Doing Impactful Research in Industry

Dr. Kunal Banerjee

Principal Data Scientist

12-Mar-2024

Brief Biography

- Professional Experience
  - **Walmart:** Principal Data Scientist (Sep 2020 – Present)
  - **Intel Labs:** Research Scientist (Aug 2015 – Aug 2020)
- Educational Background
  - **PhD:** Computer Science & Engineering, IIT Kharagpur (2010 – 2015)
  - **BTech:** Computer Science & Engineering, Heritage Inst of Tech (2004 – 2008)
- Research Highlights
  - 8 journal publications
  - More than 50 conference/workshop publicatons
  - 1043 citations (as on March 12, 2024)
  - Best PhD Thesis Awards, Best Paper Awards, Special Mentions
  - IEEE Senior Member, ACM Senior Member

Deep Learning & HPC

Deep Learning is now ubiquitous:
self-driving cars, voice-activated assistants, automatic machine translation, image recognition, cancer detection, market price forecasting, chat bots, etc.

Table: Training time and top-1 validation accuracy with ImageNet/ResNet-50

	Batch Size	Processor	DL Library	Time	Accuracy
He et al.	256	Tesla P100×8	Caffe	29 hours	75.30%
Goyal et al.	8K	Tesla P100×256	Caffe	21 hours	76.30%
Smith et al.	16K	Full TPU Pod	TensorFlow	30 mins	76.10%
Akiba et al.	32K	Tesla P100×1024	Chainer	15 mins	74.90%
Jia et al.	64K	Tesla P40×2048	TensorFlow	6.6 mins	75.80%
Mikami et al.	68K	Tesla V100×2176	NNL	224 secs	75.03%

Source: `news.developer.nvidia.com/sony-breaks-resnet-50-training-record-with-nvidia-v100-tensor-core-gpus/`

"Finally, after decades of research, deep learning, the abundance of data, the powerful computation of GPUs came together in a big bang of modern AI."

– Jensen Huang, CEO of Nvidia

Deep Learning & HPC

Deep Learning is now ubiquitous:
self-driving cars, voice-activated assistants, automatic machine translation, image recognition, cancer detection, market price forecasting, chat bots, etc.

Table: Training time and top-1 validation accuracy with ImageNet/ResNet-50

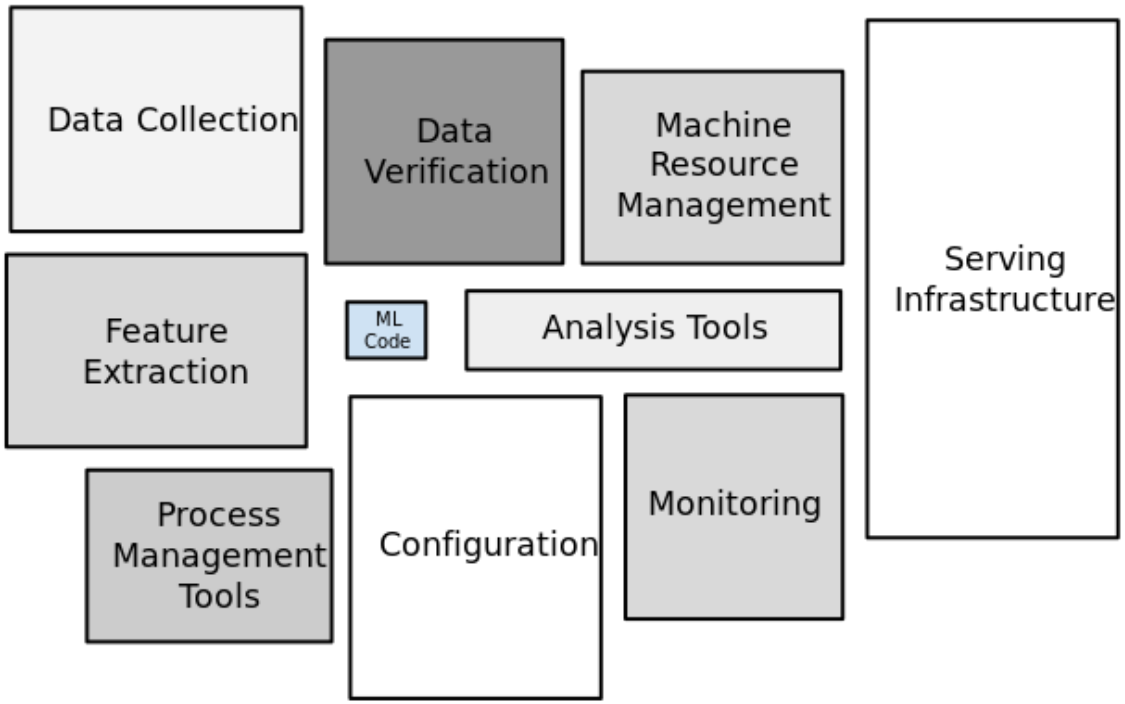
	Batch Size	Processor	DL Library	Time	Accuracy
He et al.	256	Tesla P100×8	Caffe	29 hours	75.30%
Goyal et al.	8K	Tesla P100×256	Caffe	21 hours	76.30%
Smith et al.	16K	Full TPU Pod	TensorFlow	30 mins	76.10%
Akiba et al.	32K	Tesla P100×1024	Chainer	15 mins	74.90%
Jia et al.	64K	Tesla P40×2048	TensorFlow	6.6 mins	75.80%
Mikami et al.	68K	Tesla V100×2176	NNL	224 secs	75.03%

Source: `news.developer.nvidia.com/sony-breaks-resnet-50-training-record-with-nvidia-v100-tensor-core-gpus/`

"Finally, after decades of research, deep learning, the abundance of data, the powerful computation of GPUs came together in a big bang of modern AI."

– Jensen Huang, CEO of Nvidia

Only 5% ML Code Exists in an ML System



Source: Scullery et al., Hidden Technical Debt in Machine Learning Systems, NeurIPS 2015

(Top-Down) Levels of Applying Optimization

- **Node-level:** Scaling to multiple nodes – **scope:** $> \mathbf{100}\times$
- **Model-level:** Explore alternate models and/or model compression
  E.g.: Use EfficientNet-B0 instead of Resnet-50 – **scope:** $\sim \mathbf{10}\times$
- **Compiler/Kernel-level:** Execute alternate optimized kernels
  E.g.: FFT/Winograd based convolution – **scope:** $\sim \mathbf{3-4}\times$
- **Precision-level:** Apply precision levels below FP32
  Nvidia Volta – peak FLOPS: 15 TF vs peak OPS (in FP16): 120 TF – **scope:** $\mathbf{8}\times$
  Nvidia Ampere – peak FLOPS: 19.5 TF vs peak OPS (in FP16, BFLOAT16): 312 TF (624 TF with sparsity) – **scope:** $\mathbf{32}\times$

☞ At Intel, we explored all levels for optimization; however, model-level was explored comparatively less. I got to explore this level at Walmart.

