Lightweight Learner for Shared Knowledge Lifelong Learning

Yunhao Ge [1] **Yuecheng Li** [*1] **Di Wu** [*1] **Ao Xu** [*1] **Adam M. Jones** [2] **Amanda Sofie Rios** [3]
Iordanis Fostiropoulos [1] **Shixian Wen** [4] **Po-Hsuan Huang** [2] **Zachary William Murdock** [2] **Gozde Sahin** [1]
Shuo Ni [1] **Kiran Lekkala** [1] **Sumedh Anand Sontakke** [1] **Laurent Itti** [1 2 5]

Abstract

In Lifelong Learning (LL), agents continually learn as they encounter new conditions and tasks. Most current LL is limited to a single agent that learns tasks sequentially. Dedicated LL machinery is then deployed to mitigate the forgetting of old tasks as new tasks are learned. This is inherently slow. We propose a new Shared Knowledge Lifelong Learning (SKILL) challenge, which deploys a decentralized population of LL agents that each sequentially learn different tasks, with all agents operating independently and in parallel. After learning their respective tasks, agents share and consolidate their knowledge over a decentralized communication network, so that, in the end, all agents can master all tasks. We present one solution to SKILL which uses Lightweight Lifelong Learning (LLL) agents, where the goal is to facilitate efficient sharing by minimizing the fraction of the agent that is specialized for any given task. Each LLL agent thus consists of a common task-agnostic immutable part, where most parameters are, and individual task-specific modules that contain fewer parameters but are adapted to each task. Agents share their task-specific modules, plus summary information ("task anchors") representing their tasks in the common task-agnostic latent space of all agents. Receiving agents register each received task-specific module using the corresponding anchor. Thus, every agent improves its ability to solve new tasks each time new task-specific modules and anchors are received. If all agents can communicate with all others, eventually all agents become identical and can solve all tasks. On a new, very challenging SKILL-102 dataset with 102 image classification tasks (5,033 classes in total, 2,041,225 training, 243,464 validation, and 243,464 test images), we achieve much higher (and SOTA) accuracy over 8 LL baselines, while also achieving near perfect parallelization. Code and data can be found at https://github.com/gyhandy/Shared-Knowledge-Lifelong-Learning

[*]Equal contribution [1]Thomas Lord Department of Computer Science, University of Southern California, Los Angeles, California, US [2]Neuroscience Graduate Program, University of Southern California, Los Angeles, California, US [3]Intel Labs, Santa Clara, California, United States [4]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangzhou, China [5]Dornsife Department of Psychology, University of Southern California, Los Angeles, California, US. Correspondence to: Yunhao Ge <yunhaoge@usc.edu>, Yuecheng Li <liyueche@usc.edu>, Di Wu <dwu92983@usc.edu>, Ao Xu <aoxu@usc.edu>, Laurent Itti <itti@usc.edu>.

In *ICML Workshop on Localized Learning (LLW)*, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

1. Introduction

Lifelong Learning (LL) is a relatively new area of machine learning (ML) research, in which agents continually learn as they encounter new tasks, acquiring novel task knowledge while avoiding forgetting of previous tasks (Parisi et al., 2019). This differs from standard train-then-deploy ML, which cannot incrementally learn without catastrophic interference across successive tasks (French, 1999).

Most current LL research assumes a single agent that sequentially learns from its own actions and surroundings, which, by design, is not parallelizable over time and/or physical locations. In the real world, tasks may happen in different places; for instance, we may need agents that can operate in deserts, forests, and snow, as well as recognize birds in the sky and fish in the deep ocean. The possibility of parallel task learning and sharing among multiple agents to speed up lifelong learning has traditionally been overlooked. To solve the above challenges, we propose a new Lifelong Learning challenge scenario, Shared Knowledge Lifelong Learning (SKILL): A population of originally identical LL agents is deployed to a number of distinct physical locations. Each agent learns a sequence of tasks in its location. Agents share knowledge over a decentralized network, so that, in the end, all agents can master all tasks. SKILL promises the following benefits: speedup

of learning through parallelization; ability to simultaneously learn from distinct locations; resilience to failures as no central server is used; possible synergies among agents, whereby what is learned by one agent may facilitate future learning by other agents.

However, to solve SKILL, one must address the following challenges:

Chal-1 **Distributed, decentralized learning of multiple tasks.** A solution to SKILL should support a population of agents deployed over several physical locations and each learning one or more sequential tasks. For resilience reasons, the population should not rely on a single central server.

Chal-2 **Lifelong learning ability**: Each agent must be capable of lifelong learning, i.e., learning a sequence of tasks with minimal interference and no access to previous data as each new task is learned.

Chal-3 **Shareable knowledge representation:** The knowledge representation should easily be shared and understood among agents. Agents must be able to consolidate knowledge from other agents in a decentralized, distributed fashion.

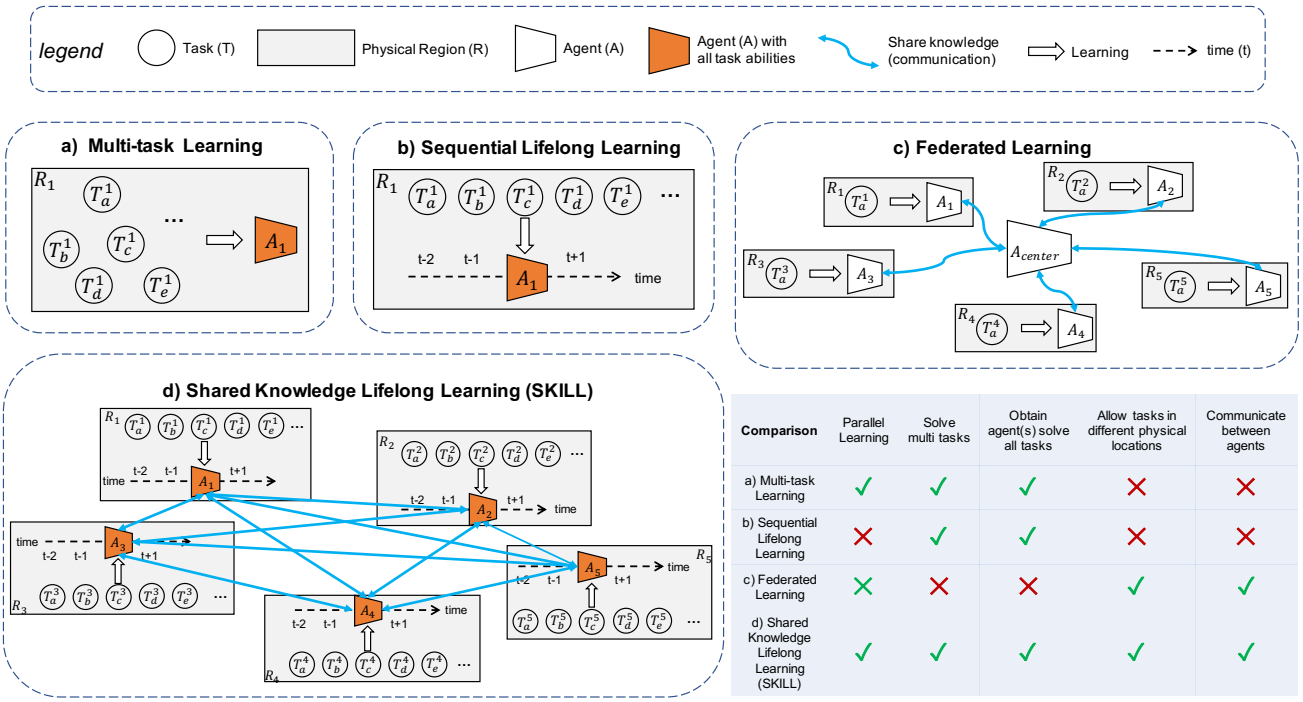
Chal-4 **Speedup through parallelization:** Shared knowledge should be sufficiently compact, so that the benefits from using multiple parallel agents are not erased by communications costs. Adding more agents should result in greater speedup compared to a single agent. We measure speedup as the the ratio of time it takes for one agent to learn all tasks compared to N agents (larger is better). As a goal for our work, we strive for a speedup of at least $0.5 \times N$ with N agents, where perfect speedup would be $1.0 \times N$ if there was no parallelization and communications overhead.

Chal-5 **Ability to harness possible synergies among tasks:** When possible, learning some tasks may improve learning speed or performance at other, related tasks.

To address the SKILL challenge, we take inspiration from neuroscience. Many approaches to LL involve at least partially retraining the core network that performs tasks (feature extraction backbone plus classification head), as every new task is learned. But transmitting and then merging these networks across multiple agents would incur very high communications and computation costs. With the exception of perceptual learning, where human visual cortex may indeed be altered when learning specific visual discrimination tasks for days or weeks (Goldstone, 1998; Dosher & Lu, 2017), there is little evidence that our entire visual cortex — from early stage filters in primary visual cortex to complex features in inferotemporal cortex — is significantly altered when we just learn, e.g., about a new class of flowers from a few exemplars. Instead, the perirhinal cortex (and more generally the medial temporal lobe) may be learning new representations for new objects by drawing upon and combining existing visual features and representations from visual cortex (Deshmukh et al., 2012). This may give rise to specialized "grandmother cells" (Bowers, 2017) (or Jennifer Aniston neurons; (Quiroga et al., 2005; Quiroga, 2017)) that can be trained on top of an otherwise rather immutable visual cortex backbone. While the grandmother cell hypothesis remains debated in neuroscience (vs. distributed representations; (Valdez et al., 2015)), here, it motivates us to explore the viability of a new lightweight lifelong learning scheme, where the feature extraction backbone and the latent representation are fixed, and each new object class learned is represented by a single new neuron that draws from this representation.

From this inspiration, we propose a simple but effective solution to SKILL based on new lightweight lifelong learning (LLL) agents. Each LLL agent uses a common frozen backbone built-in at *initialization*, so that only the last layer *(head)* plus some small adjustments to the backbone *(beneficial biases)* are learned for each task. To eliminate the need for a task oracle, LLL agents also learn and share summary statistics about their training datasets, or share a few training images, to help other agents assign test samples to the correct head (task mapper). On a new, very challenging dataset with 102 image classification tasks (5,033 classes in total, 2,041,225 training, 243,464 validation, and 243,464 test images), we achieve higher accuracy compared to 8 LL baselines, and also near-perfect parallelization speedup.

Our main contributions are: (1) We formulate a new Lifelong learning challenge, Shared Knowledge Lifelong Learning (SKILL), which focuses on parallel (sped up) task learning and knowledge sharing among agents. We frame SKILL and contrast it with multi-task learning, sequential LL, and federated learning (Sec. A). (2) A new LL benchmark dataset: SKILL-102, with 102 complex image classification tasks. To the best of our knowledge, it is the most challenging benchmark to evaluate LL and SKILL algorithms in the image classification domain, with the largest number of tasks, classes, and inter-task variance (Sec. 3). (3) A solution to the SKILL problem: Lightweight Lifelong Learning (LLL) for efficient knowledge sharing among agents, using a fixed shared core plus task-specific shareable modules. The need for a task oracle is eliminated by using a task mapper, which can automatically determine the task at inference time from just an input image (Sec. 4). (4) Our SKILL algorithm achieves SOTA performance on three main metrics: High LL task accuracy (less catastrophic forgetting), low shared (communication) resources, and high speedup ratio (Sec. 5). (5) The proposed Lightweight Lifelong Learner shows promising forward knowledge transfer, which reuses the accumulated knowledge for faster and more accurate learning of new tasks.



Comparison	Parallel Learning	Solve multi tasks	Obtain agent(s) solve all tasks	Allow tasks in different physical locations	Communicate between agents
a) Multi-task Learning	✓	✓	✓	✗	✗
b) Sequential Lifelong Learning	✗	✓	✓	✗	✗
c) Federated Learning	✗	✗	✗	✓	✓
d) Shared Knowledge Lifelong Learning (SKILL)	✓	✓	✓	✓	✓

Figure 1. SKILL vs. related learning paradigms. a) Multi-task learning (Caruana, 1997): one agent learns all tasks at the same time in the same physical location. b) Sequential Lifelong Learning (S-LL) (Li & Hoiem, 2017): one agent learns all tasks sequentially in one location, deploying LL-specific machinery to avoid task interference. c) Federated learning (McMahan et al., 2017): multiple agents learn *the same task* in different physical locations, then sharing learned knowledge (parameters) with a center agent. d) Our SKILL: different S-LL agents in different physical regions each learn tasks, and learned knowledge is shared among all agents, such that finally all agents can solve all tasks. Bottom-right table: Strengths & weaknesses of each approach.

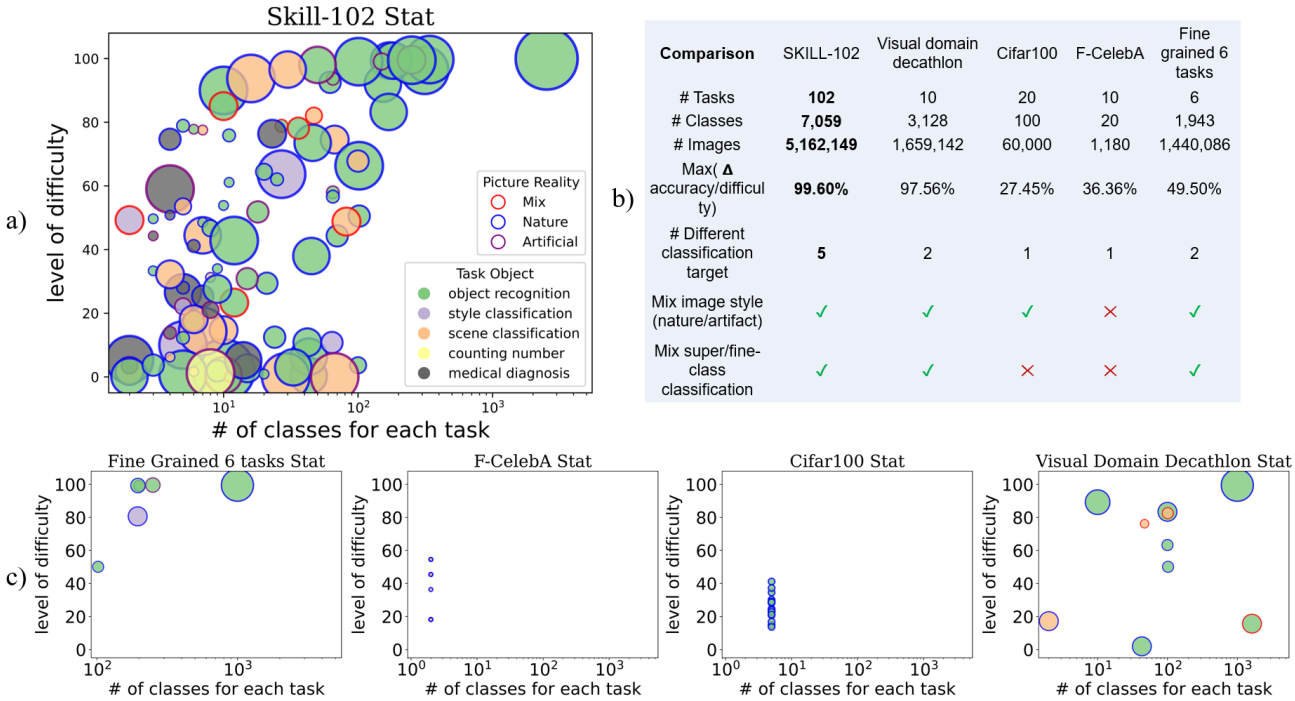
2. Related Works

Lifelong Learning (LL) focuses on continual knowledge acquisition and retention across numerous tasks, constrained by factors such as unavailability of past training data (Masana et al., 2022; Parisi et al., 2019). It employs methods like regularization, parameter isolation, and rehearsal. However, conventional LL methods often struggle with parallel learning and knowledge sharing requirements (De Lange et al., 2021; Kirkpatrick et al., 2017; Aljundi et al., 2018; Zenke et al., 2017). Multi-Task Learning (MTL) leverages information across multiple related tasks to improve performance (Zhang & Yang, 2021; Crawshaw, 2020; Ruder, 2017). Unlike the Shared Knowledge and Intelligence in Learning Locally (SKILL) approach, where tasks are spatially separate, MTL presupposes co-location of tasks. Federated Learning (FL) enables decentralized data to collaboratively train a model across clients (Kairouz et al., 2021; Li et al., 2020; Bonawitz et al., 2019). Unlike SKILL, FL often deals with agents learning the same task using a central server, and does not typically address sequences of tasks or the LL problem. Furthermore, FL's reliance on a central server makes it susceptible to complete failure (Yoon et al., 2021). A more detailed list of related works is in B

3. SKILL-102 dataset

We use image classification as the basic task framework and propose a novel LL benchmark dataset: SKILL-102 (Fig. 2). SKILL-102 consists of 102 image classification datasets. Each one supports one complex classification task, and the corresponding dataset was obtained from previously published sources (e.g., task 1: classify flowers into 102 classes, such as lily, rose, petunia, etc using 8,185 train/val/test images (Nilsback & Zisserman, 2008a); task 2: classify 67 types of scenes, such as kitchen, bedroom, gas station, library, etc using 15,524 images (Quattoni & Torralba, 2009); full dataset sequence and details in Suppl. Fig. S5.

In total, SKILL-102 is a subset of all datasets/tasks and images in DCT, and comprises 102 tasks, 5,033 classes and 2,041,225 training images (Suppl. Sec. A and Suppl. Fig. S5). After training, the algorithm is presented 243,464 test images and decides, for each image, which of the 5,033 classes it belongs to (no task oracle). To the best of our knowledge, SKILL-102 is the most challenging com-



Comparison	SKILL-102	Visual domain decathlon	Cifar100	F-CelebA	Fine grained 6 tasks
# Tasks	**102**	10	20	10	6
# Classes	**7,059**	3,128	100	20	1,943
# Images	**5,162,149**	1,659,142	60,000	1,180	1,440,086
Max(Δ accuracy/difficulty)	**99.60%**	97.56%	27.45%	36.36%	49.50%
# Different classification target	**5**	2	1	1	2
Mix image style (nature/artifact)	✓	✓	✓	✗	✓
Mix super/fine-class classification	✓	✓	✗	✗	✓

Figure 2. (a) SKILL-102 dataset visualization. Task difficulty (y-axis) was estimated as the error rate of a ResNet-18 trained from scratch on each task for a fixed number of epochs. Circle size reflects dataset size (number of images). (b) Comparison with other benchmark datasets including Visual Domain Decathlon (Rebuffi et al., 2017a), Cifar-100 (Krizhevsky et al., 2009), F-CelebA (Ke et al., 2020), Fine-grained 6 tasks (Russakovsky et al., 2014) (Wah et al., 2011), (Nilsback & Zisserman, 2008b), (Krause et al., 2013), (Saleh & Elgammal, 2015), (Eitz et al., 2012) c) Qualitative visualization of other datasets, using the same legend and format as in a).

pletely real (not synthesized or permuted) image classification benchmark for LL and SKILL algorithms, with the largest number of tasks, number of classes, and inter-task variance.

