Dataset Pruning Using Early Exit Networks

Alperen Görmez [1] **Erdem Koyuncu** [1]

Abstract

We present EEPrune, a novel dataset pruning algorithm that leverages early exit networks during training. EEPrune utilizes the innate ability of early exit networks to assess the difficulty of individual samples and applies different criteria to decide whether to prune them. Specifically, for a training sample to be discarded, the confidence level of the model at the early exit should be above a certain threshold, along with a correct classification at both the early exit and final layers. We describe several other variants of our EEPrune algorithm. Extensive experiments on CIFAR-10, CIFAR-100 and Tiny Imagenet datasets demonstrate that EEPrune and its variations consistently outperform other dataset pruning methods.

1. Introduction

Sutton's "bitter lesson" states (Sutton, 2019; Sevilla et al., 2022) that "general methods that leverage computation are ultimately the most effective, and by a large margin." Larger models trained on larger datasets with more compute still seem to be an important research direction due to major outcomes and great performances (Brown et al., 2020; OpenAI, 2023; Thoppilan et al., 2022; Chowdhery et al., 2022; Zhang et al., 2022; Dehghani et al., 2023; Schuhmann et al., 2022). However, creating larger models and training on larger datasets for longer, along with the expenses associated with retraining and hyperparameter tuning only add to the already high training costs. These ever-increasing costs may not sustainable in the long run and are thus a significant problem that need to be addressed.

Dataset pruning methods have emerged as a potential solution to address the costs of training deep learning models. By discarding redundant samples and keeping, for example, only the difficult samples for training, the model performance on the test set stays the same even when the model is trained on the pruned dataset (Paul et al., 2021; Toneva et al., 2018; Coleman et al., 2019; Nohyun et al., 2023; Sorscher et al., 2022b). While dataset pruning methods strive to decrease training expenses, they typically train an ensemble of models or train a single model fully (Paul et al., 2021; Toneva et al., 2018; Coleman et al., 2019). This is particularly undesirable for resource-constrained devices due to the large memory footprint of the ensemble and high computational demands of full training.

A different class of techniques to reduce computational complexity is conditional computation through early exit (Panda et al., 2016; Teerapittayanon et al., 2016; Kaya et al., 2019; Görmez et al., 2022; Huang et al., 2017; Yang et al., 2020; Han et al., 2021), which allows one to exploit the heterogeneous nature of real-world data (Bolukbasi et al., 2017). Specifically, an early exit model introduces multiple exit points or intermediate classifiers to an ordinary neural network, called the "base" network. An input that is confidently classified at an intermediate exit point may then "exit' early without the need for traversing the remaining layers, thereby reducing the overall computation required for inference. Early exit networks can also provide reduced training costs by allowing the end user to train only the attached exits while keeping the base model frozen, which is particularly useful for edge devices (Scardapane et al., 2020; BAC, 2020).

Earlier works thus demonstrate that utilizing not only the final outputs but also the intermediate representations can greatly reduce the overall inference or training costs of a network. However, to the best of our knowledge, this observation has not yet been exploited for dataset pruning, which becomes the focus of the present work. In fact, existing work on dataset pruning uses only the features of the final layer. We show that incorporating intermediate features in the pruning decision process can greatly improve the training performance, measured in terms of the floating point operations (FLOPs) to reach a certain level of accuracy.

The rest of this paper is organized as follows: In Section 2, we present the related work. In Section 3, we present our new dataset pruning scheme that we call EEPrune. In Section 4, we provide numerical results. Finally, in Section 5, we draw our main conclusions. Extended discussions and more experiments are provided in the appendices.

[1]Department of Electrical and Computer Engineering, University of Illinois Chicago, USA. Correspondence to: Alperen Görmez <agorme2@uic.edu>.

In *ICML Workshop on Localized Learning (LLW)*, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

2. Related Work

Our research integrates two methods that are commonly employed to lower the inference and training costs of deep learning models: early exit networks and dataset pruning.

2.1. Early Exit Networks

The first idea of adding early exits to neural networks goes back to the GoogleNet (Szegedy et al., 2015). The primary purpose of adding the early exits was to solve the vanishing gradient problem. To reduce the inference cost, researchers have incorporated an exit criterion based on the entropy or the confidence level of the output of the early exit layer (Panda et al., 2016; Teerapittayanon et al., 2016; Kaya et al., 2019; Görmez et al., 2022; Huang et al., 2017; Yang et al., 2020; Han et al., 2021). Samples that meet the exit criterion are considered easy and prediction is made at the early exit layer, resulting in a faster computation. More complex samples that do not meet the exit criterion are forwarded to the next layers for further processing. Early exit layers can be trained jointly with the base model or separately (Scardapane et al., 2020), but approaches that do not require any training also exist (Görmez et al., 2022). These make early exit networks suitable for performing training and inference for localized learning (BAC, 2020).

2.2. Dataset Pruning

Dataset pruning methods aim to select the most representative samples from a dataset and remove the rest in order to reduce the training cost (Feldman & Langberg, 2011; Agarwal et al., 2005; Sorscher et al., 2022a). This process can be accomplished through the use of scoring techniques, which evaluate the importance or difficulty of each sample in the dataset. Various scoring methods have been proposed in the literature. We summarize them in the Appendix.

