# CLACTIVE: EPISODIC MEMORIES FOR RAPID ACTIVE LEARNING

**Sri Aurobindo Munagala**[1] **Sidhant Subramanian**[1] **Shyamgopal Karthik**[2] **Ameya Prabhu**[3] **Anoop Namboodiri**[1]
IIIT Hyderabad[1] University of Tübingen[2] University of Oxford[3]

## ABSTRACT

Active Learning aims to solve the problem of alleviating labelling costs for large-scale datasets by selecting a subset of data to effectively train on. Deep Active Learning (DAL) techniques typically involve repeated training of a model for sample acquisition over the entire subset of labelled data available in each round. This can be prohibitively expensive to run in real-world scenarios with large and constantly growing data. Some work has been done to address this – notably, Selection-Via-Proxy (SVP) proposed the use of a separate, smaller "proxy" model for acquisition. We explore further optimizations to the standard DAL setup and propose CLActive: an optimization procedure that brings significant speedups which maintains a constant training time for the selection model across rounds and retains information from past rounds using Experience Replay. We demonstrate large improvements in total train-time compared to the fully-trained baselines and SVP. We achieve up to 89×, 7×, 61× speedups over the fully-trained baseline at 50% of dataset collection in CIFAR, Imagenet and Amazon Review datasets, respectively, with little accuracy loss. We also show that CLActive is robust against catastrophic forgetting in a challenging class-incremental active-learning setting. Overall, we believe that CLActive can effectively enable rapid prototyping and deployment of deep AL algorithms in real-world use cases across a variety of settings.

## 1 INTRODUCTION

Deep Active Learning (DAL) aims to alleviate the prohibitive cost of large-dataset annotation through the use of selection models to acquire a labelled subset of data. A major focus of DAL literature has been to design effective *acquisition strategies* i.e. protocols for selecting unlabelled samples to annotate. A standard DAL pipeline involves repeated training of a model and an acquisition strategy applied to the model's predictions on the unlabelled subset. However, selection models are typically trained from scratch at each round over the entire labelled set of data collected so far, thus leading to larger train times each round, which can be prohibitive.

There has been limited research into bringing down these exorbitant DAL selection times. Early works (Lewis & Catlett, 1994) explored using different classifiers for selection to cut computational expense. Selection-Via-Proxy (SVP) (Coleman et al., 2019) built on this and proposed the use of "proxy models", which were smaller counterparts to the larger models used for evaluation, and found that they achieved the same quality of samples in far less time. We propose a novel optimization procedure, CLActive, to improve train times by a further order of magnitude over past baselines to the standard DAL setup while maintaining accuracy. CLActive consists of the following improvements:

- First, incrementally update the selection model between rounds rather than train a new one every round.
- Second, use only a subset of labelled samples capped at a maximum size to train the selection model. This effectively cuts down training costs with minimal accuracy loss.
- Third, use an LR scheduler that converges faster than traditionally used schedulers, enabling training for fewer epochs each round before the selection model reaches convergence.

The use of an incremental model allows us to train for fewer epochs every round, which is further brought down with the help of fast LR schedulers. However, straightforward use of an incremental model by updating it exclusively on freshly labelled data may lead to *catastrophic forgetting*. This problem has been extensively studied in the field of Continual Learning (CL), and we use a popular CL strategy, *episodic memory*, to avoid this. Instead of training the selection model only on data acquired in the previous round, we train it on a mixture of new samples and previously seen samples. As we cap the maximum number of samples that the proxy model trains on in any given round, training time remains constant across rounds. We find that using our proposed CLActive optimization strategy achieves significant speedups in data selection compared to the old trained-from-scratch active learning baselines while also maintaining

comparable accuracies. We test extensively on the CIFAR-10, CIFAR-100, ImageNet and Amazon Review datasets for up to 50% of the datasets collected via queries and achieve 89x, 89x, 7x and 61x speedups.

Consider the extreme case where we remove the closed-world nature of the pool set in active learning: a completely unseen class is introduced to the dataset between rounds of selection. This is especially challenging for training an incremental model like CLActive, as it may learn disproportionately from the new class. Hence, the representations learnt would be poor, affecting the sample selection quality by not effectively representing the dataset, leading to lower evaluation accuracy. However, we demonstrate that CLActive is an effective optimization strategy even when samples from new unlabeled classes are gradually added to the dataset.

## 2 Related Works

### 2.1 Active Learning

Active Learning (AL) methods have been studied extensively in the past for classification and regression tasks (Settles, 2009). AL aims to solve the problem of minimizing sampling costs when dealing with large volumes of unlabelled data. Deep Active Learning (DAL) utilizes predictions from DNNs trained on partially sampled subsets of data to select additional samples for queries. Ren et al. (2020) offers a comprehensive survey of Deep Active Learning methods.

