CS230

Exploring Effects of Knowledge Distillation in Model Compression and Accuracy

Matthew Tan
Department of Computer Science
Stanford University
mratan@stanford.edu

Evan Sheehan
Department of Computer Science
Stanford University
easheehan@stanford.edu

Dian Ang Yap
Department of Computer Science
Stanford University
dayap@stanford.edu

Abstract

The field of deep learning is expanding at an unprecedented rate, spurring the creation of neural networks deeper and more robust than ever before. Unfortunately, the rapid growth of the size of these models has rendered them unwieldy to construct and even more difficult to train. As such, we propose a model for the compression of said neural nets using a teacher-student, knowledge distillation architecture coupled with a comprehensive network pruning algorithm. Drawing on works by well-respected figures in the field, such as LeCun, Hinton, and Song Han, we utilized transfer learning on MobileNet, pre-trained on ImageNet, to fit it to the Kaggle Cat Dog Dataset (due to computational constraints), and pruned it utilizing zero density and random layer removal in order to construct a student network. We then used knowledge distillation to process the outputs of the teacher-student architecture given a lambda loss, back-propagating the results of this into the student network. Through this process, we were able to iteratively observe the benefits of knowledge distillation on a student network (compared to one not using knowledge distillation) under several sets of conditions, including when pruning a smaller number of layers or when seeking to minimize the number of trainable parameters in the student subject to accuracy constraints.

1 Introduction

Current state-of-the-art deep learning models for image recognition algorithms often utilize large and deep networks. Training such models often requires utilizing multiple GPUs over an extended periods of time. This computational size requirement, as well as processing time it entails, prevents these architectures from being run on smaller systems as well as serves as a prohibitive barrier to entry. The process of distilling information from large networks into smaller ones is an incredibly dynamic field. Recent research by Song Han (SqueezeNet)[7], Bengio (Fitnets)[4] and Hinton (Knowledge Distillation)[3] has explored the potential of creating much smaller networks without substantial losses in accuracy. In some cases, further post-processing of these networks can match / exceed the larger network's performance.

Our goal in this project was to delve into Knowledge Distillation and evaluate its feasibility, accuracy and effect on the smaller networks. We tested this model on MNIST[8][14] to check its feasibility and

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

then evaluated it on a larger dataset to prove its scalability. To do so, we retrofitted MobileNet,[13] a state-of-the-art image classifier that had been pre-trained on ImageNet,[11] to the Cat Dog Dataset[10] from Kaggle by using transfer learning. We then pruned this network using various methods which will be described in this paper to obtain a smaller, pruned, student network. Subsequently, we used Knowledge Distillation with these teacher-student architectures to train the student and engage in back-propagation. Doing so allowed us to both discover and explore the effects of Knowledge Distillation on various models and hyperparameter settings.

Explicitly, the input to our network is an image. We then use two neural networks, a full-size teacher network that has already been trained and a pruned version of the teacher (the student), to classify the image as either a cat or a dog. We then feed the outputs of these two networks into a loss function that evaluates both and then backpropagates through only the student network to train it.

2 Related work

Multiple approaches to compressing neural networks have been researched over the recent years. These can generally be split into 3 main buckets. The first involves some form of matrix factorization using SVD or by flattening convolutions. The second is through weight pruning by removing unimportant layers, something which was studied by LeCun[6] in his paper Optimal Brain Damage. Finally, the last method explores weight quantization, something which was successfully used by Han et al. to reduce AlexNet by a factor of 35 without substantial loss of accuracy.

Finding a means of effectively and efficiently training these models to retain the accuracy of their original models has become essential. Hinton proposed the Knowledge Distillation model to solve this problem which utilizes a weighted average between a L_{soft} and L_{hard} and which was shown to perform with good effectiveness in MNIST. In Teacher We Trust, Shen et al.[5] presents a more structured teacher student model and utilized it for pedestrian detection with some success.

Further research has been done for this general model to further improve its usefulness. Kim et al.[2] proposed using hints in which an additional intermediary layer provides information on its weights to the student network. Deep Mutual Learning (Zhang)[1] was further proposed which utilizes an ensemble of untrained networks to simultaneously learn and solve the task together. The research being done in this area presents a promising outlook for model distillation.

3 Dataset and Features

We used 2 main datasets for this project. First is MNIST, the handwritten digits database which has 60,000 images for the train set and 10,000 images for the test set. Pre-processing was handled by Keras libraries.

