Simultaneous Classification and Novelty Detection Using Deep Neural Networks

Aristotelis-Angelos Papadopoulos [1] Mohammad Reza Rajati [2 3]

Abstract

Deep neural networks have achieved great success in classification tasks during the last years. However, one major problem to the path towards artificial intelligence is the inability of neural networks to accurately detect novel class distributions and therefore, most of the classification algorithms proposed make the assumption that all classes are known prior to the training stage. In this work, we propose a methodology for training a neural network that allows it to efficiently detect novel class distributions without compromising much of its classification accuracy on the test examples of known classes. Experimental results on the CIFAR 100 and MiniImagenet data sets demonstrate the effectiveness of the proposed algorithm. The way this method was constructed also makes it suitable for training any classification algorithm that is based on Maximum Likelihood methods.

1. Introduction

Modern neural networks have achieved superior results in classification problems recently (Krizhevsky et al., 2012; He et al., 2016). However, most of the solutions proposed so far make the assumption that data generated from all the class conditional distributions are available during training time i.e., they make the closed-world assumption. In an open world environment (Bendale & Boult, 2015) where examples from novel class distributions might appear during test time, it is necessary to build classifiers that in addition to their high classification accuracy on the known class distributions, are able to detect novel class distributions as well.

[1]Department of Electrical & Computer Engineering, University of Southern California, Los Angeles, California, USA [2]Department of Computer Science, University of Southern California, Los Angeles, California, USA [3]Intelligent Decision Analysis, Redding, California, USA. Correspondence to: Aristotelis-Angelos Papadopoulos <aristotp@usc.edu>.

Preprint. Work in progress.

(Chow, 1970) proposed a technique for rejecting examples with low confidence that does not take into account the open world environment and cannot be applied in modern high dimensional data sets. In (Scheirer et al., 2013), the open space risk is formally defined and in (Bendale & Boult, 2016), a method is proposed for bounding the open space risk by introducing a new model layer, OpenMax, in order to estimate the probability of an input being from an unknown class. Even though this study provides some theoretical guarantees on bounding the open space risk, it is impractical to apply to modern neural networks due to its high computational complexity.

In the class discovery problem setting, (Pelleg & Moore, 2004; Hospedales et al., 2013) are trying to identify instances of rare classes that are known to exist in the training data which is different from the simultaneous classification and novel class distribution detection problem where instances of novel class distributions only appear during test time. Similarly, in (Fink et al., 2006; Kuzborskij et al., 2013; Rebuffi et al., 2017), the authors are trying to incrementally learn new classes but no methods are proposed on how to detect the instances of novel class distributions during test time.

There are only a few works that have addressed the problem of novelty detection in Deep Neural Networks (DNN). (Yu et al., 2017) used the GAN framework (Goodfellow et al., 2014) to generate samples that are close to the known class distributions but are classified as fake by the discriminator. Then, they used those samples in order to train SVM classifiers in a supervised manner to detect examples from unseen classes. Similarly, (Kliger & Fleishman, 2018) used a multi-class GAN framework and the Feature Matching Loss in order to produce a generator that generates a mixture of nominal data and novel data and a discriminator that performs simultaneous classification and novelty detection. (Hendrycks & Gimpel, 2017) proposed a baseline for detecting misclassified and out-of-distibution examples based on their observation that the prediction probability of out-of-distribution examples tends to be lower than the prediction probability for correct examples.

A single-parameter variant of Platt scaling (Platt, 1999), temperature scaling, was proposed by (Guo et al., 2017) for calibration of modern neural networks. For image data, based on the idea of (Hendrycks & Gimpel, 2017), (Liang et al., 2017) observed that the additional use of temperature scaling and small perturbations at the input can push the softmax scores of in- and out-of-distribution images further apart from each other, making the out-of distribution images distinguishable. However, they never mention in their work what is the effect of this method on the accuracy of the model on the test examples generated by the already known class conditional distributions, i.e. they only focus on the novelty detection problem.

Viewing the knowledge of a model as the class conditional distribution it produces over outputs given an input (Hinton et al., 2015), the entropy of this conditional distribution can be used as a regularization method that penalizes the confident predictions of a neural network (Pereyra et al., 2017). In this approach, instead of penalizing the confident predictions of posterior probabilities performed by a neural network, we force it to make confident predictions for the examples generated by the known class distributions while at the same time, we force it to be highly uncertain for examples generated by novel class distributions. It is worth mentioning that forcing the neural network to make predictions for examples generated by novel class distributions with low probability, is equivalent to having the maximum entropy distribution in the output, i.e. the uniform distribution. As the experimental results show, not only does the neural network maintain its classification accuracy on the test set of known class distributions but it concurrently achieves high discrimination between known and novel class distributions in the space of softmax probabilities.

2. Proposed Method

In this section, we consider the general problem of multi-class classification. We start by analyzing the standard multi-class classification problems that make the closed world assumption, i.e. the assumption that the training dataset contains examples from all possible classes and then we consider open world classification problems, where not all classes are known during training and therefore, the simultaneous classification and novelty detection ability of the designed algorithms is of high importance.

