Enhancing Model Efficiency through Knowledge Distillation: A Comparative Study of ResNet34 and ResNet18 for Image Classifications

Weiming Tang (Sherry)
Auckland University of Technology

***Abstract*—This report explores the application of knowledge distillation to improve the efficiency and generalization of deep learning models for image classification. A ResNet34 model is employed as the teacher, and a ResNet18 model is used as the student. The objective is to transfer the knowledge of the high-capacity teacher model to the smaller student model, thereby reducing computational cost while maintaining performance. The methodology includes pre-processing over 12,000 images, training both models, and evaluating their accuracy, precision, recall, and F1 score. The results show that while ResNet34 achieves higher training accuracy, it suffers from overfitting, whereas ResNet18 generalizes better with only a modest decrease in performance. The student model also significantly outperforms the teacher in terms of computational efficiency, making it suitable for real-time or resource-constrained deployments. The report concludes by recommending advanced distillation strategies and lighter architectures to further improve outcomes, highlighting the potential of knowledge distillation for sustainable AI model deployment.**

***Index Terms*—Knowledge Distillation, ResNet34, ResNet18, Image Classification, Deep Learning**

I. Introduction

THe increasing deployment of deep learning models in real-world applications has raised the need for models that are not only accurate, but also computationally efficient. While deep convolutional networks like ResNet34 deliver high performance, their large size and resource requirements make them impractical for use in environments with limited computing capacity, such as mobile devices or embedded systems. This challenge has led to growing interest in Knowledge Distillation (KD), a technique that enables smaller, lightweight models (students) to learn from larger, more complex models (teachers) without substantial loss in accuracy.

This project addresses the problem of developing an efficient student model that maintains high generalization performance while significantly reducing computational overhead. Specifically, it explores the use of knowledge distillation to train a ResNet18 student model from a ResNet34 teacher model on an image classification task. The study also aims to assess the effectiveness of KD in preserving model performance, identify overfitting issues in the teacher model, and evaluate the trade-offs between accuracy and computational efficiency.

The motivation behind this work can be divided into two fields: to enhance the adoption of deep learning models in resource-constrained environments and to explore techniques that make model training and inference faster and more energy-efficient. By doing so, the project contributes to the growing need for sustainable and scalable AI solutions.

The structure of this report is as follows: Section 2 presents a literature review of key knowledge distillation techniques and recent research relevant to the project. Section 3 outlines the methodology, detailing the data preprocessing steps and the training processes for both teacher and student models. Section 4 reports the experimental results and performance metrics used to evaluate model effectiveness. Section 5 discusses the implications of the findings, identifies limitations of the current approach, and offers recommendations for improvement. Finally, Section 6 concludes the report by summarizing the key insights and suggesting potential directions for future work.

II. Literature Review

Knowledge Distillation (KD) has become a critical strategy for compressing deep neural networks while maintaining high performance. This process involves training a smaller student model to mimic the behaviour of a larger, more complex teacher model. A particularly relevant contribution in this area is the work by Xu et al. [1], which introduces Feature Normalized Knowledge Distillation (FNKD), a method designed to combat label noise during image classification.

K. Xu et al. [1] argue that standard KD methods apply a fixed temperature parameter (T) to soften outputs, which may inadequately address the varying levels of noise present across samples. Their analysis reveals that label noise inflates the L2-norm of penultimate layer features, distorting learning. To address this, they propose using the L2-norm as a sample-specific correction factor rather than a global temperature thereby allowing the model to modulate its trust in teacher outputs based on individual sample difficulty.

K. Xu et al. [1] conduct a comprehensive analysis of label noise in knowledge distillation, highlighting how visual similarity between classes, such as those in datasets like CUB-200-2011 and ImageNet can lead to ambiguity and render one-hot labels unreliable. They theoretically and empirically demonstrate that the temperature parameter used in traditional KD is closely related to the L2-norm of feature representations, suggesting that a fixed temperature fails to adapt to the varying noise levels across samples. To address this, they propose Feature Normalized Knowledge Distillation (FNKD), which dynamically adjusts the influence of the teacher model based on each sample's feature norm. Specifically, samples with lower L2-norms, indicative of higher noise, are weighted more heavily in the soft target learning process, offering a more robust alternative to static temperature scaling. Experimental results across benchmarks like CIFAR-100, CUB-200-2011, and Stanford Cars show that FNKD consistently outperforms traditional KD approaches, particularly in fine-grained classification tasks where label ambiguity is common [1].

Despite its significant contributions, the work by K. Xu et al. [1] has certain limitations that remain unaddressed. The paper does not explore the applicability of FNKD in online knowledge distillation settings or within multi-teacher frameworks, which are increasingly relevant in dynamic or federated learning environments. Additionally, while FNKD introduces a sample-specific scaling mechanism to handle feature-level label noise, it does not consider integrating confidence-calibrated teacher outputs, which could further refine the reliability of the soft targets.

In relation to the current assignment, which involves a ResNet34–ResNet18 KD configuration using a fixed temperature, adopting FNKD's adaptive temperature strategy could enhance the student model's learning by adjusting the teacher's influence based on each sample's feature norm. This would be particularly effective in mitigating the overfitting observed in the teacher model and may explain the better generalization seen in the student despite its reduced capacity. Incorporating FNKD could potentially reduce manual tuning effort and increase robustness to label noise in the training data.

C. Xu et al. [2] introduce a teacher–student collaborative knowledge distillation framework to enhance image classification performance by leveraging the diversity of ensemble learning. Instead of using a single teacher, the framework generates multiple sub-networks from a large backbone model to form an ensemble, which collaboratively guides the student through a voting mechanism and dual-consistency training. This dual consistency comprises both feature-level and logit-level constraints. It ensures better alignment between teacher and student representations, enhancing both stability and convergence. The authors demonstrate the effectiveness of this approach across CIFAR-100, Tiny ImageNet, and ImageNet.

