Streaming Active Learning with Deep Neural Networks

Akanksha Saran [1] **Safoora Yousefi** [2] **Akshay Krishnamurthy** [1] **John Langford** [1] **Jordan T. Ash** [1]

Abstract

Active learning is perhaps most naturally posed as an online learning problem. However, prior active learning approaches with deep neural networks assume offline access to the entire dataset ahead of time. This paper proposes VeSSAL, a new algorithm for batch active learning with deep neural networks in streaming settings, which samples groups of points to query for labels at the moment they are encountered. Our approach trades off between uncertainty and diversity of queried samples to match a desired query rate without requiring any hand-tuned hyperparameters. Altogether, we expand the applicability of deep neural networks to realistic active learning scenarios, such as applications relevant to HCI and large, fractured datasets.

1. Introduction

Active learning considers a supervised learning situation where unlabeled data are abundant, but acquiring labels is expensive (Settles, 2010; Dasgupta, 2011). One example of this might be classifying underlying disorders from histological images, where obtaining labels involves querying medical experts. Another might be predicting drug efficacy, where labels corresponding to candidate molecules could require clinical trials or intensive computational experiments. In these settings, we typically want to carefully consider what samples to request labels for, and to obtain labels for data that are maximally useful for progressing the performance of the model.

Active learning is a classic problem in machine learning, with traditional approaches typically considering the convex and well-specified regime (Settles, 2010; Dasgupta, 2011; Hanneke, 2014a). Much recent interest in active learning has turned to the neural network case, which requires some special considerations. One such consideration is the expense associated with fitting these neural architectures — when used in conjunction with a sequentially growing training set, as one has in active learning, the model cannot be initialized from the previous round of optimization without damaging generalization performance. Instead, practitioners typically re-initialize model parameters each time new data are acquired and train the model from scratch (Ash & Adams, 2020). This structure has repositioned active learning to focus on the batch domain, where we are interested in simultaneously labeling a batch of k samples to be integrated into the training set. The model is typically retrained only after the entire batch has been labeled.

In the convex case, where a model can easily be updated to accommodate for a single sample, active learning algorithms have tended to focus on uncertainty or sensitivity. That is, a label for a given sample should be requested if the model is highly uncertain about its corresponding label, or if incorporating this sample into the training set will greatly reduce the set of plausible model weights. In contrast, a high-performing, batch-mode active learning algorithm must also consider diversity. If two samples are relatively similar to each other, it is inefficient to include them both in the batch, regardless of the model's uncertainty about their labels; having only one such sample labeled and integrated into the current hypothesis may be enough to resolve the model's uncertainty on the other.

Popular approaches for batch active learning rely on samplers that require all unlabeled data to be simultaneously available. This reliance poses several major concerns for the deployment of these algorithms. For one, the run time of these methods is conditioned on the number of unlabeled samples in a way that makes them unusable for extremely large datasets. To exacerbate the issue, it is unclear how to deploy these algorithms on modern databases, where samples might be stored in a fractured manner and cannot easily be made available in their entirety.

It is especially unclear how to perform active learning in a streaming setting, where data are not all simultaneously available, and we do not know how many samples will be encountered. Here we might instead prefer to specify an acceptable labeling rate rather than a fixed acceptable batch size. In this streaming setup, it is further desirable to commit to a decision about whether to include an unlabeled

[1]Microsoft Research NYC [2]Microsoft Bing. Correspondence to: Akanksha Saran <akankshasaran@utexas.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

sample in the batch as soon as it is encountered, rather than only after the stream has terminated. As a concrete example, consider an HCI application where a user interacts with the world while wearing an assistive or diagnostic device (Bohus et al., 2022; Singh et al., 2016). The software on the device might involve a classifier that detects objects being interacted with or the activity being performed by the user. Requesting a label to update the model to better classify these phenomena can only be done in the moment; it would be cumbersome to ask the user to provide a label corresponding to an event that occurred far in the past. How can we efficiently identify samples from a data stream for neural networks while respecting a maximum query rate?

We propose a simple active learning algorithm, Volume Sampling for Streaming Active Learning (VeSSAL)[1], that addresses the concerns mentioned above. VeSSAL is made to accommodate the streaming setting, and as such it only needs to see each unlabeled point once in order to arrive at a decision about whether it should be labeled. This makes VeSSAL attractive even for fixed datasets that might be extremely large or fractured, as these are often interacted with using streaming, distributed database frameworks. VeSSAL is a natural choice for "committal" situations, when labeling decisions need to be made on the fly. On non-sequential datasets, where more conventional active learning algorithms could be exercised, VeSSAL can be significantly faster, especially for large batch sizes.

Despite its simplicity and flexibility, VeSSAL is surprisingly high performing. We show that VeSSAL produces models with predictive capabilities on par with state-of-the-art approaches, even though they are not restricted to this streaming, committal setting. We further demonstrate this to be the case in adversarial situations, where VeSSAL is presented with data that have been sorted to induce domain drift. VeSSAL is hyperparameter free, making it a powerful candidate for a wide range of active learning scenarios.

The paper proceeds as follows. We overview related work in Section 2. In Section 3, we present the mathematical formulation for the streaming active learning setting, along with details of our proposed algorithm. We give empirical support for our proposed approach in Section 4 via experiments on three benchmark datasets and one real-world dataset, in conjunction with two neural network architectures and three different batch sizes. We conclude with a discussion in Section 5.

2. Related Work

This section situates VeSSAL with respect to prior work on streaming active learning (Sec. 2.1) as well as batch active learning strategies for training neural networks (Sec. 2.2).

[1]Code for the implementation of VeSSAL can be found at `https://github.com/asaran/VeSSAL.git`

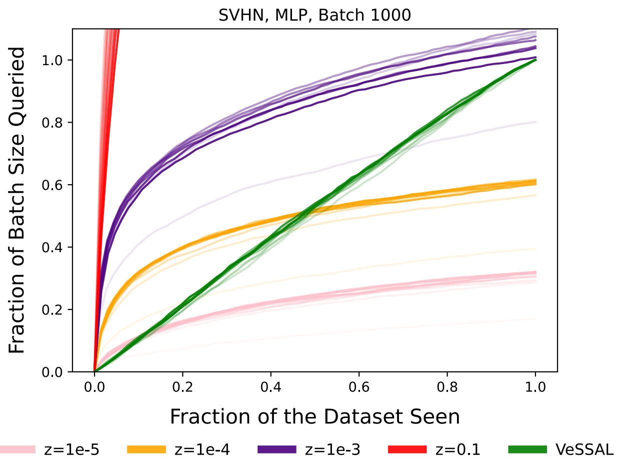


Figure 1. A comparison in terms of sampling rate for our proposed tuning approach and choosing fixed z_t values, used to scale the probability mass on a candidate point as $p_t = z_t \cdot g(x_t)^\top \Sigma^{-1} g(x_t)$. We plot the fraction of the batch size that has been selected as a function of the amount of data in the stream that has been encountered. Fixed scaling values can drastically undersample or oversample, and distribute the labeling budget inequitably across the stream. The active learning round is denoted by line opacity, with darker colors corresponding to higher round numbers — here we show the first ten rounds for each strategy.

2.1. Streaming Active Learning

Active learning has enjoyed many successes for problems in the convex learning setting (Hanneke, 2014b; Beygelzimer et al., 2009; 2010; Roth & Small, 2006; Beygelzimer et al., 2010; Huang et al., 2015; Hsu, 2010; Hanneke & Yang, 2015). Beygelzimer et al. (2009) propose a statistically consistent method using importance weighting for actively learning binary classifiers under general loss functions. Krishnamurthy et al. (2017) present a version-space based active learning method with performance guarantees for cost-sensitive multiclass classification. Similar to these methods, there is a long line of theoretical work on active learning for linear models (Hanneke, 2014b). While these approaches are indeed designed for the streaming setting, they rely on updating the linear hypothesis after each sample is labeled, precluding them from being used in conjunction with deep neural networks, where updating the model is known to be extremely expensive. Furthermore, Sun et al. (2022) consider the problem of filling a replay buffer for use in continual learning. Unlike in active learning, the model has access to a ground-truth label for each sample it encounters.

Successful streaming-based techniques with theoretical guarantees have been developed for the related setting of adaptive sampling and low-rank matrix approximation (Frieze et al., 2004; Deshpande & Vempala, 2006;

Deshpande et al., 2006; Ghashami & Phillips, 2014; Bhaskara et al., 2019). These methods find utility in several problem domains such as online PCA (Boutsidis et al., 2014; Bhaskara et al., 2019), online column subset selection (Bhaskara et al., 2019), and online k-means clustering (Braverman et al., 2011). We take inspiration from this line of prior theoretical work to design a streaming algorithm for neural batch active learning.

2.2. Batch Active Learning for Deep Neural Networks

In recent years, several advancements have been made in the area of pool-based active learning for deep neural networks (Ren et al., 2021). Prior approaches have either employed diversity-based sampling (Sener & Savarese, 2018; Geifman & El-Yaniv, 2017; Gissin & Shalev-Shwartz, 2019), uncertainty-based sampling (Gal et al., 2017; Ducoffe & Precioso, 2018; Beluch et al., 2018) or both (Ash et al., 2020; 2021). Ash et al. (2020) propose a pool-based deep active learning method which leverages gradient embeddings to capture both diversity and uncertainty by pairing a gradient-based representation with an approximate k-DPP sampling technique. Gudovskiy et al. (2020) give an active learning approach to tackle the distribution shift between the train and test data via a feature density matching method using Fisher kernels. Still, these approaches are not designed to handle the streaming setting. Instead, these samplers require access to the entire candidate pool in order to identify each valuable point to include in the batch of unlabeled samples.

