Scaling Forward Gradient With Local Losses

10.48550/arXiv.2210.03310, 2022

Mengye Ren, Simon Kornblith, Renjie Liao, Geoffrey Hinton

Reporter: Fengjiao Gong

November 24, 2022

Outline

1. Background
2. Forward Gradient Learning
3. Scaling with Local Loss
4. Experiments

Background

Automatic differentiation algorithm

- Reverse mode: two-phase

$$\boldsymbol{f}: \boldsymbol{\theta} \in \mathcal{R}^n \to \mathcal{R}^m$$
$$\boldsymbol{J_f} \in \mathcal{R}^{m\times n}$$
$$\boldsymbol{v} \in \mathcal{R}^m$$

$$\begin{array}{ccc} \boldsymbol{\theta} & \xrightarrow{\text{Forward}} & \boldsymbol{f}(\boldsymbol{\theta}) \\ \boldsymbol{v}^\intercal \boldsymbol{J_f}(\boldsymbol{\theta}) & \xleftarrow[\text{Backward}]{} & \boldsymbol{v} \end{array}$$

When in ML, $m = 1$ and $v = 1$ pytorch

$$\nabla f(\boldsymbol{\theta}) = \left[\frac{\partial f}{\partial \theta_1}, \dots, \frac{\partial f}{\partial \theta_n}\right]^\top$$

[reference] Numerical Optimization, 2nd edition, Springer, Chapter 8.2

Background

Automatic differentiation algorithm

- Reverse mode: two-phase
- Forward mode: only forward   jax.jvp

$$\boldsymbol{f}: \boldsymbol{\theta} \in \mathcal{R}^n \to \mathcal{R}^m$$
$$\boldsymbol{J_f} \in \mathcal{R}^{m \times n}$$
$$\boldsymbol{v} \in \mathcal{R}^n$$

$$\begin{matrix} \boldsymbol{\theta} \\ \boldsymbol{v} \end{matrix} \xrightarrow{\text{Forward}} \begin{matrix} \boldsymbol{f}(\boldsymbol{\theta}) \\ \boldsymbol{J_f}(\boldsymbol{\theta})\,\boldsymbol{v} \end{matrix}$$

When in ML, $m = 1$

$$scalar = \nabla f(\boldsymbol{\theta}) \cdot \boldsymbol{v}$$

directional gradient onto $\boldsymbol{v}$

$$\boldsymbol{e_1}, \ldots, \boldsymbol{e_i}, \ldots, \boldsymbol{e_n}$$

Background

Backpropagation:

$$E = \frac{1}{2}|y - d|^2 = \frac{1}{2}\left|(w - w^*)\, x\right|^2 = \frac{1}{2}|Wx|^2$$

$$\Delta W_{\mathrm{OL}} = -\eta \nabla E$$

- biologically implausible: brain does not form symmetric backward connections or perform synchronized computations
- incompatible with a massive level of model parallelism, and restricts potential hardware designs
- implicit form for the objective function - reinforcement learning

Background

Stochastic methods approximating gradient(on average)

- Weight perturbation: noise is added directly to the weight matrix

$$E'_{\mathrm{WP}} = \frac{1}{2}|(W+\psi)x|^2$$
$$\Delta W_{\mathrm{WP}} = -\frac{\eta}{\sigma^2}\left(E'_{\mathrm{WP}} - E\right)\psi$$

- Node perturbation: noise is added to the output of each unit

$$E'_{\mathrm{NP}} = \frac{1}{2}|Wx+\xi|^2$$
$$\Delta W_{\mathrm{NP}} = -\frac{\eta}{\sigma^2}\left(E'_{\mathrm{NP}} - E\right)\xi x^T$$

weight updates identical to that of direct gradient descent when averaged over all values of the noise(Gaussian)

[reference] Learning Curves for Stochastic Gradient Descent in Linear Feedforward Networks. Neural Comput 2005, Justin Werfel, Xiaohui Xie, H. Sebastian Seung;

