Deep learning from crowds

Filipe Rodrigues
Department of Management Engineering
Denmark Technical University
2800 Kgs. Lyngby, Denmark
rodr@dtu.dk

Francisco Pereira
Department of Management Engineering
Denmark Technical University
2800 Kgs. Lyngby, Denmark
camara@dtu.dk

Abstract

Over the last few years, deep learning has revolutionized the field of machine learning by dramatically improving the state-of-the-art in various domains. However, as the size of supervised artificial neural networks grows, typically so does the need for larger labeled datasets. Recently, crowdsourcing has established itself as an efficient and cost-effective solution for labeling large sets of data in a scalable manner, but it often requires aggregating labels from multiple noisy contributors with different levels of expertise. In this paper, we address the problem of learning deep neural networks from crowds. We begin by describing an EM algorithm for jointly learning the parameters of the network and the confusion matrices of the different annotators for classification settings. Then, a novel general-purpose crowd layer is proposed, which allows us to train deep neural networks end-to-end, directly from the noisy labels of multiple annotators, using backpropagation. We empirically show that the proposed approach is able to internally capture the reliability and biases of different annotators and achieve new state-of-the-art results for various crowdsourced datasets across different settings, namely classification, regression and sequence labeling.

1 Introduction

In the last decade, deep learning has made major advances in solving artificial intelligence problems in different domains such as speech recognition, visual object recognition, object detection and machine translation [20]. This success is often attributed its ability to discover intricate structures in high-dimensional data [10], thereby making it particularly well suited for tackling complex tasks that are often regarded as characteristic of humans, such as vision, speech and natural language understanding. However, typically, a key requirement for learning deep representations of complex high-dimensional data is large sets of labeled data. Unfortunately, in many situations this data is not readily available, and humans are required to manually label big collections of data.

On the other hand, in recent years, crowdsourcing has established itself as a reliable solution to annotate large collections of data. Indeed, crowdsourcing platforms like Amazon Mechanical Turk[1] and Crowdflower[2] have proven to be an efficient and cost-effective way for obtaining labeled data [24, 2], especially for the kind of human-like tasks, such as vision, speech and natural language understanding, for which deep learning methods have been shown to excel. Even in fields like medical imaging, crowdsourcing is being used to collect the large sets of labeled data that modern data-savvy deep learning methods enjoy [6, 1]. However, while crowdsourcing is scalable enough to allow labeling datasets that would otherwise be impractical for a single annotator to handle, it is well known that the noise associated with the labels provided by the various annotators can compromise practical applications that make use of such type of data [21, 4]. Thus, it is not surprising that a large

[1] http://www.mturk.com
[2] http://crowdflower.com

body of the recent machine learning literature is dedicated to mitigating the effects of the noise and biases inherent to such heterogeneous sources of data (e.g. [28, 14, 5]).

When learning deep neural networks from the labels of multiple annotators, typical approaches rely on some sort of label aggregation mechanisms prior to training. In classification settings, the simplest and most common approach is to use majority voting, which naively assumes that all annotators are equally reliable. More advanced approaches, such as the one proposed in [3] and other variants (e.g. [8, 27]) jointly model the unknown biases of the annotators and their answers as noisy versions of some latent ground truth. Despite their improved ground truth estimates over majority voting, recent works have shown that jointly learning the classifier model and the annotators noise model using EM-style algorithms generally leads to improved results [13, 16, 1].

In this paper, we begin by describing an EM algorithm for learning deep neural networks from crowds in multi-class classification settings. Then, a novel *crowd layer* is proposed, which allows us to train neural networks end-to-end, directly from the noisy labels of multiple annotators, using backpropagation. This not only allows us to avoid the additional computational overhead of EM, but also leads to a general-purpose framework that generalizes well beyond classification settings. Empirically, the proposed crowd layer is shown to be able to automatically distinguish the good from the unreliable annotators and capture their individual biases, thus achieving new state-of-the-art results in real data from Amazon Mechanical Turk for image classification, text regression and named entity recognition. As our experiments show, when compared to the more complex EM-based approaches for classification, the crowd layer is able to achieve comparable or even superior results.

