Dataset Distillation with Infinitely Wide Convolutional Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

The effectiveness of machine learning algorithms arises from being able to extract useful features from large amounts of data. As model and dataset sizes increase, dataset distillation methods that compress large datasets into significantly smaller yet highly performant ones will become valuable in terms of training efficiency and useful feature extraction. To that end, we apply a novel distributed kernel based meta-learning framework to achieve state-of-the-art results for dataset distillation using infinitely wide convolutional neural networks. For instance, using only 10 datapoints (0.02% of original dataset), we obtain over 63% test accuracy on CIFAR-10 image classificiation task, a dramatic improvement over the previous best test accuracy of 40%. Our state-of-the-art results extend across many other settings for MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, and SVHN. Furthermore, we perform some preliminary analyses of our distilled datasets to shed light on how they differ from naturally occurring data.

1 Introduction

Deep learning has become extraordinarily successful in a wide variety of settings through the availability of large datasets [Krizhevsky et al., 2012, Devlin et al., 2018, Brown et al., 2020, Dosovitskiy et al., 2020]. Such large datasets enable a neural network to learn useful representations of the data that are adapted to solving tasks of interest. Unfortunately, it can be prohibitively costly to acquire such large datasets and train a neural network for the requisite amount of time.

One way to mitigate this problem is by constructing smaller datasets that are nevertheless informative. Some direct approaches to this include choosing a representative subset of the dataset (i.e. a coreset) or else performing a low-dimensional projection that reduces the number of features. However, such methods typically introduce a tradeoff between performance and dataset size, since what they produce is a coarse approximation of the full dataset. By contrast, the approach of *dataset distillation* is to synthesize datasets that are more informative than their natural counterparts when equalizing for dataset size [Wang et al., 2018, Bohdal et al., 2020, Nguyen et al., 2021, Zhao and Bilen, 2021]. Such resulting datasets will not be from the distribution of natural images but will have nevertheless learned to capture features useful to a neural network, a phenomenon which remains mysterious and is far from being well-understood [Ilyas et al., 2019, Huh et al., 2016, Hermann and Lampinen, 2020].

The applications of such smaller, distilled datasets are diverse . For nonparametric methods that scale poorly with the training dataset (e.g. nearest-neighbors or kernel-ridge regression), having a reduced dataset decreases the associated memory and inference costs. For the training of neural networks, such distilled datasets have found several applications in the literature, including increasing the effectiveness of replay methods in continual learning [Borsos et al., 2020] and helping to accelerate neural architecture search Zhao et al. [2021], Zhao and Bilen [2021].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

In this paper, we perform a large-scale extension of the methods of Nguyen et al. [2021] to obtain new state-of-the-art (SOTA) dataset distillation results. Specifically, we apply the algorithms KIP (Kernel Inducing Points) and LS (Label Solve), first developed in Nguyen et al. [2021], to infinitely wide convolutional networks by implementing a novel, distributed metalearning framework that draws upon hundreds of accelerators per training. The need for such resources is necessitated by the computational costs of using infinitely wide neural networks built out of components occurring in modern image classification models: convolutional and pooling layers (see §B for details). The consequence is that we obtain distilled datasets that are effective for both kernel ridge-regression and neural network training.

Additionally, we initiate a preliminary study of images and labels which KIP learns. We provide a visual and quantitative analysis of the data learned and find some surprising results concerning their interpretability and their dimensional and spectral properties. Given the efficacy of KIP and LS learned data, we believe a better understanding of them would aid in the understanding of feature learning in neural networks.

To summarize, our contributions are as follows:

1. We achieve SOTA dataset distillation results on a wide variety of datasets (MNIST, Fashion-MNIST, SVHN, CIFAR-10, CIFAR-100) for both kernel ridge-regression and neural network training. In several instances, our results achieve an impressively wide margin over prior art, including over 23% and 36% absolute gain in accuracy on CIFAR-10 and SVHN image classification, respectively, when using only 10 images (Tables 1, 2).
2. We develop a novel, distributed meta-learning framework specifically tailored to the computational burdens of sophisticated neural kernels (§B.2).
3. We highlight and analyze some of the peculiar features of the distilled datasets we obtain, illustrating how they differ from natural data (§4).
4. We plan on open sourcing the datasets we obtained using thousands of GPU hours for the research community to further investigate.

