ECE1512 - Project B: Knowledge Distillation for Building Lightweight Deep Learning Models in Visual Classification Tasks

Assigned: Tuesday, March 1, 5:00 PM EST.

Due: Tuesday, April 5, 5:00 PM EST.

Introduction

Deep neural networks (DNNs) have been widely deployed on the cloud for a wide spectrum of applications from computer vision to natural language processing. The great success of deep learning is mainly due to its capability to encode large-scale data and to maneuver billions of model parameters. However, for real-time inference, it is a challenge to deploy these cumbersome deep models on edge devices with limited resources (e.g., mobile phones, IoT nodes, and embedded devices) not only because of the high computational complexity but also the large storage requirements. Hence, the simplicity of a DNN model becomes crucial for those scenarios.

Knowledge distillation (KD) [6, 13] gives the power to deploy lightweight models on such devices without sacrificing much on accuracy and performance. Knowledge distillation, formulated by Hinton *et al.* [6], is a promising methodology to "capture" and "distill" the knowledge in a complex machine learning model or an ensemble of models (known as *teacher model*) into a smaller single model (known as *student model*) that is much easier to deploy. Essentially, KD is a form of **model compression** that was first successfully demonstrated by Bucilua and collaborators in 2006 [1].

As shown in Figure 1, in knowledge distillation, a small "student" model learns to mimic a large "teacher" model and leverage the knowledge of the teacher to obtain similar or higher accuracy. The student model, trained with the combination of original training data and distilled knowledge from the teacher model, can provide a viable solution for resource-constrained hardware implementations of DNNs. They may contain richer knowledge than vanilla student models yet possess less complexity (in terms of either total number of parameters, complexity of hidden layers, network depth, or a combination of the above) than original teacher models. In the conventional KD [6], the knowledge is transferred from the teacher model to the student by minimizing a loss function, aimed at matching softened teacher logits as well as ground-truth labels.

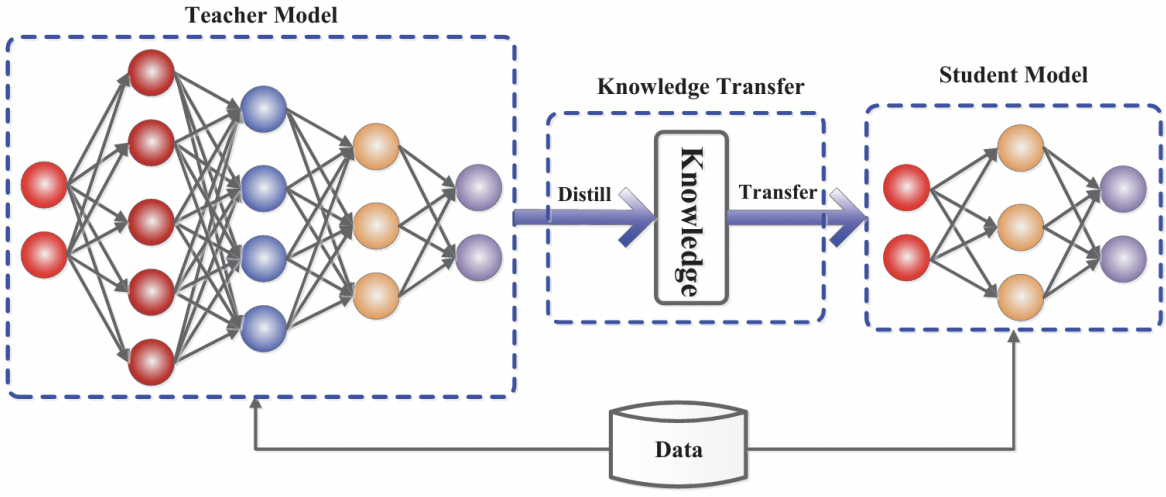

Figure 1: The generic teacher-student framework for knowledge distillation [4].

