Distance-based Confidence Score for Neural Network Classifiers

By

AMIT MANDELBAUM

THE HEBREW UNIVERSITY OF JERUSALEM

Faculty of Computer Science and Engineering
THE HEBREW UNIVERSITY OF JERUSALEM

A dissertation submitted to the Hebrew University of Jerusalem as a partial fulfillment of the requirements of the degree of MASTER OF SCIENCE in the Faculty of Computer Science and Engineering.

OCTOBER 2017

Word count: nine thousand eight hundred and fifty six

Abstract

Measuring the confidence of a classifier in its predictions is very important for many applications, and is therefore an important part of the classifier design. Yet, although deep learning has received tremendous attention in recent years, not much progress has been made in quantifying the prediction confidence of neural network classifiers. Bayesian models offer a mathematically grounded framework to reason about model uncertainty, however, Bayesian neural networks usually come with prohibitive computational costs. To this day, the most commonly used confidence scores are the simple *Max Margin*, which is the strength of the most activated output unit followed by a softmax normalization, or the (negative) entropy of this softmax output. Some recent works proposed ways to improve the confidence score provided by the entropy, but they come with a price of a highly increased train or test time.

In this work we propose a new confidence score based on the estimation of local density as induced by the network, when points are represented using the effective embedding created by the trained network in its penultimate layer. Our work is motivated by the empirical observation concerning neural networks trained for classification, which have been shown to demonstrate in parallel useful embedding properties. We note, however, that the commonly used embedding discussed above is associated with a network trained for classification only, which may impede its suitability to measure confidence reliably. In accordance, we propose two methods for improving the probabilistic interpretation of the embeddings. In the first method we modify the network loss function and add a term which penalizes for the violation of pairwise constraint. In the second method we use Adversarial Training, which is designed to improve the measurement of uncertainty, but is surprisingly helpful for achieving suitable embeddings as well.

In this work we show that our proposed confidence score is both simple to implement and scalable while also being much cheaper in terms of train and test time, compared to other recently proposed methods. We also test our method on several tasks which use a confidence score. These tasks include correct classification prediction, self-training, ensemble methods and novelty detection. In all tasks we show significant improvement over traditional, commonly used confidence scores even when those are improved with the methods mentioned above.

DEDICATION AND ACKNOWLEDGEMENTS

I stepped into the wonderful area of Deep Learning almost by accident. As a first year, electro-optics Ms.c. student I heard about this new course on Deep Learning, an area which I was unaware of at that time, and decided, just out of curiosity, to sign in. Two and a half years and many coded neural networks later I must say that I feel blessed. Blessed to be so intrigued and immersed in this such and immensely growing field. Blessed to wake up in the morning and go to sleep thinking about new ideas and directions in this field of science. Blessed to be a part of this field not because of its hype but rather from pure interest and the joy of new discoveries.

In line of all the written above I would like to thank, first and foremost, God, who have shown me this path and who gave me the courage to stop the path I was going in and start a new Computer Science Ms.c. Degree in a new place and with a new supervisor.

Secondly, I would like to thank wholeheartedly my supervisor Prof. Daphna Weinshall. Daphna agreed to take me as her student even though I was coming from a different university, a different field and she did not know me or my capabilities as a student. Our weekly meeting, the talks which were not always directly concerned with the current research were always inspiring. I learned so much from you and feel very grateful for your careful guidance and so much help with my research and whole the related environment and eco-system, including (and of course not limited to) the generous stipend.

I would like to thank all my research friends in the university for always being open to share, hear and help me think about new ideas and directions, I and already miss you.

Finally, I would like to thank me beloved family. My wife Alexandra who was always there to support me during the long nights and never ending work that is associated with being a researcher. Alexandra, who although is not from the field, was always listening and giving me new ideas. Alexandra, for your endless love and support, I could not imagine doing this without you. Last, but certainly not least, my kids, Adiel and Ofer, I thank you for being my drive force, my strength and my joy every day and night.

Table of Contents

List of Tables

LIST OF FIGURES

Chapter 1

Introduction

Classification confidence scores are designed to measure the accuracy of the model when predicting class assignment (rather than the uncertainty inherent in the data). Most generative classification models are probabilistic in nature, and therefore provide such confidence scores directly. Most discriminative models, on the other hand, do not have direct access to the probability of each prediction. Instead, related non-probabilistic scores are used as proxies, as for example the margin in SVM classifiers.

When trying to evaluate the confidence of neural network (NN) classifiers, a number of scores are commonly used. One is the strength of the most activated output unit followed by softmax normalization, or the closely related ratio between the activities of the strongest and second strongest units. Another is the (negative) entropy of the output units, which is minimal when all units are equally probable. Often, however, these scores do not provide a reliable measure of confidence, see [2, 15, 17, 21].

Why is it important to reliably measure prediction confidence? In various contexts such as medical diagnosis and decision support systems, it is important to know the prediction confidence in order to decide how to act upon it. For example, if the confidence in a certain prediction is too low, the involvement of a human expert in the decision process may be called for. Another important aspect of real world applications is the ability to recognize samples that do not belong to any of the known classes, which can also be improved with a reliable confidence score. More generally and importantly, estimating when a model is in error is of great concern to AI Safety [2]. But even independently of the application context, reliable prediction confidence can be used to boost the classifier performance via such methods as self-training or ensemble classification. In this context a better confidence score can improve the final performance of the classifier. The derivation of a good confidence score should therefore be a part of the classifier design, as important as any other component of classifiers' design.

In order to derive a reliable confidence score for NN classifiers, we focus our attention on the empirical observation concerning neural networks trained for classification, which have been shown to demonstrate in parallel useful embedding properties. Specifically, a common practice these days is to treat one of the upstream layers of a pre-trained network as a representation (or embedding) layer. This layer activation is then used for representing similar objects and train simpler classifiers (such as SVM, or shallower NNs) to perform different tasks, related but not identical to the original task the network had been trained on.

In computer vision such embeddings are commonly obtained by training a deep network on the recognition of a very large database (typically ImageNet [10]). These embeddings have been shown to provide better semantic representations of images (as compared to more traditional image features) in a number of related tasks, including the classification of small datasets [47], image annotation [12] and structured predictions [23]. Given this semantic representation, one can compute a natural multi-class probability distribution as described in Section 3.1, by estimating local density in the embedding space. This estimated density can be used to assign a confidence score to each test point, using its likelihood to belong to the assigned class.

We note, however, that the commonly used embedding discussed above is associated with a network trained for classification only, which may impede its suitability to measure confidence reliably. In fact, when training neural networks, metric learning is often used to achieve desirable embeddings (e.g., [22, 46, 50, 54]). Since our goal is to improve the probabilistic interpretation of the embedding, which is essentially based on local point density estimation (or the distance between points), we may wish to modify the loss function and add a term which penalizes for the violation of pairwise constraints as in [18]. Our experiments show that the modified network indeed produces a better confidence score, with comparable classification performance. Surprisingly, while not directly designed for this purpose, we show that networks which are trained with adversarial examples following the Adversarial Training paradigm [15, 49], also provide a suitable embedding for the new confidence score.

Our first contribution, therefore, is a new prediction confidence score which is based on local density estimation in the embedding space of the neural network. This score can be computed for every network, but in order for this score to achieve superior performance, it is necessary to slightly change the training procedure. In our second contribution we show that suitable embedding can be achieved by either augmenting the loss function of the trained network with a term which penalizes for distance-based similarity loss (as in Eq. (3.2) below), or by using Adversarial Training. The importance of the latter contribution is two fold: Firstly, we are the first to show that the density of image embeddings is improved with indirect Adversarial Training perturbations, in addition to the improved word embedding quality shown in [36] by direct Adversarial Training perturbations. Secondly, we show in Section 4 that Adversarial Training improves the results while imposing a much lighter burden of hyper-parameters to tune as compared to the distance-based loss.

The new confidence score is evaluated in comparison to other scores, using the following tasks: (i) Performance in the binary classification task of identifying each class prediction as correct or incorrect (Section 3.1). (ii) Ranking unknown points for semi-supervised self-training, in order to improve performance of the final classifier (Section 3.4) (iii) Training an ensemble of NN classifiers, where each classifier's prediction is weighted by the new confidence score (Section 3.5). (iv) Novelty detection, where confidence is used to predict whether a test point belongs to one of the known classes from the train set (Section 3.6).

