Mathematisch-
Naturwissenschaftliche
Fakultät

Methoden des Maschinellen Lernens

Bachelor thesis

An Examination of *Uncertainty aware* Top-k Metrics on MNIST and CIFAR-100

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Methoden des Maschinellen Lernens
Harald Kugler, harald.kugler@student.uni-tuebingen.de

Bearbeitungszeitraum: July-October

Betreuer/Gutachter: Prof. Dr. Philipp Hennig, Universität Tübingen
Zweitgutachter: Marius Hobbhahn, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Harald Kugler (Matrikelnr.: 4258298), October 28, 2021

Abstrakt

Um neuronale Netze (NNs) zu bewerten, werden häufig die Klassen ausgegeben, die die k höchsten Aktivierungen in ihren Ausgangsneuronen aufweisen. Das Ergebnis ist eine statische Top-k-Liste, d. h. eine Liste mit der Länge k für jedes zu klassifizierende Objekt. Bei *uncertainty aware* Top-k-Metriken hängt die Länge der Liste davon ab, wie sicher das NN über die Korrektheit der Klassifikation ist. In dieser Arbeit wurden verschiedene Algorithmen zur Implementierung von *uncertainty aware* Top-k-Metriken an den Datensätzen MNIST und CIFAR-100 evaluiert. Zu diesem Zweck wurde ein Paradigma entwickelt, das sowohl die Redundanz der Ausgabeklassen als auch die Genauigkeit der Klassifikation berücksichtigt. Die Ergebnisse zeigen, dass *uncertainty aware* Top-k-Algorithmen im gegebenen Paradigma eine sehr gute Alternative zu statischen Top-k-Listen sind, da insbesondere die Redundanz der Ausgabeklassen stark reduziert wird. Darüber hinaus wurden Vergleiche zwischen den *uncertainty aware* Algorithmen gezogen. Algorithmen, die mit einer kritischen Überlappung zweier Verteilungen arbeiten, sind robuster als welche, die mit einem *Threshold* arbeiten. Allerdings sind sie auch rechenintensiver.

Abstract

To evaluate neural networks (NNs), the classes that have the k highest activations in their output neurons are often output. The result is a static top-k list, i.e. a list with length k for each object to be classified. For uncertainty aware top-k metrics, the list length depends on how certain the NN is about the correctness of the classification. In this thesis, different algorithms implementing uncertainty aware top-k metrics were evaluated on the MNIST and CIFAR-100 datasets. For this purpose, a paradigm was established that takes into account both the redundancy of the output classes and the accuracy of the classification. The results show that uncertainty aware top-k algorithms in the given paradigm are a very good alternative to static top-k lists since especially the redundancy of the output classes is strongly reduced. Furthermore, comparisons between the uncertainty aware algorithms were drawn. Algorithms dealing with a critical overlap are more robust than algorithms dealing with a threshold value but are often computationally more costly.

Contents

1. Introduction

Since the proposition of Neural Networks (NNs) by Rosenblatt, the field has been growing largely. Artificial Intelligence (AI) is now occupying a fundamental role in society. Whether in road traffic, medicine, sports, marketing or online shopping, there is hardly a significant area of human interaction that does not increasingly use AI. Thus, NNs as part of AI have also gained importance in today's society. They help people make decisions on a daily basis, such as which product to buy, but more and more they are also making decisions without human intervention. A well-known example of this are self-driving cars, which are expected to safely participate in road traffic without human intervention in the future. For this, the car's surroundings must be perceived and correctly classified to avoid accidents which could cause casualties. Medicine also provides an example of how important correct classification is. Detecting cancer from images and choosing the right therapy are sometimes vital for survival.

Therefore, NNs have to classify precisely. Nevertheless, cases occur in which a NN classifies incorrectly. That is why it would be a useful property of a NN, if it would not only output a classification, but also how sure it is about it. Such a property is called *uncertainty aware*.
One possibility to include uncertainty into a NN is with a Bayesian framework. Whereas in classic NNs the weights are assumed to have fixed values, the weights of Bayesian NNs are assumed to follow probability distributions. Often those are Gaussians. Instead of fixed values, now weights depend on distributional parameters that have to be obtained through approximations. Laplace approximations (MacKay [1992]) are one possibility to do so.

The output of the top k classes, i.e. the k classes with the highest activation, has so far been the classical approach to measure the performance of an NN architecture. We will refer refer to this as static top-k. The metric used is accuracy, i.e. the share of how many of the presented data were correctly classified. However, the choice of k is arbitrary because, for one thing, in many cases the top-1 estimate is also the correct classification. Hence, all additionally output classes of the top-k list are redundant. On the other hand, there are cases where five is too small because the first five activations are almost equal to activations five through ten. For example, such a case could occur in image classification when the data contains 10 different classes of car brands. The NN is sure that the image shows a car, but it is not sure about the brand. *Uncertainty aware* top-k metrics could solve this problem. They use uncertainty

methods to produce a top-k list as output, where k depends on how certain the NN is of its classification. Accordingly, a very short list means it is very sure of the correct classification. However, a very long list as output means that no certain classification could be made.

This thesis provides an analysis and evaluation of different *uncertainty aware* top-k metrics. The questions of whether *uncertainty aware* methods are superior to the classical top-k metric and which of these methods is best for which case are to be answered. In addition, trade-offs are made with respect to computational effort. To do this, experiments are performed on two datasets, MNIST and CIFAR-100. Different algorithms that realize *uncertainty aware* top-k metrics and the static top-k are implemented and evaluated with respect to the correct accuracy, the average total length of the top-k metric on the test set, redundancy and computational effort. Therefore, the Accuracy-Redundancy Paradigm is introduced.

2. Theoretical Background

2.1. Neural Networks

2.1.1. Multilayer Perceptron

A NN is a directed graph with vertices and edges. The vertices are arranged in layers. The edges specify the connections between the vertices. Each vertice has an edge to each of the vertices in the previous layer. Such a layer is called *fully connected*. NNs with more than one fully connected layer are called multilayer perceptrons.
The original perceptron only had one layer (Rosenblatt [1958]). Its general idea is adapted from the human brain. Vertices, in this analogy, can be thought of as the neurons. Edges represent the synapses. The edges are also assigned a weight that specifies the strength of the connection between two neurons. Therefore, edges are often simply called weights and vertices neurons.
The output of one neuron depends on three things. First, it depends on the output of all preceding neurons connected by weights o_i. Second, what values these weights w_{ji} have (Terminology: Weight w_{ji} is the value of an edge from neuron i to neuron j). Third and final, each neuron has an activation function f_{act}. Thus, the resulting output of neuron j can be calculated by:

$$o_j = f_{act}(\sum_i w_{ji} o_i)$$

Learning is achieved by adapting the weights. For this purpose a loss or error function L is defined. The loss function L gives the error of the net over all training patterns as a function of the weights. The graph resulting from this function is called error or loss surface. Thus, finding the minimum of this function gives the best possible weight values. As this minimum cannot be find analytically, steps in direction of the steepest descent are taken, i.e. in the opposite direction of the gradient. Weight adaption by taking these steps is called *backpropagation*.

2.1.2. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) differ in certain key aspects from NNs discussed until now. Especially in the field of Computer Vision, i.e. image classification, this new network architecture is held responsible for many accomplishments.
A CNN has three different types of layers of neurons:

- *Convolutional layer*
  Convolutional layers have the task to extract certain features of the image. The later/deeper a convolutional layer is positioned within a NN, the more high-level are the features to be extracted. This is done by defining a Kernel/Filter K. A kernel is defined by a certain size, e.g. 5x5 pixels, and weights belonging to each pixel. The kernel now shifts over the image. The activation of the next layer is calculated by normal matrix multiplication followed by an activation function.
- *Pooling layer*
  Pooling layers are useful to reduce the dimensions of the image. Therefore, as in convolutional layers, a kernel is defined, e.g. 2x2 pixels. This kernel again shifts over the image. For maxpooling layers, the respective maximum in each shift is used as activation for the next layer, for average pooling layers, the average value of the kernel is calculated and used.
- *Fully-connected layer*
  In a CNN fully-conntected layers can be found at the end of the net. They are responsible for classification. In theory, all extracted features of the image are evaluated and depending on the evaluation, conclusions are drawn about the respective class of the image.