C. Xu et al. [2] also investigate the effectiveness of an ensemble strategy using a ResNet152–ResNet18 teacher–student framework on CIFAR-100. By analysing the classification errors across different sub-model exits (i.e., layers where predictions are made), the authors observed that some samples misclassified by the deepest exit were predicted correctly at shallower exits, 22%, 20%, and 14% of the time at the first, second, and third exits, respectively. This indicates that earlier layers may capture useful biased features even when the final classifier fails. Through integration of these multiple exits (ensemble learning), the network corrected many previously incorrect predictions. The study illustrates three integration cases that led to accurate final predictions, even when individual classifiers made mistakes. Experiments on CIFAR-100 and TinyImageNet confirm that using ensemble exits improves classification accuracy. Moreover, increasing the number of ensemble exits led to better performance, up to a certain point, validating the benefit of exit-level ensemble strategies.

The framework has a few limitations. It relies on training and maintaining multiple teacher networks, which increases computational cost and may not be suitable for deployment in resource-constrained environments. Moreover, the method does not investigate how such ensemble guidance might affect learning stability when scaled to very deep student architectures or online distillation settings.

This collaborative knowledge distillation framework is relevant to current assignment, where ResNet18 learns from ResNet34 using a standard one-teacher setup. In current assignment, the teacher model overfits, and the student achieves only moderate performance highlighting limitations in knowledge transfer. The approach by C. Xu et al. [2] offers an alternative by creating multiple sub-networks (or exits) from the teacher to form an ensemble, which guides the student using both feature-level and logit-level consistency. This could help address teacher's overfitting by leveraging diverse and less overfitted signals from earlier layers, which have shown to still predict correctly even when the final layer fails. Adopting a similar multi-exit or ensemble-based guidance might improve the learning stability and generalization of student model, especially for difficult samples or classes in image classification task.

Belinga et al. [3] conduct an in-depth empirical analysis of how different image classification datasets affect the effectiveness of knowledge distillation (KD). Their study explores the interaction between dataset characteristics such as size, complexity, and inter-class similarity and the performance of KD strategies. Using CNN architectures like ResNet and DenseNet, they perform experiments on MNIST, CIFAR-10, CIFAR-100, and Tiny ImageNet. The study investigates how different dataset complexities influence the effectiveness of knowledge distillation (KD) in image classification. Using a variety of datasets, the authors demonstrate that KD significantly boosts student model performance, particularly on complex and diverse data. Among the methods evaluated, Instance-level KD (IKD) outperforms Relational KD (RKD) by better capturing abstract and transferable representations. Importantly, the study finds that as dataset complexity increases, so do the performance gains of the student model, highlighting the importance of tailoring distillation methods to the nature of the data. In some cases, student models even outperform their teachers, showing KD's potential to improve generalization. These findings suggest that careful consideration of dataset characteristics is essential when designing effective KD pipelines, and recommend IKD as especially well-suited for complex classification tasks.

There are few limitations in their study. These include the restricted choice of architectures (ResNet50 and ResNet18), the focus solely on image classification tasks, and the limited selection of KD methods (RKD and IKD), all of which were chosen to maintain a controlled setup. They also note potential limitations in their evaluation metrics and a restricted literature review, which may have led to missing relevant findings.

Additionally, the study did not isolate variables in a controlled manner, as performance comparisons were made across different datasets without systematically varying individual factors. Lastly, the interpretation of results remains subjective, leaving room for alternative perspectives.

The current assignment dataset includes about 7,300 training images, 2,450 validation images, and 2,430 test images, with some images skipped during resizing to 224×224 pixels. This image size is standard for many deep learning models and helps ensure consistency in training. Based on the size and likely complexity of the current dataset, it is similar to datasets like CIFAR-100 or Tiny ImageNet, where knowledge distillation has shown strong benefits.

Chapariniya et al. [4] introduce an efficient method for recognizing human actions in still images by transferring knowledge from a larger teacher network (e.g., ResNet34) to a smaller student network (e.g., ResNet18). This approach eliminates the need for auxiliary data such as bounding boxes or body part annotations, making it more practical for real-world applications. The authors leverage soft labels generated using temperature-scaled softmax to guide the student model and utilize transfer learning from ImageNet to address limited data availability. Their experiments on the Stanford40 dataset show that knowledge distillation significantly improves the performance of the student model raising ResNet18's mean average precision (mAP) from 81.00% to 84.94%, nearly matching the teacher's 84.95%, while maintaining lower complexity.

This study has a few limitations. It does not explore deeper teacher ensembles, extensive hyperparameter tuning, or cross-dataset generalizability. It also lacks an ablation study on the impact of key components like the temperature value, loss weighting parameter (α), and squeeze-and-excitation blocks. Additionally, the framework is evaluated on only one dataset (Stanford40 Dataset), which limits its external validity. Real-time inference speed and deployment considerations on actual mobile or edge devices are also not addressed.

This article directly relates to the current assignment, as both focus on using knowledge distillation to transfer learning from a large teacher model (ResNet34) to a smaller student model (ResNet18). While the article evaluates performance using mean Average Precision (mAP), the current assignment uses accuracy, precision, recall and F1-score and as the performance metric. Despite this difference, the key idea remains the same, improving the efficiency and generalization of a lightweight model by guiding it with the outputs of a more complex one. The results in the current assignment show that ResNet18 achieved more balanced performance with better generalization and lower resource usage, which supports the effectiveness of knowledge distillation as demonstrated in the article.

Bao et al. [5] introduce a novel knowledge distillation method called Learning Temperature-Knowledge Distillation (LT-KD), which enhances student model training by dynamically adjusting the distillation temperature during the learning process. Unlike traditional KD methods that use a fixed temperature to soften teacher outputs, an approach that may not optimally suit all learning stages. LT-KD adopts a progressive strategy by starting with higher temperatures for easier knowledge transfer and gradually lowering them to increase learning difficulty. This approach mimics how human teachers guide students from simple to complex concepts. A key innovation in LT-KD is the use of a Gradient Reversal Layer (GRL) to update the temperature adaptively without introducing additional parameters. The method improves model performance on datasets such as CIFAR-100 and ImageNet without adding computational cost, and it can be integrated into various existing KD algorithms (e.g., KD, VID, DKD), where it consistently boosts Top-1 accuracy.

