Data-Free Class-Incremental Hand Gesture Recognition

Shubhra Aich[1]* † Jesus Ruiz-Santaquiteria[1]* Zhenyu Lu[1] Prachi Garg[1] K J Joseph[2]
Alvaro Fernandez Garcia[1] Vineeth N Balasubramanian[2] Kenrick Kin[3] Chengde Wan[3]
Necati Cihan Camgoz[3] Shugao Ma[3] Fernando De la Torre[1]
[1]Carnegie Mellon University [2]Indian Institute of Technology Hyderabad [3]Meta Reality Labs
*Equal contribution †Corresponding author: saich@andrew.cmu.edu

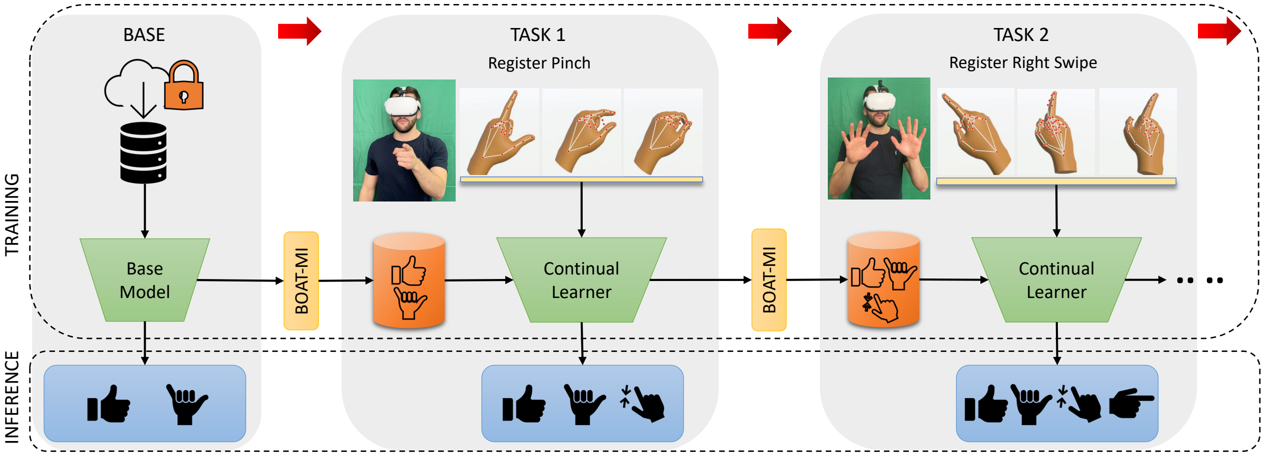

Figure 1: **Data-Free Class-Incremental Learning for Hand Gesture Recognition.** When interacting with gestures in a virtual reality (VR) environment, the user may want to add new gestures to customize their VR experience. Given that the training data of the gesture model is proprietary and inaccessible for future tasks, finetuning the existing model on a new class can lead to *catastrophic forgetting* of all old classes. Our proposed BOundary Aware ProTotypical Model Inversion (BOAT-MI) enables the model to continually adapt to new gestures while retaining previous knowledge.

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 analysing 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 $\{\mathcal{T}_1, \mathcal{T}_2, .., \mathcal{T}_i, .., \mathcal{T}_N\}$ 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 $\mathcal{T}_i$, $\mathcal{C}_\mathcal{U}$ new classes will be added to the existing model containing $\mathcal{C}_\mathcal{K}$ 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 $\mathcal{C}_\mathcal{U}$ 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, $\mathcal{C}_\mathcal{K}$. 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. $y \in \mathcal{C}_{\mathcal{K}} \cup \mathcal{C}_{\mathcal{U}}$. 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.

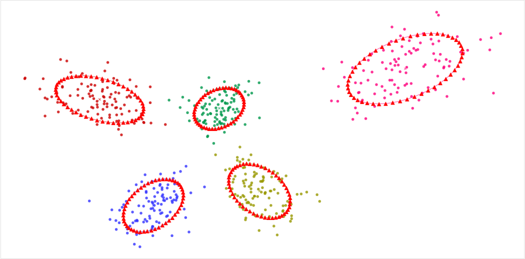

Figure 2: **(Synthetic dataset)** Visualization of the old class $\mathcal{C}_{\mathcal{O}}$ samples in the (penultimate) feature space $\mathbb{R}^2$ of the model. Individual classes are indicated by different colors. For the preliminary proof of concept, the tiny red triangles on the elliptical boundaries 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, 5 classes are considered the old set $\mathcal{C}_{\mathcal{O}}$, and the other 5 new $\mathcal{C}_{\mathcal{N}}$. We employ a simple 3 layer MLP as the classifier. Training this MLP on all 10 classes ($\mathcal{C}_{\mathcal{O}} \cup \mathcal{C}_{\mathcal{N}}$) provides the oracle accuracy of 99.6%. The baseline accuracy, that is, training on the 5 old classes in $\mathcal{C}_{\mathcal{O}}$ 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 $\mathcal{C}_{\mathcal{O}}$, merge them with the real sample set for the new classes (100 per class) in $\mathcal{C}_{\mathcal{N}}$, 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.

```
Algorithm 1: Boundary-Aware Prototypical Model Inversion
Function Main ($\mathcal{F}$, $\mathcal{H}$, $c$, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $m$, $\alpha_f$, $\alpha$, it, $\varepsilon$, $n$, $\delta$):
    # $\mathcal{F}$; $\mathcal{H}$: feature extractor; SVM classifier
    # $c$: index of class to invert
    # $\boldsymbol{\mu}$; $\boldsymbol{\Sigma}$: 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
    $\mathbf{x}_\mu \leftarrow$ Invert ($\mathcal{F}$, $\boldsymbol{\mu}$, $m$, $\alpha$, iter, $\varepsilon$, NULL)
    ys $\leftarrow$ [ ] # initialize feature list for inversion
    # Extend the list with support vectors for class $c$
    ys += $\mathcal{H}$.support_vectors[$c$]
    # Get principal directions of prototypical
    # covariance and their unique linear combinations
    # as prototypical features for inversion
    ps $\leftarrow$ GetProtoDirections($\boldsymbol{\Sigma}$, $n$)
    ys += GetProtoFeatures(ps, $\boldsymbol{\mu}$, $c$, $\alpha_f$, $\delta$)
    $n \leftarrow$ len(ys) # Final # of samples to invert
    xs $\leftarrow$ [$\mathbf{x}_\mu$] $*\ n$ # initialize input list for inversion
    for $i \leftarrow 1$ : $n$ do
        xs[i] $\leftarrow$ Invert ($\mathcal{F}$, ys[i], $m$, $\alpha$, iter, $\varepsilon$, $\mathbf{x}_\mu$)
    return xs
End Function
```

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) $\mathcal{F}$ and the SVM classifier $\mathcal{H}$ operating on the feature extractor (i.e. the penultimate layer). Also, for a particular class c, if $\boldsymbol{X} \in \mathbb{R}^{n \times k}$ denotes all the training samples (n) belonging to class c, we get the class-prototypical mean and covariance from the penultimate feature representation as $\boldsymbol{\mu} = \mathbb{E}[\mathcal{F}(\boldsymbol{X})]$ and $\boldsymbol{\Sigma} = \mathbb{E}[(\mathcal{F}(\boldsymbol{X}) - \boldsymbol{\mu})(\mathcal{F}(\boldsymbol{X}) - \boldsymbol{\mu})^T]$, 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

