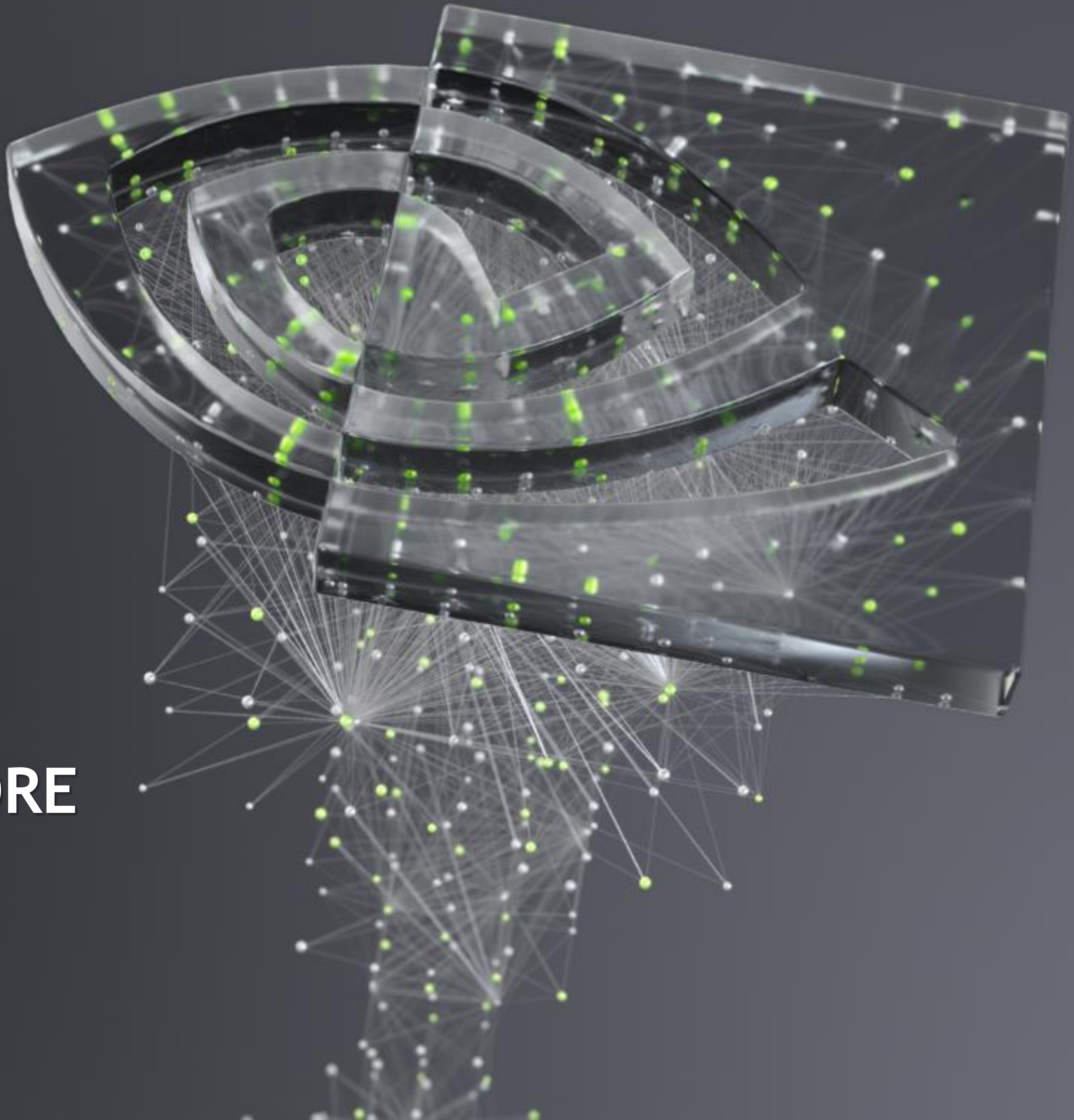
THE TRADER OF TOMORROW: SMARTER, FASTER, AND MORE PYTHONIC

John Ashley, October 2020



Timing is everything.

DON'T MISS JENSEN HUANG'S GTC KEYNOTE

October 5, 06:00 PDT (UTC-7)

SAVE THE DATE ›

Timing is everything.

NVIDIA GTC

DON'T MISS JENSEN HUANG'S GTC KEYNOTE

October 5, 06:00 PDT (UTC-7)

SAVE THE DATE ›

Timing is everything.

Portions (all) of this talk were pre-recorded in front of a live studio audience.

NVIDIA

GTC

DON'T MISS JENSEN HUANG'S GTC KEYNOTE

October 5, 06:00 PDT (UTC-7)

SAVE THE DATE ›

Timing is everything.

Portions of this talk were pre-recorded in front of a live studio audience.

The Trader of the Future

Smarter

NLP, BERT, and domain specific language models; or

“The more you learn, the more you earn.” – Warren Buffet

Faster

Optimizing AI models for inference; or

“Simplify, then add lightness.” – Colin Chapman

More Pythonic

Pointers to various useful bits of accelerated Python; or

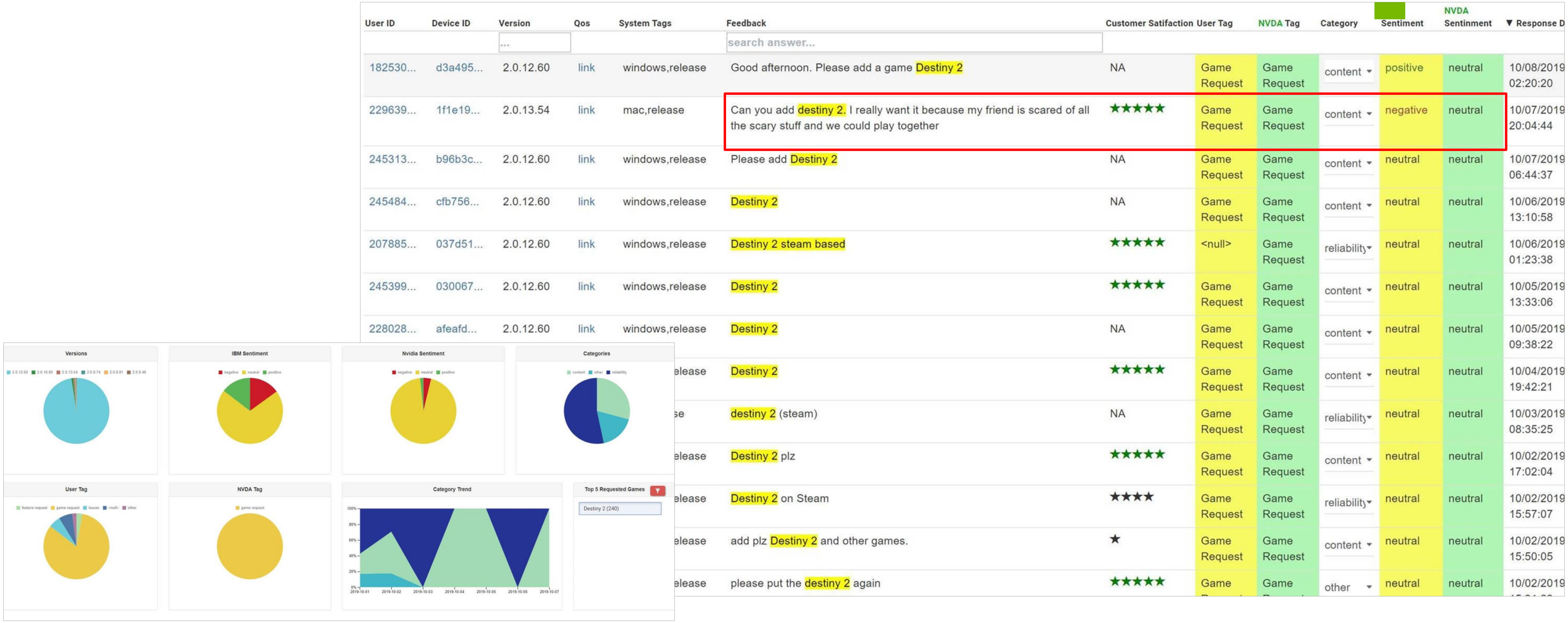
“Every sufficiently advanced LISP application will eventually reimplement Python.” – Hodgson’s Law

“The more you learn, the more you earn.”
– Warren Buffet

SMARTER

UNDERSTAND GEFORCE NOW USERS

Thousands of gamer comments every day



User ID	Device ID	Version	Qos	System Tags	Feedback	Customer Satisfaction	User Tag	NVDA Tag	Category	Sentiment	NVDA Sentinment	▼ Response D
		...			search answer...							
182530...	d3a495...	2.0.12.60	link	windows,release	Good afternoon. Please add a game Destiny 2	NA	Game Request	Game Request	content ▾	positive	neutral	10/08/2019 02:20:20
229639...	1f1e19...	2.0.13.54	link	mac,release	Can you add destiny 2. I really want it because my friend is scared of all the scary stuff and we could play together	★★★★★	Game Request	Game Request	content ▾	negative	neutral	10/07/2019 20:04:44
245313...	b96b3c...	2.0.12.60	link	windows,release	Please add Destiny 2	NA	Game Request	Game Request	content ▾	neutral	neutral	10/07/2019 06:44:37
245484...	cfb756...	2.0.12.60	link	windows,release	Destiny 2	NA	Game Request	Game Request	content ▾	neutral	neutral	10/06/2019 13:10:58
207885...	037d51...	2.0.12.60	link	windows,release	Destiny 2 steam based	★★★★★	<null>	Game Request	reliability ▾	neutral	neutral	10/06/2019 01:23:38
245399...	030067...	2.0.12.60	link	windows,release	Destiny 2	★★★★★	Game Request	Game Request	content ▾	neutral	neutral	10/05/2019 13:33:06
228028...	afeafd...	2.0.12.60	link	windows,release	Destiny 2	NA	Game Request	Game Request	content ▾	neutral	neutral	10/05/2019 09:38:22
				elease	Destiny 2	★★★★★	Game Request	Game Request	content ▾	neutral	neutral	10/04/2019 19:42:21
				se	destiny 2 (steam)	NA	Game Request	Game Request	reliability ▾	neutral	neutral	10/03/2019 08:35:25
				elease	Destiny 2 plz	★★★★★	Game Request	Game Request	content ▾	neutral	neutral	10/02/2019 17:02:04
				elease	Destiny 2 on Steam	★★★★	Game Request	Game Request	reliability ▾	neutral	neutral	10/02/2019 15:57:07
				elease	add plz Destiny 2 and other games.	★	Game Request	Game Request	content ▾	neutral	neutral	10/02/2019 15:50:05
				elease	please put the destiny 2 again	★★★★★	Game	Game	other ▾	neutral	neutral	10/02/2019

LANGUAGE UNDERSTANDING IMPROVEMENT

Reaching human level

NOT A STAC BENCHMARK

GLUE Aggregate Score

Detect grammatical errors

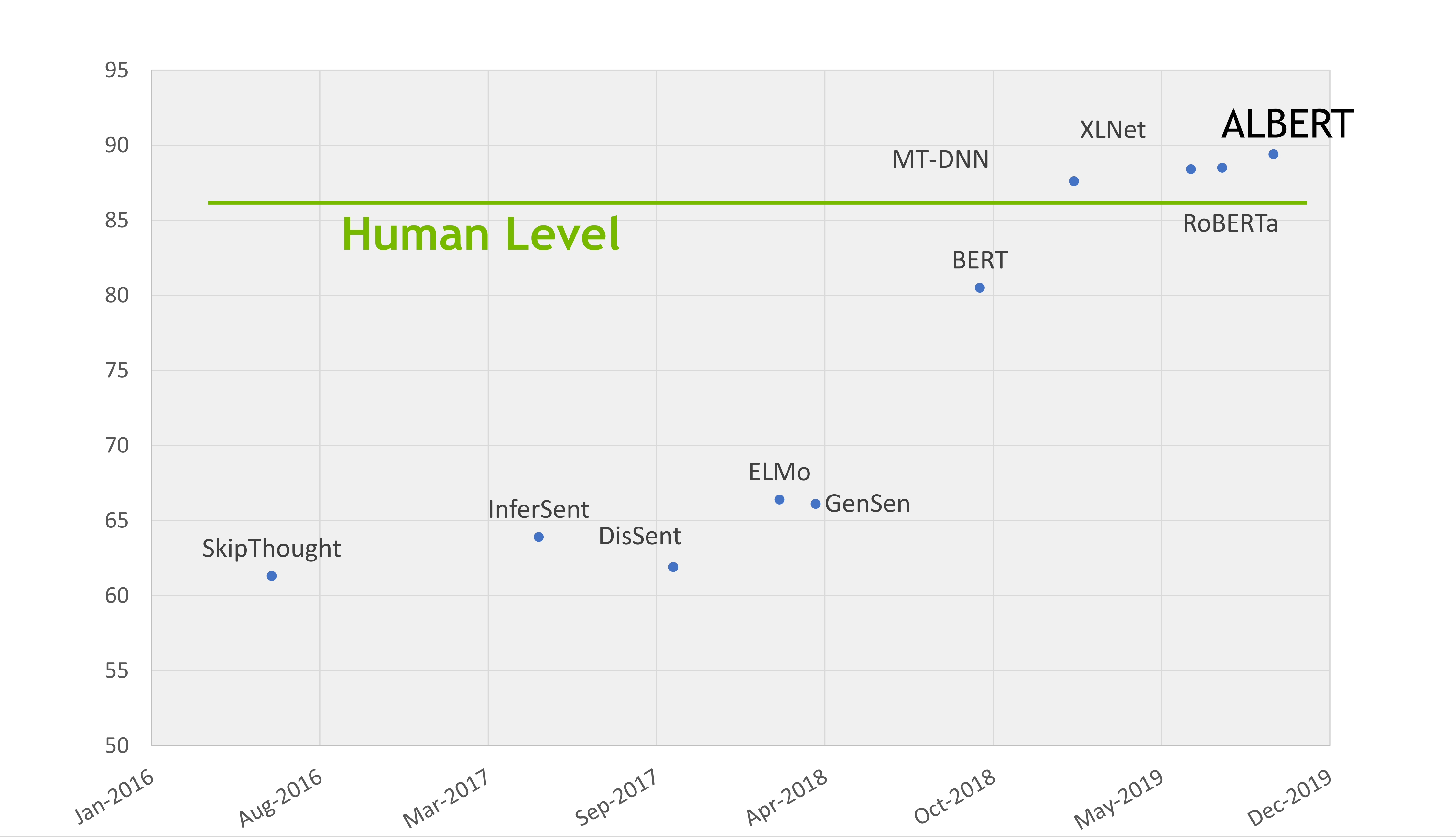
Predict if movie review is positive or negative