2.1. Standard Multi-class Classification Problems

Let us consider a classification model that can be represented by a parametrized function f_θ, where θ stands for the vector of parameters in f_θ. In order to construct a successful classification algorithm, Maximum Likelihood methods are usually used to minimize the Negative Log Likelihood (NLL) loss function (Hastie et al., 2001) which is also known as cross-entropy loss (Goodfellow et al., 2016). The cross-entropy loss function for a multi-class classification problem can be mathematically described as:

$$\mathcal{L}_{CE}(f_\theta) = \sum_{i=1}^{N}\sum_{j=1}^{K}\Big[y_j^{(i)}\log(f_\theta(x^{(i)})_j) + (1-y_j^{(i)})\log(1-f_\theta(x^{(i)})_j)\Big] \quad (1)$$

where N is the number of examples in the training dataset, K is the total number of classes, $y_j^{(i)}$ is the true label of the example $x^{(i)}$, and $f_\theta(x^{(i)})_j$ expresses the conditional probability that the observed example $x^{(i)}$ belongs to class j, $P(Y = y_j|X = x^{(i)})$. This conditional class probability is usually estimated by a softmax layer.

2.2. Open-World Multi-class Classification

Most of the classification problems studied in the research community make the closed-world assumption i.e., the assumption that there are K known classes and each class has some training examples in the data set. However, this assumption is not valid in many real world applications i.e., it is impossible for one to know all the available classes prior to training the classifier. To this end, building a multi-class classifier that can generalize well on test examples of the already known classes while concurrently detecting examples from unseen class distributions is of major importance.

In this work, we propose a solution to the aforementioned problem. Our main focus is on how we can efficiently design Deep Neural Network (DNN) classifiers that generalize well on test examples of already learned class distributions but at the same time also have the ability to accurately detect examples generated by novel class distributions. Even though our focus is on DNNs, the proposed method can be effectively applied to any multi-class classification algorithm.

Making the closed-world assumption, one can typically minimize the cross-entropy loss function in (1) in order to build a multi-class classifier that generalizes well on test examples generated by learned class distributions (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015). However, making the open-world assumption, the same classifier will fail in case it receives examples produced by unknown class distributions since it will classify them in one of the known classes. One potential solution to this problem is the use of a threshold T on the probabilities calculated by the softmax layer of a DNN i.e., in case the maximum probability calculated by the softmax layer is greater than T, then classify that example in one of the already known classes; otherwise, reject it as it was coming from a novel class distribution. However, this solution usually fails as well, mainly due to the fact that there are a lot of examples coming from learned class distributions which might be classified with low probability and examples coming from novel class distributions which might be classified with high probability (Nguyen et al., 2015). The question that naturally arises at this point is whether there is a way to construct a DNN classifier that makes highly certain predictions of the posterior probabilities for examples coming from already learned class distributions and is highly uncertain when it sees examples coming from novel (unseen) class distributions. If this is possible, then we can effectively use a suitable threshold T to discriminate examples generated by known and unknown class distributions. This threshold will classify any example with a prediction probability below a certain level as it was generated by a novel class distribution.

2.3. Simultaneous Classification and Novelty Detection

Let D_k represent all the class conditional probability distributions $(D_{k,1}, ..., D_{k,K})$ where K is the number of classes which are known during training. For instance, $D_{k,j} = P(Y = y_j|X), \forall j = 1, ..., K$. Let also D_{all} represent all the class conditional probability distributions $(D_{k,1}, ..., D_{k,K}, D_{k,K+1}, ...)$. If D_{all} were known or could be learned, then we would face a closed-world classification problem for which many solutions and deep learning algorithms have been proposed. However, in reality, only D_k is known or can be learned.

In order to build a DNN classifier that is able to simultaneously perform multi-class classification as well as detection of novel class distributions, ideally, we would like to have access to data coming from D_{all}. However, since D_{all} is generally unknown during the training phase of the algorithm, we can use labeled data coming from D_k in order to learn the class distributions $(D_{k,1}, ..., D_{k,K})$. Additionally, thanks to the plethora of available unlabeled data, we can use them during training in combination with the labeled data coming from D_k. As we will later show in our experimental results, this method can effectively detect novel class distributions during test time, *which were completely different from the probability distribution describing the unlabeled data used*. This is achieved mainly because we do not try to explicitly learn the probability distribution describing the unlabeled data, but we train the DNN to make uncertain predictions whenever it sees an example coming from a general unknown class distribution.

