DEEP NET TRIAGE: ASSESSING THE CRITICALITY OF NETWORK LAYERS BY STRUCTURAL COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep network compression seeks to reduce the number of parameters in the network while maintaining a certain level of performance. Deep network distillation seeks to train a smaller network that matches soft-max performance of a larger network. While both regimes have led to impressive performance for their respective goals, neither provide insight into the importance of a given layer in the original model, which is useful if we are to improve our understanding of these highly parameterized models. In this paper, we present the concept of deep net triage, which individually assesses small blocks of convolution layers to understand their collective contribution to the overall performance, which we call *criticality*. We call it triage because we assess this criticality by answering the question: what is the impact to the health of the overall network if we compress a block of layers into a single layer. We propose a suite of triage methods and compare them on problem spaces of varying complexity. We ultimately show that, across these problem spaces, deep net triage is able to indicate the of relative importance of different layers. Surprisingly, our local structural compression technique also leads to an improvement in overall accuracy when the final model is fine-tuned globally.

1 INTRODUCTION

As computational devices and methods become more powerful, deep learning models are able to grow ever deeper (Simonyan & Zisserman, 2014). To grow so deep, the most modern of networks have relied on clever intermediary layers—such as shortcut connection layers in He et al. (2015)—to overcome overfitting. While these methods allow for learning representations afforded only by very deep architectures, it is known that there are still more extraneous features and connections learned by these networks (LeCun et al., 1990). The question of how to best remove these redundancies has been the focus of deep compression methods (LeCun et al., 1990; Bucila et al., 2006; Han et al., 2015; Kim et al., 2015). Still others have investigated the ability of smaller—difficult to train—networks to learn from parent models via a method known as Knowledge Distillation (Hinton et al., 2015; Romero et al., 2014).

Both of these classes of approaches have demonstrated impressive performance for their respective goals; essentially, both lead to smaller networks that can match the performance of their larger, parent network. This performance is achieved in a variety of ways. For example, Han et al. (2015) reduces the number of network parameters via low-threshold pruning, followed by retraining, weight quantization and sharing, in tandem with low-rank approximations to ensure removal of redundant and unimportant weights. These methods can be thought of as finding a sparser, compressed model which best approximates the original.

Similarly, knowledge distillation methods leverage the soft-max outputs of previously trained "teacher" networks and network ensembles as guidance to train a smaller network, which would have otherwise been too difficult to train (Hinton et al., 2015; Romero et al., 2014; Saad & Solla, 1995). These knowledge distillation networks globally train the smaller network to best approximate the soft-max output of the original network, sometimes with per-layer, intermediary targets incorporated (Ba & Caurana, 2013; Lebedev & Lempitsky, 2015; Urban et al., 2016).

While impressive, these two methods do not shed any light on the *criticality*, or the relative importance of a given layer or block of layers to the overall output. Such layer-based analysis is important to understanding these increasingly deep networks, even if only in an empirical sense.

To that end, we propose an idea called *deep net triage* that independently assesses small blocks of layers with respect to their importance to the overall network health. We drive the triage by using the initial parent network as an initialization, like Bengio et al. (2007), rather than as a means of globally retraining. Triage works by removing a connected block of network layers and replacing them with a single layer; we focus on convolution layers in this paper. We iterate over all connected blocks of layers separately thereby assessing the role each set plays in the original *parent* network.

Our means for this triage is local structural compression, which approximates a section of a disassembled network and assesses the ability to approximate and relearn the original model. With structural compression we compress segments of a deep network—VGG16—and attempt to recover the compressed layer of the network via various initialization and training methodologies (Simonyan & Zisserman, 2014). We structure this as a compression problem as we are approximating multiple convolutional layers' representational abilities with a single, selectively initialized and trained layer. Distinctly, though, our primary goal is not to seek maximal compressive performance. Rather we seek to investigate the robustness of a network when faced with structural alterations, and how various learning techniques affect this behavior across data sets of assorted complexity. We seek overarching trends between these methods, layers, and data sets in hopes of developing a greater understanding for the representational ability, and robustness of neural networks.

