Forward gradients

Gradients without Backpropagation

Atılım Güneş Baydin [1] **Barak A. Pearlmutter** [2] **Don Syme** [3] **Frank Wood** [4] **Philip Torr** [5]

SCALING FORWARD GRADIENT WITH LOCAL LOSSES

Mengye Ren[1*], **Simon Kornblith**[2], **Renjie Liao**[3], **Geoffrey Hinton**[2,4]
[1]NYU, [2]Google, [3]UBC, [4]Vector Institute

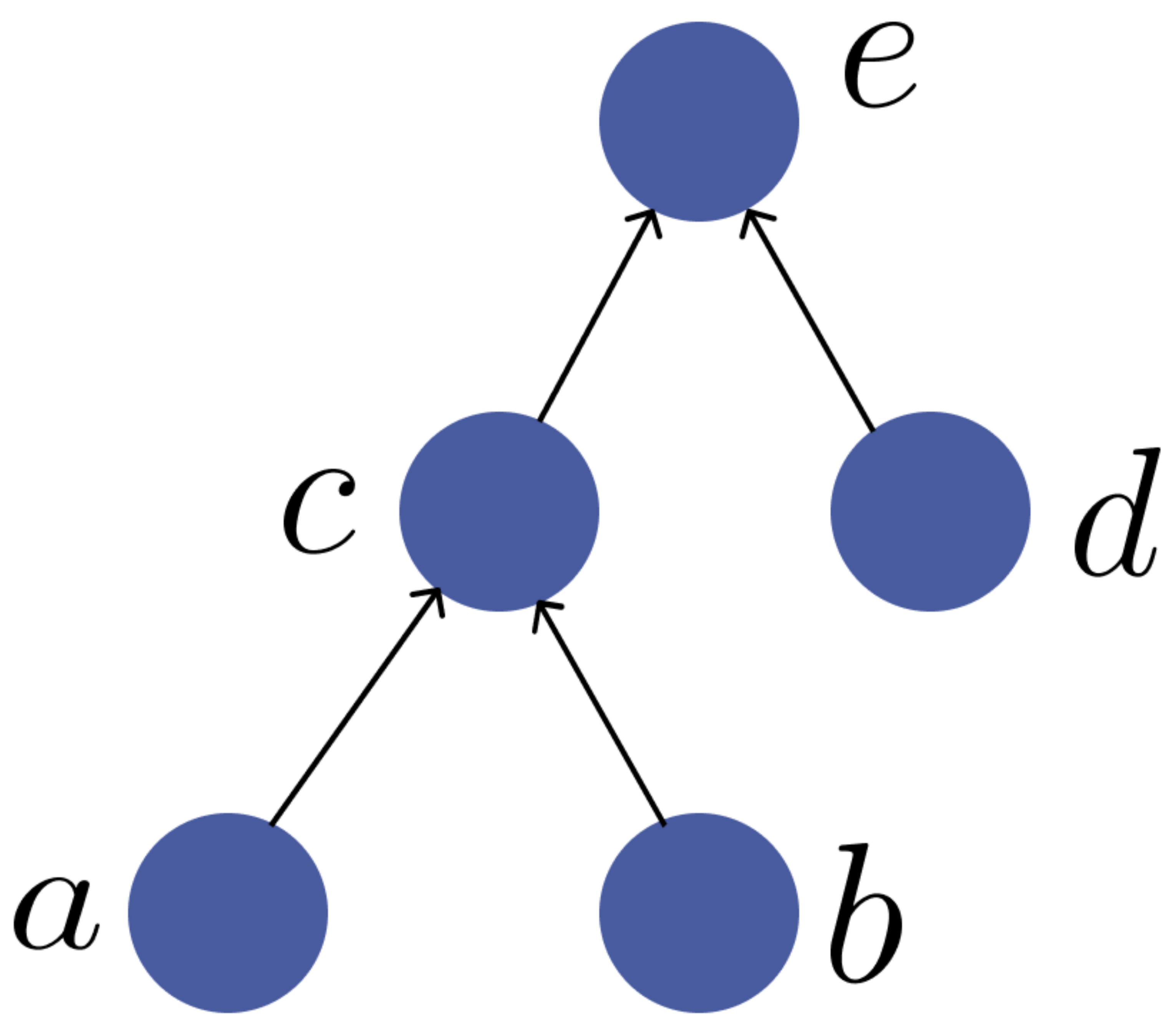
Seth, Tea Talk 07/11/23

Forward or backward?

