# The Difficulty of Training Sparse Neural Networks

**Anonymous Authors**[1]

## Abstract

We investigate the difficulties of training sparse neural networks and make new observations about optimization dynamics and the energy landscape within the sparse regime. Recent work of (Gale et al., 2019; Liu et al., 2018) has shown that sparse ResNet-50 architectures trained on ImageNet-2012 dataset converge to solutions that are significantly worse than those found by pruning. We show that, despite the failure of optimizers, there is a linear path with a monotonically decreasing objective from the initialization to the "good" solution. Additionally, our attempts to find a decreasing objective path from "bad" solutions to the "good" ones in the sparse subspace fail. However, if we allow the path to traverse the dense subspace, then we consistently find a path between two solutions. These findings suggest traversing extra dimensions may be needed to escape stationary points found in the sparse subspace.

## 1. Introduction

Reducing parameter footprint and inference latency of machine learning models is an active area of research, fostered by diverse applications like mobile vision and on-device intelligence. Sparse networks, that is, neural networks in which a large subset of the model parameters are zero, have emerged as one of the leading approaches for reducing model parameter count. It has been shown empirically that deep neural networks can achieve state-of-the-art results under high levels of sparsity (Han et al., 2015; Louizos et al., 2017; Gale et al., 2019) , and this property has been leveraged to significantly reduce the parameter footprint and inference complexity (Kalchbrenner et al., 2018) of densely connected neural networks. However, pruning-based sparse solutions require dense connectivity during training and uses the same, or even greater, computational resources compared to fully dense training, which imposes an upper limit on the size of sparse networks we can train.

**Training Sparse Networks.** In the context of sparse networks, state-of-the-art results have been obtained through training densely connected networks and modifying their topology during training through a technique known as pruning (Zhu & Gupta, 2018; Narang et al., 2017; Han et al., 2015). A different approach is to *reuse* the sparsity pattern found through pruning and train a sparse network from scratch. This can be done with a random initialization (**"scratch"**) or the same initialization as the original training (**"lottery"**). Previous work (Gale et al., 2019; Liu et al., 2018) demonstrated that both approaches achieve similar final accuracies, but lower than pruning[1]. The difference between pruning and both approaches to training while sparse can be seen in Figure 1. Despite being in the same energy landscape, "scratch" and "lottery" solutions fail to match the performance of the solutions found by pruning. Given the utility of being able to train sparse from scratch, it is critical to understand the reasons behind the failure of current techniques at training sparse neural networks.

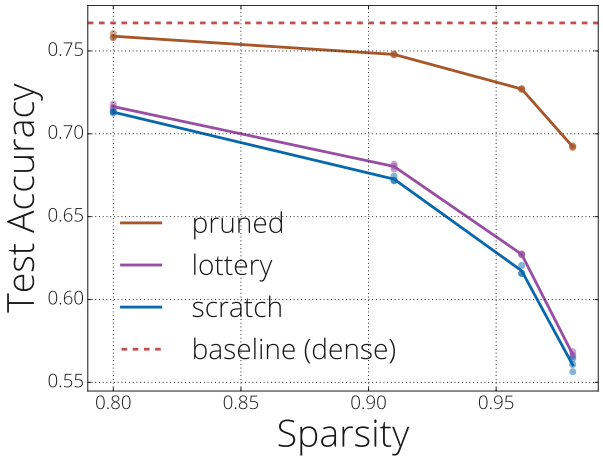


*Figure 1.* **Test accuracy of ResNet-50 networks trained on ImageNet-2012 dataset at different sparsity levels**. We observe a large gap in generalization accuracy between approaches based on pruning and other approaches. See text for details.

There exists a line of work on training sparse networks (Bellec et al., 2018; Mocanu et al., 2018; Liu et al., 2019; Mostafa & Wang, 2019) which allows the connectivity pattern to change over time. These techniques generally achieve higher accuracy compared with fixed sparse connectivity, but generally worse than pruning. Though promising, the role of changing connections during the optimization

[1](Frankle et al., 2019) closes this gap by using trained parameters from 5th epoch to initialize the network. However they don't quantify how different these values are from the solution.

is not clear. While we focus on *fixed* sparsity pattern in this work, our results give insight into why these other approaches are more successful.