The empirical evaluation of our method is described in Section 4, using a few datasets and different network architectures which have been used in previous work when using these specific datasets. Our method achieves significant improvement in all 4 tasks. When compared with two more recent methods shown to improve traditional measures of classification confidence, MC dropout [13] and Adversarial Training [49], our method achieves better results while also maintaining lower computational costs.

Finally, we also describe and analyze another hybrid architecture based on the use of 2 clone neural network classifiers. This contribution is most relevant in the context of ensemble classification, where multiple networks are trained. The basic architecture is based on a pair of identical networks trained with two related loss functions: the "regular" cross entropy loss for one network, and a linear combination of the cross-entropy loss and a margin-based similarity loss for the second network. The embedding obtained by the second network is used to compute the confidence score for the final classifier. A pair of classifiers thus constructed is shown to provide a more reliable measure of prediction confidence as compared to a single classifier. Ensembles which are based on pairs of classifiers constructed in this manner achieve better performance than ensembles composed of the same number of un-paired classifiers.

Chapter 2

Related Work

In the next few pages we describe notable works which are related to the topics which are discussed through our work.

2.1 Confidence Score for Neural Network Classifiers

The Bayesian approach learns a posterior distribution over the parameters of the neural network and uses it to estimate prediction uncertainty, as in [32, 37]. However, Bayesian neural networks are not always practical to implement, and the computational cost involved is typically high. This raises the need for a solution that can improve the classifier's confidence score and yet require only minor modifications in the training procedure.

In accordance, [13] showed that a neural network, with dropout applied before every weight layer, is mathematically equivalent to an approximation of the probabilistic deep Gaussian process [9]. Using this probabilistic interpretation, the authors proposed to use dropout during test time in a method referred below as *MC-Dropout*. Using this method, a cheap proxy to Bayesian Neural Networks is achieved by running many stochastic forward passes through the network and averaging the results, providing an improved uncertainty measure of the network's predictions.

Still, the most basic confidence scores for neural networks can be derived from the strength of the most activated output unit (also called *softmax input*). In doing so we assume that our model is reasonably good in approximating the real class posterior probabilities:

$$D_0(x) = max_k \hat{p}(C = k|x) \tag{2.1}$$

Note that the network's output, $\hat{p}(k|x)$, are not however true probabilities, since there is no guarantee that their sum be one. A probability distribution may be trivially obtained by normalizing the outputs. A more statistically meaningful score would therefore be the the normalized version of the output layer (also called *softmax output* or *max margin*):

$$D_1(x) = max_k p(\hat{k}|x) = max_k \frac{\hat{p}(k|x)}{\sum_{j=1}^{m} \hat{p}(j|x)} \tag{2.2}$$

However, these scores are an approximation of the true probability of good classification when the class chosen is most probably correct, but not otherwise. They do not tell us the confidence on the choice of class. Despite its simplicity, this equation provides a confidence score which is still one of the most commonly used today.

A confidence score that handles better a situation where there is no one class which is most probable, is the (negative) entropy of the normalized network's output [53]:

$$D_2(x) = -H(p(\hat{k}|x)) = \sum_{j=1}^{m} p(\hat{j}|x) \log p(\hat{j}|x) \tag{2.3}$$

Where $p(\hat{j}|x)$ equals to the softmax output. These scores, as well as some more complex ones (e.g. [51]), were compared in [56], which (somewhat surprisingly) demonstrated the empirical superiority of the two most basic methods ($D_1(x)$ and $D_2(x)$) described above.

2.2 Adversarial Training

Adversarial examples are points of data which are "close" to the original training samples (e.g. an image that is visually indistinguishable from the original image to humans), but are misclassified, usually with high confidence, by the neural networks. This concept was first introduced in [49] and later extended in [15], where authors noted that these examples could be used by a potential attacker in order to mislead the neural network, thus creating a possible security breach [2]. [15] suggested a brute force solution of generating adversarial examples during training and explicitly training the model not to be fooled by each of them. This training method is therefore called *Adversarial Training* and the method of generating those examples is described in Section 3.1.

In a more recent work, [29] proposed to use adversarial training in order to improve the uncertainty measure of the entropy score of the neural network. The authors suggested that Adversarial Training can be interpreted as a computationally efficient solution to smooth the predictive distributions by increasing the likelihood of the target around some neighborhood of the observed training samples. This smoothness is, in turn, contributing for both the classifier's robustness and its predictive uncertainty estimation.

Finally, in the context of embeddings, [36] showed that applying Adversarial Training directly on word embeddings can improve their quality. However, this improvement comes from a direct perturbation of the embeddings which is different from the indirect perturbation performed in our work.

2.3 Self Training

Self training is a common bootstrapping procedure which relies on the availability of a reliable confidence score for each classification prediction (see details in Section 3.4). This method has been used to improve the performance of classifiers, often in the context of semi-supervised learning and was used in a wide range of application domains, from NLP to computer vision [8, 35, 44].

In one of the most notable and early works, [55] used a classifier in an iterative way, where in each step the training data is provided by the classifier's predictions in the previous step. This approach was also used successfully in semi-supervised learning with neural networks, where the classifier is using its own predictions for training data with missing labels [30] and classification with noisy labels [43]. Specifically, the classifier uses its own predictions to change the given labels of the training data.

2.4 Ensemble Methods

Ensembles of models have been used to improve the overall performance of the final classifier (see reviews in [11, 31]). There are many ways to train an ensemble, such as boosting or bagging. In the context of this work we address the kind of ensemble which is made when using different training parameters with a single training method.

After the ensemble of networks is trained, there are also many ways to integrate the predictions of the classifiers in the ensemble, including the average prediction or voting [4]. Some ensemble methods use the confidence score to either weight the predictions of the different classifiers (average weighting), or for confidence voting where the most confident networks gets more (or even all) votes when classifying.

2.4.1 Novelty Detection

Novelty detection, where the task is to determine whether a test point belongs to a known class label or not, is another problem which becomes more relevant with the ever increasing availability of very large datasets, see reviews in [33, 41] and the recent work in [52]. We also note that novelty *detection* is quite different from the learning of classes with no examples, as in zero shot learning [40].

The ability to differentiate between known and unknown test samples is one of the fundamental requirements of a good classifier. Since classifier can never be trained on all possible classes, the performance of the network will be poor for those classes that are under-represented in the training set (see [34] for a review). Moreover, neural networks can make overconfident predictions on unseen samples, even for completely unrecognizable samples [39]. This poses a real problem for deploying neural networks in real world environments, where the classifier is usually exposed to many samples which do not belong to any of the classes seen during training. Ideally, we would like the predictions to exhibit higher uncertainty when the test data is very different from the training data. This difference in uncertainty can provide us with the ability to differentiate between known and unknown samples during testing. Recently, [21] showed that the max-margin score can be used as a good baseline for novelty detection. In our experiment we show that our confidence score gains significantly better results on this task compared to the proposed baseline.

Chapter 3

New Confidence Score

We describe next the new proposed confidence score. We then discuss how it can be used to boost classification performance with self training and ensemble methods, or when dealing with novelty detection.

3.1 New Confidence Score for Neural Network Classifiers

The main motivation behind the proposed confidence score is that, when trained properly, a neural network creates an embedding space in which points that belong to the same class will be close to each other. This effectively divides the training data into clusters of points from the same class. We can now use these clusters implicitly and assume that test points that are predicted correctly will be close or even within the cluster of training data of the same class, while misclassified points will be between clusters or even within a cluster of an entirely different class.

Our confidence score is therefore based on the estimation of local density as induced by the network, when points are represented using the effective embedding created by the trained network in its penultimate layer. Local density at a point is estimated based on the Euclidean distance in the embedded space between the point and its k nearest neighbors in the training set.

Specifically, let $f(x)$ denote the embedding of x as defined by the trained neural network classifier. Let $A(x) = \{x_{train}^{j}\}_{j=1}^{k}$ denote the set of k-nearest neighbors of x in the training set, based on the Euclidean distance in the embedded space, and let $\{y^{j}\}_{j=1}^{k}$ denote the corresponding class labels of the points in $A(x)$. A probability space is constructed (as is customary) by assuming that the likelihood that two points belong to the same class is proportional to the exponential of the negative Euclidean distance between them. In accordance, the local probability that a point x belongs to class c is proportional to the probability that it belongs to the same class as the subset of points in $A(x)$ that belong to class c.

