Amalgamating Knowledge towards Comprehensive Classification

Chengchao Shen,[1] Xinchao Wang,[2] Jie Song,[1] Li Sun,[1] Mingli Song[1]
[1]Zhejiang University, [2]Stevens Institute of Technology
{chengchaoshen,sjie,lsun,brooksong}@zju.edu.cn, xinchao.wang@stevens.edu

Abstract

With the rapid development of deep learning, there have been an unprecedentedly large number of trained deep network models available online. Reusing such trained models can significantly reduce the cost of training the new models from scratch, if not infeasible at all as the annotations used for the training original networks are often unavailable to public. We propose in this paper to study a new model-reusing task, which we term as *knowledge amalgamation*. Given multiple trained teacher networks, each of which specializes in a different classification problem, the goal of knowledge amalgamation is to learn a lightweight student model capable of handling the comprehensive classification. We assume no other annotations except the outputs from the teacher models are available, and thus focus on extracting and amalgamating knowledge from the multiple teachers. To this end, we propose a pilot two-step strategy to tackle the knowledge amalgamation task, by learning first the compact feature representations from teachers and then the network parameters in a layer-wise manner so as to build the student model. We apply this approach to four public datasets and obtain very encouraging results: even without any human annotation, the obtained student model is competent to handle the comprehensive classification task and in most cases outperforms the teachers in individual sub-tasks.

Introduction

Recent years have witnessed the unprecedented progress of deep learning. Many deep models, such as AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG (Simonyan and Zisserman 2014), GoogLeNet (Szegedy et al. 2015), and ResNet (He et al. 2016), have been proposed and applied to almost every single computer vision task, yielding state-of-the-art performances. However, the promising results come with the costs of the huge amount of annotations required and the resource-consuming training process, which may take up to weeks on multiple GPUs.

Yet encouragingly, many researchers have published online their trained models, tailored for various tasks like classification, detection, and segmentation. Reusing these trained models, either for the primary task or novel ones, can significantly reduce the effort to retrain the new models from scratch, which is in many cases not feasible at all as the annotations used to train the original models may not be publicly available.

To this end, researchers have started to look at prospective approaches to reuse trained deep models. For example, (Buciluǎ, Caruana, and Niculescu-Mizil 2006) proposes a model compression approach that trains a neural network using the predictions of an ensemble of heterogeneous models trained a priori. (Hinton, Vinyals, and Dean 2015) introduces the concept of Knowledge Distillation (KD), whose goal is to derive a compact student model that imitates the teacher by learning from the teacher's outputs. (Romero et al. 2014) makes one step further by learning a student model that is deeper and thinner than the teacher so as to improve the performance. These model-reusing approaches, despite their very promising results, focus on tackling the same task as the trained teacher models.

In this paper, we propose to investigate a new model-reusing task, which we term as *knowledge amalgamation*. Given multiple trained teacher models, each of which specializes in a different classification problem, knowledge amalgamation aims to learn a compact student model capable of handling the comprehensive classification problem. In other words, the classification problem addressed by the student is the superset of those by all the teachers. For example, say we have two teacher classifiers, the first one classifies sedan cars and SUVs while the second classifies pickups and vans. The student model is expected to be able to classify all the four types of cars simultaneously. Note that, here we assume *no human annotations* and only the predictions from the teacher models are available.

The proposed knowledge amalgamation task is, to our best knowledge, both novel and valuable. It is novel because, in contrast to prior model-reusing tasks that restrict the student model to handle the same problem as the teachers do, knowledge amalgamation learns the "super knowledge" covering the specialties from all the teachers. It is valuable because, it allows reusing the trained models, without any human annotation, to learn a compact student model that approximates or even outperforms the teacher models.

We also propose a pilot strategy towards solving the knowledge amalgamation task. Our approach comprises two steps, feature amalgamation and parameter learning. The feature amalgamation step first extracts features of the multiple teachers, obtained by feeding input samples to the teachers, and then compresses the stacked features into a compact and discriminative set. The obtained set of features are then used as the supervision information for learning the network parameters in a layer-wise manner in the parameter learning step. This strategy turns out to effective, as the learned compact student model, without any human-labeled annotations, is capable of handling the comprehensive classification task and achieves performances superior to those of the teachers on individual sub-tasks.

Our contribution is thus introducing the knowledge amalgamation task and a simple yet competent approach towards solving it, as demonstrated on several datasets. We would like to promote, via the introduction of the knowledge amalgamation task, that researchers should look at reusing trained models to novel tasks, in which way the annotation-, training-, and running-cost can be dramatically reduced.

Related Work

Knowledge Distillation Hinton et al. (Hinton, Vinyals, and Dean 2015) proposes a teacher-student paradigm where a smaller student network imitates the soft prediction of the large teacher ones. This method introduces a temperature concept to highlight the similarities among categories, benefiting the learning of student network.