Decide if an abstract correctly summarizes an article

Sentence-level Semantic equivalence

Basic reading comprehension

Pronoun disambiguation



https://gluebenchmark.com/

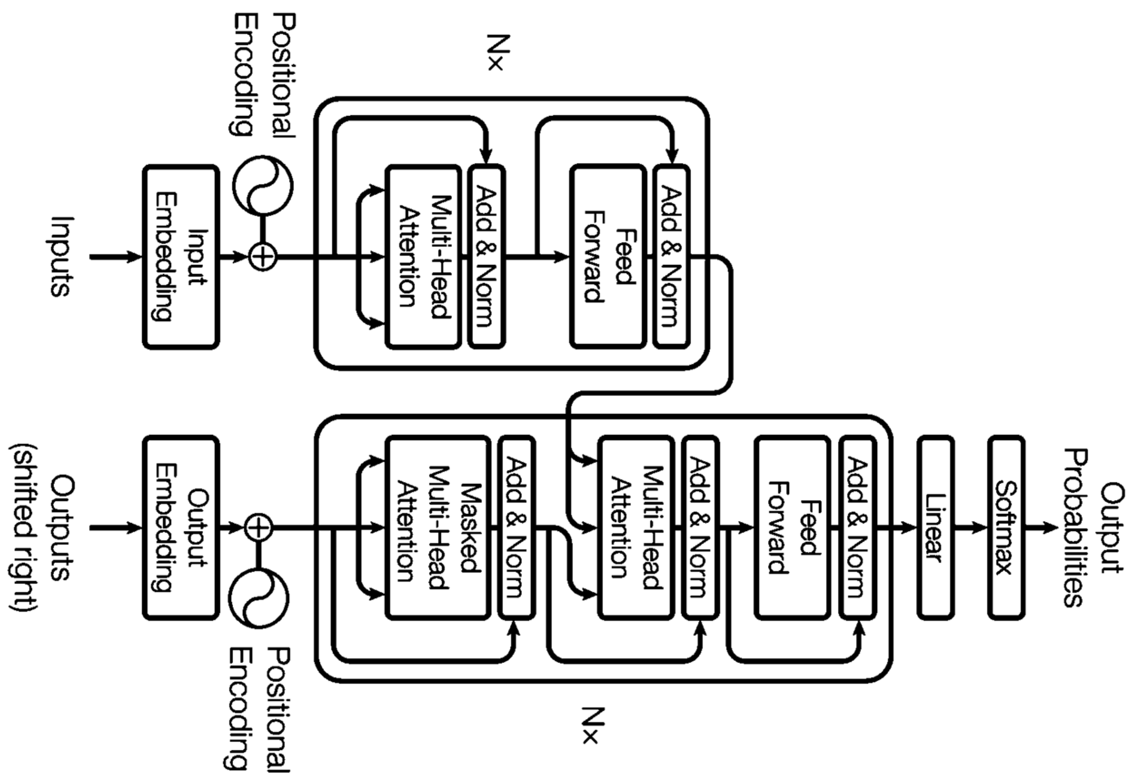
NATURAL LANGUAGE UNDERSTANDING

BERT universal language model

Input: Two sentences with 15% of words masked out

1 = “Initially he supported himself and his [redacted] by farming on a plot [redacted] family land.”

2 = “[redacted] in turn attracted the attention of [redacted] *St.* [redacted] *Post-Dispatch*, which sent a reporter to Murray to [redacted] review Stubblefield's wireless [redacted].”



Output 1: Reconstruct missing words

family, of
this, the, Louis, personally,
telephone

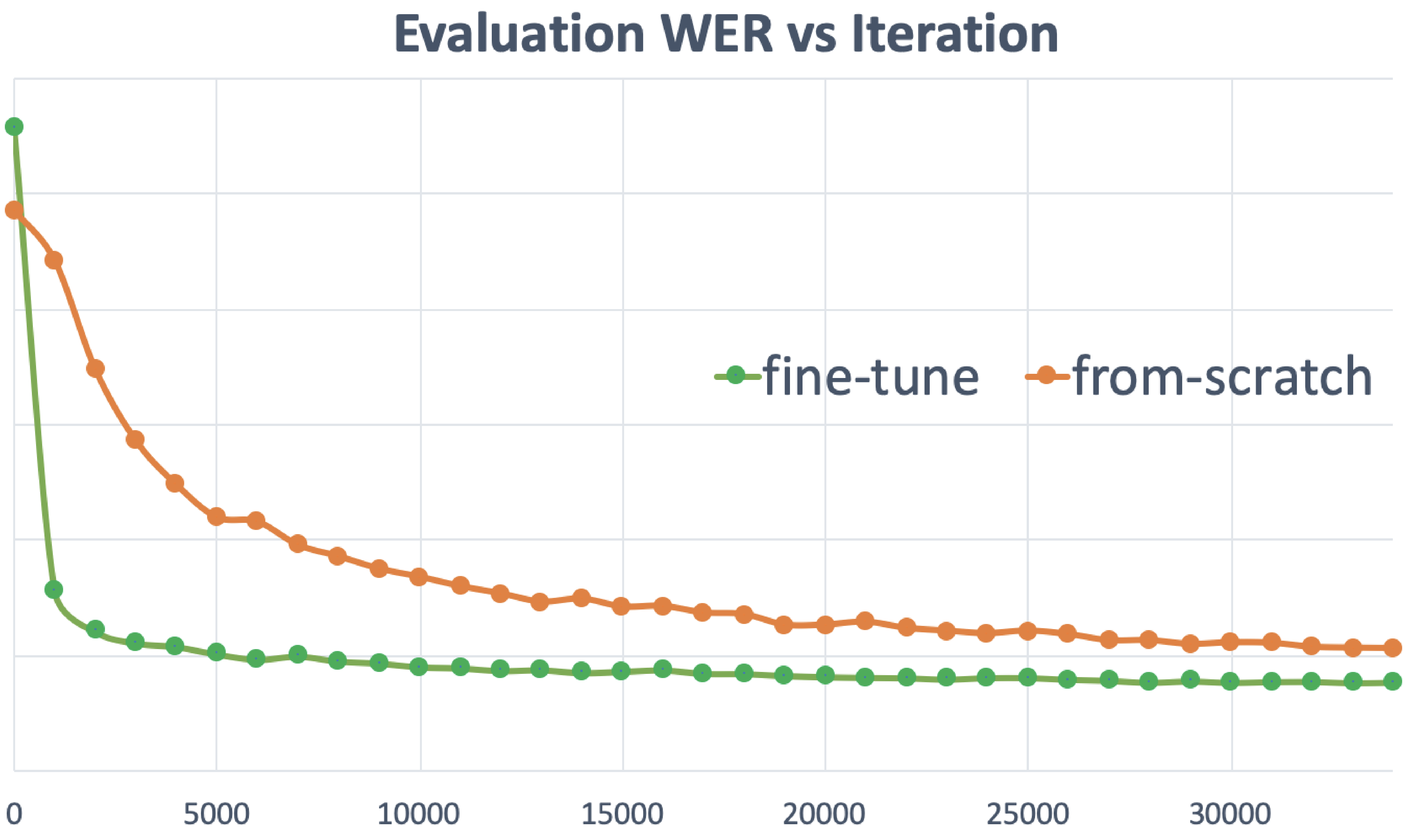
Output 2: Is two the next sentence after one?

NOT_NEXT_SENTENCE

https://arxiv.org/abs/1810.04805

THE POWER OF TRANSFER LEARNING

Domain Specific ASR - KENSHO & NVIDIA



- Jasper trained on domain specific financial data outperformed all leading ASR models

- Fine tuning was faster and had more accuracy than training from scratch

- Enables quick start and has many benefits for consulting engagements

DOMAIN SPECIFIC -- BIOBERT

Context & Specialized Knowledge Matter

NOT A STAC BENCHMARK

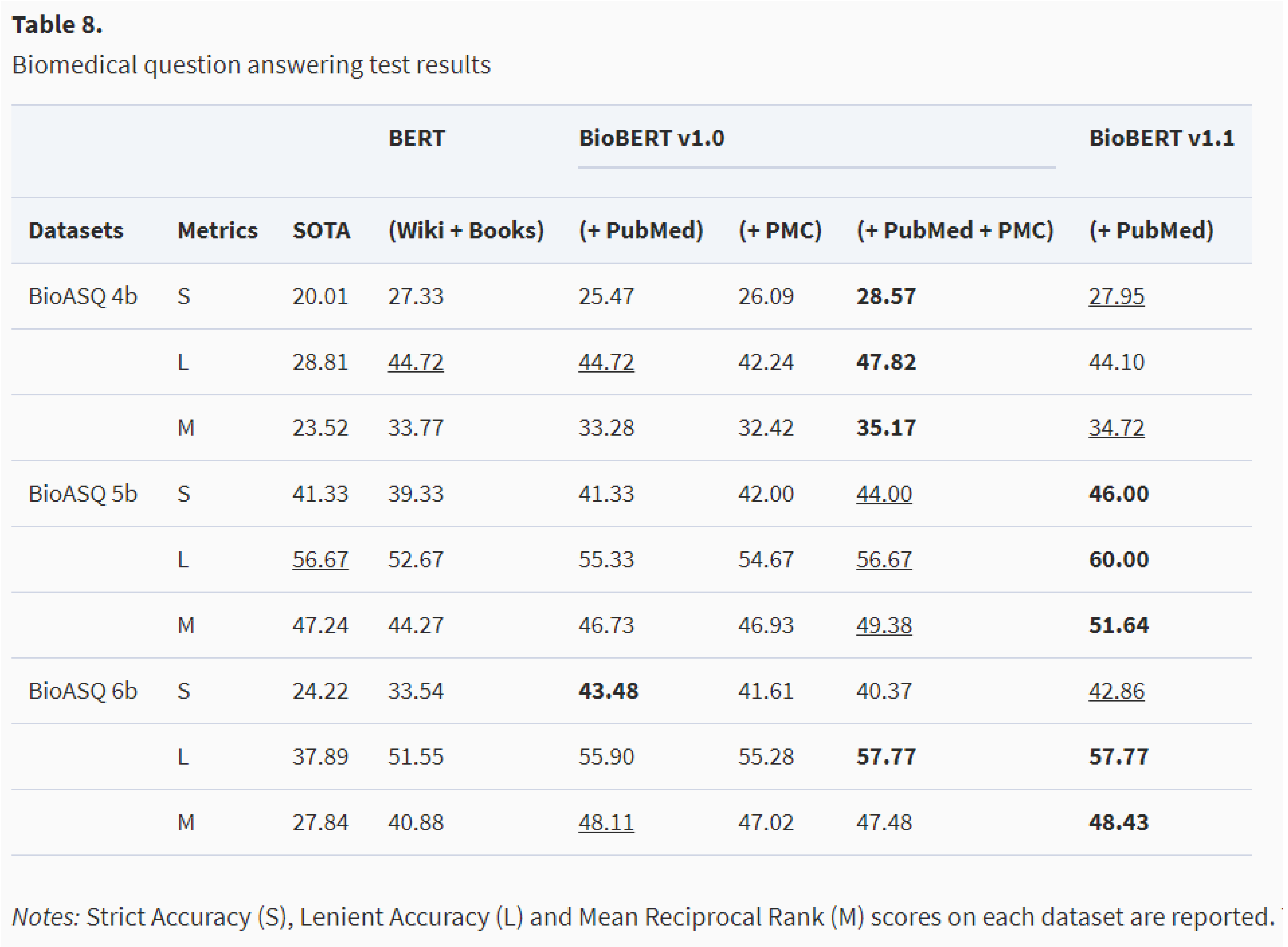
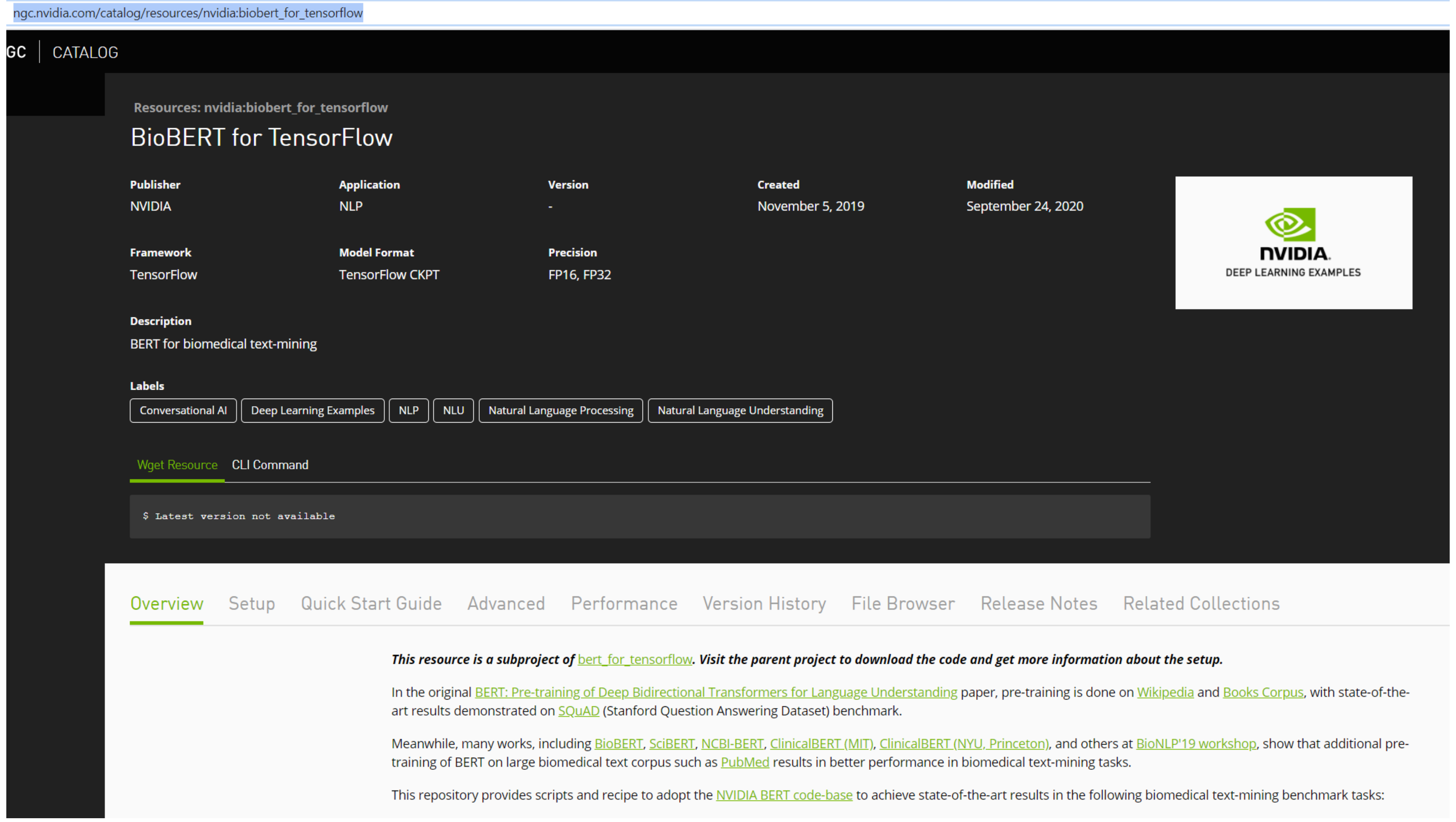


