C^3: Confidence Calibration Model Cascade for Inference-Efficient Cross-Lingual Natural Language Understanding

Anonymous ACL submission

Abstract

Cross-lingual natural language understanding (NLU) is a critical task in natural language processing (NLP). Recent advancements have seen multilingual pre-trained language models (mPLMs) significantly enhance the performance of these tasks. However, mPLMs necessitate substantial resources and incur high computational costs during inference, posing challenges for deployment in real-world and real-time systems. Existing model cascade methods seek to enhance inference efficiency by greedily selecting the lightest model capable of processing the current input from a variety of models, based on model confidence scores. Nonetheless, deep models tend to exhibit overconfidence, and confidence distributions vary across languages. This leads to the emission of confident but incorrect predictions by smaller models, hindering their ability to generalize effectively across test languages. In this study, we introduce **C**onfidence **C**alibration **C**ascade (**C**^{3}), a simple yet effective method that involves calibration prior to cascade inference, thereby enhancing cascade accuracy through more reliable predictions. Our evaluation of **C**^{3}, using both encoder-only and decoder-only language models, across five cross-lingual benchmarks covering classification and generation tasks, shows that **C**^{3} markedly surpasses all leading baselines.

1 Introduction

Pre-trained language models (PLMs), such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2020b), T5 (Raffel et al., 2020), and Llama (Touvron et al., 2023), have exhibited remarkable performance across various natural language processing (NLP) tasks. Notably, their multilingual versions have demonstrated impressive zero-shot transfer capabilities in cross-lingual settings (Pires et al., 2019; Conneau et al., 2019). In these scenarios, PLMs are fine-tuned on English data, often with limited or even without data from other languages, yet they acquire the proficiency to handle tasks in different languages. However, multilingual PLMs are typically constructed using stacked transformer layers or their variants, employing self-attention mechanisms to capture diverse and distant dependencies among tokens. The use of self-attention introduces significant computational complexity. Consequently, the inference complexity of multilingual PLMs has become a bottleneck, limiting their deployment on devices sensitive to latency and constrained by computational resources.

To fulfill the stringent requirements for efficient inference in applications, various methods have been proposed to accelerate Pre-trained Language Model (PLM) inference. These methods include model compression (Sanh et al., 2019; Jiao et al., 2020; Sun et al., 2020, 2019), early exiting (Xin et al., 2020; Zhou et al., 2020; Liao et al., 2021), and model cascading (Li et al., 2020; Wang et al., 2022). Among these, model cascading methods are particularly appealing for several reasons: 1) They do not depend on specific hardware support, such as custom chips and GPUs. 2) They eliminate the need to train an inference-efficient model from scratch on the pre-training corpora. 3) They offer flexibility to adapt to the latest, incrementally powerful PLMs. Model cascading methods involve the aggregation of multiple PLMs with different sizes. Confidence scores are computed sequentially, ranging from small to large size models, to determine the appropriate model to employ. Once a confidence score surpasses a threshold, the corresponding model is selected, and the inference process ends.

Cascade-based models, however, exhibit notable limitations in cross-lingual scenarios. The confidence score, which measures the probability of the current prediction being correct in cascade-based models, is determined by the maximum output probability, the mean of the output probability, or the entropy of the output probability. Unfor-

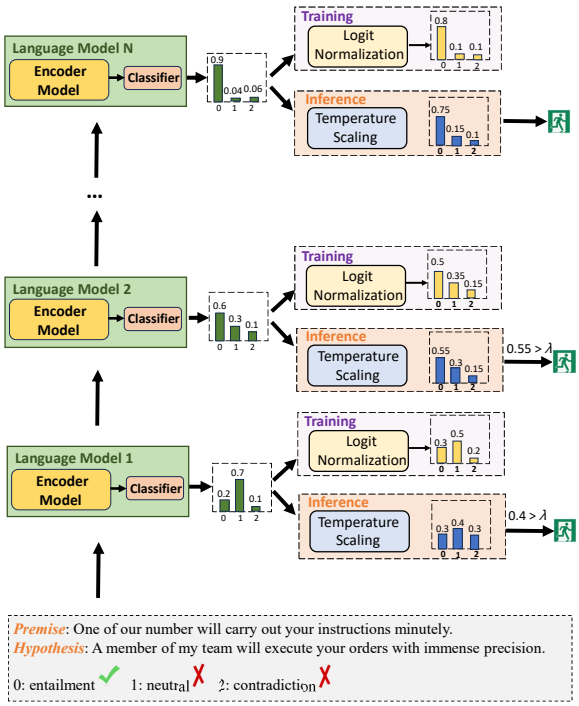


Figure 1: An illustration of our **C**^{3} framework (for classification task) for speeding up natural language inference yet retain the most accuracy, especially in OOD data. We leverage Logit Normalization at training time and Temperature Scaling at inference time to calibrate each model so that the model will yield more reliable confidence score for cascade decisions. For Large Language Model (e.g., GPT-4, Llama) inference where there is no training involved, we simply remove the training module. The λ in the figure represents the confidence score.

tunately, modern neural networks often generate unreliable confidence scores, particularly in out-of-distribution (OOD) scenarios (Guo et al., 2017; Wei et al., 2022). The cross-lingual task represents a typical OOD setting, where the training data exhibits significantly different distributions compared to the testing data. Consequently, the distribution of confidence scores on the training data differs from that on the testing data. Applying a threshold derived from the training set, based on pre-defined inference budgets, may prove either too conservative or radical for the testing data, leading to a performance or efficiency drop.

To tackle the challenges outlined earlier, we introduce a **C**onfidence **C**alibration **C**ascade (**C**^{3}) framework for efficient cross-lingual inference. The motivation behind our proposed approach is to calibrate the confidence of Multilingual Pre-trained Language Models (mPLMs), allowing the threshold determined on English data to be applicable to other languages. Specifically, we introduce a plug-in calibration step at the base of mPLMs. Initially, we normalize the logits to alleviate over-confidence during model fine-tuning. Subsequently, we implement a temperature scaling step to adjust the logits with a learnable scalar parameter. The proposed framework calibrates each individual model in the cascade, providing more reliable confidence scores. This, in turn, enhances the model's performance and generalization capabilities, leading to consistent improvements in efficiency and accuracy across different languages. Importantly, the proposed framework only requires an extra calibration module at the base of mPLMs, preserving the original architectures of mPLMs. Hence, it demonstrates flexibility to accommodate the latest models with minimal additional training overhead.

