On Edge MRI Abnormality Detection using Teacher-Knowledge Distillation

Jinrui Zhang - Harshal Kulkarni - Nihal V P
New York University
Tandon School of Engineering
Brooklyn, NY
jz6578@nyu.edu, hsk8171@nyu.edu, nv2331@nyu.edu

Code GitHub Link: https://github.com/jinrui-rey/DL_FinalProject

Abstract

In this paper, we propose an on-edge compressed model that can be deployed on mobile devices. We train this student model by transferring dark knowledge from a deeper and complex pre-trained teacher model. The objective is to show that a knowledge-distilled student model (KD), weighing in at a mere 2.5 MB, can accurately detect the presence of a tumor and predict its type based on input brain MRI images, much like the teacher model, which is 130 MB in size. KD student model can be deployed on mobile devices and personalized using continuous training from user data providing personalized solutions. Since MRI analysis requires consultation with doctor in the real world, our suggested approach adds an extra layer of service while cutting down on needless work and expenses.

Introduction

Our objective entails the analysis of brain MRI images to discern the presence of tumors and classify their respective categories. This endeavor is motivated by the practical necessity encountered in real-world scenarios, where conventional procedures entail scheduling appointments with medical professionals for MRI analysis, incurring substantial time and financial costs.

However, the integration of large-scale deep neural networks (DNNs) onto resource-constrained devices presents formidable challenges due to their demanding computational and memory requisites. Considering this knowledge distillation has emerged as a promising paradigm for mitigating these obstacles, facilitating the compression of intricate models into more parsimonious forms conducive to deployment on low-power platforms without compromising performance.

Proposed Method

Knowledge distillation, introduced by Geoffrey Hinton [6], leverages the rich knowledge encapsulated within the teacher model, training the student model to emulate its behavior by learning from its soft target probabilities or feature representations. Through this process of knowledge distillation, the student model achieves performance levels commensurate with the teacher model while necessitating fewer computational resources, thus rendering it amenable to deploy resource-constrained devices.

Training with KD for students first requires training of a teacher model. This is done as usual using cross entropy loss on ground truth labels. From this teacher model, the pre-SoftMax logits z_i computed for each class is converted into a probability q_i by following equation with *temperature* (T >= 1).

$$q_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum \exp\left(\frac{z_j}{T}\right)}$$

With higher T, we get softer probability distribution over classes. The knowledge learned from training a teacher model on the training set is distilled and partially transferred to student network by minimizing following knowledge distillation (KD) loss [2] for training KD Student model.

$$L_{KD} = \alpha T^2 * CrossEntropy(Q_S^\tau, Q_T^\tau) + (1 - \alpha) * CrossEntropy(Q_S, y_{true}) \ldots (1)$$

Q_S^τ and Q_T^τ are softened "targets" of student and teacher using same temperature (T > 1) and α another hyperparameter tunes, the weighted average between the two components of losses.

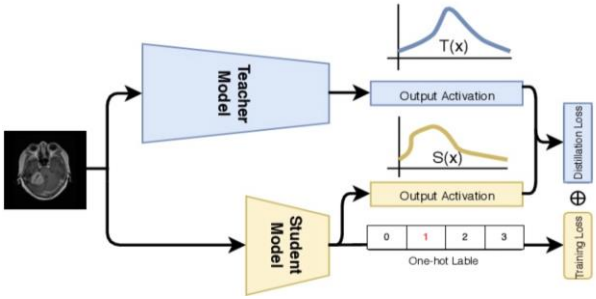


Fig. 1. The well-trained state-of-the-art teacher model (deeper and wider) displays in blue color and the distilled student model displays in cream-colored. Given an input

data (Brain MRI image), we train the student model via knowledge distillation by two losses. First is soft loss where we minimize the difference between student activation and teacher activation. This is also known as student mimic. The second is hard loss where we compare the prediction result with a one-hot label [5].

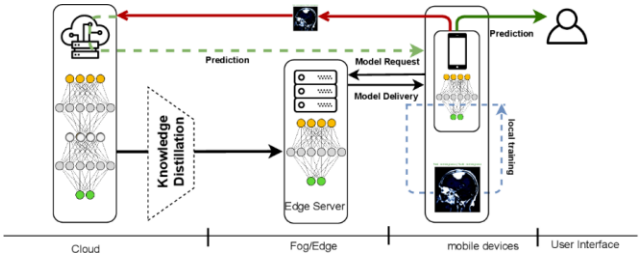
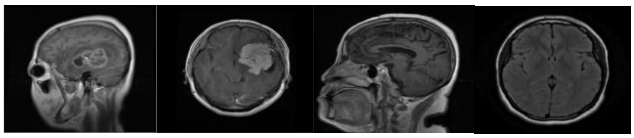


Fig. 2. Knowledge distillation-based machine learning on cloud. Pre-trained the-state-of-the-art deep networks deployed on the cloud server. The knowledge distillation is running on the cloud server. The distilled models are sent to the edge server, and they are ready to deliver to deploy on the edge devices. Users can request learning services by collecting the learning models from edge server. The local model can be privately trained with new data. If the user would like to share their data to improve the model, they could send the data to the cloud as well [5].

