

**Spring 2023**

# ADVANCED TOPICS IN COMPUTER VISION

---

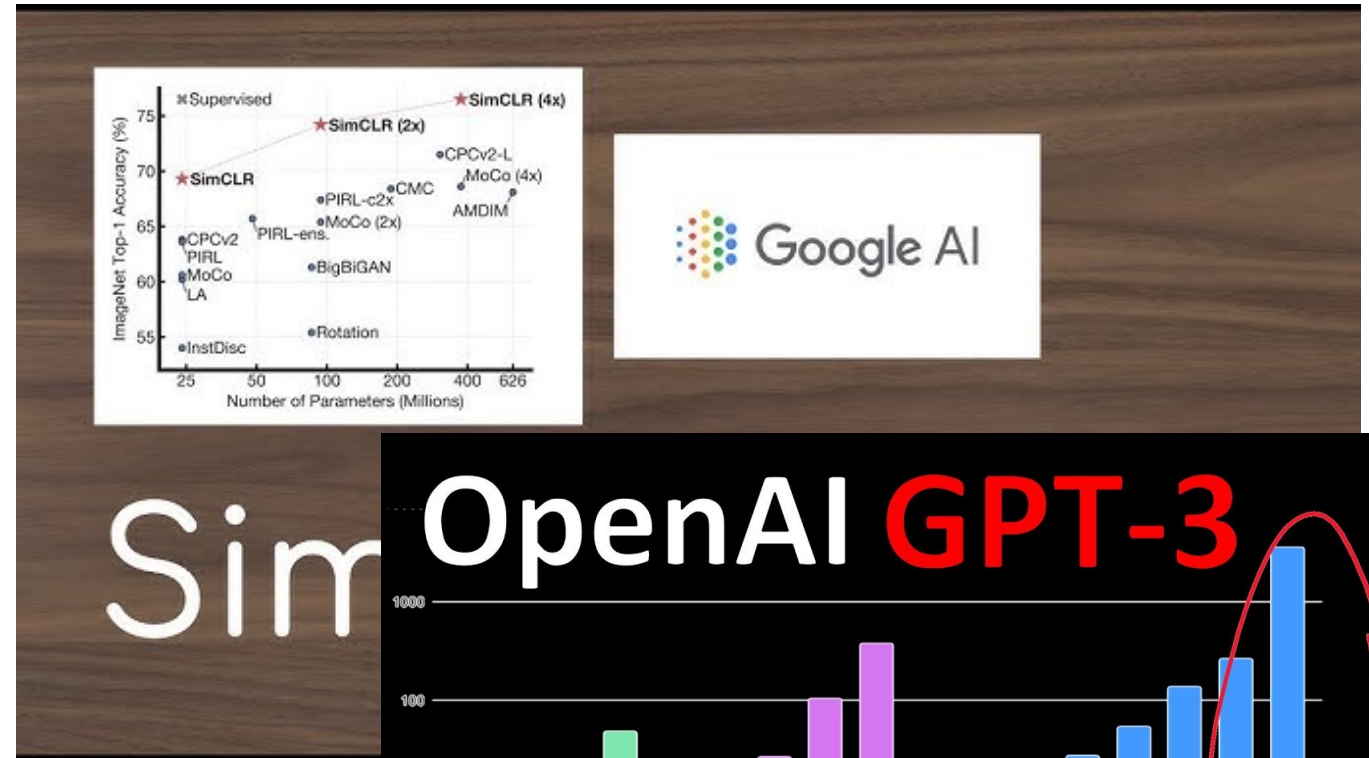
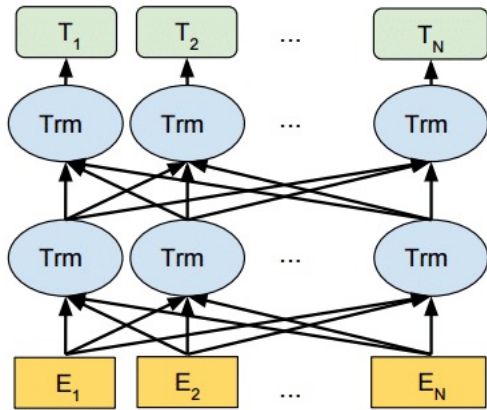
**Atlas Wang**

Assistant Professor, The University of Texas at Austin

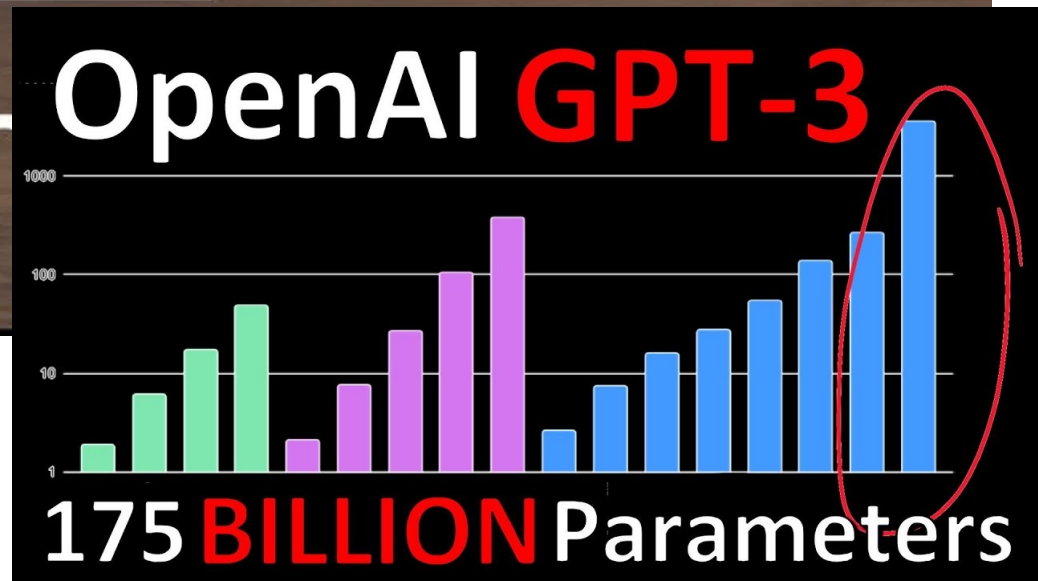
**Visual Informatics Group@UT Austin**

<https://vita-group.github.io/>

# ML researchers like to go BIG



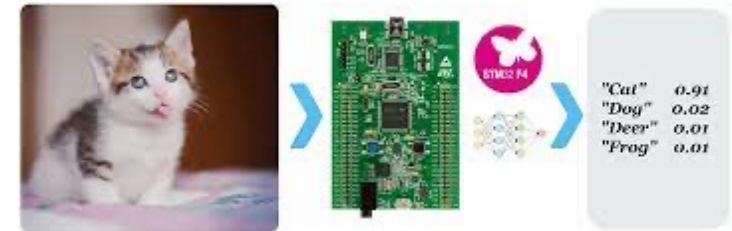
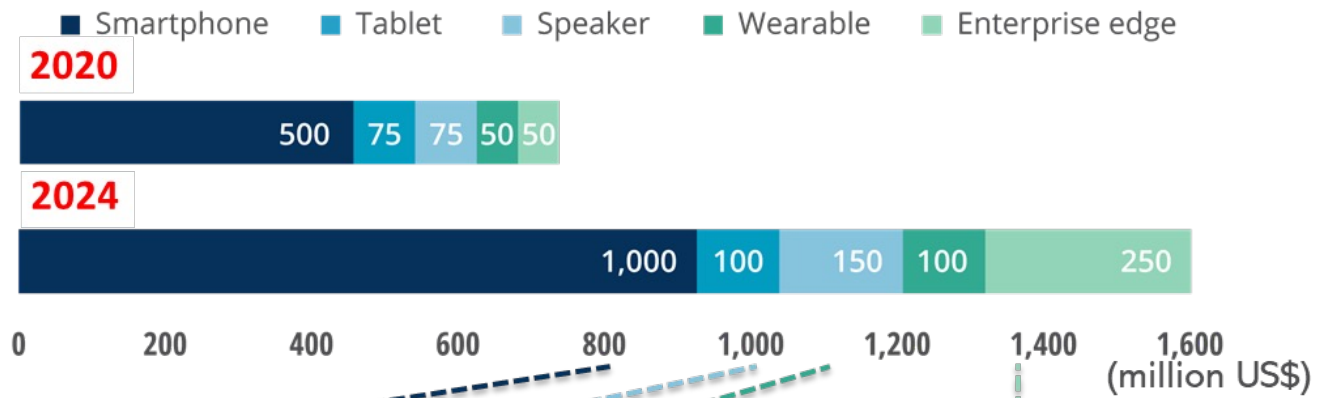
*Big NNs seem to be more capable at everything...*



**175 BILLION** Parameters

# ....While the world prefers going TINY

## Edge AI chips market trend



Phones



Speakers



Watches



Cameras

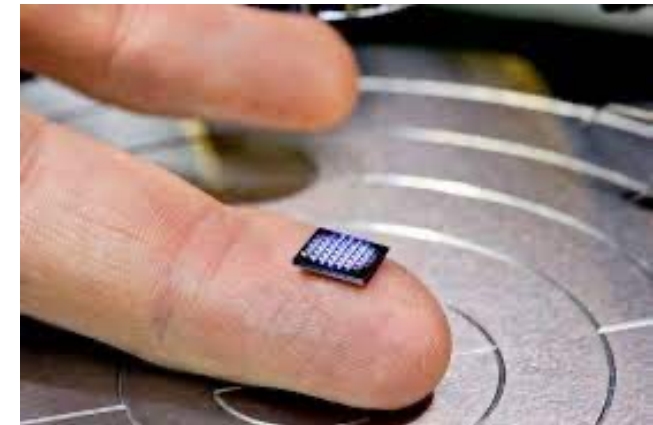


Sensors



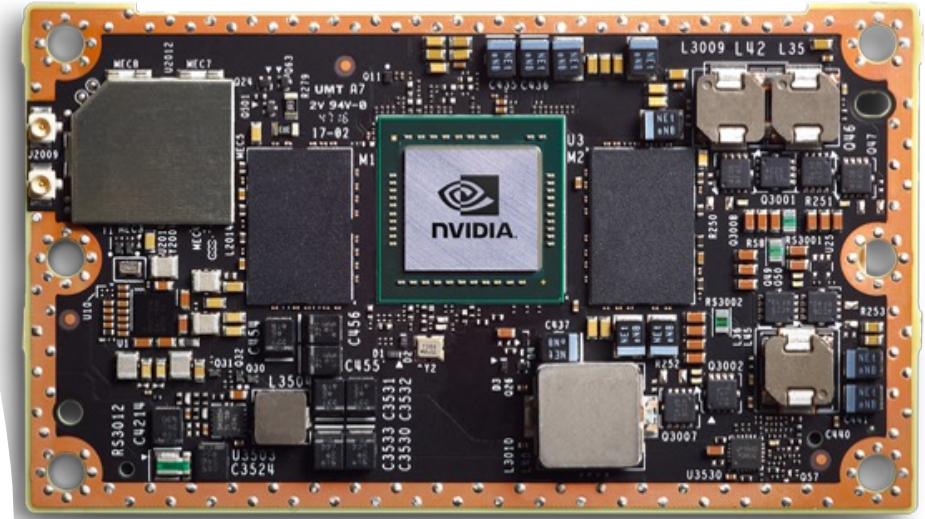
Drones

[Deloitte'19]



# Deep Learning on a Budget

- Three Top Concerns:
  - **Storage and Memory**
  - **Speed or Latency**
  - **Energy Efficiency**
- The three goals all pursue “light weight”
- ... but they are often **not aligned**\*
- ... so need to **consider all** in implementation
- ... and for both **Inference** and **Training**
- Broad economic viability requires energy efficient AI
- Energy efficiency of a brain is **100x better** than current SOTA hardware!



## Common carbon footprint benchmarks

in lbs of CO2 equivalent

Roundtrip flight b/w NY and SF (1 passenger)	1,984
Human life (avg. 1 year)	11,023
American life (avg. 1 year)	36,156
US car including fuel (avg. 1 lifetime)	126,000
Transformer (213M parameters) w/ neural architecture search	626,155

Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper

# Model Compression

- Training Phase:
  - The easiest way to extract a lot of knowledge from the training data is to learn many different models in parallel.
  - 3B: Big Data, Big Model, Big Ensemble
  - Imagenet: 1.2 million pictures in 1,000 categories.
  - AlexNet: ~ 240Mb, VGG16: ~550Mb
- Testing Phase:
  - Want small and specialist models.
  - Minimize the amount of computation and the memory footprint.
  - Real time prediction
  - Even able to run on mobile devices.