Query strategies can be broadly divided into uncertainty sampling and diversity sampling. Uncertainty based methods typically query for samples which the model is least certain about, which can be done through various heuristics (Gal et al., 2017). For example, given a probabilistic binary classifier, the sample with the posterior probability of being positive closest to 0.5 could be queried as the least certain sample (Lewis & Gale, 1994; Lewis & Catlett, 1994). Similarly, for a deep network with multiple categories, the samples with the lowest probability of the most likely predicted class could be queried. Diversity sampling methods search for samples that do not look like currently labelled samples. For example, Discriminative Active Learning (Gissin & Shalev-Shwartz, 2019) approached AL as a binary classification task, querying for samples with the least predicted certainty of belonging to either the labelled or unlabelled dataset. Sener & Savarese (2017) posed AL as a set-selection problem, with the aim of finding a small subset of data representative of the larger dataset. Ducoffe & Precioso (2018) used sensitivity adversarial perturbations as a tool to judge the distance of a sample from the decision boundary and inform sample selection.

Gissin & Shalev-Shwartz (2019) evaluated state-of-the-art AL sampling methods to find that none outperformed Uncertainty sampling for larger batch sizes as seen in DAL setups. Traditional AL works, like the ones above, have focused on finding better acquisition functions to achieve higher accuracies on the evaluation models. There has been little work towards cutting down the large train times required to run AL setups. Coleman et al. (2019) was one of the first works to explore speedups in selection time for AL algorithms, primarily proposing the use of "proxy" models for sample selection in lieu of the larger, original models. They found that the samples collected via smaller proxy models largely correlated with samples collected using the same model as the target in certain scenarios (e.g. using a ResNet-20 proxy model for a ResNet-164 target), achieving significant speedups in selection time while retaining final accuracies. Our work introduces modifications to the setup directly to bring further speedups, and we found that proxy modelling works well in tandem with our approach.

### 2.2 Continual Learning

*Catastrophic forgetting* is a phenomenon that was observed across many early studies in settings where the model would learn diverse examples in a sequential fashion. Continual Learning (CL) aims to reduce forgetting in these incremental settings. CL approaches can be broadly categorized (Delange et al., 2021; Prabhu et al., 2020) into (i) Replay methods, (ii) Regularization methods, (iii) Parameter isolation methods.

**Experience Replay**: Replay methods generally store samples in some form and "replay" them while learning new tasks. Many methods simply joint-train new tasks with subsets of stored samples (Rebuffi et al., 2017; Isele & Cosgun, 2018; Chaudhry et al., 2019). Rolnick et al. (2018) assumed a fixed storage budget and suggested reservoir sampling to select samples to remember. Continual Prototype Evaluation (De Lange & Tuytelaars, 2020) combined nearest-neighbour predictions and reservoir sampling while also not needing additional task information for data streams, unlike prior CL works. Chaudhry et al. (2019) jointly trained online samples with tiny episodic memories, finding that it improves generalization. Recent work (Prabhu et al., 2020; Chaudhry et al., 2019) has shown that experience replay using just memory is extremely effective in CL pipelines, outperforming other approaches. Simple sampling strategies like random sampling performed comparably to more complex strategies for experience replay. Additionally, given

**Algorithm 1** Active Learning with Uncertainty Sampling

**Input.** Unlabeled Data $D_U$, Initial labeled set $D_L$, Number of rounds $R$, Number of samples queried per round $q$, Epochs per round $e$
**Output.**

Initialize network with weights $W$
Train network over $D_L$
**for** $k$ in 1..$R$ **do**
    `//Query for samples`
    $L = \{L_i = \max_j p(y_i = j|x_i, W)\}$
    Acquire labels for $D_{query} = \{d : d \in D_U$ and $d \in bottomk(L)\}$
    $D_L = D_L \cup D_{query}$

    `//Train selection network`
    Create new model and initialize weights $W^k$
    Train network over $D_L$
**return** $D_L$

the minimal overhead to computation, we found that experience replay with reservoir sampling was highly suited to addressing catastrophic forgetting in class-imbalanced scenarios (section 4.4) in our CLActive pipeline.

**Constraining using Experience Replay:** Constrained optimization approaches constrain model training on new tasks such that previous tasks are not forgotten. Gradient Episodic Memory (GEM) (Lopez-Paz & Ranzato, 2017) did this by minimizing the loss for the current task by finding the closest gradient (L2 norm-wise) such that the loss at previous tasks does not increase at each parameter update. Average GEM (A-GEM) (Chaudhry et al., 2018) improved this by ensuring that the *average* loss over previous tasks does not increase instead of the *individual* loss, showing that this was much faster than GEM.

**Regularization**: Regularization-based approaches do not impose memory requirements and instead introduce extra regularization losses attempting to encapsulate prior knowledge on new tasks. Learning Without Forgetting (LWF) (Li & Hoiem, 2017) re-used model outputs as soft labels. Rannen et al. (2017) learned under-complete auto-encoders for each task to capture and then constrain significant features when new tasks were learned. Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) overcame catastrophic forgetting by "slowing down" updates on parameters which were important to older tasks. Parameter isolation methods attempt to separate and dedicate different parameters to different tasks to get around interference between tasks. Rusu et al. (2016); Xu & Zhu (2018), for example, had models with incrementally growing architectures, where a new branch would be adopted after freezing parameters after a task.