While the LT-KD technique introduces an innovative approach to dynamically adjusting distillation temperature, it has a few limitations. Firstly, its effectiveness heavily relies on the quality of the teacher model—if the teacher's knowledge is flawed, the student may inherit those inaccuracies. Secondly, as with most distillation methods, there is an inherent risk of knowledge loss due to compressing complex models into smaller ones. Additionally, although LT-KD automates temperature adjustment, it still requires careful tuning of hyperparameters such as the learning rate, difficulty weight (μ), and temperature bounds, which may vary across tasks and datasets. The method was only tested on two datasets CIFAR-100 and ImageNet. So, it is unclear how well it would work on other types of data, like medical images or noisy datasets. Also, the paper does not explore how the method would perform in real-time or online learning situations. It does not provide a clear explanation of whether the changing temperature approach is stable or guaranteed to work in all cases. Lastly, LT-KD adjusts the temperature for the whole dataset, not for individual samples, which might make it less effective when the data varies a lot or includes a lot of noise.

In current project, the student model (ResNet18) was trained using a fixed temperature across all samples, and the results showed signs of overfitting in the teacher (ResNet34) and a modest gap in performance between the two. Since the current dataset likely includes a mix of simple and complex images (similar to CIFAR-100), using the same temperature for every image may not be optimal. Some images might benefit from a higher temperature to smooth the teacher's predictions (especially noisy ones), while others might need a more confident prediction. Because LT-KD adjusts temperature globally not per sample, it may still face limitations in handling this kind of data variability.

H. Chen et al. [6] propose a new approach to knowledge distillation (KD) called Tailored Temperature for Student (TTSD). Traditional KD uses a fixed temperature to soften the teacher's output, which might not be ideal for all student architectures. TTSD addresses this by assigning a customized temperature value to each class, adapting it specifically to the student model's learning capacity. The idea is that some classes are more difficult for the student and therefore need a different level of softening in the teacher's logits.

H. Chen et al. [6] also introduce a new method called Tailored Temperature for Student in Knowledge Distillation (TTSD). It improves how student models learn by using two types of adjustable temperatures: one that is learned during early training (using a reverse gradient method), and another that changes based on how well the student is performing. This flexible approach can be added to existing distillation techniques to make them more effective. The method was tested on several datasets, including CIFAR-100, ImageNet, and Describable Textures, and it consistently improved model accuracy. Overall, TTSD makes the distillation process both simpler and more powerful, helping create smaller, more efficient models without losing performance.

While TTSD improves knowledge distillation by assigning different temperatures to each class, it still has a few limitations. It does not adapt temperature at the individual image (instance) level, which may reduce its effectiveness in datasets where samples within the same class vary greatly in difficulty. Additionally, learning a temperature for each class introduces extra parameters and increases computational complexity compared to using a fixed temperature. The method was only tested on standard academic datasets, so its performance in more complex or noisy real-world scenarios such as medical imaging or wildlife data remains uncertain. Furthermore, the technique has mainly been evaluated on lightweight student models, and its effectiveness on larger-scale student networks has not been explored.

In the current assignment, it uses a standard KD setup where ResNet34 is the teacher and ResNet18 is the student, with a fixed temperature applied to all outputs. However, the analysis shows that the student still has a performance gap and the teacher overfits, meaning the current transfer of knowledge could be suboptimal. TTSD provides a useful way to help the student model learn better by using different temperatures for each class. This means it can give extra support for classes the student finds harder to learn. In the current case, if ResNet18 struggles with certain behaviour types or image categories, TTSD can make the teacher's output more helpful for those specific classes, making it easier for the student to understand and learn from them.

Fu et al. [7] presents Interactive Knowledge Distillation (IAKD), a simple yet effective method that improves student learning by randomly swapping teacher blocks into the student model during training. This creates direct interaction between the two networks without needing extra loss functions. The authors also introduce three probability schedules to guide the swapping process. IAKD focuses on using the teacher's strong feature transformation abilities to guide the student, rather than just copying feature representations. Experiments show that IAKD improves student performance across various image classification tasks and works well alongside traditional KD methods.

While IAKD introduces a more dynamic and interactive learning process, it has some limitations. The method requires additional computation since both the teacher and student need to monitor and respond to each other's outputs during training. This may make it less efficient or harder to scale for large models or real-time applications. Also, the method has mainly been evaluated in standard academic settings, so its effectiveness on real-world, noisy, or imbalanced datasets remains uncertain. Lastly, IAKD assumes both teacher and student models can be updated interactively, which may not always be practical—especially when the teacher is a fixed pretrained model.

In the current assignment, the student model (ResNet18) learns from the teacher model (ResNet34) using a basic method, where it tries to copy the teacher's soft outputs using a distillation loss. However, the teacher tends to overfit, and the student only reaches average accuracy, which means the learning could be improved. IAKD offers a better way by letting the student model directly use parts of the teacher model during training. This way, the student not only learns what the teacher predicts but also how the teacher handles the features inside the network. Using an idea like IAKD could help ResNet18 learn better, especially when it struggles with deeper or more complex patterns.

W. C. Chen et al. [8] propose a knowledge distillation method that enhances student model learning by using feature maps (KDFM), the internal visual representations from convolutional layers instead of only copying the teacher's output predictions. To enable this, the authors introduce a feature map transformation module that aligns the size and structure of the teacher's and student's feature maps, allowing for smoother and more effective knowledge transfer. Tested on models like ResNet, VGG, and DenseNet using CIFAR-10 and CIFAR-100 datasets, this method improves classification accuracy and helps the student model better absorb the teacher's internal knowledge representations. KDFM enables lightweight convolutional neural networks with fewer layers but more trainable parameters to replicate the performance of advanced, complex models, achieving similar accuracy while delivering faster inference speeds.