The second dataset we utilized was the Cat Dog dataset from Kaggle which contains 25000 high-resolution labeled images of dogs and cats. We pre-processed this by randomly splitting them into train and test sets with a (80-20) split. We then resized these images to the size we needed (224, 224,3), which is the standard input size of MobileNet trained on ImageNet. We used this full dataset to train the teacher model; however, due to computational constraints, we used only a subset of this to train the student models (1000 Train, 2500 Test).

We also attempted to retrofit the CIFAR dataset into MobileNet using transfer learning. However, despite multiple attempts, it was not successful (< 0.2) accuracy. We attribute this to the small image size of the images and the difficulty of scaling said images into the required size/res. for MobileNet.

4 Methods

Following the Knowledge Distillation model, we utilize the Teacher-Student architecture (see Image 1.1) to train the compressed networks. The teacher-student architecture essentially runs the input in parallel for the teacher and student models and merges their outputs into an additional loss layer (defined below):

$$\text{Loss} = (1-\lambda)(y_{\text{true}} - y_{\text{pred}}) + \lambda * (y_{\text{teacher}} - y_{\text{pred}})$$

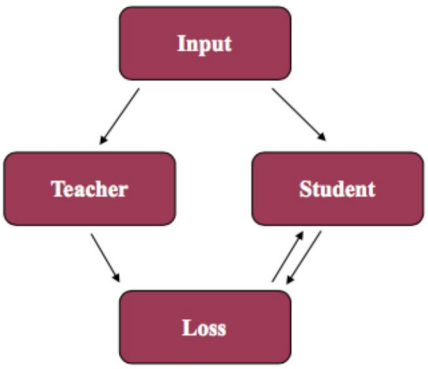

Figure 1: Teacher Student Architecture

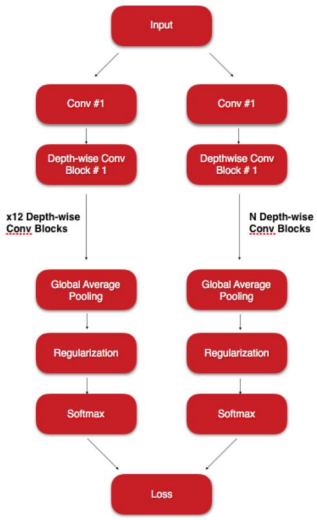

Figure 2: MobileNet Teacher Student Model

The results of this is then back-propagated to the student but not the teacher, effectively only training the student model. After some epochs, the student network is trained and can be used on its own.

Building the student network is done by compressing the teacher model via pruning. In this case, we utilized both zero density as well as random layer pruning. To do so, we summed over the absolute values of the weights in each convolutional layer in the teacher network and kept track of the number of weights in the layer which fell below a certain threshold. We then divided this number of weights in the layer below the threshold by the total number of weights in the layer to get the layer's zero density.

Multiple methods were explored over the duration of the project. Below we describe the chronological order by which we approached this research.

First, we used a simple, 2 layer dense network, each with 10 nodes, for the teacher model and a 1 layer dense network for the student model. After training with $\lambda = 0.4$, we discovered that the student network actually presented an increase in accuracy (from 92 to 93 percent).

We then tested the model on CIFAR by applying transfer learning to MobileNet. This was done by peeling off the last few layers, adding a softmax, and retraining the entire model. We received >90 percent accuracy on the train set but got less than 20 percent for the test set. Despite multiple experiments with different hyperparameters and adding various regularization/conv layers, we were largely unsuccessful with this. After performing a detailed causal analysis on this result, we concluded that the most likely cause was the disparity on the image sizes and the lack of features that CIFAR images have in comparison to ImageNet images.

Following this, we returned to testing this model on a more powerful MNIST model. We used a known Keras model that achieves >99 percent accuracy on the dataset and utilizes 2 CONV layers and 1 DENSE layer. We then removed one of the CONV layers and utilized the teacher-student architecture to retrain it. Following this, we searched through a range of loss lambda parameters to find the effectiveness of the teacher-student architecture. Finally, we attempted to improve this model by tuning batch size and maintaining this loss lambda.

