# Network Specialization via Feature-level Knowledge Distillation

Gaowen Liu[1], Yuzhang Shang[1,2], Yuguang Yao[1,3], Ramana Kompella[1]
[1]Cisco Research, [2]Illinois Institute of Technology, [3]Michigan State University
{gaoliu, yuzshang, yugyao, rkompell}@cisco.com

## Abstract

*State-of-the-art model specialization methods are mainly based on fine-tuning a pre-trained machine learning model to fit the specific needs of a particular task or application. Or by modifying the architecture of the model itself. However, these methods are not preferable in industrial applications because of the model's large size and the complexity of the training process. In this paper, the difficulty of network specialization is attributed to overfitting caused by a lack of data, and we propose a novel model specialization method by Knowledge Distillation (SKD). The proposed methods merge transfer learning and model compression into one stage. Specifically, we distill and transfer knowledge at the feature map level, circumventing logit-level inconsistency between teacher and student. We empirically investigate and prove the effects of the three parts: Models can be specialized to customer use cases by knowledge distillation. Knowledge distillation can effectively regularize the knowledge transfer process to a smaller, task-specific model. Compared with classical methods such as training a model from scratch and model fine-tuning, our methods achieve comparable and much better results and have better training efficiency on the CIFAR-100 dataset for image classification tasks. This paper proves the great potential of model specialization by knowledge distillation.*

## 1. Introduction

Specialized neural networks have recently become common for edge devices. Unlike general-purpose neural networks that are used for a diverse range of classification tasks, specialized neural networks [4, 5, 16, 31] are trained to fit the specific needs of a particular task or application. In other words, it involves making changes to a pre-trained model to adapt it to a specific domain or problem. There are several reasons why model specialization may be necessary. First, a pre-trained model may not be optimized for a specific task. Additionally, different applications may have unique requirements or constraints that cannot be met by a general-purpose model. For instance, a customer located near a harbor may not need to detect objects appearing in a ravine. In such cases, a model that is designed for a specialized purpose can be tailored to meet the specific needs of a customer or application. Furthermore, by customizing the model for the specific task, it can be optimized for accuracy and efficiency, while also reducing the size and complexity of the model. This approach can be particularly useful for resource-constrained devices, such as mobile phones or IoT devices, where minimizing the size and reducing the inference time of the model is crucial.

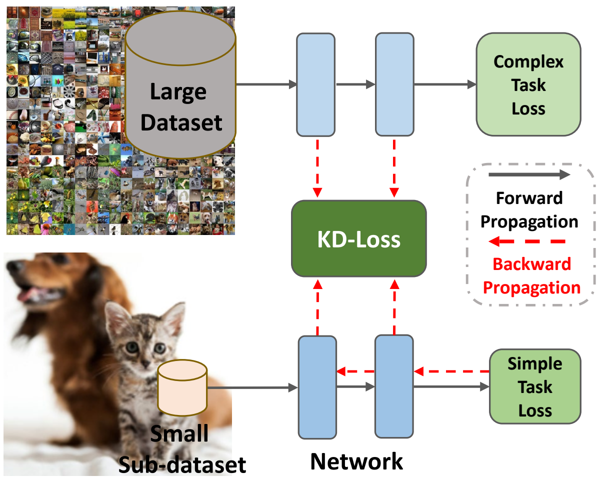


Figure 1. Knowledge distillation enables effective regularization of knowledge transfer from large dataset to smaller, task-specific models on small datasets.

There are different ways to specialize a model, depending on the specific task and the type of model being used. One common approach is to fine-tune [3] a pre-trained model, which involves re-training the last few layers of the model on a small amount of task-specific data. This allows the model to adapt to the particularities of the task while retaining the knowledge learned from the pre-trained model. However, fine-tuning a pre-trained model maintains the model size and the fine-tuned large models are not a good solution for practical use cases. If one considers the deployment of deep learning models in low compute applications, such as mobile phones, drones, unmanned vehicles, Internet of Things, or other edge applications, these fine-tuned large models become a limiting factor in taking their success to these kinds of computing platforms. In the literature, a potential solution is knowledge distillation(KD) [14].