**Active Continual Learning**: Some recent works (Belouadah et al., 2020) have explored the use of AL sample acquisition to address class imbalances in CL settings. Ayub & Wagner (2021) explored AL in the context of robots, which learn from real-world inputs and thus may face catastrophic forgetting and proposed the use of using the cluster representation of known classes during acquisition. In our work, we explore whether the entire labelled subset is important for AL pipelines or whether it is possible to achieve the same quality of samples while training the intermediate models on smaller subsets of the labelled pool. We observe that in real-life cases where new classes of data may appear and the AL model queries them, catastrophic forgetting may occur if the model only learns from new data and thus introduce solutions inspired by CL approaches to mitigate this.

## 3 Approach

Let $D_U = \{x_i\}$ denote a dataset containing $|D_U| = N$ unlabelled i.i.d samples. An initial subset $S_0$ of samples is randomly chosen from $D_U$ and is sent to an oracle for annotation. In standard DAL pipelines, a model is trained from scratch on the labelled set $D_L = \{S_0, S_1...S_i\}$ at the *i*th round, and predicts labels for the rest of the unlabelled set. Using an acquisition strategy on these predictions, a new subset $D_{query} = \{S_{i+1}\}$ is queried for annotation. This process is repeated until a budget *b* of samples are annotated, and the evaluation model is trained on the final labelled subset $D_L = \{S_0 \cup S_1 \cup S_2...\}$. This is represented in algorithm 1. Selection-via-Proxy (SVP) (Coleman et al., 2019) improves the efficiency of this process by training a smaller proxy model from scratch.

**Algorithm 2** CLActive

**Input.** Unlabeled Data $D_U$, Initial labeled set $D_L$, Number of rounds $R$, Number of samples queried per round $q$, Memory $M$, Memory Ratio $R$ Epochs per round $e$
**Output.**

```
//Initialize Memory with D_L
M = D_L

Initialize network with weights W
Train network over D_L
for k in 1..R do
    //Query for samples
    L = {L_i = max_j p(y_i = j|x_i, W)}
    Acquire q labels for D_query = {d : d ∈ D_U and d ∈ bottomk(L)}
    D_L = D_L ∪ D_query

    //Add new samples to Memory
    M = M ∪ D_query

    //Incrementally train network
    M_k = Random Subset_R M
    D_train = M_k ∪ D_query
    Update network over D_train for e epochs
return D_L
```

In the CLActive pipeline (Algorithm 2), we introduce a few improvements to the above procedure for improving efficiency. Firstly, instead of training the SVP proxy model from scratch, we incrementally train the model over training data. However, the train set $\{S_0 \cup S_1 \cup S_2 ... S_i\}$ grows every round. We maintain a memory $M = \{S_0 \cup S_1 ... S_i\}$ which stores all sampled data at any given round $i$. At each round, we fetch a random subset $M_i \subset M$ consisting of $R \times q$ samples from memory, where $R$ is the memory ratio, and $q$ is the query batch size. We form a train set $D_{train}$ as a mixture of samples acquired the previous round $D_{query} = S_i$, and the set $M_i$, i.e.:

$$D_{train,i} = S_i \cup M_i$$

And we incrementally update the model over $D_{train,i}$. Note that the size of $D_{train,i}$ remains constant compared to the standard pipeline.

Summarizing, the main differences between standard AL training procedures and CLActive are outlined in algorithms 1 and 2. The most notable differences are (i) The use of an incrementally updating model in CLActive rather than training the selection model from scratch, (ii) The Use of memory $M$ and retrieval to form a small training set $D_{train}$ each round as opposed to training on the entire labelled pool $D_L$.

### 3.1 Design

**CL Approach**: Chaudhry et al. (2019) has shown that Experience Replay (ER) using a memory containing past samples outperforms other CL approaches. Using ER is straightforward with the help of a stored memory $M$ which, in our method, contains the set of all labelled samples at any given round, and does not have any significant overhead on train times.
**Memory Sampling**: We chose reservoir sampling (random selection) for retrieving samples from memory for replay, based on prior works that have shown that reservoir sampling outperformed other memory retrieval approaches (Chaudhry et al., 2019), and given the minimal computational overhead required in random subset selection.
**Acquisition Strategy**: Gissin & Shalev-Shwartz (2019) showed that uncertainty based selection outperformed other approaches when the query batch size was small, and all approaches performed roughly the same at larger batch sizes. We chose the straightforward Least Confidence strategy for querying.
**Learning Rate**: We reduce the number of epochs for which the model incrementally trains per round to achieve selection-time speedups. However, keeping the same schedule as for a fully-trained model would mean the model does not converge in a few epochs, leading to worse queries and thus lower accuracy overall. We hence adopt the OneCycle learning rate schedule as described in Smith & Topin (2018), which enables rapid model convergence. The model begins training every round with an initial LR, anneals gradually to a maximum LR, and then to a minimum

Table 1: A comparison between random selection, SVP and CLActive (our method) on the CIFAR-10 and CIFAR-100 datasets.