3. The EEPrune Scheme

We now introduce our EEPrune scheme. Let $(x_0^{(i)}, y^{(i)}) \in D$ be a training sample-label pair from the dataset D where there are N samples and C classes, $i \in \{1, 2, \ldots, N\}$ and $y^{(i)}$ is one-hot-encoded vector of length C. Let F be a network with one early exit, as shown in Figure 1. The network input-output relationships can be expressed as

$$\begin{aligned} x_j^{(i)} &= l_j(x_{j-1}^{(i)}), j = 1, \ldots, L, \\ \hat{y}_{ee}^{(i)} &= g(x_k^{(i)}), \quad \hat{y}_{fe}^{(i)} = h(x_L^{(i)}), \end{aligned} \tag{1}$$

where l_j are layers of the base model, k is the early exit point, g is the linear layer at the early exit, $\hat{y}_{ee}^{(i)}$ is the early exit output, h is the linear layer at the final exit, and $\hat{y}_{fe}^{(i)}$ is the final exit output.

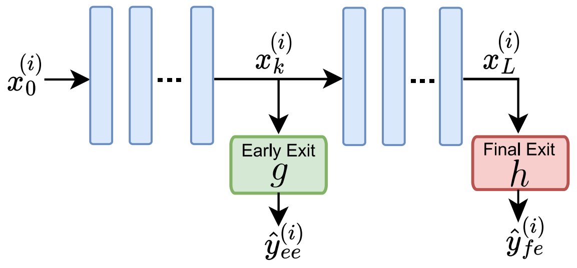


Figure 1. The computation graph of EEPrune.

Our ultimate goal is to reduce the training cost while keeping the model's performance on the test set as high as possible. We achieve this by reducing the size of the training set. More specifically, while training our early exit network on the dataset for a short time, we discard training samples from our dataset if **1)** the early exit can predict the correct class correctly, **2)** and the final exit can predict the correct class correctly, **3)** and the maximum prediction probability given by the early exit is greater than a specified threshold. In practice, we flag samples for pruning one by one over multiple epochs of learning. After E epochs, where E is a user-defined parameter, the learning process is stopped and all flagged samples can be removed from the dataset. We can then train a new model over the pruned dataset.

As can be seen, EEPrune verifies three conditions before flagging an input sample for pruning. The first condition ensures that the training sample of interest is easy enough to be correctly classified by using the features of a layer at the middle (or even an earlier point) of the network. This condition uses the easiness notion from the neural collapse phenomenon (Papyan et al., 2020; Hui et al., 2022; Görmez et al., 2022) and the early exit networks (Panda et al., 2016; Teerapittayanon et al., 2016; Kaya et al., 2019; Görmez et al., 2022). The neural collapse phenomenon, and more specifically the cascading collapse phenomenon states that the intermediate representations of the samples at each layer form clusters and the clusters get separated from each other more as we move deeper in the network. The idea of early exit networks is simply some samples are easier to predict and they can be predicted correctly at earlier layers.

The second condition appears to be obvious because when there is no computational budget, the final exit gives the best prediction. However, in some cases *overthinking* can occur, which means that although the early exit can give a correct prediction, the computations after that change the output and the final exit gives an incorrect prediction (Kaya et al., 2019). With the second condition, we guarantee that the samples which are prone to overthinking are not discarded from the training set. This condition is also important because studies show that adversarial attacks can make the network choose the final exit most of the time although the sample is easy, which leads to redundant computation (Kaya

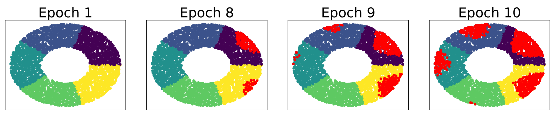


Figure 2. EEPrune discards samples (red) that are furthest away from the decision boundaries.

et al., 2019). Hence, this condition also serves as protection against adversarial attacks.

The third condition effectively determines the rate of pruning. Depending on the threshold, if the early exit model is sure enough about its prediction, that sample is pruned. Our full algorithm can be seen in the Appendix.

To better understand the effectiveness of EEPrune, we apply it to a toy dataset and analyze which samples are pruned. The dataset is shown in Figure 2. There are 5 classes with 1000 samples each. The data is two dimensional. We use a simple 3 layer feed forward neural network with 10, 10, and 5 neurons at each layer respectively. To make it an early exit network, we add a linear layer with 5 neurons after the first layer. We use $t = 0.7$ as our threshold and run EEPrune for $E = 10$ epochs. We show the pruned samples at certain epochs in Figure 2 with red. It can be seen that the pruned samples are the ones which are farthest away from any decision boundary. As the model is trained more, more data is pruned but the pruning pattern stays the same. When the model is trained on the remaining samples, it still performs the same without loss in accuracy. The behavior of EEPrune resembles that of support vector machines (SVMs): Note that in the latter, removing the non-support vectors does not change the decision boundaries. At least, for the example in Fig. 2, EEPrune correctly identifies the "non-support vector" that are irrelevant for the class decision boundaries and successfully removes them from the training process.

