CARAVAN: ASYNCHRONOUS TEST-TIME ADAPTATION FOR FASTER INFERENCE

Jiayin Kralik[1] **Ethan Zhao**[1] **Anrui Liu**[1]
Reto Achermann[1*] **Ivan Beschastnikh**[1*] **Evan Shelhamer**[1,2*]
University of British Columbia[1] Vector Institute[2]

ABSTRACT

Test-time adaptation (TTA) updates a model online during deployment to improve robustness to distribution shifts. While TTA updates give robustness, they take time: the update computation during inference makes deployment impractical for latency-sensitive systems. We present **Caravan**, an asynchronous TTA framework that decouples inference from update computation. Caravan maintains three concurrent streams that run on a *single GPU*: a high-priority inference stream and two low-priority streams for computing updates. Because updates necessarily lag behind inference, Caravan revisits sample selection to only update the normalization-layer affine parameters and running statistics after (i) entropy filtering to retain reliable samples and (ii) gradient-consistency filtering of per-sample entropy gradients w.r.t. the last normalization layer to filter conflicting updates. Caravan improves latency by up to $6.8\times$ and accuracy by 1.99% over synchronous TTA methods on ImageNet-C with ResNet50-BN.

1 INTRODUCTION

Deep networks can be highly accurate in-distribution, yet brittle under *distribution shift* (Quionero-Candela et al., 2009), where the data distribution at deployment differs from the one during training. Test-time adaptation (TTA) addresses this problem by updating a deployed model online using unlabeled test data (Liang et al., 2024; Schneider et al., 2020; Wang et al., 2020; Niu et al., 2022; 2023). These TTA methods, and many others, restrict adaptation to normalization parameters and statistics to achieve more robust, lightweight updates. However, the updates still take time.

Unfortunately, *deployment* remains challenging in latency-sensitive use cases due to these inference-time updates. For example, in Tent, the backward pass used to compute updates accounts for more than 57% of the total inference latency. Moreover, common untimed TTA evaluations overestimate real-world performance: under a fixed input rate, slower methods drastically underperform (Alfarra et al., 2023). Thus, the key questions are not only *how much* adaptation can help, but *how* to obtain its benefits under strict latency budgets.

We present **Caravan**, an asynchronous TTA framework that decouples inference from adaptation. Caravan uses a high-priority computation stream for latency-sensitive inference and two low-priority streams to compute model updates and sample selection in the background. All three streams execute concurrently on a single GPU. Caravan shares model parameters across computation streams to improve memory efficiency.

After an adaptation step is completed, Caravan publishes the updated normalization layers to the inference stream without blocking it. By decoupling adaptation *and* sample selection from inference, Caravan can implement more sophisticated sample selection algorithms without adding latency overhead to the inference path. We select samples using entropy filtering (Niu et al., 2022; 2023) and our novel gradient consistency filtering.

We evaluate the latency and accuracy of Caravan applied to Tent, SAR, and EATA adaptation on the standard corruption benchmark of ImageNet-C (Hendrycks & Dietterich, 2019) under realistic online execution. With the common ResNet-50 model He et al. (2016), Caravan achieves a latency

*Equal advising.

improvement of $6.8\times$ over synchronous TTA methods, coming within 14.7% of inference-only latency. We further show that our sample selection algorithm results in a 1.99% improvement in accuracy over synchronous TTA methods at negligible cost to inference latency.

We contribute our new system, empirical results, and a novel technique for sample selection:

- We introduce **Caravan** for asynchronous overlap of inference and adaptation on a single GPU.
- We show Caravan improves latency and accuracy for multiple TTA methods on corruption shifts.
- We show Caravan's architecture enables sample selection for improved adaptation.

