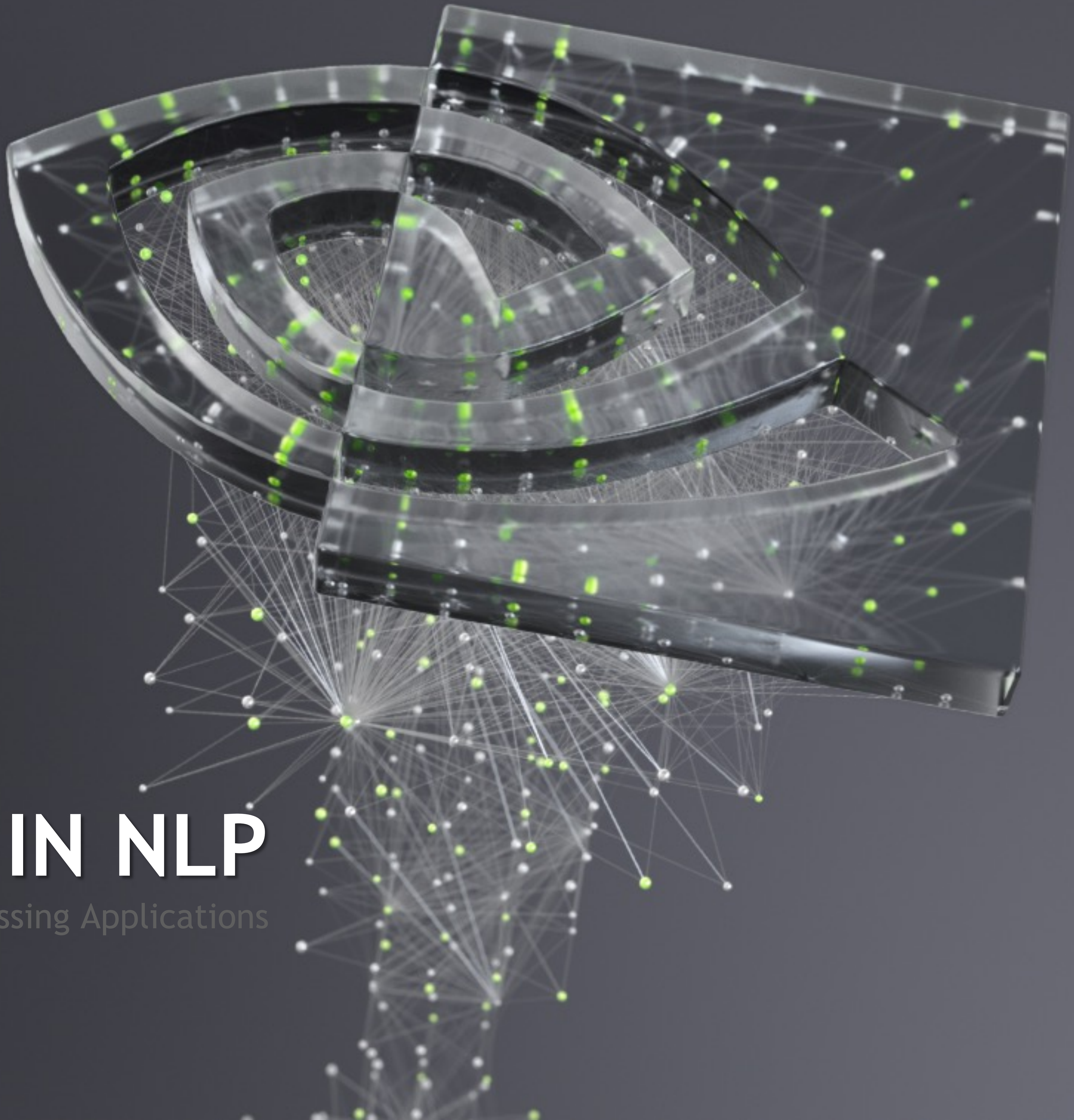
lrz

Leibniz Supercomputing Centre

of the Bavarian Academy of Sciences and Humanities

NVIDIA

DEEP
LEARNING
INSTITUTE



MACHINE LEARNING IN NLP

Building Transformer-Based Natural Language Processing Applications
(Part 1)

FULL COURSE AGENDA

Part 1: Machine Learning in NLP

Lecture: NLP background and the role of DNNs leading to the Transformer architecture

Lab: Tutorial-style exploration of a ***translation task*** using the Transformer architecture

Part 2: Self-Supervision, BERT, and Beyond

Lecture: Discussion of how language models with self-supervision have moved beyond the basic Transformer to BERT and ever larger models

Lab: Practical hands-on guide to the NVIDIA NeMo API and exercises to build a ***text classification task*** and a ***named entity recognition task*** using BERT-based language models

Part 3: Production Deployment

Lecture: Discussion of production deployment considerations and NVIDIA Triton Inference Server

Lab: Hands-on deployment of an example ***question answering task*** to NVIDIA Triton

Part 1: Machine Learning in NLP

- Lecture
  - What is NLP?
  - Problem Formulation
  - Text Representations
  - Dimensionality Reduction
  - Embeddings
  - RNNs
  - “Attention is All You Need”
- Lab
  - Transformer Architecture
  - BERT Model
  - Pretraining BERT

