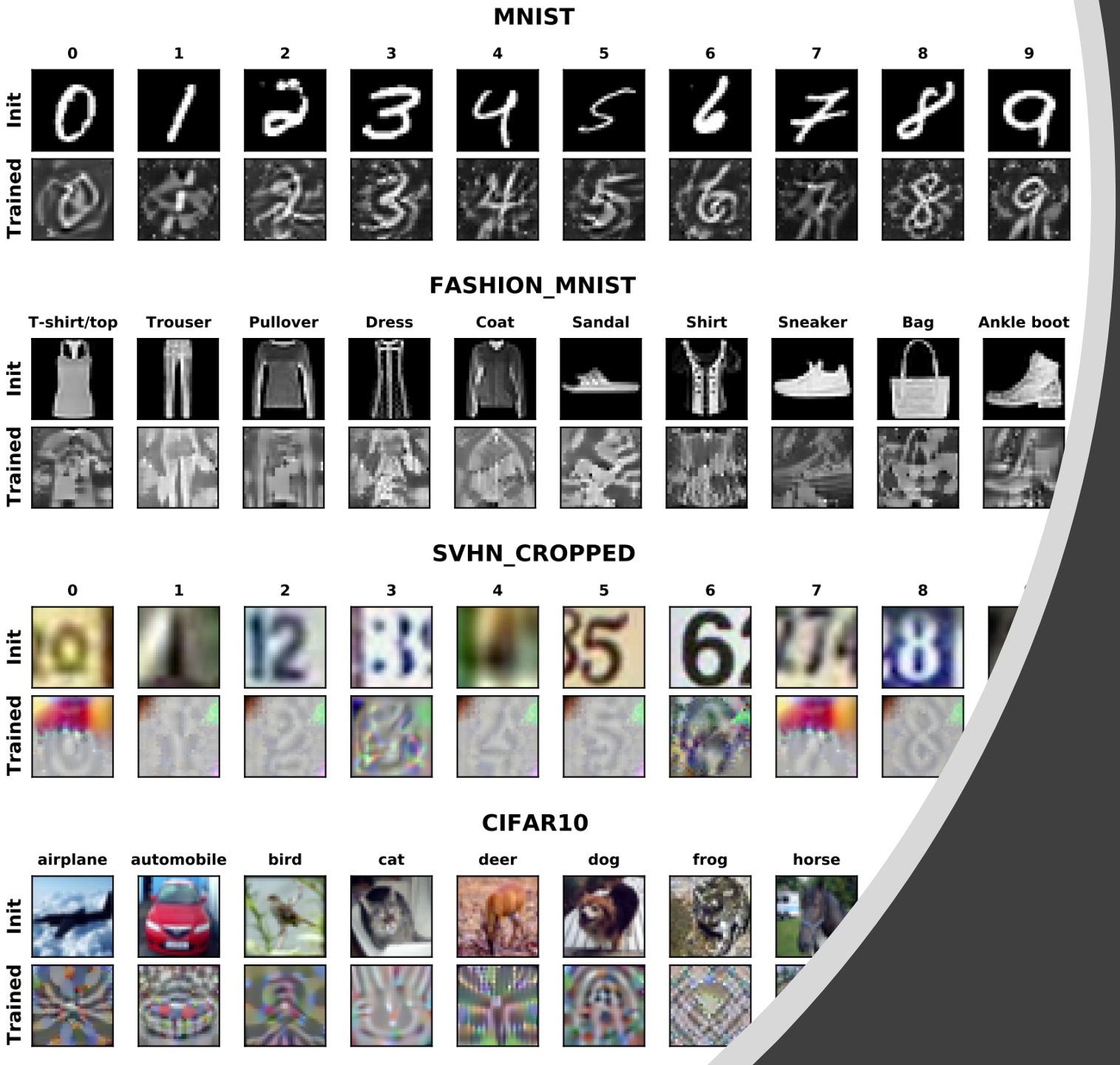
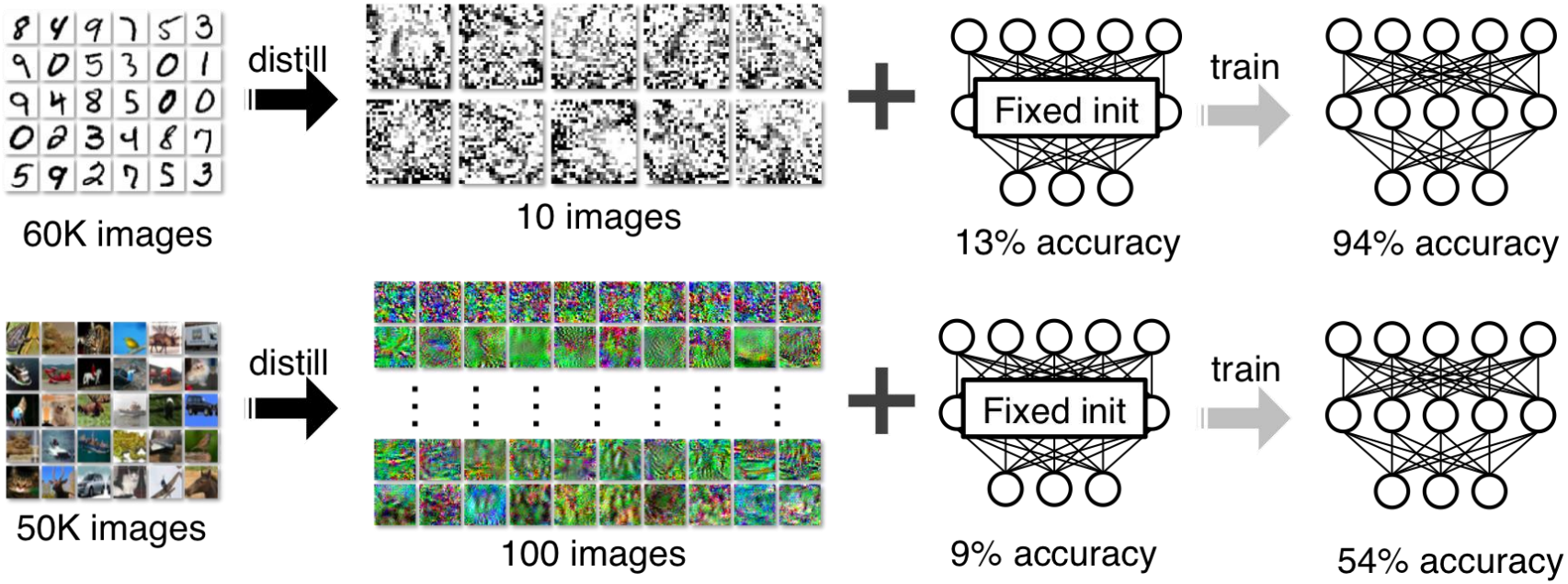


