Liquid Ensemble Selection for Continual Learning

Carter Blair
University of Waterloo
Canada
carter.blair@uwaterloo.ca

Ben Armstrong
University of Waterloo
Canada
ben.armstrong@uwaterloo.ca

Kate Larson
University of Waterloo
Canada
kate.larson@uwaterloo.ca

ABSTRACT

Continual learning aims to enable machine learning models to continually learn from a shifting data distribution without forgetting what has already been learned. Such shifting distributions can be broken into disjoint subsets of related examples; by training each member of an ensemble on a different subset it is possible for the ensemble as a whole to achieve much higher accuracy with less forgetting than a naive model. We address the problem of selecting which models within an ensemble should learn on any given data, and which should predict. By drawing on work from delegative voting we develop an algorithm for using delegation to dynamically select which models in an ensemble are active. We explore a variety of delegation methods and performance metrics, ultimately finding that delegation is able to provide a significant performance boost over naive learning in the face of distribution shifts.

KEYWORDS

Continual Learning, Liquid Democracy, Delegation, Social Choice

1 INTRODUCTION

Machine learning models often operate under the assumption that the data they are trained on and the data on which they will make predictions are independently and identically distributed (i.i.d.). However, this assumption does not reflect the reality of the world we live in. In practical settings, data distributions are constantly shifting. Deployed machine learning models that operate as though the world is static will eventually become highly inaccurate or will begin to enforce outdated knowledge about the world – often resulting in harmful bias. Instead, models must adapt to the world as it changes.

Continual learning is a machine learning framework in which models learn incrementally on training data that arrives in an ordered stream and from a shifting distribution. Naively learning on such data allows models to remain performant on recent data but leads to a problem known as *catastrophic forgetting* wherein previously learned information is forgotten [7, 21]. Similarly, test data is often modelled as coming from a similar non-stationary stream where changes in distribution typically harm model performance. Work in continual learning develops more reactive learning algorithms in order to address problems caused by distribution shifts [21].

Ensembles have long been used in many machine learning settings. In standard learning situations, ensembles typically outperform single classifiers and demonstrate a strong ability to leverage diverse models for a common purpose. Conceptually, ensembles are well-suited to data shifts: By using different classifiers to learn on different underlying distributions the ensemble can dynamically select classifiers to rely upon when making predictions and avoid forcing a single model to forget previous knowledge in order to stay up-to-date. Existing work has already found that ensembles can be applied to a continual learning setting [1, 6, 13, 20, 22, 24].

We argue that the recent use of delegative voting for ensemble pruning lays a foundation for a new continual learning algorithm. This delegative voting framework, known as *liquid democracy*, allows ensemble members to participate directly in training and inference, or to transitively delegate their action to another classifier.

While many continual learning algorithms rely upon explicit knowledge about changes in the distribution [1, 20], or remember useful training examples so they may be re-learned [4, 16, 19], our approach relies only upon knowledge about the recent learning performance of each classifier. Using this information, the algorithm makes delegation decisions which further two goals: (1) selecting the classifiers most well suited to learning/predicting under the current distribution and (2) weighting the chosen classifiers appropriately in order to optimize their prediction accuracy.

Our algorithm is highly generic; we explore a variety of delegation methods drawn from existing research on liquid democracy and several measures of classifier performance. Rather than relying solely on accuracy, we are able to consider any single-valued measure such as precision, recall or f1-score. In experiments across multiple domains of continual learning, we demonstrate the utility of adapting ideas from liquid democracy into the continual learning setting. In particular, our contributions are as follows:

- A novel framework for continual learning and dynamic ensemble selection based on principles of liquid democracy.
- New definitions of classifier performance in terms of learning rate and recent performance, tailored to the specific demands of continual learning.
- Empirical validation of our approach via experiments on classic continual learning benchmarks, demonstrating a strong ability to enhance ensemble performance.

Through these contributions, our work establishes a new direction for research in continual learning and dynamic ensemble selection. It highlights the potential of integrating concepts from delegative voting into machine learning to create more adaptable and diverse ensembles.

2 RELATED WORK

2.1 Ensembles in Social Choice

Ensembles rely on aggregation methods to combine outputs from many classifiers into a single prediction. As such, a great deal of work spanning multiple decades has considered the application of social choice results to ensemble learning. This has been illustrated through theoretical connections between ensemble learning and

Appears at the 1st Workshop on Social Choice and Learning Algorithms (SCaLA 2024) held at the 23rd International Conference on Autonomous Agents and Multiagent Systems., Armstrong, Fairstein, Mattei, Terzopoulou, May 6-7, 2024, Auckland, New Zealand. 2024.

well-known axiomatic results in social choice [18], as well as by experimental evaluation of many common voting rules for classifier prediction aggregation [5, 11].

Only one line of work has considered the application of liquid democracy to ensembles. In a setting with exactly one round of delegation, liquid democracy has been shown to not provide any significant benefit over traditional ensembles [2]. However, delegation can also be viewed as a form of ensemble pruning and reweighting. Using iterative delegations in an incremental learning setting, Armstrong and Larson showed that liquid democracy can typically achieve higher accuracy than a full ensemble while also dramatically reducing the cost of training the ensemble [3]. On some datasets, their method attained higher accuracy than the well-known ensemble algorithm Adaboost [8]. However, this result was highly data-dependant. While the incremental learning setting of Armstrong and Larson (2024) can be considered a continual learning setting, there appears to be no work applying social choice to continual learning in a setting with distributional shift.

2.2 Ensembles in Continual Learning

Ensembles are a useful method to improve performance in continual learning [1, 6, 13, 20, 22, 24]. On one extreme end of the spectrum, some ensemble methods initialize a new member for each new context they encounter [1]. However, these methods have high resource usage and require context labels in training, limiting their applicability. Other methods, such as SEED [20] and CoSCL [22], use a fixed number of models in the ensemble to avoid a linear increase in parameters with the number of contexts. These methods also require the context label to be known during training.

In contrast to other methods, we focus on keeping the architecture simple and general. We do not use a generative model, replay buffer or leverage context labels in training, and we use a fixed number of models in our ensemble to limit parameter growth. Not relying on context labels in training allows our method to work in a more diverse range of settings, including both class-incremental and domain-incremental learning settings.

3 MODEL

Our model bridges two fields of study – machine learning, and social choice. We consider an epistemic social choice setting where voters are instantiated as an ensemble of machine learning classifiers. Voters (classifiers) may vote for one of several alternatives in pursuit of a singular "correct" alternative and this corresponds to the classification action of a classifier.

We focus on a setting where data has not only features and labels, but also a context which is associated with some data distribution. Within this setting, we explore algorithms for dynamically re-weighting voters/classifiers using delegative techniques. That is, classifier weights are controlled by "delegation" between classifiers, and only non-delegating classifiers take part in inference or learning. Classifiers may participate actively (they learn/predict on examples directly) or may delegate to another classifier, not thereby increasing that classifiers prediction weight and not learning or predicting.