FOUNDATION OF COUNTLESS APPLICATIONS

NLP TASKS

Natural Language Processing

Automatic Writing

General Q&A

Intent Detection

Summary Generation

Common Sense Reasoning

Word Sense

Auto Completion

Automatic Dialogue Generation

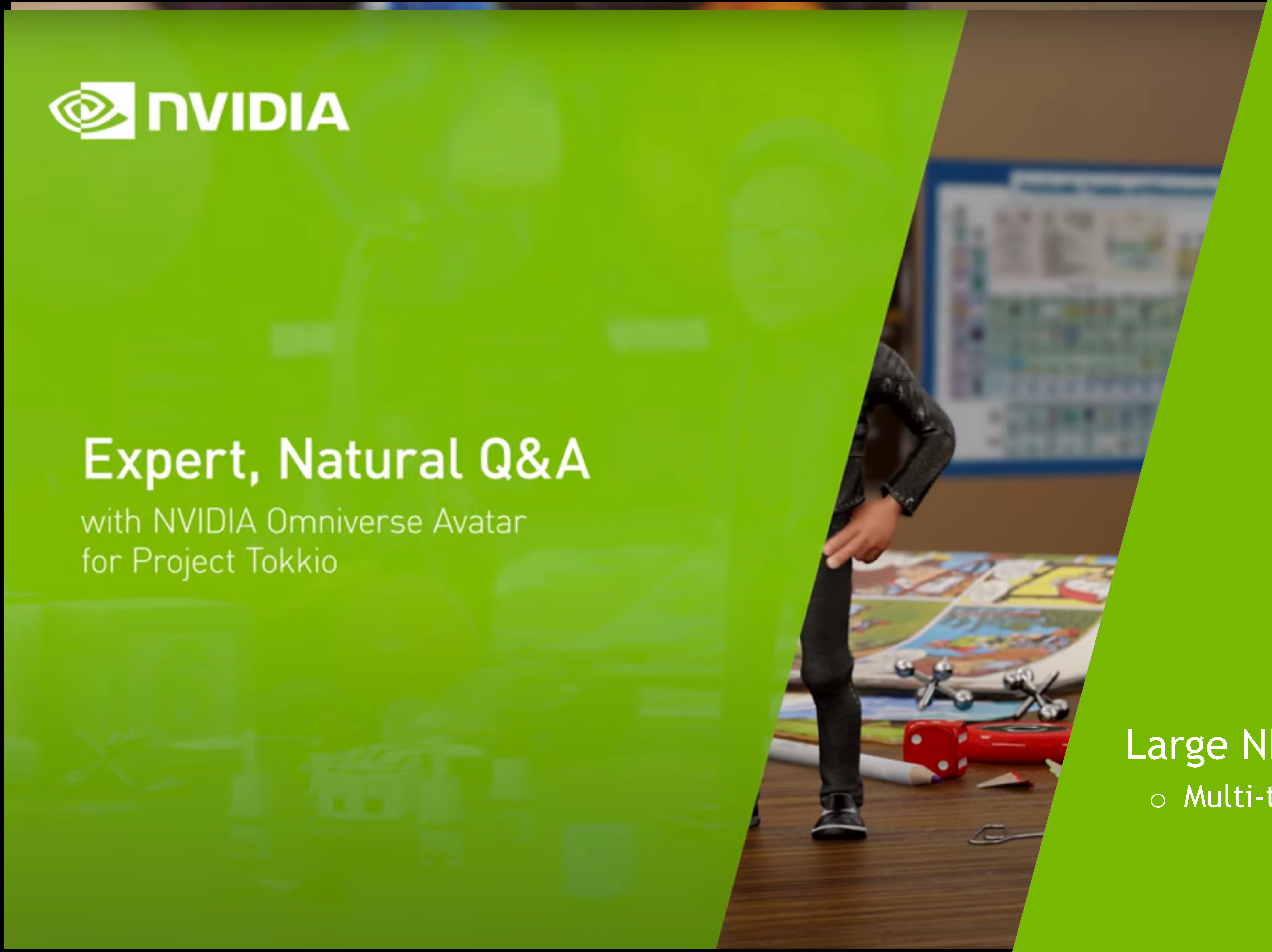
Translation

Sentiment Analysis

Code Generation

And many more….

GLIMPSE OF WHAT IS POSSIBLE, TODAY…



Large NLP models powers:

- Multi-turn Information Retrieval for Q&A

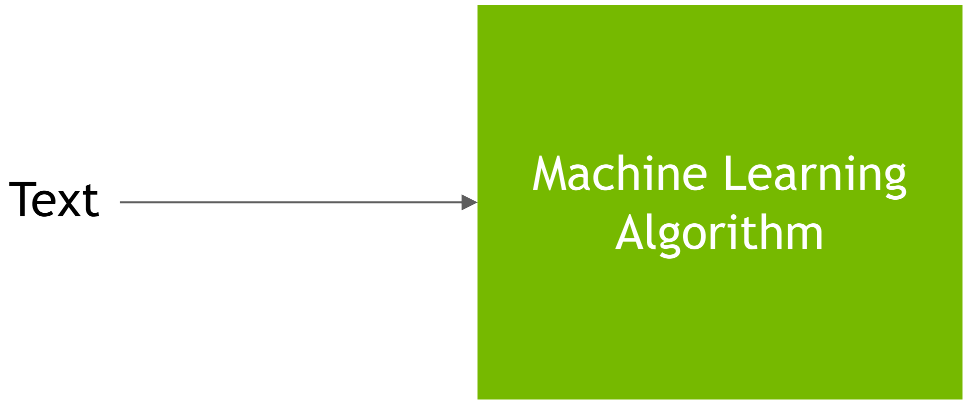
Part 1: Machine Learning in NLP

- Lecture
  - What is NLP?
  - Problem Formulation
  - Text Representations
  - Dimensionality Reduction
  - Embeddings
  - RNNs
  - “Attention is All You Need”
- Lab
  - Transformer Architecture
  - BERT Model
  - Pretraining BERT