2 Related work

The increasing popularity of crowdsourcing as a way to label large collections of data in an inexpensive and scalable manner has led to much interest of the machine learning community in developing methods to address the noise and trustworthiness issues associated with it. In this direction, one of the key early contributions is the work of Dawid and Skene [3], who proposed an EM algorithm to obtain point estimates of the error rates of patients given repeated but conflicting responses to medical questions. This work was the basis for many other variants for aggregating labels from multiple annotators with different levels of expertise, such as the one proposed in [27], which further extends Dawid and Skene's model by also accounting for item difficulty in the context of image classification. Similarly, Ipeirotis et al. [8] propose using Dawid and Skene's approach to extract a single quality score for each worker that allows to prune low-quality workers. The approaches proposed in our paper contrast with this line of work, by allowing neural networks to be trained directly on the noisy labels of multiple annotators, thereby avoiding the need to resort to prior label aggregation schemes.

Despite the generality of label aggregation approaches described above, which can be used in combination with any type of machine learning algorithm, they are sub-optimal when compared to approaches that also jointly learn the classifier itself. One of the most prominent works in this direction is the one of Raykar et al. [13], who proposed an EM algorithm for jointly learning the levels of expertise of different annotators and the parameters of a logistic regression classifier, by modeling the ground truth labels as latent variables. This idea was later extended to other types of models such as Gaussian process classifiers [16], supervised latent Dirichlet allocation [14] and, more recently, to convolutional neural networks with softmax outputs [1]. In this paper, we describe a generalization of the approach in [1] to multi-class settings, highlighting some of the technical difficulties associated with it. Then, a novel type of neural network layer is proposed, which allows the training of deep neural networks directly from the noisy labels of multiple annotators using backpropagation. This contrasts with most of works in the literature, which rely on more complex iterative procedures based on EM. Furthermore, the simplicity of the proposed approach allows for straightforward extensions to regression and structured prediction problems.

Regarding applications areas for multiple-annotator learning, they vary widely. Some of the most popular ones include: image classification [23, 26], computer-aided diagnosis/radiology [13, 6], object detection [25, 17], text classification [14], natural language processing [24, 15] and speech-related tasks [11]. In this paper, we will use data from some of these areas to evaluate different approaches. Given that these are precisely some of the areas that have seen the most dramatic improvements due to recent contributions in deep learning [10, 20], we believe that developing novel efficient algorithms for learning deep neural networks from crowds is of great importance to the field.

3 EM algorithm for deep learning from crowds

Let $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N$ be a dataset of size N, where for each input vector $\mathbf{x}_n \in \mathbb{R}^D$ we are given a vector of crowdsourced labels $\mathbf{y}_n = \{y_n^r\}_{r=1}^R$, with y_n^r representing the label provided by the r^{th} annotator in a set of R annotators. Following the ideas in [13, 28, 14], we shall assume the existence of a latent true class z_n whose value is, in this particular case, determined by a softmax output layer of a deep neural network parameterized by $\mathbf{\Theta}$, and that each annotator then provides a noisy version of z_n according to $p(y_n^r|z_n, \mathbf{\Pi}^r) = Multinomial(y_n^r|\boldsymbol{\pi}_{z_n}^r)$. This formulation corresponds to keeping a per-annotator confusion matrix $\mathbf{\Pi}^r = (\boldsymbol{\pi}_1^r, \ldots, \boldsymbol{\pi}_C^r)$ to model their expertise, where C denotes the number of classes. Further assuming that annotators provide labels independently of each other, we can write the complete-data likelihood as

$$p(\mathcal{D}, \mathbf{z}|\mathbf{\Theta}, \mathbf{\Pi}^1, \ldots, \mathbf{\Pi}^R) = \prod_{n=1}^N p(z_n|\mathbf{x}_n, \mathbf{\Theta}) \prod_{r=1}^R p(y_n^r|z_n, \mathbf{\Pi}^r). \tag{1}$$

Based on this formulation, we derive an Expectation-Maximization (EM) algorithm for jointly learning the reliabilities of the annotators $\mathbf{\Pi}^r$ and the parameters of the neural network $\mathbf{\Theta}$. The expected-value of the complete-data log-likelihood under a current estimate of the posterior distribution over latent variables $q(z_n)$ is given by

$$\mathbb{E}\Big[\ln p(\mathcal{D}, \mathbf{z}|\mathbf{\Theta}, \mathbf{\Pi}^1, \ldots, \mathbf{\Pi}^R)\Big] = \sum_{n=1}^N \sum_{z_n} q(z_n) \ln \left(p(z_n|\mathbf{x}_n, \mathbf{\Theta}) \prod_{r=1}^R p(y_n^r|z_n, \mathbf{\Pi}^r) \right), \tag{2}$$

