# Optimizer Fusion: Efficient Training with Better Locality and Parallelism

Zixuan Jiang, Jiaqi Gu, Mingjie Liu, Keren Zhu, David Z. Pan

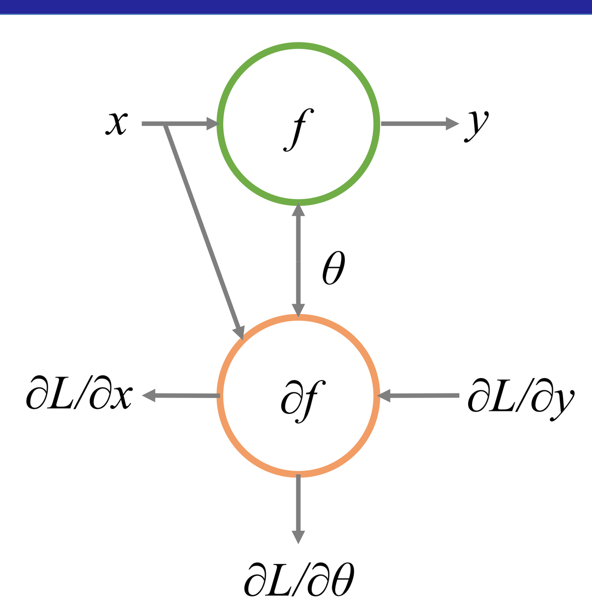
Electrical and Computer Engineering, The University of Texas at Austin

## Machine learning frameworks

- Machine learning algorithms and frameworks **coevolve**.
- Static (symbolic) and dynamic (eager, imperative) computation flow.
- Two critical components.

| Automatic differentiation | Iterative optimization methods |
|---|---|

## Automatic differentiation




- Compute gradients based on chain rule.

## Iterative optimization methods

- Optimization algorithms in general form.

**Input:** objective function $f$
Initialize the starting point $\theta^{(0)}$
**for** $t = 1, 2, \dots$ **do**
  **if** stopping criterion is met **then**
    **return** $\theta^{(t-1)}$
  **end if**
  $\Delta\theta = \pi(f, \theta^{(0)}, \theta^{(1)}, \dots, \theta^{(t-1)})$
  $\theta^{(t)} = \theta^{(t-1)} + \Delta\theta$
**end for**

- Gradient descent

$$\pi_1 = -\eta \nabla f(\theta^{(t-1)})$$

- Gradient descent with momentum

$$\pi_2 = -\eta \sum_{\tau=0}^{t-1} \alpha^{t-\tau-1} \nabla f(\theta^{(\tau)})$$

- Newton's method

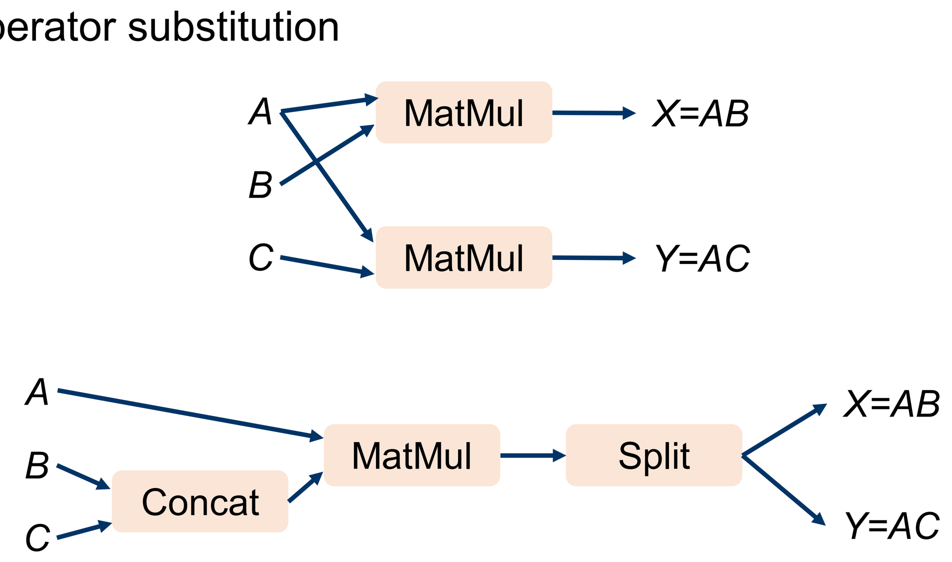
$$\pi_3 = -\eta \nabla^2 f(\theta^{(t-1)})^{-1} f(\theta^{(t-1)})$$