| D% | Method | Epochs | CIFAR10 | | CIFAR100 | |
|---|---|---|---|---|---|---|
| | | | Error (↓) | Speedup (↑) | Error (↓) | Speedup (↑) |
| 10 | Random | - | 20.2 ± 0.49 | - | 61 ± 0.31 | - |
| | SVP-ResNet164 | 181 | 18.7 ± 0.31 | 1× | 61.2 ± 1.09 | 1× |
| | | 50 | 19.2 ± 0.35 | 3.4× | 62.5 ± 2.49 | 3.3× |
| | SVP-ResNet20 | 181 | 18.1 ± 0.28 | 3.8× | 62.4 ± 1.07 | 4.0× |
| | | 50 | 21.6 ± 0.55 | 7.7× | 62.7 ± 1.40 | 11.5× |
| | CLActive-ResNet20 | 50 | 16.7 ± 0.06 | 9.4× | 59.4 ± 0.15 | 7.9× |
| | | 30 | 17.1 ± 0.55 | 15.1× | 60.5 ± 0.84 | 12.7× |
| | | 20 | 17.6 ± 0.19 | 21.6× | 60.1 ± 0.47 | 18.4× |
| 20 | Random | - | 12.9 ± 0.50 | - | 42.3 ± 0.61 | - |
| | SVP-ResNet164 | 181 | 10.4 ± 0.38 | 1× | 42.2 ± 0.67 | 1× |
| | | 50 | 11.7 ± 0.55 | 3.6× | 44.1 ± 0.24 | 3.6× |
| | SVP-ResNet20 | 181 | 10.5 ± 0.42 | 5.8× | 41.4 ± 0.25 | 5.8× |
| | | 50 | 13.7 ± 0.22 | 11.1× | 43.5 ± 0.58 | 18.9× |
| | CLActive-ResNet20 | 50 | 9.9 ± 0.07 | 14.6× | 42.9 ± 0.4 | 13× |
| | | 30 | 9.7 ± 0.03 | 23.8× | 42.7 ± 0.48 | 21.1× |
| | | 20 | 10.2 ± 0.07 | 34.6× | 44.2 ± 0.35 | 30.4× |
| 30 | Random | - | 10.1 ± 0.18 | - | 36.2 ± 0.36 | - |
| | SVP-ResNet164 | 181 | 7.4 ± 0.16 | 1× | 33.9 ± 0.33 | 1× |
| | | 50 | 8.4 ± 0.15 | 3.6× | 35.2 ± 0.23 | 3.6× |
| | SVP-ResNet20 | 181 | 7.4 ± 0.23 | 6.7× | 33.8 ± 0.37 | 6.6× |
| | | 50 | 10.4 ± 0.04 | 13.7× | 35.4 ± 0.23 | 22.4× |
| | CLActive-ResNet20 | 50 | 7.2 ± 0.09 | 19.4× | 34.2 ± 0.2 | 19.3× |
| | | 30 | 7.2 ± 0.07 | 31.8× | 34.7 ± 0.1 | 31.5× |
| | | 20 | 7.1 ± 0.12 | 46.7× | 34.8 ± 0.15 | 46.1× |
| 40 | Random | - | 8.6 ± 0.12 | - | 32.0 ± 0.61 | - |
| | SVP-ResNet164 | 181 | 6.1 ± 0.32 | 1× | 29.9 ± 0.18 | 1× |
| | | 50 | 7.2 ± 0.15 | 3.6× | 30.8 ± 0.47 | 3.6× |
| | SVP-ResNet20 | 181 | 5.9 ± 0.19 | 7.0× | 29.8 ± 0.10 | 7.0× |
| | | 50 | 8.3 ± 0.23 | 15.6× | 30.9 ± 0.23 | 24.3× |
| | CLActive-ResNet20 | 50 | 6.0 ± 0.18 | 31.7× | 29.9 ± 0.07 | 24.7× |
| | | 30 | 5.9 ± 0.15 | 52× | 29.9 ± 0.12 | 40.5× |
| | | 20 | 6.0 ± 0.03 | 76.4× | 30.0 ± 0.17 | 59.5× |
| 50 | Random | - | 7.5 ± 0.15 | - | 29.4 ± 0.20 | - |
| | SVP-ResNet164 | 181 | 5.3 ± 0.06 | 1× | 26.9 ± 0.21 | 1× |
| | | 50 | 5.9 ± 0.14 | 3.6× | 27.5 ± 0.43 | 3.6× |
| | SVP-ResNet20 | 181 | 5.4 ± 0.41 | 7.2× | 26.6 ± 0.14 | 7.2× |
| | | 50 | 7.1 ± 0.15 | 17.2× | 27.9 ± 0.64 | 25.1× |
| | CLActive-ResNet20 | 50 | 5.4 ± 0.06 | 36.7× | 26.4 ± 0.13 | 37.1× |
| | | 30 | 5.4 ± 0.09 | 50× | 27.0 ± 0.09 | 60.7× |
| | | 20 | 5.4 ± 0.15 | 89.1× | 27.0 ± 0.24 | 89.2× |

LR much lower than the initial LR by the end of the round. This scheduling allows us to reach a peak accuracy point in fewer epochs for our selection models.

