Towards More Accurate Uncertainty Estimation In Text Classification

Jianfeng He[†], Xuchao Zhang[†], Shuo Lei[†*], Zhiqian Chen[+], Fanglan Chen[†], Abdulaziz Alhamadani[†], Bei Xiao[‡], Chang-Tien Lu[†]
[†]Discovery Analytics Center, Virginia Tech, Falls Church, VA, USA
[+] Computer Science and Engineering, Mississippi State University, Starkville, Mississippi, USA
[‡]Department of Computer Science, American University, Washington, DC, USA
[†]{jianfenghe, xuczhang, slei, fanglanc, hamdani, ctlu}@vt.edu,
[+]zchen@cse.msstate.edu, [‡]bei.xiao@american.edu

Abstract

The uncertainty measurement of classified results is especially important in areas requiring limited human resources for higher accuracy. For instance, data-driven algorithms diagnosing diseases need accurate uncertainty score to decide whether additional but limited quantity of experts are needed for rectification. However, few uncertainty models focus on improving the performance of text classification where human resources are involved. To achieve this, we aim at generating accurate uncertainty score by improving the confidence of winning scores. Thus, a model called MSD, which includes three independent components as "mix-up", "self-ensembling", "distinctiveness score", is proposed to improve the accuracy of uncertainty score by reducing the effect of overconfidence of winning score and considering the impact of different categories of uncertainty simultaneously. MSD can be applied with different Deep Neural Networks. Extensive experiments with ablation setting are conducted on four real-world datasets, on which, competitive results are obtained.

1 Introduction

Text classification is a popular topic with broad applications. A successful and common model for text classification is Deep Neural Network (DNN). However, some real-world applications expect results with higher accuracy than the ones achieved by state-of-the-art algorithms. Hence, the most uncertain predictions need domain experts for further decisions (Zhang et al., 2019). To efficiently leverage the limited human resources, it is essential to calculate *uncertainty score* of the model prediction, which quantifies how unconfident the model prediction is. This paper aims at generating more accurate uncertainty score through DNNs in the text classification with human involvement in the testing process. This is different from active learning, which involves experts in the training process.

Though various metrics of the uncertainty score have been studied (Dong et al., 2018; Wang et al., 2019; Shen et al., 2019; Xiao and Wang, 2019; Kumar et al., 2019), the existing metrics directly or indirectly depend on *winning score*, which is the maximum probability in a semantic vector (softmax vector from the last layer of a DNN model) (Thulasidasan et al., 2019). Therefore, improving *Confidence of Winning Score* (CWS), which describes how confident the winning score matches the sample uncertainty and represents the accuracy of the winning score, is helpful to improve the accuracy of uncertainty score. To show the effect of improving CWS, this paper considers a basic way to measure uncertainty score, which is the reciprocal of winning score (Snoek et al., 2019). However, we face two challenges in improving CWS: (1) how to reduce effect of *overconfidence of winning score*[1] to boost negative correlation between the winning score and sample uncertainty, (2) how to generate winning scores by considering comprehensive categories of uncertainty in one model rather than only one or two categories of uncertainty at a time.

The overconfidence of winning scores has been neglected by vast previous works in Natural Language Processing. We identify the presence of *overconfidence* for the training samples: because the winning scores of training samples are all set as 1 by one-hot labels, each sample will have the same uncertainty score. Consequently, the training sample uncertainty will be the same. Together, the winning scores and sample uncertainty are the same for various training samples. Hence, the negative correlation between the winning scores and sample uncertainty cannot be guaranteed, which is a negative effect of the overconfidence. The effect will affect calculating the uncertainty scores. Concretely, in the testing process, we apply different

*Corresponding author.

[1]Noted as overconfidence in the paper.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 8362–8372,
November 16–20, 2020.

predicted winning scores to match different sample uncertainty based on a latent assumption that the predicted winning score is negatively correlated to the sample uncertainty. However, the assumption is biased because of the negative effect of the overconfidence. To mitigate the impact of overconfidence, we generate new training sample representations with different winning scores, which are also negatively correlated to the sample uncertainty.

Additionally, the process generating the winning score should consider the impact of different categories of uncertainty simultaneously, while vast of the previous works (Shen et al., 2019; Wang et al., 2019; Xiao and Wang, 2019; Zhang et al., 2019) only consider one or two categories of uncertainty at a time[2]. We assume the partial consideration will decrease the CWS, and so will the accuracy of uncertainty score. We verify this assumption by our ablation experiments. The uncertainty of a model prediction is derived from two parts: *data uncertainty* and *model uncertainty*. The data uncertainty (Rohekar et al., 2019) is further divided into two categories: *epistemic uncertainty* comes from lack of knowledge, such as only few training data or out-of-distribution testing data; *aleatoric uncertainty* is caused by noisy data in the generation of both training data and testing data. The model uncertainty (Liu et al., 2019) also has two categories: *parametric uncertainty* comes from different possibilities of parameter values in estimating model parameters under the current model structure and training data; *structural uncertainty* is uncertainty about whether the current model design (e.g., layers, loss functions) is reasonable or sufficient for the current task and training data. Since the solution of structural uncertainty requires extremely high computations, such as Neural Architecture Search (NAS) (Zoph and Le, 2016; Xie et al., 2019), we only reduce or scale the other three categories of uncertainty simultaneously to improve the CWS.

