CS419M : Course Project Final Report - Group 24
Learning “Dark Knowledge” from Teacher

Yash Salunkhe - 210020156
Anusha Dhakras - 210020019
Arnav Jain - 22b2527

1 Abstract

This course project focuses on applying machine learning techniques to the concept of distilled knowledge transfer from a teacher model to student model, inspired by seminal research in this domain. Leveraging advanced ML algorithms, we aim to develop models capable of automatically distilling complex information into its most salient and comprehensible form for effective educational dissemination. Drawing upon principles elucidated in the original paper, our project will explore various approaches such as natural language processing, neural networks, and knowledge representation to encode and distill knowledge. Through empirical evaluations and comparative analyses, we seek to assess the efficacy of different ML methodologies in facilitating distilled knowledge transfer across diverse educational domains.

2 Introduction

Distilled knowledge in neural networks involves transferring complex model insights from a resource-intensive ”teacher” to a more efficient ”student” model. Through guided learning and model compression, the essence of the teacher’s expertise is compacted into a smaller form, balancing computational efficiency with predictive accuracy.
Distillation of knowledge in neural networks was introduced by Geoffrey Hinton, Oriol Vinyals, and Jeff Dean in their seminal paper titled ”Distilling the Knowledge in a Neural Network” published in 2015. This pioneering work proposed a method for transferring knowledge from a large, complex ”teacher” model to a smaller, more efficient ”student” model. The technique involved training the student model to mimic not only the outputs but also the learned representations of the teacher model, thereby compressing its knowledge into a more compact form. This paper laid the foundation for subsequent research in knowledge distillation.

3 Methodology

The teacher model used in our approach was **ResNet18**. ResNet18, a variant of the Residual Neural Network architecture, comprises 18 layers with skip connections.

These skip connections alleviate vanishing gradient problems, enabling efficient training of deeper networks while maintaining model accuracy and interpretability. For our dataset, we used **CIFAR-10**. We trained all networks for 10 epochs at a learning rate = 0.001 with the Adam optimizer. For the student model, we made 2 smaller models - LightNN and SimpleNN. Here, we studied varying model complexity on the learning process.

LightNN

The LightNN architecture consists of two main components:

1. Features:
    - A convolutional layer with 3 input channels and 16 output channels, using a kernel size of 3 and padding of 1, followed by ReLU activation.
    - Max pooling layer with a kernel size of 2 and stride of 2.
    - Another convolutional layer with 16 input channels and 16 output channels, using a kernel size of 3 and padding of 1, followed by ReLU activation.
    - Another max pooling layer with a kernel size of 2 and stride of 2.
2. Classifier:
    - A fully connected layer with 1024 input features and 256 output features, followed by ReLU activation.
    - A dropout layer with a dropout probability of 0.1.
    - Another fully connected layer with 256 input features and 10 output neurons.

SimpleNN

The SimpleNN architecture also consists of two main components:

1. Features:
    - A convolutional layer with 3 input channels and 8 output channels, using a kernel size of 3 and padding of 1, followed by ReLU activation.
    - Max pooling layer with a kernel size of 2 and stride of 2.
    - Another convolutional layer with 8 input channels and 8 output channels, using a kernel size of 3 and padding of 1, followed by ReLU activation.
    - Another max pooling layer with a kernel size of 2 and stride of 2.
2. Classifier:
    - A fully connected layer with 512 input features (calculated from $8 \times 8 \times 8$) and 64 output features, followed by ReLU activation.

- A dropout layer with a dropout probability of 0.2.
- Another fully connected layer with 64 input features and 10 output neurons.

The LightNN model has **267738** trainable parameters while SimpleNN has **34290** trainable parameters Clearly, the LightNN model is more complex as it has around 8-fold more trainable parameters.

4 Results

The default parameters used were -

- Learning Rate = 0.001 (Adam optimizer)
- Temperature = 2
- Distillation loss weight = 0.25
- Cross entropy loss weight = 0.75

Model	Epochs	With Distillation (Y/N)	Final Training Loss	Test Accuracy on CIFAR-10
ResNet18	10	N	0.1078	80.30%
LightNN	10	N	0.5326	70.94%
LightNN	10	Y	0.8676	71.01%
SimpleNN	10	N	1.07159	62.46%
SimpleNN	10	Y	1.6413	62.50%

The test accuracy increases for the same model as we use knowledge distillation. The loss for knowledge distillation models is higher as it contains both the cross entropy loss and the distillation loss.

