ICML International Conference On Machine Learning

Memory-Efficient LLMs Training with Dynamic Sparsity

From Stability to Practical Scaling

Qiao Xiao*, Boqian Wu*, Patrik Okanovic, Tomasz Sternal, Maurice van Keulen,
Elena Mocanu, Mykola Pechenizkiy, Decebal Constantin Mocanu, Torsten Hoefler

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF TWENTE.

UNIVERSITÉ DU LUXEMBOURG

ETH zürich

TU/e

Outline

01 **Background**
LLM memory, sparsity, and the DST.

02 **Motivation**
Why topology updates may affect stability in LLM pre-training.

03 **Methodology**
SMET: warm-up, density-aware LR scaling, and sparse states.

04 **Experiments**
Quality, stability, memory, sparsity level, and block-wise speed.

05 **Conclusion & future work**
Toward stable, scalable, and memory-efficient sparse LLM training.

Memory Is a Key Bottleneck in LLM Pre-training

ICML International Conference On Machine Learning

Weights

parameters

1×

Gradients

one gradient per trainable weight

1×

Adam optimizer

m and v buffers

2×

Activations

sequence × batch × layers

variable

TU/e

Memory-Efficient Training Methods

Family	Training memory	Final model	Core mechanism
LoRA / ReLoRA	lower trainable states	dense	adapter updates
GaLore	low-rank gradients	dense	update projection
Low-precision training	lower-precision tensors	dense	reduce precision
Sparse training	**sparse throughout**	**sparse**	**Sparse weights**

Some methods reduce training memory but end with dense models. Sparse training can preserve efficiency opportunities after training.

From Dense Training to Sparse Training

Dense training

Train all parameters

→ **High memory cost**

Static sparse training

Fixed sparse topology

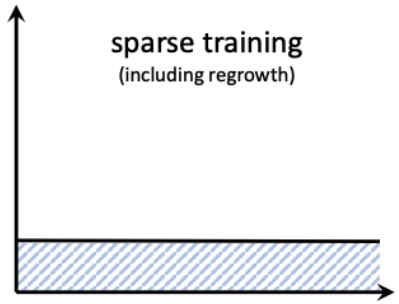
→ **Limited flexibility**

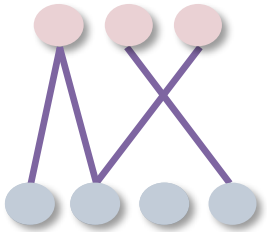
Dynamic sparse training

Adaptive sparse topology

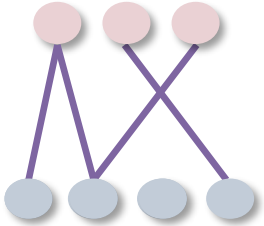
→ **Better allocation**

How Does Dynamic Sparse Training Work?

