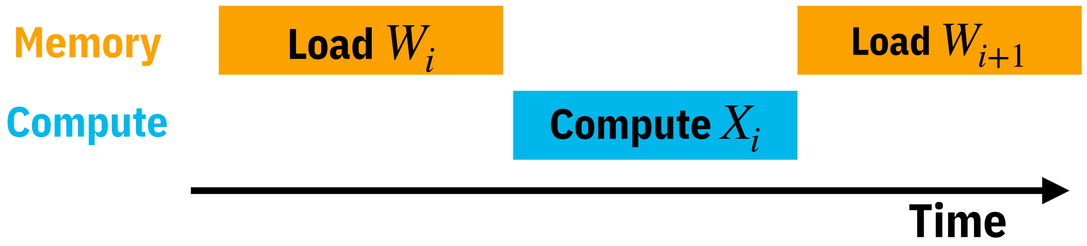
EECE695D: Efficient ML Systems

Compute / Memory: Deep Neural Networks (continued)

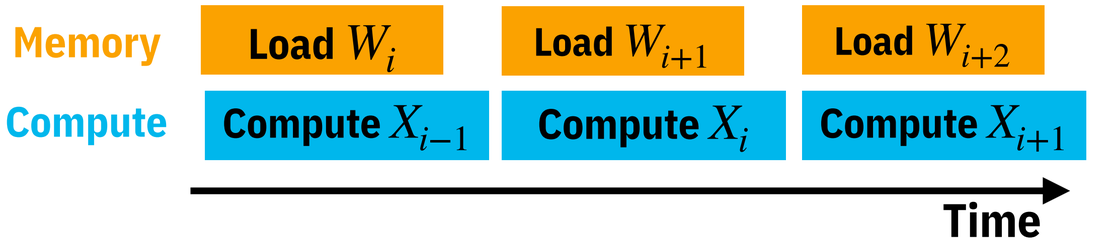
- NeurIPS 2022 results are out—Wish you all got lucky enough!
- Office hour today! 5PM~6PM @ My office
- Assignment#1 will be out very soon! Stay tuned.

Recap

- DNN inference
  - Overlapping compute with data transfer / Double buffering
  - Convolutional layers and compute-bound cases



- DNN training
  - Backpropagation — Re-using previous computations for compute-efficiency
  - Reverse mode autodiff vs. Forward mode autodiff



Side Note: Gradient Checkpointing

- In a sense, backpropagation is a way to use extra memory for less compute
  (for storing activations computed during forward)

Q. Can we use less extra memory,
at the cost of slightly increased compute?

A. Yes—no need to store all activations.

Instead, compute them again with extra forward operations!

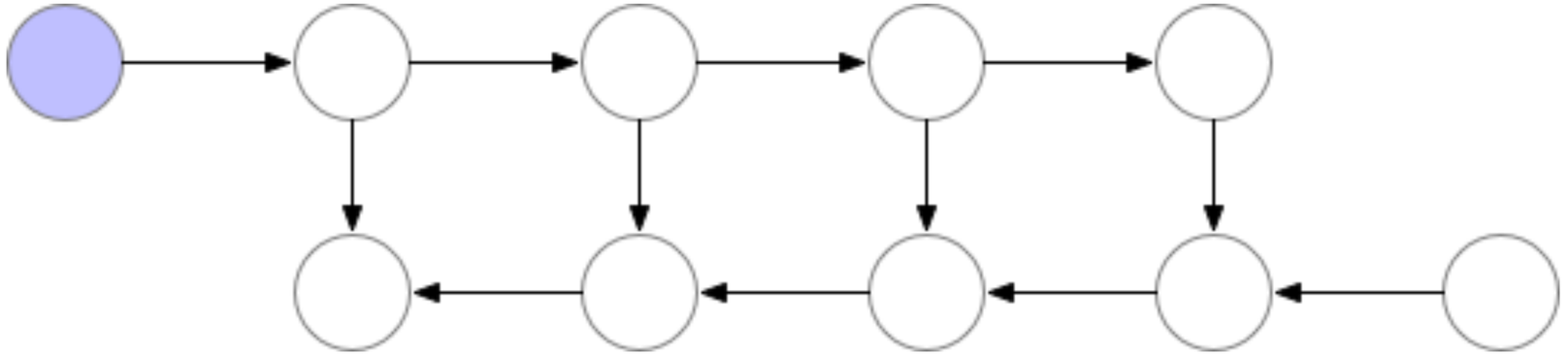
Training Deep Nets with Sublinear Memory Cost