# Two Main Streams

- **“Transfer”**: How to transfer knowledge from big general model (teacher) to small specialist models (student)?
  - Example: “Distilling the Knowledge in a Neural Network”, G. Hinton et. al., 2015
- **“Compress”**: How to reduce the size of the same model, during or after training, without losing much accuracy.
  - Example: “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, S. Han et. al., 2016
- **Comparison**: Knowledge Transfer provides a way to train a new small model inheriting from big general models, while Deep Compression Directly does the surgery on big models, using a pipeline: pruning, quantization & Huffman coding.

# Knowledge Transfer/“Distillation”: Main Idea

- Introduce “Soft targets” as one way to transfer the knowledge from big models.
  - Classifiers built from a softmax function have a great deal more information contained in them than just a classifier;
  - The correlations in the softmax outputs are very informative.

- Hard Target: the ground truth label (one-hot vector)
- Soft Target:  $q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$  T is “temperature”, z is logit
- More information in soft targets

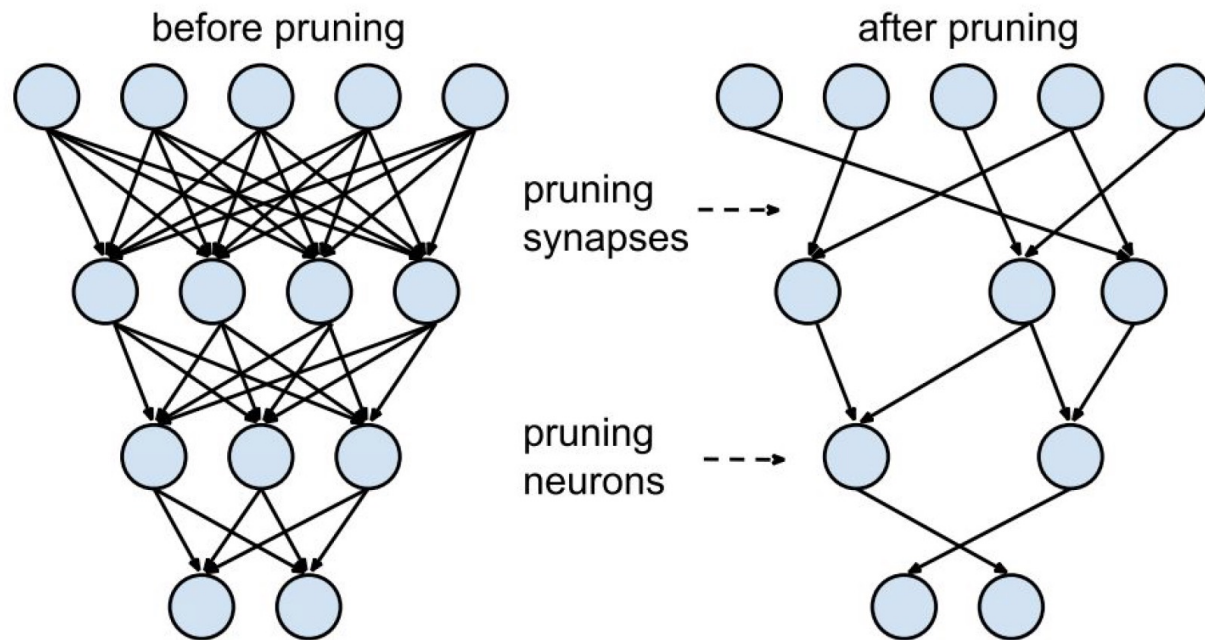
cow	dog	cat	car	
0	1	0	0	original hard targets
cow	dog	cat	car	
.05	.3	.2	.005	softened output of ensemble

**Hinton’s Observation:** If we can extract the knowledge from the data using very big models or ensembles of models, it is quite easy to distill most of it into a much smaller model for deployment.

**More follow-up observations:** teachers can be weak, or even the same as student ...

Deep  
Compression:  
Main Idea (i)

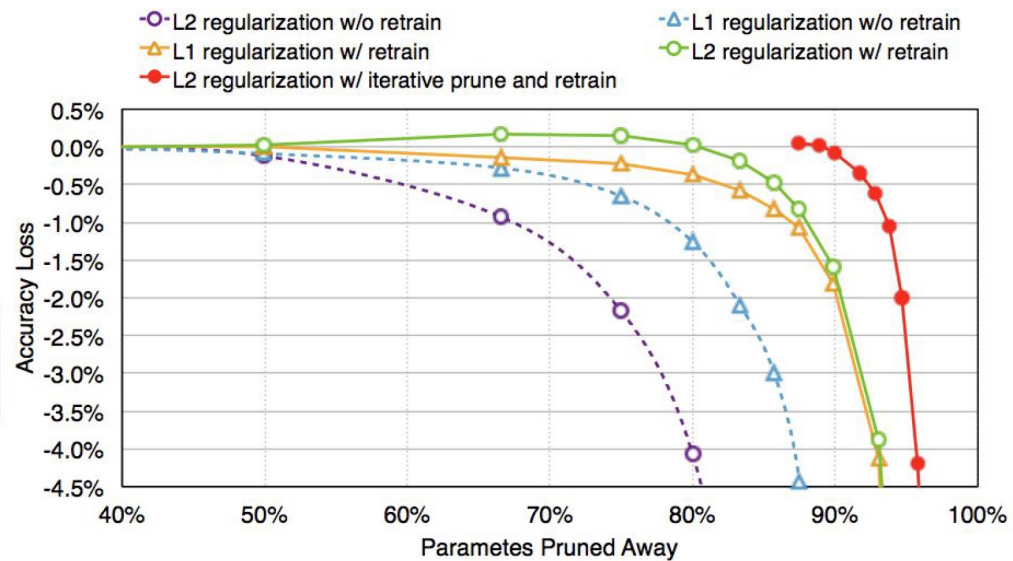
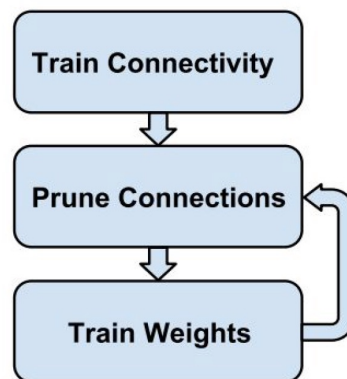
# Pruning





# Deep Compression: Main Idea (ii)

## Retrain to Recover Accuracy



Network pruning can save 9x to 13x parameters without drop in accuracy

# Deep Compression: Main Idea (iii)

## Weight Sharing (Trained Quantization)

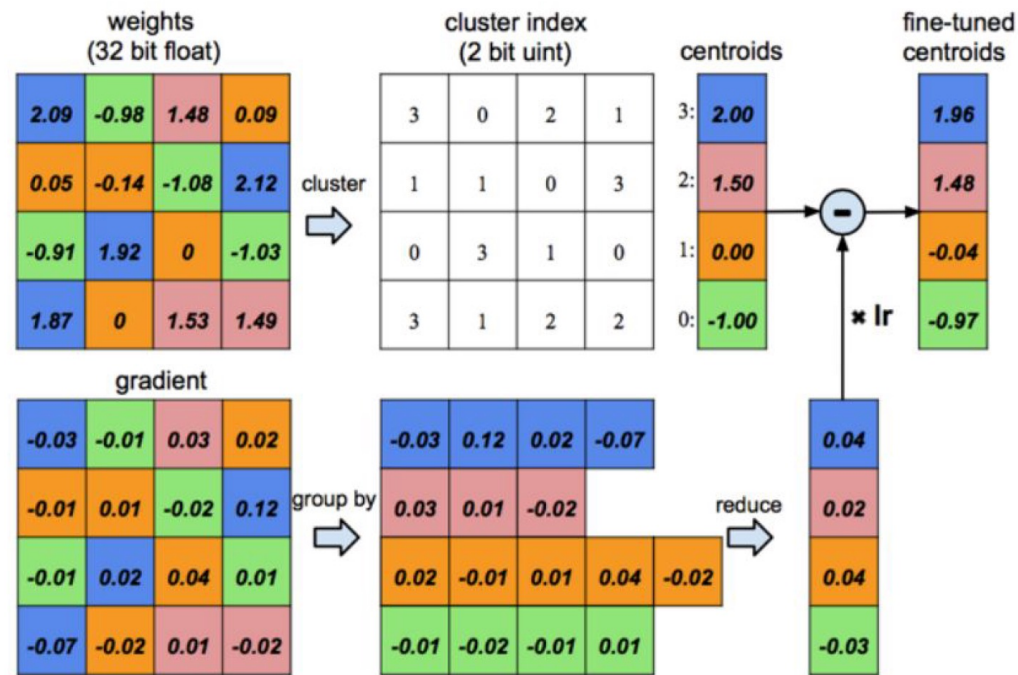
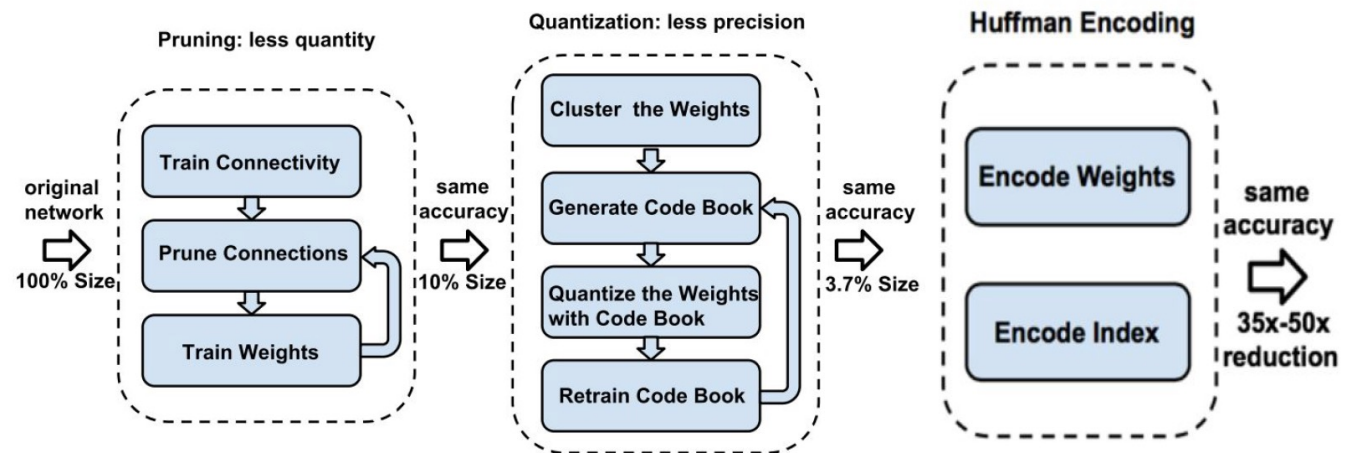


Figure 3: Weight sharing by scalar quantization (top) and centroids fine-tuning (bottom)

# Deep Compression: Main Idea (iv)

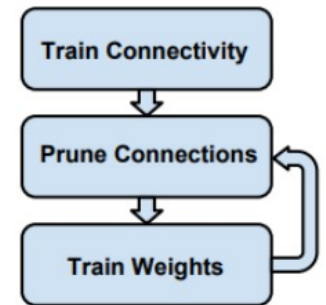
## Huffman Coding



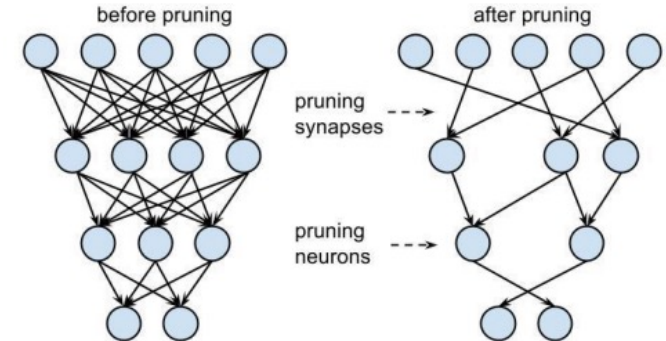
# More About Pruning

- Determining **low-saliency parameters**, given a pre-trained network
- Follows the framework proposed by LeCun et al. (1990):

1. **Train** a deep model until convergence
2. **Delete** “unimportant” connections w.r.t. a certain criteria
3. **Re-train** the network
4. **Iterate** to step 2, or stop



- Defining **which connection is unimportant** can vary
  - Weight magnitudes ( $L^2$ ,  $L^1$ , ...)
  - Mean activation [Molchanov et al., 2016]
  - Avg. % of Zeros (APoZ) [Hu et al., 2016]
  - Low entropy activation [Luo et al., 2017]
  - ...



