods specifically researched to gain insight into the datasets that train neural networks are not explored yet.

The purpose of this thesis is to research and propose such a method through EMM. Therefore, the research question of this thesis is: How can we employ exceptional model mining on training datasets of convolutional neural networks? To this end, a new EMM model class is researched based on neural networks. Due to the broad range and purpose of neural networks, the research in this thesis and the EMM model class is limited to convolutional neural networks.

The training dataset does not include target variables necessary for EMM. Instead, the convolutional neural network should provide these target variables. Thus a first subquestion that needs to be answered is: What suitable target variables do we gather from convolutional neural networks?

After we have decided upon the target variables, the model class needs a quality measure over these target variables. This quality measure determines the exceptionality of a subgroup within the model class. Thus the second subquestion this thesis tries to answer is: Given the target variables of convolutional neural networks, what quality measure can we use to find interesting exceptional subgroups of the training dataset?

Approach We can use neural networks that are trained with either the training dataset or a subgroup thereof to define the quality measure. We propose two approaches to solve the problem. The first approach uses the weights of trained convolutional neural networks as target variables and compare them to obtain a distance used for the quality measure. The second approach uses the outputs of the neurons on convolutional layers as target variables and count exceptionally low or high outputs as the quality measure.

Outline The structure of this thesis is as follows. First related work is described in Chapter 2. In Chapter 3, we give a rundown of the preliminaries which are exceptional model mining and neural networks. Then in Chapter 4, we describe our method which introduces a convolutional neural network model class for exceptional model mining, separated in two approaches. Next we report on numerous experimental EMM tasks with the model class in Chapter 5. A discussion is held in Chapter 6 where we compare our approaches. Finally, in Chapter 7, we conclude this research and give suggestions for future research.

Chapter 2

Related work

The domain of data mining, otherwise called knowledge discovery in databases, includes many different problems and solutions. In this chapter, a number of problems and solutions related to this thesis are discussed.

2.1 Local Pattern Mining

Data mining can be separated into two categories; global models and local patterns. A local pattern [14] involves only a small part of all the data in a database and is an anomaly that deviates from the statistical model of the entire database. It is these patterns that local pattern mining is meant to discover. Within the field of local pattern mining many different techniques have been introduced. These techniques include EMM, which is the local pattern mining technique our research is based on. In the following subsections, some of the most groundbreaking or otherwise related local pattern mining techniques are discussed.

2.1.1 Frequent Itemset Mining

The goal of frequent itemset mining is to find the itemsets that appear together frequently in many entries of a database. An itemset is defined to appear frequently if its support is greater than or equal to a minimum support threshold. Support is defined as the number of occurrences divided by the total number of entries. Apriori [3] is introduced as an algorithmic solution, as a brute-force approach is too computationally expensive. The algorithm prunes the search space with the Apriori principle: $\forall X, Y : (X \in Y) \implies support(X) \geq support(Y)$. This allows for pruning of all supersets of a set which is found to be infrequent. The inclusion of a user-defined minimum support threshold is extremely common in local pattern mining techniques, to ensure that the subgroup occurs frequently enough to be relevant.

2.1.2 Association Rule Mining

Frequent itemset mining was actually introduced as a subproblem to the problem of mining association rules. which show what itemsets appear frequently with what other itemsets. They are defined by an implication X $\rightarrow$ Y, where X and Y are itemsets and the occurrence of X leads to the occurrence of Y. X is called the antecendent and Y the consequent. The goal of association rule mining is to find all association rules whose support and confidence pass a minimum support and confidence threshold. This means that for these association rules, if the antecendent occurs in an itemset, it is likely the consequent occurs in that itemset as well. This problem was introduced by Agrawal et al. [2] who solved the problem with the AIS algorithm. A year later, the Apriori [3] algorithm was introduced which outperforms AIS. Finding the high confidence association rules with a brute-force approach is too computationally expensive. Instead, they can be obtained from the frequent itemsets. A straightforward algorithm to this end is: "For every large itemset l, find

all non-empty subsets of l. For every such subset a, output a rule of the form $a \implies (l - a)$ if the ratio of support(l) to support (a) is at least *minconf*."

2.1.3 Contrast Set Mining

Closely related to association rule mining is the problem of contrast set mining [4]. In contrast set mining, the entries in a database are first split in mutually exclusive groups. The goal of contrast set mining is to find all contrast sets whose support differs meaningfully across groups. A contrast set is defined by a conjunction of attribute-value pairs, similar to how X and Y in association rule mining are defined. This definition of a subgroup is also used for EMM and is very common within local pattern mining techniques. The contrast set then consists of all database entries for whom this conjunction holds true. The support of a contrast set in this goal is the support of that contrast set with respect to a group. Thus instead of comparing the contrast set size with the database size, it is compared with the group size. The STUCCO algorithm [4] finds all contrast sets, while pruning in three different ways to reduce computational cost.

2.1.4 Emerging Pattern Mining

Another problem in the domain of itemsets is emerging pattern mining [6]. Here we have two datasets of itemset entries. An itemset is again described by a conjunction of attribute-value pairs. The growth-rate of an itemset X from dataset D_1 to dataset D_2 is defined by the support of X in D_2 divided by the support of X in D_1. An itemset is said to be an emerging pattern if its growth-rate is greater than or equal to a given value ρ. The goal of emerging pattern mining is then to find all emerging patterns. It is similar to contrast set mining as we could consider the datasets in emerging pattern mining to be the same as the groups in contrast set mining. The definition of a contrast set and an emerging pattern does differ however. Algorithmic solutions include MBD-LLBORDER [6] and iEPMiner [11].

2.1.5 Subgroup Discovery

In subgroup discovery [30], a set of objects with properties is given. The goal of subgroup discovery is to find the subsets of objects which are the most statistically skewed with respect to a chosen target property. The subgroups are described by rules, such that they are linked to relations between different properties. A rule is defined as $condition \rightarrow target_{value}$, where *condition* is a conjunction of attribute-value pairs on the object property values and $target_{value}$ is a condition on the target property value. Algorithmic solutions for subgroup discovery differ on four main elements. First the allowed type of target variable may differ. The three possible types are binary, nominal and numeric target variables. An algorithm may allow for any combination of these types. Second, the description language may differ. Third, they may use different quality measures. A quality measure is used to give a value to the interestingness of a subgroup. A wide array of quality measures exist already [15]. Fourth and final, they may employ a different search strategy. A straightforward exhaustive search is not feasible as the search space is exponential in the number of properties.

2.1.6 Exceptional Model Mining

Exceptional Model Mining (EMM) is a framework that was introduced in 2008 by Dennis Leman, Ad Feelders and Arno Knobbe [21]. It originated to eliminate a drawback that subgroup discovery had. Subgroup discovery finds subsets of a dataset for which the distribution of a target attribute differs substantially from the distribution of the entire dataset for that target attribute. The shortcoming of subgroup discovery is that the target attribute must be a single target variable. EMM eliminates this drawback by using a class model over multiple targets. Various class models have been discussed for different purposes, such as correlation, association, simple linear regression, classification, general linear regression (all [9]), Bayesian Networks [10] and rank correlation [8].

For each of these models, one or more quality measures are discussed. A quality measure is needed to quantify the interestingness of the distribution between the subgroup target values and dataset target values.

2.1.7 Graph Pattern Mining

Graph Pattern Mining consists of multiple data mining tasks, each concerned with finding frequent graph patterns. A graph pattern [5] is a subgraph found within a collection of graphs or a single massive graph. A graph is a subgraph of another graph if there exists a subgraph isomorphism. It is considered frequent if it passes a user-defined support threshold. They are commonly used in the domains bioinformatics, cheminformatics, social network analysis, computer vision and multimedia.

As neural networks can be considered graphs, one might think these techniques could be applied to them. However, neural networks are structured according to the specification of the user instead of by the data. Therefore, different training datasets do not change the graph structure of neural networks. Frequent patterns found would thus not relate to any subgroup of data.

2.1.8 Graph Pattern Mining with EMM

Some research to employ EMM on graph related data has already been done. Kaytoue et al. [17] introduces a method for augmented graphs. Augmented graphs are graphs whose nodes or edges have attributes. Then EMM can obtain subgroups of data corresponding to subgraphs, by refining over these attributes as normal. They propose to use the χ^2 test of independence and Weighted Relative Accuracy measure as quality measure. As a result, they obtain exceptional contextual subgraph mining by using a method rooted in EMM on augmented graphs. Lemmerich et al. [23] propose first-order Markov chains as a new model class to find exceptional transition behavior. Their quality measure is based on the distance between the Markov transition matrices of the subgroup and the overall dataset.

Again, these techniques can not be applied to neural networks as their structure does not depend on the data, but instead on user specification.

2.2 Convolutional Neural Networks

A convolutional neural networks is a type of artificial neural network that includes a convolutional layer. It is these type of neural networks that this thesis combines with EMM. A convolutional layer is useful to detect patterns with spatial relations in data and it also retains these relations.

Convolutional neural networks are useful for complex tasks for which the data has spatial relations. Domains of convolutional neural networks include images, videos, speech and natural language. In the domain of images, common tasks are facial recognition [28], handwriting classification [25] and object recognition [18]. As videos are a sequence of images, convolutional neural networks can be used for the same tasks in videos as they were used for images. In the domain of speech, they are mainly used for speech classification [1]. For natural language processing they are used for text classification [32] and sentiment analysis [7].

The convolutional neural network used in this thesis is trained for object recognition tasks. While the image datasets experimented with in this thesis are commonly used, they have a moderate complexity. In Section 3.2, we will see more of the related work in convolutional neural networks that is used in this thesis.

Chapter 3

Preliminaries

The preliminaries of this work are exceptional model mining and neural networks. Both are discussed in this chapter.

3.1 Exceptional Model Mining

EMM considers a dataset to be a set of records where a record r is given by $r = \{a_1, ..., a_k, t_1, ..., t_m\}$. Here, the k attributes a are called descriptors and the m attributes t are the targets. The set of descriptors a is taken from the domain $\mathcal{A}$. We can then select subgroups based on the descriptors by use of a description D, which is a conjunction of attribute-value pairs. The subgroup consists of all records that satisfy the description. The set of possible descriptions is called the description language, denoted by $\mathcal{D}$. A quality measure φ is a function $\varphi : \mathcal{D} \to \mathcal{R}$ that maps a description to a value. Then, the most exceptional subgroups are those for which their quality measure value differs the most from the quality measure value of the entire dataset. Alternatively, subgroups can be compared to their complement instead to find a different kind of exceptionality. Likewise, the most exceptional subgroups are then those for which their quality measure value differs the most from the quality measure value of its complement.

Due to the enormous search space of subgroups, which is possibly exponentially large in the number of records [9], EMM typically opts for an heuristic algorithmic strategy. A beam search is employed, where for each depth the best w descriptions are kept in the beam and used for refinement on the next depth. These best w descriptions are the subgroups with the highest quality measure value found until then. Refinements of a description D are made with each descriptor a. The refinements of a descriptor a are based on its type, which can be either binary, nominal or numeric. They are made as follows:

- Binary: add $D \cap (a = 0)$ and $D \cap (a = 1)$
- Nominal: add $D \cap (a = v)$ and $D \cap (a \neq v)$ for each value v of a
- Numeric: first compute split points $s_1, ...s_{b-1}$ where $s_j = v_{(\lfloor j \frac{n}{b} \rfloor)}$ from the sorted list of values $v_1, ..., v_n$ of a. The number of bins b is given as parameter to the algorithm. Then add $D \cap (a \leq s_j)$ and $D \cap (a \geq s)$ for each split point. There is another option available.
  - If we choose the bins to be determined statically, the entire list of values $v_1, ..., v_n$ of a is passed to determine the split points.
  - If we choose the bins to be determined dynamically, the entire list of values $v_1, ..., v_n$ of a is first filtered. Only the values that correspond to the set of records r where $D(a) = 1$ of current description D are used to determine the split points. This dynamic discretization is preferred, as more information is obtained [9].

These refinements cause the algorithm to obtain more specific and smaller subgroups on higher depths. If the quality measure value of the worst candidate subgroup in the beam is lower than that of a newly considered subgroup, it is replaced. Furthermore, a minimum subgroup size or support can be used to disregard subgroups that are not large enough to be interesting. A non-trivial exhaustive EMM algorithm named GP-growth is introduced in [22] based on an extension of the Frequent Pattern tree data structure [13]. However, this algorithm does not work for every model class.

3.2 Neural networks

Neural networks are a technique to solve various learning tasks of complex data. The neural networks are not explicitly programmed with rules, but instead self-learn characteristics of the data. A neural network consists of various sequential layers which each have a number of artificial neurons. An artificial neuron takes a vector of numeric inputs $x_0, ..., x_n$ and transforms it to a single output, as seen in Figure 3.1. It has a vector of weights $w_0, ..., w_n$ corresponding to each input it has, and a possible bias b. The output of a neuron is then the sum of its inputs multiplied by their weights plus the possible bias value. Usually this output is then transformed by an activation function ϕ, to ensure the output is within a desirable range.

$$output = \phi(\sum_i (w_i x_i) + b)$$

After forwarding the initial input through the entire neural network, we obtain an output. To attempt to obtain the correct output, the neural network first learns on a training dataset. This training dataset has example inputs and might have outputs depending on whether the problem we try to solve is a supervised problem. For the learning phase, a loss function L, also called cost function, is first defined. The loss function is a function that takes the input vector x, desired output y, weight vector w and bias b. Its output should have the lowest value, typically 0, when the optimal solution is found. It is used as a reference point to determine in what direction weights should be changed so that the loss function lowers. The changing of weights is done by a process called back propagation.

Back propagation updates a weight according to an equation that uses stochastic gradient descent. It first computes the output with the current weights. Then, the loss rate can be obtained by comparing the output with the correct output. Based on the loss rate, the weights are updated by applying the chain rule on local derivatives. A learning rate parameter α also impacts how heavily the weights are updated. A single weight update is as follows:

$$w_0 \leftarrow w_0 - \alpha \frac{\partial}{\partial w_0} L(x, y; w, b)$$

By doing enough passes, also called epochs, over the training data, the loss function lowers. It reaches a local optimum if α is sufficiently small.

Convolutional neural networks add convolutional layers to their structure. The idea of convolutional layers was introduced in 1980 [12], after which in 1989 [19] back propagation was applied to the concept. The convolutional layers allow for learning of spatial relations in the data. The neurons of a convolutional layers are actually matrices or windows of weights, also called filters or kernels. These matrices slide over the input and compute the dot product at each position to produce an output value, as seen in Figure 3.2. The output values retain the spatial placement of the input. The collection of the output values of a neuron is called a feature and the set of all features of a convolutional layer is called a feature map.

Other layers of neural networks besides the dense layer, described in the beginning of this section, and the convolutional layer, include max pooling and dropout. A max pooling layer [20] is frequently used after a convolutional layer. Similar to convolutional layer, a max pooling layer splits up the input in multiple windows. Then in each window, only the maximum value is retained. This leads to a reduction in variables that are used for training subsequent layers.

