上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Week 6 Extension: Gradient Checkpointing

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

AI tools assisted in generating some figures in these slides. All such content has been reviewed, and the instructor is responsible for its accuracy.

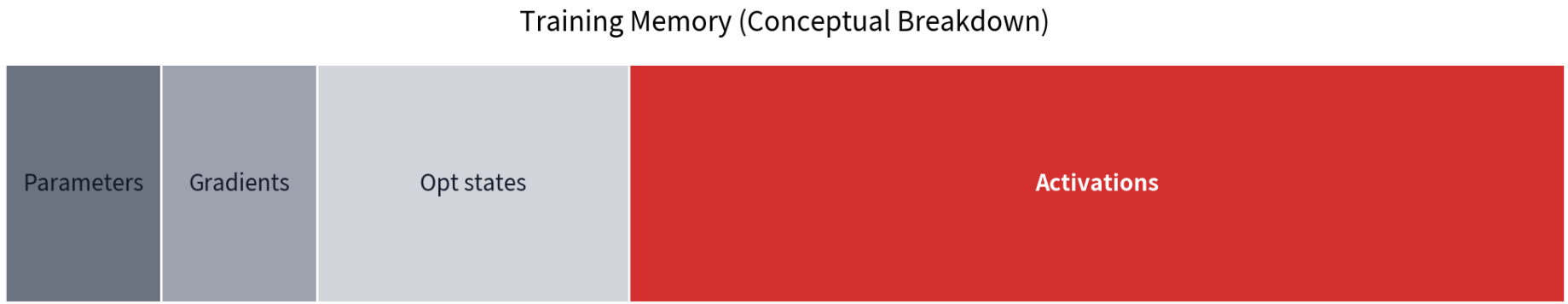
Why Does Training Run Out of Memory?

Training memory includes:

- parameters, gradients, optimizer states, activations

For deep nets, activations are often the bottleneck.

Question: Can we reduce memory without changing the gradient?



Key Idea: Store Less, Recompute More

Backprop needs forward activations.

Naive training:

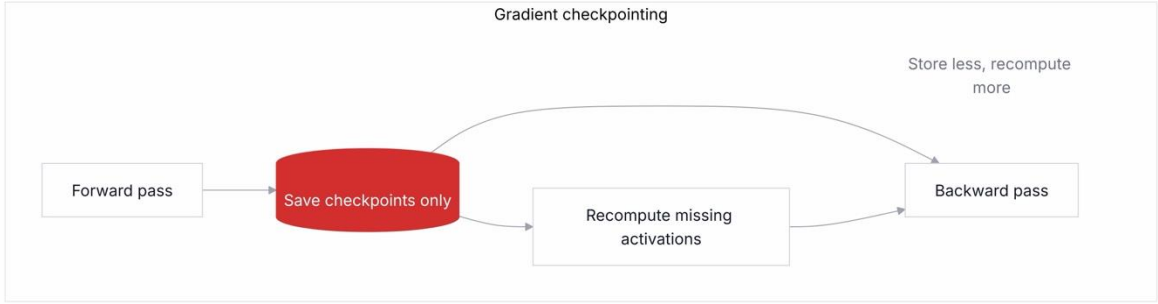
- save almost everything

Gradient checkpointing:

- save only selected activations
- recompute missing ones during backward

Tradeoff:

- memory ↓, compute ↑



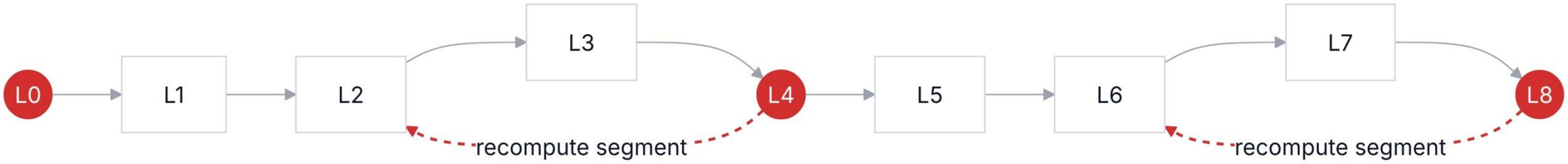
Example: Split the Network into Segments

