outbrain

A Bag of Tricks for Scaling CPU-based Deep FFMs to more than 300m Predictions per Second (AdKDD 24')

Blaž Škrlj
Outbrain
bskrlj@outbrain.com

Benjamin Ben-Shalom
Outbrain
bbshalom@outbrain.com

Grega Gašperšič
Outbrain
ggaspersic@outbrain.com

Adi Schwartz
Outbrain
aschwartz@outbrain.com

Ramzi Hoseisi
Outbrain
rhoseisi@outbrain.com

Naama Ziporin
Outbrain
nziporin@outbrain.com

Davorin Kopič
Outbrain
dkopic@outbrain.com

Andraž Tori
Outbrain
atori@outbrain.com

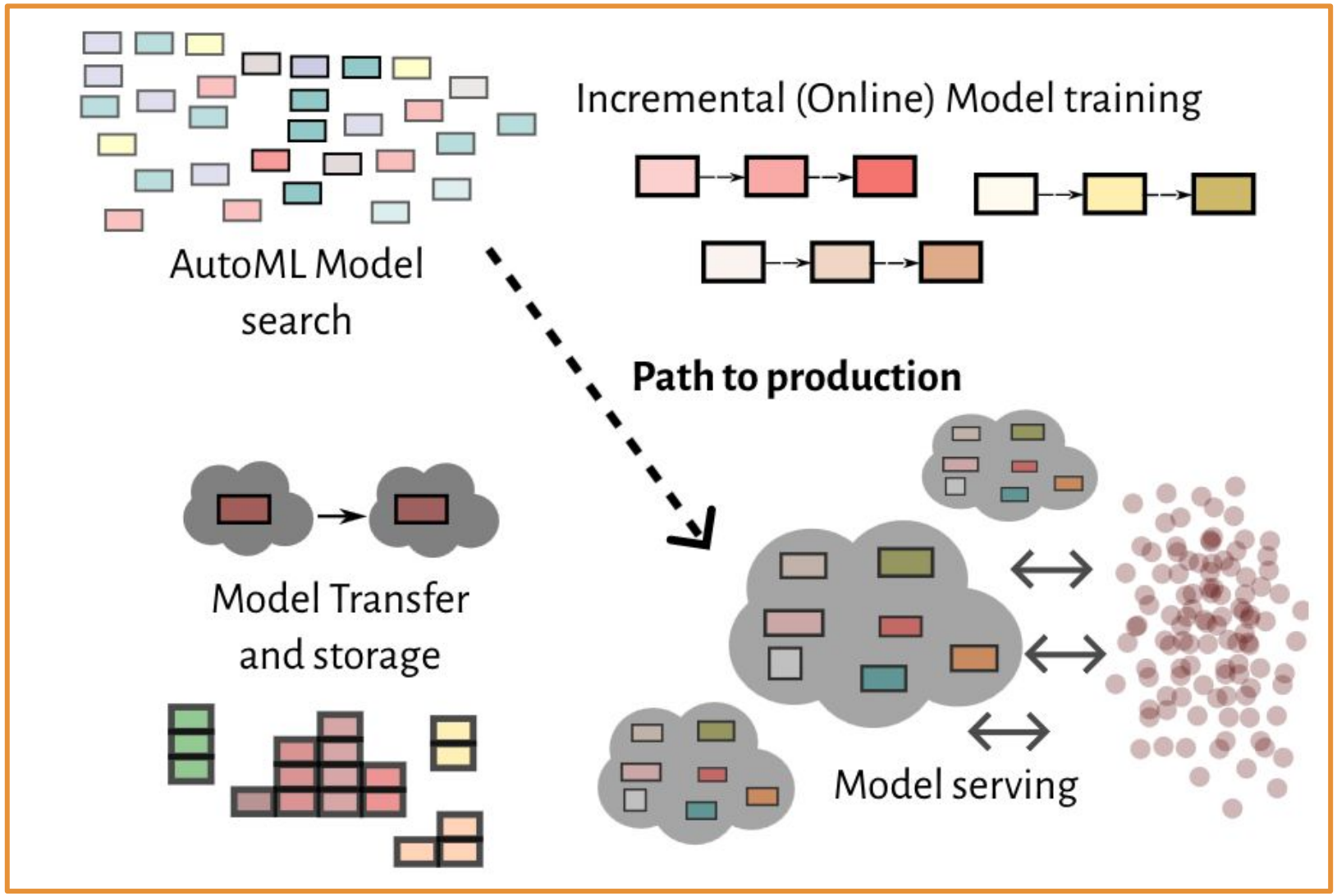
Highly-optimized single-instance learning frameworks

- Vowpal Wabbit
- **Fwumious Wabbit**

Mainstream frameworks (batch-based, **sophisticated architectures**)

- Torch
- Tensorflow
- Jax
- …

In practice: 300m+ predictions per second


Incremental (Online) Model training
AutoML Model search
Path to production
Model Transfer and storage
Model serving

A Bag of tricks

- Architecture of the **DeepFFMs** considered
- **Hogwild-based** training
- **Sparse** gradient updates
- Placement of this model in broader (**serving**) context
- **Speeding** up serving
- **Quantization** and weight transfers