The primary contributions of this paper is **C**^{3}, a flexible and effective framework for enhancing efficiency in cross-lingual inference. To the best of our knowledge, this is the first work dedicated to the design of inference-efficient models specifically tailored for cross-lingual scenarios. Based on the observation of a notable overconfidence phenomenon in both encoder-only PLMs and decoder-only PLMs in cross-lingual scenarios, and considering that the extent of overconfidence appears to be correlated with linguistic distance, we introduce a plug-in calibration module to address this issue. Extensive experiments on five cross-lingual benchmarks, including XNLI, PAWS-X, QAM, GSM8k, and TabMWP, where the first three are text classification datasets and the later two are generation datasets, indicate that the proposed **C**^{3} outperforms baselines significantly and achieves a good efficiency-accuracy trade-off, e.g., preserving 98.10% of BERT's performance and 95.28% of Llama-2's performance on classification task and an average of 74.3% performance on generation task only half the computation costs.

2 Related Work

2.1 Inference Acceleration for PLMs

Existing approaches for accelerating the inference of pre-trained language models (PLMs) can be categorized into two types: 1) model compression-based methods and 2) dynamic network-based methods. The proposed method is more relevant to dynamic network-based methods, which are discussed in more detail as follows. The discussion about the model compression-based methods can be found in the Appendix A.

Dynamic network-based methods leverage models with different sizes based on input data when inference. Early existing methods and cascade methods are two typical dynamic networks.

Early exiting Methods. There are a number of early exiting methods that accelerate model inference by emitting predictions from an inner layer. FastBERT (Liu et al., 2020a), DeeBERT (Xin et al., 2020), and SDN (Kaya et al., 2019), are score-based methods, using the entropy of the prediction probability and the maximum of the predicted distribution as the score for exit decision. BERxiT (Xin et al., 2021) is another type of early-exiting method that learns to exit. Also, patience-based methods, such as PABEE (Zhou et al., 2020), SENTEE (Li et al., 2021), and LeeBERT (Zhu, 2021), ensemble scores from multiple layers to make decision. However, the problem with early existing methods is that the model loses high-level semantic understanding without going into the higher layers.

Cascade Methods. The main difference between cascade and early existing methods is cascade methods choose models with different sizes to make predictions while early existing methods choose a layer to exit with respect to a specific model. More specifically, cascades combine different sizes of models and go through each model sequentially from the smallest one to the largest one. Once the prediction from one model is confident enough, the prediction is emitted. There are many previous works that incorporate model cascade for faster inference on certain tasks. For example, the (Viola and Jones, 2001) built a cascade of classifiers with increasing complexity to speed up facial recognition; CascadeBERT ensembles two models prediction and regularizes them with a difficulty-awareness regularization (Li et al., 2020); Window-Based Cascade caches output from multiple levels and compare them to a window threshold (Xia and Bouganis, 2023).

2.2 Model Calibration

As models become larger, they also tend to be less calibrated, meaning the confidence output by the model does not reflect the true probability. In fact, large models are found to be easily overconfident in a lot of scenarios. We found that the problem of miscalibration is essentially important in cascades, where confidence is directly used as a metric to determine whether the prediction from the smaller models are reliable. There are various calibration methods in previous works. For example, Temperature Scaling divides the logits by a learned parameter from the validation set to calibrate the model after training (Guo et al., 2017); Logit Normalization incorporates a modified loss function to learn to produce logits with smaller norms (Wei et al., 2022); AugMax uses data augmentation techniques to calibrate models (Wang et al., 2021).

3 Preliminaries: Model Cascade

In this section, we introduce the overall flow of the vanilla model cascade. The model cascade consists of n PLMs, denoted as $\{M_i\}_{i=1}^n$, with the sizes of these PLMs arranged in ascending order, i.e. $|M_i| < |M_j|$ for $i < j$ where $|\cdot|$ represents the size of the model. The cascade model leverages each PLM for inference sequentially and computes the confidence score based on prediction logits of current PLM. If the confidence score exceeds a predefined threshold, the cascade model terminates its inference. Specifically, for an input text sequence t and the current PLM M_i, the output logits can be represented as

$$L = M_i(t), \tag{1}$$

where $L = \{l_j\}_{j=1}^q$ and q is the number of classes in the label. Subsequently, it computes the confidence score based on the logits L, such as by using averaged probability or maximum probability. We illustrate the calculation using maximum probability as an example:

$$C = \arg\max_j \frac{\exp l_j}{\sum_k \exp(l_k)}. \tag{2}$$

If it has traveled all models or $C > \lambda$, where λ is a threshold, the inference ends and returns the current logits L.

4 Confidence Calibration Cascade

Given n pre-trained language models (PLMs) with ascending sizes $\{M_i\}_{i=1}^n$ and source language (English in our paper) training set $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$, where x^i is the input sequence, y^i is the corresponding label and N is the number of training data, the aim is to fine-tune these PLMs and select the most suitable PLM that enhances the inference efficiency while preserving accuracy on both source and target languages with respect to the largest PLM M_n.

4.1 Overview

To enhance model inference efficiency in cross-lingual scenarios, we propose a **c**onfidence **c**alibration model **c**ascade method (**C**^{3}), as illustrated in Fig. 1. The motivation behind this approach is to select the most lightweight model for

each input based on the model confidence scores. To ensure the reliability of the confidence score, we introduce an additional calibration module integrated into vanilla cascade methods, which includes logit normalization and temperature scaling. We present the proposed **C**^{3} with encoder-only language models in Section 4.2 and decoder-only language models in Section 4.3.

