Supplementary Material: Active Domain Adaptation via Clustering Uncertainty-weighted Embeddings

1. Appendix

Contents

1.1. Performance of CLUE on Standard AL

While CLUE is designed as an Active DA strategy, we nevertheless study its suitability for traditional active learning in which models are typically trained from "scratch". We benchmark its performance against competing methods in two settings: Finetuning from an ImageNet [18] initialization on the Clipart→Sketch shift from DomainNet, following the same experimental protocol we use for Active DA (but set softmax temperature T= 1.0 as uncertainty is less reliable when learning from scratch), and the standard SVHN benchmark used for active learning [1]. For AL on SVHN, we match the setting in Ash *et al*. [1], initializing a ResNet18 [7] CNN with random weights and perform 100 rounds of active learning with per-round budget of 100. As summarized in Figure 2a, on DomainNet CLUE significantly outpeforms prior work. On SVHN (Fig 2b), CLUE is on-par with state-of-the-art AL methods, and statistically significantly better than uniform sampling over most rounds.

1.2. Analyzing CLUE: When do gains saturate?

Due to the computational expense of running active domain adaptation on multiple shifts with a large CNN (ResNet34 [7]) on a large dataset (DomainNet [15]), in the main paper we restrict ourselves to 10 rounds with a per-round budget of 500. As a check on when performance gains saturate, we benchmark performance of CLUE + MME [19] on Clipart→Sketch for 40 rounds with per-round budget of 500 ($= 20k$ labels in total), against a few representative

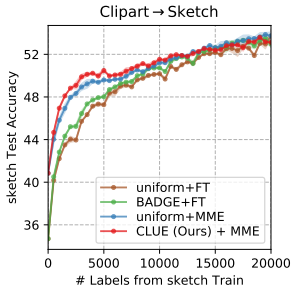


Figure 1: C→S: When do gains with CLUE saturate?

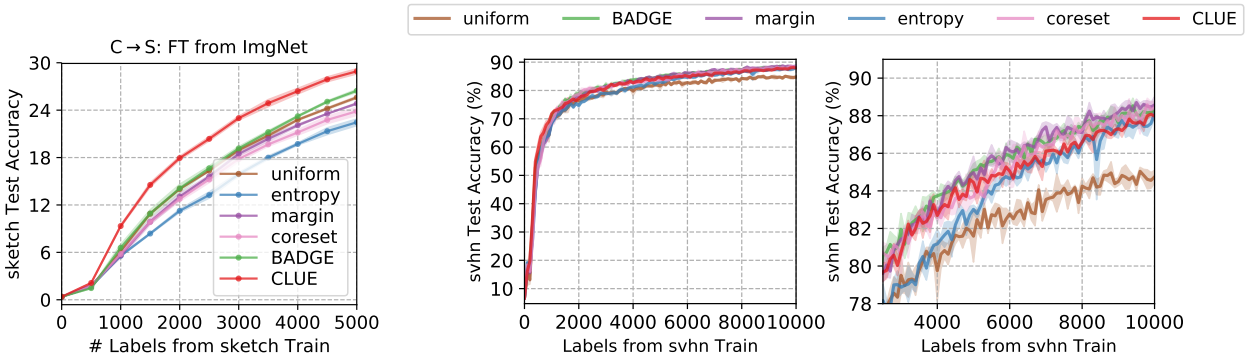
baselines: BADGE [1]+FT (state-of-the-art AL method with finetuning), uniform+MME [19] (uniform sampling with state-of-the-art semi-supervised DA method), and uniform + FT (uniform sampling with finetuning). Results are presented in Fig. 1. As seen, CLUE's strongest performance improvements are seen in the initial stages of training, where it significantly outperforms competing methods. Performance then begins to saturate roughly around the 15k labels mark, and performance differences across methods narrow.

1.3. Comparing CLUE and BADGE

Both CLUE and BADGE are hybrid label acqusition strategies that combine uncertainty and diversity sampling. We now elaborate upon the differences between the two:

Conceptual comparison. CLUE and BADGE *conceptually* differ in 3 key ways: **i) Feature space**. BADGE: Operates in "gradient embedding" space computed as an outer product of penultimate-layer instance embeddings and model output scores. CLUE: Operates on penultimate-layer embeddings scaled by model uncertainty. **ii) Uncertainty measure**. BADGE: Uses gradient wrt model's top-1 prediction. CLUE: Uses predictive entropy. **iii) Diversity measure**. BADGE: Runs KMeans++ in gradient embedding space. CLUE: Runs uncertainty-weighted K-Means on instance embeddings and selects nearest neighbors to centroids.

