Dataset condensation with gradient matching (DCGM)

ICLR, 2021

source: https://arxiv.org/abs/2006.05929

작성자: 이미지처리팀 류채은

link: https://www.notion.so/Dataset-condensation-with-gradient-matching-DCGM-8decb16262814a7eb0f549fe87708858?pvs=4

🧩 TL;DR

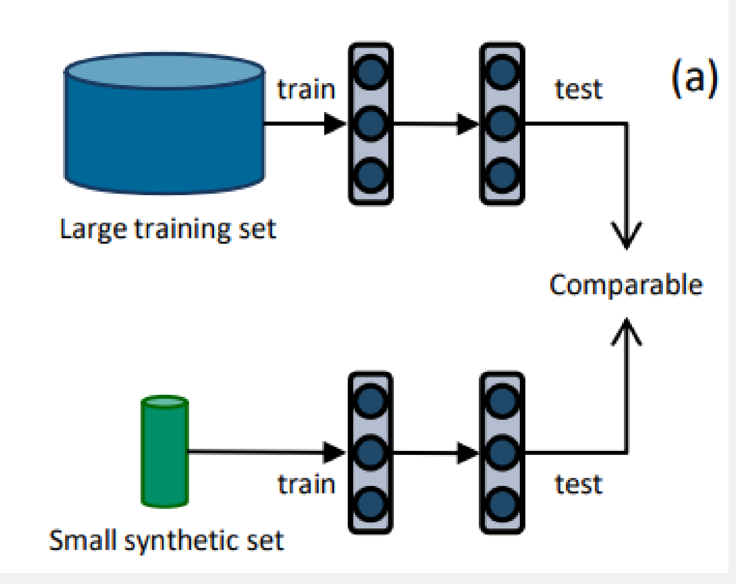
Proposes a training set synthesis technique for data-efficient learning that learns to **condense a large dataset into a small set of informative synthetic samples** for training DNNs, and aims to **obtain high generalization performance of the trained model on the synthesized small set** that is on par with the network trained on the original large dataset. This goal is reached **by learning to produce similar gradients** of the two networks - one trained on the synthesized small data and the other trained on the original large data.

Prerequisites

✅ **Dataset Condensation**

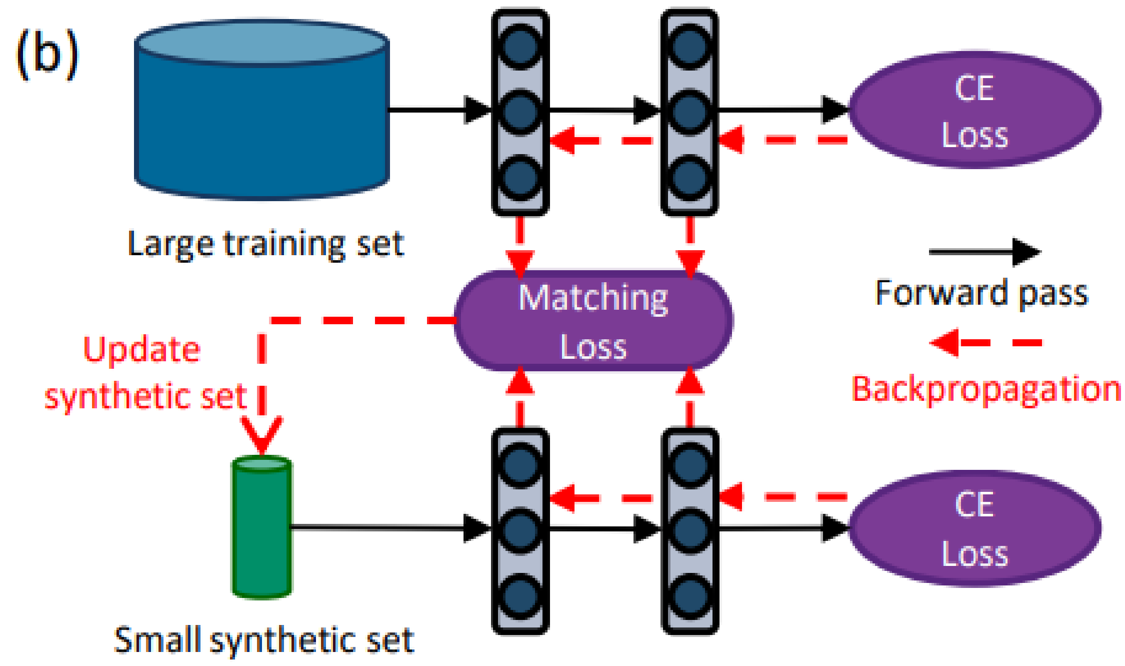
: Aims to generate a small set of synthetic images that can match the performance of a network trained on a large image dataset.

