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

Seungwon Lee [1] **Sima Behpour** [2] **Eric Eaton** [1]

Abstract

Effective lifelong learning across diverse tasks requires the transfer of 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, balancing transfer performance on all tasks with avoiding catastrophic forgetting, 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.

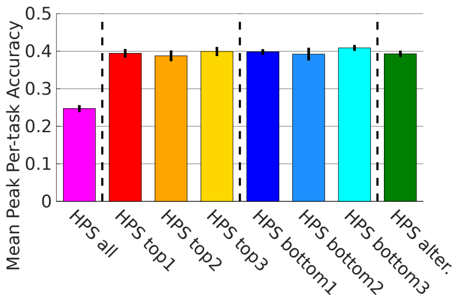
In scenarios where tasks share substantial similarities, many multi-task methods have found success using a static configuration of the knowledge to share (Caruana, 1997; Yang & Hospedales, 2017; Lee et al., 2019; Liu et al., 2019b; Bulat et al., 2020), such as sharing the lower layers of a deep network 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 & Hospedales, 2017; Xiao et al., 2018; Cao et al., 2018) 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 consecutively. The optimal transfer configuration may vary between tasks or it may vary over time. And indeed, we may *not* want to transfer at every layer, as some task-specific layers may need to be interleaved with shared knowledge in order to customize that shared knowledge to individual tasks. To verify this premise and motivate our work, we conducted a simple brute-force initial 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 several static transfer configurations: transfer 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 in hindsight 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 for both multi-task and lifelong learning. This verifies that selective transfer is beneficial for learning multiple tasks simultaneously or consecutively.

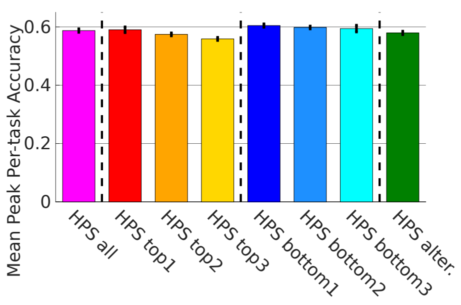
Instead of only considering such a restricted set of static

[1]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA [2]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Seungwon Lee <leeswon@seas.upenn.edu>.

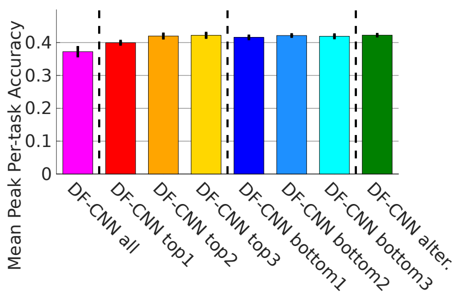
Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).



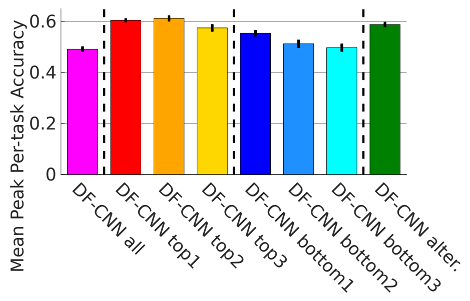
(a) Multi-Task CNN with HPS / CIFAR-100



(b) Multi-Task CNN with 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 intervals. This empirically shows that the optimal transfer configuration varies and is superior to transfer at all layers.

configurations in brute-force search, our goal is to automate this process of selective transfer *during* learning, enabling the learner to customize the transfer configuration to each task. We investigate the use of architecture search during learning to dynamically adjust the transfer configuration between tasks and over time, using expectation-maximization (EM) to train 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 nets to transfer different sets of layers for each task, allowing more flexibility over prior work in branching-based configurations for selective transfer. It also introduces little computational cost over the base learner in comparison to training selective transfer modules between task networks, and in contrast to the expense of brute-force search over all transfer configurations. To evaluate its effectiveness and generality, we applied LASEM to three architectures in several lifelong learning scenarios and compared it against various lifelong learning and architecture search methods.

2. The Effects of Different Transfer Configurations

This section further describes the initial 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 configuration 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 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 nets (Caruana, 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. Instead of using the same convolutional filters for multiple tasks, the convolutional filters are dynamically generated from a task-*independent* layer-*dependent* shared tensor through a task-specific deconvolutional operation and tensor contraction. Similar to HPS, the DF-CNN maintains task-specific fully connected topmost layers. When training the task models consecutively, gradients flow through to update the shared tensors and the task-specific parameters that transform those shared tensors to construct the task CNN. 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, with all others remaining task-specific. This architecture is the opposite of *Top* $d-k$, so 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 evaluated the performance of these transfer configurations on the CIFAR-100 (Krizhevsky & 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, using two of the available domains: "Product" images and "Real World" images. The CNN architectures used for each data set and optimization settings follow prior work and are detailed in Appendix A. During training, we measured the peak per-task accuracy on held-out test data, averaging results over five trials.

