Dataset Meta-Learning with Kernel Ridge Regression

Kernel Inducing Points (KIP), Label Solve, Infinitely-wide Neural Networks

Jaehoon Lee (jaehlee@google.com)

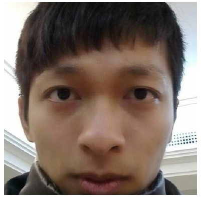
ML Collective

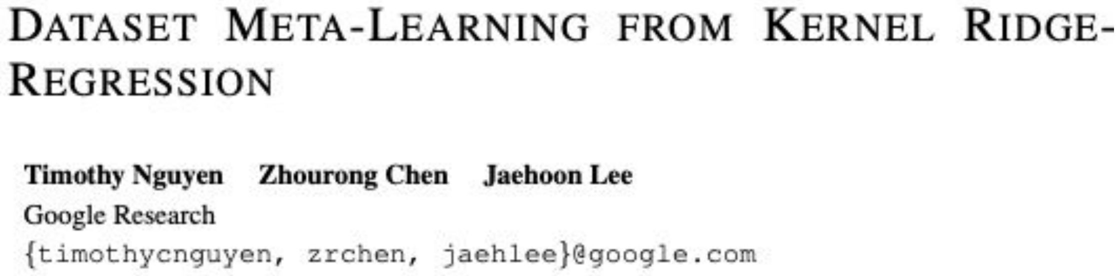
Deep Learning: Classics and Trends

July 30, 2021

Based on

arXiv: 2011.00050
ICLR 2021

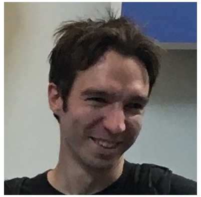


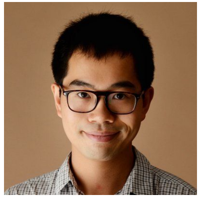


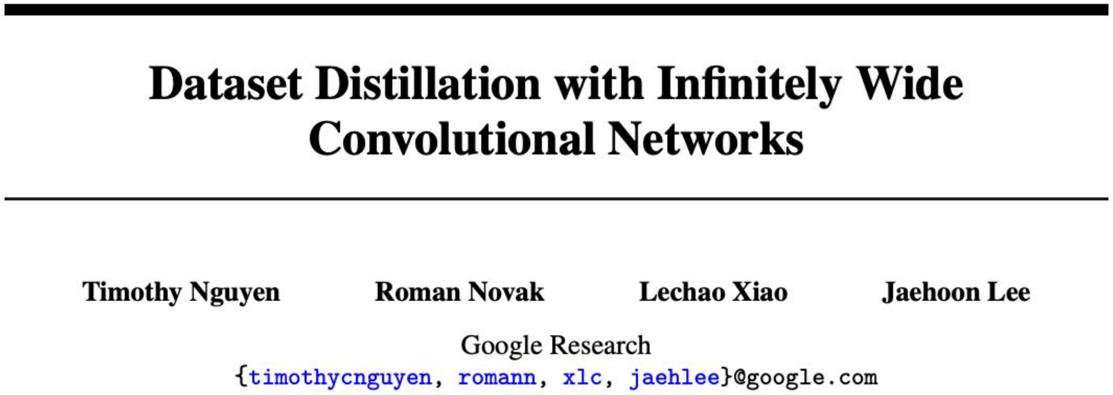
DATASET META-LEARNING FROM KERNEL RIDGE-REGRESSION

Timothy Nguyen Zhourong Chen Jaehoon Lee
Google Research
{timothycnguyen, zrchen, jaehlee}@google.com

arXiv: 2107.13034





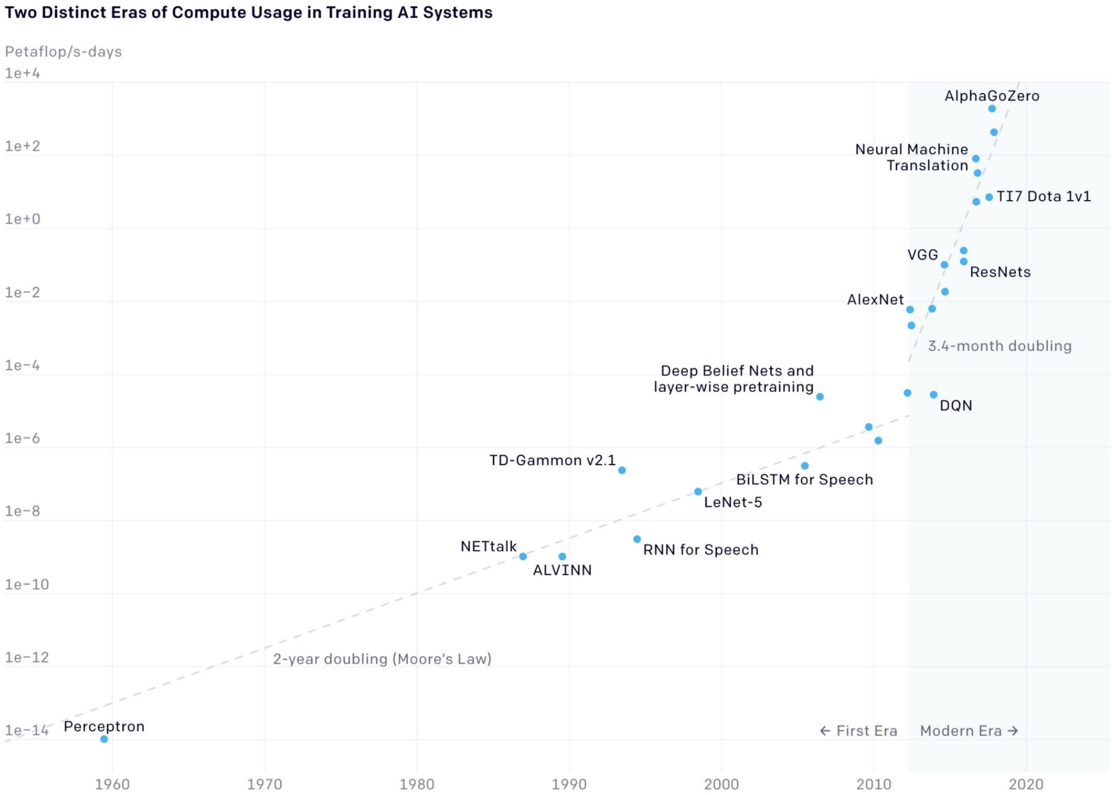


Dataset Distillation with Infinitely Wide Convolutional Networks

Timothy Nguyen Roman Novak Lechao Xiao Jaehoon Lee
Google Research
{timothycnguyen, romann, xlc, jaehlee}@google.com

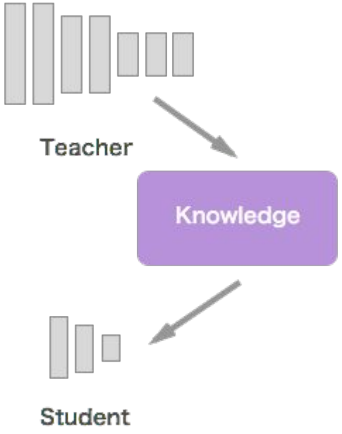
Compute costs in AI

Source: OpenAI Blog, AI and Compute

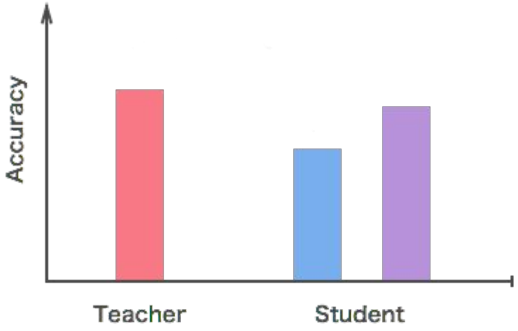


Distillation in Deep Learning

- Hinton et al. *Distilling the Knowledge in a Neural Network*, 2014.
- Transfer learned output probabilities from a large (possibly ensembled) model to a smaller one



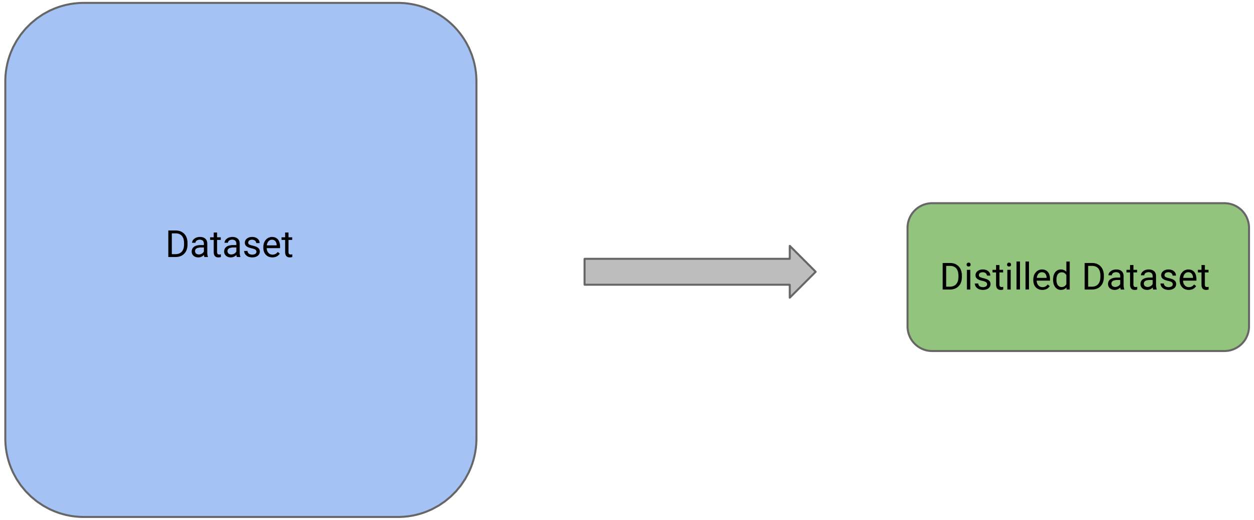
$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$