## 4 Experiments

We demonstrate the accuracy and speedups achieved through our approach on image and textual datasets. We report the accuracies obtained by our models at 10, 20, 30, 40, and 50% of dataset collection and corresponding speedups relative to a fully-trained AL baseline. Our initial labelled subsets are chosen at random, and we use the Least Confidence query strategy for further data selection. Our code can be found at https://github.com/AuroMun/CLActive.

### 4.1 Datasets

**CIFAR-10/100**: The CIFAR-10 and CIFAR-100 datasets consist of 60,000 32x32 images in total, with a split of 50,000 and 10,000 for train and test. We start with a randomly chosen subset of 1,000 labelled samples and train our selection model on this subset to query for 4,000 new labels, and thereafter query in rounds of 5,000 samples till we

Table 2: A comparison between random selection, SVP and CLActive using a fastText selection model on the Amazon Review Polarity and Amazon Review Full datasets.

| D% | Method | Review Polarity | | Review Full | |
|---|---|---|---|---|---|
| | | Error (↓) | Speedup (↑) | Error (↓) | Speedup (↑) |
| 10 | Random | 6.5 ± 0.03 | - | 41.7 ± 0.19 | - |
| | SVP-VDCNN | 5.8 ± 0.08 | 1× | 41.9 ± 0.54 | 1× |
| | SVP-fastText | 6.9 ± 0.81 | 10.6× | 42.7 ± 0.77 | 8.7× |
| | CLActive-fastText | 6.8 ± 0.32 | 10.7× | 42.0 ± 0.34 | 9× |
| 20 | Random | 5.6 ± 0.07 | - | 39.9 ± 0.05 | - |
| | SVP-VDCNN | 4.8 ± 0.04 | 1× | 39.7 ± 0.22 | 1× |
| | SVP-fastText | 5.2 ± 0.17 | 20.6× | 39.8 ± 0.02 | 17.7× |
| | CLActive-fastText | 5.0 ± 0.04 | 20.8× | 39.1 ± 0.06 | 17.9× |
| 30 | Random | 5.2 ± 0.07 | - | 39.0 ± 0.09 | - |
| | SVP-VDCNN | 4.5 ± 0.01 | 1× | 38.6 ± 0.01 | 1× |
| | SVP-fastText | 4.6 ± 0.01 | 32.2× | 38.7 ± 0.05 | 26.7× |
| | CLActive-fastText | 4.6 ± 0.04 | 34.5× | 38.0 ± 0.07 | 29.4× |
| 40 | Random | 4.9 ± 0.01 | - | 38.4 ± 0.14 | - |
| | SVP-VDCNN | 4.3 ± 0.02 | 1× | 38.2 ± 0.03 | 1× |
| | SVP-fastText | 4.3 ± 0.01 | 41.9× | 38.1 ± 0.06 | 35.1× |
| | CLActive-fastText | 4.2 ± 0.04 | 47.3× | 37.3 ± 0.17 | 41.1× |
| 50 | Random | 4.7 ± 0.03 | - | 37.9 ± 0.01 | - |
| | SVP-VDCNN | 4.2 ± 0.02 | 1× | 37.6 ± 0.01 | 1× |
| | SVP-fastText | 4.3 ± 0.02 | 51.3× | 37.7 ± 0.05 | 43.1× |
| | CLActive-fastText | 4.1 ± 0.03 | 61.6× | 36.9 ± 0.06 | 54.7× |

reach 50% of the data collected. We use a pre-activation ResNet164 as the evaluation model in all our experiments for fair comparison, which is trained for 181 epochs following the fixed LR schedule given by Coleman et al. (2019). We use a ResNet-20 model for selection. We repeat our experiments varying the number of epochs the selection model is trained for each round, between 20, 30 and 50 epochs.

**ImageNet**: The ImageNet dataset consists of a much larger 1.28 million training images spread over 1,000 classes and 50,000 validation images. We start with a randomly chosen subset of 25,263 labelled samples and query for 102,493 new samples to reach 10% of the dataset labelled and then continue for five rounds of 128,117 (which is 10% of the dataset) samples till 50% of the dataset is labelled. We use a ResNet-50 model for evaluation throughout for fair comparison to baselines, training it for 100 epochs with the LR schedule from Coleman et al. (2019). We use a ResNet-18 model trained for 50 epochs in our experiments for selection.

