Appendices to the ICML 2021 paper

Sharing Less is More: Lifelong Learning in Deep Networks with Selective Layer Transfer

by Seungwon Lee, Sima Behpour, and Eric Eaton

A. Experiment Details

This section provides detail on the experiments from the main paper. The experiments are based on three image recognition datasets: CIFAR-100 (Krizhevsky & Hinton, 2009), Office-Home (Venkateswara et al., 2017), and STL-10 (Coates et al., 2011).

CIFAR-100 consists of images of 100 classes. The lifelong learning tasks are created following previous work (Lee et al., 2019) by separating the dataset into ten disjoint sets of ten classes, and randomly selecting 4% of the original training data to generate training and validation sets in the ratio of 5.6:1 (170 training and 30 validation instances per task). The images are used after normalization. For the CIFAR-100 experiment with 40 tasks, each image class is included in four different tasks ensuring that any pair of tasks does not have the same set of classes.

The Office-Home dataset has images of 65 classes in four domains. Again following Lee et al. (2019), lifelong learning tasks are generated by choosing ten disjoint groups of thirteen classes in two domains: Product and Real-World. There is no pre-defined training/testing split in Office-Home, so we randomly split the images in the ratio 6:1:3 for the training, validation, and test sets. The image sizes are not uniform, so we resized all images to be 128-by-128 pixels and re-scaled each pixel value to the range of $[0, 1]$.

We introduce a lifelong learning variant of the STL-10 dataset, which contains ten classes. We constructed 20 three-way classification tasks by randomly choosing the classes, applying Gaussian noise to the images (with a mean and variance randomly sampled from $\{-10\%, -5\%, 0\%, 5\%, 10\%\}$ of the range of pixel values) after re-scaling each pixel value to the range of $[-0.5, 0.5]$, and randomly swapping channels. Note that any pair of tasks differs by at least one image class, the mean and variance of the Gaussian noise, or the order of channels for the swap. We sampled 25% of the given training data and split it into training and validation sets with the ratio 5.7:1 (318 training and 57 validation instances per task). All of the original STL-10 test data are used for held-out evaluation of performance.

The architectural details of the task models used for each data set are described in Figure 7. We used the following values for the hyper-parameters of the algorithms, following the original papers wherever possible:

- The multi-task CNN with hard parameter sharing (HPS) has no additional hyper-parameters.
- Tensor factorization has a scale of the weight orthogonality constraint, whose value was chosen by grid search among $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ following the original paper (Bulat et al., 2020).
- DF-CNN requires the size of the shared tensors and the parameters of the task-specific mappings to be specified. Following the original paper (Lee et al., 2019), we chose the spatial size of the shared tensors to be half the spatial size of the convolutional filters, and the spatial size of the deconvolutional filters as 3×3. For each convolutional layer with input channels c_{in} and output channels c_{out}, the number of channels in the shared tensors was one-third of $c_{in} + c_{out}$ and the number of output channels of the deconvolutional filters was two-thirds of $c_{in} + c_{out}$.
- DEN has several regularization terms and the size of the dynamic expansion. We used the regularization values in the authors' published code, and set the size of the dynamic expansion to be 32 by choosing the most favorable value among $\{8, 16, 32, 64\}$. APD-Net has two regularization terms for the sparsity of additive parameters λ_1 and catastrophic forgetting λ_2. As described in the original paper (Yoon et al., 2020), we used $4e^{-4}$ and 100 as the value of λ_1 and λ_2, respectively.
- ProgNN requires the compression ratio of the lateral connections, which we set to be 2, following the original paper (Rusu et al., 2016).
- For DARTS, we used the hyper-parameter settings described in the original paper (Liu et al., 2019a).

A lifelong learner has access to the training data of only the current task, and it optimizes the parameters of the current task model as well as any shared knowledge, depending on the algorithm. After the pre-determined number of training epochs, the task switches to a new one regardless of the convergence of the lifelong learner, which favors learners that can rapidly adapt to each task. When the learner encounters a new task, it initializes newly introduced parameters of the new task model, but re-uses the parameters of shared components, which initialize only once at the beginning of the first task. As mentioned earlier, these new task-specific parameters and shared parameters are optimized according to the training data of the new task for another batch of training epochs. We used the RMSProp optimizer with the hyper-parameter values (such as learning rate and the number of training epochs per task) described in Table 4. In our experiments, we conservatively have LASEM take a single M-step per E-step by setting $numMSteps = 1$.

