**Data-Free Class-Incremental Hand Gesture Recognition**

Shubhra Aich¹ † Jesus Ruiz-Santaquiteria¹ † Zhenyu Lu¹ Prachi Garg¹ K J Joseph² Alvaro Fernandez Garcia¹ Vineeth N Balasubramanian² Kenrick Kin³ Chengde Wan³ Necati Cihan Camgoz³ Shugao Ma³ Fernando De la Torre¹

¹Carnegie Mellon University  ²Indian Institute of Technology Hyderabad  ³Meta Reality Labs

*Equal contribution  †Corresponding author: saich@andrew.cmu.edu

---

**Abstract**

This paper investigates data-free class-incremental learning (DFCIL) for hand gesture recognition from 3D skeleton sequences. In this class-incremental learning (CIL) setting, while incrementally registering the new classes, we do not have access to the training samples (i.e. data-free) of the already known classes due to privacy. Existing DFCIL methods primarily focus on various forms of knowledge distillation for model inversion to mitigate catastrophic forgetting. Unlike SOTA methods, we delve deeper into the choice of the best samples for inversion. Inspired by the well-grounded theory of max-margin classification, we find that the best samples tend to lie close to the approximate decision boundary within a reasonable margin. To this end, we propose BOAT-MI – a simple and effective boundary-aware prototypical sampling mechanism for model inversion for DFCIL. Our sampling scheme outperforms SOTA methods significantly on two 3D skeleton gesture datasets, the publicly available SHREC 2017, and EgoGesture3D – which we extract from a publicly available RGBD dataset. Both our codebase and the EgoGesture3D skeleton dataset are publicly available: https://github.com/humansensinglab/dfcil-hgr.

1. Introduction

Humans innately rely on their hands to communicate, feel and interact with their environment. Recent studies have shown that seeing one’s hands tracked in real-time in a virtual environment without controllers is the most compelling method of user engagement in virtual reality (VR) [49]. The latest VR headsets such as Meta Quest [15, 16], and VUZIX [3] leverage high-precision hand-tracking features to render compelling user immersion abilities. The hand-tracking and individual finger-tracking technologies in these platforms are mature enough for natural interactions in the virtual world, and provide a highly immersive “user presence” in virtual environments [29].

While existing VR systems are equipped with promising hand gesture recognition performance, they are designed using a pre-defined set of hand gestures that an end user ought to follow for interaction with the virtual world. However, considering the variations in user preferences across cultures and demographics [1, 2], individual users may also wish to register custom gesture categories and personalize their VR experience. Such a provision has several advantages. As hands play a key role in the recognition and expression of
human emotions [41], registering custom gestures allows users to express themselves better, thus enhancing immersion and user presence in the virtual environment. Studies in psychology show that multi-lingual people tend to use native language-specific gestures [37], and in a VR system where hand gestures are the primary medium of natural interaction, it makes users more comfortable with the system and promotes frequent use. Moreover, such a provision can significantly enhance accessibility and inclusivity for people with disabilities or special needs. Going beyond simple customization to individual preferences, such a continual gesture learning system can also provide technology developers with capabilities to add suites of new gestures for different VR application domains, such as extending a generic gaming VR system for education, rehabilitation or manufacturing.

A gesture recognition model is typically trained on a large-scale dataset of pre-defined (‘base class’) gestures before being deployed on the user’s VR edge device. When a user or developer registers new gesture categories, a naive update of the model based on the user’s new data can result in catastrophic forgetting [32] of the pre-defined gesture categories. The pre-trained model deployed on each user device is often trained on proprietary organizational data or private data that cannot be accessed during re-training in subsequent time steps when a user registers new gestures. Additionally, the user-provided novel gesture training data at each continual step is also private to the user and has to be discarded after adapting the model. Hence, effectively registering new gesture classes to an existing model without access to data pertaining to previous tasks is a significant problem in AR/VR domains. We address this important problem in this work, and aim to build a life-long extensible model, to which a user can register new gesture classes sequentially throughout its lifetime of deployment. Figure 1 illustrates our problem setting.

A common strategy in continual learning methods, which focuses on addressing catastrophic forgetting, is to store a small set of exemplars of previously seen classes and replay them to the model along with novel class data to mitigate forgetting of previous knowledge [40]. Such replay-based methods have been shown to provide state-of-the-art performance across various continual learning approaches in recent surveys [30, 28]. However, adding custom user-specific gestures in a continual manner in AR/VR systems precludes the possibility of storing or using previous class samples due to privacy concerns, hence motivating a data-free class-incremental learning framework. Considering that we cannot access previous task data but have access to the inference model after training each task, model inversion to obtain data impressions of previous tasks becomes a natural choice of solution (as shown in Figure 1). This is referred to as Data-Free Class-Incremental Learning (DFCIL) in recent works [55, 45, 13]. The limited efforts in this problem setting so far [55, 45, 13] largely rely on a knowledge distillation strategy to get the inverted samples and further to regularize the model while fine-tuning to mitigate catastrophic forgetting. We instead set out to answer the following question: How to choose the best samples for model inversion to minimize catastrophic forgetting in data-free class-incremental learning? Besides, while existing efforts are focused on image data, to the best of our knowledge, ours is the first such effort on 3D skeleton-based dynamic hand gesture data in an AR/VR context.