2 Related Work

Efficient TTA. Several approaches seek to reduce update cost or frequency. Forward-only adaptation (FOA) removes backpropagation by optimizing a small set of added parameters using derivative-free methods (Niu et al., 2024). SNAP (Cha et al., 2025) reduces the adaptation rate using sparse updates and employs inference-time normalization correction to maintain accuracy. CoLA (Chen et al., 2024) distributes TTA across multiple devices to parallelize inference and updates. Our work focuses on the single-GPU, fully TTA setting, addressing its key practical limitation: the latency overhead of online adaptation. By decoupling inference from adaptation, Caravan lets adaptation to proceed in the background without blocking inference. Caravan handles the forward and backward passes of existing TTA methods without altering their implementation.

System Optimizations. GPU inference systems target conventional deployment without adaptation and often include small batch sizes, resulting in substantial GPU underutilization (Gao et al., 2024; Elvinger et al., 2025). Fine-grained GPU sharing increases utilization through concurrent execution, focusing on fast context switching (Bai et al., 2020), microsecond-scale preemption (Han et al., 2022), interference-aware scheduling (Strati et al., 2024), and rapid GPU memory handover (Wang et al., 2025). These systems primarily address inter-job concurrency of independent workloads. In contrast, TTA's inference and adaptation depend on one another. Caravan overlaps inference and adaptation of a single model, allowing adaptation to utilize otherwise idle GPU resources while preserving low-latency inference.

Sample Selection for TTA. By default, updates are computed on every sample of every batch. To improve robustness and efficiency, TTA methods select samples to reduce noisy or redundant updates. For example, EATA selects low-entropy and non-redundant samples (Niu et al., 2022), SAR selects samples that induce stable gradients (Niu et al., 2023), and SNAP selects based on a memory of past inputs to lower update frequency (Cha et al., 2025). We contribute a new sample selection algorithm (Section 3.3) without adding inference latency due to our asynchronous design.

3 Method: Asynchronous Execution of Inference and Updates

3.1 System Overview

Caravan is an asynchronous test-time adaptation (TTA) framework that decouples prediction from updates for low-latency deployment. Caravan maintains two pre-trained model replicas on a single GPU: an *inference replica* for immediate predictions and an *adaptation replica* for background updates. Following prior TTA methods (Schneider et al., 2020; Wang et al., 2020; Niu et al., 2022; 2023), Caravan's adaptation is restricted to normalization layers (e.g., BN for ResNet), updating only their affine parameters and running statistics. To reduce memory usage, the two replicas share unadapted parameters but maintain separate normalization states to avoid interference.

In deployment, each test input is first processed by the inference replica to produce predictions. After the forward pass, and without blocking, Caravan selects samples and enqueues them into a GPU buffer. When the buffer reaches the adaptation batch size, the adaptation replica performs a forward-backward update step on the buffered samples. After each update, it snapshots the updated normalization affine parameters and running statistics, and then records a CUDA event. Before each inference forward pass, the inference replica polls for a newer parameter snapshot and applies the updates only after the corresponding event has completed; otherwise, it proceeds without updating.

3.2 CUDA STREAMS AND EXECUTION MODEL

Caravan explicitly exploits GPU concurrency with three CUDA streams: a high-priority **inference stream** (prediction forward passes), a low-priority **selection stream** (filtering and buffering), and a low-priority **adaptation stream** (forward/backward/optimizer). Inference launches first on the high-priority stream and records a completion event. Sample selection is scheduled on a separate stream after inference completes, ensuring it does not delay prediction. Adaptation begins only when a full batch of buffered samples is available and executes entirely on the adaptation stream. Stream priorities ensure that ready inference kernels are preferred when resources are contending, preserving low inference latency while using idle GPU time for background adaptation.

3.3 SAMPLE SELECTION

Because adaptation (needing forward and backward) is slower than inference, only a subset of test inputs can contribute to each update. Caravan uses a two-stage selection policy that favors samples that are both reliable and directionally consistent.

Entropy filter. We select samples with confident predictions. We compute the prediction entropy $H(p_\theta(y|x))$ from the inference output and keep samples with $H < \tau_H$; high-entropy samples are excluded because they are more likely to yield noisy gradients (Niu et al., 2022; 2023). This step is lightweight and runs asynchronously on the selection stream.

