Implicit Regularization in Deep Neural Networks: An Optimization Perspective

Kamaljeet Singh
Statistics & Data Science
The University of Arizona,
kamaljeetsingh@arizona.edu

Abstract

The generalization behavior of deep neural networks is currently a critical area of research, especially as studies increasingly focus on it from the perspective of over-parameterization. This approach sheds light on how models with many parameters can still generalize effectively to unseen data, even after over-fitting. However, recent research attributed this phenomenon to "Implicit Regularization" of the optimization algorithm [1, 8, 6, 9]. In this project, we aim to study implicit regularization in deep neural networks (DNNs) from an optimization perspective. Fitting a neural network with more parameters than data points leads to the challenge of parameter unidentifiability. This results in a highly non-convex optimization problem, often characterized by multiple local minima. Despite these complexities, it has been claimed that gradient descent converges to minimum norm solutions or solutions with flat minima. Additionally, it has been observed that adaptive optimization methods tend to generalize poorly compared to stochastic gradient descent (SGD)[5]. This combination of factors makes our investigation into implicit regularization particularly relevant and interesting. Additionally, most research has focused on one- or two-layer neural networks with specific loss functions on benchmark datasets. Our aim is to explore this phenomenon across different architectures and datasets. We also aim to explore how mini-batch training aids in regularization, which may explain the success of SGD.

1 Introduction

The generalization ability of deep neural networks, even when trained on noisy data, has emerged as a fascinating area of study in recent research. The phenomenon challenges traditional machine learning theories and raises compelling questions. In this project we aim to explore the role of optimization algorithms in shaping this behavior? However, it is not solely the optimization technique that is responsible for this behavior. The network architecture (whether deep or wide), along with over-parameterization, also plays a crucial role in this.

1.1 Mini-batch

Modern machine learning algorithms process huge amounts of data and train a very large number of parameters. This requires high computational effort at every step, especially for calculating gradients at every data point in the case of neural networks. However, this issue is addressed using mini-batches, where each step is based on a small subset of the data. Recent research suggests that mini-batches not only offer computational benefits but also contribute to good generalization behavior, as there is implicit regularization associated with mini-batch training. [8] suggest that SGD with random shuffling implicitly regularizes the training process by modifying the loss function to include a penalty on the norms of mini-batch gradients. This regularization helps the mean SGD iterate stay close to the gradient flow path, promoting better generalization.

Problem description. In this section, we describe the optimization problem under consideration. The problem is defined by a dataset, a corresponding loss function, and an iterative optimization algorithm. The details are as follows:

The dataset is denoted as $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where:

- $x_i \in \mathbb{R}^d$ represents the input features for the i-th data point.
- y_i denotes the label or target associated with the i-th data point.

Let $f(x, w)$ represent the neural network function parameterized by $w \in \mathbb{R}^p$, where $p \gg n$ (overparameterized setting). The function $f(x, w)$ is learned by minimizing the overall loss function $C(w)$, defined as:

$$C(w) = \sum_{i=1}^N C_i(w),$$

where $C_i(w)$ is the per-sample loss for the i-th data point. The loss function quantifies the error between the model's predictions and the true labels.

Common examples of loss functions include:

1. **Squared Loss Function:**

$$C(w) = \sum_{i=1}^N (y_i - f(x_i, w))^2,$$

which is commonly used for regression tasks.

2. **Cross-Entropy Loss:** For classification problems, the cross-entropy loss is often used. For instance, in binary classification, the loss is given by:

$$C(w) = -\sum_{i=1}^N \left[y_i \log(f(x_i, w)) + (1 - y_i) \log(1 - f(x_i, w)) \right],$$

where $f(x_i, w)$ outputs the predicted probability for class 1.

To minimize the loss function $C(w)$, we can use any version of the Gradient Descent algorithm, such as Full Gradient Descent, Mini-Batch Gradient Descent, or Gradient Descent with Momentum. The parameter update rule at iteration t is given by:

$$w_{t+1} = w_t - \eta \cdot p_t,$$

where p_t is the descent direction proposed by the corresponding algorithm, and η is the learning rate, which determines the step size for each update.