These design choices influence both BADGE's effectiveness and efficiency. For $D-$dim. penultimate-layer embeddings and $C-$dim. (for C classes) output scores, *each* gradient embedding has CD-dims – on DomainNet, this is $176k$-



(a) C→S: 10 rounds,B = 500

(b) SVHN: 100 rounds, B= 100

Figure 2: Active learning performance of CLUE on DomainNet C→S (finetuning from ImageNet initialization) and SVHN (finetuning from scratch). CLUE significantly outperforms state-of-the art active learning methods in the first case and performs on-par in the second.

dims with a ResNet34 and ∼1.4 million dims with AlexNet. In addition to being expensive to compute, KMeans++ in such high dimensional spaces is less effective as distance measures becomes less reliable. In comparison, CLUE operates in a significantly lower-dimensional feature space (512/4096 for ResNet34/AlexNet). We believe these differences lead to CLUE being more computationally efficient (Tab. 3 in main paper) and effective on average than BADGE (Tab. 1-2 in main paper), especially on hard shifts (*e.g.* S→P, C→Q in Tab. 1 of main paper).

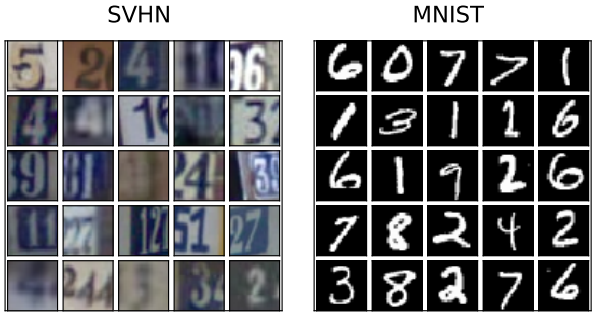
AL method	SVHN → MNIST			DSLR → Amazon		
	30	60	150	30	60	150
BADGE	**89.9**±0.9	93.1±0.2	**96.4**±0.1	**58.2**±1.2	61.6±0.2	71.3±1.4
CLUE	**91.1**±0.5	**93.9**±0.5	**96.2**±0.2	**60.2**±1.2	**65.6**±0.5	**72.7**±0.9

Table 1: Active DA acc. and 1 std dev over 3 runs with MME.

Empirical comparison. On DomainNet, CLUE achieves small but *consistent* gains over BADGE (Tab. 1 in main paper) – with MME, across 4 diverse shifts × 10 rounds = 40 settings, and accounting for error bars, CLUE does as well or better than BADGE on 38/40 (better on 24/40). On other benchmarks, we sometimes observe large gains on other benchmarks (+5.2% on DIGITS at B=30, Tab. 2). However, at later rounds on SVHN→MNIST and DSLR→Amazon, BADGE and CLUE perform similarly. With MME, and after including error bars, CLUE matches or outperforms BADGE on 24/30 (DIGITS) and 10/10 (Office) settings (Tab. 1 shows 3 budgets), with BADGE doing slightly better from rounds 20-24 on DIGITS. We conjecture this is because gradient embedding dimensionality on DIGITS for 10-way classification is low, which leads to BADGE being as effective.

1.4. Dataset details

DomainNet. For our primary experiments, we use the DomainNet [15] dataset that consists of 0.6 million images



(a) (b)

Figure 3: DIGITS qualitative examples

spanning 6 domains, available at http://ai.bu.edu/M3SDA/. For our experiments, we use 4 shifts from 5 domains: Real, Clipart, Sketch, Painting, and Quickdraw. Table 2 summarizes the train/test statistics of each of these domains, while Fig. 4 provides representative examples from each. As models use ImageNet initialization, we avoid using Real as a target domain.

	Real	Clipart	Painting	Sketch	Quickdraw
Train	120906	33525	48212	50416	120750
Test	52041	14604	20916	21850	51750

Table 2: DomainNet [15] train/test statistics

DIGITS. We present results on the SVHN [12]→MNIST [10] domain shift. Both datasets consist of images of the digits 0-9. SVHN consists of 99289 (73257 train, 26032 test) RGB images whereas MNIST contains 70k (60k train, 10k test) grayscale images. Fig. 3 shows representative examples.

1.5. Code and Implementation Details

We use PyTorch [13] for all our experiments. Most experiments were run on an NVIDIA TitanX GPU.

