



UTD THE UNIVERSITY OF TEXAS AT DALLAS

---

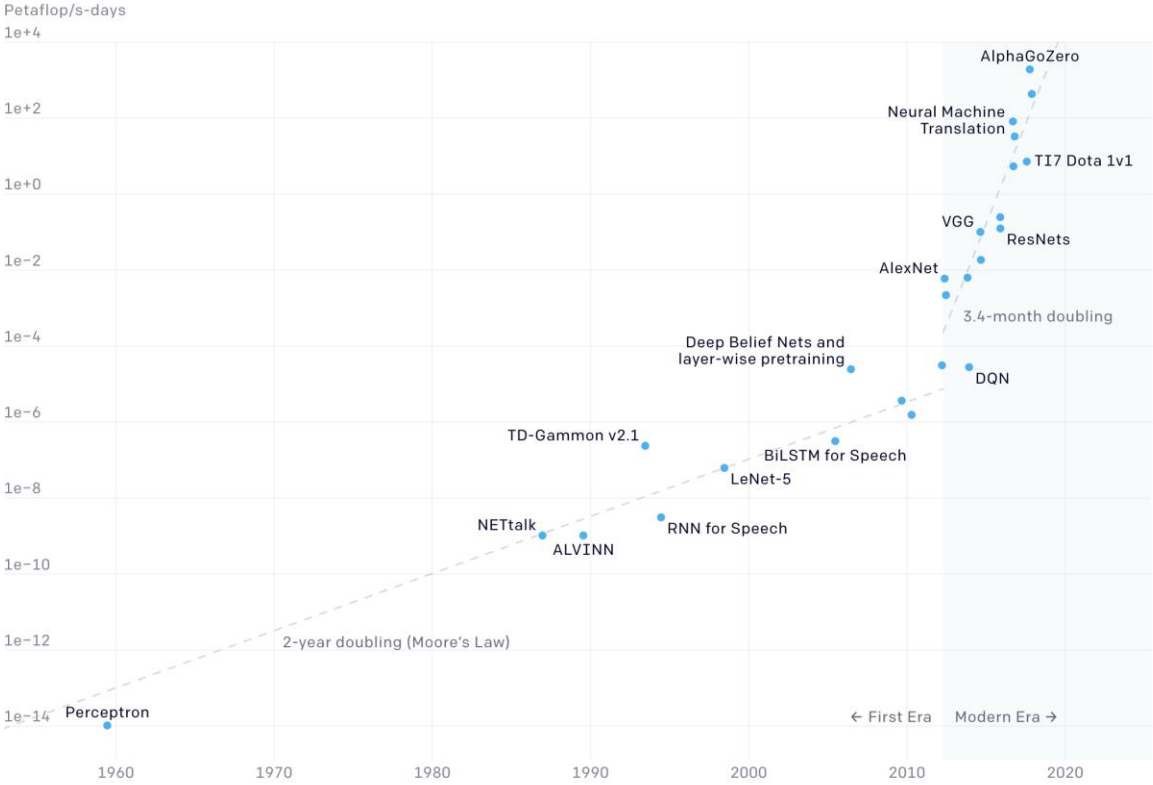
# ZEN: A Cross-architecture Generalizable Dataset Distillation Approach