While using feature maps enhances knowledge transfer, the method has a few limitations. Firstly, aligning and transforming feature maps between teacher and student requires additional computation and memory, which may reduce efficiency. Secondly, it assumes that feature map alignment alone is sufficient for effective learning, which may not always hold true, especially when the teacher and student architectures differ greatly. Lastly, the method was mainly tested on small to medium-scale datasets like CIFAR, so its effectiveness on larger or noisier real-world datasets is still uncertain.

It relates to the current assignment by offering an alternative way to improve the performance of student model, ResNet18. In current project, the student learns from a deeper teacher model (ResNet34) using traditional knowledge distillation based on soft outputs. However, the results show that the teacher overfits and the student only achieves moderate accuracy. By using a method like KDFM, the student model could learn not just from the teacher's final predictions but also from how the teacher processes information inside the network. This could help the student model perform better and run faster, even with fewer layers. It offers a more advanced and effective way to reduce the performance gap between the student and teacher models, which is useful for the issues you're facing in the current assignment.

Qin et al. [9] present a novel framework to enhance the performance of lightweight neural networks for medical image segmentation by transferring knowledge from larger, high-performing models. To address the challenges of deploying large models in real-world clinical settings due to their high computational demands, the authors introduce a distillation architecture comprising three core modules: Prediction Map Distillation (PMD), Importance Map Distillation (IMD), and a newly proposed Region Affinity Distillation (RAD). RAD is specifically designed to capture and transfer semantic region differences such as tissue boundaries between organs and tumors, which are often ambiguous in medical images. The framework was tested on two benchmark CT datasets (LiTS and KiTS19), showing significant improvements in segmentation accuracy up to a 32.6% increase in Dice score, while maintaining low computational cost. The lightweight student models trained with this method not only approached the performance of their teacher networks but in some cases even surpassed them, highlighting the method's effectiveness and potential for practical deployment in resource-constrained medical environments.

There are a few limitations when applying the proposed method to 3D medical image segmentation. First, handling 3D feature maps significantly increases computational cost and memory usage, making it difficult to deploy in practice. Second, transferring useful knowledge becomes more challenging because the target regions, such as tumors, are often very small compared to the large background areas in 3D images. Third, some medical image datasets are not suitable for 3D modelling due to limited depth information. For example, datasets with a small number of slices can result in the loss of meaningful spatial context along the z-axis.

The work offers a deeper and more structured KD approach, introducing modules like Prediction Map Distillation (PMD), Importance Map Distillation (IMD), and Region Affinity Distillation (RAD), which target both low-level and high-level knowledge transfer. While the current assignment focuses on classification with soft targets from logits, this paper demonstrates that deeper feature-level alignment and region-based contrast transfer can significantly improve a student model's performance especially in tasks with complex spatial structures, such as image segmentation.

Deep learning models like AlexNet, VGG, and ResNet work well for classifying breast cancer images in the BreakHis dataset, but they are too large and complex to run on devices with limited computing power. To solve this, Sepahyand & Abdali-Mohammadi [10] introduce a lightweight model using knowledge distillation. It uses two big teacher models (VGG and ResNeXt) to train smaller student models with fewer layers. The method transfers both final outputs and middle-layer features (called dark knowledge) from the teacher to the student. The student model based on ResNeXt achieved a high accuracy of 97.09%, while being much smaller with about 69 million fewer parameters, almost 1 GB less GPU memory use, and over 268 times more compression than the teacher model. The accuracy dropped only slightly (by 1.75%), showing that the student model still performed very well compared to other advanced methods. However, a key limitation is that if the teacher model exhibits variance in its predictions, this flawed knowledge may be transferred to the student, degrading its performance. To address this, the authors suggest using multiple teacher models with ensemble strategies to provide more reliable and consistent guidance during training.

Both the paper and the current assignment aim to reduce the size and complexity of deep learning models (like ResNet34) while maintaining strong classification performance through knowledge distillation (KD). The paper's approach of transferring both final output logits and intermediate feature maps (dark knowledge) from teacher to student closely aligns with the goals of ResNet34–ResNet18 setup in the current assignment, where the student learns from the softened outputs of the teacher. Additionally, both observe that the student model generalizes better, despite having fewer parameters and lower accuracy, which suggests that KD can mitigate overfitting in large teacher models an issue ResNet34 clearly faces. The limitation identified in the paper that variance in the teacher's predictions may degrade the student's performance is also highly relevant, as the project mentions overfitting in the teacher and its possible negative influence on the student.

Both the article and the current project show that the student model can generalize better, even though it has fewer parameters and slightly lower accuracy. This suggests that knowledge distillation (KD) can help reduce the overfitting problem found in large teacher models like ResNet34.In the current assignment, ResNet18 is faster and more efficient, making it better for real-time or low-resource settings which is consistent with the article.

A. *Compare and Contrast*

1) *Similarities*

TABLE I: Summary of LT-KD and TTSD Distillation Methods

Aspect	**LT-KD [5] & TTSD [6]**
Purpose	Adaptive temperature
Distillation Basis	Soft teacher outputs
Model Efficiency	Small student, good performance
Integration	Compatible with KD, DKD, VID
Test Results	Higher accuracy on CIFAR-100 and ImageNet
Dataset Limitation	Not validated on noisy/real-world data
No Sample-wise Temp.	Uniform temperature for all samples

TABLE II: Summary of KD Techniques from [4] and [9]

Aspect	**[4] & [9]**
Technique	Transfer knowledge from a larger teacher to a smaller student model.
Goal	High accuracy + low complexity
No Extra Labels	No bounding box / extra labels
Performance	Student nearly matches teacher performance
Practical Use	Real-world applications
Dataset Limit	Few datasets

TABLE III: Summary of FNKD and KDFM

Aspect	**FNKD[1] & KDFM[8]**
Goal	Better student model learning
Improves KD	Both improve over standard KD
Target	Image classification
Performance	Improved accuracy over baseline KD
Limitation in Scale	Tested mainly on small/medium datasets