Motivated by the disparity in accuracy observed in Figure 1, we perform a series of experiments to improve our understanding of the difficulties present in training sparse neural networks, and identify possible directions for future work.

More precisely, our **main contributions** are:

- A set of experiments showing that the objective function is monotonically decreasing along the straight lines that interpolate from:
  - the original dense initialization
  - the original dense initialization projected into the sparse subspace
  - a random initialization in the sparse subspace

  to the solution obtained by pruning[2]. Demonstrating that even when the optimization process fails, there was a monotonically decreasing path to the solution.
- In contrast, the linear path between the *scratch* and the *pruned* solutions depicts a high energy barrier between the two solutions. Our attempts to find quadratic and cubic Bézier curves (Garipov et al., 2018) with a decreasing objective between the two sparse solutions fails suggesting that the optimization process gets attracted into a "bad" local minima.
- Finally, by removing the sparsity constraint from the path, we are consistently able to find decreasing objective Bézier curves between the two sparse solutions. This result suggest that allowing for dense connectivity might be necessary and sufficient to escape the stationary point converged in the sparse subspace.

The rest of the paper is organized as follows: In §2, we describe the experimental setup. In §3 we present the results from these experiments, followed by a discussion in §4.

## 2. Methods

**Training methods.** The different training strategies considered in this paper are summarized in Figure 2. In *dense training*, we train the densely connected model and apply model pruning (Zhu & Gupta, 2018) to find the **Pruned Solution (P-S)**. The other strategies start instead with a sparse connectivity pattern (represented as a binary mask) obtained from the pruned solution. The solution obtained from the same random initialization as the pruned solution (Frankle & Carbin, 2018) is denoted **Lottery Solution (L-S)**, while the

[2] sparse subspace refers to the sparsity pattern found by pruning and same in all settings.

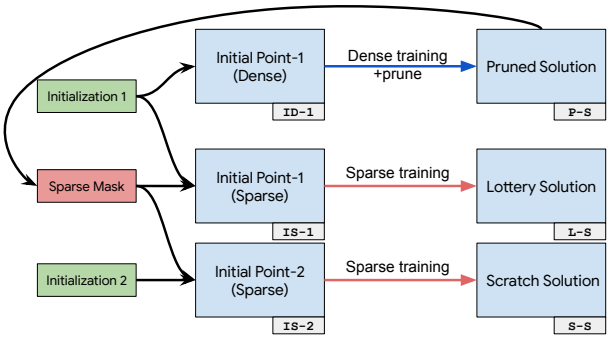


*Figure 2.* **Experimental setup**. In this paper we consider three different methods for obtaining sparse solutions: pruned, lottery and scratch. Pruned solution is obtained by starting with a densely connected network and gradually removing connections during training, whereas the other two solutions are obtained by training sparse networks from start.

solution obtained from another random initialization (Liu et al., 2018) is named **Scratch Solution (S-S)**. All of our experiments are based on the Resnet-50 architecture (Wu et al., 2018) and the Imagenet-2012 dataset (Russakovsky et al., 2015). Abbreviations defined in Figure 2 below the boxes are re-used to in the remaining of the text to indicate start and end points of the interpolation experiments.

**Pruning strategy.** We use magnitude based model pruning (Zhu & Gupta, 2018) in our experiments. This has been shown(Gale et al., 2019) to perform as well as the more complex and computationally demanding variational dropout (Molchanov et al., 2017) and $\ell_0$ regularization approaches (Christos Louizos, 2018). In our experiments, we choose the 3 top performing pruning schedules for each sparsity level using the code and checkpoints provided by Gale et al. (2019). The hyper-parameters involved in the pruning algorithm were found by grid search separately for each sparsity level. The 80% sparse model loses almost no accuracy over the baseline, while the 98% sparse model drops to 69% top-1 accuracy (see Figure 1-*pruned*). Training details and the exact pruning schedules used in our experiments are detailed in Appendix A.

