

Appendix

We supplement the main text with additional results that comprise the following details.

- **Pseudocode:** We present an algorithm that describes the step-by-step procedure for fine-tuning vision-language models on the VQA task, incorporating our proposed calibration loss (*AlignCal* loss)
- **Architecture/finetuning procedure:** We briefly describe the architecture of the VLMs used. Additionally, we include the fine-tuning procedure and implementation details.
- **Theoretical insight:** We provide a detailed formulation and a theoretical justification for why the proposed loss function minimizes the expected calibration error.
- **Additional Results** Main text comprised of calibration comparison in Base VLM Gemma 3 4B model, FL fine-tuned Gemma 3 4B model and Agentic Framework results. Supplementary includes calibration performance of different other SOTA VLM Models. We also illustrate here the calibration comparison between FL and FL + *AlignCal* fine-tuned LLaVA Onevision model on ScienceQA and VQARad dataset. We also provide inference time analysis.
- **Ablation studies on number of agents, debate rounds and *AlignCal* loss**
- **Broader Impact**
- **Reproducibility Checklist**

Pseudocode

We provide the pseudo-code of our finetuning procedure in Alg. 1.

Algorithm 1: Fine-tuning VQA Model with Focal + *AlignCal* Loss

Input: $\{(I_i, Q_i, y_i)\}_{i=1}^N$ (images, questions, ground-truth answers), Pre-trained VLM \mathcal{M} with parameters ϕ

Parameter: learning rate η , epochs E , LoRA parameters θ , batch size B , focal parameter γ , *AlignCal* weight α

Output: Calibrated fine-tuned VLM \mathcal{M}'

```

1: Initialize LoRA parameters  $\theta$ , and freeze  $\phi$ 
2: for epoch = 1 to  $E$  do
3:   for each batch  $B \subseteq \mathcal{D}$  do
4:     Compute forward pass to get  $P_{\phi+\theta}(w_i \mid w_{0:i-1})$ 
5:     Compute AlignCal + FL loss  $\mathcal{L}_{\phi+\theta}^{AlignCal + FL}$ 
6:     Compute gradients of loss function w.r.t.  $\theta$ :
        $\nabla_{\theta} \mathcal{L}_{\phi+\theta}^{AlignCal + FL}$ 
7:     Update LoRA parameters:  $\theta \leftarrow \theta - \eta \cdot \nabla_{\theta} \mathcal{L}_{\phi+\theta}^{AlignCal + FL}$ 
8:   end for
9: end for
10:  $\theta^* \leftarrow \theta$ 
11:  $\mathcal{M}' \leftarrow \mathcal{M}$  with updated LoRA parameters  $\theta^*$ 
12: return  $\mathcal{M}'$ 

```

Architectures

We describe below the architectures that we experimented with.

1. **LLaVA-OneVision (Li et al. 2024)** is a Vision-Language Model capable of generating text conditioned on one or more images or videos. It also allows for strong transfer learning across different scenarios (single image, multi image, and video) enhancing the capabilities of the contemporary VLM scene. It combines a SigLIP vision encoder with a Qwen2 language backbone for effective results.
2. **Gemma 3 4B (Team et al. 2025)** is part of Google DeepMind’s Gemma3 family. It utilizes a SigLIP vision encoder with Gemma 4B as the language backbone. It is compact enough for broader deployment and is well-suited for a variety of text generation and image understanding tasks like question answering, summarization, and reasoning.
3. **Qwen/Qwen2.5-VL-3B-Instruct(Bai et al. 2025)** is a 3B-parameter Vision-Language Instruct model from the Qwen/VL series. For the vision encoder, it implements window attention into the ViT, which is then further optimized with SwiGLU and RMSNorm to align it with the Qwen2.5 VL architecture. The language backbone comprises of 3B instruction-tuned variant of the Qwen2.5 LLM. It’s designed for generating structured outputs using it’s proficient reasoning and analysis skills.
4. **Phi-4-multimodal-instruct(Abouelenin et al. 2025)** is a lightweight open-source multimodal transformer model with 5.6B parameters. It processes text, image, and audio inputs to produce enhanced text outputs, which is possible due to supervised fine-tuning, fine-tuning on human preferences, and RLHF. The model architecture is composed of multiple advanced image encoders and the pretrained Phi-4-Mini-Instruct as the backbone language model.

