EXPLORATION INTO GRADIENT-BASED CORESET METHODS FOR TARGETED SUBSET SELECTION

Evelyn Zhu, Neha Hulkund, Sara Beery
Massachusetts Institute of Technology
{evelynz,nhulkund,beery}@mit.edu

ABSTRACT

In real-world machine learning applications (e.g. detecting broken bones in x-rays), models are deployed in specific settings (e.g. a particular hospital), rather than the domain broadly. Discrepancies between training and deployment distributions lead to suboptimal performance, highlighting the need to curate training data for *fine-tuned foundation models for specific deployment needs*. In this work, we propose a novel algorithm called `Grad-Match-ACF` and evaluate its performance against traditional coreset methods for targeted data subset selection to fine-tune specialized vision models. While traditional coreset methods aim to approximate the training distribution, `Grad-Match-ACF` reformulates coreset selection for out-of-distribution targets by explicitly aligning the coreset budget with the label class frequencies of the deployment. We demonstrate that `Grad-Match-ACF` performs the best across most deployments on the DataSˆ3 benchmark. Beyond better aligning with the objective of targeted subset selection, `Grad-Match-ACF` achieves up to a 18x speed-up improvement over state-of-the-art gradient-matching coreset methods for real-world, long-tailed deployments. While `Grad-Match-ACF` shows more reliability than traditional gradient-matching coreset methods for selecting on large datasets, there is still room for improving the scalability of these methods for targeted subset selection.

1 INTRODUCTION

Modern foundation models have demonstrated remarkable general-purpose capabilities across a wide array of vision tasks (Radford et al., 2021; Oquab et al., 2023). However, specialized models trained on domain-specific data still outperform generalist models (Yang et al., 2025; Bai et al., 2024). Effective specialization strategies are therefore crucial to ensure models are deployment-optimal in high-risk settings. Hulkund et al. (2025) proposes DS3, deployment-aware data selection (also known as targeted subset selection) that aims to maximize performance on a specific target distribution during fine-tuning. Traditional data selection methods, such as coreset selection (Moser et al., 2025), aim to select the most informative subset of the training pool, but are not built for targeted subset selection.

In this work, we investigate the efficiency of coreset methods for targeted subset selection in supervised image classification. We evaluate performance on three classification datasets in the DataSˆ3 benchmark (iWildCam (Beery et al., 2021), GeoDE (Ramaswamy et al., 2023), and AutoArborist (Beery et al., 2022)), each with four distinct deployments. Motivated by recent work demonstrating the effectiveness of gradient-based methods for adapting models to target deployments (Xia et al., 2024), we focus our analysis on the gradient-matching coreset algorithm `Grad-Match` (Killamsetty et al., 2021a). Our analysis reveals that `Grad-Match` suffers from two key limitations for targeted subset selection: failures in long-tailed distributions and computational inefficiency for large datasets.

To address these concerns, we propose `Grad-Match-ACF` (Adaptive Class Fraction), a novel extension of `Grad-Match` for targeted subset selection. While the standard `Grad-Match` algorithm allocates budget fractions uniformly across label classes, `Grad-Match-ACF` adjusts class-wise fractions according to the deployment-specific label frequencies. As such, `Grad-Match-ACF` is better suited for handling long-tailed distributions and large-scale datasets frequently encountered

in real-world applications. Empirically, we find that our proposed method `Grad-Match-ACF` chooses superior best-performing subsets compared to `Grad-Match` in the majority of deployments in DataSˆ3. Moreover, `Grad-Match-ACF` achieves up to a 18x speed-up improvement over `Grad-Match` on iWildCam deployments. `Grad-Match-ACF` demonstrates more reliability than traditional `Grad-Match` on large, real-world datasets. However, we also observe that for the largest dataset (Auto Arborist), gradient-matching remains computationally expensive; bridging the gap between selection performance and computational efficiency at scale is still an open challenge.

2 Related Work

Data curation and benchmarking. The emergence of foundation models has increased efforts in data quality and curation. Benchmarks such as DataComp (Gadre et al., 2023) and DataPerf (Mazumder et al., 2023) evaluate how data selection affects the performance of large-scale models aimed for *generalization* across many downstream tasks. The DataSˆ3 benchmark (Hulkund et al., 2025), in contrast, focuses on *specialization*, where the goal is to select a representative subset optimized for a specific deployment.

Coreset selection for deep learning. The objective of traditional coreset selection is to select a subset of the full training data that preserves model performance while reducing training time and computation costs. Moser et al. (2025) comprehensively surveys and analyzes the tradeoffs of recent coreset methods, categorizing methods such as `Grad-Match` and `Glister` as "training-oriented" because they utilize training signals like gradients and loss values to inform selection. The DeepCore framework (Guo et al., 2022) similarly benchmarks coreset methods across various selection fractions.

The Gap: However, two main gaps exist in the coreset literature. First, traditional coreset methods aim to approximate performance on the training distribution, whereas targeted subset selection seeks to maximize performance on a *deployment* distribution that may differ substantially from training. While some coreset methods attempt to handle class imbalance (Killamsetty et al., 2021a;b), they are not built for targeted subset selection (Figure 3). Second, existing literature primarily evaluates these coreset algorithms on small, well-balanced image datasets, such as CIFAR-10 (Moser et al., 2025; Guo et al., 2022). Even ImageNet evaluations often fail to demonstrate results for larger budget fractions (Guo et al., 2022). The efficacy of these methods on large, long-tailed real-world datasets remains largely unexplored.

3 Methods

3.1 Problem formulation

Notation. Let $\mathcal{U} = \{(x_i, y_i)\}_{i=1}^{n}$ denote the training pool, where each sample is drawn from a source distribution P_U. Similarly, let $\mathcal{Q} = \{(x_j, y_j)\}_{j=1}^{m}$ denote the query set, where $(x_j, y_j) \sim P_Q$. We define $f \in (0, 1]$ as the fixed budget fraction (an upper bound on the fraction of samples from the training pool the algorithm can select). The selected subset is denoted as $\mathcal{S} \subset \mathcal{U}$, where $|\mathcal{S}| \leq f \cdot |\mathcal{U}|$.

Problem formulation. Traditional coreset literature typically assumes coreset methods can have access to a query set $\mathcal{Q}$ (sometimes referred to as the validation set (Moser et al., 2025)) that is, by default, in-distribution to the training pool ($P_Q = P_U$). Some works attempt to address targeted subset selection via a class-imbalanced query set ($P_Q \neq P_U$), but only demonstrate results in synthetic datasets (Killamsetty et al., 2021a;b). Our work operates in the same $P_Q \neq P_U$ setting, but better reflects real-world deployments where class imbalance naturally occurs.

3.2 Datasets

We use the three classification datasets from the DataSˆ3 benchmark Hulkund et al. (2025), since they have explicitly labeled query sets we can use in our supervised subset selection setting. These datasets are iWildCam for wildlife imagery species classification, GeoDE for diverse household object classification, and AutoArborist for urban street image tree classification. More details about each dataset are given in Appendix A.2.

3.3 BASELINES

We evaluate the following baselines against our proposed new method, `Grad-Match-ACF`, introduced in Section 3.4. Since our focus is on the supervised setting, we remove all classes from the training set that are not present in the query set. For each baseline, we try different budget fractions $f \in \{0.05, 0.1, 0.25, 0.5, 0.75, 0.9\}$.

No Filtering. We utilize all available training data that share labels with the query set.

Random. From the class-filtered training pool, we sample a set fraction uniformly at random.

Match-Dist. We use stratified random sampling to match the relative frequency of the query set.

TSDS. `TSDS` formulates data selection as an optimization problem utilizing optimal transport to minimize distribution discrepancy between the training set and query set while incorporating a regularizer to ensure data diversity and penalize near-duplicates (Liu et al., 2024).

Glister. `Glister` (Killamsetty et al., 2021b) uses a bilevel optimization approach, where the outer objective optimizes over samples and the inner objective optimizes over model parameters. In particular, the goal is to maximize the log-likelihood on a held-out validation set. We include `Glister` as a baseline because it outperforms gradient matching-based coreset methods in Guo et al. (2022) and Moser et al. (2025).

Grad-Match. The `Grad-Match` algorithm (Killamsetty et al., 2021a) views subset selection as an optimization problem to find a subset whose gradient closely matches the query set. `Grad-Match` leverages Orthogonal Matching Pursuit to iteratively select the best gradient approximation up until a specified budget. With gradient-based methods showing strong performance for targeted subset selection on language data (Xia et al., 2024), we focus our analysis on the `Grad-Match` baseline.

3.4 GRAD-MATCH-ACF

Drawbacks of `Grad-Match` for targeted subset selection.

(1) Grad-Match fails to match deployments with long-tailed distributions. Like other traditional coreset methods, `Grad-Match` is built to approximate the training distribution, effectively ignoring the critical shifted, long-tailed distribution of the deployment set. In Section 4, we demonstrate that `Grad-Match`'s label distribution fails to follow that of the deployment distribution.

(2) Grad-Match is computationally expensive for larger datasets. Despite its theoretical promise, `Grad-Match` is limited in practical applications for real-world deployments. The greedy selection process is computationally expensive as it requires calculating per-sample gradients for the entire training pool (Moser et al., 2025), making it unusable for larger datasets. For example, as shown in Figure 6, we find that `Grad-Match` is slower than `Glister` and is unable to finish for subsets greater than 50% in larger datasets in DataSˆ3.

These drawbacks stem from how `Grad-Match` is implemented. The default `Grad-Match` implementation computes per-class approximation [1], where Orthogonal Matching Pursuit is performed once for each class individually. In particular, for every class c, `Grad-Match` assigns a per-class subset size threshold k_c, or how many samples from that class the algorithm can select at most, defined as $k_c = f \cdot |U_c|$ where U_c is the subset of the training pool with class c and f is the total budgeted fraction of the coreset. For a fixed f, the per-class subset size k_c is proportional to the number of samples of c in the training set. This leads `Grad-Match` to select a subset with a label distribution following exactly that of the training pool, which is not necessarily optimal for targeted subset selection. To bridge this gap, we propose `Grad-Match-ACF` (Adaptive Class Fraction), which decouples the per-class budget from the training pool's label distribution.

`Grad-Match-ACF`. The core intuition of `Grad-Match-ACF` is to align the per-class subset sizes with the expected deployment environment rather than the potentially biased training data. By calculating a per-class subset size that matches the label frequency in the *query set*, we ensure that the selection process prioritizes the labels most relevant to the target deployment. This adap-

[1] The alternative is to treat the entire dataset as one pool and pick the most informative points globally. However, the per-class approximation not only reduces memory usage, but also speeds up the algorithm's overall running time (Killamsetty et al., 2021a).

tive weighting prevents majority classes in the training data from over-shadowing critical minority classes required for deployment success.