**Interpolation in parameter space.** Visualizing the energy landscape of neural network training is an active area of research. Goodfellow et al. (2015) measured the training loss on MNIST (Lecun et al., 1998) along the line segment between the initial point $\theta_s$ and the solution $\theta_e$, observing a monotonically decreasing curve. Motivated by this, we were curious if this was still true (a) for Resnet-50 on Imagenet-2012 dataset and (b) if it was still true in the sparse subspace. We hypothesized that if (a) was true but (b) was not, then this could help explain some of the training difficulties encountered with sparse networks.

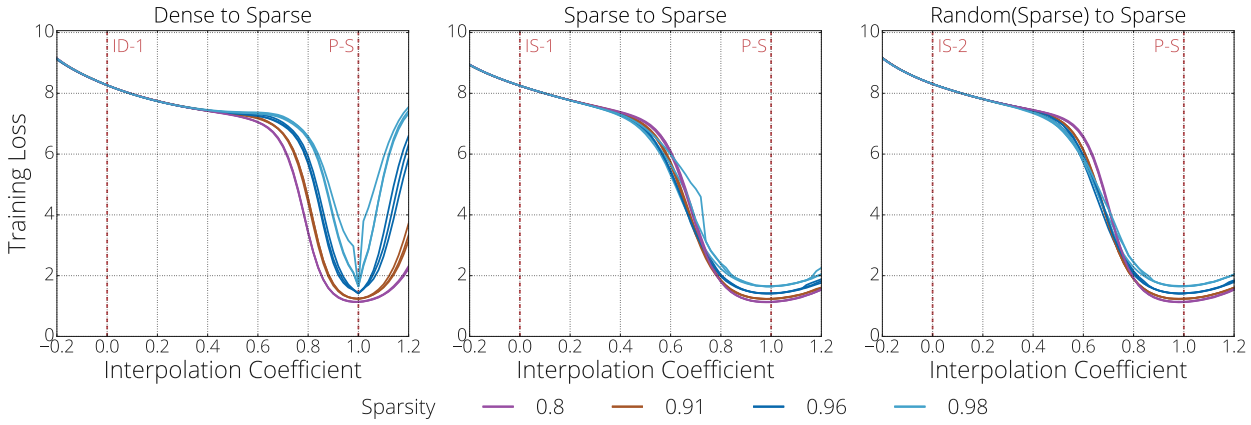


*Figure 3.* **Linear interpolation experiments** between various initial and final points. Interpolations are created with 0.02 increments and evaluated on 500k data-augmented images from training set. Initial and final points (corresponding to coefficients 0 and 1 respectively) are labeled with abbreviations as presented in Figure 2. From all points considered there exist a monotonically decreasing path to the solution found through pruning.

In our linear interpolation experiments, we generate networks along the segment $\theta = t\theta_{\mathbf{e}} + (1-t)\theta_{\mathbf{s}}$ for $t \in [-0.2, 1.2]$ with increments of $0.01$ and evaluate them on the training set of 500k images. Interpolated parameters include the weights, biases and trainable batch normalization parameters. We enable training mode for batch normalization layers so that the batch statistics are used during the evaluation. The objective is identical to the objective used during training, which includes a weight decay term scaled by $10^{-4}$.

We now seek to find a non-linear path between the initial point and solution using parametric Bézier curves of order $n = 2$ and 3. These are curves given by the expression

$$B_n(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \theta_i \ ,$$

where $\theta_0 = \theta_e$ and $\theta_n = \theta_s$ . We optimize the following,

$$\min_{\theta_1, \cdots, \theta_{n-1}} \int_0^1 L(B_n(t))dt \ ,$$

using the stochastic method as proposed by Garipov et al. (2018) with a batch size of 2048 where $L(\theta)$ denotes the training loss as a function of trainable parameters. Mirroring our original training settings, we set the weight decay coefficient to $10^{-4}$. We performed a hyper-parameter search over base learning rates $(1, 10^{-1}, 10^{-2}, 10^{-3})$ and momentum coefficients $(0.9, 0.95, 0.99, 0.995)$, obtaining similar learning curves for most of the combinations. We choose $0.01$ as the base learning rate and $0.95$ as the momentum coefficient for these path finding experiments.