Bounds on Performance in HPC

- **Compute-bound:** Kernel spends most of its time in calculations, e.g., matrix multiplication – compute: $O(n^3)$, memory: $O(n^2)$
- **Bandwidth-bound:** Kernel spends most of its time in fetching the data, e.g., matrix addition – compute: $O(n^2)$, memory: $O(n^2)$
- **Latency-bound:** Kernel spends most of its time in memory fetches without saturating the global memory bus, e.g., accessing `A[B[i]]`

☞ Bandwidth-bound and latency-bound together constitute **memory-bound** kernels

☞ (Normal) Convolution is compute-bound except 1×1 convolutions, which being element-wise multiplications are memory-bound

☞ Personally, never worked with latency-bound kernels although these sometimes occur in HPC, e.g., FM-index based sequence search in computational-biology

☞ There are other types of bounds as well, e.g., I/O-bound

Bounds on Performance in HPC

- **Compute-bound:** Kernel spends most of its time in calculations, e.g., matrix multiplication – compute: $O(n^3)$, memory: $O(n^2)$
- **Bandwidth-bound:** Kernel spends most of its time in fetching the data, e.g., matrix addition – compute: $O(n^2)$, memory: $O(n^2)$
- **Latency-bound:** Kernel spends most of its time in memory fetches without saturating the global memory bus, e.g., accessing `A[B[i]]`

☞ Bandwidth-bound and latency-bound together constitute **memory-bound** kernels

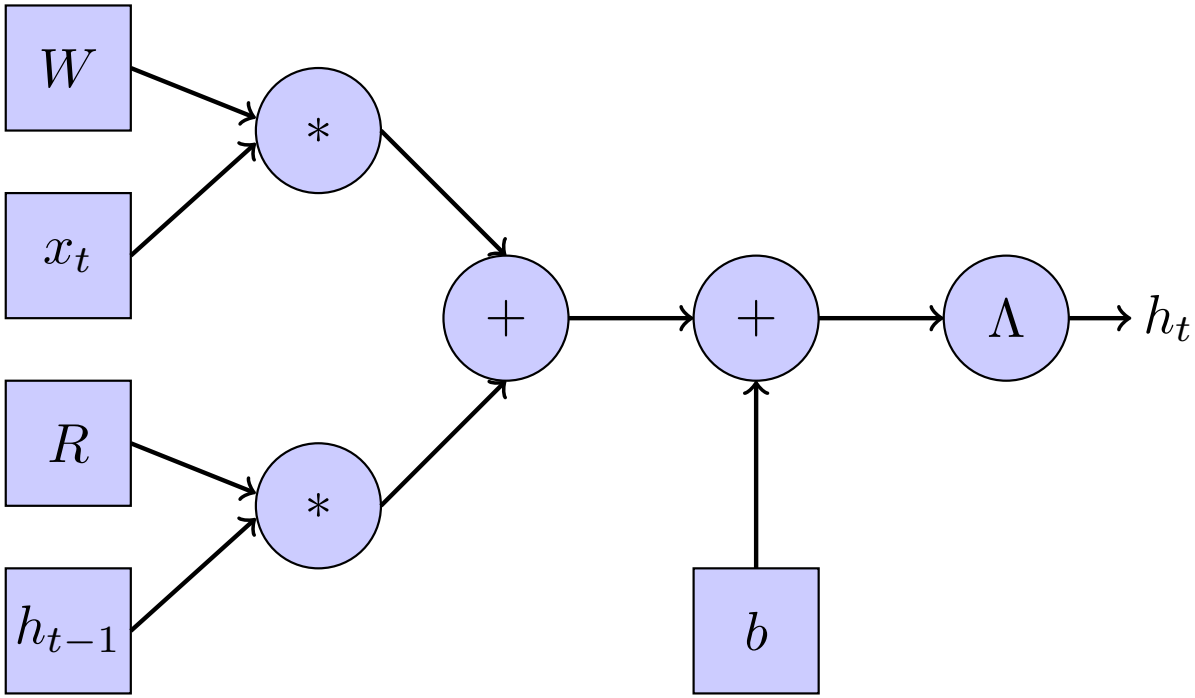
☞ (Normal) Convolution is compute-bound except 1×1 convolutions, which being element-wise multiplications are memory-bound

☞ Personally, never worked with latency-bound kernels although these sometimes occur in HPC, e.g., FM-index based sequence search in computational-biology

☞ There are other types of bounds as well, e.g., I/O-bound

Recurrent Neural Network (RNN)

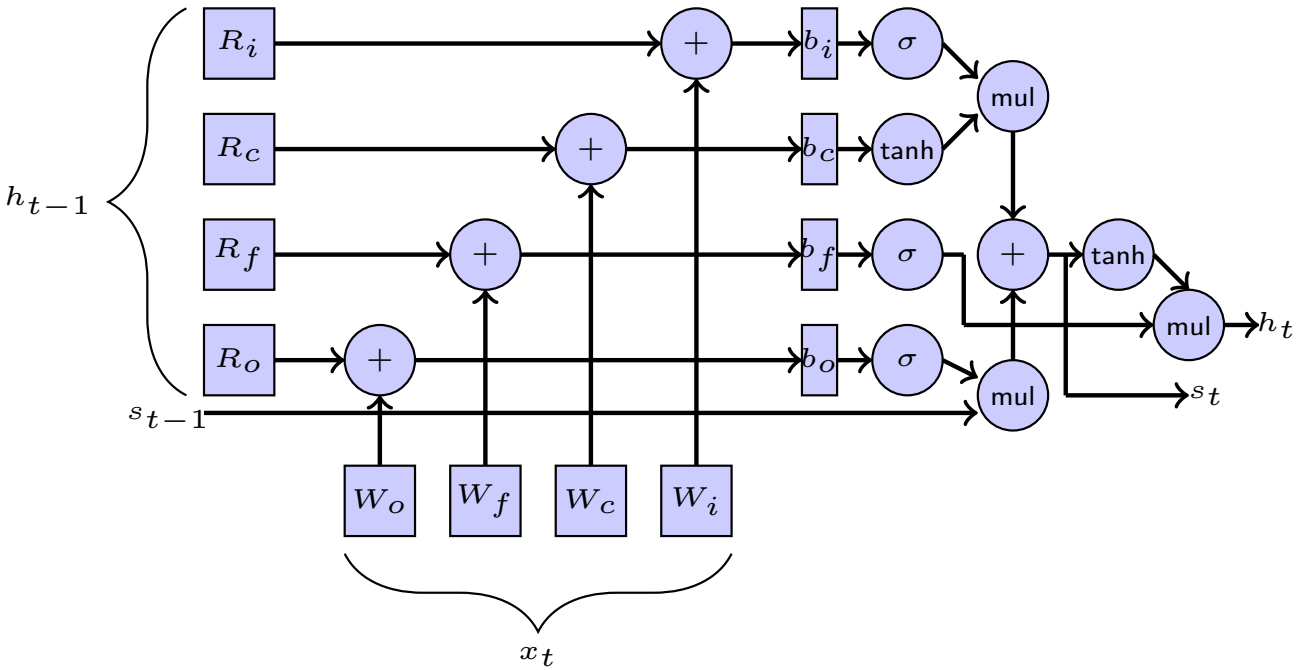
$$h_t = \Lambda(W * x_t + R * h_{t-1} + b)$$



$\Lambda \in \{\sigma, \tanh, \text{ReLU}\}$, i.e., a non-linear activation function