Following (Hinton, Vinyals, and Dean 2015), FitNet (Romero et al. 2014) adopts a deeper but thinner student network to learn the knowledge of a teacher. To improve the optimization of deep student network, not only the soft prediction but also the intermediate representation are taken into consideration to supervise the training of the student network. Specifically, the intermediate representation includes both the feature maps from the convolutional layers and the feature vectors from the intermediate fully connected layers.

DK^{2}PNet (Wang, Deng, and Wang 2016) introduces a dominant convolutional kernel method to compress convolutional layers. AT (Zagoruyko and Komodakis 2017) exploits two types of spatial attention maps, activation-based and gradient-based from teacher network, to guide the learning of student network. NST (Huang and Wang 2017) regards the knowledge distillation as a distribution matching problem, where the student network is trained to match the distribution of intermediate representation with that of the teacher network.

The knowledge distillation task, despite its solid motivation and proven significance, has a major goal-wise limitation. It aims at learning a student model only from one teacher and thus expects the student to master only the specialization from that teacher. By contrast, the proposed knowledge amalgamation task enables the student to learn from multiple teacher models and amalgamates all of their knowledge so as to handle the "super" task.

In addition, the work of (Huang and Wang 2017) demonstrates that when the number of classes is large, the variants of knowledge distillation approaches (Romero et al. 2014; Wang, Deng, and Wang 2016; Zagoruyko and Komodakis 2017; Huang and Wang 2017) yield worse classification performances than the original version of (Hinton, Vinyals, and Dean 2015). Such variants thus do not fit our purpose, as we aim to amalgamate from multiple teachers with potentially large number of classes shown in our experiments.

Transfer Learning Transfer learning is proposed to transfer knowledge from source domain to target domain so as to reduce the demand for labeled data on the target domain (Pan, Yang, and others 2010). It can be roughly categorized into cross-domain (Long et al. 2013; Huang, Huang, and Krähenbühl 2018; Hu, Lu, and Tan 2015; Ding et al. 2018) and cross-task transfer learning (Hong et al. 2016; Cui et al. 2018; Gholami, Rudovic, and Pavlovic 2017). More specifically, cross-domain transfer learning aims to transfer knowledge among datasets with different data distributions but the same categories. And cross-task transfer learning tries to alleviate the deficit of data for categories on the target task by transferring knowledge from other categories on the source task. However, knowledge amalgamation focuses on amalgamating the existing models with unlabeled data to obtain a versatile neural network.

Cross-modal transfer learning (Huang, Peng, and Yuan 2017; Gupta, Hoffman, and Malik 2015; Xu et al. 2018) transfers knowledge among different modalities to improve the performance on the target modality with the same categories, which is different from knowledge amalgamation. FMR (Yang et al. 2017) is proposed to introduce extra features into Convolutional Neural Network (CNN) to improve the performance on the original classification task. It is different from knowledge amalgamation, which amalgamates multiple teachers for the comprehensive classification task instead of the original one.

Knowledge Amalgamation Task

We give the definition of the knowledge amalgamation task as follows. Assume that we are given N teacher models $\{t_i\}_{i=1}^N$ trained a priori, each of which implements a specific classification problem. Let $\mathcal{D}_i$ denote the set of classes handled by model t_i. Without loss of generality, we assume $\mathcal{D}_i \neq \mathcal{D}_j, \forall i \neq j$. In other words, for any pair of models t_i and t_j, we assume they classify different sets of classes. The goal of knowledge amalgamation is to derive a compact student model that is able to conduct the comprehensive classification task, in other words, to be able to simultaneously classify all the classes in $\mathcal{D} = \cup_{i=1}^N D_i$.

The student model is thus expected to be more powerful as it handles the "super" classification problem, and meanwhile more portable as it is smaller and more resource-efficient than the ensemble of the teacher models.

The Proposed Method

Towards solving the proposed knowledge amalgamation task, a simple yet effective pilot approach is introduced. In what follows, we first give an overview of the method, then detail the two steps, and finally show the training objective.

Overview

The pilot approach assumes that, for the time being, the teacher models share the same network architecture. This assumption might be arguably strong but it does hold in