## 3. Results and Discussion

Our experiments highlight a gap in our understanding of energy landscape of sparse deep networks. Why does training a sparse network from scratch gets stuck at a neighborhood of a stationary point with a significantly higher objective? This is in contrast with recent work that has proven that such a gap does not exist for certain kinds of over-parameterized dense networks (Sagun et al., 2014; Choromanska et al., 2014). Since during pruning dimensions are slowly removed, we conjecture that this prevents the optimizer from getting stuck into "bad" local minima. The failure of the optimizer is even more surprising in the light of the linear interpolation experiments of Section 3.1, which show that sparse initial points are connected to the pruning solutions through a path in which the training loss is monotonically decreasing.

In high dimensional energy landscapes, it is difficult to assess whether the training converges to a local minimum or to a higher order saddle point. Sagun et al. (2017) shows that the Hessian of a convolutional network trained on MNIST is degenerate and most of its eigenvalues are very close to zero indicating an extremely flat landscape at solution. Dauphin et al. (2014) comments on Bray & Dean (2007)'s results and argues that critical points that are far from the global minima are most likely to be saddle points. In Section 3.2, we examine the linear interpolation between solutions and attempt to find a parametric curve between them with decreasing loss. This is because finding a decreasing path from the high loss solution ("scratch") to the low loss solution("pruned") would demonstrate that the former solution is at a saddle point.

### 3.1. Path between start and end

Linear interpolations from Initial-Point-1 (Dense), Initial-Point-1 (Sparse) and Initial-Point-2 (Sparse) to Pruned Solution at different sparsity levels are shown in Figure 3 respectively; all cases show monotonically decreasing curves. The

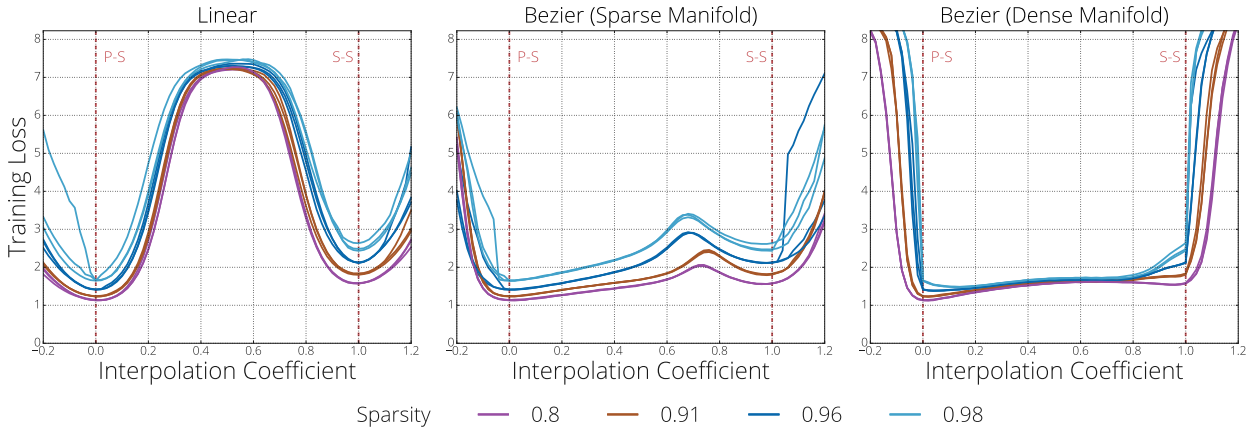


*Figure 4.* Interpolation experiments between pruned (P-S) and scratch(S-S) sparse solutions: **(left)** linear interpolation **(middle, right)** Bézier curves minimized in (sparse, dense) manifolds. Loss values are calculated using 500k images from the training set.

training loss represented in the y-axis consists of a cross entropy loss and an $\ell_2$ regularization term. While in Figure 3 the $y$ axis represents the full training loss, the two terms composing this loss are shown separately in Appendix C. There we observe that the sum is dominated by the cross entropy loss.