In more detail, we consider an ensemble of classifiers $V = \{v_1, v_2, ..., v_n\}$, which we equivalently refer to as voters, and a non-stationary data stream $\mathcal{D}$ containing m classes. Data arrives in batches of B samples over T steps, with each batch associated with some (possibly repeating) context $c_t \in C$; $\mathcal{D} = \{(\mathcal{X}_1, \mathcal{Y}_1, c_1), ..., (\mathcal{X}_\mathcal{T}, \mathcal{Y}_\mathcal{T}, c_T)\}$. Each example in a dataset can be decomposed into three pieces: features $\mathcal{X}$, a class label $\mathcal{Y}$, and a context label C. Together, the class label and context label make up a *global label* $\mathcal{G}$. Depending upon the setting, this stream may include training data, testing data, or training data *and* testing data. Figure 1 gives an example of the difference between a class label and a context label.

A delegation function, $d : V \times T \rightarrow V$, determines which classifiers are learning or doing inference on any given batch. A delegation function value $d(v_i, t) = v_j$ indicates that, during batch t, v_i delegates to v_j. In most cases, the batch is clear from context, and we write $d(v_i) = v_j$ to indicate a delegation in the current batch. By default, classifiers delegate to themselves. Classifiers that delegate to anyone other than themselves are referred to as "delegators" and are *inactive* while a classifier that delegates to itself is *active* and may be referred to as a "guru." The set of all gurus is found through repeated application of the delegation function until a self-delegation is reached: $G_t(V) = \{v \in V | d^*(v, t) = v\}$. Only classifiers that are gurus participate in learning or inference, depending on the setting.

The weight of each classifier is also controlled by delegation. Specifically, a weight vector W_t contains the weight associated with each classifier on batch t. If no classifiers delegate (that is, if they all delegate to themselves), then each classifier has a weight of 1. Generally, each classifier has a weight equal to the number of delegations they receive, including their own:

$$W_t[i] = \begin{cases} 0 & \text{if } v_i \notin G_t(V) \\ |\{v_j \in V | d^*(v_j, t) = v_i\}| & \text{otherwise} \end{cases}$$

An estimate of the performance of each classifier is continually refined. Performance can be measured as prediction accuracy or any other single-valued metric that reflects the performance of a classifier on a batch of data. A = $[a_{t,i}]$ stores the performance of each classifier v_i on each batch t. We are often interested in the trend of a classifier's performance as it learns. That is, over recent batches, has it become more or less accurate? We use this to approximate two measures: (1) whether or not the classifier is still improving its performance/benefits from continued training, and (2) as a proxy for whether a context shift has happened. If a classifier's performance drops rapidly, this is an indication that the underlying data has shifted distributions. To evaluate this performance trend, we calculate the slope of the linear regression line of the performance values of each classifier over some parameter w of recent batches. We denote the performance slope of v_i as $q_{i,w}$ and all such slopes as Q. The set of all classifiers with a higher slope than v_i is $N^+(v_i) = \{v_j \in V | q_{j,w} > q_{i,w}\}$. Intuitively, this corresponds to the classifiers that are "better" than v_i; those showing more recent improvement in accuracy. Typically, we refer to the current weight and estimated performance of classifier v_i as w_i and q_i, respectively, only including a time subscript where necessary.

Algorithm 1 Continually Learning Liquid Democracy Ensemble

Input: Batch size B, number of classifiers n, delegation mechanism $DelMech$
Initialize all classifiers $\{v_i\}_{i=1}^{n}$ and all delegations to $d(v_i) = v_i$
for t = 1, ..., T **do**
  Receive batch $\{(x_j, y_j)\}_{j=1}^{B}$ from data stream
  for $v_i \in V$ **do**
    v_i makes predictions $\{\hat{y}_{i,j}\}_{j=1}^{B}$ on current batch
    if $d(v_i) = v_i$ (classifier is guru) **then**
      v_i learns on batch
    end if
  end for
  Update delegations: $DelMech(V)$
end for

To generate a prediction, each guru in the ensemble generates class probabilities for each possible class. These are given a weight equal to the guru's weight and summed to get a weighted probability for each class. The class with the highest weighted probability is the prediction of the ensemble. This procedure is based on that of the SEED method from Rypeść et al. [20].

3.1 Continual Learning

In continual learning, a learner attempts to learn from a non-stationary stream of data arriving in batches, each associated with some context. A context may appear in multiple contiguous batches, but, importantly, after a context stops arriving, it does not return during the training phase. This can lead to what is commonly referred to as "catastrophic forgetting," which is characterized by the learner forgetting how to classify samples from contexts learned early in training after further learning in other contexts has occurred.

We explore two of the standard scenarios within continual learning, differentiated based upon the information available to the learner and the label space [21, 23]:

(1) In *class-incremental learning*, contexts have disjoint class labels. Classifiers only know the current context during training, not testing. The classifier must learn to predict both the label and context given only the input. Formally, the goal is to learn a mapping $f : \mathcal{X} \to \mathcal{Y} \times \mathcal{C}$.
(2) In *domain-incremental learning*, each context has the same class labels, but they may appear at different frequencies in different contexts. The goal is to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$.

Figure 1 gives an example of the learning target of both learning methods. Algorithm 1 outlines the general behaviour of our continual learning training algorithms: Voters all start as gurus, then, for each batch of data, the ensemble receives the batch, has all gurus learn the batch, and updates delegations based on training performance. Note that in all settings, we assume the training data is ordered or non-stationary, but there are no assumptions about the test set. We can construct the test set as useful for evaluation. For example, it is common to report test accuracy over all contexts and within context. We may assume that the test set is non-stationary in some cases.

4 DELEGATION MECHANISMS

We develop a delegation mechanism suitable for continual learning that aims to be adaptive to context shifts while incorporating existing delegation methods as a parameter. Our mechanism, k-Best-Accuracy-Trend (k-BAT), does not depend on knowledge about the current context or store previous examples. Rather, the mechanism uses a weak signal about the recent performance of each classifier in the ensemble to determine whether/where each classifier should delegate.

4.1 k-Best-Accuracy-Trend Delegations

Our algorithm is described in Algorithm 2 and here in words. In continual learning, data is organized into contexts which are further subdivided into batches. k-BAT works over batches, using two parameters to record performance and make decisions: A `metric` parameter evaluates the performance of classifiers and a `delegation probability function` determines where classifiers will delegate.

During learning, k-BAT is called once per batch after the active voters in the ensemble have learned on the batch. When it begins, k-BAT calculates and records the prediction performance of all classifiers on the recent batch according to the given `metric` (this can be any signal about performance such as accuracy, f1-score, etc.). Subsequently, for each classifier, linear regression is performed over the most recent w batches and the slope of the result is recorded (a positive slope indicates that a classifier is generally improving, while a negative slope indicates the reverse).

After recording the performance, delegation begins. The k classifiers with the best performance trend are chosen to act as gurus. Each remaining classifier chooses a single other classifier to whom they delegate. Each delegation is made according to the given `delegation probability function`, which may use accuracy, weight, or other information to determine the probability that each classifier will delegate to each other classifier.