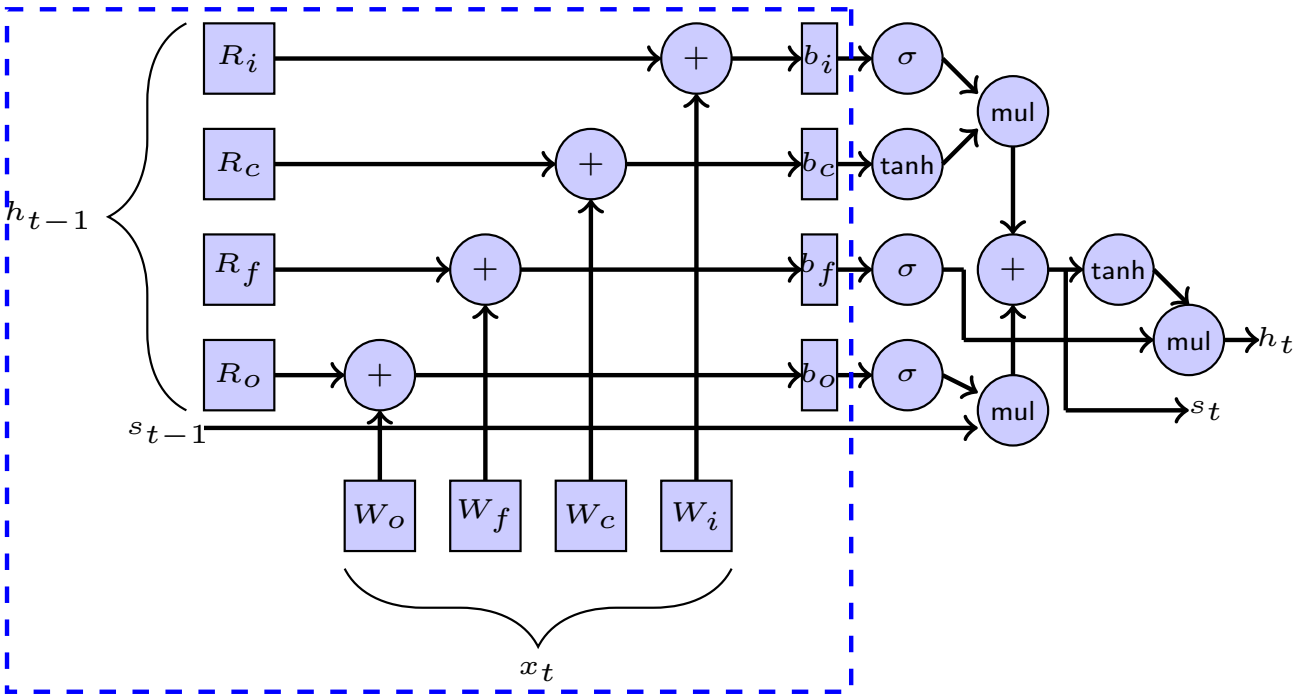
Long Short-Term Memory (LSTM)

$$i_t = \sigma(W_i * x_t + R_i * h_{t-1} + b_i)$$
$$c_t = \tanh(W_c * x_t + R_c * h_{t-1} + b_c)$$
$$f_t = \sigma(W_f * x_t + R_f * h_{t-1} + b_f)$$
$$o_t = \sigma(W_o * x_t + R_o * h_{t-1} + b_o)$$
$$s_t = f_t \circ s_{t-1} + i_t \circ c_t$$
$$h_t = o_t \circ \tanh(s_t)$$



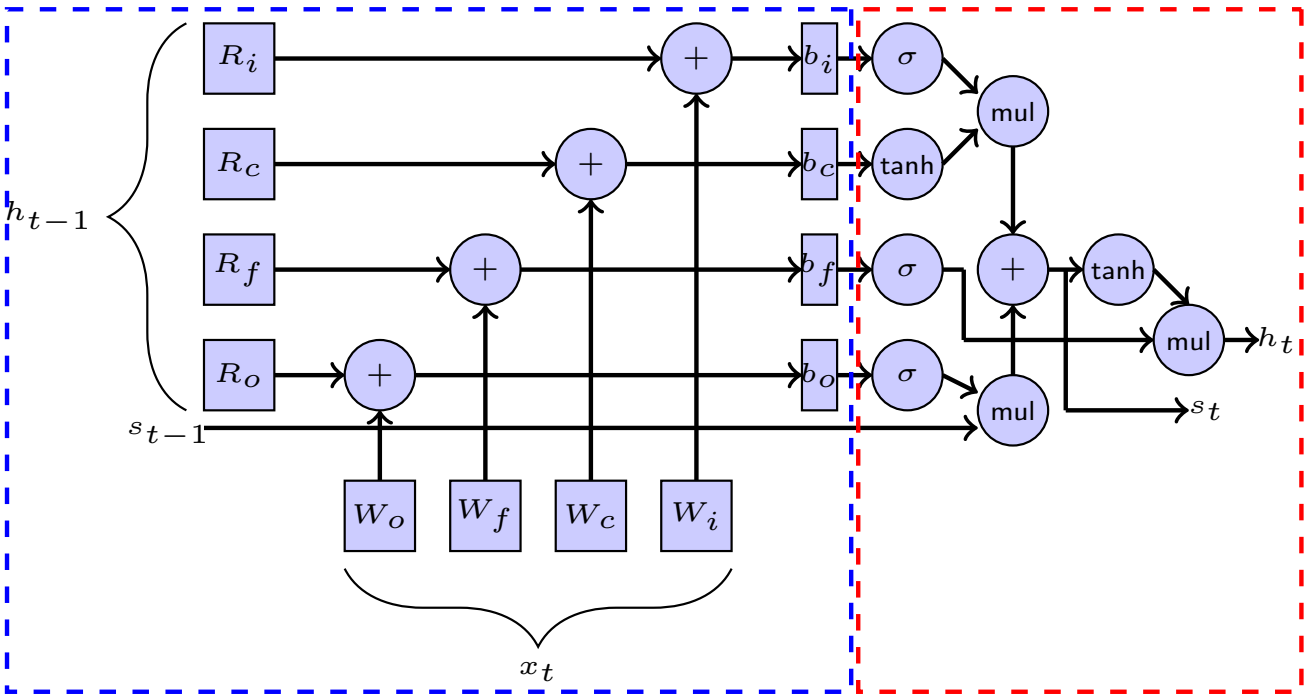
Long Short-Term Memory (LSTM)

$$i_t = \sigma(W_i * x_t + R_i * h_{t-1} + b_i)$$
$$c_t = \tanh(W_c * x_t + R_c * h_{t-1} + b_c)$$
$$f_t = \sigma(W_f * x_t + R_f * h_{t-1} + b_f)$$
$$o_t = \sigma(W_o * x_t + R_o * h_{t-1} + b_o)$$
$$s_t = f_t \circ s_{t-1} + i_t \circ c_t$$
$$h_t = o_t \circ \tanh(s_t)$$



Long Short-Term Memory (LSTM)

$$
\begin{aligned}
i_t &= \sigma(W_i * x_t + R_i * h_{t-1} + b_i) \\
c_t &= \tanh(W_c * x_t + R_c * h_{t-1} + b_c) \\
f_t &= \sigma(W_f * x_t + R_f * h_{t-1} + b_f) \\
o_t &= \sigma(W_o * x_t + R_o * h_{t-1} + b_o) \\
s_t &= f_t \circ s_{t-1} + i_t \circ c_t \\
h_t &= o_t \circ \tanh(s_t)
\end{aligned}
$$



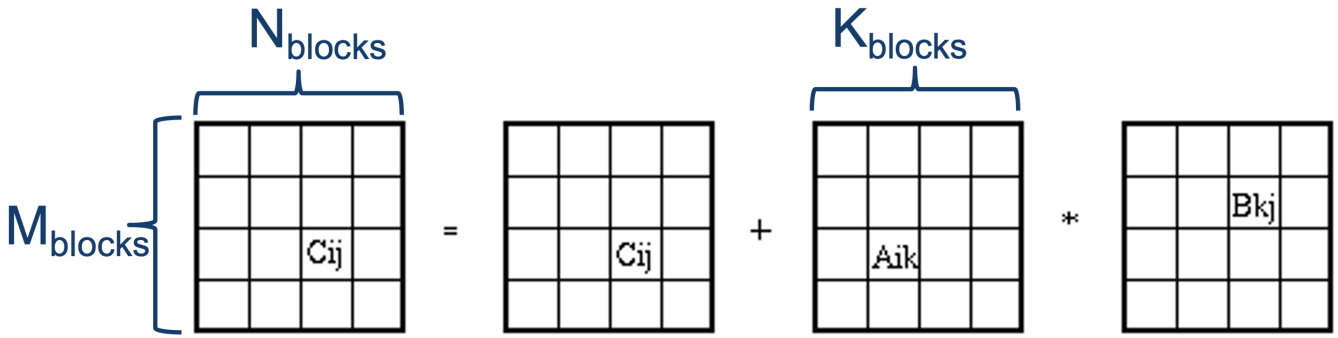
Implementation of RNN, LSTM, GRU

- Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are special cases of RNN – same methodology applies in general
- Main computation consists of GEMMs $W_{\{i,f,o,c\}} * x_t$ and $R_{\{i,f,o,c\}} * h_{t-1}$
- Element-wise operations are applied to the GEMM results
- Analogous equations for back-propagation kernels
- Perform two large GEMMs ($W * x$ and $R * h$) or one larger GEMM ($WR * xh$), then perform element-wise operations
  - ✓ Easy to implement – rely on vendor-optimized GEMM
  - ✗ Element-wise operations exposed as bandwidth-bound kernel (vs in-cache reuse of GEMM outputs)

