Learning “Dark Knowledge” from Teacher

Group 9
GitHub

IIT Bombay

May 6, 2024

Running The Code

Due to computational requirements we resorted to running the code on Google Colab. The project utilised Python 3.10 and PyTorch 1.12 as its major dependencies alongside allied librariries such as torchvision, numpy,matplotlib
Our work is also hosted at GitHub

Overview of the Topic

This discord between training and test objectives leads to the development of machine learning models that yield good accuracy on curated validation datasets but often fail to meet performance, latency, and throughput benchmarks at the time of inference on real-world test data.

Knowledge distillation helps overcome these challenges by capturing and “distilling” the knowledge in a complex machine learning model or an ensemble of models into a smaller single model that is much easier to deploy without significant loss in performance

Overview of the Topic

The hypothesis is that the student model will learn to mimic the predictions of the teacher model. This can be achieved by using a loss function, termed the distillation loss, that captures the difference between the logits of the student and the teacher model respectively. As this loss is minimized over training, the student model will become better at making the same predictions as the teacher.

$$\mathcal{L}_{\text{KD}} = \alpha * T^2 * D_{KL}(P||Q) + (1 - \alpha) * \mathcal{L}_{\text{CrossEntropy}}$$

We used $\alpha = 0.5$ and T $= 1$

Our Work

We train a network to judge how well a smaller network with lesser parameters could learn and mimic the original network and thereby save us resources. For this purpose, we have trained a CNN with and without the Knowledge Distillation Loss to know how much of a role does KD Loss play in training a newer network using "Dark Knowledge"

Our Work

We also tried experimenting with a smaller student network however going to smaller architectures (lesser number of parameters) wasn't feasible as training such a network become tough as accuracy over 10s of Epochs stayed close to 0.1 which is to be expected from random guessing as well

Dataset

We used the CIFAR-10 Dataset that contains 10 classes of images. The dataset was firstly rescaled to 224x224 to match the architecture used by ResNet and also normlaized using the mean and standard deviations of the ImageNet dataset

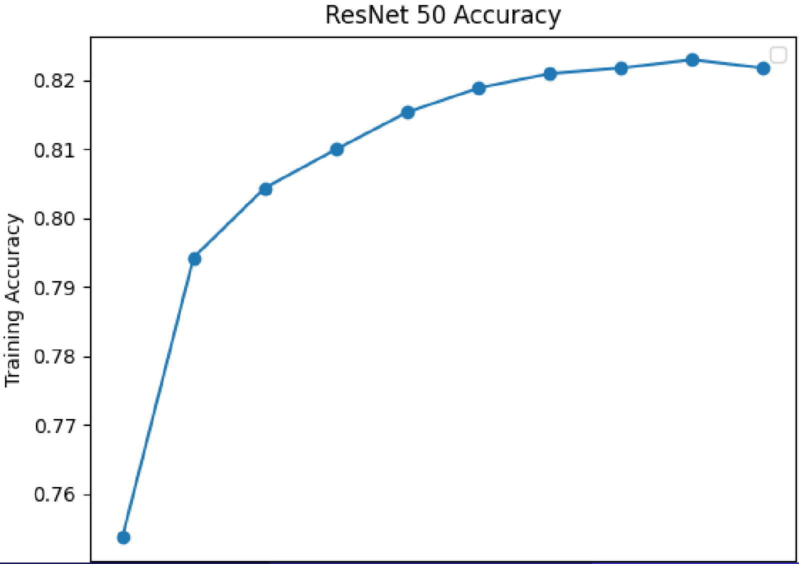
Teacher Network

The Teacher Network was a ResNet 50 Model trained for 10 Epochs with the Cross Entropy Loss