CLUE. We use the weighted K-Means implementation in scikit-learn [14] to implement `CLUE`. Cluster centers are initialized via K-means++ [3]. The implementation uses the Elkan algorithm [6] to solve K-Means. For n objects, k clusters, and e iterations ($= 300$ in our experiments), the time complexity of the Elkan algorithm is roughly $\mathcal{O}(nke)$ [4], while its space complexity is $\mathcal{O}(nk)$.

DomainNet experiments. We use a ResNet34 [7] CNN architecture. For active DA (round 1 and onwards), we use the Adam [9] optimizer with a learning rate of 10^{-5}, weight decay of 10^{-5} and train for 20 epochs per round (with an epoch defined as a complete pass over labeled target data) with a batch size of 64. For unsupervised adaptation (round 0), we use Adam with a learning rate of $3\text{x}10^{-7}$, weight decay of 10^{-5}, and train for 50 epochs. Across all adaptation methods, we tune loss weights to ensure that the average labeled loss is approximately 10 times as large as the average unsupervised loss. We use random cropping and random horizontal flips for data augmentation. We set loss weights for supervised source training $\lambda_{\mathcal{S}} = 0.1$, supervised target training $\lambda_{\mathcal{T}} = 1$, and min-max entropy (for `MME`) $\lambda_{\mathcal{H}} = 0.1$.

Tuning softmax temperature. In Active DA, it is unrealistic to assume access to a large validation set on the target to tune hyperparameters. To tune softmax temperature T for `CLUE` that trades off uncertainty and diversity, we thus create a small heldout validation set of just 1% of target data (482 examples) on the Clipart→Sketch shift, and perform a grid search over temperature values. We select T$= 0.1$ based on its relatively consistent performance (Fig. 5) across rounds on C→S, and use it across shifts on DomainNet. We reiterate that T is an optional hyperparameter that may be tuned for a performance boost if a small validation set is available. As demonstrated in Sec 4.5 of the main paper, `CLUE` achieves SoTA results even with the default value of $T = 1.0$. Further, we also find that T generalizes across shifts, suggesting that in practical scenarios it may be sufficient to tune it only on a single validation shift within a benchmark.

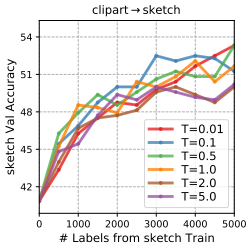


Figure 5: C→S: Tuning softmax temperature with a small target validation set (1% data).

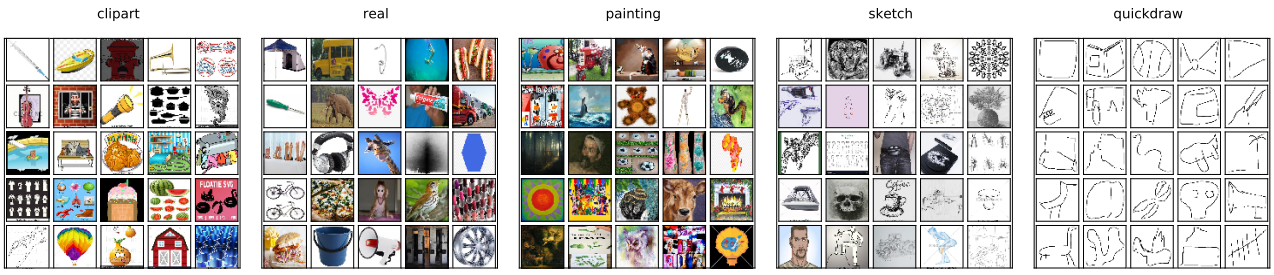
DIGITS experiments. We use the modified LeNet architecture proposed in Hoffman et al. [8] and exactly match the experimental setup in `AADA` [22]. We use the Adam [9] optimizer with a learning rate of $2\text{x}10^{-4}$, weight decay of 10^{-5}, batch size of 128, and perform 60 epochs of training per-round. We halve the learning rate every 20 epochs. We set loss weights for supervised source training $\lambda_{\mathcal{S}} = 0.1$, for supervised target training $\lambda_{\mathcal{T}} = 1$, and min-max entropy (for `MME`) $\lambda_{\mathcal{H}} = 1$.

1.5.1 Baseline Implementations

We elaborate on our implementation of the `BADGE` [1] and `AADA` [22] baselines.

