Continual Personalization of Human Actions To New Categories Over Time

Prachi Garg

CMU-RI-TR-24-27

May 30, 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my Mom, Dad, and Naniji.

Abstract

As extended reality (XR) is redefining how users interact with computing devices, research in human action recognition is gaining prominence. Typically, models deployed on immersive computing devices are static and limited to their default set of classes. The goal of our research is to provide users and developers with the capability to personalize their experience by adding new action classes to their device models continually. Importantly, a user should be able to add new classes in a low-shot and efficient manner, while this process should not require storing or replaying any of user's sensitive training data. We formalize this problem as privacy aware few-shot continual action recognition. Towards this end, we propose *POET: Prompt-offset Tuning.* While existing prompt tuning approaches have shown great promise for continual learning of image, text, and video modalities; they demand access to extensively pretrained transformers. Breaking away from this assumption, POET demonstrates the efficacy of prompt tuning a significantly lightweight backbone, pretrained exclusively on the base class data. We propose a novel spatio-temporal learnable prompt offset tuning approach, and are the first to apply such prompt tuning to *Graph Neural Networks*. We contribute two new benchmarks for our new problem setting in human action recognition: (i) NTU RGB+D dataset for activity recognition, and (ii) SHREC-2017 dataset for hand gesture recognition. We find that POET consistently outperforms comprehensive benchmarks.

Acknowledgments

First, I want to express my sincere gratitude to Prof. Fernando De La Torre for being a great advisor and mentor to me in my two years at CMU. He gave me the opportunity to work on my dream research project, provided full freedom to define and lead my own research direction and entrusted me with the best research collaboration I could have asked for, guiding me every step of the way with kindness and transparency. For the first time, I learnt to stress less and have more fun in research. My biggest takeaways working with him have been effective communication, prioritization, and appreciating complementary skill sets in others. I shall continue to imbibe his emphasis on *simplicity* in idea, research, and writing going forward. I am also grateful to him for always encouraging open discussion and giving me a platform where I could confidently and uninhibitedly ask for and go after things I truly wanted.

My relatively exploratory thesis has been made possible solely by the hands-on guidance and enthusiasm of my collaborators (/advisory mentors). I have enjoyed defining, solving and writing this research so much that I decided to pursue a Ph.D. I want to thank Prof. Vineeth N Balasubramanian for joining us, for his creative research ideas, and the best architecture-level technical discussions. I have been collaborating with him for the past 4 years and I am most grateful to him for seeing the best in me, pushing me to be more ambitious, and teaching me to think for myself. I also got the opportunity to work with Joseph K J who is a visionary expert in continual learning. His relentlessly optimistic ways and research organization have significantly influenced my attitude towards research. I am very happy to have worked with Shugao Ma who was always excited about the research directions we were proposing, taught me how to design specific technical experiments to understand the model and push the project ahead. Necati Cihan Camgoz always had the most interesting questions about our inferences - I would come out of every Meta meeting wondering what it would take to think like him. I'm grateful to Kenrick Kin for his unique perspective and help with problem motivation and writing. Chengde Wan and Weiguang Si always shared their insights into the product side of things and helped me understand the actual use and application of our research.

I am truly grateful for the discussions, guidance, and feedback I received from my thesis committee, Prof. Deva Ramanan and Prof. Kris Kitani. I really look up to them and have enjoyed the discussions I had during their courses. I am inspired by Russell Mendonca's research and grateful for our discussions since first semester. I am also grateful to Prof. Srinivasa Narsimhan who taught me my favourite CMU course 'Physics Based Vision' and I will have the opportunity to work with during this summer. I'd like to thank Prof. C V Jawahar for giving me an opportunity to conduct research on open-ended problems that excited me during my Research Fellowship at IIIT-H before CMU, and his backing and advice thereon. A very special thanks to Prof. Frederic Jurie who at the end of a 3 month summer research internship (France, 2019) strongly encouraged me to pursue a Ph.D, told me “I am going to train much bigger models”, and has been excited about my journey till date.

I would like to thank my labmates from the Human Sensing Lab - specially Jianjin, Kevin, Jinqi, Chen, Cheng, Charles, Zoltan, Shubhra and Yehonathan. I have learnt a lot from my buddies Oliver and Younjoon and I'll particularly miss our Monday lunches and Mala hotpots. I am incredibly grateful to seniors Nikos, Anurag, Nupur, Zhiqiu, Mihir, Swami, Sriram, Tarasha, Rishi, Mononito, Nikhil, Sally, Shivam, Heng, Simin, Vidhi and Unnat for being great mentors and friends to me at CMU. They have really looked out for me and my research and provided very honest, blunt and valuable suggestions in the right moments when it mattered the most. Thanks for sharing your research outlook with me.

My favourite part about CMU is the peer group that I've had the privilege of knowing. I am incredibly fortunate to have friends who are not only constantly trying to improve themselves and grow everyday, but also want the absolute best for others. Here I witnessed the best competition-collaboration equilibirum I had always dreamt of. Bharath, Aman and Jay have really helped broaden my horizon and helped me see my own power. I am amazed by our conversations. Special thanks to Roshan and Yin for being there in very tough moments. I would also like to thank my roommate Aishwarya for giving me a much needed rejuvenated viewpoint on all aspects of life. I have cherished all conversations and study sessions I had with my fellow MSR friends from first semester until now. I'm sure I can tell you exactly what I have learnt from every single person.

I am lucky to have a rock solid support system from my 3 closest old friends: Sakshi who has for 13 years helped me see through the unnecessary, grounded me, constantly reminded me who I am and what I should be focusing on; Vaibhav who I call before every impossible deadline - the most notable being the ICLR one week submission run last year to hear, “I know you, you always make it in the end moment”; and Shyam for always reminding me how far I've come, and helping me see the brighter side of every situation. Finally, I have to thank my parents and brother for their lifelong unwavering support, encouragement and wisdom. My parents have turned into best friends and confidantes to me, brainstorming and making a plan with me to tackle every hurdle head-on. They have been encouraging me to take the road not (less) taken, take all the chances on myself and take as many risks as possible because, “Even if nothing works out, we will still have your back”. They take great joy in learning more about research and taking them along personally and professionally has been an amazingly fulfilling privilege. Thanks to my dad for being my champion.

Funding

This work was in part supported by the Meta-Reality Labs “Open-Set Gesture Recognition for Virtual Reality” Grant.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

A key input modality to virtual, augmented and mixed reality (often together termed as extended reality, XR) devices today is through recognizing human activity and hand gestures based on body and hand pose estimates. Recognizing human actions[1] facilitates seamless user interactions in head-mounted XR devices such as the Meta Quest 3 and Apple Vision Pro. If the provided action recognition models are static, then developers and users are limited to a predefined set of action categories. It is only expected that these devices are usually shipped with action recognition models that recognize a predefined set of action categories. With the growing use of such devices in new contexts and the increasing demand for personalized technology delivery, there is an impending need to enable the action recognition models in such systems to adapt and learn new user actions over time. Defining their own action categories allows users to customize their experience and expand the functionality of their XR devices. Addressing this need is the primary objective of this work.

Adapting human action models to new user categories over time faces a few challenges. Firstly, the model must be capable of learning new actions with minimal amount of training data so users can add new classes by providing just a few training examples per class. Secondly, due to the increasing use of XR devices for personal assistance, there is a need for privacy preservation in user action recognition-based pipelines [14]. Hence, the adaptation of such action recognition models to new user

[1]We use human action as an umbrella term for both hand gesture and body activity in this work for ease of presentation.

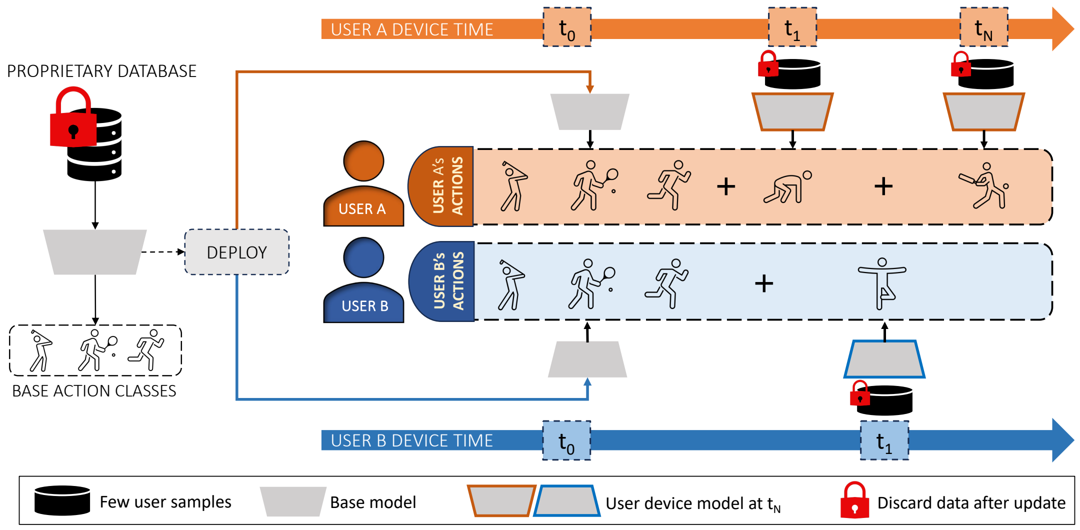


Figure 1.1: Proposed POET method **continually adapts** skeleton-based human action recognition models pretrained on a pre-defined set of categories **to new user categories with few training examples**. Users can thus expand the capabilities of XR systems with novel action classes by providing a few examples of each new class. We discard the user-sensitive data as soon as the model is updated on the new categories.

categories must also be 'data-free', i.e., it cannot store and replay previously seen user training data in subsequent continual sessions. Considering these requirements, we leverage the recent success of 'data-free' prompt-based learning and propose a new spatio-temporal prompt offset tuning approach to eficiently adapt the default model without finetuning.

Human action recognition systems are moving to skeleton-based approaches, especially in applications that require low-shot action recognition capabilities such as medical action recognition [27, 57]. Skeletons offer a robust and compact alternative to videos in such low-shot regimes, due to their relatively low dimensionality and lesser variance under background conditions. While there have been a wide variety of efforts in skeleton-based human action recognition over the years [36, 53, 54], there have been fewer efforts on adapting such models to newer user categories. Efforts like [1, 22] attempted to continually learn new user categories over time in skeleton-based human action recognition, but relied on fully-supervised data for the new classes. On the other hand, few-shot learning works [27, 47, 57] adapt a pre-trained skeleton-based action recognition model to new data, but without explicitly retaining past categories. In this work, we seek to learn new user categories in trained human action models with *very few labeled samples* for the new classes, while being *data-free* (not storing samples from previously trained categories). Fig 1.1 summarizes our overall objective. One could view our setting as privacy-aware few-shot continual learning for skeleton-based action recognition.

To this end, we propose a prompt offset tuning methodology that can be integrated with existing backbone architectures for skeleton-based human action recognition. Our learnable (soft) prompts are selected from a shared knowledge pool of prompts based on an input instance dependent attention mechanism. In particular, we propose prompt selection using an ordered query-key matching that enables a temporal *prompt* frame order selection consistent with the input instance. We show that such an approach allows us to learn new user categories without having to store data from past classes, without overwriting the pre-existing categories. To the best of our knowledge, this is the first effort on leveraging prompt tuning for skeleton-based models, as well as on spatio-temporal prompt selection and tuning.

1.0.1 Contributions

Our key contributions are summarized below:

- We formalize a novel problem setting which continually adapts human action models to new user categories over time, in a privacy aware manner.
- To address this problem, we propose a novel spatio-temporal Prompt Offset Tuning methodology (POET). In particular, it is designed to seamlessly plug-and-play with a pre-trained model's input embedding, without any significant architectural changes.
- Our comprehensive experimental evaluation on two benchmark datasets brings out the efficacy of our proposed approach.

Chapter 2

Related Work

2.0.1 Prompt Tuning

The idea of prompting, as it originated from Large Language Models (LLMs), is to include additional information, known as a text prompt, to condition the model's input for generating an output relevant to the prompt. Instead of applying a discrete, pre-defined 'hard' language prompt token, *prompt and prefix tuning* [20, 23] formalized the concept of applying 'soft prompts' to the input. A set of learnable parameters are prepended (concatenated) to the input text and trained along with the classifier while keeping the backbone parameters frozen. Similar to prompt tuning of LLMs, recent works have popularized prompt tuning of vision transformers [16] as an effective way of adapting large pretrained models to downstream tasks [50, 58]. However, it remains unexplored and undefined (to the best of our knowledge) for *non-transformer* architectures such as GNNs.

2.0.2 Prompt Tuning for Continual Learning

Prompt tuning provides a simple and cost-effective way of learning task-specific signal condensed into 'soft prompts'. For continual learning, training a set of prompts for each sequential task provides a natural alternative to storing privacy violating exemplars and replaying them. Training task-specific prompts for each sequential task is straightforward when authors assume access to task identity at both train and inference time, like in Progressive Prompts [34]. However, if task identity is unavailable at inference, the model will not know which task's prompts or classifier to use for evaluating a test sample. In this respect, S-prompts [48] and A-la-carte prompt tuning (APT) [3] learn an independent set of prompts for each domain/task and employ a KNN-based search for domain/task identity at test time. Since these methods learn stand-alone prompts for every task, the prompt feature space is task-specific, and there is no forgetting of old knowledge when learning new tasks (by design). At the same time however, these 'no forgetting' prompts *cannot share knowledge* across tasks.

This leads to another ideology for continual prompt tuning, i.e., treat each prompt unit as being a part of a larger **shared (knowledge) pool** of prompts. Then the desired number of prompt units can be selected from the pool, conditioned on the input instance itself [41, 49, 50]. Given the scarcity of new data in our setting, we hypothesize that sharing of knowledge will benefit new tasks and draw inspiration from this line of works. Most recently, Adaptive Prompt Generator (APG) [42] challenges the intensive ImageNet21K pre-training assumption as it prompts a ViT pretrained only on the continual benchmark's base class data (similar to us). However, they use replay and knowledge distillation-style 'anti-forgetting learning', in addition to using prompts. Even though our backbone is trained only on the base classes, we propose a **simple prompt tuning-only** strategy to counter forgetting. This implies that a prompt strategy is all we need to continually add new action semantics in a few-shot manner.

2.0.3 Few-Shot Class Incremental Learning

FSCIL is a challenging continual learning setting where a model overfits to new classes, with the simultaneous heightened (often complete) forgetting of old knowledge as soon as the base model is fine-tuned on few-shot data [9, 43]. Since the backbone feature extractor is the only source of previously seen knowledge, if it is updated, knowledge is lost forever. Typically, existing works decouple the learning of (backbone) feature representations from the classifier by *learning* the model *only on the base data* and relying on non-parametric class-mean classifiers for classification in subsequent steps [13, 31, 56]. This leads to a feature-classifier misalignment issue [32, 52] because new class prototypes are extracted from a backbone representation trained only on the base classes. We hypothesize that optimizing input prompt vectors along with a dynamically expanding parametric classifier on top of a frozen backbone can alleviate this misalignment issue. Our work not only provides a fresh perspective into FSCIL, but to our best knowledge is also the only work not designed for and evaluated on image benchmarks.

Chapter 3

Approach

3.1 Preliminaries

3.1.1 Skeleton Action Recognition Using Graph Representations

Our input $x \in \mathbb{R}^{T \times J \times 3}$ is a video sequence of T frames, each frame containing J joints of the human body (25 joints) or hand skeleton (22 joints), with joint coordinates given in 3D Cartesian coordinate system. Such a skeleton action sequence is naturally represented as a graph topology $G = \{\mathcal{V}, \mathcal{E}\}$ with $\mathcal{V}$ vertices and $\mathcal{E}$ edges. Graphs are modeled using Graph Neural Networks (GNNs) [12], which can either be sparse graph convolutional networks (GCN) or fully connected graph transformers (GT). In Fig. 3.1, we define a GNN architecture as $f(x) = f_c \circ f_g \circ f_e(x)$; input x is first passed through an input feature embedding layer f_e, followed by a graph feature extractor f_g composed of a stack of convolutional layers (in GCNs) or attention layers (in GTs), and finally a classifier f_c which predicts the action class label y. The input feature embedding is given by $x_e = f_e(x), x_e \in \mathbb{R}^{T \times J \times C_e}$.