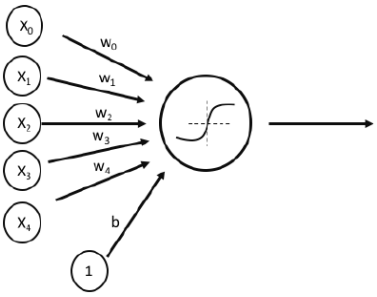


Figure 3.1: A single neuron, from TU/e slides of course Recommender Systems

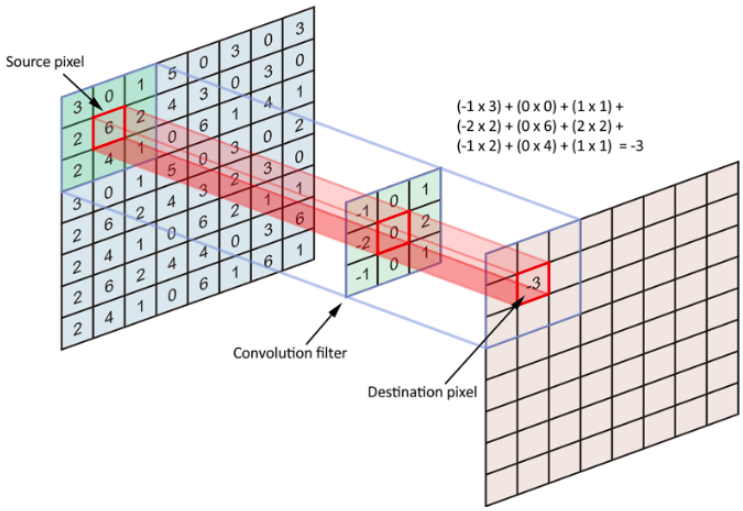


Figure 3.2: The effect of a convolutional layer, from [27]

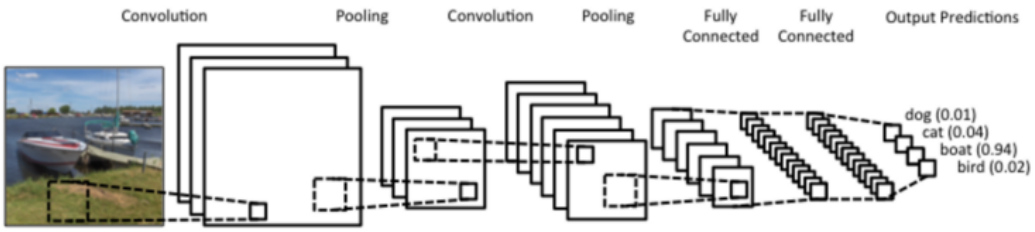


Figure 3.3: A convolutional neural network, from [29]

A dropout layer [26] causes a number of randomly selected neurons of the previous layer to be deactivated during training. This random selection is done for each given input. It causes the neurons to learn patterns that are independent of each other, which reduces overfitting on the training data.

A special activation function that is also used in this thesis is the softmax activation function. It is typically used on the final layer as it normalizes the output of all neurons on that layer to sum to one, similar to a probability distribution. It is useful for classification tasks, as the classes can each correspond to a neuron. Then the class with the highest probability can serve as the output. A convolutional neural network then processes a image by going through each subsequent layer, as seen in Figure 3.3.

Chapter 4

Method

We saw that numerous models have already been explored in the context of EMM, but neural networks are quite different from these models. Abstractly speaking, our first approach creates descriptions corresponding to candidate subgroups and trains a neural network for each candidate subgroup. We also train a neural network with the entire dataset or neural networks with the complement of each candidate subgroup. Then we compare the candidate subgroup neural networks with either the entire dataset neural network or complement neural network. We use some quality measure for this goal, to get a numeric value. The candidate subgroup with the highest quality measure value is then the subgroup with the highest exceptionality. However, training data of neural networks is quite different from the data used for models such as correlation. In training data, we do not appear to have a target variable, while in correlation datasets, the target variables are included in the datasets. Instead, the training data leads to a neural network which should provide target variables. In the next section, an approach to this problem is discussed. After that, we see a quality measure for this approach. Even later on in this chapter, from Section 4.4 onwards, we introduce a second approach to employ EMM on convolutional neural network training data.

4.1 Convolutional neural networks as an EMM model

The structure of training data for a neural network can vary wildly depending on its purpose. They are examples of the data that needs to be learned. They can be images, natural language or speech. What they share in common is that they must be encoded into a vector. The vector is then used as input to the neural network. Besides this vector, we might also have a ground truth vector, if we are doing supervised training. This ground truth represents the desired output value, such as a class label for a classification task. For unsupervised training, there is no ground truth value in the training data. In the following subsections we will discuss the choice of neural network architecture and EMM descriptors.

4.1.1 Neural network architecture

There is only a single restriction on what neural network architecture can be used. The proposed quality measures that we will see later requires there to be a convolutional layer. A high number of neural networks needs to be trained, so the training time of a model needs to be taken into consideration. If possible, use layers that speedup the training process, such as max-pooling. While a good test accuracy is not necessarily needed, the architecture should be representative of what is normally used. Using a architecture that is not complex enough leads to irrelevant conclusions. Thus the training and test accuracy should be at least somewhat in-line with their expected values.

A single neural network architecture should be defined. Each subgroup starts with a fresh

instance of this architecture. Each fresh instance should start with the same weight initialization, so that each neural network instance begins training from the same starting weights. This single weight initialization should still be randomized according to some random distribution, so that they are symmetry-breaking. Otherwise, if each convolutional filter on a layer initializes to the same starting weights, they would end up learning the same feature.

4.1.2 Descriptors

A simple approach to model neural network as an EMM model would be to use the values in the input vector of the neural network as the descriptors. This gives plenty of descriptors. This approach has two disadvantages. The input to a neural network is normally high dimensional. The first disadvantage is that a singular value of a high dimensional vector is too specific. Refining with a singular value filters out too much of the data, to the point where it tells very little about the relation with the entire dataset or complement. The second disadvantage of this approach is that the descriptions resulting from these descriptors are unfit for interpretation. It is not useful to reason about a subgroup description based on one or a few values of a high dimensional vector. However, when singular values represent binary or nominal variables, it is more natural to reason about the descriptions that they lead to. If they are also not too specific, they could be fit as a descriptor. However, different descriptors are needed in most cases.

If doing supervised learning, we can consider the element(s) of the ground truth vector to be used as a descriptor. This would lead to candidate subgroups that are fit for interpretation. However, subgroups that are produced with these descriptors lead to the class imbalance problem [33]. This problem occurs when the training data distribution is heavily skewed towards a single or few classes. When we use the ground truth variable as descriptors, we get an extreme case of class imbalance since we restrict the candidate subgroup to only either a single class or a few class. Thus the neural network is trained to never predict classes that fall outside of the candidate subgroup. In the extreme case of a description that refines to only one class, it leads to a candidate subgroup with training data on which neural network trivially trains to always output that class. Then the training accuracy is 100%, but the neural network barely needs to train. So these descriptors are also unfit. The suggested solutions to the class imbalance are unfit due to the purpose of EMM. Solutions such as over-sampling and under-sampling change the distribution of the training data by adding copies of training examples. The distribution would then be similar to the whole dataset, while the purpose of EMM is to find exceptional subgroups which correspond to differing distributions. These copies would also violate the descriptions imposed on the subgroup, as their classes would not be included in the description. Another solution named threshold-moving suggests to punish wrong predictions harder if they are from the underrepresented class. This would not be suitable since our descriptions can lead to subgroups where underrepresented classes exist with zero training examples. Thus this change has no effect during training.

A third consideration would be to aggregate the values of the training data. The choice of what values to aggregate is highly dependent on the dataset, and should thus be chosen on a case-by-case basis. The aggregation should be chosen such that we obtain features as descriptors that lead to descriptions that are fit for interpretation and are not too specific anymore. Furthermore it is important that they do not directly relate to the targets, as this leads to class imbalance problem again.

In Section 5.1 we will see our choice of descriptors for the two datasets that we experiment with. After the descriptors are chosen, the refinement process of EMM obtains candidate subgroups that are used for training. It is important to minimize the randomization of the training process, and preferably eliminate it. Otherwise, the quality measure value is impacted by the randomization and thus not consistent. After a model is trained, we need a quality measure.

4.2 A quality measure for convolutional neural networks

The quality measure should indicate the exceptionality of a subgroup in comparison with either the entire dataset or the complement of the subgroup. The subgroup and either the entire dataset or the complement give us two trained neural networks, which each contain a set of weights per layer. Our quality measure compares these two trained neural networks. It needs to results in a numeric value, where a higher numeric value should indicate that the subgroup is more exceptional. Then we can find the most exceptional neural network in the EMM framework, which is the candidate subgroup with the highest quality measure value. The definition of exceptionality thus depends on our quality measure.

As stated before, we require the neural network architecture to contain a convolutional neural layer. The convolutional layer consists of a set of kernels, otherwise known as filters. Such a kernel represents a pattern that the convolutional neural network attempts to detect. Intuitively, if the training data of two subgroups is similar, the neural networks learn the same patterns and thus end up with equal weights on their convolutional layer. So we use a quality measure that can measure the distance between the convolutional layers.

To do so, we compute the pairwise distance of each kernel of the first model to each kernel of the second model. Now it could be the case that both neural networks have the same kernels, but in a different permutation. Thus we have to find the matching of each kernel of the first model with a distinct kernel of the second model, such that we obtain the minimum total distance. This is an instance of the assignment problem and numerous algorithm exist to solve it, such as the Hungarian algorithm (also known as Kuhn-Munkres algorithm or Munkres assignment algorithm). The solution to this assignment problem is the minimum total distance, which we use as the value of the quality measure.

Formally, we have two convolutional layers, $conv_A$ of neural network A and $conv_B$ of neural network B. Both convolutional layers are defined as a set of k kernels, where each kernel has equal dimensions m by n. Let $w_{2,3}^{A,1}$ represent the weight on the second column and third row of kernel one of the convolutional layer of neural network A. Let C be a matrix of k by k. Let $C[i,j]$ be the distance of kernel i of $conv_A$ to kernel j of $conv_B$. This distance is computed by

$$C[i,j] = \sum_{u=1}^{m}\sum_{v=1}^{n}(|w_{u,v}^{A,i} - w_{u,v}^{B,j}|)$$

After computing this distance for every cell, we have the resulting cost matrix C. This can then solved by the Hungarian algorithm or a similar algorithm. The resulting value is the quality measure value.

We can additionally choose which convolutional layers of the neural network architecture should be included in the quality measure value. Herein lies a choice, we could compare each convolutional layer or simply the first convolutional layer. Choosing the first convolutional layer allows for more insight into the quality measure value, as their kernels represent features that are more easily interpretable. Furthermore, they also allow for a significantly lower computational cost for two reasons. First, there simply are less convolutional layers to compare. On top of that, every convolutional layer after the first one usually has significantly higher dimensions due to the workings of convolutional layers. Whereas the first convolutional layer only has depth 1, all others convolutional layers have the number of kernels on the previous convolutional layer as depth. It is suggested to use the first convolutional layer. The experiment in Section 5.4 is done to give a basis for this preference.

4.2.1 Runtime analysis

The worst-case computational complexity of the EMM algorithm, according to Theorem 1 of [9], is:

$$\mathcal{O}(dwkN(c + M(N,m) + \log(wq))) \tag{4.1}$$

These symbols represent:

- d: Beam depth
- w: Beam width
- k: Number of descriptors
- N: Number of records in the dataset
- c: Cost of comparing two models from the model class
- m: Number of targets
- *M(N, m)*: Cost of learning a model from N records on m targets
- q: Result set size

For the proposed CNN class model, we only impact the comparison cost c and the model learning time *M(N, m)*. The comparison cost is comprised of creating the cost matrix and solving it.

The cost of creating the cost matrix is dependent upon the number of convolutional layers that are used for comparison, the number of kernels, and the kernel size of each convolutional layer. If only the first layer is used, this is reduced to only the number of kernels and kernel size of that layer. Let us denote the number of kernels as r and the kernel size as s^3, where s is the largest kernel dimension. This corresponds to a 2D convolutional layer, a 3D convolutional layer's kernel size would be s^4. The number of cells in the cost matrix is r^2. Each cell corresponding to a distance between two kernels compares each weight of the two kernels only once with another weight, thus each cell has $\mathcal{O}(2s^3) = \mathcal{O}(s^3)$ comparisons. As each comparison takes $\mathcal{O}(1)$, the cost of creating the cost matrix is $\mathcal{O}(r^2 s^3)$.

If t convolutional layers are used, this cost increases. Let us denote the number of kernels of convolutional layer i as r_i. The first convolutional layer has a kernel depth similar to the input depth. On subsequent convolutional layers, the kernel size differs, as the depth dimension of the kernel on convolutional layers after the first is equal to the number of kernels on the previous convolutional layer. Thus that kernel size is $s_i^2 r_{i-1}$. Now we have a cost matrix for each convolutional layer with $\mathcal{O}(r_i^2 s_i^2 r_{i-1})$ comparisons and cost. We can then sum this cost over each convolutional layer as well as include the first layer to obtain the cost of creating the cost matrix

$$\mathcal{O}(r_0^2 s_3^2) + \mathcal{O}(\sum_{i=1}^{t-1} r_i^2 s_i^2 r_{i-1})$$

For the cost of solving the cost matrix, we assume the Hungarian algorithm is used. The worst-case running time of the Hungarian algorithm is $\mathcal{O}(n^3)$. Here, n is the size of the largest dimension of the cost matrix, which corresponds to r_i. So the cost of solving the cost matrix is $\mathcal{O}(r_i^3)$ for each convolutional layer i. This is summed to $\sum_{i=0}^{t-1} r_i^3$.

The comparison cost c for the first layer only is thus $\mathcal{O}(r_0^2 s_0^2 + r_0^3)$. If comparing all t convolutional layers, the comparison cost c becomes

$$\mathcal{O}\left(r_0^2 s_0^3 + \sum_{i=1}^{t-1} r_i^2 s_i^2 r_{i-1} + \sum_{i=0}^{t-1} r_i^3\right)$$

After filling this equation into Equation 4.1, we obtain the following worst-case computational complexity for this quality measure:

$$\mathcal{O}\left(dwkN\left(r_0^2 s_0^3 + \sum_{i=1}^{t-1} r_i^2 s_i^2 r_{i-1} + \sum_{i=0}^{t-1} r_i^3 + M(N,m) + \log(wq)\right)\right) \tag{4.2}$$

The machine learning time $M(N, m)$ cannot be expressed in similar terms and is therefore left abstract in this analysis. In practice, it is far greater than the comparison cost, if only using the first convolutional layer, and far greater than the term $\log(wq)$.

4.3 Additional remarks

Now that we have proposed a way to incorporate convolutional neural networks into EMM, there are still a number of smaller related topics to this model and quality measure to be discussed.