4.2 Confidence Calibration for Language Model Cascade

In the vanilla model cascade method, a sample sequentially progresses through models of increasing size. A prediction is made when a model's confidence score exceeds a predetermined threshold, λ, or when the final model in the sequence is reached.

Our proposed **C**^{3} approach to cascading language models introduces two additional steps to the standard model cascade procedure. These steps aim to align the confidence distributions across various languages and models of different sizes. The first step involves integrating a Logit Normalization layer during the fine-tuning of each language model. Consider n language models $\{M_1, .., M_n\}$. Each model is fine-tuned using a task-specific loss function $\ell(\boldsymbol{x}, \boldsymbol{y})$, where the input logits are normalized. This can be expressed as:

$$\ell(\boldsymbol{x}, \boldsymbol{y}) = -\log \frac{\exp(\frac{\boldsymbol{l}_y}{\tau \|\boldsymbol{l}\|})}{\sum_{i=1}^{q} \exp(\frac{l_i}{\tau \|\boldsymbol{l}\|})}, \quad (3)$$

where q is the number of classes, $\boldsymbol{l} = [\boldsymbol{l}_1, ..., \boldsymbol{l}_q]$ is the logits, $\|\boldsymbol{l}\|$ is the norm of the logits, and τ is a hyper-parameter to modulate the model's output logits. The rationale for Logit Normalization is to counteract the tendency of the cross-entropy loss function, which often encourages the model to produce increasingly large logits during training, leading to overconfident Softmax scores. By maintaining a constant ℓ_2 norm of logits, Logit Normalization seeks to address this issue.

The second step involves applying temperature scaling to each fine-tuned model. Temperature scaling adjusts the model output logits by scaling them with a temperature parameter learned from the validation data. More precisely, we learn the temperature parameter T_i for each model M_i, and then we scale the model logits $\boldsymbol{l}$ by the temperature when calculating the confidence score C.

$$C_i = \arg\max_j \frac{\exp(\boldsymbol{l}_j / T_i)}{\sum_k \exp(\boldsymbol{l}_k / T_i)} \quad (4)$$

Table 1: **Example C^3 one-shot prompt on Llama-2 for PAWS-X task.** The answer candidates set would be {Yes, No}, and the embedding set would be {y, n}, where y is the Llama embedding of the word "Yes," and n is the Llama embedding for the word "No."

User
Do (Sentence1, The NBA season of 1975 – 76 was the 30th season of the National Basketball Association.) and (Sentnece2, The 1975 – 76 season of the National Basketball Association was the 30th season of the NBA.) have different meaning?
Answer: No

Do (Sentence1: It is the seat of Zerendi District in Akmola Region.) and (Sentence2: It is the seat of the district of Zerendi in Akmola region.) have different meaning?
Answer:

Assistant
No

These enhancements aim to calibrate model predictions and enhance the reliability of confidence scores, a critical factor in model selection within a cascade framework.

4.3 Confidence Calibration for Large Language Model Cascade

In this section, we illustrate the application of our proposed method **C**^{3} to large language models (LLMs). As with **C**^{3} for encoder-only LMs, we compile a collection of n LLMs of varying sizes. However, in contrast to the full fine-tuning approach described in Section 4.2, the adaptation of LLMs to specific downstream tasks is more effectively achieved through zero-shot or few-shot in-context learning. This is due to the substantial number of parameters in LLMs and their inherent capacity for generalization. Following this approach, we incorporate the task description and examples in English into a prompt, as detailed in Table 1. The LLM M_i then generates logits for the next token from the vocabulary, which can be represented as:

$$\boldsymbol{l} = \text{Logits}_{M_i}(\text{Prompt}). \quad (5)$$

For cross-lingual NLU tasks, particularly classification scenarios, we extract the logit corresponding to each class from l using the token IDs for each class. This yields the logits for each class, denoted as $\{\boldsymbol{l}_{\boldsymbol{c}_j}\}_{j=1}^{q}$, where $\boldsymbol{c}_j$ is the j-th class. We then apply temperature scaling to calibrate each model's output using a temperature T_i same as Section 4.2

to achieve model confidence. Finally, we determine a threshold λ for model selection during inference. The complete framework is outlined in Alg. 1 in the appendix.

Furthermore, more generally, for generation tasks, unlike classification tasks where a set of logits per class can be obtained for temperature scaling, we employ Word-level Sequence Entropy (WSE) as a calibration method. Specifically, we compute the entropy for each token in the generated sequence, i.e.

$$E(\boldsymbol{Z}_i) = -log(p(\boldsymbol{Z}_i)), \tag{6}$$

where $\boldsymbol{Z}_i$ is the ith token in the generated sequence. For each token $\boldsymbol{Z}_i$, we remove it from the sequence and utilize a pre-trained sentence similarity model to calculate the similarity, denoted as $\boldsymbol{R}_i$. The entropy of the entire sequence is then determined by

$$E(\boldsymbol{Z}) = \frac{1}{n}\sum_{i=0}^{n} E(\boldsymbol{Z}_i)(1 - \boldsymbol{R}_i). \tag{7}$$

Finally, we establish a threshold, λ, for $\boldsymbol{Z}_i$ to manage the cascading of models effectively.

5 Experiment

In this section, we evaluate the performance of $\mathbf{C}^3$ on three cross-lingual benchmarks, aiming to answer the following questions: (1) How does $\mathbf{C}^3$ perform compared to state-of-the-art baselines? (2) Is the proposed calibration method effective to reduce the calibration error? (3) How does the performance change with varing hyper-parameter τ in Logit Normalization?