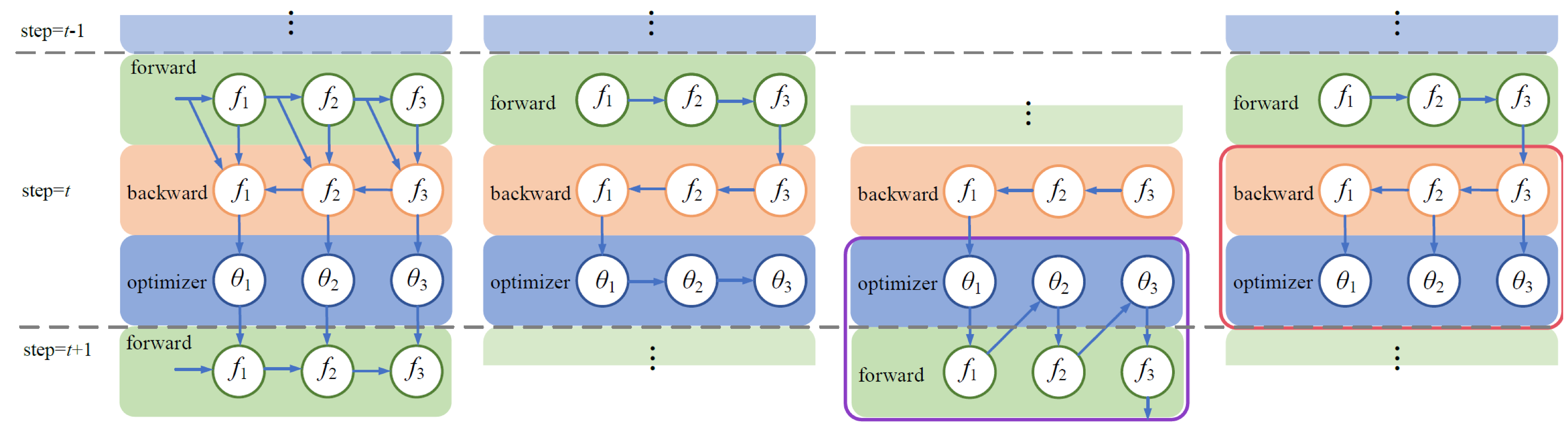
## Graph optimization

- Operator fusion
  Fuse convolution, batch normalization, and ReLU.

Conv → BN → ReLU    Conv-BN-ReLU

- Operator substitution




(a) Data dependency graph
(b) Baseline method
(c) Forward-fusion
(d) Backward-fusion

## Our methods

Baseline

Forward → Backward → Optimizer step

Forward-fusion

Optimizer step → Forward

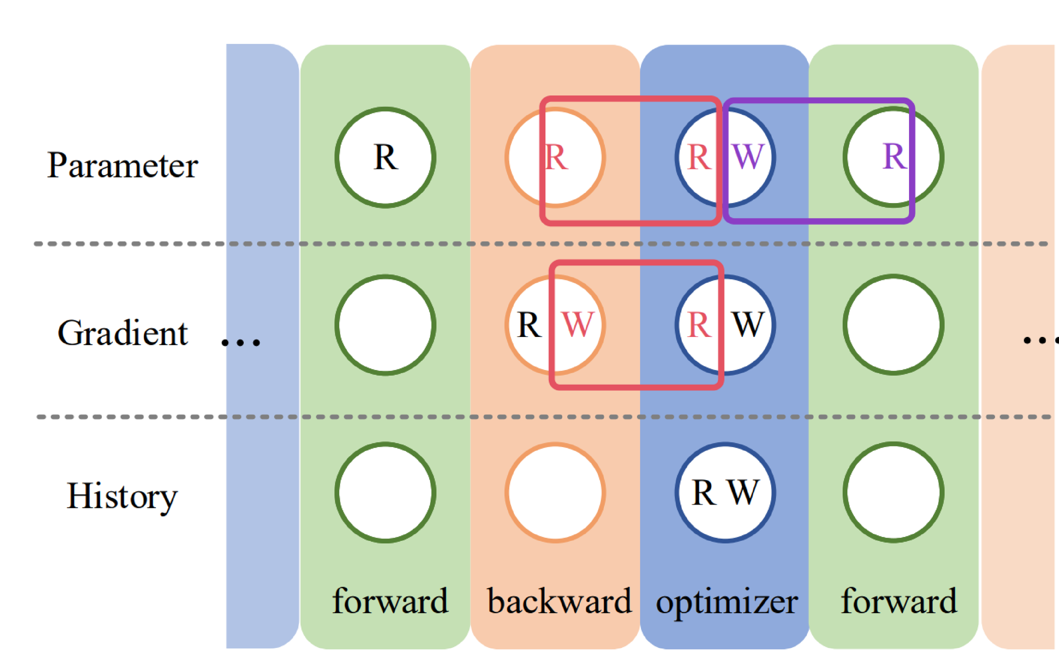
- Fuse the parameter update with the next forward pass.
- Update parameters as **late** as possible.