$$\frac{\partial \mathbf{f}_\theta}{\partial \theta_{l,i}} = \frac{\partial \mathbf{f}_\theta}{\partial \mathbf{x}_L} \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_{L-1}} \cdots \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{x}_l} \frac{\partial \mathbf{x}_l}{\partial \theta_{l,i}}$$

- **Backprop**: compute function $\mathbf{f}_\theta(\mathbf{x}_0)$ and right-multiply, working backwards through the network
- **Forwardprop:** multiply Jacobians, $\left\{\frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{x}_l}\right\}$, from R to L in same forward pass as evaluating $\mathbf{f}_\theta(\mathbf{x}_0)$

Why backprop?

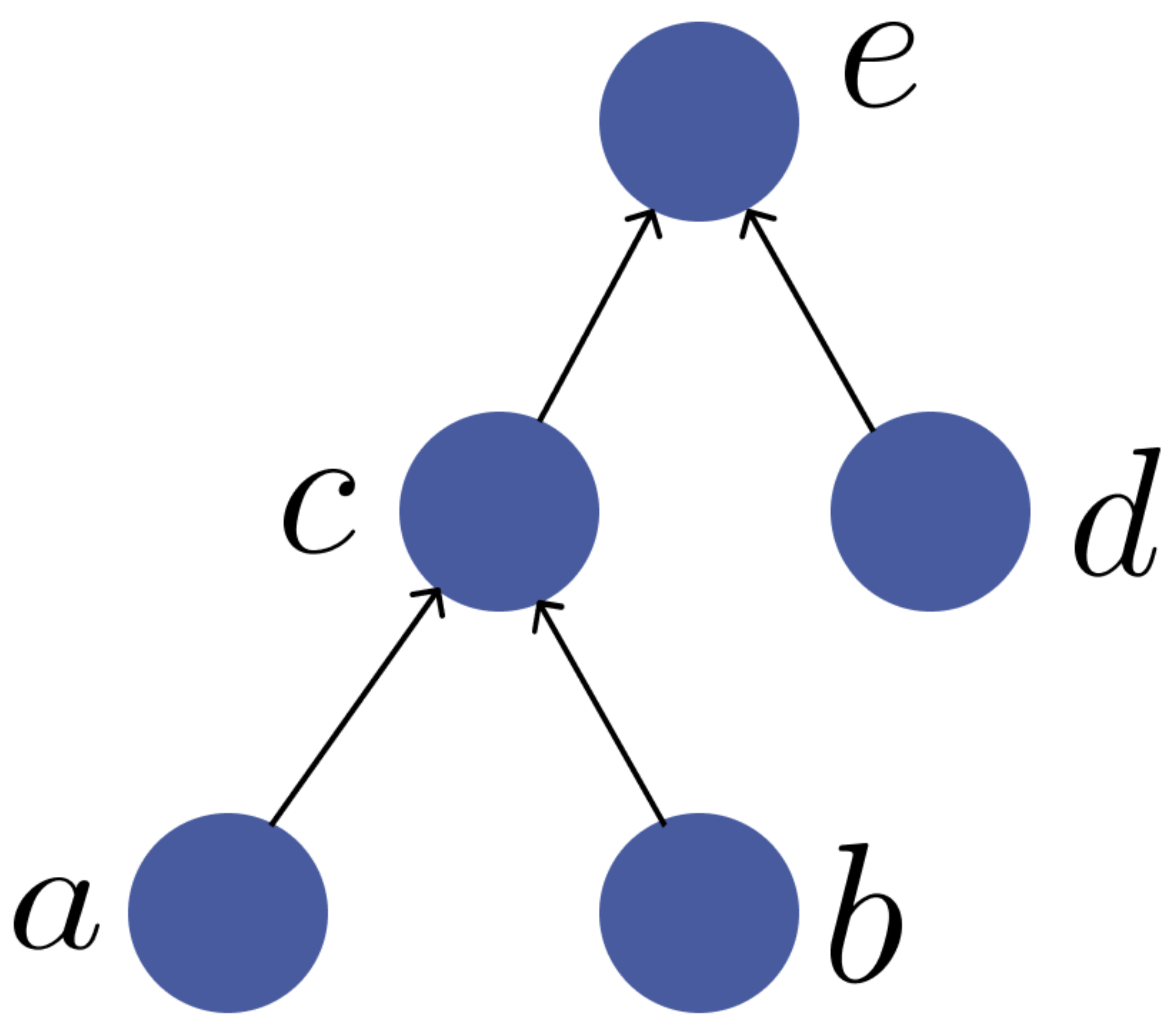


- To compute e.g. $\partial e / \partial a$ with forwardprop
  - First compute $\partial c / \partial a$ and $c(a, b)$
  - Then compute $\partial e / \partial c$ and "push-forward"

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial c} \cdot \frac{\partial c}{\partial a}$$

- Memory efficient: don't need to store intermediates $c(a, b)$
- But if we now want e.g. $\partial e / \partial b$, we have to repeat this procedure from scratch

Why backprop?



- With backprop
  - Evaluate e once,
  - Use this (+ intermediates) to compute gradients w.r.t all variables

Why backprop?

- In general, to compute gradient of function $\mathrm{f}_\theta : \mathbb{R}^N \to \mathbb{R}^M$ w.r.t $\theta \in \mathbb{R}^D$