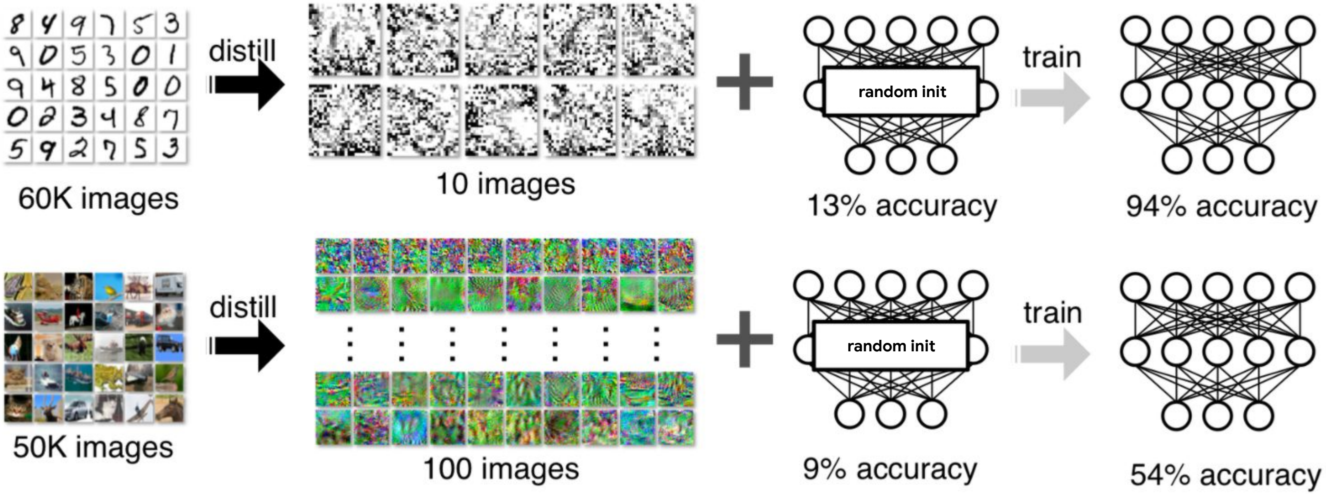
Source: Knowledge Distillation by U. Upadhyay

Dataset Distillation

- Given a dataset, construct a smaller dataset that performs nearly as well as the original
- Goal: **compress** knowledge of entire dataset into a few **synthetic** data points



Dataset Distillation



Dataset distillation on MNIST and CIFAR10

Source: Dataset Distillation Project Page

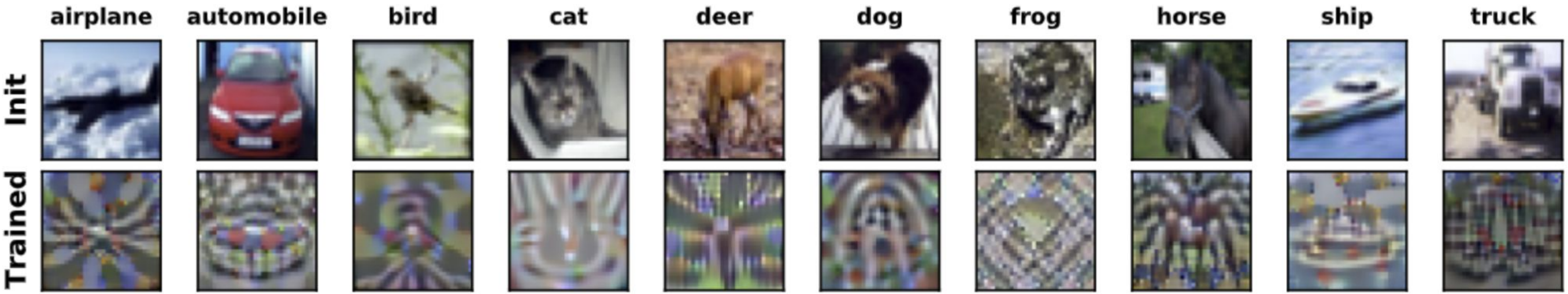
Why Distill Dataset?

- Space efficiency
  - Data storage
  - Especially for nonparametric methods like k-NN, kernel methods

- Compute efficiency
  - Accelerate training
  - faster hyperparameter, architecture search

- Representational efficiency
  - Learn a few prototypical examples instead of requiring many instances

Sneak preview: distilling CIFAR-10



- 50,000 train images → 10 data points (1 img/cls) : **Test set accuracy 64%**
- outperforms ~128x more natural images

Prior Work I

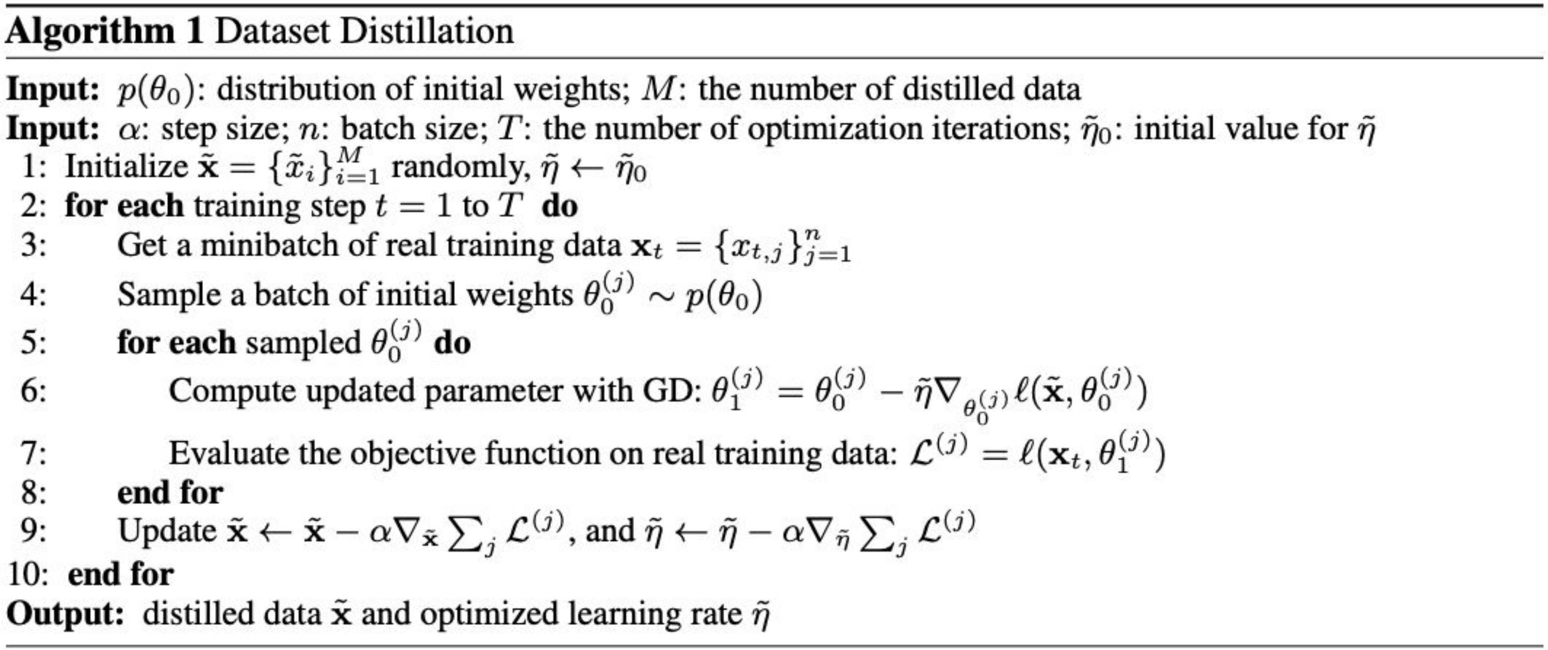
- Simple baselines (model independent, subset selection)
  - Random subset of natural images
  - Coresets / instance selection
  - k-median, k-means clustering

- Shortcomings
  - Rely on heuristics and existence of representative sample
  - Not guarantee for downstream task (validation loss)

Prior Work II

- Training set synthesis
  - Dataset distillation (DD): Wang et al., 2018
  - Soft-label dataset distillation (SLDD): Sucholutsky & Schonlau 2019
  - Label Distillation (LD): Bohdal et al., 2020
  - Dataset Condensation (DC): Zhao et al., 2021
  - Differentiable Siamese Augmentation (DSA): Zhao & Bilen 2021

Prior Work: Dataset Distillation (DD)