After getting moderate results from the previous MNIST model, we then decided to run the teacher-student model on a much larger model. We considered and prototyped on InceptionV3 and ImageNet, however, we quickly realized the massive computational power necessary to run this model. To simplify, we then used MobileNet with similar images to ImageNet but with a smaller number of possible outputs. We used the Cat Dog Kaggle dataset because of the higher resolution images (similar to those in ImageNet), and because of its small class size. We used the full Cat Dog image database to train the teacher model, however, once again hindered by the lack of computational power, we utilized only a subset of this to train the student network.

Hyperparam	Brief Description
min_layers	Number of layers to remove using zero density
random	Number of random layers to remove
pipe_weights	Whether or not to initialize weights of student from teacher.
KD_batch_size	Batch size of the teacher-student architecture
KD_Num_epochs	Number of epochs for teacher student architecture
Loss Lambda	Lambda for loss function

Figure 3: Hyperparameters trained

Figure 4: Sample image of cat dog dataset

To evaluate MobileNet, we first retrofitted the last layers in the teacher (including softmax, etc.) to match the Cat Dog Dataset and then applied transfer learning in order to fit it. This allowed the teacher to achieve upwards of 95% accuracy on the 5000 examples in the test set. We then removed layers from it, using various methods to create the student, including zero density, random layers, etc. We believed that, given the Optimal Brain Damage paper, zero density constituted the most promising approach for pruning while still maintaining high accuracy. We also decided to explore random layer pruning as a control/comparison alternative to zero density. Additionally, we evaluated different ways of weight initialization for the student, including pre-loading the weights from teacher for the remaining layers that had not been pruned as well as maintaining the randomly-initialized weights. We then retrained the teacher-student architecture, keeping the teacher's weights frozen so that backpropagation was only carried out on the student. Various weightings of the teacher-student architecture were further explored, including varying the loss lambda parameter and tuning other hyperparameters (batch size, number of epochs, etc.).

5 Experiments/Results/Discussion

Throughout our testing, the metrics we followed most closely were student accuracy, number of layers removed, and number of parameters in the student. This was since our project centered around compressing a network, by both number of layers as well as number of parameters, while still maintaining the highest accuracy possible for the student. The hyperparameters we focused on tuning most intently were whether or not to pipe the teacher's weights into the student before training it, how many layers to randomly remove or remove via zero density, and the number of epochs and training set size for the student. This was because, given the sheer volume of diverse student networks we were required to train to explore all areas previously laid out, our computational resources were a severely limiting factor. Also, our literary framework directed us to optimize for convergence time (number of epochs trained) for the students as well.

In our initial tests conducted on the MNIST dataset, we observed that using different lambda loss parameters allowed for faster convergence and even higher accuracy than simply learning from the ground truth. Based on our data, we hypothesize that a lambda loss somewhere between .4 and .7 seems to be the optimum value for obtaining the highest accuracy marks with this architecture. Similarly, our testing seems to indicate that a batch size somewhere between 75 and 100 elements led to the best accuracy results. We hypothesize that having a small loss lambda allows for some form of regularization against images which the teacher network is unable to classify correctly, allowing the student network to learn from much easier dataset images. Also, an interesting extrapolation of our data seems to be that fully trusting the teacher network appears to negatively affect the accuracy of the model. This could be an encouraging sign, given that it implies that, after a certain point in training, the student is more than capable of "standing on its own legs" in an efficient and accurate manner.

To briefly mention our testing with the CIFAR dataset, we observed that we were unable to effectively undertake transfer learning to train the teacher model, likely, in our opinion, do to the fact that CIFAR images are much smaller, and also of a much lower resolution, than ImageNet photos, which MobileNet was initially pre-trained on.

Moving to our largest and most robust implementation of the concept, which utilized the MobileNet architecture and the Kaggle Cat Dog dataset, we observed that knowledge distillation appeared to work exceptionally well, compared to standard pruning, when the number of layers removed was decreased by only a small number amount. However, it seemed to have detrimental effects when the proportion of layers removed from the teacher increased to a significant fraction of the overall size of the network. We hypothesize that this may have been due to the fact that, after extensively pruning the teacher to obtain a student, the two networks may process images in such a different manner that it requires far more than 3 epochs of training to observe the lambda loss imparting a net benefit to the student, as the architectures gradually begin to work cooperatively. Similarly, since we initialized many of the student networks to possess the same weights as the teacher network (ie each respective layer possessed the same weights as its corresponding layer in the teacher network), we hypothesize this may have amplified the distance which the student network was required to traverse via backpropagation in order to reach the optimum weights. On an intriguing note, we observed that, when pruned via random layer removals, it appeared as though the student network utilizing knowledge distillation saw a much more appreciable increase in accuracy with the increase of the number of parameters in the network than its counterpart, or, conversely, it functioned at a much higher accuracy than the student not using knowledge distillation when both models possessed the same number of parameters. We are of the opinion that this finding may tacitly support the theory that knowledge distillation allows a student to be compressed to an even larger degree than one not using knowledge distillation, all the while permitting it to still attain better accuracy marks. Once again, we would caution against undue extrapolation in this case given the sample size of the testing data.

