Calibrated Language Models and How to Find Them with Label Smoothing

Jerry Huang [♯ 1 2] **Peng Lu** [♯ 1] **Qiuhao Zeng** [3]

Abstract

Recent advances in natural language processing have enabled the fine-tuning of large language models (LLMs) into powerful interactive agents with improved instruction-following ability. However, this can impact confidence calibration for reliable model output, which has not been researched in full. In this work, we examine various open-sourced LLMs, where we identify significant calibration degradation after instruction tuning. Seeking a practical solution, we look towards label smoothing, which has been shown as an effective method to regularize for overconfident predictions but has yet to be widely adopted in the supervised fine-tuning (SFT) of LLMs. We provide insight into why label smoothing can maintain calibration throughout the SFT process, but identify settings remain where the effectiveness of smoothing is severely diminished. We posit the cause to stem from the ability to become overconfident, which has a direct relationship with the hidden and vocabulary size of models, which we justify theoretically and experimentally. Finally, we address an outstanding issue regarding the memory footprint of the cross-entropy loss computation with label smoothing, designing a customized kernel to dramatically reduce memory consumption without sacrificing speed or performance in comparison to existing solutions.

1. Introduction

Tremendous progress has been made in building models that follow natural language instructions (Ouyang et al., 2022; Sanh et al., 2022; Chung et al., 2024) through the use of LLMs pre-trained on large amounts of data as well as high-quality datasets that enable them to learn to interact in a human-like manner (Bach et al., 2022; Wang et al., 2022; 2023). However, such models have demonstrated a propensity for over-confidence in their predictions (Zhao et al., 2021; Jiang et al., 2021; Xiong et al., 2024), eliciting concerns over their use in more high-stakes decision-making scenarios. Such observations are not new with respect to neural networks, which have consistently been shown to suffer from over-confident predictions and over-estimate the likelihood of their correctness (Guo et al., 2017; Szegedy et al., 2016; Müller et al., 2019; Naeini et al., 2015; Minderer et al., 2021). To improve this, methods such as temperature scaling (Guo et al., 2017) and label smoothing (Müller et al., 2019) have been proposed as solutions with varying effectiveness, spurring additional work in ensuring that predictions and confidence remain matching (Lin et al., 2017; Mukhoti et al., 2020; Pereyra et al., 2017; Liu et al., 2022).

In this work, we focus on label smoothing (LS) for calibration in SFT settings. Motivated by previous works showing the effectiveness of LS for calibration in different settings, we first verify its effectiveness. We demonstrate that while it can be effective, problems begin to emerge in the case of large vocabulary LLMs (LV-LLMs). To explain this phenomenon, we attempt to establish a link between the predictive abilities of such LLMs and their size. We show that in such settings, an explicit link between the lower bound of the model entropy and the hidden size of the model causes models to fail to become overconfident (Zhao et al., 2021), negating the potential benefits and use of label smoothing. We further show how alternative techniques, such as temperature scaling and logit capping, explicitly act as a mechanism to steer models toward overconfidence, allowing the benefits of label smoothing to once again emerge.

Nevertheless, a problematic setting still remains. Growing vocabulary sizes cause large amounts of memory to be consumed to materialize the relevant logits and probabilities, making training difficult. While efficient methods (Wijmans et al., 2025; Hsu et al., 2024; PyTorch, 2024) have been proposed by implementing hardware-level optimization that significantly reduces this memory bottleneck, they often cannot support label smoothing. To address this, we identify specific optimizations that can be made in computing matrices in GPU memory, allowing for the support of label smoothing with minimal increases in memory or computational speed. Thus we introduce a new kernel that enables us to use label smoothing more efficiently as a whole.

[♯]Equal contribution [1]Université de Montréal [2]Mila - Quebec AI Institute [3]University of Western Ontario. Corresponding author: Peng Lu <peng.lu@umontreal.ca>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

To summarize our contributions, we: **1)** We point out that common SFT practices significantly degrade LLM model calibration. **2)** We demonstrate and justify why label smoothing is an appropriate approach to mitigating this concern. **3)** However, we further identify specific issues where label smoothing remains prohibitive, particularly with large vocabulary LLMs, and why existing methods fall short. **4)** We demonstrate that optimizations exist, which we incorporate in custom kernels which enable us to perform label smoothing with significant memory and throughput improvements without performance sacrifices.

2. Related Work

Model Calibration (Brier, 1950; Murphy, 1972; DeGroot & Fienberg, 1983) is a concept of matching the prediction probabilities yielded for different inputs to the expected accuracy on these inputs. In a K-way classification setting, let $\mathcal{X} \in \mathbb{R}^D$ and $\mathcal{Y} \in \{\gamma_k\}_{k=1}^K$ indicate the input and label space, respectively. Let f be a classifier and $f(\hat{y}|\boldsymbol{x}) = \hat{c}$ be the confidence of prediction, i.e., the maximum of probabilities among K dimensions corresponding to its prediction $\hat{y}$. A model is *perfectly-calibrated* when

$$P(\hat{y} = y|\hat{c} = c) = c \;\; \forall c \in [0, 1]. \tag{1}$$

Qualitatively, model calibration can be derived as $\mathbb{E}\left[|P(\hat{y} = y|\hat{c} = c) - c|\right]$. One metric that has been widely used for measuring calibration is the expected calibration error (ECE) (Naeini et al., 2015), which is a weighted average of bin-wise mis-calibration. The ECE divides the confidence score of N samples into M uniform confidence bins $\{B_m\}_{m=1}^M$ and takes the mean of the gap between accuracy (acc) and confidence (conf) over the bins weighted by the number of samples in the bins.

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} \left|\text{acc}\left(B_m\right) - \text{conf}\left(B_m\right)\right|. \tag{2}$$

Additional metrics that have been proposed include the Root Mean Square Calibration Error (RMS-CE) (Hendrycks et al., 2019), which places greater emphasis on large calibration deviations, and the Static and Adaptive Calibration Errors (SCE/ACE) (Nixon et al., 2019), which measure miscalibration over fixed and data-dependent binning schemes, respectively. These metrics offer complementary perspectives to standard calibration error measures, enabling a more comprehensive assessment of model confidence alignment.

Label Smoothing (LS) has been demonstrated to be a promising paradigm in settings to prevent models from becoming overconfident (Szegedy et al., 2016; Müller et al., 2019) or when noise exists in the provided labels (Lukasik et al., 2020; Wei et al., 2022b; Lu et al., 2023). Consider a model parameterized by $\boldsymbol{\theta}$ to model a conditional distribution $P(\cdot|\boldsymbol{x}; \boldsymbol{\theta})$, where the final operation is a $\mathrm{softmax}$. Consider the model to apply a function $f(\cdot; \boldsymbol{\theta})$ on $\boldsymbol{x}$ and $\hat{\boldsymbol{\sigma}}(\boldsymbol{x}; \boldsymbol{\theta}) \in [0, 1]^K$ to be the post-$\mathrm{softmax}$ output. Then

$$P(\gamma_i|\boldsymbol{x}; \boldsymbol{\theta}) = \hat{\boldsymbol{\sigma}}(\boldsymbol{x}; \boldsymbol{\theta})_i = \frac{\exp(\boldsymbol{\ell}(\boldsymbol{x})_i)}{\sum_{k=1}^K \exp(\boldsymbol{\ell}(\boldsymbol{x})_k)}, \tag{3}$$

where $\boldsymbol{\ell}(\boldsymbol{x}) \in \mathbb{R}^K$ is the pre-`softmax` output of the model, commonly referred to as the logits or log-probabilities. Models are usually trained by minimizing a cross-entropy (CE) loss on a dataset $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ sampled from an unknown distribution $p(\boldsymbol{x}, y)$ in order to learn the true conditional distribution $p_{y|\boldsymbol{x}}(y|\boldsymbol{x})$, computed as

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}}^{\text{CE}}(\boldsymbol{\theta}) &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \delta_{y_n}^{\gamma_k} \log P(\gamma_k|\boldsymbol{x}; \boldsymbol{\theta}) \\
&\approx -\mathbb{E}_{p(\boldsymbol{x},y)}\left[\sum_{k=1}^{K} p(\gamma_k|\boldsymbol{x}) \log P(\gamma_k|\boldsymbol{x}; \boldsymbol{\theta})\right] \\
&= -\mathbb{E}_{p(\boldsymbol{x},y)}[\text{KL}\left[\boldsymbol{\sigma}(\boldsymbol{x})\|\hat{\boldsymbol{\sigma}}(\boldsymbol{x}; \boldsymbol{\theta})\right]] + c \\
&= \mathcal{L}_{p(\boldsymbol{x},y)}^{\text{CE}}(\boldsymbol{\theta}),
\end{aligned} \tag{4}$$

where δ_i^j is the Kronecker delta with value 1 only when $i = j$. Label smoothing mixes the original distribution with a discrete uniform distribution $\mathcal{U} = [1/K]^K \in \mathbb{R}^K$ using a smoothing rate $\beta \in [0, 1]$. The loss then becomes

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}}^{\text{LS}}(\boldsymbol{\theta}) &= -\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=1}^{K}\left[(1-\beta)\delta_{y_n}^{\gamma_k} + \frac{\beta}{K}\right]\log P(\gamma_k|\boldsymbol{x}; \boldsymbol{\theta})\right] \\
&= (1-\beta)\mathcal{L}_{\mathcal{D}}^{\text{CE}}(\boldsymbol{\theta}) + \frac{\beta}{K}\sum_{i=1}^{N}\text{KL}[\boldsymbol{u}\|\hat{\boldsymbol{\sigma}}(\boldsymbol{x}_n; \boldsymbol{\theta})] + c \\
&\approx -\mathbb{E}_{p(\boldsymbol{x},y)}\left[\text{KL}\left[(1-\beta)\boldsymbol{\sigma}(\boldsymbol{x}) + \beta\boldsymbol{u}\|\hat{\boldsymbol{\sigma}}(\boldsymbol{x}; \boldsymbol{\theta})\right]\right] + c \\
&= \mathcal{L}_{p(\boldsymbol{x},y)}^{\text{LS}}(\boldsymbol{\theta}).
\end{aligned} \tag{5}$$

Thus, label smoothing can be understood to a regularization term that encourages a uniform distribution over the output labels, hence preventing it from over-fitting to the training data and encouraging a model to be less confident on all samples by smoothing the true conditional being learned.

3. Smoothing and Calibration in LLMs

Preliminaries. Define an auto-regressive LLM to be parameterized by parameters $\boldsymbol{\theta}$. A model represents an embedding function $g(\cdot; \boldsymbol{\theta}_e) : \mathbb{R}^N \to \mathbb{R}^{D\times N}$ where L is the length of a discrete input sequence $\boldsymbol{x}$ and D is the hidden size of the model that produces an embedding $\boldsymbol{E} \in \mathbb{R}^{D\times N}$. This is followed by a classifier $\boldsymbol{C}(\boldsymbol{\theta}_c) \in \mathbb{R}^{D\times|\boldsymbol{V}|}$ and a $\mathrm{softmax}$ operation to produce a probability distribution over $\boldsymbol{V}$.

A common practice is to tune $\boldsymbol{\theta} = [\boldsymbol{\theta}_e, \boldsymbol{\theta}_c]$ on a dataset through supervised fine-tuning (SFT), or instruction tuning (Wei et al., 2022a; Ouyang et al., 2022; Chung et al., 2024; Dubois et al., 2023). During SFT of an LLM, an input sequence $\boldsymbol{x}$ of length L consists of a sequence of discrete vocabulary tokens $\boldsymbol{x}_i \in \boldsymbol{V}\ \forall i \in [N]$. The first m tokens in $\boldsymbol{x}$ consist of the instruction, while the rest is considered the target output. The goal of SFT is to minimize a CE objective over the output portion of the sequence $\boldsymbol{x}_{m+1:N}$, computed as a loss over the individual elements of the sequence

$$\mathcal{L}_{\boldsymbol{x}}(\boldsymbol{\theta}) = \sum_{j,v} \delta^v_{\boldsymbol{x}_j} \log\left(\boldsymbol{C}(\boldsymbol{\theta}_c)^\top g(v|\boldsymbol{x}_{1:j-1}; \boldsymbol{\theta}_e)\right). \quad (6)$$

In essence, the learning problem is a $|\boldsymbol{V}|$-class classification problem for each element in the target portion of the input sequence, where the prediction for any specific position is influenced by all previous elements in the sequence.