4. Lightweight Lifelong Learner for SKILL

To satisfy the requirements of SKILL (see Introduction), we design Lightweight Lifelong Learning (LLL) agents. The design motivation is as follows: We propose to decompose agents into a generic, pretrained, common representation backbone endowed into all agents at manufacturing time, and small task-specific decision modules. This enables distributed, decentralized learning as agents can learn their own tasks independently (Chal-1). It also enables lifelong learning (Chal-2) in each agent by creating a new task-specific module for each new task. Because the shared modules are all operating in the common representation of the backbone, this approach also satisfies (Chal-3). Using compact task-specific modules also aims to maximize speedup through parallelization (Chal-4). Finally, we show a few examples where knowledge from previously learned tasks may both accelerate the learning and improve the performance on new tasks (Chal-5).

Fig. 3 shows the overall pipeline and 4 roles for each agent. Agents use a common frozen backbone and only a compact task-dependent "head" module is trained per agent and task, and then shared among agents. This makes the cost of both training and sharing very low. Head modules simply consist of (1) a classification layer that operates on top of the frozen backbone ((Appendix Sec. **??**)), and (2) a set of beneficial biases (BB) (Appendix Sec. **??**) that provide lightweight task-specific re-tuning of the backbone, to address potentially large domain gaps between the task-agnostic backbone and the data distribution of each new task. To eliminate the need for a task oracle, LLL agents also learn and share task anchors, in the form of summary statistics about their training datasets, or share a few training images, to help other agents assign test samples to the correct head at test time (task mapper). Two representations for task anchors, and the corresponding task mapping mechanisms, are explored: Gaussian Mixture Model Classifier (GMMC)(Rios & Itti, 2020) (Appendix Sec. **??**) and Mahalanobis distance classifier (MAHA)(Lee et al., 2018) (Appendix Sec. **??**). Receiving agents simply accumulate received heads and task anchors in banks, and the anchors for all tasks received so far by an agent are combined to form a task mapper within that agent. We currently assume a fully connected communication network among all agents, and every agent, after learning a new task, broadcasts its head and task anchor to all other agents. Hence, all agents become identical after all tasks have been learned and shared, and they all can master all tasks. At test time,

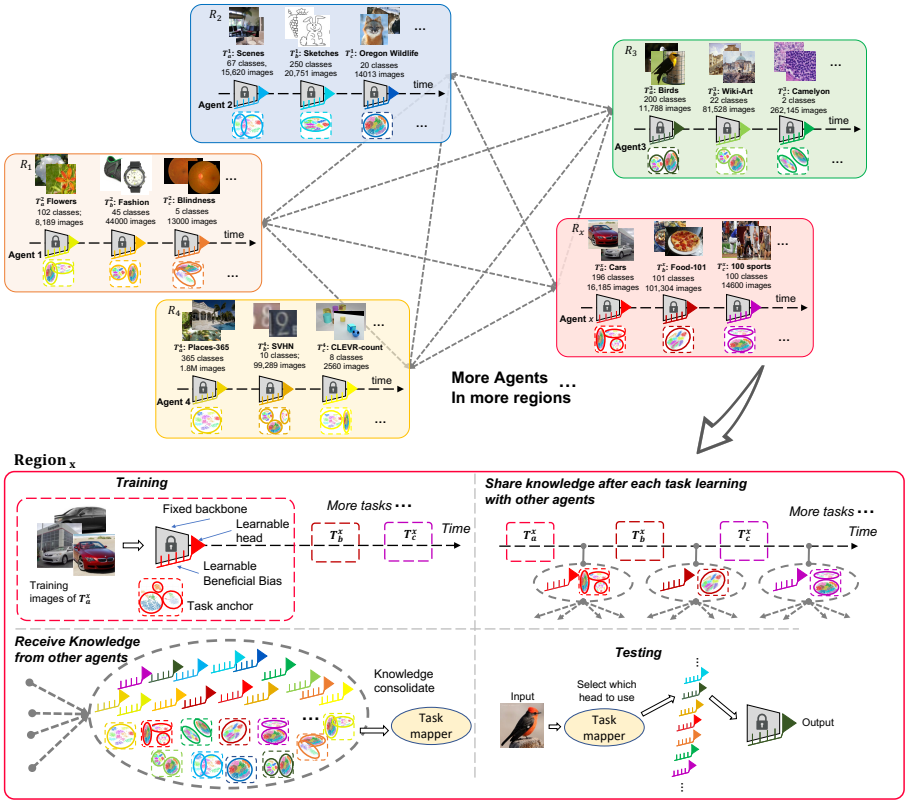


Figure 3. Algorithm design. Top: overall pipeline, where agents are deployed in different regions to learn their own tasks. Subsequently, learned knowledge is shared among all agents. Bottom: Zoom into the details of each agent, with 4 main roles (Appendix Sec. Q)

using one of all identical agents, we first run input data through the task mapper to recover the task, and then invoke the corresponding head to obtain the final system output. The task mapper eliminates the need for a task oracle at test time. The combination of using a pre-trained backbone, task-specific head and BB, and task mapper enables lifelong learning in every agent with minimal forgetting as each agent learns a sequence of tasks.

Pretrained backbone: We use the xception (Chollet, 2017) pretrained on ImageNet (Deng et al., 2009), as it provides a good balance between model complexity and expressivity of the embedding.

Beneficial Biases: To address potentially large domain shifts between ImageNet and future tasks (e.g., line-drawing datasets, medical imaging datasets, astronomy datasets), we designed beneficial biases (BB). Inspired by the Beneficial Perturbation Network (BPN) of (Wen et al., 2021), BB provides a set of task-dependent, out-of-network bias units which are activated per task. These units take no input. Their constant outputs add to the biases of the neurons already present in the backbone network; thus, they provide one bias value per neuron in the core network. This is quite lightweight, as there are far fewer neurons than weights in the backbone (22.9M parameters but only 22k neurons in xception). Different from BPN, which works best in conjunction with an LL method like EWC (Kirkpatrick et al., 2017) or PSP (Cheung et al., 2019), and only works on fully-connected layers, BB does not require EWC or PSP, and can perform as an add-on module on both convolutional layers (Conv) and fully-connected layers (FC). Specifically, for each Conv layer, we have

$$y = Conv(x) + b + B \tag{1}$$

with input feature $x \in \mathbb{R}^{w*h*c}$, output feature $y \in \mathbb{R}^{w'*h'*c'}$. $b \in \mathbb{R}^{c'}$ is the original frozen bias of the backbone, and $B \in \mathbb{R}^{c'}$ is our learnable beneficial bias. The size of B is equal to the number of kernels (c') in this Conv layer. (w, h, c and w', h', c' denote the width, height and channels of the input and output feature maps respectively.) For FC layers,

$$y = FC(x) + b + B \tag{2}$$

with $x \in \mathbb{R}^{l}$, $y \in \mathbb{R}^{l'}$, $b \in \mathbb{R}^{l'}$ and $B \in \mathbb{R}^{l'}$. The size of B (beneficial bias) is equal to the number of hidden units (l') in this FC layer.

GMMC task mapper: To recover task at test time, each agent also learns Gaussian Mixture clusters (GMMC) (Rios

& Itti, 2020) that best encompass each of its tasks' data, and shares the cluster parameters (means + diagonal covariances). This is also very fast to learn and very compact to share. As shown in Fig. 3(bottom right), during training, each agent clusters its entire training set into k Gaussian clusters:

$$f(x) = \sum_{i=1}^{k} \phi_i \mathcal{N}(x|\mu_i, \Sigma_i), \qquad \sum_{i=1}^{k} \phi_i = 1 \tag{3}$$

We use $k = 25$ clusters for every task (ablation studies in Appendix). In sharing knowledge, each agent performs a "teacher" role on its learned task and shares the mean and diagonal covariance of its clusters with all other agents (students). In receiving knowledge, each agent performs a "student" role and just aggregates all received clusters in a bank to form a task mapper with kT clusters, keeping track of which task any given cluster comes from: $D_{map}() = \{(\mathcal{N}_1, \phi_1) : 1, ..., (\mathcal{N}_{kT}, \phi_{kT}) : T\}$. At test time, a image x_i is evaluated against all clusters received so far, and the task associated with the cluster closest to the test image is chosen: $Task = D_{map}((\mathcal{N}_m, \phi_m))$, where $m = \arg\max_m(P(m, x_i))$. The probability of image x_i belonging to the mth Gaussian cluster is given by:

$$P(m, x_i) = \frac{\phi_m \mathcal{N}(x|\mu_m, \Sigma_m)}{\sum_{n=1}^{kT} \phi_n \mathcal{N}(x|\mu_n, \Sigma_n)} \tag{4}$$

Mahalanobis task mapper: To perform as a task mapper, the Mahalanobis distance (MAHA) method (Lee et al., 2018) learns C class-conditional Gaussian distributions $\mathcal{N}(x|\mu_c, \hat{\Sigma})$, c = 1,2, ... C, where C is the total number of classes of all T tasks and $\hat{\Sigma}$ is a tied covariance computed from samples from all classes. The class mean vectors and covariance matrix of MAHA are estimated as: $\mu_c = \frac{1}{N_c} \sum_{i:y_i=c} x_i$ (N_c: number of images in each class) and $\hat{\Sigma} = \frac{1}{N} \sum_{c=1}^{C} \sum_{i:y_i=c} (x_i - \mu_c)(x_i - \mu_c)^T$, ($N$: total number of images shared to the student agent). In training, each teacher agent computes the mean of each class within its task and randomly samples a variable number m of images per class. In our experiments, we use $m = 5$ images/class for every task. During sharing knowledge, each agent shares the sample class means along with the saved images with all other agents. The shared images received by the student agents are used to compute the tied covariance. Similar to GMMC, the student agents also maintain a task mapper to keep track of which task any given class comes from. For a test image x, MAHA computes the Mahalanobis distance for all classes received so far and assigns the test image to the task associated with the smallest Mahalanobis distance, defined as:

$$\arg\min_c (x - \mu_c)^T \hat{\Sigma}^{-1} (x - \mu_c) \tag{5}$$

System implementation details: (1) Frozen xception backbone (Chollet, 2017), with 2048D latent representation. (2) Each agent learns one "head" per task, which consists of one fully-connected layer with 2048 inputs from the backbone and c outputs for a classification task with c classes (e.g., task 1 is to classify $c = 102$ types of flowers), and BB biases that allow us to fine-tune the backbone without changing its weights, to mitigate large domain shifts. (3) Each agent also fits $k = 25$ Gaussian clusters in the 2048D latent space to its training data. (4) At test time, a test image is presented and processed forward through the xception backbone. The GMMC classifier then determines the task from the nearest Gaussian cluster. The corresponding head is loaded and it produces the final classification result: which image class (among 5,033 total) the image belongs to. (5) The workflow is slightly different with the Mahalanobis task mapper: while GMMC clusters are learned separately at each teacher for each task as the task is learned, the Mahalanobis classifier is trained by students after sharing, using 5 images/class shared among agents. (6) Agents are implemented in pyTorch and run on desktop-grade GPUs (e.g., nVidia 3090, nVidia 1080).

5. Experiments and results

We first compare how well LLL can learn multiple tasks sequentially in a single agent, compared to baselines LL algorithms (Appendix Sec. L.1). We plot the absolute accuracy averaged over all tasks learned so far in Fig. 4. Experiments show our method achieve such a great performance with even lower computational cost compared to other baselines (Appendix Sec. L.5). More experiments on the first task performance (Fig. S11) and the first 10 tasks (Sec. L.4) are shown in Appendix. S11.

Table. 1 shows the computation and networking expenditures for our approach and our modified SUPSUP to learn all tasks in the SKILL-102 dataset in a distributed manner. We measure everything in terms of MACs (multiply-accumulate operations, which are implemented as one atomic instruction on most hardware). The amount of data shared per task for our approach is quite small (see Suppl. Fig. S5). Our results in Table. 1 show: (1) **Our approach has very low parallelization overhead**, which leads to almost perfect speedup $> 0.99N$ for all variants. (Suppl. Sec. G). Students either do nothing (just accumulate received knowledge in a bank) or update their Mahalanobis task mapper. (2) **The baselines have comparatively much higher training cost, yet their performance is poor.** More details see Suppl. Sec. L.5

As our system learns many tasks, it may occur that some tasks overlap with others, i.e., they may share similar images and/or class labels. Here, we explore two approaches to handle such overlap: a corrective method K.1 and a

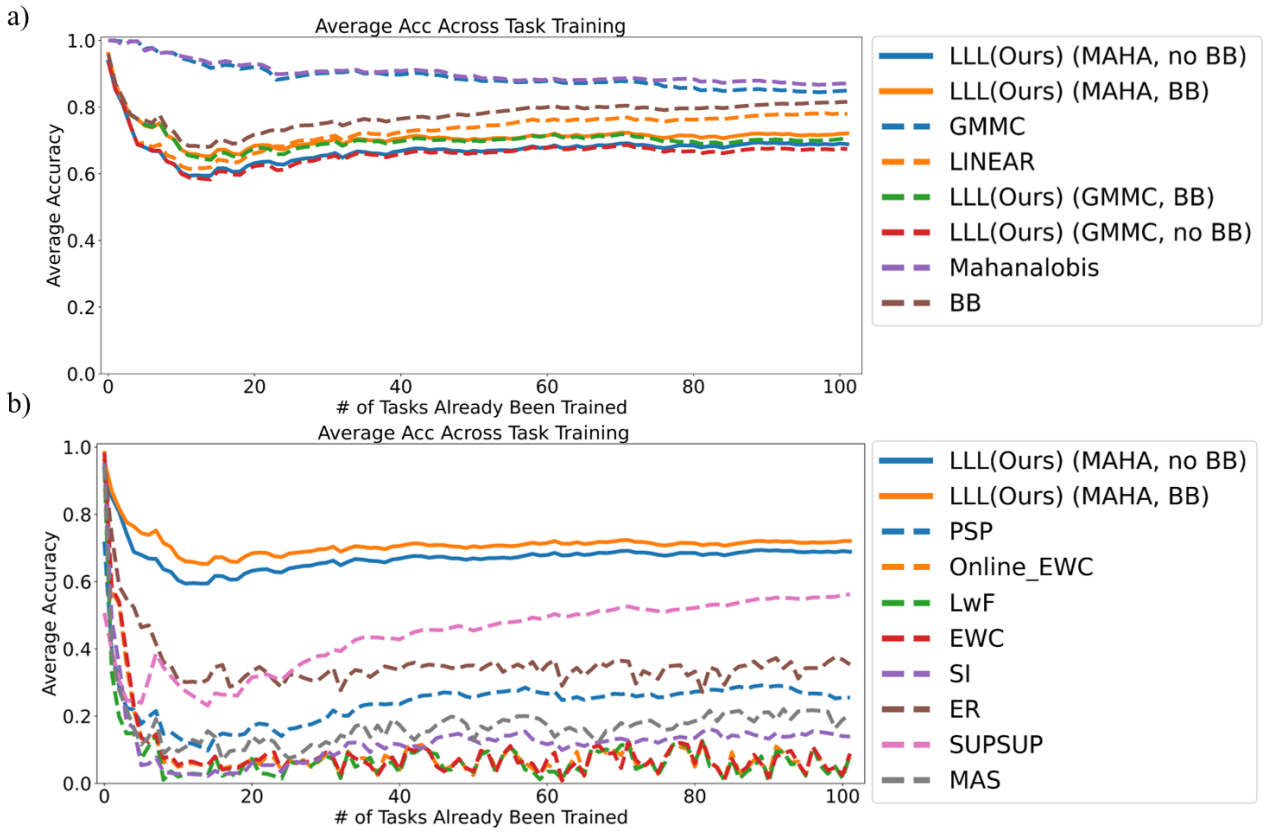


Figure 4. Average absolute accuracy on all tasks learned so far, as a function of the number of tasks learned. Our LLL approach is able to maintain higher average accuracy than all baselines. BB provides a small but reliable performance boost (LLL w/BB vs. LLL w/o BB). The sharp decrease in early tasks carries no special meaning except for the fact that tasks 4,8,10 are significantly harder than the other tasks in the 0-10 range, given the particular numbering of tasks in SKILL-102. Note how again SUPSUP has a low accuracy for the very first task. This is because of the nature of its design; indeed, SUPSUP is able to learn some other tasks in our sequence with high accuracy (Suppl. Fig. S5).

learning-based method K.2. The corrective approach identifies semantic similarity between class names to alleviate the issue of task overlap, providing a consistent improvement in accuracy. The learning-based approach leverages previously learned weights from similar classes to initialize the learning of new classes, promoting faster convergence and improved performance.

We proposed a specific strategy to handle the overlapping of tasks that are likely to occur with a large number of learned tasks K.2. We also introduced an approach that reuses learned knowledge from old tasks to expedite the learning speed and increase the accuracy of new tasks. Experimental results validated our strategies, showing faster learning times and enhanced accuracy

Please note, this is a concise overview and for a more in-depth understanding, kindly refer to the appendix for omitted details.

5.1. Discussion and Future Work

We have presented a novel lightweight approach to lifelong learning, demonstrating superior performance and almost perfect benefit from parallelization on the SKILL-102 benchmark. SKILL-102, possibly the largest non-synthesized lifelong learning challenge dataset, contains real data in each task with large inter-task variance, making it a more realistic test scenario (Cheung et al., 2019).

Our approach utilizes a lightweight task-specific architecture and rapid adaptation through a compact Beneficial Biases (BB) module, maximizing leverage obtained from task-agnostic knowledge. This design shows improved performance in large scale lifelong learning tasks and addresses our SKILL challenge effectively.

The success of our approach can be attributed to the lightweight learner design: a *fixed backbone* representing shared task-agnostic knowledge, *Beneficial Biases* that shift the backbone to solve domain gaps with compact pa-

Table 1. Analysis of computation expenditures and accuracy for our approach and the baselines, to learn all 102 tasks (with a total of 5,033 classes, 2,041,225 training images) in a single agent. Here we select LLL, no BB, MAHA as reference (1x CPU usage) since it is the fastest approach, yet still has higher accuracy than all baselines. For our approach, MAHA leads to slightly higher accuracy than GMMC, at roughly the same computation cost. All baselines perform worse that our approach, even though they also requires more computation than our approaches that do not use BB. BB adds significatly to our computation cost, but also leads to the best accuracy when used with MAHA.