Gradient-consistency filter. We select samples with similar gradients. For each sample, we compute a low-overhead proxy gradient: the gradient of the entropy objective with respect to the affine parameters of the *final* normalization layer. Let g_i be the per-sample proxy gradient; we normalize each gradient and estimate a dominant direction by taking the mean of the batch: $\bar{g} = \frac{1}{m}\sum_i \frac{g_i}{\|g_i\|}$. We retain sample i iff its cosine similarity $\cos(g_i, \bar{g}) > \tau_G$. Gradient-consistency filtering and updates run exclusively on the adaptation stream, inducing no overhead on the inference path.

The selected samples drive both gradient and statistics updates. These statistics use an exponential moving average as in Schneider et al. (2020), enabling accumulation across many small batches.

4 EXPERIMENTS: ACCURACY AND LATENCY

We evaluate Caravan on robustness to image corruptions and answer:

1. **Accuracy/Latency:** Can Caravan improve accuracy and latency (§ 4.1)?
2. **Feasibility Study:** When can asynchrony lead to better latency and accuracy (§ 4.2)?
3. **Upper bound:** What accuracy is achievable without latency constraints (§ 4.3)?

Experimental Setup We adapt ResNet50-BN (He et al., 2016) on ImageNet-C (Hendrycks & Dietterich, 2019) following the standard *episodic* setting of adapting online to each shift then resetting. We report averages (top-1 accuracy and inference latency) across 15 corruption types, averaged over all five severity levels. We compare against no adaptation and the TTA methods Tent, ETA, EATA, SAR, and SNAP using a batch size of 16. For Tent, ETA, EATA, and SAR, we evaluate Caravan editions that maintain the *same* loss and trainable parameters for updates but run adaptation asynchronously. We measure the latency of the inference forward pass of one batch, averaged over multiple measurements on a single NVIDIA L40S GPU. See § A.1 for hyperparameter details.

4.1 ACCURACY–LATENCY EVALUATION

We evaluate the latency and accuracy of Caravan and compare it against other methods on ImageNet-C for ResNet-50 with a batch size of 16. For SNAP, we select adaptation rates of 10% and 90% of processed samples. Table 1 shows the results. We observe that without adaptation (No Adapt.) inference is fastest (9.23 ms) but has limited robustness (39.43%). Synchronous TTA substantially improves accuracy (52.62–56.51%), at the cost of up to $7.8\times$ increased inference latency. Applying Caravan to existing TTA methods results in significant latency improvements (down to 10.59–11.57 ms). However, we also observe slight reductions in accuracy with the methods' default sample selection techniques. As adaptation cannot keep up with inference, there is a delay in applying

Table 1: ImageNet-C accuracy (%) by corruption (columns) and method (rows) for ResNet-50 on an NVIDIA L40S GPU. Caravan (ours) is Tent + Caravan + our sample selection. Δ(Acc.) is the change in mean accuracy vs. baseline. † marks the change over the best-performing baseline (EATA). The final column reports average latency (ms) across all corruptions. Best values in bold.

