Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)

Learning Shared Knowledge for Deep Lifelong Learning Using Deconvolutional Networks

Seungwon Lee[1] , **James Stokes**[1,2] and **Eric Eaton**[1]
[1]University of Pennsylvania, Philadelphia, PA, USA
[2]Flatiron Institute, New York, NY, USA
leeswon@seas.upenn.edu, jstokes@flatironinstitute.org, eeaton@cis.upenn.edu

Abstract

Current mechanisms for knowledge transfer in deep networks tend to either share the lower layers between tasks, or build upon representations trained on other tasks. However, existing work in non-deep multi-task and lifelong learning has shown success with using factorized representations of the model parameter space for transfer, permitting more flexible construction of task models. Inspired by this idea, we introduce a novel architecture for sharing latent factorized representations in convolutional neural networks (CNNs). The proposed approach, called a *deconvolutional factorized CNN*, uses a combination of deconvolutional factorization and tensor contraction to perform flexible transfer between tasks. Experiments on two computer vision data sets show that the DF-CNN achieves superior performance in challenging lifelong learning settings, resists catastrophic forgetting, and exhibits reverse transfer to improve previously learned tasks from subsequent experience without retraining.

1 Introduction

Much of the success of deep discriminative learning, including visual recognition [Krizhevsky *et al.*, 2012; Simonyan and Zisserman, 2015; He *et al.*, 2016] and image segmentation, stems from the remarkable ability of neural networks to generalize to unseen data that is drawn from the same underlying data generating process as the training set. Comparatively less success has been achieved when deep networks are deployed in an online multi-task learning (MTL) or lifelong learning fashion [Chen and Liu, 2016]. In these settings, the system faces the more formidable challenge of repeatedly generalizing to new data distributions (i.e., tasks), by leveraging knowledge of previously encountered tasks. Such a system should improve performance on new tasks via transfer, without compromising performance (i.e., catastrophic forgetting [Kirkpatrick *et al.*, 2017]) on previously encountered tasks, and additionally permit new knowledge to benefit previously learned tasks (i.e., reverse transfer [Ruvolo and Eaton, 2013]). Therefore, in contrast to single-task learning, lifelong learning systems must exploit the relationships that exist between the tasks to facilitate transfer.

A natural approach to the lifelong learning problem is to exploit the compositional nature of neural networks. In general, we expect a deep neural net to learn hierarchical features whose level of abstraction correlates closely with depth in the network. Many methods for MTL and lifelong learning with deep nets exploit this property, sharing lower layers of the network between tasks while enforcing separate topmost layer(s) for each task. Such hard parameter sharing (HPS) methods typically train the shared lower layers in a multi-task setting, under the hope that they will acquire universal features suitable for multiple related tasks [Baxter, 2000]. These methods support limited transfer to new tasks by training task-specific classifiers on top of these pre-trained lower layers.

Rather than imposing rigid task relationships by explicit weight sharing, it is desirable to learn task relationships organically from data. Non-deep MTL methods [Kumar and Daume, 2012; Maurer *et al.*, 2013; Ruvolo and Eaton, 2013] have shown success with this notion, enabling transfer between tasks by factorizing the model parameter space via a shared latent knowledge base. Each task-specific model is then reconstructed as a linear combination of components from the shared knowledge base. Although effective in providing flexible transfer, this approach is currently limited to learning with shallow models. Deep MTL methods such as tensor factorization [Yang and Hospedales, 2017], progressive neural networks [Rusu *et al.*, 2016], and dynamic filters [Jia *et al.*, 2016; Ha *et al.*, 2017] relate multiple deep networks and permit transfer, but do not provide sufficiently flexible transfer mechanisms nor support all characteristics (such as reverse transfer) needed for lifelong learning.

Such a factorized knowledge base used for shallow MTL, however, could be adapted to connect the weight matrices between multiple deep networks since a deep network can be thought of as a stack of shallow models. This factorized form of the knowledge base would permit flexible transfer between multiple deep networks, adapting the transfer based on the similarity of each network's respective task.

This idea is the focus of this paper, in which we propose a new shared-knowledge neural architecture that exploits the structure of convolutional neural networks (CNNs). Inspired by the success of deconvolutional networks for image segmentation [Noh *et al.*, 2015], we introduce a deconvolutional approach to share latent knowledge across multiple CNNs. Our approach, *deconvolutional factorization*, can be viewed

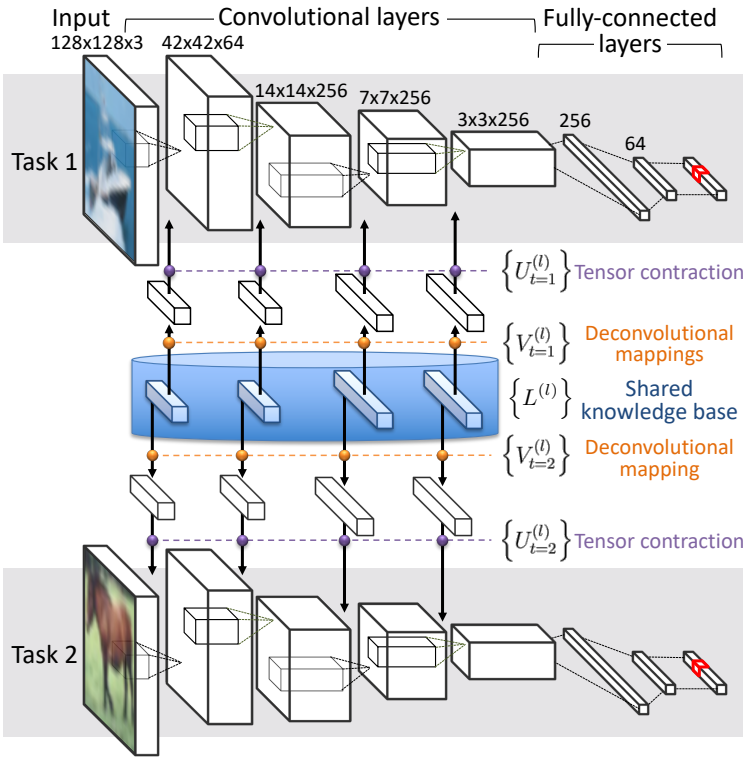


Figure 1: The DF-CNN architecture, depicted for two tasks. Each task is modeled by a CNN (in the gray shaded boxes). The filters of each convolutional layer are reconstructed from the shared latent knowledge base (blue) via the learned deconvolutional mapping (orange) and tensor contraction (purple) that facilitate transfer between the CNNs. The number of layers and the size of each layer's output corresponds to the application of our approach to the CIFAR-100 data set.

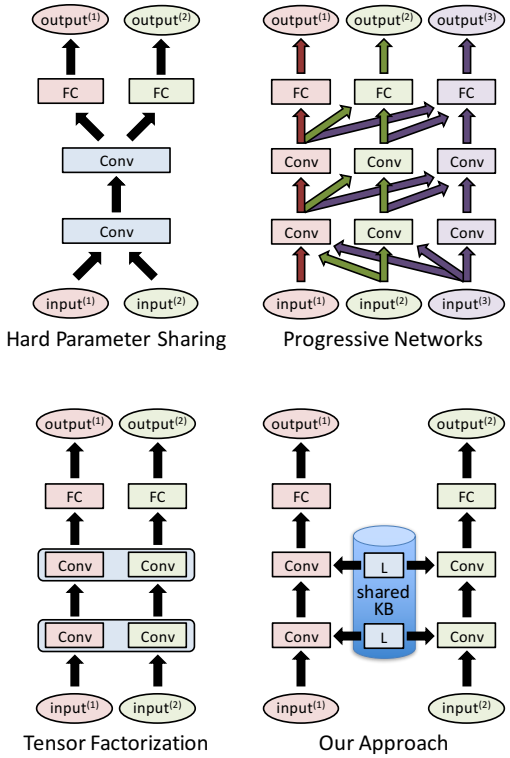


Figure 2: Comparison of our approach for soft knowledge sharing to other neural network architectures for learning multiple tasks (pink, green, and purple denote different tasks). This example uses a simplified three-layer network architecture. Light blue layers are shared or coupled between task models.

as conceptually analogous to using a sparse factorization of the linear model parameter space for transfer in shallow MTL [Kumar and Daume, 2012; Maurer *et al.*, 2013] and lifelong learning [Ruvolo and Eaton, 2013] methods, but generalized to deep networks. In our adaptation of factorized transfer to deep convolutional networks, the model parameters to be factored are the filters of the convolutions on each layer, and the sparsity condition follows from the fact that the deconvolution operator maps from a compact latent representation to a higher-dimensional feature space. This contrasts with existing approaches which utilize tensor factorization [Yang and Hospedales, 2017] with regularization to achieve sparsity.