3.1. Variations of EEPrune

EEPrune is simple and flexible, new data pruning schemes can be easily be proposed. Here, we describe two natural ideas based on accumulating the loss values of the exits and assigning weights to the original conditions of EEPrune.

3.1.1. EEPRUNE-LOSS

According to a previous study conducted by (Lee et al., 2015), early exits can act as a form of regularization when the exit losses are accumulated and the exit layers are optimized together. In an ideal scenario, easy samples would have both low early exit losses and low final exit losses, but this may not always be the case. We propose evaluating the samples based on the sum of the exit losses, akin to how the exit layers are trained jointly by accumulating losses. By sorting the samples based on their aggregate losses, we can prune those with the lowest overall loss.

3.1.2. EEPRUNE-SOFT

EEPrune is a strict pruning method in the sense that it requires all three conditions to be met for a sample to be pruned. However, it can be made more lenient by assigning weights to the conditions. To this end, we propose assigning weights $\omega_1, \omega_2, \omega_3$ to the three conditions, where the sum of the weights is one. This approach allows for a better assessment of the conditions, with each weight indicating the degree to which the condition should be met in order for a sample to be pruned. For example, if the first condition is satisfied, the sample would be assigned a score of ω_1, and so on for the other conditions.

While this weighting system provides a more refined approach to scoring the samples, it may not be sufficient in cases where there is a tie. To address this limitation, we propose the addition of a fourth score, $(\max \hat{y}_{ee}^{(i)} + \max \hat{y}_{fe}^{(i)}) \cdot \omega_4$, which averages the maximum element of the early exit prediction vector and the final exit prediction vector for the sample of interest. Here, ω_4 should be very small to avoid dominating the other scores. By incorporating this fourth score, we can better differentiate between samples with similar scores and make more informed decisions about which samples to prune. Finally, the samples can be sorted according to their total score and the samples with the lowest total score can be pruned.

4. Numerical Results

In this section, we provide numerical results. In our experiments, we used CIFAR-10, CIFAR-100 and Tiny Imagenet datasets (Krizhevsky et al., 2009; Le & Yang, 2015). We utilized two widely-used and highly-regarded models: MobileNetV3-large (Howard et al., 2019) and ResNet-50 (He et al., 2016). For the MobileNetV3-large model, we added a linear layer after the ninth network layer to serve as the early exit point. For the ResNet-50 model, we added a linear layer after the seventh bottleneck block to serve as the early exit point.

4.1. Training Details

We train our models for 200 epochs using stochastic gradient descent with a single NVIDIA RTX A6000 GPU. To reduce overfitting and improve generalization, we use label smoothing with $\alpha = 0.1$ (Müller et al., 2019) and mixup with $\alpha_{mixup} = 0.2$ (Zhang et al., 2017). We also employ cosine learning rate decay with a warm-up duration of 5 epochs to gradually reduce the learning rate over the course of training. To optimize GPU memory usage and accelerate training, we set different batch sizes for the MobileNetV3-large and ResNet-50 models. We use 2048 for MobileNetV3-large and 1024 for ResNet-50 models. To ensure the stability and reliability of our results, we repeat each method-model-dataset evaluation 3 times and report the mean and standard deviation in our figures.

4.2. Experiments

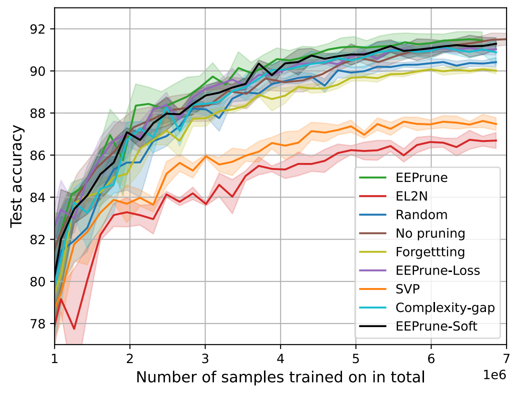
We conducted six experiments for each method, model, and dataset combination to prune 10% to 60% of the training set. After each dataset pruning phase, the models are reinitialized and trained from scratch on the pruned dataset. We repeated each experiment 3 times. Due to limited space, we only present some of the experimental results here. Additional figures can be found in the Appendix.

For EEPrune and its variations, we used $E = 10$ for CIFAR datasets and $E = 15$ for the Tiny Imagenet dataset. Since EEPrune depends on satisfying three conditions simultaneously, it was not always possible to find enough samples that satisfied the conditions at each of the aforementioned percentages. Therefore, we ran experiments for $t \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and stopped the training early (i.e., before 200 epochs) for some t values to ensure that all methods used an equal number of forward and backward passes for a fair comparison.