PROBLEM FORMULATION

MACHINE LEARNING

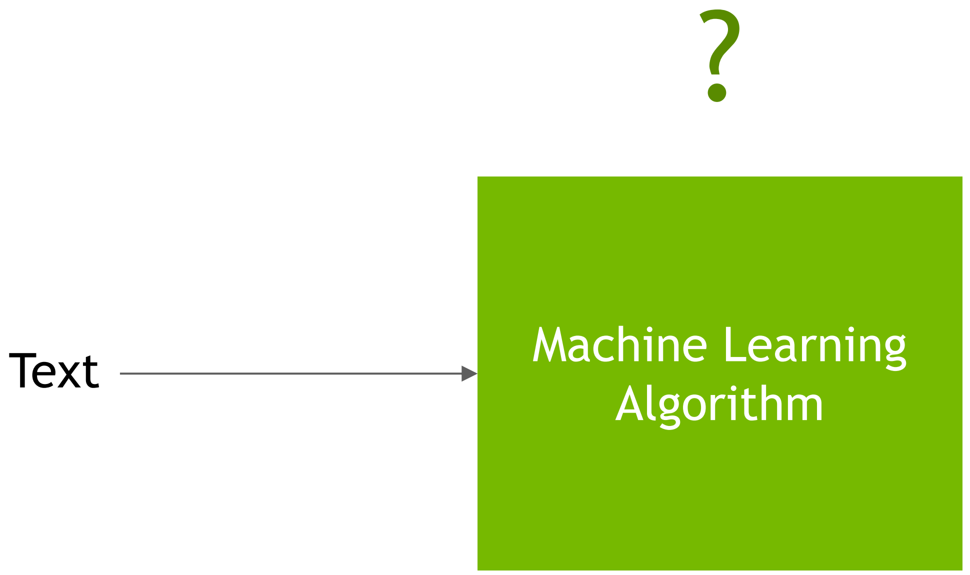
Discovering the discussed structures in text