	Forward prop	**Backprop**
Time	$O(FD)$	$O(FM)$
Memory	$O(1)$	$O(L)$

where F is complexity of evaluating $\mathrm{f}_\theta(\cdot)$ and L is depth

- In ML, usually $M << D$ and so backprop is more time-efficient

Directional derivatives

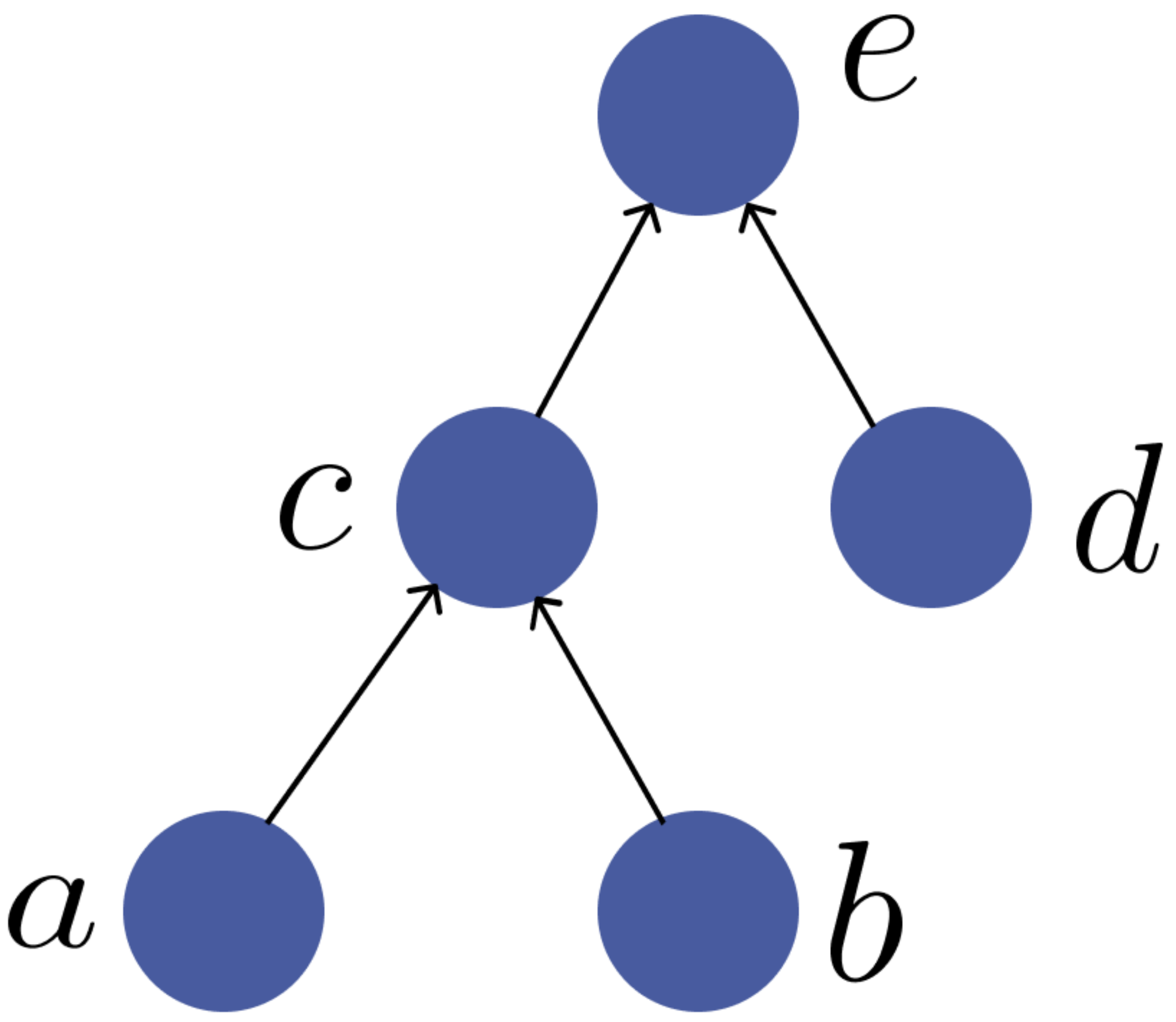
Gradients without Backpropagation

Atılım Güneş Baydin [1] Barak A. Pearlmutter [2] Don Syme [3] Frank Wood [4] Philip Torr [5]

- To make forward prop efficient, Baydin et al use it for the directional derivative $\nabla \mathbf{f}_\theta \cdot \mathbf{v}$ along a random vector $\mathbf{v} \sim p(\mathbf{v})$
- If $\{v_i\}$ are iid, zero-mean and unit-variance; the expression $(\nabla \mathbf{f}_\theta \cdot \mathbf{v}) \cdot \mathbf{v}$ is an unbiased estimator for $\nabla \mathbf{f}_\theta$ (proof to follow)
- Note that $\nabla \mathbf{f}_\theta$ is never actually computed

Baydin, Pearlmutter, Syme, Wood, Torr. Gradients without Backpropagation, 2022

Directional derivatives



- To estimate $\nabla e = \left[\frac{\partial e}{\partial a}, \frac{\partial e}{\partial b}, \frac{\partial e}{\partial d}\right]^T$