We perform our analysis using five approaches to structural compression for deep net triage, and four different data sets. We find that of the five approaches, methods which fine-tune over the entirety of the network achieve best performance across all data sets. Furthermore, these fine-tuned models are able to match or even exceed the performance of the baseline model. This suggests that for superior performance, a network cannot be altered without again retraining over the entire network. We additionally demonstrate that the criticality of any single layer in the network is not sufficient to inhibit relearning of the representations of the parent network. Finally, we show that knowledge distillation is an effective means of transferring the learned representations from a teacher to a student at any given intermediate point, even when the layer is altered or compressed, and improves a model's convergence.

2 Methods of Deep Net Triage

Here, we first describe the concept of *deep net triage*. We then look at how structural compression is performed, and how it is used to compress a series of layers in a network down to a single layer. We then describe and contrast the methods we use to initialize and train a compressed network.

2.1 Deep Net Triage

The methods below each offer a perspective into the compressed block in question. They seek to probe into how the compressed layer is improving its representational ability, assess how well these representations are performing at the task at hand, and seek to warm start the process through various initialization or guided training steps. We refer to this as deep net triage to emphasize the process of determining which structural compressions and experiments matter most. As we cannot fully describe the optimization of these highly parameterized models, we must seek other ways of inferring facets from their performance.

2.2 Structural Compression

The VGG16 network is comprised of five blocks of convolutional layers, each of which are separated by max pooling layers. Within each block, the number of convolutional filters per layer is held constant. To perform structural compression, we take one such block and approximate the functions

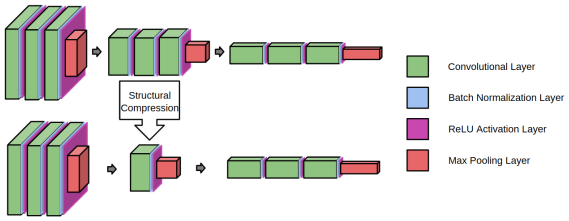

Figure 1: A pictorial representation of structural compression. A series of convolutional layers is approximated by a single layer.

learned by the two or three layers therein with a single layer, f_c. This is also depicted pictorially in Figure 1.[1]

$$f_c(x_i, W_{f_c}, b_{f_c}) \approx f_2(f_1(x_i, W_{f_1}, b_{f_1}), W_{f_2}, b_{f_2}) \tag{1}$$

This new layer, f_c, given input x_i, contains learned weight matrix W_{f_c} and bias vector b_{f_c}, is thus tasked with approximating the final representations learned by the two previously existent layers, f_1 and f_2. Where f_2 is fed the output of f_1, and has learned parameters W_2 and b_2. Likewise, x_i is the input to f_1 and f_1 is parameterized by learned weight matrix W_1 and bias vector b_1. Given this compressed layer, we then explore various strategies to promote learning and transfer of knowledge from the parent network to the compressed network.

2.3 Methods of Initialization and Training

We designed five different methods of initialization and training to promote learning in the compressed network and evaluate where and how representations were learned: performing parameter updates only in the compressed layer with randomly initialized weights as the compressed layer (CL-RW); retraining the entire compressed model with randomly initialized weights at the compressed layer (CM-RW); retraining only the compressed layer with weights initialized as a mean of the corresponding block's network weights (CL-MW); retraining the entire compressed model with compressed layer weights initialized as a mean of that block's parent network weights (CM-MW); and, creating a Student-Teacher network and training the compressed layer output tensors against the parent block's output tensors, before fine-tuning over the compressed layer weights (STN).

2.3.1 CL-RW

CL-RW considers the possibility that a newly compressed and inserted layer can fine-tune itself to adapt and merge into an already learned and frozen model. We do not presume that the features learned by the prior block of layers is the richest set of features that could be used by the later parts of the model.

After compressing a block of the trained, parent VGG16 model via structural compression, we then freeze all of the weights in the model outside of the compressed layer. The compressed layer is initialized with a zero-mean, Glorot Uniform distribution to allow for a new set of richer features to be learned (Glorot & Bengio, 2010). For successful integration, these features need to be successfully learned to fit and feed into the previously trained features of the surrounding layers.