Without checkpointing: save every layer activation

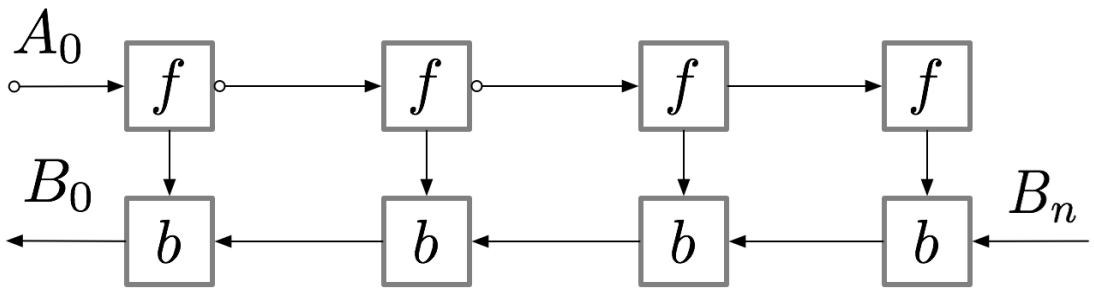
With checkpointing:

- save only segment boundaries
- rerun forward inside each segment during backward

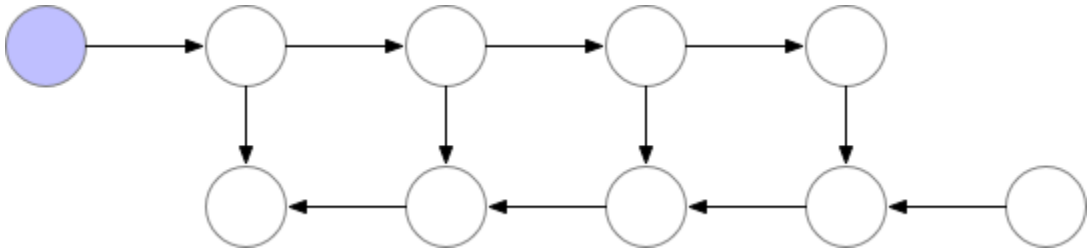
Same model, same loss, same gradient.
Different memory/computation tradeoff.