Tangentially, we observed that certain random layers that were removed seemed to possess more gravity in altering results when they were taken out than others, something which requires additional steps to rectify with the results we previously noted. Additionally, we saw that, in general, knowledge distillation seemed to cause the student using it to converge at a much faster rate than its counterpart (or it required fewer epochs to converge), but the student using knowledge distillation also experienced appreciable drops in accuracy at a much faster pace than the student not using it as the number of layers removed began to increase. We hypothesize that, given the number of epochs our computational constraints relegated us to, we cannot sufficiently draw any broader conclusions on this specific subtopic. Ideally, we would have run far more than 3 epochs per student with the full dataset to properly train the models, but GPU and cloud processing limitations hindered our ability to fully explore the space of possible avenues comprehensively. However, we feel our observational results span a diverse and dynamic range of the pruning, knowledge distillation, and model compression spaces and provide a robust undergirding for further investigation in said fields.

6 Conclusion/Future Work

Limited computational power has constrained the extensiveness of our results. However these results present a strong case for the possibility of compressing models without much loss of accuracy, and different methods of removing layers were seen to impact the accuracy of our student models. Running the model over the range of loss-lambdas for the Cat Dog Dataset may show us much more interesting details. Further, the Cat Dog Dataset, while consisting of large images, is relatively small in size. Testing the model on larger datasets such as ImageNet may yield much more useful / interesting results, though it will require far more computational power than we currently have access to. Finally, a technique discussed in the related works but not explored in this was the use of hints and the controlling of both their position and number. In some ways, pre-loading the weights is some form of hint, although a possible extension would be to run it on randomly initialized weights and see what the effect of this would be.