	Training (MACs)	**CPU usage VS. Ours, no BB, MAHA**	**Average accuracy after learning 102 tasks**
LLL(Ours)-Single Agent, no BB, GMMC	1.73E+16	~1x	67.43%
LLL(Ours)-Single Agent, BB, GMMC	1.56E+18	~90.7x	70.58%
LLL(Ours)-Single Agent, no BB, Mahalanobis	**1.73E+16**	**1x (reference)**	**68.87%**
LLL(Ours)-Single Agent, BB, Mahalanobis	1.56E+18	~90.7x	72.1%
EWC	1.75E+18	~101.3x	8.86%
PSP	6.28E+17	~36.4x	25.49%
ER	4.53E+18	~262.8x	35.32%
SUPSUP	1.01E+18	~58.6x	56.22%
EWC-ONLINE	1.55E+18	~90.1x	7.77%
LwF	1.56E+18	~90.5x	8.41%
SI	2.07E+18	~120.1x	13.89%
MAS	2.06E+18	~119.6x	20.54%

rameters, and a *GMMC/MAHA* global task anchor representing tasks in a common task-agnostic latent space, eliminating the need for a task oracle at test time.

For future work, we propose testing the use of BB on partial layers to increase speed. We also aim to utilize visual semantic information for class matching, potentially through the use of the GMMC mapper for new class recognition. Further exploration of task mapping techniques beyond GMMC and MAHA may also be valuable.

6. Conclusions

We have proposed a new framework for shared-knowledge, parallelized LL. On a new, very challenging SKILL-102 dataset, we find that this approach works much better than previously SOTA baselines, and is much faster. Scaling to > 500 difficult tasks like the ones in our new SKILL-102 dataset seems achievable with the current implementation.

Broader impacts statement: We believe that LLL will spur a new generation of distributed LL systems, as it makes LL more accessible to edge systems and more parallelizable. Thus, broader impacts are expected to be positive in enabling more lightweight devices to learn at the edge and to share what they have learned.

Acknowledgement: This work was supported by DARPA (HR00112190134), C-BRIC (one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA), and the Army Research Office (W911NF2020053). The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Benjamin, A. S., Rolnick, D., and Kording, K. Measuring and regularizing networks in function space. *arXiv preprint arXiv:1805.08289*, 2018.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

Bowers, J. S. Grandmother cells and localist representations: a review of current thinking, 2017.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Cheung, B., Terekhov, A., Chen, Y., Agrawal, P., and Olshausen, B. Superposition of many models into one. *arXiv preprint arXiv:1902.05522*, 2019.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Crawshaw, M. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Deshmukh, S. S., Johnson, J. L., and Knierim, J. J. Perirhinal cortex represents nonspatial, but not spatial, information in rats foraging in the presence of objects: comparison with lateral entorhinal cortex. *Hippocampus*, 22(10): 2045–2058, 2012.

Dosher, B. and Lu, Z.-L. Visual perceptual learning and models. *Annual review of vision science*, 3:343–363, 2017.

Eitz, M., Hays, J., and Alexa, M. How do humans sketch objects? *ACM Transactions on graphics (TOG)*, 31(4): 1–10, 2012.

Elsayed, G. F., Goodfellow, I., and Sohl-Dickstein, J. Adversarial reprogramming of neural networks. *arXiv preprint arXiv:1806.11146*, 2018.

Evci, U., Dumoulin, V., Larochelle, H., and Mozer, M. C. Head2toe: Utilizing intermediate representations for better transfer learning. *arXiv preprint arXiv:2201.03529*, 2022.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Goldstone, R. L. Perceptual learning. *Annual review of psychology*, 49(1):585–612, 1998.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Ke, Z., Liu, B., and Huang, X. Continual learning of a mixed sequence of similar and dissimilar tasks. *Advances in Neural Information Processing Systems*, 33: 18493–18504, 2020.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, 2009.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., and van de Weijer, J. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008a.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008b.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Parisi, S., Rajeswaran, A., Purushwalkam, S., and Gupta, A. The unsurprising effectiveness of pre-trained vision models for control. *arXiv preprint arXiv:2203.03580*, 2022.

Quattoni, A. and Torralba, A. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420, 2009. doi: 10.1109/CVPR.2009.5206537.

Quiroga, R. Q. *The forgetting machine: Memory, perception, and the Jennifer Aniston neuron*. BenBella Books, 2017.

Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C., and Fried, I. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters. volume 30, 2017a.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017b.

Rios, A. and Itti, L. Closed-loop memory gan for continual learning. *arXiv preprint arXiv:1811.01146*, 2018.

Rios, A. and Itti, L. Lifelong learning without a task oracle. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 255–263. IEEE, 2020.

Robins, A. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge (2014). *arXiv preprint arXiv:1409.0575*, 2(3), 2014.

Saleh, B. and Elgammal, A. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505.00855*, 2015.

Valdez, A. B., Papesh, M. H., Treiman, D. M., Smith, K. A., Goldinger, S. D., and Steinmetz, P. N. Distributed representation of visual objects by single neurons in the human brain. *Journal of Neuroscience*, 35(13):5180–5186, 2015.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The caltech-ucsd birds-200-2011 dataset. 2011.

Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Wen, S., Rios, A., Ge, Y., and Itti, L. Beneficial perturbation network for designing general adaptive artificial intelligence systems. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

Yoon, J., Jeong, W., Lee, G., Yang, E., and Hwang, S. J. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pp. 12073–12086. PMLR, 2021.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995. PMLR, 2017.

Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., and Kuo, C.-C. J. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1131–1140, 2020.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

supplementary

A. Shared knowledge in lifelong learning (SKILL)

The chief motivation for SKILL is to enable the next generation of highly-efficient, parallelizable, and resilient lifelong learning.

Assumptions: (1) A population of N agents wants to learn a total of T different tasks separated into N physical regions. (2) Each agent i asynchronously learns $1 \leq T_i \leq T$ tasks, in sequence, from the distinct inputs and operating conditions it encounters. As in standard LL, training data from previous tasks is not available anymore while learning the next task. (3) Each agent performs as a *"teacher"* for its T_i tasks, by sharing what it has learned with the other $N-1$ agents; at the same time, each agent also performs as a *"student"* by receiving knowledge from the other $N-1$ agents. In the end, every agent has the knowledge to solve all T tasks. Fig. 1 contrasts SKILL with other learning paradigms. Note how here we use "teacher" and "student" to distinguish the two roles that every agent will perform; this is different from and not to be confused with other uses of student/teacher terminology, for example in knowledge distillation. (4) There is a perfect task oracle at training time, i.e., each agent is told which tasks it should learn. (5) There is a clear separation between tasks, and between training and test phases.

Evaluation metrics:

(1) CPU/computation expenditure. This metric is important to gauge the efficacy of an approach and its ability to scale up with more agents operating in parallel. Wall-clock time is the main metric of interest, so that speedup can be achieved through parallelism. Thus, if N agents learn for 1 unit of time, wall-clock time would be 1, which is an N-fold speedup over a single sequential agent. In practice, speedup $< N$ is expected because of overhead for sharing, communications, and knowledge consolidation. Because wall clock time assumes a given CPU or GPU speed, we instead report the number of multiply-accumulate (MAC) operations. **(2) Network/communication expenditure.** Sharing knowledge over a network is costly and hence should be minimized. To relate communications to computation, and hence allow trade-offs, we assume a factor $\alpha = 1,000$ MACs / byte transmitted. It is a hyperparameter in our results that can be easily changed to adapt to different network types (e.g., wired vs. wireless). **(3) Performance:** After the population of N agents has collectively learned all T tasks, we report aggregated (averaged) performance over all T tasks (correct classification rate over all tasks). Note how here we assume that there is *no task oracle at test time*. After training, agents should be able to handle any input from any task without being told which task that input corresponds to. SKILL does not assume a free task oracle because transmitting training data across agents is potentially very expensive. Thus, agents must also share information that will allow receiving agents to know when a new test input relates to each received task.

Open questions: *What knowledge should be shared?* SKILL agents must share knowledge that is useful to other agents and avoid sharing local or specialized knowledge that may be misleading, in conflict with, or inappropriate to other agents. The shared knowledge may include model parameters, model structure, generalizations/specializations, input data, specific contextual information, etc. There are also size/memory/communication constraints for the shared knowledge. *When and how to share?* Different communication network topologies and sharing frequencies likely would lead to different results. Here, we will sidestep this problem and assume a fully connected communication network, and broadcast sharing from each agent to all others each time a new task has been learned.

B. Related Works

B.1. Lifelong Learning

Lifelong Learning (LL) aims to develop AI systems that can continuously learn to address new tasks from new data, while preserving knowledge learned from previously learned tasks (Masana et al., 2022). It also refers to the ability to continually learn over time by accommodating new knowledge while retaining previously learned experiences (Parisi et al., 2019). LL is challenging because it is usually assumed that the training data from previous tasks is not available anymore while learning new tasks; hence one cannot just accumulate training data over time and then learn from all the data collected so far. Instead, new approaches have been proposed, which fall under three main branches (De Lange et al., 2021): *(1) Regularization methods* add an auxiliary loss term to the primary task objective to constrain weight updates, so as to minimally disturb previously learned knowledge while learning new tasks. The extra loss can be a penalty on the parameters (EWC (Kirkpatrick et al., 2017), MAS (Aljundi et al., 2018) and SI (Zenke et al., 2017)) or on the feature-space (FDR (Benjamin et al., 2018)), such as using Knowledge Distillation (LwF (Li & Hoiem, 2017), DMC (Zhang et al., 2020)). *(2) Parameter-Isolation methods* assign a fixed set of model parameters to a task and avoid over-writing them when new tasks are learned (SUPSUP (Wortsman et al., 2020)), PSP (Cheung et al., 2019) and BPN (Wen et al., 2021). *(3) Rehearsal methods* use a buffer containing sampled training data from previous tasks, as an auxiliary to a new task's training set. The buffer can be used either at the end of the task training (iCaRL, ER (Rebuffi et al., 2017b; Robins, 1995)) or during training (GSS,

AGEM, AGEM-R, GSS, DER, DERPP (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Aljundi et al., 2019; Buzzega et al., 2020)). However, most traditional LL algorithms cannot satisfy the requirement of SKILL: parallel learning for speeding up, and sharing knowledge among agents.

B.2. Multi-task Learning

Multi-Task Learning (MTL) aims to leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks (Zhang & Yang, 2021; Crawshaw, 2020; Ruder, 2017). The main difference between MTL and SKILL is that MTL assumes that all tasks are located in the same physical region, and that one can access the datasets of all tasks at the same time (Zhang & Yang, 2021). While MTL learns multiple tasks together, SKILL assumes that different knowledge sources are separated in different physical regions and different agents should learn them in parallel.

B.3. Federated Learning

Federated learning (FL) is a machine learning setting where many clients (e.g., mobile devices, networked computers, or even whole organizations) collaboratively train a model under the orchestration of a central server, while keeping the training data decentralized (Kairouz et al., 2021; Li et al., 2020; Bonawitz et al., 2019). As shown in Fig. 1, compared with SKILL: (1) FL agents usually learn the *same task* from multiple partial datasets in different locations, relying on the central server to accumulate and consolidate the partial knowledge provided by each agent. In contrast, SKILL agents solve different tasks, and share knowledge over a decentralized network. (2) Each SKILL agent may learn multiple tasks in sequence, and hence must solve the LL problem of accumulating new knowledge while not forgetting old knowledge. Sequences of tasks and the LL problem are usually not a primary focus of FL, with a few exceptions (Yoon et al., 2021) which still do not directly apply to SKILL, as they focus on a single task for all agents and use a central server. Because federated learning relies on a central server, it is susceptible to complete failure if that server is destroyed; in contrast, in SKILL, as long as not all agents are destroyed, the surviving agents can still share and master some of the tasks.

B.4. Other methods that may help solve SKILL

One related direction is to share a compact representation dataset: Dataset distillation (Wang et al., 2018) combines all training exemplars into a small number of super-exemplars which, when learned from using gradient descent, would generate the same gradients as the larger, original training set. However, the distillation cost is very high, which could not satisfy the $0.5N$ speedup requirement. Another related direction is to reuse shared parameters for different tasks: Adversarial reprogramming (Elsayed et al., 2018) computes a single noise pattern for each task. This pattern is then added to inputs for a new task and fed through the original network. The original network processes the combined input + noise and generates an output, which is then remapped onto the desired output domain. However, the cost of the reprogramming training process is high, which could not satisfy the $0.5N$ speedup requirement. Another related direction is to use a generative adversarial network (GAN (Goodfellow et al., 2020)) to learn the distribution of different datasets and generate for replay. Closed-loop GAN (CloGAN (Rios & Itti, 2018)) could be continuously trained with new data, while at the same time generating data from previously learned tasks for interleaved training. However, the GAN needs more parameters to transmit, and the high training time does not satisfy the $0.5N$ speedup requirement.

C. Dataset subsampling details

Our SKILL-102 dataset comprises 102 distinct tasks that were obtained from previously published datasets.

Here, we subsampled the source datasets slightly, mainly to allow some of the baselines to converge in a reasonable amount of time. For dataset sampling, the following rules were used:

- For iNaturalist Insecta, since it contains a lot of classes, 500 classes were randomly sampled.
- For all other tasks, all classes are kept.
- For all tasks, round(54000/c) training images and round(6000/c) validation images and round(6000/c) test images are used for each class. If a class does not contain enough images, then all images for that class are used.
- The exact datasets as we used them in our experiments will be made available online after publication, to allow other researchers to reproduce (or beat!) our results.

The sequence of datasets and number of images in each dataset are shown in Fig. S5.

D. GMMC number of clusters

Fig. S1 shows the GMMC performance with different numbers of clusters.

# of clusters	2	5	10	15	20	25	30	35	40
Accuracy	74.89%	76.98%	78.27%	82.06%	82.08%	82.38%	82.24%	81.79%	81.77%

Figure S1. On a small subset of tasks, we found that $k = 25$ GMMC clusters provided the best compromise between generalization and overfitting.

E. Mahalanobis training MACs

The slope of MACs/image is higher until the number of training samples reaches 4,000. After that, the slope does not change. If we use 5 images per class to train, then the number of training samples would reach 4,000 after task 12. So for the majority of the tasks, the average MACs per image for training the Mahalanobis distance is around 250k.

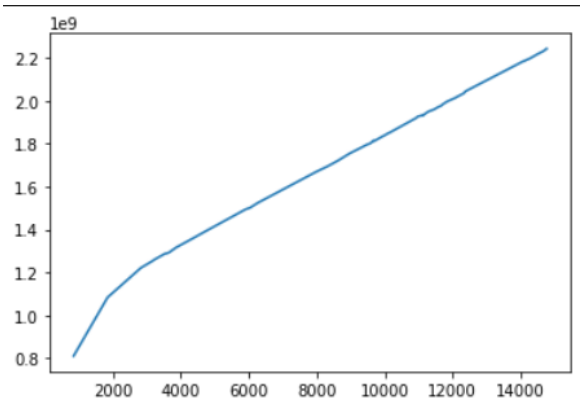


Figure S2. MACs for Mahalanobis training (vertical axis) as a function of number of training images (horizontal axis).

F. CPU analysis

We compute everything in terms of MACs/image processed. There are a few caveats:

- Data sharing does not occur per training image, but rather per task (e.g., share 25 GMMC cluster means+diagonal covariances per task). Hence we first compute communication bytes/task and then convert that to "MACs equivalent" by assuming that sharing 1 byte takes the equivalent of 1,000 MACs. This value is a hyper-parameter than can be tuned depending on network type. Over wired Ethernet, it corresponds to 1.5 million MACs per packet (with MTU of 1500 bytes).
- Mahalanobis training time increases with the number of tasks received to date, as shown in Fig. S2.
- ER training increases over time as more tasks are added:
  - We first train task 1 using the whole task 1 training set (subsampled version described above).
  - Then train task 2 using the whole task 2 training set + 10 images/class of task 1 (chosen randomly). In what follows we use γ to represent this fraction of data used for rehearsing of old tasks, and we denote by S a nominal dataset size per task (2,500 images on average). Hence, for task 2, the episodic buffer method uses $S \times (1+\gamma)$ images. With a normalized training time of 1 to learn one task, learning task 2 for this baseline takes normalized time $1 + \gamma$.
  - Then train task 3 using the whole task 3 training set + 10 images/class of task 1 + 10 images/class of task 2. Normalized training time $1 + 2\gamma$.
  - Then train task 4 using the whole task 4 training set + 10 images/class of task 1 + 10 images/class of task 2 + 10 images/class of task 3. Normalized training time $1 + 3\gamma$.
  - etc. So the total normalized training time for N tasks is $(1)+(1+\gamma)+(1+2\gamma)+(1+3\gamma)+...+(1+(N-1)\gamma) = N+\gamma(1+2+...+N-1) = N+\gamma(N-1)(N-2)/2$. With $N = 102$, the total training time for all tasks is $N+5050\gamma$. In our experiments, our subsampled training sets averaged 254 images/class and hence $\gamma = 10/254 = 0.04$ on average, leading to a total normalized training time of 304 (broken down as a cost of 102 to learn the from 102 datasets, plus 202 to rehearse old tasks as we learn new tasks).
  - This is for $\gamma = 0.04$ but performance is low, so using a higher γ is warranted for the episodic buffer approach. This is very costly, though. In the limit of retaining all images, which would give best performance, the training time of this approach is $102 + 5050 = 5152$ times the time it takes to learn one task. So, while the single-agent will require anywhere between $304 \times T$ and $5152 \times T$ to learn 102 tasks sequentially, our approach will learn all 102 tasks in parallel during just T.

Additional details used for our computations are in Fig. S3.

Teacher	Student	Runtime	
7.58E+11	0.00E+00	8.44E+09	unfrozen xception MACs/image
0.00E+00	0.00E+00	8.44E+09	frozen backbone MACs/image
1.21E+07	0.00E+00	1.01E+05	last layer only MACs/image
1.08E+09	0.00E+00	3.58E+07	BB only MACs/image
4.96E+06	0.00E+00	8.55E+07	GMMC MACs/image
2.50E+05	2.50E+05	6.07E+08	Mahalanobis MACs/image
8.56E+11	0.00E+00	8.44E+09	EWC MACs/image
3.07E+11	0.00E+00	3.42E+09	PSP MACs/image
2.22E+12	0.00E+00	8.44E+09	ER MACs/image
4.95E+11	0.00E+00	4.15E+09	SUPSUP MACs/image
7.61E+11	0.00E+00	8.44E+09	EWC-ONLINE MACs/image
7.65E+11	0.00E+00	8.44E+09	LwF MACs/image
1.02E+12	0.00E+00	8.44E+09	SI MACs/image
1.01E+12	0.00E+00	8.44E+09	MAS MACs/image
7.58E+11	0.00E+00	8.48E+09	BB backbone MACs/image

Parameters	
# tasks (=N)	102
MACs/byte transmitted	1000
bytes/parameter	4
bytes/image	268203
# xception weights	20884814
xception latent dims	2048
xception forward MACs	8.44E+09
# GMMC clusters	25
GMMC params/cluster	4096
avg classes/task	49.34
median classes/task	12.00
avg train img/task	20011.95
avg test img/task	2386.93
Mahalanobis img/class	5
# BB biases	50312
# training epochs (avg)	30
# train images	2041219
# test images	243467
ER replay Factor	2.931372549

Computed from parameters	
GMMC shared bytes/task	409600
last layer bytes/task	404391
BB biases bytes/task	201248
Mahalanobis bytes/task	66165680.1
SUPSUP bytes/task	3000000

Figure S3. Additional details for how we compute MACs and speedup. Different assumptions (e.g., higher or lower MACs/byte transmitted) can be used, which would update the results in the main paper Figs. 9 and 10.