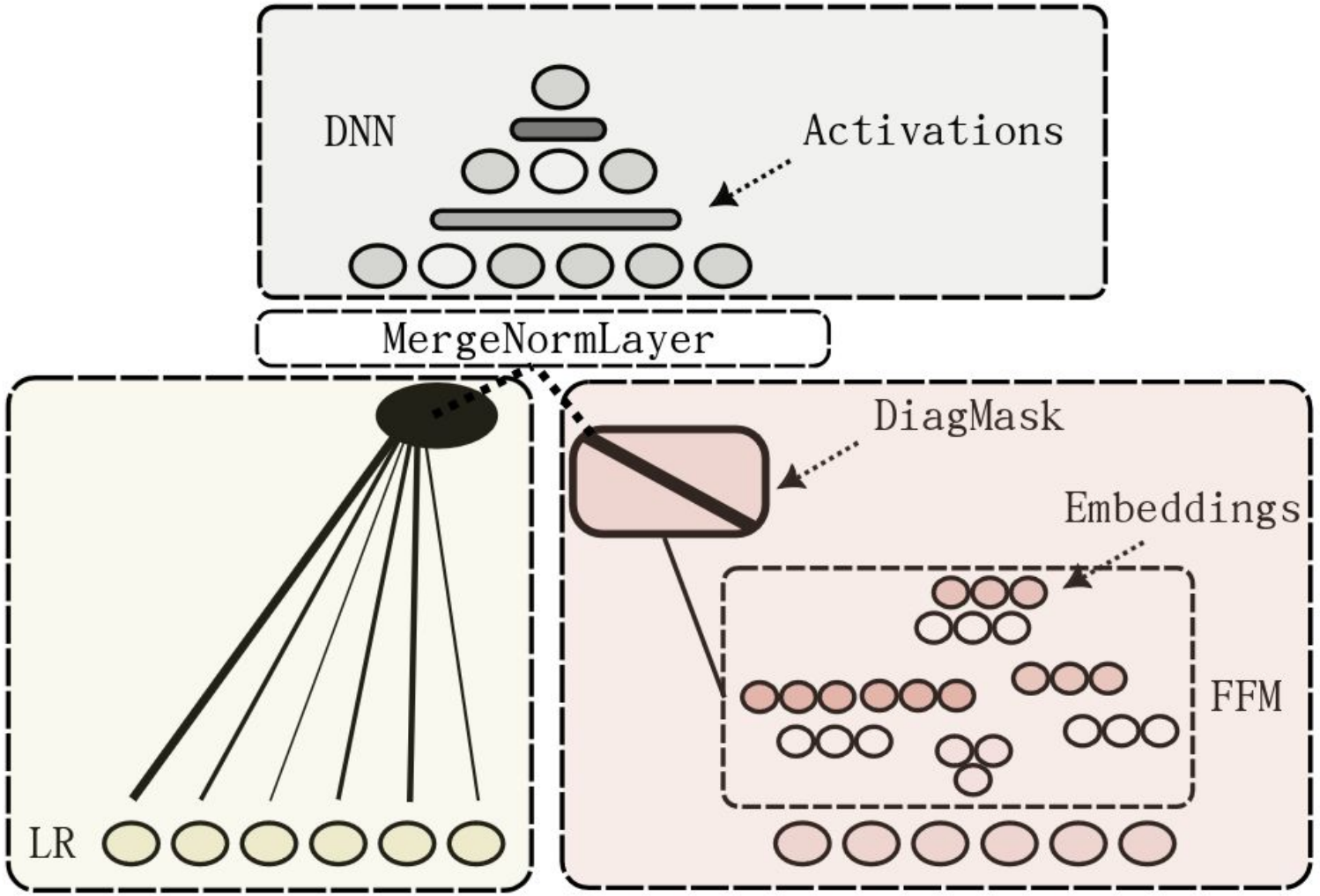
https://www.pexels.com/search/cat%20bag/

https://github.com/outbrain/fwumious_wabbit

Architecture

Few main components:

- Initially just FFMs + LR
- Deep block ***showed substantial lifts***



A benchmark

- Same amount of compute “per search”
- Data sets with higher volatility - **single-instance learning shines**
- Single-instance learning more stable (median window perf)
- Vowpal **works well** too

Avazu (window=30k)

algo	avg	median	max	std	min	test
VW-linear	0.6832	0.7016	0.8200	0.0668	0.4664	0.7596
VW-mlp	0.6755	0.6984	0.8200	0.0748	0.4664	0.7596
FW-DeepFFM	0.7648	0.7654	0.8507	0.0243	0.4764	0.7916
FW-FFM	0.7524	0.7524	0.8234	0.0227	0.4816	0.7693
DCNv2	0.7750	0.7745	0.8326	0.0202	0.5005	0.7763

