# 《CONCURRENT ADVERSARIAL LEARNING FOR LARGEBATCH TRAINING》ICLR2022文章阅读

[ICLR2022]https://arxiv.org/pdf/2106.00221.pdf

use adversarial learning to increase the batch size in large-batch training

## 1. 背景介绍

在进行大批量训练时，传统的优化器由于会倾向于收敛到尖锐的局部最小值，会导致模型的测试性能下降。虽然可以通过增加数据增强来减缓这个问题，但数据增强的效果会随着批量大小的增加而减弱。对抗学习是一种可以让模型的决策表面更加平滑，偏向于收敛到更加平坦的区域的方法。但是，对抗学习在每个训练步骤都需要进行至少两次的梯度计算，这就导致了对抗学习在大批量训练中的应用变得困难，因为这至少会使得训练时间翻倍。为了解决这个问题，作者提出了并行对抗学习（ConAdv）方法，避免了两次顺序的梯度计算，从而降低了训练时间。

## 2. 数据增强

作者首先讨论了数据增强在大批量训练中的作用。

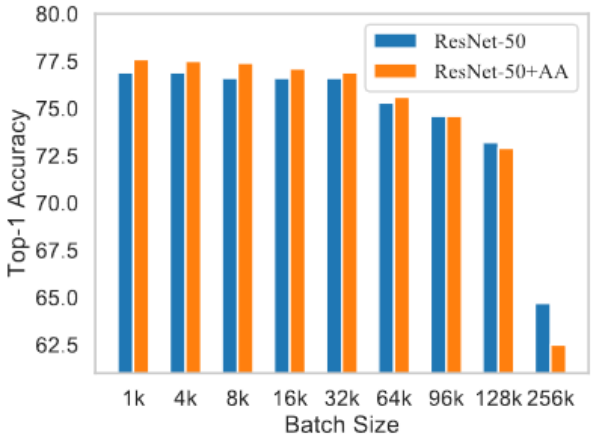



Figure 2: Augmentation Analysis

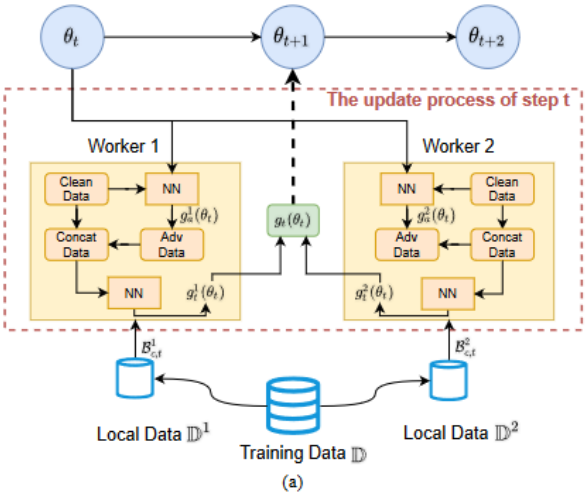
虽然AA在批次大小≤64K的情况下有助于提高泛化性能，但随着批次大小的增加，性能提升减小。更进一步，当批次大小足够大时（例如，128K或256K），它可能产生负面效果。原因在于，数据增强增加了训练数据的多样性，导致在进行较少的训练迭代时，模型的收敛速度会减慢。这些发现促使作者探索新的大批量训练方法，即并行对抗学习（ConAdv）。

## 3. DisAdv

对抗性学习可以被视为一种自动进行数据增强的方式。与定义固定规则来增强数据不同，对抗性学习执行基于梯度的对抗性攻击以找到对抗性示例。因此，对抗性学习导致更平滑的决策边界，这通常伴随着更flat的局部最小值。

在DisAdv策略中，每个工作节点首先从其本地数据集中采样小批量数据，然后从参数服务器下载权重，并使用这些权重来计算对抗性梯度。这个过程中，采用了一步投影梯度下降（PGD）方法来近似最优的对抗性示例，然后用这些对抗性样本以及干净的样本一起更新模型的权重。然而，该方法需要在每个步骤进行两次梯度计算，这在大批量训练中是不可取的，因为这样会消耗大量时间。

