IN SEARCH OF THEORETICALLY GROUNDED PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep learning relies on resource-heavy linear algebra operations which can be prohibitively expensive when deploying to constrained embedded and mobile devices, or even when training large-scale networks. One way to reduce a neural network's resource requirements is to sparsify its weight matrices - a process often referred to as pruning. It is typically achieved by removing least important weights as measured by some salience criterion, with pruning by magnitude being the most popular option. This, however, often makes close to random judgments. In this paper we aim to closely investigate the concept of model weight importance, with a particular focus on the magnitude criterion and its most suitable substitute. To this end we identify a suitable Statistical framework and derive deep model parameter asymptotic theory to use with it. Thus, we derive a statistically-grounded pruning criterion which we compare with the magnitude pruning both qualitatively and quantitatively. We find this criterion to better capture parameter salience, by accounting for its estimation uncertainty. This results in improved performance and easier post-pruned re-training.

1 INTRODUCTION

Deep learning is powered by resource intensive computations, to the extent that only models that have been carefully modified to regulate the system resources they consume can be deployed to mobile and wearable technology (Sze et al., 2017). Currently, the management of its energy, latency, and memory demands is one of the most pressing practical challenges. Many argue that fully realizing the potential of deep learning requires a shift towards a structured consideration of these resource requirements and their inclusion in the architecture design and model selection process (LeCun et al., 1990; Han et al., 2015; Yang et al., 2017; Chen et al., 2018).

One way to manage the trained model's resource requirements is to manipulate its size by altering its parameter matrices. Weight matrices can be shrunk by using approximations, such as SVD or Tucker decompositions (Bhattacharya & Lane, 2016; Kim et al., 2015), or sparsification (Dong et al., 2017; Castellano et al., 1997). Alternatively, floating-point precision of values can be reduced, sometimes all the way to binary (Gupta et al., 2015; Rastegari et al., 2016). Finally, specific elements of the matrices can be dropped from the model. This requires the practitioner to specify some importance (salience) criterion which determines the order in which parameters are supposed to be pruned.

State-of-the-art model compression frameworks often make use of pruning with the magnitude salience criterion (Li et al., 2016; Yang et al., 2017; He et al., 2018; Yang et al., 2018), which states that a weight's importance is its magnitude (Han et al., 2015). Evidence suggests this criterion may have behavior very close to pruning weights at random (Molchanov et al., 2016) - a result we further observe in our experiments in this paper. Furthermore, the relative performance of pruning criteria varies considerably and without a discernible pattern across domains and data sets (Augasta & Kathirvalavakumar, 2013). Such results collectively call for a closer investigation of the concept of parameter importance, the salience criteria, and their role in the pruning process.

In this paper we set out to understand and improve the pruning process. Our contributions are:

1. We identified suitable Statistical framework for parameter salience determination.
2. We derived deep model's parameter distribution compatible with the framework.
3. We experimentally verified the superior accuracy and easier post-pruning re-training recovery delivered by the Statistical framework.

We emphasize that in this work, we explicitly do not consider methods that induce sparsity during training or otherwise trade optimization target for extra sparsity.

2 LIMITS OF MAGNITUDE PRUNING FROM A STATISTICAL VIEW

The Magnitude criterion defines parameter salience as their magnitude and intuitively justifies this as the size of their contribution to the network's prediction. While simple to implement, this method makes several implicit assumptions that we enumerate below. Crucially, it assumes all weights to be estimated with the same uncertainty, which is highly unlikely to hold.

Norm-regularization. First implicit requirement of magnitude pruning is for the norm-based regularization to be a part of the model's loss function. Unless the network is norm-regularized, weights that don't get updated remain at their initialized state and thus have random magnitude (salience) values. Thus, it fails to determine weight importance unless aided by the environment.

Normalization. The second implicit assumption is that the data and intermediate activations have been normalized, e.g. through the use of Batch Normalization. Take, for instance, two weights, one estimated at 0.1 the other at 0.01. Magnitude pruning would judge the former as more important because it is larger, arguing that this implies bigger contribution. However, this may not be the case if the former's activation, on average, has 100x lower magnitude.

Uncertainty. Finally, when weights of a neural network obtained during training are seen as mean estimates from some distribution, magnitude pruning does not take into account the uncertainty, the variance, with which they were estimated. In the example above, if the larger weight also has larger associated standard deviation, it is arguably less important to the overall prediction.

3 THEORETICAL FOUNDATION FOR WALD PRUNING

The Wald statistic is a Statistical tool used to test hypotheses about model parameters in general multivariate models. It requires the model's parameter's asymptotic distribution to be approximately normal and estimable. We derive Deep Learning's distribution in this form by recognizing it to be a member of the *M-Estimator* family and utilizing the associated asymptotic theory[1].

