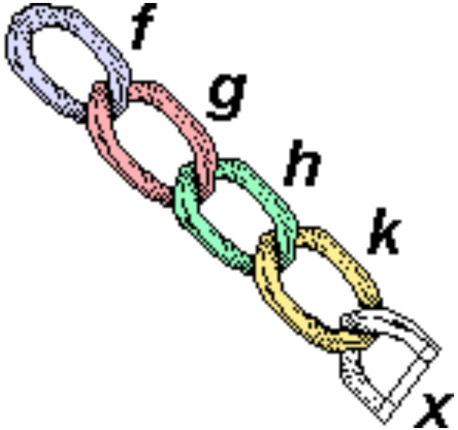
动手学深度学习 v2

李沐 · Amazon资深首席科学家

链式法则和自动求导



Generalize to Vectors

- Chain rule for scalars:

$$y = f(u),\ u = g(x) \qquad \frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$$

- Generalize to vectors straightforwardly

$$\underset{(1,n)}{\frac{\partial y}{\partial \mathbf{x}}} = \underset{(1,)}{\frac{\partial y}{\partial u}}\,\underset{(1,n)}{\frac{\partial u}{\partial \mathbf{x}}} \qquad \underset{(1,n)}{\frac{\partial y}{\partial \mathbf{x}}} = \underset{(1,k)}{\frac{\partial y}{\partial \mathbf{u}}}\,\underset{(k,n)}{\frac{\partial \mathbf{u}}{\partial \mathbf{x}}} \qquad \underset{(m,n)}{\frac{\partial \mathbf{y}}{\partial \mathbf{x}}} = \underset{(m,k)}{\frac{\partial \mathbf{y}}{\partial \mathbf{u}}}\,\underset{(k,n)}{\frac{\partial \mathbf{u}}{\partial \mathbf{x}}}$$

Example 1

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$$

Assume $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n, \quad y \in \mathbb{R}$

$$z = (\langle \mathbf{x}, \mathbf{w} \rangle - y)^2$$

Compute $\dfrac{\partial z}{\partial \mathbf{w}}$

Decompose

$$a = \langle \mathbf{x}, \mathbf{w} \rangle$$

$$b = a - y$$

$$z = b^2$$

$$\begin{aligned}
\frac{\partial z}{\partial \mathbf{w}} &= \frac{\partial z}{\partial b} \frac{\partial b}{\partial a} \frac{\partial a}{\partial \mathbf{w}} \\
&= \frac{\partial b^2}{\partial b} \frac{\partial a - y}{\partial a} \frac{\partial \langle \mathbf{x}, \mathbf{w} \rangle}{\partial \mathbf{w}} \\
&= 2b \cdot 1 \cdot \mathbf{x}^T \\
&= 2\left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right) \mathbf{x}^T
\end{aligned}$$

Example 2

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$$

Assume $\mathbf{X} \in \mathbb{R}^{m \times n}, \quad \mathbf{w} \in \mathbb{R}^{n}, \quad \mathbf{y} \in \mathbb{R}^{m}$

$$z = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

Compute $\dfrac{\partial z}{\partial \mathbf{w}}$

Decompose

$$\mathbf{a} = \mathbf{X}\mathbf{w}$$
$$\mathbf{b} = \mathbf{a} - \mathbf{y}$$
$$z = \|\mathbf{b}\|^2$$

$$\begin{aligned}
\frac{\partial z}{\partial \mathbf{w}} &= \frac{\partial z}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{w}} \\
&= \frac{\partial \|\mathbf{b}\|^2}{\partial \mathbf{b}} \frac{\partial \mathbf{a} - \mathbf{y}}{\partial \mathbf{a}} \frac{\partial \mathbf{X}\mathbf{w}}{\partial \mathbf{w}} \\
&= 2\mathbf{b}^T \times \mathbf{I} \times \mathbf{X} \\
&= 2\left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^T \mathbf{X}
\end{aligned}$$

Auto Differentiation (AD)

- AD evaluates gradients of a function specified by a program at given values
- AD differs to
  - Symbolic differentiation

```
In[1]:= D[4 x^3 + x^2 + 3, x]

Out[1]= 2 x + 12 x^2
```

