LabelBench: A Comprehensive Framework for Benchmarking Adaptive Label-Efficient Learning

Jifan Zhang JIFAN@CS.WISC.EDU
Gregory Canal, Yinglun Zhu, Robert D. Nowak
University of Wisconsin, Madison, WI

Yifang Chen YIFANGC@CS.WASHINGTON.EDU
Arnav M. Das, Gantavya Bhatt,
Stephen Mussmann, Jeffrey Bilmes, Simon S. Du, Kevin Jamieson
University of Washington, Seattle, WA

Abstract

Labeled data are critical to modern machine learning applications, but obtaining labels can be expensive. To mitigate this cost, machine learning methods, such as transfer learning, semi-supervised learning and active learning, aim to be *label-efficient*: achieving high predictive performance from relatively few labeled examples. While obtaining the best label-efficiency in practice often requires combinations of these techniques, existing benchmark and evaluation frameworks do not capture a concerted combination of all such techniques. This paper addresses this deficiency by introducing LabelBench, a new computationally-efficient framework for joint evaluation of multiple label-efficient learning techniques. As an application of LabelBench, we introduce a novel benchmark of state-of-the-art active learning methods in combination with semi-supervised learning for fine-tuning pretrained vision transformers. Our benchmark demonstrates significantly better label-efficiencies than previously reported in active learning. LabelBench's modular codebase is open-sourced for the broader community to contribute label-efficient learning methods and benchmarks. The repository can be found at: `https://github.com/EfficientTraining/LabelBench`.

1. Introduction

Large pretrained models provide practitioners with strong starting points in developing machine-learning-powered applications (Radford et al., 2021; Yu et al., 2022; Kirillov et al., 2023). While zero-shot and few-shot predictions can provide solid baselines, linear probing (which freezes the model and trains a layer on top) and fine-tuning based on human annotation yield significantly better performance (Radford et al., 2021; Yu et al., 2022). Label-efficient learning, which aims to achieve high predictive performance with fewer labels, has received much attention lately due to the high annotation cost of labeling large-scale datasets.

Transfer learning, semi-supervised learning (SSL) and active learning (AL) all study different aspects of label-efficient learning. Modern transfer learning leverages large general-purpose models pretrained on web-scale data and fine-tunes the model to fit application-specific examples. Semi-supervised learning utilizes a large set of unlabeled examples to estimate the underlying data distribution and more efficiently learn a good model. Active learning incrementally and adaptively annotates only those examples deemed to be informative

by the model. To date, however, no existing literature has studied all the above methods under a single unified benchmarking framework for fine-tuning large pretrained models.

In this paper, we present LabelBench, a comprehensive benchmarking framework for *label-efficient* learning. Our framework tackles computational challenges that arise when scaling these techniques to large neural network architectures. Specifically, incorporating AL involves periodically retraining the model based on the latest labeled examples. While repeatedly training small convolutional neural networks is practically feasible (Sener and Savarese, 2017; Ash et al., 2019, 2021; Beck et al., 2021; Zhan et al., 2022; Lüth et al., 2023), retraining large-scale models is extremely compute intensive, making large scale AL experiments prohibitively expensive. Inspired by selection-via-proxy (Coleman et al., 2019), we propose lightweight retraining schemes (based on freezing all but the last layer of large pretrained models) for the purpose of data selection and labeling, but evaluate final model performance with a single end-to-end fine-tuning. This technique yields a ten-fold reduction in training cost, and reaps most of the label-efficiency gains of using AL.

To showcase the power of our framework, we conduct experiments that benchmark multiple deep active learning algorithms in combination with semi-supervised learning and large pretrained models; compared to existing active learning literature, our experiments yield state-of-the-art label-efficiencies.

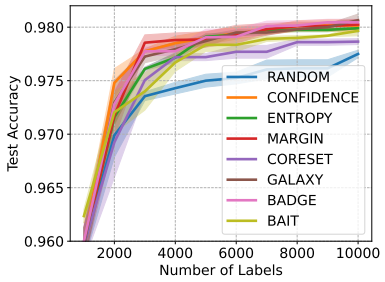


Figure 1: Generalization Acc, CIFAR-10 with SSL trainer

Result Highlights When comparing to existing deep active learning literature, under similar annotation budgets, we observe a more than 75% (compared to 30%) and 60% savings (compared to 5%) in annotation cost on CIFAR-10 (Figure 1) and CIFAR-100 (Figure 3)[1]. This improvement is further demonstrated in our experiments on ImageNet (Figure 3(a, d)). Under any fixed annotation budget, our experiments suggest active learning algorithms can consistently boost test accuracy by more than 1.2% and pool accuracy by more than 5%. Compared to the previous best results in this setting Emam et al. (2021), our results yield at least 10% higher test accuracy.

Related works In appendix A, we give a thorough review on existing label-efficient learning literature with an emphasis on the interplay among large pretrained models, semi-supervised learning and active learning.

2. Label Efficient Fine-tuning Framework

We propose a framework for label-efficient learning consisting of three widely-adopted components in modern deep learning: initialization with a large pretrained model, data annotation, and fine-tuning on downstream tasks. Our framework supports traditional AL, but also takes advantage of large pretrained models and SSL to further improve the

1. As reported in seminal papers and common benchmarks such as Ash et al. (2019) (Figure 16), Ash et al. (2021) (Figure 10), Beck et al. (2021) (Figure 1) and Lüth et al. (2023) (Figure 6-8), they see less than 25% reductions in annotation cost on CIFAR-10. CIFAR-100 results are rarely reported since algorithms do not generally see active gains on the dataset.

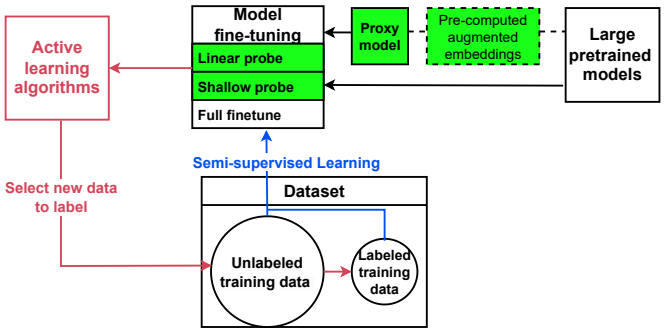


Figure 2: A modular framework consisting of pretrained models, SSL trainer and AL strategies.

label-efficiency. As shown in Figure 2, our framework starts with a large pretrained model as initialization. Data annotation follows a closed-loop procedure, where one starts with a pool of unlabeled examples in the beginning and iteratively gathers more human annotations. At any iteration, given a partially labeled pool we utilize semi-supervised training to obtain the best performing model. Informed by this trained model, an active learning strategy selects unlabeled examples it deems the most informative and sends those examples to be labeled. At the end of the iteration, the newly annotated labels are recorded into the dataset. The greatest challenge in implementing this framework comes from incorporating large-scale model training while meeting a limited computational budget. Unlike past deep active learning methods (Sener and Savarese, 2017; Ash et al., 2019, 2021) that utilize smaller neural network architectures (e.g., ResNet-18), the computational cost of fine-tuning large pretrained models at every iteration of the data collection loop is a significant burden. To address this challenge, we propose using a *selection-via-proxy* (Coleman et al., 2019) approach (Section 2.1), along with additional code optimization to improve the computational and memory efficiencies for large-scale datasets (Appendix C.2) .

2.1 Selection via Proxy

During each iteration of data collection, there are three potential strategies in fine-tuning the large pretrained model: fine-tuning the model end-to-end, training only a linear probe (Alain and Bengio, 2016), and training a nonlinear probe with a shallow neural network. In the latter two strategies, the learner freezes the pretrained encoder and attaches to it a less computationally intensive model at the output (i.e. a linear classifier or shallow network). This can greatly reduce the computational cost of retraining, but often the final model does not perform as well as one in which the full model is retrained.

To better trade-off between retraining/inference cost and the final model performance, we propose a *selection-via-proxy* approach, which is inspired by Coleman et al. (2019). In the referenced work, a less computationally intensive proxy is created by carefully scaling down the original model architecture and training for fewer epochs. In our framework, we exploit a more straightforward approach by employing the linear probe and shallow network models as potential proxies. During every iteration of the data annotation loop, the learner only retrains the proxy model, which informs the selection of unlabeled examples to be annotated. After collecting a sufficient amount of labeled examples or reaching the labeling budget limits, the learner then switches to end-to-end fine-tuning at the last batch to further boost the

performance of the final model. As a result, selection-by-proxy reduces the GPU time and back-propagation-induced cost by *more than ten-fold* (Table 2).

3. Benchmarking Active Learning Algorithms

To demonstrate the utility of our framework, we conduct experiments comparing popular deep AL strategies in combination with large pretrained models and SSL. Our results presented in Section 3.2 show better label efficiencies than state-of-the-art deep AL literature. Moreover, we discuss the accuracy gap by using selection-via-proxy under different settings.

3.1 Experiment Setup

Here we give a brief introduction of our setups and leave details to Appendix F. Our benchmark studies the following annotation procedure:

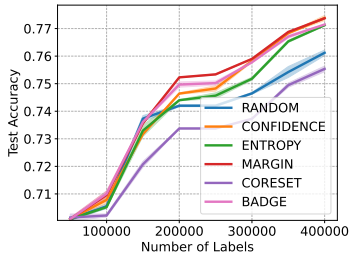
1. **Initial large pretrained model and performance metrics.** We use pretrained CLIP (Radford et al., 2021) and CoCa (Yu et al., 2022) with the ViT-B32 architecture as image encoders. For end-to-end fine-tuning, we attach the image encoder with a zero-shot prediction linear head. For proxy models, we initialize it with random weights.
2. **Adaptive annotation loop.** We collect the first batch of labels by sampling uniformly at random and then iterate over the following steps to annotate batches of examples.
   - **Model training.** At the beginning of each iteration, the dataset is partially labeled. We use the SSL technique FlexMatch (Zhang et al., 2021) to fine-tune the vision transformer or train the proxy model from scratch.
   - **Data selection and annotate.** Given the trained model, we use a data selection strategy to select unlabeled examples for annotation. We benchmark against prevalent active learning algorithms such as confidence sampling (Lewis, 1995), margin sampling (Scheffer et al., 2001), entropy sampling (Settles, 2009), BADGE (Ash et al., 2019) and GALAXY (Zhang et al., 2022).
3. **Final Model.** After the annotation budget is exhausted, regardless if the proxy model is used for selection or not, we fine-tune the pretrained CLIP or CoCa model end-to-end by FlexMatch on the collected labeled examples as well as the remaining unlabeled examples.

Datasets and Performance Metrics We test on standard datasets including CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009), as well as more realistic and challenging datasets including iWildCam Beery et al. (2021) and fMoW Christie et al. (2018). We report the (1) generalization performance which aims to learn accurate models that generalize beyond examples in the pool while spending limited budget on oracle annotation, such as human labeling. (2) pool performance aims to annotate all examples in the pool with a limited labeling budget.