Algorithm 1 Dataset Distillation

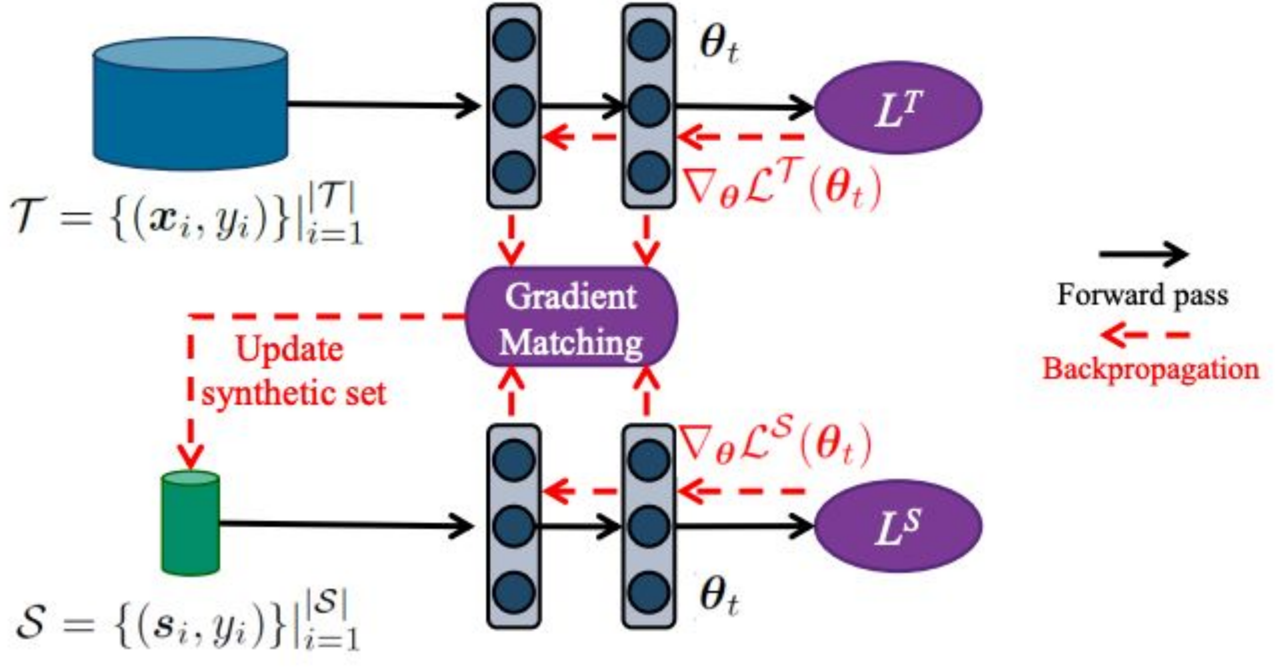
```
Input: $p(\theta_0)$: distribution of initial weights; $M$: the number of distilled data
Input: $\alpha$: step size; $n$: batch size; $T$: the number of optimization iterations; $\tilde{\eta}_0$: initial value for $\tilde{\eta}$
Initialize $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M}$ randomly, $\tilde{\eta} \leftarrow \tilde{\eta}_0$
for each training step $t = 1$ to $T$ do
    Get a minibatch of real training data $\mathbf{x}_t = \{x_{t,j}\}_{j=1}^{n}$
    Sample a batch of initial weights $\theta_0^{(j)} \sim p(\theta_0)$
    for each sampled $\theta_0^{(j)}$ do
        Compute updated parameter with GD: $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta}\nabla_{\theta_0^{(j)}}\ell(\tilde{\mathbf{x}}, \theta_0^{(j)})$
        Evaluate the objective function on real training data: $\mathcal{L}^{(j)} = \ell(\mathbf{x}_t, \theta_1^{(j)})$
    end for
    Update $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \alpha\nabla_{\tilde{\mathbf{x}}}\sum_j \mathcal{L}^{(j)}$, and $\tilde{\eta} \leftarrow \tilde{\eta} - \alpha\nabla_{\tilde{\eta}}\sum_j \mathcal{L}^{(j)}$
end for
Output: distilled data $\tilde{\mathbf{x}}$ and optimized learning rate $\tilde{\eta}$
```

Wang et al., Dataset Distillation, arXiv 2018

Prior Work: Dataset Condensation (DC)

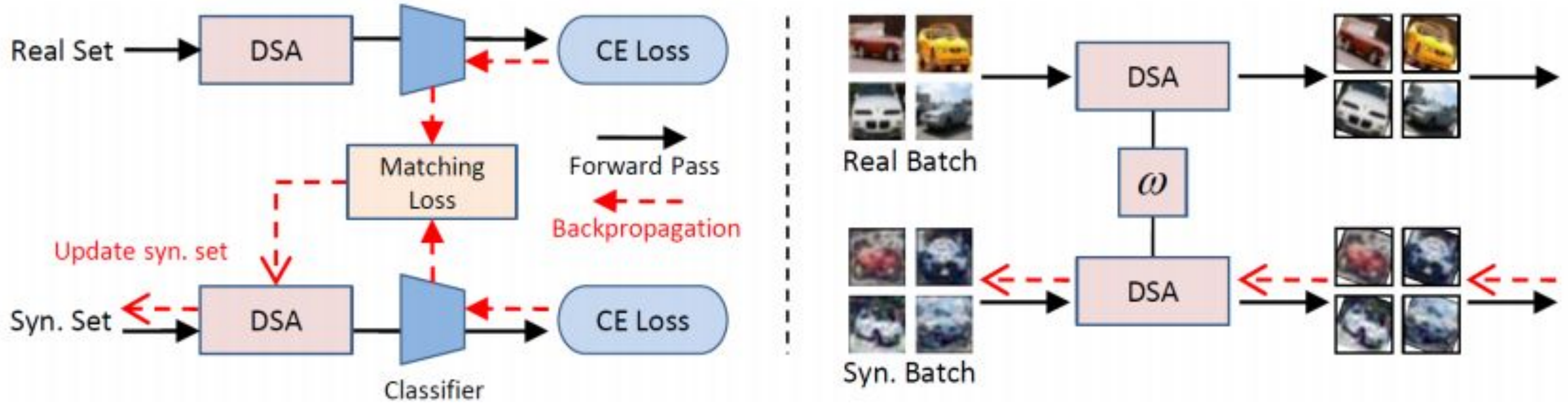
Large Training Set (target)

Small Synthetic Set (support)


$\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}|_{i=1}^{|\mathcal{T}|}$
$\boldsymbol{\theta}_t$
L^T
$\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}_t)$
Gradient Matching
Update synthetic set
$\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_t)$
L^S
$\boldsymbol{\theta}_t$
$\mathcal{S} = \{(\boldsymbol{s}_i, y_i)\}|_{i=1}^{|\mathcal{S}|}$
Forward pass
Backpropagation

Zhao et al., Dataset Condensation with Gradient Matching, ICLR 2021

Prior Work: Dataset Condensation with DSA (DSA)



Zhao & Bilen, Dataset Condensation with Differentiable Siamese Augmentation, ICML 2021

Infinitely Wide Neural Networks

- In the limit of infinite width, neural networks become tractable:

NN with MSE loss ⟷ kernel ridge-regression with corresponding neural kernel

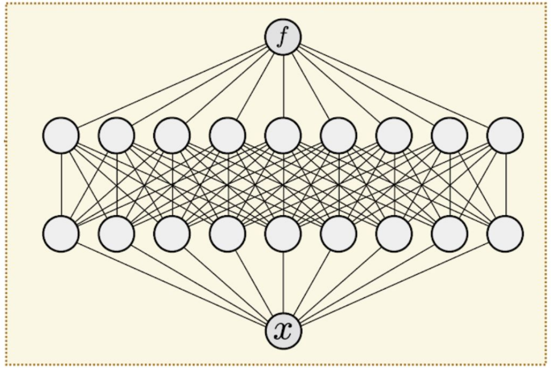


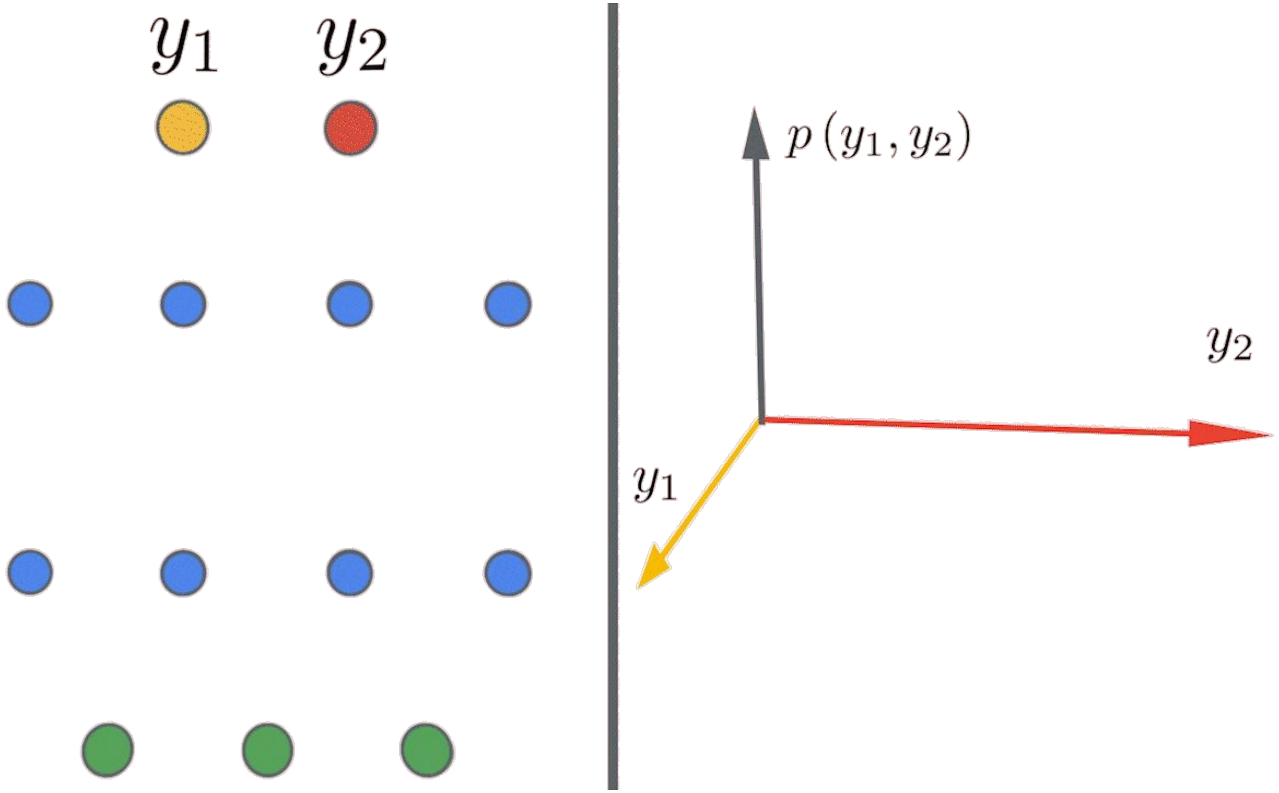
Image credit: Roman Novak

$$\xrightarrow[n \to \infty]{}$$

- Neural Network Gaussian Process (Neal 1996, Lee & Bahri et al. 2018, Matthews et al. 2018)
- Neural Tangent Kernel (Jacot et al. 2018)