Table 8.
Biomedical question answering test results

			BERT	BioBERT v1.0			BioBERT v1.1
Datasets	**Metrics**	**SOTA**	**(Wiki + Books)**	**(+ PubMed)**	**(+ PMC)**	**(+ PubMed + PMC)**	**(+ PubMed)**
BioASQ 4b	S	20.01	27.33	25.47	26.09	**28.57**	27.95
	L	28.81	44.72	44.72	42.24	**47.82**	44.10
	M	23.52	33.77	33.28	32.42	**35.17**	34.72
BioASQ 5b	S	41.33	39.33	41.33	42.00	44.00	**46.00**
	L	56.67	52.67	55.33	54.67	56.67	**60.00**
	M	47.24	44.27	46.73	46.93	49.38	**51.64**
BioASQ 6b	S	24.22	33.54	**43.48**	41.61	40.37	42.86
	L	37.89	51.55	55.90	55.28	**57.77**	**57.77**
	M	27.84	40.88	48.11	47.02	47.48	**48.43**

Notes: Strict Accuracy (S), Lenient Accuracy (L) and Mean Reciprocal Rank (M) scores on each dataset are reported.



ngc.nvidia.com/catalog/resources/nvidia:biobert_for_tensorflow

GC | CATALOG

Resources: nvidia:biobert_for_tensorflow

BioBERT for TensorFlow

Publisher	Application	Version	Created	Modified
NVIDIA	NLP	-	November 5, 2019	September 24, 2020

Framework	Model Format	Precision
TensorFlow	TensorFlow CKPT	FP16, FP32

Description
BERT for biomedical text-mining

Labels
Conversational AI | Deep Learning Examples | NLP | NLU | Natural Language Processing | Natural Language Understanding

```
$ Latest version not available
```

This resource is a subproject of bert_for_tensorflow. Visit the parent project to download the code and get more information about the setup.

In the original BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper, pre-training is done on Wikipedia and Books Corpus, with state-of-the-art results demonstrated on SQuAD (Stanford Question Answering Dataset) benchmark.

Meanwhile, many works, including BioBERT, SciBERT, NCBI-BERT, ClinicalBERT (MIT), ClinicalBERT (NYU, Princeton), and others at BioNLP'19 workshop, show that additional pre-training of BERT on large biomedical text corpus such as PubMed results in better performance in biomedical text-mining tasks.

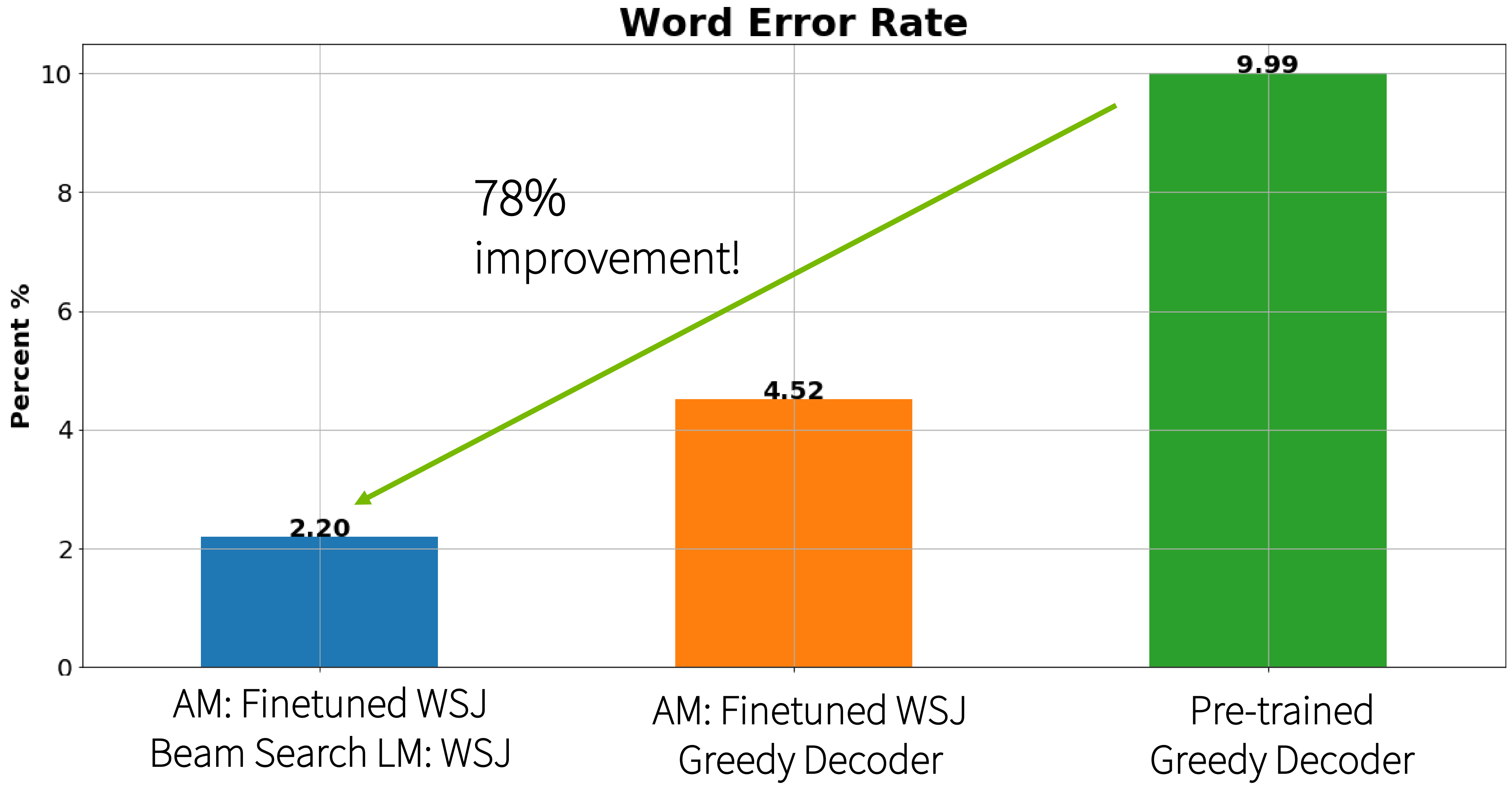
This repository provides scripts and recipe to adopt the NVIDIA BERT code-base to achieve state-of-the-art results in the following biomedical text-mining benchmark tasks:

https://ngc.nvidia.com/catalog/resources/nvidia:biobert_for_tensorflow

HOW TO BUILD YOUR OWN DOMAIN SPECIFIC ASR MODELS

https://ngc.nvidia.com/catalog/containers/nvidia:**nemo_asr_app_img**

Pre-trained Quartznet model LibriSpeech (old fiction books) → **Finetune Acoustic Model** with Wall Street Journal data (modern business news) → **Train Language Model** with Wall Street Journal → **Compare Models'** performance → **Export model** for Deployment



