UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

ANTS LAB

MEST: Accurate and Fast Memory-Economic Sparse Training Framework on the Edge

Geng Yuan[1], Xiaolong Ma[1], Wei Niu[2], Zhengang Li[1], Zhenglun Kong[1], Ning Liu[3], Yifan Gong[1], Zheng Zhan[1], Chaoyang He[4], Qing Jin[1], Siyue Wang[1], Minghai Qin, Bin Ren[2], Yanzhi Wang[1], Sijia Liu[5], Xue Lin[1]

[1]Northeastern University, [2]College of William and Mary, [3]Midea Group, [4]University of Southern California, [5]Michigan State University

NeurIPS 2021 Spotlight Paper

Presented by Rui Chen

Outline

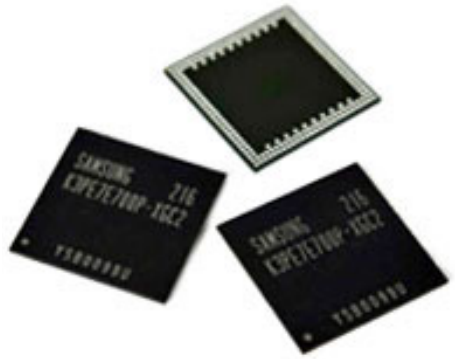
- Background & Motivation
- Memory-Economic Sparse Training
- Explore Data Efficiency in Sparse Training
- Experimental Results
- Conclusion

Background

- The tremendous storage and computing costs of DNNS
  - Most weight pruning works are aiming to accelerate **inference** stage.
  - Training a DNN model consumes **a lot more** time and resources.
  - Introduce sparsity into DNN training process, which can potentially **accelerate** the DNN training and reduce the hardware costs.
  - Critical to **edge devices** with limited resources.

	Computations	Time consumption
Inference*	~e9 FLOPs	In milliseconds
Training*	~e18 FLOPs	In hours/days

Samsung Mobile
RAM (2 GB)



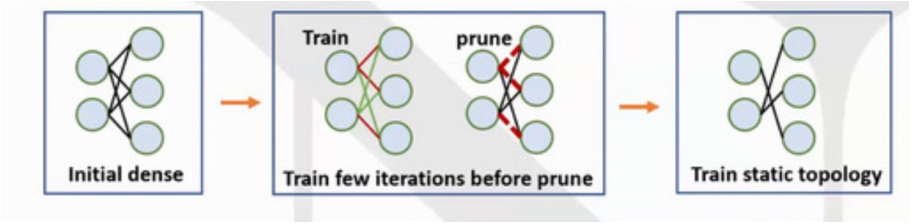
* Using ResNet-50 on ImageNet-1k as example

Background: way to accelerate training with sparsity

- Static Sparse Training: Find sparse mask at early training stage

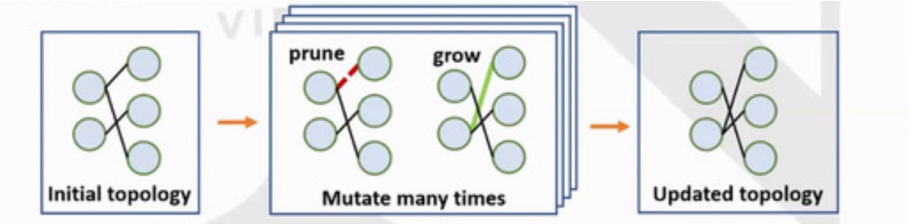
SNIP[2], GraSP[3]

  - Start with a **dense** model
  - Train few iterations to find sparse mask.
  - Train **static sparse** model.



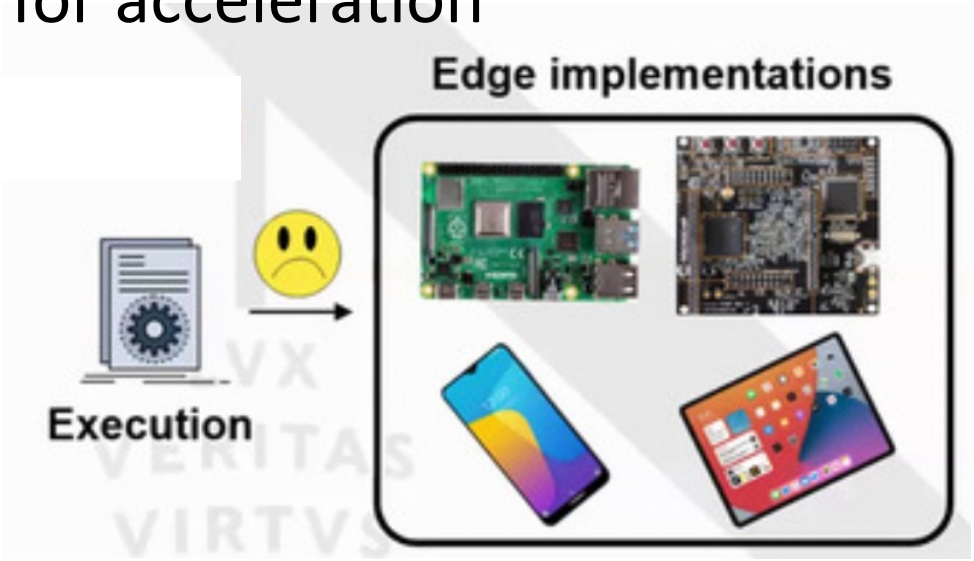
- Dynamic Sparse Training algorithms: SET[4], DeepR[5], DSR[6]

  - Start with random sparse mask
  - Dynamically prune and grow back weights
  - **Maintain sparse** model throughout training