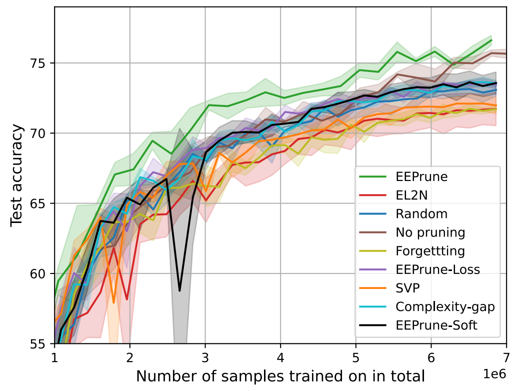
For EEPrune-Loss, after training for E epochs, we calculated the sum of the early exit loss and the final exit loss for each sample, sorted them, and pruned the lowest scoring $10\%, 20\%, 30\%, 40\%, 50\%, 60\%$ of samples based on their loss. For EEPrune-Soft, we used $\omega_1 = 0.4$, $\omega_2 = 0.2$, $\omega_3 = 0.4$, and $\omega_4 = 0.001$ for weighting the conditions.

We compared our 3 methods with the following baselines: EL2N (Paul et al., 2021), forgetting (Toneva et al., 2018), SVP (Coleman et al., 2019), complexity gap (Nohyun et al., 2023), no pruning, and random pruning. We explain each baseline further in the Appendix.

From Figure 3a, and it is evident that for the MobileNetV3-large model on the CIFAR-10 dataset, EEPrune, EEPrune-Loss, and EEPrune-Soft methods outperform the no pruning baseline at a moderate pruning rate of 30%. Complexity gap and random pruning achieves comparable performance, but EL2N, SVP, and forgetting methods exhibit noticeably worse performance than EEPrune. For low and high pruning rates, please see the Appendix. EEPrune and its variations continue to exhibit superior performance compared to all other methods. At high pruning rates, it is evident that EEPrune variations clearly outperform all other methods, including EEPrune itself. This demonstrates that EEPrune has an advantage in that its conditions can be easily modified to achieve excellent performance across various pruning rates. Figure 3b shows that EEPrune outperforms every other method. We attribute this to the fact that by pruning the simpler samples, the model can make the training set distribution more closely resemble the distribution of the test set. It is worth noting that at high pruning rates, random pruning surpasses all non-early-exit-based pruning methods. In the low and moderate pruning regimes, the complexity gap method performs very similarly to EEPrune variations but not EEPrune itself.



(a) CIFAR-10, MobileNetV3-Large



(b) CIFAR-100, ResNet-50

Figure 3. Dataset pruning performances of various algorithms. 30% of the dataset is pruned.

5. Conclusion

We presented EEPrune, a dataset pruning algorithm that utilizes the inherent ability of early exit networks to distinguish easy samples from the difficult ones. We showed that EEPrune achieves superior performance across various pruning rates as compared to existing approaches. We note that EEPrune and variants do not need a full training session to identify the easy samples, unlike competing algorithms. This makes them valuable for resource constrained settings.

Acknowledgements

This work was supported in part by the Army Research Lab (ARL) under Grant W911NF-21-2-0272, and in part by the National Science Foundation (NSF) under Grant CNS-2148182.

References

Optimized training and scalable implementation of conditional deep neural networks with early exits for fog-supported iot applications. *Information Sciences*, 521: 107–143, 2020. ISSN 0020-0255.

Agarwal, P. K., Har-Peled, S., Varadarajan, K. R., et al. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52(1):1–30, 2005.

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pp. 527–536. PMLR, 2017.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways, 2022.

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., van Steenkiste, S., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M. P., Gritsenko, A., Birodkar, V., Vasconcelos, C., Tay, Y., Mensink, T., Kolesnikov, A., Pavetić, F., Tran, D., Kipf, T., Lučić, M., Zhai, X., Keysers, D., Harmsen, J., and Houlsby, N. Scaling vision transformers to 22 billion parameters, 2023.

Feldman, D. and Langberg, M. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 569–578, 2011.

Görmez, A., Dasari, V. R., and Koyuncu, E. E2cm: Early exit via class means for efficient supervised and unsupervised learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2022. doi: 10.1109/IJCNN55064.2022.9891952.

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., and Wang, Y. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (11):7436–7456, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

Huang, G., Chen, D., Li, T., Wu, F., Van Der Maaten, L., and Weinberger, K. Q. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

Hui, L., Belkin, M., and Nakkiran, P. Limitations of neural collapse for understanding generalization in deep learning. *arXiv preprint arXiv:2202.08384*, 2022.

Kaya, Y., Hong, S., and Dumitras, T. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pp. 3301–3310. PMLR, 2019.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. Deeply-supervised nets. In *Artificial intelligence and statistics*, pp. 562–570. PMLR, 2015.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Nohyun, K., Choi, H., and Chung, H. W. Data valuation without training of a model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XIzO8zr-WbM.

OpenAI. Gpt-4 technical report, 2023.

Panda, P., Sengupta, A., and Roy, K. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 475–480. IEEE, 2016.

Papyan, V., Han, X., and Donoho, D. L. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

Paul, M., Ganguli, S., and Dziugaite, G. K. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34: 20596–20607, 2021.

Scardapane, S., Scarpiniti, M., Baccarelli, E., and Uncini, A. Why should we add early exits to neural networks? *Cognitive Computation*, 12(5):954–966, 2020.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S., Crowson, K., Schmidt, L., Kaczmarczyk, R., and Jitsev, J. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Settles, B. Active learning literature survey. 2009.

Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., and Villalobos, P. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., and Morcos, A. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022a.

Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., and Morcos, A. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022b.

Sutton, R. The bitter lesson. *Incomplete Ideas (blog)*, 13(1), 2019.

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., and Choi, Y. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9275–9293, 2020.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Teerapittayanon, S., McDanel, B., and Kung, H.-T. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

Thoppilan, R., Freitas, D. D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H., Jin, A., Bos, T., Baker, L., Du, Y., Li, Y., Lee, H., Zheng, H. S., Ghafouri, A., Menegali, M., Huang, Y., Krikun, M., Lepikhin, D., Qin, J., Chen, D., Xu, Y., Chen, Z., Roberts, A., Bosma, M., Zhou, Y., Chang, C., Krivokon, I., Rusch, W., Pickett, M., Meier-Hellstern, K. S., Morris, M. R., Doshi, T., Santos, R. D., Duke, T., Soraker, J., Zevenbergen, B., Prabhakaran, V., Diaz, M., Hutchinson, B., Olson, K., Molina, A., Hoffman-John, E., Lee, J., Aroyo, L., Rajakumar, R., Butryna, A., Lamm, M., Kuzmina, V., Fenton, J., Cohen, A., Bernstein, R., Kurzweil, R., Aguera-Arcas, B., Cui, C., Croak, M., Chi, E. H., and Le, Q. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239, 2022. URL https://arxiv.org/abs/2201.08239.

Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Yang, L., Han, Y., Chen, X., Song, S., Dai, J., and Huang, G. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2369–2378, 2020.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

A. The EEPrune Algorithm

Algorithm 1 EEPrune

```
Input: Data $D$, early exit network $F$ with parameters $\theta$, threshold $t$, epoch $E$
$P \leftarrow \{\}$
for $epoch = 1$ to $E$ do
  for $({x_0}^{(i)}, y^{(i)}) \in D$ do
    $\hat{y}_{ee}^{(i)}, \hat{y}_{fe}^{(i)} = F({x_0}^{(i)})$
    $J = -\sum_{c=1}^{C} {y_c}^{(i)} \log(\hat{y}_{ee_c}^{(i)}) - \sum_{c=1}^{C} {y_c}^{(i)} \log(\hat{y}_{fe_c}^{(i)})$
    $\theta \leftarrow \theta - \gamma \frac{\partial J}{\partial \theta}$
    if $\arg\max \hat{y}_{ee}^{(i)} = \arg\max y^{(i)}$ and $\arg\max \hat{y}_{fe}^{(i)} = \arg\max y^{(i)}$ and $\max \hat{y}_{ee}^{(i)} > t$ then
      $P \leftarrow P \cup \{{x_0}^{(i)}\}$
    end if
  end for
end for
$D \leftarrow D \setminus P$
Train $F$ on $D$ as desired
```

B. Dataset Pruning Methods

The Error-L2-Norm (EL2N) method (Paul et al., 2021) creates an ensemble of a few models and trains them for a short period of time. The error vectors from each model are then averaged, and the samples with the lowest error norm are discarded. By discarding the samples with the lowest error norm, this method ensures that the most critical samples are retained for further training. In our experiments, we used an ensemble of 10 models with the same architecture as our main model trained on the pruned dataset. Each model in the ensemble was trained for 15 epochs. We calculated the average ℓ_2-norm of the models' error (i.e., prediction minus the ground truth label vector) for each sample and pruned the lowest $10\%, 20\%, 30\%, 40\%, 50\%, 60\%$. Considering EEPrune and its variants use only one model and use $E = 10$ for a short amount of training, it can be seen that EEPrune and its variations use much less computational resources for pruning.

Forgetting method (Toneva et al., 2018) tracks the number of times each training sample is first predicted correctly but later incorrectly throughout an entire training session. Selection via proxy (SVP) method (Coleman et al., 2019) does the same, but uses smaller models and shorter training. Dataset cartography method (Swayamdipta et al., 2020) also analyzes the training dynamics by looking at the model's confidence in its output and the variance of its predictions. In our experiments, we trained the model on the full dataset for 200 epochs and tracked the amount of forgetting events for each sample. We then sorted the samples based on the amount of forgetting events and pruned the lowest $10\%, 20\%, 30\%, 40\%, 50\%, 60\%$. For SVP, we followed the same approach as forgetting, but used a MobileNetV3-small and ResNet-18 model instead of MobileNetV3-large and ResNet-50 respectively. We also trained the model for 50 epochs instead of the full 200 epochs. Since EEPrune and its variants use $E = 10$ for a short amount of training, it can be seen that EEPrune and its variations use much less computational resources for pruning compared to forgetting and SVP.

