ACTIVE CLASS SELECTION FOR FEW-SHOT CLASS-INCREMENTAL LEARNING

Christopher McClurg
Pennsylvania State University
cam7498@psu.edu

Ali Ayub
University of Waterloo
a9ayub@uwaterloo.ca

Harsh Tyagi
Pennsylvania State University
hkt5106@psu.edu

Sarah M. Rajtmajer
Pennsylvania State University
smr48@psu.edu

Alan R. Wagner
Pennsylvania State University
alan.r.wagner@psu.edu

ABSTRACT

For real-world applications, robots will need to continually learn in their environments through limited interactions with their users. Toward this, previous works in few-shot class incremental learning (FSCIL) and active class selection (ACS) have achieved promising results but were tested in constrained setups. Therefore, in this paper, we combine ideas from FSCIL and ACS to develop a novel framework that can allow an autonomous agent to continually learn new objects by asking its users to label only a few of the most informative objects in the environment. To this end, we build on a state-of-the-art (SOTA) FSCIL model and extend it with techniques from ACS literature. We term this model Few-shot Incremental Active class SeleCtiOn (FIASco). We further integrate a potential field-based navigation technique with our model to develop a complete framework that can allow an agent to process and reason on its sensory data through the FIASco model, navigate towards the most informative object in the environment, gather data about the object through its sensors and incrementally update the FIASco model. Experimental results on a simulated agent and a real robot show the significance of our approach for long-term real-world robotics applications.

1 INTRODUCTION

A primary challenge faced by robots deployed in the real world is continual adaptation to dynamic environments. Central to this challenge is object recognition (Ayub & Wagner, 2020d), a task typically requiring labeled examples. In this work, we address the problem of parsimonious object labelling wherein a robot may request labels for a small number of objects about which it knows least.

In recent years, several works have been directed toward the problem of Few-Shot Class Incremental Learning (FSCIL) (Tao et al., 2020; Ayub & Wagner, 2020c) to develop models of incremental object learning that can learn from limited training data for each object class. The literature has made significant progress toward developing robots that can continually learn new objects from limited training data while preserving knowledge of previous objects. However, existing methods make strong assumptions about the training data that are rarely true in the real world. For example, FSCIL assumes that in each increment the robot will receive a fully labeled image dataset for the object classes in that increment, and the robot will not receive more data for these classes again (Tao et al., 2020; Ayub & Wagner, 2020c;d). In real world environments, however, robots will most likely encounter many unlabeled objects in their environment, and they will have to direct their learning toward a smaller subset of those unknown objects.

Active learning is a subfield of machine learning that focuses on improving the learning efficiency of models by selectively seeking labels from within a large unlabeled data pool (Settles, 2009; Ayub & Fendley, 2022). Related to active learning is active class selection (ACS) in which a model seeks labels for specific object classes (Lomasky et al., 2007). ACS can allow autonomous robots operating in real-world environments to focus their learning objects about which they know least. Most ACS models, however, have been designed for batch learning, i.e., they require all the previous training data to be available when learning in an increment (Lomasky et al., 2007). Further, both active learning and ACS techniques have previously been tested on static datasets rather than with real agents/robots (Lomasky et al., 2007; Yoo & Kweon, 2019; Siddiqui et al., 2020).

In this paper, we combine ideas from ACS and FSCIL to develop a framework that can allow an autonomous agent roaming in its environment to continually adapt by learning about the most informative objects through interaction

Figure 1: Pepper learns about the environment by actively selecting classes to incrementally train on.

with its human users. Toward this, we build on a state-of-the-art (SOTA) FSCIL model and extend it with techniques from ACS literature. We term this model Few-shot Incremental Active class SeleCtiOn (FIASco). We further integrate a potential field-based navigation technique with our model to develop a complete framework that can allow an agent to process and reason about its sensory data, navigate towards the most informative object in the environment, gather the data for the object through its sensors and incrementally update the FIASco model. We perform extensive evaluations of our approach in a simulated Minecraft environment and with a real robot in a laboratory setting. The main contributions of the paper are as follows: (1) We develop a novel framework extending FSCIL techniques with ideas from ACS and integrating it with autonomous agents. (2) Our experiments on a simulated and a real autonomous agent demonstrate the effectiveness and applicability of our framework for the long-term deployment of robots in real-world environments.

2 Background

Class-incremental learning (CIL) considers the problem where labeled data is provided to the learner in increments of full classes. When applied to neural networks, CIL results in catastrophic forgetting, where the model forgets the previously learned classes and classification accuracy erodes (Kirkpatrick et al., 2017a). A limitation of recent CIL methods is the reliance on storing a portion of the data from prior classes when learning new classes (Rebuffi et al., 2017; Castro et al., 2018; Wu et al., 2019). These methods, often storing high-dimensional images, are not practical in situations when the system has limited memory. To avoid storing real images, some CIL methods use a regularization loss term to prevent the weights of the model from changing drastically when learning new classes (Kirkpatrick et al., 2017b; Li & Hoiem, 2018; Dhar et al., 2019). Other CIL methods regenerate images of the old classes with generative models (Ostapenko et al., 2019; Ayub & Wagner, 2021).

In a preliminary experiment, we compare the performance of a recent clustering approach (CBCL-PR) (Ayub & Wagner, 2020c;a;b) against three popular CIL algorithms in a few-shot class-incremental learning setting: iCARL, PODNet, and DER. iCaRL (Rebuffi et al., 2017) stores exemplars in memory, uses a regularization term called distillation-loss (Hinton et al., 2015), and Nearest Class Mean (NCM) to classify data (Mensink et al., 2013; Dehghan et al., 2019). PODnet (Douillard et al., 2020) stores proxy vectors in memory, uses a spatially-based distillation-loss, and also uses an NCM classifier. DER (Yan et al., 2021) uses a two-stage approach that freezes previously learned representations and then augments the model with features from a fine-tuned extractor. The results of this preliminary experiment are contained in the appendix.

Few-shot class-incremental learning (FSCIL) adapts the class-incremental learning problem by limiting the number of training examples per class. Specifically, the data is first divided among training and test sets such that $x_i \in (X^{train} \cup X^{test})$, $y_i \in (y^{train} \cup y^{test})$. Then the training data is divided into increments $x_i^{train} \in (D_0^{train} \cup D_1^{train} \cup ...D_n^{train})$, $y_i^{train} \in (C_0 \cup C_1 \cup ...C_n)$ such that each increment is composed of a unique set of classes (i.e., $\forall i, j \ni i \neq j, C_i \cap C_j = \emptyset$). In the i-th increment, the model only trains on the corresponding training data $\{D_i^{train}, C_i\}$. The model is then evaluated on a test set that includes all classes seen so far (i.e., $\{\bigcup_{j=1}^{i} D_j^{test}, \bigcup_{j=1}^{i} C_j\}$). The size of an increment is D_0^{train} containing N_b of full classes. A problem setting which contains N classes per increment and k examples per class is known as N-way k-shot learning. In FSCIL, the problem is typically formatted with 100 full classes in the first increment, and then 10-way 5-shot learning for the remaining increments (Tao et al., 2020).

In a preliminary experiment, we compare the performance of CBCL-PR (Ayub & Wagner, 2020c;a;b) against five other FSCIL algorithms: TOPIC, SPPR, Decoupled-DeepEMD, CEC, and FACT. TOPIC (Tao et al., 2020) represents knowledge with a neural gas network in order to preserve the topology of the feature space. SPPR (Zhu et al., 2021) uses prototype learning, including random episode selection to adapt the feature representation and a dynamic relation projection between old and new classes. Decoupled-DeepEMD (Zhang et al., 2020) decouples the training of the embedding and the classifier; the embedding is trained on the initial increment of 100 full classes, while the subsequent increments replace class-specific classifiers with new mean embeddings. CEC (Zhang et al., 2021) trains an additional graph model to adapt prototypes of old and new classes. FACT (Zhou et al., 2022) is the current state-of-art, which uses prototypes to limit the embedding space of old classes, reserving space for new classes. The results of this preliminary experiment are contained in the appendix.

