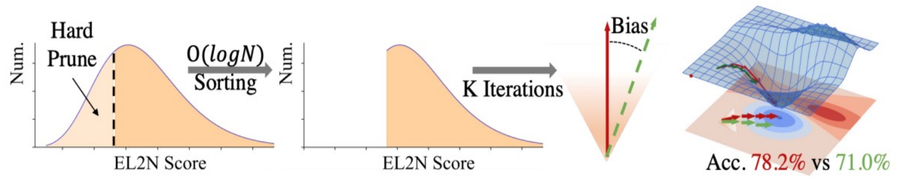
# InfoBatch: Lossless Training Speed Up by Unbiased Dynamic Data Pruning

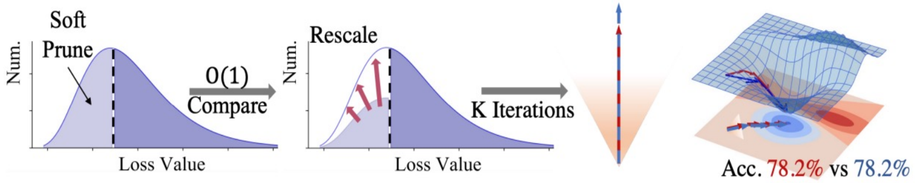
Ziheng Qin[1]*, Kai Wang1*, Yang You[1]‡

https://github.com/NUS-HPC-AI-Lab/InfoBatch

## How to Achieve Unbiased Dynamic Data Pruning?
## --Unbiased Prune and Rescale + Annealing




(a) Previous methods prune samples via setting some heuristic metrics (e.g. EL2N score). The hard pruning operation results in biased gradient expectation (green lines) and sub-optimal training results.




(b) InfoBatch soft prunes samples with small loss values and rescales the updates. Thereby InfoBatch maintains approximately the same gradient expectation direction (blue lines) as training on the original dataset.

$$\mathcal{P}_t(z) = \begin{cases} r, & \mathcal{H}_t(z) < \bar{\mathcal{H}}_t \\ 0, & \mathcal{H}_t(z) \geq \bar{\mathcal{H}}_t \end{cases}, \qquad \gamma_t(z) = 1/(1-\mathcal{P}_t(z))$$

Pruning Probability — Mean/Quantile of Loss — Rescaling Factor

By Importance Sampling, $\mathbb{E}[\nabla_\theta \mathcal{L}(\mathcal{S}_t)] \simeq \frac{|\mathcal{D}|}{|\mathcal{S}_t|}\mathbb{E}[\nabla_\theta \mathcal{L}(\mathcal{D})]$

Reduced step number $\frac{|\mathcal{S}_t|}{|\mathcal{D}|}$ is compensated by enlarged update $\frac{|\mathcal{D}|}{|\mathcal{S}_t|}$

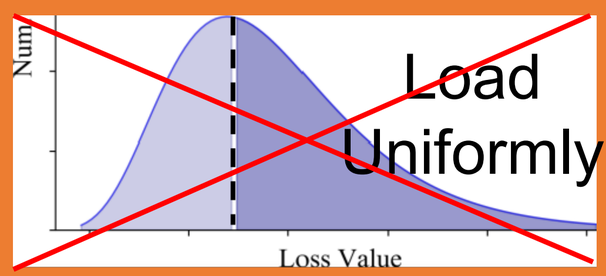
**Annealing:** To avoid remaining bias (when remaining epoch number is low), full data is used in the last epochs.

Zangwei Zheng[1], Jianyang Gu[1], Xiangyu Peng[1], Zhaopan Xu[1], Daquan Zhou[1], Lei Shang[2], Baigui Sun[2], Xuansong Xie[2]

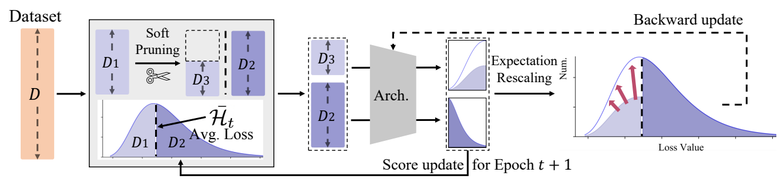
I like Deep Learning
But I don't like
High Computation Cost
& Long Training Time

# InfoBatch

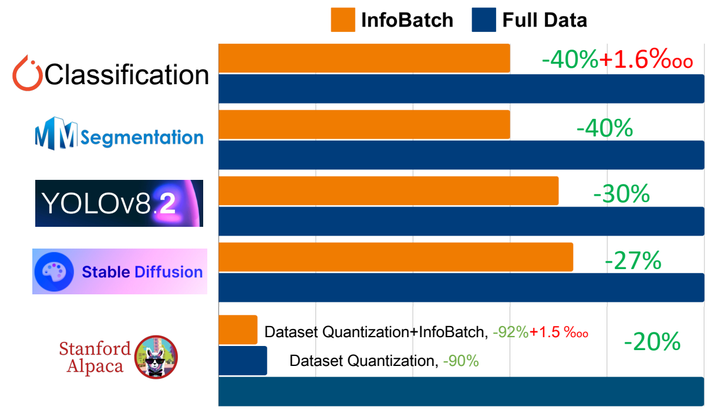
Accelerate hours with seconds!




All Samples Are Equally Important During Whole Training? No!




## Acceleration On Diverse Tasks




InfoBatch is able to accelerate training on diverse tasks, with high compatibility to various dataset/model/optimizers.

Plug-in-and-play with adding three lines!

```
train_data = infobatch.InfoBatch(train_data, num_epoch, ratio, delta)                    +
train_loader = torch.utils.data.DataLoader(train_data ..., sampler=train_data.sampler)   +
criterion = nn.CrossEntropyLoss(reduction='none')                                        +-
…   loss = train_data.update(loss)                                                       +
```

## Future Work

- InfoBatch studied on better sample efficiency. Other dimension of data efficiency are still to be explored, e.g. token efficiency, (domain) knowledge efficiency…
- InfoBatch's granularity (mean and quantile thresholds) and metric (loss) could be further explored.

NUS National University of Singapore

Alibaba.com