Network Transplanting (extended abstract)

Quanshi Zhang\textsuperscript{a}, Yu Yang\textsuperscript{b}, Qian Yu\textsuperscript{c} and Ying Nian Wu\textsuperscript{b}
\textsuperscript{a}Shanghai Jiao Tong University, \textsuperscript{b}University of California, Los Angeles, \textsuperscript{c}University of California, Berkeley

\section*{Introduction}

\hspace{1em} Besides end-to-end learning a black-box neural network, in this paper, we propose a new deep-learning methodology, \textit{i.e.} network transplanting. Instead of learning from scratch, network transplanting aims to merge several convolutional networks that are pre-trained for different categories and tasks to build a generic, distributed neural network.

Network transplanting is of special values in both theory and practice. We briefly introduce key deep-learning problems that network transplanting deals with as follows.

\section*{Future potential of learning a universal net}

Instead of learning different networks for different applications, building a universal net with a compact structure for various categories and tasks is one of ultimate objectives of AI. In spite of the gap between current algorithms and the target of learning a huge universal net, it is still meaningful for scientific explorations along this direction. Here, we list key issues of learning a universal net, which are not commonly discussed in the current literature of deep learning.

\begin{itemize}
  \item \textbf{The start-up cost w.r.t. sample collection} is also important, besides the total number of training annotations. Traditional methods usually require people to simultaneously prepare training samples for all pairs of categories and tasks before the learning begins. However, it is usually unaffordable, when there is a large number of categories and tasks. In comparison, our method enables a neural network to sequentially absorb network modules of different categories one-by-one, so the algorithm can start without all data.
  \item \textbf{Massive distributed learning & weak centralized learning}: Distributing the massive computation of learning the network into local computation centers all over the world is of great practical values. There exist numerous networks locally pre-trained for specific tasks and categories in the world. Centralized network transplanting physically merges these networks into a compact universal net with a few or even without any training samples.
\end{itemize}
Figure 1: Building a transplant net. We propose a theoretical solution to incrementally merging category modules from teacher nets into a transplant (student) net with a few or without sample annotations. The transplant net has an interpretable, modular structure. A category module, e.g., a cat module, provides cat features to different task modules. A task module, e.g., a segmentation module, serves for various categories. We show two typical operations to learn transplant nets. Blue ellipses show modules in teacher nets used for transplant. Red ellipses indicate new modules added to the transplant net. Unrelated adapters in each step are omitted for clarity.

Table 1: Comparison between network transplanting and other studies. Note that this table can only summarize mainstream in different research directions considering the huge research diversity.

| Annotation cost | Sample preparation | Interpretability | Catastrophic forgetting | Modular manipulate | Optimization |
|-----------------|--------------------|------------------|-------------------------|--------------------|--------------|
| Directly learning a multi-task net | Massive | Simultaneously prepare samples for all tasks and categories | Low | — | Not support | back prop. |
| Transfer / meta- / continual-learning | Some support weakly-supervised learning | Some learn a category/task after another | Usually low | Most algorithmically alleviate | No support | back prop. |
| Transplanting | A few or w/o annotations | Learns a category after another | High | Physically avoid | Support | back-back prop. |

Figure 2: Overview. (left) Given a teacher net and a student net, we aim to learn an adapter $h$ via distillation. We transplant the category module $f$ to the student net by using $h$ to connect $f$ and $g_S$, in order to enable the task module $g_S$ to deal with the new category $f$. Without hurting the performance of old jobs is a key issue for continual learning (Rusu et al. 2016; Schwarz et al. 2018). Our method exclusively learns the adapter to physically prevent the learning of new categories from changing existing modules.

We can summarize contributions of this study as follows. (i) We propose a new deep-learning method, network transplanting with a few or even without additional training annotations, which can be considered as a theoretical solution to three issues in Section . (ii) We develop an optimization algorithm, i.e., back-distillation, to overcome specific challenges of network transplanting. (iii) Preliminary experiments proved the effectiveness of our method. Our method significantly outperformed baselines.

Algorithm of network transplanting

**Overview**: As shown in Fig. 2, we are given a teacher net for a single or multiple tasks w.r.t. a certain category $c$. Let the category module $f$ in the bottom of the teacher net have $m$ layers, and it connects a specific task module $g_T$ in upper layers. We are also given a transplant net with a generic task module $g_S$, which has been learned for multiple categories except for the category $c$. **Modular interpretability → more controllability**: Besides the discrimination power, the interpretability is another important property of a neural network. Our transplant net clarifies the functional meaning of each intermediate network module, which makes the knowledge-transferring process more controllable.