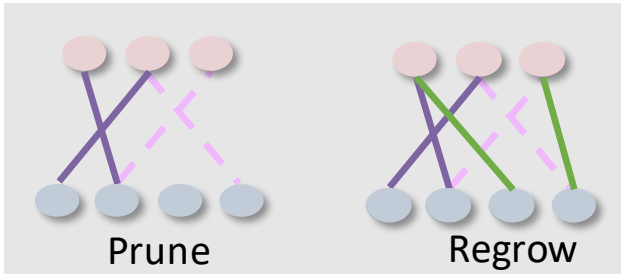


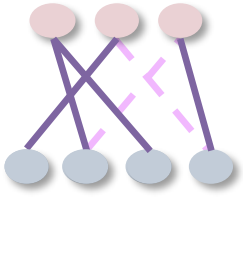


Static Sparse Training



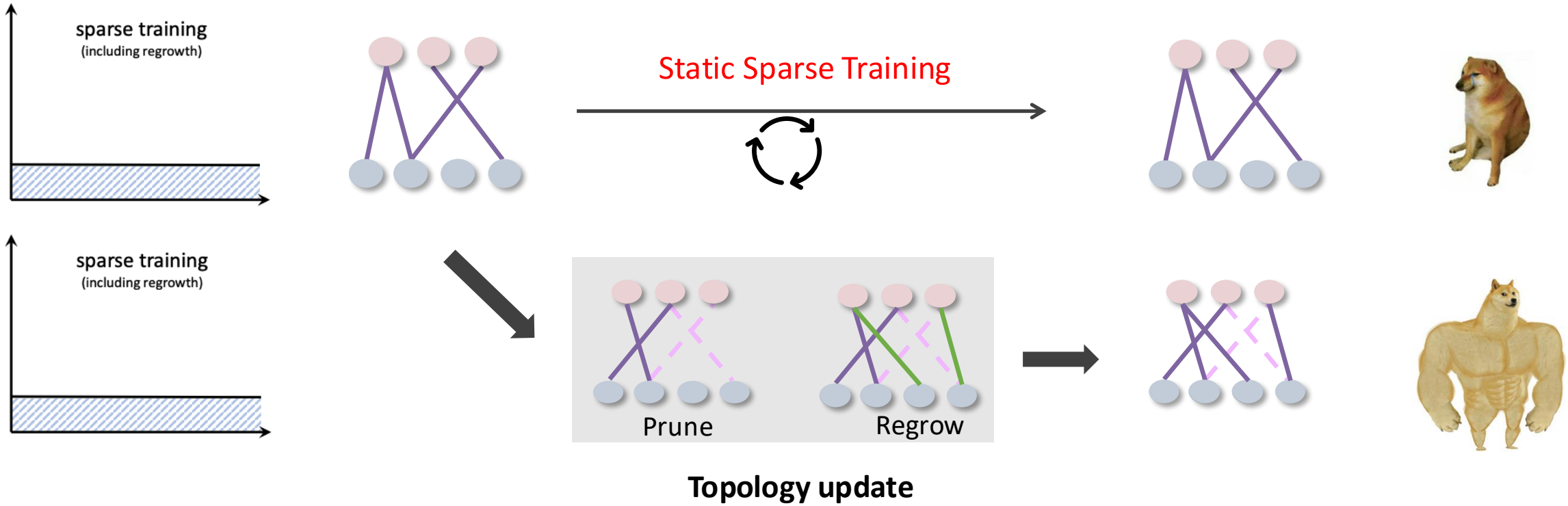
Dynamic Sparse Training





Topology update

How Does Dynamic Sparse Training Work?



Can DST Scale to LLM Pre-training?

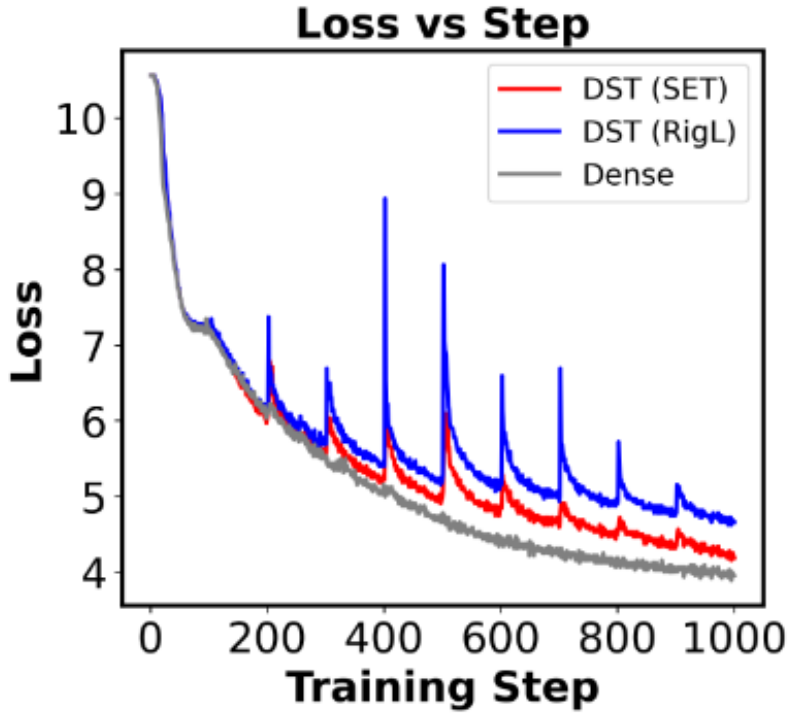
ICML International Conference On Machine Learning

1	**Quality**	Can sparse pre-training remain accuracy performance of dense training?
2	**Stability**	LLM pre-training is sensitive to optimization instability, while DST continuously changes network connectivity.
3	**Efficiency**	Can sparsity translate into real memory and compute savings?