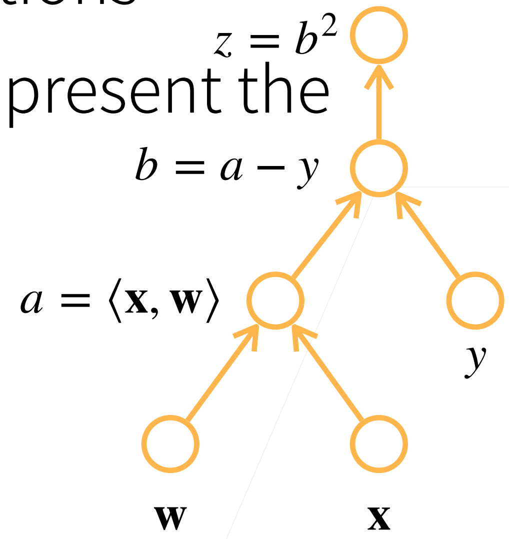
  - Numerical differentiation

$$\frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Computation Graph

Assume $z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$

- Decompose into primitive operations
- Build a directed acyclic graph to present the computation



Computation Graph

- Decompose into primitive operations
- Build a directed acyclic graph to present the computation
- Build explicitly
  - Tensorflow/Theano/MXNet

```
from mxnet import sym

a = sym.var()
b = sym.var()
c = 2 * a + b
# bind data into a and b later
```

Computation Graph

- Decompose into primitive operations
- Build a directed acyclic graph to present the computation
- Build explicitly
  - Tensorflow/Theano/MXNet
- Build implicitly though tracing
  - PyTorch/MXNet

```python
from mxnet import autograd, nd

with autograd.record():
    a = nd.ones((2,1))
    b = nd.ones((2,1))
    c = 2 * a + b
```

Two Modes

- By chain rule $$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u_n}\frac{\partial u_n}{\partial u_{n-1}}\ldots\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial x}$$

- Forward accumulation

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u_n}\left(\frac{\partial u_n}{\partial u_{n-1}}\left(\ldots\left(\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial x}\right)\right)\right)$$

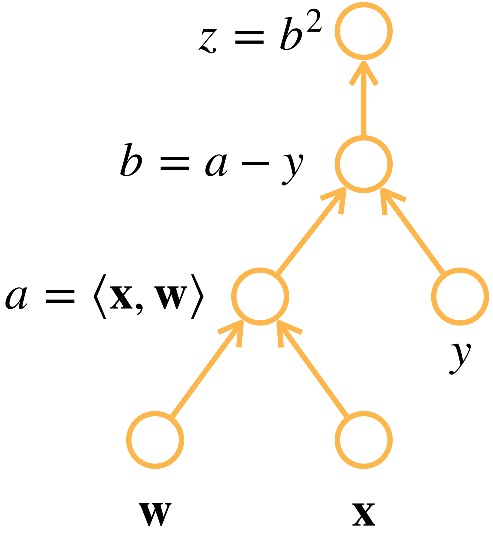
- Reverse accumulation (a.k.a Backpropagation)

$$\frac{\partial y}{\partial x} = \left(\left(\left(\frac{\partial y}{\partial u_n}\frac{\partial u_n}{\partial u_{n-1}}\right)\ldots\right)\frac{\partial u_2}{\partial u_1}\right)\frac{\partial u_1}{\partial x}$$

