A Comparison Between Few-Shot Learning and Knowledge Distillation

Anonymous CVPR submission

Paper ID *****

1. Introduction

Applications of deep learning have come a long way in different domains such as computer vision and natural language processing, to name a few. The trend of achieving better performance on various tasks has seen the surge of development of models which are large in terms of parameters and take a huge amount of time to train [2, 16, 23]. This gave rise to the research trend of developing models that are just as capable as these enormous models but smaller in terms of the number of parameters, such as model pruning [10, 12], quantization [4, 15, 21], knowledge distillation [9, 28] etc. These smaller models are easier to deploy on edge devices and easier to experiment with. We have also seen a lot of strategies to train a model using a limited number of labeled samples such as few-shot learning [20], domain adaptation [6] etc. Such cases typically occur in niche domains, for example, medical data. In this project, we compare two popular efficient Deep Learning methods: Knowledge Distillation and Few-Shot Learning. Knowledge Distillation involves training a smaller, less resource-intensive model (the student) using a larger, pretrained model (the teacher) as a guide. On the opposite end, Few-Shot Learning involves training a model on a very limited amount of data and is typically employed in scenarios where acquiring large datasets is impractical. Our aim is to compare and contrast between these two popular methods to study the effect of model size and dataset size on the performance of the model. To summarize, the main highlights of this study are:

1. We are comparing two models, (i) a smaller student model trained through knowledge distillation from a teacher model pretrained on a different dataset, and (ii) a larger, few-shot trained teacher model that's big in size and was pretrained on a large dataset but only has a few in-domain examples to learn from.

2. We want to see how well the big teacher model can transfer its knowledge to the smaller student model, especially since the student is learning from a completely different set of data than the teacher was trained on.

3. For the Few-Shot setup, the larger model has to perform with very little in-domain data. It's interesting to see how this limitation affects its performance.

4. We want to find out which scenario impacts performance of a model more adversely: having a lot of data but a smaller model (as with the student) or having a massive model but very little in-domain data (as with the teacher).

5. The main idea is to better understand resource-efficient deep learning techniques. We are looking into how different methods work and their effectiveness under various constraints.

All of our experiment codes can be found in the Github repository: `https://github.com/preetom-saha-arko/kd_vs_few_shot`.

2. Related Works

The role of knowledge distillation [9] is to transfer knowledge from a larger teacher model with more parameters which was pretrained with sufficient resources to a smaller student model. The goal is to develop a smaller variant of the larger teacher model, which would be close to the teacher model in terms of performance. There are various forms of such transfer such as the regular response based transfer method where the output of the student tries to match the teacher [9]. In this case, the student model tries to match the prediction labels of the teacher in addition to the ground truth prediction labels. Also there is another variant where the student focuses on matching its intermediate representation with the intermediate representation of the teacher model [17] in addition to optimizing its regular loss. There is also another strategy of knowledge distillation that involves transferring knowledge using multiple teacher networks [11, 29]. There are concepts of transferring the attention maps from one model to another in image based networks [8, 30].

Few-shot learning refers to training a machine learning model using a very small amount of labeled data. Multiple strategies have been developed over the years to tackle the problem of limited labeled data. Broadly they can be categorized to various learning paradigms such as (i) meta learning where the idea is to train a model on a variety of learning tasks so that it can quickly adapt to new tasks using only a few examples and (ii) non meta learning where a model is trained on a related task with ample data, hoping the model will generalize well when fine-tuned on a few-shot task [14]. There is also the popular approach of metric learning which is a type of meta learning that focuses on learning distances or similarities between inputs. Algorithms such as Prototypical Network [18], Relation Network [22], Reptile [13] are popular approaches to this. Strategies such as simple shot [26], Transductive Finetuning [5] have been developed which can be characterized under non meta learning. But the overall goal of these models and methods is to allow a model to learn or generalize from a limited number of examples—sometimes as few as one or two per class by using query and support sets and training a model with few classes.

3. Knowledge Distillation

Knowledge distillation [9] involves transferring knowledge from a complex, cumbersome model (the teacher) to a simpler, more lightweight model (the student) while maintaining or even improving performance. In other words, it involves training a student model to mimic the behavior of a teacher model. The process typically consists of two stages:

1. **Teacher Model Training:** The teacher model, often a large and complex neural network, is trained on a given dataset to achieve high performance on a specific task. The teacher model's predictions serve as the ground truth or soft targets for the student model.