Method	Gau.	Shot	Imp.	Def.	Gla.	Mot.	Zoom	Snow	Fro.	Fog	Brit.	Cont.	Elas.	Pix.	JPEG	Avg.	Δ(Acc.)	Lat.
No Adapt.	31.30	28.90	25.97	38.60	26.61	38.57	36.06	32.24	37.92	45.39	67.96	38.75	44.76	44.76	53.68	39.43	–	**9.23**
batch size = 64:																		
EATA	55.14	55.11	53.97	51.78	51.58	60.11	59.32	57.86	54.67	66.27	71.47	63.47	64.26	67.27	64.45	59.78	–	105.01
ETA	55.04	55.09	53.91	51.72	51.59	59.93	59.31	57.72	54.54	66.33	71.34	63.40	64.25	67.30	64.27	59.72	–	80.93
SAR	57.62	57.26	56.17	45.89	42.83	55.21	48.82	54.54	56.34	62.98	75.47	65.74	52.65	63.98	65.47	57.40	–	81.13
Tent	52.10	51.52	50.35	48.35	48.04	57.09	57.31	54.37	51.71	65.09	71.43	58.44	63.19	66.21	63.38	57.24	–	78.48
batch size = 16:																		
SNAP (10)	46.85	45.74	44.73	42.56	41.95	51.42	52.19	48.35	47.94	61.26	69.55	55.99	58.96	62.41	59.37	52.62	–	52.65
EATA	51.79	51.82	50.66	47.56	47.56	56.83	55.76	55.11	51.53	63.51	68.74	60.24	61.11	64.28	61.12	56.51	–	53.43
+ Caravan	51.23	50.82	49.91	47.41	46.82	55.74	55.87	53.89	51.12	63.36	68.72	56.72	61.02	64.24	61.35	55.88	-0.63	11.57
ETA	50.97	51.44	50.36	46.72	46.82	56.25	55.30	54.60	51.18	63.27	68.37	59.05	60.56	64.02	60.77	55.98	–	20.67
+ Caravan	51.16	51.02	49.96	47.14	46.82	55.64	55.46	53.99	50.99	63.24	68.44	56.62	60.63	64.01	61.01	55.74	-0.24	11.08
SAR	49.53	48.14	48.25	45.10	45.34	54.93	54.53	52.53	49.83	62.70	68.83	58.51	60.44	63.83	60.60	54.87	–	71.78
+ Caravan	48.65	48.05	46.88	43.51	43.26	52.58	53.69	50.34	48.75	61.79	68.66	53.99	59.64	62.95	60.16	53.53	-1.35	10.91
Tent	49.51	49.06	48.04	45.42	45.06	54.58	54.14	52.15	48.71	62.46	68.60	54.65	60.22	63.54	60.27	54.43	–	19.24
+ Caravan	49.89	49.51	48.33	45.74	45.39	54.50	54.61	52.12	48.95	62.63	68.62	52.43	60.38	63.70	60.80	54.51	+0.08	11.15
+ Caravan (ours)	**53.45**	**53.10**	**52.07**	**49.79**	**48.88**	**58.28**	**58.56**	**55.93**	**53.41**	**65.93**	**72.08**	**60.52**	**64.00**	**67.07**	**64.47**	**58.50**	+1.99†	10.59

updates to the inference model. Notice that Tent + Caravan achieves higher accuracy with fewer updates, suggesting that even adapting on a randomly selected subset of samples can benefit from asynchronous execution. Caravan (ours) extends Tent + Caravan with the sample selection algorithm of § 3.3, which selects *confident and consistent* samples. It achieves the best accuracy across all corruptions, improving the mean to 58.50%, and exceeds the best baseline (EATA) by 1.99 points while maintaining an average latency of only 10.59 ms.

4.2 Feasibility Study: When is Caravan Possible?

To evaluate Caravan's sensitivity to batch size (i.e., how well it can overlap inference and adaptation), we measure the effects of varying inference and adaptation batch sizes on accuracy and latency. We run Tent + Caravan on ImageNet-C with ResNet-50 at inference and adaptation batch sizes ranging from 1 to 64. We observe that accuracy is primarily controlled by the adaptation batch size and independent of the test batch size, while latency is primarily controlled by the test batch size and mostly independent of the adaptation batch size. A combination of (1, 64) for the test and adaptation batch sizes yields good results for latency and accuracy. See Figure 1 in § A.3 for the full results across batch sizes.

4.3 Best Achievable Accuracy

We report the best achievable accuracy by evaluating TTA methods under their original, unconstrained settings, using a batch size of 64. All methods perform worse at batch size 16 compared to a batch size of 64, as shown in Table 1 (top). Caravan targets the complementary deployment setting, where inference operates at small batch sizes under strict latency constraints. Despite this constraint, Caravan extracts a reliable adaptation signal from a limited number of background updates and attains 58.50% mean accuracy (Table 1), which is within 1.28 percentage points of the best unconstrained baseline (EATA at 59.78%), while operating at near–no-adaptation latency. However, further analysis suggests that the gains come from updating normalization statistics with an exponential moving average (EMA), rather than relying on per-batch test statistics.

