Learning to Learn from Weak Supervision by Full Supervision

Dehghani, M.; Severyn, A. ; Rothe, S.; Kamps, J.

Publication date
2017

Document Version
Final published version

Learning to Learn from Weak Supervision by Full Supervision

Mostafa Dehghani
University of Amsterdam
dehghani@uva.nl

Aliaksei Severyn, Sascha Rothe
Google Research
{severyn,rothe}@google.com

Jaap Kamps
University of Amsterdam
kamps@uva.nl

Abstract

In this paper, we propose a method for training neural networks when we have a large set of data with weak labels and a small amount of data with true labels. In our proposed model, we train two neural networks: a *target network*, the learner and a *confidence network*, the meta-learner. The *target network* is optimized to perform a given task and is trained using a large set of unlabeled data that are weakly annotated. We propose to control the magnitude of the gradient updates to the *target network* using the scores provided by the second *confidence network*, which is trained on a small amount of supervised data. Thus we avoid that the weight updates computed from noisy labels harm the quality of the *target network* model.

1 Introduction

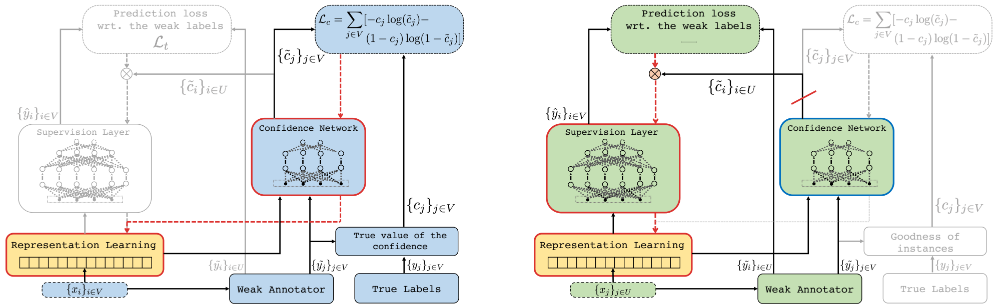
Using weak or noisy supervision is a straightforward approach to increase the size of the training data [Dehghani et al., 2017b, Patrini et al., 2016, Beigman and Klebanov, 2009, Zeng et al., 2015, Bunescu and Mooney, 2007]. The output of heuristic methods can be used as weak or noisy signals along with a small amount of labeled data to train neural networks. This is usually done by pre-training the network on weak data and fine tuning it with true labels [Dehghani et al., 2017b, Severyn and Moschitti, 2015a]. However, these two independent stages do not leverage the full capacity of information from true labels and using noisy labels of lower quality often brings little to no improvement. This issue is tackled by noise-aware models where denoising the weak signal is part of the learning process [Patrini et al., 2016, Sukhbaatar et al., 2014, Dehghani et al., 2017a].

In this paper, we propose a method that leverages a small amount of data with true labels along with a large amount of data with weak labels. In our proposed method, we train two networks in a multi-task fashion: a *target network* which uses a large set of weakly annotated instances to learn the main task while a *confidence network* is trained on a small human-labeled set to estimate confidence scores. These scores define the magnitude of the weight updates to the *target network* during the back-propagation phase. From a meta-learning perspective [Andrychowicz et al., 2016, Finn et al., 2017, Ravi and Larochelle, 2016], the goal of the *confidence network*, as the meta-learner, trained jointly with the *target network*, as the learner, is to calibrate the learning rate of the *target network* for each instance in the batch. I.e., the weights $\boldsymbol{w}$ of the *target network* f_w at step $t+1$ are updated as follows:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\eta_t}{b} \sum_{i=1}^{b} c_\theta(x_i, \tilde{y}_i) \nabla \mathcal{L}(f_{\boldsymbol{w_t}}(x_i), \tilde{y}_i) \tag{1}$$

where η_t is the global learning rate, $\mathcal{L}(\cdot)$ is the loss of predicting $\hat{y} = f_w(x_i)$ for an input x_i when the label is $\tilde{y}$; $c_\theta(\cdot)$ is a scoring function learned by the *confidence network* taking input instance x_i and its noisy label $\tilde{y}_i$. Thus, we can effectively control the contribution to the parameter updates for the *target network* from weakly labeled instances based on how reliable their labels are according to the *confidence network* (learned on a small supervised data).

NIPS Workshop on Meta-Learning (MetaLearn 2017), Long Beach, CA, USA.