2. **Student Model Training:** The student model, usually a smaller and simpler neural network, is trained to replicate the teacher model's predictions. However, instead of directly mimicking the teacher model's output, the student model learns from both the ground truth labels and the soft targets provided by the teacher model.

Neural networks typically produce class probabilities by using a "softmax" output layer that converts the logit, z_i, computed for each class into a probability, q_i, by comparing z_i with the other logits.

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

where T is a temperature that is normally set to 1. Using a higher value for T produces a softer probability distribution over classes.

In the simplest form of distillation, knowledge is transferred to the distilled model by training it on a transfer set and using a soft target distribution for each case in the transfer set that is produced by using the cumbersome model with a high temperature in its softmax. The same high temperature is used when training the distilled model, but after it has been trained it uses a temperature of 1. When the correct labels are known for all or some of the transfer set, this method can be significantly improved by also training the distilled model to produce the correct labels. A weighted average of two different objective functions is used here. The first objective function is the cross entropy with the soft targets and this cross entropy is computed using the same high temperature in the softmax of the distilled model as was used for generating the soft targets from the cumbersome model. The second objective function is the cross entropy with the correct labels. This is computed using exactly the same logits in softmax of the distilled model but at a temperature of 1. Best results are generally obtained by using a considerably lower weight on the second objective function. Since the magnitudes of the gradients produced by the soft targets scale as $1/T^2$, it is important to multiply them by T^2 when using both hard and soft targets. This ensures that the relative contributions of the hard and soft targets remain roughly unchanged if the temperature used for distillation is changed while experimenting with meta-parameters.

Each case in the transfer set contributes a cross-entropy gradient, dC/dz_i, with respect to each logit, z_i of the distilled model. If the cumbersome model has logits v_i which produce soft target probabilities p_i and the transfer training is done at a temperature of T, this gradient is given by:

$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T}\left(\frac{e^{z_i/T}}{\sum_j e^{z_j/T}} - \frac{e^{v_i/T}}{\sum_j e^{v_j/T}}\right)$$

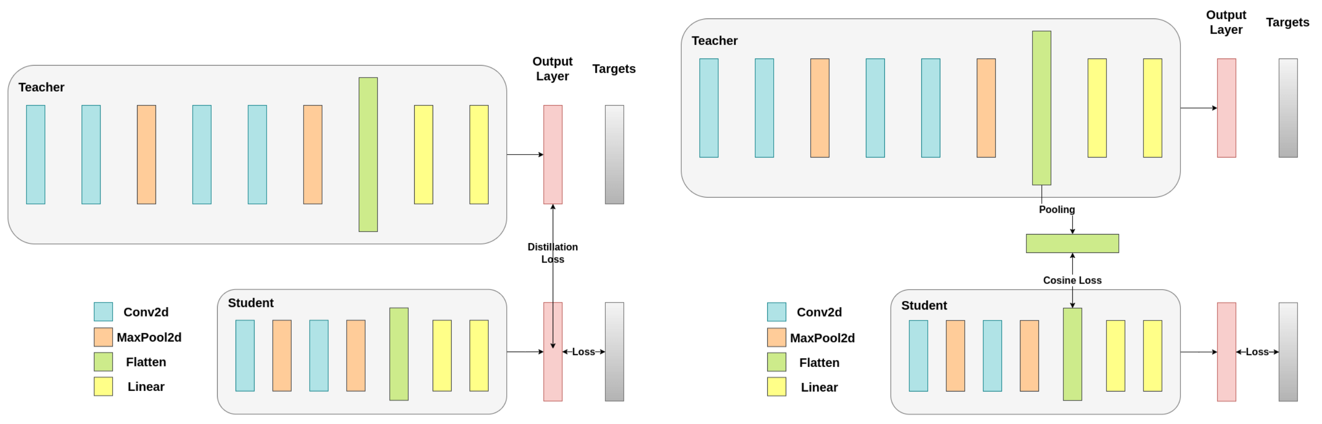
If the temperature is high compared with the magnitude of the logits, we can approximate:

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}\left(\frac{1 + z_i/T}{N + \sum_j z_j/T} - \frac{1 + v_i/T}{N + \sum_j v_j/T}\right)$$