Based on this local probability, the confidence score $D_3(x^i_{test})$ for the assignment of point x^i_{test} to class $c = \hat{y}^i$ is defined as follows:

$$D_3(x^i_{test}) = \frac{\sum_{j=1, y^j=\hat{y}^i}^{k} e^{-||f(x^i_{test}) - f(x^j_{train})||_2}}{\sum_{j=1}^{k} e^{-||f(x^i_{test}) - f(x^j_{train})||_2}} \tag{3.1}$$

(3.1) returns a number between 0 to 1 such that the higher the local density of similarly labeled train points, the higher the score is. Henceforth (3.1) is referred to as *Distance score*.[1] We note here that while intuitively it might be beneficial to add a scaling factor to the distance in (3.1), such as the mean distance, we found it to have a deteriorating effect. This finding is in line with other works (e.g. [45]) which used similar equations for similarity with no scaling.

As mentioned is Section 1, in order to achieve an effective embedding it helps to modify somewhat the training procedure of the neural network classifier. The simplest, and most straightforward, modification is to augment the network's loss function during training with an additional term. The resulting loss function is a linear combination of two terms, one for classification denoted $\mathscr{L}_{class}(X,Y)$, and another pair-wise distance based loss for the embedding denoted $\mathscr{L}_{dist}(X,Y)$. This is defined as follows:

$$\mathscr{L}(X,Y) = \mathscr{L}_{class}(X,Y) + \alpha \mathscr{L}_{dist}(X,Y) \tag{3.2}$$

where

$$\mathscr{L}_{dist}(X,Y) = \frac{1}{P} \sum_{p=1}^{P} L_{dist}(x^{p_1}, x^{p_2}) \tag{3.3}$$

$$L_{dist}(x^i, x^j) = \begin{cases} ||f(x^i) - f(x^j)||_2 & \text{if } y^i = y^j \\ max\{0, (m - ||f(x^i) - f(x^j)||_2)\} & \text{if } y^i \neq y^j \end{cases}$$

$\mathscr{L}_{dist}$ is defined by all pairs of points, denoted (x^{p_1}, x^{p_2}). For each training minibatch, this set is sampled with no replacement from the training points in the minibatch, with half as many pairs as the size of the minibatch. In our experiments, $\mathscr{L}_{class}(X,Y)$ is the regular cross entropy loss. We note here that we also tried distance-based loss functions which do not limit the distance between points of the same class to be exactly 0 (such as those in [22] and [50]). However, those functions produced worse results, especially when the dataset had many classes.

Surprisingly, a desirable embedding can also be achieved by Adversarial Training, using the *fast gradient method* suggested in [15]. In this method, given an input x with target y, and a neural network with parameters θ, adversarial examples are generated using:

$$x' = x + \epsilon \; sign(\nabla_x L_{class}(\theta, x, y)) \tag{3.4}$$

[1]Related measures of density, such as a simple count of the "correct" neighbors or the inverse of the distance, behave similarly and perform comparably though slightly less well.

In each step an adversarial example is generated for each point x in the batch and the current parameters of the network, and classification loss is minimized for both the regular and adversarial examples. Although originally designed to improve robustness, this method seems to improve the network's embedding for the purpose of density estimation, possibly because along the way it increases the distance between pairs of adjacent points with different labels. Finally, while this method can increase (and even double) the network's training time, it has the advantage of having fewer hyper parameters as compared to the distance-based loss. We note that we have tried using the distance-based loss and adversarial training together while training the network, but this procedure lead to a deterioration of the results.

3.2 Alternative Confidence Scores

As described in Section 2.1, given a trained network, two measure are usually used to evaluate classification confidence:

Max margin: the maximal activation, after normalization, in the output layer of the network.

Entropy: the (negative) entropy of the activations in the output layer of the network.

As also noted in Section 2.1, the empirical study in [56] showed that these two measures are typically as good as any other existing method for the evaluation of classification confidence.

Two recent methods have been shown to improve the reliability of the confidence score based on Entropy: *MC-Dropout* [13] and *Adversarial Training* [15] as it was used in [29] to improve the entropy score. In terms of computational cost, adversarial training increases (and sometimes doubles) the training time, due to the computation of additional gradients and the addition of the adversarial examples to the training set. MC-Dropout, on the other hand, does not change the training time but increases the test time by orders of magnitude (typically 100-fold). Both methods are complementary to our approach, in that they focus on modifications to the actual computation of the network during either train or test time. After all is done, they both evaluate confidence using the Entropy score. As we show in our experiments, adversarial training combined with our proposed confidence score improves the final results significantly.

3.3 Our method: Comparative Computational Analysis

Unlike the two methods described above, *MC-Dropout* and *Adversarial Training*, our method (or confidence score) takes an existing network and computes a new confidence score from the network's embedding and output activation. It can use any network, with or without adversarial training or MC dropout. If the loss function of the network is suitably augmented (see discussion above), empirical results in Section 4 show that our score always improves results over the Entropy score of the given network.

Train and test computational complexity: Considering the distance-based loss, [50] showed that computing distances during the training of neural networks have negligible effect on training time. Alternatively, when using adversarial training, additional computational cost is incurred as mentioned above, while on the other hand fewer hyper parameters are left for tuning. During test time, our method requires carrying over the embeddings of the training data and also the computation of the k nearest neighbors for each sample.

Nearest neighbor classification has been studied extensively in the past 50 years, and consequently there are many methods to perform either precise or approximate k-nn with reduced time and space complexity (see [16] for a recent empirical comparison of the main methods). In our experiments, while using either Condensed Nearest Neighbours [19] or Density Preserving Sampling [5], we were able to reduce the memory requirements of the train set to 5% of its original size without affecting performance. At this point the additional storage required for the nearest neighbor step was much smaller than the size of the networks used for classification, and the increase in space complexity became insignificant.

With regards to time complexity, recent studies have shown how modern GPU's can be used to speed up nearest neighbor computation by orders of magnitude [3, 14]. [25] also showed that k-nn approximation with 99% recall can be accomplished 10-100 times faster as compared to precise k-nn.

Combining such reductions in both space and time, we note that even for a very large dataset, including for example 1M images embedded in a 1K dimensional space, the computation complexity of the k nearest neighbors for each test sample requires at most 5M floating-point operations. This is comparable and even much faster than a single forward run of this test sample through a modern, relatively small, ResNets [20] with 2-30M parameters. Thus, our method scales well even for very large datasets.

3.4 Self Training

In self training, a certain confidence score is used to rank all test samples in descending order, at which point the K top ranked test samples are extracted. After adding the extracted K test samples to the train set, and using the corresponding labels predicted by the network from which we took the confidence score, a regular classification network is retrained from scratch. To evaluate performance, we score the accuracy of the retrained classification network on the entire test data, including the samples added to the train set (since their true labels remains unknown).

3.5 Ensembles of Classifiers

There are many ways to define ensembles of classifiers, and different ways to put them together. Here we focus on ensembles which are obtained when using different training parameters with a single training method. This specifically means that we train several neural networks using random initialization of the network parameters, along with random shuffling of the train points.

Henceforth *Regular Networks* will refer to networks that were trained only for classification with the regular cross-entropy loss, *Distance Networks* will refer to networks that were trained with the loss function defined in (3.2), and *AT Networks* will refer to networks that were trained with adversarial examples as defined in (3.4).

Ensemble methods differ in how they weigh the predictions of different classifiers in the ensemble. A number of options are in common use (see [31] for a recent review), and in accordance are used for comparison in the experimental evaluation section: 1) softmax average, 2) simple voting, 3) weighted softmax average (where each softmax vector is multiplied by its confidence score), 4) confidence voting (where the most confident network gets $\frac{n}{2}$ votes), and 5) dictator voting (the decision of the most confident network prevails). We evaluated methods $3-5$ with weights defined by either the Entropy score or the Distance score defined in (3.1).