The calibration results for different VLM models mentioned in Table 4 are illustrated in Figures 3 and 4.

Finetuning Procedure and Implementation

In our agentic framework, we employ Qwen2.5-VL-3B-Instruct (Bai et al. 2025), Llava-OneVision (Li et al. 2024), Gemma-3-4B (Team et al. 2025), and Phi-4-Multimodal-Instruct (Abouelenin et al. 2025), each quantized to 4-bit precision using the BitsAndBytes library. We adopt standard chain-of-thought prompting and a knowledge-based agent architecture.

All experiments were conducted on an NVIDIA A100 40 GB GPU. We perform LoRA fine-tuning with rank 8, a scaling factor of 8, and a dropout rate of 0.05. For attention-only adapters, LoRA modules are injected into the `q_proj` and `v_proj` layers. Calibration fine-tuning is run for 6 epochs on the VQARad dataset and 10 epochs on the ScienceQA dataset. We use a batch size of 2 and optimize with the fused AdamW optimizer, starting from a learning rate of 2×10^{-4} . The ScienceQA training set comprises 12 000 examples,

while VQARad contains 1 793 examples. For the calibration loss, we set $\lambda = 2$.

PostHoc Calibration For comparison with post-hoc techniques, we set aside 10% of the training data as a hold out set to conduct post-hoc calibration. For Temperature Scaling (TS), we perform a grid search between the range of 0 to 10 with a step-size of 0.1 to find the optimal temperature value that gives the least NLL on the hold-out set. In Dirichlet Calibration (DC), we append a single-layer neural network to the DNN’s output and impose ODIR (Kull et al. 2019) regularization on that layer’s weights.

Calibration via finetuning We also compared FL + *AlignCal* calibration results with other train-time calibration results.

Label Smoothing (LS) (Szegedy et al. 2016) In LS the hard one-hot targets are replaced with a smoothed distribution q_i , where for each sample i and class j :

$$q_{i,j} = \begin{cases} 1 - \alpha, & j = y_i, \\ \frac{\alpha}{K - 1}, & j \neq y_i, \end{cases}$$

with α a hyperparameter ($\alpha = 0.1$ in our experiments). The label-smoothing loss is then

$$LS = - \sum_{i=1}^N \sum_{j=1}^K q_{i,j} \log \hat{p}_{i,j},$$

where $\hat{p}_{i,j}$ denotes the model’s predicted probability for class j on sample i .

Focal Loss (FL) (Mukhoti et al. 2020) Focal loss down-weights well-classified examples by a factor $(1 - \hat{p}_{i,y_i})^\gamma$. Formally,

$$FL = - \sum_{i=1}^N (1 - \hat{p}_{i,y_i})^\gamma \log \hat{p}_{i,y_i},$$

where γ is the focusing parameter. In the experiments we used γ to be 2.

Additional Results

Figure 5 shows the calibration of the LLaVA-OneVision model fine-tuned with focal loss and with focal loss + *AlignCal*. On the VQARad dataset, ECE decreases from 0.105 (FL fine-tuned) to 0.093 (FL + *AlignCal* fine-tuned). ACE decreases from 0.114 to 0.101, and MCE from 0.338 to 0.305. On the ScienceQA dataset, ECE decreases from 0.0964 (focal loss fine-tuned) to 0.0682 (FL + *AlignCal* fine-tuned). ACE decreases from 0.0919 to 0.0683, and MCE from 0.1604 to 0.0981.

Figure 6 shows the calibration comparisons between training-time calibration methods. The label-smoothing fine-tuned model yields an ECE of 0.2819, an MCE of 0.3962, and an ACE of 0.2419. When using the focal-loss fine-tuned model, calibration performance improves: ECE decreases to 0.178, MCE to 0.430, and ACE to 0.155. The best calibration performance is achieved by the focal loss + *AlignCal* model, with an ECE of 0.137, an MCE of 0.425, and an ACE of 0.115.

Ablation studies

Number of Agents From Table 2 Dropping weaker agents (4→3), improves calibration beyond any individual stage-1 agent, highlighting the debate process’s strength. Among individual agents, the *Knowledge Agent* is least calibrated, while the *CoT Agent* is the strongest single model.