If we now assume that the logits have been zero-meaned separately for each transfer case so that $\sum_j z_j = \sum_j v_j = 0$, the above equation simplifies to:

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{NT^2}(z_i - v_i)$$

So in the high temperature limit, distillation is equivalent to minimizing $1/2\,(z_i - v_i)^2$, provided the logits are zero-meaned separately for each transfer case. At lower temperatures, distillation pays much less attention to matching logits that are much more negative than the average. This is potentially advantageous because these logits are almost completely unconstrained by the cost function used for training



(a) Regular Knowledge Distillation (b) Representation Distillation

Figure 1. Different Variations of Knowledge Distillation [1]

the cumbersome model so they could be very noisy. On the other hand, the very negative logits may convey useful information about the knowledge acquired by the cumbersome model. When the distilled model is much too small to capture all of the knowledge in the cumbersome model, intermediate temperatures work best which strongly suggests that ignoring the large negative logits can be helpful.

The approach mentioned above is regular knowledge distillation, as shown in Figure 1a. There is another approach called representation distillation [3] [24], which tries to match the representation of hidden layers between the cumbersome model and the distilled model instead of matching their final predictions, as shown in Figure 1b.

4. Few-Shot Learning

4.1. Preliminaries

Following the definition of [14] we first define few-shot learning.

In a regular supervised machine learning setup, we focus on a learning task which can be defined as task λ with a labeled dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$ which has n data points. For training purpose, we usually split the dataset to two parts, train and test set (and optionally a validation or evaluation set but that is omitted in this definition). The datasets can be defined as D_{train} and D_{test}. Here we have:

$$D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{t} \tag{1}$$

and

$$D_{\text{test}} = \{(x_i, y_i)\}_{i=t+1}^{n} \tag{2}$$

where t denotes the number of training samples.

We optimize parameters W on the training set D_{train} and then validate its generalization on the test set D_{test}. Thus the learning problem here is to approximate the function H with parameters W:

$$y \approx H(x; W) \text{ where } (x, y) \in D_{\text{test}} \tag{3}$$

and

$$W = \arg\min_{W} \sum_{(x,y)\in D_{\text{train}}} \mathcal{L}(H(x, W), y) \tag{4}$$

If we consider the task λ as a simple image classification task where x is the input and y is the provided output label, we try to approximate the function H as defined before with parameters W. For usual case when we have sufficient training data per class, i.e., t is a large number in equation 1. But when t is small, it becomes very challenging to approximate the function and then the model ends up having poor generalization performance on the test set defined in equation 2. We can define this as a few-shot classification problem where the number of examples (also known as shots) are small to learn good model or approximate the function H.

In this study, we particularly focus on a certain few-shot learning algorithm or model called Prototypical Network [19]. Before delving deep into how the model works, we first define a few terms.

1. **N-way-K-shot**: Few-shot classification is defined in such N-way-K-shot task [7, 25] where N is the number of classes in a classification task and K is the number of chosen examples per class seen during training. Usually this K is a small number (eg. 1, 3, 5 or 10). The N-way-K-shot tasks are usually sampled from a larger dataset with classes higher in number than N. Thus we have $|D_{\text{train}}| = N \times K$
2. **Support Set**: Support set consists of a small collection of labeled examples used to teach the model about specific categories during training.

3. **Query Set**: The query set consists of the samples from the new and old class of data on which the model needs to generalize using previous knowledge and information

4. **Episodes and Episodic Training**: The whole training is organized into small batches, where each batch (episode) consists of a support set and a query set sampled randomly from the entire dataset.

4.2. Prototypical Network

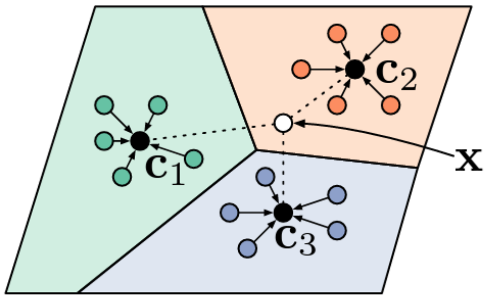


Figure 2. Prototypical Networks for Few Shot Classification scenario. Here the c_k denotes the Few-shot prototypes that are computed as the mean of embedded support set images sampled randomly for each class and x donates the query image whose classification is being done with respect to the calculated prototypes

The prototypical network computes a prototype per class which is a P dimensional vector representation as seen in Figure 2 defined as c_k. This is computed as the mean of all embeddings of support images from the class with the help of a learnable function f_ψ parameterized by learnable parameter ψ. For our experiment, this model is the backbone network that we used. More details are laid out in the next section where we describe the experiments. After the prototypes have been computed, the query samples q are classified as nearest neighbor prototypes in feature space, with respect to Euclidean distances.