Implementation of RNN, LSTM, GRU

- Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are special cases of RNN – same methodology applies in general
- Main computation consists of GEMMs $W_{\{i,f,o,c\}} * x_t$ and $R_{\{i,f,o,c\}} * h_{t-1}$
- Element-wise operations are applied to the GEMM results
- Analogous equations for back-propagation kernels
- Perform two large GEMMs ($W * x$ and $R * h$) or one larger GEMM ($WR * xh$), then perform element-wise operations
  - ✓ Easy to implement – rely on vendor-optimized GEMM
  - ✗ Element-wise operations exposed as bandwidth-bound kernel (vs in-cache reuse of GEMM outputs)

Batch-Reduce GEMM



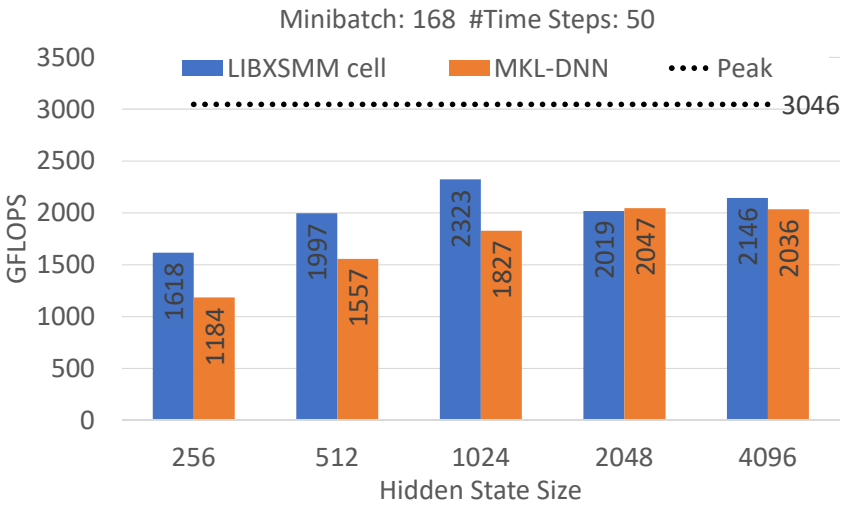
$$C_{ij} = C_{ij} + \sum_{k=1}^{K_{blocks}} A_{ik} * B_{kj}$$

Batch reduce GEMM

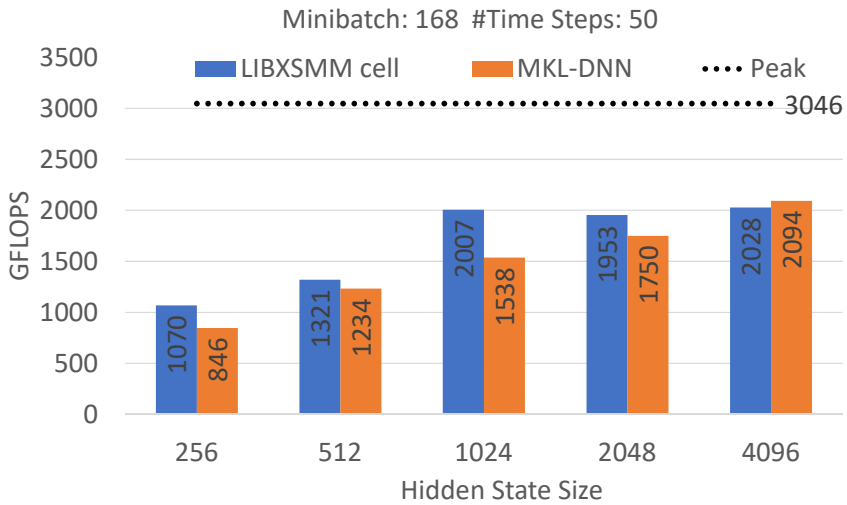
- ☞ The single batch-reduce GEMM kernel can act as the innermost kernel for a variety of deep learning topologies (CNN, RNN, LSTM, GRU, MLP) delivering SOTA performance
- ☞ Boosts **programmer productivity** which otherwise is spent tuning various kernels across different topologies
- ☞ All code available in `https://github.com/hfp/libxsmm`
- ★ E Georganas, K Banerjee et al., "Harnessing Deep Learning via a Single Building Block," IPDPS 2020

Comaprison with Intel® MKL-DNN

☞ Intel® MKL-DNN is an open source performance library from Intel



Forward pass results

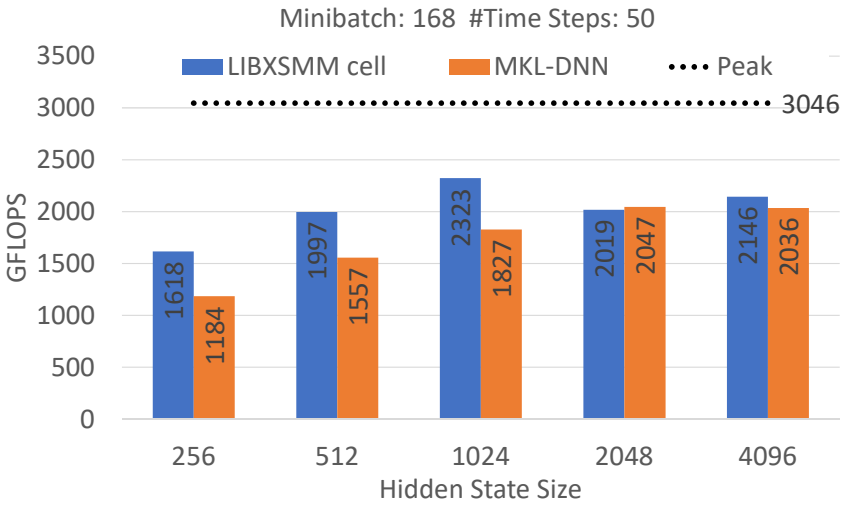


Backward/weight update pass results

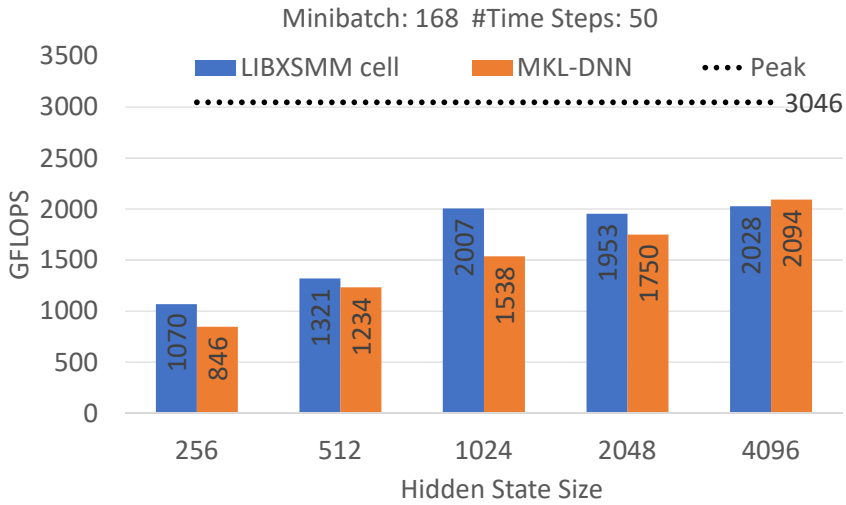
- **Machine:** Single socket Xeon Platinum 8180 with 28 Cores
- For small/medium sized problems, forward pass is up to $\mathbf{1.4}\times$ faster, while for backward/weight update it is up to $\mathbf{1.3}\times$ faster
- For large weight matrices the two approaches have similar performance because GEMM has *cubic* scaling whereas element-wise has *quadratic* scaling