Question **How can we make LLM pre-training more efficient with DST, without sacrificing quality or stability?**

TU/e

Observation1: topology updates trigger loss spikes

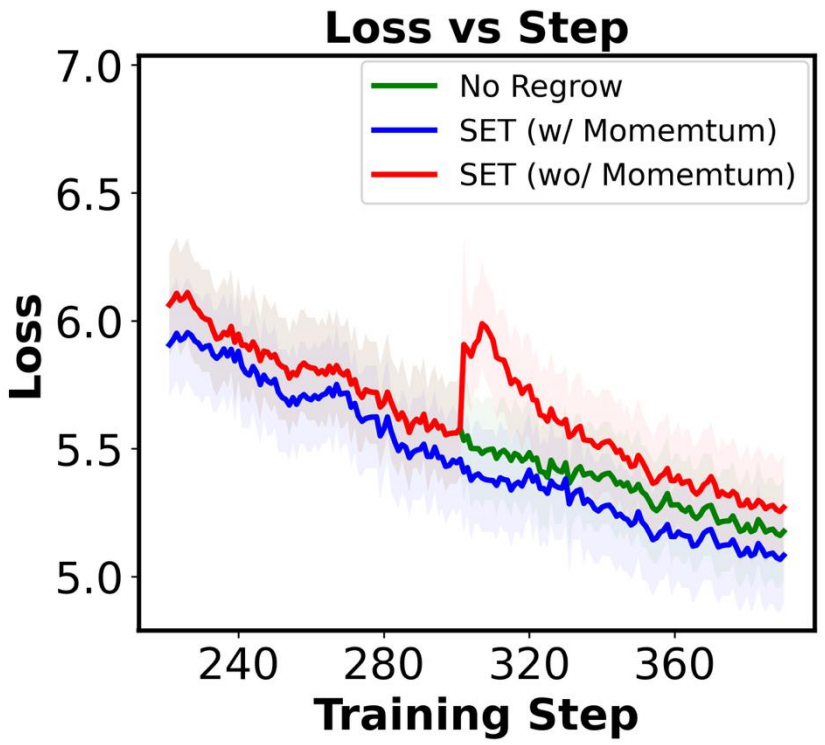


Loss comparison (↓) for pre-training LLaMA-240M on the C4 dataset (DST methods with a density of 0.25).

- After each topology update, SET and RigL show sharp transient loss increases.
- Dense training does not show this sawtooth behavior.

What part of the topology update causes these spikes?

Observation2: the spike is caused by regrowth process



Loss comparison (↓) for pre-training LLaMA-240M on the C4 dataset under a density of 0.25.

01 Pruning only

No loss spikes is observed without regrowth

02 Regrowth

Loss spikes appear after new connections are activated

03 State preservation

substantially reduces the spike

Observation3 : newly regrown weights cold-start under Adam

Adam update magnitude: $|\Delta\theta| = \alpha \cdot |\hat{m}_t| / (\sqrt{\hat{v}_t} + \varepsilon)$

Mature weights

After many steps, Adam has stable EMA estimates:

$$|\Delta\theta_{\text{normal}}| \approx \alpha \cdot \frac{|\mathbb{E}[g_t]|}{\sqrt{\mathbb{E}[g_t^2]} + \varepsilon} < \alpha$$

Effective updates are usually below the nominal learning-rate scale.

Newly regrown weights

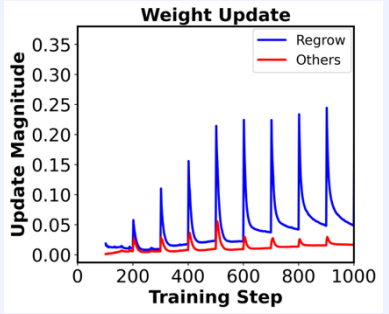
Regrowth in training: t ≫ 1, but moments are started as **m = 0, v = 0.**

$$|\Delta\theta_{\text{regrow}}| \approx \alpha \cdot \frac{1-\beta_1}{\sqrt{1-\beta_2}} > \alpha$$

The update magnitudes are larger than learning rate.

Cold-start effect

Regrown weights receive abrupt, and oversized updates.



Takeaway: newly regrown weights should not immediately behave like mature weights; they should enter optimization gradually.