G. Summary of our new SKILL-102 for image classification

Fig. S5 shows a summary of 102 datasets we are using along with the accuracy of all our methods. Note that TM stands for Task Mapper. The red text indicates datasets with large domain gap which were mentioned in Sec. 6, the blue text indicates datasets with poor GMMC accuracy which are further examined below. Fig. S6 shows the baselines performance on SKILL-102.

H. Cases of low accuracy in GMMC

In this section, we analyze in details the failures of GMMC on three datasets: Office Home Art, Dragon Ball, and Malacca Historical Buildings.

In certain cases, several datasets may share a common characteristic, such as all of them are anime pictures (e.g. Dragon Ball, Pokemon, and One-Piece). GMMC may capture the tasks' characteristics as animation but fail to further distinguish different tasks. On the other side, Mahalanobis focus on the characteristics classwise, witch captures the difference among classes (e.g., Wukong vs. Abra characters in Fig. S4-a) and hence is able to distinguish them.

Another case is that two tasks may share similar objects (e.g., Office Home Art, Office Home, and Stanford Online Products; Fig. S4-b). Although represented in different tasks, these are the same types of objects in the real life. We address these GMMC confusions with our proposed "corrective approach" that would declare correct classification for equivalent labels belonging to different tasks.

Other cases may include that one task is too general; for example, Watermark non Watermark includes a large variety of images with or without a watermark which may also confuse GMMC as many similar images are present in other datasets (Fig. S4-c).

I. Amount of data shared by LLL

The analysis below includes 2 options not exercised in the main text of this paper:

- **Head2Toe:** If the input domain encountered by an agent is very different than what the frozen backbone was trained on, sharing only the last layer(s) + BPN biases may not always work well, because the features in the backbone are not able to well represent the new domain. Our backbone is pretrained on ImageNet, which is appropriate for many image classification and visually-guided RL tasks in the natural world. However, the latent features may not be well suited for highly artificial worlds. This was recently addressed by (Evci et al., 2022), who showed that this problem can be alleviated using a last layer that connects to several intermediary layers, or even to every layer in the network, as opposed to only the penultimate layer. Hence, instead of sharing the last layer, we may share a so-called *Head2toe* layer when a large domain shift is encountered. Note that AR will also be used in this case as it is another way to counter large domain shifts: the AR pattern essentially recasts an input from a very different domain back into the ImageNet domain, then allowing the frozen backbone to extract rich and meaningful features in that domain. Also see (Parisi et al., 2022) for ideas similar to Head2toe, with applications in RL.

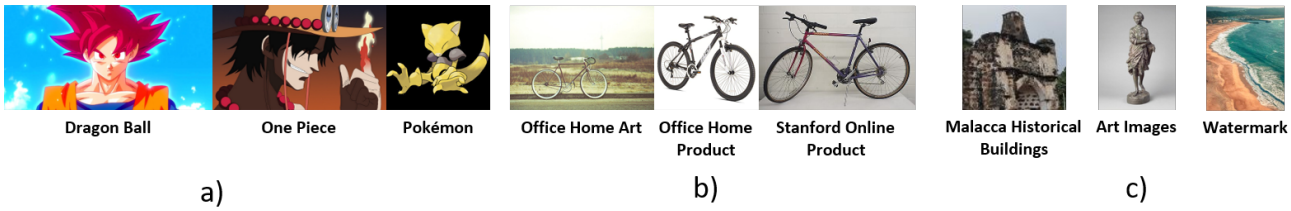


Figure S4. Here we analyze the top 3 tasks into which images may be misclassified by GMMC. a) Out of 18 test images from the (very small) Dragon Ball Dataset, 4 are correctly classified as belonging to Dragon Ball Dataset, 11 are misclassified as belonging to the One Piece dataset, and 2 are misclassified as belonging to the Pokemon dataset. Since all three datasets contain cartoon images, GMMC was confused to classify some images into an incorrect dataset. b) Out of 252 test images from Office Home Art, 86 are correctly classified, 33 are classified as belonging to the Stanford Online Product dataset, and 20 are classified as Office Home Product dataset. These three datasets have many objects in common such as bicycles, chairs, and tables. Hence, it is easy for GMMC to get confused. c) Out of 18 test images from Malacca Historical Buildings, 7 were correctly classified, 5 are classified as Art Images, and 5 are classified as belonging to the Watermark dataset. The Art Image and Watermark datasets contain a large variety of images which may confuse the GMMC to make wrong predictions.

- **Adversarial reprogramming (AR) (Elsayed et al., 2018):** Adversarial reprogramming is quite similar in spirit to BB, with the main difference being that it operates in the input (image) space as opposed to BB operating in the activation space. In adversarial reprogramming, one computes a single noise pattern for each task. This pattern is then added to inputs for a new task and fed through the original network. The original network processes the combined input + noise and generates an output, which is then remapped onto the desired output domain. Unfortunately, the CPU cost of this approach is prohibitive with respect to $0.5N$ speedup.

We denote the number of BB biases by $\mathcal{N}$ in what follows (for xception, $\mathcal{N} = 17,472$). If Head2toe connects to the same feature maps as BB, then the number of weights is $\mathcal{N} \times c$ for c output classes. We assume that each task is modeled with k GMMC clusters ($k = 25$ currently), and each is represented by a 2048D mean and 2048D diagonal covariance. We denote by 4 the number of bytes per floating point number.

For a classification task with c classes: An agent receives an image as input and produces a vector of c output values (on SKILL-102, c is 49.34 on average), where the highest output value is the most likely image class for the input image (Table S1).

J. Discussion and Future Works

We have proposed a new lightweight approach to lifelong learning that outperforms all baselines tested, and also can benefit almost perfectly from parallelization. We tested the approach on a new SKILL-102 benchmark dataset, which we believe is the largest non-synthetized lifelong learning challenge dataset to date. While many previous efforts have employed many tasks, those were usually synthesized, e.g., as permutations over a single base dataset (e.g., 50 permuted-MNIST tasks in (Cheung et al., 2019)). SKILL-102 contains novel real data in each task, with large inter-task variance, which is a better representative of realistic test scenarios. Our proposed lightweight LL points to a new direction in LL research, as we find that one can simply use lightweight task-specific weights (head) combined with maximizing the leverage obtained from task-agnostic knowledge that is rapidly adapted by a compact BB module to handle each new task. Our results show how this lightweight design is better able to handle large scale lifelong learning tasks, and also solves our SKILL challenge very well.

We credit our good performance on both sequential lifelong learning and the SKILL challenge to our particular lightweight lifelong learner design: a *fixed backbone*, which represent task-agnostic knowledge shared among agents, to minimize the complexity of task-specific knowledge parameters (the head); *Beneficial Biases*, which on-demand shift the backbone to solve possibly large domain gaps for each new task, with very compact parameters; a *GMMC/MAHA* global task anchor for learned tasks, representing the tasks in the common task-agnostic latent space of all agents, which is easy to share and consolidate, and which eliminates the need for a task oracle at test time. Our results show that combination of this three components help our LLL work well.

Our approach uses a pretrained backbone to represent task-agnostic knowledge, which our results show is a very effective strategy. For fair comparison, we also use the same pretrained backbone as the initialization for the baselines (except PSP and SUPSUP; see above). However, our fixed backbone design often cannot handle large domain gaps between new tasks and ImageNet. This is the reason why we

Note: TM = Task Mapper	Sample Dataset				Full Dataset				Perfect Task Mapper			GMMC			MAHA	
		Train	Validation	Test		Train	Validation	Test	no BB	BB	TM	no BB	BB	TM	no BB	BB
Dataset Name	# of classes	# of images	# of images	# of images	# of classes	# of images	# of images	# of images	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy
102_Category_Flower_Dataset	102	6455	865	865	102	6455	865	865	93.64%	95.95%	92.95%	87.63%	89.36%	95.38%	90.29%	92.02%
MIT_Indoor_Scenes	67	12350	1587	1587	67	12350	1587	1587	76.69%	77.88%	71.90%	56.96%	57.72%	85.63%	67.49%	68.30%
Caltech-UCSD_Birds_200	200	9405	1191	1191	200	9405	1191	1191	74.22%	74.06%	96.47%	71.70%	71.70%	99.50%	74.06%	73.97%
Stanford_Cars	196	12723	1693	1693	196	12723	1693	1693	53.93%	63.85%	97.22%	53.16%	62.67%	99.65%	53.69%	63.67%
Fine-Grained_Visual_Classification_of_Aircraft	70	8000	1000	1000	70	8000	1000	1000	46.10%	70.60%	99.20%	45.80%	70.30%	99.20%	45.90%	70.50%
VOC_2012_Human_Action_Subset	11	3424	435	435	11	3424	435	435	71.95%	73.79%	70.80%	56.78%	56.09%	67.36%	52.87%	52.64%
Chars74k	62	8902	1150	1150	62	8902	1150	1150	59.57%	70.26%	84.70%	52.61%	61.91%	86.78%	54.43%	64.09%
Stanford_Street_View_House_Numbers	10	54000	6000	6000	10	504323	63047	63047	74.35%	95.10%	93.18%	69.28%	88.65%	89.55%	66.58%	85.20%
iNaturalist_Reptilia	313	53360	5930	5930	313	71751	9026	9026	39.83%	42.07%	84.01%	35.97%	37.77%	88.72%	37.98%	40.03%
iNaturalist_Fungi	341	53161	6121	6121	341	74541	9409	9409	57.34%	61.36%	88.76%	51.23%	55.14%	93.89%	54.88%	58.98%
iNaturalist_Amphibia	170	38220	4815	4815	170	38220	4815	4815	37.20%	39.50%	82.78%	33.25%	35.31%	81.77%	33.79%	35.76%
iNaturalist_Arachnida	153	33674	4245	4245	153	33674	4245	4245	52.49%	56.09%	84.66%	46.15%	48.95%	85.82%	48.20%	50.91%
iNaturalist_Mollusca	169	36921	4659	4659	169	36921	4659	4659	65.06%	67.91%	84.67%	56.62%	59.33%	87.55%	59.61%	61.79%
iNaturalist_Actinopterygii	183	37195	4723	4723	183	37195	4723	4723	61.00%	63.10%	89.86%	56.55%	58.48%	92.25%	58.27%	60.41%
iNaturalist_Insecta	500	54000	6000	6000	2526	549151	69336	69336	66.23%	71.30%	71.87%	50.20%	54.17%	88.22%	62.27%	66.85%
CORe50	10	54000	6000	6000	10	130390	16308	16308	97.85%	99.27%	98.52%	96.42%	97.80%	96.65%	94.68%	95.93%
Sketches	250	15999	2000	2000	250	15999	2000	2000	59.70%	65.80%	98.55%	58.50%	64.55%	99.65%	59.45%	65.55%
WikiArt_Dataset	27	34522	4037	4037	27	62917	7880	7880	50.61%	54.47%	68.91%	34.33%	37.60%	69.93%	36.64%	38.92%
Describable_Textures_Dataset	47	4497	563	563	47	4497	563	563	72.47%	74.07%	64.83%	50.09%	51.69%	77.80%	59.86%	60.57%
GTSRB	43	41463	5184	5184	43	41463	5184	5184	96.66%	99.69%	97.78%	94.54%	97.51%	96.01%	92.92%	95.79%
CelebA	5	40566	4622	4622	5	99837	12483	12483	82.04%	91.19%	97.34%	80.46%	88.92%	95.61%	79.27%	87.43%
Office-Home_Clipart	65	3300	445	445	65	3300	445	445	77.53%	79.55%	61.12%	47.64%	50.11%	68.31%	55.51%	57.30%
Office-Home_Product	65	3361	454	454	65	3361	454	454	90.97%	89.87%	45.59%	42.51%	42.29%	71.81%	68.94%	69.16%
Office-Home_Art	65	1793	252	252	65	1793	252	252	80.95%	79.76%	34.13%	29.76%	29.76%	48.02%	42.86%	43.65%
Food-101	101	54035	5959	5959	101	80738	10100	10100	66.64%	68.43%	78.54%	52.69%	53.83%	81.66%	56.37%	57.61%
EuroSAT	10	21600	2700	2700	10	21600	2700	2700	94.37%	96.15%	95.81%	90.70%	92.26%	94.48%	89.48%	91.04%
PatchCamelyon	2	54000	6000	6000	2	221983	27750	27750	87.63%	88.65%	98.63%	86.38%	87.35%	77.02%	67.30%	67.80%
Diabetic_Retinopathy_Detection	5	28096	3515	3515	5	28096	3515	3515	75.07%	76.75%	93.89%	71.01%	72.48%	72.34%	54.43%	55.80%
RVL-CDIP	16	54000	6000	6000	16	318696	39846	39846	69.47%	76.33%	97.15%	67.47%	74.37%	96.73%	67.18%	74.00%
HistAerial	7	50637	5763	5763	7	109933	13747	13747	86.73%	87.65%	97.78%	84.87%	85.82%	94.88%	82.58%	83.65%
OrigamiSet1.0	3	1193	151	151	3	1193	151	151	82.12%	83.44%	86.75%	70.86%	70.86%	83.44%	66.89%	67.55%
Brazilian_Coins	5	2443	308	308	5	2443	308	308	85.06%	93.18%	97.73%	83.12%	91.56%	99.35%	85.06%	92.86%
iMaterialist_Fashion_2019	46	36117	4537	4537	46	36117	4537	4537	28.72%	29.42%	79.15%	22.94%	23.21%	73.68%	20.08%	20.50%
Rice_Image_Dataset	5	54000	6000	6000	5	59761	7471	7471	99.38%	100.00%	99.87%	99.25%	99.87%	99.95%	99.33%	99.95%
Vegetable_images_Dataset	15	16796	2100	2100	15	16796	2100	2100	100.00%	100.00%	95.14%	95.14%	95.14%	93.57%	93.57%	93.57%
garbage_classification	12	12383	1555	1555	12	12383	1555	1555	94.98%	95.63%	62.19%	59.42%	60.39%	48.87%	46.69%	46.82%
Facial_Expression_Recognition_2013	7	27175	3401	3401	7	27175	3401	3401	53.87%	60.01%	98.06%	52.78%	58.78%	96.30%	52.04%	58.13%
7000_Labeled_Pokemon	150	5305	749	749	150	5305	749	749	69.83%	83.04%	81.04%	58.74%	68.09%	84.78%	63.95%	73.16%
Manga_Facial_Expressions	7	357	49	49	7	357	49	49	53.06%	55.10%	65.31%	34.69%	36.73%	95.92%	48.98%	51.02%
10_Monkey_Species	10	1007	130	130	10	1007	130	130	98.46%	99.23%	86.15%	85.38%	85.38%	100.00%	98.46%	99.23%
Oregon_Wildlife	20	5648	714	714	20	5648	714	714	88.38%	87.68%	91.60%	82.77%	82.07%	89.50%	81.65%	81.23%
Blood_Cell_Images_Dataset	4	10005	1254	1254	4	10005	1254	1254	83.33%	87.00%	100.00%	83.33%	87.00%	99.84%	83.17%	86.84%
Retinal_OCT_2017	4	42158	4896	4896	4	61339	7669	7669	85.11%	89.44%	99.88%	85.01%	89.32%	99.29%	84.50%	88.79%
APTOS_2019_Blindness_Detection	5	2798	353	353	5	2798	353	353	79.60%	79.32%	88.95%	70.54%	71.10%	92.35%	72.52%	71.67%
Cataract_Dataset	4	479	61	61	4	479	61	61	60.66%	60.66%	65.57%	45.90%	47.54%	95.08%	59.02%	60.66%
Freiburg_Groceries_Dataset	25	3921	506	506	25	3921	506	506	78.66%	79.05%	94.66%	74.90%	75.30%	95.06%	75.69%	75.69%
Fashion_Product_Images_Dataset	43	34885	4384	4384	43	34885	4384	4384	95.87%	96.44%	71.92%	69.41%	69.80%	63.16%	60.22%	60.61%
Apparel_Images_Dataset	24	9080	1146	1146	24	9080	1146	1146	83.33%	94.42%	70.77%	60.82%	67.02%	73.47%	63.44%	70.51%
Zalando_Clothing_and_Models	6	8527	1070	1070	6	8527	1070	1070	88.88%	90.75%	89.44%	79.07%	81.12%	80.47%	71.50%	72.62%
PlantDoc-Dataset	27	2017	265	265	27	2017	265	265	59.62%	62.64%	74.72%	45.66%	47.55%	79.25%	47.17%	51.70%
Images_LEGO_Bricks	50	32000	4000	4000	50	32000	4000	4000	81.88%	90.05%	99.35%	81.28%	89.43%	99.53%	81.43%	89.58%
Art_Images_Type_Classification	5	5684	713	713	5	5684	713	713	88.64%	90.88%	67.32%	61.85%	62.69%	65.92%	61.85%	62.41%
Multi-Class_Weather_Dataset	4	875	112	112	4	875	112	112	92.86%	98.21%	65.18%	64.29%	64.29%	72.32%	69.64%	72.32%
Simpsons_Characters_Data	42	17464	2209	2209	42	17464	2209	2209	72.88%	87.37%	96.51%	70.67%	84.97%	94.66%	70.35%	84.07%
Intel_Image_Classification	6	13597	1703	1703	6	13597	1703	1703	92.48%	92.60%	83.21%	77.33%	77.75%	84.20%	77.80%	78.21%
House_Room_Image_Dataset	5	4136	519	519	5	4136	519	519	88.63%	88.63%	86.51%	76.88%	76.88%	68.02%	60.50%	60.50%
UIUC_Sports_Event_Dataset	8	1258	160	160	8	1258	160	160	100.00%	99.37%	69.38%	69.38%	68.75%	73.13%	73.13%	72.50%
Land-Use_Scene_Classification	21	8399	1050	1050	21	8399	1050	1050	96.00%	96.29%	86.86%	83.43%	83.52%	88.48%	85.05%	85.33%
ASL_Alphabets_Dataset	29	53998	6003	6003	29	69600	8700	8700	99.78%	99.95%	99.60%	99.38%	99.55%	99.78%	99.57%	99.73%
Yoga-82	82	15486	1982	1982	82	15486	1982	1982	52.72%	64.78%	86.63%	47.68%	57.82%	83.40%	46.97%	56.66%
Russian_Letter_Dataset	33	30097	3785	3785	33	30097	3785	3785	81.51%	93.34%	99.05%	80.74%	92.47%	99.10%	80.98%	92.52%
UMIST_Face_Database	20	788	112	112	20	788	112	112	100.00%	100.00%	91.96%	91.96%	91.96%	100.00%	100.00%	100.00%
iFood2019	251	53815	6009	6009	251	104150	13159	13159	42.69%	44.55%	72.14%	30.82%	32.15%	85.52%	37.28%	38.92%
Oxford_Buildings	11	665	90	90	11	665	90	90	85.56%	83.33%	74.44%	68.89%	65.56%	90.00%	78.89%	76.67%
Texture_Dataset	64	6808	861	861	64	6808	861	861	99.65%	99.88%	90.59%	90.36%	90.48%	98.03%	97.68%	97.91%
electronic-components	36	8087	1033	1033	36	8087	1033	1033	47.82%	46.08%	88.38%	42.01%	41.14%	88.67%	42.21%	41.24%
Hurricane_Damage_Dataset	2	16838	2106	2106	2	16838	2106	2106	95.82%	97.91%	96.44%	92.64%	94.54%	88.79%	85.61%	87.32%
chest_xray	2	4658	583	583	2	4658	583	583	96.74%	97.60%	99.49%	96.23%	97.08%	97.08%	93.83%	94.68%
PAD-UFES-20	6	1807	231	231	6	1807	231	231	64.50%	67.10%	74.89%	49.35%	50.22%	85.71%	54.98%	57.14%
Brain_Tumor_Dataset	4	2292	289	289	4	2292	289	289	87.89%	92.39%	97.92%	86.16%	90.66%	99.65%	87.54%	92.04%
Kannada-MNIST	10	48000	6000	6000	10	48000	6000	6000	98.12%	99.42%	99.60%	97.73%	99.05%	99.90%	98.02%	99.32%
Breast_Ultrasound	3	620	79	79	3	620	79	79	82.28%	83.54%	84.81%	69.62%	72.15%	96.20%	79.75%	81.01%
BookCover30	30	45575	5700	5700	30	45575	5700	5700	26.46%	27.89%	83.44%	21.21%	22.49%	82.89%	21.47%	22.68%
boat-types-recognition	9	1160	150	150	9	1160	150	150	93.33%	93.33%	74.67%	72.00%	71.33%	76.67%	73.33%	72.67%
rock-classification	7	1620	206	206	7	1620	206	206	75.73%	76.21%	57.77%	45.15%	44.17%	63.11%	49.03%	48.54%
dermnet	23	14238	1794	1794	23	14238	1794	1794	44.20%	45.82%	95.21%	41.64%	43.42%	91.36%	40.25%	41.14%
dragon-ball-super-saiyan-dataset	6	112	18	18	6	112	18	18	50.00%	55.56%	22.22%	22.22%	16.67%	94.44%	44.44%	50.00%
concrete-crack	2	30720	3841	3841	2	30720	3841	3841	99.84%	99.92%	98.54%	98.44%	98.49%	95.81%	95.68%	95.76%
Malacca_Historical_Buildings	3	126	18	18	3	126	18	18	100.00%	100.00%	38.89%	38.89%	38.89%	100.00%	100.00%	100.00%
Satellite_Images_to_Predict_African_Poverty	4	20455	2558	2558	4	20455	2558	2558	77.01%	81.04%	90.34%	69.62%	73.53%	79.75%	62.16%	65.01%
Skin_Cancer_MNIST_HAM10000	7	8003	1005	1005	7	8003	1005	1005	77.91%	77.31%	96.42%	75.82%	75.02%	89.95%	70.95%	70.15%
watermarked-not-watermarked-images	2	24917	3115	3115	2	24917	3115	3115	70.88%	82.02%	54.70%	39.07%	45.04%	29.82%	21.25%	24.69%
Large-Scale_Fish_Dataset	9	7526	941	941	9	7526	941	941	99.89%	100.00%	98.51%	98.41%	98.51%	97.56%	97.45%	97.56%
DeepWeedsX	9	13996	1756	1756	9	13996	1756	1756	81.66%	84.17%	94.99%	77.68%	79.84%	91.80%	74.83%	76.88%
IP102_Dataset	102	36302	4303	4303	102	60089	7564	7564	59.08%	59.49%	69.77%	41.90%	42.57%	66.77%	41.78%	42.41%
planets-and-moons-dataset	11	1304	165	165	11	1304	165	165	99.39%	100.00%	96.97%	96.97%	96.97%	99.39%	98.79%	99.39%
polish-craft-beer-labels	100	6275	848	848	100	6275	848	848	99.41%	100.00%	99.29%	98.70%	99.29%	100.00%	99.41%	100.00%
Kvasir-Capsule_Dataset	14	30208	3784	3784	14	30208	3784	3784	93.97%	96.51%	99.84%	93.90%	96.43%	98.84%	92.97%	95.40%
NEU-Surface-Defect-Database	6	1439	180	180	6	1439	180	180	100.00%	100.00%	87.78%	87.78%	87.78%	99.44%	99.44%	99.44%
colorectal-histology-mnist	8	3992	504	504	8	3992	504	504	88.69%	87.50%	94.25%	84.13%	82.74%	97.02%	85.71%	84.72%
100_Sports	100	11558	1500	1500	100	11558	1500	1500	92.27%	92.73%	78.07%	72.20%	72.13%	92.40%	86.33%	86.20%
Labeled_Surgical_Tools_and_Images	5	2400	302	302	5	2400	302	302	85.76%	94.70%	97.02%	83.11%	92.05%	92.05%	78.48%	87.75%
Mechanical_Tools_Classification_Dataset	8	5861	737	737	8	5861	737	737	91.18%	90.64%	79.10%	73.81%	73.00%	62.14%	58.21%	57.26%
Galaxy10	10	14083	1766	1766	10	14083	1766	1766	63.53%	77.01%	99.55%	63.08%	76.61%	99.60%	63.25%	76.73%
Stanford_Online_Products	12	54000	6000	6000	12	94375	11804	11804	81.57%	80.82%	72.95%	62.50%	62.10%	53.45%	45.30%	44.78%
NWPU-RESISC45	45	25192	3150	3150	45	25192	3150	3150	86.57%	89.78%	84.89%	72.83%	75.87%	93.40%	80.79%	83.81%
FaceMask_Dataset	3	11001	1377	1377	3	11001	1377	1377	98.55%	98.77%	88.96%	87.65%	88.02%	83.22%	82.21%	82.57%
OnePiece_Dataset	18	9191	1156	1156	18	9191	1156	1156	78.11%	84.69%	86.07%	67.30%	73.10%	84.95%	67.65%	73.18%
ilab_80m	15	12000	1500	1500	15	12000	1500	1500	90.33%	90.40%	96.53%	87.20%	87.13%	95.60%	86.33%	86.33%
CLEVR_v1.0	8	54000	6000	6000	8	67994	8503	8503	48.73%	73.65%	99.90%	48.68%	73.55%	99.97%	48.73%	73.63%
ilab_atari	67	52735	5921	5921	67	295048	36911	36911	99.97%	100.00%	99.09%	99.07%	99.09%	99.27%	99.24%	99.27%
ilab_deepvp	9	53911	6003	6003	9	59422	7433	7433	58.09%	90.07%	97.63%	57.45%	88.77%	90.35%	54.67%	83.47%
Total # / Average Accuracy	5033	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	67.43%	70.58%	87.10%	68.87%	72.10%