---

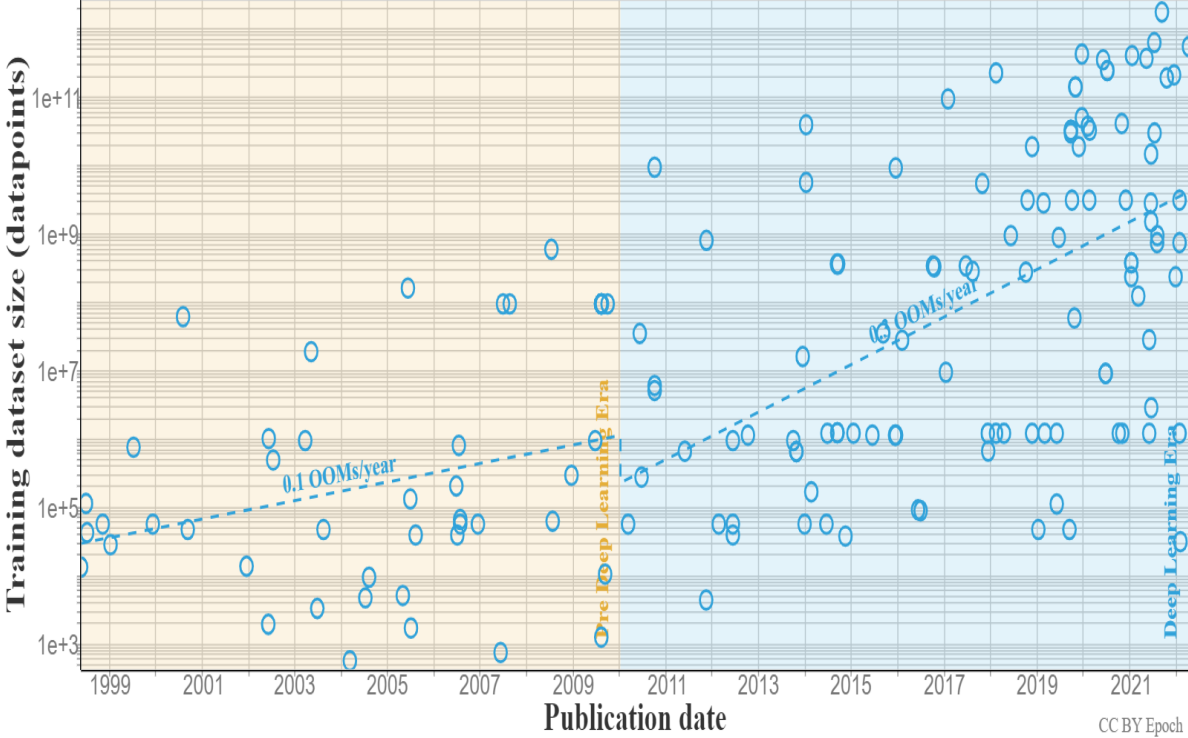
Group 17

# Compute Costs in AI

Two Distinct Eras of Compute Usage in Training AI Systems



# Dataset Sizes in AI

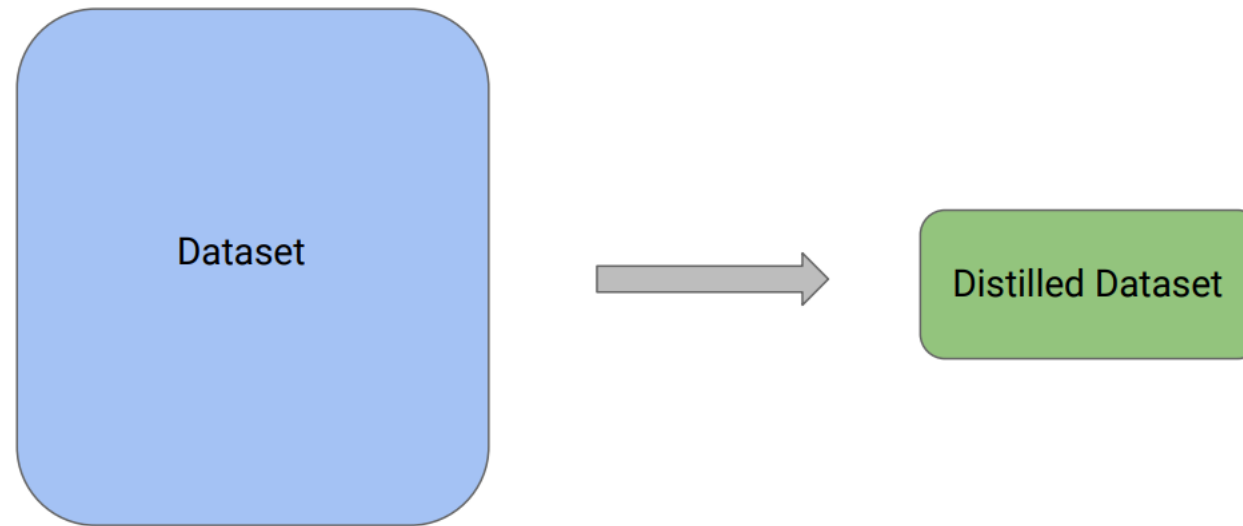


Source: Open AI Blog, [AI and Compute](#)

