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

Anonymous Authors[1]

Abstract

Effective lifelong learning across diverse tasks requires diverse knowledge, yet transferring irrelevant knowledge may lead to interference and catastrophic forgetting. In deep networks, transferring the appropriate granularity of knowledge is as important as the transfer mechanism, and must be driven by the relationships among tasks. We first show that the lifelong learning performance of several current deep learning architectures can be significantly improved by transfer at the appropriate layers. We then develop an expectation-maximization (EM) method to automatically select the appropriate transfer configuration and optimize the task network weights. This EM-based selective transfer is highly effective, as demonstrated on three algorithms in several lifelong object classification scenarios.

1. Introduction

Transfer at different layers within a deep network corresponds to sharing knowledge between tasks at different levels of abstraction. In multi-task scenarios that involve diverse tasks, reusing low-layer representations may be appropriate for tasks that share feature-based similarities, while sharing high-level representations may be more appropriate for tasks that share more abstract similarities. Selecting the appropriate granularity of knowledge to transfer is an important architectural consideration for deep networks that support multiple tasks.

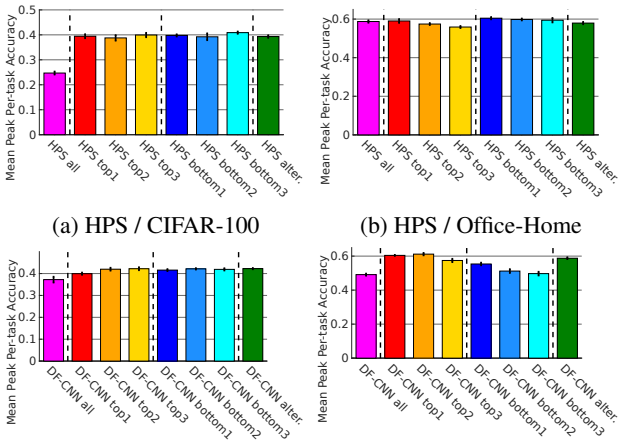
In scenarios where tasks share substantial similarities, many multi-task methods have found success using a universal configuration of the knowledge sharing (Caruana, 1993; Yang and Hospedales, 2017; Lee et al., 2019; Liu et al., 2019; Bulat et al., 2020), such as sharing the lower layers of a deep network with upper-level task-specific heads. As tasks become increasingly diverse, the appropriate granularity for transfer may vary between tasks based on their relationships, necessitating more selective transfer. Prior work in selective sharing for deep networks has typically either (1) branched the network into a tree structure (Lu et al., 2017; Yoon et al., 2018; Vandenhende et al., 2019; He et al., 2018), which emphasizes the sharing of lower layers or (2) introduced new learning modules between task models (Yang and Hospedales, 2017; Xiao et al., 2018; Cao et al., 2018; Rusu et al., 2016) which increases the complexity of training. The transfer configuration could then be optimized in batch settings to maximize performance across the tasks.

However, the problem of selective transfer is further compounded in continual or lifelong learning settings, in which tasks are presented sequentially. The optimal transfer configuration may vary between tasks or over time. To verify this premise and motivate our work, we conducted a simple experiment: we took a multi-task CNN with shared layers and a lifelong learning CNN that uses factorized transfer (DF-CNN (Lee et al., 2019)) and varied the set of CNN layers that employed transfer (with task-specific fully connected layers at the top). Using two data sets, we considered transferring at all CNN layers, transfer at the top-k CNN layers, transfer at the bottom-k CNN layers, and alternating transfer/no-transfer CNN layers. The results are shown in Figure 1, with details given in Section 2. Clearly, we see that the optimal transfer configuration varies between task relationships and transfer mechanisms. Restricting the transfer layers significantly improves performance over the naïve approach of transferring at all layers, with the alternating configuration performing extremely well in both cases.

To enable a more flexible version of selective transfer, we investigate the use of architecture search to dynamically adjust the transfer configuration between tasks and over time. We use expectation-maximization (EM) to learn both the parameters of the task models and the layers to transfer within the deep net. This approach, Lifelong Architecture Search via EM (LASEM), enables deep networks to transfer different sets of layers for each task, allowing more flexibility over branching-based configurations for selective transfer.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.