3.6 Novelty Detection

Novelty detection seeks to identify points in the test set which belong to classes unseen in the train set. To evaluate performance in this task we train a network with a known benchmark dataset, while augmenting the test set with test points from another dataset that includes different classes. Each confidence score is used to differentiate between known and unknown samples. This is a binary classification task, and therefore we evaluate performance using ROC curves.

CHAPTER 4

EXPERIMENTAL EVALUATION

Here we empirically evaluate the benefits of our proposed approach, comparing the performance of the new confidence score with alternative existing scores in the 4 different tasks described in Chapter 3.

4.1 Experimental Settings

4.1.1 Datasets

For the evaluation we used 3 data sets: CIFAR-100 [28], STL-10 [7] (32×32 version) and SVHN [38]. In all cases, as is commonly done, the data was pre-processed using global contrast normalization (GCN) and ZCA whitening [26]. No other method of data augmentation was used for CIFAR-100 and SVHN, while for SVHN we also did not use the additional $\tilde{5}$00K labeled images[1]. For STL-10 we used cropping and flipping (horizontal and vertical) to check the robustness of our method to heavy data augmentation.

4.1.2 Networks Architecture and Training Parameters

We now describe the neural networks used in the following experiments for the respective datasets, including the hyper parameters used during training. All networks were trained using the TensorFlow environment [1] with Exponential-Linear-Unit activations (ELU) [6] for non-

[1]Note that reported results denoted as "state-of-the-art" for these datasets often involve heavy augmentation. In our study, in order to be able to do the exhaustive comparisons described below, we opted for the un-augmented scenario as more flexible and yet informative enough for the purpose of comparison between different methods. Therefore our numerical results should be compared to empirical studies which used similar *un-augmented settings*. We specifically selected commonly used architectures that achieve good performance, close to the results of modern ResNets, and yet flexible enough for extensive evaluations.

linear activation. The networks were augmented with an L_2 regularizer with weight decay factor of 10^{-4}. In all experiments we used batch size of 100.

CIFAR-100 and STL-10: for both datasets we used the network suggested in [6] with the following architecture:

$C(192,5) \Rightarrow P(2) \Rightarrow C(192,1) \Rightarrow C(240,3) \Rightarrow P(2) \Rightarrow C(240,1) \Rightarrow C(260,3) \Rightarrow P(2) \Rightarrow C(260,1) \Rightarrow C(280,2) \Rightarrow P(2) \Rightarrow C(280,1) \Rightarrow C(300,2) \Rightarrow P(2) \Rightarrow C(300,1) \Rightarrow FC(100)$

$C(n,k)$ denotes a convolution layer with n kernels of size $k \times k$ and stride 1. $P(k)$ denotes a max-pooling layer with window size $k \times k$ and stride 2, and $FC(n)$ denotes a fully connected layer with n output units. For STL-10 the last layer was replaced by FC(10). During training (only) we applied dropout [48] before each max pooling layer (excluding the first) and after the last convolution, with the corresponding drop probabilities of $[0.1, 0.2, 0.3, 0.4, 0.5]$. Note that the last convolution layer output feature map size is 1×1, and thus, after reshaping, it can be treated as an embedding vector of size 300.

We trained the network using Momentum optimizer [42]. For CIFAR-100 training was carried over 80,000 steps and with the following learning rates: [steps 0-30K: 10^{-2}, steps 30K-50K: $5 * 10^{-3}$, steps 50K-65K: $5 * 10^{-4}$, steps 65K-80K: $5 * 10^{-5}$]. For STL-10 training was carried over 16,000 steps only using the following learning rates: [steps 0-6K: 10^{-2}, steps 6K-10K: $5 * 10^{-3}$, steps 10K-13K: $5 * 10^{-4}$, steps 13K-16K: $5 * 10^{-5}$].

SVHN: for this dataset dataset we used the following architecture:

$C(32,3) \Rightarrow C(32,3) \Rightarrow C(32,3) \Rightarrow P(2) \Rightarrow C(64,3) \Rightarrow C(64,3) \Rightarrow C(64,3) \Rightarrow P(2) \Rightarrow C(128,3) \Rightarrow C(128,3) \Rightarrow C(128,3) \Rightarrow P(2) \Rightarrow C(256,3) \Rightarrow C(256,3) \Rightarrow P(2) \Rightarrow FC(256) \Rightarrow FC(10)$

We trained the network using ADAM [27] optimizer for 16K steps with the following learning rates: [steps 0-6K: 10^{-3}, steps 6K-10K: $5 * 10^{-4}$, steps 10K-16K: $5 * 10^{-5}$].

Distance Loss Parameters: For the networks trained with distance loss, for each batch, we randomly picked pairs of points so that at least 20% of the batch included pairs of points from the same class. The margin m in (3.2) was set to 25 all cases and the parameter α in (3.2) was set to 0.2.

Adversarial Training: we used (3.4) following [15], fixing $\epsilon = 0.1$ in all the experiments.

4.1.3 Testing Parameters

Distance Score: for the distance score we observed that the number of k nearest neighbors could be set to the maximum value, which is the number of samples in each class in the train

Table 4.1: AUC Results of Correct Classification.

Confidence Score	CIFAR-100 (Classifier acccuracy: 60%)				STL-10 (Classifier acccuracy: 70.5%)				SVHN (Accuracy: 93.5%)		
	Reg.	Dist.	AT	MCD	Reg.	Dist.	AT	MCD	Reg.	Dist.	AT
Margin	0.828	0.834	0.844	0.836	0.804	0.775	0.812	0.804	0.904	0.896	0.909
Entropy	0.833	0.837	0.851	0.833	0.806	0.786	0.816	0.810	0.916	0.907	0.918
Distance	0.789	**0.853**	0.843	0.726	0.798	0.824	**0.863**	0.671	0.916	**0.925**	**0.925**

Table 4.1, legend. Leftmost column: *Margin* and *Entropy* denote the commonly used confidence scores described in Section 3.2. *Distance* denotes our proposed method described in Section 3.1. Second line: *Reg.* denotes networks trained with the entropy loss, *Dist.* denotes networks trained with the distance loss defined in (3.2), *AT* denotes networks trained with adversarial training as defined in (3.4), and *MCD* denotes MC-Dropout when applied to networks normally trained with the entropy loss. Since the network trained for SVHN was trained without dropout, *MCD* was not applicable.

data. We also observed that smaller numbers (even $k = 50$) often worked as well. In accordance, k was set to 500 (maximum value) in CIFAR-100 and STL-10 and to 50 in SVHN.

In general, the results reported below are not sensitive to the specific values of the hyper-parameters as listed above, with only minor changes occurring when changing the values of k, α and the margin m.

MC-Dropout: as proposed in [13], we used MC dropout in the following manner. We trained each network as usual, but computed the predictions while using dropout during test. This was repeated 100 times for each test example, and the average activation was delivered as output.

4.2 Error Prediction of Multi-class Labels

4.2.1 Single Network

We first compare the performance of our confidence score in the binary task of evaluating whether the network's predicted classification label is correct or not. While our results are independent of the actual accuracy, we note that the accuracy is comparable to those achieved with ResNets when not using augmentation for CIFAR-100 or when using only the regular training data for SVHN (see [24] for example).

Performance in this binary task is evaluated using ROC curves computed separately for each confidence score. Results on all three datasets can be seen in Table 4.1. In all cases our proposed distance score, when computed on a suitably trained network, achieves significant improvement over the alternative scores, even when those are enhanced by using either Adversarial Training or MC-Dropout.

In this part we also tried two more alternative methods: De-normalized Max-Margin (2.1) and applying the Distance score directly on the the Softmax, with or without training the network with the distance loss (3.2) on the Softmax layer itself. Both these methods performed poorly and therefore will not be referenced any further.

4.2.2 Ensemble of Two Networks

To further test our distance score we evaluate performance over an ensemble of two networks. Results are shown in Table 4.2. Here too, the distance score achieves significant improvement over all other methods. We also note that the difference between the distance score computed over *Distance networks* and the entropy score computed over adversarially trained networks, is now much higher as compared to this difference when using only one network. As we show in Section 4.4, adversarial training typically leads to decreased performance when using an ensemble of networks and relying only on the entropy score (probably due to a decrease in variance among the classifiers). This observation further supports the added value of our proposed confidence score.