(a) Full Supervision Mode: Training on batches of data with true labels. (b) Weak Supervision Mode: Training on batches of data with weak labels.

Figure 1: Our proposed multi-task network for learning a target task using a large amount of weakly labeled data and a small amount of data with true labels. Faded parts of the network are disabled during the training in the corresponding mode. Red-dotted arrows show gradient propagation. Parameters of the parts of the network in red frames get updated in the backward pass, while parameters of the network in blue frames are fixed during the training.

Our approach is similar to [Andrychowicz et al., 2016], where a separate recurrent neural network called *optimizer* learns to predict an optimal update rule for updating parameters of the *target network*. The optimizer receives a gradient from the *target network* and outputs the adjusted gradient matrix. As the number of parameters in modern neural networks is typically on the order of millions the gradient matrix becomes too large to feed into the optimizer, so the approach presented in [Andrychowicz et al., 2016] is applied to very small models. In contrast, our approach leverages additional weakly labeled data where we use the *confidence network* to predict per-instance scores that calibrate gradient updates for the *target network*.

Our setup requires running a weak annotator to label a large amount of unlabeled data, which is done at pre-processing time. For many tasks, it is possible to use a simple heuristic to generate weak labels. This set is then used to train the *target network*. In contrast, a small human-labeled set is used to train the *confidence network*, which estimates how good the weak annotations are, i.e. controls the effect of weak labels on updating the parameters of the *target network*. This helps to alleviate updates from instances with unreliable labels that may corrupt the *target network*.

In this paper, we study our approach on sentiment classification task.Our experimental results suggest that the proposed method is more effective in leveraging large amounts of weakly labeled data compared to traditional fine-tuning. We also show that explicitly controlling the *target network* weight updates with the *confidence network* leads to faster convergence.

2 The Proposed Method

In the following, we describe our recipe for training neural networks, in a scenario where along with a small human-labeled training set a large set of weakly labeled instances is leveraged. Formally, given a set of unlabeled training instances, we run a weak annotator to generate noisy labels. This gives us the training set U. It consists of *tuples* of training instances x_i and their weak labels $\tilde{y}_i$, i.e. $U = \{(x_i, \tilde{y}_i), \ldots\}$. For a small set of training instances with true labels, we also apply the weak annotator to generate weak labels. This creates the training set V, consisting of *triplets* of training instances x_j, their weak labels $\tilde{y}_j$, and their true labels y_j, i.e. $V = \{(x_j, \tilde{y}_j, y_j), \ldots\}$. We can generate a large amount of training data U at almost no cost using the weak annotator. In contrast, we have only a limited amount of data with true labels, i.e. $|V| << |U|$.

In our proposed framework we train a multi-task neural network that jointly learns the confidence score of weak training instances and the main task using controlled supervised signals. The high-level representation of the model is shown in Figure 1: it comprises two neural networks, namely the *confidence network* and the *target network*. The goal of the *confidence network* is to *estimate the confidence score* $\tilde{c}_j$ of training instances. It is learned on triplets from training set V: input x_j, its weak label $\tilde{y}_j$, and its true label y_j. The score $\tilde{c}_j$ is then used to control the effect of weakly annotated training instances on updating the parameters of the *target network*.

The *target network* is in charge of *handling the main task* we want to learn. Given the data instance, x_i and its weak label $\tilde{y}_i$ from the training set U, the *target network* aims to predict the label $\hat{y}_i$. The *target network* parameter updates are based on noisy labels assigned by the weak annotator, but the magnitude of the gradient update is based on the output of the *confidence network*.

Both networks are trained in a multi-task fashion alternating between the *full supervision* and the *weak supervision* mode. In the *full supervision* mode, the parameters of the *confidence network* get updated using batches of instances from training set V. As depicted in Figure 1b, each training instance is passed through the representation layer mapping inputs to vectors. These vectors are concatenated with their corresponding weak labels $\tilde{y}_j$. The *confidence network* then estimates $\tilde{c}_j$, which is the probability of taking data instance j into account for training the *target network*.

In the *weak supervision* mode the parameters of the *target network* are updated using training set U. As shown in Figure 1a, each training instance is passed through the same representation learning layer and is then processed by the supervision layer which is a part of the *target network* predicting the label for the main task. We also pass the learned representation of each training instance along with its corresponding label generated by the weak annotator to the *confidence network* to estimate the *confidence score* of the training instance, i.e. $\tilde{c}_i$. The confidence score is computed for each instance from set U. These confidence scores are used to weight the gradient updating the *target network* parameters during back-propagation. It is noteworthy that the representation layer is shared between both networks, so the *confidence network* can benefit from the largeness of set U and the *target network* can utilize the quality of set V.