The learning is done by minimizing the negative log-probability $\mathcal{J}(\psi) = -\log p_\psi(y = k \mid q)$ of the true class k via some optimization algorithm. Here the probability p_ψ is a valid probability distribution which is generated over the image classes for a single query point q using the softmax function which is computed over the distances to the calculated prototypes in the embedded vector space. Each of the training episodes is formed by choosing a subset of classes randomly from the total classes and then chooses a subset of examples in each class as the support set and then a subset of examples to act as the query set.

5. Our Approach

5.1. Definition of The Task

Given a large teacher model pretrained on a large dataset, we perform knowledge distillation from the teacher model to a small student model, but using a different dataset D. Let's name the knowledge-distilled student model A.

The same pretrained large teacher model is trained using few-shot learning on a subset of the same dataset D. Let's name this few-shot learned model B.

Our goal is to compare the performance of model A and model B on the same test set to see what hurts the performance of a model more- unavailability of in-domain data or restricted model size.

5.2. Dataset

We performed image classification task on the PathMNIST dataset which is a part of larger dataset called MedMNIST v2 [27], a large-scale MNIST-like dataset containing the collection of biomedical images including 12 datasets for 2D and 6 datasets for 3D. We focus on the specific PathMNIST dataset which contains 107,180 images of colon pathology reports and there are a total of 9 classes in the dataset. Due to time and computational resource constraints, we used images of size 128×128 instead of the highest available size 224×224. The training, validation and test splits contain 89996, 10004 and 7180 images respectively, which was predefined in the dataset.

5.3. Knowledge Distillation

We chose **ResNet50** pretrained on ImageNet1K as the teacher model and **Mobilenetv3 small** as the student model. This ResNet50 model has 1000 neurons in the final layer, but we need a final layer containing 9 neurons, corresponding to 9 classes. As we cannot use the final logits from the pretrained teacher model, our intuition was that the representation of the model before the final classification layer should be similar between the teacher and the student. So we used representation distillation. The teacher model generates a vector of size 2048 before the final classification layer while the student model generates a vector of size 1024 before the final classification layer. To match dimensionality, we projected the 1024 dim vector found from the student model to 2048 dim using a linear (or fully connected) layer. Then we performed cosine similarity between these two vectors, each having 2048 dim. The ImageNet1K classification layer of the teacher model was discarded.

The final layer of the student model contains 9 neurons with softmax activation. The image classification task is performed by the student model.

The overall architecture of our model used for knowledge distillation has been shown in Figure 3.

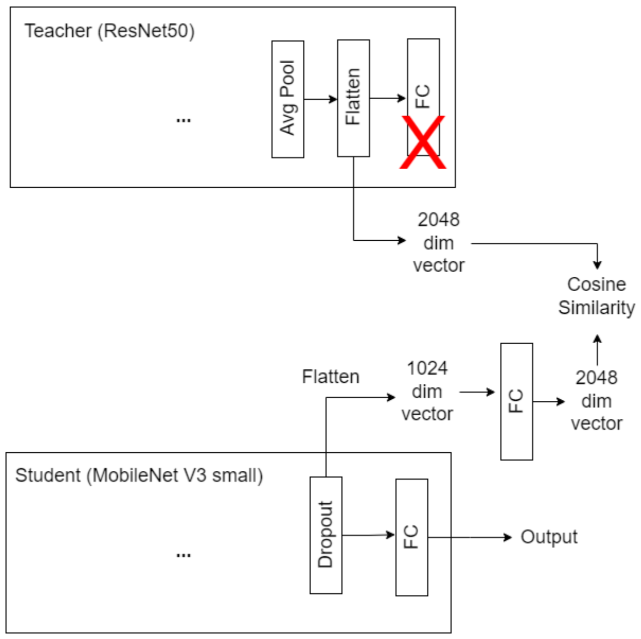


Figure 3. Knowledge Distillation Model Used in Our Experiment (all the layers before the final few layers have been omitted for clarity)

The loss function consists of two parts:

- Cross entropy loss (L_{cross}) which encourages the prediction of the student model to be as close as possible to the actual label
- Cosine embedding loss (L_{cosine}) which encourages similarity between the representations from teacher and student

The total loss is a weighted sum of these two components.