Knowledge distillation is a process of training a smaller and computationally efficient model to mimic the behavior and predictions of a larger and more complex model. The goal of knowledge distillation is to transfer the knowledge learned by a large model to a smaller model so that the smaller model can achieve similar performance while being faster and more lightweight (Hinton, 2015 [14]). Classical distillation methods achieve high efficiency and accuracy on general-purpose tasks but neglect use cases for specialized tasks. To overcome these limitations, we propose that knowledge distillation can be used to build a task-specific specialized model by transferring the knowledge learned by a larger, more complex model to a smaller, more efficient model that is specialized for a particular task.

We summarize our main contribution as follows:

1. We propose a novel knowledge distillation framework as an approach to construct a specialized network. By distilling the knowledge from the general-purpose teacher model, the specialized student model can achieve comparable or even superior performance, while requiring less computational resources for training and inference.

2. The proposed framework overcomes the problem of overfitting in specialized neural networks by allowing the student model to learn a better representation from the more complex teacher model.

3. Experiments demonstrate that the proposed method can significantly improve the performance of specialized network training for simpler tasks with small datasets.

Overall, the proposed framework is a promising approach for improving the performance of specialized networks in simpler tasks with small datasets.

## 2. Related Work

### 2.1. Model specialization

Model specialization [4, 5, 16, 31] is a rapidly evolving field of machine learning, with ongoing research and development aimed at improving the performance, efficiency, and interpretability of models for specific tasks and domains. Previous works of model specialization including transfer learning [24, 37] and fine-tuning [3], neural architecture search [20, 26, 27, 38], meta-learning [10, 23, 29, 32], have shown substantial gains for a wide range of downstream tasks.

Transfer learning [24, 37] is a powerful technique for model specialization. A main approach to transfer learning is fine-tuning. Fine-tuning [3] pre-trained models are the most popular method to achieve model specialization. Fine-tuning a model refers to the process of taking a pre-trained model and training it further on a new task or a new dataset. The pre-trained model has already learned to recognize a set of features on a large-scale dataset and can be used as a starting point for the new task. Fine-tuning allows the model to learn new features that are specific to the new task while retaining the knowledge learned from the pre-trained model. Despite the strong empirical performance of fine-tuned models, fine-tuning is an unstable process: training the same model with multiple random seeds can result in a large variance in the task performance.

Neural architecture search (NAS) [20, 26, 27, 38] is a technique that uses machine learning algorithms to automatically discover the optimal architecture for a given task [25]. Pioneers works such as NASRL [39] and MetaQNN [1] have reached state-of-the-art classification accuracy on image classification tasks. This demonstrates that automated neural architecture design is feasible. However, NAS requires a large amount of computation and time to search for an optimal neural network architecture.

Meta-learning [10, 23, 29, 32], or learning to learn, is a technique that involves training a model to learn how to adapt to new tasks more quickly and effectively. The popular approach to address this problem is either by utilizing gradients [10] or evolutionary [8] procedures. There are also a number of approaches for label prediction based on limited number of training data. However, Meta-learning algorithms are highly sensitive to hyperparameters. Finding the optimal set of hyperparameters can be challenging, and it often requires a lot of trial-and-error experimentation, which prohibits its applications for edge devices.

Overall, model specialization is a vibrant area of research with many exciting developments and applications. However, none of these algorithms does help with the model deployment on resource-constrained edge devices. While edge applications are mainstream use cases for model specialization.

### 2.2. Training on small dateset

Training a machine learning model on a small dataset [2, 9] or a small task can present several difficulties. With a small dataset, the model may learn to memorize the training examples rather than generalize patterns. This can result in poor performance when the model is applied to new, unseen data. A small dataset may have limited variation in the feature space, making it difficult for the model to learn meaningful patterns. This can result in poor performance or models that fail to generalize to new data. To overcome these challenges, we introduce knowledge distillation to enhance the performance of the model trained on the smaller dataset for user-defined specialized use cases.