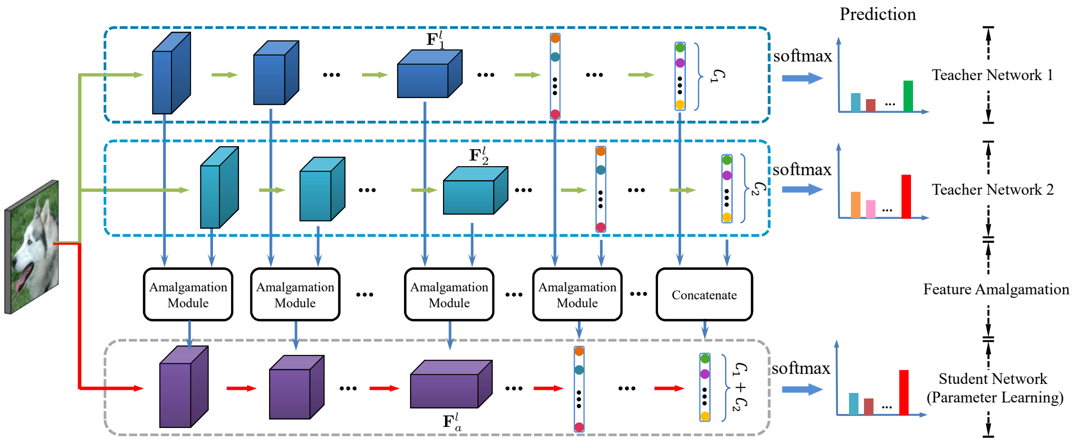


Figure 1: The overall workflow of the proposed approach in the two-teacher case. It consists of two steps, feature amalgamation and parameter learning. The feature amalgamation step, as depicted by the black block, computes the features of the student model from those of the teachers. For example, the feature maps F_1^l and F_2^l from the teachers are fed into the feature amalgamation module to obtain the compact feature map F_a^l of the student. The parameter learning step, as depicted by the red arrows, computes the network parameters of the student network, given the features amalgamated from the first step.

many cases especially on large-scale datasets, where multiple models are trained on subsets of the classes. We hope this proposed approach could serve as a baseline method towards solving the knowledge amalgamation, based on which further research could improve. Specifically, the proposed approach follows a two-step procedure: feature amalgamation and parameter learning. In the feature amalgamation step, we derive a set of learned features for each layer of each teacher model, obtained by feeding input samples to each such teacher. The features from the same layer across different teachers are then concatenated and further compressed into a compact set, which is treated as the corresponding feature map for the student. In the parameter learning step, we treat the obtained feature sets as the supervision information for learning the parameters of the student network. This is achieved by looking at the feature sets from two consecutive layers and then computing the corresponding network parameters between them.

The overall process of the knowledge amalgamation, in the case of two teacher models, is shown in Figure 1. The details of the feature amalgamation step and the parameter learning step are given as follows.

Feature Amalgamation

We start by discussing first the feature amalgamation from two teacher models, and then two possible solutions for multi-teacher feature amalgamation, followed by the score-vector amalgamation.

Amalgamation from Two Teacher Models We first consider the case of feature amalgamation from two teacher models. A straightforward amalgamation approach would be to directly concatenate the feature sets, obtained by feeding inputs to the teacher models, on the same layer of the two teachers. In this way, however, the obtained student model would be very cumbersome: the student would be four times as large as the teachers, as between the two layers we will have twice as many the inputs and twice the outputs.

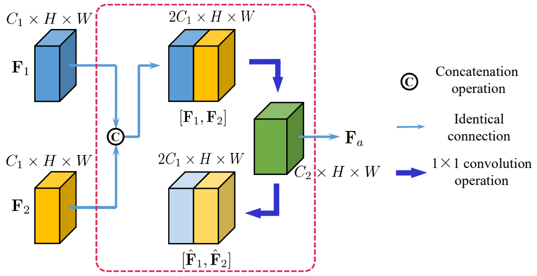


Figure 2: Feature amalgamation from two teacher models. F_1 and F_2 respectively denote the features from two teacher models, each of which has C_1 channels. They are further encoded to a compact feature map F_a of C_2 channels, where $C_1 < C_2 < 2C_1$. The whole module is trained using an auto-encoder architecture that enforces F_a to preserve the information of input features.

Recall that the goal of amalgamation is to obtain a compact model that is more resource-efficient and thus handy to deploy. To this end, we apply an auto-encoder architecture that compresses the concatenated features from the two teachers, as depicted in Figure 2. We choose auto-encoder because it reduces the size of the feature maps and meanwhile preserves the critical information, as the compact features approximately reconstruct the original concatenated one.

A convolution kernel of 1×1 that has demonstrated its success (Szegedy et al. 2015; He et al. 2016) in many state-of-the-art CNN architectures is adopted to implement the auto-encoder. This kernel is used to reduce the channel number of feature maps and the computation load, and meanwhile preserves the size of the receptive field. We write

$$F_{a,c} = \sum_{c'=1}^{C_{in}} w_{c,c'} \cdot F_{c'}, \tag{1}$$

where $w_{c,c'}$ denotes the c'-th channel weight of the 1×1 convolution kernel, $c \in \{1, \ldots, C_{out}\}$, $F_{c'}$ denotes the c'-th channel of the input feature map F, $F_{a,c}$ denotes the c-th channel of the output feature map F_a, C_{in} and C_{out} denote the channel numbers of input feature map F and output feature map F_a, respectively. Note that, we have $C_{out} < C_{in}$ due to the feature compression.