Figure S5. Basic stats of datasets and accuracy on TM task mapper.

Dataset Name	EWC	PSP	ER	MAS	SI	LwF	Online-EWC	SUPSUP
102_Category_Flower_Dataset	0.58%	1.04%	57.23%	1.62%	0.92%	0.35%	0.46%	50.64%
MIT_Indoor_Scenes	3.21%	3.02%	9.01%	4.91%	2.71%	2.02%	1.89%	28.86%
Caltech-UCSD_Birds_200	0.25%	0.42%	21.24%	0.67%	0.50%	0.59%	0.34%	12.59%
Stanford_Cars	0.41%	0.59%	13.59%	0.65%	0.47%	0.41%	0.35%	6.44%
Fine-Grained_Visual_Classification_of_Aircraft	1.70%	1.90%	19.70%	5.80%	2.90%	2.40%	2.90%	24.10%
VOC_2012_Human_Action_Subset	9.20%	15.86%	2.76%	11.72%	15.86%	7.59%	7.59%	20.92%
Chars74k	0.96%	3.22%	22.00%	3.13%	1.48%	0.17%	0.70%	73.22%
Stanford_Street_View_House_Numbers	11.05%	9.98%	55.85%	9.52%	10.43%	9.32%	8.80%	91.93%
iNaturalist_Reptilia	0.44%	0.20%	2.56%	0.32%	0.34%	0.30%	0.32%	1.70%
iNaturalist_Fungi	0.31%	0.34%	13.67%	0.28%	0.29%	0.20%	0.42%	8.45%
iNaturalist_Amphibia	0.56%	0.69%	3.26%	0.66%	0.46%	0.69%	0.62%	4.11%
iNaturalist_Arachnida	0.64%	0.73%	6.12%	0.97%	0.64%	0.90%	0.49%	5.91%
iNaturalist_Mollusca	0.75%	0.64%	11.05%	0.69%	0.56%	0.67%	0.71%	8.97%
iNaturalist_Actinopterygii	0.70%	0.51%	9.65%	0.83%	0.38%	0.42%	0.57%	5.27%
iNaturalist_Insecta	0.27%	0.17%	14.73%	0.25%	0.10%	0.33%	0.17%	3.77%
CORe50	10.35%	10.37%	19.03%	10.22%	11.77%	9.25%	10.17%	81.62%
Sketches	0.25%	0.65%	43.60%	0.40%	0.45%	0.05%	0.45%	19.65%
WikiArt_Dataset	3.12%	6.69%	4.58%	6.47%	1.41%	2.65%	4.24%	26.13%
Describable_Textures_Dataset	1.95%	1.95%	12.08%	4.26%	1.42%	1.60%	1.95%	20.60%
GTSRB	3.67%	4.26%	74.46%	5.34%	3.03%	4.57%	2.91%	86.67%
CelebA	12.12%	25.98%	50.58%	25.94%	25.96%	20.45%	10.80%	80.44%
Office-Home_Clipart	1.35%	1.57%	21.12%	3.15%	1.57%	1.57%	1.80%	39.55%
Office-Home_Product	2.20%	1.10%	26.21%	1.32%	2.20%	1.76%	0.44%	44.05%
Office-Home_Art	4.76%	1.59%	3.97%	4.37%	1.19%	3.97%	3.17%	9.92%
Food-101	1.12%	1.04%	8.00%	1.16%	1.26%	1.16%	1.19%	11.90%
EuroSAT	12.22%	10.41%	50.37%	20.96%	11.11%	7.85%	8.26%	84.11%
PatchCamelyon	12.60%	49.75%	41.08%	51.27%	52.48%	3.48%	7.28%	83.58%
Diabetic_Retinopathy_Detection	52.96%	32.09%	15.99%	72.40%	70.15%	26.15%	5.88%	72.57%
RVL-CDIP	6.33%	7.72%	29.40%	5.97%	6.33%	4.15%	8.07%	65.43%
HistAerial	8.47%	24.08%	49.44%	23.10%	15.50%	10.93%	1.94%	63.28%
OrigamiSet1.0	36.42%	34.44%	6.62%	50.33%	23.18%	16.56%	13.25%	43.05%
Brazilian_Coins	16.56%	4.55%	51.62%	21.43%	19.81%	1.62%	3.57%	93.18%
iMaterialist_Fashion_2019	1.43%	2.07%	1.68%	17.57%	0.42%	1.17%	0.82%	24.71%
Rice_Image_Dataset	14.48%	51.83%	82.18%	33.10%	20.00%	19.65%	8.72%	99.15%
Vegetable_Images_Dataset	9.90%	11.14%	51.71%	6.95%	7.90%	5.86%	9.29%	95.90%
garbage_classification	4.50%	33.57%	5.72%	37.49%	5.40%	6.69%	5.98%	68.75%
Facial_Expression_Recognition_2013	12.88%	24.26%	26.64%	17.82%	17.17%	14.64%	11.76%	44.46%
7000_Labeled_Pokemon	0.40%	0.93%	53.81%	0.80%	0.67%	0.40%	0.67%	43.93%
Manga_Facial_Expressions	12.24%	18.37%	40.82%	22.45%	14.29%	16.33%	14.29%	30.61%
10_Monkey_Species	9.23%	1.54%	43.85%	11.54%	9.23%	7.69%	9.23%	43.08%
Oregon_Wildlife	6.02%	6.58%	12.89%	5.04%	5.88%	6.30%	4.76%	31.65%
Blood_Cell_Images_Dataset	21.93%	17.86%	35.41%	24.96%	25.20%	14.27%	15.95%	93.70%
Retinal_OCT_2017	11.76%	31.41%	57.56%	30.92%	29.78%	15.77%	29.13%	86.66%
APTOS_2019_Blindness_Detection	42.78%	51.56%	43.63%	51.84%	17.85%	20.96%	5.95%	67.99%
Cataract_Dataset	16.39%	47.54%	50.82%	39.34%	16.39%	9.84%	19.67%	49.18%
Freiburg_Groceries_Dataset	7.31%	3.75%	22.13%	8.50%	5.14%	4.55%	4.94%	44.07%
Fashion_Product_Images_Dataset	0.25%	34.08%	40.08%	36.95%	4.88%	0.57%	0.59%	43.25%
Apparel_Images_Dataset	4.28%	8.99%	30.37%	6.11%	8.55%	4.89%	5.15%	82.02%
Zalando_Clothing_and_Models	8.97%	41.40%	32.34%	46.07%	16.26%	24.11%	19.72%	72.43%
PlantDoc-Dataset	5.28%	4.15%	13.96%	7.17%	3.77%	3.77%	3.77%	18.11%
Images_LEGO_Bricks	1.68%	3.33%	28.90%	1.73%	2.18%	2.70%	2.60%	4.45%
Art_Images_Type_Classification	13.46%	33.94%	8.42%	37.59%	16.83%	21.46%	10.10%	76.72%
Multi-Class_Weather_Dataset	17.86%	43.75%	25.00%	45.54%	32.14%	24.11%	6.25%	83.93%
Simpsons_Characters_Data	2.04%	3.98%	22.45%	10.82%	5.93%	2.22%	2.90%	77.14%
Intel_Image_Classification	12.16%	31.24%	21.96%	20.26%	15.68%	9.92%	6.28%	80.50%
House_Room_Image_Dataset	21.19%	26.40%	10.79%	19.27%	17.73%	26.20%	18.69%	47.78%
UIUC_Sports_Event_Dataset	10.00%	20.00%	20.63%	28.75%	13.13%	12.50%	10.00%	61.25%
Land-Use_Scene_Classification	7.52%	7.05%	34.38%	14.00%	4.76%	3.81%	3.52%	67.90%
ASL_Alphabets_Dataset	4.45%	32.13%	58.35%	5.43%	3.28%	3.28%	4.11%	96.39%
Yoga-82	1.21%	1.06%	12.46%	2.57%	1.46%	0.91%	1.21%	27.70%
Russian_Letter_Dataset	2.99%	30.12%	44.76%	3.57%	1.59%	6.58%	4.83%	86.92%
UMIST_Face_Database	2.68%	24.11%	89.29%	5.36%	6.25%	4.46%	3.57%	99.11%
iFood2019	0.37%	0.78%	6.57%	0.63%	0.45%	0.22%	0.28%	4.58%
Oxford_Buildings	10.00%	25.56%	50.00%	38.89%	38.89%	4.44%	6.67%	52.22%
Texture_Dataset	1.74%	3.48%	80.60%	6.27%	1.05%	1.51%	1.28%	91.99%
electronic-components	2.81%	2.71%	9.68%	4.74%	3.19%	4.07%	3.78%	24.10%
Hurricane_Damage_Dataset	4.51%	85.57%	48.81%	65.34%	34.24%	20.47%	45.77%	94.82%
chest_xray	7.38%	71.36%	63.81%	72.90%	27.10%	6.86%	2.23%	95.71%
PAD-UFES-20	12.99%	32.47%	23.38%	31.60%	17.75%	11.69%	1.73%	50.65%
Brain_Tumor_Dataset	13.15%	37.37%	59.52%	30.10%	26.64%	8.30%	12.46%	79.93%
Kannada-MNIST	11.73%	98.97%	93.65%	10.02%	10.00%	12.10%	9.57%	98.85%
Breast_Ultrasound	0.00%	59.49%	45.57%	17.72%	54.43%	29.11%	13.92%	70.89%
BookCover30	3.67%	4.33%	3.56%	4.46%	2.81%	3.02%	3.77%	12.56%
boat-types-recognition	18.00%	22.00%	8.00%	34.00%	17.33%	13.33%	9.33%	34.00%
rock-classification	11.65%	26.21%	6.80%	24.27%	17.48%	9.71%	10.19%	31.55%
dermnet	5.02%	5.57%	13.10%	5.07%	2.79%	3.85%	3.07%	26.42%
dragon-ball-super-saiyan-dataset	22.22%	27.78%	33.33%	0.00%	16.67%	27.78%	16.67%	33.33%
concrete-crack	8.10%	98.54%	81.31%	49.00%	45.77%	1.17%	0.83%	99.11%
Malacca_Historical_Buildings	5.56%	77.78%	88.89%	33.33%	33.33%	22.22%	27.78%	66.67%
Satellite_Images_to_Predict_African_Poverty	7.66%	28.46%	29.36%	29.44%	21.23%	8.80%	1.64%	64.19%
Skin_Cancer_MNIST_HAM10000	1.49%	61.49%	31.74%	66.77%	66.37%	2.69%	15.32%	71.94%
watermarked-not-watermarked-images	13.32%	57.37%	1.06%	54.64%	50.75%	9.47%	4.01%	79.78%
Large-Scale_Fish_Dataset	11.37%	51.33%	74.18%	13.28%	10.95%	9.56%	10.41%	96.28%
DeepWeedsX	17.65%	53.02%	29.33%	51.88%	51.20%	10.59%	35.88%	65.95%
IP102_Dataset	0.49%	3.25%	8.58%	1.86%	1.46%	0.65%	0.91%	14.36%
planets-and-moons-dataset	7.88%	43.64%	96.36%	11.52%	7.88%	9.70%	6.67%	100.00%
polish-craft-beer-labels	1.53%	11.91%	95.87%	2.83%	1.53%	0.59%	0.12%	95.40%
Kvasir-Capsule_Dataset	0.63%	77.80%	36.73%	72.60%	25.08%	0.37%	1.14%	90.12%
NEU-Surface-Defect-Database	10.00%	73.89%	81.11%	32.78%	16.67%	6.67%	7.22%	93.33%
colorectal-histology-mnist	6.35%	28.37%	64.68%	31.75%	12.50%	8.73%	4.96%	89.68%
100_Sports	1.93%	4.73%	28.27%	3.80%	1.47%	0.93%	1.67%	40.20%
Labeled_Surgical_Tools_and_Images	12.25%	32.78%	53.31%	45.03%	24.83%	11.26%	13.58%	90.40%
Mechanical_Tools_Classification_Dataset	5.29%	26.73%	6.65%	32.16%	25.64%	9.77%	7.33%	43.83%
Galaxy10	11.61%	14.84%	39.30%	12.29%	14.95%	5.95%	11.49%	51.02%
Stanford_Online_Products	9.12%	26.77%	8.83%	20.92%	8.47%	8.47%	7.92%	39.28%
NWPU-RESISC45	1.97%	23.30%	41.40%	11.81%	2.25%	2.25%	2.03%	53.08%
FaceMask_Dataset	13.94%	80.46%	75.16%	62.31%	29.99%	8.50%	9.44%	94.34%
OnePiece_Dataset	8.22%	25.17%	22.15%	9.86%	5.71%	7.27%	6.83%	55.10%
ilab_80m	8.87%	40.27%	46.00%	26.33%	7.60%	7.73%	7.93%	67.00%
CLEVR_v1.0	10.12%	60.70%	29.35%	18.32%	20.47%	11.03%	13.17%	65.72%
ilab_atari	6.23%	99.93%	99.71%	14.63%	1.45%	7.50%	5.57%	99.27%
ilab_deepvp	91.84%	88.59%	93.02%	90.60%	41.41%	89.91%	92.09%	84.41%
Average Accuracy	8.86%	25.49%	35.32%	20.54%	13.89%	8.41%	7.77%	56.22%

Figure S6. Accuracy of baselines after all 102 tasks have been learned in the order shown.