Dataset Distillation with Infinitely Wide Convolutional Networks

Mikhail Kuznetsov
4th year
CMC MSU

What is data distillation?



Dataset distillation on MNIST and CIFAR10

[1]

Distillation method.Recap

Ridge Regression

$$\min_{w} \|Xw - y\|_2^2 + \lambda \|w\|_2^2$$

$$w^* = \left(X^\top X + \lambda I\right)^{-1} X^\top y$$

$$y_{\text{new}} = X_{\text{new}} w^*$$

Kernel Ridge Regression

$$\min_{\beta} \|K_{XX}\beta - y\|_2^2 + \lambda \beta^\top K_{XX} \beta$$

$$\beta^* = \left(K_{XX} + \lambda I\right)^{-1} y$$

$$y_{\text{new}} = K_{X_{\text{new}} X} \beta^*$$

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

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

Distillation method.KIP

Algorithm 1: Kernel Inducing Point (KIP)

Require: A target labeled dataset (X_t, y_t) along with a kernel or family of kernels.

1: Initialize a labeled support set (X_s, y_s).
2: **while** not converged **do**
3: Sample a random kernel. Sample a random batch $(\bar{X}_s, \bar{y}_s)$ from the support set. Sample a random batch $(\bar{X}_t, \bar{y}_t)$ from the target dataset.
4: Compute the kernel ridge-regression loss given by (7) using the sampled kernel and the sampled support and target data.
5: Backpropagate through $\bar{X}_s$ (and optionally $\bar{y}_s$ and any hyper-parameters of the kernel) and update the support set (X_s, y_s) by updating the subset $(\bar{X}_s, \bar{y}_s)$.
6: **end while**
7: **return** Learned support set (X_s, y_s)