2) Differences

TABLE IV: Differences Between LT-KD and TTSD

Aspect	**LT-KD [5]**	**TTSD [6]**
Temperature Granularity	Global temperature for all data	Class-specific temperature
Temperature Adaptation	Gradual (high to low) using Gradient Reversal Layer (GRL)	Two-stage: GRL + performance-based
Parameter Use	No extra parameters	Extra parameters for each class
Handles Variation	Poor with noisy or mixed data	Better for class-level differences
Complexity	Simple and easy to use	More complex and heavy

TABLE V: Comparison Between KD Methods in [4] and [9]

Aspect	**[4]**	**[9]**
Domain	Action recognition (still images)	Medical image segmentation
KD Method	Soft labels + temp-scaled softmax	PMD + IMD + RAD modules
Task Type	Classification (ResNet18/34)	Segmentation
Metric	mAP	Dice Score
Novelty	Transfer learning + simple setup	RAD module for region detail
3D Concern	Not applicable	High cost, z-axis issue

TABLE VI: Differences Between FNKD and KDFM

Aspect	**[1] FNKD**	**[8] KDFM**
Focus	Handles label noise	Enhances feature transfer
Key Mechanism	L2-norm-based sample-specific scaling	Feature map alignment and transformation
Noise Handling	Yes, adjusts for noisy labels	No, does not focus on noise
Computation Load	Low (no extra structure needed)	Higher (needs transformation modules)

III. Methodology

A. Data Preprocessing

The dataset went through four preprocessing steps to ensure compatibility with the deep learning models and to optimize performance during training and inference.

1) Image Loading: Images were loaded from the designated train, validation, and test directories. For the training and validation sets, class labels were automatically inferred based on the folder structure, where each subdirectory name (e.g., african_elephant, airliner) represented a distinct class.

2) Image Resizing: To standardize input dimensions across the dataset, all images were resized to 224×224 pixels. This resizing process involved a combination of scaling and cropping to preserve the aspect ratio while conforming to the expected input size of ResNet-based architectures. Resizing to 224×224 means it adjust all images, so they are the same size. This standardization is essential because neural networks process data in fixed-size batches.Scaling adjusts the image size proportionally so that the smallest side becomes 224 pixels. This avoids distorting the image (e.g., turning a square into a rectangle). Cropping then trims the image (usually the centre or a random part during training) to exactly 224×224 pixels. This step helps keep the image square and ensures the final size is correct.

3) Normalization: Image pixel values were first converted to 32-bit floating point and scaled to the range [0, 1]. Subsequently, each image channel was normalized using the mean and standard deviation computed from the training dataset. This step was essential to facilitate faster convergence and improve model performance.

4) Tensor Conversion: After normalization, the images were transformed into PyTorch tensors in NCHW format (batch size, channels, height, width). These tensors were then saved as .pt files to allow for efficient loading during the training and evaluation phases.

B. Model Architecture and Training Process

1) Teacher Model (ResNet34): The teacher model used in this project is ResNet34, a deep convolutional neural network developed by Microsoft, known for its residual connections that help prevent vanishing gradient issues in deeper architectures. The model was initialized with pretrained weights from the ImageNet dataset by setting weights=ResNet34_Weights.DEFAULT. This transfer learning approach allows the model to leverage rich, generalized features learned from over 1 million images across 1,000 categories.

Model Architecture: ResNet34 is a 34-layer deep convolutional neural network designed using the concept of residual learning. It begins with a 7×7 convolutional layer followed by max pooling to reduce spatial dimensions. The core of the architecture consists of four groups of residual blocks, which include 3, 4, 6, and 3 blocks respectively, using 3×3 convolutions and skip connections that allow the model to bypass certain layers. These residual connections help prevent vanishing gradient problems, enabling efficient training of deep networks. After the residual layers, the model applies global average pooling to flatten the features, followed by a fully connected layer that outputs class scores. In this project, the final layer of ResNet34 was modified to match the number of target classes. Pretrained weights from ImageNet were used to leverage previously learned visual features, making training more effective for the specific dataset.

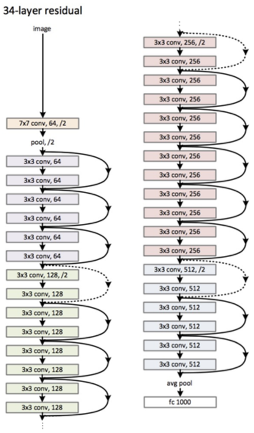

Fig. 1: ResNet34 Model Architecture

Training Procedure:

Loss Function: The model was trained using CrossEntropyLoss, which is standard for multi-class classification tasks. It measures the dissimilarity between the predicted class distribution and the true class label.

Optimizer: The Adam optimizer was used with a learning rate of 0.001. Adam is widely used due to its adaptive learning rate and momentum, making it well-suited for deep networks.

Epochs and Batch Size: The model was trained over 50 epochs with a batch size of 64. A batch size of 64 allows for efficient utilization of GPU memory while providing sufficient gradient estimates for stable optimization. Training for 50 epochs gives the model ample opportunity to learn complex patterns in the data without excessive overfitting. Due to the slow training performance in a CPU environment, the process was migrated to Google Colab, where a GPU-accelerated environment significantly improved training speed and efficiency.

Model Selection: To ensure optimal generalization performance, the model's accuracy was evaluated on the validation set after each training epoch. The model state corresponding to the highest validation accuracy was selected as the best-performing version. This version was then saved as resnet34_checkpoint.pkl for later use, such as evaluation on the test set or transfer to a student model during the knowledge distillation process.

2) Student Model (ResNet18): ResNet18 model was employed as the student model in the knowledge distillation framework. Unlike the teacher model, ResNet18 was initialized without pretrained weights to ensure it learned representations guided by both the ground truth labels and the softened outputs of the teacher model. Unlike one-hot ground truth labels, the soft labels produced by ResNet34 carry richer information, including inter-class relationships and confidence scores. To facilitate the distillation process, the outputs of both ResNet34 and ResNet18 were passed through a softmax function with a temperature parameter (T > 1), which softened the output distributions. These softened probabilities helped ResNet18 to better mimic the behaviour and decision boundaries of ResNet34, thereby improving its generalization despite having fewer parameters.