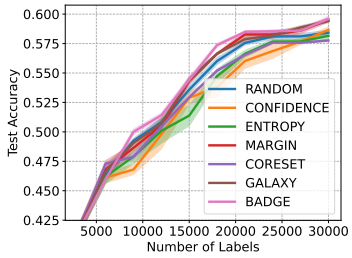
3.2 Results and Discussion

In this section we present a summary of performance evaluations on various combinations of models and AL strategies. See Appendix H for additional results

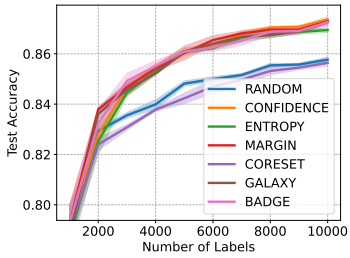
End-to-End Fine-Tuning First, we summarize our results when end-to-end fine-tuning the large pretrained model at every iteration of the data collection loop. When comparing the results of AL strategies to random sampling, we consistently see label efficiency gains across all datasets (Figures 3). Such label efficiency gain is especially significant on pool performances, with active learning strategies saving up to 50% of the annotation budget for



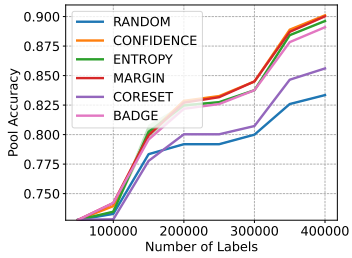
(a) Generalization Acc, ImageNet



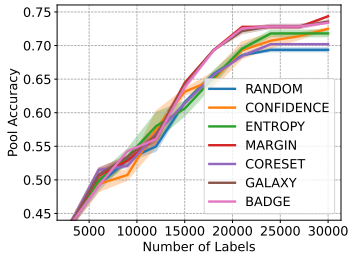
(b) Generalization Accuracy, fMoW



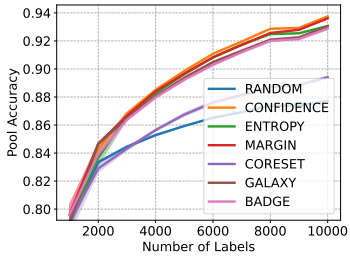
(c) Generalization Acc, CIFAR-100



(d) Pool accuracy on ImageNet

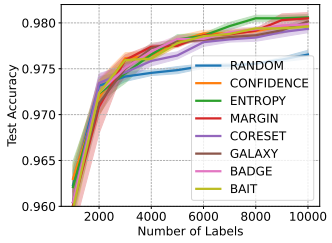


(e) Pool accuracy on fMoW

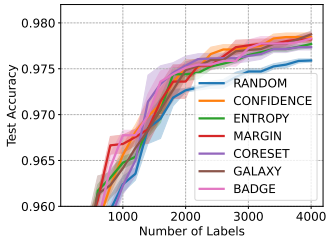


(f) Pool accuracy on CIFAR-100

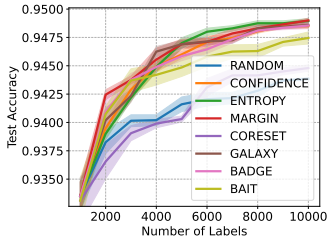
Figure 3: Performances of different data selection strategies on ImageNet, fMoW and CIFAR-100 with AL + FlexMatch + CLIP ViT-B32. Each result of fMoW and CIFAR-100 is averaged over four trials and each result of ImageNet is over two trials due to limited computing resources. The confidence intervals are based on standard error.



(a) Selection with shallow network, evaluation on fine-tuning, batch size of 1000



(b) Selection with shallow network, evaluation on fine-tuning, batch size of 200



(c) Generalization accuracy on CIFAR-10 with passive training.

Figure 4: (a) and (b): Generalization performance on CIFAR-10 when using different proxy models for data selection. (c): Generalization performance when using supervised trainer instead of SSL (we use selection with end-to-end fine-tuning here). Each result is averaged over four trials with standard error shown as confidence interval.

ImageNet (Figure 3(d)). Notably, similar gains can also be observed pretrained CoCa model as shown in Figure 8. In general, when comparing performance of different AL strategies on (near) balanced datasets (ImageNet, CIFAR-10, CIFAR-100 and fMoW), margin sampling surprisingly performs among the top in terms of both generalization and pool accuracy. On imbalanced dataset like iWildcam (see Figure 6), GALAXY demonstrates a clear advantage in terms of generalization and pool macro F1 scores. Finally, CORESET underperforms in

	Test Accuracy			Pool Accuracy		
	Fine-tune	Shallow Network	Linear Probe	Fine-tune	Shallow Network	Linear Probe
Confidence	**77.38 ± .13**	76.96 ± .12	**77.23 ± .10**	**90.11 ± .01**	**88.93 ± .01**	**89.01 ± .02**
Entropy	77.12 ± .04	76.63 ± .11	76.81 ± .01	89.62 ± .01	88.33 ± .02	88.70 ± .003
Margin	77.37 ± .04	**77.15 ± .01**	77.09 ± .10	90.02 ± .03	88.75 ± .03	88.84 ± .01
Coreset	75.54 ± .15	75.33 ± .17	75.54 ± .08	85.60 ± .01	84.84 ± .03	84.52 ± .01
BADGE	77.15 ± .02	76.83 ± .04	76.85 ± .20	89.10 ± .04	87.64 ± .02	88.20 ± .03
Random	76.12 ± .14	76.12 ± .14	76.12 ± .14	83.35 ± .01	83.35 ± .01	83.35 ± .01
Best	77.38 ± .13	77.15 ± .01	77.23 ± .10	90.11 ± .01	88.93 ± .01	89.01 ± .02

Table 1: Selection-via-proxy results of ImageNet using CLIP ViT-B32. The results are evaluated with 400,000 labels. Confidence intervals are standard errors based on two trials.

most cases. These findings underscore the importance of further evaluating AL strategies on realistic datasets.

Importance of AL + SSL + Large Pretrained Models Comparing to existing literature of AL + SSL (Lüth et al., 2023; Chan et al., 2021; Mittal et al., 2019; Siméoni et al., 2021) and AL + large pretrained models (Tamkin et al., 2022), our experiment yields the largest percentage of annotation cost savings to reach the same level of accuracy as random sampling. This reinforces the importance of studying the combination of active learning, semi-supervised learning and large pretrained models under an unified framework.

We compare the effect of using SSL versus regular passive training when combined with AL + large pretrained models. By comparing Figure 1 with Figure 4(c), we see that the accuracy gains from each of AL and SSL become less significant than the gains of each of them alone. However, the combination of both AL + SSL provides the highest accuracy boost. Moreover, in terms of label savings in reaching the same accuracy, we find AL is the most efficient when combining with SSL and large pretrained models, indicating the increasing importance of studying active learning in the new era of large pretrained models.

Selection-via-proxy We also study the effectiveness and drawbacks of selection-via-proxy by comparing it against *selection with end-to-end fine-tuning*. As shown in Tables 1, 4, 7, selection-via-proxy performs similarly to selection with end-to-end fine-tuned models in terms of *test accuracy*. On the other hand, we found that selection-via-proxy is slightly less effective than selection with fine-tuning in terms of *pool accuracy* - there is an approximately 1% reduction in performance in fMoW and ImageNet experiments.

To further investigate the label-efficiency tradeoff of the two methods, in Figure 1(a) and Figure 4(a), we plot their performances respectively after collecting every batch of labels. The gap between selection-via-proxy and selection with fine-tuning diminishes quickly with more iterations of data selection. As shown in Figure 4(b), we can further close the gap in lower-budget settings by collecting more rounds of annotations with smaller batches. Indeed, to achieve 97.75% accuracy (random sampling's accuracy with 10,000 labels), selection-via-proxy only requires 2750 labels (with batch size of 200), comparable to selection with fine-tuning's label-efficiency in Figure 4(a). We note that smaller batches are only computationally feasible for selection-via-proxy, as one can only end-to-end fine-tune a small number of times under a limited budget.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *ArXiv*, abs/1906.03671, 2019.

Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Sham M. Kakade. Gone fishing: Neural active learning with fisher embeddings. In *Neural Information Processing Systems*, 2021.

Myles Bartlett, Sara Romiti, Viktoriia Sharmanska, and Novi Quadrianto. Okapi: Generalising better by making statistical matches match. *arXiv preprint arXiv:2211.05236*, 2022.

Nathan Beck, Durga Sivasubramanian, Apurva Dani, Ganesh Ramakrishnan, and Rishabh K. Iyer. Effective evaluation of deep active learning on image classification tasks. *ArXiv*, abs/2106.15324, 2021.

Sara Beery, Arushi Agarwal, Elijah Cole, and Vighnesh Birodkar. The iwildcam 2021 competition dataset. *arXiv preprint arXiv:2105.03494*, 2021.

David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HklkeR4KPB`.

Zalán Borsos, Marco Tagliasacchi, and Andreas Krause. Semi-supervised batch active learning via bilevel optimization. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3495–3499, 2021. doi: 10.1109/ICASSP39728.2021.9414206.

Zhaowei Cai, Avinash Ravichandran, Paolo Favaro, Manchen Wang, Davide Modolo, Rahul Bhotika, Zhuowen Tu, and Stefano Soatto. Semi-supervised vision transformers at scale. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=7a2IgJ7V4W`.

Yao-Chun Chan, Mingchen Li, and Samet Oymak. On the marginal benefit of active learning: Does self-supervision eat its cake? In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3455–3459, 2021. doi: 10.1109/ICASSP39728.2021.9414665.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020.

Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018.

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch active learning at scale. *Advances in Neural Information Processing Systems*, 34:11933–11944, 2021.

Cody Coleman, Edward Chou, Julian Katz-Samuels, Sean Culatana, Peter Bailis, Alexander C Berg, Robert Nowak, Roshan Sumbaly, Matei Zaharia, and I Zeki Yalniz. Similarity search for efficient active learning and search of rare concepts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6402–6410, 2022.

Cody A. Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter D. Bailis, Percy Liang, Jure Leskovec, and Matei A. Zaharia. Selection via proxy: Efficient data selection for deep learning. *ArXiv*, abs/1906.11829, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020.

Zeyad Ali Sami Emam, Hong-Min Chu, Ping-Yeh Chiang, Wojciech Czaja, Richard Leapman, Micah Goldblum, and Tom Goldstein. Active learning at the imagenet scale. *arXiv preprint arXiv:2111.12880*, 2021.

Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan Ö Arık, Larry S Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling cost. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 510–526. Springer, 2020.

Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8175–8195. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/hacohen22a.html`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Siyu Huang, Tianyang Wang, Haoyi Xiong, Jun Huan, and Dejing Dou. Semi-supervised active learning with temporal output discrepancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3447–3456, October 2021.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, 2021.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European Conference on Computer Vision*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Manuel Lagunas, Brayan Impata, Victor Martinez, Virginia Fernandez, Christos Georgakis, Sofia Braun, and Felipe Bertrand. Transfer learning for fine-grained classification using semi-supervised learning and visual transformers. *arXiv preprint arXiv:2305.10018*, 2023.