NOT A STAC BENCHMARK

“Simplify, then add lightness.” – Colin Chapman

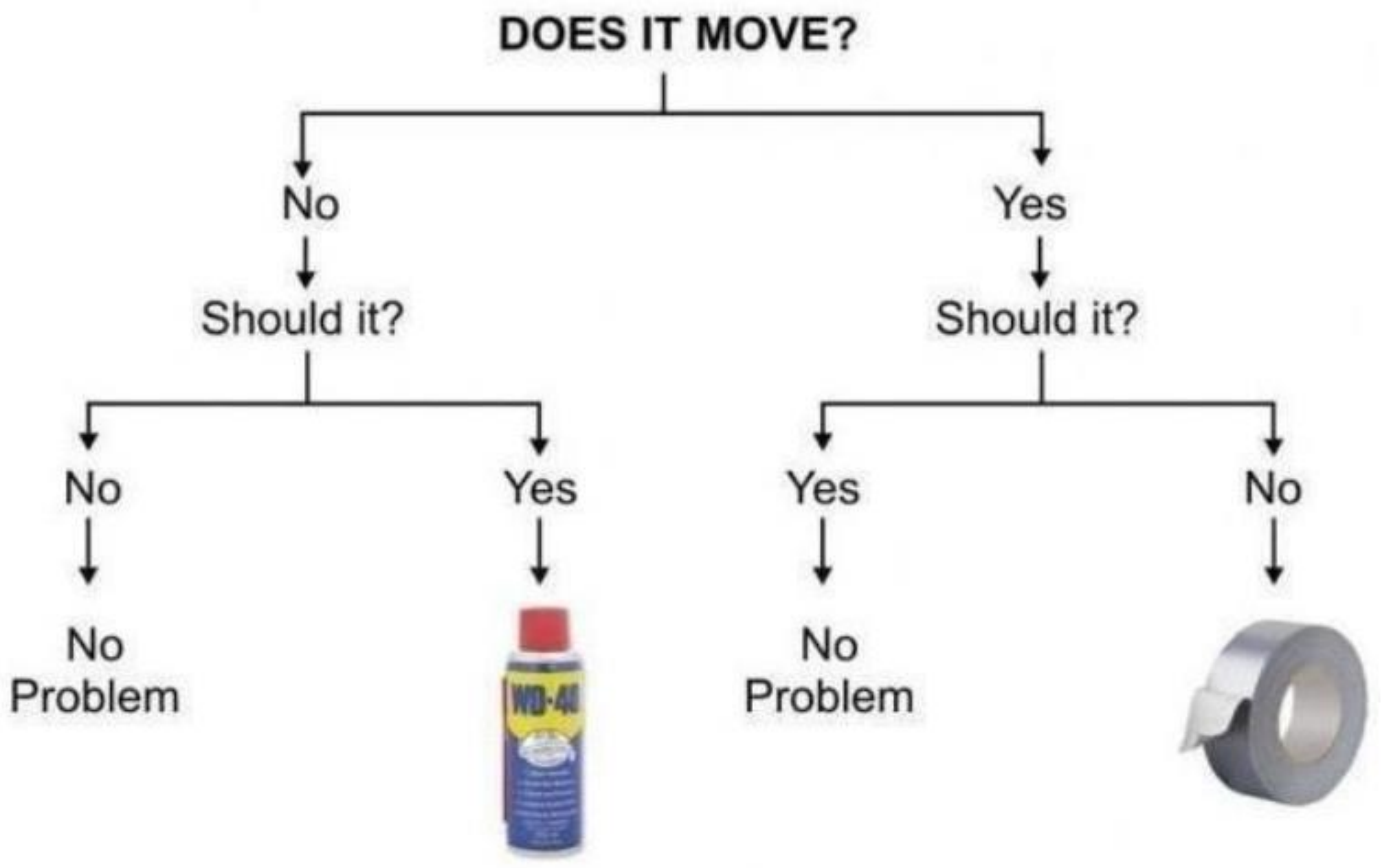
FASTER

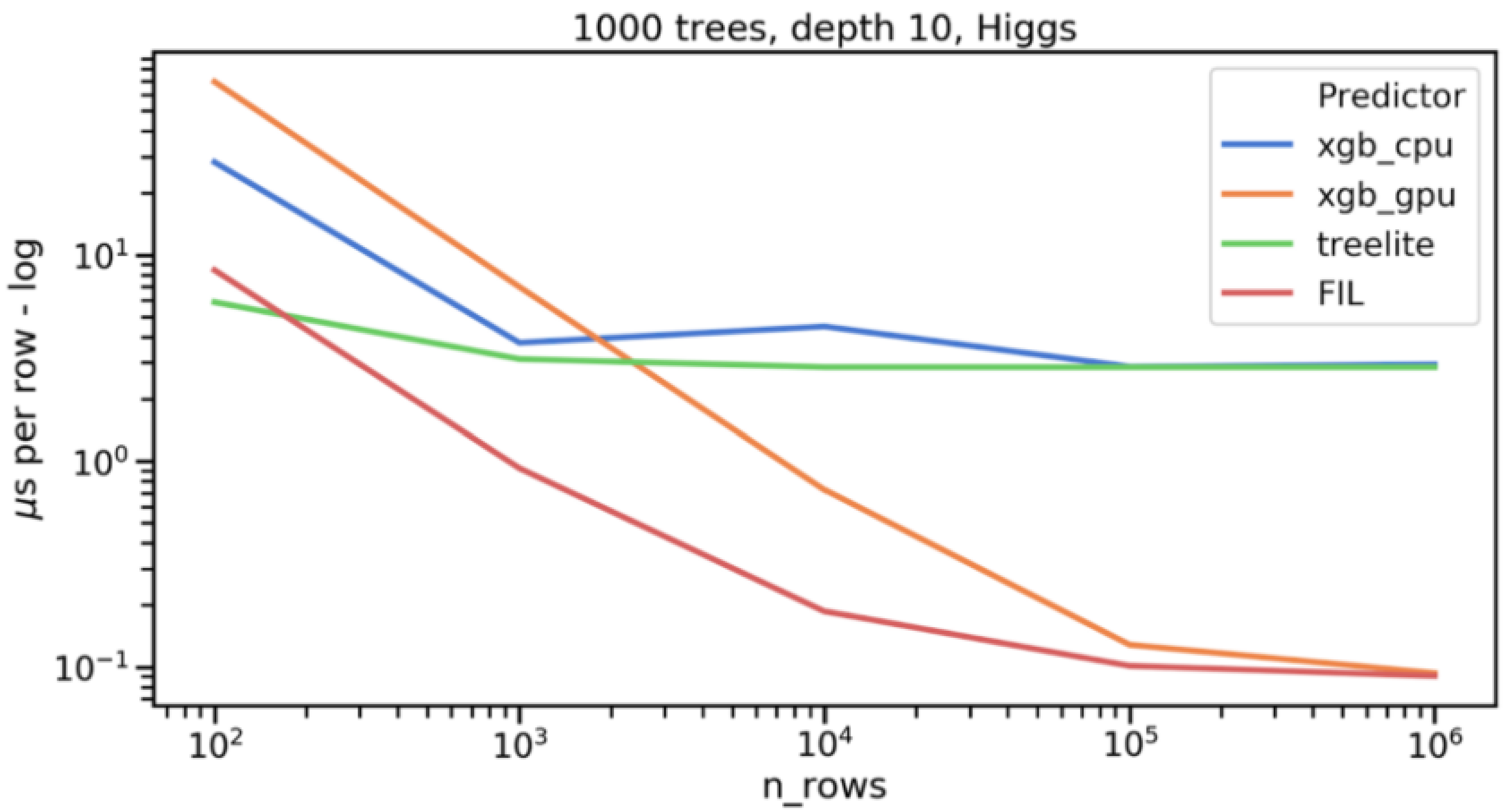
By IFCAR - Own work, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=18245654

NOT A STAC BENCHMARK

FASTER TREES

Forest Inference Library





Inference time scaling as batch size increases

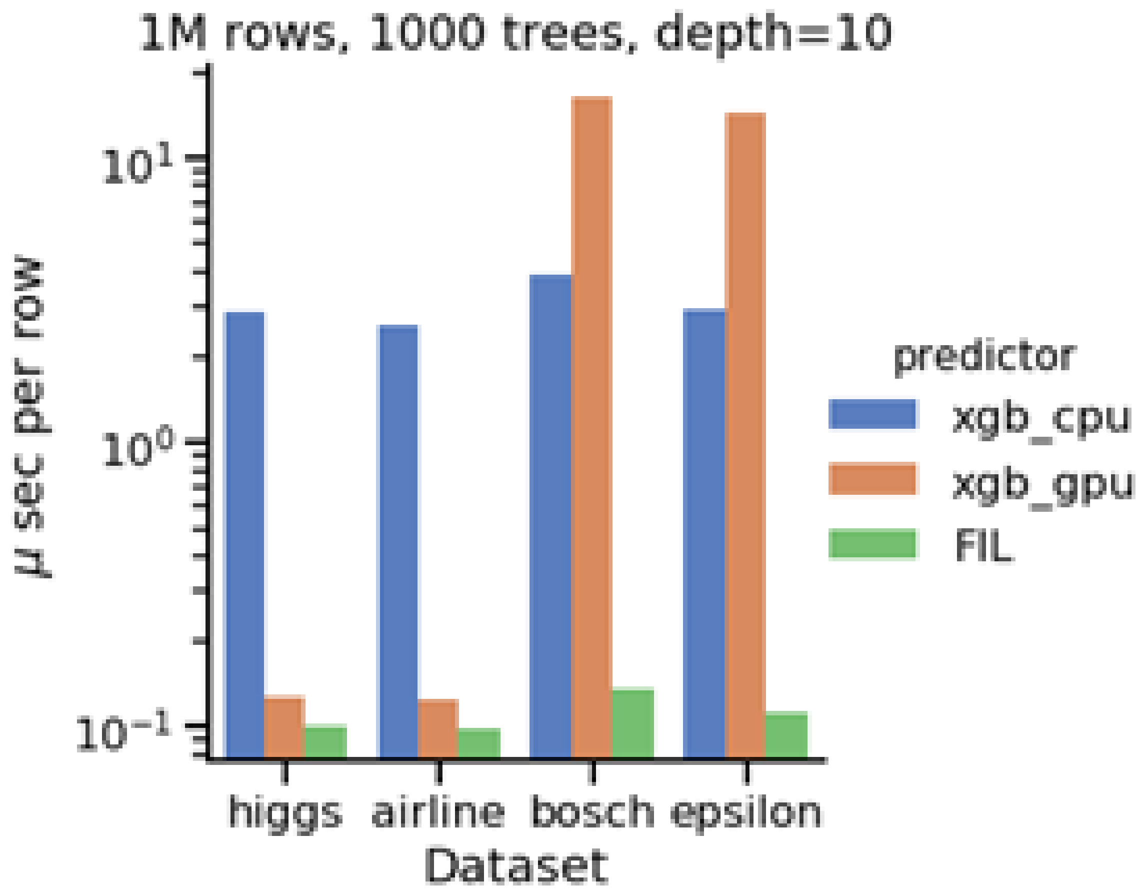
https://medium.com/rapids-ai/rapids-forest-inference-library-prediction-at-100-million-rows-per-second-19558890bc35

NVIDIA Fraud Detection Example Using PaySim Dataset

Accelerating Inferencing Using Forest Inferencing Library (FIL) on NVIDIA GPUs

NOT A STAC BENCHMARK

35X Faster Using FIL



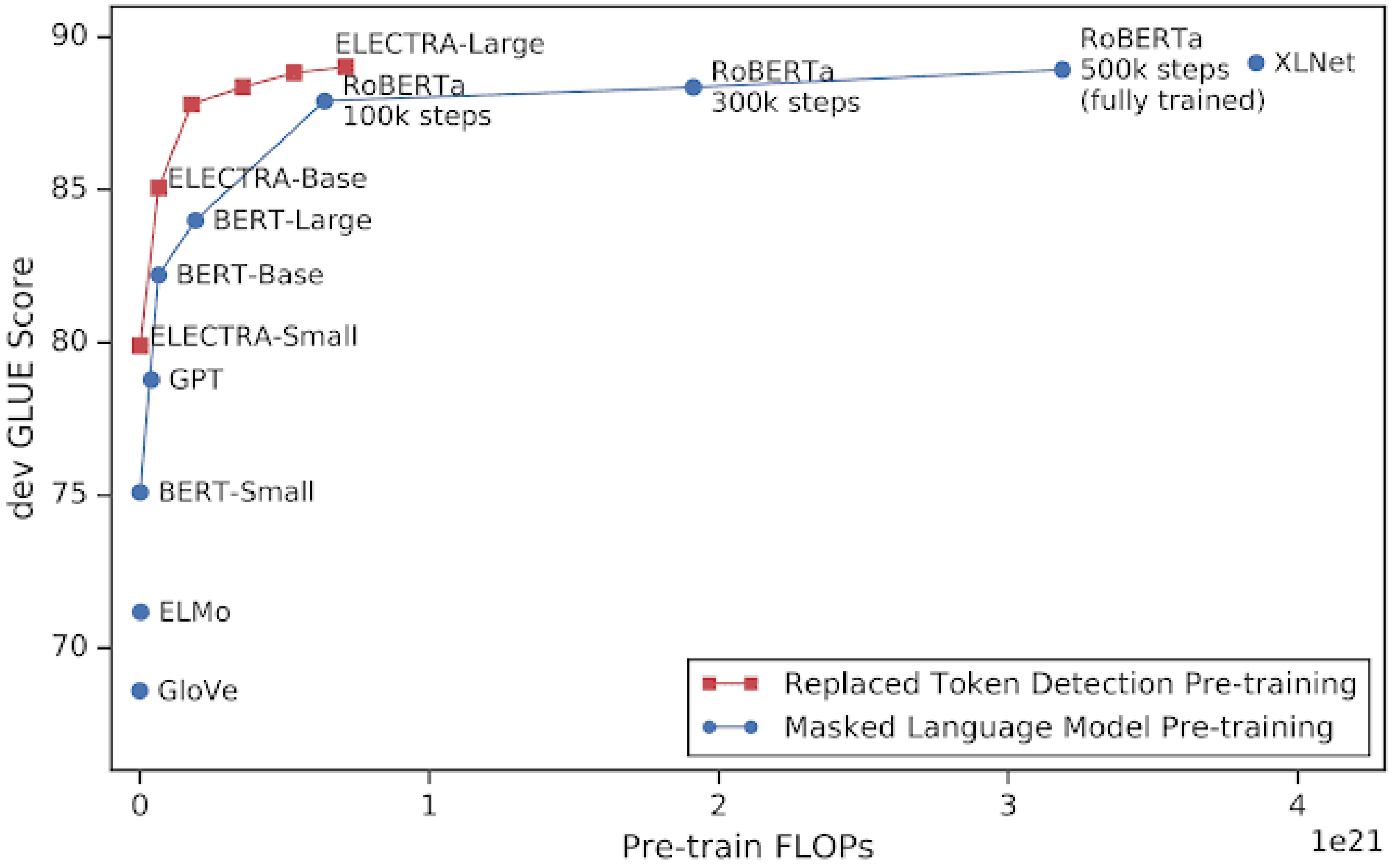
CPU and GPU performance across datasets

Image credit: https://medium.com/rapids-ai/rapids-forest-inference-library-prediction-at-100-million-rows-per-second-19558890bc35

```
from cuml import ForestInference
# Load the classifier previously saved with xgboost
# model_save()
import sklearn.datasets
model_path = 'xgb.model’

# Generate random sample data
fm = ForestInference.load(model_path,
output_class=True)

# Generate predictions (as a gpu array)
X_test, y_test =
sklearn.datasets.make_classification()

fil_preds_gpu = fm.predict(X_test.astype('float32'))
```