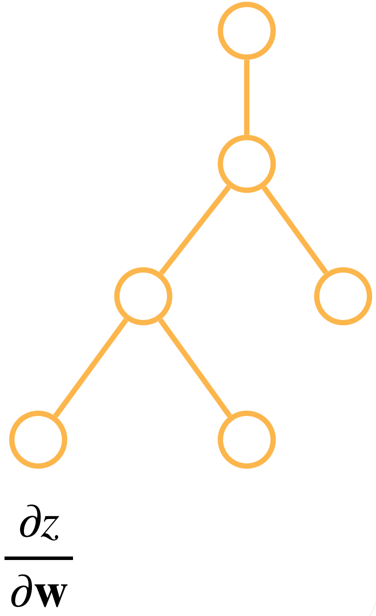
Reverse Accumulation

Assume $z = (\langle \mathbf{x}, \mathbf{w} \rangle - y)^2$

Forward



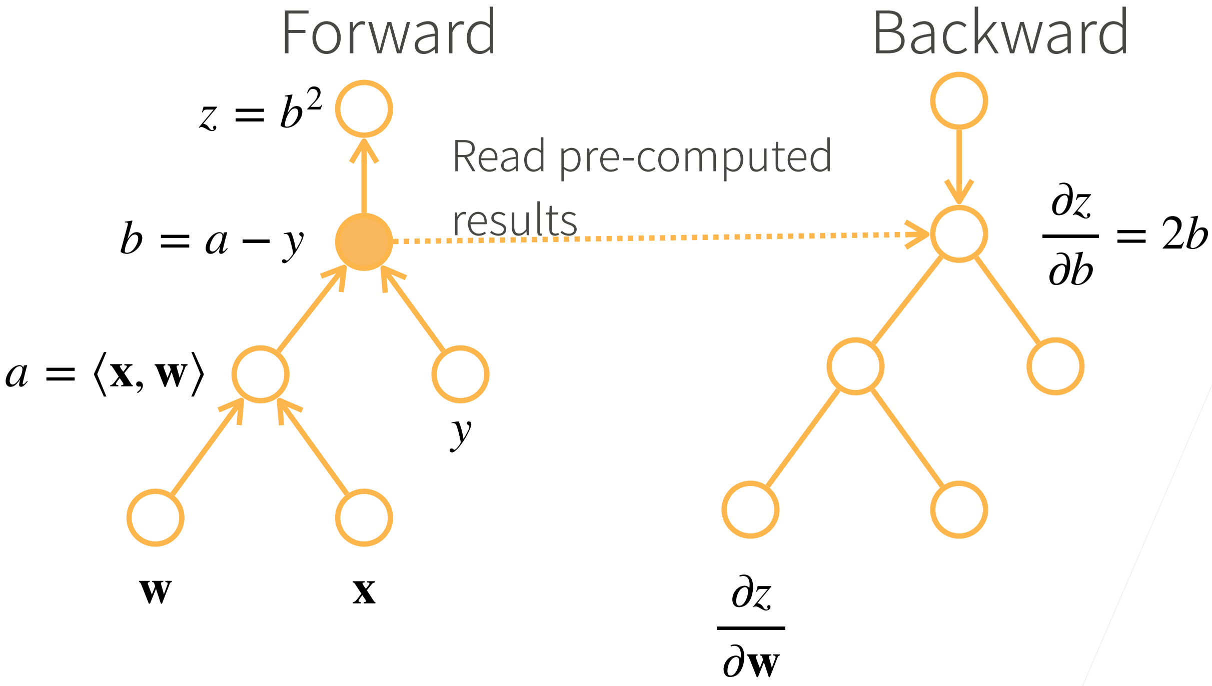
Backward



https://courses.d2l.ai/zh-v2

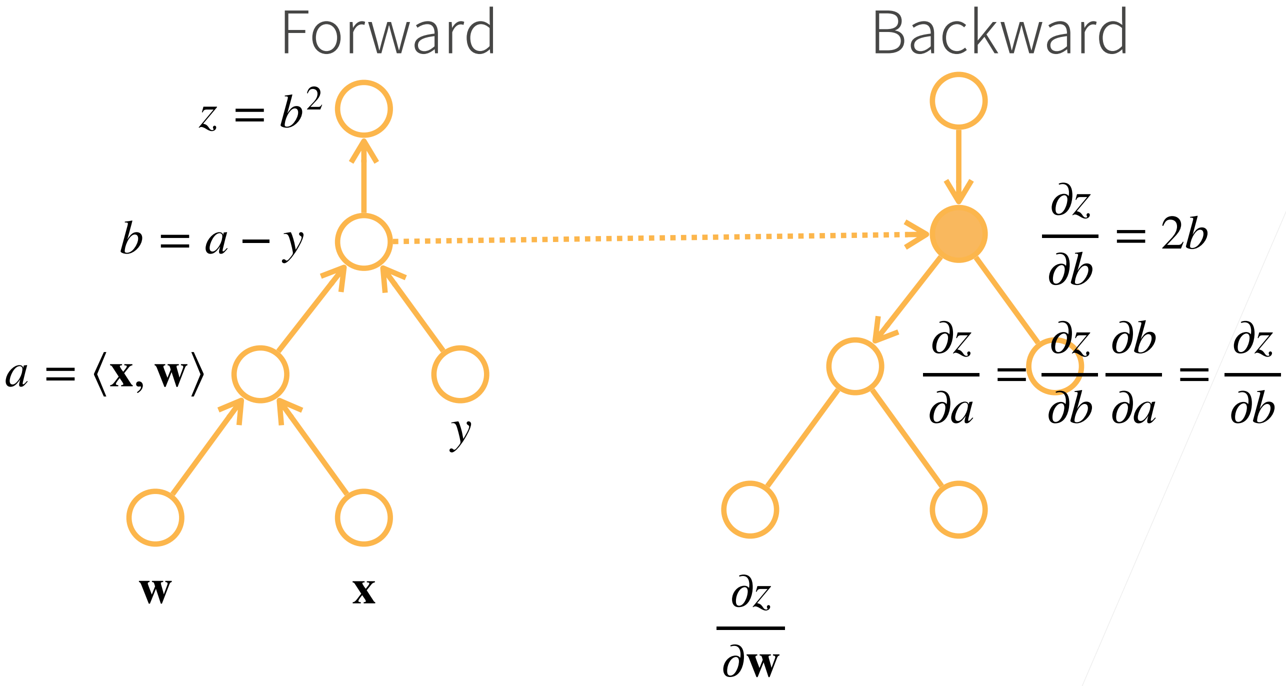
Reverse Accumulation

Assume $z = (\langle \mathbf{x}, \mathbf{w} \rangle - y)^2$



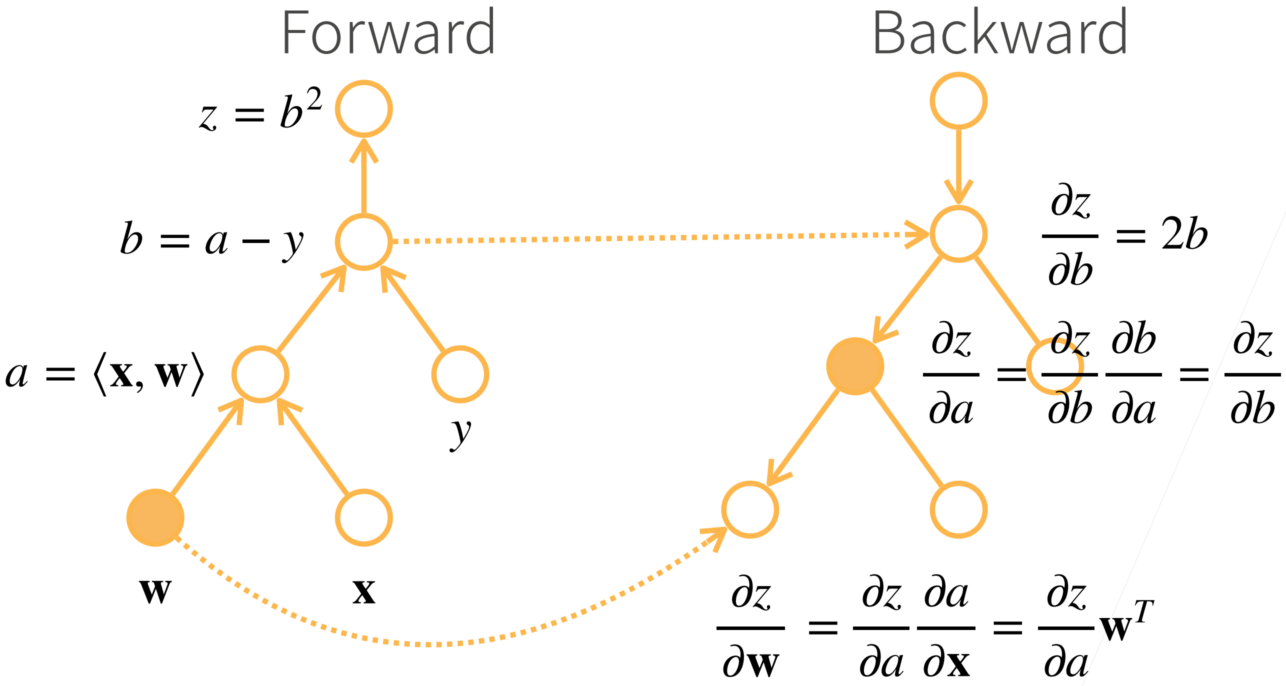
Reverse Accumulation