3.1 THE ASYMPTOTIC DISTRIBUTION OF PARAMETERS

Let Θ be the parameter space for a general deep model with an objective (loss) function Q_n and parameters θ, where n is the number of data points and $Q_n = \frac{1}{n}\sum m(y, x, \theta)$—an average of some per-example loss function $m(y, x, \theta)$, where (y, x) is an example from the dataset D. Now, let us assume the following:

ID *Identification Condition: Q has a unique minimum at θ_0 in the parameter space Θ .*

CMP *Compactness Condition: Θ is compact (closed and bounded).*

DOM *Domination Condition:* $\mathsf{E} sup_{\theta 2 \Theta} \mathsf{k} m(y, x, \theta) \mathsf{k} < 1$.

WKC *Weak Continuity Condition: m is weakly continuous.*

IID *IID Data:* $\mathsf{f} y_i, x_i \mathsf{g}_{i=1}^{n}$ *is independently identically distributed.*

I *Interior condition: θ_0 is in the interior of (not at the edge of) Θ.*

D *Differentiability condition: m is twice continuously differentiable for any (y, x).*

SD *Score Distribution:* $\frac{1}{n}\sum_{t=1}^{n} s(y_t, x_t, \theta_0) \xrightarrow[d]{} N(0, \Sigma)$, *$\Sigma$ is positive semi-definite and* $s(y_t, x_t, \theta) = \frac{\partial m(y_t, x_t, \theta)}{\partial \theta}$[2]

[1] A detailed argument for deep learning's membership in the family and extended justification of the assumptions are presented in the Appendix. Details behind the distribution result are available in Hayashi (2000).

[2] This is the distribution of the average of loss gradients with respect to the parameter vector evaluated at the true parameter vector and the individual samples.

LDH *Local Dominance on the Hessian:* $\exists \mathcal{N} : \mathsf{E}\left[\sup_{\theta \in \mathcal{N}} \|\mathbf{H}(y_t, x_t, \theta)\|\right] < 1$, *where* $\mathbf{H}(y_t, x_t, \theta) = \frac{\partial^2 m(y_t, x_t, \theta)}{\partial \theta \partial \theta'}$ [3]

HN *Hessian Non-singularity condition (HN):* $\mathsf{E}\left[\mathbf{H}(y_t, x_t, \theta)\right]$ *is non-singular.*

Then, asymptotically, as the sample size n grows to infinity, the model parameters are distributed as:

$$\hat{\theta} \quad N\left(\theta_0, \left(\mathsf{E}\left[\mathbf{H}(y_t, x_t, \theta_0)\right]\right)^{-1} \Sigma \left(\mathsf{E}\left[\mathbf{H}(y_t, x_t, \theta_0)\right]\right)^{-1}\right) \tag{1}$$

The following three assumptions are fundamental to the application of the theory to deep models[4]:

Score Distribution assumption (SD). First, the assumption that is driving the functional form of the parameter distribution above is the estimable normal Score Distribution (SD). Since the object of interest is an average, we can derive its distribution by an application of the Central Limit Theorem (CLT). The specific choice of CLT is domain specific:

1. In applications for which we can assume independence of examples and stable, finite first and second moments of the distribution of gradients, the basic Lindeberg–Lvy CLT can be applied (Billingsley, 1995).
2. When the stability of the second moments can't be assumed, the Lyapunov CLT can be used instead (Hayashi, 2000).
3. For time series applications, we can call on the Gordin's CLT, which relaxes the requirement of example independence to an existence of a limit to the time-series' memory (Hayashi, 2000). That is it assumes past observations to become irrelevant as time passes.

Identification and Hessian Non-singularity conditions (ID & HN). If the model's minima are isolated, the ID can be trivially satisfied if we restrict the parameter space Θ to contain a single minimum. Then, the weight distribution is derived for the given minimum (the one model converged to). Furthermore, the HN is satisfied because the isolated minimum requires the Hessian to be positive definite and hence non-singular. Statistical theory, however, suggests that the minima are likely to be connected, violating the isolation requirement. This was observed in practice by Sagun et al. (2017) and Draxler et al. (2018). In such case, Gill & King (2004) suggest working with an approximation of a weight distribution, as opposed to the exact formulation.