## Dataset Distillation

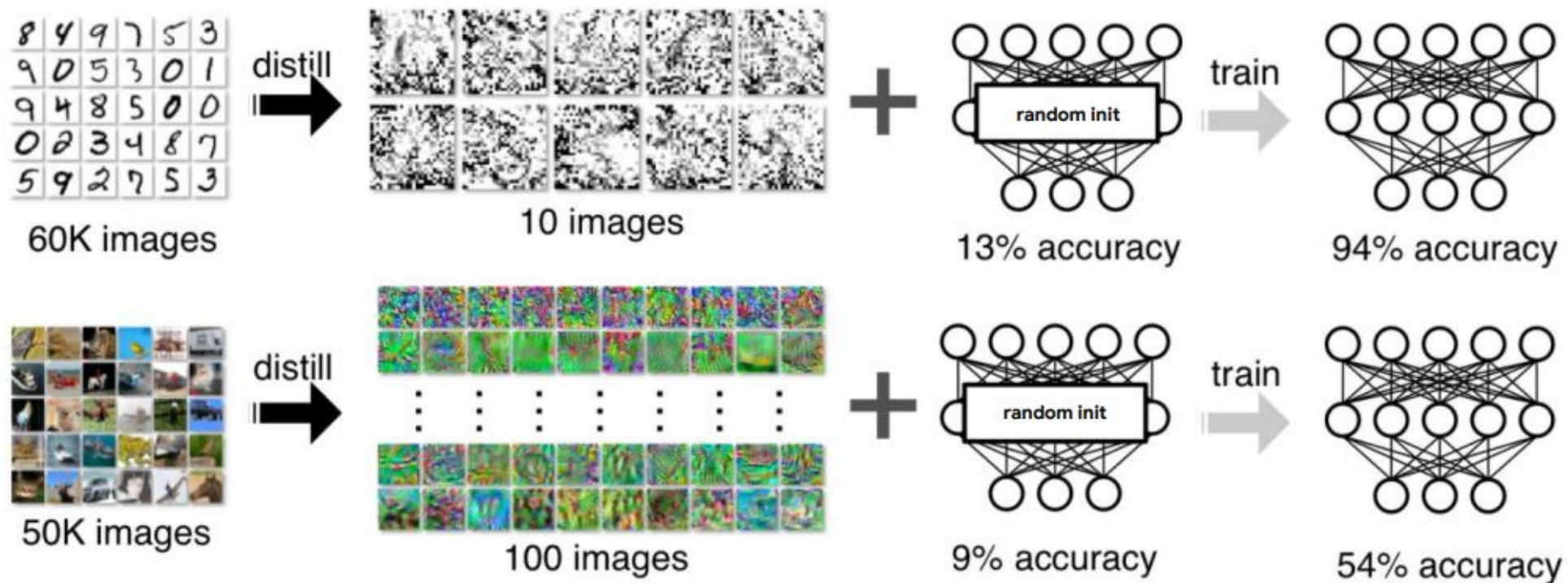
---

- Given a dataset, construct a smaller subset that performs nearly as well as the original.



Goal: **Distill** knowledge of entire dataset into a few **synthetic** data points