This procedure transforms the original parameter set $\boldsymbol{\theta}$ into a new set, denoted $\boldsymbol{\theta}_{\text{SFT}}$, which often enables the model to follow human-provided instructions with remarkable accuracy and fluency. However, this fine-tuning process can also lead to a deterioration in the model's calibration, as illustrated in Figure 1, potentially reducing its reliability in estimating uncertainty or confidence.

Why does Instruction Tuning Lead to Mis-calibration? Our results raise a question: *Why does instruction tuning degrade calibration?* To better understand this, SFT can be viewed through the lens of out-of-distribution (OOD) generalization and calibration error. In particular, we assume that the SFT data consists of an in-distribution (ID) dataset whereas the downstream dataset on which generalization and calibration are tested constitutes an OOD dataset.

Suppose we have an ideal parameter set $\boldsymbol{\theta}^*$ that minimizes calibration error on an unknown dataset $\mathcal{D}$, defined as $\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\|f(\boldsymbol{x};\boldsymbol{\theta}) - c(\boldsymbol{x})\|_2^2]$ where $c(\boldsymbol{x}) = \mathbb{E}_{y\sim f(\boldsymbol{x};\boldsymbol{\theta})}[y]$ is the expected label given a prediction $f(\boldsymbol{x};\boldsymbol{\theta})$. In other words, $f(\cdot;\boldsymbol{\theta})$ always produces the calibrated prediction for the label given an input $\boldsymbol{x}$ such that the output confidence matches the expected label over the subset of samples with the same confidence value. Thus the goal of SFT is to learn a set of parameters $\boldsymbol{\theta}^*$ that outputs reliable prediction probability on samples from both an unseen OOD domain, which is defined by a distribution $p_{\text{OOD}}(\boldsymbol{x}, y)$, as well as the ID distribution $p_{\text{ID}}(\boldsymbol{x}, y)$. Thus the goal is to minimize

$$\mathcal{L}_p(\boldsymbol{\theta}) = \mathbb{E}_p[\|f(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x};\boldsymbol{\theta}^*)\|_2^2], \quad (7)$$

for $p \in \{p_{\text{OOD}}(\boldsymbol{x}, y), p_{\text{ID}}(\boldsymbol{x}, y)\}$.

Oh et al. (2024) demonstrate that under such a setting, simultaneously maintaining accuracy and calibration of the final classifier (in the case of an auto-regressive LLM, this is the language modeling head) has a direct relationship to the diversity of the feature embeddings as follows:

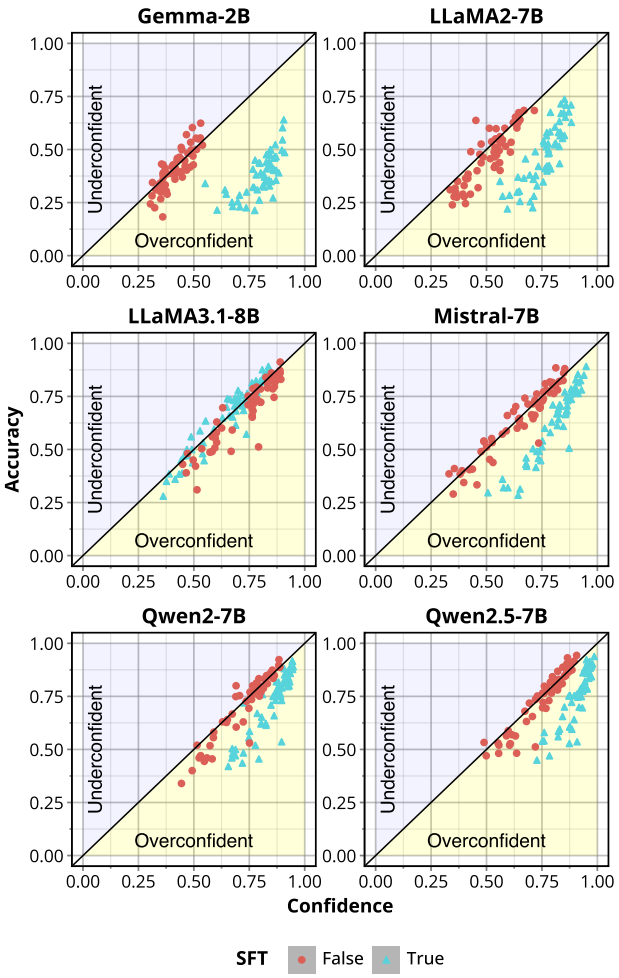

Figure 1. Reliability diagrams of open-sourced pre-trained models with (red) and without instruction-tuning (blue) on the MMLU dataset (Hendrycks et al., 2021). The horizontal axis represents the model's confidence in each answer choice for each question, while the vertical axis shows the accuracy on each question. The solid diagonal indicates perfect calibration, separating areas where predictions are deemed over-confident or under-confident. Instruction-tuning visibly leads to over-confidence, regardless of the instruction-tuning dataset (which differs between models).

Lemma 3.1 (Oh et al. 2024)**.** *Let $f(\cdot;\boldsymbol{\theta}) : \mathcal{X} \to [0,1]^K$ be a real-valued function of the form $f(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{i=1}^d f_i(\boldsymbol{x}[i];\boldsymbol{\theta})$ where $f_i(\cdot;\boldsymbol{\theta})$ is an arbitrary one-dimensional function, and f is in a hypothesis class $\mathcal{F}$ that has pseudo dimension $\mathcal{P}_{dim}(\mathcal{F}) = d_f$. Let $\mathcal{D}_{ID} = \left\{\left(\boldsymbol{x}_{ID}^{(n)}, y_{ID}^{(n)}\right)\right\}_{n=1}^N$ be a dataset sampled from the ID distribution. If $(\boldsymbol{x}[1], \ldots, \boldsymbol{x}[d])$ have matching marginals for ID and OOD, and $(\boldsymbol{x}[i], \boldsymbol{x}[j])$ is a bi-variate Gaussian for*

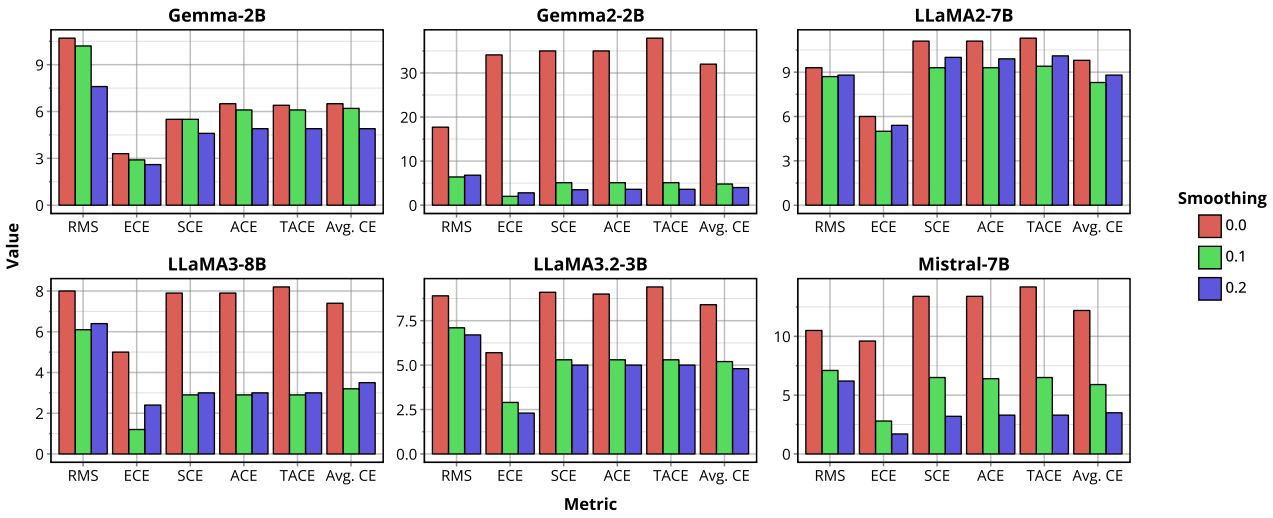

Figure 2. Effects of instruction-tuning on calibration, presented under a number of different calibration error metrics (where lower is better). Values can range from 0 to 100. Models are all fine-tuned on a `Tulu3` (Wang et al., 2023) SFT dataset and evaluated on MMLU. We can observe that across all models, which have various structural differences, the use of label smoothing is capable of reducing calibration error while having negligible effects on downstream performance accuracy on the task.

every $i, j \in [d]$, then for any $\delta \in (0,1)$ and for all f, the following bounds hold with probability at least $1-\delta$:

$$\mathcal{L}_{p_{OOD}}(\boldsymbol{\theta}) \leq \mathcal{L}_{\mathcal{D}_{ID}}(\boldsymbol{\theta}) + \frac{d}{\sigma_{min}\left(\tilde{\Sigma}_{p_{ID}(\boldsymbol{x})}\right)} + \Delta + \mathcal{O}\left(\sqrt{\log\left(\frac{N}{d_h}\right)^{d_h}\left(\frac{1}{N\delta}\right)}\right), \tag{8}$$

$$\begin{aligned}&\mathbb{E}_{p_{OOD}(\boldsymbol{x},y)}\left[\|f(\boldsymbol{x};\boldsymbol{\theta}) - y\|_2^2\right] + \mathbb{E}_{p_{OOD}(\boldsymbol{x},y)}\left[\|c(\boldsymbol{x})\|_2^2\right] - 1 \\ \leq &\mathcal{L}_{\mathcal{D}_{ID}}(\boldsymbol{\theta}) + \frac{d}{\sigma_{min}\left(\tilde{\Sigma}_{p_{ID}(\boldsymbol{x})}\right)} + \Delta \\ &+ \mathcal{O}\left(\sqrt{\log\left(\frac{N}{d_h}\right)^{d_h}\left(\frac{1}{N\delta}\right)}\right),\end{aligned} \tag{9}$$

where

$$\begin{aligned}\theta^* &= \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\ \mathcal{L}_{p_{OOD}(\boldsymbol{x},y)}(\boldsymbol{\theta}) + \mathcal{L}_{p_{ID}(\boldsymbol{x},y)}(\boldsymbol{\theta}), \\ \Delta &= \mathcal{L}_{p_{OOD}(\boldsymbol{x},y)}(\boldsymbol{\theta}^*) + \mathcal{L}_{p_{ID}(\boldsymbol{x},y)}(\boldsymbol{\theta}^*),\end{aligned} \tag{10}$$

and $\tilde{\Sigma}_{p_{ID}(\boldsymbol{x})} = \mathbb{E}_{p_{ID}(\boldsymbol{x})}[\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^\top]$ is a covariance matrix with a strictly positive minimum singular value of d-dimensional normalized input $\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}[1], ..., \tilde{\boldsymbol{x}}[d])$, where $\tilde{\boldsymbol{x}}[i] = (\boldsymbol{x}[i] - \mathbb{E}[\boldsymbol{x}[i]])\sqrt{Var(\boldsymbol{x}[i])}$ and $\sigma_{min}(M)$ is the smallest singular value of a matrix $M \in \mathbb{R}^{d_1 \times d_2}$.