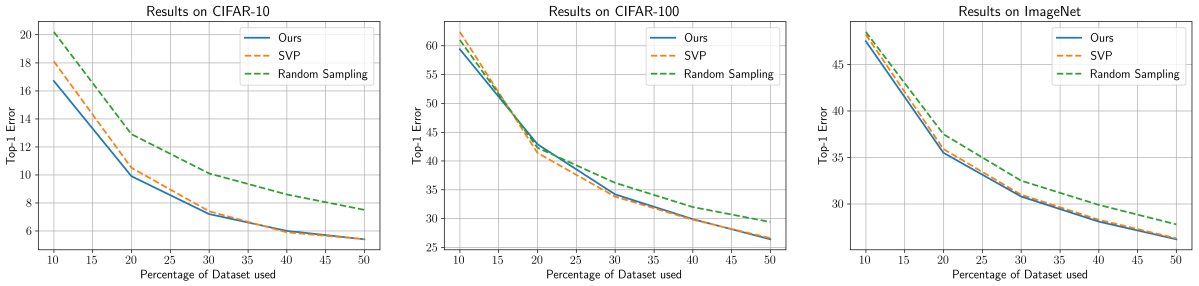
**Amazon Reviews**: For testing CLActive on the text classification domain, We use the Amazon Review Polarity and Amazon Review Full (Zhang et al., 2015; Zhang & LeCun, 2015) datasets. The Amazon Review Polarity dataset consists of 3.6 million text reviews which are either positive or negative, and 0.4 million validation reviews; and the Amazon Review Full dataset consists of 3 million text reviews across scores of 1 to 5, with 0.65 million validation reviews. The size of these datasets allows us to test our method in a high scale setting for the textual domain.

### 4.2 Experimental Details

We use a Random baseline for accuracy comparison, where a random subset of data at the total budget is queried and used for evaluation directly. Results from multiple SVP (Coleman et al., 2019) models are also shown - for CIFAR-10 and CIFAR-100, SVP-ResNet164 models are used for benchmarking. In SVP-ResNet164, ResNet-164 is used as the selection model and is initialized and trained from scratch every round on the entire collected dataset up to that point using a fixed LR schedule. SVP-ResNet20 uses a ResNet20 model for selection instead. In all SVP runs, models are re-initialized between rounds. We utilize the OneCycleLR scheduler which has an implementation provided in PyTorch, with an initial LR of 0.01, annealing to a maximum of 0.2 and then to a minimum of 0.001 within a single round. We vary the number of epochs per round for the selection model between 20, 30, and 50 for our CIFAR experiments, and 50 epochs for our ImageNet experiments. In all of our experiments, we only train on a subset of labelled data at each round. We form this by taking a small random subset from previously labelled samples and combining them with the newly queried samples, as described in section 3. We set the memory ratio $R$ to 1 for all our experiments, training our selection model on an equal number of old and new samples every round. We run all experiments thrice and report the mean and standard deviation of the resulting error. We choose the Least Confidence acquisition strategy for querying in every rounds unless explicitly specified otherwise (e.g. Random baselining).

Table 3: A comparison between random selection, SVP and CLActive on the ImageNet dataset.

| D % | Method | Epochs | Error (↓) | Speedup (↑) |
|---|---|---|---|---|
| 10 | Random | - | 48.5 ± 0.04 | - |
| | SVP-ResNet50 | 90 | 48.2 ± 0.37 | 1× |
| | SVP-ResNet18 | 90 | 48.3 ± 0.31 | 1.2× |
| | | 45 | 48.3 ± 0.31 | 2.1× |
| | CLActive-ResNet18 | 50 | 47.5 ± 0.10 | 2.4× |
| 20 | Random | - | 37.5 ± 0.34 | - |
| | SVP-ResNet50 | 90 | 35.9 ± 0.22 | 1× |
| | | 45 | 36.3 ± 0.03 | 1.8× |
| | SVP-ResNet18 | 90 | 36.1 ± 0.19 | 1.3× |
| | | 45 | 36.3 ± 0.07 | 2.5× |
| | CLActive-ResNet18 | 50 | 35.5 ± 0.20 | 3.4× |
| 30 | Random | - | 32.5 ± 0.12 | - |
| | SVP-ResNet50 | 90 | 31.0 ± 0.10 | 1× |
| | | 45 | 31.3 ± 0.02 | 1.8× |
| | SVP-ResNet18 | 90 | 31.1 ± 0.12 | 1.4× |
| | | 45 | 31.3 ± 0.02 | 2.7× |
| | CLActive-ResNet18 | 50 | 30.8 ± 0.10 | 3.8× |
| 40 | Random | - | 29.9 ± 0.42 | - |
| | SVP-ResNet50 | 90 | 28.3 ± 0.32 | 1× |
| | | 45 | 28.3 ± 0.19 | 1.8× |
| | SVP-ResNet18 | 90 | 28.2 ± 0.13 | 1.5× |
| | | 45 | 28.4 ± 0.17 | 2.9× |
| | CLActive-ResNet18 | 50 | 28.1 ± 0.20 | 5.5× |
| 50 | Random | - | 27.8 ± 0.13 | - |
| | SVP-ResNet50 | 90 | 26.3 ± 0.16 | 1× |
| | | 45 | 26.5 ± 0.17 | 1.7× |
| | SVP-ResNet18 | 90 | 26.4 ± 0.02 | 1.6× |
| | | 45 | 26.6 ± 0.08 | 3.1× |
| | CLActive-ResNet18 | 50 | 26.2 ± 0.06 | 7.2× |