Our results in Figure 1 reveal that permitting transfer at all layers does not guarantee the best performance. Notably, we see that 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. It benefits from permitting the non-transfer layers to customize transferred knowledge to the individual task, but it is not consistently as good for HPS. This observation complicates learning on novel tasks, since the best transfer configuration depends both on the architecture and the task relations in the data set.

3. Architecture Search for the Optimal Transfer Configuration

The experiment presented above reveals that the transfer configuration can have a significant effect on lifelong 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.

3.1. The Transfer Configuration Search Problem

We can view the transfer configuration as a new hyperparameter for each task model. Even with the constraint that all task models use the same transfer configuration, the search space grows exponentially as the network gets deeper (i.e., 2^d configurations for d CNN layers). In more flexible settings where the transfer configuration is customized to each task, this search space grows even more: linearly in the number of tasks. Although we could compound this problem further by permitting partial transfer *within* a layer, we focus on optimizing layer-based transfer configurations.

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 element $c_{(t)}^{(l)}$ is a binary indicator whether or not the lth layer involves transfer. With a slight abuse of notation, we can compactly notate $\boldsymbol{c}_{(t)}$ by a set of indices of transferred layers. For example, the Alternating configuration described in Section 2 can be denoted by $\boldsymbol{c} = [0,1,0,1] \equiv \{2,4\}$; Figure 2 depicts this particular configuration for three approaches.

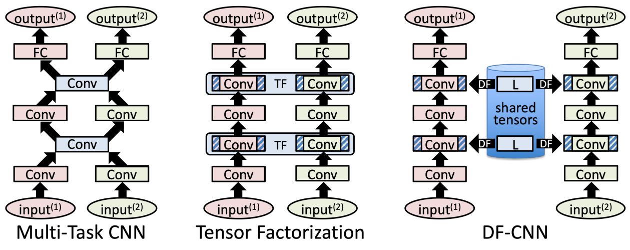


Figure 2: The Alternating $\{2,4\}$ transfer configuration for three approaches using CNNs with four conv. layers and one fully connected layer. Models are shown for two tasks, red and green, with shared or transfer-based layers in blue.

3.2. Lifelong Architecture Search via EM

Our goal is to determine the task-specific transfer configuration while simultaneously optimizing the log-likelihood 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 can employ expectation-maximization (EM) to perform this joint optimization. We call this approach Lifelong Architecture Search via EM (LASEM). Note that LASEM is agnostic to the knowledge-transfer methods and applicable both to classification and regression tasks. For each layer l, LASEM maintains a shared set of transfer-based model parameters $\boldsymbol{\theta}_s^{(l)}$ and, for each new task, a set of task-specific model parameters $\boldsymbol{\theta}_t^{(l)}$, using the chosen configuration $\boldsymbol{c}_{(t)}$ to determine which combination of parameters will be used to form the specific model for that task. In brief, the E-step estimates the usefulness of the representation that each transfer configuration $\boldsymbol{c} \in \mathcal{C}$ can learn from the given data (i.e., the data likelihood $P(y_{new} \mid X_{new}, \boldsymbol{c})$), while the M-step optimizes the parameters of the task model and shared knowledge. We next detail this approach.

First, we 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

$$P(\boldsymbol{c}_{(t)} = \boldsymbol{c}) = \pi_t(\boldsymbol{c}) = \frac{n_{\boldsymbol{c}} + 1}{\sum_{\tilde{\boldsymbol{c}} \in \mathcal{C}} (n_{\tilde{\boldsymbol{c}}} + 1)} \ , \qquad (1)$$

where $\boldsymbol{c}_{(t)}$ denotes the configuration for task t, and $n_{\boldsymbol{c}}$ is the number of previous mini-batches for which $\boldsymbol{c}$ is the most probable configuration derived from the posterior analytically. This estimate, which we adopt in the experiments, considers each transfer configuration solely based on the current task's data. Alternative priors could instead be used, such as measuring the historic transfer configuration frequency over all tasks (which assumes substantial similarity among tasks) or measuring configuration frequency over related tasks (which requires a notion of task similarity, such as via a task descriptor (Isele et al., 2016; Sinapov et al., 2015), or as determined dynamically by the task model's relation to shared knowledge). During development, we also considered estimating the prior based on the probability of configurations averaged over training samples instead of the mini-batches $n_{\boldsymbol{c}}$, but this alternative was not different statistically from Equation 1 in an empirical evaluation on CIFAR100 and Office-Home.