The dependence of the bound on the minimal singular value of the covariance matrix indicates that as the set of learnt feature embeddings (the embedding of the context in this scenario) becomes less mutually dependent, both calibration error and classification error can be minimized. However, prior works have shown that fine-tuning can significantly reduce the diversity of such features (Mukhoti et al., 2024; Kumar et al., 2022; Huh et al., 2024), justifying why standard SFT can significantly degrade calibration (Figure 2).

The Effects of Label Smoothing. To understand the effects of label smoothing from a model calibration perspective, we adopt an optimization-based viewpoint to analyze the implicit constraints imposed by the regularization (Bertsekas, 1999). This perspective reveals how label smoothing modifies the solution space of the model's predictions, encouraging not only smoother output distributions but also a reduction in the model's confidence on any single class. First, define

Definition 3.2. The **logit distance** vector for $\boldsymbol{x}$, $\boldsymbol{d}(\boldsymbol{x})$, is

$$\boldsymbol{d}(\boldsymbol{x}) = \left[\max_{1\leq i\leq K} \boldsymbol{\ell}(\boldsymbol{x})_i - \boldsymbol{\ell}(\boldsymbol{x})_k\right]_{k=1}^{K} \in \mathbb{R}^K. \tag{11}$$

One way of ensuring that a model does not over-estimate a specific class is to enforce this as a hard constraint, which results in equal logits among all classes and a $\mathrm{softmax}$ output of $\boldsymbol{o} = f(\boldsymbol{x};\boldsymbol{\theta}) = [1/K]^K$. As such, it is often preferable to enforce this as a soft-penalty function $\mathcal{P} : \mathbb{R}^K \to \mathbb{R}$ into the objective function minimized during training. Recalling Equation (5), we can relate this soft-penalty to the additional KL-divergence introduced by the label smoothing objective.

Proposition 3.3. *A linear penalty (or a Lagrangian term) for the hard constraint $\boldsymbol{d}(\boldsymbol{x}) = \mathbf{0}$ is bounded from above and below by* $\mathrm{KL}\left(\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right)$*, up to additive constants*

$$\mathrm{KL}[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)] - \log K \leq \sum_{i=1}^{K} \frac{\boldsymbol{d}\left(\boldsymbol{x}\right)_i}{K} \leq \mathrm{KL}\left[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right]. \tag{12}$$

The proof (in Appendix A.1) indicates that label smoothing approximately minimizes, for a linear penalty, the constraint $\boldsymbol{d}(\boldsymbol{x}) = \mathbf{0}$, encouraging equality among the logits for each class to ensure that over-confidence is penalized. More importantly, however, it is noted that

Proposition 3.4. *Define a likelihood model* $p\left(y|\boldsymbol{x}; \boldsymbol{\theta}\right) = \mathrm{Cat}\left(\mathrm{softmax}\left(f\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right)\right)$*, a categorical distribution with parameters* $\boldsymbol{z} = \mathrm{softmax}\left(f\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right) \in \Delta(\Theta)$ *where* $\Delta\left(\Theta\right)$ *denotes a probability simplex over the parameter space* Θ*. The label smoothing objective is equivalent to Maximum A Posteriori (MAP) estimation on the* softmax *probability vector under the independence assumption* $p\left(\boldsymbol{z}|\boldsymbol{x}\right) = p\left(\boldsymbol{z}\right)$*.*

A proof is provided in Appendix A.2. This MAP formulation above relies on the provided label y for each sample $\boldsymbol{x}$, without exploiting the potential similarities among different samples in the empirical training dataset for more accurate estimation. Łukasz Rajkowski (2019) showed that using MAP estimation can lead to greater separability and diversity of individual samples, under the assumption that the sample is sampled from a normal distribution. Chi et al. (2024) further prove this to be the case with Transformer-based language models. In conjunction with prior claims from Oh et al. (2024), this proposition indicates that label smoothing can in fact learn more diverse input features, further explaining the improvement in calibration.

4. Label Smoothing for Large Vocabularies

We conduct SFT training with and without LS on a Tulu3 dataset (Wang et al., 2023) for different pre-trained language model families, including Llama (Grattafiori et al., 2024), Gemma (Gemma Team, 2024) and Mistral (Jiang et al., 2023). While we can note the usefulness of label smoothing for model calibration shown in Figure 2, it becomes clear that its effectiveness is much less visible in some cases. Take for instance three `LLaMA3` models of sizes `1B`, `3B` and `8B`, which we fine-tune on the same instruction dataset (Figure 3). While label smoothing shows an improvement in calibration for the `8B`-sized model, this diminishes significantly to the `3B` model and `1B` model, both in the baseline model (where no SFT procedure has been performed) as well as SFT with multiple different datasets. We therefore seek to investigate the underlying causes of our empirical findings, aiming to shed light on a previously unexplained phenomenon.

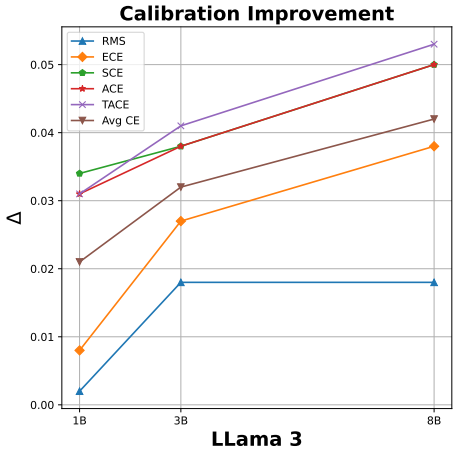

Figure 3. Calibration of different `LLaMA3` models fine-tuned on the same SFT dataset. As the size of the model decreases, the calibration of the model sees less improvement from the use of LS.

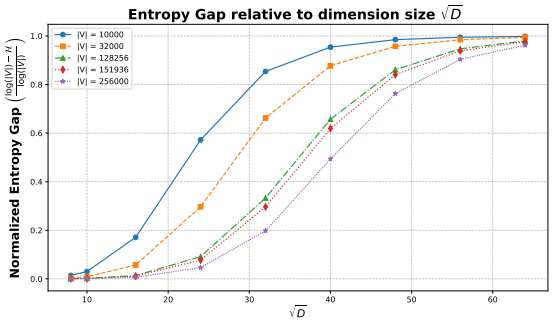

Figure 4. Relative entropy bound for different LLM vocabulary sizes with varying hidden sizes (D). Our visualization shows the normalized entropy gap for varying hidden sizes of the LM head. This gap is calculated by taking the difference between the entropy upper and lower bounds and dividing by the upper bound ($\log|\boldsymbol{V}|$). A lower ratio indicates the model is restricted to producing concentrated predictions.

We study the concentration behavior of the LM head by analyzing its relationship between entropy and model size, which provides an explanation of diminishing returns of label smoothing. We start by providing the following lemma, which establishes a bound on the LLM logits prior to the language modeling head.

Lemma 4.1. *Let* $\boldsymbol{C} \in \mathbb{R}^{D \times |\boldsymbol{V}|}$ *be a matrix with* $\|\boldsymbol{C}\|_2 = \sigma_C$*, and let* $\boldsymbol{h} \in \mathbb{R}^D$ *be a vector such that each entry of* $\boldsymbol{h}$ *satisfies* $|h_i| \leq \sigma_h$*. The upper bound for* $\left\|\boldsymbol{C}^\top \boldsymbol{h}\right\|_2$ *is:*

$$\left\|\boldsymbol{C}^\top \boldsymbol{h}\right\|_2 \leq \sigma_C \cdot \sigma_h \cdot \sqrt{D}. \tag{13}$$

Thus, the norm of the final logit vector is also bounded and depends on the dimension dimension, which enables an entropy-based analysis of the LM head's predictions as a function of both the vocabulary size $|\boldsymbol{V}|$ and the embedding dimension D.

Theorem 4.2. *(LM head Entropy Lower Bound). Let* $\rho = \sigma_C \sigma_h$, $\boldsymbol{u} = \boldsymbol{C}^\top \boldsymbol{h}$ *and* $\gamma = \exp\left(-\rho\sqrt{\frac{D|\boldsymbol{V}|}{|\boldsymbol{V}|-1}}\right)$, *then the entropy* $\mathcal{H}_{\boldsymbol{u}}$ *of prediction of the LM head holds that:*

$$\mathcal{H}_{\boldsymbol{u}} \geq \log\left(1 + (|\boldsymbol{V}| - 1)\,\gamma\right) + \frac{\rho \cdot \gamma\sqrt{D\,|\boldsymbol{V}|\,(|\boldsymbol{V}| - 1)}}{1 + (|\boldsymbol{V}| - 1)\,\gamma}. \quad (14)$$

Intuitively, the above lemma and theorem (proofs in Appendix A.3) indicates that for large $|\boldsymbol{V}|$, the minimum entropy is lower bounded by $\Omega\left(|\boldsymbol{V}|\, e^{-\rho\sqrt{D}}\right)$, thus increasing linearly with the vocabulary size $|\boldsymbol{V}|$ and decreasing exponentially with $\sqrt{D}$. We illustrate how this changes in Figure 4, where the normalized entropy gap for different $|\boldsymbol{V}|$ is shown. Given the same $|\boldsymbol{V}|$, the concentration behavior of the LM head is primarily influenced by the size of the hidden dimension. As the hidden size increases, the model is increasingly capable of attaining a lower entropy, while the bound is smaller for larger $|\boldsymbol{V}|$ at the same D, highlighting why large vocabulary LLMs at smaller sizes are less prone to overconfidence during tuning.

Remark 4.3. Models with smaller D with large vocabulary size $|\boldsymbol{V}|$ suffer from a lack of concentration ability due to their limited hidden size. By consequence, label smoothing cannot help with calibration, as it serves to only penalize overconfidence while having no specific benefits for under-confidence due to mixing with the maximum entropy (uniform) distribution.

However, we note that from this analysis, the entropy bound is influenced by $|\boldsymbol{V}|$ and D through the distribution of $\boldsymbol{h}$. However, as $\boldsymbol{h}$ are the logits of the model, these can be manipulated before the $\mathrm{softmax}$ directly. Thus it follows that under-confidence in models can be attained through manipulation of the $\left\|\boldsymbol{C}^\top \boldsymbol{h}\right\|$:

Remark 4.4. Fix $|\boldsymbol{V}|$ and D. Using a temperature $\tau < 1$ will modify ρ, leading to a decreased lower bound.

We note that various methods can serve to manipulate this bound. The above remark notes the effects of temperature scaling, which directly uses a constant temperature T to scale the logits before the $\mathrm{softmax}$. This enables the manipulation of σ_h in Lemma 4.1 without changing any additional values, thereby increasing or decreasing the entropy bound based on the choice of T. This explains how temperature scaling can enable better model calibration, especially through the choice of a $T > 1$ used to divide the logits, thereby decreasing σ_h which has the downstream effect of increasing the minimum entropy bound of the model. This can (for smaller label class sizes) significantly improve calibration, as the initial model has the potential for over-confidence which temperature can serve to mitigate. Similarly, the use of logit softcapping, as in the `Gemma2` family of models, applies a similar change in σ_h, reducing the lower bound and thereby enabling models of smaller size to become over-confident.