To answer this question, we take inspiration from statistical learning theory, specifically max-margin classification [17, 47]. We follow the notion that samples near the boundary in the feature space (such as support vectors in a Support Vector Machine) are relevant candidates to preserve the decision boundaries among the classes and improve generalization. Therefore, in principle, while performing model inversion, choosing the support vectors close to the decision boundaries should keep the decision boundary of the known classes intact. We hence devise an algorithm first to sample such points in the model’s latent representation space, perform model inversion on such samples and finetune the model on a new task with these inverted samples. In order to choose a diverse set of support vectors, we also consider class prototypes and guide the choice of such samples using these prototypes. We hence call the proposed approach Boundary Aware proTotypical Model Inversion (BOAT-MI). We extensively evaluate the proposed BOAT-MI mechanism for DFCIL on the task of dynamic hand gesture recognition from 3D skeleton data. It is evident from the experimental results that BOAT-MI indeed helps to preserve the decision boundary significantly better than state-of-the-art (SOTA) methods. To summarize, our contributions are as follows:

• We propose a boundary-aware prototypical model inversion (BOAT-MI) strategy for data-free class-incremental learning, which focuses on preserving user privacy in 3D skeleton-based hand gesture recognition systems. In particular, we systematically investigate the choice of sample selection for model inversion, and take inspiration from the theory of max-margin classification in choosing samples near the boundary in the model inversion process. To the best of our knowledge, this is the first such effort on 3D skeleton data.

• To this end, we also contribute a large-scale 3D skeleton gesture recognition dataset, whereby EgoGesture3D is re-annotated for 3D skeleton keypoints from the original RGB-D EgoGesture dataset [57].

• We comprehensively evaluate the proposed BOAT-MI method for our target task of class-incremental dynamic hand gesture recognition on 3D skeleton data. The experimental results demonstrate significant improvements over SOTA methods for the proposed continual learning setup, particularly tailored to real-world settings after de-
ployment on users’ devices. We hope this will serve as a new benchmark for continual learning research in hand gesture recognition.

2. Related Works

2.1. 3D Skeleton-Based Hand Gesture Recognition

3D skeleton-sequences are being increasingly used as inputs for action and gesture recognition tasks due to their robustness to background interference, illumination and viewpoint changes as well as reduced training complexity as compared to RGB-D inputs. Several deep learning based methods have been used to model skeleton-based hand gesture sequences. Authors in [35] propose a two-stage CNN and LSTM framework to learn spatial and temporal joint features respectively. More recent approaches for action recognition employ Graph Convolutional Networks [54, 10] to model a spatio-temporal graph of the skeleton sequence. STST [58] and DSTA-Net [42] design specialized transformer blocks to learn spatial and temporal features in a decoupled manner. DG-STA [9] is a fully connected graph transformer. It applies multi-head spatial attention over the spatial skeleton graph, followed by multi-head temporal attention on the graph’s temporal edges. We use DG-STA as our architectural backbone due to its simplicity of construction, feasibility of model inversion and code availability.

2.2. Continual Gesture and Activity Recognition

Recently, there has been an active interest in making real-world gesture and activity-based human-robot interaction systems continually learn new user classes. However, most existing works focus on sensory data from accelerometers, ambient sensors, or surface electromyographic (sEMG) signals [5, 19, 6]. Authors in [4] propose a lifelong adaptive learning framework that processes motion sensor-based HAR datasets in a task-free continual fashion using experience replay and continual prototype adaptation.

More recently, [23] proposes an exemplar memory enhancement strategy for class-incremental learning (CIL) of static, single-image gestures such as in NUS II and Sign Language MNIST [39]. CIL has also been explored for action recognition in videos [38, 48]. Both these works address video continual learning using regularization and episodic memory replay based methods. Authors in CatNet [50] are the first to attempt class-incremental hand gesture recognition. They use the EgoGesture dataset [57] and propose a two-stream RGB and depth framework which replays previous class exemplars based on the iCARL [40] algorithm. Our work differs from these works in two key aspects. Firstly, incremental learning has not yet been explored for skeleton-based dynamic hand gesture recognition. Secondly, unlike these methods which majorly rely on replay of stored exemplars from previous classes to mitigate forgetting, we circumvent user privacy, data security, and scalability concerns by proposing a novel data-free class-incremental framework.