In the E-step, the posterior over 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 (X_{new}, y_{new}):

$$P(\boldsymbol{c} \,|\, X_{new}, y_{new}) \propto P(\boldsymbol{c}_{(t)} \!=\! \boldsymbol{c}) P(y_{new} \,|\, X_{new}, \boldsymbol{c}) \ . \quad (2)$$

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 vector ($\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_t$) over multiple possible configurations, we take the sum of the corresponding gradients weighted by the posterior estimate in Equation 2:

$$\begin{aligned}\boldsymbol{\theta}_s^{(l)} \leftarrow \boldsymbol{\theta}_s^{(l)} + \lambda \sum_{\boldsymbol{c} \in \mathcal{C} \,:\, c^{(l)}=1} & P(\boldsymbol{c} \mid X_{new}, y_{new}) \\ & \nabla \log P(y_{new} \mid X_{new}, \boldsymbol{c}) \quad \forall l \in \{1, \ldots, d\} \ ,\end{aligned} \qquad (3)$$

and

$$\begin{aligned}\boldsymbol{\theta}_t^{(l)} \leftarrow \boldsymbol{\theta}_t^{(l)} + \lambda \sum_{\boldsymbol{c} \in \mathcal{C} \,:\, c^{(l)}=0} & P(\boldsymbol{c} \mid X_{new}, y_{new}) \\ & \nabla \log P(y_{new} \mid X_{new}, \boldsymbol{c}) \quad \forall l \in \{1, \ldots, d\} \ .\end{aligned} \qquad (4)$$

The main difference between the update rules in Equations 3 and 4 is the condition for the index of the summation. Since one gradient step on the configurations $\{\boldsymbol{\theta}_s^{(l)}, \boldsymbol{\theta}_t^{(l)}\}_{l=1}^{d}$ may have little effect on the likelihood, we can hold the likelihood fixed to take multiple M-steps per E-step by iterating Equations 3 and 4.

LASEM is specified in Algorithm 1 and illustrated in Figure 3. The parameters of the transfer-based components are initialized only at the beginning of the lifelong learning process (line 1), while the parameters of the task-specific components and the prior probability of configurations are initialized when the algorithm encounters a new task (lines 7–10). Each iteration, the lifelong learner obtains a mini-batch of labeled data (X_{new}, y_{new}) drawn i.i.d for some task t, training the task model online. Typically, there would be multiple consecutive mini-batches experienced per task

Algorithm 1 Lifelong Architecture Search via EM

```
 1: $\{\boldsymbol{\theta}_s^{(l)}\}_{l=1}^{d} \leftarrow$ randomInitializer(TASK_NET_SIZE)
 2: ▷ Loop over entire training lifetime
 3: while isMoreTrainingDataAvailable() do
 4:    ▷ Get current task data
 5:    $(X_{new}, y_{new}, t) \leftarrow$ getNextTrainingMinibatch()
 6:    ▷ Initialize parameters for new tasks
 7:    if isNewTask($t$) then
 8:       $\{\boldsymbol{\theta}_t^{(l)}\}_{l=1}^{d} \leftarrow$ randomInitializer(TASK_NET_SIZE)
 9:       $\pi_t \leftarrow$ priorInitializer()
10:    end if
11:    ▷ Update task model and transfer configuration
12:    Calculate likelihood $P(y_{new} \mid X_{new}, \boldsymbol{c}) \ \forall \boldsymbol{c} \in \mathcal{C}$
13:    Calculate posterior:                                ▷ Eq 2
          $P(\boldsymbol{c} \mid X_{new}, y_{new}) \propto \pi_t(\boldsymbol{c}) P(y_{new} \,|\, X_{new}, \boldsymbol{c})$
          $\forall \boldsymbol{c} \in \mathcal{C}$
14:    loop $numMSteps$ times do                           ▷ Eq 3 & 4
15:       $\{\boldsymbol{\theta}_s^{(l)}, \boldsymbol{\theta}_t^{(l)}\}_{l=1}^{d} \leftarrow$ gradOptimizer$\Big(X_{new}, y_{new}, \lambda,$
                $P(\mathcal{C} \mid X_{new}, y_{new}), \{\boldsymbol{\theta}_s^{(l)}, \boldsymbol{\theta}_t^{(l)}\}_{l=1}^{d}\Big)$
16:    end loop
17:    $\pi_t \leftarrow$ priorUpdater$(\pi_t, P(\mathcal{C} \mid X_{new}, y_{new}))$    ▷ Eq 1
18: end while
```