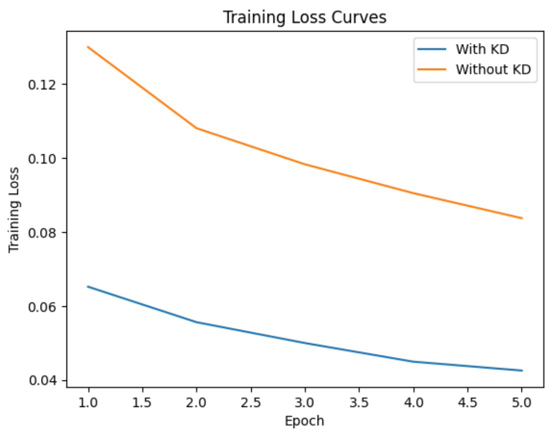

Figure 1: Plot showing the variation of training loss(CE) with epochs

The complex model LightNN performs better than SimpleNN in both cases, which is expected theoretically.

Effect of varying parameters

Temperature

In knowledge distilled models, the temperature parameter serves as a scaling factor during the softening process of the teacher's logits (the output of the model before applying the softmax function) before transferring knowledge to the student model.

When a higher temperature is used, it results in a softer probability distribution over the classes, meaning the model becomes less confident in its predictions. Conversely, a lower temperature leads to a sharper distribution, where the model is more confident in its predictions.

We used 3 temperature values for experiments - 2, 1.5 and 1.01 on LightNN.

Temperature	Final Training Loss	Test Accuracy
2	0.8676	71.01%
1.5	0.8093	70.65%
1.01	NaN	70.44%

The test accuracy for the undistilled model was 70.94%.
The test accuracy decreases with decreasing the temperature value. However this does not imply that one should increase it uncontrollably as it will degrade model's performance after a value.
We surprisingly get NaN loss in the case of T = 1.01. When the temperature parameter is set to 1 or close to 1 in knowledge distillation, it essentially maintains the original softmax distribution of the teacher's predictions without any scaling. This can lead to a problem known as "hard distillation," where the student model simply replicates the teacher's behavior without gaining any additional insights.
When using a temperature of 1 in knowledge distillation, the softmax function produces sharp probability distributions. If the logits (pre-softmax values) have large magnitudes, exponentiation in the softmax can lead to numerical instability, resulting in overflow or underflow and ultimately causing NaN (Not a Number) loss values.

Distillation and Cross-Entropy Loss weights

During training, the total loss is computed as -

$$\text{Loss} = \text{CE Weight} \cdot \text{CE Loss} + \text{Distillation Weight} \cdot \text{Distillation Loss}$$

where CE = Cross-Entropy, CE Weight + Distillation Weight = 1. We tested this on LightNN. The default split was CE Weight = 0.75 and Distillation Weight = 0.25.

The loss increases while reducing CE Weight(thereby decreasing Distillation Weight). Essentially, adjusting the distillation and CE weights in the loss function affects the trade-off between preserving the teacher's knowledge and optimizing the student's performance. Increasing the distillation weight prioritizes knowledge transfer, favoring

CE Weight	Distillation Weight	Final Training Loss	Test Accuracy
0.75	0.25	0.8676	71.01%
0.5	0.5	1.2057	70.67%
0.25	0.75	1.4483	71%

fidelity to the teacher's predictions. Conversely, increasing the CE weight emphasizes traditional classification accuracy, potentially sacrificing knowledge retention for improved performance on the target task.

5 References

- Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." arXiv preprint arXiv:1503.02531 (2015).
- He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

6 Acknowledgement

Our group would like to thank Prof. Abir De (Dept. of Computer Science & Engineering) for his constant guidance and this opportunity to work on this project as well as the TAs of this course who handled the logistics and cleared out doubts in a well-explained manner.