2.3.2 CM-RW

CM-RW follows the intuition that while more meaningful features can be learned than those originally learned in the full, parent model, training across the whole network is necessary to successfully integrate the compressed layer's features into the overall compressed model.

[1] Note that this functional representation does not explicitly show Batch Normalization's tunable parameters.

After compressing a block of the trained, parent VGG16 model via structural compression, the compressed model is trained in its entirety. Again, the compressed layer is initialized as a zero-mean, Glorot Uniform distribution.

2.3.3 CL-MW

We suspect that there is value in the representations learned by the parent model, CL-MW initializes the structurally compressed layer with the help of these trained weights. As shown in Eq. (1), the compressed layer strives to learn some approximation of the three layers it has replaced. We therefore take an average over the N previously learned feature filters, f_i, across the entire block in the parent model, and load this averaged filter tensor, f_{avg}, into each of the filters in the new, compressed layer. We do this simple averaging for the filter tensors, bias vectors, and Batch Normalization coefficients (Ioffe & Szegedy, 2015).

$$f_{avg} = \frac{1}{N}\sum_{i=1}^{N} f_i \quad where \quad f_i \in \mathbb{R}^{3x3x1} \tag{2}$$

We then freeze the surrounding, previously learned weights, and optimize the compressed layer.

2.3.4 CM-MW

As described in Section 2.3.4 we load an average filter from the parent's convolutional block into the filters of the structurally compressed layer. We then train over the model as done in Section 2.3.2

2.3.5 STN

Similar to using intermediate hints, as done in Romero et al. (2014), or Student-Teacher networks as introduced in Saad & Solla (1995), we strive to mimic the output of the parent network at the end of a compressed layer. To do so, we construct a Student-Teacher network with a loss evaluated after the max pool operation at the end of the compressed layer. The loss is applied after the max pooling function to ensure dimensional equivalence between the Student and Teacher networks. We seek to minimize the difference between the tensor output of the structurally compressed layer and the tensor output of the respective convolutional block in the parent network. We evaluate an L2 Loss across the vectorized difference tensor between the teacher and the student output, averaged over the batch.

$$\mathcal{L}_{\mathcal{STN}} = \min_{W_s, b_s} \frac{1}{N}\sum_{i=1}^{N} \|s(x_i, W_s, b_s) - t(x_i, W_t, b_t)\|^2 \tag{3}$$

Here, $\mathcal{L}_{\mathcal{STN}}$ is the loss function for our Student-Teacher Network, N is the number of samples in a batch, x_i is the sample input to VGG16 models s and t with respective weight matrices W_s and W_t.

After minimizing the loss between the tensor outputs of the Student and Teacher, we then fine-tine the compressed model for classification by updating the compressed layer's weights and freezing all others in the model.

3 EXPERIMENTS

We applied deep net triage via local structural compression on the VGG16 network at each of its convolutional blocks, reducing the number of convolutional layers in the block down to one. For each of the five convolutional blocks in VGG16, we train the compressed model with each of the six methods in Section 2. Each combination of VGG-16 variant and training scheme was evaluated on each of the following data sets: MNIST, CIFAR10, CIFAR100, and Stanford Dogs (LeCun & Cortes, 2010; Krizhevsky et al.; Khosla et al., 2011). Each of these four data sets has been