2.3. Data-Free Class-Incremental Learning

Rehearsal based methods store a small set of exemplars [40, 18] or features [20, 52] of previously seen classes and replay them along with new class data to mitigate forgetting. [36, 43] generate synthetic images for replay, but the generator needs to be stored through the lifetime of the model’s deployment and the synthesized images may contain sensitive user information [34]. Another category of approaches such as Learning Without Forgetting (LwF) [24], MCIL [26], and LwF:MC [40] use knowledge distillation as a data regularization technique, so as to ensure that the new model makes predictions similar to a frozen copy of the old (teacher) model for old classes. Typically, several methods [18, 51, 8, 22, 53] including iCARL [40] combine knowledge distillation with exemplar replay of previous classes. In order to transfer knowledge without exemplars, recent works introduce data-free class-incremental learning where the previous step inference model can be inverted to obtain old class images which in turn are used for replay along with new class data. DeepDream [33] was the first such method which optimized noise into images using image prior regularization. DeepInversion [55] improved DeepDream’s inverted image quality by proposing feature distillation based on Batch Normalization statistics. The ABD [45] work further improved model inversion image synthesis by analyzing the cause of poor performance of inverted images in DeepInversion. They propose a modified cross-entropy loss and importance-weighted feature distillation regularization to improve DFCIL. Most recently, [13] introduces relational knowledge distillation to guide the new model to learn new-class representations that are better compatible with old class representations. Also, authors in [27] attempt DFCIL for Person Re-identification. While all above stated methods use a student-teacher knowledge distillation approach to invert images, we do not use knowledge distillation and instead propose a SVM-based prototypical boundary sampling algorithm for model inversion.

3. BOAT-MI: Our Methodology

3.1. Problem Formulation

In class-incremental learning, a model sequentially learns a series of $N$ incoming tasks $\{T_1, T_2, ..., T_i, ..., T_N\}$ as and when they become available. Each task consists of data from a new set of classes, such that classes across all tasks is non-overlapping. In a new task $T_i$, $C_i$ new classes will be added to the existing model containing $C_k$ existing classes from all previous steps. In the training phase for task (or step) $i$, the model only has access to the training data for $C_i$ new classes. In data-free class-incremental learning, we put an additional constraint that the model does not have access to the original samples corresponding to already old classes, $C_k$. After learning task $i$, during inference for a given test sample, the model classifier predicts for all classes that the
continual learner has seen till now, i.e. $y \in C_O \cup C_N$. Unlike task-incremental learning where task identity is provided at inference time, this is the harder class-incremental setting where task-identity is not provided at inference.

Figure 2: (Synthetic dataset) Visualization of the old class $C_O$ samples in the (penultimate) feature space $\mathbb{R}^2$ of the model. Individual classes are indicated by different colors. For the preliminary proof of concept, the tiny red triangles on the elliptical boundaries refer to the boundary samples used for model inversion. Note that the exact estimation of the decision boundaries is noisy even for this highly simplified classification task. Such noise is bypassed with a margin [17, 47] for the ellipses here. Best viewed in color.

### 3.2. Preliminary Proof of Concept

First, we validate our hypothesis on the effectiveness of the samples close to the decision boundaries compared to other samples. In this regard, we generate a simple, low-dimensional synthetic dataset (supplementary material, Figure 1 – 6D gaussian blobs, 10 classes, 200 samples per class). We take half of the data for training and the other half for evaluation. To emulate the DFCIL setup, we consider the old set $C_O$, and the other 5 new $C_N$. We employ a simple 3 layer MLP as the classifier. Training this MLP on all 10 classes ($C_O \cup C_N$) provides the oracle accuracy of 99.6%. The baseline accuracy, that is, training on the 5 old classes in $C_O$ is 100%. Next, as shown in Figure 2, we invert the elliptical boundary samples from the feature space (penultimate layer of MLP) – intentionally set to $\mathbb{R}^2$ for the ease of visualization and sampling, the dimensionality of which is sufficient for the synthetic dataset we have. This way, we generate 50 inverted samples per old class in $C_O$, merge them with the real sample set for the new classes (100 per class) in $C_N$, and finetune the baseline model. After finetuning, we obtain 90.8% accuracy over the complete test set (including both old and new classes), with 83.0% for the old samples, and 98.6% for the new samples. In comparison, the random sampling strategy provides only about 58.6% total accuracy with 21.2% and 96.0% for the old and new test samples, respectively. This preliminary results lead us to investigate the expressive power of the boundary samples further for model inversion on our target task of 3D gesture recognition.