Table 2: Calibration Metrics by Agent Types

Agent	ACE	ECE	MCE
COT Agent	0.1003	0.1002	0.3164
Knowledge Agent	0.1502	0.1616	0.6138
Self Ask Agent (SA)	0.1598	0.1436	0.2529
Search Augmented Agent	0.1140	0.1050	0.3380
3 Agents (Dropped Knowledge)	0.0919	0.0964	0.1604
2 Agents (Dropped Knowledge, SA)	0.1010	0.0930	0.3050

Table 3: Calibration Metrics by λ (*AlignCal* loss)

λ	ACE	ECE	MCE
1	0.1394	0.1404	0.4147
2	0.1150	0.1370	0.4250
3	0.1778	0.1878	0.3875
4	0.1932	0.1971	0.3971

Ablation of *AlignCal* loss (Table 3), λ : $\lambda = 2$ gives the best calibration

Debate rounds. Increasing the number of rounds hurts calibration (ECE 0.146 → 0.1881, ACE 0.133 → 0.1826), so we adopt a single round for efficiency; this pattern is consistent with recent findings on test-time scaling (?)

Inference times: Per question: 40s (VQA-RAD y/n), 120s (ScienceQA MCQ). Two sequential stages. VQA-RAD: 6.35s + 33.65s; ScienceQA: 9.33s + 110.67s (4 agents). Agents ran sequentially due to GPU limits; parallel agents & stage pipelining can boost throughput.

Prompt Template used in Agentic Debate Framework

1. Stance generation

State your answer (as short as possible, in one or a few words), then rate: the level of ambiguity in the input query (a float from 0 to 1); the level of complexity of the input query (a float from 0 to 1); and your level of ability for solving the input query (a float from 0 to 1). Note that your uncertainty about correctness is affected by input ambiguity, task complexity, and your own knowledge and abilities. Based on this, give a float (between 0 and 1) indicating your overall confidence that your answer is correct. Format: Answer: <your short answer> Confidence: <float in [0,1]>

2. Argument generation

You are participating in a debate on the question:

“{QUERY}”. Your assigned stance on the question is: “{STANCE}”. Generate arguments or evidence (no more than three sentences) explaining why your assigned stance is correct. If the question is ambiguous, explicitly state the assumptions or interpretations you adopt. Be concise and exclude anything irrelevant or unhelpful to supporting the stance.

3. Final confidence elicitation

Given the question: “{QUERY}”, your original answer is “{STANCE}” with a confidence score of {ORIGINAL-CONFIDENCE}. An argument from the opposing side is “{ARGUMENTAGAINST}”, which received the following rating and feedback from other deliberators: “{FEEDBACKAGAINST}”. Note that {NUMBER_AGAINST} people disagreed with you.

An argument supporting your original answer is “{ARGUMENT_FOR}”, which received the following rating and feedback from other deliberators: “{FEEDBACKSUPPORTING}”. Note that {NUMBER-SUPPORTING} people agreed with you.

Provide your final answer to the question (as short as possible). Considering your original belief, group consensus, and these new observations—and weighing arguments from multiple sides (including your own)—give a brief rationale for whether you would adjust your original confidence score.

Recall your original confidence is {ORIGINAL-CONFIDENCE}. Given your rationale “{CONFIDENCERATIONALE}”, provide your final confidence score (a float in $[0,1]$) in the exact format: Confidence: <float in $[0,1]$ >.

Broader Impact

The proposed approach to improving calibration in agentic Visual Question Answering (VQA) systems has potential implications across both practical and ethical dimensions. By aligning model confidence more closely with correctness, especially in multi-agent architectures, our work enhances the reliability and trustworthiness of AI systems operating under visual uncertainty. This is particularly valuable in high-stakes domains such as medical diagnostics, autonomous navigation, and assistive technologies, where incorrect yet overconfident predictions can lead to harmful outcomes.

Better-calibrated systems also support more interpretable and human-aligned decision-making, as users can more accurately assess when to rely on model predictions. This contributes to safer human-AI collaboration and accountability, which is increasingly critical as AI systems gain autonomy.