Note that k-BAT only operates during *training*; during test time, all classifiers in the ensemble are active. The idea is that when making predictions, classifiers trained on a specific context are more confident and accurate when predicting classes from that context, while those not trained on that context are less confident and spread their probability mass more evenly between the classes. This means the accurate, confident classifiers mainly determine the final prediction on classes from the context they were trained on.

Below, we describe the individual delegation probability functions that we explore.

4.2 Delegation Probability Functions

Definition 4.1. The **Random Better** Delegation Probability Function assigns delegation probabilities such that each classifier with a higher regression slope than the delegator is selected with equal probability.

$$p^{\text{rand\_better}}(v_i, v_j) = \begin{cases} \frac{1}{|N^+(v_i)|} & j \in N^+(v_i) \\ 0 & \text{otherwise} \end{cases}$$

Definition 4.2. The **Proportional Better** Delegation Probability Function assigns delegation probabilities such that each classifier with a higher regression slope than the delegator is assigned a

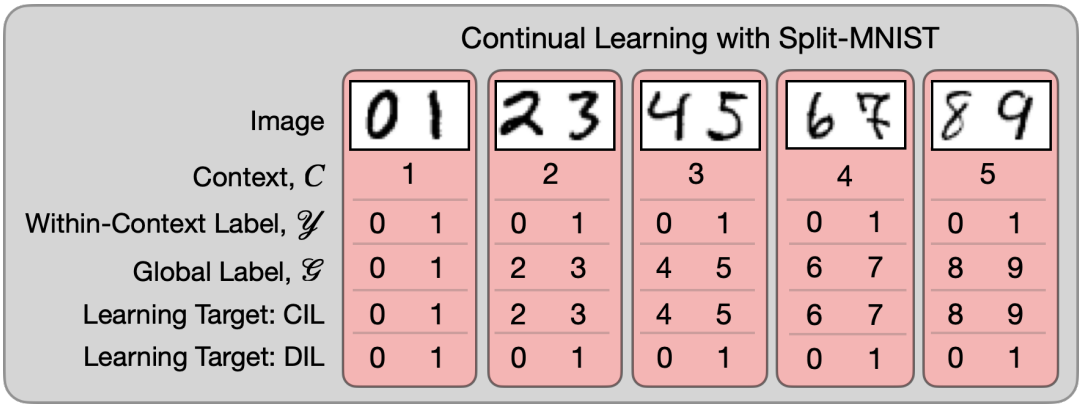

Figure 1: Examples of label types and learning targets within continual learning. Classes are grouped into disjoint contexts which are concatenated to form a data stream. Class Incremental learning learns the global label – a unique combination of context label and within-context label. Domain Incremental learning learns only the label within the context. (Note: This figure is inspired by a similar figure from van de Ven et al. [21].)

Algorithm 2 Best Accuracy Trend Delegation Mechanism

Input: V, w, `metric`, `delegation_probability_function`
$P \leftarrow n \times n$ matrix initialized to 0
for $v_i \in V$ **do**
  $q_i \leftarrow$ linear regression slope of `metric` over last w batches.
end for
for $v_i \in V$ **do**
  for each $v_j \in V$ **where** $i \neq j$ **do**
    if $q_i > q_j$ and $d(v_i) = v_j$ **then**
      $d(v_i) := v_i$
    else
      $P_{i,j} :=$ `delegation_probability_function`(v_i, v_j)
    end if
  end for
  Select a v_j according to probabilities $P_{i,-}$
  $d(v_i) := v_j$
end for

probability proportional to their regression slope value. Shown here are delegation probabilities before normalization.

$$p^{\text{prop\_better}}(v_i, v_j) \propto \begin{cases} \frac{q_j - q_i}{\sum_{v_k \in N^+(v_i)} q_k - q_i} & j \in N^+(v_i) \\ 0 & \text{otherwise} \end{cases}$$

Definition 4.3. The **Proportional Weighted** Delegation Probability Function assigns delegation probabilities such that each classifier with a higher regression slope than the delegator is assigned a probability proportional to both their regression slope value and the weight of the delegate's guru. A lower weight leads to a higher delegation probability. Shown here are delegation probabilities before normalization.

$$p^{\text{prop\_weighted}}(v_i, v_j) \propto \begin{cases} \frac{1}{w_{d^*(j)}} \frac{q_j - q_i}{\sum_{v_k \in N^+(v_i)} q_k - q_i} & j \in N^+(v_i) \\ 0 & \text{otherwise} \end{cases}$$

Definition 4.4. The **Max Diversity** Delegation Probability Function assigns delegation probabilities such that each delegating classifier delegates to the most similar classifier to themselves that has a higher regression slope. Many diversity metrics exist in the ensemble literature [10]; however, in this work, we calculate diversity in a pairwise manner by determining the class prediction probabilities for each classifier (a function easily available for most machine learning models). The "diversity" value of any pair of classifiers is the $l2$ norm of the difference between their class probabilities.

$$p^{\text{diverse}}(v_i, v_j) = \begin{cases} 1 & v_j = \arg\min_{v_j \in N^+(v_i)} \text{diversity}(v_i, v_j) \\ 0 & \text{otherwise} \end{cases}$$

4.3 Student-Expert Delegations

The Student-Expert extends delegation to test time by running delegation separately on two sets of classifiers: one set which learns (students) and another which is used for inference (experts). Typically, in continual learning, evaluation is done after training on each context individually or with the assumption that the test stream is drawn i.i.d from all contexts. This can be useful for assessing the degree of forgetting, but in reality, the test stream is likely to be non-stationary, similar to the training stream. We can leverage the non-stationary nature of the test stream with a method that dynamically shifts which members of the ensemble are making predictions.

The Student-Expert delegation mechanism does just this by adding delegation to the test phase. We use delegation to dynamically shift which members of the ensemble are making predictions. The Student-Expert delegation mechanism can be seen as an extension of the k-Best-Accuracy-Trend (k-BAT) delegation mechanism described in subsection 4.1. It maintains k_s "student" classifiers and k_e "expert" classifiers. $k - BAT$ is run normally for each batch of data to pick k_s students who learn on that batch. Separately, $k-BAT$ is run to pick k_e experts, which will make predictions on the batch. Currently, we use only the Proportional Better delegation probability function to select gurus. Students measure performance using a

parameterized window size of w and a given performance metric as in k-BAT, while experts measure performance by comparing classifier prediction accuracy on only the most recent batch.

5 EXPERIMENTS

We evaluate our algorithms along several axes – number of gurus, best performance metric, and delegation mechanism. To do this, we run three experiments: First, a simple experiment demonstrates in detail the behaviour of an ensemble using k-BAT within a class-incremental setting. Second, a more in-depth experiment explores the behaviour of a variety of parameterizations of our algorithms for class-incremental learning. Finally, a similar experiment explores the same algorithms applied to domain-incremental learning. Our results show strong performance across both settings and surprising differences in optimal parameters for class- and domain-incremental learning.