Active Class Selection (ACS) considers the problem where the learner can improve learning efficiency by requesting more data from a specific class (Lomasky et al., 2007). In prior work, ACS was piloted to enable an artificial nose to efficiently learn to discriminate vapors (Lomasky et al., 2007). In a batch learning setting, the learner used feedback from the previous batch to influence the class distribution among samples in the next class. A recent approach to ACS, PAL-ACS, demonstrated high performance by generating pseudo-examples, transforming an ACS problem into an active learning problem (Kottke et al., 2021). This study was, however, limited to synthetic data.

Active incremental learning considers the problem where incremental learning and active learning are combined. In active learning, the learner may actively request labels for training data. One study assumed labels are no longer provided in the CIL setting (Belouadah et al., 2020). Another study allowed a learner to incrementally select points for labeling from a point cloud (Lin et al., 2020). A third study allowed a learner to incrementally select examples for annotation by a human expert (Brust et al., 2020). In these studies, the incremental learner selects training data to label, which defines the active learning problem. In contrast, this paper uses incremental learning to select classes to receive additional training instances, which is an active class selection problem.

3 Model Description

Our goal is to develop a model (FIASco) that can not only learn incrementally, but can also select – from observed classes in a novel environment– classes which to receive more training instances. This problem is a modified class-incremental learning problem, whereas the next training class is determined by environmental availability and agent affinity. To learn incrementally, we ran preliminary experiments (see appendix) to identify CBCL-PR (Ayub & Wagner, 2020b; 2023) as the most promising approach for this problem. The identified approach not only produces SOTA results on few-shot incremental learning benchmarks, but also represents object classes as clusters, which have intrinsic statistics that can be used to to select the next training class in an environment. An overview of the model is shown in Figure 2. In this section, we describe the components of FIASco, including incremental learning with clustering (Section 3.1), active class selection with cluster statistics (Section 3.2), and navigation using a potential field created by cluster-averaged statistics of the observed classes in the environment (Section 3.3).

3.1 Incremental Learning with Clusters

In each increment, the learner receives the training examples (images) for new classes. Feature vectors of the images are generated using a pre-trained convolutional neural net as a feature extractor. Clusters are created from feature vectors that are within a tolerable distance of one another, enabling discrimination between classes and consolidation of these classes into long-term memory. For more details of this clustering approach, please see the appendix or related literature (Ayub & Wagner, 2020c;a;b).

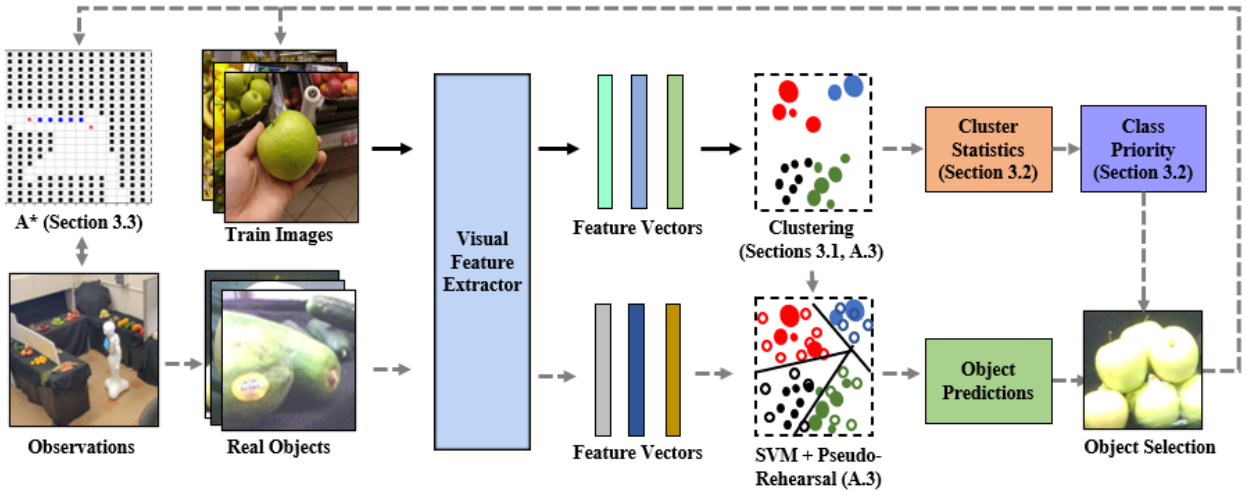

Figure 2: This flow summarizes the training phase of FIASco. An agent uses a fixed feature extractor to obtain and cluster feature vectors from training images (solid line). The resulting centroids are used to fit a linear SVM, which is then used for predicting real objects. Cluster statistics are used to inform the agent which real objects to pursue and request more examples (training images). The training process combines few-shot class-incremental learning with active class selection. **Please see the appendix for additional info on clustering or pseudo-rehearsal.*

3.2 Active Class Selection with Cluster Statistics

We extend the learning approach to use feedback from cluster statistics. Specifically, the cluster space allows for measures – cluster weight, class weight, and cluster variance – to guide the selection of new samples for training.

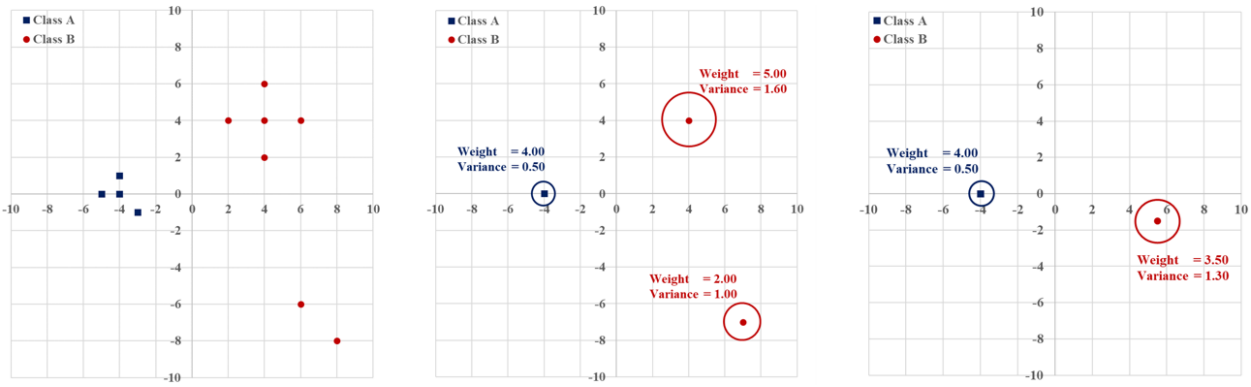

Figure 3: *Left*. An example distribution of data, where each point represents a training instance plotted in a two-dimensional feature space. *Middle*. The clustering process groups similar training instances, extracting useful information such as cluster weight and cluster variance. *Right*. The cluster-averaged class weight and class variance can be used for determining the next class to request.

Cluster weight is the number of training examples included in an individual cluster within a class. Likewise, class weight is the number of training examples per class. Cluster variance is calculated in a recursive manner such that prior training data is not needed. As defined by Welford's method, the n-th update ($n > 1$) of a cluster's variance is s_n^2 (Welford, 1962; Knuth, 2014):

$$(n-1)s_n^2 - (n-2)s_{n-1}^2 = (x_n - \bar{x}_n)(x_n - \bar{x}_{n-1}) \tag{1}$$

These internal measures give direct feedback for active class selection (ACS). Recall that previous ACS methods use results from the previous batch as feedback to specify the distribution of classes in the next batch. In incremental

learning, the learner does not control the size of new batches. Therefore, class selection is instead an ordering of preferred classes:

1. *Low Class Weight:* Prioritize classes with lower class weight. The intuition for this ordering is that adding instances to a class with fewer instances will likely add useful information (new clusters), increasing overall accuracy.
2. *Low Cluster Weight:* Prioritize classes with lower average cluster weight. The intuition for this ordering is that adding instances to classes with undeveloped clusters (outliers) will be more likely to impact (shift/ add weight to) the class-specific space, increasing overall accuracy.
3. *Low Cluster Variance:* Prioritize classes with lower average cluster variance. The intuition for this ordering is that adding instances from classes with less noise will likely add valuable information with minimal overall noise.
4. *High Cluster Variance:* Prioritize classes with higher average cluster variance. The intuition for this ordering is that adding instances from classes with more uncertainty will likely provide more distinct clusters within the class.