In the case of standard Gradient Descent, p_t is the gradient of the loss function with respect to the parameters w, given by:

$$p_t = \nabla C(w_t) = \left. \frac{\partial C(w)}{\partial w} \right|_{w=w_t}.$$

In the case of Mini-Batch Stochastic Gradient Descent (SGD), the parameter update rule is given by:

$$w_{t+1} = w_t - \eta \frac{1}{|B_t|} \sum_{i \in B_t} \nabla C_i(w_t),$$

where: B_t denotes the mini-batch of data samples selected at iteration t.
$|B_t|$ is the size of the mini-batch.
$\nabla C_i(w_t)$ is the gradient of the loss function $C_i(w)$ with respect to the model parameters w_t for the i-th sample in the mini-batch.

Related work. The generalization ability of over-parameterized neural networks is one of the main research focuses in modern machine learning. In [3], [4], and [2], the generalization properties of over-parameterized models are discussed thoroughly. These works make an honest attempt to bridge the gap between classical theory and the behavior of modern machine learning models, particularly from the perspectives of interpolation and over-parameterization.

In [6], the good generalization behavior of over-parameterized models was initially explored from the training dynamics perspective. It was claimed that the nature of the optimization problem in such models gives rise to implicit regularization, which penalizes gradient descent trajectories that exhibit large loss gradients. Furthermore, it was shown that this implicit regularization steers gradient descent toward solutions in flat minima. As a result, these solutions are robust to small perturbations in the parameters, enhancing the model's generalization ability. However, implicit regularization in mini-batch SGD was further explored in [8], which shows that implicit regularization is even stronger in minibatch SGD, in coherence with what was ususally observed in numerical experiments.

Moreover, some research also claims that gradient descent selects the minimum norm solution among all possible solutions (including local or global minima). This behavior was similarly observed for Stochastic Gradient Descent (SGD).

However, it has also been observed that adaptive optimization methods like Adam, RMSprop, etc., do not generalize as well as SGD. This has been reported in several studies [10] [7]. In [5], an approach was proposed to achieve both faster training and good generalization by switching to SGD during the second half of training. However, this raises several important questions regarding the generalization behavior of adaptive methods, such as:

- Why do adaptive optimization methods fail to generalize as well as SGD?
- Can we assert that the speed of adaptive methods comes at the cost of generalization?
- How does the norm of the solution evolve across different optimization techniques?
- Does momentum steer the solution toward different basins of attraction?
- How does the implicit regularization term behave for adaptive methods compared to SGD?

2 Methodology

In training neural networks using an infinitesimal step size, SGD follows the gradient flow of the loss function over the entire dataset. However, sometimes using a learning rate larger than the one optimal for minimizing the training error can still yield very good accuracy. This discrepancy arises because, with a small but finite learning rate, the mean SGD iterate stays close to the gradient flow path of a modified loss function. This modified loss consists of the original training loss and a regularization term that penalizes large batch gradients.

To support this argument, the author of [8] used backward error analysis as follows:

The modified loss $\tilde{C}_{\text{SGD}}(\omega)$ for Stochastic Gradient Descent (SGD) combines the original training loss $C(\omega)$ with an implicit regularization term penalizing large batch gradients. It is given as:

$$\tilde{C}_{\text{SGD}}(\omega) = C(\omega) + \frac{\eta}{4m} \sum_{k=0}^{m-1} \|\nabla \tilde{C}_k(\omega)\|^2. \tag{1}$$

Where:

- $C(\omega)$: Original training loss.
- ϵ: Learning rate.
- m: Number of batches per epoch.
- $\|\nabla \tilde{C}_k(\omega)\|^2$: The squared norm of the gradient for minibatch k.

To compare the modified losses for Gradient Descent (GD) and SGD, $\tilde{C}_{\text{SGD}}(\omega)$ is expanded as:

$$\tilde{C}_{\text{SGD}}(\omega) = C(\omega) + \frac{\eta}{4}\|\nabla C(\omega)\|^2 + \frac{\eta}{4m} \sum_{i=0}^{m-1} \|\nabla \tilde{C}_i(\omega) - \nabla C(\omega)\|^2. \tag{2}$$

Where:

- The second term, $\frac{\eta}{4}\|\nabla C(\omega)\|^2$, penalizes sharp regions of the full-batch gradient.
- The third term, $\frac{\eta}{4m} \sum_{i=0}^{m-1} \|\nabla \tilde{C}_i(\omega) - \nabla C(\omega)\|^2$, penalizes non-uniform gradients across minibatches.