David D Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA, 1995.

Carsten T. Lüth, Till J. Bungert, Lukas Klein, and Paul F. Jaeger. Toward realistic evaluation of deep active learning algorithms in image classification. *ArXiv*, abs/2301.10625, 2023.

Sudhanshu Mittal, Maxim Tatarchenko, Özgün Çiçek, and Thomas Brox. Parting with illusions about deep active learning. *ArXiv*, abs/1912.05361, 2019.

Stephen Mussmann, Julia Reisler, Daniel Tsai, Ehsan Mousavi, Shayne O'Brien, and Moises Goldszmidt. Active learning with expected error reduction. *arXiv preprint arXiv:2211.09283*, 2022.

Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning, 2020.

Friedrich Pukelsheim. *Optimal design of experiments*. SIAM, 2006.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Comput. Surv.*, 54(9), oct 2021. ISSN 0360-0300. doi: 10.1145/3472291. URL https://doi.org/10.1145/3472291.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis: 4th International Conference, IDA 2001 Cascais, Portugal, September 13–15, 2001 Proceedings 4*, pages 309–318. Springer, 2001.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv: Machine Learning*, 2017.

Burr Settles. Active learning literature survey. 2009.

Oriane Siméoni, Mateusz Budnik, Yannis Avrithis, and Guillaume Gravier. Rethinking deep active learning: Using unlabeled data at model training. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1220–1227, 2021. doi: 10.1109/ICPR48806.2021.9412716.

Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. Flava: A foundational language and vision alignment model. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15617–15629, 2021.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 596–608. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/06964dce9addb1c5cb5d6e3d9838f733-Paper.pdf.

Alex Tamkin, Dat Nguyen, Salil Deshpande, Jesse Mu, and Noah Goodman. Active learning helps pretrained models learn the intended task. *arXiv preprint arXiv:2204.08491*, 2022.

Jesper E. van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, Feb 2020. ISSN 1573-0565. doi: 10.1007/s10994-019-05855-6. URL https://doi.org/10.1007/s10994-019-05855-6.

Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *ArXiv*, abs/2208.10442, 2022a.

Yidong Wang, Hao Chen, Yue Fan, Wang SUN, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie, and Yue Zhang. USB: A unified semi-supervised learning benchmark for classification. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022b. URL https://openreview.net/forum?id=QeuwINa96C.

Ziting Wen, Oscar Pizarro, and Stefan B. Williams. Training from a better start point: Active self-semi-supervised learning for few labeled samples. 2022.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971, 2022.

Zhen Xing, Qi Dai, Han Hu, Jingjing Chen, Zuxuan Wu, and Yu-Gang Jiang. Svformer: Semi-supervised video transformer for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18816–18826, 2023.

Lewei Yao, Runhu Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. Filip: Fine-grained interactive language-image pre-training. *ArXiv*, abs/2111.07783, 2021.

Ofer Yehuda, Avihu Dekel, Guy Hacohen, and Daphna Weinshall. Active learning through a covering lens. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22354–22367. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/8c64bc3f7796d31caa7c3e6b969bf7da-Paper-Conference.pdf`.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *Trans. Mach. Learn. Res.*, 2022, 2022.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel C. F. Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. Florence: A new foundation model for computer vision. *ArXiv*, abs/2111.11432, 2021.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1204–1213, 2021.

Xueying Zhan, Qingzhong Wang, Kuan-Hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B. Chan. A comparative survey of deep active learning. *ArXiv*, abs/2203.13450, 2022.

Bowen Zhang, Yidong Wang, Wenxin Hou, HAO WU, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18408–18419. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/995693c15f439e3d189b06e89d145dd5-Paper.pdf`.

Jifan Zhang, Julian Katz-Samuels, and Robert Nowak. Galaxy: Graph-based active learning at the extreme. *arXiv preprint arXiv:2202.01402*, 2022.

Xiaojin Zhu, John D. Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, 2003.

Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

Appendix A. Related Work

Large pretrained models have demonstrated a wide range of generalization abilities on downstream language and vision tasks. Most of these models are trained on web-scale data with supervised (Kolesnikov et al., 2019; Dosovitskiy et al., 2020; Zhai et al., 2021) or self-supervised techniques (Radford et al., 2021; Jia et al., 2021; Yuan et al., 2021; Singh et al., 2021; Yao et al., 2021; Wang et al., 2022a; Yu et al., 2022). While these models are powerful by themselves, adapting them to applications often requires transfer learning by fine-tuning on human annotated examples. Below we survey existing literature on label-efficient learning with an emphasis on the interplay among large pretrained models, semi-supervised learning and active learning.

A.1 Semi-supervised Training

In traditional supervised learning the model is only trained on the set of *labeled* examples, while in SSL the model training is also informed by the remaining *unlabeled* examples in the pool. Intuitively, SSL leverages the assumption that examples lying "nearby" to one another should belong to the same class, and therefore during training the model is encouraged to produce the same model output for these examples (for an overview of SSL we refer the interested reader to Zhu (2005); van Engelen and Hoos (2020); Ouali et al. (2020)). Broadly speaking, modern SSL methods implement this principle using a combination of *Consistency Regularization* — where model outputs of neighboring examples are regularized to be similar — and *Pseudo Labeling* — where unlabeled examples that the model is confident on are assigned artificial labels to supplement supervised training Sohn et al. (2020); Berthelot et al. (2020). In our pipeline we implement one such SSL method called FlexMatch Zhang et al. (2021), due to its simplicity and effectiveness.

Semi-supervised Training of Large Pretrained Models The application of SSL to fine-tuning large pretrained models is a nascent area of research. Cai et al. (2022) pioneered the application of SSL methods to large-scale vision transformers by using a multi-stage pipeline of pretraining followed by supervised fine-tuning and finally semi-supervised fine-tuning. Lagunas et al. (2023) apply this pipeline to a fine-grained classification e-commerce task and demonstrate improved performance compared to standard supervised training. SSL training on transformer architectures has also been successfully applied to video action recognition Xing et al. (2023). USB Wang et al. (2022b) is a benchmark that includes SSL evaluations on large pretrained models such as ViT; however, it does not incorporate AL into its pipeline, as we do here.

A.2 Active Learning

If we have a large pool of unlabeled examples and a limited labeling budget, one must select a subset of the data for label annotation. Various strategies have been proposed to identify an informative subset that produces a good model from a limited budget of labels. Experimental design (Pukelsheim, 2006) studies the setting where this subset is chosen before any annotations are observed. Pool-based active learning (Settles, 2009) examines iterative adaptive annotation: labels from previously annotated examples can be used to determine which examples to choose for annotation in the next iteration. Active learning algorithms

are generally designed to maximize one or both of the intuitive concepts of *uncertainty* and *diversity*. Uncertainty, measured in a variety of ways (Settles, 2009), refers to the uncertainty of a trained model for the label of a given point (Lewis, 1995; Scheffer et al., 2001), while diversity refers to selecting points with different features (Sener and Savarese, 2017). Many algorithms maximize a combination of these two concepts (Ash et al., 2019, 2021; Citovsky et al., 2021; Zhang et al., 2022).

Active Learning for Fine-Tuning Large Pretrained Models Recent literature in deep active learning has started to utilize large pretrained models for large-scale datasets. Coleman et al. (2022) proposes a computationally efficient method to annotate billion-scale datasets by actively labeling examples only in the neighborhood of labeled examples in the SimCLR (Chen et al., 2020) embedding space. Tamkin et al. (2022) found that uncertainty sampling yields larger annotation saving for large pretrained models than for traditional ResNet. LabelBench serves as a more comprehensive large-scale benchmark for these studies, where we combine SSL training in our framework. We further take into account the expensive cost of fine-tuning large pretrained models at every iteration of active data collection.

In addition, numerous papers have utilized self-supervised or unsupervised learning methods to initialize their models (Siméoni et al., 2021; Chan et al., 2021; Wen et al., 2022; Lüth et al., 2023) on the unlabeled datasets. However, their methods do not utilize existing large pretrained models.

Active Learning with Semi-supervised Training Since AL and SSL seek to maximize model performance using only a minimal budget of labeled points, it is natural to combine both techniques to maximize label efficiency. This practice dates back to Zhu et al. (2003), which labels examples that minimize expected classification error in a Gaussian Field SSL model. In the context of deep learning, Lüth et al. (2023); Chan et al. (2021); Mittal et al. (2019); Siméoni et al. (2021) benchmark various AL methods in SSL settings. Huang et al. (2021) develops a hybrid AL/SSL approach for computer vision tasks, and Gao et al. (2020) develops a consistency-based AL selection strategy that is naturally compatible with SSL methods. Borsos et al. (2021) approaches AL in the context of SSL as a problem of dataset summarization, and demonstrates improved performance on keyword detection tasks. Hacohen et al. (2022); Yehuda et al. (2022) both use FlexMatch as a baseline SSL method in their AL experiments, further corroborating our choice of FlexMatch in our own pipeline.

Appendix B. Call for Contribution and Future Work

We call on the broader community to further develop different components of LabelBench: below we provide suggested contributions for each directory of our framework. Our codebase is modular, so one can easily start on any single component without a thorough understanding of the entire codebase.

Trainer. Our experiments demonstrate the potential label savings provided by combining active learning with FlexMatch. To expand upon these results, it would be valuable to develop a benchmark of additional SSL methods, evaluated in combination with AL and large pretrained models. To do so, one could instantiate specific SSL trainer classes that inherit our template SSL trainer.

Active Learning Strategy. As exhibited by our results, combining AL with SSL and large pretrained models results in highly accurate and label-efficient models that demonstrate clear gains over passive learning. Therefore, we call on the AL community to evaluate their active selection algorithms under our more comprehensive and up-to-date benchmark. Additionally, besides several widely-used AL strategies we have implemented here, there exists a large suite of active selection methods in the literature that could potentially be implemented and evaluated Ren et al. (2021).

Datasets and Metrics. In future benchmarks built on top of LabelBench, we plan on incorporating datasets with distribution shift evaluation data. We believe this is a valuable future direction that aligns with many real-world scenarios. Introducing tasks beyond image classification is also an important next step, e.g., natural language processing tasks, vision tasks such as object detection and segmentation, and generative modeling in both vision and language applications.

Models. With the computational speed-ups afforded by selection-via-proxy and our pre-computation steps, we can scale the model to ViT-L and ViT-H architectures (Dosovitskiy et al., 2020) without incurring significant computational costs. In addition, there are also other choices of proxy. For example, LORA (Hu et al., 2021), which injects the trainable rank decomposition matrices in the intermediate layers instead of the final header is popular in natural language processing. Moreover, in developing benchmarks that better reflect real-world applications, contributors could implement a blend of selection-via-proxy and end-to-end fine-tuning during example selection, instead of fine-tuning at every iteration or only once at the end of labeling.

