On the Calibration of Pre-trained Language Models using Mixup Guided by Area Under the Margin and Saliency

Seo Yeon Park and Cornelia Caragea
Computer Science
University of Illinois Chicago
spark313@uic.edu   cornelia@uic.edu

Abstract

A well-calibrated neural model produces confidence (probability outputs) closely approximated by the expected accuracy. While prior studies have shown that mixup training as a data augmentation technique can improve model calibration on image classification tasks, little is known about using mixup for model calibration on natural language understanding (NLU) tasks. In this paper, we explore mixup for model calibration on several NLU tasks and propose a novel mixup strategy for pre-trained language models that improves model calibration further. Our proposed mixup is guided by both the Area Under the Margin (AUM) statistic (Pleiss et al., 2020) and the saliency map of each sample (Simonyan et al., 2013). Moreover, we combine our mixup strategy with model miscalibration correction techniques (i.e., label smoothing and temperature scaling) and provide detailed analyses of their impact on our proposed mixup. We focus on systematically designing experiments on three NLU tasks: natural language inference, paraphrase detection, and commonsense reasoning. Our method achieves the lowest expected calibration error compared to strong baselines on both in-domain and out-of-domain test samples while maintaining competitive accuracy.

1 Introduction

Training a well-calibrated classifier that produces a match between confidence (the probability output that a model assigns to a prediction) and correctness (accuracy), is important in modern neural networks. As an example, if an AI-based application *knows what it does not know*, or in other words, the chance that the current prediction is wrong, a human is more helpful to correct the error. However, many works reveal that current deep neural networks are prone to over-confidence, which implies that the models' confidence is not reliable (Guo et al., 2017). This is a critical issue on the deployment of AI-based user applications such as the healthcare domain (Zhu et al., 2018; Li et al., 2019) or safety-critical domain (Sarabadani, 2019) due to the problem of prediction trustworthiness.

Recently, the study of calibration on neural network models especially on natural language processing tasks has started to receive attention. To overcome the problem of miscalibration, numerous suggestions on how to address it have been proposed. For example, Guo et al. (2017) revealed that using temperature scaling before the final softmax layer reduces calibration errors. Müller et al. (2019), Kumar and Sarawagi (2019), and Wang et al. (2020a) found that label smoothing and its variants yield better calibration for neural machine translation. Desai and Durrett (2020) also reported that the aforementioned miscalibration correction methods can be applied to calibrate pre-trained language models which are often miscalibrated potentially due to over-parameterization.

Mixup (Zhang et al., 2018) is a data augmentation method for deep neural networks in which additional samples are generated during training by combining random pairs of training inputs and their associated labels. While simple to implement, mixup has been shown to improve both predictive performance and model calibration, particularly on image classification tasks due to its regularization effect through data augmentation (Thulasidasan et al., 2019). The recent success of mixup on image classification has led to the development of various mixup strategies for NLU especially those that use hidden state representations (Guo et al., 2019a; Chen et al., 2020; Zhang et al., 2020; Sun et al., 2020; Kong et al., 2020; Yin et al., 2021). However, most prior works on NLU focus on performance improvement using mixup rather than model calibration. Despite its benefits for calibration, a mixup for correcting miscalibrated predictions is still an under-explored topic in NLU. While Kong et al. (2020) explored BERT (Devlin et al., 2019) cali-

5364

bration using mixup for both in-domain and out-of-domain, they only focused on generating mixup samples by utilizing the distance between instances in the feature space. In contrast, we propose a novel mixup method, in which we first leverage the behavior of a model on individual samples during training (training dynamics), which can reveal samples with distinct pronounced characteristics—whether they are easy-to-learn or hard-to-learn/ambiguous for the model, and then we generate mixup samples by mixing easy-to-learn with hard-to-learn/ambiguous samples according to their similarity/dissimilarity provided by saliency maps. Saliency maps capture how much each data portion contributes to the final classification decision of a sample (Simonyan et al., 2013). Intuitively, easy-to-learn samples help with model optimization, whereas hard-to-learn or potentially ambiguous samples are essential for learning since they are the most challenging for the model (Swayamdipta et al., 2020), and mixing them using saliency maps can yield better calibrated models (more realistic model confidence), e.g., mixing easy-to-learn with hard-to-learn/ambiguous samples by similarity in saliency maps can benefit in-domain calibration and by dissimilarity can benefit out-of-domain calibration. To monitor training dynamics, we use the Area Under the Margin (AUM) statistic (Pleiss et al., 2020) which measures how different a true label for a sample is compared to a model's *beliefs* at each epoch and is calculated as the average difference between the logit values for a sample's assigned class and its highest non-assigned class across training epochs.

Moreover, we combine our mixup with well-known miscalibration correction methods such as label smoothing and temperature scaling (Guo et al., 2017) to investigate their impact on our proposed mixup. We conduct a comprehensive set of experiments using BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) to show the efficacy of our mixup approach by testing on three NLU tasks: natural language inference, paraphrase detection, and commonsense reasoning. We achieve the lowest Expected Calibration Error (ECE) without accuracy drops in comparison with strong baseline methods. Our contributions are as follows:

- We propose a novel mixup method which is guided by AUM and saliency signals and is targeted at improving model calibration. Specifically, we compare logits to categorize samples into two sets (i.e., a set of easy-to-learn samples and another set of hard-to-learn/ambiguous samples), and interpolate samples across these two sets by finding the most similar and most dissimilar samples from the other set while leveraging saliency (to compute sample similarities) for pre-trained language models' calibration on in-domain and out-of-domain data.

- We combine our method with miscalibration correction techniques (i.e., label smoothing, temperature scaling) to investigate their impact on our proposed mixup.

- We conduct comprehensive experiments showing that our method achieves the lowest expected calibration errors (ECEs) on both in-domain and out-of-domain samples compared with strong baselines without accuracy drops on multiple NLU tasks, namely, natural language inferences, paraphrase detection, and commonsense reasoning.