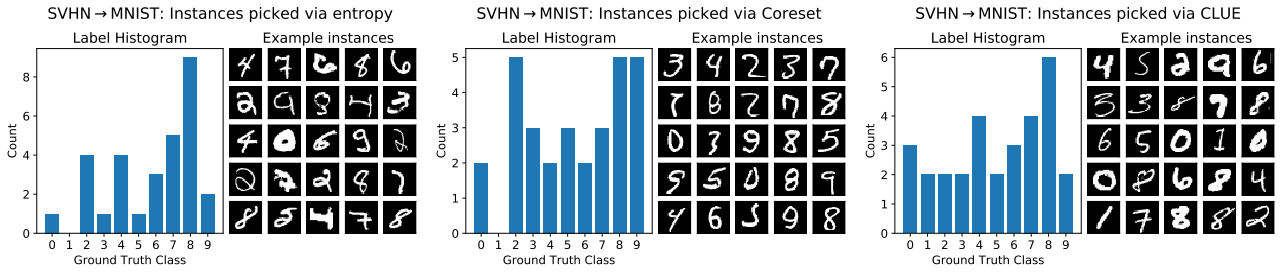
`BADGE`. BADGE "gradient embeddings" are computed by taking the gradient of model loss with respect to classifier weights, where the loss is computed as cross-entropy between the model's predictive distribution and its most confidently predicted class. Next, K-Means++ [3] is run on these embeddings to yield a batch of samples.

`AADA`. In `AADA`, a domain discriminator G_d is learned to distinguish between source and target features obtained from an extractor G_f, in addition to a task classifier G_y. For active sampling, points are scored via the following importance weighting-based acquisition function ($\mathcal{H}$ denotes model entropy): $s(x) = \frac{1-G_d(G_f(x))}{G_d^*(G_f(x))}\mathcal{H}\left(G_y\left(G_f(x)\right)\right)$, and top B instances are selected for labeling. In practice, to generate less redundant batches we randomly sample B instances from the top-2% scores, as recommended by the authors. Consistent with the original work, we also add an entropy minimization objective with a loss weight of 0.01.



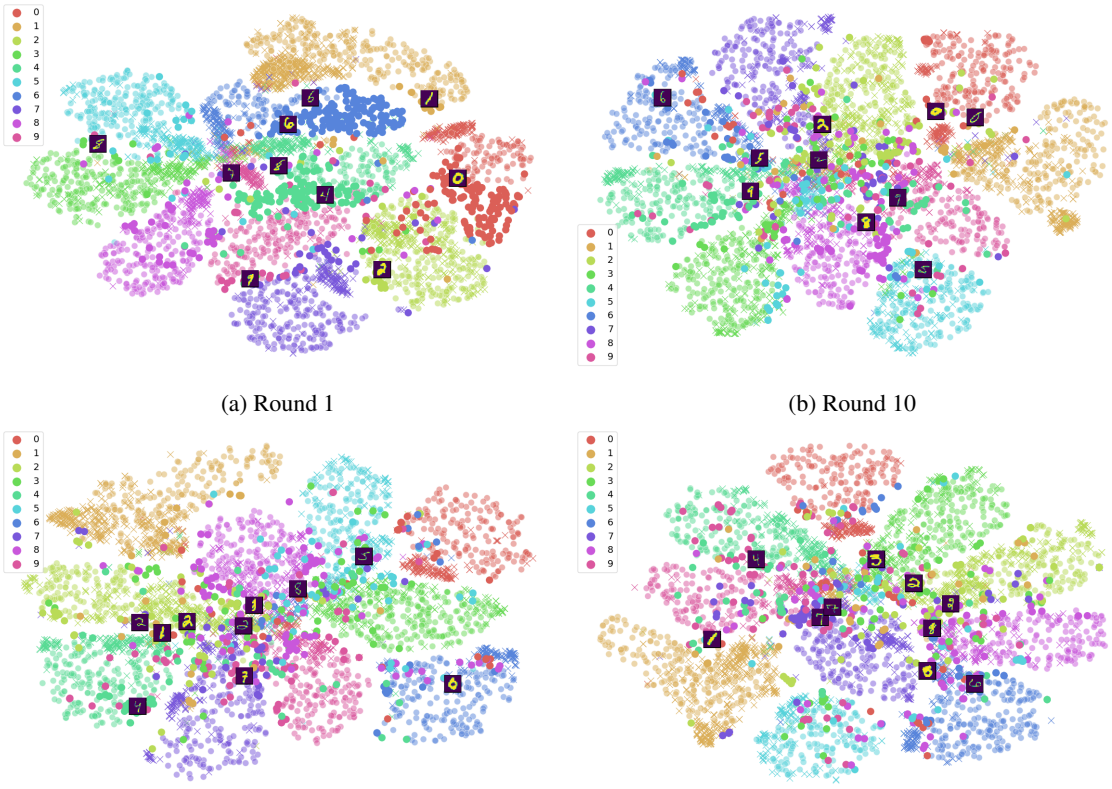
(a) (b) (c) (d) (e)

Figure 4: DomainNet [15] qualitative examples



(a) (b) (c)

Figure 6: SVHN→MNIST: Label histograms and examples of instances selected by entropy, coreset, and CLUE at Round 1 with $B = 30$.