To address the above two challenges, we propose a model called MSD, which is named as the initials of its components ("Mix-up", "Self-ensembling", and "Distinctiveness score") aiming at handling overconfidence and various uncertainty with flexibility. The flexibility means that MSD is effective on different DNN models (Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Transformer (Vaswani et al., 2017)), and each component in MSD is independent, which can be arbitrarily assembled. The main contributions of our work can be summarized as follows,

Reducing impact of overconfidence. To reduce the impact of overconfidence in calculating uncertainty scores, we apply mix-up to generate new sample representations boosting the negative correlation between the winning scores and sample uncertainty.

Considering various uncertainty comprehensively. We propose MSD with three components to handle the epistemic uncertainty, aleatoric uncertainty, and parametric uncertainty simultaneously, so that the uncertainty score is more accurate.

Designing flexibility of MSD. MSD can be applied with different DNNs (CNN, RNN, and Transformer). Each component in MSD can be assembled with other components arbitrarily due to their independence.

Implementing extensive experiments. We evaluated MSD by the improvement of text classification accuracy in simulating human involvement. The experiments of MSD with ablation setting on four datasets achieved competitive results, which demonstrated that MSD generates more accurate uncertainty scores.

2 Related work

Methods mitigating uncertainty: One main solution to mitigate uncertainty is Bayesian Neural Network (BNN) (Klein et al., 2017), which is a neural network with a prior distribution on its weights. Based on BNN, variational Bayesian inference is proposed, which finds an approximated distribution of parameters for the true distribution of parameters by Kullback-Leibler (KL) divergence (Xiao and Wang, 2019; Wen et al., 2018; Louizos and Welling, 2017; Malinin and Gales, 2019). Further, as an approximation of variational Bayesian inference, Monte Carlo dropout is proposed (Gal and Ghahramani, 2016; Kendall and Gal, 2017). This is implemented by training a model with dropout before every layer, and also performing the dropout in the testing process to derive results from different sampled parameter sets. Plus, an approximation of Monte Carlo dropout is tried by only adding dropout before the last layer (Riquelme et al., 2018; Snoek et al., 2019). Besides BNN, noise injection is the other main technique to mitigate uncertainty. It has two categories: parameter noise injection adds noise perturbation in network weights (Plappert et al., 2017); data noise injection

[2]Please refer to our appendix for detailed comparisons

directly inputs noise perturbation into data (Dong et al., 2018).

Metrics scaling uncertainty: Many metrics about uncertainty score are proposed based on the softmax vectors. As an important element in the sofmax vectors, winning score is proposed in (Hendrycks and Gimpel, 2016). Furthermore, temperature scaling (Guo et al., 2017) is proposed to get the calibrated probability by adding a scalar parameter to each class in calculating softmax vector. Applying winning score as prediction confidence is proposed in (Niculescu-Mizil and Caruana, 2005; Guo et al., 2017). This confidence is further applied in Expected Calibration Error (Naeini et al., 2015), which is the absolute value of the difference between the accuracy and confidence of results. Besides, Overconfidence Error is proposed by applying winning score as confidence and penalizing samples with confidence values greater than accuracy values (Thulasidasan et al., 2019). In addition, four metrics for result confidence are proposed in (Wang et al., 2019) by combining expectation and variance of predictions from different sampled parameter sets. In addition, cross-entropy is applied to calculate uncertainty score by dropout sampling and bin counting in (Zhang et al., 2019), which also considers text classification with human involvement. Different from previous works, we improve the accuracy of uncertainty score by reducing the effect of overconfidence and considering three categories of uncertainty simultaneously.

3 Model

3.1 Basic Text Classification Model

In the traditional text classification model (Zhang et al., 2019; Shen et al., 2018), given an original text, we apply preprocessing (tokenization, lemmatization, etc.) to get its tokens in discrete numbers. Then, a pre-trained token embedding, such as word2vec (Mikolov et al., 2013) or Glove (Pennington et al., 2014) is applied as a projector. After that, a sequence of dense vectors for i-th text $\mathbf{Z}_i = [\mathbf{z}_{i1}, \mathbf{z}_{i2}, ..., \mathbf{z}_{in}]$ is derived by the embedding, where $\mathbf{z}_{ij}$ is the embedding of j-th word. The $\mathbf{Z}_i$ is fed to a sequence model f, such as CNN or RNN. Finally, we get i-th text representation $\mathbf{x}_i$ from the penultimate layer of f with dropout, and predicted semantic vector $\mathbf{y}_i = [y_{i1}, y_{i2}, ..., y_{ic}]$ from the last layer of f, where c is the number of classes and y_{ij} is the probability that i-th text belongs to j-th class. Finally, the f is trained by cross-entropy loss between the predicted semantic vector $\mathbf{y}_i$ and one-hot label $\widehat{\mathbf{y}}_i = [\widehat{y}_{i1}, \widehat{y}_{i2}, ..., \widehat{y}_{ic}]$ as follows,

$$L_{CE} = \sum_{j=1}^{c} \widehat{y}_{ij} log(y_{ij}). \quad (1)$$

In the testing process, uncertainty score U is formulated as follows,

$$U = \frac{1}{\max(\mathbf{y}_i^*)} \quad (2)$$

where $\mathbf{y}_i^*$ is semantic vector of i-th testing sample and $\max(\mathbf{y}_i^*)$ is the winning score of $\mathbf{y}_i^*$. Then, U conveys the uncertainty of model result.

3.2 Overview Of MSD

Fig. 1 illustrates the training process of our model. In the first row, after prepossessing training text, we calculate the text representations, which is output of the penultimate layer with dropout. Then, we mix these representations in the batch-level. These mix-up-generated representations are fed into a fully connected (FC) layer for final semantic vectors. In the second row, we apply another model, which implements self-ensembling, with independent optimized parameters but the same structure as the one in the first row.