(= **작은 합성 데이터를 만들어서 원본의 엄청 큰 데이터를 대체하자**!)



Dataset Condensation

Overview



DCGM's approach

Learn a synthetic set such that a deep network trained both on it and the large set produces similar gradients w.r.t. its weights. The synthetic data can later be used to train a network from scratch in a small fraction of the original computational load.

(= **큰 데이터에 학습 시킨 모델의 기울기랑 만든 작은 데이터에 학습 시킨 모델의 기울기 차이를 최소화하는 방향으로 효율적인 작은 데이터를 합성하자**)

Methodology

1. Problem Definition

: **DATASET CONDENSATION WITH CURRICULUM GRADIENT MATCHING**

🔑 **Notations**

- Original Large Data : $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{T}|}$
- Synthesized Small Data: $S = \{(s_i, y_i)\}_{i=1}^{|S|}$
- Deep Neural Network ϕ with parameters θ
- Random Initialization Distribution: P
- $\mathrm{opt} - \mathrm{alg}$: a specific optimization procedure with a fixed number of steps ζ
- $\mathcal{L}$: Empirical Loss

🧠 **Idea**

Wish the parameters on synthesized small data θ^S to be close to not only the final parameters on the original large data θ^T but also to follow a similar path to θ^T throughout the optimization (= guided optimization)

(= 합성된 데이터에 모델의 parameters가 학습되는 방향이 원본에서 모델의 parameters가 학습되는 방향과 동일하길 바람!)

For T iterations,

$$\min_S E_{\theta_0 \sim P_{\theta_0}} [\sum_{t=0}^{T-1} D(\theta_t^S, \theta_t^{\mathcal{T}})] \qquad (1)$$

subject to $\theta_{t+1}^S(S) = \mathrm{opt} - \mathrm{alg}_\theta(\mathcal{L}^S(\theta_t^S), \zeta^S)$ and $\theta_{t+1}^{\mathcal{T}} = \mathrm{opt} - \mathrm{alg}_\theta(\mathcal{L}^{\mathcal{T}}(\theta_t^{\mathcal{T}}), \zeta^{\mathcal{T}}) \qquad (2)$

$(1) \Rightarrow$ Minimize average difference D between DNN's parameters on synthesized small data θ_t^S and original data $\theta_t^{\mathcal{T}}$ at each step t with parameters randomly initialized as θ_0 for T iterations

$(2) \Rightarrow$ update parameters θ on each data $(S, \mathcal{T})$ by minimizing loss $\mathcal{L}$ on the respective data with optimization procedure $\text{opt} - \text{alg}$ with specified number of steps ζ

(= 작은 합성된 데이터 S 에 훈련한 모델의 parameters와 원본 큰 데이터 $\mathcal{T}$에 훈련한 모델의 parameters가 같게 되도록 작은 합성된 데이터 S를 만들어줘!)

> 📎 $\text{opt} - \text{alg}$
> : step gradient desent optimization
>
> [update rule with learning rate η_θ]
>
> $$\theta_{t+1} \leftarrow \theta_t - \eta_\theta \Delta_\theta \mathcal{L}(\theta_t)$$

Final Formulation with observation $D(\theta_t^S, \theta_t^{\mathcal{T}}) \simeq 0$ in preliminary experiments:

With θ as θ^S,

$$\min_S E_{\theta_0 \sim P_{\theta_0}} [\sum_{t=0}^{T-1} D(\Delta_\theta \mathcal{L}^S(\theta_t), \Delta_\theta \mathcal{L}^{\mathcal{T}}(\theta_t))] \qquad (3)$$

$(3) \Rightarrow$ minimize the difference between gradients with respect to loss on synthesized small set S and original set $\mathcal{T}$

(= 작은 합성 데이터 S에 대한 기울기와 원본 큰 데이터 $\mathcal{T}$에 대한 기울기가 최소화되도록 작은 합성 데이터 S를 만들어줘!)

> 📎 D: Gradient Matching Loss
>
> = a sum of layerwise losses
>
> with l as layer index, and L as the number of layers with weights,
> $D(\Delta_\theta \mathcal{L}^S, \Delta_\theta \mathcal{L}^{\mathcal{T}}) = \sum_{l=1}^{L} d(\Delta_{\theta^{(l)}} \mathcal{L}^S, \Delta_{\theta^{(l)}} \mathcal{L}^{\mathcal{T}})$
>
> *out = number of output nodes for each layer
>
> $$d(A, B) = \sum_{i=1}^{\text{out}} (1 - \frac{A_i \cdot B_i}{||A_i|| ||B_i||})$$