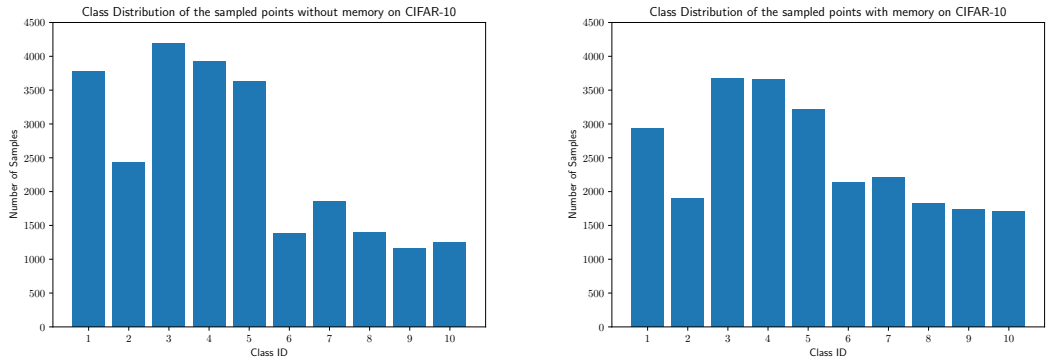

(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet

Figure 1: Comparison of accuracies for CLActive, SVP as well as random sampling

For text classification, We use the fastText (Joulin et al., 2016) selection model in our experiments and do not use an incrementally updated model - we simply train a new fastText model every round, but only on the latest queried batch of samples. We do not make any LR modifications to the fastText models either. The results are noted in table 2.

### 4.3 Results

**CIFAR-10/100**: We compare accuracies achieved after every 10% interval in the dataset from 10% to 50% and present the information in table 1. SVP-ResNet164 denotes that the model used for training was a ResNet164 model. At each point of the dataset collected, from 10% to 50%, we show the error obtained by a ResNet-164 evaluation model trained on 181 epochs given the samples chosen by various methods. The baseline model is denoted by SVP-ResNet164 trained with 181 epochs per round. We include results with the SVP-ResNet164 selection model trained on 50 epochs and the SVP-ResNet-20 selection model trained on 181 and 50 epochs. We cite these results directly from Coleman et al. (2019). SVP models trained with fewer epochs (SVP-ResNet164 at 50 epochs, SVP-ResNet20




(a) Without Memory

(b) With Memory

Figure 2: Results for training with and without memory on CIFAR-10 on our special setup. Memory ensures that the classes are sampled more uniformly leading to better accuracies on the test set.

at 50 epochs) appear to suffer from significant accuracy degradation. At 50%, SVP-ResNet164-50epochs and SVP-ResNet20-50 epochs have increased average error of 0.6% and 1.8% respectively on CIFAR-10. This is unsurprising, as the selection models in the SVP pipeline are trained from scratch every round - and training a new model for a few epochs would not allow it to converge, affecting the quality of samples. In contrast, samples collected using CLActive seem to evaluate with little or no loss from the baseline models at each point of dataset collection on both CIFAR-10 and CIFAR-100 datasets. We achieve far higher overall speedups even in comparison to SVP-ResNet20-50epoch models. At 50%, we gain an 89.2× speedup training a ResNet-20 selection model for 20 epochs per round, compared to SVP-ResNet20-50epoch's speedup 25.1× (which also came at an accuracy cost). We also find that our models trained on 20 epochs and 30 epochs do not perform significantly worse than using 50 epochs every round in the CIFAR-10 experiments; our 20-epochs models show 0.3 increased average error at 20% of dataset collection for CIFAR-10 and no increased error at later rounds. However, our 30-epoch and 20-epoch models show a slightly higher error increase compared to our 50-epoch models on CIFAR-100 - around 0.6 higher at both the 30% and 50% points.

**ImageNet**: Our ImageNet results are shown in table 3. Here, SVP-ResNet50 on 90 epochs denote the baseline, and we also include SVP-ResNet50 at 45 epochs and SVP-ResNet18 on 90 and 45 epochs for comparison. As above, we take these results from Coleman et al. (2019). As seen with the CIFAR experiments, we observe that SVP-ResNet50 and SVP-ResNet18 at 45 epochs both perform slightly worse than the baseline at various points, e.g. with 0.4 higher average error at 20% and 0.3 higher average error at 30%. In contrast, our models perform consistently without any degradation at all throughout. At 50% of the dataset collected, our setup trains over 7× faster than the baseline and more than twice as fast as SVP-ResNet18-45 epochs.

**Amazon Reviews**: On the Amazon Reviews dataset, we find that simply training fastText models on new batches of queried samples performs equivalently to the full baselines on both the Review Polarity and Full datasets at a large scale. Since fastText models are extremely fast themselves, our modification brings around a 1.3× speedup over simply using a fastText proxy. As noted in table 2, this brings our method to a significant 61.6× and 54.7× speedup over the baseline VDCNN model. This demonstrates that our method extends well across domains.

#### 4.3.1 SPEEDUPS

There are three improvements to classical AL techniques that provide speedups:

- Usage of proxy models as in Coleman et al. (2019)
- Large reduction in the number of epochs trained per iteration
- Reducing the number of samples used for training in each iteration

The speedup gained due to each of these improvements can be observed from tables 1 and 3. For CIFAR, the SVP-ResNet20-181epochs run shows the speedup on using a proxy model alone; SVP-ResNet164-50epochs shows the speedup on training for fewer epochs alone, and CLActive-ResNet20 models show the combined speedups from using a proxy model, training for fewer epochs and also using a reduced number of samples for selection model training.