We evaluate our proposed method on image recognition in challenging lifelong learning settings, where the system learns tasks consecutively. Our results showcase the strong benefits of the flexible transfer process provided by deconvolutional factorization over HPS, progressive networks, and other current methods. We also show that our approach converges rapidly to a high-performance model by utilizing knowledge transfer, and that it resists catastrophic forgetting.

2 The Deconvolutional Factorized CNN

A lifelong learning system faces a series of consecutive tasks $\mathcal{Z}_1, \ldots, \mathcal{Z}_{T_{\max}}$, and must learn a model (i.e., a classifier) for each task [Chen and Liu, 2016]. The system has no *a priori* information about the task distribution, order, or total number of tasks $T_{\max}$. This paper focuses on the classification setting, where each task $\mathcal{Z}_t$ has an associated data feature space $\mathcal{X}_t$ and label space $\mathcal{Y}_t$, from which labeled examples are drawn. Here, each learning task $\mathcal{Z}_t$ admits an associated convolutional neural network $CNN_t : \mathcal{X}_t \mapsto \mathcal{Y}_t$ trained on labeled data for that task. Each CNN_t has d layers and is trained consecutively, possibly with transfer from any previously learned tasks $\mathcal{Z}_1, \ldots, \mathcal{Z}_{t-1}$.

Our architecture, called a *deconvolutional factorized CNN* (DF-CNN), seeks to address this lifelong learning problem using deep convolutional networks with a shared knowledge base to enable transfer between tasks (Fig. 1). To facilitate transfer, our architecture maintains a shared latent knowledge base that connects the various layers *across* the task-specific CNNs. Recall that a CNN is composed of multiple layers of stacked filters, each of which is parameterized. The filters of the CNNs are generated from the learned latent knowledge base by the deconvolution operator (transposed convolution), followed by a tensor contraction. The remainder of this section explains this process. Unlike previous methods that involve tensor factorization to achieve sparsity, our proposal is naturally sparse by virtue of the deconvolution operator.

2.1 Factorized Transfer

To enable transfer among the different CNN_t task models, we draw inspiration from the use of factorized transfer in shallow MTL [Kumar and Daume, 2012; Maurer *et al.*, 2013] and lifelong learning [Ruvolo and Eaton, 2013] methods. These shallow methods learn a set of T task-specific linear models parameterized by $\boldsymbol{W} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T] \in \mathbb{R}^{d \times T}$ and assume that these model parameters admit a rank-constrained matrix factorization of the form $\boldsymbol{W} = \boldsymbol{L}\boldsymbol{S}$, where $\boldsymbol{L} \in \mathbb{R}^{d \times k}$ is a basis over the model parameter space, $\boldsymbol{S} \in \mathbb{R}^{k \times T}$ are the coefficients over this basis to reconstruct the parameters, and k is the dimension of the latent space. In effect, these approaches learn a knowledge base $\boldsymbol{L}$ that represents a shared subspace for the model parameters, and facilitate transfer to new tasks by learning models within this subspace.

Since these methods only operate on linear task models, we cannot adopt them directly for use on deep nets. However, we show next that we can adapt this notion of a shared knowledge base in combination with a novel type of factorized transfer via deconvolution to operate effectively on CNNs.

2.2 Factorized Transfer via Deconvolution

For each convolutional layer $l \in \{1, \ldots, d\}$ of each task-specific CNN_t, let $W_t^{(l)} \in \mathbb{R}^{h \times w \times c_{\text{in}} \times c_{\text{out}}}$ denote its corresponding filters where h and w are the filter height and width, and c_{in} and c_{out} are the numbers of input and output channels.

To enable transfer between the convolutional layers of different CNN_t task models, we introduce a task-*independent* layer-*dependent* shared knowledge base $L^{(l)}$ for each layer l, which is shared across all tasks. Following similar assumptions used in factorized transfer for shallow models, we assume that each $W_t^{(l)}$ is derived from the corresponding shared latent knowledge base $L^{(l)}$, enabling connections between filters at the l-th layer of different task models.

Specifically, we utilize a deconvolutional mapping and a tensor contraction to factorize the filters $\{W_t^{(l)}\}_{t=1}^T$ into the shared knowledge base $L^{(l)}$, which we take to be a 3rd-order tensor $L^{(l)} \in \mathbb{R}^{\hat{h} \times \hat{w} \times \hat{c}}$. We first deconvolve $L^{(l)}$ into

$$D_t^{(l)} = \text{deconv}(L^{(l)}; V_t^{(l)}) \ , \tag{1}$$

where $D_t^{(l)}$ is a 3rd-order tensor of size $h \times w \times c$, $V_t^{(l)} \in \mathbb{R}^{p \times p \times \hat{c} \times c}$ is the filter of the task-dependent deconvolutional mapping, and p is the spatial size of the deconvolutional filters. The deconvolutional mapping learns to generate a basis of convolutional filters within $L^{(l)}$. We then apply the tensor contraction to reconstruct each $W_t^{(l)}$ based on $D_t^{(l)}$:

$$W_t^{(l)} = D_t^{(l)} \bullet U_t^{(l)} = \sum_{k=1}^{c} D_{t,(\cdot,\cdot,k)}^{(l)} U_{t,(k,\cdot,\cdot)}^{(l)} \ , \tag{2}$$

where $U_t^{(l)}$ is a 3rd-order tensor of size $c \times c_{\text{in}} \times c_{\text{out}}$, and both subscripts $(k, \cdot, \cdot)$ and $(\cdot, \cdot, k)$ express the elements' index in the tensor. Similar to channel-wise convolution, the tensor contraction expresses the filter as a linear combination of the basis vectors, transforming $D_t^{(l)}$ to reconstruct the convolution filter $W_t^{(l)}$ by changing the size of channels. Note that $V_t^{(l)}$ and $U_t^{(l)}$ are task-specific, and serve to transform the shared knowledge bases into a model specific to task $\mathcal{Z}_t$.

Similar to the work of dynamic filter generation [Jia *et al.*, 2016; Ha *et al.*, 2017], a single operation can be employed to expand the knowledge base into a large task-specific filter. However, in our proposal, deconvolution and tensor contraction are used instead as a two-staged expansion, distinguishing between the transfer process along the spatial axis of the images and along the channels of the images. We also explore two alternate formulations of factorized transfer in the Online Appendix[1] as ablated versions of our approach.

Algorithm 1 DF-CNN (λ, $kbSize$, $transformSize$)

$L^{(1:d)} \leftarrow \text{randInit}(kbSize)$
while isAnotherTaskAvailable() **do**
  $(X_t, y_t, t) \leftarrow \text{getNextTaskTrainingData}()$
  if isNewTask(t) **then**
    $\left(V_t^{(1:d)}, U_t^{(1:d)}\right) \leftarrow \text{randomInit}(transformSize)$
  end if
  while continueBatchTraining() **do**
    // reconstruct NN from shared KB
    for $l = 1$ to d **do**
      $D_t^{(l)} \leftarrow \text{deconv}(L^{(l)}, V_t^{(l)})$
      $W_t^{(l)} \leftarrow \text{tensorDot}(D_t^{(l)}, U_t^{(l)})$
    end for
    $taskNet_t \leftarrow \text{buildNeuralNet}\left(W_t^{(1:d)}\right)$
    // update shared KB and task-specific transforms
    $X_b, y_b \leftarrow \text{drawMiniBatch}(X_t, y_t)$
    $\left(L^{(1:d)}, V_t^{(1:d)}, U_t^{(1:d)}\right)$
      $\leftarrow \text{gradientOptimizer}(X_b, y_b, taskNet_t, \lambda)$
  end while
end while

2.3 Training DF-CNN Architecture

Our learning approach must update both the shared knowledge bases and task-specific knowledge transformations while training on each task in a lifelong setting. The knowledge bases $\{L^{(l)}\}_{l=1}^d$ and task-specific knowledge transformations $\{(V_t^{(l)}, U_t^{(l)})\}_{l=1}^d$ can be trained end-to-end via gradient-based optimization. The process of training the DF-CNN is described in Algorithm 1. As hyperparameters, the algorithm requires the learning rate λ of the optimizer, the size of the knowledge bases ($kbSize \in \mathbb{R}^{3d}$), and the dimensions of the task-specific tensors $V_t^{(l)}$ and $U_t^{(l)}$ ($transformSize \in \mathbb{R}^{(4d+3d)}$).

