Better Fine-Tuning via Instance Weighting for Text Classification

Zhi Wang,† Wei Bi, Yan Wang, Xiaojiang Liu
1Department of Control and Systems Engineering, Nanjing University, Nanjing, China
2Tencent AI Lab, Shenzhen, China
heuyuanmingong@gmail.com, {victoriabi, brandenwang, kieranliu}@tencent.com

Abstract

Text classification is one of the fundamental tasks in Natural Language Processing (NLP) with many important applications such as sentiment analysis (Landi et al., 2016; Xu et al., 2017; Zhang, Huang, and Zhao, 2018), information extraction (Angeli et al., 2015; Manning, 2015), and topic labeling (Wang and Manning, 2016). Deep Neural Network (DNN) based classifiers have achieved state-of-the-art performance on many text classification tasks (Cao et al., 2017; Yang et al., 2017; Zhang et al., 2017). However, two limitations generally exist when these methods are used in practice: a) Classifiers are often trained from scratch; b) A large annotated dataset is often required (Lu et al., 2014; Howard and Ruder, 2018). To overcome these limitations, transfer learning is usually applied by transferring knowledge learned on a large source domain to help the classification task on the target domain.

In general, there are mainly two categories of approaches for transfer learning on DNN-based text classifiers. The first category is to transfer the source domain corpora (i.e., out-domain data), which detects instances that are close to the target domain (i.e., pseudo in-domain data), then trains a model using the augmented set including the in-domain and the pseudo in-domain data (Wang et al., 2017a,b). Such approaches involve the search of the pseudo in-domain data from a potentially very large source domain and repeatedly re-train the model using the augmented data, thus the computational efficiency may be problematic. Moreover, the scoring function to select the pseudo in-domain data often needs to be designed for each task specifically. Thus, a more popular class of approaches is to transfer the model parameters, which pre-trains a model using the source corpus and then fine-tune the parameters on the target domain (Kim et al., 2015; McCann et al., 2017). It enables a fast and efficient adaptation to the target domain without repeatedly accessing or processing the large source dataset. Considering these superiorities, we study the fine-tuning method in this paper.

Most previous literature on fine-tuning a pre-trained model focuses on how to pre-train a set of parameters that can help the target task most. For example, Pennington et al. (2014) pre-trained the popular GloVe model by leveraging the statistical information on the nonzero elements in a word-word co-occurrence matrix. Howard and Ruder (2018) pre-trained a language model using the long short-term memory network and used it to initialize the first layer of the target classifier. Based on a set of pre-trained parameters, existing methods directly apply standard optimization algorithms such as SGD or Adam (Sutskever et al., 2013; Kingma and Ba, 2014) for fine-tuning on the target domain. However, these optimization algorithms are originally designed for randomly initialized DNN models.

In this work, we propose an Instance Weighting based Fine-tuning (IW-Fit) method, which revises the fine-tuning stage to improve the classification accuracy on the target domain when a pre-trained model from the source domain is given. The idea of our method is illustrated in Figure 1. At each fine-tuning epoch, we will assign dynamic weights to target instances to accomplish two goals: 1) identify and learn the specific knowledge of the target domain effectively; 2) well preserve the shared knowledge between the source and the target domains. The designed instance weighting metrics used in IW-Fit are model-agnostic, which are easy to implement for general DNN-based classifiers. Experimental results show that IW-Fit can consistently improve the classification accuracy on the target domain.
Figure 1: Illustration of the proposed Instance Weighting based Fine-tuning (IW-Fit). The weights are dynamically adjusted based on the current classifier.

- 1) Most existing fine-tuning methods rely on a pre-defined ranking to select instances from the source domain. This can be slow and it fails to select the best instances for transfer learning.
- 2) The pre-trained model is fine-tuned with a fixed weight, which can lead to suboptimal results.

We design two model-agnostic metrics to calculate the weights for each instance in the source domain. The two designed metrics are based on two different concepts: the prediction loss and the variance of historical prediction losses. A good property of the metric is that they do not depend on the specific design of the DNN-based classifier, thus can be easily applied to any network structure.