More recently, Ban et al. (2022) proposed a stream-based deep active learning approach, albeit not a batch active learning algorithm. Instead, they query only a single point at a time, and add it to their replay buffer after its corresponding label has been obtained. Because this work is outside of the batch setting, there are no diversity considerations to the label acquisition rule. Instead, decisions are only based on predictive uncertainty, as measured by the difference in probability mass between the most likely and second most likely predicted label. Lavania et al. (2021) propose a streaming submodular maximization-based active learning approach, but the queies are sampled in a non-committal fashion (unlike VeSSAL).

Several works have designed active learning approaches specifically for image classification (Kovashka et al., 2016) and object detection (Choi et al., 2021; Brust et al., 2018; Senzaki & Hamelain, 2021). Sun & Gong (2019) leverage deep reinforcement learning to train a data selection policy for training neural networks to perform image classification. Roy et al. (2018) use an uncertainty based sampling approach for active learning via the paradigm of "query by committee", leveraging the disagreement between convolutional layers for Single Shot Multibox Detector architectures (Liu et al., 2016). Still, these approaches assume access to the entire dataset for decision making, and cannot be used in the streaming setting.

Algorithm 1 Volume sampling for streaming active learning (VeSSAL)

```
Require: Neural network $f(x;\theta)$, unlabeled stream of samples $U$, ideal sampling rate $q$
 1: Initialize $t = 1$
 2: Initialize $\hat{\Sigma}_0^{-1} = \lambda^{-1} I_d$ {regularized by $\lambda$ for stability}
 3: Initialize $A_0 = 0_{d,d}$ {covariance over all data}
 4: Initialize $B = \emptyset$ {set of chosen samples}
 5: for $x_t \in U$: do
 6:     $A_t \leftarrow \frac{t-1}{t} A_{t-1} + \frac{1}{t} g(x_t) g(x_t)^\top$
 7:     $p_t = q \cdot g(x_t)^\top \hat{\Sigma}_t^{-1} g(x_t)\, \mathrm{tr}(\hat{\Sigma}_t^{-1} A_t)^{-1}$
 8:     with probability $\min(p_t, 1)$:
 9:         Query label $y_t$ for sample $x_t$
10:         $B \leftarrow B \cup (x_t, y_t)$
11:         $\hat{\Sigma}_{t+1}^{-1} \leftarrow \hat{\Sigma}_t^{-1} - \frac{\hat{\Sigma}_t^{-1} g(x_t) g(x_t)^\top \hat{\Sigma}_t^{-1}}{1+g(x_t)^\top \hat{\Sigma}_t^{-1} g(x_t)}$ {rank-1 Woodbury update}
12:     else:
13:         $\hat{\Sigma}_{t+1}^{-1} \leftarrow \hat{\Sigma}_t^{-1}$
14:     $t \leftarrow t + 1$
15:     return labeled batch $B$ for retraining $f$
16: end for
```

Brust et al. (2018) use uncertainty-based margin sampling for streaming active learning with object detectors. They present various methods to aggregate uncertainty estimates for all objects in an image to determine their selection strategy. While their approach is designed for continual object learning settings, it is limited to the problem of object detection and cannot be applied to classification tasks.

3. Volume Sampling for Streaming Active Learning (VeSSAL)

Neural active learning algorithms that incorporate diversity can largely be thought of as making two design decisions. The first decision is how unlabeled candidate samples should be represented. Common choices include using the penultimate layer representation of the current state of the network (Sener & Savarese, 2018) or using a hypothetical gradient that might be induced by a given sample (Ash et al., 2020). In either case, once data are in this space, the second decision is regarding how unlabeled points should be selected in order to encourage batch diversity.

In VeSSAL, these decisions are made to produce a high-performing active learner that is amenable to the streaming, committal setting. Specifically, we assume that each candidate x_t is seen only once, and that we must make a decision about whether or not to include it in the batch B as soon as it is encountered. Once the labeling budget k is allocated, we retrain the model and repeat the process.

VeSSAL performs approximate volume sampling over unlabeled candidate points in a gradient space computed with respect to the last layer of the neural network. For a neural network f with parameters θ, last-layer parameters $\theta_L \in \theta$ and cross-entropy loss function ℓ, the gradient representation for a sample x_t is

$$g(x_t) = \frac{\partial}{\partial \theta_L} \ell(f(x_t;\theta), \hat{y}_t), \tag{1}$$

where $\hat{y}$ denotes the most likely label according to the current state of the model, i.e. $\hat{y}_t = \arg\max f(x_t;\theta)$.

A typical way of doing volume sampling is to select a batch of points with probability proportional to the determinant of their gram or covariance matrix (Kulesza et al., 2012). In the latter case, this would mean the probability mass on a batch of samples B is proportional to

$$\det\Big(\sum_{x\in B} g(x)g(x)^\top\Big), \tag{2}$$

where $|B| = k$, the pre-specified labeling budget.

There are several reasons to favor the covariance matrix version over the gram matrix. From a theoretical point of view, when used in an outer product, $g(x)$ could be thought of as a rank-1 approximation of the Fisher information matrix, $I(x;\theta) := \mathbb{E}_{y\sim p_\theta(\cdot|x,\theta)} \nabla^2 \ell(x, y; \theta)$ (Ash et al., 2021). As such, this construction is reminiscent of a classic goal in active learning, which is to maximize the determinant for the Fisher (MacKay, 1992). This objective is attractive because, in the realizable setting, it selects samples that maximize the information gained by model parameters after labeling.

From a more practical point of view, the covariance matrix will stay fixed in dimensionality even as the batch size k is changed. In the gram matrix alternative, which is suggested in Ash et al. (2020), the size of the matrix grows with k, potentially becoming intractable for larger batch sizes.

A process that samples from a distribution characterized by a determinant like this is referred to as a determinantal point process (DPP). Sampling from a DPP is usually done via Markov Chain Monte Carlo, and making this procedure efficient is an active area of research with wide-ranging statistical applications (Bardenet et al., 2017). Still, the mixing times associated with these algorithms generally makes them too inefficient to be used in conjunction with modern active learning algorithms. For example, BADGE (Ash et al., 2020) suggests using kmeans++ as a surrogate for DPP sampling, and Coreset (Sener & Savarese, 2018) uses a furthest-first traversal approach (though not in gradient space).

These sampling approaches are workable surrogates for true volume sampling, but they require all data to be simultaneously accessible. Our algorithm demonstrates that this is not necessary and that near state-of-the-art performance can be obtained by a sampler that (1) sees samples in a streaming fashion, such that each point is only observed once and that (2) commits to a labeling decision as soon as a sample is encountered.

VeSSAL choses a sample x_t for labeling with probability p_t proportional to the determinantal contribution of its gradient when considering other items that have already been chosen,

$$p_t \propto \det\Big(\hat{\Sigma}_t + g(x_t)g(x_t)^\top\Big)$$

Using the matrix determinant lemma (Greub, 2012; Strang, 2006), the expression for p_t reduces to

$$\begin{aligned} p_t &\propto \det(\hat{\Sigma}_t)(1 + g(x_t)^\top \hat{\Sigma}_t^{-1} g(x_t)) \\ &\propto g(x_t)^\top \hat{\Sigma}_t^{-1} g(x_t). \end{aligned}$$

Here $\hat{\Sigma}_t$ is the covariance over samples that have already been selected for inclusion in the batch, $\sum_{x\in B} g(x)g(x)^\top$.

To compute a p_t in practice, $g(x_t)^\top \hat{\Sigma}_t^{-1} g(x_t)$ must be scaled by some value z_t, which reflects the labeling budget available to the algorithm. Because the amount of data in the stream might not be known, we consider tuning z_t to reflect a desired labeling frequency q. In HCI applications (Bohus et al., 2022; Singh et al., 2016; Wang et al., 2021), for example, we might not know how long a user will interact with a device (the total number of candidate samples), but we might instead have some sense of an acceptable frequency with which labels can be queried. If we instead do know the total number of samples, then we could consider q to be the ratio between the labeling budget and the size of the candidate set. Specifically, we desire to find some scalar z_t such that

$$\mathbb{E}_x[p_t] = \mathbb{E}_x\Big[z_t \cdot g(x)^\top \hat{\Sigma}_t^{-1} g(x)\Big] = q. \tag{3}$$

How should this z_t be chosen? Because the statistics of gradient representations $g(x)$ vary with the state of f, one fixed value is unlikely to work well across model architectures, datasets, batch sizes, and rounds of data selection. Instead, we aim to find an adaptive strategy, such that we both obtain the desired sampling frequency q and that we do so in a way that does not allocate probability mass disproportionately across temporal regions of the stream.