The shared knowledge bases $\{L^{(l)}\}_{l=1}^d$ are randomly initialized prior to learning the first task. Upon receiving the labeled training data for each new task $\mathcal{Z}_t$, the task-specific knowledge transformations $\{(V_t^{(l)}, U_t^{(l)})\}_{l=1}^d$ are first initialized randomly, and then these parameters along with the knowledge bases $\{L^{(l)}\}_{l=1}^d$ are updated according to the observed training instances (X_t, y_t). During training on task

[1]The online appendix is available on the third author's website at http://www.seas.upenn.edu/~eeaton/papers/Lee2019Learning.pdf

$\mathcal{Z}_t$, the knowledge transformations for all tasks except t are held unchanged, while the shared knowledge bases are updated. Since the convolutional filters for each CNN_t are generated dynamically from the shared knowledge bases, changes to $\{L^{(l)}\}_{l=1}^{d}$ can affect previously trained networks $CNN_1, \ldots, CNN_{t-1}$ *without* retraining those networks; this phenomenon is known as reverse transfer [Ruvolo and Eaton, 2013]. Despite the lack of explicit mechanisms to prevent catastrophic forgetting [Rusu *et al.*, 2016] (i.e., severe negative reverse transfer) in these previously learned models, we show empirically that deconvolutional factorization of the model parameter space resists catastrophic forgetting and indeed exhibits positive reverse transfer—these results mirror similar results on lifelong learning of shallow linear models via factorized transfer [Ruvolo and Eaton, 2013].

3 Alternative Approaches to Deep Multi-Task and Lifelong Learning

Now that we have introduced the DF-CNN, we briefly compare it to alternative approaches for sharing inter-task knowledge in deep networks (see Fig. 2).

Explicit weight-sharing. Hard parameter sharing (HPS) is widely used for MTL and lifelong learning in neural networks [Caruana, 1993; Yim *et al.*, 2015; Ranjan *et al.*, 2017; Huang *et al.*, 2013; Bell and Renals, 2015], sharing lower network layers for feature extraction with task-specific output layers. The lowest layers can also be specific to the input domain to allow adaptation across different feature spaces. The explicit sharing of layers forces them to learn universal features for multiple tasks, with the task-specific layers mapping from the universal features to the output appropriate to each task. One variant of this architecture correlates task-specific fully-connected layers by a tensor normal distribution to learn the relations between tasks [Long *et al.*, 2017]. Moreover, there are deep lifelong learning methods that automatically modify the architecture and size of the neural network [Lu *et al.*, 2017; Yoon *et al.*, 2018]. This type of approach can add hidden units as necessary, split them into disjoint groups for different feature spaces, and consolidate groups of hidden units to avoid over-fitting and enforce transfer between tasks. Although these methods are more flexible than HPS for learning task relationships, they still explicitly share lower layers across all or partial sets of tasks. In contrast, our approach uses a shared knowledge base to flexibly relate layers across the task CNNs instead of explicitly sharing them.

Pipelined transfer. Instead of learning a monolithic network with explicit weight sharing, another approach to deep MTL and lifelong learning constructs task-specific subnetworks that learn each task, but make additional lateral connections to utilize learned representations from other tasks [Misra *et al.*, 2016; Rusu *et al.*, 2016; Gao *et al.*, 2019; Pinto and Gupta, 2017; Liu *et al.*, 2017b]. This architecture enables the network to learn and maintain task-dependent low- and high-level features, so it is more robust to handling diverse tasks and to avoiding catastrophic forgetting than explicit weight sharing. However, these approaches need to train the cross-task connections, so the size of the network increases at most quadratically with the number of tasks. The quantity of task-specific parameters in our DF-CNN is linearly proportional to the number of tasks T_{max}, so the total network size grows more slowly with T_{max} while maintaining the flexibility of sharing representations at any level.

Shared knowledge base. Approaches that transform shared knowledge to construct task-specific networks have a clear connection to our architecture. Sharable detectors for face alignment [Liu *et al.*, 2017a] have used the sparse representation of a shared basis to express a 2nd-order tensor of regression sub-networks for facial landmarks. As another example, deep MTL via tensor factorization [Yang and Hospedales, 2017] employed tensor decomposition to define a sharable basis and task-specific mappings of both 2nd-order tensors for fully-connected layers and 4th-order tensors for convolutional layers. Despite similar underlying ideas, our method exploits a deconvolution operation for mapping between the knowledge base and the task-specific filter parameters, thereby enabling the network to extract and share more abstract knowledge across tasks. The key advantage of our deconvolutional factorization is that it provides a flexible mechanism for sharing knowledge between task CNNs, allowing the architecture to flexibly reuse knowledge based on actual task relationships.

Dynamic filter generation. Our proposed idea is also related to architectures that employ dynamic filters [Jia *et al.*, 2016; Ha *et al.*, 2017], which dynamically generate task-specific CNN filters conditioned upon each task's input. The filter-generating network of these dynamic filters can serve to facilitate transfer across tasks. These works focus primarily upon learning domain-specific transformations (e.g., steerable filters) to relate rather similar tasks, whereas our approach learns shared representations to facilitate flexible transfer between tasks at multiple levels of abstraction. Additionally, the filter-generating networks typically employ a general-purpose neural network architecture (i.e., a multilayer perceptron or a convolutional network), which takes only the input space and desired output spaces into consideration. Consequently, the filter-generating networks and resulting filters may not differentiate between spatial patterns and patterns across channels—key properties of convolutional filters [Chen *et al.*, 2017; Hu *et al.*, 2018]—which is distinguished in the DF-CNN by separate deconvolution and tensor contraction steps.

4 Evaluation on Lifelong Learning Scenarios

We evaluated our proposed approach against a variety of alternative methods on lifelong learning scenarios using two visual recognition data sets. For completeness, an empirical evaluation on MTL scenarios is included in the Appendix[1].

4.1 Baseline Approaches

Our experiments compare our proposed approach to single-task learning and three baselines that are representative of the different approaches described in Section 3:

Single-task learning. STL trains a separate and isolated network for each task. STL has a clear disadvantage against transfer-based methods in the few-data or noisy-data regime.

Hard parameter sharing. HPS [Caruana, 1993] involves sharing the lower layers across tasks, with separate task-specific output layers. A useful heuristic which we follow is that all convolution layers are shared while all fully-connected layers are task-specific. HPS is expected to perform well when tasks share a common set of useful representations, but may fail when tasks are sufficiently dissimilar.

Progressive neural networks. ProgNN [Rusu *et al.*, 2016] allows each task model to build upon its predecessors. Note that their paper applied the architecture to reinforcement learning, while our evaluation focuses on supervised settings. For the construction of the ProgNNs, we reduced the dimension of the previous task models' features by a factor of two (for details, see [Rusu *et al.*, 2016, Section 2: Adapters]).

Dynamically expandable network. DEN [Yoon *et al.*, 2018] grows the size of the neural network according to the performance on the current task. The DEN adapts to a new task by a combination of selective retraining, expansion of the network to improve performance, and splitting to avoid catastrophic forgetting. We used the DEN implementation provided by the original authors.