Simplicity in Large Numbers: Neural Networks

Image Credit: Tom Small



Neural Tangents python software library!

github.com/google/neural-tangents

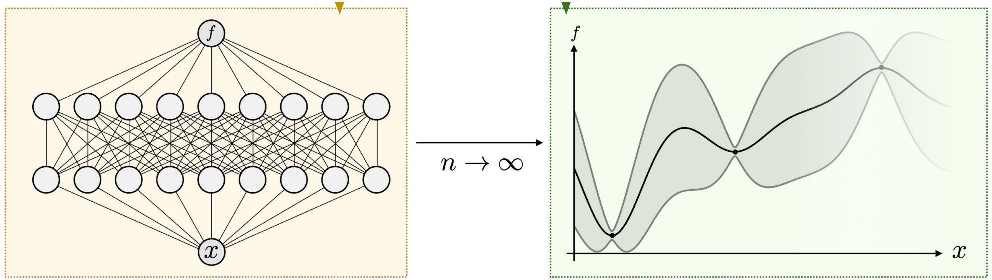
[Novak & Xiao et al., ICLR 2020]

```
pip install neural-tangents
```

```
import neural_tangents as nt

layers = []
layers += [nt.stax.Conv(512, (3, 3), W_std=2.0**0.5, b_std=0.1), nt.stax.Relu()] * 8
layers += [nt.stax.GlobalAvgPool(), nt.stax.Dense(10, W_std=2.0**0.5, b_std=0.1)]

init_fn, apply_fn, kernel_fn = nt.stax.serial(*layers)
```



$\mathcal{K}(\cdot,\cdot)$ (NNGP)

$\Theta(\cdot,\cdot)$ (NTK)

CPU / GPU / TPU support!

Distill Data for Kernel Ridge-Regression (KRR)

- Closed-form solution: meta-learning becomes first-order optimization

- Given $D = (X_s, y_s)$, then

$$A_D(X_t) = K_{X_t X_s}(K_{X_s X_s} + \lambda I)^{-1} y_s$$

where

$$K_{UV} = (K(u, v))_{u \in U, v \in V}.$$

- Expect learned D to transfer well to corresponding finite-width network for K a neural kernel

Kernel Inducing Points (KIP)

- Given $D_{\mathrm{train}} \subset \mathcal{P}$
- Sample targets (X_t, y_t) from D_{train}
- Perform gradient update to support images X_s (and possibly labels y_s)

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

- Output: $D = (X_s, y_s)$ a distilled version of D_{train}

Kernel Inducing Points Variations

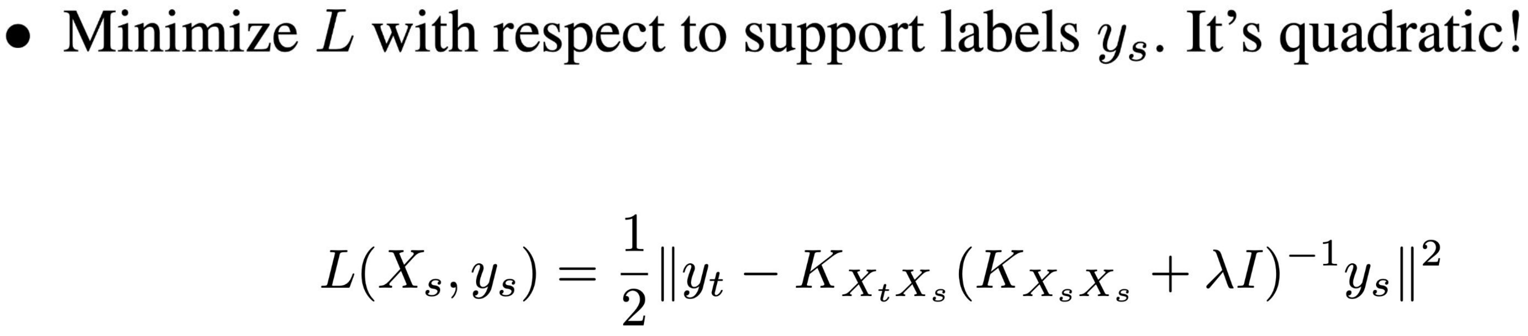
- Learn or fix labels

- Augment targets
    - can be very effective, especially when learning w/ labels

- Randomly sample kernel from a family of kernels
    - improves generalization across corresponding kernels / NNs

- Initialize inputs from natural images or random noise

Label Solve (LS)

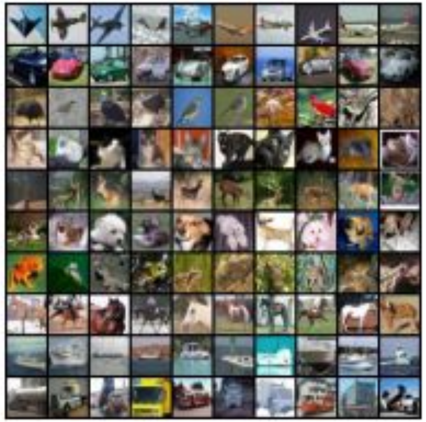


- Minimize L with respect to support labels y_s. It's quadratic!

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

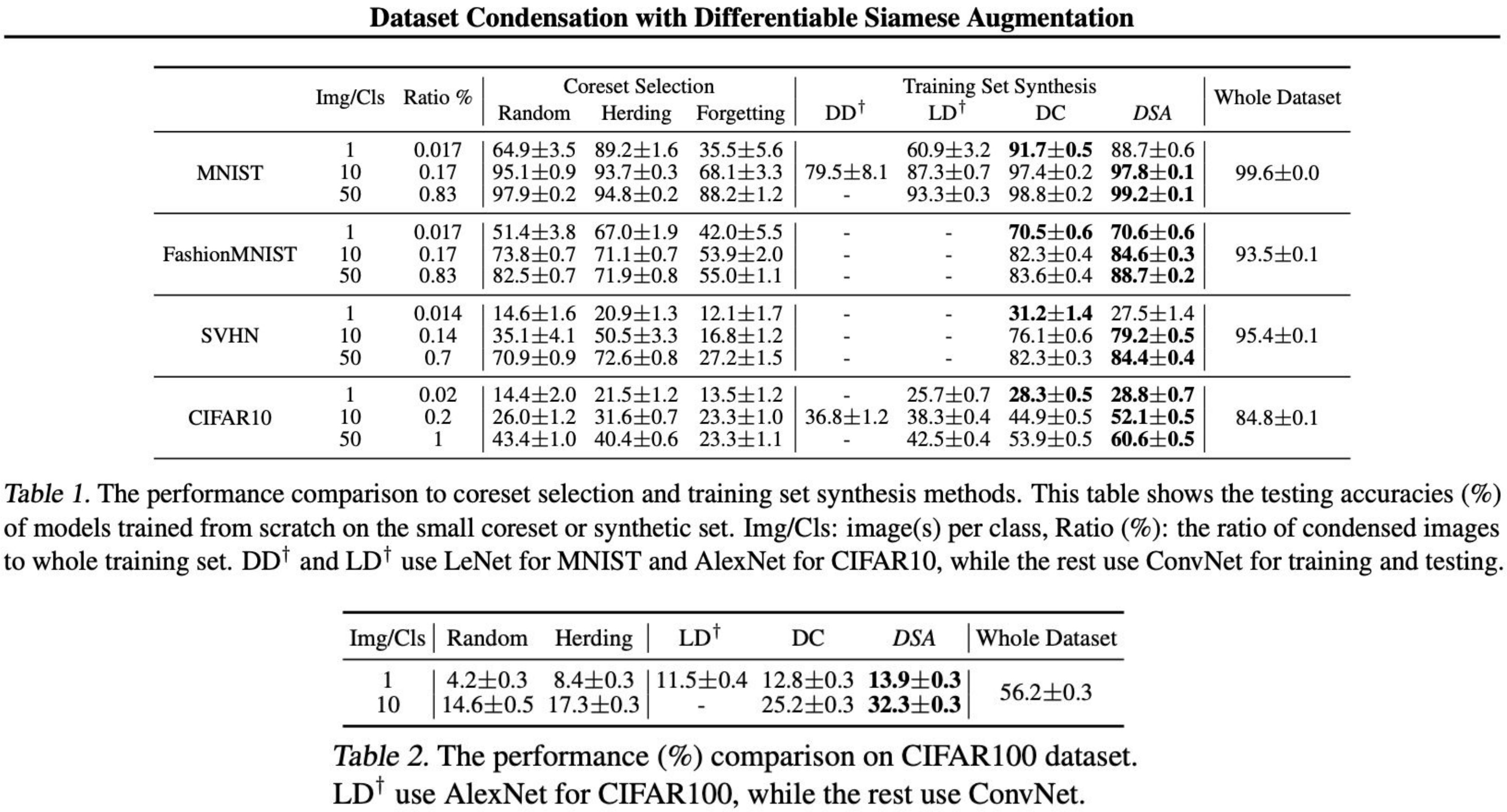
- No need for iterative process!