Amalgamation from Multiple Teacher Models Two ways to amalgamate features from more than two teacher models are proposed as follows.

- **Incremental Feature Amalgamation (IFA)**: we conduct amalgamation in a progressive manner, by each time amalgamating two sets of feature maps, as depicted in Figure 3 (a).
- **Direct Feature Amalgamation (DFA)**: we directly amalgamate feature maps from multiple teachers, as depicted in Figure 3 (b). Similar to the two-teacher case, the feature sets are concatenated into one and then passed through an auto-encoder to obtain a compressed feature set.

Although the architecture of DFA is more intuitively straightforward, IFA is in fact easier to generalize as the same auto-encoder can be repetitively adopted and thus extended to arbitrary number of teachers, while DFA needs to retrain the whole auto-encoder when a new teacher is added.

Amalgamation of Score Vectors The score vector can be regarded as the response scores of the categories to be classified. For disjoint teacher models that handle non-overlapping classes, we directly concatenate their score vectors as the amalgamated one, and use the amalgamated score vector as the target for the student, as shown in Figure 1. In fact, the same strategy can also be used for teachers with overlapping classes, in which case we treat the multiple entries of the overlapping categories in the concatenate score vector as different classes during training but as the same one at test time. We also test the results of preserving only one entry and removing the rest entries of each overlapping category, which can be found in our supplementary material.

Parameter Learning

In the parameter learning stage, the obtained compact feature maps in consecutive layers are treated as the supervision information to learn the weights in between. Specifically, this is achieved by first learning the weights in a layer-wise manner and then fine-tuning all the layers jointly. To facilitate the layer-wise training, a feature adaptation strategy is adopted.

Layer-wise Parameter Learning Let F_a^l and F_a^{l-1} respectively denote the compact features of the l-th and $(l-1)$-th layer in the student network obtained by feature amalgamation. In the layer-wise parameter learning step, F_a^{l-1} is fed as input and goes through a series of operations including pooling, activation and convolution to approximate F_a^l. We write

$$\hat{F}_a^l = \mathrm{conv}(\mathrm{pool}(\mathrm{activation}(F_a^{l-1}))), \tag{2}$$

where $\hat{F}_a^l$ corresponds to the estimated features in the l-th layer. Since the pooling layer and activation layer have no parameters, $\mathrm{pool}(\mathrm{activation}(F_a^{l-1}))$ is deterministic for a given F_a^{l-1}. Therefore, the goal of the layer-wise learning stage is to obtain the weights of the convolutional layer. This leads to a linear optimization problem, which is much easier to be solved than optimizing all the parameters of the network jointly.

Algorithm 1 Knowledge Amalgamation from Multiple Teachers

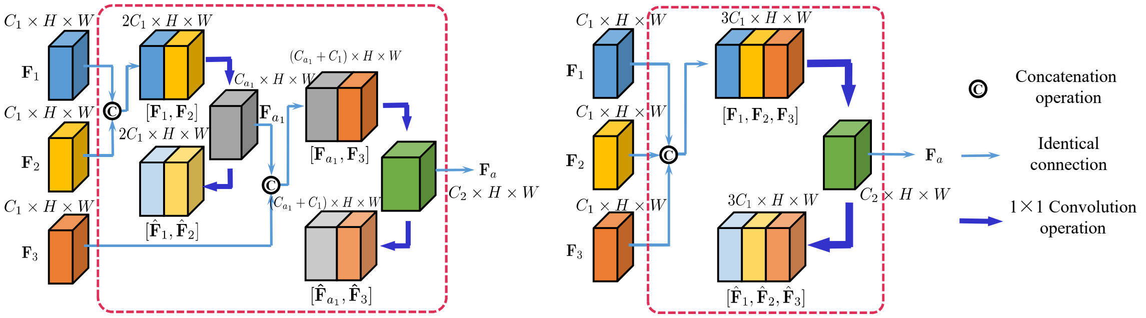
```
Input: N trained teacher models $T = \{t_i\}_{i=1}^N$, and unlabeled samples $\mathcal{D} = \{x_k\}_{k=1}^K$.
Output: The parameters of the student model $S$: $\{\Theta_l\}_{l=1}^L$
for $l = 1$ to $L - 1$ do
    Obtain $\{F_i^l\}_{i=1}^N$ from $\{t_i\}_{i=1}^N$ with $\mathcal{D}$;
    Amalgamate feature maps $\{F_i^l\}_{i=1}^N$ to obtain $F_a^l$;
    Compute the output of $S$ from $l$-th layer: $\hat{F}_a^l$;
    Compute the loss $\mathcal{L}_{\mathrm{PL}}^l$ according to Eq. 5;
    Update the parameters $\Theta_l$ using SGD;
end for
Obtain $L$-th layer score vectors $\{F_i^L\}_{i=1}^N$ from $\{t_i\}_{i=1}^N$;
Obtain score vector for $S$: $F_a^L \leftarrow \mathrm{concat}(\{F_i^L\}_{i=1}^N)$;
Compute the output of $S$ from $L$-th layer: $\hat{F}_a^L$;
Compute the loss $\mathcal{L}_{\mathrm{PL}}^L$ according to Eq. 5;
Jointly update the parameters $\{\Theta_l\}_{l=1}^L$ using SGD.
```