3.1.2 Problem Definition.

Given a base trained model $\mathcal{M}$ deployed on a user's device, we would like to learn a new set of classes in T subsequent user sessions (also called tasks) $\{\mathcal{US}^{(1)}, ..., \mathcal{US}^{(T)}\}$

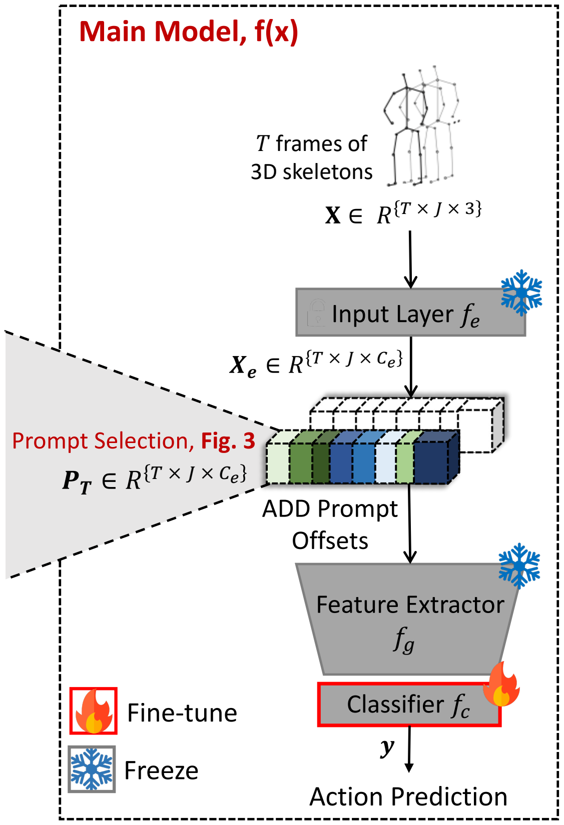

Main Model, f(x)
T frames of 3D skeletons
$\mathbf{X} \in R^{\{T \times J \times 3\}}$
Input Layer f_e
$\boldsymbol{X_e} \in R^{\{T \times J \times C_e\}}$
Prompt Selection, Fig. 3
$\boldsymbol{P_T} \in R^{\{T \times J \times C_e\}}$
ADD Prompt Offsets
Feature Extractor f_g
Classifier f_c
$\boldsymbol{y}$
Action Prediction
Fine-tune
Freeze

Figure 3.1: **POET: Prompt-offset Tuning** proposes to offset the input feature embedding x_e of the main model by learnable prompt parameters P_T for privacy-aware few-shot continual action recognition. We explain prompt selection mechanism in Fig. 3.2.

corresponding to training datasets $\{\mathcal{D}^{(1)}, ..., \mathcal{D}^{(T)}\}$; where $\mathcal{D}^{(t)} = (x_i^t, y_i^t)_{i=1}^{|\mathcal{D}^{(t)}|}$ are skeleton action sequence and label pairs provided by the user, $x_i^t \in \mathbb{R}^{T \times J \times 3}$, $y_i^t \in \mathbb{R}^{\mathcal{Y}^{(t)}}$. In each session, the user typically provides a few training instances F (e.g. $F \leq 5$) for each of the N new classes being added, such that $|\mathcal{D}^{(t)}| = NF$. The base model's session $\mathcal{UB}^{(0)}$ is assumed to have a large number of default action classes $\mathcal{Y}^{(0)}$ trained on sufficient data $\mathcal{D}^{(0)}$, which is most often proprietary and cannot be accessed in later user sessions. In each session, the user adds new action classes, i.e. $\mathcal{Y}^{(t)} \cap \mathcal{Y}^{(t')} = \emptyset, \forall t \neq t'$[1]. Due to the aforementioned privacy constraints, in any training session $\mathcal{US}^{(t)}$, the model has access to only $\mathcal{D}^{(t)}$; after training, this data is

[1]We make this assumption considering this is a first of such efforts; allowing for overlapping action classes and users to 'update' older classes would be interesting extensions of our proposed work.

made inaccessible for use in subsequent sessions (no exemplar or prototypes stored). After training on every new session $\mathcal{US}^{(t)}$, the model is evaluated on the test set of all classes seen so far $\cup_{i=0}^{t}\mathcal{Y}^{(i)}$. The challenge is to alleviate forgetting of old classes while not overfitting to the user-provided samples. One could view our setting as privacy-aware few-shot class-incremental learning for human actions, a problem of practical relevance – especially in human action recognition – which has not received adequate attention yet, to the best of our knowledge.

3.2 Methodology: POET

Overview: In Sec 3.2.1, we first define our spatio-temporal prompt offsets (called POET) to prompt tune a GNN (see Fig. 3.1), explaining why it is different from prompt tuning transformers for text, images and videos. Next, in Sec. 3.2.2 we explain our novel end-to-end optimization of the prompt codebook which enables prompt selection across novel continual tasks even though our query function is pretrained only on base task data (Fig. 3.2). Finally, our proposed prompt attachment *adds* the same number of prompts as the number of temporal frames in the input, such that the *selected prompts are temporally consistent with the input* (Sec 3.2.3).

3.2.1 Constructing Spatio-Temporal Prompt Offsets

Learnable (or soft [20]) prompts are parameter vectors in a continuous space which are optimized to adapt the pretrained frozen backbone model to each continual task. We define our spatio-temporal prompt offsets $P_{T'}$ as a set of T' prompts, each prompt P_i having length equal to the #joints in a frame J and feature dimension same as the input feature embedding x_e, i.e., $P_i \in \mathcal{R}^{J \times C_e}$.

Prompt tuning of GNNs? There exist prior works which generalize transformers to graphs [6, 11, 29]. *Then why is it non-trivial to attach prompts to a GNN?* NLP transformers [45] treat sentences as a fully connected graph with every word attending to every other word in the sentence as there are no explicit word interactions or edge information in sentences [12]. It is the same case with vision transformers [10] which compute attention among all image patches. This lack of pre-defined edge information makes it straightforward to prepend an arbitrary number of tokens for the

MHSA mechanism. However, our spatio-temporal input is a natural graph skeleton of the human joint-bone structure and *has a well defined edge connectivity*. This edge information is exploited as inductive bias in GNNs. If we treat joints as tokens and concatenate prompts along spatial or temporal dimensions, not only does the graph topology lose semantic meaning, but the forward pass operation becomes undefined (especially in GCNs). Hence, we denote our unknown prompt attachment function as f_p. The logit distribution can then be obtained as:

$$y = f(x, P_{T'}) = f_c \circ f_g \circ f_p(f_e(x), P_{T'}). \tag{3.1}$$

In every subsequent session $t > 0$, the classifier output dimension expands by N to accommodate for the new classes. Importantly, disparate from most existing continual prompt tuning works, our feature extractor backbone f_g is trained *only on the base class data* $\mathcal{D}^{(0)}$ and will never itself be fine-tuned on action semantics from any of the continual sessions $\mathcal{US}^{(t)}, t > 0$. After the base session training, parameters of f_g, f_e are frozen, and only the prompt-associated parameters and classifier f_c are updated, refer to Fig. 3.1.

3.2.2 Prompt Selection and Attachment Mechanism

Prompt Codebook Design.

As mentioned in Sec. 2.0.2, to encourage knowledge sharing in our few-shot continual setting, we choose to construct a single prompt pool $\boldsymbol{P}$, sharing encoded knowledge across all tasks:

$$\boldsymbol{P} = \{P_1, ..P_i, ..., P_M\}, \qquad P_i \in \mathcal{R}^{J \times C_e}; M = \#\text{prompts at time t}. \tag{3.2}$$

For the prompt selection process (Fig. 3.2), we construct a bijective key-value codebook, treating prompts in the pool $\boldsymbol{P}$ as values and defining learnable key vectors as $\boldsymbol{K} = \{K_1, .., K_i, .., K_M\}, K_i \in \mathcal{R}^{C_e}$. A query Q and keys $\boldsymbol{K}$ matching process is used to find and fetch the T' closest values (prompts in the pool). This vector quantization process is enabled by a query function $f_q(x)$. The query function is an encoder which maps an input instance x to a query Q as:

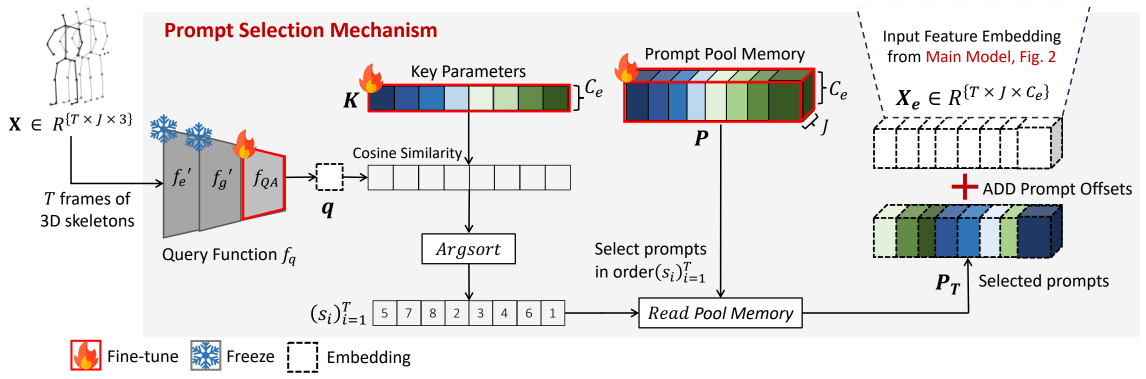


Figure 3.2: **Construction of our prompts** P_T: Input-dependent query Q is matched with keys $\boldsymbol{K}$ using *sorted* cosine similarity to get an ordered index sequence $(s_i)_{i=1}^T$ of the top T keys. This ordered index sequence is used to select the corresponding ordered prompt sequence P_T from prompt pool $\boldsymbol{P}$. We *add* P_T to x_e from main model (Fig. 3.1), thereby offsetting it. Our experimental evaluation confirms that such an additive spatio-temporal prompt offsetting can balance the plasticity to learn new classes from few-shot gestures, while maintaining the stability on the previously learned classes.

$$Q = f_q(x) = f_{QA} \circ f_g' \circ f_e'(x), \qquad x \in \mathbb{R}^{T\times J\times 3} \rightarrow Q \in \mathcal{R}^{C_e}. \tag{3.3}$$

where the *query adaptor* f_{QA} is a fully connected layer mapping the f_g' output dimension to the desired prompt embedding dimension C_e. The most relevant T' prompt indices are then selected using cosine similarity $\gamma(.)$ as:

$$\underset{T'}{\operatorname{argmax}}\, \gamma(f_q(x), \boldsymbol{K}). \tag{3.4}$$

Coupled Optimization in incremental steps, $t > 0$. To move queries closer to their aligned T' keys during training, we use a vector quantization clustering loss inspired from VQ-VAE [44] and similar to Learning to Prompt (L2P) [50]:

$$\max_{\theta_{QA}, \theta_{K_{T'}}} \lambda \Sigma_{i=1}^{T'} \gamma(f_q(x), K_i). \tag{3.5}$$

Even though L2P [50] and Dual-P [49] serve as strong starting points for our continual prompt selection mechanism, they decouple the optimization of prompt pool from the keys. This implies that the cross entropy loss for every new task ($t > 0$) only updates the classifier and prompts, but not the keys and query function, as the argmax operator in Eq. 3.5 prevents backpropagation of gradients to the keys. *However, this would be a fatal flaw in our setting:* We assume no large-scale pretraining on a superset of semantic knowledge the model could potentially see during its lifetime. Thus, for all practical purposes, our query function f_q is pretrained *only on the base class data* $\mathcal{D}^{(0)}$. For it to discern all old and new class samples, it must be updated as the model learns new classes. As shown in Fig 3.2, we propose to couple this optimization process such that the cross entropy loss for new tasks updates (i) the classifier f_c, (ii) selected prompts $P_{T'}$, (iii) keys $K_{T'}$, as well as (iv) query adaptor f_{QA}. We freeze the query feature extractor layers f'_g, f'_e in $t > 0$ to prevent catastrophic forgetting of base knowledge. We approximate the gradient by the straight-through estimator reparameterization trick as in [2, 44]. Our proposed coupled cross entropy loss:

$$\min_{\theta_{QA}, \theta_{K_{T'}}, \theta_{P_{T'}}, \theta_{f_c}} \mathcal{L}(f(x, P_{T'}), y). \tag{3.6}$$

This coupled mechanism establishes a *prompt optimization framework which is imperative to prompt tuning without pretraining.* It enabled us to design our spatio-temporal prompt offsets described next.

Additive Prompt Attachment. We achieve best results when the number of selected prompts T' in $P_{T'}$ is same as the number of temporal frames in the input skeleton, T. We find a naive addition operation yields best results empirically:

$$f_p(x_e, P_{T'}) = x_e + P_{T'}, T' = T. \tag{3.7}$$

Note, we use T instead of T' in the following sections, and in Figures 3.1 and 3.2.

3.2.3 Ordered Prompt Sequence Selection

Since the number of selected prompts corresponds with the number of temporal frames in the input skeleton sequence, we make the prompt selection mechanism temporally consistent with respect to the input. We achieve this by simply sorting the top T keys based on their the query-key cosine similarity to get an *ordered key index sequence* $(s_i)_{i=1}^{T}$, replacing a naive top T' index selection in Eq. 3.5 as:

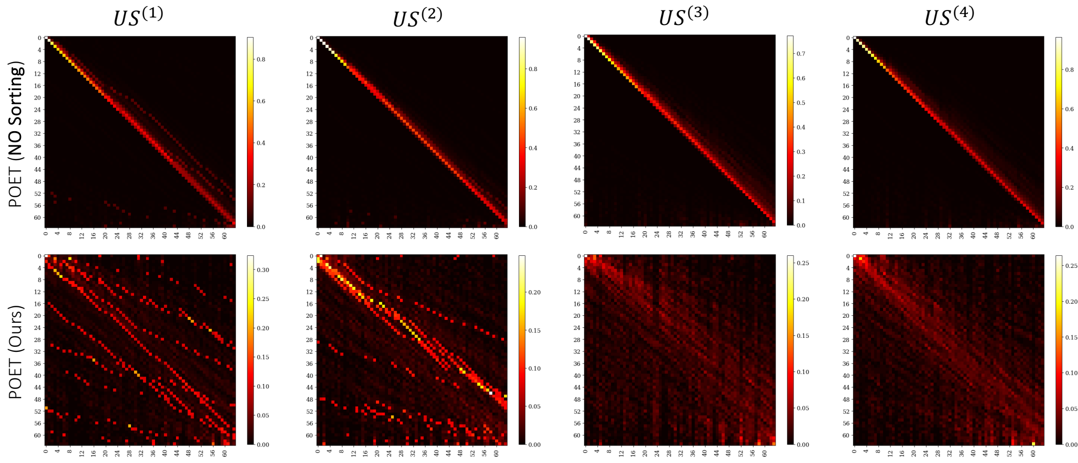


Figure 3.3: **Task-level Prompt Sequence Analysis.** Here we visualize the order $(s_i)_{i=1}^{T}$ in which the $M = 64$ prompts in the prompt pool are selected at train time, with the addition of 5 new classes in each user session $\mathcal{US}^{(t)}$. X-axis: prompt index number, Y-axis: index position in selected sequence. *Bottom:* Even though the same 64 prompts are selected and updated, the ordered prompt selection helps POET learn new knowledge better and renders temporal consistency. *Top:* The no sorting case uses the default sequence, giving equal importance to all prompts (diagonal matrices).

$$_{(s_i)_{i=1}^{T} \subseteq [1,M]} \gamma(f_q(x), \boldsymbol{K}). \tag{3.8}$$

We use this order $(s_i)_{i=1}^{T}$ to read the prompt pool memory and select final prompts P_T.

