A Data Pruning Method with Feature Distillation for Improved Computational Efficiency

1st Mike Soricelli
Computer & Information Science
University of Massachusetts Dartmouth
Dartmouth, USA
msoricelli@umassd.edu

2nd Russell Thompson
NUWC Division Newport
Naval Sea Systems Command
Newport, USA
russell.j.thompson31.civ@us.navy.mil

3rd Youchou Chang
Computer & Information Science
University of Massachusetts Dartmouth
Dartmouth, USA
ychang1@umassd.edu

4th Christopher J Hixenbaugh
NUWC Division Newport
Naval Sea Systems Command
Newport, USA
christopher.j.hixenbaugh2.civ@us.navy.mil

***Abstract*—As the capabilities of Neural Network models grow, so does the cost of power consumption and time, which consequently makes training state of the art models on resource constraint devices more challenging. In this paper, we present a novel method to reduce training costs by performing a version of data pruning, which prunes batches of data based on how far their feature extractions are from a set of feature distilled vectors. We call this new method Data pruning via Feature distillation. We employ a neural network layer, referred to as a feature distilled layer, which maintains a set of vectors which aim to approximate the distribution of extracted features within a neural network. This set of feature distilled vectors dynamically adjusts throughout the training process to accomplish this approximation while also adjusting to changes in the network throughout training. We demonstrate that this method significantly reduces overall training time while maintaining or even improving model performance across various datasets and architectures. Experiments conducted on MNIST, CIFAR10, CIFAR100, and CalTech-256 using ResNet models show performance increases of up to 47.41% speed up and a 40.11% decrease in GPU power consumption, while also maintaining accuracy within 1% of the baseline model. Our work presents as a flexible addition to a neural network which can be added in after feature extraction layers to be used for accelerating training through data pruning.**

***Index Terms*—Dataset pruning, Data Distillation, Vector Quantization**

I. Introduction

The exponential growth in the size and complexity of neural network models has led to a corresponding increase in training time and computational resources required. This trend poses significant challenges for researchers and practitioners, particularly in resource-constrained environments. While various techniques have been proposed to address this issue, there remains a pressing need for innovative approaches that can substantially reduce training time without compromising model performance.

The work is supported by the Office of Naval Research (ONR) grant N000142312123

This paper presents a novel method for accelerating neural network training by selectively performing backpropagation on samples that best represent the distribution of extracted features. Our approach draws inspiration from the literature that focuses on data-centric methods to improve training efficiency [1], [4], [9], [15], [17], [19], [21]–[26]. The methods that encapsulate data-centric approaches include dataset distillation, dataset pruning, and curriculum learning. We consider our approach to be a mix of dataset pruning and dataset distillation.

Dataset distillation is a technique that aims to condense the knowledge from a large dataset into a smaller, synthetic dataset while preserving the performance of models trained on it. This approach reduces storage requirements and accelerates training times. Dataset pruning, on the other hand, involves selectively removing less informative or redundant samples from the original dataset to create a more efficient subset for training. Unlike distillation, pruning works with actual data points rather than synthesizing new ones.

At the core of our method is the utilization of a neural network layer that functions similarly to the vector quantization scheme proposed in [20], which maintains a codebook of representative samples. The difference with our method is that we do not use the layer to quantize the output vectors; instead, we maintain a similar discretized set of quantized feature vectors and calculate the loss between the input feature vectors for a given batch with the closest discretized vector to determine whether backpropagation should be performed on the given batch of data. This discretized set of vectors serves a role analogous to the synthetic dataset in data distillation approaches, acting as an approximation for the extracted feature distribution. We will refer to this set as the *feature distilled set of vectors*. We leverage this distribution to perform a form of dataset pruning by selectively applying backpropagation to samples that contribute most significantly to model learning. We refer to this method as *pruning via Feature Distillation*.

Our approach differs from traditional data distillation meth-

ods in that it operates dynamically during the training process. Our approach also differs in that we are not synthesizing data points; rather, we are synthesizing extracted feature representations that are used as feature distilled vectors in our feature distillation layer. This adaptive strategy allows for a more nuanced and efficient learning process, potentially capturing subtle shifts in the importance of different data points as training progresses.