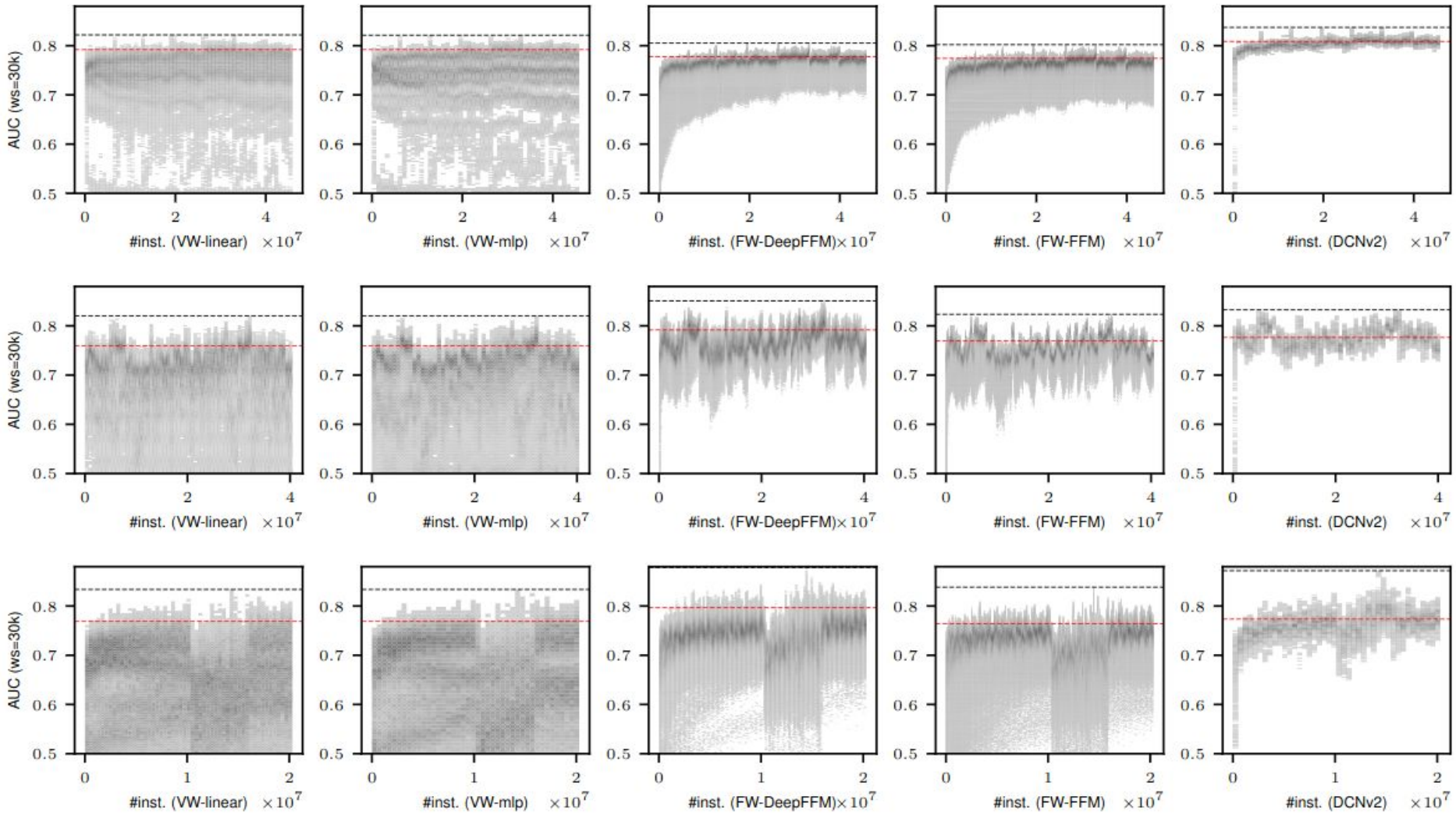
Criteo (window=30k)

algo	avg	median	max	std	min	test
VW-linear	0.7340	0.7460	0.8219	0.0556	0.4768	0.7920
VW-mlp	0.7247	0.7425	0.8211	0.0670	0.4768	0.7920
FW-DeepFFM	0.7655	0.7689	0.8053	0.0179	0.4796	0.7803
FW-FFM	0.7578	0.7621	0.8020	0.0198	0.4682	0.7742
DCNv2	0.8042	0.8052	0.8370	0.0118	0.4958	0.8085

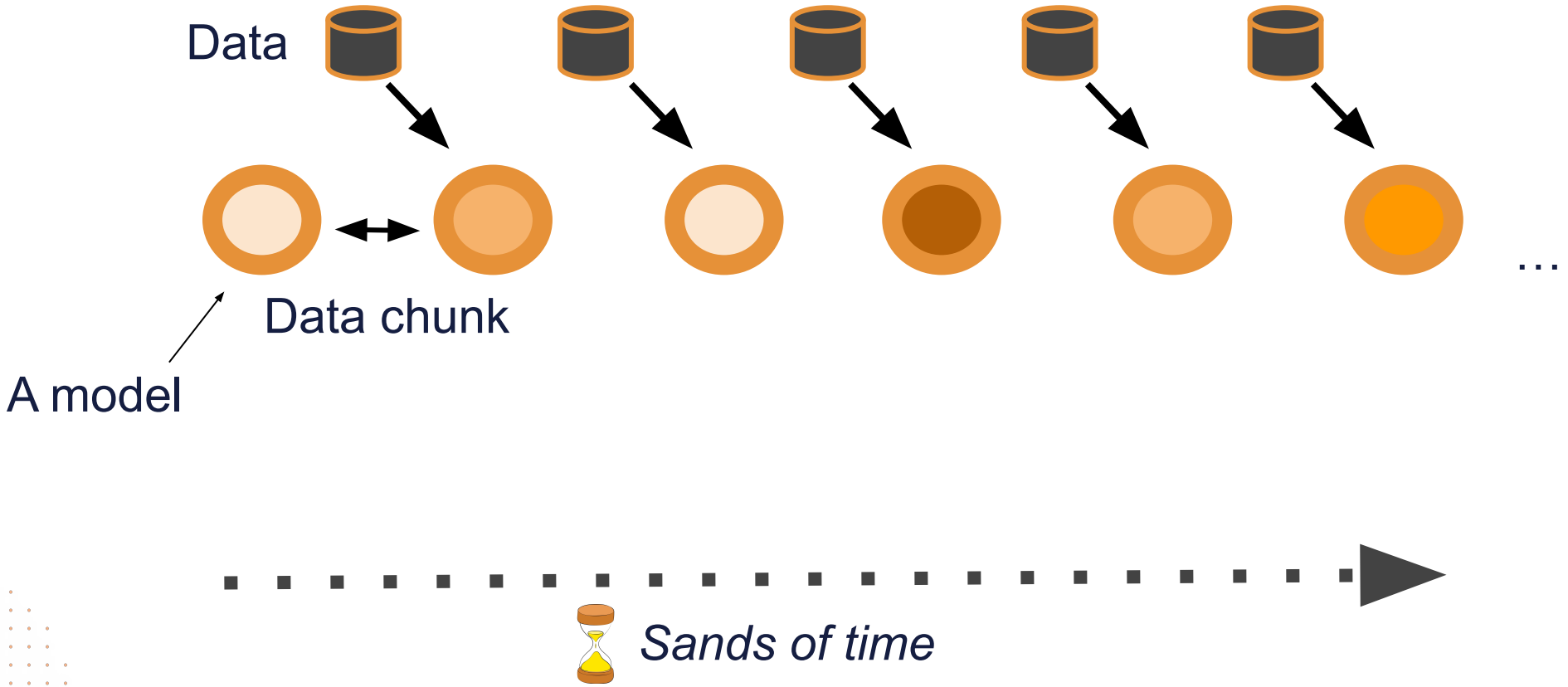
KDDCup2012 (window=30k)

algo	avg	median	max	std	min	test
VW-linear	0.6333	0.6419	0.8336	0.0807	0.3430	0.7688
VW-mlp	0.6309	0.6402	0.8336	0.0869	0.3759	0.7688
FW-DeepFFM	0.7323	0.7400	0.8781	0.0414	0.3687	0.7967
FW-FFM	0.7228	0.7318	0.8382	0.0391	0.3651	0.7641
DCNv2	0.7589	0.7610	0.8718	0.0301	0.4792	0.7734