where the posterior $q(z_n)$ is obtained by making use of Bayes' theorem given the previous estimate of the model parameters $\{\mathbf{\Theta}_{\text{old}}, \mathbf{\Pi}_{\text{old}}^1, \ldots, \mathbf{\Pi}_{\text{old}}^R\}$, yielding

$$q(z_n = c) \propto p(z_n = c|\mathbf{x}_n, \mathbf{\Theta}_{\text{old}}) \prod_{r=1}^R p(y_n^r|z_n = c, \mathbf{\Pi}_{\text{old}}^r). \tag{3}$$

This corresponds to the E-step of EM. In the M-step, we find the new maximum likelihood for the model parameters. The update for the annotators' reliability parameters is given by

$$\pi_{c,l}^r = \frac{\sum_{n=1}^N q(z_n = c)\, \mathbb{I}(y_n^r = l)}{\sum_{n=1}^N q(z_n = c)}, \tag{4}$$

where $\mathbb{I}(y_n^r = l)$ is an indicator function that takes the value 1 when $y_n^r = l$, and zero otherwise. In practice, since crowd annotators typically only label a small portion of the data, it is particularly important to impose Dirichlet priors on each $\boldsymbol{\pi}_c^r$ and compute MAP estimates instead, in order to avoid numerical issues.

As for estimating the parameters of the deep neural network $\mathbf{\Theta}$, we follow the approach in [1] and use the noise-adjusted ground-truth estimates $q(z_n)$ to backpropagate the error through the network using standard stochastic optimization techniques such as stochastic gradient descent (SGD) or Adam [9]. Kindly notice, how this raises the important question of how to schedule the EM steps. If we perform one EM iteration per mini-batch, we risk not having enough evidence to estimate the annotators reliabilities. On the other hand, if we run SGD or Adam until convergence, then the computational overhead of EM becomes quite large. In practice, we found that, typically, one EM iteration per training epoch provides good computational efficiency without compromising accuracy. However, this seems to vary among different datasets, thus making it hard to tune in practice.

One key fundamental aspect for the development of this EM approach was the probabilistic interpretation of the softmax output layer of deep neural networks for classification. Unfortunately, such probabilistic interpretation is typically not available when considering, for example, continuous output variables, thereby making it difficult to generalize this approach to regression problems. Furthermore, notice that if the target variable is a sequence (or any other structured prediction output), then the marginalization over the latent variables in (2) quickly become intractable, as the number of possible label sequences to sum over grows exponentially with the length of the sequence.

4 Crowd layer

In this section, we propose the *crowd layer*: a special type of network layer that allows us to train deep neural networks directly from the noisy labels of multiple annotators, thereby avoiding some of the aforementioned limitations of EM-based approaches for learning from crowds. The intuition is rather simple. The crowd layer takes as input what would normally be the output layer of a deep neural network (e.g. softmax for classification, or linear for regression), and learns an annotator-specific mapping from the output layer to the labels of the different annotators in the crowd that captures the annotator reliabilities and biases. In this way, the former output layer becomes a bottleneck layer that is shared among the different annotators. Figure 1 illustrates this bottleneck structure in the context of a simple convolutional neural network for classification problems with 4 classes and R annotators.

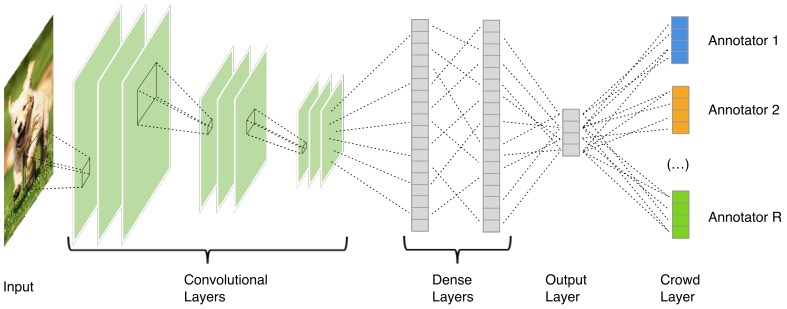


Figure 1: Bottleneck structure for a CNN for classification with 4 classes and R annotators.