One option for adaptively adjusting z_t as selection progresses might be a multiplicative weights approach, where we select a scaling parameter from a distribution over a number of z_t values, and constantly update this distribution to reflect whatever choice is giving us the best rate. Another

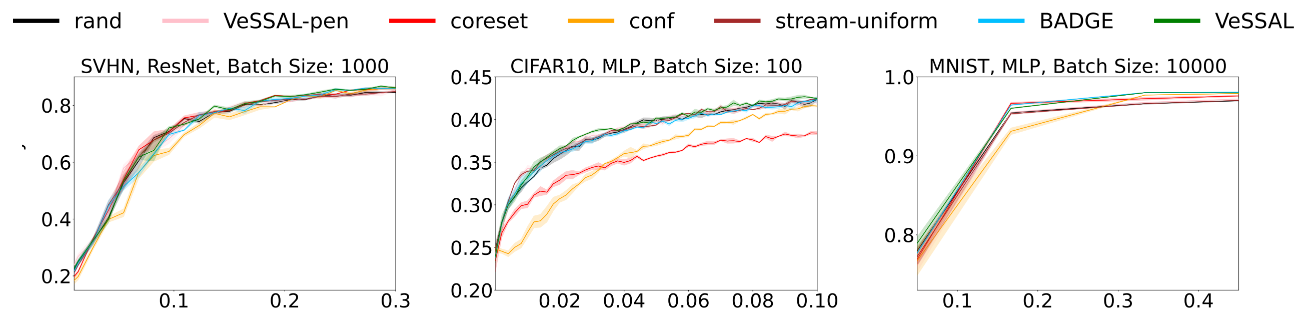


Figure 2. Learning curves for different neural active learning methods tested with i.i.d data streams. Two network architectures, three batch sizes, and three datasets are shown here. These plots have been zoomed to highlight discriminative regions, but complete results are shown in the Appendix and are aggregated in Figure 3 (a).

option is to use gradient descent, making z_t larger or smaller at each step in the service of minimizing error between the current and target sampling rate q. Unfortunately, these approaches are unlikely to work well, because the underlying distribution given by the determinantal contribution changes with each selected point. Specifically, every time a sample is added to the batch, and $\hat{\Sigma}_t$ is updated, the distribution of responses $g(x_t)^\top \hat{\Sigma}_t^{-1} g(x_t)$ can change drastically. This domain shift precludes adaptive solutions from efficiently finding a suitable z_t like those mentioned above, because they assume the underlying distribution is stationary.

To circumvent this, we simply rewrite the expectation to disentangle $\hat{\Sigma}_t^{-1}$ from statistics relating to $g(x)$:

$$\begin{aligned}\mathbb{E}_x\left[z_t \cdot g(x)^\top \hat{\Sigma}_t^{-1} g(x)\right] &= z_t \cdot \mathbb{E}_x\left[\operatorname{tr}\left(g(x)^\top \hat{\Sigma}_t^{-1} g(x)\right)\right] \\ &= z_t \cdot \mathbb{E}_x\left[\operatorname{tr}\left(\hat{\Sigma}_t^{-1} g(x) g(x)^\top\right)\right] \\ &= z_t \cdot \operatorname{tr}\left(\hat{\Sigma}_t^{-1} \mathbb{E}_x\left[g(x) g(x)^\top\right]\right).\end{aligned}$$

Here, our ability to find a suitable z_t relies only on our ability to estimate the covariance of $g(x)$, and is not affected by the frequently changing $\hat{\Sigma}_t^{-1}$. If we approximate $\mathbb{E}_x\left[g(x)g(x)^\top\right]$ as $\frac{1}{t}\sum_{i=1}^t g(x_i)g(x_i)^\top$, this immediately suggests a way to compute p_t that is amenable to the streaming setting:

$$p_t = \frac{q \cdot g(x_t)^\top \hat{\Sigma}_t^{-1} g(x_t)}{\operatorname{tr}\left(\frac{1}{t}\hat{\Sigma}_t^{-1} \sum_{i=1}^t g(x_i) g(x_i)^\top\right)}. \tag{4}$$

Empirically, we find that this auto-tuning of the probability mass on each sample to be far more effective than using a fixed value for z_t. In Figure 1, we demonstrate that this approach not only consistently matches the desired labeling frequency q, but it also distributes our labeling budget equitably across the data stream. This is evident from the identity line between proportion of data seen and budget consumed. In contrast, fixed values of z_t can drastically oversample or undersample, by a degree that varies with each round of active learning. Further, because of the nature of the determinant, these fixed-z_t versions often sample far more aggressively in the beginning of the stream than the end.

The complete VeSSAL approach is presented as Algorithm 1. In it, the estimated covariance over all samples is denoted as A, which is initialized to all zeros. We increment $\hat{\Sigma}$ efficiently using a Woodbury update on each chosen sample (Woodbury, 1950).

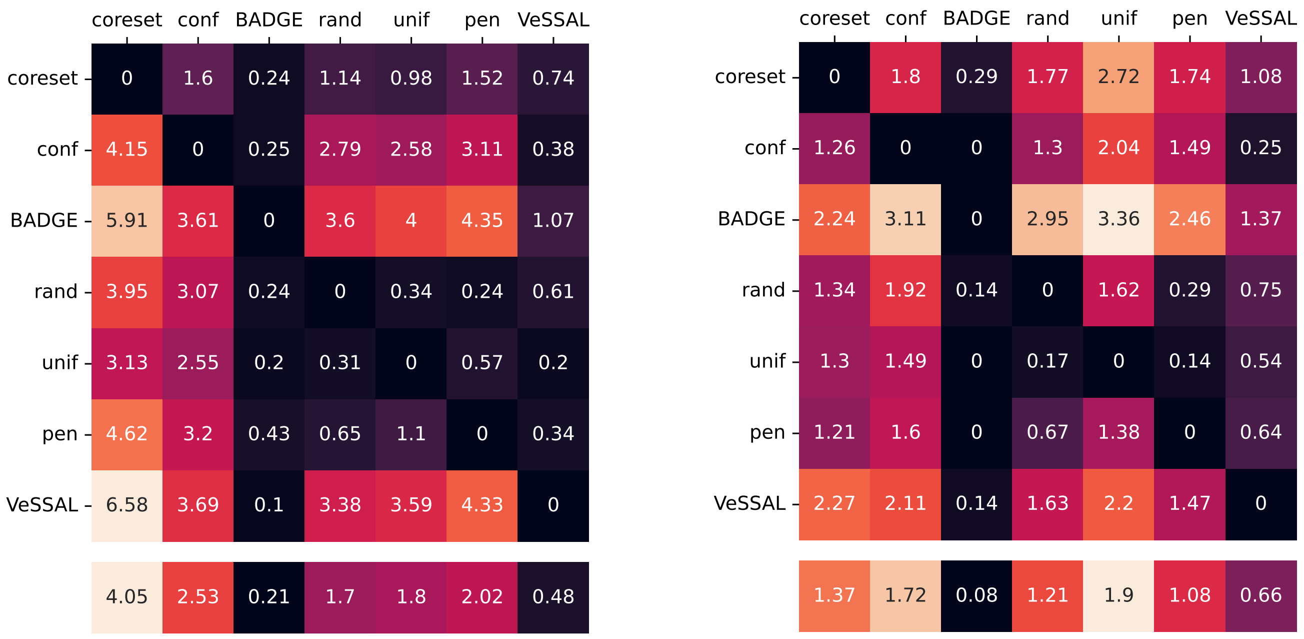
One interesting note is that our estimate for z_t is only as good as our estimate of $\mathbb{E}_x\left[g(x)g(x)^\top\right]$, which we obtain as a simple average of outer products of the $g(x_t)$ vectors observed in the stream. A consequence of this is that the estimate will be biased if data are ordered in the stream in a non-I.I.D. fashion. In the following section, we empirically demonstrate that this appears to not be an issue — VeSSAL performs on-par with state-of-the-art, non-streaming algorithms regardless of how data are ordered.

4. Experiments

We evaluate the performance of VeSSAL against several baselines on three academic benchmark datasets and one real-world dataset. In addition to measuring performance for a variety of architectures and batch sizes, we evaluate our approach in terms of robustness to feature drift in the data stream, and in terms of its fidelity to the predefined query rate.

Baselines. We compare our method against the following set of baselines. Most of these are non-streaming, meaning they have access to all unlabeled data when selecting samples to query. We also introduce two streaming baselines.