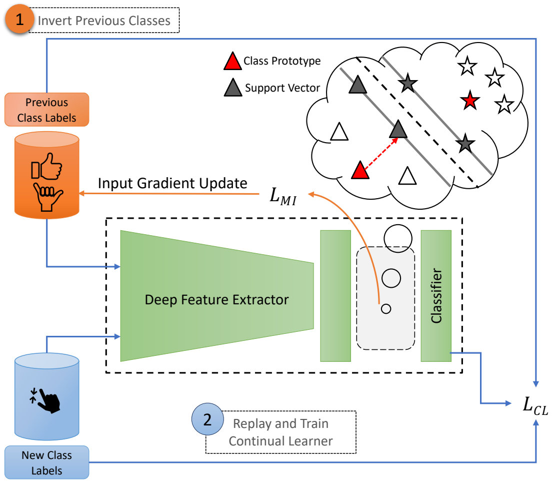

Figure 3: Our proposed BOAT-MI pipeline. **1. Model Inversion:** (a) Initialize the input with inverted class-prototypical means; (b) Match the feature representation of the input with a corresponding boundary-aware target feature using the $\mathcal{L}_{MI}(=\mathcal{L}_2)$ norm; finally, (c) the gradient of the norm is used to update the input. Note that model is frozen during this step. **2. Continual model adaptation:** The previous step model is updated with the new class samples and replay of inverted previous class samples using $\mathcal{L}_{CL}$.

the samples on the margin of the decision boundaries. The SVM classifier makes the decision as a linear combination of a handful of SVs. Therefore, SVs are on the list of potential candidate samples. Even though SVs are sufficient for a vanilla classification problem, standalone SVs are deemed to be insufficient for the class-incremental setup. In Figure 4, classes 1 and 2 are old to SVM, and so, their SVs are able to protect the corresponding decision boundaries. However, as the new class (with index 3) appears incrementally, old SVs (class 1 in this figure) are unable to preserve them while learning the new class features.

Based upon this observation, in the feature space, for old classes $\mathcal{C}_{\mathcal{O}}$, SV-like samples must be present in all significant directions to safeguard against potential invasions from the incremental classes $\mathcal{C}_{\mathcal{N}}$ in the future. Generating pseudo SVs by shooting rays in different directions from the class-prototypical mean $\boldsymbol{\mu}$ is not difficult. However, given the high dimensionality of the feature space ($\mathbb{R}^{128}$ in our case), covering all these directions and their linear combinations would be combinatorially explosive. This curse of dimensionality can be avoided by considering only the principal directions of the class-prototypical covariance $\boldsymbol{\Sigma}$ in the feature space. Moreover, for the old classes, the choice of principal components makes sense since these indicate the axes of expansion over the learned class-specific features in the feature space. We call these pseudo SVs evolving from the linear combination of class-prototypical mean and variance *proto SVs*. Later in Section 4, we show that proto SVs are able to preserve the 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:

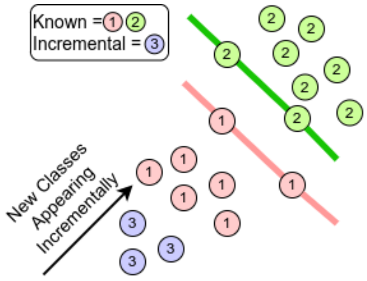

Figure 4: A hypothetical example describing the inadequacy of the standard SVs for the incremental classification problem. Classes 1 (red) and 2 (green) are old to SVM *a priori*. So, their SVs (on the red and green lines) are able to preserve the corresponding decision boundaries when the decision is to be made between these two old classes only. However, as the new class (3) appears incrementally, old SVs for class 1 fails to safeguard its insiders.

1. Get old class prototype (mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$).
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 $\boldsymbol{\Sigma}$.
4. Generate more dummy PDs with linear combination of existing PDs.
5. Cast rays following the PDs from the prototypical mean $\boldsymbol{\mu}$ 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) Finetune the model with SIs representing old classes $\mathcal{C}_{\mathcal{O}}$ and new samples from the new or incremental classes $\mathcal{C}_{\mathcal{N}}$.

Also, ray casting from the class-prototypical mean $\boldsymbol{\mu}$ is done iteratively with a learning rate α_f until the ray hits the boundary. Also, a margin δ, normalized with respect to the distance between the mean $\boldsymbol{\mu}$ 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 δ inside the boundary. The feature inversion procedure (feature → 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 $\mathcal{L}_2$ 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 $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ 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.

$$\mathcal{L}_{CL} = \sum_{i=1}^{2N} \frac{-1}{2N_{\tilde{\boldsymbol{y}}_i} - 1} \mathcal{L}_i^{sup}$$

$$\mathcal{L}_i^{sup} = \sum_{\substack{j=1 \\ i \neq j}}^{2N} \mathbf{1}_{\tilde{\boldsymbol{y}}_i = \tilde{\boldsymbol{y}}_j} \cdot \log \frac{\exp\left(\frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_j}{\tau}\right)}{\sum_{k=1}^{2N} \mathbf{1}_{i \neq k} \cdot \exp\left(\frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_k}{\tau}\right)} \tag{1}$$

where $N_{\tilde{\boldsymbol{y}}_i}$ is the number of minibatch samples belonging to the same class, $\tilde{\boldsymbol{y}}_i$, as the sample of index i, $\boldsymbol{z}$ is the feature vector and τ 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 [1] – 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 × wrist, 1 × palm, and 4 × each finger × 5 fingers. The original training and validation sets contains 1980 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.

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.

Dataset	Classes	Sequences	Label		FPV	Data Type
			Cat.	Seg.		
SHREC 2017 [44]	14/28	2800	✓		×	Depth, 3D skeleton
DHG [11]	14/28	2800	✓		×	Depth, 3D skeleton
FPHA [14]	45	1175	✓		✓	RGB-D, 3D skeleton
EgoGesture3D	83	24161	✓	✓	✓	RGB-D, Depth, 3D skeleton

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**. EgoGesture [57, 7] contains 14416 training, 4768 validation, and 4977 test samples encompassing 83 gesture categories. The original RGB-D and depth dataset was collected from 50 subjects (18 females and 32 males), in 4 indoor and 2 outdoor scenes. Unlike other datasets, EgoGesture covers variation in background, illumination, clutter and also captures people performing gestures when they are walking. It also has temporal segmentation annotations of individual gesture sequences from continuous videos. We employ MediaPipe [56] for 3D skeleton extraction. Unlike SHREC-2017, 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

[1] All the data were accessed and processed at and by CMU.