(a) HPS / CIFAR-100 (b) HPS / Office-Home

(c) DF-CNN / CIFAR-100 (d) DF-CNN / Office-Home

Figure 1: Accuracy of CNN models averaged over ten tasks in a lifelong learning setting with 95% confidence interval. This empirically shows that the optimal transfer configuration varies, and choosing the correct configuration is superior to transfer at all layers.

It also introduces little additional computation in comparison to exhaustive search over all transfer configurations or training selective transfer modules between task networks. To demonstrate its effectiveness, we applied LASEM to three architectures that support knowledge transfer between tasks in several lifelong learning scenarios and compared it against other lifelong learning and architecture search methods.

2. The effect of different transfer configurations in lifelong learning

This section further describes the experiments mentioned in the introduction as motivation for our proposed LASEM method. The hypothesis of our work is that lifelong deep learning can benefit from using a more flexible transfer mechanism that selectively chooses the transfer architecture for each task. This would permit it to dynamically select, for each task model, which layers to transfer and which to keep as task-specific (enabling it to customize transferred knowledge to an individual task).

To determine the effect of different transfer configurations, we conducted a set of initial experiments using two established methods:

Multi-Task CNN with hard parameter sharing (HPS): This approach shares the hidden CNN layers between all tasks, and maintains task-specific fully connected output layers. It is one of the most common methods for multi-task learning of neural networks (Caruana, 1993; 1997), and is widely used.

Deconvolutional factorized CNN (DF-CNN): The DF-CNN (Lee et al., 2019) adapts CNNs to a continual learning setting by sharing layer-wise knowledge across tasks. The convolutional filters of each task model are dynamically generated from a task-*independent* layer-*dependent* shared tensor through a series of task-specific mapping. When training the task models consecutively, gradients flow through to update the shared tensors and the task-specific parameters, so this transfer architecture enables the DF-CNN to learn and compress knowledge universal among the observed tasks into the shared tensors.

Both these methods utilize a set of transfer-based CNN layers and non-transfer task-specific layers. For a network with d CNN layers, there are 2^d potential transfer configurations. To explore the effect of transfer at different layers, we varied the transfer configuration among several options:

- *All*: Transfer at all CNN layers. Note that the original DF-CNN used this configuration.
- *Top* k: Transfer across task models occurs only at the k highest CNN layers, with all others remaining task-specific. We would expect this transfer configuration to benefit tasks that share high-level concepts but have low-level feature differences.
- *Bottom* k: Transfer occurs only at the k lowest CNN layers, which is opposite of the *Top* $d-k$. We would expect it to benefit tasks that share perceptual similarities but have high-level differences.
- *Alternating*: This configuration alternates transfer and non-transfer layers, enabling the non-transfer task-specific layers to further customize the transferred knowledge to the task.

We evaluate the performance of various transfer configurations on the CIFAR-100 (Krizhevsky and Hinton, 2009) and Office-Home (Venkateswara et al., 2017) data sets, following the lifelong learning experimental setup used in previous work (Lee et al., 2019). CIFAR-100 involves ten consecutive tasks of ten-way image classification, where any object class occurs in only one task. Office-Home involves ten tasks of thirteen-way classification, separated into two domains: 'Product' images and 'Real World' images. The CNN architectures used for each data set and optimization settings follow prior work (Lee et al., 2019). During training, we measured the peak per-task accuracy on held-out test data, averaging over five trials.

Our results, shown in Figure 1, reveal that permitting transfer at all layers does not guarantee the best performance. This observation complicates learning on novel tasks, since the best transfer configuration depends both on the algorithm and the task relations in the data set. Notably, we see that