Specifically, we first define a per-class budget fraction f_c as:

$$f_c = \min(1, f \cdot \frac{|\mathcal{Q}_c|}{|\mathcal{Q}|} \cdot \frac{|\mathcal{U}|}{|\mathcal{U}_c|}), \tag{1}$$

where $\frac{|\mathcal{Q}_c|}{|\mathcal{Q}|}$ represents the relative prevalence of class c in the target query set, and $\frac{|\mathcal{U}|}{|\mathcal{U}_c|}$ accounts for the density of that class within the training pool $\mathcal{U}$. All fraction values are clamped at 1, a relevant nuance for cases where a class is highly prevalent in the deployment but very scarce in the training set. Given f_c, the per-class subset size threshold is $k_c = f_c \cdot |U_c|$. The complete algorithm for `Grad-Match-ACF` is shown in Algorithm 1 in Appendix A.3.

4 Results and Discussion

Dataset	Deploy #	No Filter	Random	Match-Dist	TSDS	GLISTER	GRAD-MATCH	GRAD-MATCH-ACF (Ours)
iWildCam	Deploy 1	0.823 (7.8k)	0.824 (1.9k) ↑	0.822 (4.0k) ↓	0.818 (2.8k) ↓	0.825 (7.0k) ↑	**0.834 (3.9k)** ↑	0.825 (5.6k) ↑
	Deploy 2	0.647 (51k)	0.689 (46k) ↑	0.709 (14k) ↑	**0.724 (7.2k)** ↑	0.687 (38k) ↑	0.661 (4.6k) ↑	0.697 (13k) ↑
	Deploy 3	0.783 (56k)	0.787 (50k) ↑	0.782 (18k) ↓	**0.8 (17k)** ↑	0.794 (42k) ↑	0.777 (28k) ↓	**0.8 (19k)** ↑
	Deploy 4	0.653 (56k)	0.729 (42k) ↑	0.728 (10k) ↑	**0.739 (8.0k)** ↑	0.696 (28k) ↑	0.666 (14k) ↑	0.708 (8.7k) ↑
GeoDE	Deploy 1	0.904 (48k)	0.905 (43k) ↑	0.902 (24k) ↓	0.901 (25k) ↓	**0.91 (43k)** ↑	0.908 (42k) ↑	0.9 (36k) ↓
	Deploy 2	0.916 (48k)	0.913 (36k) ↓	0.904 (27k) ↓	0.904 (23k) ↓	0.914 (43k) ↓	**0.919 (43k)** ↑	0.916 (38k)
	Deploy 3	0.997 (1.3k)	0.996 (955) ↓	0.996 (758) ↓	0.996 (607) ↓	0.997 (955)	**0.998 (637)** ↑	**0.998 (1.1k)** ↑
	Deploy 4	0.941 (2.3k)	0.933 (2.1k) ↓	0.911 (1.4k) ↓	0.93 (1.3k) ↓	0.935 (2.1k) ↓	0.938 (2.1k) ↓	**0.943 (2.1k)** ↑
AutoArborist	Deploy 1	**0.668 (331k)**	0.659 (298k) ↓	0.661 (148k) ↓	0.648 (166k) ↓	0.667 (298k) ↓	0.634 (73k) ↓	0.576 (33k) ↓
	Deploy 2	0.289 (479k)	0.305 (431k) ↑	**0.325 (148k)** ↑	0.29 (249k) ↑	0.305 (390k) ↑	0.259 (52k) ↓	0.304 (54k) ↑
	Deploy 3	0.507 (480k)	0.509 (432k) ↑	0.536 (233k) ↑	0.504 (244k) ↓	0.501 (432k) ↓	0.518 (90k) ↑	**0.542 (101k)** ↑
	Deploy 4	0.295 (474k)	0.296 (427k) ↑	**0.319 (202k)** ↑	0.29 (245k) ↓	0.3 (427k) ↑	0.274 (97k) ↓	0.314 (111k) ↑

Table 1: Best-performing subsets for all methods across all labeled datasets and deployments for LoRA ViT finetuning. We ran each method with budget fraction $\in \{0.05, 0.1, 0.25, 0.5, 0.75, 0.9\}$, and the best-performing subset across the budget fractions is shown (full results with std is included in 2 and per-fraction results are included in A.4.3). **Bold** = best mean in the row. ↑/↓ indicate mean is higher/lower than the No Filter baseline for that deployment. We include ResNet50 full-finetuning results in Table 3.

`Grad-Match-ACF` demonstrates highest performance across most deployments. We first evaluate the general efficacy of coreset methods for targeted subset selection. As shown in Table 2, `Grad-Match-ACF` selects the best subset (bolded) in 4/12 deployments, more than any other method. Moreover, `Grad-Match-ACF` outperforms the No Filter baseline in 9/12 deployments, compared to 4/12 for `TSDS`, 7/12 for `Glister`, and 7/12 for `Grad-Match`. This consistent improvement demonstrates (i) that a well-chosen subset is superior to utilizing the full training data; and (ii) gradient matching-based methods are well-suited for targeted subset selection problems.

`Grad-Match-ACF` better captures deployment distributions than `Grad-Match` and `Match-Dist` in isolation. Table 2 demonstrates that `Grad-Match-ACF` chooses a superior best-performing subset compared to `Grad-Match` in seven out of eleven real-world deployments (with one tie). Compared to `Match-Dist`, `Grad-Match-ACF` outperforms on an even 6/12 deployments for LoRA ViT finetuning and 9/12 deployments for ResNet50 full-finetuning (Table 3). We also plot the performance of `Grad-Match-ACF`, `Grad-Match`, and `Match-Dist` across all budget fraction values in Figure 4. For 31/48 budget fraction settings within iWildCam and GeoDE, selection with `Grad-Match-ACF` outperforms selection with `Grad-Match`; against `Match-Dist`, `Grad-Match-ACF` outperforms in 27/48 (Tables 4, 5).

These results suggest that targeted subset selection requires both distinct advantages offered by `Match-Dist` and `Grad-Match` in isolation. While `Match-Dist` selects the subset by aligning the label distribution of the deployment, it lacks the gradient signals used by `Grad-Match` to select high-influence points. `Grad-Match-ACF` effectively combines both of these ideas. The advantage is evident in `Grad-Match-ACF`'s ability to better match the target deployment distribution. For example, Figure 1 illustrates consistently lower total variation distance between subsets selected by `Grad-Match-ACF` and the test set on all iWildCam deployments. Moreover, Figure 7 provides a comparison of the actual label distributions, showing that `Grad-Match-ACF` follows the deployment distribution while `Grad-Match` deviates substantially.

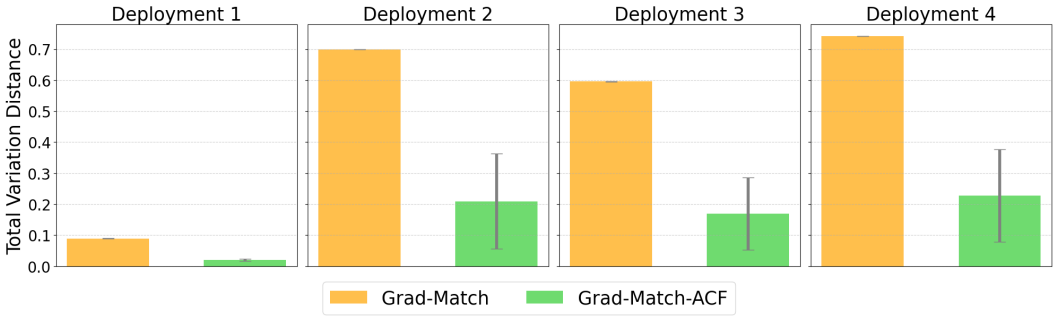

Figure 1: Comparison of total variation distance (TVD) between the selected subset (`Grad-Match` and `Grad-Match-ACF`) and target deployment for all iWildCam deployments. Gray error bars indicate std. The lower TVD for `Grad-Match-ACF` indicates closer alignment to the target deployment distributions than `Grad-Match`. The actual label distributions are shown in Figure 7.

`Grad-Match-ACF` is more computationally efficient than `Grad-Match`. We also find that `Grad-Match-ACF` is faster than `Grad-Match` across most iWildCam deployments and fraction budgets. As illustrated in Figure 2, `Grad-Match-ACF` exhibits an 18.69x speed-up improvement for a 50% budget fraction on iWildCam deployment 2. The exception is iWildCam deployment 1, where `Grad-Match-ACF` is slightly slower than `Grad-Match`; however, this is the smallest deployment by far, with a training pool of 7.8k images (compared to 51k-56k images in the other deployments). The latency gains of `Grad-Match-ACF` are more visible on the larger deployments. We also note that for budget fractions > 0.5, `Grad-Match` was not able to finish selection due to numerical instability issues when running on our hardware (NVIDIA A100 GPUs), though this was not an issue for `Grad-Match-ACF`. As mentioned before, previous surveys benchmark coreset methods on smaller datasets like CIFAR-10 (Guo et al., 2022; Moser et al., 2025) or smaller budget fractions for ImageNet (Guo et al., 2022). Our results here demonstrate the failure modes that occur in large, real-world deployments, and how `Grad-Match-ACF` offers a viable path toward reliable subset selection in such environments.

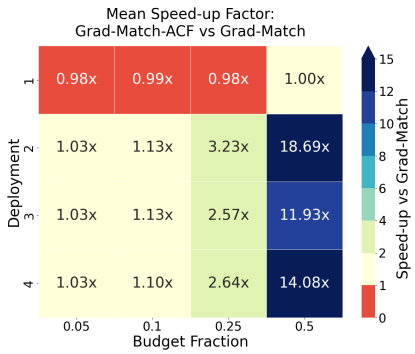

Figure 2: Execution speed-up of `Grad-Match-ACF` relative to `Grad-Match` across all iWildCam deployments for budget fraction $\in \{0.05, 0.1, 0.25, 0.5\}$. Each cell is the mean speed-up factor across three seeds. `Grad-Match` was unable to finish with larger budget fractions (0.75 and 0.9) due to numerical instability issues when running on our hardware (NVIDIA A100 GPUs), though this was not an issue for `Grad-Match-ACF`. `Grad-Match-ACF` provides comparable latency for smaller deployments/budget fractions, and offers up to a 18x speed-up over `Grad-Match` for larger budget fractions. Full execution times are in Table 10.

However, gradient matching-based methods still fail to scale reliably. Despite these improvements, we observe the practical limitations of gradient matching-based selection for large-scale datasets. For Auto Arborist, the largest labeled dataset in the DataS^3 benchmark, both `Grad-Match` and `Grad-Match-ACF` failed to reliably complete the selection process for fraction values greater than $f = 0.25$, as indicated in Figure 4. This failure stems from convergence issues in the non-negative least squares solver in the `Grad-Match` implementation, which encounters numerical instability or time constraints on larger datasets. In contrast, `Glister` and the other baselines proved to be significantly more reliable. While the performance on smaller fractions suggests that `Grad-Match` and `Grad-Match-ACF` possess a high potential for identifying representative subsets on Auto Arborist, scalability limitations constrain their practical utility for larger datasets. Bridging the gap between performance and computational efficiency is essential for scalable subset selection on large, real-world foundational model datasets.

