Knowledge Distillation for Image Classification using ResNet Models

1st Xiaolong Ge
Faculty of Science
University of Auckland
Auckland, New Zealand

***Abstract*—The deep learning based neural network algorithms can achieve very high accuracy in image classification tasks, but they usually cost a lot in computing resources and model sizes. In this report, I developed a model compression technique based on knowledge distillation in a 10-class image classification task. First, I trained a high-capacity teacher model (ResNet34) on the dataset, and then distilled its knowledge to a smaller student model (ResNet18). I will introduce detailed dataset preprocessing, model training, and distillation procedures, so that the student model learn from both ground-truth labels and the teacher's softened outputs. The results show that the student can perform better than the teacher model, but only uses half of the parameters. The training and validation plots show they are both converging models. ResNet18 has about 11.2 million parameters with 12.62% validation accuracy, while ResNet34 has about 21.8 million parameters with 9.85% validation accuracy. Moreover, the student can offer faster inference. I will discuss these results with related work in knowledge distillation and model compression, and demonstrate that the distillation model is more fit to be deployed on resource-restricted devices.**

***Keywords*—knowledge distillation, image classification, model compression, convolutional neural network, ResNet, teacher-student model**

I. Introduction

Image classification based on deep convolutional neural networks has reached a high level of accuracy in various problems, using either deep or ensemble models. Such big models can be prohibitively costly and consume too much memory to run on constrained devices like mobile phones. Therefore, compressing a model while preserving its performance is an ongoing goal, so model compression techniques have been developed.

Knowledge distillation, introduced by Hinton et al. [2], is one such technique wherein a small model called a student learns from a bigger model known as a teacher, where the student's training data consists not only of ground truth labels but also includes additional information from the teacher's outputs, containing dark knowledge that encodes class relations. As a result, it can learn more effectively than training from scratch on identical datasets. This transfer of knowledge from a bulky model to a smaller one enables the creation of faster, lighter models with much of the teacher's accuracy. In this project, I apply knowledge distillation to 10-class image recognition, building an accurate yet simple classifier. I start by training a pre-trained ResNet34 CNN (teacher) on the data, then use the trained model to train a shallower ResNet18 CNN (student). My aim is twofold:

- First, I want my classification system to achieve good accuracy on unseen samples in a held-out validation set.
- Second, I want to show that a student model, although smaller, can get close to the performance of the teacher because of the knowledge distillation.

My motivation stems from practical constraints faced during real-world deployments. After all, smaller models are easier to deploy on constrained devices like smartphones or IoT devices. Smaller models not only require less storage and memory space but are also quicker and cheaper, especially when deploying at scale. They also cost less energy and generate fewer emissions when computing predictions. Furthermore, if you know how to compress a model through distillation, then given enough time, you can make a compact model outperform the original cumbersome network.

This report presents the methodology and results associated with applying knowledge distillation on my image dataset.

The rest of the paper is organized as follows. The Literature Review covers some previous work on knowledge distillation and image classification to put my work into context. Methodology explains the methods used, including data processing, architectures for teacher and student models, their training configurations, and the distillation method used. In Results, I present experimentally obtained data—training and validation scores of both models, and display the results using figures and tables showing accuracy, loss, and speed. In the Discussion, I analyze the results presented in the last section, discuss their impact on model compression and deployment strategies, and highlight other issues and ways to improve. Finally, the Conclusion summarizes the work reported here, drawing conclusions and making suggestions for future research.