Objective

The goal of this project is to equip you with the tools and technologies required to transfer knowledge from a larger model to a smaller one that can be used practically in real-world settings. Specifically, the project will focus on the setting of Knowledge distillation as a model compression technique. The project is divided into 2 tasks, (1) using conventional knowledge distillation framework as a model compression method for a popular digit classification dataset, **"MNIST"** [8] and (2) using transfer learning and knowledge distillation to train a lightweight model for mimicking a pre-trained larger model in a clinical histopathology dataset, **"MHIST"** [17]. The detailed descriptions of the datasets and tasks are included in the next sections. To begin with this project, you must create a GitHub repository for Project B as follows:

1. Create a group (individual) project personal GitHub repository. Use a proper naming convention, *ECE1512_2022W_ProjectRepo_NameStudent(s).*
2. Create a Project B sub-directory.
3. Populate your project B sub-directories with project related material.

Datasets

The following is a detailed description of two datasets that you will need for Project B. Both datasets can be found in the ***Project_B_Supp.zip*** file, which has been uploaded to Quercus.

MNIST:

- **Paper:** Gradient-based learning applied to document recognition [8]: `http://yann.lecun.com/exdb/mnist/`
- **Description:** This dataset is a popular digit classification dataset which is a subset of a larger set available from NIST. The MNIST dataset is divided into 10 classes, each of which represents a digit between 0-9. The digits have been size-normalized and centered in a fixed-size image.
- **Availability:** Publicly available (for academic purposes).
- **Resources needed:** CPU
- **Data size:** 60000 train data + 10000 test data.

Minimalist HIStopathology (MHIST):

- **Paper:** A Petri Dish for Histopathology Image Analysis [17]: `https://arxiv.org/abs/2101.12355`
- **GitHub link:** `https://bmirds.github.io/MHIST/`
- **Description:** MHIST is minimalist in that it comprises a straightforward binary classification task of fixed-size colorectal polyp images, a common and clinically-significant task in gastrointestinal pathology. MHIST contains 3152 fixed-size images, each with a gold-standard label determined from the majority vote of seven board-certified gastrointestinal pathologists, that can be used to train a baseline model without additional data processing. MHIST is not an equally balanced dataset that contains different number of images per class: (a) 2162 images per class HP: hyperplastic polyp (benign), (b) 990 images per class SSA: sessile serrated adenoma (precancerous).
- **Availability:** Publicly available (for academic purposes).
- **Resources needed:** CPU
- **Data size:** 2175 train data + 977 test data.

Experimental Setup – What you need

1. Prerequisites:
   (a) Python 3.X
   (b) Sci-kit-learn (suggested)
   (c) NumPy (suggested)
   (d) Matplotlib (suggested)
2. Download ***Project_B_Supp.zip*** from Quercus.
3. ***mhist_dataset*** is a folder containing two subfolders, including the images and their corresponding annotations, for the MHIST dataset. All 3152 images are in ***images.zip*** file. Annotations are included in ***annotations.csv***. Note that this file includes each image file name and its corresponding majority-vote label and degree of annotator agreement expressed as the number of annotators who marked the image as SSA (e.g., 6 indicates 6/7 agreement with a ground truth of SSA and 2 would indicate 5/7 agreement with a ground truth of HP).
4. ***Task1.ipynb*** is a Python notebook showing how the teacher and student models are training in conventional KD framework. You will use this file to implement conventional KD for MNIST dataset.

GPU Requirements

Note that you **do not need a GPU to successfully complete Project B. You should be able to train your models on your local (personal) machines.** However, if you so wish, you can use a GPU by accessing Google Colab. Following are the steps to enable a GPU using Colab:

1. Upload the code base to Colab using your Google Drive.
2. Navigate to *Runtime* → *Change runtime* type in the top bar.
3. Change runtime accelerator to *GPU* and click *Save*.
4. Use *device = torch.device("cuda:0")* after importing libraries and call *.to(device)* function to transfer your model and tensors to GPU. Make sure that model and tensor both are placed on GPU.