In the testing process, besides computing the reciprocals of winning scores with dropout mechanism, distinctiveness scores is also calculated by the Mahalanobis distance between the testing samples and distributions of training samples. Finally, the uncertainty score is calculated by adding the reciprocal of winning score and distinctiveness score.

3.3 MSD Training: Mix-up

Since the overconfidence is caused by the training samples with same winning scores due to the one-hot labels, and adding noise perturbation in the training process is a way to mitigate aleatoric uncertainty, we apply mix-up (Zhang et al., 2017; Thulasidasan et al., 2019) to jointly address the two issues. Mix-up generates new sample representations with various winning scores.

Concretely, we have i-th sample representation $\mathbf{x}_i$ from the penultimate layer of f with dropout. In a batch, we randomly mix i-th and j-th samples' representations ($\mathbf{x}_i$ and $\mathbf{x}_j$) and one-hot labels ($\widehat{\mathbf{y}}_i$ and $\widehat{\mathbf{y}}_j$) to get a mix-up sample representation $\widetilde{\mathbf{x}}$ and ground truth label $\widetilde{\mathbf{y}}$. We formulate mix-up as,

$$\widetilde{\mathbf{x}} = \alpha \mathbf{x}_i + (1 - \alpha)\mathbf{x}_j \quad (3)$$

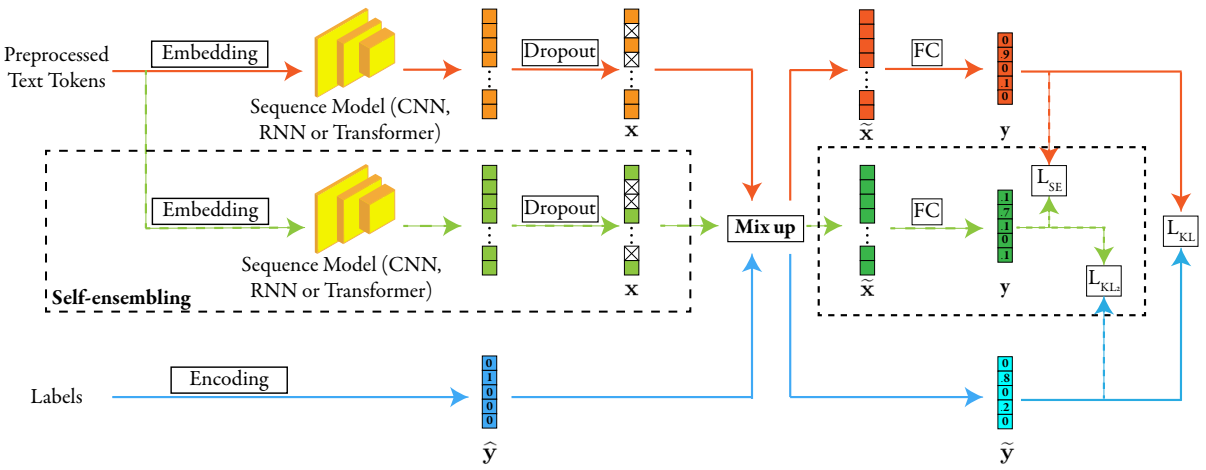

Figure 1: Diagram of training process of MSD. Orange arrows, green arrows, and blue arrows represent data flow of the first (default) model, second model, and labels respectively. Since self-ensembling is optional, it is illustrated as dotted lines. The distinctiveness score is not shown in the diagram since it is applied in the testing process. The numbers shown in $\mathbf{y}$, $\widehat{\mathbf{y}}$ and $\widetilde{\mathbf{y}}$ are probabilities of the semantic vectors.

$$\widetilde{\mathbf{y}} = \alpha\widehat{\mathbf{y}}_i + (1-\alpha)\widehat{\mathbf{y}}_j \tag{4}$$

where α is a random number ranging from Ω to 1.00. The Ω is set above 0.5, so the i-th sample's semantics will be the main semantics of $\widetilde{\mathbf{x}}$, which is regarded as class of $\widetilde{\mathbf{x}}$ in MSD. Since the *difference* among winning scores and *negative correlation* between the winning scores and sample uncertainty are essential to reduce the impact of overconfidence, we analyze the two factors below.

Difference: Since $1 \geq \alpha \geq \Omega > 0.5$, $\widetilde{\mathbf{y}}$ has a winning score as α if $\mathbf{x}_i$ and $\mathbf{x}_j$ have different classes, or as 1 if two samples have the same class. Then, firstly, α or 1 is randomly chosen; secondly, the specific value of α is randomly sampled. Thus, different values of winning scores of training samples are achieved by the mix-up.

Negative correlation: Since $\widetilde{\mathbf{x}}$ includes i-th sample's representation $\mathbf{x}_i$ with ratio $\alpha > 0.5$, $\mathbf{x}_j$ can be regarded as noise of $\widetilde{\mathbf{x}}$. In one scenario, when $\mathbf{x}_i$ and $\mathbf{x}_j$ have different classes, $\mathbf{x}_j$ has obvious effect from noise on $\mathbf{x}_i$ due to different distributions in various semantics. In this case, when α is greater, $\widetilde{\mathbf{x}}$ has less noise from different semantic distributions. Then, $\widetilde{\mathbf{x}}$ is less adulterated and has higher confidence belonging to the class of $\mathbf{x}_i$. Since now winning score equals to α and the transitivity in math: the higher winning score is, the less adulterated $\widetilde{\mathbf{x}}$ is, which means $\widetilde{\mathbf{x}}$ has less uncertainty. In another scenario, where $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to same class, we assume that $\mathbf{x}_j$ has no effect from noise on $\mathbf{x}_i$, because $\mathbf{x}_j$ belongs to same distributions as $\mathbf{x}_i$ due to same semantics. Thus, the $\widetilde{\mathbf{x}}$ is the least uncertain in its class, and its winning score is the highest as 1. Hence, the negative correlation is boosted by mix-up.