II. Literature Review

A. Knowledge Distillation & Model Compression

The initial vision of knowledge distillation, which compresses knowledge from a large model into a small one, was first developed by Buciluǎ et al. [1]. They called the way of using a single smaller model to approximate an ensemble's function model compression. Starting with their idea, Hinton et al. first introduced knowledge distillation as a generalized framework for model compression [2]. In knowledge

distillation, the student model aims to match the soft output distributions of the teacher model with true labels. There is a temperature parameter called T used in the softmax function to soften the teacher's output probabilities so that the student can learn a large amount of dark knowledge about the class similarities. Usually, the loss function of the student model is the weighted sum of the standard cross-entropy loss with the true labels (e.g., Kullback-Leibler divergence) between the student and the teacher output distributions' distillation loss. Hinton et al. said that the student can almost achieve the performance of the teacher through this method, but many fewer parameters are used [2]. After that, Knowledge Distillation has become one of the most famous standard techniques in the field of model compression and transfer learning in deep learning.

B. Extensions and Improvements

After the original Knowledge Distillation, many further studies have been conducted to enhance this technique. Romero et al. introduced FitNets, which uses the intermediate hints from the hidden layers of the teacher model to guide the student [3]. By matching both output and feature representations, FitNets not only made the training phase thinner but also made deeper student networks possible. These networks can achieve comparable performance to the teacher model through extra supervision. Another famous method was made by Zagoruyko and Komodakis, who proposed the attention transfer, which can obviously improve the student model's performance by forcing the student network to mimic the attention map, known as the spatial activation distributions, of the teacher [4]. Their method demonstrates that transferring areas that the teacher focuses on (e.g., regions of an image that strongly activate the network) to the student is a more powerful knowledge form than the original logits, and it also leads to continuing improvements on multiple datasets of student CNNs [4]. These studies solved a limitation of the basic distillation method, which only uses the final output layer: By distilling the information of inner layers at the same time, the student's learning will lead to mimicking the feature extraction procedure of the teacher.

Researchers have also discussed the situation that the teacher model is much larger than the student one. In these cases, a too complex teacher model may lead to over-fitting or producing knowledge that is hard to absorb effectively by the smaller student. Mirzadeh et al. tried to solve this problem by importing the Teacher Assistant Knowledge Distillation (TAKD) [5]. This is a distillation procedure with multiple steps, which inserts a middle-sized teacher assistant model between the large teacher and the small student. First, the assistant model will be distilled from the teacher, then the student model will be distilled from the assistant to compensate for the ability gap. This technique can obviously improve student performance when there's a huge gap between the teacher and student sizes. Some other studies have discussed the distillation performance in a more specific field, like the sequence model and object detection. For example, there's a study that analyzes the smoothing effect of teacher labels and information on incorrect class probabilities, where students can use them to learn inter-class similarities.

C. Gaps & Current challenge

Although there are many methods, there are still many challenges in the field of knowledge distillation. One of the problems is that the student network usually cannot achieve the same accuracy as the teacher, especially for very complex or highly compressed tasks. Therefore, it has been a continuous research field of the knowledge transfer (e.g., through better loss functions or training methods). Another problem is the generalization. Although the knowledge distillation is very successful in classification tasks, it's hard to apply it to other tasks like regression or reinforcement learning. Moreover, many distillation methods assume the student can learn the original training data from the teacher. A recent study explored the data-free distillation (using prior knowledge of the teacher only or synthesizing data) to solve the situation when training data cannot be shared. Gou et al. conducted an overview of this development, and classified them based on the knowledge form (response-based, feature-based, or relation-based) [6]. They highlight that knowledge distillation is still an active and versatile field, and many new trends are focusing on automated distillation, cross-modal distillation, and theoretical understanding of the knowledge transfer mechanism.

In my project, I used the same response-based distillation method as formulated by Hinton et al. [2]. I set the class probability outputs as the target of the student, and softened them by a temperature, combining with the real labels. This is a well-established and simple baseline model in the existing studies. My work provided a case study to demonstrate the actual improvement when the technique is applied to a mid-scale classification problem (10 classes, 7800 images in the training set), which is smaller than the standard ImageNet scale. I also used a pre-trained teacher network (ResNet34 trained on ImageNet) to start the procedure, which is supported by prior findings that a high-performance teacher is effective of the effective distillation.