Task 1: Knowledge Distillation in MNIST Dataset [25 Marks]

In this task, you will use conventional KD to transfer the knowledge of a CNN model to a smaller, fully connected network for the MNIST dataset, and then compare the performance of the student with and without KD. Please proceed as directed below.

1. Read the paper **"Distilling the Knowledge in a Neural Network"** [6] carefully, and then answer the following questions: [2.5 Marks]

   (a) What is the purpose of using KD in this paper? [0.5 Marks]

   (b) In the paper, what knowledge is transferred from the teacher model to the student model? [0.5 Marks]

   (c) What is the temperature hyperparameter T? Why do we use it when transferring knowledge from one model to another? What effect does the temperature hyperparameter have in KD? [0.5 Marks]

   (d) Explain in detail the loss functions on which the teacher and student model are trained in this paper. How does the task balance parameter affect student learning? [0.5 Marks]

   (e) Can we look at the KD as a regularization technique, here? Explain your rationale. [0.5 Marks]

2. Figure 2 (Figure 3, resp.) depicts the backbone CNN (fully connected, resp.) architecture for the teacher (the student, resp.) model on the MNIST dataset. Load the *Task1.ipynb* file and build both the teacher and student models in the "**Model creation**" section inside *Task1.ipynb* file. [2 Marks]

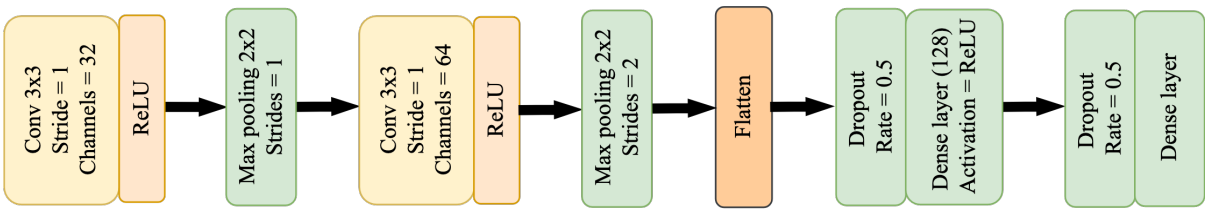

Figure 2: The proposed CNN teacher model architecture on MNIST

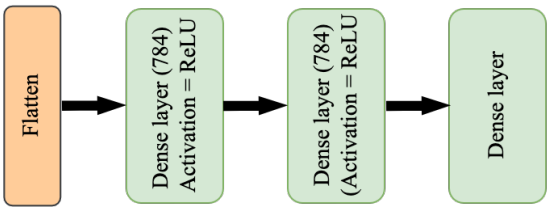

Figure 3: The proposed fully connected student model architecture on MNIST

3. Complete function *compute_teacher_loss* for loss (objective) function of the teacher model in the "**Teacher loss function**" section. Similarly, write your program in *distillation_loss* and *compute_student_loss* functions to define the loss (objective) function of the student model. [1.5 Marks]

4. To perform training and evaluation for a given model, complete the function *train_and_evaluate* in the section "**Train and evaluation**". The optimizer should be Adam, with a learning rate of 0.001 for all models. (Hint: The function *compute_num_correct* has already been implemented for you to compute the number of correctly classified images in a batch. Use this function to evaluate your model.) [1 Marks]

5. Train the teacher (the student, resp.) model for **12** epochs with *compute_teacher_loss* (*compute_student_loss*, resp.) in the pre-assigned "**Training models**" section of *Task1.ipynb* file. Upon completion of training, evaluate the performance of the trained teacher and the student model

on the test data, assuming that the most-confident class is predicted for each image. Report the test accuracy for both the teacher and student models. (Note that in this part, you need to tune the distillation hyperparameters, such as task balance α and the distillation temperature, T.) [1 Marks]

6. Plot a curve representing student test accuracy vs. temperature hyperparameters (T = 1,2,4,16,32, and 64) when the task balance parameter is 0.5. Explain the effect of temperature hyperparameter in this experiment. [1 Marks]