2 Setup

Background on infinitely wide convolutional networks. Recent literature has established that Bayesian and gradient-descent trained neural networks converge to Gaussian Processes (GP) as the number of hidden units in intermediary layers approaches infinity (see §5). These results hold for many different architectures, including convolutional networks, which converge to a particular GP in the limit of infinite channels [Garriga-Alonso et al., 2019, Novak et al., 2019, Arora et al., 2019]. The kernel function of the limiting GP depends on the architecture of the network (e.g. depth, stride, pooling etc) and on the inference procedure. Bayesian networks are described by the Neural Network Gaussian Process (NNGP) kernel, while gradient descent networks are described by the Neural Tangent Kernel (NTK). For many architectures, including ones we consider in this work, these kernels can be computed in closed form. Since we are interested in synthesizing datasets that can be used with both kernel methods and common gradient-descent trained neural networks, we focus on NTK in this work.

Infinitely wide networks have been shown to achieve SOTA (among non-parametric kernels) results on image classification tasks [Novak et al., 2019, Arora et al., 2019, Li et al., 2019, Shankar et al., 2020] and even rival finite-width networks in certain settings [Arora et al., 2020, Lee et al., 2020]. This makes such kernels especially suitable for our task. As convolutional models, they encode useful inductive biases of locality and translation invariance [Novak et al., 2019], which enable good generalization. Yet as kernel methods, they allow simple closed-form differentiation of the generalization loss (1) with respect to the training set.[1]

Specific models considered. The central infinite-width model we consider in this work is a simple 3-layer ReLU CNN with average pooling layers that we refer to as **ConvNet** throughout the text. This architecture is the default model used by [Zhao and Bilen, 2021, Zhao et al., 2021],[2] and was chosen

[1] However, inference with convolutional neural kernels poses its own unique computational challenges, which we describe and address in §B.2.

[2] We do not use instance normalization since it has no straightforward infinite-width analog.

in order to make our results easily comparable to these distillation methods. In several other settings we also consider convolutional networks without pooling layers which we label as **ConvVec**,[3] and networks with no convolutions and only fully-connected layers, which we label as **FC**. Depth of architecture is indicated by an integer suffix. All these models are easily implemented using the JAX [Bradbury et al., 2018] based Neural Tangents library [Novak et al., 2020].

Background on algorithms. We review the KIP (Kernel Inducing Points) and Label Solve (LS) algorithms introduced in Nguyen et al. [2021]. Given a kernel K, the kernel ridge-regression (KRR) loss function trained on a support dataset (X_s, y_s) and evaluated on a target dataset (X_t, y_t) is given by

$$L(X_s, y_s) = \frac{1}{2} \left\| y_t - K_{X_t X_s}(K_{X_s X_s} + \lambda I)^{-1} y_s \right\|_2^2, \tag{1}$$

where if U and V are sets, K_{UV} is the matrix of kernel elements $(K(u, v))_{u \in U, v \in V}$. Here $\lambda > 0$ is a fixed regularization parameter. The KIP algorithm consists of optimizing (1) with respect to the support set (either just the X_s or along with the labels y_s). Here, we sample (X_t, y_t) from a target dataset $\mathcal{D}$ at every (meta)step, and update the support set using a gradient-based methods. Additional variations include augmenting the X_t or sampling a different kernel K (from a fixed family of kernels) at each step.

The Label Solve algorithm consists of solving for the least-norm minimizer of (1) with respect to y_s. This yields the labels

$$y_s^* = \left(K_{X_t X_s}(K_{X_s X_s} + \lambda I)^{-1} \right)^+ y_t, \tag{2}$$

where A^+ denotes the pseudo-inverse of the matrix A. Note that here $(X_t, y_t) = \mathcal{D}$, i.e. the labels are solved for the whole target set.

In our applications of KIP and Label Solve, the target dataset $\mathcal{D}$ is always significantly larger than the support set (X_s, y_s). Hence, the learned support set or solved labels can be regarded as distilled or compressed versions of their respective targets. We also initialize our support images to be a subset of natural images, though they could also be initialized randomly.

Based on the infinite-width correspondence outlined above and in §5, dataset distillation using KIP or LS that is optimized for KRR should extend to the corresponding finite-width neural network training. Our experimental results in §3 validate this expectation across many settings.

3 Experimental Results

3.1 Kernel Distillation Results

We apply the KIP and LS algorithms using the ConvNet architecture on the datasets MNIST [LeCun et al., 2010], Fashion MNIST [Xiao et al., 2017], SVHN [Netzer et al., 2011], CIFAR-10 [Krizhevsky, 2009], and CIFAR-100. Here, the goal is to condense the train dataset down to a learned dataset of size 1, 10, or 50 images per class. We consider a variety of settings (image preprocessing method, whether to use augmentations, and whether to train the labels for KIP), the full details of which are described in §A. We list a simple, consistent subset of these results in Table 1. We highlight here that a crucial ingredient for our strong results in the RGB dataset setting is the use of regularized ZCA preprocessing.