As a final note, we also used a hybrid architecture using a matched pair of one classification network (of any kind) and a second *Distance network*. The embedding defined by the *Distance network* is used to compute the distance score for the predictions of the first classification network. Surprisingly, this method achieves the best results in both CIFAR-100 and SVHN while being comparable to the best result in STL-10. This method is used later in Section 4.4 to improve accuracy when running an ensemble of networks.

Table 4.2: AUC Results of Correct classification - Ensemble of 2 Networks.

Confidence Score	CIFAR-100			STL-10			SVHN		
	Reg.	Dist.	AT	Reg.	Dist.	AT	Reg.	Dist.	AT
Max margin	0.840	0.846	0.856	0.802	0.792	0.814	0.909	0.901	0.911
Entropy	0.844	0.839	0.857	0.807	0.798	0.816	0.918	0.912	0.920
Distance (1)	0.775	0.863	0.862	0.815	0.834	**0.866**	0.916	0.924	0.926
Distance (2)	0.876	0.872	**0.879**	0.833	0.832	0.859	0.918	**0.929**	0.927

Table 4.2, legend. Notations are similar to those described in the legend of Table 4.1, with one distinction: *Distance (1)* now denotes the regular architecture where the distance score is computed independently for each network in the pair using its own embedding, while *Distance (2)* denotes the hybrid architecture where one network in the pair is fixed to be a *Distance* network, and its embedding is used to compute the distance score for the prediction of the second network in the pair.

4.3 Self-training

Next, we compare performance when using the different confidence scores for self training as described in Section 3.4. Recall that the confidence score is used to rank the test points, where subsequently the highest ranking points are taken with their corresponding labels, obtained from the the same networks which the confidence score was taken from. Next, these points are used to augment the training set of a regular network. This means that while the ranking is done by different networks (Distance, AT or Regular), the part of the re-training is done on the same network.

In the graphs below, we varied the number of test points used for self-training (always taking the top ranking points as determined by each confidence score respectively), and subsequently evaluated accuracy over the entire test data. This procedure was repeated 5 times with different networks, and the standard deviation of the accuracy difference in each repetition was used to measure the statistical significance of the results. For better readability we show only the top 4 performing methods which were used for ranking: a) entropy score on a Regular network, b) Distance score on a Distance network. c) entropy score on an AT network, d) Distance score on an AT network.

CIFAR-100: This dataset contains 50K train samples and 10K test samples. Fig. 4.1 shows the significant advantage of using our proposed confidence score for self training. Surprisingly, while the distance score on the AT network is not the best in terms determining if a classification is correct (as seen in Table 4.1), its ranking of test points produced the best results for the self training task. This might be related to Adversarial Training giving "better" labels to misclassified points, or having better ranking of them.

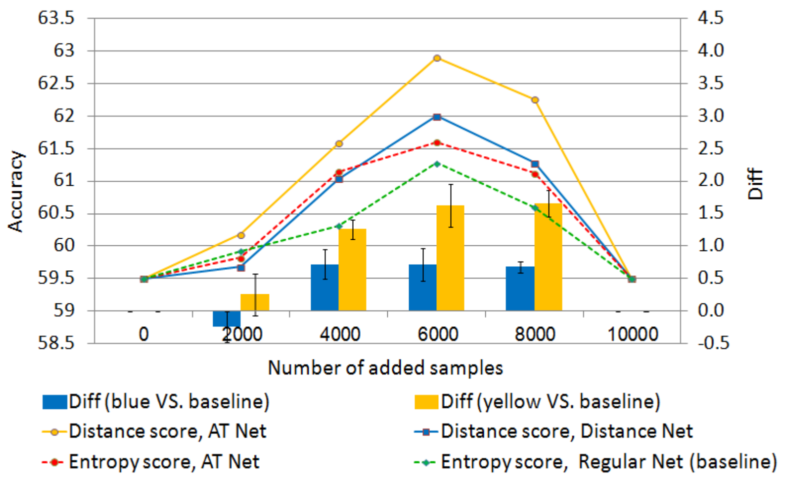

Figure 4.1: Accuracy when using self training with CIFAR 100. The X-axis denotes the number of top ranking test points used for self training. For each method the legend states the scoring method used for grading, and the networks which it was used on. Absolute accuracy is shown for our scoring method and the best alternative method using line plots (corresponding to the left Y-axis). The differences in accuracy between the two methods which uses our score and the baseline method are shown using bar plots (right Y-axis), showing also the standard deviation of the difference over the 5 repetitions.

Since adversarial training typically requires longer training time than using the distance loss

(3.2), we notice that using our distance score together with a *Distance network* also achieves a significant improvement over the baseline method.

In order to further evaluate the performance of self training based on our confidence score and following [44], we computed for each self training session the average number of test points which should have been labeled to begin with (and thus effectively be moved from the set of test points to the set of train points), in order to achieve the same accuracy on the test dataset (inclusive). Results of this analysis are shown in Table 4.3. Thus, in order to achieve the same accuracy as achieved with self-training, we need to "have known" the true labels of about 800 test samples (out of 10,000).

Table 4.3: Average accuracy on CIFAR-100 test set, when using the real labels for a sub-set of the test samples.

Number of samples	500	600	700	800	900
Accuracy (%)	61.31	61.85	62.58	63.05	63.65

STL-10: This dataset contains 5K train samples, which after augmentation becomes 30K samples, 8K test samples, and 100K samples of unlabeled data including images that do not belong to any of the classes in the train set. We can evaluate self-training in two scenarios: in the non-transductive semi-supervised scenario, we rank for self-training only samples from the unlabeled set; in the transductive semi-supervised scenario, we rank for self-training samples from only the test set. Notice that since the number of samples added from the unlabeled dataset is bigger compared to the original train set, we must multiply the training time by 3 in the non-transductive scenario in order to guarantee full convergence.

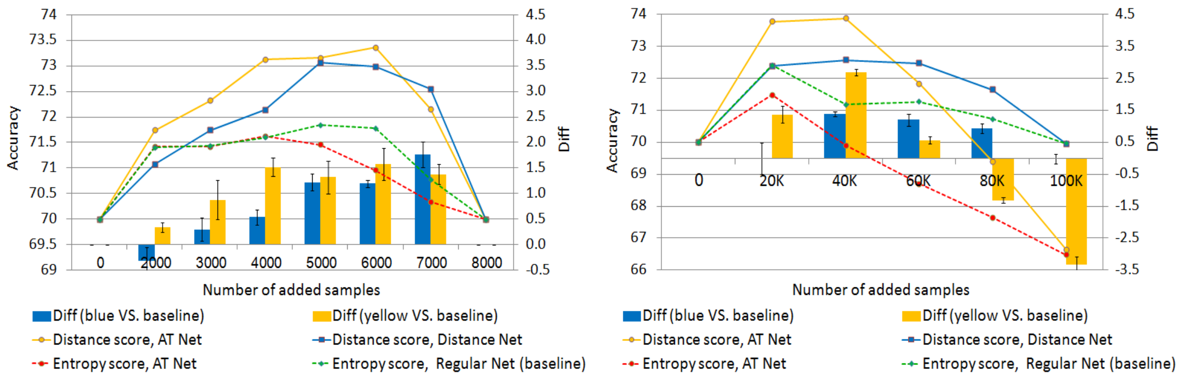

Figure 4.2: Accuracy when using self training with STL-10, see caption of Fig. 4.1. Left: the transductive semi-supervised scenario, using only samples from the test data. Right: the non transductive semi-supervised scenario.

Results are shown in Fig. 4.2. We can see that, like the case with CIFAR-100, the distance based-methods achieve significant improvement over baseline method. However, in the non-transductive scenario the results of the methods based on adversarial training are significantly worse when increasing the number of unlabeled samples beyond a certain value. This might be partially caused due to the lower accuracy of the AT network (68.8%) during training, compared to the accuracy of Regular and Distance networks (70%).

SVHN: This dataset contains roughly 73K train samples and 26K test samples. Results are shown in Fig. 4.3. Similarly to the case in the CIFAR-100 dataset the Distance score computed on the AT network performed significantly better than all other methods, although it was worse (in this case, much worse) in the correct classification task (see Table 4.1). We also note that even without adversarial training, our Distance score achieves consistent (albeit small) improvement over the baseline method.