SMET: Stable and Memory-Efficient Sparse LLM Pre-training

ICML International Conference On Machine Learning

01 **Optimizer-state warm-up**

newly regrown weights enter with s warm-up period.

02 **Density-aware LR scaling**

Compensate for reduced effective fan-in in sparse model

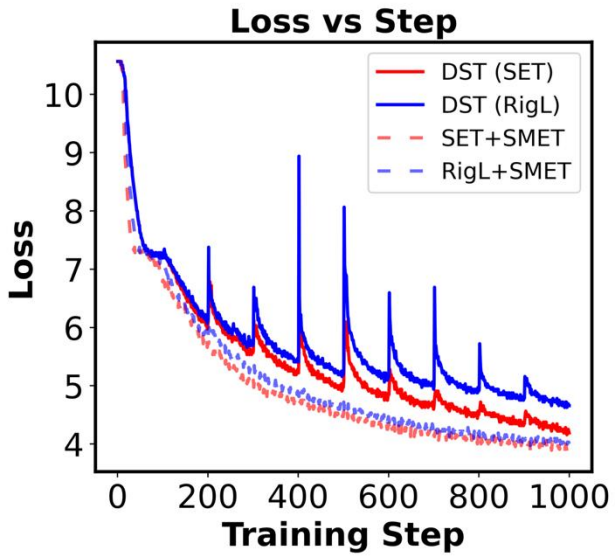
03 **Sparse training states**

gradients and Adam moments are stored only for active parameters.

TU/e

Part 1: Optimizer Warm-up for Regrown Weights

Newly activated weights should be optimized as new parameters, rather than inheriting the training age of mature weights.



Loss comparison (↓) for pre-training LLaMA-240M on the C4 dataset under a density of 0.25.

Warm up its early updates

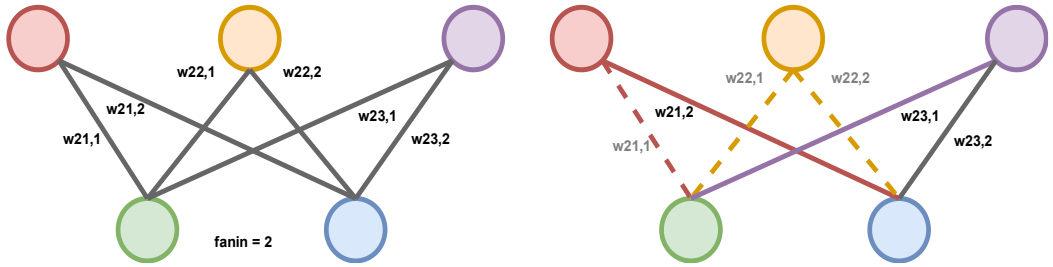
$$\alpha_i(\tau_i) = \alpha \cdot \min\left(\frac{\tau_i}{W}, 1\right)$$

$$|\Delta\theta| = \alpha \cdot |\hat{m}_t| / (\sqrt{\hat{v}_t} + \varepsilon)$$

A warm-up strategy makes regrown weights update gradually, reducing loss spikes.

Part 2: Density-Aware Learning Rate Scaling

Sparse layers reduce effective fan-in, so we compensate with density-aware learning-rate scale up.



Scale the learning rate by density

$$\alpha_s = \frac{\alpha_0}{\sqrt{d}}$$

01 Reduced fan-in

Sparse layers change effective parameterization.

02 Compensation

Warm-up stabilizes; LR scaling restores progress.