Tianqi Chen [1], **Bing Xu** [2], **Chiyuan Zhang** [3], and **Carlos Guestrin** [1]

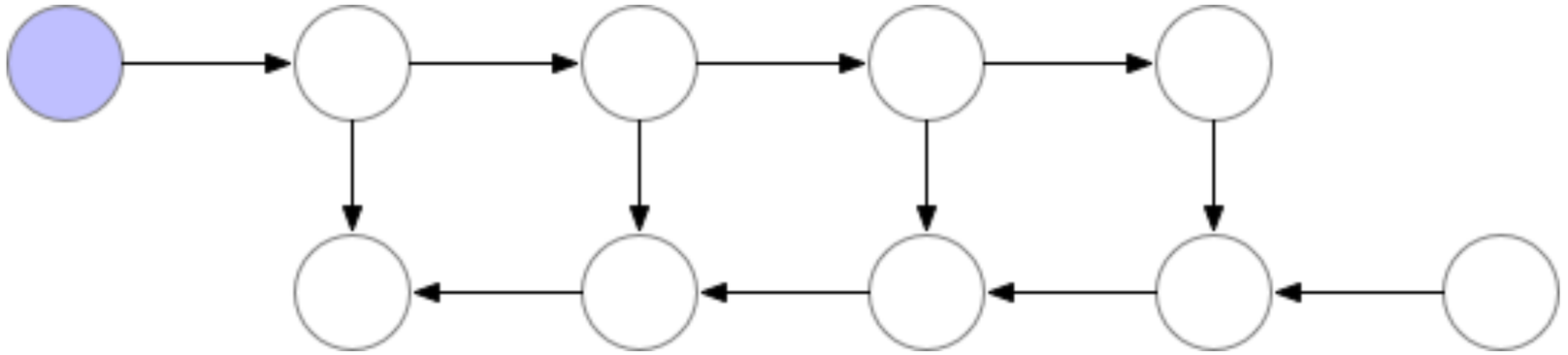
[1] Unveristy of Washington [2] Dato. Inc [3] Massachusetts Institute of Technology