NLP MODELS ARE LARGE

The Inference cost is high

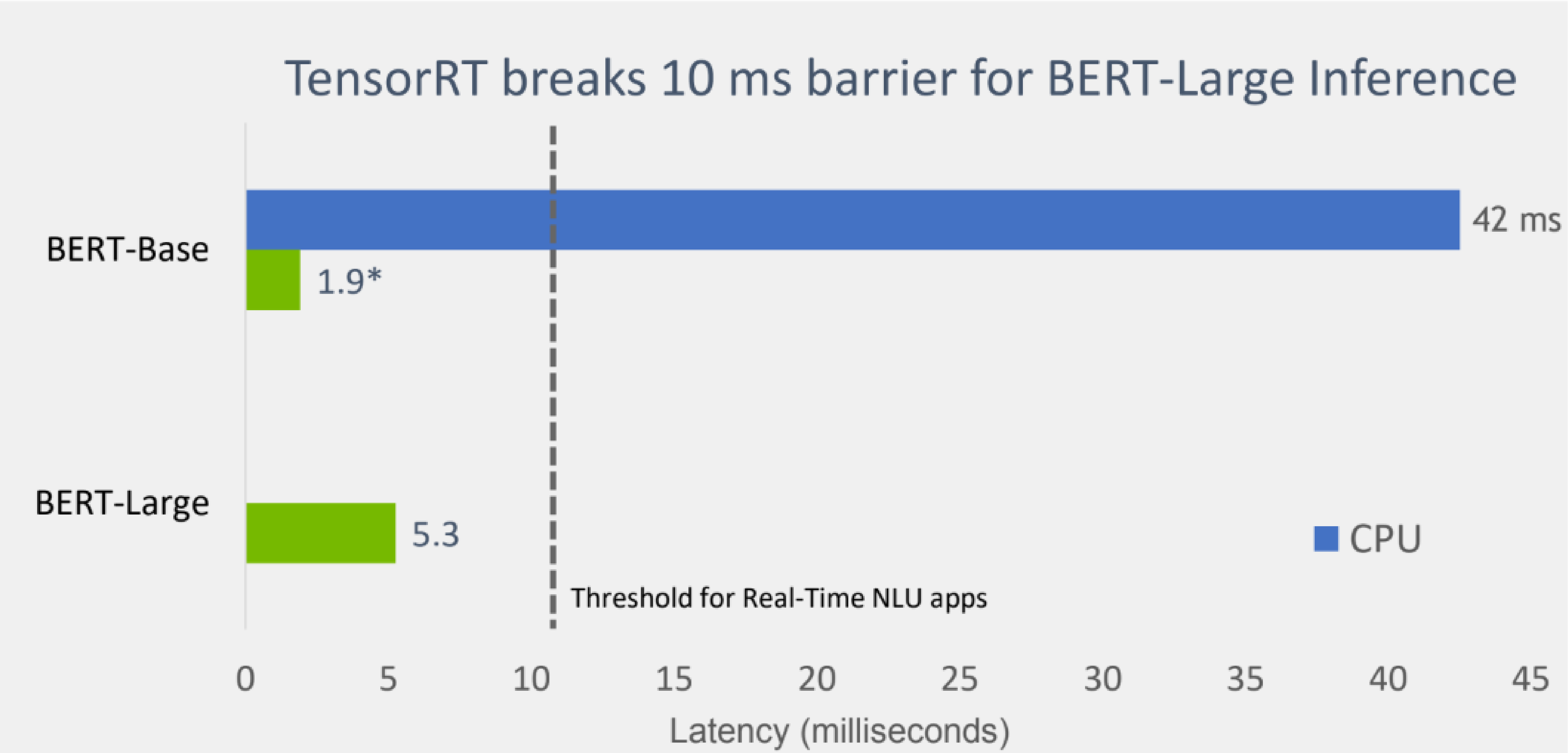


NOT A STAC BENCHMARK

https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

BERT-LARGE INFERENCE IN 5.3 ms

Makes Real-Time Natural Language Understanding Possible



TensorRT breaks 10 ms barrier for BERT-Large Inference

NOT A STAC BENCHMARK

* In our tests, OpenVINO Release 2019 R2 did not execute BERT-Large and exited with an error

[BERT Sample Code in TensorRT Repo]

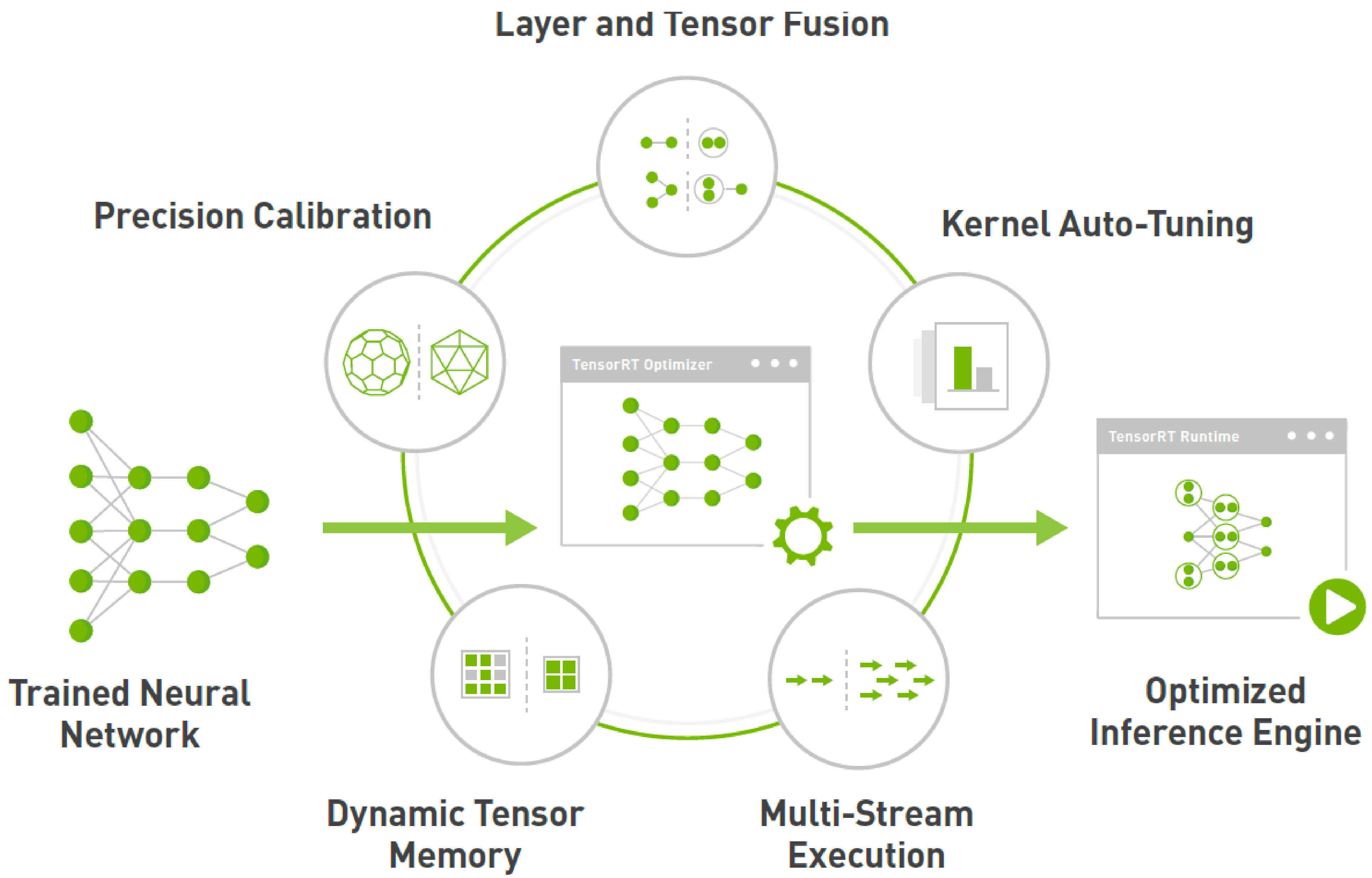
[Jupyter Python Notebook]

[Blog: Real-Time Natural Language Understanding with BERT Using TensorRT]

developer.nvidia.com/tensorrt

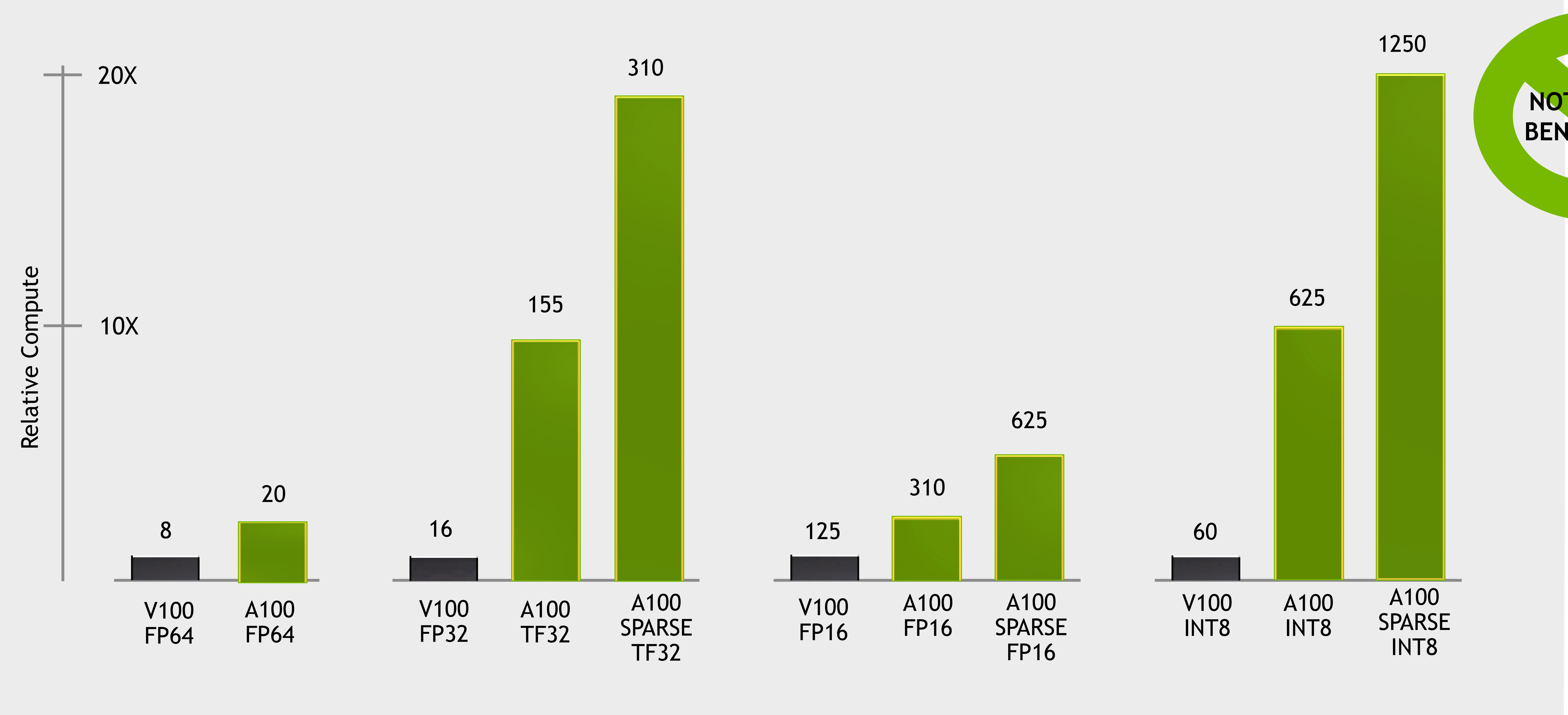
TENSORRT

Simplification and the addition of lightness.



INCREASING IMPORTANCE OF PRUNING AND QUANTIZATION

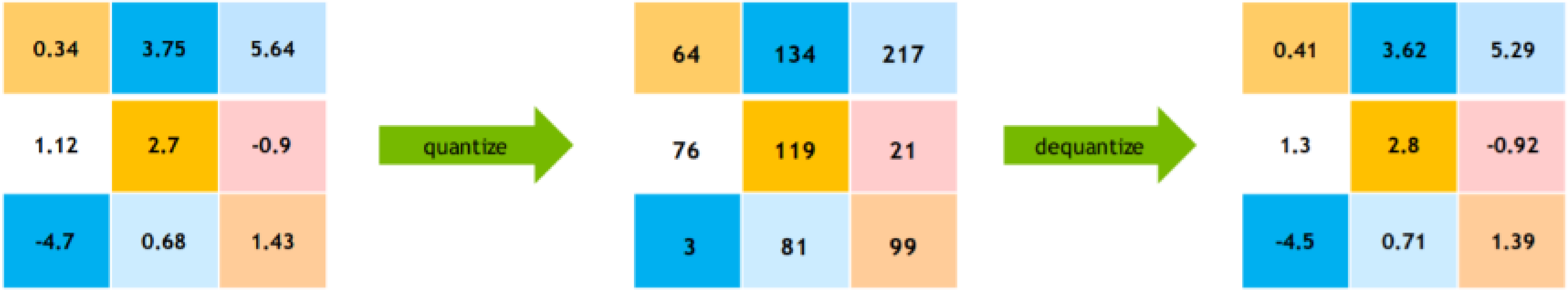
Hardware acceleration for reduced precision arithmetic and sparsity



NOT A STAC BENCHMARK

QUANTIZATION

The idea



0.34	3.75	5.64
1.12	2.7	-0.9
-4.7	0.68	1.43

FP32
(pre-quantized)

64	134	217
76	119	21
3	81	99

INT8
(quantized)

0.41	3.62	5.29
1.3	2.8	-0.92
-4.5	0.71	1.39

FP32
(dequantized)