We also evaluated the performance of tensor factorization [Yang and Hospedales, 2017] on MTL scenarios (see the Appendix). However, the method showed reasonable performance only when it was initialized by trained STL models, so we omitted the tensor factorization method from the lifelong learning experiments reported here in the main paper.

4.2 Experimental Setup

We evaluated all approaches in a lifelong learning setting, in which tasks were presented sequentially.

Data sets and tasks. We generated two lifelong learning problems using the CIFAR-100 [Krizhevsky and Hinton, 2009] and Office-Home [Venkateswara *et al.*, 2017] data sets. For CIFAR-100, we created a series of 10 image classification tasks, where each task consists of ten distinct classes. For each task, we sampled only 4% of the available CIFAR-100 data following a lifelong learning assumption of limited per-task training data [Chen and Liu, 2016], and split it into training and validation sets in the ratio 5.6:1 (170 training and 30 validation instances per task). We used all CIFAR-100 test images for the test set (1,000 instances per task). The Office-Home dataset is naturally split into multiple domains, and we focus on two of those domains: Product images and Real-World images. We created 5 image classification tasks from each of these two domains, resulting in 10 tasks with 13 image classes per task. There is no pre-specified training/validation/test split in the original data set, so we randomly split the data into those with a 60%, 10%, and 30% ratio, respectively. This results in approximately 550 training, 90 validation, and 250 test instances.

Methodology. All models were trained end-to-end on only one task at any moment, and the task was switched to the next one after every 2,000 (CIFAR-100) and 1,000 (Office-Home) training epochs, regardless of the model's convergence. The optimal hyper-parameters for each model were determined by performance on the validation sets. In addition to the lifelong learning baseline models, we compared two versions of STL: one with 3.28M (CIFAR-100) or 26.8M (Office-Home) parameters total (328k or 2.68M per individual task model) and the other with 9.35M (CIFAR-100) or 129.3M (Office-Home) parameters total (935k or 12.9M per task model). Our full DF-CNN has 7.96M (CIFAR-100) or 201.8M (Office-Home) parameters total, so we also examined a reduced-size DF-CNN with 2.8M parameters total in order to better match the total number of parameters of the baseline approaches in the CIFAR-100 experiments. Details of training process, architecture of the networks and hyper-parameters used for each data set are described in Appendix B. Note that, since the purpose of the evaluation is to compare the approaches for knowledge transfer across tasks in a lifelong learning setting, the experiment design (and consequently individual model performance) may differ from other single-task-focused publications on these methods or data sets.

Metrics. We assessed performance of our DF-CNN as well as aforementioned approaches by measuring accuracy on the held-out test set for all learned tasks at each timestep. To aggregate performance across tasks, we also computed the following metrics:

- *Peak Per-Task Accuracy:* The best test accuracy of each task during its training phase. This metric focuses on the approach's peak performance on the current task.
- *Catastrophic Forgetting Ratio:* The ratio of a task's test accuracy after training on subsequent tasks to its peak per-task accuracy. This ratio shows how much the approach can maintain its performance on older tasks.
- *Convergence:* We measure the convergence of training on a task as the number of epochs over the training set needed for the test accuracy to reach 98% of its peak per-task accuracy. The number of training epochs till convergence shows the effect of knowledge transfer from previously learned tasks.

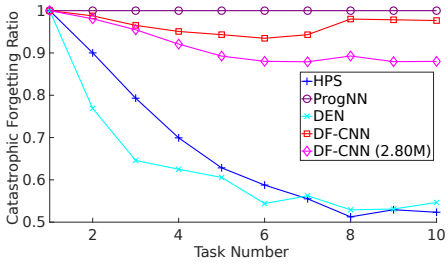
4.3 Results on Lifelong Learning

The performance of all approaches is summarized in Figures 3 (CIFAR-100) and 5 (Office-Home). For the CIFAR-100 experiments, we also depict the lifelong learning process by visualizing the dynamic test accuracy of each task model over time, averaged over 5 trials, in Fig. 4. Once a task has been learned, we repeatedly evaluated its model's performance as the system learns more tasks, exploring the effect of learning subsequent tasks on previous task models. Significant decreases in task performance after training indicate catastrophic forgetting [Kirkpatrick *et al.*, 2017]; increases in performance when training on other tasks indicate (positive) reverse transfer. The counterpart to Fig. 4 for the Office-Home experiments is in Appendix D.

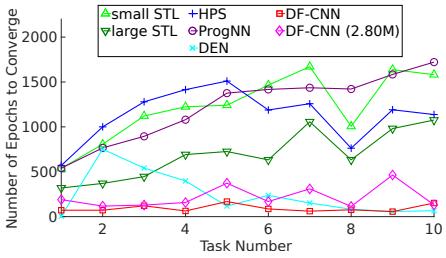
First, we can observe that HPS and DEN suffer from catastrophic forgetting as the shared layers were adapted to new tasks, as shown by the rapid decline in performance once learning on each task finishes (Fig. 4a as well as Fig. 3b and Fig. 5b). Additionally, both models could not achieve a peak per-task accuracy comparable to or better than that of STL consistently in the the CIFAR-100 experiments, and even HPS did not converge to peak per-task accuracy faster than STL. This means that the adaptation of the knowledge

Model	Peak Acc.	Time (10k sec)
STL (small)	31.2%±2.2	4.99±0.007
STL (large)	34.3%±1.1	5.46±0.005
HPS	24.7%±0.6	5.14±0.014
ProgNN	31.3%±1.1	9.21±0.019
DEN	30.2%±0.4	1.12±0.028
DF-CNN	37.2%±1.3	6.66±0.013
DF-CNN (2.8M)	36.2%±3.5	5.34±0.008

(a) Peak per-task accuracy and training time with 95% confidence intervals

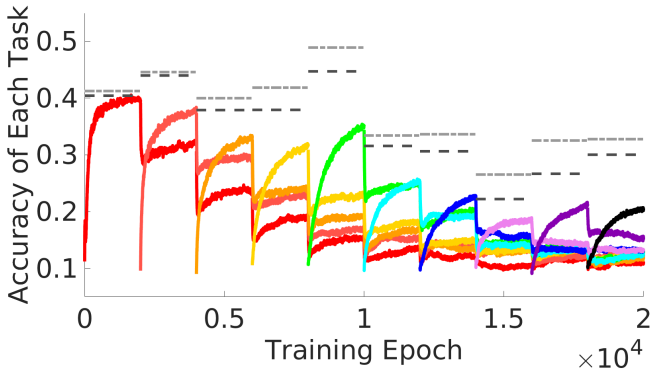


(b) Catastrophic forgetting ratio

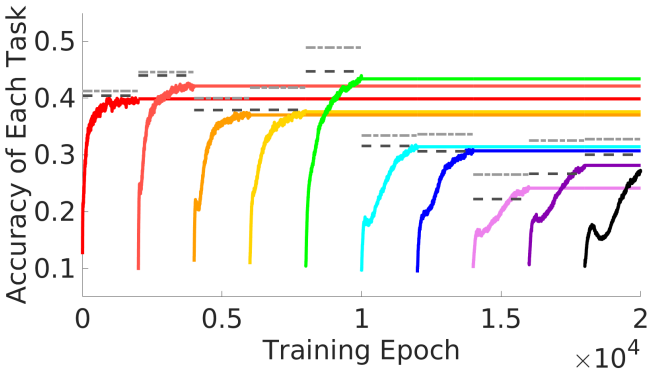


(c) Speed of convergence

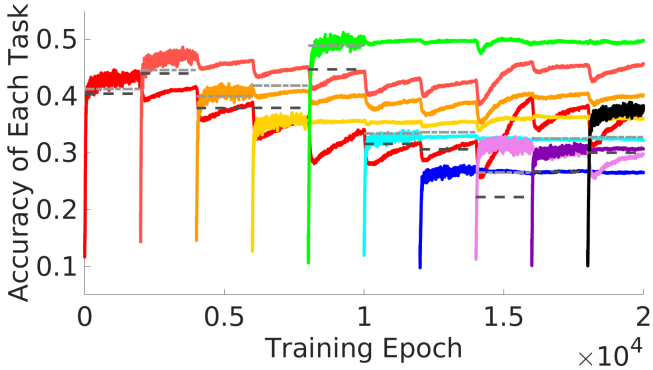
Figure 3: Performance metrics of models on CIFAR-100 lifelong learning tasks, averaged over all independent training trials. Our DF-CNN framework shows less catastrophic forgetting than HPS while achieving peak per-task accuracy better than others and being trained faster than others in terms of the speed of convergence.