Human Brain  
Prunes too!

50 Trillion  
Synapses



This image is in the public domain

Newborn

1000 Trillion  
Synapses



This image is in the public domain

1 year old

500 Trillion  
Synapses



This image is in the public domain

Adolescent

# Optimal Brain Damage (OBD)

- Network pruning **perturbs weights  $\mathbf{W}$**  by **zeroing** some of them
- How the **loss  $L$**  would be changed when  $\mathbf{W}$  is perturbed?
- **OBD** approximates  $L$  by the **2<sup>nd</sup> order Taylor series**:

$$\delta L \simeq \underbrace{\sum_i \frac{\partial L}{\partial w_i} \delta w_i}_{\text{1st order}} + \underbrace{\frac{1}{2} \sum_i \frac{\partial^2 L}{\partial w_i^2} \delta w_i^2 + \frac{1}{2} \sum_{i,j} \frac{\partial^2 L}{\partial w_i \partial w_j} \delta w_i \delta w_j}_{\text{2nd order}} + O(\|\delta \mathbf{W}\|^3)$$

- **Problem:** Computing  $H = \left( \frac{\partial^2 L}{\partial w_i \partial w_j} \right)_{i,j}$  is usually intractable
  - Requires  $O(n^2)$  on **# weights**
  - Neural networks usually have enormous number of weights
    - e.g. AlexNet: **60M** parameters  $\Rightarrow H$  consists  $\approx 3.6 \times 10^{15}$  elements

# Optimal Brain Damage (OBD)

- **Problem:** Computing  $H = \left( \frac{\partial^2 L}{\partial w_i \partial w_j} \right)_{i,j}$  is usually intractable

- Two additional assumptions for tractability

1. **Diagonal approximation:**  $H = \frac{\partial^2 L}{\partial w_i \partial w_j} = 0$  if  $i \neq j$

2. **Extremal assumption:**  $\frac{\partial L}{\partial w_i} = 0 \quad \forall i$

- $\mathbf{W}$  would be in a **local minima** if it's pre-trained

- Now we get:  $\delta L \simeq \frac{1}{2} \sum_i \frac{\partial^2 L}{\partial w_i^2} \delta w_i^2 + O(\|\delta \mathbf{W}\|^3)$ 
  - It only needs  $\text{diag}(H) := \left( \frac{\partial^2 L}{\partial w_i^2} \right)_i$

- $\text{diag}(H)$  can be computed in  $O(n)$ , allowing a **backprop-like algorithm**
  - For details, see [LeCun et al., 1987]

# Optimal Brain Damage (OBD)

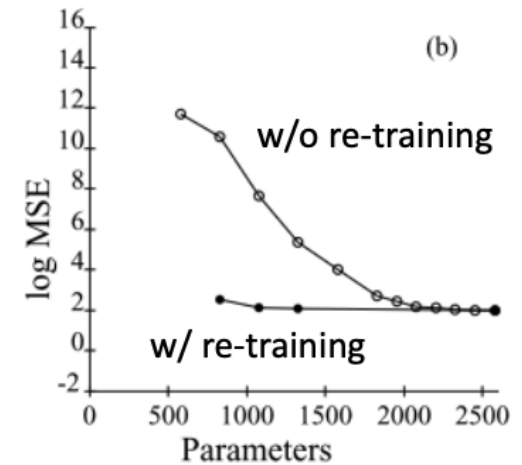
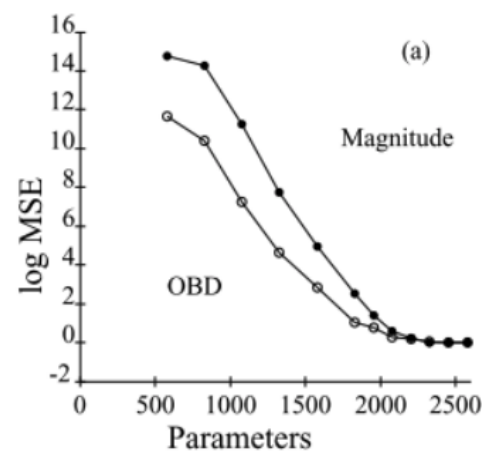
- How the **loss**  $L$  would be changed when  $\mathbf{W}$  is perturbed?

$$L(\delta\mathbf{W}) \simeq \frac{1}{2} \sum_i \frac{\partial^2 L}{\partial w_i^2} \delta w_i^2 =: \sum_i \frac{1}{2} h_{ii} \delta w_i^2$$

- The **saliency** for each weight  $\Rightarrow s_i := \frac{1}{2} h_{ii} |w_i|^2$

$$s_i := |w_i|$$

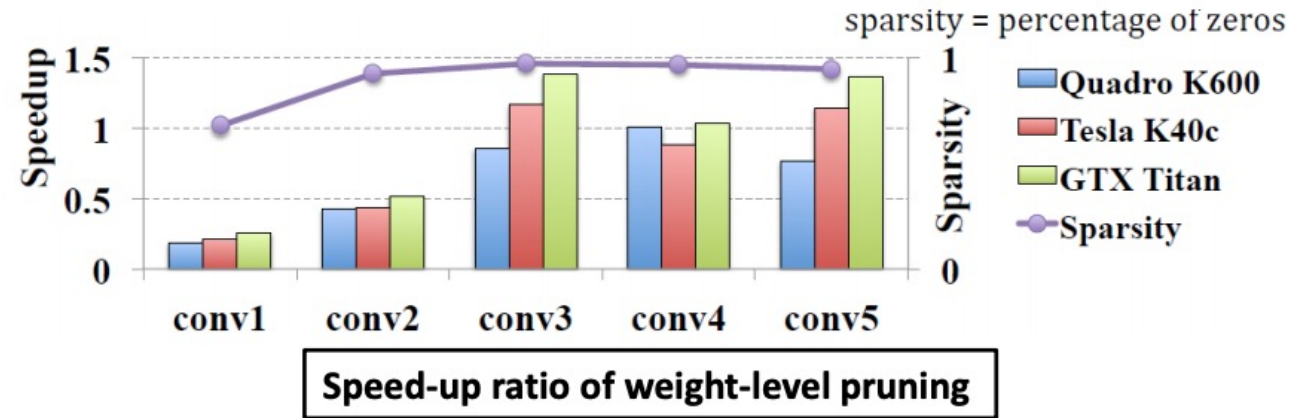
- OBD shows **robustness on pruning** compared to magnitude-based deletion
- After re-training, the original test accuracy is **recovered**





# Structured Sparsity

- “Un-structured” **weight-level pruning** may not engage a **practical speed-up**
  - Despite of extremely high sparsity, actual speed-ups in GPU is limited



Non-structured sparsity (poor data pattern)



Structured sparsity (regular data pattern)



5x speedup after concatenation of nonzero rows and columns

# Structured sparsity

- **Structured sparsity** can be induced by adding **group-lasso regularization**

$$\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) + \lambda \sum_{l=1}^L R_g(\mathbf{W}^{(l)}), \quad R_g(\mathbf{w}) = \sum_{g=1}^G \|\mathbf{w}^{(g)}\|_2$$

- **Filter-wise and channel-wise:**

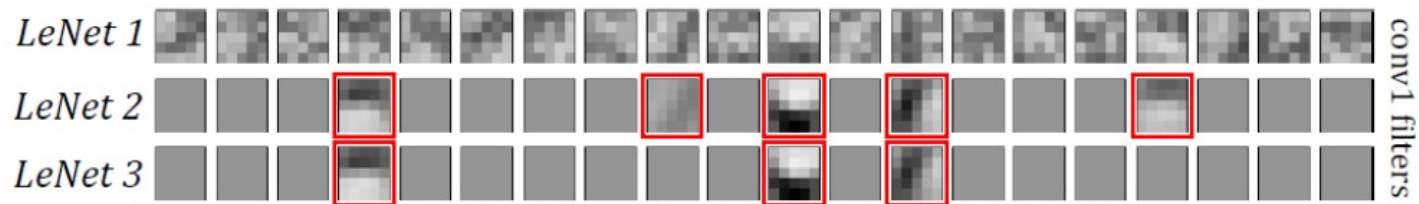
$$R_g(\mathbf{W}^{(l)}) = \sum_{n_l=1}^{N_l} \|\mathbf{W}_{n_l, :, :, :}^{(l)}\|_2 + \sum_{c_l=1}^{C_l} \|\mathbf{W}_{:, c_l, :, :}^{(l)}\|_2$$

# filters
# channels

Table 1: Results after penalizing unimportant filters and channels in *LeNet*

<i>LeNet</i> #	Error	Filter # <sup>§</sup>	Channel # <sup>§</sup>	FLOP <sup>§</sup>	Speedup <sup>§</sup>
1 ( <i>baseline</i> )	0.9%	20—50	1—20	100%—100%	1.00×—1.00×
2	0.8%	5—19	1—4	25%—7.6%	1.64×—5.23×
3	1.0%	3—12	1—3	15%—3.6%	1.99×—7.44×

<sup>§</sup>In the order of *conv1—conv2*



Fewer but smoother feature extractors

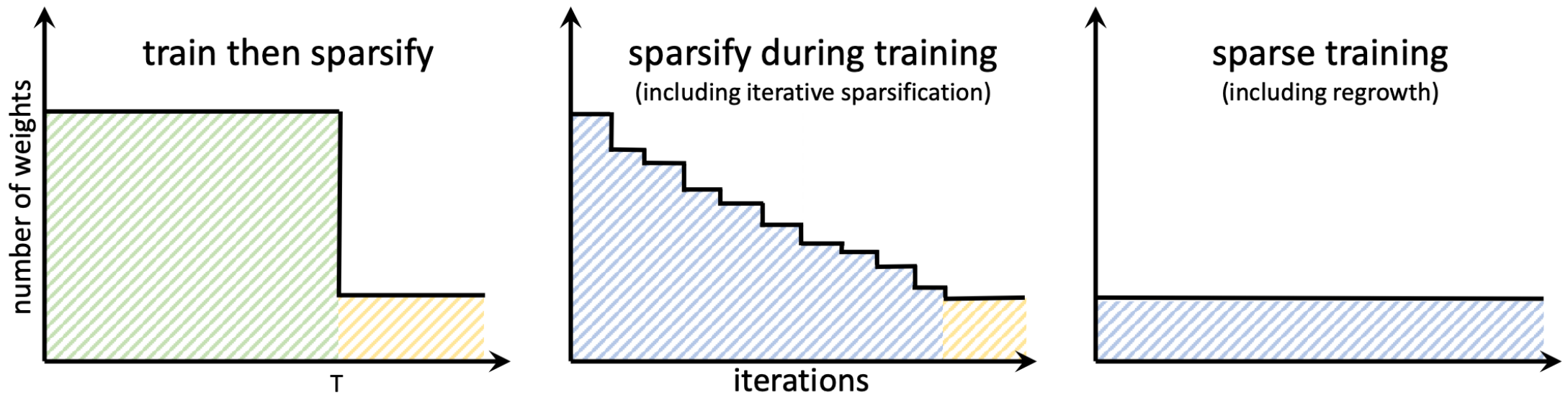


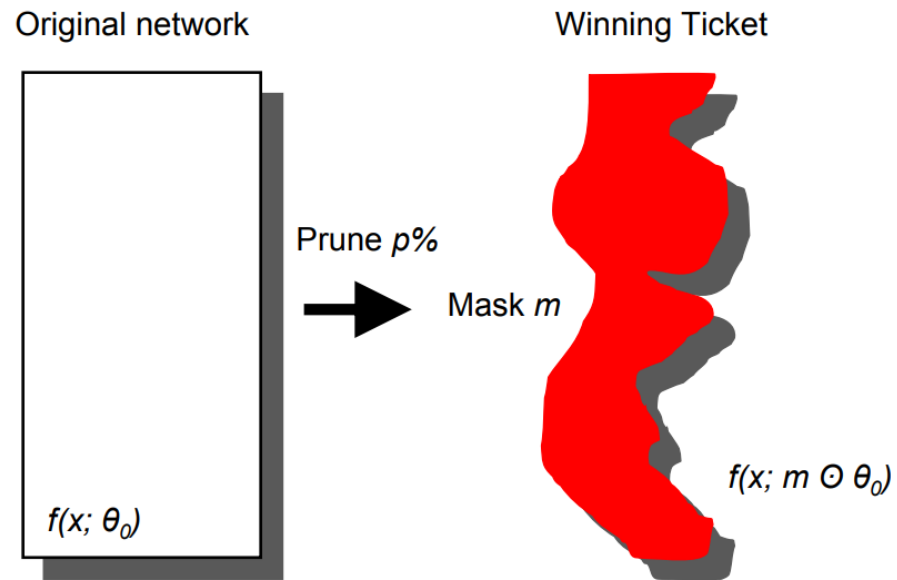
Figure 7: Overview of structural sparsification schedules.