Text

Machine Learning Algorithm

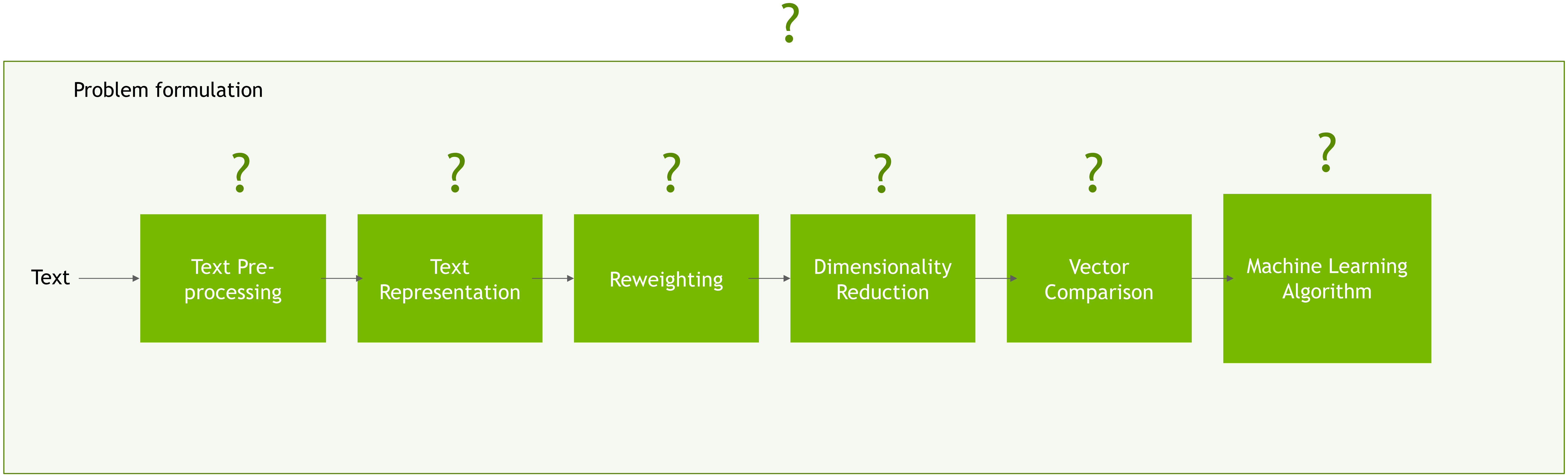
MACHINE LEARNING

Discovering the discussed structures in text



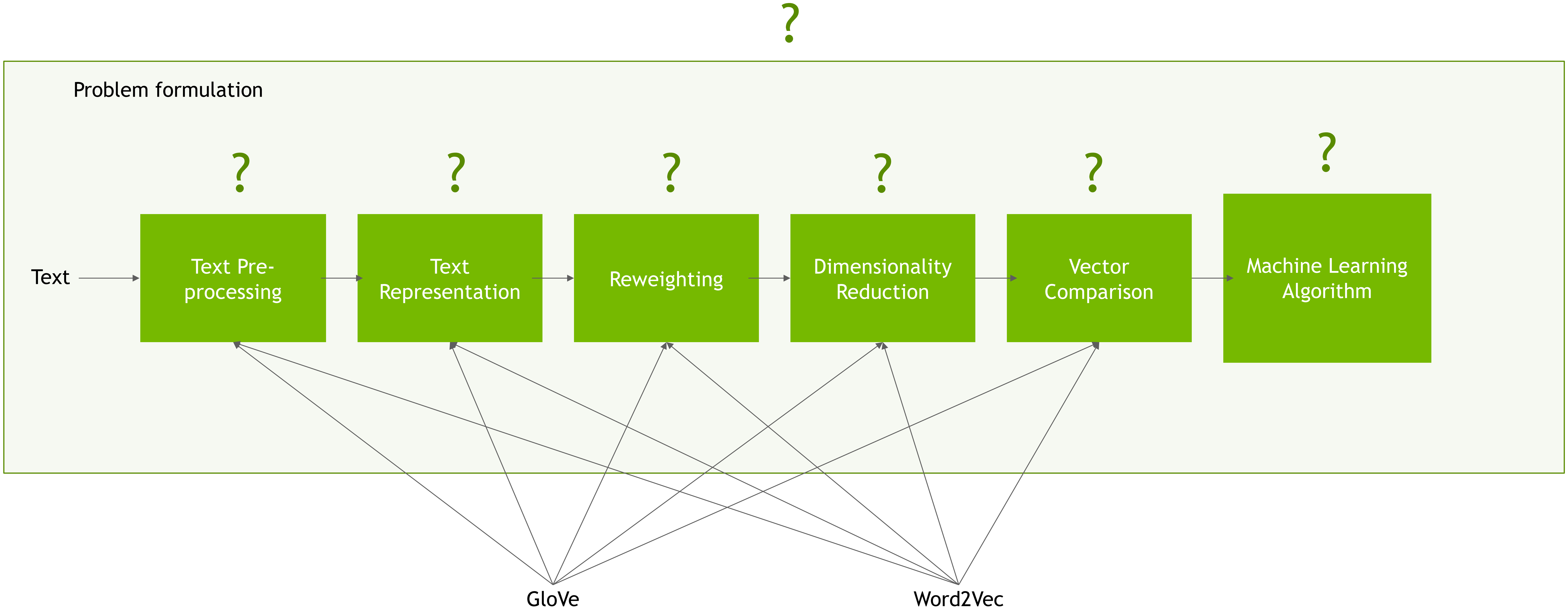
MACHINE LEARNING

Design decisions



MACHINE LEARNING

All linear combinations feasible



?

Problem formulation

?

?

?

?

?

?

Text

Text Pre-processing

Text Representation

Reweighting

Dimensionality Reduction

Vector Comparison

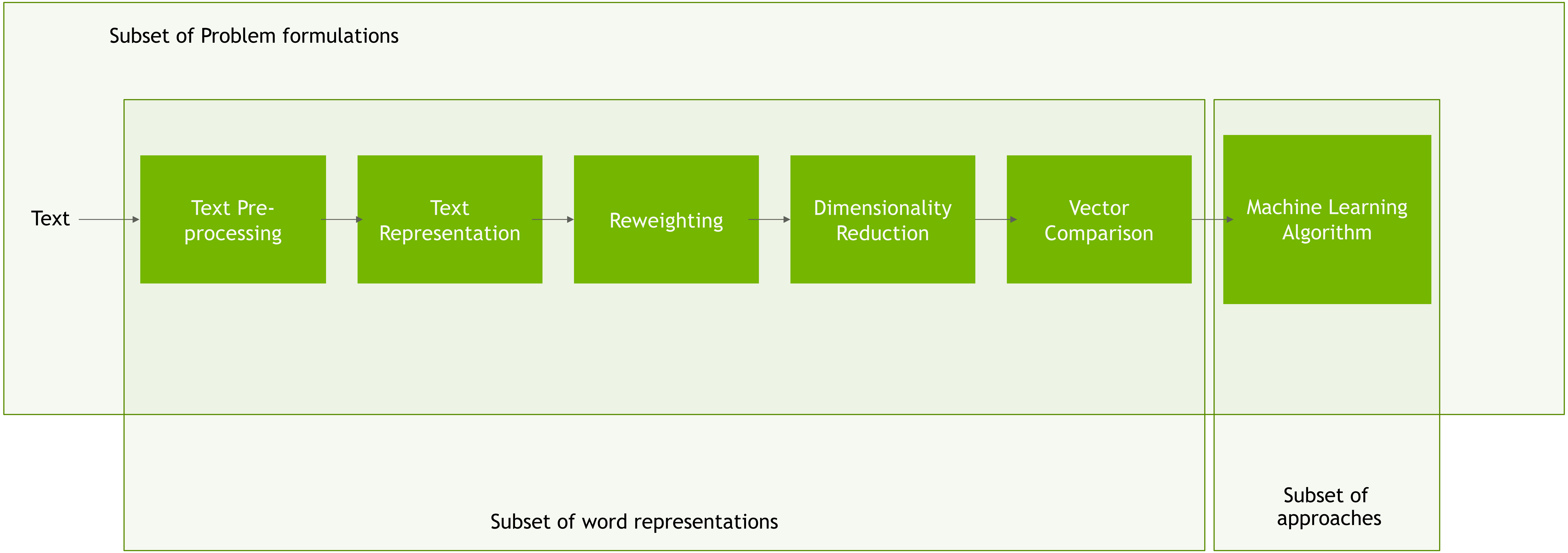
Machine Learning Algorithm

GloVe

Word2Vec

MACHINE LEARNING

In this class



Subset of Problem formulations

Text → Text Pre-processing → Text Representation → Reweighting → Dimensionality Reduction → Vector Comparison → Machine Learning Algorithm