5 Conclusion

We introduce Caravan, an asynchronous TTA framework that decouples inference from adaptation and sample selection using high-priority and low-priority execution streams to compute updates off the critical path. Its design enables more options for sample selection without adding extra computation or blocking to the inference path. We demonstrate that this approach not only significantly improves inference latency but also increases accuracy.

ACKNOWLEDGMENTS

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants (RGPIN-2025-06826, RGPIN-2025-04203, and RGPIN-2025-06878) and the Canada Foundation of Innovation (CFI). Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. ES is supported by a Canada CIFAR AI Chair. We thank William Bowman for his valuable input.

REFERENCES

Motasem Alfarra, Hani Itani, Alejandro Pardo, Shyma Alhuwaider, Merey Ramazanova, Juan C Pérez, Zhipeng Cai, Matthias Müller, and Bernard Ghanem. Evaluation of test-time adaptation under computational time constraints. *arXiv preprint arXiv:2304.04795*, 2023.

Zhihao Bai, Zhen Zhang, Yibo Zhu, and Xin Jin. PipeSwitch: Fast pipelined context switching for deep learning applications. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 499–514. USENIX Association, November 2020. ISBN 978-1-939133-19-9. URL https://www.usenix.org/conference/osdi20/presentation/bai.

Hyeongheon Cha, Dong Min Kim, Hye Won Chung, Taesik Gong, and Sung-Ju Lee. SNAP: Low-latency test-time adaptation with sparse updates. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=8JwMjKDppz.

Guohao Chen, Shuaicheng Niu, Deyu Chen, Shuhai Zhang, Changsheng Li, Yuanqing Li, and Mingkui Tan. Cross-device collaborative test-time adaptation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=YyMiO0DWmI.

Paul Elvinger, Foteini Strati, Natalie Enright Jerger, and Ana Klimovic. Measuring gpu utilization one level deeper. *arXiv preprint arXiv:2501.16909*, 2025.

Yanjie Gao, Yichen He, Xinze Li, Bo Zhao, Haoxiang Lin, Yoyo Liang, Jing Zhong, Hongyu Zhang, Jingzhou Wang, Yonghua Zeng, Keli Gui, Jie Tong, and Mao Yang. An empirical study on low gpu utilization of deep learning jobs. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400702174. doi: 10.1145/3597503.3639232. URL https://doi.org/10.1145/3597503.3639232.

Mingcong Han, Hanze Zhang, Rong Chen, and Haibo Chen. Microsecond-scale preemption for concurrent GPU-accelerated DNN inferences. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 539–558, Carlsbad, CA, July 2022. USENIX Association. ISBN 978-1-939133-28-1. URL https://www.usenix.org/conference/osdi22/presentation/han.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, June 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, July 2024. ISSN 1573-1405. doi: 10.1007/s11263-024-02181-w. URL http://dx.doi.org/10.1007/s11263-024-02181-w.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pp. 16888–16905. PMLR, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *arXiv preprint arXiv:2302.12400*, 2023.

Shuaicheng Niu, Chunyan Miao, Guohao Chen, Pengcheng Wu, and Peilin Zhao. Test-time model adaptation with only forward passes. *arXiv preprint arXiv:2404.01650*, 2024.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. MIT Press, Cambridge, MA, USA, 2009.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33:11539–11551, 2020.

Foteini Strati, Xianzhe Ma, and Ana Klimovic. Orion: Interference-aware, fine-grained gpu sharing for ml applications. In *Proceedings of the Nineteenth European Conference on Computer Systems*, EuroSys '24, pp. 1075–1092, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704376. doi: 10.1145/3627703.3629578. URL https://doi.org/10.1145/3627703.3629578.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Jiali Wang, Yankui Wang, Mingcong Han, and Rong Chen. Colocating ml inference and training with fast gpu memory handover. In *Proceedings of the 2025 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '25, USA, 2025. USENIX Association. ISBN 978-1-939133-48-9.