Training a CNN can be achieved with *backpropagation* similar to multilayer perceptrons.

2.1.3. Residual Networks (ResNets)

ResNets are CNNs which differ in only one key aspect from them. They have so called skip connections between certain building blocks. The skip connections are using the identity function as their activation function. Thus, the identity of a previous layer is added to the activation after the building block. To solve the common problem of non-matching dimensions arising from convolution and pooling layers, the identity mapping is multiplied with a linear projection.
The resulting network architecture has empirically shown to perform better when deeper CNNs are necessary (He et al. [2016]).

2.2. Laplace Approximations

In classic NNs, weights are determined by gradient descent. After learning the weights, they are assumed to be fixed. This procedure can be seen as a frequentistic approach to interpret NNs. The parameter offering the best explanation of the data at hand is chosen, which essentially is a maximum likelihood estimation.
In Bayesian NNs, the data is assumed to be fixed while the weights are interpreted as random variable. The probability distribution of the weights can be calculated with the Bayes theorem and is called the posterior $p(\theta \mid \mathcal{D})$. However, this posterior is intractable because the integral of the denominator cannot be solved analytically. That is why the posterior of the weights has to be approximated if a Bayesian framework is desired.
Laplace approximations (MacKay [1992]) offer one possibility to achieve this. They construct a Gaussian distribution $\mathcal{N}(\theta \mid \mu, \Sigma)$ around a single given mode. This mode is the mean μ of the Gaussian and can be determined by standard backpropagation. The covariance matrix Σ is constructed as the inverse of the Hessian of the loss landscape.
Since the computation of such an inverse is often too expensive, assumptions about the Hessian need to be made in order to keep the computational effort within reasonable limits.

- *Diagonal Laplace Approximations*
  In the diagonal Laplace approximations, it is assumed that the Hessian H has the following property. Every entry h_{ij} of the matrix is zero if $i \neq j$, i.e. only on the main diagonal there are entries unequal zero. The inverse of such a matrix can be easily calculated:

$$H^{-1} = \begin{cases} \frac{1}{h_{ij}} & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

  Taking only the diagonal elements of a Hessian into account means that the covariance between two dimensions (weights) is not considered, i.e. some information on the loss landscape is lost.
  To reduce the computational effort even further, one can also assume that only the weights in the last layer are normally distributed. The weights in all previous layers are assumed to be fixed. This is called a Laplace approximation of the last layer.

2.3. Laplace Bridge

Obtaining posterior distributions is computationally costly. First, Laplace Approximations have to be performed. In a second step it is necessary to draw samples from these approximations. The third and final step then consists of applying the softmax

function.
Sampling is necessary because the distribution in the output space is non-parametric and therefore cannot be described analytically. This issue is addressed by the work of Hobbhahn et al.. They used a method that resolves computationally costly sampling which is called the *Laplace Bridge*. The *Laplace Bridge* approximates the posterior distribution analytically. Thus, the computational effort due to sampling is saved.
The *Laplace Bridge* is based on the observation that a Dirichlet distribution looks quite similar to a Gaussian if you change its variables with the Softmax-Function.

$$\sigma : \mathbb{R}^K \to [0,1]^K : \sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Now, if we assume that the Gaussian in the logit space is given, a direct map from the parameters μ and Σ of the Gaussian to the parameters α_k of the Dirichlet distribution in the standard basis exists.

$$\alpha_k = \frac{1}{\Sigma_{kk}}\left(1 - \frac{2}{K} + \frac{e^{\mu_k}}{K^2} \sum_{l}^{K} e^{-\mu_l}\right)$$

This is useful as the Dirichlet function in the standard basis is defined on the intervall $I = [0,1]$.
An algorithm for *uncertainty aware* top-k based on the *Laplace Bridge* is described in Section 3.5.

3. Algorithms

3.1. Static Top-k

Top-k outputs only the classes belonging to the k output neurons with the highest activations in descending order, e.g. the top five classes.
Therefore, strictly speaking, this algorithm does not provide a result that takes uncertainty into account. Rather, it serves as a benchmark which should be outperformed by the uncertainty aware algorithms to justify such approaches in the first place.

3.2. Simple Top-X% of Probability Vectors

The activations of the output neurons are mapped to values between zero and one, which allows the resulting values to be interpreted as probabilities for the corresponding classes. The mapping can be accomplished with the softmax function:

$$\sigma : \mathbb{R}^K \rightarrow [0,1]^K : \sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

This algorithm sorts the classes in descending order based on the probabilities resulting from the softmax function and starts adding classes to the uncertainty aware top-k list. In the process, the probabilities of the added classes are summed up until a defined threshold value is exceeded, e.g. 0.5. For the probabilities in Figure 3.1 this would mean that class 5 and class 6 are included in the top-k list because added together they exceed the threshold value of 0.5. Consequently, all the other classes are left out the top-k list because after exceeding the threshold, the algorithm terminates and no further classes are added to the list. This algorithm is referred to as simple top-x throughout this thesis.

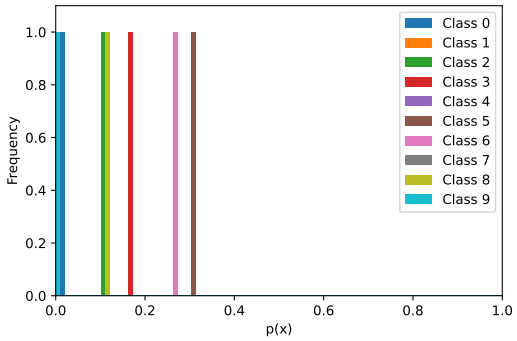

Figure 3.1.: Visulization of a softmaxed output of a NN with 10 output neurons. simple top-x and LA-mean adding these values up in descending order until a threshold value is exceeded. LA-mean is generating these values out of the means of samples shown in Figure 3.2.

3.3. Top-X% of Probability Vectors for Last-Layer Laplace Approximations

This algorithm works quite similarly to the simple top-x algorithm described in the Section above. The only difference between these two algorithms is that this time Laplace approximations are used. Only the weights of the last layer of the neural network are approximated using Laplace. Consequently, all other weights in the previous layers are static after they have been trained. The last-layer Laplace approximation achieves promising results with respect to modeling uncertainty in a NN Kristiadi et al. [2020].
From the resulting Gaussian distribution, 1000 output vectors are now generated by sampling. Each output vector is then normalized using the softmax function. After that, the mean value of all output vectors is calculated. The predicted classes of the top-k list are emitted using the same principle as in the simple top-x algorithm with the help of a threshold.
The composition of the softmax function and the mean are not commutative. Thus, the results differ from the simple top-x. This algorithm will be called LA-mean from now on.

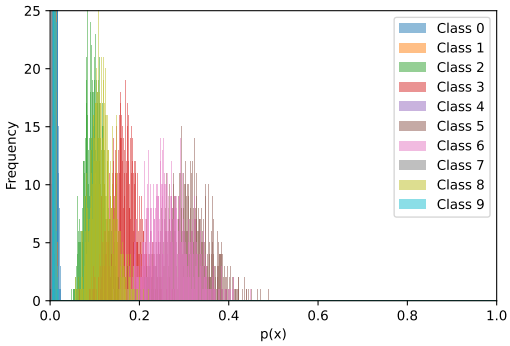

Figure 3.2.: Visualization of the distributions generated by sampling. These are needed for the LA algorithms (LA-mean and LA-overlap). LA-overlap checks the overlap between two histograms by comparing two quantiles. LA-mean is building the mean for each class to generate an output as shown in Figure 3.1.