2 Related Work

Model Calibration Calibration on NLU tasks has been widely studied in related literature. Nguyen and O'Connor (2015) provided the method of how to analyze the calibration of non-neural NLP models. Guo et al. (2017) examined the calibration of modern deep neural networks and revealed that techniques such as temperature scaling and dropout affect the calibration on binary/multiclass classification tasks. Wang et al. (2020b) investigated the calibration of neural machine translation models and found that inference suffers from serious miscalibration. Jagannatha and Yu (2020) demonstrated that neural networks show high calibration error on structured predictions such as NER, POS, and QA, and proposed to use a binary class forecaster to calibrate the predictor confidence for a defined output entity of interest. Desai and Durrett (2020) explored pre-trained language models' calibration in combination with temperature scaling and label smoothing both on in-domain and out-of-domain datasets. Jung et al. (2020) jointly optimized two objectives (a cross-entropy loss and a calibration loss) and directly penalized the difference between the predicted and the true posterior probabilities dynamically over the training steps. He et al. (2021) obtained better calibration on natural language understanding tasks by augmenting and training the classifier jointly with an energy-based model using noise-contrastive estimation.

Mixup Mixup (Zhang et al., 2018) is a method for data augmentation in which additional samples are generated during training by convexly combining random pairs and their associated labels, and aims to alleviate overfitting. Verma et al. (2019) showed that manipulating hidden representations rather than manipulating input-level features on mixup results in better regularization effects due to the fact that it encourages the neural network to focus more on representations of the real training examples in a low dimensional subspace. Many works have empirically noticed regularization effects that improve model performance on deep neural networks. For example, Guo et al. (2019a) explored the NLU specific mixup strategy by using sentence and word embeddings on CNNs and LSTMs to add performance gains in supervised text classification. Chen et al. (2020) proposed mixup for semi-supervised learning in which labeled and unlabeled samples are interpolated with their hidden representations to improve the performance of text classification. Zhang et al. (2020) explored mixup for sequence labeling tasks with active learning to improve the performance of supervised sequence labeling tasks. Yin et al. (2021) proposed mixup that interpolates every instance in a mini-batch to boost the performance of NLU tasks on the pre-trained language model RoBERTa (Liu et al., 2019). Similar to us, Yoon et al. (2021) explored mixup by incorporating saliency signals to generate augmented samples. Precisely, they use saliency signals to select a span of text from one sample to be replaced with another text span from another sample. However, in contrast, our method first divides data samples into two categories (easy-to-learn and hard-to-learn/ambiguous categories) according to their AUM (Pleiss et al., 2020) distribution monitored over training epochs and then uses saliency to find the most similar/dissimilar samples across these two data categories.

Recently, several works started to explore mixup for NLU model calibration. For example, Thulasidasan et al. (2019) investigated the impact of mixup for model calibration of NLU but only explored in-domain settings with simple deep learning architecture such as CNNs. Kong et al. (2020) explored BERT calibration using mixup as a regularization component on in-domain and out-of-domain. However, their mixup method only relied on the feature space distance between samples. In contrast, we explore a novel mixup method in which we categorize the training samples into two sets using AUM (Pleiss et al., 2020) and combine samples across these two sets based on saliency signals, for in-domain and out-of-domain model calibration.

3 Approach

3.1 Mixup

Background Let $\mathcal{D}_{train} = \{(x_i, y_i)\}_{i=1,\cdots,n}$ be a training set and f a language model. Mixup training generates vicinity training samples according to the rule introduced in Zhang et al. (2018):

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j$$
$$\tilde{y} = \lambda y_i + (1 - \lambda)y_j \tag{1}$$

where x_i and x_j are two randomly sampled input points, y_i and y_j are their associated one-hot encoded labels, and λ is a mixing ratio sampled from a Beta(α, α) distribution with a hyper-parameter α. In mixup, training data is augmented by linearly interpolating training samples in the input space.

3.2 Proposed Approach

We propose a mixup method targeted at improving model calibration that synthesizes samples guided by the Area Under the Margin (AUM) (Pleiss et al., 2020) and saliency (Simonyan et al., 2013).

Data Categorization In our method, we first categorize $\mathcal{D}_{train}$ into two sets (a set of easy-to-learn samples and a set of hard-to-learn/ambiguous samples) according to the AUM of each sample. Given a sample (x_i, y_i), we compute $AUM(x_i, y_i)$ as the area under the margin averaged across all training epochs T. Specifically, at some epoch $t \in T$, the margin is defined as:

$$M^t(x_i, y_i) = z_{y_i} - max_{y_i!=k}(z_k) \tag{2}$$

where $M^t(x_i, y_i)$ is the margin of example x_i with gold label y_i, z_{y_i} is the logit corresponding to the gold label y_i, and $max_{y_i!=k}(z_k)$ is the largest *other* logit corresponding to label k not equal to y_i. Precisely, the margin measures how different a gold label is compared to a model's *beliefs* at each epoch t. The AUM of (x_i, y_i) across all epochs is:

$$AUM(x_i, y_i) = \frac{1}{T}\sum_{t=1}^{T} M^t(x_i, y_i) \tag{3}$$

Intuitively, the samples with high AUM are easy-to-learn (the model's belief matches the gold label), but they are essential for model optimization, while