Comaprison with Intel® MKL-DNN

☞ Intel® MKL-DNN is an open source performance library from Intel



Forward pass results



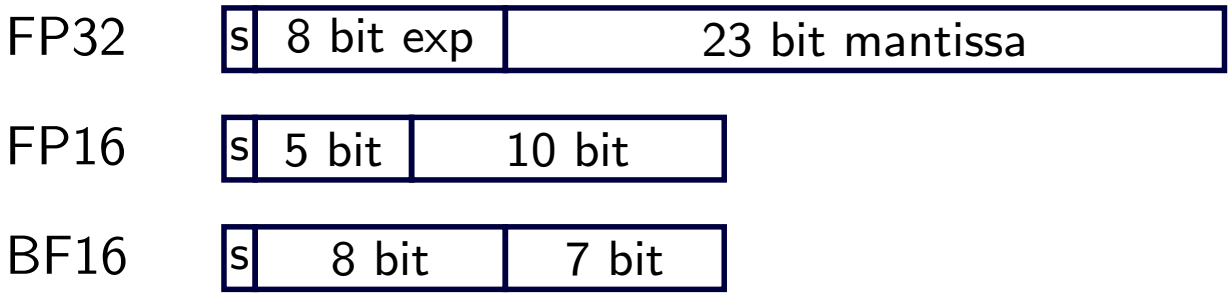
Backward/weight update pass results

★ K Banerjee et al., "Optimizing Deep Learning LSTM Topologies on Intel Xeon Architecture," ISC 2019 (Best Research Poster – AI & ML track)

★ K Banerjee et al., "Optimizing Deep Learning RNN Topologies on Intel Architecture," JSFI 2019 (invited journal paper)

Low-Precision Datatype & bfloat16

Low-Precision: Any datatype below FP32



BF16 has several advantages over FP16:

- It can be seen as a short version of FP32, skipping the least significant 16 bits of mantissa
- There is no need to support denormals; FP32, and therefore also BF16, offer more than enough range for deep learning training tasks
- Hardware exception handling is not needed

Source: https://software.intel.com/sites/default/files/managed/40/8b/bf16-hardware-numerics-definition-white-paper.pdf

★ D Kalamkar et al., "A Study of BFLOAT16 for Deep Learning Training," 2019

☞ BF16 has been adopted by Intel, Nvidia, AMD and others after we showcased its efficacy

A Perturbation Theory View of Low-Precision Inference

A Perturbation Theory View of Low-Precision Inference

fire engine (8a-2w)

A Perturbation Theory View of Low-Precision Inference

fire engine (8a-2w)

A Perturbation Theory View of Low-Precision Inference

ice bear (8a-4w)

A Perturbation Theory View of Low-Precision Inference

ice bear (8a-4w)

A Perturbation Theory View of Low-Precision Inference

old English sheepdog (8a-6w)

A Perturbation Theory View of Low-Precision Inference

old English sheepdog (8a-6w)

A Perturbation Theory View of Low-Precision Inference

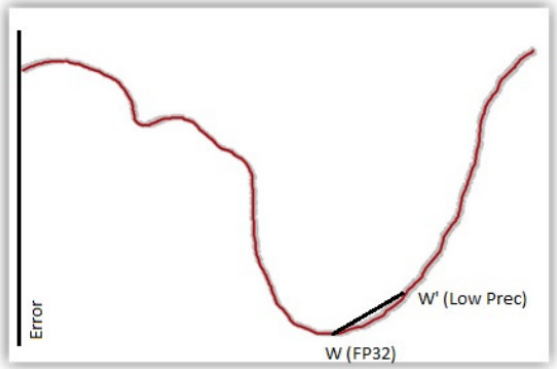
dalmatian (8a-8w)

A Perturbation Theory View of Low-Precision Inference

dalmatian (8a-8w)

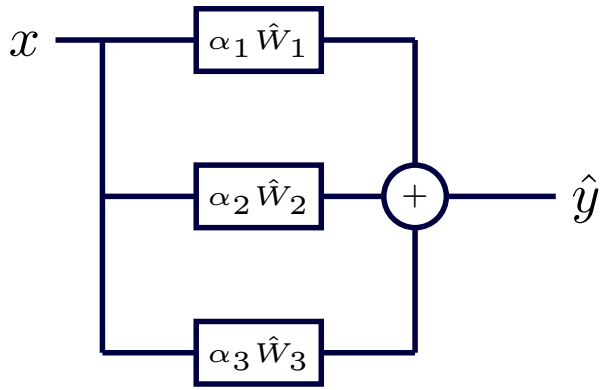
A Perturbation Theory View of Low-Precision Inference

- Low precision (and sparsification): adding noise to weights/activations
- Need to preserve the output of each layer such that no/little loss of accuracy
- 8-bit weight and 8-bit activations (8a-8w) shown to work well
  - $\sim$1% loss of accuracy from FP-32 on very deep CNNs (e.g. ResNet101)
- Sub-8-bit weights and/or activations incur noticeable drop in accuracy
- Our observation:
  - Not all the weights may be well-represented by sub-8-bit precision
  - Not all the weights may need 8-bit precision
- Our approach: Keep low-precision weights in a neighborhood of FP32 weights using only 8-bit activations and Ternay weights



Ternary Residual Inference

Ternary Weights: $\alpha\hat{W} \simeq W, \hat{W}_i = sign(W_i)$, if $|W_i| > T$, and 0 otherwise
$E(\alpha, T) = \|W - \alpha\hat{W}\|_F^2$, $\alpha^*, T^* = \underset{\alpha \geq 0, T \geq 0}{argmin} E(\alpha, T), \alpha \geq 0, \hat{W}_i \in \{-1, 0, +1\}$



Theoretical understanding of sensitivity of weights and/or activations to final classification accuracy

- Highly sensitive layers (e.g. first conv layer) require more precision
- We want uniform low-precision operations rather than multi-precision

Ternary Residual Edge: add more ternary edges selectively s.t.

- More ternary compute for certain parts of the augmented network
- Significant recovery of loss with overall less compute cost than 8a-8w

★ A Kundu, K Banerjee et al., "Ternary Residual Networks," SysML 2018

Ternary Residual Inference (contd.)