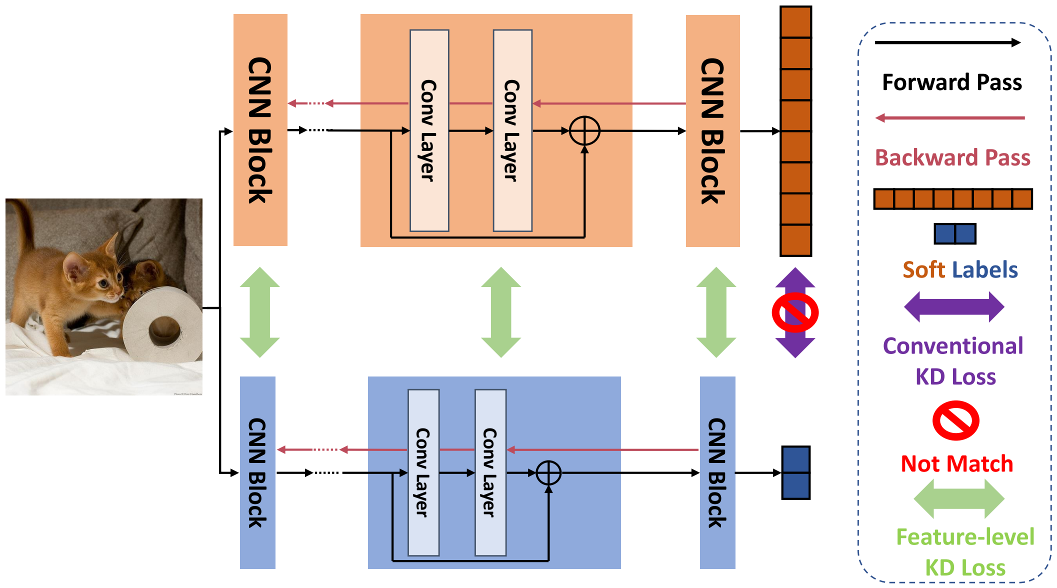


Figure 2. An overview of our proposed SKD. For the teacher-student backbone in our network specialization task, the conventional knowledge distillation loss does not fit. The input and output feature maps of each module are used to format the knowledge to regularize the training of the specialized network.

### 2.3. Knowledge distillation

The idea of knowledge distillation was introduced by Hinton et al. [14]. The authors proposed a method to transfer knowledge from a large, well-trained neural network (called the teacher network) to a smaller and faster network (called the student network). The method involves training the student network to match the outputs of the teacher network on a given set of inputs, while also minimizing the difference between the logits (i.e., the unnormalized probabilities) of the teacher and student networks. Since then, knowledge distillation has become a popular technique in deep learning, with many variations and applications in various fields such as computer vision, natural language processing, and speech recognition. Since [28], most of the research attention has shifted from logits-based knowledge distillation [6, 11, 15, 18, 21, 35] to feature-based knowledge distillation. The Performance of feature-based distillation [12, 19, 28, 33, 34, 36] is superior on various tasks. In this paper, knowledge distillation is proposed as a novel model specialization methods on small dataset.

## 3. Method

In this section, we first formulate the network specialization explicitly. We then demonstrate that overfitting is one of the challenges associated with network specialization. Motivated by this, we propose a knowledge distillation (KD) based regularization method for mitigating overfitting. In this way, the performance of specialized networks can be boosted.

### 3.1. Problem Formulation and Motivation

We first formalize our task, network specialization. Suppose we have a complex task with a large dataset $\mathcal{D}$ (e.g. 1000-class classification on ImageNet [7]), but we really care about a relatively simple task with a sub-dataset $\mathcal{D}_s$ (e.g. cat and dog distinction), which is contained in the complex task. Strictly,

$$\mathcal{D}_s = \{\mathbf{x}_i, y_i\}_{i=0}^{M} \subseteq \mathcal{D} = \{\mathbf{x}_j, y_j\}_{j=0}^{N}, \tag{1}$$

in which $\{\mathbf{x}_i, y_i\}$ is the $i$-th sample-label pair, and $M \ll N$.

As a matter of fact, this setting is widely required by the industry for real-world deployment of deep learning algorithms. For example, a customer who wants to detect vehicles may not require the animal classification capabilities of a pre-trained model. However, in this setting, there are a number of challenges that need to be addressed. Insufficient training data can result in overfitting, which is one of the most notorious challenges. It also leads to poor representation ability of specialized networks as shown in Fig. 6 (b).