On top of that, Figure 2 demonstrates the noisy nature of the exact estimation of the decision boundaries regardless of the simplicity of the problem domain and dataset. This observation is aligned with the notion of margin in max-margin classification literature [17, 47]. In fact, bypassing this noise with a reasonable margin is a key to the success of our proposed algorithm as described later.

Following the preliminary proof of concept above, we now present our boundary-aware prototypical model inversion (BOAT-MI) mechanism in detail. The high-level approach is depicted in Algorithm 1. The complete algorithm is provided in the supplementary material.

### 3.3. Methodology Description

First, we define the components necessary for the depiction of our approach (Algorithm 1). Our BOAT-MI algorithm requires a learnable feature extractor (like a deep network) $F$ and the SVM classifier $H$ operating on the feature extractor (i.e. the penultimate layer). Also, for a particular class $c$, if $X \in \mathbb{R}^{n \times k}$ denotes all the training samples ($n$) belonging to class $c$, we get the class-prototypical mean and covariance from the penultimate feature representation as $\mu = E[F(X)]$ and $\Sigma = E[(F(X) - \mu)(F(X) - \mu)^T]$, respectively.

Now, the core technical question that we attempt to address here is to decide on which samples we should invert for maximal gain. According to the theory of support vector machine (SVM) [47], support vectors (SVs) are essentially

---

**Algorithm 1: Boundary-Aware Prototypical Model Inversion**

**Function** Main($F$, $H$, $c$, $\mu$, $\Sigma$, $m$, $\alpha_f$, $\alpha$, $it$, $\varepsilon$, $n$, $\delta$):

- # $F$, $H$: feature extractor; SVM classifier
- # $c$: index of class to invert
- # $\mu$, $\Sigma$: prototypical mean, covariance for class $c$
- # $m$, $\alpha_f$, $\alpha$: momentum; forward/reverse LR
- # $it$, $\varepsilon$: max iterations; tolerance threshold
- # $n$, $\delta$: max # of samples per class; margin
- # Use inverted mean for initialization later

$\mathbf{x}_\mu \leftarrow \text{Invert}(F, \mu, m, \alpha, it, \varepsilon, \text{NULL})$

$\mathbf{y}_s \leftarrow []$ # initialize feature list for inversion

Extend the list with support vectors for class $c$

$\mathbf{y}_s += H.\text{support_vectors}[c]$

# Get principal directions of prototypical
# covariance and their unique linear combinations
# as prototypical features for inversion

$\mathbf{p}_s \leftarrow \text{GetProtoDirections}(\Sigma, n)$

$\mathbf{y}_s += \text{GetProtoFeatures}(\mathbf{p}_s, \mu, c, \alpha_f, \delta)$

$n \leftarrow \text{len}(\mathbf{y}_s)$ # Final # of samples to invert

$\mathbf{x}_s \leftarrow [\mathbf{x}_\mu] * n$ # initialize input list for inversion

for $i \leftarrow 1 : n$ do

$\mathbf{x}_s[i] \leftarrow \text{Invert}(F, \mathbf{y}_s[i], m, \alpha, \text{iter, } \varepsilon, \mathbf{x}_\mu)$

return $\mathbf{x}_s$

**End Function**
We call these pseudo SVs evolving from the linear combination with a corresponding boundary-aware target feature using the vanilla classification problem, standalone SVs are deemed to be insufficient for the class-incremental setup. In Figure 4, we show that proto SVs for the incremental classification problem. Classes 1 (red) and 2 (green) are old to SVM a priori. So, their SVs (on the red and green lines) are able to preserve the corresponding decision boundaries when the decision is to be made between these two old classes only. However, as the new class (3) appears incrementally, old SVs for class 1 fail to safeguard its insiders.

Class integrity more than standard SVs in most cases with the best results coming from having both on board for model inversion. Therefore, the high-level steps for BOAT-MI for a single old class are as follows:

1. Get old class prototype (mean $\mu$ and covariance $\Sigma$).
2. Get the support vectors (SVs) from the SVM classifier $\mathcal{H}$ learned on top of the (deep) features extracted from $\mathcal{F}$.
3. Get the significant principal directions (PDs) of the prototypical covariance $\Sigma$.
4. Generate more dummy PDs with linear combination of existing PDs.
5. Cast rays following the PDs from the prototypical mean $\mu$ to reach the boundary for proto-SV generation.
6. (Figure 3, Step 1) Run model inversion on SVs + Proto-SVs from the above step to generate support inputs (SIs).
7. (Figure 3, Step 2) Fine-tune the model with SIs representing old classes $C_{O}$ and new samples from the new or incremental classes $C_{N}$.