The idea is then that when using the labels of a given annotator to propagate errors through the whole neural network, the crowd layer adjusts the gradients coming from the labels of that annotator according to its reliability by scaling them and adjusting their bias. In doing so, the bottleneck layer of the network now receives adjusted gradients from the different annotators' labels, which it aggregates and backpropagates further through the rest of the network. As it turns out, through this crowd layer, the network is able to account for unreliable annotators and even correct eventual systematic biases in their labeling. Moreover, all of that can be done naturally within the backpropagation framework.

Formally, let $\boldsymbol{\sigma}$ be the output of a deep neural network with an arbitrary structure. Without loss of generality, we shall assume the vector $\boldsymbol{\sigma}$ to correspond to the output of a softmax layer, such that σ_c corresponds to the probability of the input instance belonging to class c. The activation of the crowd layer for each annotator r is then defined as $\mathbf{a}^r = f_r(\boldsymbol{\sigma})$, where f_r is an annotator-specific function, and the output of the crowd layer simply as the softmax of the activations $\mathbf{o}^r = e^{a_c^r} / \sum_{l=1}^{C} e^{a_l^r}$.

The question is then how to define the function mapping $f_r(\boldsymbol{\sigma})$. In the experiments section, we study different alternatives. For classification problems a reasonable assumption is to consider a matrix transformation, such that $f_r(\boldsymbol{\sigma}) = \mathbf{W}^r \boldsymbol{\sigma}$, where $\mathbf{W}^r$ is an annotator-specific matrix. Given a cost function $E(\mathbf{o}^r, y^r)$ between the expected output of the r^{th} annotator and its actual label y^r, we can compute the gradients $\partial E / \partial \mathbf{a}^r$ at the activation $\mathbf{a}^r$ for each annotator and backpropagate them to the bottleneck layer, leading to

$$\frac{\partial E}{\partial \boldsymbol{\sigma}} = \sum_{r=1}^{R} \mathbf{W}^r \frac{\partial E}{\partial \mathbf{a}^r}. \tag{5}$$

The gradient vector at the bottleneck layer then naturally becomes a weighted sum of gradients according to the labels of the different annotators. Moreover, if the annotator is likely to mislabel class c as class l, then the matrix $\mathbf{W}^r$ can actually adjust the gradients accordingly. The problem of missing labels from some of the annotators can be easily addressed by setting their gradient contributions to zero. As for estimating the annotator weights $\{\mathbf{W}^r\}_{r=1}^R$, since they parameterize the mapping from the output of the bottleneck layer $\boldsymbol{\sigma}$ to the annotators labels $\{\mathbf{o}^r\}_{r=1}^R$, they can be estimated using standard stochastic optimization techniques such as SGD or Adam [9]. Once the network is trained, the crowd layer can be removed, thus exposing the output of bottleneck layer $\boldsymbol{\sigma}$, which can readily be used to make predictions for unseen instances.

An obvious concern with the approach described above is identifiability. Therefore, it is essential to not over-parametrize $f_r(\boldsymbol{\sigma})$, since adding parameters beyond necessary can make the output of

the bottleneck layer $\boldsymbol{\sigma}$ lose its interpretability as a shared estimated ground truth. Another important aspect is parameter initialization. In our experiments, we found that the best practice is to initialize the crowd layer with identities, i.e. zeros for additive parameters, ones for scalar parameters, identity matrix for multiplicative matrices, etc.

A particularly important aspect to note, is that the framework described above is quite general. For example, it can be straightforwardly applied to sequence labeling problems without further changes, or be adapted to regression problems by considering univariate scalar and bias parameters per annotator in the crowd layer.

5 Experiments

The proposed crowd layer was implemented as a new type of layer in Keras[3], so that using it in practice requires only a single line of code. Similarly, the EM-based approach described in Section 3 was also implemented in Keras. We evaluate the proposed crowd layer in 3 tasks: image classification, text regression and named entity recognition.

5.1 Image classification

We begin by evaluating the proposed crowd layer in a more controlled setting, by using simulated annotators with different levels of expertise on a large image classification dataset consisting of 25000 images of dogs and cats from Kaggle[4], where the goal is to distinguish between the two species. Let the dog and cat classes be represented by 1 and 0, respectively. Since this is a binary classification task, we can easily simulate annotators with different levels of expertise by assigning them individual sensitivities α^r and specificities β^r, and sampling their answers from a Bernoulli distribution with parameter α^r if the true label is 1, and from a Bernoulli distribution with parameter β^r otherwise. Using this procedure, we simulated 5 annotators with the following values of $\alpha^r = [0.6, 0.9, 0.5, 0.9, 0.9]$ and $\beta^r = [0.7, 0.8, 0.5, 0.2, 0.9]$.