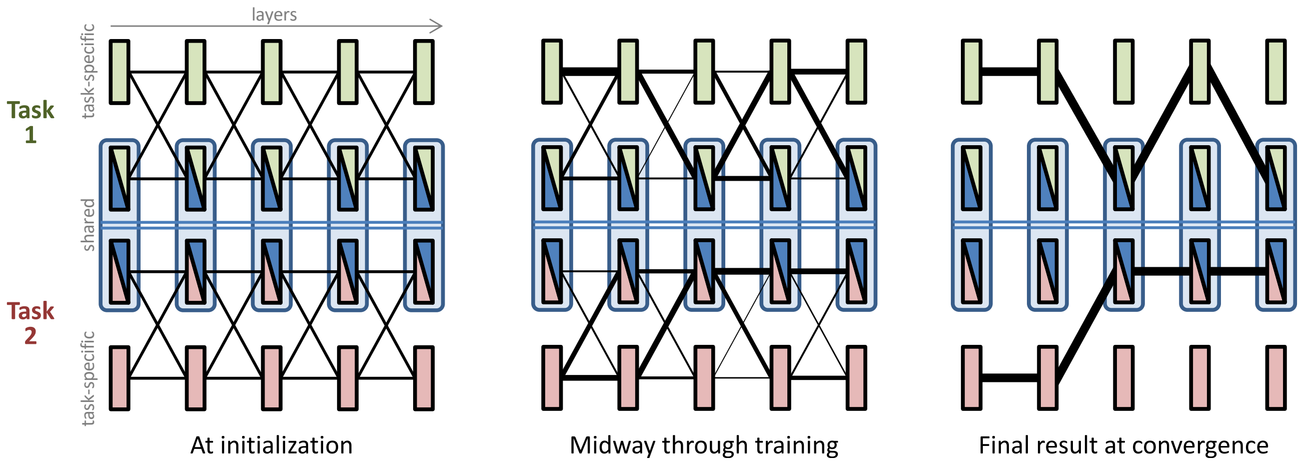


Figure 3: An illustration of LASEM's dynamic progression of the transfer configurations during training. This example depicts two task models (green and red), considering that there are multiple other task models that are not depicted. The line thickness connecting adjacent layers represents the likelihood of the connection, which LASEM learns from data while simultaneously training the individual layers. At initialization, all configurations are given uniform weight, which could be modified to incorporate prior knowledge of task similarity. As LASEM learns the transfer configuration likelihoods and layer weights from data, the posterior updates dynamically until the transfer configuration finalizes at convergence to a configuration customized to each task.

before the environment would switch to a new task. The algorithm applies the E-step (lines 12, 13, and 17) and M-step (lines 14–16) using the data mini-batch at each iteration. LASEM takes E- and M-steps for each mini-batch, so over the consecutive mini-batches per task, this process would be similar to the alternation of multiple E- and M-steps performed by the standard EM algorithm. We consider lifelong learning scenarios in which a learner has no control over tasks, so the current task can be switched to another one regardless of the convergence of the learning algorithm. Consequently, EM convergence is not required per task since convergence may occur over multiple tasks; however, the convergence can still be monitored by checking the probability weight over transfer configurations during training.

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. This approximation reduces the number of parameters and permits parameter updates across transfer configurations via a single gradient step, conserving memory.

3.3. Scaling to Numerous Layers

For a d-layer neural network with a time complexity of $N(\cdot)$, the per-iteration computational complexity of both the E- and M-steps are $O(2^d N(\cdot))$. As the network depth d increases, it is well known that neural architecture search (NAS) requires exponentially additional computation. To remedy this issue, LASEM can adopt a similar solution to that of other NAS methods (Pham et al., 2018; Liu et al., 2019a) by considering transfer configurations over *groups* of layers instead of individual layers, thereby reducing the search space. We can redefine 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 $|\mathcal{P}| \ll d$. Consequently, this reduces the search space from 2^d to $2^{|\mathcal{P}|}$. Appendix D provides further details on this layer-grouping approach; we evaluate this variation in Section 4.4.

3.4. Computational Cost

The computational cost of LASEM depends heavily on the tasks, the choice of architecture for the task models, and how quickly the transfer configuration converges. We show empirically in Section 4.2 that LASEM in practice introduces relatively little additional time complexity ($\sim$30–50% over the base learner). Empirically, the E-step takes $\sim$15–20% time of the M-step, but frequent switching of the configuration by the E-step may interfere with the M-step, consequently harming the convergence speed. Taking more M-steps per E-step (by increasing $numMSteps$) can improve this stability and consequently the computational complexity by accelerating convergence. LASEM's memory requirements are analyzed in Appendix E.

4. Experiments

We evaluated LASEM following the same experimental protocol for lifelong learning as used in Section 2; see Appendix A for details. In addition to using the CIFAR100 and Office-Home data sets, we introduce a lifelong learning