$$
\begin{aligned}
\nabla e &= \left[\frac{\partial e}{\partial a}, \frac{\partial e}{\partial b}, \frac{\partial e}{\partial d}\right]^T \\
&\approx (\nabla e \cdot \mathbf{v}) \cdot \mathbf{v}, \quad \text{where } \mathbf{v} = [v_a, v_b, v_d]^T \sim p(\mathbf{v}) \\
&\approx \left(\frac{\partial e}{\partial a} \cdot v_a + \frac{\partial e}{\partial b} \cdot v_b + \frac{\partial e}{\partial d} \cdot v_d\right) \cdot \mathbf{v} \\
&\approx \left(\frac{\partial e}{\partial c}\left(\frac{\partial c}{\partial a} \cdot v_a + \frac{\partial c}{\partial b} \cdot v_b\right) + \frac{\partial e}{\partial d} \cdot v_d\right) \cdot \mathbf{v}
\end{aligned}
$$

Unbiasedness proof

- The inner product is

$$\nabla \mathbf{f}_\theta \cdot \mathbf{v} = \sum_k \frac{\partial f}{\partial \theta_k} v_k = \frac{\partial f}{\partial \theta_i} v_i + \sum_{j \neq i} \frac{\partial f}{\partial \theta_j} v_j$$

- Therefore, the *i*th element of $\mathbb{E}_{\mathbf{v}}\left[(\nabla \mathbf{f}_\theta \cdot \mathbf{v}) \cdot \mathbf{v}\right]$ is