Experiments: Datasets



- Common Benchmarks: MNIST, Fashion-MNIST, SVHN, CIFAR-10/100
- Distilling to: 1, 10, 50 images/class (except for CIFAR-100)

Prior Work: DC & DSA



Dataset Condensation with Differentiable Siamese Augmentation

	Img/Cls	Ratio %	Coreset Selection			Training Set Synthesis				Whole Dataset
			Random	Herding	Forgetting	DD†	LD†	DC	*DSA*	
MNIST	1	0.017	64.9±3.5	89.2±1.6	35.5±5.6		60.9±3.2	**91.7±0.5**	88.7±0.6	99.6±0.0
	10	0.17	95.1±0.9	93.7±0.3	68.1±3.3	79.5±8.1	87.3±0.7	97.4±0.2	**97.8±0.1**	
	50	0.83	97.9±0.2	94.8±0.2	88.2±1.2	-	93.3±0.3	98.8±0.2	**99.2±0.1**	
FashionMNIST	1	0.017	51.4±3.8	67.0±1.9	42.0±5.5	-	-	**70.5±0.6**	**70.6±0.6**	93.5±0.1
	10	0.17	73.8±0.7	71.1±0.7	53.9±2.0	-	-	82.3±0.4	**84.6±0.3**	
	50	0.83	82.5±0.7	71.9±0.8	55.0±1.1	-	-	83.6±0.4	**88.7±0.2**	
SVHN	1	0.014	14.6±1.6	20.9±1.3	12.1±1.7	-	-	**31.2±1.4**	27.5±1.4	95.4±0.1
	10	0.14	35.1±4.1	50.5±3.3	16.8±1.2	-	-	76.1±0.6	**79.2±0.5**	
	50	0.7	70.9±0.9	72.6±0.8	27.2±1.5	-	-	82.3±0.3	**84.4±0.4**	
CIFAR10	1	0.02	14.4±2.0	21.5±1.2	13.5±1.2	-	25.7±0.7	**28.3±0.5**	**28.8±0.7**	84.8±0.1
	10	0.2	26.0±1.2	31.6±0.7	23.3±1.0	36.8±1.2	38.3±0.4	44.9±0.5	**52.1±0.5**	
	50	1	43.4±1.0	40.4±0.6	23.3±1.1	-	42.5±0.4	53.9±0.5	**60.6±0.5**	

Table 1. The performance comparison to coreset selection and training set synthesis methods. This table shows the testing accuracies (%) of models trained from scratch on the small coreset or synthetic set. Img/Cls: image(s) per class, Ratio (%): the ratio of condensed images to whole training set. DD† and LD† use LeNet for MNIST and AlexNet for CIFAR10, while the rest use ConvNet for training and testing.

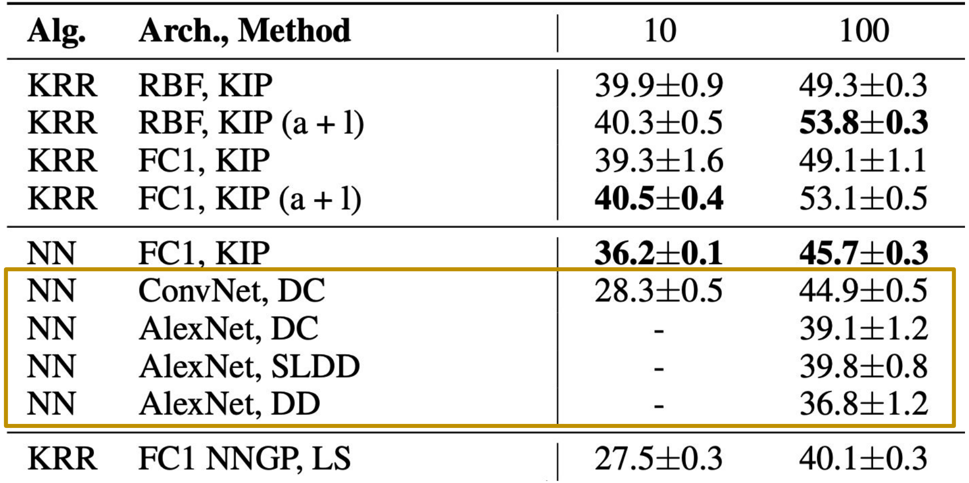
Img/Cls	Random	Herding	LD†	DC	*DSA*	Whole Dataset
1	4.2±0.3	8.4±0.3	11.5±0.4	12.8±0.3	**13.9±0.3**	56.2±0.3
10	14.6±0.5	17.3±0.3	-	25.2±0.3	**32.3±0.3**	

Table 2. The performance (%) comparison on CIFAR100 dataset. LD† use AlexNet for CIFAR100, while the rest use ConvNet.

Results using cheap kernels

- Classical kernels (RBF / Laplace) are cheap and only depend on dot products: $K(x, x')$ is a function of $x \cdot x$, $x \cdot x'$, $x' \cdot x'$

- Fully-Connected (FC) neural kernels are also dot product kernels



Alg.	Arch., Method	10	100
KRR	RBF, KIP	39.9±0.9	49.3±0.3
KRR	RBF, KIP (a + l)	40.3±0.5	**53.8±0.3**
KRR	FC1, KIP	39.3±1.6	49.1±1.1
KRR	FC1, KIP (a + l)	**40.5±0.4**	53.1±0.5
NN	FC1, KIP	**36.2±0.1**	**45.7±0.3**
NN	ConvNet, DC	28.3±0.5	44.9±0.5
NN	AlexNet, DC	-	39.1±1.2
NN	AlexNet, SLDD	-	39.8±0.8
NN	AlexNet, DD	-	36.8±1.2
KRR	FC1 NNGP, LS	27.5±0.3	40.1±0.3

Prior Work

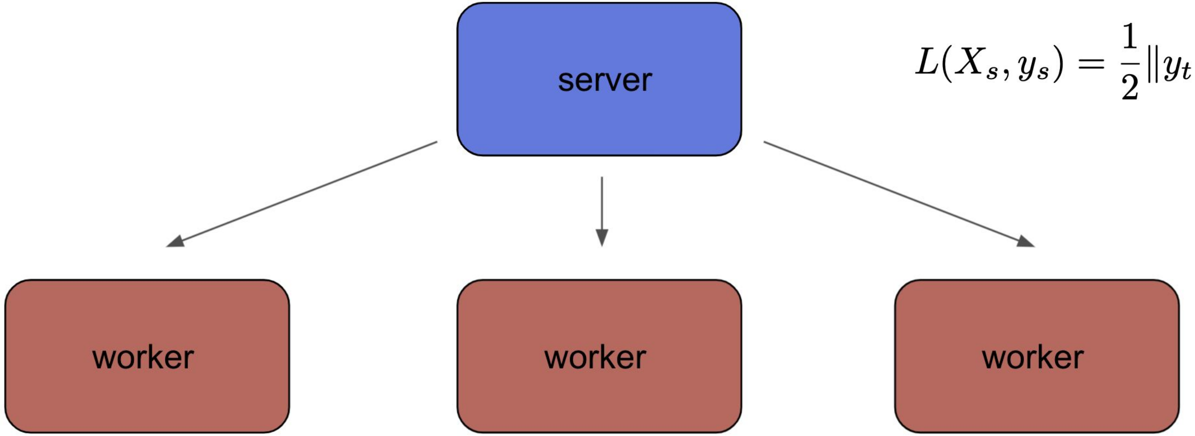
CIFAR-10 Distillation
1 Img/Cls,10 Img/Cls

Architecture → Dataset size ↓	**Fully-connected**	**CNNs**	**CNNs w/ pooling**
O(100)	🥱	🤗	🙂
O(10,000)	CIFAR10: **O(10) GPU-minutes**	CIFAR10: **O(1) GPU-hours**	CIFAR10: **O(1000) GPU-hours**
O(1,000,000)	🤔	😱	☠️

