References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, third edition, 1999.

[3] Grey Ballard, James Demmel, Laura Grigori, Mathias Jacquelin, Hong Diep Nguyen, and Edgar Solomonik. Reconstructing householder vectors from tall-skinny qr. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014.

[4] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. `http://www.fairmlbook.org`.

[5] Cenk Baykal, Lucas Liebenwein, Igor Gilitschenski, Dan Feldman, and Daniela Rus. Sipping neural networks: Sensitivity-informed provable pruning of neural networks. *CoRR*, abs/1910.05422, 2019. URL `http://arxiv.org/abs/1910.05422`.

[6] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

[7] Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 968–977. SIAM, 2009.

[8] Peter Businger and Gene H Golub. Linear least squares solutions by householder transformations. *Numerische Mathematik*, 7(3):269–276, 1965.

[9] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[10] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/32e0bd1497aa43e02a42f47d9d6515ad-Paper.pdf`.

[11] Tony F Chan and Per Christian Hansen. Some applications of the rank revealing qr factorization. *SIAM Journal on Scientific and Statistical Computing*, 13(3):727–741, 1992.

[12] Shivkumar Chandrasekaran and Ilse CF Ipsen. On rank-revealing factorisations. *SIAM J. on Matrix Anal. and Apps.*, 15(2):592–622, 1994.

[13] H. Cheng, Z. Gimbutas, Per-Gunnar Martinsson, and V. Rokhlin. On the compression of low rank matrices. *SIAM Journal on Scientific Computing*, 26(4):1389–1404, 2005.

[14] Alexandra Chouldechova and Aaron Roth. A snapshot of the frontiers of fairness in machine learning. *Communications of the ACM*, 63:82–89, 2020.

[15] Ali Civril and Malik Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47-49):4801–4811, 2009.

[16] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[17] Sam Corbett-Davies and Sharad Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv:1808.00023*, 2018.

[18] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pages 2206–2216. PMLR, 2020.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[20] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, 2014.

[21] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[22] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[23] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*, 2020.

[24] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT⋆ '19, page 329–338. Association for Computing Machinery, 2019. ISBN 9781450361255.

[25] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[26] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Univ. Press, Baltimore, third edition, 1996.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[28] M. Gu and S. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996.

[29] Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension bounds for piecewise linear neural networks. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1064–1068. PMLR, 07–10 Jul 2017. URL http://proceedings.mlr.press/v65/harvey17a.html.

[30] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *CoRR*, abs/1808.06866, 2018. URL http://arxiv.org/abs/1808.06866.

[31] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Conference on Computer Vision and Patter Recognition*, 2019.

[32] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision*, 2017.

[33] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, and Song Li, Li-Jia nad Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision*, 2018.

[34] Kenneth L Ho and Lexing Ying. Hierarchical interpolative factorization for elliptic operators: differential equations. *Communications on Pure and Applied Mathematics*, 69(8):1415–1451, 2016.

[35] K.L. Ho and L. Greengard. A fast direct solver for structured linear systems by recursive skeletonization. *SIAM Journal on Scientific Computing*, 34(5):A2507–A2532, 2012. doi: 10.1137/120866683.

[36] K.L. Ho and L. Ying. Hierarchical interpolative factorization for elliptic operators: Integral equations. *Communications on Pure and Applied Mathematics*, 2015. ISSN 1097-0312. doi: 10.1002/cpa.21577. URL http://dx.doi.org/10.1002/cpa.21577.

[37] Yoo Pyo Hong and C-T Pan. Rank-revealing qr factorizations and the singular value decomposition. *Mathematics of Computation*, 58(197):213–232, 1992.

[38] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

[39] Zehao Huang and Naiyan Wan. Data-driven sparse structure selection for deep neural networks. In *European Conference on Computer Vision*, 2018.

[40] Yerlan Idelbayev and Miguel A. Carreira-Perpinan. Low-rank compression of neural nets: Learning the rank of each layer. In *2020 IEEE conference on computer vision and pattern recognition*, pages 8049–8059. IEEE, 2020.

[41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[42] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[43] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[44] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *International Conference on Learning Representations*, 2015.

[45] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H.S. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

[46] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

[47] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2007.

[48] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxkOlSYDH.

[49] Lucas Liebenwein, Cenk Baykal, Brandon Carter, David Gifford, and Daniela Rus. Lost in pruning: The effects of pruning neural networks beyond test accuracy. In *Proceedings of the 4th MLSys Conference*, 2021.

[50] Lucas Liebenwein, Alaa Maalouf, Dan Feldman, and Daniela Rus. Compressing neural networks: Towards determining the optimal layer-wise decomposition. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2107.11442.

[51] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1529–1538, 2020.

[52] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao haung, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *International Conference on Computer Vision*, 2017.

[53] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

[54] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[55] Alaa Maalouf, Harry Lang, Daniela Rus, and Dan Feldman. Deep learning meets projective clustering. *ArXiv*, abs/2010.04290, 2021.

[56] Michael W Mahoney and Petros Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[57] Per-Gunnar. Martinsson. *Fast Direct Solvers for Elliptic PDEs*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2019. doi: 10.1137/1.9781611976045. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611976045.

[58] Per-Gunnar Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. *J. Comput. Phys.*, 205(1):1–23, May 2005. ISSN 0021-9991. doi: 10.1016/j.jcp.2004.10.033. URL http://dx.doi.org/10.1016/j.jcp.2004.10.033.

[59] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30(1):47–68, 2011.

[60] Charles T. Marx, Flaviou du Pin Calmon, and Berk Ustum. Predicitive multiplicity in classification. In *International Conference on Machine Learning*. PMLR, 2020.

[61] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021. ISSN 0360-0300. doi: 10.1145/3457607. URL https://doi.org/10.1145/3457607.

[62] Victor Minden, Kenneth L Ho, Anil Damle, and Lexing Ying. A recursive skeletonization factorization based on strong admissibility. *Multiscale Modeling & Simulation*, 15(2):768–796, 2017.

[63] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT press, 2 edition, 2018.