Assume $z = (\langle \mathbf{x}, \mathbf{w} \rangle - y)^2$



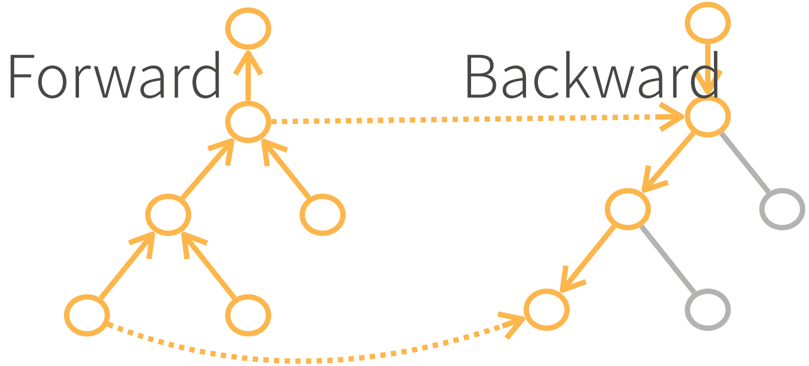
Reverse Accumulation

Assume $z = (\langle \mathbf{x}, \mathbf{w} \rangle - y)^2$



Reverse Accumulation Summary

- Build a computation graph
- Forward: Evaluate the graph, store intermediate results
- Backward: Evaluate the graph in a reversed order
  - Eliminate paths not needed