Appendix C. Details on Label Efficient Fine-tuning Framework

C.1 Computational cost on Selection via Proxy

	End-to-end Fine-Tune		Shallow Network (proxy)	
Training Stage	GPU Hours	AWS Dollars	GPU Hours	AWS Dollars
Precomputation	0	\$0	5	\$15
Retraining	1900	\$5700	57	\$180
Final Model	100	\$300	100	\$300
Total	2000	\$6300	162	\$495

Table 2: Estimated cost of neural network training for ImageNet experiments when collecting 600,000 labels with 20 iterations (batches of 30,000 labels per iteration). Here we display the total cost of running 12 trials with CLIP ViT-B32 and FlexMatch SSL training (Zhang et al., 2021). All AWS dollars are based on on-demand rates of EC2 P3 instances.

C.2 Codebase

Our codebase consists of five components: datasets, model, training strategy (for supervised and semi-supervised training), active learning strategy and metrics. We would like to highlight the following advantages of our implementation:

- **Modularity.** As shown in Figure 5, adding any new instance, such as a new dataset or training strategy, simply involves implementing a new function. This allows future contributors to solely focus on any isolated component without a thorough understanding of the entire repository.
- **Self-report mechanism.** We include configuration files of all experiment setups. In addition, we keep track of all experiment results in the results directory for fair comparisons. Researchers are encouraged to self-report their research findings by submitting pull requests to our repository.
- **Significant speed-up of existing AL implementation.** Running some AL algorithms can be time-prohibitive when scaled to large datasets with large numbers of classes. In our implementation, we speed up popular active learning algorithms such as BADGE (Ash et al., 2019) and BAIT (Ash et al., 2021) by orders of magnitude in comparison to existing implementations (Appendix G).

```
# Add a new dataset.
@register_dataset(
    "my_dataset", MULTI_CLASS)
def get_dataset(...):
    ...

# Add a new SSL Algorithm.
Class MyTrainer(SemiTrainer):
    def train_step(img,
                   aug_img,
                   ...):
        ...
```

Figure 5: Our modular codebase allows one to work solely in one directory without a thorough knowledge of the entire codebase. Implementing a new dataset or semi-supervised learning trainer is as easy as implementing a single function.

Appendix D. Definition of Metrics

For K labels, define the confusion matrix C where $C_{i,j} = \Pr(Y = i, \hat{Y} = j)$.

The balanced accuracy is

$$\frac{1}{K} \sum_{i=1}^{K} \frac{C_{i,i}}{\sum_{j=1}^{K} C_{i,j}} \tag{1}$$

Define the precision and recall for a class i as

$$P_i = \frac{C_{i,i}}{\sum_{j=1}^{K} C_{j,i}} \tag{2}$$

$$R_i = \frac{C_{i,i}}{\sum_{j=1}^{K} C_{i,j}} \tag{3}$$

Then, the macro F1 score is

$$\frac{1}{K} \sum_{i=1}^{K} \frac{2}{\frac{1}{P_i} + \frac{1}{R_i}} \tag{4}$$

Appendix E. Active Learning Strategies

We describe the active learning setup and introduce some basic active learning strategies in this section.

We start by describing the active learning setups. The learner starts with a large pool of unlabeled examples $U = \{x_i\}_{i \in [n]}$ and a small fraction of labeled examples L, where each example x comes from the input space $\mathcal{X}$ with some unknown label y belonging to labeling space $\mathcal{Y}$. At the beginning of every batch, adhering to a certain active learning strategy, the algorithm adaptively selects new examples to label (i.e., moving the labeled examples from U to L) based on the current model h. We use $h_\theta(x)$ to denote the predicated softmax vector; we also use $[h_\theta(x)]_i$ to denote the i-th coordinate of the prediction. The model h is then retrained based on the updated dataset L, U with a certain training strategy. The ultimate goal is to use as small of a labeling budget as possible to achieve some desired performance (e.g., small error).

Below we introduce some active learning strategies that have been used in our experiments.

- Confidence (Lewis, 1995): An uncertainty-based active learning strategy that selects examples with the least confidence score in terms of the top predicated class, i.e., $\max_i [h_\theta(x)]_i$.
- Entropy (Settles, 2009): An uncertainty-based active learning strategy that selects examples with the highest entropy of the predicted distribution $h_\theta(x)$.
- Margin (Scheffer et al., 2001): An uncertainty-based active learning strategy that selects examples with the smallest prediction margin between the top-2 classes, i.e., $[h_\theta(x)]_{i^\star} - \max_{i \neq i^\star} [h_\theta(x)]_i$, where $i^\star = \arg\max [h_\theta(x)]_i$.
- BADGE (Ash et al., 2019): An active learning strategy that incorporates both uncertainty and diversity in sampling using k-means++ in the hallucinated gradient space.
- BAIT (Ash et al., 2021): An active learning strategy that incorporates both uncertainty and diversity by sampling from a Fisher-based selection objective using experimental design. BAIT can be viewed as a more general version of BADGE.
- GALAXY (Zhang et al., 2022): A graph-based active learning strategy that incorporates both uncertainty and diversity by first building a graph and then adaptively sampling examples on the shortest path of the graph.

Appendix F. Set-up details

F.1 Performance Metrics details

We report results on the following two tasks of label-efficient learning.

- **Label-efficient generalization** aims to learn accurate models that generalize beyond examples in the pool while spending limited budget on oracle annotation, such as human labeling. We refer to the models' performances on test data as *generalization performance*. In this paper, we report performances on in-distribution test data (drawn from the same distribution as the pool). As will be mentioned in Section B, one may be able to extend this benchmark to distribution shift cases.
- **Label-efficient annotation** aims to annotate all examples in the pool with a limited labeling budget. When the dataset is partially labeled by a human, a model trained based on existing annotations can serve as a pseudo annotation tool that labels the rest of the unlabeled examples. We refer to the percentage of labels (both human annotated and pseudo labels) that agree with ground-truth labels as the *pool performance*. Examples of label-efficient annotation applications include product cataloging, categorizing existing userbases, etc.

To quantify performance, we use the standard accuracy for (near) balanced datasets, and balanced accuracy and macro F1 score for imbalanced datasets. Balanced accuracy and macro F1 score are measured as unweighted averages of per-class accuracies and per-class F1 scores, respectively.

F.2 Datasets details

We first test on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009), all of which are standard datasets used in previous AL and SSL papers. To further evaluate LabelBench on more realistic datasets, we also test on iWildCam Beery et al. (2021) and fMoW Christie et al. (2018), parts of the WILDS benchmark Koh et al. (2021). To the best of our knowledge, only a handful of existing studies, such as Tamkin et al. (2022); Mussmann et al. (2022); Bartlett et al. (2022), have evaluated label-efficient algorithms on these datasets, albeit under different experimental setups. The WILDS benchmark was originally intended to represent distribution shifts faced in the wild (i.e., OOD test sets); here we limit our evaluation to in-domain (ID) test set performance as an initial exploratory step. Using these datasets provides several advantages: 1) Both of them are highly imbalanced. 2) Fine-tuning pretrained large-scale models on them is more challenging than on ImageNet (e.g., ID test accuracy on fMoW is 73.3% Wortsman et al. (2022) when fine-tuning ViT-L14 end-to-end). 3) Unlike ImageNet and CIFAR10, whose examples are gathered by querying search engines with human validation, iWildCam and fMoW gather labels directly from human annotators, which aligns more closely with our pool-based active learning setting.

F.3 Hyper-parameter tuning

Adhering to the guidelines proposed by Lüth et al. (2023), we are transparent about our method configuration, which many active learning studies fail to report. For each dataset, we utilize a separate validation set, typically with size around 10% of the training pool. We begin the process by adjusting the hyper-parameters on a subset of the training data, which

is randomly queried and constitutes around 10% of the total training pool. The selection of hyper-parameters is mainly based on the criterion of achieving the highest accuracy on the validation set. These hyper-parameters are then consistently applied in all subsequent data collection batches and across varied experimental settings (e.g., experiments with different batch sizes). While it's arguable that this fixed hyper-parameter approach may not always yield optimal results, it is practically suitable in real-world scenarios and allows for fair comparison in this paper.

Appendix G. Speeding Up Existing Active Learning Algorithms

Notation. Let $U = \{x_1, ..., x_N\}$ denote the set of N unlabeled examples and K denote the number of classes in a dataset. For each $i \in [N]$, we further use $p_i \in \mathbb{R}^K$ and $\widehat{y}_i \in [K]$ to denote the predictive probability and predictive label respectively on example x_i. Lastly, we use $v_1, ..., v_N \in \mathbb{R}^d$ to denote the penultimate layer output of a neural network where d is the number of dimensions.

Implementation of BADGE The current implementation of BADGE (`https://github.com/JordanAsh/badge`) explicitly computes gradient embeddings g_i for each unlabeled example x_i. In particular, each g_i is a Kd-dimensional vector and can be computed via vectorizing $q_i v_i^\top$ where $q_i \in \mathbb{R}^K$ is defined as

$$q_{i,j} = \begin{cases} 1 - p_{i,j} & \text{if} \quad j = \widehat{y}_i \\ -p_{i,j} & \text{otherwise} \end{cases}$$

During each iteration of BADGE (B iterations in total for each batched selection of B examples), the dominating computation lies in computing the ℓ-2 distance between N pairs of gradient embeddings. Currently, this is implemented by naively computing $\|g_i - g_j\|_2$ with an $O(Kd)$ complexity each.

We instead use the following decomposition:

$$\begin{aligned} \|g_i - g_j\|_2 &= \|g_i\|_2 + \|g_j\|_2 - 2 g_i^\top g_j \\ &= \|q_i\|_2 \cdot \|v_i\|_2 + \|q_j\|_2 \cdot \|v_j\|_2 - 2 \cdot (q_i^\top q_j) \cdot (v_i^\top v_j). \end{aligned}$$

where the last expression can be computed with $O(K + d)$ complexity, effectively reducing the computational time by an order of magnitude. In our ImageNet experiment, this means a 512-fold reduction in computation time.

Implementation of BAIT The current implementation of BAIT (`https://github.com/JordanAsh/badge`) uses an apparent approximation to the Fisher information for a low-rank approximation. Note that the multi-class Fisher information defined in appendix A.2 of Ash et al. (2021) is not full-rank, causing numerical problems with taking the inverse. In our implementation, we multiply the Fisher information by a orthogonal transformation that removes a dimension to make the Fisher information full-rank.

Define the orthogonal transformation as $T \in \mathbb{R}^{k \times (k-1)}$ that removes the null space along the direction of the vectors of all ones. Using the notation of appendix A.2 of Ash et al. (2021), we can let:

$$P = T^\top(\text{diag}(\pi) - \pi\pi^\top)T \tag{5}$$
$$U = x \otimes P^{1/2} \tag{6}$$

Then,

$$UU^\top = (x \otimes P^{1/2})(x \otimes P^{1/2})^\top \tag{7}$$
$$= xx^\top \otimes P \tag{8}$$
$$= I(x; W) \tag{9}$$

and thus we can use the Woodbury matrix identity for faster matrix inverse updates.

Because the Fisher information matrix is very large, we perform PCA to reduce the dimensionality.