03 Complementary Effect

Stable training with more effective learning

Part 3: Sparse Training States for Memory Efficiency

Sparse Gradients

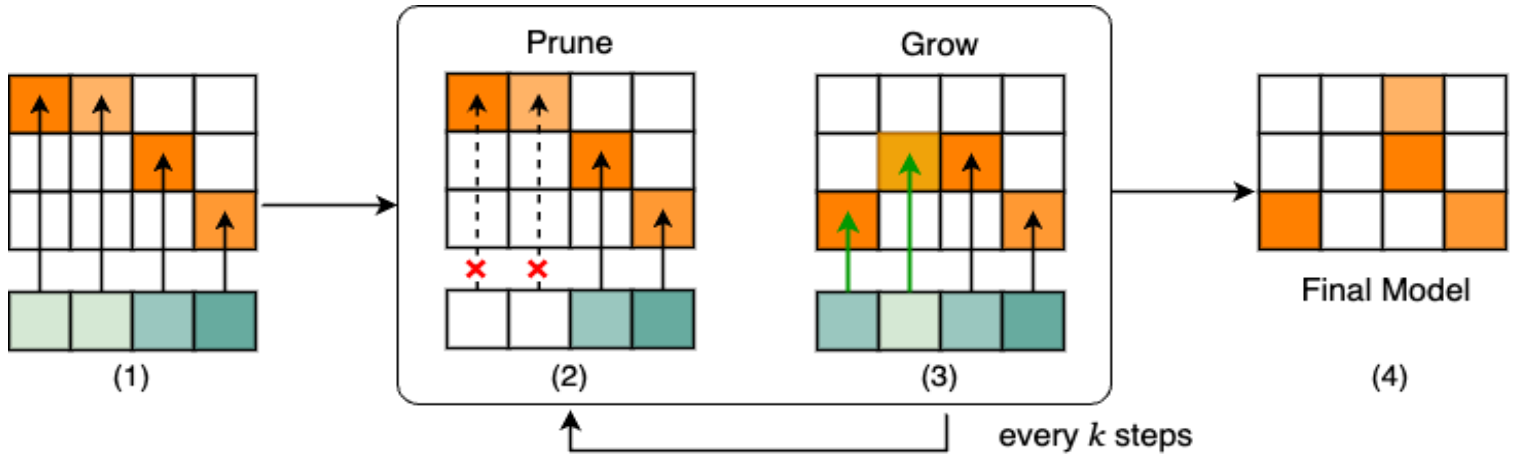
Only active weights retain gradients.

Sparse Adam States

Store m, v, only for active weights.

Block-wise Storage

One index represents a block of active weights, reducing metadata overhead.



SMET in Summary

1 Stability

Give regrown weights a local age

- Reset local timestep for newly regrown weights
- Apply a short optimizer-state warm-up
- Makes early updates gradual and stabilizes training

2 Effectiveness

Compensate sparse parameterization

- Sparse layers have reduced effective fan-in
- Dense-model learning rate becomes too conservative as density decreases
- Density-aware scaling restores learning progress

3 Memory Efficiency

Turn sparsity into memory savings

- Store gradients only at active weights
- Store Adam states only for active sparse blocks
- Use block-wise metadata to reduce overhead

Together, these three components make SMET stable, effective, and memory-efficient for sparse LLM pre-training.

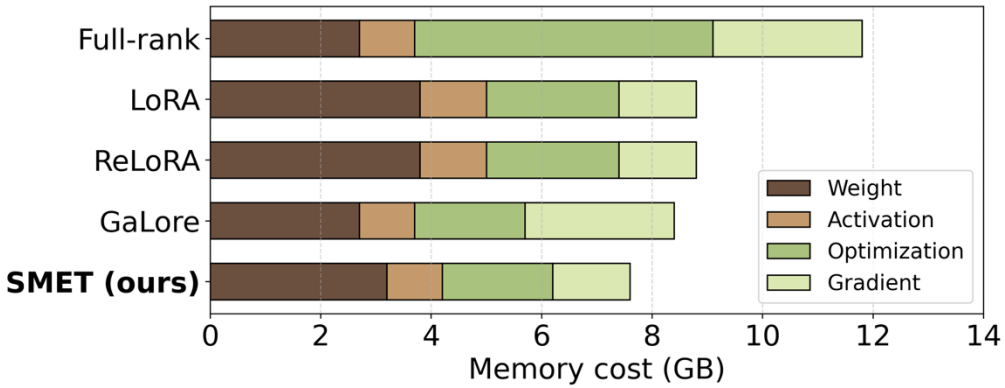
Overall Performance

- **Perplexity comparison**

Method	60M	130M	350M	1B
Dense	34.06	25.08	18.80	15.56
LoRA	34.99	33.92	25.58	19.21
ReLoRA	37.04	29.37	29.08	18.33
GaLore	34.88	25.36	18.95	15.64
SMET	**33.38**	**25.21**	**18.90**	**15.61**

Perplexity comparison (↓) for pre-training LLaMA-based models of various sizes on the C4 dataset. Comparison is from 60M to 1B, with training budgets range from 1.1B to 13.1B tokens. (SMET with a density of 0.4)