### 3.2. Knowledge Distillation for Specialized Network

We define a fully-connected neural network with $L$ layers of widths $d_1, \cdots d_L (d = \sum_{k=1}^{L} d_k)$ as the form of function $f : \mathbb{R}^{d_0} \longmapsto \mathbb{R}^{d_L}$:

$$f(\mathbf{x}) = (T^L \circ \sigma \circ T^{L-1} \circ \cdots \circ \sigma \circ T^1)(\mathbf{x}), \tag{2}$$

where each $T^{(k)} : \mathbb{R}^{d_{k-1}} \longmapsto \mathbb{R}^{d_k}$ is an affine function ($d_0$ and $d_L$ are the sizes of network's input and output feature maps) and $\sigma$ performs element-wise activation for feature maps. For $k$-th layer of the networks, $T^{(k)}(\mathbf{u}) = \mathbf{W}^k\mathbf{u} + \mathbf{b}^k$, where $\mathbf{W}^k$ and $\mathbf{b}^k$ stand for the weight matrix and bias vector, respectively. For generality purpose, we discard the bias term of the network, so that the network can be simplified as:

$$f(\mathbf{W}^1, \cdots, \mathbf{W}^L; \mathbf{x}) = (\mathbf{W}^L \circ \sigma \circ \mathbf{W}^{L-1} \circ \cdots \circ \sigma \circ \mathbf{W}^1)(\mathbf{x}). \tag{3}$$

Notably, it is sufficient to consider networks with the most straightforward fully-connected layers, since layer with complex structures such as convolution layer can also be denoted as the form of matrix multiplication. We consider a convolution layer with $i$ input channels and $o$ output channels, and the size of the kernel is $w \times h$, resulting in $iowh$ parameters. We can re-arrange the parameters to a matrix of size $o \times ihw$, such that this convolution layer can also be processed in the same way as the other fully-connected layers do. Hence, our analysis has no loss for generality in this configuration of function $f$.

Following Eq. 3, we define the function form of the teacher network as $f_T(\mathbf{W}_T^1, \cdots, \mathbf{W}_T^{L_T}; \mathbf{x})$, and the student network as $f_S(\mathbf{W}_S^1, \cdots, \mathbf{W}_T^{L_S}; \mathbf{x})$, such that the feature-based KD paradigm can be interpreted as:

$$\forall \mathbf{x} \in \mathbf{Data}, \quad \underset{\mathbf{W}_S^1, \cdots, \mathbf{W}_S^{L_S}}{\arg\min} \; Dist(\mathcal{T}(f_T(\mathbf{x})), \mathcal{T}(f_S(\mathbf{x}))), \tag{4}$$

where given the same data, the ultimate goal of KD paradigm is to minimize the distance between teacher and student for optimizing the latter's parameters $\{\mathbf{W}_S^i\}$. Particularly, $Dist(\cdot, \cdot)$ is a distance function, and $\mathcal{T}(\cdot)$ is a transformation to turn feature maps into more measurable and learnable knowledge. By utilizing those designed knowledge, the student network is forced to mimic the teacher network and hopefully obtains comparable performance with lighter architecture.

**Regularization Property.**

The outputs of the last layer (i.e. soft labels) are explored by some conventional knowledge distillation methods and transformed into various types of knowledge [14,36]. Then, the defined knowledge aligns the teacher and student. However, those methods cannot be naively scaled into our setting. As shown in Fig. 2, the students and teachers have different architectures, especially with respect to the output dimensions of classifier heads, which hinders the alignment of teacher and student.

To tackle this problem, we implement KD at the feature map level. Specifically, we absorb the ideas of [12, 13, 30],

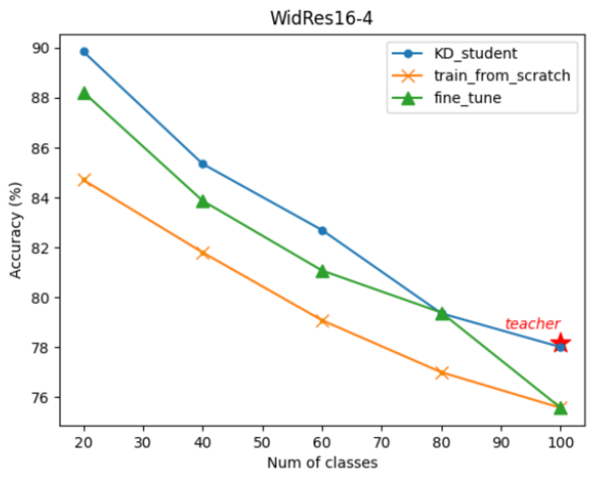


Figure 3. Experimental results compare the performance of knowledge distillation (KD) with models trained from scratch and fine-tuning.

