Image Classification with Knowledge Distillation

Ben Nguyen
Auckland, New Zealand
nguyenthaiminh1201@gmail.com

***Abstract*—This project explores the application of Knowledge Distillation in image classification. A ResNet34 model pretrained on ImageNet is used as the teacher, and a ResNet18 model is trained as the student without pretrained weights. The results demonstrate that the student model achieves competitive classification accuracy while maintaining a significantly smaller size. This project highlights the potential of knowledge distillation for deploying deep learning models in resource-limited environments.**

I. Introduction

The data is not only numbers and text; the image can help us to get much useful information because pictures and videos are everywhere. The artificial intelligence field of computer vision makes it possible for computers and other systems to extract useful data from digital photos, movies, and other visual inputs. The science and technology behind seeing machines is what makes them so. As a branch of science, computer vision aims to build computer vision systems using its theories and models. Image classification is one of the most important computer vision tasks because of its critical importance in modern technologies. It includes labelling or tagging a complete image based on previously collected training data from images that have already been categorised. Although the procedure might seem straightforward at first, it actually involves analysing images at the pixel level to identify the best label for the entire image. This gives us useful information and insights that help us make defensible choices and implementable solutions.

More complex patterns and subtleties in data may typically be captured by larger models with more parameters. Larger models are therefore better able to learn finer details, recognise more complex pixel relationships, and represent complex features like textures, shapes, and lighting variations in computer vision tasks. This results in improved performance on tasks like object detection, image classification, and segmentation. Larger models, however, might need a lot more processing power for both inference and training. This entails higher energy consumption, longer training periods, and the requirement for strong GPUs or TPUs.

Knowledge distillation could resolve many problems related to the model's size. The goal is to significantly reduce training and storage costs by compressing a large (teacher) model into a smaller (student) model with a more effective design without reducing performance.

This project uses PyTorch to implement Knowledge distillation, with ResNet34 serving as the teacher model, transferring knowledge to the ResNet18 student model, and comparing their performance in image classification tasks.

II. Literature Review

Knowledge distillation (KD) was proposed as an approach that enables a student network to learn the soft targets output by a teacher network [1] in 2015. The basic goal of knowledge distillation is to train a smaller model, called a student model, to do the same tasks as the larger model, called a teacher model. The teacher model is frequently trained on a particular task and dataset in order to get state-of-the-art performance. However, the student model is more computationally efficient and smaller, which makes it appropriate for deployment on devices with limited resources. By transferring knowledge from the teacher model to the student model, we can enhance the student model's functionality without significantly increasing its complexity. The teacher model's predictions are used to train the student model, giving it more information and improving its performance. Soft target training, attention transfer, and feature mimicry are examples of knowledge transfer strategies.

The lightweight student model uses the loss function to replicate the output generated by the teacher model in the distillation loss. Both the cross-entropy loss between the student's and teacher's outputs (both following temperature scaling) and the cross-entropy loss between the student's output and the real labels are included in the weighted total.

$$\begin{aligned} L_{distil} = \alpha \cdot T^2 \cdot \text{cross entropy}(y_{student}, y_{teacher}/T) \\ +(1-\alpha) \cdot \text{cross entropy}(y_{student}, t) \end{aligned} \quad (1)$$

where y_student is the student model output, y_teacher is the teacher model output, t is the true labels, $\alpha$is a hyperparameter that regulates the weight given to each term, and T is the temperature parameter used for temperature scaling. Another crucial aspect of knowledge distillation is the application of temperature scaling to soften the soft logits generated by the softmax function. The softmax function is then used after the logits have been divided by a temperature parameter T. The ”softness” of the probability distribution is determined by the temperature parameter; softer distributions are produced at higher temperatures. Additionally, a hyperparameter known as alpha in the distillation loss controls the weight given to each phrase. The relative importance of the two cross-entropy losses is determined by the value of alpha in the distillation loss function. In reality, alpha is typically set between 0.5 and 1, where a higher value indicates that the teacher model's output is given more weight. The student model learns to generate outputs that are accurate and comparable to those of the teacher model by reducing the distillation loss. This can be useful in situations where a smaller model is preferable due to restricted computational resources, but the larger model's accuracy is still needed.

Bucila et al. [3] suggested a method for simulating the output of an ensemble of models in order to train a single neural network without significant loss in performance. Ba and Caruana [2] adopt that idea and had success in training shallow neural nets to mimic deeper models, suggesting that there probably exist better algorithms for training shallow feed-forward nets than those currently available. Hinton et al. [1] suggested knowledge distillation as a broader application of [3], which uses the teacher model's prediction as a "soft label" and suggests temperature cross entropy loss rather than L2 loss.