Feature Adaption A straightforward way to compute the weights of the convolutional layers is to solve directly the linear transformation that maps $\mathrm{pool}(\mathrm{activation}(F_a^{l-1}))$ to F_a^l. This however turns out to be sub-optimal, as F_a^{l-1} is obtained directly from feature amalgamation and is fixed, meaning that F_a^{l-1} is not adjustable to the non-parametric operations like pooling and activation. As such, the non-parametric layers may remove some discriminant information from F_a^{l-1}, making the parameter learning troublesome. For example, the ReLU layer will suppress all the non-positive values from the feature map F_a^{l-1}, which might be the critical information to be passed to F_a^l.

To facilitate the learning, we introduce a Feature Adaption Module (FAM) to the layer-wise parameter learning stage, and transform the features into a form that can be well adaptive to other non-parametric layers. Specifically, a 1×1 convolution operation is adopted to implement FAM. We write

$$\hat{F}_a^l = \mathrm{conv}(\mathrm{pool}(\mathrm{activation}(\mathrm{FAM}(F_a^{l-1})))). \tag{3}$$

Joint Parameter Learning The layer-wise learning yields errors in the optimization stage, which accumulates layer by layer across the whole deep network. To remedy this, after the layer-wise learning, we look at all the parameters simultaneously and train them end to end, in which way the convolutional layers adopt to each other better.


$C_1 \times H \times W$
$\mathbf{F}_1$
$2C_1 \times H \times W$
$[\mathbf{F}_1, \mathbf{F}_2]$
$C_{a_1} \times H \times W$
$\mathbf{F}_{a_1}$
$(C_{a_1}+C_1) \times H \times W$
$[\mathbf{F}_{a_1}, \mathbf{F}_3]$
$\mathbf{F}_2$
$[\hat{\mathbf{F}}_1, \hat{\mathbf{F}}_2]$
$\mathbf{F}_a$
$C_2 \times H \times W$
$\mathbf{F}_3$
$[\hat{\mathbf{F}}_{a_1}, \hat{\mathbf{F}}_3]$
$3C_1 \times H \times W$
$[\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3]$
$[\hat{\mathbf{F}}_1, \hat{\mathbf{F}}_2, \hat{\mathbf{F}}_3]$
Concatenation operation
Identical connection
1×1 Convolution operation

(a) Incremental Feature Amalgamation (IFA)

(b) Direct Feature Amalgamation (DFA)

Figure 3: Feature amalgamation from multiple teacher models. (a) IFA amalgamates features progressively by each time looking at two teachers, while (b) DFA amalgamates features from multiple teachers in one shot. More specifically, IFA first amalgamates the features F_1 and F_2 to obtain features F_{a_1}, which is then amalgamated with F_3 to obtain final feature set F_a; DFA, on the other hand, simultaneously amalgamates F_1, F_2 and F_3.

Loss Functions

Loss of Feature Amalgamation Recall that the target of feature amalgamation step is to remove redundant information of the concatenated features and obtain a compact feature map that preserves the critical information of the multiple teachers. Our objective is therefore set to be the L_2 loss to reconstruct the origin feature maps of l-th layer, as follows:

$$\mathcal{L}^l_{\mathrm{FA}} = \frac{1}{2}\|[\hat{F}^l_1, \ldots, \hat{F}^l_N] - [F^l_1, \ldots, F^l_N]\|^2. \tag{4}$$

Loss of Parameter Learning The loss function of the parameter learning stage, including both layer-wise learning and joint learning, is taken to be

$$\mathcal{L}^l_{\mathrm{PL}} = \frac{1}{2}\|\hat{F}^l_a - F^l_a\|^2, \tag{5}$$

where $\hat{F}^l_a$ and F^l_a correspond to the compact feature maps for layer-wise learning and to the score vector for joint learning. The complete algorithm for knowledge amalgamation from multiple teacher models is summarized in Algorithm 1, where SGD stands for Stochastic Gradient Descent.

Experiments

To evaluate the effectiveness of our proposed method, we conduct experiments on several publicly available benchmarks. More experimental results can be found in the supplementary material.