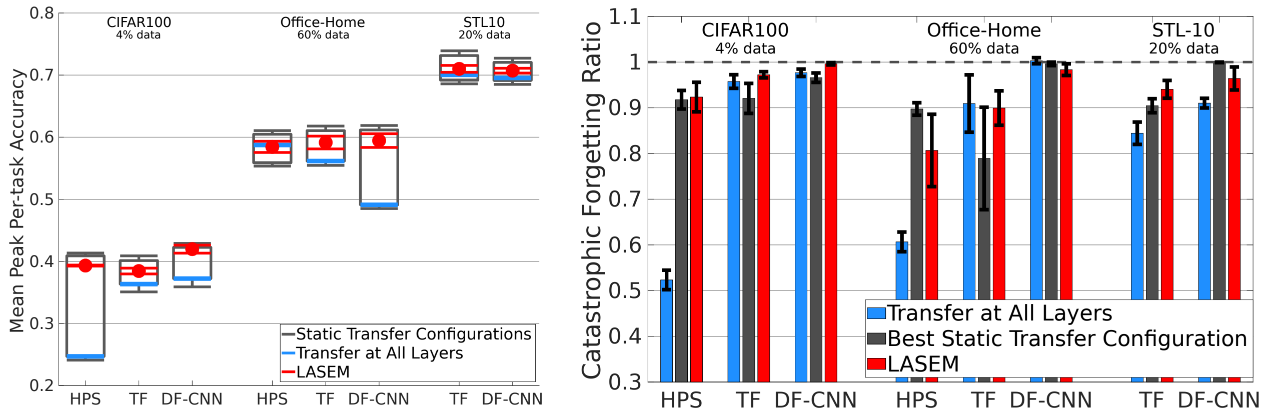


Figure 4: (Left) Performance of LASEM applied to three methods and three lifelong scenarios. Black boxes show the range of mean accuracies that different static configurations can achieve, with the blue lines denoting mean performance of Transfer at All Layers. The red dots denote the mean performance of LASEM. Whiskers depict 95% confidence intervals. (Right) Mean catastrophic forgetting ratio after training all tasks. Less forgetting is indicated by a ratio near 1.0; see Appendix C. The best static transfer configuration was chosen in hindsight via an expensive brute-force search.

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. Considering a limited-data regime, we constructed 20 tasks of three-way classification using roughly 20% and 5% of the given training data for training and validation, respectively. To increase the task variations within STL-10, for each task we randomly chose three image 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 different random seeds. The code and data set generators are available at https://github.com/Lifelong-ML/LASEM.

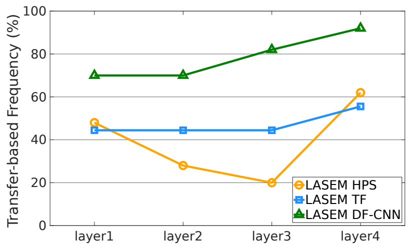
4.1. Performance of LASEM

We applied LASEM to three algorithms: a multi-task CNN with hard-parameter sharing (HPS) (Caruana, 1997), Tensor Factorization (TF) (Yang & 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 the parameters of each task model are generated via transfer.

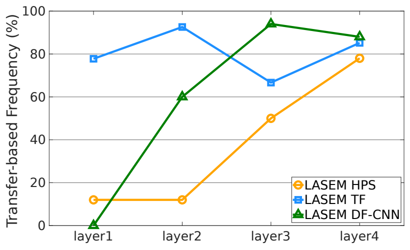
Figure 4 (left) compares the performance of the task-specific transfer configurations discovered by LASEM (shown in red) to using a single static transfer configuration (black boxes). These black boxes depict the performance range of the methods using various static transfer configurations (i.e., All, 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 configurations. Examining the catastrophic forgetting ratio (Figure 4-right, with details in Appendix C) reveals the importance of selecting the appropriate transfer configuration for maintaining performance on previous tasks, revealing that LASEM exhibited less forgetting than baselines in most cases, especially on the DF-CNN. Moreover, LASEM imposes little additional cost in order to determine the transfer configuration. In timing experiments (Table 3), we found that, compared to training with a pre-determined static configuration, LASEM requires only 30–50% additional wall-clock time to search over 16 configurations of a network with four layers, and only double the time to search over 64 configurations of a network with six layers. In stark contrast, brute-force search over 16 transfer configurations in our experiments required a mean of 890% over the base learner's cost (with a range of 560–1,500% additional cost).

The frequencies of the transfer configurations chosen by LASEM are depicted in Figure 5; see Appendix B for detailed results. Figure 5 shows the proportion of time each layer was chosen to be transfer-based. We see that HPS tends to prefer task-specific layers, while TF and DF-CNN are more likely to use transfer layers due to their ability to adapt transferred knowledge. We can also see trends among the chosen layers, such as DF-CNN preferring transfer among higher layers.



(a) CIFAR-100



(b) Office-Home

Figure 5: Frequency of each layer being transfer-based according to the selection of LASEM. Generally, upper layers are preferable for transfer, but there are exceptions, i.e. HPS on CIFAR-100.

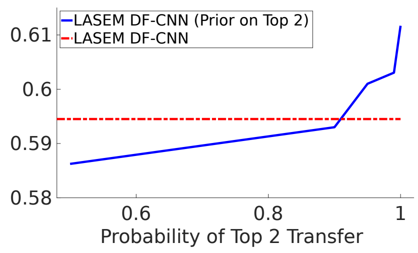


Figure 6: Performance of LASEM DF-CNN compared to LASEM with a fixed posterior distribution over the optimal configuration (on Office-Home).

4.2. Comparison to Other Selective Transfer Methods

We compared LASEM on Office-Home against other methods that employ some notion of selective transfer, including the Dynamically Expandable Network (DEN) (Yoon et al., 2018), the Additive Parameter Decomposition Network (APD-Net) (Yoon et al., 2020), the Progressive Neural Net (ProgNN) (Rusu et al., 2016), and Differentiable Architecture Search (DARTS) (Liu et al., 2019a). DEN is a lifelong learner that extends HPS by expanding, splitting, and selectively retraining the network to introduce both shared and task-specific parameters in each layer if required. APD-Net has base parameters shared across tasks like HPS, but introduces task-specific masks and additive parameters for adaptation to each task. ProgNN learns lateral connections from earlier task models to the current task model. 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 create a transfer-based layer. DARTS is another general framework for neural architecture search, which simultaneously determines both the most suitable operation of each layer and the best architecture of stacking these layers.