We extensively evaluate IW-Fit on two text classification tasks under various settings: Amazon to Yelp and Yelp-2015 to Yelp-2016/Yelp-2017. Our experimental results show that IW-Fit can consistently improve the classification accuracy on the target domain. In summary, our contributions are threefold:

1. We propose Instance Weighting based Fine-tuning (IW-Fit), a method applied on the fine-tuning stage given a set of pre-trained parameters.
2. We design two model-agnostic metrics to calculate the weights used in IW-Fit and their mixing variants.
3. We perform extensive experiments to verify that IW-Fit can consistently improve the classification accuracy on the target domain over several baselines.

Related Work

With respect to the contribution of this paper, we discuss two threads of related work in this section: DNN-based fine-tuning methods and methods that involve the idea of instance weighting in NLP tasks.

DNN-based fine-tuning methods transfer the model parameters from a pre-trained model to the target domain for fine-tuning. In many state-of-the-art models for various NLP tasks, the popular pre-trained word embeddings from the Skip-grams model (Mikolov et al., 2013) or the GloVe (Pennington, Socher, and Manning, 2014) were usually utilized as an initialization. Johnson and Zhang (2017) improved the pre-trained region embedding by extending ShallowCNN with unsupervised training on the extra unlabeled data. Felbo et al. (2017) sequentially unfroze and fine-tuned a single layer at a time to enhance a regularizing effect at the expense of extra computation. Howard and Ruder (2018) performed a two-step approach by fine-tuning the pre-trained language model first and then the entire classifier on the target domain. Similarly, Chung, Lee, and Glass (2018) pre-trained a complex question-answering (QA) model on a large source domain and transferred all model parameters to the target domain for fine-tuning.

As stated in the introduction, the above methods focus on the pre-training stage. In our work, we try to improve the fine-tuning stage with the idea of instance weighting, which can be combined with all these methods complimentarily.

Another thread of work is about using the idea of instance weighting in various NLP tasks. Previous approaches assigned each instance/domain a weight relying on designed rules or statistical methods. Jiang and Zhai (2007) iteratively pruned instances across domains for better domain adaptation under a semi-supervised learning setting. Neural machine translation domain adaptation methods utilized a pre-trained language model (Wang et al., 2017b), or the sentence embedding from a machine translation model (Wang et al., 2017a), to score the out-domain data used for jointly training with the in-domain data. Data selection methods (Ruder and Plank, 2017; Ruder, Ghaffari, and Breslin, 2017) optimized an instance weighting function of many specific domain similarity features to select relevant training instances that are useful for transfer learning. The idea could be traced back to curriculum learning (Bengio et al., 2009; Jiang et al., 2015), which required a manually pre-defined ranking to weigh training instances for speeding up learning.

In this paper, we investigate the idea of instance weighting used in the DNN-based fine-tuning methods. Different from the previous instance weighting methods, we assign different weights to the target data only, and the fine-tuning method is trained on the target data without repeatedly accessing or processing any source data.

Instance Weighting based Fine-tuning

In this section, we first formulate the problem of the DNN-based fine-tuning method. Then, the framework of the proposed Instance Weighting based Fine-tuning (IW-Fit) method is presented, following by the designed weighting metrics used in IW-Fit explained in detail. Finally, we present two mixing variants of the two weighting metrics.

Problem Formulation

We are given a static source task $\mathcal{T}_s$ and a target task $\mathcal{T}_t$ with $\mathcal{T}_s \neq \mathcal{T}_t$. Our goal is to improve the classification accuracy in $\mathcal{T}_t$. Specifically, we have a source dataset $\mathcal{D}_s$, and a target dataset $\mathcal{D}_t = \{(x, y)\}$. Here, $x$ denotes the input of an instance and $y \in \{0, 1\}^m$ is the corresponding one-hot label vector over the $m$ classes. Let $g_\theta(\cdot)$ denote the discriminative function of a DNN parameterized by $\theta$. In text classification, the loss function, $f$, is generally set as the cross-entropy loss over $m$ classes:

$$f(y, g_\theta(x)) = -y^T \log p(y|x),$$

where $p(y|x)$ is the prediction probability vector output by the softmax layer of the DNN. Given an observed dataset $\mathcal{D}$,
we usually train the DNN with the following objective:

$$\theta^* = \arg \min_{\theta} \frac{1}{|D|} \sum_{(x,y) \in D} f(y, g_\theta(x)). \quad (2)$$