(a) Round 1 (b) Round 10

(c) Round 20 (d) Round 30

Figure 7: SVHN→MNIST: TSNE visualization of feature space and instances picked by CLUE at rounds 1, 10, 20, and 30. Circles denote target points and crosses denote source points.

1.6. CLUE: Qualitative Analysis

In this section, we attempt to get a sense of the behavior of CLUE versus other methods via visualizations and qualitative examples on the SVHN→MNIST shift. Fig. 8 shows confusion matrices of model predictions before (*left*) and after (*right*) performing unsupervised adaptation (via MME) at round 0. As seen, MME aligns some classes (eg. 1's and 9's) remarkably well even without access to target labels. However, large misalignments remain for some other classes (0, 4, and 6).

Visualizing selected points. In Fig. 6, we visualize instances selected by three strategies at Round 0 – entropy [23], coreset [20], and CLUE, with $B = 30$. We visualize the ground truth label distribution of the selected instances, as well as qualitative examples. As seen, strategies vary across

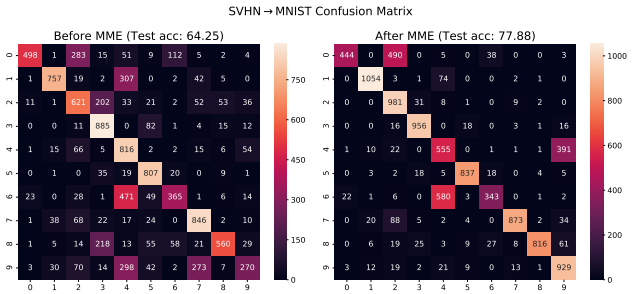


Figure 8: SVHN→MNIST: Confusion matrix of model predictions before and after MME at round 0.

methods. "Entropy" tends to pick a large number of 8's, and selects high-entropy examples that (on average) appear challenging even to humans. "Coreset" tends to have a wider spread over classes. CLUE appears to interpolate between the behavior of these two methods, selecting a large number of 8's (like entropy) but also managing to sample atleast a few instances from every class (like coreset).

t-SNE visualization over rounds. In Fig. 7, we illustrate the sampling behavior of CLUE over rounds via t-SNE [11] visualizations. We follow the same conventions as Fig.3 of the main paper, and visualize the logits of a subset of incorrect (large, opaque circles) and correct (partly transparent circles) model predictions on the target domain, along with instances sampled via CLUE. We oversample incorrect target predictions to emphasize regions of the feature space on which the model currently underperforms. Across all four stages, we find that CLUE samples instances that are uncertain (often present in a cluster of incorrectly classified instances) and diverse in feature space. This behavior is seen even at later rounds when classes appear better separated.

1.7. Extended Description of the CLUE Objective

We describe in more detail the CLUE objective presented (Eq. 4) in the main paper. Recall that we seek to identify target instances that are diverse in model feature space. Considering the L_2 distance in the CNN representation space $\phi(\cdot)$ as a dissimilarity measure, we quantify the dissimilarity between instances in a set X_k in terms of its variance $\sigma^2(X_k)$ given by [24]:

$$\begin{aligned}\sigma^2(X_k) &= \frac{1}{2|X_k|^2}\sum_{\mathbf{x_i},\mathbf{x_j}\in X_k}||\phi(\mathbf{x_i})-\phi(\mathbf{x_j})||^2 \\ &= \frac{1}{|X_k|}\sum_{\mathbf{x}\in X_k}||\phi(\mathbf{x})-\mu_\mathbf{k}||^2 \\ \text{where}\quad \mu_\mathbf{k} &= \frac{1}{|X_k|}\sum_{\mathbf{x}\in X_k}\phi(\mathbf{x})\end{aligned} \tag{1}$$

A small $\sigma^2(X_k)$ indicates that a set X_k contains instances that are similar to one other. Our goal is to identify sets of instances that are representative of the unlabeled target set, by partitioning the unlabeled target data into K sets, each with small $\sigma^2(X_k)$. Formulating this as a set-partitioning problem with partition function $\mathcal{S}: X_T \rightarrow \{X_1, X_2, ..., X_K\}$, we seek to find the $\mathcal{S}$ that minimizes the sum of variance over all sets:

$$\underset{\mathcal{S}}{\text{argmin}}\sum_{k=1}^{N}\sigma^2(X_k) \tag{2}$$

where $\sigma^2(X_k)$ is defined in Eq. 1.