A Appendix

A.1 Hyperparameters

Caravan applies an entropy threshold of $\texttt{E\_MARGIN} = 0.4 \ln(1000)$, matching the settings used in EATA and SAR (Niu et al., 2022; 2023). For gradient-consistency filtering, we use a cosine similarity threshold of $\texttt{GRAD\_SIM\_TH} = 0.1$. BatchNorm running statistics are updated using an exponential moving average with momentum 0.05. All methods use SGD with momentum 0.9, learning rate 1.25×10^{-4}, and batch size 16 except where otherwise noted.

A.2 Caravan Sample Selection (with Equations)

In Caravan, inference executes in a high-priority stream to maintain the latency of the critical path for model predictions, while adaptation runs continuously in the background in a lower-priority stream. Because adaptation involves both forward and backward passes, it proceeds at a lower rate than inference. Assuming a constant rate of inputs during deployment, it is therefore not possible to adapt on all inputs. As a result, only a subset of test inputs can contribute to adaptation, underlining the role for sample selection for effective updates.

Caravan employs a two-stage sample selection policy that balances reliability and efficiency.

Stage 1: Entropy-based filtering. As in prior fully test-time adaptation methods (Niu et al., 2022; 2023), Caravan first applies *entropy-based filtering* to identify reliable samples. Given logits $z_i \in \mathbb{R}^C$ and predictive distribution $p_i = \mathrm{softmax}(z_i)$, the predictive entropy is

$$\mathcal{H}(p_i) = -\sum_{c=1}^{C} p_{i,c} \log p_{i,c}. \tag{1}$$

Samples with entropy below a threshold τ_e are considered eligible for adaptation: $\mathcal{R} = \{i \mid \mathcal{H}(p_i) < \tau_e\}$. This step is lightweight and is executed asynchronously on the low-priority selection stream after inference completes, ensuring that it does not interfere with inference on the critical path.

Stage 2: Gradient-consistency filtering. Entropy filtering removes high-uncertainty samples, but the remaining candidates can still induce *conflicting* adaptation signals (e.g., gradients that point in opposite directions), which can cancel out or destabilize updates. Caravan therefore applies a second-stage filter that keeps only candidates whose entropy gradients *align* with the dominant (consensus) gradient direction of the candidate set.

Let ϕ_{last} denote the affine parameters (scale and bias) of the final normalization layer. For each candidate $i \in \mathcal{R}$, we compute the per-sample entropy gradient direction with respect to ϕ_{last} and measure its cosine similarity to the batch consensus direction:

$$g_i = \nabla_{\phi_{\text{last}}} \mathcal{H}(p_i), \tag{2a}$$

$$\tilde{g}_i = \frac{g_i}{\|g_i\|_2 + \epsilon}, \tag{2b}$$

$$\bar{g} = \frac{1}{|\mathcal{R}|} \sum_{j \in \mathcal{R}} \tilde{g}_j, \tag{2c}$$

$$\hat{g} = \frac{\bar{g}}{\|\bar{g}\|_2 + \epsilon}, \tag{2d}$$

$$s_i = \tilde{g}_i^\top \hat{g}, \tag{2e}$$

$$\mathcal{S} = \{\, i \in \mathcal{R} \mid s_i \geq \tau_g \,\}. \tag{2f}$$

Eq. equation 2e is a *cosine similarity* because both $\tilde{g}_i$ (Eq. equation 2b) and $\hat{g}$ (Eq. equation 2d) are ℓ_2-normalized (up to ϵ), so $s_i \in [-1, 1]$ measures the alignment between sample i's gradient direction and the consensus direction. We keep samples with $s_i \geq \tau_g$ (Eq. equation 2f). If $|\mathcal{S}|$ falls below a minimum keep count $k_{\min}$, we instead retain the top-$k_{\min}$ samples ranked by s_i to ensure each update has sufficient support.