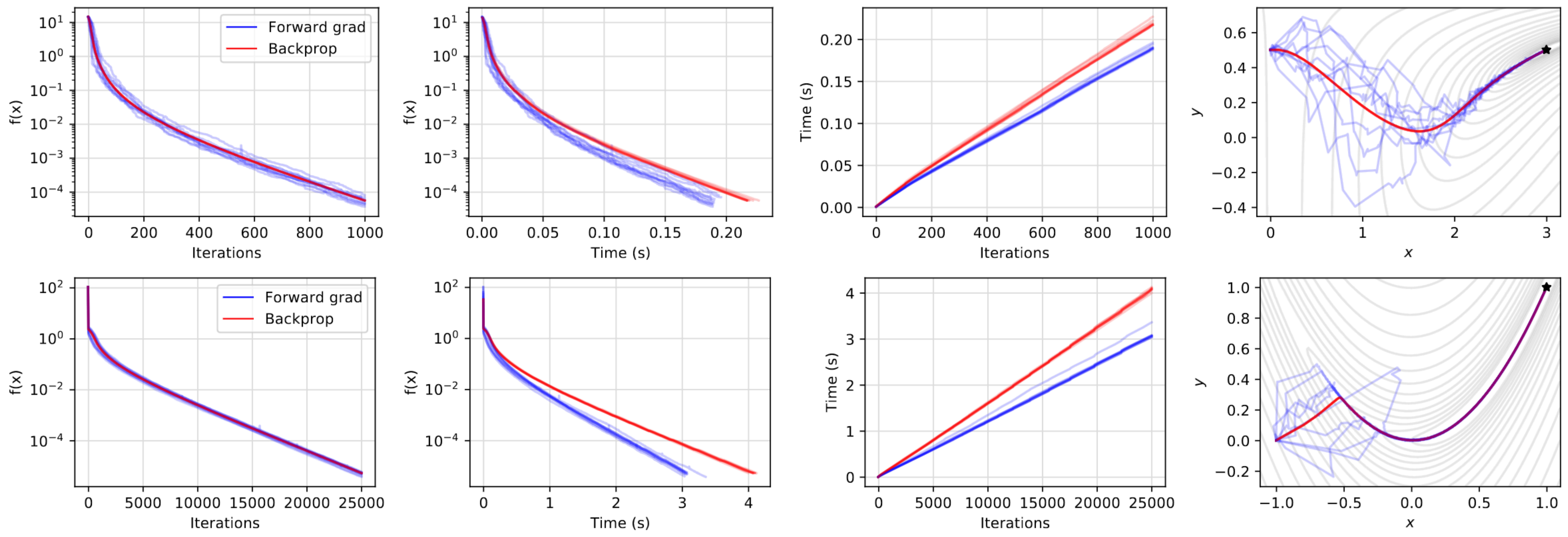
$$\frac{\partial f}{\partial \theta_i} \mathbb{E}\left[v_i^2\right] + \sum_{j \neq i} \frac{\partial f}{\partial \theta_j} \mathbb{E}\left[v_i v_j\right]$$

- $p(\mathbf{v})$ is chosen s.t. v_i are mean zero, variance one, and i.i.d
    - **Mean zero + variance one** — first expectation is 1
    - **I.i.d + mean zero** — second expectation is 0
- Thus *i*th element of $\mathbb{E}_{\mathbf{v}}\left[(\nabla \mathbf{f}_\theta \cdot \mathbf{v}) \cdot \mathbf{v}\right]$ is *i*th element of the true gradient