3.2.4 Mitigating Prompt Pool Collapse

After coupling the prompt pool and keys, we observed in our initial experiments with pool size $M > T$ that the same set of prompts are getting selected across training iterations and incremental steps (Figure 3.4, A). More concretely, as the vector quantization loss (Equation 3.6) brings the query close to the selected keys, the same set of active prompts get selected and optimized in each iteration, not using the other prompts at all. This is similar to the well known issue of "codebook collapse" in VQ-VAE [8, 51, 55]. Based on this observation, we design two prompt pool update mechanisms for the 2 cases:

1. $M = T, \forall t$. **No pool expansion, Algorithm 1.** All the prompts are selected in all tasks. But, the *order of their selection* $(s_i)_{i=1}^{T}$ varies with each input

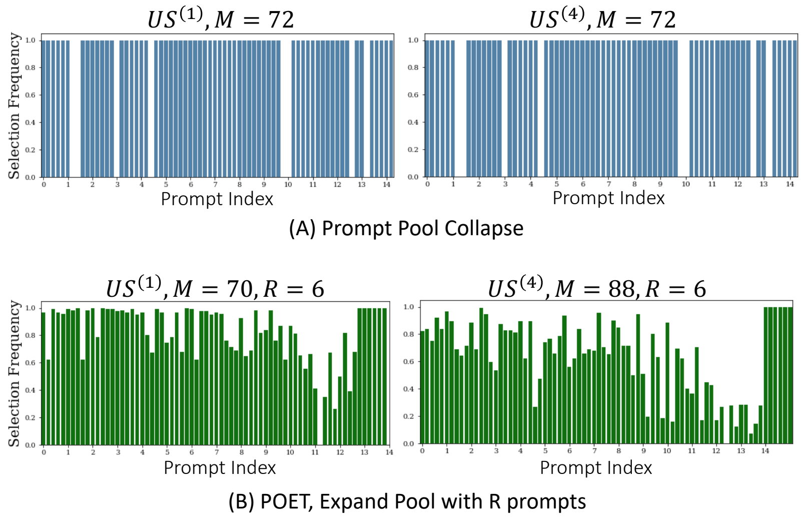


Figure 3.4: $M > T$ **Case:** *Prompt Pool Collapse.* When the pool size (M) is greater than the required number of prompts (T), certain prompt indices remain completely unused across all tasks. *Our POET pool expansion* algorithm alleviates pool collapse.

instance and is discriminative such that certain prompts tend to appear before others in $(s_i)_{i=1}^{T}$ for different input semantics. In Figure 3.3, we visualize the position occupied by prompt indices in the ordered sequence $(s_i)_{i=1}^{T}$. Entropy increase across tasks $t = 1$ to $t = 4$ shows that our selection mechanism learns to select a unique temporal code for all inputs.

2. $M = T + (R * t), t > 0$. **Expand pool with R prompts**. We also propose an order-aware prompt pool expansion algorithm (supplementary) that selects prompts from an expanded pool in a temporally coherent manner, for $t > 0$. This alleviates prompt pool collapse as shown in Figure 3.4, B.

3.2.5 Prompt Pool Expansion

We further present the order-preserving prompt pool expansion Algorithm 2 that (A) expands pool to learn new knowledge, (B) freezes previous prompts to prevent forgetting, and (C) forces usage of new prompts at the end of the sequence, hence alleviating the prompt pool collapse issue while preserving already existing temporal order statistics Figure 3.4 of main paper. We find $R = 6$ new prompts to be the best empirically for NTU RGB+D and $R = 2$ for SHREC 2017 using our 30% validation

Algorithm 1 POET (No pool expansion case) at Train Time, $t \geq 1$

```
Input: Query function $f_q$, prompt keys $\boldsymbol{K} = \{K_j\}_{j=1}^{M}$, prompt pool $\boldsymbol{P} = \{P_j\}_{j=1}^{M}$; main model input embedding layer $f_e$, graph feature extractor $f_g$, classifier $f_c$ pretrained from time $t-1$.
Note, M=T
Initialize: $\boldsymbol{P}, \boldsymbol{K}$ from $t-1$; Expand $f_c$ by $N$ new classes. Initialize $f_c$ as: (i) copy $f_c^{old}$ weights, (ii) $f_c^{new} \leftarrow Mean(f_c^{old})$
Freeze: query layers $f'_g, f'_e$; main model layers $f_e, f_g$
for epochs and batch $(x_i^t, y_i^t)_{i=1}^{NK}$ do
  1. Compute query feature $Q = f_q(x)$
  2. Compute cosine similarity $\gamma(.)$ b/w query $Q$ and keys $\boldsymbol{K}$
  3. Sort $\gamma(.)$; Get ordered key index sequence $K_T = (s_i)_{i=1}^{T}$ (Equation 3.8)
  4. Index prompt pool $\boldsymbol{P}$ in order $(s_i)_{i=1}^{T} \rightarrow$ Get final prompt $P_T$
  5. Compute input embedding $x_e = f_e(x)$
  6. Attach/Add prompt as per Equation 3.7, use prompted input to get prediction.
  7. Use clustering loss (Equation 3.5) to update query adaptor $QA$, keys $K_T$
  8. Use cross entropy loss (Equation 3.6) to update $QA, K_T, P_T, f_c$
  9. Classifier $f_c$ is (i) cosine classifier for NTU RGB+D;
     (ii) regular FC layer, but freeze $f_c^{old}$ by zeroed gradients for SHREC 2017,
end // See $t = 0$ training protocol in Supplementary Algorithm 2.
```

set of incremental sessions. Algorithm 2 presents our algorithm for prompt pool expansion.

Algorithm 2 Prompt Pool Expansion at Train Time, $t \geq 1$

```
**Input:** Query function $f_q$, keys $\boldsymbol{K} = \{\boldsymbol{k_j}\}_{j=1}^{T}$, prompt pool $\mathbf{P} = \{\boldsymbol{P_j}\}_{j=1}^{T}$; main model $f_e$, $f_g$, $f_c$
**Expand:**
Pool and keys by $R$ new prompts as: $\mathbf{P}_M \to \mathbf{P}_{M+R}; \boldsymbol{K}_M \to \boldsymbol{K}_{M+R}$
Where $\mathbf{P}_{M+R} = \{\mathbf{P}_M; \mathbf{P}_R\}$ (attach new prompts at the end of existing tensor)
**Initialize:** New prompts $\mathbf{P}_i \leftarrow \mathcal{U}(0,1)$; new keys $\boldsymbol{K}_i \leftarrow Mean(\boldsymbol{K}_M)$
**Construct** $\mathbf{P}_T$ as:
  1. Find $T - R$ key indices $\boldsymbol{K}_{T-R}$ using Eq. 3.7. Use this sequence to read previous prompts in the pool $\mathbf{P}_M$ and form $\mathbf{P}_{T-R}$.
  2. Concatenate $\mathbf{P}_R$ new prompts at the end of the sequence: $\mathbf{P}_T = \{\mathbf{P}_{T-R}; \mathbf{P}_R\}$ (i.e. explicitly use $R$ new prompts).
**Freeze:** Previous task prompts in the pool $\mathbf{P}_M$.
**Train:** New prompts $\mathbf{P}_R$, all keys $\boldsymbol{K}_{M+R}$ (to learn global inter-task selection), query adaptor $f_{QA}$, and classifier $f_c$.
```

Chapter 4

Experiments

4.0.1 Benchmark Details

Datasets. We evaluated our method on well-known action recognition datasets: (i) activity recognition on the NTU RGB+D dataset [39]; and (ii) hand gesture recognition on the SHREC-2017 dataset [40]. As we introduce a new problem setting in human action recognition, we contribute two new benchmarks to the community for this setting, on the NTU RGB+D and SHREC-2017 datasets.

For the NTU RGB+D dataset, we divide the 60 daily action categories into 40 base classes, learning the remaining 20 classes in subsequent user sessions. In few-shot learning parlance, our protocol is 4-task 5-way 5-shot, i.e. 5 novel classes using 5 user training instances in 4 user sessions. Each input 3D skeleton sequence has 64 temporal frames, each consisting of 25 body keypoints, such that $x \in \mathcal{R}^{64\times25\times3}$. We use the spatio-temporal GCN, CTR-GCN [7], as the architecture for NTU RGB+D, where we choose the joint input modality for better interpretability of prompt tuning.

For SHREC-2017, we divide the 14 fine-grained hand gesture classes into 8 base classes and 6 classes learned in subsequent user sessions. This is done in a 3-task 2-way 5-shot protocol, i.e. 2 novel classes using 5 user training instances in 3 user sessions. For each input instance of SHREC-2017, we use 8 temporal frames each having 22 hand keypoints, such that input $x \in \mathcal{R}^{8\times22\times3}$. We use a fully-connected graph transformer backbone, DG-STA [6] for SHREC-2017. We select DG-STA due to easily reproducible code and to check if our method POET works equally well across graph convolutional networks and graph transformers. The base model in NTU RGB+D is trained on 26,731 samples. However, the base model for SHREC 2017 is a relatively lightweight backbone trained on only 1146 samples (which could be sensitive to the training samples provided in user sessions). Hence, we conduct experiments on 5 sets of the 5 user samples, and report mean and standard deviation across all 5-shot runs for this benchmark.

Evaluation Metrics. Following earlier work in similar settings [31], we report: (i) Average accuracy '*Avg*' of all classes seen so far, and (ii) Harmonic Mean A_{HM} between '*accuracy only on old classes*' and '*accuracy only on new classes*' after learning each new user session. Note that the average accuracy tends to be biased towards the base session $\mathcal{T}^{(0)}$ performance due to more number of base classes. A higher A_{HM} implies better stability-plasticity trade-off between new task performance and old tasks' retention. A lower 'Avg' and higher A_{HM} may indicate better plasticity to a new task. Unlike many earlier CIL efforts, we report accuracies for both old and new classes in each user session for transparency.

Implementation Details. We observe that a key source of forgetting in our setting is from the classifier as the logits tend to become heavily biased towards the few-shot samples of new classes. We use a cosine classifier for activity recognition experiments on CTR-GCN. For gesture recognition on the lightweight DG-STA, we use a standard fully-connected layer as classifier, but freeze old class parameters in the classifier by zeroing their gradients. We attach prompts after the 1st layer of DG-STA and 1st CTR-GC block of CTR-GCN. For both datasets, we have equal or higher learning rates in user sessions when compared to the base model's training in order to accommodate new knowledge in the model (for better plasticity). For exact implementation details (including learning rates, epochs, hyperparameter analysis), see Appendix A. In earlier efforts that more generally tune prompts for class-incremental learning [41, 46, 48, 49, 50], it is common to rely on an ImageNet21K pretrained ViT [37] or CLIP [33] as the backbone. However, such backbones do not exist for skeleton-based human action recognition. Our base feature extractor is hence trained on the base session dataset itself without any pretraining, making this one of the first efforts of prompt tuning without extensive pretraining (scale difference of 3-5 times lower order of magnitude).

Results. Since there are no existing baselines for our proposed setting in skeletal action recognition, we compare our method by adapting continual learning (CL) baselines to skeletal data in Sec 4.0.2, Tables **??**, 4.2. We first compare POET with prompt tuning based class-incremental learning (CIL) approaches originally designed for images (L2P [50], CODA-P [41], APT [3]) and find that it has very low performance on new classes as they do not update their query function. We find any fine-tuning or knowledge distillation based approaches (LWF [25], EWC[17], LUCIR [15]) lead to rapid forgetting of base knowledge as the model overfits to user's few-shots. We also compare with multiple variants of Feature Extraction (FE) to check if prompts truly have merit (POET=FE+Prompts) and provide upper bound baselines. In Sec 4.0.3, we first show the importance of prompts in POET by removing the prompts. We discuss the value of our coupled optimization, query function update and *ordered key index selection* in our prompt selection ablation Tab 4.3. We also study the impact of proposed additive prompt tuning as compared to other possible prompt attachments f_p in Tab 4.4.

4.0.2 Comparison with State-of-the-Art

POET sets the SOTA on existing prompt tuning works (Tab ??,4.2). We adapt three standard CIL works that prompt tune ViTs for images - L2P [50], CODA-P [41] and APT [3] to our setting. L2P and CODA-P share prompt pool across tasks (similar to us), whereas APT learns task-specific prompts. L2P decouples the optimization of keys from the prompt pool and concatenates the selected prompts. Since concatenation is not defined for our modality and GNN backbone, we show results with two adaptations (i) concatenation along temporal dimension (**L2P*, CODA-P***), and (ii) concatenation along a rolled out spatio-temporal dimension in **L2P**, CODA-P**** followed by a linear layer remapping to the required original dimensionality. CODA-P [41] couples keys with the prompt pool by using a cosine similarity weighing over all prompts in the pool, forming a 'soft prompt selection', different from our 'ordered hard prompt selection'. In **APT**, we train prompt-classifier pairs for each continual task separately (ˆ denotes task-specific), and use task identity at test time. See adaptation details in Appendix A. These methods by design rely on a strongly pretrained query function (ImageNet 1k or 21k) which does not require updates, perhaps explaining the poor 'New' class performance in our setting.

Standard Continual learning Baselines. We compared with two well established knowledge-distillation approaches, learning without forgetting (**LWF**) and **LUCIR**. Both of them perform poorly on both old and new classes. **EWC** [17] learns better on new but does not retain old knowledge. We conclude that any CL method that fine-tunes the backbone feature representation in subsequent sessions $t > 0$ will not be able to retain base/old class knowledge (a finding consistent with existing FSCIL literature for images [9, 43]). We also adapt and compare with one of the latest FSCIL baselines **ALICE** [31], originally developed for image classification benchmarks on our gesture recognition benchmark in Table 4.2. Note the high retention of base task performance (due to non-parametric classifier on top of frozen base model). However, it suffers from poor plasticity and adaptation to new classes. This is the issue of feature-classifier misalignment that we hoped to alleviate through prompt tuning.

Fine-tuning (FE) and Feature Extraction (FE) Baselines. We implement standard continual learning baselines to understand stability-plasticity trade-offs in our new benchmarks. In all these baselines, we expand the classifier output dimension by N new classes. In **'FT (Fine-Tuning)'**, we tune all model parameters on cross entropy loss of new task. FSCIL is challenging for this modality as old task performance sharply reduces to zero starting from $\mathcal{US}^{(1)}$ as model overfits to user's few-shots. **'FE (Feature Extraction)'**[1] differs from FT as we freeze the feature extractor to preserve base knowledge. This serves as a competitive baseline in our findings. In **'FE, frozen'**, we zero out the gradients of previous class weights in classifier f_c to prevent forgetting from the classifier. 'FE' and 'FE, Frozen' exhibit different New-Old trade-offs in Tables **??**, 4.2 because the scale of pretraining is different (gesture more lightweight than activity).

Upper-bound baselines, top section Tables ??, 4.2. In **'Joint (oracle)'** experiment, we train on all task data at the same time in a multi-task (non-sequential) manner. Training POET in a multi-task manner (Joint POET) outperforms Joint oracle by 3.7% demonstrating the strength of our approach. In addition to these *generalist* upper bounds, we point out that '**FE, Task-specific^**' is a competitive *specialist* upper bound. In this, we perform feature extraction from base model to each task individually, storing separate task-specific models ($\mathcal{US}^{(0)} \rightarrow \mathcal{US}^{(i)}$, $i > 0$). POET outperforms 'New' accuracy compared with this baseline, achieving a forward

[1]'FE' is the same as 'w/o prompts' in Table 4.3. We highlight key baselines in gray color.