- Further fine-tuning via block-wise ternary residual
  - Partition the weights in disjoint blocks (of N elements)
  - Convert to ternary, and add r number of residual blocks (if necessary)
- N and r control the model size, accuracy, and ternary compute
- We can adjust the accuracy-compute trade-off on-the-fly during inference (unlike other models) by disabling least important residual blocks
- Results of Ternary Residual conversion of ResNet-101 pre-trained on ImageNet
  - Total number of blocks (without residual) is $1\times$
  - Overall $\sim 2\times$ savings in compute cost comparing to 8a-8w with similar accuracy

Block Size	$\sim 1\%$		$\sim 2\%$	
	#Blocks	mult reduction	#Blocks	mult reduction
$N = 64$	$2.4\times$	$26\times$	$2\times$	$32\times$
$N = 256$	$2.8\times$	$90\times$	$2.4\times$	$105\times$

Text Extraction from Images: Challenges

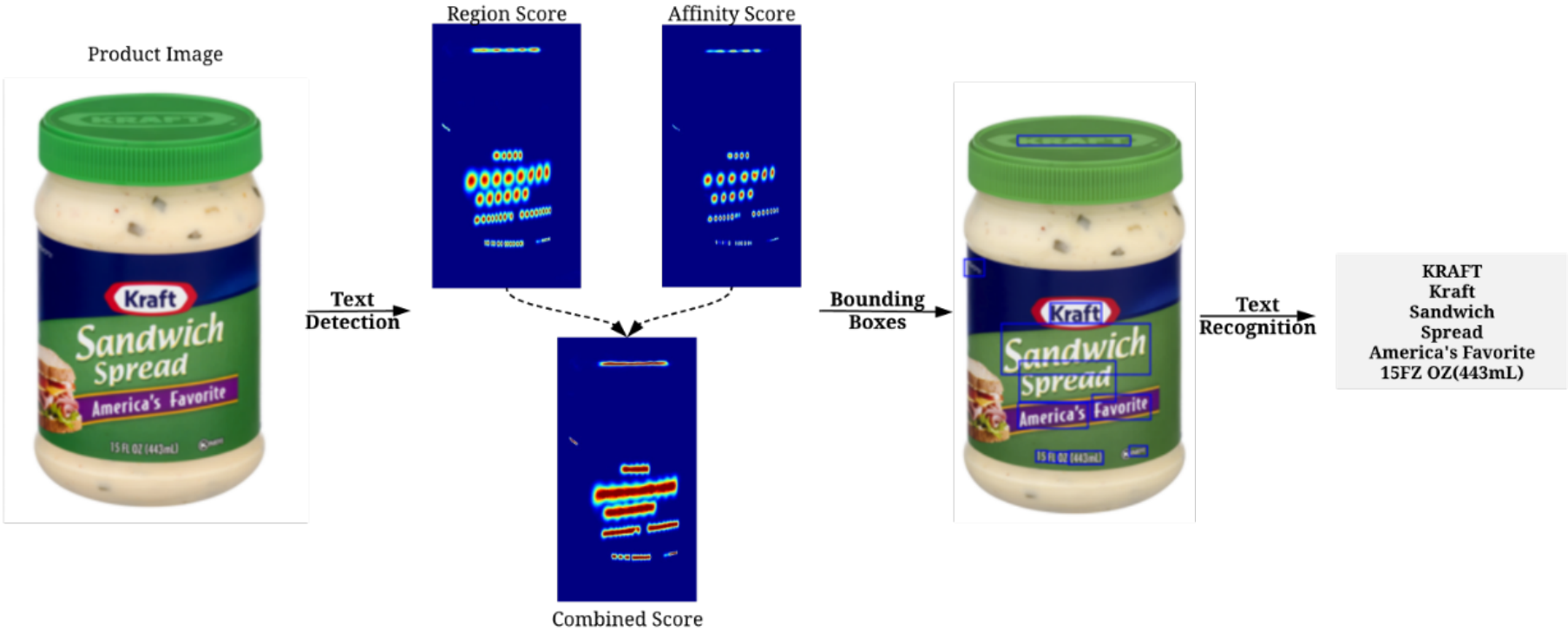
Characteristics of texts on product images:

- Non-standard fonts and sizes
- Text can be vertical or inverted
- Text can be irregularly oriented or curved
- Non-dictionary words (e.g., brand names)
- High local entropy

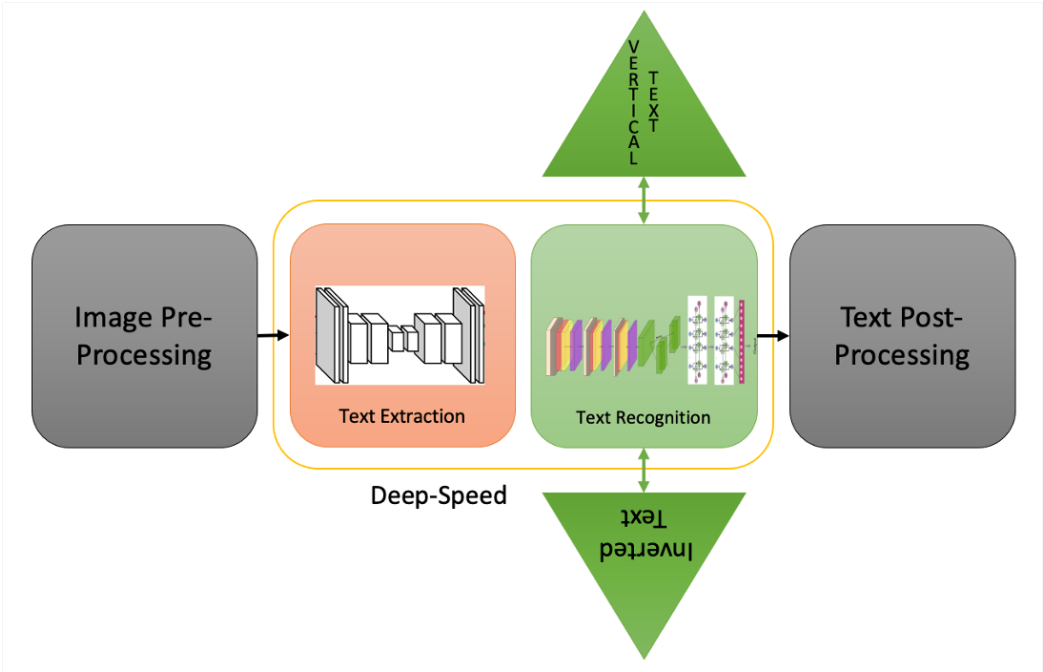
Text Extraction from Images: Use-cases

- In the context of scale at which Walmart operates, the text from an image can be **a richer and more accurate source of data** than human inputs
- Used in *several applications* such as Attribute Extraction (MRP, Country of Origin, Ingredients), Offensive Text Classification, Product Matching
- The solution provided is proven to work at million image scale for various retail business units within Walmart while saving 30% computational cost in both the training and the inference stages
- ★ P Dugar et al., "From Pixels To Words: A Scalable Journey Of Text Information From Product Images To Retail Catalog," CIKM 2021

Generic Pipeline for Text Extraction from Images



Walmart's Pipeline for Text Extraction from Images



Handling Vertical Texts

Case 1

Case 2

- Case 1 can be handled by rotating the text by 90° or 270°
- Case 2 requires slicing each character and then putting these together to get the word