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

Dual number: $x + x'\varepsilon,\ \varepsilon^2 = 0$

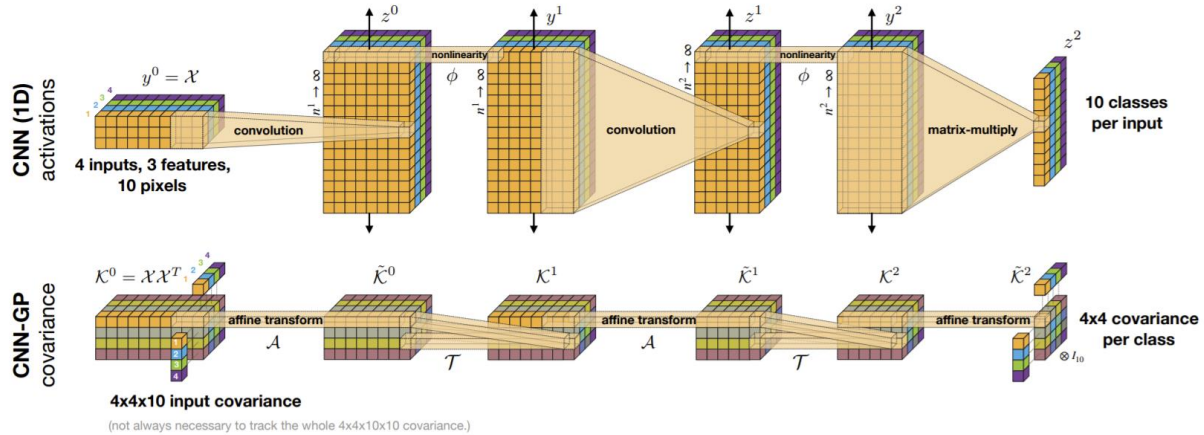
$$f(x + x'\varepsilon) = f(x) + x' f'(x)\varepsilon$$

$$\frac{\partial L}{\partial X_s} = \frac{\partial L}{\partial (K(X_s, X_s))} \frac{\partial K(X_s, X_s)}{\partial X_s} + \frac{\partial L}{\partial (K(X_t, X_s))} \frac{\partial K(X_t, X_s)}{\partial X_s}$$

[JAX jvp]

[KIP Github]

Infinitely Wide Convolutional Networks



Layer	Tensor Op	NNGP Op	NTK Op
0 (input)	$y^0 = \mathcal{X}$	$\mathcal{K}^0 = \mathcal{X}\mathcal{X}^T$	$\Theta^0 = 0$
0 (pre-activations)	$z^0 = \mathrm{Conv}\left(y^0\right)$	$\tilde{\mathcal{K}}^0 = \mathcal{A}\left(\mathcal{K}^0\right)$	$\tilde{\Theta}^0 = \tilde{\mathcal{K}}^0 + \mathcal{A}\left(\Theta^0\right)$
1 (activations)	$y^1 = \phi\left(z^0\right)$	$\mathcal{K}^1 = \mathcal{T}\left(\tilde{\mathcal{K}}^0\right)$	$\Theta^1 = \dot{\mathcal{T}}\left(\tilde{\mathcal{K}}^0\right) \odot \tilde{\Theta}^0$
1 (pre-activations)	$z^1 = \mathrm{Conv}\left(y^1\right)$	$\tilde{\mathcal{K}}^1 = \mathcal{A}\left(\mathcal{K}^1\right)$	$\tilde{\Theta}^1 = \tilde{\mathcal{K}}^1 + \mathcal{A}\left(\Theta^1\right)$
2 (activations)	$y^2 = \phi\left(z^1\right)$	$\mathcal{K}^2 = \mathcal{T}\left(\tilde{\mathcal{K}}^1\right)$	$\Theta^2 = \dot{\mathcal{T}}\left(\tilde{\mathcal{K}}^1\right) \odot \tilde{\Theta}^1$
2 (readout)	$z^2 = \mathrm{Dense} \circ \mathrm{Flatten}\left(y^2\right)$	$\tilde{\mathcal{K}}^2 = \mathrm{Tr}\left(\mathcal{K}^2\right)$	$\tilde{\Theta}^2 = \tilde{\mathcal{K}}^2 + \mathrm{Tr}\left(\Theta^2\right)$

An example of the translation of a convolution neural network into a sequence of kernel operations.[2]

Neural Tangents [lib] [paper]

Preprocessing.ZCA-regulized

1) Flatten the features for each train image and
then standardize each feature across the train dataset.
2) Feature-feature covariance matix $C = U\Sigma U^T$
3) Let $W_\lambda = U\phi_\lambda(\Sigma)U^T$ where ϕ_λ: μ to $(\mu + \lambda\overline{\mathrm{tr}}C)^{-1/2}$, $\overline{\mathrm{tr}}(C) = \mathrm{tr}(C)/\mathrm{len}(C)$
4) New features: standardize + @ W_λ

(if lambda = 0 -> standart ZCA = I cov matrix)