Shared params and data	Size (bytes)	Implemented: $\mathcal{N} = 17,472, c = 49.34, k = 25$
Last layer weights	$2048 \times c \times 4$	404 KBytes
BB biases	$\mathcal{N} \times 4$	70 KBytes
GMMC clusters	$k \times (2048 + 2048) \times 4$	409 KBytes
Optional: Head2toe	adds $\mathcal{N} \times c \times 4$	adds 3.45 MBytes
Optional: AR pattern	adds $299 \times 299 \times 3$	adds 268 KBytes
Alternative: 5 images/task for MAHA	$5 \times 299 \times 299 \times 3$	1.34 MBytes

Table S1. Total average sharing per task in our current implementation with GMMC+BB: 404+70+409 = 883 KBytes/task; for Mahalanobis+BB: 404+70+1341=1.81 MBytes/task.

proposed BB to relieve the domain gap by shifting the fixed parameters towards each new task with compact biases. Similar to other parameter-isolation methods, our model structure is growing on demand (though slowly) with the number of tasks (we add a new head per task, while they add new masks, keys, etc). Rehearsal-based baselines (e.g., ER) also grow, by accumulating more rehearsing exemplars over time. While some baselines do not grow (e.g., EWC), they also perform very poorly on our challenging SKILL-102 dataset.

To further speed up our method with BB, we could use BB on only partial layers. For instance, if we use BB only the last half of the layers in the backbone, we will use only half of the current time to train the model. In future experiments, we will test whether this still gives rise to a significant accuracy benefit.

Currently, we use the CLIP embedding space to match a new class with learned old classes, which uses only language knowledge (class labels). As a future work, we will use GMMC as a class matching mechanism to utilize the visual semantic information for matching. Specifically, when a agent learns a new class, the agent will collect a few shots (e.g., 10 images) of the new class and then use the GMMC mapper (trained on all previous tasks) to decide whether these images belong to a learned task or not, with a threshold. If most of the images are matched to a learned task, we can then summon the shared head of that task to classify them, now to obtain a possible match for individual previously learned classes. If most of the images are classified into one previously learned specific class, we can use the weights of that class to initialize the new class weights, similar to what we have done in Sec. **??**.

A good task mapper is essential in our approach if one is to forego the use of a task oracle. Thankfully, our results show that task mapping can be achieved with high accuracy even with a simple model like GMMC (over 84% correct for 102 tasks). Indeed, in our SKILL challenge, the task mapper is only solving a 102-way classification problem at the granularity of tasks, vs. a 5,033-way full SKILL classification challenge. Here, we focused on GMMC and MAHA, but many other alternatives could be investigated in future work. Our choice of GMMC was based on previous research that compared it to several other techniques, including self-organizing maps and multilayer perceptrons (Rios & Itti, 2020).

K. Shared Knowledge Accumulation, Reuse and Boost

As our system learns many tasks, it may occur that some tasks overlap with others, i.e., they may share similar images and/or class labels. Here, we explore two approaches to handle such overlap.

K.1. Corrective approach to task overlap/synergy

Since our LLL learners can learn a large number of tasks while solving SKILL problems, some synergy can occur across tasks if one is able to detect when two classes from two different tasks are equivalent, as shown in Fig. S7. We implemented a method to compare the semantic distance of the predicted class name and the actual class name. Originally, after the GMMC infers the task that a test image may come from, we would immediately consider the image as misclassified if the predicted task is wrong. With the consideration of semantic similarity between class names, we will now always load the prediction head corresponding to the predicted task given by GMMC and use it to infer the class name. If the class was guessed incorrectly, but, in the end, the final class name is equivalent to the correct one, then we can declare success on that particular test image (Fig. S7). To obtain the pairwise similarity, we constructed a similarity matrix that stores the semantic distance, measured by the cosine similarity of word embeddings, for all the class names. Those embeddings were obtained from CLIP's (Radford et al., 2021) text encoder based on GPT-2. If the similarity between the predicted class name and the actual class name is greater than a threshold (empirically chosen for now), then we declare it a correct prediction.

As our 102-task dataset contains 5,033 object classes, the full similarity matrix is triangular 5,033 x 5,033 (too large to display here).

Figure S7. Left: similar classes with a cosine similarity in the CLIP embedding greater than 0.90. Right: similar classes with a cosine similarity greater than 0.95. This can help correct spurious errors where, for example, a test image from class "bike" from the Stanford_Online_Products dataset could also be considered correctly classified if the system output were "bicycle" from the Sketches dataset.

The approach yields a small but consistent improvement in accuracy (Fig. S8). This is one way in which we can handle possible overlap between tasks, which may inevitably arise when large numbers of tasks are learned.

K.2. Learning approach to task overlap/synergy

The ability to reuse learned knowledge from old tasks to boost the learning speed and accuracy of new tasks is a potential desirable feature of LL algorithms. Here, this might be achieved if, when learning a new task, an LLL agent could "borrow" the knowledge of old tasks, from not only itself but also the shared knowledge from any other agents.

One important design feature of our LLL agents is that they can share *partial heads* across tasks: Our heads are a single layer with 2,048 inputs (from the xception backbone) and c outputs for a task with c classes. Thus, each of the c output neurons is connected to the 2,048 inputs, but there are no lateral connections. This means that we can consider each of the c output neurons individually as evidence provider for their associated class (remember the analogy to "grandmother cells" in Introduction). We can then cherry-pick sets of 2,048 weights corresponding to individual classes of previously learned tasks, and use them as initialization for some similar classes in new tasks to be learned. As we show below, this greatly accelerates learning of the similar new classes, compared to starting with randomized initial weights, and also yields higher accuracy on these new classes.

1) New task is a combination of old tasks: To validate our idea, a new learning paradigm is proposed to use previously learned weights when a new task contains classes that were already learned previously. This experiment considers two datasets and two sets of weights representing the old knowledge, and a new dataset that contains all classes from both datasets. Simply normalizing and concatenating the linear weights leads to poor performance. Hence, instead, we normalize the weights during training by their *p-norm*, and concatenate the normalized weights as the new task's weights. The experiment was conducted over 190 combinations of 2 datasets chosen from 20 datasets, and the average results show that there is a very small accuracy loss initially (epoch 0). After a few extra training epochs, we reach a higher accuracy than training new weights from scratch (random initialization; Fig. S9).

The mathematical version. During training, a constraint is added. Let W_1 be the full set of $2,048 \times c$ weights of the first dataset, and W_2 be the weights of the second dataset, where $W_1 = \begin{pmatrix} \cdots w_1^1 \cdots \\ \vdots \\ \cdots w_n^1 \cdots \end{pmatrix}$ and $W_2 = \begin{pmatrix} \cdots w_1^2 \cdots \\ \vdots \\ \cdots w_n^2 \cdots \end{pmatrix}$. Normal Linear Layer training's forward path is $\hat{y}$ = Wx. Define W' as $\begin{pmatrix} w_1' \\ \vdots \\ w_n' \end{pmatrix}$ where $w_i' = w_i/$p-norm(w_i) Hence, during training, Linear Layer training's forward path is $\hat{y}$ = W'x. And concatenate(W_1', W_2') was used as the weights for the new combined task.

Since the weights are normalized class-wise and not task-wise, this method can be used on any combination of previously leaned classes. For example, for 10 tasks containing 100 classes c_1 c_{100} and a new task containing c_1, c_3, c_{10}, c_{20}, we can simply find the corresponding $w_1', w_3', w_{10}', w_{20}'$, and concatenate them together.

Choice of p: We find that using the 2-norm causes the classifier to converge to a state where all weight vectors have the same magnitude, which causes an accuracy drop for the old task. Hence, we choose to use the infinity norm, which is still modulated by the weight magnitudes, and is still easy to transfer.

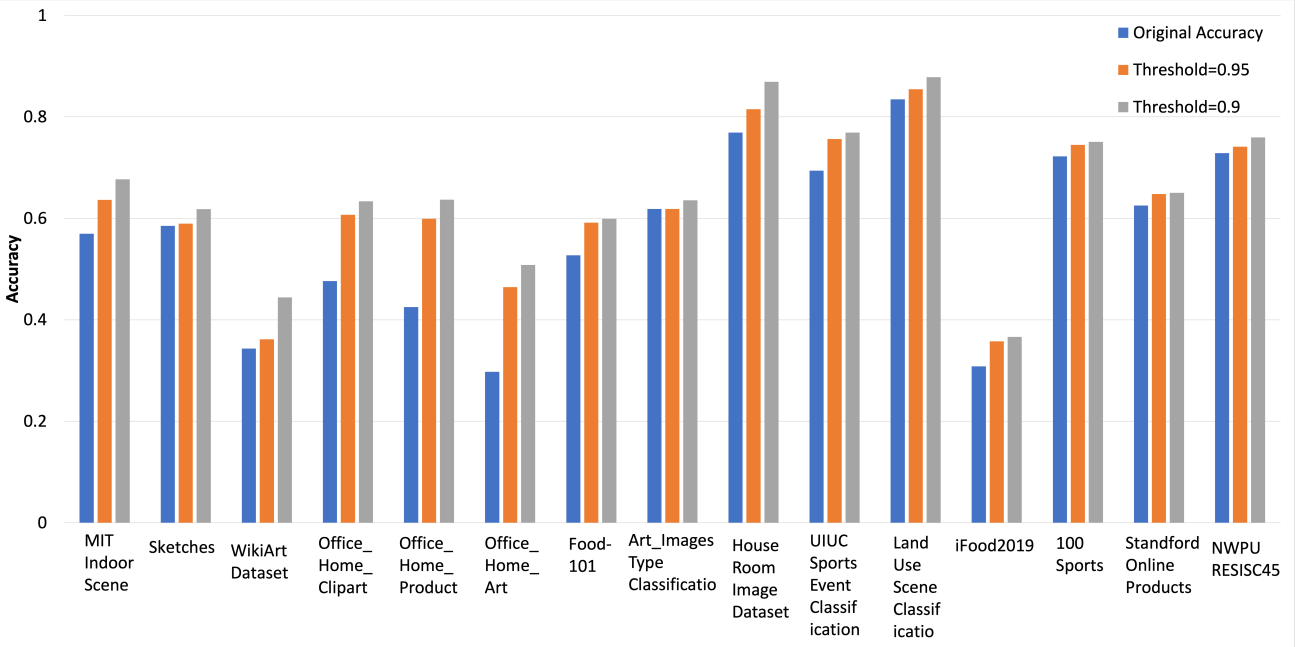


Figure S8. Correcting spurious errors by realizing when two distinct classes from two tasks actually are the same thing. The approach provides a small but consistent improvement in accuracy over baseline (which declares failure as soon as task mapping failed), here shown on 15 datasets that have some overlap.

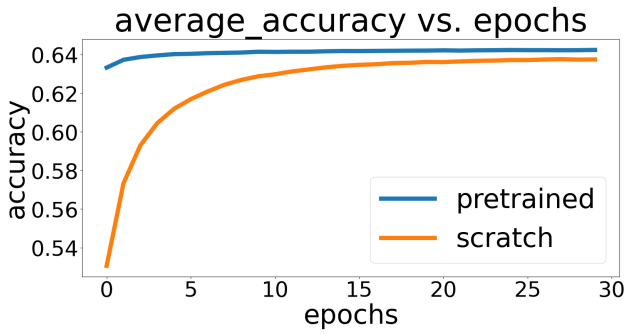


Figure S9. Learning speed for a given object class when the corresponding weights are initialized randomly (orange) vs. from previously learned weights of a similar object class found in one of the previously learned tasks (blue), averaged for 190 combinations of two previously learned tasks. In this example, best accuracy is already reached after just 1 to 2 training epochs in the blue curve. In contrast, it takes up to 30 epochs to train from random initialization, with still a final accuracy in the orange curve that is lower than the blue curve. This approach hence leads to significant learning speedup when tasks contain some similar classes.

2) New task is different but similar to old tasks. In the previous setting, we assumed the new task classes are a combination of different old task classes. In a more general situation, the new task classes are all new classes that all other agents never learned before, but we could still borrow some learned knowledge from similar learned classes. For instance, as shown in Fig. S7, the knowledge of classes shown on top of the figure may be helpful to learn the new classes shown at the bottom.

We conduct four different experiments (for 4 pairs of datasets that share some related classes) to show the knowledge boost when we learn a new task. We first check if a learned old task shares similar knowledge with the new one. For instance, before we learn the *MIT indoor scenes dataset*, we find that the *House Room Image Dataset* contains classes that are similar to the new classes, in the CLIP embedding space. So we match each class from *MIT indoor scenes dataset* to the previously learned classes, which in this case come from the *House Room Image Dataset*. If the class similarity is larger than a threshold, we treat it as a matched class, then we use the similar old class weights to initialize the weights of the new class. If a new class was not matched with old classes, we use random initialization. We also conduct a corresponding controlled experiment by using random initialization for all new classes. The results of all 4 experiments are shown in Table S2

In a more general learning scenario, the new task classes

	Datasets	initialization	All	10 shot	5 shot	3 shot
Old task	House Room Image Dataset	random	0.86	0.77	0.73	0.52
New task	MIT Indoor Scenes	ours	0.89	0.83	0.8	0.71
Old task	Standford Online Products	random	0.78	0.61	0.61	0.6
New task	Office Home Product	ours	0.8	0.62	0.59	0.6
Old task	100 Sports	random	1	0.98	0.92	0.92
New task	UIUC Sports Event Dataset	ours	1	0.99	0.97	0.97
Old task	iFood2019	random	0.61	0.42	0.35	0.3
New task	Food-101	ours	0.64	0.5	0.46	0.43

Table S2. Boosted LLL learning when previously learned weights from similar classes can be used to initialize learning of new classes. We repeat the experiment with either learning from all images in the training set of the new task, or only 10, 5, or 3 images per class. Overall, re-using previously learned weights of similar classes boosts accuracy, usually (but not always) more so when the new task is only learned from a few exemplars (which is much faster than learning from scratch from the whole dataset).

may correspond to similar but not necessarily identical classes in different old tasks. For example, having learned about SUVs may help learn about vans more quickly. Here we conduct two new experiments (Fig. S10). In EXP-1, the new task is sketch image classification, and the classification weights of each class are initialized from a learned old class that is harvested from different learned tasks with the help of CLIP-based word matching. For instance, the classification weights of *van* are initialized with the weights of *SUV* from the previously learned *Stanford-CARs dataset*; the classification weights of *Topwear* are initialized with the weights of *t-shirt* from the learned *Fashion-Product* dataset, etc (total: 5 pairs of classes). The results show that our initialization leads to better performance (0.98 v.s. 0.93) and faster convergence (3 v.s. 4 epochs) compared with random initialization. This shows that our method can reuse the shared knowledge from other agents to improve the learning of new tasks. Similar performances are shown in EXP-2, which is identical to EXP-1 except that it uses 5 different pairs of classes.

K.3. Further boost with Head2Toe

A possibly complementary approach to BB to address domain gaps is Head2Toe (Evci et al., 2022), where the last layer can now directly draw from potentially any of the previous layers in the backbone. This has been shown to help alleviate domain gaps, as some of the lower-level features in the backbone may be useful to solve tasks with a big gap, even though the top-level backbone features may not. However, Head2Toe has a very high computation cost to select which layers should connect to the head, which is why we have not used it in our main results. Here, we explore how that cost of selection of the most appropriate layers to connect to the head for a given task can be eliminated by re-using the computations already expended for BB: Intuitively, layers which have large absolute BB magnitude may also be the most useful to connect to the head.

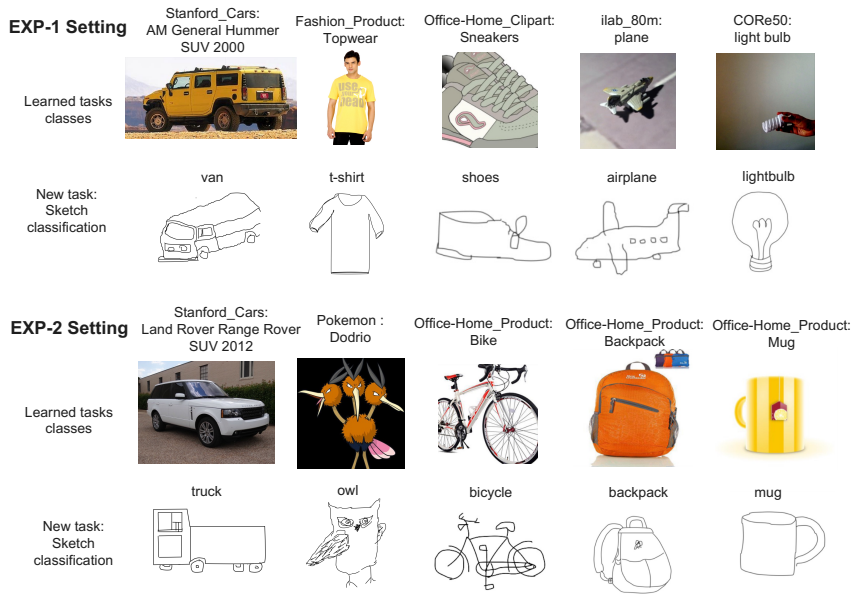
Compared to the conventional Head2Toe (Evci et al., 2022) with two-stage training (first, select which layers will connect to the head, then train those connections), our new BB+H2T uses the biases that have been previously trained and stored in the BB network for feature selection. Specifically, we first concatenated all the biases in the BB network and selected the top 1% largest biases. Then, we picked the feature maps corresponding to the selected indices, average pooled them and flattened them to 8,192-dimensional vectors. After that, we concatenated all flattened feature vectors along with the logits of the last layer (after pooling layer, before softmax) in the BB network. Finally, we trained the concatenated vector with Adam optimizer, 0.001 learning rate, and 100 epochs. This approach, when combined with BB and MAHA, improved performance averaged over all tasks by 0.78% (when a perfect task mapper is available; or by 0.56% when using MAHA).