Experiment

The dataset [7], contains a total of 7023 brain MRI images, partitioned into 5712 samples for training and 1311 samples for testing. Glioma, meningioma, pituitary, and a class of MRI pictures devoid of malignancies are the four different classes of MRI images included in this dataset. Notably, the MRI pictures include both horizontal and sagittal perspectives. To enforce uniformity, data preprocessing was carried out due to the variance in input image sizes. Figure 3 shows some examples from the dataset.



Glioma *Meningioma* *Pituitary* *Non tumor*

Fig 3: One sample from every category of the dataset

Teacher Model

We experimented with various networks including Resnet18[8], VGG [1], Efficient Net [9], MobileNet [3] and tiny VGG. To benefit from knowledge transfer we used pre-trained models and adjusted them for 4 class classification purposes. Only the classifier layer was updated during backward pass, and convolution layers were frozen. The VGG model yielded the best test results (Fig. 4 illustrates the model architecture). With ~100k trainable parameters, only thirty epochs were required to achieve approximately 95% accuracy. Figure 5 shows plots for accuracy and loss during the training and testing stages.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 128, 128]           1,792
              ReLU-2         [-1, 64, 128, 128]               0
            Conv2d-3         [-1, 64, 128, 128]          36,928
              ReLU-4         [-1, 64, 128, 128]               0
         MaxPool2d-5           [-1, 64, 64, 64]               0
            Conv2d-6           [-1, 128, 64, 64]          73,856
              ReLU-7           [-1, 128, 64, 64]               0
            Conv2d-8           [-1, 128, 64, 64]         147,584
              ReLU-9           [-1, 128, 64, 64]               0
        MaxPool2d-10           [-1, 128, 32, 32]               0
           Conv2d-11           [-1, 256, 32, 32]         295,168
             ReLU-12           [-1, 256, 32, 32]               0
           Conv2d-13           [-1, 256, 32, 32]         590,080
             ReLU-14           [-1, 256, 32, 32]               0
           Conv2d-15           [-1, 256, 32, 32]         590,080
             ReLU-16           [-1, 256, 32, 32]               0
        MaxPool2d-17           [-1, 256, 16, 16]               0
           Conv2d-18           [-1, 512, 16, 16]       1,180,160
             ReLU-19           [-1, 512, 16, 16]               0
           Conv2d-20           [-1, 512, 16, 16]       2,359,808
             ReLU-21           [-1, 512, 16, 16]               0
           Conv2d-22           [-1, 512, 16, 16]       2,359,808
             ReLU-23           [-1, 512, 16, 16]               0
        MaxPool2d-24             [-1, 512, 8, 8]               0
           Conv2d-25             [-1, 512, 8, 8]       2,359,808
             ReLU-26             [-1, 512, 8, 8]               0
           Conv2d-27             [-1, 512, 8, 8]       2,359,808
             ReLU-28             [-1, 512, 8, 8]               0
           Conv2d-29             [-1, 512, 8, 8]       2,359,808
             ReLU-30             [-1, 512, 8, 8]               0
AdaptiveAvgPool2d-31             [-1, 512, 7, 7]               0
           Linear-32                   [-1, 4]         100,356
================================================================
Total params: 14,815,044
Trainable params: 100,356
Non-trainable params: 14,714,688
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 71.44
Params size (MB): 56.51
Estimated Total Size (MB): 128.14
----------------------------------------------------------------
```

Fig 4: Teacher Model (VGG) architecture and summary

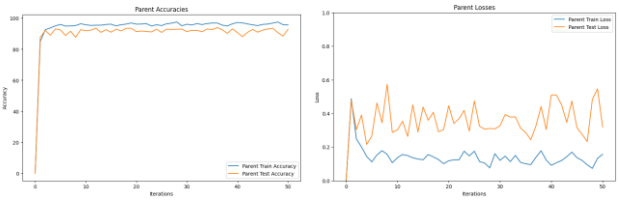


Fig 5: **Teacher Model** plots for accuracy and loss per epoch.

We get very low training loss and good accuracy from the very first epoch onwards since pre-trained VGG is employed for knowledge transfer.

Student Model

We defined a **custom student model** with 2 CNN blocks followed by a classifier layer. Figure 6 shows the complete student model architecture. The student model only includes 12626 parameters and a total size of 2.25 MB.