Sparsity beyond  
post-training  
compression

- Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241), 1-124.

# Lottery Ticket Hypothesis

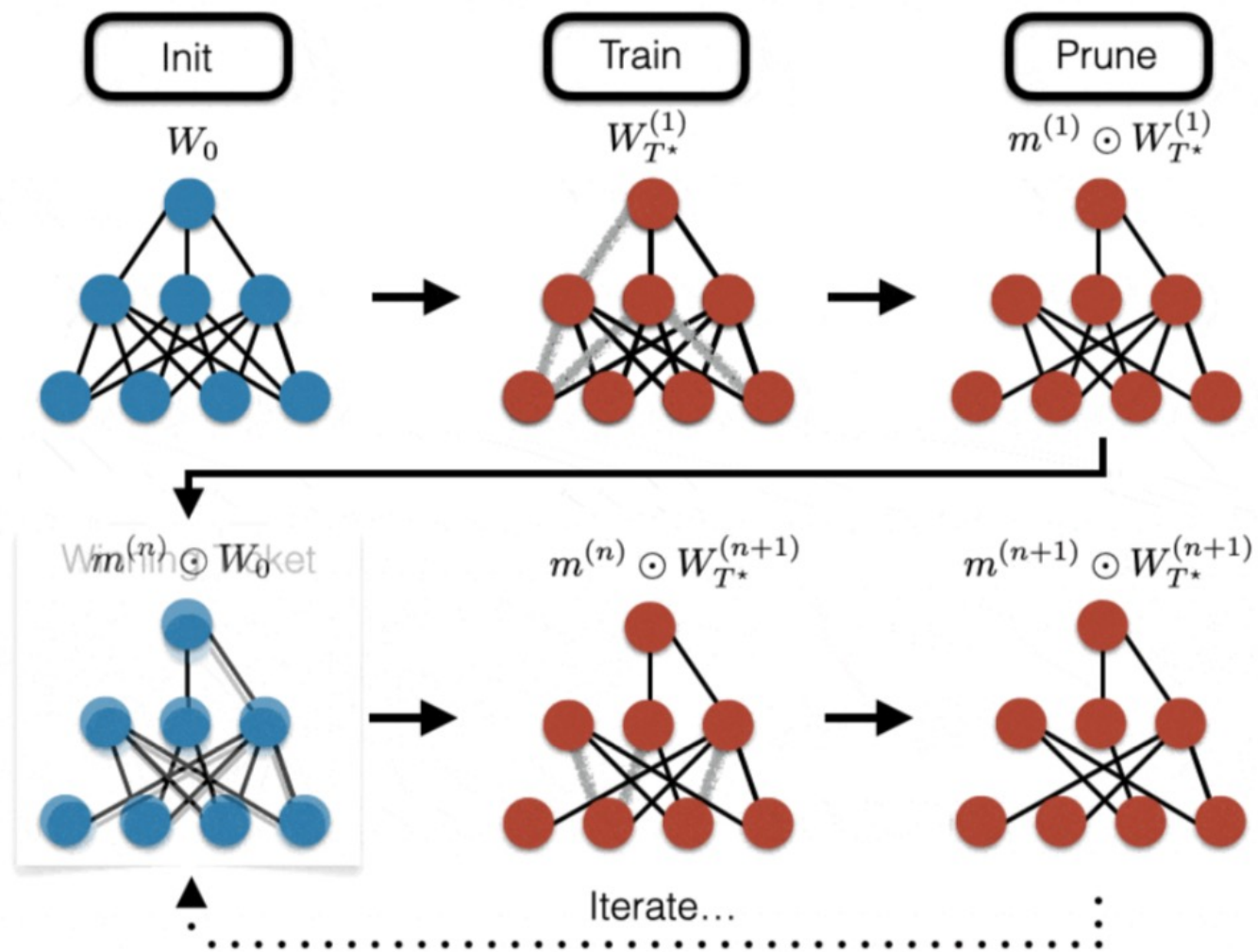
**The Lottery Ticket Hypothesis.** *A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.*



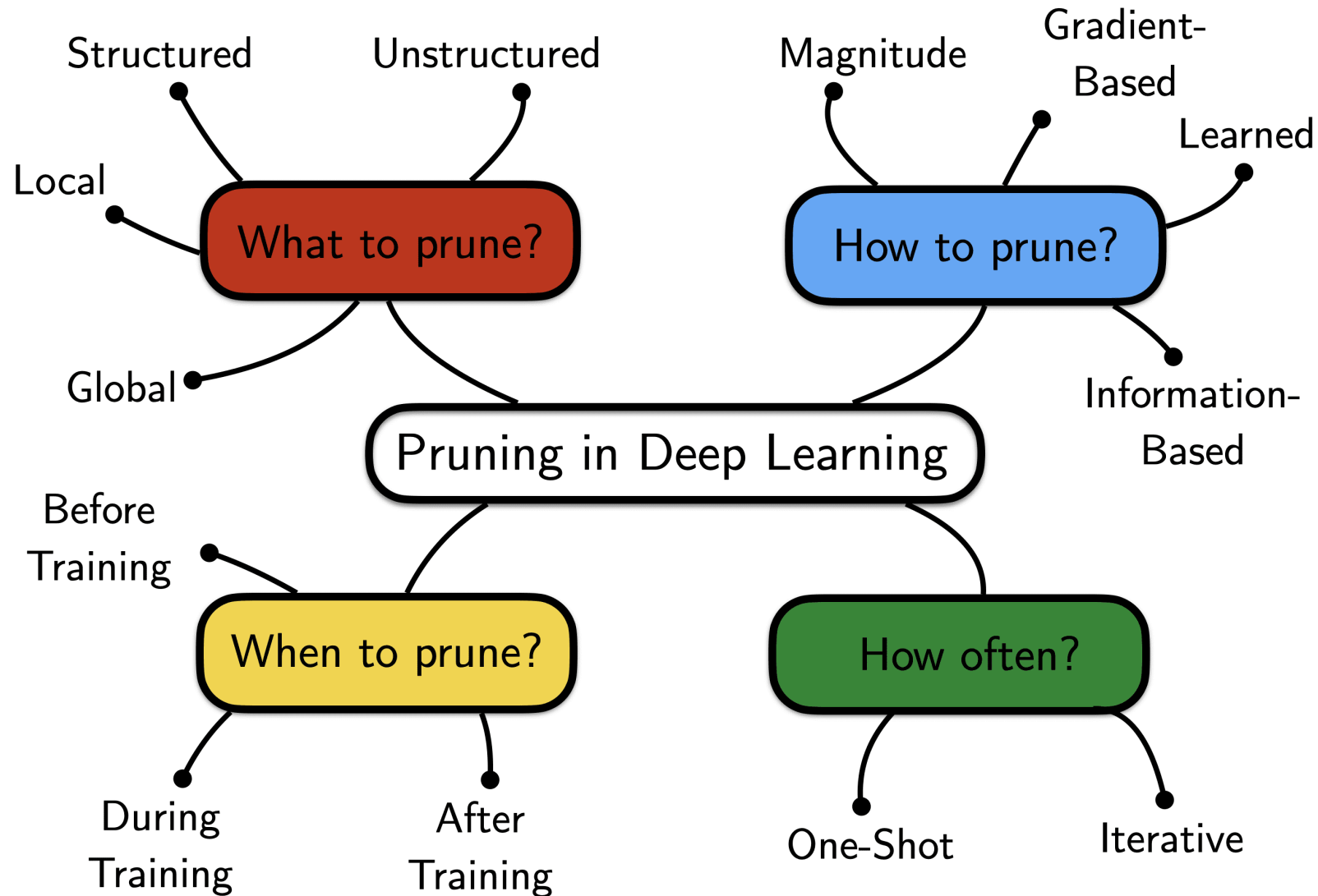
- Winning Ticket gives
  - Better or same results
  - Shorter or same training time
  - Notably fewer parameters
  - Is trainable from the beginning