[64] Ben Mussay, Margarita Osadchy, Vladimir Braverman, Samson Zhou, and Dan Feldman. Data-independent neural pruning via coresets. In *International Conference on Learning Representations*, 2020.

[65] Bo Peng, Wenming Tan, Zheyang Li, Shun Zhang, Di Xie, and Shiliang Pu. Extreme network compression via filter group approximation. In *European Conference on Computer Vision*, 2018.

[66] Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In *International Conference on Machine Learning*, 2019.

[67] Gregorio Quintana-Ortí, Xiaobai Sun, and Christian H. Bischof. A blas-3 version of the qr factorization with column pivoting. *SIAM J. on Sci. Comp.*, 19(5):1486–1494, 1998. doi: 10.1137/S1064827595296732.

[68] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. *ArXiv*, abs/2007.00753, 2020.

[69] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[70] 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.

[71] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, A. Alemi Alexander, and Andrew Gordon Wilson. Does knowledge distillation really work? In *Advances in neural information processing systems*, 2021.

[72] Raphael J. L. Townshend, Martin Vögele, Patricia Suriana, Alexander Derry, Alexander Powers, Yianni Laloudakis, Sidhika Balachandar, Brandon M. Anderson, Stephan Eismann, Risi Kondor, Russ B. Altman, and Ron O. Dror. ATOM3D: tasks on molecules in three dimensions. *CoRR*, abs/2012.04035, 2020. URL https://arxiv.org/abs/2012.04035.

[73] Joel A Tropp. Column subset selection, matrix factorization, and eigenvalue optimization. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 978–986. SIAM, 2009.

[74] Sergey Voronin and Per-Gunnar Martinsson. Efficient algorithms for cur and interpolative matrix decompositions. *Advances in Computational Mathematics*, 43(3):495–516, 2017.

[75] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.

[76] Huanrui Yang, Wei Wen, and Hai Li. Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. In *International Conference on Learning Representations*, 2020.

[77] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[78] Tao Zhuang, Zhixuan Zhang, Yuhen Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. In *Advances in neural information processing systems*, 2020.

[79] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu1, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in neural information processing systems*, 2018.

Appendix

A Notation

Matrices are denoted with capital letters such as A, and vectors with lower case a. In situations where we partition a matrix into pieces, the partitions will be referred to as A_{ij}. Individual entries in a matrix will be referred to as lower case letters with two subscripts, a_{ij}. $\sigma_k(A)$ denotes the k-th leading singular value of A, and $\kappa(A)$ the condition number. For a matrix $A \in \mathbb{R}^{n\times m}$ we let $A_{\mathcal{J},\mathcal{I}}$ denote a sub-selection of the matrix A using sets $\mathcal{J} \subset [n]$ to denote the selected rows and $\mathcal{I} \subset [m]$ to denote the selected columns; : denotes a selection of all rows or columns.

B Fixed-rank interpolative decompositions

As stated in Section 3, a formal algorithmic statement is given for computing fixed-rank interpolative decompositions.

Algorithm 2 Interpolative Decomposition

Input: matrix $A \in \mathbb{R}^{n\times m}$, rank-$k$
Output: interpolative decomposition $A_{:,\mathcal{I}}T$
Compute column-pivoted QR factorization

$$A\begin{bmatrix}\Pi_1 & \Pi_2\end{bmatrix} = \begin{bmatrix}Q_1 & Q_2\end{bmatrix}\begin{bmatrix}R_{11} & R_{12}\\ & R_{22}\end{bmatrix}.$$

where $\Pi_1 \in \mathbb{R}^{m\times k}$, $R_{11} \in \mathbb{R}^{k\times k}$, $R_{12} \in \mathbb{R}^{k\times(m-k)}$, and remaining dimensions as required.
$A_{:,\mathcal{I}} \leftarrow A\Pi_1$
$T \leftarrow \begin{bmatrix}I_k & R_{11}^{-1}R_{12}\end{bmatrix}\Pi^\top$

C Proofs

Theorem 4.1. *Consider a model $h_{FC} = u^\top g(W^\top x)$ with m hidden neurons and a pruned model $\widehat{h}_{FC} = \widehat{u}^\top g(\widehat{W}^\top x)$ constructed using an ϵ accurate ID with n data points drawn i.i.d from $\mathcal{D}$. The risk of the pruned model $\mathcal{R}_p$ on a data set $(x, y) \sim D$ assuming $\mathcal{D}$ is compactly supported on $\Omega_x \times \Omega$ is bounded by*

$$\mathcal{R}_p \leq \mathcal{R}_{ID} + \mathcal{R}_0 + 2\sqrt{\mathcal{R}_{ID}\mathcal{R}_0},$$

where $\mathcal{R}_{ID}$ is the risk associated with approximating the full model by a pruned one and with probability $1-\delta$ satisfies

$$\mathcal{R}_{ID} \leq \epsilon^2 M + M(1+\|T\|_2)^2 n^{-\frac{1}{2}}\left(\sqrt{2\zeta dm\log(dm)\log\frac{en}{\zeta dm\log(dm)}} + \sqrt{\frac{\log(1/\delta)}{2}}\right).$$

Here, $M = \sup_{x\in\Omega_x}\|u\|_2^2\|g(W^T x)\|_2^2$ and ζ is a universal constant that depends on g.