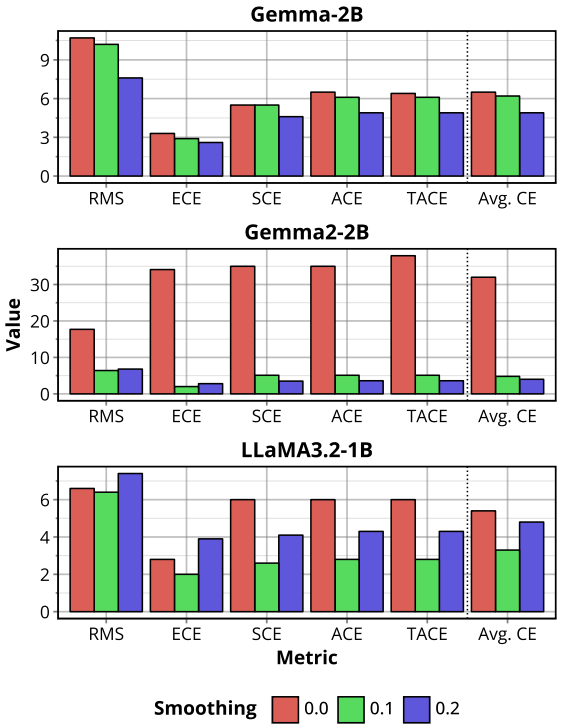

Figure 5. Effect of label smoothing on large vocabulary models with a smaller hidden size (2048). `Gemma-2B` observes a smaller change compared to `LLaMA3.2-1B`, due to having the largest vocabulary size. However, `Gemma2-2B` observes a large change in part thanks to the softcapping of logits.

5. Efficient Smoothed Cross-Entropy

Despite our analysis on the effectiveness of label smoothing for calibrating LLMs, areas of concern still exist regarding practical applicability. In this section, we introduce such limitations and how the label smoothing setting is distinct from existing solutions that attempt to mitigate them.

When the vocabulary $\boldsymbol{V}$ is large, the final cross-entropy begins to consume a significant amount of memory, as a $N \times |\boldsymbol{V}|$ matrix must be materialized to compute the loss. While optimizations have been implemented to manage other model components (Dao et al., 2022; Dao, 2024), this logit matrix begins to constitute the vast majority of this footprint when the vocabulary is large (Grattafiori et al., 2024; Gemma Team, 2024; Yang et al., 2024). In this section, we introduce an efficient smoothed cross-entropy computation, with its core implementation and results that demonstrate its effectiveness.

5.1. Limitations of Existing Solutions

Efficient implementation of cross-entropy computation (Wijmans et al., 2025; Hsu et al., 2024; PyTorch, 2024) often

makes use of the fact that for a non-smoothed loss, computing the entire logit matrix is unnecessary. Instead, only the relevant row from the classifier head $\boldsymbol{C}$ is needed, as this is the only raw logit value that is relevant to the loss. Such approaches therefore store into an output only this single logit, reducing overhead by a factor of nearly $\frac{1}{|\boldsymbol{V}|}$. However, this is not feasible for smoothed losses. Reusing notation from Section 3, consider an LS loss on a sequence

$$\mathcal{L}_{\boldsymbol{x}}^{\text{LS}} = \sum_{i=1}^{N} \mathcal{L}_{\boldsymbol{x}_i}^{\text{LS}} = \sum_{i=1}^{N} \Bigg[\underbrace{(1-\beta)\, \boldsymbol{C}_{\boldsymbol{x}_i}^{\top} \boldsymbol{E}_i}_{\text{(1)Target Loss}} + \underbrace{\frac{\beta}{|\boldsymbol{V}|} \sum_{v \in \boldsymbol{V}} \boldsymbol{C}_v^{\top} \boldsymbol{E}_i}_{\text{(2)Smoothing Loss}} - \underbrace{\log \sum_{v \in \boldsymbol{V}} \exp\left(\boldsymbol{C}_v^{\top} \boldsymbol{E}_i\right)}_{\text{(3)LSE}} \Bigg], \tag{15}$$

(2) in the above equation indicates that it must be the case that all logits need to be explicitly added to the loss, leaving existing solutions unfeasible for this specific setting. However, we demonstrate that we can indeed compute these in a manner that limits memory consumption usage, without influencing the throughput and applicability of the proposed solution to other scenarios such as no label smoothing.

5.1.1. Forward Pass

To naively compute a CE loss as in Equation (15), one could individually compute each loss component. However, we notice that many of these computations can be reused, such as the computation of $\boldsymbol{C}_v^{\top} \boldsymbol{E}_i$ in both (2) and (3). As such, for more efficient computation, we can compute all components in parallel in on-chip shared memory (SRAM) while making the best use of the GPU cache structure.

5.2. Implementation

Our forward pass (Algorithm 1) fuses the computation of all three components of Equation (15) to minimize memory and indexing costs, rendering the computation efficient on modern GPUs (Kerr et al., 2017). First, the embeddings $\boldsymbol{E}$ and classifier $\boldsymbol{C}$ are divided into chunks $\boldsymbol{E}_n$ of $\boldsymbol{E}$ with size $D \times N_B$ and $\boldsymbol{C}_m$ of $\boldsymbol{C}$ with size $D \times V_B$, operated on independently. The standard output $\boldsymbol{O} = \boldsymbol{C}^{\top} \boldsymbol{E} \in \mathbb{R}^{|\boldsymbol{V}| \times N}$ is divided into blocks of size $V_B \times N_B$ which store the products $\boldsymbol{O}_{nv} = \boldsymbol{C}_m^{\top} \boldsymbol{E}_n$. $\boldsymbol{C}_m$ and $\boldsymbol{E}_n$ are further split along the D dimension into d chunks of size D_B. Thus chunks $\boldsymbol{E}_{n,d}$ and $\boldsymbol{C}_{m,d}$ of size $D_B \times N_B$ and $D_B \times V_B$ can be used to accumulate $\boldsymbol{O}_{nv} = \sum_d \boldsymbol{C}_{m,d}^{\top} \boldsymbol{E}_{n,d}$ directly in SRAM before being written into global memory.

To compute (3) or the LSE, the above strategy is sufficient by having each block first compute a matrix multiplication, then the log-sum-exp along the vocabulary dimension m for its block, and finally update the LSE with this result. We make use of a trick introduced by Wijmans et al. (2025),

Algorithm 1 Memory-efficient forward pass

Inputs: $\boldsymbol{E} \in \mathbb{R}^{D \times N}, \boldsymbol{C} \in \mathbb{R}^{D \times |\boldsymbol{V}|}, \boldsymbol{x} \in \mathbb{R}^{N}$.
Block sizes N_B, V_B, and D_B.
Smoothing $\beta \in [0, 1]$.
Outputs: $\text{LSE} = \log \sum_v \exp(\boldsymbol{C}_v^{\top} \boldsymbol{E}) \in \mathbb{R}^{N}$.
$\boldsymbol{o} = (1-\beta)\left(\boldsymbol{C}^{\top} \boldsymbol{E}_i\right)_{\boldsymbol{x}} + \frac{\beta}{|\boldsymbol{V}|} \sum_v \boldsymbol{C}_v^{\top} \boldsymbol{E} \in \mathbb{R}^{N}$.

```
LSE, o = −∞_N, 0_N
for all pairs of blocks E_n, C_v, x_n do
    A_nv = 0_{V_B × N_B}
    for blocks E_{n,d}, C_{v,d} do
        A_nv += C_{v,d}^⊤ E_{n,d}
        M = (C_{x_n,d} == C_{v,d})
        if M ≠ 0_{N_B × V_B} then
            o_n += (1 − β) · Σ [ (C_{m,d}^⊤ E_{n,d})^⊤ ⊙ M ]
        end if
    end for
    LSE_nv = log Σ exp(A_nv^⊤)
    LSE_n = log(exp(LSE_n) + exp(LSE_nv))
    if β ≠ 0 then
        o_n += (β/|V|) Σ A_nv^⊤
    end if
end for
```

where blocks along the same N dimension range but different $|\boldsymbol{V}|$ dimension range are written in the same location, in order to reduce memory usage on SRAM. This is implemented directly using a lock mechanism, where blocks exchange a single lock per group and update the LSE online.

Similar to the above, if the label smoothing parameter β is not 0 (indicating that smoothing is used), (2) in Equation (15) can be computed through reuse of intermediate results for (3), by summing over the vocabulary dimension m of $\boldsymbol{O}_{nm}$ and updating an output $\boldsymbol{o} \in \mathbb{R}^{N}$ with the result.

To understand how to fuse the computation of (1) into this matrix multiplication, we can first consider that the input sequence $\boldsymbol{x}$ can also be split into chunks $\boldsymbol{x}_n$ of size N_B. Because $\boldsymbol{x}_n$ contains information about the target labels, $\boldsymbol{x}_n$ can be directly used to compute the memory addresses of the target classifier chunk $\boldsymbol{C}_{\boldsymbol{x}_i,d}$. Because content from an address for each $\boldsymbol{C}_{m,d}$ must be loaded into SRAM to compute $\boldsymbol{O}_{nv}$, this means that a direct address comparison can be used to create a mask $\boldsymbol{M} \in \mathbb{R}^{N_B \times V_B}$, which has value 1 only where the rows in $\boldsymbol{C}_{\boldsymbol{x}_i,d}$ matches $\boldsymbol{C}_{m,d}$ and 0 otherwise. This enables us to add (1) to $\boldsymbol{o}$ by only adding the label-corresponding rows to the loss

$$\boldsymbol{o}_n \mathrel{+}= (1-\beta) \sum_m \left[\left(\boldsymbol{E}_{n,d}^{\top} \boldsymbol{C}_{m,d} \right) \odot \boldsymbol{M} \right], \tag{16}$$

where $\odot$ is the element-wise matrix multiplication.

Thus for inference, we can directly compute the entire loss

Table 1. Results of different models w/ or w/o LS on different datasets. All models are performed with a 5-shot evaluation. We report the reliability of models with expected calibration error (ECE) and root mean square calibration error (RMS).

SFT Dataset	Model	MMLU			HELLASWAG			ARC-EASY		
		Acc. ↑	ECE ↓	RMS ↓	Acc. ↑	ECE ↓	RMS ↓	Acc. ↑	ECE ↓	RMS ↓
Alpaca	Mistral-7B + SFT ($\beta = 0$)	0.579	0.134	0.120	0.302	0.127	0.160	0.803	0.099	0.154
	Mistral-7B + SFT ($\beta = 0.1$)	0.590	**0.094**	**0.104**	0.304	**0.087**	**0.124**	0.806	**0.071**	**0.131**
	LLaMA3-8B + SFT ($\beta = 0$)	0.638	0.113	0.113	0.375	0.162	0.085	0.863	0.070	0.127
	LLaMA3-8B + SFT ($\beta = 0.1$)	0.636	**0.073**	**0.094**	0.374	**0.087**	**0.037**	0.864	**0.037**	**0.098**
	Gemma2-2B + SFT ($\beta = 0$)	0.528	0.343	0.180	0.302	0.127	0.160	0.773	0.131	0.174
	Gemma2-2B + SFT ($\beta = 0.1$)	0.532	**0.125**	**0.121**	0.304	**0.087**	**0.124**	0.764	**0.069**	**0.127**
Tulu3Mixture	Mistral-7B + SFT ($\beta = 0$)	0.600	0.096	0.105	0.369	0.044	0.085	0.843	0.078	0.135
	Mistral-7B + SFT ($\beta = 0.1$)	0.603	**0.028**	**0.071**	0.375	**0.021**	**0.067**	0.840	**0.030**	**0.094**
	LLaMA3-8B + SFT ($\beta = 0$)	0.651	0.050	0.080	0.361	0.049	0.091	0.857	0.058	0.114
	LLaMA3-8B + SFT ($\beta = 0.1$)	0.646	**0.012**	**0.061**	0.356	**0.025**	**0.064**	0.858	**0.035**	**0.097**
	Gemma2-2B + SFT ($\beta = 0$)	0.533	0.341	0.177	0.273	0.082	0.128	0.758	0.086	0.142
	Gemma2-2B + SFT ($\beta = 0.1$)	0.531	**0.020**	**0.064**	0.271	**0.041**	**0.087**	0.755	**0.029**	**0.101**
OpenHermes	Mistral-7B + SFT ($\beta = 0$)	0.602	0.071	0.094	0.546	0.041	0.071	0.867	0.066	0.100
	Mistral-7B + SFT ($\beta = 0.1$)	0.602	**0.014**	**0.059**	0.552	**0.021**	**0.042**	0.857	**0.036**	**0.076**
	LLaMA3-8B + SFT ($\beta = 0$)	0.654	0.038	0.077	0.552	0.063	0.074	0.880	0.065	0.112
	LLaMA3-8B + SFT ($\beta = 0.1$)	0.646	**0.016**	**0.059**	0.554	**0.038**	**0.037**	0.880	**0.041**	**0.089**
	Gemma2-2B + SFT ($\beta = 0$)	0.541	0.353	0.180	0.364	0.125	0.143	0.816	0.131	0.175
	Gemma2-2B + SFT ($\beta = 0.1$)	0.542	**0.016**	**0.063**	0.362	**0.077**	**0.096**	0.813	**0.038**	**0.096**

efficiently without directly needing to materialize the complete logit matrix in global memory.