Table 4.1: **Activity Recognition Results (%, ↑), Comparison with SOTA:** NTU RGB+D [39] dataset on CTR-GCN [7] backbone. After training on each incremental task, we report Average of all classes seen so far ('Avg'). We also report (i) A_{HM}, (ii) only old classes accuracy ('Old'), (iii) only new classes accuracy ('New') in the last session. Numbers in brackets are wrt POET. *POET achieves the best stability-plasticity trade-off across all baselines indicated by the $A_{HM} = 58.3\%$. POET also has the highest Avg across all user sessions outside of upper bound baselines in orange*.

	$\mathcal{UB}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(2)}$	$\mathcal{US}^{(3)}$	$\mathcal{US}^{(4)}$			
Method	Base	Avg	Avg	Avg	Old	New	Avg	A_{HM}
Upper Bound Baselines								
Joint (Oracle)	88.4	79.0	71.0	66.8			**63.5** (+4.3)	
Joint POET (Oracle)							**67.2** (+8.0)	
FE, Task-Specificˆ	88.4	59.1ˆ	39.4ˆ	45.5ˆ	**72.1** (+12.8)	38.8 (-18.6)	NA	NA
FE+Replay	88.4	81.0	74.9	71.9	29.7 (-29.6)	72.8 (+15.4)	69.2 (+10.0)	42.2 (-16.1)
Linear Probing Baselines (Stability-plasticity trade-offs of backbone)								
FE	88.4	74.5	66.3	49.5	39.2 (-20.1)	46.8 (-10.6)	39.9	42.7 (-15.6)
FE, Frozen	88.4	78.1	62.3	41.7	29.5	25.5	29.2	27.4
FE+Replay†	88.4	73.1	60.4	61.6	60.3	28.8	57.7	39.0
FT	88.4	6.8	6.0	3.8	0.0	25.1	2.1	0.0
Standard Continual Learning Baselines								
LWF [25]	88.4	8.7	3.2	2.9	0.0 (-59.3)	23.1	1.9	0.0
EWC [17]	88.4	6.4	6.2	3.1	0.0 (-59.3)	28.9	2.4	0.0
Experience Replay	88.4	34.1	46.4	46.3	50.3 (-9.0)	25.4	48.3	33.8
Experience Replay†	88.4	7.0	6.0	10.0	10.1	32.2	12.0	15.4
LUCIR [15]	87.9	2.2	2.0	2.7	1.3 (-58.0)	2.5	1.4	0.0
Continual Prompt Tuning Baselines (Adapted from Image Classification)								
CODA-P [41]*	87.4	76.3	68.4	59.7	60.9 (+1.6)	0.7 (-56.7)	55.7 (-3.5)	1.3
CODA-P [41]**	87.6	4.5	28.8	0.6	35.4	0.7	31.1	1.3
L2P [50]*	**88.6**	78.5	70.7	63.7	61.2 (+1.9)	0.0 (-57.4)	56.2 (-3.0)	0.0
L2P [50]**	84.7	2.7	2.8	2.6	2.4	0.4	2.2	0.6
APT [3]ˆ	86.6	29.2ˆ	30.6ˆ	35.9ˆ	NA	35.0 (-22.4)	NA	NA
POET (Ours)	87.9	**82.8**	**76.8**	**68.6**	59.3	**57.4**	**59.2**	**58.3**

transfer on each $t > 0$. This indicates that prompt tuning benefits New performance due to the pre-existing knowledge in the shared knowledge pool. Avg in sessions $0 < t < 4$ indicates New for task-specificˆ .

Experience Replay Baselines, Tab ??. Even though our privacy-aware setting prohibits previous data replay, we compare with **'Experience Replay'** (store and replay 5-samples of base and incremental sessions) and **'Experience Replay†'** (store only 5-shots of incremental sessions) for completeness. **'FE+Replay'** serves as the best upper bound (even better than Experience Replay as we are freezing backbone in addition to replay). It is noteworthy that POET (which is FE+prompts) learns an implicit 'data-free' form of prompt pool memory, and yet has a better A_{HM}

Table 4.2: **Gesture Recognition Results (%, ↑), Comparison with SOTA:** SHREC 2017 [40] dataset on DG-STA [6] graph transformer backbone. Reporting mean and standard deviation across 5 runs. Bracket accuracy difference is wrt POET. **POET achieves best** $A_{HM} = 56.2\%$.

	$\mathcal{UB}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(2)}$	$\mathcal{US}^{(3)}$			
Method	Base	Avg	Avg	Old	New	Avg	A_{HM}
Joint (Oracle)	88.8	79.4 ± 0.7	77.3 ± 2.1			**70.9 ± 1.2**	**62.4 ± 0.4**
FT	88.8	20.3 ± 0.8	12.4 ± 2.1	0.0	85.8 ± 9.4	13.4 ± 1.5	0.0
FE	88.8	62.7 ± 2.4	41.9 ± 6.9	17.5 ± 5.1	77.3 ± 8.8	26.8 ± 3.4	28.5 ± 6.4
FE, Frozen	88.8	71.3 ± 1.9	61.4 ± 2.7	44.7 ± 3.2 (-1.2)	54.5 ± 6.7 (-17.9)	46.2 ± 2.7	49.1 ± 4.3
LWF [25]	88.8	20.2 ± 1.4	12.5 ± 1.0	0.0	**88.4 ± 13.7**	13.8 ± 2.1	0.0
L2P [50]	88.8	20.3 ± 5.9	10.5 ± 4.8	8.2 ± 4.0	6.9 ± 8.5	7.9 ± 3.9	7.5 ± 5.5
CODA-P [41]**	87.7	15.6 ± 4.5	11.6 ± 1.9	7.9 ± 1.8	14.1 ± 21.4	8.8 ± 2.4	10.1 ± 3.2
ALICE [31]	92.1	72.4 ± 5.7	63.3 ± 7.6	**62.5 ± 6.8** (+16.6)	11.9 ± 9.9 (-60.5)	**54.6** ± 6.9	20.0 ± 8.1 (-36.2)
POET (Ours)	91.9	73.2 ± 3.7	61.9 ± 1.8	45.9 ± 2.6	72.4 ± 7.1	50.0 ± 1.6	**56.2 ± 1.6**

trade-off as compared to explicitly stored and replayed samples from previous classes in FE+replay.

4.0.3 Ablation Studies and Analysis

Importance of prompts in POET. First, we *consider the contribution of prompt offsets in POET*. Since we only attach prompts to address continual learning in POET, removing prompts gives the Feature Extraction (FE) baseline ('w/o prompts', Table 4.3) where the backbone is frozen after base training and only the classifier is expanded and updated on classification loss of new classes. POET improves both, 'Old' (↑ 20.1%) and 'New' (↑ 10.6%) marked in blue.

Prompt Selection Mechanism. In Table 4.3, we investigate our prompt selection mechanism and optimization choices. The **'w/o coupled optim.'** experiment is a direct comparison of our additive *prompt attachment* with the de-coupled optimization in L2P [50]. Updating key parameters but keeping only query adaptor QA frozen after base session training (**'w/o QA update'**) reduces 'New' only performance of Task 4 by 4.6% as the query function stays fixed at base session learning and is not discriminative towards new classes. **'W/o clustering loss'** from Eq. 3.6, performance drops starting from $\mathcal{UB}^{(0)}$ itself. The only difference between the experiment **'w/o sorting'** and 'POET (M=T)' is that we do not *sort* the cosine similarity before selecting top T indices (same as Fig 3.3). The 10.8% ↑ in 'New' performance validates that our prompt selection mechanism is learning to chose a distinct temporal ordering for prompt tuning of new input samples. With pool expansion (**'POET,** $M > T$**'**),

Table 4.3: **Prompt Selection Mechanism Analysis (%, $\uparrow$):** POET on NTU RGB+D dataset. 'w/o' denotes removing that component from our method POET, numbers in brackets are wrt *POET ($M = T$)* experiment. 'Avg' accuracy metric is biased towards 'Old' classes accuracy, A_{HM} is a good indicator of trade-off between 'New' and 'Old'.

	$\mathcal{UB}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(2)}$	$\mathcal{US}^{(3)}$	$\mathcal{US}^{(4)}$			
Method	Base	Avg	Avg	Avg	Old	New	Avg	A_{HM}
w/o prompts	88.4	74.5	66.3	49.5	39.2 (-20.1)	46.8 (-10.6)	39.9	42.7
w/o coupled optimization	88.0	82.8	75.3	65.8	56.5 (-2.8)	51.3 (-6.1)	56.1	53.8
w/o clustering loss	85.5 (-2.4)	81.6	74.3	64.5	**62.0**	18.2 (-39.2)	57.0	28.1
w/o QA update	87.9	82.8	77.4	69.1	59.4	52.8 (-4.6)	58.7	55.9
w/o sorting	88.2	82.2	75.2	68.8	59.9	46.6 (-10.8)	58.8	52.4
POET ($M > T$)	87.9	82.7	77.2	68.8	60.3 (+1.0)	54.4 (-3.0)	**59.8**	57.2
POET ($M = T$)	87.9	82.8	76.8	68.6	59.3	**57.4**	59.2	**58.3**

we get more flexibility in the stability-plasticity trade-offs depending on how many new prompts we attach. For $R = 6$, 'Old' retention is improved. In Table 4.3, we *add* T prompts, keeping prompt attachment constant and only vary prompt selection.

Prompt Attachment Mechanism. In Table 4.4, we keep our end-to-end optimization and ordered prompt selection as a constant and ablate prompt shape and attachment operator $f_p(.)$. Drawing a parallel with transformers which concatenate prompts along the token dimension, we conduct experiments concatenating prompts along the (i) temporal dimension of the skeleton input feature embedding $\mathbf{X_e}$ (*'CONCAT temporal'*) and (ii) feature dimension C_e (*'CONCAT feature'*). We find that addition works better than concatenation and cross attention. We also verify our hypothesis that selecting the same number of prompts as the input temporal dimension ($T = 64$ for NTU RGB+D and $T = 8$ for SHREC-2017) yields better results as compared to adding a single prompt frame to each input embedding frame (*'Addition $T' = 1$'* experiment).

Table 4.4: **Prompt Attachment Analysis (%, ↑):** *Adding* #prompts same as #input frames (T=64) is the best prompt attachment choice $f_p(.)$ (results on NTU RGB+D).

	$\mathcal{UB}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(2)}$	$\mathcal{US}^{(3)}$	$\mathcal{US}^{(4)}$			
Method	Base	Avg	Avg	Avg	Old	New	Avg	A_{HM}
CONCAT temporal dim., $T' = 64$	88.6	70.3	62.4	49.8	33.6	50.5	35.1	40.3
CONCAT feature dim., $T' = 64$	87.7	82.4	75.5	66.9	57.1	41.5	56.0	48.1
Cross Attention, $T' = 64$	88.1	2.2	2.0	1.8	0.0	19.8	1.65	0.0
ADD, $T' = 1$	88.7	73.3	62.7	45.5	33.7	47.0	34.8	39.3
ADD, $T' = 64$ (Ours)	87.9	82.8	76.8	68.6	59.3	57.4	**59.2**	**58.3**

Chapter 5

Discussion

5.0.1 Differences From Existing Continual Prompt Tuning Works

1. **Enable prompt tuning w/o extensive pretraining** As explained in Section 3.2.2, we optimize prompts, keys, and query adaptor in an end-to-end coupled optimization in order to update the query on unseen classes. Before us, CODA-P [41] also coupled their keys with the prompt pool. However, their coupling was done by using the cosine similarity to weigh all prompts in the pool, forming a 'soft prompt selection' instead of our 'ordered hard prompt selection'. Most importantly, these methods use a well pretrained query function which does not require updates. Finally, due to our lightweight backbone, there is significant forgetting from the classifier and we follow Algorithm 1 line 9 to mitigate the same.
2. **Spatio-temporal prompt offsets for GNNs:** Before us, PIVOT [46] trains separate spatial and temporal prompts on a pretrained CLIP for video continual learning. However, our **additive** spatio-temporal prompt offsets are a *simple plug-and-play for prompt tuning of any GNN architecture*, making it invariant to datasets, backbones and input sizes. POET is also a *prompt-only strategy* that does not require additional replay or regularization for continual learning.

5.0.2 Interpreting Prompt Offset Tuning of GNNs

Even though our inspiration for POET came from existing continual prompt tuning works [3, 41, 49, 50], our additive prompt offsets are open to interpretation. (i) Adding our selected prompts P_T to input feature embedding x_e acts like an input-dependent transformation for spatio-temporal joints. (ii) As our prompts have same size as x_e, it can also be thought of as a learned prompt encoding, bearing similarity with learnable position encoding works [12, 24, 26]. Our work is different as the purpose of prompt offsets is to dynamically condition the input for adapting the backbone continually, instead of learning positions. (iii) It also bears similarity with auto-decoders like DeepSDF [30] which learn latent codes for each data point (or style, shape) and use relevant codes along with a frozen decoder at inference.

Moreover, prompt tuning can also be thought of as a *parameter isolation* technique for continual learning [28, 34, 35, 38]. In these works, isolated sets of network parameters are learnt for each continual task, freezing the rest and task-ID is used to select the relevant set of parameters at inference. Prompts isolate parameters implicitly since the backbone is frozen and our learnable prompt selection mechasnism is trained to select the input dependent parameters that are relevant. In this context, POET's ordered prompt selection as seen in Fig. 3.3 learns to *isolate* the *relevant sequence of prompts for each input action sequence.*

Chapter 6

Conclusion and Future Works

6.1 Conclusion

The problem of continually adapting human action models to new user categories over time has gained prominence with the rising availability of XR devices. However, this setting poses unique challenges: (i) the user may be able to provide only a few samples for training, and (ii) accessing data from earlier sessions may violate privacy considerations. We hence propose a method based on prompt offset tuning to address this problem in this work. Prompt tuning to address learning over newer tasks has been attempted in recent years. However, these works have: (1) typically been designed for image-based tasks, (2) relied on strongly pre-trained transformer backbones, (3) required full supervision for new tasks, and (4) exclusively applied prompt tuning to transformer architectures. This work departs from these four characteristics. Our work demonstrates that prompt offset tuning is a promising option to evolve and adapt skeleton-based human action models to new user classes. The careful design of each component of the proposed methodology finds validation in the promising results across well-known skeleton-based action recognition benchmarks. Our ablation studies and analysis corroborate our design choices in our implementation. Looking ahead, it will be interesting to explore how our approach and its design choices adapt when a "generalist backbone" trained on a large corpus of action recognition data becomes accessible. Extending our method for differential privacy is another interesting direction of future work.

6.2 Broader Impact and Limitations

Privacy-aware human action recognition in extended reality devices: In order to protect users' privacy and security in head mounted devices, we incorporate privacy awareness in our continual learning setup: (i) By not storing any old class exemplar data or prototypes for replay in continual user sessions. All data is trained in a session and discarded henceforth. (ii) By using only 3D skeleton joint input modality for action recognition, we circumvent the visual privacy violation and user identity revelation in video-based HAR [14, 18, 21]. While we are a privacy-aware continual learning setting, we do not claim differential privacy and studying differential privacy in our prompts will be an interesting future work direction.

Data-free adaptation of action models for new user categories: Our key motivation for a *data-free* prompt tuning-based action recognition model adaptation to new categories over time is to maintain privacy of past sessions' data. However, this also has other advantages. Firstly, a data-free solution does not require a memory budget on the edge device for replay of old class data. Secondly, it has become commonplace to have access to large pre-trained backbones, but there is limited knowledge and often lack of access to such a dataset for such pre-training. Finally, prompts via a vector-quantized prompt pool memory offer a compact, learnable and automatically retrievable bottleneck of task-specific information (like in auto-decoders). Even if businesses can store and retrain their model on all previous data to continually adapt to new data, training large models incurs high carbon footprints [19]. Prompts offer a cost-efficient and low carbon footprint solution to retraining large models from scratch every time new data of value becomes available. Evaluating our design choices to large pre-trained models for skeletal data, as and when they become available, is another direction of future work.