Remarkably, we outperform all prior baselines across all dataset settings. Our results are especially strong in the small support size regime, with our 1 image per class results for KRR outperforming over 100 times as many natural images (see Table A1). We also obtain a significant margin over prior art, where on RGB datasets our margin ranges between 20% and 36% absolute gain in test accuracy.

3.2 Kernel Transfer

In §3.1, we focused on obtaining state-of-the-art dataset distillation results for image classification using a specific kernel (ConvNet). Here, we consider the variation of KIP in which we sample from a

[3] The abbreviation "Vec" stands for vectorization, to indicate that activations of the network are vectorized (flattened) before the top fully-connected layer instead of being pooled.

family of kernels (which we call sampling KIP). We validate that sampling KIP introduces robustness to the learned images in that they perform well for the family of kernels sampled during training.

Table 1: **Comparison with other methods.** The left group consists of neural network based methods. The right group consists of kernel ridge-regression. Underlined numbers indicate second best result. All settings for KIP involve the use of augmentations and label-learning. Grayscale datasets use standard channel-wise preprocessing while RGB datasets use regularized ZCA preprocessing.

	Imgs/Class	LD[1]	DC	DSA	KIP FC	LS ConvNet[2]	KIP ConvNet[2]
MNIST	1	60.9±3.2	91.7±0.5	88.7±0.6	85.5±0.1	73.4	**96.5±0.1**
	10	87.3±0.7	97.4±0.2	97.8±0.1	97.2±0.2	96.4	**99.1±0.1**
	50	93.3±0.3	98.8±0.1	99.2±0.1	98.4±0.1	98.3	**99.5±0.1**
Fashion MNIST	1	-	70.5±0.6	70.6±0.6	-	65.3	**76.7±0.2**
	10	-	82.3±0.4	84.6±0.3	-	80.8	**88.8±0.1**
	50	-	83.6±0.4	88.7±0.2	-	86.9	**91.0±0.1**
SVHN	1	-	31.2±1.4	27.5±1.4	-	23.9	**64.3±0.4**
	10	-	76.1±0.6	79.2±0.5	-	52.8	**80.9±0.5**
	50	-	82.3±0.3	**84.4±0.4**	-	76.8	**84.0±0.1**
CIFAR-10	1	25.7±0.7	28.3±0.5	28.8±0.7	40.5±0.4	26.1	**63.4±0.1**
	10	38.3±0.4	44.9±0.5	52.1±0.5	53.1±0.5	53.6	**75.5±0.1**
	50	42.5±0.4	53.9±0.5	60.6±0.5	58.6±0.4	65.9	**79.7±0.1**
CIFAR-100	1	-	12.8±0.3	13.9±0.3	-	23.8	**33.3±0.3**
	10	-	25.2±0.3	32.3±0.3	-	39.2	**49.5±0.3**

[1] LD [Bohdal et al., 2020], DC [Zhao et al., 2021], DSA [Zhao and Bilen, 2021], KIP FC [Nguyen et al., 2021]. Note that LD uses AlexNet architecture.
[2] This work.

Table 2: **Transfer of KIP and LS to neural network training.** Datasets obtained from KIP and LS using the ConvNet kernel are optimized for kernel ridge-regression and thus have reduced performance when used for training the corresponding finite-width ConvNet neural network. Remarkably, the loss in performance is mostly moderate and even small in many instances. Grayscale datasets use standard channel-wise preprocessing while RGB datasets use regularized ZCA preprocessing.

	Img/Class	DC/DSA	KIP to NN	Perf. change	LS to NN	Perf. change
MNIST	1	**91.7±0.5**	90.1	-5.5	71.3	-2.1
	10	**97.8±0.1**	97.5	-1.1	95.3	-1.1
	50	**99.2±0.1**	98.4	-0.7	98.0	-0.3
Fashion-MNIST	1	70.6±0.6	**73.8**	-9.5	61.3	-4.0
	10	84.6±0.3	**86.8**	-1.3	79.9	-1.0
	50	**88.7±0.2**	87.8	-4.7	84.9	-1.9
SVHN	1	31.2±1.4	**57.4**	-8.2	24.1	0.1
	10	**79.2±0.5**	75.1	-1.5	53.5	0.7
	50	**84.4±0.4**	80.4	-1.1	76.4	-0.5
CIFAR-10	1	28.8±0.7	**50.0**	-9.1	24.8	-1.3
	10	52.1±0.5	**62.7**	-4.6	49.2	-4.4
	50	60.6±0.5	**68.4**	-4.7	62.1	-3.8
CIFAR-100	1	13.9±0.3	**15.8**	-18.0	12.0	-11.8
	10	**32.3±0.3**	28.3	-17.4	25.0	-14.2

In Figure 1, we plot test performance of sampling KIP when using the kernels ConvNet, Conv-Vec3, and Conv-Vec8 (denoted by "all") alongside KIP trained with just the individual kernels. Sampling