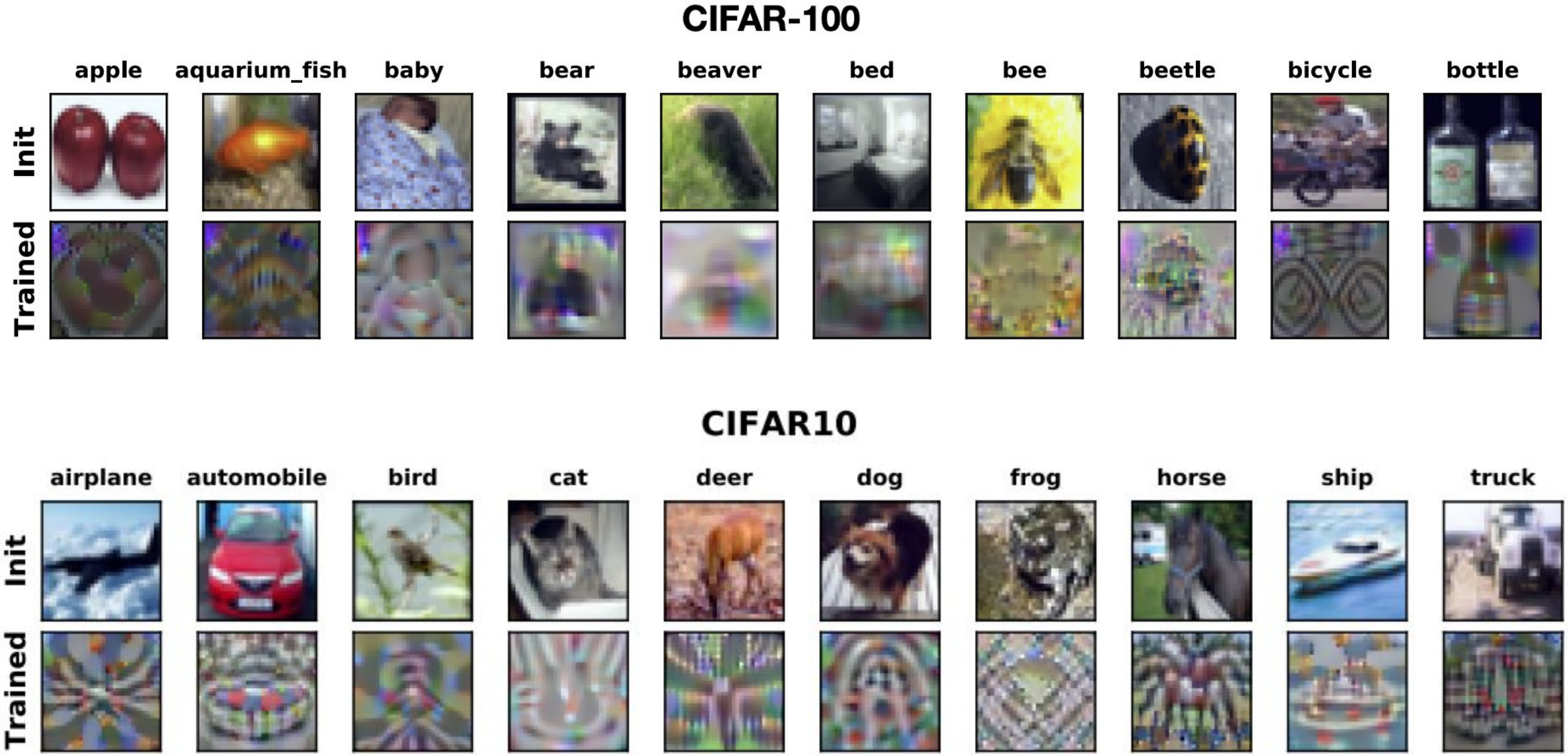
Slide credit: Roman Novak

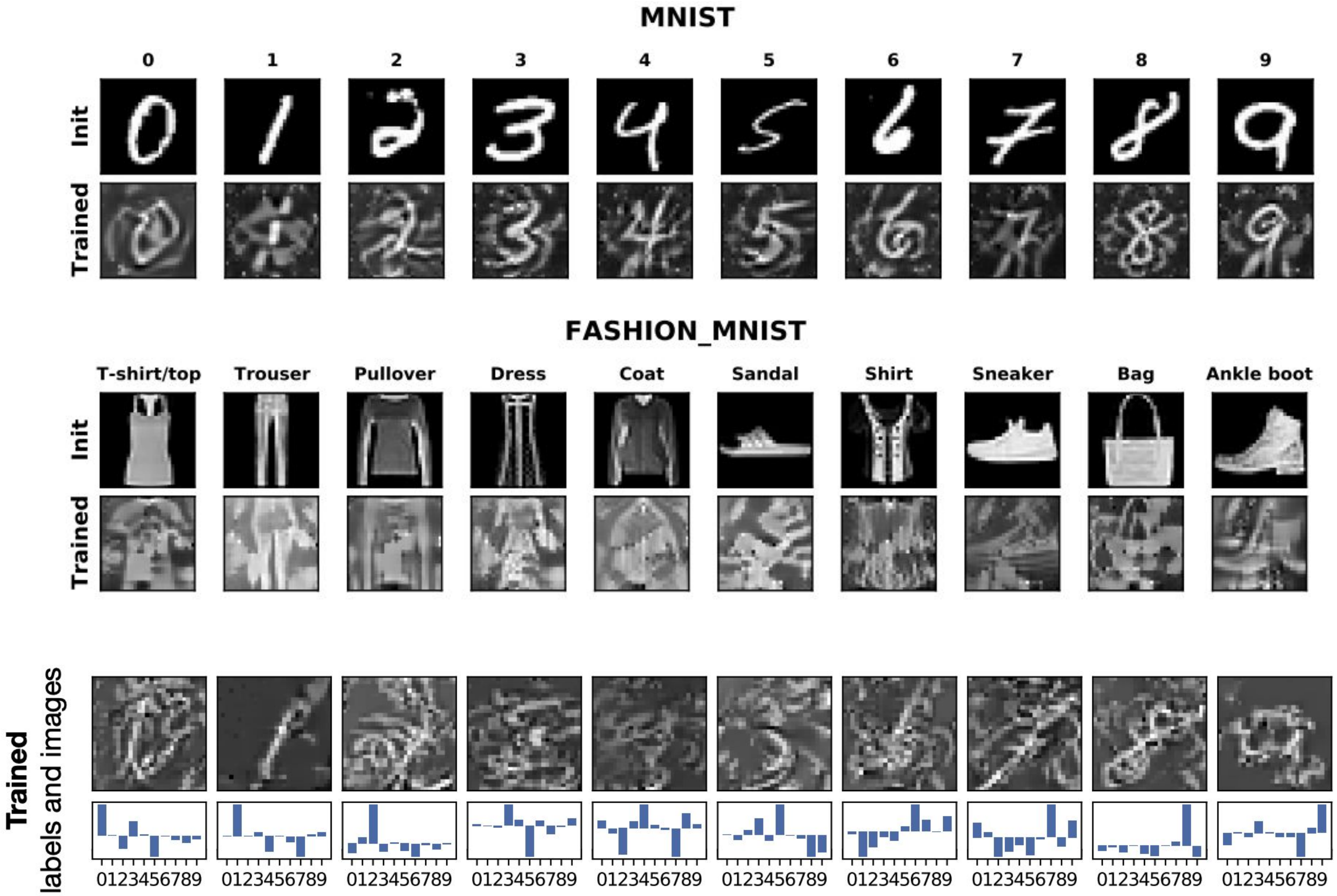
Large-scale Distributed Meta-Learning

- Convolutional kernels more powerful but compute intensive
- Client-Server model of distributed computation
  - Both kernel elements and gradients in distributed fashion
- Leverage 100s - 1000s freebie (preemptible, unused) GPUs

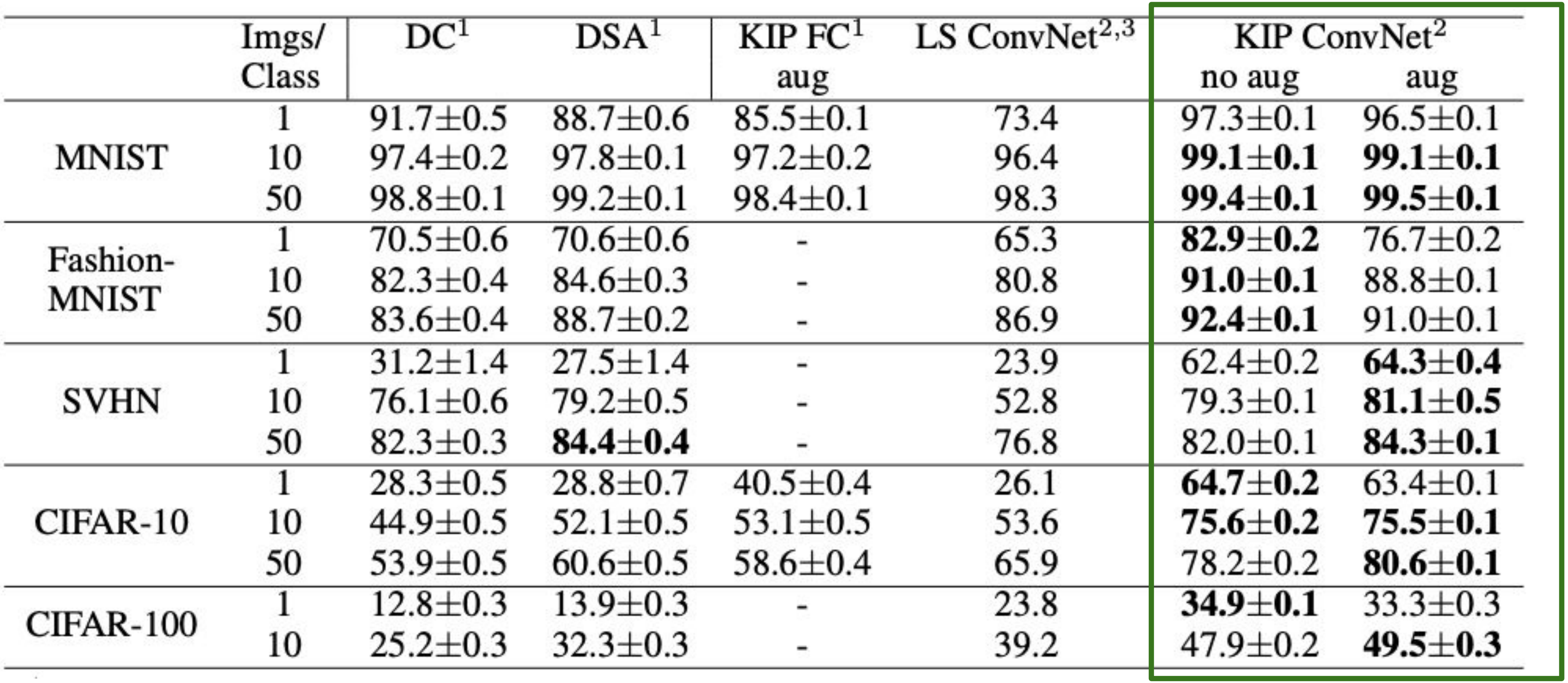


$$L(X_s, y_s) = \frac{1}{2}\|y_t - K_{X_t X_s}(K_{X_s X_s} + \lambda I)^{-1} y_s\|^2$$


CIFAR-100
apple
aquarium_fish
baby
bear
beaver
bed
bee
beetle
bicycle
bottle
Init
Trained
CIFAR10
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
Init
Trained


MNIST
0
1
2
3
4
5
6
7
8
9
Init
Trained
FASHION_MNIST
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
Init
Trained
Trained
labels and images
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789
0123456789

State of the Art on Dataset Distillation



	Imgs/ Class	DC[1]	DSA[1]	KIP FC[1]	LS ConvNet[2,3]	KIP ConvNet[2]	
				aug		no aug	aug
MNIST	1	91.7±0.5	88.7±0.6	85.5±0.1	73.4	97.3±0.1	96.5±0.1
	10	97.4±0.2	97.8±0.1	97.2±0.2	96.4	**99.1±0.1**	**99.1±0.1**
	50	98.8±0.1	99.2±0.1	98.4±0.1	98.3	**99.4±0.1**	**99.5±0.1**
Fashion-MNIST	1	70.5±0.6	70.6±0.6	-	65.3	**82.9±0.2**	76.7±0.2
	10	82.3±0.4	84.6±0.3	-	80.8	**91.0±0.1**	88.8±0.1
	50	83.6±0.4	88.7±0.2	-	86.9	**92.4±0.1**	91.0±0.1
SVHN	1	31.2±1.4	27.5±1.4	-	23.9	62.4±0.2	**64.3±0.4**
	10	76.1±0.6	79.2±0.5	-	52.8	79.3±0.1	**81.1±0.5**
	50	82.3±0.3	**84.4±0.4**	-	76.8	82.0±0.1	**84.3±0.1**
CIFAR-10	1	28.3±0.5	28.8±0.7	40.5±0.4	26.1	**64.7±0.2**	63.4±0.1
	10	44.9±0.5	52.1±0.5	53.1±0.5	53.6	**75.6±0.2**	**75.5±0.1**
	50	53.9±0.5	60.6±0.5	58.6±0.4	65.9	78.2±0.2	**80.6±0.1**
CIFAR-100	1	12.8±0.3	13.9±0.3	-	23.8	**34.9±0.1**	33.3±0.3
	10	25.2±0.3	32.3±0.3	-	39.2	47.9±0.2	**49.5±0.3**