# Lottery Ticket Hypothesis

## Searching for Tickets: Iterative Magnitude Pruning

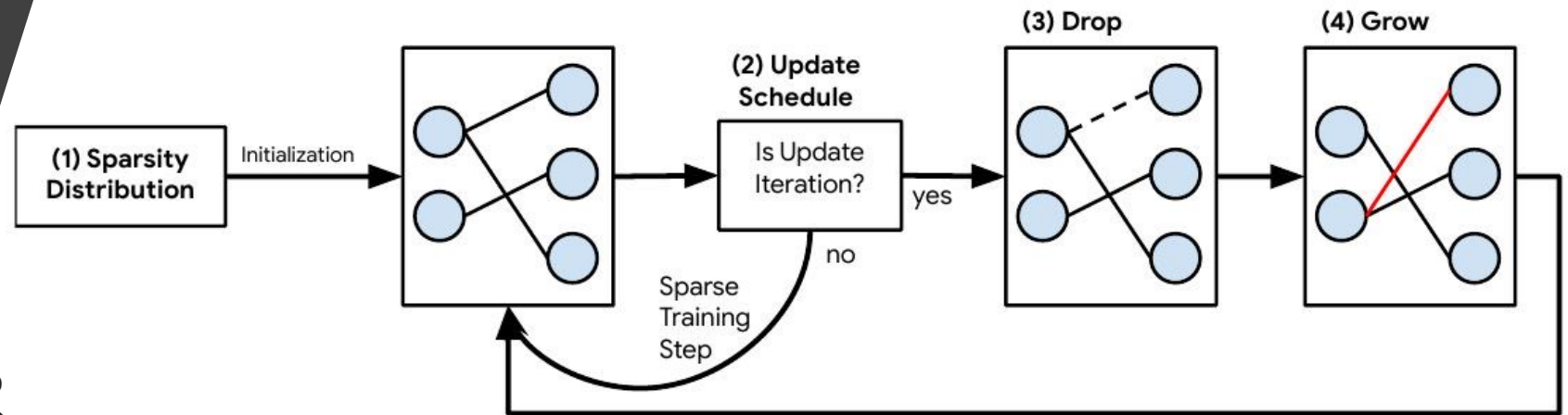


# Summary of Pruning



# End-to-End (Dynamic) Sparse Training

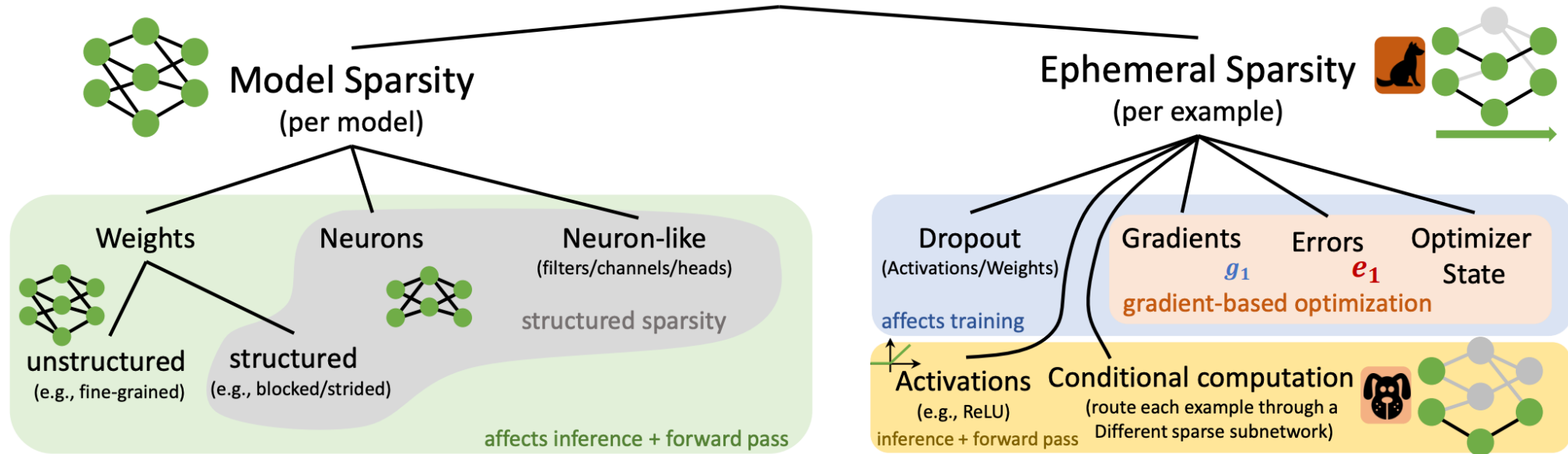
- **Sparsity distribution.** The simplest is “uniform” - every layer has the same sparsity. More advanced ones work better, e.g., bigger layers are pruned more than small layers (called “ERK”)
- **Update schedule.** Sparsification happens at a certain frequency during training (btw, sparse training usually costs more epochs to converge)
- **Drop criterion.** The weights with the **lowest magnitude** are dropped.
- **Grow criterion.** The weights receiving the **highest gradient** will be re-added (zero-init). The number grown connections is the same as the dropped.



Evci, Utku, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. "Rigging the lottery: Making all tickets winners." *ICML 2020*

Figure 1: Dynamic sparse training changes connectivity during training to aid optimization.

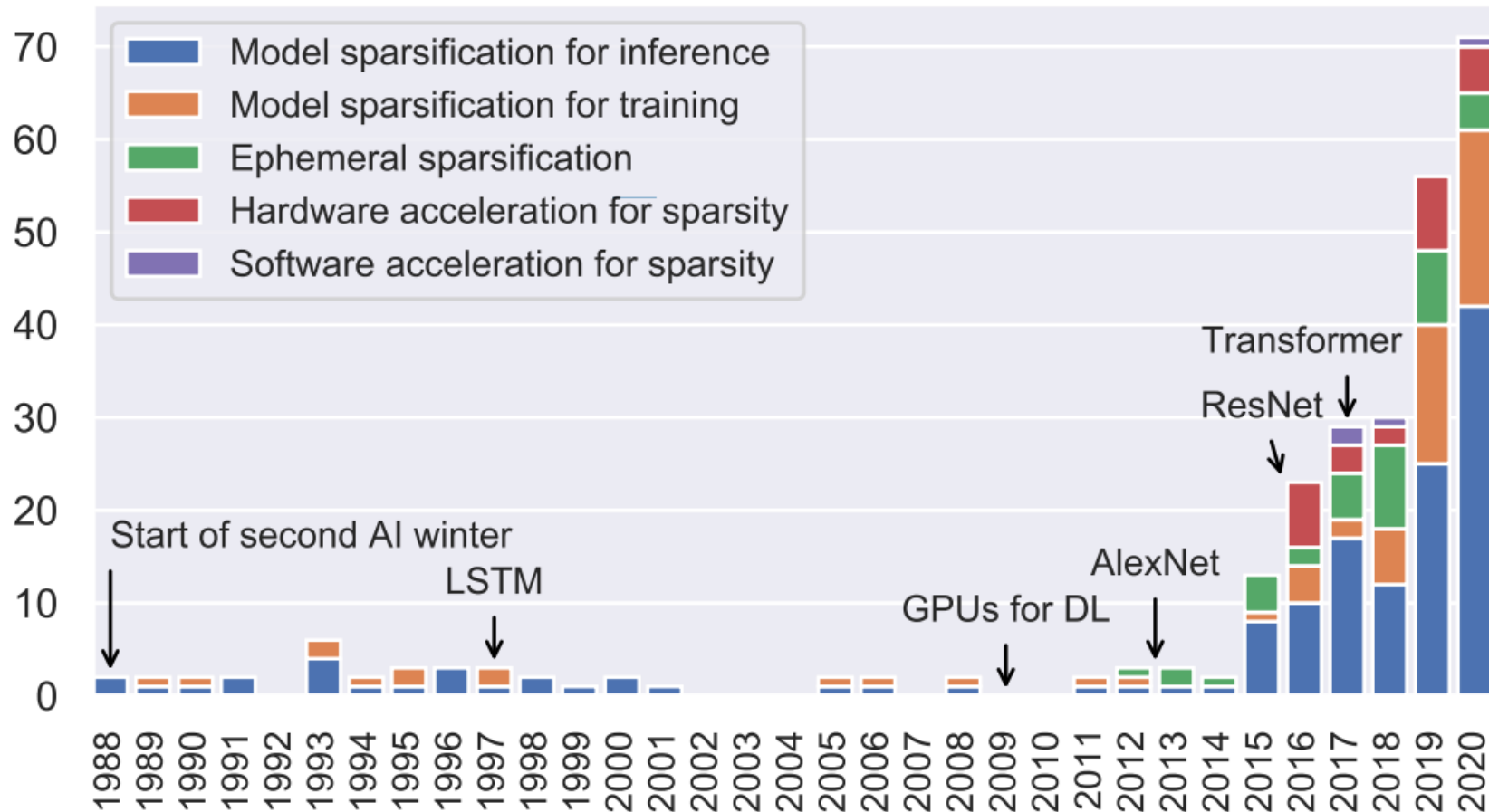
# Sparsification



“Sparsity”, in broader terms

- Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241), 1-124.





Sparse NN  
research is on fire!

- Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241), 1-124.

# Sparsity May Cry Benchmark (SMC-Bench)

(ICLR'23) Large-scale, arduous and diverse benchmark for sparse NNs with 4 tasks and 12 datasets

Table 2: Summary of models and datasets that we used to evaluate on SMC-Bench.

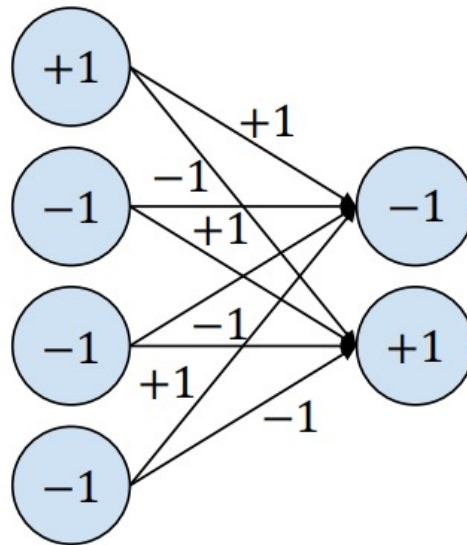
Task	Datasets	Models	Source
Commonsense Reasoning	CSQA	RoBERTa Large	Facebook AI Research Sequence-to-Sequence Toolkit (Ott et al., 2019)
	WinoGrande	RoBERTa Large	Facebook AI Research Sequence-to-Sequence Toolkit (Ott et al., 2019)
	RACE	RoBERTa Large	Facebook AI Research Sequence-to-Sequence Toolkit (Ott et al., 2019)
Arithmetic Reasoning	MAVPS	GTS	GitHub repository (Patel et al., 2021)
		Graph2Tree	GitHub Repository (Patel et al., 2021)
	ASDiv-A	Graph2Tree	GitHub Repository (Patel et al., 2021)
		Graph2Tree	GitHub Repository (Patel et al., 2021)
Protein Thermostability Prediction	SVAMP	GTS	GitHub Repository (Patel et al., 2021)
		Graph2Tree	GitHub Repository (Patel et al., 2021)
	HotProtein (HP-S)	ESM-1B	GitHub Repository (Rives et al., 2021)
	HotProtein (HP-S <sup>2</sup> C5)	ESM-IF1	GitHub Repository (Rives et al., 2021)
	HotProtein (HP-S <sup>2</sup> C2)	ESM-1B	GitHub Repository (Rives et al., 2021)
	Meltome Atlas	ESM-1B	GitHub Repository (Rives et al., 2021)
Multilingual Translation	2-to-2	mBART	Facebook AI Research Sequence-to-Sequence Toolkit (Ott et al., 2019)
	5-to-5	mBART	Facebook AI Research Sequence-to-Sequence Toolkit (Ott et al., 2019)
	10-to-10	mBART	Facebook AI Research Sequence-to-Sequence Toolkit (Ott et al., 2019)

Short Story:  
Almost No Sparsity Works!

# More About Quantization

- Neural networks can be even **binarized (+1 or -1)**
  - DNNs trained to use **binary** weights and **binary** activations
- Expensive **32-bit MAC (Multiply-ACcumulate)**  $\Rightarrow$  Cheap **1-bit XNOR-Count**
  - “MAC == XNOR-Count”: when the weights and activations are  $\pm 1$

$\nwarrow$   
# 1s in bits



Binarized weights



Binarized feature maps



# Binary Neural Networks

- **Idea:** Training real-valued nets ( $W_r$ ) treating binarization ( $W_b$ ) **as noise**
  - Training  $W_r$  is done by **stochastic gradient descent**

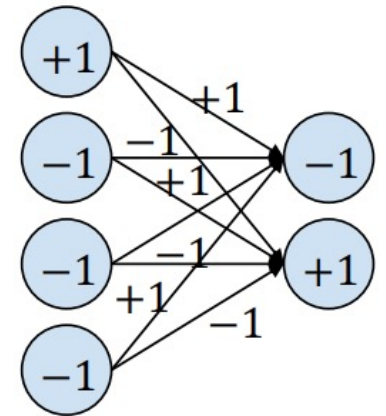
- **Binarization** ( $W_r \rightarrow W_b$ ) occurs for each forward propagation
  - On each of **weights**:  $W_b = \text{sign}(W_r)$
  - ... also on each **activation**:  $a_b = \text{sign}(a_r)$

- Gradients for  $W_r$  is estimated from  $\frac{\partial L}{\partial W_b}$  [Bengio et al., 2013]
  - “Straight-through estimator”: **ignore** the binarization during backward!

$$\frac{\partial L}{\partial W_r} = \frac{\partial L}{\partial W_b} \mathbf{1}_{|W_r| \leq 1}$$

$$\frac{\partial L}{\partial a_r} = \frac{\partial L}{\partial a_b} \mathbf{1}_{|a_r| \leq 1}$$

- Cancelling gradients for better performance
  - When the value is too large

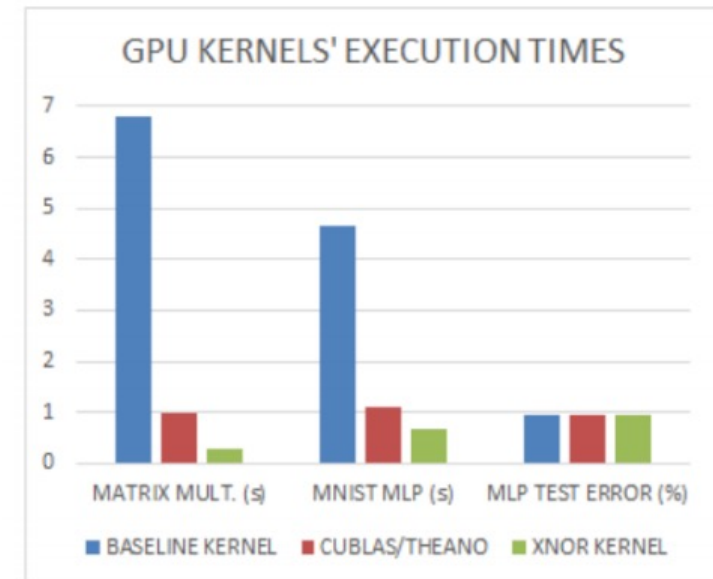


# Binary Neural Networks

- BNN yields **32x less memory** compared to the baseline 32-bit DNNs
  - ... also expected to reduce energy consumption drastically