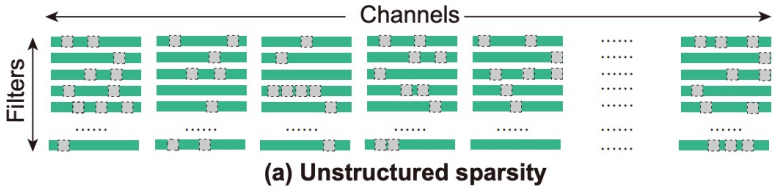
Motivation

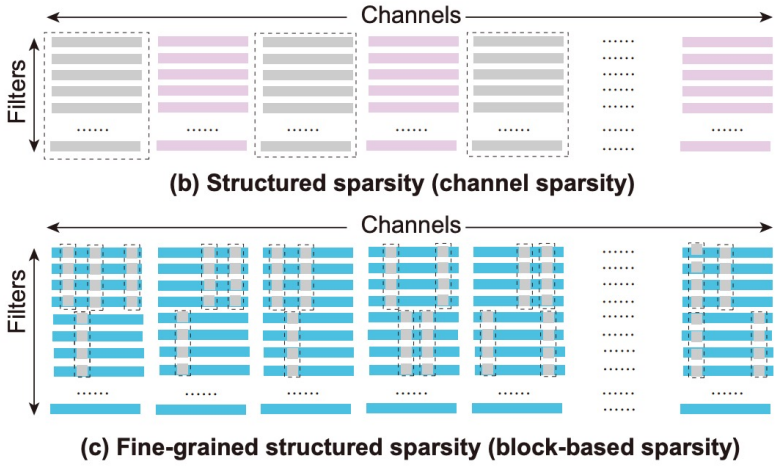
- FLOPs are NOT the only criteria for acceleration
  - **Memory** bandwidth, peak **memory** consumption, **memory** access are all matters.
  - Many factors affect the actual acceleration due to **hardware parallelism.**
- Unstructured sparsity is not conducive for acceleration
  - Poor data locality.
  - Load balance issue.
  - Heavy control-flow instructions.

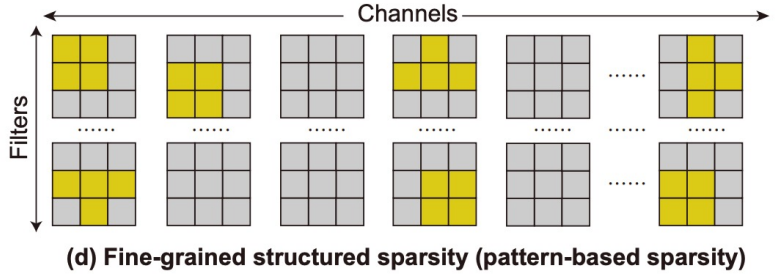


Motivation

- Four types of sparsity schemes

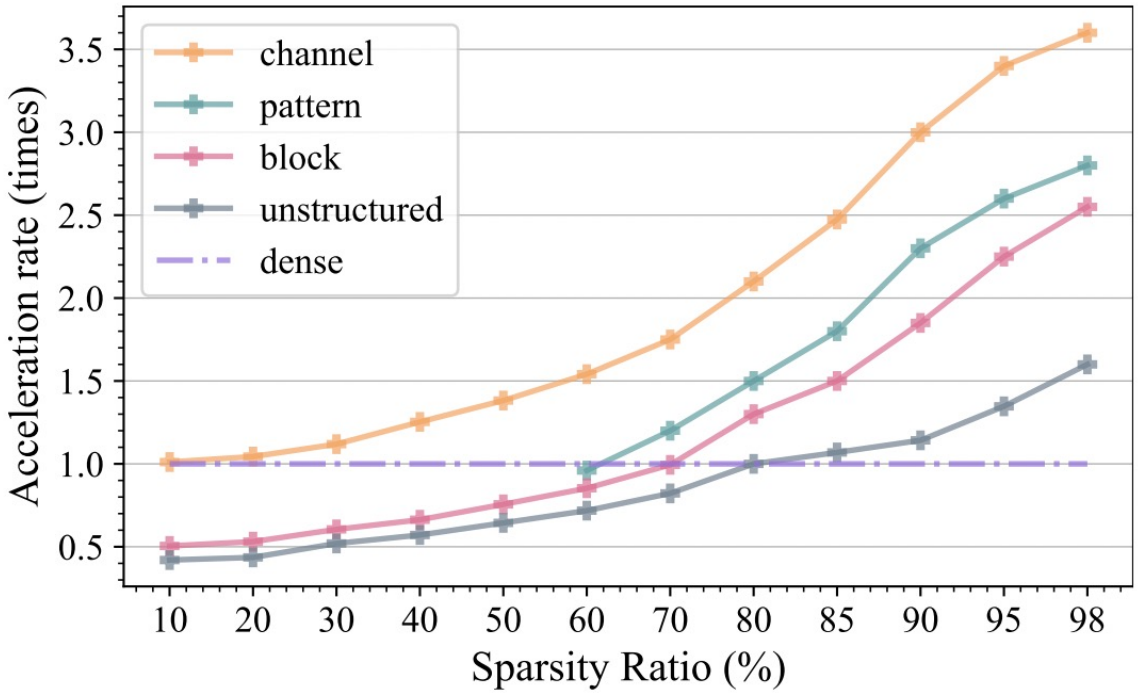






Motivation

- Training speed under various sparsity schemes



CONV layer on ResNet-32

- The results here are measured with a Samsung Galaxy S20 smartphone.
- Block and unstructured sparsity schemes cannot achieve any acceleration when sparsity ratio is below 70% and 80%

Motivation

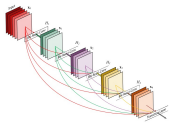
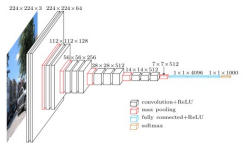
- Not memory-economic for end-to-end training on edge devices
  - Requires **dense model training** to find desired sparse mask.
  - Requires frequent access to **dense gradients** and compute **dense backpropagation.**

Samsung Galaxy A02s

Processor
Octa-core

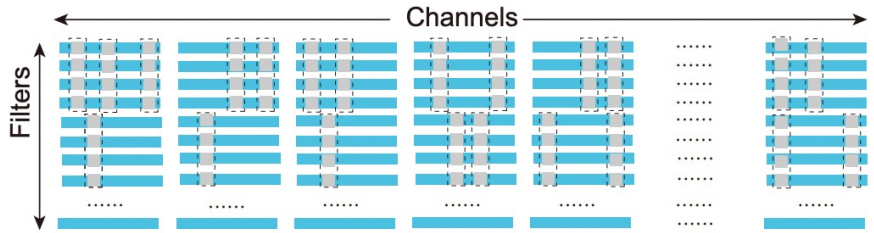
RAM
3GB / 4GB

Storage
32GB / 64GB + Up to 1TB(Micro SD card)