Baydin, Pearlmutter, Syme, Wood, Torr. Gradients without Backpropagation, 2022

Results

Toy optimisation problems



Baydin, Pearlmutter, Syme, Wood, Torr. Gradients without Backpropagation, 2022

Results

MNIST, MLP

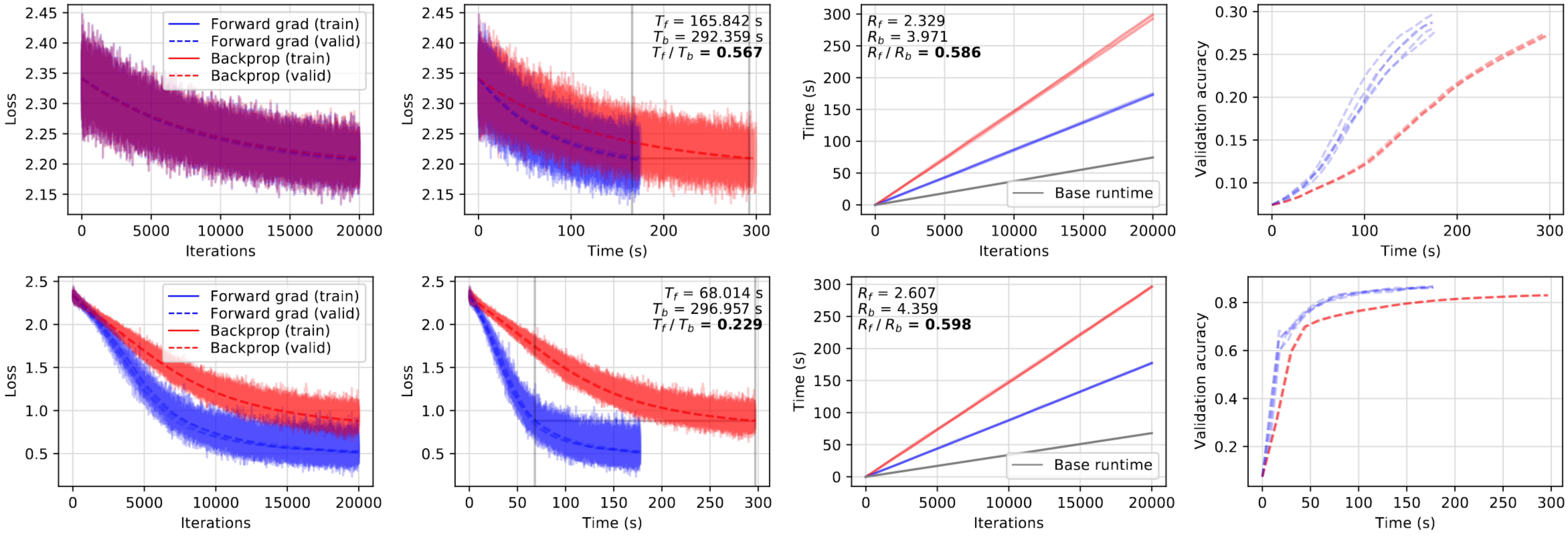


Figure 4. Comparison of forward gradient and backpropagation for the multi-layer NN, showing two learning rates. Top row: learning rate 2×10^{-5}. Bottom row: learning rate 2×10^{-4}. Showing five independent runs per experiment.

Baydin, Pearlmutter, Syme, Wood, Torr. Gradients without Backpropagation, 2022

Results

MNIST, CNN

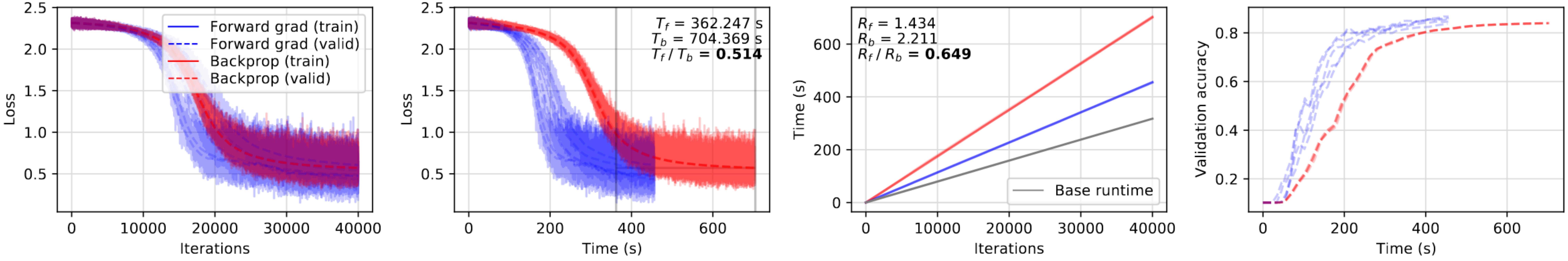


Figure 5. Comparison of forward gradient and backpropagation for the CNN. Learning rate 2×10^{-4}. Showing five independent runs.