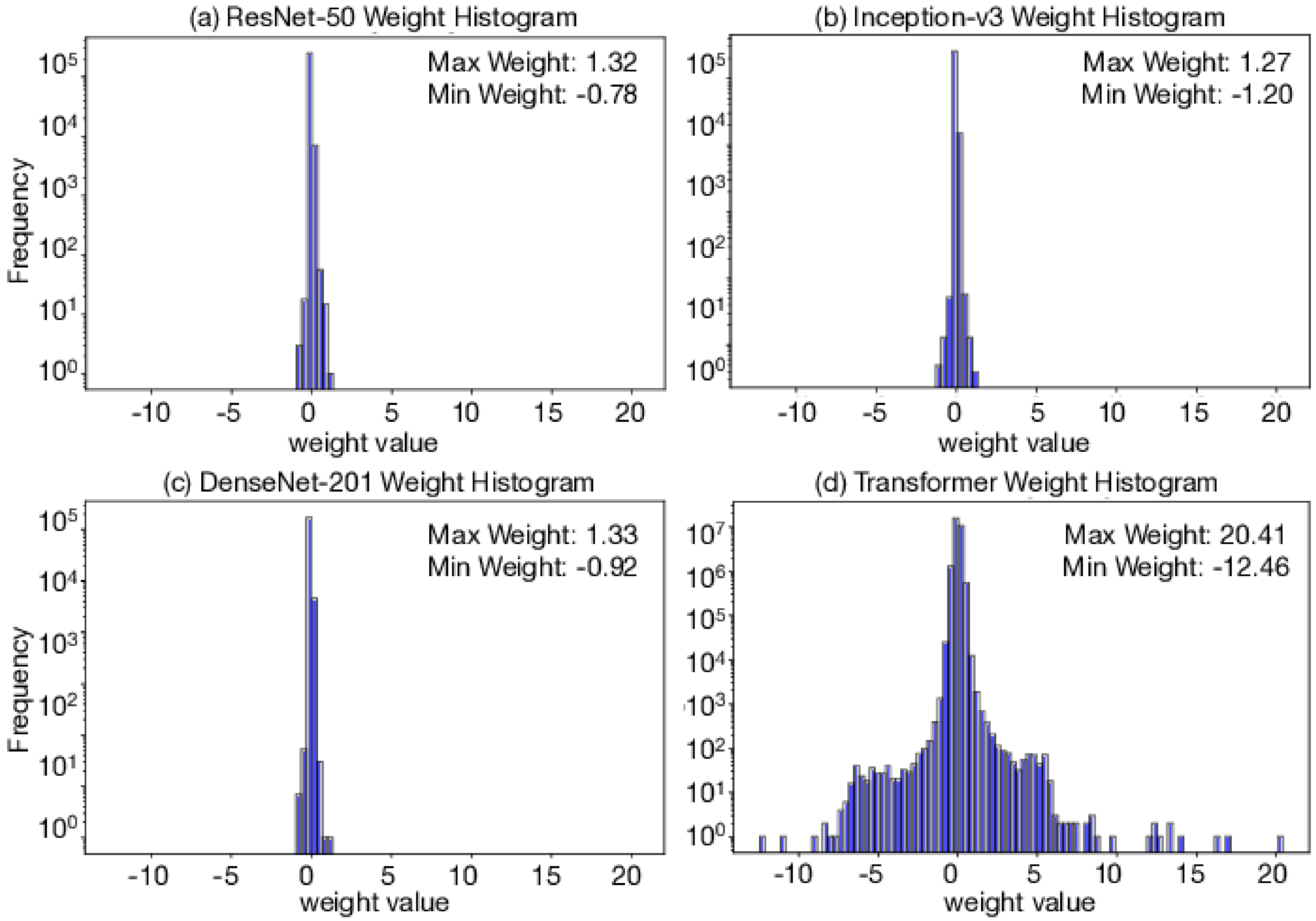
PRUNING

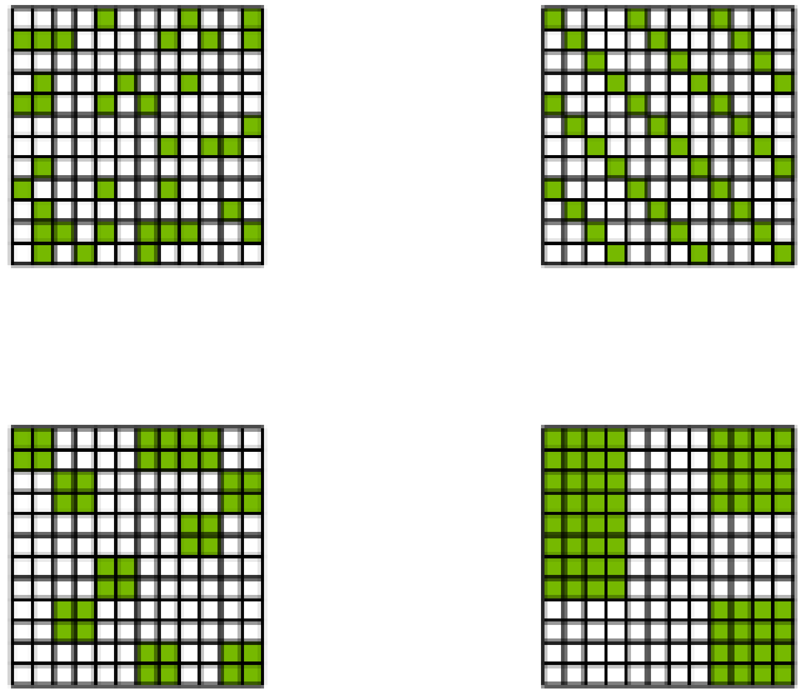
The idea

NOT A STAC BENCHMARK

The opportunity:

- Reduced memory bandwidth
- Reduced memory footprint
- Acceleration (especially in presence of hardware acceleration)

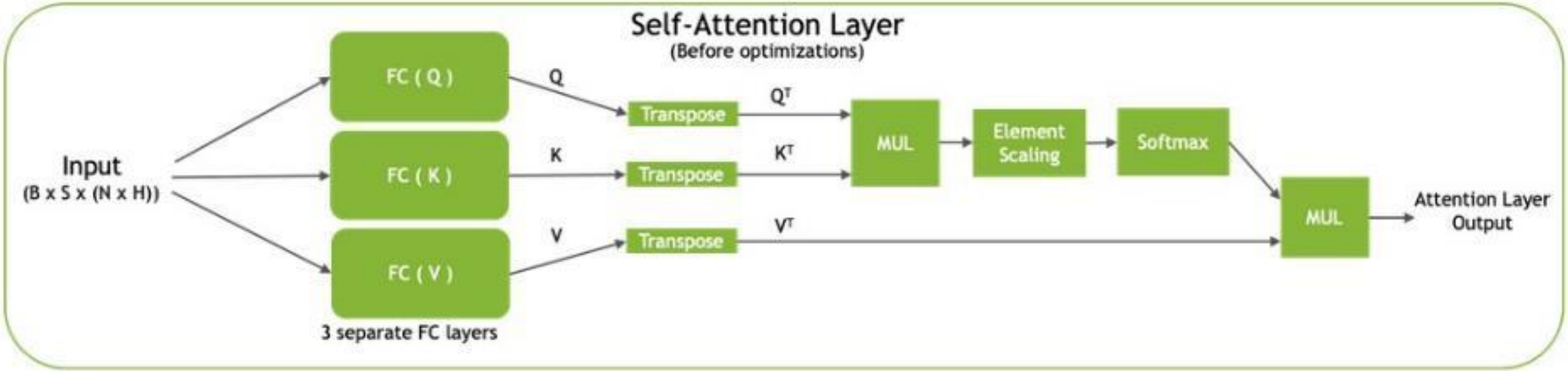


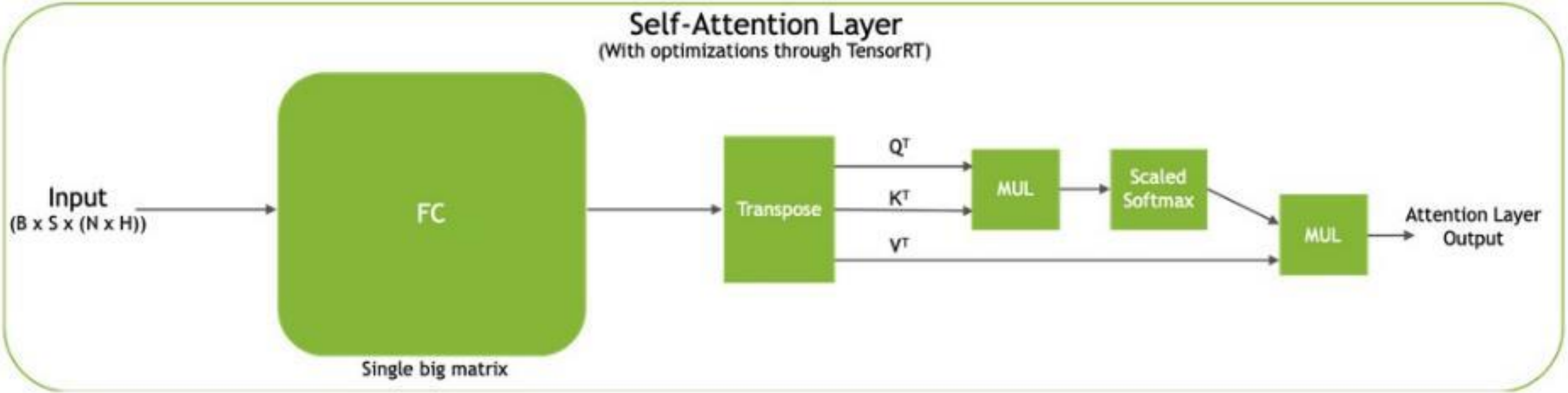


Tambe, T., Yang, E. Y., Wan, Z., Deng, Y., Reddi, V. J., Rush, A., ... & Wei, G. Y. (2019). AdaptivFloat: A Floating-point based Data Type for Resilient Deep Learning Inference. *arXiv preprint arXiv:1909.13271*.

CUSTOM PLUGINS

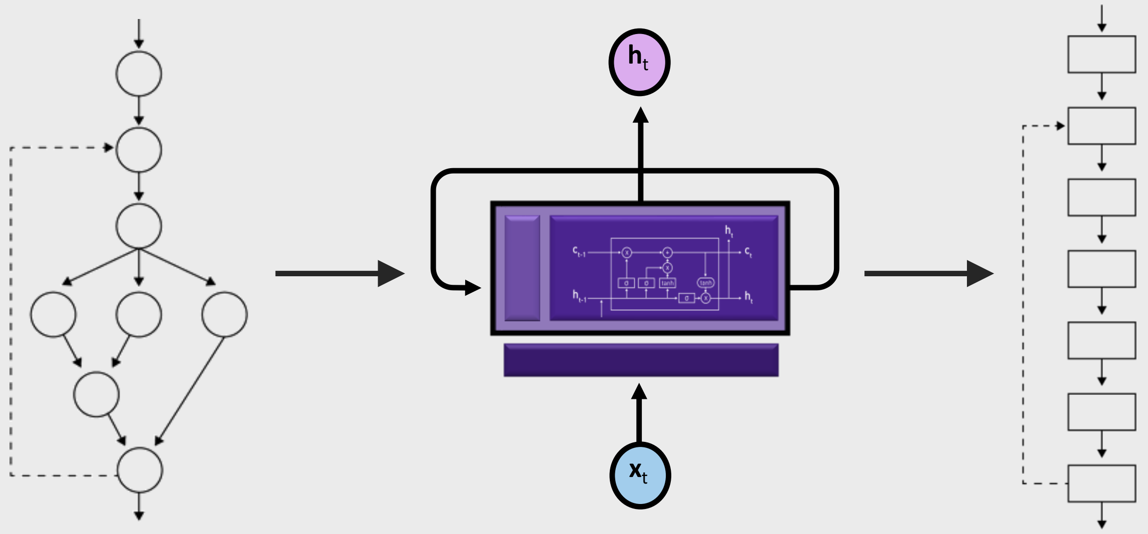
Self-attention layer



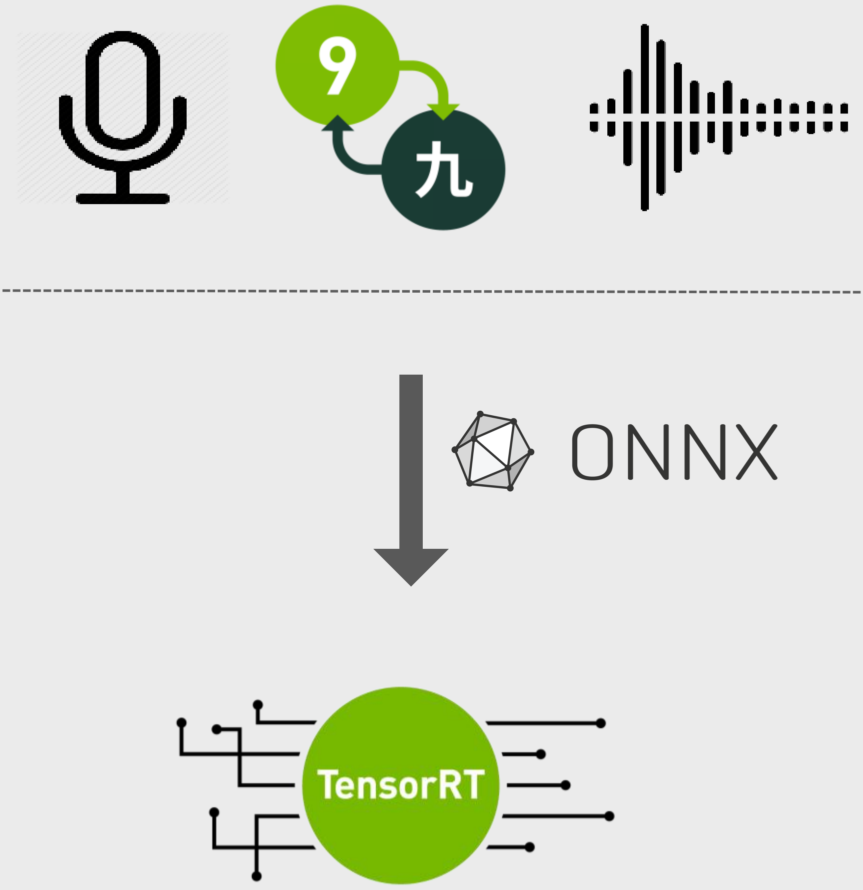


TensorRT 7 FAMILY

ASR, NLU & TTS | 1000+ Kernels | FP32, FP16, INT8



Compiler Supports RNNs, Transformers and CNNs



20+ ONNX Ops & Dynamic Shapes Enhancements Accelerating Speech

[ASR With Jasper Example]

[NLU With BERT Example]

[TTS With Tacotron 2+Waveglow Blog & Example]