Appendix A

Additional Results and Discussion: Thesis Committee Suggestions

Based on the thesis committee's suggestions, this section discusses primarily discusses how our approach POET mitigates catastrophic forgetting in our continual learning setting, robustness of our benchmarks, and limitations of our proposed approach.

A.0.1 How Does POET Mitigate Catastrophic Forgetting?

In this section we dive deeper into understanding how our proposed Prompt Offset Tuning methodology *mitigates catastrophic forgetting* in our few-shot class incremental setting for action recognition. We report mean and standard deviation across 10 experimental runs (10 sets of few-shots) for robustness ablation. The 'Old' only accuracy gives a direct comparison of the forgetting.

Our classifier expands in each continual session and fine-tuning the entire classifier on cross entropy loss of new classes leads to erosion of previous class weights in the classifier. **'C.U.P.'** refers to our **classifier update protocol** used to prevent forgetting from the classifier. We use a cosine normalized classifier for NTU RGB+D dataset (see Sec. B) which helps mitigate forgetting from the classifier as shown by the **11.7% $\downarrow$ in 'Old'** class performance of **'POET w/o C.U.P.'**, w.r.t. POET.

In **'POET w/o Prompts'**, we simply remove our prompts altogether, keeping the cosine normalized classifier to observe the prompt only affect. While 'Old' performance

Table A.1: **POET Ablation Table:** We exhaustively study the contribution of various POET components towards mitigating forgetting of old knowledge (Old) as well as learning new knowledge (New). **C.U.P.** refers to our **classifier update protocol** (cosine normalization).

	Prompt	Prompt Selection	Prompt Integration	$\mathcal{UB}^{(0)}$	$\mathcal{US}^{(4)}$			
Ablation			(Operator, #Prompts)	Base (↑)	Old (↑)	New (↑)	Avg (↑)	A_{HM} (↑)
POET (Ours)	✓	✓	ADD T	87.9	57.2 ± 1.0	**55.8** ± 5.9	**57.1** ± 1.1	**56.3** ± 3.2
Importance of prompts in POET								
POET w/o **Prompts**				87.9	60.8 ± 0.5 (+3.6)	18.4 ± 1.0 (-37.4)	57.3 ± 0.4	28.3 ± 1.1
POET w/o **C.U.P.**	✓	✓	ADD T	89.2	45.5 ± 1.4 (-11.7)	53.6 ± 3.7 (-2.2)	46.2 ± 1.2	49.1 ± 1.5
POET w/o **{Prompts, C.U.P.}**				88.4	40.0 ± 1.6 (-17.2)	51.0 ± 2.3 (-4.8)	44.8 ± 1.1	40.9 ± 1.4
POET w/o **{Prompts, C.U.P., Freezing}**				88.4	0.2 ± 0.5 (-57.0)	36.0 ± 10.1 (-19.8)	3.2 ± 0.8	0.3 ± 1.0
Impact of Ordered Selection								
POET w/o **Ordered Selection**	✓	✓	ADD T	88.2	59.9 ± 1.1 (+2.7)	52.5 ± 4.2 (-3.2)	59.3 ± 0.9	55.9 ± 2.2
Relative importance of our Prompt Selection versus Prompt Integration								
POET Selection w/o **Addition**	✓	✓	Cross-Attend T	82.9	57.0 ± 2.0 (-0.2)	31.0 ± 4.8 (-24.8)	54.9 ± 1.8	39.9 ± 4.0
POET Addition w/o **Selection**	✓	Standalone	ADD T	88.6	58.8 ± 3.2 (+1.6)	54.0 ± 3.4 (-1.8)	58.4 ± 1.2	56.2 ± 2.2
POET Addition w/o **Selection**	✓	Standalone	ADD 1	88.2	56.9 ± 1.1 (-0.3)	54.7 ± 5.3 (-1.1)	56.7 ± 1.2	55.7 ± 3.1
POET w/o **{Selection, Addition}**	✓	Standalone	Cross-Attend T	43.3	25.6 ± 1.3 (-31.6)	5.0 ± 4.1 (-50.8)	23.9 ± 1.2	7.8 ± 4.9

is slightly better, the backbone doesn't learn new knowledge as shown by **37.4% ↓ in 'New'** class performance. If we remove both prompts and classifier update protocol **'POET w/o Prompts, C.U.P.'**, we get vanilla Feature Extraction without any freezing or regularizing of classifier weights. It suffers in both 'Old' and 'New' classes. Further, as highlighted in the main paper as well, **freezing our backbone during the continual user sessions** $t > 0$ is of prime importance given our few-shot setting where the overfitting to few training samples exacerbates the overwriting of existing knowledge, leading to a complete washout of previous knowledge as seen by the **0.2%** 'Old' performance accuracy.

Notice, in the **absence of our ordered prompt selection, 'New'** performance **suffers by 3.3%** as the same prompts are selected and updated every time. In POET, we ensure the prompts get selected in the right temporal sequence, learning new temporal semantics for new classes hence enabling better adaptation to new classes.

Finally, within prompts, we replace our prompt selection mechanism completely by standalone T prompts or a single prompt. This shows how POET's spatio-temporal temporally consistent selection mechanism helps learn new orderings of T prompts as compared to attaching prompts without selecting them using an input dependent query. We also use **cross attention along the temporal dimension**

as attachment operator $f_p(.)$ and find addition consistently outperforms. **'POET w/o Selection, Addition'** experiment shows the importance of our design choices as unsuitable selection and integration mechanisms can be catastrophic to continual learning performance.

A.0.2 Backward Forgetting Metric (BWF)

Table A.2: **Backward Forgetting Metric (%, ↓):** Here we present BWF after user session $\mathcal{US}^{(4)}$. POET significantly mitigates forgetting on old classes.

	$\mathcal{US}^{(4)}$
Method	Forgetting (↓)
FE+Replay	-0.90 ± 0.65
FE	52.83 ± 2.09
FT	54.08 ± 9.17
FE, Freeze	34.02 ± 0.83
POET (Ours)	29.91 ± 0.34

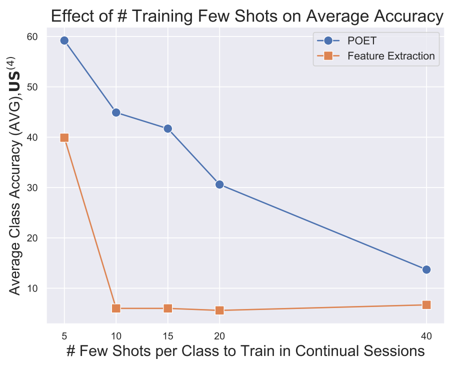
In addition to these results, we report the average forgetting metric $\mathbf{F_k}$ based on existing works in continual learning [4, 5] after the model has been trained continually on all user sessions (after $\mathcal{US}^{(4)}$) as:

$$F_k = \frac{1}{k-1} \sum_{j=1}^{k-1} f_j^k \tag{A.1}$$

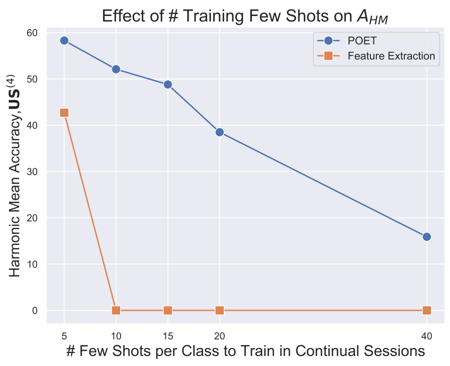
where f_j^k is the forgetting on previous task 'j' after the model is trained with all the new classes up till task k:

$$f_j^k = \max_{l \in \{1,\ldots,k-1\}} a_{l,B_{l,j}} - a_{k,B_{k,j}} \tag{A.2}$$

In effect, this is the same as difference in performance of each previous task 'j' at the end of $\mathcal{US}^{(4)}$ from when the task was first introduced (*'New'* in $\mathcal{US}^{(j)}$).



(a) Average Accuracy (AVG)



(b) A_{HM}

Figure A.1: Effect of variation in number of few-shot samples used for training in user sessions $\mathcal{US}^{(1)}$-$\mathcal{US}^{(4)}$ on stability-plasticity trade-offs in our few-shot continual setting.

A.0.3 Role of Number of Few-Shots in Continual Learning

We also vary the number of few shots used for training per new class in continual sessions in Fig A.1. The Feature Extraction baseline reduces to zero A_{HM} because the 'Old' class performance reduces to zero. We find our prompt offsets in POET (=FE+Prompts) significantly help retain old class performance as compared to Feature Extraction, without any explicit forgetting measure due to our ordered prompt selection and clustering loss.

It can be noted that our method is particularly well suited for very few training samples per user (< 15) and may require additional explicit regularization or freezing of prompt pool to mitigate forgetting for large number of training samples (> 20).

A.0.4 Robustness of Our Benchmarks: Reporting Mean Across 10 Unique Sets of Few-Shots

In this section, we evaluate the robustness of our main results on NTU RGB+D activity recognition benchmark 4.1 to the choice of few-shots used for training each new class. In Table 4.1, we had reported results for a specific set of few-shots used across experiments. In Table A.3, we report mean and standard deviation across 10 unique sets of few-shots used for training. *We find consistent performance and* *trends in both the tables.* We note high standard deviation only in the 'New' accuracy as these classes are trained using only 25 unique samples (5 classes, 5 shots each) and 'New' performance depends on the choice of few-shots used. We had already reported mean and standard deviation across 5 runs for the hand gesture recognition benchmark in Table 4.2.

Table A.3: **Activity Recognition Results (%, ↑), Comparison with SOTA:** NTU RGB+D [39] dataset on CTR-GCN [7] backbone. After training on each incremental task, we report Average of all classes seen so far ('Avg'). We also report (i) A_{HM}, (ii) old classes accuracy ('Old'), (iii) new classes accuracy ('New') in the last session. We report Mean and STD across 10 sets of 5-shots. *POET achieves the best stability-plasticity trade-off across all baselines indicated by the $A_{HM} = 56.3\%$. POET also has the highest Avg across all user sessions outside of upper bound baselines in orange.*

	$\mathcal{UB}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(2)}$	$\mathcal{US}^{(3)}$	$\mathcal{US}^{(4)}$			
Method	Base (↑)	Avg (↑)	Avg (↑)	Avg (↑)	Old (↑)	New (↑)	Avg (↑)	A_{HM} (↑)
Upper Bounds								
Joint (Oracle)	88.4	79.0	71.0	66.8			**63.5**	
Joint POET (Oracle)							**67.2**	
FE, Task-Specific^	88.4	70.1 ± 2.6	52.5 ± 5.8	44.8 ± 5.0	70.3 ± 2.1	46.7 ± 2.0	NA	NA
FE+Replay	88.4	82.4 ± 1.1	78.2 ± 1.2	74.5 ± 1.2	**73.1 ± 1.0**	43.3 ± 3.3	**70.6 ± 1.2**	54.3 ± 2.6
Continual Linear Probing								
FE	88.4	72.0 ± 1.1	60.4 ± 2.4	47.7 ± 2.1	40.0 ± 1.6	51.0 ± 2.3	40.9 ± 1.4	44.8 ± 1.1
FE, Frozen	88.4	76.1 ± 1.0	52.4 ± 4.1	38.3 ± 2.7	28.4 ± 1.6	22.4 ± 4.5	27.9 ± 1.4	24.8 ± 3.0
FE+Replay†	88.4	72.0 ± 1.5	59.5 ± 4.0	58.7 ± 2.8	56.7 ± 2.5	34.7 ± 5.6	54.9 ± 2.7	42.8 ± 4.4
FT	88.4	6.2 ± 1.4	4.3 ± 1.5	2.8 ± 1.0	0.2 ± 0.5	36.0 ± 10.1	3.2 ± 0.8	0.3 ± 1.0
Standard Continual Learning								
LWF [25]	88.4	6.2 ± 1.5	2.8 ± 0.7	3.7 ± 1.3	0.0 ± 0.0	38.9 ± 8.8	3.2 ± 0.7	0.0 ± 0.0
EWC [17]	88.4	6.6 ± 1.5	4.1 ± 1.4	3.1 ± 0.9	0.0 ± 0.0	42.1 ± 9.5	3.5 ± 0.8	0.0 ± 0.0
Experience Replay	88.4	35.1 ± 8.3	50.6 ± 5.0	60.6 ± 5.4	54.6 ± 6.5	43.7 ± 14.6	53.7 ± 7.1	47.8 ± 11.2
Experience Replay†	88.4	6.2 ± 1.5	9.0 ± 2.6	11.2 ± 3.0	10.9 ± 2.6	34.6 ± 7.9	12.9 ± 3.0	16.3 ± 3.5
LUCIR [15]	87.9	4.3 ± 2.1	4.1 ± 1.3	2.7 ± 0.8	0.2 ± 0.4	26.0 ± 9.2	2.3 ± 0.9	0.4 ± 0.8
Continual Prompt Tuning								
CODA-P [41]*	87.4	76.1 ± 1.0	66.7 ± 1.3	58.6 ± 2.7	56.5 ± 2.9	0.5 ± 0.4	51.8 ± 2.7	1.1 ± 0.7
L2P [50]*	88.6	78.9 ± 0.1	71.0 ± 1.0	64.2 ± 0.1	62.0 ± 0.7	0.0 ± 0.0	56.8 ± 0.6	0.0 ± 0.0
APT [3]^	86.6	27.3 ± 1.6	30.8 ± 3.4	37.6 ± 2.3	NA	33.4 ± 2.0	NA	NA
POET (Ours)	87.9	**82.3 ± 0.6**	**76.8 ± 0.9**	**68.4 ± 0.7**	**57.2 ± 1.0**	**55.8 ± 5.9**	**57.1 ± 1.1**	**56.3 ± 3.2**

Appendix B

Additional Results

B.1 Analyzing Prompt Tuning Architectural Choices

B.1.1 Classifier Update Protocol in $\mathcal{US}^{(t)}, t > 0$

Existing related work such as in few-shot class-incremental learning learn non-parametric classifiers by extracting class-mean prototypes, and do not expand the classifier with any new weights. However, we need to expand and train the classifier in order to obtain the error gradients for updating the prompts. We observe that a key source of forgetting is from the classifier as the logits tend to become biased towards the few-training samples of new classes. For SHREC/DG-STA, we observe that fine-tuning the entire classifier leads to a significant drop in performance of old classes (experiment 'FE' in Table 4.2 of main paper). This is because the frozen backbone is trained only on 1146 training samples from the 8 base session classes. Hence, the SHREC backbone exhibits low stability for retaining old knowledge, when updated on new data in subsequent sessions $t > 0$. To alleviate this, we use a simple classifier update trick wherein after expanding the classifier in every continual session $t > 0$, we turn the gradients of the old parameters in the classifier to zero before the error backpropagation. This trick has also been shown to work in prior continual prompt tuning works L2P [50], CODA-P [41].

Table B.1: **Experimenting with different classifiers for NTU RGB+D.** Here, we report the task-specific accuracy for each task the model has learnt so far, after learning every new task. Notice the sharp **forgetting of the *intermediate* 'New-Old'** tasks in regular classifier and our improvement using a cosine normalized classifier. Note, we are using a dynamically expanding parametric classifier in all experiments.