(Nohyun et al., 2023) proposed a novel training-free scoring method called the *complexity gap score*. This approach involves measuring the difference between the data complexity when a sample is removed from the training set. The data complexity measure is defined as the extent to which a data sample contributes to the movement of network parameters during training (Nohyun et al., 2023). In our experiments we used the source code provided by (Nohyun et al., 2023) to calculate the complexity gap score for each dataset and we pruned the lowest $10\%, 20\%, 30\%, 40\%, 50\%, 60\%$ samples based on the score. Calculating complexity gap scores does not require training, therefore in that sense it requires less computational resources for pruning compared to EEPrune. Similarly, random pruning also does not require training. However, the amount of computational resources EEPrune demands is still very low, e.g. $E = 10$ epochs of training, and EEPrune performs better than complexity gap method and random pruning by a significant margin.

Two other areas related to dataset pruning are active learning and dataset distillation. Active learning involves selecting the most informative data to label from a pool of unlabeled data, with the aim of maximizing the model's accuracy (Settles, 2009; Sener & Savarese, 2017). This approach differs from dataset pruning, which discards some labeled data once and for all to keep the size of the training set small and training cost low. In contrast, active learning repeats the process of selecting unlabeled data and training the model on it. Dataset distillation also aims to reduce the size of the dataset but does so by creating a smaller and often more difficult to interpret dataset from a larger, more human-understandable dataset (Wang et al., 2018).

C. More Results

Here we present the entirety of our results.

As seen from Figure 4, EEPrune and its variations achieve the same performance as the no pruning baseline. At low pruning rates, EEPrune and its variations show comparable performances to each other, but at high pruning rates EEPrune-Soft outperforms the others. EL2N, SVP and forgetting consistently performs poorer than EEPrune and its variations. Complexity gap method performs on par witth EEPrune at low pruning rates, but its performance decreases as pruning rate increases. Also, it is important to note that random pruning outperforms EL2N, SVP and forgetting across all pruning rates. Figure 6 shows that EEPrune performs the best among all pruning rates. At 60% pruning rate, all baselines perform poorly except EEPrune.

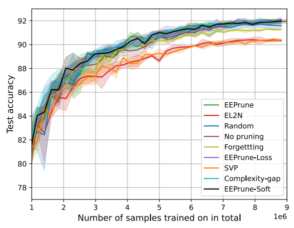
When the number of classes increases, it can be seen from Figure 5 that EEPrune performs the best among all methods, it even outperforms the no pruning baseline. We attribute this phenomenon to the pruning of simpler samples, which enables the model to align the distribution of the training set more closely with that of the test set. When we increase the number of classes even more, Figure 8 shows that EEPrune and its variations continue to perform better than the others.

D. Future Work

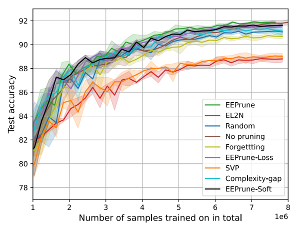
EEPrune's simplicity and practical approach to assessing sample difficulty make it an attractive method for modification to improve the quality of dataset pruning. We outline some promising research directions that could enhance its performance.

One potential modification draws inspiration from contrastive learning and data augmentation techniques (Chen et al., 2020). Similar to how contrastive learning generates positive pairs from augmented samples, EEPrune could be adapted to discard a sample from the dataset if a specific set of augmented versions of the sample also satisfy the pruning conditions.

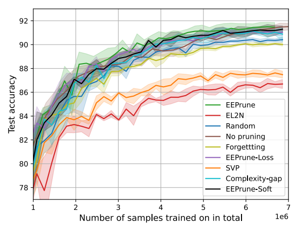
As the amount of unlabeled data is typically much greater than the amount of labeled data, EEPrune may also be extended to unsupervised learning tasks. Early exit networks have been shown to be effective in unsupervised learning tasks too, making them a promising candidate for further exploration in this area (Görmez et al., 2022). It is possible to modify the conditions of EEPrune to make it applicable to unsupervised learning tasks, opening up new avenues for cost-efficient training. Finally, the effect of having more exits and changing the exit locations can be investigated, which we leave as a future work.