After the mix-up, we feed $\widetilde{\mathbf{x}}$ rather than $\mathbf{x}$ to FC layer for its predicted semantic vector $\mathbf{y}$. However, we do not use cross-entropy loss (Eq. 1) in MSD, because it will learn the winning scores close to 1 due to no limit on the upper bound, which cannot ensure the negative correlation. Instead, we use KL divergence loss as one of our loss functions,

$$L_{KL} = \sum_{j=1}^{c} \widetilde{y}_{ij} log(\frac{\widetilde{y}_{ij}}{y_{ij}}) \tag{5}$$

because $\widetilde{\mathbf{y}}_i$ approximates to provide both upper and lower bound limitations by its non-zero element(s). The $\widetilde{y}_{ij}$ is a random value in each batch and each epoch. The overconfidence is reduced by mix-up due to *difference* and *negative correlation*. Besides, $\mathbf{x}_j$ can be regarded as random noise perturbation. Therefore, the aleatoric uncertainty is mitigated.

3.4 MSD Training: Self-ensembling

The parametric uncertainty comes from different sets of weights achieving similar training losses. Although the dropout can mitigate parametric uncertainty, previous works ignore the effect of self-ensembling (Laine and Aila, 2016; Park et al., 2018), which can boost the model combination and further decrease parametric uncertainty. We assume the dropout reduces the parametric uncertainty by loss generated within a model, while the self-ensembling reduces it from loss generated between models. The loss generated between models can help stabilize the model weights, because it

can provide extra limitations besides the loss generated in a model, which reduce feasible weight sets. Plus, the designed component should aim at mitigating parametric uncertainty while have little impact on the model performance. Consider that the self-ensembling calculates the loss between the same models, which has more effect on model robustness and less impact on model performance, we apply the self-ensembling in addition to dropout to further mitigate the parametric uncertainty.

We construct another model with the same framework (e.g. layers, loss functions, dropout rate), and apply a self-ensemble loss L_{SE} to minimize the difference between two outputs from two models (the first model and the second model[3]) with the same framework and inputs,

$$L_{SE} = D[f_{\theta_1}(\widetilde{\mathbf{x}}, \phi_1), f_{\theta_2}(\widetilde{\mathbf{x}}, \phi_2)] \quad (6)$$

where θ_a is parameter set of a-th model, ϕ_a represents randomly sampled dropout neurons in neural network f, and $D[\mathbf{y_1}, \mathbf{y_2}]$ is a metric between two semantic vectors. D is Mean Square Error (MSE).

Although we already have the loss L_{SE}, we add KL divergence loss L_{KL_2} in the second model for the same setting. The L_{KL_2} is same as Eq. 5, while y_{ij} comes from the second model. We formulate MSD loss function L_{MSD} as follows,

$$L_{MSD} = L_{KL} + \lambda_1 L_{KL_2} + \lambda_2 L_{SE} \quad (7)$$

where λ_1 and λ_2 equal to 1 and a positive value respectively, when we apply self-ensembling, otherwise they both equal to 0. Together, the parametric uncertainty is further reduced.

3.5 MSD Testing: Distinctiveness Score

We also consider the epistemic uncertainty. Though out-of-distribution testing samples are known as the sources of epistemic uncertainty, they show that the epistemic uncertainty is the distinctiveness between the testing and training texts. However, it is not easy to consider the distinctiveness in the training process, because the training process is not aware of distributions of the testing samples. Therefore, we assume each class-level distribution of the training data can be modeled as a multivariable Gaussian distribution. We consider distance between a testing sample and each class-level Gaussian distribution as one part of the distinctiveness score. Motivated by (Lee et al., 2018), we apply Mahalanobis distance as follows,

$$m_{is} = (\mathbf{x}_i^* - \mu_s)^T \mathbf{\Sigma}^{-1} (\mathbf{x}_i^* - \mu_s) \quad (8)$$

where $\mathbf{x}_i^*$ is the representation of i-th testing sample in the first model without mix-up, and μ_s is the mean of representations of all training samples that belong to s-th class. $\mathbf{\Sigma}^{-1}$ is inverse of the covariance of all training samples. We do not apply the covariance in class-level to avoid singular matrices. After we obtain the Mahalanobis distance $\mathbf{m}_i = [m_{i1}, m_{i2}, ..., m_{ic}]$ of i-th testing sample to each class-conditional Gaussian distribution, we can also have a predicted class from this view, which is the class with the smallest distance in $\mathbf{m}_i$. In this way, we design penalty p as the other part in the distinctiveness score, which is not considered in (Lee et al., 2018), as below,

$$p_i = \begin{cases} 0 & r_m = r_y \\ \xi & r_m \neq r_y \end{cases} \quad (9)$$

where r_m is a classified result by $\mathbf{m}_i$ and r_y is the class with maximum probabilities in predicted semantic vector $\mathbf{y}_i^*$. ξ is a constant and set as 10 in our work. Our distinctiveness score d_i is,

$$d_i = log(\beta_1 \times p_i + \beta_2 \times min(\mathbf{m}_i)) \quad (10)$$

where log is a logarithm to the base 10; β_1 and β_2 are constants, both set as 1. Thus, the epistemic uncertainty is scaled in the uncertainty score to improve its accuracy. And the component improves CWS indirectly, because it remedies CWS for missing the epistemic uncertainty in the training process.