Algorithm 1 : Identify high/low AUM samples

Require: $\mathcal{D}_{train} = \{(x_i, y_i)\}_{i=1,\cdots,n}$; model f
1: **function** DATA-CATEGORIZATION($\mathcal{D}_{train}$)
2:     $\mathcal{D}_{high} \leftarrow \emptyset, \mathcal{D}_{low} \leftarrow \emptyset$
3:     Train f for T epochs and compute $AUM(x_i, y_i)$ for each i as in Eq. (3)
4:     **for** each $(x_i, y_i) \in \mathcal{D}_{train}$ **do**
5:         **if** $AUM(x_i, y_i) < median$ **then**
6:             $\mathcal{D}_{low} \leftarrow \mathcal{D}_{low} \cup (x_i, y_i)$
7:         **else if** $AUM(x_i, y_i) \geq median$ **then**
8:             $\mathcal{D}_{high} \leftarrow \mathcal{D}_{high} \cup (x_i, y_i)$
9:         **end if**
10:    **end for**
11:    **return** $\mathcal{D}_{high}, \mathcal{D}_{low}$
12: **end function**

the samples with low AUM are hard-to-learn or ambiguous (and hence they are the most challenging for the model), but they are essential for learning. Our proposed mixup method first splits $\mathcal{D}_{train}$ into two data categories depending on whether the AUM value is high or low, namely, $\mathcal{D}_{high}$ and $\mathcal{D}_{low}$. In experiments, we compute the median AUM over the entire training samples and use it as a threshold to split the dataset. If a sample has a lower AUM than the threshold, we add the sample to $\mathcal{D}_{low}$, otherwise we add it to $\mathcal{D}_{high}$. Accordingly, we balance $\mathcal{D}_{high}$ and $\mathcal{D}_{low}$, but other splits are possible. We then conduct a mixup operation by referring to each other set. Mixing easy-to-learn and hard-to-learn adjusts the difficulty of samples and hence adjusts models' confidence according to samples' difficulties and yields better calibrated models. The data categorization step is summarized in Algorithm 1.

Mixup using Saliency Signals We conduct a mixup operation on the two data categories generated by Algorithm 1 using saliency signals (as detailed below). For the mixup, rather than selecting random samples from $\mathcal{D}_{high}$ and $\mathcal{D}_{low}$ to mix, we utilize saliency signals to select samples. To measure saliency, gradient-based methods are usually used for saliency computation (Li et al., 2016; Rei and Søgaard, 2018; Yoon et al., 2021). Following this idea, we simply compute the gradient of the classification loss L with respect to each logit value $z_i \in \mathbf{z}$ and take the absolute value of the gradient components as the saliency map or signature S for a sample $(x_i, y_i) \in \mathcal{D}_{train}$. For a sample (x_i, y_i), we then leverage its saliency map S to find the most similar and most dissimilar samples from

Algorithm 2 : Proposed Mixup

Require: $\mathcal{D}_{train} = \{(x_i, y_i)\}_{i=1,\cdots,n}$; model f
1: $\mathcal{D}_{high}, \mathcal{D}_{low} \leftarrow$ DATA-CATEGORIZATION($\mathcal{D}_{train}$)
2: **for** $k := 0$ to T **do**
3:    $Total\_Loss \leftarrow 0$
4:    **for** $i := 0$ to $|\mathcal{D}_{train}|$ **do**
5:       $Loss \leftarrow CrossEntropy(f(x_i), y_i)$
6:       Construct a saliency map S by computing the gradient of $Loss$ with respect to $\mathbf{z}$
7:       **if** $(x_i, y_i) \in \mathcal{D}_{high}$ **then**:
8:          Find the most similar/dissimilar samples from $\mathcal{D}_{low}$ using Eq. (4)
9:       **else if** $(x_i, y_i) \in \mathcal{D}_{low}$ **then**:
10:      Find the most similar/dissimilar samples from $\mathcal{D}_{high}$ using Eq. (5)
11:     **end if**
12:     Generate two mixup samples, one for (x_i, y_i) and its most similar sample and another for (x_i, y_i) and its most dissimilar sample, using Eq. 1.
13:     Compute $CrossEntropy$ loss for each mixup sample
14:     $Loss \leftarrow \beta Loss + \gamma Loss' + \delta Loss''$
15:    **end for**
16:    $Total\_Loss \leftarrow Total\_Loss + Loss$
17:    Update the model weights
18: **end for**

the other data category that (x_i, y_i) does not belong to according to its AUM, in order to calibrate in-domain and out-of-domain data. For example, if $(x_i, y_i) \in \mathcal{D}_{high}$, we find its most similar sample (x_i', y_i') and its most dissimilar sample (x_i'', y_i'') from $\mathcal{D}_{low}$, that return the largest and smallest cosine similarity, respectively, with the saliency map S of (x_i, y_i). That is, the most similar and most dissimilar samples to $(x_i, y_i) \in \mathcal{D}_{high}$ are calculated as follows:

$$(x_i', y_i') = \underset{(x_j, y_j) \in \mathcal{D}_{low}}{\operatorname{argmax}} \; CosSim(S, S^{(x_j, y_j)})$$
$$(x_i'', y_i'') = \underset{(x_j, y_j) \in \mathcal{D}_{low}}{\operatorname{argmin}} \; CosSim(S, S^{(x_j, y_j)})$$

$$(4)$$

Similarly, if (x_i, y_i) belongs to $\mathcal{D}_{low}$, we find the most similar/dissimilar samples from $\mathcal{D}_{high}$ that return the largest/smallest cosine similarity with S

as follows:

$$(x_i', y_i') = \underset{(x_j, y_j) \in \mathcal{D}_{high}}{\text{argmax}} \; CosSim(S, S^{(x_j, y_j)})$$

$$(x_i'', y_i'') = \underset{(x_j, y_j) \in \mathcal{D}_{high}}{\text{argmin}} \; CosSim(S, S^{(x_j, y_j)})$$

$$(5)$$

We then generate two mixup samples for a given sample (x_i, y_i) by interpolating the selected samples, which are the most similar sample (x_i', y_i'), and the most dissimilar sample (x_i'', y_i''). For the mixup operation, we follow the original mixing ratio sampling strategy which is shown in Eq. (1). The ratio λ is sampled from a Beta(α, α) distribution with a hyper-parameter α.