3.2 Deriving the Wald Statistic-based Pruning Criterion

The Wald test is a general framework for testing the significance of explanatory features (Hayashi, 2000). In this context, a lack of significance means that features can be set to 0 with little loss of model performance. The test assumes that the difference between the optimal and the estimated parameters is approximately Normally distributed. We apply the test by considering γ-many weights to be prunable. The null hypothesis H_0, which states that some weights can be set to 0, and Wald's salience measure W would take the following form:

$$H_0 : a\left(\theta_0\right) = 0$$

$$W = n \quad a\left(\hat{\theta}\right)^T \left[\left(\frac{\partial a\left(\theta\right)}{\partial \theta}\right) \Upsilon \left(\frac{\partial a\left(\theta\right)}{\partial \theta}\right)^T\right]^{-1} a\left(\hat{\theta}\right) \qquad \chi^2_\gamma \tag{2}$$

In this equation, Υ is the parameters' covariance matrix and a here is defined as $a(\theta_0) = \mathbf{p}^T\theta_0$, where $\mathbf{p}$ is a vector that contains γ-many 1s and is otherwise 0. If the null hypothesis is true, elements of the parameter vector θ_0 picked out by $\mathbf{p}$ have to be 0.

The salience criterion based on the distribution in Equation 1 is then:

$$W = n \quad a\left(\hat{\theta}\right)^T \left[\left(\frac{\partial a\left(\theta\right)}{\partial \theta}\right) \widehat{Avar}\left(\hat{\theta}\right) \left(\frac{\partial a\left(\theta\right)}{\partial \theta}\right)^T\right]^{-1} a\left(\hat{\theta}\right) \qquad \chi^2_\gamma, \quad \textit{where} \tag{3}$$

[3] So that for any consistent estimator $\bar{\theta}$ we have $\sum_{t=1}^{n} \mathbf{H}(y_t, x_t, \bar{\theta}) \xrightarrow[d]{!} \mathsf{E}\left[\mathbf{H}(y, x, \theta_0)\right]$.

[4] The rest are met close to trivially. Therefore we isolate them in the appendix.

$$\widehat{Avar}\left(\hat{\theta}\right) = \left(\mathsf{E}\left[\mathbf{H}(y_t, x_t, \theta_0)\right]\right)^{-1} \hat{\Sigma} \left(\mathsf{E}\left[\mathbf{H}(y_t, x_t, \theta_0)\right]\right)^{-1}$$

The $\hat{\theta}$ are the weight estimates, $\hat{\Sigma}$ is the estimated score (gradient) covariance matrix.

In the pruning's single weight importance setup $\gamma = 1$. The function $a\left(\hat{\theta}\right)$ will one-hot select the parameter at position k, where k is the investigated parameter's index. This means that the inner portion of the sandwich form in Equation 3 picks out the kk-th (diagonal) element of the $\widehat{Avar}\left(\hat{\theta}\right)$ square matrix. Finally, the W_k, the salience of the k-th parameter is therefore defined as:

$$W_k = n \quad \hat{\theta}_k \left[\widehat{Avar}\left(\hat{\theta}\right)_{kk}\right]^{-1} \hat{\theta}_k \qquad \chi^2_1 \tag{4}$$

This criterion, which we call *Wald criterion* in a nod to its roots in the statistical theory, bares close resemblance to other pruning criteria previously developed and adopted in the deep learning literature:

1. Similarly to magnitude pruning, the magnitude of a weight estimate is captured by $\hat{\theta}_k$.
2. Similarly to Optimal Brain Damage (LeCun et al., 1990), the salience measure uses Hessians but it is not specific to the mean square error loss function.
3. Similarly to N2PS (Tong & Tanaka, 2015), it uses gradient-derived information but only after convergence, when values are stable.

In addition to being derived using statistical theory, Wald pruning can also be justified with intuitions that were used to justify the criteria above, as it incorporates all three of them. Therefore, beside providing a theory-based alternative, this novel use of the Wald statistic in deep model pruning reconciles the differences between previous pruning approaches by appropriately combining them into a single holistic method.

3.3 Asymptotic Distribution Parameter Estimation

Finally, in order to compute the Wald-based salience measure, we need to be able to estimate the asymptotic variance, $Avar\left(\hat{\theta}\right)$, as given in Equation 3. Below we provide the two estimators needed to compute it - the estimator of the Hessians ($\mathbf{H}$) and of the score covariance matrix ($\hat{\Sigma}$).