3.6 MSD Testing: Uncertainty Score

After we have trained our model by applying mix-up and self-ensembling, the winning scores will have higher confidence and accuracy due to reducing the overconfidence, aleatoric uncertainty, and parametric uncertainty. Regardless of whether we use self-ensembling or not, we only apply the first model to calculate the mean of predicted semantic vectors $\bar{\mathbf{y}}_i^*$ with dropout mechanism. Concretely, given a testing sample $\mathbf{x}_i^*$, we obtain k different predicted semantic vectors $\mathbf{y}_{i1}^*, \mathbf{y}_{i2}^*, ..., \mathbf{y}_{ik}^*$ by k times tryouts with the same dropout rate, from which, $\bar{\mathbf{y}}_i^*$ is the mean of k different $\mathbf{y}_i^*$. The maximum probability in $\bar{\mathbf{y}}_i^*$ is our winning score. Besides training for more confident winning scores, we also scale

[3]The first model is our default model, and the second model is only required when we apply self-ensembling. They are shown in the first row and second row respectively in Fig. 1.

distinctiveness score d_i to measure the impact of epistemic uncertainty. We calculate our final uncertainty score U as,

$$U = \gamma_1 \times \frac{1}{\max(\bar{\mathbf{y}}_i^*)} + \gamma_2 \times d_i \qquad (11)$$

where γ_1 and γ_2 are constants.

4 Experiments

Focusing on the text classification with human involvement, we evaluate the performance of MSD on four real-world datasets. Sec. 4.1 shows an overview of our experiment settings. Sec. 4.2 compares the performance between MSD and the state-of-the-art methods, and analyzes results of ablation experiments and parameter sensitivity analysis.

4.1 Experimental Setup

We apply Glove embedding (Pennington et al., 2014), which is pretrained with dimension of 200, as our word embedding by default. For CNN model, we train MSD by setting a sequence model as a 3-layer CNN by default, with batch size of 32, momentum of 0.9, initial learning rate as 0.001 by Adam (Kingma and Ba, 2014), kernel size of each layer as 3, 4, 5, respectively, as well as dropout rate of 0.3. For RNN model, Bidirectional Gated Recurrent Units (BiGRU) (Jabreel et al., 2018) is applied as an example of RNN model with two hidden layers. For Transformer, we apply XLnet (Yang et al., 2019) as an example[4].

4.1.1 Datasets

The four real-world-based datasets used in our experiments are as follow: (1) **20 Newsgroups** (20News) (Lang, 1995) includes 20 different news categories with 20,000 documents in it. (2). **Amazon Reviews** (Amazon) (McAuley and Leskovec, 2013) is a collection of reviews from Amazon from May 1996 to July 2013. For better comparison, we apply data from Sports and outdoors category, which is same as (Zhang et al., 2019). This dataset has 272,630 text samples with sentimental rating labels from 1 to 5. (3) **IMDb Reviews** (IMDb) has binary sentimental rating with 50,000 popular movie reviews. (4). **Yelp Reviews** (Yelp) (Zhang et al., 2015) is a collection with sentimental rating labels from 1 to 5. It has two parts: the first part has 130,000 samples for each rating; the second part has 10,000 samples for each rating.

[4]More details and the experimental results on RNN and Transformer are shown in the appendix.

For the first three datasets, we apply the same split setting as (Zhang et al., 2019), where for each dataset, 70% of samples form the training set, 10% of samples form the validation set, and the rest 20% form the testing set. For the Yelp dataset, we choose 9,000 samples randomly from the second part for each label as training set and the rest 1,000 samples for each label as the validation set, while all samples in the first part form the testing set.

4.1.2 Metrics

To evaluate the performance improvement of text classification with human involvement, which shows accuracy of uncertainty scores, we scale classification accuracy in different eliminated ratios. Concretely, for a testing set S with q samples and eliminated ratio r, we remove the most uncertain samples S_r from S based on uncertainty score ranking, where S_r has $r \times q$ samples. The more accurate uncertainty score we obtain, the more misclassified samples will be removed with the same r. Thus, if a model generates more accurate uncertainty scores, then the F1 scores for the rest testing samples will be higher with the same r. Because uncertainty score is more crucial for semantics with less training samples (e.g. "patient data samples" versus "the data for the healthy" in disease detection), we apply macro F1 score for the rest testing samples in the different eliminated ratios.

4.1.3 Baselines and Ablation Setting

We compare MSD with a state-of-the-art method, which achieves superior improvement of F1 scores in text classification with human involvement (Zhang et al., 2019). It proposes two methods: Dropout-Entropy (**DE**) is a dropout-entropy based model, and **DE+Metric** is a **DE** model along with metric learning. As for MSD, we divide MSD into three sub-models for ablation study: **MSD1** is a sub-model with only mix-up component; **MSD2-a** (abbreviate as **MSD2**) is one with two components, we apply mix-up and self-ensembling components by default; to show the flexibility of MSD, we design **MSD2-b**, which has two components as mix-up and distinctiveness score; and **MSD3** is one with all three components.