2.1 Model Training

Our optimization objective is composed of two terms: (1) the *confidence network* loss $\mathcal{L}_c$, which captures the quality of the output from the *confidence network* and (2) the *target network* loss $\mathcal{L}_t$, which expresses the quality for the main task.

Both networks are trained by alternating between the *weak supervision* and the *full supervision* mode. In the *full supervision* mode, the parameters of the *confidence network* are updated using training instance drawn from training set V. We use cross-entropy loss function for the *confidence network* to capture the difference between the predicted confidence score of instance j, i.e. $\tilde{c}_j$ and the target score c_j: $\mathcal{L}_c = \sum_{j\in V} -c_j \log(\tilde{c}_j) - (1-c_j)\log(1-\tilde{c}_j)$, The target score c_j is calculated based on the difference of the true and weak labels with respect to the main task. In the *weak supervision* mode, the parameters of the *target network* are updated using training instances from U. We use a weighted loss function, $\mathcal{L}_t$, to capture the difference between the predicted label $\hat{y}_i$ by the *target network* and target label $\tilde{y}_i$: $\mathcal{L}_t = \sum_{i\in U} \tilde{c}_i \mathcal{L}_i$, where $\mathcal{L}_i$ is the task-specific loss on training instance i and $\tilde{c}_i$ is the confidence score of the weakly annotated instance i, estimated by the *confidence network*. Note that $\tilde{c}_i$ is treated as a constant during the weak supervision mode and there is no gradient propagation to the *confidence network* in the backward pass (as depicted in Figure 1a).

We minimize two loss functions jointly by randomly alternating between full and weak supervision modes (for example, using a 1:10 ratio). During training and based on the chosen supervision mode, we sample a batch of training instances from V with replacement or from U without replacement (since we can generate as much train data for set U).

3 Experiments

In this section, we apply our method to *sentiment classification* task. This task aims to identify the sentiment (e.g., positive, negative, or neutral) underlying an individual sentence. Our *target network* is a convolutional model similar to [Deriu et al., 2017, Severyn and Moschitti, 2015a,b, Deriu et al., 2016]. In this model, the *Representation Learning Layer* learns to map the input sentence s to a dense vector as its representation. The inputs are first passed through an embedding layer mapping the sentence to a matrix $S \in \mathbb{R}^{m\times|s|}$, followed by a series of 1d convolutional layers with max-pooling. The *Supervision Layer* is a feed-forward neural network with softmax instead as the output layer which returns the probability distribution over all three classes. As the the *Weak Annotator*, for the sentiment classification task is a simple unsupervised lexicon-based method [Hamdan et al., 2013, Kiritchenko et al., 2014], which averages over predefined sentiment score of words [Baccianella et al., 2010] in the sentence. More details about the sentiment classification model and the experimental setups are provided in Appendix A and Appendix B, respectively. In the following, we briefly introduce our baselines, dataset we have used, and present results of the experiments.

Baselines. We evaluate the performance of our method compared to the following baselines: (**WA**) Weak Annotator, i.e. the unsupervised method that we used for annotating the unlabeled data. (**WSO**) Weak Supervision Only, i.e. the *target network* trained only on weakly labeled data. (**FSO**) Full Supervision Only, i.e. the *target network* trained only on true labeled data. (**WS+FT**) Weak Supervision + Fine Tuning, i.e. the *target network* trained on the weakly labeled data and fine tuned on true labeled data. (**NLI**) New Label Inference [Veit et al., 2017] is similar to our proposed neural architecture inspired by the teacher-student paradigm [Hinton et al., 2015], but instead of having the *confidence*

Table 1: Performance of the baseline models as well as our proposed method on different datasets in terms of Macro-F1. $^{\blacktriangle}$or$^{\triangledown}$indicates that the improvements or degradations with respect to weak supervision only (WSO) are statistically significant, at the 0.05 level using the paired two-tailed t-test.