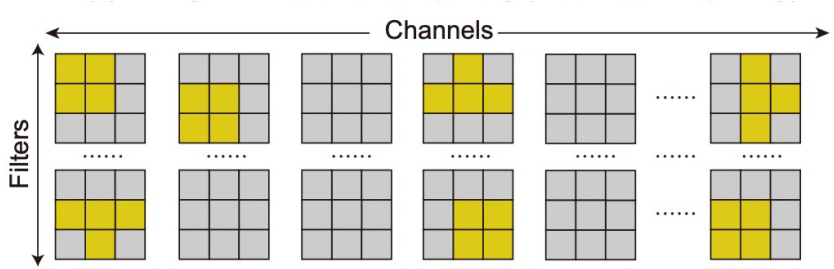
	Dense Model Size	Our Sparse Model Size
ResNet-32 	462MB	23MB
VGG-19 	4964MB	247MB

Motivation: MEST Design

- Hardware-friendly sparsity



(c) Fine-grained structured sparsity (block-based sparsity)



(d) Fine-grained structured sparsity (pattern-based sparsity)

- Memory-economic computation
  - Not involve any dense information
  - Keep entire training sparse
- Explore data efficiency

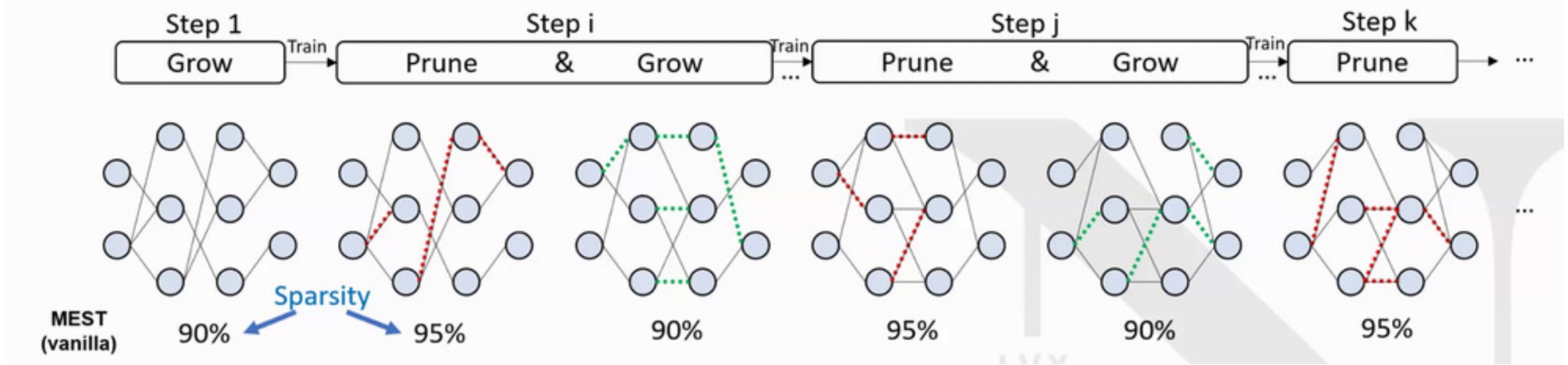
Easy examples	Easy to learn	Inessential
Hard examples	Hard to learn	Important

Outline

- Background & Motivation
- Memory-Economic Sparse Training
- Explore Data Efficiency in Sparse Training
- Experimental Results
- Conclusion

Sparse Training with MEST

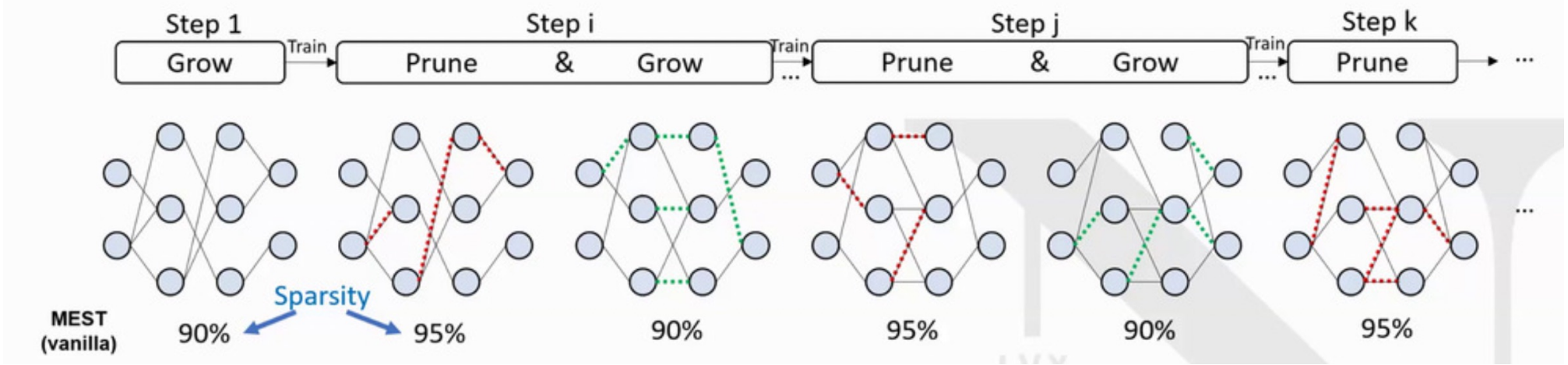
- End-to-end memory-economic sparse training



- Periodically remove less important non-zero weights from the sparse model and grow zero weights back during the sparse training process.