Get Started with ASR, NLU, TTS Today

TensorRT ONNX PARSER

High-Performance Inference for ONNX Models

Optimize and deploy models from ONNX-supported frameworks to production

Apply TensorRT optimizations to any ONNX framework (Caffe 2, Microsoft Cognitive Toolkit, MxNet & PyTorch)

Import TensorFlow and Keras through converters (tf2onnx, keras2onnx)

Use with C++ and Python apps

20+ New Ops in TensorRT 7

Support for Opset 11 (See List of Supported Ops)

ONNX

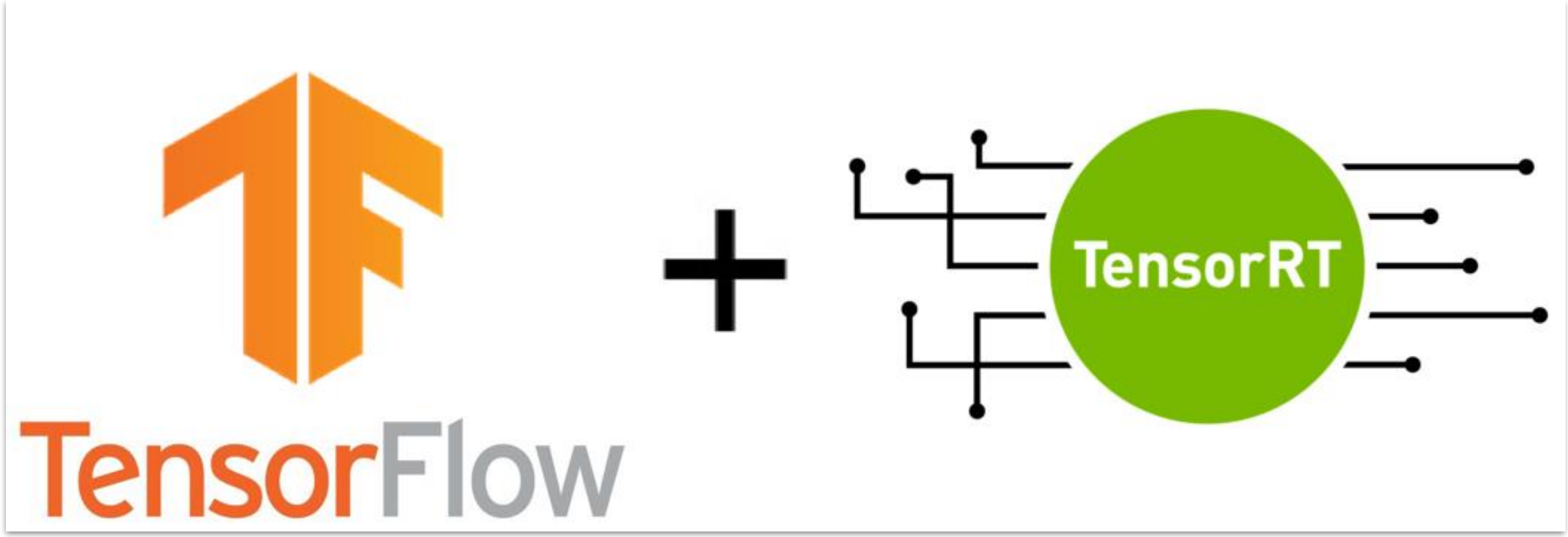
Caffe2 Chainer Cognitive Toolkit

mxnet MATLAB PyTorch

PaddlePaddle

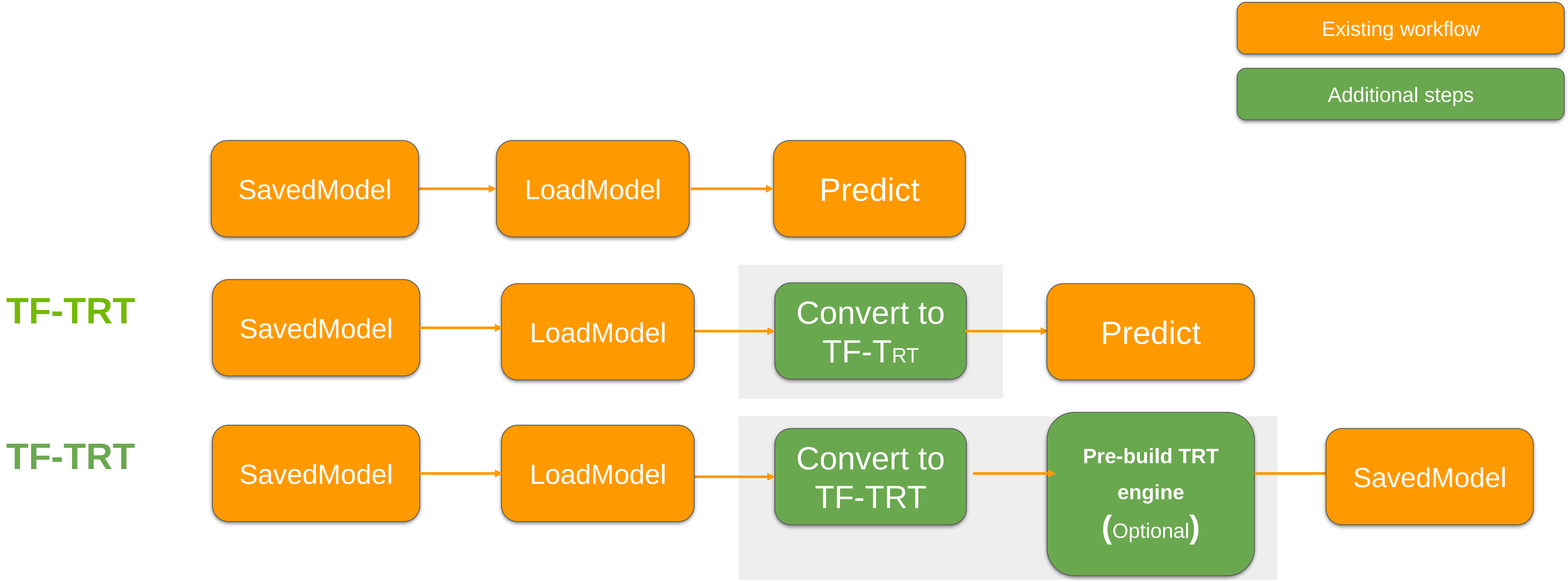
TF-TRT = TF +TRT

Optimize TF inference while still using the TF ecosystem



HOW TO USE?

TF-TRT 2.x Workflow



Existing workflow

Additional steps

SavedModel → LoadModel → Predict

TF-TRT: SavedModel → LoadModel → Convert to TF-TRT → Predict

TF-TRT: SavedModel → LoadModel → Convert to TF-TRT → Pre-build TRT engine (Optional) → SavedModel

INT8 TF-TRT API IN TENSORFLOW 2.0

TF-TRT API

```
from tensorflow.python.compiler.tensorrt import trt_convert as trt
conversion_params = trt.TrtConversionParams(
    precision_mode=trt.TrtPrecisionMode.INT8)

converter = trt.TrtGraphConverterV2(
    input_saved_model_dir=input_saved_model_dir,
    conversion_params=conversion_params)

converter.convert(calibration_input_fn=my_input_fn)

#optionally build TRT engines before deployment
converter.build(input_fn=my_input_fn)

converter.save(output_saved_model_dir)
```

Jupyter notebook example: https://github.com/tensorflow/tensorrt/blob/master/tftrt/examples/image-classification/TF-TRT-inference-from-saved-model.ipynb

CONTINUOUS PERFORMANCE IMPROVEMENT

Developers’ Software Optimizations Deliver Better Performance on the Same Hardware

Monthly DL Framework Updates & Stack Optimizations Drive Performance

cuDNN - Highly tuned standard training routines

cuBLAS - Highly tuned matrix multiplication

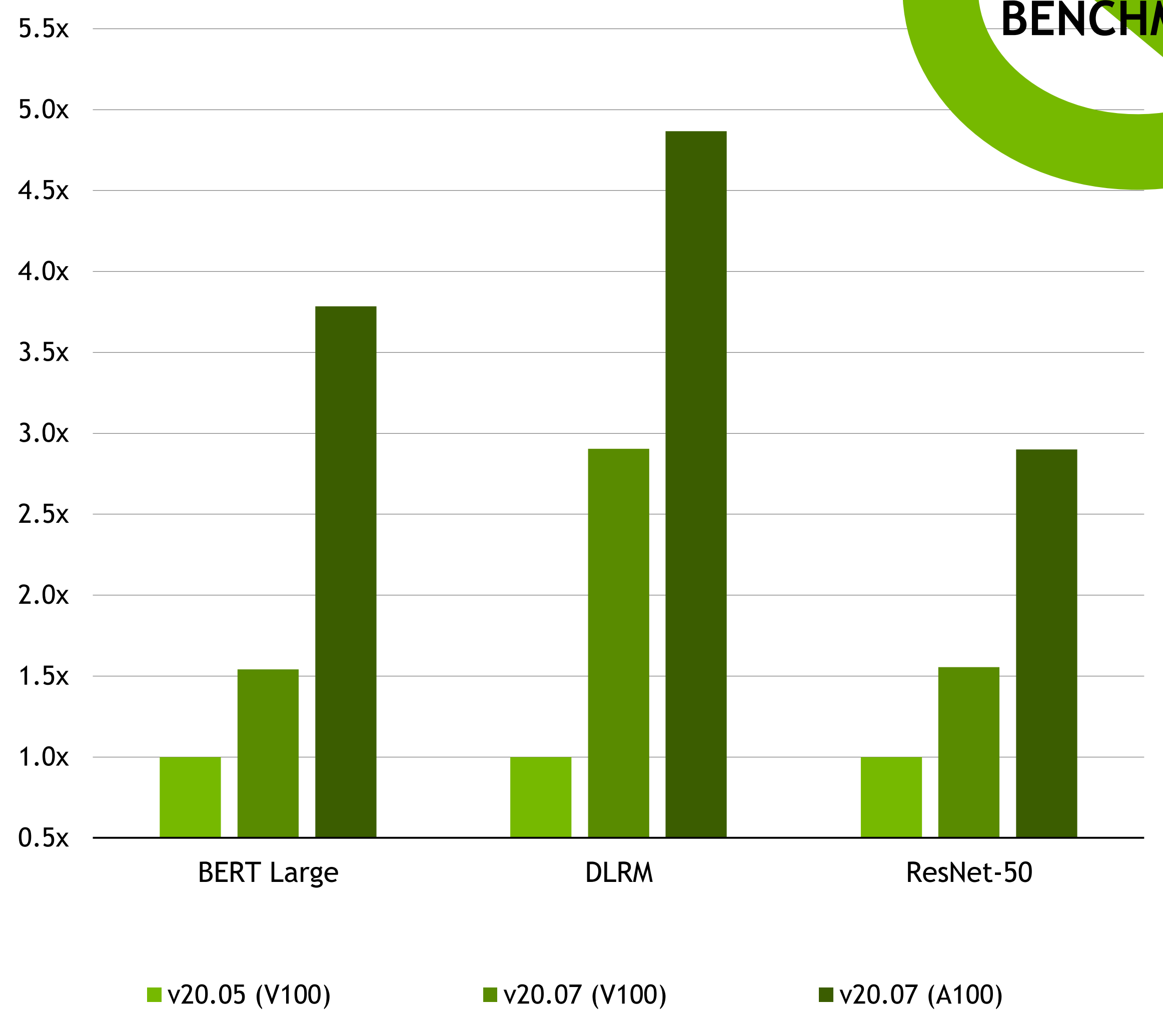
DALI – Moves compute intensive pre-processing to GPUs

NCCL – Faster training across multi-GPU architecture

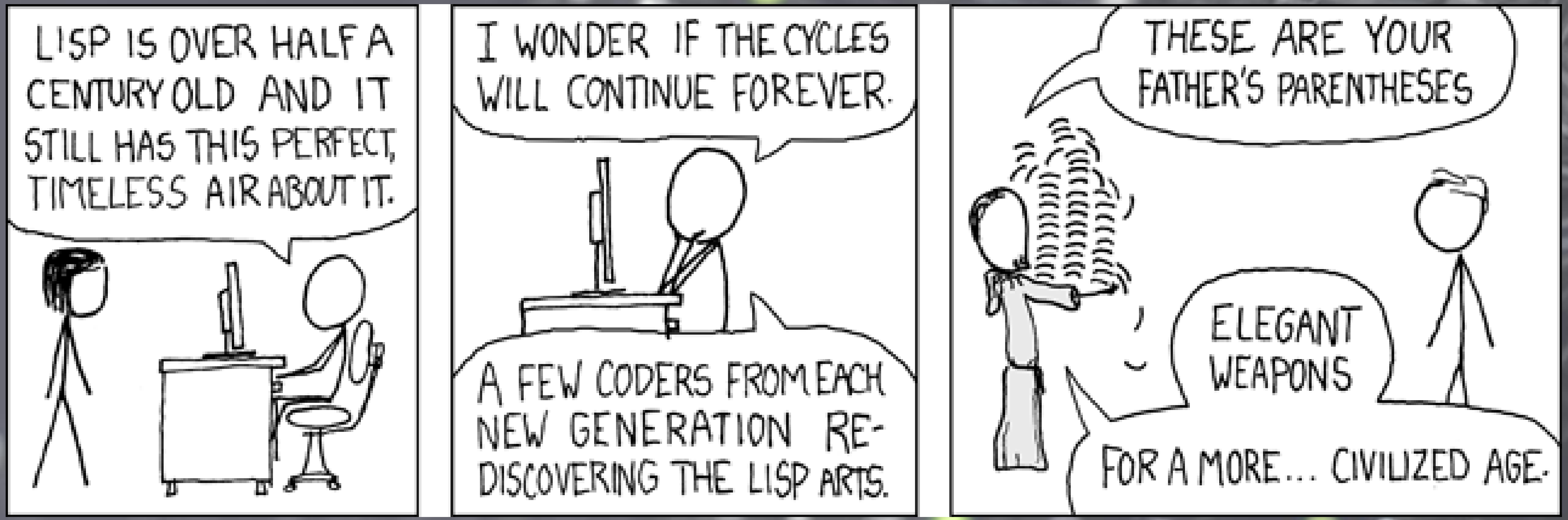
Framework – Latest versions w/ newest features and superior perf

MONTHLY UPDATES DELIVER FASTER TRAINING PERFORMANCE

NOT A STAC BENCHMARK



BERT-Large and ResNet-50 v1.5 Training performance with TensorFlow on a single node 8x V100 (32GB) & A100 (40GB). Mixed Precision.
Batch size for BERT: 10 (V100), 24 (A100), ResNet: 512 (V100, v20.05), 256 (v20.07)
DLRM Training performance with PyTorch on 1x V100 & 1x A100. Mixed Precision. Batch size 32768. DRLM trained with v20.03 and v20.07


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.