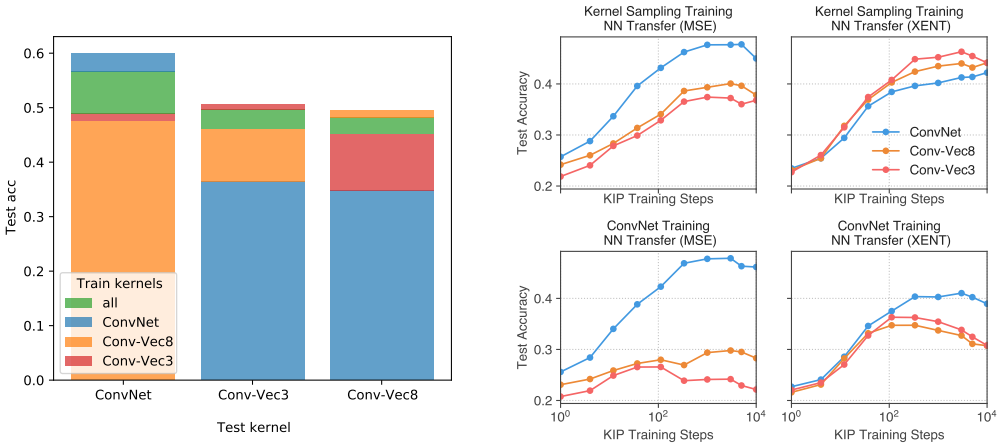

Figure 1: **KIP with kernel sampling vs individual kernel.** *Left:* Evaluation of three kernels, ConvNet, Conv-Vec3, Conv-Vec8 for KRR with respect to four train settings: sampling KIP ("all") which uses all the kernels or else KIP trained with the individual kernels. For all three kernels, "all" is a close second place, outperformed only if the kernel used for training is exactly the same as the one used for testing. *Right:* We take the learned images of the four train settings described above and transfer them to finite-width neural networks corresponding to ConvNet, Conv-Vec3, Conv-Vec8. Each point is a neural network trained on a specified KIP learned checkpoint. Top row is sampling KIP images and bottom row is the baseline using just ConvNet for KIP. These plots indicate that sampling KIP improves performance across the architectures that are sampled, for both MSE and cross entropy loss.

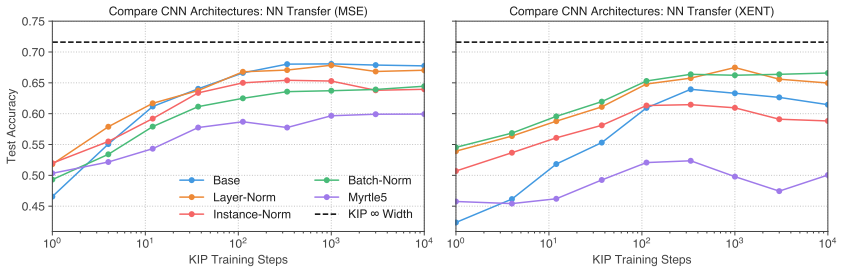

Figure 2: **Out-of-distribution neural network variations.** KIP ConvNet images (trained with fixed labels) are tested on variations of the ConvNet neural network, including those which have various normalization layers (layer, instance, batch). A similar architecture to ConvNet, the Myrtle5 architecture (without normalization layers) Shankar et al. [2020], which differs from the ConvNet architecture by having an additional convolutional layer at the bottom and a global average pooling that replaces the final local average pooling at the top, is also tested. Finally, mean-square error is compared with cross-entropy loss (left versus right).

KIP performs well at test time when using any of the three kernels, whereas datasets trained using a single kernel have a performance drop when using a different kernel.

3.3 Neural Network Transfer

In this section, we study how our distilled datasets optimized using KIP and LS transfer to the setting of finite-width neural networks. The main results are shown in Table 2. Since the datasets are optimized for kernel ridge-regression and not for neural network training itself, we expect some performance loss when transfering to finite-width networks. Remarkably, the drop due to this transfer is quite moderate or small and sometimes the transfer can even lead to gain in performance (see LS for SVHN dataset when using 1 or 10 images per class).

Overall, our transfer to finite-width networks outperform prior art based on DC/DSA [Zhao et al., 2021, Zhao and Bilen, 2021] in the 1 image per class setting for all the RGB datasets (SVHN, CIFAR-10, CIFAR-100). Moreover, for CIFAR-10, we outperform DC/DSA for the 10 and 50 images per class settings.