Experimental Setup

Dataset The first two datasets we adopt, CUB-200-2011 (Wah et al. 2011) and Stanford Dogs (Khosla et al. 2011), are related to animals and the last two, FGVC-Aircraft (Maji et al. 2013) and Cars (Krause et al. 2013), are related to vehicles. CUB-200-2011 consists of 11,788 images from 200 bird species, Stanford Dogs contains 12,000 images about 120 different kinds of dogs, FGVC-Aircraft consists of 10,000 images of 100 aircraft variants, and Cars comprises 16,185 images of 196 classes of cars.

For each dataset, we randomly split their categories, which are considerably correlated, into parts of equal size to train two networks. These networks are regarded as teachers to guide the learning of student network that recognizes all categories. In our supplementary material, we also show the amalgamation results across different datasets where the categories are uncorrelated.

Implementation The proposed method is implemented using PyTorch (Paszke et al. 2017) on a Quadro P5000 16G GPU. In our experiment, all the teacher models adopt the same AlexNet architecture (Krizhevsky, Sutskever, and Hinton 2012), obtained by finetuning the ImageNet pretrained models[1]. The student model has a very similar network architecture as teachers. The only difference is that the student model has in each layer a different number of kernels, i.e., a different number of feature map channels. Intuitively, the number of kernels within the student model should be larger than that of the teachers, as the student is more "knowledgeable" than the teachers due to its capability of handling the whole set of classes, and meanwhile be smaller than the sum of all teachers, as it is assumed that the features from teachers share redundancies. Please refer to the supplementary material for the detailed configuration of the network architecture.

Experimental Results

Knowledge Amalgamation from Two Teachers To verify the effectiveness of our approach, we evaluate the performance of our learned student model that amalgamates knowledge from two teacher models and implements classification task of both teachers. The following four methods are compared.

[1] https://download.pytorch.org/models/alexnet-owt-4df8aa71.pth

Table 1: Performance of knowledge amalgamation from two teachers on comprehensive classification task. The best accuracy is marked in **bold** font.

	Stanford Dogs		**CUB-200-2011**		**FGVC-Aircraft**		**Cars**	
Method	**Params**	**Accuracy**	**Params**	**Accuracy**	**Params**	**Accuracy**	**Params**	**Accuracy**
Ensemble	∼114.4M	43.5%	∼114.8M	41.4%	∼114.4M	47.1%	∼114.8M	37.8%
Baseline	∼69.9M	10.4%	∼70.2M	30.0%	∼69.8M	39.9%	∼70.2M	17.0%
Layer-wise Learning	∼69.9M	38.4%	∼70.2M	31.8%	∼69.8M	39.8%	∼70.2M	33.6%
Joint Learning	∼69.9M	**45.3**%	∼70.2M	**42.6**%	∼69.8M	**49.4**%	∼70.2M	**40.6**%

Table 2: Comparing the results of the teacher models and the learned student models on Stanford Dogs and CUB-200-2011 dataset. *Layer* denotes layer-wise parameter learning strategy, and *Joint* denotes joint learning strategy .

	Stanford Dogs			**CUB-200-2011**		
Method	**whole**	**part1**	**part2**	**whole**	**part1**	**part2**
Teacher1	-	60.8%	-	-	**53.8**%	-
Teacher2	-	-	58.5%	-	-	49.7%
Layer	38.4%	54.1%	54.3%	31.8%	44.8%	41.2%
Joint	**45.3**%	**61.5**%	**59.6**%	**42.3**%	53.2%	**50.3**%

Table 3: Comparing the results of the teacher models and the learned student models on FGVC-Aircraft and Cars dataset. *Layer* denotes layer-wise parameter learning strategy, and *Joint* denotes joint learning strategy .

	FGVC-Aircraft			**Cars**		
Method	**whole**	**part1**	**part2**	**whole**	**part1**	**part2**
Teacher1	-	67.6%	-	-	52.2%	-
Teacher2	-	-	58.8%	-	-	50.1%
Layer	39.8%	59.0%	50.8%	33.6%	47.3%	43.4%
Joint	**49.4**%	**67.8**%	**59.2**%	**40.6**%	**53.0**%	**50.4**%

- **Ensemble:** We concatenate the score vectors from the two teacher models and classify the input sampling by assigning the class of the highest score in the concatenated score vector as the label of the input.
- **Baseline:** We learn a student model by applying Hinton's knowledge distillation method (Hinton, Vinyals, and Dean 2015), which has proven a superior performance to its variants on large number ($\geq$100) of classes. Specifically, we stack the score vectors from the teachers and use the concatenated vector as the target to train the student.
- **Layer-wise Learning:** After the feature amalgamation step, we conduct only layer-wise parameter learning to obtain the student network parameters.
- **Joint Learning:** Our complete model, with parameters first layer-wise learned and then jointly learned.

The comparative results are shown in Table 1. On all benchmark datasets, our complete method *Joint Learning* achieves the highest performance among the four methods, and demands significantly fewer parameters than *Ensemble*.