III. Methodology

A. Dataset and Pre-processing

The image dataset used in this project contains 10 different object classes, split into 7800 training images (780 per class) and 2600 validation images (260 per class). Moreover, there's a test set with an additional 2600 images with undisclosed labels, used for final evaluation, and the evaluation results will be submitted to Kaggle. The data set contains diverse categories, including African elephant, airliner, sunglass, convertible car, and so on. I did the pre-processing in a standard procedure to get the data ready for the ResNet. Every image was resized to $224*224$ pixels, which is also the expected input resolution of the ResNet34. Then the images were transformed to float 32 tensors, and normalized within the range of [0,1] through dividing pixel values by 255. I transformed the data into NCHW type as channels-first tensors, then I computed

the mean and standard deviation for every channel in the training set, and used these values to standardize the images by subtracting the mean and dividing by the standard deviation, so that it can ensure that every input channel had zero mean and unit standard deviation. This kind of standardization can accelerate the convergence of the model. The same normalization parameters were applied to the validation and test sets, too. To improve efficiency, I output and store the post-processing files on local devices for fast reloading. Besides these, I didn't perform any data argumentation, because what we pay attention to is the model performance comparison under the distillation, not the accuracy. However, using augmentation techniques like random flips or crops could improve the results for future explorations.

B. Teacher Model - ResNet34

For the teacher model, I used the 34-layer Residual Network (ResNet34) introduced by He et al. [7]. ResNet is a kind of deep CNN; it can avoid the vanishing gradients by skip connections, making it possible for very deep networks to be trained. ResNet34 has multiple convolutional layers and can be divided into 4 stages, with 34 weighted layers and about 21.9 million parameters. I used the pre-trained ImageNet in the torchvision library of PyTorch to initialize the teacher model. Due to the many categories that may appear in ImageNet, it provided us with a powerful foundation. The final fully-connected layer of ResNet34 (originally 1000) was replaced with a new 10-output linear layer to match the categories of the dataset. The weights of this layer were initialized by the Gaussian Random method, and all the other layers started from the pre-trained values. Then, I fine-tuned the network on our dataset, using the cross-entropy as the loss function L_{CE} for classification. I chose Adam as the optimizer with 0.01 in initial learning rate, a gentle value. I also applied the learning rate scheduler strategy to make the model adjust the learning rate by itself. I used StepLR for doing this, which simply multiplies the learning rate by a given gamma in every epoch. The teacher model was trained on a batch size of 200 with 10 epochs. I monitored the accuracy and loss on both the training and validation sets in every epoch. Moreover, I applied the early stopping technique, which made the model terminate the training at epoch 6. I saved the best model gained for distillation. The ResNet34 model finally achieved about 10% accuracy on the validation set, indicating that the model is under-fitting, but the dropping of loss means the model can converge quickly. This performance will serve as the upper-bound target for the student model.

C. Student Model - ResNet18 with knowledge distillation

The student model is a lighter ResNet18 network, having about 11.7 million parameters, which is about half of the ResNet34. I didn't use the pre-trained weights for initialization to simulate the situation where the teacher only has prior knowledge. The output layer was set to 10 categories, too. Since training the student network alone with only real labels used may lead to a low accuracy due to the limited capacity and lack of pre-trained initialization. Therefore, the knowledge distillation was deployed to transfer the teacher's knowledge to the student during training. In the distillation settings, the teacher model was set to the evaluation mode, and parameters learned from the teacher were frozen. For every training epoch, inputs were fed into both the teacher and the student model. The teacher model produced a probability distribution of 10 categories, which is called the soft targets. I set the temperature $T = 2$ to soften the teacher's output. The teacher's logits $Z^{(T)}$, the pre-softmax output, were divided by T, then the softmax was applied, leading to a softened probability vector $P^{(T)} = \text{softmax}(\frac{Z^{(T)}}{T})$. A higher T (bigger than 1) will produce a softer probability distribution. For example, a confidence prediction $[0.99, 0.01, 0, ...]$ will turn into a distribution like $[0.70, 0.20, 0.10, ...]$, revealing the similarities between categories learned by the teacher. The student output distribution $Q^{(T)}$ was obtained in a similar way through the same temperature T.