Sparse Training with MEST

- End-to-end memory-economic sparse training



- Prune:

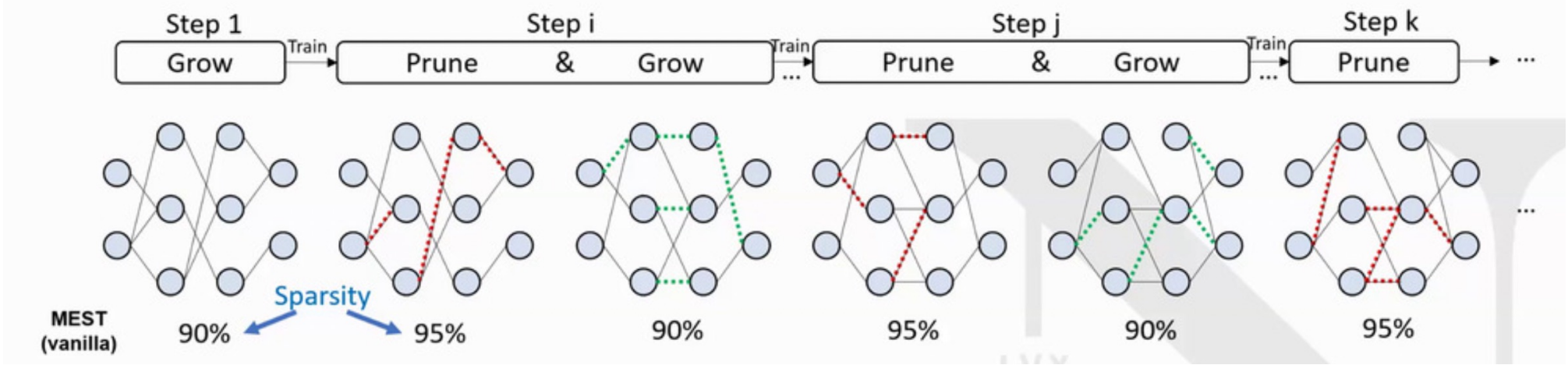
$$Scr_{w_\tau} = |w_\tau| + |\lambda \frac{\partial \ell(W_{\tau-1}, D)}{\partial w_{\tau-1}}|$$

Magnitude of weight

Magnitude of gradients

Sparse Training with MEST

- End-to-end memory-economic sparse training

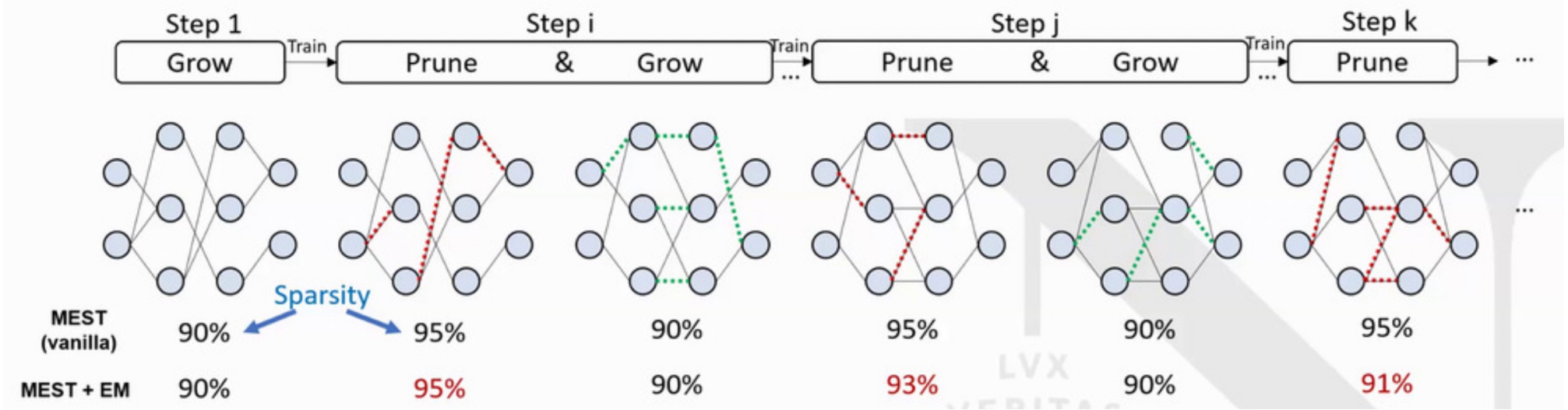


- Grow: randomly select weights to grow back

Do not require dense model information

Sparse Training with MEST

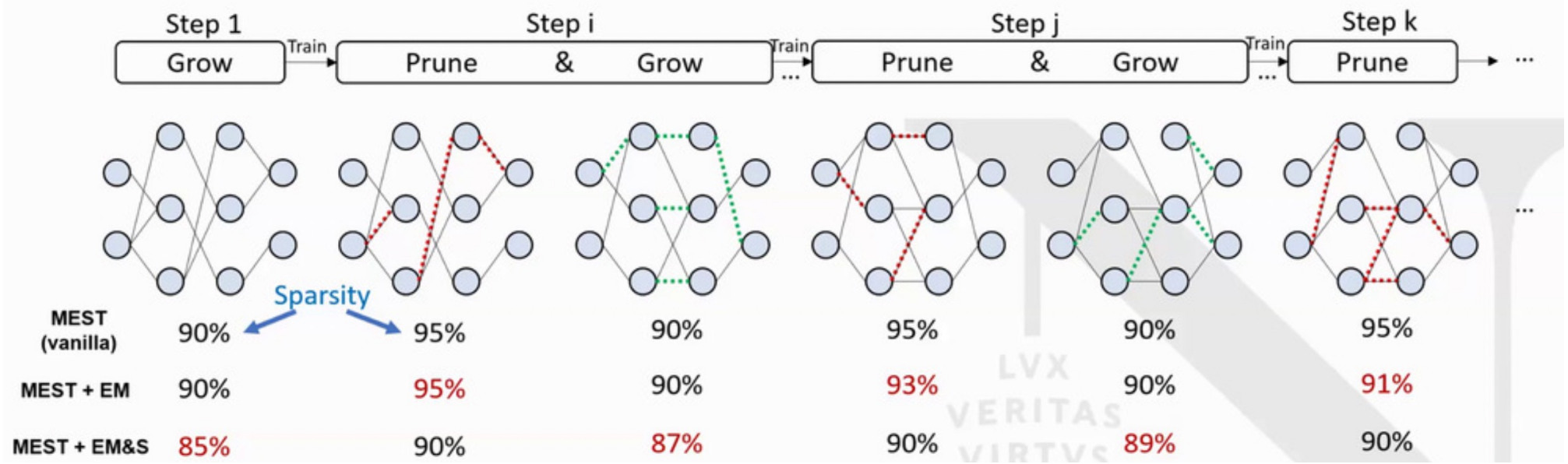
- Elastic Mutation: Gradually reduce the mutation ratio in *Prune phase*



- Maintains a sufficient search space while making the sparse model smoothly stabilize to an optimized structure.