5.1 Experimental Setup

Datasets To evaluate the performance of our proposed $\mathbf{C}^3$ on **classification** tasks, we conduct comprehensive experiments on three widely-used cross-lingual benchmarks: XNLI (Conneau et al., 2018), PAWS-X (Yang et al., 2019), and QAM (Liang et al., 2020). These datasets cover natural language inference, paraphrase identification, and question-answering matching tasks, respectively. Model fine-tuning is exclusively performed using the English training set, with subsequent evaluation conducted directly on testing data in other languages. Additionally, to evaluate the performance of our proposed $\mathbf{C}^3$ on **generation** tasks, we also use GSM8k (Cobbe et al., 2021) and TabMWP (Lu et al., 2023) datasets. These two datasets both evaluate a model's mathematical reasoning capabilities, featuring questions as mathematical problems and answers in free-form text. Due to limited computational resources for running Large Language Models, we randomly selected 500 samples from the test set of each dataset for evaluation. Subsequently, we utilized the Google Translate API to translate these examples into five other languages: Spanish, German, Chinese, Japanese, and Korean. Detailed dataset statistics are presented in Table 7 in the appendix.

Baselines We adopt the following state-of-the-art methods as baselines: (1) **PABEE.** The early exiting method PABEE emits predictions from an inner layer of a model if the prediction remains consistent for a certain number of consecutive instances (Zhou et al., 2020); (2) **DeeBERT**, which is another early exiting method that utilizes off-ramps layers to determine whether the prediction is confident enough at an inner layer (Xin et al., 2020); (3) **CasacdeBERT**, which is a cascade-based method that performs inference based on complete models with early stopping criteria and a difficulty-aware regularization (Li et al., 2020); (4) **Cascade**, which is also a cascade-based method that goes through each model sequentially until the stopping restriction is met (Wang et al., 2022).

5.2 Performance Comparison

In this section, we vary the threshold for each method and compare their accuracy on the test set, maintaining an identical speed-up ratio to answer RQ1. The speed-up ratio (S) is computed by dividing the number of Floating Point Operations (FLOPs) that the largest model (XLM-RoBERTa-large and Llama2-70b-chat-hf) would require by the number of FLOPs of the current method. Consistent with previous work, we also consider three distinct speed-up ratios, i.e. 2, 3, and 4, for a comprehensive comparison.

5.2.1 Encoder-only Language Model

We fine-tune encoder-only language models, including DistilBERT, mBERT-base, XLM-RoBERTa-base, and XLM-RoBERTa large, on the XNLI, PAWS-X, and QAM datasets, as illustrated in tables from Table 2 to Table 4. According to these tables, the proposed method ($\boldsymbol{C}^3$) outperforms existing methods. Notably, $\mathbf{C}^3$ exhibits superior performance compared to other cascade-based methods, especially in languages distant from English. For instance, on Thai, $\mathbf{C}^3$ achieves an accuracy of 67.76%, surpassing the

Table 2: **Result comparison for cascading BERT, multilingual BERT base, XLM-RoBERTa base, and XLM-RoBERTa large on XNLI.** The blue rows represents a speed-up ration of 2. The red rows represents a speed-up ratio of 3. And the yellow rows represents a speed-up ration of 4.

Method	ar	bg	de	el	en	es	fr	hi	ru	sw	th	tr	ur	vi	zh	Avg.
PABEE	60.03	65.23	70.65	82.97	82.97	76.44	75.47	69.06	74.75	57.84	68.12	71.11	62.49	73.87	70.62	69.92
DeeBERT	38.32	61.47	46.13	59.70	58.90	55.73	56.27	51.08	48.42	51.52	45.71	57.21	55.05	49.48	48.42	52.23
CascadeBERT	71.36	75.12	77.06	72.62	86.72	80.36	79.18	63.65	74.10	56.89	56.76	68.57	62.48	70.35	73.59	71.25
Cascades	75.51	79.64	79.64	78.24	86.72	**82.11**	81.31	72.01	77.98	63.31	70.71	74.25	68.04	77.30	75.90	76.18
C^3	**75.75**	**81.00**	**80.74**	**79.94**	**87.31**	81.76	**81.50**	**73.81**	**78.30**	**66.99**	**74.09**	**75.47**	**68.86**	**77.68**	**75.99**	**77.28**
PABEE	52.89	52.89	62.07	61.25	76.03	70.09	68.8	57.86	62.99	46.70	60.56	62.98	53.21	65.09	63.89	61.15
DeeBERT	39.90	40.48	40.58	40.02	49.50	48.56	49.20	44.91	42.47	39.50	41.86	41.12	39.48	41.60	45.19	42.96
CascadeBERT	68.61	71.26	70.52	68.51	84.66	76.99	75.69	57.58	69.48	51.66	46.97	63.07	58.44	64.69	70.36	66.56
Cascades	71.32	76.39	77.22	74.39	**86.48**	**80.19**	79.02	66.78	74.91	59.24	59.50	69.70	**64.43**	74.49	**73.59**	72.51
C^3	**71.44**	**78.44**	**78.60**	**75.87**	86.07	79.68	**79.46**	**68.38**	**76.67**	**62.16**	**67.76**	**72.04**	63.35	**75.99**	73.53	**73.96**
PABEE	48.62	48.62	54.95	53.58	73.45	63.39	63.43	51.85	58.08	44.27	54.59	57.28	48.74	60.92	59.08	56.06
DeeBERT	38.92	37.52	39.76	37.02	43.39	40.36	41.04	37.88	40.12	37.30	38.48	36.23	35.91	41.60	39.58	39
CascadeBERT	63.49	67.74	69.40	63.35	83.64	76.10	74.86	54.93	68.89	48.50	42.43	59.2	55.63	61.38	68.44	63.86
Cascades	68.48	73.11	74.39	70.58	85.14	**78.82**	77.26	62.23	72.14	54.45	54.59	65.01	**60.85**	70.23	**71.75**	69.26
C^3	**69.88**	**75.07**	**75.51**	**72.42**	**85.45**	78.60	**78.28**	**64.17**	**74.49**	**56.49**	**58.22**	**66.99**	60.82	**72.30**	70.98	**70.64**

Table 3: **Result comparison for cascading BERT, multilingual BERT base, XLM-RoBERTa base, and XLM-RoBERTa large on PAWS-X.** The blue rows represents a speed-up ration of 2. The red rows represents a speed-up ratio of 3. And the yellow rows represents a speed-up ration of 4.