To ensure that the more informative/uncertain instances play a larger role in identifying representative instances, we employ weighted-variance, where an instance is weighted by its informativeness. Let h_i denote the scalar weight corresponding to the instance $\mathbf{x_i}$. The weighted variance [16] $\sigma^2_\mathcal{H}(X_k)$ of a set of instances is given by:

$$\begin{aligned}\sigma^2_\mathcal{H}(X_k) &= \frac{1}{\sum_{\mathbf{x_i}\in X_k}h_i}\sum_{\mathbf{x_i}\in X_k}h_i||\phi(\mathbf{x_i})-\mu_\mathbf{k}||^2 \\ \text{where}\quad \mu_\mathbf{k} &= \frac{1}{\sum h_i}\sum_{\mathbf{x_i}\in X_k}h_i\phi(\mathbf{x_i})\end{aligned} \tag{3}$$

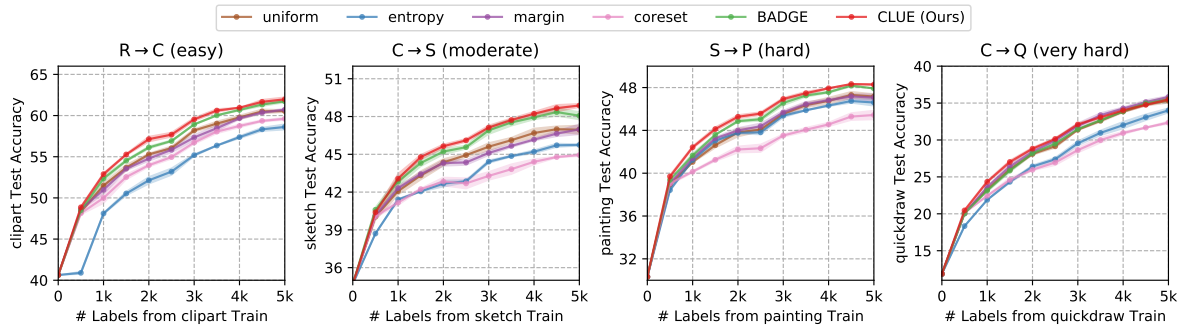
Considering the informativeness (weight) of an instance to be its uncertainty under the model, given by $\mathcal{H}(Y|\mathbf{x})$ (defined in Eq. 1 in main paper), we rewrite the set-partitioning objective in Eq. 2 to minimize sum of weighted variance of a set (from Eq. 3):

$$\begin{aligned}&\underset{\mathcal{S}}{\text{argmin}}\sum_{k=1}^{K}\sigma^2_\mathcal{H}(X_k) \\ =&\underset{\mathcal{S}}{\text{argmin}}\sum_{k=1}^{K}\frac{1}{Z_k}\sum_{\mathbf{x}\in X_k}\mathcal{H}(Y|\mathbf{x})||\phi(\mathbf{x})-\mu_\mathbf{k}||^2 \\ &\text{where }\mu_\mathbf{k} = \frac{1}{\sum_{x\in X_k}\mathcal{H}(Y|\mathbf{x})}\sum_{\mathbf{x}\in X_k}\mathcal{H}(Y|\mathbf{x})\phi(\mathbf{x}) \\ &\text{and}\quad Z_k = \sum_{x\in X_k}\mathcal{H}(Y|\mathbf{x})\end{aligned} \tag{4}$$