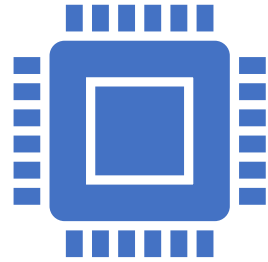
## Dataset Distillation



Dataset distillation on MNIST and CIFAR10

## Why Dataset Distillation?

---



### Space efficiency

Data storage

Especially for nonparametric methods like  
k-NN, kernel methods



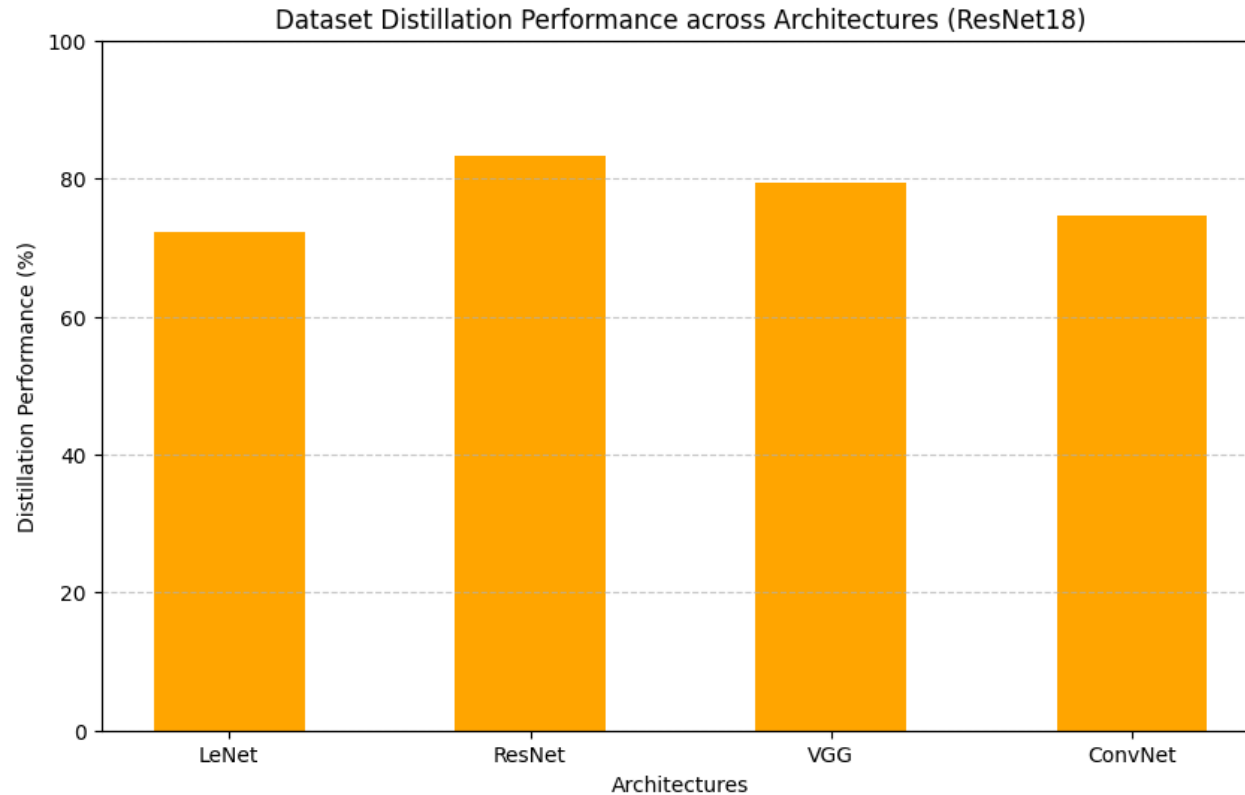
### Compute efficiency

Faster model training

Faster hyperparameter, architecture  
search


## Issues with Existing Dataset Distillation Approaches

Existing Dataset Distillation approaches generalize poorly to different model architectures



Dataset distilled on **ResNet** generalizes poorly when applied to **different model architectures**.

- Standard Dataset Distillation

$$S^* = \operatorname{argmin}_S \mathbb{E}_{\theta \sim P(\theta)} L_T(\operatorname{argmin}_{\theta} L_S(\theta))$$


Generalizes to only one model architecture

- ZEN

$$S^* = \operatorname{argmin}_S \max_{i=1 \dots A} L_T^i(\operatorname{argmin}_{\theta^i} L_S^i(\theta^i))$$

How do we optimize it?

The key idea of **ZEN** is to synthesize samples that generalize to multiple architectures by minimizing the worst-case loss.

# How to solve that?

**Idea:** Solving bi-level optimization by estimating the gradients of synthetic data and using gradient descent.

Estimate synthetic gradients using implicit function theorem

$$S^* = \operatorname{argmin}_S \max_{i=1 \dots A} L_T^i(\operatorname{argmin}_{\theta^i} L_S^i(\theta^i)) \longrightarrow \text{bi-level optimization}$$



# High-Order Optimization Approximation

**Implicit Differentiation:** We calculate the synthetic data gradient using chain rule,

$$\frac{\partial \max_{i=1 \dots A} L_T^i(\theta^{i*}(S), S)}{\partial S}$$

$$= \mathbf{Co} \bigcup_{i=1 \dots A} \left\{ \underbrace{\frac{\partial L_T^i}{\partial S}}_{(a)} + \underbrace{\frac{\partial L_T^i}{\partial \theta^{i*}(S)}}_{(b)} \times \underbrace{\frac{\partial \theta^{i*}(S)}{\partial S}}_{(c)} \right.$$

$$\left. \mid L_T^i(\theta^{i*}(S), S) = \max_{i=1 \dots A} L_T^i(\theta^{i*}(S), S) \right\}$$