2. Algorithm

Algorithm 1: Dataset condensation with gradient matching

Input: Training set $\mathcal{T}$

Required: Randomly initialized set of synthetic samples $\mathcal{S}$ for C classes, probability distribution over randomly initialized weights $P_{\boldsymbol{\theta}_0}$, deep neural network $\phi_{\boldsymbol{\theta}}$, number of outer-loop steps K, number of inner-loop steps T, number of steps for updating weights $\varsigma_{\boldsymbol{\theta}}$ and synthetic samples $\varsigma_{\mathcal{S}}$ in each inner-loop step respectively, learning rates for updating weights $\eta_{\boldsymbol{\theta}}$ and synthetic samples $\eta_{\mathcal{S}}$.

for $k = 0, \cdots, K-1$ **do**
- Initialize $\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}$
- **for** $t = 0, \cdots, T-1$ **do**
  - **for** $c = 0, \cdots, C-1$ **do**
    - Sample a minibatch pair $B_c^{\mathcal{T}} \sim \mathcal{T}$ and $B_c^{\mathcal{S}} \sim \mathcal{S}$ ▷ $B_c^{\mathcal{T}}$ and $B_c^{\mathcal{S}}$ are of the same class c.
    - Compute $\mathcal{L}_c^{\mathcal{T}} = \frac{1}{|B_c^{\mathcal{T}}|}\sum_{(\boldsymbol{x},y)\in B_c^{\mathcal{T}}} \ell(\phi_{\boldsymbol{\theta}_t}(\boldsymbol{x}), y)$ and $\mathcal{L}_c^{\mathcal{S}} = \frac{1}{|B_c^{\mathcal{S}}|}\sum_{(\boldsymbol{s},y)\in B_c^{\mathcal{S}}} \ell(\phi_{\boldsymbol{\theta}_t}(\boldsymbol{s}), y)$
    - Update $\mathcal{S}_c \leftarrow \texttt{opt-alg}_{\mathcal{S}}(D(\nabla_{\boldsymbol{\theta}}\mathcal{L}_c^{\mathcal{S}}(\boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}}\mathcal{L}_c^{\mathcal{T}}(\boldsymbol{\theta}_t)), \varsigma_{\mathcal{S}}, \eta_{\mathcal{S}})$
  - Update $\boldsymbol{\theta}_{t+1} \leftarrow \texttt{opt-alg}_{\boldsymbol{\theta}}(\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_t), \varsigma_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\theta}})$ ▷ Use the whole $\mathcal{S}$

Output: $\mathcal{S}$

*$K \simeq$ trials, $T \simeq$ epochs

Experiments & Results

	Img/Cls	Ratio %	Random	Coreset Selection Herding	K-Center	Forgetting	Ours	Whole Dataset
MNIST	1	0.017	64.9±3.5	89.2±1.6	89.3±1.5	35.5±5.6	**91.7±0.5**	99.6±0.0
	10	0.17	95.1±0.9	93.7±0.3	84.4±1.7	68.1±3.3	**97.4±0.2**	
	50	0.83	97.9±0.2	94.9±0.2	97.4±0.3	88.2±1.2	**98.8±0.2**	
FashionMNIST	1	0.017	51.4±3.8	67.0±1.9	66.9±1.8	42.0±5.5	**70.5±0.6**	93.5±0.1
	10	0.17	73.8±0.7	71.1±0.7	54.7±1.5	53.9±2.0	**82.3±0.4**	
	50	0.83	82.5±0.7	71.9±0.8	68.3±0.8	55.0±1.1	**83.6±0.4**	
SVHN	1	0.014	14.6±1.6	20.9±1.3	21.0±1.5	12.1±1.7	**31.2±1.4**	95.4±0.1
	10	0.14	35.1±4.1	50.5±3.3	14.0±1.3	16.8±1.2	**76.1±0.6**	
	50	0.7	70.9±0.9	72.6±0.8	20.1±1.4	27.2±1.5	**82.3±0.3**	
CIFAR10	1	0.02	14.4±2.0	21.5±1.2	21.5±1.3	13.5±1.2	**28.3±0.5**	84.8±0.1
	10	0.2	26.0±1.2	31.6±0.7	14.7±0.9	23.3±1.0	**44.9±0.5**	
	50	1	43.4±1.0	40.4±0.6	27.0±1.4	23.3±1.1	**53.9±0.5**	

Table 1: The performance comparison to coreset methods. This table shows the testing accuracies (%) of different methods on four datasets. ConvNet is used for training and testing. Img/Cls: image(s) per class, Ratio (%): the ratio of condensed images to whole training set.

Coreset 방법보다 잘하더라..

Vizualization

Figure 2: Visualization of condensed 1 image/class with ConvNet for MNIST, Fashion-MNIST, SVHN and CIFAR10.

Condensed dataset S