Subset of word representations

Subset of approaches

Part 1: Machine Learning in NLP

- Lecture
  - What is NLP?
  - Problem Formulation
  - Text Representations
  - Dimensionality Reduction
  - Embeddings
  - RNNs
  - “Attention is All You Need”
- Lab
  - Transformer Architecture
  - BERT Model
  - Pretraining BERT

TEXT REPRESENTATIONS

The bag of words

- Bag of words/ngrams – feature per word/ngram

the cat sat on the mat

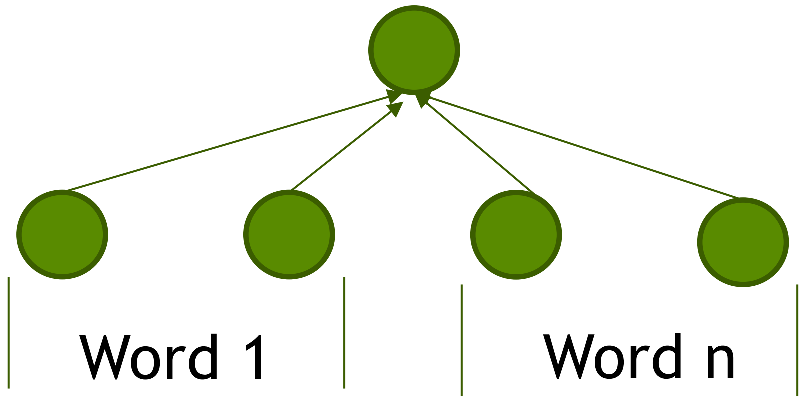
cat	sat	on	the	mat	quickly
1	1	1	2	1	0

… |Vocabulary|

THE BAG OF WORDS

Key challenges

- Sparse Input (1-hot)



p >> n (overfitting!)

lots of data required,
low accuracy

- No semantic generalization
- *dog:* 1 0 0 0 0 … 0
- *cat:* 0 0 1 0 0 … 0

DISTRIBUTED WORD REPRESENTATIONS

DISTRIBUTIONAL HYPOTHESIS

The intuition

‘You can tell a word by the company it keeps’

Firth 1957

‘Distributional statements can cover all of the material of a language without requiring support from other types of information’

Harris 1954

‘The meaning of a word is its use in the language’