Method	en	fr	es	de	zh	ja	ko	Avg.
PABEE	90.35	87.75	85.75	86.2	79.4	74.45	73.90	82.54
DeeBERT	75.9	71.7	70.6	64.7	64.8	59.2	62.4	67.04
CascadeBERT	92.25	88.3	87.7	87.8	79.75	75.6	75.88	83.89
Cascades	**95.6**	**91.45**	90.85	90.2	84.4	80.4	79.85	87.53
C^3	**95.6**	90.85	**91.5**	**90.45**	**84.5**	**81.7**	**81.6**	**88.03**
PABEE	63.55	63.3	57.6	58.25	58.35	62.25	59.2	60.36
DeeBERT	67.45	64.8	63.9	60.75	60.4	56.8	57.4	61.64
CascadeBERT	94.2	87.35	87.85	85.7	77.85	75.15	72.6	82.96
Cascades	95.2	**90.8**	90.2	90.05	81.65	77.7	76.75	86.05
C^3	**95.6**	89.95	**90.85**	**90.25**	**82.45**	**78.55**	**78.75**	**86.63**
PABEE	60.3	63.2	57.25	57.4	57.75	56.8	58.1	58.69
DeeBERT	60	61.25	60.15	61	58.15	56.1	56.65	59.04
CascadeBERT	94.25	85.55	85.7	84.05	75.9	73.1	70.5	81.29
Cascades	95.2	**89.75**	89.3	88.3	80.3	75.7	74.75	84.76
C^3	**95.5**	88.8	**89.45**	**88.45**	**80.55**	**77.35**	**76.55**	**85.24**

baseline Cascade by 8.26% at a speed-up ratio of 3.

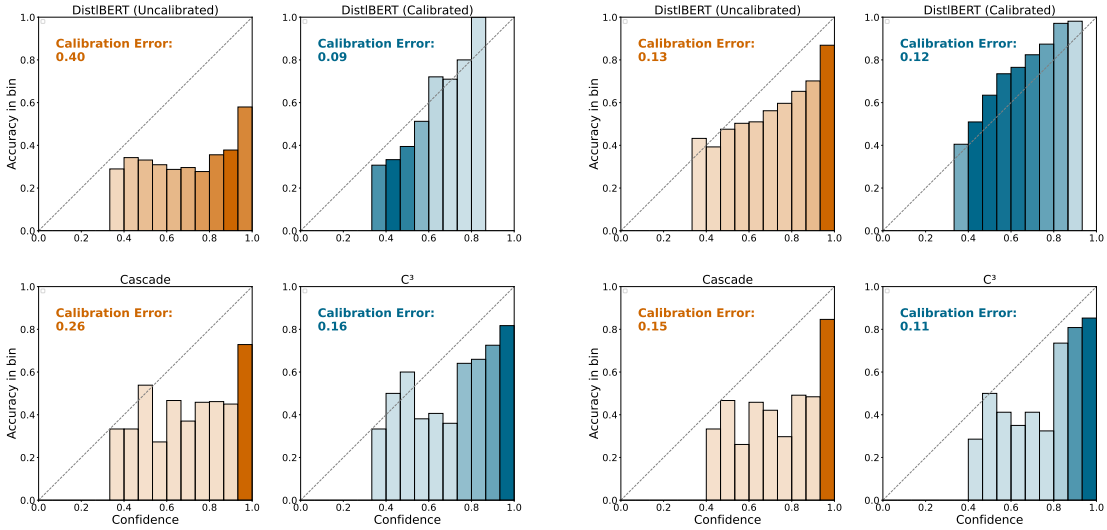
5.2.2 Decoder-only Language Model

We also conduct experiments using Llama2 as the backbone on PAWS-X to investigate the generalization capability of the proposed method (**C**^{3}) for large-scale decoder-only language models on classification task. The results are presented in Table 5. Notably, the framework exhibits only a marginal 3% decrease in accuracy when subjected to a two-fold speed-up, another approximately 3% decrease for a three-fold speed-up, and less than an additional 2% decrease for a four-fold speed-up. It's noteworthy that our findings lack accuracy data for the baseline Cascade. This absence is due to the remarkably concentrated confidence scores of Cascade, predominantly clustering around the value 1, resulting from the overconfident predictions by Llama-2. Consequently, establishing a threshold to achieve a specific speedup ratio becomes unfeasible, marking a limitation of Cascade.

We further present the results of our experiments using Llama2 as the backbone architecture on the GSM8k and TabMWP datasets in Table 5, demonstrating the generalizability of our method to generation tasks. In the majority of languages within these datasets, our method surpasses the Cascade method by margins ranging from 2% to 6.2%. In other instances, the two methods exhibit comparable performance.

5.3 Analysis

In this section, we study the effectiveness of the proposed calibration module to answer RQ2. We compute the calibration error of baseline Cascade and the proposed **C**^{3} on XNLI with a 2x speedup ratio. We show the results on four languages, in-



(a) Thai. (b) Spanish.

Figure 2: **ECE comparison between Cascade (in orange) and C^3 (in blue).** In each subfigure, the top two figures are the ECE of the smallest model in these two methods, and the bottom two figures are the overall ECE.

Table 4: **Result comparison for cascading BERT, multilingual BERT base, XLM-RoBERTa base, and XLM-RoBERTa large on QAM.** The blue rows represents a speed-up ration of 2. The red rows represents a speed-up ratio of 3. And the yellow rows represents a speed-up ration of 4.

Method	en	de	fr	Avg.
PABEE	64.3	64.12	65.75	64.72
DeeBERT	63.83	56.82	63.58	61.41
CascadeBERT	65.56	64.7	64.22	64.83
Cascades	71.41	68.4	68.83	69.55
C^3	**72.45**	**70.7**	**70.19**	**71.11**
PABEE	60.6	59.54	61.64	60.59
DeeBERT	57.8	53.32	57.36	56.16
CascadeBERT	66.28	62.98	64.18	64.48
Cascades	69.9	66.59	67.35	67.95
C^3	**70.95**	**67.54**	**68.55**	**69.01**
PABEE	59.41	57.94	58.5	58.62
DeeBERT	56.38	51.99	55.67	54.68
CascadeBERT	66.13	61.86	62.94	63.64
Cascades	68.52	**64.64**	65.73	66.3
C^3	**69.36**	64.45	**66.3**	**66.7**

cluding Thai in Fig. 2a, Spanish in Fig. 2b. See Appendix D for ECE of English and Swahili. Based on figures, we have following findings.