Dataset	CIFAR-100	Office-Home	STL-10
Number of Tasks T	10 or 40	10	15 or 20
Type of Task	Heterogeneous Classification		Semi-heterogeneous Classification
Classes per Task	10	10	5 ($T = 15$) or 3 ($T = 20$)
Amount of Training Data	4%	-	10% or 25%
Ratio of Training and Validation Set	5.6:1	6:1	1:1 or 5.7:1
Size of Image	32×32	128×128	96×96
Optimizer	RMS Prop		
Learning Rate	1×10^{-4}	2×10^{-5}	1×10^{-4}
Epoch per Task	2000 or 200 (ResNet-18)	1000	500
Ratio of M-steps to E-step ($numMSteps$)	1		

Table 4: Parameters of the lifelong learning experiments. We used the notion of task type (e.g., heterogeneous) introduced in previous work (Yang & Hospedales, 2017), which is based on the distribution similarity of the tasks' data.

B. LASEM-Discovered Transfer Configurations

LASEM dynamically searches through the space of transfer configurations during the learning process, as illustrated in Figure 3. At initialization, the probability of all transfer configurations is uniform, which dynamically changes during training alongside the layers for each task model, until the transfer configuration finalizes at convergence.

Figure 8 shows the most frequently learned transfer configurations as well as the proportion of the time each layer was chosen to be transfer-based or task-specific. For CIFAR-100 and Office-Home, there is tendency of transferring top layers more than bottom layers. However, an interesting observation is that non-tree structures, such as Alternating $\{2, 4\}$ and sharing middle layers [0,1,1,0], are often chosen. This contradicts the assumption of a tree structure made often by related research, and supports the consideration of more complex transfer configurations for diverse tasks.

The top eight most-chosen configurations of STL-10, unlike the other datasets which employed deep nets with fewer CNN layers, plateau with a peak less than 10%. This is likely due to the smaller number of STL-10 tasks, more flexible (deeper) network, and a much larger number of possible transfer configurations than in the other two experiments. The tensor factorization model for STL-10 seems to prefer transfer at higher layers over transfer at lower layers, while the preference for DF-CNN is more varied.

C. Avoiding Catastrophic Forgetting

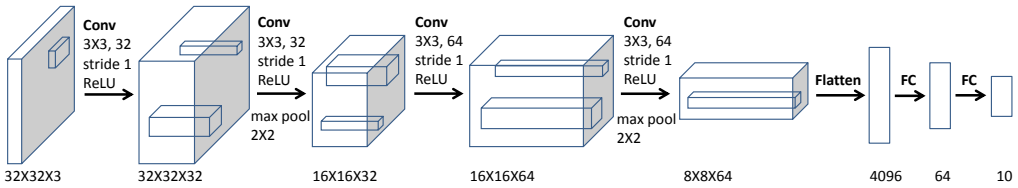
We investigated the ability of LASEM to avoid catastrophic forgetting in addition to the mean peak per-task accuracy. The catastrophic forgetting ratio is shown in Figure 9. The catastrophic forgetting ratio (Lee et al., 2019) measures the ability of the lifelong learning algorithm to maintain its performance on previous tasks during subsequent learning. A low ratio indicates that there is negative reverse transfer from new tasks to previously learned tasks, and so the learner experiences catastrophic forgetting. A ratio greater than 1 can be interpreted as positive backward transfer. As depicted in Figure 9, LASEM is able to retain the performance of previous tasks compared to transferring at all CNN layers and transferring at specific CNN-layers for all tasks (using a static transfer configuration).