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

Method	Task 0	Task 1		Task 2		Task 3		Task 4		Task 5		Task 6		Mean (Task 1 → 6)	
Oracle	89.4														
		G↑	IFM ↓	G↑	IFM↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM↓	G↑	IFM↓
Base [24]	90.5	78.3	7.8	53.3	29.6	30.0	53.7	27.9	56.0	12.2	78.3	11.9	78.6	35.6	50.7
Fine tuning [24]		78.6	6.0	57.4	26.2	34.2	48.8	34.0	48.6	16.4	71.8	16.1	71.9	39.5	45.5
Feature extraction [24]		79.9	9.3	69.3	12.6	62.3	19.1	56.5	24.1	50.6	24.3	45.1	32.4	60.6	20.3
LwF [24]		79.8	9.3	64.1	20.2	31.9	50.2	29.1	53.1	13.1	76.8	11.5	79.3	38.2	48.2
LwF.MC [45]		56.0	10.9	32.6	39.5	21.8	58.5	19.3	61.6	16.7	53.3	16.1	45.4	27.1	44.9
DeepInversion [55]		79.9	4.3	65.9	14.9	53.1	29.5	49.5	32.8	34.2	47.7	32.1	49.7	52.5	29.8
ABD [45]		78.8	4.0	64.6	12.6	54.3	20.3	53.2	24.6	46.1	20.0	40.4	23.3	56.2	17.5
R-DFCIL [13]		78.7	**3.3**	65.5	**4.5**	54.4	20.5	49.8	26.9	41.5	25.0	38.6	33.3	54.8	18.9
BOAT-MI (Ours)		**83.7**	4.5	**76.0**	7.3	**71.4**	**7.0**	**69.4**	**13.3**	**64.1**	**9.5**	**58.1**	**11.2**	**70.5**	**8.8**

Table 3: Results (average of 3 runs with different class order) for class-incremental learning over six task in EgoGesture3D.

Method	Task 0	Task 1		Task 2		Task 3		Task 4		Task 5		Task 6		Mean (Task 1 → 6)	
Oracle	75.8														
		G↑	IFM ↓	G↑	IFM↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM↓	G↑	IFM↓
Base [24]	78.1	60.4	13.5	18.9	63.3	9.3	82.4	8.0	84.3	5.9	88.5	5.9	88.3	18.1	70.0
Fine tuning [24]		59.2	10.4	15.9	66.5	9.3	82.2	7.7	84.4	5.2	89.7	5.0	90.1	17.1	70.6
Feature extraction [24]		69.8	14.5	60.8	17.2	51.5	30.4	46.4	33.7	41.2	39.4	36.8	42.9	51.1	29.7
LwF [24]		69.1	0.1	43.0	27.3	18.8	67.3	11.3	78.7	6.5	87.5	6.3	87.7	25.8	58.1
LwF.MC [45]		36.8	9.8	24.2	41.3	17.0	64.6	12.8	73.3	10.2	78.3	9.7	80.2	18.4	57.9
DeepInversion [55]		68.1	14.1	44.3	32.2	24.7	59.2	16.2	71.2	11.6	78.8	10.0	81.0	29.1	56.1
ABD [45]		68.8	14.8	61.1	17.1	54.3	26.2	49.0	31.1	43.2	37.5	39.0	40.6	52.6	27.9
R-DFCIL [13]		70.3	**3.0**	61.4	**4.1**	53.2	**14.9**	46.1	17.8	39.2	35.0	35.2	36.3	50.9	18.5
BOAT-MI (Ours)		**75.3**	4.6	**70.9**	7.3	**64.0**	15.1	**58.6**	**16.3**	**52.3**	**26.8**	**45.8**	**32.2**	**61.2**	**17.1**

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 metric to capture this imbalance explicitly. Therefore, in this paper, we address this concern with a new metric called ***Instantaneous Forgetting Measure (IFM)*** = $\frac{|L-G|}{L+G} \times 100$. Here, L is the (local) accuracy only over the incremental classes that first appeared at a continual learning step, and G is the (global) accuracy over all the classes encountered by the model up until that stage including the incremental classes. IFM is a percentage metric with 0% and 100% indicating perfect balance $(G = L)$ and pure imbalance (either $G = 0$ or $L = 0$), respectively. Compared to the vanilla difference $(L - G)$, which is linear in both G and L, IFM penalizes with higher weight when one of G or L is significantly lower than the other. We present the global accuracy G and IFM in all the tables here ($G \uparrow$ and $IFM \downarrow$). The up and down arrows indicate the notions of accuracy (*higher is better*) and error (*lower is better*), respectively.

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 $1 \rightarrow 6$ 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 $/IFM \downarrow$). 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. 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 domain, the investigation of which is beyond the scope of this work and our focus as the title of this paper indicates.

Table 4: Comparison (average of 3 runs with different class order) of normalized margins with Prototypical MI on SHREC-2017.

Normalized Margin (δ)	Task 0	Task 1		Task 2		Task 3		Task 4		Task 5		Task 6		Mean (Task 1 → 6)	
		G↑	IFM ↓	G↑	IFM↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM↓	G↑	IFM↓
0.0	91.7	82.3	6.6	75.2	8.8	69.6	12.5	68.0	16.5	58.9	21.2	53.4	25.2	67.9	15.1
0.1		82.5	5.2	74.1	8.2	69.6	9.3	68.4	14.0	**63.6**	**9.8**	56.2	16.7	69.1	**10.5**
0.2		82.7	5.1	74.7	8.1	70.3	8.1	69.2	14.0	63.3	13.5	**56.5**	**15.8**	69.5	**10.5**
0.5		83.7	4.8	77.5	7.6	72.0	8.0	70.3	15.1	61.2	16.0	56.3	21.0	**70.2**	12.1
0.8		84.8	5.1	78.3	9.8	72.5	10.5	69.5	16.4	60.7	17.3	50.1	29.8	69.3	14.8
Random		84.1	6.2	77.5	10.1	69.0	15.4	65.2	18.7	56.0	23.8	44.4	37.2	66.0	18.6

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

Sampler	Task 0	Task 1		Task 2		Task 3		Task 4		Task 5		Task 6		Mean (Task 1 → 6)	
		G↑	IFM ↓	G↑	IFM↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM↓	G↑	IFM↓
SV	92.0	81.7	7.4	74.9	11.5	63.2	18.6	59.7	24.0	50.0	28.9	39.0	43.2	61.4	22.3
Proto		82.7	5.1	74.7	8.1	70.3	8.1	69.2	14.0	63.3	13.5	56.5	15.8	69.5	10.8
SV+Proto		83.7	4.5	76.0	7.3	71.4	7.0	69.4	13.3	64.1	9.5	**58.1**	**11.2**	**70.5**	**8.8**

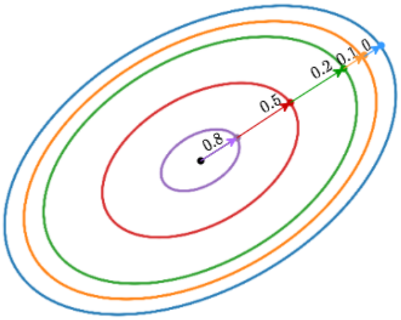

Figure 5: An illustration of the comparison provided in Table 4. The blue ellipse indicates the hypothetical decision boundary. To show that samples close to the decision boundary with a margin are better than insiders or random candidates, we test the performance of inverted features taken at predefined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

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 ($\delta = \{0.1, 0.2\}$) provide better global accuracies than the insiders $\delta = 0.8$ as the task gets harder (Task 5 and 6). Sampling on the exact boundary ($\delta = 0$) is worse than the strategies with a reasonable margin ($\delta = \{0.1, 0.2\}$) 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

[1] Dartmouth Folklore Archive – Hand Gestures. `https://journeys.dartmouth.edu/folklorearchive/fall-2019/hand-gestures-across-cultures`.

[2] Dimensions of Body Language – Chapter 5. `https://westsidetoastmasters.com/resources/book_of_body_language/chap5.html`.

[3] VUZIX Smart Glasses. `https://www.vuzix.com/pages/smart-glasses`.