3.4. Overlap of Samples for Last-Layer Laplace Approximations

Again, last-layer Laplace approximations are used to create a Gaussian distribution. As above, this distribution is used to draw 1000 samples. After that, the output vectors are again normalized with the softmax function. Now, there exists a histogram of samples for each class (Figure 3.2).
To model uncertainty a critical overlap x is defined (e.g. 0.05). The first step is to sort the classes in descending order according to the respective sample means. After that, the x-quantile of the upper class is compared with the $(1-x)$-quantile of the lower class. If it is smaller or equal, the lower class is added to the top-k list and becomes the upper class in the next comparison. The class with the next smaller mean becomes the lower class. If the x-quantile is larger, the algorithm terminates and no further classes are added to the top-k list.
In the upper paragraph an algorithm is described which creates an uncertainty-aware top-k list from the comparison of two adjacent classes. However, another possibility is to take only the class with the largest mean value and to perform the comparison with all lower classes. The classes with a sufficiently large overlap with the top class are added to the top-k list.
The first method, which compares two adjacent classes, will be referred to as LA-overlap adjacent. The second, which only compares the top class with the lower classes, will be called LA-overlap top.

3.5. Uncertainty aware with the Laplace Bridge

Uncertainty aware can also be realized through a method proposed by Hobbhahn et al. without sampling. The Laplace Bridge is used to get a Dirichlet distribution in the output space. As the marginal distribution over each component of a Dirichlet relative to all the other components is a Beta distribution over these parameters, one can use the following algorithm to get an uncertainty aware top-k metric.
First, the class predictions are sorted in descending order based on their probability. If the adjacent beta marginals of the top two classes overlap more than a defined critical overlap (e.g. 0.05), the second class is added to the top-k classes. This process is repeated until two adjacent beta marginals do not overlap sufficiently (next comparison is between the second and third class). In this case the algorithm terminates. The result is a list of the top-k classes for this specific classification problem.
As in Section 3.4, it is again possible to make the comparison just between top class and all other classes. The adjacent method is in this case called LB adjacent, the other one LB top.

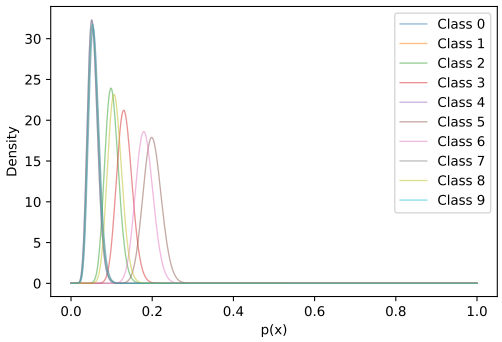

Figure 3.3.: Marginal distributions of a Dirichlet. The α-parameters of the Dirichlet are calculated through the Laplace Bridge. LB algorithms test those marginals for a sufficient overlap.

4. Methods

4.1. Datasets

For this thesis, experiments were conducted to evaluate the algorithms in Section 3 using two different datasets, MNIST and CIFAR-100.
The MNIST data set consists of images that show handwritten digits. So in total there are 10 different classes in the data set (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). The training set includes a total of 60.000 images. The test set consists of 10.000 images. Every image has a pixel size of 32x32. As the images in MNIST are black and white, every image only provides one input channel (LeCun et al. [1998]).
The CIFAR-100 data set has ten times as many classes as MNIST, thus 100 classes exist. Nevertheless, it includes 60.000 training and 10.000 test images as well as MNIST. The images of CIFAR are colored and therefore have three input channels (Krizhevsky et al. [2009]).

4.2. Network Architectures

For the MNIST experiments a CNN consisting out of two convolution and three linear, fully connected layers was trained on the full training set. The kernel size in the convolution layers corresponded to 5x5 pixels. After each convolution layer the outputs were maxpooled through a kernel of 2x2 pixels. The three fully connected layers consisted of 120, 84 and 10 neurons, respectively.
Training was performed with a learning rate of 0.1. The loss function was the cross entropy loss $L = -\sum_i^n t_i \cdot log(p_i)$. As the optimizer function the stochastic gradient descent was used. A batch consisted of four images. Training was iterated over the whole training set two times.
CIFAR-100 experiments were performed with a ResNet. We used pretrained weights which made training unnecessary. The ResNet consisted of 18 layers, 17 convolutional layers and one fully-connected layer. Beginning at the second layer, skip connections over two layers each were integrated. In the CIFAR-100 Experiments we used a batchsize of 128.

4.3. The Accuracy-Redundancy Paradigm (ARP)

In classic NN Performance can be measured by Receiver Operating Characterisitcs (ROC). ROC curves show the performance of a classifier as a function of different decision thresholds. Therefore, the true positive rate and the false positive rate are measured and plotted against each other in a 2-dimensional graph. The principle of ROC is best explained on an example of a binary classifier. The true positive rate is hereby the correct classified elements of class 1 divided by all elements of class 1 that the NN has seen. The false positive rate is defined as the elements classified as class 1 although they were belonging to class 2 divided by all elements of class 2.
We adapt this scheme to measure performance of *uncertainty aware* top-k metrics. The true positive rate is quite the same as in the normal ROC setup. It measures how many percent of the test set consisting out of N images are classified correctly, i.e. the accuracy rate r_a.

$$r_a = \frac{correct}{N}$$

The false positive elements now equal those classes that have been added to the top-k metric although the correct class is already an element of it. We call those elements *redundant*. To get the redundancy rate r_r we have to divide by all classes that have been outputted. The average length *avgl* is the mean of the lengths of all produced top-k lists through evaluating the test images. $Avgl_{cor}$ is the average length of all correct classified images.

$$r_r = \frac{r}{avgl_{cor} * r_a * N}$$

We call this new scheme the ARP. An *uncertainty aware* metric is considered good if it has a very high accuracy and a small redundancy, i.e. the data point of this metric in the ARP is close to the upper left corner of the graph which can be seen as the optimum (Figure 4.1).

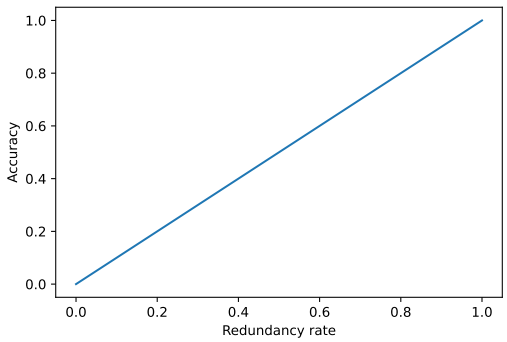

Figure 4.1.: The plot of the ARP. It captures two dimensions, the redundancy rate and the accuracy, of an *uncertainty aware* top-k algorithm. The optimal algorithm has an accuracy rate of 1. Simultaneously, its redundancy rate equals 0.

5. Experiments

5.1. MNIST

5.1.1. Data Aquisition

A CNN with an architecture described in Section 4.2 was trained four times with training sets of different sizes. The first training was conducted with the full training set provided from MNIST. This net is called CNN-full from now on. The other three training sets contained 1000, 2000 and 3000 images, respectively, which were randomly chosen out of the full training set. We call these nets CNN-1k, CNN-2k and CNN-3k, respectively. Then, for each of these trained NNs, last-layer Laplace approximations were performed. In order to do this the package *backpack* (Dangel et al. [2020]) was used. In the final step the validation set was predicted for every algorithm described in Section 3.
Static top-k was evaluated for every possible k, i.e. ten times ($k \in \{1,2,3,...,10\}$).
For the algorithms that add up the outputs until a certain threshold value is reached (top-x, LA-mean), ten different threshold values were used:

$$0.997, 0.97, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.25, 0.1$$

For algorithms that generate a top-k based on critical overlaps of distribution or histograms of samples (LA-overlap adjacent, LA-overlap top, LB adjacent, LB top), ten different critical overlaps were used:

$$0.001, 0.01, 0.025, 0.05, 0.1, 0.2, 0.3, 0.5, 0.75, 0.9$$

The choice of these thresholds and critical overlaps is somewhat arbitrary. However, the choice attempts to select values that provide the greatest possible insights.
In total, the validation set was predicted 280 times (4 training data sets x 7 algorithms x 10 thresholds/critical overlaps/k).