To further illustrate these measures, consider a distribution of two classes of data, as shown in Figure 3. Each instance of data is initially plotted in the two-dimensional vector space (left). The clustering process (middle) extracts useful cluster information, such as weight and variance. Finally, the extracted information can be cluster-averaged per each class of data (right). Which class of data should be requested next for the purpose of training? According to the low class weight metric, class A should be requested $(4.0 < 7.0)$. According to the low cluster weight metric, class B should be requested $(3.5 < 4.0)$. For low cluster variance, class A should be requested $(0.5 < 1.3)$. Of course, class B should be requested for high cluster variance $(1.3 > 0.5)$.

3.3 Navigation from Active Class Selection

Integrating our incremental ACS approach on an autonomous agent requires developing a method for navigation to move towards the most informative data samples. The selected method for navigation was a potential field approach, simplified from (Koren et al., 1991). Figure 4 shows a potential field created from agent observations in the simulation.

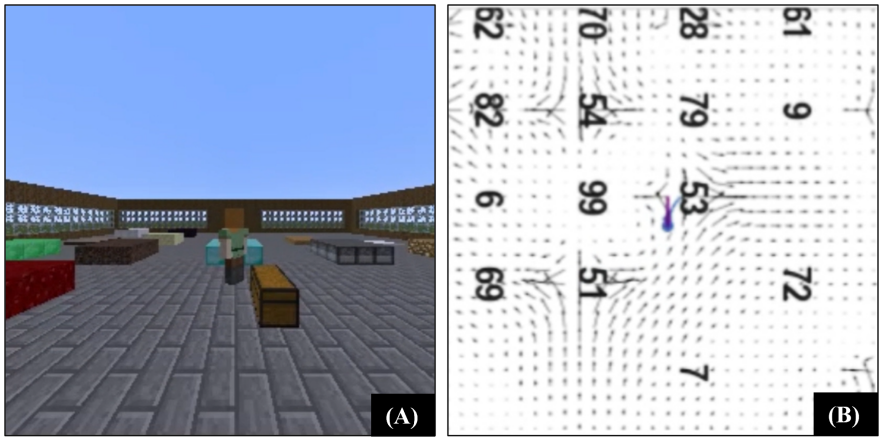

Figure 4: A view of the agent in simulation (A) and the potential field created for navigation (B).

Motivated to apply these methods on a real robot that can make some inference about distal objects ($d \leq d_{far}$) and then identify objects at a closer distance ($d \leq d_{close} < d_{far}$), the learner is given similar characteristics. In experiment, distances for class identification and feature extraction were set to d_{far} and d_{close}, respectively. Objects within distance d_{far} would be included in the learner's internal potential field, where the true class label would be known by the robot (i.e., close enough to ask a person for the true labels). For the i-th object in the potential field, an attractive or repulsive force f_i was assigned based on the order of class priority determined in ACS. The potential field is then defined by equation (2), where the i-th observation is made at (x_i, y_i) and the robot position is (x_0, y_0). Objects are only learned when the robot is within the distance d_{close}, where an image can be taken and features extracted for training.

$$(F_x, F_y) = \left(\sum_{i=1}^{n_i} \frac{f_i}{x_i - x_0}, \sum_{i=1}^{n_i} \frac{f_i}{y_i - y_0} \right) \tag{2}$$

A common problem with potential fields is that the agent can get stuck in a local minima. Past solutions for this local minima problem have included adding small, random perturbations or adjusting the gain of a particular contribution to a potential field (Arkin, 1989). In our simulated experiment, the number of time steps spent inside a relative location is counted. If the learner exceeds a specified count limit, it is directed back to the start position. Every time the learner returned to the start position, it is sent in a new direction (i.e., if the learner came from the North, it is randomly sent East, South, or West).

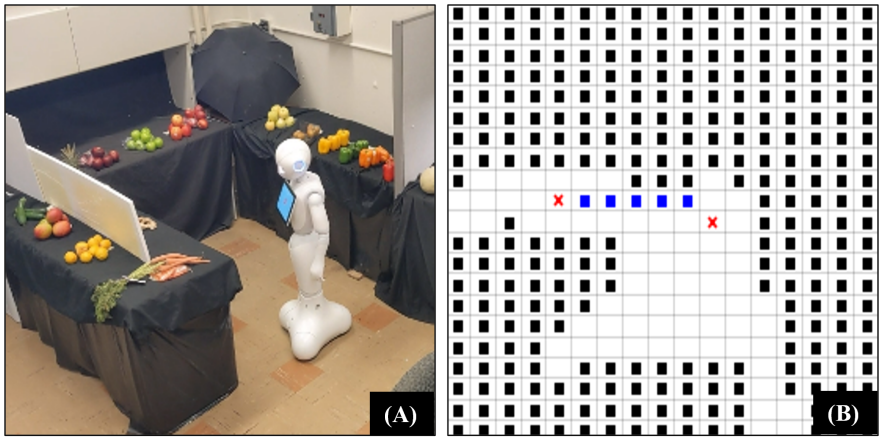

Figure 5: A view of the robot in the environment (A) and the A* path created for navigation (B).

In our experiment with a real robot, sensor error also presented problems. That is, not only is there possibility of getting stuck in a local minima, an undetected obstacle would also prevent movement of the robot. To mitigate the effects of sensing error, rather that use a continually-adapting potential field, the robot observed its surroundings once, then used A* path-planning (Hart et al., 1968) to get to the location of selected class. This navigation method has less benefit for actively selecting classes; please see the results from Section 5.2 for a discussion, or the appendix for more info on the A* method. Figure 5 shows the A* path planning from robot observations in the environment.

4 Experiment: FSCIL-ACS in Minecraft

Our first experiment is an image classification task within the Minecraft simulation environment. We aim to show that a simulated robot can use internal feedback based on what it has learned about the environment (cluster space) to more efficiently seek unknown objects in the environment.

4.1 Experimental Setup

Overview. A robot in Minecraft is given two minute intervals to search the environment for new visual examples of objects. The robot navigates with an internal potential field, created from objects within an observable distance ($d < d_{far} = 15$). The robot can observe visual examples of an object *only* when it stands over that object ($d < d_{close} = 1$). After the interval of searching, the robot processes the visual examples by updating its cluster space (FIASco) or re-training on all of the previous training data (SVM). Finally, the robot makes predictions on the test data (static subset of original dataset) and classification accuracy is recorded. The robot's affinity to different classes of items is updated using the ACS methods described in Section 3.2 , which directly affects the future potential field for navigation. The experiment continues for 360 minutes. Please see the supplemental material for experiment replication notes.

Baselines. Cluster-based ACS methods were compared with a batch learner using 'uniform' and 'redistricting' class selection. The 'uniform' method randomly sets the class order so that all classes have an equal opportunity to be prioritized. The 'redistricting' method uses cross-validation to determine the most volatile (changing predictions when new samples are added in the validation stage) classes to prioritize. The cluster-based ACS methods are described in

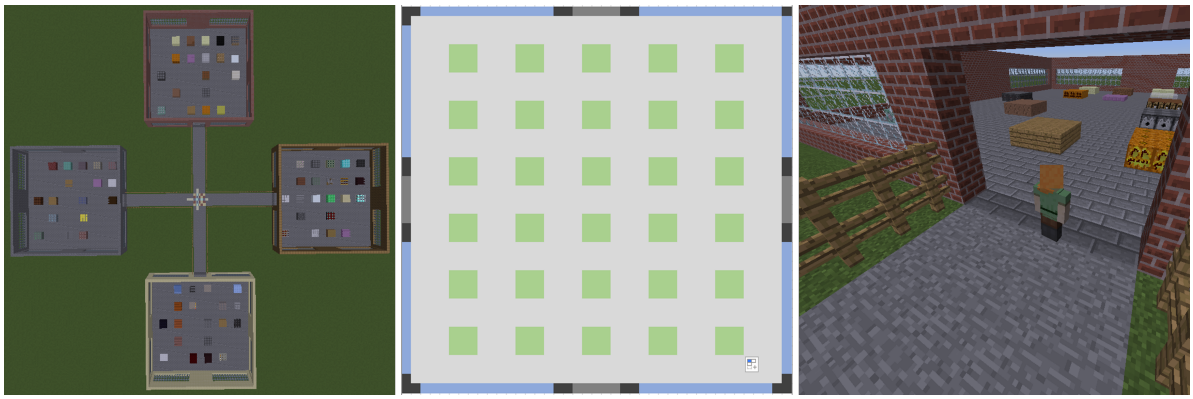

Figure 6: *Left*. The map depicts the layout of the simulation environment. *Middle*. A single building has 30 containers of items, which are arranged randomly. *Right*. The agent navigates between buildings, looking for particular items.