Total loss = $p \times L_{cross} + (1-p) \times L_{cosine}$, where $0 \leq p \leq 1$

Here p is a hyperparameter that controls the trade-off between the cross-entropy loss and the cosine embedding loss. We experimented with different values of p and found out that $p = 0.75$ gives the best validation loss. We used Adam optimizer with a learning rate of 1e-4 for training.

5.4. Few-Shot Learning

We randomly sampled 30 images per class to create a few-shot dataset. This step was crucial because we had a large number of images per class in the dataset. If we directly used all the images, then this experiment would not be a few-shot learning experiment. We used Prototypical Network [18] for all of our few-shot learning experiments.

We performed 5-way-5-shot training on Prototypical Network using ResNet50 as the backbone. To compare it with an extreme case, we also performed 5-way-1-shot training on Prototypical Network using ResNet18 (instead of ResNet50) as the backbone. In 5-way-1-shot setting, the large ResNet50 model overfits to the very limited training data, so we used a smaller ResNet18 model. In each case, the training involved 10000 episodes. We used Adam optimizer and set the learning rate to 8e-4 for 5-way-5-shot setup and 4e-4 for 5-way-1-shot setup setup. As randomly sampled support set is used to get the actual labels in few-shot learning, we cannot evaluate our model on the test set in the same way we did in knowledge distillation. Evaluating a model on the test set is also performed in episodes in few-shot learning, so we evaluated our model using 100000 episodes on the test set to get a good estimate of the performance of our model on actual test set.

5.5. Baselines

We compare the performance of knowledge distillation and few-shot learning with the following baselines:

1. Pretrained teacher model (ResNet50) fully fine-tuned on PathMNIST
2. Pretrained teacher model transfer-learned on PathMNIST, i.e. freezing all the layers except the classification head
3. Student model trained from scratch on PathMNIST

The knowledge-distilled student model should perform better than 3. 1 has been chosen as the skyline- all the models in our experiment should perform worse than this. 2 is expected to perform a bit inferior to 1, but the performance gap should be small.

6. Results

During training, we saved the model after the epoch when we got the lowest validation loss and this model was used for later evaluations.

Table 1 demonstrates the performance of various models including the baselines on the test set of PathMNIST dataset. The first three experiments refer to the baseline experiments. Additional information regarding batch size, learning rate and number of training epochs have been mentioned in the table for ease of understanding. And finally in experiment 5 and 6, additional information indicating what type of N-way-K-shot strategy was used in the experiment has also been mentioned in the experiment name. It is also to be noted that the highest scores have been made bold and the second highest scores have been underlined.

We observe from Table 1 that finetuning only the final layer of the large teacher model (ResNet50) can still yield

Experiment No	Experiment Name	Accuracy(%)	F1 Score
1	Finetune Teacher - all layers (batch=256, lr:1e-5, epochs=20)	91.88	0.895
2	Finetune Teacher - only final layer (batch=256, lr:1e-4, epochs=20)	**92.06**	0.886
3	Student Scratch Trained (batch=256, lr:1e-4, epochs=50)	74.81	0.713
4	Knowledge Distillation (batch=256, lr:1e-4, epochs=50)	83.97	0.796
5	FSL - Proto - 5/5	89.66	**0.896**
6	FSL - Proto - 5/1	82.77	0.828

Table 1. Accuracy and F1 Score on Test Set with Different Methods.

us very good accuracy and F1 score, even better than fine-tuning all the layers of the teacher network. This model has the highest accuracy among all other models. This might suggest that the final layer carries significant representational power for the task at hand. Experiment 1 and 5 both demonstrate strong performance in terms of the F1 score (0.895 and 0.896, respectively), highlighting that both full-network fine-tuning and 5-way-5 shot few-shot learning can effectively leverage the learned features from their respective training schemes. Experiment 1 and 2 show similar performance in terms of both accuracy and F1 score, as expected.