For this particular problem we used a fairly standard CNN architecture with 4 convolutional layers with 3x3 patches. The first 2 convolutional layers compute 32 features and the other 2 compute 64 features. Each layer is followed by a 2x2 max pooling operation and all layers use ReLU activation functions. The output of the convolutional layers is then fed to a fully-connected (FC) layer with 128 ReLU units and finally goes to an output layer with a softmax activation. We use batch normalization [7] and apply 50% dropout between the FC and output layers. The proposed approach further adds a crowd layer on top of the softmax output layer during training. The base architecture was selected from a set of possible configurations using the true labels by optimizing the accuracy on a validation set (consisting of 20% of the train set) through random search. It is important to note that it is supposed to be representative of a set of typical approaches for image classification rather than being the single best possible architecture in the literature for this particular dataset. Furthermore, our main interest in this paper is the contribution of the crowd layer to the training of the neural network.

The proposed CNN with a crowd layer (referred to as "DL-Crowds") is compared with: a CNN trained on the result of majority voting - "DL (Maj. Voting)"; a CNN trained on the output of the label aggregation approach proposed by Dawid and Skene [3] - "DL (Dawid & Skene)"; and a CNN trained using the EM approach described in Section 3 - "DL-EM". As a reference point, we also compare with a CNN trained on true labels - "DL (True)". We consider 3 variants of the proposed crowd layer with different annotator-specific functions f_r with increasing number of parameters: a vector of per-class weights $\mathbf{w}^r$, such that $f_r(\boldsymbol{\sigma}) = \mathbf{w}^r \odot \boldsymbol{\sigma}$ (referred to as "VW"); a vector of per-class biases $\mathbf{b}^r$, such that $f_r(\boldsymbol{\sigma}) = \boldsymbol{\sigma} + \mathbf{b}^r$ ("VB"); and a version with a matrix of weights $\mathbf{W}^r$, such that $f_r(\boldsymbol{\sigma}) = \mathbf{W}^r\boldsymbol{\sigma}$ ("MW"). In our experiments, we found that for approaches with more parameters than MW, such as $f_r(\boldsymbol{\sigma}) = \mathbf{W}^r\boldsymbol{\sigma} + \mathbf{b}^r$, identifiability issues start to occur.

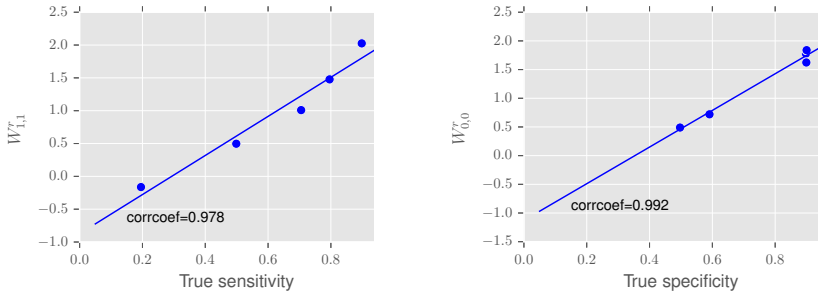
Of the 25000 images in the Dogs vs Cats dataset, 50% were used for training and the remaining for testing the different approaches. In order to account for the effect of the random initialization that is used for most of the parameters in the network, we performed 30 executions of all approaches and report their average accuracies in Table 1. We can immediately verify that both the EM-based and the crowd layer approaches significantly outperform the majority voting and Dawid & Skene baselines,

[3] https://keras.io/

[4] https://www.kaggle.com/c/dogs-vs-cats

Table 1: Accuracy results for classification datasets: Dogs vs. Cats and LabelMe.