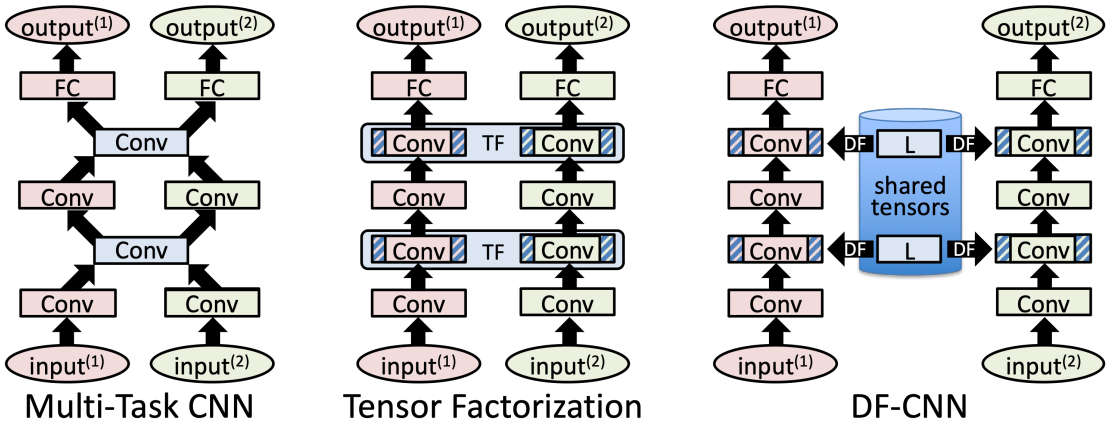


Figure 2: The Alternating $\{2, 4\}$ transfer configuration for three different architectures with four convolutional layers and one task-specific fully-connected layer. Models are illustrated for two tasks, red and green, with transfer-based layers denoted in blue.

the DF-CNN, which is designed for lifelong learning, can be improved beyond the original version (Lee et al., 2019) by allowing transfer at only some layers. Furthermore, we can see that the optimal transfer configuration varies between data sets and algorithms. For instance on Office-Home, sharing lower layers in the HPS multi-task CNN achieves better performance on average, but transferring upper layers works better for the DF-CNN. Similarly, the Alternating configuration consistently achieves near the best performance for the DF-CNN, benefiting from permitting the non-transfer layers to customize transferred knowledge to the individual task, but it is not consistently as good for HPS.

3. Architecture search for the optimal transfer configuration

The experiment presented above reveals that the transfer configuration can have a significant effect on lifelong learning performance, and that the best transfer configuration varies. These observations inspire our work to develop a more flexible mechanism for selective transfer in lifelong learning by viewing the transfer configuration as a new hyper-parameter for each task model. The search space grows exponentially as the neural network gets deeper (i.e., 2^d configurations for d CNN layers) and linearly as the more tasks are learned.

Formally, a layer-based transfer configuration for task t can be specified by a d-dimensional binary vector $\boldsymbol{c}_t \in \mathcal{C} = \{0, 1\}^d$, where each $c_{t,j}$ is a binary indicator whether or not the jth layer involves transfer. We can compactly notate $\boldsymbol{c}_t$ by a set of its indices containing True entries. For example, the Alternating configuration described in Section 2 can be denoted by $\boldsymbol{c} = [0, 1, 0, 1] = \{2, 4\}$ (refer to Figure 2).

Our goal is to optimize the task-specific transfer configuration while simultaneously optimizing the parameters of the task models and shared knowledge in a lifelong setting. Treating $\boldsymbol{c}_t$ as a latent variable of the model for task t, we employ expectation-maximization (EM) to perform this joint optimization. For each new task, LASEM maintains a set of transfer-based parameters $\boldsymbol{\theta}_s^{(l)}$ and task-specific parameters $\boldsymbol{\theta}_t^{(l)}$ for each layer l, using the chosen configuration $\boldsymbol{c}_t$ to determine which combination of parameters will be used to form the specific model. In brief, the E-step estimates the usefulness of the representation that each transfer configuration $\boldsymbol{c}_i \in \mathcal{C}$ can learn from the given data (i.e., the likelihood of data), while the M-step optimizes parameters of the task model and shared knowledge.