In order to overcome information loss, particularly when the teacher's predictions are imprecise, Guoqiang et al. [5] suggest a logit-based knowledge distillation method that directly uses the teacher's raw logits. This experiment demonstrates that it enhances model accuracy and generalisation, leading to a considerable improvement in performance in both picture classification and object recognition tasks. Multiloss Joint Gradient Control Knowledge Distillation (MJKD), a unique knowledge distillation technique proposed by Wei et al. [6], works by efficiently merging gradient control with feature- and logit-based knowledge distillation techniques. The accuracy of MJKD on Tiny-ImageNet for the ResNet18 MobileNetV2 pair is 63.53%.

This study focuses on the fundamental idea of knowledge distillation, specifically the process of training a smaller student model with the prediction outputs (also known as "soft labels") produced by a larger, pretrained teacher model. This method focuses on learning from the teacher's softening probability distribution across courses, which captures more complex information than hard labels alone, as opposed to transferring intermediate features or attention maps. In order to do this, the training objective takes into account the temperature cross-entropy loss and applies a temperature-scaled softmax to both the teacher and student outputs. This straightforward configuration highlights knowledge distillation's capacity to condense model knowledge while preserving performance, offering a straightforward but powerful starting point.

III. Methodology

In this section, we will discuss the methodology used in our image classification task with knowledge distillation. The process involves Data Preprocessing, Teacher Model Training, and Student Model Training.

A. Data Preprocessing

The dataset includes 3 folders: train, validation, and test. Each train and validation folder contains subfolders for each class of images. There are 10 classes, including african elephant, airliner, banana, convertible car, golden retriever, goldfish, parachute, rugby, sunglasses, and tiger cat. Each class has 780 images and 260 images for training and validation, respectively, and 2600 images for test with no label, which are used to evaluate the final performance.

The preprocessing steps are as follows:

- **Loading Raw Images**: Images and their corresponding labels were initially loaded from the dataset into the correct train, validation and test set using appropriate data loaders. This step ensured that the raw data were correctly paired and ready for preprocessing.
- **Resizing and Cropping Images**: All images were resized to a fixed resolution of 224 x 224 pixels to maintain consistency and meet the input requirements of standard CNN architectures.
- **Normalisation**: The image pixel values were converted to float32 type and scaled to a range of [0, 1] by dividing each value by 255. This normalisation improves numerical stability during model training.
- **Converting to Tensor Format**: The data was rearranged from the default HWC (height, width, channels) format to NCHW format, which is required by PyTorch. The images and labels were then converted into PyTorch tensors for efficient processing.
- **Per-Channel Standardization**: Standardization was applied using the mean and standard deviation computed from the training set. This centers the data and ensures consistent scale across all input features.
- **Saving Preprocess Sets**: To speed up future data loading and avoid repeated preprocessing, the transformed tensors were saved as .pt files using PyTorch's torch.save() function.

B. Teacher Model Training (ResNet34)

In this project, I used the pretrained model ResNet34 as teacher model. Residual network is a CNN model proposed by four researchers. It has a good effect in image classification and target recognition [7]. The advantage of the residual network is that it can alleviate the problem of gradient disappearance in a neural network. ResNet-34 is used as the primary network in this article.

The core concept behind the ResNet model is its use of residual learning, where the original function F(x) is reformulated as F(x) + x. This transformation simplifies the optimisation process compared to directly learning F(x). By leveraging residual vector representations within images, the model decomposes complex tasks into multi-scale residual subproblems, addressing the challenges associated with training deep neural networks. As a result, ResNet significantly enhances performance even as the network depth increases, outperforming traditional CNN architectures [7]. Based on these advantages, this study employs ResNet-34 as the backbone for the image classification model.

For the training process, a ResNet34 model was loaded from the PyTorch torchvision library with pretrained weights from ImageNet. To adapt the model to the specific classification task, the final fully connected layer was replaced with a custom head consisting of a dropout layer followed by a linear layer matching the number of target classes. To leverage the pretrained features while reducing training time and overfitting, all layers of the network were frozen except for the last fully connected layer, which was left trainable. This fine-tuning strategy allows the model to retain general visual features learned from ImageNet while adapting to the specific characteristics of the target dataset.