REFERENCES

Yang Bai, Yang Zhou, Jun Zhou, Rick Siow Mong Goh, Daniel Shu Wei Ting, and Yong Liu. From generalist to specialist: Adapting vision language models via task-specific visual instruction tuning. *arXiv preprint arXiv:2410.06456*, 2024. URL https://arxiv.org/abs/2410.06456.

Sara Beery, Dan Morris, and Siyu Yang. Efficient pipeline for camera trap image review, 2019. URL https://arxiv.org/abs/1907.06772.

Sara Beery, Arushi Agarwal, Elijah Cole, and Vighnesh Birodkar. The iwildcam 2021 competition dataset. *arXiv preprint arXiv:2105.03494*, 2021. URL https://arxiv.org/abs/2105.03494.

Sara Beery, Guanhang Wu, Trevor Edwards, Filip Pavetic, Bo Majewski, Shreyasee Mukherjee, Stanley Chan, John Morgan, Vivek Rathod, and Jonathan Huang. The auto arborist dataset: A large-scale benchmark for multiview urban forest monitoring under domain shift. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 36, 2022. URL https://google.github.io/auto-arborist/.

Leslie A. Brandt, Abigail Derby Lewis, Robert T. Fahey, Lydia Scott, Lindsay E. Darling, and Christopher W. Swanston. A framework for adapting urban forests to climate change. *Environmental Science & Policy*, 66:393–402, 2016. URL https://api.semanticscholar.org/CorpusID:156304225.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, 36, 2023. URL https://arxiv.org/abs/2304.14108.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. *International Conference on Database and Expert Systems Applications*, 2022. URL https://arxiv.org/abs/2204.08499.

Neha Hulkund, Alaa Maalouf, Levi Cai, Daniel Yang, Tsun-Hsuan Wang, Abigail O'Neil, Timm Haucke, Sandeep Mukherjee, Vikram Ramaswamy, Judy Hansen Shen, Gabriel Tseng, Mike Walmsley, Daniela Rus, Ken Goldberg, Hannah Kerner, Irene Chen, Yogesh Girdhar, and Sara Beery. Datasˆ 3: Dataset subset selection for specialization. *arXiv preprint arXiv:2504.16277*, 2025. URL https://arxiv.org/abs/2504.16277.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. *International Conference on Machine Learning*, 2021a. URL https://arxiv.org/abs/2103.00123.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *Proceedings of the AAAI conference on artificial intelligence*, 35, 2021b. URL https://arxiv.org/abs/2012.10630.

Zifan Liu, Amin Karbasi, and Theodoros Rekatsinas. Tsds: Data selection for task-specific model finetuning. *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2410.11303.

Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Alicia Parrish, Hannah Rose Kirk, et al. Dataperf: Benchmarks for data-centric ai development. *Advances in Neural Information Processing Systems*, 36, 2023. URL https://arxiv.org/abs/2207.10062.

Brian B. Moser, Arundhati S. Shanbhag, Stanislav Frolov, Federico Raue, Joachim Folz, and Andreas Dengel. A coreset selection of coreset selection literature: Introduction and recent advances. *arXiv preprint arXiv:2505.17799*, 2025. URL https://arxiv.org/abs/2505.17799.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. URL https://arxiv.org/abs/2304.07193.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *International conference on machine learning*, 2021. URL https://arxiv.org/abs/2103.00020.

Vikram V. Ramaswamy, Sing Yu Lin, Dora Zhao, Aaron B. Adcock, Laurens van der Maaten, Deepti Ghadiyaram, and Olga Russakovsky. Geode: a geographically diverse evaluation dataset for object recognition. *Advances in Neural Information Processing Systems*, 36, 2023. URL https://arxiv.org/abs/2301.02560.

Shreya Shankar, Yoni Halpern, Eric Breck, James Atwood, Jimbo Wilson, and D. Sculley. No classification without representation: Assessing geodiversity issues in open data sets for the developing world. *arXiv: Machine Learning*, 2017. URL https://api.semanticscholar.org/CorpusID:26262581.

Oliver Wearn and Paul Glover-Kapfer. Camera-trapping for conservation: a guide to best-practices, 10 2017.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024. URL https://arxiv.org/abs/2402.04333.

Chenghan Yang, Ruiyu Zhao, Yang Liu, and Ling Jiang. Survey of specialized large language model. *ArXiv*, abs/2508.19667, 2025. URL https://api.semanticscholar.org/CorpusID:280918613.

A Appendix

A.1 Code

All code for running baseline and `Grad-Match-ACF` experiments are open-source, available at `https://github.com/ezhu04/task_datacomp`.

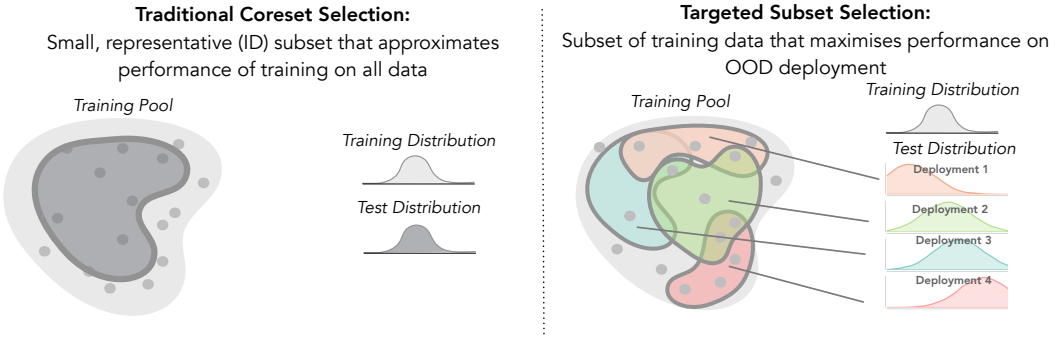

Figure 3: Traditional coreset methods aim to compress the training set into a smaller subset that approximates the performance of training on the full dataset. In contrast, our setting explicitly departs from this objective: we select subsets to maximise performance on a separate, deployment distribution which may differ significantly from the training distribution.

A.2 Dataset Descriptions

iWildCam: To monitor this biodiversity loss, ecologists deploy camera traps—motion-activated cameras placed in the wild (Wearn & Glover-Kapfer, 2017)—and process the data with machine learning models (Beery et al., 2019). However, variations in illumination, camera angle, background, vegetation, color, and animal frequencies across different locations cause these models to generalize poorly to new deployments. To study this problem, we use the iWildCam 2020 dataset, comprising of $203,029$ images from 323 different camera traps spread across multiple countries in different parts of the world. The task is multi-class species classification from 182 different animal species. The deployments are split geospatially to be: (1) Central America, (2) Eastern Africa, (3) Southern Africa, and (4) Southeast Asia.

GeoDE: Object classification datasets are often constructed by scraping images from the web but contain geographical biases (Shankar et al., 2017). GeoDE is a crowd-sourced dataset of 61,490 images made up of 40 different objects and 6 world regions, crowdsources a dataset that is roughly balanced across 40 different objects and six world regions, showing that common objects (stoves, bicycles, etc), vary in appearance across the world. The four depolyments are: (1) objects in Indonesia, (2) objects in Nigeria, (3) indoor objects, and (4) outdoor objects.

AutoArborist: Ecological imagery for environmental monitoring, such as automated tree classification, provides policymakers with critical, data-driven insights to support climate adaptation, urban planning, and more (Brandt et al., 2016). The data is a multi-view, fine-grained visual tree categorization dataset containing street-level images of over 1 million public zone trees from 300 genus-level categories across 23 major cities in the US and Canada. Deployments in AutoArborist correspond to the development models for use by individual cities. The deployment cities of (1) Surrey, (2) Calgary, (3) Los Angeles, and (4) Washington DC each have distinct deployment distributions, leading to a need for specialization of these models.

A.3 The Grad-Match-ACF Algorithm

Our `Grad-Match-ACF` algorithm, formalized in Algorithm 1, builds upon the standard `Grad-Match`algorithm that is defined in Killamsetty et al. (2021a) and implemented in Guo et al. (2022). We utilize the same approach of selecting samples class-by-class, and we reference the Orthogonal Matching Pursuit (OMP) framework described in Killamsetty et al. (2021a). The key

difference in our algorithm is the adaptive per-class budget fraction. While `Grad-Match` uses the same budget fraction f for all classes, `Grad-Match-ACF` uses the frequency of a class in the *query* set to determine the class's budget fraction. The adaptive class fraction is defined in Equation 1. We provide the pseudocode for `Grad-Match-ACF` in Algorithm 1 and highlight the important changes from `Grad-Match` in the comments.

Algorithm 1 GRAD-MATCH-ACF Algorithm

Require: Train set: $\mathcal{U}$; query set: $\mathcal{Q}$; initial subset: $\mathcal{X}^{(0)}$; budget fraction: f; TOL: ϵ; initial params: θ_0; learning rate: α; total epochs: T; selection interval: R; Batchsize: B

$\mathcal{C} \leftarrow \text{UniqueLabels}(\mathcal{Q})$ ▷ Get all the classes in the query (validation) set
for $c \in \mathcal{C}$ **do** do
  $f_c = \min(1, f \cdot \frac{|\mathcal{Q}_c|}{|\mathcal{Q}|} \cdot \frac{|\mathcal{U}|}{|\mathcal{U}_c|})$ ▷ `Grad-Match` implicitly uses $f_c = f$
  $k = \text{round}(|\mathcal{U}_c| \cdot f_c)$ ▷ Per-class subset size uses the adaptive class fraction
  for epochs t in $1, \ldots, T$ **do**
    if $(t \bmod R == 0)$ **then**
      $\mathcal{X}^t, w^t = \text{OMP}(L_T, L_Q, \theta_t, k, \epsilon)$
    else
      $\mathcal{X}^t = \mathcal{X}^{t-1}$
    end if
    $\theta_{t+1} = \text{BatchSGD}(\mathcal{X}^t, w^t, \alpha, L_T, B, \text{Epochs} = 1)$
  end for
end for
Output final model parameters θ_T