- a. Synthetic data direct gradient (e.g., gradient from training data loss)
- b. parameter direct gradient
- c. best-response Jacobian

We approximate (c) by using the Implicit function theorem,

$$\frac{\partial \theta^{i*}(S)}{\partial S} = - \underbrace{\left[ \frac{\partial L_T^i}{\partial \theta^i \partial \theta^{iT}} \right]^{-1}}_{(d)} \times \underbrace{\frac{\partial L_T^i}{\partial S \partial \theta^{iT}}}_{(e)}$$

- d. Hessian Inverse w.r.t model parameters

We approximate the Hessian inverse using the Neumann series approximation[1],

$$\left[ \frac{\partial^2 L_T^i}{\partial \theta^i \partial \theta^{iT}} \right]^{-1} \approx \lim_{P \rightarrow \infty} \sum_{p=0}^P \left[ I - \frac{\partial L_T^i}{\partial \theta^i \partial \theta^{iT}} \right]^p$$

• 1. Lorraine, J., Vicol, P., & Duvenaud, D. (2020). Optimizing millions of hyperparameters by implicit differentiation. *International Conference on Artificial Intelligence and Statistics*, 1540–1552

## Optimization

---

### Algorithm 1: ZEN Framework

---

**Input:** Labeled Dataset:  $\mathcal{T}$ , Model Initialization

Distributions:  $\{p^i\}_{i=1}^A$

**Input:** Training steps:  $T$ , Number of Model Families:  $A$

**Output:** Synthetic Dataset:  $S$

```
1 Set  $t = 0$ ; learning rate  $\alpha, \beta$ ;  
2 Initialize synthetic images  $S$  to random tensors;  
3 repeat  
4    $i = 1$ ;  
5   repeat  
6     Sample Initial Weights for all models  $\theta_0^i \sim p(\theta^i)$ ;  
7   until  $i > A$   
8   /* Inner loop optimization */  
9   for  $i = 1, \dots, A$  do  
10    for  $j = 1, \dots, J$  do  
11       $\theta_j^i(S) = \theta_{j-1}^i - \alpha \nabla_{\theta} L_S^i(\theta_{j-1}^i, S)$   
12    /* Outer loop optimization, set  $\forall_{i=1}^A \theta^{i*} = \theta_J^i$  */  
13    Approximate inverse Hessian via Eq. (7);  
14    Calculate best-response Jacobian by Eq. (6);  
15    Calculate synthetic dataset gradient  $\nabla_S$  via Eq. (6);  
16    update Synthetic Images via  
       $S = S - \beta \cdot \nabla_S \max_{i=1 \dots A} L_T^i(\theta^{i*}, S)$ ;  
17  return  $S$   
18 until  $t < T$ 
```

---

## Results

# MNIST Results

Dataset	Img/Cls	Dataset Distillation Strategy	C\T	LeNet	ResNet	VGG	ConvNet
MNIST	10	DD	LeNet	76.51	78.24	77.71	72.59
			ResNet	72.31	83.31	79.31	74.56
			VGG	74.34	80.58	83.41	73.21
			ConvNet	78.32	79.54	78.64	79.5
MNIST	10	DC	LeNet	82.7	81.23	78.76	86.45
			ResNet	81.46	86.4	82.28	90.21
			VGG	79.43	84.31	81.7	90.16
			ConvNet	76.45	83.21	79.12	91.7
MNIST	10	ZEN	Combination	<b>84.32</b>	86.32	<b>84.53</b>	85.58

Table 1. The table shows the cross-architecture performance in terms of testing accuracy (%) for the MNIST dataset with a condensed 10 image/class representation. In the table, "C" represents the model used for dataset distillation, and "T" represents the model trained on the distilled images.