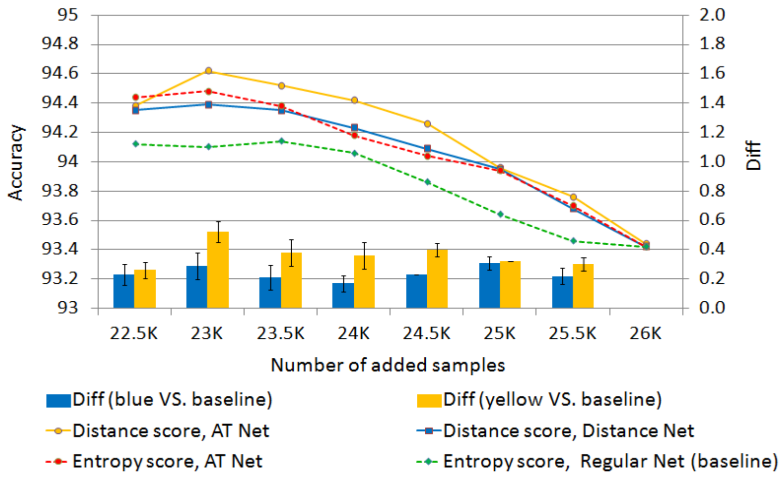

Figure 4.3: Accuracy when using self training with SVHN, see caption of Fig. 4.1.

4.4 Ensemble Methods

In order to evaluate the improvement in performance when using our confidence score to direct the integration of classifiers in an ensemble, we used a few common ways to define the integration procedure, and a few ways to construct the ensemble itself. In all comparisons the number of networks in the ensemble remained fixed at n. Our experiments included the following ensemble compositions: (a) n *regular networks*, (b) n *distance networks*, (c) n *AT (Adversarially Trained) networks*, and (d-f) n networks such that $\frac{n}{2}$ networks belong to one kind of networks (*regular*, *distance* or *AT*) and the remaining $\frac{n}{2}$ networks belong to another kind, spanning all 3 combinations.

As described in Section 3.5, the predictions of classifiers in an ensemble can be integrated using different criteria. In general we found that all the methods which did not use our distance score (3.1), including methods which used any of the other confidence score for prediction weighting, performed less well than a simple average of the softmax activation (method 1 in Section 3.5).

Otherwise the best performance was obtained when using a weighted average (method 3 in Section 3.5) with weights defined by our distance score (3.1). With variants (d-f) we also checked two options of obtaining the distance score: (i) Each network defined its own confidence score; (ii) in light of the advantage demonstrated by hybrid networks as shown in Section 4.2 and for each pair of networks from different kinds, the distance score for both was computed while using the embedding of only one of networks in the pair. MC-Dropout was not used in this section due to its high computational cost.

While our experiments included all variants and all weighting options, only 4 cases are shown in the following description of the results in order to improve readability: 1) the combination achieving best performance; 2) the combination achieving best performance when not using adversarial training (as *AT* entails additional computational load at train time); 3) the ensemble variant achieving best performance without using the distance score (baseline), 4) ensemble average when using adversarial training without distance score. Additional results for most of the other conditions we tested can be found in Appendix A. Each experiment was repeated at least 5 times, with no overlap between the networks.

CIFAR-100 and STL-10: Fig. 4.4 shows the ensemble accuracy for the methods mentioned above when using these datasets. It can be clearly seen that weighting predictions based on the distance score from (3.1) improves results significantly. The best results are achieved when combining *Distance Networks* and *Adversarial Networks*, with significant improvement over an ensemble which is composed of only one kind of networks (i.e. ensemble of Distance Networks or AT networks only, the results of these kind of ensembles are not shown in the graph). Still, we note importantly that the distance score is used to weight **both** kind of networks. Since Adversarial Training is not always applicable due to its computational cost at train time, we show that the combination of *Distance networks* and *Regular networks* can also lead to significant improvement in performance when using the distance score and the hybrid architecture described in Section 4.2. Finally we note that *Adversarial networks* alone achieve very poor results when using the original ensemble average, further demonstrating the value of the distance score in improving the performance of an ensemble of Adversarial networks alone.

SVHN: Results for this dataset are shown in Fig.4.4. While not as significant as those in the other datasets (partly due to the high initial accuracy), they are still consistent with them, demonstrating again the power and robustness of the distance score.

4.5 Novelty Detection

Finally, we compare the performance of the different confidence scores in the task of novelty detection. In this task the confidence score is used to decide another binary classification problem: does the test example belong to the set of classes the networks had been trained on, or rather

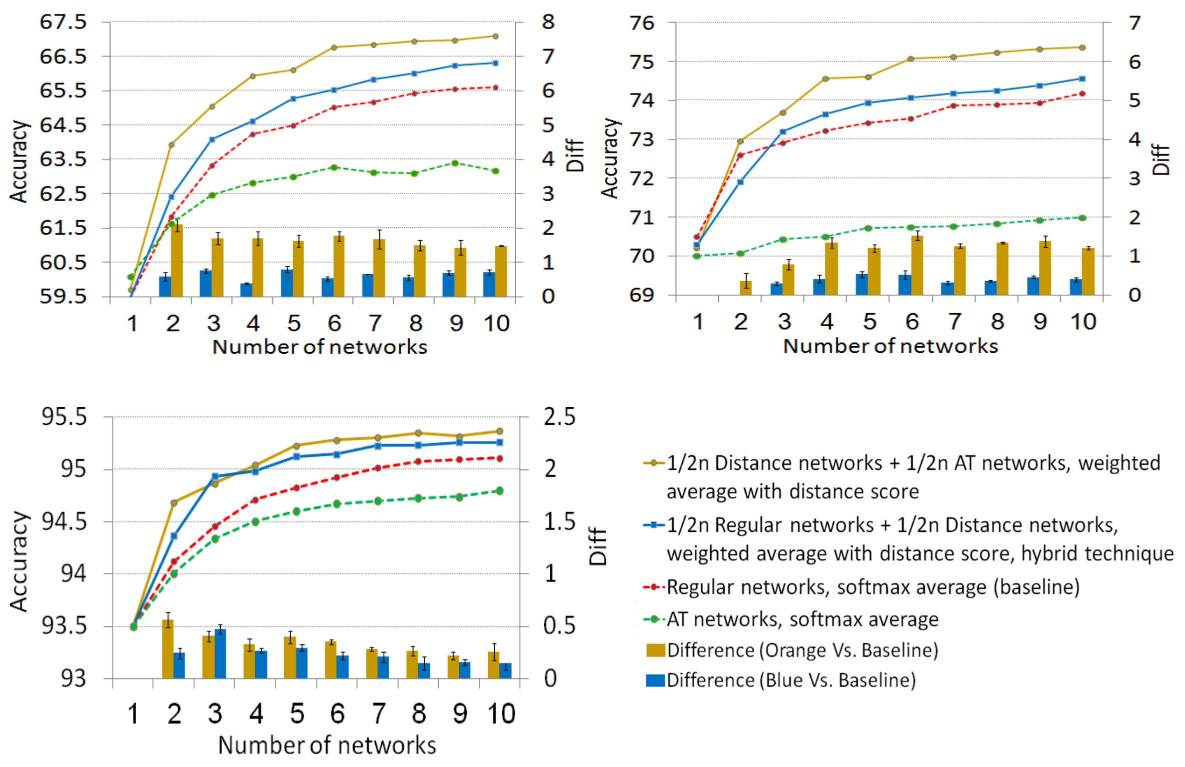

Figure 4.4: Accuracy when using an ensemble of networks with CIFAR-100 (top left), STL-10 (top right) and SVNH (bottom). The X-axis denotes the number of networks in the ensemble. Absolute accuracy (marked on the left Y-axis) is shown for the 2 most successful ensemble methods among all the methods we had evaluated (blue and yellow solid lines, see text), and 2 methods which did not use our distance score including the best performing method in this set (red dotted line, denoted baseline). *Hybrid technique* denotes that for each of the $\frac{n}{2}$ pairs of regular and distance networks, we computed the distance score for **both** networks in the pair using the embeddings of the distance network only. Differences in accuracy between the two top performers and the top baseline method are shown using a bar plot (marked on the right Y-axis), with standard deviation of the difference over (at least) 5 repetitions.

to some unknown class? Performance in this binary classification task is evaluated using the corresponding ROC curve of each confidence score.