4.2 Experimental Results

4.2.1 Results of CNN model

Table 1, 2, 3, 4 report the F1 score improvement in the text classification with various eliminated ratios (10%, 20%, 30%, 40%) for CNN model. The

improved ratios of F1 scores compared with no uncertainty elimination (0% column), are illustrated after the F1 scores. The parameter setting of Ω, λ_2, γ_1, γ_2 are given after each MSD in order. Three datasets (20 Newsgroups, Amazon, Yelp) are compared in macro F1 scores, except IMDb, which applies weighted F1 score for better comparison with (Zhang et al., 2019). From the tables, we conclude as below.

1) **Better values of F1 scores:** MSDs (MSD1, MSD2, MSD3) improve F1 scores in values when certain portions of the most uncertain samples are eliminated. Especially for the Amazon dataset, the DE and DE+Metric both have negative growth when more uncertain samples are removed with eliminated ratios increased. This shows the accuracy of uncertainty score scaled in the testing is low for Amazon by DE and DE+Metric, while MSDs achieve significant increase on F1 when the most uncertain samples are eliminated, such as 26.64% increase in 40% elimination. In the 20News, MSDs achieve slightly lower F1 scores compared with DE+Metric, although slightly higher in F1 scores compared with DE. This is caused by obvious difference between texts with various semantics, so the uncertainty influence weakens and MSD is not very effective in the 20News.

2) **Better improved ratios of F1 scores:** If the uncertainty scores are more accurate, higher improvement in the ratios of F1 scores would also be achieved. In comparison to DE and DE+Metric, MSDs always achieve better improved ratios of F1. Thus, MSDs generate more accurate uncertainty score. Especially, though MSD2 has lower F1 score compared with DE+Metric in 0% elimination, it still gets higher F1 score in 40% elimination in IMDb. Plus, though the F1 scores of MSDs are not higher compared with DE+Metric in 20News, higher improved ratios of F1 scores are achieved by MSDs. Thus, MSD is also competitive in comparison with baselines in 20News.

3) **Effectiveness for each component by ablation setting:** Our proposed three components can be applied independently and further improve accuracy of uncertainty scores by combining them in the most situations, which shows effect of comprehensive consideration of uncertainty. In the 20News and Yelp datasets, when one or two components are added, we find consistent increase on F1 scores from MSD1 to MSD2, and from MSD2 to MSD3. Though the MSD3 does not achieve consistently higher improvement in various ratio eliminations in the IMDb and Amazon datasets, the performance of MSD2 is consistently higher than MSD1. It shows the effectiveness of self-ensembling in reducing the influence of uncertainty. Besides, the MSD3 achieves higher improvement of F1 scores in some eliminated ratios compared with MSD2 in the IMDb and Amazon datasets. We explain this as: the out-of-distribution testing texts do not distribute evenly in various eliminated ratios.

4.2.2 Results of Transformer and RNN model

Table 5 and Table 6 report the F1 score improvement in text classification with various eliminated ratios (10%, 20%, 30%, 40%) for BiGRU and XLnet respectively. Then, we conclude as below.

1) **Higher performance in macro F1 by MSD3:** From Tables 5 and 6, MSD3 achieves higher improved ratios of F1 scores in different eliminated ratios. Though the MSD2-b has higher F1 scores with eliminated ratios 10% and 20%, the other F1 scores of MSD3 in the two tables are still the highest in each eliminated ratio. The superior improvement of both F1 scores and ratios of F1 scores shows the joint effect of three components. Furthermore, the results of MSD2-b and MSD3 show the effect of distinctiveness scores for macro F1 in Amazon, which has imbalanced data distributions. Besides, though MSD2-a performs poorly by mix-up and self-ensembling, this performance is reasonable. Because XLnet is a pretrained model, we have parameters of only two FC layers to train, which has much less feasible solutions of possible parameters compared with the CNN and RNN models. Thus, further decrease of feasible solutions of possible parameters brings negative effect in this case.

2) **Flexibility of MSD:** From the Table 5 and Table 6 for RNN and Transformer respectively, as well as Tables 1, 2, 3, and 4 for CNN model, we can observe the competitive performance of MSD in text classification F1 scores compared with two baselines. This verifies that MSD is effective to assemble with other DNNs (CNN, RNN and Transformer). Besides, the ablation setting of MSD1, MSD2-a, MSD2-b and MSD3 shows that the three components in MSD can be assembled arbitrarily based on the characteristics of datasets.

4.2.3 Parameter Sensitivity Analysis

The impact of different Ω for the mix-up, various λ_2 for the self-ensembling and γ_1, γ_2 for the dis-

Table 1: Accuracy of uncertainty scores shown by improvement of macro F1 scores for the 20News (CNN model)

Methods (Ω, λ_2, γ_1, γ_2)	Uncertainty Ratio (Macro F1, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
DE	0.752	0.796(5.96%)	0.835(11.05%)	0.872(16.04%)	0.900(19.70%)
DE+Metric	0.774	**0.826**(6.70%)	**0.866**(11.97%)	**0.904**(16.87%)	**0.929**(20.02%)
MSD1 (1, 0, 1, 0)	0.751	0.808(**7.44%**)	0.854(13.50%)	0.894(**18.83%**)	0.923(**22.70%**)
MSD2 (1, 0.1, 1, 0)	0.760	0.812(6.92%)	0.849(11.73%)	0.886(16.59%)	0.920(21.47%)
MSD3 (1, 0.1, 1, 0.01)	0.760	0.812(6.95%)	0.856(**12.62%**)	0.889(16.98%)	0.921(21.22%)