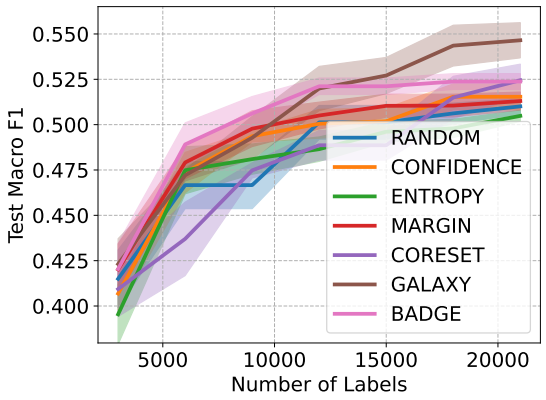
In Ash et al. (2021), an expensive greedy strategy is used to build the selected set. Our implementation is based on "swaps", that is, removing an example and adding an example. In particular, we begin with an initially randomly drawn selected set, then one-by-one propose an example to remove and propose to add the best example from a random sample of 10 unlabeled examples. If the proposed swap would improve the objective function, the swap is performed.

Appendix H. More results

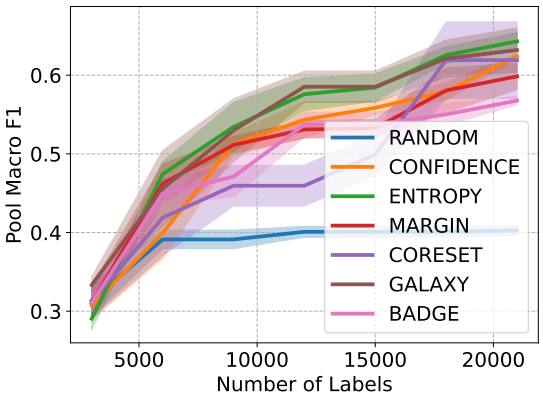
Here we provide more experimental results. Notice that we only implement BAIT in CIFAR-10 due to its high computational and memory complexity – For d embedding dimension and K-classes, its memory complexity is $\mathcal{O}(K^2d^2)$. In addition, we omit GALAXY for ImageNet as mentioned in the main paper due to its expensive computational complexity on large datasets.

We also note that results on iWildcam have much higher standard error and variance than other datasets. We attribute this observation to the imbalance nature of the dataset, which may increase the variance if some rare classes have no annotated examples at all.

H.1 End-to-end Fine-tuning

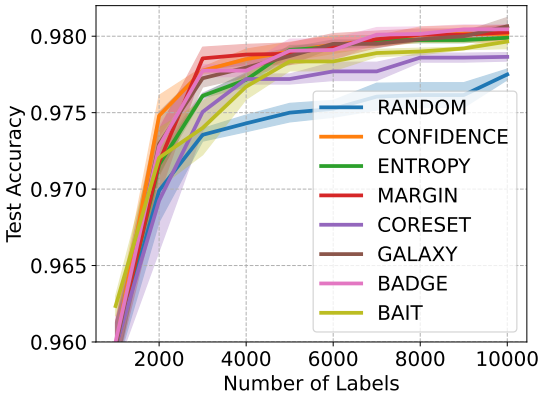


(a) Generalization macro F1 on iWildcam, AL + FlexMatch + Pretrained CLIP ViT-B32

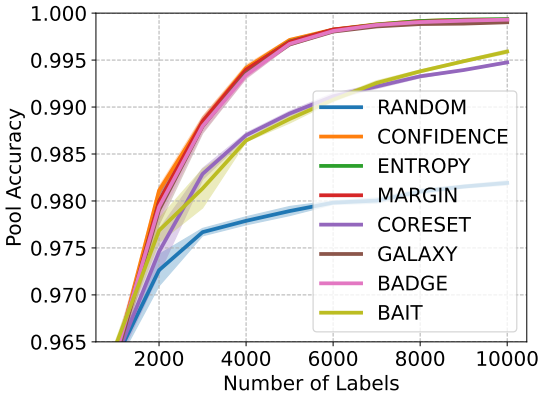


(b) Pool macro F1 on iWildcam

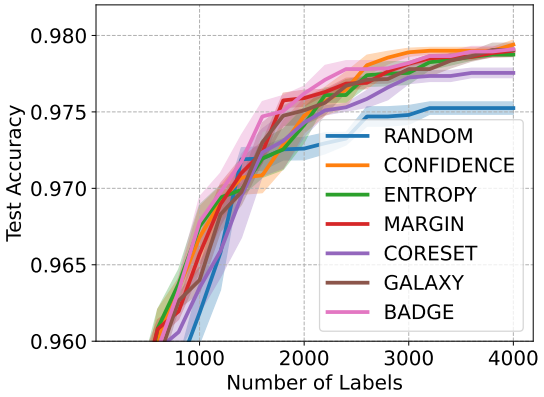
Figure 6: End-to-end fine-tune performance on iWildcam, AL + FlexMatch + Pretrained CLIP ViT-B32



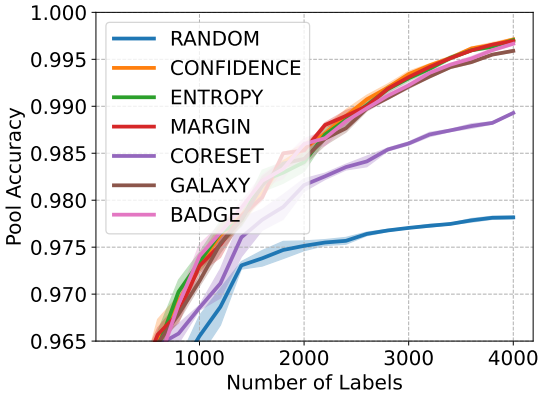
(a) Generalization Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 1000



(b) Pool Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 1000

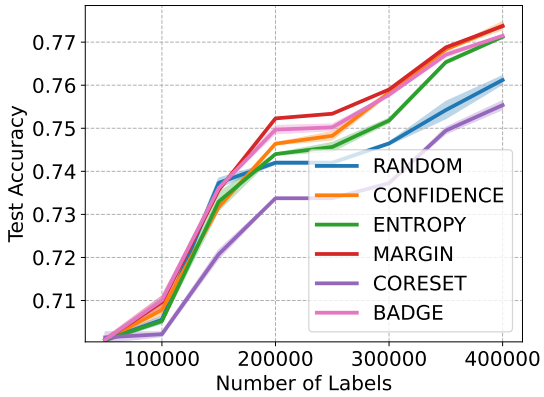


(c) Generalization Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 200

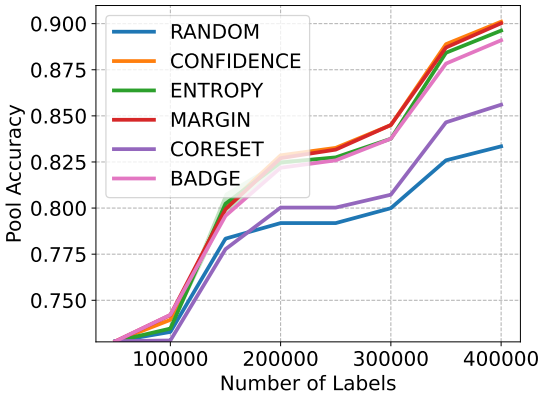


(d) Pool Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 200

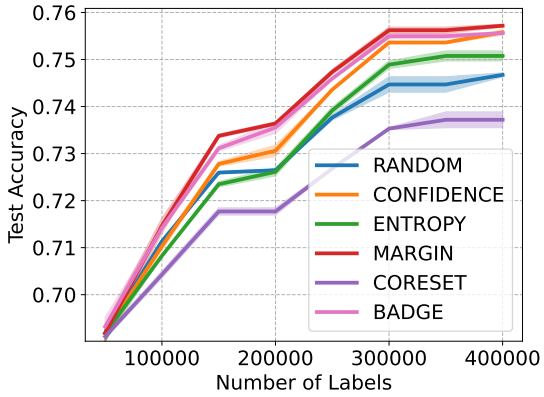
Figure 7: End-to-end fine-tune performance on CIFAR-10.



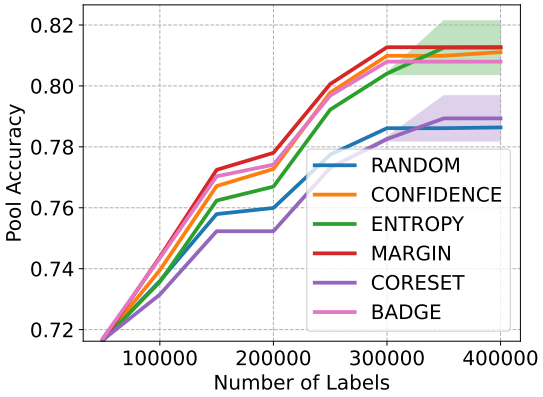
(a) Generalization Accuracy on ImageNet, AL + FlexMatch + Pretrained CLIP ViT-B32



(b) Pool Accuracy on ImageNet, AL + FlexMatch + Pretrained CLIP ViT-B32

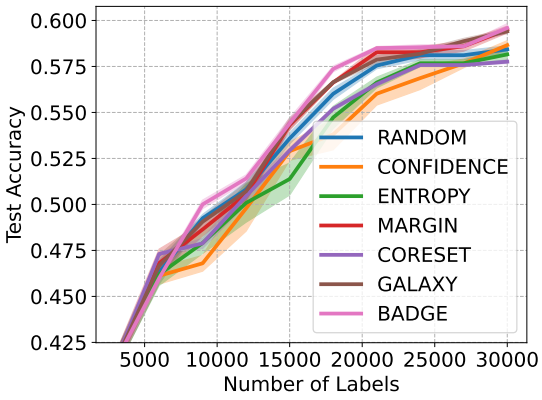


(c) Generalization Accuracy on ImageNet, AL + FlexMatch + Pretrained CoCa ViT-B32

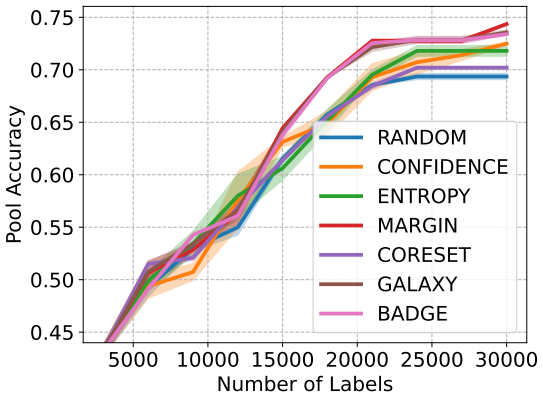


(d) Pool Accuracy on ImageNet, AL + FlexMatch + Pretrained CoCa ViT-B32

Figure 8: End-to-end fine-tune performance on ImageNet.