Sparse Training with MEST

- Soft Memory Bound Elastic Mutation: Gradually increase the mutation ratio in *Grow phase*



- Avoid forcing the existing weights in the model to be removed if they are more important than newly grown weights

Sparse Training with MEST

Algorithm 1: MEST with (Soft) Elastic Mutation

Input: Network with uninitialized weight W in a total of L layers, target sparsity ratio s, p, τ, $\Delta\tau$, τ_{stop}.

Output: A sparse model satisfying the target sparsity requirement.

Initialize W with random values and random sparse mask according to the sparsity requirements.

while $\tau < \tau_{stop}$ **do**

  if *MEST+EM* **then**

    if $(\tau \mod \Delta\tau) = 0$ ▷ do weight mutation **then**

      Decay p if τ reaches a decaying milestone.

      for *each layer weight tensor* W^l **do**

        $W^l \leftarrow \texttt{ArgRemoveTo}(W^l, s+p)$

        $W^l \leftarrow \texttt{ArgGrowTo}(W^l, s)$

  if *MEST+EM&S* **then**

    Decay p if τ reaches a decaying milestone.

    for *each layer weight tensor* W^l **do**

      $W^l \leftarrow \texttt{ArgGrowTo}(W^l, s-p)$

    Training for $\Delta\tau$ epochs; ▷ $\tau \leftarrow \tau + \Delta\tau$

    for *each layer weight tensor* W^l **do**

      $W^l \leftarrow \texttt{ArgRemoveTo}(W^l, s)$

Continue sparse training from the epoch τ_{stop} to τ_{end}.

EM: Gradually reduce the mutation ratio in Prune phase

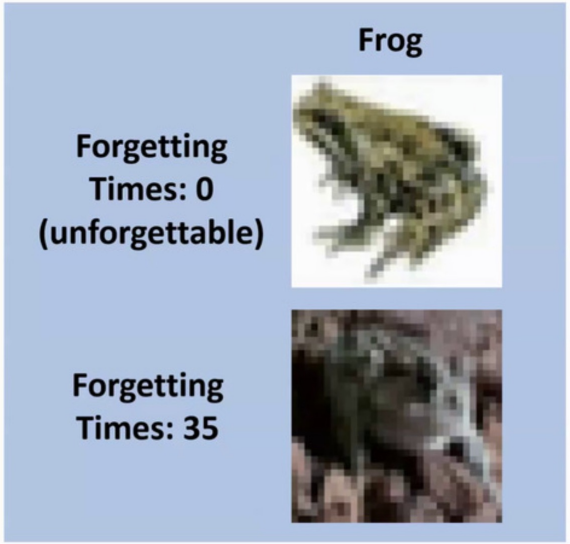
EM&S: Adding an 'undo' mechanism to the mutation process

Outline

- Background & Motivation
- Memory-Economic Sparse Training
- Explore Data Efficiency in Sparse Training
- Experimental Results
- Conclusion

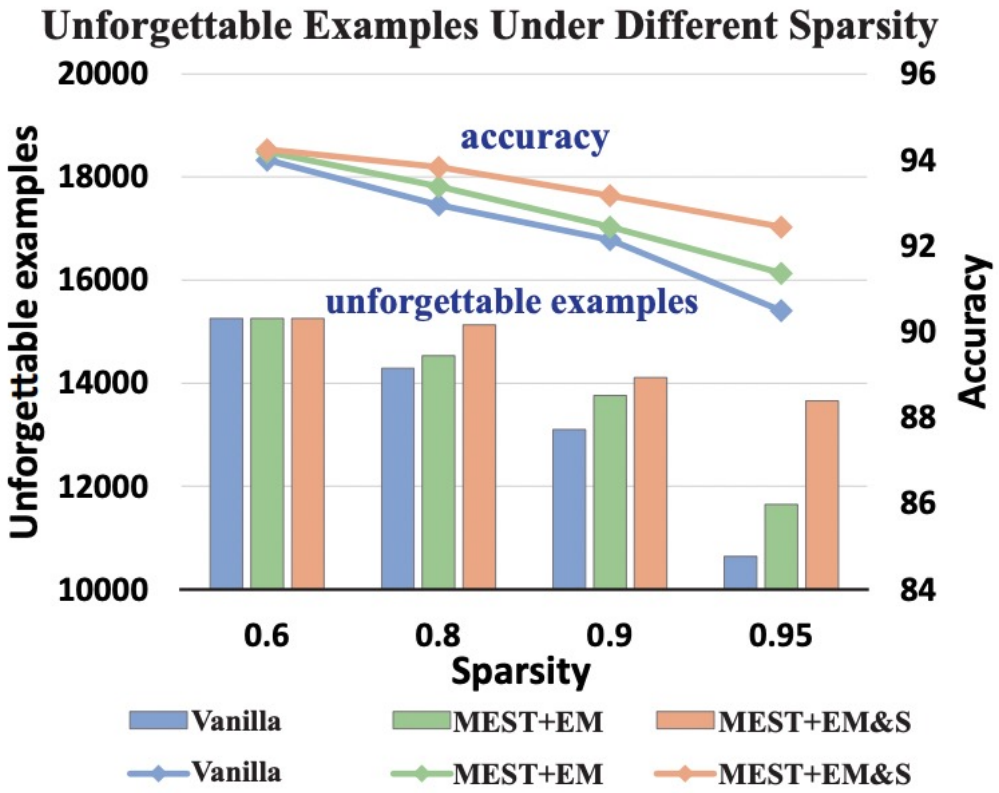
Notion [23]

- **Forgetting event:** an individual training example transitions from being classified correctly to incorrectly over the training process.
- **Unforgettable example:** an example will never be misclassified after it has been correctly classified, which is easy to learn



- We can further accelerate sparse training by removing unforgettable examples.