We used two contrived datasets to evaluate performance in this task, following the experimental construction suggested in [29]. In the first experiment, we trained the network on the STL-10 dataset, and then tested it on both STL-10 and SVHN test sets. In the second experiment we switched between the datasets (and changed the trained network) making SVHN the known dataset and STL-10 the novel one. The task requires to discriminate between the known and the novel datasets. For comparison we computed novelty, as one often does, with a one-class SVM classifier while using the same embeddings. Novelty thus computed showed much poorer performance, possibly because this dataset involves many classes (one class SVM is typically used with a single class), and therefore these results are not included here.

Table 4.4: AUC results for novelty detection. Left: STL-10 (known) and SVHN (novel). Right: SVHN (known) and STL-10 (novel).

Confidence Score	STL-10/SVHN			SVHN/STL-10		
	Reg.	Dist.	AT	Reg.	Dist.	AT
Max margin	0.8078	0.849	0.86	0.9116	0.9225	0.9851
Entropy	0.8099	0.8566	0.8697	0.9171	0.9334	0.9918
Distance	0.7983	0.8701	**0.901**	0.9037	0.9339	**0.9965**

Results are shown in Table 4.4. Adversarial training, which was designed to handle this sort of challenge, is not surprisingly the best performer. Nevertheless, we see that our proposed confidence score improves the results even further, again demonstrating its added value.

Chapter 5

Summary and Conclusions

In this work we proposed a new confidence score for multi-class neural network classifiers. The method proposed for obtaining this new score is scalable, simple to implement, and can fit any kind of neural network. This method is different from other commonly used methods as it is based on measuring the point density in the effective embedding space of the network, thus providing a more coherent statistical measure for the distribution of the network's predictions.

We showed that the embedding of neural networks that were trained with a standard classification loss, are not well suited for our proposed score. We therefore proposed two ways of achieving such embeddings: the pair-wise distance loss and Adversarial training. While both methods are not entirely new, our work is the first one to use these methods to achiev a better confidence score using the networks' embeddings.

We demonstrated the superiority of the new score several ways. First, we showed that computing our proposed score is cheaper in terms of both train and test time, compared to the improved entropy score achieved with MC-Dropout [13] and Adverserial Training [29]. Secondly, we showed that our score of achieves superior results in 4 different tasks that require the use of a confidence score. Specifically, our proposed confidence score achieves better results when predicting the classifier's prediction as correct or incorrect, when ranking test points for self-training, when combining an ensemble of networks using a weighted average, and finally when trying to detect unknown (novel) test samples. In all tasks we used a number of different datasets and employed task-appropriate network architectures. We also repeated our experiment several times in order to better establish the statistical significance of the results.

In a wider perspective, machine learning, and specifically deep learning methods become increasingly popular. As the popularity of these methods rises the more they become a crucial part of the field. However, the classifier can not be simply trusted to provide reliable predictions all the time. Moreover, classifiers failing to indicate when they are likely to be mistaken can limit their adoption or cause serious accidents. In many fields, like medical diagnosis, security or manufacturing, accurate predictions can be a matter of life and death. In those cases, it's critical to know how confident the classifier is and when an expert human intervention is required. In this context, the more reliable the confidence score of the classifier is, the more effectively it can be used, saving a lot of human effort and even lives.

We hope that our contribution to this area will help researchers and practitioners in all fields make better and more clever use of their algorithms. Knowing better when not to use your classifier means that when you use it, you can trust it more and make the best use out of it.

Appendix A

Ensemble methods variants tested during experiments

On the next page there is a table with all the variants we tested during the ensemble methods experiments, along with sample results on the CIFAR-100 dataset, when using 10 networks. Notice that *Hybrid Technique* means that for each pair of networks from different kinds, the distance score for both was computed while using the embedding of only one of networks in the pair (either the Distance networks or the AT ones)

Table A.1: All the variants we tested during the ensemble methods experiments, along with a sample of results on the CIFAR-100 dataset

Networks used	Ensemble method	Results on CIFAR-100 10 networks
Regular	Softmax Average	65.71
Regular	Weighted average with entropy score	65.64
Regular	Voting	65.35
Regular	Confidence voting with entropy score	64.26
Regular	Dictator voting with entropy score	64.67
Regular	Weighted average with distance score	63.30
Regular	Confidence voting with distance score	63.12
Regular	Dictator voting with distance score	63.50
Distance	Softmax Average	65.61
Distance	Weighted average with entropy score	64.93
Distance	Voting	64.53
Distance	Confidence voting with entropy score	64.41
Distance	Dictator voting with entropy score	65.13
Distance	Weighted average with distance score	66.23
Distance	Confidence voting with distance score	65.87
Distance	Dictator voting with distance score	65.66
AT	Softmax Average	63.18
AT	Weighted average with entropy score	62.98
AT	Voting	62.93
AT	Confidence voting with entropy score	62.93
AT	Dictator voting with entropy score	63.06
AT	Weighted average with distance score	64.13
AT	Confidence voting with distance score	64.21
AT	Dictator voting with distance score	64.35
1/2n Distance + 1/2n Regular	Softmax Average	65.62
1/2n Distance + 1/2n Regular	Weighted average with entropy score	65.25
1/2n Distance + 1/2n Regular	Voting	64.07
1/2n Distance + 1/2n Regular	Confidence voting with entropy score	63.72
1/2n Distance + 1/2n Regular	Dictator voting with entropy score	64.19
1/2n Distance + 1/2n Regular	Weighted average with distance score - hybrid technique	66.37
1/2n Distance + 1/2n Regular	Confidence voting with distance score - hybrid technique	65.59
1/2n Distance + 1/2n Regular	Dictator voting with distance score - hybrid technique	65.16
1/2n Distance + 1/2n AT	Softmax Average	66.58
1/2n Distance + 1/2n AT	Weighted average with entropy score	65.39
1/2n Distance + 1/2n AT	Voting	65.44
1/2n Distance + 1/2n AT	Confidence voting with entropy score	65.32
1/2n Distance + 1/2n AT	Dictator voting with entropy score	66.46
1/2n Distance + 1/2n AT	Weighted average with distance score	67.09
1/2n Distance + 1/2n AT	Confidence voting with distance score	66.04
1/2n Distance + 1/2n AT	Dictator voting with distance score	65.76
1/2n Distance + 1/2n AT	Weighted average with distance score - hybrid technique	66.81
1/2n Distance + 1/2n AT	Confidence voting with distance score - hybrid technique	65.73
1/2n Distance + 1/2n AT	Dictator voting with distance score - hybrid technique	64.96
1/2n AT + 1/2n Regular	Softmax Average	64.35
1/2n AT + 1/2n Regular	Weighted average with entropy score	63.75
1/2n AT + 1/2n Regular	Voting	63.86
1/2n AT + 1/2n Regular	Confidence voting with entropy score	63.69
1/2n AT + 1/2n Regular	Dictator voting with entropy score	64.12
1/2n AT + 1/2n Regular	Weighted average with distance score - hybrid technique	65.12
1/2n AT + 1/2n Regular	Confidence voting with distance score - hybrid technique	65.32
1/2n AT + 1/2n Regular	Dictator voting with distance score - hybrid technique	64.96

BIBLIOGRAPHY

[1] M. ABADI, A. AGARWAL, P. BARHAM, E. BREVDO, Z. CHEN, C. CITRO, G. S. CORRADO, A. DAVIS, J. DEAN, M. DEVIN, ET AL., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, arXiv preprint arXiv:1603.04467, (2016).

[2] D. AMODEI, C. OLAH, J. STEINHARDT, P. CHRISTIANO, J. SCHULMAN, AND D. MANÉ, *Concrete problems in ai safety*, arXiv preprint arXiv:1606.06565, (2016).

[3] A. S. AREFIN, C. RIVEROS, R. BERRETTA, AND P. MOSCATO, *Gpu-fs-k nn: A software tool for fast and scalable k nn computation using gpus*, PloS one, 7 (2012), p. e44000.