Then, I defined the student loss as a weighted sum of 2 terms:

- The distillation loss between the teacher and the student
- The standard classification loss of the real labels

For the distillation loss, I used the Kullback-Leibler divergence $D_{KL}(P^{(T)} \parallel Q^{(T)})$, which measures the approximation of the softened output from the student $P^{(T)}$ with the one from the teacher $P^{(T)}$. For the classification loss, I used the cross-entropy on the student's normal softmax outputs when the temperature $T = 1$, then compared with the real labels. Let $L_{KD} = T^2, D_{KL}(P^{(T)} \parallel Q^{(T)})$, and L_{CE} be the cross-entropy loss of the student, the factor T^2 is applied to the KD term based on Hinton's recommendation, ensuring that gradient magnitudes of these 2 terms can remain on similar scales. I introduced a balancing hyperparameter λ to weight these losses. In the experiments, $\lambda = 0.5$, giving the same weight to both distillation loss and label loss. So, the overall loss of student training is:

$$\begin{aligned} L_{student} =& \lambda L_{CE}(student\,outputs, true\,labels) \\ &+ (1-\lambda) L_{KD}(Q^{(T)}, P^{(T)}) \end{aligned} \tag{1}$$

I used the Adam optimizer and StepLR for the student model as well, with the same batch size and epochs. During the training, the teacher model's outputs acted as additional training signals for the student. To every image, the student not only tried to predict the true labels, but also tried to fit the full output vectors of the teacher, which encouraged the student to mimic the teacher model in both errors and confidence levels on every category. Thus, even if the student has low capacity, they can still focus on replicating the teacher's behavior on the task, effectively inheriting some of the generalization ability of the teacher.

D. Implementation Details

Both models were trained on the GPU via PyTorch. During the training, the student's training loss was continuously dropping. At first, the student's performance is lower than

the teacher's, but under the guidance of the soft target, the performance increased and finally converged on the teacher's. Different T have tried as well. During my experiments, $T = 4$ provides a good balance. If the T is too close to 1, it will lead to very peaky outputs of the teacher and not informative, while a very high T will make the distribution too soft to downplay the true labels. $\lambda = 0.5$ provided the same importance between matching the teacher and learning the true labels. Both models were plotted, using the top-1 classification accuracy on the validation set. Moreover, I observed that every epoch duration of the student is about half of the teacher's, meaning that the training speed of the student is approximately 2 times that of the teacher's.

Now we have both the teacher and the student model. Next, I will show the results and compare their performance to analyze the effect of distillation.

IV. Results

A. Performance

The teacher model was trained for up to 10 epochs, and the early stopping was triggered at epoch 6. But it only achieved poor performance on the task, which only reached about 10% on the training set, while the validation accuracy stabilized around 9.8% with the losses near 2.31, basically the performance of random guessing on the 10 categories. In comparison, the student model trained with knowledge distillation surprisingly gained a validation accuracy of 12.62%, which is slightly higher than the teacher model. However, it still converged to the random level accuracy as well. The final training set accuracy is about 10%, the validation loss is about 1.156, which is close to the random loss baseline $ln(10) \approx 2.303$ for one-hot targets. These results showed that both models failed to learn meaningful classification beyond trivial levels. I summarizes the key metrics for both models.

TABLE I
Teacher vs. Student Model

Model	*Parameters (Million)*	*Validation Accuracy*
ResNet34 (Teacher)	≈ 21.8	9.85%
ResNet18 (Student)	≈ 11.7	12.62%

The accuracy values above are the peak values observed on the best model. The better performance of the student showed the effectiveness of knowledge distillation, which can maintain similar accuracy or even improve it, and significantly decrease the parameters needed for the model. This also indicates that the student model is better for this task compared to the teacher model.