Table 4: Comparison of error rates at 50% of CIFAR-10 labelled in a class-constrained scenario between a model using memory versus without.

| **Method** | **Error at 50%** |
|---|---|
| Random | 7.5 ± 0.15 |
| Memory-OFF | 7.0 ± 0.21 |
| Memory-ON | 6.2 ± 0.17 |

### 4.4 Active Learning with Adding New Classes

**Motivation:** Real-life scenarios where AL is beneficial typically involve large amounts of unlabelled data. Past DAL approaches assume that all unlabeled data is available from the start; however, these approaches do not scale well with dataset size. A model in the classical AL setup trains on the entire set of labelled samples at each round, and as the labelled dataset grows in size, the time taken to train the model on the updated pool increases. However, incrementally updating a continuous model can make it prone to *catastrophic forgetting* when the data stream is non-i.i.d - in the presence of new class data, the model may "forget" how to classify earlier classes, affecting the quality of queried samples.

**Setup:** We modify our AL setup to demonstrate this: after the initial subset is chosen, half the classes are locked till 30% of the dataset is collected. The training model can query the entire dataset at that point and runs another round of training and querying 10% of the dataset to gain 50% dataset labelling overall. In this modified setup, we run two variations of the algorithm: one where the model updates only on the freshly labelled samples each round (denoted as Memory-OFF), and one where the model trains on a mixture of half the samples from memory and half the samples from the freshly labelled set (denoted as Memory-ON).

**Results:** We note our results in Table 4. We find that Memory-OFF suffers an accuracy hit and barely performs better than random sampling, whereas the Memory-ON model performs well above random. This is unsurprising, as the Memory-OFF model was prone to suffer from forgetting in later rounds where all classes were unlocked, as opposed to the Memory-ON model, which trained on a subset of memory at every point. Additionally, we also plot the distributions of queried samples in figure 2: here, we can see that the Memory-OFF model (a) has a very uneven distribution of samples in the first half of classes [1-5] compared to classes [6-10], and the Memory-ON model (b) has a lesser class imbalance in comparison. This can be explained by the fact that the selection model at 40% of dataset collection in the Memory-OFF scenario has only trained on classes belonging to [6-10] and thus queried for samples exclusively from the [1-5] classes, increasing overall imbalance. The Memory-ON model at 40% dataset collection does not face this to the same extent, as it trains on a mixture of all classes.

## 5 Conclusion

We demonstrated a set of improvements to the standard DAL pipeline that work in conjunction with proxy modelling and enabled us to achieve another nearly two orders of magnitude of speedup overall. While the computation costs and selection time of training a selection model repeatedly in the classical setup grow linearly as rounds progress and more of the dataset is labelled, our setup maintains a constant train time and computation cost. On CIFAR-10 and CIFAR-100, CLActive achieved up to $89\times$ overall speedup while still performing significantly above random and showing little degradation in accuracy from baseline. We achieved over $7\times$ speedups on the much larger ImageNet dataset, again with no degradation in accuracy. We showed that this extends beyond image classification by testing models on the Amazon Review datasets, where we learn only from new samples, achieving $61\times$ and $54.7\times$ speedups on the Review Polarity and Review Full datasets, respectively. We found that while simply updating the model with new samples was prone to catastrophic forgetting and thus suffered degradation in accuracy, CLActive alleviates this by the addition of replaying episodic memories. Additionally, we also explored the removal of the "closed-world" assumption that DAL setups typically carried and explored the effects of seeing unseen classes of a dataset in later rounds. In summary, our proposed setup, CLActive, (i) maintains constant training costs despite growing dataset sizes; (ii) is much faster than models trained in the classical setup; and (iii) is robust towards the presence of unseen classes, making DAL feasible and cheaper to run in real-world scenarios.

## References

Ali Ayub and Alan R Wagner. Learning novel objects continually through curiosity. *arXiv preprint arXiv:2103.07758*, 2021.

Eden Belouadah, Adrian Popescu, Umang Aggarwal, and Léo Saci. Active class incremental learning for imbalanced datasets. In *European Conference on Computer Vision*, pp. 146–162. Springer, 2020.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning, 2019.

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

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. *arXiv preprint arXiv:2009.00919*, 2020.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

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

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pp. 1183–1192. PMLR, 2017.

Daniel Gissin and Shai Shalev-Shwartz. Discriminative active learning, 2019.

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pp. 148–156. Elsevier, 1994.

David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pp. 3–12. Springer, 1994.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2017.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pp. 524–540. Springer, 2020.

Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1320–1328, 2017.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A survey of deep active learning, 2020.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Burr Settles. Active learning literature survey. *Technical Report*, 2009.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2018.

Ju Xu and Zhanxing Zhu. Reinforced continual learning. *arXiv preprint arXiv:1805.12369*, 2018.

Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657, 2015.