5.1.2. Results

All results of the MNIST experiments can be found in the tables in Appendix A.

- CNN-full yields a static Top-1 overall accuracy of 98.16%. This is very high, which can be traced to the simplicity of MNIST. CNN-3k provides a static top-1 overall accuracy of 89.34%. The other two nets, CNN-2k and CNN-1k, have a

static top-1 overall accuracy of 51.36% and 29.81%, respectively. So, the fewer training images the net has seen, the lower the overall accuracy.
As expected, static top-2 yields higher overall accuracies for each of the four nets. CNN-full provides a overall accuracy of 99.67%. This means that in 1.51% of the images in the validation set is the classes with the second highest activation in its output neuron is the correct class. CNN-3k, CNN-2k and CNN-1k provide overall accuracies of 97.10%, 67.74% and 46.46%, respectively. Consequently, the number of images that are correctly classified based on the second highest activation increases when the uncertainty of the NN is higher due to manipulations on the amount of training data.
This scheme can be confirmed by the static top-3 algorithm. The less training data a net has received, the more cases there are where the class corresponding to the neuron with the third highest activation is the correct one. The static top-3 accuracies of the respective NNs are 99.89%, 98.68%, 79.35% and 56.49% (CNN-full, CNN-3k, CNN-2k and CNN-1k). For CNN-full this means that only eleven images exists where the resulting outputs of the NN does not yield the correct class amongst the three highest activations.
The average lengths of the static top-k algorithm are trivial and given by the defined k.

- Simple top-x is the first algorithm which deals with thresholds manipulated for analysis. As one might expect, the average lengths and overall accuracy decreases with smaller threshold values. CNN-full provides an average length of 1.55 with an accuracy of 99.96% if the threshold value equals 0.997. For MNIST this means only four images of the validation set were classified incorrectly. Hence, simple top-x with a threshold value of 0.997 outperforms static top-3 in both accuracy and average length.
  The average length decreases most between the threshold values 0.997 and 0.97 from 1.55 to 1.15. This means that the strongest softmaxed activation is often between these two values in CNN-full. After that, the average length decreases only slightly until, at a threshold of 0.1, no second class is included in the top-k list in any case.
  Note that the simple top-x algorithm with a threshold value of 0.1 is always equivalent to the static top-1 algorithm in the MNIST case. This is because, according to the pigeonhole principle, if there are ten output neurons, at least one of them will have a softmaxed output equal or higher than 0.1. Thus, the threshold value is exceeded with the first element of the resulting top-k list, which leads to the termination of the algorithm. This is not the case for other data sets with more than ten classes and hence more than ten output neurons in the NN.
  Training with less images leads to a smaller accuracy rate and to larger average lengths. Still, CNN-3k yields a very high accuracy at 99.93% with the highest explored threshold value. However, its average length is about three times as large as in CNN-full (4.90). The redundancy rate is quite similar as in the

CNN-full case.
In CNN-2k, the pattern continues as average lengths increases and accuracy decreases throughout the thresholds. For the two highest thresholds 0.997 and 0.97 on average more than half of the classes are included in the top-k list. Reducing the threshold leads to a higher decrease in accuracy. The range of accuracy is now between 70.14% and 99.93%. This means that more cases occur where the top-1 output is not the correct one and does not exceed the threshold value.
An average length of 10.0 occurs in the CNN-1k net for the highest three threshold values. Thus, every class is included in the top-k list. Of course in such a case the informative value of the corresponding data is very small. Also noteworthy is that the average length always equals the threshold value rounded up to tenths times ten. Such a behavior would also be shown by a NN that always raises the same softmaxed output for every neuron in the output layer, i.e. the NN does not know anything about the data. However, the accuracy rates do not confirm this assumption as they are higher than the threshold value. So although the softmaxed outputs are very close to each other, the largest values still correspond to the correct class more often than not.

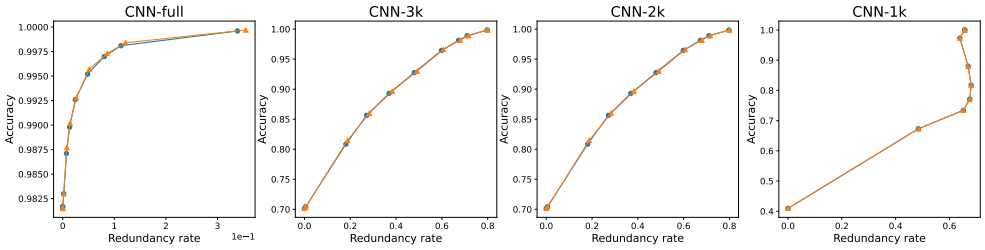

Figure 5.1.: Simple top-x vs. LA-mean in the ARP for the tested NNs. We see that the two algorithms do not differ by a significant amount for either of the examined NNs.

- LA-mean produces results that are very similar to simple top-x. However, there are minor differences in the three indicators accuracy, average length and redundancy rate. LA-mean shows somewhat higher values here in any case. However, these are not significant in ARP, as Figure 5.1 also shows. In each of the four NNs, the accuracy-redundancy curve does not differ. The similarity between those to algorithms means that the composition of the mean and the softmax function yields similar values in the output space.

- Two groups of algorithms dealing with a critical overlap were examined. In one, the distribution of the top class was tested against all distributions of the lower classes for critical overlap (LB and LA-overlap top); in the other, the respective adjacent classes were considered (LB and LA-overlap adjacent).
LA-overlap top has a smaller average length than its pendant LA-overlap adjacent throughout all conditions. This seems fairly obvious, as the difference

of the means of the two compared distributions in LA-overlap adjacent is smaller than the same difference in LA-overlap top. Hence, the quantiles are more likely to overlap. This leads to more classes being included in the top-k list. This discrepancy is largest in the case of a small critical overlap, i.e. a value of 0.001. In CNN-full LA-overlap adjacent has an average length of 1.41 whereas LA-overlap top provides an average length of 1.05. CNN-3k, CNN-2k and CNN-1k show an average length of 3.73, 7.70 and 10.0 with LA-overlap adjacent and an average length of 1.59, 2.87 and 8.56 with LA-overlap top, respectively. Especially in CNN-2k the reduction of the average length is substantial. Also we see that the LA-overlap top algorithm avoids to put out all classes for the smallest observed overlap in CNN-1k.
Although average length of the adjacent algorithm exceeds the one of the top algorithm, interestingly accuracy does not differ significantly between those two algorithms. This is can be seen in Figure 5.2 where similar accuracy rates of the algorithms are achieved. In CNN-full for the three smallest critical overlaps accuracy rates over 99.00% are achieved, which is also the case for the LA-overlap adjacent algorithm. In the other three NNs the difference in terms of accuracy grows. Nevertheless, the algorithms are in the same order of magnitude.
The higher average length of the adjacent algorithm leads to a higher redundancy rate. A exception is thrown here by CNN-1k. LA-overlap adjacent provides here better results with a critical overlap of 0.2. Note that this is a one time exception and the data points of LA-overlap top are distributed more equally over the graph.

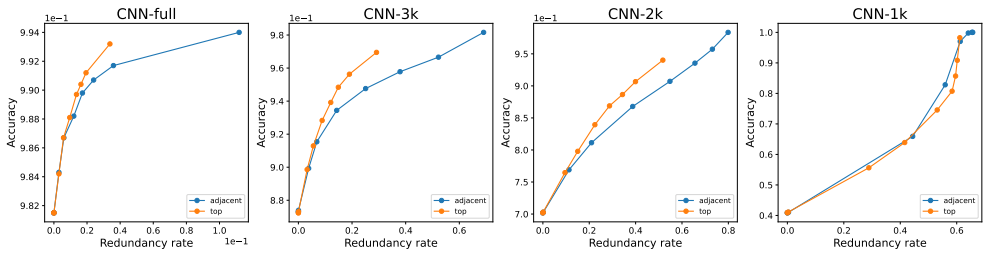

Figure 5.2.: LA-overlap top vs. LA-overlap adjacent in the ARP for the tested NNs. The top algorithm provides smaller redundancy rates with approximately the same accuracy rates compared to the adjacent algorithm.