[4] Rebecca Adaimi and Edison Thomaz. Lifelong adaptive machine learning for sensor-based human activity recognition using prototypical networks. *Sensors*, 2022.

[5] Alessandro Avi, Andrea Albanese, and Davide Brunelli. Incremental online learning algorithms comparison for gesture and visual smart sensors. In *IJCNN*, 2022.

[6] Alessio Burrello, Marcello Zanghieri, Cristian Sarti, Leonardo Ravaglia, Simone Benatti, and Luca Benini. Tackling time-variability in semg-based gesture recognition with on-device incremental learning and temporal convolutional networks. In *2021 IEEE Sensors Applications Symposium*, 2021.

[7] Congqi Cao, Yifan Zhang, Yi Wu, Hanqing Lu, and Jian Cheng. Egocentric gesture recognition using recurrent 3d convolutional neural networks with spatiotemporal transformer modules. In *ICCV*, 2017.

[8] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018.

[9] Yuxiao Chen, Long Zhao, Xi Peng, Jianbo Yuan, and Dimitris Metaxas. Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. In *BMVC*, 2019.

[10] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. In *CVPR*, 2020.

[11] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre. Skeleton-based dynamic hand gesture recognition. In *CVPRW*, 2016.

[12] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM Conference on Computer and Communications Security*, 2015.

[13] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *ECCV*. Springer, 2022.

[14] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *CVPR*, 2018.

[15] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, et al. Megatrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM ToG*, 2020.

[16] Shangchen Han, Po-chen Wu, Yubo Zhang, Beibei Liu, Linguang Zhang, Zheng Wang, Weiguang Si, Peizhao Zhang, Yujun Cai, Tomas Hodan, et al. Umetrack: Unified multi-view end-to-end hand tracking for vr. In *SIGGRAPH Asia*, 2022.

[17] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009.

[18] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, 2019.

[19] Saurav Jha, Martin Schiemer, and Juan Ye. Continual learning in human activity recognition: an empirical analysis of regularization. *arXiv preprint arXiv:2007.03032*, 2020.

[20] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.

[21] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *NeurIPS*, 2020.

[22] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *ICCV*, 2019.

[23] Mingxue Li, Yang Cong, Yuyang Liu, and Gan Sun. Class-incremental gesture recognition learning with out-of-distribution detection. In *IROS*, 2022.

[24] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE TPAMI*, 2018.

[25] Dan Liu, Libo Zhang, and Yanjun Wu. Ld-congr: A large rgb-d video dataset for long-distance continuous gesture recognition. In *CVPR*, 2022.

[26] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, 2020.

[27] Yichen Lu, Mei Wang, and Weihong Deng. Augmented geometric distillation for data-free incremental person reid. In *CVPR*, 2022.

[28] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 2022.

[29] Katerina Mania and Alan Chalmers. The effects of levels of immersion on memory and presence in virtual environments: A reality centered approach. *Cyberpsychology & behavior*, 2001.

[30] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE TPAMI*, 2022.

[31] Joanna Materzynska, Guillaume Berger, Ingo Bax, and Roland Memisevic. The jester dataset: A large-scale video dataset of human gestures. In *CVPRW*, 2019.

[32] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Elsevier, 1989.

[33] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. `https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`, 2015.

[34] Vaishnavh Nagarajan, Colin Raffel, and Ian J Goodfellow. Theoretical insights into memorization in gans. In *NeurIPS Workshop*, 2018.

[35] Juan C Nunez, Raul Cabido, Juan J Pantrigo, Antonio S Montemayor, and Jose F Velez. Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. *Pattern Recognition*, 2018.

[36] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*, 2019.

[37] Şeyda Özçalışkan, Ché Lucero, and Susan Goldin-Meadow. Is seeing gesture necessary to gesture like a native speaker? *Psychological Science*, 2016.

[38] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *ICCV*, 2021.

[39] Pramod Kumar Pisharady, Prahlad Vadakkepat, and Ai Poh Loh. Attention based detection and recognition of hand postures against complex backgrounds. *IJCV*, 2013.

[40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[41] Paddy Ross and Tessa Flack. Removing hand form information specifically impairs emotion recognition for fearful and angry body stimuli. *Perception*, 2020.

[42] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Decoupled spatial-temporal attention network for skeleton-based action-gesture recognition. In *ACCV*, 2020.

[43] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *NeurIPS*, 2017.

[44] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, J. Guerry, B. Le Saux, and D. Filliat. 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. In *Eurographics Workshop on 3D Object Retrieval*, 2017.

[45] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *ICCV*, 2021.

[46] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[47] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[48] Andrés Villa, Kumail Alhamoud, Victor Escorcia, Fabian Caba, Juan León Alcázar, and Bernard Ghanem. vclimb: A novel video class incremental learning benchmark. In *CVPR*, 2022.

[49] Jan-Niklas Voigt-Antons, Tanja Kojic, Danish Ali, and Sebastian Möller. Influence of hand tracking as a way of interaction in virtual reality on user experience. In *QoMEX*, 2020.

[50] Zhengwei Wang, Qi She, Tejo Chalasani, and Aljosa Smolic. Catnet: Class incremental 3d convnets for lifelong egocentric gesture recognition. In *CVPRW*, 2020.

[51] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019.

[52] Ye Xiang, Ying Fu, Pan Ji, and Hua Huang. Incremental learning using conditional adversarial networks. In *ICCV*, 2019.

[53] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *ACM Multimedia*, 2014.

[54] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018.

[55] Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *CVPR*, 2020.

[56] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[57] Yifan Zhang, Congqi Cao, Jian Cheng, and Hanqing Lu. Egogesture: A new dataset and benchmark for egocentric hand gesture recognition. *IEEE TMM*, 2018.

[58] Yuhan Zhang, Bo Wu, Wen Li, Lixin Duan, and Chuang Gan. Stst: Spatial-temporal specialized transformer for skeleton-based action recognition. In *ACM Multimedia*, 2021.