Method	SemEval-14	SemEval-15
$\text{WA}_{\text{Lexicon}}$	0.5141	0.4471
WSO	0.6719	0.5606
FSO	0.6307	0.5811
WS+FT	0.7080$^{\blacktriangle}$	0.6441$^{\blacktriangle}$
NLI	0.7113$^{\blacktriangle}$	0.6433$^{\blacktriangle}$
L2LWS_{ST}	0.7183$^{\blacktriangle}$	0.6501$^{\blacktriangle}$
L2LWS	**0.7362**$^{\blacktriangle}$	**0.6626**$^{\blacktriangle}$
$\text{SemEval}^{1\text{th}}$	0.7162$^{\blacktriangle}$	0.6618$^{\blacktriangle}$

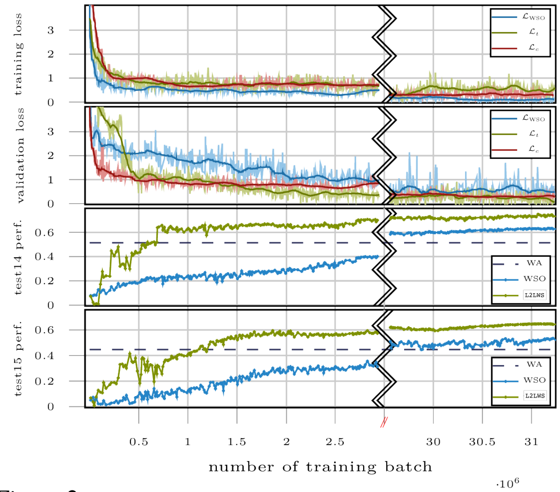


Figure 2: Loss of the *target network* ($\mathcal{L}_t$) and the *confidence network* ($\mathcal{L}_c$) compared to the loss of WSO ($\mathcal{L}_{\text{WSO}}$) on training/validation set and performance of L2LWS, WSO, and WA on test sets with respect to different amount of training data on sentiment classification.

network to predict the "confidence score" of the training instance, there is a *label generator network* which is trained on set V to map the weak labels of the instances in U to the *new labels*. The new labels are then used as the target for training the *target network*. (**L2LWS_{ST}**) Our model with different training setup: Separate Training, i.e. we consider the *confidence network* as a separate network, without sharing the representation learning layer, and train it on set V. We then train the *target network* on the controlled weak supervision signals. (**L2LWS**) Learning to Learn from Weak Supervision with Joint Training is our proposed neural architecture in which we jointly train the *target network* and the *confidence network* by alternating batches drawn from sets V and U (as explained in Section 2.1).

Data. For train/test our model, we use SemEval-13 SemEval-14, SemEval-15, twitter sentiment classification task. We use a large corpus containing 50M tweets collected during two months as unblabled set.

Results and Discussion. We report the official SemEval metric, Macro-F1, in Table 1. Based on the results, L2LWS provides a significant boost on the performance over all datasets. Typical fine tuning, i.e. WS+FT, leads to improvement over weak supervision only. The performance of NLI is worse than L2LWS as learning a mapping from imperfect labels to accurate labels and training the *target network* on new labels is essentially harder than learning to filter out the noisy labels, hence needs a lot of supervised data. L2LWS_{ST} performs worse than L2LWS since the training data V is not enough to train a high-quality *confidence network* without taking advantage of the shared representation that can be learned from the vast amount of weakly annotated data in U. We also noticed that this strategy leads to a slow convergence compared to WSO. Besides the general baselines, we also report the best performing systems, which are also convolution-based models ([Rouvier and Favre, 2016] on SemEval-14; [Deriu et al., 2016] on SemEval-15). Our proposed model outperforms the best systems.

Controlling the effect of supervision to train neural networks not only improves the performance, but also provides the network with more solid signals which speeds up the training process. Figure 2 illustrates the training/validation loss for both networks, compared to the loss of training the *target network* with weak supervision, along with their performance on test sets, with respect to different amounts of training data for the sentiment classification task. As shown, training, $\mathcal{L}_t$ is higher than $\mathcal{L}_{\text{WSO}}$, but the target labels with respect of which the loss is calculated, are weak, so regardless overfitting problem and lack of generalization, a very low loss means fitting the imperfection of the weak data. However, $\mathcal{L}_t$ in the validation decreases faster than $\mathcal{L}_{\text{WSO}}$ and compared to WSO, the performance of L2LWS on both test sets increases quickly and L2LWS passes the performance of the weak annotator by seeing fewer instances annotated by WA.