We demonstrate that by selectively applying backpropagation to samples identified through our vector quantization scheme, we can significantly reduce the overall training time while maintaining, and in some cases even improving, model performance. Our experimental results across a range of datasets and model architectures show consistent improvements in training efficiency, with particularly notable gains on large-scale datasets and complex models.

This paper makes the following key contributions:

- A novel feature distillation method for selective backpropagation in neural network training.
- A dynamic update mechanism of the feature distribution that adapts to evolving data importance during training.
- Empirical evidence demonstrating significant reductions in training time across various benchmarks.
- Analysis of the trade-offs between training speed, model performance, and computational resource utilization.

II. Related Works

Dataset pruning involves methods that attempt to find a subset of the data that aims to maintain the most informative training samples while removing the most redundant ones. Research on data selection methods date back to the early 2000s when [9] introduces coresets for k-means clustering, laying the groundwork for data summarization methods. Data pruning has also played a role in modern machine learning research. [19] explores the idea of identifying forgettable examples during training that could be removed from training without affecting the model performance. Scoring based methods have been used to help identify importance of training examples early in the training process, which is a key heuristic for modern data pruning techniques [15]. Optimization-based sample selection methods that considers the impact of removing training examples on the model's generalizability to select a small subset of data for training is used in [23]. A more recent development is a dynamic uncertainty based pruning approach [10]. The aim here is to target training samples where the model expresses the most uncertainty in predictions over the course of training.

Data condensation is another Data-Centric approach that aims to improve training efficiency of a model while also maintaining model performance by deriving a representative synthetic dataset from an original dataset with significantly reduced size. [21] introduces the concept of dataset distillation in their seminal paper, demonstrating that a small number of synthetic training images learnt from an original dataset can be used to train a model and match accuracy of the original dataset on simpler tasks. further advances in this field are made with a method that performs dataset condensation with gradient matching [25]. This method defines a gradient matching training objective for a synthetic dataset by minimizing distance of gradients produced by the synthetic dataset and original dataset when training a model. An alternative approach to gradient matching is data condensation based on distribution matching [24]. This method trains the synthetic dataset to match the distribution of the original dataset, which produces similar performance to prior methods while also avoiding the computational complexity of prior methods when training the synthetic dataset. Another popular method for performing data distillation is Matching Training Trajections(MTT) [3]. MTT aims to optimize the synthetic data by trying to match parameter trajectories between the synthetic data and real data. More recent works in this field include parameterizing synthetic datasets in the intermediate feature space of generative models instead of using the pixels of the synthetic image as optimization parameters [4]. Also, Generalized Various Backbone and Statistical Matching (G-VBSM) aim to create synthetic datasets using an ensemble of multiple backbones and Generalized statistic matching between the synthetic data set and the real data set [17]. Other methods are utilizing state of the art generative models to create synthetic images for distilled data sets. Latent diffusion models are used to synthesize data for distilled data sets [18].

Vector quantization has been utilized for different purposes in deep learning. Vector quantization is used to compress a neural network by quantizing the weights of fully connected dense layers in [7]. They were able to achieve a compression of up to 16-24 times with only a 1% drop in accuracy. [20] introduced a vector quantized variational autoencoder, which has an encoder that learns discrete latent representations rather than continuous representations.

III. Methods

The goal of this method is to be able to leverage information about the extracted feature distribution to determine which data samples are more representative of the data distribution that we are trying to learn from. To accomplish this goal we need to be able to approximate the distribution of extracted features with a finite set of feature vectors. We also need to establish a useful heuristic to determine when an incoming batch is valid for backpropagation and when it is not.

A. Feature Distillation

To approximate the data distribution we use the feature distilled set of vectors. Throughout training this set of vectors gets updated based on methods used from [20] . We utilize an exponential moving average(EMA) method to update the distilled vectors, which was shown to lead to more stable learning of a codebook in [20] in order to minimize the loss between the features vectors that come in from the encoder for each training batch with the distilled vectors that each input

maps to. We will refer to this loss as the distillation loss and it is defined in [20]. The distillation loss is defined as the L2 norm of the feature representation vector of the image and the closest distillation vector that it maps to and can be seen in equation 1, where z(x) is the feature vector for input x and d is the closest distilled vector.