and construct our loss objective as follows: $\mathcal{L}_{KD} =$

$$\sum_{i=1}^{L-1} \frac{\|\mathbf{FM}_T^i - \mathbf{FM}_S^i\|_2 + \|\mathbf{TM}_T^i\|_{SN} - \|\mathbf{TM}_S^i\|_{SN}}{\beta^{L-1-i}}, \tag{5}$$

where $\mathbf{FM}_T^i$ is the $i$-th layer's feature maps of teacher (corresponding to the feature-based KD loss in [12]), $\|\mathbf{TM}_T^i\|_{SN}$ and $\|\mathbf{TM}_S^i\|_{SN}$ for each $i \in \{1, \ldots, L\}$ (corresponding to the feature-based Lipschitz KD loss in [30]), and $\beta$ is a coefficient greater than 1. Hence, the $\beta^{L-1-i}$ decreases with $i$ increasing and consequently the $\frac{\|\mathbf{FM}_T^i - \mathbf{FM}_S^i\|_2 + \|\mathbf{TM}_T^i\|_{SN} - \|\mathbf{TM}_S^i\|_{SN}}{\beta^{L-1-i}}$ increases. In this way, we give more weight on higher layer features since they are closer to the features performing tasks.

Combined with the cross entropy loss $\mathcal{L}_{CE}$ and knowledge distillation loss $\mathcal{L}_{KD}$, we can obtain the overall loss:

$$\mathcal{L} = \frac{\lambda}{2} \cdot \mathcal{L}_{KD} + \mathcal{L}_{CE}, \tag{6}$$

where $\lambda$ is used to control the degree of designed KD loss.

Feature-based KD is advantageous for our task, network specialization, in which it can regularize the training of the specialized network. In other words, compared with training the specialized network from scratch (i.e., singly optimizing $\mathcal{L}_{CE}$ in Eq. 6), $\mathcal{L}_{KD}$ can serve as a regularization term. More theoretical details can be found in [12, 30], and empirical validation is conducted in the next section.

## 4. Experimental results

In this session, we perform experiments on image classification tasks to validate the effectiveness of our proposed algorithm. We compare our method with the state-of-the-art model specialization method which is fine-tuning. In

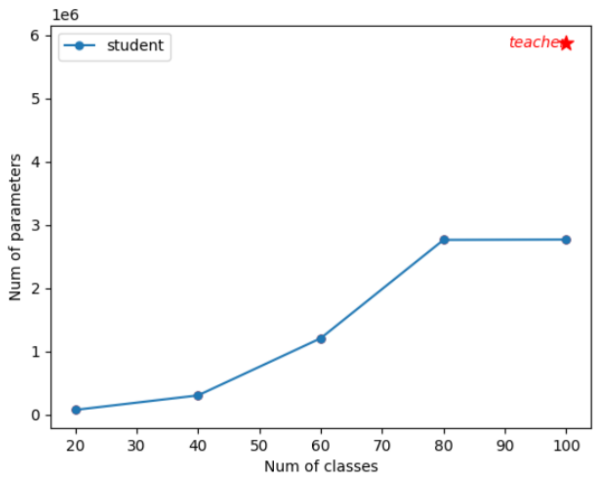


Figure 4. Model size.

addition, we conducted a series of experiments comparing our method to different models trained from scratch under various settings.

**CIFAR-100 [18]** The CIFAR-100 dataset is a popular benchmark for image classification tasks, and many state-of-the-art models have been trained and evaluated on this dataset. The dataset contains 100 classes of 600 images each, with 500 images in the training set and 100 images in the test set.

