Dataset Condensation With Gradient Matching
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

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

   for $k = 0, \cdots, K - 1$ do
   
   Initialize $\theta_0 \sim P_{\theta_0}$

   for $t = 0, \cdots, T - 1$ do

   for $c = 0, \cdots, C - 1$ do

   Sample a minibatch pair $B_c^T \sim \mathcal{T}$ and $B_c^S \sim S \quad \triangleright B_c^T$ and $B_c^S$ are of the same class $c$.

   Compute $L_c^T = \frac{1}{|B_c^T|} \sum_{(x, y) \in B_c^T} \ell(\phi_{\theta_t}(x), y)$ and $L_c^S = \frac{1}{|B_c^S|} \sum_{(s, y) \in B_c^S} \ell(\phi_{\theta_t}(s), y)$

   Update $S_c \leftarrow \text{opt-alg}_S(D(\nabla_\theta L_c^S(\theta_t), \nabla_\theta L_c^T(\theta_t)), \zeta_s, \eta_S)$

   Update $\theta_{t+1} \leftarrow \text{opt-alg}_\theta(L_c^S(\theta_t), \zeta_\theta, \eta_\theta) \quad \triangleright$ Use the whole $S$

**Output:** $S$
Experiments

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

<table>
<thead>
<tr>
<th></th>
<th>Img/Cls</th>
<th>Ratio %</th>
<th>Random</th>
<th>Coreset Selection</th>
<th>Forgetting</th>
<th>Ours</th>
<th>Whole Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MNIST</strong></td>
<td>1</td>
<td>0.017</td>
<td>64.9±3.5</td>
<td>89.2±1.6</td>
<td>35.5±5.6</td>
<td><strong>91.7±0.5</strong></td>
<td>99.6±0.0</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.17</td>
<td>95.1±0.9</td>
<td>93.7±0.3</td>
<td>68.1±3.3</td>
<td><strong>97.4±0.2</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.83</td>
<td>97.9±0.2</td>
<td>94.9±0.2</td>
<td>88.2±1.2</td>
<td><strong>98.8±0.1</strong></td>
<td></td>
</tr>
<tr>
<td><strong>FashionMNIST</strong></td>
<td>1</td>
<td>0.017</td>
<td>51.4±3.8</td>
<td>67.0±1.9</td>
<td>42.0±5.5</td>
<td><strong>70.5±0.6</strong></td>
<td>93.5±0.1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.17</td>
<td>73.8±0.7</td>
<td>71.1±0.7</td>
<td>53.9±2.0</td>
<td><strong>82.3±0.4</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.83</td>
<td>82.5±0.7</td>
<td>71.9±0.8</td>
<td>55.0±1.1</td>
<td><strong>83.6±0.4</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SVHN</strong></td>
<td>1</td>
<td>0.014</td>
<td>14.6±1.6</td>
<td>20.9±1.3</td>
<td>12.1±1.7</td>
<td><strong>31.2±1.4</strong></td>
<td>95.4±0.1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.14</td>
<td>35.1±4.1</td>
<td>50.5±3.3</td>
<td>16.8±1.2</td>
<td><strong>76.1±0.6</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.7</td>
<td>70.9±0.9</td>
<td>72.6±0.8</td>
<td>27.2±1.5</td>
<td><strong>82.3±0.3</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CIFAR10</strong></td>
<td>1</td>
<td>0.02</td>
<td>14.4±2.0</td>
<td>21.5±1.2</td>
<td>13.5±1.2</td>
<td><strong>28.3±0.5</strong></td>
<td>84.8±0.1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.2</td>
<td>26.0±1.2</td>
<td>31.6±0.7</td>
<td>23.3±1.0</td>
<td><strong>44.9±0.5</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>1</td>
<td>43.4±1.0</td>
<td>40.4±0.6</td>
<td>23.3±1.1</td>
<td><strong>53.9±0.5</strong></td>
<td></td>
</tr>
</tbody>
</table>
Dataset Distillation, Wang et al., 2018
<table>
<thead>
<tr>
<th>Plane</th>
<th>Car</th>
<th>Bird</th>
<th>Cat</th>
<th>Deer</th>
<th>Dog</th>
<th>Frog</th>
<th>Horse</th>
<th>Ship</th>
<th>Truck</th>
</tr>
</thead>
</table>

**Dataset Distillation, Wang et al., 2018**
Cross-Architecture Performance on MNIST

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

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

End-to-End Incremental Learning, Castro et al., 2018
Neural Architecture Search

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

<table>
<thead>
<tr>
<th></th>
<th>Random</th>
<th>Herding</th>
<th>Ours</th>
<th>Early-stopping</th>
<th>Whole Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance (%)</td>
<td>76.2</td>
<td>76.2</td>
<td><strong>84.5</strong></td>
<td><strong>84.5</strong></td>
<td>85.9</td>
</tr>
<tr>
<td>Correlation</td>
<td>-0.21</td>
<td>-0.20</td>
<td><strong>0.79</strong></td>
<td>0.42</td>
<td>1.00</td>
</tr>
<tr>
<td>Time cost (min)</td>
<td><strong>18.8</strong></td>
<td><strong>18.8</strong></td>
<td><strong>18.8</strong></td>
<td><strong>18.8</strong></td>
<td><strong>8604.3</strong></td>
</tr>
<tr>
<td>Storage (imgs)</td>
<td>$10^2$</td>
<td>$10^2$</td>
<td>$10^2$</td>
<td>$10^4$</td>
<td>$5 \times 10^4$</td>
</tr>
</tbody>
</table>

CIFAR10; 720 convnets
...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