Intuitively, by synthesizing the original sample and the most similar sample from the other data category, we calibrate in-domain data. The augmented sample mimics in-domain sample since it aligns the most with the original sample. Furthermore, by selecting the sample from the other category, we allow the generated mixup sample to combine easy-to-learn and hard-to-learn samples properly. By synthesizing the original and the most dissimilar sample from the other data category, we calibrate out-of-domain data. The augmented sample mimics out-of-domain instances since we pick a sample that is the most dissimilar to the original sample. As above, by selecting the sample from the other category, we allow the augmented sample to contain both information of easy-to-learn and hard-to-learn samples, useful for both optimization and learning. Note that our mixup method mixes samples on the level of `[CLS]` hidden state representations generated by task-specific layer on top of the pre-trained language model. We summarize the process in Algorithm 2. We combine each loss by weighted sum (see Alg. 2) where β, γ, δ are hyper-parameters that sum up to 1. In our experiments, we conduct our mixup operation using mini-batch SGD to update the model weights. Note that other saliency measures are possible to compute similarity/dissimilarity between samples and will be an interesting future direction.

3.3 Calibration Metrics

A model is perfectly calibrated when the confidence estimate $\hat{p}$ of the model is equal to true probability (accuracy) $\mathbb{P}(\hat{y} = y | \hat{p}) = \hat{p}$. (Naeini et al., 2015; Guo et al., 2017; Desai and Durrett, 2020). This can be empirically approximated by discretizing the probability interval into a fixed number of

bins $M = 10$ where each bin b_m contains predicted probabilities that encompass the interval. The expected calibration error (ECE) is calculated by weighting the average of the difference between each bin's accuracy and confidence as follows:

$$\text{acc}(b_m) = \frac{1}{|b_m|} \sum_{i \in b_m} \mathbb{1}(\hat{y}_i = y_i)$$

$$\text{conf}(b_m) = \frac{1}{|b_m|} \sum_{i \in b_m} \hat{p}_i$$

$$\text{ECE} = \sum_{m=1}^{M} \frac{|b_m|}{N} |\text{acc}(b_m) - \text{conf}(b_m)|$$

where N is the total number of predictions.

3.4 Miscalibration Correction Methods

We explore the combination of miscalibration correction methods (described below) with mixup to investigate their impact on our proposed mixup for model calibration.

Label Smoothing (LS) In supervised learning, one-hot encoded labels fail to provide uncertainty of inputs due to the fact that all the probability mass is given to one class. This results in over-confident models since the largest logit becomes larger than the others which removes the uncertainty of label space. Label smoothing (LS) is a solution to penalize this by preventing the models from becoming over-confident. In this work, we incorporate label smoothing with our proposed mixup. We generate smoothed one-hot target signal while creating mixup instances by distributing $\frac{\sigma}{|y|-1}$ mass over non ground-truth classes, where $\sigma \in (0, 1)$ is a hyper-parameter and $|y|$ is the number of classes.[1]

Temperature Scaling (TS) Temperature scaling (TS) is a post-processing step which re-scales the logit vector **z** using a single scale parameter temperature, $T > 0$ for all classes. TS has the effect of softening the outputs to be uniform with $T > 1$, while $T \to 0$ has the effect of collapsing probability mass to one class. We explore the effect of TS when incorporated with our proposed mixup.

4 Experiments

4.1 Tasks and Datasets

We evaluate our calibration-targeted mixup on three natural language understanding tasks: natural lan-

---

[1]For example, the smoothed one-hot target of [1,0,0] is [0.99, 0.0005, 0.0005] when $\sigma = 0.001$ and $|y| = 3$.

guage inference, paraphrase detection, and commonsense reasoning. We evaluate the models in-domain (training and testing on data from the same distribution) and out-of-domain (training and testing on data from different distributions). Mixup reduces the number of undesirable oscillations when predicting especially on out-of-distribution samples (Zhang et al., 2018). Hence, effective mixup should be less prone to over-fitting when handling out-of-distribution data. To test the benefits of our proposed method for pre-trained language model calibration, we use in-domain trained models to predict out-of-distribution test samples. We describe our in-domain and out-of-domain sets as follows.

Natural Language Inference Stanford Natural Language Inference (SNLI) is a natural language inference task to predict if the relation between a hypothesis and a premise is *entailment, contradiction,* or *neutral* (Bowman et al., 2015). Multi-Genre Natural Language Inference (MNLI) captures natural language inference with more diverse domains (Williams et al., 2018) than SNLI.

Paraphrase Detection Quora Question Pairs (QQP) is a paraphrase detection task to test if two questions are semantically equivalent (Iyer et al., 2017). TwitterPPDB (TPPDB) is to determine whether sentence pairs from Twitter convey similar semantics when they share URLs (Lan et al., 2017)

Commonsense Reasoning Situations With Adversarial Generations (SWAG) is a commonsense reasoning task to choose the most plausible continuation of a sentence among four candidates (Zellers et al., 2018). HellaSWAG is a dataset built using adversarial filtering to generate challenging out-of-domain samples. It is distributionally different in that its examples exploit statistical biases in pre-trained models.

4.2 Comparison Methods

In this work, we explore the mixup effects on NLU with the goal of producing better calibrated models, in particular pre-trained language models, which are BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). We consider the following baselines:

- **Pre-trained Language Models** : Pre-trained language models fine-tuning on each downstream task using BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019).

- **Mixup** (Zhang et al., 2018; Thulasidasan et al., 2019): Mixup augments training data by linearly interpolating randomly selected training samples in the input space. The interpolation of Mixup is performed on the input embeddings obtained from the first layer of the language model.

- **Manifold-mixup (M-mixup)** (Verma et al., 2019) : An extension of Mixup, which interpolates training samples in the hidden feature space. The interpolation of Manifold-mixup is performed on the features obtained from the last layer of the language model.