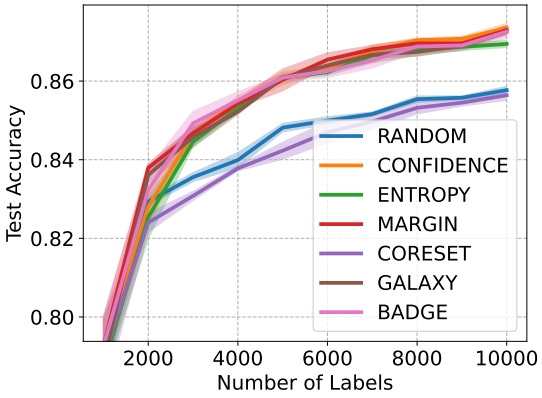


(a) Generalization Accuracy on FMoW, AL + FlexMatch + Pretrained CLIP ViT-B32

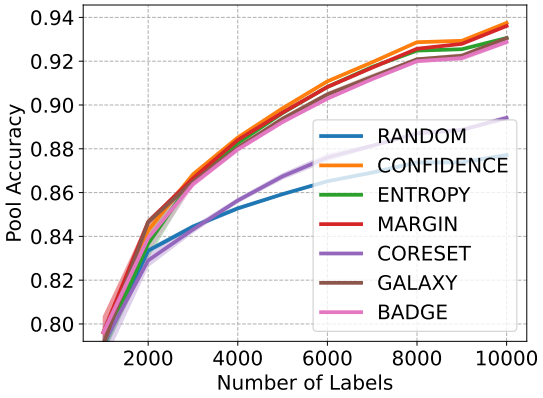


(b) Pool Accuracy on FMoW, AL + FlexMatch + Pretrained CLIP ViT-B32

Figure 9: End-to-end fine-tune performance on FMoW.



(a) Generalization Accuracy on CIFAR-100, AL + FlexMatch + Pretrained CLIP ViT-B32

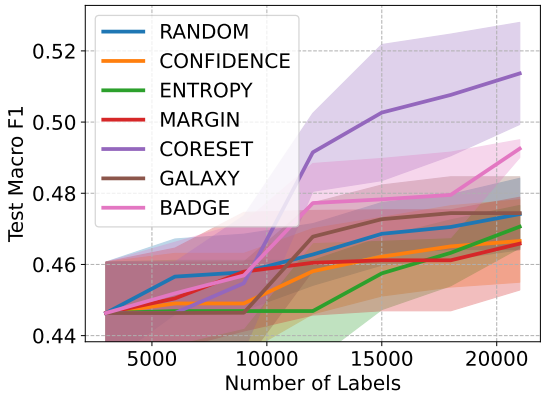


(b) Pool Accuracy on CIFAR-100, AL + FlexMatch + Pretrained CLIP ViT-B32

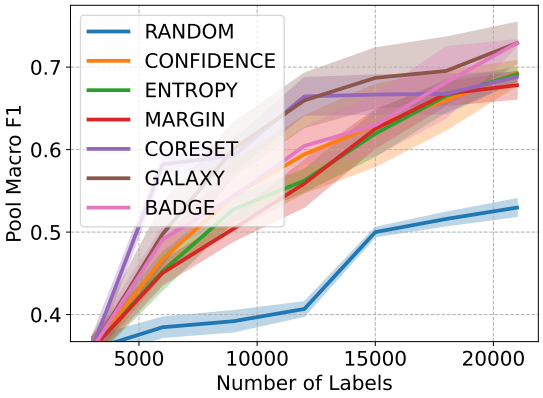
Figure 10: End-to-end fine-tune performance on CIFAR-100.

H.2 Learning Linear Probes

Note this section differs from the selection-via-proxy plots (Figures 4(a,b)) in that we are measuring the raw performance of linear probes instead of having an additional evaluation step by fine-tuning the model end-to-end.

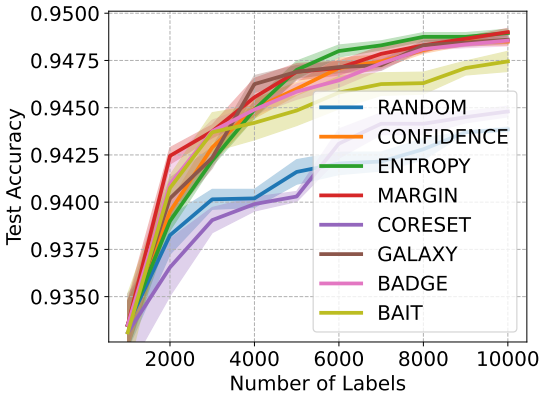


(a) Generalization macro F1 on iWildcam, AL + FlexMatch + Pretrained CLIP ViT-B32

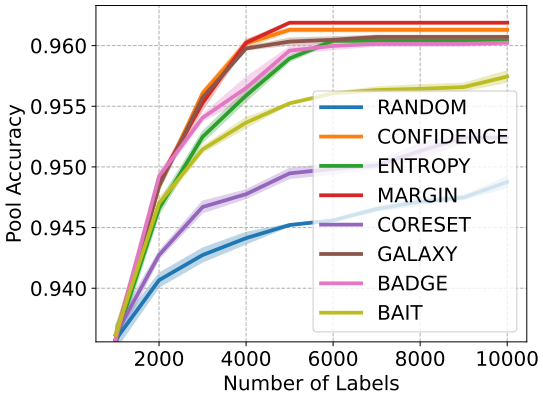


(b) Pool macro F1 on iWildcam

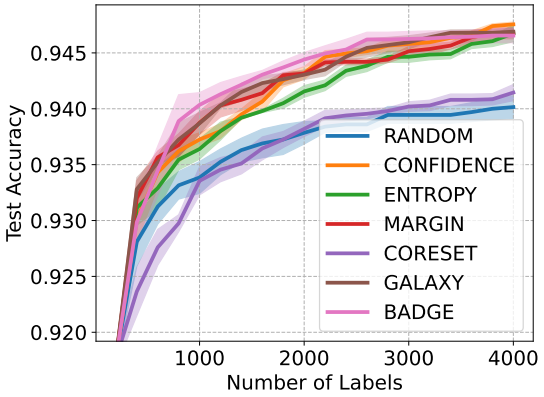
Figure 11: Linear probe performance on iWildcam, AL + FlexMatch + Pretrained CLIP ViT-B32



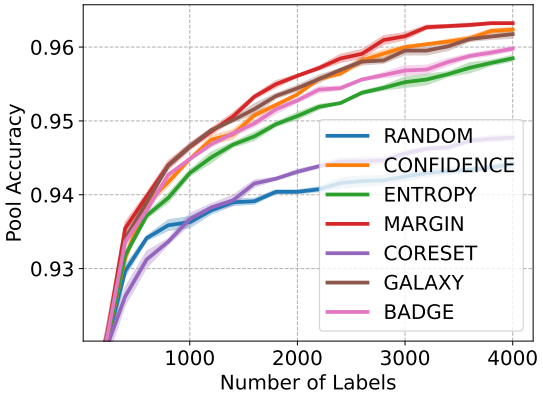
(a) Generalization Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 1000



(b) Pool Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 1000

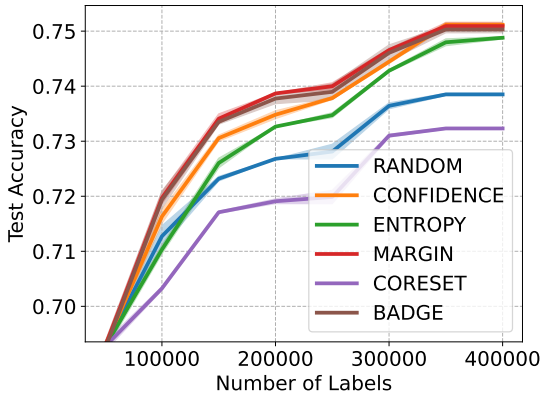


(c) Generalization Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 200

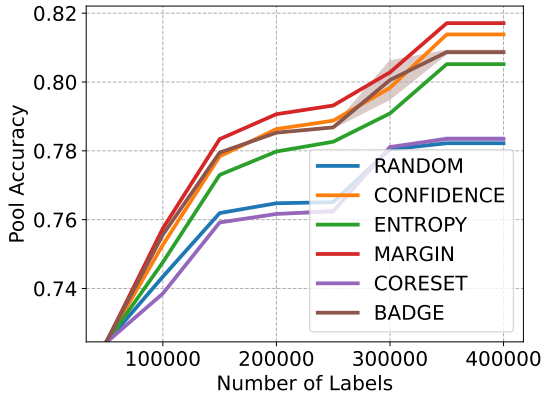


(d) Pool Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 200

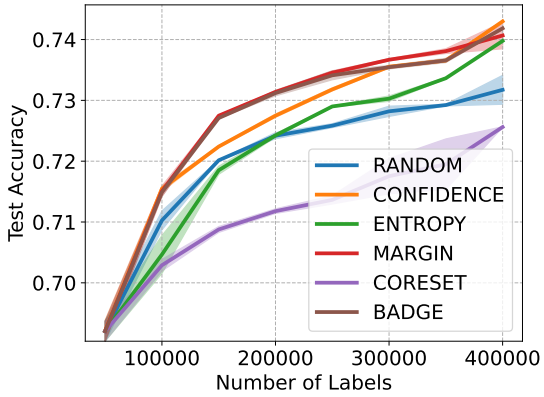
Figure 12: Linear probe performance on CIFAR-10.



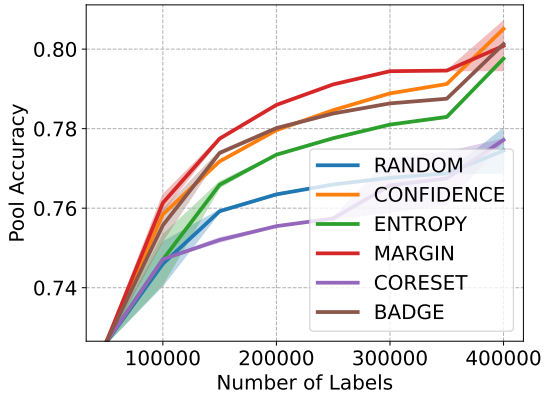
(a) Generalization Accuracy on ImageNet, AL + FlexMatch + Pretrained CLIP ViT-B32



(b) Pool Accuracy on ImageNet, AL + FlexMatch + Pretrained CLIP ViT-B32

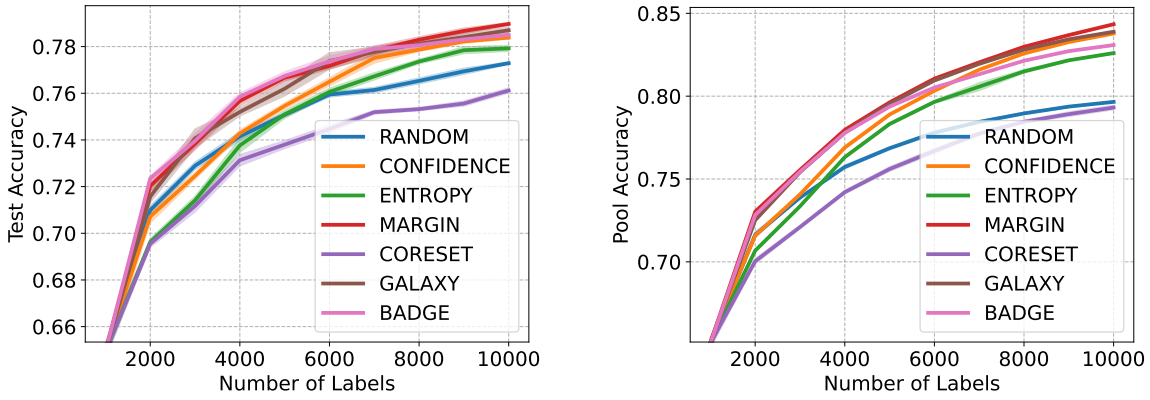


(c) Generalization Accuracy on ImageNet, AL + FlexMatch + Pretrained CoCa ViT-B32



(d) Pool Accuracy on ImageNet, AL + FlexMatch + Pretrained CoCa ViT-B32

Figure 13: Linear probe performance on ImageNet.