First, we look at the covariance estimator. Since the necessary conditions of the SD assumption imply the existence of the first and the second score moments, we have, also, justified the existence and estimability of the score variance.

$$\begin{aligned} \mathsf{V}\left[s\left(y_t, x_t, \theta_0\right)\right] &= \mathsf{E}\left[s\left(y_t, x_t, \theta_0\right)^2\right] \quad \mathsf{E}\left[s\left(y_t, x_t, \theta_0\right)\right]^2 \\ &=_1 \mathsf{E}\left[s\left(y_t, x_t, \theta_0\right)^2\right] \quad 0 = \mathsf{E}\left[s\left(y_t, x_t, \theta_0\right)^2\right] \end{aligned} \tag{5}$$

Where $=_1$ holds because the model is assumed to have converged. Then, since $\hat{\theta} \underset{p}{!} \theta_0$, we can write the sample score covariance estimator as:

$$\begin{aligned} \hat{\Sigma} = \mathsf{V}\left[\cancel{p} \frac{1}{n} \sum_{t=1}^{n} s\left(y_t, x_t, \theta_0\right)\right] &= \frac{1}{n} \mathsf{V}\left[\sum_{t=1}^{n} s\left(y_t, x_t, \theta_0\right)\right] \\ &= \frac{n}{n} \mathsf{V}\left[s\left(y, x, \theta_0\right)\right] = \frac{1}{n} \sum_{t=1}^{n} \left[s\left(y_t, x_t, \hat{\theta}\right) s\left(y_t, x_t, \hat{\theta}\right)^T\right] \end{aligned} \tag{6}$$

Where all is defined as above.

Secondly, we focus on the Hessian estimator. Since $\hat{\theta} \underset{p}{!} \theta_0$, the Local Dominance condition further implies that:

$$\mathsf{E}\left[\mathbf{H}(y_t, x_t, \hat{\theta})\right] \underset{p}{!} \mathsf{E}\left[\mathbf{H}(y_t, x_t, \theta_0)\right] = \mathsf{E}\left[\mathbf{H}(y_t, x_t, \theta_0)\right] = \mathsf{E}\left[\mathbf{H}(y_t, x_t, \hat{\theta})\right] \tag{7}$$

where all is defined as above.

Therefore we can rewrite the estimated deep model parameter distribution as:

$$\hat{\theta} \quad N\left(\theta_0, \left(\mathsf{E}\left[\mathbf{H}(y_t, x_t, \hat{\theta})\right]\right)^{-1} \frac{1}{n} \sum_{t=1}^{n} \left[s\left(y_t, x_t, \hat{\theta}\right) s\left(y_t, x_t, \hat{\theta}\right)^T\right] \left(\mathsf{E}\left[\mathbf{H}(y_t, x_t, \hat{\theta})\right]\right)^{-1}\right) \tag{8}$$

where all is defined as above.

This is a fully estimable empirical distribution. It can be used to test weight exclusion hypotheses, as in the pruning setup, but it can also be used to test weight equality restrictions, as would required by quantization applications. Its usefulness, therefore, reaches beyond the present setup.

4 Experiment Results

In this section, we present the experimental results of pruning in several setups using the magnitude and the Wald statistics. We aim to quantify the method's relative performance right after pruning and during as well as after model re-training.

4.1 Method

We adapt Han et al. (2015)'s setup. We use a three-step pruning procedure to prune $p\%$ (i.e., pruning threshold) of weights: (a) a network is trained until convergence; (b) salience score is computed for all parameters and the least salient $p\%$ are set to 0 and frozen; (c) the network is retrained to recover some of the lost performance. We consider two modes of pruning—*one-shot*, where all $p\%$ of parameters are pruned at once, and *iterative*, where parameters are pruned in smaller groups and steps (b) and (c) are repeated until the desired sparsity is reached. In our experiments we consider $p = 60\%, 80\%, 90\%, 92\%, 94\%, 96\%, 98\%, 99\%$.

To approximate Hessian computation for our models, we note that optimizing a model with a Softmax output and the cross-entropy loss function is equivalent to Maximum Likelihood problem[5]. In this case, the approximation $\mathbf{H}(x) = \mathbf{J}(x)^T\mathbf{J}(x)$ holds exactly (Hayashi (2000, p. 510); Newey & Mcfadden (1994, p. 2147)).

Some networks contain Dropout (Srivastava et al., 2014) and Batch Normalization (Ioffe & Szegedy, 2015) layers which have to be adapted for pruned networks. For Dropout, which can be seen as randomized pruning with probability t, we update the Dropout probability to $t^{\overline{p}}$ (Han et al., 2015). For Batch Normalization, we chose to reset the learned mean and variance parameters, letting the network to re-learn them during retraining.

We use the MNIST dataset (LeCun et al., 1998) for our experiments, which is a popular 10-class handwritten digit classification problem, and use TensorFlow (Abadi et al., 2016) to implement the models and perform training.