Handling Inverted Texts

Confidence scores are **low**

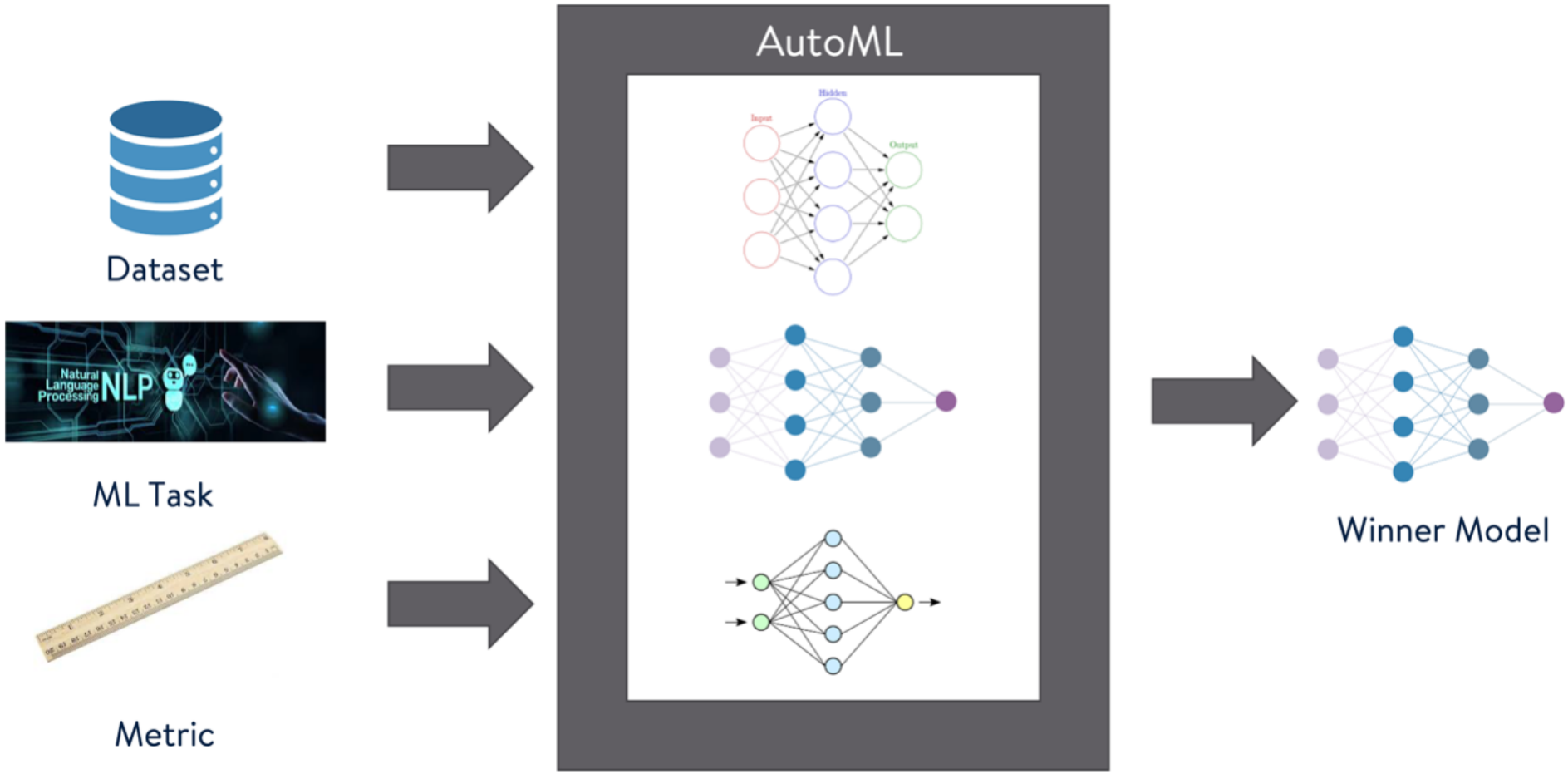
Confidence scores are **high**

Text Extraction from Images in Action: Cargo Identification

Further enhancements:

- ★ P Dugar et al., "Don't Miss the Fine Print! An Enhanced Framework To Extract Text From Low Resolution Images," VISAPP 2022
- ★ S Misra et al., "Designing a Vision Transformer based Enhanced Text Extractor from Product Images," CoDS-CoMAD 2023
- ★ S Misra et al., "BARGAIN: A Super-Resolution Technique to Gain High-Resolution Images for Barcodes," CoDS-CoMAD 2024

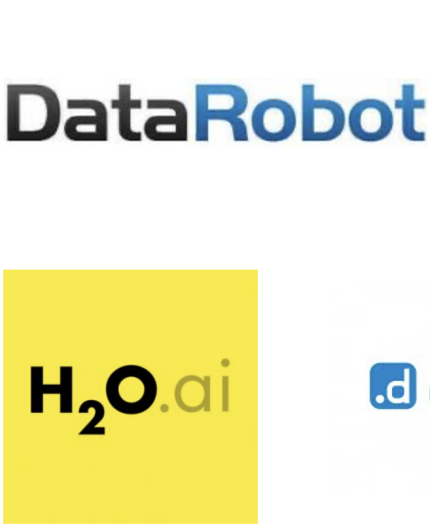
What is AutoML?



Advantages of AutoML

- **Democratizing AI**
  - You don't need to be an ML expert to use ML now
- **Reduced Time-to-Market**
  - Less time spent in development
  - Less chance of bugs in the code
- **Realizing Operational Excellence**
  - AutoML injects more standardization into the generation of models
  - Models may be generated keeping scaling in mind
- **Reduced Cost**
  - Less head count needed (Data Scientists are typically hard to retain)
  - An efficient model leads to less inference cost

What are some off-the-shelf AutoML solutions?







Introduction to WALTS *(Do we even need it?)*

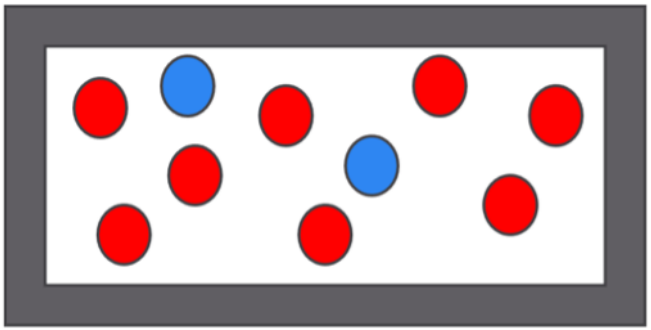
WALTS = Walmart AutoML Libraries, Tools and Services

Motivation: WALTS tries to address some of the shortcomings that we discovered in other commercial AutoML tools as mentioned below.

1. Customizability
2. Transparency
3. Low-precision Models
4. Scope for Unique Enhancements
5. Data Sensitivity
6. Reduced Time-to-Explore
7. Cost Efficient

★ R Bajaj, K Banerjee et al., "WALTS: Walmart AutoML Libraries, Tools and Services," SEAA 2022

Benefits of WALTS: Customizability



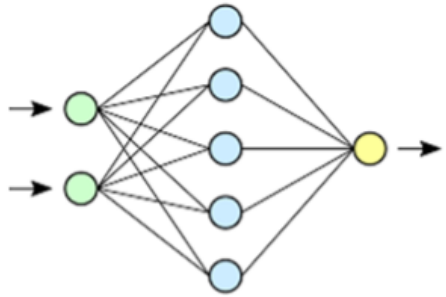
Model = It predicts "red" always
Accuracy = 0.8
Balanced Accuracy = 0.5

- At the time of publication, all off-the-shelf AutoML services provide a fixed set of metrics with no provision to extend this set by the users
- Some of these, at times, can even be misleading, e.g., accuracy when the classes are imbalanced
- After talking to internal clients' requirements, we may easily introduce new metrics, such as, balanced accuracy which is better suited for imbalanced classes

Benefits of WALTS: Transparency

Accuracy: 82%
Latency: 100 ms

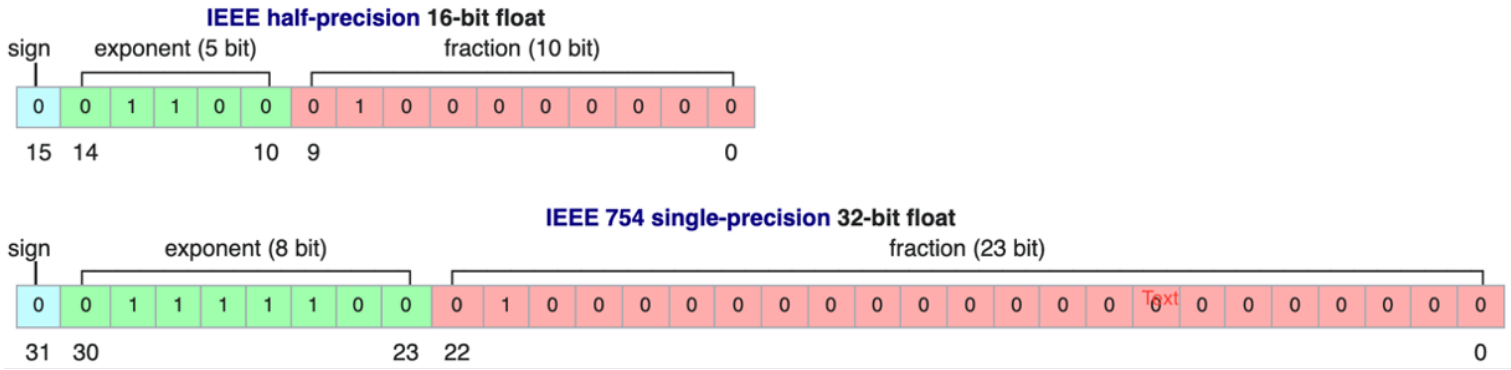
VS



Accuracy: 80%
Latency: 50 ms

- Typically, the AutoML frameworks report only the winner model
- WALTS provides details of all the explored models and their performance
- What if there are some hard latency requirements?
  - The client may be happy with the second-best model (in terms of accuracy, say) if it fits her latency needs — this kind of support can only be given by WALTS