Theoretical Formulation

Formally, given a model’s softmax outputs $\mathbf{p} = (p_1, \dots, p_K)$ derived from logits z_i , the true label y , the top predicted confidence $p_{\max} = \max_i p_i$, and the predicted probability of the ground truth class p_y , we define the **soft-calibration loss** as:

$$\mathcal{L}_{\text{cal}}(p_y, p_{\max}) = p_y(1 - p_{\max}) + (1 - p_y)p_{\max} \quad (3)$$

The full training objective therefore becomes:

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{FL}} + \lambda \mathcal{L}_{\text{cal}},$$

where λ is a tunable hyperparameter and \mathcal{L}_{FL} is the Focal Loss (Mukhoti et al. 2020), which is a strictly proper scoring rule that encourages accuracy and is also found to implicitly enhance calibration. Our calibration term is not an ad-hoc tweak; it arises naturally as a plug-in surrogate to the Upper Bound on Classification Error (UBCE):

$$\begin{aligned} \text{UBCE} &= \Pr(t = 0) \mathbb{E}[p_{\max} | t = 0] \\ &\quad + \Pr(t = 1) (1 - \mathbb{E}[p_{\max} | t = 1]), \end{aligned}$$

where $t = \mathbb{I}\{\hat{y} = y\}$ is an indicator function for whether the prediction is correct and $\Pr(\cdot)$ defines the probability. Equivalently, algebraic regrouping gives us:

$$\text{UBCE} = \mathbb{E}[t(1 - p_{\max}) + (1 - t)p_{\max}].$$

This form makes it clear that UBCE is the expected absolute gap between correctness and confidence – hence a conservative upper bound on calibration metrics. Applying the law of total expectation (tower rule) and conditioning on an input x , we have:

$$\begin{aligned} \text{UBCE} &= \mathbb{E}_x [\mathbb{E}[t(1 - p_{\max}) + (1 - t)p_{\max} | x]] \\ &= \mathbb{E}_x [\mathbb{E}[t | x] (1 - p_{\max}) + \mathbb{E}[1 - t | x] p_{\max}]. \end{aligned}$$

Here the outer expectation is over inputs x drawn from the data distribution, and the inner expectation is over the randomness in the correctness indicator t given x . Since p_{\max} is deterministic conditioned on x , it can be pulled outside the inner expectation.

Let $q(x) := \mathbb{E}[t | x]$ be the true conditional probability of correctness given an input x . Then,

$$\text{UBCE} = \mathbb{E}_x [q(x)(1 - p_{\max}) + (1 - q(x))p_{\max}].$$

In practice, $q(x)$ is unknown because it depends on the non-differentiable indicator function $\mathbb{I}\{\hat{y} = y\}$. To obtain a differentiable surrogate, we use the model’s own soft belief about correctness. Instead of taking $\hat{y} = \arg \max_i p_i$, imagine sampling a label $\hat{y} \sim p(\cdot | x)$ from the model’s softmax output. Then the expectation of the indicator function $\mathbb{E}[t | x] = p_y$; that is, the probability that a randomly drawn label equals the true label is exactly p_y . This makes p_y a natural, smoothed estimator for $q(x)$. This replacement is an instance of the classical plug-in principle (Grunewald 2018). Accordingly, we define our soft surrogate loss per example as:

$$\mathcal{L}_{\text{cal}}(p_y, p_{\max}) = p_y(1 - p_{\max}) + (1 - p_y)p_{\max} \quad (4)$$

so that,

$$\begin{aligned} \mathbb{E}[\mathcal{L}_{\text{cal}}] &= \mathbb{E}_x [p_y(1 - p_{\max}) + (1 - p_y)p_{\max}] \\ &\approx \text{UBCE}, \end{aligned}$$

with equality in expectation under the assumption that $p_y \approx \mathbb{E}[t | x]$. This construction gives a smooth, differentiable surrogate to UBCE that admits gradient-based optimization while preserving the structure of the original bound.