Forward Gradient Learning

- Weight-perturbed forward gradient

Define "forward gradient" $g : \mathcal{R}^n \to \mathcal{R}^n$ for function $f : \mathcal{R}^n \to \mathcal{R}$ as

$$g_w(\boldsymbol{\theta}) = (\nabla f(\boldsymbol{\theta}) \cdot \boldsymbol{v})\boldsymbol{v} \qquad \text{(unbiased)}$$

where perturbation vector $\boldsymbol{v}$ is a multivariate random variable

$$\boldsymbol{E}[v_i v_j] = 0 \quad (i.i.d)$$
$$\boldsymbol{E}[v] = 0, \ Var[v] = 1$$

For function $f : \mathcal{R}^n \to \mathcal{R}^m$,

$$g_w\left(w_{ij}\right) = \left(\sum_{i'j'} \nabla w_{i'j'} v_{i'j'}\right) v_{ij}$$

[reference] Gradients without Backpropagation, 2022, Atilim Günes Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip H. S. Torr

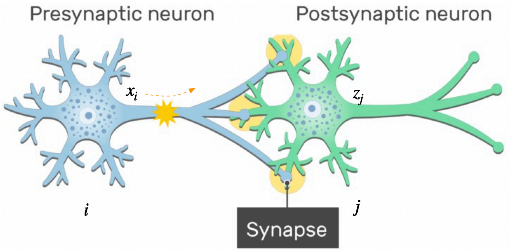
Forward Gradient Learning

- Activity-perturbed forward gradient this paper

$$g_a(w_{ij}) = x_i \left(\sum_{j'} \nabla z_{j'} u_{j'} \right) u_j \qquad \text{(unbiased)}$$

where

$$\begin{cases} x_i & \text{activity of the } i\text{-th pre-synaptic neuron} \\ z_j & \text{activity of the } j\text{-th post-synaptic neuron} \\ u_j & \text{perturbation of } z_j \end{cases}$$



less number of perturbation dimensions

Forward Gradient Learning

	Unbiased?	Avg. Variance (shared)	Avg. Variance (independent)
$g_w(\cdot)$	Yes	$\frac{pq+2}{N}V + (pq+1)S$	$\frac{pq+2}{N}V + \frac{pq+1}{N}S$
$g_a(\cdot)$	Yes	$\frac{q+2}{N}V + (q+1)S$	$\frac{q+2}{N}V + \frac{q+1}{N}S$

Table 1: Comparing weight (g_w) and activity (g_a) perturbation. V=dimension-wise avg. gradient variance, S=dimension-wise avg. squared gradient norm; p=fan-in; q=fan-out; N=batch size.

Both: variance still grows with larger networks

Scaling with Local Loss

One way: divide the network into submodules, each with a separate loss function.

- Blockwise loss: in depth $\Leftrightarrow$ "stop gradient" operator
  Each module consists of several layers.
  Each module has a loss function to update its parameters.

Scaling with Local Loss

- Blockwise loss: in depth ⇔ "stop gradient" operator
- Patchwise loss: spatial token
  A separate loss patchwise along these spatial dimensions
  Each spatial token represents a patch in the image

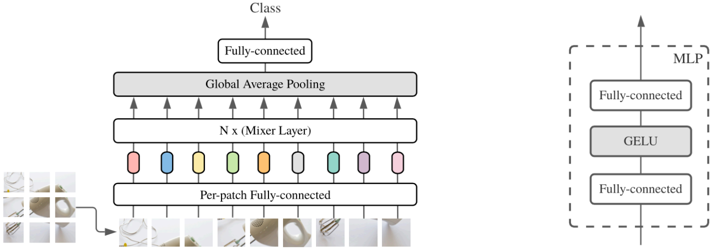


Figure 1: MLP-Mixer consists of per-patch linear embeddings, Mixer layers, and a classifier head.