Model Architecture: ResNet18 is a convolutional neural network composed of 18 layers, designed with the principle of residual learning to ease the training of deep networks. It starts with a 7×7 convolutional layer and a max pooling layer to reduce the spatial size of the input image. The core of the model is built from four sequential stages of residual blocks, each containing two convolutional layers with 3×3 filters and identity shortcut connections that skip over the layers. These residual connections allow gradients to flow more easily through the network, improving convergence and reducing the risk of vanishing gradients. After these residual stages, the network applies global average pooling to condense spatial features into a fixed-length vector, which is then passed through a fully connected layer to produce the final classification output. In this project, the final fully connected layer was adjusted to match the specific number of classes in the dataset. Unlike the teacher model, ResNet18 was trained from scratch (without pretraining) using knowledge distillation, where it learned to mimic the behaviour of the pretrained ResNet34 teacher model.

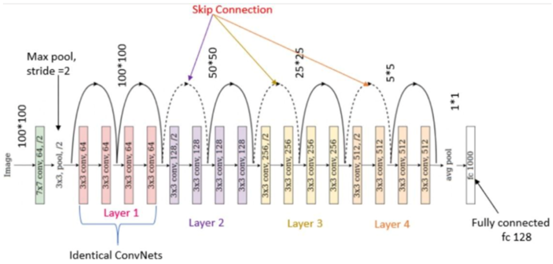

Fig. 2: ResNet18 Model Architecture

Training process: Resnet18 was trained using a combination of two loss functions: the standard CrossEntropyLoss and the Knowledge Distillation (KD) loss. CrossEntropyLoss measures the discrepancy between the predicted class probabilities and the true labels, ensuring the student learns directly from ground truth. The KD loss, implemented via Kullback-Leibler (KL) divergence, encourages the student to mimic the softened output probabilities of the pretrained teacher model (ResNet34). These ”soft labels” capture richer relationships between classes and reflect the learned knowledge of the teacher.

Optimizer: It is similar to the teacher model—the Adam optimizer was used with a learning rate of 0.001 and a weight decay of 1e-5 to help prevent overfitting by penalizing large weights. It sets up a learning rate scheduler that decays the learning rate over time: step_size=10 means the scheduler updates the learning rate every 10 epochs, and gamma=0.1 reduces the learning rate by 90% at each step. Using a learning rate scheduler like StepLR helps improve the training process in several ways. Initially, a higher learning rate allows the model to learn quickly and explore the loss landscape more broadly. As training progresses, the scheduler gradually reduces the learning rate, which helps the model fine-tune its parameters and make more precise adjustments. This reduction prevents the model from overshooting the optimal solution and helps it settle into a better minimum. When combined with the Adam optimizer and weight decay, the scheduler supports both fast early learning and stable later training, leading to smoother improvements in training and validation accuracy.

$$\mathcal{L}_{\text{total}} = \alpha \cdot T^2 \cdot \text{KL}\left(\text{Softmax}\left(\frac{z_s}{T}\right), \ \text{Softmax}\left(\frac{z_t}{T}\right)\right) \\ + (1-\alpha) \cdot \mathcal{L}_{\text{CE}}(z_s, y)$$

- $\mathcal{L}_{\text{total}}$: Total loss used to train the student model
- $\mathcal{L}_{\text{CE}}(z_s, y)$: Cross-entropy loss between student logits z_s and true labels y
- $\text{KL}(\cdot, \cdot)$: Kullback–Leibler divergence between softened student and teacher outputs
- z_s: Student logits
- z_t: Teacher logits (no gradient)
- T: Temperature for softening logits (e.g., 4.0)
- α: Weighting factor between KD and CE losses (e.g., 0.7)

Model Selection: After each epoch, the student model's performance was evaluated on the validation set. The model checkpoint with the highest validation accuracy was saved as resnet18_checkpoint.pkl. This best-performing student model, informed by both hard labels and the teacher's soft outputs, represents an efficient and compact version of the teacher, suitable for deployment in resource-constrained environments.

IV. Results

A. *Resnet34(teacher model)*

The Figure 3 below exhibits signs of overfitting for Resnet 34 model. Its training accuracy rapidly reaches near-perfect levels approximate 0.99 and remains consistently high throughout the training process. However, the validation accuracy fluctuates in the range of 0.58 to 0.62 without notable improvement after the initial few epochs, indicating poor generalization to unseen data. Correspondingly, the training loss steadily decreases toward zero, while the validation loss progressively increases and becomes quite high (above 2.5), further confirming the overfitting behaviour of the model.

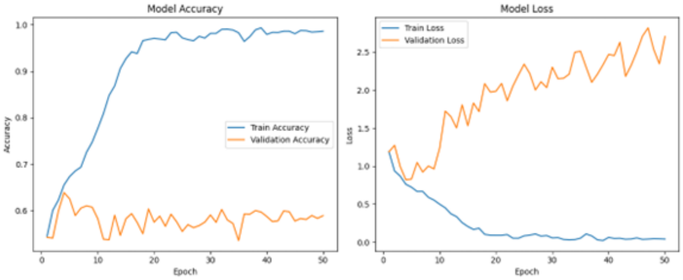

Fig. 3: ResNet34 Model accuracy and loss

B. *Resnet18(student model)*

The Figure 4 below shows Resnet 18 is a more balanced learning pattern compared to ResNet34. The training accuracy gradually improves over the epochs and stabilizes around 0.68, while the validation accuracy also increases smoothly and settles around 0.6, indicating better generalization. The training loss significantly drops and flattens out, and the validation loss decreases and remains stable around 1.0. This consistency between training and validation metrics suggests that ResNet18 is less prone to overfitting and achieves a more generalizable performance on unseen data.