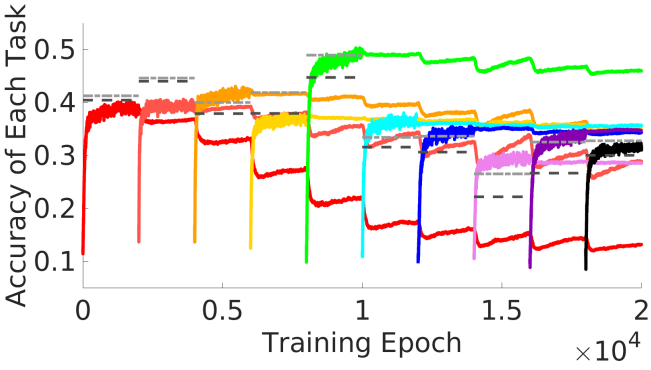
(a) Hard-Parameter Sharing (2.69M parameters total)



(b) Progressive Neural Net (3.51M parameters total)



(c) DF-CNN (7.96M parameters total)



(d) DF-CNN (2.80M parameters total)

Figure 4: Mean test accuracy in lifelong learning on CIFAR-100. Each color corresponds to one task by presentation order. Once a task has been learned (the thicker jagged part of the learning curves), the continuation of the line depicts the task model's performance as the system learns future tasks. Any significant decrease corresponds to catastrophic forgetting. The dark and light gray dotted lines show, respectively, the best test accuracy of a small STL model (3.28M parameters) and a large STL model (9.35M parameters). Note the higher performance of DF-CNN over all other methods and even the larger STL model, with relatively little forgetting overall. Best viewed in color.

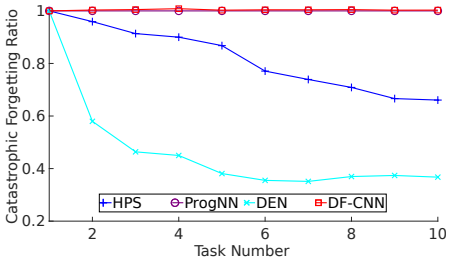
in these explicit weight-sharing models is not guaranteed to have a positive effect on training, even after the neural network shifts its attention to focus more on current tasks and forget knowledge of previous tasks.

In contrast to HPS and DEN, ProgNN is able to retain its performance on previous tasks after learning new tasks, because it is designed not to update the parameters for previous tasks. The lateral connections of ProgNN improved test accuracy in a few tasks (e.g., the 7–9th tasks of the CIFAR-100 experiment), but the benefit of transfer was marginal in comparison with the single-task learners. Moreover, ProgNN requires approximately twice as much training time as others.

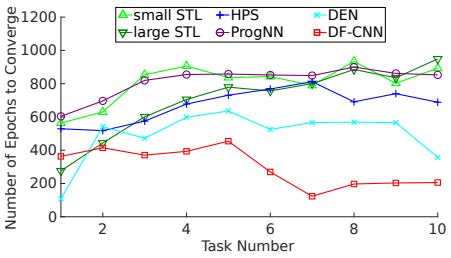
DF-CNN showed significant improvement in peak per-task accuracy over STL, HPS, and ProgNN for the CIFAR-100 ex-

Model	Peak Acc.	Time (10k sec)
STL (small)	45.5%±0.5	3.79±0.009
STL (large)	51.9%±0.9	6.09±0.005
HPS	52.0%±0.7	3.79±0.012
ProgNN	46.4%±1.0	11.7±0.003
DEN	31.6%±1.0	4.09±0.010
DF-CNN	49.1%±0.6	4.11±0.004

(a) Peak per-task accuracy and training time with 95% confidence intervals



(b) Catastrophic forgetting ratio



(c) Speed of convergence

Figure 5: Performance metrics on Office-Home lifelong learning tasks, averaged over all independent training trials. DF-CNN shows almost no catastrophic forgetting and achieves a good level of performance faster than other baselines.

periments, and peak per-task accuracy better than STL for the Office-Home experiments. Moreover, DF-CNN converges to 98% of peak per-task accuracy more than twice as fast as other approaches do. The improvement in peak per-task accuracy compared to STL, together with the improved convergence speed relative to STL, demonstrate positive knowledge transfer from older tasks within the DF-CNN framework.

The performance of previous task models within the DF-CNN deteriorated slightly as the system observed new tasks because the shared knowledge base was updated by the new tasks without consideration to previous tasks (Fig. 3b and 5b). However, the rate of losing performance is much slower than the degradation of HPS and DEN in both lifelong learning experiments, and we can even observe performance recovering over time (Fig. 4c). Especially, in the CIFAR-100 experiments (Fig. 4c), the performance of the model on the earliest tasks appears to have the most degradation, and it maintains almost constant post-training performance once the shared knowledge base became mature (e.g., the 4-10th tasks). Moreover, we can find positive reverse transfer of knowledge from new tasks to older tasks, starting at the 8th task of the CIFAR-100 experiments, which had not occurred during the training of other baseline models.

The reduced-size DF-CNN (∼2.8M trainable parameters) catastrophically lost its performance on the first task because of its reduced capacity of the shared knowledge. Even with the limited capacity of both shared knowledge and task-specific knowledge transformation, the reduced-size DF-CNN still showed improvement in accuracy, speed of convergence and robust retention of performance on previous tasks as compared to the baselines. These results support the benefit of knowledge transfer between tasks through the shared knowledge and deconvolutional mapping.

In summary, the alternative approaches to lifelong learning achieved improvement in peak per-task accuracy in comparison with STL on one of two data sets, but they slowly converged to their peak performance and showed other weaknesses such as catastrophic forgetting or large training time. Contrary to this, our DF-CNN achieved good peak per-task accuracy, fast convergence, knowledge retention and reasonable training time simultaneously. This result supports that our DF-CNN is able to extract and transfer knowledge between tasks more flexibly than competing methods while resisting catastrophic forgetting in the lifelong setting.

5 Conclusion

Deconvolutional factorization provides an effective means of knowledge transfer between CNNs in lifelong learning settings. Even though our DF-CNN architecture must train more parameters as compared to other approaches with the same base model (i.e., the individual task CNNs), it converges faster while providing comparable or better accuracy than competing approaches. Critically, the use of deconvolutional factorization and tensor contraction provides for flexible transfer between tasks, enabling the DF-CNN to resist catastrophic forgetting.

Acknowledgements

We would like to thank Sima Behpour, Jorge Mendez, Boyu Wang, and the anonymous reviewers for their helpful feedback on this work. The research presented in this paper was partially supported by the Lifelong Learning Machines program from DARPA/MTO under grant #FA8750-18-2-0117 and by AFRL grant #FA8750-16-1-0109.

References

[Baxter, 2000] J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12(1):149–198, February 2000.

[Bell and Renals, 2015] P. Bell and S. Renals. Regularization of context-dependent deep neural networks with context-independent multi-task training. In *Proc. of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4290–4294, 2015.

[Caruana, 1993] R. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proc. of the 10th International Conference on Machine Learning*, pages 41–48, 1993.

[Chen and Liu, 2016] Z. Chen and B. Liu. *Lifelong Machine Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2016.

[Chen *et al.*, 2017] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, and T. Chua. SCA-CNN: spatial and channel-wise attention in convolutional networks for image captioning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[Gao *et al.*, 2019] Y. Gao, Q. She, J. Ma, M. Zhao, W. Liu, and A. L. Yuille. NDDR-CNN: Layer-wise feature fusing in multi-task CNN by neural discriminative dimensionality reduction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3205–3214, 2019.

[Ha *et al.*, 2017] D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. In *Proc. of the International Conference on Learning Representations*, 2017.