Section 3.2. Note that 'uniform' is also run for FIASco and that 'high cluster variance' is most similar to the previous 'redistricting' method without the time-consuming validation step.

Environment. Minecraft was used because it offers a large number of items and user control to create maps, enabling a realistic, yet constrained spatio-temporal situation for an agent (Johnson et al., 2016). The experiment map (Figure 6, left) contained four buildings. These buildings housed four unique groups of classes, grouped by the similarity of class-averaged feature vectors (centroids). Within a building, items were randomly, uniquely assigned to one of the thirty containers (Figure 6, middle). These containers served as the link to real-world items. As an agent approached the location of a container, it would observe a certain type of Minecraft item. This observation was then mapped to a class of the training dataset. While in the proximity of a container, the agent could choose to learn about the class by standing directly over the container. In this case, the agent would receive a random 5-9 instances of a class for training, after which the container would be empty. The container does not restock until the next round of exploration, after the agent trains and updates its class affinity.

Data. Two datasets were used for training and testing of the image classifier: CIFAR-100 (Krizhevsky et al., 2009) and the Grocery Store (Klasson et al., 2019) datasets. CIFAR-100 contains 60,000 32x32 images, evenly distributed among 100 classes. The classes include various types of objects, such as "beaver" or "rocket." The Grocery Store dataset contains 5,125 348x348 pixel images, non-uniformly distributed among 81 classes. The classes include various goods found in grocery stores, such as types of fruits, vegetables, and packages. Both datasets were modified to have a 90:10 stratified train-test split. Please see the Appendix for more information about the data selection.

Implementation. The fixed feature extractor in this experiment was a Resnet-34 model pre-trained with Imagenet. The test was run with ten random seeds and the average was determined. For clustering, the distance threshold D and number of pseudo-exemplars N_P were determined by validation. For the CIFAR-100 test, the values for D and N_P were set to 17 and 5, respectively. For the Grocery Store test, the values for D and N_P were 15 and 40, respectively. For batch learning, a support vector machine with a linear kernel was used (Boser et al., 1992) to make test predictions given all extracted features.

4.2 Experimental Results

Results are shown in Figure 7. The metric used for comparison was average incremental accuracy. Note that the accuracy computed in this experiment is different from the preliminary study: rather than testing over only *seen* classes, the learner is tested over *all* classes in the environment. The highest performer in the CIFAR-100 test was FIASco with 'low class weight' ACS (44.2%), an improvement of 3.7% over the best case of batch learning 'uniform' ACS. The highest performer in the Grocery Store test was FIASco using 'low class weight' ACS (63.4%), an improvement of 5.3% over the best case of batch learning 'uniform' ACS.

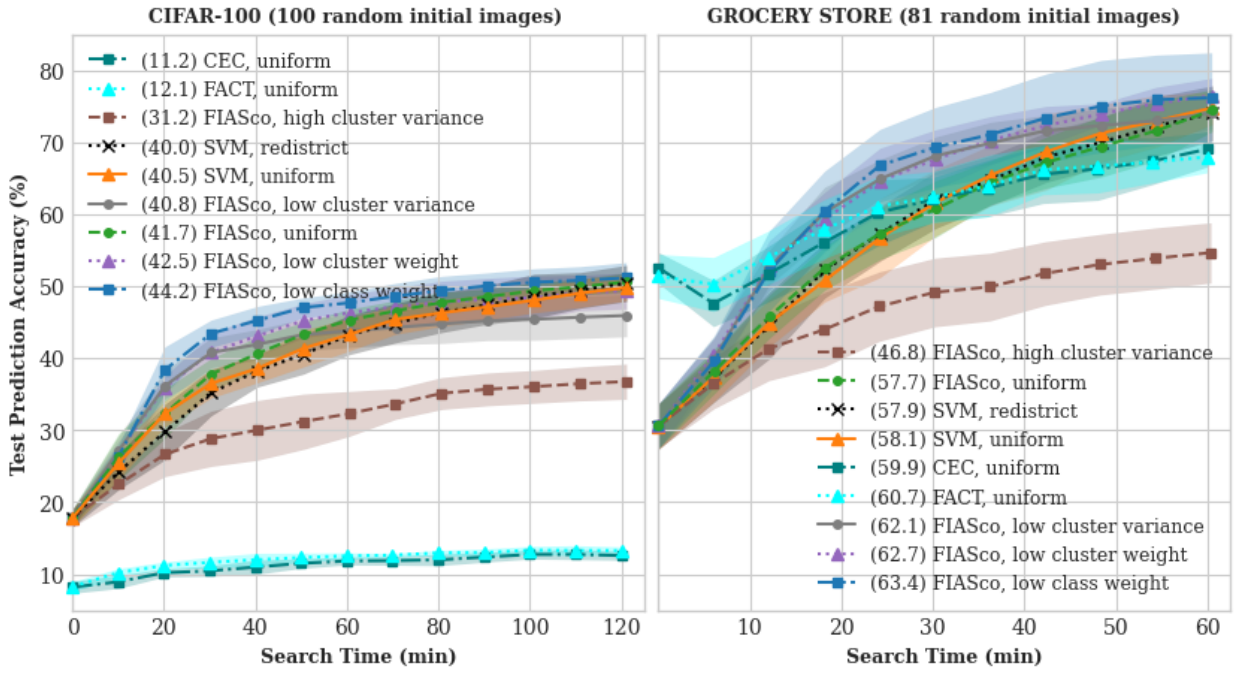

Figure 7: Test prediction accuracy over time in Minecraft simulation. Note that SVM classifier is a batch learner, while FIASco, CEC, and FACT do not re-use training data. Average incremental accuracy is indicated.

5 Experiment: FSCIL-ACS with Pepper

In the final experiment, a Softbank Pepper robot was tasked with an image classification in an indoor environment. We aim to demonstrate that a real robot can use active class selection to more efficiently seek unknown objects (see Figure 1).

5.1 Experimental Setup

Overview. The robot is given sixty iterations to search the environment for new visual examples of objects. An iteration consists of the robot (1) relocating, (2) searching, (3) choosing an object, and (4) receiving training examples. To relocate, the robot first rotates with range sensors to define a localized map; an end location is chosen among the free space, and A* path planning is used (Hart et al., 1968). To search, the robot uses a top camera, which provides up to 2560x1080 pixel resolution at 5 fps. After taking images of the surrounding area, the robot uses the YOLO algorithm (Redmon et al., 2016) pre-trained on the Microsoft COCO dataset (Lin et al., 2014) for object localization. To choose an object, the robot uses centroids for (initially weaker) classification with active class selection to pick the most desirable class. To receive training examples, the robot shows the human experimenter an image of the desired class, for which the human can give the true label of the predicted class, as well as ten visual examples. After every iteration, the robot updates its cluster space of learned classes. The robot's affinity to different classes of items is updated using the ACS methods. At the end of every three iterations, the robot makes predictions on the test data and classification accuracy is recorded.