Before ZCA

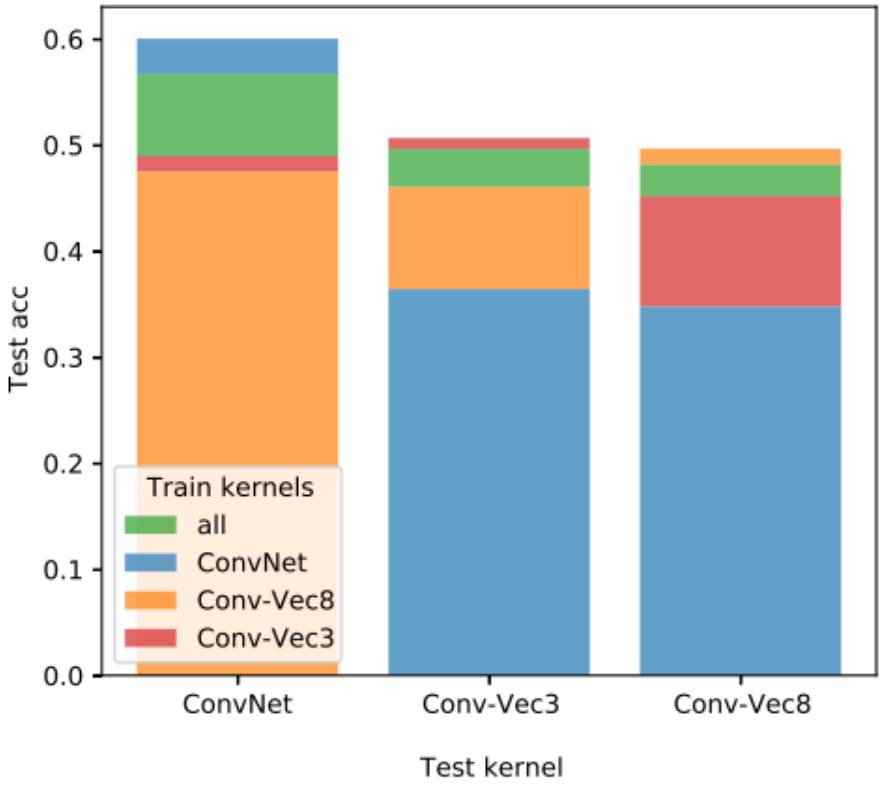
After ZCA [3]

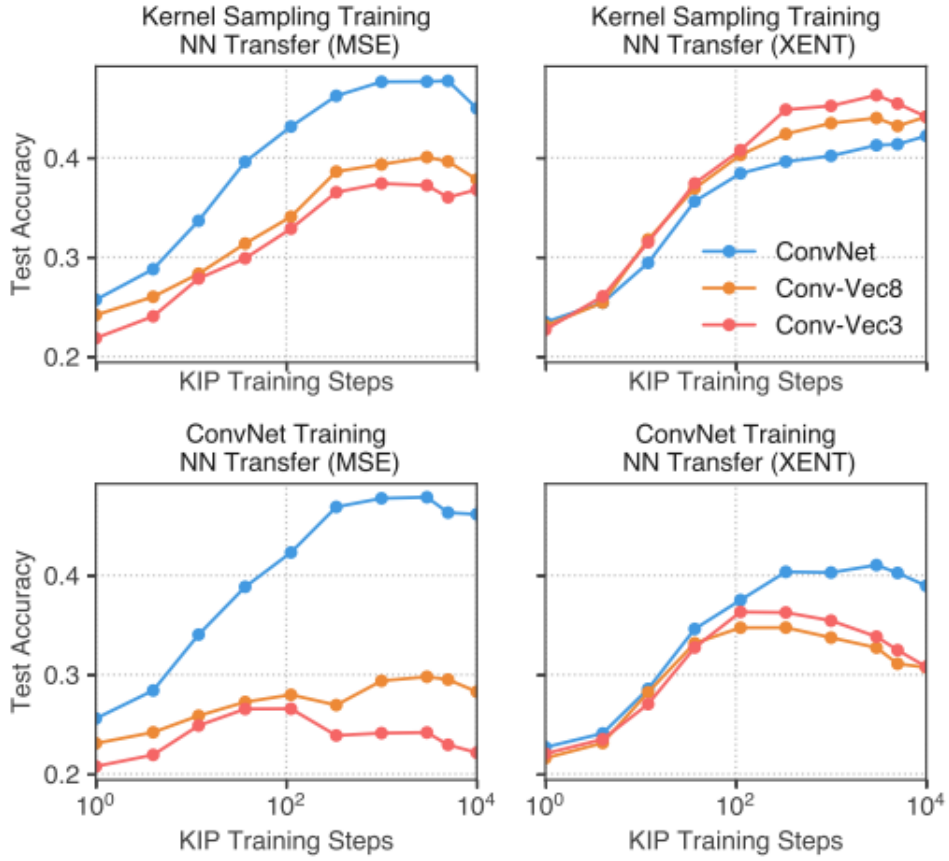
Experiments.KIP vs ALL

	Imgs/ Class	DC[1]	DSA[1]	KIP FC[1] aug	LS ConvNet[2,3]	KIP ConvNet[2] no aug	KIP ConvNet[2] 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**