- **BADGE:** A recent, hyperparameter-free approach that incorporates both uncertainty and diversity in sampling using k-means++ in the hallucinated gradient space (Eq. 1) (Ash et al., 2020) (non-streaming).



	coreset	conf	BADGE	rand	unif	pen	VeSSAL
coreset	0	1.6	0.24	1.14	0.98	1.52	0.74
conf	4.15	0	0.25	2.79	2.58	3.11	0.38
BADGE	5.91	3.61	0	3.6	4	4.35	1.07
rand	3.95	3.07	0.24	0	0.34	0.24	0.61
unif	3.13	2.55	0.2	0.31	0	0.57	0.2
pen	4.62	3.2	0.43	0.65	1.1	0	0.34
VeSSAL	6.58	3.69	0.1	3.38	3.59	4.33	0
	4.05	2.53	0.21	1.7	1.8	2.02	0.48

(a) I.I.D. Data Stream

	coreset	conf	BADGE	rand	unif	pen	VeSSAL
coreset	0	1.8	0.29	1.77	2.72	1.74	1.08
conf	1.26	0	0	1.3	2.04	1.49	0.25
BADGE	2.24	3.11	0	2.95	3.36	2.46	1.37
rand	1.34	1.92	0.14	0	1.62	0.29	0.75
unif	1.3	1.49	0	0.17	0	0.14	0.54
pen	1.21	1.6	0	0.67	1.38	0	0.64
VeSSAL	2.27	2.11	0.14	1.63	2.2	1.47	0
	1.37	1.72	0.08	1.21	1.9	1.08	0.66

(b) Non-I.I.D. Data Stream

Figure 3. Pairwise penalty matrix for all experiments with (a) I.I.D. data streams and (b) Non-I.I.D data streams. Each cell corresponds roughly to the amount of times the row algorithm outperforms the column algorithm by a statistically significant amount. Averages are shown at the bottom, where lower values imply better-performing algorithms. VeSSAL is the highest-performing streaming approach, and is only bested by BADGE, a non-streaming baseline.

- **rand:** A naive random sampling baseline (non-streaming).
- **conf:** An uncertainty-based method that selects samples with smallest probability $p_{\hat{y}}$ of the top predicted class: $p_{\hat{y}} = \max f(x, \theta)$ (Wang & Shang, 2014) (non-streaming).
- **coreset:** A diversity-based method that uses a greedy approximation to the k-center problem on representations from the model's penultimate layer (Sener & Savarese, 2018) (non-streaming).
- **VeSSAL-pen:** This baseline is similar to VeSSAL but uses the penultimate layer embeddings instead of hallucinated gradients of the last layer, making it a purely diversity-based hyperparameter-free approach (streaming).
- **stream-uniform:** A naive baseline for the streaming setting where data points are sampled at a fixed frequency as they arrive (streaming).

At each round of active learning, streaming algorithms are only permitted to see each unlabeled example once, at whatever time it is presented. Further, they must commit to a labeling decision as soon as a sample is encountered, and are unable to refine their decisions as more data arrive. This puts the streaming approaches at a marked disadvantage in comparison to their non-streaming peers.

Datasets. We evaluate all algorithms on three image benchmarks, namely `SVHN` (Netzer et al., 2011), `MNIST` (LeCun et al., 1998), and `CIFAR10` (Krizhevsky, 2009), and one real-world dataset from Bohus et al. (2022). We refer to this dataset as `CLOW` and use it with permission from the authors. `CLOW` is collected through an augmented reality (AR) human-computer interaction device, where users provide object labels through a headset as they interact with objects in their home. This dataset includes 43 object classes and $\sim$ 11K training samples. More details about the dataset and its preprocessing are described in Appendix A.2.

Setup. We perform multiple rounds of active learning in all experiments, with a fixed budget k in each round. We consider a single round of active learning to correspond to a single batch of acquired points. We conduct as many rounds as are required to fully label the dataset under consideration. On these datasets, where the number of candidates is known, we choose to let the target rate q_t evolve throughout the streaming process as $q_t = \frac{k-|B_t|}{n-t}$, where n is the total number of samples in the unlabeled pool and B_t is the set of sam-

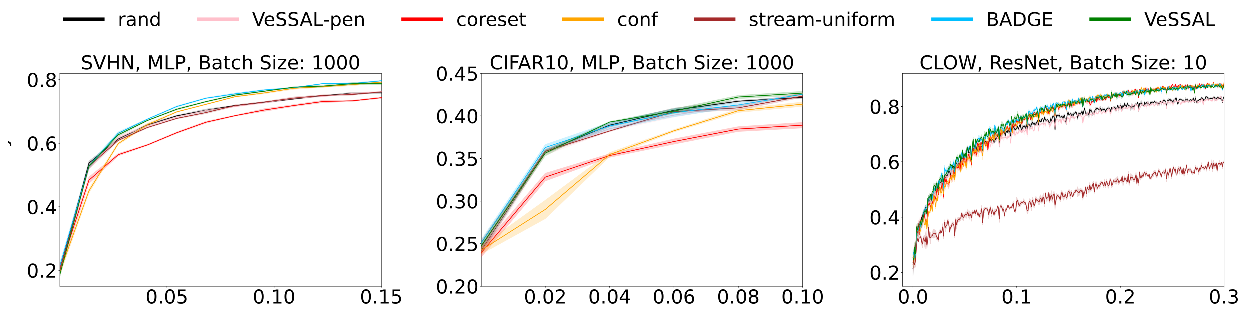


Figure 4. Learning curves for different neural active learning methods tested with non-I.I.D. data streams. Two network architectures, two batch sizes, and three datasets are shown here. These plots have been zoomed to highlight discriminative regions, but complete results are shown in the Appendix and are aggregated in Figure 3(b).

ples that have been added to the batch so far. This allows the algorithm to adjust its sampling behavior in case of a flawed approximation of $\mathbb{E}_x[g(x)g(x)^\top]$. Still, even with this precaution, the sampling rate seems to be somewhat constant at $\frac{k}{n}$ (Figure 1). We emphasize that in the case of real-world streaming settings, where the total number of unlabeled samples is unknown, q_t can be set to any desired frequency.

It is worth mentioning that this setup technically makes our approach committal only for a fixed round of active learning. That is, a sample that is not selected at one round will be made available again on the next. Our approach is made to work with truly committal environments, but we adopt this setup so that we can readily compare with non-committal, non-streaming, batch-mode active learning benchmarks.

After each round of data acquisition, we train the models from scratch and measure accuracy on a held-out test set. We primarily experiment with two architectures, a two-layer MLP and an 18-layer ResNet (He et al., 2016). All datasets are considered with both architectures except for `MNIST`, for which we only use the MLP. Models are trained with the Adam optimizer (Kingma & Ba, 2014) with a fixed learning rate 0.001 until they reach $> 99\%$ training accuracy. We experiment with different budgets per round $k \in \{100, 1\text{K}, 10\text{K}\}$ for the benchmark datasets and $k \in \{10, 100, 1\text{K}\}$ for the `CLOW` dataset (since it has a total of $\sim$ 11K training samples). Each experiment was repeated three times, and we show both mean and standard error in our learning curve plots. In all experiments we start with 100 labeled samples and acquire the rest of the labeled samples via active learning. All methods are implemented in PyTorch (Paszke et al., 2017). We set λ in Algorithm 1 to .01 in all VeSSAL experiments, which ensures a numerically stable inversion.

In the subsections that follow, we show experimental results under two different assumptions about the data stream: I.I.D data streams and adversarially ordered data streams.

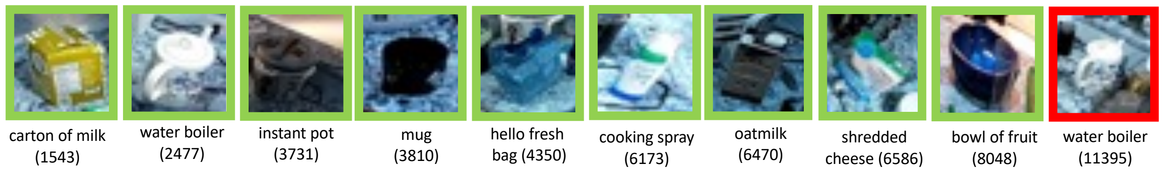
4.1. I.I.D. Data Stream

In this section we discuss experimental results when the data stream is randomized, meaning there is no induced correlation between consecutive data points or in the order of samples in the stream. We show that VeSSAL is superior or equal to non-streaming algorithms in this setting.

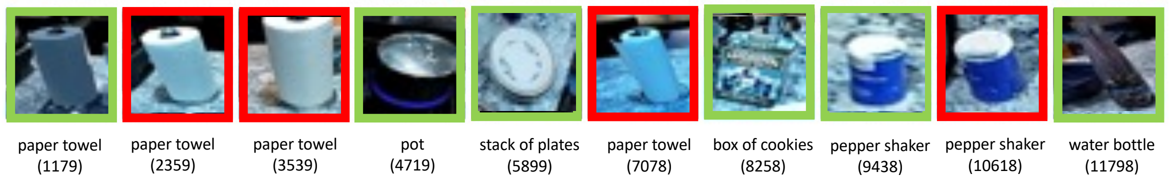
Representative learning curves averaged over three replicates for various choices of batch size, dataset, and model architecture in this setting are shown in Figure 2. In each, VeSSAL performs about as well as the highest-performing, non-streaming baseline, despite being restricted to the demands of the streaming setting. Detailed results for each combination of dataset, batch size, and network architecture are shown in Appendix B.

We aggregate results following the protocol of Ash et al. (2020). For each active learning experiment, we only consider a subset of labeling budgets where learning is still progressing. This is due to the fact that with labeling budgets approaching the data-set size, all algorithms achieve similar accuracy. At exponentially spaced intervals, we calculate if a given row algorithm outperforms a given column algorithm by a statistically significant margin according to a two-sided t-test. When this happens we increment the corresponding cell of Figure 3(a) by 1 divided by the total number of evaluations in the experiment. More experimental details can be found in Appendix A.1.