	$\mathcal{US}^{(0)}$	$\mathcal{US}^{(1)}$		$\mathcal{US}^{(2)}$			$\mathcal{US}^{(3)}$				$\mathcal{US}^{(4)}$					
Activity	Base	$\mathcal{US}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(2)}$	$\mathcal{US}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(2)}$	$\mathcal{US}^{(3)}$	$\mathcal{US}^{(0)}$	$\mathcal{US}^{(1)}$	$\mathcal{US}^{(2)}$	$\mathcal{US}^{(3)}$	$\mathcal{US}^{(4)}$	**Avg**
Regular, Freeze	89.2	73.5	**72.5**	71.3	2.1	**63.8**	65.3	0.0	0.0	52.9	58.7	0.0	0.0	0.0	48.9	43.1 (-16.1)
Regular, Tune	89.2	80.9	64.9	67.4	22.1	34.4	57.2	6.32	24.3	26.9	45.0	4.6	**20.9**	13.3	21.4	35.2 (-24.0)
Cosine (Ours)	87.9	**84.9**	65.3	**83.2**	**56.0**	45.8	**78.2**	**36.3**	**34.5**	**60.0**	**71.3**	**18.5**	19.8	**46.2**	**57.4**	**59.2**

We observe that this trick proves sub-optimal in the NTU RGB+D (CTR-GCN) dataset. In Table B.1, we evaluate performance on each usr session individually after learning every new user session. We observe that both, freezing ('**Regular, Freeze**') or finetuning ('**Regular, Tune**') of old class parameters in classifier suffers from poor stability-plasticity trade-offs. *We observe that the intermediate tasks ($\mathcal{US}^{(1)}$, $\mathcal{US}^{(2)}$, $\mathcal{US}^{(3)}$) learnt using few shot data particularly suffer severe forgetting as soon as the next task is learnt.* This is not the case for $\mathcal{US}^{(0)}$ performance in incremental sessions $t > 0$ because the base feature extractor is trained on sufficient data in the activity benchmark (26,731 samples) and frozen henceforth, retaining performance on the base task $\mathcal{US}^{(0)}$ even in $t > 0$. We call this the **'New-old forgetting'**. To address the biasing of logits towards new classes, we replace the main model's linear classification layer f_c with a cosine normalization classifier θ_c^T as:

$$p_{(x)} = \frac{exp(\eta < \theta_{c_i}^T f_g(x) >)}{\Sigma_j exp(\eta < \theta_{c_j}^T f_g(x) >)} \tag{B.1}$$

where η is a learnable scale parameter we learn in the base session and freeze in subsequent sessions. Also, in incremental sessions, we initialize the new-class parameters in the classifier as mean of previous class parameters. Notice the significant boost in performance across all tasks $\mathcal{US}^{(i)}$ in the cosine normalized classifier in Table B.1.

B.1.2 Analyzing Where to Attach Prompts

We study the impact of attaching our prompt offsets at different main model layers in CTR-GCN in Figure B.1. As mentioned in Sec C.0.2, output feature dimensionality (number of channels) of the first four layers in CTR-GCN is the same, $C_e = 64$. We desire that the feature dimension size of the (i) prompt parameters, (ii) key parameters, and (iii) query adaptor be consistent with the main model feature embedding dimension at the layer being prompted. Hence, we only prompt the first four layers for a fair comparison as results may vary with variation in size of feature dimension being prompted. We created a separate 30% validation set from the training data of $t \geq 1$ classes for this analysis. Our findings in Figure B.1 indicate that the highest 'New' task performance at the end of all four user sessions is achieved by prompting layer L1. We select layer L1 across all experiments in the paper.

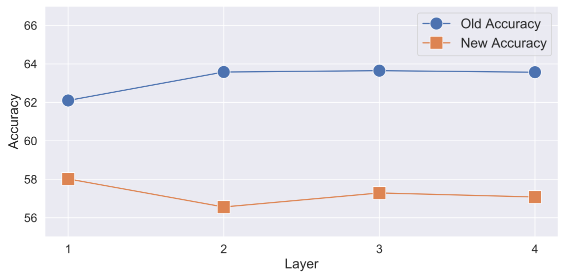


Figure B.1: Empirical analysis to study the impact of the layer at which our prompt is attached. Y-axis shows 'Old' and 'New' classes accuracy after Task 4 (after learning all 60 classes). We add a prompt of size $P_{T'} \in \mathcal{R}^{64,25,64}$ to different layers $\{1, 2, 3, 4\}$ of CTR-GCN, evaluated on the NTU RGB+D validation set. We select layer L1 due to its high performance on new classes.

B.2 Additional Results

B.2.1 Impact of Prompts in POET

In Figures B.2 and B.3, we qualitatively study the impact of prompts by removing prompts from POET (the corresponding feature extraction baselines 'FE' for NTU RGB+D and 'FE++' for SHREC). Prior continual learning works rely on

ImageNet21K pretrained ViT [37] (L2P [50], Dual-P [49], CODA-P [41]) or WebImageText pretrained CLIP model [33] (S-Prompts [48], PIVOT [46]) for prompt tuning. In Fig. B.4, we show the significant disparity in scale of pretraining dataset as we use only base class dataset from the benchmark itself for pretraining and every new user session sees a non-overlapping set of classes. Despite of this, POET shows promising results.

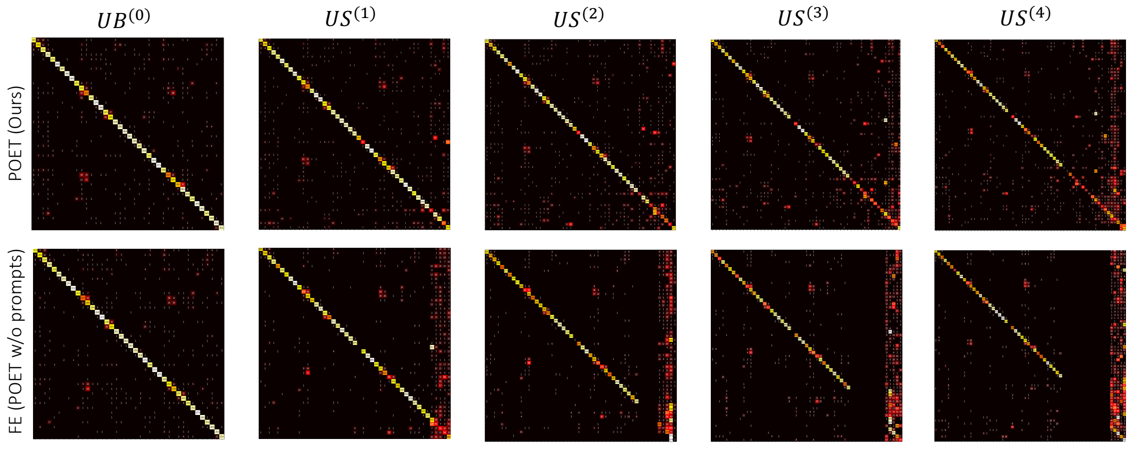


Figure B.2: *Confusion matrices showing the impact of our prompt offsets across 4 user sessions in* ***NTU RGB+D activity recognition benchmark***. We compare confusion across new and old actions in POET and POET w/o prompts ablation. Starting from 40 default classes in $\mathcal{UB}^{(\prime)}$, we learn new classes $\{\mathcal{US}^{(1)}$: sneeze, stagger, fall, touch head, touch chest$\}$; $\{\mathcal{US}^{(2)}$: touch back, touch neck, nausea, user fan, punch$\}$; $\{\mathcal{US}^{(3)}$: kick, push, pat back, point finger, hug$\}$; $\{\mathcal{US}^{(4)}$: give, touch pocket, handshake, walk towards, walk away$\}$. Prompts enable retention of the intermediate **'New-Old'** classes very well, while FE gets heavily biased towards the new classes (see last 5 columns in each matrix).

B.2.2 Stability-Plasticity Trade-offs via New/Old performance

In Fig. B.5, we study the average accuracy of only new classes (New) and only old classes (Old) after every user session in activity recognition benchmark. As stated in Sec **??** of main paper, we observe that (i) any method that does not freeze the backbone such as knowledge distillation (LWF and LUCIR), prior-based regularization methods (like EWC), or vanilla Fine-tuning baselines (FT) completely forget old performance from $\mathcal{US}^{(1)}$ itself. POET is short of only FE+Replay which is an upper bound. (ii) Any existing prompt tuning work which does not update their query function such as L2P, CODA-P, APT in graph (B), is unable to learn new classes well.

In Fig. B.6, we observe similar trends. Additionally, (i) ALICE retains old knowledge very well as it does not use a parametric classifier for incremental sessions. However, ALICE is unable to learn new classes well. We also observe that (ii) DG-STA backbone has very high plasticity when fine-tuned on new tasks (see New performance of FT and LWF in graph B). But these baselines while plastic, completely forget old classes. POET achieves the best stability-plasticity trade-offs (indicated by A_{HM} in main paper).

B.2.3 Robustness to class order in user sessions

The default continual order in which different gesture classes appear till now was: $\{\mathcal{US}^{(0)}$: Grab, Tap, Expand, Pinch, Rotate-CW, Rotate-CCW, Swipe-R, Swipe-L$\}$ $\rightarrow \{\mathcal{US}^{(1)}$: Swipe-U, Swipe-D$\} \rightarrow \{\mathcal{US}^{(2)}$: Swipe-x, Swipe-+$\} \rightarrow \{\mathcal{US}^{(3)}$: Swipe-v,

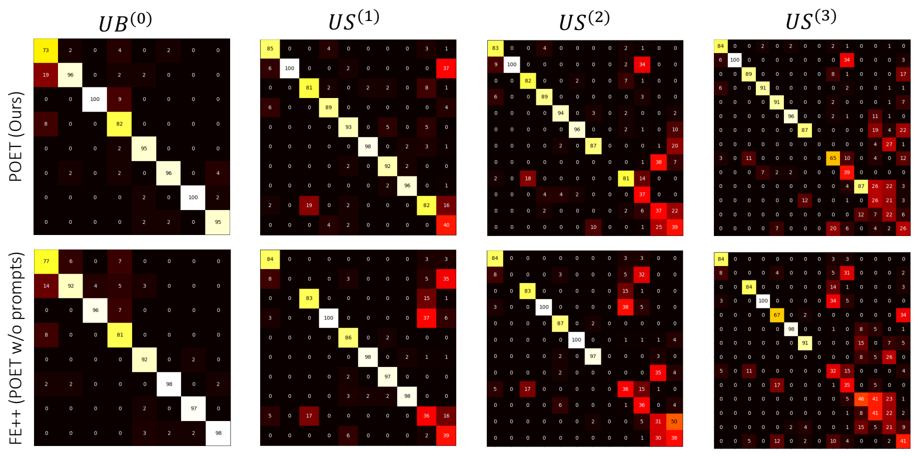


Figure B.3: *Confusion matrices showing the impact of our prompt offsets across 3 user sessions in* ***SHREC 2017 gesture recognition benchmark****.* $\mathcal{US}^{(0)}$ has hand gestures grab, tap, expand, pinch, rotate clockwise, rotate counter-clockwise, swipe right, swipe left$\}$. $\{\mathcal{US}^{(1)}$: swipe up, swipe down$\}$, $\{\mathcal{US}^{(1)}$: swipe-x, swipe-+$\}$, $\{\mathcal{US}^{(1)}$: swipe-v, shake$\}$. Even though the classes are fine-grained, the prompts help retain old class semantics well.

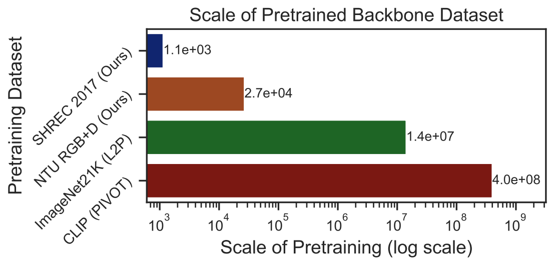
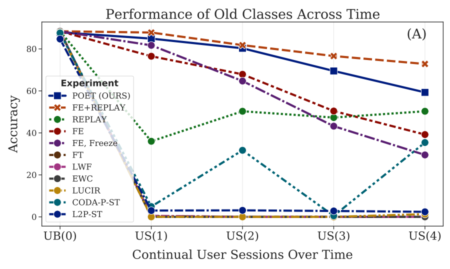
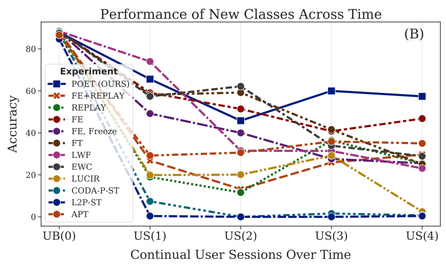


Figure B.4: **Scale of pretraining used for the prompt tuning backbones.** For (Our) benchmarks on NTU RGB+D and SHREC 2017, numbers represent the base class training data used. Our POETs continually learn new actions mitigating catastrophic forgetting, without massive pretraining, and only rely on prompts.

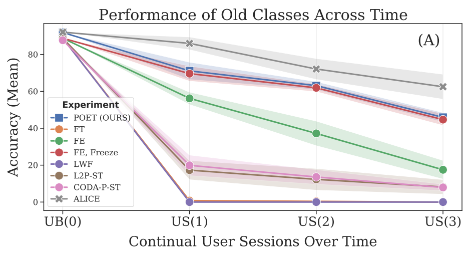


[t]0.5

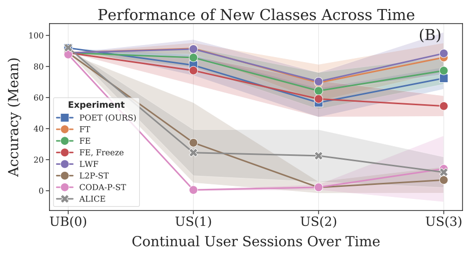


[t]0.5

Figure B.5: Old and New class performance for NTU RGB+D.



[t]0.5



[t]0.5

Figure B.6: Old and New class performance for SHREC 2017. Reporting Mean and STD over 5-sets of user few-shots.

Shake}. In Fig B.7, we swap the base and incremental classes in SHREC benchmark to a new ordering: $\{\mathcal{US}^{(0)}$: Swipe-R, Swipe-L, Swipe-U, Swipe-D, Swipe-x, Swipe-+, Swipe-v, Shake$\} \rightarrow \{\mathcal{US}^{(1)}$: Grap, Tap$\} \rightarrow \{\mathcal{US}^{(2)}$: Expand, Pinch$\} \rightarrow \{\mathcal{US}^{(3)}$: Rotate-CW, Rotate-CCW$\}$. We find **'POET'** gives an $AVG = 57.3$ as compared to **'ALICE'**, $AVG = 55.9$ and **'FE, Freeze'**, $AVG = 55.3$ at the end of 3 user sessions even though our backbone is now trained on a different set of classes and we completely reversed the semantic order in which prompts learn different fine-grained gesture classes. This demonstrates robustness to variation in continual class order across tasks.

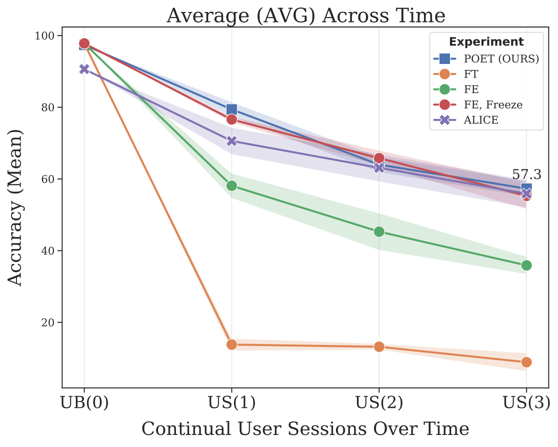


Figure B.7: Here, we change the set of classes in each session from the default order seen before. We report average accuracy of all classes learnt by the model after adding each new session. New ordering: $\{\mathcal{US}^{(0)}$: Swipe-R, Swipe-L, Swipe-U, Swipe-D, Swipe-x, Swipe-+, Swipe-v, Shake$\} \rightarrow \{\mathcal{US}^{(1)}$: Grap, Tap$\} \rightarrow \{\mathcal{US}^{(2)}$: Expand, Pinch$\} \rightarrow \{\mathcal{US}^{(3)}$: Rotate-CW, Rotate-CCW$\}$.