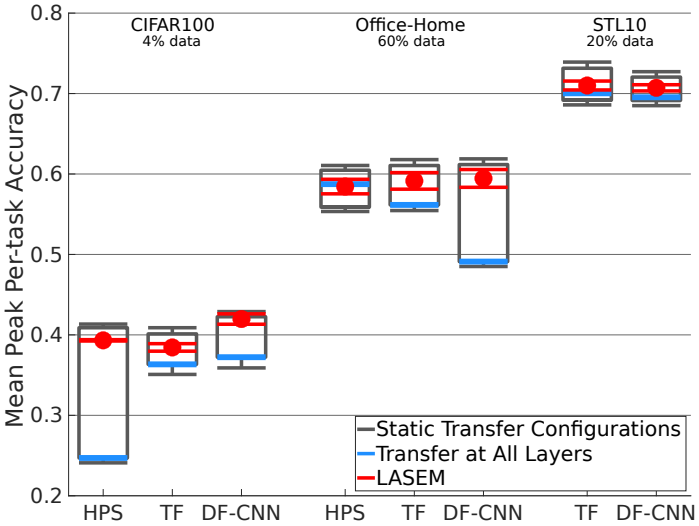
We first consider how to model the prior π_t on possible configurations of the current task's $\boldsymbol{c}_t$. Using a simple frequency-based probability estimate with Laplace smoothing represents the prior probability of each configuration as

$$P(\boldsymbol{c}_{(t)} = \boldsymbol{c}_i) = \pi_t(\boldsymbol{c}_i) = \frac{n_{\boldsymbol{c}_i} + 1}{\sum_i (n_{\boldsymbol{c}_i} + 1)} \ , \qquad (1)$$

where $\boldsymbol{c}_{(t)}$ denotes the configuration for task t, and $n_{\boldsymbol{c}_i}$ is the number of mini-batches whose most probable configuration is $\boldsymbol{c}_i$. This estimate considers each transfer configuration solely based on the current task's data, but alternative priors could instead be used, such as measuring the most frequent transfer configuration historically over all tasks or measuring the most frequent configuration over related tasks (which requires a notion of task similarity, such as via a task descriptor (Isele et al., 2016; Sinapov et al., 2015)).

In the E-step, the posterior on configurations is derived by combining the above prior and likelihood, which can be computed from the output of the task network on the current task's data $\mathcal{D}_{new} := (X_{new}, y_{new})$:

$$P(\boldsymbol{c}_i | \mathcal{D}_{new}) \propto \pi_t(\boldsymbol{c}_i) P(y_{new} | X_{new}, \boldsymbol{c}_i) \ . \qquad (2)$$



Architecture	Acc. (%)	Rel. Acc.
CIFAR-100 (10 Tasks)		
HPS	39.3 ± 0.1	96.2
TF	38.4 ± 0.5	95.8
DF-CNN	42.0 ± 0.6	99.3
Office-Home (10 Tasks)		
HPS	58.4 ± 0.9	96.6
TF	59.1 ± 1.0	96.9
DF-CNN	59.5 ± 1.1	97.2
STL-10 (20 Tasks)		
TF	71.0 ± 0.6	97.1
DF-CNN	70.7 ± 0.4	98.2

Figure 3: (Left) Lifelong learning performance of LASEM applied to three methods and three scenarios. Black boxes show the range of mean accuracies that different static configurations can achieve, with the blue lines denoting mean performance of employing transfer at all layers. The red dots denote the mean performance of LASEM. The whiskers depict 95% confidence intervals. (Right) Accuracy and relative accuracy of LASEM with respect to the best static transfer configuration.

The M-step improves the log-likelihood via the estimated probability distribution over the transfer configurations. Both $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_t$ are updated by the aggregated gradients of the log-likelihood in cases where the transfer configurations match the corresponding parameter. To combine the gradients of a specific parameter tensor over multiple possible configurations, we take the sum of the corresponding gradients weighted by the above posterior estimate.