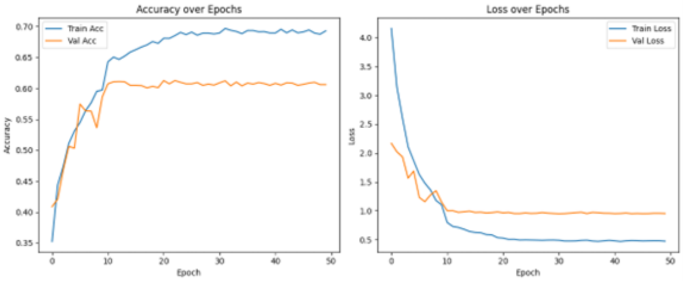

Fig. 4: ResNet18 Model accuracy and loss

The comparison between ResNet34 (Teacher) and ResNet18 (Student) highlights notable differences in performance and efficiency. ResNet34 achieves a very high training accuracy of approximately 0.99, but its validation accuracy fluctuates around 0.6, indicating overfitting. Its training loss drops to near zero while the validation loss rises to around 2.5, further confirming poor generalization. In contrast, ResNet18 achieves a moderate training accuracy of around 0.68 and a stable validation accuracy of about 0.6, with both training and validation losses remaining low and steady. This suggests better generalization. Additionally, ResNet18 is more efficient, being faster and more compact compared to the larger and slower ResNet34.

C. *Model performance evaluation*

As shown in table VII, ResNet34 slightly outperforms ResNet18 across all evaluation metrics, accuracy, precision, recall, and F1-score. The differences are relatively small, all under 3%, indicating that the student model performs reasonably well despite its reduced complexity. Specifically, ResNet34 achieves an accuracy of 0.6387 compared to 0.6126 for ResNet18, reflecting a modest trade-off in performance due to the smaller model size. In terms of precision and recall, the teacher model is marginally better at correctly identifying positive cases and minimizing false negatives. The F1-score, which balances precision and recall, also favours ResNet34, further confirming its slightly superior performance. Nonetheless, the close values demonstrate the effectiveness of knowledge distillation, as the student model retains much of the predictive capability while being more efficient.

Metric	**ResNet34** (Teacher)	**ResNet18** (Student)
Accuracy	0.6387	0.6126
Precision	0.6427	0.6131
Recall	0.6387	0.6126
F1-Score	0.6286	0.6070

TABLE VII: Model performance comparison

D. *computational efficiency comparison*

The table VIII below displays that ResNet34, with over 217,976,72 parameters and an estimated size of 83.15 MB, is significantly larger and more complex than ResNet18, which has around 116,895,12 parameters and a size of 44.59 MB. This added complexity in ResNet34 leads to a longer inference time (5.23 ms per image) and higher GPU memory usage during training (3194 MB), compared to ResNet18's faster inference time of 2.89 ms per image and lower training memory usage of 2574 MB. The training time for Resnet34 is 0.038s per batch, The training time for Resnet 18 is 0.019s per batch. Thus, ResNet18 trains nearly twice as fast per batch compared to ResNet34. Despite both models showing negligible GPU memory usage during inference (likely due to caching and small batch size), ResNet18 stands out for its speed and efficiency, making it more suitable for real-time applications or resource-constrained environments. In contrast, ResNet34 may be more appropriate when higher accuracy is required, and computational resources are not a limiting factor.

Metric	**ResNet34 (teacher)**	**ResNet18 (student)**
Total Parameters	217,976,72	116,895,12
Trainable Parameters	217,976,72	116,895,12
Estimated Size (MB)	83.15	44.59
Inference Time (ms/image)	5.23	2.89
GPU Memory Used Inference (MB)	0	0
Training Time per batch (s)	0.036	0.02
GPU Memory Used Training (MB)	3194	2574

TABLE VIII: Computational efficiency

V. Discussion

A. *Implication of findings*

The objective of this project was to explore the effectiveness of knowledge distillation by comparing a larger teacher model (ResNet34) with a smaller student model (ResNet18) in terms of generalization, performance, and computational efficiency. The experimental results showed that ResNet34 achieved near-perfect training accuracy (0.99) but suffered from overfitting, as its validation accuracy plateaued around 0.6 and validation loss steadily increased beyond 2.5. This suggests poor generalization despite high training performance. In contrast, ResNet18 exhibited a more balanced learning pattern, with both training and validation accuracy stabilizing around 0.6 and validation loss remaining low and consistent (1.0), indicating better generalization to unseen data. In terms of performance metrics, ResNet34 slightly outperformed ResNet18 in accuracy (0.6387 vs. 0.6126), precision (0.6427 vs. 0.6131), recall (0.6387 vs. 0.6126), and F1-score (0.6286 vs. 0.6070). While these differences are marginal (all under 0.03%), they demonstrate that the student model was able to retain much of the teacher's predictive capability, affirming the effectiveness of knowledge distillation. The results imply that knowledge distillation enables a compact model to mimic the performance of a more complex one, making it a valuable technique in resource-limited scenarios.

From a computational perspective, ResNet18 proved to be significantly more efficient. With fewer parameters (around 116,895,12 vs. 217,976,72), smaller model size (44.59 MB vs. 83.15 MB), faster inference time (2.89 ms vs. 5.23 ms), and nearly half the training time per batch (0.019s vs. 0.038s), it is well-suited for real-time applications or deployments on constrained devices. This trade-off between slight performance loss and substantial efficiency gains highlights a key advantage of using student models in production environments.

The findings from this project highlight a few important implications. First, the use of knowledge distillation effectively enabled the smaller ResNet18 model to retain much of the predictive performance of the larger ResNet34, despite having significantly fewer parameters and faster computation. This makes student models ideal for deployment in resource-constrained environments such as mobile devices or embedded systems, where model size and speed are critical. Additionally, the better generalization ability of ResNet18, compared to the overfitted ResNet34, suggests that smaller models may be more robust in real-world settings with noisy or limited data. Smaller models are also cheaper to train and use less energy, making them more sustainable. These findings support using knowledge distillation along with other methods like pruning or quantization to build smaller, efficient models. In short, with the right training, small models can still perform well and are easier to use in many real-world situations.