B.2.4 Ordered Key Index Selection $(s_i)_{i=1}^{T}$: Qualitative Results

In Section **??** of main paper, we explained our sorted *ordered key index selection* for selecting temporally consistent prompts from the pool. In Figure 3.2 of main paper, we visualized $(s_i)_{i=1}^{T}$ ordering statistics at the task-level. In Figure B.8, we investigate class-wise ordering statistics at inference time, performing inference after each continual task. We find that the ordering statistics are not only disparate for different classes, the statistics for a class remain consistent across continual tasks. The temporal discriminability in these studies further establishes that our learnable prompt selection mechanism is temporally with the 4D input skeleton. Finally, we also demonstrate instance-level statistics in Figure B.9 and Figure B.10 as our prompt mechanism is designed to select relevant (temporal ordered) prompts conditioned on every input instance. This means that our method does not depend on disparate task-wise or class-wise dataset splits and can even be used for online continual learning settings that do not have clear task boundaries.

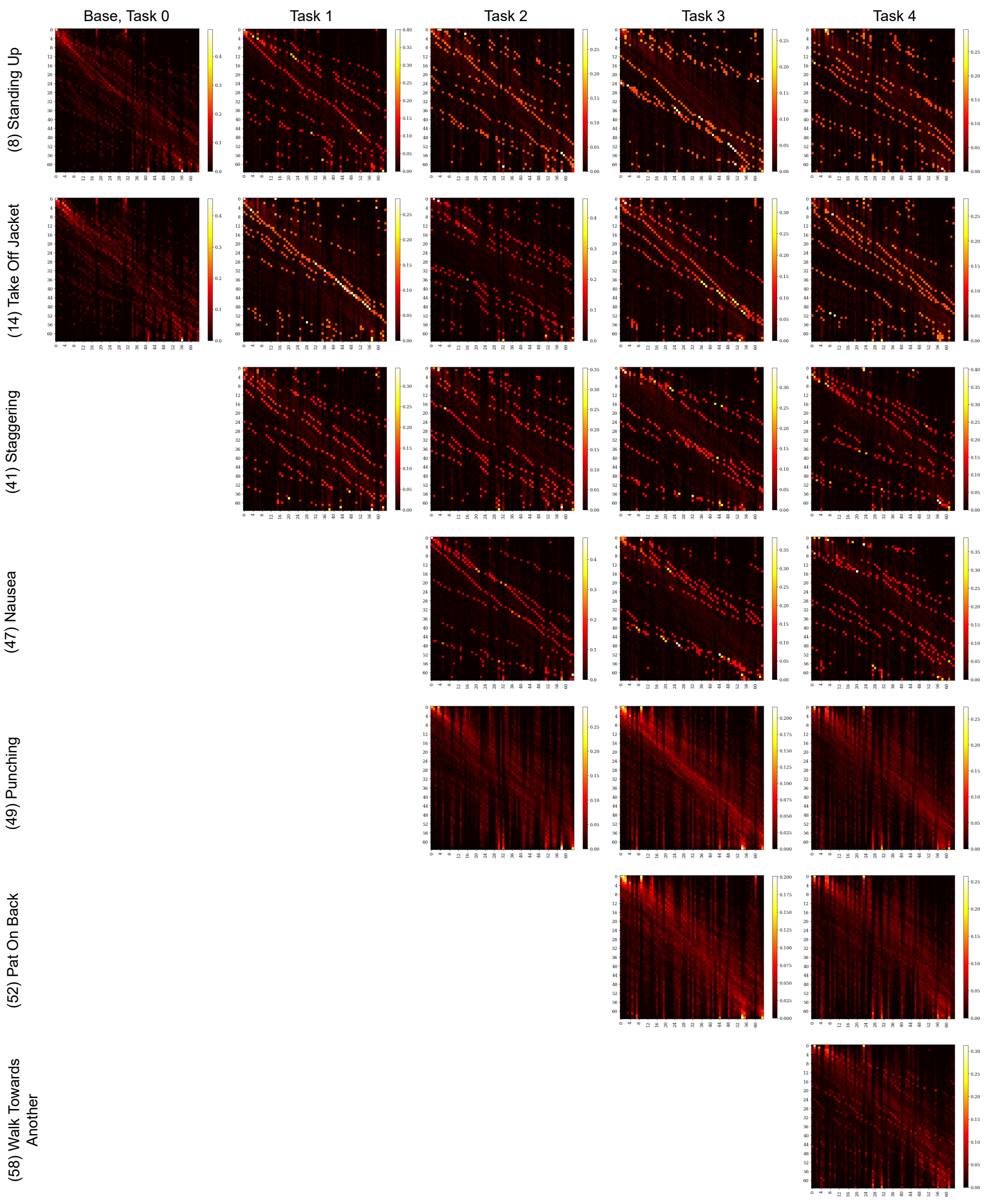
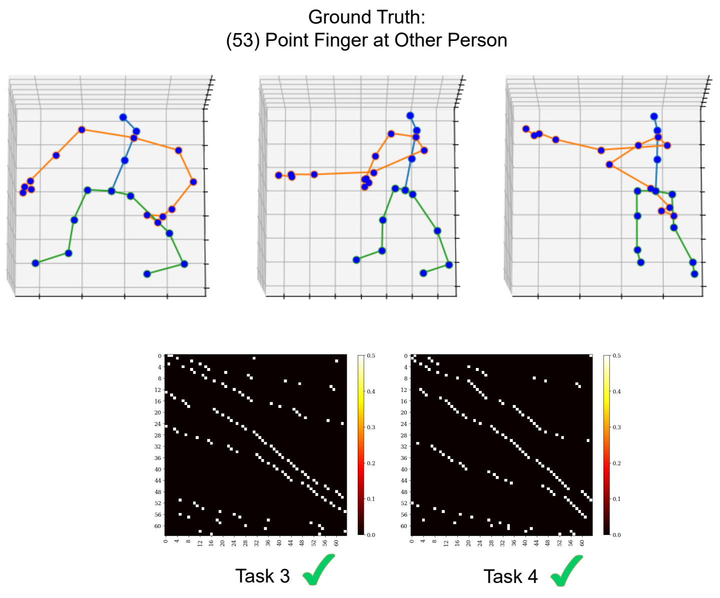


Figure B.8: **Class-level Ordered Prompt Selection using POET, Task t is same as $\mathcal{US}^{(t)}$:** Here, we analyse the ordered prompt selection statistics for our method for different classes at **test time**. For each class shown in column 1, we plot the prompt selection order at test time for each continual model checkpoint (starting from when that class was first introduced to the continual system and checking after updating the model on new classes each time). We **observe** that class-wise selection statistics are retained even after Task 4 (notice the plots for different classes in Task 4). Even for classes introduced as part of the same task (class 47, Nausea and class 49, Punching both introduced in Task 2), their ordered prompt selection is unique and consistent even after updating the model on new data in subsequent continual sessions.



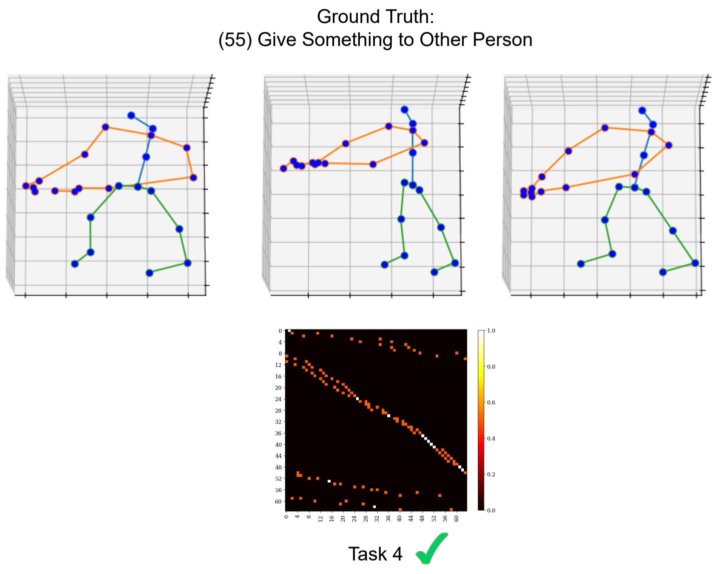
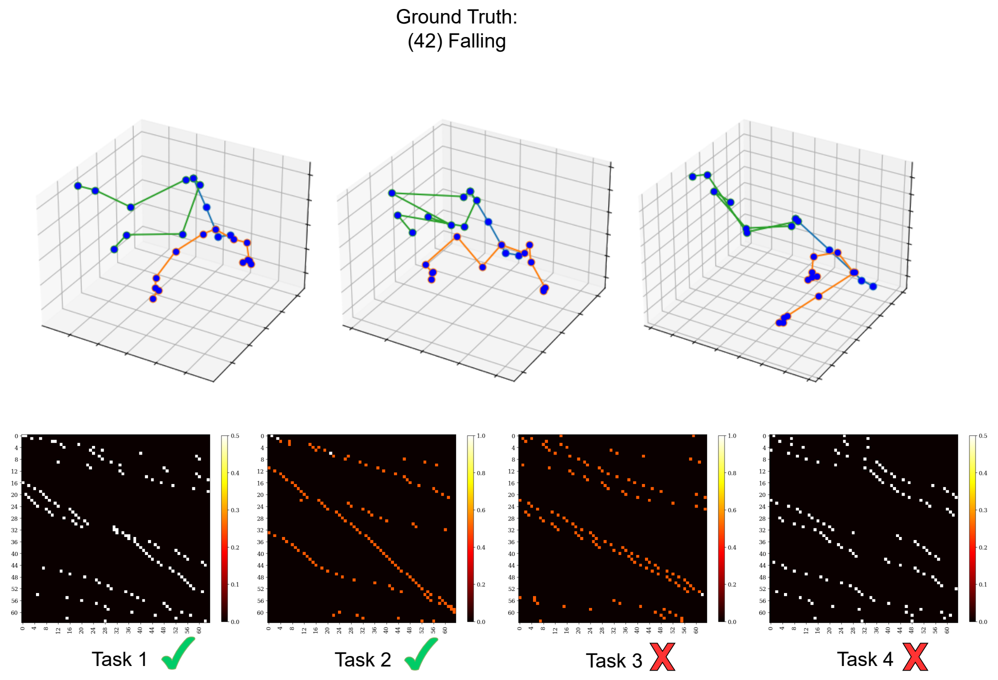


Figure B.9: **Instance-Level Ordered Prompt Selection using POET:** Our proposed method POET is an input instance-based prompt tuning approach for FSCIL, as the prompts are selected conditioned on each input instance itself. Hence, here we study instance-level prediction on the test set. The sample of class *Point Finger, class ID 53* is evaluated after $\mathcal{US}^{(3)}$ and $\mathcal{US}^{(4)}$ as the class was added to the model in $\mathcal{US}^{(3)}$. The sample of class *Give Something, class ID 55* is continually learnt and evaluated after $\mathcal{US}^{(4)}$. We point out the unique ordered key index sequence for the 2 instances, which could have been easily confused by the model due to their semantic similarity. The ordering matrix for *Point Finger* remains consistent across tasks, even after adding 5 new classes in $\mathcal{US}^{(4)}$.



Wear a Shoe	3.09
Falling	3.08
Hug	2.92

Wear a Shoe	2.86
Falling	2.76
Walk Towards	2.67

Figure B.10: **Instance-Level Ordered Prompt Selection using POET:** We also show a failure case of our proposed approach. After learning the class *Falling* in $\mathcal{US}^{(1)}$, we evaluate it after every new continual task. Even though it correctly predicts a test set instance in $\mathcal{US}^{(1)}$ and $\mathcal{US}^{(2)}$, it tends to get confused by the class *Wearing a Shoe* at $\mathcal{US}^{(3)}$ and $\mathcal{US}^{(4)}$. Notice, this coincides with a disruption in the ordering statistics.

Appendix C

Implementation and Training Details

C.0.1 Training Details

We use the same hyperparameters across all experiments for the NTU RGB+D activity recognition benchmark (in Tables ??, 4.3, 4.4 of the main paper, and Supplementary Table B.1 and Figure B.5). In the SHREC 2017 gesture benchmark also, all our experiments follow the exact same hyperparameter combination and learning strategy in Table 4.2 and Figure B.6.

Activity recognition on NTU RGB+D benchmark. In the base session $\mathcal{UB}^{(0)}$, we train the CTR-GCN [7] backbone for 50 epochs with initial LR=0.1. We use a batch size of 64 as in the original paper. Every continual user session $\mathcal{US}^{(t)}$ is trained for 5 epochs with an initial LR=0.1.

Gesture recognition on SHREC 2017. We train DG-STA [6] model in the base session $\mathcal{UB}^{(0)}$ for 300 epochs and initial LR=0.001, using batch size 32 and dropout set to 0.2 (default hyperparameters from the DG-STA paper). It is updated for 30 epochs in each user session $\mathcal{US}^{(t)}$, starting with initial LR=0.01.

We select higher initial LRs in continual sessions $t > 0$ because starting with a lower learning rate as compared to base session (as is standard practice in continual learning to prevent catastrophic forgetting) renders limited plasticity and the model is completely unable to learn new knowledge. Our choice enables learning of new knowledge from the few user samples, and we can study the model's stability-plasticity trade-offs, optimizing for a balance between the two. The continual session learning rates given above are used to update the (i) the classifier f_c, (ii) selected prompts in $\mathbf{P}$, and (iii) selected keys in $\boldsymbol{K}$. But for updating query adaptor f_{QA}, we use a learning rate of 0.01, in $t > 0$ for both benchmarks. At the same time, we freeze all other layers in the query model. We find this adapts query adaptor to new tasks without overwriting existing base knowledge. In each continual session $t > 0$, we use a batch size of 25 for NTU RGB+D, as there are 5 new classes each having 5 training samples (single batch per epoch). Similarly, batch size is 10 for SHREC 2017 with 2 new classes with 5 training samples each. For the clustering loss coefficient in Eq. 3.6, we use $\lambda = 0.1$ for all experiments.

Algorithm 3 Initialization & Training of Prompts, $t = 0$

```
Input: Model $f^P(.) = f_c^P \circ f_g^P \circ f_e^P(.)$, pretrained only on base $\mathcal{UB}^{(0)}$ data.
Initialize:
  1. Main model $f$ as: $f_e \leftarrow f_e^P, f_g \leftarrow f_g^P, f_c \leftarrow f_c^P$.
  2. Prompt pool $\mathbf{P} = \{\boldsymbol{P_j}\}_{j=1}^T$, Keys $\boldsymbol{K} = \{\boldsymbol{k_j}\}_{j=1}^T$ from $\mathcal{U}(0,1)$.
  2. Query function model $f_q$ as: $f'_e \leftarrow f_e^P, f'_g \leftarrow f_g^P$. $f_{QA}$ is randomly initialized.
Freeze: Query function layers $f'_g, f'_e$.
for epochs and batch in base dataset $(\mathbf{X}_i^0, \mathbf{y}_i^0)_{i=1}^{|\mathcal{D}^{(0)}|}$ do
  1. Get query feature $\boldsymbol{q}$ (Eq. 3.4) ; Compute $\gamma(.)$ b/w query $\boldsymbol{q}$ and keys $\boldsymbol{K}$
  2. Sort $\gamma(.)$; Get ordered key index sequence $(s_i)_{i=1}^T$ (Eq. 3.7)
  3. Read pool memory $\mathbf{P}$ in order $(s_i)_{i=1}^T \rightarrow$ Get prompt offsets $\mathbf{P_T}$
  4. Get $\mathbf{X_e}$; Add $\mathbf{P_T}$ to it (Eq. 3.8); get prediction $\mathbf{y}$ from prompted input (Eq. 3.1)
  5. Use cross entropy loss (Equation 3.5) to update prompt associated parameters $f_{QA}, \boldsymbol{K}, \mathbf{P}$ and all main model parameters $f_g, f_c, f_e$.
  6. Use clustering loss (Equation 3.6) to update $f_{QA}, \boldsymbol{K}$.
end
Freeze: Main model feature extractor $f_g$, input embedding layer $f_e$ for time $t > 0$.
```

C.0.2 Base Session $\mathcal{UB}^{(0)}$: Prompt Instantiation and Training

Prompt instantiation, CTR-GCN: CTR-GCN is a spatio-temporal graph convolutional network architecture with 10 multi-scale temporal convolutional (TCN-GCN) layers followed by an average pool over the spatial and temporal dimensions, and a final linear classification layer. The output feature dimensionality of the first four layers (L1-L4) is 64 channels, next four (L5-L8) is 128 and final two layers (L9 and L10) have 256 channels. An input skeleton sequence has 64 temporal frames, each consisting of 25 body joints, such that $x \in \mathcal{R}^{64\times25\times3}$.