5.2.1. BACKWARD PASS

To implement a backward pass, borrowing logic from Wijmans et al. (2025) is sufficient to produce a backward pass that is significantly faster than existing alternatives. As Wijmans et al. (2025) do not consider label smoothing, their backward implementation requires some modification. As the gradient with respect to LSE is

$$\begin{aligned}\nabla^{\text{LSE}}_{\mathcal{L}^{\text{LS}}} \boldsymbol{E} &= \left((\texttt{softmax}(\boldsymbol{C}^\top \boldsymbol{E}) \cdot \nabla_{\mathcal{L}^{\text{LS}}} \text{LSE}) \boldsymbol{C}\right)^\top, \\ \nabla^{\text{LSE}}_{\mathcal{L}^{\text{LS}}} \boldsymbol{C} &= \left((\texttt{softmax}(\boldsymbol{C}^\top \boldsymbol{E}) \cdot \nabla_{\mathcal{L}^{\text{LS}}} \text{LSE})^\top \boldsymbol{E}\right)^\top,\end{aligned} \tag{17}$$

incorporating the derivative with respect to (1)/(2) only requires adding a constant to $\texttt{softmax}(\boldsymbol{C}^\top \boldsymbol{E})$ depending on whether the element is the true label or not.

While the forward and backward pass can be further fused to compute both the output and gradient simultaneously, this requires computing LSE_n before any `softmax` value $\boldsymbol{S}_{nv}$, otherwise the computed $\boldsymbol{S}_{nv}$ will use an incomplete log-sum-exp scaling factor (each row $\boldsymbol{S}_n$ depends on LSE_n to have been computed in its entirety). This thus requires a barrier to block subsequent execution code until all preceding computations have been completed by all workers. While can save memory, we observe that specific tricks such as average logit sorting and gradient filtering, introduced by Wijmans et al. (2025), enable faster computation through standalone forward and backward passes with minimal increased memory utilization (2MB overhead).

5.3. Experiments

Setup. For all models trained on the `Alpaca` (Dubois et al., 2023), `Tulu3Mixture` (Wang et al., 2023) and `OpenHermes` (Teknium, 2023) SFT datasets, we adhere to the recommended training configuration outlined by Wang et al. (2023). We employ the AdamW optimizer for training and conduct a grid search over the learning rates $\{\texttt{5e}-\texttt{6}, \texttt{2e}-\texttt{5}, \texttt{5e}-\texttt{5}, \texttt{2e}-\texttt{4}\}$ to determine the optimal setting for each model. To facilitate stable training and prevent over-fitting, we use a batch size of 128 and apply a dropout rate of 0.1.

Calibration Results. Table 1 provides a comprehensive comparison of the accuracy and calibration performance of various large language models (LLMs) with and without label smoothing (LS) across different supervised fine-tuning (SFT) datasets. The evaluation is conducted on three widely used benchmark datasets: MMLU, HellaSwag, and ARC-Easy, ensuring a robust assessment of model performance. Across all experimental settings, applying LS with a smoothing factor of $\beta = 0.1$ consistently leads to improved calibration, as indicated by lower Expected Calibration Error (ECE) and Root Mean Square (RMS) calibration error, while preserving model accuracy. Notably, `LLaMA3-8B` and `Mistral-7B` achieve the best calibration results, particularly when trained on `OpenHermes` and `Tulu3Mixture`. These findings underscore the effectiveness of LS as a simple yet powerful technique for enhancing model reliability without sacrificing predictive accuracy.

Table 2. Memory and time to compute losses and gradients. Results are computed on a batch size of 8192 tokens in a single sequence, generated from a `Gemma2-2B` model (vocabulary size of 256K and hidden size 2304). Experiments are conducted using `PyTorch` 2.4.0 and `CUDA` 12.1. Further see Table 3 in Appendix B.

Method	`fwd`		`bwd`		`fwd+bwd`	
	Memory	Time	Memory	Time	Memory	Time
Smoothing $\beta > 0$						
Ours	1.1 MB	24.2 ms	1,163 MB	49.3 ms	1,164 MB	72.9 ms
`torch.compile`	4,000 MB	22.8 ms	12,000 MB	38.3 ms	16,000 MB	62.3 ms
Baseline	24,000 MB	41.4 ms	16,000 MB	62.5 ms	28,000 MB	104.9 ms
Smoothing $\beta = 0$						
Ours	1.1 MB	24.0 ms	1,163 MB	49.2 ms	1,164 MB	72.9 ms
Cut-Cross Entropy[1]	1.1 MB	23.6 ms	1,163 MB	49.2 ms	1,164 MB	72.4 ms
`torch.compile`	4,000 MB	20.6 ms	4,000 MB	33.9 ms	8,000 MB	55.0 ms
Baseline	24,000 MB	38.7 ms	16,000 MB	55.8 ms	28,000 MB	96.0 ms

5.4. Benchmarking and Testing

Benchmarking. Table 2/3 provides the primary comparison of our custom kernel against existing alternatives, both with and without label smoothing. We compare in terms of both memory and time. First, in terms of memory, measured as the peak amount of GPU storage necessary for the computation, our kernels surpass other options, beating the next closest competitor (Liger Kernels) by requiring over 75% less memory. Time-wise, we trail a compiled `torch` implementation by less than 10 ms while utilizing only ≈7% the amount of memory as they use. Furthermore, our method can be applied in more general settings. Under such circumstances, we have two observations. First, we can match Cut-Cross-Entropy (CCE) with a very minimal increase in computation speed (less than 0.5% overhead for a forward pass and less than 1% for a combined forward/backward). Furthermore, while an efficient compiled `torch` implementation remains slightly more efficient, label smoothing causes an increase in running time as well as memory consumption for this specific implementation, factors that are not observed with our method. Accordingly, we can conclude that overall, our method provides a more robust and efficient alternative.

Training Results. We further compare models trained with our custom kernel and to those using a baseline/reference `torch` implementation in Figure 6. We demonstrate here that both methods are indistinguishable in terms of loss curves as well as gradient norms, indicating the correctness and stability of the kernel.

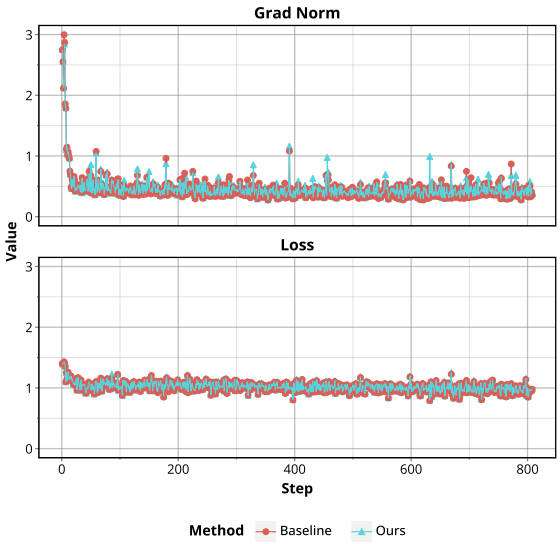

Figure 6. Training curves comparing our implementation against `torch.nn.CrossEntropyLoss` with $\beta = 0.1$. Training uses a `LLaMA3.2-3B` model using the `Alpaca` dataset.

6. Conclusion

In this paper, we present a novel perspective regarding the use of label smoothing as a functional mechanism to retain calibration after the supervised fine-tuning process (SFT) that is commonly used to train LLMs. We first identify why SFT can degrade calibration from a statistical standpoint before showing how label smoothing can help alleviate this concern. However, we also identify a setting where the use of label smoothing appears ineffective, particularly in the case of large language models with large vocabularies but smaller hidden sizes, demonstrating how such model construction implicitly impacts the entropy of predictions and leads to difficulty in becoming over-confident during SFT. Finally, we demonstrate a practical issue regarding the computational concerns of label smoothing in these settings. To further address this issue, we identify GPU accelerator level optimizations and provide a custom computational kernel, written in Triton, that enables us to maximize the use of accelerator memory/bandwidth and train models, improving both training/inference speed as well as memory consumption without sacrifices in stability.

Acknowledgements

Jerry Huang was supported by the NSERC Canada Graduate Scholarships — Doctoral (CGS-D) program (funding reference number 589326) as well as the Bourse d'Excellence Hydro-Québec program. This work was made possible in part thanks to computational resources from Calcul Québec[2] and the Digital Research Alliance of Canada (DRAC)[3].

[1] This method does not support label smoothing.

[2] https://www.calculquebec.ca/

[3] https://alliancecan.ca/en

Impact Statement

This paper proposes a method to improve calibration in large-vocabulary language models. We anticipate minimal societal impact from this work.

References

Bach, S. H., Sanh, V., Yong, Z. X., Webson, A., Raffel, C., Nayak, N. V., Sharma, A., Kim, T., Bari, M. S., Févry, T., Alyafeai, Z., Dey, M., Santilli, A., Sun, Z., Ben-David, S., Xu, C., Chhablani, G., Wang, H., Fries, J. A., Al-Shaibani, M. S., Sharma, S., Thakker, U., Almubarak, K., Tang, X., Radev, D. R., Jiang, M. T., and Rush, A. M. Promptsource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022 - System Demonstrations, Dublin, Ireland, May 22-27, 2022*, pp. 93–104, 2022.

Bertsekas, D. P. *Nonlinear Programming*. 1999.

Brier, G. W. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

Chi, T., Fan, T., and Rudnicky, A. Attention alignment and flexible positional embeddings improve transformer length extrapolation. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 132–148, 2024.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V. Y., Huang, Y., Dai, A. M., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. Scaling instruction-finetuned language models. *Journal on Machine Learning Research*, 25:70:1–70:53, 2024.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

DeGroot, M. H. and Fienberg, S. E. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 32(1):12–22, 1983.

Dubois, Y., Li, C. X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacafarm: A simulation framework for methods that learn from human feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Gemma Team. Gemma 2: Improving open language models at a practical size, 2024.

Grattafiori, A. et al. The llama 3 herd of models, 2024.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, 2017.

Hendrycks, D., Mazeika, M., and Dietterich, T. G. Deep anomaly detection with outlier exposure. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Hsu, P.-L., Dai, Y., Kothapalli, V., Song, Q., Tang, S., Zhu, S., Shimizu, S., Sahni, S., Ning, H., and Chen, Y. Liger kernel: Efficient triton kernels for llm training, 2024.

Huh, M., Cheung, B., Wang, T., and Isola, P. Position: The platonic representation hypothesis. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de Las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023.

Jiang, Z., Araki, J., Ding, H., and Neubig, G. How can we know *When* language models know? on the calibration of language models for question answering. *Trans. Assoc. Comput. Linguistics*, 9:962–977, 2021.

Kerr, A., Merrill, D., Demouth, J., and Tran, J. Cutlass: Fast linear algebra in cuda c++, December 2017.

Kumar, A., Raghunathan, A., Jones, R. M., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 2999–3007, 2017.