**Implementation Details.** On CIFAR-100, we optimize the teacher and student model using Adam [17] and SGD with Nesterov [22] respectively, where the momentum term and weight decay in Nesterov are set to 0.9 and $5 \times 10^{-4}$. Moreover, the learning rate is initialized to 0.1. The learning rate is decayed by 0.1, 0.01, and 0.002 at 100, 150, and 175 epochs. In addition, we train the teacher and student model for 200 epochs.

**Main Results** We discuss experimental results on CIFAR-100 to examine our algorithm. We evaluate the models' performance in terms of accuracy for the image classification task. To evaluate the performance of the proposed method across different dataset sizes, we divided the dataset into 5 sub-datasets with varying numbers of classes, including 20, 40, 60, and 80. This enabled us to compare the effectiveness of the method across different dataset sizes and to gain insights into how it performs on smaller and larger datasets.

The teacher network used in this experimental setting is a wide-residual network with 28 layers and a widening factor of 4, denoted as WRN-28-4. The student network is WRN-16-4.

The first baseline involves network specialization via fine-tuning, while the second baseline is training a WRN-16-4 model from scratch. The accuracy is reported in table 1 and visualized in figure 3. Table 1 presents the results

| Method | 20 classes | 40 classes | 60 classes | 80 classes | 100 classes |
|---|---|---|---|---|---|
| SKD | 90.96 | 85.67 | 82.8 | 80.625 | 78.01 |
| Training from scratch | 84.706 | 81.802 | 79.085 | 77.002 | 75.576 |
| Fine-tune | 88.225 | 83.872 | 81.078 | 79.395 | 75.576 |

Table 1. The accuracy of specialized models trained from scratch and trained by knowledge distillation on the same neural network architecture (WideResNet 16-4)

| Groups | 0-19 | 20-39 | 40-59 | 60-79 | 80-99 | Avg |
|---|---|---|---|---|---|---|
| Accuracy | 90.95 | 91.7 | 90.1 | 89.3 | 92.8 | 90.97 |

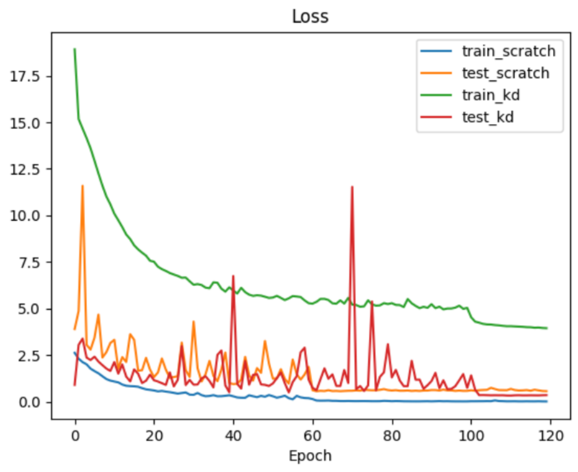
Table 2. Accuracy on randomly selected small tasks (WideResNet 16-4).

of our proposed approach SKD for network specialization, along with two baseline methods. Accuracies are compared across different sub-dataset, in Table 1, each column represents results on a sub-dataset. For each of the 5 subdatasets, we compared the accuracy of the proposed SKD method (first line) with two baselines (second and third lines). Our results indicate that, compared to training from scratch, the proposed SKD method achieved an increase in accuracy of 3.2% on the 100-class dataset and 7.3% on the 20-class dataset. Moreover, when compared to fine-tuning, the SKD method achieved an accuracy improvement of 3.2% on the 100-class dataset and 3.1% on the 20-class dataset. These findings demonstrate that the SKD method outperforms the baselines, particularly on smaller datasets, providing evidence that the proposed SKD method is an effective approach for improving the accuracy of models trained on small datasets, and can achieve better performance than traditional model specialization methods.