These equations illustrate how SGD introduces implicit regularization, affecting both the sharpness and uniformity of the gradients.

However, for adaptive methods, this regularization term seems ineffective, as evident from the numerical experiments in [5]. We propose below modified loss for RMSProp

$$\tilde{C}_{rms}(\omega) = C(\omega) + \frac{\eta}{4m} \sum_{k=0}^{m-1} \|s_t \odot \nabla \tilde{C}_k(\omega)\|^2$$

where

$$s_t = \gamma s_{t-1} + (1-\gamma) g_t^2 \quad \text{(Running average of squared gradients)}$$

$$g_t = \nabla \tilde{C}(\omega_t) \quad \text{(Gradient of the loss function for the minibatch)}$$

3 Numerical experiment

To support our claims, we use the MNIST dataset and a 5-layer fully connected neural network with a total of p parameters. The input is a 28×28 image of a handwritten digit, resulting in an input dimension of 784. All layers except the last one use the ReLU activation function, while the last layer uses the Softmax activation function. The dataset consists of 10 classes (digits 0 through 9). The training and testing datasets each contain 10,000 samples. We then add 30% random noise to the data to allow the model to potentially overfit the noise and thereby observe its generalization behavior more clearly.

The following plot shows the training and testing errors over 300 epochs with a batch size of 64. The learning rate is 0.001 for all the three methods.