Let D_u be the probability distribution generating the unlabeled data used during training. Then, we can formulate the following constrained optimization problem:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \mathcal{L}_{CE}(f_\theta) \\ \text{subject to} \quad & \max_{l=1,...,K}\left(\frac{e^{z_l}}{\sum_{j=1}^K e^{z_j}}\right) = 1,\ \forall x^{(i)} \sim D_k \\ & \max_{l=1,...,K}\left(\frac{e^{z_l}}{\sum_{j=1}^K e^{z_j}}\right) = \frac{1}{K},\ \forall x^{(i)} \sim D_u \end{aligned} \tag{2}$$

where $\mathcal{L}_{CE}(f_\theta)$ is given by the following equation:

$$\begin{aligned} \mathcal{L}_{CE}(f_\theta) = \sum_{j=1}^{K} \sum_{(x^{(i)}, y_j^{(i)}) \sim D_k} \Big[& y_j^{(i)} log(f_\theta(x^{(i)})_j) \\ & + (1 - y_j^{(i)}) log(1 - f_\theta(x^{(i)})_j) \Big] \end{aligned} \tag{3}$$

The optimization problem described by (2)-(3) is trying to minimize the cross-entropy loss function over the examples sampled from the probability distribution D_k subject to two additional constraints. Let z denote the vector representation of the example $x^{(i)}$ in the feature space produced by the last layer of the DNN. Then the first constraint forces the maximum probability calculated by the softmax layer to 1 for all examples coming from the probability distribution D_k, while the second constraint forces the maximum probability calculated by the softmax layer to $\frac{1}{K}$ for all examples coming from the probability distribution D_u. In other words, the first constraint is trying to make the DNN predict examples from known classes with high certainty, while the second constraint forces the DNN to be highly uncertain for examples of classes it has never seen before.

Due to the high difficulty we might face when solving the nonconvex constrained optimization problem described by (2)-(3), let us introduce Lagrange multipliers (Boyd & Vandenberghe, 2004) and convert it into the following unconstrained optimization problem:

$$\underset{\theta}{\text{minimize}} \quad \mathcal{L}_l(f_\theta) \tag{4}$$

where the index l stands for the Lagrangian and $\mathcal{L}_l(f_\theta)$ can be written as:

$$\begin{aligned} \mathcal{L}_l(f_\theta) &= \mathcal{L}_{CE}(f_\theta) \\ &+ \lambda_1 \sum_{x^{(i)} \sim D_k} \left(1 - \max_{l=1,...,K}\left(\frac{e^{z_l}}{\sum_{j=1}^K e^{z_j}}\right)\right) \\ &+ \lambda_2 \sum_{x^{(i)} \sim D_u} \left(\frac{1}{K} - \max_{l=1,...,K}\left(\frac{e^{z_l}}{\sum_{j=1}^K e^{z_j}}\right)\right) \end{aligned} \tag{5}$$

where it is worth mentioning that in (4)-(5), we used only one Lagrange multiplier for each set of constraints existing in (2)-(3) instead of using one for each constraint in order to avoid introducing a large number of hyperparameters in the designed algorithm.

Now, in order to avoid the third term of (5) from taking negative values, we can transform (5) into the following form:

$$\begin{aligned}\mathcal{L}_{ls}(f_\theta) &= \mathcal{L}_{CE}(f_\theta) \\ &+ \lambda_1 \sum_{x^{(i)} \sim D_k} \left(1 - \max_{l=1,\dots,K}\left(\frac{e^{z_l}}{\sum_{j=1}^{K} e^{z_j}}\right)\right)^2 \\ &+ \lambda_2 \sum_{x^{(i)} \sim D_u} \left(\frac{1}{K} - \max_{l=1,\dots,K}\left(\frac{e^{z_l}}{\sum_{j=1}^{K} e^{z_j}}\right)\right)^2\end{aligned} \tag{6}$$

where the only difference between (5) and (6) is that in (6), we use the squared values of the second and the third term of (5).

The standard method for minimizing loss functions like the one described by (6) is the use of numerical optimization algorithms (Nocedal & Wright, 2006). However, as we also observed in our experiments, it is usual that the prediction of the DNN for each $x^{(i)} \sim D_u$ constantly changes after a few number of steps of the optimization algorithm when we force the maximum probability produced by the softmax to take values close to $\frac{1}{K}$. Our goal is the creation of the maximum entropy distribution for the examples generated by unknown class distributions in the output of the DNN and the aforementioned phenomenon makes it harder. To overcome this drawback, we will further modify (6) and construct the following optimization problem:

$$\underset{\theta}{\text{minimize}} \quad \mathcal{L}(f_\theta) \tag{7}$$

where $\mathcal{L}(f_\theta)$ is given by:

$$\begin{aligned}\mathcal{L}(f_\theta) &= \mathcal{L}_{CE}(f_\theta) \\ &+ \lambda_1 \sum_{x^{(i)} \sim D_k} \left(1 - \max_{l=1,\dots,K}\left(\frac{e^{z_l}}{\sum_{j=1}^{K} e^{z_j}}\right)\right)^2 \\ &+ \lambda_2 \sum_{x^{(i)} \sim D_u} \sum_{l=1}^{K} \left(\frac{1}{K} - \frac{e^{z_l}}{\sum_{j=1}^{K} e^{z_j}}\right)^2\end{aligned} \tag{8}$$

where the difference between (6) and (8) is that in (8), for each $x^{(i)} \sim D_u$, we force all the prediction probabilities produced by the softmax to take values close to $\frac{1}{K}$ instead of just the maximum one in order to create a uniform distribution in the output of the DNN for the examples generated by unknown class distributions.