Proof. We can write the risk for this network as

$$\mathcal{R}_p = \mathbb{E}(\|\widehat{u}^\top g(\widehat{W}^\top x) - y\|^2),$$

and adding and subtract the original network yields

$$\begin{aligned}
\mathbb{E}(\|\widehat{u}^\top g(\widehat{W}^\top x) - y\|^2) &= \mathbb{E}(\|(\widehat{u}^\top g(\widehat{W}^\top x) - u^\top g(w^\top x)) + (u^\top g(w^\top x) - y)\|^2)\\
&\leq \mathbb{E}((\|(\widehat{u}^\top g(\widehat{W}^\top x) - u^\top g(w^\top x))\| + \|(u^\top g(w^\top x) - y)\|)^2)\\
&\leq \mathbb{E}(\|(\widehat{u}^\top g(\widehat{W}^\top x) - u^\top g(w^\top x))\|^2)\\
&\quad + \mathbb{E}(2\|(\widehat{u}^\top g(\widehat{W}^\top x) - u^\top g(w^\top x))\|\|(u^\top g(w^\top x) - y)\|)\\
&\quad + \mathbb{E}(\|(u^\top g(w^\top x) - y)\|^2)\\
&\leq \mathcal{R}_{ID} + 2\sqrt{\mathcal{R}_{ID}\mathcal{R}_0} + \mathcal{R}_0.
\end{aligned}$$

Now, we bound $\mathcal{R}_{ID}$ by considering the interpolative decomposition to be a learning algorithm learning the function $u^\top g(W^\top X)$. Specifically, we use Lemma 4.2 to bound $\mathcal{R}_{ID}$ as

$$\mathcal{R}_{ID} \leq \widehat{\mathcal{R}}_{ID} + M(1+\|T\|_2)^2 n^{-\frac{1}{2}} \left(\sqrt{2p\log(en/p)} + 2^{-\frac{1}{2}}\sqrt{\log(1/\delta)}\right).$$

where p is the pseudo-dimension. We can then use Lemma 4.4 to bound the empirical risk of the interpolative decomposition as

$$\widehat{\mathcal{R}}_{ID} \leq \epsilon^2 \|u\|_2^2 \|g(W^T X)\|_2^2/n,$$

and it follows that

$$\widehat{\mathcal{R}}_{ID} \leq \epsilon^2 \sup_{x\in\Omega_x} \|u\|_2^2 \|g(W^T x)\|_2^2.$$

□

Lemma 4.2. *Under the assumptions of Theorem 4.1, for any $\delta \in (0,1)$, $\mathcal{R}_{ID}$ satisfies*

$$\mathcal{R}_{ID} \leq \widehat{\mathcal{R}}_{ID} + M(1+\|T\|_2)^2 n^{-\frac{1}{2}} \left(\sqrt{2p\log(en/p)} + 2^{-\frac{1}{2}}\sqrt{\log(1/\delta)}\right)$$

with probability $1-\delta$, where $M = \sup_{x\in\Omega_x} \|u\|^2\|g(W^T x)\|^2$ and $p = \zeta dm\log(dm)$ for some universal constant ζ that depends only on the activation function.

Proof. Considering the interpolative decomposition as a learning algorithm to learn $u^\top g(W^\top X)$, we can use Theorem 11.8 in [63] to bound the risk on the data distribution. Given a maximum on the loss function η, and the ReLU activation function,

$$\mathcal{R}_{ID} \leq \widehat{\mathcal{R}}_{ID} + \frac{\eta}{n^{1/2}}(\sqrt{2p\log en/p} + 2^{-1/2}\sqrt{\log(1/\delta)})$$

with probability $(1-\delta)$.[6] Here, the constant η is bounded by Lemma C.1. Bartlett et al. [29] show that the p-dimension for a ReLU network is $O(\mathcal{W}Llog(\mathcal{W}))$ where $\mathcal{W}$ is the number of weights and L is the number of layers. Here, that translates to $p = \zeta dm\log(dm)$ for some constant ζ that depends only on the choice of activation function. □

Lemma 4.4. *Following the notation of Theorem 4.1, an ID pruning to accuracy ϵ yields a compressed network that satisfies*

$$\widehat{\mathcal{R}}_{ID} \leq \epsilon^2 \|u\|_2^2 \|g(W^T X)\|_2^2/n,$$

where $X \in \mathbb{R}^{d\times n}$ is a matrix whose columns are the pruning data.

Proof.

$$\hat{\mathcal{R}}_{ID} = \frac{1}{n}\sum_{i=1}^{n} |u^\top g(W^\top x_i) - \widehat{u}^\top g(\widehat{W}^\top x_i)|^2 \tag{5}$$

Here, we can appeal to our definition of the ID to bound each term in the sum.

$$|u^\top g(W^\top x_i) - \widehat{u}^\top g(\widehat{W}^\top x_i)| = |u^\top g(W^\top x_i) - u^\top T^\top g(P^\top W^\top x_i)| \tag{6}$$

By our definition of an ϵ-accurate interpolative decomposition,

$$|u^\top g(W^\top x_i) - \widehat{u}^\top g(\widehat{W}^\top x_i)| \leq \epsilon\|u\|\|g(W^T x_i)\| \tag{7}$$

Therefore,

$$\hat{\mathcal{R}}_{ID} \leq \frac{1}{n}\epsilon^2\|u\|^2\|g(W^T X)\|^2 \tag{8}$$

□

[6] e is the base of the natural log.

Lemma C.1. *The maximum η of the loss function associated with approximating the full network with the pruned on is bounded as*

$$\eta \leq \sup_{x \in \Omega_x} \|u\|_2^2 \|g(W^T x)\|_2^2 (1 + \|T\|))^2$$

Proof.

$$\eta = \max_{x,W,u} \|u^\top g(W^\top x) - \widehat{u}^\top g(\widehat{W}^\top x)\|^2$$

For any $x \in \Omega_x$ we have the bound

$$\begin{aligned}
\|u^\top g(W^\top x) - \widehat{u}^\top g(\widehat{W}^\top x)\|^2 &\leq (\|u^\top g(W^\top x)\| + \|\widehat{u}^\top g(\widehat{W}^\top x)\|)^2 \\
&\leq (\|u^\top g(W^\top x)\| + \|u^\top T^\top g(P^\top W^\top x)\|)^2 \\
&\leq (\|u^\top g(W^\top x)\|(1 + \|T\|))^2.
\end{aligned}$$

Therefore,

$$\eta \leq \sup_{x \in \Omega_x} \|u\|_2^2 \|g(W^T x)\|_2^2 (1 + \|T\|))^2$$

□

Remarks C.2. We can explicitly measure the norm of the interpolation matrix T that appears in the upper bound of Lemma 4.2. Moreover, we expect this to be small because there exists an interpolation matrix such that $t_{ij} \leq 2 \forall \{i, j\}$ [47]. The better the interpolation matrix, the better the bound.