https://arxiv.org/pdf/1604.06174v2.pdf

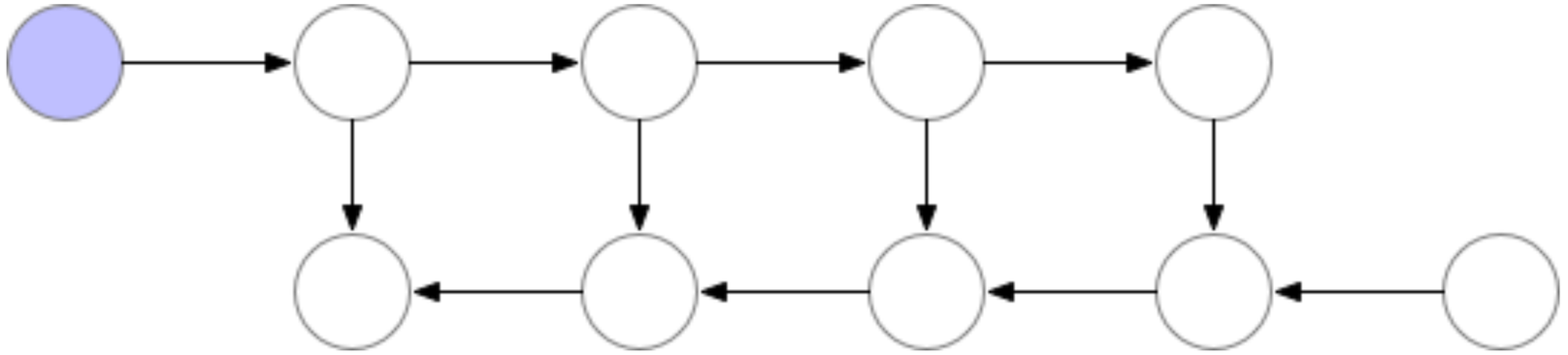
Vanilla



Low-Memory (w/o checkpoint)



Checkpointing



https://github.com/cybertronai/gradient-checkpointing

Side Note: Optimization algorithm!

- Some optimizers have extra “state variables” for various reasons.
  - For instance, we sometimes utilize “momentum” terms.
    Beware of the memory overhead!!

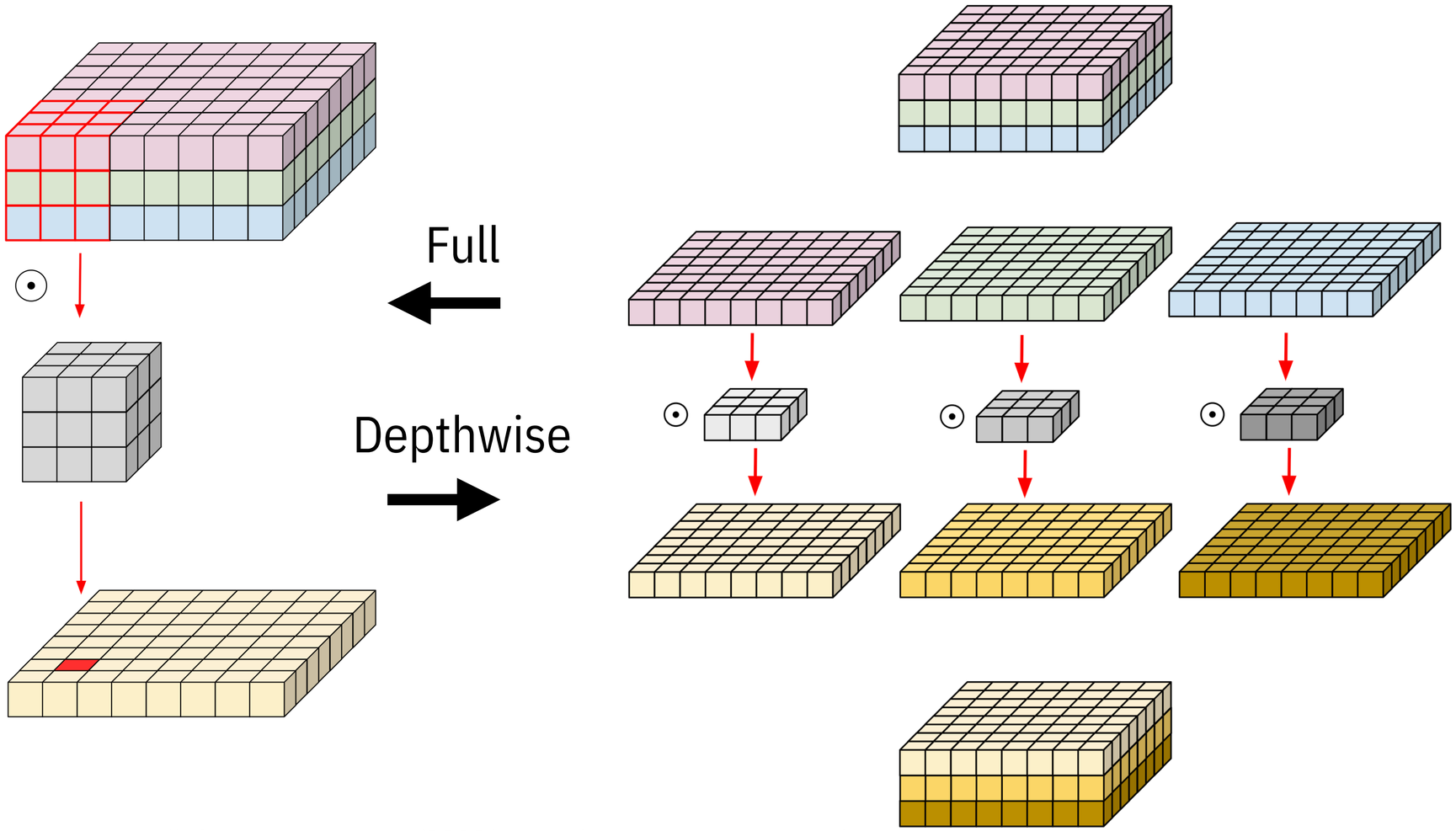
Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Kingma and Ba, “Adam: A method for stochastic optimization,” ICLR 2015.