L. Additional Experiment Results

L.1. Baselines

We implemented 8 baselines from the literature. For those that require a task oracle, we (unfairly to us) grant them a perfect task oracle (while our approach uses imperfect GMMC or Mahalanobis task mappers). When possible, we re-implement the baselines to use the same pretrained xception backbone as our approach. This ensures a fair comparison where every approach is granted the same amount of pre-training knowledge and the same feature processing ability. The two exceptions are PSP (Cheung et al., 2019) that uses ResNet-18, and SUPSUP (Wortsman et al., 2020) that uses ResNet-50.

Our baselines fall in the following 3 categories (De Lange et al., 2021): *(1) Regularization methods* add an auxiliary loss term to the primary task objective to constraint weight updates. The extra loss can be a penalty on the parameters (EWC (Kirkpatrick et al., 2017), MAS (Aljundi et al.,



EXP-1 Results

Weight Initialization	Accuracy	Converge Epochs
Random	0.93	4
Ours	**0.98**	**3**

EXP-2 Results

Weight Initialization	Accuracy	Converge Epoch
Random	0.90	4
Ours	**0.95**	**3**

Figure S10. Two experiments where the weights from previously learned similar but not identical classes are successful in boosting learning of new classes. Left: pairs of similar classes (according to CLIP). Right: accuracy achieved with weight transfer vs. random initialization.

2018) and SI (Zenke et al., 2017)) or on the feature-space (FDR (Benjamin et al., 2018)), such as using Knowledge Distillation (DMC (Zhang et al., 2020)). We use EWC as the representative of this category: one agent learns all 102 tasks in sequence, using EWC machinery to constrain the weights when a new task is learned, to attempt to not destroy performance on previously learned tasks. We also use SI, MAS, LwF, and Online-EWC as baselines of this type. *(2) Parameter-Isolation methods* assign a fixed set of model parameters to a task and avoid over-writing them when new tasks are learned (SUPSUP (Wortsman et al., 2020), PSP (Cheung et al., 2019)). We use PSP as the representative of this category: one agent learns all 102 tasks in sequence, generating a new PSP key for each task. The keys help segregate the tasks within the network in an attempt to minimize interference. We used the original PSP implementation, which uses a different backbone than ours. PSP accuracy overall hence may be lower because of this, and thus we focus on trends (decline in accuracy as more tasks are added) as opposed to only absolute accuracy figures. We also used SUPSUP as baseline of this type. *(3) Rehearsal methods* use a buffer containing sampled training data from previous tasks, as an auxiliary to a new task's training set. The buffer can be used either at the end of the task training (iCaRL, ER (Rebuffi et al., 2017b; Robins, 1995)) or during training (GSS, AGEM, AGEM-R, GSS, DER, DERPP (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Aljundi et al., 2019; Buzzega et al., 2020)). We use ER and as the representative of this category: One agent learns all 102 tasks in sequence. After learning each task, it keeps a memory buffer with 10 images/class (size hence keeps increasing when new tasks are learned) that will later be used to rehearse old tasks. When learning a new task, the agent learns from all the data for that task, plus rehearses old tasks using the memory buffer.

L.2. Accuracy on first task

To gauge how well our approach is achieving lifelong learning, we plot the accuracy on the first task as we learn from 1 to 102 tasks, in Fig. S11. There is nothing special in our dataset about the first task, except that it is the first one. A good LL system is expected to maintain its accuracy on task 1 even as more subsequent tasks are learned; conversely, catastrophic interference across tasks would rapidly decrease task 1 accuracy with more learned tasks. Overall, our approach maintains the highest accuracy on task 1 over time, and virtually all of the accuracy degradation over time is due to increasing confusion in the task mapper (e.g., curves for Mahalanobis task mapper alone and LLL w/BB w/MAHA are nearly shifted versions of each other). Indeed, once the task is guessed correctly, the corresponding head always performs exactly the same, no matter how many tasks have been learned.

L.3. Task mapper accuracy after learning 1 to 102 tasks:

To investigate how well our approach is expected to scale with more tasks, we computed task mapper accuracy on all tasks learned so far, after learning 1, 2, 3, ... 102 tasks. This allows us to evaluate degradation with more tasks that is due to increasing confusion in the task mapper, as opposed to being due to classification difficulty of newly added tasks. Results are shown in Fig. S12: Task mapping accuracy starts at 100% after learning 1 task (all test samples are correctly assigned to that task by Mahalanobis or GMMC), then decreases as more tasks are learned, eventually still achieving 87.1% correct after 102 tasks for MAHA, and 84.94% correct for GMMC. It is important to note that in our approach, any loss in accuracy with more tasks only comes from the task mapper: once the correct head is selected for a test sample, the accuracy of that head remains the same no matter how many heads have been added to the system. In contrast, other baseline methods may suffer from catastrophic forgetting for both the task mapper and the classification model when more tasks are learned, as further examined below.

When using GMMC task mapping, the regression line is $y = -0.0012x + 0.952$, which intercepts zero for $T = 800$ tasks. Thus, with the distribution of tasks in our dataset, we extrapolate that $T = 500$ is realistic as is. Since task interference in our system only comes from GMMC, pushing beyond $T = 500$ might require more than $k = 25$ GMMC clusters per task, which would increase CPU and communications expenditure. When using Mahalanobis task mapping, the results are similar with an intercept at $T = 978$, though this approach incurs a slightly higher communications cost (discussed below).

L.4. Normalized accuracy on first 10 tasks

We compare our method to the baselines on the first 10 tasks, when up to 20 subsequent tasks are learned. A good LL system should be able to maintain accuracy on the first 10 tasks, while at the same time learning new tasks. Because in SKILL-102 different tasks have different levels of difficulty, we normalize accuracy here to focus on degradation with an increasing number of new tasks. For example, the accuracy of our method (LLL w/o BB) when learning a single task is 92.02% for task 1, but only 52.64% for task 6, which is much harder. Here, we define a **normalized accuracy** as the accuracy divided by the initial accuracy just after a given task was learned (which is also the best ever accuracy obtained for that task). This way, normalized accuracy starts at 100% for all tasks. If it remains near 100% as subsequent tasks are learned, then the approach is doing a good job at minimizing interference across tasks. Conversely, a rapidly dropping normalized accuracy with an increasing number of subsequent tasks learned indicates that catastrophic interference is happening.

Our results in Fig. S13 show that, although not perfect, our approach largely surpasses the baselines in its ability to maintain the accuracy of previously learned tasks, with the exception of SUPSUP, which suffers no degradation (see caption of Fig. S13 for why).

L.5. Computation and communication costs, SKILL metrics:

The baselines are sequential in nature, so trying to implement them using multiple agents does not make sense as it would only add communication costs but not alleviate the sequential nature of these LL approaches. For example, for the EWC baseline, one could learn task 1 on agent A then communicate the whole xception weights to agent B (22.9 M parameters = 91.6 MBytes) plus the diagonal of the Fisher Information matrix (another 22.9 M parameters), then agent B would learn task 2 and communicate its resulting weights and Fisher matrix to agent C, etc. Agent B cannot start learning task 2 before it has received the trained weights and Fisher matrix from agent A because EWC does not provide a mechanism to consolidate across agents. Thus, we first consider one agent that learns all 102 tasks sequentially, with no communication costs.

Table. 1 shows the computation expenditures (training time in terms of the number of multiply-accumulate (MAC) operations needed to learn all 102 datasets) for our approach and the baselines. Our approach overall has by far the lowest computation burden when BB is not used, yet all 4 variants of our approach perform better than all baselines. BB increases accuracy but at a significant computation cost: This is because, to compute BB biases, one needs to compute gradients through the entire frozen backbone, even though those gradients will only be used to update biases while the weights remain frozen in the backbone.

Our approach presents the advantage that it can also be parallelized over multiple agents that each learn their own tasks in their own physical region. All agents then learn their assigned tasks in parallel. Each agent is the "teacher" for its assigned tasks, and "student" for the other tasks. Then all agents broadcast their shared knowledge to all other agents. As they receive shared knowledge, the students just accumulate it in banks, and update their task mapper. After sharing, all agents know all tasks (and are all identical). As mentioned above, the main source of performance degradation in our approach is in the task mapper, which gets increasingly confused at T increases.

For our baselines, we are not aware of a way to parallelize their operation, except that we were able to create a modified version of SUPSUP that works on several parallel pro-

cessors. In our modified SUPSUP, each agent learns a mask for each of its tasks, then communicates its masks to all other agents. At test time, we (unfairly to us) grant it a perfect task oracle, as our GPUs did not have enough memory to use the built-in task mapping approach of SUPSUP, given our 102 tasks and 5,033 classes (this would theoretically require 1.02 TB of GPU memory).

Table. S3 shows the computation and networking expenditures for our approach and our modified SUPSUP to learn all tasks in the SKILL-102 dataset. Because some algorithms run on GPU (e.g., xception backbone) but others on CPU (e.g., GMMC training), and because our tasks use datasets of different sizes, we measure everything in terms of MACs (multiply-accumulate operations, which are implemented as one atomic instruction on most hardware). To measure MACs for each component of our framework, we used a combination of empirically measured, framework-provided (e.g., pyTorch can compute MACs from the specification of all layers in a network), or sniffed (installing a hook in some algorithm that increments a counter each time a MAC is executed). To translate communication costs to MACs, we assume a nominal cost of $\alpha = 1,000$ MACs to transmit one byte of data. This is a hyperparameter in our results that can be changed based on deployment characteristics (e.g., wireless vs. wired network). The amount of data shared per task for our approach is quite small (details in Suppl. Sec. G): 813 KBytes/task for LLL with GMMC, no BB; 883 KBytes/task for LLL with GMMC, BB; 1.74 MBytes for LLL with MAHA, no BB; and 1.81 MBytes for LLL with MAHA, BB (on average, given that our tasks have 49.34 classes each on average; see Suppl. Fig. S5).

Our results in Table. S3 show:

1. **Our approach has very low parallelization overhead**, which leads to almost perfect speedup $> 0.99N$ for all variants. Indeed, teachers just learn their task normally, plus a small overhead to train GMMC on their own tasks, when GMMC is used. Communications are less than 2 MBytes per task (Suppl. Sec. G). Students either do nothing (just accumulate received knowledge in a bank) or update their Mahalanobis task mapper.

2. **The baselines have comparatively much higher training cost, yet their performance is poor.** Performance of episodic buffer / rehearsing methods might be improved further by increasing buffer size, but note that in the limit (keeping all training data for future rehearsing), this gives rise to a $> 5,000\times$ increase in training time (Suppl. Sec. D).

M. GMMC visual explanation

A visual explanation of how GMMC works in LLL agents is shown in Fig. S14.

N. Pairs of similar classes according to CLIP

Table S4, Table S5, Table S6, and Table S7 show examples of pairs of similar classes according to CLIP embedding. The first and the second column are the names of similar class pairs from two different tasks (i.e iFood2019 and Food-101). The third column is the cosine similarity score between the CLIP embeddings of the name of the class pairs.

O. Performance on Visual Domain Decathlon

We also perform our methods on a well-known benchmark Visual Domain Decathlon (Ke et al., 2020) in Fig. S15. The baselines and our method implementations are the same as the experiments in SKILL-102 dataset.

P. Inspiration from neroscience

Many approaches to LL involve at least partially retraining the core network that performs tasks (feature extraction backbone plus classification head), as every new task is learned. But transmitting and then merging these networks across multiple agents would incur very high communications and computation costs. With the exception of perceptual learning, where human visual cortex may indeed be altered when learning specific visual discrimination tasks for days or weeks (Goldstone, 1998; Dosher & Lu, 2017), there is little evidence that our entire visual cortex — from early stage filters in primary visual cortex to complex features in inferotemporal cortex — is significantly altered when we just learn, e.g., about a new class of flowers from a few exemplars. Instead, the perirhinal cortex (and more generally the medial temporal lobe) may be learning new representations for new objects by drawing upon and combining existing visual features and representations from visual cortex (Deshmukh et al., 2012). This may give rise to specialized "grandmother cells" (Bowers, 2017) (or Jennifer Aniston neurons; (Quiroga et al., 2005; Quiroga, 2017)) that can be trained on top of an otherwise rather immutable visual cortex backbone. While the grandmother cell hypothesis remains debated in neuroscience (vs. distributed representations; (Valdez et al., 2015)), here, it motivates us to explore the viability of a new lightweight lifelong learning scheme, where the feature extraction backbone and the latent representation are fixed, and each new object class learned is represented by a single new neuron that draws from this representation.

Table S3. Analysis of computation and network expenditures for our parallelized LLL approach and our parallelized SUPSUP, to learn all $T = 102$ tasks. Our approach supports any number of agents N such that $1 \leq N \leq T$. Maximum speedup is expected when $N = T$ and each agent learns one task, then shares with all others. Here, we report numbers for $T = 102$, $N = 51$, and each agent learns 2 tasks in sequence. Note that in our approach, accuracy is not affected by N, only the amount of parallelization speedup increases with N. Note how in this table we still report MACs but taking parallelization into account (e.g., teacher CPU for N agents is single-agent CPU divided by N). **Teacher CPU:** Time to learn tasks from their training datasets, plus to possibly prepare data for sharing (e.g., compute GMMC clusters). **Communications:** Our LLL agents communicate either GMMC clusters or Mahalanobis training images, while our modified SUPSUP communicates masks. Here we assume that there is a communication bottleneck at the receiver (student): the shared data from 100 tasks needs to be received serially, over a single networking interface for each student. Hence our communication figures are for all the shared data from all other tasks apart from those an agent learned itself. We convert communication costs to equivalent MACs by assuming 1,000 MACs per byte transmitted. BB adds a small extra communication cost, to transmit the biases. **Student CPU:** For GMMC, students do not do any extra work (hence, student CPU is 0); for Mahalanobis, students compute a covariance matrix for all 102 tasks. **Speedup factor:** is just total MACs for single agent divided by total MACs for parallel agents and by N. All approaches here achieve near perfect parallelization ($> 0.99N$, where $1.0N$ is perfect). **Accuracy:** In addition to being faster when BB is not used, our LLL variants still all outperform the parallel SUPSUP in accuracy, by a large margin ($> 10\%$).

	Teacher CPU (MACs)	Communi--cations (bytes)	Student CPU (MACs)	Total (MACs)	Parallelization efficiency (xN)	CPU usage vs. Ours-SKILL, no BB, MAHA	Average accuracy after learning 102 tasks
LLL(Ours)-Multiple Agents, no BB, GMMC	1.69E+14	8.22E+07	0.00E+00	1.69E+14	0.99999519	~0.96x	67.43%
LLL(Ours)-Multiple Agents, BB, GMMC	1.53E+16	1.03E+08	0.00E+00	1.53E+16	0.999999934	~87.2x	70.58%
LLL(Ours)-Multiple Agents, no BB, Mahalanobis	**1.69E+14**	**6.72E+09**	**5.00E+09**	**1.76E+14**	**0.996630551**	**1x (reference)**	**68.87%**
LLL(Ours)-Multiple Agents, BB, Mahalanobis	1.53E+16	6.74E+09	5.00E+09	1.53E+16	0.999962712	~87.3x	72.1%
Parallel SUPSUP, Perfect Task Oracle	9.91E+15	3.03E+08	0.00E+00	9.91E+15	0.999999697	~56.4x	56.22%

Table S4. Matched Class for MIT_Indoor_Scene and House_Room_Images

	learned_class(weight source)	target_class	score
0	Dinning	dining_room	0.9106
1	Kitchen	kitchen	0.9995
2	Bathroom	bathroom	1.0
3	Bedroom	bedroom	1.0
4	Livingroom	livingroom	1.0

Q. 4 Main Roles of SKILL Agents

1) Training: agents use a common pre-trained and frozen backbone, stored in ROM memory at manufacturing time (gray trapezoid with lock symbol). The backbone allows the agent to extract compact representations from inputs (e.g., with an xception backbone, the representation is a latent vector of 2048 dimensions, and inputs are 299×299 RGB images). Each agent learns a task-specific head (red triangle) for each new task. A head consists of the last fully-connected layer of the network plus our proposed LL beneficial biasing units (BB) that provide task-dependent tuning biases to all neurons in the network (one float number per neuron). During training, each agent also learns a GMMC or Mahalanobis task anchor which will form a task mapper. **2) Share knowledge** with other agents: each agent shares the learned task-specific head, Beneficial Bias (BB), and GMMC module (or training images for Mahalanobis) with all other agents. **3) Receive knowledge** from other agents: each agent receives different heads and GMMC/Mahalanobis task mapper anchors from other agents. All heads are stored in a head bank and all task anchors are consolidated to form a task mapper. **4) Testing:** At test time, an input is first processed through the task mapper. This outputs a task ID, used to load up the corresponding head (last layer + beneficial biases) from the bank. The network is then equipped with the correct head and is run on the input to produce an output.