步骤图如下：



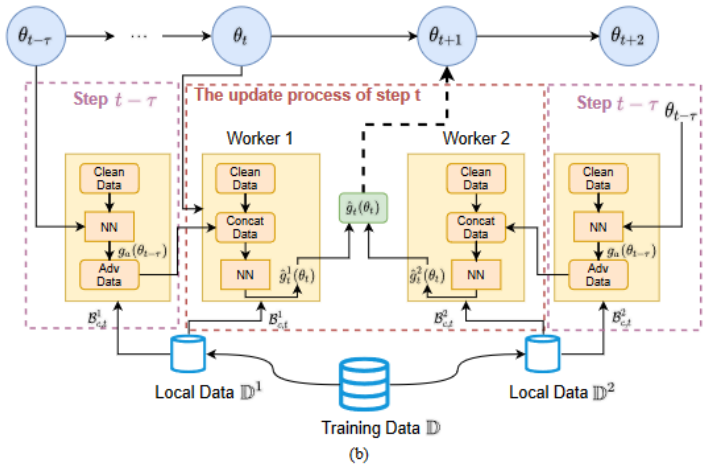

第一次梯度计算用于得到对抗样本，第二次梯度计算用于更新权重。

## 4. ConAdv

由于DisAdv的两次梯度计算无法并行，这使得对抗学习不适用于大批量训练，因此，作者提出了并行对抗学习（ConAdv）。主要想法是利用之前的权重来计算对抗样本，使得两次权重计算解耦，从而可以并行。

具体来说，它在每一步$t$都会使用过时的权重$\theta_{t-\tau}$来计算梯度并获得近似的对抗样本。

$$g_a(\theta_{t-\tau}) = \nabla_x \mathcal{L}(\theta_{t-\tau}; x_i, y_i), \quad \hat{x}_i(\theta_{t-\tau}) = x_i + \alpha \cdot g_a(\theta_{t-\tau})$$



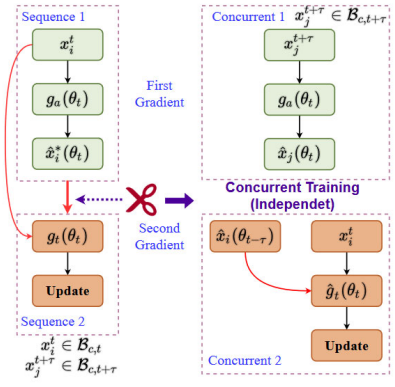




Figure 3: Vanilla Training and Concurrent Training

**Algorithm 1** ConAdv

**for** $t = 1, \cdots, T$ **do**
  **for** $x_i \in \mathcal{B}^k_{c,t}$ **do**
    Compute Loss:
    $\mathcal{L}(\theta_t; x_i, y_i)$ using main BN,
    $\mathcal{L}^k_a(\theta_t; \hat{x}_i(\theta_{t-\tau}), y_i)$ using adv BN,
    $\mathcal{L}_B(\theta_t) = \mathbb{E}_{\mathcal{B}^k_{c,t}} \mathcal{L}(\theta_t; x_i, y_i) +$
    $\mathbb{E}_{\mathcal{B}^k_{a,t}} (\hat{x}_i(\theta_{t-\tau}), y_i)$
    Minimize the $\mathcal{L}_B(\theta_t)$ and obtain $g^k_t(\theta_t)$
  **end for**
  **for** $x_i \in \mathcal{B}^k_{c,t+\tau}$ **do**
    Calculate adv gradient $g^k_a(\theta_t)$on $\mathcal{B}^k_{c,t+\tau}$
    Obtain adv examples $(\hat{x}_i(\theta_t), y_i)$
  **end for**
**end for**
Aggregate: $\hat{g}_t(\theta_t) = \frac{1}{K} \sum_{k=1}^{K} \hat{g}^k_t(\theta_t)$
Update weight $\theta_{t+1}$ on parameter sever