5.1 Experimental Setup

Data: As is common in continual learning work, we use two MNIST datasets in our experiments: Split MNIST and Rotated MNIST [21]. Split MNIST consists of grey-scale images of digits from 0 to 9, with all images scaled to the same dimensions [25]. The feature values are pixels, and the class is the digit value. We use Split MNIST as a stream of data partitioned into multiple contexts, each containing a mutually exclusive subset of the classes. Specifically, we partition the stream into five disjoint contexts, each containing two adjacent digits (i.e. {(0, 1), (2, 3), ..., (8, 9)}. We exclusively use Split MNIST in the class-incremental setting where we are interested in learning the global label. On this data, the global label is simply the digit value (0-9).

In order to test our methods in the domain-incremental learning setting, we use a similar but slightly more complex data set: Rotated MNIST. This data again consists of images of digits, as in Split MNIST. However, the digits are now rotated around the center of the image from anywhere between 180° and -180°. We initialize Rotated MNIST into a stream consisting of 5 disjoint contexts. Each context corresponds to a specific rotation amount and contains only images corresponding to that rotation. The context label is the rotation amount, and the within-context label is the digit value (0-9). Each context's rotation is sampled uniformly at random from between 180° and -180°.

Parameter Values: We explore a range of parameters in both learning settings. All four delegation probability functions for k-BAT are tested for $k \in \{1, 2, 3, 4\}$. We test each delegation probability function and report the best one. Results for other probability delegation functions are shown in the appendix. Additionally, we test three classifier competency metrics for k-BAT: accuracy, balanced accuracy, and F1 score. In addition to this, we use between 1 and 4 experts in Student-Expert delegation. Experiments all use ensembles of size 8 except for the larger experiment on class-incremental learning, which explored an ensemble of size 30 with up to 11 gurus, and the smaller illustrative experiment, which used an ensemble size of 5.

Our models are all simple feed-forward neural networks with two hidden layers which use the Adam optimizer, implemented using Pytorch [17]. In line with previous work that compares ensemble methods with multiple learners to methods which only use a

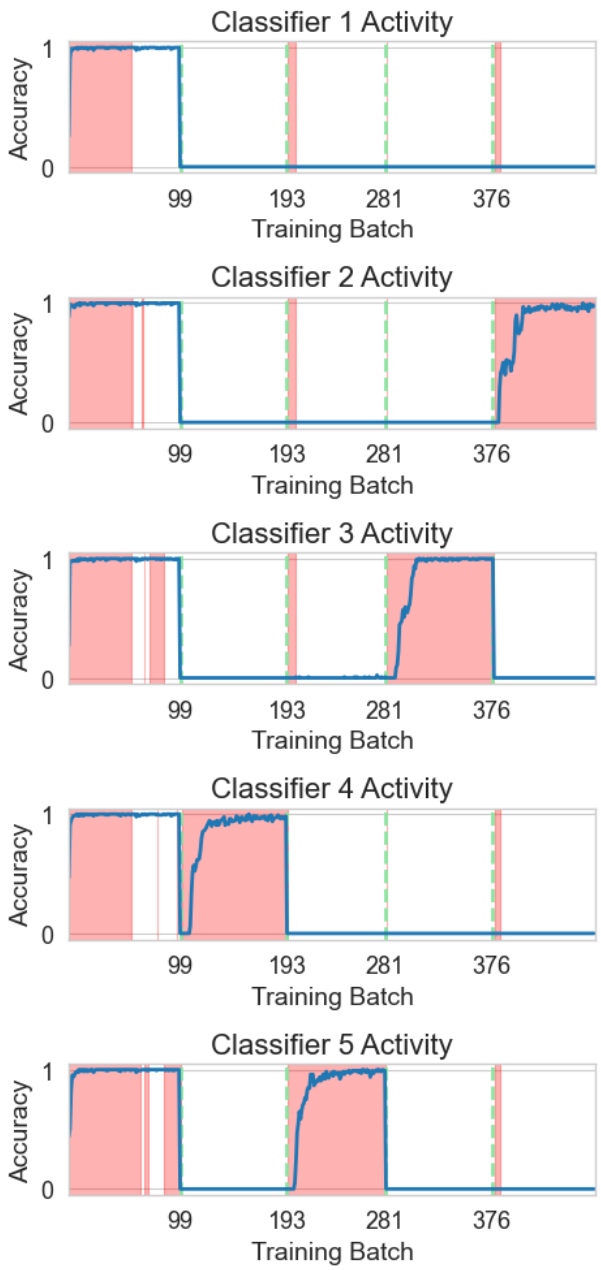

Figure 2: Active learning periods of each classifier on Split MNIST using k-BAT with $k = 1$ and probabilistic better delegation. Red periods indicate a classifier is actively learning. With a window size of 50, all classifiers learn on the first 50 batches. Subsequently, delegation begins, and one classifier learns at a time. At context shifts (green lines) various classifiers briefly begin to learn but k-BAT effectively picks a single classifier to learn the majority of a context without any knowledge of the context label.

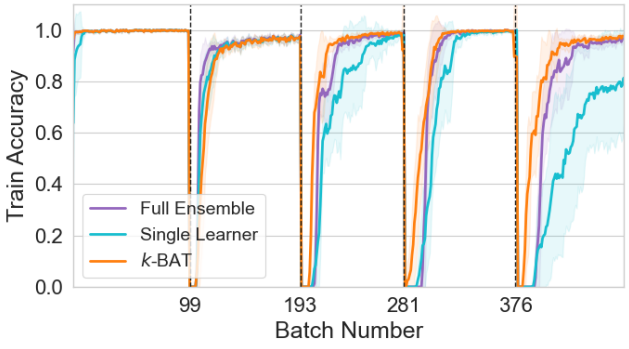

Figure 3: Training accuracy on MNIST of k-BAT (k = 1) using probabilistic delegation, a full ensemble, and a single classifier. Results are averaged over 10 trials across 5 disjoint contexts. In early contexts, all models learn at similar rates and achieve similar performance. Later, k-BAT continues to learn new contexts quickly while the learning rate of both other models drops.

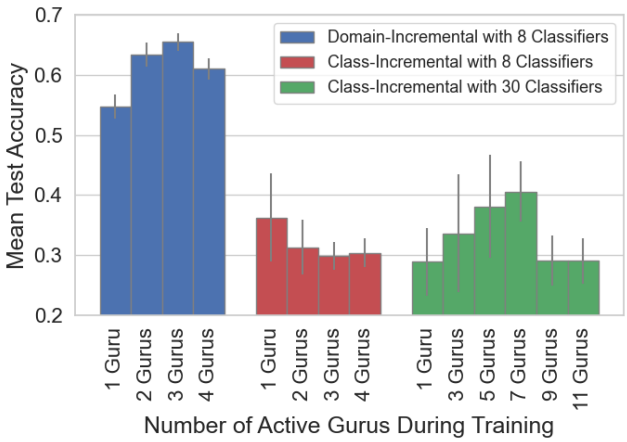

Figure 4: Test accuracy according to value of k for k-BAT on domain-incremental learning and two sizes of ensemble doing class-incremental learning. Results are averaged over the probability delegation functions and delegation metrics. While the small-scale experiment on class-incremental learning performs best with only 1 guru, the other settings are optimized with larger numbers of gurus.

single learner, we keep the total number of trainable parameters for each continual learning method roughly equal [22]. That is, when comparing the performance of an ensemble of 8 classifiers to a single network, each network in the ensemble has roughly one-eighth the number of trainable parameters as the solitary network.