To train the model efficiently, the following hyperparameters were selected: MAX_EPOCH = 10, INIT_LR = 0.001, and BATCH_SIZE = 64. The number of epochs was set to 10 to balance training time with performance, as the model was built on top of pretrained weights and did not require extensive training. The initial learning rate of 0.001 is a commonly used starting point for fine-tuning with the Adam optimiser, offering a good trade-off between convergence speed and stability. A batch size of 64 was chosen to provide a stable estimate of the gradient while maintaining efficient GPU memory usage during training.

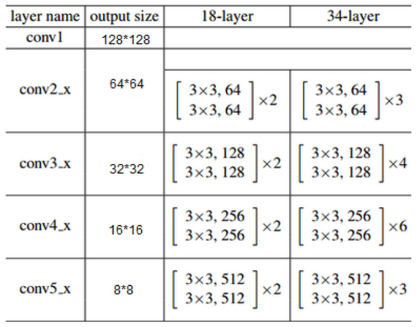
For optimisation, the Adam algorithm was employed due to its adaptive learning rate and momentum properties, which make it well-suited for training deep neural networks. Adam combines the advantages of both AdaGrad and RMSProp [8], adjusting learning rates based on first and second moments of the gradients, which helps accelerate convergence. The loss function used was CrossEntropyLoss, which is standard for multi-class classification problems. It measures the discrepancy between the predicted probability distribution and the true class labels, encouraging the model to assign high confidence to correct predictions. Together, Adam and CrossEntropyLoss provide a robust and effective framework for training image classification models.

C. *Student Model Training (ResNet18)*

I used the model ResNet18 as Student Model. ResNet18 and ResNet34 share the same foundational architecture built around residual blocks, but differ in depth and complexity. ResNet18 has 18 layers with learnable parameters, structured as 8 basic residual blocks, each containing two 3x3 convolutional layers. In contrast, ResNet34 deepens the architecture by using 16 residual blocks, where each block also contains two 3x3 convolutional layers. As a result, ResNet18 performs fewer operations per forward pass, while ResNet34 can capture more hierarchical features thanks to its additional layers. Both networks begin with a 7x7 convolution and max pooling layer, followed by four block groups, but ResNet34 includes more residual blocks per group: [3, 4, 6, 3] compared to ResNet18's [2, 2, 2, 2] (Figure 1).

This architectural increase leads to a significant difference in the number of parameters. ResNet18 has approximately 11.7 million parameters, whereas ResNet34 has around 21.8 million parameters [9]. The increased parameter count in ResNet34 allows for higher model capacity and better performance on complex datasets, but it also increases the computational cost and memory usage.

For the knowledge distillation training process, a ResNet18 model was initialised from the PyTorch torchvision library



layer name	output size	18-layer	34-layer
conv1	128*128		
conv2_x	64*64	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$
conv3_x	32*32	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$
conv4_x	16*16	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$
conv5_x	8*8	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$

Fig. 1. The structure of the Resnet-18 and Resnet-34 network

without pretrained weights. To adapt it for the specific classification task, the final fully connected layer was replaced with a dropout layer followed by a linear layer matching the number of target classes. Knowledge distillation was then applied to improve the performance of this lightweight student model by transferring knowledge from a more powerful pretrained teacher model ResNet34. During training, the student model was optimised not only to match the true labels using CrossEntropyLoss, but also to mimic the soft output probabilities (logits) of the teacher model using a distillation loss such as Kullback-Leibler (KL) divergence.

$$Loss\_total = \alpha \cdot T^2 \cdot \text{KL}\left(\text{softmax}\left(\frac{z_T}{\tau}\right) \parallel \text{softmax}\left(\frac{z_S}{\tau}\right)\right) + (1-\alpha) \cdot \text{CE}(y, \text{softmax}(z_S)) \tag{2}$$

For training the student model with knowledge distillation, the following hyperparameters were carefully selected to balance performance and training stability. The maximum number of epochs was set to 40 to provide sufficient training time for the student model, which starts without pretrained weights and requires more iterations to converge. The initial learning rate was set to 0.001, a commonly used value with the Adam optimiser, offering stable and effective convergence for deep networks. A batch size of 64 was chosen to ensure efficient GPU utilisation while maintaining a stable gradient estimate. The temperature parameter was set to 2, which softens the output probability distributions from both the teacher and student models, helping the student learn the inter-class relationships encoded in the teacher's outputs. A loss ratio of 0.25 was used to weight the distillation loss (KL divergence) relative to the standard cross-entropy loss. This ensures the student benefits from both the soft labels provided by the teacher and the hard ground-truth labels, with a balanced emphasis. Additionally, a learning rate scheduler was used with a step size of 10, reducing the learning rate every 10 epochs. This strategy helps the optimiser fine-tune the model in later stages of training, encouraging better convergence and reducing the risk of overshooting minima.