Side Note: Depthwise Convolution

- In most modern “efficient” neural network architectures (e.g., MobileNet, EfficientNet), people use **depthwise convolutions** instead of full convolutions.



Params

Full. c^2k^2

DW. ck^2

FLOPs

Full. $2c^2k^2d^2$

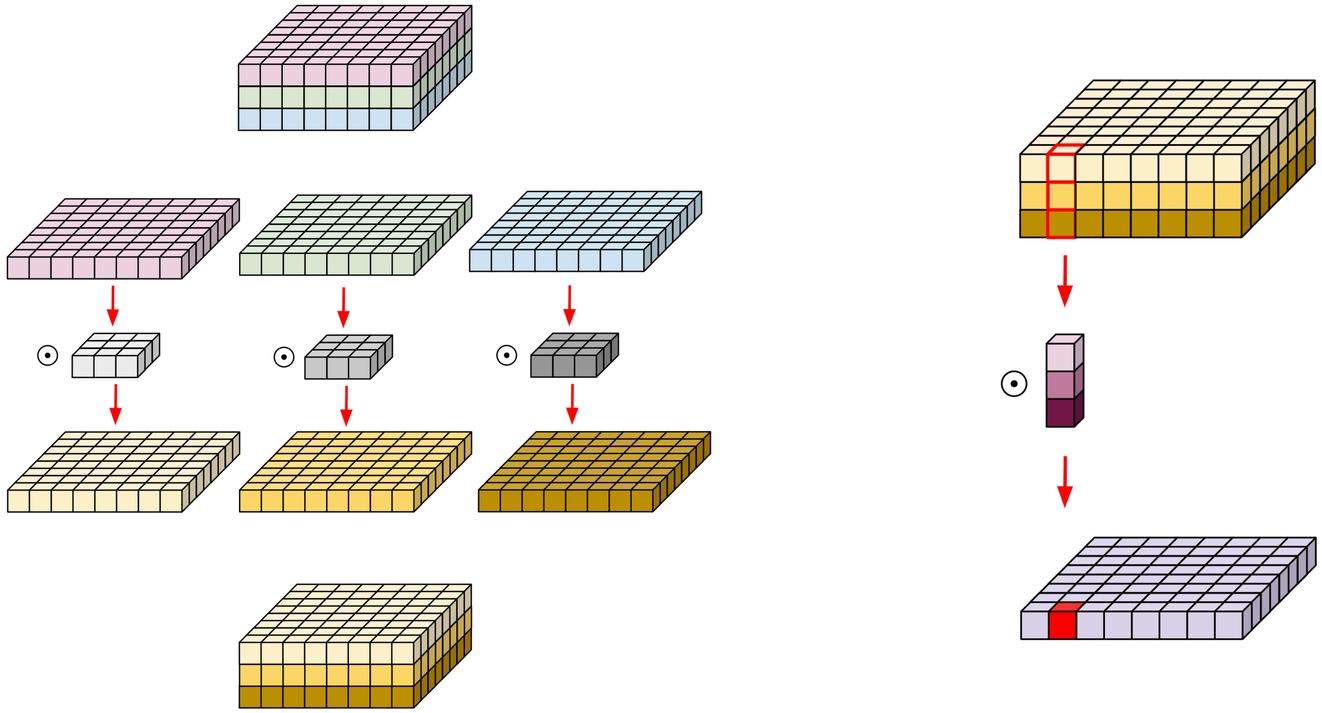
DW. $2ck^2d^2$

Image source: https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/

Side Note: Depthwise Convolution

- **Problem.** Channel information cannot be mixed!

  ⇒ Use one-by-one convolution to mix the layers # e.g., Inverted Bottleneck



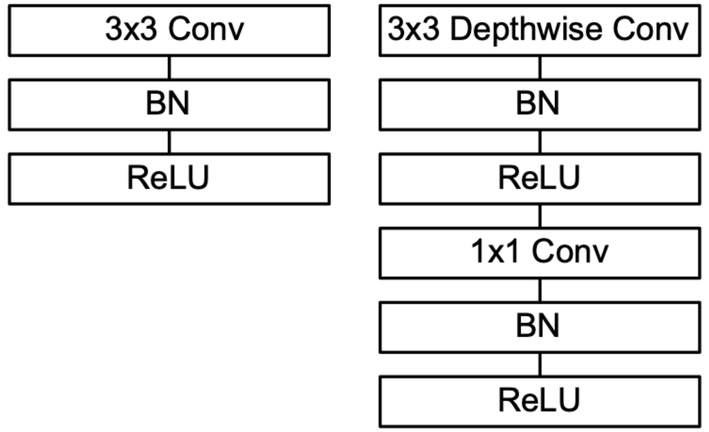