- **Memory consumption**

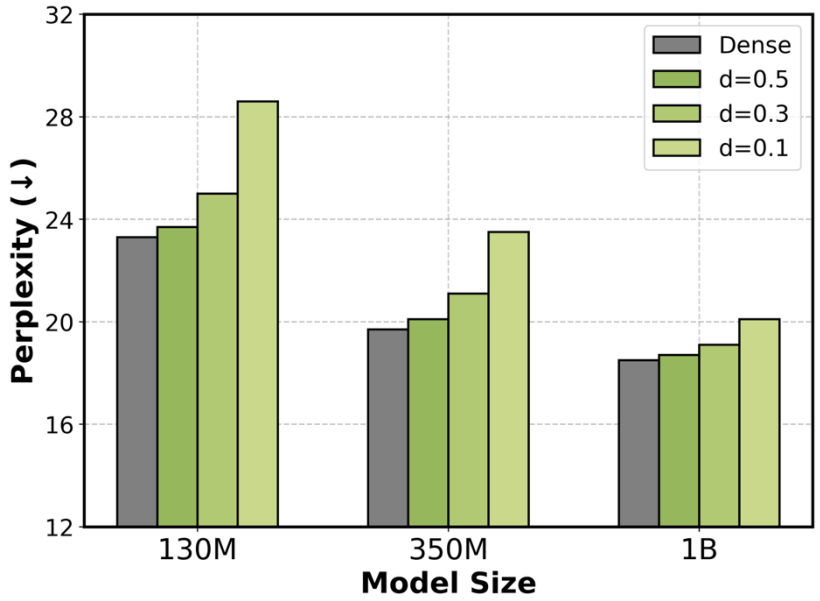


Estimated memory consumption for pre-training a LLaMA-1B model using different training methods in BF16.

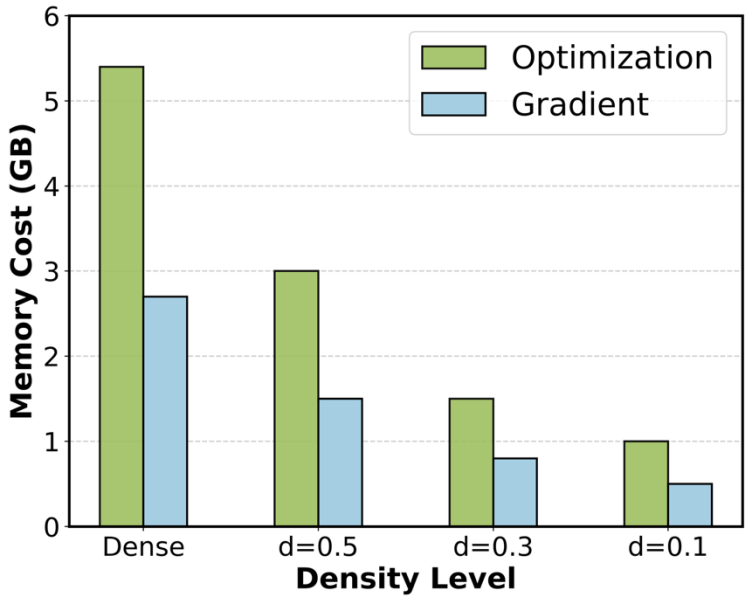
SMET stays close to dense training and beats memory-efficient baselines.