**Bottleneck of transferring upper modules**: Most deep-learning strategies are not suitable to directly transfer pre-trained upper modules. Network transplanting just allows us to modify the lower adapter, when we transfer a pre-trained upper task module $g_S$ to a new category. It is not permitted to modify the upper module. However, it is difficult to optimize a lower adapter if the upper $g_S$ is fixed.

**Catastrophic forgetting**: Continually learning new jobs
The initial transplant net with a task module $g_s$ (before transplanting) can be learned via traditional scenario of learning from samples of some categories. We can roughly regard $g_s$ to encode generic representations for the task. Similarly, the category module $f$ extracts generic features for multiple tasks. Thus, we do not fine-tune $g_s$ or $f$ to avoid decreasing their generality.

Our goal is to transplant $f$ to $g_{S}$ by learning an adapter $h$ with parameters $\theta_h$, so that the task module $g_{S}$ can deal with the new category module $f$.

The basic idea of network transplanting is that we use the cascaded modules of $h$ and $g_{S}$ to mimic the specific task module $g_{T}$ in the teacher net. We call the transplant net a student net. Let $x$ denote the output feature of the category module $f$ given an image $I$, i.e. $x = f(I)$. $y_T$ and $y_S$ are given as outputs of the teacher net and the student net, respectively. Thus, network transplanting can be described as

$$y_T = g_T(x), \quad y_S = g_S(h(x)), \quad y_T \approx y_S \Rightarrow g_S(h(\cdot)) \approx g_T(\cdot) \quad (1)$$

**Problem of space projection & back-distillation**

It is a challenge to let an adapter $h$ project the output feature space of $f$ properly to the input feature space of $g_s$. The information bottleneck theory shows that a network selectively forgets certain space of middle-layer features and gradually focuses on discriminative features during the learning process. Thus, both the output of $f$ and the input of $g_s$ have vast forgotten space. Features in the forgotten input space of $g_s$ cannot pass most feature information through ReLU layers in $g_s$ and reach $y_S$. The forgotten output space of $f$ is referred to the space that does not contains $f$’s output features.

Vast forgotten feature spaces significantly boost difficulties of learning. Since valid input features of $g_s$ usually lie in low-dimensional manifolds, most features of the adapter fall inside the forgotten space. i.e. $g_s$ will not pass most information of input features to network outputs. Consequently, the adapter will not receive informative gradients of the loss for learning.

To learn good projections, we propose to force the gradient (also known as attention, Jacobian) of the student net to approximate that of the teacher, which is a necessary condition of $g_s(h(\cdot)) \approx g_T(\cdot)$.

$$g_s(h(\cdot)) \approx g_T(\cdot) \implies \forall J(\cdot), \quad \frac{\partial J(y_S)}{\partial x} \propto \frac{\partial J(y_T)}{\partial x} \quad (2)$$

where $J(\cdot)$ is an arbitrary function of $y$ that outputs a scalar. $\theta_h$ denote parameters of the adapter $h$. Therefore, we use the following distillation loss for back-distillation:

$$\min_{\theta_h} \text{Loss} = \mathcal{L}(y_S, y^*) + \lambda \| \frac{\partial J(y_S)}{\partial x} - \frac{\partial J(y_T)}{\partial x} \| ^2 \quad (3)$$

where $\mathcal{L}(y_S, y^*)$ is the task loss of the student net; $y^*$ denotes the ground-truth label; $\alpha$ is a scaling scalar. This formulation is similar to the Jacobian distillation in [Srinivas and Fleuret 2018]. We omit $\mathcal{L}(y_S, y^*)$, if we learn the adapter without additional training labels.

**Learning via back distillation**

It is difficult for most recent techniques, including those for Jacobian distillation, to directly optimize the above back-distillation loss. To overcome the optimization problem, we need to make gradients of $J$ agnostic with regard to feature maps. Thus, we propose two pseudo-gradients $D_S', D_T'$ to replace $D_S$, $D_T$ in the loss, respectively. The pseudo-gradients $D_S', D_T'$ follow the paradigm in Eqn. (4b).

$$D(X, \theta_h) = \mathcal{L}(y_S, y^*) + \lambda \| \frac{\partial J(y_S)}{\partial x} - \frac{\partial J(y_T)}{\partial x} \| ^2 \quad (4a)$$

$$= f'_\text{conv} \circ f'_\text{max} \circ f'_\text{pool} \circ \cdots \circ f'_\text{conv}(G_y) \quad (4b)$$

where we define $G_y = \frac{\partial J}{\partial y}$. Just like in Eqn. (2), we assume $g_s(h(\cdot)) \approx g_T(\cdot) \Rightarrow D_S' \approx D_T'$. $f'_i \circ f'_j(\cdot) \overset{\text{def}}{=} f'_j(f'_i(\cdot))$, each $f'$ is the derivative of the layer function $f$ for back-propagation. $X$ denotes a set of feature maps of all middle layers, and $x^{(m)} \in X$ is the feature map of the $m$-th layer. In Eqn. (4b), we revise layer-wise operations to the computation of gradients, in order to make gradients $D'$ agnostic with regard to $X$.