[He *et al.*, 2016] K. He, X. Zhang, and S. Ren. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[Hu *et al.*, 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[Huang *et al.*, 2013] J. Huang, J. Li, D. Yu, L. Deng, and Y. Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Proc. of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7304–7308, 2013.

[Jia *et al.*, 2016] X. Jia, B. D. Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. *Advances in Neural Information Processing Systems 29*:667–675, 2016.

[Kirkpatrick *et al.*, 2017] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, and A. A. Rusu, et al. Overcoming catastrophic forgetting in neural networks. *Proc. of the National Academy of Sciences*, 114(13):3521–3526, 2017.

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

[Krizhevsky *et al.*, 2012] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*:1097–1105, 2012.

[Kumar and Daume, 2012] A. Kumar and H. Daume. Learning task grouping and overlap in multi-task learning. In *Proc. of the 29th International Conference on Machine Learning*, pages 1383–1390, 2012.

[LeCun *et al.*, 1998] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[Liu *et al.*, 2017a] H. Liu, J. Lu, J. Feng, and J. Zhou. Learning deep sharable and structural detectors for face alignment. *IEEE Transactions on Image Processing*, 26(4):1666–1678, 2017.

[Liu *et al.*, 2017b] P. Liu, X. Qiu, and X. Huang. Adversarial multi-task learning for text classification. In *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[Long *et al.*, 2017] M. Long, Z. Cao, J. Wang, and P. S. Yu. Learning multiple tasks with multilinear relationship networks. *Advances in Neural Information Processing Systems 30*:1594–1603, 2017.

[Lu *et al.*, 2017] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[Maurer *et al.*, 2013] A. Maurer, M. Pontil, and B. Romera-Paredes. Sparse coding for multitask and transfer learning. In *Proc. of the 30th International Conference on Machine Learning*, pages 343–351, 2013.

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

[Noh *et al.*, 2015] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. of the 2015 IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[Pinto and Gupta, 2017] L. Pinto and A. Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *Proc. of the 2017 IEEE International Conference on Robotics and Automation*, pages 2161–2168, 2017.

[Ranjan *et al.*, 2017] R. Ranjan, V. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[Rusu *et al.*, 2016] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, and K. Kavukcuoglu, et al. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[Ruvolo and Eaton, 2013] P. Ruvolo and E. Eaton. ELLA: An efficient lifelong learning algorithm. In *Proc. of the 30th International Conference on Machine Learning*, pages 507–515, 2013.

[Simonyan and Zisserman, 2015] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of the International Conference on Learning Representations*, 2015.

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

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

[Yim *et al.*, 2015] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim. Rotating your face using multi-task deep neural network. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 676–684, 2015.

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

A Alternative Formulations of Deconvolutional Factorization

The main paper framed deconvolutional factorization using a combination of deconvolution and a tensor contraction for transfer. This section describes two alternate formulations of deconvolutional factorized transfer. These alternative formulations can be thought of as ablated versions of our approach. Our evaluation of these alternatives in subsequent sections of the Appendix provides insight into how each aspect of our approach contributes to the overall transfer performance.

The two alternatives are described below:

(a) **DF-CNN.direct:** Instead of the two-staged expansion via deconvolution and tensor contraction, the *direct* alternative only uses the deconvolutional mapping to construct the task-specific convolutional filter, eliminating the tensor contraction. The resulting 3rd-order tensor $D_t^{(l)}$ has $h \times w \times (c_{in} \cdot c_{out})$ elements, which can be reshaped to the appropriate shape of the convolutional filter ($h \times w \times c_{in} \times c_{out}$).

(b) **DF-CNN.tc2:** Tensor contraction in the main paper used a 3rd-order tensor to expand the output of the deconvolutional mapping along two channels (c_{in} and c_{out}) at the same time. Alternatively, it is also possible to break the tensor contraction into two tensor contractions using 2nd-order tensors $\widetilde{U}_{t,1}^{(l)} \in \mathbb{R}^{c_1 \times c_{\text{in}}}$ and $\widetilde{U}_{t,2}^{(l)} \in \mathbb{R}^{c_2 \times c_{\text{out}}}$. This enforces the expansion related to tensor contraction to be specific to each channel.

We evaluated these two formulations on multi-task learning scenarios and compared them to the DF-CNN formulation from the main paper and the other baselines. The evaluation of each formulation can be found in Appendix C.

B Evaluation Setup

This section provide additional details on the evaluation from the main paper, as well as the experimental setup used for the MTL experiments included in this Appendix.

B.1 Training Procedures of Multi-Task and Lifelong Learning

We evaluated all baseline methods and our DF-CNN method on both multi-task and lifelong learning scenarios.

Multi-task setting: The MTL model switches its accessible task at every training epoch, alternating among the tasks to train them all simultaneously in a round-robin setup. At each epoch, a data set $(X_t, Y_t) \subseteq (\mathcal{X}_t, \mathcal{Y}_t)$ of a single task is chosen uniformly at random from the set of possible tasks $t \in \{1, \ldots, T_{max}\}$. The learning model is trained on mini-batches sampled from the data set (X_t, Y_t) without repetition. Whenever the data set becomes exhausted, another task and its data set are chosen at random and mini-batch training proceeds on the chosen task.

Lifelong learning setting: For a specific trial, the set of tasks is first permuted to obtain an ordering; we can renumber those tasks to respect the permutation ordering as $\mathcal{Z}_1, \ldots, \mathcal{Z}_{T_{max}}$. Then, for each $t \in \{1, \ldots, T_{max}\}$ sequentially, the learning model is trained on mini-batches of the task's data set $(X_t, Y_t) \subseteq (\mathcal{X}_t, \mathcal{Y}_t)$ for the fixed number of epochs. Once that fixed number of epochs is reached, learning proceeds to the next task in the sequence.

B.2 Additional Data Sets for the MTL Scenarios

For the MTL scenarios, we created visual recognition tasks from two well-known image recognition data sets: MNIST [LeCun *et al.*, 1998] and the more challenging CIFAR-10 [Krizhevsky and Hinton, 2009]. For each data set, we constructed task distributions corresponding to *distinct* one-vs-all or binary classification tasks, which we refer to as the homogeneous and heterogeneous task distributions, respectively, as Yang and Hospedales [2017] did. Each task in the homogeneous distribution involves distinguishing one class from all remaining classes. In contrast, each task in the heterogeneous distribution involves distinguishing images of two classes, which are disjoint from all other tasks. The heterogeneous distribution is more challenging than the homogeneous one because the heterogeneous tasks share no common data, unlike the homogeneous tasks.

B.3 Network Architectures and Hyper-parameters

All architectures transfer knowledge between the convolution layers of the task models, while the fully-connected layers are task-independent. For fair comparison, the configuration of the architectures (e.g., the sizes of the convolutional filters) of all models were set to be the same. Additionally, to survey the effect of the size of the trainable model on performance, some experiments employed larger STL models with more trainable parameters, or a reduced-size DF-CNN with fewer parameters. Method-specific settings (e.g., the size of the knowledge base in our approach, the tensor factorization method, etc.) were tuned independently using a validation-based grid search.

Below we describe the parameters used for each combination of data set and experiment setting.

MNIST MTL: The task networks for all methods have two layers of convolution and max-pooling followed by two fully-connected layers. The first convolution layer has 32 filters of size 5×5, and the second convolution layer has 64 filters of size 5×5. All convolution layers use ReLU activation and are followed by max-pooling layers, whose size is 2×2. After these alternating convolution and max-pooling layers, we include a dropout layer followed by two fully-connected layers with 32 and 1 output(s). The activation functions of these layers are ReLU and softmax, respectively.

The models were trained by minimizing the cross-entropy loss on sampled mini-batches of size 10. The RMSProp optimizer was used with an initial learning rate of 0.001 and exponential decay of 0.004.

CIFAR-10 MTL: The networks have four layers of convolution and two max-pooling followed by two fully-connected layers. The first and second convolution layers have 32 filters of size 3×3, and other two convolution layers have 64 filters of size 3×3. All convolution layers use ReLU activation. The second and the fourth convolution layers are followed by max-pooling layers, and the size of all max-pooling layers is 2×2. The last max-pooling layer is followed by one dropout

layer, and then two fully-connected layers with 64 and 1 output(s). The activation functions of the fully-connected layers are ReLU and softmax. A reduced DF-CNN whose number of parameters is comparable to other baselines has {24, 24, 48, 48} filters in the convolutional layers and 32 hidden units in the first fully-connected layer.