The same pattern can be found when comparing LB adjacent and LB top.
LB adjacent provides higher average lengths than LB top for the same reason described above. However, the absolute differences are not as large as for the LA-overlap algorithms. LB adjacent provides an average length of 1.0048, whereas LB top yields an average length of 1.0026 with a critical overlap value of 0.001 in CNN-full. In CNN-3k and CNN-2k, this difference increases. CNN-3k provides an average length of 1.12 for LB top and 1.26 for LB adjacent,

CNN-3k 2.20 for LB top and 3.30 for LB adjacent.
Again, there are no significant differences in the accuracy rate between the top and the adjacent algorithm. LB adjacent provides a maximum of 99.24% correct classified images. Comparing this to the maximum of LB top which is 99.23% shows that LB adjacent classifies only one additional image correctly. Just as above for the LA-overlap algorithms, this difference increases in CNN-3k and CNN-2k. However, the accuracy remains in the same order of magnitude for the two algorithms.
In CNN-1k, there is almost no difference between the LB top and the LB adjacent algorithm. Only for a critical overlap value of 0.3 they differ in terms of average length as LB top does not output every possible class in this case (average length: 9.9321). For lower critical overlaps both algorithms have an average length of ten, i.e. they output all classes. For higher critical overlaps no further class is included into the top-k list. This leads to an average length of 1.00 in these cases.

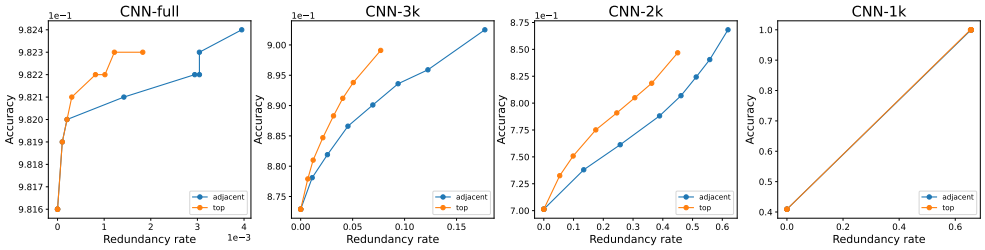

Figure 5.3.: LB top vs. LB adjacent in the ARP for the tested NNs. In CNN-full, CNN-3k and CNN-2k, the top algorithm provides smaller redundancy rates accompanied with approximately the same accuracy rates. In CNN-1k, both algorithms output either one or ten classes for the most of the examined critical overlaps.

- Overall comparison of the algorithms in the ARP yields the following results. The LB algorithm is very conservative in a sense that not many classes are added to the top-k list. Although this leads to a small redundancy rate throughout the evaluated critical overlaps, accuracy does not improve as well as in the other algorithms. In CNN-1k, LB-top does include either all classes or none of them in the top-k list. Hence, LB-top does not provide meaningful results in this case. Excluding CNN-1k because of this, accuracy increases in CNN-2k the most. Surprisingly, the associated curve even drops below the one of static top-k. This means that static top-2 has a higher accuracy with a smaller redundancy rate than LB-top with a critical overlap of 0.001.
  In all other cases static top-k is the most redundant algorithm. The more training data the net has seen, the bigger is the gap in terms of redundancy between static top-k and the other algorithms. In CNN-full even static top-2 reaches a higher redundancy rate than every other algorithm, while in CNN-1k the curves of the simple top-x and static top-k are nearly identical.

Simple top-x is the algorithm that reaches the highest accuracy rates. Especially in CNN-full and CNN-3k, this is achieved by reasonable redundancy rates. As training data is reduced, the redundancy rates resemble more and more the ones of static top-k.

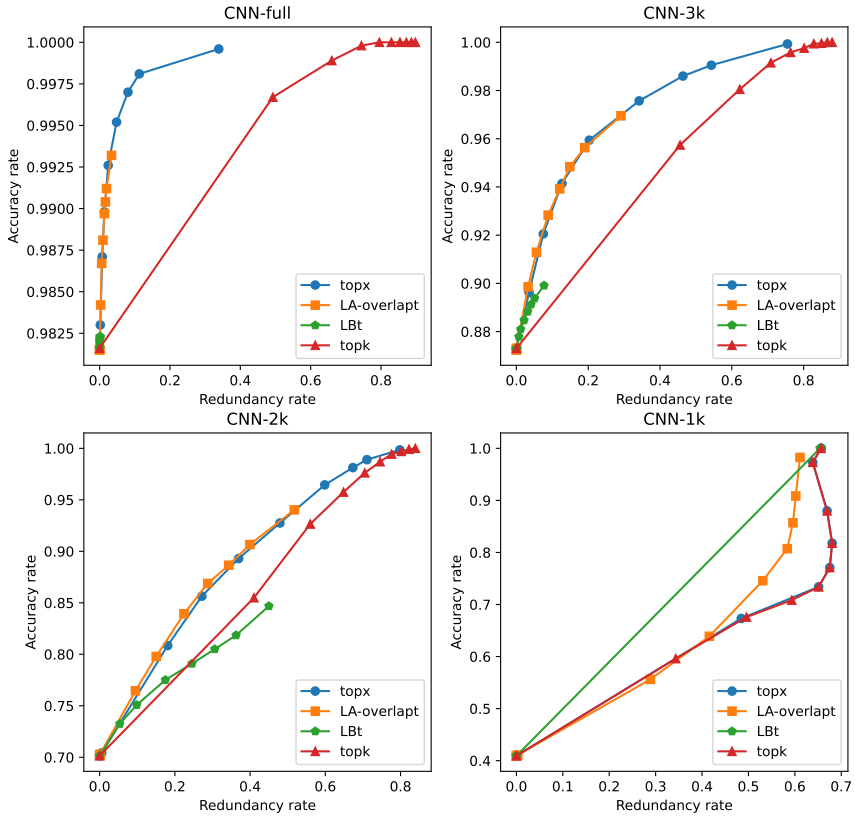

Figure 5.4.: Overall comparison of the examined algorithms in the ARP. The adjacent-algorithms were left out because they are more redundant but not more accurate than the top-algorithms. LA-mean is left out because of the similarity of results to simple top-x. Static top-k is the most redundant algorithm. The curves LA-overlap top and simple top-x are very similar to each other. However, LA-overlap is more robust against the manipulation of its criteria (critical overlap). Simple top-x reaches higher redundancy rates if its threshold value is small. LB top is the most conservative algorithm in adding classes to the top-k list. Thus, it does not reach the accuracy rates of the other algorithms.

5.2. CIFAR-100

5.2.1. Data Aquisition

In the CIFAR-100 Experiments a ResNet18 with pretrained weights was used (Section 4.2). Again, in order to perform the last-layer Laplace approximations *backpack* (Dangel et al. [2020]) was used. As CIFAR-100 contains 100 classes, it does not make sense to evaluate static top-k for every possible k. Therefore, ten different values between 0 and 20 were chosen:

$$1, 2, 3, 4, 5, 7, 10, 12, 15, 20$$

For the other two groups of algorithms, i.e. algorithms that deal with critical overlaps respectively algorithms that deal with threshold values, the same values as in the MNIST experiments were used (Section 5.1).
This means the following values were used as thresholds:

$$0.997, 0.97, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.25, 0.1$$

Algorithms that deal with critical overlaps were evaluated with those values:

$$0.001, 0.01, 0.025, 0.05, 0.1, 0.2, 0.3, 0.5, 0.75, 0.9$$

Thus, the CIFAR-100 validation set was evaluated a total of 70 times (7 algorithms x 10 thresholds/overlaps/k).

5.2.2. Results

All results of the CIFAR-100 experiments can be found in the tables in Appendix B.