Baydin, Pearlmutter, Syme, Wood, Torr. Gradients without Backpropagation, 2022

Results

Runtime and memory

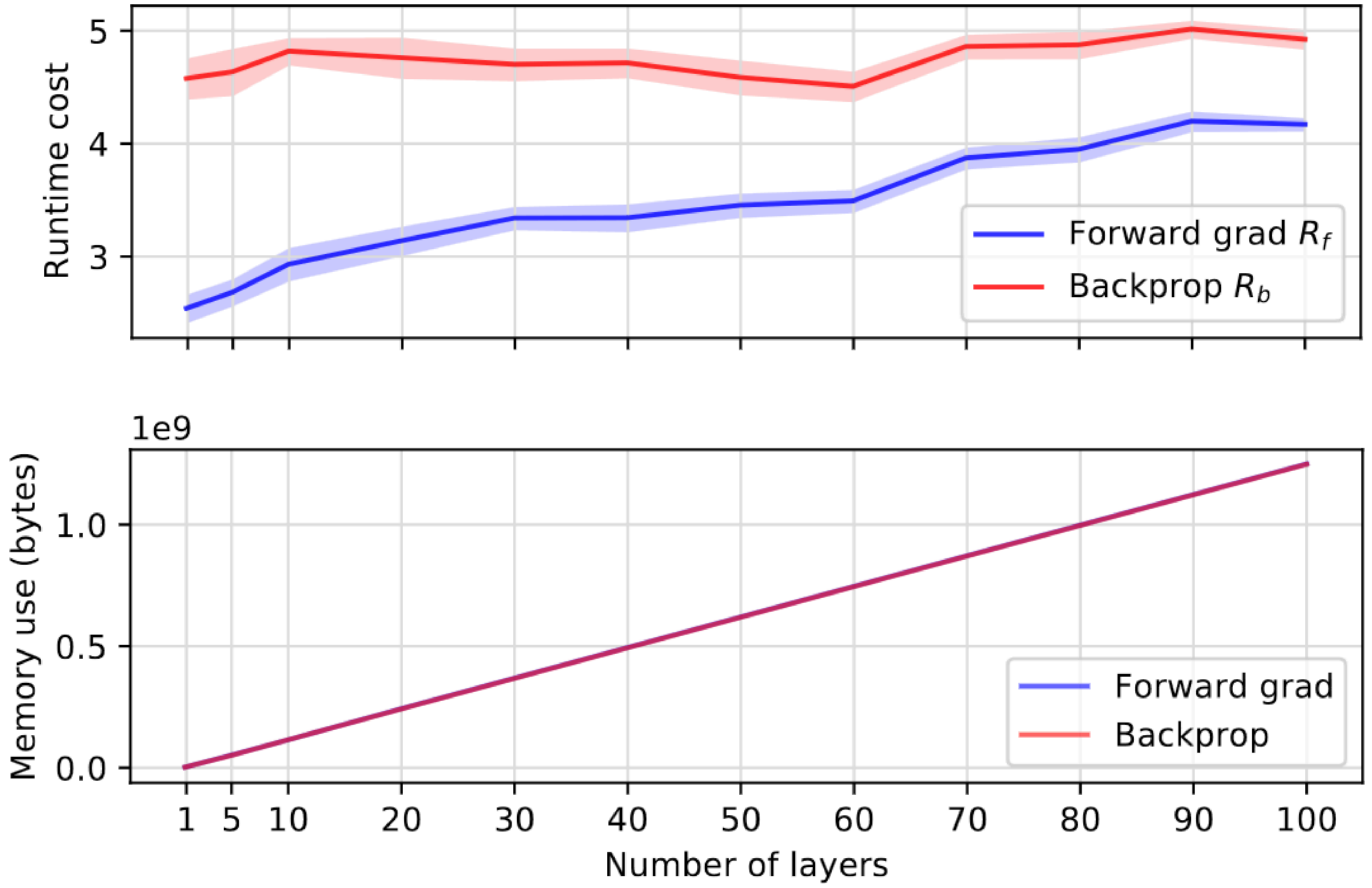


Figure 6. Comparison of how the runtime cost and memory usage of forward gradients and backpropagation scale as a function NN depth for the MLP architecture where each layer is of size 1024. Showing mean and standard deviation over ten independent runs.