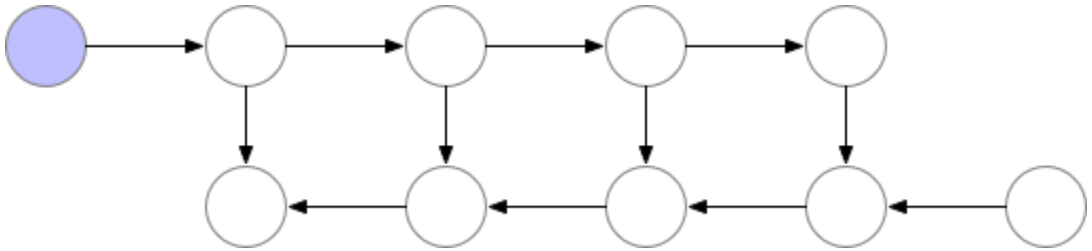
Gradient Checkpointing



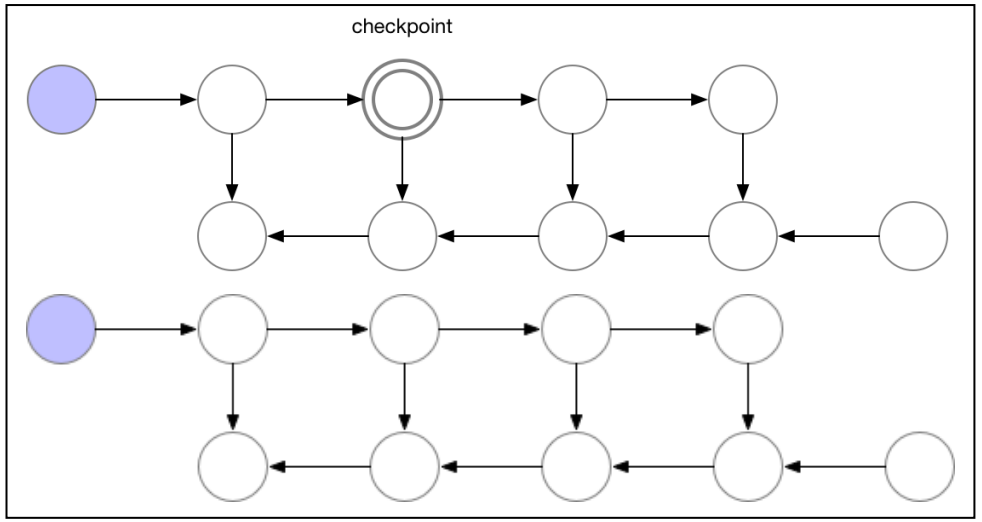
Forward & backward graph.



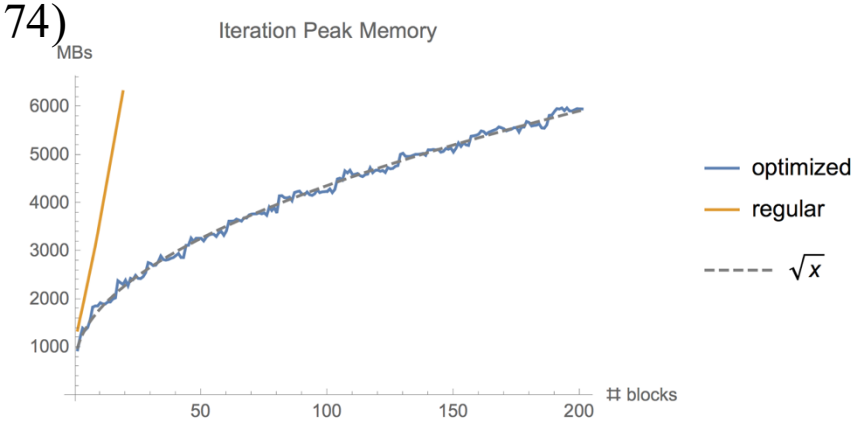
(1) Traditional gradient computation: save all the input tensors.



(2) Memory-poor gradient computation: re-compute the input tensors when needed.



(1) Gradient checkpointing: trade-off between computation and memory cost. (Chen et al. Training Deep Nets with Sublinear Memory Cost. arXiv 1604.06174)



Memory cost on ResNet.

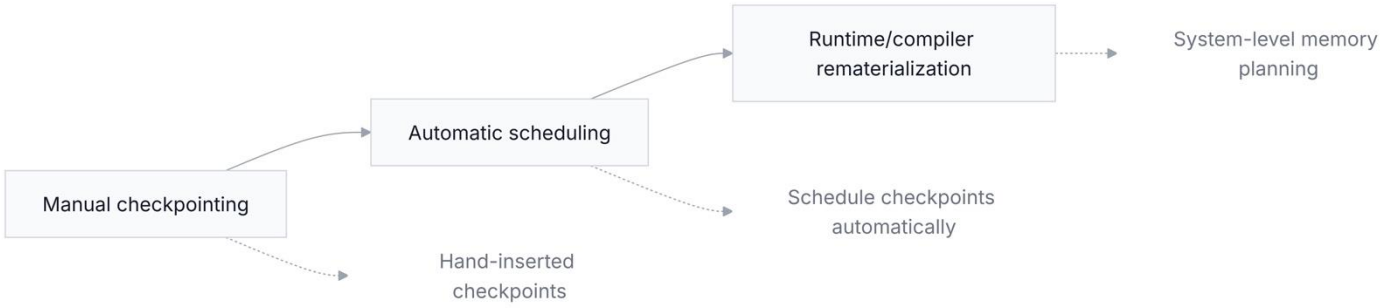
Beyond Checkpointing: Rematerialization

Modern view:

- checkpointing is one form of rematerialization
- do not store everything, recompute intermediates on demand

Representative directions:

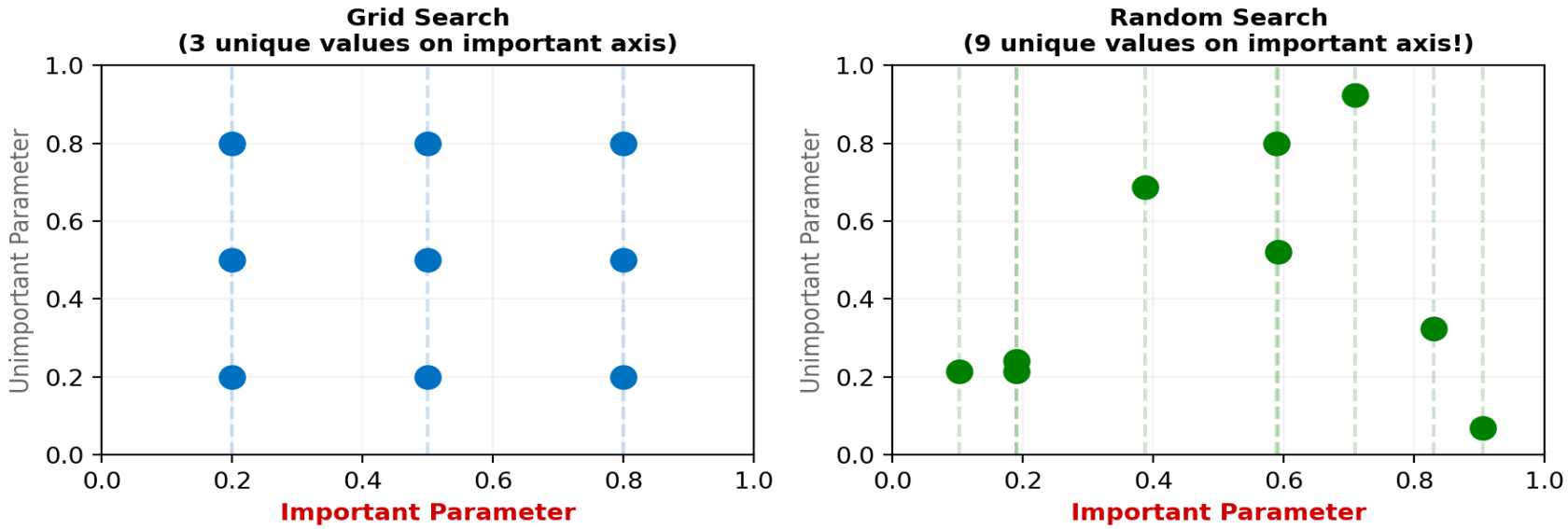
- manual checkpoint placement
- automatic scheduling
- compiler/runtime memory planning



Revisit: Why Random Search Works

Bergstra & Bengio. Random Search for Hyper-Parameter Optimization. JMLR 2012

- **Low effective dimensionality: only a few hyperparameters matter**
  - Grid: N^(1/d) unique values per dim
  - Random: N unique values per dim!
- **Same budget, much better coverage**



Representative Papers and Systems

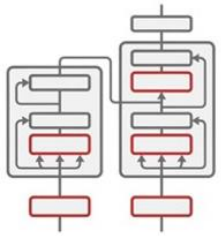
Suggested milestones:

1. Chen et al., 2016. Training Deep Nets with Sublinear Memory Cost — classic checkpointing
2. Gruslys et al., 2016. Memory-Efficient Backpropagation Through Time — sequence / RNN setting
3. Jain et al., 2020. Checkmate: Breaking the Memory Wall with Optimal Tensor Rematerialization — automatic checkpoint scheduling
4. Kirisame et al., 2021. Dynamic Tensor Rematerialization — modern rematerialization
5. PyTorch / JAX / XLA practice — now built into mainstream training stacks

Why Modern AI Uses It

Especially useful for:

- large Transformers
- long-context models
- diffusion models
- video models
- high-resolution training



Reason:

- activation memory scales badly with: depth, sequence length, resolution, hidden size

Message: large-model training is often memory-bound before compute-bound.