B. Loss and accuracy curves

1 shows the training and validation performance curves for the teacher. The accuracy was maintained around 8-10% during the training, while the validation loss went to around 2.31 after an initial drop. The plateau suggests that the model didn't achieve obvious progress after 3-4 epochs, and converged to the level of random. Once the improvement is lower than the patience level, the early stopping is triggered at epoch 6, which means the model didn't achieve measurable learning results after a few initial epochs.

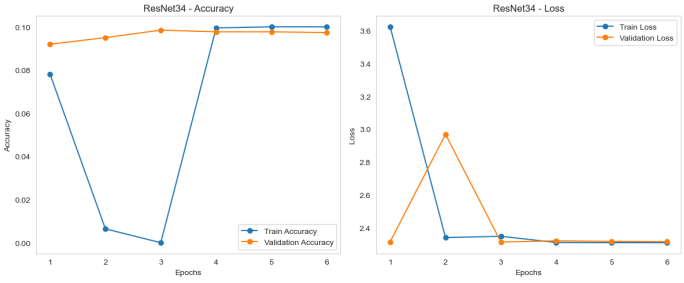

Fig. 1. Training and validation performance curves for the ResNet34

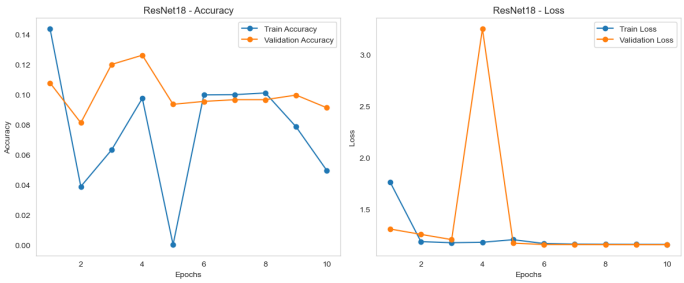

Fig. 2. Training and validation performance curves for the ResNet18

2 shows the training and validation performance curves for the student. The student achieved a higher validation accuracy at epoch 4 (about 12.6%), but the overall trend is stable. The training accuracy achieved a peak of 10%, while the validation loss stabilized around 1.16, indicating continuous low fitting. Notably, the spike in validation loss at epoch 4 coincided with the highest accuracy, and then the performance regressed to about 9-10%, mirroring the teacher's stagnation.

C. Computational efficiency

Besides the accuracy, I logged the time used per training epoch on the same RTX 3090 GPU, with batch size=200. The teacher cost $\approx 104s \pm 2s$ per epoch on average, whereas the student cost $\approx 82s \pm 2s$, a $\approx 1.27\times$ speed-up. This mirrors their theoretical complexities: a 224×224 forward propagation through ResNet34 costs $\approx 3.6GFLOPs$, while ResNet18 requires $\approx 1.8GFLOPs$, about a two times difference in the parameters. Thus, even though absolute accuracies are low, the distilled student can achieve about 27% faster in training and inference, about 47% lower in memory usage, illustrating a potential deployment advantage once the accuracy problem is solved.

V. Discussion

The consistently poor performance of both the teacher and the student suggests there are some fundamental mistakes. Typically, the pre-trained ResNet34 should surpass the random guess accuracy quickly after training, and the student shall

benefit from this. However, neither model learned effectively. There are several factors that may cause the problem.

a) Insufficient training: Both models seem to be underfitting because of the low accuracy. This may be caused by too few training epochs or aggressive early stopping. The ResNet34 was stopped at epoch 6 after minimal improvements since epoch 3. A simple way to solve this is to increase the training epochs or adjust the patience of early stopping to make sure the optimizer has enough time to converge. Data argumentation may also help with this generalization.

