Calibrated Language Models and How to Find Them with Label Smoothing

Peng Lu[*1] Jerry Huang[*12] Qiuhao Zeng[3]

[1]Université de Montréal [2]Mila - Quebec AI Institute [3]Western University

May 21, 2025

Contents

Introduction

Label Smoothing

- has been demonstrated to be a promising paradigm in settings to prevent models from becoming overconfident or when noise exists in the provided labels.

Label Smoothing

- has been demonstrated to be a promising paradigm in settings to prevent models from becoming overconfident or when noise exists in the provided labels.
- 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

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

  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.

Label Smoothing

- has been demonstrated to be a promising paradigm in settings to prevent models from becoming overconfident or when noise exists in the provided labels.
- 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

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

  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

$$\mathcal{L}_{\mathcal{D}}^{\mathsf{CE}}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\delta_{y^{(n)}}^{\gamma_k}\log P(\gamma_k|\boldsymbol{x};\boldsymbol{\theta}) \approx -\mathbb{E}_{p(\boldsymbol{x},y)}\left[\sum_{k=1}^{K}p(\gamma_k|\boldsymbol{x})\log P(\gamma_k|\boldsymbol{x};\boldsymbol{\theta})\right]$$

Label Smoothing

- 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

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}^{\mathsf{CE}}(\boldsymbol{\theta}) &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\delta_{y^{(n)}}^{\gamma_k}\log P(\gamma_k|\boldsymbol{x};\boldsymbol{\theta}) \approx -\mathbb{E}_{p(\boldsymbol{x},y)}\left[\sum_{k=1}^{K} p(\gamma_k|\boldsymbol{x})\log P(\gamma_k|\boldsymbol{x};\boldsymbol{\theta})\right] \\ &= -\mathbb{E}_{p(\boldsymbol{x},y)}[\mathrm{KL}\left[\boldsymbol{\sigma}(\boldsymbol{x})\|\hat{\boldsymbol{\sigma}}(\boldsymbol{x};\boldsymbol{\theta})\right]] + c = \mathcal{L}_{p(\boldsymbol{x},y)}^{\mathsf{CE}}(\boldsymbol{\theta}),\end{aligned} \tag{2}$$

where δ_i^j is the Kronecker delta with value 1 only when $i = j$.

Label Smoothing

- 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

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}^{\mathsf{CE}}(\boldsymbol{\theta}) &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\delta_{y^{(n)}}^{\gamma_k}\log P(\gamma_k|\boldsymbol{x};\boldsymbol{\theta}) \approx -\mathbb{E}_{p(\boldsymbol{x},y)}\left[\sum_{k=1}^{K} p(\gamma_k|\boldsymbol{x})\log P(\gamma_k|\boldsymbol{x};\boldsymbol{\theta})\right] \\ &= -\mathbb{E}_{p(\boldsymbol{x},y)}[\mathrm{KL}\left[\boldsymbol{\sigma}(\boldsymbol{x})\|\hat{\boldsymbol{\sigma}}(\boldsymbol{x};\boldsymbol{\theta})\right]] + c = \mathcal{L}_{p(\boldsymbol{x},y)}^{\mathsf{CE}}(\boldsymbol{\theta}),\end{aligned} \tag{2}$$

  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 $\beta \in [0, 1]$.

Confidence Calibration

- **Model Calibration** is a concept of matching the prediction probabilities yielded for different inputs to the expected accuracy on these inputs.

Confidence Calibration

- **Model Calibration** 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 $\hat{y}$.