Architecture	ScienceQA							VQARad						
	↑Acc.	↑F1	↑Prec.	↑Rec.	↓ACE	↓ECE	↓MCE	↑Acc.	↑F1	↑Prec.	↑Rec.	↓ACE	↓ECE	↓MCE
Granite-vision-3.3-2b	65%	0.467	0.501	0.442	0.439	0.343	0.75	59.00%	0.542	0.611	0.573	0.310	0.309	0.498
SmolVLM-Instruct	49%	0.325	0.401	0.281	0.296	0.295	0.477	62.4%	0.62	0.623	0.62	0.138	0.123	0.224
InternVL-4B	84%	0.569	0.585	0.557	0.014	<u>0.021</u>	0.109	<u>68.00%</u>	0.674	0.684	<u>0.675</u>	0.682	0.681	0.682
Ristretto-3B	81%	<u>0.652</u>	<u>0.664</u>	<u>0.641</u>	<u>0.017</u>	0.016	<u>0.124</u>	64.50%	0.636	0.648	<u>0.638</u>	0.647	0.645	0.651
Ovis2-4B	65%	0.443	0.441	0.427	0.351	0.351	0.739	69.00%	<u>0.672</u>	0.730	0.682	0.432	0.456	0.69
H2Ovl-mississippi-2b	72%	0.470	0.474	0.47	0.172	1.722	0.6712	65.3%	0.644	<u>0.692</u>	0.665	0.653	0.654	0.656
DeepSeekVL2-Tiny	<u>82%</u>	0.696	0.706	0.699	0.438	0.028	0.1741	59%	0.49	<u>0.602</u>	0.548	<u>0.28</u>	<u>0.128</u>	<u>0.28</u>

Table 4: Comparison of state-of-the-art model performance on the ScienceQA and VQARad datasets. Bold values indicate best performance for each metric within each dataset, underlined values indicate second-best performance.

From a probabilistic calibration perspective, if the model is well-calibrated in the sense that its softmax outputs reflect the true posterior—i.e., $p_y \approx \Pr(y|x)$ —then p_y also approximates $\Pr(\hat{y} = y|x)$, making the plug-in substitution for $q(x)$ increasingly accurate. There is a self-correcting feedback: early in training p_y might poorly estimate $q(x)$, but the calibration loss penalizes discrepancies between p_{\max} and this current belief. Improving those discrepancies tends to make p_y a more honest estimate of correctness, which in turn makes the surrogate tighter.

Gradient Analysis

The gradient of the *AlignCal* loss becomes,

$$\frac{\partial \mathcal{L}_{\text{AlignCal}}}{\partial z_i} = (1 - 2p_y)p_{\hat{y}}(\delta_{i,\hat{y}} - p_i) + (1 - 2p_{\hat{y}})p_y(\delta_{i,y} - p_i),$$

which is based on using p_y as the probabilistic estimate of $\Pr(t|x)$. In the case,

- when there is an overconfident incorrect, i.e., $\hat{y} \neq y$, i.e., $p_{\hat{y}} > 0.5, p_y < 0.5$,

$$\frac{\partial \mathcal{L}_{\text{AlignCal}}}{z_{\hat{y}}} > 0,$$

which causes a push to decrease the value of $z_{\hat{y}}$ in gradient descent. We see that the other gradient becomes,

$$\frac{\partial \mathcal{L}_{\text{AlignCal}}}{z_y} < 0,$$

which causes a push to increase the value of z_y and therefore increasing p_y . This combined with the cross-entropy like loss brings about rapid error correction.

- when we have a confident correct prediction, i.e., $\hat{y} = y$ and $p_y > 0.5$, we have

$$\frac{\partial \mathcal{L}_{\text{AlignCal}}}{z_y} < 0,$$

which again pushes the model to be more confident in the correct predictions.

- when we have an underconfident correct, i.e., $p_{\hat{y}} = p_y$ and $p_y < 0.5$, we rely on using p_y as the estimate of probability of correctness. Here the calibration loss acts

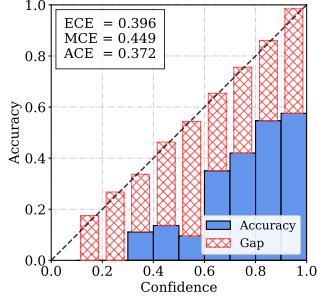
to dampen the probabilities since we don't want to output incorrect answers with high probabilities, i.e.,

$$\frac{\partial \mathcal{L}_{\text{AlignCal}}}{z_y} > 0,$$

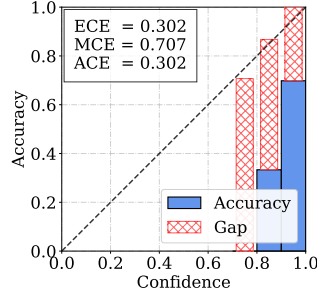
which pushes to decrease z_y with gradient descent. However, when we combine this loss with our CE-like loss (focal loss), the CE gradient pulls p_y to higher values. This interplay causes the final prediction to move towards making $p_y \rightarrow 1$, but in a controlled manner.