4.3.1 Epochs relative to subgroup size

A neural network uses the training set for training. It optimizes its loss function with either the entire training set at once, or it splits up the training set in batches and optimizes its loss function with each batch once. This process, called an epoch, is then done a set number of times.

A subgroup contains only a subset of the training data. The training process for the neural network of the subgroup thus trains on a lesser amount of data. This results in an unfair comparison between the neural networks of the subgroup and either the whole dataset or the complement. A smaller subgroup trains less, thus it changes its weights less. This results in a higher distance than a larger subgroup normally obtains, since the weights of a subgroup rarely train in the opposite direction compared to the whole dataset or its complement.

To counteract this issue, a dynamic number of epochs should be used per subgroup based on its size. Let n_{base} be the number of training examples the of base neural network (whole dataset or subgroup complement) trains on and let e_{base} be the number of training epochs. We also know $n_{subgroup}$ and need to determine $e_{subgroup}$. We would like to compare neural networks that have optimized on an equal number of training examples, so we determine the number of epochs of the subgroup as follows:

$$n_{base} \cdot e_{base} = n_{subgroup} \cdot e_{subgroup}$$

We can solve this for $e_{subgroup}$. In addition, the number of epochs has to be an integer, while the obtained value could have a fraction. So we have to round it, for which we use the ceiling function. The floor function or rounding to the nearest integer would be fine as well. So we obtain:

$$e_{subgroup} = \left\lceil \frac{n_{base} \cdot e_{base}}{n_{subgroup}} \right\rceil \tag{4.3}$$

4.3.2 Average of multiple CNN

When we discussed the neural network architecture, we stated that every subgroup should started from the same initial starting position. This was done by using the same neural network architecture and initializing the convolutional layer(s) to the same weights. This does have a drawback, different initializations could give us a different ordering of the subgroups by exceptionality. To negate this, we can use more neural networks of the same architecture with a different weight initialization. Then each subgroup would train on these additional neural network as well, resulting in additional quality measure values of a subgroup per neural network. The final quality measure value of the subgroup is then the arithmetic mean of each neural network quality measure value.

As neural network architecture complexity increases, training time increases as well. Thus multiplying the training time by the number of neural networks per subgroup could become very costly. Fortunately, as neural network architecture complexity increases, the variance in the ordering of subgroups when using different convolutional layer weight initializations reduces. Thus more complex neural network architectures have less need of these additional neural networks.

A different or complementing approach is to simply increase the number of results returned by the EMM algorithm and thus increase the number of subgroups that we can investigate. When we do not use multiple CNN, the downside is that we can be less confident that the most exceptional subgroups found are truly the most exceptional subgroups out of all the subgroups that were inspected.

Section 5.5 will give an indication of the relation between CNN complexity and the necessity to average over multiple CNNs.

4.4 A second quality measure

Neural networks can be considered to have two tasks, representation learning and machine learning. Representation learning takes the input of the neural network and encodes it to a different representation, usually with a different dimensionality. An example of this would be word embedding [24]. During natural language processing a list of words needs to be given of input. As this list can be incredibly large, representation learning is used to transform the list into a vector with a much smaller dimension. Machine learning then maps the new representation to a target variable.

For convolutional neural networks, the convolutional layers are used for representation learning. Their neutrons or kernels correspond to patterns that are detected on images, resulting in features. These features serve as a new representation instead of pixel values. We use the features to obtain the exceptionality of an image. A feature with a higher sum of its values is deemed more activated than one with a lower sum value of its values. We consider a feature of an image to be exceptional if it has an extremely low or high value, compared to the overall distribution of that feature for the set of images. A low value indicates that the feature does barely activate on that image, thus the sought pattern does not or barely appear in the image. A high value indicates the feature activates abundantly on that image, thus the sought pattern abundantly appears in the image. The underlying intuition is that on one hand an image with many barely activated features might not be well represented by the learned representation. Therefore it might be hard for the neural network to correctly classify it, but this is not a necessity. In fact, the absence of the feature could actually be an indicator of a certain class. It could also be the case that a class is so trivial to classify due to having distinct images that very few specific kernels are needed to identify the class. Still, a subgroup composed of this kind of images would be interesting. On the other hand, an image with many abundantly activated features is well represented by the learned representation and in turn might be easier to classify. Again, this is not a necessity and the exact opposite might hold. The image could be hard too classify because a high number of features is activated, or a high number of features is trained for such an image as it is hard to classify due to its complexity.

The quality measure aims to find subgroups with images who have the highest number of such features with exceptionally low or high values. We do also have to take the values of the input image itself into account. An input image with overall high values has more potential to have features with extremely high values than an input image with average values. As such, to determine whether a feature is exceptionally high or low, we look at the fraction of the feature values sum divided by the input image sum.

In contrast to the previous quality measure, we only want a single neural network. This way, we learn a single feature representation for the entire dataset. Then, for each subgroup, we obtain the feature maps of each image in that subgroup. For each feature of each feature map, we sum the values of the feature. This sum is divided by the corresponding input image sum. The resulting value is compared with a lower threshold value and upper threshold value to deem it exceptionally high or low. If the feature is deemed exceptional, we increment the subgroups count of exceptional features. Finally, we need to take into account the subgroup size, as they have more input images and thus a higher count of exceptional features. Thus the final quality measure value is the final count divided by the subgroup size.

This computation can be further optimized. An input image of the dataset is expected to be included in many subgroups. So the number of total images considered throughout the candidate subgroups of the EMM run is usually far greater than the dataset size. If we simply precompute the count of exceptional features for each image in the dataset, instead of computing it multiple times for most images during the EMM run, we reduce the number of times we have to compute this count to once for each image. Then during the EMM, we can simply retrieve the count of exceptional features of all images in a subgroup. We sum these counts and divide the sum by the subgroup size to obtain the quality measure value. Pseudocode of this precomputing step can be found in Algorithm 1.

The threshold values have to be determined in some way. This leads to a few problems. First, the value sum of a feature can vary a lot for each feature. Secondly, features of two different

convolutional layers can have different dimensions, as a convolutional layer or max pooling layer decreases the dimension of the image. This leads to inherent lower values of later convolutional layers. Thirdly, values of two different datasets vary wildly. Therefore a single lower and upper threshold value cannot be used, and ideally should be determined dynamically per problem per layer per feature. A solution to this problem is not included in this thesis, but suggestions for future research are made in Section 7.2. For now, a set of manually determined threshold values is suggested, which should be determined by looking at the distribution of the fraction of feature sum divided by input image sum, per image. Since the number of features is too high, we merely determine threshold values for each layer per problem. This is still feasible and the distribution does not change too much per feature on a single layer.

Algorithm 1 getCounts(convolutional_layers, dataset)

```
for layer in convolutional_layers do
    for image in dataset do
        feature_map ← layer.getFeatureMap(image)
        counts[image][layer] ← 0
        for feature in feature_map do
            if sum(feature)/sum(image) < lower_threshold OR sum(feature)/sum(image) > upper_threshold then
                counts[image][layer] ← counts[image][layer] + 1
            end if
        end for
    end for
end for
return counts
```

If we run an EMM run with comparing to the whole dataset, the quality measure value is the difference in average count per image for the entire dataset and the subgroup. If we run with the complement option, the quality measure value is the difference in average count per image between the subgroup and its complement.

4.4.1 Alternative approach

A simpler, alternative approach is to directly use the fraction $\frac{\text{feature map sum}}{\text{image sum}}$ as the quality measure value. That is, the sum of all values in each feature map of all layers divided by the sum of all values in the input image. This would eliminate the effort needed to obtain appropriate threshold values. However, the results when using this quality measure are insignificant. Without dividing by the image sum, the most exceptional images would simply be those with the most image sum. They would simply have more input on which the features can activate. Thus we divide by the image sum, which should make it so that the image sum does not have an impact on the quality measure value of a subgroup, as we use it to normalize this value. This is false in practice, as the input images with the least image sums were found to be the most exceptional. In fact, the input sum is almost directly proportional to the rank of the image. For this reason, the more complicated approach where we determine whether each individual feature is exceptional is preferred. It does however not complete eliminate this relation, as we will see in Section 5.6 and 5.7.

4.4.2 Runtime analysis

We start our runtime analysis again from Equation 4.1. As before, we only impact the model learning time $M(N, m)$ and comparison cost c. We furthermore have two extra factors, preprocessing P and training a single neural network T. Both of these factors occur outside of the EMM algorithm, so including them into the worst-case computational complexity results in:

$$\mathcal{O}(T + P + (dwkN(c + M(N, m) + \log(wq))) \tag{4.4}$$

The training time for a neural network is highly dependent on the neural network complexity and training settings. It is usually smaller than the EMM algorithm runtime.

The preprocessing step consists of computing the number of counts of features surpassing the boundary threshold. The steps are shown in Algorithm 1. As in our previous quality measure running time analysis, we denote the number of convolutional layers by t. The size of the feature map referenced on line 5 is equal to the number of kernels on convolutional layer i, denoted by r_i. Then obtaining the sum of a feature is $\mathcal{O}(p_i^2)$, where p_i is the highest dimension of a feature on convolutional layer i. We assume that the image sums are available. They can be computed once outside of Algorithm 1, and their computational cost is far lower than that of Algorithm 1. We let the cost of getFeatureMap on line 3 remain abstract and denote it by F. All together we obtain $\mathcal{O}(\sum_{i=0}^{t-1}\sum_{i=0}^{N-1}(F+\sum_{0}^{r_i-1}p_i^2))$. This can be simplified such that we obtain:

$$\mathcal{O}(P) = \mathcal{O}\left(\sum_{i=0}^{t-1}(N(F+r_i p_i^2))\right) \tag{4.5}$$

The model learning time $M(N,m)$ is $\mathcal{O}(N)$, as we need to sum the count values of all images in the subgroup.

The comparison cost is simply a comparison between two sums of counts, thus it is $\mathcal{O}(1)$.

So we can conclude the that the worst-case computational complexity of the EMM algorithm with this quality measure, by filling it into Equation 4.1, is:

$$\mathcal{O}\left(T+\sum_{i=0}^{t-1}(N(F+r_i p_i^2)) + dwkN(N+\log(wq))\right) \tag{4.6}$$

Chapter 5

Experimental research

In this chapter, a number of experiments is described. For each experiment, first their setup is stated, then their results, and afterwards a brief discussion is conducted about the results.

5.1 Datasets

The experimental research is done on two different datasets, MNIST and CIFAR-100. These are discussed in the subsections below. Both are image datasets suited for classification tasks.

5.1.1 MNIST

The MNIST dataset[1] consists of images of 28 by 28 pixels depicting black and white handwritten digits. The values of the pixels are normalized to be between 0 and 1. A complete black pixel corresponds to a one value and a complete white pixel corresponds to a zero value. The depicted digits range from 0 through 9. We only make use of the training set, which consists of 60,000 images.

Since using the individual pixels as descriptors are too specific and lead to subgroups that are hard to interpreted, we use different descriptors. We use two kinds of descriptors for this dataset. The first kind is aggregation of each row and column into a descriptor. Thus we obtain 28 row descriptors and 28 column descriptors, which are all numeric descriptors. The second kind is descriptor that corresponds to the results of a clustering method. The clustering method we use is SpectACl [16], which claims to find different writing styles per cluster on the MNIST dataset. The parameters we use for the SpectACl algorithm are $r = 10$ and $\varepsilon = 10.0$. This results in a single nominal cluster descriptor, which has 10 clusters.

The CNN architecture that we use for MNIST can be found at Table A.12. Furthermore the ADADELTA [31] optimizer is used.

5.1.2 CIFAR-100

The CIFAR-100 dataset[2] consists of images of 32 by 32 pixels depicting colored classes. Each pixel has three channels, corresponding to its red, blue and green value. These values range from 0 through 255, but we normalize them such that they range from 0 to 1. Unlike MNIST, a complete black pixel corresponds to three zero values and a complete white pixel corresponds to three 255 values or three normalized 1 values. A total of 100 classes exists, which are split into 20 superclasses which each have 5 classes. The classes include animals, flora, people and objects. Each class has 500 training images. We use the total number of 50,000 training images.

[1] `http://yann.lecun.com/exdb/mnist/`
[2] `https://www.cs.toronto.edu/~kriz/cifar.html`

We use a different approach for the descriptors than MNIST. Instead of aggregating singular rows or columns, we aggregate entire color channels. So we obtain three descriptors; total red, total green and total blue. We also add another descriptor which aggregates all channels, named total color. All descriptors are numeric.

The CNN architecture that we use for CIFAR100 can be found at Table A.13. Furthermore the ADADELTA [31] optimizer is used.

5.2 First quality measure on MNIST

The first experiment is done on the MNIST dataset with the first quality measure. We run the EMM algorithm twice, with once comparing to the entire dataset and once to the subgroup complement. As the first quality measure has a high running time, we only explore through depth 3, using a beam width of only 3. The only difference in settings between the two runs is the comparison mode and the minimum subgroup size. We opt for a higher minimum subgroup size (1000 instead of 100) when comparing with the complement, as an extremely skewed split would obtain results similar to the entire dataset comparison. A split that is not as extremely skewed should thus be more meaningful for this setting. The full list of settings can be found at Section A.3.1.

5.2.1 Results

The most exceptional subgroup found when comparing with the entire dataset has description:

$$row6 \leq 1.5 \wedge row23 \leq 2.0 \wedge cluster = 8.0$$

A sample of this subgroup can be found in Figure 5.1a. It consists mostly of the digits 1, 4, 7 and 9. They are mostly written standing upright with a low font weight. The rest of the top 5 on depth 3 are nearly the same subgroup. They all include the description $row23 \leq 2.0 \wedge cluster = 8.0$. On depth 2 at rank 2, we can still find a completely different subgroup. This subgroup has description:

$$column24 \geq 6.1 \wedge cluster = 3.0$$

A sample can be found in Figure 5.1b. This subgroup consists solely of the digit 0. In fact, cluster 3 consists almost solely of the digit 0. We can also see that the drawn digits in this subgroup have a relatively high font weight. Samples of cluster 3 and 8 can be found in Figure 5.2a and 5.2b respectively. Furthermore, we see that at depth 1 the top 5 subgroups only consist of the cluster descriptor. This is most likely because the cluster descriptor is simply more refining than the row and column descriptors. A list of the full results is shown in Table A.1.