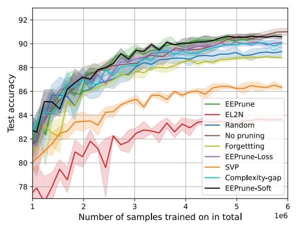
(a) 10%



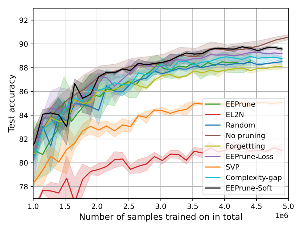
(b) 20%



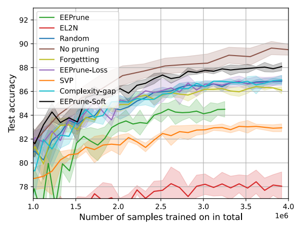
(c) 30%



(d) 40%

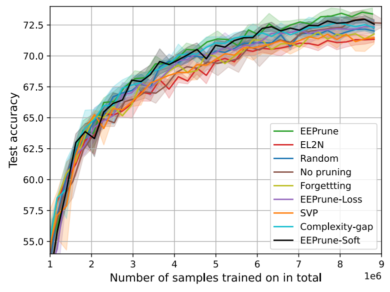


(e) 50%

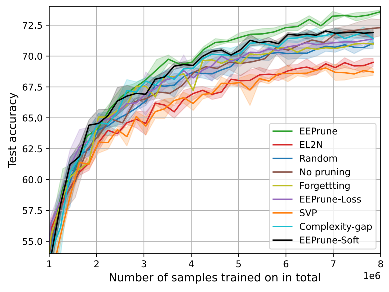


(f) 60%

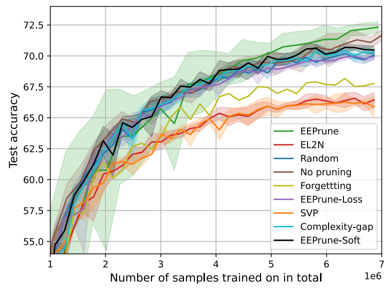
Figure 4. Dataset pruning performances of various algorithms on CIFAR-10 at various pruning rates. The model is MobileNetV3-large.



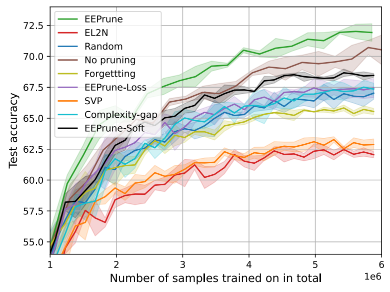
(a) 10%



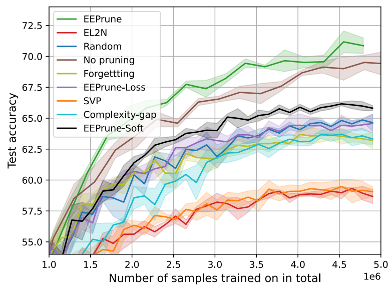
(b) 20%



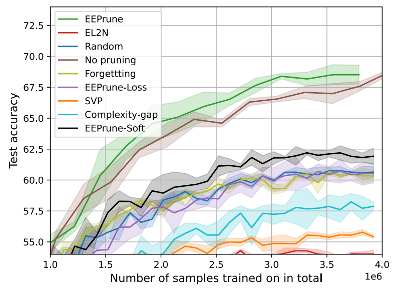
(c) 30%



(d) 40%

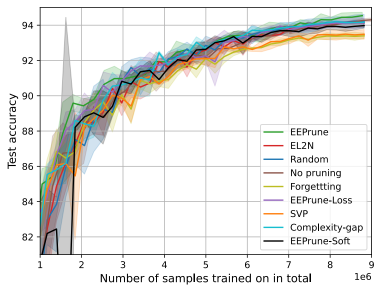


(e) 50%

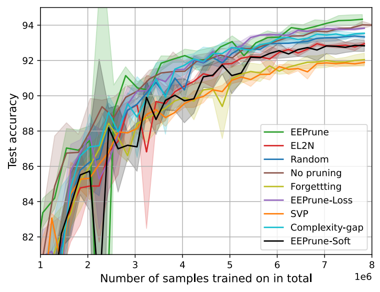


(f) 60%

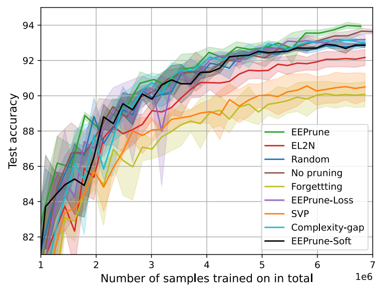
Figure 5. Dataset pruning performances of various algorithms on CIFAR-100 at various pruning rates. The model is MobileNetV3-large.