Method	Dogs vs. Cats	LabelMe (MTurk)
sLDA (Maj. Voting) [14]	-	70.100 ($\pm$ 0.413)
MA-sLDAc [14]	-	78.120 ($\pm$ 0.397)
DL (Maj. Voting)	71.377 ($\pm$ 1.123)	76.744 ($\pm$ 1.208)
DL (Dawid & Skene)	76.750 ($\pm$ 1.282)	80.792 ($\pm$ 1.066)
DL-EM	80.184 ($\pm$ 1.454)	82.677 ($\pm$ 0.981)
DL-Crowds (VW)	79.534 ($\pm$ 1.064)	81.051 ($\pm$ 0.899)
DL-Crowds (VW+B)	79.688 ($\pm$ 1.406)	81.886 ($\pm$ 0.893)
DL-Crowds (MW)	80.265 ($\pm$ 1.230)	83.151 ($\pm$ 0.877)
DL (True)	84.912 ($\pm$ 1.248)	90.038 ($\pm$ 0.652)



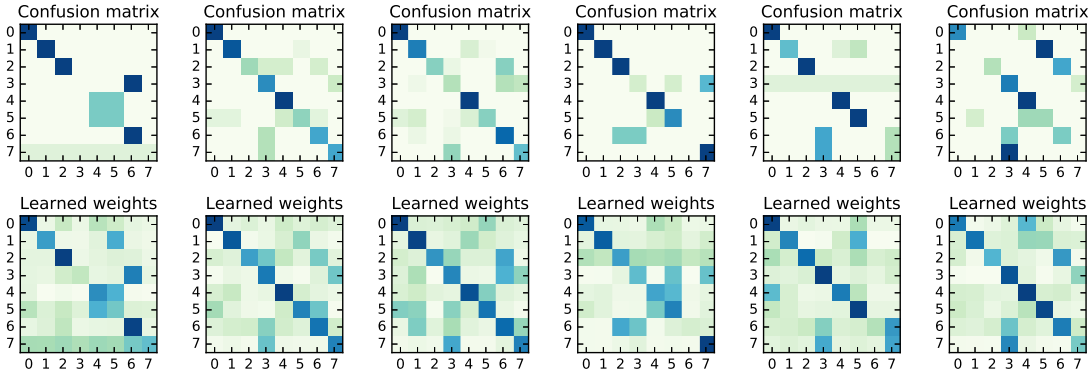
(a) Sensitivities (b) Specificities

Figure 2: Comparison between the true sensitivities and specificities of the annotators and the diagonal elements of their respective weight matrices $\mathbf{W}^r$ for the Dogs vs. Cats dataset.

thus demonstrating the gain of learning from the answers of multiple annotators directly rather than relying on aggregation schemes prior to training. Regarding the different variants of the proposed crowd layer, we can verify that they all provide comparable results, but being the MW approach the one that gives the best average accuracy. In order to better understand what the MW approach is doing, we inspected the weight matrices $\mathbf{W}^r$ of each annotator r. Figure 2 shows the relationship between the diagonal elements of $\mathbf{W}^r$ and the sensitivities and specificities of the corresponding annotators, highlighting the strong linear correlation between the two. This evidences that the proposed crowd layer is able to internally represent the reliabilities of the annotators.

Having verified that the crowd layer was performing well for simulated annotators, we then moved on to evaluating it in real data from Amazon Mechanical Turk (AMT). For this purpose, we used the image classification dataset from [14] adapted from part of the LabelMe data [18], whose goal is to classify images according to 8 classes: “highway", “inside city", “tall building", “street", “forest", “coast", “mountain" or “open country". It consists of a total of 2688 images, where 1000 of them were used to obtain labels from multiple annotators from Amazon Mechanical Turk. Each image was labeled by an average of 2.547 workers, with a mean accuracy of 69.2%. The remaining 1688 images were using for evaluating the different approaches.

Since the training set is rather small, we use the pre-trained CNN layers of the VGG-16 deep neural network [22] and apply only one FC layer (with 128 units and ReLU activations) and one output layer on top, using 50% dropout. The last column of Table 1 shows the obtained results. We can once more verify that the EM-based and crowd layer approaches outperform the majority voting and Dawid & Skene baselines, and also the probabilistic approaches proposed in [14] based on supervised latent Dirichlet allocation (sLDA). However, unlike for the Dogs vs. Cats dataset, the differences between the different function mappings f_r for crowd layer become more evident. This can be justified by the ability of the MW version to be able to model the biases of the annotators. Indeed, if we compare the learned weight matrices $\mathbf{W}^r$ with the respective confusion matrices of the annotators, we can notice how they resemble each other. Figure 3 shows this comparison for 6 annotators, where the



(a) annotator 1 (b) annotator 9 (c) annotator 20 (d) annotator 23 (e) annotator 39 (f) annotator 45

Figure 3: Comparison between the weight matrices $\mathbf{W}^r$ and the corresponding confusion matrices.