In our problem setting, we aim at obtaining the optimal parameters $\theta^*$ that can achieve a high classification accuracy on the target domain $D_t$. In fine-tuning methods, we first train a model on the source domain $D_s$. Next, we can apply existing methods (Howard and Ruder, 2018; Chung, Lee, and Glass, 2018; Felbo et al., 2017) to select a set of all pre-trained model parameters as the initialization for the target model. Then, the target model starts to update itself using the target dataset $D_t$ until it is converged. By using standard optimization algorithms such as SGD (Sutskever et al., 2013) or Adam (Kingma and Ba, 2014), we update the model parameters by treating all target instances with equal importance.

### Framework and Two Weighting Metrics

Our proposed IW-Fit method is presented in Algorithm 1. On the fine-tuning stage, we assign a weight $w_i$ to each instance in one mini-batch according to metrics introduced later, and update the network by a weighted sum of their individual gradients as:

$$\theta \leftarrow \theta - \alpha \sum_{i=1}^{k} w_i \cdot \nabla \log \theta f(y_i, g_\theta(x_i)), \quad (3)$$

where $\alpha$ is the learning rate, and $k$ is the mini-batch size.

Next, we will introduce our two metrics to calculate the instance weight $w_i$ of each instance $x_i$ from the source domain. The weighting metrics can be calculated as follows:

**Prediction loss:**

The prediction loss in (1) can be used as the first metric to assign the instance weight:

$$w = \frac{1}{\tau}(-y^T \log p(y|x) + \epsilon), \quad (4)$$

where $\epsilon$ is a smoothness constant for preventing the weight of an instance with a small loss being zero, and $\tau$ is a normalization constant making the average of weights from a mini-batch equal to 1. When $\epsilon$ gets larger, all $w_i$’s will be close to 1, and this metric reduces to uniform weighting.

**Variance of historical prediction losses:**

At epoch $t$ of our algorithm, assume that $h^{t-1} = [f^1, \ldots, f^{t-1}]$ is the vector containing the historical terms of the prediction loss $f = -y^T \log p(y|x)$. Our second metric can be computed based on the variance of prediction losses:

$$w = \frac{1}{\tau} (\text{std}(h^{t-1}) + \epsilon), \quad (5)$$

where $\text{std}(h^{t-1})$ is the estimated standard derivation plus its confidence interval in $h^{t-1}$:

$$\text{std}(h^{t-1}) = \sqrt{\epsilon(h^{t-1}) + \frac{c^2(h^{t-1})}{|h^{t-1}| - 1}},$$

where $c^2(h^{t-1})$ is the estimated variance of prediction losses, and $|h^{t-1}|$ is the number of stored prediction losses.

### Mixing Variants of Weighting Metrics

As stated in the above framework, the prediction loss is more effective at the early fine-tuning epochs. On the
contrary, the variance of historical prediction losses should be used at some latter epochs. Thus, we further present two mixing variants to combine these two weighting metrics.

Hard-mixing
We adopt the prediction loss at the first $\eta$ burn-in epochs and then switch to use the variance of prediction losses as:

$$w = \frac{1}{\eta} \left[ \mathbf{1}_{t \leq \eta} \cdot (\mathbf{y}^T \log \mathbf{p}(\mathbf{y}|\mathbf{x})) + \mathbf{1}_{t > \eta} \cdot \text{std}(h^{t-1}) + \epsilon \right], \quad (6)$$

where the indicator function $\mathbf{1}_A$ equals 1 when $A$ is true and 0 otherwise.