+ ZCA, + label-learning, +- aug

Experiments.Kernels Choice



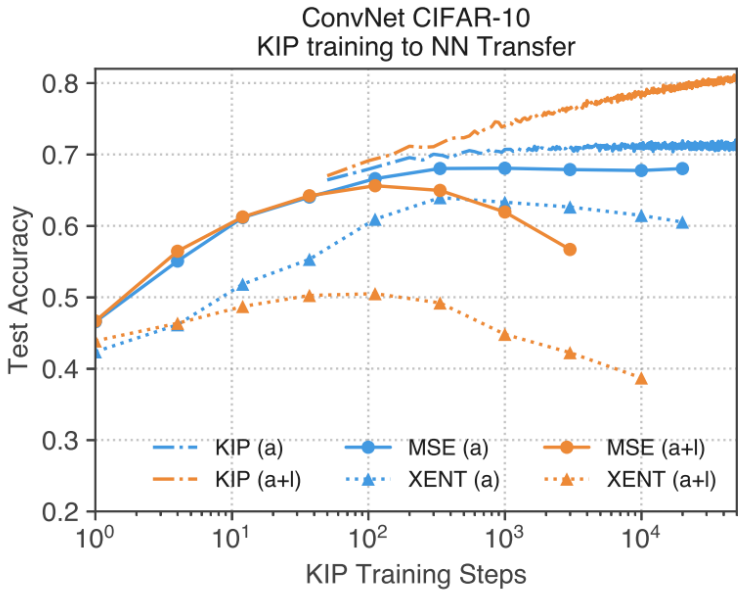


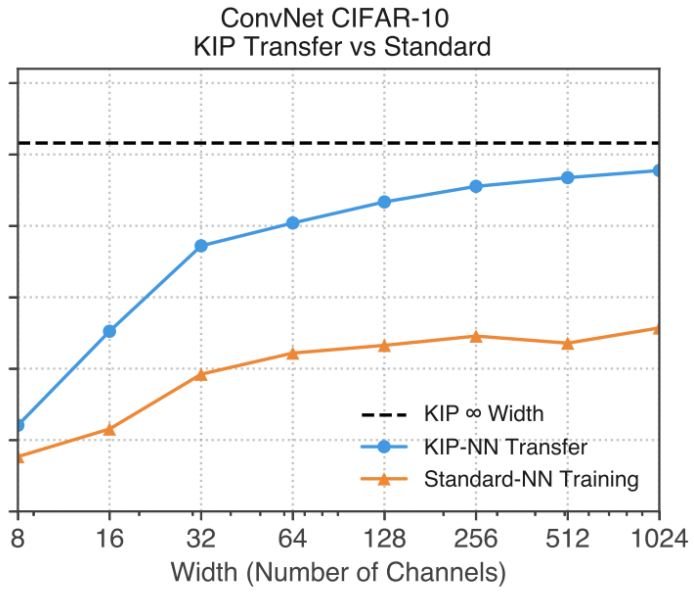
Experiments.Transfer

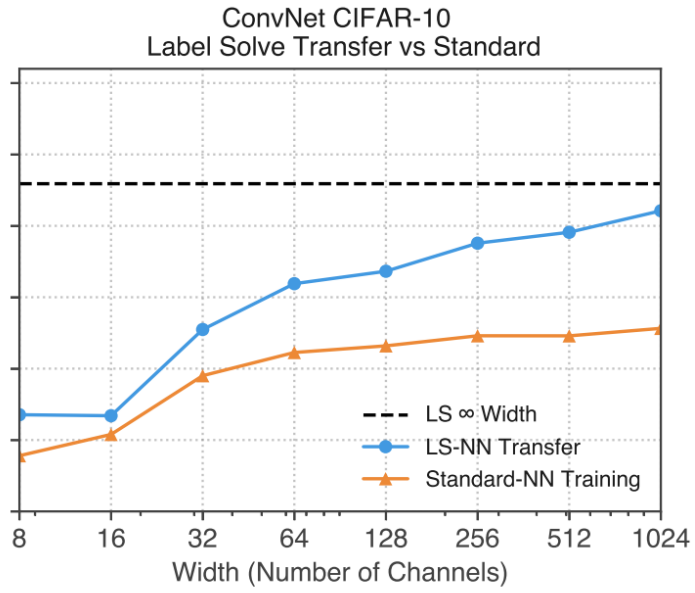
	Imgs/Class	DC/DSA	KIP to NN	Perf. change	LS to NN	Perf. change
MNIST	1	**91.7±0.5**	90.1±0.1	-5.5	71.0±0.2	-2.4
	10	**97.8±0.1**	97.5±0.0	-1.1	95.2±0.1	-1.2
	50	**99.2±0.1**	98.3±0.1	-0.8	97.9±0.0	-0.4
Fashion-MNIST	1	70.6±0.6	**73.5±0.5***	-9.8	61.2±0.1	-4.1
	10	84.6±0.3	**86.8±0.1**	-1.3	79.7±0.1	-1.2
	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