Also, ray casting from the class-prototypical mean $\mu$ is done iteratively with a learning rate $\alpha_{f}$ until the ray hits the boundary. Also, a margin $\delta$, normalized with respect to the distance between the mean $\mu$ and the boundary vector (Figure 5), is used to avoid noise inherent in the boundary estimation process. In other words, the proto-SV feature is taken to be the on the margin, which is $\delta$ inside the boundary. The feature inversion procedure (feature $\rightarrow$ input) deviates from the vanilla implementation [12] in three aspects. First, we initialize the input tensor with the inverted class-prototypical mean. Second, we employ the normalized $L_{2}$ function as the distance metric. Third, we replace the vanilla gradient descent with its momentum-based counterpart [46]. All these modifications are empirically found to expedite convergence.
3.4. Architecture and Loss Functions

As mentioned in Section 2.1, we employ the DG-STA [9] architecture as our 3D skeleton-based gesture recognition backbone. Regarding the error criterion, the usage of a single prototypical mean \( \mu \) and covariance matrix \( \Sigma \) per class requires the reinforcement of compact clustering in the learnable feature space of the deep architecture. To ensure compactness, we use the supervised contrastive learning loss SupCon [21] (Equation 1), which we empirically find to be a better alternative to the standard cross-entropy loss.

\[
L_{CL} = \frac{-1}{2N^i} \sum \frac{1}{2N^i} \log \left( \frac{1}{\tilde{N}^i} \sum_{j=1}^{2N} \delta_{i,j} \cdot \exp \left( \frac{z_i \cdot z_j}{\tau} \right) \right)
\]

\[
L_{sup} = \sum_{j=1}^{2N} \delta_{i,j} \cdot \log \left( \frac{1}{\tilde{N}^i} \sum_{j=1}^{2N} \delta_{i,j} \cdot \exp \left( \frac{z_i \cdot z_j}{\tau} \right) \right)
\]

where \( N^i \) is the number of minibatch samples belonging to the same class, \( \tilde{y}^i \) as the sample of index \( i, z \) is the feature vector and \( \tau \) is a temperature parameter that controls the degree of concentration or dispersion of distributions.

4. Experiments

4.1. Datasets

We employ two datasets for gesture recognition – one with ego-centric and another with second-person views.

SHREC-2017: SHREC-2017 [44] comprises of 14 coarse and fine-grained gestures performed with both, one finger (index finger and thumb) and all the fingers captured with the short-range Intel Real Sense Depth camera. The hand skeleton contains 22 keypoints – 1 \( \times \) wrist, 1 \( \times \) palm, and 4 \( \times \) each finger \( \times \) 5 fingers. Each sample contains 1920 and 840 samples from a disjoint set of 20 and 8 subjects, respectively. Since the test set labels are hidden, we recast the validation set as our test set and divide the training set into train/val splits with roughly 20 samples per class for validation. This updated split configuration is public with the codebase.

EgoGesture3D: Table 1 shows a comparison of existing 3D skeleton-based hand gesture recognition datasets. All these datasets have been collected from Intel Real Sense Depth or RGB-D cameras. The FPHA [14] dataset consists of first-person daily hand actions interacting with 26 object categories. Ideally, in addition to SHREC-2017, we want to set up a continual learning benchmark on a dataset with a large number of classes for easy incremental splits. It can be seen from the table that existing 3D skeleton hand gesture datasets are small scale datasets with less number of classes. Hence, we consider extracting 3D skeleton annotations from an existing large-scale RGB-D hand gesture recognition dataset. Even though RGB-D datasets like Jester [31] and the more recent LD-ConGR [25] have more gesture sequences as compared to EgoGesture [57], the number of classes in EgoGesture are significantly higher. Moreover, SHREC and DHG are second person view datasets and it is useful to also study continual learning performance for egocentric views. Hence, we construct a 3D skeleton version of the EgoGesture dataset and call it EgoGesture3D.

EgoGesture57, EgoGesture contains both single and dual-handed gestures. Visualizations are provided in the supplementary.

4.2. Experimental Setup

We demonstrate results for 7 class-incremental tasks across two dataset benchmarks. Task 0 represents the accuracy of the model learned on the base classes (8 for SHREC-2017 and 59 for EgoGesture3D). Each of the next tasks (Task 1 → 6) adds 1 (SHREC-2017) and 4 (EgoGesture3D) classes at a time. In a VR/AR context, the user may want to add a single-class at a time when they realise the need for a customized gesture. Also, SHREC-2017 has a relatively small number of classes (only 14) for a continual learning benchmark. Hence, studying performance degradation at every stage (after adding a single class) is more relevant to our setting for SHREC-2017. From Table 2, it is evident that the forgetting turns out to be severe after the first few tasks (i.e. from Task 4 and onward). In EgoGesture, we study adding 4 classes at a time to contrast with SHREC and see the affect of adding multiple gestures at every stage. We report individual task performance rather than an averaging over 5 continual tasks as done in [45]. As discussed later, comparing with such granularity helps to get a better sense of the methods while benchmarking.