4 Conclusion

In this paper, we propose a neural network architecture that unifies learning to estimate the confidence score of weak annotations and training neural networks with controlled weak supervision. We apply the model to the sentiment classification task, and empirically verify that the proposed model speeds up the training process and obtains more accurate results.

References

Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

Eyal Beigman and Beata Beigman Klebanov. Learning with annotation noise. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 280–287. Association for Computational Linguistics, 2009.

Razvan Bunescu and Raymond Mooney. Learning to extract relations from the web using minimal supervision. In *ACL*, 2007.

Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision. *arXiv preprint arXiv:1711.00313*, 2017a.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural ranking models with weak supervision. In *SIGIR'17*, 2017b.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval*, pages 1124–1128, 2016.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th international International World Wide Web Conference (WWW'17)*, pages 1045–1052, 2017.

Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(1): 3873–3923, 2014.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Hussam Hamdan, Frederic Béchet, and Patrice Bellot. Experiments with dbpedia, wordnet and sentiwordnet as resources for sentiment analysis in micro-blogging. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pages 455–459, 2013.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762, 2014.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Semeval-2016 task 4: Sentiment analysis in twitter. *Proceedings of SemEval*, pages 1–18, 2016.

Giorgio Patrini, Alessandro Rozza, Aditya Menon, Richard Nock, and Lizhen Qu. Making neural networks robust to label noise: a loss correction approach. *arXiv preprint arXiv:1609.03683*, 2016.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 451–463, 2015.

Mickael Rouvier and Benoit Favre. Sensei-lif at semeval-2016 task 4: Polarity embedding fusion for robust sentiment analysis. *Proceedings of SemEval*, pages 202–208, 2016.

Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962. ACM, 2015a.

Aliaksei Severyn and Alessandro Moschitti. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*, pages 464–469, 2015b.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

Yuan Tang. Tf.learn: Tensorflow's high-level module for distributed machine learning. *arXiv preprint arXiv:1612.04251*, 2016.

Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *The Conference on Computer Vision and Pattern Recognition*, 2017.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pages 1753–1762, 2015.

Appendices

A Sentiment Classification Model

In the sentiment classification task, we aim to identify the sentiment (e.g., positive, negative, or neutral) underlying an individual sentence. The model we used as the *target network* is a convolutional model similar to [Deriu et al., 2017, Severyn and Moschitti, 2015a,b, Deriu et al., 2016].

Each training instance x consists of a sentence s and its sentiment label $\tilde{y}$. The architecture of the *target network* is illustrated in Figure 3. Here we describe the setup of the target network, i.e. description of the representation learning layer and the supervision layer.

The Representation Learning Layer learns a representation for the input sentence s and is shared between the *target network* and *confidence network*. It consists of an embedding function $\varepsilon : \mathcal{V} \rightarrow \mathbb{R}^m$, where $\mathcal{V}$ denotes the vocabulary set and m is the number of embedding dimensions.

This function maps the sentence to a matrix $S \in \mathbb{R}^{m \times |s|}$, where each column represents the embedding of a word at the corresponding position in the sentence. Matrix S is passed through a convolution layer. In this layer, a set of f filters is applied to a sliding window of length h over S to generate a feature map matrix O. Each feature map o_i for a given filter F is generated by $o_i = \sum_{k,j} S[i:i+h]_{k,j} F_{k,j}$, where $S[i:i+h]$ denotes the concatenation of word vectors from position i to $i+h$. The concatenation of all o_i produces a feature vector $o \in \mathbb{R}^{|s|-h+1}$. The vectors o are then aggregated over all f filters into a feature map matrix $O \in \mathbb{R}^{f \times (|s|-h+1)}$.

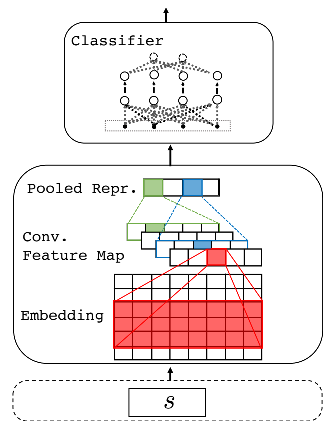


Figure 3: The *target network* for the sentiment classification task.

We also add a bias vector $b \in R^f$ to the result of a convolution. Each convolutional layer is followed by a non-linear activation function (we use ReLU) which is applied element-wise. Afterward, the output is passed to the max pooling layer which operates on columns of the feature map matrix O returning the largest value: $pool(o_i) : \mathbb{R}^{1 \times (|s|-h+1)} \rightarrow \mathbb{R}$ (see Figure 3). This architecture is similar to the state-of-the-art model for Twitter sentiment classification from Semeval 2015 and 2016 [Severyn and Moschitti, 2015b, Deriu et al., 2016].