The models were trained by minimizing the cross-entropy loss on sampled mini-batches of size 20. The RMSProp optimizer was used with an initial learning rate of 0.00025 and exponential decay of 0.001.

CIFAR-100 lifelong learning: All models follow the same architectural settings of the CIFAR-10 MTL experiments. We compared lifelong learning models against two versions of STL: one that used 3.28M trainable parameters total (328K per individual CNN), and one with 9.35M parameters total (935K per CNN)—many more total parameters than in our DF-CNN. The larger STL model had convolutional layers of {48, 48, 96, 96} filters and a first fully-connected layer of 128 hidden units. A smaller DF-CNN whose number of parameters (2.8M) is comparable to the baselines used convolutional layers of only {24, 24, 48, 48} channels and a first fully-connected layer of 32 hidden units.

The models were trained to minimize the cross-entropy loss by RMSProp optimizer on sampled mini-batches of size 10. The initial learning rate is 0.0001 with an exponential decay of 0.00025.

Office-Home lifelong learning: The architecture of all models is the same as published code of the original work [Venkateswara *et al.*, 2017] (https://github.com/hemanthdv/da-hash). Because of memory saturation within the GPU, we removed the last convolutional layer from the architecture in the original work and used three fully-connected layers with 256, 64, and 13 outputs. The larger STL model had convolutional filters of {128, 256, 512, 512} channels and fully-connected layers with 1024 and 128 hidden units (there were no changes in the fully-connected layers for outputs).

The models were trained to minimize the cross-entropy loss by RMSProp optimizer on sampled mini-batches of size 16. The initial learning rate is 0.000005 with an exponential decay of 0.001.

C Evaluation on Multi-Task Scenarios

In addition to the lifelong learning experiments given in the main paper, we also evaluated the DF-CNN and baseline approaches in MTL experiments on the MNIST and CIFAR-10 data sets.

Additional baseline model for the MTL experiments: *Tensor factorization* (TF) [Yang and Hospedales, 2017] for parameter sharing has been shown to perform as good or better than the best MTL model of previous work on the MNIST, AdienceFaces, and Omniglot data sets. Our experiments used their released code for sharing the parameters of the convolution layers, and we added task-specific fully-connected layers on top of the convolution layers.

Methodology: We evaluated the two data sets at five different fractions of the available training data to form training and validation sets: 3%, 5%, 7%, 10%, and 30% for MNIST and 4%, 10%, 30%, 50%, and 70% for CIFAR-10. For each randomly selected fraction, the instances were split into training and validation sets in the ratio 5:1. For MNIST, the sizes of the test sets are 2,000 and 1,800 instances per task for the homogeneous and heterogeneous distributions, respectively. For CIFAR-10, the test data is 2,000 instances per each task, which corresponds to all of the testing portion of the data set.

MTL Results: Tables 1 and 2 show the mean accuracy of the methods for MTL on the homogeneous and heterogeneous task distributions, averaged over 10 trials. On MNIST, HPS outperformed all other baselines in the homogeneous setting. Hand-written digits share similar features between different digit classes, so explicit layer sharing in HPS gave it an advantage, which disappeared on the heterogeneous tasks. Other MTL models also transfer knowledge between tasks, but their performance was comparable to or worse than STL, except for DF-CNN.direct, which was better.

In the heterogeneous MNIST tasks, STL is unable to perform well, and HPS loses its competitive edge compared to the soft parameter sharing of TF and DF-CNN. In this more difficult task distribution, TF and DF-CNN are able to exploit their additional degrees of freedom to flexibly fit the diverse tasks. Despite extensive work and correspondence with the authors, on the homogeneous tasks we were unable to reproduce the TF results by Yang and Hospedales [2017], although TF does outperform HPS for the heterogeneous case.

The evaluation on CIFAR-10 exhibited several aspects which qualitatively differentiate it from MNIST. First and foremost, STL exhibited better relative accuracy, especially in the limited-data regime, compared to MNIST. Interestingly, however, HPS was unable to outperform STL even using 90% of the training data, which highlights the weakness of HPS in learning shared features for disparate tasks. We hypothesize that the CIFAR-10 classes have less common features than MNIST, perhaps explaining this negative transfer in HPS. In contrast to the simpler MNIST data set, DF-CNN.direct and DF-CNN outperformed all other baselines on CIFAR-10, even when it has a comparable number of parameters to HPS. This showcases the advantage of DF-CNN's flexible transfer when tasks are increasingly complex and differentiated.

TF showed weak performance on the CIFAR-10 MTL experiments, so we also trained TF by initializing its parameters to those optimized for STL (as in [Yang and Hospedales, 2017]). Using TF with such good initialization ("TF (Distillation)" in Table 2) shows improvement from STL when given a sufficient amount of training data, but the overall amount of training time (STL for initialization and TF) is more than twice as much as other methods.

DF-CNN.direct outperformed DF-CNN and DF-CNN.tc2 in most of the MTL experiments by virtue of its large number of trainable parameters. However, the performance benefit from such excessive trainable parameters is offset by the larger data and computational time requirements for training. On the other hand, DF-CNN.tc2 has the advantage over the other two DF-CNN models of having fewer parameters to learn. Its limitation, however, is that it is difficult to achieve accuracy comparable to DF-CNN in the MTL experiments.

MTL Type	Model	Model Size	Training Data Fraction				
			3%	5%	7%	10%	30%
MNIST Homogeneous	STL	1.5M	3.23%	2.16%	1.95%	1.50%	0.88%
	HPS	1.1M	**2.77**%	**1.83**%	**1.60**%	**1.28**%	0.81%
	TF	1.5M	3.49%	2.43%	2.24%	1.65%	0.93%
	DF-CNN.direct	12.9M	**2.77**%	2.02%	1.82%	1.36%	**0.75**%
	DF-CNN	2.6M	3.25%	2.24%	1.98%	1.55%	0.92%
	DF-CNN.tc2	1.5M	3.29%	2.32%	1.91%	1.58%	0.92%
MNIST Heterogeneous	STL	0.8M	0.99%	0.83%	0.72%	0.39%	0.15%
	HPS	0.6M	0.95%	0.67%	0.61%	0.40%	0.16%
	TF	0.8M	0.95%	0.72%	0.52%	0.40%	0.17%
	DF-CNN.direct	5.0M	0.82%	**0.57**%	**0.45**%	**0.34**%	**0.14**%
	DF-CNN	1.1M	**0.78**%	0.59%	0.46%	0.35%	0.17%
	DF-CNN.tc2	0.8M	0.92%	0.61%	0.57%	0.40%	0.16%

Table 1: Summary of error rates in the MNIST MTL tasks. Hard parameter sharing performed the best on homogeneous distributions. However, on heterogeneous MTL, models using soft constraints on parameter sharing achieved better accuracy.

MTL Type	Model	Model Size	Training Data Fraction				
			4%	10%	30%	50%	70%
CIFAR-10 Homogeneous	STL	3.3M	**26.0**%	**21.8**%	**17.2**%	15.3%	14.3%
	HPS	2.7M	31.5%	29.2%	20.5%	15.9%	14.5%
	TF	3.3M	38.8%	33.8%	27.9%	24.0%	20.8%
	DF-CNN	8.0M	26.5%	23.0%	**17.3**%	**14.5**%	**12.7**%
	DF-CNN	2.3M	27.9%	23.2%	17.9%	15.6%	13.8%
	DF-CNN.tc2	3.3M	29.0%	24.9%	19.4%	16.8%	15.1%
	TF (Distillation)	3.3M	28.1%	22.4%	16.5%	14.2%	12.9%
CIFAR-10 Heterogeneous	STL	1.6M	**19.6**%	**15.3**%	10.6%	9.2%	8.2%
	HPS	1.4M	22.1%	19.7%	12.4%	9.6%	8.3%
	TF	1.7M	32.7%	29.9%	20.5%	15.1%	11.9%
	DF-CNN.direct	22.8M	**19.8**%	**15.4**%	**9.8**%	**7.7**%	**6.7**%
	DF-CNN	4.0M	20.6%	16.5%	10.5%	8.1%	6.9%
	DF-CNN	1.2M	21.6%	16.6%	11.2%	8.8%	7.4%
	DF-CNN.tc2	1.6M	23.4%	18.2%	11.9%	10.1%	8.6%
	TF (Distillation)	1.7M	20.2%	14.5%	9.5%	7.7%	6.9%

Table 2: Summary of error rates in the CIFAR-10 MTL tasks. Because of the complexity of the sub-tasks of MTL, single-task learning outperformed the multi-task baselines. Our DF-CNN.direct, DF-CNN, and even a reduced-size DF-CNN with comparable numbers of trainable parameters met or exceeded the accuracy of single-task learning once given sufficient training data.