D. LASEM for Transfer Over Groups of Layers

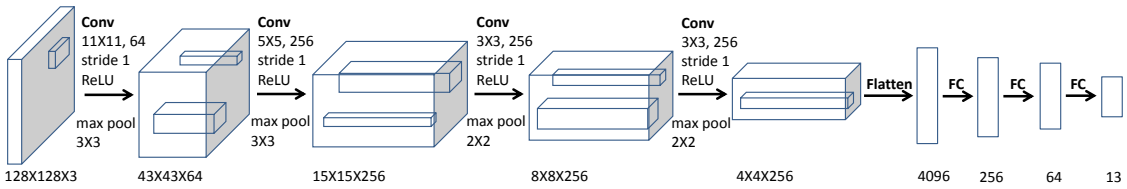
As discussed in Section 3, it is a well-known problem in neural architecture search that the search over layer-based transfer configurations requires time exponential to depth of the network d. One common technique to compensate for this problem, as used by other methods (Pham et al., 2018; Liu et al., 2019a), is to search over groups of layers instead of individual layers, thereby reducing the size of the search space. LASEM easily supports this same technique by redefining the transfer configuration space $\mathcal{C} = \{0, 1\}^d$ to be binary indicators over a partition $\mathcal{P}$ of the set of layer indices $\{1, \ldots, d\}$, where the cardinality $|\mathcal{P}| \ll d$. Consequently, this reduces the search space from 2^d to $2^{|\mathcal{P}|}$. Most naturally, the partition $\mathcal{P}$ should ensure that either adjacent layers (e.g., $\{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$) or nearby[1] layers (e.g. $\{\{1, 3\}, \{2, 4\}, \{5\}, \{6\}\}$) are grouped together.

First, we evaluated this variation of LASEM in lifelong learning scenarios using the STL-10 data set. Different from the aforementioned experiments using STL-10, this experiment consisted of 15 five-way classification tasks by

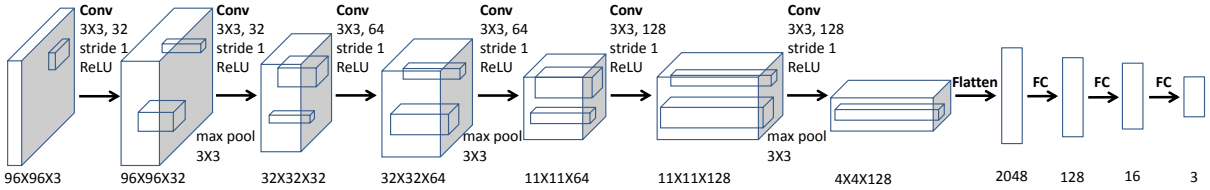
[1]This pattern of grouping nearby layers together, instead of only adjacent, may allow more flexible adaptation of transferred knowledge to individual tasks, similar to the Alternating configuration explored in Section 2.



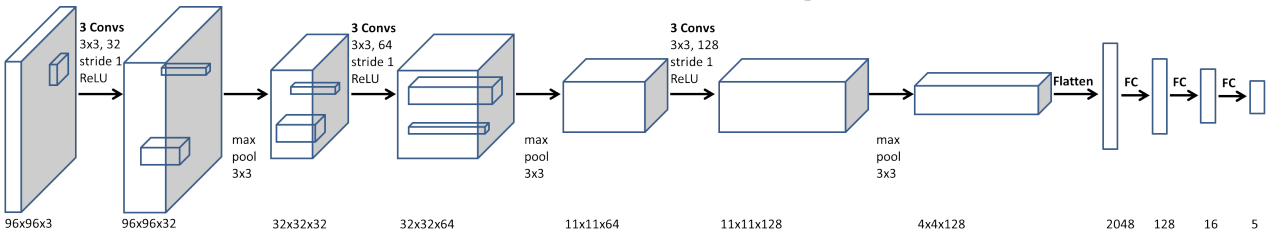
(a) Architecture of task models of CIFAR-100 experiment