D Correlation between random trials

We introduce a metric that we call "model correlation" in order to evaluate different pruning methods. We define the correlation between two models on a data set as the percent of labels that the two models agree on, irrespective of if those labels agree with the ground truth. There are situations in which the user of a network may care about more than just the simple accuracy — it may matter which items a network is most likely to get wrong, and how. It is also possible for two networks to have the same accuracy but perform very differently on subsets of the dataset. We include this metric after fine tuning to demonstrate the efficacy of our compression method relative to methods that necessitate extensive fine tuning and, therefore, may not correlate well with the original model. Here we provide some baseline measurements of what affects model correlation, in order to better understand this metric.

Our baseline measurement of model correlation is the model correlation between two same-sized but randomly initialized networks trained using the same hyper parameters but with different (random) data orders. We first test the effects for a fully connected one hidden layer network on FashionMNIST.

Size	Same Initialization	Same Data Order	Vary Both	Accuracy
500	94.6	94.0	93.3	89.21
2000	96.7	94.7	94.5	89.63
4000	97.3	95.4	95.1	89.74

Table 3: Correlation data for a single hidden layer fully connected network on the FashionMNIST data set. We keep the same initialization but vary the data order (Same Initialization), use the same data order but vary the initialization (Same Data Order) or vary both the data order and initialization (Vary both). This data is averaged over 9 trials. We see that starting at the same initialization increases the correlation at the end of training, and, interestingly, that using the same data order during training can increase the correlation as well.

We continue our experiments on the CIFAR10 dataset using the VGG-16 architecture. We find that two differently randomly initialized models trained using different data orders to the same state-of-the-art accuracy (93.6%) agree on classifications 93.0% of the time. The effect of data order is also seen on CIFAR10 VGG-16 networks. When we prune to 50% FLOPS and then re-train a VGG-16 network using magnitude pruning, if the data order is the same as the original network, then the correlation is 94.01%. However, using a different data order the correlation is 93.15%. The model correlation breaks down quickly when we use a large learning rate (.1 for 200 epochs).

E Comparison methods

Here we provide a key for the various methods we compare to in the main text, along with their classifications within our taxonomy. We give both the paper citation and implementation citation.

Dense	**Structure Preserving**	**Corrects Next Layer**	**No Local FT**	
Name	Citation	Implementation	Table	Figures
ID	(Ours)	(Ours)	All	All
PFP	Liebenwein et al. [48]	Liebenwein et al. [48]	-	2
AMC	He et al. [33]	He et al. [33]	2	-

Dense	**Structure Preserving**	**Corrects Next Layer**	**Local FT**	
Name	Citation	Implementation	Table	Figures
Thi	Luo et al. [54]	Liebenwein et al. [48]	2	2
CP	He et al. [32]	He et al. [32]	2	-
NS	Liu et al. [52]	Zhuang et al. [78]	1	-

Dense	**Structure Preserving**	**No Correction**		
Name	Citation	Implementation	Table	Figures
Uni	Uniform Random Filter Pruning	Liebenwein et al. [48]	-	2
Soft	He et al. [30]	Liebenwein et al. [48]	-	2
StructMag	Li et al. [46]	Liebenwein et al. [48]	1	2
FPGM	He et al. [31]	(Ours)	1	-
HRank	Lin et al. [51]	Lin et al. [51]	1	-

Dense	**Extra Layers**			
Name	Citation	Implementation	Table	Figures
ALDS	Liebenwein et al. [50]	Liebenwein et al. [50]	-	1, 3
Messi	Maalouf et al. [55]	Liebenwein et al. [50]	-	1, 2
PCA	Zhang et al. [77]	Liebenwein et al. [50]	-	1, 3
SVD	Denton et al. [20]	Liebenwein et al. [50]	-	1, 2
LRank	Idelbayev and Carreira-Perpinan [40]	Liebenwein et al. [50]	1, 2	3
Polar	Zhuang et al. [78]	Zhuang et al. [78]	1	-

Sparse				
Name	Citation	Implementation	Table	Figures
SiPP	Baykal et al. [5]	Baykal et al. [5]	-	2
Snip	Lee et al. [45]	Baykal et al. [5]	-	2
Thres	Li et al. [46]	Liebenwein et al. [48]	-	2

Table 4: Taxonomy of pruning and methods and look-up table for references. As you go further down the list, methods tend to become more different from our own.

Algorithm 3 Pruning a multilayer network with iterative interpolative decompositions

Input: Neural net $h(x; W^{(1)}, \ldots, W^{(L)})$, pruning set X, step size λ, FLOPs ratio ρ
Output: Pruned network $h(x; \widehat{W}^{(1)}, \ldots, \widehat{W}^{(L)})$

for $l \in \{1, \ldots, L\}$ **do**
  $\widehat{W}^{(l)} \leftarrow W^{(l)}$
end for
$F \leftarrow \text{Compute\_FLOPs}(h(x; \widehat{W}^{(1)}, \ldots, \widehat{W}^{(L)})$
$F_\rho \leftarrow F * \rho$
while $F > F_\rho$ **do**
  $S, K \leftarrow \{\}$
  for $l \in \{1, \ldots, L\}$ **do**
    $Z \leftarrow h_{1:l}(X; W^{(1)}, \ldots, W^{(l)})$ {layer l activations}
    $R \leftarrow \text{Pivot\_QR}(\text{Reshape}(Z))$ {reshape if Conv layer}
    $k \leftarrow \text{num\_channel}(\widehat{W}^{(l)} \times \lambda$ {calculates proportion of channels or neurons to potentially remove}
    $Err \leftarrow |R[k+1, k+1]/R[1, 1]|$ {error from ID approximation}
    $F_l \leftarrow \text{Compute\_FLOPs}(\widehat{W}^{(l)}, \widehat{W}^{(l+1)})$ {compute FLOPs of current and next layer}
    $S.\text{append}(Err/F_l)$ {weighted layer prunability score}
    $K.\text{append}(k)$
  end for
  $l \leftarrow \text{argmin}(S)$
  $\widehat{W}^{(l)} \leftarrow$ ID prune layer l to $K[l]$ neurons or channels
  $F \leftarrow \text{Compute\_FLOPs}(h(x; \widehat{W}^{(1)}, \ldots, \widehat{W}^{(L)})$
end while