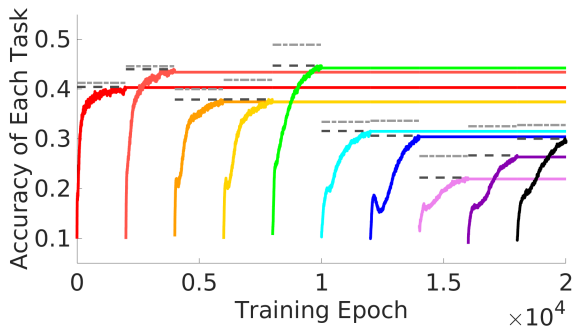
D Additional Results on Lifelong Learning

This section provides additional results from the lifelong learning experiments that were omitted from the main paper due to space limitations. Fig. 6 visualizes the average test accuracy of STL and the DF-CNN.tc2 model on the CIFAR-100 experiments; these plots augment Fig. 4 in the main paper. Fig. 7 visualizes the lifelong learning process on the Office-Home experiments, depicting the averaged test accuracy of STL, HPS, ProgNN, and DF-CNN. These plots complement the results in Fig. 5.

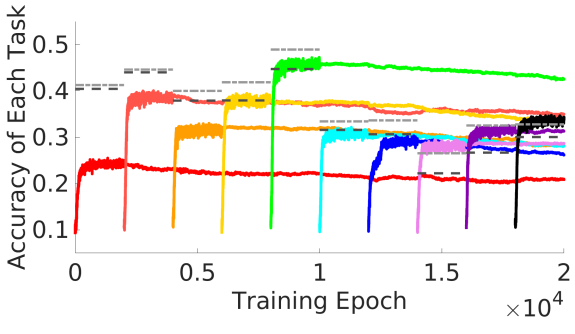
We now describe additional analysis of these results. Note that the test accuracy of STL (on both experiments) increases slowly and reaches the peak accuracy at the end of training on the task. When it is compared with the test accuracy of DF-CNN and DF-CNN.tc2, which reach their peak accuracy with relatively few epochs, this slow convergence of STL shows the benefit of transfer in learning a new task.

For the Office-Home experiments, HPS achieved improvements in peak per-task accuracy and convergence speed on the latest tasks (tasks 5 to 10). This positive transfer is possible within the Office-Home experiments because the earliest five tasks and the latest five tasks are image classification tasks with the same image classes, except for the background of the images. Therefore, explicit sharing of parameters is able to encourage positive knowledge transfer from previous to new tasks. Despite the benefit of explicit weight sharing, HPS still suffers from catastrophic forgetting.

ProgNN shows almost the same behavior as STL on Office-Home. ProgNN introduces more parameters through the lateral connections and requires more training time than STL and the other baseline methods, However, transfer via these lateral connections in ProgNN provides little benefit over STL, and no discernible benefits over DF-CNN.

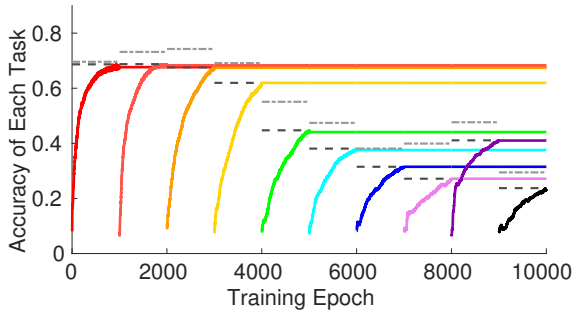


(a) Single-Task Learning (3.28M parameters total)

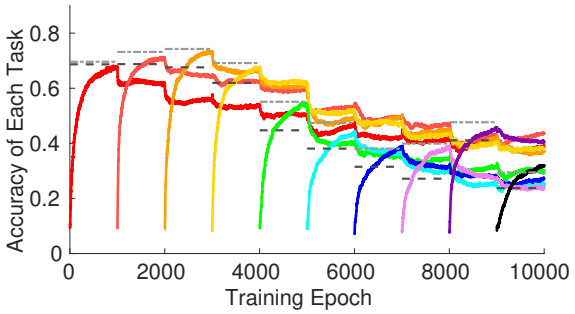


(b) DF-CNN.tc2 (3.00M parameters total)

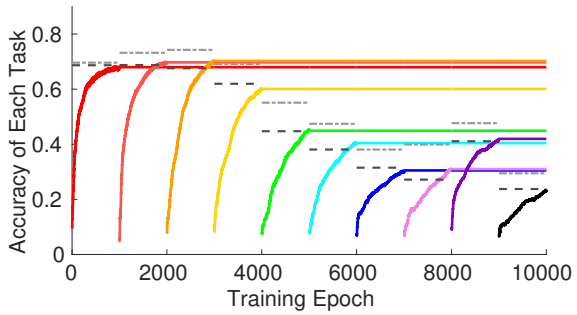
Figure 6: Mean test accuracy of STL and DF-CNN.tc2 in a lifelong learning setting on CIFAR-100. These plots augment Figure 4 in the main paper. Best viewed in color.



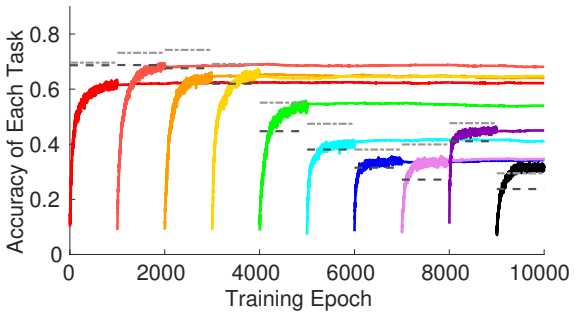
(a) Single-Task Learning (26.8M parameters total)



(b) Hard-Parameter Sharing (12.3M parameters total)



(c) Progressive Neural Net (32.7M parameters total)



(d) DF-CNN (201.8M parameters total)

Figure 7: Mean test accuracy of a STL, HPS, ProgNN, and DF-CNN in a lifelong learning setting on Office-Home. Interestingly, the DF-CNN showed negligible catastrophic forgetting while learning on new tasks. Best viewed in color.

DF-CNN required a few tasks to mature the shared knowledge base before maintaining the accuracy of previously learned tasks on the CIFAR-100 experiments, but it is able to maintain the accuracy of even the first task on the Office-Home experiments. Despite the lack of forgetting on the earliest tasks, the peak per-task accuracy of the earliest tasks is sub-optimal compared to STL. It is not until the later tasks, once the knowledge base is sufficiently mature, that the DF-CNN is able to match or exceed the performance of STL. At this time, we can see the positive effect of knowledge transfer (fast convergence and good peak per-task accuracy) on the later tasks in the DF-CNN.

Interestingly, DF-CNN.tc2 on the CIFAR-100 experiments shows qualitatively similar behavior to DF-CNN on Office-Home. Training a new task model using DF-CNN.tc2 does not negatively affect previously learned task models, but its peak performance on the earliest tasks are not optimal in comparison with STL (grey dotted line in Fig. 6b). Again, once the knowledge base matures sufficiently, for later tasks we see similar improvements due to transfer.