To minimize overhead, the gradients in Eq. equation 2 are computed only with respect to ϕ_{last} (a small parameter set). After selection, the adaptation update still applies to the full set of adaptable normalization parameters.

Implementation detail. We compute the selection scores s_i without updating running statistics (i.e., with frozen normalization momentum), so that only the finally selected samples in $\mathcal{S}$ influence the normalization state.

EMA updates of running statistics. Unlike prior methods that rely directly on per-batch test statistics (Wang et al., 2020; Niu et al., 2022; 2023), Caravan updates normalization running statistics using an exponential moving average (EMA) driven by the selected adaptation batches. For normalization layers with running statistics (e. g., batch normalization), let $\mu_\ell^{(t)}$ and $(\sigma^2)_\ell^{(t)}$ denote the running mean and variance of layer ℓ at adaptation step t, and let $\hat{\mu}_\ell(\mathcal{B}_t)$ and $\widehat{\sigma^2}_\ell(\mathcal{B}_t)$ be the batch statistics computed on the selected batch $\mathcal{B}_t$. Caravan applies

$$\mu_\ell^{(t+1)} = (1 - \alpha)\, \mu_\ell^{(t)} + \alpha\, \hat{\mu}_\ell(\mathcal{B}_t), \tag{3}$$

$$(\sigma^2)_\ell^{(t+1)} = (1 - \alpha)\, (\sigma^2)_\ell^{(t)} + \alpha\, \widehat{\sigma^2}_\ell(\mathcal{B}_t), \tag{4}$$

where $\alpha \in (0, 1]$ is the EMA rate (implemented via the normalization momentum). By filtering out samples with conflicting gradients, this approach encourages a more consistent update direction over time, even when each adaptation step is driven by relatively few samples.

Importantly, gradient-consistency filtering is applied exclusively on the adaptation stream and therefore does not introduce synchronization or overhead on the inference path.

A.3 Feasibility Study across Batch Sizes

Here we report (Figure 1) the full span of inference and adaptation batch sizes (from 1 to 64) to identify when asynchrony is most effective.

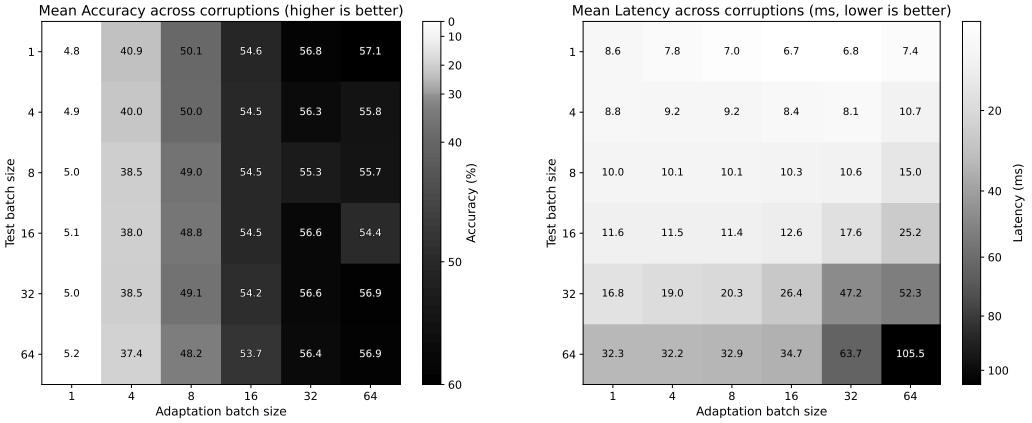

Figure 1: **Scope of asynchronous concurrency.** Mean top-1 accuracy (left) and mean inference latency (right) over the grid of inference and adaptation batch sizes for Tent+Caravan. Accuracy improves with larger batch size, but not monotonically: larger batches imply fewer updates. Latency improves with smaller batch size, and concurrency relies on the sum of inference and adaptation batch sizes being sufficiently small (note the increase to the bottom and right).