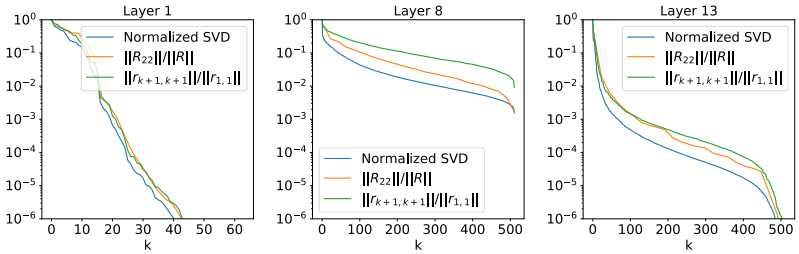


Figure 4: Metrics for different layers in VGG-16 for Cifar-10. The leftmost layer is the first convolutional layer. The center figure is one of the middle convolutional layers, and the rightmost figure is the last convolutional layer. As we can see, the singular value decay varies throughout the network.

F Setting $k(\epsilon)$ per layer in deep networks with iterative pruning

By Definition 3.1, an ϵ-accurate interpolative decomposition is associated with a number k of selected columns. We observe that for deep networks it is not straightforward to apply a single accuracy ϵ to the entire network. Figure 4 illustrates the representative variety in the layer-wise singular value decay for a trained VGG-16 model. For our method a sharper singular value decay indicates greater prunability. However, not all layers contribute equally to the number of FLOPS and the number of parameters. Typically convolutional layers contribute disproportionately to the number of FLOPs compared to the number of parameters they contain. When we prune neurons or channels in a layer, that has an effect on the number of FLOPS performed by that layer, and also in the next layer. Therefore, we iteratively prune the network by finding the layer that will allow us to prune the most FLOPs compared to the error that we expect from pruning that layer, and pruning that layer. This is given in Algorithm 3.

G Sensitivity of parameters

When we prune using Iterative ID, we have a choice of hyperparameter in how what percent of channels to cut per iteration (pruning fraction α in Algorithm 3). In Figure 5, we demonstrate that the

choice of pruning fraction does not greatly affect the accuracy of the network after iterative pruning. However, using a smaller α results in the network taking significantly more time to prune.

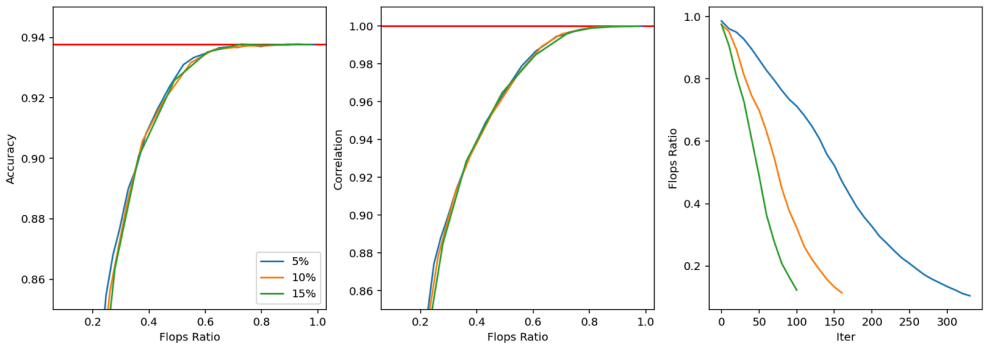


Figure 5: Iterative pruning performed at different amounts of channels/neurons cut per iteration in the chosen layer. We see that the choice of number of channels per iteration does not have a large impact on the outcome, though smaller percents take a longer time.

H Additional experimental details and results

H.1 Illustrative example

We created a simple synthetic data set for which we know the form of a relatively minimal model representation.

Draw n points iid. from the unit circle in 2 dimensions. Next select 2 (normalized) random vectors v_1 and v_2. All labels are initialized to zero, and add or subtract 1 to the label for each time it produces a positive inner product with one of the random vectors. With the ReLU activation function it is possible to correctly label all points with a one hidden layer fully connected network of width 4. Construct pairs of neurons, with each pair aligned with the center of one of the random vectors. We can think of the pair as creating a flat function by using one neuron to "cut the top" off of the round part of the function created by the other.

We parameterize this with an angle ϕ away from the pair. If each neuron in the pair has the same weight magnitude w, coefficient $\pm u$ and a bias $b \pm \delta/2$, then given a particular w, b and δ, we can write:

$$u(\mathrm{ReLU}(w\cos\phi - b + \delta/2) - \mathrm{ReLU}(w\cos\phi - (b - \delta/2)))$$

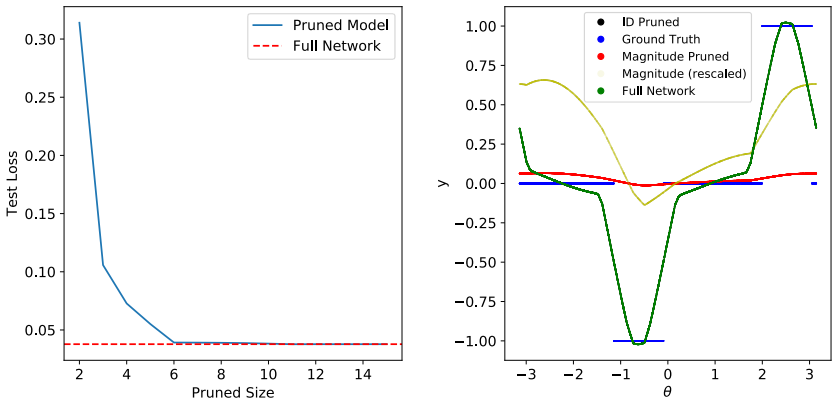


Figure 6: A plot of loss v.s. number of neurons kept (left) and a plot of the function evaluation for the full network, ID pruning, and magnitude pruned model (right). The data is drawn from the unit circle, and we parameterize X in terms of an angle θ. To see more detail in the magnitude pruned function, we draw it with a scale of 10x applied uniformly. We see that it takes very few (12) neurons to completely represent the function approximated by the network. In fact, the ID pruned function is visually indistinguishable from the full model in this case.