Soft-mixing
In the hard-mixing variant, the hyperparameter $\eta$ needs to be carefully adjusted, and the two metrics are mutually exclusive at each fine-tuning epoch. To explore a potentially more flexible combination of the two metrics, we propose to use a soft-mixing variant as:

$$w = \frac{1}{\eta} \left[ \beta \cdot (\mathbf{y}^T \log \mathbf{p}(\mathbf{y}|\mathbf{x})) + (1 - \beta) \cdot \text{std}(h^{t-1}) + \epsilon \right], \quad (7)$$

where $\beta$ is a balancing ratio that decreases linearly from 1 to 0 at fine-tuning epochs. Experimental results show that IW-Fit with the soft-mixing weighting metric generally performs the best among all compared fine-tuned methods.

Convergence analysis
Following Reddi et al. (2016), we give a convergence analysis in Theorem 1 to show that IW-Fit stabilizes and converges to a stationary point, and the convergence rate is $O(1/\sqrt{T})$. The proof can be found in Appendix A.

Theorem 1 Suppose the loss function of IW-Fit in (1) $f \in \mathcal{F}_n$, where $\mathcal{F}_n$ is the class of finite-sum Lipschitz smooth functions, has $\sigma$-bounded gradients, and the instance weight $w$ is clipped to be bounded by $[w, \overline{w}]$. Let $\alpha_t = \alpha = c/\sqrt{T}$ where $c = \sqrt{2(\epsilon f(0) - f(\theta^*))}/L\overline{w}^{2}$, and $\theta^*$ is an optimal solution. Then, the iterates of IW-Fit satisfy:

$$\min_{0 \leq t \leq T-1} \mathbb{E}[\| \nabla f(\theta^t) \|^2] \leq \sqrt{\frac{2(f(0) - f(\theta^*))L\overline{w}}{T\bar{w}}} \sigma.$$  

Experiments
In this section, we first introduce the two transfer learning tasks conducted in our experiments, following by compared baselines and training configurations. Next, the empirical results of IW-Fit with different configurations are reported. We want to empirically answer the following questions:

1. Can IW-Fit improve the classification accuracy on the target domain?
2. Why can IW-Fit boost the fine-tuning performance on the target domain?
3. How robust is IW-Fit?

We present our first set of experimental results to answer the first question. Testing accuracies of all methods show the advantage of IW-Fit over the baselines. Further, we divide the target instances into two types according to their differences from the source domain and observe that instances differing more from the source domain can benefit more. Finally, we demonstrate the robustness of IW-Fit regarding various aspects in our last experiment section.

Tasks
Our classification task is to use the review content to predict the number of stars ranging from 1 to 5 that the user has given (Johnson and Zhang, 2017; Tang, Qin, and Liu, 2015; Yang et al., 2016). To simulate the transfer learning setting and verify the effectiveness of IW-Fit, we conduct two tasks:

- **Amazon to Yelp**: We conduct this task for cross-domain transfer. The source dataset from 2015 Amazon reviews (Zhang, Zhao, and LeCun, 2015; Conneau et al., 2016) is mainly about products of online shopping. The target dataset of Yelp reviews is obtained from the latest 2018 Yelp Dataset Challenge, which is mainly about the daily life such as restaurants or home services.

- **Yelp-2015 to Yelp-2016 and Yelp-2015 to Yelp-2017**: The topics on the web change frequently and the reviews can be easily outdated. From the results shown in later sections, transfer learning is beneficial for this cross-time transfer task. From Yelp, we select out reviews in the year 2015/2016/2017 as the subsets Yelp-2015/2016/2017. We use Yelp-2015 as the source dataset, and Yelp-2016/2017 as the target datasets, respectively. No overlapped data exist among Yelp-2015, 2016 and 2017.

Dataset construction: In each set of experiments, we randomly select 100,000 instances from the source domain to construct the source dataset. Correspondingly, 10,000 training instances and 1,000 testing instances are randomly sampled from each target domain to construct the target dataset. The number of instances in each category is the same for all datasets on both the source and the target domains. We also try various data sizes for both domains in our last experiment section. To demonstrate the performance stability of the compared methods, we randomly sample three sets of source/target datasets for each task and run three sets of experiments for each of our experimental settings.