Curve-based analysis reveals much more than just pushing one scalar



Model **warmup**



Outbrain

Main issue - too slow

HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent

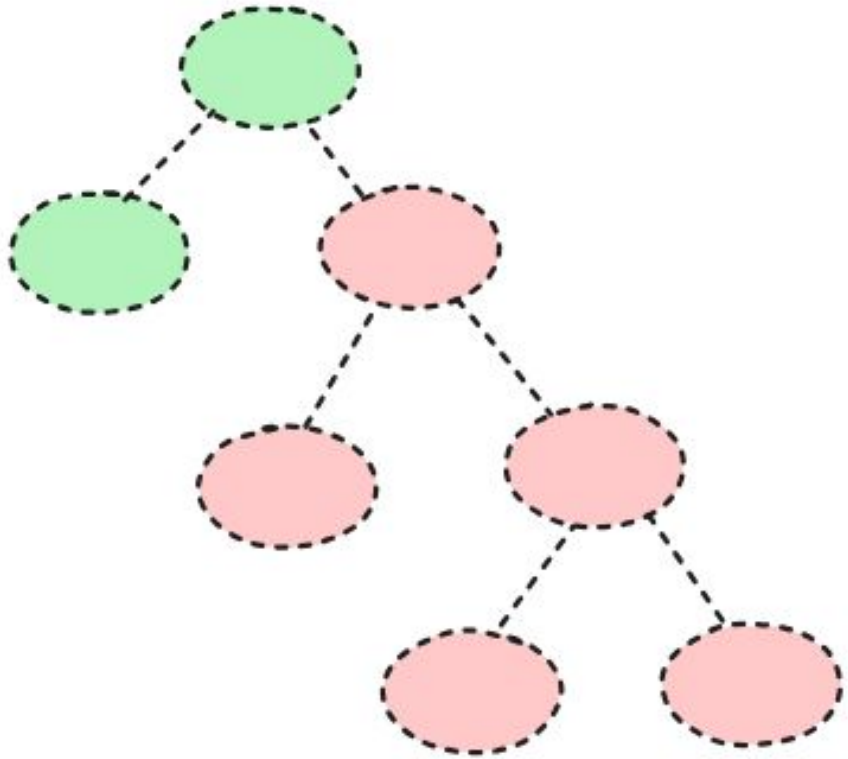
Feng Niu, Benjamin Recht, Christopher Re, Stephen J. Wright

Key idea: Embrace the **race condition** (data-level), and just merge outputs of updates.

Small drop/**non-deterministic** nature for the benefit of much **faster training**.

Implementation	Warmup time (same period)
FW-deepFFM-control	8d
FW-deepFFM-hogwild	23h (48 threads)
Implementation	**Online training (same period)**
FW-deepFFM-control	20m
FW-deepFFM-hogwild	4m (4 threads)

Sparse weight updates



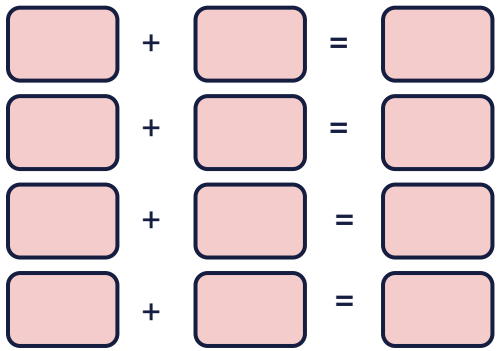
Sometimes, instance-level update kind of makes no sense, resulting in zero global gradient ..

- Rewriting “deep” part of FW so that it accounts for this property enables **skipping whole branches of backprop/update logic**, substantially speeding up training.
- Basically “if gradient at this point == 0, just skip a lot of code”