In the class-incremental setting, k-BAT and the Student-Expert method use a window size, w, of 50 batches. In the domain-incremental setting, these methods use a window size of 400 batches. In all cases, we use a batch size of 128.

Benchmarks: We compare our algorithms with three existing approaches to continual learning. A significant component of our algorithms is that they do not explicitly remember previously seen examples (as in Replay-based continual learning) or get told context identities. As such, the methods we compare against also do not use Replay or context identities. Each method we use is found in the Avalanche Continual Learning library [14]. Conceptually each method is similar, however, their implementation details differ:

Elastic Weight Consolidation [9]: EWC slows down updates on network weights that are more important to previously learned examples.

Learning without Forgetting [12]: LwF is similar to EWC but evaluates itself before and after learning new data to try to maintain predictions on previous data.

Synaptic Intelligence [25]: Synaptic Intelligence uses biologically motivated techniques to record the importance of each network weight to prediction. Weights important to particular tasks are changed by smaller amounts.s

5.2 Results

In our first experiment, we construct a slightly simplified scenario using the split MNIST data set to illustrate how the k-BAT method works. To do so, we plot the individual learning curves of each classifier as well as the periods in which they were active on the training set. Figure 2 shows that, roughly speaking, the classifiers divide and conquer, with roughly one classifier learning each context. This is non-trivial since we do not know the index at which the context changes or use the context label to inform which classifier is learning. This is achieved by using the slope of the learning curve over the last w batches. Classifiers that have learned on fewer samples are usually more plastic [15] and, thus, have a steeper learning curve, making them more likely to receive a delegation. Then, when the context switches, the accuracy of a classifier that was learning the previous context will significantly drop, and the slope of the learning curve will eventually become negative. With a negative slope, this classifier will likely delegate to another classifier. Further, since all classifiers that were not learning in the previous context will likely not experience a large drop in accuracy, the slope of their learning curves will be, on average, zero, and thus, they will rescind their delegation to the classifier that was learning in the previous context.

The initial period where all classifiers are active can be explained by the fact that the classifiers do not delegate until their window is full, so by design, all classifiers will be active on the first w batches. Since not all classifiers are learning in each context, the classifiers specialize in a reduced number of contexts and thus, the catastrophic forgetting is greatly reduced.

In addition to better performance on the test set, Figure 3 shows that in later contexts, the k-BAT Delegation ensemble learns more quickly, which is likely because the classifiers learning in the BAT delegation ensemble have been trained for fewer iterations than the classifiers in the full ensemble, and thus have higher plasticity [15].

5.2.1 Class-Incremental Learning. In our second experiment, we perform a more thorough investigation of the k-BAT and Student-Expert methods using various parameter combinations in our delegative framework. In particular, we vary the number of active

Method	Mean Acc.	C_1	C_2	C_3	C_4	C_5
Student-Expert	91.75 ± 0.39	93.05 ± 0.85	89.95 ± 1.07	92.08 ± 0.42	92.90 ± 0.15	90.79 ± 0.33
GEM	**95.78 ± 0.22**	98.39 ± 0.23	**95.19 ± 0.65**	**94.98 ± 0.76**	**95.79 ± 0.35**	94.55 ± 0.33
Replay	72.95 ± 1.76	80.85 ± 3.00	60.96 ± 4.77	51.45 ± 6.30	74.20 ± 4.73	**97.27 ± 0.35**
MIR	52.42 ± 10.00	60.00 ± 21.01	57.34 ± 14.61	64.48 ± 22.05	0.12 ± 0.21	80.16 ± 14.99
k-BAT	43.27 ± 3.38	73.18 ± 14.16	2.45 ± 5.10	93.28 ± 4.34	35.88 ± 17.49	11.54 ± 17.10
CWR	19.98 ± 0.01	**99.88 ± 0.05**	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
LwF	19.37 ± 0.14	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	96.87 ± 0.70
MAS	25.26 ± 3.18	18.04 ± 11.09	1.99 ± 2.87	0.34 ± 0.57	9.48 ± 8.93	96.47 ± 0.26
RWalk	19.43 ± 0.06	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	97.17 ± 0.28
Single Learner	19.44 ± 0.05	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	97.19 ± 0.26
Full Ensemble	19.28 ± 0.08	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	96.40 ± 0.41

Table 1: Test accuracy on Split MNIST for all methods averaged over 10 trials in the class-incremental learning setting evaluated after training on all contexts. Results are grouped into 3 sections; the top section shows the performance possible without memory and when leveraging structure within the test set. The middle section compares against pre-existing methods that rely on memory. The bottom section compares methods that do not use memory. Our methods show that ensembles are able to provide significant performance boosts when leveraging structure in test data (Student-Expert with 1 train, 1 test guru), as well as when not memorizing previous examples or being given the context label (k-BAT with $k = 1$, Probabilistic Better delegation probability function and balanced accuracy as the performance metric).

Method	Mean Acc.	C_1	C_2	C_3	C_4	C_5
Student Expert	92.46 ± 1.39	93.59 ± 0.62	89.21 ± 3.27	92.34 ± 2.41	92.12 ± 3.94	95.05 ± 0.68
GEM	**95.41 ± 0.60**	**94.66 ± 1.44**	**94.93 ± 1.50**	**94.72 ± 0.90**	**95.71 ± 0.81**	97.05 ± 0.21
Replay	75.25 ± 8.73	64.86 ± 16.41	71.25 ± 21.33	64.59 ± 16.54	78.40 ± 11.79	97.13 ± 0.36
MIR	61.97 ± 11.91	44.83 ± 24.31	57.07 ± 32.44	48.97 ± 23.19	61.83 ± 22.91	97.15 ± 0.31
k-BAT	74.78 ± 7.24	73.92 ± 19.46	64.51 ± 25.03	65.59 ± 18.22	85.08 ± 9.95	84.82 ± 8.49
CWR	55.00 ± 8.06	65.32 ± 16.09	43.12 ± 19.88	43.83 ± 14.78	62.34 ± 17.54	60.40 ± 13.44
LwF	70.81 ± 9.06	56.92 ± 21.97	62.23 ± 30.19	59.89 ± 20.91	77.81 ± 14.57	**97.20 ± 0.27**
MAS	65.78 ± 7.57	62.66 ± 19.10	58.79 ± 21.73	51.29 ± 17.00	72.80 ± 13.26	83.38 ± 4.10
RWalk	67.77 ± 9.77	53.48 ± 21.63	62.51 ± 28.99	54.77 ± 21.32	71.48 ± 16.10	96.61 ± 0.39
Single Learner	62.29 ± 11.36	45.83 ± 24.37	57.09 ± 32.70	48.97 ± 22.85	62.51 ± 21.16	97.04 ± 0.18
Full Ensemble	59.42 ± 11.86	42.10 ± 25.15	54.10 ± 33.59	45.58 ± 22.74	59.09 ± 23.05	96.23 ± 0.44