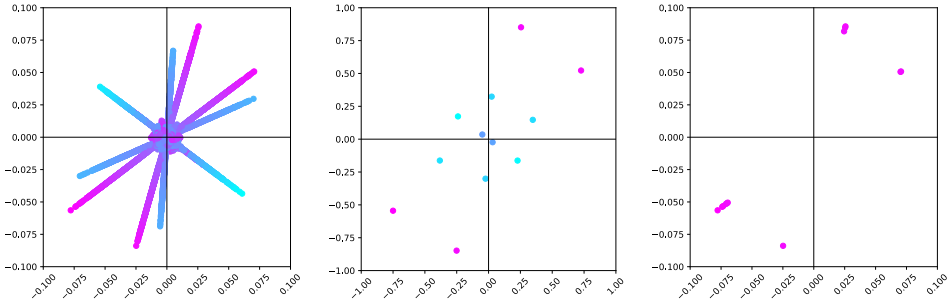


Figure 7: Neuron visualization for a simple 2-d regression task. The axes are the weights of the neurons, and the color is the bias. We have the weights trained for a full model(left) with 5000 hidden nodes. The 12 neurons kept by the ID (center) represent the function well. These are re-scaled by the magnitude of their coefficients in the second layer to display the effect of the ID. Magnitude pruning (right) keeps duplicate neurons that do not represent the function. The test loss is 0.037786 for the full model, 0.037781 for the model pruned with ID, and 0.33 magnitude-pruned model.

As long as $w > b$, this looks like a step function, where the sides get steeper as w approaches infinity. This allows us to perfectly label all of the points given twice the number of neurons as we have random vectors v_1 and v_2, which is much smaller than the number of data points n.

We train an over-parameterized single hidden layer network to perform fairly well on this task, using an initialization scale that is common in some machine learning platforms such as TensorFlow [1]. This is shown in figure 6. However, magnitude pruning will not necessarily recognize the structure of the minimum representative network because both of the neurons in the pair construction may not have a large magnitude. On this example we see that the interpolative decomposition is able to select neurons which resemble a close to minimal representative network.

In Figure 7 we see that the ID well represents the original network by ignoring duplicate neurons and taking into account differences in the bias. The ID even achieves slightly better test loss than the original model. Magnitude pruning does not approximate the original model well; it keeps duplicate neurons and fails to find important information with the same number of neurons.

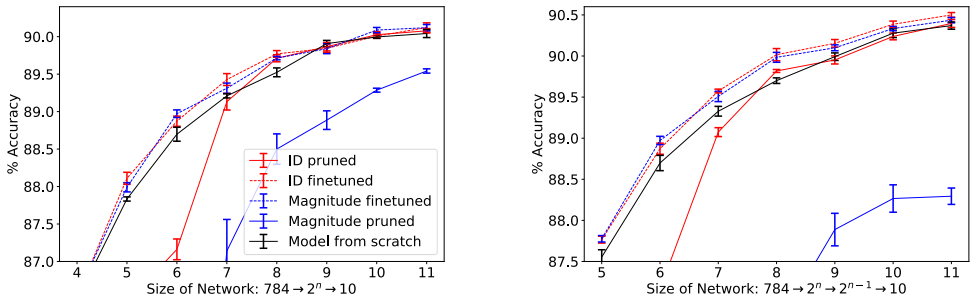


Figure 8: Accuracy for a one hidden layer (left) and two hidden layer (right) fully connected neural networks of varying size on Fashion-MNIST for ID and magnitude pruning, as well as training directly. These curves are averaged over 5 trials; error bars report uncertainty in the mean.

H.2 Fashion MNIST

For one and two hidden layer networks on Fashion MNIST [75] we compare the performance of ID and magnitude pruning to networks of the same size trained from scratch. Each pruning method is used to prune to half the number of neurons of the original model, and is then fine-tuned for 10 epochs. We use a stochastic gradient descent optimizer with a learning rate of 0.3 which decays by a factor of 0.9 for each epoch to train the initial networks. Each was trained for 50 epochs. Our pruning set was 10000 images, which did not need to be held out from training for the simple data set. The fine tuning ran for 10 epochs with an initial learning rate of 0.1 with a decay rate of 0.6 for two layer networks, and 0.2 with a learning rate decay of 0.7 for one layer. Larger ID-pruned models (greater than 512 neurons) can be fine-tuned with a much smaller learning rate of 0.002. These learning rates and number of epochs may not be optimal but were determined through brief empirical tests. We used 5 random seeds for each size of model. The error bars are reported as the uncertainty in the mean, defined in terms of the standard deviation σ, and number of independent trials N, $\sigma_{mean} = \sigma/\sqrt{N}$. Results are shown in Figure 8; before fine-tuning the ID achieves a significantly higher test accuracy than magnitude pruning. ID pruning with fine-tuning outperforms training from scratch. At sufficiently large network sizes, ID pruning alone performs similarly to training a network from scratch. When we start to see diminishing returns from adding more neurons, the performance of the various methods begin to converge.