#Hidden layers	1	2	3	4
Speedup (sparse updates)	1.3x	1.8x	2.4x	3.5x

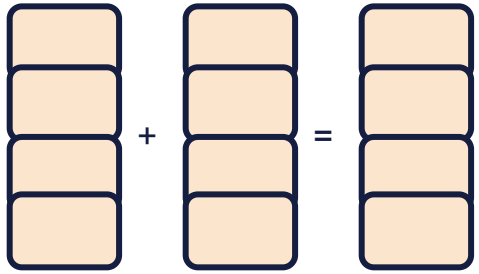
SIMD-based forward pass

- Single Instruction, Multiple Data



no SIMD

SIMD



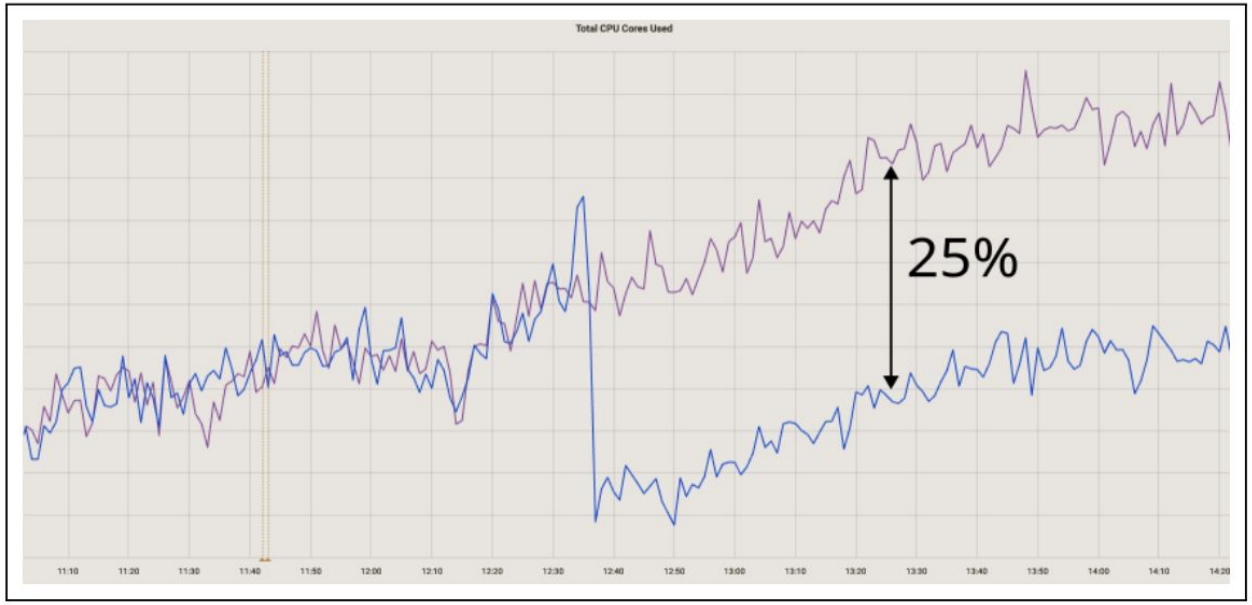


Figure 5: Relative impact of SIMD-enabled (blue, after drop) vs. SIMD-disabled (purple) FW in production (inference).

Context caching

- **Context = same,** candidates = different (per batch)
- What if we **cached context** (Radix tree-based cache)?

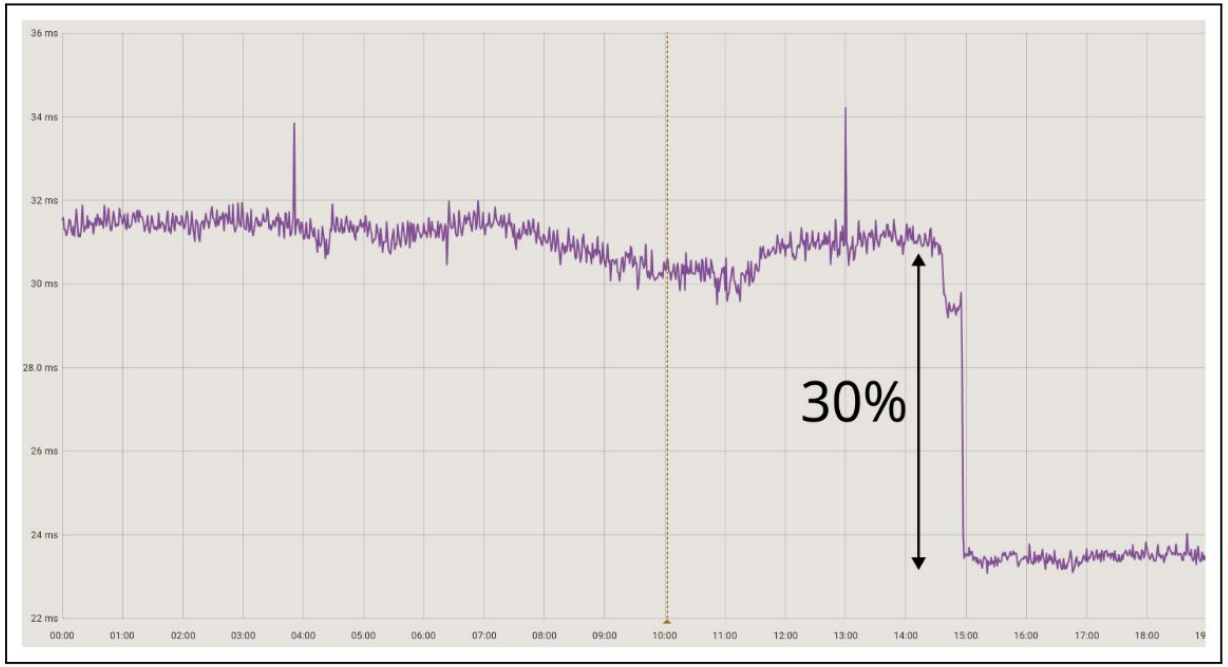
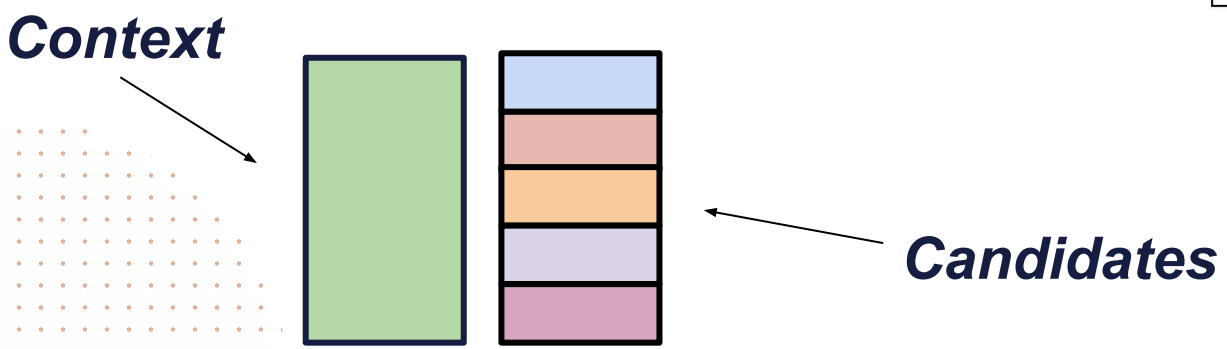
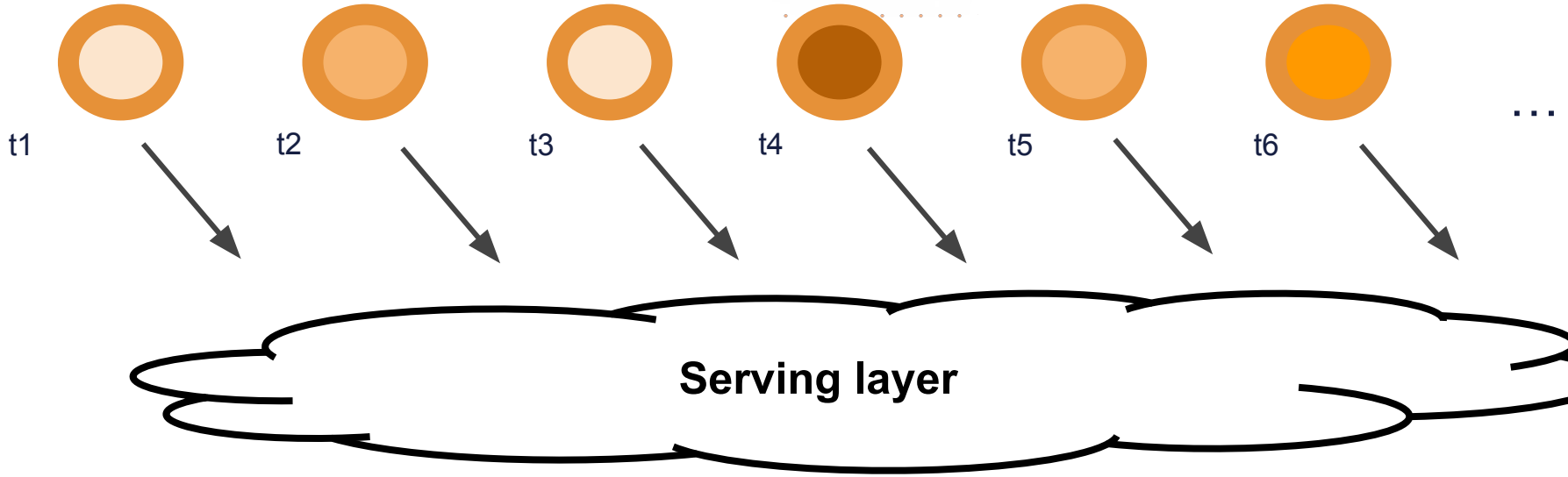