A.4 ADDITIONAL RESULTS

Dataset	Deploy #	No Filter	Random	Match-Dist	TSDS	GLISTER	GRAD-MATCH	GRAD-MATCH-ACF (Ours)
iWildCam	Deploy 1	0.823 ± 0.006 (7.8k)	0.824 ± 0.021 (1.9k) ↑	0.822 ± 0.017 (4.0k) ↓	0.818 ± 0.014 (2.8k) ↓	0.825 ± 0.023 (7.0k) ↑	**0.834 ± 0.007 (3.9k)** ↑	0.825 ± 0.01 (5.6k) ↑
	Deploy 2	0.647 ± 0.06 (51k)	0.689 ± 0.028 (46k) ↑	0.709 ± 0.016 (14k) ↑	**0.724 ± 0.012 (7.2k)** ↑	0.687 ± 0.056 (38k) ↑	0.661 † (4.6k) ↑	0.697 ± 0.027 (13k) ↑
	Deploy 3	0.783 ± 0.016 (56k)	0.787 ± 0.021 (50k) ↑	0.782 ± 0.006 (18k) ↓	**0.8 ± 0.015 (17k)** ↑	0.794 ± 0.026 (42k) ↑	0.777 ± 0.009 (28k) ↓	**0.8 ± 0.017 (19k)** ↑
	Deploy 4	0.653 ± 0.056 (56k)	0.729 ± 0.028 (42k) ↑	0.728 ± 0.0 (10k) ↑	**0.739 ± 0.02 (8.0k)** ↑	0.696 ± 0.024 (28k) ↑	0.666 ± 0.031 (14k) ↑	0.708 ± 0.003 (8.7k) ↑
GeoDE	Deploy 1	0.904 ± 0.002 (48k)	0.905 ± 0.006 (43k) ↑	0.902 ± 0.004 (24k) ↓	0.901 ± 0.002 (25k) ↓	**0.91 ± 0.006 (43k)** ↑	0.908 ± 0.014 ‡ (42k) ↑	0.9 ± 0.006 ‡ (36k) ↓
	Deploy 2	0.916 ± 0.005 (48k)	0.913 ± 0.003 (36k) ↓	0.904 ± 0.009 (27k) ↓	0.904 ± 0.001 (23k) ↓	0.914 ± 0.003 (43k) ↓	**0.919 † (43k)** ↑	0.916 ± 0.008 (38k)
	Deploy 3	0.997 ± 0.001 (1.3k)	0.996 ± 0.004 (955) ↓	0.996 ± 0.001 (758) ↓	0.996 ± 0.002 (607) ↓	0.997 ± 0.001 (955)	**0.998 ± 0.001 (637)** ↑	**0.998 ± 0.001 (1.1k)** ↑
	Deploy 4	0.941 ± 0.011 (2.3k)	0.933 ± 0.021 (2.1k) ↓	0.911 ± 0.01 (1.4k) ↓	0.93 ± 0.015 (1.3k) ↓	0.935 ± 0.014 (2.1k) ↓	0.938 ± 0.009 (2.1k) ↓	**0.943 ± 0.008 (2.1k)** ↑
AutoArborist	Deploy 1	**0.668 ± 0.004 (331k)**	0.659 ± 0.013 (298k) ↓	0.661 ± 0.014 (148k) ↓	0.648 ± 0.009 (166k) ↓	0.667 ± 0.003 (298k) ↓	0.634 ± 0.02 ‡ (73k) ↓	0.576 ± 0.008 (33k) ↓
	Deploy 2	0.289 ± 0.008 (479k)	0.305 ± 0.006 (431k) ↑	**0.325 ± 0.009 (148k)** ↑	0.29 ± 0.006 (249k) ↑	0.305 ± 0.004 (390k) ↑	0.259 † (52k) ↓	0.304 ± 0.006 ‡ (54k) ↑
	Deploy 3	0.507 ± 0.023 (480k)	0.509 ± 0.007 (432k) ↑	0.536 ± 0.002 (233k) ↑	0.504 ± 0.012 (244k) ↓	0.501 ± 0.022 (432k) ↓	0.518 ± 0.013 ‡ (90k) ↑	**0.542 † (101k)** ↑
	Deploy 4	0.295 ± 0.01 (474k)	0.296 ± 0.011 (427k) ↑	**0.319 ± 0.006 (202k)** ↑	0.29 ± 0.009 (245k) ↓	0.3 ± 0.007 (427k) ↑	0.274 ± 0.006 ‡ (97k) ↓	0.314 † (111k) ↑

Table 2: Best-performing subsets for all methods across all labeled datasets and deployments for LoRA ViT finetuning. **Bold** = best mean in the row. ↑ /↓ indicate mean is higher/lower than the No Filter baseline for that deployment. Results are reported as the mean accuracy ± std across seeds. Subset sizes are included in parentheses. While three runs were intended for all configurations, † and ‡ indicate results based on $n = 1$ and $n = 2$ runs, respectively.

Dataset	Deploy #	No Filter	Random	Match-Dist	TSDS	GLISTER	GRAD-MATCH	GRAD-MATCH-ACF (Ours)
iWildCam	Deploy 1	0.743 ± 0.003 (7.8k)	0.773 ± 0.031 (3.9k) ↑	0.748 ± 0.052 (2.9k) ↑	0.751 ± 0.041 (3.7k) ↑	**0.782 ± 0.028 (5.8k)** ↑	0.759 ± 0.006 (7.0k) ↑	0.762 ± 0.007 (3.8k) ↑
	Deploy 2	0.577 ± 0.023 (51k)	0.617 ± 0.038 (5.1k) ↑	**0.646 ± 0.029 (14k)** ↑	0.637 ± 0.028 (11k) ↑	0.615 ± 0.043 (38k) ↑	0.609 † (46k) ↑	0.612 ± 0.056 (11k) ↑
	Deploy 3	0.669 ± 0.009 (56k)	0.618 ± 0.012 (14k) ↓	**0.681 ± 0.03 (14k)** ↑	0.665 ± 0.011 (16k) ↓	0.645 ± 0.019 (50k) ↓	0.645 ± 0.013 ‡ (42k) ↓	0.673 ± 0.011 (19k) ↑
	Deploy 4	0.416 ± 0.119 (56k)	**0.513 ± 0.023 (50k)** ↑	0.485 ± 0.094 (10k) ↑	0.509 ± 0.1 (13k) ↑	0.508 ± 0.063 (50k) ↑	0.506 ± 0.111 ‡ (42k) ↑	0.51 ± 0.071 (8.7k) ↑
GeoDE	Deploy 1	0.822 ± 0.009 (48k)	0.801 ± 0.013 (24k) ↓	0.795 ± 0.011 (24k) ↓	0.797 ± 0.004 (22k) ↓	0.818 ± 0.02 (43k) ↓	**0.823 ± 0.014 ‡ (42k)** ↑	0.813 ± 0.004 ‡ (36k) ↓
	Deploy 2	0.828 ± 0.007 (478k)	0.841 ± 0.006 (43k) ↑	0.822 ± 0.008 (27k) ↓	0.802 ± 0.022 (26k) ↓	0.822 ± 0.007 (43k) ↓	**0.845 † (43k)** ↑	0.828 ± 0.006 (38k)
	Deploy 3	0.979 ± 0.001 (1.3k)	**0.984 ± 0.003 (1.1k)** ↑	0.975 ± 0.011 (489) ↓	0.982 ± 0.005 (467) ↑	0.981 ± 0.007 (127) ↑	0.982 ± 0.004 (1.1k) ↑	0.984 ± 0.021 (637) ↑
	Deploy 4	0.907 ± 0.021 (2.3k)	0.915 ± 0.011 (2.1k) ↑	0.888 ± 0.025 (1.4k) ↓	0.889 ± 0.015 (857) ↓	**0.917 ± 0.008 (2.1k)** ↑	0.911 ± 0.007 (2.1k) ↑	0.913 ± 0.013 (1.8k) ↑
AutoArborist	Deploy 1	0.669 ± 0.094 (331k)	0.6 ± 0.017 (298k) ↓	0.542 ± 0.012 (135k) ↓	0.646 ± 0.002 (150k) ↓	**0.719 ± 0.008 (298k)** ↑	0.527 ± 0.012 ‡ (73k) ↓	0.536 ± 0.008 (33k) ↓
	Deploy 2	0.28 ± 0.046 (479k)	0.28 ± 0.008 (431k)	0.245 ± 0.014 (163k) ↓	0.28 ± 0.018 (249k)	**0.307 ± 0.004 (325k)** ↑	0.213 † (52k) ↓	0.247 ± 0.003 (37k) ↓
	Deploy 3	0.51 ± 0.13 (480k)	0.473 ± 0.014 (432k) ↓	0.351 ± 0.034 (233k) ↓	0.568 ± 0.025 (244k) ↑	**0.626 ± 0.023 (432k)** ↑	0.562 ± 0.024 ‡ (90k) ↑	0.606 † (101k) ↑
	Deploy 4	0.282 ± 0.037 (474k)	0.266 ± 0.009 (427k) ↓	0.242 ± 0.037 (224k) ↓	0.287 ± 0.012 (245k) ↑	**0.309 ± 0.023 (427k)** ↑	0.246 ± 0.032 ‡ (97k) ↓	0.275 † (111k) ↓

Table 3: Best-performing subsets for all methods across all labeled datasets and deployments for ResNet50 full-finetuning. **Bold** = best mean in the row. ↑ /↓ indicate mean is higher/lower than the No Filter baseline for that deployment. Results are reported as the mean accuracy ± std across seeds. Subset sizes are included in parentheses. While three runs were intended for all configurations, † and ‡ indicate results based on $n = 1$ and $n = 2$ runs, respectively.

A.4.1 EFFICIENCY PLOTS FOR LORA VIT FINETUNING

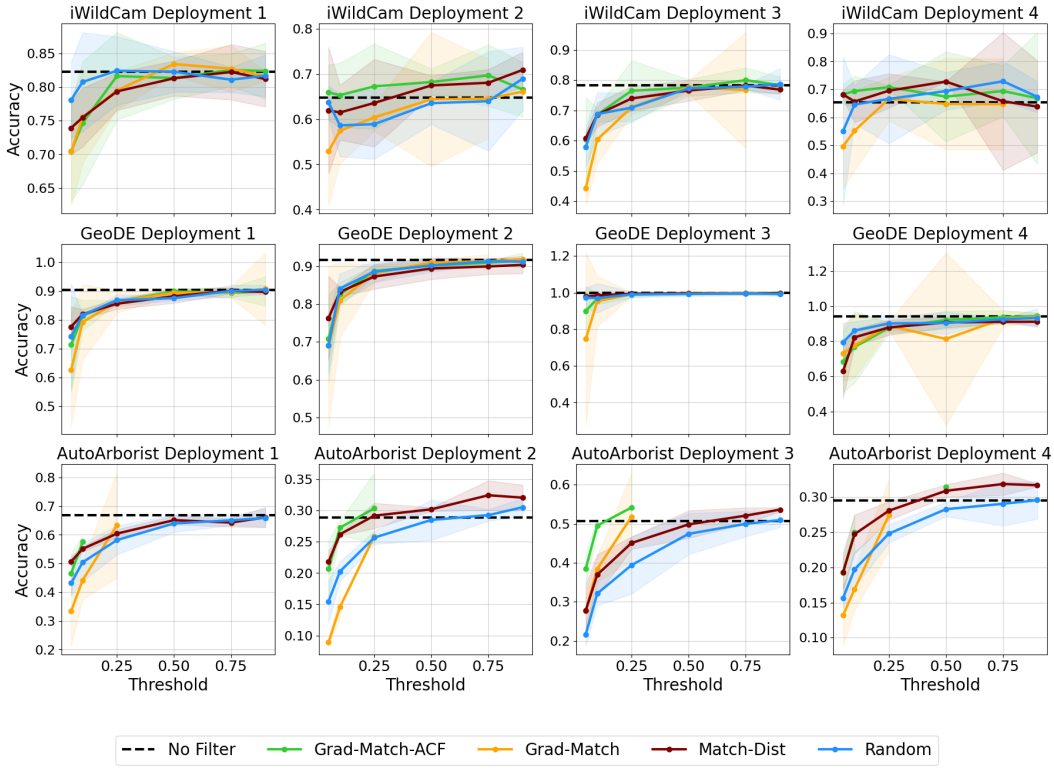

Figure 4: The sample efficiency of GRAD-MATCH and GRAD-MATCH-ACF for the LoRA ViT results, with No Filter, Random Filter, and Match-Dist as comparison points. Error bars around each line indicate 95% confidence intervals. We find that GRAD-MATCH-ACF outperforms GRAD-MATCH in 31 out of 48 budget/deployment settings within GeoDE and iWildCam. However, both struggle with computational efficiency for larger scale datasets, such as AutoArborist, as shown in the line plots where they often are non-computable past 25% of the dataset.