Baselines. Cluster-based ACS methods (Section 3.2) were compared with a batch learner using 'uniform' class selection, which randomly sets the class order so that all classes have an equal opportunity to be prioritized.

Environment. This test was completed in an indoor environment, where items were purchased from a local grocery store to represent classes in the Grocery Store dataset (Klasson et al., 2019). Black cloths were used to cover tables and serve as a backdrop for items. Please see supplemental materials for images of the included classes.

Data. The Grocery Store dataset (Klasson et al., 2019) was used for training and testing of the image classifier, as in Section 4. The continually-trained image classifier was used for object recognition of the real objects in the experiment. A subset of 41 classes of the Grocery store dataset was used, comprised of items that could be primarily

stored at room temperature. The dataset was modified to have a 90:10 stratified train-test split. Real items were distributed randomly by their coarse labels, such that similar items were grouped together (e.g., Red Delicious and Yellow Delicious apples). Please see the Appendix for more information about the data selection.

Implementation. The fixed feature extractor in this experiment was a Resnet-34 model pre-trained with Imagenet. For clustering, the distance threshold D and number of pseudo-exemplars N_P were determined by validation. For this test, the values for D and N_P were 15 and 40, respectively. For batch learning, a support vector machine with a linear kernel was used (Boser et al., 1992) to make test predictions given all extracted features.

5.2 Experimental Results

Results are shown in Figure 8.

The metric used for comparison was average incremental accuracy. The accuracy computed in this experiment is the same as in Section 4: the learner is tested over *all* classes in the environment. The highest performer in the test was FIASco with 'high cluster variance' ACS (60.7%), an improvement of 0.4% over the best case of batch learning 'uniform' ACS.

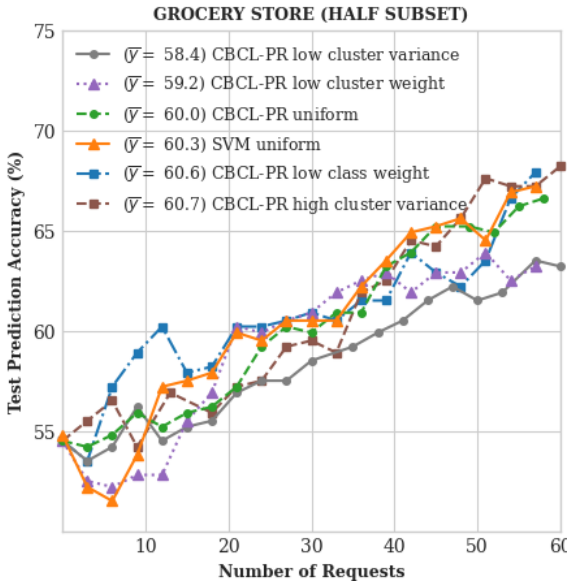

Figure 8: Test prediction accuracy over iterations in indoor environment with Pepper. Note that SVM classifier is a batch learner, while FIASco does not re-use training data.

While both experiments have a learner using the same measures to prioritize classes, there is a difference in the value of particular measures (e.g., high cluster variance.) This difference is likely due to the slight change in process, where the robot learner is making an initially weak prediction about the detected object classes before requesting a class (see Section 5.1). Hence, a wrong prediction about a class with a high variance may actually provide valuable insight into the divisions of nearby classes.

In terms of average incremental accuracy, the FIASco model does not show as much improvement over ACS with SVM, as compared to the simulated experiment. This result is likely due to the limitations in real navigation, noted in Section 3.3. When the agent moved in simulation, the potential field was updated at every time step, calculating attractive weights for each new position of observed class. In the real environment, the robot made one full turn to observe its surroundings, then followed a path prescribed by A*. As the robot moved, new class observations were not included as options for the robot. The reason for this change was due to our particular robot being susceptible to drift error and sensor noise; we choose to reduce the sensing demand such that the robot would not get itself stuck as frequently. Note that the navigation method is kept constant in each experiment, so the comparison of ACS methods still holds true. In future studies, it would be helpful to improve the robot controller so that the more reactive navigation method could be used.

6 Conclusion

To the authors' knowledge, active class selection (ACS) has not previously been combined with few-shot incremental learning (FSCIL). This paper extends an incremental learner to use cluster statistics as feedback for actively selecting classes to learn. We have shown that the selected incremental learner (CBCL-PR) is not only state-of-the-art in a pure few-shot class incremental learning setting, but also that the cluster space is valuable to intrinsically motivate the learner to select specific classes. In both Minecraft simulation and real indoor environments, a robot that used cluster statistics for active class selection out-performed uniform batch-learning.

A challenge of any (machine) learner is to gather labeled data for supervised training. We lay the groundwork for more efficient gathering and usage of labeled data, relaxing previous assumptions that have hindered the feasibility of robot learning. As opposed to previous methods in FSCIL, we do not rely on a prescribed class order, nor require training on half the dataset prior to incremental learning. These assumptions are both unrealistic and not applicable to

a robot learning in a new environment. As opposed to previous methods in ACS, we incorporate more current efforts of incremental learning such that computational complexity is more favorable in the long-term (see Appendix).

Future work should build on the merging of active class selection and incremental learning. The most obvious reason is that it is critical to bridge the gap between robot and agent learning. Additionally, there is opportunity to further advance the state-of-art in FSCIL-ACS. For instance, in the context of clustering, a combination of statistics could be used to guide class selection. More broadly, alternative internal measures could be used as feedback for class selection. Regardless, the advantages in combining ACS and FSCIL motivate a new direction for robot learning.

ACKNOWLEDGMENTS

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0197.

REFERENCES

Ronald C Arkin. Motor schema—based mobile robot navigation. *The International journal of robotics research*, 8(4): 92–112, 1989.

Ali Ayub and Carter Fendley. Few-shot continual active learning by a robot. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=35I4narr5A.

Ali Ayub and Alan R. Wagner. Centroid based concept learning for RGB-D indoor scene classification. *British Machine Vision Conference (BMVC)*, 2020a.

Ali Ayub and Alan R. Wagner. What am i allowed to do here?: Online learning of context-specific norms by pepper. In Alan R. Wagner, David Feil-Seifer, Kerstin S. Haring, Silvia Rossi, Thomas Williams, Hongsheng He, and Shuzhi Sam Ge (eds.), *Social Robotics*, pp. 220–231, Cham, 2020b. Springer International Publishing. ISBN 978-3-030-62056-1.

Ali Ayub and Alan R Wagner. Cognitively-inspired model for incremental learning using a few examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 222–223, 2020c.

Ali Ayub and Alan R Wagner. Tell me what this is: few-shot incremental object learning by a robot. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8344–8350. IEEE, 2020d.

Ali Ayub and Alan R. Wagner. EEC: Learning to encode and regenerate images for continual learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Ali Ayub and Alan R. Wagner. CBCL-PR: A cognitively inspired model for class-incremental learning in robotics. *IEEE Transactions on Cognitive and Developmental Systems*, 2023.

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.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, 1992.

Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. Active and incremental learning with weak supervision. *KI-Künstliche Intelligenz*, 34(2):165–180, 2020.

Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *The European Conference on Computer Vision (ECCV)*, September 2018.

M. Dehghan, Z. Zhang, M. Siam, J. Jin, L. Petrich, and M. Jagersand. Online object and task learning via human robot interaction. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2132–2138, May 2019.

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pp. 86–102. Springer, 2020.

Elizabeth Gibney. Could machine learning fuel a reproducibility crisis in science? *Nature*.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 4246–4247. AAAI Press, 2016. ISBN 9781577357704.

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, 2017a.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017b.

Marcus Klasson, Cheng Zhang, and Hedvig Kjellström. A hierarchical grocery store image dataset with visual and semantic labels. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

Donald E Knuth. *Art of computer programming, volume 2: Seminumerical algorithms*. Addison-Wesley Professional, 2014.

Yoram Koren, Johann Borenstein, et al. Potential field methods and their inherent limitations for mobile robot navigation. In *ICRA*, volume 2, pp. 1398–1404, 1991.