Confidence Calibration

- **Model Calibration** 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

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

Label Smoothing and Calibration

Label Smoothing Helps Calibration

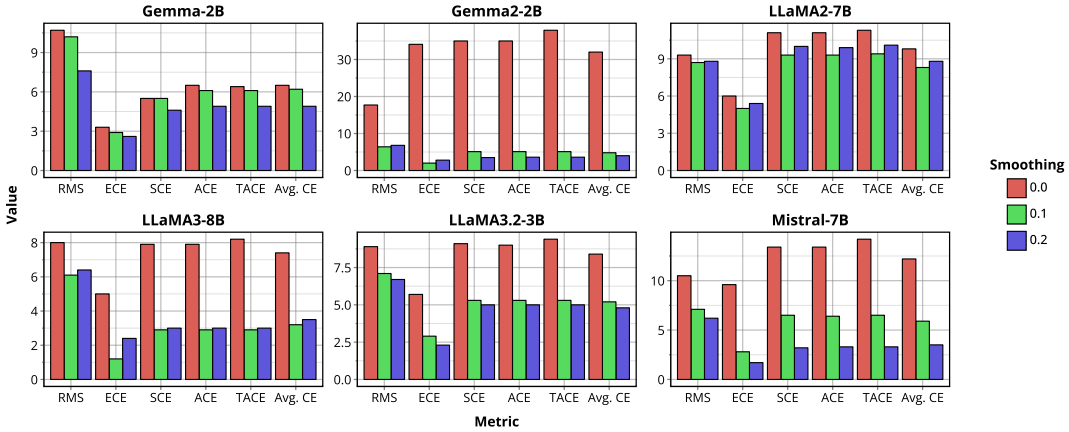


Figure: Effects of instruction-tuning on calibration. We can observe that across all models, which have various structural differences, the use of label smoothing is capable of reducing calibration error while having negligible effects on downstream performance accuracy on the task.

Effectiveness of Label Smoothing

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**

Why?

Logit Distance

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

$$\boldsymbol{d}(\boldsymbol{x}) = \left[\max_{1 \leq i \leq K} \boldsymbol{\ell}(\boldsymbol{x})_i - \boldsymbol{\ell}(\boldsymbol{x})_k\right]_{k=1}^{K} \in \mathbb{R}^K. \tag{3}$$

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.

Why?

Label Smoothing as a Linear Constraint

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

$$\mathrm{KL}[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)] - \log K \leq \sum_{i=1}^{K} \frac{\boldsymbol{d}\left(\boldsymbol{x}\right)_i}{K} \leq \mathrm{KL}\left[\boldsymbol{u} \| \hat{\boldsymbol{\sigma}}\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right]. \tag{4}$$

Label Smoothing and MAP Estimation

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 $p\left(\boldsymbol{z}|\boldsymbol{x}\right) = p\left(\boldsymbol{z}\right)$.

But What Happes For Large Vocabulary LLMs?

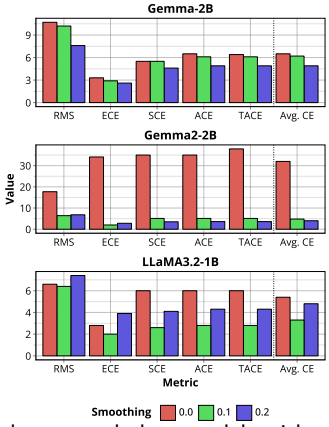


Figure: Effect of label smoothing on large vocabulary models with a smaller hidden size (2048). `Gemma-2B` observes a smaller change compared to `LLaMA3.2-1B`, due to having the largest vocabulary size. However, `Gemma2-2B` observes a large change in part thanks to the softcapping of logits.

Why Not For Large Vocabulary LLMs?

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:

$$\mathcal{H}_{\boldsymbol{u}} \geq \log\left(1 + (|\boldsymbol{V}| - 1)\,\gamma\right) + \frac{\rho \cdot \gamma\sqrt{D\,|\boldsymbol{V}|\,(|\boldsymbol{V}| - 1)}}{1 + (|\boldsymbol{V}| - 1)\,\gamma}. \tag{5}$$

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.

Efficient Kernel

Background

Consider the label smoothed loss:

$$\mathcal{L}_{\boldsymbol{x}}^{\mathsf{LS}} = \sum_{i=1}^{N} \mathcal{L}_{\boldsymbol{x}_i}^{\mathsf{LS}} = \sum_{i=1}^{N} \Bigg[\underbrace{(1-\beta)\, \boldsymbol{C}_{\boldsymbol{x}_i}^{\top} \boldsymbol{E}_i}_{\text{(1)Target Loss}} + \underbrace{\frac{\beta}{|\boldsymbol{V}|} \sum_{v \in \boldsymbol{V}} \boldsymbol{C}_v^{\top} \boldsymbol{E}_i}_{\text{(2)Smoothing Loss}} \underbrace{- \log \sum_{v \in \boldsymbol{V}} \exp\left(\boldsymbol{C}_v^{\top} \boldsymbol{E}_i \right)}_{\text{(3)LSE}} \Bigg], \quad (6)$$

(2) means all logits need to be explicitly added to the loss.

