Supplementary Materials of Zoo-Tuning: Adaptive Transfer from A Zoo of Models

Yang Shu [*1] **Zhi Kou** [*1] **Zhangjie Cao** [1] **Jianmin Wang** [1] **Mingsheng Long** [1]

1. Experiment Details

In this section, we supplement more experiment details for reproducing the experiment results.

1.1. Implementation Details in Reinforcement Learning

In the experiments of transfer learning in reinforcement learning, we use the network architecture in Data-Efficient Rainbow, which consists of 2 convolutional layers: 32 filters of size 5×5 with the stride of 5 and 64 filters of size 5×5 with the stride of 5, followed by a flatten layer and 2 noisy linear layer with the hidden size of 256. We train two models with this architecture on the Seaquest and Riverraid games, respectively, as our source models. We train for 1×10^5 steps with the same hyper-parameters as those in Data-Efficient Rainbow. We then perform Zoo-Tuning on the first 2 layers for each target task. To be specific, we replace the first 2 layers in the original architecture with adaptive aggregation (AdaAgg) layers, which learn to adaptively aggregate the parameters of these layers in source models to form the corresponding layers of the target model. The last 2 layers of the architecture are not modified, which are randomly initialized and trained together with the previous AdaAgg layers. During the test stage, we calculate an average of 10 episodes as the final result, and report the mean and variance of results with 5 different random seeds.

1.2. Implementation Details in Image Classification and Facial Landmark Detection

In the experiments of classification and facial landmark detection, we use 5 pretrained models trained from various datasets and tasks as stated in the main text. Each pretrained model consists of a ResNet-50 backbone and several task-specific head layers. For each target task, we perform Zoo-Tuning on the ResNet-50 backbone. We modify each convolutional layer in the backbone with an AdaAgg layer to aggregate knowledge from all 5 pretrained models. We do not perform channel alignment on top 1×1 convolutional layers in the backbone, which preserves the accuracy and further reduces the parameters and computational complexity of our method. We keep other layers such as pooling layers, batch normalization layers, and activation layers unchanged. For each batch normalization layer in the backbone, we initialize its learnable linear transformation parameters with the average of the corresponding parameters in 5 pretrained models, which gives the layer a smooth warm-up.

Based on the modified backbone with AdaAgg layers, we add a task-specific head for each task. The task-specific head is randomly initialized and trained together with the AdaAgg backbone. For classification tasks, the head is composed of a linear layer. For facial landmark detection, the head is composed of three 4×4 transpose convolution layers with the stride of 2, and a 1×1 convolution layer. For each target task, we run experiments for 3 times and report the average results of the 3 runs.

1.3. Computing Infrastructure

For reinforcement learning experiments, we implement all the methods based on PyTorch 1.5.0, torchvison 0.6.0, and CUDA 10.2 libraries. For the other two computer vision experiments, we use PyTorch 1.1.0, torchvison 0.3.0, and CUDA 10.0 libraries for the classification tasks. We use PyTorch 1.0.0, torchvison 0.2.1, and CUDA 9.0 libraries for facial landmark detection tasks. We use a machine with 32 CPUs, 256 GB memory, and one NVIDIA TITAN X GPU.

2. Experiment Results

In this section, we supplement more experiment results.

2.1. More Transfer Learning Baselines

In this section, we design two more transfer learning methods for transferring from a zoo of models. The first method, called Concat-Ensemble-Distill (**CED**), performs a knowledge distillation from model ensembles but uses the concatenation of their features instead of direct averaging their

[*]Equal contribution [1]School of Software, BNRist, Tsinghua University, Beijing, China. E-mail: Yang Shu (shuy18@mails.tsinghua.edu.cn). Correspondence to: Mingsheng Long <mingsheng@tsinghua.edu.cn>.