Each method is compared with two variants where miscalibration correction methods (label smoothing, LS and temperature scaling, TS) are applied.[2]

4.3 Implementation Details

We use the same set of hyper-parameters across all tasks as Desai and Durrett (2020) for a fair comparison. We train models with a maximum of 3 epochs. For BERT, we set batch size of 16, a learning rate of 1e-5, gradient clip of 1.0, and no weight decay. For RoBERTa, we set batch size of 32, a learning rate of 2e-5, gradient clip of 1.0, and weight decay of 0.1. We follow the published train/validation/test split by Desai and Durrett (2020).[3] For mixup, we use a mixing ratio sampling strategy hyper-parameter $\alpha = 0.4$. We use loss weight hyper-parameters, β, γ, δ, values as $0.8/0.1/0.1$ respectively. We did hyper-parameter search for label smoothing $\sigma \in [0.001, 0.003, 0.01, 0.03, 0.1, 0.3]$. We use $\sigma = 0.01/0.03/0.3$ for BERT, $\sigma = 0.003/0.03/0.3$ for RoBERTa on SNLI/QQP/SWAG, respectively. We use threshold values for splitting data into two groups $\mathcal{D}_{high}$ and $\mathcal{D}_{low}$ (the median AUM over full training samples) as $3.5/4.4/2.5$ for BERT, $3.4/4.0/2.7$ for RoBERTa on SNLI/QQP/SWAG, respectively. For all results, we report the mean across five training runs with random restarts. Finally, all experiments are conducted on a single NVIDIA RTX A5000 24G GPU with a total time for fine-tuning all models being under 24 hours. Temperature scaling (TS) searches are performed in the range of [0.01,5.0] with a granularity of 0.01 using development datasets. TS is completed very fast since it uses separate cached logits.

---

[2]For vanila pre-trained language models with/without label smoothing results, we use the reported results from Desai and Durrett (2020).

[3]https://github.com/shreydesai/calibration

	In-Domain						Out-of-Domain					
	SNLI		QQP		SWAG		MNLI		TwitterPPDB		HellaSWAG	
	No TS	TS	No TS	TS	No TS	TS	No TS	TS	No TS	TS	No TS	TS
BERT	$2.54_{0.8}$	$1.14_{1.0}$	$2.71_{0.5}$	$0.97_{0.1}$	$2.49_{1.8}$	$0.85_{0.4}$	$7.09_{2.1}$	$3.61_{1.7}$	$8.51_{0.6}$	$7.15_{0.9}$	$12.62_{2.8}$	$12.83_{2.1}$
BERT + LS	$7.12_{0.3}$	$8.37_{0.5}$	$6.33_{0.4}$	$8.16_{0.7}$	$10.01_{1.0}$	$10.89_{1.1}$	$3.74_{1.4}$	$4.05_{0.9}$	$6.30_{0.8}$	$5.78_{0.7}$	$5.73_{0.6}$	$5.34_{0.9}$
Mixup	$7.73_{1.1}$	$3.18_{0.9}$	$9.04_{0.8}$	$3.36_{1.1}$	$7.08_{1.0}$	$2.08_{0.6}$	$19.51_{2.1}$	$3.56_{1.7}$	$11.70_{1.6}$	$5.03_{1.3}$	$10.93_{2.0}$	$4.24_{1.6}$
Mixup + LS	$7.92_{1.0}$	$2.63_{0.8}$	$9.65_{0.4}$	$2.49_{1.3}$	$7.44_{0.7}$	$1.15_{0.2}$	$18.57_{1.2}$	$2.31_{1.0}$	$11.16_{0.8}$	$4.58_{1.1}$	$8.57_{1.3}$	$3.95_{1.1}$
M-Mixup	$3.17_{0.8}$	$1.77_{0.3}$	$8.55_{1.2}$	$6.11_{1.1}$	$5.18_{0.6}$	$1.09_{0.4}$	$12.92_{2.6}$	$2.34_{1.9}$	$12.10_{2.3}$	$7.98_{2.6}$	$9.82_{1.2}$	$5.12_{0.9}$
M-Mixup + LS	$3.40_{0.4}$	$5.14_{0.7}$	$3.49_{0.2}$	$3.71_{0.7}$	$5.24_{0.5}$	$1.26_{0.2}$	$16.76_{1.3}$	$4.57_{0.9}$	$6.29_{1.1}$	$6.54_{1.7}$	$8.32_{0.7}$	$3.64_{0.6}$
Ours	$\mathbf{1.29_{0.4}}$	$\mathbf{0.77_{0.7}}$	$2.05_{0.6}$	$1.02_{0.6}$	$\mathbf{2.01_{0.4}}$	$0.81_{0.2}$	$2.73_{2.5}$	$3.50_{0.6}$	$5.69_{0.7}$	$\mathbf{3.16_{1.2}}$	$5.49_{1.9}$	$4.11_{1.5}$
Ours + LS	$1.85_{0.3}$	$1.05_{1.0}$	$\mathbf{1.70_{0.9}}$	$\mathbf{0.95_{0.1}}$	$2.09_{0.7}$	$\mathbf{0.79_{0.3}}$	$\mathbf{2.26_{1.0}}$	$\mathbf{1.70_{0.5}}$	$5.37_{1.0}$	$3.54_{1.1}$	$\mathbf{4.26_{0.8}}$	$\mathbf{3.28_{0.7}}$
	In-Domain						Out-of-Domain					
	SNLI		QQP		SWAG		MNLI		TwitterPPDB		HellaSWAG	
	No TS	TS	No TS	TS	No TS	TS	No TS	TS	No TS	TS	No TS	TS
RoBERTa	$1.93_{0.5}$	$0.84_{0.8}$	$2.33_{0.1}$	$0.88_{0.6}$	$1.76_{1.0}$	$\mathbf{0.76_{0.7}}$	$3.62_{3.2}$	$1.46_{2.5}$	$9.55_{0.6}$	$7.86_{0.5}$	$11.93_{3.2}$	$11.22_{2.9}$
RoBERTa + LS	$6.38_{0.6}$	$8.70_{1.0}$	$6.11_{0.3}$	$8.69_{0.6}$	$8.81_{0.3}$	$11.40_{0.6}$	$4.50_{1.4}$	$5.93_{1.9}$	$8.91_{0.3}$	$5.31_{0.7}$	$2.14_{1.4}$	$2.23_{1.1}$
Mixup	$7.67_{0.8}$	$4.51_{0.7}$	$3.41_{0.5}$	$1.64_{0.6}$	$3.60_{0.9}$	$1.03_{0.9}$	$16.85_{1.3}$	$5.65_{0.9}$	$11.03_{0.9}$	$5.41_{0.8}$	$7.02_{0.2}$	$3.90_{0.6}$
Mixup + LS	$6.10_{0.7}$	$1.99_{0.5}$	$6.56_{0.9}$	$2.96_{0.5}$	$2.52_{0.1}$	$0.85_{0.3}$	$10.89_{1.1}$	$1.82_{0.3}$	$9.01_{1.6}$	$3.09_{1.1}$	$7.75_{1.7}$	$2.41_{0.7}$
M-Mixup	$7.32_{0.8}$	$4.56_{0.4}$	$3.54_{0.5}$	$5.05_{0.6}$	$1.68_{1.2}$	$0.96_{0.3}$	$19.78_{3.1}$	$7.65_{1.3}$	$7.18_{1.8}$	$8.76_{2.1}$	$5.63_{2.8}$	$3.43_{1.5}$
M-Mixup + LS	$3.51_{1.0}$	$3.00_{0.9}$	$2.82_{0.7}$	$3.03_{0.6}$	$1.83_{1.5}$	$0.94_{0.4}$	$8.23_{1.6}$	$5.08_{1.0}$	$6.17_{0.9}$	$6.91_{1.1}$	$4.27_{0.6}$	$2.88_{1.6}$
Ours	$1.34_{0.7}$	$\mathbf{0.63_{0.5}}$	$2.47_{0.6}$	$1.41_{0.2}$	$1.24_{0.1}$	$1.03_{0.2}$	$1.41_{1.9}$	$\mathbf{1.18_{1.4}}$	$\mathbf{3.94_{0.9}}$	$\mathbf{1.89_{1.2}}$	$2.40_{1.8}$	$2.08_{1.5}$
Ours + LS	$\mathbf{1.28_{0.6}}$	$1.02_{0.6}$	$\mathbf{2.18_{0.7}}$	$\mathbf{0.84_{0.4}}$	$\mathbf{1.12_{0.4}}$	$\mathbf{0.81_{0.1}}$	$\mathbf{1.37_{1.7}}$	$1.60_{1.3}$	$3.96_{1.6}$	$2.67_{1.8}$	$\mathbf{1.86_{0.9}}$	$\mathbf{1.70_{1.2}}$