Baselines
Since we aim at improving the transfer learning performance on the fine-tuning stage, our focus is to compare with the fine-tuning baseline in this paper. Additionally, several other baselines are compared in our experiment:

- **SrcOnly**: It trains the model only on the source data, i.e., $\mathcal{D} = \mathcal{D}_s$. The obtained model is usually not generalized well on the target domain due to different data distributions between $\mathcal{D}_s$ and $\mathcal{D}_t$.

- **TgtOnly**: The model is trained on the target data with $\mathcal{D} = \mathcal{D}_t$. However, we often have limited training data on the target domain.

- **All**: Both datasets are adopted to train the model, i.e., $\mathcal{D} = \{\mathcal{D}_s, \mathcal{D}_t\}$. This method also suffers from the data distribution gap between $\{\mathcal{D}_s, \mathcal{D}_t\}$ and $\mathcal{D}_t$. 
Training Configurations

Since we are interested in validating the usefulness of IW-Fit on the fine-tuning stage, we use the same DNN structures, hyperparameters, and datasets for pre-training and fine-tuning for all compared methods for a fair comparison.

We adopt the recurrent neural network with long short-term memory (LSTM) as the text encoder due to its superior performance in various text classification problems (Liu et al., 2016; Liu, Qiu, and Huang, 2017; Lin et al., 2017; Yang, Salakhutdinov, and Cohen, 2017). The encoder is followed by a fully-connected (FC) layer with ReLU activation and a Softmax layer that outputs the final prediction probabilities. The hyperparameters are set as: \textit{vocabulary size} = 30,000, \textit{embedding size} = 100, \textit{LSTM hidden size} = 200, \textit{FC layer size} = 200, \textit{dropout rate} = 0.2, \textit{L2 regularisation} = $1e^{-4}$, \textit{batch size} = 32, \textit{fine-tuning epochs} = 50, \textit{burn-in epochs} = 10, \textit{learning rate} = $1e^{-3}$. We also change to use a CNN-based text encoder and vary different parameter settings in section Robustness Study.

Due to the randomness of DNNs, we repeat five runs over each training dataset and report the mean and the standard error regarding the testing accuracy on the target domain.

Results of All Compared Methods

Due to the limit of space, we show one set of results for each task. The other two sets of results for all the tasks are provided in the Appendices.

Table 1 reports the classification accuracy results on the task from Amazon to Yelp. \textsuperscript{2} Table 2 reports the results on the task from Yelp-2015 to Yelp-2016/Yelp-2017. It can be observed that all fine-tuned methods perform better than the baselines. IW-Fit achieves superior performance compared to the fine-tuned method with uniform weighting. Specifically, IW-Fit with the soft-mixing weighting metric performs the best with an increased accuracy of 1.08% on the task Amazon to Yelp, 1.12% on task Yelp-2015 to Yelp-2016, and 0.82% on the task Yelp-2015 to Yelp-2017. We attribute this advantage to the flexible combination of the two metrics, which enables the algorithm to emphasize domain-specific instances while balancing the model to preserve the shared knowledge. In the other two sets of experiments in Appendix B, the accuracies increase by 0.80% and 0.65% on the task

\textsuperscript{2}For all reported results, bold numbers are significantly better than the second best results with \textit{p-values} smaller than 0.01.

<table>
<thead>
<tr>
<th>Method</th>
<th>Amazon to Yelp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baselines</td>
<td></td>
</tr>
<tr>
<td>TgtOnly</td>
<td>50.22 ± 0.11</td>
</tr>
<tr>
<td>SrcOnly</td>
<td>46.30 ± 0.06</td>
</tr>
<tr>
<td>All</td>
<td>50.94 ± 0.08</td>
</tr>
<tr>
<td>Fine-tuned</td>
<td></td>
</tr>
<tr>
<td>Uniform</td>
<td>52.46 ± 0.06</td>
</tr>
<tr>
<td>Gradient</td>
<td>52.80 ± 0.09</td>
</tr>
<tr>
<td>Variance</td>
<td>53.20 ± 0.07</td>
</tr>
<tr>
<td>Hard-mixing</td>
<td>53.44 ± 0.08</td>
</tr>
<tr>
<td>Soft-mixing</td>
<td>53.54 ± 0.07</td>
</tr>
</tbody>
</table>

Table 1: Testing accuracies from Amazon to Yelp.