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

Image source: https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/
Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv 2017

A bit more about “efficiency metrics”

Take a pause...

- We have been talking about many metrics to quantify DNN efficiency:
    - #Parameters, FLOPs, data transfer (memory bandwidth), Peak memory, ...

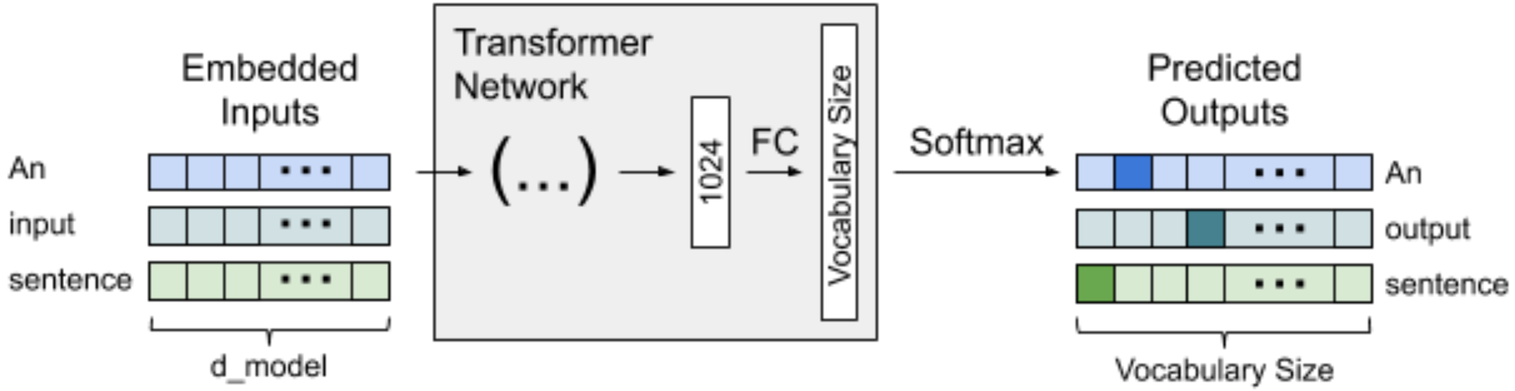
- In academic papers—many focusing only on inference phase—people usually report 1~2 out of three metrics:
    - # Parameters
    - Inference FLOPs
    - Speed