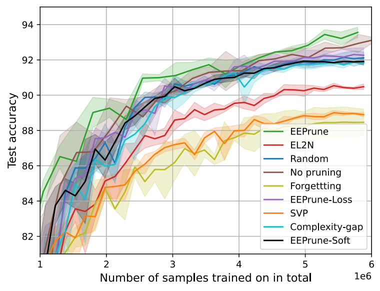
(a) 10%



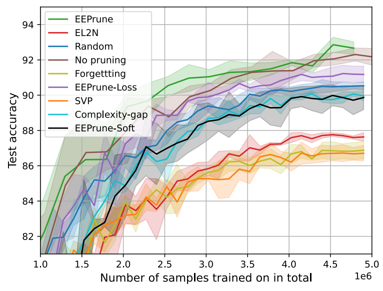
(b) 20%



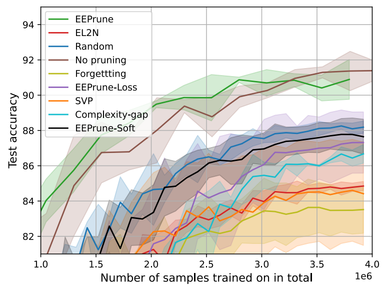
(c) 30%



(d) 40%

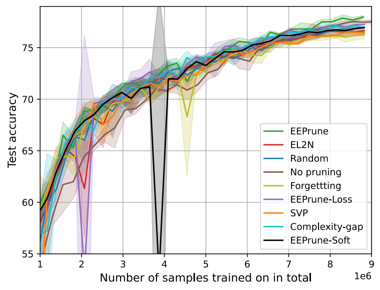


(e) 50%

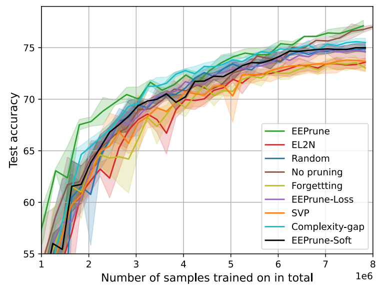


(f) 60%

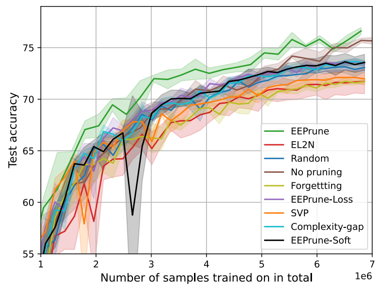
Figure 6. Dataset pruning performances of various algorithms on CIFAR-10 at various pruning rates. The model is ResNet-50.



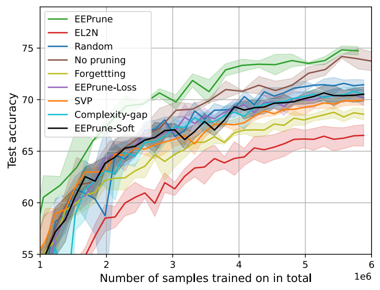
(a) 10%



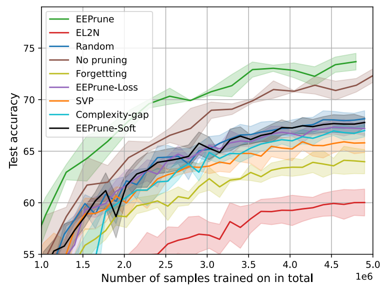
(b) 20%



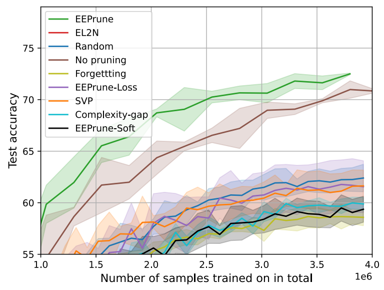
(c) 30%



(d) 40%

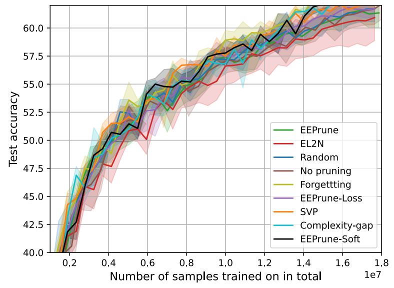


(e) 50%

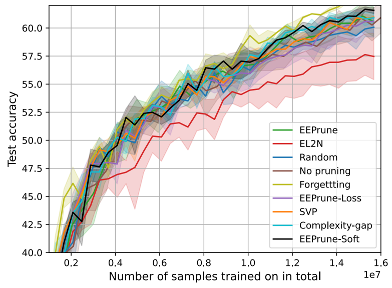


(f) 60%

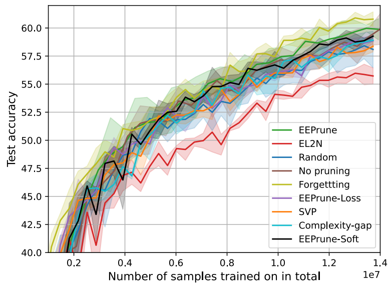
Figure 7. Dataset pruning performances of various algorithms on CIFAR-100 at various pruning rates. The model is ResNet-50.



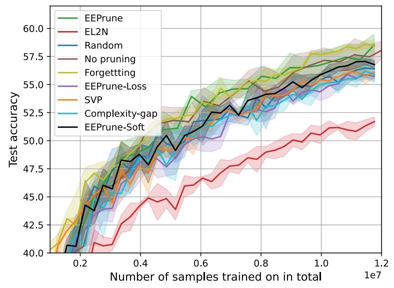
(a) 10%



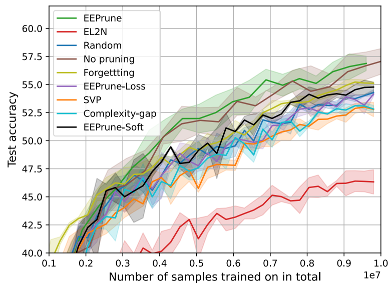
(b) 20%



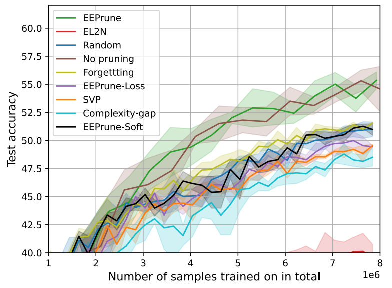
(c) 30%



(d) 40%



(e) 50%



(f) 60%

Figure 8. Dataset pruning performances of various algorithms on Tiny Imagenet at various pruning rates. The model is MobileNetV3-large.