Table 1 summarizes the performance of these methods and our approach in terms of the mean peak per-task accuracy, catastrophic forgetting ratio, and training time. APD-Net, ProgNN and LASEM DF-CNN are statistically indistinguishable and perform better than the other methods in terms of accuracy. However, APD-Net is weaker in retaining knowledge from earlier tasks, as shown by its catastrophic forgetting ratio being significantly lower than ProgNN and LASEM DF-CNN. LASEM DF-CNN is ~14% faster than ProgNN, whose time complexity is proportional to the square of the number of tasks. DEN and DARTS have better training times, but fail to perform as well. LASEM shows high accuracy regardless of the base lifelong learner (e.g., HPS, TF, or DF-CNN) while introducing relatively little additional time complexity (~30–50% over the base learner; see Table 3).

Selective Sharing	Accuracy(%)	Forgetting Ratio	Time (k sec)
DEN	48.00 $\pm$ 0.60	0.28 $\pm$ 0.01	55.9
APD-Net	**59.58** $\pm$ **0.45**	0.83 $\pm$ 0.03	21.5
ProgNN	**60.03** $\pm$ **0.45**	1.00 $\pm$ 0.00	96.7
DARTS HPS	45.64 $\pm$ 1.20	0.70 $\pm$ 0.07	43.8
DARTS DF-CNN	56.77 $\pm$ 0.49	0.35 $\pm$ 0.04	33.2
LASEM HPS	58.44 $\pm$ 0.90	0.81 $\pm$ 0.08	70.2
LASEM TF	59.14 $\pm$ 0.80	0.90 $\pm$ 0.04	77.3
LASEM DF-CNN	**59.45** $\pm$ **1.10**	0.98 $\pm$ 0.01	83.2

Table 1: Comparison of peak per-task accuracy, forgetting, and training time for the same epochs between baselines and LASEM on Office-Home, $\pm$ 95% confidence interval.

4.3. Effects of Non-Optimal Transfer Configurations

Besides the capability to customize the transfer configuration to each task, LASEM has a key difference from using a static transfer configuration as in Section 2: LASEM updates both the transfer-based and task-specific parameters ($\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_t$) by gradients backpropagated from the loss of *all* configurations weighted by the posterior. Consequently, gradients from non-optimal configurations might act as noise or be counterproductive to the optimization process.

To measure the impact of this aspect on LASEM, we performed an ablative experiment using a static probability on the transfer configurations, instead of the posterior derived from data. This makes LASEM always select the same transfer configuration for all task models, with the experiment controlling the amount of adverse effects from non-optimal configurations during LASEM's optimization.

We first determined the optimal transfer configuration (Top 2 in this experiment), and gave it a static probability of selection, which varied from $P = 0.5$ to 1 with a uniform distribution over other configurations. Figure 6 compares the full LASEM DF-CNN against the ablated version. Knowing the correct transfer configuration *a priori* (when $P = 1$) certainly does improve performance, but the overall performance difference is relatively small as P varies. Therefore, the effect of interference from non-optimal configurations is minimal, but does exist. Using a more informed prior over the configurations based on task similarity may further improve LASEM, which we leave to future work.

4.4. Scalability Experiments

To evaluate scalability, we applied LASEM to a larger deep net, ResNet-18 (He et al., 2016), and a longer sequence of tasks than in the prior experiment. Since the search space of transfer configurations is exponential in the number of layers, we used the approach described in Section 3.3, bundling the 17 convolutional layers of ResNet-18 into groups to reduce the search space. Appendix D provides details on the experiments and results on other architectures.

As shown in Table 2, LASEM improves both the peak per-task accuracy and catastrophic forgetting ratio of HPS using the ResNet-18 HPS architecture, performing well under the restricted transfer configuration search space. Interestingly, the relative difference of training times between learning with and without LASEM decreases when given more tasks to learn, which can be attributed to faster convergence of the EM process. These results show that LASEM can scale effectively to deeper architectures through group-based transfer configurations and larger numbers of tasks.

Selective Sharing	Accuracy(%)	Forgetting Ratio	Time (k sec)
CIFAR-100 (10 Tasks)			
ResNet HPS	38.51 ± 0.53	0.54 ± 0.03	4.47
LASEM ResNet HPS 4G	39.47 ± 0.30	0.79 ± 0.05	11.1
LASEM ResNet HPS 5G	39.07 ± 1.10	0.79 ± 0.08	14.4
LASEM ResNet HPS 6G	**40.00 ± 0.65**	0.75 ± 0.06	25.1
LASEM ResNet HPS 7G	39.32 ± 0.33	0.74 ± 0.07	46.9
CIFAR-100 (40 Tasks)			
ResNet HPS	38.01 ± 0.27	0.41 ± 0.02	63.4
LASEM ResNet HPS 4G	**39.89 ± 0.73**	0.62 ± 0.03	94.1
LASEM ResNet HPS 5G	38.89 ± 0.11	0.55 ± 0.07	109.2
LASEM ResNet HPS 6G	39.17 ± 0.62	0.56 ± 0.09	154.1

Table 2: Scalability of LASEM on ResNet-18 with varying numbers of layer groups (#G), and a longer task sequence in the lifelong learning setting. LASEM shows improvement in both the peak per-task accuracy and catastrophic forgetting with ± 95% confidence intervals.