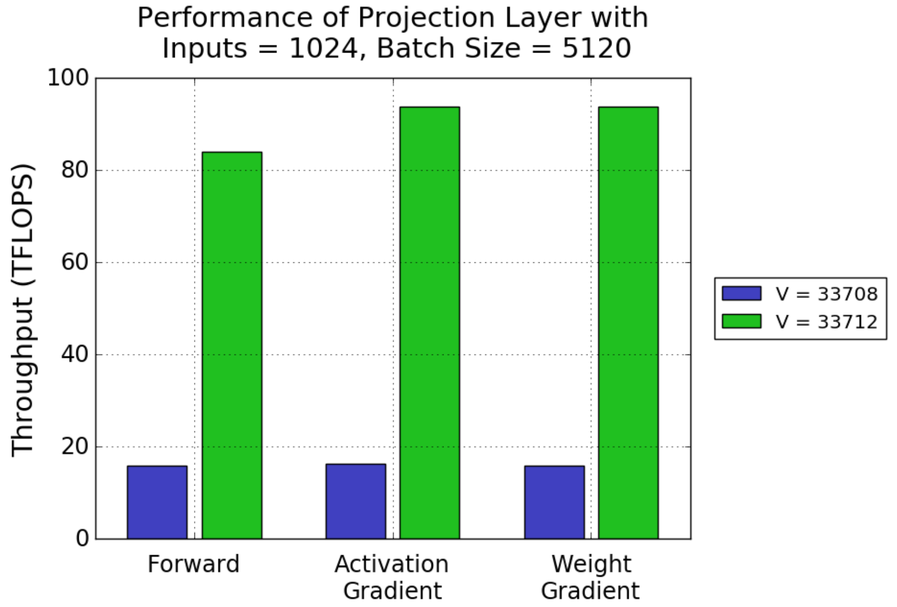
Let us take some time to look at these quantities carefully...

#Params

The number of “trainable” parameters in the neural network.

- Indicator of model storage cost (e.g., on SSD)
- Indicator of the number of memory accesses ……. but grouping matters
- Indicator of computational cost …… but parameter reuse, grouping matters
- Indicator of peak memory usage …… but activations also matter!





https://developer.nvidia.com/blog/optimizing-gpu-performance-tensor-cores/

#FLOPs

The number of floating point operations (mutliplication / addition)

- Indicator of computational cost
- Indicator of inference time ..... but ignores parallelism, hardware, overhead
- Indicator of energy (for compute) ..... but ignores overhead, grouping

Table 1. Proportions for operator classes in PyTorch.

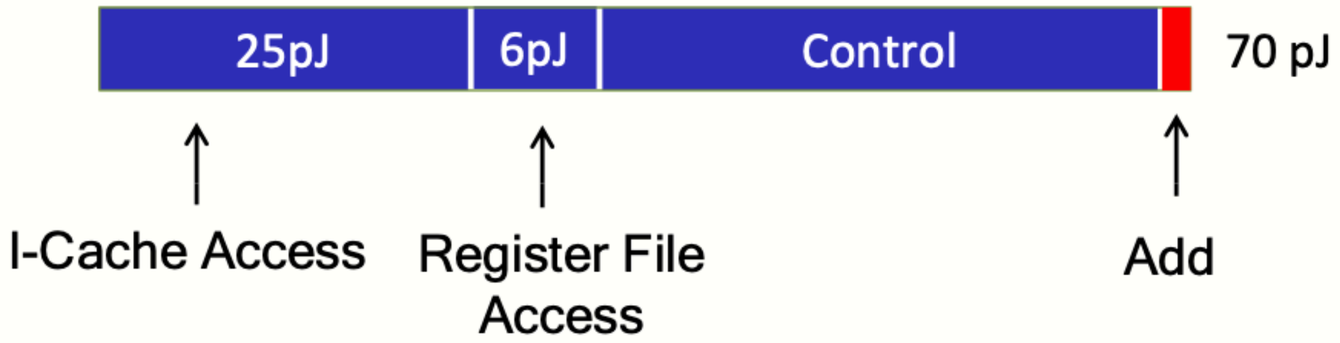
Operator class	% flop	% Runtime
△ Tensor contraction	99.80	61.0
□ Stat. normalization	0.17	25.5
○ Element-wise	0.03	13.5