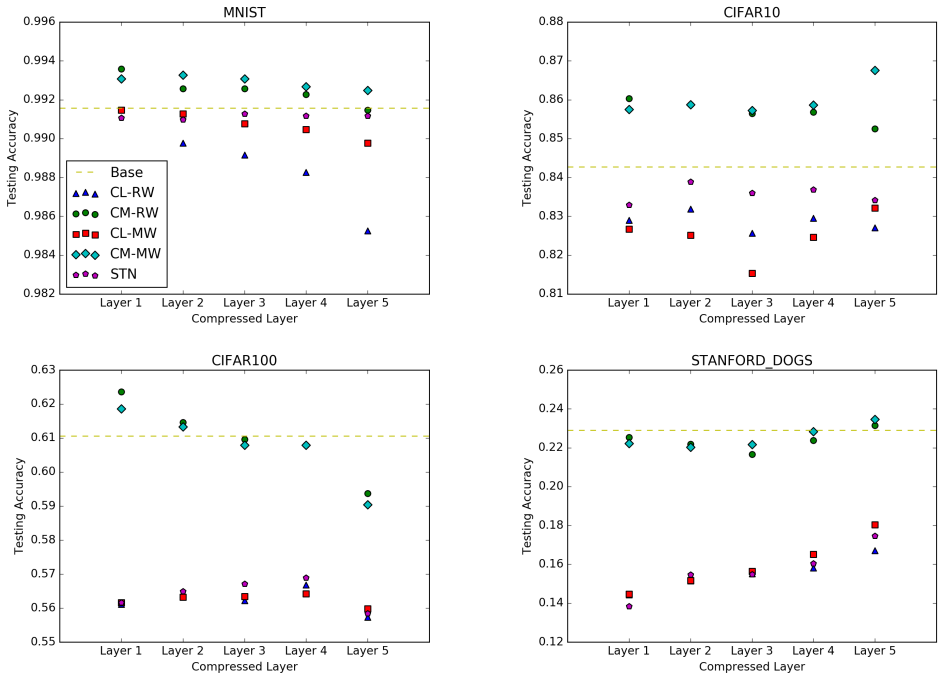

Figure 2: For each data set, experiments are given in terms of which layer was compressed. The five tested experimental regimes are shown along with the baseline achieved by the uncompressed network in yellow.

chosen intentionally to evaluate the validity of structural compression across a range of problem complexities: from coarse-grained MNIST to fine-grained Stanford Dogs.[2]

MNIST The MNIST data set contains 60,000 training and 10,000 testing 28x28 images of greyscale images of digits. To feed MNIST into VGG16, greyscale was expanded into RGB by duplicating the channels and upscaling bilinearly to a final shape of (224, 224, 3). The mean and standard deviation of the training set was calculated to zero-mean and normalize the samples during training and testing.

CIFAR10 The CIFAR10 data set contains 50,000 training and 10,000 testing 32x32 RGB images. The images were bilinearly upsampled to a final size of (224, 224, 3). The training data was used to zero-mean and normalize the samples during training and testing.

CIFAR100 The CIFAR100 data set contains 50,000 training and 10,000 testing 32x32 RGB images of animals and constitutes a finer grained version of the CIFAR10 data set.

Stanford Dogs Stanford Dogs is a 120 class subset of Imagenet which contains 12,000 training and 8,500 testing various dimension RGB images of breeds of dogs. Images were resized bilinearly to (224, 224, 3), mean-zeroed and normalized.

4 Experimental Results

As we intend to assess the criticality of layers in the model through deep net triage, we first consider the maximum accuracy achieved by a compressed model. The compressed models vary by the layer compressed, and the method of triage used.

[2]In all experiments across each data set, hyperparameters were kept constant to eliminate variability in results.

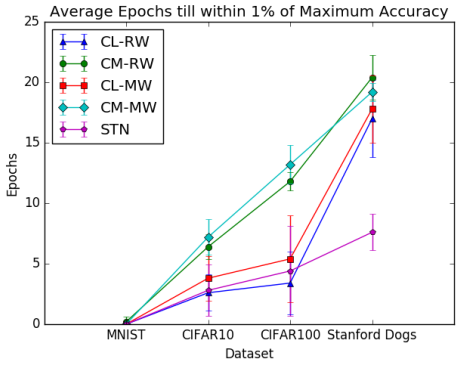

Figure 3: The average number of epochs for each tested method to converge to within 99% of the final maximal accuracy value of the network.