ZCA preprocessing, +- aug, * - best with trained labels

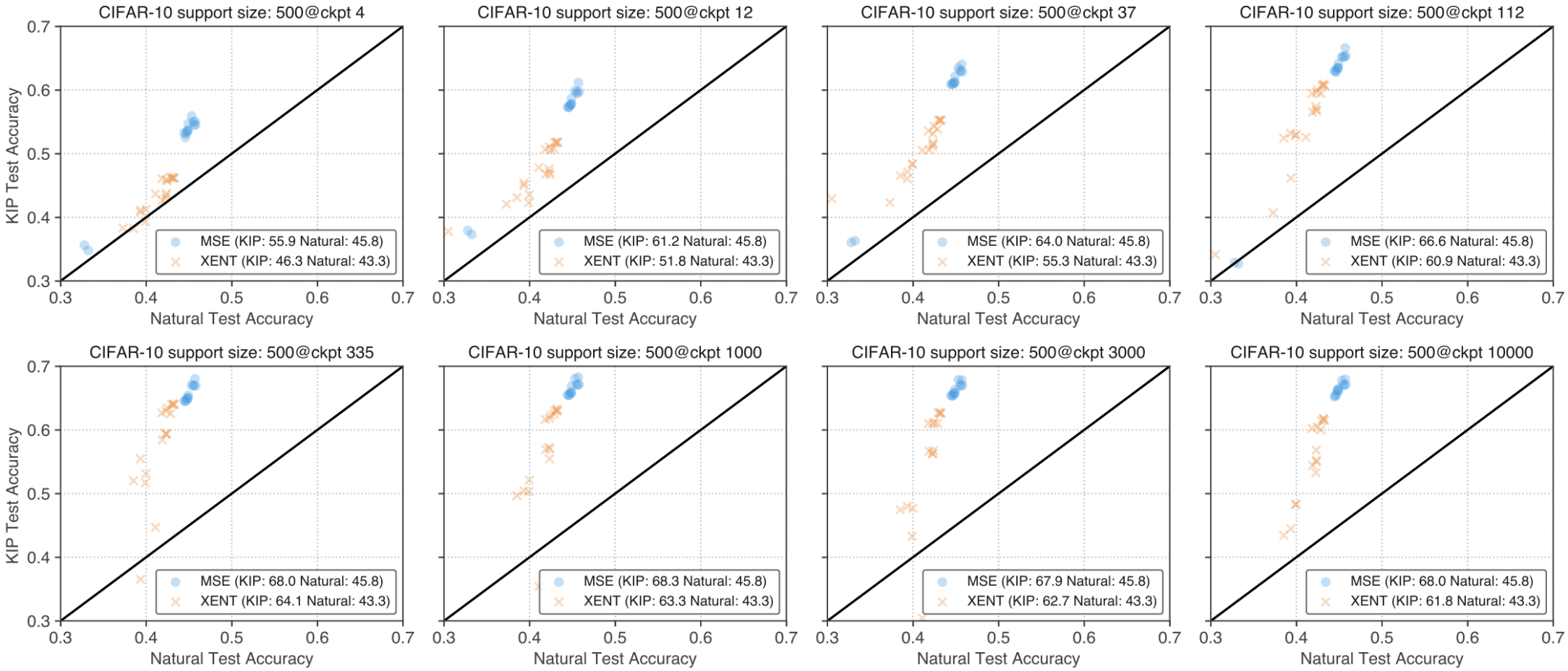
Experiments.Transfer



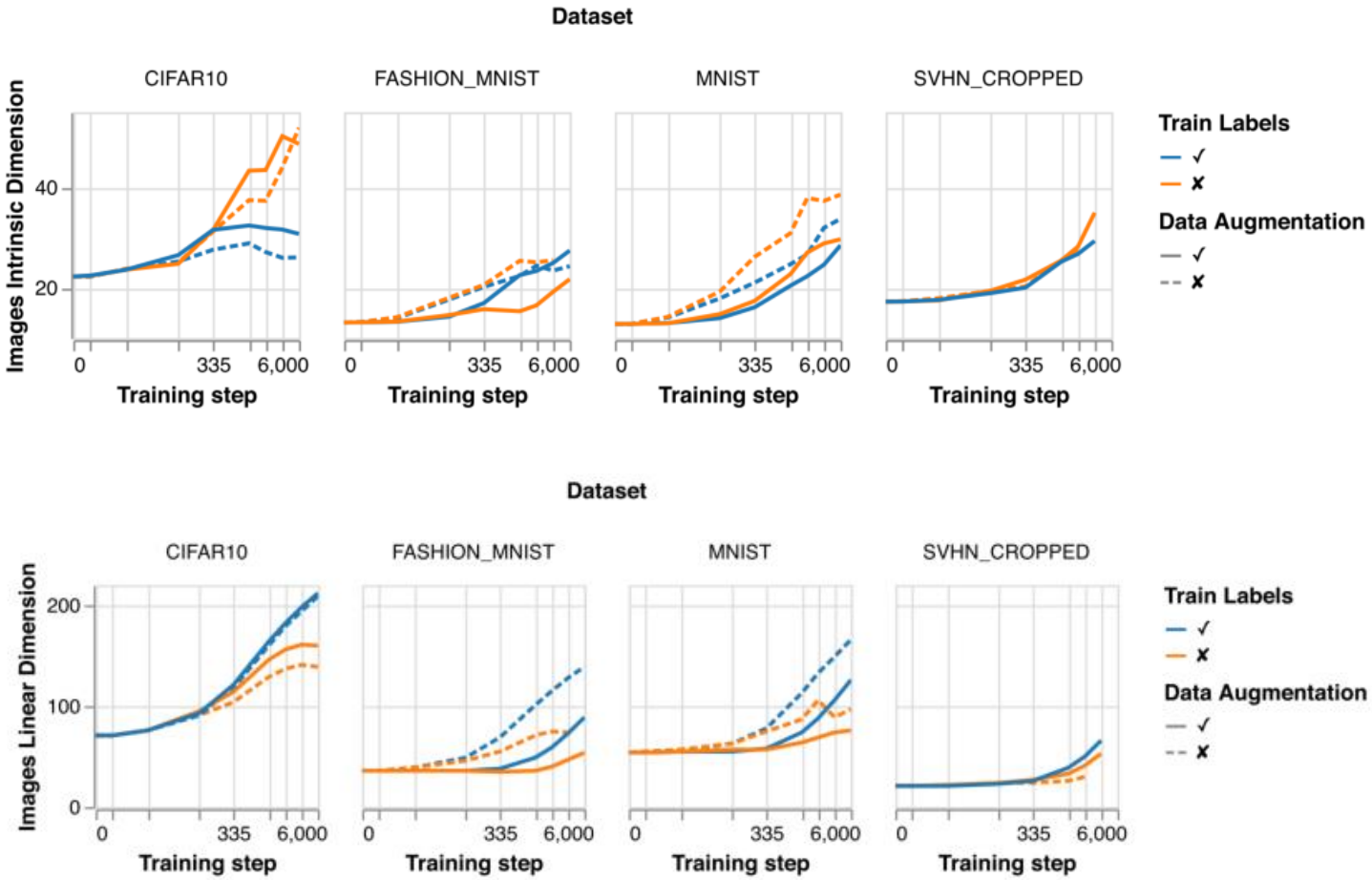




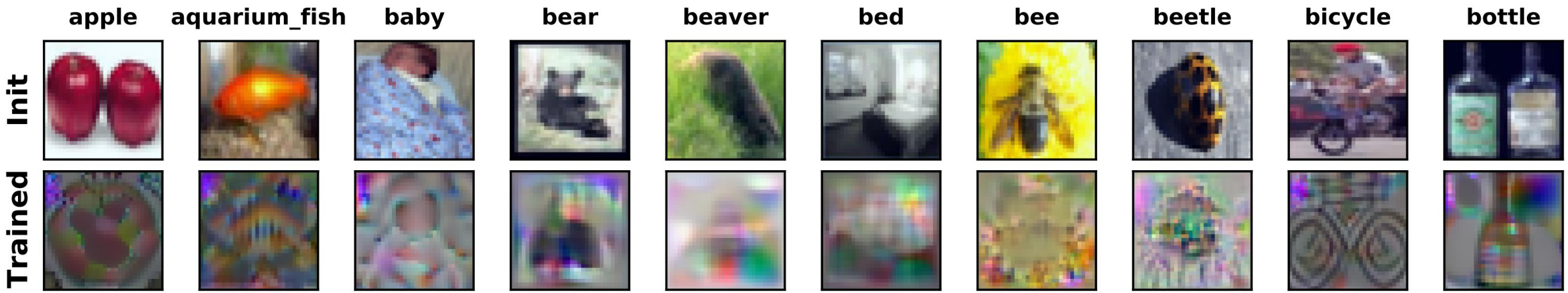