Mark Horowitz, "Computing's Energy Problem (and what we can do about it)" ISSCC 2014

- PyTorch itself takes up a lot of time—when we use PyTorch to perform a single addition.

Image source: https://horace.io/brrr_intro.html

Instruction Energy Breakdown



Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Operation	Energy**	Overhead*
HFMA	1.5pJ	2000%
HDP4A	6.0pJ	500%
HMMA	110pJ	22%
IMMA	160pJ	16%

HFMA - Half-Precision Fused Multiply-Add
HDP4A - 4-way dot-product
HMMA - 16 x 16 Matrix Multiplication (FP16)
IMMA - 32 x 32 Integer MatMul (Int8)

Mark Horowitz, "Computing's Energy Problem (and what we can do about it)" ISSCC 2014
Bill Dally, "Hardware for Deep Learning," SysML 2018

Speed

There are many definitions...

- Throughput. Number of examples that are processed within a specific period of time
  (e.g., examples/sec)
- Latency. Inference time of a given example to pass through
  (a bit decoupled from throughput; batching!)
- Wall-clock time. Time for processing a fixed set of examples.

- Also influenced also by pipeline bubble (e.g., no compute can be done at the very beginning!), halting from compute/memory bottlenecks...
- Memory access cost is usually the key.

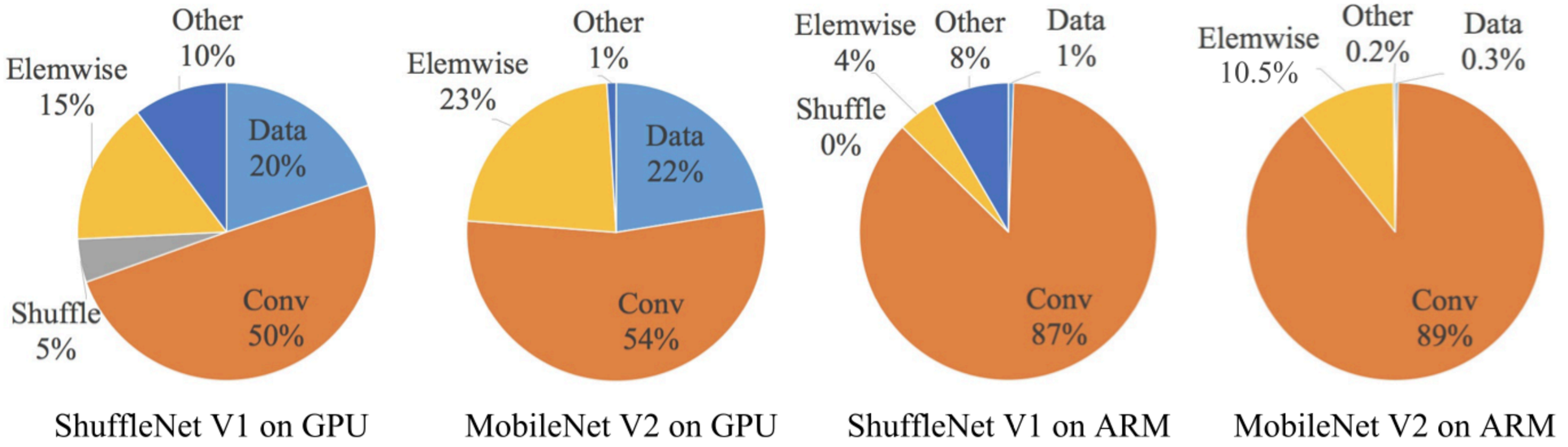


Fig. 2: Run time decomposition on two representative state-of-the-art network architectures, *ShuffeNet v1* [15] ($1\times$, $g = 3$) and *MobileNet v2* [14] ($1\times$).