First, we observe that the proposed calibration method effectively reduces the calibration error for each individual model. In Fig. 2a, the expected calibration error (ECE) of the uncalibrated DistilBERT model on Thai data is notably high at approximately 0.4. Furthermore, we note a pattern of overconfidence in the uncalibrated model, with around 50% of data having a confidence level exceeding 0.8, yet their actual accuracy is only around 40%. This discrepancy is represented in the color depth of the histogram. In contrast, our proposed calibration method mitigates this overconfidence and miscalibration, as evidenced by the significantly reduced ECE of 0.09 for the calibrated DistilBERT, marking a substantial 78% decrease. Calibration contributes to more reliable model confidence. A similar trend is observed in the case of Spanish data, as shown in Fig. 2b, where the ECE for the uncalibrated DistilBERT is 12.62, and after calibration, it decreases to 11.88.

Second, the entire model benefits from the calibration process. As depicted in Fig. 2a, the ECE for uncalibrated model cascade is 25.81, whereas the ECE for calibrated cascade is reduced to 15.75, representing a substantial decrease of approximately 40% in ECE. Examining accuracy, the calibrated cascade achieves an accuracy of 74.09%, which is 3.38% higher than that of the uncalibrated cascade. Similar trends are observed in Spanish data, where the ECE for the uncalibrated cascade is 15.31, and the ECE for the calibrated cascade is 11.10.

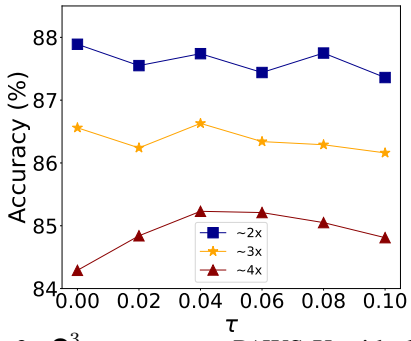


Figure 3: C^3 accuracy on PAWS-X with different τ values. We found that τ has a minimal impact on the accuracy. A τ of 0.04 works the best with a higher speed-up ratio.

Table 5: **Zero-shot esult comparison for Llama2 on PAWS-X (Classification), GSM8k (Generation), and TabMWP (Generation)**. Cascade method accuracy around 2, 3, 4 times speed-up on PAWS-X cannot be measured in our experiment because the concentrated confidence scores of Cascade, predominantly clustering around the value 1, resulting from the overconfident predictions by Llama-2. For example, the Cascade model is bounded at the ceiling at a 4.89 speed-up ratio. Any ratio smaller than that number cannot be measured as we alter the threshold.

Dataset	Speed-up	Model	en	fr	es	de	zh	ja	ko	Avg.
PAWS-X	1x	LLama-2-70B	78.2	72.05	71.15	72.8	67.6	65.9	64.7	70.34
	∼2x	Cascade	-	-	-	-	-	-	-	-
		$\mathbf{C}^3$	77.35	69.45	67.85	65.95	65.05	61.05	62.45	67.02
	∼3x	Cascade	-	-	-	-	-	-	-	-
		$\mathbf{C}^3$	70.05	68.05	66.45	63.9	62.7	59.85	60.05	64.44
	∼4x	Cascade	-	-	-	-	-	-	-	-
		$\mathbf{C}^3$	67	67.15	65.1	62.95	61.3	59.35	59.6	63.21
GSM8k	1x	LLama-2-70B	54.4	-	42.6	38.2	26.6	23.4	19.6	34.13
	∼2x	Cascade	30.4	-	21.8	18.2	**15.4**	11.8	**11.8**	18.23
		$\mathbf{C}^3$	**33.6**	-	**26.4**	**22**	13.4	**14.6**	10.4	**20.06**
	∼3x	Cascade	22.4	-	15.8	11.4	**12.2**	8.4	8.4	13.1
		$\mathbf{C}^3$	**25.4**	-	**20.8**	**15.2**	11.4	**14.6**	**8.6**	**16**
	∼4x	Cascade	18.8	-	13.4	10.2	**10.4**	7.6	7	11.23
		$\mathbf{C}^3$	**22.2**	-	**16.8**	**12.8**	9.8	**9**	**7.2**	**12.96**
TabMWP	1x	LLama-2-70B	65.2	-	51.8	39.8	35.2	38	40.2	45.03
	∼2x	Cascade	**64.8**	-	**46.4**	39.4	29.8	32.4	30.8	**40.6**
		$\mathbf{C}^3$	57.8	-	45.4	**39.6**	**33.4**	**34.4**	**32.2**	40.47
	∼3x	Cascade	53.8	-	44	37.4	28.8	29.4	26.4	36.7
		$\mathbf{C}^3$	**55**	-	**44.4**	**38**	**31.8**	**31.8**	**29.2**	**38.3**
	∼4x	Cascade	**54**	-	42	36.2	27.8	**30.6**	25.6	36.03
		$\mathbf{C}^3$	53.2	-	**42.2**	**38.4**	**30.8**	**30.6**	**28**	**37.2**

5.4 Sensitivity w.r.t. Hyper-parameters

We explore the sensitivity of our method concerning the parameter τ in logit normalization. In Fig. 3, we present the average accuracy of our model on the PAWS-X dataset with varying values of τ. With a two-fold speed-up, τ exerts minimal influence on cascade accuracy. However, when a higher speed-up ratio is required, we observe that a τ value of 0.04 corresponds to the peak accuracy.