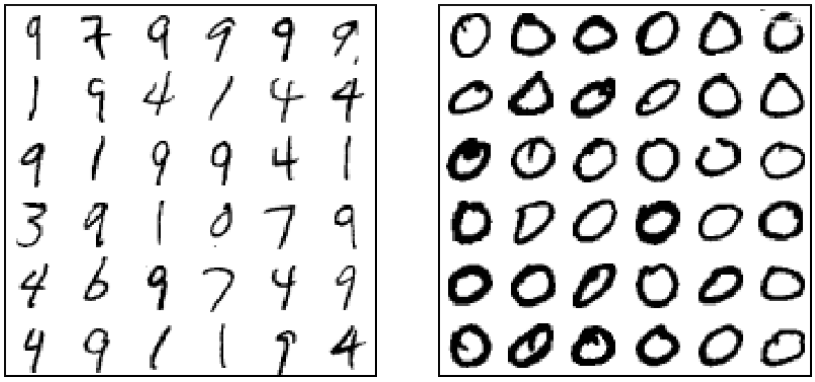
The most exceptional subgroup found when comparing with its complement has description:

$$column8 \geq 10.5 \wedge cluster = 3.0$$

A sample can be found in Figure 5.3a. This subgroup is extremely similar to the rank 2 on depth 2 subgroup of the entire dataset comparison. This subgroup consists solely of the digit 0. It has a subgroup size of 1029, barely staying above lower bound. We also see that this subgroup was discovered on depth 2, even though we explored depth 3 as well. The rest of the top 5 of depth 3 further consists of refinements of this subgroup, while the top 5 of depth 2 consists of very similar subgroups. It is only on depth 1 that we find very different subgroups. Its first place has description:

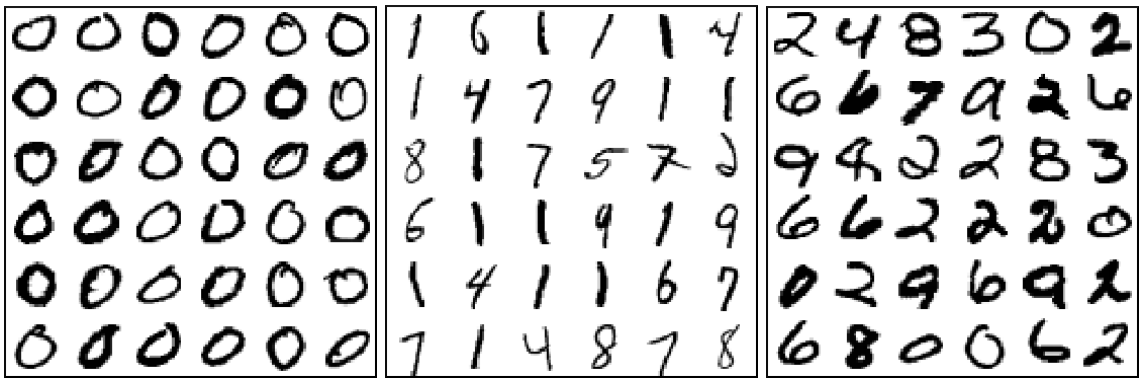
$$row3 \leq 0.0$$

A sample of its complement can be found in Figure 5.3b. The complement of this subgroup consists mostly of the digit 6, some digits 2 and sparingly of other digits. The complement has a total size of 3435, so the subgroup itself is very similar to the original dataset. A list of the full results of the complement run is shown in Table A.2.



(a) Rank 1 on depth 3 (b) Rank 2 on depth 2

Figure 5.1: Most exceptional subgroup samples for experiment in Section 5.2



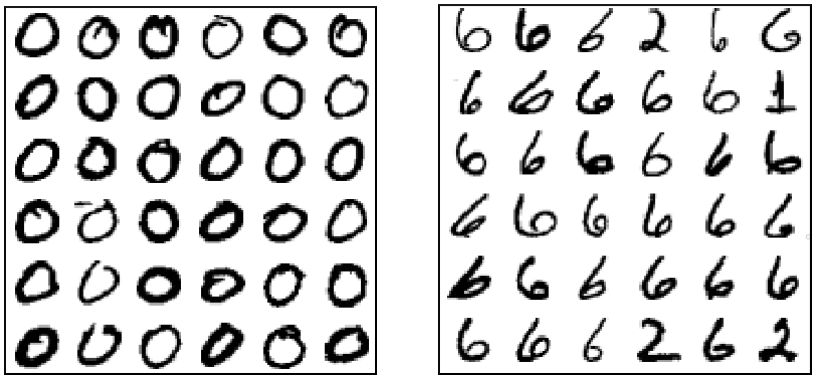
(a) Cluster 3 (b) Cluster 8 (c) Cluster 9

Figure 5.2: Samples of various clusters of SpectACl on MNIST

5.2.2 Discussion

It is not immediately apparent why the subgroup described at the start of the previous section, which consists mostly of the digits 1, 4, 7 and 9, is the most exceptional with our quality measure. The features learned for this subgroup differ the most from the features learned for the entire dataset. The digits seem to not be slanted in general, thus there might not be any or few features learned to detect slanted digits. These missing features might mostly account for the very high quality measure, but we cannot say this with certainty.

The other subgroup consisting mostly of zeros is more easily explained. The validity of this subgroup can be questioned due to the class imbalance problem as discussed in Section 4.1.2. The class imbalance means that the loss function that the neural network attempts to optimize during the learning process is quickly optimized. Therefore, the weights barely change, meaning that the quality measure comes close to the distance between the initial weights and the weights of the CNN trained on the entire dataset. This distance naturally is very large explaining the high quality measure value. This subgroup is ranked lower than the subgroup ranked first, thus we can note that the first ranked subgroup actually trained some of its weights in the "opposite" direction. Therefore the weights after its training process resulted in a distance higher than the random initialization. Most of the top ranked subgroups when comparing with the complement are very similar to this subgroup for the same reasons. Even the subgroup whose complement consists mostly of digits 6 and 2 is still very imbalanced. Its neural network only has to distinguish sixes



(a) Rank 1 on depth 3 (b) Rank 1 on depth 1 complement

Figure 5.3: Most exceptional subgroup samples for complement experiment in Section 5.2

and twos.

In the full list of results of the complement run, we can also see that the randomization of the training process is not completely eliminated. On depth 1 we have descriptions $cluster = 3.0$ and $cluster \neq 3.0$. These descriptions are each other's complement. Thus the two neural networks they each train and compare are trained with the same two data subsets. They end up with different QM values, respectively 32.0 and 31.8. Though these values do not differ much, it indicates there is still randomness left. The descriptions $cluster = 8.0$ and $cluster \neq 8.0$ show the same problem, indicating that it occurs consistently.

5.3 First quality measure on CIFAR-100

We carry out another experiment with the first quality measure, this time on the CIFAR-100 dataset. Similar to the first experiment, we do one run that compares to the entire dataset and one run that compares that to subgroup complement. The settings are very similar to the MNIST runs, except a higher number of epochs is used to achieve a reasonable accuracy with the CIFAR-100 subgroups. The EMM settings can be found at Section A.3.2.

5.3.1 Results

The most exceptional subgroup found when comparing with the entire dataset has description:

$$blue \leq 728.3 \wedge blue \geq 674.3 \wedge color \geq 1600.5$$

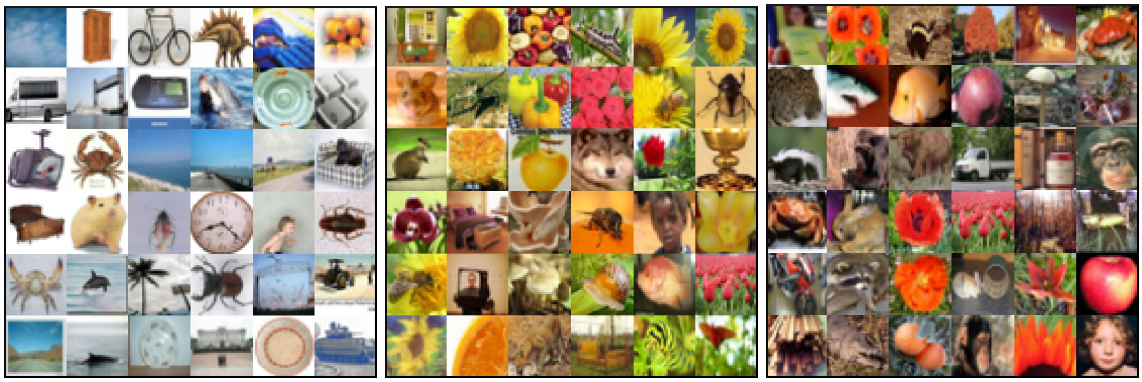
It consists mostly of objects and animals with a light background, sometimes a light blue background. It has a very varied number of classes and it does also not have a few classes with disproportional many images. The second ranked subgroup is similar, although it tends to include more environmental landscapes and more light blue backgrounds. The third ranked subgroup has description:

$$color \geq 1348.4 \wedge red \geq 469.1 \wedge blue \leq 366.3$$

The most common image is of yellow-green flora. The most common class is sunflowers. The classes are less varied than the other top ranked subgroups. The fourth ranked subgroup has description:

$$green \leq 430.3 \wedge color \geq 1176.6 \wedge blue \leq 366.3$$

It consists mostly of animals and flora. Brown and orange colors are common. Some images have very high red values, mostly images of the poppy flower class. The fifth ranked subgroup is similar to the first ranked subgroup, only it consist almost solely of object images with white background.



(a) Rank 1 on depth 3 (b) Rank 3 on depth 3 (c) Rank 4 on depth 3

Figure 5.4: Most exceptional subgroup samples for experiment in Section 5.3

(a) Rank 1 on depth 3 (b) Rank 1 on depth 3 complement

Figure 5.5: Most exceptional subgroup samples for complement experiment in Section 5.3

Samples of the subgroups ranked first, third and fourth can be found in Figure 5.4. A list of the full results is shown in Table A.3.

The most exceptional subgroup found when comparing with the complement has description:

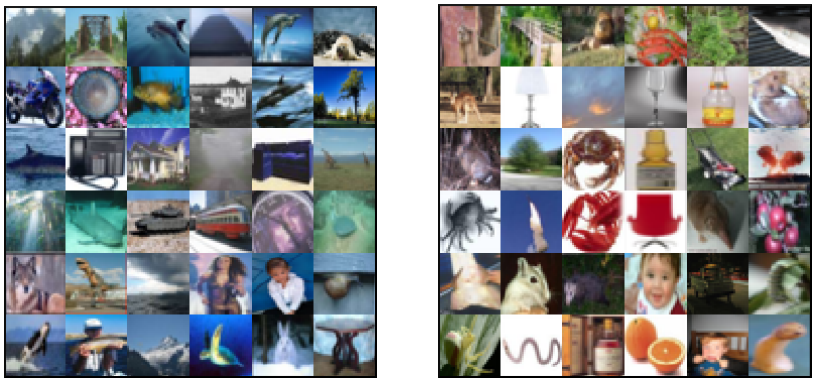
$$red \geq 847.1 \wedge green \geq 684.0 \wedge blue \geq 516.4$$

The subgroup of this description includes many images of objects or animals with white backgrounds. In contrast, the complement consists mostly of images with fully colored backgrounds, although there are sparingly some images with a white background as well. Samples of both can be found in Figure 5.5. This split is also seen with the second ranked description. Rank three and five are similar, but consist mostly of images with a light colored background.

The fourth ranked subgroup has description:

$$color \leq 1598.9 \wedge blue \leq 581.7 \wedge blue \geq 516.4$$

This subgroup leads to a different subset of images. As the descriptions indicates, these images contain a specific amount of blue, which is the highest bin on depth 1, but the lowest bin on depth 2. This results in a varied set of images, where usually this blue can be found in the background of sea or sky, but sometimes it is found in the object depicted. Samples of this subgroup and its complement can be found in Figure 5.6. A list of the full results is shown in Table A.4.



(a) Rank 4 on depth 3 (b) Rank 4 on depth 3 complement

Figure 5.6: Exceptional subgroup samples for complement experiment in Section 5.3

5.3.2 Discussion

It is hard to find the underlying reasons for the most exceptional subgroups. We can see that they tend to consist of mainly one color or a small range of colors, but this is simply due to the chosen descriptors. A reason for a big portion of the quality measure value is likely then the absence of weight training for the colors that are not present in a subgroup. For example, the subgroups depicted in Figure 5.4b and 5.4c have very little blue due to the $blue \leq 366.3$ part of their descriptions. Their kernel weights for the blue channel thus change very little during the training process in comparison to the neural network that was trained with the entire dataset, as the images featuring blue which train these weights are not included in the subgroup. Similarly, the subgroup depicted in Figure 5.6a is focused on blue images and therefore trains its red and green weights less.

The subgroup depicted in Figure 5.5a and to a lesser extent the subgroup depicted in Figure 5.4a contain a lot of white color instead, meaning they have a lot of red, green and blue. Thus they do not eliminate a color and they depict objects of various colors. Therefore the same reason as for the other subgroups does not hold. Due to the white background, the kernels need to detect the patterns on the images in a different way. The white background is detected by high values, while the object is detected by lower values, as it has less color itself. Perhaps this could be a considerable part of the high quality measure value, as this requires the weights to be trained in a significantly different way to be effective.

We also see that the most exceptional subgroups of the two runs are not too different. The subgroup sizes on depth 3 with 50,000 training images and 3 bins have an approximate size of $\frac{50{,}000}{3^3} \approx 1,850$. Thus the complement of such a subgroup would consist of approximately 48,150 images compared to the whole dataset size of 50,000. The negligible size seems to be the reason that the comparison choice barely has impact.

5.4 Convolutional layer selection

The quality measure compares the convolutional layers of two neural networks. However, as discussed in Section 4.2, a choice can be made about which convolutional layers to use. Two variants that we experiment with are using only the first convolutional layer and using all convolutional layers. We discussed already two reasons for preferring only the first layer, namely interpretability and computational cost. The disproportionately extra computational cost of using extra layers besides the first comes from the much higher depth of these layers. This complicates using all convolutional layers for the quality measure, since all convolutional layers except the first naturally have a higher distance due to having a higher depth. Therefore a third variant we investigate

is to use each convolutional layer but normalize them. The normalization that this experiment explores is to let each convolutional layer contribute equally to the final quality measure value. Thus the distance between a convolutional layer of two models, is the sum of the distance between the kernels of each depth, divided by the depth.

This experiment is performed on the MNIST dataset. A neural network architecture is used comprised of two convolutional layers. Both convolutional layer have 32 kernels, each with a size of 3 by 3. The second convolutional layer thus has depth 32 compared to depth 1 of the first convolutional layer. The complete configuration of the EMM run can be found at Section A.3.3. We run the EMM algorithm with each of the earlier described three variants.

We are interested whether the resulting set of subgroups differs a lot. The best 20 subgroups are returned for each subgroup. We compile the best 10 subgroups of each variant into a complete set. Then for each subgroup in the complete set, we check if they are listed in the best 20 subgroups of each variant.

5.4.1 Results

The obtained results can be found in Table A.5 in the Appendix. The complete subgroup consists of the top 10 subgroups of the first layer variant, plus the subgroups with description row9 ≤ 3.16 and row10 ≤ 3.05. These subgroups are marked with bold text in the table.

5.4.2 Discussion

We see that the top 10 subgroups of the quality measure with only the first layer are actually the same subgroups as the top 10 subgroups of the quality measure with all layers normalized. They only differ in order. The order of the subgroups ranked 11 through 20 is actually the exact same for these two quality measure variants. The reason for this becomes apparent when we take a closer look at how the quality measure values are build up. There occurs a consistent pattern, where the comparison distance between the first convolutional layer is approximately 3 to 4 times higher than the average comparison distance of the second convolutional layer per depth. Therefore, the quality measure for the quality measure variant of normalizing all layers is mostly influenced by the first convolutional layer. It makes sense that the results of these variants thus only vary little.