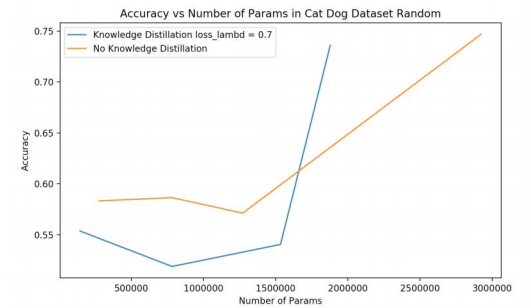

Figure 5: Accuracy vs Number of params for Random

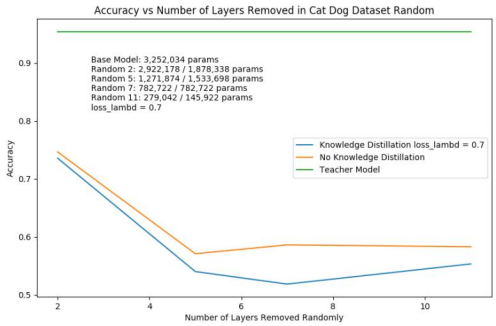

Figure 6: Accuracy vs Number of Layers Removed for Random

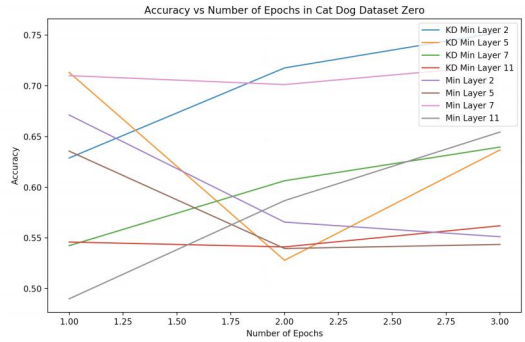

Figure 7: Accuracy vs Number of Epochs for Zero Density

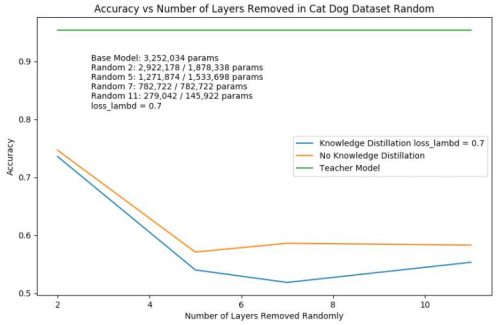

Figure 8: Accuracy vs Number of Layers Removed

7 Contributions

Evan Sheehan -

1. Individually designed and built both the teacher and student architectures for MobileNet in Keras
2. Wrote the code for transfer learning the teacher (MobileNet) network
3. Conducted the transfer learning on the teacher
4. Wrote code to load the Cat Dog Dataset and format it for Keras
5. Trained the teacher network/wrote the code to train it on the Cat Dog Dataset
6. Built the pruning algorithm that processed the requested layers for removal, including random layer pruning, zero density pruning, and specific layer removal
7. Wrote the code and designed the program that actually pruned the teacher model - builds brand new student models, block-by-block, in code via an automated process
8. Wrote the code for piping the teacher's weights into the student
9. Tuned hyperparameters for training the teacher/students
10. Collaborated with Matthew Tan to train the students locally/on AWS and tune hyperparameters
11. Collaborated with Matthew Tan to merge the Keras models/teacher/student with the knowledge distillation testing harness and file management systems

Matthew Tan -

1. Trained, built and tuned MNIST models for both proof of concept and final models.
2. Wrote testing harness scripts for quick evaluation
3. Wrote Knowledge Distillation harness for models.
4. Analyzed and plotted data.
5. Wrote scripts for proper logging including: saving weights, various print statements and reloading networks.

6. Tuned hyperparameters for the Cat Dog Dataset.
7. Attempted to apply transfer learning for the CIFAR dataset and tune hyperparameters.

Dian Ang Yap

1. Extracted models from literature review
2. Analyzed existing models and categorized such models based on strengths and weaknesses
3. Experimented with channel pruning by manually removing specific channels, which can be continued in future works
4. Adopted keras Surgeon model and experimented on dataset to rank channels or layers to prune
5. Tuned Keras Surgeon with Matthew Tan's weights, which can be further developed in further directions

References

All works are cited the first time they appear, per standard citation style

Special thanks to Guillaume and Yoann for there mentorship and guidance.

[1] Ying Zhang, Tao Xiang, Timothy M. Hospedales, Huchuan Lu, "Deep Mutual Learning.", Dalian University of Technology, Queen Mary University of London, University of Edinburgh, 2017.

[2] Jaeyoung Kim, Mostafa El-Khamy, Jungwon Lee, "BRIDGENETS: STUDENT-TEACHER TRANSFER LEARNING BASED ON RECURSIVE NEURAL NETWORKS AND ITS APPLICATION TO DISTANT SPEECH RECOGNITION.", Samsung Semiconductor, Inc. USA, 2018.

[3] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, "Distilling the Knowledge in a Neural Network.", Google Inc. 2015.

[4] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou , Antoine Chassang, Carlo Gatta and Yoshua Bengio, "FITNETS: HINTS FOR THIN DEEP NETS", Universitat de Barcelona, Universit'e de Montr'eal, Ecole Polytechnique de Montr'eal, Centre de Visi'o per Computador, ICLR 2015.

[5] Jonathan Shen, Noranart Vesdapunt, Vishnu N. Boddeti, and Kris M. Kitani, "In Teacher We Trust: Learning Compressed Models for Pedestrian Detection.", Carnegie Mellon University, Michigan State University, 2016.

[6] Yann Le Cun, John S. Denker, and Sara A. Solla, "Optimal Brain Damage.", AT and T Bell Laboratories, Holmdel, NJ, 07733.

[7] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashra , William J. Dally, Kurt Keutzer, "SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE.", DeepScale and UC Berkeley, Stanford University, ICLR 2017.

[8] MNIST Dataset: Yann Le Cun, http://yann.lecun.com/exdb/mnist/
[9] CIFAR-10 Dataset: Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, https://www.cs.toronto.edu/kriz/cifar.html
[10] Cat-Dog Dataset: https://www.kaggle.com/c/dogs-vs-cats
[11] ImageNet Dataset: http://www.image-net.org/

[12] Keras Codebase: https://keras.io/
[13] MobileNet Keras Implementation: https://keras.io/applications/#mobilenet
[14] MNIST Keras Implementation: https://keras.io/datasets/#mnist-database-of-handwritten-digits
[15] Github Repository: https://github.com/mratan1/CS230-Final-Project