4.5. Comparison to Task-Wise Brute-Force Search

We compared LASEM to the performance of task-wise brute-force search over transfer configurations, shown in Table 3. The brute-force search method trains every possible transfer configuration and chooses the best one for each task in hindsight, thereby allowing task models to use different transfer configurations. The accuracy of brute-force search is almost indistinguishable from LASEM at 95% confidence, but it requires 4.0–9.5× time. This result shows that LASEM's dynamic transfer configuration performs well while reducing training time.

5. Related Work

The simplest transfer mechanism is hard parameter sharing (HPS), which directly reuses parameters (e.g., layers) between task models (Caruana, 1997). HPS is beneficial when tasks share identical features, but its structural rigidity degenerates performance as tasks become diverse. Constraints such as regularization (Kirkpatrick et al., 2017; Yoon et al., 2018; He et al., 2018), orthogonality (Suteu & Guo, 2019; Riemer et al., 2019; Farajtabar et al., 2020) or attention (Serra et al., 2018; Yoon et al., 2020; Abati et al., 2020) may reduce interference among tasks, but can deter positive transfer. Using tree-like structures (Lu et al., 2017; Vandenhende et al., 2019; He et al., 2018) as the transfer configuration for HPS in multi-task nets gives flexibility, but assumes that lower-level representations are shared, which may not be the case for diverse tasks, as shown in this paper.

Soft parameter sharing (Duong et al., 2015; Bilen & Vedaldi, 2017) builds task-specific networks with weights that are related to other task models via implicit constraints. This architecture provides flexibility to the representations that each task network can learn, so it typically outperforms HPS for more diverse tasks. Success has often been found by using task-agnostic shared knowledge with a task-specific mapping from that shared knowledge to the task models, facilitating transfer between tasks (Yang & Hospedales, 2017; Bulat et al., 2020; Lee et al., 2019; Liu et al., 2019b). These works focus on the mapping operation, but put less importance in which layers to transfer, as we explored.

Direct reuse of learned representations from previous tasks models (Rusu et al., 2016; Misra et al., 2016; Cao et al., 2018; Xiao et al., 2018) prevents forgetting, but only permits forward transfer to new tasks (not reverse transfer) and exhibits super-linear training time w.r.t the number of tasks. Progress and compress (Schwarz et al., 2018) tackles this issue by combining progressive neural nets and elastic weight consolidation (Kirkpatrick et al., 2017), but this method suffers from the same capacity limitations of HPS for diverse tasks since one neural net must handle all learned tasks.

Neural architecture search (NAS) examines both the operators and their order in a neural net to optimize performance (Elsken et al., 2018). Our problem of selective layer-based

Architecture	LASEM	Brute-force Search		Transfer All Layers	
	Accuracy (%)	Accuracy (%)	Relative Time	Accuracy (%)	Relative Time
CIFAR-100 (10 Tasks)					
HPS	39.3 $\pm$ 0.1	40.4 $\pm$ 0.3	6.55	24.7 $\pm$ 0.6	0.78
TF	38.4 $\pm$ 0.5	39.9 $\pm$ 1.1	8.81	36.3 $\pm$ 1.0	0.64
DF-CNN	42.0 $\pm$ 0.6	42.6 $\pm$ 0.7	9.45	36.3 $\pm$ 1.3	0.59
Office-Home (10 Tasks)					
HPS	58.4 $\pm$ 0.9	59.4 $\pm$ 0.2	4.72	54.9 $\pm$ 0.7	0.72
TF	59.1 $\pm$ 1.0	58.7 $\pm$ 0.3	5.22	56.2 $\pm$ 0.7	0.66
DF-CNN	59.5 $\pm$ 1.1	58.8 $\pm$ 0.3	4.04	49.1 $\pm$ 0.6	0.61

Table 3: Comparison of test accuracy and training time for the same epochs to brute-force configuration search, $\pm$ 95% confidence. Time is measured as the ratio relative to LASEM. (Time > 1.0 means that training is slower than LASEM.)

transfer is an instance of NAS. Strategies for NAS include reinforcement learning (Tan et al., 2019; Chang et al., 2019), evolutionary algorithms (Fernando et al., 2017) and gradient-based learning (Alet et al., 2018). In contrast to these methods, DARTS makes optimization more feasible by using a soft selection of the operators: a weighted sum of operations. Most NAS methods including DARTS train better once the architecture has stabilized, but the weights of DARTS' soft selections are susceptible to vanishing gradients, so it is slower to stabilize than LASEM.