When comparing the first layer variant and all layers variant, the opposite happens. The comparison distance of the variant with all convolutional layers is much higher than the variant with only a single layer, approximately seven times as high. This is thus mainly attributed to the second convolutional layer. However, we still see very little variation in the results. Both variants have eight subgroups in common between their top 10 subgroups. There exist two subgroups in the top 10 of the all layers variant that are not contained the top 10 of the first layer variant. Description $row9 \leq 3.16$ at rank 4 for the all layers variant can still be found at rank 15 for the first layer variant. In the same way, description $row10 \leq 3.05$ at rank 10 for the all layers variant can still be found at rank 12 for the first layer variant.

The two subgroups of the first layer and all layers normalized variants top 10 have descriptions $column20 \leq 2.65$ and $column21 \leq 0.63$. These descriptions can still be found in the all layers variant top 20 at rank 11 and rank 14 respectively.

To conclude, we can answer the goal of this experiment as follows: All subgroups in the complete set can be found in the top 20 of each variant. From this experiment, we have shown that the three variants are related to the point where the results are quite similar. By simply increasing the number of results we inspect, each variant contains the set of best subgroups. Therefore the variant should not be chosen by the results it may offer, but rather for other reasons such as interpretability and computational cost. For both of these reasons, the first layer variant should be chosen.

Of course, this is not definite proof that these results hold for any other configuration or dataset, but it at least gives an indication of a preferred quality measure.

5.5 Number of Neural Networks selection

In Section 4.3.2 we proposed to use the average of a number of neural networks to account for randomization in the weight initialization of neural networks. We also stated that this becomes less necessary as the neural network complexity increases. In this experiment we show the effect of the randomization. This should also serve for a guideline on when to use multiple neural networks.

We compare four different runs. These four runs are divided in two MNIST runs and two CIFAR-100 runs. The settings for the two MNIST run are equal, and similarly the settings for the two CIFAR-100 runs are equal. The settings can be found in Section A.3.4 for both MNIST and CIFAR-100. Both MNIST runs use the same convolutional neural network structure, but each with a different weight initialization for the convolutional layers. The same holds for the CIFAR-100 runs. The convolutional neural network structures are found in Tables A.12 and A.13 for MNIST and CIFAR-100 respectively. We see that the neural network complexity is higher for CIFAR-100.

5.5.1 Results

The top 5 subgroup descriptions and their quality measure values on depths one through three of both runs are listed in Tables A.6 and A.7 for MNIST and CIFAR-100 respectively. A cell colored green indicates this subgroup is also found at the same rank as the other run with different weight initialization. Similarly, a cell colored yellow indicates this subgroup is also found, but at a different rank within the top five. A cell colored red indicates this subgroup is not found by the other run within the top five. We see that the MNIST table has 6 green cells, 4 yellow cells and 20 red cells, while the CIFAR-100 table has 6 green cells, 14 yellow cells and 10 red cells.

5.5.2 Discussion

We see that the subgroups obtained for the first depth on the MNIST dataset are mostly similar, but the subgroups obtained on higher depths are all different with a single exception. The CIFAR-100 dataset subgroups differ much less for the two runs. On the first depth, it does perform slightly worse, but on higher depth it performs much better.

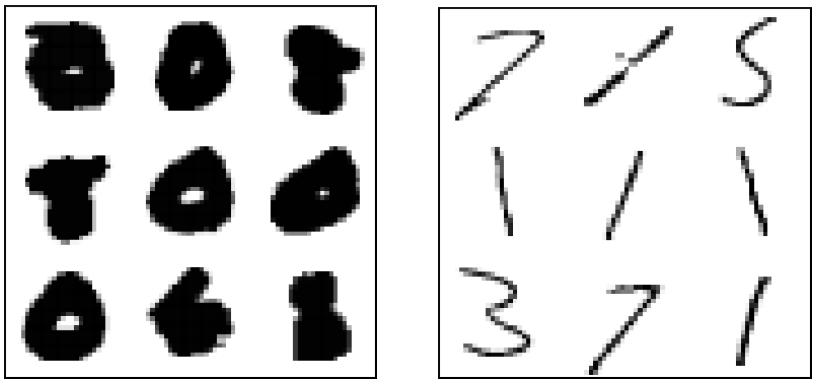
Ultimately, this experiment is very limited in the number of runs due to the running time of a single run. Therefore, these results are only meant to give an indication instead of a strict rule. Another point of skepticism in this experiment is that the CIFAR-100 experiment has a smaller search space than MNIST. The CIFAR-100 run has four numeric descriptors, whereas the MNIST run has 56 numeric descriptors.

In the end, this experiment should give a indication of whether it is useful to use multiple weight initializations. Also, as mentioned before, we can increase the number of subgroups returned in the results and more rigorously inspect them.

5.6 Second quality measure on MNIST

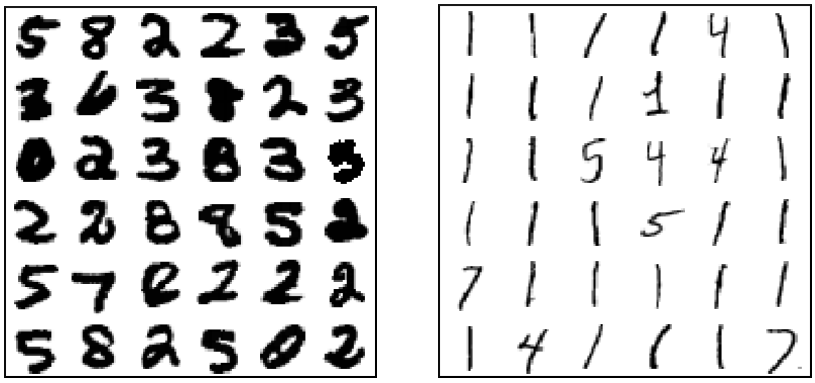
We conduct two EMM runs on the MNIST dataset with the second quality measure as described in Section 4.4. We are interested in subgroups that consist either of images with either many barely activated features or many abundantly activated features. Since one kind of these subgroups can drown out the other, we do two runs. In the first run, by setting the upper threshold value to infinity, we find only subgroups of the kind with barely activated features. Similarly in the second run, by setting the lower threshold value to minus infinity, we find only subgroups of the kind with abundantly activated features. The threshold values used can be found in Table A.14. The EMM parameters can be found in Section A.3.5.

The nine most exceptional features with these settings, for each set of threshold values, are shown in Figure 5.7. The features maps for the most exceptional image, for both sets of threshold values, are shown in Figure 5.9 for comparison.



(a) Most exceptionally low features (b) Most exceptionally high features

Figure 5.7: Most exceptional images for experiment in Section 5.6



(a) Rank 1 most exceptionally low features (b) Rank 1 most exceptionally high features

Figure 5.8: Most exceptional subgroup samples for experiment in Section 5.6

5.6.1 Results

The most exceptional subgroups with regards to images with the most barely activated features all include cluster 9 in their description. A sample of this cluster can be seen in Figure 5.2c. Furthermore, the descriptions also refine columns around the middle (columns 13 through 18) to have a high value. The top result is described by:

$$column15 \geq 12.1 \wedge column18 \geq 10.9 \wedge cluster = 9.0$$

A sample of this subgroup is shown in Figure 5.8a.

The most exceptional subgroups with regard to images with the most abundantly activated features all include cluster 8 in their description. A sample of this cluster can be seen in Figure 5.2b. The descriptions also tend to further refine a set of spatially related rows to have a low value. These rows include row 8, 9, 10, 15, 16 and 17. The top results is described by:

$$row17 \leq 2.0 \wedge row9 \leq 2.3 \wedge cluster = 8.0$$

A sample of this subgroup is shown in Figure 5.8b. The full lists of results can be found in Table A.8 and Table A.9.

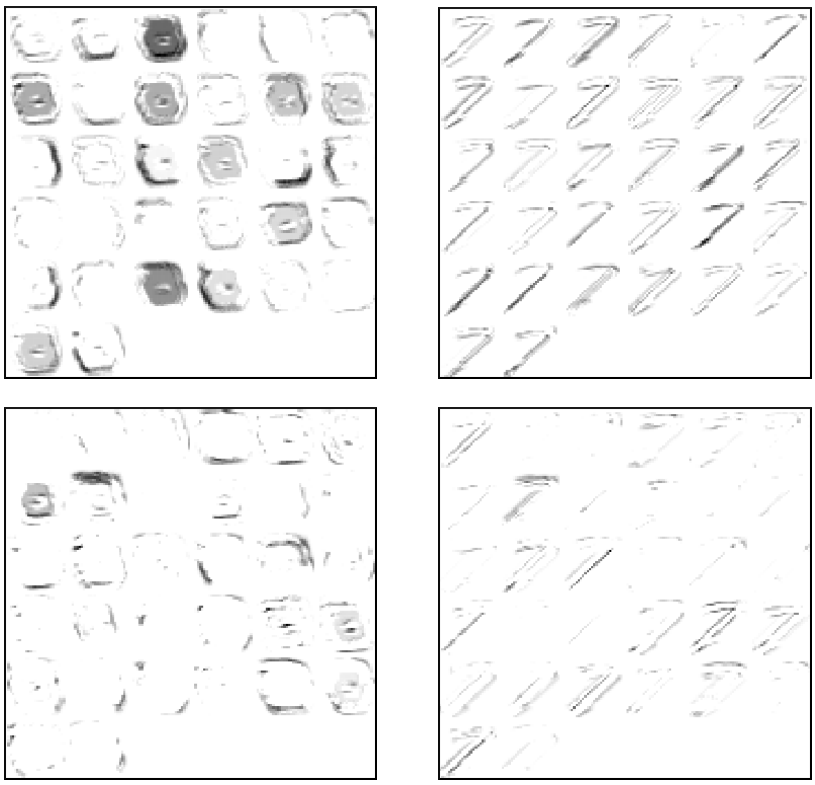


Figure 5.9: Feature maps of image with most exceptionally high activated features (left) and low activated features (right) for convolutional layers 0 (top) and 1 (bottom)

5.6.2 Discussion

The most exceptional images for the MNIST dataset seem to be either digits with a high font weight and much curves or digits with a low font weight and little to no curves. The former lead to the digits 0, 2, 3, 5 and 8. The latter lead mostly to the digit 1, and some 4, 5 and 7. The reason why the high font weight images have many barely activated features is a combination of two factors. The first factor is that many patterns do simply not activate that much on the entire image, they detect only a side edge of the digit, as seen in the feature maps. The second factor is that the high font weight image themselves have a high sum of their values. This leads to the fraction of feature sum divided by image sum being very low, thus not passing the lower threshold for a high number of features. Only for a few features the feature sum is actually proportionally high compared to the image sum.

Low font weight images on the other have many abundantly activated features because the opposite of the second reason above holds for them. Their image sum is very low, due to their low font weight. The features seemingly do not activate much less, but relatively to the image sum they do. Thus the fraction of feature sum divided by image sum is high, passing the upper threshold for a high number of features.

The most exceptional images found with these threshold values are also similar to the samples of the most exceptional subgroup. It is unfortunate however, that all high ranked subgroup descriptions on higher depths are too similar, resulting in basically being the same kind of images.

(a) Most exceptionally low features (b) Most exceptionally high features

Figure 5.10: Most exceptional images for experiment in Section 5.7

5.7 Second quality measure on CIFAR-100

We conduct two EMM runs on the CIFAR-100 dataset with the second quality measure, similar to the the previous experiment on MNIST. We use two sets of threshold values again, to find subgroups with the most barely activated features and the most abundantly activated features. The threshold values used can be found in Table A.15. The EMM parameters can be found in Section A.3.6.

The nine most exceptional features with these settings, for each set of threshold values, are shown in Figure 5.10. Some of the features maps for the most exceptional image, for both sets of threshold values, are shown in Figure 5.12 for comparison.

5.7.1 Results

The descriptors allow for many subgroup descriptions which end up with roughly the same selection of images. Therefore, only few complete descriptions are stated here. In Table A.10 and Table A.11 full lists of results can be found.

The subgroups with the most barely activated features tend to choose the low bins for each color. This results in a subgroup consisting of images which have low values and thus contain a lot of black. The most exceptional subgroup has the description:

$$red \geq 62.5 \wedge color \leq 284.7 \wedge green \leq 161.6 \wedge red \leq 284.7 \wedge blue \leq 299.9$$

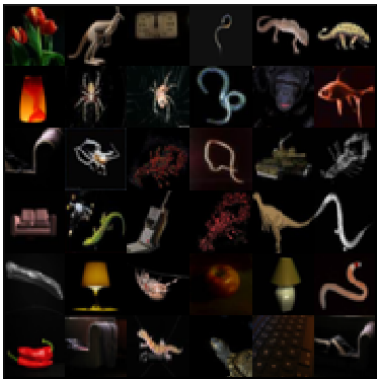
A sample of images of this subgroup is shown in Figure 5.11a. On depth 1, the beam of width 16 already contains mostly descriptions that place upper bound restrictions on the descriptors. However, there are some lower bound descriptions on rank 6, 9 through 13, 15 and 16. On depth 2, the beam is entirely filled with descriptions that place upper bound restrictions on the color descriptors.

The subgroups with the most abundantly activated features tend to choose the low bins for red, while choosing the high bins for blue. In general, upper bound restrictions are also placed on the amount of green and total color, though the lowest bins are not selected. Unsurprisingly, this results in subgroups consisting of mostly blue, occasionally blue-green images. These images mostly belong to the classes shark, dolphin, seal, whale and turtle. The most exceptional subgroup has the description:

$$red \leq 133.9 \wedge green \leq 483.9 \wedge color \leq 1800.9 \wedge blue \geq 598.1$$

A sample of images of this subgroup is shown in Figure 5.11b.

(a) Rank 1 most exceptionally low features



(b) Rank 1 most exceptionally high features

Figure 5.11: Most exceptional subgroup samples for experiment in Section 5.7

5.7.2 Discussion

If we look at the most exceptional images found in the dataset, we see that the images with barely activated features are images of a single color with some variance in saturation and value. For eight of the nine images this is the color blue, the other image is green. Now, what stands out is that these colors are a single channel of the red, blue and green channels. The effect of this in the feature maps is that we see fewer feature activating. As the images are also fully colored with no black, their image sum is high. This leads to the many barely activated features, divided by the image sum, passing the lower threshold.

The images with abundantly activated features are images of a single or few colors object on a mostly black background. Some objects only consist of one color, but what stands out is that these are all colors that are mostly not pure red, blue or green. The spider and two snakes are uniformly brown, orange and silver respectively. The effect of this in the feature maps is that we see that many features are activating. The black background means these pixels are all zero values, leading to a low image sum. This causes the fraction that we compare with the threshold value to be high, as the denominator stays low.