The purpose of the experiments that follow is to aid the understanding of the pruning decisions and their implications for the post-pruning recovery process. For that reason we select the data set MINST, as it is straightforwardly interpretable and well suited for demonstrating intuitions. While other, larger, data sets exist, critically, MINST allows us to reason, with high degree of confidence, about specific input layer parameters. That would not be practical with more complex data sets. Using this data set, We train architectures LeNet-300-100 and LeNet5, as they are tailored to this

[5]For all deep models with Cross-entropy loss and Softmax output layer, there exists a probability model specification, such that its Maximum Likelihood objective function matches one-to-one the deep network's loss function.

data set. Again, while larger architectures exist, their use would be impractical as their depth would obscure, rather than illuminate the core intuitions of this paper.

4.2 EXPERIMENTS

In what follows we provide three core experiments that examine Walt-based Pruning with magnitude and random pruning criteria.

Salience Map Comparison. In this first experiment, we compare the input layer salience values (i.e. the salience map) for a two-layer MLP under three different criteria. We deliberately design this experiment under MNIST so that it is straightforward to reason qualitatively about what input layer regions should be more important than others.

We make use of the LeNet-300-100 (LeCun et al., 1998), which is a two-layer MLP model, with 300 and 100 hidden units in each layer respectively, followed by the output layer with 10 neurons for each digit class.

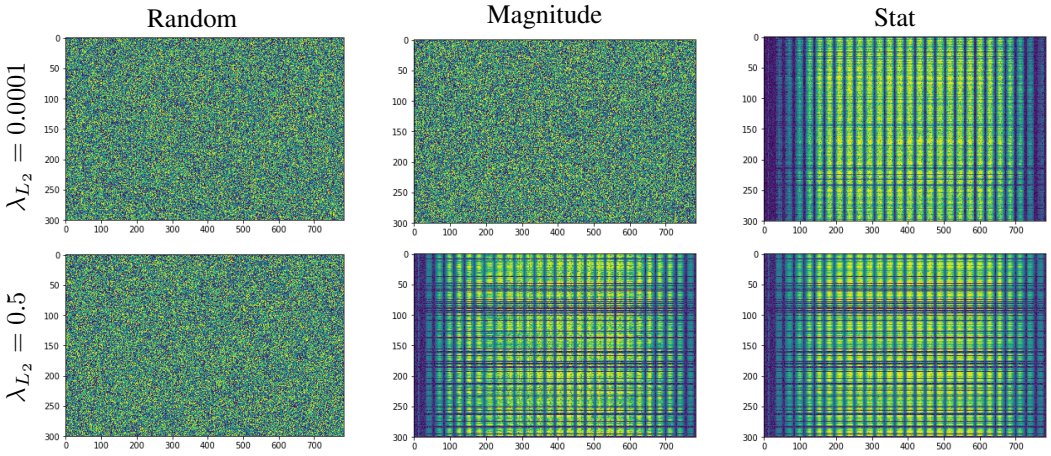

Figure 1: Salience map for the first layer (768 inputs, 300 outputs) of the LeNet-300-100 model for 3 different criteria with L_2 regularization at $\lambda = 0.0001$ and $\lambda = 0.5$. Note that less salient areas correspond to edges of the input image, which are empty in MNIST. Unless regularized heavily, Magnitude criterion is indistinguishable from Random.

The Figure 1 shows the salience maps of the input (784 by 300) fully-connected layer. A reasonable expectation is that the background of the MNIST images ought to be disregarded. It is the same in all examples, thus weights connected to it are at their initialized values.

The Wald pruning exhibits the reasonable behavior across all regularization setups. The magnitude does not. It requires an extreme regularization of around $\lambda = 0.5$ to pick up the same pattern. This, however, comes at a significant cost. The best performing regularization constant $\lambda = 0.0001$ delivered 94.3% accuracy, while the $\lambda = 0.5$ scored only 67.9%. We conclude that only very severe norm-based regularization induces the magnitude to behave in a manner significantly differing from random. This means, that magnitude criterion delivers non-random decisions only in degenerate setups. Wald criterion, on the other hand, correctly predicts background irrelevance irrespective of how much help it receives from the network regularizer.

Compression Frontier - Fully Connected Network. Next, we investigate what the accuracy implications of the criteria are. The Figure 2 shows how the model's accuracy changes compared to the full model (fraction 0.0) as more parameters are pruned (L_2 regularization at $\lambda = 0.0001$). We show results for the aforementioned iterative and one-shot pruning schedules.

As was also observed by Han et al. (2015), the one-shot pruning is strictly inferior in performance. This is to be expected as the criteria were derived for individual / small groups of parameters. The one-shot setup seriously violates this setup.