[4] E. BAUER AND R. KOHAVI, *An empirical comparison of voting classification algorithms: Bagging, boosting, and variants*, Machine learning, 36 (1999), pp. 105–139.

[5] M. BUDKA AND B. GABRYS, *Density-preserving sampling: robust and efficient alternative to cross-validation for error estimation*, IEEE transactions on neural networks and learning systems, 24 (2013), pp. 22–34.

[6] D.-A. CLEVERT, T. UNTERTHINER, AND S. HOCHREITER, *Fast and accurate deep network learning by exponential linear units (elus)*, arXiv preprint arXiv:1511.07289, (2015).

[7] A. COATES, H. LEE, AND A. Y. NG, *An analysis of single-layer networks in unsupervised feature learning*, Ann Arbor, 1001 (2010), p. 2.

[8] M. CULP AND G. MICHAILIDIS, *An iterative algorithm for extending learners to a semi-supervised setting*, Journal of Computational and Graphical Statistics, 17 (2008), pp. 545–571.

[9] A. DAMIANOU AND N. LAWRENCE, *Deep gaussian processes*, in Artificial Intelligence and Statistics, 2013, pp. 207–215.

[10] J. DENG, W. DONG, R. SOCHER, L.-J. LI, K. LI, AND L. FEI-FEI, *ImageNet: A Large-Scale Hierarchical Image Database*, in CVPR09, 2009.

[11] T. G. DIETTERICH, *Ensemble methods in machine learning*, in International workshop on multiple classifier systems, Springer, 2000, pp. 1–15.

[12] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, *Long-term recurrent convolutional networks for visual recognition and description*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 2625–2634.

[13] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, arXiv preprint arXiv:1506.02142, 2 (2015).

[14] V. Garcia, E. Debreuve, and M. Barlaud, *Fast k nearest neighbor search using gpu*, in Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on, IEEE, 2008, pp. 1–6.

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572, (2014).

[16] H. Gunadi, *Comparing nearest neighbor algorithms in high-dimensional space*, (2011).

[17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, *On calibration of modern neural networks*, arXiv preprint arXiv:1706.04599, (2017).

[18] R. Hadsell, S. Chopra, and Y. LeCun, *Dimensionality reduction by learning an invariant mapping*, in Computer vision and pattern recognition, 2006 IEEE computer society conference on, vol. 2, IEEE, 2006, pp. 1735–1742.

[19] P. Hart, *The condensed nearest neighbor rule (corresp.)*, IEEE transactions on information theory, 14 (1968), pp. 515–516.

[20] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[21] D. Hendrycks and K. Gimpel, *A baseline for detecting misclassified and out-of-distribution examples in neural networks*, arXiv preprint arXiv:1610.02136, (2016).

[22] E. Hoffer and N. Ailon, *Deep metric learning using triplet network*, in International Workshop on Similarity-Based Pattern Recognition, Springer, 2015, pp. 84–92.

[23] H. Hu, G.-T. Zhou, Z. Deng, Z. Liao, and G. Mori, *Learning structured inference neural networks with label relations*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2960–2968.

[24] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, *Deep networks with stochastic depth*, in European Conference on Computer Vision, Springer, 2016, pp. 646–661.

[25] V. HYVÖNEN, T. PITKÄNEN, S. TASOULIS, E. JÄÄSAARI, R. TUOMAINEN, L. WANG, J. CORANDER, AND T. ROOS, *Fast k-nn search*, arXiv preprint arXiv:1509.06957, (2015).

[26] A. KESSY, A. LEWIN, AND K. STRIMMER, *Optimal whitening and decorrelation*, The American Statistician, (2017).

[27] D. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[28] A. KRIZHEVSKY AND G. HINTON, *Learning multiple layers of features from tiny images*, (2009).

[29] B. LAKSHMINARAYANAN, A. PRITZEL, AND C. BLUNDELL, *Simple and scalable predictive uncertainty estimation using deep ensembles*, arXiv preprint arXiv:1612.01474, (2016).

[30] D.-H. LEE, *Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks*, in Workshop on Challenges in Representation Learning, ICML, vol. 3, 2013.

[31] H. LI, X. WANG, AND S. DING, *Research and development of neural network ensembles: a survey*, Artificial Intelligence Review, (2017), pp. 1–25.

[32] D. J. MACKAY, *Bayesian methods for adaptive models*, PhD thesis, California Institute of Technology, 1992.

[33] M. MARKOU AND S. SINGH, *Novelty detection: a review,Äîpart 2:: neural network based approaches*, 83 (2003), pp. 2499–2521.

[34] S. MARSLAND, *Novelty detection in learning systems*, Neural computing surveys, 3 (2003), pp. 157–195.

[35] D. MCCLOSKY, E. CHARNIAK, AND M. JOHNSON, *Effective self-training for parsing*, in Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics, Association for Computational Linguistics, 2006, pp. 152–159.

[36] T. MIYATO, A. M. DAI, AND I. GOODFELLOW, *Adversarial training methods for semi-supervised text classification*, arXiv preprint arXiv:1605.07725, (2016).

[37] R. M. NEAL, *Bayesian learning for neural networks*, vol. 118, Springer Science & Business Media, 2012.

[38] Y. NETZER, T. WANG, A. COATES, A. BISSACCO, B. WU, AND A. Y. NG, *Reading digits in natural images with unsupervised feature learning*, in NIPS workshop on deep learning and unsupervised feature learning, vol. 2011, 2011, p. 5.

[39] A. Nguyen, J. Yosinski, and J. Clune, *Deep neural networks are easily fooled: High confidence predictions for unrecognizable images*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 427–436.

[40] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, *Zero-shot learning with semantic output codes*, in Advances in neural information processing systems, 2009, pp. 1410–1418.

[41] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, *A review of novelty detection*, Signal Processing, 99 (2014), pp. 215–249.

[42] N. Qian, *On the momentum term in gradient descent learning algorithms*, Neural networks, 12 (1999), pp. 145–151.

[43] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, *Training deep neural networks on noisy labels with bootstrapping*, arXiv preprint arXiv:1412.6596, (2014).

[44] C. Rosenberg, M. Hebert, and H. Schneiderman, *Semi-supervised self-training of object detection models*, (2005).

[45] R. Salakhutdinov and G. E. Hinton, *Learning a nonlinear embedding by preserving class neighbourhood structure.*, in AISTATS, vol. 11, 2007.

[46] F. Schroff, D. Kalenichenko, and J. Philbin, *Facenet: A unified embedding for face recognition and clustering*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.

[47] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, *Cnn features off-the-shelf: an astounding baseline for recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 806–813.

[48] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting.*, Journal of Machine Learning Research, 15 (2014), pp. 1929–1958.

[49] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199, (2013).

[50] O. Tadmor, T. Rosenwein, S. Shalev-Shwartz, Y. Wexler, and A. Shashua, *Learning a metric embedding for face recognition using the multibatch method*, in Advances In Neural Information Processing Systems, 2016, pp. 1388–1389.

[51] R. TIBSHIRANI, *A comparison of some error estimates for neural network models*, Neural Computation, 8 (1996), pp. 152–163.

[52] N. VINOKUROV AND D. WEINSHALL, *Novelty detection in multiclass scenarios with incomplete set of class labels*, arXiv preprint arXiv:1604.06242, (2016).

[53] E. A. WAN, *Neural network classification: A bayesian interpretation*, IEEE Transactions on Neural Networks, 1 (1990), pp. 303–305.

[54] J. WESTON, F. RATLE, H. MOBAHI, AND R. COLLOBERT, *Deep learning via semi-supervised embedding*, in Neural Networks: Tricks of the Trade, Springer, 2012, pp. 639–655.

[55] D. YAROWSKY, *Unsupervised word sense disambiguation rivaling supervised methods*, in Proceedings of the 33rd annual meeting on Association for Computational Linguistics, Association for Computational Linguistics, 1995, pp. 189–196.

[56] H. ZARAGOZA AND F. D,ÄÔALCHÉ BUC, *Confidence measures for neural network classifiers*, in Proceedings of the Seventh Int. Conf. Information Processing and Management of Uncertainty in Knowlegde Based Systems, 1998.