As with MNIST, we observe that the most exceptional subgroups are comprised of images that are similar to the most exceptional images. It is unfortunate that all high ranked subgroup descriptions on higher depths are too similar, resulting in basically being the same kind of images. For the barely activated features, you would imagine that there would also be subgroup descriptions with complete red or green backgrounds, such as the bottom right image of the left Figure 5.10. The reason why subgroup descriptions that attempt to refine to images of complete green or red backgrounds do not have as high of a quality measure value is because there are not enough of these images. Images with complete blue backgrounds are plentiful, due to the number of aquatic animal classes. Green or red background subgroup descriptions also include many images that do not have uniform colored backgrounds, and these images are not as exceptional.

The same problem occurs for the abundantly activated exceptional subgroups. All top ranked subgroups are minor variants of the first ranked subgroup. The black background gives these images an unfair advantage, since our quality measure divides by the image sum.

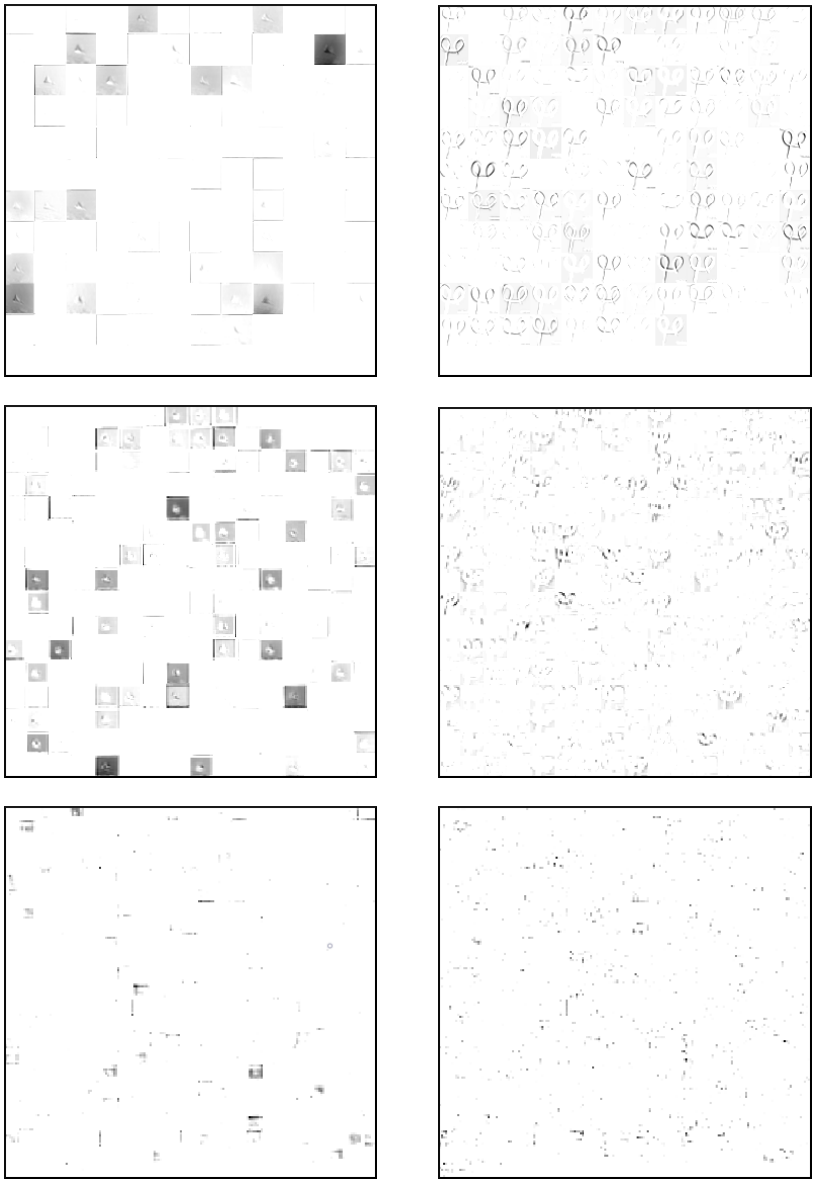


Figure 5.12: Feature maps of image with most exceptionally high activated features (left) and low activated features (right) for convolutional layers 0 (top), 2 (middle) and 5 (bottom)

Chapter 6

Discussion

The experimental results in the previous chapter show that both quality measures find exceptional subgroups of different natures. The first quality measure was proposed to find subgroups which train the neural network in an exceptional way compared to the entire dataset or complement, leading to a different set of patterns that the kernels try to find. It tends however to find subgroups whose loss function is most easily solved out of the considered subgroups. This causes the neural network to be optimized too fast, thus their weights change the least. This in turn leads to the highest convolutional layer weight distance, which is what the quality measure value is. Furthermore, the complexity of the loss function is highly dependable on the balance of the classes. Thus a subgroup consisting of only data of one class leads to a exceptionally high quality measure value. This questions whether training each neural network is even necessary to find the most exceptional subgroups of this quality measure. It might be possible to determine the exceptionality of a subgroup through the class distribution, to an extent. So while the subgroup found by this quality measure might still be interesting, they are not as promising as hoped. Rather, subgroups whose loss function would be exceptionally complex might be more interesting, as they could indicate erroneous predicts if their loss function can not be solved optimally.

As a neural network needs to be trained for each subgroup, this quality measure has a high runtime. Therefore we are limited on a number of settings depending on the available computing resources. The neural network complexity that can feasibly be used is limited. As more complex problems require more complex neural networks, the complexity of problems that can feasibly be explored is also limited. The number of subgroups that can be explored is also limited. Fortunately, the heuristic nature of EMM helps alleviate this problem. Another drawback is the randomization present in the neural network weight initialization. This leads to variance in obtained results. For this we proposed training multiple instances in Section 4.3.2, but this does not completely eliminate the drawback. The fairness of comparing two neural networks trained on two different subgroups is also questionable. The complexity of each subgroup's training cost function differs due to two aspects. First, their size is most likely different. For this we proposed using epochs relative to subgroup size in Section 4.3.1, but this does not negate their differing cost function complexity. Furthermore, adaptive learning rate optimizers still behave different on different training set sizes and thus subgroup size still impacts unfairness. The second aspect is the already mentioned class imbalance that occurs in subgroups.

The second quality measure was proposed to find subgroups whose feature maps have exceptionally high or low activated features compared to the entire dataset or complement. It tends to find subgroups according to this intention, but the total value of the input to the neural network is also a big factor. The most exceptional subgroups thus consist of either training data with exceptionally high activated features with a low total input, or training data with exceptionally low activated features with a high total input.

A major benefit over the first quality measure is that only a single neural network has to be trained. Thus the complexity of the neural network should be no limitation. Another benefit in comparison to the first quality measure is the interpretability of its results. By visualizing the

feature maps, we can find the underlying reason for the exceptionally high quality measure value. The threshold values that are needed are a drawback. They are determined from the dataset values itself. For now they are manually determined as well, further questioning the validity of results, as it makes them guesswork and prone to be fine tuned to obtain wanted results.

Overall, the first quality measure has numerous more drawbacks than the second. Therefore the second quality measure seems preferable. It might still be worth it to also use the first quality measure to find different exceptional subgroups, if the runtime is feasible.

Chapter 7

Conclusions and future research

7.1 Conclusions

In this thesis we have introduced a method to find interesting subgroups in training datasets of convolutional neural networks. The method uses exceptional model mining (EMM) [9] to fulfill this goal, by introducing a new model class. A model class requires target variables and a quality measure over these target variables to quantify their exceptionality.

Our model class differs from earlier proposed model classes as our target variables are not contained directly in the dataset. Rather, we use the dataset to train a neural network of which we obtain target variables. In this thesis we have introduced two approaches with different target variables. Our first approach uses the weights of the kernels of convolutional layers as target variables, as each subgroup trains the neural network differently. Our second approach uses the output of convolutional layers as target variables, as each subgroup gives different output.

Next we introduced a quality measure for each approach. For the first approach we use the distance between the weights of kernels of convolutional layers. This quality measure finds subgroups which learn different features, but its results were impacted by class imbalance and limited in neural network complexity due to the required training time for each subgroup. We can see results on the MNIST dataset in Figures 5.1 and 5.3, and the results on the CIFAR-100 dataset in Figures 5.4, 5.5 and 5.6. For the second approach we use the number of convolutional layer neurons whose output is exceptionally low or high. This quality measure finds subgroups which are either poorly or well represented by the features learned by the neural network, but the subgroups also seem impacted by the total sum of input. We can see results on the MNIST dataset in Figure 5.8 and the results on the CIFAR-100 dataset in Figure 5.11.

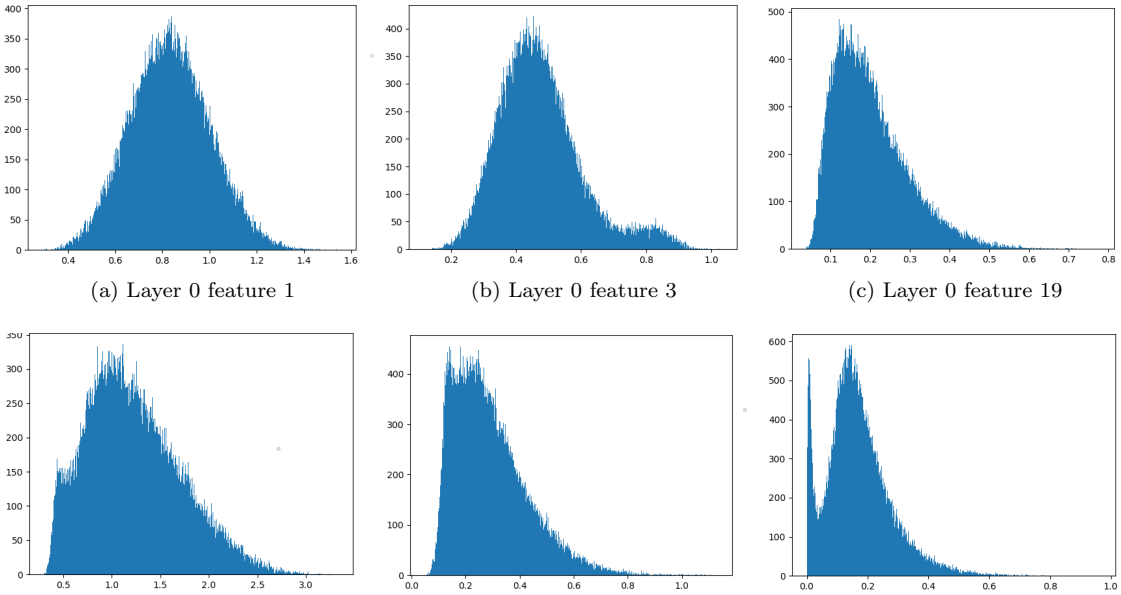
With this model class and these quality measures, we have introduced a method to employ EMM on training datasets of convolutional neural networks, fulfilling our research question.

7.2 Future research

The model class introduced in this thesis is limited to convolutional neural networks, as they are based upon convolutional layers. A logical next step would be to investigate other models suited for other specific neural networks, such as long short-term networks, or perhaps a more general model class suited for any neural network.

Other domains besides images and other tasks beside classification that use convolutional neural networks can also be explored with the introduced model class. A study could be done for its use in the domains mentioned in 2.2. Perhaps some of these are not directly compliant with the model class. For these domains or tasks slight modifications to this model class can be researched to make it compliant.

For the second quality measure we saw that threshold values are needed. During the research for this thesis, manually set threshold values were used. As this requires additional labor and



(a) Layer 0 feature 1 (b) Layer 0 feature 3 (c) Layer 0 feature 19

(d) Layer 1 feature 7 (e) Layer 1 feature 29 (f) Layer 1 feature 9

Figure 7.1: MNIST distributions of the threshold comparison values of all images of a chosen feature and layer.

leaves room for tinkering of results, it would be ideal to determine these thresholds dynamically. The only seemingly available values to base the thresholds on are the obtained comparison values during a run themselves. A feature of an image is deemed exceptional if the total value of its feature map divided by the total value of the image surpasses the lower or upper threshold. The distribution of this comparison value for all images per feature differs, this is shown in figure 7.1. For this figure, the MNIST dataset is used to train the CNN described in section A.12. The described comparison value is obtained for all images of six different features and shown in a histogram. If such distributions represent a probability distribution, those could be used to determine what an extreme value would be. For example, the top left figure seems to represent a Gaussian distribution. The lower and upper threshold value could then be $\mu - 2\sigma$ and $\mu + 2\sigma$ respectively, which would result in 5% of the values being deemed exceptional. We see in the figure that there is not a single fitting distribution, thus a complex solution is needed to determine the probability distributions.

Bibliography

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[4] Stephen D Bay and Michael J Pazzani. Detecting group differences: Mining contrast sets. *Data mining and knowledge discovery*, 5(3):213–246, 2001.

[5] Hong Cheng, Xifeng Yan, and Jiawei Han. Mining graph patterns. In *Frequent pattern mining*, pages 307–338. Springer, 2014.

[6] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 43–52. Citeseer, 1999.

[7] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[8] Lennart Downar and Wouter Duivesteijn. Exceptionally monotone modelsthe rank correlation model class for exceptional model mining. *Knowledge and Information Systems*, 51(2):369–394, 2017.

[9] Wouter Duivesteijn, Ad J Feelders, and Arno Knobbe. Exceptional model mining. *Data Mining and Knowledge Discovery*, 30(1):47–98, 2016.

[10] Wouter Duivesteijn, Arno Knobbe, Ad Feelders, and Matthijs van Leeuwen. Subgroup discovery meets bayesian networks–an exceptional model mining approach. In *2010 IEEE International Conference on Data Mining*, pages 158–167. IEEE, 2010.

[11] Hongjian Fan and Kotagiri Ramamohanarao. Efficiently mining interesting emerging patterns. In *International Conference on Web-Age Information Management*, pages 189–201. Springer, 2003.

[12] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[13] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.

[14] David J Hand. Pattern detection and discovery. In *Pattern Detection and Discovery*, pages 1–12. Springer, 2002.

[15] Franciso Herrera, Cristóbal José Carmona, Pedro González, and María José Del Jesus. An overview on subgroup discovery: foundations and applications. *Knowledge and information systems*, 29(3):495–525, 2011.

[16] Sibylle Hess, Wouter Duivesteijn, Philipp Honysz, and Katharina Morik. The spectacl of nonconvex clustering: A spectral approach to density-based clustering. 2019.

[17] Mehdi Kaytoue, Marc Plantevit, Albrecht Zimmermann, Anes Bendimerad, and Céline Robardet. Exceptional contextual subgraph mining. *Machine Learning*, 106(8):1171–1211, 2017.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Dennis Leman, Ad Feelders, and Arno Knobbe. Exceptional model mining. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 1–16. Springer, 2008.

[22] Florian Lemmerich, Martin Becker, and Martin Atzmueller. Generic pattern trees for exhaustive exceptional model mining. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 277–292. Springer, 2012.