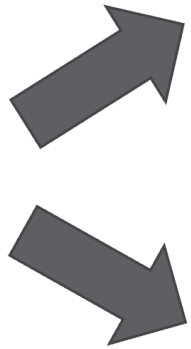
Benefits of WALTS: Low-precision Models



- Low-precision models lead to **reduced memory footprint** and **reduced latency**
  - Nvidia V100 - peak FLOPS: 15 TF vs peak OPS (in FP16): 120 TF — scope: **8x**
  - Nvidia A100 - peak FLOPS: 19.5 TF vs peak OPS (in FP16): 312 TF — scope: **32x**
  - There are other low-precision datatypes: BFLOAT16, INT16, INT8, ...
- Other AutoML frameworks produce models with FP32 datatype only
  - You need a separate tool, e.g., TensorFlow Lite, to do the subsequent conversion
- WALTS produces low-precision versions for some models if V100 is provided

Benefits of WALTS: Scope for Unique Enhancements

34	23	65		55
26		54	-4	
11	21	32	49	7
18	43	97	44	67
45		43	23	17
16	4		54	
9	58	49	33	57
65	35	23	76	45



11	21	32	49	7
18	43	97	44	67
9	58	49	33	57
65	35	23	76	45

34	23	65	6	55
26	43	54	-4	35
11	21	32	49	7
18	43	97	44	67
45	9	43	23	17
16	4	20	54	26
9	58	49	33	57
65	35	23	76	45

- We plan to extend WALTS with unique enhancements, e.g., noise-aware training, intelligent data imputation
- We found that other AutoML tools drop rows with missing data instead of imputing these

Benefits of WALTS: Data Sensitivity

- WALTS being an in-house tool is more secure than others
- Some compliance policies, e.g., HIPAA, may prevent uploading data to the cloud

Benefits of WALTS: Reduced Time-to-explore & Cost Efficiency

Reduced Time-to-explore:

- Commercial AutoML's complex search algorithms may find ad-hoc architectures at the cost of trying out a combinatorial number of different parameters (layers, connections)
- WALTS's simple policy of exploring only a pre-defined set of models may be less time consuming
- Moreover, the models explored by WALTS being already popular, the clients may be more open to adopting these

Cost Efficiency:

- We do not plan to impose minimum charges for WALTS

WALTS AutoML Algorithm

Inputs: A task T, a dataset D and a metric m

Output: A model M

1: Select set $S = M_1, M_2, ..., M_K$ of candidate models based on task T and type of data in D

2: $m_{max} \leftarrow 0$, $M_{max} \leftarrow \Phi$

3: for each model M_i in S do

4: Train M_i on dataset D and note its performance m_i

/* Note that we use Optuna for hyper-parameter optimization of M_i in this step */

5: if $m_i > m_{max}$ then

6: $m_{max} \leftarrow m_i$, $M_{max} \leftarrow M_i$

7: end if

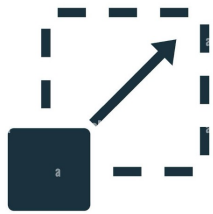
8: end for

9: return M_{max}

Implementation Considerations

RELIABILITY

EFFICIENCY



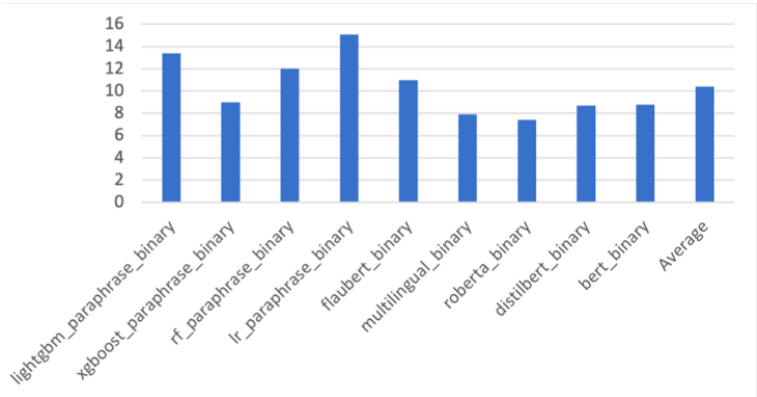
SCALABILITY

WALTS Results and Impacts

Training Results with WALTS

Task	Dataset	Model	Accuracy	F1 Score	Precision	Recall	TTT (min)
Image Classification	CIFAR-100	**ConvNeXt-B**	**67.9**	**67.7**	**70.9**	**67.9**	10.83
		EfficientNet B0	49.1	48.8	52.4	49.1	28.33
		MobileNet V3	47.8	48.3	52.8	47.8	17.25
		ResNext-50-32x4d	53.3	52.7	55.3	53.3	21.82
		VGG16	57.7	57.4	60.9	57.7	10.95
		Xception	46.1	45.8	49.4	46.1	19.83
	Tiny ImageNet	**ConvNeXt-B**	**80.5**	**80.5**	**82.0**	**80.5**	24.27
		EfficientNet B0	58.3	58.1	60.1	58.3	66.67
		MobileNet V3	57.9	57.9	60.1	57.9	39.00
		ResNext-50-32x4d	64.2	64.1	66.1	64.2	39.47
		VGG16	58.6	58.5	60.6	58.6	18.00
		Xception	50.9	50.8	52.9	50.9	33.83
Text Classification	Dispute Categorization	BERT-base	90.4	89.7	90.6	90.4	9.45
		BERT-multilingual	90.8	90.2	91.2	90.8	18.15
		DistilBERT	91.3	90.9	91.9	91.3	16.98
		FlauBERT	87.6	86.6	87.5	87.6	26.20
		Linear Regression	89.5	89.4	89.3	89.5	52.00
		RoBERTa	**91.6**	**91.4**	**92.5**	**91.6**	17.42
		XGBoost	91.1	90.9	90.9	91.1	47.00
Named Entity Recognition	Annotated Corpus for NER using GBM	BERT	99.1	99.1	99.3	99.1	22.00
		BERT-multilingual	98.9	99.0	99.2	98.9	26.51
		DeBERTa	99.0	99.0	99.1	99.0	24.23
		DistilBERT	**99.2**	**99.2**	**99.3**	**99.2**	16.05
		RoBERTa	98.5	98.6	99.0	98.5	22.30

%age Reduction in Time-to-Train



- Improved productivity and scalability
  - Reduced head count
  - Reduced development time
- Efficient resource and infrastructure management
  - Reduced infrastructure costs
- Bridging skill gaps and reducing scope of error in applying AI/ML algorithms

"Research is seeing what everybody else has seen and thinking what nobody else has thought."

– *Albert Szent-Györgyi*

"Research is formalized curiosity. It is poking and prying with a purpose."

– *Zora Neale Hurston*

Thank you!

Email: Kunal.Banerjee1@walmart.com
Homepage: `https://kunalbanerjee.github.io/`