- **23x faster** on kernel execution times
  - BNN allows us to use XNOR kernels
  - **3.4x** faster than cuBLAS

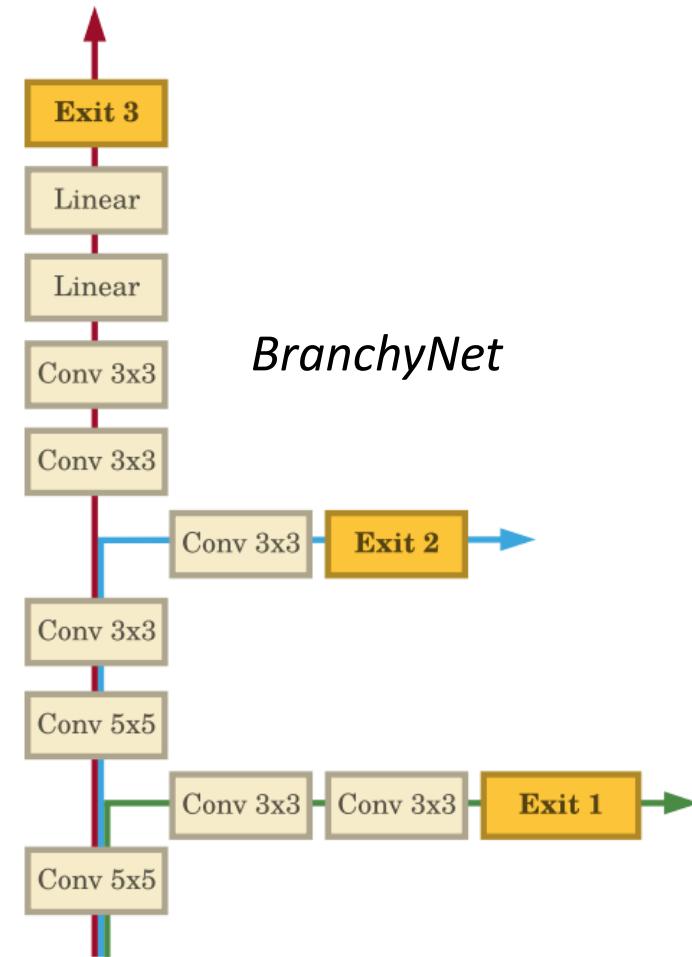
Operation	MUL	ADD
8bit Integer	0.2pJ	0.03pJ
32bit Integer	3.1pJ	0.1pJ
16bit Floating Point	1.1pJ	0.4pJ
32tbit Floating Point	3.7pJ	0.9pJ



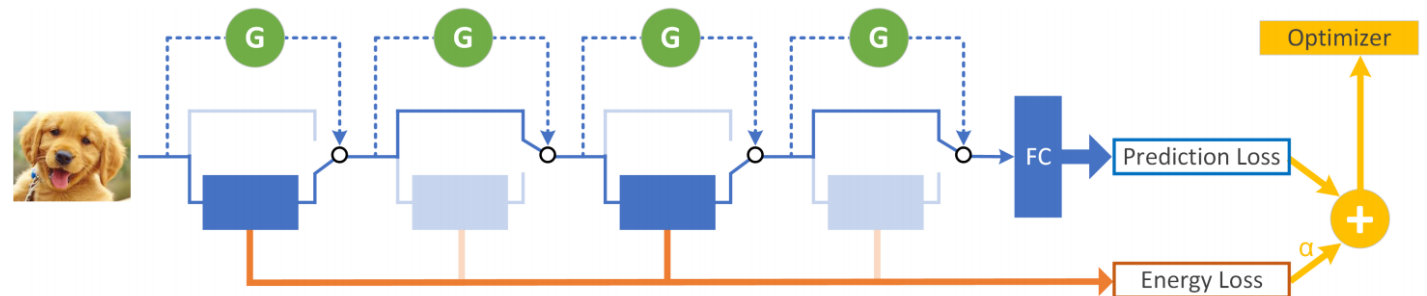
- **BNN** achieves comparable error rates over existing DNNs

# Dynamic Inference

- Only execute a fraction of the network per needed
- Can enable both “input-dependent” and “resource-dependent” forms



*SkipNet*



# Real-World Efficient ML: Way to Go

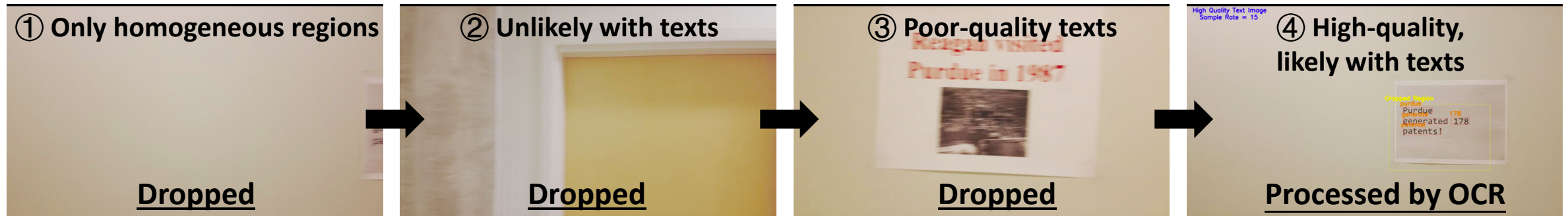
- Jointly utilizing several compression means
  - Also, can choose efficient “by-design” models (MobileNets, or even non-deep models, etc.)
  - Channel pruning is in fact very similar to NAS
- **Data processing** is often a key concern, maybe more important
- **Hardware co-design** is another key concern
- Resource constraints & user demands often **change over time**
- From single task to multi-task and lifelong learning ...

# Demo: Energy-Efficient UAV-Based Text Spotting System

- **Task:** UAV-based low-energy video understanding ([Raspberry Pi 3B+](#))
- Our group has been leading the show!
  - **2021 IEEE Low-Power Computer Vision (LPCV) Challenge, 1st prize (video track)** among 31 university & company teams that submitted 249 independent solutions
  - **2020 IEEE Low-Power Computer Vision (LPCV) Challenge, 2nd prize (video track)**, among ~ 90 solutions



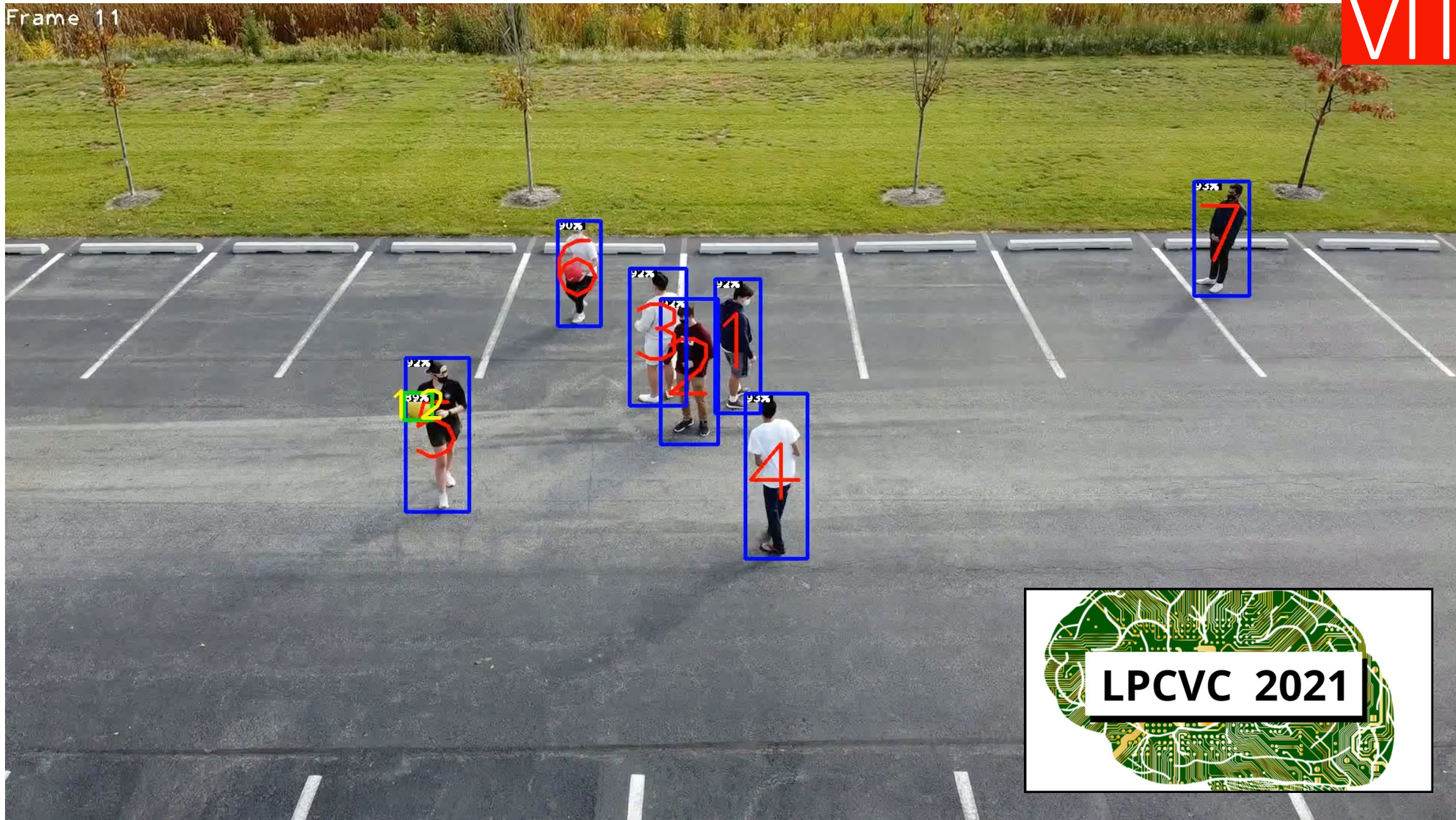
## 2020 Low-Power Computer Vision Challenge





Frame 11

VITA



94%

99%

12

90%

94%

3

94%

2

94%

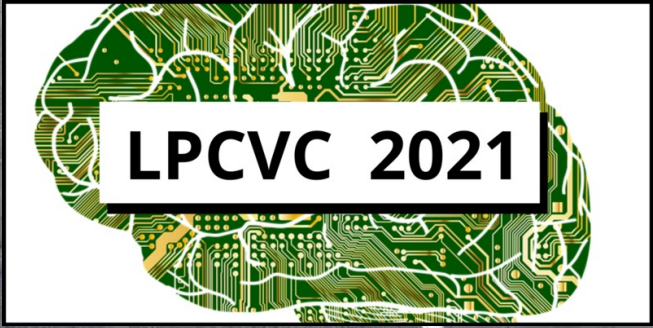
1

95%

4

95%

7



# From Efficient Inference to Efficient Training

## Two type of demands dominate:

- **“Personalization”** (or adaptation, continual learning) at the **edge** (**resource-constrained** device): saving communication bandwidth /energy & protecting data privacy etc.
  - Mostly **fine-tuning** (new unseen data, etc.)
- **“Scaling up”** bigger models at the **data center** (**resource-rich** cloud server), while keep relatively affordable training budget & suppressing carbon footprint, etc.
  - Both **training from scratch**, and **transfer learning** (new task type, new data, etc.)