[23] Florian Lemmerich, Martin Becker, Philipp Singer, Denis Helic, Andreas Hotho, and Markus Strohmaier. Mining subgroups with exceptional transition behavior. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 965–974. ACM, 2016.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

[25] Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE, 2003.

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

[27] Matthew Stewart. Simple introduction to convolutional neural networks. `https://www.towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac/`, 2019 (accessed March 11, 2019).

[28] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3476–3483, 2013.

[29] Ujjwalkarn. An intuitive explanation of convolutional neural networks. `https://www.ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/`, 2016 (accessed March 11, 2019).

[30] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 78–87. Springer, 1997.

[31] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[32] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[33] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge & Data Engineering*, (1):63–77, 2006.

Appendix A

Experiment setups and results

A.1 Results

	Description	QM value
Depth 1		
Rank 1	cluster = 3.0	28.5
Rank 2	cluster = 8.0	28.0
Rank 3	cluster = 6.0	24.2
Rank 4	cluster = 4.0	23.4
Rank 5	cluster = 9.0	22.4
Depth 2		
Rank 1	row23 $\leq$ 2.0 $\wedge$ cluster = 8.0	32.7
Rank 2	column24 $\geq$ 6.1 $\wedge$ cluster = 3.0	32.4
Rank 3	column8 $\geq$ 10.5 $\wedge$ cluster = 3.0	32.4
Rank 4	row19 $\leq$ 2.1 $\wedge$ cluster = 8.0	32.3
Rank 5	column14 $\leq$ 6.9 $\wedge$ cluster = 8.0	32.3
Depth 3		
Rank 1	row3 $\geq$ 0.0 $\wedge$ row23 $\leq$ 2.0 $\wedge$ cluster = 8.0	32.9
Rank 2	row1 $\leq$ 0.0 $\wedge$ row23 $\leq$ 2.0 $\wedge$ cluster = 8.0	32.9
Rank 3	row22 $\leq$ 1.6 $\wedge$ row23 $\leq$ 2.0 $\wedge$ cluster = 8.0	32.8
Rank 4	row27 $\leq$ 0.0 $\wedge$ row23 $\leq$ 2.0 $\wedge$ cluster = 8.0	32.8
Rank 5	row6 $\leq$ 1.5 $\wedge$ row23 $\leq$ 2.0 $\wedge$ cluster = 8.0	32.8

Table A.1: Experiment results of Section 5.2

	Description	QM value
Depth 1		
Rank 1	row3 $\leq$ 0.0	33.2
Rank 2	cluster = 3.0	32.0
Rank 3	cluster $\neq$ 3.0	31.8
Rank 4	cluster = 8.0	29.1
Rank 5	cluster $\neq$ 8.0	28.9
Depth 2		
Rank 1	column8 $\geq$ 10.5 $\wedge$ cluster = 3.0	36.2
Rank 2	column19 $\leq$ 6.6 $\wedge$ cluster = 3.0	35.9
Rank 3	column7 $\geq$ 8.6 $\wedge$ cluster = 3.0	34.9
Rank 4	column24 $\geq$ 6.1 $\wedge$ cluster = 3.0	34.8
Rank 5	column9 $\geq$ 11.4 $\wedge$ cluster = 3.0	34.3
Depth 3		
Rank 1	column8 $\geq$ 10.5 $\wedge$ cluster = 3.0	36.2
Rank 2	row5 $\geq$ 0.0 $\wedge$ column8 $\geq$ 10.5 $\wedge$ cluster = 3.0	36.2
Rank 3	column27 $\geq$ 0.0 $\wedge$ column8 $\geq$ 10.5 $\wedge$ cluster = 3.0	36.2
Rank 4	row27 $\leq$ 0.0 $\wedge$ column8 $\geq$ 10.5 $\wedge$ cluster = 3.0	36.2
Rank 5	row3 $\geq$ 0.0 $\wedge$ column8 $\geq$ 10.5 $\wedge$ cluster = 3.0	36.2

Table A.2: Complement experiment results of Section 5.2

	Description	QM value
Depth 1		
Rank 1	color $\geq$ 1600.5	106.5
Rank 2	blue $\leq$ 366.3	105.7
Rank 3	green $\leq$ 432.5	105.4
Rank 4	green $\geq$ 546.5	105.2
Rank 5	color $\leq$ 1263.3	104.3
Depth 2		
Rank 1	blue $\geq$ 674.3 $\wedge$ color $\geq$ 1600.5	126.8
Rank 2	red $\geq$ 469.1 $\wedge$ blue $\leq$ 366.3	126.2
Rank 3	color $\geq$ 1176.6 $\wedge$ blue $\leq$ 366.3	126.0
Rank 4	green $\geq$ 689.3 $\wedge$ color $\geq$ 1600.5	125.5
Rank 5	green $\geq$ 411.3 $\wedge$ blue $\leq$ 366.3	125.4
Depth 3		
Rank 1	blue $\leq$ 728.3 $\wedge$ blue $\geq$ 674.3 $\wedge$ color $\geq$ 1600.5	139.9
Rank 2	color $\leq$ 2171.5 $\wedge$ blue $\geq$ 674.3 $\wedge$ color $\geq$ 1600.5	137.2
Rank 3	color $\geq$ 1348.4 $\wedge$ red $\geq$ 469.1 $\wedge$ blue $\leq$ 366.3	137.0
Rank 4	green $\leq$ 430.3 $\wedge$ color $\geq$ 1176.6 $\wedge$ blue $\leq$ 366.3	137.0
Rank 5	green $\geq$ 810.7 $\wedge$ blue $\geq$ 674.3 $\wedge$ color $\geq$ 1600.5	136.4

Table A.3: Experiment results of Section 5.3

	Description	QM value
Depth 1		
Rank 1	blue $\leq$ 366.3	104.3
Rank 2	blue $\geq$ 366.3	102.2
Rank 3	blue $\geq$ 516.4	101.4
Rank 4	color $\leq$ 1600.5	100.7
Rank 5	blue $\leq$ 516.4	98.9
Depth 2		
Rank 1	color $\geq$ 2038.1 $\wedge$ blue $\geq$ 516.4	112.0
Rank 2	green $\geq$ 684.0 $\wedge$ blue $\geq$ 516.4	110.8
Rank 3	blue $\leq$ 581.7 $\wedge$ blue $\geq$ 516.4	109.1
Rank 4	color $\leq$ 2038.1 $\wedge$ blue $\geq$ 516.4	108.9
Rank 5	red $\geq$ 705.3 $\wedge$ blue $\geq$ 516.4	108.8
Depth 3		
Rank 1	red $\geq$ 847.1 $\wedge$ green $\geq$ 684.0 $\wedge$ blue $\geq$ 516.4	123.6
Rank 2	red $\geq$ 849.5 $\wedge$ color $\geq$ 2038.1 $\wedge$ blue $\geq$ 516.4	120.3
Rank 3	blue $\leq$ 710.7 $\wedge$ color $\geq$ 2038.1 $\wedge$ blue $\geq$ 516.4	117.3
Rank 4	color $\leq$ 1598.9 $\wedge$ blue $\leq$ 581.7 $\wedge$ blue $\geq$ 516.4	117.0
Rank 5	red $\leq$ 758.8 $\wedge$ green $\geq$ 684.0 $\wedge$ blue $\geq$ 516.4	117.0

Table A.4: Complement experiment results of Section 5.3

	First layer	All layers	All layers normalized
Rank 1	column9 $\leq$ 1.56	column9 $\leq$ 1.56	column9 $\leq$ 1.56
Rank 2	column20 $\leq$ 2.65	row15 $\geq$ 7.87	row15 $\geq$ 7.87
Rank 3	row15 $\geq$ 7.87	row16 $\geq$ 7.97	column20 $\leq$ 2.65
Rank 4	row16 $\geq$ 7.97	**row9 $\leq$ 3.16**	row16 $\geq$ 7.97
Rank 5	column21 $\leq$ 0.63	row23 $\geq$ 6.83	row23 $\geq$ 6.83
Rank 6	row23 $\geq$ 6.83	row21 $\geq$ 7.47	column21 $\leq$ 0.63
Rank 7	column8 $\leq$ 0.0	row5 $\geq$ 2.62	row5 $\geq$ 2.62
Rank 8	row5 $\geq$ 2.62	row24 $\leq$ 1.46	row21 $\geq$ 7.47
Rank 9	row21 $\geq$ 7.47	row21 $\leq$ 2.77	column8 $\leq$ 0.0
Rank 10	row24 $\leq$ 1.46	**row10 $\leq$ 3.05**	row24 $\leq$ 1.46
Rank 11	row21 $\leq$ 2.77	column20 $\leq$ 2.65	row21 $\leq$ 2.77
Rank 12	**row10 $\leq$ 3.05**	row22 $\geq$ 7.45	**row10 $\leq$ 3.05**
Rank 13	row17 $\geq$ 7.52	column8 $\leq$ 0.0	row17 $\geq$ 7.52
Rank 14	row6 $\leq$ 0.0	column21 $\leq$ 0.63	row6 $\leq$ 0.0
Rank 15	**row9 $\leq$ 3.16**	row17 $\geq$ 7.52	**row9 $\leq$ 3.16**
Rank 16	row22 $\geq$ 7.45	row6 $\leq$ 0.0	row22 $\geq$ 7.45
Rank 17	row24 $\geq$ 4.78	row24 $\geq$ 4.78	row24 $\geq$ 4.78
Rank 18	row20 $\geq$ 6.78	row14 $\geq$ 7.38	row20 $\geq$ 6.78
Rank 19	row14 $\geq$ 7.38	column19 $\leq$ 4.49	row14 $\geq$ 7.38
Rank 20	column19 $\leq$ 4.49	row20 $\geq$ 6.78	column19 $\leq$ 4.49

Table A.5: Experiment results of Section 5.4

	1st seed		2nd seed	
	Description	QM	Description	QM
Depth 1				
Rank 1	cluster = 8.0	29.9	cluster = 3.0	28.9
Rank 2	cluster = 3.0	28.1	cluster = 8.0	25.6
Rank 3	cluster = 6.0	24.7	cluster = 6.0	24.7
Rank 4	cluster = 4.0	24.2	cluster = 4.0	22.7
Rank 5	cluster = 9.0	23.2	row21 $\leq$ 3.1	21.9
Depth 2				
Rank 1	column14 $\leq$ 6.9 $\wedge$ cluster = 8.0	35.1	row20 $\leq$ 2.1 $\wedge$ cluster = 8.0	32.2
Rank 2	row8 $\leq$ 2.3 $\wedge$ cluster = 8.0	34.3	row22 $\leq$ 2.1 $\wedge$ cluster = 8.0	31.9
Rank 3	row10 $\leq$ 2.3 $\wedge$ cluster = 8.0	34.2	row14 $\leq$ 2.4 $\wedge$ cluster = 8.0	31.7
Rank 4	row17 $\leq$ 2.3 $\wedge$ cluster = 8.0	34.2	row9 $\leq$ 2.3 $\wedge$ cluster = 8.0	31.7
Rank 5	row7 $\leq$ 2.0 $\wedge$ cluster = 8.0	34.1	row7 $\leq$ 2.0 $\wedge$ cluster = 8.0	31.5
Depth 3				
Rank 1	row6 $\leq$ 0.0 $\wedge$ column14 $\leq$ 6.9 $\wedge$ cluster = 8.0	37.2	row9 $\leq$ 2.1 $\wedge$ row22 $\leq$ 2.1 $\wedge$ cluster = 8.0	33.6
Rank 2	column16 $\leq$ 5.8 $\wedge$ column14 $\leq$ 6.9 $\wedge$ cluster = 8.0	37.0	column14 $\geq$ 4.9 $\wedge$ row22 $\leq$ 2.1 $\wedge$ cluster = 8.0	33.4
Rank 3	row15 $\leq$ 3.8 $\wedge$ column14 $\leq$ 6.9 $\wedge$ cluster = 8.0	36.9	row16 $\geq$ 2.0 $\wedge$ row22 $\leq$ 2.1 $\wedge$ cluster = 8.0	33.4
Rank 4	row17 $\leq$ 3.0 $\wedge$ column14 $\leq$ 6.9 $\wedge$ cluster = 8.0	36.9	row25 $\leq$ 1.3 $\wedge$ row14 $\leq$ 2.4 $\wedge$ cluster = 8.0	33.1
Rank 5	row14 $\leq$ 3.5 $\wedge$ column14 $\leq$ 6.9 $\wedge$ cluster = 8.0	36.6	row9 $\leq$ 3.4 $\wedge$ row20 $\leq$ 2.1 $\wedge$ cluster = 8.0	33.0

Table A.6: MNIST experiment results of Section 5.5

	1st seed		2nd seed	
	Description	QM	Description	QM
Depth 1				
Rank 1	green $\leq$ 432.5	106.3	blue $\leq$ 366.3	100.7
Rank 2	blue $\leq$ 366.3	104.8	color $\leq$ 1263.3	99.3
Rank 3	color $\leq$ 1263.3	104.6	green $\leq$ 432.5	98.7
Rank 4	red $\leq$ 445.5	102.0	green $\geq$ 546.5	96.9
Rank 5	blue $\geq$ 516.4	100.2	red $\leq$ 445.5	96.2
Depth 2				
Rank 1	green $\geq$ 411.3 $\wedge$ blue $\leq$ 366.3	117.5	color $\geq$ 1151.2 $\wedge$ green $\leq$ 432.5	115.2
Rank 2	blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	115.81	blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	114.9
Rank 3	blue $\geq$ 333.3 $\wedge$ color $\leq$ 1263.3	115.51	green $\geq$ 411.3 $\wedge$ blue $\leq$ 366.3	114.7
Rank 4	red $\leq$ 339.5 $\wedge$ color $\leq$ 1263.3	114.4	red $\geq$ 469.1 $\wedge$ blue $\leq$ 366.3	114.1
Rank 5	blue $\geq$ 312.4 $\wedge$ blue $\leq$ 366.3	113.96	red $\geq$ 417.3 $\wedge$ color $\leq$ 1263.3	113.8
Depth 3				
Rank 1	red $\geq$ 417.0 $\wedge$ blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	131.7	red $\geq$ 417.0 $\wedge$ blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	126.9
Rank 2	green $\geq$ 335.2 $\wedge$ blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	129.3	color $\geq$ 1341.7 $\wedge$ green $\geq$ 411.3 $\wedge$ blue $\leq$ 366.3	126.7
Rank 3	green $\leq$ 248.4 $\wedge$ blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	128.1	red $\geq$ 556.6 $\wedge$ green $\geq$ 411.3 $\wedge$ blue $\leq$ 366.3	126.0
Rank 4	color $\geq$ 1341.7 $\wedge$ green $\geq$ 411.3 $\wedge$ blue $\leq$ 366.3	125.4	green $\geq$ 335.2 $\wedge$ blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	125.6
Rank 5	color $\geq$ 950.4 $\wedge$ blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	125.2	color $\geq$ 950.4 $\wedge$ blue $\leq$ 249.5 $\wedge$ color $\leq$ 1263.3	125.0