Impact of Model Sparsity on Dataset Efficiency



- The unforgettable examples exist in sparse training.
- More sparsity, less unforgettable examples.
- Better sparse training algorithm, more unforgettable training examples.

Higher acceleration!!

Identifying unforgettable examples

Phase 1

Phase 2

Training start → Training with full dataset → ... → Training with reduced dataset → ... → Training end

Collecting statistics while training

use

use

Forgetting times:
Img1: 12158,
Img2: 65778,
Img3: 3855,
Img4: 32109,
Img4: 2377,
Img5: 998,
Img6: 3211,
Img7: 33418,
...
ImgN: 12759.

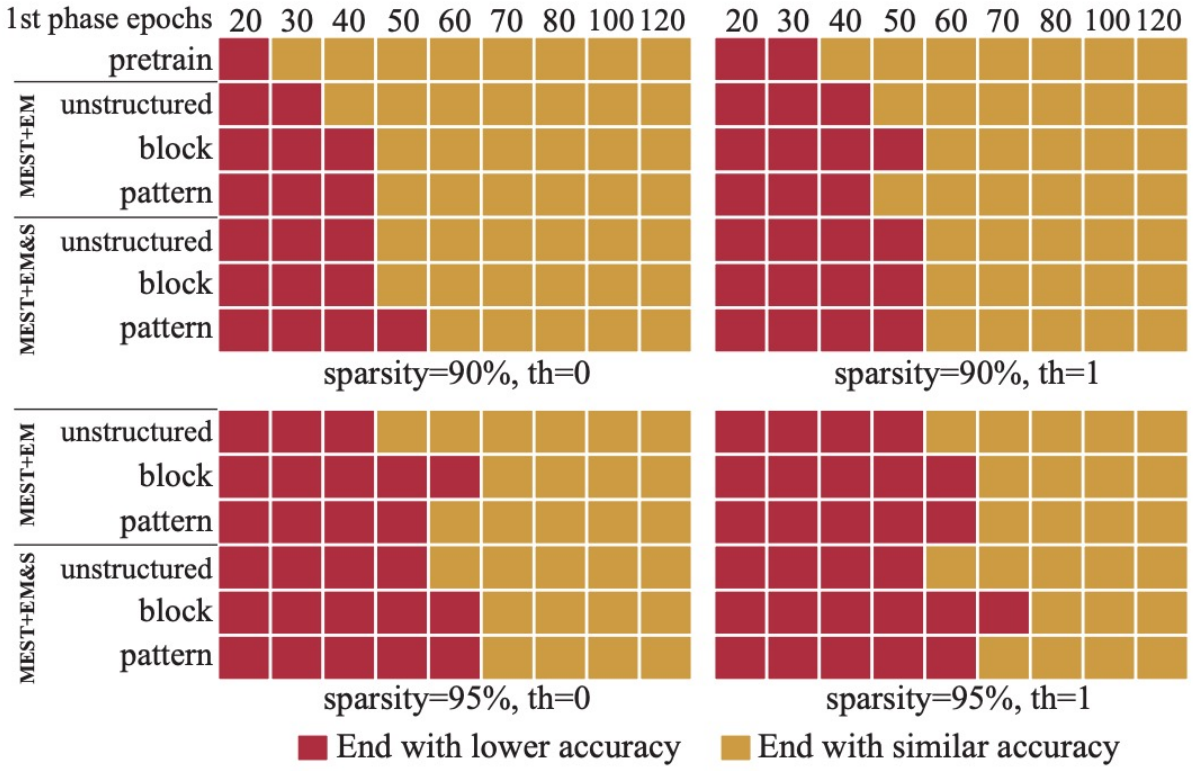
Less data
Fewer iterations
Faster training

Low forgetting times
High forgetting times

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Identifying unforgettable examples



- Models with higher sparsity ratios take longer to identify a good set of unforgettable examples.
- The larger number of examples to be removed, the more training epochs are required for the first training phase.

Outline

- Background & Motivation
- Memory-Economic Sparse Training
- Explore Data Efficiency in Sparse Training
- Experimental Results
- Conclusion

Experimental Setup

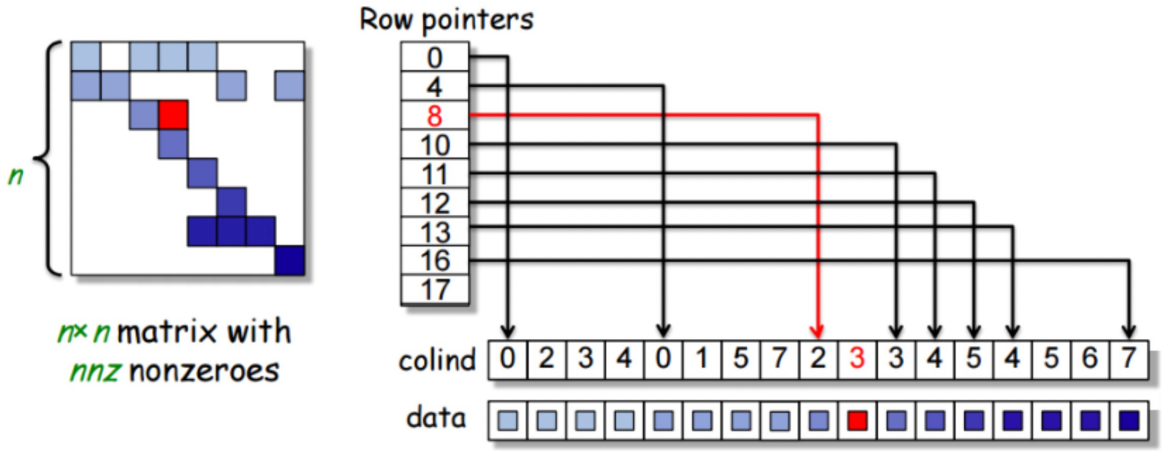
- Dataset: CIFAR-10/100, ImageNet-2012
- Models: ResNet-32 and VGG-19

- Device: A Samsung Galaxy S20 smartphone with Qualcomm Adreno 650 mobile GPU.