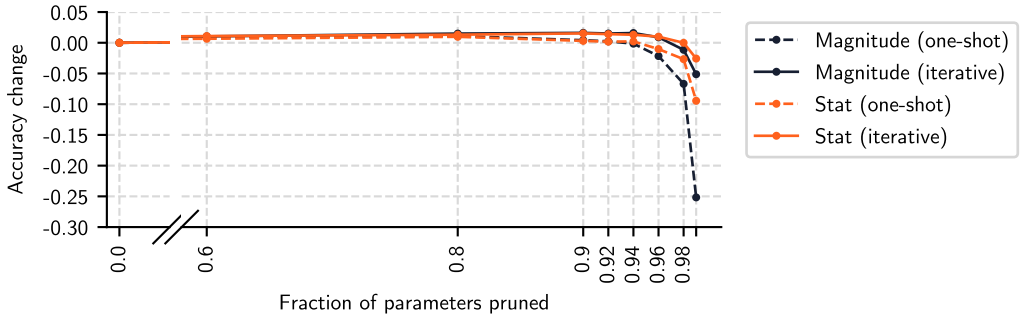

Figure 2: Accuracy versus fraction of parameters pruned of the LeNet-300-100 model for both Wald (Stat) and magnitude pruning, in both one-shot and iterative modes. Wald pushes out the possibility frontier by circa 0.8%, that is roughly 2130 parameters.

In either setup, the Wald criterion pushes the frontier by roughly 0.8% of compression, or equivalently by circa 2.5% of accuracy retention. This is reflected in savings of around 2130 parameters.

The margin of outperformance might feel underwhelming at first, but it has to be recognized that the fully connected layers of LeNet-300-100 have a significant degree of flexibility, whereby weights within a layer are relatively easily substitutable[6]. Consequently, a lot of inefficient pruning is compensated for by more aggressive[7] re-training. Furthermore, the state of the art frameworks compete for every tenth of a percent improvement in either metric (Yang et al., 2017; 2018). Therefore, 0.8% pruning shift and 2.5% accuracy improvement are very significant when viewed in the appropriate scale.

Compression Frontier - Convolutional Network. The natural extension is to test the same pruning-accuracy setup on a convolutional network. We choose to work with LeNet5 (LeCun et al., 1998) for its similarity in size to the LeNet-300-100. It is a model which consists of 2 convolutional and pooling layers followed by 3 fully-connected layers.

Figure 3, as did Figure 2, shows the accuracy change resulting from varying levels of compression.

One-shot pruning, again, outperforms the iterative alternative. This time, however, the importance of the individual weight setup is revealed to be much stronger for the Wald criterion than to the magnitude. This is to be expected if we recognize that the magnitude pruning is likely to have close to random behavior.

Relative to the fully connected setup, we observe larger accuracy drops at the far end of pruning, in line with Han et al. (2015).

Finally, the frontier shift occurs later, indicating that convolution's weight magnitudes are probably more informative about their importance. Further experiments are needed to make that conclusion.

Recovery post-pruning. Overall, the above results suggest convergence of pruned model performance during the re-training process. We test this implication in this final experiment. Figure 4 shows how training loss evolves in the re-training process at an intermediate 90% pruning ratio.

Indeed, the prediction was confirmed for the iterative schedule. The Wald criterion started with a head start, which it gradually, but not completely, lost to flexibility. The one-shot schedule hurt the Wald criterion much more. This reinforces our confidence in the above justifications.

4.3 Discussion

Wald pruning is superior. The Wald criterion successfully internalizes the weights' uncertainty by scaling them by their variances. Consequently, it delivers superior performance with margin that is comparable in scale to the state of the art relevance levels (Yang et al., 2017; 2018).

[6]Activations are not fixed, re-training can re-learn those that were lost if sufficient number of parameters in the reminder of the network were retained.

[7]Larger learning rate.

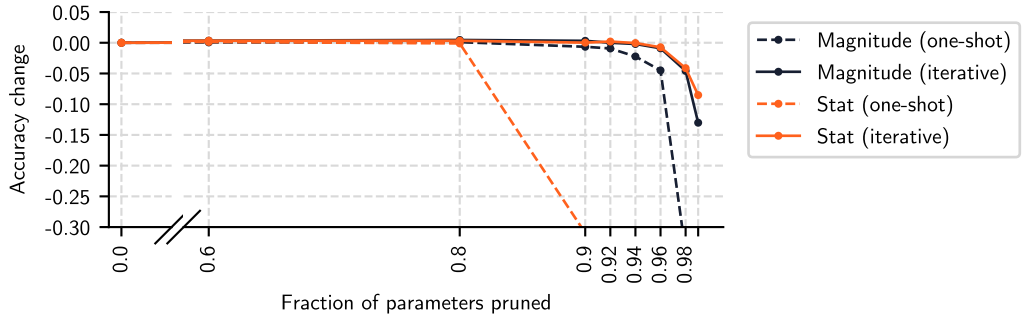

Figure 3: Accuracy versus fraction of parameters pruned of the LeNet5 model for both Wald (Stat) and magnitude pruning, in both one-shot and iterative modes. Wald was derived for individual/small groups of weights, using it en-mass is detrimental.