Table A.7: CIFAR-100 experiment results of Section 5.5

	Description	QM value
Depth 1		
Rank 1	cluster = 9.0	6.00
Rank 2	column15 $\geq$ 9.8	5.4
Rank 3	column14 $\geq$ 9.5	5.3
Rank 4	column16 $\geq$ 10.0	5.3
Rank 5	column13 $\geq$ 8.9	5.0
Depth 2		
Rank 1	column17 $\geq$ 10.5 $\wedge$ cluster = 9.0	8.4
Rank 2	column16 $\geq$ 10.3 $\wedge$ cluster = 9.0	8.4
Rank 3	column15 $\geq$ 10.3 $\wedge$ cluster = 9.0	8.3
Rank 4	column14 $\geq$ 10.3 $\wedge$ cluster = 9.0	8.3
Rank 5	cluster = 9.0 $\wedge$ column16 $\geq$ 10.0	8.3
Depth 3		
Rank 1	column15 $\geq$ 12.1 $\wedge$ column18 $\geq$ 10.9 $\wedge$ cluster = 9.0	10.4
Rank 2	column14 $\geq$ 12.0 $\wedge$ column18 $\geq$ 10.9 $\wedge$ cluster = 9.0	10.3
Rank 3	column13 $\geq$ 12.3 $\wedge$ column17 $\geq$ 10.5 $\wedge$ cluster = 9.0	10.3
Rank 4	column15 $\geq$ 12.7 $\wedge$ column17 $\geq$ 10.5 $\wedge$ cluster = 9.0	10.3
Rank 5	column14 $\geq$ 12.5 $\wedge$ column17 $\geq$ 10.5 $\wedge$ cluster = 9.0	10.3

Table A.8: Experiment results of Section 5.6 for left of table A.14

	Description	QM value
Depth 1		
Rank 1	cluster = 9.0	3.8
Rank 2	row18 $\leq$ 3.2	3.5
Rank 3	row17 $\leq$ 3.5	3.5
Rank 4	row12 $\leq$ 3.5	3.5
Rank 5	row16 $\leq$ 3.8	3.4
Depth 2		
Rank 1	row13 $\leq$ 2.4 $\wedge$ cluster = 8.0	6.3
Rank 2	row14 $\leq$ 2.4 $\wedge$ cluster = 8.0	6.3
Rank 3	row12 $\leq$ 2.3 $\wedge$ cluster = 8.0	6.2
Rank 4	row15 $\leq$ 2.4 $\wedge$ cluster = 8.0	6.2
Rank 5	row11 $\leq$ 2.3 $\wedge$ cluster = 8.0	6.2
Depth 3		
Rank 1	row17 $\leq$ 2.0 $\wedge$ row9 $\leq$ 2.3 $\wedge$ cluster = 8.0	10.4
Rank 2	row16 $\leq$ 2.1 $\wedge$ row9 $\leq$ 2.3 $\wedge$ cluster = 8.0	10.3
Rank 3	row9 $\leq$ 2.0 $\wedge$ row16 $\leq$ 2.4 $\wedge$ cluster = 8.0	10.3
Rank 4	row8 $\leq$ 2.0 $\wedge$ row16 $\leq$ 2.4 $\wedge$ cluster = 8.0	10.3
Rank 5	row17 $\leq$ 2.0 $\wedge$ row10 $\leq$ 2.3 $\wedge$ cluster = 8.0	10.3

Table A.9: Experiment results of Section 5.6 for right of table A.14

	Description	QM value
Depth 1		
Rank 1	blue $\geq$ 598.1	397.2
Rank 2	green $\geq$ 612.5	375.9
Rank 3	color $\geq$ 1804.3	370.6
Rank 4	red $\geq$ 649.7	368.8
Rank 5	blue $\geq$ 483.4	368.8
Depth 2		
Rank 1	red $\leq$ 538.6 $\wedge$ blue $\geq$ 598.1	500.8
Rank 2	blue $\geq$ 451.4 $\wedge$ red $\leq$ 385.8	497.4
Rank 3	color $\leq$ 1800.9 $\wedge$ blue $\geq$ 598.1	479.2
Rank 4	green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	465.3
Rank 5	green $\geq$ 421.7 $\wedge$ red $\leq$ 385.8	459.3
Depth 3		
Rank 1	red $\leq$ 284.3 $\wedge$ color $\leq$ 1800.9 $\wedge$ blue $\geq$ 598.1	652.1
Rank 2	blue $\geq$ 672.9 $\wedge$ color $\geq$ 1169.4 $\wedge$ red $\leq$ 385.8	625.0
Rank 3	red $\leq$ 316.3 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	618.8
Rank 4	blue $\geq$ 656.8 $\wedge$ green $\geq$ 421.7 $\wedge$ red $\leq$ 385.8	608.2
Rank 5	color $\leq$ 1470.2 $\wedge$ red $\leq$ 538.6 $\wedge$ blue $\geq$ 598.1	606.6
Depth 4		
Rank 1	red $\leq$ 98.9 $\wedge$ color $\leq$ 1479.6 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	792.2
Rank 2	red $\leq$ 133.9 $\wedge$ green $\leq$ 483.7 $\wedge$ color $\leq$ 1800.9 $\wedge$ blue $\geq$ 598.1	763.7
Rank 3	green $\geq$ 593.5 $\wedge$ red $\leq$ 284.3 $\wedge$ color $\leq$ 1800.9 $\wedge$ blue $\geq$ 598.1	724.4
Rank 4	color $\leq$ 1196.4 $\wedge$ red $\leq$ 316.3 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	712.2
Rank 5	red $\leq$ 185.1 $\wedge$ color $\leq$ 1479.6 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	712.2
Depth 5		
Rank 1	color $\geq$ 1017.8 $\wedge$ red $\leq$ 98.9 $\wedge$ color $\leq$ 1479.6 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	797.4
Rank 2	green $\geq$ 300.1 $\wedge$ red $\leq$ 98.9 $\wedge$ color $\leq$ 1479.6 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	794.9
Rank 3	red $\leq$ 98.9 $\wedge$ color $\leq$ 1479.6 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	792.2
Rank 4	blue $\leq$ 780.0 $\wedge$ red $\leq$ 98.9 $\wedge$ color $\leq$ 1479.6 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	786.2
Rank 5	red $\geq$ 27.9 $\wedge$ red $\leq$ 98.9 $\wedge$ color $\leq$ 1479.6 $\wedge$ green $\leq$ 602.6 $\wedge$ blue $\geq$ 598.1	767.2

Table A.10: Experiment results of Section 5.7 for left of table A.15

	Description	QM value
Depth 1		
Rank 1	green $\leq$ 376.6	225.1
Rank 2	color $\leq$ 1107.0	225.0
Rank 3	blue $\leq$ 299.9	219.8
Rank 4	red $\leq$ 385.8	203.0
Rank 5	color $\leq$ 1330.4	174.0
Depth 2		
Rank 1	green $\leq$ 249.5 $\wedge$ blue $\leq$ 299.9	369.8
Rank 2	green $\leq$ 245.8 $\wedge$ color $\leq$ 1107.0	369.6
Rank 3	color $\leq$ 752.3 $\wedge$ green $\leq$ 376.6	367.6
Rank 4	color $\leq$ 758.2 $\wedge$ blue $\leq$ 299.9	367.4
Rank 5	color $\leq$ 759.6 $\wedge$ red $\leq$ 385.8	362.7
Depth 3		
Rank 1	green $\leq$ 157.9 $\wedge$ blue $\leq$ 206.9 $\wedge$ red $\leq$ 385.8	542.9
Rank 2	color $\leq$ 488.2 $\wedge$ blue $\leq$ 182.2 $\wedge$ green $\leq$ 376.6	538.3
Rank 3	color $\leq$ 484.9 $\wedge$ green $\leq$ 249.5 $\wedge$ blue $\leq$ 299.9	538.1
Rank 4	green $\leq$ 161.6 $\wedge$ red $\leq$ 284.7 $\wedge$ blue $\leq$ 299.9	537.7
Rank 5	color $\leq$ 486.6 $\wedge$ blue $\leq$ 206.9 $\wedge$ red $\leq$ 385.8	537.7
Depth 4		
Rank 1	color $\leq$ 285.4 $\wedge$ green $\leq$ 157.9 $\wedge$ blue $\leq$ 206.9 $\wedge$ red $\leq$ 385.8	673.1
Rank 2	color $\leq$ 284.7 $\wedge$ green $\leq$ 161.6 $\wedge$ red $\leq$ 284.7 $\wedge$ blue $\leq$ 299.9	671.4
Rank 3	color $\leq$ 285.4 $\wedge$ blue $\leq$ 116.1 $\wedge$ green $\leq$ 257.9 $\wedge$ red $\leq$ 385.8	669.8
Rank 4	color $\leq$ 284.2 $\wedge$ blue $\leq$ 130.5 $\wedge$ red $\leq$ 261.4 $\wedge$ green $\leq$ 376.6	667.1
Rank 5	green $\leq$ 87.0 $\wedge$ blue $\leq$ 131.0 $\wedge$ red $\leq$ 256.0 $\wedge$ color $\leq$ 1107.0	659.4
Depth 5		
Rank 1	red $\geq$ 62.5 $\wedge$ color $\leq$ 284.7 $\wedge$ green $\leq$ 161.6 $\wedge$ red $\leq$ 284.7 $\wedge$ blue $\leq$ 299.9	675.1
Rank 2	red $\geq$ 61.3 $\wedge$ color $\leq$ 284.2 $\wedge$ blue $\leq$ 130.5 $\wedge$ red $\leq$ 261.4 $\wedge$ green $\leq$ 376.6	674.2
Rank 3	color $\leq$ 285.4 $\wedge$ green $\leq$ 157.9 $\wedge$ blue $\leq$ 206.9 $\wedge$ red $\leq$ 385.8	673.1
Rank 4	red $\geq$ 62.8 $\wedge$ color $\leq$ 285.4 $\wedge$ blue $\leq$ 116.0 $\wedge$ green $\leq$ 257.9 $\wedge$ red $\leq$ 385.8	672.4
Rank 5	red $\geq$ 62.8 $\wedge$ color $\leq$ 285.4 $\wedge$ green $\leq$ 157.9 $\wedge$ blue $\leq$ 206.9 $\wedge$ red $\leq$ 385.8	672.4

Table A.11: Experiment results of Section 5.7 for right of table A.15

A.2 Neural network architectures

Conv2D(filters=32, kernel_size=(5, 5), activation=relu, use_bias=False, padding='same', kernel_initializer=he_uniform)
Conv2D(filters=32, kernel_size=(3, 3), activation=relu, use_bias=False, padding='same', kernel_initializer=he_uniform)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.25)
Flatten()
Dense(units=128, activation=relu)
Dropout(0.5)
Dense(units=n_classes, activation=softmax)

Table A.12: Neural network architecture variant one

Conv2D(filters=128, kernel_size=(3, 3), activation=relu, padding='same', kernel_initializer=he_uniform)
Conv2D(filters=128, kernel_size=(3, 3), activation=relu, padding='same', kernel_initializer=he_uniform)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.25)
Conv2D(filters=256, kernel_size=(3, 3), activation=relu, padding='same', kernel_initializer=he_uniform)
Conv2D(filters=256, kernel_size=(3, 3), activation=relu, padding='same', kernel_initializer=he_uniform)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.25)
Conv2D(filters=512, kernel_size=(3, 3), activation=relu, padding='same', kernel_initializer=he_uniform)
Conv2D(filters=512, kernel_size=(3, 3), activation=relu, padding='same', kernel_initializer=he_uniform)
MaxPooling2D(pool_size=(2, 2))
Dropout(0.25)
Flatten()
Dense(units=1024, activation=relu)
Dropout(0.5)
Dense(units=n_classes, activation=softmax)

Table A.13: Neural network architecture variant two

A.3 EMM configurations

A.3.1 Experiment first quality measure MNIST

- Comparison mode: whole dataset / complement
- Beam width: 3
- Beam depth: 3
- Number of bins: 3

- Dynamic bins: Yes
- Minimum subgroup size: 100 / 1000
- Epochs: 3
- Batch size: 128
- Number of models: 3

A.3.2 Experiment first quality measure CIFAR-100

- Comparison mode: whole dataset / complement
- Beam width: 3
- Beam depth: 3
- Number of bins: 3
- Dynamic bins: Yes
- Minimum subgroup size: 100 / 1000
- Epochs: 20
- Batch size: 128
- Number of models: 1

A.3.3 Experiment convolutional layer selection

- Comparison mode: whole dataset
- Beam width: 8
- Beam depth: 1
- Number of bins: 4
- Dynamic bins: Yes
- Minimum subgroup size: 10000
- Epochs: 3
- Batch size: 128
- Number of models: 3

A.3.4 Experiment number of neural networks selection

MNIST:

- Comparison mode: whole dataset
- Beam width: 3
- Beam depth: 3
- Number of bins: 3
- Dynamic bins: Yes

- Minimum subgroup size: 100
- Epochs: 3
- Batch size: 128
- Number of models: 1

CIFAR-100:

- Comparison mode: whole dataset
- Beam width: 3
- Beam depth: 3
- Number of bins: 3
- Dynamic bins: Yes
- Minimum subgroup size: 100
- Epochs: 20
- Batch size: 128
- Number of models: 1

A.3.5 Experiment second quality measure MNIST

- Comparison mode: whole dataset
- Beam width: 8
- Beam depth: 3
- Number of bins: 3
- Dynamic bins: Yes
- Minimum subgroup size: 100
- Epochs: 20
- Batch size: 128
- Number of models: 1

A.3.6 Experiment second quality measure CIFAR-100

- Comparison mode: whole dataset
- Beam width: 16
- Beam depth: 5
- Number of bins: 5
- Dynamic bins: Yes
- Minimum subgroup size: 50
- Epochs: 50
- Batch size: 128
- Number of models: 1

A.4 Threshold values

	Lower threshold	Upper threshold
Layer 0	0.2	∞
Layer 1	0.05	∞

	Lower threshold	Upper threshold
Layer 0	$-\infty$	1.5
Layer 1	$-\infty$	1.0

Table A.14: Threshold values used for experiment in Section 5.6

	Lower threshold	Upper threshold
Layer 0	$\frac{1}{100,000}$	∞
Layer 1	$\frac{1}{100,000}$	∞
Layer 2	$\frac{1}{100,000}$	∞
Layer 3	$\frac{1}{100,000}$	∞
Layer 4	$\frac{1}{1,000,000}$	∞
Layer 5	$\frac{1}{1,000,000}$	∞

	Lower threshold	Upper threshold
Layer 0	$-\infty$	$\frac{1}{100}$
Layer 1	$-\infty$	$\frac{1}{100}$
Layer 2	$-\infty$	$\frac{1}{100}$
Layer 3	$-\infty$	$\frac{1}{100}$
Layer 4	$-\infty$	$\frac{1}{100}$
Layer 5	$-\infty$	$\frac{1}{100}$

Table A.15: Threshold values used for experiment in Section 5.7