In Figure 3-left, we observe a long flat plateau followed by a sharp drop: this is unlike typical learning curves, which are steepest at the beginning and then level off. Model pruning allows the optimizer to take the path of steepest descent while still allowing it to find a good solution as dimensions are slowly removed.

Finally, the linear interpolation from a random point sampled from the original initialization distribution ("scratch") also depicts a decreasing curve (Figure 3-right), almost identical to the interpolations that originates from the lottery initialization (Figure 3-middle). This brings further evidence against the "lottery ticket" initialization being special relative to other initializations.

### 3.2. Path between two solutions

The training loss along the linear segment and the parametric Bézier curve connecting the scratch and the pruned solutions are shown in Figure 4. As observed by (Keskar et al., 2017), linear interpolation (Figure 4-left) depicts a barrier between solutions, as high as the values observed by randomly initialized networks. The sparse parametric curve (Figure 4-middle) found through optimization also fails at connecting the two solutions with a monotonically decreasing path (although it has much smaller loss value than the straight line). Using a third order Bézier curve also fails to decrease the maximum loss value over the second order curve (Appendix D). The failure of the third order curve does not prove that a path cannot be found. However, as a second order curve was sufficient to connect solutions in dense networks (Garipov et al., 2018), it does show that if such a path exists, then it must be significantly more complex than those necessary in dense networks.

We continue our experiments by removing the sparsity constraint from the quadratic Bézier curve and optimize over the full parameter space (Figure 4-right). With all dimensions unmasked, our algorithm consistently finds paths along which the objective is significantly smaller. While this path is not strictly monotonically decreasing, this is not unexpected, given that our algorithm minimizes the integral of the objective over the interpolation segment and so monotonicity is not enforced. We leave the exploration of monotonically decreasing paths for future work.

## 4. Conclusion and Future Work

Our work provides insights into the dynamics of optimization in the sparse regime which we hope will guide progress towards better regularization techniques, initialization schema, and/or optimization algorithms for training sparse networks.

Training of sparse neural networks is still not fully understood from an **optimization** perspective. In the sparse regime, we show that optimizers converge to stationary points with a sub-optimal generalization accuracy. This is despite monotonically decreasing paths existing from the initial point to the pruned solution. And despite nearly monotonically decreasing paths in the dense subspace from the "bad" local minimum to the pruned one.

Optimizers that avoid "bad" local minima in the sparse regime are yet to be found. We believe that understanding why popular optimizers used in deep learning fail in the sparse regime will yield important insights leading us towards more robust optimizers in general.

## References


Bellec, G., Kappel, D., Maass, W., and Legenstein, R. A. Deep Rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018.

Bray, A. J. and Dean, D. S. Statistics of critical points of Gaussian fields on large-dimensional spaces. *Phys. Rev. Lett.*, 2007.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The Loss Surface of Multilayer Networks. *ArXiv*, 2014.

Christos Louizos, Max Welling, D. P. K. Learning Sparse Neural Networks through $L_0$ Regularization. In *International Conference on Learning Representations*, 2018.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and Attacking the Saddle Point Problem in High-dimensional Non-convex Optimization. In *Advances in neural information processing systems*, 2014.

Frankle, J. and Carbin, M. The Lottery Ticket Hypothesis: Training Pruned Neural Networks. *ArXiv*, 2018.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. The Lottery Ticket Hypothesis at Scale. *ArXiv*, 2019.

Gale, T., Elsen, E., and Hooker, S. The State of Sparsity in Deep Neural Networks. *ArXiv*, 2019.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In *Advances in Neural Information Processing Systems*, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

Goodfellow, I. J., Vinyals, O., and Saxe, A. M. Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations*, 2015.

Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *ArXiv*, 2017.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, 2015.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A., Dieleman, S., and Kavukcuoglu, K. Efficient Neural Audio Synthesis. In *International Conference on Machine Learning*, 2018.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations*, 2017.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the Value of Network Pruning. *ArXiv*, 2018.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Evolving and Understanding Sparse Deep Neural Networks using Cosine Similarity. *ArXiv*, 2019.

Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, 2017.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 2018.

Molchanov, D., Ashukha, A., and Vetrov, D. Variational Dropout Sparsifies Deep Neural Networks. In *International Conference on Machine Learning*, 2017.

Mostafa, H. and Wang, X. Parameter Efficient Training of Deep Convolutional Neural Networks by Dynamic Sparse Reparameterization. *ArXiv*, 2019.

Narang, S., Diamos, G. F., Sengupta, S., and Elsen, E. http://arxiv.org/abs/1704.05119. *ArXiv*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein,

M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

Sagun, L., Ugur Guney, V., Ben Arous, G., and LeCun, Y. https://ui.adsabs.harvard.edu/abs/2014arXiv1412.6615S. *ArXiv*, 2014.

Sagun, L., Evci, U., Güney, V. U., Dauphin, Y., and Bottou, L. Empirical Analysis of the Hessian of Over-Parametrized Neural Networks. In *International Conference on Learning Representations*, 2017.

Wu, S., Zhong, S., and Liu, Y. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, 2018.

Zhu, M. and Gupta, S. To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. In *International Conference on Learning Representations*, 2018.

## A. Experimental details

Our experiments use the code made publicly available by [3]. The pruning algorithm uses magnitude based iterative strategy to reach to a predefined final sparsity goal over the coarse of the training. We use a batch size of 4096 and train the network with 48000 steps (around 153 epochs). Our learning rate starts from 0 and increases linearly towards 0.1 in 5 epoch and stays there until 50th epoch. The learning rate is dropped by a factor of 10 afterwards at 50th, 120th and 140th epochs (Goyal et al., 2017).

*Table 1.* Pruning strategies used in various settings.

| SPARSITY | START | END | FREQUENCY |
|---|---|---|---|
| 0.8 | 12500 | 40000 | 2000 |
| | 12500 | 40000 | 500 |
| | 12500 | 36000 | 1000 |
| 0.91 | 12500 | 36000 | 2000 |
| | 10000 | 40000 | 4000 |
| | 7500 | 36000 | 1000 |
| 0.96 | 7500 | 36000 | 1000 |
| | 10000 | 32000 | 500 |
| | 7500 | 40000 | 2000 |
| 0.98 | 12500 | 36000 | 500 |
| | 7500 | 32000 | 500 |
| | 12500 | 32000 | 500 |

Due to high sensitivity observed, we don't prune the first convolutional layer and cap the maximum sparsity for the final fully connected layer with 80%. Top 3 performing pruning schedules selected for each sparsity level are shared in Table 1. We calculate the average $\ell_1$ norm of the gradient for the first setting in the table and obtain 4e-6. As is the case for most iterative (especially stochastic) methods, our solutions do not qualify as stationary points since the gradient is never exactly zero. By slight abuse of language we refer to stationary point as any point where the gradient is below of $10^{-5}$.

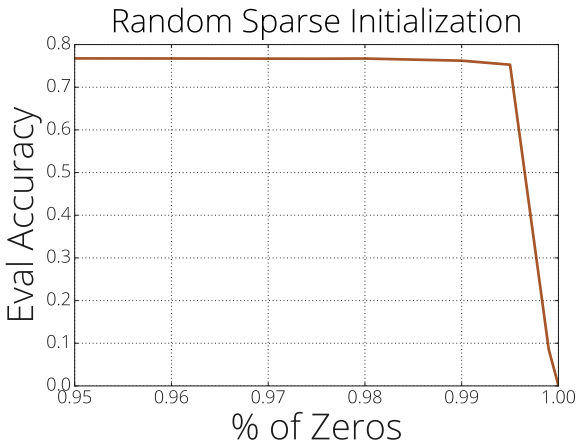


*Figure 5.* At the beginning of the training we randomly set a fraction of weights to zero and train the network with default parameters for 32k steps. We observe a sudden drop only if more than 99% of the parameters are set to zero.