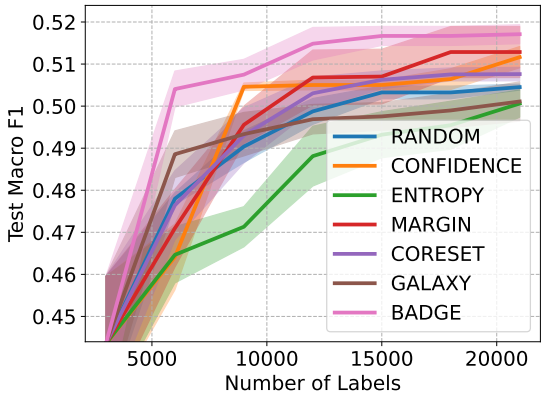
(a) Generalization Accuracy on CIFAR-100, AL + FlexMatch + Pretrained CLIP ViT-B32

(b) Pool Accuracy on CIFAR-100, AL + FlexMatch + Pretrained CLIP ViT-B32

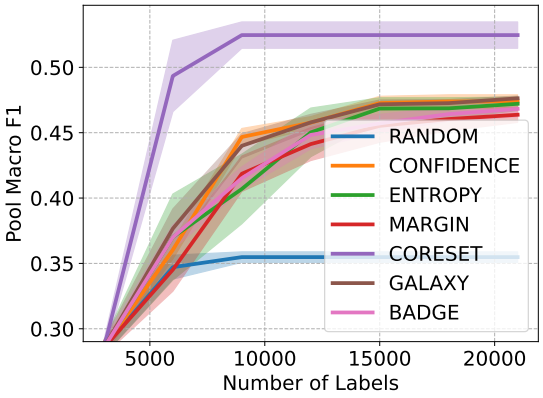
Figure 14: Linear probe performance on CIFAR-100.

H.3 Learning a Shallow Neural Network

Note this section differs from the selection-via-proxy plots (Figures 4(a,b)) in that we are measuring the raw performance of shallow networks instead of having an additional evaluation step by fine-tuning the model end-to-end.

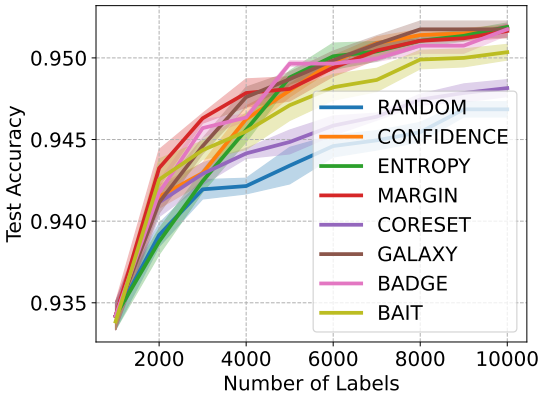


(a) Generalization macro F1 on iWildcam, AL + FlexMatch + Pretrained CLIP ViT-B32

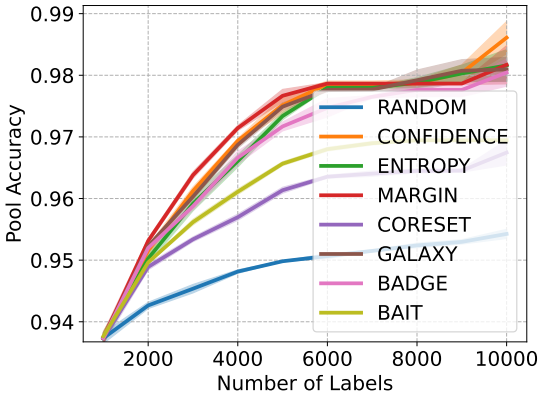


(b) Pool macro F1 on iWildcam

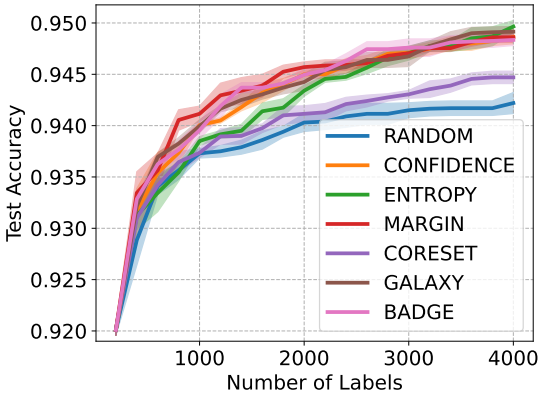
Figure 15: Shallow network performance on iWildcam, AL + FlexMatch + Pretrained CLIP ViT-B32



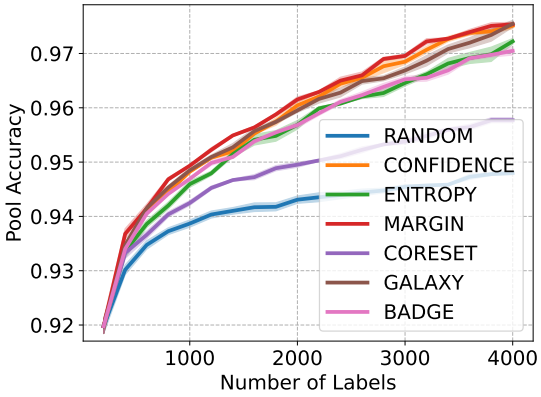
(a) Generalization Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 1000



(b) Pool Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 1000

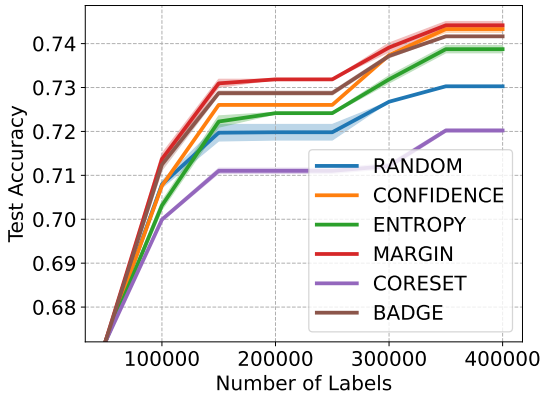


(c) Generalization Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 200

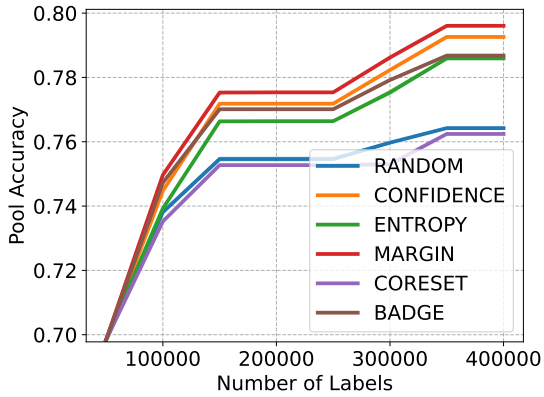


(d) Pool Accuracy on CIFAR-10, AL + FlexMatch + Pretrained CLIP ViT-B32, Batch Size = 200

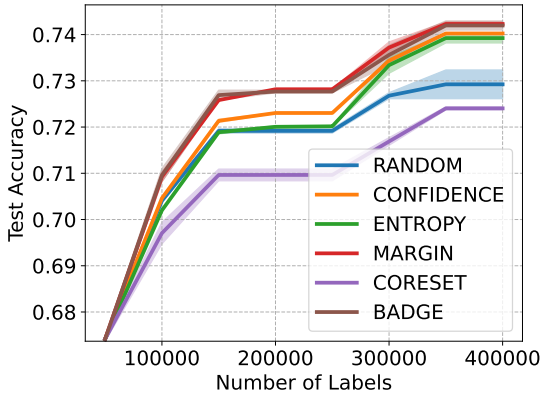
Figure 16: Shallow network performance on CIFAR-10.



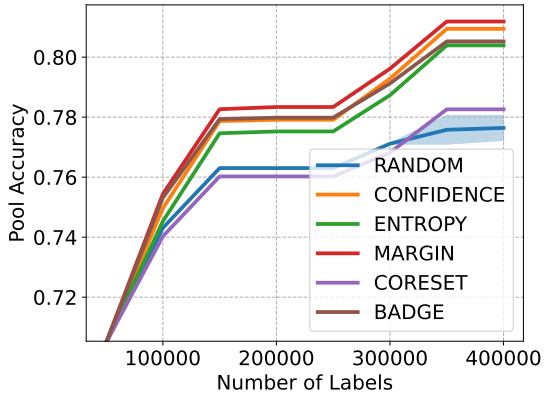
(a) Generalization Accuracy on ImageNet, AL + FlexMatch + Pretrained CLIP ViT-B32



(b) Pool Accuracy on ImageNet, AL + FlexMatch + Pretrained CLIP ViT-B32

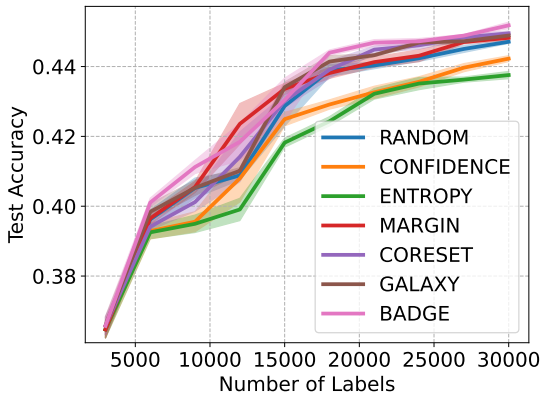


(c) Generalization Accuracy on ImageNet, AL + FlexMatch + Pretrained CoCa ViT-B32

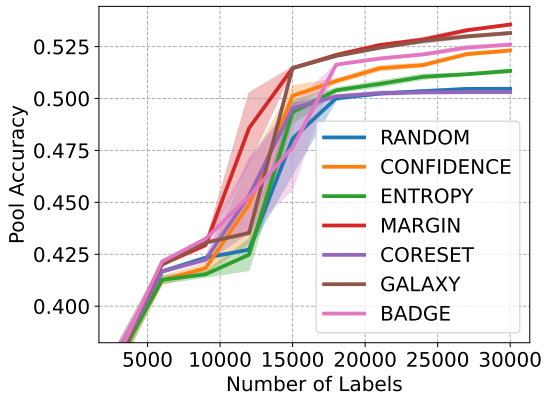


(d) Pool Accuracy on ImageNet, AL + FlexMatch + Pretrained CoCa ViT-B32

Figure 17: Shallow network performance on ImageNet.