Table 1: Expected Calibration Error (ECE) in percentage (%) on BERT (top) and RoBERTa (bottom). Bold text shows the best ECE. Lower ECE implies better-calibrated models. We report the mean ECE across five runs with random restarts. The subscript represents the corresponding standard deviation (e.g., $1.29_{0.4}$ indicates 1.29 ± 0.4).

4.4 Results

We show the comparison of experimental results (ECE) on BERT and RoBERTa in Table 1. For each task, we train the model on in-domain training set, and evaluate its expected calibration errors (ECEs) on in-domain and out-of-domain test sets. We make the following observations:

First, for in-domain data, label smoothing (LS) does not exhibit its effectiveness on pre-trained language models' calibration. Specifically, for in-domain data, pre-trained language models with LS (i.e., BERT+LS/RoBERTa+LS) achieve higher expected calibration errors (ECEs) compared with vanilla pre-trained language models (i.e., BERT/RoBERTa) on all tasks. In contrast, out-of-domain gains benefit from LS (except RoBERTa on MNLI). From these results, we conclude that simply incorporating label uncertainty (through label smoothing) is not an effective regularization method since LS does not consistently improve the model calibration (especially for the in-domain setting). While temperature scaling (TS) corrects the miscalibration of vanilla pre-trained language models (see BERT/RoBERTa No TS vs. TS in the table), it fails to correct miscalibrated pre-trained language models with LS (see BERT+LS/RoBERTa+LS No TS vs.

TS) in-domain. Interestingly, for some cases of out-of-domain data, pre-trained language models with LS show comparatively low ECEs while TS further reduces ECEs (e.g., BERT(LS) on TwitterPPDB/HellaSWAG, RoBERTa(LS) on TwitterP-PDB). However, its impact is not enough as it still results in high ECE. This implies that TS is not a notable strategy either to pre-trained language models' calibration. Accordingly, we conclude that stronger regularization techniques are required to calibrate the pre-trained language models.

Second, we find that mixup on the hidden feature space (i.e., M-Mixup) generally yields lower ECE than mixup on the input embedding space (i.e., Mixup) on most tasks. We infer that Mixup generates augmented samples that are not "good" for model calibration (i.e., semantically or syntactically) and fails to encourage regularization effects that arise from mixup. We observe that mixup training with LS is beneficial to reduce ECEs on some tasks. We find that TS leads to much lower ECEs on Mixup and M-Mixup (with and without LS) on most tasks. However, this implies that baseline mixup methods fail to produce well-calibrated models independently (without LS or TS). This supports our intuition and motivation for the design of a more robust approach of mixup.