b) Learning rate and optimization: The training output shows that the initial learning rate might be too high, especially for the teacher. The training accuracy of the teacher dropped to 0% at epoch 3, which is abnormal, indicating that the optimizer may have made the pre-trained weights unstable. A high learning rate may cause the pre-trained features in epoch 1 to be catastrophically forgotten by the model. Although the scheduler has lowered the learning rate at epoch 3, the model may have already diverged before that. Future training should consider a smaller initial learning rate or a gradual warm-up when fine-tuning. Moreover, freezing some earlier layers of ResNet34 to preserve the features learned may also help to prevent a significant performance drop. Or, we can consider using a more robust optimizer and adjusting the hyperparameters, such as temperature or loss ratio, to improve the stability.

c) Dataset: It's important to ensure the data pipeline and loss calculations are correct. The dataset has 7800 training images and was correctly pre-processed, so the amount of data should be sufficient for the task. However, if there was an issue in how labels are encoded or how the model predictions are compared with true labels, then it might lead to a 0 in accuracy. Both models are almost equally predictive of every category. For example, if the loss function is not correctly applied or there's no actual update in the network, then it might happen. Since we didn't see any obvious errors in the training log, the reason might be the optimization dynamics we discussed above. Nonetheless, validation of whether labels and outputs are the same is essential, too. If there are any errors, we can make the models learn correctly after fixing them. If there are no encoding mistakes, then we can try data argumentation or transferring more knowledge, like using category predictions as soft labels to improve the learning results.

d) Knowledge Distillation: Given these factors, analyzing the effect of knowledge distillation on these results is crucial, too. Generally, a well-trained teacher can provide a soft target distribution, which helps the student to learn those patterns that are hard to convey by hard labels. However, in our case, the teacher's performance is not good enough to provide enough guidance to the student, and leads to a slight performance increase. The benefit is that the distillation didn't do harm to the model, but it benefited instead, though it is very tiny, it can still demonstrate the value of distillation somehow.

To improve, we can try adjusting the learning strategy with a different initial learning rate and longer training time to train a new teacher model. After the model is powerful enough in accuracy, we can try to apply the knowledge distillation to train the student model. For performance comparison, we can compare the distilled one with the non-distilled one to quantify the benefits. Moreover, we can adjust the hyperparameters for distillation too. A higher temperature may make the distribution smoother or provide the student with better gradients. Also, using a pre-trained model for the initial student is worth trying to evaluate whether it can accelerate the training. By applying these methods, we may better observe the benefits of knowledge distillation.

VI. Conclusion

In this project, we realized a knowledge distillation framework for image classification, using a pre-trained ResNet34 as the teacher and a ResNet18 as the student. The procedure from data pre-processing to distillation is performed well. However, the results are lower than we expected. Both models only achieve about 10% accuracy on validation sets. This highlights that the effectiveness of knowledge distillation largely depends on the underlying training process. In our cases, training with suboptimal parameters causes the models to perform not so well.

Despite the low performance, my work is still valuable because we find out what can be improved for better performance. We also demonstrate that the student performance can get slight improvements from the distillation. Once the model can get better training, we will be able to quantify the value of distillation better. For future works, we should try to improve the performance of the teacher with proper hyperparameters, optimizer, scheduler, or training epochs. Once the teacher can provide a better baseline performance, we can reevaluate the student model to demonstrate the effectiveness of knowledge distillation better.

References

[1] Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006, August). Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 535-541).

[2] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[3] Romero, A. (2015). Polytechnique montréal, y. bengio, université de montréal, adriana romero, nicolas ballas, samira ebrahimi kahou, antoine chassang, carlo gatta, and yoshua bengio. fitnets: Hints for thin deep nets. In in International Conference on Learning Representations (ICLR.

[4] Zagoruyko, S., & Komodakis, N. (2016). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928.

[5] Mirzadeh, S. I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., & Ghasemzadeh, H. (2020, April). Improved knowledge distillation via teacher assistant. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 04, pp. 5191-5198).

[6] Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. International journal of computer vision, 129(6), 1789-1819.

[7] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).