Figures 2 and 3 provide a closer look at KIP transfer changes under various settings. The first of these tracks how transfer performance changes when adding additional layers as function of the number of KIP training steps used. The normalization layers appear to harm performance, which can be anticipated from their absence in the KIP and LS optimization procedures. For Figure 3, we observe that as KIP training progresses, the downstream finite-width network's performance also improves in general. A notable exception is observed when learning the labels in KIP, where longer training steps lead to deterioration of information useful to training finite-widthneural networks. We also observe that as predicted by infinite-width theory Jacot et al. [2018], Lee et al. [2019], the overall gap between KIP or LS performance and finite-width neural network decreases as the width increases. While our best performing transfer is obtained with width 1024, Figure 3 (middle) suggest that even with modest width of 64, our transfer can outperform prior art of 60.6% by Zhao and Bilen [2021].

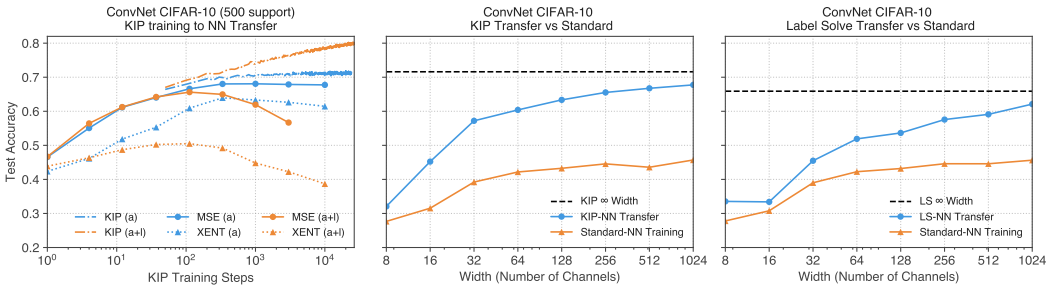

Figure 3: **Variations for neural network transfer.** *Left:* Plot of transfer performance as a function of KIP training steps across various train settings. Here (a) denotes augmentations used during KIP training and (a+l) denotes that additionally the labels were learned. MSE and XENT denote mean-square-error and cross entropy loss for the neural network, where for (a+l), the labels for the neural network are the argmax of the learned labels. *Middle:* Exploring the effect of width on transferability of vanilla KIP data. *Right:* The effect of width on the transferability of label solved data.

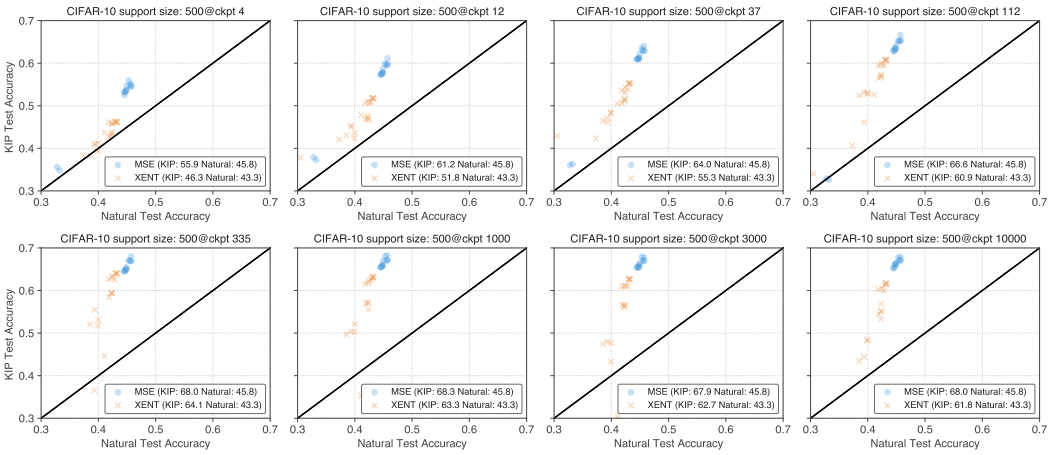

Figure 4: **Hyperparameter robustness.** In the above, 500 KIP images across eight different checkpoints are used to train the ConvNet neural network. Each point in each plot is a neural network training with a different hyperparameter, and its location records the final test accuracy when training on natural images versus the KIP images obtained from initializing from such images. For both MSE and cross entropy loss, KIP images consistently exceed natural images across many hyperparameters.