Here, higher-performing algorithms are associated with a lower column-wise average, displayed at the bottom of the figure. We see that overall, VeSSAL is the highest performing streaming method. When considering all baselines, VeSSAL is only outperformed by BADGE, which is not encumbered by streaming requirements. In a small number of experiments, VeSSAL surprisingly even manages to outperform BADGE.



(a) VeSSAL



(b) Uniform Rate Sampling

Figure 5. The first round of queries on the data stream from the `CLOW` dataset for different streaming active learning methods with an MLP and budget $k = 10$ (top: VeSSAL, bottom: uniform rate sampling). Red boxes denote repeated classes and green boxes denote unique classes. VeSSAL only repeats a single class, corresponding to an object view that is quite different than the other selection of the same class. Each queried image is described by its label name and time stamp (in parentheses) depicting the index at which it arrives in the stream. Here, there are about 11.8k candidates, and the indices suggest that sampling mass is well-distributed across the stream.

4.2. Non-I.I.D. Data Stream

To investigate the robustness of our algorithm to non-I.I.D. circumstances, we compare all methods under naturally occurring or artificially induced domain drift, where the observed data distribution is non-stationary. Note that non-streaming baselines are not at all affected by this change, and it is only a burden for streaming approaches that use sequential estimates of data statistics for decision making. Despite the disadvantage, we show that VeSSAL is still performing roughly as well as state-of-the-art, non-streaming baselines.

Artificial Drift We adversarially sort the unlabeled data to introduce domain drift. To do so, sort two academic datasets (`CIFAR-10`, `SVHN`) by their first principal component.

Natural Drift The `CLOW` data stream is sorted by the timestamp at which objects were encountered and labeled by a user, and hence naturally contains feature drift (Figure 5). For this dataset, MLP and ResNet-18 architectures have pretrained components. In the MLP case, we use data representations taken from the visual encoder of the multimodal CLIP model (Radford et al., 2021), and although the MLP is trained from scratch at each round, the CLIP feature extractor is fixed. In the ResNet case, we use a model that has been pretrained on ImageNet (Russakovsky et al., 2015), and refine the entire model with actively selected data. Each time new data are acquired, the ResNet is reset to to the ImageNet pretrained weights before being updated.

Figure 4 highlights the effectiveness of VeSSAL under the challenging setting of feature drift in the data stream. VeSSAL performs both on par with other non-streaming skyline approaches and better than other streaming baseline methods. Detailed learning curves for all datasets, architectures, and hyperaparameters are shown in Appendix C. In Figure 3(b), a pairwise comparison matrix analogous to Figure 3(a) shows that, with the exception of BADGE, VeSSAL outperforms all streaming and non-streaming baselines in presence of distribution shift. Again, VeSSAL sometimes outperforms BADGE even though BADGE sees all unlabeled data before sampling. Figure 5 contains qualitative evidence that VeSSAL samples diverse images even when data points are correlated and the streaming uniform sampling baseline repeatedly queries duplicate objects.

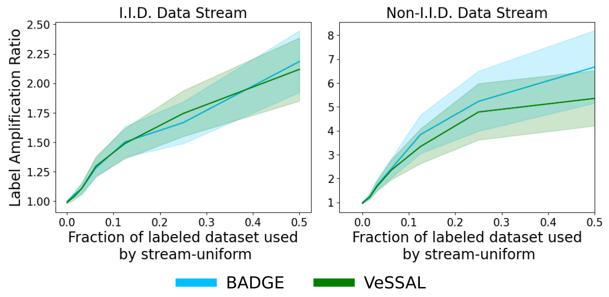


Figure 6. Label amplification for BADGE and VeSSAL with respect to stream-uniform, averaged over all experiments. VeSSAL achieves roughly the same label efficiency as BADGE in the I.I.D. case, even though VeSSAL is limited by streaming requirements. In the non-I.I.D. setting, which is adversarial for streaming algorithms like VeSSAL but not for pool-based algorithms like BADGE, VeSSAL only does slightly worse than BADGE.

4.3. Label Amplification

The promise of active learning is that it can deliver significantly more predictive power for a fixed labeling budget than naive sampling. This is demonstrated by Figure 6, where algorithm performance is cast in terms of "label amplification" instead of accuracy. As learning progresses, we plot the ratio between the number of samples used by streaming uniform sampling and the number of samples required by active sampling to achieve the same performance. For a good active learning algorithm, labeling amplification will be much larger than one, reflecting the increase in labeling efficiency over passive sampling. Our plots average over all experiments conducted, and despite VeSSAL being constrained to the streaming, committal setting, they show that it is roughly as efficient as BADGE in the I.I.D. streaming case and only slightly worse in the non-I.I.D. case.

4.4. Compute Requirements

VeSSAL is able to decide whether to include a given unlabeled sample in the batch as soon as it is encountered. In doing so, and unlike previous neural batch active learning algorithms, VeSSAL does not need to compare every unlabeled candidate point to every sample that has already been selected. Particularly for large batch sizes, this makes VeSSAL substantially more time efficient than baseline neural active learning algorithms. Figure 7 (logged y-axis version shown in Appendix Figure 15) demonstrates this by comparing the run time of several active learning algorithms as a function of their query batch size using the CIFAR-10 dataset. VeSSAL enjoys run times times that stay nearly fixed for increasing batch sizes, while other approaches have compute requirements that grow superlinearly. For large batch sizes, VeSSAL can be more efficient than its counterparts by several orders of magnitude.

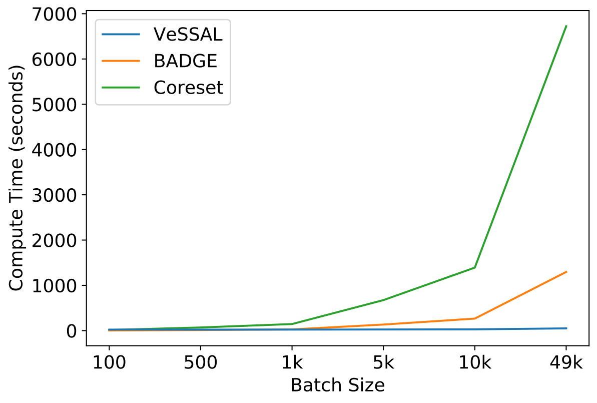


Figure 7. The compute time required to select a batch for three different algorithms on CIFAR-10 as a function of the query batch size. While non-streaming algorithms require compute that grows superlinearly as a function of labeling budget, VeSSAL stays relatively constant. Results are averaged over five replicates and each algorithm was given identical computational resources.

5. Discussion

We presented VeSSAL, a new approach for batch active learning with deep neural networks in a streaming setting. Unlike prior pool-based active learning approaches for deep neural networks, our method can commit to queries as soon as samples are made available to the model from a data stream. Even in fixed-data settings, VeSSAL performs roughly as well as state-of-the-art methods — despite the fact that they are not hindered by streaming constraints. We envision several potential benefits of the proposed approach, expanding the applicability of neural networks for real-world, interaction-centric applications. Our algorithm can be run in settings that are inherently streaming, committal, or on datasets too large to be entirely stored in one place.

Our work also opens up exciting directions for future research. Currently, VeSSAL assumes that the number of label classes is known a priori. In real-world applications, where an interaction environment continues to evolve, the number of label classes may grow over time. Moreover, some queries can be costlier than others. For example, asking a user for an object label in their peripheral vision could distract them from performing the current task. Addressing these challenges are exciting topics for further work.

Acknowledgements

The authors would like to thank Sean Andrist, Dan Bohus, and the Platform for Situated Intelligence (PSI) team at Microsoft Research Redmond for providing inspiration on the problem statement, feedback on our approach, and access to the `CLOW` dataset.

References

Ash, J., Goel, S., Krishnamurthy, A., and Kakade, S. Gone fishing: Neural active learning with fisher embeddings. *Advances in Neural Information Processing Systems*, 34: 8927–8939, 2021.

Ash, J. T. and Adams, R. P. On warm-starting neural network training. *Advances in Neural Information Processing Systems*, 2020.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. Deep batch active learning by diverse, uncertain gradient lower bounds. *International Conference on Learning Representations*, 2020.

Ban, Y., Zhang, Y., Tong, H., Banerjee, A., and He, J. Improved algorithms for neural active learning. *Advances in eural Information Processing Systems*, 2022.

Bardenet, R., Lavancier, F., Mary, X., and Vasseur, A. On a few statistical applications of determinantal point processes. *ESAIM: Proceedings and Surveys*, 60:180–202, 2017.

Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9368–9377, 2018.

Beygelzimer, A., Dasgupta, S., and Langford, J. Importance weighted active learning. In *Twenty-Sixth International Conference on Machine Learning*, 2009.

Beygelzimer, A., Hsu, D. J., Langford, J., and Zhang, T. Agnostic active learning without constraints. In *Neural Information Processing Systems*, 2010.

Bhaskara, A., Lattanzi, S., Vassilvitskii, S., and Zadimoghaddam, M. Residual based sampling for online low rank approximation. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1596–1614. IEEE, 2019.

Bingham, E. and Mannila, H. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250, 2001.

Bohus, D., Andrist, S., Feniello, A., Saw, N., and Horvitz, E. Continual learning about objects in the wild: An interactive approach. In *Proceedings of the 2022 International Conference on Multimodal Interaction*, pp. 476–486, 2022.