(Supplementary) Data-Free Class-Incremental Hand Gesture Recognition

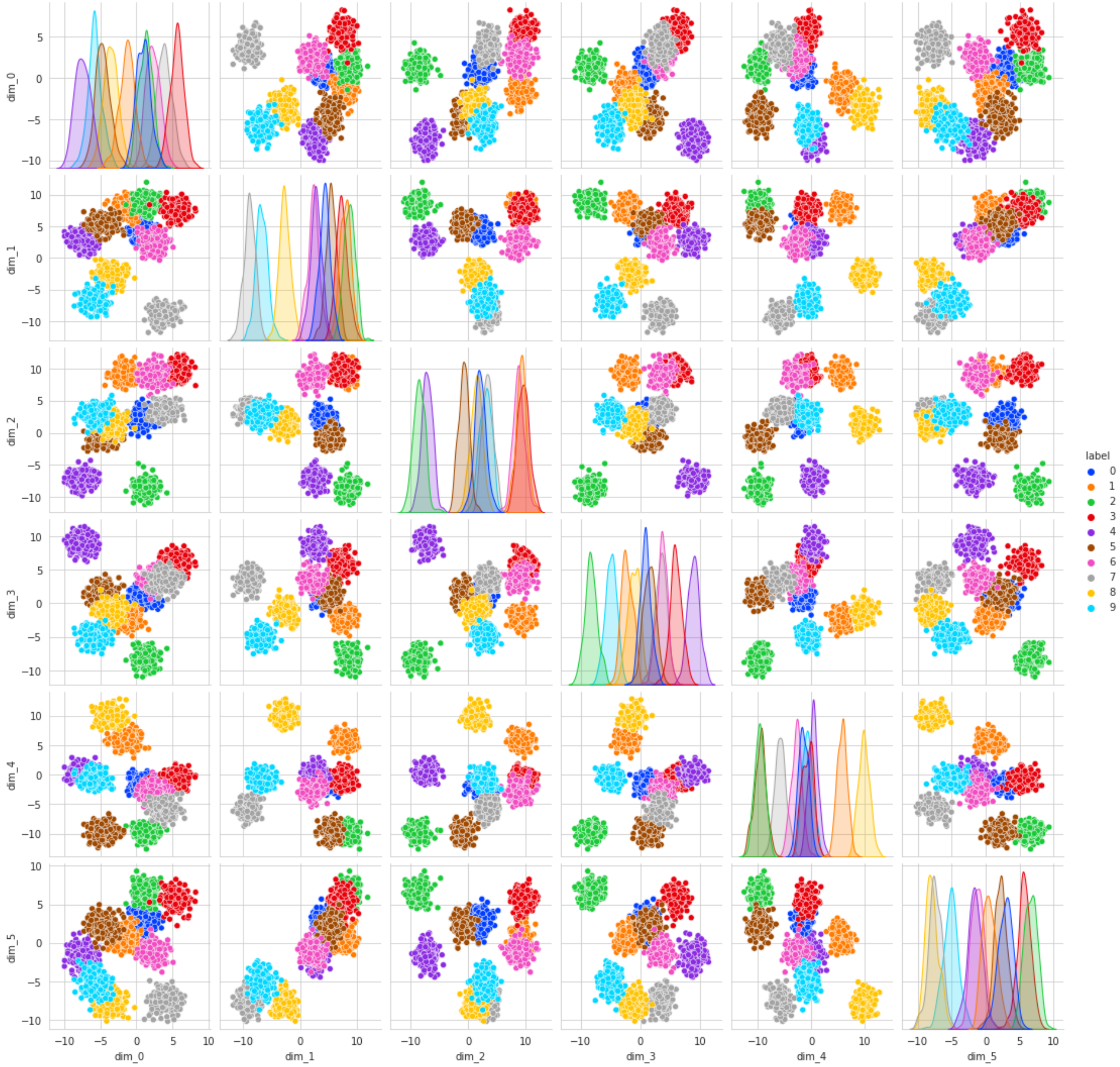

Figure 1: The pair plot of the synthetic dataset (6D Gaussian blobs) used for the preliminary proof of concept in Section 3 (BOAT-MI: Our Methodology) of the main paper. Different colors indicate different class indices (labels on the right). For the DFCIL setup, the first 5 classes (label $0 \to 4$) are considered the old set $\mathcal{C}_{\mathcal{O}}$, and the other 5 (label $5 \to 9$) new $\mathcal{C}_{\mathcal{N}}$.

Algorithm: Boundary-Aware Prototypical Model Inversion (Part 1)

```
Function Main ($\mathcal{F}$, $\mathcal{H}$, $c$, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $m$, $\alpha_f$, $\alpha$, $it$, $\varepsilon$, $n$, $\delta$):
    # $\mathcal{F}; \mathcal{H}$: feature extractor; SVM classifier
    # $c$: index of class to invert
    # $\boldsymbol{\mu}; \boldsymbol{\Sigma}$: 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
    $\mathbf{x}_\mu \leftarrow$ Invert ($\mathcal{F}$, $\boldsymbol{\mu}$, $m$, $\alpha$, $iter$, $\varepsilon$, $NULL$)
    $\mathbf{ys} \leftarrow [\,]$ # initialize feature list for inversion
    # Extend the list with support vectors for class $c$
    $\mathbf{ys} \mathrel{+}= \mathcal{H}$.support_vectors$[c]$
    # Get principal directions of prototypical
    # covariance and their unique linear combinations
    # as prototypical features for inversion
    $\mathbf{ps} \leftarrow$ GetProtoDirections$(\boldsymbol{\Sigma}, n)$
    $\mathbf{ys} \mathrel{+}=$ GetProtoFeatures$(\mathbf{ps}, \boldsymbol{\mu}, c, \alpha_f, \delta)$
    $n \leftarrow$ len$(\mathbf{ys})$ # Final # of samples to invert
    $\mathbf{xs} \leftarrow [\mathbf{x}_\mu] * n$ # initialize input list for inversion
    for $i \leftarrow 1 \; : \; n$ do
        $\mathbf{xs}[\mathbf{i}] \leftarrow$ Invert ($\mathcal{F}$, $\mathbf{ys}[\mathbf{i}]$, $m$, $\alpha$, $iter$, $\varepsilon$, $\mathbf{x}_\mu$)
    end
    return $\mathbf{xs}$
End Function
Function GetProtoDirections ($\boldsymbol{\Sigma}$, $n$):
    # $\boldsymbol{\Sigma}$: class prototypical covariance
    # $n$: maximum # of samples allowed
    # Get principal directions retaining 95% variance
    $\mathbf{ps} \leftarrow$ PCA$(\boldsymbol{\Sigma}, 0.95)$
    $\mathbf{ps} \mathrel{+}= (-\mathbf{ps})$ # extend with negative directions
    if $ClampNumSamples(\mathbf{ps}, n)$ then
        return $\mathbf{ps}$
    end
    $n \leftarrow n -$ len$(\mathbf{ps})$ # update max # of samples
    $ord = 2$ # start from adding 2nd order interactions
    while $TRUE$ do
        # add unique linear interactions of order $ord$
        # $p_{new} = \mathbf{ps}[i] + \mathbf{ps}[j]$ for $ord = 2$
        # $p_{new} = \mathbf{ps}[i] + \mathbf{ps}[j] + \mathbf{ps}[k]$ for $ord = 3$
        $\mathbf{ps\_l} \leftarrow$ LinearInteractions$(\mathbf{ps}, ord)$
        # normalize to get the unit vectors
        $\mathbf{ps\_l} \leftarrow$ Normalize$(\mathbf{ps\_l})$
        if $ClampNumSamples(\mathbf{ps\_l}, n)$ then
            $\mathbf{ps} \mathrel{+}= \mathbf{ps\_l}$ # list extension
            return $\mathbf{ps}$
        end
        $\mathbf{ps}$.append$(\mathbf{ps\_l})$
        $n \leftarrow n -$ len$(\mathbf{ps\_l})$ # update max # of samples
        $ord \leftarrow ord + 1$ # increment order
    end
End Function
```