A.4.2 EFFICIENCY PLOTS FOR RESNET50 FULL-FINETUNING

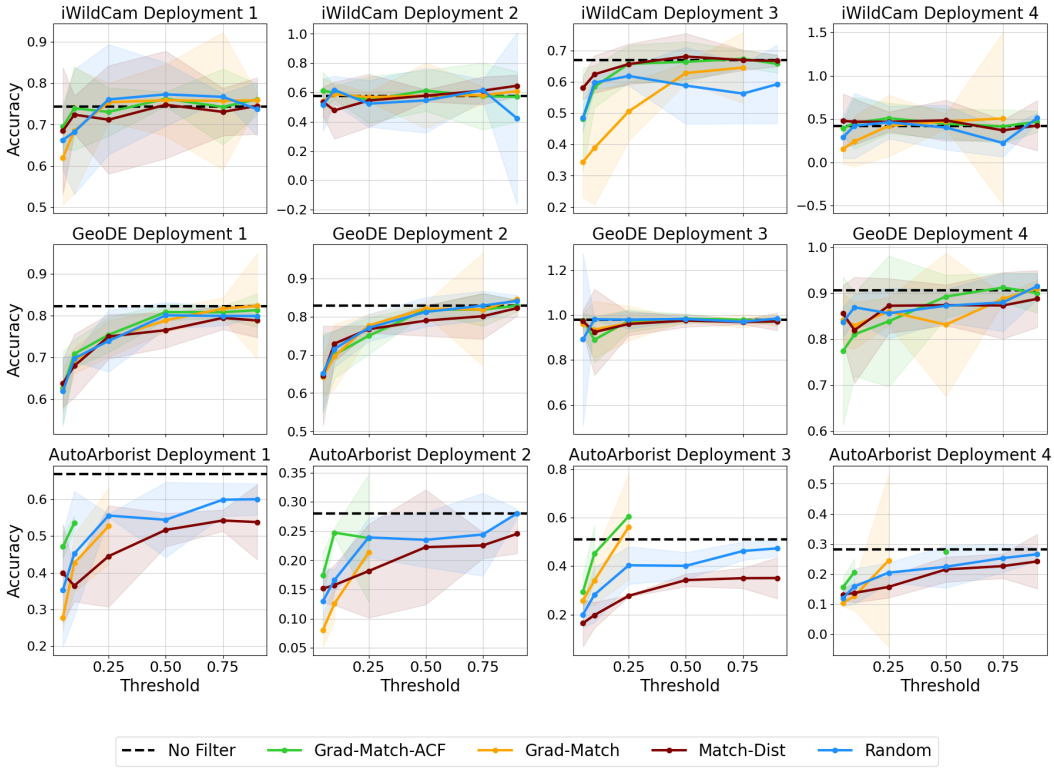

Figure 5: The sample efficiency of GRAD-MATCH and GRAD-MATCH-ACF for the ResNet50 results, with No Filter, Random Filter, and Match-Dist as comparison points. Error bars around each line indicate 95% confidence intervals. We find that GRAD-MATCH-ACF outperforms GRAD-MATCH in 29 out of 48 budget/deployment settings within GeoDE and iWildCam. However, both struggle with computational efficiency for larger scale datasets, such as AutoArborist, as shown in the line plots where they often are non-computable past 25% of the dataset.

A.4.3 BASELINE PERFORMANCE ACROSS ALL FRACTIONS AND DATASETS FOR VIT LORA FINETUNING

Deployment	Method	Fraction						
		5%	10%	25%	50%	75%	90%	100%
Deploy 1	Random	0.781 ± 0.023	0.808 ± 0.03	0.824 ± 0.021	0.823 ± 0.012	0.811 ± 0.008	0.817 ± 0.012	0.823 ± 0.006
	Match-Dist	0.739 ± 0.016	0.755 ± 0.018	0.794 ± 0.011	0.813 ± 0.01	0.822 ± 0.017	0.812 ± 0.017	0.823 ± 0.006
	TSDS	0.772 ± 0.019	0.799 ± 0.018	0.799 ± 0.019	0.818 ± 0.014	0.814 ± 0.018	0.797 ± 0.033	0.823 ± 0.006
	GLISTER	0.662 ± 0.055	0.676 ± 0.047	0.758 ± 0.014	0.804 ± 0.007	0.818 ± 0.012	0.825 ± 0.023	0.823 ± 0.006
	GRAD-MATCH	0.704 ± 0.032	0.755 ± 0.028	0.795 ± 0.011	0.834 ± 0.007	0.827 ± 0.014	0.817 ± 0.003	0.823 ± 0.006
	GRAD-MATCH-ACF	0.704 ± 0.03	0.747 ± 0.037	0.816 ± 0.026	0.813 ± 0.013	0.825 ± 0.01	0.824 ± 0.017	0.823 ± 0.006
Deploy 2	Random	0.637 ± 0.015	0.586 ± 0.028	0.589 ± 0.031	0.635 ± 0.019	0.64 ± 0.044	0.689 ± 0.028	0.647 ± 0.06
	Match-Dist	0.62 ± 0.056	0.615 ± 0.024	0.636 ± 0.039	0.674 ± 0.009	0.68 ± 0.012	0.709 ± 0.016	0.647 ± 0.06
	TSDS	0.689 ± 0.037	0.673 ± 0.026	0.724 ± 0.012	0.706 ± 0.01	0.715 ± 0.014	0.7 ± 0.012	0.647 ± 0.06
	GLISTER	0.548 ± 0.063	0.568 ± 0.035	0.64 ± 0.044	0.611 ± 0.029	0.687 ± 0.056	0.671 ± 0.018	0.647 ± 0.06
	GRAD-MATCH	0.529 ± 0.048	0.574 ± 0.03	0.604 ± 0.012	0.644 ± 0.016 ‡	0.645 ± 0.035	0.661 †	0.647 ± 0.06
	GRAD-MATCH-ACF	0.659 ± 0.025	0.653 ± 0.029	0.672 ± 0.038	0.682 ± 0.012	0.697 ± 0.027	0.665 ± 0.024	0.647 ± 0.06
Deploy 3	Random	0.58 ± 0.026	0.688 ± 0.029	0.709 ± 0.02	0.773 ± 0.002	0.778 ± 0.003	0.787 ± 0.021	0.783 ± 0.016
	Match-Dist	0.608 ± 0.008	0.686 ± 0.016	0.74 ± 0.016	0.766 ± 0.014	0.782 ± 0.006	0.769 ± 0.006	0.783 ± 0.016
	TSDS	0.702 ± 0.013	0.731 ± 0.016	0.772 ± 0.015	0.785 ± 0.014	0.79 ± 0.013	0.8 ± 0.015	0.783 ± 0.016
	GLISTER	0.601 ± 0.018	0.64 ± 0.06	0.68 ± 0.039	0.77 ± 0.009	0.794 ± 0.026	0.774 ± 0.009	0.783 ± 0.016
	GRAD-MATCH	0.443 ± 0.022	0.603 ± 0.032	0.71 ± 0.01	0.777 ± 0.009	0.767 ± 0.021 ‡	–	0.783 ± 0.016
	GRAD-MATCH-ACF	0.605 ± 0.055	0.686 ± 0.009	0.765 ± 0.04	0.776 ± 0.011	0.8 ± 0.017	0.784 ± 0.011	0.783 ± 0.016
Deploy 4	Random	0.552 ± 0.107	0.646 ± 0.028	0.666 ± 0.064	0.694 ± 0.027	0.729 ± 0.028	0.674 ± 0.022	0.653 ± 0.056
	Match-Dist	0.682 ± 0.006	0.657 ± 0.03	0.695 ± 0.025	0.728 ± 0.0	0.658 ± 0.099	0.639 ± 0.066	0.653 ± 0.056
	TSDS	0.676 ± 0.01	0.677 ± 0.06	0.739 ± 0.02	0.735 ± 0.013	0.688 ± 0.045	0.706 ± 0.027	0.653 ± 0.056
	GLISTER	0.567 ± 0.103	0.606 ± 0.039	0.657 ± 0.069	0.696 ± 0.024	0.633 ± 0.041	0.667 ± 0.081	0.653 ± 0.056
	GRAD-MATCH	0.497 ± 0.061	0.552 ± 0.016 ‡	0.666 ± 0.031	0.648 ± 0.066	0.647 ± 0.018 ‡	–	0.653 ± 0.056
	GRAD-MATCH-ACF	0.682 ± 0.015	0.694 ± 0.022	0.708 ± 0.003	0.674 ± 0.065	0.694 ± 0.037	0.668 ± 0.095	0.653 ± 0.056

Table 4: Selection performance for all methods on all iWildCam deployments for LoRA ViT finetuning. Results are reported as (mean accuracy $\pm$ std). While three runs were intended for all configurations, † and ‡ indicate results based on $n = 1$ and $n = 2$ runs, respectively. – indicates instances where $n = 0$.