Table S5. Matched Class for Office-Home_Product and Standford_Online_Products

	learned_class(weight source)	target_class	score
0	stapler	Paper_Clip	0.757
1	toaster	Oven	0.838
2	coffee	Mug	0.882
3	cabinet	File_Cabinet	0.9126
4	lamp	Lamp_Shade	0.9453
5	sofa	Couch	0.9736
6	bicycle	Bike	0.978
7	mug	Mug	1
8	chair	Chair	1
9	fan	Fan	1
10	kettle	Kettle	1

Table S6. Matched Class for UIUC_Sports_Event_Dataset and 199_Sports

	learned_class(weight source)	target_class	score
0	bocce	bowling	0.7837
1	badminton	tennis	0.823
2	sailing	sailboat racing	0.9
3	snowboarding	snow boarding	0.9355
4	RockClimbing	rock climbing	0.989
5	polo	polo	0.9995
6	croque_madame	croque_madame	1
7	Rowing	rowing	1

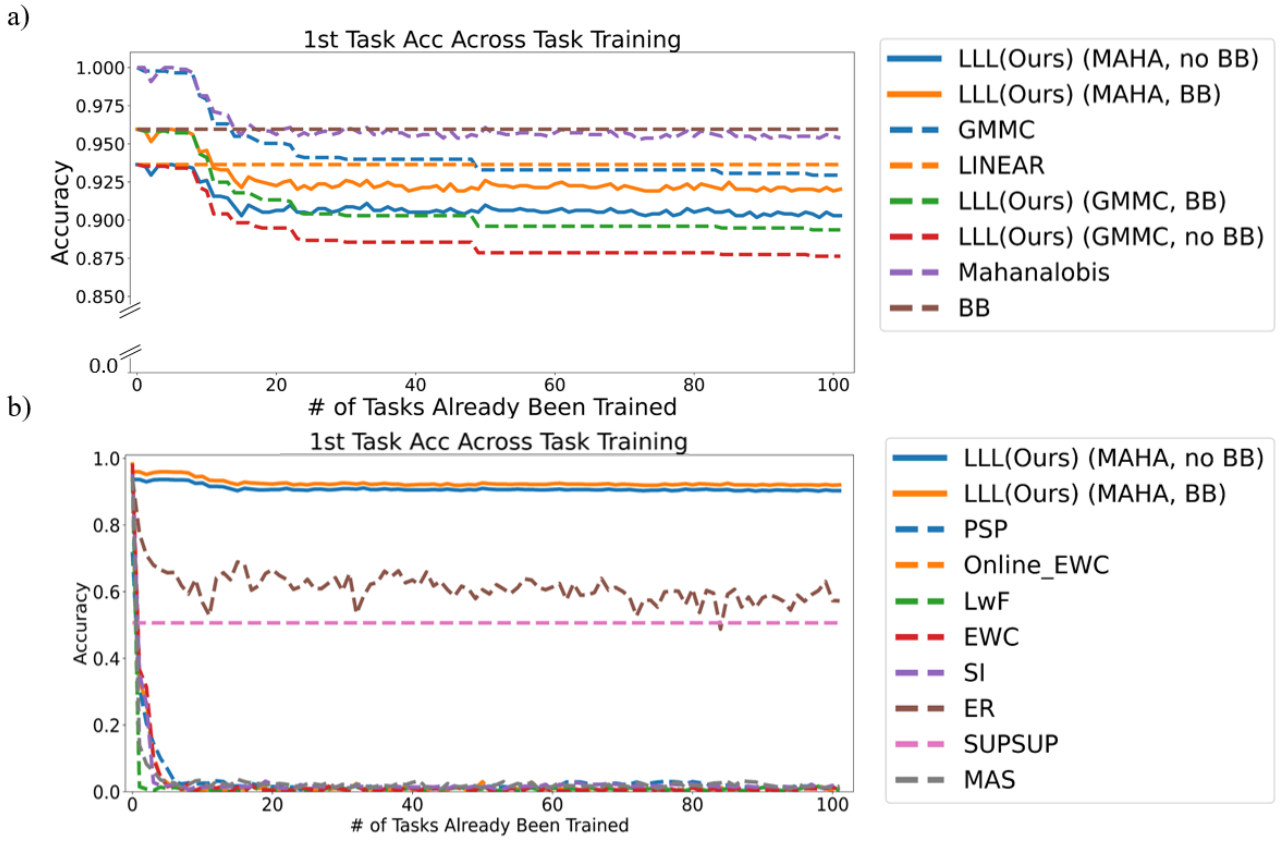


Figure S11. Accuracy on task 1 (learning to classify 102 types of flowers) as a function of the number of tasks learned. a) Comparison between our methods. b) Comparison between our best and other baselines. Our approach is able to maintain accuracy on task 1 much better than the baselines as more and more tasks are learned: while our approach does suffer some interference, task 1 accuracy remains to within 90% of its initial best even after learning 101 new tasks (for the 4 LLL variants, BB=beneficial biases, MAHA=Mahalanobis Distance task mapper, GMMC=GMMC task mapper). In contrast, the accuracy of EWC, PSP, and several other baselines on task 1 catastrophically degrades to nearly zero after learning just 10 new tasks, even though we granted these methods a perfect task oracle. The best performing baseline, ER, is of the episodic buffer type (a fraction of the training set of each task is retained for later rehearsing while learning new tasks), with an un-bounded buffer that grows by 10 images/class. This methods does incur higher (and increasing) training costs because of the rehearsing (Suppl. Sec. D.) Note how SUPSUP does not experience any degradation on task 1, which is a desirable feature of this approach. However, a drawback is that SUPSUP is not able, even from the beginning, to learn task 1 as well as other methods (50.64% accuracy vs. over 90% for most other approaches). We attribute this to SUPSUP's limited expressivity and capacity to learn using masks over a random backbone, especially for tasks with many classes. Indeed, SUPSUP can perform very well on some other tasks, usually with a smaller number of classes (e.g., 91.93% correct on SVHN, 93.18% on Brazillian Coins, 99.11% on UMNIST Face Dataset; see Supp. Fig. S6).

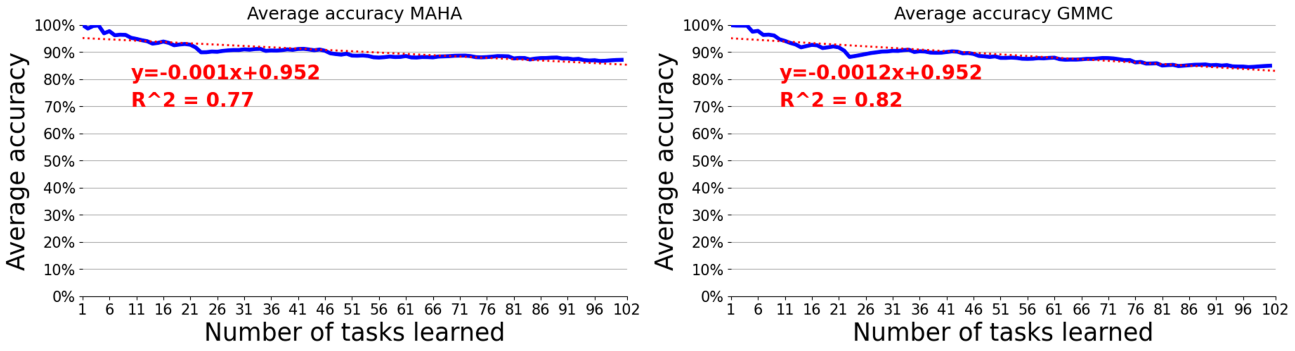


Figure S12. Task mapper accuracy on all tasks learned so far, as a function of the number of tasks learned, when using Mahalanobis (left) or GMMC (right) task mappers. Our approach is able to maintain good task mapping accuracy as the number of tasks increases.

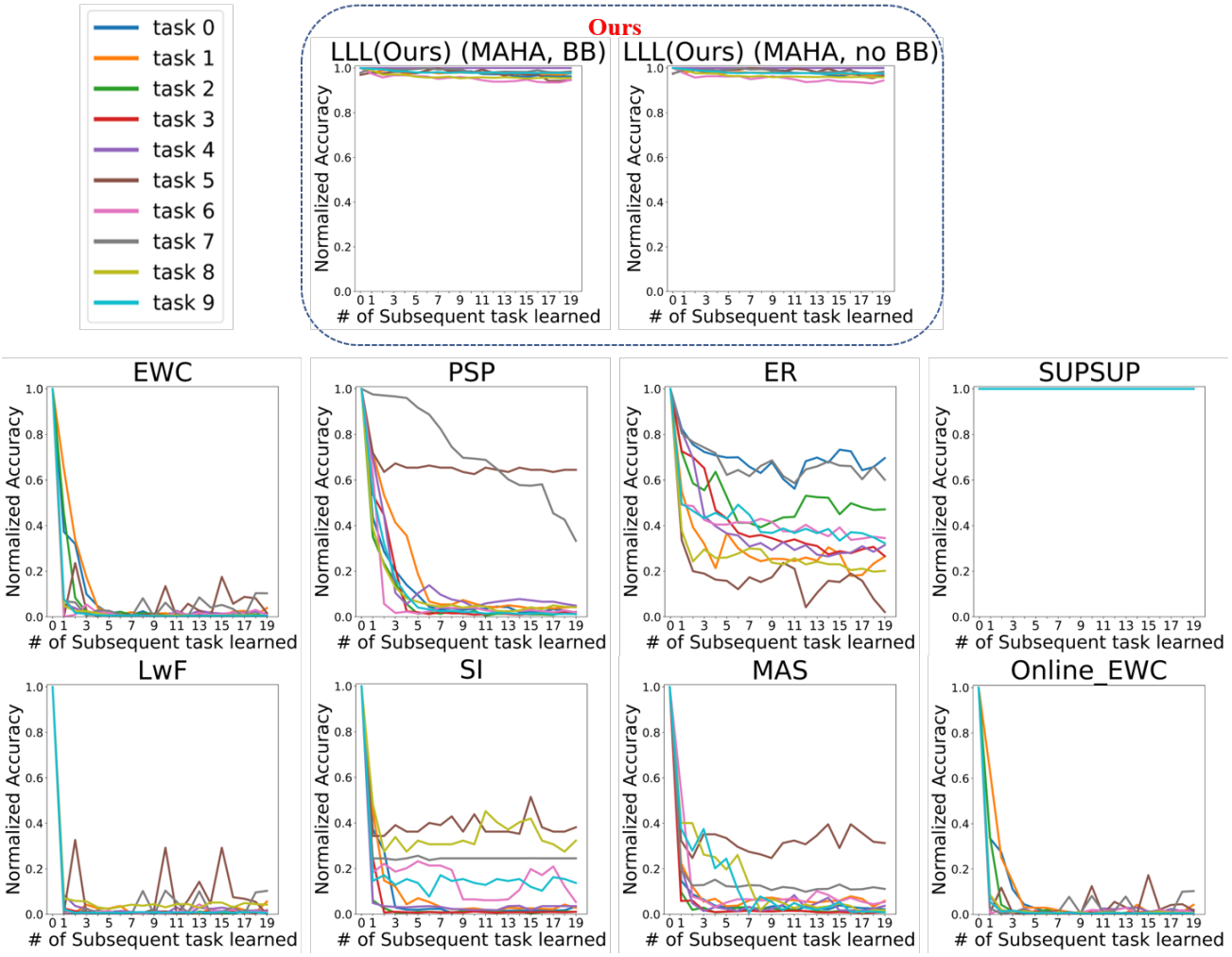


Figure S13. Normalized accuracy on the first 10 tasks (one per curve color) as up to 20 additional tasks are learned. Our LLL approach is able to maintain high normalized accuracy on the first 10 tasks, while all other baselines except SUPSUP suffer much stronger catastrophic interference. SUPSUP is a special case as there is no interference among successive tasks when a perfect task oracle is available. Hence normalized accuracy for all tasks remains at 100%. However, we will see below that the absolute accuracy of SUPSUP is not as good.

Table S7. Food-101 vs iFood2019

	learned_class(weight source)	target_class	score
0	cheese_plate	grilled_cheese_sandwich	0.8574
1	cup_cakes	cupcake	0.904
2	steak	steak_au_poivre	0.9185
3	scallops	scallop	0.929
4	breakfast_burrito	burrito	0.939
5	nachos	nacho	0.9517
6	dumplings	dumpling	0.9536
7	mussels	mussel	0.955
8	churros	churro	0.9585
9	spring_rolls	spring_roll	0.9585
10	chicken_wings	chicken_wing	0.9604
11	escargots	escargot	0.962
12	waffles	waffle	0.966
13	baby_back_ribs	baby_back_rib	0.966
14	oysters	oyster	0.9663
15	beignets	beignet	0.97
16	tacos	taco	0.9727
17	donuts	donut	0.9756
18	crab_cakes	crab_cake	0.9756
19	deviled_eggs	deviled_egg	0.976
20	macarons	macaron	0.9775
21	pancakes	pancake	0.982
22	pad_thai	pad_thai	0.999
23	grilled_salmon	grilled_salmon	0.999
24	fried_calamari	fried_calamari	0.999
25	omelette	omelette	0.9995
26	beef_carpaccio	beef_carpaccio	0.9995
27	hamburger	hamburger	0.9995
28	clam_chowder	clam_chowder	0.9995
29	chocolate_cake	chocolate_cake	0.9995
30	lobster_roll_sandwich	lobster_roll_sandwich	0.9995
31	macaroni_and_cheese	macaroni_and_cheese	0.9995
32	seaweed_salad	seaweed_salad	0.9995
33	shrimp_and_grits	shrimp_and_grits	0.9995
34	sushi	sushi	0.9995
35	creme_brulee	creme_brulee	0.9995
36	sashimi	sashimi	0.9995
37	cheesecake	cheesecake	0.9995
38	chicken_curry	chicken_curry	0.9995
39	fried_rice	fried_rice	0.9995
40	pork_chop	pork_chop	0.9995
41	bruschetta	bruschetta	0.9995
42	edamame	edamame	0.9995
43	cannoli	cannoli	0.9995
44	caprese_salad	caprese_salad	0.9995
45	red_velvet_cake	red_velvet_cake	0.9995
46	spaghetti_bolognese	spaghetti_bolognese	1
47	spaghetti_carbonara	spaghetti_carbonara	1
48	takoyaki	takoyaki	1
49	tiramisu	tiramisu	1
50	tuna_tartare	tuna_tartare	1

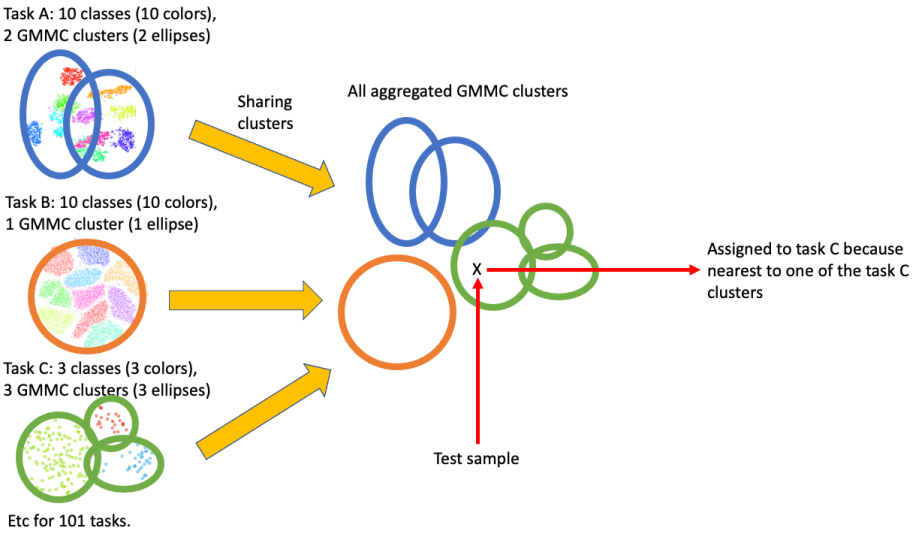


Figure S14. GMMC task mapper. (left) Each teacher clusters its entire training set into a number of Gaussian clusters. Here, a variable number of clusters is shown for each task, but in our results we use 25 clusters for every task. Each teacher then shares the mean and diagonal covariance of its clusters with all students. (right) Students just aggregate all received clusters in a bank, keeping track of which task any given cluster comes from. At test time, a sample is evaluated against all clusters received so far, and the task associated with the cluster closest to the test sample is chosen.

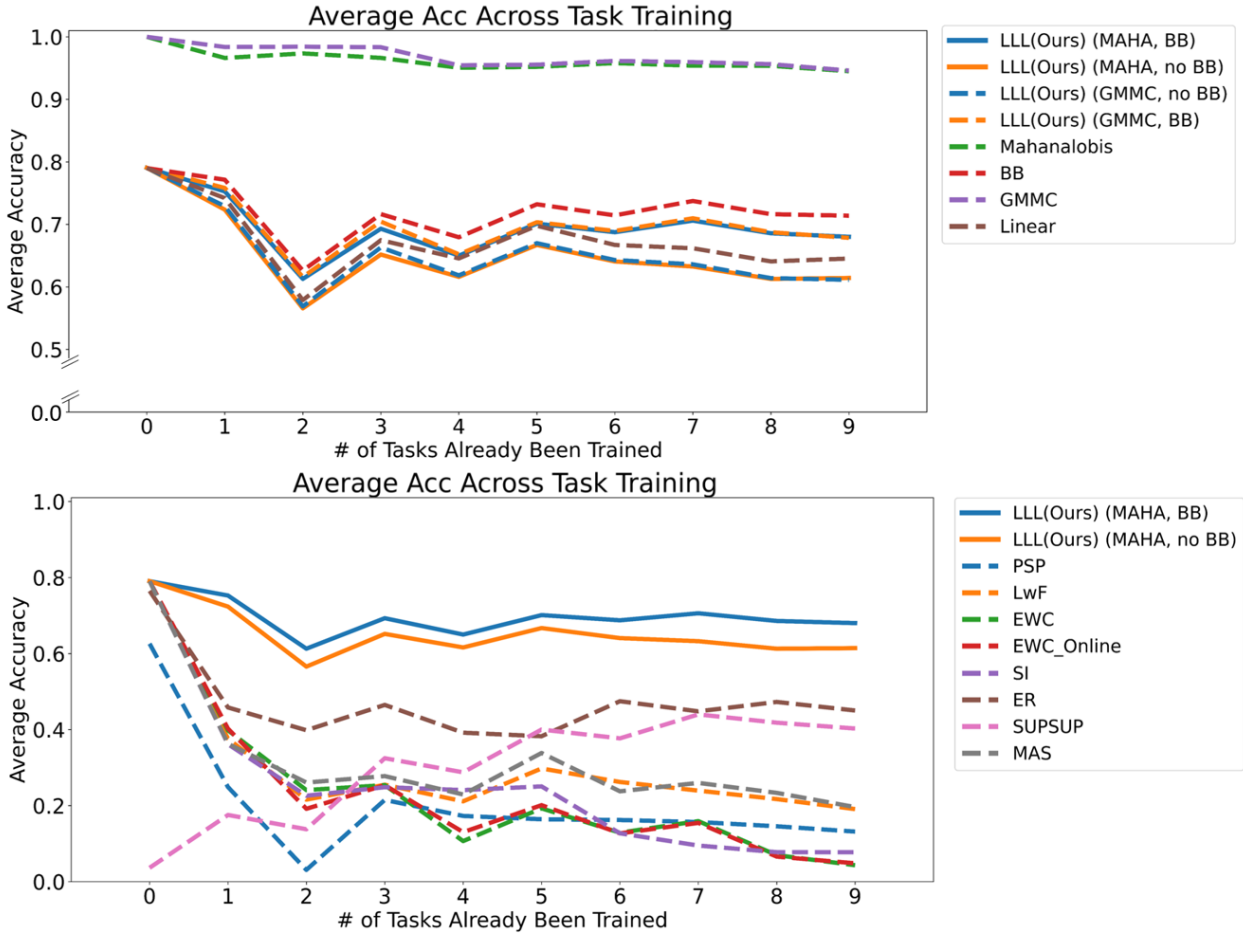


Figure S15. Average absolute accuracy on 10 Visual Domain Decathlon tasks learned so far, as a function of the number of tasks learned.