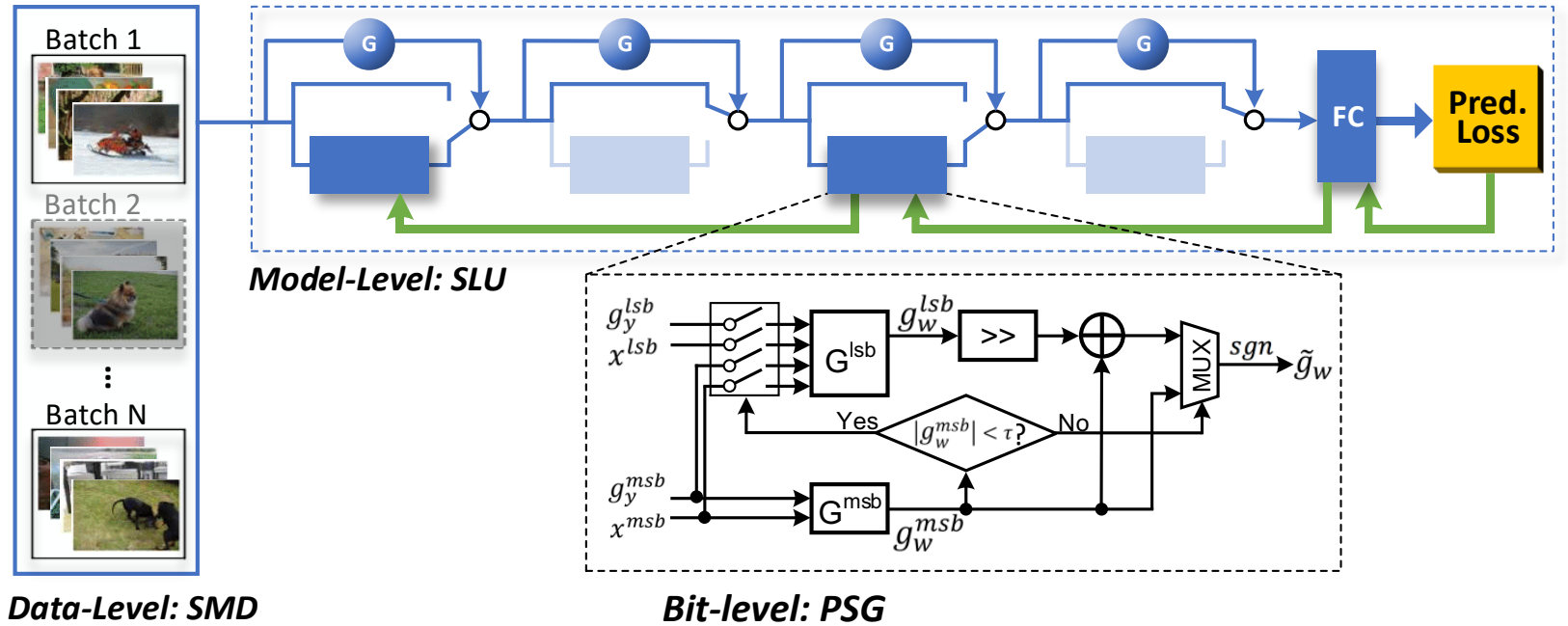
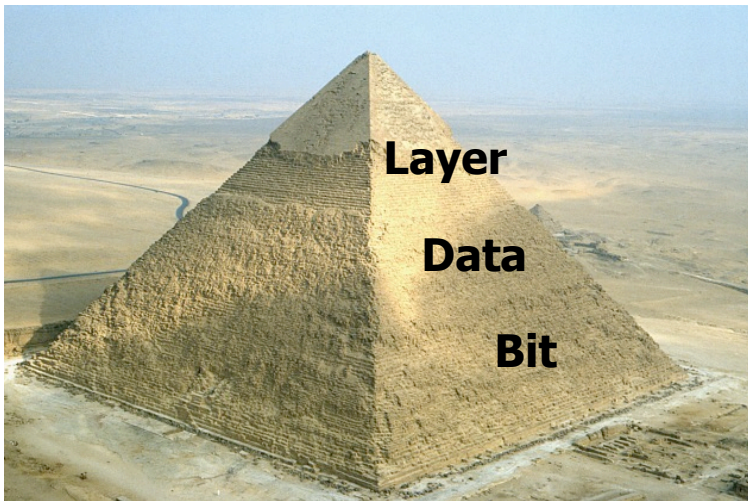
# Edge-based Training: Lessons from Efficient Inference?

- **Training v.s. Inference:** one-pass feedforward v.s. iterative forward + backward
- **Lessons that we learned from Inference:**
  - Model parameters are not born equally, and many redundancies do exist
  - Know your specific goal: saving memory, latency and energy are often not aligned
  - To achieve energy goal, realistic energy models and/or hardware measurements are very helpful
  - Consider a more “end-to-end” effort beyond just the model itself (data, hardware, architecture...)
- **New Challenges posed for Training:**
  - Saving per-sample (mini-batch) complexity (both feed-forward and backward)
  - The empirical convergence (how many iterations needed) matters more than per-MB complexity
  - Data access/movement bottlenecks are (even more) crucial

# E2-Train: Energy-Efficient CNN Training (NeurIPS'19)



## Motivation:

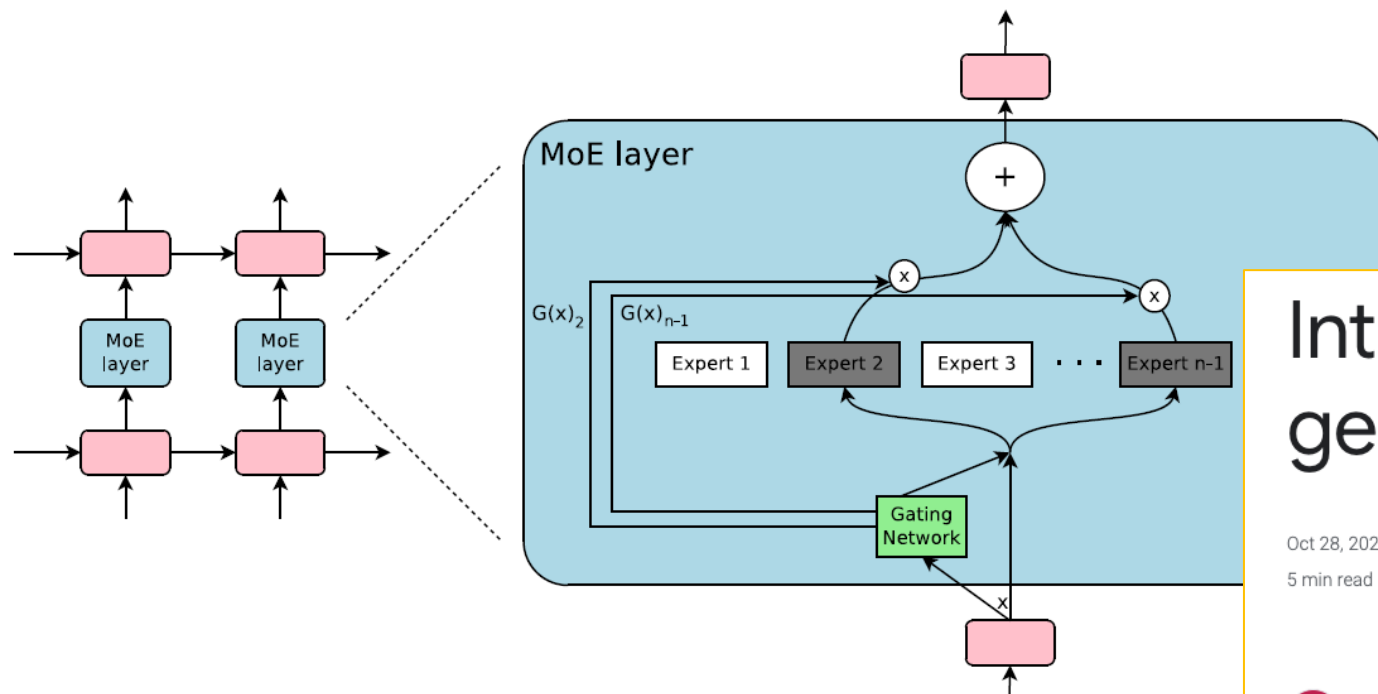


## “Three-Pronged” Approach:

- **Data-Level:** stochastic mini-batch dropping
- **Layer-Level:** selective layer update
- **Bit-Level:** predictive sign gradient descent

Datasets	Models	Accuracy (vs. Original One)	Energy Savings
CIFAR-10	MobileNetV2	92.06% (vs. 92.47%)	<b>88%</b>
	ResNet-110	93.01% (vs. 93.57%)	<b>83%</b>
CIFAR-100	MobileNetV2	71.61% (vs. 71.91%)	<b>88%</b>
	ResNet-110	71.63% (vs. 71.60%)	<b>84%</b>

# Efficiently Scaling and Training from Scratch: Mixture of Experts (MoEs)



## Introducing Pathways: A next-generation AI architecture

Oct 28, 2021  
5 min read

Too often, machine learning systems overspecialize at individual tasks, when they could be doing much better if we're building Pathways—a new AI architecture that will handle many tasks at once, learn from a wider range of data, and reflect a better understanding of the world.