7. Compare the performance of the student which is trained with and without KD by reporting their test accuracy. To do this, build again the student model as in Figure 2 and write your program in *compute_plain_cross_entropy_loss* function to train a student from scratch without KD in "**Train student from scratch**" section (for the fair comparison, use the same hyperparameter as the student model with KD). [2 Marks]

8. Until this point, you had evaluated the teacher, the student with KD, and the student without KD to gain an understanding of the effect of KD on training the student model. Because the goal of using KD in this project is model compression, compare the number of parameters and floating-point operations per second (FLOPs) for the teacher and student models. Explain your findings. [1 Marks]

9. Use one of the XAI methodologies that you used in Project A in order to explain the behaviour of the teacher, student with KD, and student without KD models for a given input. [3 Marks]

10. Now that you have had a chance to implement, understand, and train the teacher and student models in the KD framework, you will use the state-of-the-arts KD algorithm to improve the performance of the student model. Read only one of the following papers- [10, 9, 13, 12, 3, 16, 7, 11, 2]. What is the main novel idea of the paper? [2 Marks]

11. How does the proposed method improve the student performance in comparison to the conventional KD? [2 Marks]

12. What are some limitations of the proposed method? How can they be addressed? [2 Marks]

13. Conduct an experiment to implement the novel component of the proposed method with the same models as in Figures 2 and 3. Report the student test accuracy, explain your approach, and its outcomes in detail. [4 Marks]

Task 2: Knowledge Distillation in MHIST Dataset [10 Marks]

In this task, we would like to use pre-trained models as the teacher and student networks. A pre-trained model is a saved network that was previously trained on a large dataset such as ImageNet [14], typically on a large-scale image-classification task. You either use the pretrained model as is or use **transfer learning** to customize this model to a given task. The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset. Load *mhist_dataset* from *Project_B_Supp.zip* file and then proceed as follows:

1. The teacher and student models are pre-trained **ResNet50V2** [5] and **MobileNetV2** [15] networks. Examine these structures and respond to the following questions: [3 Marks]

   (a) How can we adapt these models for the MHIST dataset using transfer learning? Talk about the **Feature Extraction** and **Fine-Tuning** processes during transfer learning. [0.5 Marks]

   (b) What is a residual block in ResNet architectures? [0.5 Marks]

   (c) What are the differences between the ResNetV1 and ResNetV2 architectures? [0.5 Marks]

   (d) What are the differences between the MobileNetV1 and MobileNetV2 architectures? [0.5 Marks]

   (e) How can ResNet architectures, regardless of model depth, overcome the vanishing gradient problem? [0.5 Marks]

(f) Is MobileNetV2 a lightweight model? Why? [0.5 Marks]

2. Repeat all the steps of Task 1 with the MHIST dataset, using the pre-trained ResNet50V2 and MobileNetV2 architectures (The experimental setup can be found in Table 1). Here are some hints that you can use them during programming: [5 Marks]

   - Decrease learning rate by 0.1 when you are fine-tuning the teacher and student models in transfer learning.
   - Use data augmentation before training your models.
   - All models should be trained using the **Adam** optimizer.
   - Because the MHIST dataset is not balanced, i.e., the number of images per class is not equal, test accuracy should not be used to evaluate model performance. (Think about the other metrics such as F1-score, AUC, etc. which are more suitable for this task).

Table 1: Experimental Setup for training the teacher and student models in Task 2.

Model	Architecture	Initial_epochs	Fine_tune_epochs	Batch_size	Learning_rate
Teacher	ResNet50V2	10	25	32	1e-4
Student + KD	MobileNetV2	10	25	32	1e-3
Student from scratch	MobileNetV2	10	25	32	1e-3

3. Explain the effect of transfer learning and knowledge distillation in the performance of the student model. Do pre-trained weights help the teacher and student models perform well on the MHIST dataset? Does knowledge transfer from the teacher to the student model increase the student's performance? [2 Marks]

Notes

1. This project should be completed in groups of one or two students.
2. Coding is expected for this project, and your code must be included in your submitted report – use the Python programming language.
3. External code may be used only if properly cited.
4. Include as many data visualization results as possible – a good visual is worth more than a thousand words (as per your ECE1512 Lecture 1 handout – "One picture is worth more than ten thousand words" (anonymous))