In this way, we conduct the back-distillation algorithm by $\min_{\theta_h} \text{Loss} = \mathcal{L}(y_S, y^*) + \lambda \| D_S' - D_T' \|^2$. The distillation loss can be optimized by propagating gradients of gradient maps to the upper layers, and we consider this as back-back-propagation. Tables 2 and 3 have exhibited the superior performance of the back-back-propagation.

**Experiments**

To simplify the story, we limit our attention to testing network transplanting operations. We do not discuss other related operations, e.g. the fine-tuning of category and task modules and the case in Fig. 1(a), which can be solved via traditional learning strategies.

We designed three experiments to evaluate the proposed method. In Experiment 1, we learned toy transplant nets by inserting adapters between middle layers of pre-trained CNNs. Then, Experiments 2 and 3 were designed considering the real application of learning a transplant net with two task modules (i.e. modules for object classification and segmentation) and multiple category modules. As shown in Fig. 3(b,d), we can divide the entire network-transplanting procedure into an operation sequence of transplanting category modules to the segmentation module. Therefore, we separately conducted the
two sequences of transplanting operations in Experiments 2 and 3 for more convincing results.

We compared our back-distillation method (namely back-distill) with two baselines. All baselines exclusively learned the adapter without fine-tuning the task module for fair comparisons. The first baseline only optimized the task loss \( L(y_s, y_t) \) without distillation, namely direct-learn. The second baseline is the traditional distillation, namely distill, where the distillation loss is CrossEntropy(y\(_s\), y\(_t\)). The distillation was applied to outputs of task modules g\(_s\) and g\(_t\), because except for outputs, other layers in g\(_s\) and g\(_t\) did not produce features on similar manifolds. We tested the distill method in object segmentation, because unlike single-category classification, segmentation outputs had correlations between soft output labels.

### Exp. 1: Adding adapters to pre-trained CNNs

In this experiment, we conducted a toy test, i.e. inserting and learning an adapter between a category module and a task module to test network transplanting. Here, we only considered networks with VGG-16 structures for single-category classification.

In Table 2, we compared our back-distill method with the direct-learn baseline when we inserted an adapter with a single conv-layer and when we inserted an adapter with three conv-layers. Table 2(left) shows that compared to the 9.79%–38.76% error rates of the direct-learn baseline, our back-distill method yielded a significant lower classification error (1.55%–1.75%). Even without any training samples, our method still outperformed the direct-learn method with 100 training samples.

### Exp. 3: Operation sequences of transplanting category modules to the segmentation module

In this experiment, we evaluated the performance of transplanting category modules to the segmentation module. Five FCNs were strongly supervised using all training samples for single-category segmentation. We considered the segmentation module of the dog as a generic one. We transplanted category modules of other four mammal categories to this task module.

Table 3 compares pixel-level segmentation accuracy between our method and the direct-learn baseline. We tested our method with a few (10–100) training samples. Our method exhibited 10%–12% higher accuracy than the direct-learn baseline.

### Conclusions and discussion

In this paper, we focused on a new task, i.e. merging pre-trained teacher nets into a generic, modular transplant net with a few or even without training annotations. We discussed the importance and core challenges of this task. We developed the back-distillation algorithm as a theoretical solution to the challenging space-projection problem. The back-distillation strategy significantly decreases the demand for training samples. Experimental results demonstrated the superior efficiency of our method. Our method without any training samples even outperformed the baseline with 100 training samples, as shown in Table 2(left).

### References

[Rusu et al. 2016] Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. In arXiv:1606.04671.

[Schwarz et al. 2018] Schwarz, J.; Luketina, J.; Czarnecki, W. M.; Grabska-Barwinska, A.; Teh, Y. W.; Pascanu, R.; and Hadsell, R. 2018. Progress & compress: A scalable framework for continual learning. In arXiv:1805.06370.

[Srinivas and Fleuret 2018] Srinivas, S., and Fleuret, F. 2018. Knowledge transfer with jacobian matching. In ICML.