[reference] MLP-Mixer: An all-MLP Architecture for Vision. NeurIPS 2021, Google Research, Brain Team

Scaling with Local Loss

- Blockwise loss: in depth
- Patchwise loss: spatial token
- Groupwise loss: feature channels
  Split channels into groups and each group has a loss function.
  Channels communicate within each group.

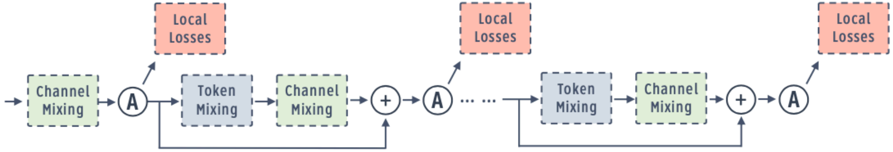


Figure 1: A LocalMixer network consists of several mixer blocks. A=Activation function (ReLU).

Suboptimal performances

- Feature aggregator: A global view of the inputs to make a decision

Scaling with Local Loss

Feature aggregator: keep dimensions

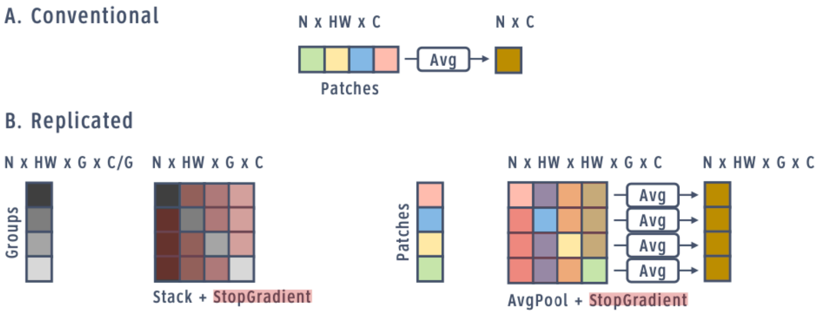


Figure 3: Feature aggregator designs. A) In the conventional design, average pooling is performed to aggregate features from different spatial locations. B) We propose the replicated design, features are first concatenated across groups and then averaged across spatial locations. We create copies of the same feature with different stop gradient masks so that we obtain more local losses instead of a global one. The stop gradient mask makes sure that perturbation in one spatial group corresponds to its loss function. The numerical value of the loss function is the same as the conventional design.

Scaling with Local Loss

Feature aggregator

- Channel group: *copied, communicated to one another, masked except active group itself*

$$\mathbf{x}_{p,g} = \left[\mathrm{StopGrad}\left(x_{p,1} \ldots x_{p,g-1}\right), x_{p,g}, \mathrm{StopGrad}\left(x_{p,g+1}, \ldots, x_{p,G}\right)\right]$$

- Spatial location: *copied, communicated, masked, then locally averaged*

$$\overline{\mathbf{x}}_{p,g} = \frac{1}{P}\left(\mathbf{x}_{p,g} + \sum_{p' \neq p} \mathrm{StopGrad}\left(\mathbf{x}_{p',g}\right)\right)$$

where p and g index the patches and groups respectively.

Scaling with Local Loss

Objective Function— Image Representation Learning

- Supervised classification loss: *attach a shared linear layer on top of the aggregated features for a cross entropy loss*

$$L^s_{p,g} = -\sum_k t_k \log \operatorname{softmax}\left(W\overline{\mathbf{x}}_{p,g}\right)_k$$

- Contrastive InfoNCE loss: *attach a linear feature projector*

$$L^c_{p,g} = -\sum_n \log \frac{\left(W\overline{\mathbf{x}}^{(1)}_{n,p,g}\right)^\top \operatorname{StopGrad}\left(W\overline{\mathbf{x}}^{(2)}_n\right)}{\sum_m \left(W\overline{\mathbf{x}}^{(1)}_{n,p,g}\right)^\top \operatorname{StopGrad}\left(W_{\mathbf{x}^{(2)}_m}\right)}$$

where $x^{(1)}_n$ and $x^{(2)}_n$ are two different views of the n-th sample.