(b) Architecture of task models of Office-Home experiment



(c) Architecture of task models of STL-10 experiment



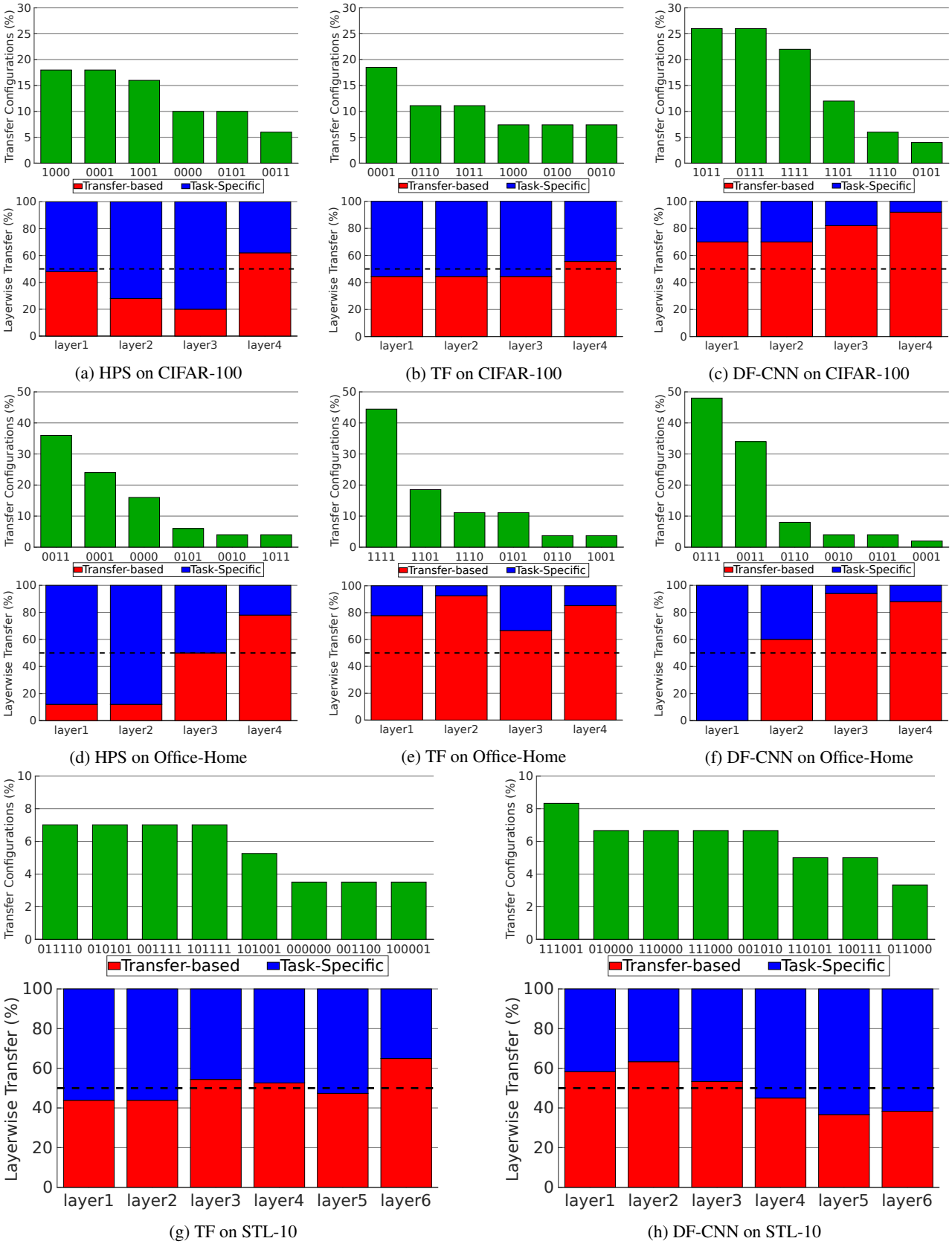
(d) Architecture of task models of STL-10 experiment for the group-based LASEM

Figure 7: Details of the task model architectures used in the experiments. Text by each convolutional layer describes the filter sizes and the number of channels. All convolutional layers are zero-padded.

random selection of the classes. We sampled only 10% of the given training data and split it into training and validation sets with the ratio 1:1. For this scenario, we trained a DF-CNN transferring at all layers and layer-based LASEM on a DF-CNN with 9 convolutional layers; details are in Figure 7. The group-based LASEM used the partition $\{\{1,2,3\},\{4,5,6\},\{7,8,9\}\}$, splitting nine convolutional layers into three groups of three adjacent layers.

The DF-CNN achieved a mean task-wise best accuracy of 45.4 ± 0.4 with a training time of 7.58×10^4 seconds. The original layer-based LASEM DF-CNN exceeded the capacity of our computing source (an Intel core i7 workstation with dual 1080 Ti GPUs). However, the group-based LASEM DF-CNN achieved a mean accuracy of 46.0 ± 0.7 % in 7.02×10^4 seconds. This showcases LASEM's ability to support group-based transfer configurations in addition to layer-based configurations.