$$L_{distillation} = \|z(x) - d\|_2^2 \tag{1}$$

To minimize this loss using EMA equations 2, 3, and 4 from [20] can be utilized. In equation 2, d_k represents one of the k distilled vectors from the distilled vector set, $\tilde{d_k}$ represents the average of the feature vectors that maps to d_k, and β represents a decay parameter set between 0 and 1 (usually set to 0.99). In equation 3, n_k represents the number of feature vectors that map to the distilled vector k, and N_k represents the amount of feature vectors that have mapped to distilled vector k during the update. The goal of equation 2 is to update vector d_k based on the average of the vectors that map to it over the training iterations, and equation 3 aims to estimate the number of vectors that are mapping to d_k. Equation 4 uses these 2 equations to normalize the distilled vectors to maintain a balanced representation across the distillation set. Normalization ensures that each vector's influence is proportional to its usage, preventing any single vector from dominating due to frequency

$$d_k \leftarrow \beta d_k + (1 - \beta)\tilde{d_k} \tag{2}$$

$$n_k \leftarrow \beta n_k + (1 - \beta)N_k \tag{3}$$

$$d_k^{\text{normalized}} = \frac{d_k}{n_k} \tag{4}$$

We refer to this process as feature distillation because over the course of training we are adjusting these vectors to match the distribution of the feature representation of the data after the data has been passed through convolutional layers. We refer to this as a distillation method because we are condensing the distribution of feature vectors into a discrete set of vectors. This draws similarities to data distillation methods that aim to condense a given data set to a smaller dataset to train a model, the differences being that we use the distilled set to guide training by pruning data samples, not to train the model itself using synthetic samples which is what occurs in data distillation methods.

B. pruning

To determine when we should consider skipping backpropagation for a given batch of data we need to use a designed method that aims to remove batches that are far away from closest distilled features. To accomplish this we use a very simple method where we pick a quantile of the ordered distillation loss calculations of the prior epoch to set a loss threshold for the next training iteration. This threshold is then utilized within the distillation layer to set a flag for a given batch to halt backpropagation if the distillation loss of that batch exceeds the threshold value. We observe that the movement in distillation loss stabilizes after early stages of training, this results in this quantile method performing as a rough estimate of a threshold that would accomplish pruning a user defined portion of samples, with very little overhead costs. We also factor in the difference of threshold estimates between successive training epochs to get a better estimate of where the quantiles will be in the next training epoch. This addition helps us get a better value to use for the threshold, which will result in the number of pruned samples to be closer to the user defined percentage.

Algorithm 1 lays out the step by step process with how this works. In our experiments we also consider adding a schedule to decay the targeted threshold over the course of training to the algorithm. We use a linearly decaying schedule to modify the target quantile from epoch to epoch, other types of schedules will be something to that is left for future works.

Algorithm 1 Training Loop with EMA Loss Thresholding

Initialize model parameters
Initialize codebook vectors e_k
Initialize empty list for losses $L_{previous}$
Initialize intitial threshold to sufficiently large value
Define hyperparameter: quantile q (e.g., $q = 0.5$ for the 50th percentile)
for each epoch **do**
  Reset losses for this epoch: $L_{current} \leftarrow []$
  for each batch of data (x_i) **do**
    Compute the distillation loss during using:

$$L_i = \|z(x) - d\|_2^2$$

    if $L_i > threshold$ **then**
      skip backpropogation continue to next batch
    else
      Update model parameters via backpropagation
    end if
    Append L_i to $L_{current}$
  end for
  Update the threshold:

$$\text{threshold} = \text{percentile}(L_{\text{previous}}, p)$$

  if $epoch > 0$ **then**
    $\Delta\text{threshold} = \text{threshold} - \text{threshold}_{\text{previous}}$
    $\text{threshold} = \text{threshold} + \Delta\text{threshold}$
  end if
end for

C. manifold learning

The general idea of this method is to leverage a discrete representation of the feature space to determine if incoming data is representative of the overall data distribution in the feature space. Why does this work in the feature space? We consider the assumption that deep feature representations in a neural network map to a valid lower dimensional manifold.

This is often referred to as the "manifold hypothesis". This hypothesis is explored in many works we refer the reader to explore these works to learn more [14], [5], [11]. Based on this hypothesis we use the feature space to approximate a lower dimensional manifold using the discrete vector representation. By pruning data points that are far away from our discrete vectors we are more likely to remove data points which are far away from the manifold which are likely to be less informative samples (noisy, outliers).