- when we have a low confidence incorrect answer $p_{\hat{y}} \neq p_y$ and $p_y < 0.5, p_{\hat{y}} < 0.5$, we operate in a similar fashion to the previous case and dampen the probabilities of both $p_{\hat{y}}$ and p_y . However, only p_y gets an additional pull with the CE loss here, driving that value higher towards 1.

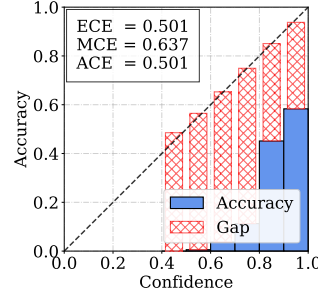
Therefore, from the gradient analysis, it is clear that our loss acts to improve calibration by sharpening probabilistically correct predictions and dampening cases when the model is likely to be incorrect. As the model's beliefs improve over training, it becomes representative of $q(x)$.



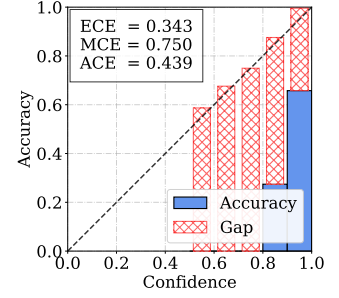
(a) Gemma-3-4B



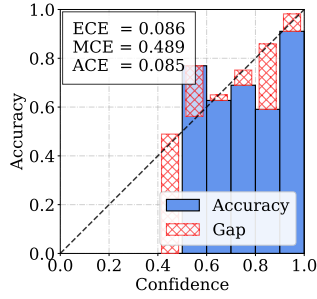
(b) Qwen2.5-VL-3B-Instruct



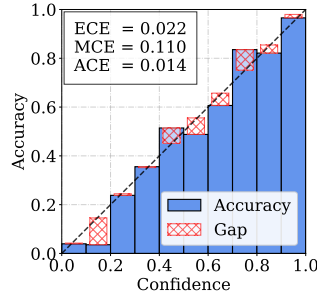
(c) LLAVA-OneVision



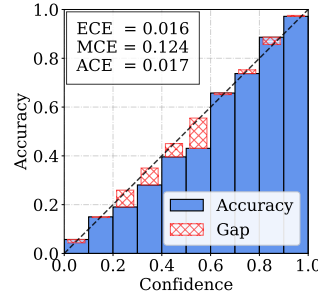
(d) Granite-vision-3.3-2B



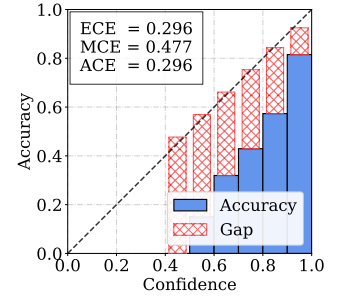
(e) Phi-4-Multimodal-Instruct



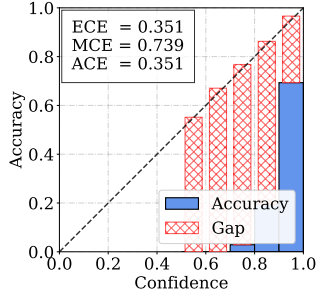
(f) InternVL-4B



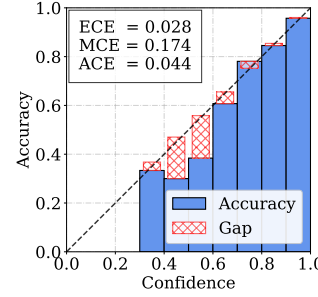
(g) Ristretto-3B



(h) SmolVLM



(i) Ovis2-4B



(j) DeepSeekVL2-Tiny

Figure 3: Reliability plots for base VLMs on ScienceQA for (a) Gemma-3-4B, (b) Qwen2.5-VL-3B-Instruct, (c) LLAVA-OneVision, (d) Granite-vision-3.3-2B, (e) Phi-4-Multimodal-Instruct, (f) InternVL-4B, (g) Ristretto-3B, (h) SmolVLM, (i) Ovis2-4B, and (j) DeepSeekVL2-Tiny. We observe that majority of the base VLM models are overconfident except for 3f, 3g and 3j.