The result from experiment 4 with representation based knowledge distillation while not matching the top performance of fine-tuned models, still outperforms the student model trained from scratch (experiment 3). This suggests that transferring knowledge from a larger pretrained model (ResNet50) to a smaller, less complex model (MobileNetV3 small) can significantly boost performance of the student model. This inference could be crucial for application domains with constrained resources in terms of computing and power such as edge devices.

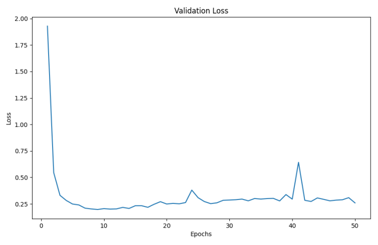
The drop in the performance between the models of experiment 5 and 6 is expected and suggests that reducing the number of examples per class in the support set leads to loss in performance.

One of the interesting insights we obtained from the experiment is evident in the results of experiment 4, 5 and 6. In 5-way 5-shot network in experiment 5, the accuracy was 89.66% which was the second highest just short by 2.4 points from the best performing model (experiment 2) and the F1 score was 0.896 which was the best score among all the trained models, even beating the finetuned teacher models from experiment 1 and 2. This model surpassed the knowledge distilled student model both in terms of accuracy and F1-score. The 5-way 1-shot trained teacher model fell short closely with respect to the student model by merely 1.2 points but to our surprise performed better in terms of F1 score. It is interesting because in experiment 6, we opted to use a lightweight ResNet18 model as the regular teacher model (ResNet50) was drastically overfitting. This lightweight model even with 1-shot setting is still able to surprass the student model which was knowledge distilled from a larger teacher model and even trained from scratch in experiment 3. From our experimental setting with this particular PathMNIST medical domain dataset, we can infer that restricted model size (experiment 4) hurts performance more than limited availability of in-domain data (experiment 5 and 6).

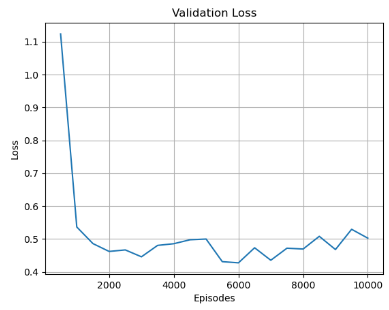
We also add the learning curves of experiment 4, 5 and 6 in Figure 4 and 5.

From the plot of validation loss of experiment 4 in Figure 4a, we can see that the training is somewhat stable. The loss sharply drops down at the start of training for around 10 epochs. Then the loss remains more or less stable with a slight upward trend. The training was stopped at 50 epochs due to computational resource issues and also observing from the learning curve that the model might have overfitted if trained further. From the validation accuracy plots of experiment 4 in 5a, we see that the accuracy similarly jumps high at the start of the experiment then keeps on becoming stable. We also experimented with other learning rates and other loss hyperparameter p as mentioned in previous section and found out the specific hyperparamters that leads to stable training for knowledge distillation and reported them here.

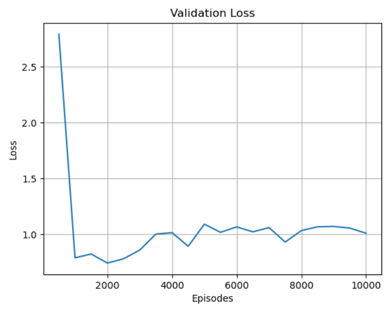
We can see from figure 4b, for few-shot learning with 5-shot method, the validation loss starts to show a rising trend when it hits 10000 episodes. Similar case is seen for the loss curve of figure 4c which was the extreme case of 1-shot learning. This is where we stopped training the model because further training may have resulted in overfitting to the dataset. The validation accuracy plot for 5-way 5-shot training in Figure 5b shows that the learning is quite stable as the training goes on with no abrupt downward trend. We also notice that it has reached saturation quite early during training after around 3000 episodes. For validation accuracy plot of 5-way 1-shot training in Figure 5c, we observe that the accuracy reaches a peak after around 2000 episodes and then sharply decreases down, after which point the curve does not fluctuate much. We experimented with other learning rate setups and the ones we report here turned out to have the most stable training among all other hyperparameters.