We also compare the performance of the learned student model with those of the teachers. Let *part1* and *part2* denote the categories handled by the two teachers models, *teacher1* and *teacher2*, respectively, and let *whole* denote the complete set of categories. As shown in Table 2 and Table 3, our complete student model, *joint learning*, in fact outperforms the teacher models on the corresponding subtasks. For example, on the Stanford Dogs dataset, the student model achieves a *part1* accuracy of 61.5% and a *part2* one of 59.6%, while those for *teacher1* and *teacher2*, which specialize in handling *part1* and *part2*, are 60.8% and 58.5% respectively. These interesting and encouraging results show that our approach is indeed able to learn the amalgamated knowledge from both teachers, and the knowledge learned from one teacher benefits the classification task of the other.

Knowledge Amalgamation from Multiple Teachers We also test the performance of multi-teacher amalgamation. We first conduct experiments on amalgamating different numbers of teacher models. We split the Stanford Dogs dataset comprising 120 classes into four even parts, each of which contains 30 classes, and then test the classification performances on these parts by amalgamating knowledge from two, three and all four teachers using the DFA model. We show the results in Table 4. Interestingly, the more teachers used for amalgamation, the higher the classification performance is. For example, the performance on *part1* increases from 68.7% to 69.9% and further to 70.3% for two, three and four teachers. This again indicates that the potentially complementary knowledge from multiple classification tasks indeed benefits each other.

We then compare the two schemes for multi-teacher amalgamation, DFA and IFA. Note that the performances of DFA and IFA differ only when amalgamating from more than two teachers, and thus we compare their performances using three and four teachers. We show the results in Table 5. The performances of the two strategies are in general much the same where DFA yields slightly better results on one part while IFA on the others. In our supplementary material, we also provide experimental results of IFA with different amalgamating orders.

Ablation Study

We also conduct ablation studies to validate the proposed method as follows.

Feature Adaption To show the effectiveness of FAM, we compare the classification performances of the proposed model with FAM turned on and off. As shown in Table 6, when FAM is turned on, the accuracies are significantly higher than those with FAM turned off on all the datasets. This indicates that the FAM is indeed able to transform

Table 4: Classification performances of the student models, whose knowledge is amalgamated from different numbers of teachers using DFA.

	Stanford Dogs			
Method	**part1**	**part2**	**part3**	**part4**
From 2 teachers	68.7%	66.1%	-	-
From 3 teachers	69.9%	67.7%	63.8%	-
From 4 teachers	**70.3%**	**68.0%**	**65.4%**	67.3%

Table 5: Classification performances of the student models, whose knowledge is amalgamated from different numbers of teachers using DFA and IFA.

	Stanford Dogs			
Method	**part1**	**part2**	**part3**	**part4**
DFA from 3 teachers	**69.9%**	67.7%	63.8%	-
IFA from 3 teachers	69.6%	**69.1%**	**64.7%**	-
DFA from 4 teachers	**70.3%**	68.0%	65.4%	67.3%
IFA from 4 teachers	69.9%	**68.8%**	**65.5%**	**67.9%**

amalgamated features into a form that better adapts to non-parametric layers in the network, and meanwhile preserves and passes the critical information to the next layer.

Layer-wise Parameter Learning We also investigate the power of the layer-wise learning strategy, by comparing the student model with and without the layer-wise learning, which correspond to the *joint learning* method and the *baseline* described in the previous section. We show their training and testing errors versus the epochs in Figure 4. The test error of *joint learning* with layer-wise learning is significantly lower than that of the *baseline* without layer-wise learning. In fact, despite not shown here, without layer-wise learning the test error at 300 epochs remains to be 60.1%, as indicated by the *baseline* in Table 1. We may thus safely conclude that layer-wise learning indeed facilitates the training compared to learning from scratch as done for *baseline*.

Joint Parameter Learning We compare in Table 2 and Table 3 the results of *layer-wise learning* only and *joint learning*, where the latter one outperforms the former on all the four datasets. This validates in part our hypothesis that the *layer-wise learning* accumulates errors across layers during training, which can be alleviated by *joint learning*.

Table 6: Classification performances of the student model with and without FAM. For simplicity, "Dogs" denotes "Stanford Dogs", "CUB" denotes "CUB-200-2011" and "Aircraft" denotes "FGVC-Aircraft".

Method	Dogs	CUB	Aircraft	Cars
W/O FAM	25.3%	33.9%	39.7%	30.7%
W/ FAM	**45.3%**	**42.6%**	**49.4%**	**40.6%**

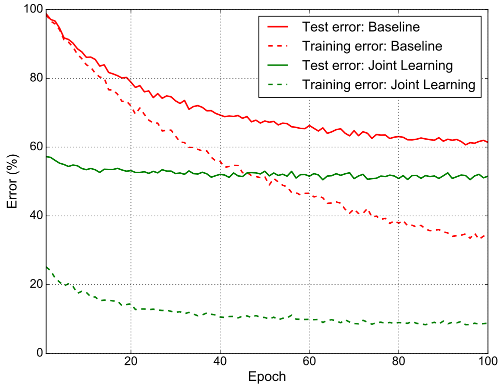


Figure 4: Training and test errors of *baseline* and *joint learning* versus the number of epochs on Stanford Dogs. *Joint learning* converges to much lower errors.