	In-Domain			Out-of-Domain		
	SNLI	QQP	SWAG	MNLI	TwitterPPDB	HellaSWAG
BERT	$90.04_{0.3}$	$90.27_{0.3}$	$79.40_{0.4}$	$73.52_{0.3}$	$87.63_{0.4}$	$34.48_{0.2}$
BERT + LS	$87.11_{0.8}$	$87.51_{0.4}$	$74.91_{0.3}$	$72.06_{1.2}$	$87.82_{0.6}$	$36.48_{1.8}$
Mixup	$88.82_{0.2}$	$89.12_{0.5}$	$74.98_{2.3}$	$69.19_{0.8}$	$87.45_{0.3}$	$33.22_{0.4}$
Mixup + LS	$88.74_{0.4}$	$89.24_{0.2}$	$75.75_{0.5}$	$69.37_{1.1}$	$87.69_{0.6}$	$35.65_{1.7}$
M-Mixup	$86.40_{0.3}$	$89.37_{0.6}$	$76.96_{0.4}$	$66.61_{0.6}$	$86.51_{0.8}$	$34.57_{1.4}$
M-Mixup + LS	$87.50_{0.7}$	$87.17_{0.6}$	$76.09_{0.9}$	$64.88_{0.9}$	$86.55_{1.1}$	$33.71_{0.6}$
Ours	$90.01_{0.4}$	$90.13_{0.2}$	$78.94_{0.8}$	$73.48_{0.4}$	$88.04_{0.7}$	$34.63_{0.4}$
Ours + LS	$90.14_{0.3}$	$90.32_{0.2}$	$79.26_{0.6}$	$72.36_{0.6}$	$87.62_{0.9}$	$34.97_{0.5}$
	In-Domain			Out-of-Domain		
	SNLI	QQP	SWAG	MNLI	TwitterPPDB	HellaSWAG
RoBERTa	$91.23_{0.3}$	$91.11_{0.2}$	$82.45_{1.2}$	$78.79_{0.2}$	$86.72_{0.2}$	$41.68_{1.1}$
RoBERTa + LS	$89.73_{0.4}$	$87.64_{0.4}$	$79.13_{0.4}$	$77.40_{0.5}$	$87.48_{1.2}$	$40.05_{0.9}$
Mixup	$90.59_{0.4}$	$89.20_{1.4}$	$79.91_{1.5}$	$75.74_{0.7}$	$84.74_{0.6}$	$40.92_{1.4}$
Mixup + LS	$90.44_{0.6}$	$87.45_{0.7}$	$79.16_{0.4}$	$76.44_{1.0}$	$87.48_{0.4}$	$39.95_{1.0}$
M-Mixup	$90.30_{0.5}$	$89.47_{0.7}$	$73.79_{0.8}$	$73.69_{1.0}$	$86.04_{0.7}$	$41.60_{0.8}$
M-Mixup + LS	$90.97_{0.4}$	$88.44_{1.0}$	$79.61_{0.6}$	$75.55_{0.9}$	$86.49_{1.5}$	$41.88_{1.1}$
Ours	$91.61_{0.5}$	$89.19_{0.4}$	$81.47_{0.8}$	$78.01_{0.6}$	$87.13_{0.8}$	$40.95_{1.4}$
Ours + LS	$91.24_{0.3}$	$89.75_{0.6}$	$82.69_{0.7}$	$78.86_{0.5}$	$87.63_{1.0}$	$41.37_{1.1}$

Table 2: The comparison of accuracy (%) on BERT (top) and RoBERTa (bottom). We report the mean accuracy across five training runs with the standard deviation shown in subscript (e.g., $90.01_{0.4}$ indicates 90.01 ± 0.4).

Third, we observe that our proposed mixup yields the best calibrated models (lowest ECEs) both on in-domain and out-of-domain data (except on SWAG with RoBERTa). We observe that often LS effectively operates along with our proposed mixup and achieves the lowest ECEs on most tasks on in-domain and out-of-domain settings. In contrast to baseline mixup methods, our proposed mixup performs well on in-domain and out-of-domain even without applying post-calibration correction TS (see ECE values of baselines compared with our ECE values). We also observe that TS improves the model calibration further on our mixup training in most cases. Accordingly, we confirm the robustness of our AUM and saliency guided mixup for pre-trained language models calibration.

Accuracy We explore the accuracy of mixup training and show comparisons in Table 2. We make the following observations: 1) Both BERT+LS/RoBERTa+LS generally lead to substantial accuracy drops especially on in-domain compared with BERT/RoBERTa (i.e., 4.49% accuracy drops on SWAG). This implies that label smoothing (LS) fails to improve model generalization by simply manipulating labels (changing from hard to soft labels). This potentially leads to a loss of information that is correlated to model generalization (Müller et al., 2019). 2) Mixup and M-Mixup

fail to achieve an accuracy that is as good as that of vanilla pre-trained language models, potentially due to an increased chance of manifold intrusion resulting from conflicts between the synthetic samples of the mixup and original training data (Guo et al., 2019b). 3) In contrast, our proposed mixup method generally achieves competitive accuracy regardless of applying LS or not. This evidence supports the robustness of our proposed mixup. Note that TS does not affect the model's accuracy because it does not change the maximum of the softmax function.

4.5 Ablation Study

Effect of AUM and Saliency We investigate the effectiveness of each component (i.e., AUM and saliency) in our proposed mixup. As shown in Table 3, our proposed mixup without the AUM (i.e., -AUM) and without saliency (i.e., -Saliency) generally increase the expected calibration errors. In our method without using AUM, we randomly divide training data into two categories and conduct mixup operation based on saliency map. In our method without using saliency, we randomly pick two samples from the opposite low and high AUM set and conduct mixup operation. The results demonstrate that both metrics (AUM and saliency) are required to improve model calibration.

		In-Domain						Out-of-domain					
		SNLI		QQP		SWAG		MNLI		TwitterPPDB		HellaSWAG	
		No TS	TS	No TS	TS	No TS	TS	No TS	TS	No TS	TS	No TS	TS
BERT	Ours	1.85	1.05	1.70	0.95	2.09	0.79	2.26	1.70	5.37	3.54	4.26	3.28
	- AUM	2.74	0.95	4.43	1.39	2.15	1.44	7.74	1.68	9.08	4.17	11.45	2.21
	- Saliency	2.34	3.16	5.97	4.94	4.19	1.11	9.51	4.14	6.03	6.79	7.91	4.28
	- dissimilar	0.60	0.76	1.51	1.16	4.33	0.81	4.91	2.52	8.33	4.32	12.60	6.38
	- similar	3.76	4.94	2.88	1.98	5.58	2.87	8.38	3.07	7.67	5.55	18.91	3.24
RoBERTa	Ours	1.28	1.08	2.18	0.84	1.12	0.81	1.37	1.60	3.96	2.67	1.86	1.70
	- AUM	5.18	2.25	3.59	0.79	2.31	1.39	11.29	5.75	8.09	1.78	12.46	3.79
	- Saliency	2.91	2.63	0.98	1.02	1.41	1.27	4.80	4.54	6.92	4.78	6.82	3.37
	- dissimilar	2.01	0.93	2.98	1.58	2.52	0.73	6.69	4.77	5.17	4.54	11.39	6.43
	- similar	2.69	2.33	5.14	3.40	3.10	2.62	2.01	1.84	10.43	8.94	7.87	6.11

Table 3: Ablation study to investigate the effect of each component in our proposed mixup. We report results (% ECE) of our mixup without using AUM (i.e., -AUM), without using saliency (i.e., -Saliency), without utilizing the most dissimilar sample selected from the other data category obtained by AUM (i.e., -dissimilar), and without utilizing the most similar sample selected from the other data category obtained by AUM (i.e., -similar).