Deployment	Method	Fraction						
		5%	10%	25%	50%	75%	90%	100%
Deploy 1	Random	0.743 ± 0.073	0.817 ± 0.001	0.869 ± 0.008	0.876 ± 0.003	0.901 ± 0.008	0.905 ± 0.006	0.904 ± 0.002
	Match-Dist	0.777 ± 0.028	0.82 ± 0.007	0.857 ± 0.007	0.883 ± 0.01	0.902 ± 0.004	0.898 ± 0.002	0.904 ± 0.002
	TSDS	0.746 ± 0.02	0.81 ± 0.015	0.849 ± 0.015	0.875 ± 0.009	0.889 ± 0.006	0.901 ± 0.012	0.904 ± 0.002
	GLISTER	0.68 ± 0.097	0.805 ± 0.016	0.864 ± 0.014	0.896 ± 0.004	0.904 ± 0.002	0.91 ± 0.006	0.904 ± 0.002
	GRAD-MATCH	0.627 ± 0.078	0.793 ± 0.053	0.864 ± 0.015	0.895 ± 0.004	0.897 ± 0.008	0.908 ± 0.014 ‡	0.904 ± 0.002
	GRAD-MATCH-ACF	0.715 ± 0.067	0.816 ± 0.022	0.864 ± 0.002	0.9 ± 0.004	0.895 ± 0.007	0.9 ± 0.006 ‡	0.904 ± 0.002
Deploy 2	Random	0.69 ± 0.029	0.841 ± 0.016	0.886 ± 0.011	0.902 ± 0.007	0.913 ± 0.003	0.912 ± 0.002	0.916 ± 0.005
	Match-Dist	0.763 ± 0.045	0.832 ± 0.005	0.873 ± 0.013	0.894 ± 0.011	0.9 ± 0.008	0.904 ± 0.009	0.916 ± 0.005
	TSDS	0.776 ± 0.018	0.824 ± 0.022	0.874 ± 0.009	0.896 ± 0.007	0.904 ± 0.001	0.902 ± 0.005	0.916 ± 0.005
	GLISTER	0.617 ± 0.042	0.815 ± 0.029	0.877 ± 0.003	0.911 ± 0.004	0.908 ± 0.004	0.914 ± 0.003	0.916 ± 0.005
	GRAD-MATCH	0.69 ± 0.085	0.81 ± 0.04	0.881 ± 0.01	0.91 ± 0.003	0.914 ± 0.002 ‡	0.919 †	0.916 ± 0.005
	GRAD-MATCH-ACF	0.709 ± 0.046	0.826 ± 0.013	0.888 ± 0.006	0.902 ± 0.005	0.91 ± 0.002	0.916 ± 0.008	0.916 ± 0.005
Deploy 3	Random	0.972 ± 0.022	0.97 ± 0.032	0.99 ± 0.006	0.992 ± 0.005	0.996 ± 0.004	0.991 ± 0.004	0.997 ± 0.001
	Match-Dist	0.98 ± 0.006	0.984 ± 0.006	0.994 ± 0.002	0.995 ± 0.001	0.994 ± 0.001	0.996 ± 0.001	0.997 ± 0.001
	TSDS	0.956 ± 0.057	0.982 ± 0.009	0.995 ± 0.003	0.995 ± 0.001	0.996 ± 0.002	0.995 ± 0.003	0.997 ± 0.001
	GLISTER	0.961 ± 0.019	0.957 ± 0.038	0.991 ± 0.004	0.996 ± 0.002	0.997 ± 0.001	0.997 ± 0.002	0.997 ± 0.001
	GRAD-MATCH	0.747 ± 0.19	0.952 ± 0.055	0.991 ± 0.005	0.998 ± 0.001	0.997 ± 0.002	0.996 ± 0.003	0.997 ± 0.001
	GRAD-MATCH-ACF	0.9 ± 0.055	0.966 ± 0.012	0.99 ± 0.007	0.995 ± 0.002	0.997 ± 0.001	0.998 ± 0.001	0.997 ± 0.001
Deploy 4	Random	0.795 ± 0.044	0.86 ± 0.021	0.902 ± 0.015	0.904 ± 0.026	0.926 ± 0.002	0.933 ± 0.021	0.941 ± 0.011
	Match-Dist	0.63 ± 0.061	0.822 ± 0.019	0.879 ± 0.018	0.908 ± 0.012	0.911 ± 0.003	0.911 ± 0.01	0.941 ± 0.011
	TSDS	0.652 ± 0.212	0.801 ± 0.047	0.875 ± 0.021	0.899 ± 0.014	0.909 ± 0.004	0.93 ± 0.015	0.941 ± 0.011
	GLISTER	0.681 ± 0.083	0.829 ± 0.036	0.882 ± 0.002	0.929 ± 0.02	0.931 ± 0.018	0.935 ± 0.014	0.941 ± 0.011
	GRAD-MATCH	0.732 ± 0.063	0.776 ± 0.074	0.889 ± 0.009	0.813 ± 0.198	0.928 ± 0.019	0.938 ± 0.009	0.941 ± 0.011
	GRAD-MATCH-ACF	0.684 ± 0.07	0.768 ± 0.082	0.875 ± 0.012	0.921 ± 0.021	0.938 ± 0.015	0.943 ± 0.008	0.941 ± 0.011

Table 5: Selection performance for all methods on all GeoDE deployments for LoRA ViT finetuning. Results are reported as (mean accuracy $\pm$ std). While three runs were intended for all configurations, † and ‡ indicate results based on $n = 1$ and $n = 2$ runs, respectively. – indicates instances where $n = 0$.

Deployment	Method	Fraction						
		5%	10%	25%	50%	75%	90%	100%
Deploy 1	Random	0.432 ± 0.018	0.504 ± 0.027	0.582 ± 0.02	0.64 ± 0.012	0.651 ± 0.007	0.659 ± 0.013	0.668 ± 0.004
	Match-Dist	0.507 ± 0.007	0.551 ± 0.004	0.604 ± 0.006	0.651 ± 0.006	0.643 ± 0.006	0.661 ± 0.014	0.668 ± 0.004
	TSDS	0.438 ± 0.012	0.527 ± 0.016	0.595 ± 0.0	0.623 ± 0.009	0.645 ± 0.006	0.648 ± 0.009	0.668 ± 0.004
	GLISTER	0.167 ± 0.013	0.246 ± 0.031	0.366 ± 0.024	0.533 ± 0.046	0.637 ± 0.018	0.667 ± 0.003	0.668 ± 0.004
	GRAD-MATCH	0.334 ± 0.049	0.442 ± 0.027	0.634 ± 0.02 ‡	–	–	–	0.668 ± 0.004
	GRAD-MATCH-ACF	0.466 ± 0.029	0.576 ± 0.008	–	–	–	–	0.668 ± 0.004
Deploy 2	Random	0.155 ± 0.014	0.203 ± 0.006	0.257 ± 0.004	0.285 ± 0.013	0.293 ± 0.004	0.305 ± 0.006	0.289 ± 0.008
	Match-Dist	0.218 ± 0.007	0.262 ± 0.001	0.292 ± 0.008	0.302 ± 0.001	0.325 ± 0.009	0.321 ± 0.008	0.289 ± 0.008
	TSDS	0.148 ± 0.018	0.201 ± 0.013	0.251 ± 0.007	0.268 ± 0.002	0.287 ± 0.01	0.29 ± 0.006	0.289 ± 0.008
	GLISTER	0.156 ± 0.002	0.183 ± 0.016	0.252 ± 0.015	0.29 ± 0.003	0.3 ± 0.002	0.305 ± 0.004	0.289 ± 0.008
	GRAD-MATCH	0.089 ± 0.002	0.145 ± 0.008	0.259 †	–	–	–	0.289 ± 0.008
	GRAD-MATCH-ACF	0.207 ± 0.018	0.273 ± 0.005	0.304 ± 0.006 ‡	–	–	–	0.289 ± 0.008
Deploy 3	Random	0.217 ± 0.012	0.321 ± 0.012	0.394 ± 0.03	0.474 ± 0.021	0.5 ± 0.013	0.509 ± 0.007	0.507 ± 0.023
	Match-Dist	0.278 ± 0.027	0.369 ± 0.021	0.451 ± 0.006	0.498 ± 0.014	0.521 ± 0.008	0.536 ± 0.002	0.507 ± 0.023
	TSDS	0.234 ± 0.009	0.325 ± 0.015	0.415 ± 0.016	0.474 ± 0.004	0.497 ± 0.013	0.504 ± 0.012	0.507 ± 0.023
	GLISTER	0.135 ± 0.009	0.174 ± 0.032	0.265 ± 0.04	0.388 ± 0.043	0.489 ± 0.004	0.501 ± 0.022	0.507 ± 0.023
	GRAD-MATCH	0.277 ± 0.021	0.384 ± 0.013	0.518 ± 0.013 ‡	–	–	–	0.507 ± 0.023
	GRAD-MATCH-ACF	0.385 ± 0.016	0.495 ± 0.008	0.542 †	–	–	–	0.507 ± 0.023
Deploy 4	Random	0.157 ± 0.006	0.197 ± 0.01	0.248 ± 0.005	0.283 ± 0.004	0.29 ± 0.013	0.296 ± 0.011	0.295 ± 0.01
	Match-Dist	0.193 ± 0.011	0.247 ± 0.011	0.281 ± 0.005	0.309 ± 0.004	0.319 ± 0.006	0.317 ± 0.001	0.295 ± 0.01
	TSDS	0.138 ± 0.024	0.205 ± 0.009	0.247 ± 0.004	0.275 ± 0.005	0.282 ± 0.011	0.29 ± 0.009	0.295 ± 0.01
	GLISTER	0.071 ± 0.013	0.095 ± 0.013	0.15 ± 0.022	0.221 ± 0.013	0.27 ± 0.003	0.3 ± 0.007	0.295 ± 0.01
	GRAD-MATCH	0.132 ± 0.017	0.169 ± 0.012	0.274 ± 0.006 ‡	–	–	–	0.295 ± 0.01
	GRAD-MATCH-ACF	0.193 ± 0.009	0.25 ± 0.01	–	0.314 †	–	–	0.295 ± 0.01

Table 6: Selection performance for all methods on all Auto Arborist deployments for LoRA ViT finetuning. Results are reported as (mean accuracy $\pm$ std). While three runs were intended for all configurations, † and ‡ indicate results based on $n = 1$ and $n = 2$ runs, respectively. – indicates instances where $n = 0$.