Liu, B., Ayed, I. B., Galdran, A., and Dolz, J. The devil is in the margin: Margin-based label smoothing for network calibration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 80–88, 2022.

Lu, P., Rashid, A., Kobyzev, I., Rezagholizadeh, M., and Langlais, P. LABO: towards learning optimal label regularization via bi-level optimization. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 5759–5774, 2023.

Lukasik, M., Bhojanapalli, S., Menon, A. K., and Kumar, S. Does label smoothing mitigate label noise? In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6448–6458, 2020.

Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., Tran, D., and Lucic, M. Revisiting the calibration of modern neural networks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 15682–15694, 2021.

Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P. H. S., and Dokania, P. K. Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Mukhoti, J., Gal, Y., Torr, P., and Dokania, P. K. Fine-tuning can cripple your foundation model; preserving features may be the solution. *Transactions on Machine Learning Research*, 2024, 2024.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4696–4705, 2019.

Murphy, A. H. Scalar and vector partitions of the probability score: Part i. two-state situation. *Journal of Applied Meteorology and Climatology*, 11(2):273–282, 1972.

Naeini, M. P., Cooper, G. F., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pp. 2901–2907, 2015.

Nixon, J., Dusenberry, M. W., Zhang, L., Jerfel, G., and Tran, D. Measuring calibration in deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 38–41, 2019.

Oh, C., Lim, H., Kim, M., Han, D., Yun, S., Choo, J., Hauptmann, A. G., Cheng, Z.-Q., and Song, K. Towards calibrated robust fine-tuning of vision-language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, December 10-15, 2024, Vancouver, BC, Canada*, 2024.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., and Hinton, G. E. Regularizing neural networks by penalizing confident output distributions. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017.

PyTorch. torchtune: Pytorch's finetuning library, April 2024.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N. V., Datta, D., Chang, J., Jiang, M. T., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Févry, T., Fries, J. A., Teehan, R., Scao, T. L., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2818–2826, 2016.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Naik, A., Ashok, A., Dhanasekaran, A. S., Arunkumar, A., Stap, D., Pathak, E., Karamanolakis, G., Lai, H. G., Purohit, I., Mondal, I., Anderson, J., Kuznia, K., Doshi, K., Pal, K. K., Patel, M., Moradshahi, M., Parmar, M., Purohit, M., Varshney, N., Kaza, P. R., Verma, P., Puri, R. S., Karia, R., Doshi, S., Sampat, S. K., Mishra, S., A, S. R., Patro, S., Dixit, T., and Shen, X. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 5085–5109, 2022.

Wang, Y., Ivison, H., Dasigi, P., Hessel, J., Khot, T., Chandu, K. R., Wadden, D., MacMillan, K., Smith, N. A., Beltagy, I., and Hajishirzi, H. How far can camels go? exploring the state of instruction tuning on open resources. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022a.

Wei, J., Liu, H., Liu, T., Niu, G., Sugiyama, M., and Liu, Y. To smooth or not? when label smoothing meets noisy labels. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23589–23614, 2022b.

Wijmans, E., Huval, B., Hertzberg, A., Koltun, V., and Krähenbühl, P. Cut your losses in large-vocabulary language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.

Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., and Hooi, B. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2.5 technical report, 2024.

Zhai, S., Likhomanenko, T., Littwin, E., Busbridge, D., Ramapuram, J., Zhang, Y., Gu, J., and Susskind, J. M. Stabilizing transformer training by preventing attention entropy collapse. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40770–40803, 2023.

Zhang, Z. and Sabuncu, M. R. Self-distillation as instance-specific label smoothing. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12697–12706, 2021.

Łukasz Rajkowski. Analysis of the Maximal a Posteriori Partition in the Gaussian Dirichlet Process Mixture Model. *Bayesian Analysis*, 14(2):477 – 494, 2019.

A. Proofs

A.1. Proof of Proposition 3.3

Proposition 3.3. A linear penalty (or a Lagrangian term) for the hard constraint $\boldsymbol{d}(\boldsymbol{x}) = \mathbf{0}$ is bounded from above and below by $\mathrm{KL}\left(\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right)$, up to additive constants

$$\mathrm{KL}[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)] - \log K \leq \sum_{i=1}^{K} \boldsymbol{d}\left(\boldsymbol{x}\right)_i / K \leq \mathrm{KL}\left[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right]. \tag{18}$$

Proof. We adapt the proof of Liu et al. (2022). Given the KL divergence

$$\mathrm{KL}\left[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right] = -\frac{1}{K} \sum_{k=1}^{K} \log P\left(\gamma_i | \boldsymbol{x}; \boldsymbol{\theta}\right) + \text{const}$$

we have that

$$\begin{aligned} \mathrm{KL}\left[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right] &= -\frac{1}{K} \sum_{k=1}^{K} \log \left(\frac{e^{\boldsymbol{\ell}(\boldsymbol{x}; \boldsymbol{\theta})_i}}{\sum_{j=1}^{K} e^{\boldsymbol{\ell}(\boldsymbol{x}; \boldsymbol{\theta})_j}} \right) + \text{const} \\ &= -\frac{1}{K} \sum_{k=1}^{K} \log \left(\sum_{j=1}^{K} e^{\boldsymbol{\ell}(\boldsymbol{x}; \boldsymbol{\theta})_j} - \boldsymbol{\ell}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)_i \right) + \text{const} \end{aligned} \tag{19}$$

Considering the property of the LogSumExp (LSE) function, it follows that

$$\max_j \ \boldsymbol{\ell}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)_j \leq \log \sum_{j=1}^{K} e^{\boldsymbol{\ell}(\boldsymbol{x}; \boldsymbol{\theta})_j} \leq \max_j \ \boldsymbol{\ell}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)_j + \log\left(K\right)$$

and

$$\mathrm{KL}\left[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right] - \log K \leq -\frac{1}{K} \sum_{k=1}^{K} \left(\max_j \ \boldsymbol{\ell}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)_j - \boldsymbol{\ell}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)_k \right) \leq \mathrm{KL}\left[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right] \tag{20}$$

and given the definition of $\boldsymbol{d}\left(\boldsymbol{x}\right)$, then the additional penalty $\mathrm{KL}\left[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right]$ imposed by LS in addition to the standard cross-entropy loss $\mathcal{L}^{\mathrm{CE}}$ is approximately optimizing a linear penalty (or a Lagrangian) for the constraint

$$\boldsymbol{d}\left(\boldsymbol{x}\right) = \mathbf{0}$$

to encourage equality of the logits. □

A.2. Proof of Proposition 3.4

Proposition 3.4. Define a likelihood model $p\left(y | \boldsymbol{x}; \boldsymbol{\theta}\right) = \mathrm{Cat}\left(\mathrm{softmax}\left(f\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right)\right)$, a categorical distribution with parameters $\boldsymbol{z} = \mathrm{softmax}\left(f\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right) \in \Delta(\Theta)$ where $\Delta\left(\Theta\right)$ denotes a probability simplex over the parameter space Θ. The label smoothing objective is equivalent to Maximum A Posteriori (MAP) estimation on the $\mathrm{softmax}$ probability vector under the independence assumption $p\left(\boldsymbol{z} | \boldsymbol{x}\right) = p\left(\boldsymbol{z}\right)$.

Proof. This proof is an adaptation of Zhang & Sabuncu (2020). Suppose a provide set of examples $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{i=1}^{N}$ sampled from $\mathcal{X} \times \mathcal{Y}$. The goal is to find a set of parameters $\boldsymbol{\theta}$ to parameterize a function f that maps inputs $\boldsymbol{x} \in \mathcal{X}$ to corresponding labels $y \in \mathcal{Y}$. Suppose that the likelihood $p\left(y | \boldsymbol{x}, \boldsymbol{z}\right) = \mathtt{Cat}(\boldsymbol{z})$ be a categorical distribution with parameter $\boldsymbol{z} \in \Delta\left(\Theta\right)$ and the conditional prior $p\left(\boldsymbol{z} | \boldsymbol{x}\right) = \mathtt{Dir}\left(\boldsymbol{\alpha}_{\boldsymbol{x}}\right)$ be a Dirichlet distribution with instance-specific parameter $\boldsymbol{\alpha}_{\boldsymbol{x}}$.

Due to conjugacy of the Dirichlet prior, a closed-form solution of $\hat{\boldsymbol{z}}_i = \frac{\boldsymbol{C}_i + \boldsymbol{\alpha}_{\boldsymbol{x}_i} - 1}{\sum_j \boldsymbol{C}_j + \boldsymbol{\alpha}_{\boldsymbol{x}_j} - 1}$, where $\boldsymbol{C}_i$ corresponds to number of occurrences of the i-th category, can be easily obtained.

Thus the MAP estimation $\hat{\boldsymbol{z}}_i \approx \operatorname{softmax}\left(f_{\boldsymbol{w}}\left(\boldsymbol{x}_i\right)\right)$ can be amortized with a given training set, resulting in an optimization problem of:

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \log p\left(\boldsymbol{z} | \boldsymbol{x}_n, y_n; \boldsymbol{\theta}, \boldsymbol{\alpha}_x\right) \tag{21}$$

$$= \max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log p\left(y = y_n | \boldsymbol{z}, \boldsymbol{x}_n; \boldsymbol{\theta}\right) + \log p\left(\boldsymbol{z} | \boldsymbol{x}_n; \boldsymbol{\theta}, \boldsymbol{\alpha}_x\right)$$

$$= \max_{\boldsymbol{w}} \underbrace{\frac{1}{N} \sum_{n=1}^{N} \log \left[\operatorname{softmax}\left(f\left(\boldsymbol{x}_n; \boldsymbol{\theta}\right)\right)\right]_{y_n}}_{\text{Cross Entropy}} + \underbrace{\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \left([\boldsymbol{\alpha}_{\boldsymbol{x}_n}]_k - 1\right) \log [\boldsymbol{z}]_k}_{\text{Instance-specific Regularization}} \tag{22}$$

where $[\cdot]_k$ denotes the k-th element of a vector. Using the assumption that $p\left(\boldsymbol{z}|\boldsymbol{x}\right) = p\left(\boldsymbol{z}\right)$, a sensible choice of prior would be a uniform distribution across all possible labels. Choosing $[\boldsymbol{\alpha}_{\boldsymbol{x}}]_k = [\boldsymbol{\alpha}]_k = \frac{\beta}{k} + 1$ for all $k \in \{1, \ldots, K\}$ for some hyper-parameter β, the MAP objective becomes

$$\mathcal{L}_{LS} = \frac{1}{N} \sum_{n=1}^{N} -\log [\boldsymbol{z}]_{y_n} + \frac{\beta}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} -\frac{1}{K} \log [\boldsymbol{z}]_k. \tag{23}$$

With some simple rearrangement of terms,

$$\begin{aligned}
\mathcal{L}_{LS} &= \frac{1}{N} \sum_{i=1}^{N} -\log [\boldsymbol{z}]_{y_n} + \frac{\beta}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} -\frac{1}{K} \log [\boldsymbol{z}]_k \\
&= -\frac{(1+\beta)}{N} \sum_{n=1}^{N} \left(\frac{K+\beta}{K(1+\beta)} \log [\boldsymbol{z}]_{y_n} + \sum_{K \neq y_n} \frac{\beta}{K(1+\beta)} \log [\boldsymbol{z}]_K \right)
\end{aligned}$$

Thus the above objective is equivalent to the label smoothing regularization with $1 - \epsilon = \frac{k+\beta}{k(1+\beta)}$, up to a constant factor of $(1+\beta)$. □