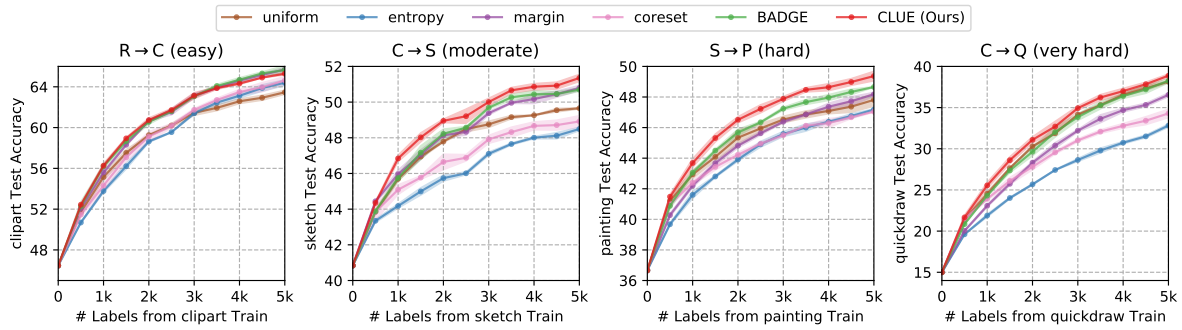
This, gives us the overall set-partitioning objective for CLUE.

1.8. Full performance plots

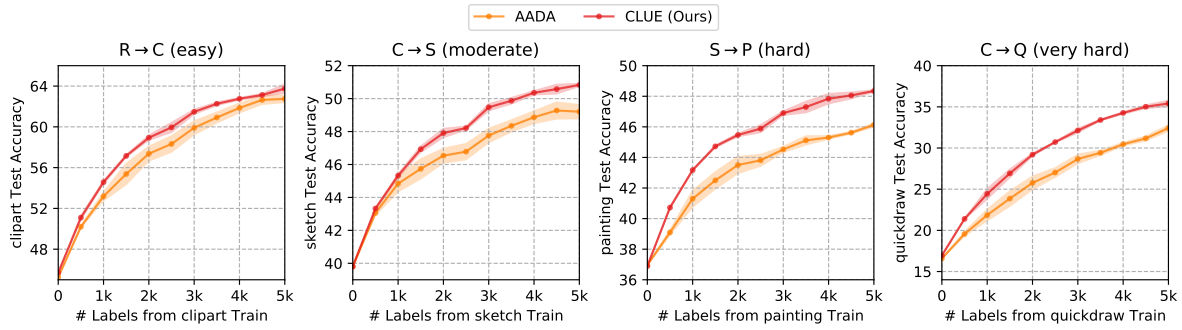
For ease of comparison, we presented performance at 3 intermediate sampling budgets in Tables 1 and 2 in the main paper. In Tables 9, 11, 10, we include the full corresponding performance plots for completeness. Results are presented across 3 learning strategies: finetuning (FT) a source model, semi-supervised DA via MME starting from a source model, and semi-supervised DA via DANN starting from a source model. As is common in active learning, we present results as learning curves, and report performance means and 1 standard deviation over 3 experimental runs via shading.



(a) Finetuning (`ft`) a source model on target labels.



(b) Semi-supervised DA via `MME` [19] (state-of-the-art semi-supervised DA method), starting from a source model.



(c) Semi-supervised DA via `DANN` [5], starting from a source model.

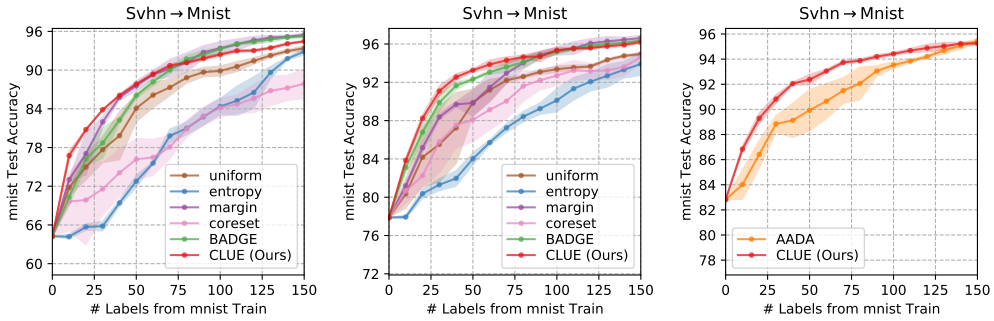
Figure 9: Full plots for Active DA results on DomainNet, corresponding to Table 1 in the main paper. We plot accuracies on target test set for 4 DomainNet shifts of increasing difficulty spanning 5 domains: Real (R), Clipart (C), Sketch (S), Painting (P) and Quickdraw (Q). We perform 10 rounds of Active DA with $B = 500$. We compare `CLUE` against state-of-the art methods for AL (`entropy` [23], `margin` [17], `coreset` [20], `BADGE` [1]) and Active DA (`AADA`), spanning different AL paradigms: uncertainty sampling (U), diversity sampling (D), and hybrid (H) combinations of the two. We use multiple learning strategies: **(a)** finetuning (`ft`) from source, **(b)** `MME` [19] (state-of-the-art semi-supervised DA method) from source, and **(c)** semi-supervised DA via `DANN` [5] from source. Best viewed in color. We report accuracy mean and 1 standard deviation (via shading) over 3 runs.