Conclusion and Future Work

In this paper, we propose a new model-reusing task, termed *knowledge amalgamation*, which aims at learning a compact student model capable of handing the "super" task from multiple teachers, each of which specializes in a different task. This task is in our opinion a research-worthy topic in the sense that it allows amalgamating well-trained models, many of which are learned using large-scale or private datasets that are not publicly available, to derive a lightweight student model that approximates or even outperforms the teachers.

To this end, we propose a pilot approach towards solving this task. The proposed approach follows a two-step strategy by first conducting feature amalgamation from the multiple teachers and then treating the obtained features as guidance to learn the parameters of the student network. We conduct experiments on four datasets to validate the proposed approach, which yields very promising results: the learned student model can in fact perform better than the teachers at their specializations, at a model scale that is much smaller than the ensemble of the teachers. We also justify the validness of several components by conducting the ablation study.

Admittedly, this pilot approach in the current form, despite its very encouraging results, indeed has some limitations. We assume that the teacher models and the student one share the same network architecture, which might be a strong assumption in some real-world scenarios. In our near-future work, we will therefore investigate amalgamating knowledge from teachers of different network architectures, which truly allows us to reuse the knowledge of massive well-trained neural networks in the wild.

In the longer term, we will explore how to bridge the semantic gap among different network architectures and reuse the amalgamated knowledge to new tasks, enabling the *amalgamated knowledge transfer*.

Acknowledgments This work is supported by Natonal Basic Research Program of China under Grant No. 2015CB352400, National Natural Science Foundation of China (61572428, U1509206), Fundamental Research Funds for the Central Universities (2017FZA5014), and Key Research and Development Program of Zhejiang Province (2018C01004).

References

Buciluă, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model compression. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 535–541.

Cui, Y.; Song, Y.; Sun, C.; Howard, A.; and Belongie, S. 2018. Large scale fine-grained categorization and domain-specific transfer learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4109–4118.

Ding, Z.; Li, S.; Shao, M.; and Fu, Y. 2018. Graph adaptive knowledge transfer for unsupervised domain adaptation. In *European Conference on Computer Vision (ECCV)*, 37–52.

Gholami, B.; Rudovic, O.; and Pavlovic, V. 2017. Punda: Probabilistic unsupervised domain adaptation for knowledge transfer across visual categories. In *The IEEE International Conference on Computer Vision (ICCV)*, 3601–3610.

Gupta, S.; Hoffman, J.; and Malik, J. 2015. Cross modal distillation for supervision transfer. *arXiv preprint arXiv:1507.00448*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *Advances in Neural Information Processing Systems (NIPS)*.

Hong, S.; Oh, J.; Lee, H.; and Han, B. 2016. Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3204–3212.

Hu, J.; Lu, J.; and Tan, Y.-P. 2015. Deep transfer metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 325–333.

Huang, Z., and Wang, N. 2017. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*.

Huang, H.; Huang, Q.; and Krähenbühl, P. 2018. Domain transfer through deep activation matching. In *European Conference on Computer Vision (ECCV)*, 611–626.

Huang, X.; Peng, Y.; and Yuan, M. 2017. Cross-modal common representation learning by hybrid transfer network. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1893–1900.

Khosla, A.; Jayadevaprakash, N.; Yao, B.; and Fei-Fei, L. 2011. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *International IEEE Workshop on 3D Representation and Recognition (3dRR)*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 1097–1105.

Long, M.; Wang, J.; Ding, G.; Sun, J.; and Philip, S. Y. 2013. Transfer feature learning with joint distribution adaptation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2200–2207.

Maji, S.; Kannala, J.; Rahtu, E.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Pan, S. J.; Yang, Q.; et al. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 22(10):1345–1359.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems (NIPS)*.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. Fitnets: Hints for thin deep nets. *International Conference on Learning Representations (ICLR)*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical report.

Wang, Z.; Deng, Z.; and Wang, S. 2016. Accelerating convolutional neural networks with dominant convolutional kernel and knowledge pre-regression. In *European Conference on Computer Vision (ECCV)*, 533–548.

Xu, D.; Ouyang, W.; Wang, X.; and Sebe, N. 2018. Padnet: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, Y.; Zhan, D.; Fan, Y.; Jiang, Y.; and Zhou, Z. 2017. Deep learning for fixed model reuse. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2831–2837.

Zagoruyko, S., and Komodakis, N. 2017. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations (ICLR)*.