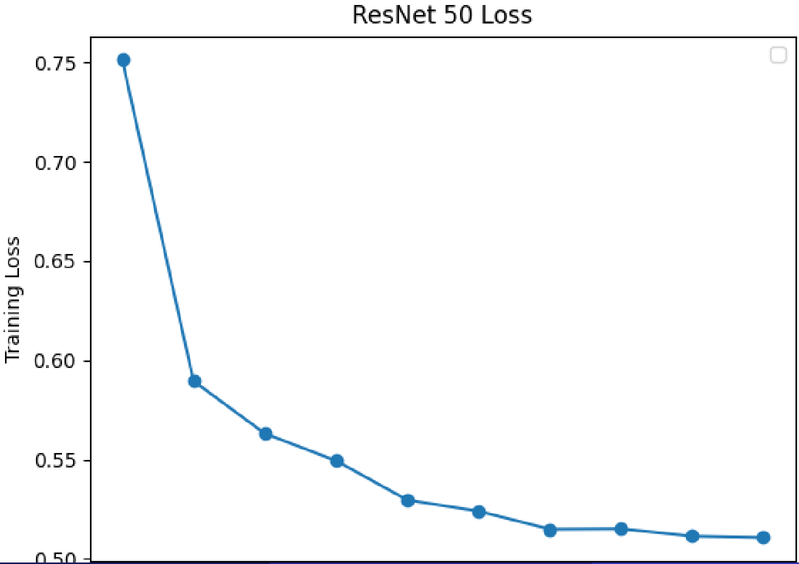
Epoch	Training Loss	Accuracy
1	0.7513	0.7537
2	0.5895	0.7943
3	0.5631	0.8044
4	0.5494	0.81
5	0.5296	0.8154
6	0.524	0.8189
7	0.5149	0.821
8	0.5151	0.8218
9	0.5114	0.823
10	0.5107	0.8218

Table: Teacher Model

Teacher Network



Teacher Network



Student Network

Student network was a custom CNN built using PyTorch. We trained two networks, one with the CrossEntropyLoss (referred as Without KD from hereon) and one with Knowledge Distillation Loss (referred as Without KD from hereon)

$$\mathcal{L}_{\text{KD}} = \alpha * T^2 * D_{KL}(P||Q) + (1 - \alpha) * \mathcal{L}_{\text{CrossEntropy}}$$

It was observed that in the initial epochs KD Loss helped the model learn a lot quickly over its counterpart by having $\approx 15\%$ higher accuracy.
However as the number of epochs increases both the models start to converge to similar results however the model with KD loss still holds an upperhand with 1% better accuray by the 10th Epoch
This could be attributed to the fact that ResNet 50 is not the most complex model available and was quite similar in architecture of the custom CNN used

Results and Inferences

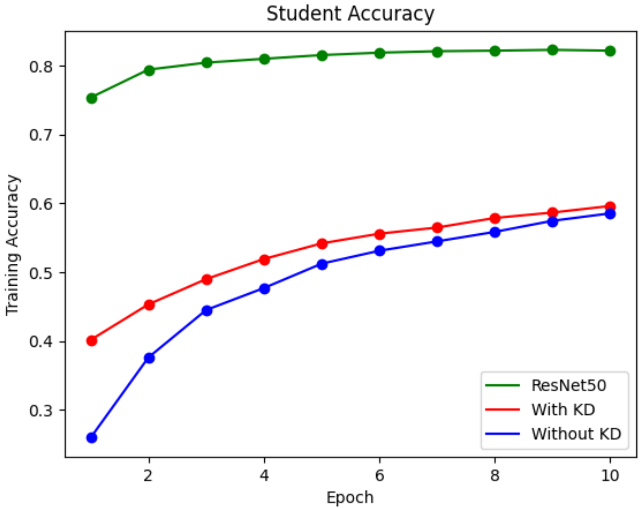


Figure: Model Accuracy

Results and Inferences

Epoch	With KD	Without KD
1	0.4017	0.2597
2	0.4533	0.3762
3	0.49	0.4451
4	0.5192	0.477
5	0.5419	0.5124
6	0.556	0.5312
7	0.5649	0.5448
8	0.5788	0.5586
9	0.5867	0.5747
10	0.5962	0.5855

Table: Student Accuracy

Results and Inferences

Both the models showed similar Training Loss curves

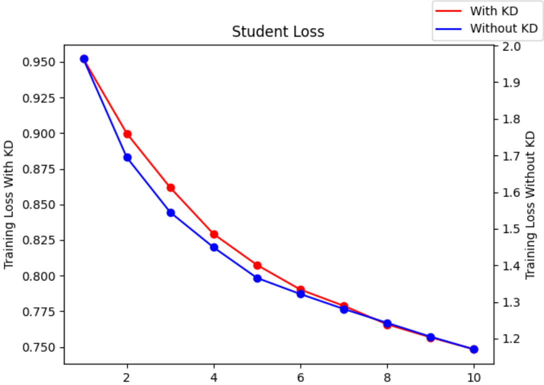


Figure: Student Model Loss

Other Attempts

Using a much smaller custom CNN, it was observed that using KD Loss did not make a difference as the CNN was too simple to generate usable predictions and hence in either case (with and without KD loss) it was giving accuracy close to 10% which is expected from random guessing as well