- The best-suited mutation interval is around 5 epochs on CIFAR10/100 and 2 epochs on ImageNet.

Experimental Setup

- Memory economy: Compressed Sparse Column (CSC)



- Training Speedup: Compiler Optimization
  - Matrix Reorder: group the rows (filters) in the weight matrix that have similar computation patterns together
  - Parameter Auto-tuning: search the execution-related and performance-critical tunable parameters in an automatic manner

Evaluation: Unstructured sparsity

Table 1: Accuracy comparison with SOTA works using ResNet-32 on CIFAR-10 and CIFAR-100.

	Dataset	Memory Footprint	CIFAR-10 (*Dense: 94.88*)			CIFAR-100 (*Dense: 74.94*)		
	Sparsity ratio		90%	95%	98%	90%	95%	98%
Lottery Ticket	LT [49]	dense	92.31	91.06	88.78	68.99	65.02	57.37
	SNIP [9]	dense	92.59	91.01	87.51	68.89	65.22	54.81
	GraSP [10]	dense	92.38	91.39	88.81	69.24	66.50	58.43
Dynamic sparse training	DeepR [15]	sparse	91.62	89.84	86.45	66.78	63.90	58.47
	SET [14]	sparse	92.30	90.76	88.29	69.66	67.41	62.25
	DSR [13]	sparse	92.97	91.61	88.46	69.63	68.20	61.24
	MEST (vanilla)	sparse	92.12±0.13	90.86±0.11	88.78±0.26	69.35±0.36	67.85±0.23	62.58±0.31
	MEST+EM	sparse	92.56±0.07	91.15±0.29	**89.22±0.11**	**70.44±0.26**	**68.43±0.32**	**64.59±0.27**
	MEST+EM&S	sparse	**93.27±0.14**	**92.44±0.13**	**90.51±0.11**	**71.30±0.31**	**70.36±0.05**	**67.16±0.25**