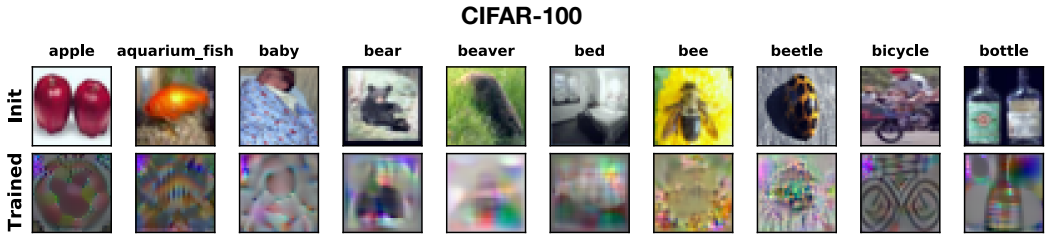

Figure 5: **Examples of learned images.** Images are initialized from natural images in the top row (`Init`) and converge to images in the bottom row (`Trained`). Experimental details in §A.

Finally, in Figure 4, we investigate the performance of KIP images over the course of training of a single run, as compared to natural images, over a range of hyperparameters. We find the KIP images's outperformance of natural images is consistent across hyperparameters and checkpoints. This suggests that our KIP images may also be effective for accelerated hyperparameter search, an application of dataset distillation explored in Zhao et al. [2021], Zhao and Bilen [2021].

Altogether, we find our neural network training results encouraging. First, it validates the applicability of infinite-width methods to the setting of finite width [Huang and Yau, 2020, Dyer and Gur-Ari, 2020, Andreassen and Dyer, 2020, Yaida, 2020, Lee et al., 2020]. Second, we find some of the transfer results quite surprising, including efficacy of label solve and the use of cross-entropy for certain settings (see §A for the full details).

4 Understanding KIP Images and Labels

We provide a brief analysis of the images and labels learned from KIP, with some additional analysis provided in the Appendix. We discover some surprising results:

Visual Analysis. A visual inspection of our learned data leads to rich and intriguing observations in terms of interpretability. Figure 5 shows examples of KIP learned images from CIFAR-100. The resulting images often incorporate features associated with the respective class and can strongly deviate from the starting point (for example, classes `apple` and `bottle`, where initial image containing two objects turned into a distilled image containing a single object). Note that while our images appear less interpretable to human eye than the ones produced by Nguyen et al. [2021] (Figure 2), they lead to a dramatic increase in performance over prior state of the art (Table 1). Investigating and quantifying aspects that make these images generalize so well is a promising avenue for future work. We show examples from other datasets in Figure 9.

In Figure 6, we compare MNIST KIP data learned with and without label learning. For the latter case when images and labels optimized jointly, images become less interpretable, while labels become more informative, encoding richer information than simple one-hot class affiliation. This consistently leads to superior KRR results, but appears to not be leveraged as efficiently in the NN transfer setting (Table 2). Experimental details can be found in §A.

Dimensional Analysis. We study the intrinsic dimension (ID) of KIP images and find that they tend to grow. Our method of estimating the ID of a dataset follows [Facco et al., 2017], which uses the two nearest-neighbor distances among points in the dataset to reduce the estimation to a linear regression problem. Figure 7 shows that the ID increasing for the learned KIP images as a function of the training step across a variety of configurations (training with or without augmentations/label learning) and datasets. One might expect that a distillation procedure should decrease dimensionality. On the other hand, Ansuini et al. [2019] showed that ID increases in the earlier layers of a trained neural network. It remains to be understood if this latter observation has any relationship with our increased ID. Note that ZCA preprocessing, which played an important role for getting the best performance for our RGB datasets, increases dimensionality of the underlying data (see the Appendix).

Spectral Analysis. Another observation that distinguishes KIP images from natural images concerns how their spectral components contribute to their performance. In Figure 8, we spot how different spectral bands of KIP learned images affect test performance as compared to their initial natural