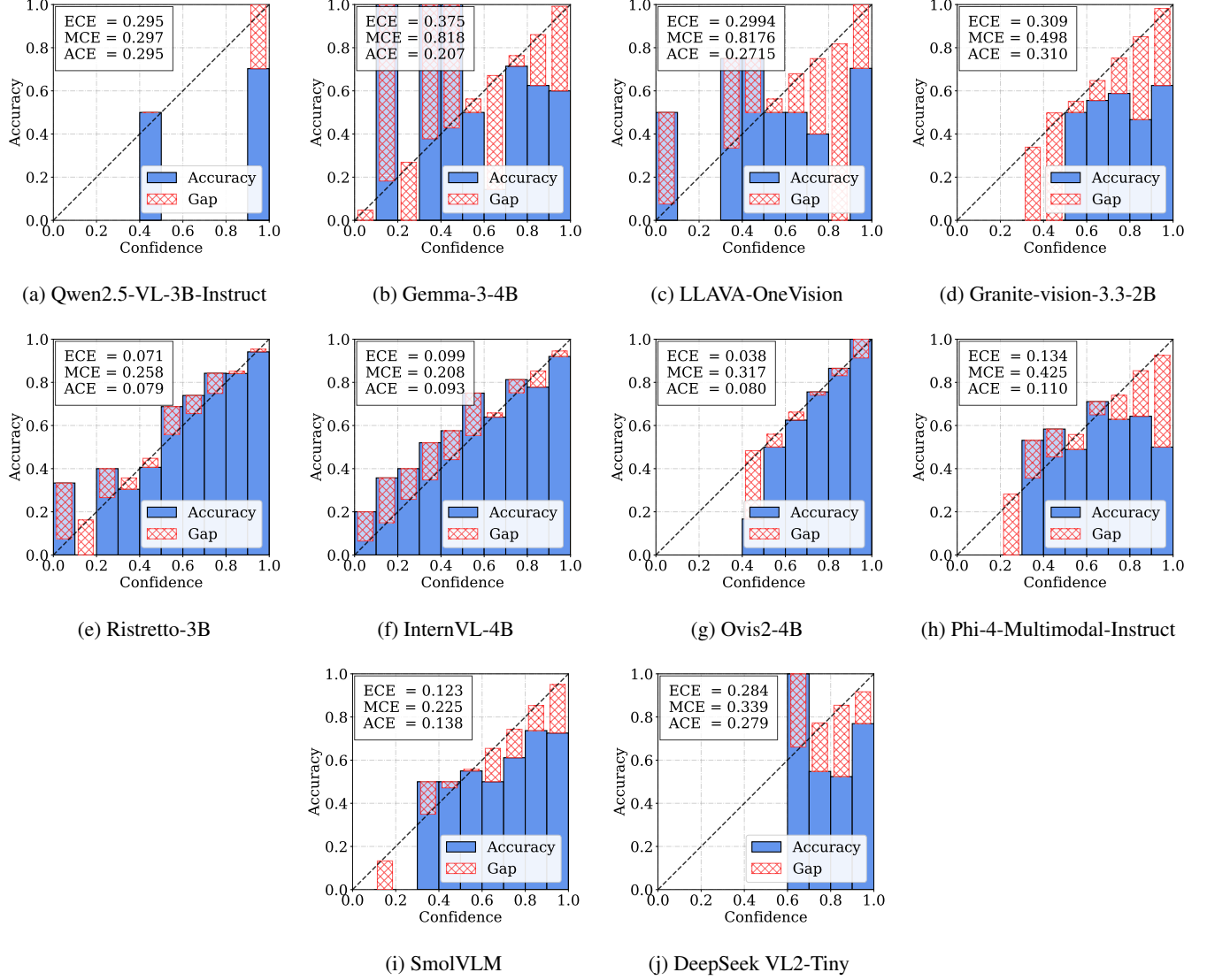


Figure 4: Reliability plots for base VLMs on VQARad for (a) Qwen2.5-VL-3B-Instruct, (b) Gemma-3-4B, (c) LLAVA-OneVision, (d) Granite-vision-3.3-2B, (e) Ristretto-3B, (f) InternVL-4B, (g) Ovis2-4B, (h) Phi-4-Multimodal-Instruct, (i) SmolVLM, and (j) DeepSeek VL2-Tiny. We observe that majority of the base VLM models are either overconfident or underconfident except for 4f and 4g on the VQARad dataset.