Figure 4: Impact of context caching on inference time.



Reminder: training -> serving

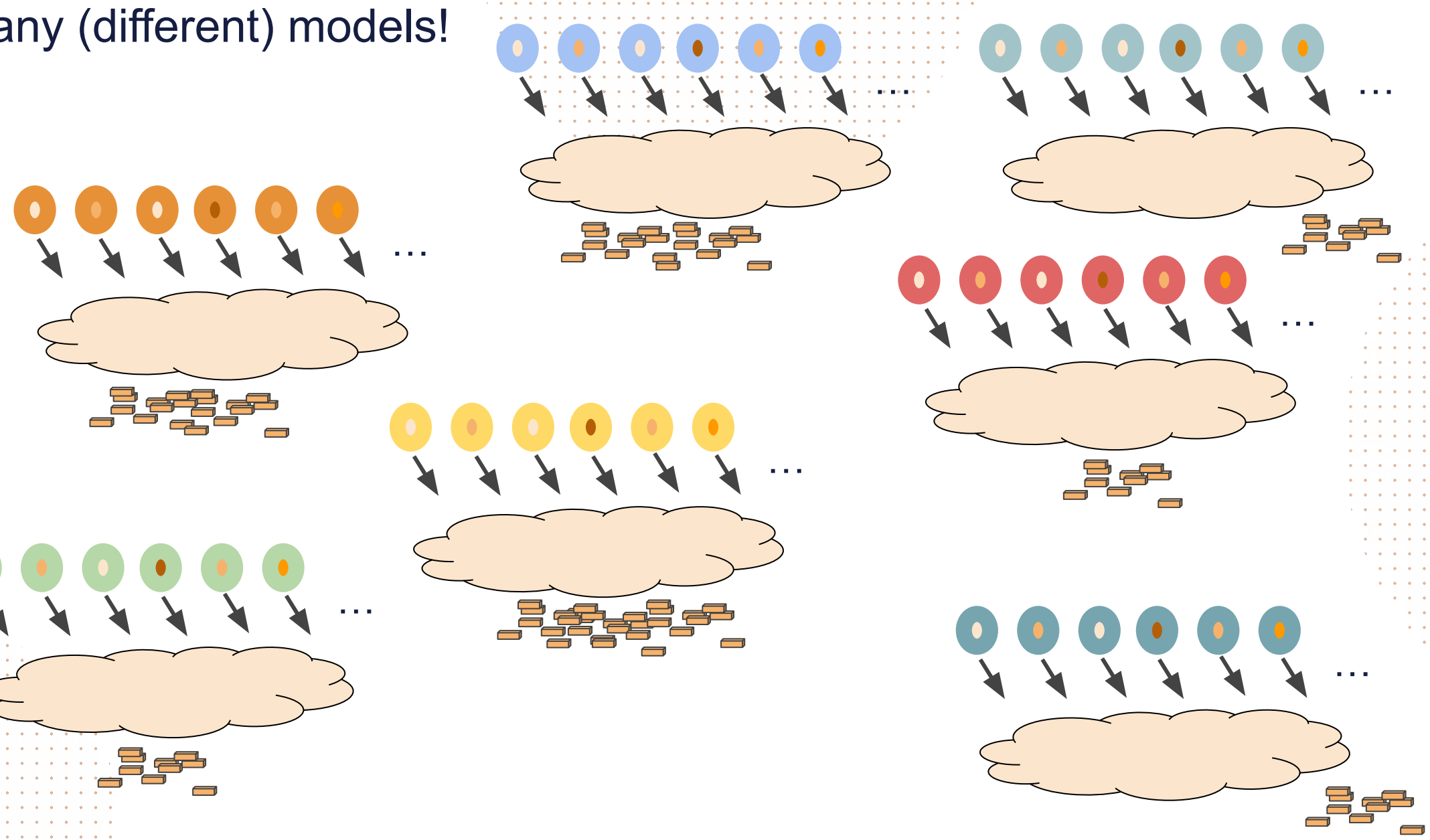
A model



Many requests

Outbrain

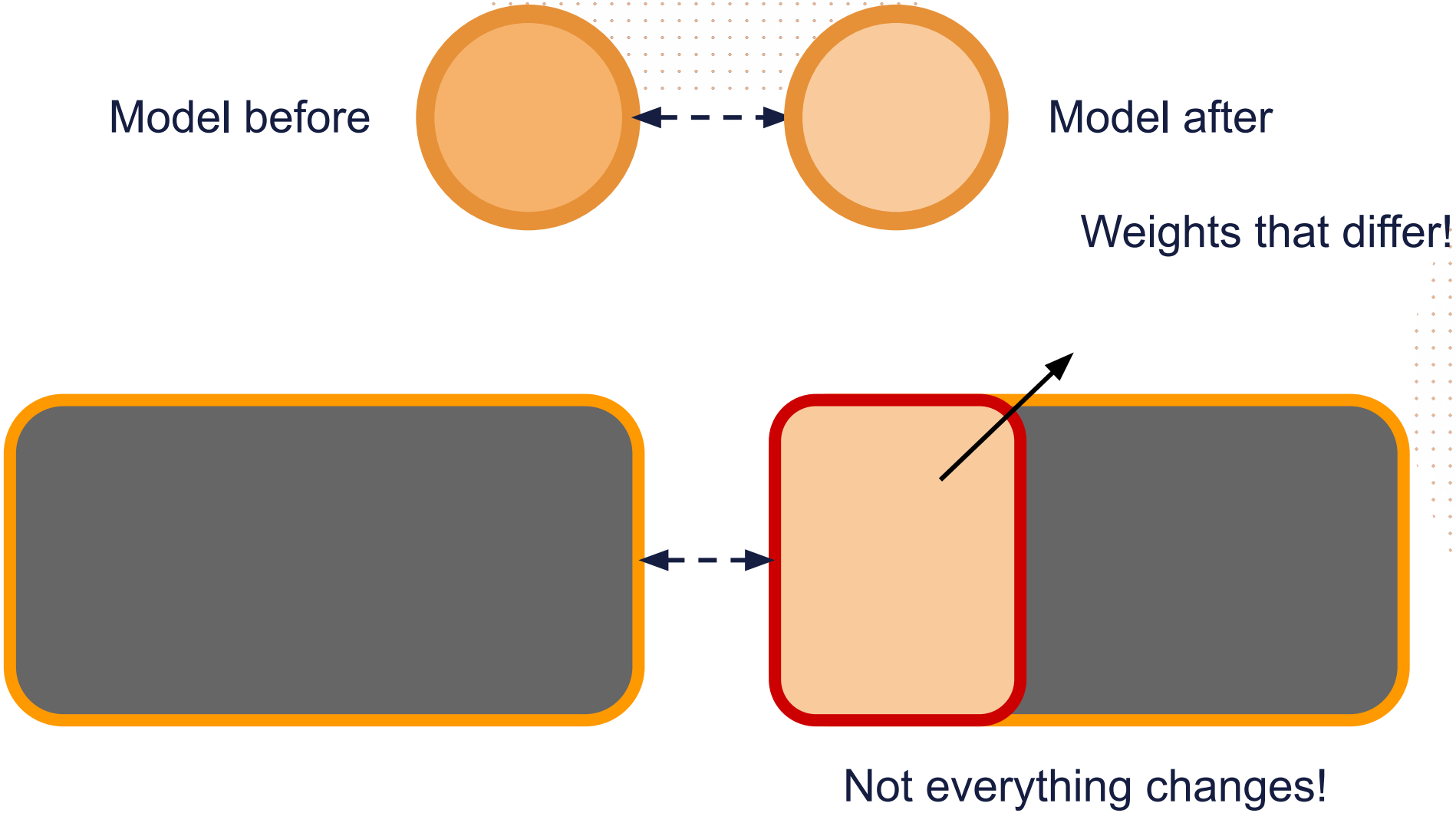
Many (different) models!