4.3. Evaluation Metrics

One of the most fundamental concerns while developing incremental/continual learning systems is the imbalance between the information preserved for the old tasks/classes and the new ones. For example, for DFCIL, the naive approaches

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classes</th>
<th>Sequences</th>
<th>Label</th>
<th>FPV</th>
<th>Data Type</th>
<th>Cat. Seg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHREC 2017 [44]</td>
<td>14/28</td>
<td>2800</td>
<td>✓</td>
<td>✓</td>
<td>Depth, 3D-skeleton</td>
<td>✓</td>
</tr>
<tr>
<td>FPHA [14]</td>
<td>45</td>
<td>1175</td>
<td>✓</td>
<td>✓</td>
<td>RGB-D, 3D skeleton</td>
<td>✓</td>
</tr>
<tr>
<td>EgoGesture3D</td>
<td>83</td>
<td>24161</td>
<td>✓</td>
<td>✓</td>
<td>RGB-D, Depth, 3D skeleton</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1: A comparison of different 3D skeleton-based dynamic hand gesture recognition datasets. Label Cat. refers to gesture classification, Seg. refers to temporal segmentation.
iterate over the new class samples without accessing any known class samples. This causes learning the new classes with significantly high accuracy while the known class information being washed away. Consequently, there is a huge imbalance between the accuracy of the newly learned classes in that incremental stage and that of the already known classes until the previous stage. We find this issue prevalent in all the DFCIL methods to date. To our knowledge, there is no such methods (Table 2 and 3) for our continual gesture recognition benchmarks. To cast light on this issue, note that all the SOTA methods used for benchmarking were originally developed for an image classification problem. Hence, to adopt these methods for gesture recognition, we individually tuned their hyperparameters to be the optimal one for our setup. Therefore, to our understanding, this dominance of feature extraction over the recent methods is not due to the sub-optimal hyperparameter setup. Instead, we hypothesize this behavior can be credited towards the shift in problem domain from images to 3D gestures. This leads to a possibility that our BOAT-MI mechanism may fall short in the image learning steps, we present the global accuracy and our proposed forgetting measure ($G \uparrow$ and $IFM \downarrow$). Our proposed approach achieves significantly higher global accuracy in each stage there with 13% and 6.8% improvements on the most difficult stage (Task 6) for SHREC-2017 (Table 2) and EgoGesture3D (Table 3), respectively, over the next best methods. Moreover, this improvement on global accuracy comes with a significantly lower instantaneous forgetting – 12.1% and 4.1% lower $IFM$ than the second best method for SHREC-2017 (Table 2) and EgoGesture3D (Table 3), respectively. Overall, BOAT-MI outperforms the SOTA methods in all aspects across the board.

**Surprisingly good results with feature extraction** [24]: Unlike the SOTA DFCIL methods [13, 45, 55], following LwF [24], we decided to include the simple baselines (base, finetuning, and feature extraction) for comparison as well. Surprisingly, Feature Extraction excels some of the SOTA methods (Table 2 and 3) for our continual gesture recognition benchmarks. Learning feature extraction over the recent methods is not due to the sub-optimal hyperparameter setup. Instead, we hypothesize this behavior can be credited towards the shift in problem domain from images to 3D gestures. This leads to a possibility that our BOAT-MI mechanism may fall short in the image domain, the investigation of which is beyond the scope of this work and our focus as the title of this paper indicates.

### 4.4. SOTA Comparison

Table 2 and 3 provide the comparison of our approach with the SOTA DFCIL methods. As discussed in the experimental setup in Section 4.2, Task 0 represents the base classification accuracy, and Task 1 → 6 indicate the steps of continual data-free class registration results. For each of the continual learning steps, we present the global accuracy and our proposed forgetting measure ($G \uparrow$ and $IFM \downarrow$).