Daniel Kottke, Georg Krempl, Marianne Stecklina, Cornelius Styp von Rekowski, Tim Sabsch, Tuan Pham Minh, Matthias Deliano, Myra Spiliopoulou, and Bernhard Sick. Probabilistic active learning for active class selection. *arXiv preprint arXiv:2108.03891*, 2021.

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

Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, Dec 2018.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Yaping Lin, George Vosselman, Yanpeng Cao, and Michael Ying Yang. Active and incremental learning for semantic als point cloud segmentation. *ISPRS journal of photogrammetry and remote sensing*, 169:73–92, 2020.

Rachel Lomasky, Carla E Brodley, Matthew Aernecke, David Walt, and Mark Friedl. Active class selection. In *European Conference on Machine Learning*, pp. 640–647. Springer, 2007.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Michael L. Mack, Bradley C. Love, and Alison R. Preston. Building concepts one episode at a time: The hippocampus and concept formation. *Neuroscience Letters*, 680:31–38, 2018.

T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, Nov 2013.

Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11321–11329, June 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Mengye Ren, Michael L Iuzzolino, Michael C Mozer, and Richard S Zemel. Wandering within a world: Online contextualized few-shot learning. *arXiv preprint arXiv:2007.04546*, 2020.

Burr Settles. Active learning literature survey. 2009.

Yawar Siddiqui, Julien Valentin, and Matthias Niessner. Viewal: Active learning with viewpoint entropy for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. URL /se3/wp-content/uploads/2014/09/WelinderEtal10_CUB-200.pdf,http://www.vision.caltech.edu/visipedia/CUB-200.html.

Dongrui Wu and Thomas D Parsons. Active class selection for arousal classification. In *International Conference on Affective Computing and Intelligent Interaction*, pp. 132–141. Springer, 2011.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3014–3023, June 2021.

Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Dagmar Zeithamova, Margaret L. Schlichting, and Alison R. Preston. The hippocampus and inferential reasoning: building memories to navigate future decisions. *Frontiers in Human Neuroscience*, 6, 2012. doi: 10.3389/fnhum.2012.00070.

Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12455–12464, June 2021.

Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: A python toolbox for class-incremental learning, 2021.

Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9046–9056, 2022.

Kai Zhu, Yang Cao, Wei Zhai, Jie Cheng, and Zheng-Jun Zha. Self-promoted prototype refinement for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6801–6810, 2021.

A Appendix

A.1 Preliminary Study

Overview. Two settings for few-shot class-incremental learning are considered, denoted *traditional* and *pure* FSCIL. In *traditional* FSCIL, as described in Section 2, the learner receives N_b-base classes of full data in the first session $(t = 1)$, then incrementally learns on the remaining data using N-classes per session with k-examples per class (i.e., N-way, k-shot) on subsequent sessions $(t > 1)$. The primary difficulties for learning in this setting include catastrophic forgetting and over-fitting due to class imbalance. In *pure* FSCIL, as introduced here, the learner receives no base classes of full data, but rather incrementally trains with N-way k-shot learning from the first session $(t \geq 1)$. The primary challenges of this setting include catastrophic forgetting and learning without a large portion of the dataset.

Baselines. CBCL-PR was compared with nine other methods: CBCL, iCarL, PODNet, DER, TOPIC, SPPR, Decoupled-DeepEMD, CEC, and FACT. The methods formulated for class-incremental learning (iCarL, PodNet, and DER) were adapted from previous works (Zhou et al., 2021; Douillard et al., 2020) to limit the number of shots. FACT and CEC were re-run from original code for the *pure* FSCIL setting. Note that TOPIC and Decoupled-DeepEMD did not have full code available for reproduction, while SPPR performed significantly worse with any reduction in N_b base classes, so these methods are excluded from *pure* FSCIL comparison.

Data. The Caltech-UCSD Birds 200 (CUB-200) image dataset was used for a preliminary study (Welinder et al., 2010). CUB-200 contains 11,788 images, uniformly distributed over 200 classes. The classes are different species of birds.

Implementation. All methods used Resnet-18 pre-trained with Imagenet as a backbone. The preliminary study used ten random seeds. As in previous work (Tao et al., 2020), *traditional* FSCIL incorporated 100 base classes and 10-way 5-shot incremental learning. The first session $(t = 1)$ trained for 50 epochs with an initial learning rate of 0.1, decreased to 0.01 and 0.001 at 30 and 40 epochs, respectively. Subsequent sessions $(t > 1)$ were trained with a learning rate of 0.01 for 100 epochs. The mini-batch size was 128. For *pure* FSCIL, no base classes were used, allowing 10-way 5-shot incremental learning from the first session. The learning rate was 0.01 for 100 epochs for all sessions $(t \geq 1)$. The mini-batch size was 64.

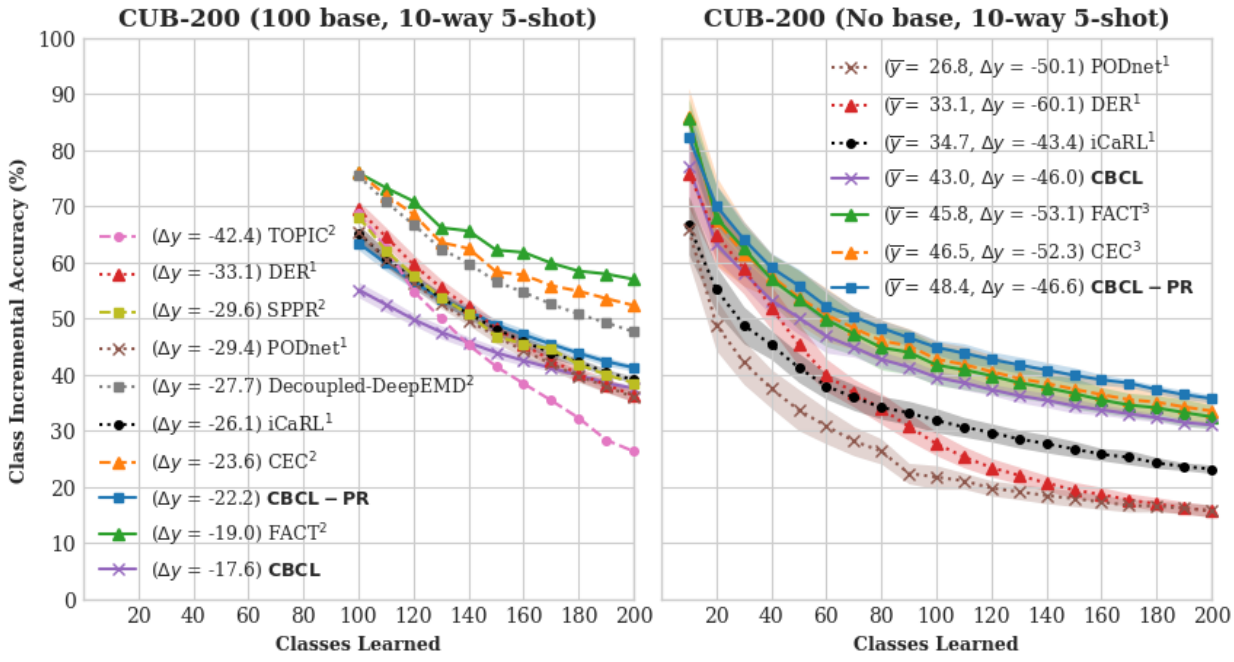

Figure 9: A comparison of *traditional* and *pure* FSCIL on the CUB200 dataset. Note that [1]CIL is modified for FSCIL, [2]results are as reported, and [3]traditional FSCIL is modified for pure FSCIL. Average incremental accuracy ($\overline{y}$) and performance decay (Δy) are indicated. $\pm\sigma$ is plotted as transparent.