At this point, it is worth mentioning that during the experiments, we observed that the classification accuracy of a designed algorithm is more robust to changes in the values of the hyperparameters λ_1 and λ_2 whenever the algorithm has already reached the desired level of accuracy. For instance, if we start training the DNN with a relatively high value of λ_1, due to random weight initialization, there is high chance that the initial prediction probabilities of the DNN are forced towards 1 making the DNN to get stuck in a low accuracy level. Therefore, it is recommended to split the training of the algorithm into two stages. During the first stage, we train the algorithm by minimizing the cross-entropy loss function as described by (3) until the algorithm reaches the desired level of classification accuracy on the test examples sampled from the probability distribution D_k. In the second stage, we continue the training of the algorithm by minimizing the loss function given by (8). In case one starts the training of the algorithm using directly the loss function given by (8) then, there is high risk that the classification accuracy remains at a very low level especially in the case where the values of the hyperparameters λ_1 and λ_2 are high. This can be justified using the fact that the loss function in (8) forces the maximum probability calculated by the softmax layer to 1 for all examples coming from the probability distribution D_k and therefore, if the value of the hyperparameter λ_1 is high, the algorithm will force the initial prediction close to 1, which is most probably incorrect due to random weight initialization, leading to a low classification accuracy. The summary of the proposed training method is described in Algorithm 1.

Algorithm 1 Classification & Novelty Detection

```
Require: probability distributions $D_k$ and $D_u$
Require: $N_k$,$N_u$: number of training examples from $D_k$ and $D_u$
Require: hyperparameters $\lambda_1$ and $\lambda_2$
Require: number of epochs $N_1$ and $N_2$
Require: batch size $b$
 1: Randomly initialize $\theta$
 2: for $i = 1$ to $N_1$ do
 3:   for $j = 1$ to $\frac{N_k}{b}$ do
 4:     Sample $b$ training examples from $D_k$
 5:     Run a gradient descent step to minimize the cross-entropy loss function given by (3)
 6:   end for
 7: end for
 8: for $i = N_1$ to $N_1 + N_2$ do
 9:   for $j = 1$ to $\frac{N_k+N_u}{b}$ do
10:     Sample $\frac{b}{2}$ training examples from $D_k$ and $\frac{b}{2}$ training examples from $D_u$ to form a batch $b$
11:     Run a gradient descent step to minimize the loss function given by (8)
12:   end for
13: end for
```

3. Experimental Results

In this section, we present the experimental results of the proposed algorithm. We run three main experiments in which we build Convolutional Neural Networks (CNNs) to perform image classification. In the first experiment, we use the CIFAR 100 dataset (Krizhevsky, 2009) where we use

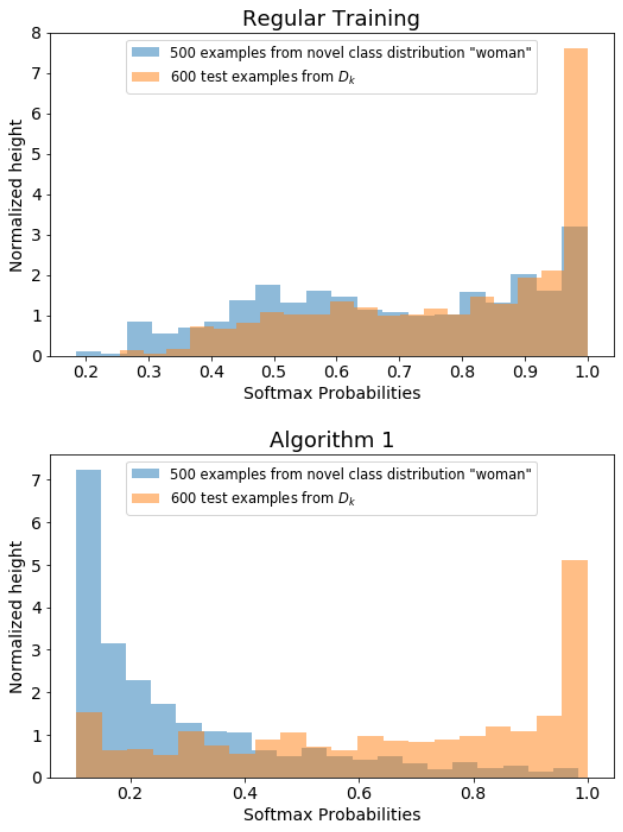

Figure 1. CIFAR 100 Experiments for novel class distribution “woman” belonging to the superclass *people*: Top: Histogram of Softmax probabilities after training the CNN for 200 epochs by minimizing the cross-entropy loss function (test accuracy: 60.3%). Bottom: Histogram of Softmax probabilities after training the CNN for 200 epochs with Algorithm 1 by choosing $N_1 = 100$, $N_2 = 100$, $\lambda_1 = 0.1$, $\lambda_2 = 0.2$ (test accuracy: 62.3%).