A.3. Proof of Lemma 4.1

Lemma 4.1. Let $\boldsymbol{C} \in \mathbb{R}^{D \times |\boldsymbol{V}|}$ be a matrix with $\|\boldsymbol{C}\|_2 = \sigma_C$, and let $\boldsymbol{h} \in \mathbb{R}^D$ be a vector such that each entry of $\boldsymbol{h}$ satisfies $|h_i| \leq \sigma_h$. The upper bound for $\|\boldsymbol{C}\boldsymbol{h}\|_2$ is:

$$\|\boldsymbol{C}\boldsymbol{h}\|_2 \leq \sigma_C \cdot \sigma_h \cdot \sqrt{D}.$$

Proof. For any vector $\boldsymbol{h} \in \mathbb{R}^D$, it follows that:

$$\|\boldsymbol{C}\boldsymbol{h}\|_2 \leq \|\boldsymbol{C}\|_2 \cdot \|\boldsymbol{h}\|_2 .$$

Substituting $\|\boldsymbol{C}\|_2 = \sigma_C$, we obtain:

$$\|\boldsymbol{C}\boldsymbol{h}\|_2 \leq \sigma_C \cdot \|\boldsymbol{h}\|_2 .$$

And

$$\|\boldsymbol{h}\|_2 \leq \sqrt{\sum_{i=1}^{D} \sigma_h^2} = \sqrt{D} \cdot \sigma_h.$$

Substituting the bound on $\|\boldsymbol{h}\|_2$ into the inequality for $\|\boldsymbol{C}\boldsymbol{h}\|_2$, we have the norm of logit vector $u \in \mathbb{R}^V$:

$$\|\boldsymbol{u}\|_2 = \|\boldsymbol{C}\boldsymbol{h}\|_2 \leq \sigma_C \cdot \|\boldsymbol{h}\|_2 \leq \sigma_C \cdot \sqrt{D} \cdot \sigma_h.$$

□

A.4. Proof of Theorem 4.2

Proof. We begin our analysis of the entropy of the prediction distribution of LM head $\boldsymbol{p} \in \mathbb{R}^V$.

$$p_i = \frac{\exp(u_i)}{\sum_{j=1}^{V} \exp(u_j)}. \tag{24}$$

The entropy of $\boldsymbol{p}$ is then:

$$\mathcal{H}(\boldsymbol{p}) = -\sum_{j=1}^{V} p_i \log(p_i). \tag{25}$$

Without loss of generality, we assume $\|\boldsymbol{u}\|_2 =\leq \rho\sqrt{D}$, where $\rho = \sigma_c\sigma_h$. Then we aim to address the following constrained optimization problem:

$$\min_{\boldsymbol{u}} \mathcal{H}(\boldsymbol{u}),\ s.t\ \|\boldsymbol{u}\| \leq \rho \cdot \sqrt{D}, \tag{26}$$

We derive the global minimum for it with a Lagrangian multiplier and set the corresponding gradients equal to 0 then follow the analysis of Zhai et al. (2023):

$$\mathcal{L}(\boldsymbol{u}, \lambda) = \mathcal{H}(\boldsymbol{u}) + \lambda(\|\boldsymbol{u}\|^2 - \rho^2 D), \tag{27}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{u}, \lambda)}{\partial u} = 0,\ \frac{\partial \mathcal{L}(\boldsymbol{u}, \lambda)}{\partial \lambda} = 0. \tag{28}$$

Then we get:

$$\lambda u_i = \sum_{j=1}^{V} \frac{\exp(u_j)}{Z} \left[\delta_{i,j} - \frac{\exp(u_i)}{Z}\right] \left[1 + \log\left(\frac{\exp(u_j)}{Z}\right)\right], \tag{29}$$

$$= p_k[\log(p_k) + \mathcal{H}(\boldsymbol{u})].$$

$$\|\boldsymbol{u}\| = \rho^2 D. \tag{30}$$

Assume that for the minimizer u^* there exists an index i such that $u_i^* = 0$, we have:

$$\log(p_i^*) + \mathcal{H}(\boldsymbol{u}) = -\sum_{j=1}^{|\boldsymbol{V}|} p_j \log\left(\frac{p_j}{p_i^*}\right) = -\sum_{j=1}^{|\boldsymbol{V}|} p_j \log(e^{u_j}) = -\mathbb{E}u. \tag{31}$$

$$\forall u_m \neq 0, u_n \neq 0,\ p_m \frac{\log(p_m) + \mathcal{H}(\boldsymbol{u})}{u_m} = p_n \frac{\log(p_n) + \mathcal{H}(\boldsymbol{u})}{u_n} \tag{32}$$

$$\longrightarrow p_m + \frac{\mathbb{E}u}{u_m} = p_n + \frac{\mathbb{E}u}{u_n} \longleftrightarrow p_m = p_n.$$

which contradict to $\|\boldsymbol{u}\| = \rho^2 D$. Instead, assume $\forall_n u_n \neq 0$, based on Equation (30), we have:

$$\forall u_n \neq u_m,\ \frac{p_m}{u_m}\left[\log(p_m) + \mathcal{H}(\boldsymbol{u})\right] = \frac{p_n}{u_n}\left[\log(p_n) + \mathcal{H}(\boldsymbol{u})\right] \tag{33}$$

$$\longrightarrow e_m^u \left(1 - \frac{\mathbb{E}\,(u)}{u_m}\right) = e_n^u \left(1 - \frac{\mathbb{E}\,(u)}{u_n}\right). \tag{34}$$

We could assume that a solution $\boldsymbol{u}$ must contain at least one negative component. To illustrate this, consider $\boldsymbol{u}$ where $u_i > 0$ component-wise and $\|\boldsymbol{u}\| \leq \rho\sqrt{D}$. We can always shift $\boldsymbol{u}$ by a vector $\boldsymbol{v}$, where $v_m = v_n$ for all m, n, ensuring that $\|\boldsymbol{u} - \boldsymbol{v}\| \leq \rho\sqrt{D}$ and that $\boldsymbol{u} - \boldsymbol{v}$ has at least one negative component. Since all components of $\boldsymbol{v}$ are equal and softmax is shift-invariant, it follows that $\mathrm{softmax}(\boldsymbol{u}) = \mathrm{softmax}(\boldsymbol{u} - \boldsymbol{v})$. Additionally, without loss of generality, we assume $\mathbb{E}u > 0$ based on the same reasoning. Let $u_m, u_n < 0$, then based on Equation (34):

$$e_m^u \left(1 - \frac{\mathbb{E}(u)}{u_m}\right) = e_n^u \left(1 - \frac{\mathbb{E}(u)}{u_n}\right) > 0 \tag{35}$$

As $f(x) = e^x \left(1 - \frac{\alpha}{x}\right)$ is monotonously increasing in $x \in (-\infty, 0)$ and $x \in [\alpha, \infty)$ for $\alpha > 0$, it is easy to see $u_m = u_n$. If $u_n < 0$ and $u_m > 0$, then $u_m > \mathbb{E}u$. As $f(x)$ is monotonous in x for both $x < 0$ and $x > \alpha$, we have a solution that $\boldsymbol{u}$ must have 2 unique values, one positive and one negative. Let the different elements be a, b. The minimizer $\boldsymbol{u}$ contains $|\boldsymbol{V}| - 1$ b and one a. Based on $\|\boldsymbol{u}\| = \rho^2 D$, we have:

$$a = \rho\sqrt{D}\sqrt{1 - \frac{1}{|\boldsymbol{V}|}}, b = -\rho\sqrt{D}\sqrt{\frac{1}{|\boldsymbol{V}|\left(|\boldsymbol{V}| - 1\right)}} \tag{36}$$

The corresponding entropy of the minimizer $\boldsymbol{u}$:

$$\log\left(1 + (|\boldsymbol{V}| - 1)\exp\left(-\rho\sqrt{\frac{D\,|\boldsymbol{V}|}{|\boldsymbol{V}| - 1}}\right)\right) + \frac{\rho\sqrt{D\,|\boldsymbol{V}|\,(|\boldsymbol{V}| - 1)}\exp\left(-\rho\sqrt{\frac{|\boldsymbol{V}|}{|\boldsymbol{V}|-1}}\right)}{1 + (|\boldsymbol{V}| - 1)\exp\left(-\rho\frac{|\boldsymbol{V}|}{|\boldsymbol{V}|-1}\right)} \tag{37}$$

□

B. Additional Results

B.1. Calibration Metrics

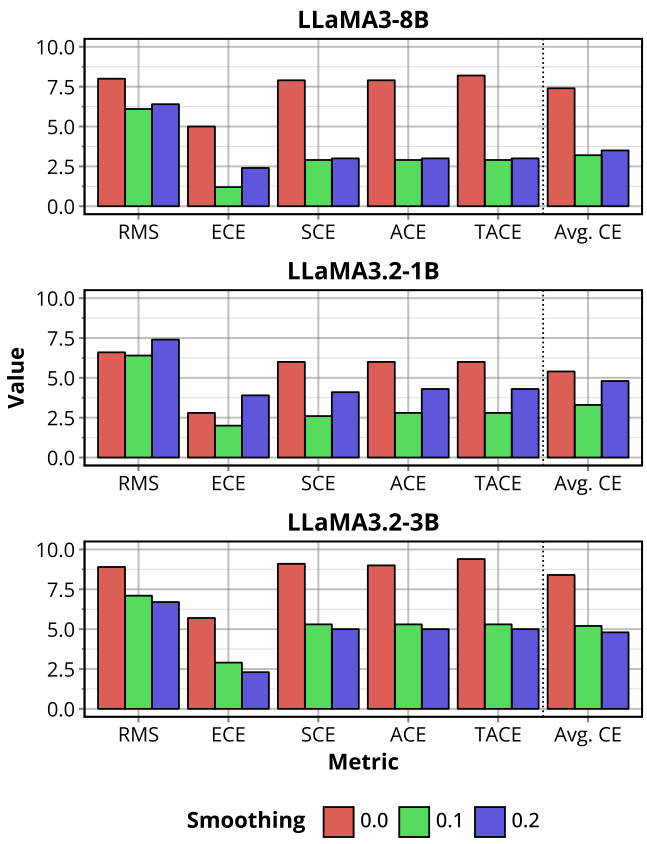

Figure 7. Calibration of different `LLaMA3` models fine-tuned on the same SFT dataset. As the size of the model decreases, the calibration of the model sees less improvement from the use of LS.