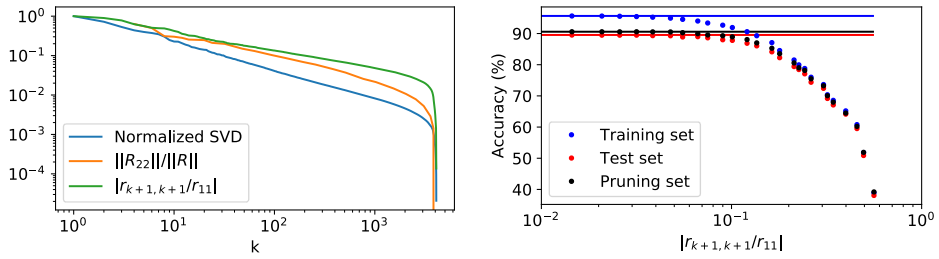
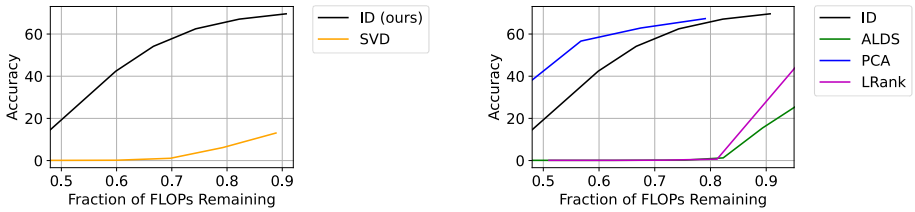


Figure 9: Left: Normalized singular value decay, $\|R_{22}\|_2/\|R\|_2$, and $|r_{k+1,k+1}/r_{1,1}|$ for a matrix from (4)) from a one hidden layer network (i.e., $g(W^\top X)$) trained on Fashion MNIST. We see that the metrics generally correlate well in this setting. Right: Accuracy of a single hidden layer network pruned from width 4096 to k. The horizontal lines are the accuracy of the full sized network. The difference between pruning and test accuracy is due to a slight class imbalance in the canonical test set.

H.3 Additional ImageNet experiments on MobileNet V1

Here we give additional pruning experiments on Mobilenet V1 for ImageNet. We see that the ID is competitive against compression methods which do not do any local fine tuning (Figure 10a,



(a) Compare with methods that do not use any fine tuning. (b) Comparing with methods that use a local fine tuning correction.

Figure 10: MobileNet V1 compression results on ImageNet. Note that the matrix methods often work by changing the depth of convolution in the network, and given that MobileNet uses depth-wise separable convolutions, it is not surprising that matrix methods would perform poorly.

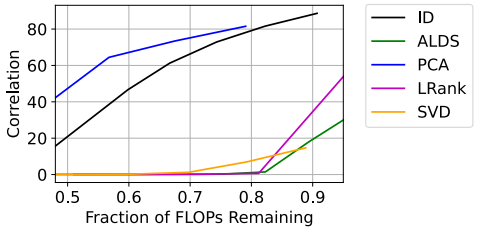
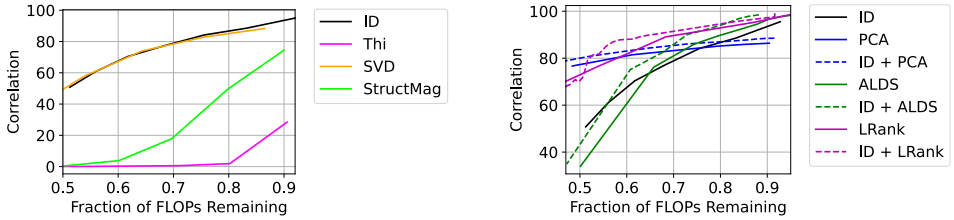


Figure 11: MobileNet correlation results compared against various compression methods. Note that several other compression methods work by doing decompositions on the weight matrix and changing the number of groups in the convolution. Due to the MobileNet architecture, this may result in a very low accuracy for methods that work well on other architectures.

Figure 10b includes methods which do use local fine tuning; ID performs better than two out of the three. Interestingly, we see that although the PCA and LRank methods performed well on an overparameterized network such as VGG-16, they do not work well on a much more efficient network.

H.4 Imagenet pre fine tuning correlation

Here we report correlation results om ImageNet with the MobileNet V1 and VGG-16 models. For methods which do not incorporate a fine tuning correction (Figure 12a), we see that the ID proves as good as any other method we compare to. Figure 12b compares to methods which do incorporate a local fine tuning correction. Similar to accuracy, we see that composing ID with another compression method improves upon either method.



(a) Comparing with methods that do not use a local fine tuning correction. (b) Comparing with methods that use a fine tuning correction.

Figure 12: VGG-16 correlation results on ImageNet

Prune set size	Pre-fine-tuning accuracy
5k	68.03
10k	68.06

Table 5: Pre-fine-tuning accuracy compared to prune set size for VGG16 model pruned to 25% FLOPs reduction on ImageNet.

H.5 Hyper-parameter details

CIFAR-10 The VGG-16 models are trained with 5 random seeds and with hyper-parameter specifications and code provided by Liu et al. [53]. The test set is randomly partitioned into a prune set and new test set. The Iterative ID used a held out set of size 1000. For the iterative ID on VGG-16, we prune 10% of a layer per iteration. For VGG-16 fine tuning we use SGD with initial learning rate of 5e-3, reduced to 2.5e-3 after 20 epochs and again reduced to 1e-3 after a another 20 epochs have passed. We use a batch size of 128, momentum of 0.9, and weight decay of 5e-4.

The full-size Mobilenet V1 network for Cifar-10 was trained using the ADAM optimizer for 120 epochs, using the default parameters of lr=.001, betas= (.9, .999). The test set is randomly partitioned into a prune set and new test set.

ImageNet For VGG-16 the iterative ID algorithm uses a randomly held out set of 5000 images with a stepsize parameter of 5%. For fine-tuning, we use SGD with a learning rate of 1e-7, batch size of 256, momentum of 0.9, and weight decay of 1e-4. For ID+PCA, we switched to learning rate 1e-8 at 75 epochs.

For Mobilenet V1 we prune to a constant fraction using the ID with a randomly sampled held out set of 1000 images.

H.6 Estimating compute

The experiments on the illustrative example and Fashion MNIST were performed on a 2019 iMac running an Intel I9. The illustrative example computes in minutes. We trained 15 different model configurations, 8 one hidden layer, and 7 two hidden layer network sizes. For 5 random seeds, we used $5 * (50 + 10 + 10) = 350$ epochs per model size, and trained 15 different model configurations. We used 10,000 images for the pruning set, however, the runtime for computing the ID is cheap compared to training, using the scipy QR decomposition function which calls a LAPACK subroutine [2].

The experiments on CIFAR-10 were performed on a NVIDIA GeForce GTX 1080Ti on a university compute cluster. Epochs of fine tuning are specified in the tables of the main paper. Computing the interpolative decomposition were comparatively cheap, only requiring 1000 data points. Computing costs for any compression method were negligible compared to fine tuning.