Table 2: Test accuracy for all methods averaged over 10 trials in the domain-incremental learning setting using the Rotated MNIST dataset. All methods are evaluated after training on all contexts. k-BAT uses the random better delegation probability function and F1 score as the performance metric and 2 gurus during training. Student-Expert uses the probabilistic better delegation probability function and accuracy as the metric with 2 classifiers active during training and testing. Most methods show strong test performance on the most recently learned data while exhibiting moderate forgetfulness on previously learned data. k-BAT exhibits much stronger performance than existing methods that do not rely on replay. The Student-Expert algorithm (top row) highlights the performance possible on an ordered test stream, even without memory.

learning gurus during training for both the k-BAT and the Student-Expert method, and we vary the number of active predicting gurus for the Student-Expert method. We also vary the delegation probability function and the classifier competency metric for k-BAT.

Table 1 shows the average accuracy of each method on the full test set as well as the average accuracy in each context on the Split MNIST dataset. Both k-BAT and the Student-Expert method significantly outperform naive methods (Full Ensemble and Single Learner). The Student-Expert method is the only one to leverage order in the test set and thus significantly outperforms all similar methods that do not rely on memory. Order in the test set is rarely available, so we separate these results from others. However, even k-BAT, which does not receive any batch labels during testing, is more accurate than other methods that do not rely on memory.

The performance on each of the five contexts reveals that k-BAT and Student-Expert successfully remember information from the first four contexts, indicating a strong ability to avoid catastrophic forgetting. In regard to methods which predict on the whole test set statically, our k-BAT method with 1 active learning guru performs particularly well, achieving greater than zero accuracy in each context.

The Student-Expert model shows the benefit that comes with structure in an online test stream. Overall, we see that Student-Expert with 1 guru during both training and testing achieved a 91.75% test accuracy while k-BAT with 1 guru achieved only a 43.27% accuracy. However, even k-BAT is much higher than all other methods that similarly don't rely on memory.

Figure 4 shows that, in the larger class-incremental setting with 30 classifiers, having as many as 7 gurus can maximize performance. However, in smaller ensembles, 1 guru appears optimal. When varying the delegation probability function and classifier competency metric in k-BAT, we found no significant differences between any of them. A comparison of test accuracy across these parameters is shown in Appendix A.

5.2.2 Domain-Incremental Learning. In our third experiment, we investigate the performance of k-BAT and Student-Expert in the domain-incremental setting. The results of this experiment can be seen in Table 2. The Student-Expert method significantly outperforms all other methods that do not use memory, achieving an average test accuracy of 92.46% with 2 active gurus during training and testing. This is primarily due to the fact that Student-Expert operates on an ordered, online test stream, as discussed previously. Of the remaining methods that don't use memory, k-BAT with $k = 2$ achieves the highest average accuracy of 74.78%.

Impressively, both Student-Expert and k-BAT perform similarly well in the first context as in some later contexts. Overall, both of our methods avoid significant forgetting.

When averaging over all parameter configurations for k-BAT, using $k = 3$ resulted in the best performance. Similar to our findings in the class-incremental setting, each performance metric leads to similar performance. Of the delegation probability functions, max-diversity slightly outperforms the other options, but the difference is insignificant. These results can be seen in Appendix A.

6 DISCUSSION

In this paper, we have developed new ensemble training algorithms based on delegative techniques for two types of continual learning: class-incremental and domain-incremental. Our first algorithm, k-BAT, shows a strong ability to select a unique ensemble member to learn each new context, or domain shift, without any knowledge of the context itself or the context label. This effectively teaches different members of the ensemble on different distributions and drastically reduces the catastrophic forgetting common to continual learning settings. Subsequently, our Student-Expert delegation mechanism extends k-BAT to allow delegation during testing to improve performance in the presence of ordered test data. This manages to avoid catastrophic forgetting and showcases the power of delegation for dynamically selecting classifiers to both learn and make predictions.

Curiously, in our class-incremental learning experiments with only 8 classifiers in the ensemble, we found that 1 guru led to the highest accuracy. This is not the case with domain-incremental learning, where multiple gurus are superior. Intuitively, when there are more classifiers than contexts, having multiple classifiers learn on a context should have better performance than when just one classifier is learning. Our result suggests that either (1) individual classifiers are quite strong in the class-incremental setting and that more gurus may be useful when the learning task is more difficult, or (2) there is room for improvement in the delegation algorithm. In either case, further investigation will be useful.

Generally, our experiments have varied several important parameters to demonstrate both the optimal parameter values and the robustness of our algorithms. However, there is a great deal of work that could extend this research in compelling directions. Some specific paths include:

- Optimal delegations: We have no guarantee of the quality of our delegative process. Both theoretical and further experimental work will be important to identify an upper bound on the performance improvement granted by our delegation mechanisms.
- New performance metrics for delegation: Our delegation mechanisms accept a generic signal of quality as a means of deciding delegation patterns. Measuring the change in this value over time acts as an approximation of each classifier's learning rate. While we experimented with a number of common metrics such as accuracy and f1-score, there may be metrics that more directly reveal which classifiers are optimal to learn on any given data.
- Further comparison with existing methods to avoid catastrophic forgetting in continual learning. We compared our results with three methods well suited for domain-incremental learning, however these methods prove to work poorly on the class-incremental setting.

REFERENCES

[1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert Gate: Lifelong Learning With a Network of Experts. 3366–3375. https://openaccess.thecvf.com/content_cvpr_2017/html/Aljundi_Expert_Gate_Lifelong_CVPR_2017_paper.html

[2] Ben Armstrong and Kate Larson. 2021. On the Limited Applicability of Liquid Democracy. *3rd Games, Agents, and Incentives Workshop at the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2021).

[3] Ben Armstrong and Kate Larson. 2024. Liquid Democracy for Low-Cost Ensemble Pruning. arXiv:2401.17443

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient Lifelong Learning with A-GEM. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Hkf2_sC5FX

[5] Cristina Cornelio, Michele Donini, Andrea Loreggia, Maria Silvia Pini, and Francesca Rossi. 2021. Voting with Random Classifiers (VORACE): Theoretical and Experimental Analysis. *Autonomous Agents and Multi-Agent Systems* 35, 2 (2021), 22.

[6] Thang Doan, Seyed Iman Mirzadeh, and Mehrdad Farajtabar. 2023. Continual Learning Beyond a Single Model. https://doi.org/10.48550/arXiv.2202.09826 arXiv:2202.09826 [cs].