Performance Transfers to Finite Neural Networks

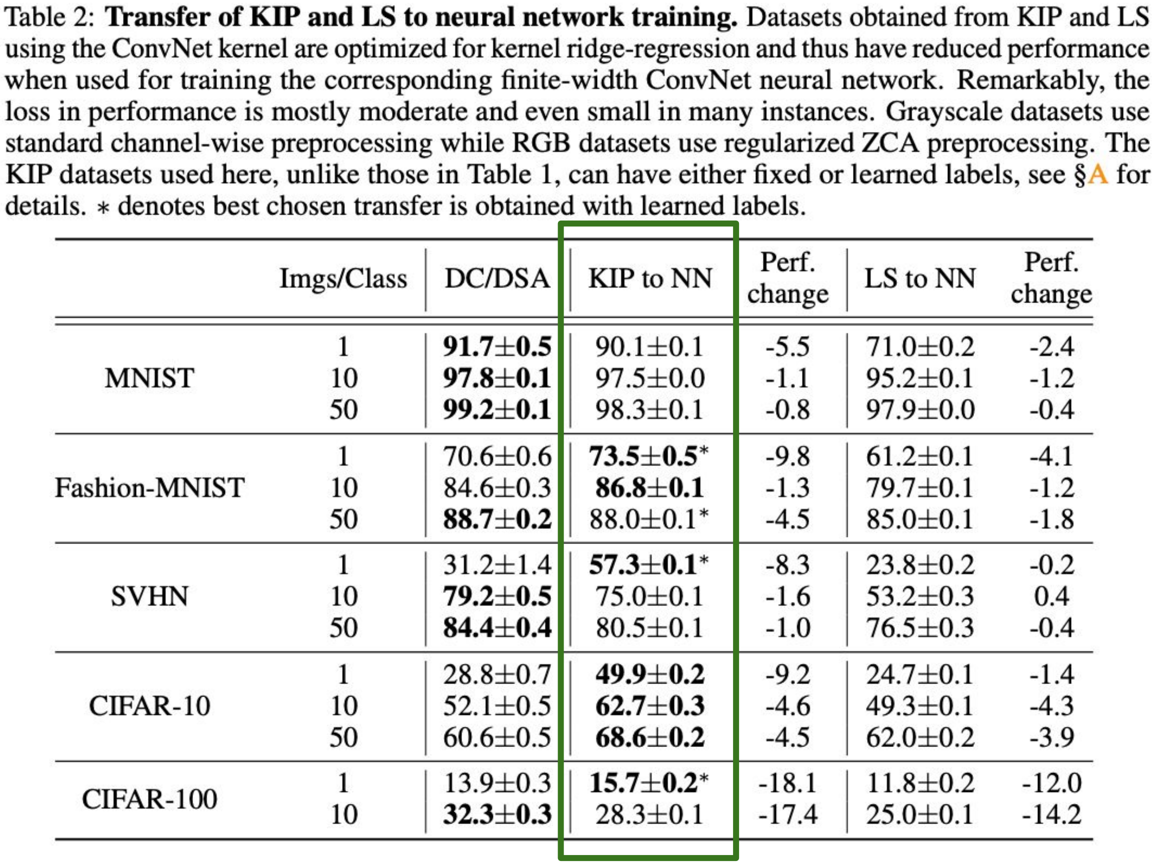


Table 2: **Transfer of KIP and LS to neural network training.** Datasets obtained from KIP and LS using the ConvNet kernel are optimized for kernel ridge-regression and thus have reduced performance when used for training the corresponding finite-width ConvNet neural network. Remarkably, the loss in performance is mostly moderate and even small in many instances. Grayscale datasets use standard channel-wise preprocessing while RGB datasets use regularized ZCA preprocessing. The KIP datasets used here, unlike those in Table 1, can have either fixed or learned labels, see §A for details. ∗ denotes best chosen transfer is obtained with learned labels.

	Imgs/Class	DC/DSA	KIP to NN	Perf. change	LS to NN	Perf. change
MNIST	1	**91.7±0.5**	90.1±0.1	-5.5	71.0±0.2	-2.4
	10	**97.8±0.1**	97.5±0.0	-1.1	95.2±0.1	-1.2
	50	**99.2±0.1**	98.3±0.1	-0.8	97.9±0.0	-0.4
Fashion-MNIST	1	70.6±0.6	**73.5±0.5***	-9.8	61.2±0.1	-4.1
	10	84.6±0.3	**86.8±0.1**	-1.3	79.7±0.1	-1.2
	50	**88.7±0.2**	88.0±0.1*	-4.5	85.0±0.1	-1.8
SVHN	1	31.2±1.4	**57.3±0.1***	-8.3	23.8±0.2	-0.2
	10	**79.2±0.5**	75.0±0.1	-1.6	53.2±0.3	0.4
	50	**84.4±0.4**	80.5±0.1	-1.0	76.5±0.3	-0.4
CIFAR-10	1	28.8±0.7	**49.9±0.2**	-9.2	24.7±0.1	-1.4
	10	52.1±0.5	**62.7±0.3**	-4.6	49.3±0.1	-4.3
	50	60.6±0.5	**68.6±0.2**	-4.5	62.0±0.2	-3.9
CIFAR-100	1	13.9±0.3	**15.7±0.2***	-18.1	11.8±0.2	-12.0
	10	**32.3±0.3**	28.3±0.1	-17.4	25.0±0.1	-14.2

Ablation study with CIFAR-10

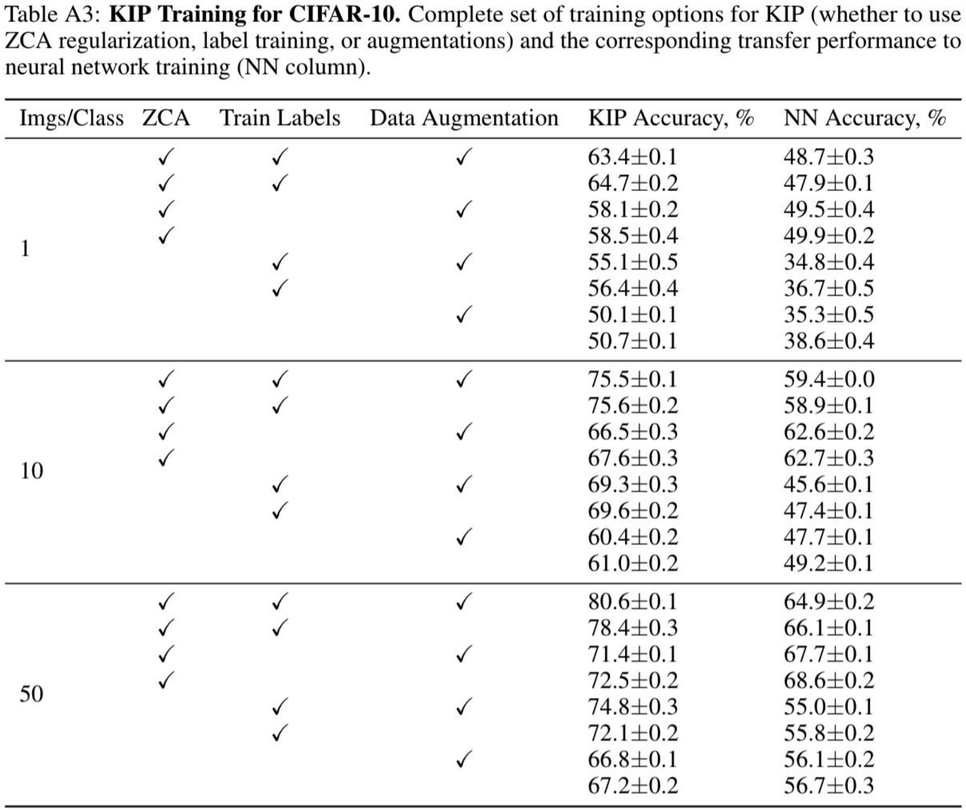


Table A3: **KIP Training for CIFAR-10.** Complete set of training options for KIP (whether to use ZCA regularization, label training, or augmentations) and the corresponding transfer performance to neural network training (NN column).

