Dataset Condensation With Gradient Matching

Zhao et al. (https://arxiv.org/abs/2006.05929)
Presented by Matyas Skalicky, February 2021
Dataset Condensation with Gradient Matching

Bo Zhao, Konda Reddy Mopuri, Hakan Bilen
School of Informatics
The University of Edinburgh
{bo.zhao, kmopuri, hbilen}@ed.ac.uk

Abstract

As the state-of-the-art machine learning methods in many fields rely on larger datasets, storing them and training models on them becomes more expensive. This paper proposes a training set synthesis technique for data-efficient learning, called Dataset Condensation, that learns to condense a large dataset into a small set of informative samples for training deep neural networks from scratch. We formulate this goal as a gradient matching problem between the gradients of a deep neural network trained on the original data and our synthetic data. We rigorously evaluate its performance in several computer vision benchmarks and demonstrate that it significantly outperforms the state-of-the-art methods. Finally we explore the use of our method in continual learning and neural architecture search and show that it achieves promising gains on a tight budget of memory and computations.
Dataset Condensation

- Reduce large training dataset into a **small set of informative examples** 📊 to train a neural network 🧠
- We want to achieve **comparable performance** with a model trained on full training dataset
Related work

- Distilling the Knowledge in a Neural Network (Hinton & Vinyals)
- Traditional core-set construction methods
Distilling Knowledge in a Neural Network

- Compress knowledge from ensemble into a smaller model
- Train the smaller model on the predictions of larger one
Traditional (core-set) Methods

- Select the most representative data samples
- Presence of representative samples not guaranteed
- Not necessarily optimal solution for downstream task
- Rely on heuristics (criterion for representativeness)
Dataset condensation

From parameter to gradient matching
Dataset Condensation

- Traditional goal is to find synthetic dataset $S$
- Such that model trained on $S$ has comparable performance with model trained on full dataset $T$
- Nested loop optimization is computationally expensive

\[
S^* = \arg\min_S \mathcal{L}^T(\theta^S(S)) \quad \text{subject to} \quad \theta^S(S) = \arg\min_\theta \mathcal{L}^S(\theta)
\]
Parameter Matching

- We want model trained on synthetic data to converge to similar weights as model trained on full data
- Huge parameter space of $S$, expensive calculation

$$
\min_S D(\theta^S, \theta^T) \quad \text{subject to} \quad \theta^S(S) = \arg\min_\theta \mathcal{L}^S(\theta)
$$

distance between weights trained on synthetic and full data
inner loop can be expensive for large models
we wish synthetic model's weights not only to be close to the original weights but also to follow a similar path throughout the optimization
Gradient Matching

- Match the gradients of the real and synthetic training loss by updating the condensed samples $S$
- Ideally converge for any initial weights of synthetic model
- Weights are almost same, we don't need 2 sets of weights

$$
\min_S \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[ \sum_{t=0}^{T-1} D(\nabla_\theta \mathcal{L}^S(\theta_t), \nabla_\theta \mathcal{L}^T(\theta_t)) \right]
$$
Algorithm 1: Dataset condensation with gradient matching

Input: Training set $\mathcal{T}$

1 Required: Randomly initialized set of synthetic samples $S$ for $C$ classes, probability distribution over randomly initialized weights $P_{\theta_0}$, deep neural network $\phi_\theta$, number of outer-loop steps $K$, number of inner-loop steps $T$, number of steps for updating weights $\zeta_\theta$ and synthetic samples $\zeta_S$ in each inner-loop step respectively, learning rates for updating weights $\eta_\theta$ and synthetic samples $\eta_S$.

2 for $k = 0, \cdots, K - 1$ do
3     Initialize $\theta_0 \sim P_{\theta_0}$
4     for $t = 0, \cdots, T - 1$ do
5         for $c = 0, \cdots, C - 1$ do
6             Sample a minibatch pair $B_c^T \sim \mathcal{T}$ and $B_c^S \sim S$ $\triangleright$ $B_c^T$ and $B_c^S$ are of the same class $c$.
7             Compute $L_c^T = \frac{1}{|B_c^T|} \sum_{(x,y) \in B_c^T} \ell(\phi_\theta(x), y)$ and $L_c^S = \frac{1}{|B_c^S|} \sum_{(s,y) \in B_c^S} \ell(\phi_\theta(s), y)$
8             Update $S_c \leftarrow \text{opt-alg}_S(D(\nabla_\theta L_c^S(\theta_t), \nabla_\theta L_c^T(\theta_t)), \zeta_S, \eta_S)$
9             Update $\theta_{t+1} \leftarrow \text{opt-alg}_\theta(L^S(\theta_t), \zeta_\theta, \eta_\theta)$ $\triangleright$ Use the whole $S$