IV. Experiments

A. Experimental Settings

To test our methods we perform image classification tasks across 5 different datasets including MNIST [13], CIFAR10, CIFAR100 [12], CalTech-256 [8], and a Brain Tumor Imagery dataset [2]. MNIST consists of 70K 28x28 grayscale images of handwritten digits, the dataset contains 10 classes for the digits 0-9. CIFAR10 and CIFAR100 are image datasets that contain 60K 32x32 images of diverse classes, with the datasets having 10 and 100 classes respectively. CalTech-256 is a dataset that contains 30,607 real-world images across 257 categories, with varying pixel sizes that we scaled to 224x224. The Brain Tumor Imagery dataset contains images of 4 different types of brain tumors with image resolutions of 512x512 which we scale to 224x224 for training.

To perform these tasks we train a series of ResNet models on these datasets. We design a standard ResNet implementation for baseline metrics and we create a separate custom ResNet Implementation with the included feature distilled layer to perform training with batch pruning. The feature distilled layer is inserted after the convolutional layers of the model, in between the flattening layer of the model and the dense classification layers of the model. For our experiments, we employed a standard set of hyperparameters commonly used in image classification tasks. The network was trained using Adam optimizer with a momentum of with a weight decay of 1e-4. The learning rate was held constant throughout training at 0.001. We trained the model for a total of 100 epochs with a batch size of 512. Data augmentation techniques were applied during training, including random crops of 32x32 pixels with padding of 4. For regularization. Experiments were conducted using the PyTorch framework and we performed experiments utilizing the Google Colab environment with a T4 GPU and an Nvidia Jetson AGX Orin developer kit. These hyperparameters were chosen based on their general use in the field.

We perform experiments to determine the accuracy difference between the proposed method and the baseline. We also measure the power and time consumption of training with the two methods and calculate the relative efficiency gains. We also test how the performance of this method translates to different sized ResNet architectures and we perform ablation studies on threshold parameters and codebook size.

B. Performance

We evaluate our method by performing tests across MNIST, CIFAR10, and CIFAR100. Table 1 reports results from training a custom ResNet9 model. With all 3 datasets we see a consistent performance efficiency improvement, while maintaining very high accuracy. We never observe a drop of more than 1% in accuracy and in fact we see a slight accuracy improvement on CIFAR100 (+0.19%).

TABLE I

DataSet	Speed-up	Energy Usage	Accuracy (+/-)
MNIST	1.42x	68.3%	-0.15%
CIFAR10	1.36x	74.3%	-0.38%
CIFAR100	1.36x	74.3%	+0.19%

C. Cross Architecture analysis:

We also compared our method on different vanilla ResNet18, 34, 50, 101 architectures trained on CIFAR100. Results can be seen in Table 2. What we see from these results is that performance improvements are relatively consistent across the board. The one downside that we do notice a little more instability with our heuristic pruning method. In these tests we were targeting to prune 50% of the batches, but with ResNet 50 we experienced around 43% of batches to be removed, which lead to a slightly larger drop in accuracy (-1.67%)

TABLE II

Model	Speed up	Energy usage	Accuracy(+/-)
ResNet18	1.38x	61.3%	-0.75%
ResNet34	1.47x	59.9%	-0.81%
ResNet50	1.40x	61.4%	-1.67%
ResNet101	1.42x	61.0%	-1.50%

D. Using schedules:

We also decided to explore using a linear decaying schedule for setting the quantile that should be used for setting the threshold from epoch to epoch, as compared to the method we describe above that uses a static quantile number throughout training. For testing setup we start training with the quantile number set at 0.99 and we decay it by a rate of 0.01 per epoch. The formulation for the update can be seen in equation 5. We also stop decaying once the quantile number falls reaches 0.10. Results here are evaluated by performing tests across MNIST, CIFAR10, CIFAR100, and CalTech256. With all 4 datasets we see an insignificant decrease in test accuracy with gains in both time consumption and power consumption. Results can be seen in Table 4.