Table 2: Results for MovieReviews (MTurk) dataset.

Method	MAE	RMSE	R^2
sLDA (Mean) [14]	-	-	31.573 ($\pm$ 1.647)
MA-sLDAr [14]	-	-	35.553 ($\pm$ 1.282)
DL (Mean)	1.215 ($\pm$ 0.048)	1.498 ($\pm$ 0.050)	31.496 ($\pm$ 4.690)
DL-Crowds (S)	1.228 ($\pm$ 0.041)	1.508 ($\pm$ 0.044)	30.560 ($\pm$ 4.101)
DL-Crowds (S+B)	1.163 ($\pm$ 0.031)	1.440 ($\pm$ 0.033)	37.086 ($\pm$ 2.407)
DL-Crowds (B)	1.130 ($\pm$ 0.025)	1.411 ($\pm$ 0.028)	39.276 ($\pm$ 2.374)
DL (True)	1.050 ($\pm$ 0.029)	1.330 ($\pm$ 0.036)	45.983 ($\pm$ 2.895)

color intensity of the cells increases with the relative magnitude of the value, thus demonstrating that the crowd layer is able to learn the labeling patterns of the annotators.

5.2 Text regression

As previously mentioned, one of the key advantages of the proposed crowd layer is its straightforward extension to other types of target variables. In this section, we consider a regression problem based on the dataset introduced in [14]. The dataset consists of 5006 movie reviews, where the goal is to predict the rating given to the movie (on a scale of 1 to 10) based on the text of the review. Using AMT, the authors collected an average of 4.96 answers from a pool of 137 workers for 1500 movie reviews. The remaining 3506 reviews were used for testing. Letting the (continuous) output of the bottleneck layer be denoted μ, we considered 3 variants of the proposed crowd layer with different annotator-specific functions f_r: a per-annotator scale parameter s^r, such that $f_r(\mu) = s^r \mu$ (referred to as "S"); a per-annotator bias parameter b^r, such that $f_r(\mu) = \mu + b^r$ ("B"); and a version with both: $f_r(\mu) = s^r \mu + b^r$ ("S+B").

The neural network architecture used for this problem consists of a 3x3 convolutional layer with 128 features and 5x5 max pooling, a 5x5 convolutional layer with 128 features and 5x5 max pooling, and a FC layer with 32 hidden units. All layers, except for the output one, use ReLU activations. Table 2 shows the obtained results for 30 runs of the different approaches. These results clearly show that the proposed crowd layer, particularly the "B" variant, clearly outperforms a neural network trained on the mean answer of the annotators - "DL (Mean)" - as well as all the approaches proposed in [14] based on supervised LDA. In order to better understand what the crowd layer in the "B" variant is doing, we plotted learned b^r values in comparison with the true biases of the annotators according to the ground truth. Figure 4 shows this comparison, in which we can verify that the learned values of b^r are highly correlated with the true biases of the annotators, thus showing that crowd layer is able to account for annotator bias when learning from the noisy labels of multiple annotators.

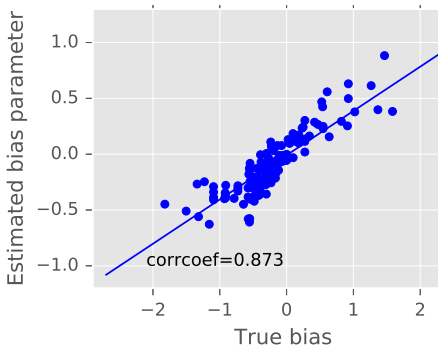


Figure 4: Relationship between the learned b^r parameters and the true biases of the annotators.

Table 3: Results for CoNLL-2003 NER (MTurk) dataset.

Method	Precision	Recall	F1
CRF (Maj. Voting) [15]	0.455	0.809	0.582
CRF-MA [15]	0.494	0.856	0.626
DL (Maj. Voting)	0.664 ($\pm$ 0.017)	0.464 ($\pm$ 0.021)	0.546 ($\pm$ 0.014)
DL-Crowds (VW)	0.709 ($\pm$ 0.013)	0.472 ($\pm$ 0.020)	0.566 ($\pm$ 0.016)
DL-Crowds (VW+B)	0.603 ($\pm$ 0.013)	0.609 ($\pm$ 0.012)	0.606 ($\pm$ 0.007)
DL-Crowds (MW)	0.660 ($\pm$ 0.018)	0.593 ($\pm$ 0.013)	0.624 ($\pm$ 0.010)
DL (True)	0.711 ($\pm$ 0.013)	0.740 ($\pm$ 0.009)	0.725 ($\pm$ 0.008)