https://courses.d2l.ai/zh-v2

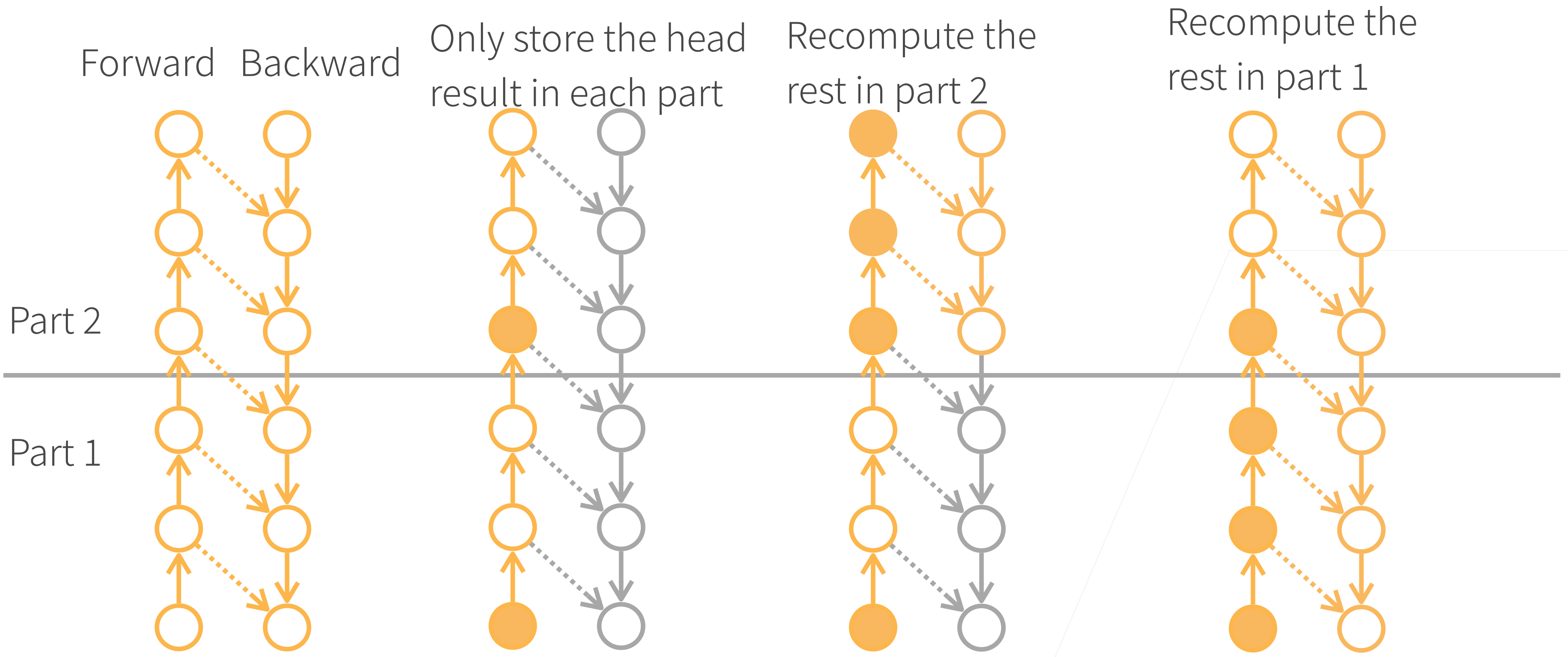
Complexities

- Computational complexity: O(n), n is #operations, to compute all derivatives
  - Often similar to the forward cost
- Memory complexity: O(n), needs to record all intermediate results in the forward pass
- Compare to forward accumulation:
  - O(n) time complexity to compute one gradient, O(n*k) to compute gradients for k variables
  - O(1) memory complexity

[Advanced] Rematerialization

- Memory is bottleneck for backward accumulation
  - Linear to #layers and batch size
  - Limited GPU memory (32GB max)
- Trade computation for memory
  - Save a part of intermediate results
  - Recompute the rest when needed

Rematerialization



Complexities

- An additional forward pass
- Assume m parts, then O(m) for head results, O(n/m) to store one part’s results
  - Choose $m = \sqrt{n}$ then the memory complexity is $O\left(\sqrt{n}\right)$
- Applying to deep neural networks
  - Only throw aways simple layers, e.g. activation, often <30% additional overhead
  - Train 10x larger networks, or 10x large batch size