<table>
<thead>
<tr>
<th>Method</th>
<th>Task 0</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
<th>Task 6</th>
<th>Mean (Task 1 → 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base [24]</td>
<td>59.2</td>
<td>10.4</td>
<td>15.9</td>
<td>66.5</td>
<td>9.3</td>
<td>82.2</td>
<td>7.7</td>
<td>84.4</td>
</tr>
<tr>
<td>Feature extraction [24]</td>
<td>69.8</td>
<td>14.5</td>
<td>60.8</td>
<td>17.2</td>
<td>51.5</td>
<td>67.3</td>
<td>11.3</td>
<td>78.7</td>
</tr>
<tr>
<td>LwF [24]</td>
<td>79.9</td>
<td>53.7</td>
<td>69.3</td>
<td>32.6</td>
<td>26.2</td>
<td>91.1</td>
<td>19.1</td>
<td>56.5</td>
</tr>
<tr>
<td>LwF.MC [45]</td>
<td>80.5</td>
<td>56.0</td>
<td>70.9</td>
<td>23.5</td>
<td>12.6</td>
<td>78.1</td>
<td>12.6</td>
<td>64.6</td>
</tr>
<tr>
<td>Deepinversion [55]</td>
<td>82.9</td>
<td>39.4</td>
<td>74.3</td>
<td>13.9</td>
<td>4.7</td>
<td>79.9</td>
<td>24.4</td>
<td>59.2</td>
</tr>
<tr>
<td>ABD [45]</td>
<td>83.7</td>
<td>4.5</td>
<td>70.9</td>
<td>3.7</td>
<td>4.5</td>
<td>83.7</td>
<td>1.8</td>
<td>54.8</td>
</tr>
<tr>
<td>R-DFCIL [13]</td>
<td>83.7</td>
<td>4.5</td>
<td>70.4</td>
<td>4.3</td>
<td>4.5</td>
<td>83.7</td>
<td>1.8</td>
<td>54.8</td>
</tr>
<tr>
<td>BOAT-MI (Ours)</td>
<td>83.7</td>
<td>4.5</td>
<td>70.4</td>
<td>4.3</td>
<td>4.5</td>
<td>83.7</td>
<td>1.8</td>
<td>54.8</td>
</tr>
</tbody>
</table>

**Table 2: Results (average of 3 runs with different class order) for class-incremental learning over six tasks in SHREC-2017.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Task 0</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
<th>Task 6</th>
<th>Mean (Task 1 → 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base [24]</td>
<td>60.4</td>
<td>13.5</td>
<td>18.9</td>
<td>63.3</td>
<td>9.3</td>
<td>82.4</td>
<td>8.0</td>
<td>84.3</td>
</tr>
<tr>
<td>Feature extraction [24]</td>
<td>69.8</td>
<td>14.5</td>
<td>60.8</td>
<td>17.2</td>
<td>51.5</td>
<td>30.4</td>
<td>46.4</td>
<td>33.7</td>
</tr>
<tr>
<td>LwF [24]</td>
<td>79.9</td>
<td>4.3</td>
<td>65.9</td>
<td>14.9</td>
<td>53.1</td>
<td>29.5</td>
<td>49.5</td>
<td>32.8</td>
</tr>
<tr>
<td>LwF.MC [45]</td>
<td>80.5</td>
<td>56.0</td>
<td>70.9</td>
<td>23.5</td>
<td>12.6</td>
<td>78.1</td>
<td>12.6</td>
<td>64.6</td>
</tr>
<tr>
<td>Deepinversion [55]</td>
<td>82.9</td>
<td>39.4</td>
<td>74.3</td>
<td>13.9</td>
<td>4.7</td>
<td>79.9</td>
<td>24.4</td>
<td>59.2</td>
</tr>
<tr>
<td>ABD [45]</td>
<td>83.7</td>
<td>4.5</td>
<td>70.4</td>
<td>4.3</td>
<td>4.5</td>
<td>83.7</td>
<td>1.8</td>
<td>54.8</td>
</tr>
<tr>
<td>R-DFCIL [13]</td>
<td>83.7</td>
<td>4.5</td>
<td>70.4</td>
<td>4.3</td>
<td>4.5</td>
<td>83.7</td>
<td>1.8</td>
<td>54.8</td>
</tr>
<tr>
<td>BOAT-MI (Ours)</td>
<td>83.7</td>
<td>4.5</td>
<td>70.4</td>
<td>4.3</td>
<td>4.5</td>
<td>83.7</td>
<td>1.8</td>
<td>54.8</td>
</tr>
</tbody>
</table>

**Table 3: Results (average of 3 runs with different class order) for class-incremental learning over six task in EgoGesture3D.**
Table 4: Comparison (average of 3 runs with different class order) of normalized margins with Prototypical MI on SHREC-2017.