## B. Sparse Initialization

Initialization methods that control variance of activations during the first forward and backward-pass is known to be crucial for training deep neural networks (Glorot & Bengio, 2010; He et al., 2015). However, with batch normalization (Ioffe & Szegedy, 2015) and skip connections the importance of initialization is expected to be less pronounced. *sparse-init* experiments shared in Figure 1 can be seen as a demonstration of such tolerance. In sparse-init experiments we train a dense

[3] https://github.com/google-research/google-research/tree/master/state_of_sparsity

ResNet-50 but use the sparse binary mask found by pruning to set a fraction of initial weights to zero. At all sparsity levels considered (0.8, 0.91, 0.96, 0.98), we observe that the training succeeds and reaches to final accuracy around 76% matching the performance of the original training. Thus the initialization point alone cannot be the reason for the failure of sparse training.

To understand the extent which we are able to train sparsely initialized networks without compromising performance, we perform experiments where we randomly set a given fraction of weights to zero. Figure 5 shows the results. We observe no significant difference until 99.5% after which we observe a sharp drop in performance. The initialization requires a very small number of non-zero weights to succeed.

## C. Loss Decomposition

Figure 6 depicts the value of the $\ell_2$ regularization term over the linear interpolations described in Section 3.1. The curve demonstrates that the solutions found are consistently of lower weight magnitude than their initialization, and they also demonstrate that the regularization terms are a factor of ten smaller than the objective function.

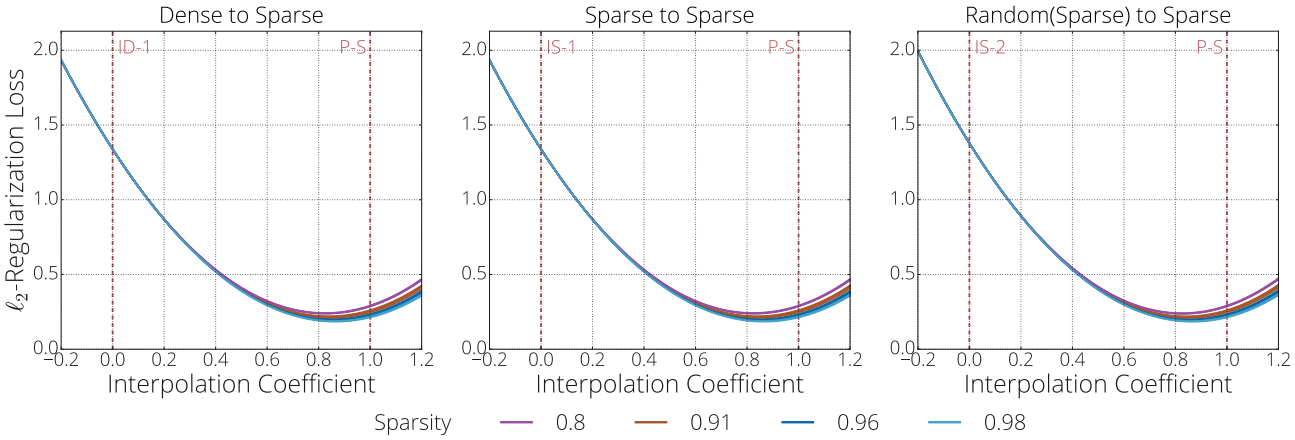


*Figure 6.* Regularization term of the linear interpolations of Figure 3.

Figure 7 depicts the value of the cross entropy loss over the linear interpolation described above. Figure 7-(left) the sparse to sparse and random sparse to sparse interpolation maintain the monotonic decreasing pattern observed in the interpolations plots of the total loss (Figure 3), whereas dense to sparse interpolation shows a slight increase in the objective before a rapid descent.