Algorithm: Boundary-Aware Prototypical Model Inversion (Part 2)

```
Function Invert ($\mathcal{F}$, $\mathbf{y}$, $m$, $\alpha$, $iter$, $\varepsilon$, $\mathbf{x_0}$):
    # $\mathcal{F}; \mathbf{y}$: feature extractor; target feature
    # $m; \alpha$: momentum; learning rate
    # iter; $\varepsilon$: max # of iterations; tolerance threshold
    # $\mathbf{x_0}$: input initialization, can be NULL
    if $\mathbf{x_0} == NULL$ then
        $\mathbf{x} \leftarrow$ input prediction tensor (zero-initialized)
    else
        $\mathbf{x} \leftarrow \mathbf{x_0}$
    end
    $\mathbf{v} \leftarrow$ gradient update tensor (zero-initialized)
    for $i \leftarrow 1 : iter$ do
        $\hat{\mathbf{y}} \leftarrow \mathcal{F}(\mathbf{x})$  # predict feature
        $\mathcal{L} \leftarrow ||\mathbf{y} - \hat{\mathbf{y}}||_2 / ||\mathbf{y}||_2$  # distance norm
        if $\mathcal{L} < \varepsilon$ then
            break
        end
        $\mathbf{v} \leftarrow m\,\mathbf{v} + \nabla_{\mathbf{x}}\mathcal{L}$  # gradient tensor update
        $\mathbf{x} \leftarrow \mathbf{x} - \alpha\,\mathbf{v}$  # gradient descent
    end
    return $\mathbf{x}$
End Function
Function GetProtoFeatures ($\mathbf{ps}, \boldsymbol{\mu}, c, \alpha_f, \delta$):
    # $\mathbf{ps}$: list of principal directions and combinations
    # $c, \boldsymbol{\mu}$: class index and class prototypical mean
    # $\alpha_f, \delta$: forward learning rate and margin
    $\mathbf{ys} \leftarrow [\,]$ # initialize feature list for inversion
    for $i \leftarrow 1 \; : \; len(\mathbf{ps})$ do
        $\mathbf{f} \leftarrow \boldsymbol{\mu}$
        for $k \leftarrow 1 \; : iter$ do
            $\mathbf{f} \leftarrow \mathbf{f} + \alpha_f\;\mathbf{ps}[\mathbf{i}]$
            if $Classify(\mathbf{f}) \neq c$ then
                $\mathbf{f} \leftarrow (1-\delta)\,\mathbf{f} + \delta\,\boldsymbol{\mu}$
                ASSERT Classify $(\mathbf{f}) == c$
            end
        end
        $\mathbf{ys}$.append$(\mathbf{f})$
    end
    return $\mathbf{ys}$
End Function
Function ClampNumSamples ($\mathbf{ps}$, $n$):
    # $\mathbf{ps}$: list of samples to clamp
    # $n$: maximum # of samples allowed
    if $n > len(\mathbf{ps})$ then
        return FALSE # not clamped
    end
    # randomly select $n$ samples in-place
    $\mathbf{ps} \leftarrow$ RandomSelect$(\mathbf{ps})$
    return TRUE # clamped
End Function
```

Table 1: Results with BatchNorm (average of 3 runs with different class order) for DFCIL over six tasks in SHREC-2017.

Method	Task 0	Task 1		Task 2		Task 3		Task 4		Task 5		Task 6		Mean (Task 1 → 6)	
Oracle (DGSTA-BN)	90.3														
		G↑	IFM ↓	G↑	IFM↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM↓	G↑	IFM↓
DeepInversion-BN [6]	92.8	81.4	7.0	62.6	18.2	52.9	29.1	46.3	35.6	34.8	48.1	31.5	51.3	51.6	31.6
ABD-BN [4]		81.7	1.9	72.2	2.0	65.9	10.7	57.3	14.8	51.7	16.3	42.7	25.2	61.9	11.8
R-DFCIL-BN [3]		72.0	5.1	61.5	7.3	53.2	9.6	50.7	5.1	46.6	8.9	39.9	0.6	54.0	6.1
BOAT-MI-BN (Ours)		86.5	7.5	82.2	9.3	74.7	1.2	72.6	2.6	65.5	8.7	62.3	1.7	**74.0**	**5.2**

Table 2: Results with BatchNorm (average of 3 runs with different class order) for DFCIL over six task in EgoGesture3D.

Method	Task 0	Task 1		Task 2		Task 3		Task 4		Task 5		Task 6		Mean (Task 1 → 6)	
Oracle (DGSTA-BN)	75.0														
		G↑	IFM ↓	G↑	IFM↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM ↓	G↑	IFM↓	G↑	IFM↓
DeepInversion-BN [6]	77.4	66.4	16.1	52.3	24.6	35.0	46.5	24.5	59.7	20.5	65.0	15.1	72.8	35.6	47.5
ABD-BN [4]		65.7	17.4	58.1	19.4	49.5	31.1	45.1	35.2	42.7	37.2	39.5	40.4	50.1	30.1
R-DFCIL-BN [3]		68.3	5.9	56.0	11.9	49.0	17.9	44.4	20.1	39.9	29.4	36.8	30.6	49.1	19.3
BOAT-MI-BN (Ours)		73.0	4.6	66.9	7.1	60.8	15.9	57.9	16.3	52.1	22.2	49.0	27.6	**60.0**	**15.6**

1. Experimental Details

We follow the original implementations of all DFCIL methods that we used for benchmarking in this work. However, these methods were originally designed for the image classification problem, as we discussed in Section 4.4 of the main paper. Therefore, we had to adjust them to our keypoint-based gesture recognition domain and tune their hyperparameters individually for our setup. All hyperparameter values and configuration files will be made available along with the codebase.

Base model training: We use the original DG-STA [1] architecture as our backbone to process the 3D keypoint sequences. We train the Oracle (upper bound, all classes in one task) and Task 0 models (base classes for each setup) with Adam optimizer (learning rate of 1e-3, $\beta_1 = 0.9$ and $\beta_2 = 0.999$) for 150 epochs.

Learning rate for incremental tasks: All the methods start from the same Task 0 weights for Task 1. We modify the initial learning rate for the incremental tasks depending on the method since a stronger regularization requires a higher learning rate to learn the new classes. In this regard, we select the optimal learning rate for each method with a grid search in the range $\{$1e-5 : 1e-3$\}$.

Synthetic data generation: Following ABD [4], for a fair comparison among the model inversion-based approaches (DeepInversion [6], ABD [4] and R-DFCIL [3]), we use the same model inversion strategy for the synthetic data generation. However, we optimize the randomly initialized inputs directly instead of training a model to generate the samples. After a grid search we find $\{\alpha_{lr}, \alpha_{con}, \alpha_{stat}, \alpha_{temp}\}$ as $\{$1e-1, 1, 2e2, 1e1$\}$.

R-DFCIL hyperparameters: R-DFCIL [3] includes three hyperparameters to weight each of the loss terms, the **local classification loss** (λ_{lce}), the **hard knowledge distillation loss** (λ_{hkd}) and the **relational knowledge distillation loss** (λ_{rkd}). For our experiments, we found $\{\lambda_{lce}, \lambda_{hkd}, \lambda_{rkd}\}$ as $\{$1, 1.5e-1, 7.5e-1$\}$ as the optimal values for SHREC-2017 and $\{$1, 1e-1, 1e-1$\}$ for EgoGesture3D.

BOAT-MI implementation: The complete `Python`-style pseudocode is provided in Algorithm 1 and 2. In Algorithm 1 Line 1, `Function Main` is the driver for our proposed model inversion mechanism for a single class c. Similar to the other baselines, we employ DG-STA [1] backbone as our feature extractor $\mathcal{F}$ here. We use the radial basis function as the kernel for our SVM classifier ($\mathcal{H}$) here. To generate proto-SVs for model inversion, it uses the principal directions of the prototypical variance $\mathbf{\Sigma}$ (Line 29, Algorithm 1). Ray casting from the class-prototypical mean $\boldsymbol{\mu}$ is done iteratively with a learning rate α_f (Line 29-36, Algorithm 2) until the ray hits the boundary. Also, a margin δ, normalized with respect to the distance between the mean $\boldsymbol{\mu}$ and the boundary vector, is used to avoid noise inherent in the boundary estimation process (Figure 2 main paper, Line 33, Algorithm 2). In other words, the proto-SV feature is taken to be the on the margin, which is δ inside the boundary.