[7] Robert M French. 1999. Catastrophic Forgetting in Connectionist Networks. *Trends in cognitive sciences* 3, 4 (1999), 128–135.

[8] Yoav Freund and Robert E Schapire. 1995. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. In *European conference on computational learning theory*. Springer, 23–37.

[9] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[10] Ludmila I Kuncheva and Christopher J Whitaker. 2003. Measures of Diversity in Classifier Ensembles and their Relationship with the Ensemble Accuracy. *Machine learning* 51 (2003), 181–207.

[11] Florin Leon, Sabina-Adriana Floria, and Costin Bădică. 2017. Evaluating the Effect of Voting Methods on Ensemble-Based Classification. In *2017 IEEE International Conference on INnovations in Intelligent Systems and Applications*. IEEE, 1–6.

[12] Zhizhong Li and Derek Hoiem. 2017. Learning Without Forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[13] Yu Liu, Sarah Parisot, Gregory Slabaugh, Xu Jia, Ales Leonardis, and Tinne Tuytelaars. 2020. More Classifiers, Less Forgetting: A Generic Multi-classifier

Paradigm for Incremental Learning. In *Computer Vision – ECCV 2020 (Lecture Notes in Computer Science)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 699–716. https://doi.org/10.1007/978-3-030-58574-7_42

[14] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M Van de Ven, et al. 2021. Avalanche: An End-to-End Library for Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3600–3610.

[15] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. 2023. Understanding Plasticity in Neural Networks. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 23190–23211. https://proceedings.mlr.press/v202/lyle23b.html ISSN: 2640-3498.

[16] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. 2020. Continual Deep Learning by Functional Regularisation of Memorable Past. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 4453–4464. https://proceedings.neurips.cc/paper/2020/hash/2f3bbb9730639e9ea48f309d9a79ff01-Abstract.html

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An Imperative sStyle, High-performance Deep Learning Library. *Advances in neural information processing systems* 32 (2019).

[18] David M Pennock, Pedrito Maynard-Reid II, C Lee Giles, and Eric Horvitz. 2000. A Normative Examination of Ensemble Learning Algorithms.. In *ICML*. 735–742.

[19] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. iCaRL: Incremental Classifier and Representation Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Honolulu, HI, 5533–5542. https://doi.org/10.1109/CVPR.2017.587

[20] Grzegorz Rypeść, Sebastian Cygert, Valeriya Khan, Tomasz Trzciński, Bartosz Zieliński, and Bartłomiej Twardowski. 2024. Divide and not Forget: Ensemble of Selectively Trained Experts in Continual Learning. http://arxiv.org/abs/2401.10191 arXiv:2401.10191 [cs].

[21] Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. 2022. Three Types of Incremental Learning. *Nature Machine Intelligence* 4, 12 (2022), 1185–1197.

[22] Liyuan Wang, Xingxing Zhang, Qian Li, Jun Zhu, and Yi Zhong. 2022. CoSCL: Cooperation of Small Continual Learners is Stronger Than a Big One. In *Computer Vision – ECCV 2022 (Lecture Notes in Computer Science)*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 254–271. https://doi.org/10.1007/978-3-031-19809-0_15

[23] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *arXiv preprint arXiv:2302.00487* (2023).

[24] Yeming Wen, Dustin Tran, and Jimmy Ba. 2020. BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning. https://doi.org/10.48550/arXiv.2002.06715 arXiv:2002.06715 [cs, stat].

[25] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual Learning Through Synaptic Intelligence. In *International conference on machine learning*. PMLR, 3987–3995.

A ADDITIONAL EXPERIMENT RESULTS

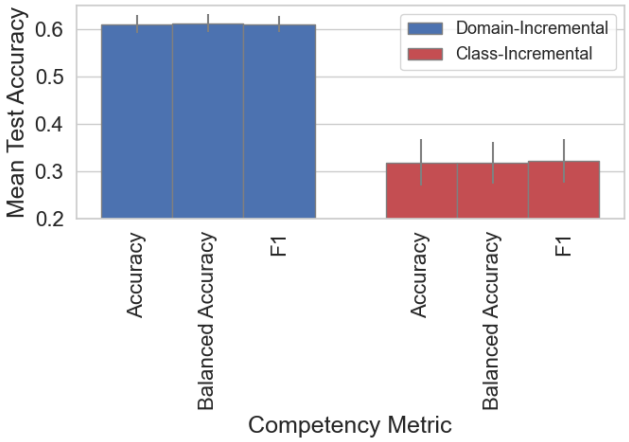

Figure 5: Comparison of performance metrics for both class- and domain-incremental learning. In both cases, there is no significant difference between any metrics we explored.

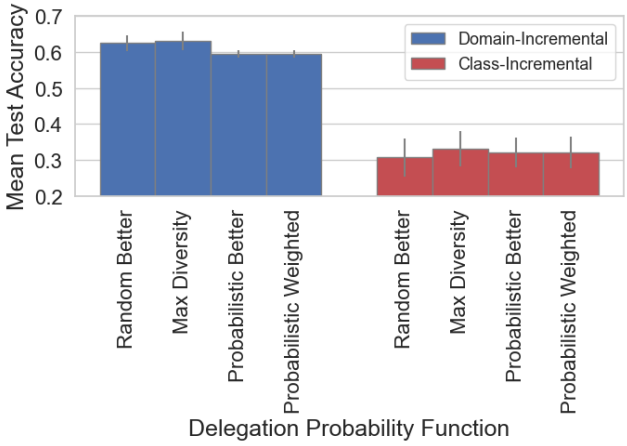

Figure 6: Comparison of delegation probability functions for both class- and domain-incremental learning. While Max Diversity may be slightly better, there is no significant difference between the functions in the experiments we have run.