GPU: NVIDIA GeForce 1080Ti + cuDNN 7.0
ARM: Qualcomm Snapdragon 810

Ma et al., "ShuffleNet v2: Practical Guidelines for efficient CNN architecture design," 2018

A disturbing fact

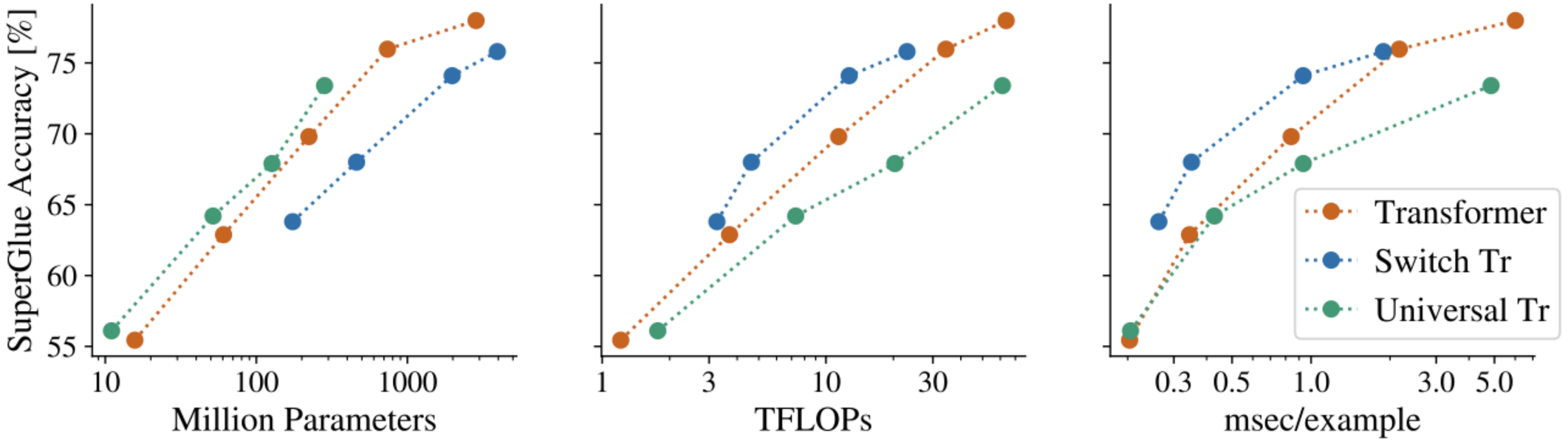


Figure 1: Comparison of standard Transformers, Universal Transformers and Switch Transformers in terms of three common cost metrics: number of parameters, FLOPs, and throughput. Relative ranking between models is reversed between the two cost indicators. Experiments and the computation of cost metrics were done with Mesh Tensorflow (Shazeer et al., 2018), using 64 TPU-V3.

Dehghani et al., "The efficiency misnomer," ICLR 2022

We are doomed... but is that all?

- These all pessimistic results, regarding the design of a “unified framework” for efficient ML.
  - A wide range of knowledge—encompassing ML algorithms, hardware platform, compilers, kernels, network architecture—desperately needed.
- Think of this as a chance—improvements in any aspect is a contribution!
- In the remainder of this course, we take a “case study” type of an approach: look at small but well-defined problems and approaches, to understand their promises and limitations
  - Eventually, you may be able to find your own research question ;)

- **Next up.** But first, a bit of a theory behind overparameterization.