Output: $S$
Experiments

- Dataset condensation
- Cross-architecture generalization
- Effects of activation, normalization & pooling
Examples of condensed class images
## Dataset Condensation

| Dataset       | Img/Cls | Ratio % | Random    | Coreset Selection | Forgetting | Ours      | Whole Dataset |
|---------------|---------|---------|-----------|-------------------|------------|-----------|---------------|
| **MNIST**     | 1       | 0.017   | 64.9±3.5  | 89.2±1.6          | 35.5±5.6   | 91.7±0.5  | 99.6±0.0      |
|               | 10      | 0.17    | 95.1±0.9  | 93.7±0.3          | 68.1±3.3   | 97.4±0.2  |               |
|               | 50      | 0.83    | 97.9±0.2  | 94.9±0.2          | 88.2±1.2   | **98.8±0.1** |               |
| **FashionMNIST** | 1       | 0.017   | 51.4±3.8  | 67.0±1.9          | 42.0±5.5   | **70.5±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** |               |
|               | 50      | 0.83    | 82.5±0.7  | 71.9±0.8          | 55.0±1.1   | **83.6±0.4** |               |
| **SVHN**      | 1       | 0.014   | 14.6±1.6  | 20.9±1.3          | 12.1±1.7   | **31.2±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** |               |
|               | 50      | 0.7     | 70.9±0.9  | 72.6±0.8          | 27.2±1.5   | **82.3±0.3** |               |
| **CIFAR10**   | 1       | 0.02    | 14.4±2.0  | 21.5±1.2          | 13.5±1.2   | **28.3±0.5** | 84.8±0.1      |
|               | 10      | 0.2     | 26.0±1.2  | 31.6±0.7          | 23.3±1.0   | **44.9±0.5** |               |
|               | 50      | 1       | 43.4±1.0  | 40.4±0.6          | 23.3±1.1   | **53.9±0.5** |               |
Dataset Distillation, Wang et al., 2018
Dataset Distillation, Wang et al., 2018
Cross-Architecture Performance on MNIST

- Condensed images using training architecture C used to train unseen architecture T

| C  \ T    | MLP    | ConvNet | LeNet  | AlexNet | VGG     | ResNet   |
|----------|--------|---------|--------|---------|---------|----------|
| MLP      | 70.5±1.2 | 63.9±6.5 | 77.3±5.8 | 70.9±11.6 | 53.2±7.0 | 80.9±3.6 |
| ConvNet  | 69.6±1.6 | 91.7±0.5 | 85.3±1.8 | 85.1±3.0 | 83.4±1.8 | 90.0±0.8 |
| LeNet    | 71.0±1.6 | 90.3±1.2 | 85.0±1.7 | 84.7±2.4 | 80.3±2.7 | 89.0±0.8 |
| AlexNet  | 72.1±1.7 | 87.5±1.6 | 84.0±2.8 | 82.7±2.9 | 81.2±3.0 | 88.9±1.1 |
| VGG      | 70.3±1.6 | 90.1±0.7 | 83.9±2.7 | 83.4±3.7 | 81.7±2.6 | 89.1±0.9 |
| ResNet   | **73.6±1.2** | **91.6±0.5** | **86.4±1.5** | **85.4±1.9** | **83.4±2.4** | **89.4±0.9** |
Applications

- Continual learning
- Neural architecture search
Application in Continual Learning

- Ability of model to **learn continually** from a stream of data
- New tasks are learned incrementally while **preserving the performance on the old tasks** (*catastrophic forgetting*)
Neural Architecture Search

- Training complex architectures with large data is expensive
- Condensed images can be used to quickly identify best neural topology

|                      | Random | Herding | Ours  | Early-stopping | Whole Dataset |
|----------------------|--------|---------|-------|----------------|---------------|
| Performance (%)      | 76.2   | 76.2    | 84.5  | 84.5           | 85.9          |
| Correlation          | -0.21  | -0.20   | 0.79  | 0.42           | 1.00          |
| Time cost (min)      | 18.8   | 18.8    | 18.8  | 18.8           | 8604.3        |
| Storage (imgs)       | $10^2$ | $10^2$  | $10^2$| $10^4$         | $5 \times 10^4$ |
...leaky ReLu over ReLu and average pooling over max pooling enable learning better condensed images, as they allow for denser gradient flow.
Thanks for your attention

https://github.com/VICO-UoE/DatasetCondensation