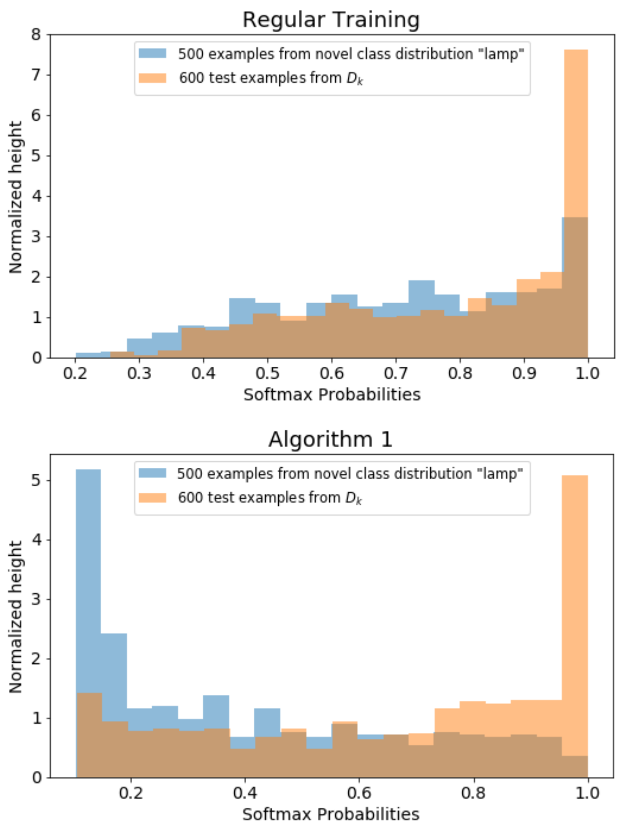

Figure 2. CIFAR 100 Experiments for novel class distribution “lamp” belonging to the superclass *household electrical devices*: Top: Histogram of Softmax probabilities after training the CNN for 200 epochs by minimizing the cross-entropy loss function (test accuracy: 60.3%). Bottom: Histogram of Softmax probabilities after training the CNN for 200 epochs with Algorithm 1 by choosing $N_1 = 100$, $N_2 = 100$, $\lambda_1 = 0.15$, $\lambda_2 = 0.3$ (test accuracy: 61.4%).

completely different classes to represent D_k, D_u as well as the novel class distributions that we want to detect during the test setting. In the second experiment, we apply the same setting as before for the MiniImagenet data set (Ravi & Larochelle, 2017), i.e. distinct classes are used for representing D_k, D_u as well as the novel class distributions that we want to detect during the test setting. Last, in the third experiment, we use classes from MiniImagenet to represent D_k as well as the novel class distributions that we want to detect during the test setting and we use sampled classes from the Caltech 101 data set (Fei-Fei et al., 2007) to represent D_u in order to demonstrate that the proposed method can effectively detect novel class distributions even by *using randomly sampled unlabeled data*. At this point, it should be mentioned that we did not try to build state-of-the-art image classifiers for each of the data sets used. Our main goal during the experiments was to demonstrate how we can effectively train a DNN classifier to perform simultaneous classification and novelty detection by concurrently maintaining the desired level of accuracy on the test examples sampled from D_k. All of our experiments[1] are performed using TensorFlow (Abadi et al., 2016).

3.1. Results on the CIFAR 100

The CIFAR 100 dataset consists of 20 distinct superclasses each of which contains 5 different classes. For our experiments, we randomly selected the superclasses *aquatic mammals* and *flowers* to represent D_k in order to build a CNN that performs a 10-class classification task. Out of the 600 examples that each class contains, we randomly selected 480 for training, 60 for validation, and 60 for the test set. As far as it concerns the unlabeled data used during training, we randomly selected the superclasses *fruit and vegetables* and *insects* to represent D_u. We also randomly sampled 480 examples from each class of those superclasses in order to have an equal number of labeled and unlabeled data during the second stage of the training. Last, we randomly

[1]The code for the experiments will be released online upon acceptance of the paper.

selected the superclasses *household electrical devices* and *people* to represent the novel class distributions in order to test whether the CNN will be able to detect them.

The CNN model we built consisted of 4 convolutional layers followed by 2 fully connected layers. In the first convolutional layer, we used 64 filters with 4×4 convolutions and a stride of 1, followed by a ReLU nonlinearity and a 2×2 max-pooling layer with a stride of 2. The second convolutional layer had 64 filters with 4×4 convolutions and a stride of 1, followed by a ReLU nonlinearity, a 2x2 max-pooling layer with a stride of 2 and dropout with rate 0.3. The third and fourth convolutional layers consisted of 64 filters with 3×3 convolutions and a stride of 1, followed by a ReLU activation function, a 2×2 max-pooling layer with a stride of 1 and dropout with rate 0.3. Last, the dimension of the next fully connected layer was 128 and we used a dropout rate of 0.4.

For the training purposes, we used the Adam optimization algorithm (Kingma & Ba, 2014). During the regular CNN training, we used a constant step size of 10^{-4} for 200 epochs. For Algorithm 1, we trained the CNN model for 100 epochs with a step size of 10^{-4}, the next 50 epochs with a step size of $4 \cdot 10^{-5}$ and the last 50 epochs with a step size of 10^{-5}.

As it can be observed in Figures 1 and 2, simply minimizing the cross-entropy loss function makes the DNN to be highly certain for all its predictions, *even for examples generated by novel class distributions*. On the other hand, using the training method proposed in Algorithm 1 makes the DNN predict test examples from D_k with high probability while being highly uncertain for examples sampled by novel class distributions. This result allows us to appropriately choose a suitable threshold T in order to discriminate examples generated by D_k and novel class distributions without affecting the classification accuracy of the DNN on test examples sampled from D_k.