Modular networks learn sub-networks of particular low-level functions (i.e., modules) and how to construct the full task network using these modules. The main challenge of training modular networks is the interdependence of searching for the optimal compositional structure and learning the optimal modules (Rosenbaum et al., 2019), so algorithms such as meta-learning (Alet et al., 2018), expectation-maximization (Kirsch et al., 2018) and reinforcement learning (Rosenbaum et al., 2018; Chang et al., 2019) have been used to tackle this challenge. Despite the similarity of their high-level problems to this paper, these works on modular nets ensure the reusability of modules by the simultaneous access to data from multiple tasks in batch up front; only recently has the issue of modular reusability been explored in the lifelong setting (Mendez & Eaton, 2021). In this work, we search over simplified configurations of shared or task-specific layers; the latter need not be applicable to multiple tasks.

Empirical investigation of routing networks, a type of modular networks, has emphasized the importance of two key characteristics in ensuring representational power: the diversity of modules per layer and the depth of the network (i.e., the number of routing paths) (Ramachandran & Le, 2019). Although the problem setting of routing networks is different from ours, these observations may benefit our understanding of LASEM. The diversity of per-layer experts is dictated by the various mechanisms of the base lifelong learners (e.g., HPS, TF or DF-CNN), and so base learners that ensure more diversity in the layers (such as the DF-CNN) are likely to fare better. This may also broach the idea of having multiple potential shared layers to select from, increasing diversity at additional computational cost. The issue of depth is dependent on the chosen task model, but LASEM is capable of scaling to numerous layers as discussed in Section 3.3 and Appendix D.

6. 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. Choosing the optimal transfer configuration significantly improves the performance of the DF-CNN and TF over the original methods. Using a data-driven dynamic transfer configuration reduces the assumptions of algorithm designers in terms of task similarities.

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.

Acknowledgments

We are grateful to Jorge Mendez, Kyle Vedder, David Kent, Marcel Hussing, Meghna Gummadi, and the anonymous reviewers for their helpful feedback on this work. The research presented in this paper was partially supported by the DARPA Lifelong Learning Machines program under grant FA8750-18-2-0117, the DARPA SAIL-ON program under contract HR001120C0040, and the Army Research Office under grant W911NF20-1-0080. The views and conclusions in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, ARO, or the U.S. Government.

References

Abati, D., Tomczak, J., Blankevoort, T., Calderara, S., Cucchiara, R., and Bejnordi, B. E. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Alet, F., Lozano-Perez, T., and Kaelbling, L. P. Modular meta-learning. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 856–868, 2018.

Bilen, H. and Vedaldi, A. Universal representations: the missing link between faces, text, planktons, and cat breeds. *CoRR*, abs/1701.07275, 2017.

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

Cao, J., Li, Y., and Zhang, Z. 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*, pp. 4290–4299, 2018.

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

Chang, M., Gupta, A., Levine, S., and Griffiths, T. L. Automatically composing representation transformations as a means for generalization. In *Proceedings of the International Conference on Learning Representations*, 2019.

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

Duong, L., Cohn, T., Bird, S., and Cook, P. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pp. 845–850, 2015.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *CoRR*, abs/1808.05377, 2018.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. Orthogonal gradient descent for continual learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2020.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

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

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

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Kirsch, L., Kunze, J., and Barber, D. Modular networks: Learning to decompose neural computation. *Advances in Neural Information Processing Systems*, 31, 2018.

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

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

Liu, H., Simonyan, K., and Yang, Y. DARTS: differentiable architecture search. In *Proceedings of the International Conference on Learning Representations*, 2019a.

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

Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. 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.

Mendez, J. and Eaton, E. Lifelong learning of compositional structures. In *Proceedings of the International Conference on Learning Representations*, 2021.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3994–4003, 2016.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. Efficient neural architecture search via parameter sharing. In *Proceedings of the International Conference on Machine Learning*, 2018.

Ramachandran, P. and Le, Q. V. Diversity and depth in per-example routing models. In *Proceedings of the International Conference on Learning Representations*, 2019.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *Proceedings of the International Conference on Learning Representations*, 2019.

Rosenbaum, C., Klinger, T., and Riemer, M. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *Proceedings of the International Conference on Learning Representations*, 2018.

Rosenbaum, C., Cases, I., Riemer, M., and Klinger, T. Routing networks and the challenges of modular and compositional computation. *CoRR*, abs/1904.12774, 2019.

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

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for continual learning. In *Proceedings of the International Conference on Machine Learning*, pp. 4528–4537, 2018.

Serra, J., Suris, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the International Conference on Machine Learning*, pp. 4548–4557, 2018.

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

Suteu, M. and Guo, Y. Regularizing deep multi-task networks using orthogonal gradients. *CoRR*, abs/1912.06844, 2019.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. MnasNet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

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

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

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

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

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

Yoon, J., Kim, S., Yang, E., and Hwang, S. Scalable and order-robust continual learning with additive parameter decomposition. In *Proceedings of the International Conference on Learning Representations*, 2020.