In Figure 2, we show the maximum attained compressed model accuracy for each of the compressed layers trained on every data set. From these figures, we can see that regardless of initialization method, the best performing compressed models are consistently those where weighted updates have has occurred across the entire model: CM-RW and CM-MW. Across all data sets, every model which performs above the baseline performed fine-tuning on all of the weights. This suggests that while layers can be picked apart and replaced independent of the rest of the network—with accuracy still within roughly 5% of the baseline—in order to maximize performance of a given model, one must train across the overall model. This suggests that the representations learned in the compressed layer cannot be fully utilized until the entire model is adapted to that layer too. These figures additionally show that the quality of the compressed representations learned is invariant to what layer they are being learned in: that criticality is equivalent across model layers. All compressed layer models are equally capable of relearning the representations of the parent network, or moreover capable of learning even richer representations than those of the parent.

To summarize, this comparison of maximum accuracy across layers, experiments, and data sets indicates two key concepts: that parent model accuracy can be surpassed by a training regime which retrains on the entire model; and, that the criticality of all layers is equal.

We now consider the number of epochs until each method attains 99% of its maximal accuracy, as shown in Figure 3. Here we can see that a data set's granularity or complexity, is directly proportional to the time for a network trained on it to converge. Additionally, this graphic shows the effects of the various initialization schema. We can see that only a very slight benefit is derived from initializing the compressed layer's weights to an average of those of the parent. We note that initializing from a Student-Teacher network also very clearly speeds up the time for the network to converge across all data sets. This indicates that valuable knowledge is being transferred from the Teacher network to the Student which is helping the network perform better not just at the intermediary layer, but also at the final output. We hypothesize given the results shown in Figure 2 and Figure 3 that if the compressed network pre-trained via the Student-Teacher network were additionally fine-tuned over the entire model, that it would both converge the fastest and achieve the highest accuracy.

5 Conclusions and Future Work

We present a novel method of analyzing deep learning methods which we refer to as *deep net triage*. By drawing from the field of deep network compression and knowledge distillation we design experiments which question the criticality of layers within a network structure, and assess the representations learned therein. We structurally compress a layer at a time, while conducting a series of experiments across these layers on various data sets to infer about the layer's ability to learn representations, recover from compression, and integrate itself into the global network.

We show through experimentation that structurally compressed and fine-tuned models can perform equivalent to, or better than parent, uncompressed models in a layer invariant manner. Additionally, we show that parent-inspired initialization regimes applied only at the layer are unable to compete

with fine-tuning over the entire global model. Lastly, we show that Student-Teacher models evaluated at intermediate layers in the form of hints from uncompressed parent models promote faster convergence to maximal accuracies, despite being unable to outperform full model training methods.

Through this work, we hope to spur others to devise and rigorously test targeted assessments of deep networks, as we do in our *deep net triage*. While, as a community, we may continue to develop ever better performing methods for given problem spaces, we will never truly advance as a field until further intuition for and understanding of deep networks is developed. As optimization and statistical theory progresses on one side, so too must experimentalists approach from the other.

REFERENCES

Lei Jimmy Ba and Rich Caurana. Do deep nets really need to be deep? *CoRR*, abs/1312.6184, 2013. URL http://arxiv.org/abs/1312.6184.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160, 2007.

Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. 2006.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. URL http://arxiv.org/abs/1510.00149.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *CoRR*, abs/1511.06530, 2015. URL http://arxiv.org/abs/1511.06530.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Vadim Lebedev and Victor S. Lempitsky. Fast convnets using group-wise brain damage. *CoRR*, abs/1506.02515, 2015. URL http://arxiv.org/abs/1506.02515.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, pp. 598–605. Morgan-Kaufmann, 1990. URL http://papers.nips.cc/paper/250-optimal-brain-damage.pdf.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014. URL http://arxiv.org/abs/1412.6550.

David Saad and Sara A. Solla. Exact solution for on-line learning in multilayer neural networks. *Phys. Rev. Lett.*, 74:4337–4340, May 1995. doi: 10.1103/PhysRevLett.74.4337. URL https://link.aps.org/doi/10.1103/PhysRevLett.74.4337.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep and convolutional? *arXiv preprint arXiv:1603.05691*, 2016.