Backward-fusion

Backward → Optimizer step

- Fuse the parameter update with the current backward pass.
- Update parameters as **early** as possible.

## Locality and parallelism

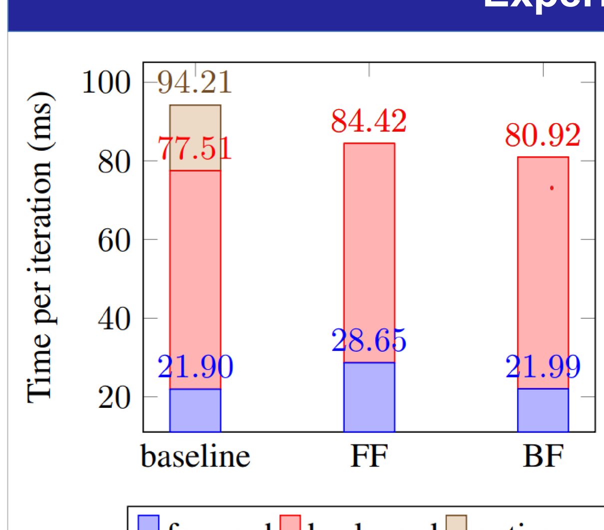



- Memory transactions and data locality in the training process. *R* and *W* represent memory read and write, respectively. History means parameter history needed in the optimizer, e.g., momentum.

| Method | Locality | Parallelism | Global information |
|---|---|---|---|
| Baseline | × | × | √ |
| Forward-fusion | √ | × | √ |
| Backward-fusion | √ | √ | × |

- In some iterative optimization methods, the gradients will be post-processed after the all the gradients are available. An example is clipping gradients by its norm.
- The *forward-fusion* method is applicable when the global information is needed.