- Static top-1 reaches an accuracy of 76.23%. This already shows that CIFAR-100 is a more complex data set than MNIST. With $k = 20$, static top-k has an accuracy of 98.57. Thus, not for every image in the test does the correct class rank amongst the 20 highest activations. The biggest difference can be found between $k = 1$ and $k = 2$, showing that in just under 13% of cases, the correct class is the one that belongs to the neuron with the second highest output.
- The two algorithms dealing with threshold values reveal similar results as in the MNIST experiments. Again, simple top-x and LA-mean provide nearly the same curve in the ARP. Again, the values for the three measured variables of accuracy, average length and redundancy rate are slightly higher for LA-mean, but not significantly different. LA-mean has an accuracy of 97.95% and an average length of 11.38 for a threshold of 0.997, whereas the simple top-x algorithm provides an accuracy of 97.91% and an average length of 11.16 in this case. Unlike as in the MNIST experiments, the average length in the case of the smallest threshold value 0.1 is larger than 1 although not by far (1.0011).

So, there are a few cases where the activation belonging to the most activated neuron is smaller then 0.1. In the ARP we see a linear increase throughout the thresholds (Figure 5.5). This is also a difference to the MNIST experiments.

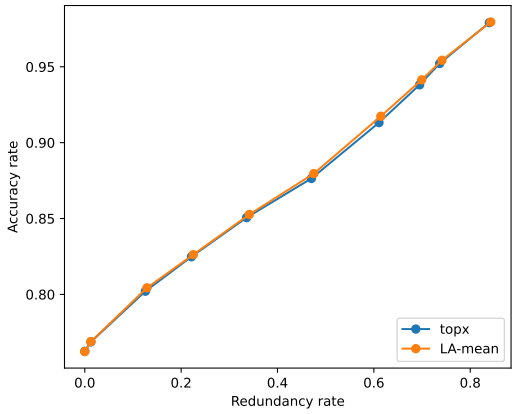

Figure 5.5.: Simple top-x vs. LA-mean in the ARP on the CIFAR-100 data set. They nearly produce the same outputs in the ARP.

- As in the MNIST experiments, algorithms that examine the overlap of the top class to other classes provide smaller redundancy rates compared to the ones comparing two adjacent classes. Mainly, this is because they do not include not as much classes in the top-k list, i.e. they have smaller average lengths. For instance, LB top has an average length of 2.08, whereas LB adjacent provides an average length of 2.85 for the smallest examined critical overlap. A even bigger difference occurs at the LA overlap algorithms. LA-overlap top has an average length of 1.53, while LA-overlap adjacent results in an average length of 4.21 with a critical overlap of 0.001.
  A similarity in pattern compared to the MNIST experiments continues in terms of accuracy. Both LB algorithms show accuracy rates over 80.00% for the three smallest examined overlaps. Specifically, LB top provides rates of 81.81%, 80.78% and 80.11% for overlaps of 0.001, 0.01 and 0.025, respectively. LB adjacent yields percentages of 83.27%, 82.02% and 81.31% for these cases. The same is true for the LA-overlap algorithms even for the four smallest overlaps. In LA-overlap top, accuracy rates range from 84.41% to 80.71%, in LA-overlap adjacent from 88.86% to 83.09% in these cases. The fact that the top algorithms provide reasonable accuracy rates together with reduced redundancy rates caused by smaller average lengths is also shown in the ARP(Figure 5.6). We see that the curves corresponding to the top algorithms are closer to the top left corner representing the optimum.

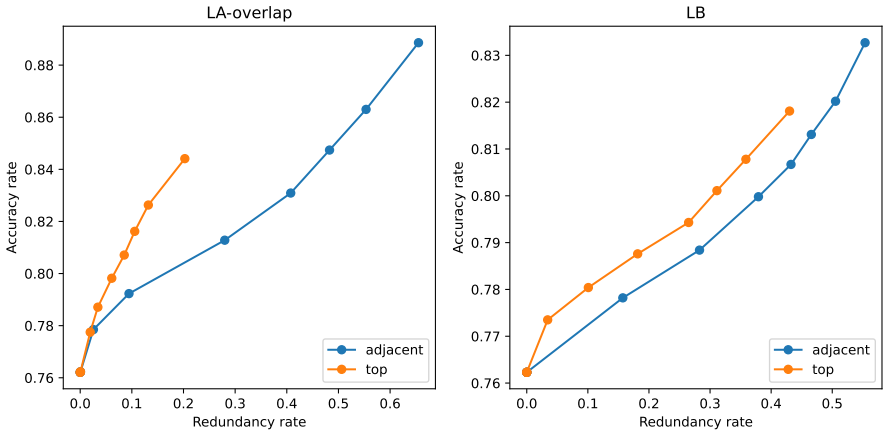

Figure 5.6.: Top vs. adjacent algorithms in the ARP on the CIFAR-100 data set. Left: LA-overlap adjacent vs. LA-overlap top. Right: LB adjacent vs. LB top. In both cases, the top algorithms provide smaller redundancy rates. At the same time, they result in accuracy rates that are not significantly smaller than the ones of the adjacent algorithms.

- Overall comparison of the examined algorithms shows that LA-overlap top is the most conservative algorithm, i.e. has the smallest average lengths. This is in contrast to the MNIST experiments where LB top provided the smaller average lengths. Nevertheless, LA-overlap reaches higher accuracy rate leading to smaller redundancy rates what every other algorithm is achieving. The highest accuracy overall is reached by static top-20 at the cost of the highest redundancy rate. Simple top-x provides a similar curve but with smaller redundancy rates. However, simple top-x reaches very high redundancy rates at 0.8 for small thresholds as well, meaning that every 4 out of 5 elements in a top-k list are redundant. This issue appears not to be a problem in LA-overlap. Noteworthy is that the curve corresponding to LB top is below all other curves, even the one of static top-k.

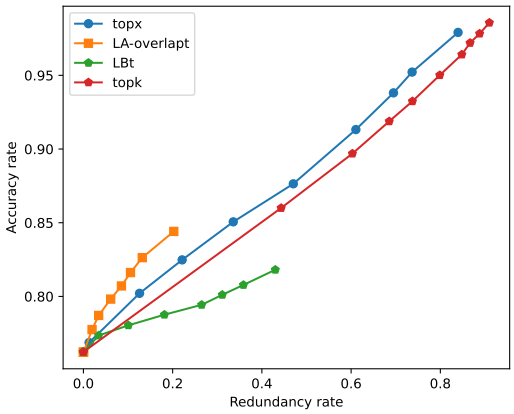

Figure 5.7.: Overall comparison of the examined algorithms on CIFAR-100 in the ARP. The adjacent algorithms were left out because they are more redundant but not significant more accurate than the top-algorithms. LA-mean is left out because of the similarity of results to simple top-x. The results yielded by simple top-x and static top-k are very similar. Simple top-x provides a slightly smaller redundancy rate throughout the examined thresholds. LA-overlap top results in the smallest redundancy rate but does not yield accuracy rates as high as the first two. LB top yields the smallest accuracy rates. Nevertheless, its redundancy rate much higher than the one LA-adjacent top provides.