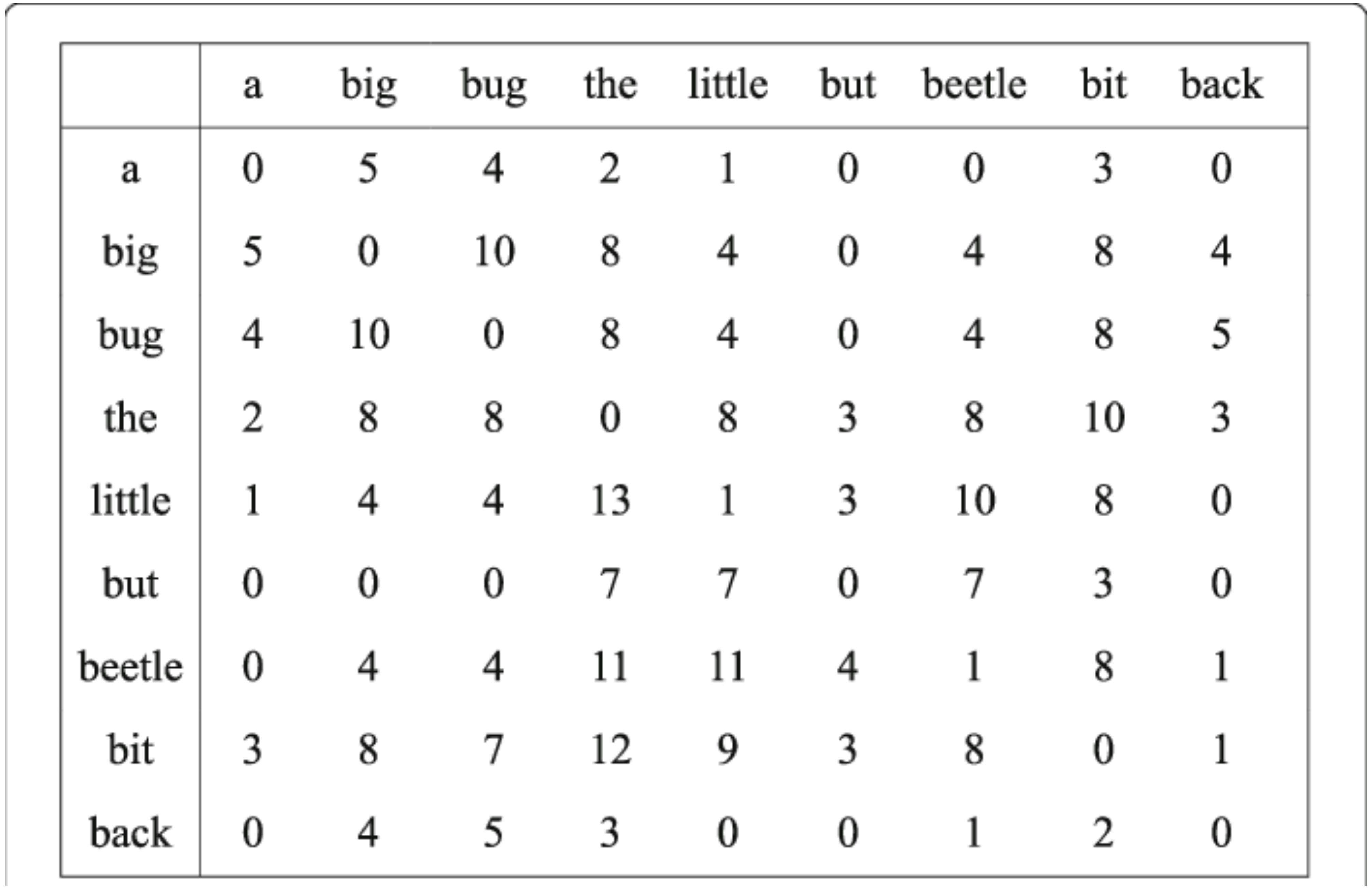
Wittgenstein 1953

‘The complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously.’

Firth 1957

CO-OCCURRENCE PATTERNS

The latent information



	a	big	bug	the	little	but	beetle	bit	back
a	0	5	4	2	1	0	0	3	0
big	5	0	10	8	4	0	4	8	4
bug	4	10	0	8	4	0	4	8	5
the	2	8	8	0	8	3	8	10	3
little	1	4	4	13	1	3	10	8	0
but	0	0	0	7	7	0	7	3	0
beetle	0	4	4	11	11	4	1	8	1
bit	3	8	7	12	9	3	8	0	1
back	0	4	5	3	0	0	1	2	0

CO-OCCURRENCE PATTERNS

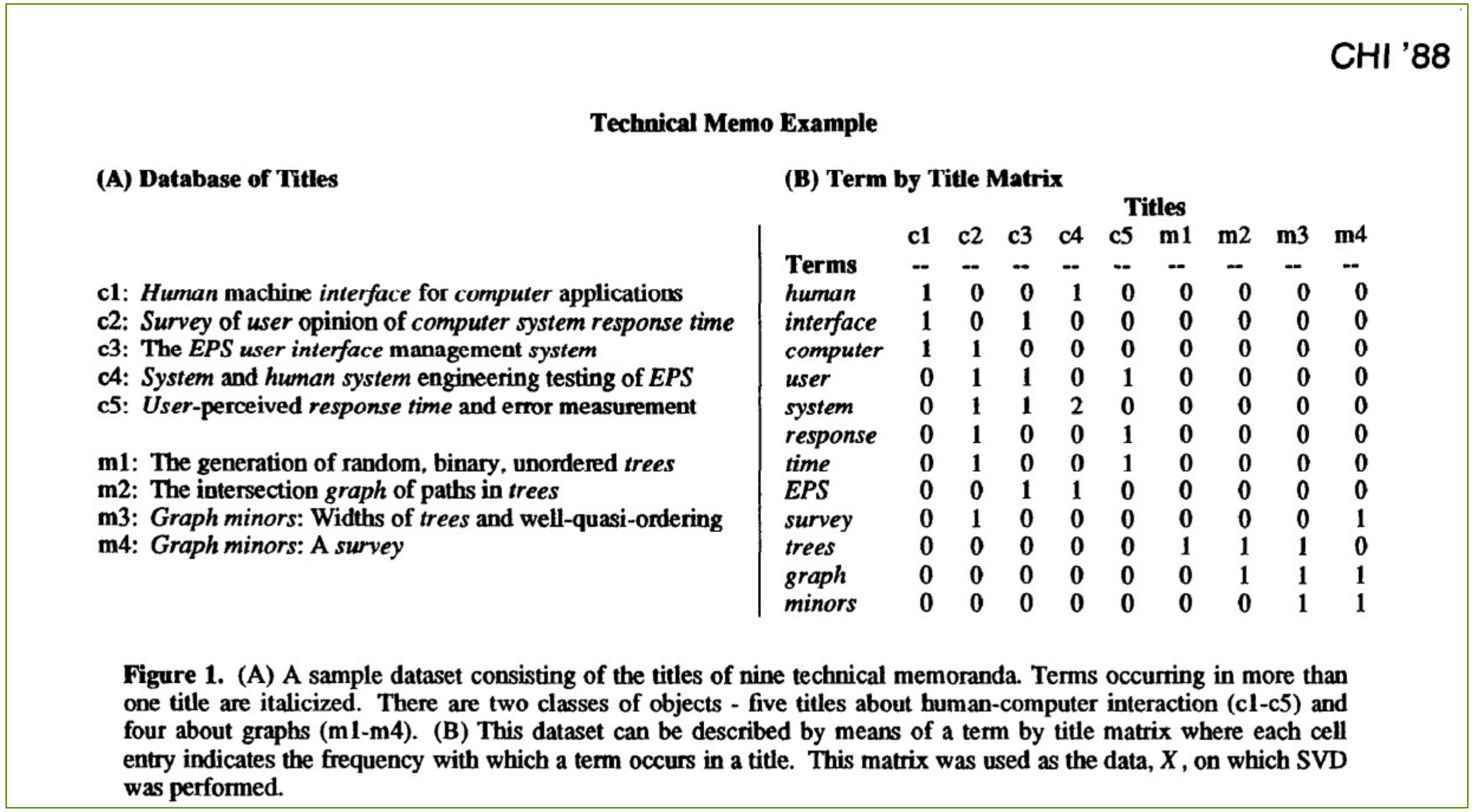
Where to find them?

