Simulating Training Dynamics to Reconstruct Training Data from Deep Neural Networks

Hanling Tian, Yuhang Liu, Mingzhen He, Zhengbao He, Zhehao Huang, Ruikai Yang, Xiaolin Huang

PAMI, Shanghai Jiao Tong University, Shanghai, China

Task Definition

- Given a neural network $f_{\boldsymbol{\theta}}$ with the initial parameters $\boldsymbol{\theta}_0$ and the final parameters $\boldsymbol{\theta}_f$, dataset reconstruction aims to invert the parameters to the training dataset $\mathcal{D}$.
- Considering the learning algorithm $\mathcal{A}$, which is characterized by the hyperparameters H, the training process maps $\mathcal{D}$ to $\boldsymbol{\theta}_f$: $\boldsymbol{\theta}_f = \mathcal{A}_H(\mathcal{D}; \boldsymbol{\theta}_0)$.
- Dataset reconstruction is an inverse problem $\mathcal{D} = \mathcal{A}_H^{-1}(\boldsymbol{\theta}_f; \boldsymbol{\theta}_0)$.

Revisiting Training

- Considering stochastic gradient descent:

direction keeps the same

$$\boldsymbol{\theta}_0 - \boldsymbol{\theta}_f = \sum_{k=1}^{T} \sum_{j=1}^{N} \left(\eta \cdot \sum\nolimits_{\boldsymbol{x}, y \in \mathcal{B}_{k,j}} \frac{1}{|\mathcal{B}_{k,j}|} \nabla_{\boldsymbol{\theta}} \ell \left(f_{\boldsymbol{\theta}_{k,j}}(\boldsymbol{x}), y \right) \right)$$

$$\boldsymbol{\theta}_{k,j+1} = \boldsymbol{\theta}_{k,j} - \eta \cdot \sum\nolimits_{\boldsymbol{x}, y \in \mathcal{B}_{k,j}} \frac{1}{|\mathcal{B}_{k,j}|} \nabla_{\boldsymbol{\theta}} \ell \left(f_{\boldsymbol{\theta}_{k,j}}(\boldsymbol{x}), y \right), \qquad \boldsymbol{\theta}_{k,j+1} = \boldsymbol{\theta}_{k,N}$$

- For MLPs, the direction of $\nabla_{\boldsymbol{\theta}}$ keeps almost the same for each data point:

combining like terms

$$\boldsymbol{\theta}_0 - \boldsymbol{\theta}_f = \sum_{i=1}^{|\mathcal{D}|} \lambda_i \cdot \nabla_{\boldsymbol{\theta}} \ell \left(f_{\boldsymbol{\theta}_0}(\boldsymbol{x}_i), y_i \right)$$

- Optimize dummy images $\hat{\boldsymbol{x}}$ and scaling factors λ to reconstruct dataset from parameters.

Training Dynamics Linearity

- To measure the linearity of training dynamics, we calculate the cosine similarity between gradient pairs across different training epochs.

$$\mathcal{M}_{lin} = \frac{2}{|\mathcal{D}|}\frac{1}{T(T-1)}\sum_{i=1}^{|\mathcal{D}|}\sum_{1\leq t_1\neq t_2\leq T}\frac{\langle g_{i,t_1}, g_{i,t_2}\rangle}{\|g_{i,t_1}\|\|g_{i,t_2}\|}$$

- The linearity of MLP increases as width expands, which aligns with the theoretical predictions of neural tangent kernel (NTK).
- The non-linear training dynamics make dataset reconstruction from DNNs challenging.

Model	Width	$\mathcal{M}_{\mathrm{lin}}(\uparrow)$
MLP	200	0.9134
	500	0.9165
	1000	0.9306
	2000	0.9396
	4000	0.9535
ResNet-18	/	0.5988

Simulation of Training **Dy**namics (**SimuDy**)

Algorithm 1 Reconstructing training data using SimuDy.

Input: Network function $f_{\boldsymbol{\theta}}$, initial parameters $\boldsymbol{\theta}_0$, final parameters $\boldsymbol{\theta}_f$, dataset size n, training learning rate η, training steps T, batch size $|\mathcal{B}|$, dissimilarity function $\boldsymbol{d}(\cdot,\cdot)$, optimizer `Optim`;

Output: Reconstructed images via SimuDy;

Initialize dummy images $\hat{\boldsymbol{x}}$ with random noise

Assign labels to images randomly, ensuring an equal number of labels for each class

Randomly divide the dataset into batches of size $|\mathcal{B}|$, and the number of batches is N

$\hat{\boldsymbol{\theta}}_{1,1} \leftarrow \boldsymbol{\theta}_0$ ▷ Begin with the initial parameters

repeat

  for $k = 1$ **to** T **do** ▷ Simulate the training process for T epochs

    for $j = 1$ **to** N **do** ▷ Perform N updates for each epoch

      $\boldsymbol{g}_{k,j} = \sum_{\hat{\boldsymbol{x}},\hat{y}\in\mathcal{B}_{k,j}} \frac{1}{|\mathcal{B}_{k,j}|} \nabla_{\boldsymbol{\theta}} \ell(f_{\hat{\boldsymbol{\theta}}_{k,j}}(\hat{\boldsymbol{x}}), \hat{y})$ ▷ Compute the gradient for each update

      $\hat{\boldsymbol{\theta}}_{k,j+1} \leftarrow \hat{\boldsymbol{\theta}}_{k,j} - \eta \cdot \texttt{grad\_clip}(\boldsymbol{g}_{k,j})$ ▷ Clip gradients and update parameters

    end for

    $\hat{\boldsymbol{\theta}}_{k+1,1} \leftarrow \hat{\boldsymbol{\theta}}_{k,N+1}$

  end for

  $\hat{\boldsymbol{\theta}}_f \leftarrow \hat{\boldsymbol{\theta}}_{T,N+1}$ ▷ Get simulated final parameters