The results here indicate that using a scheduling with this method does lead to more stable training and less degradation in accuracy compared to the static method. This makes sense since we are effectively training with the entire dataset early on in training and we only start pruning a significant amount of the data later on in training once we have learned a better representation of the feature distillation set. Although we see

more stable training we also see less gains in energy increases, which also makes sense since we are effectively using all the data early on in training.

$$q_{\text{new}} = \text{q} - \alpha * \text{epoch} \tag{5}$$

TABLE III

Dataset	Speed up	Energy usage	Accuracy(+/-)
MNIST	1.25x	78.55%	-0.02%
CIFAR-10	1.26x	83.44%	-0.65%
CIFAR-100	1.17x	86.97%	-0.19%

E. comparison to other methods:

To test the performance of our method we decided to perform comparative experiments with other pruning based methods. To compare with our method we use a few coreset methods [6], [16] and we use an uncertainty based pruning method [10]. We perform all the tests with the same training settings as our other tests trained on CIFAR100. The results from our experiments can be seen in Table 4. The results show that our method is able to achieve a higher accuracy than the other methods. While the other methods do demonstrate faster training times they also require a trained model before the pruning can take place as opposed to our method where we dynamically prune data throughout training.

F. Tests on high resolution datasets:

We also perform tests to see how our method would perform on higher resolution datasets to test how our method performs on harder classification tasks that are more representative to real world tasks. The datasets we use are CalTech-256 and a Brain Tumor Imagery dataset. We observe significant accuracy degradation(-2.29%) when performing our method on CalTech-256, we suggest this may be due to the general difficulty of the dataset(more categories, higher resolution). On the Brain Imagery dataset we experience similar issues with the higher resolution images. We observe an accuracy degradation of (-2.56%). This behavior can be explained by the fact that the manifold learning in the feature space for higher dimensional images is much harder and complex than with low dimensional images. Results for these tests can be seen in table 5.

G. Distillation Set size:

During our prior experiments we use a set of 100 feature distilled vectors to use for our feature distilled layer in our initial tests. What we notice that for the smaller datasets that this number worked fine, but for the higher resolution datasets we found this number to be deficient. As discussed in the last section, the higher dimensional images map to a more complex manifold requiring more distilled vectors be represented well. For the high resolution datasets we use 5000 feature distilled vectors. This does enhance performance, but further analysis is required to find the optimal number of distilled vectors given the problem settings.

TABLE IV

Method	Speedup vs Baseline	Accuracy	Accuracy Δ
Baseline	1.00x	68.16%	-
Dyn-Unc	1.96x	63.66%	-6.60%
Entropy	1.96x	61.59%	-9.64%
EL2N	1.99x	55.06%	-13.10%
Ours	1.29x	67.75%	-0.60%

TABLE V

Dataset	Speed-up	Energy Usage	Accuracy (+/-)
Caltech	1.43x	65.71%	-2.29%
Brain Tumor	1.13x	62.76%	-2.56%

V. Discussion

The proposed method of Data Pruning via Feature Distillation presents a novel approach to accelerating deep learning training while maintaining model accuracy. This new method offers several notable advantages:

A. Efficiency Gains

The experimental results show significant improvements in training efficiency across the board with a number of datasets and model architectures. The method achieved a 1.47x speedup and a 40.11% decrease in GPU power consumption while maintaining accuracy within 1% of the baseline model while also closely matching the training trajectory of a baseline model.The improvements we observe in both time and energy efficiency demonstrate the potential of this method for resource-constrained environments.

B. Cross-Architecture Performance

We performed tests across different ResNet architectures(ResNet18, ResNet34, ResNet50, ResNet101). The findings suggests that the method is adaptable to models of different scale. While results vary across the networks slightly it seems that the depth of the method can be successfully applied to networks of different depths. Also it is worth noting that larger distillation sets are used for deeper networks to improve performance vs smaller networks.

C. Tradeoffs and Limitations

While the method shows promising results, there are some tradeoffs to consider:

Accuracy vs. Efficiency: Although we see efficiency improvements across the board with this method and promising results on MNIST, CIFAR10, and CIFAR100, we do see significant accuracy degradation when tests were performed on CalTech-256 and our Brain Tumor Imagery dataset, this result indicates that there still is work to do with our method on higher resolution images. Due to the added complexity of the data manifold the manifold learning is much more difficult with a discrete set of vectors than it is with lower dimensional images. More analysis is required to alleviate this problem. That being said when we used schedules we were able to reduce accuracy degradation since we were being less

aggressive with pruning early on in training, but we also did not save as much energy with this method.