Possible relationships:

- Word to documents (very sparse and very wide) ➡
- Word to word (very dense and compact)
- Word to user / person
- Word to user behaviour
- Word to product
- Word to custom feature (e.g. movie raking)

Not only metrices:

- Word to user to product



CHI '88

Technical Memo Example

(A) Database of Titles

c1: *Human* machine *interface* for *computer* applications
c2: *Survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: *User*-perceived *response time* and error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors*: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

(B) Term by Title Matrix

Titles

	c1	c2	c3	c4	c5	m1	m2	m3	m4
Terms	--	--	--	--	--	--	--	--	--
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Figure 1. (A) A sample dataset consisting of the titles of nine technical memoranda. Terms occurring in more than one title are italicized. There are two classes of objects - five titles about human-computer interaction (c1-c5) and four about graphs (m1-m4). (B) This dataset can be described by means of a term by title matrix where each cell entry indicates the frequency with which a term occurs in a title. This matrix was used as the data, *X*, on which SVD was performed.

Part 1: Machine Learning in NLP

- Lecture
  - What is NLP?
  - Problem Formulation
  - Text Representations
  - Dimensionality Reduction
  - Embeddings
  - RNNs
  - “Attention is All You Need”
- Lab
  - Transformer Architecture
  - BERT Model
  - Pretraining BERT

DIMENSIONALITY REDUCTION

Rationale

The need for compact and computationally efficient representations

More robust notions of distance exposing the information captured by our distributional representation

LSA/LSI

LSA/LSI

Latent Semantic Analysis / Latent Semantic Indexing

?

LLSA/LSI

Truncated SVD

Terms x Documents

$$X = T * S * P^T$$

Dumais, Susan T., et al. "Using latent semantic analysis to improve access to textual information." *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1988.

LSA/LSI

Truncated SVD

Terms x Documents

$$X = T * S * P^T$$

K largest singular values

$$X = T_k * S_k * P_k^T$$

Dumais, Susan T., et al. "Using latent semantic analysis to improve access to textual information." *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1988.

LSA/LSI

Truncated SVD

Terms x Documents

$$X = T * S * P^T$$

K largest singular values

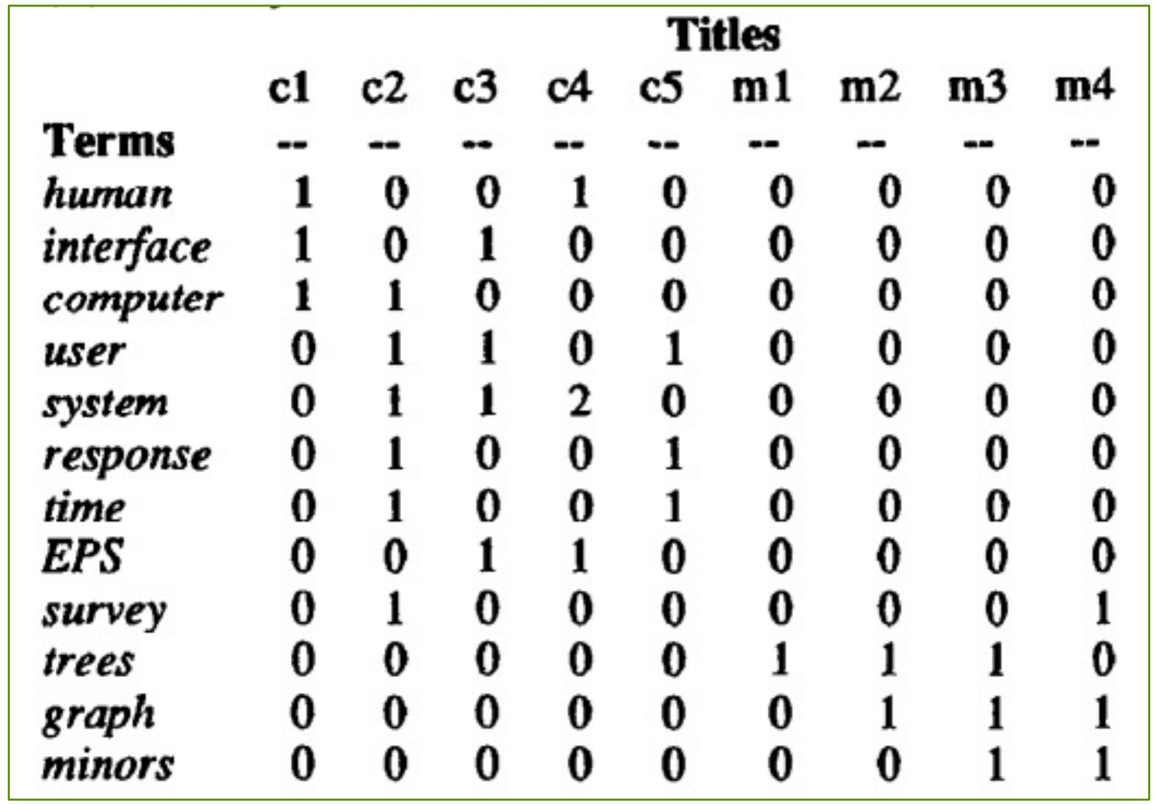
$$X = T_k * S_k * P_k^T$$

Latent Semantic Space

Dumais, Susan T., et al. "Using latent semantic analysis to improve access to textual information." *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1988.