  $\mathcal{L}_{\text{recon}} = \boldsymbol{d}(\boldsymbol{\theta}_f - \boldsymbol{\theta}_0, \hat{\boldsymbol{\theta}}_f - \boldsymbol{\theta}_0) + \alpha \cdot \mathcal{L}_{\text{TV}}(\boldsymbol{x})$ ▷ Compute reconstruction loss

  $\hat{\boldsymbol{x}} \leftarrow \texttt{Optim}(\hat{\boldsymbol{x}}, \partial\mathcal{L}_{\text{recon}}/\partial\hat{\boldsymbol{x}})$ ▷ Update dummy images

until the reconstruction loss converges.

Reconstructions from MLP

(a) Reconstructions from MLP trained on 100 images using Buzaglo et al.'s, SSIM = 0.1426

(b) Reconstructions from MLP trained on 100 images using Loo et al.'s, SSIM = 0.1384

(c) Reconstructions from MLP trained on 100 images using SimuDy, SSIM = 0.3374

Reconstructions from ResNet-18

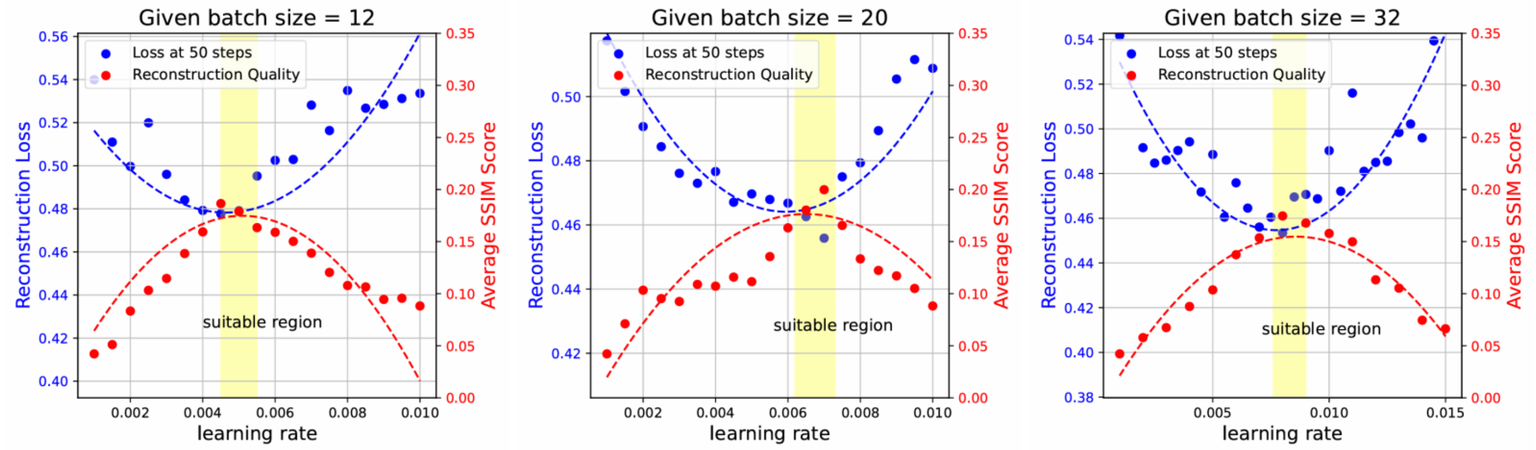
(a) Reconstructions from ResNet trained on 50 images using Buzaglo et al.'s, SSIM = 0.0297

(b) Reconstructions from ResNet trained on 50 images using Loo et al.'s, SSIM = 0.0774

(c) Reconstructions from ResNet trained on 50 images using SimuDy, SSIM = 0.1982

Reconstructions with Unknown Training Hyper-parameters

- Lower value of loss at 50 steps (blue) indicates better reconstruction performance (red).
- We preset $|\mathcal{B}|$ and tune η by grid search based on the first dozens of steps' loss.



Reconstructions with Unknown Dataset Size

- The contribution of each data point from the dummy dataset to the parameter changes is calculated by the norm of overall gradients throughout training.
- The matched data's average norm of total gradients is 1.5458, ranging from 1.1204 to 1.9478. In contrast, for unmatched data, the average norm is only 0.5466, with extremes of 0.2172 to 0.8438.
- When the dataset size is unknown, we use a larger size setting and transform extra dummy images from random noise to insignificant images which have relatively small gradients during training.

(a) Top 40 images from the dummy dataset of size 60, SSIM = 0.2295

(b) Bottom 20 images from the dummy dataset of size 60

Conclusion & Future Work

- We propose SimuDy to reconstruct training data from parameters of trained DNNs, which are more practical than MLPs in realistic applications.
- Extensive experiments show that SimuDy outperforms previous methods when dealing with non-linear training dynamics. Additionally, we demonstrate our method's effectiveness and robustness with unknown hyper-parameter settings of model training.
- Find more effective solutions to optimization challenges caused by the increased uncertainty for decoupling gradients of more data.
- Characterize training dynamics more efficiently, thus minimizing computational resource demands, such as by leveraging the low-dimensional nature of the model training process.
- We hope our work will illuminate deep learning interpretability and stimulate further exploration into the relationship between memorization and generalization of DNNs on larger datasets.

Thanks!