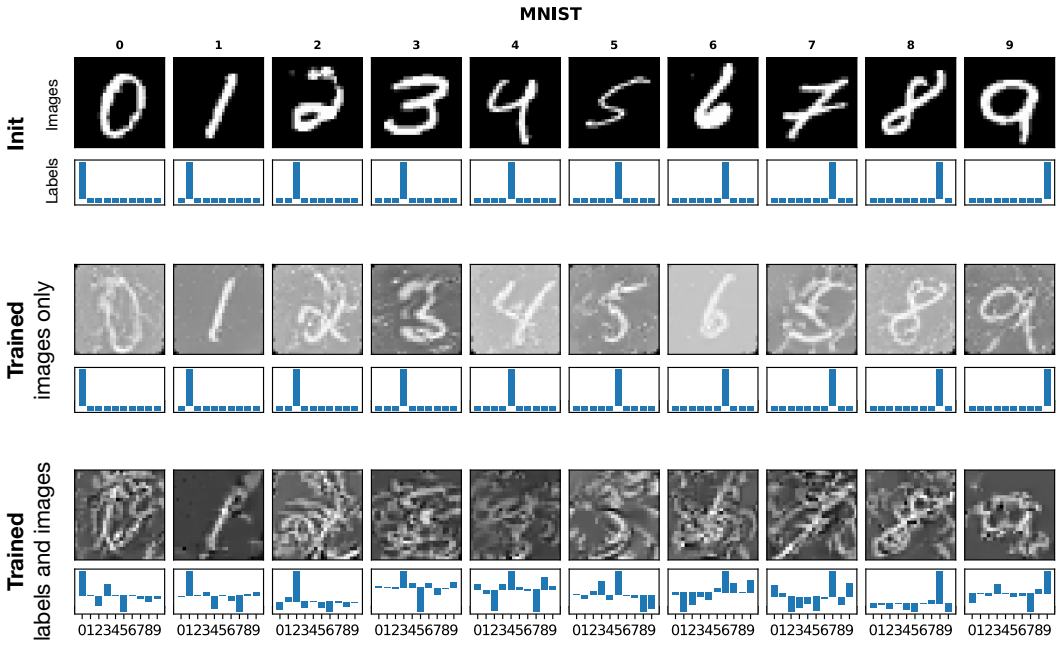

Figure 6: **Dataset distillation with trainable and non-trainable labels.** *Top row:* initialization of support images and labels. *Middle row:* trained images if labels remain fixed. *Bottom row:* trained images and labels, jointly optimized.

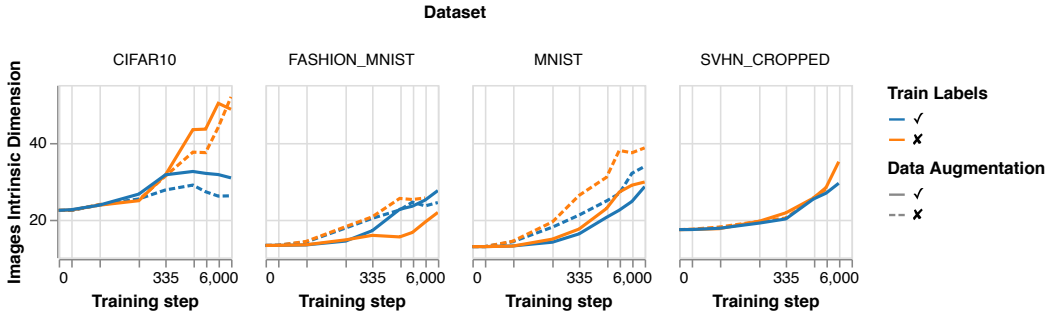

Figure 7: **Intrinsic dimension of learned datasets grows during training.** As training progresses (left to right), intrinsic dimension of the learned dataset grows, indicating that training transforms the data manifold in a non-trivial way. See Figures 9 and 6 for visual examples of learned images, and Figures 10 and 11 for similar observations using other dimensionality metrics and/or settings. All images here have standard preprocessing. Full experimental details in §A.

images. Here, we use the FC2, Conv-Vec8, and ConvNet architectures. We note that for natural images (light bars), most of their performance is captured in the top 20% of eigenvalues. For KIP images, the performance is either more evenly distributed across the bands (FC and Conv-Vec8) or else is skewed towards the tail (ConvNet).

5 Related Work

Dataset distillation was first studied in Wang et al. [2018]. The works Sucholutsky and Schonlau [2019], Bohdal et al. [2020] build upon Wang et al. [2018] by distilling labels. Zhao et al. [2021] proposes condensing a training set by harnessing a gradient matching condition Wang et al. [2018]. Zhao and Bilen [2021] takes this idea further by applying a suitable augmentation strategy. Note

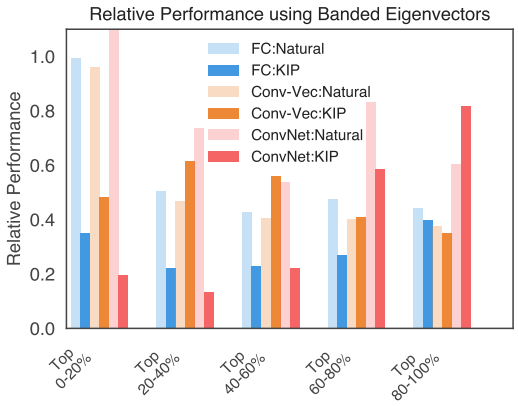

Figure 8: **Spectral contribution to test accuracy shifts to the tail.** By setting the ridge-parameter to zero in kernel-ridge regression and replacing $K_{X_s X_s}^{-1}$ by the spectral projection onto various eigenspaces, we can explore how different spectral bands affect test accuracy. We plot the relative change in test accuracy using contiguous bands of 20% of the eigenvalues.

that while Zhao and Bilen [2021] is limited in augmentation expressiveness (they have to a single augmentation per training iteration), we can sample augmentations independently per image in our target set per train step. Interestingly enough, in most instances, we do not observe a benefit from using augmentations. Our work together with [Nguyen et al., 2021] is, to the best of our knowledge, the only works using kernel-based methods for dataset distillation on image classification datasets.