B.2. Reliability Diagrams

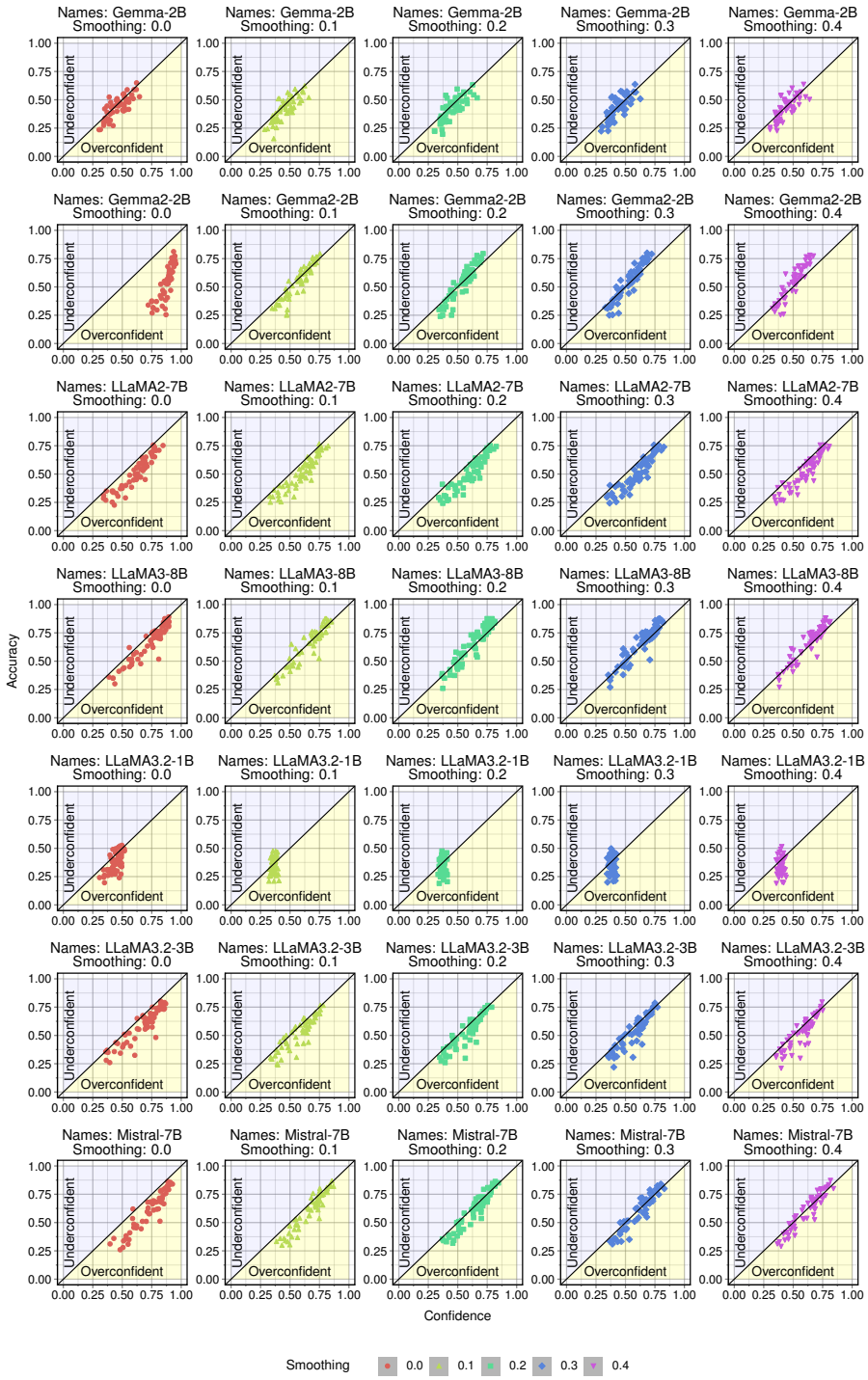

Figure 8. Reliability diagrams of models fine-tuned on the Tulu3 SFT Dataset and tested on the MMLU dataset.

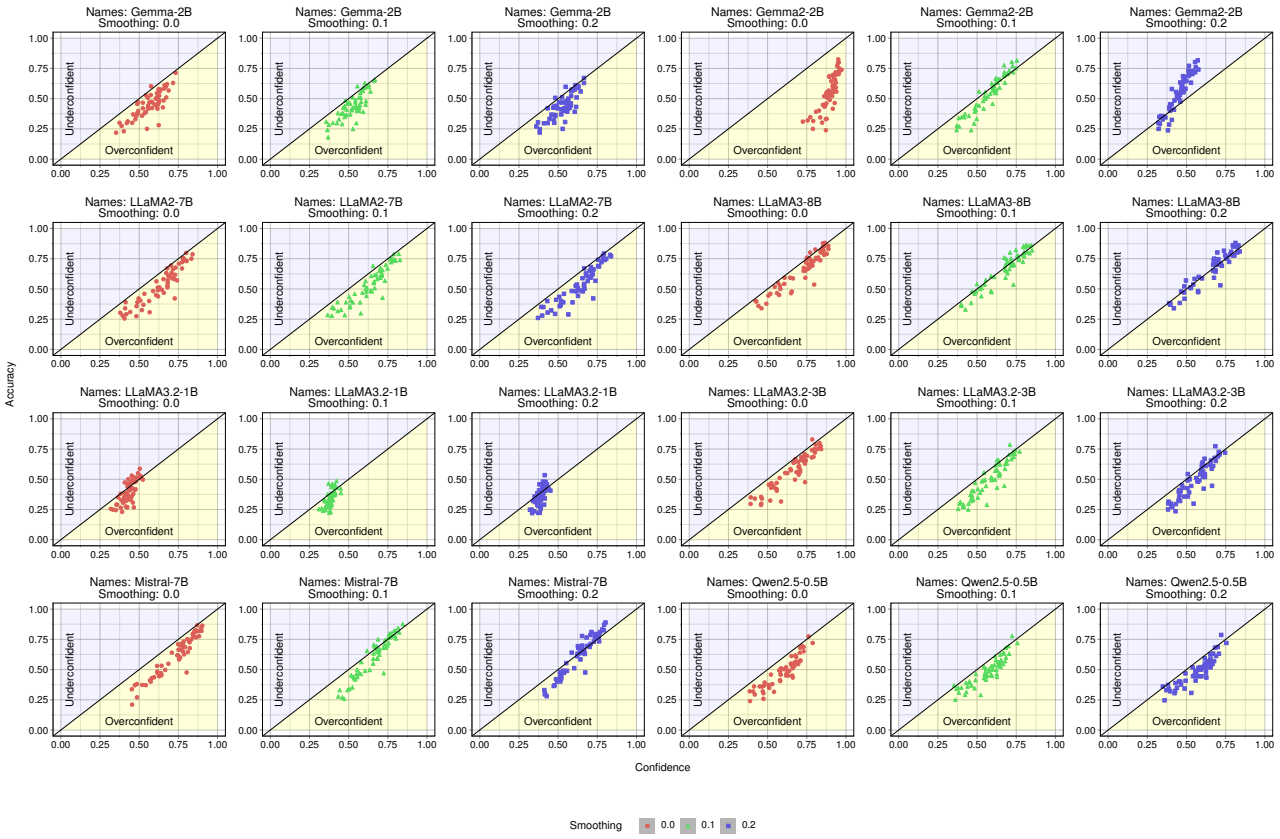

Figure 9. Reliability diagrams of models fine-tuned on the OpenHermes-2.5 SFT Dataset and tested on the MMLU dataset.

B.3. Efficient Smoothed Cross-Entropy

B.3.1. Training

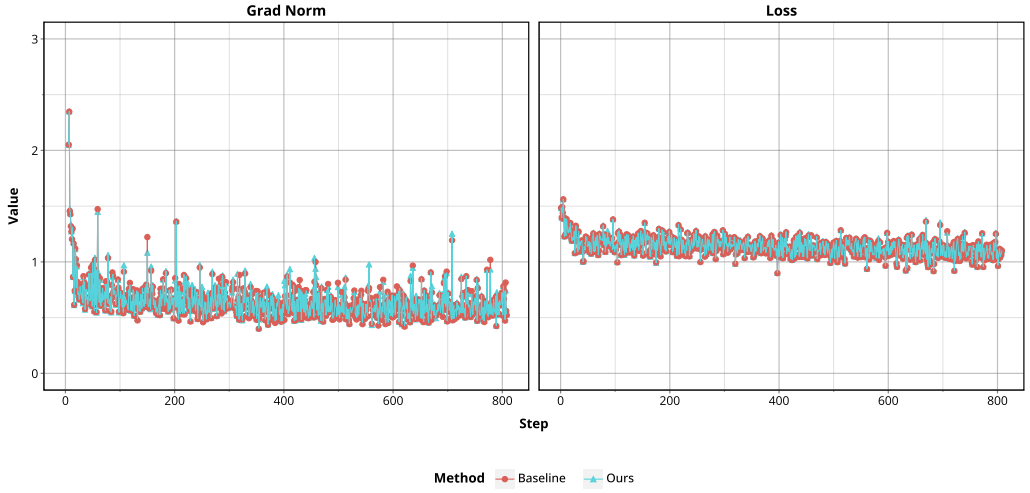

Figure 10. Training curves for `LLaMA3.2-1B`.

B.3.2. BENCHMARKING

Table 3. Memory footprint and time to compute losses and gradients. Results are computed on a batch size of 8192 tokens in a single sequence, generated from a `Gemma2` with 2 billion parameters (vocabulary size of 256K and hidden size 2304). Experiments are conducted on an H100-SXM5 GPU with 80GB of RAM, PyTorch 2.4.0 and CUDA 12.1.

Method	`fwd`		`bwd`		`fwd+bwd`	
	Memory	Time	Memory	Time	Memory	Time
Lower Bound	0.004 MB		1,161 MB		1,161 MB	
Smoothing $\beta > 0$						
Ours	1.1 MB	24.2 ms	1,163 MB	49.3 ms	1,164 MB	72.9 ms
+ (No Vocab Sorting)	0.09 MB	24.1 ms	1,162 MB	62.8 ms	1,162 MB	86.8 ms
+ (No Grad. Filter)	0.09 MB	24.0 ms	1,162 MB	177.1 ms	1,162 MB	201.5 ms
Fused Version	0.09 MB	100.5 ms	1,162 MB	0.2 ms	1,162 MB	100.6 ms
+ (No Grad. Filter)	0.09 MB	200.5 ms	1,162 MB	0.2 ms	1,162 MB	200.5 ms
Liger Kernels (Hsu et al., 2024)[4]	NA	NA	NA	NA	5,349 MB	155.0 ms
`torch.compile`	4,000 MB	22.8 ms	12,000 MB	38.3 ms	16,000 MB	62.3 ms
Baseline (`torch.nn.CrossEntropyLoss`)	24,000 MB	41.4 ms	16,000 MB	62.5 ms	28,000 MB	104.9 ms
Smoothing $\beta = 0$						
Ours	1.1 MB	24.0 ms	1,163 MB	49.2 ms	1,164 MB	72.9 ms
+ (No Vocab Sorting)	0.09 MB	23.9 ms	1,162 MB	62.4 ms	1,162 MB	85.2 ms
+ (No Grad. Filter)	0.09 MB	23.9 ms	1,162 MB	177.3 ms	1,162 MB	201.4 ms
Fused Version	0.09 MB	100.5 ms	1,162 MB	0.2 ms	1,162 MB	100.6 ms
+ (No Grad. Filter)	0.09 MB	200.5 ms	1,162 MB	0.2 ms	1,162 MB	200.5 ms
Cut-Cross Entropy (Wijmans et al., 2025)[5]	1.1 MB	23.6 ms	1,163 MB	49.2 ms	1,164 MB	72.4 ms
+ (No Vocab Sorting)	0.09 MB	23.5 ms	1,162 MB	62.5 ms	1,162 MB	85.0 ms
+ (No Grad. Filter)	0.09 MB	23.5 ms	1,162 MB	177.1 ms	1,162 MB	201.4 ms
Liger Kernels (Hsu et al., 2024)[6]	NA	NA	NA	NA	5,349 MB	154.8 ms
Chunked Cross Entropy (`torchtune`) (PyTorch, 2024) (8 chunks)	13,000 MB	30.4 ms	2,000 MB	51.0 ms	13,000 MB	82.8 ms
`torch.compile`	4,000 MB	20.6 ms	4,000 MB	33.9 ms	8,000 MB	55.0 ms
Baseline (`torch.nn.CrossEntropyLoss`)	24,000 MB	38.7 ms	16,000 MB	55.8 ms	28,000 MB	96.0 ms

B.4. Experimental Details

Training. We conducted a learning rate sweep over learning rates [5e-6, 2e-5, 5e-5, 2e-4] with a summing reduction. We further tested label smoothing hyper-parameters $[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]$, where 0.0 is no smoothing. We used the open-instruct repository at commit `e363290` for our training setup,[7] with modifications to account for our kernel as well as specific experimental hyper-parameter settings and baselines.

Evaluation. Our implementation is based on the MMLU official repository[8]. We first evaluate our models on MMLU and then modify the files here to directly adapt the evaluation dataset to the other tasks at hand. We follow MMLU and use the following prompt for all tasks: *'The following are multiple choice questions (with answers) about {}.\n\n'.format(query)*.

[7] https://github.com/allenai/open-instruct
[8] https://github.com/hendrycks/test