A.4.4 BASELINE PERFORMANCE ACROSS ALL FRACTIONS AND DATASETS FOR RESNET50 FULL-FINETUNING

Deployment	Method	Fraction						
		5%	10%	25%	50%	75%	90%	100%
Deploy 1	Random	0.662 ± 0.03	0.682 ± 0.061	0.761 ± 0.054	0.773 ± 0.031	0.767 ± 0.003	0.737 ± 0.026	0.743 ± 0.003
	Match-Dist	0.685 ± 0.062	0.724 ± 0.018	0.712 ± 0.053	0.748 ± 0.052	0.731 ± 0.023	0.745 ± 0.028	0.743 ± 0.003
	TSDS	0.719 ± 0.01	0.725 ± 0.03	0.734 ± 0.032	0.725 ± 0.055	0.751 ± 0.041	0.715 ± 0.034	0.743 ± 0.003
	GLISTER	0.501 ± 0.095	0.546 ± 0.089	0.674 ± 0.071	0.728 ± 0.027	0.782 ± 0.028	0.776 ± 0.028	0.743 ± 0.003
	GRAD-MATCH	0.62 ± 0.046	0.685 ± 0.061	0.754 ± 0.019	0.759 ± 0.035	0.757 ± 0.067	0.759 ± 0.006	0.743 ± 0.003
	GRAD-MATCH-ACF	0.695 ± 0.022	0.739 ± 0.041	0.731 ± 0.023	0.762 ± 0.007	0.743 ± 0.037	0.761 ± 0.015	0.743 ± 0.003
Deploy 2	Random	0.514 ± 0.073	0.617 ± 0.038	0.523 ± 0.064	0.546 ± 0.089	0.615 ± 0.029	0.422 ± 0.237	0.577 ± 0.023
	Match-Dist	0.537 ± 0.04	0.477 ± 0.078	0.544 ± 0.073	0.578 ± 0.016	0.612 ± 0.038	0.646 ± 0.029	0.577 ± 0.023
	TSDS	0.506 ± 0.033	0.587 ± 0.077	0.554 ± 0.08	0.566 ± 0.044	0.637 ± 0.028	0.614 ± 0.011	0.577 ± 0.023
	GLISTER	0.52 ± 0.008	0.559 ± 0.059	0.548 ± 0.092	0.6 ± 0.003	0.615 ± 0.043	0.581 ± 0.022	0.577 ± 0.023
	GRAD-MATCH	0.54 ± 0.062	0.588 ± 0.031	0.568 ± 0.013	0.578 ± 0.025 ‡	0.58 ± 0.023	0.609 †	0.577 ± 0.023
	GRAD-MATCH-ACF	0.612 ± 0.052	0.598 ± 0.019	0.549 ± 0.064	0.612 ± 0.056	0.574 ± 0.092	0.571 ± 0.069	0.577 ± 0.023
Deploy 3	Random	0.485 ± 0.017	0.597 ± 0.008	0.618 ± 0.012	0.588 ± 0.049	0.562 ± 0.038	0.592 ± 0.05	0.669 ± 0.009
	Match-Dist	0.58 ± 0.015	0.624 ± 0.024	0.656 ± 0.026	0.681 ± 0.03	0.669 ± 0.013	0.667 ± 0.007	0.669 ± 0.009
	TSDS	0.567 ± 0.023	0.598 ± 0.038	0.654 ± 0.004	0.659 ± 0.039	0.665 ± 0.011	0.649 ± 0.011	0.669 ± 0.009
	GLISTER	0.52 ± 0.037	0.484 ± 0.099	0.528 ± 0.027	0.609 ± 0.021	0.617 ± 0.011	0.645 ± 0.019	0.669 ± 0.009
	GRAD-MATCH	0.344 ± 0.045	0.389 ± 0.073	0.505 ± 0.038	0.628 ± 0.013	0.645 ± 0.013 ‡	–	0.669 ± 0.009
	GRAD-MATCH-ACF	0.481 ± 0.066	0.584 ± 0.027	0.657 ± 0.024	0.664 ± 0.026	0.673 ± 0.011	0.657 ± 0.012	0.669 ± 0.009
Deploy 4	Random	0.289 ± 0.098	0.422 ± 0.152	0.463 ± 0.076	0.403 ± 0.104	0.223 ± 0.064	0.513 ± 0.023	0.416 ± 0.119
	Match-Dist	0.478 ± 0.127	0.47 ± 0.107	0.468 ± 0.046	0.485 ± 0.094	0.369 ± 0.048	0.425 ± 0.117	0.416 ± 0.119
	TSDS	0.503 ± 0.103	0.385 ± 0.086	0.423 ± 0.136	0.423 ± 0.13	0.509 ± 0.1	0.48 ± 0.037	0.416 ± 0.119
	GLISTER	0.295 ± 0.165	0.196 ± 0.112	0.346 ± 0.054	0.415 ± 0.102	0.447 ± 0.105	0.508 ± 0.063	0.416 ± 0.119
	GRAD-MATCH	0.156 ± 0.069	0.242 ± 0.032 ‡	0.422 ± 0.143	0.474 ± 0.035	0.506 ± 0.111 ‡	–	0.416 ± 0.119
	GRAD-MATCH-ACF	0.393 ± 0.033	0.454 ± 0.12	0.51 ± 0.071	0.445 ± 0.069	0.415 ± 0.077	0.47 ± 0.049	0.416 ± 0.119

Table 7: Selection performance for all methods on all iWildCam deployments for ResNet50 full-finetuning. Results are reported as (mean accuracy ± std). While three runs were intended for all configurations, † and ‡ indicate results based on $n = 1$ and $n = 2$ runs, respectively. – indicates instances where $n = 0$.

Deployment	Method	Fraction						
		5%	10%	25%	50%	75%	90%	100%
Deploy 1	Random	0.619 ± 0.033	0.698 ± 0.003	0.74 ± 0.03	0.801 ± 0.013	0.799 ± 0.009	0.799 ± 0.001	0.822 ± 0.009
	Match-Dist	0.638 ± 0.023	0.68 ± 0.031	0.75 ± 0.021	0.765 ± 0.017	0.795 ± 0.011	0.789 ± 0.016	0.822 ± 0.009
	TSDS	0.64 ± 0.04	0.672 ± 0.038	0.698 ± 0.028	0.753 ± 0.011	0.797 ± 0.004	0.791 ± 0.022	0.822 ± 0.009
	GLISTER	0.612 ± 0.014	0.692 ± 0.001	0.747 ± 0.015	0.797 ± 0.016	0.814 ± 0.018	0.818 ± 0.02	0.822 ± 0.009
	GRAD-MATCH	0.62 ± 0.021	0.692 ± 0.012	0.744 ± 0.018	0.789 ± 0.011	0.817 ± 0.011	0.823 ± 0.014 ‡	0.822 ± 0.009
	GRAD-MATCH-ACF	0.627 ± 0.035	0.709 ± 0.014	0.754 ± 0.017	0.809 ± 0.008	0.808 ± 0.006	0.813 ± 0.004 ‡	0.822 ± 0.009
Deploy 2	Random	0.651 ± 0.039	0.715 ± 0.019	0.77 ± 0.014	0.812 ± 0.015	0.829 ± 0.015	0.841 ± 0.006	0.828 ± 0.007
	Match-Dist	0.645 ± 0.052	0.729 ± 0.019	0.767 ± 0.004	0.789 ± 0.016	0.801 ± 0.024	0.822 ± 0.008	0.828 ± 0.007
	TSDS	0.621 ± 0.026	0.728 ± 0.019	0.761 ± 0.019	0.774 ± 0.015	0.796 ± 0.017	0.802 ± 0.022	0.828 ± 0.007
	GLISTER	0.657 ± 0.02	0.654 ± 0.051	0.759 ± 0.006	0.798 ± 0.013	0.811 ± 0.01	0.822 ± 0.007	0.828 ± 0.007
	GRAD-MATCH	0.643 ± 0.021	0.696 ± 0.034	0.777 ± 0.012	0.821 ± 0.007	0.818 ± 0.016 ‡	0.845 †	0.828 ± 0.007
	GRAD-MATCH-ACF	0.65 ± 0.042	0.7 ± 0.025	0.75 ± 0.019	0.818 ± 0.004	0.818 ± 0.017	0.828 ± 0.006	0.828 ± 0.007
Deploy 3	Random	0.891 ± 0.154	0.981 ± 0.002	0.979 ± 0.016	0.982 ± 0.004	0.97 ± 0.004	0.984 ± 0.003	0.979 ± 0.001
	Match-Dist	0.973 ± 0.019	0.924 ± 0.077	0.96 ± 0.021	0.975 ± 0.011	0.969 ± 0.004	0.971 ± 0.016	0.979 ± 0.001
	TSDS	0.962 ± 0.002	0.887 ± 0.155	0.958 ± 0.011	0.982 ± 0.005	0.975 ± 0.008	0.97 ± 0.005	0.979 ± 0.001
	GLISTER	0.975 ± 0.01	0.981 ± 0.007	0.978 ± 0.022	0.981 ± 0.011	0.965 ± 0.005	0.973 ± 0.011	0.979 ± 0.001
	GRAD-MATCH	0.96 ± 0.019	0.935 ± 0.051	0.963 ± 0.031	0.975 ± 0.001	0.972 ± 0.012	0.982 ± 0.004	0.979 ± 0.001
	GRAD-MATCH-ACF	0.977 ± 0.013	0.89 ± 0.041	0.974 ± 0.02	0.984 ± 0.009	0.979 ± 0.002	0.975 ± 0.009	0.979 ± 0.001
Deploy 4	Random	0.837 ± 0.013	0.87 ± 0.011	0.856 ± 0.022	0.873 ± 0.014	0.88 ± 0.025	0.915 ± 0.011	0.907 ± 0.021
	Match-Dist	0.856 ± 0.025	0.82 ± 0.047	0.873 ± 0.024	0.874 ± 0.018	0.874 ± 0.029	0.888 ± 0.025	0.907 ± 0.021
	TSDS	0.779 ± 0.042	0.82 ± 0.03	0.859 ± 0.015	0.889 ± 0.015	0.86 ± 0.013	0.874 ± 0.039	0.907 ± 0.021
	GLISTER	0.784 ± 0.056	0.849 ± 0.008	0.862 ± 0.029	0.87 ± 0.041	0.905 ± 0.004	0.917 ± 0.008	0.907 ± 0.021
	GRAD-MATCH	0.842 ± 0.023	0.83 ± 0.02	0.86 ± 0.012	0.832 ± 0.063	0.888 ± 0.004	0.911 ± 0.007	0.907 ± 0.021
	GRAD-MATCH-ACF	0.774 ± 0.065	0.81 ± 0.037	0.839 ± 0.057	0.893 ± 0.019	0.913 ± 0.013	0.901 ± 0.017	0.907 ± 0.021

Table 8: Selection performance for all methods on all GeoDE deployments for ResNet50 full-finetuning. Results are reported as (mean accuracy ± std). While three runs were intended for all configurations, † and ‡ indicate results based on $n = 1$ and $n = 2$ runs, respectively. – indicates instances where $n = 0$.