```
----------------------------------------------------------------
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 3, 128, 128]              84
              ReLU-2         [-1, 3, 128, 128]               0
            Conv2d-3         [-1, 3, 128, 128]              84
              ReLU-4         [-1, 3, 128, 128]               0
         MaxPool2d-5           [-1, 3, 64, 64]               0
            Conv2d-6           [-1, 3, 64, 64]              84
              ReLU-7           [-1, 3, 64, 64]               0
            Conv2d-8           [-1, 3, 64, 64]              84
              ReLU-9           [-1, 3, 64, 64]               0
        MaxPool2d-10           [-1, 3, 32, 32]               0
          Flatten-11                [-1, 3072]               0
          Linear-12                   [-1, 4]          12,292
================================================================
Total params: 12,628
Trainable params: 12,628
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 2.02
Params size (MB): 0.05
Estimated Total Size (MB): 2.25
----------------------------------------------------------------
```

Fig 6: **Student Model** architecture and summary

First, we set baseline by training Student model with normal cross entropy loss function. Then we train student model with KD loss function defined in equation 1 (optimized to use hyperparameter T = 1 and α = 0.5 taking equal contribution from KD loss and cross entropy loss). It uses a pre-trained teacher model and computes KD loss which is used to update the model parameters during backward pass. Figure 7 shows the Train loss and Train accuracies for baseline student model and KD student model.
As seen from Figures 7 and 8, the KD student model performs better compared to the baseline student model from epoch 1 due to the assistance from the teacher model as shown in equation 1.

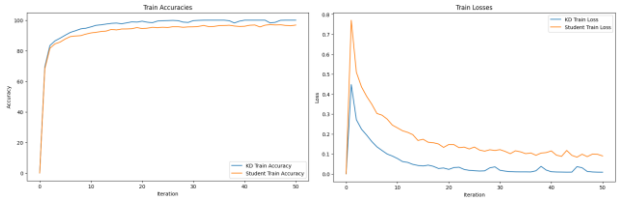


Fig 7: **Student Model and KD Student Model** Train accuracies and Train loss per epoch.

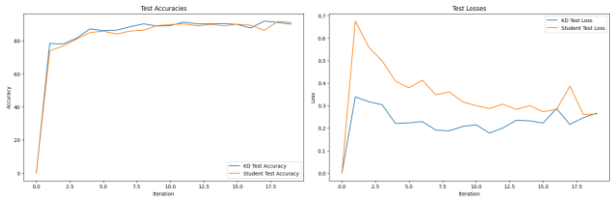


Fig 8: **Student Model and KD Student Model** Test accuracies and Test loss per epoch.

Results

Table 1 displays the Teacher model of size 128MB with approximately 100k trainable parameters achieving ~94% accuracy. Student Model with only 12.6k parameters and 2.25 MB size achieves 90.989% accuracy with normal training. Whereas with KD loss function, the same student model achieves 92.15% accuracy with lesser test loss value (Refer figure 7 and figure 8 loss curves). This implies that the KD student model not just improves its Train/Test accuracy but also predicts results with more confidence. In other words, when compared to Baseline Student Model, KD student model assigns higher probability to correct label showcasing better generalization ability.
There is a 98.25% model size reduction from the Teacher model to the student model. The 2.25 MB student model can be easily deployed on user-side mobile devices. Similar to the teacher model, this KD student model accurately identifies and categorizes tumors.

Model	Trainable params	Final Test Accuracy (%)	Estimated Model Size (MB)
Teacher VGG	100,356	95.67	128.14
Baseline Student	12,628	90.98	2.25
KD Student	12,628	92.15	2.25

Table 1. Table shows Model labels, corresponding trainable parameters count, final test accuracies and estimated model size.

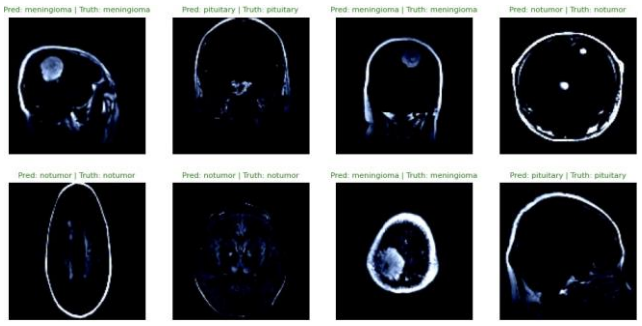


Fig 9. **KD Student model** prediction and corresponding truth label for a few images from the test dataset.

References

[1] Simonyan, K. & Zisserman, A. Very deep convolutional networks for largescale image recognition. In Proceedings of the 3rd International Conference on Learning Representations. (DBIP, San Diego, CA, 2014).

[2] https://github.com/haitongli/knowledge-distillation-pytorch; https://cs230.stanford.edu/files_winter_2018/projects/6940224.pdf

[3] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNet's: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[4] Chat GPT 3.5

[5] Z. Tao, Q. Xia, and Q. Li, "Neuron manifold distillation for edge deep learning," in Proc. IEEE/ACM 29th Int. Symp. Qual. Service, 2021, pp. 1–10.

[6] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. In Advances in Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning.

[7] https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset

[8] He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016.

[9] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (ICML), pages 6105–6114, 2019.

[10] https://claude.ai/chat