(a) Validation Loss - Exp 4

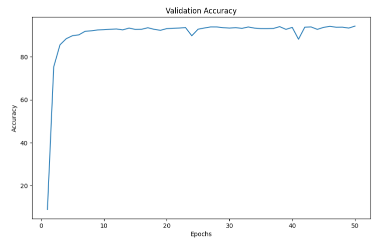


(b) Validation Loss - Exp 5

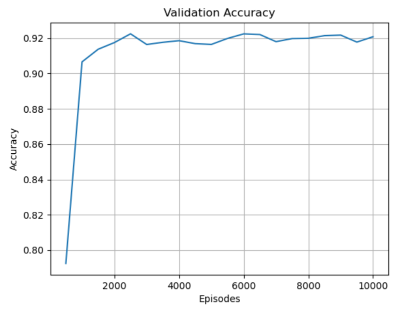


(c) Validation Loss - Exp 6

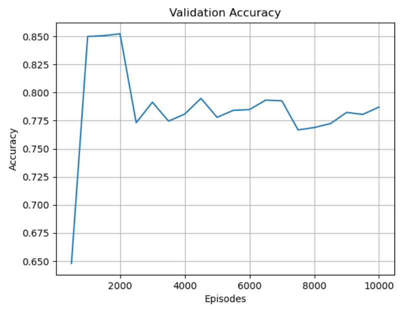
Figure 4. Validation Loss of Experiment 4 (Knowledge Distillation), Experiment 5(5-5 Few Shot Learning) and Experiment 6(5-1 Few Shot Learning)



(a) Validation Accuracy - Exp 4



(b) Validation Accuracy - Exp 5



(c) Validation Accuracy - Exp 6

Figure 5. Validation Accuracy of Experiment 4 (Knowledge Distillation), Experiment 5(5-5 Few Shot Learning) and Experiment 6(5-1 Few Shot Learning)

7. Conclusion and Future Work

We compared two resource-constrained settings in an image classification task, one is knowledge distillation and another is few-shot learning. The results suggest that the size of the model (or the number of parameters of the model) is very crucial for learning, even more important than the size of in-domain dataset. A more detailed study using some other datasets and different network architectures (such as transformer based models) is needed to validate such observation. Further experiments can be performed on other modalities (such as text, audio etc) to see whether similar behavior is observed there as well.

References

[1] Knowledge distillation tutorial. https://pytorch.org/tutorials/beginner/knowledge_distillation_tutorial.html. Accessed: 2024-04-30. 3

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[3] Romero Adriana, Ballas Nicolas, K Samira Ebrahimi, Chassang Antoine, Gatta Carlo, and Bengio Yoshua. Fitnets: Hints for thin deep nets. *Proc. ICLR*, 2(3):1, 2015. 3

[4] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[5] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019. 2

[6] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021. 1

[7] 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. PMLR, 2017. 3

[8] Jianping Gou, Liyuan Sun, Baosheng Yu, Shaohua Wan, and Dacheng Tao. Hierarchical multi-attention transfer for knowledge distillation. *ACM Transactions on Multimedia*

Computing, Communications and Applications, 20(2):1–20, 2023. 1

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2

[10] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann. Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 709–718. IEEE, 2018. 1

[11] Yuang Liu, Wei Zhang, and Jun Wang. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113, 2020. 1

[12] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019. 1

[13] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 2

[14] Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning. *arXiv preprint arXiv:2203.04291*, 2022. 2, 3

[15] Haotong Qin, Zhongang Cai, Mingyuan Zhang, Yifu Ding, Haiyu Zhao, Shuai Yi, Xianglong Liu, and Hao Su. Bipointnet: Binary neural network for point clouds. *arXiv preprint arXiv:2010.05501*, 2020. 1

[16] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1

[17] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 1

[18] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 2, 5

[19] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 3

[20] Yisheng Song, Ting Wang, Puyu Cai, Subrota K Mondal, and Jyoti Prakash Sahoo. A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities. *ACM Computing Surveys*, 2023. 1

[21] Christopher Subia-Waud and Srinandan Dasmahapatra. Weight fixing networks. In *European Conference on Computer Vision*, pages 415–431. Springer, 2022. 1

[22] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018. 2

[23] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

[24] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, 2019. 3

[25] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016. 3

[26] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens Van Der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019. 2

[27] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023. 4

[28] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294, 2017. 1

[29] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294, 2017. 1

[30] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016. 1