95% sparsity

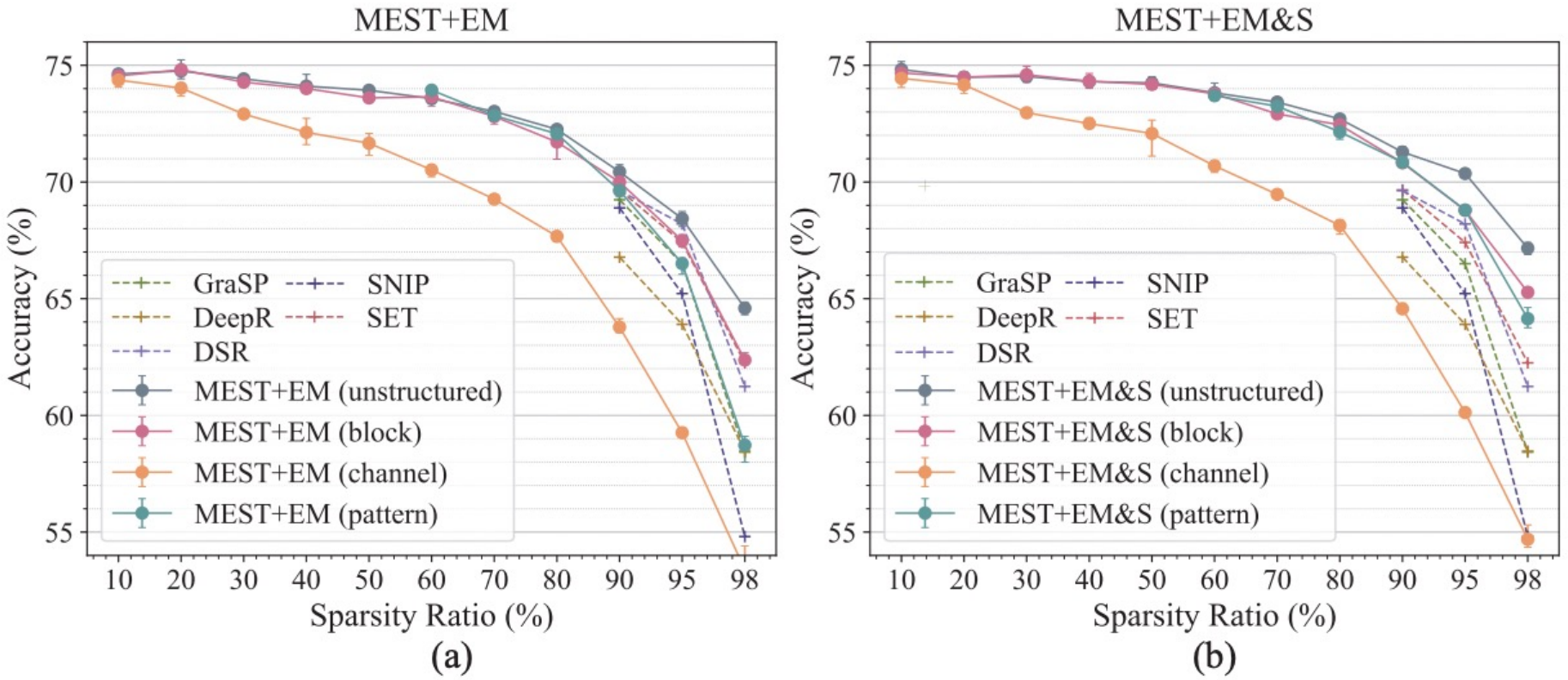
462MB

23MB

46MB

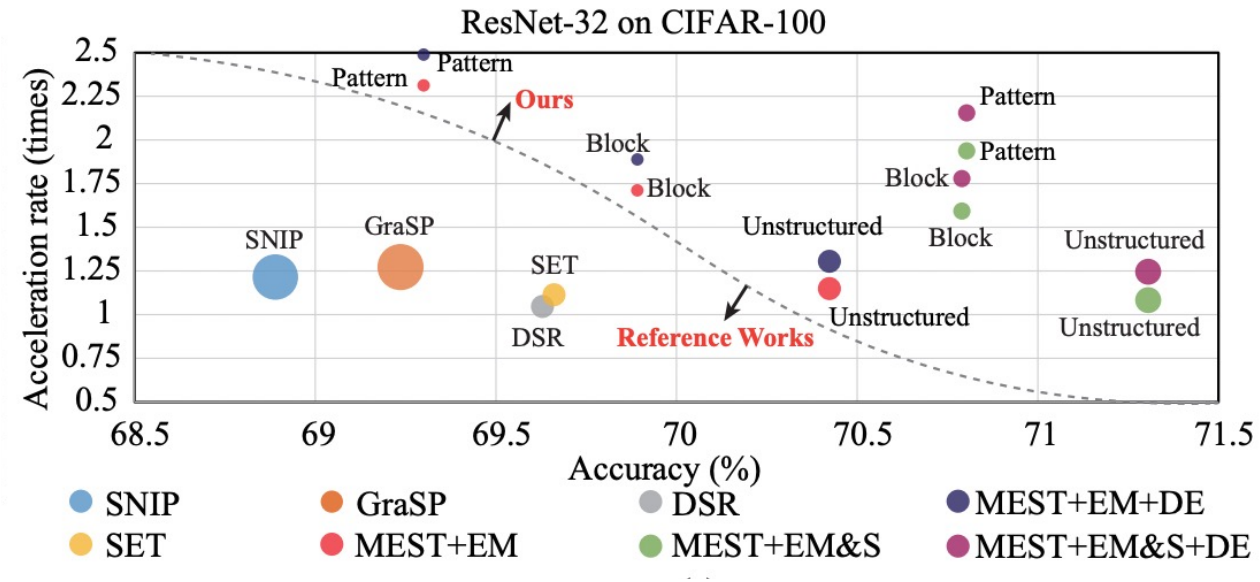
- The proposed MEST+EM (MEST+EM&S) keeps the entire training process sparse, and do NOT involve any dense information.

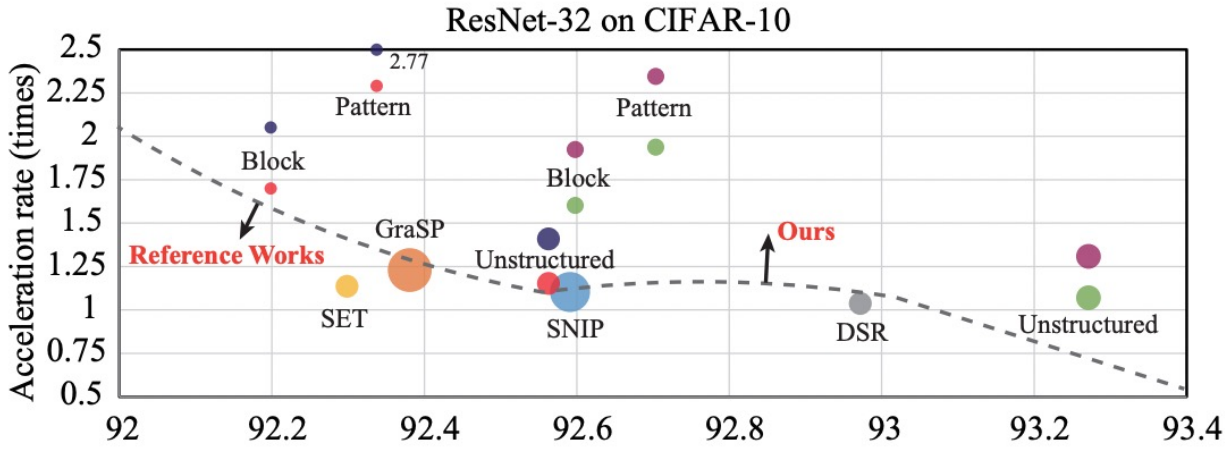
Evaluation: ResNet-32 with CIFAR-100



- With our MEST+EM&S, the accuracy of most sparsity schemes are boosted and outperforms the reference works.

Evaluation: Training acceleration & Memory footprints





- Unstructured sparsity can only achieve minor training acceleration.
- Our data-efficient training approach can effectively provide an extra speedup to all our results.

Evaluation: Layer-wise sparsity

ResNet-50 and CIFAR-100 dataset

- Hybrid: 1) pattern-based sparsity on 3x3 CONV layers and 2) block-based sparsity on 1x1 CONV layers
- Different size of CONV layers assign different sparsity ratios.
- All the schemes have the same sparsity ratio (90%).

Scheme	Sparsity Ratio	Accuracy (%)	Training Speed ($s/iter$)
Dense	0%	77.18	11.92
MEST+EM			
Pattern	44% (90%)	75.36	8.18
Block	90%	72.82	6.25
Hybrid (uniform)	90%	72.87	5.39
Hybrid (1.12:1)	90%	73.24	5.55
Hybrid (proportional)	90%	73.56	5.62
MEST+EM&S			
Pattern	44% (90%)	75.88	8.36
Block	90%	73.68	6.79
Hybrid (uniform)	90%	73.72	5.99
Hybrid (1.12:1)	90%	73.98	6.08
Hybrid (proportional)	90%	74.12	6.15

- The uniform sparsity ratio is the fastest.
- Using non-uniform sparsity ratios, the accuracy can be improved.

Combination of dataset compression and model sparsity

ResNet-32 and CIFAR-10 dataset

- using MEST+EM&S with unstructured sparsity
- ① further increasing the sparsity or ② incorporating data-efficient training

Scheme	baseline	①	②
Sparsity	90%	95%	90%
Removed examples	0	0	17900
Phase-1 epochs	-	-	40
Final accuracy (%)	93.27	92.44	93.02

- Incorporating data-efficient training instead of further increasing the model sparsity can deliver higher accuracy.

Conclusion

- Propose a novel Memory-Economic Sparse Training framework with enhancements by (soft) Elastic Mutation for accurate and fast execution on the edge.
- Systematically investigates the sparse training problem with respect to the sparsity schemes
- Investigate and incorporate the data-efficient training in sparse training scenario to further boost the accelerations


THANK YOU
UNIVERSITY of HOUSTON | ENGINEERING