<table>
<thead>
<tr>
<th>Method</th>
<th>Yelp-2016</th>
<th>Yelp-2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baselines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TgtOnly</td>
<td>50.23 ± 0.11</td>
<td></td>
</tr>
<tr>
<td>SrcOnly</td>
<td>56.43 ± 0.10</td>
<td></td>
</tr>
<tr>
<td>All</td>
<td>56.97 ± 0.11</td>
<td></td>
</tr>
<tr>
<td>Fine-tuned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniform</td>
<td>57.18 ± 0.09</td>
<td></td>
</tr>
<tr>
<td>Gradient</td>
<td>57.82 ± 0.10</td>
<td></td>
</tr>
<tr>
<td>Variance</td>
<td>57.60 ± 0.08</td>
<td></td>
</tr>
<tr>
<td>Hard-mixing</td>
<td>58.13 ± 0.09</td>
<td></td>
</tr>
<tr>
<td>Soft-mixing</td>
<td>58.30 ± 0.09</td>
<td></td>
</tr>
</tbody>
</table>


Figure 2: Testing accuracies of the fine-tuned methods with uniform and soft-mixing weighting: (a) at the fine-tuning epochs; (b) on two types of target instances.

Amazon to Yelp. In appendix C, the accuracies increase by 0.76% and 0.64% on the task Yelp-2015 to Yelp-2016, and 0.70% and 1.22% on the task Yelp-2015 to Yelp-2017. As can be seen, IW-Fit can consistently improve the classification accuracy on the target domain.

In Figure 2-(a), we further plot the testing accuracies of IW-Fit with soft-mixing weighting metric and the fine-tuned method with uniform weighting along the fine-tuning epochs on the task Amazon to Yelp. We can observe that with the use of IW-Fit, the testing accuracy on the target domain consistently increases during the entire fine-tuning process.

Results and Analysis of IW-Fit

To understand why IW-Fit can help fine-tuning, we first analyze what types of target instances can benefit from our method. We extract the top 1000 frequent tokens in Amazon and Yelp domains, respectively. The two sets of tokens have an overlap of 575 tokens. Then, we divide the 1000 testing instances in Yelp domain into two types: 1) \textit{tgt-only} instances that do not contain any frequent Amazon tokens; 2) \textit{overlap} instances that contain tokens which appear in both domains. \textit{Tgt-only} instances should be considered more different from the source domain than \textit{overlap} instances.

Figure 2-(b) shows the testing accuracies on the two types of target instances. The test accuracy on the 478 \textit{overlap} instances increases slightly by 0.10%, which shows that IW-Fit well preserves the shared knowledge between the source and the target domains. The performance on the other 522 \textit{tgt-only} instances is improved obviously with an increased accuracy of 1.98%. It can be observed that a better boost can
be achieved on instances that differ more from the source domain. This indicates that the advantage of IW-Fit is due to capturing more specific knowledge of the target domain while well preserving the shared knowledge.

Table 3 gives some training instances with different weights on the task Amazon to Yelp using IW-Fit. The weights are computed according to the prediction loss at Epoch 1 and according to the variance of historical prediction losses at Epoch 20. We can observe that by both metrics, instances with high weights tend to contain more domain-specific words, which appear more frequently on the target than the source domain. Correspondingly, instances with low weights are likely to contain domain-independent words that commonly appear on both domains. This result also illustrates the effectiveness of the proposed metrics for assigning instance weights used in IW-Fit.