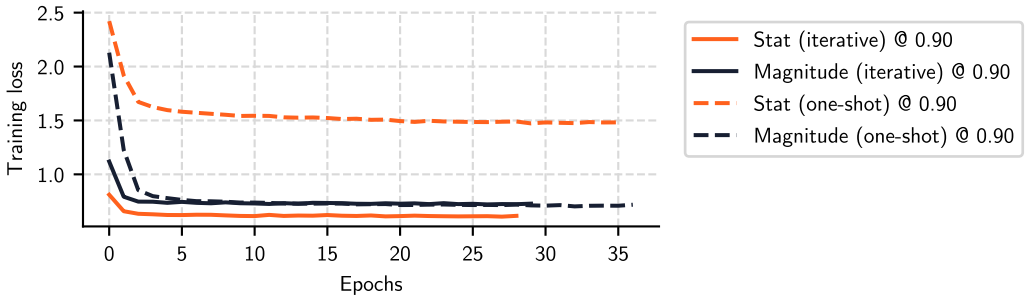

Figure 4: Train loss evolution during re-training of the LeNet5 model after pruning 90% of the weights. When used properly, Wald gains performance advantage at the beginning of the re-training process which it retains throughout.

Wald pruning requires less re-training. Wald pruning retains more important weights in the model which allows the re-training to start at a lower training loss and converge sooner. This is particular relevant for deep models that are expensive to re-train.

Models recover considerably in re-training. The model can recover to original levels of accuracy even when a large fraction of parameters was pruned close to randomly thanks to the easy substitutability of model parameters.

Iterative pruning is preferable. One-shot pruning severely violates the assumptions under which the criteria were computed. Both performances are impacted, with Wald criterion being much more sensitive. Unlike for magnitude pruning, one might straightforwardly derive the Wald criterion for groups of parameters. This would, however require computing such criterion for all possible pruning group compositions, which would be too expensive and defeat the purpose.

5 Conclusion

In this paper we analyzed the use of weight salience pruning with a focus on the magnitude criterion. We argued it fails to account for parameter uncertainty in its salience measure. Therefore, we derived a Statistics-founded, Wald, alternative, which internalizes the consideration. The novel criterion delivers superior accuracy and requires less intensive re-training post-pruning. The over-performance is on scale relevant to the current state of the art.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-

scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

M. Gethsiyal Augasta and T. Kathirvalavakumar. Pruning algorithms of neural networks — a comparative study. *Central European Journal of Computer Science*, 3(3):105–115, 2013.

Sourav Bhattacharya and Nicholas D. Lane. From smart to deep: Robust activity recognition on smartwatches using deep learning, 2016.

Patrick Billingsley. *Probability and measure*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Y. J. Wiley & Sons, New York ; Chichester, 3rd ed. edition, 1995. ISBN 0471007102.

Giovanna Castellano, Anna Maria Fanelli, and Marcello Pelillo. An iterative pruning algorithm for feedforward neural networks. *IEEE Trans. Neural. Networks*, 8(3):519–531, 1997.

Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. Understanding the limitations of existing energy-efficient design approaches for deep neural networks. 2018.

Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *CoRR*, abs/1705.07565:1–15, 2017. URL `http://arxiv.org/abs/1705.07565`.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht. Essentially no barriers in neural network energy landscape. 80:1308–1317, 2018.

Jeff Gill and Gary King. What to do when your hessian is not invertible: Alternatives to model respecification in nonlinear estimation. *Sociological Methods & Research*, 33(1):54–87, August 2004. ISSN 0049-1241.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. February 2015.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626:1–9, 2015. URL `http://arxiv.org/abs/1506.02626`.

Fumio Hayashi. *Econometrics*. Princeton University Press, Princeton, N.J. ; Oxford, 2000. ISBN 0691010188.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

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. November 2015.

Yann LeCun, John S. Denker, and Sara A. Solla. *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-100-7. URL `http://dl.acm.org/citation.cfm?id=109230.109298`.

Yann LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 0018-9219.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. URL `http://arxiv.org/abs/1608.08710`.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, abs/1611.06440, 2016. URL `http://arxiv.org/abs/1611.06440`.