5.5 Case Study

In this section, we present concrete cases to illustrate the functioning of the proposed $\mathbf{C}^3$. We examine several examples across different languages, including English, Bulgarian, German, Thai, and Chinese, where calibration rectifies predictions, as demonstrated in Figure 4 in the appendix. These cases reveal that vanilla cascade models frequently exhibit overconfidence, potentially leading to the selection of smaller models in error. In contrast, the proposed $\mathbf{C}^3$ adeptly chooses appropriate models for inference. For instance, when presented with the premise "They said we're providing accommodations for your stay" and the hypothesis "They're covering housing costs," the cascade model displays unwavering confidence in the prediction from the second model, while $\mathbf{C}^3$ selects the largest model. This highlights the prevalence of miscalibration, which often results in excessive overconfidence, causing the model to emit a prediction without progressing to larger models. Calibration plays a pivotal role in mitigating this issue, a pattern consistently observed across various languages.

6 Conclusion

In this paper, we propose a simple, yet effective method, **C**onfidence **C**alibration **C**ascade ($\mathbf{C}^3$) that enhances cross-lingual inference accuracy through more reliable confidence scores. We introduce calibration methods for both Language Model cascade and Large Language Model cascade. We conduct extensive experiment on cross-lingual benchmarks, including XNLI, PAWS-X, and QAM. By comparing with state-of-the-art methods, the results demonstrates the effectiveness of $\mathbf{C}^3$. Furthermore, $\mathbf{C}^3$ also demonstrates strong calibration results compared to vanilla cascade methods.

Limitations

The proposed $\mathbf{C}^3$ framework depends on a hyperparameter τ at a tuning process, which might require additional effort to tune. Fortunately, our experiment shows that the $\mathbf{C}^3$ model is not sensitive to this hyperparameter, and thus this issue can be easily addressed.

References

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2020. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

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

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2020. Cascadebert: Accelerating inference of pre-trained language models via calibrated complete models cascade. *arXiv preprint arXiv:2012.14682*.

Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2021. Accelerating bert inference for sequence labeling via early-exit. *arXiv preprint arXiv:2105.13878*.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, pages 20852–20867. PMLR.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation. *arXiv*, abs/2004.01401.

Kaiyuan Liao, Yi Zhang, Xuancheng Ren, Qi Su, Xu Sun, and Bin He. 2021. A global past-future early exit method for accelerating inference of pre-trained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2013–2023, Online. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhe Zha, Zhiruo Wang, Haotang Deng, and Qi Ju. 2020a. FastBERT: a self-distilling bert with adaptive inference time. In *Proceedings of ACL 2020*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. Roberta: A robustly optimized bert pretraining approach. In *International Conference on Learning Representations*.

Zejian Liu, Fanrong Li, Gang Li, and Jian Cheng. 2021. Ebert: Efficient bert inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4814–4823.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *International Conference on Learning Representations (ICLR)*.

Gunho Park, Baeseong Park, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. 2022. nuqmm: Quantized matmul for efficient inference of large-scale generative language models. *arXiv preprint arXiv:2206.09557*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Hanlin Tang, Xipeng Zhang, Kai Liu, Jianchen Zhu, and Zhanhui Kang. 2022. Mkq-bert: Quantized bert with 4-bits weights and activations. *arXiv preprint arXiv:2203.13483*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee.

Haotao Wang, Chaowei Xiao, Jean Kossaifi, Zhiding Yu, Anima Anandkumar, and Zhangyang Wang. 2021. Augmax: Adversarial composition of random augmentations for robust training. *Advances in neural information processing systems*, 34:237–250.

Xiaofang Wang, Dan Kondratyuk, Eric Christiansen, Kris M Kitani, Yair Alon, and Elad Eban. 2022. Wisdom of committees: An overlooked approach to faster and more accurate models. In *International Conference on Learning Representations*.

Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li. 2022. Mitigating neural network overconfidence with logit normalization. In *International Conference on Machine Learning*, pages 23631–23644. PMLR.

Guoxuan Xia and Christos-Savvas Bouganis. 2023. Window-based early-exit cascades for uncertainty estimation: When deep ensembles are more efficient than single models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main Volume*, pages 91–104.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. In *Proc. of EMNLP*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc.

Wei Zhu. 2021. Leebert: Learned early exit for bert with cross-level optimization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2968–2980.

A Related Work

Model Compression-based Methods. There are a number of model compression methods which can be applied to accelerate PLM inference, such as knowledge distillation, pruning, and weight quantization. Knowledge distillation is to leverage a powerful teacher model to guide the learning of a lightweight student model. The lightweight student can support efficient inference. For example, DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2020), MobileBERT (Sun et al., 2020), and PKD (Sun et al., 2019) use knowledge distillation to learn a lightweight BERT. (Liang et al., 2023), (Zhou et al., 2023), (Gu et al., 2023), (Gu et al., 2023) and (Shridhar et al., 2023) apply knowledge distillation to train a small generative model. Pruning focuses on finding redundant parameters and sets them as zero to achieve highly sparse neural network. For example, EBERT (Liu et al., 2021) and (Chen et al., 2020) propose a dynamic structured pruning method and a lottery ticket hypothesis (Frankle and Carbin, 2018) for BERT, and Sheared Llama (Xia et al., 2023) designs a structured pruning method for generative pre-trained language models. Weight quantization is to map the model weights into low-precision integers or floating-point numbers, which are hardware-friendly and efficient for matrix computation. Specifically, Q8BERT (Zafrir et al., 2019) proposes 8-bit quantization for BERT; MKQ-BERT (Tang et al., 2022) designs 4-bit quantization method for BERT; BinaryBERT (Bai et al., 2020) explore how to quantize BERT in 1 bit; Smoothquant (Xiao et al., 2023) and nuQmm (Park et al., 2022) study how to apply quantization for generative language models. However, these methods often need to train compressed models from scratch, which is expensive. Pruning and quantization rely on specialized hardware support, which is not flexible.

B Prompt Design

We include the prompt we use for Llama-2 in our experiment on PAWS-X in this section. Detailed prompt is shown in Table 6.

C Algorithm

D ECE Figures

We report the ECE figures for comparison between the Cascade (in orange) and **C**^{3} (in blue) on Swahili and English in Figure 6 and Figure 5.