Effect of selecting the most similar and dissimilar samples We explore the effectiveness of selecting the most similar and dissimilar samples, which are used for mixing purposes for in-domain and out-of-domain calibration, respectively. Specifically, in our proposed mixup, we synthesize additional samples that mimic in-domain data by selecting the most similar sample from the other category (e.g., an easy-to-learn sample is mixed with a hard-to-learn/ambiguous sample that is most similar to the easy-to-learn sample, by saliency maps). This is because the selected sample aligns the most with the given sample. This intuitively results in better model generalization due to the effect arising from data augmentation (i.e., augmenting samples that are particularly similar to in-domain data) and allows better in-domain calibration. Similarly, we calibrate out-of-domain by augmenting a sample that mimics out-of-domain distribution. This is because we select the sample that is the most different from a given sample by selecting the most dissimilar sample from the other category. To verify this intuition, we conduct our proposed mixup when excluding the most similar instance (i.e., -similar) and the most dissimilar instance (i.e., -dissimilar), respectively.

Table 3 shows the results of this ablation. We observe that our proposed mixup without using the most dissimilar sample (i.e., -dissimilar) results in higher ECEs compared with our mixup that uses dissimilar samples on all tasks in the out-of-domain setting for both BERT and RoBERTa. Interestingly, we observe that our proposed mixup without

using the most similar sample (i.e., -similar) results in higher ECEs compared with our mixup that uses the most similar samples on in-domain and out-of-domain data for both BERT and RoBERTa. These results support that selecting the most similar/dissimilar samples effectively calibrates pre-trained models for in-domain/out-of-domain data.

5 Conclusion

We proposed a novel mixup guided by the Area Under the Margins (AUM) and saliency maps to mitigate the miscalibration of pre-trained language models BERT and RoBERTa. We showed that our proposed mixup method achieves the lowest Expected Calibration Errors (ECEs) for both pre-trained language models on various types of natural language understanding tasks, for both in-domain and out-of-domain data. For future work, we will enhance our proposed mixup further, focusing not only on model calibration but also on performance gains. Exploring different saliency maps for computing sample similarity/disimilarity (and its degree) is another interesting future direction.

Acknowledgements

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019a. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019b. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3714–3722.

Tianxing He, Bryan McCann, Caiming Xiong, and Ehsan Hosseini-Asl. 2021. Joint energy-based model training for better calibrated natural language understanding models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1754–1761, Online. Association for Computational Linguistics.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. Quora question pairs.

Abhyuday Jagannatha and Hong Yu. 2020. Calibrating structured output predictors for natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2078–2092, Online. Association for Computational Linguistics.

Taehee Jung, Dongyeop Kang, Hua Cheng, Lucas Mentch, and Thomas Schaaf. 2020. Posterior calibrated training on sentence classification tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2723–2730, Online. Association for Computational Linguistics.

Lingkai Kong, Haoming Jiang, Yuchen Zhuang, Jie Lyu, Tuo Zhao, and Chao Zhang. 2020. Calibrated language model fine-tuning for in- and out-of-distribution data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1326–1340, Online. Association for Computational Linguistics.

Aviral Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Fei Li, Yonghao Jin, Weisong Liu, Bhanu Pratap Singh Rawat, Pengshan Cai, and Hong Yu. 2019. Fine-tuning bidirectional encoder representations from transformers (bert)–based models on large-scale electronic health record notes: An empirical study. *JMIR Med Inform*, 7(3):e14830.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in Neural Information Processing Systems*, 32:4694–4703.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Khanh Nguyen and Brendan O'Connor. 2015. Posterior calibration and exploratory analysis for natural language processing models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1587–1598, Lisbon, Portugal. Association for Computational Linguistics.

Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. 2020. Identifying mislabeled data using the area under the margin ranking. In *Advances in Neural Information Processing Systems*, volume 33, pages 17044–17056. Curran Associates, Inc.

Marek Rei and Anders Søgaard. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 293–302, New Orleans, Louisiana. Association for Computational Linguistics.

Sarah Sarabadani. 2019. Detection of adverse drug reaction mentions in tweets using ELMo. In *Proceedings of the Fourth Social Media Mining for Health Applications (#SMM4H) Workshop & Shared Task*, pages 120–122, Florence, Italy. Association for Computational Linguistics.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip Yu, and Lifang He. 2020. Mixup-transformer: Dynamic data augmentation for NLP tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3436–3440, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293.

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*, volume 32. Curran Associates, Inc.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. 2020a. On the inference calibration of neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3070–3079, Online. Association for Computational Linguistics.

Shuo Wang, Zhaopeng Tu, Shuming Shi, and Yang Liu. 2020b. On the inference calibration of neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3070–3079.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Wenpeng Yin, Huan Wang, Jin Qu, and Caiming Xiong. 2021. BatchMixup: Improving training by interpolating hidden states of the entire mini-batch. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4908–4912, Online. Association for Computational Linguistics.

Soyoung Yoon, Gyuwan Kim, and Kyumin Park. 2021. SSMix: Saliency-based span mixup for text classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3225–3234, Online. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Rongzhi Zhang, Yue Yu, and Chao Zhang. 2020. SeqMix: Augmenting active sequence labeling via sequence mixup. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8566–8579, Online. Association for Computational Linguistics.

Henghui Zhu, Ioannis C Paschalidis, and Amir M Tahmasebi. 2018. Clinical concept extraction with contextual word embedding. In *NIPS Machine Learning for Health Workshop*.