The input embedding after layer L1 is $x_e \in \mathcal{R}^{64\times25\times64}$, such that $C_e = 64$. We start from a prompt pool $\mathbf{P} = \{\boldsymbol{P_j}\}_{j=1}^{T}$ of size $M = T = 64$. Each prompt in the pool $P_j \in \mathcal{R}^{25\times64}$ is designed to match the spatial and feature dimensions of the input embedding x_e. There are $M = 64$ keys, each having feature dimension $\boldsymbol{k_j} \in \mathcal{R}^{64}$. Query adaptor f_{QA} maps a feature embedding of size 256 (from the last layer of query model f'_g) to $C_e = 64$ for feature dimension compatibility with the keys and prompts. We select $T = 64$ prompts from the pool. After instantiating the prompt and key parameters, we train the prompt pool $\mathbf{P}$, keys $\boldsymbol{K}$, query adaptor f_{QA}, along with all main model parameters f_g, f_c, f_e on the base session data $\mathcal{D}^{(0)}$ as per Algorithm 3.

Prompt instantiation, DG-STA: DG-STA is a fully connected graph transformer architecture with multi-head spatial and temporal attention layers. For every input skeleton hand gesture sequence, we use 8 temporal frames, each having 22 hand joint coordinates such that input is $x \in \mathcal{R}^{8\times22\times3}$. Output of the transformer's input embedding layer f_e is $x_e \in \mathcal{R}^{8\times22\times128}$. As DG-STA expects a fully connected spatio-temporal graph across all joints in all frames, this is reshaped to a size $x_e \in \mathcal{R}^{176\times128}$ before passing it to the first attention layer of the transformer. We add our prompt to this reshaped embedding. We start from a pool of size $M = 8$ prompts. As DG-STA is a transformer architecture, the output feature dimensionality remains constant (at 128) and the query adaptor input and output dimensions are the same ($C_e = 128$). The base session here is also trained using Algorithm 3.

C.0.3 Additional Dataset Details

The NTU RGB+D dataset has been collected from Microsoft Kinect V2 sensors from three different camera viewpoints by annotating 40 human users. The 60 classes consist of 40 daily action categories (drinking water, reading, writing, etc.), 9 health-related actions (coughing, sneezing, headache, etc.), and 11 mutual actions (handshaking, pushing another person, walking towards another person, etc.). NTU RGB+D has 40,320 training and 16,560 testing samples across 60 classes. We use the first 40 daily action classes as the base model data, and update on $5 \times 5 = 25$ training samples in each of the 4 incremental session. The base model is trained on 26731 training samples from the 40 base classes.

The SHREC 2017 dataset has 14 fine-grained hand gestures captured using the short-range Intel Real Sense Depth, from 28 human subjects in second-person view. There are 1980 training and 840 testing samples. The base model for this benchmark is trained on 1146 training samples from 8 classes, and updated on $2 \times 5 = 10$ training samples in each of the 3 incremental session.

Given the relatively small sizes of both datasets, we follow a class-incremental setting in our work, viz., our user sessions include new classes over time, not necessarily new users. Evaluating our work on user-specific continual streams is left as future work for larger datasets where this is feasible.

Class splits in user sessions. There are two existing continual benchmarks for 3D skeleton-based action recognition: The experimental protocol of NTU RGB+D based class incremental benchmark [22] involves a single continual session, learning 50 classes in the base session, and 10 new classes in a single-incremental session. We consider this to be limiting and too simplistic to study our problem setting on. Moreover, their code is not publicly available. On the other hand, the more recent data-free class incremental learning for hand gesture benchmark [1] learns 8 classes in base session and updates the model on a single class over 6 incremental sessions. We believe that this too does not lend itself to our setting, when there are only 5-shots for training each continual class and the base model is trained on small-scale data.

C.0.4 Adaptation of Baselines to Problem Setting

Adapting Learning to Prompt (L2P): We experiment with selecting $T' = 4$ and $T' = 64$ (same as POET) for this comparison on CTR-GCN backbone. We find the results with 4 prompts marginally better, hence we report those in Table 3 of main paper. We experiment with both temporal concatenation and spatio-temporal concatenation followed by remapping. For DG-STA, we select 8 prompts from a pool of size $M = 10$, each prompt (22, 128); and concatenate this prompt of size (8, 22, 128) along the spatio-temporal dimension 176 of the input embedding (176, 128). We map this back to (176, 128) using a fully connected layer. In experiments where we remap using FC layer, we update this layer as well in future incremental sessions. To update the expanded classifier, we make logits of previous classes -inf, same as the classifier training protocol used in L2P, Dual-P and CODA-P.

Adapting CODA-P: For activity recognition on CTR-GCN backbone, we construct a 4 dimensional CODA-prompt of size (100, T', 25, 64), such that the prompt component dimension of 100 gets collapsed after weighing and we can concatenate prompt of size (T', 25, 64) along the temporal or rolled out spatio-temporal dimension (same as L2P). The size of memory buffer (T') is kept consistent with L2P experiments.

For gesture recognition on DG-STA backbone, [41] tunes a ViT-B/16 pre-trained on ImageNet-1K architecture instead of prompt tuning. This refers to concatenating half of the prompt to K and V of the MSA layer instead of concatenating along the token dimension. However, we don't have the luxury to modify the input embedding size or assume that the backbone is a transformer. Also for a fair comparison with L2P, we concatenate a fixed sized prompt to the input embedding and use a fully connected layer to map the feature dimension back to default input embedding size of (176, 128). Hence, sizes are as follows: initial prompt size (100, 5, 128), Attention (100, 128), Key (100, 128), each have 100 prompt components. After alpha weighting, fixed sized prompt P (5, 128) gets concatenated along joint dimension. We don't update query adaptor in this experiment. We implement all loss functions, including the orthogonality loss as is. Also, note that we attach prompts only at the input embedding layer for fair comparison of the prompting strategy.

Adapting ALICE [31]: For training the base session, we add a projection head to the feature extractor before the classification layer. Like the paper, we use two augmentations of every input and losses from the two augmentations are averaged before backpropagation. We use angular penalty for training the classifier. After base session learning, the projection head and classification layer are discarded, only learning feature extractor. Next, we use cosine distance and class-wise mean to generate class-prototypical feature vectors from the feature extractor's output. These prototypes are used for nearest mean classification. For incremental steps, no training is done. Only new class means are computed and evaluation is performed.

Adapting LUCIR [15] and EWC [17]: We initialize model from previous checkpoint, such that classifier has random weights for new classes and previous classifier weights are copied over to previous class parameters in the classifier. Cross entropy loss is computed between logits from all the classes and current task ground truth labels. All regularization loss terms are implemented as proposed in their respective papers. For LwF [25], we use a $\lambda = 1.0$.

Bibliography

[1] Shubhra Aich, Jesus Ruiz-Santaquiteria, Zhenyu Lu, Prachi Garg, K J Joseph, Alvaro Fernandez, Vineeth N Balasubramanian, Kenrick Kin, Chengde Wan, Nicati Cihan Camgoz, Shugao Ma, and Fernando De la Torre. Data-free class-incremental hand gesture recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 1, C.0.3

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3.2.2

[3] Benjamin Bowman, Alessandro Achille, Luca Zancato, Matthew Trager, Pramuditha Perera, Giovanni Paolini, and Stefano Soatto. a-la-carte prompt tuning (apt): Combining distinct data via composable prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14984–14993, 2023. 2.0.2, 4.0.1, 4.0.2, 4.1, 5.0.2, A.3

[4] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018. A.0.2

[5] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. A.0.2

[6] Yuxiao Chen, Long Zhao, Xi Peng, Jianbo Yuan, and Dimitris N Metaxas. Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2019. (document), 3.2.1, 4.0.1, 4.2, C.0.1

[7] Yuxin Chen, Ziqi Zhang, Chunfeng Yuan, Bing Li, Ying Deng, and Weiming Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13359–13368, 2021. (document), 4.0.1, 4.1, A.3, C.0.1

[8] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford,

and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020. 3.2.4

[9] Songlin Dong, Xiaopeng Hong, Xiaoyu Tao, Xinyuan Chang, Xing Wei, and Yihong Gong. Few-shot class-incremental learning via relation knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1255–1263, 2021. 2.0.3, 4.0.2

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3.2.1

[11] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020. 3.2.1

[12] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=wTTjnvGphYj`. 3.1.1, 3.2.1, 5.0.2

[13] Michael Hersche, Geethan Karunaratne, Giovanni Cherubini, Luca Benini, Abu Sebastian, and Abbas Rahimi. Constrained few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9057–9067, 2022. 2.0.3

[14] Carlos Hinojosa, Miguel Marquez, Henry Arguello, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. Privhar: Recognizing human actions from privacy-preserving lens. In *European Conference on Computer Vision*, pages 314–332. Springer, 2022. 1, 6.2

[15] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839, 2019. 4.0.1, 4.1, A.3, C.0.4

[16] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022. 2.0.1

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 4.0.1, 4.0.2, 4.1, A.3, C.0.4

[18] Sudhakar Kumawat and Hajime Nagahara. Privacy-preserving action recognition via motion difference quantization. In *European Conference on Computer Vision*,

pages 518–534. Springer, 2022. 6.2

[19] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019. 6.2

[20] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 2.0.1, 3.2.1

[21] Ming Li, Xiangyu Xu, Hehe Fan, Pan Zhou, Jun Liu, Jia-Wei Liu, Jiahe Li, Jussi Keppo, Mike Zheng Shou, and Shuicheng Yan. Stprivacy: Spatio-temporal privacy-preserving action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5106–5115, 2023. 6.2

[22] Tianjiao Li, Qiuhong Ke, Hossein Rahmani, Rui En Ho, Henghui Ding, and Jun Liu. Else-net: Elastic semantic network for continual action recognition from skeleton data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13434–13443, 2021. 1, C.0.3

[23] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 2.0.1

[24] Yang Li, Si Si, Gang Li, Cho-Jui Hsieh, and Samy Bengio. Learnable fourier features for multi-dimensional spatial positional encoding. *Advances in Neural Information Processing Systems*, 34:15816–15829, 2021. 5.0.2

[25] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 4.0.1, 4.1, 4.2, A.3, C.0.4

[26] Xuanqing Liu, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. In *International conference on machine learning*, pages 6327–6335. PMLR, 2020. 5.0.2

[27] Ning Ma, Hongyi Zhang, Xuhui Li, Sheng Zhou, Zhen Zhang, Jun Wen, Haifeng Li, Jingjun Gu, and Jiajun Bu. Learning spatial-preserved skeleton representations for few-shot action recognition. In *European Conference on Computer Vision*, pages 174–191. Springer, 2022. 1

[28] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 5.0.2

[29] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021. 3.2.1

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven

Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 5.0.2

[31] Can Peng, Kun Zhao, Tianren Wang, Meng Li, and Brian C Lovell. Few-shot class-incremental learning from an open-set perspective. In *European Conference on Computer Vision*, pages 382–397. Springer, 2022. 2.0.3, 4.0.1, 4.0.2, 4.2, C.0.4

[32] Federico Pernici, Matteo Bruni, Claudio Baecchi, Francesco Turchini, and Alberto Del Bimbo. Class-incremental learning with pre-allocated fixed classifiers. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6259–6266. IEEE, 2021. 2.0.3

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 4.0.1, B.2.1

[34] Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*, 2023. 2.0.2, 5.0.2

[35] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8119–8127, 2018. 5.0.2

[36] Bin Ren, Mengyuan Liu, Runwei Ding, and Hong Liu. A survey on 3d skeleton-based action recognition using learning method. *Cyborg and Bionic Systems*, 2020. 1

[37] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021. 4.0.1, B.2.1

[38] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 5.0.2

[39] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016. (document), 4.0.1, 4.1, A.3

[40] 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 Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association,

2017. ISBN 978-3-03868-030-7. doi: 10.2312/3dor.20171049. (document), 4.0.1, 4.2

[41] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11909–11919, 2023. 2.0.2, 4.0.1, 4.0.2, 4.1, 4.2, 1, 5.0.2, A.3, B.1.1, B.2.1, C.0.4

[42] Yu-Ming Tang, Yi-Xing Peng, and Wei-Shi Zheng. When prompt-based incremental learning does not meet strong pretraining. *arXiv preprint arXiv:2308.10445*, 2023. 2.0.2

[43] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12183–12192, 2020. 2.0.3, 4.0.2

[44] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 3.2.2, 3.2.2

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3.2.1

[46] Andrés Villa, Juan León Alcázar, Motasem Alfarra, Kumail Alhamoud, Julio Hurtado, Fabian Caba Heilbron, Alvaro Soto, and Bernard Ghanem. Pivot: Prompting for video continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24214–24223, 2023. 4.0.1, 2, B.2.1

[47] Xiang Wang, Shiwei Zhang, Zhiwu Qing, Changxin Gao, Yingya Zhang, Deli Zhao, and Nong Sang. Molo: Motion-augmented long-short contrastive learning for few-shot action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18011–18021, 2023. 1

[48] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *Advances in Neural Information Processing Systems*, 35:5682–5695, 2022. 2.0.2, 4.0.1, B.2.1

[49] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer, 2022. 2.0.2, 3.2.2, 4.0.1, 5.0.2, B.2.1

[50] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 2.0.1, 2.0.2, 3.2.2, 3.2.2, 4.0.1, 4.0.2, 4.1, 4.2, 4.0.3, 5.0.2, A.3, B.1.1, B.2.1

[51] Will Williams, Sam Ringer, Tom Ash, David MacLeod, Jamie Dougherty, and John Hughes. Hierarchical quantized autoencoders. *Advances in Neural Information Processing Systems*, 33:4524–4535, 2020. 3.2.4

[52] Yibo Yang, Haobo Yuan, Xiangtai Li, Zhouchen Lin, Philip Torr, and Dacheng Tao. Neural collapse inspired feature-classifier alignment for few-shot class-incremental learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=y5W8tpojhtJ`. 2.0.3

[53] Rujing Yue, Zhiqiang Tian, and Shaoyi Du. Action recognition based on rgb and skeleton data sets: A survey. *Neurocomputing*, 2022. 1

[54] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. A comprehensive survey of vision-based human action recognition methods. *Sensors*, 19(5):1005, 2019. 1

[55] Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22798–22807, 2023. 3.2.4

[56] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9046–9056, 2022. 2.0.3

[57] Anqi Zhu, Qiuhong Ke, Mingming Gong, and James Bailey. Adaptive local-component-aware graph convolutional network for one-shot skeleton-based action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6038–6047, 2023. 1

[58] Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15659–15669, 2023. 2.0.1