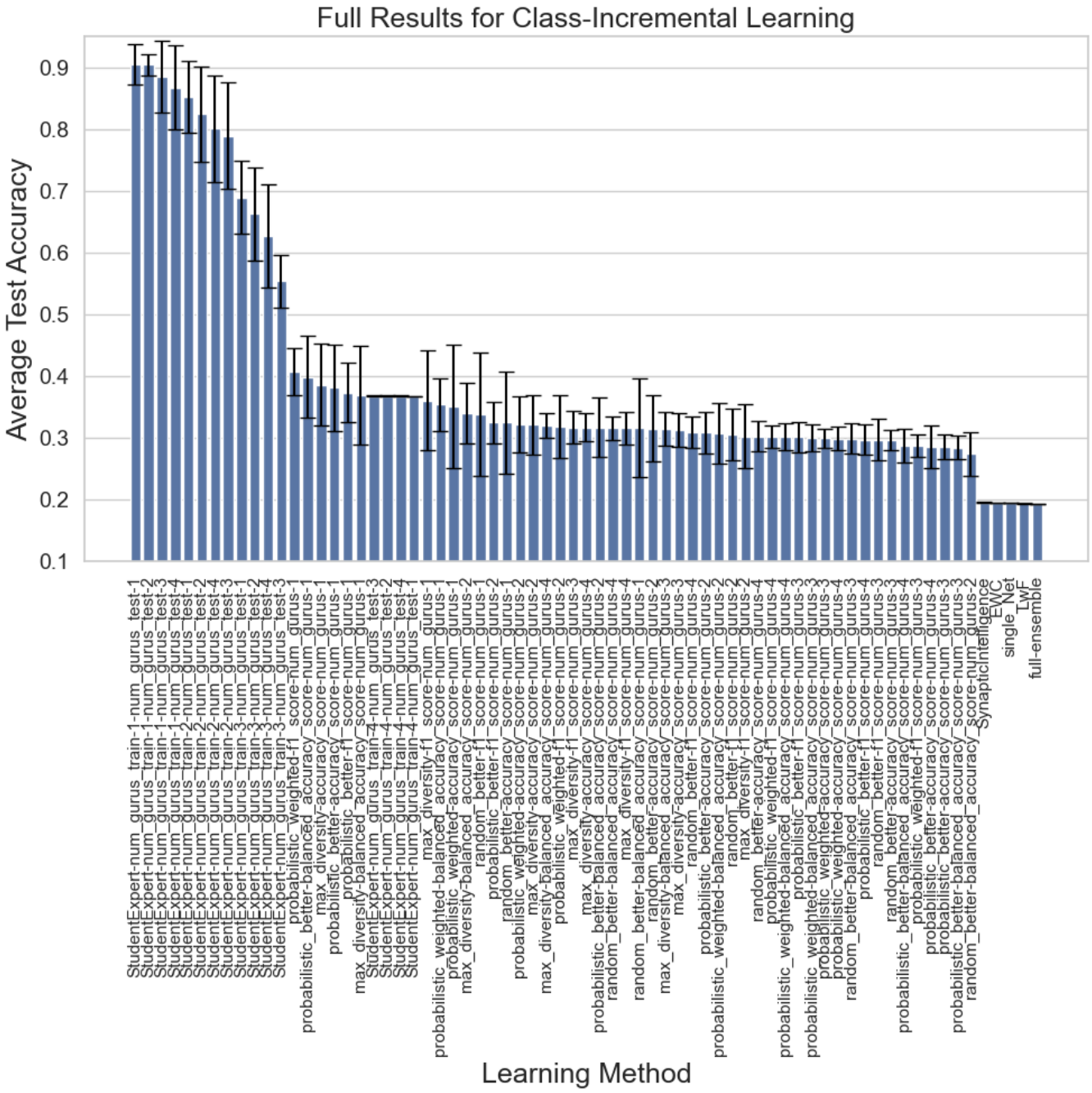

Figure 7: Average test accuracy of all compared methods on class-incremental learning. Student-Expert methods utilize the ordered test set and inevitably outperform other methods. Unlike in domain-incremental learning, methods with only 1 guru typically perform better than others.

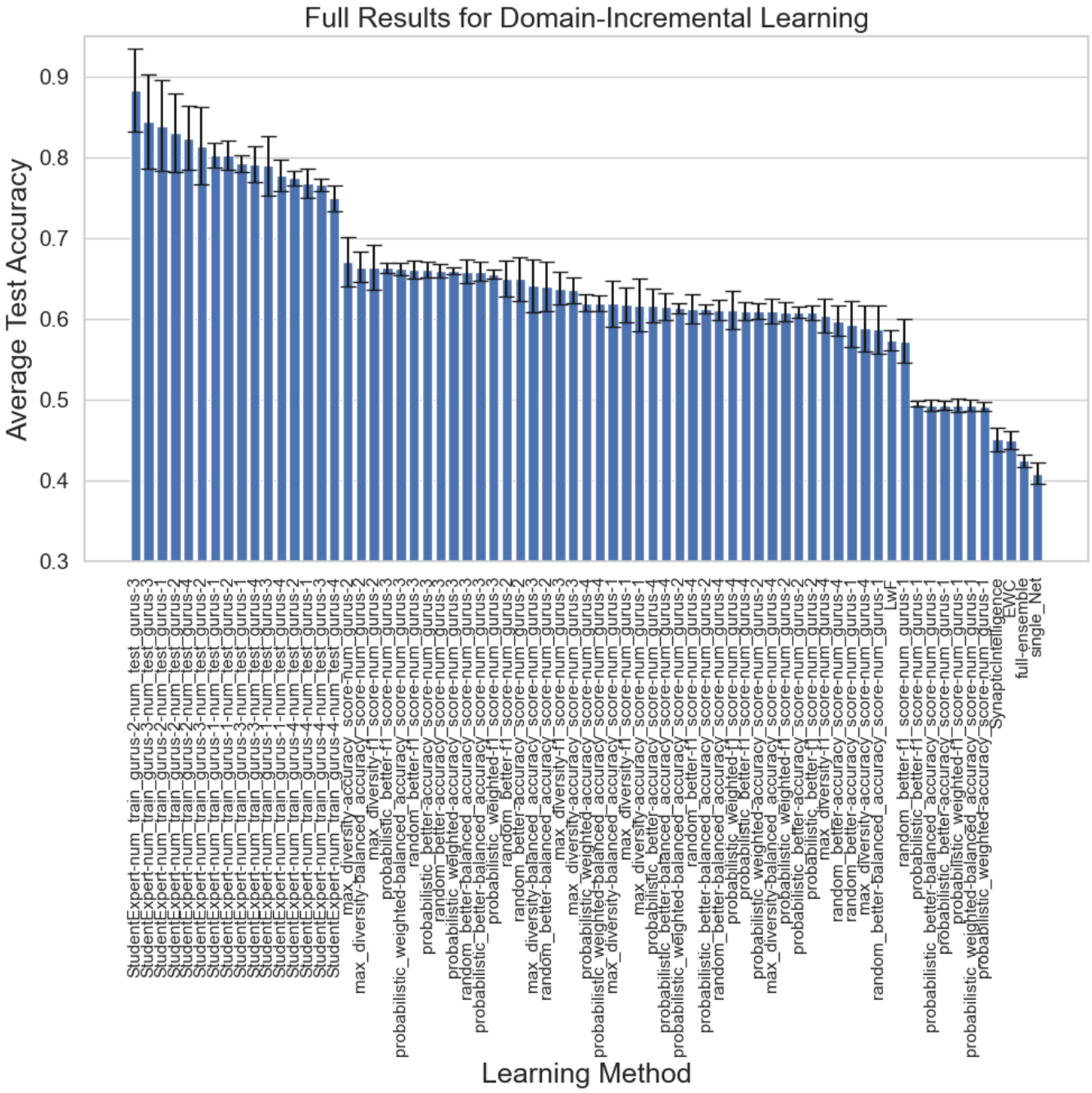

Figure 8: Average test accuracy of all compared methods on domain-incremental learning. Student-Expert methods utilize the ordered test set and inevitably outperform other methods. Unlike in class-incremental learning, methods with more than one guru typically perform better than others.