B. *Limitation and recommendation*

A few of limitations were identified during the project that may have impacted the overall performance and effectiveness of the knowledge distillation process. Firstly, the teacher model (ResNet34) exhibited clear signs of overfitting, as evidenced by its near-perfect training accuracy (0.99) and significantly higher validation loss (2.5). This overfitting suggests that the teacher learned patterns that were overly specific to the training data and did not generalize well to unseen examples. Since the student model (ResNet18) is trained to mimic the softened output of the teacher, any lack of generalization in the teacher can be partially inherited by the student. As a result, the quality of the distilled knowledge may have been compromised, limiting the student model's ability to learn robust and transferable features. Another limitation lies in the training configuration, which used a fixed learning rate (0.001), fixed batch size (64), and a predefined learning rate scheduler without tuning. These hyperparameters, while commonly used, may not have been optimal for the dataset or the specific architecture, potentially affecting convergence and final accuracy. Conducting a systematic hyperparameter search, such as grid search or random search, could help identify better combinations that enhance both training stability and final performance.

To improve the project and address its current limitations, a few enhancements can be considered. Firstly, the teacher model (ResNet34) exhibited clear signs of overfitting, which may have negatively impacted the quality of knowledge transferred to the student model. A more robust and better-generalized teacher model, such as one trained with dropout, data augmentation, or a deeper architecture like ResNet50 or EfficientNet—could potentially improve the student model's performance. Secondly, the training process relied on fixed hyperparameters, which may not have been optimal for the dataset or models used. Conducting hyperparameter tuning (e.g., grid search or random search for learning rate, batch size, temperature, and KD loss weight) could lead to better convergence and overall performance. Furthermore, although ResNet18 is a compact and efficient student model, experimenting with even more lightweight architectures such as MobileNetV2, EfficientNet-B0, or ShuffleNet could provide better trade-offs between accuracy and computational efficiency, making them more suitable for deployment in real-world, resource-constrained environments. Lastly, consider applying a method like KDFM (feature maps), which enables the student model to learn not just from the teacher's final outputs but also from its internal feature representations and processing steps. This deeper level of knowledge transfer can enhance the student model's performance and efficiency, even with a simpler architecture. By capturing richer information from the teacher, KDFM provides a more effective way to reduce the performance gap between teacher and student models, making it particularly useful for overcoming the limitations identified in this project.

VI. Conclusion

In conclusion, this project investigated the effectiveness of knowledge distillation by training a lightweight student model (ResNet18) to mimic the behaviour of a larger, more complex teacher model (ResNet34) in an image classification task. The key findings revealed that although ResNet34 achieved higher accuracy on training data, it suffered from overfitting, while ResNet18 demonstrated better generalization on unseen data. Despite having fewer parameters and faster computation, the student model retained much of the predictive power of the teacher, highlighting the practical benefits of knowledge distillation in balancing performance with efficiency. Future research could explore advanced distillation techniques such as Feature Normalized KD (FNKD), Learning Temperature KD (LT-KD), or Tailored Temperature for Student (TTSD), which dynamically adjust the learning process based on sample difficulty or class-level variation. Additionally, experimenting with more lightweight student architectures like MobileNet or EfficientNet and incorporating hyperparameter tuning may further improve generalization and training stability. Overall, this work contributes valuable insights into how smaller models can be effectively trained using knowledge distillation to achieve high accuracy with reduced computational cost. It underscores the relevance of KD for deploying deep learning models in resource-constrained environments and lays a strong foundation for continued exploration into more adaptive and scalable distillation frameworks.

References

[1] K. Xu, L. Rui, Y. Li, and L. Gu, "Feature normalized knowledge distillation for image classification," in *Lecture Notes in Computer Science*, 2020, pp. 664–680. doi:10.1007/978-3-030-58595-2_40.

[2] C. Xu, W. Gao, T. Li, N. Bai, G. Li, and Y. Zhang, "Teacher-student collaborative knowledge distillation for image classification," *Applied Intelligence*, vol. 53, no. 2, pp. 1997–2009, May 2022. doi:10.1007/s10489-022-03486-4.

[3] A. G. Belinga, C. S. T. Koumetio, M. E. Haziti, and M. E. Hassouni, "Knowledge distillation in image Classification: The impact of datasets," *Computers*, vol. 13, no. 8, p. 184, Jul. 2024. doi:10.3390/computers13080184.

[4] M. Chapariniya, S. S. Ashrafi and S. B. Shokouhi, "Knowledge Distillation Framework for Action Recognition in Still Images," in *Proc. ICCKE*, Mashhad, Iran, 2020, pp. 274–277. doi:10.1109/ICCKE50421.2020.9303716.

[5] Z. Bao, T. Tong, D. Du, and S. Wang, "Image Classification Network Compression Technique Based on Learning Temperature-Knowledge Distillation," in *Proc. ICNC-FSKD*, Guangzhou, China, 2024, pp. 1–7. doi:10.1109/ICNC-FSKD64080.2024.10702237.

[6] H. Chen, T. Zeng, K. Xiao, S. Wu, X. Liu and H. Su, "Tailored Temperature for Student in Knowledge Distillation," in *Proc. IEEE SWC*, Nadi, Fiji, 2024, pp. 760–766. doi:10.1109/SWC62898.2024.00132.

[7] S. Fu, Z. Li, Z. Liu, and X. Yang, "Interactive Knowledge Distillation for image classification," *Neurocomputing*, vol. 449, pp. 411–421, Apr. 2021. doi:10.1016/j.neucom.2021.04.026.

[8] W.-C. Chen, C.-C. Chang, and C.-R. Lee, "Knowledge Distillation with Feature Maps for Image Classification," in *Lecture Notes in Computer Science*, 2019, pp. 200–215. doi:10.1007/978-3-030-20893-6_13.

[9] D. Qin *et al.*, "Efficient Medical Image Segmentation Based on Knowledge Distillation," *IEEE Transactions on Medical Imaging*, vol. 40, no. 12, pp. 3820–3831, Dec. 2021. doi:10.1109/TMI.2021.3098703.

[10] M. Sepahyand and F. Abdali-Mohammadi, "Joint learning method with teacher–student knowledge distillation for on-device breast cancer image classification," *Computers in Biology and Medicine*, vol. 155, p. 106476, Dec. 2022. doi:10.1016/j.compbiomed.2022.106476.