“Every sufficiently advanced LISP application will eventually reimplement Python.” – Hodgson’s Law

MORE PYTHONIC

CAN WE HAVE FAST DEVELOPMENT AND FAST EXECUTION?

Yes, if we leverage the whole Python ecosystem

https://rapids.ai/

OPEN SOURCE

APACHE ARROW   blazingSQL   DASK   GoAi   nuclio   Numba   scikit learn

dmlc XGBoost

CONTRIBUTORS

ANACONDA   blazingSQL   Capital One   Chainer   CuPy   DEEPWAVE DIGITAL   GUNROCK

NVIDIA   QUANSIGHT   Walmart Labs

ADOPTERS

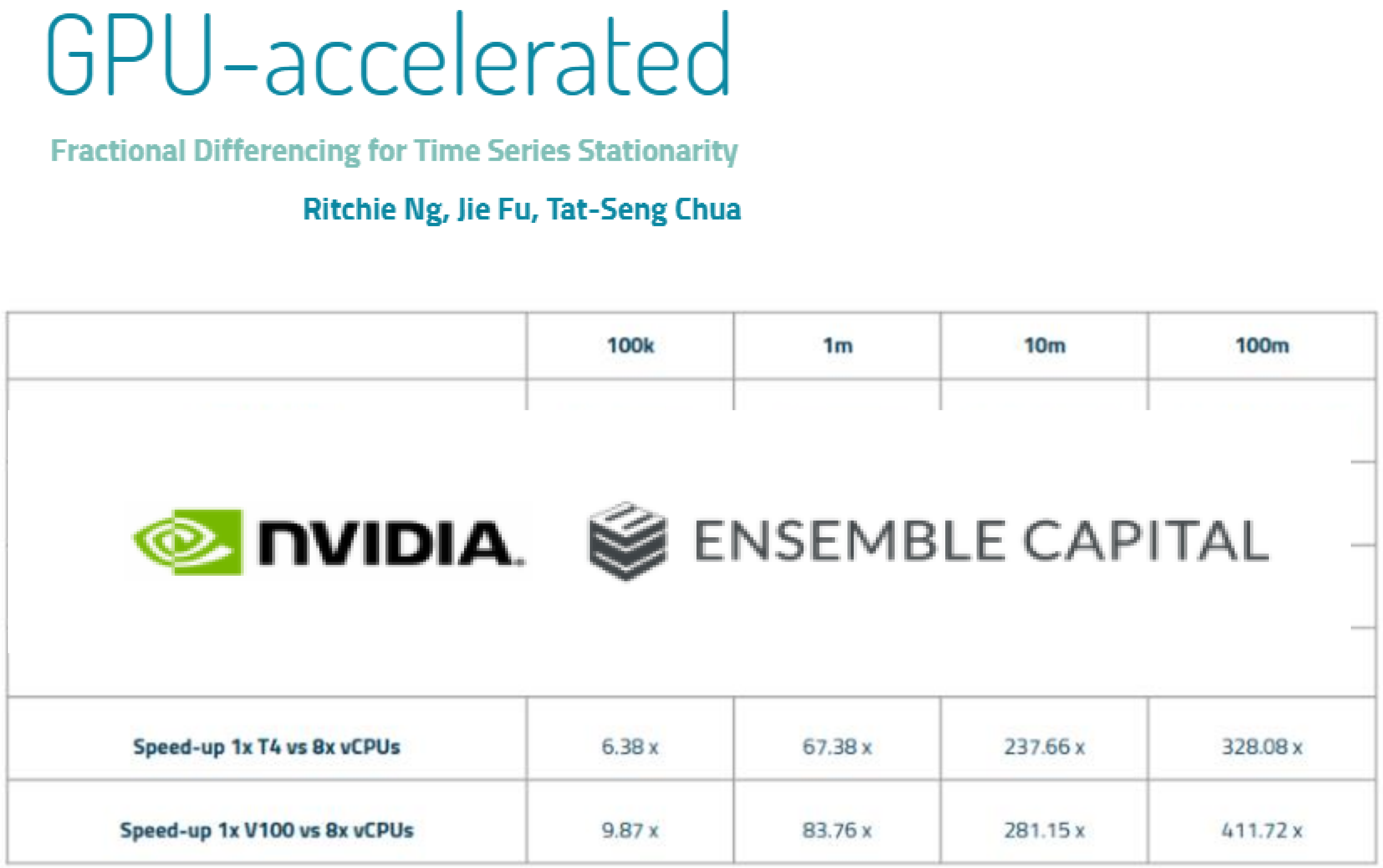
anyscale   Booz Allen   Capital One   databricks   DEEPWAVE DIGITAL   graphistry   H2O.ai

IBM   iguazio   Inria   kinetica   MAPR   omni sci   plotly | Dash

Preferred Networks   PyTorch   RAY   SaturnCloud   Uber   URSA LABS

FRACTIONAL DIFFERENCING

Easy & Fast



GPU-accelerated

Fractional Differencing for Time Series Stationarity

Ritchie Ng, Jie Fu, Tat-Seng Chua

NVIDIA
ENSEMBLE CAPITAL

	100k	1m	10m	100m
Speed-up 1x T4 vs 8x vCPUs	6.38 x	67.38 x	237.66 x	328.08 x
Speed-up 1x V100 vs 8x vCPUs	9.87 x	83.76 x	281.15 x	411.72 x

```
from numba import cuda

def moving_dot_product_kernel(in_data, out, window_size, weights):
    ...
    # [Single loop] Compute fractional differencing values
    for i in range(cuda.threadIdx.x + window_size - 1, in_data.size, cuda.blockDim.x):
        # Compute dot product of preceding window_size rows
        rolling_dot_product = 0.0
        k = 0
        for j in range(i - window_size + 1, i + 1):
            rolling_dot_product += in_data[j] * weights[k][0]
            k += 1

        out[i] = rolling_dot_product

def frac_diff_gpu(df, d, floor=1e-3):
    ...
    gdf_raw = cudf.from_pandas(df).reset_index(drop=True)
    gdf_raw.columns = ['in_data']
    ...
    # Bring weights to GPU
    gdf_weights = cudf.DataFrame()
    ...
    threads_per_block = 518
    ...
    # Get fractionally differenced time series through GPU function
    gdf_raw_fd = gdf_raw.apply_chunks(moving_dot_product_kernel,
                             incols=['in_data'],
                             outcols=dict(out=np.float64),
                             kwargs=dict(window_size=weights_window_size, weights=weights),
                             chunks=list(range(0, data_length, trunk_size)) + [data_length],
                             tpb=threads_per_block)

    # Bring to CPU for normal manipulation
    df_raw_fd = gdf_raw_fd.to_pandas().dropna().iloc[:-1, 1]

    return df_raw_fd, weights
```

NOT A STAC BENCHMARK

https://www.researchgate.net/publication/335159299_GFD_GPU_Fractional_Differencing_for_Rapid_Large-scale_Stationarizing_of_Time_Series_Data_while_Minimizing_Memory_Loss

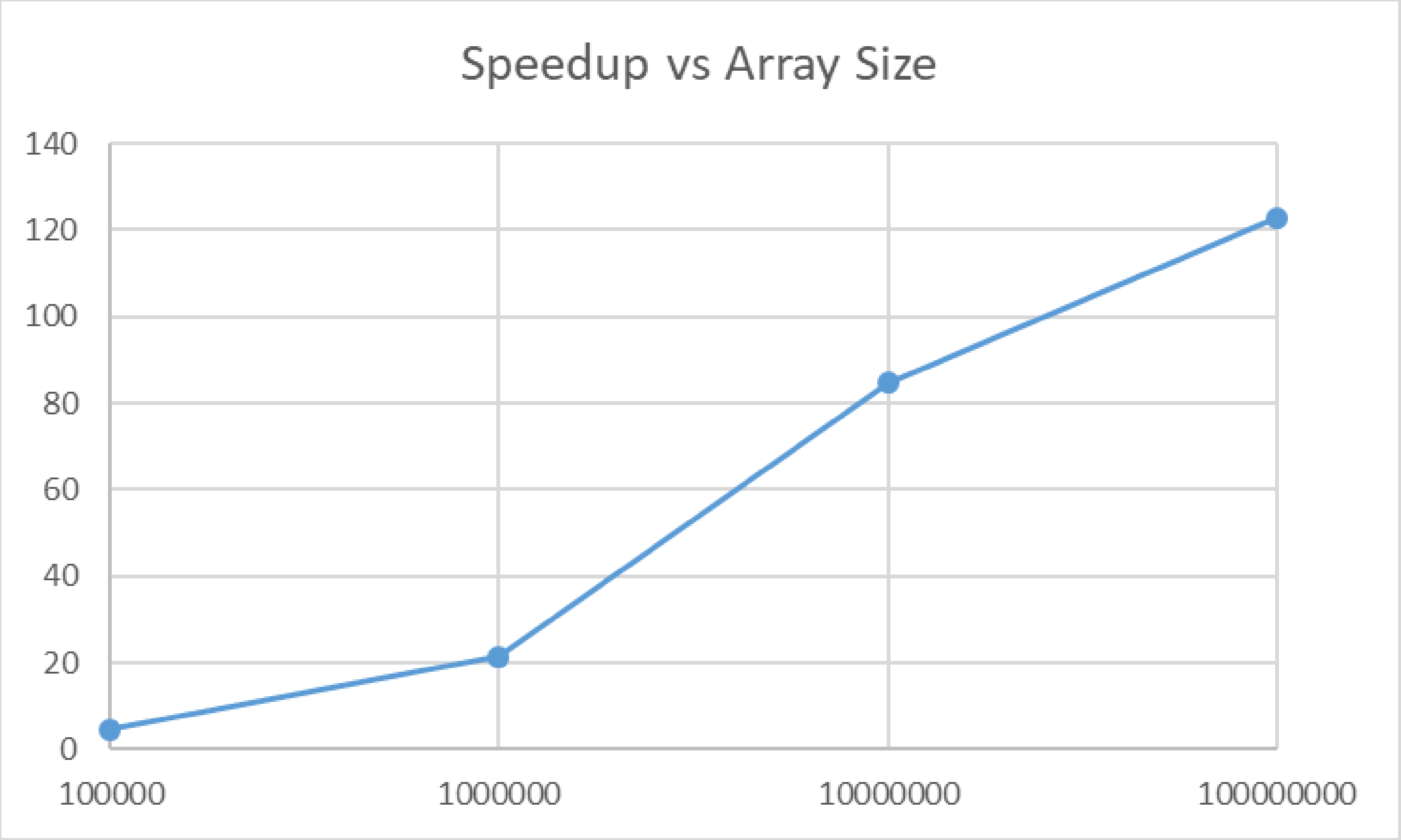
https://github.com/ritchieng/fractional_differencing_gpu/blob/master/notebooks/gpu_fractional_differencing.ipynb

FRACTIONAL DIFFERENCING

With Numba JIT, even faster!

NOT A STAC BENCHMARK

```
# allocate the output array
    gpu_out = numba.cuda.device_array_like(gpu_in)

     …

    # call the conv kernel
    kernel[(number_of_blocks,),
           (number_of_threads,),
           0,
           shared_buffer_size * 8](gpu_in,
                                   weights,
                                   gpu_out,
                                   window,
                                   array_len,
                                   thread_tile,
                                   min_periods)
    return gpu_out, weights_out
```



```
@cuda.jit
def kernel(in_arr, weight_arr, out_arr, window,
           arr_len, thread_tile, min_size):
…
shared = cuda.shared.array(shape=0,
                           dtype=numba.float64)
  …
   # copy the weights into the shared
   for j in range(0, window, block_size):
       element_id = tx + j
       if (((tx + j) < window) and (element_id < window)):
           shared[thread_tile * block_size + window - 1 + tx +
                j] = weight_arr[tx + j]
       cuda.syncthreads()
   # slice the shared memory for each threads
   start_shared = tx * thread_tile
   his_len = min(window - 1,
                 starting_id + tx * thread_tile)
   # slice the global memory for each threads
   start = starting_id + tx * thread_tile
   end = min(starting_id + (tx + 1) * thread_tile, arr_len)
   sub_outarr = out_arr[start:end]
   sub_len = end - start
   conv_window(shared, his_len, sub_outarr,
               window, sub_len,
               window - 1 + start_shared,
               thread_tile * block_size + window - 1,
               min_size)
```

https://medium.com/rapids-ai/fast-fractional-differencing-on-gpus-using-numba-and-rapids-part-1-b271a6b68b41

THANK YOU!