1.9. Future Work

Our work suggests a few promising directions of future work. First, one could experiment with alternative uncertainty measures in `CLUE` instead of model entropy, including those (such as uncertainty from deep ensembles) that have been shown to be more reliable under a dataset shift [21]. Further, one could incorporate specialized model architectures from few-shot learning [2, 19] to deal with the label sparsity in the target domain. Finally, while we restrict our task to image classification in this paper, it is important to also study active domain adaptation in the context of related tasks such as object detection and semantic segmentation.

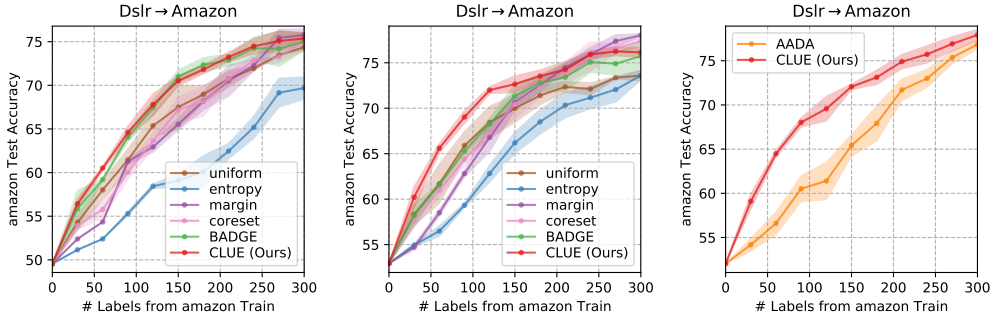
References

[1] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning



(a) Finetuning (ft) (b) MME [19] (c) DANN [5]

Figure 10: Full plots for Active DA results on SVHN→MNIST (DIGITS) with $B = 10$, corresponding to Table 2 (middle), in the main paper. We use multiple learning strategies: **(a)** finetuning (ft) from source, **(b)** MME [19] (state-of-the-art semi-supervised DA method) from source, and **(c)** semi-supervised DA via DANN [5] from source. Best viewed in color. We report accuracy mean and 1 standard deviation (via shading) over 3 runs.



(a) Finetuning (ft) (b) MME [19] (c) DANN [5]

Figure 11: Full plots for Active DA results on DSLR→Amazon (Office) with $B = 30$, corresponding to Table 2 (right), in the main paper. We use multiple learning strategies: **(a)** finetuning (ft) from source, **(b)** MME [19] (state-of-the-art semi-supervised DA method) from source, and **(c)** semi-supervised DA via DANN [5] from source. Best viewed in color. At each round, we report accuracy mean and 1 standard deviation (via shading) over 3 runs.

by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[3] A David. Vassilvitskii s.: K-means++: The advantages of careful seeding. In *18th annual ACM-SIAM symposium on Discrete algorithms (SODA), New Orleans, Louisiana*, pages 1027–1035, 2007.

[4] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 147–153, 2003.

[5] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[6] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 600–607, 2002.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning*, pages 1989–1998, 2018.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[12] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images

with unsupervised feature learning. In *Neural Information Processing Systems (NeurIPS)*, 2011.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[14] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[15] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[16] George R Price. Extension of covariance selection mathematics. *Annals of human genetics*, 35(4):485–490, 1972.

[17] Dan Roth and Kevin Small. Margin-based active learning for structured output spaces. In *European Conference on Machine Learning*, pages 413–424. Springer, 2006.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[19] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8050–8058, 2019.

[20] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[21] Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019.

[22] Jong-Chyi Su, Yi-Hsuan Tsai, Kihyuk Sohn, Buyu Liu, Subhransu Maji, and Manmohan Chandraker. Active adversarial domain adaptation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 739–748, 2020.

[23] Dan Wang and Yi Shang. A new active labeling method for deep learning. In *2014 International joint conference on neural networks (IJCNN)*, pages 112–119. IEEE, 2014.

[24] Yuli Zhang, Huaiyu Wu, and Lei Cheng. Some new deformation formulas about variance and covariance. In *2012 Proceedings of International Conference on Modelling, Identification and Control*, pages 987–992. IEEE, 2012.