Fixing This

Saving computations

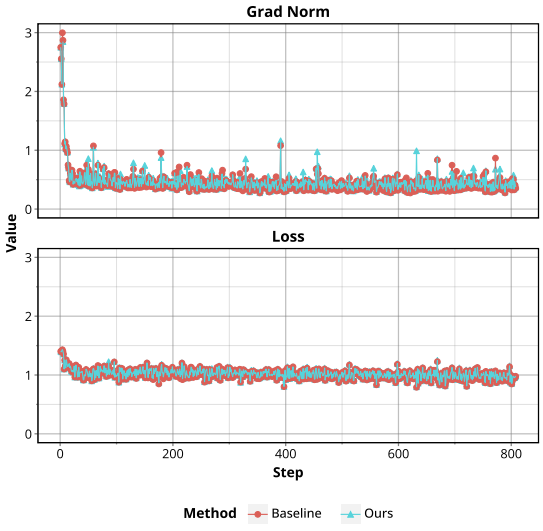
You can fix this by storing intermediate block-wise logit computations and use the final LSE within the $\mathrm{softmax}$ computation!

This forms the basis of an efficient kernel we build.

Efficient Kernel

Method	fwd		bwd		fwd+bwd	
	Memory	Time	Memory	Time	Memory	Time
Smoothing $\beta > 0$						
Ours	1.1MB	24.2ms	1163MB	49.3ms	1164MB	72.9ms
`torch.compile`	4000MB	22.8ms	12000MB	38.3ms	16000MB	62.3ms
Baseline	24000MB	41.4ms	16000MB	62.5ms	28000MB	104.9ms
Smoothing $\beta = 0$						
Ours	1.1MB	24.0ms	1163MB	49.2ms	1164MB	72.9ms
Cut-Cross Entropy	1.1MB	23.6ms	1163MB	49.2ms	1164MB	72.4ms
`torch.compile`	4000MB	20.6ms	4000MB	33.9ms	8000MB	55.0ms
Baseline	24000MB	38.7ms	16000MB	55.8ms	28000MB	96.0ms

Efficient Kernel



Conclusion

Final Takeaways

- Instruction-following large language models, though powerful, suffer from confidence calibration concerns.

Final Takeaways

- Instruction-following large language models, though powerful, suffer from confidence calibration concerns.
- Label smoothing, though simple, is effective at reducing this concern particularly during the instruction-tuning phase of training.

Final Takeaways

- Instruction-following large language models, though powerful, suffer from confidence calibration concerns.
- Label smoothing, though simple, is effective at reducing this concern particularly during the instruction-tuning phase of training.
- However, label smoothing diminishes in effectiveness as the vocabulary size decreases, necessitating the need for alternative methods such as temperature tuning or logit scaling.

Final Takeaways

- Instruction-following large language models, though powerful, suffer from confidence calibration concerns.
- Label smoothing, though simple, is effective at reducing this concern particularly during the instruction-tuning phase of training.
- However, label smoothing diminishes in effectiveness as the vocabulary size decreases, necessitating the need for alternative methods such as temperature tuning or logit scaling.
- Additionally, label smoothing is expensive due to needing to needing to materializing all logits within the GPU memory.

Contributions

1. We build a better understanding of when and why label smoothing can help calibrate instruction-tuned LLMs.

Contributions

1. We build a better understanding of when and why label smoothing can help calibrate instruction-tuned LLMs.
2. We provide a efficient kernel that computes cross entropy losses with label smoothing, significantly reducing memory usage while improving computation speed.

Thank you!

peng.lu@umontreal.ca