Experiments.Hyperparameters



Experiments.Datasets Dims

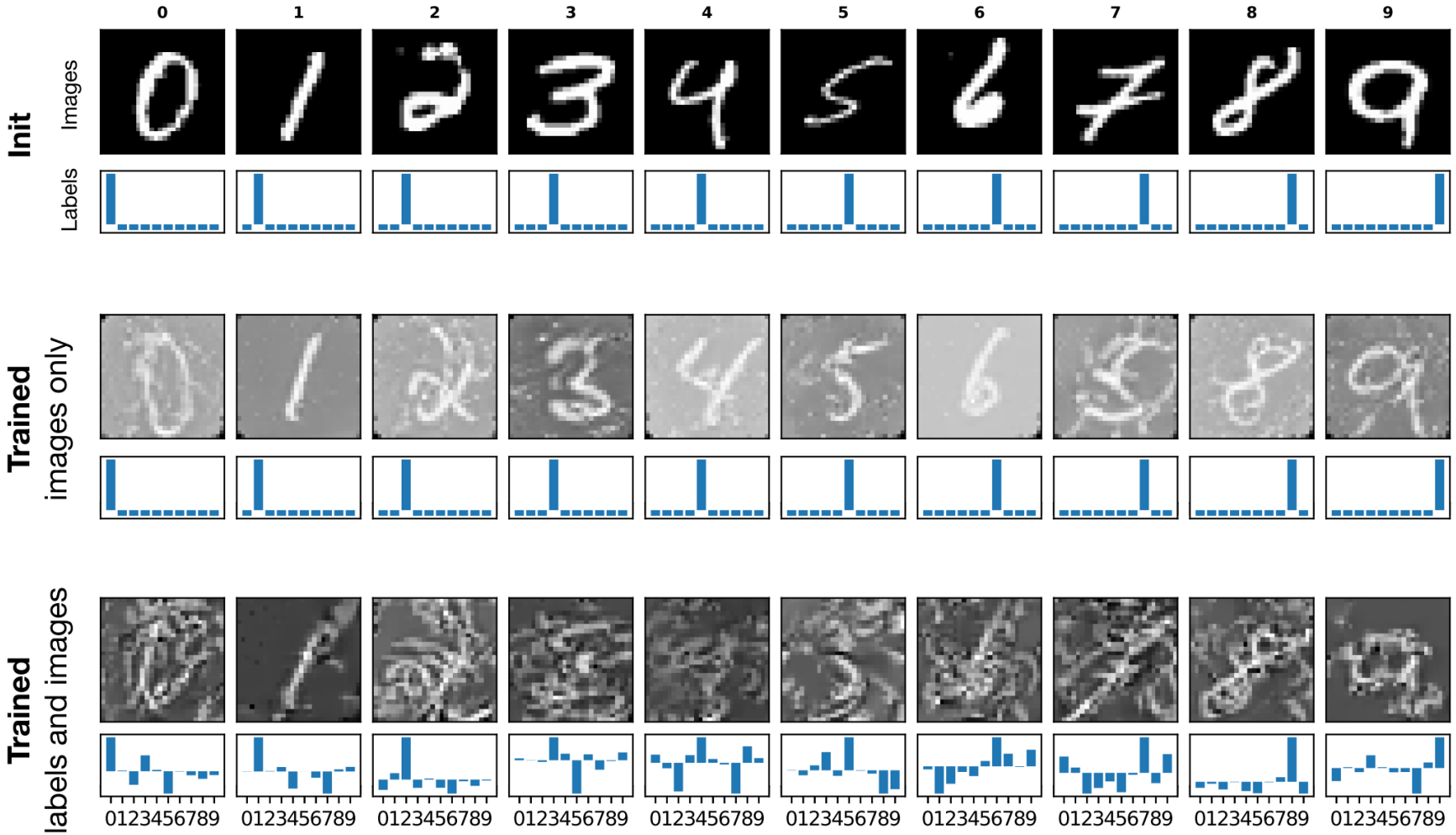


Experiments.Learned Images



CIFAR-100

Experiments.Learned Images



MNIST

Papers

- [1] Wang, T., Zhu, J., Torralba, A. and Efros, A. Dataset Distillation, 2018.
- [2] Novak R. et al. Neural tangents: Fast and easy infinite neural networks in python, 2019.
- [3] Pal K. K., Sudeep K. S. Preprocessing for image classification by convolutional neural, 2016.