$$\begin{aligned} \boldsymbol{\theta}_s^{(l)\prime} &\leftarrow \boldsymbol{\theta}_s^{(l)} + \lambda \textstyle\sum_{i:c_{i,l}=1} P(\boldsymbol{c}_i|\mathcal{D}_{new}) \nabla \log \mathcal{L}(\mathcal{D}_{new}|\boldsymbol{c}_i) \\ \boldsymbol{\theta}_t^{(l)\prime} &\leftarrow \boldsymbol{\theta}_t^{(l)} + \lambda \textstyle\sum_{i:c_{i,l}=0} P(\boldsymbol{c}_i|\mathcal{D}_{new}) \nabla \log \mathcal{L}(\mathcal{D}_{new}|\boldsymbol{c}_i) \end{aligned} \tag{3}$$

for $l \in \{1, \cdots, d\}$. Here, $\mathcal{L}(\mathcal{D}_{new}|\boldsymbol{c}_i)$ is the likelihood of data given configuration $\boldsymbol{c}_i$: $P(y_{new}|X_{new}, \boldsymbol{c}_i)$. The main difference in the update rules in Equation 3 is the condition for the index of the summation.

Our LASEM follows lifelong learning framework in which the learner encounters tasks sequentially. The parameters of the transfer-based components are initialized once at the beginning of the learning process, while the parameters of the task-specific components and prior probability of configurations are initialized every new task. LASEM iterates E- and M-steps on data of the current task until task switches, and the best transfer configuration for the task is determined by the learned posterior probability. To reduce the complexity of computation, LASEM uses one set of transfer-based and task-specific parameters ($\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_t$) for all transfer configurations, rather than maintaining distinct sets of parameters for each configuration.

4. Experiments

We evaluated Lifelong Architecture Search via EM (LASEM) following the same experimental protocol for lifelong learning as used in Section 2. In addition to using CIFAR100 and Office-Home, we introduce a lifelong learning version of the STL-10 data set (Coates et al., 2011). STL-10 has 5,000 training and 8,000 test images divided evenly among 10 classes, with higher resolution than CIFAR-100. We constructed 20 tasks of three-way classification using 20% and 5% of the given training data for training and validation, respectively. To increase the task variations, for each task we randomly chose three images classes, applied Gaussian noise to the images with a random mean and variance, and randomly permuted the channels. All results were averaged over five trials with random task orders. Details of these three experiments as well as architectures of neural networks can be found in Appendix A.

4.1. Performance of LASEM

We applied LASEM to three lifelong learning architectures: a multi-task CNN with hard-parameter sharing (HPS) (Caruana, 1997), Tensor Factorization (TF) (Yang and Hospedales, 2017; Bulat et al., 2020) and the Deconvolutional Factorized CNN (DF-CNN) (Lee et al., 2019). HPS interconnects CNNs in tree structures, with task models explicitly using the same parameters of all shared layers. In contrast, the TF and DF-CNN task models explicitly share only a fraction of tensors, and parameters of each task model are generated via transfer.

Selective Sharing	Accuracy(%)	Train. Time (k sec)
DEN	48.00 ± 0.60	55.9 ± 0.6
ProgNN	$\mathbf{60.03} \pm \mathbf{0.45}$	96.7 ± 0.0
DARTS HPS	45.64 ± 1.20	43.8 ± 0.0
DARTS DF-CNN	56.77 ± 0.49	33.2 ± 0.0
LASEM HPS	58.44 ± 0.90	70.2 ± 0.0
LASEM TF	59.14 ± 0.80	77.3 ± 0.1
LASEM DF-CNN	$\mathbf{59.45} \pm \mathbf{1.10}$	83.2 ± 0.2

Table 1: Comparison of test accuracy and training time for the same epochs between selective transfer methods and LASEM, $\pm$ standard errors. The best accuracies are in bold and indistinguishable at 95% confidence.

Figure 3 compares the performance of the task-specific transfer configurations discovered by LASEM (red) to using a single static transfer configuration(black boxes). These black boxes depict the performance range of the methods using various transfer configurations (i.e., All, Top k, Bottom k, Alternating) for all task models, with All shown in blue. To estimate this range, we tested eight (50%) and 16 (25%) of the possible static configurations for the four-CNN-layer (CIFAR-100 and Office-Home) and six-CNN-layer (STL-10) task models, respectively.