We initialize the embedding matrix with word2vec embeddings pretrained on a collection of 50M tweets.

The Supervision Layer receives the vector representation of the inputs processed by the representation learning layer and outputs a prediction $\tilde{y}$. We opt for a simple fully connected feed-forward network with l hidden layers followed by a softmax. Each hidden layer z_k in this network computes $z_k = \alpha(W_k z_{k-1} + b_k)$, where W_k and b_k denote the weight matrix and the bias term corresponding to the k^{th} hidden layer and $\alpha(.)$ is the non-linearity. These layers follow a softmax layer which returns $\tilde{y}_i$, the probability distribution over all three classes. We employ the weighted cross entropy loss:

$$\mathcal{L}_t = \sum_{i \in B_U} \tilde{c}_i \sum_{k \in K} -\tilde{y}_i^k \log(\hat{y}_i^k), \tag{2}$$

where B_U is a batch of instances from U, and $\tilde{c}_i$ is the confidence score of the weakly annotated instance i, and K is a set of classes.

The Weak Annotator for the sentiment classification task is a simple unsupervised lexicon-based method [Hamdan et al., 2013, Kiritchenko et al., 2014]. We use SentiWordNet03 [Baccianella et al., 2010] to assign probabilities (positive, negative and neutral) for each token in set U. Then a sentence-level distribution is derived by simply averaging the distributions of the terms, yielding a noisy label $\tilde{y}_i \in \mathbb{R}^{|K|}$, where $|K|$ is the number of classes, i.e. $|K| = 3$. We empirically found that using soft labels from the weak annotator works better than assigning a single hard label. The target label c_j for the *confidence network* is calculated by using the mean absolute difference of the true label and the weak label: $c_j = 1 - \frac{1}{|K|}\sum_{k \in K} |y_j^k - \tilde{y}_j^k|$, where y_j is the one-hot encoding of the sentence label over all classes.

B Experimental Setups

The proposed architectures are implemented in TensorFlow [Tang, 2016, Abadi et al., 2015]. We use the Adam optimizer [Kingma and Ba, 2014] and the back-propagation algorithm. Furthermore, to prevent feature co-adaptation, we use *dropout* [Srivastava et al., 2014] as a regularization technique in all models.

In our setup, the *confidence network* to predict $\tilde{c}_j$ is a fully connected feed forward network. Given that the *confidence network* is learned only from a small set of true labels and to speed up training we initialize the representation learning layer with pre-trained parameters, i.e., pre-trained word embeddings. We use ReLU as a non-linear activation function α in both *target network* and *confidence network*.

Collections. We test our model on the twitter message-level sentiment classification of SemEval-15 Task 10B [Rosenthal et al., 2015]. Datasets of SemEval-15 subsume the test sets from previous editions of SemEval, i.e. SemEval-13 and SemEval-14. Each tweet was preprocessed so that URLs and usernames are masked.

Data with true labels. We use train (9,728 tweets) and development (1,654 tweets) data from SemEval-13 for training and SemEval-13-test (3,813 tweets) for validation. To make our results comparable to the official runs on SemEval we use SemEval-14 (1,853 tweets) and SemEval-15 (2,390 tweets) as test sets [Rosenthal et al., 2015, Nakov et al., 2016].

Data with weak labels. We use a large corpus containing 50M tweets collected during two months for both, training the word embeddings and creating the weakly annotated set U using the lexicon based method explained in Section A.

Parameters and Settings. We tuned hyper-parameters for each model, including baselines, separately with respect to the true labels of the validation set using batched GP bandits with an expected improvement acquisition function [Desautels et al., 2014]. The size and number of hidden layers for the classifier and the *confidence network* were separately selected from $\{32, 64, 128\}$ and $\{1, 2, 3\}$, respectively. We tested the model with both, 1 and 2 convolutional layers. The number of convolutional feature maps and the filter width is selected from $\{200, 300\}$ and $\{3, 4, 5\}$, respectively. The initial learning rate and the dropout parameter were selected from $\{1E-3, 1E-5\}$ and $\{0.0, 0.2, 0.5\}$, respectively. We considered embedding sizes of $\{100, 200\}$ and the batch size in these experiments was set to 64.