[b]0.45

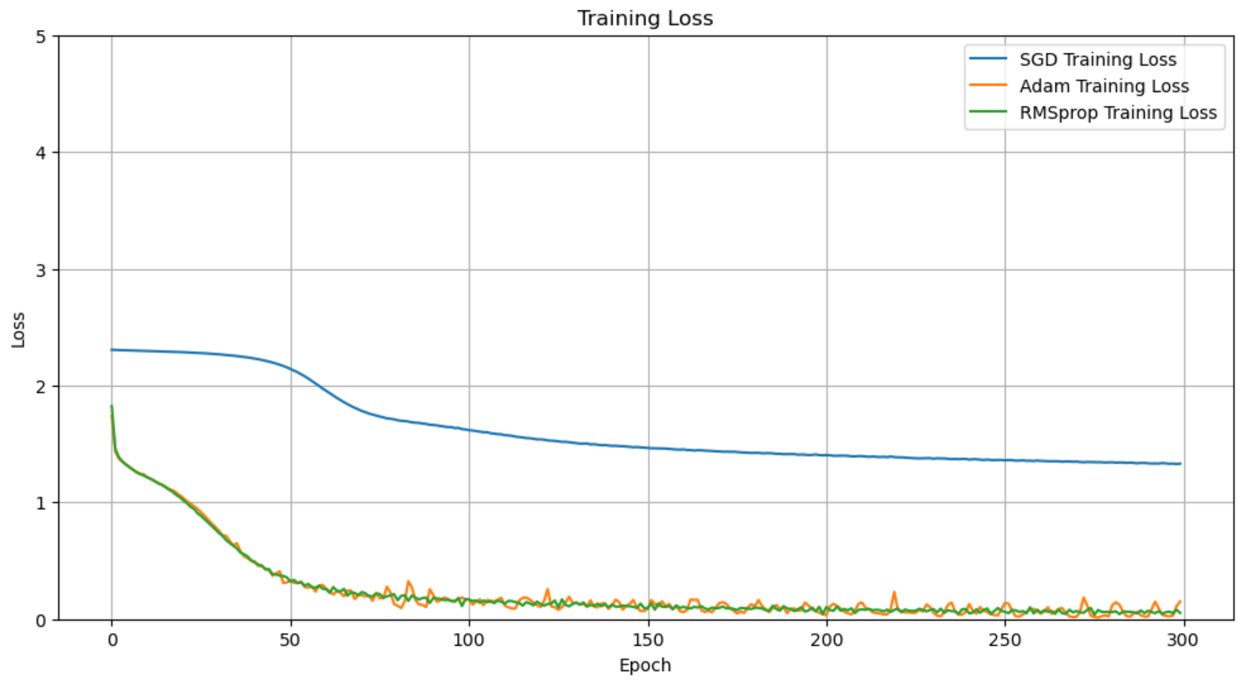

Figure 1: Training Error

[b]0.45

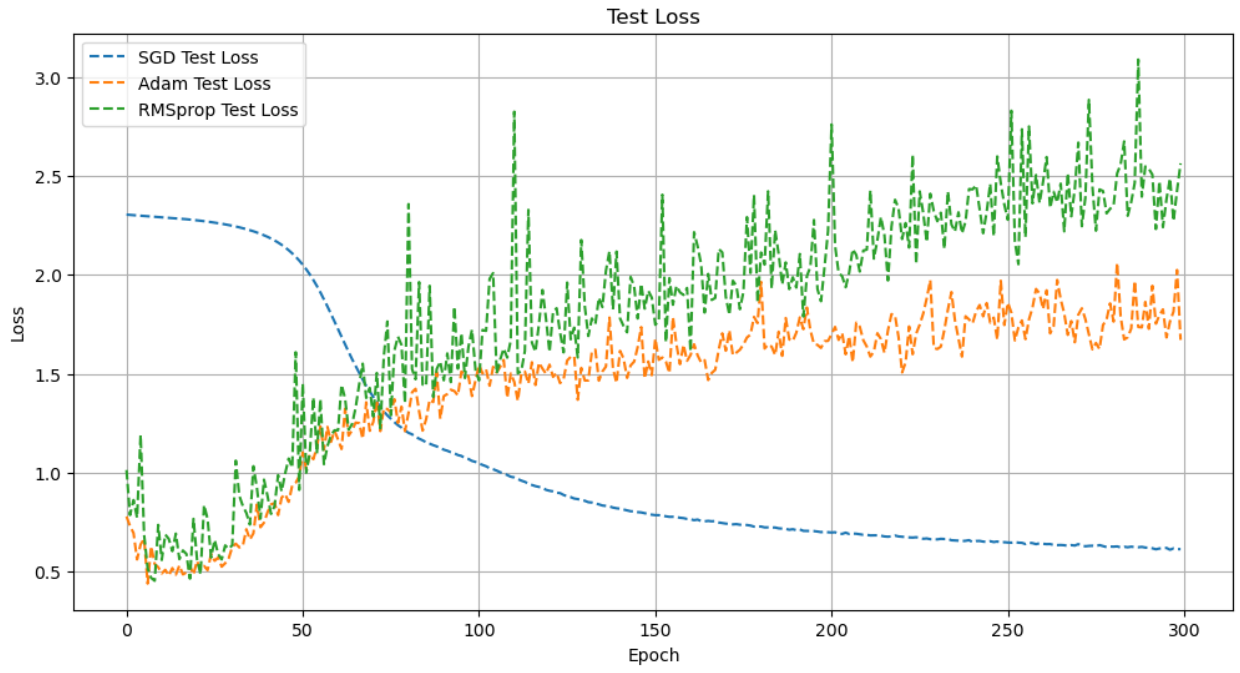

Figure 2: Testing Error

The implicit regularization term is plotted during the training step in this figure. The implicit regularization term for SGD is given by $\frac{\eta}{4m}\sum_{k=0}^{m-1}\|\nabla\tilde{C}_k(\omega)\|^2$, and for RMSprop, it is given by $\frac{\eta}{4m}\sum_{k=0}^{m-1}\|s_t\odot\nabla\tilde{C}_k(\omega)\|^2$.