Results. The left plot of Figure 9 shows the *traditional* FSCIL setting. In this setting, the usual metric for comparison is performance decay (Δy), a difference between first and last session incremental accuracy. This metric is consistent with the goal of few-shot class-incremental learning to prevent catastrophic forgetting; however, the results are not entirely clear when the lowest performance decay (best case) has a lower incremental accuracy (worst case) over a majority of the test. Attempts have been made to minimize this ambiguity by re-using the first session training or selecting hyper-parameters such that the first session has similar accuracy to baseline methods (Tao et al., 2020). However, hyperparameters and full model architecture are not always shared, such that a major disadvantage of the setting is reproducibility (Gibney); moreover, it is unrealistic for a robot to train on half of the dataset before entering a new environment. Regardless, in *traditional* FSCIL, the best performer in performance decay is CBCL (-17.6%), an improvement of 2.4% over FACT. The newer CBCL-PR ranks third in terms of performance decay (-22.2%). It should be noted that other works have addressed more naturalistic learning paradigms, as opposed to the traditional FSCIL setting. For example Ren et. al define a new learning setting that is not based on episodes of training and testing but rather online, continual learning (Ren et al., 2020).

The right plot of Figure 9 shows the *pure* FSCIL setting. The results are less dependent on the accuracy of the first session, which makes for a fairer overall comparison; furthermore, this setting is more realistic for robots learning in unknown environments. The primary metric for comparison in this setting was average incremental accuracy ($\overline{y}$), which naturally considers performance decay (Δy) along with the rate of decay. The best performer in this setting was CBCL-PR (48.4%), an improvement of 2.3% over CEC.

A.2 CBCL-PR Algorithm Details

CBCL-PR is an updated version of CBCL (Ayub & Wagner, 2020c;a) and it is composed of the following primary components: fixed feature extractor, agg-Var clustering, and the generation of pseudo-exemplars for test prediction.

In each increment, the learner receives the training examples (images) for new classes. Feature vectors of the images are generated using a pre-trained CNN *feature extractor*. With the exception of the preliminary study, the feature extractor was a Resnet-34 model pre-trained with ImageNet.

The learner applies *Agg-Var clustering* on the feature vectors of new classes. Agg-Var clustering, inspired by the concept learning models of the hippocampus and the neocortex (Zeithamova et al., 2012; Mack et al., 2018), enables the learner to discriminate between classes and consolidate these classes into long-term memory. Within the cluster space, each new class is initialized by creating a centroid of a new cluster using the first feature vector in the training set. Next, each additional feature vector x_i^j (i.e., i-th image in class j) is compared to all the existing centroids for class j. If the Euclidean distance between x_i^j and the closest centroid is greater than a pre-defined distance threshold D, a new centroid is created for class j and equated to x_i^j. If the distance is less than D, the closest centroid is updated with a weighted mean: the n-th update ($n > 1$) is calculated by equation (3) below:

$$n\bar{x}_n = x_n + (n-1)\bar{x}_{n-1} \tag{3}$$

Note that prior training data is not needed to calculate a new centroid, as the old centroid $\bar{x}_{n-1}$ and new feature vector $x_n = x_i^j$ are sufficient. This process results in a collection of centroids for the class j, $C^j = \{c_1^j, ..., c_{N_j}^j\}$, where N_j is the number of centroids for class j.

In the update of the cluster space, covariance matrices of new clusters are recorded prior to the discarding of training data. These covariance matrices are used to generate a Gaussian distribution of *pseudo-exemplars* centered on their respective centroid. A linear SVM[1] is trained using pseudo-exemplars of the old classes and feature vectors of the new classes. During testing, feature vectors of the test images are generated using the pre-trained CNN feature extractor and passed through the linear SVM to classify test images based on their feature vectors. This process is shown in Figure 10.

A.3 A* Algorithm Details

In the final experiment, the FIASco algorithm was demonstrated on a Pepper robot navigating an indoor environment. A* path planning was used in order to get from point A (current location) to point B (goal location). The A* algorithm (Hart et al., 1968) is a common search method that incrementally extends the path until a goal state is reached (or until the maximum number of iterations have been attempted, i.e., failure). Specifically, the agent extends the path

[1]For slightly higher accuracy, a shallow neural net was used in the preliminary study. Either classifier can be used.

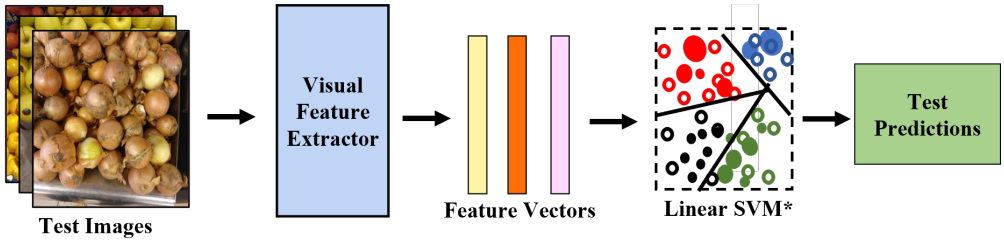

Figure 10: This flow summarizes the testing phase of FIASco, where extracted features from test images are passed through the trained linear SVM to make predictions.

with the next node n minimizing the cost function (equation 4), where $f(n)$ is the total cost, $h(n)$ is the path distance from start, and $g(n)$ is the estimated path distance to goal.

$$f(n) = h(n) + g(n) \tag{4}$$

In order to use this algorithm, the robot first used range sensors to compute a two-dimensional grid map. With an RGB camera, the robot localized objects and placed them on this two-dimensional grid map. Based on which object was most desirable, a goal location was selected. The A* algorithm was used to determine a path, given current and goal locations, and the two-dimensional map of the relative surroundings.

A.4 Computational Cost Details

The batch learner (SVM) and clustering approach (FIASco) both rely on a support vector machine with linear kernel to make test predictions. The data used to train the classifier is the same data type (thus, same dimension), so a comparison of computational complexity depends solely on the number of data points. For the batch learner, this collection of data includes every training instance the learner has collected to a given iteration. Conversely, the clustering approach of FIASco uses incoming data to update the centroids of the cluster space, for which the centroids (fewer than the total training instances) are the data points; however, the pseudo-exemplars should also be considered. Fortunately, the number of pseudo-exemplars are fixed per class and do not grow without bound. These observations can be seen in a plot of training time versus run time, shown in Figure 11.

With the CIFAR-100 and Grocery store datasets, the FIASco only uses 5 and 40 pseudo-exemplars per class, respectively (see Section 4). Additionally, the agent is permitted to explore longer in the CIFAR-100 environment, since the dataset is much larger. These factors explain the trend of computational cost. Initially, the difference between training instances and centroids is minimal, as the agent sees many new objects, creating many new centroids for a majority of incoming data. Furthermore, as the agent has not explored much of the environment, the total number of training points is small, as compared to the total number of pseudo-exemplars. Of course, as time progresses, the impact of the pseudo-exemplars is reduced relative to the total number of data points. Here, clustering would see much benefit in the long term.

A.5 Tabular Results

This section includes the tabular results from the simulation (Section 4) and real-world (Section 5) experiments.

A.6 Potential Field Details

Regarding the forces used in the potential field, ACS methods were used to prioritize the order the classes. Then, the classes were divide by quartile. The top 25 percent of classes received the highest magnitude of attraction, the next 25 percent received the next highest attraction and so on. Table 4 describes the different splits attempted. The actual splits used in the simulation experiment were those described as *mod 1*. Note that $-$ and $+$ reflect attraction and repulsion, respectively, while the integer that follows reflects magnitude.