**Robustness Study**

In this section, we conduct various experiments to demonstrate the robustness of IW-Fit regarding five aspects including: 1) using CNN as the text encoder; 2) varying different parameters for fine-tuning; 3) varying the target data size for fine-tuning; 4) varying the source data size for pre-training; 5) varying the training mini-batch size. Here we show the results of Uniform and Soft-mixing for better readability. Full results of all compared methods can be found in the Appendices.

**Using CNN as the text encoder**

Since our method and the proposed metrics do not depend on the specific design of the DNN-based classifiers, we investigate the performance of IW-Fit under different DNN structures by changing to use Text-CNN (Kim, 2014) as our text encoder. Table 4 shows the testing accuracies of Soft-mixing and Uniform. It can be observed that the proposed model-agnostic method still achieves statistically significant improvement on the fine-tuning stage under the CNN architecture.

**Varying the sets of model parameters for fine-tuning**

In Problem Formulation, we have discussed that previous fine-tuning methods study to transfer different sets of parameters for fine-tuning. For the LSTM-based text classifier used in our experiments, the model parameters include word embeddings, parameters in LSTM and those in FC layers. To test the robustness of our method regarding the transferred model parameters, we keep some model parameters fixed and only fine-tune the other parameters.

Tables 5 and 8 report the results when the encoder including word embedding and LSTM is fixed (i.e., only fine-tune parameters in FC), when only the word embedding is fixed (i.e., fine-tune parameters in LSTM and FC) and when no parameters are fixed (i.e., fine-tune all parameters). We can see that in these three fine-tuning settings, Soft-Mixing always performs better than Uniform. Additionally, both methods yield the best performance when fine-tuning all the model parameters. We conjecture that this is due to the appropriate model structure for both the source and the target datasets so that fine-tuning all the model parameters results in the best performance.

**Varying the fine-tuning data size.**

Moreover, the performance of the fine-tuned methods should be verified on datasets with different sizes (Chung, Lee, and Glass, 2018; Howard and Ruder, 2018). This experiment is conducted to study the relationship between the amount of target data and IW-Fit. We vary the training data size of the target dataset in the fine-tuning process. The results are shown in Tables 6 and 9. As expected, the performance of Uniform and Soft-mixing is improved when the number of training data used for fine-tuning increases. Soft-mixing achieves the best performance for most settings, and it obtains a consistent improvement in testing accuracy by 0.38% to 1.12% compared to Uniform.

**Varying the pre-training data size.**

We also vary the size of the source dataset used for pre-training to analyze the relationship between the amount of source data and IW-Fit. The results are shown in Tables 7 and 10. Usually, the performance of the fine-tuned methods can be boosted as the source data size increases. It can be inferred that model pre-trained on a larger source domain can provide a better initialization of parameters for the target domain. Consistent with the above observations, Soft-mixing can stably improve the classification accuracy by 0.50% ~ 1.30% on the target domain.

**Varying the training mini-batch size**

Generally, small batch sizes tend to produce convergence in fewer epochs while large ones offer more data-parallelism and better computational efficiency (Keskar et al., 2016). To investigate the robustness of IW-Fit regarding the training batch size, we vary the batch size from 16 to 64 on the task Amazon to Yelp. Table 11 reports the testing accuracies of Soft-mixing and Uniform. It shows that the proposed method achieves better classification accuracy regarding different mini-batch sizes during fine-tuning.

**Conclusion and Future Work**

In this paper, we proposed the IW-Fit method, which incorporates the idea of instance weighting into the fine-tuning
stage to improve the text classification accuracy on the target domain. The proposed method enabled the algorithm to emphasize the specific knowledge on the target domain, while well preserving the shared knowledge between the source and the target domains. Rather than relying on specific rules or statistical approaches, the weighting metrics are model-agnostic and easy to implement for general DNN-based structures. We performed extensive experiments on two kinds of transfer learning tasks to verify the advantage of IW-Fit. Experimental results showed the effectiveness of the weighting metrics, and that IW-Fit consistently boosted the classification accuracy on the fine-tuning stage. In the future, we will try to extend our idea for applications on more NLP tasks such as sequence labeling and text generation.

### References


Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J.