Note that the number of principal directions (PDs) is pretty low compared to the dimensionality of the feature vector. Along with these raw PDs, we consider upto third-order interactions, i.e. simple unweighted linear combination of PDs ($p[i]+p[j]$ and $p[i]+p[j]+p[k]$, Line 37-40, Algorithm 1, $p[\cdot]$ is the list of PDs) ignoring duplicates, which we found to be empirically sufficient for all our studies. Thus, our augmentations are deterministic (not random), conditioned on the set of PDs.

The feature inversion procedure (feature → input, Line 1-22, Algorithm 2) deviates from the vanilla implementation [2] in three aspects. First, we initialize the input tensor with the inverted class-prototypical mean (Line 9, Algorithm 2). Second, we employ the normalized $\mathcal{L}_2$ function as the distance metric (Line 14, Algorithm 2). Third, we replace the vanilla gradient descent with its momentum-based counterpart [5] (Line 18-19, Algorithm 2). All these modifications are empirically found to expedite the convergence.

The forward learning rate (α_f), momentum (m), and reverse learning rate (α) are set to 0.05, 0.9, and 1.0 respectively. We use the value of 0.2 as our normalized margin (δ) based upon the ablation study presented in Table 4 of the main paper. The rest of the parameters are set similarly to the baselines. All these configurations will be open-sourced with the codebase.

2. Additional Experiments

The effect of stat alignment loss during model inversion: Model inversion-based methods like DeepInversion, ABD and R-DFCIL use the BatchNorm (BN) statistics to compute a regularization loss that aligns the synthetic and real data distributions during model inversion. This loss is based on the KL divergence between the synthetic data distribution and the BatchNorm distribution of the previous task model, which reflects the real data statistics from the previous training. However, our backbone architecture, DG-STA [1], does not have BatchNorm layers and thus this loss term is not included. To investigate how the **stat alignment loss** affects model inversion for these baselines, we replace the LayerNorm layers with BatchNorm layers in DG-STA and run additional experiments.

Table 1 and 2 show the results for SHREC-2017 and EgoGesture3D setups, respectively. As can be seen, the incorporation of the BN layer does not change the overall ranking of the methods. With BN, our proposed approach achieves significantly higher global accuracy in each stage there with 12.1% and 9.9% improvements on average for SHREC-2017 (Table 1) and EgoGesture3D (Table 2), respectively, over the next best methods. Such improvements are also accompanied by the lower mean instantaneous forgetting in general – 0.9% and 3.7% lower IFM than the second best method for SHREC-2017 (Table 1) and EgoGesture3D (Table 2), respectively. Therefore, regardless of the choice of normalization layers, BOAT-MI excels the SOTA methods by a large margin.

Performance degradation/comparison to when all data is available, with large number of incremental stages: We first clarify that the upper bound of performance when all data is available – 75.8% – is reported on the "Oracle" row of Table 3 in our paper. To study this further with large number of incremental stages, we experimented with an extreme incremental learning setting on EgoGesture3D, where we added one class at a time to a model trained on the 59 base classes. Concretely, instead of adding 4 classes at a time over 6 incremental stages (as in paper), we added 1 new class at a time over 24 incremental stages. These results are shown in Figure 2 for both global accuracy and IFM. Our (BOAT-MI's) performance drops by about half after 24 incremental stages – much lower compared to SOTA methods.

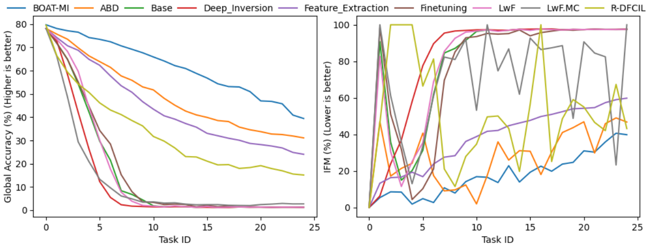

Figure 2: Global accuracy (%) and IFM (%) for $59 + 1 \times 24$ setup on EgoGesture3D dataset. Like other setups, our BOAT-MI excels the SOTA approaches here as well. Best viewed in digital format.

3. EgoGesture3D Processing

The new 3D skeleton dataset used in this paper is the derivative of the EgoGesture [8] comprising RGB-D images. Following the nomenclature of the parent dataset, we call it EgoGesture3D. Thanks to the MediaPipe [7] API that we use for 3D skeleton extraction from the images. The temporal association is turned on during the detection process with minimum detection and tracking confidences of 0.8 and 0.5, respectively. These numbers are chosen empirically for better extraction results.

However, merely using the extracted output from the temporally segmented gesture sequences is problematic for several reasons. First, the MediaPipe API fails to detect all the keypoints in all frames successfully. Next, it detects additional spurious hands (more than 2) in some of the frames. Also, in a few cases, for two hands detection, we find the handedness prediction to be the same (both left or both right hands). These issues are resolved with proper heuristics for each frame independently following the temporal detection with MediaPipe. Moreover, the baseline DG-STA model provides the optimal performance with 8 frames generally equally distant along the time dimension. Therefore, we save the samples with at least 8 frames of successful detections. In this regard, we experimented with interpolating with the less number of original frames. We find the oracle accuracy comparatively much lower that prevents us from saving samples with less than 8 frames. For the single-handed gestures, the values for the keypoints representing the absent hand are set to zeros. Very occasionally hand swapping occurs for just a few frames (i.e. 1-2 frames detect right hands with all others left). We keep these detections unchanged as some gestures may contain one hand mostly with the other in a cameo. Figure 3 and 4 show sample gestures in each column with the 2D projection of the skeleton on top of the corresponding RGB images.