User
Do (Sentence1, Example1: Sentence1) and (Sentnece2, Example1: Sentence2) have different meaning?
Answer: Example1: Answer.

Do (Sentence1: Example2: Sentence1) and (Sentence2: Example2: Sentence1) have different meaning?
Answer: Example2: Answer.

Do (Sentence1: Example2: Sentence1) and (Sentence2: Example3: Sentence1) have different meaning?
Answer: Example3: Answer.

Do (Sentence1: Question: Sentence1) and (Sentence2: Question: Sentence2) have different meaning?
Answer:

Assistant
`Answer`

Table 6: **C**^{3} **prompt on Llama-2 for PAWS-X task.** The answer candidates set would be {Yes, No}, and the embedding set would be {y, n}, where y is the Llama embedding of the word "Yes," and n is the Llama embedding for the word "No."

Algorithm 1: Calibration Cascade Existing

Input: Models $\{M_1, ..., M_n\}$, threshold λ, data x
for $i \leftarrow 1$ **to** n **do**
  // get the logits after temperature scaling
  logits = $M_i(x)$ / temperature(Mi)
  // get the probability for each class label
  Pr = softmax(logits)
  // get the confidence
  confidence = max(Pr)
  if *confidence* > λ *or* i == n **then**
    return logits
  end
end

E Experiment

E.1 Implementation Details

In this section, we introduce the implemtation details of our **C**^{3} method in two different settings: language model cascade (fine-tuning) and large

	Premise	Hypothesis	Label	Model chosen by C^3	Model chosen by Cascade
en	They said we're providing accommodations for your stay.	They're covering housing costs.	Entailment	4 ✓	3 ✗
bg	Така че не съм много сигурен защо.	Не знам защо се е случило това.	Entailment	4 ✓	2 ✗
de	Es gibt so viel was ich darüber erzählen könnte, ich überspringe das einfach.	Ich werde nicht darüber reden, obwohl es viel zu decken gibt.	Entailment	4 ✓	2 ✗
th	เธอกล่าวว่ามีน้ำตาไหลออกมาจากตาของเธอ และเธอกล่าวว่าโจมาปรากฏตัวที่ชานบ้าน	เธอร้องไห้ ที่ได้เจอ Joe เพราะเธอมีความสุขมาก	Neutral	4 ✓	3 ✗
zh	在这个赤裸城市里有许多故事。	我还没听说过这些故事。	Contradiction	3 ✓	2 ✗

Figure 4: Case study of five languages.

Table 7: Dataset we incorparate in our experiment.

Dataset	# Train (English)	# Dev (per langauge)	# Test (per langauge)	Languages
XNLI	393k	2.49k	5.01k	15
PAWS-X	49.4k	2k	2k	7
QAM	100k	10k	10k	3
GSM8k (translated)	-	-	500	6
TabMWP (translated)	-	-	500	6

language model cascade (zero/few-shot learning).

Language Model Cascade We initialize four encoder models from pretrained multilingual DistilBERT, multilingual BERT base, XLM-RoBERTa base, and XLM-RoBERTa large respectively. Each pretrained model is taken from the checkpoint published on Huggingface model hub. We then train each model independently on the English split in the training set for {4, 4, 5, 5} epoches with batch sizes of {32, 32, 16, 16} and learning rates of {3e-5, 2e-5, 1e-5, 1e-5} on 4 Nvidia A6000 GPUs.

Large Language Model Cascade We initialize three casual language models from pretrained Llama-2-7b-chat-hf, Llama-2-13b-chat-hf, and Llama-2-70b-chat-hf published on Hugginface model hub. Without any training, we apply few-shot learning by giving the model three examples from the English split in the prompt.

E.2 Datasets

E.3 Case Study

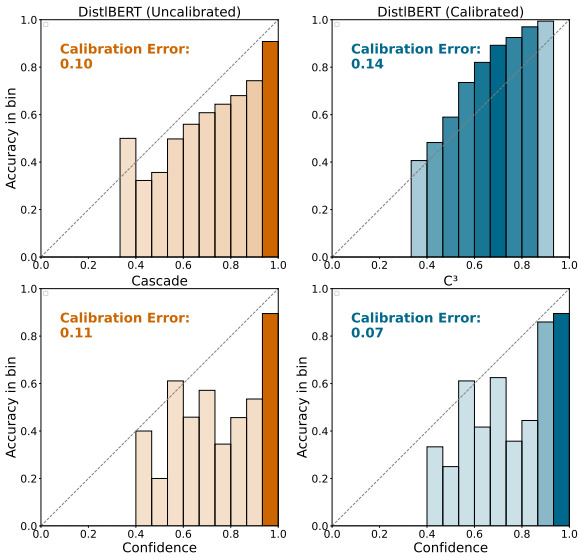


Figure 5: ECE comparison between Cascade and C^3 in English language.

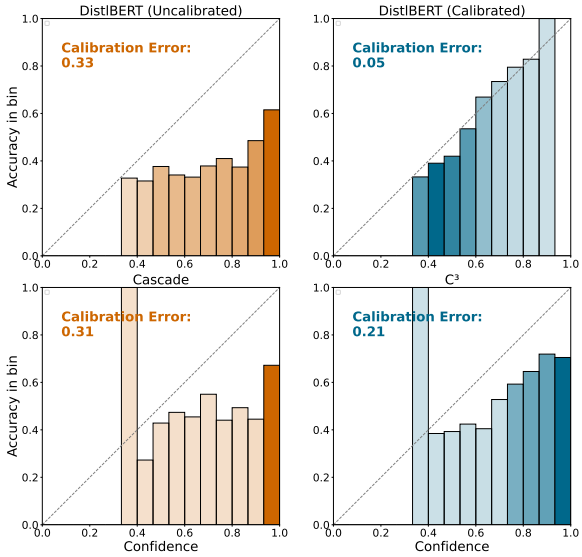


Figure 6: ECE comparison between Cascade and $\mathbf{C}^3$ in Swahili language.