6. Discussion

In this thesis, six algorithms that output a *uncertainty aware* top-k list and the static top-k algorithm were presented and analyzed. For this purpose, experiments on two datasets, MNIST and CIFAR-100, were conducted.
In order to compare the algorithms, a paradigm based on ROC-curves was introduced. The ARP (Section 4.3) captures two dimensions of *uncertainty aware* metrics, i.e. the accuracy and the redundancy rate.
While accuracy is a very straightforward dimension with hardly any ambiguities in interpretation, the redundancy rate offers room for discussion. In this thesis, we used the formula $r_r = \frac{r}{Avgl_{cor} * r_a * N}$ to calculate the redundancy rate. It is meant to measure how many of all classes that are in the top-k lists are redundant. Since one could argue that at least one class must be in such a list by design, one could also consider the formula $r_r = \frac{r}{(Avgl_{cor} - 1) * r_a * N}$. In this case only the to the top-k list added classes are taken into account in the denominator. However, one downside of this method is that algorithms that are very conservative in adding classes to the top-k list, such as LB top or adjacent in the MNIST experiments, can still have very high redundancy rates. For example, if the length of the top-k list equals two only in one case and in all other instances is equal to one, the redundancy rate is zero if the added class is not redundant, or one if it is. Furthermore, with this formula it is possible that a division by zero is possible when for no image further classes are added to the top-k list. To intercept this special case is relatively easy. One could just define the redundancy rate in such a case to be zero as there are no redundant classes at all.
After all, in both of these measurements for redundancy rate the flaw still exists that they only handle the classes already added to a top-k list. Therefore, no statement is made about how many additional classes were not added. This could be fixed by defining a basic set of possible redundant classes which could be outputted. Therefore, for bigger datasets, a maximal list length k_{max} of the top-k list has to be defined. Then, the redundant classes of the static top-k algorithm with $k = k_{max}$ are used in the denominator. However, this method always produces incredible small values in redundancy rates for nets with a good base accuracy because a list length of k_{max} is almost never reached by the used *uncertainty aware* top-k algorithms in the experiments conducted here. Furthermore, it rewards algorithms which are very restrictive in adding classes to the top-k list like the LB top and adjacent in the MNIST experiments.
Both alternative methods for measuring the redundancy rate were tried in the experiments, but did not yield results that would have led to other interpretations.

As already mentioned earlier, the ARP captures two dimension of *uncertainty aware* top-k lists. Possibly useful features besides accuracy and redundancy that convey information about the quality of the algorithms are not considered. For example, one such useful feature could be the length of the top-k lists for misclassified images. Since the length of the top-k lists indicates how certain the NN is about the correct class, a good algorithm would have larger list lengths for misclassified images. A first indication that this might be the case is the fact that the average length of correctly classified images is often slightly shorter than the average length of all images. Nevertheless, ARP does not help to compare this feature in different algorithms.

With the used ARP, *uncertainty aware* algorithms outperform the static top-k algorithm. In both the MNIST and CIFAR-100 experiments, they show much lower redundancy rates with approximately the same or even higher accuracy. However, in order to evaluate a NN a certain threshold or a critical overlap has to be defined for these methods. As the individual optimum might be different for every NN, preliminary investigations are necessary in order to provide a meaningful result of the NN's performance. Static top-k has the advantage that given a k comparison of different NN architectures can be achieved in the simplest way possible.
Furthermore, *uncertainty aware* algorithms are computationally more costly than static top-k. Especially the algorithms that involves sampling from a distribution, i.e. the LA-mean and the two LA-overlap algorithms, surpass the computational effort of static top-k by far. Nevertheless, there are algorithms that are not that computationally costly. Simple top-x provides an *uncertainty aware* list with a computational effort within $O(n)$ where n defines the number of classes. The same is valid for the LB algorithms once Laplace approximations have been performed.
Mainly simple top-x seems to be a cheap but in many cases very efficient method to realize *uncertainty aware* top-k once the threshold value is defined. On the other hand LA-mean yields very similar results but with a much bigger computational effort. Thus, the LA-mean might not be a useful algorithm.
The LB algorithms provide qualitative different results for the two examined datasets. They seem to be very conservative in adding classes to the top-k list for simple datasets like MNIST. For CIFAR-100 there are algorithms that produce shorter top-k lists on average (LA-overlap top). However, the results of both datasets have in common that they are not similar to the results that the LA-overlap algorithms provide. This is somewhat surprising since the Laplace Bridge attempts to approximate the posterior distributions. The approximation does not appear to have been sufficiently accurate, at least for the two datasets examined.
As already mentioned, the LA-overlap algorithms are by far the most costly algorithm examined due to sampling that produce . Nevertheless, it yields advantages over top-x because defining the critical overlap does not have as large of an influence on accuracy as defining a threshold value. In other words, by choosing the right threshold value the performance of a NN can be easily manipulated.
Comparison of the single *uncertainty aware* algorithms dealing with critical overlap shows that the algorithms comparing the top class with the others seem to be

superior to the ones comparing adjacent classes. This is very straightforward as the means of the distributions of the classes resemble each other more if the NN is uncertain. Thus, the distributions belonging to these means are more likely to overlap, especially if an algorithm is used that compares adjacent classes. That is why the adjacent algorithms have higher redundancy rates and averages lengths, although they do not yield significantly higher accuracy rates. Conclusively, we state that top algorithms provide better results than adjacent algorithms.

Last but not least, it should be noted that the two datasets used are relatively small, with 60,000 training images and 10,000 test images each. Nevertheless, as already mentioned, the results differ to a remarkable degree for some algorithms. To draw more general conclusions it is necessary to examine further and more complex datasets like ImageNet.

7. Conclusion

In conclusion, we state that *uncertainty aware* top-k metrics seem to be a useful tool to measure performance of NNs. All of the examined algorithms outperform the classic static top-k approach in the ARP. This paradigm is not flawless but seems to be a first approach to compare different *uncertainty aware* top-k algorithms. Especially, how to include a third dimension dealing with false classified images should be discussed. Simple top-x is the cheapest algorithm but nevertheless appears to be valid approach with good results. At the same time, due to the necessity of defining a threshold it runs the risk of manipulating the performance measuring of a NN. In LA-overlap, this risk does not exist as it deals with critical overlaps. However, this algorithms is computationally very costly due to sampling. The LB algorithms try to reduce the cost of sampling through an analytical solution of the posterior. Since the results of LB and LA-overlap differ by a far amount we must state that the calculated distributions do not represent a good approximation of the true distribution in the population. Of course, the conclusions drawn here should be taken with caution, since only two relatively small datasets were studied. In order to draw more general conclusions, the results found here should be validated in experiments with larger and more challenging datasets such as ImageNet.

A. MNIST Results

Static Top-k

CNN-full

k	10	9	8	7	6	5	4	3	2	1
Accuracy	1.00	1.00	1.00	1.00	1.00	1.00	0.9998	0.9989	0.9967	0.9816
Average length	10.00	9.00	8.00	7.00	6.00	5.00	4.00	3.00	2.00	1.00

CNN-3k

k	10	9	8	7	6	5	4	3	2	1
Accuracy	1.00	1.00	0.9996	0.9994	0.9976	0.9958	0.9915	0.9805	0.9574	0.8730
Average length	10.00	9.00	8.00	7.00	6.00	5.00	4.00	3.00	2.00	1.00

CNN-2k

k	10	9	8	7	6	5	4	3	2	1
Accuracy	1.00	0.9991	0.9970	0.9944	0.9870	0.9762	0.9575	0.9265	0.8549	0.7014
Average length	10.00	9.00	8.00	7.00	6.00	5.00	4.00	3.00	2.00	1.00

CNN-1k

k	10	9	8	7	6	5	4	3	2	1
Accuracy	1.00	0.9731	0.8799	0.8178	0.7710	0.7339	0.7084	0.6759	0.5961	0.4093
Average length	10.00	9.00	8.00	7.00	6.00	5.00	4.00	3.00	2.00	1.00

Simple Top-X

CNN-full

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	0.9996	0.9981	0.9970	0.9952	0.9926	0.9898	0.9871	0.9830	0.9817	0.9816
Average length	1.5457	1.1493	1.1077	1.0677	1.0383	1.0233	1.0142	1.0038	1.0001	1.00

CNN-3k

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	0.9993	0.9905	0.9860	0.9757	0.9594	0.9414	0.9205	0.8963	0.8731	0.8730
Average length	4.904400	2.5674	2.1684	1.7363	1.4016	1.2512	1.1518	1.0744	1.0003	1.00

CNN-2k

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	0.9985	0.9890	0.9813	0.9645	0.9275	0.8930	0.8564	0.8085	0.7042	0.7014
Average length	7.8996	5.3359	4.6263	3.6174	2.6049	2.044700	1.6917	1.4190	1.0144	1.00

