Pixelated Butterfly: Fast ML with sparsity

Beidi Chan

NN Training bottlenecks

- → Exponentially ↑ Data volume
- → ↑ model size
- → ↑ resources

Popular direction: Sparsity

Not new
→ lots of history

Existing approaches
- → Pruning: Deep Comp
- → Approx matmul: Reformer, SLIDE, Mongoose, Scatterbrain

Hard to ↑ speed without ↓ accuracy

Challenge & Goals

Challenges
- → Dynamic sparsity can maintain accuracy but SLOW DOWN training time
  - → SOTA requires up to 5X more epochs
- → Unstructured sparsity is NOT Hardware efficient (Hooker et. al)
- → Sparse attention target one module & hence does not speed up all layers
  - → In many apps, MLP layer is the bottleneck

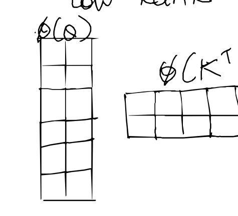
Ideal Sparsity pattern

- → Static, simple yet accurate
- → Aligned with available hardware
- → Applied to most NN layers

TLDR: Butterfly + Low rank is a simple & effective sparsity pattern

① Attention Approximation: Sparse or low-rank

Trade accuracy for efficiency



Not all cells have values
Reformer, Sparse Transformer

Use smaller matrix & only learn that

Linformer, Performer
↓
Founder started a startup.

Hard to find robust approx. that performs well on wide variety of tasks

Can we combine sparse & low rank?

YES!
Reformer (Sparse) + Performer (Low Rank)

$$SV + (\phi(Q)\,(\phi(K^T)V)) \approx \mathrm{softmax}(QK^T)V$$

Long-short Transformer
Scalable Optimal Transport

Butterfly matrices: Divide-and-conquer

Similar of FFT

Transform of size N
→ size N/2 → size N/4, size N/4
→ size N/2 → size N/4, size N/4

Class of fast linear maps that encodes the above pattern called butterfly

Each butterfly matrix is a product of log N sparse factors

Red dots indicates location of non zeros. The pattern is fixed & represents the recursion tree.

These matrices are trainable with grad descent

We can compose butterfly matrices to capture any sparse matrix with near optimal space & time complexity.

Butterfly + Low-rank can
- → avoid dynamic overhead (fixed pattern)
- → apply to most matmul-based layers

HOWEVER, IT IS NOT HARDWARE EFFICIENT

Pixelated butterfly

Issues	Pixelated Butterfly
→ Slow speed: Sparsity patterns are not block aligned → not friendly to hardware	Block Butterfly: Block-aligned sparsity pattern
→ Difficulty of parallelisation: Products of many ops → sequential ops	Flat butterfly: First order approx: Turn product to sum
→ Reduced expressiveness: Flat butterfly are necessarily high rank → cannot represent low rank matrices	Low rank term: ↑ expressiveness of Flat block Butterfly matrices

Combine Flat block butterfly + Low Rank

Workflow:
- → Give model schema & budget allocation
- → Pick a simple sparsity pattern based on flat block butterfly. Use this to produce mask on attention & also MLP layers

Theoretical properties

Theorem 1: Block butterfly retains the expressiveness of Butterfly & flat butterfly can approx residual form of butterfly

Theorem 2: FBB ↑ expressive than sparse or low rank alone

Theorem 3: Training wide & sparse NN converge locally

Applications

- → On CIFAR & Imagenet, train upto 2.3X faster & no accuracy loss
- → 2.5X faster than GPT-2 for LM tasks

↑ model size ⇒ ↑ savings using FBB

NTK suggests that FBB generalizes close to best model

Conclusion

- → Butterfly + low-rank works well
- → Pixelated butterfly → static & block sparsity direction

Future directions

Go beyond dense models i.e PDE solving, MRI reconstruction

Pixelfly-hardware co-design

Efficient sparsity/butterfly on next gen ML accelerators

Data sparsity:

Find structures in data & speed up training from a different angle.