Our use of kernels stems from the correspondence betwee infinitely-wide neural networks and kernel methods [Neal, 1994, Lee et al., 2018, Matthews et al., 2018, Jacot et al., 2018, Novak et al., 2019, Garriga-Alonso et al., 2019, Arora et al., 2019]. These correspondences underlie the transferability of our results in the KRR setting to the training of neural networks.

6 Conclusion

We performed an extensive study of dataset distillation using the KIP and LS algorithms applied to convolutional architectures, obtaining SOTA results on a variety of image classification datasets. In some cases, our learned datasets were more effective than a natural dataset one hundred times larger in size. There are many interesting future directions to our work:

First, the understanding of how various resources (e.g. data, parameter count, compute) scale when optimizing for neural network performance has acquired a significant importance as machine learning models continue to stretch computational limits [Hestness et al., 2017, Kaplan et al., 2020]. Developing our understanding of how to harness smaller, yet more useful representations data would aid in such endeavors. In particular, it would be especially interesting to explore how well datasets can be compressed as they scale up in size.

Second, LS and KIP with label learning shows that optimizing labels is a very powerful tool for dataset distillation. On the other hand, the strategy of optimizing labels appears to be under-explored, as there are relatively few representations of labels that are commonly used in practice (typically one-hot labels or some mild softening of them). The labels we obtain are quite far away from standard, interpretable labels and we feel their effectiveness deserves further study.

Third, the use of regularized ZCA preprocessing greatly enhances performance, both for KRR and neural network transfer. This contrasts with standard preprocessing, in which the neural network transfer often incurs a much larger drop in performance. This indicates that there is a nontrivial relationship between features that are performant for kernels those that are performant for neural networks, one that does not readily arise out of the infinite-width correspondence.

To perform our experiments, we developed a novel, distributed metalearning framework that leverages hundreds of GPUs per training run. Given the enormous scale of such experiments, we plan to release our learned datasets as a benefit to the research community.

References

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Flexible dataset distillation: Learn labels instead of images. *arXiv preprint arXiv:2006.08572*, 2020.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2021.

Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, 2021.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features, 2019.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Katherine L Hermann and Andrew K Lampinen. What shapes feature representations? exploring datasets, architectures, and training. *arXiv preprint arXiv:2006.12433*, 2020.

Zalán Borsos, Mojmír Mutnỳ, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *arXiv preprint arXiv:2006.03875*, 2020.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021.

Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019.

Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8141–8150. Curran Associates, Inc., 2019.

Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S Du, Wei Hu, Ruslan Salakhutdinov, and Sanjeev Arora. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.

Vaishaal Shankar, Alex Chengyu Fang, Wenshuo Guo, Sara Fridovich-Keil, Ludwig Schmidt, Jonathan Ragan-Kelley, and Benjamin Recht. Neural kernels without tangents. In *International Conference on Machine Learning*, 2020.

Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkl8sJBYvH.

Jaehoon Lee, Samuel S Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. In *Advances in Neural Information Processing Systems*, 2020.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL https://github.com/google/neural-tangents.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, 2019.

Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. In *International Conference on Machine Learning*, 2020.

Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1gFvANKDS.

Anders Andreassen and Ethan Dyer. Asymptotics of wide convolutional neural networks. *arxiv preprint arXiv:2008.08675*, 2020.

Sho Yaida. Non-Gaussian processes and neural networks at finite widths. In *Mathematical and Scientific Machine Learning Conference*, 2020.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):1–8, 2017.

A. Ansuini, A. Laio, J. Macke, and D. Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *NeurIPS*, 2019.

Ilia Sucholutsky and Matthias Schonlau. Soft-label dataset distillation and text dataset distillation. *arXiv preprint arXiv:1910.02551*, 2019.

Radford M. Neal. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Sam Schoenholz, Jeffrey Pennington, and Jascha Sohl-dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory F. Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *CoRR*, abs/1712.00409, 2017. URL http://arxiv.org/abs/1712.00409.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] (Links provided in Contributions at end of introduction)
   (b) Did you describe the limitations of your work? [Yes] (See various ablation studies in Section 3 which show how performance varies when various settings change.)
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] (No obvious or direct ethical implications of our work to the best of our knowledge.)
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] (Forthcoming in the Appendix)
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] (We did report error bars for most of our results, namely those in Table 1. In Table 2, we did our neural network runs with a single random seed. This will be easy for us to redress and we will do so during the rebuttal.)
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] (Details alluded to in the Contributions section, will be further detailed in Appendix)
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]