3.2. Results on the MiniImagenet

The MiniImagenet data set consists of 100 classes in total each of which contains 600 examples. For our experiments, we randomly selected 10 classes to represent D_k. Out of those, we used 480 examples for the training, 60 for validation, and 60 for the test setting. Then, we randomly selected another 8 classes to represent D_u and we used all of their examples, giving us a total of 4800 unlabeled examples so that we get an equal number of labeled and unlabeled data during the second stage of the training. Last, we randomly selected another 14 classes that represent the novel class distributions to examine whether the CNN will be able to detect them during the test setting.

It should be noted that as part of data preprocessing, we cropped the images into 84×84. The CNN model we built consisted of 4 convolutional layers and 2 fully connected layers. In the first convolutional layer, we used 64 filters with 7×7 convolutions and a stride of 1, followed by a ReLU nonlinearity and a 3×3 max-pooling layer with a stride of 2. The second convolutional layer had 64 filters with 5×5 convolutions and stride of 1, followed by a ReLU nonlinearity, a 3×3 max-pooling layer with a stride of 2 and dropout with rate 0.3. The next two convolutional layers consisted of 64 filters with 3×3 convolutions and a stride of 1, followed by a ReLU nonlinearity, a 3×3 max-pooling layer with a stride of 2 and dropout with rate 0.3. Last, the dimension of the next fully connected layer was 128 and we used a dropout rate of 0.4.

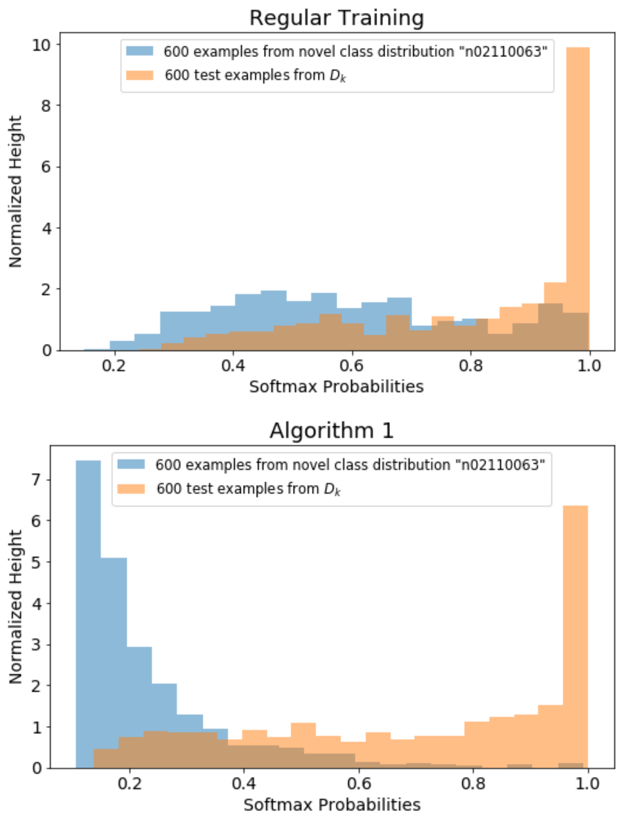

Figure 3. MiniImagenet Experiments for novel class distribution "n02110063": Top: Histogram of Softmax probabilities after training the CNN for 200 epochs by minimizing the cross-entropy loss function (test accuracy: 76.2%). Bottom: Histogram of Softmax probabilities after training the CNN for 200 epochs with Algorithm 1 by choosing $N_1 = 100$, $N_2 = 100$, $\lambda_1 = 0.06$, $\lambda_2 = 0.06$ (test accuracy: 76.3%).

For the training purposes, we used the Adam optimization algorithm. For both the regular CNN training and Algorithm 1, we trained the CNN model for 100 epochs with a step size of 10^{-4}, the next 50 epochs with a step size of $4 \cdot 10^{-5}$ and the last 50 epochs with a step size of 10^{-5}.

As it can be observed in Figure 3, whenever we train the DNN by simply minimizing the cross-entropy loss as described by (3), it is impossible to discriminate the novel

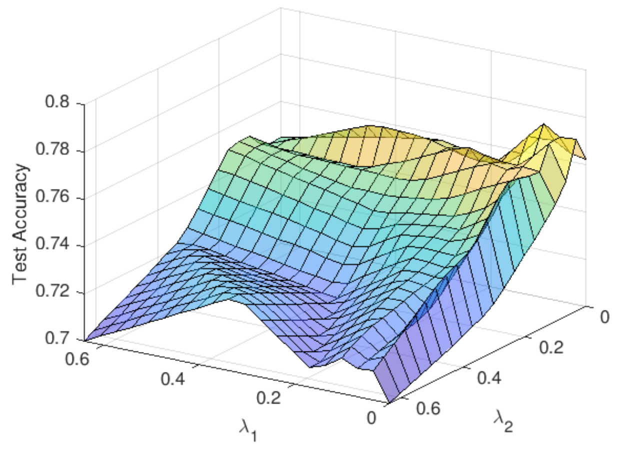

Figure 4. Test accuracy on 600 examples from 10 randomly sampled classes from the MiniImagenet data set representing D_k for different values of the hyperparameters λ_1 and λ_2. Maximum accuracy: 78.7% for $\lambda_1 = \lambda_2 = 0.04$.

class distribution from the classes generated by D_k since the DNN has become biased to making predictions with high certainty. On the other hand, training the DNN as proposed by Algorithm 1 allows us to achieve a significant discrimination by concentrating the majority of examples sampled from the novel class distribution close to $\frac{1}{K}$ (0.1 in this experiment) while maintaining the classification accuracy on the test examples generated by D_k at the desired level.