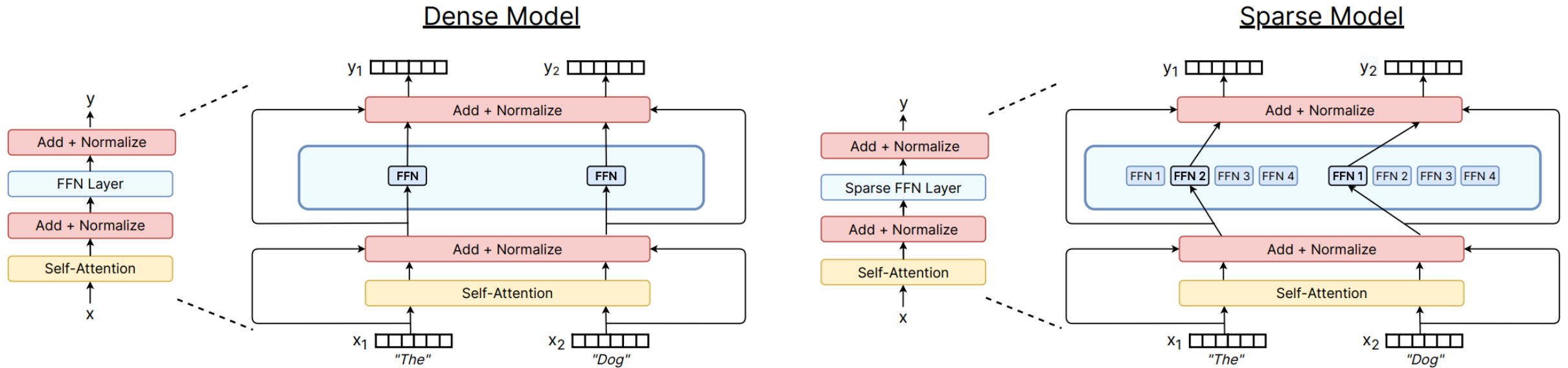


Jeff Dean  
Google Senior Fellow and SVP, Google Research

# Why MoE?

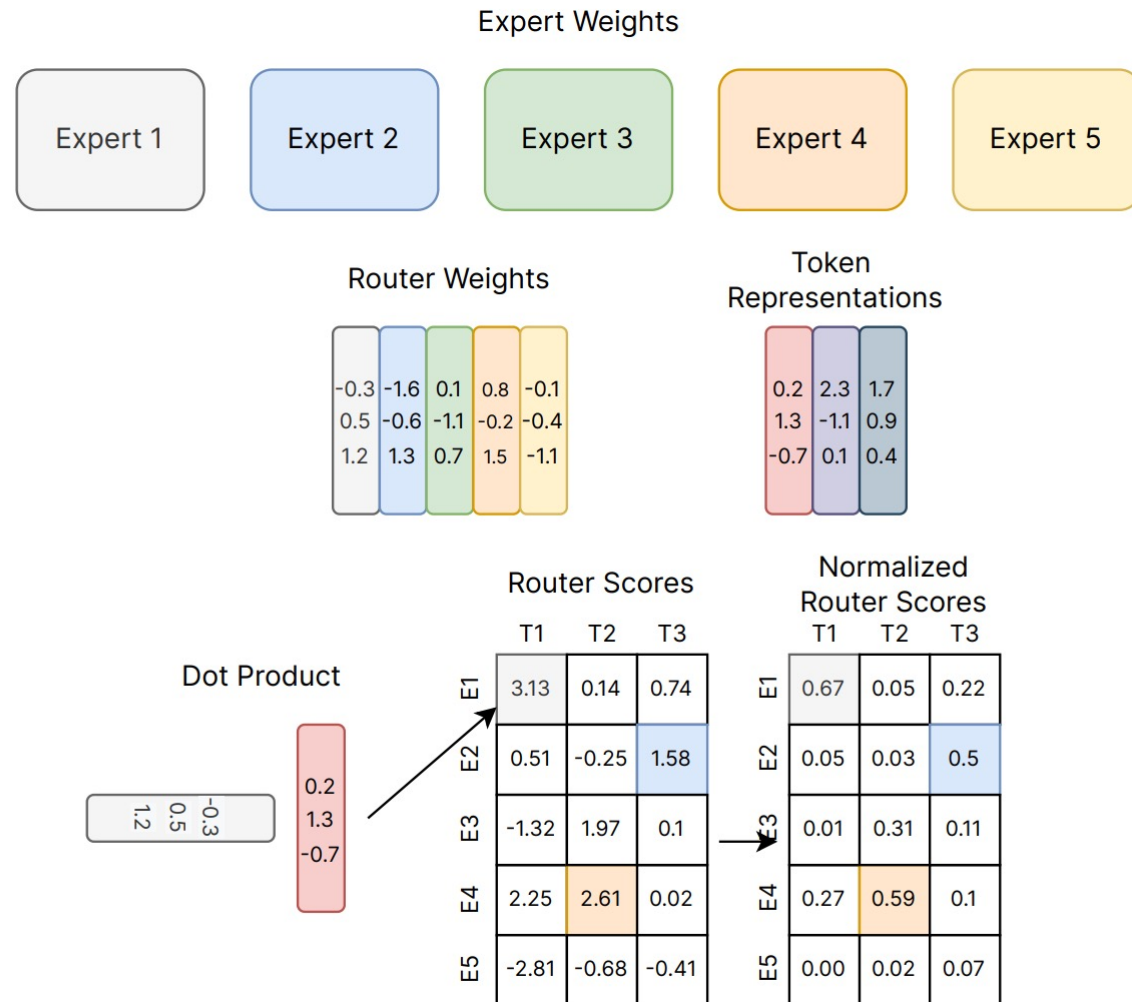
- MoE is a special type of sparsity (**dynamic, structured, end-to-end**)
  - “Modalized” structure is naturally good for distributed training/parallelism
  - “Block-level” sparsity is hardware-friendly
  - “End-to-end” sparsity keeps the memory /compute low at any point of training
- MoE is also a special type of dynamic inference
  - Dynamically activate an “input-dependent” subnetwork for a new test sample
  - The activation is controlled by a routing network (top-k classifier, RL, hashing...)
- MoE can be straightforwardly extended to “divide and conquer” ...
  - Multi-task learning
  - Multi-modality learning

# Dense versus Sparse MoE Transformer



Fedus, William, Jeff Dean, and Barret Zoph. "A review of sparse expert models in deep learning." *arXiv preprint arXiv:2209.01667* (2022).

# Schematic of Routing Network (using top- $k$ as example)

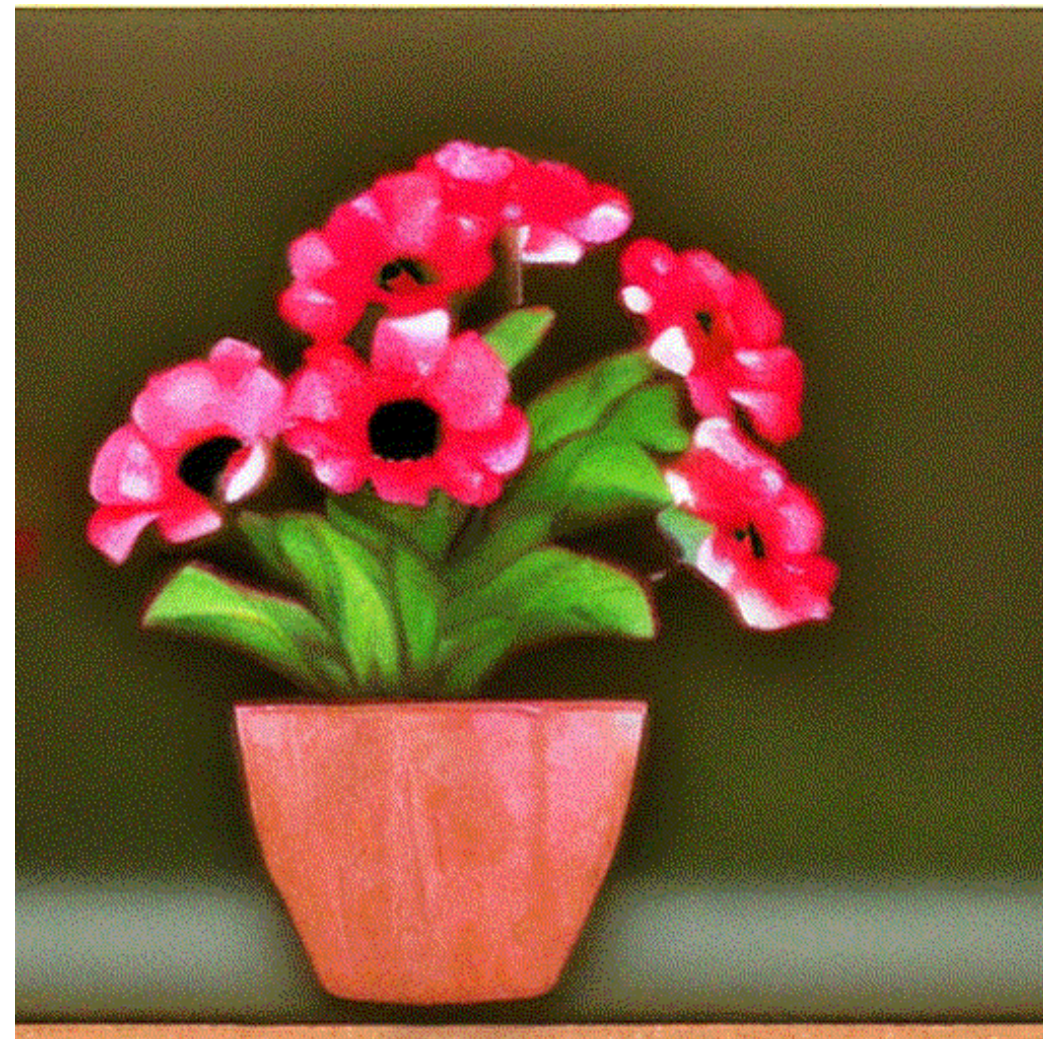
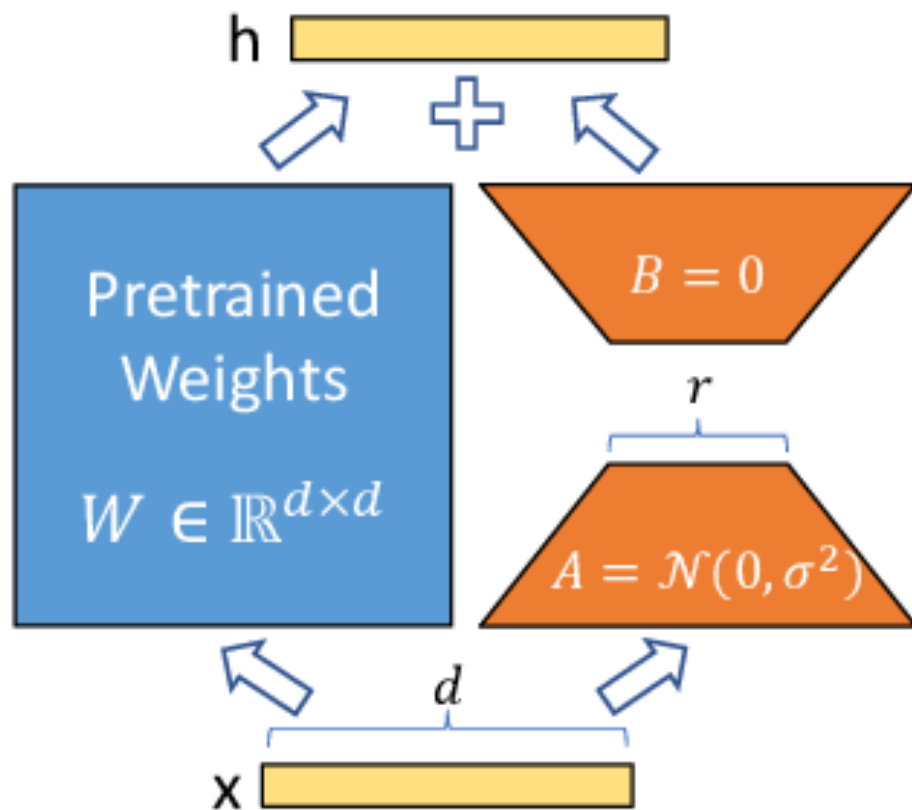


Many open challenges remain on routing!

- Expert load balancing
- Representational Collapse
- "In-situ" change sparsity  $k$ ?
- ...



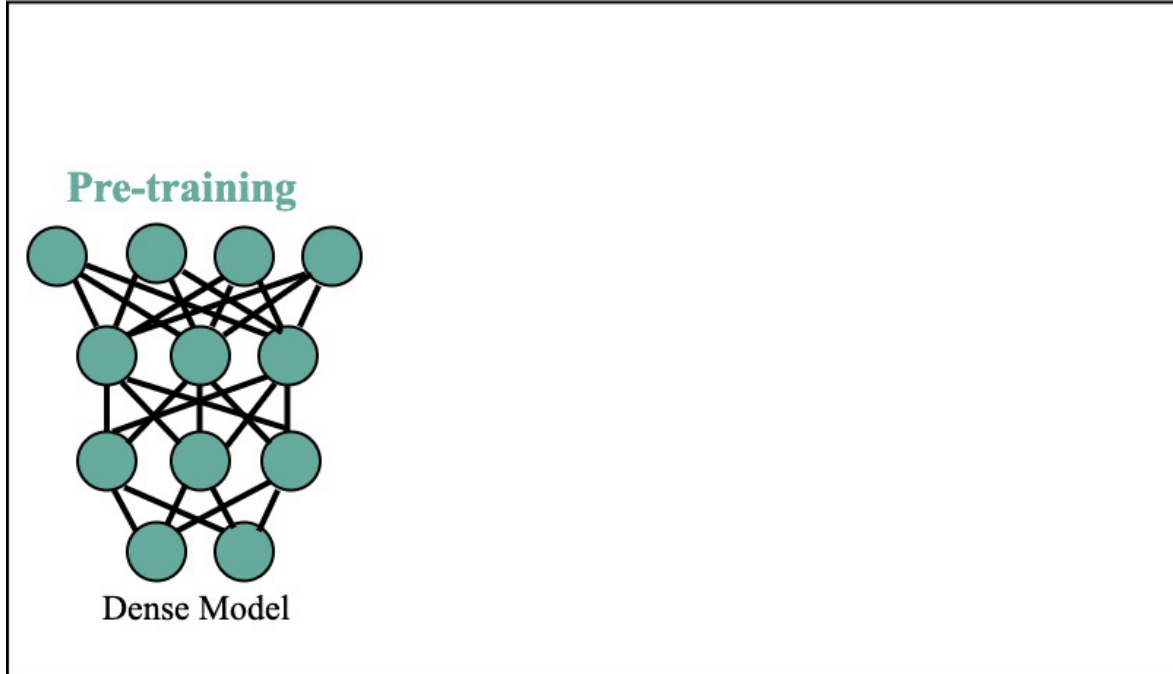
# LoRA: Low-Rank Fine-Tuning



Hu, Edward J., Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "LoRA: Low-Rank Adaptation of Large Language Models." ICLR 2022

**Recent success:** fine-tune GenAI Text2Image Models! (<https://github.com/cloneofsimon/lora>)

# Sparse Transfer Learning using Lottery Ticket Hypothesis (NeurIPS'20, ICLR'21, CVPR'21, ...)



**Take Home Message:** *LTH can find you a good mask on pre-trained models (supervised or self-supervised), in NLP, CV and even multi-modality, so the sparse subnetwork is the same transferrable!*

## MIT News

ON CAMPUS AND AROUND THE WORLD

SUBSCRIBE

### Shrinking massive neural networks used to model language

A new approach could lower computing costs and increase accessibility to state-of-the-art natural language processing.

Daniel Ackerman | MIT News Office  
December 1, 2020





The University of Texas at Austin  
**Electrical and Computer  
Engineering**  
*Cockrell School of Engineering*