Table 2: Accuracy of uncertainty scores shown by improvement of weighted F1 scores for the IMDb (CNN model)

Methods (Ω, λ_2, γ_1, γ_2)	Uncertainty Ratio (Weighted F1, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
DE	0.880	0.913(3.75%)	0.939(6.70%)	0.957(8.75%)	0.970(10.22%)
DE+Metric	0.884	**0.918**(3.85%)	**0.944**(6.79%)	0.961(8.71%)	0.974(10.18%)
MSD1 (1, 0, 1, 0)	0.874	0.907(3.87%)	0.933(6.79%)	0.952(8.95%)	0.967(**10.75%**)
MSD2 (1, 1, 1, 0)	0.883	**0.918**(3.92%)	**0.944**(6.82%)	0.961(8.85%)	**0.976**(10.46%)
MSD3 (1, 1, 1, 0.1)	0.882	**0.918**(**4.04%**)	0.943(**6.88%**)	**0.962**(**9.08%**)	0.974(10.49%)

Table 3: Accuracy of uncertainty scores shown by improvement of macro F1 scores for the Amazon (CNN model)

Methods (Ω, λ_2, γ_1, γ_2)	Uncertainty Ratio (Macro F1, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
DE	0.438	0.447(2.07%)	0.439(3.15%)	0.438(1.39%)	0.428(-2.18%)
DE+Metric	0.432	0.443(2.56%)	0.439(1.60%)	0.431(-0.31%)	0.418(-3.27%)
MSD1 (1, 0, 1, 0)	0.434	0.458(5.40%)	0.463(6.52%)	0.464(6.76%)	0.472(8.73%)
MSD2 (1, 0.1, 1, 0)	0.453	**0.480**(5.83%)	**0.502**(10.67%)	0.505(11.38%)	0.530(17.01%)
MSD3 (1, 0.1, 1, 0.1)	0.435	0.467(**7.44%**)	0.490(**12.82%**)	**0.520**(**19.57%**)	**0.550**(**26.64%**)

Table 4: Accuracy of uncertainty scores shown by improvement of macro F1 scores for the Yelp (CNN model)

Methods (Ω, λ_2, γ_1, γ_2)	Uncertainty Ratio (Macro F1, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
DE	0.562	0.583(3.61%)	0.598(6.44%)	0.614(9.16%)	0.629(11.84%)
DE+Metric	0.568	0.590(3.71%)	0.605(6.44%)	0.619(8.94%)	0.634(11.50%)
MSD1 (1, 0, 1, 0)	0.567	0.591(4.23%)	0.610(7.47%)	0.626(10.36%)	0.642(13.22%)
MSD2 (1, 0.1, 1, 0)	0.571	0.596(**4.41%**)	0.616(7.89%)	0.635(11.25%)	0.654(14.55%)
MSD3 (1, 0.1, 1, 0.01)	0.571	**0.597**(**4.41%**)	**0.617**(**7.94%**)	**0.636**(**11.30%**)	**0.655**(**14.63%**)

Table 5: Accuracy of uncertainty scores shown by improvement of macro F1 scores for the Amazon (BiGRU)

Methods (Ω, λ_2, γ_1, γ_2)	Uncertainty Ratio (Macro F1, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
DE	0.477	0.486(1.75%)	0.478(0.18%)	0.478(0.23%)	0.478(0.23%)
DE+Metric	0.471	0.478(1.50%)	0.466(-0.94%)	0.467(-0.84%)	0.466(-0.96%)
MSD1 (1, 0, 1, 0)	0.456	0.462(1.28%)	0.462(1.4%)	0.469(2.92%)	0.481(5.64%)
MSD2 (1, 0.1, 1, 0)	0.457	0.460(0.58%)	0.460(0.59%)	0.470(2.81%)	0.484(5.80%)
MSD3 (1, 0.1, 1, 0.1)	0.456	**0.497**(**8.88%**)	**0.524**(**14.95%**)	**0.531**(**16.39%**)	**0.508**(**11.36%**)

Table 6: Accuracy of uncertainty scores shown by improvement of macro F1 scores for the Amazon (XLnet)

Methods (Ω, λ_2, γ_1, γ_2)	Uncertainty Ratio (Macro F1, Improved Ratio)				
	0%	**10%**	**20%**	**30%**	**40%**
DE	0.422	0.422(0.00%)	0.428(1.38%)	0.423(0.26%)	0.424(0.38%)
DE+Metric	0.438	0.444(1.29%)	0.447(1.96%)	0.448(2.35%)	0.447(2.04%)
MSD1 (1, 0, 1, 0)	0.426	0.442(3.85%)	0.446(4.80%)	0.452(6.14%)	0.439(3.22%)
MSD2-a (1, 0.01, 1, 0)	0.415	0.436(5.03%)	0.440(6.06%)	0.434(4.46%)	0.422(1.56%)
MSD2-b (1, 0, 1, 1)	0.424	**0.451**(6.22%)	**0.470**(10.87%)	0.486(14.89%)	0.501(17.99%)
MSD3 (1, 0.01, 1, 1)	0.417	0.447(**7.16%**)	0.467(**11.96%**)	**0.487**(**16.81%**)	**0.509**(**21.95%**)