<table>
<thead>
<tr>
<th>Normalized Margin (δ)</th>
<th>Task 0</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
<th>Task 6</th>
<th>Mean (Task 1 → 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G↑ IFM↓ G↑ IFM↓ G↑ IFM↓</td>
<td>G↑ IFM↓ G↑ IFM↓ G↑ IFM↓</td>
<td>G↑ IFM↓ G↑ IFM↓ G↑ IFM↓</td>
<td>G↑ IFM↓ G↑ IFM↓ G↑ IFM↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.6</td>
<td>0.8</td>
<td>1.2</td>
<td>1.6</td>
<td>2.0</td>
<td>2.4</td>
<td>2.8</td>
<td>3.2</td>
</tr>
<tr>
<td>0.1</td>
<td>0.7</td>
<td>1.0</td>
<td>1.3</td>
<td>1.7</td>
<td>2.0</td>
<td>2.3</td>
<td>2.6</td>
<td>3.0</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8</td>
<td>1.1</td>
<td>1.4</td>
<td>1.8</td>
<td>2.1</td>
<td>2.4</td>
<td>2.7</td>
<td>3.1</td>
</tr>
<tr>
<td>0.5</td>
<td>0.9</td>
<td>1.2</td>
<td>1.5</td>
<td>1.9</td>
<td>2.2</td>
<td>2.5</td>
<td>2.8</td>
<td>3.2</td>
</tr>
<tr>
<td>Random</td>
<td>1.0</td>
<td>1.3</td>
<td>1.6</td>
<td>1.9</td>
<td>2.2</td>
<td>2.5</td>
<td>2.8</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Table 5: Comparison (average of 3 runs with different class order) of different boundary samples for MI on SHREC-2017.

<table>
<thead>
<tr>
<th>Sampler</th>
<th>Task 0</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
<th>Task 6</th>
<th>Mean (Task 1 → 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SV</td>
<td>0.6</td>
<td>0.8</td>
<td>1.2</td>
<td>1.6</td>
<td>2.0</td>
<td>2.4</td>
<td>2.8</td>
<td>3.2</td>
</tr>
<tr>
<td>Proto</td>
<td>0.7</td>
<td>1.0</td>
<td>1.3</td>
<td>1.7</td>
<td>2.0</td>
<td>2.3</td>
<td>2.6</td>
<td>3.0</td>
</tr>
<tr>
<td>SV+Proto</td>
<td>0.8</td>
<td>1.1</td>
<td>1.4</td>
<td>1.8</td>
<td>2.1</td>
<td>2.4</td>
<td>2.7</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Figure 5: An illustration of the comparison provided in Table 4. The blue ellipse indicates the hypothetical decision boundary. To show that samples close to the decision boundary with a margin are better than insiders or random candidates, we test the performance of inverted features taken at predefined normalized margins (Table 4, δ = {0.0, 0.1, 0.2, 0.5, 0.8}) along the same ray. Here δ = 0.0 (blue) is the exact boundary sample and δ = 0.8 (purple) has the highest margin – just 0.2 (normalized) away from the class prototypical mean.

4.5. Ablation Studies

Effect of margin on boundary-aware sampling: First, we attempt to empirically validate the claim that features sampled close to the boundary with a margin prevents catastrophic forgetting better in the DFCIL setup. An illustration of our validation scheme is shown in Figure 5. The class prototypical mean in the feature space is indicated by the black dot at the center of the ellipse and the blue ellipse represents the approximated decision boundary for a particular class. For each sample on the boundary (blue dot), we cast a ray from the mean to the boundary sample, and select features at different distances normalized by the mean ↔ boundary point distance. This way we ensure fairness of the sample selection process for the comparison of different normalized margins (δ = {0.0, 0.1, 0.2, 0.5, 0.8}). The results are reported in Table 4. During the early stages of continual learning (Task 1 → 3), samples with different margins are all in the same ballpark. However, it is evident that samples near the boundary (δ = {0.1, 0.2}) provide better global accuracies than the insiders δ = 0.8 as the task gets harder (Task 5 and 6). Sampling on the exact boundary (δ = 0) is worse than the strategies with a reasonable margin (δ = {0.1, 0.2}) due to the noisy approximation of the decision boundary. The same goes for forgetting IFM. In addition, we also include the results with the features randomly sampled at different distances which performs evidently poorer than the strict margin based samplers.

Comparison of different sampling strategies: As described in Algorithm 1 and Section 3.1, our BOAT-MI approach contains two kinds of boundary-aware samples – standard support vectors retrieved from the support vector machine and the prototypical boundary features. We provide a comparison of these two types of samplers and their combinations in Table 5. The comparatively poorer performance of the standard SVs can be explained with their inadequacy in a class-incremental learning setup (Figure 4). However, combining both the standard SVs and their prototypical counterparts for inversion brings the best on the table.

5. Conclusion and Future Work

In this paper, we explore the problem of DFCIL for gesture recognition from 3D skeleton sequences. Compared to the main line of development for the SOTA DFCIL methods mostly geared towards various knowledge distillation for better performance, we made a detour to instead figure out the best set of features for model inversion for DFCIL in the domain of gesture recognition. Our intuitive and theoretically aligned feature selection mechanism, proposed in this paper, excels the SOTA methods by a significant margin in all stages of continual learning including 13% and 6.8% improvements on the last and most difficult step for the two 3D skeleton datasets. Although evaluated on our focus area of 3D gesture recognition, we believe the domain agnostic nature of the proposed boundary-aware selection mechanism will be impactful in the future development of DFCIL frameworks, in general.
References