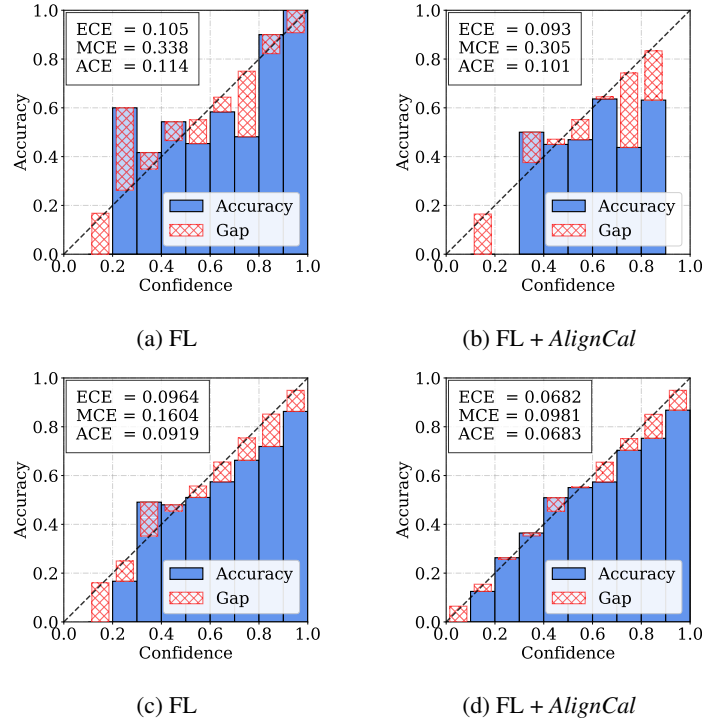


Figure 5: Reliability plot comparison between (5a) FL and (5b) FL + *AlignCal* finetuned LLAVA-OneVision model on VQARad dataset, (5c) FL and (5d) FL + *AlignCal* finetuned LLAVA-OneVision model on ScienceQA dataset. Our proposed loss FL + *AlignCal* is effective at improving ECE from 10.5% to 9.3% on the VQARad dataset and a reduction from 9.64% to 6.8% on the ScienceQA dataset. MCE and ACE also follows a similar decreasing trend.

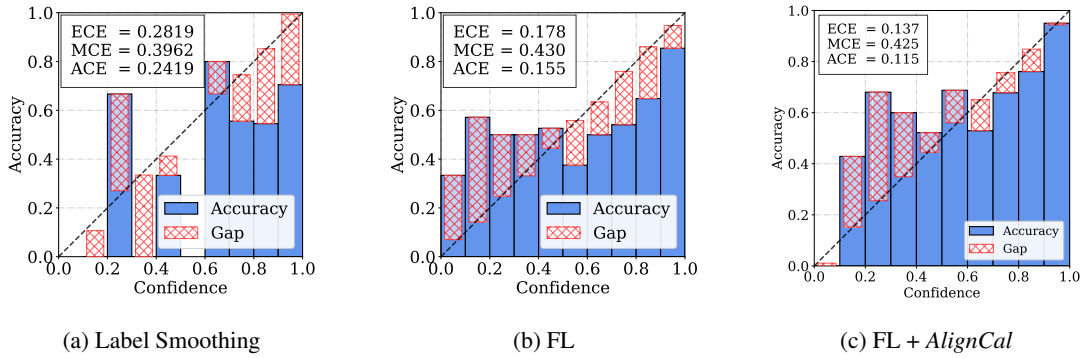


Figure 6: Reliability plot comparison between (6a) Label Smoothing , (6b) FL , (6c) FL + *AlignCal* finetuned Gemma 3 4B on VQARad dataset. Our proposed loss FL + *AlignCal* is effective at improving ECE from LS to FL + *AlignCal* by 15%. It further shows a improvement of 4% from FL to FL + *AlignCal* on the VQARad dataset. MCE and ACE also follow a similar decreasing trend.