LocalMixer

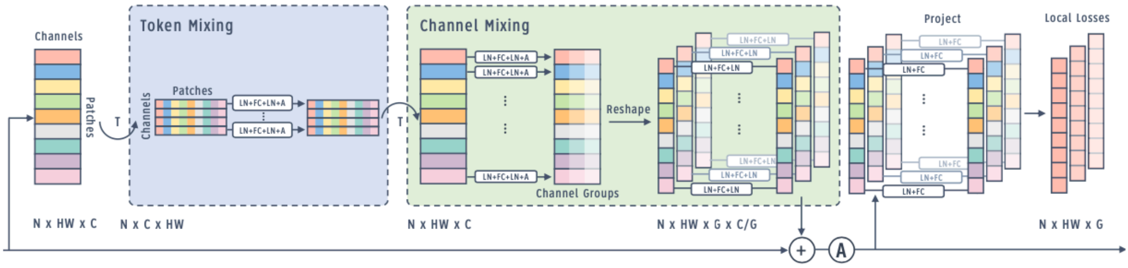


Figure 2: A LocalMixer residual block with local losses. Token mixing consists of a linear layer and channels are grouped in the channel mixing layers. Layer norm is applied before and after every linear layer. LN=Layer Norm; FC=Fully Connected layer; A=Activation function (ReLU); T=Transpose.

Network architecture

- FC: each spatial patch performs computations without interfering with others
- Last Layer: always uses backprop to update weights

Experiments

- Dataset

	sample	*size*	*class*	*learning*
MNIST	70000	28×28	10	supervised
CIFAR-10	60000	32×32	10	supervised + contrastive
ImageNet	1.3million	224×224(*resized*)	1000	supervised + contrastive

- LocalMixer architecture for each dataset

Type	Blocks	Patches	Channels	Groups	Params	Dataset
LocalMixer S/1/1	1	1×1	256	1	272K	MNIST
LocalMixer M/1/16	1	1×1	512	16	429K	MNIST
LocalMixer M/8/16	4	8×8	512	16	919K	CIFAR-10
LocalMixer L/8/64	4	8×8	2048	64	13.1M	CIFAR-10
LocalMixer L/32/64	4	32×32	2048	64	17.3M	ImageNet

Table 2: LocalMixer Architecture Details

Experiments

- Baselines

	abbr	*description*
BP	Backprop	standard
L-BP	Local Backprop	adds local losses
LG-BP	Local Greedy Backprop	adds stop gradient operators
FA	Feedback Alignment	standard
DFA	Direct Feedback Alignment	random & fixed backward weights
L-FA	Local Feedback Alignment	adds local losses
LG-FA	Local Greedy Feedback Alignment	adds a stop gradient
FG-W	Weight-perturbed forward gradient	
FG-A	Activity-perturbed forward gradient	
LG-FG-W	Local Greedy FG-W	adds local objective functions
LG-FG-A	Local Greedy FG-A	adds local objective functions

Experiments

Dataset	**MNIST**	**MNIST**	**CIFAR-10**	**ImageNet**
Network	**S/1/1**	**M/1/16**	**M/8/16**	**L/32/64**
Metric	Test / Train Err. (%)	Test / Train Err. (%)	Test / Train Err. (%)	Test / Train Err. (%)
BP	2.66 / 0.00	2.41 / 0.00	33.62 / 0.00	36.82 / 14.69
L-BP	2.38 / 0.00	2.16 / 0.00	30.75 / 0.00	42.38 / 22.80
LG-BP	2.43 / 0.00	2.81 / 0.00	33.84 / 0.05	54.37 / 39.66
BP-free algorithms				
FA	**2.82** / **0.00**	2.90 / **0.00**	39.94 / 28.44	94.55 / 94.13
L-FA	3.21 / **0.00**	2.90 / **0.00**	39.74 / 28.98	87.20 / 85.69
LG-FA	3.11 / **0.00**	**2.50** / **0.00**	39.73 / 32.32	85.45 / 82.83
DFA	3.31 / **0.00**	3.17 / **0.00**	38.80 / 33.69	91.17 / 90.28
FG-W	9.25 / 8.93	8.56 / 8.64	55.95 / 54.28	97.71 / 97.58
FG-A	3.24 / 1.53	3.76 / 1.75	59.72 / 41.29	98.83 / 98.80
LG-FG-W	9.25 / 8.93	5.66 / 4.59	52.70 / 51.71	97.39 / 97.29
LG-FG-A	3.24 / 1.53	2.55 / **0.00**	**30.68** / **19.39**	**58.37** / **44.86**