In order to demonstrate the robustness of our algorithm to the changes of the values of the hyperparameters λ_1 and λ_2, we also plot the test accuracy of the CNN on the test examples sampled from D_k for different values of λ_1 and λ_2 as shown in Figure 4. Observe that the CNN maintains a high classification accuracy for a large range of values of the hyperparameters λ_1 and λ_2.

3.3. Results on the combined MiniImagenet and Caltech 101

In this experiment, we want to demonstrate that the proposed Algorithm 1 can be applied to a combined dataset. Assuming that we want to build a DNN classifier for a classification task, we sample training examples from 10 classes from the MiniImagenet data set to represent the class distributions belonging to D_k. As far as it concerns D_u, we selected unlabeled data from the Caltech 101 data set. Our main purpose is to build a CNN model that achieves a high classification accuracy on test examples sampled from D_k while being able to detect novel class distributions from the MiniImagenet data set.

From each of the 10 classes that we randomly picked from the MiniImagenet data set to represent D_k, we sampled 480 examples for training, 60 for validation, and 60 for the test

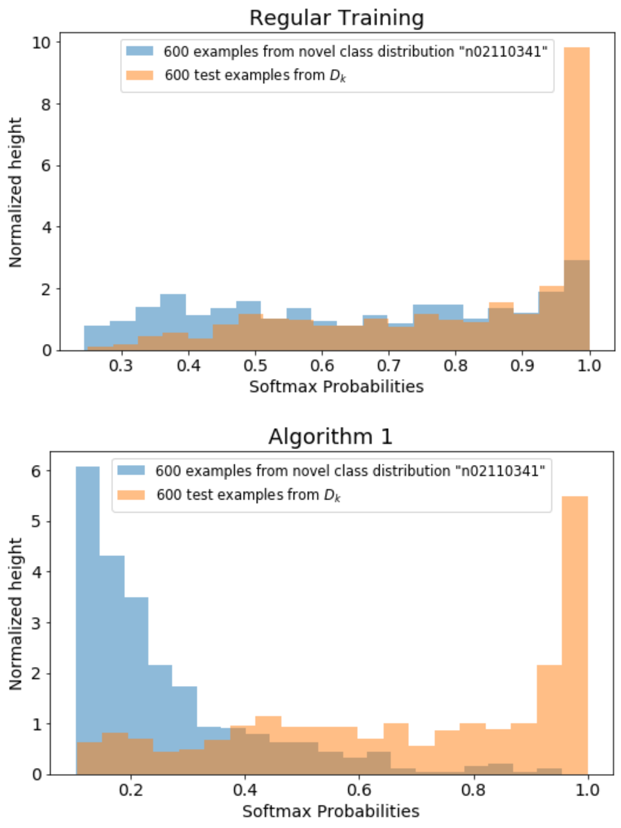

Figure 5. Combined MiniImagenet and Caltech 101 Experiments for novel class distribution "n02110341": Top: Histogram of Softmax probabilities after training the CNN for 200 epochs by minimizing the cross-entropy loss function (test accuracy: 78.2%). Bottom: Histogram of Softmax probabilities after training the CNN for 200 epochs with Algorithm 1 by choosing $N_1 = 100$, $N_2 = 100$, $\lambda_1 = 0.02$, $\lambda_2 = 0.06$ (test accuracy: 76.8%).

set. We also randomly selected another 14 classes from the MiniImagenet data set with 600 examples each in order to use them as the novel class distributions. Last, we also selected 4800 images from the Caltech 101 data set in order to represent unlabeled data generated by D_u.

As part of data preprocessing, we cropped the images of both the MiniImagenet and Caltech 101 data sets into 84×84. The CNN as well as the optimization algorithm used during training were identical to those of section 3.2.

As can be clearly seen in Figure 5, Algorithm 1 achieves simultaneous classification and novel class distribution detection even in the case where random unlabeled data are used (in our case images from the Caltech 101 data set were used). However, it should be mentioned that we need to be careful about the classes where the unlabeled data belong i.e., if a class "dalmatian dog" belongs to D_k then we should avoid using the same class in D_u in order not to confuse the CNN by making it to treat that class as known and novel concurrently.