Boutsidis, C., Garber, D., Karnin, Z., and Liberty, E. Online principal components analysis. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pp. 887–901. SIAM, 2014.

Braverman, V., Meyerson, A., Ostrovsky, R., Roytman, A., Shindler, M., and Tagiku, B. Streaming k-means on well-clusterable data. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pp. 26–40. SIAM, 2011.

Brust, C.-A., Käding, C., and Denzler, J. Active learning for deep object detection. *arXiv preprint arXiv:1809.09875*, 2018.

Choi, J., Elezi, I., Lee, H.-J., Farabet, C., and Alvarez, J. M. Active learning for deep object detection via probabilistic modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10264–10273, 2021.

Dasgupta, S. Two faces of active learning. *Theoretical computer science*, 2011.

Deshpande, A. and Vempala, S. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 292–303. Springer, 2006.

Deshpande, A., Rademacher, L., Vempala, S. S., and Wang, G. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(1):225–247, 2006.

Ducoffe, M. and Precioso, F. Adversarial active learning for deep networks: a margin based approach. *arXiv:1802.09841*, 2018.

Frieze, A., Kannan, R., and Vempala, S. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, 2017.

Geifman, Y. and El-Yaniv, R. Deep active learning over the long tail. *arXiv:1711.00941*, 2017.

Ghashami, M. and Phillips, J. M. Relative errors for deterministic low-rank matrix approximations. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 707–717. SIAM, 2014.

Gissin, D. and Shalev-Shwartz, S. Discriminative active learning. *arXiv:1907.06347*, 2019.

Greub, W. H. *Linear algebra*, volume 23. Springer Science & Business Media, 2012.

Gudovskiy, D., Hodgkinson, A., Yamaguchi, T., and Tsukizawa, S. Deep active learning for biased datasets via fisher kernel self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9041–9049, 2020.

Hanneke, S. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 2014a.

Hanneke, S. Theory of active learning. *Foundations and Trends in Machine Learning*, 7(2-3), 2014b.

Hanneke, S. and Yang, L. Minimax analysis of active learning. *J. Mach. Learn. Res.*, 16(1):3487–3602, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hsu, D. J. *Algorithms for active learning*. PhD thesis, UC San Diego, 2010.

Huang, T.-K., Agarwal, A., Hsu, D. J., Langford, J., and Schapire, R. E. Efficient and parsimonious agnostic active learning. *Advances in Neural Information Processing Systems*, 28, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kovashka, A., Russakovsky, O., Fei-Fei, L., Grauman, K., et al. Crowdsourcing in computer vision. *Foundations and Trends® in computer graphics and Vision*, 10(3): 177–243, 2016.

Krishnamurthy, A., Agarwal, A., Huang, T.-K., Daumé III, H., and Langford, J. Active learning for cost-sensitive classification. In *International Conference on Machine Learning*, pp. 1915–1924. PMLR, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Kulesza, A., Taskar, B., et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

Lavania, C., Wei, K., Iyer, R., and Bilmes, J. A practical online framework for extracting running video summaries under a fixed memory budget. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 226–234. SIAM, 2021.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *IEEE*, 1998.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. SSD: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

MacKay, D. J. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Roth, D. and Small, K. Margin-based active learning for structured output spaces. In *European Conference on Machine Learning*, 2006.

Roy, S., Unmesh, A., and Namboodiri, V. P. Deep active learning for object detection. In *BMVC*, pp. 91, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252, 2015.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

Senzaki, Y. and Hamelain, C. Active learning for deep neural networks on edge devices. *arXiv preprint arXiv:2106.10836*, 2021.

Settles, B. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

Singh, K. K., Fatahalian, K., and Efros, A. A. Krishnacam: Using a longitudinal, single-person, egocentric dataset for scene understanding tasks. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–9. IEEE, 2016.

Strang, G. *Linear algebra and its applications.* Belmont, CA: Thomson, Brooks/Cole, 2006.

Sun, L. and Gong, Y. Active learning for image classification: A deep reinforcement learning approach. In *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*, pp. 71–76. IEEE, 2019.

Sun, S., Calandriello, D., Hu, H., Li, A., and Titsias, M. Information-theoretic online memory selection for continual learning. *journal=International Conference on Learning Representations*, 2022.

Wang, D. and Shang, Y. A new active labeling method for deep learning. In *International Joint Conference on Neural Networks*, 2014.

Wang, J., Wang, X., Shang-Guan, Y., and Gupta, A. Wanderlust: Online continual object detection in the real world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10829–10838, 2021.

Woodbury, M. A. *Inverting modified matrices*. Statistical Research Group, 1950.

A. Additional Experimental Details

A.1. Heatmaps for pairwise comparisons between algorithms

Figures 3(a) and 3(b) show a comprehensive pairwise comparison of all algorithms, summarizing experiment results over all datasets (D), architectures (A), batch sizes (k), and total labeling budgets (L). Intuitively, entry $M_{i,j}$ in the matrix is the number of settings where algorithm i outperforms algorithm j by a statistically significant amount. For each active learning experiment, we only consider labeling budgets r where a random strategy has not yet hit 99% of its final performance. We checkpoint the learning curves at exponentially spaced intervals until reaching this point, $L_i = N_0 + 2^i k \leq r$, $L = [L_0, L_1, \cdots]$, for an number of seed samples N_0.

For each (D, A, k, L) combination, at each we have 3 scores for algorithm i, and 3 scores for algorithm j, leading to 3 score deltas $d^1_{i,j}, d^2_{i,j}, d^3_{i,j}$ (since each experiment was repeated 3 times). We apply the two-sided t-test on these score deltas to decide if an algorithm significantly wins a pairwise comparison. Specifically we compute the t-score $t = \sqrt{3}\mu/\sigma$ as follows:

$$\mu = \frac{1}{3}\sum_{s=1}^{3} d^s_{i,j} \qquad \sigma = \sqrt{\frac{1}{2}\sum_{s=1}^{3}(d^s_{i,j} - \mu)^2}$$

If the $t > 2.92$ algorithm i is considered to outperform algorithm j, and vice versa if $t < -2.92$. Suppose there are $n_{D,A,B}$ labeling budgets for each (D, A, B) combination. Then when an algorithm i wins a pairwise comparison with algorithm j, a value of $\frac{1}{|L|}$, where $|L|$ is the total number of evaluations in the experiment, is added to the corresponding entry in the matrix $M_{i,j}$.

A.2. Details of the `CLOW` dataset

Bohus et al. (2022) introduced a dataset collected via a mixed-reality interactive approach for continual learning about objects in the wild (CLOW). We refer to this dataset as `CLOW` in our work. The dataset was collected at two sites – home environments of end-users wearing a mixed-reality headset as they go about performing everyday tasks. We use data from Site 2 as reported in (Bohus et al., 2022). The mixed-reality multimodal interface enables users to label objects in their surroundings via their gaze, speech, and gestures. The system uses color and depth cameras to detect and track objects in the scene. Different views of an object are captured by the device as the user interacts with and labels it. The stream of images thus naturally exhibits correlation due to the notion of object permanence in the environment across time. The labels from the user are linked to multiple object views collected over time, and can be used to improve object recognition over time. While the object label collection occurs via a user-initiated approach, the authors of this work identify the need for an active learning solution to minimize the labeling burden on a user while an object recognition model continues to learn about objects in the environment.

Preprocessing the `CLOW` dataset: `CLOW` is collected at two home environments (Site1 and Site 2) (Bohus et al., 2022), with streaming images of objects being recorded at 5Hz. We use data from Site 2 which consists of a total of 47 object classes and 55,657 object images. Bohus et al. (2022) filter this dataset to remove images with blur and occlusion via hand-defined thresholds on linear and angular speed of the headset as well as the overlap of human hands with object views. The filtered version of their dataset at Site 2 consists of 15,095 images from 47 object classes. We further filter the data stream from Site 2 by removing images with an interaction of less than 5 frames (1 second), i.e. if an object appears for less than 5 consecutive frames in the image stream, then we discard images from such a short interaction. This provides us a total of 14,981 images from 43 object classes. We split this data stream into train and test sets based on the following criteria: for each user interaction with an object instance, the first 80% are used as part of the train set and the last 20% of the interaction is used as part of the test set. This provides us with 11,899 training images and 3082 test images. This strategy ensures that we have 20% of the data for each object in the stream as part of the evaluation set. The distribution of the 11,899 training images into different object classes is shown in Fig 8. We downsample each image to size $32 \times 32 \times 3$ before passing it as an input to the CLIP visual encoder model (Radford et al., 2021) or the ImageNet pretrained ResNet-18 model (Russakovsky et al., 2015). To efficiently compute the covariance matrix as part of VeSSAL for this dataset with 43 object classes, we employ random projection (Bingham & Mannila, 2001) to reduce the dimensionality of the gradient embeddings to size 2560.