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

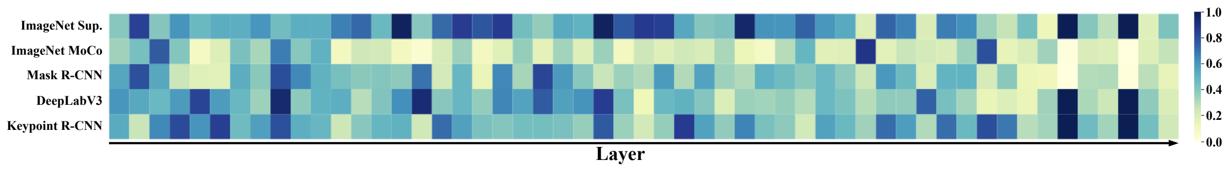

Figure 1. Visualization of gating values in each layer of source models. We use the 5 vision pretrained models as source models and use the facial landmark detection dataset COFW as the target task. From the left to right is from the bottom layer to the top layer. The darker color means a higher gating value.

predictions. Specifically, it adopts a network architecture consisting of a shared backbone with multi-heads to distill the high-level features from each source model in the zoo. Then it uses the concatenation of all these features from multi-heads for the target task. The second method, called Feature Extraction (**FE**), extracts the features (specifically, features of the penultimate layers) from all the source models in the model zoo and concatenates them to serve as the new features for the target data. It then trains a target classifier with the concatenated features. The two methods are extensions of feature-level transfer learning methods to the model zoo transfer setting. We perform two variants for each method based on whether the source models are firstly fine-tuned with target data (+Finetuned). Note that fine-tuning all source models causes high computation costs.

Table 1. Comparison with the proposed baselines.

MODEL	CIFAR	COCO	INDOORS
CED	81.84	82.19	73.15
CED+FINETUNED	81.91	82.01	74.57
FE	80.01	56.52	66.15
FE+FINETUNED	83.00	83.34	74.66
ZOO-TUNING	**83.77**	**84.91**	**75.39**

As shown in Table 1, Zoo-Tuning still outperforms these baselines. Compared with them, Zoo-Tuning has two main advantages. (1) It adaptively leverages the knowledge in pretrained models to facilitate target learning tasks instead of using the fixed features. (2) These baselines only consider high-level features while Zoo-Tuning enables aggregation in all layers, leading to a deeper transfer of knowledge.

2.2. Visualization of Gating Values

We also visualize the gating values of each layer in each pretrained model learned by Zoo-Tuning on the facial landmark detection benchmark COFW, as shown in Figure 1.

Different layers in different source models have diverse gating values, which indicates that different layers and source pretrained models have different underlying influences on the target task. Note that this task is more complicated than classification, as the relationship between source and target tasks is unclear, and there is little prior knowledge. We also find that the single model performance cannot necessarily determine its importance when transferring from multiple models under this situation. For example, though fine-tuning from the MoCo pretrained model performs better than other models, it does not have dominant gating values in Zoo-Tuning. This makes transferring from a zoo of models difficult, especially on more complex downstream tasks, and shows the advantage of using Zoo-Tuning, which learns to adaptively aggregate the source models without handcrafted designs.

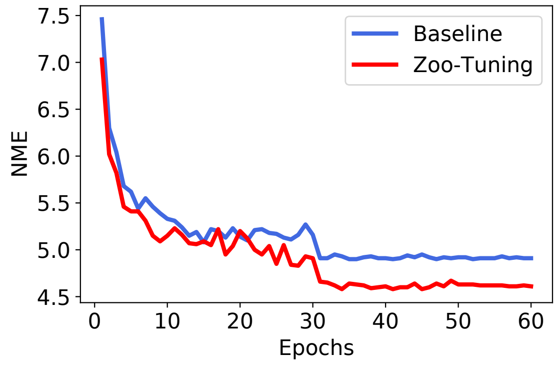

Figure 2. The normalized mean error of the fine-tuning baseline and Zoo-Tuning on WFLW dataset.

2.3. Training Curves

We plot the curves of the normalized mean error (NME) on target test data while training with the fine-tuning baseline and Zoo-Tuning methods on the WFLW dataset in Figure 2. We can observe that Zoo-Tuning shows a similar convergence speed compared with fine-tuning from one model but consistently achieves better generalization results.