ImageNet experiments were conducted on a university compute cluster. For fine tuning NVIDIA GeForce RTX 3090, RTX A6000, or TITAN RTX were used. The compression portion was conducted on a NIVIDIA GeForce GTX 1080Ti or FTX 2080Ti. Epochs of fine tuning are specified in the tables of the main paper. The compute cost of the compression methods before fine tuning (including interpolative decomposition) were trivial compared to any amount of global fine tuning.

I Sensitivity to pruning set

I.1 Pruning set size

We compare the pre-fine-tuning accuracy as a function of the pruning set size. Here we show that the accuracy is not particularly sensitive to the pruning set size. Table 5 shows on ImageNet that our ID-based pruning method is robust to the prune set size, and in fact quite efficient. Figure 13 shows on CIFAR-10 that this trend persists across a wide range of compression levels. Note that the number of pruning examples must be at least the number of neurons or channels that we prune to for each layer.

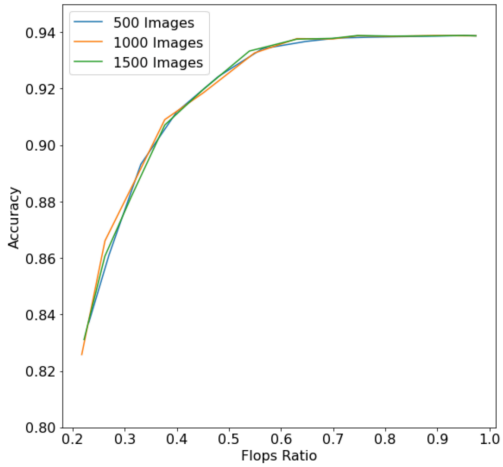


Figure 13: Comparing the pre-fine-tuning accuracy and FLOP reduction for different pruning set sizes on VGG-16 Cifar-10 model. We see that above the minimum threshold, pruning set size has a minimial impact on the accuracy of the pruned model.

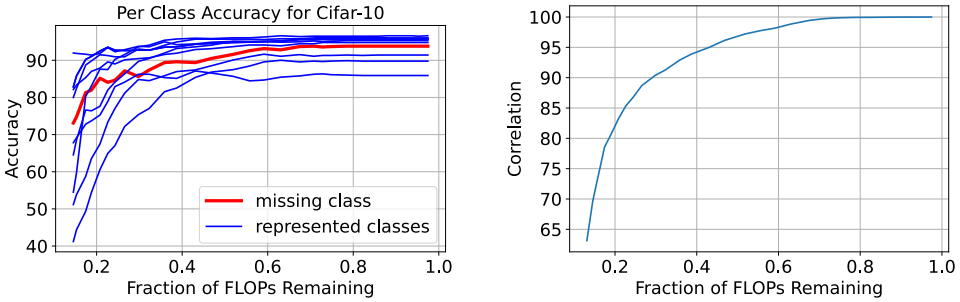
I.2 Pruning set contents

We want our method to be robust to the data selection method used to generate our pruning set. We test our method to this specific sensitivity by removing an entire class from the pruning set.

We begin with a full-sized VGG-16 CIFAR-10 model that was trained on all 10 classes. Then we draw a pruning set from only 9 of the classes, completely leaving out images from one of the classes. We prune to 50% of the original FLOPs, using Iterative ID, and compare the per-class test accuracies for each of the 10 classes. Figure 14a shows that the ID maintains good accuracy even on images from the class that was excluded from the pruning set.

We expect that other methods which preserve the model's decision boundaries (and therefore correlation) will likely show similar results. However, methods which require extensive fine tuning and effectively re-train the network will likely not recover accuracy on the missing class. To demonstrate this, we prune the same full-sized network using magnitude pruning using the same pruning set to 50% FLOPs reduction, and then fine tune with only 9 classes in the fine tuning set. As shown in Figure 15a, the accuracy for the other 9 classes recovers, however, the accuracy for the class that was removed does not.

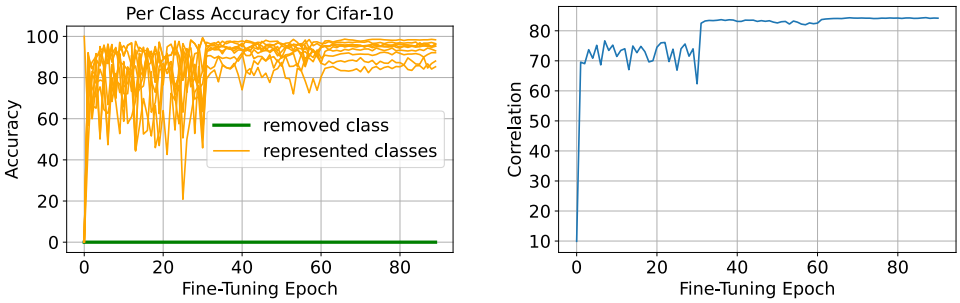
This experiment suggests that pruning methods which maintain properties of the original model may potentially be able to maintain fairness (i.e. per-class accuracy) even when some classes of data are under-represented. Our compression method was able to reasonably preserve per-class accuracies (our measure of fairness), even without access to data from one of those classes.



(a) Per class accuracies while pruning a VGG-16 model using only data from 9 classes.

(b) Model Correlation when ID pruning using data from only 9 classes

Figure 14: Per-class accuracies as we prune a VGG-16 model on CIFAR-10 with only access to data from 9/10 classes. No fine tuning was done. We see that ID maintains reasonable accuracy on all (including the unrepresented) classes, and stays correlated with the original model.



(a) Per class accuracies while fine tuning a magnitude-pruned model with only 9 classes

(b) Model Correlation while fine-tuning a magnitude pruned model using data from only 9 classes.

Figure 15: Comparing class accuracies for a 50% FLOPS VGG-16 model pruned using magnitude pruning, and then fine-tuned using only 9 out of 10 classes of CIFAR-10. We see that the model recovers on the represented classes, but not on the unrepresented class.