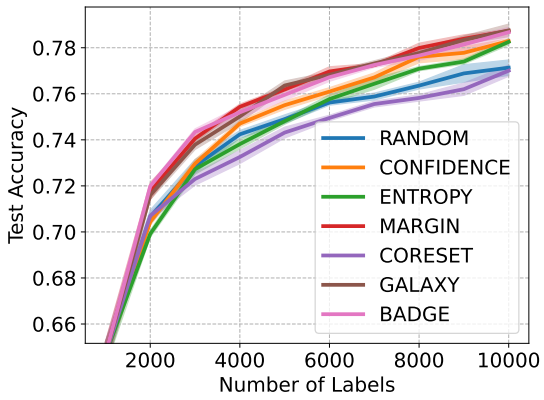


(a) Generalization Accuracy on FMoW, AL + FlexMatch + Pretrained CLIP ViT-B32

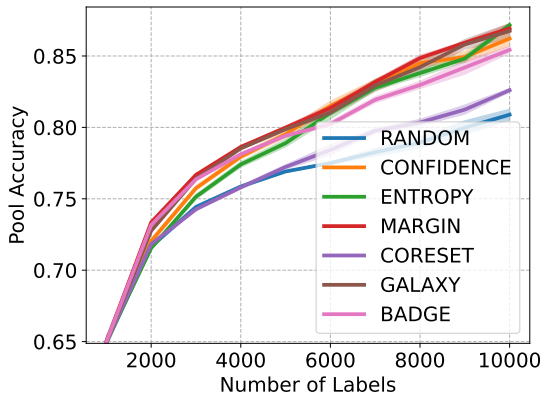


(b) Pool Accuracy on FMoW, AL + FlexMatch + Pretrained CLIP ViT-B32

Figure 18: Shallow network performance on FMoW.



(a) Generalization Accuracy on CIFAR-100, AL + FlexMatch + Pretrained CLIP ViT-B32



(b) Pool Accuracy on CIFAR-100, AL + FlexMatch + Pretrained CLIP ViT-B32

Figure 19: Shallow network performance on CIFAR-100.

H.4 Comparison Between Selection-Via-Proxy and Selection with Fine-tuning

	Test Accuracy			Pool Accuracy		
	Fine-tune	Shallow Network	Linear Probe	Fine-tune	Shallow Network	Linear Probe
Confidence	$75.58 \pm .08$	$75.29 \pm .03$	$\mathbf{75.43 \pm .04}$	$81.11 \pm .02$	$80.35 \pm .01$	$80.46 \pm .03$
Entropy	$74.95 \pm .08$	$74.69 \pm .06$	$74.91 \pm .01$	$80.40 \pm .11$	$79.76 \pm .08$	$79.89 \pm .02$
Margin	$\mathbf{75.65 \pm .15}$	$\mathbf{75.38 \pm .06}$	$75.38 \pm .12$	$\mathbf{81.26 \pm .03}$	$\mathbf{80.44 \pm .08}$	$\mathbf{80.50 \pm .06}$
Coreset	$73.44 \pm .05$	$73.31 \pm .12$	$72.95 \pm .18$	$78.14 \pm .06$	$77.79 \pm .02$	$77.51 \pm .14$
BADGE	$75.49 \pm .12$	$75.26 \pm .12$	$75.37 \pm .15$	$80.78 \pm .05$	$80.20 \pm .02$	$80.28 \pm .04$
Random	$74.61 \pm .15$	$74.61 \pm .15$	$74.61 \pm .15$	$78.64 \pm .01$	$78.64 \pm .01$	$78.64 \pm .01$
Best	$75.65 \pm .15$	$75.38 \pm .06$	$75.43 \pm .04$	$81.26 \pm .03$	$80.44 \pm .08$	$80.50 \pm .06$

Table 3: Selection-via-proxy results of ImageNet using CoCa ViT-B32. The results are evaluated with 400,000 labels. Confidence intervals are standard errors based on two trials.

	Test Accuracy			Pool Accuracy		
	Fine-tune	Shallow Network	Linear Probe	Fine-tune	Shallow Network	Linear Probe
Confidence	$97.84 \pm .07$	$97.85 \pm .05$	$97.86 \pm .05$	$99.92 \pm .02$	$99.67 \pm .02$	$99.63 \pm .02$
Entropy	$97.89 \pm .08$	$97.87 \pm .14$	$97.82 \pm .06$	$\mathbf{99.93 \pm .01}$	$99.65 \pm .02$	$99.61 \pm .01$
Margin	$\mathbf{97.97 \pm .12}$	$97.88 \pm .17$	$97.80 \pm .03$	$\mathbf{99.93 \pm .01}$	$\mathbf{99.68 \pm .01}$	$\mathbf{99.64 \pm .02}$
Coreset	$97.79 \pm .06$	$97.81 \pm .19$	$97.77 \pm .07$	$99.48 \pm .02$	$98.94 \pm .03$	$98.69 \pm .03$
GALAXY	$97.94 \pm .20$	$\mathbf{97.98 \pm .12}$	$97.84 \pm .10$	$99.90 \pm .01$	$99.66 \pm .02$	$99.60 \pm .02$
BADGE	$97.95 \pm .08$	$97.84 \pm .10$	$\mathbf{97.87 \pm .06}$	$\mathbf{99.93 \pm .01}$	$99.61 \pm .02$	$99.58 \pm .03$
BAIT	$97.87 \pm .16$	$97.85 \pm .14$	$97.84 \pm .12$	$99.59 \pm .04$	$99.32 \pm .02$	$99.32 \pm .02$
Random	$97.59 \pm .22$	$97.59 \pm .22$	$97.59 \pm .22$	$98.18 \pm .05$	$98.18 \pm .05$	$98.18 \pm .05$
Best	$97.97 \pm .12$	$97.98 \pm .10$	$97.87 \pm .06$	$99.93 \pm .01$	$99.68 \pm .01$	$99.64 \pm .02$

Table 4: Selection-via-proxy results of CIFAR-10 using CLIP ViT-B32. The results are evaluated with 10,000 labels. Confidence intervals are standard errors based on four trials.

	Test Accuracy			Pool Accuracy		
	Fine-tune	Shallow Network	Linear Probe	Fine-tune	Shallow Network	Linear Probe
Confidence	**87.33 ± .26**	**86.37 ± .19**	**86.38 ± .17**	**93.75 ± .08**	**90.86 ± .03**	**90.90 ± .39**
Entropy	86.89 ± .22	86.12 ± .18	86.14 ± .13	93.06 ± .05	90.56 ± .12	90.61 ± .05
Margin	87.30 ± .21	86.40 ± .36	86.68 ± .03	93.61 ± .03	90.76 ± .17	90.72 ± .05
Coreset	85.58 ± .28	85.08 ± .32	85.30 ± .38	89.41 ± .18	87.82 ± .50	87.63 ± .12
GALAXY	87.22 ± .20	86.28 ± .35	86.44 ± .24	93.05 ± .08	90.50 ± .02	90.64 ± .06
BADGE	87.20 ± .38	86.42 ± .23	86.55 ± .18	92.88 ± .15	90.16 ± .04	90.27 ± .19
Random	85.77 ± .20	85.77 ± .20	85.77 ± .20	87.72 ± .09	79.66 ± .03	79.66 ± .03
Best	87.33 ± .26	86.37 ± .19	86.38 ± .17	93.75 ± .08	90.86 ± .03	90.90 ± .39

Table 5: Selection-via-proxy results of CIFAR-100 using CLIP ViT-B32. The results are evaluated with 10,000 labels. Confidence intervals are standard errors based on four trials.

	Test Macro F1			Pool Macro F1		
	Fine-tune	Shallow Network	Linear Probe	Fine-tune	Shallow Network	Linear Probe
Confidence	46.44 ± 2.14	48.89 ± 4.48	49.96 ± 3.87	62.40 ± .60	58.81 ± .94	59.93 ± 3.14
Entropy	50.00 ± .77	48.52 ± 8.04	50.54 ± 3.25	**64.30 ± 2.24**	**65.47 ± 2.04**	62.76 ± 1.85
Margin	50.55 ± 1.08	50.91 ± 1.80	**52.26 ± 2.00**	56.90 ± 3.17	56.52 ± 2.77	61.73 ± 1.63
Coreset	52.08 ± 1.71	**53.13 ± 1.94**	50.13 ± .77	49.33 ± 13.9	44.71 ± 11.6	38.71 ± .33
GALAXY	**52.39 ± 3.48**	49.87 ± 1.84	51.80 ± 3.85	62.41 ± 2.88	59.74 ± 1.54	**62.87 ± 1.35**
BADGE	49.88 ± 1.61	51.85 ± 0.82	50.51 ± 1.83	56.05 ± .52	54.31 ± 3.47	53.86 ± 1.52
Random	49.83 ± 1.26	49.83 ± 1.26	49.83 ± 1.26	38.47 ± .97	38.47 ± .97	38.47 ± .97
Best	52.39 ± 3.48	53.13 ± 1.94	52.26 ± 2.00	64.30 ± 2.24	65.47 ± 2.04	62.87 ± 1.35

Table 6: Selection-via-proxy results of iWildcam using CLIP ViT-B32. The results are evaluated with 21,000 labels. Confidence intervals are standard errors based on four trials.

	Test Accuracy			Pool Accuracy		
	Fine-tune	Shallow Network	Linear Probe	Fine-tune	Shallow Network	Linear Probe
Confidence	58.66 ± .49	57.82 ± .37	58.25 ± .37	72.47 ± .32	70.91 ± .41	71.42 ± .27
Entropy	58.14 ± .75	57.75 ± .35	58.02 ± .29	71.02 ± 1.40	70.87 ± .27	71.02 ± .21
Margin	59.51 ± .37	58.80 ± .06	58.98 ± .30	**74.36 ± .19**	**71.63 ± .19**	**71.62 ± .08**
Coreset	57.71 ± .26	57.35 ± .07	56.75 ± .69	68.43 ± .42	66.50 ± .40	66.07 ± .57
GALAXY	59.41 ± .22	58.91 ± .19	59.10 ± .28	73.56 ± .43	71.32 ± .76	71.42 ± .28
BADGE	**59.59 ± .47**	**59.25 ± .27**	**59.17 ± .28**	73.30 ± .16	70.92 ± .05	71.12 ± .58
Random	58.40 ± .34	58.40 ± .34	58.40 ± .34	68.46 ± .13	68.46 ± .13	68.46 ± .13
Best	59.59 ± .47	59.25 ± .27	59.17 ± .28	74.36 ± .19	71.63 ± .19	71.62 ± .08

Table 7: Selection-via-proxy results of fMoW using CLIP ViT-B32. The results are evaluated with 30,000 labels. Confidence intervals are standard errors based on four trials.