Deployment	Method	Fraction						
		5%	10%	25%	50%	75%	90%	100%
Deploy 1	Random	0.354 ± 0.062	0.452 ± 0.068	0.556 ± 0.002	0.544 ± 0.041	0.599 ± 0.018	0.6 ± 0.017	0.669 ± 0.094
	Match-Dist	0.399 ± 0.054	0.365 ± 0.017	0.445 ± 0.056	0.517 ± 0.018	0.542 ± 0.012	0.538 ± 0.041	0.669 ± 0.094
	TSDS	0.408 ± 0.028	0.443 ± 0.046	0.548 ± 0.023	0.618 ± 0.026	0.646 ± 0.002	0.644 ± 0.015	0.669 ± 0.094
	GLISTER	0.146 ± 0.036	0.184 ± 0.052	0.258 ± 0.018	0.505 ± 0.054	0.668 ± 0.024	0.719 ± 0.008	0.669 ± 0.094
	GRAD-MATCH	0.278 ± 0.013	0.426 ± 0.031	0.527 ± 0.012 ‡	–	–	–	0.669 ± 0.094
	GRAD-MATCH-ACF	0.471 ± 0.013	0.536 ± 0.008	–	–	–	–	0.669 ± 0.094
Deploy 2	Random	0.13 ± 0.023	0.166 ± 0.008	0.239 ± 0.014	0.235 ± 0.019	0.244 ± 0.029	0.28 ± 0.003	0.28 ± 0.046
	Match-Dist	0.152 ± 0.01	0.158 ± 0.014	0.182 ± 0.032	0.223 ± 0.04	0.225 ± 0.009	0.245 ± 0.014	0.28 ± 0.046
	TSDS	0.117 ± 0.016	0.186 ± 0.013	0.238 ± 0.012	0.255 ± 0.031	0.249 ± 0.011	0.28 ± 0.018	0.28 ± 0.046
	GLISTER	0.127 ± 0.004	0.158 ± 0.019	0.228 ± 0.033	0.288 ± 0.014	0.307 ± 0.004	0.306 ± 0.006	0.28 ± 0.046
	GRAD-MATCH	0.081 ± 0.011	0.126 ± 0.012	0.213 †	–	–	–	0.28 ± 0.046
	GRAD-MATCH-ACF	0.174 ± 0.016	0.247 ± 0.003	0.238 ± 0.012 ‡	–	–	–	0.28 ± 0.046
Deploy 3	Random	0.2 ± 0.022	0.282 ± 0.046	0.403 ± 0.031	0.401 ± 0.022	0.462 ± 0.015	0.473 ± 0.014	0.51 ± 0.13
	Match-Dist	0.165 ± 0.039	0.198 ± 0.021	0.278 ± 0.001	0.342 ± 0.01	0.35 ± 0.017	0.351 ± 0.034	0.51 ± 0.13
	TSDS	0.192 ± 0.038	0.276 ± 0.02	0.407 ± 0.044	0.508 ± 0.041	0.536 ± 0.057	0.568 ± 0.025	0.51 ± 0.13
	GLISTER	0.117 ± 0.017	0.129 ± 0.031	0.244 ± 0.031	0.417 ± 0.056	0.586 ± 0.044	0.626 ± 0.023	0.51 ± 0.13
	GRAD-MATCH	0.259 ± 0.008	0.34 ± 0.04	0.562 ± 0.024 ‡	–	–	–	0.51 ± 0.13
	GRAD-MATCH-ACF	0.295 ± 0.031	0.452 ± 0.046	0.606 †	–	–	–	0.51 ± 0.13
Deploy 4	Random	0.121 ± 0.011	0.159 ± 0.024	0.205 ± 0.013	0.225 ± 0.028	0.253 ± 0.019	0.266 ± 0.009	0.282 ± 0.037
	Match-Dist	0.132 ± 0.008	0.138 ± 0.013	0.158 ± 0.014	0.216 ± 0.017	0.227 ± 0.016	0.242 ± 0.037	0.282 ± 0.037
	TSDS	0.118 ± 0.005	0.162 ± 0.017	0.226 ± 0.018	0.253 ± 0.027	0.273 ± 0.006	0.287 ± 0.012	0.282 ± 0.037
	GLISTER	0.075 ± 0.005	0.074 ± 0.01	0.115 ± 0.019	0.2 ± 0.016	0.275 ± 0.014	0.309 ± 0.023	0.282 ± 0.037
	GRAD-MATCH	0.105 ± 0.005	0.127 ± 0.019	0.246 ± 0.032 ‡	–	–	–	0.282 ± 0.037
	GRAD-MATCH-ACF	0.157 ± 0.013	0.207 ± 0.018	–	0.275 †	–	–	0.282 ± 0.037

Table 9: Selection performance for all methods on all Auto Arborist deployments for ResNet50 full-finetuning. Results are reported as (mean accuracy $\pm$ std). While three runs were intended for all configurations, † and ‡ indicate results based on $n = 1$ and $n = 2$ runs, respectively. – indicates instances where $n = 0$.

A.4.5 LATENCY RUNTIMES ACROSS BASELINES

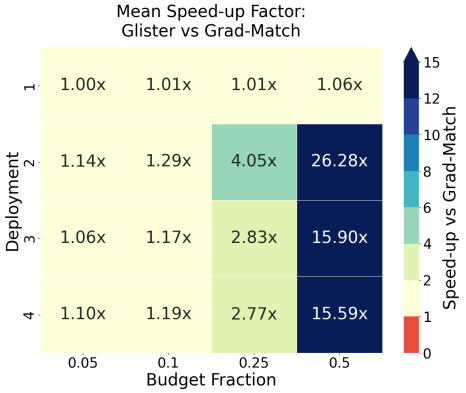

Figure 6: Execution speed-up of `Glister` relative to `Grad-Match` across all 4 iWildCam deployments for budget fractions $\in \{0.05, 0.1, 0.25, 0.5\}$. Each cell represents the mean speed-up factor across three random seeds. `Grad-Match` was unable to finish with larger budget fractions such as 0.75 and 0.9 due to numerical instability issues when running selection on our hardware (NVIDIA A100 GPUs), though this was not an issue for `Glister`. `Glister` provides comparable latency for smaller deployments/budget fractions, and offers up to a 26x speed-up improvement over `Grad-Match` for larger budget fractions.

Deployment	Method	Fraction					
		0.05	**0.1**	**0.25**	**0.5**	**0.75**	**0.9**
Deploy 1	GRAD-MATCH	9.53 ± 0.03	9.66 ± 0.09	9.55 ± 0.13	10.07 ± 0.08	11.05 ± 1.19	11.29 ± 0.11
	GRAD-MATCH-ACF	9.75 ± 0.34	9.77 ± 0.33	9.77 ± 0.23	10.06 ± 0.04	10.80 ± 0.09	11.82 ± 0.23
	GLISTER	9.53 ± 0.11	9.56 ± 0.16	9.48 ± 0.03	9.48 ± 0.33	9.25 ± 0.04	9.31 ± 0.02
Deploy 2	GRAD-MATCH	15.37 ± 0.37	17.69 ± 0.20	55.73 ± 0.59	356.51 †	1257.76 ± 8.35	–
	GRAD-MATCH-ACF	14.93 ± 0.06	15.59 ± 0.06	17.26 ± 0.15	19.07 ± 0.02	23.59 ± 0.07	28.67 ± 0.48
	GLISTER	13.51 ± 0.11	13.71 ± 0.19	13.75 ± 0.06	13.57 ± 0.32	13.22 ± 0.03	13.23 ± 0.11
Deploy 3	GRAD-MATCH	29.68 ± 0.43	32.89 ± 0.24	79.00 ± 0.54	432.66 ± 1.02	–	–
	GRAD-MATCH-ACF	28.90 ± 0.03	29.21 ± 0.03	30.79 †	36.27 ± 0.80	47.54 ± 0.88	58.43 ± 1.43
	GLISTER	27.89 ± 0.35	28.11 ± 0.16	27.90 ± 0.33	27.21 ± 0.63	26.98 ± 0.08	26.84 ± 0.26
Deploy 4	GRAD-MATCH	33.45 ± 1.54	36.21 ± 2.05	84.35 ± 3.27	462.20 ± 34.97	–	–
	GRAD-MATCH-ACF	32.35 ± 2.89	32.81 ± 2.16	31.95 ± 0.20	32.82 ± 0.39	33.13 ± 0.47	35.36 ± 1.38
	GLISTER	30.45 ± 1.44	30.42 ± 1.50	30.42 ± 1.43	29.65 ± 0.12	29.45 ± 0.58	29.46 ± 0.32

Table 10: Comparison of selection latency across iWildCam deployments. Results are reported as (mean $\pm$ std) execution time in minutes across varying budget fractions, measured on an NVIDIA A100 GPU. Three runs were intended for all configurations. † indicates instances with only 1 run, and – indicates instances where `GRAD-MATCH` failed to complete selection.

A.4.6 GRAD-MATCH VS GRAD-MATCH-ACF SUBSET LABEL DISTRIBUTIONS

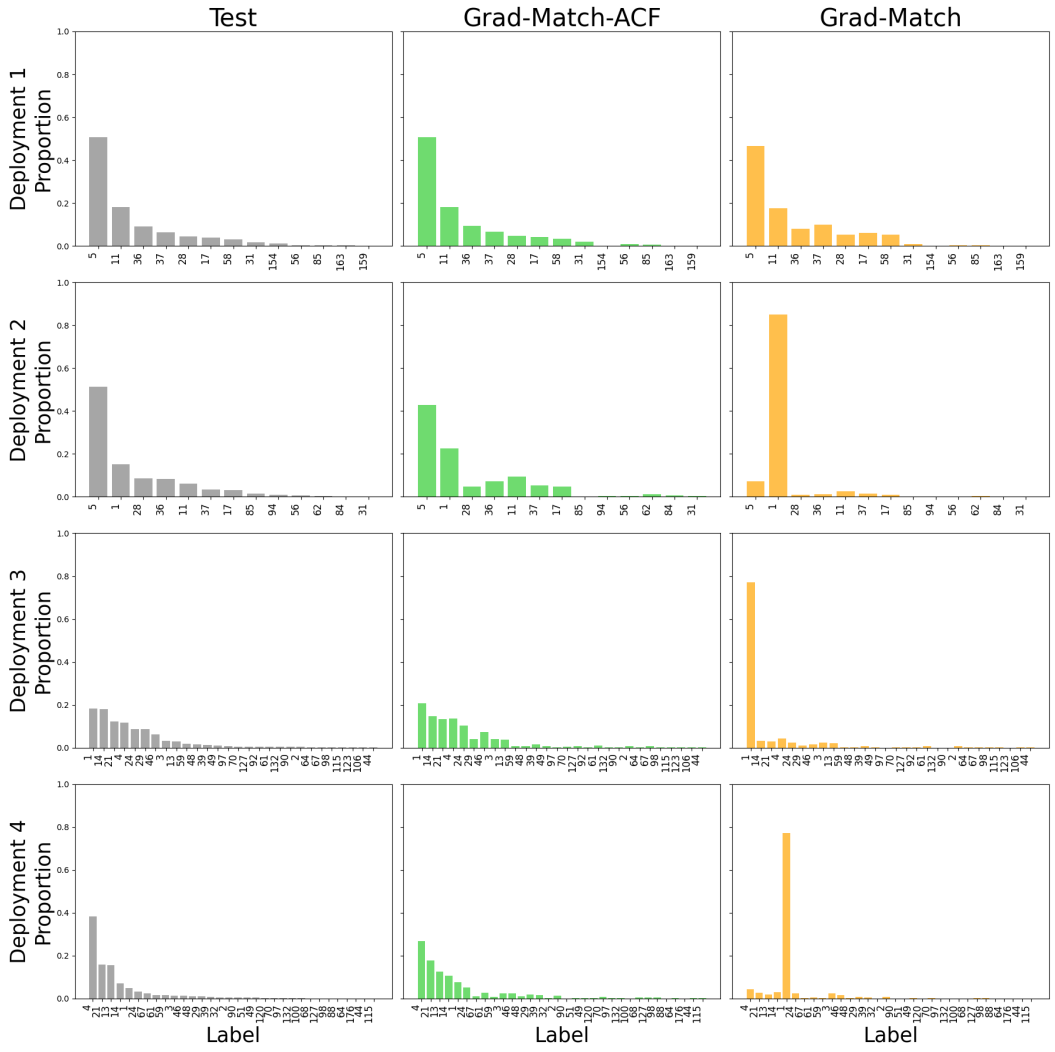

Figure 7: Comparison of label distributions for GRAD-MATCH-ACF and GRAD-MATCH across all iWildCam deployments for budget fraction of 0.25. Subplots show the label proportions of the target test set (column 1) versus subsets selected by GRAD-MATCH-ACF (column 2) and GRAD-MATCH (column 3). Clearly, GRAD-MATCH-ACF matches the target distribution more closely than GRAD-MATCH does.