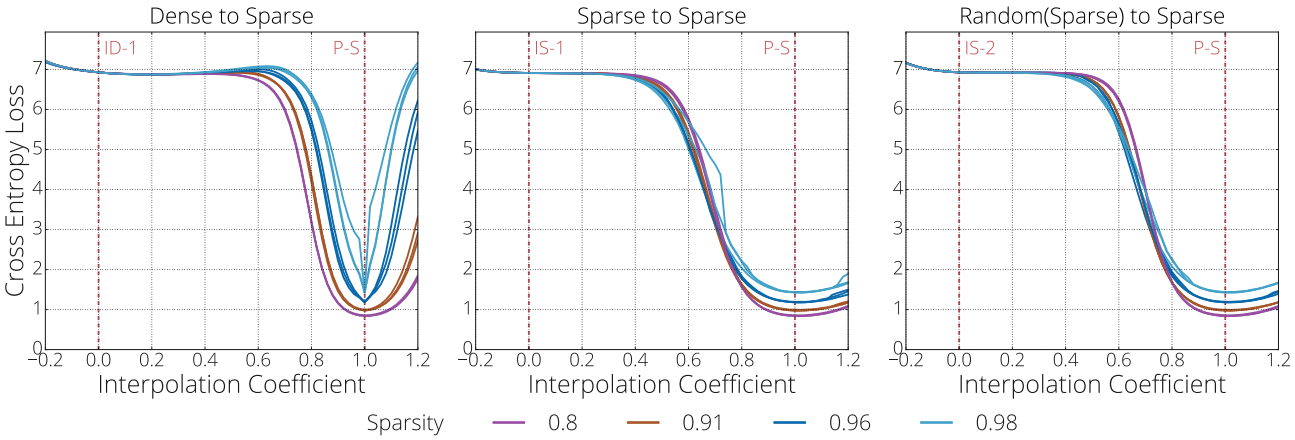


*Figure 7.* Cross entropy term of the linear interpolations of Figure 3.

Dense-sparse cross entropy is increasing. Time to drop is much less with original initializations, higher with random and

least with dense.

## D. Path Finding Experiments with Cubic Bézier Curve

In Section 3.2 our experiments fail to find paths along which the loss is decreasing between the "pruned" and "scratch" solutions. Would optimizing more complex parametric curves change the result? Figure 8 depicts the objective along the third order Bézier curves. Though the integral of the loss over the segment between solutions seems less than Figure 4-middle, we still observe a small barrier between solutions.

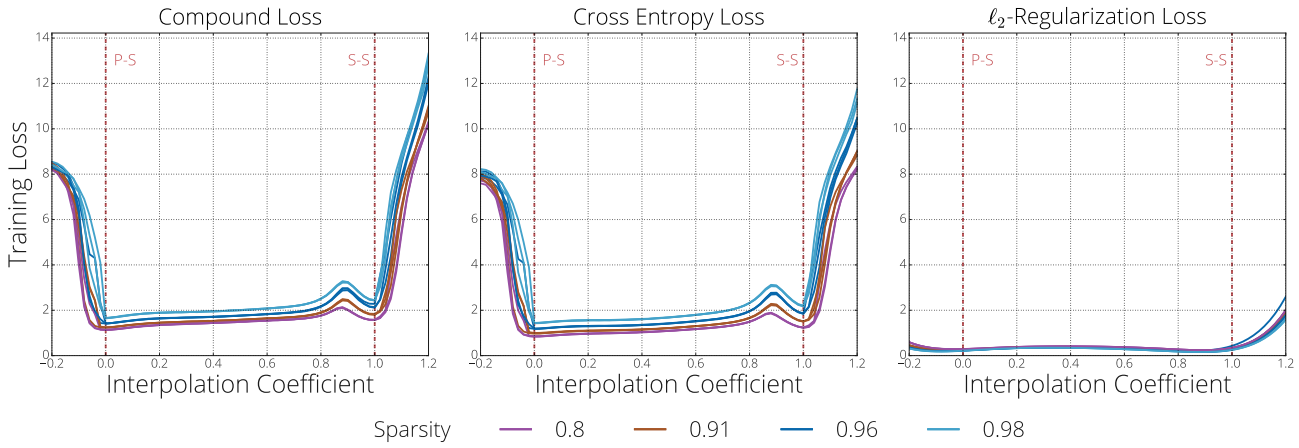


*Figure 8.* Training loss along the third order (cubic) Bézier curve found between the "pruning" and "scratch" solutions.