Table 3: Supervised learning for image classification

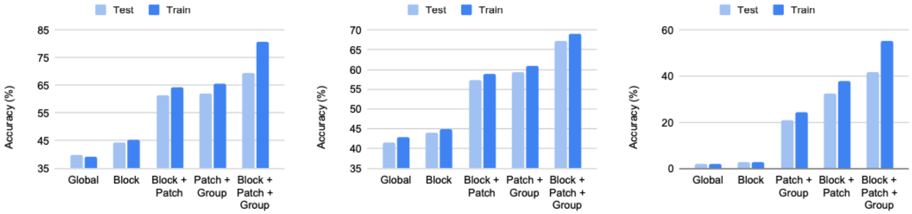
Experiments

Dataset Network Metric	CIFAR-10 M/8/16 Test / Train Err. (%)	CIFAR-10 L/8/64 Test / Train Err. (%)	ImageNet L/32/64 Test / Train Err. (%)
BP	24.11 / 21.08	17.53 / 13.35	55.66 / 49.79
L-BP	24.69 / 21.80	19.13 / 13.60	59.11 / 52.50
LG-BP	29.63 / 25.60	23.62 / 16.80	68.36 / 62.53
BP-free algorithms			
FA	45.87 / 44.06	67.93 / 65.32	82.86 / 80.21
L-FA	37.73 / 36.13	31.05 / 26.97	83.18 / 79.80
LG-FA	36.72 / 34.06	30.49 / 25.56	82.57 / 79.53
DFA	46.09 / 42.76	39.26 / 37.17	93.51 / 92.51
FG-W	53.37 / 51.56	50.45 / 45.64	91.94 / 89.69
FG-A	54.59 / 52.96	56.63 / 56.09	97.83 / 97.79
LG-FG-W	52.66 / 50.23	52.27 / 48.67	91.36 / 88.81
LG-FG-A	**32.88 / 29.73**	**26.81 / 23.90**	**73.24 / 66.89**

Table 4: Self-supervised contrastive learning with linear readout

Experiments

Effect of local losses

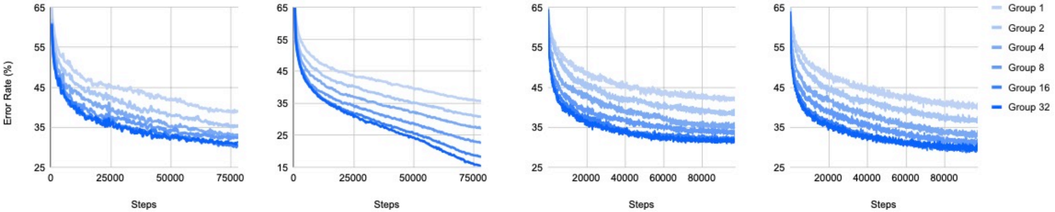


(a) CIFAR-10 Supervised M/8 (b) CIFAR-10 Contrastive M/8 (c) ImageNet Supervised L/32

Figure 6: Effect of adding local losses at different locations on the performance of forward gradient

Experiments

Effect of local groups



(a) Supervised Test (b) Supervised Train (c) Contrastive Test (d) Contrastive Train

Figure 7: Error rate of M/8/* during CIFAR-10 training using different number of groups.

Thanks!