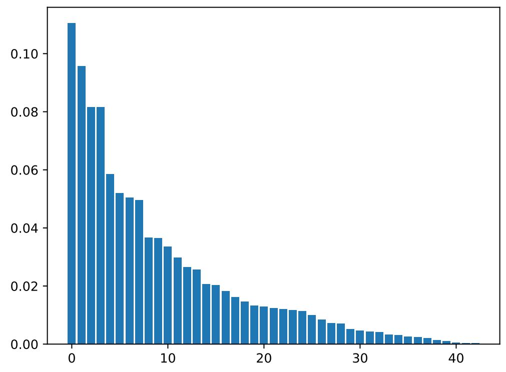


Figure 8. Distribution of 43 class labels from the Site2 `CLOW` dataset. The y-axis denoted the fraction of images that belong to a specific class label (class ID denoted by the indices of the x-axis).

B. Learning curves for I.I.D. Data Stream Experiments

We show learning curves below for all experiments with randomized data streams from the following datasets: `SVHN` (Fig. 9), `CIFAR-10` (Fig. 10), `MNIST` (Fig. 11)).

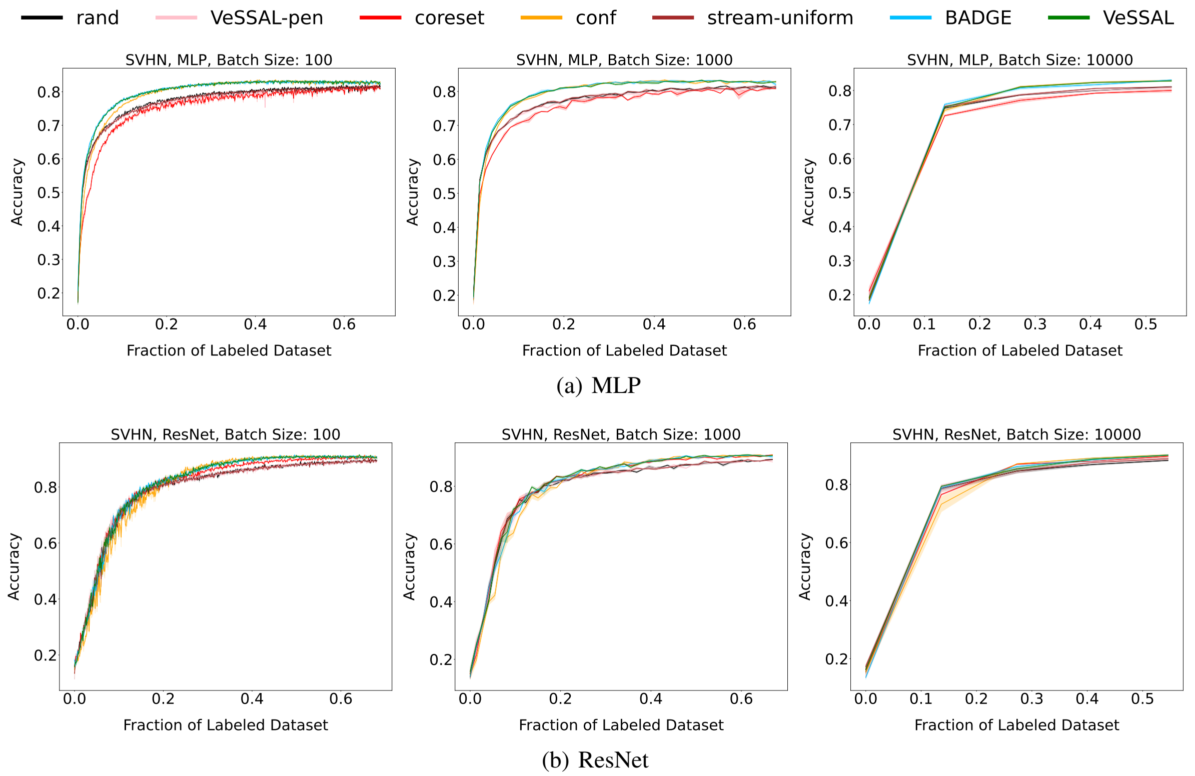
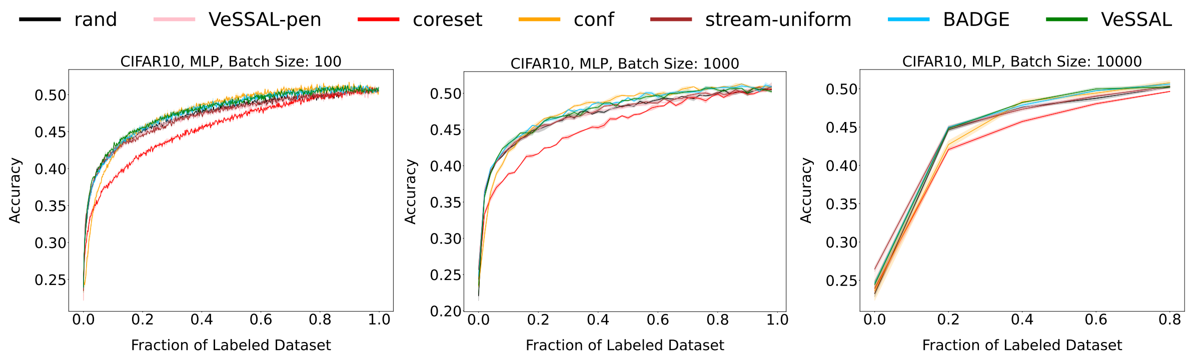
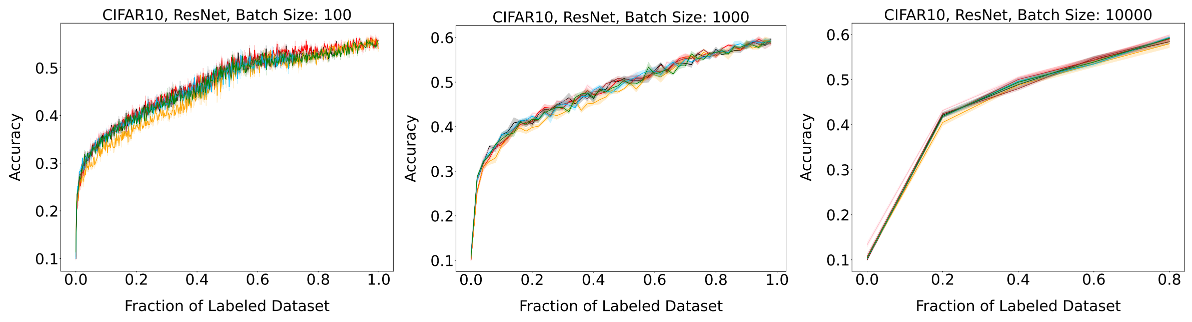


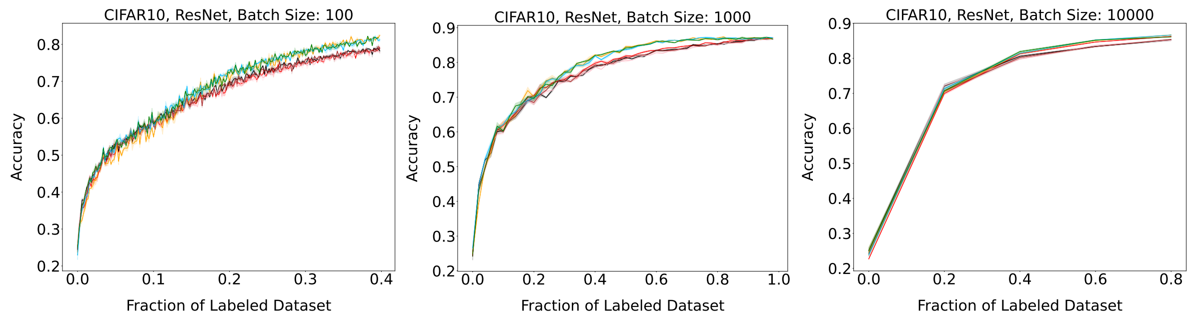
Figure 9. Learning curves for the `SVHN` dataset with two network architectures and three batch sizes. Streaming active learning algorithms observe data in a randomized order.



(a) MLP



(b) ResNet



(c) ResNet w/ Data Augmentation

Figure 10. Learning curves for the `CIFAR-10` dataset with two network architectures (MLP, ResNet) and three batch sizes (100, 1000, 10000). We show results for both ResNet training (b) without data augmentation (similar to Ash et al. (2020)) and (c) with data augmentation. Streaming active learning algorithms observe data in a randomized order.

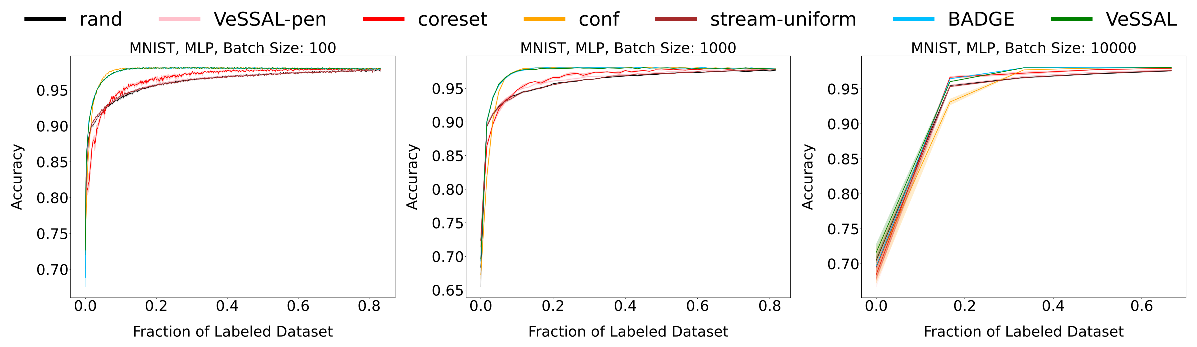


Figure 11. Learning curves for the `MNIST` dataset with the MLP network architecture and three batch sizes. Streaming active learning algorithms observe data in a randomized order.