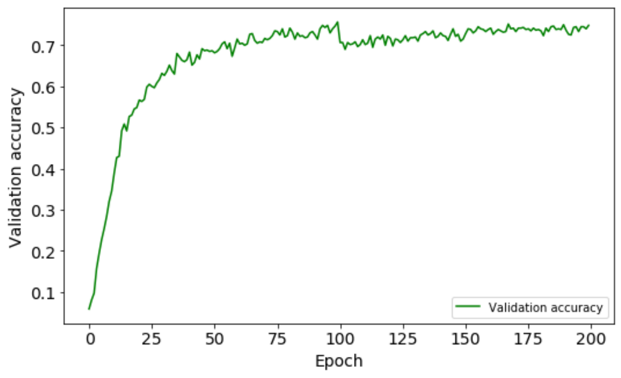

Figure 6. Validation accuracy on 600 examples from 10 randomly sampled classes from the MiniImagenet data set representing D_k during training with Algorithm 1 on the combined MiniImagenet and Caltech 101 data set by choosing $N_1 = 100$, $N_2 = 100$, $\lambda_1 = 0.02$, $\lambda_2 = 0.06$.

In Figure 6, we plot the validation accuracy of the CNN model during the whole training process with Algorithm 1 on the combined MiniImagenet and Caltech 101 data set. As it can be observed, after 100 epochs of training where we change the objective function that we minimize into the form given by (8), there is a drop in the validation accuracy which is later increased by continuing the training for another N_2 epochs. At the end of training with Algorithm 1, the CNN model can reach the same level of validation and test accuracy on the examples generated by D_k while concurrently being able to discriminate between known and novel class distributions.

4. Conclusion and Future Work

We introduced an algorithm that can perform simultaneous classification and novel class distribution detection and can be applied to any classification problem that is based on Maximum Likelihood methods. The basic ingredients of our algorithm are two regularization terms whose goal is to force the neural network to make confident predictions of the posterior probabilities for examples generated by known class distributions while being highly uncertain for examples belonging to novel classes. To achieve this, a combination of labeled and unlabeled data are used during training.

The experimental results on the CIFAR 100 and MiniImagenet data sets demonstrate that the proposed algorithm acquires high discrimination ability between known and novel class distributions compared to the results obtained by regular training methods. Moreover, the results on the combined MiniImagenet and Caltech 101 dataset show that the proposed algorithm can achieve high discrimination between known and novel classes even in the case where randomly sampled unlabeled data are used. Last, it is worth mentioning that the proposed algorithm does not significantly affect the classification accuracy of the model on the test examples of the known classes for a large range of values of the hyperparameters λ_1 and λ_2 allowing us to perform simultaneous classification and novelty detection.

There are several research directions to extend this work. First, the extension of the proposed method in order to be able to sufficiently discriminate examples from classes which are “close” to each other, i.e. in the case where the class “shark” belongs to the known classes and the CNN receives an example from the novel class “dolphin”, could lead to new state-of-the-art results in this research topic. Additionally, the demonstrated robustness of the proposed algorithm on the test examples of the known class distributions suggests an important research question on how these regularization terms affect the change of the weights of the neural network. Last, the application of the proposed algorithm to more classification problems needs to be examined.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv e-prints*, art. arXiv:1603.04467, 2016.

Bendale, A. and Boult, T. Towards open world recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Bendale, A. and Boult, T. Towards open set deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Chow, C. On optimum recognition error and reject tradeoff. *IEEE Trans. Inf. Theor.*, 16(1):41–46, 1970.

Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Comput. Vis. Image Underst.*, 106(1):59–70, 2007.

Fink, M., Shalev-Shwartz, S., Singer, Y., and Ullman, S. Online multiclass learning by interclass hypothesis sharing. In *International Conference on Machine Learning (ICML)*, 2006.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. The MIT Press, 2016.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.

Generative adversarial nets. In *International Conference on Neural Information Processing Systems (NIPS)*, pp. 2672–2680, 2014.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*. Springer. NY, USA, 2001.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hospedales, T. M., Gong, S., and Xiang, T. Finding rare classes: Active learning with generative and discriminative models in active learning. *IEEE Trans. on Knowledge and Data Engineering*, 25(2):374–386, 2013.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, 2014.

Kliger, M. and Fleishman, S. Novelty Detection with GAN. *arXiv e-prints*, art. arXiv:1802.10560, 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. 2012.

Kuzborskij, I., Orabona, F., and Caputo, B. From n to n+1: Multiclass transfer incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

Liang, S., Li, Y., and Srikant, R. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. art. arXiv:1706.02690, 2017.

Nguyen, A. M., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 427–436, 2015.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, NY, USA, second edition, 2006.

Pelleg, D. and Moore, A. Active learning for anomaly and rare-category detection. In *International Conference on Neural Information Processing Systems (NIPS)*, 2004.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., and Hinton, G. E. Regularizing neural networks by penalizing confident output distributions. In *International Conferece on Learning Representations (ICLR)*, 2017.

Platt, J. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 1999.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Rebuffi, S., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Scheirer, W., Rocha, A., Sapkota, A., and Boult, T. Toward open set recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7), 2013.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Yu, Y., Qu, W.-Y., Li, N., and Guo, Z. Open-Category Classification by Adversarial Sample Generation. *arXiv e-prints*, art. arXiv:1705.08722, 2017.