Baydin, Pearlmutter, Syme, Wood, Torr. Gradients without Backpropagation, 2022

Scaling Forward Propagation

Mengye Ren[1]*, Simon Kornblith[2], Renjie Liao[3], Geoffrey Hinton[2,4]
[1]NYU, [2]Google, [3]UBC, [4]Vector Institute

- Reduces variance of forward gradients:
  - Apply perturbation vector $\mathbf{v}$ to activation gradients instead of weights
  - Add local losses on blocks, spatial patches and groups of channels
- "LocalMixer" architecture inspired by MLPMixer
- Experiments scaled to ImageNet

Results
Supervised learning

Dataset Network Metric	MNIST S/1/1 Test / Train Err. (%)	MNIST M/1/16 Test / Train Err. (%)	CIFAR-10 M/8/16 Test / Train Err. (%)	ImageNet L/32/64 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

BP = Backprop

L = Local losses

LG = Greedy local losses

FA = Feedback alignment (random, fixed backward weights)

DFA = Direct feedback alignment

FG = Forward gradients

W = Weight-perturbed (Baydin et al)

A = Activation-perturbed

Ren, Kornblith, Liao, Hinton. Scaling Forward Gradients with Local Losses. 2023

Results

Self-supervised learning

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

BP = Backprop

L = Local losses

LG = Greedy local losses

FA = Feedback alignment (random, fixed backward weights)

DFA = Direct feedback alignment

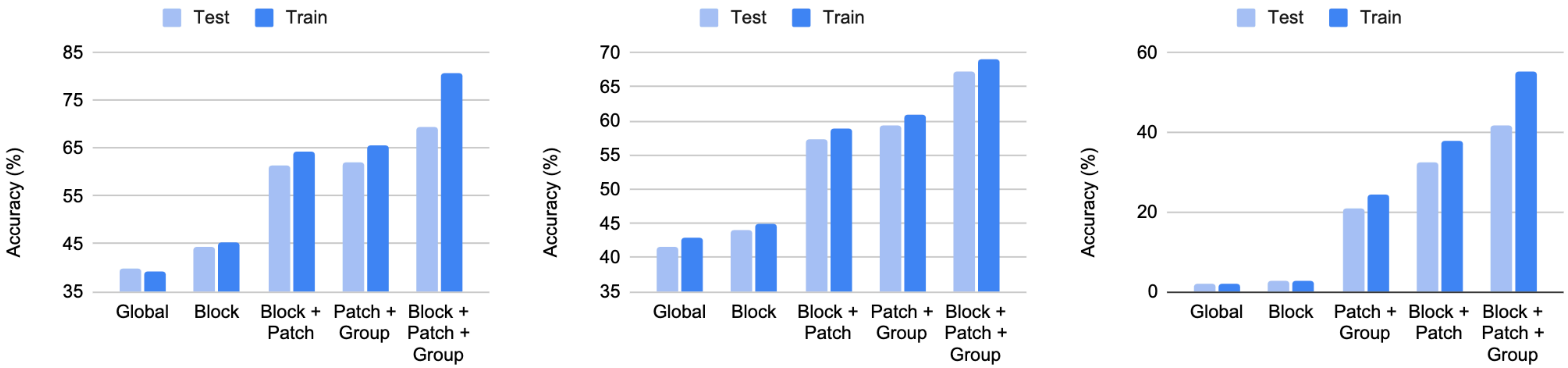
FG = Forward gradients

W = Weight-perturbed (Baydin et al)

A = Activation-perturbed

Ren, Kornblith, Liao, Hinton. Scaling Forward Gradients with Local Losses. 2023

Results
Local losses



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

Ren, Kornblith, Liao, Hinton. Scaling Forward Gradients with Local Losses. 2023

Conclusion

- Forward propagation has some compelling properties
  - Reduced memory relative to backprop, particularly for deeper networks
  - Gradient + function eval in a single forward pass
- Limitations for case with few outputs and many inputs addressed by using random directional derivatives (Baydin et al., 2022)
- Scaled to large models and datasets by Ren et al., 2023