C. Learning curves for non-I.I.D. Data Stream Experiments

We show learning curves for all experiments with non-i.i.d. data streams from the following datasets: `SVHN` (Fig. 12), `CIFAR-10` (Fig. 13), `CLOW` (Fig. 14)).

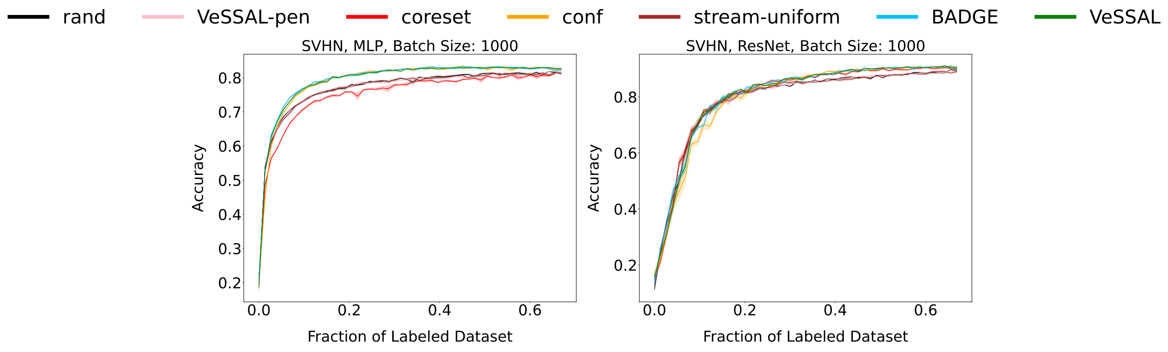


Figure 12. Learning curves for the `SVHN` dataset with two network architectures and batch size 1000. Streaming active learning algorithms observe data sorted by their 1st principal component.

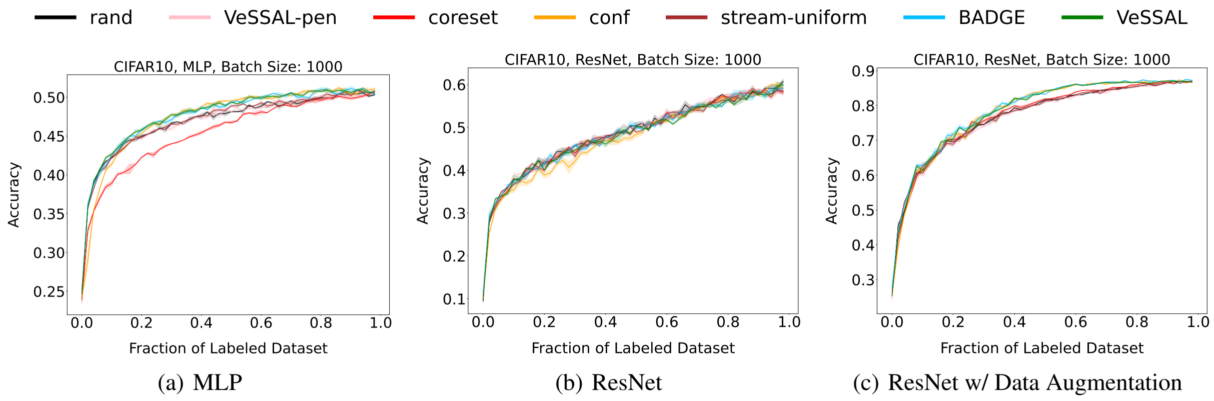
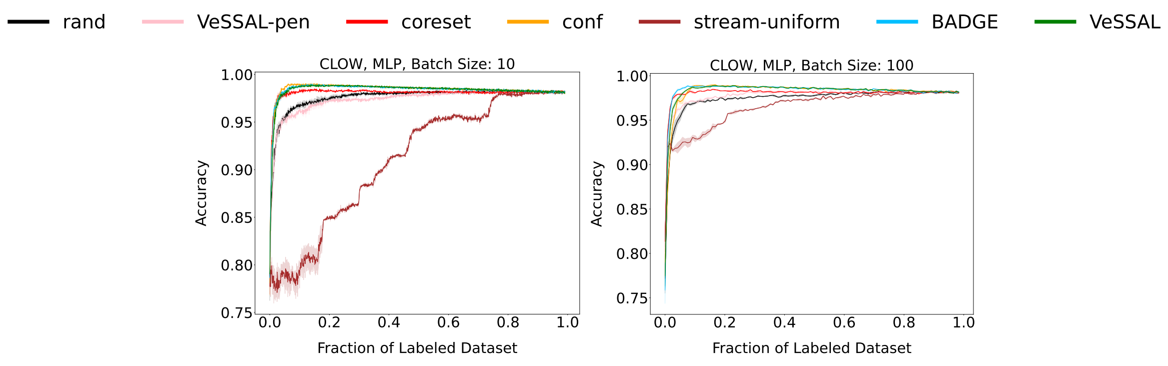
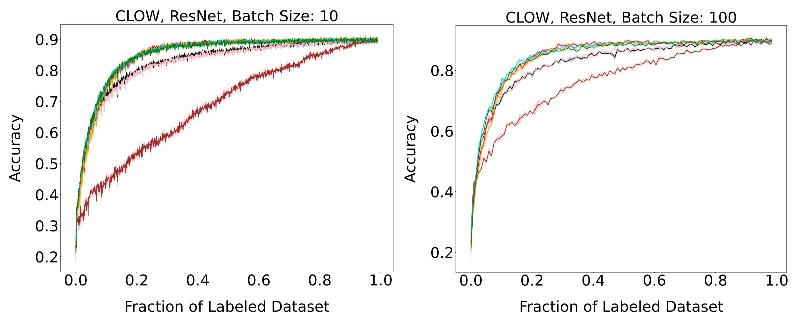


Figure 13. Learning curves for the `CIFAR-10` dataset with two network architectures and batch size 1000. Streaming active learning algorithms observe data sorted by their 1st principal component.



(a) MLP



(b) ResNet

Figure 14. Learning curves for the `CLOW` dataset with two network architectures and two batch sizes. Streaming active learning algorithms observe object images ordered by timestamps at which users interacted with and provided a label for the corresponding object.

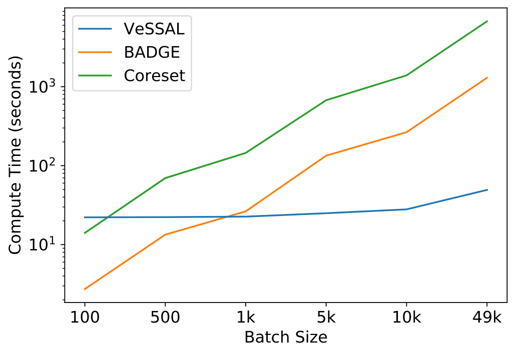


Figure 15. A y-axis logged version of Figure 7, showing compute time required to select a batch for three different algorithms on CIFAR-10 as a function of the query batch size. While non-streaming algorithms require compute that grows superlinearly as a function of labeling budget, VeSSAL stays relatively constant. Results are averaged over five replicates and each algorithm was given identical computational resources.

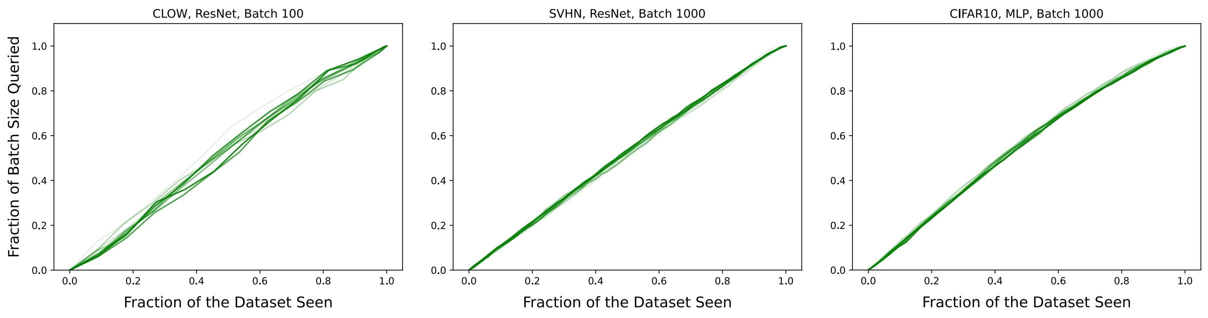


Figure 16. VeSSAL distributes its labeling budget equitably across different non-I.I.D. data streams. Here we show several different datasets, acquisition batch sizes, and model architectures, and all datasets (excluding `CLOW`, which is already non-I.I.D.) are sorted by their first principal component.