## Results

# CIFAR10 Results

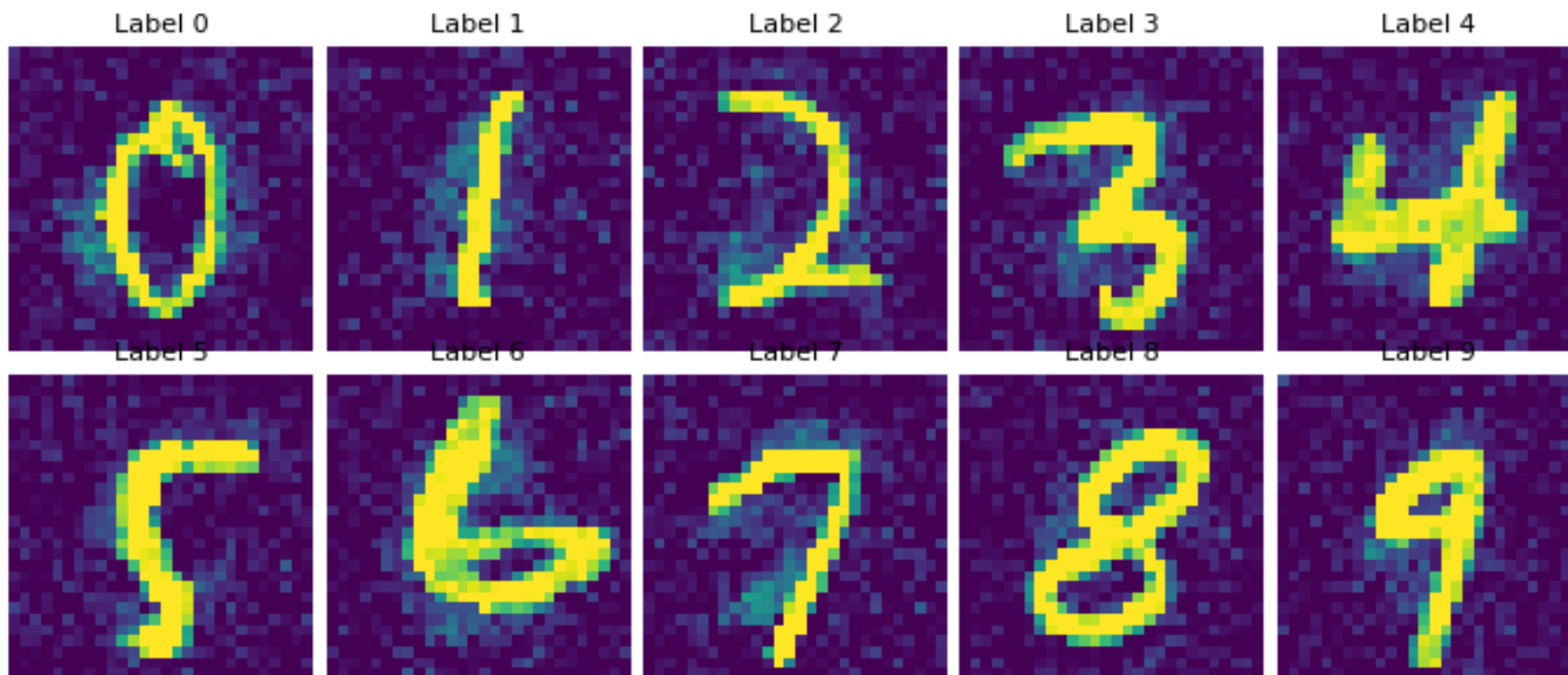
Dataset	Img/Cls	Dataset Distillation Strategy	C\T	AlexNet	ResNet	VGG	ConvNet
CIFAR10	10	DD	AlexNet	37.1	41.2	39.21	34.36
			ResNet	35.45	48.35	45.8	39.98
			VGG	33.2	43.6	47.21	38.56
			ConvNet	32.43	41.84	45.12	41.29
CIFAR10	10	DC	AlexNet	39.1	43.78	38.35	39.24
			ResNet	37.54	51.31	43.72	42.38
			VGG	35.76	45.67	44.25	41.86
			ConvNet	34.82	46.24	40.12	43.29
CIFAR10	10	ZEN	Combination	<b>45.3</b>	<b>51.78</b>	<b>48.53</b>	<b>44.16</b>

Table 2. The table shows the cross-architecture performance in terms of testing accuracy (%) for the CIFAR10 dataset with a condensed 10 image/class representation. In the table, "C" represents the model used for dataset distillation, and "T" represents the model trained on the distilled images.