CNN-1k

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	1.00	1.00	1.00	0.9731	0.8799	0.8178	0.7710	0.7339	0.6732	0.4093
Average length	10.00	10.00	10.00	9.00	8.00	7.00	6.00	5.00	2.9448	1.00

LA-mean

CNN-full

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	0.9997	0.9984	0.9973	0.9957	0.9928	0.9902	0.9877	0.9830	0.9816	0.9815
Average length	1.5849	1.1615	1.1149	1.0723	1.0410	1.0248	1.0156	1.0040	1.0001	1.00

CNN-3k

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	0.9993	0.9921	0.9871	0.9777	0.9621	0.9447	0.9242	0.8981	0.8729	0.8727
Average length	5.1001	2.6895	2.2704	1.7973	1.4344	1.2709	1.1654	1.0807	1.0005	1.0000

CNN-2k

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	0.9985	0.9891	0.9817	0.9661	0.9300	0.8966	0.8601	0.8148	0.7041	0.7016
Average length	7.9709	5.4417	4.7274	3.7042	2.6768	2.0968	1.7248	1.4404	1.0148	1.0000

CNN-1k

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	1.00	1.00	1.00	0.9731	0.8803	0.8174	0.7704	0.7339	0.6727	0.4097
Average length	10.00	10.00	10.00	9.00	8.00	7.00	6.00	5.00	2.9452	1.00

LA-overlap adjacent

CNN-full

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9940	0.9917	0.9907	0.9898	0.9882	0.9867	0.9843	0.9815	0.9815	0.9815
Average length	1.1418	1.0493	1.0355	1.0269	1.0193	1.0115	1.0060	1.00	1.00	1.00

CNN-3k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9816	0.9666	0.9578	0.9476	0.9344	0.9154	0.8993	0.8739	0.8731	0.8731
Average length	3.7304	2.3479	1.7825	1.4595	1.2551	1.1296	1.0731	1.0013	1.00	1.00

CNN-2k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9835	0.9573	0.9355	0.9071	0.8679	0.8113	0.7690	0.7029	0.7019	0.7019
Average length	7.7022	5.4235	4.0417	2.9064	2.0160	1.4473	1.2242	1.0027	1.00	1.00

CNN-1k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	1.00	1.00	1.00	0.9979	0.9704	0.8282	0.6591	0.4104	0.4086	0.4086
Average length	10.00	9.9994	9.9322	9.5448	8.5483	6.1015	3.4194	1.0085	1.00	1.00

LA-overlap top

CNN-full

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9932	0.9912	0.9904	0.9897	0.9881	0.9867	0.9842	0.9815	0.9815	0.9815
Average length	1.0488	1.0309	1.0269	1.0230	1.0167	1.0113	1.0059	1.00	1.00	1.00

CNN-3k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9695	0.9563	0.9484	0.9392	0.9283	0.9129	0.8986	0.8731	0.8724	0.8724
Average length	1.5947	1.3702	1.2891	1.2339	1.1739	1.1121	1.0666	1.0011	1.00	1.00

CNN-2k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9402	0.9066	0.8865	0.8688	0.8394	0.7979	0.7646	0.7030	0.7020	0.7020
Average length	2.8679	2.1822	1.9383	1.7420	1.5381	1.3285	1.1946	1.0025	1.00	1.00

CNN-1k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9825	0.9085	0.8569	0.8073	0.7457	0.6390	0.5562	0.4111	0.4091	0.4091
Average length	8.5600	7.2600	6.3965	5.5279	4.3530	2.8841	1.9799	1.0089	1.00	1.00

LB adjacent

CNN-full

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9824	0.9823	0.9822	0.9822	0.9821	0.9820	0.9819	0.9816	0.9816	0.9816
Average length	1.0048	1.0038	1.0037	1.0035	1.0019	1.0006	1.0004	1.00	1.00	1.00

CNN-3k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9025	0.8959	0.8936	0.8901	0.8866	0.8819	0.8781	0.8729	0.8729	0.8729
Average length	1.2549	1.1692	1.1313	1.0969	1.0665	1.0382	1.0177	1.00	1.00	1.00

CNN-2k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.8682	0.8405	0.8243	0.8069	0.7881	0.7613	0.7380	0.7013	0.7013	0.7013
Average length	3.2977	2.6930	2.3726	2.0841	1.7853	1.4225	1.2001	1.00	1.00	1.00

CNN-1k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.4091	0.4091	0.4091
Average length	10.00	10.00	10.00	10.00	10.00	10.00	10.00	1.00	1.00	1.00

LB top

CNN-full

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.9823	0.9823	0.9822	0.9822	0.9821	0.9820	0.9819	0.9816	0.9816	0.9816
Average length	1.0026	1.0020	1.0017	1.0014	1.0008	1.0006	1.0004	1.00	1.00	1.00

CNN-3k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.8991	0.8938	0.8912	0.8883	0.8847	0.8810	0.8779	0.8729	0.8729	0.8729
Average length	1.1223	1.0835	1.0689	1.0548	1.0391	1.0234	1.0134	1.00	1.00	1.00

CNN-2k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.8468	0.8185	0.8050	0.7909	0.7751	0.7508	0.7325	0.7013	0.7013	0.7013
Average length	2.1996	1.8189	1.6423	1.4880	1.3363	1.1899	1.1071	1.00	1.00	1.00

CNN-1k

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.4091	0.4091	0.4091
Average length	10.00	10.00	10.00	10.00	10.00	10.00	9.9321	1.00	1.00	1.00

B. CIFAR-100 Results

Static Top-k

k	1	2	3	4	5	7	10	12	15	20
Accuracy	0.7623	0.8600	0.8969	0.9188	0.9324	0.9501	0.9641	0.9720	0.9783	0.9857
Average length	1.00	2.00	3.00	4.00	5.00	7.00	10.00	12.00	15.00	20.00

Simple Top-X

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	0.9791	0.9522	0.9381	0.9132	0.8764	0.8506	0.8248	0.8021	0.7687	0.7625
Average length	11.1585	6.3517	5.2861	3.8568	2.5477	1.9014	1.5356	1.3043	1.0442	1.0011

LA-mean

Threshold	0.997	0.97	0.95	0.9	0.8	0.7	0.6	0.5	0.25	0.1
Accuracy	0.9795	0.9543	0.9414	0.9174	0.8796	0.8527	0.8262	0.8043	0.7689	0.7623
Average length	11.3753	6.4955	5.4026	3.9362	2.5890	1.9266	1.5503	1.3151	1.0451	1.0011

LA-overlap adjacent

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.8886	0.8630	0.8474	0.8309	0.8128	0.7923	0.7785	0.7623	0.7621	0.7621
Average length	4.2084	3.0251	2.4913	2.0526	1.5937	1.1949	1.0686	1.001	1.00	1.00

LA-overlap top

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.8441	0.8263	0.8162	0.8071	0.7982	0.7871	0.7775	0.7623	0.7621	0.7621
Average length	1.5386	1.3471	1.2769	1.2206	1.1604	1.0966	1.0563	1.001	1.00	1.00

LB adjacent

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.8327	0.8202	0.8131	0.8067	0.7998	0.7884	0.7782	0.7623	0.7623	0.7623
Average length	2.8501	2.4596	2.2338	2.0585	1.8229	1.5108	1.2487	1.00	1.00	1.00

LB top

Overlap	0.001	0.01	0.025	0.05	0.10	0.20	0.30	0.50	0.75	0.90
Accuracy	0.8181	0.8078	0.8011	0.7943	0.7876	0.7804	0.7735	0.7623	0.7623	0.7623
Average length	2.0838	1.7921	1.6357	1.5041	1.3300	1.1820	1.0801	1.00	1.00	1.00

Bibliography

Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJlrF24twB`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Marius Hobbhahn, Agustinus Kristiadi, and Philipp Hennig. Fast predictive uncertainty for classification with bayesian deep networks. *CoRR*, abs/2003.01227, 2020. URL `https://arxiv.org/abs/2003.01227`.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*, pages 5436–5446. PMLR, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.