Additionally, we evaluated the group-based LASEM using the ResNet-18 architecture (He et al., 2016) in two lifelong learning scenarios using the CIFAR-100 data set: one with 10 ten-way classification tasks and a longer scenario with 40 ten-way classification tasks. To generate 40 tasks of ten-way classification, each image class is used within four classification tasks, and each task has a unique set of classes. For the aforementioned experiments using the CIFAR-100, each task is trained for 2,000 epochs, but for the ResNet-18 eval-

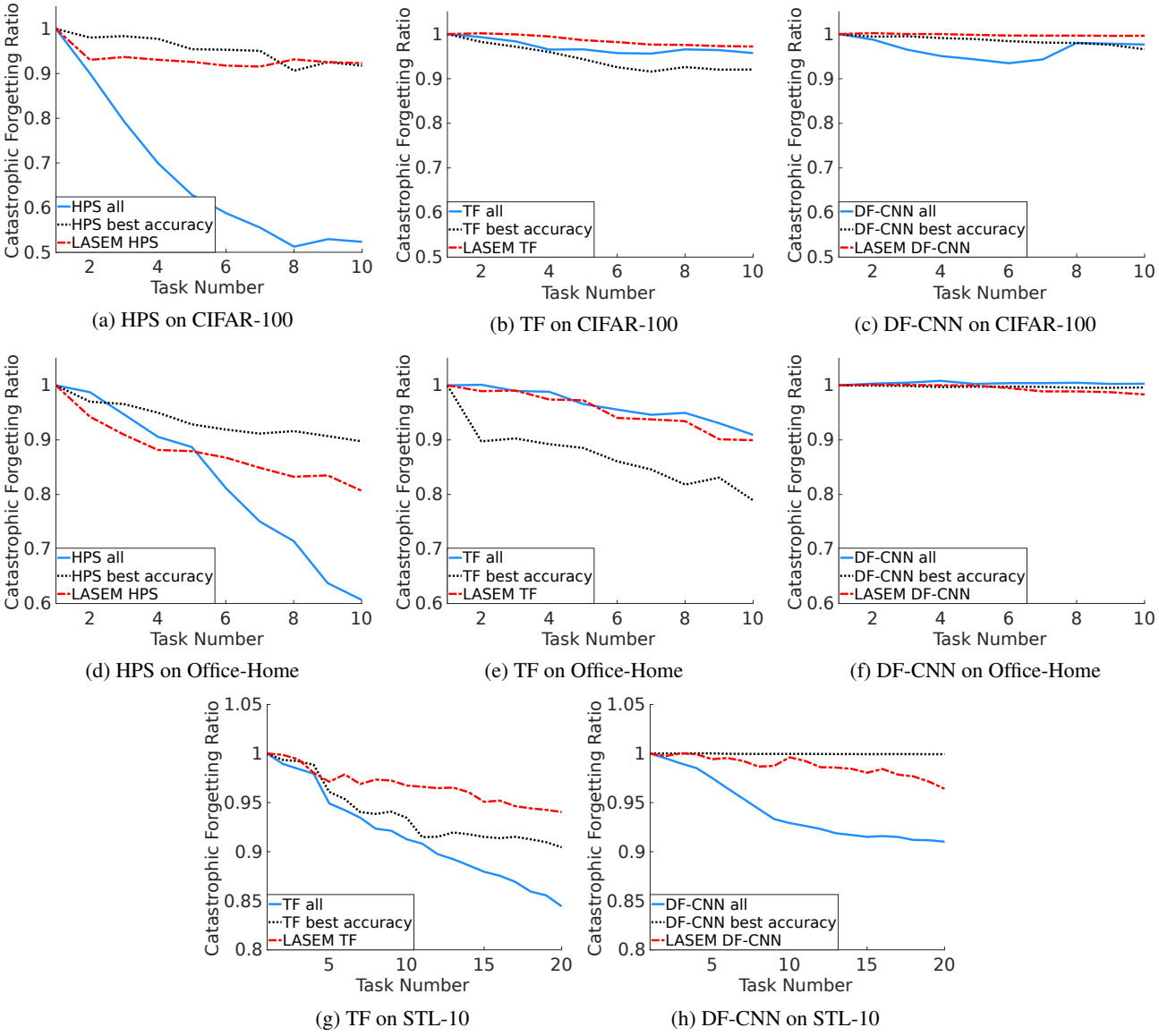


(a) HPS on CIFAR-100 (b) TF on CIFAR-100 (c) DF-CNN on CIFAR-100

(d) HPS on Office-Home (e) TF on Office-Home (f) DF-CNN on Office-Home

(g) TF on STL-10 (h) DF-CNN on STL-10

Figure 8: (Top) Histogram of the most-selected configurations (i.e., the binary vectors $\boldsymbol{c}_t$, where 1 denotes that a CNN layer employs transfer). (Bottom) The fraction of time each layer was selected to be transfer-based (red) or task-specific (blue).