Computational Complexity: While there is performance efficiency gains by leveraging the feature distilled layer for data pruning, there is also added memory overhead that may become an issue for larger models and higher resolution images. This especially can be an issue in resource-constrained environments.

D. Future Directions

Several avenues for future research emerge from this work:

Exploring schedules for threshold: When using a linearly decaying schedule we were able to see more stable training. This strategy may lead to better results in the future. Trying out different decaying schedules may lead to better efficiency or more stable training. Also other modifications to the pruning heuristic we use can be explored as well.

Extension to Other Domains: Future work will include experimenting with this method on different types on neural network architectures and different data modalities to demonstrate broader impact beyond image classification tasks.

Integration with Other Techniques: We also plan to run future experimentation combining our method with other forms of efficiency-boosting techniques, such as model compression or quantization. Demonstrating compatibility with well established methods could lead to greater viability of our method and even greater performance gains when combined with other methods.

Improved analysis for Manifold learning: To improve performance on more complex datasets integrating analytical techniques to understand the characteristics of the lower dimensional manifold would be beneficial for tuning hyperparameter to a specific learning task. This could lead to more stable performance on complex tasks that this method currently struggles with.

VI. Conclusion

Data Pruning via Feature Distillation presents a promising approach to improve speed and energy efficiency of Neural Network training. This is accomplished by selectively applying backpropogation on batches of data based on how close the batch data is to a quantized vector in feature space which we refer to as a feature distilled set. We perform experiments on MNIST, CIFAR10, CIFAR100, CalTech-256, and a brain tumor imagery dataset. Results on MNIST, CIFAR10, and CIFAR100 demonstrate significant efficiency gains while also maintaining accuracy within 1%, we were also able to achieve a 1.43x training speedup with accuracy degradation of 2.29% when training on CalTech-256. These results indicate that our method can be a useful tool to be used for Neural Network training within resource-constrained environments. While there are areas of future work that needs to be studied and further improvements to be made, the results demonstrated in this work present a promising approach towards improving efficiency of Neural Network training.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[2] Sartaj Bhuvaji. Brain tumor classification (mri), 2020. Accessed March 18, 2025.

[3] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[4] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Generalizing dataset distillation via deep generative prior, 2023.

[5] Uri Cohen, SueYeon Chung, Daniel D. Lee, and Haim Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature communications*, 11(1):746, 2020.

[6] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2020.

[7] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization, 2014.

[8] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech 256, Apr 2022.

[9] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 291–300, New York, NY, USA, 2004. Association for Computing Machinery.

[10] Muyang He, Shuo Yang, Tiejun Huang, and Bo Zhao. Large-scale dataset pruning with dynamic uncertainty. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 7713–7722, 2023.

[11] Bobak T. Kiani, Jason Wang, and Melanie Weber. Hardness of learning neural networks under the manifold hypothesis. *arXiv preprint arXiv:2406.01461*, 2024.

[12] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Gabriel Loaiza-Ganem et al. Deep generative models through the lens of the manifold hypothesis: A survey and new connections. *arXiv preprint arXiv:2404.02954*, 2024.

[15] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[16] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*, volume 34, pages 20596–20607, 2021.

[17] Shitong Shao, Zeyuan Yin, Muxin Zhou, Xindong Zhang, and Zhiqiang Shen. Generalized large-scale data condensation via various backbone and statistical matching, 2024.

[18] Duo Su, Jianlong Wang, Xiaoqian Luo, Minfeng Zhang, Yandong Guo, Yue Zhang, Yihao Gao, and Ying Shan. D4m: Dataset distillation via disentangled diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[19] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning, 2019.

[20] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018.

[21] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020.

[22] Yulin Wang, Yang Yue, Rui Lu, Yizeng Han, Shiji Song, and Gao Huang. Efficienttrain++: Generalized curriculum learning for efficient visual backbone training, 2024.

[23] Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence, 2023.
[24] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching, 2022.
[25] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching, 2021.
[26] Ganlong Zhao, Guanbin Li, Yipeng Qin, and Yizhou Yu. Improved distribution matching for dataset condensation, 2023.