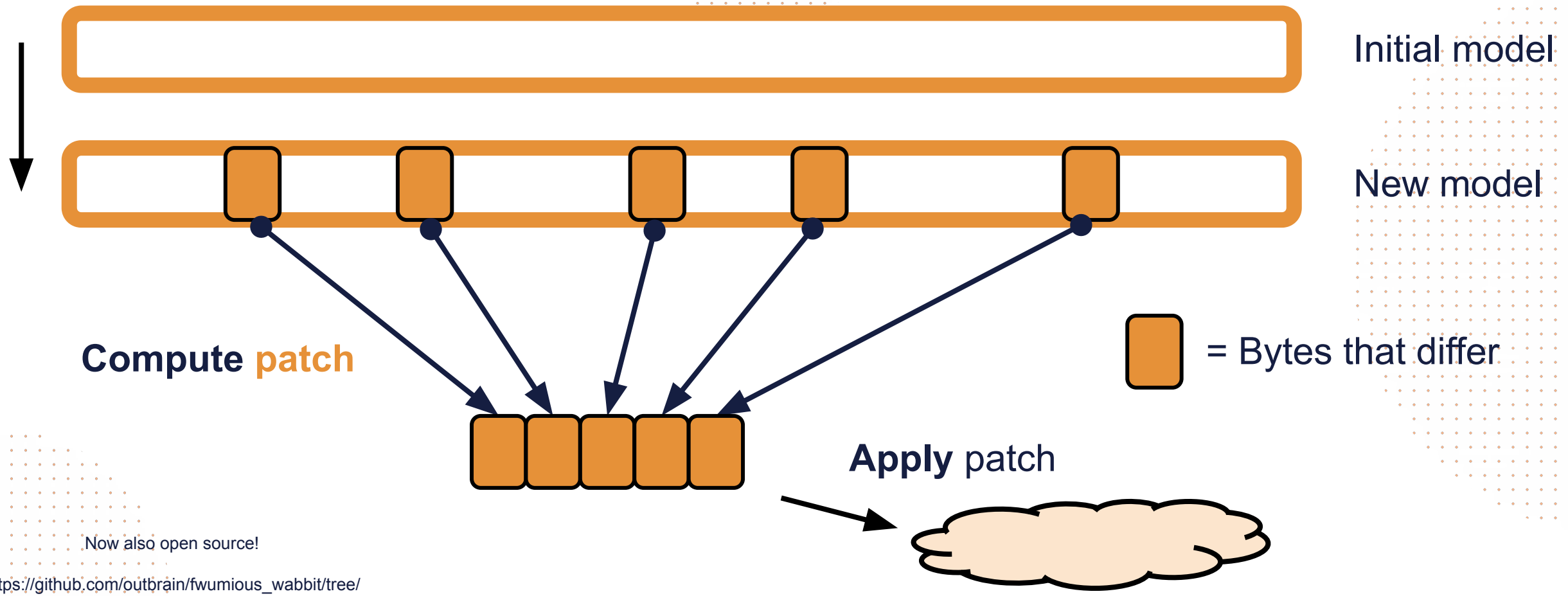
~8G -> ~80M

Outbrain

Model diffs



Weight patcher

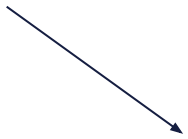


Now also open source!

https://github.com/outbrain/fwumious_wabbit/tree/main/weight_patcher

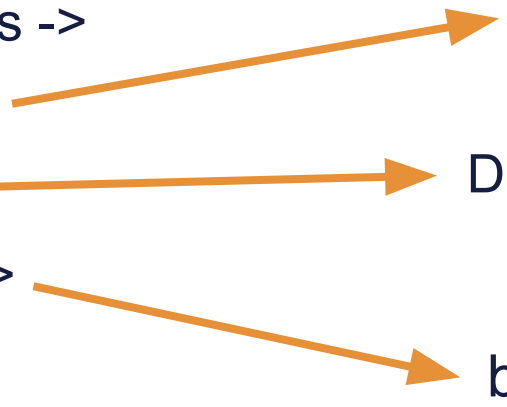
Quantization algorithm

1. One weight pass to get **statistics about weight space**
2. Compute quantization **bin sizes and bounds + dequant. header**
3. Second pass to **quantize** existing weights and store
4. (Serving) **dequantize** + serve



((weight - min) / weight_increment).round() **-> f16 ->** bytes

- Model train ->
- Inference weights ->
- *quantization* ->
- Patcher ->
- dequantization ->
- Serving layer



f32 -> bf16

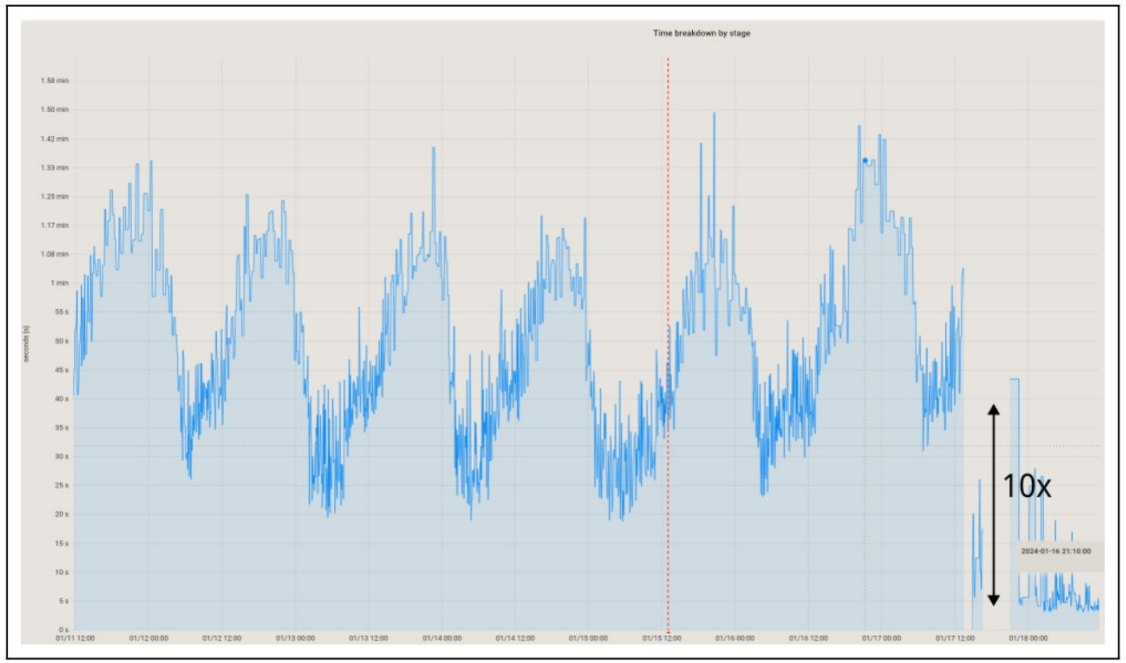
Impact: Half the size

Doesn’t care, works with bytes directly

Impact: up to 30x reduction

bin info -> f32

Weight processing	Avg. time spent	Update file size
no procecssing (baseline)	/	100%
fw-quantization	2s	50%
fw-patcher	45s	30±5%
fw-patcher + fw-quantization	**8s**	**3±2%**



What's next?

- **Different structure** of deep layers
- Better **quantization**
- **Hogwild** alternatives
- **Transfer learning**
- Different tasks (**mlc, mcc**)
- ***Inference quantization***