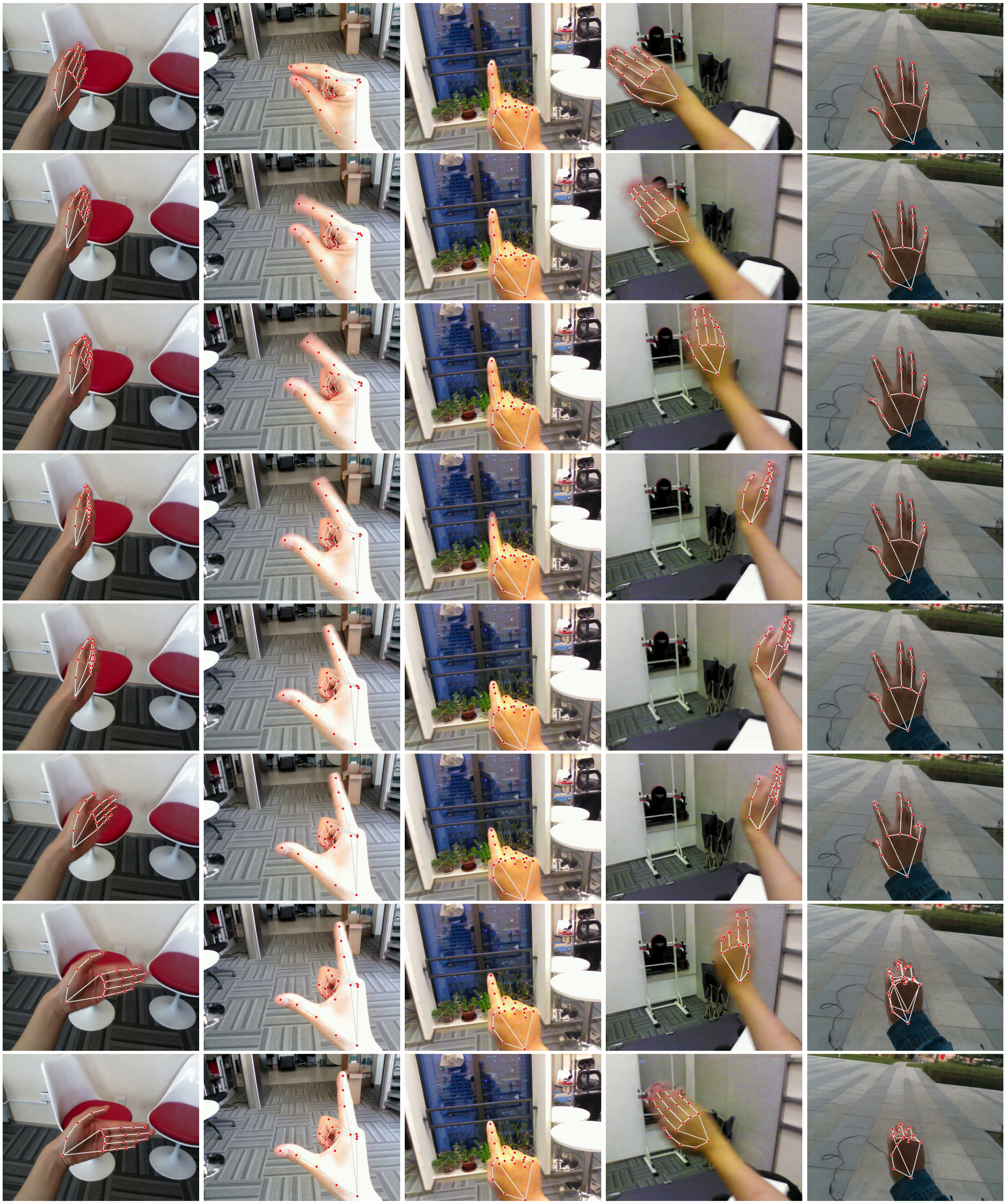

Figure 3: 2D projection of the EgoGesture3D skeletons on EgoGesture RGB images. (Top to bottom) Each column shows the temporal sequence for a single gesture (8 successfully detected frames selected in temporal order at random). (Left to right) [1] wave palm towards right; [12] zoom in with two fingers; [16] click with index finger; [39] wave hand; [48] grab.

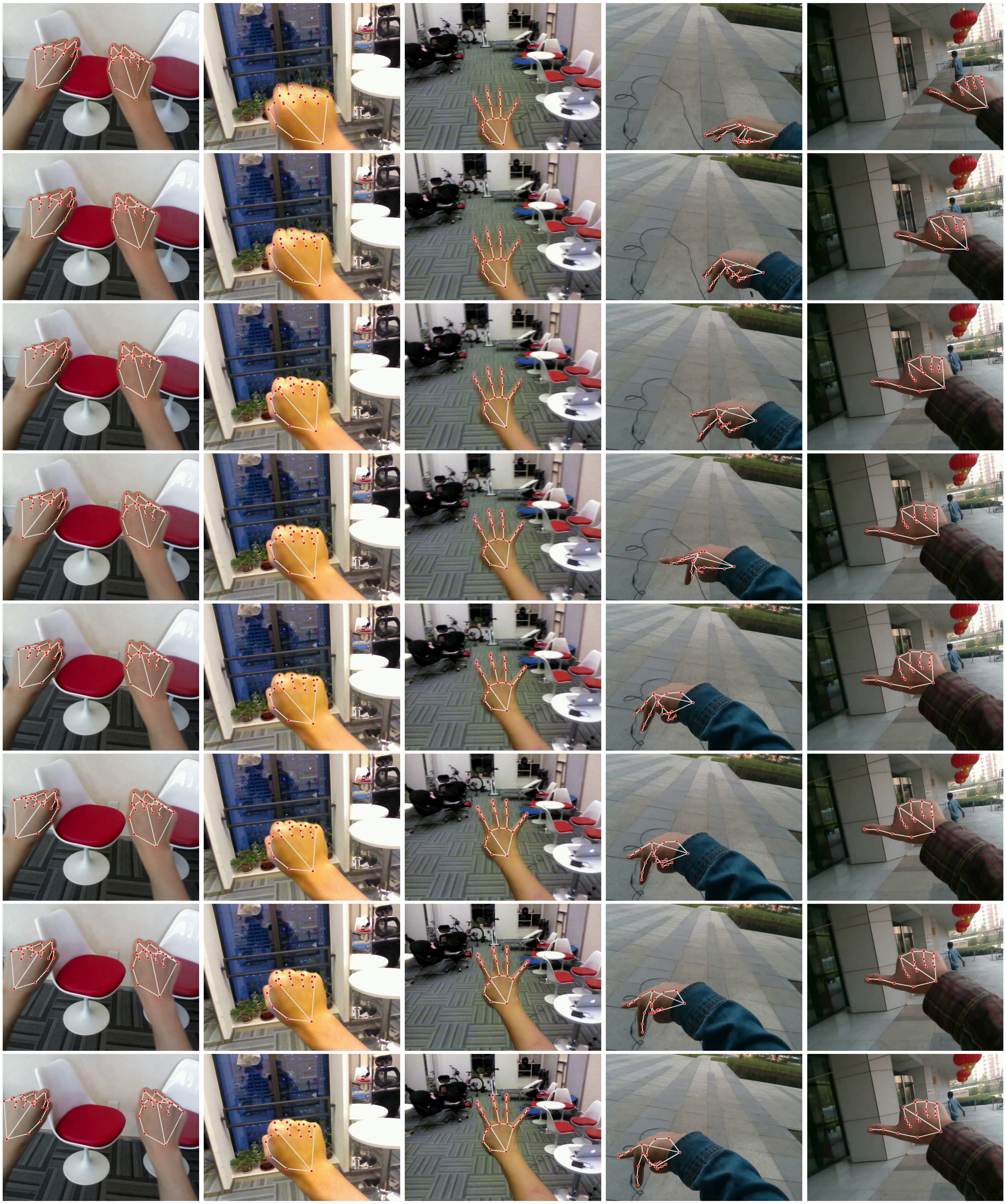

Figure 4: 2D projection of the EgoGesture3D skeletons on EgoGesture RGB images. (Top to bottom) Each column shows the temporal sequence for a single gesture (8 successfully detected frames selected in temporal order at random). (Left to right) [8] zoom in with two fingers; [21] static fist; [28] number 4; [49] walk; [66] thumb towards left.

References

[1] Yuxiao Chen, Long Zhao, Xi Peng, Jianbo Yuan, and Dimitris Metaxas. Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. In *BMVC*, 2019.

[2] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM Conference on Computer and Communications Security*, 2015.

[3] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *ECCV*. Springer, 2022.

[4] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *ICCV*, 2021.

[5] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[6] Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *CVPR*, 2020.

[7] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[8] Yifan Zhang, Congqi Cao, Jian Cheng, and Hanqing Lu. Egogesture: A new dataset and benchmark for egocentric hand gesture recognition. *IEEE TMM*, 2018.