5.3 Named entity recognition

Lastly, we evaluated the proposed crowd layer on a named entity recognition (NER) task. For this purpose, we used the AMT dataset introduced in [15] which is based on the 2003 CoNLL shared NER task [19], where the goal is to identify the named entities in the sentence and classify them as persons, locations, organizations or miscellaneous. The dataset consists of 5985 labeled sentences using a pool of 47 workers. The remaining 3250 sentences of the original dataset were used for testing. The neural network architecture used for this problem consists of a layer of 300-dimensional word embeddings initialized with the pre-trained weights of GloVe [12], followed by a 5x5 convolutional layer with 512 features, whose output is then fed to a GRU cell with a 50-dimensional hidden state. The individual hidden states of the GRU are then passed to a FC layer with a softmax activation. The crowd layer uses the same annotator function mappings f_r used for image classification. Table 3 shows the obtained average results over 30 runs. These results clearly demonstrate that the proposed approach significantly outperforms the use of majority voting, and provides similar results to the multi-annotator approach based on conditional random fields (CRF-MA) proposed in [15], while reducing the training time by at least one order of magnitude when compared to the latter.

6 Conclusion

This paper proposed the *crowd layer* - a novel neural network layer that enables us to train deep neural networks end-to-end, directly from the labels of multiple annotators and crowds, using backpropagation. Despite its simplicity, the crowd layer is able to capture the reliabilities and biases of the different annotators and adjust the error gradients that are backpropagated during training accordingly. As our empirical evaluation shows, the proposed approach significantly outperforms other approaches that rely on the aggregation of the annotators' answers prior to training, and provides results that are comparable to more complex, harder to setup and more computationally demanding EM-based approaches. Furthermore, unlike the latter, the crowd layer is easily generalizable beyond classification problems, which we empirically demonstrate using real data from Amazon Mechanical Turk for text regression and named entity recognition tasks.

References

[1] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab. Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE transactions on medical imaging*, 35(5):1313–1321, 2016.

[2] M. Buhrmester, T. Kwang, and S. D. Gosling. Amazon's mechanical turk a new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[3] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *J. of the Royal Statistical Society. Series C*, 28(1):20–28, 1979.

[4] P. Donmez and J. Carbonell. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proc. of the 17th ACM Conf. on Information and Knowledge Management*, pages 619–628, 2008.

[5] B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[6] H. Greenspan, B. van Ginneken, and R. M. Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[8] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.

[9] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[11] G. Parent and M. Eskenazi. Speaking to the crowd: Looking at past achievements in using crowdsourcing for speech and predicting future challenges. In *INTERSPEECH*, pages 3037–3040. Citeseer, 2011.

[12] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[13] V. Raykar, S. Yu, L. Zhao, G. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from Crowds. *J. Mach. Learn. Res*, pages 1297–1322, 2010.

[14] F. Rodrigues, M. Lourenço, B. Ribeiro, and F. Pereira. Learning supervised topic models for classification and regression from crowds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[15] F. Rodrigues, F. Pereira, and B. Ribeiro. Sequence labeling with multiple annotators. *Machine Learning*, pages 1–17, 2013.

[16] F. Rodrigues, F. Pereira, and B. Ribeiro. Gaussian process classification and active learning with multiple annotators. In *Proc. of the 31st Int. Conf. on Machine Learning*, pages 433–441, 2014.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[18] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: a database and web-based tool for image annotation. *Int. J. of Computer Vision*, 77(1-3):157–173, 2008.

[19] E. Sang and F. D. Meulder. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL*, volume 4, pages 142–147, 2003.

[20] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[21] V. Sheng, F. Provost, and P. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining*, pages 614–622, 2008.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of venus images. In *Advances in Neural Information Processing Systems*, pages 1085–1092, 1995.

[24] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast - but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 254–263, 2008.

[25] H. Su, J. Deng, and L. Fei-Fei. Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, volume 1, 2012.

[26] P. Welinder, S. Branson, P. Perona, and S. Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

[27] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043, 2009.

[28] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy. Learning from multiple annotators with varying expertise. *Mach. Learn.*, 95(3):291–327, 2014.