## Results

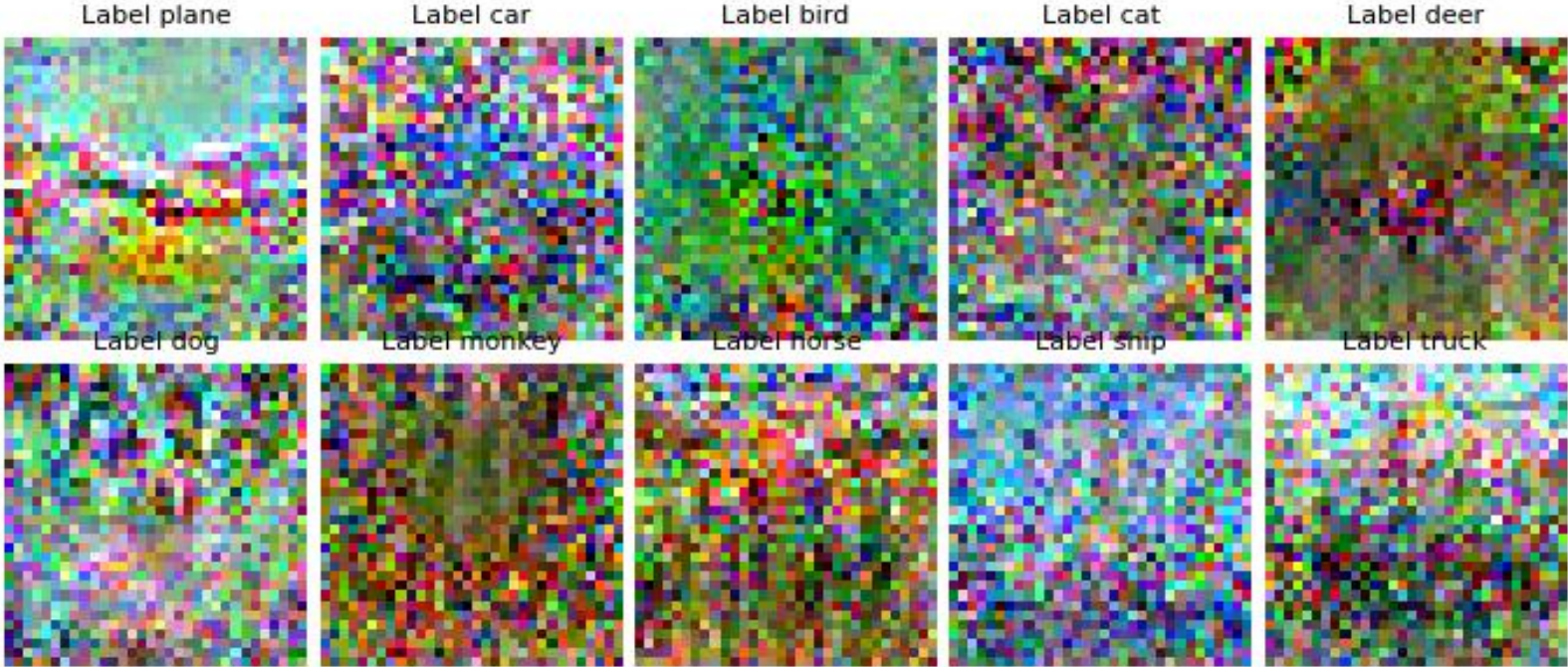
# MNIST Synthesized Images

Dataset: MNIST, Arch: LeNet, Step: 29



# CIFAR10 Synthesized Images

Dataset: Cifar10, Arch: AlexCifarNet, Step: 29, LR: 0.0089



## Conclusion

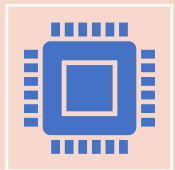
---



We developed an efficient dataset distillation framework for better cross-architecture generalization.



Empirical results show that our approach achieves better cross-architecture generalization compared to the baselines considered.



Even though ZEN achieved better cross-architecture generalization performance, it still underperforms latest dataset distillation works that uses a different problem formulation. It would be interesting to extend ZEN to consider latest dataset distillation formulations.



Questions?