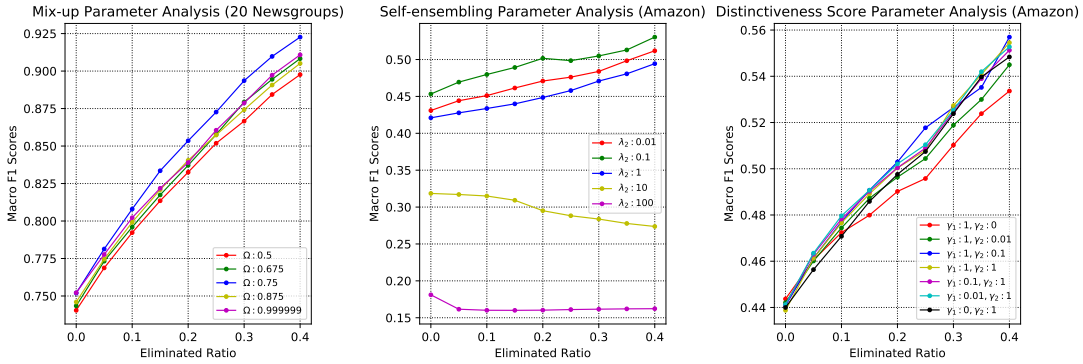

Figure 2: Diagrams for parameter sensitive analysis. The left panel shows how mix-up parameter Ω affects F1 scores. Middle panel shows how the self-ensembling parameter λ_2 affects F1 scores. Right panel shows how changes in γ_1 and γ_2 for distinctiveness score affect F1 scores.

tinctiveness score is discussed as below.

1) **Parameters for mix-up:** The left panel in Fig. 2 shows the effectiveness of different Ω. We apply $\Omega = 0.999999$ to approximate no mix-up. From the subfigure, we find: (1) the F1 scores are slightly sensitive to different Ω, while the improved ratios of F1 scores are not sensitive to the change of Ω. (2) For the 20News, when $\Omega = 0.75$, the macro F1 scores are the highest in different ratios, which are higher than the F1 scores of $\Omega = 0.999999$. This shows the effectiveness of mix-up in improving the accuracy of uncertainty score.

2) **Parameters for self-ensembling:** The impact of self-ensembling parameter λ_2 is shown in the middle panel in Fig. 2. This panel shows: (1) the F1 scores and their improved ratios in various eliminated ratios are significantly sensitive to λ_2, especially when λ_2 is greater than 1. (2) For Amazon dataset, macro F1 scores are the highest when $\lambda_2 = 0.1$ rather than $\lambda_2 = 0.01$. This again verifies the effectiveness of self-ensembling in improving the accuracy of uncertainty score.

3) **Parameters for distinctiveness score:** The right panel in Fig. 2 shows the impact of various γ_1 and γ_2 for distinctiveness score. We can see that: (1) the F1 scores are slightly sensitive to different γ_1 and γ_2. (2) The F1 scores of $\gamma_1 = 1, \gamma_2 = 0.1$ is around 2% higher compared with those of $\gamma_1 = 1, \gamma_2 = 0$, and nearly 1% higher compared with those of $\gamma_1 = 0, \gamma_2 = 1$. This presents the effect of distinctiveness score in generating more accurate uncertainty score.

5 Conclusion

We aims at generating more accurate uncertainty score to improve the performance of text classification with human involvement. We propose MSD with three independent components to improve the CWS by mitigating the effect of overconfidence and handling the impact of three categories of uncertainty. MSD can be applied to various DNNs (CNN, RNN and Transformer) and each component in MSD can be arbitrarily assembled. Extensive experiments on four real-world datasets demonstrate that MSD obtains more accurate uncertainty scores, and superiorly improved classification performance when partial uncertain predictions are simulatively assigned to the experts.

References

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. *arXiv preprint arXiv:1805.04604*.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org.

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*.

Mohammed Jabreel, Fadi Hassan, and Antonio Moreno. 2018. Target-dependent sentiment analysis of tweets using bidirectional gated recurrent neural networks. In *Advances in Hybridization of Intelligent Methods*, pages 39–55. Springer.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

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

Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. 2017. Learning curve prediction with bayesian neural networks. *International Conference on Learning Representations*.

Ananya Kumar, Percy S Liang, and Tengyu Ma. 2019. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems*, pages 3787–3798.

Samuli Laine and Timo Aila. 2016. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177.

Jeremiah Liu, John Paisley, Marianthi-Anna Kioumourtzoglou, and Brent Coull. 2019. Accurate uncertainty estimation and decomposition in ensemble learning. In *Advances in Neural Information Processing Systems*, pages 8950–8961.

Christos Louizos and Max Welling. 2017. Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org.

Andrey Malinin and Mark Gales. 2019. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 14520–14531.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632.

Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. 2018. Adversarial dropout for supervised and semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. 2017. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*.

Carlos Riquelme, George Tucker, and Jasper Snoek. 2018. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*.

Raanan Yehezkel Rohekar, Yaniv Gurwicz, Shami Nisimov, and Gal Novik. 2019. Modeling uncertainty by learning a hierarchy of deep neural connections. In *Advances in Neural Information Processing Systems*, pages 4246–4256.

Aili Shen, Daniel Beck, Bahar Salehi, Jianzhong Qi, and Timothy Baldwin. 2019. Modelling uncertainty in collaborative document quality assessment. In

Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019), pages 191–201.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. 2019. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. 2019. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 13888–13899.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan, and Maosong Sun. 2019. Improving back-translation with uncertainty-based confidence estimation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 791–802.

Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. 2018. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*.

Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7322–7329.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. 2019. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1284–1293.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Xuchao Zhang, Fanglan Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2019. Mitigating uncertainty in document classification. In *Proceedings of NAACL-HLT*, pages 3126–3136.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.