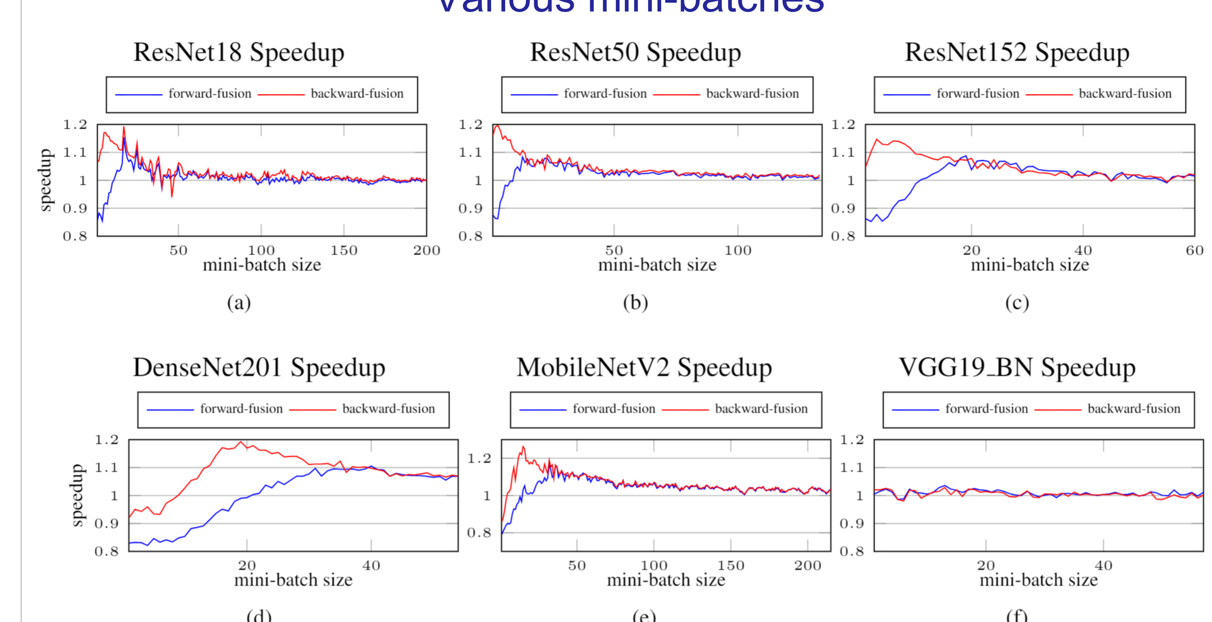
## Experiments




- Training time breakdown of MobileNetV2 with mini-batch size 32. FF, BF are short for *forward-fusion*, *backward-fusion*.
- Our *forward-fusion* and *backward-fusion* improve the training throughput by 12% and 16%.

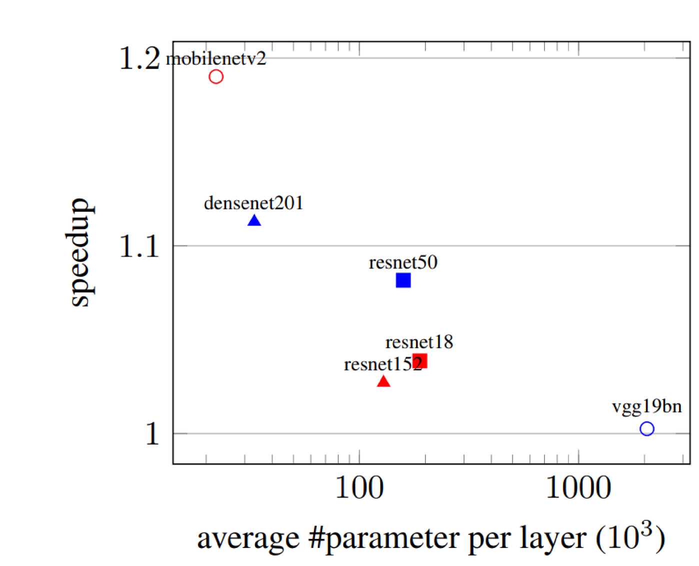
## Experiments (continued)

Various mini-batches



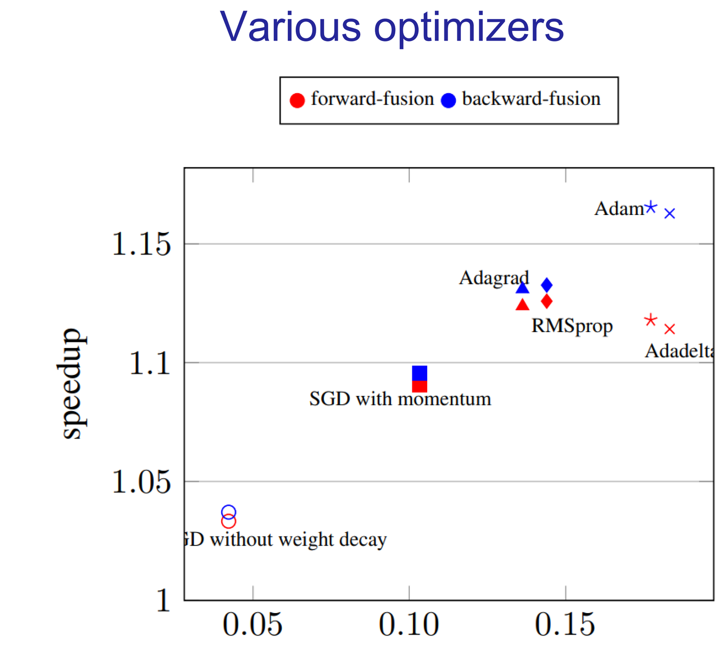

- The absolute training time saved by our methods is independent of the mini-batch size.
- The relative speedup will decrease as the mini-batch size grows.

Various models




- The smaller the average number of parameters per layer, the more locality we can leverage so that our methods can achieve higher training speed.

Various optimizers




- The horizontal axis represents the ratio of the optimizer time to a whole iteration time.
- The more runtime-costly the optimizer, the higher speedup we can achieve.

## Conclusion

- Conventional eager execution in machine learning frameworks separate the updating of trainable parameters from forward and backward computations.
- We propose two methods forward-fusion and backward-fusion to better leverage the locality and parallelism.
- Experimental results demonstrate the effectiveness and efficiency of our methods across various configurations.
- Our proposed methods are orthogonal to other optimization methods and do not affect the training results. We keep all the features of the eager execution.