What to submit & Evaluation [5 Marks]

1. **Project evaluation will be based on a submitted written report.** You need to submit a written report (one per group) **in PDF format using the ECE1512 Q page submission utility.** Similar to Project A, your report should be approximately **20 pages in IEEE style** (either one column or two columns). List any additional references you have used in your work. You need to provide the code, data, figures and any other auxiliary material) used in your work. To that end, your written report should include a link to your Project B GitHub repository.
2. **Project evaluation will include an examination of the code, notebooks, figures, and other auxiliary material posted on your GitHub repo.** It is highly recommended that you include a README document on your project's GitHub page.
3. **Ouriginal "plagiarism" scores** will be visible upon (report) submission, so make sure that you are not penalized for plagiarism in uncited text and code portions (if applicable).
4. **Late submissions will not be accepted.**

Resources

Certain concepts and methods used in this project may be unfamiliar to you. Refer to these online resources for more details (cite if code is used):

- Basic Concepts of Knowledge Distillation
- A survey on Knowledge Distillation
- PyTorch documentation
- Measuring FLOPs for a given model
- Transfer learning
- Transfer learning and fine-tuning
- An Overview of ResNet and its Variants
- Why MobileNet and Its Variants (e.g., ShuffleNet) Are Fast
- Review: MobileNetV2
- Creation a GitHub repo
- Dark Knowledge in Neural Networks
- Distinguished Lecture for KD - Geoffrey Hinton

References

[1] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006. https://dl.acm.org/doi/10.1145/1150402.1150464.

[2] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4794–4802, 2019. http://openaccess.thecvf.com/content_ICCV_2019/papers/Cho_On_the_Efficacy_of_Knowledge_Distillation_ICCV_2019_paper.pdf.

[3] Xiang Deng and Zhongfei Zhang. Comprehensive knowledge distillation with causal intervention. *Advances in Neural Information Processing Systems*, 34, 2021. https://openreview.net/forum?id=ch9qlCdrHD7.

[4] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021. https://link.springer.com/article/10.1007/s11263-021-01453-z.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. https://arxiv.org/abs/1603.05027.

[6] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. https://arxiv.org/abs/1503.02531.

[7] Takumi Kobayashi. Extractive knowledge distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3511–3520, 2022. https://openaccess.thecvf.com/content/WACV2022/papers/Kobayashi_Extractive_Knowledge_Distillation_WACV_2022_paper.pdf.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf.

[9] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5191–5198, 2020. https://ojs.aaai.org/index.php/AAAI/article/view/5963/5819.

[10] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. Subclass distillation. *arXiv preprint arXiv:2002.03936*, 2020. https://arxiv.org/abs/2002.03936.

[11] Dae Young Park, Moon-Hyun Cha, Daesin Kim, Bohyung Han, et al. Learning student-friendly teacher networks for knowledge distillation. *Advances in Neural Information Processing Systems*, 34, 2021. https://proceedings.neurips.cc/paper/2021/file/6e7d2da6d3953058db75714ac400b584-Paper.pdf.

[12] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019. https://openaccess.thecvf.com/content_CVPR_2019/html/Park_Relational_Knowledge_Distillation_CVPR_2019_paper.html.

[13] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. https://arxiv.org/abs/1412.6550.

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. https://arxiv.org/abs/1409.0575.

[15] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf.

[16] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, 2019. https://openreview.net/pdf?id=SkgpBJrtvS.

[17] Jerry Wei, Arief Suriawinata, Bing Ren, Xiaoying Liu, Mikhail Lisovsky, Louis Vaickus, Charles Brown, Michael Baker, Naofumi Tomita, Lorenzo Torresani, et al. A petri dish for histopathology image analysis. In *International Conference on Artificial Intelligence in Medicine*, pages 11–24. Springer, 2021. https://arxiv.org/abs/2101.12355.