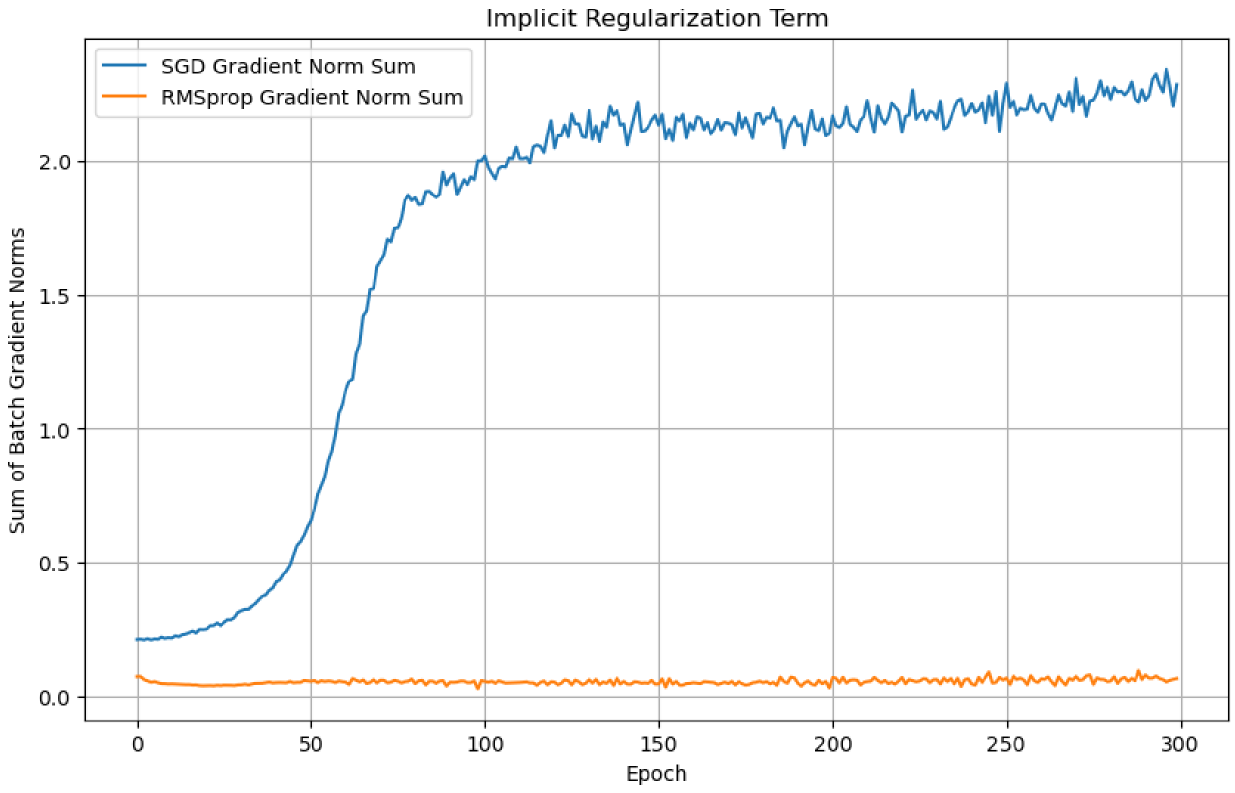

Figure 3: Implicit Regularization

We also plot the norm of the solution during training; the plot is shown below:

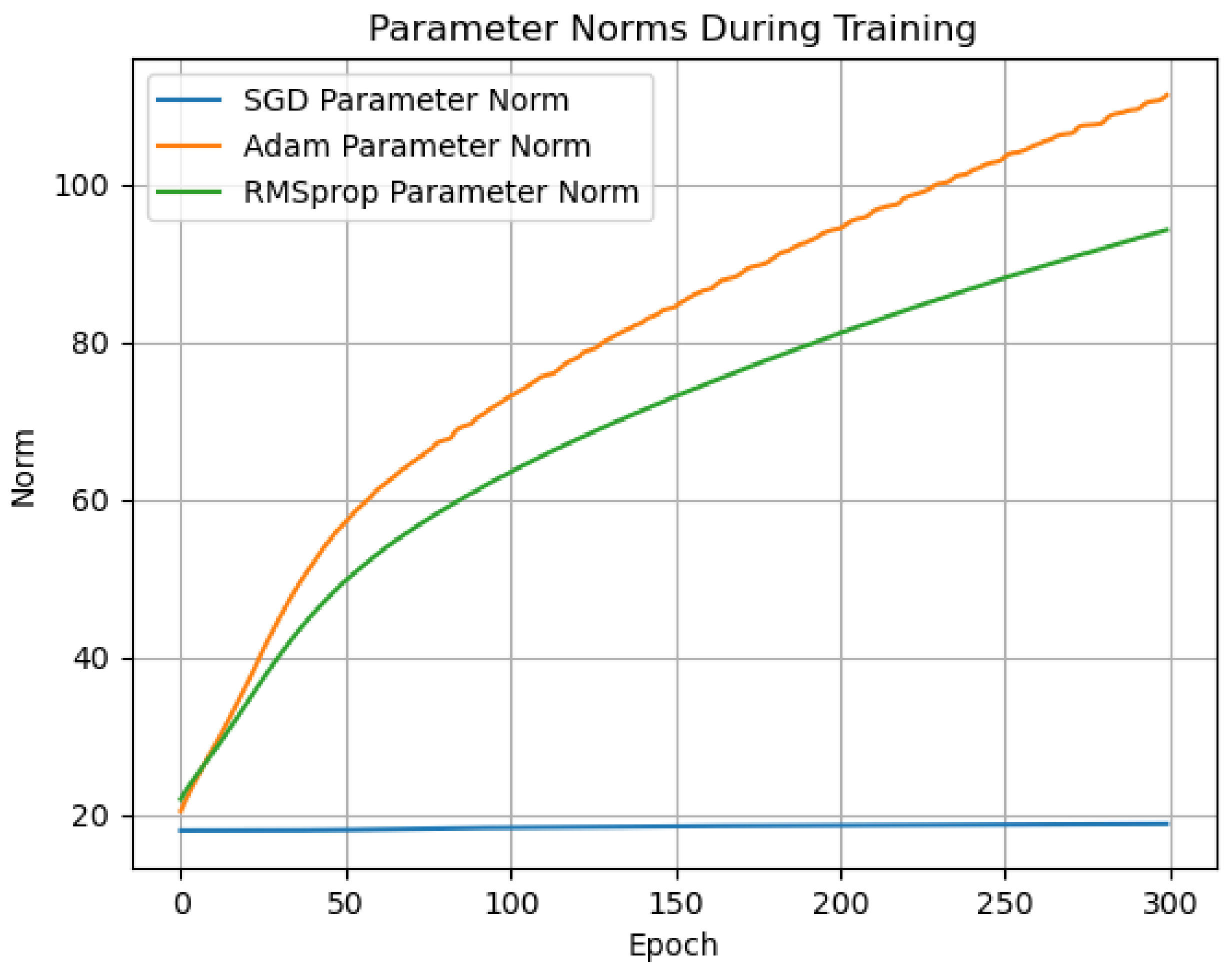

Figure 4: Parameter Norm L_2

4 Concluding Remarks

In this project, we conclude that the implicit regularization term is absent in adaptive optimization methods. This is clearly evident from Figure 3; however, there is no strong theoretical foundation to support our argument. Furthermore, the norm plot in Figure 4 supports our claim that small-norm solutions are favored by SGD. However, adaptive methods do not exhibit this property. In the future, we can strengthen our conjecture with theory using similar backward error analysis. Additionally, it would be interesting to investigate how explicit regularization impacts implicit regularization. Finally, exploring the effect of different architectures on implicit regularization would be a worthwhile direction for future research.

References

[1] David G. T. Barrett and Benoit Dherin. Implicit gradient regularization, 2022.

[2] Mikhail Belkin. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation, 2021.

[3] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, July 2019.

[4] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning, 2018.

[5] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd, 2017.

[6] Behnam Neyshabur. Implicit regularization in deep learning, 2017.

[7] Han Nguyen, Hai Pham, Sashank J. Reddi, and Barnabás Póczos. On the algorithmic stability and generalization of adaptive optimization methods, 2022.

[8] Samuel L. Smith, Benoit Dherin, David G. T. Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent, 2021.

[9] Bohan Wang, Qi Meng, Wei Chen, and Tie-Yan Liu. The implicit bias for adaptive optimization algorithms on homogeneous neural networks, 2021.

[10] Yingxue Zhou, Belhal Karimi, Jinxing Yu, Zhiqiang Xu, and Ping Li. Towards better generalization of adaptive gradient methods. *Advances in Neural Information Processing Systems*, 2020-December, 2020. Publisher Copyright: © 2020 Neural information processing systems foundation. All rights reserved.; 34th Conference on Neural Information Processing Systems, NeurIPS 2020 ; Conference date: 06-12-2020 Through 12-12-2020.