(a) HPS on CIFAR-100 (b) TF on CIFAR-100 (c) DF-CNN on CIFAR-100

(d) HPS on Office-Home (e) TF on Office-Home (f) DF-CNN on Office-Home

(g) TF on STL-10 (h) DF-CNN on STL-10

Figure 9: Catastrophic forgetting ratio of transfer at all CNN layers (blue), best static transfer configuration (black) and LASEM (red), exhibiting the benefit of LASEM. Note that the y-axis range differs for each data set.

Number of Groups	Partition
4 Groups	$\{\{1,2,3,4,5\},\{6,7,8,9\},\{10,11,12,13\},\{14,15,16,17\}\}$
5 Groups	$\{\{1,2,3,4,5\},\{6,7,8,9\},\{10,11,12,13\},\{14,15\},\{16,17\}\}$
6 Groups	$\{\{1,2,3,4,5\},\{6,7,8,9\},\{10,11\},\{12,13\},\{14,15\},\{16,17\}\}$
7 Groups	$\{\{1,2,3,4,5\},\{6,7\},\{8,9\},\{10,11\},\{12,13\},\{14,15\},\{16,17\}\}$

Table 5: Partitions for the experiments on the group-based LASEM on ResNet-18.

uation, each task is trained only for 200 epochs due to the convergence of performance. All experimental settings, except the number of tasks and the number of training epochs, remain the same as the settings described in Appendix A. For these experiments, we trained a ResNet-18 HPS transferring all convolutional layers and the group-based LASEM HPS using the four partitions shown in Table 5. During training, the larger ResNet-18 yielded a few inaccurate predictions with high confidence, which caused the computed likelihood $P(\boldsymbol{c} \mid X_{new}, y_{new})$ (Equation 2) of the trans-

fer configuration for the entire minibatch to become 0. To compensate, we treated the minibatch as a collection of independent instances from which to estimate the likelihood of the transfer configuration, and so took $P(\boldsymbol{c} \mid X_{new}, y_{new})$ as the mean of these individual estimates. This eliminated the effect of the high-confidence inaccurate predictions driving the product to 0, yielding reasonable estimates of the configuration likelihood. The ResNet-18 results we report use this method for the likelihood computation in LASEM.

As described in Section 4.4 and Table 2, the group-based LASEM ResNet-18 HPS outperforms ResNet-18 HPS in both the peak per-task accuracy and catastrophic forgetting ratio. The lifelong learner loses performance of earlier tasks as it learns more tasks due to the limitation of HPS, but LASEM reduces the amount of forgetting because it determines the layers to share according to the similarity between tasks as discovered from data. The comparison of HPS on ResNet-18 with and without LASEM shows that it is capable of scaling to deeper architectures and larger numbers of tasks.

E. Memory Usage Comparison

In this section, we analyze LASEM's memory requirements to show that it is approximately equivalent to the other methods considered in the paper. Let the base learner require $O(A)$ non-transfer-based task-specific storage or $O(B)$ transfer-based task-specific storage with $O(S)$ shared knowledge. LASEM shares network parameters across transfer configurations to minimize memory, so the *current* task model stores two parameter sets at a cost of $O(A+B)$. Earlier task models require only $O(\max(A, B))$ storage, yielding a total memory requirement of $O(S + (T + 1)\max(A, B))$ for T tasks when the base learner constructs one network per task. Compared to this, the model with the best static transfer configuration requires $O(\max(A, B))$ additional storage per task, resulting in a total memory requirement of $O(S + T\max(A, B))$. Brute-force search over transfer configurations requires at least $O(2\max(A, B))$, one for parameters of the best configuration and the other for parameters of the current training, yielding $O(S + (T + 1)\max(A, B))$. Hence, LASEM's memory requirements are approximately equivalent to the alternative methods. The memory requirement may differ from the above analysis according to the base lifelong learning architecture used (such as HPS or a modular network), but if that base learner requires $O(T)$ memory for T tasks, LASEM needs only $O(T + 1)$ memory.