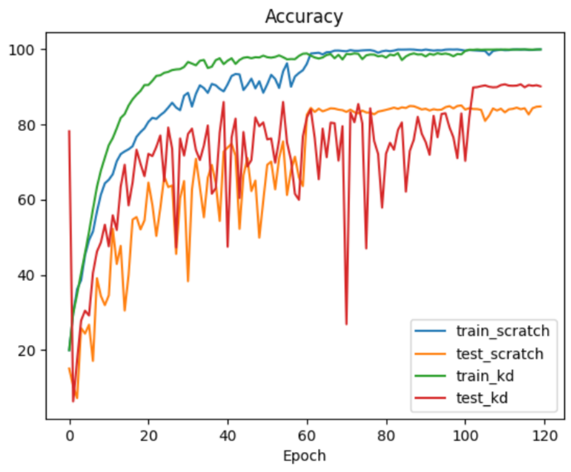
Figure 4 shows the number of model parameters needed in order to keep a certain accuracy ($79\% \pm 0.2\%$). Experiments indicate that given smaller datasets, models can be effectively compressed to a smaller size. This is particularly relevant for scenarios where resource-constrained devices are used, a smaller model can be designed to meet the customer-defined accuracy requirements while minimizing memory and processing requirements.

**Classification on small-size dataset** Classification of specialized tasks usually comes with a small-size dataset. We further conducted a series of experiments with 5 groups of a random selection of 20 classes in table 2. The stability of the results across various combinations of classes in the 20-class setting provides further evidence that the proposed method is effective for specialized tasks on small datasets.

**Representation ability study** The t-SNE plots of feature representations obtained from training with knowledge distillation and training from scratch are compared in figure 6. The plots reveal that the features extracted from the knowledge distillation process exhibit well-defined clusters

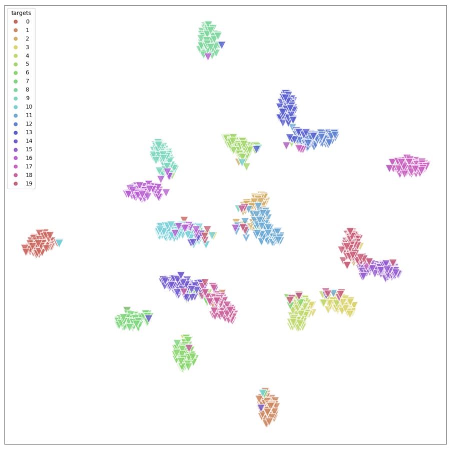


(a) Training and validation Loss.

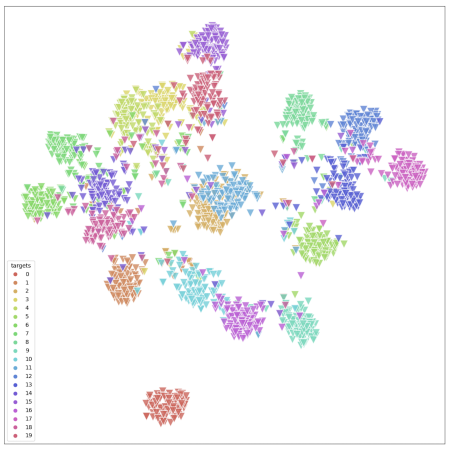


(b) Training and validation accuracy.

Figure 5. Line plots of the training and validation loss values over several training epochs on the 20-class setting.



(a) Training with knowledge distillation.



(b) Training from scratch.

Figure 6. Visualizing feature representation using t-SNE.

and clear boundaries, while the features obtained by training from scratch appear to be mixed and less well-separated. These observations lead us to conclude that the proposed method is effective in learning a better feature representation for the given task.

**Further analysis** We present the visualization of training and validation loss values over several training epochs on the 20-class setting in figure.5. **(a)** as training losses decrease, compared with the validation loss of training from scratch (orange line), which remains stable during long iterations, the one of training with KD (red line) consistently decreases. **(b)** When turning off the KD loss, the performance on the validation set drops (orange line v.s. red line) while the training accuracy stays at the same level (blue and green lines). As seen in both observations, KD can mitigate the overfitting of the specialized network if trained on a simpler task with a small dataset.

## 5. Conclusion

In this work, we have proposed a novel knowledge distillation framework (SKD) that addresses the challenge of constructing specialized networks for simpler tasks with small datasets. SKD overcomes the problem of overfitting in specialized neural networks by allowing the student model to learn a better representation from the more complex teacher model. Experimental results have demonstrated that the proposed framework is effective in improving the performance of specialized network training, resulting in significant performance gains in various settings of simpler tasks with small datasets. The proposed framework can serve as a valuable tool for practitioners and researchers working in areas where tasks are simple and data is limited and can lead to the development of more efficient and effective models for practical applications.