Imgs/Class	ZCA	Train Labels	Data Augmentation	KIP Accuracy, %	NN Accuracy, %
1	✓	✓	✓	63.4±0.1	48.7±0.3
	✓	✓		64.7±0.2	47.9±0.1
	✓		✓	58.1±0.2	49.5±0.4
	✓			58.5±0.4	49.9±0.2
		✓	✓	55.1±0.5	34.8±0.4
		✓		56.4±0.4	36.7±0.5
			✓	50.1±0.1	35.3±0.5
				50.7±0.1	38.6±0.4
10	✓	✓	✓	75.5±0.1	59.4±0.0
	✓	✓		75.6±0.2	58.9±0.1
	✓		✓	66.5±0.3	62.6±0.2
	✓			67.6±0.3	62.7±0.3
		✓	✓	69.3±0.3	45.6±0.1
		✓		69.6±0.2	47.4±0.1
			✓	60.4±0.2	47.7±0.1
				61.0±0.2	49.2±0.1
50	✓	✓	✓	80.6±0.1	64.9±0.2
	✓	✓		78.4±0.3	66.1±0.1
	✓		✓	71.4±0.1	67.7±0.1
	✓			72.5±0.2	68.6±0.2
		✓	✓	74.8±0.3	55.0±0.1
		✓		72.1±0.2	55.8±0.2
			✓	66.8±0.1	56.1±0.2
				67.2±0.2	56.7±0.3

See paper appendix for ablation on other datasets

(Regularized) ZCA preprocessing

- $X \in \mathbb{R}^{n \times d}$, dataset
- $\tilde{X}^T \in \mathbb{R}^{d \times n}$: standardized design matrix
- $C = \tilde{X}^T \tilde{X} = U \Sigma U^T$ (SVD)
- Regularization parameter $\lambda \geq 0$:

$$\phi_\lambda(\mu) = \frac{1}{\mu + \lambda \overline{\mathrm{tr}}(C)}$$

- $W_\lambda = U \phi_\lambda(\Sigma) U^T$ (whitening matrix)
- Regularized ZCA:

$$X \mapsto \texttt{LayerNorm}(X).\texttt{dot}(W_\lambda)$$

Observation: Intrinsic Dimension Grows with KIP

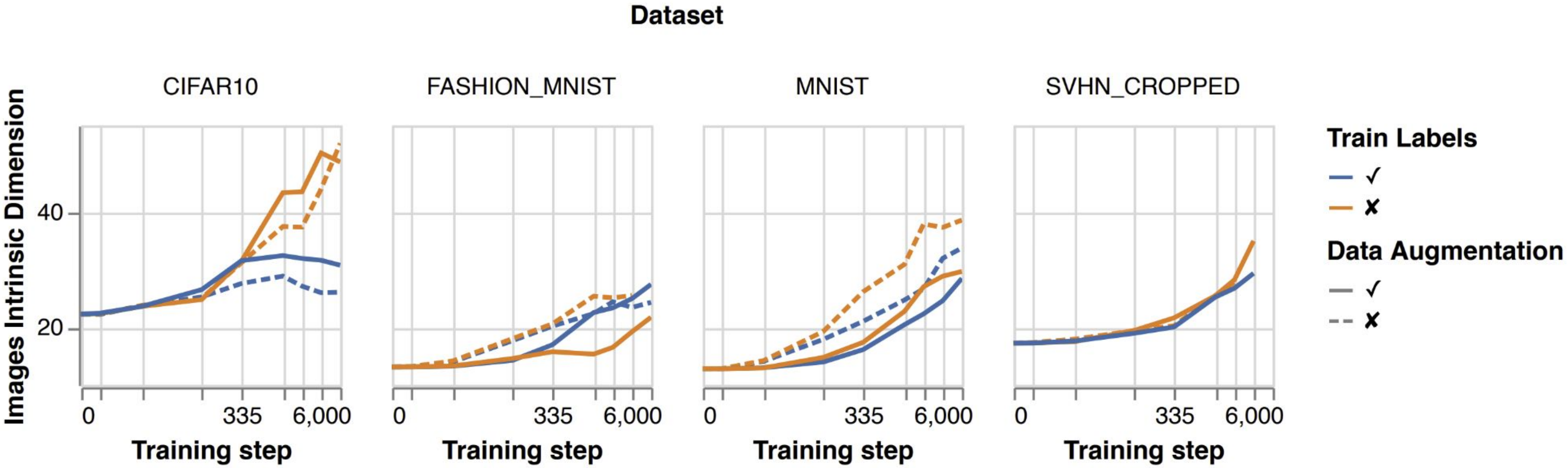


Figure 7: **Intrinsic dimension of learned datasets grows during training.** As training progresses (left to right), intrinsic dimension of the learned dataset grows, indicating that training transforms the data manifold in a non-trivial way. See Figures A2 and 6 for visual examples of learned images, and Figures A3 and A4 for similar observations using other dimensionality metrics and/or settings. All images here have standard preprocessing. Full experimental details in §A.

KIP & LS well behaved under large-width theory

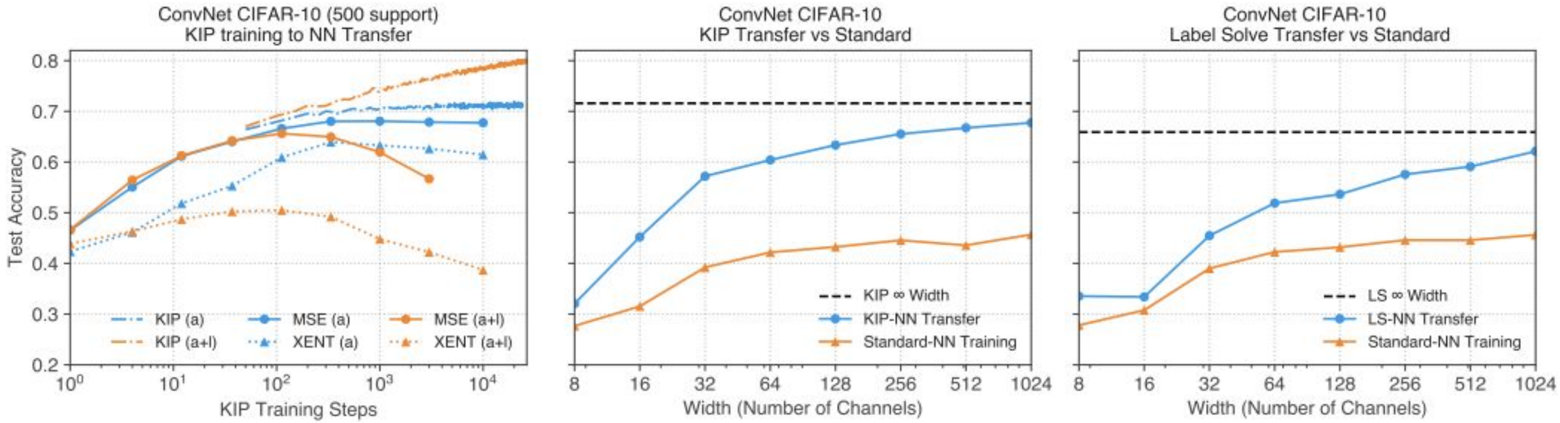


Figure 3: **Variations for neural network transfer.** *Left:* Plot of transfer performance as a function of KIP training steps across various train settings. Here (a) denotes augmentations used during KIP training and (a+l) denotes that additionally the labels were learned. MSE and XENT denote mean-square-error and cross entropy loss for the neural network, where for the case of XENT and (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.

NN transfer robust to training hyperparameter choices

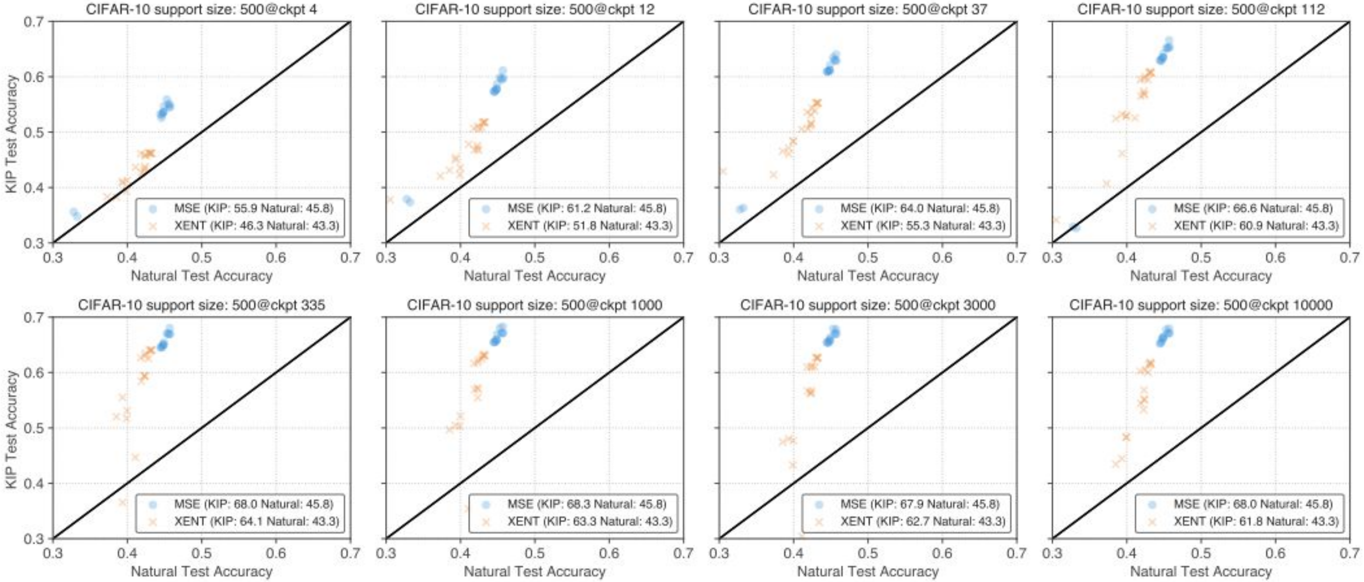


Figure 4: **Hyperparameter robustness.** In the above, 500 KIP images across eight different checkpoints are used to train the ConvNet neural network. Each point in each plot is a neural network training with a different hyperparameter, and its location records the final test accuracy when training on 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.

Conclusion / Future Work

- **Conclusion**
  - KIP & LS with conv architecture achieve SoTA on Dataset Distillation
  - Obtained by leveraging infinite-width correspondence
  - Implemented distributed meta-learning: OSS learned datasets
    - https://github.com/google-research/google-research/tree/master/kip

- **Future Work**
  - Better understanding of learned dataset
  - Scaling up for more challenging dataset: ImageNet, non-Image data
    - Efficient kernel computation: via sketching? arXiv: 2106.07880
  - Applications in privacy or federated learning, ... ?

Thank you!
Questions?