This dual-objective training helps the student model generalise better by learning the teacher's richer representation of class similarities, even without direct exposure to the same large-scale pretraining data. Knowledge distillation thus allows the ResNet-18 model to achieve improved accuracy with fewer parameters and lower computational cost, making it suitable for deployment in resource-constrained environments.

IV. Results

This section presents the outcomes of the image classification task, highlighting the performance of the student model trained with knowledge distillation. Key evaluation metrics, including accuracy and loss, are reported to assess the model's effectiveness. Visualisations with accuracy/loss curves are provided to illustrate the model's learning behavior and classification performance across different classes. The results of the student model (ResNet18) are compared directly with those of the teacher model (ResNet34) to demonstrate the impact of knowledge distillation.

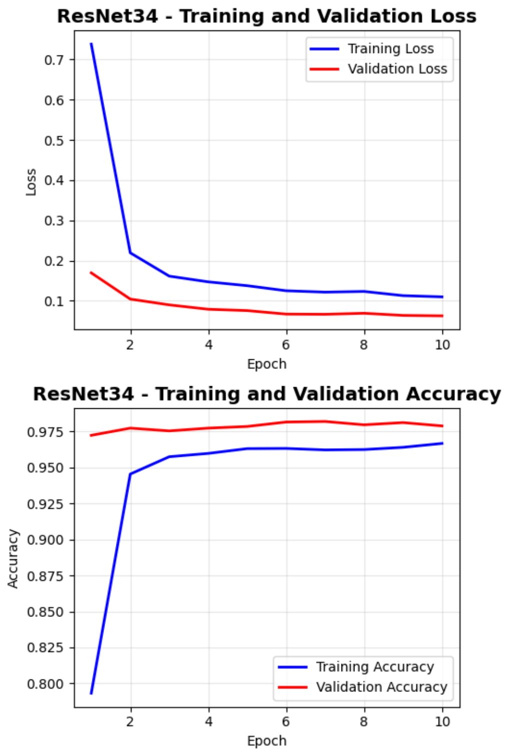


Fig. 2. Training/Validation loss and accuracy of Teacher Model (ResNet34) over 10 epochs

As shown in Figure 2, the training and validation curves of the ResNet34 model demonstrate strong and consistent performance over 10 epochs. As shown in the top plot, the training loss decreased significantly from approximately 0.73 to 0.11, while the validation loss also dropped steadily from 0.16 to around 0.06, indicating that the model is learning effectively without overfitting. The plot below shows a corresponding increase in accuracy: training accuracy improved from 80% to over 96%, while validation accuracy remained consistently high, reaching around 98%. The close alignment between the training and validation curves suggests good generalisation, and the minimal gap between them indicates that the model performs well on unseen data. Overall, these results highlight the robustness and effectiveness of ResNet34 as a teacher model for image classification.

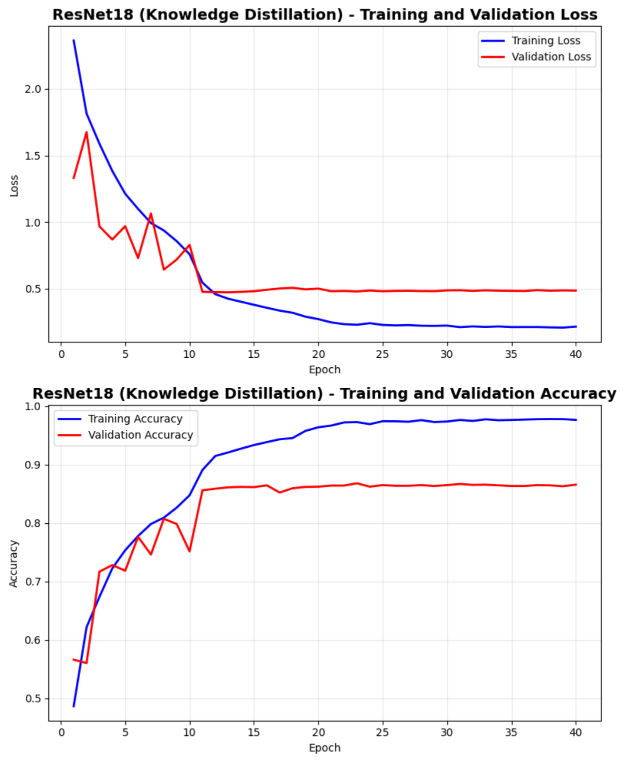


Fig. 3. Training/Validation loss and accuracy of Teacher Model (ResNet34) over 10 epochs