## References

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, 2017. 2

[2] Bjorn Barz and Joachim Denzler. Deep learning on small datasets without pre-training using cosine loss. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1371–1380, 2020. 2

[3] Yoshua Bengio. Learning deep architectures for ai. In *Foundations and Trends in Machine Learning*, volume 2, pages 1–127. Now Publishers, 2009. 1, 2

[4] Fadi Biadsy, Youzheng Chen, Xia Zhang, Oleg Rybakov, Andrew Rosenberg, and Pedro J. Moreno. A scalable model specialization framework for training and inference using submodels and its application to speech model personalization, 2022. 1, 2

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. 1, 2

[6] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *ICCV*, 2019. 3

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 3

[8] Chrisantha Fernando, Jakub Sygnowski, Simon Osindero, Jane X Wang, Tom Schaul, Denis Teplyashin, Pablo Sprechmann, Alexander Pritzel, and Andrei A Rusu. Meta-learning by the baldwin effect. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018. 2

[9] Manuel Fernández-Delgado, Eva Cernadas, Senen Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014. 2

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017. 2

[11] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *ICML*, 2018. 3

[12] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *ICCV*, 2019. 3, 4

[13] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019. 4

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2014. 2, 3, 4

[15] Yi-Cheng Huang, Kuo-Chun Hung, Chun-Chang Liu, Ting-Hsueh Chuang, and Shean-Juinn Chiou. Customized convolutional neural networks technology for machined product inspection, 2022. 3

[16] Yeong-Hwa Jin, Keon-Ho Lee, and Dong-Wan Choi. Querynet: Querying neural networks for lightweight specialized models. *Information Sciences*, 589:186–198, 2019. 1, 2

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[18] Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Computer Science Department, University of Toronto*, 2009. 3, 5

[19] Sihao Lin, Hongwei Xie, Bing Wang, Kaicheng Yu, Xiaojun Chang, Xiaodan Liang, and Gang Wang. Knowledge distillation via the target-aware transformer. In *CVPR*, 2022. 3

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 2

[21] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI*, 2020. 3

[22] Yurii E Nesterov. A method of solving a convex programming problem with convergence rate. *Doklady Akademii Nauk*, 269(3):543–547, 1983. 5

[23] Alex Nichol, Joshua Achiam, and John Schulman. Reptile: A scalable metalearning algorithm. In *International Conference on Learning Representations*, 2018. 2

[24] David N Perkins and Gavriel Salomon. *Transfer of learning*. Pergamon, Oxford, England, 1992. 2

[25] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. volume abs/1802.03268, 2018. 2

[26] Esteban Real, Chen Liang, David R So, and Quoc V Le. Automl-zero: Evolving machine learning algorithms from scratch. *arXiv preprint arXiv:2003.03384*, 2020. 2

[27] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021. 2

[28] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015. 3

[29] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2016. 2

[30] Yuzhang Shang, Bin Duan, Ziliang Zong, Liqiang Nie, and Yan Yan. Lipschitz continuity guided knowledge distillation. In *ICCV*, pages 10675–10684, 2021. 4

[31] Joel Shor, Dotan Emanuel, Oran Lang, Omry Tuval, Michael Brenner, Julie Cattiau, Fernando Vieira, Maeve McNally, Taylor Charbonneau, Melissa Nollstadt, Avinatan Hassidim,

and Yossi Matias. Personalizing asr for dysarthric and accented speech with limited data. In *Proc. Interspeech*, pages 784–788, 2019. 1, 2

[32] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. 2

[33] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020. 3

[34] Kafeng Wang, Xitong Gao, Yiren Zhao, Xingjian Li, Dejing Dou, and Cheng-Zhong Xu. Pay attention to features, transfer learn faster cnns. In *ICLR*, 2020. 3

[35] Chenglin Yang, Lingxi Xie, Chi Su, and Alan L Yuille. Snapshot distillation: Teacher-student optimization in one generation. In *CVPR*, 2019. 3

[36] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *CVPR*, 2022. 3, 4

[37] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the Institute of Radio Engineers*, 109(1):43–76, Jan. 2021. 2

[38] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. 2

[39] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 2