We can see that LASEM chose transfer configurations that perform toward the top of each range, especially on the DF-CNN designed for lifelong learning. LASEM clearly outperforms transfer at all layers. Automatically selecting the transfer configuration becomes even more beneficial for methods that have a wide range of performances for different static configurations. Moreover, LASEM imposes little additional cost of computation in order to determine the transfer configuration. In timing experiments, we found that, compared to training with a pre-determined static transfer configuration, LASEM requires only 30% additional wall-clock time to search over 16 configurations of a network with four CNN layers, and only double the time to search over 64 configurations of a network with six CNN layers. Brute force search over configurations requires 15$\times$ and 63$\times$ *additional* time per task, respectively, over the base learner.

4.2. Comparison to other selective transfer algorithms

We further compared the performance of LASEM against other methods that employ some notion of selective transfer, including the Dynamically Expandable Network (DEN) (Yoon et al., 2018), the Progressive Neural Net (ProgNN) (Rusu et al., 2016) and Differentiable Architecture Search (DARTS) (Liu et al., 2018), on Office-Home. DEN is a lifelong learning architecture that extends tree structure sharing of HPS by expanding, splitting, and selectively retraining the network to introduce both shared and task-specific parameters in each layer if required. ProgNN learns additional layer-wise lateral connections from earlier task models to the current task model, which support one directional transfer to avoid negative knowledge transfer (known as catastrophic forgetting). Both DEN and ProgNN can support complex transfer configurations due to their lack of constraints, such as no assumption of a tree-structured configuration. For example, a ProgNN with all zero-weighted lateral connections for a level creates a task-specific layer, and zero-weighted current task model connections creates a transfer-based layer. DARTS is the gradient-based framework for neural architecture search, determining both the most suitable operation of each layer and the best architecture of stacking these layers. To enable the selection of architecture learnable, DARTS introduces a soft selection of the operators, making each layer a weighted sum of operations.

Table 1 summarizes the performance of these methods and our approach. ProgNN and LASEM DF-CNN are statistically indistinguishable and perform better than the other methods. LASEM DF-CNN is $\sim$14% faster than ProgNN for learning ten tasks. This gap could become larger as more tasks are learned because time complexity of ProgNN is proportional to the square of the number of tasks while the complexity of DF-CNN is linear. DEN and DARTS have better training times, but fail to perform as well. Note that LASEM shows high accuracy regardless of the base lifelong learner (e.g., HPS, TF, or DF-CNN) while introducing relatively little additional time complexity ($\sim$30% over the base learner's time).

5. Conclusion

We have shown that the transfer configuration can have a significant impact on lifelong learning, and that the configuration can be dynamically selected during the lifelong learning process with minimal computational cost. The improvement gained by choosing the optimal transfer configuration significantly improves the performance of the DF-CNN and TF over the original method and previously published results. Although we focused on layer-based transfer, LASEM could easily be extended to support partial transfer within a layer by imposing within-layer partitions and redefining the transfer configuration space C to support those partitions. Discovering these partitions directly from data, or providing more flexible mechanisms for partial within-layer transfer may further improve performance over the layer-based transfer we explore in this paper.

References

A. Bulat, J. Kossaifi, G. Tzimiropoulos, and M. Pantic. Incremental multi-domain learning with network latent tensor factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

J. Cao, Y. Li, and Z. Zhang. Partially shared multi-task convolutional neural network with local constraint for face attribute learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4290–4299, 2018.

R. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the International Conference on Machine Learning*, pages 41–48, 1993.

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

A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

X. He, Z. Zhou, and L. Thiele. Multi-task zipping via layer-wise neuron sharing. In *Advances in Neural Information Processing Systems*, pages 6016–6026, 2018.

D. Isele, M. Rostami, and E. Eaton. Using task features for zero-shot knowledge transfer in lifelong learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1620–1626, 2016.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

S. Lee, J. Stokes, and E. Eaton. Learning shared knowledge for deep lifelong learning using deconvolutional networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 2837–2844, 2019.

H. Liu, K. Simonyan, and Y. Yang. DARTS: differentiable architecture search. *CoRR*, abs/1806.09055, 2018.

H. Liu, F. Sun, and B. Fang. Lifelong learning for heterogeneous multi-modal tasks. In *Proceedings of the International Conference on Robotics and Automation*, pages 6158–6164. IEEE, 2019.

Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

J. Sinapov, S. Narvekar, M. Leonetti, and P. Stone. Learning inter-task transferability in the absence of target task samples. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 725–733, 2015.

S. Vandenhende, B. D. Brabandere, and L. V. Gool. Branched multi-task networks: deciding what layers to share. *CoRR*, abs/1904.02920, 2019.

H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

L. Xiao, H. Zhang, W. Chen, Y. Wang, and Y. Jin. Learning what to share: leaky multi-task network for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2055–2065, 2018.

Y. Yang and T. Hospedales. Deep multi-task representation learning: a tensor factorisation approach. In *Proceedings of the International Conference on Learning Representations*, 2017.

J. Yoon, E. Yang, and S. Hwang. Lifelong learning with dynamically expandable networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

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 and 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 Lee et al. (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 without any pre-processing except re-scaling all pixel values to the range of $[0, 1]$.

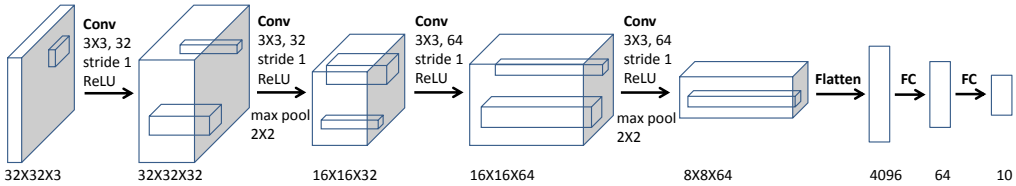
The Office-Home dataset has images of 65 classes in four domains. Again following Lee et al., lifelong learning tasks are generated by choosing ten disjoint groups of thirteen classes in two domains: Product and Real-World. There is no pre-defined training/testing split in Office-Home, so we randomly 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), 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.

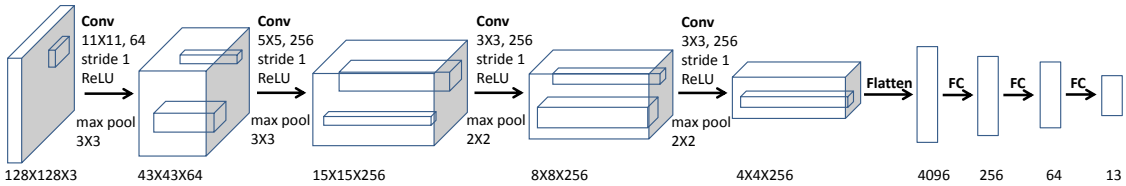
The architectural details of the task models used for each data set are described in Figure 4. We used the following values for the hyper-parameters of the algorithms, following the original papers wherever possible:

- 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 $\{8, 16, 32, 64\}$.
- 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., 2018).

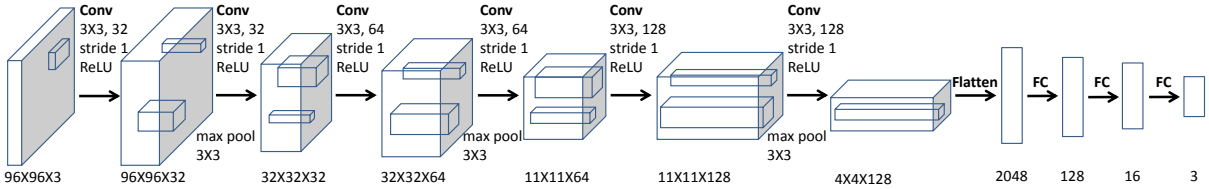
A lifelong learner has access to the training data of only the current task, and it optimizes the parameters of the current task model as well as any shared knowledge, depending on the algorithm. After the pre-determined number of training epochs, the task switches to a new one regardless of the convergence 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 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

Figure 4: 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.

Dataset	CIFAR-100	Office-Home	STL-10
Number of Tasks	10	10	20
Type of Task	Heterogeneous Classification		
Classes per Task	10	10	3
Amount of Training Data	4%	-	25%
Ratio of Training and Validation Set	5.6:1	6:1	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	1000	500

Table 2: Parameters of the lifelong learning experiments