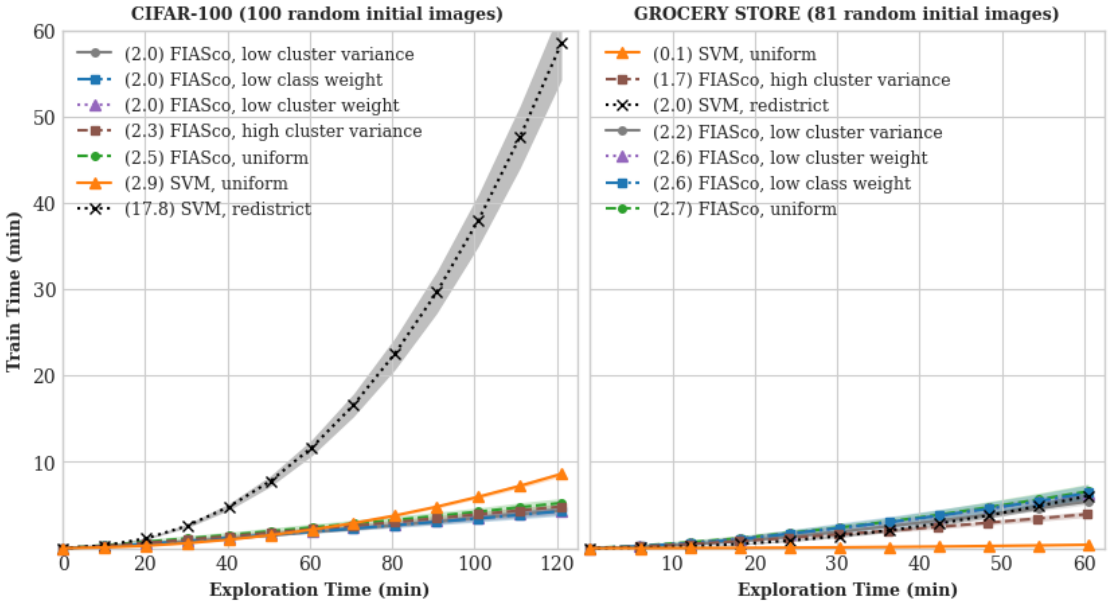

Figure 11: Train time versus exploration time in Minecraft simulation. Note that average incremental train time is indicated and $\pm\sigma$ is plotted as transparent.

Table 1: Test prediction accuracy (%) as a function of run time, as the agent learns in Minecraft with CIFAR-100 data. This data corresponds to the left side of Fig. 7. The highest accuracy at a given increment is indicated with bold font.

Method	**0**	**12**	**24**	**36**	**48**	**60**	**72**	**84**	**96**	**108**	**120 min.**
FIASco, low cluster weight	17.9	29.5	38.4	42.5	44.8	46.3	47.3	47.9	48.4	49.0	49.4
FIASco, high cluster variance	17.8	23.9	27.9	29.6	30.7	32.3	33.9	35.3	35.9	36.4	36.8
FIASco, low cluster variance	17.9	**29.9**	38.0	41.8	43.0	43.8	44.5	45.0	45.3	45.7	45.9
FIASco, low class weight	17.9	29.0	**40.8**	**44.5**	**46.9**	**47.7**	**48.8**	**49.8**	**50.4**	**50.7**	**51.2**
FIASco, uniform	17.9	27.7	34.8	39.3	42.7	45.3	46.8	48.0	49.0	49.9	50.6
SVM, redistrict	17.8	25.7	32.5	37.4	40.0	43.2	45.2	46.8	48.1	49.4	50.4
SVM, uniform	17.8	27.0	34.2	38.0	41.1	43.3	45.6	46.6	47.6	48.8	49.8
CEC, uniform	8.2	9.4	10.3	10.6	11.4	11.8	12.0	12.1	12.6	12.6	12.6
FACT, uniform	8.3	10.5	11.4	11.8	12.3	12.5	12.7	13.0	13.2	13.2	13.3

A.7 Dataset Notes

Reason for datasets. The CIFAR-100 (Krizhevsky et al., 2009) dataset was chosen as a common benchmark in continual learning tasks (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017). The Grocery Store dataset (Klasson et al., 2019) represents an ecologically valid dataset that is very close to a real-world dataset. The authors specify the data was collected from 18 different grocery stores with realistic features such as misplaced objects, varying distances, angles, lighting conditions, etc.

Nature of training data. In both experiments, the data was modified to have a 90:10 stratified train-test split. The test data (10 percent) was used for the evaluation of the classifier at every iteration. In the simulation experiment, the learner selects from Minecraft objects within an observable distance. Each Minecraft object has a 1:1 mapping to classes in the offline dataset, either CIFAR-100 or Grocery Store. Thus, when the agent is near n Minecraft objects, it processes as being near n specific classes and selects a class. The learner receives training instances from the offline dataset corresponding to the selected class. In the real-world experiment, the learner detects real objects using an RGB camera, predicts the classes for which the objects belong with its classifier, and then selects from the predicted classes to train. The learner receives training instances from the offline dataset corresponding to the real object requested (from pointing). When the learner receives training instances from the offline dataset, it either stores the training features (SVM learner) or uses the training features to update the cluster space and ACS methods (FIASco).

Table 2: Test prediction accuracy (%) as a function of run time, as the agent learns in Minecraft with Grocery store data. This data corresponds to the right side of Fig. 7. The highest accuracy at any point is indicated with bold font.

Method	**0**	**6**	**12**	**18**	**24**	**30**	**36**	**42**	**48**	**54**	**60 min.**
FIASco, low cluster weight	30.6	40.4	52.9	59.3	64.6	67.6	70.2	72.3	73.9	75.4	76.5
FIASco, high cluster variance	30.6	36.5	41.3	44.0	47.2	49.2	49.9	51.8	53.0	53.9	54.7
FIASco, low cluster variance	30.6	39.9	52.6	60.2	64.9	68.1	69.9	71.6	72.3	73.0	74.0
FIASco, low class weight	30.6	39.6	52.1	**60.4**	**66.8**	**69.3**	**71.1**	**73.4**	**75.0**	**75.9**	**76.2**
FIASco, uniform	30.6	38.2	45.8	52.5	57.3	60.7	64.4	67.2	69.4	71.6	74.4
SVM, redistrict	30.5	37.3	44.6	52.2	57.3	62.0	64.7	68.0	70.0	72.2	74.0
SVM, uniform	30.5	37.4	44.9	50.8	56.7	61.6	65.4	68.7	71.3	73.0	74.8
CEC, uniform	**52.6**	47.6	51.7	56.0	60.2	62.0	63.6	65.5	66.3	67.4	69.0
FACT, uniform	51.4	**50.3**	**54.0**	57.9	61.1	62.4	63.9	66.2	66.7	67.2	68.0

Table 3: Test prediction accuracy (%) as a function of requested classes, as Pepper learns with Grocery store data. This data corresponds to Fig. 8. The highest accuracy at any point is indicated with bold font.

Method	**0**	**6**	**12**	**18**	**24**	**30**	**36**	**42**	**48**	**54**	**60 requests**
FIASco, low class weight	54.5	**57.2**	**60.2**	**58.2**	**60.2**	**60.9**	61.5	63.9	62.2	66.6	67.9
FIASco, high cluster variance	54.5	56.5	56.9	57.2	59.2	58.9	**62.5**	64.2	**67.6**	**67.2**	**68.2**
FIASco, low cluster variance	54.5	54.2	54.5	55.5	57.5	58.5	59.9	61.5	61.5	63.5	63.2
FIASco, low cluster weight	54.5	52.2	52.8	56.9	59.9	**60.9**	**62.5**	61.9	62.9	62.5	63.2
FIASco, uniform	54.5	54.8	55.2	56.2	59.2	59.9	60.9	63.9	65.2	66.2	66.6
SVM, uniform	54.8	51.5	57.2	57.9	59.5	60.5	62.2	**64.9**	65.6	66.9	67.2

Table 4: A breakdown of attraction splits per quartile of attractive classes.

Magnitude Descriptor	**Q1**	**Q2**	**Q3**	**Q4**
attract + repulse	−20	−10	+10	+20
attract + ignore	−20	−10	00	00
mod 1	−20	−10	+05	+05
mod 2	−20	−10	−05	−05
mod 3	−20	−10	−10	−10
attract only	−20	−20	−20	−20

Future work. Evaluating these methods in a more cluttered or less structured environment could also be an interesting problem. Testing on such cluttered data for ACS has not been explored in most prior works (Lomasky et al., 2007; Wu & Parsons, 2011; Kottke et al., 2021). This application might also require object segmentation or object detection, which is currently out of the scope of this work. We do note that grocery stores are highly structured environments in which products are naturally categorized and organized in a logical manner. Moreover, items are placed on the shelves in a reasonably uncluttered manner.