Whitney K. Newey and Daniel Mcfadden. Chapter 36 large sample estimation and hypothesis testing. *Handbook of Econometrics*, 4:2111–2245, 1994. ISSN 1573-4412.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. March 2016.

Levent Sagun, Utku Evci, Ugur V. Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *CoRR*, abs/1706.04454:1–15, 2017. URL http://arxiv.org/abs/1706.04454.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 2017.

Zhiqiang Tong and Gouhei Tanaka. A pruning method based on weight variation information for feedforward neural networks. *IFAC-PapersOnLine*, 48(18):221 – 226, 2015. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2015.11.040. URL http://www.sciencedirect.com/science/article/pii/S2405896315022983. 4th IFAC Conference on Analysis and Control of Chaotic Systems CHAOS 2015.

Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning, 2017.

Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications, 2018.

APPENDIX

5.1 DEEP LEARNING AS M-ESTIMATOR

Given a Statistical model parameterized by θ, and the data $D = f(x_i, y_i)g_{i=1}^n$, an Extremum Estimator is an estimator that maximizes a given scalar value of an objective function $Q(\theta, D_n)$, subject to constrained parameter space, Θ. An M-Estimator is a special case of an Extremum Estimator with the objective function defined to be a sample average of a real valued datum-defined function:

$$\hat{\theta} := argmax\left[Q_n(\theta)\right] \; s.t. \; \theta \; 2 \; \Theta \quad \mathsf{R}^p \tag{9}$$

$$Q_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} m(y_i, x_i, \theta) \tag{10}$$

where $m(y_i, x_i, \theta)$ is a real valued function of the i-th datum (y_i, x_i) (the label y_i, the input x_i) and the parametrization θ. We are free to specify it as the Deep Learning's loss function:

$$m(y_i, x_i, \theta) = \quad \Lambda\left[y_i, \Phi(x_i, \theta)\right]$$

where $\Lambda[.,.]$ is the loss function[8] defined on the label and the prediction (square loss, cross-entropy etc.), and $\Phi(x_i, \theta)$ is the a priori known functional representation of the network parameterized by θ. Changing the sign and turning it into a minimization gets us to:

$$\hat{\theta} := argmin\left[\frac{1}{n}\sum_{i=1}^{n} \Lambda\left[y_i, \Phi(x_i, \theta)\right]\right] \; s.t. \; \theta \; 2 \; \Theta \quad \mathsf{R}^p \tag{11}$$

This is then equivalent to the minimization objective of the given neural network $\Phi(.,.)$. Deep Models can be, therefore, constructed as a special case of an M-Estimator.

5.2 STRAIGHTFORWARDLY MET ASSUMPTIONS

The trivially met assumptions are: the DOM which is met for a reasonably small neighborhood Θ - of which we have a full control; WKC is met trivially for all commonly used Deep Learning loss functions, most notably the cross-entropy; IID is met trivially for a data set of a reasonable quality; the I condition is satisfied as we are fully flexible to pick the parameter space Θ.

The CMP, compactness, may seem to be a strong assumption, but it is not. An existence of an arbitrary compact and bound parameter sub-space suffices, therefore we don't need the ability to neither quantify nor identify it. This better translates as our belief that no single parameter can take an infinitely large value. In Deep Learning, this expresses our preference against dominating parameters that alone drive the predictions of a network. That is, this is an assumption of a non-overfitted model.

The D condition, is satisfied for most networks with an important caveat. The ReLU activation function is non differentiable around zero. This is ultimately carried over to the objective function. Strictly speaking this is a violation of the assumption. However, two options are open to us. First, we can replace the ReLU with its twice differentiable asymptotic equivalent the SoftPlus activation function ($\sigma_{ReLU}(X) = lim_{\alpha \; ! \; 1}\left[\frac{1}{\alpha} log\left(1 + e^{\alpha X}\right)\right]$). Second, we can recognize that ReLU is a limit of a function which satisfies this assumption, and treat it as an approximation within the machine precision.

The LDH assumption states that the norm of the hessian is finite at the optimum. In other words, that the gradients in neither direction increase at an infinite rate. We can either check it numerically, or we can derive it for some layer types from easier to check conditions. For softmax-type output layers, for instance, this condition is equivalent to the full rank of the previous hidden layer node

[8] We change the sign in order to stay in the maximization paradigm.

covariance matrix. The proof follows exactly the same lines as that for the logit with log likelihood objective function in Hayashi (2000, p. 510). This means that for as long as the previous layer does not contain perfectly correlated nodes, the loss function hessian will be bounded and thus estimable. Such feature correlation would imply an obvious redundancy that can be diagnosed and remedied by dropping all but one of the correlated features. A re-training would be needed afterwards.