Effect on sparsity levels

- **Quality under different densities**

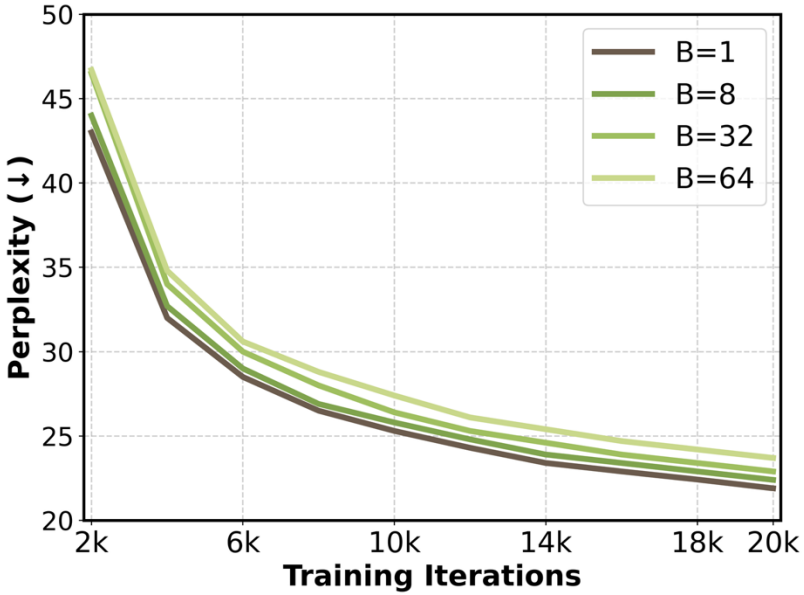


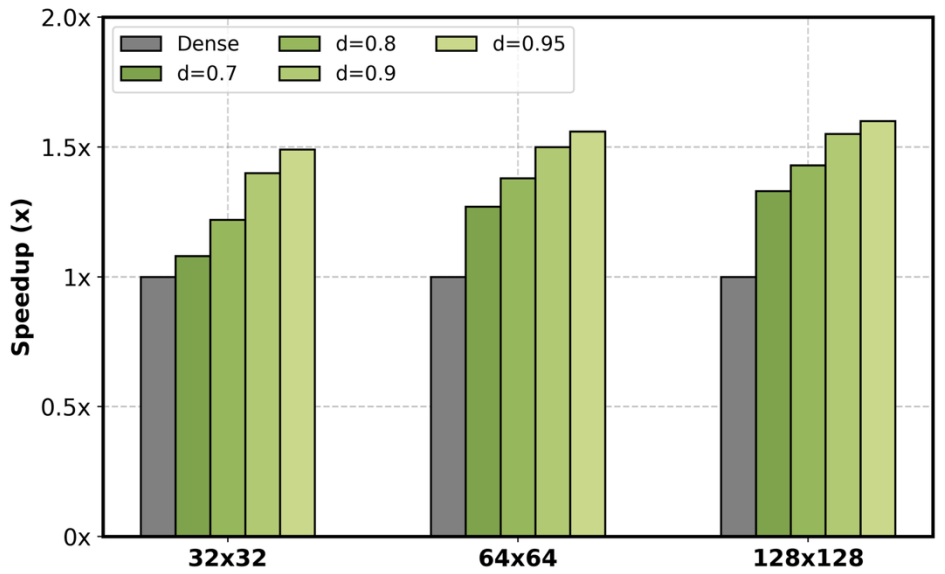
- **Training-state memory at 1B model**



As model size increases, sparse pre-training achieves a better quality–memory trade-off.

Block-wise sparsity: a bridge toward hardware-friendly deployment





SMET supports block-wise sparsity and can enable inference speedups with sparse kernels.

Ablation study

- **Regrowth Strategies**

Method	d=0.5	d=0.3	d=0.2	d=0.1
Static Sparse	22.90	23.55	25.40	27.44
RigL	25.75	27.83	28.85	29.90
$SMET_{grad}$	22.52	24.01	24.78	26.50
SET	24.71	26.38	27.87	28.96
SMET	**21.61**	**22.93**	**23.93**	**25.85**

- **Warm-up and LR scaling**

Config	d=0.5	d=0.3	d=0.2	d=0.1
Adam	24.71	26.38	27.87	28.98
+ Warm-up	22.52	24.08	25.59	27.67
+ LR Scaling	24.79	25.05	26.53	28.20
SMET	**21.61**	**22.93**	**23.93**	**25.85**

Conclusion

1. Diagnose

DST causes recurring loss spikes in LLM pre-training.

2. Design

SMET makes optimizer updates work with changing connections

3. Improve

SMET enables more stable and stronger sparse pre-training.

SMET turns dynamic sparsity into a practical solution for stable LLM pre-training.

Future Work

Activation + KV efficiency

Extend the sparse-memory idea beyond parameters to activations and KV cache.

Extremely high sparsity

Explore on extremely high sparsity levels, such as more than 90% sparsity.

Beyond pre-training

Study instruction tuning, and RLHF under dynamic sparsity.

The long-term goal is not only lower memory, but end-to-end efficient LLM training and inference.

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Thank You!

q.xiao@tue.nl

Paper

Code

TU/e