The training and validation curves for the ResNet18 (Figure 3) model trained with knowledge distillation show a clear improvement in performance over the course of 40 epochs. As depicted in the top plot, the training loss steadily decreased from over 2.3 to below 0.2, indicating effective learning. While the validation loss initially showed some fluctuations in the early epochs, it eventually stabilised around 0.48, suggesting improved generalisation as training progressed. The right plot shows the accuracy trends, where training accuracy increased sharply to approximately 97%, and validation accuracy reached around 86%. Despite a noticeable gap between training and validation accuracy, the student model achieved strong performance given its smaller size and lack of pretraining. These results demonstrate the effectiveness of knowledge distillation, allowing ResNet18 to learn meaningful representations from the larger teacher model (ResNet34) and achieve competitive accuracy with significantly fewer parameters.

V. Discussion

The aim of this project was to create an effective image classification model using knowledge distillation, with ResNet34 serving as the teacher model and ResNet18 as the student model. The findings show that the smaller student model, despite not being pretrained, performed well after distillation, achieving approximately 86% validation accuracy. While this is lower than the teacher model's 98%, it marks a notable improvement compared to starting from scratch. These results highlight the success of knowledge distillation in transferring generalisation capabilities and feature representations from a more complex model to a simpler one.

The findings have important implications for deploying deep learning models in real-world applications where computational resources are limited. The student model required fewer parameters and less memory, yet retained a high level of accuracy, making it well-suited for edge devices or mobile environments. This demonstrates that knowledge distillation can be a practical strategy for optimising deep neural networks without sacrificing too much performance, especially in scenarios where latency and resource usage are critical.

However, there were several limitations observed. The student model showed a noticeable gap between training and validation accuracy, suggesting some degree of overfitting or under-generalisation. Additionally, fluctuations in validation loss during early training indicate sensitivity to hyperparameters and temperature settings in the distillation process. Future improvements could include tuning the distillation temperature, experimenting with different loss weightings (α), or incorporating data augmentation techniques to further enhance generalisation.

This project provided valuable insights into the benefits of knowledge distillation. By implementing and analysing both the teacher and student models, a deeper understanding was gained of how soft targets can guide a lightweight network to learn meaningful patterns. The experience reinforced the trade-offs between model complexity and performance and highlighted how distillation can serve as a bridge between the two. Overall, this project enhanced both theoretical understanding and practical skills in model compression and training strategies in deep learning.

VI. Conclusion

In conclusion, this project effectively transferred knowledge from a bigger ResNet34 teacher model to a more compact ResNet18 student model, demonstrating the efficacy of knowledge distillation. The student model greatly reduced model complexity while achieving high performance by utilising temperature-scaled cross-entropy loss and soft labels. The findings demonstrate that knowledge distillation can be a useful strategy for model compression without significantly compromising accuracy. Future research might examine more sophisticated distillation strategies for combining attention transfer, intermediate feature matching, or self-distillation procedures. Its generalizability could further be confirmed by extending this methodology to different datasets or areas. All things considered, this work advances the expanding subject of model efficiency and demonstrates how knowledge distillation can be used to implement deep learning models in contexts with limited resources.

References

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

[2] Ba, J., & Caruana, R. (2014). Do deep nets really need to be deep?. Advances in neural information processing systems, 27.

[3] Bucilu, 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).

[4] https://github.com/FLHonker/Awesome-Knowledge-Distillation

[5] Chen, G., & Luo, Y. (2025). MCMKD: A Multi-Temperature Cosine-Weighted Model Ensemble for Logit-Based Knowledge Distillation. In 2025 2nd International Conference on Electronic Engineering and Information Systems (EEISS) (pp. 1-5).

[6] He, W., Pan, J., Zhang, J., Zhou, X., Liu, J., Huang, X., & Lin, Y. (2024). Multiloss Joint Gradient Control Knowledge Distillation for Image Classification. Electronics, 13(20), 4102. https://doi.org/10.3390/electronics13204102

[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).

[8] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[9] https://paddleclas.readthedocs.io/en/latest/models/ResNet_and_vd_en