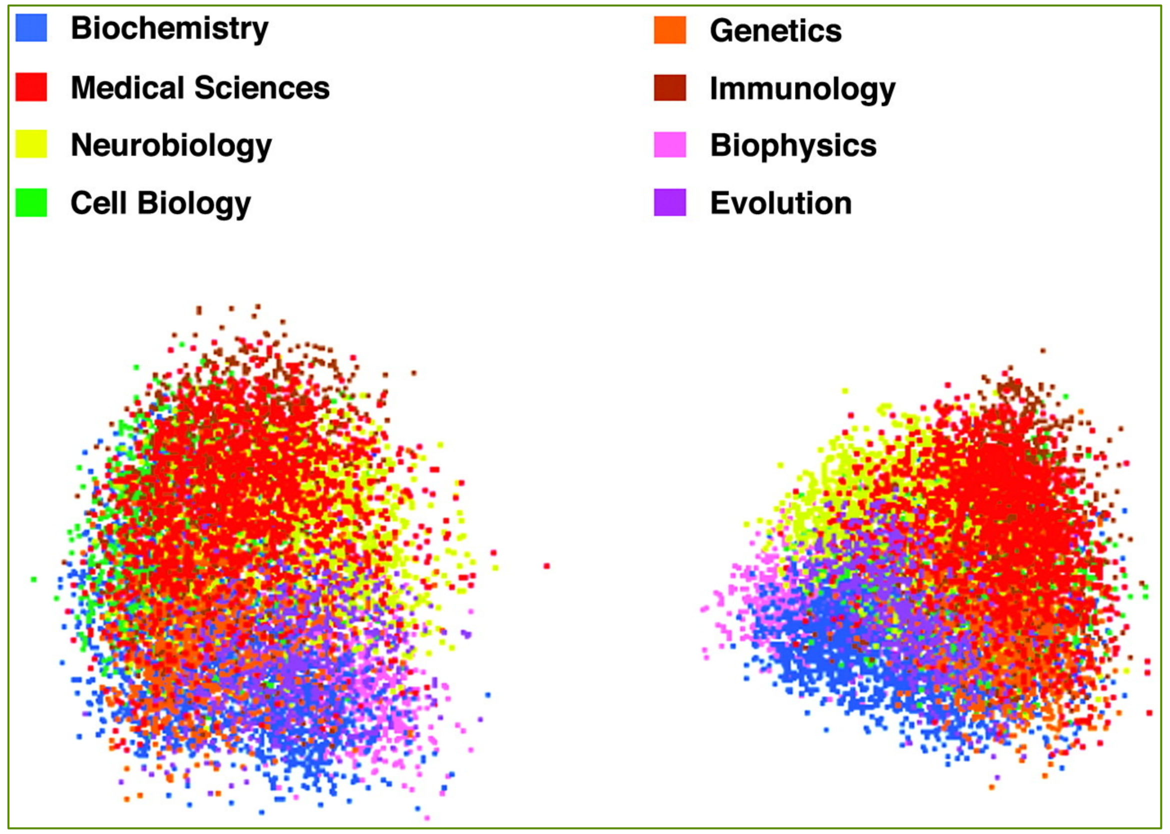
LSA/LSI

Documents that are similar are closer



	Titles								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
Terms	--	--	--	--	--	--	--	--	--
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

$$X = T_k * S_k * P_k^T$$



Landauer, Thomas K., Darrell Laham, and Marcia Derr. "From paragraph to graph: Latent semantic analysis for information visualization." *Proceedings of the National Academy of Sciences* 101.suppl 1 (2004): 5214-5219.

LSA/LSI

Its so 1988

Dumais, Susan T., et al. "Using latent semantic analysis to improve access to textual information." *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1988.

DID WE MAKE FURTHER PROGRESS?

STATUS AS OF 2010

Yes and No

Clustering Based Representations

- Brown Clustering
- HMM-LDA
- CRF Chunker with HMM
- …

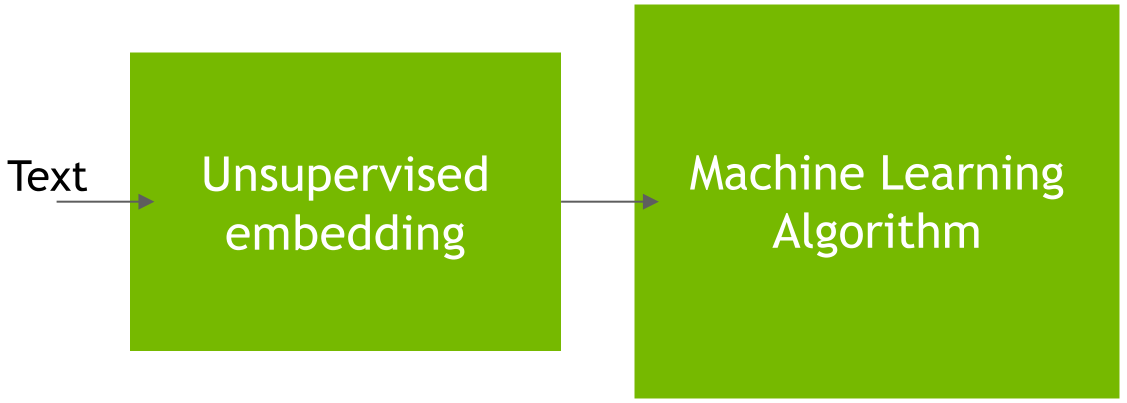
Distributional Representations

- LSA / LSI
- pLSA
- LDA
- HAL
- ICA
- Random Indexing
- …

Distributed Representations

- Collobert and Weston embeddings
- HLBL embeddings
- …

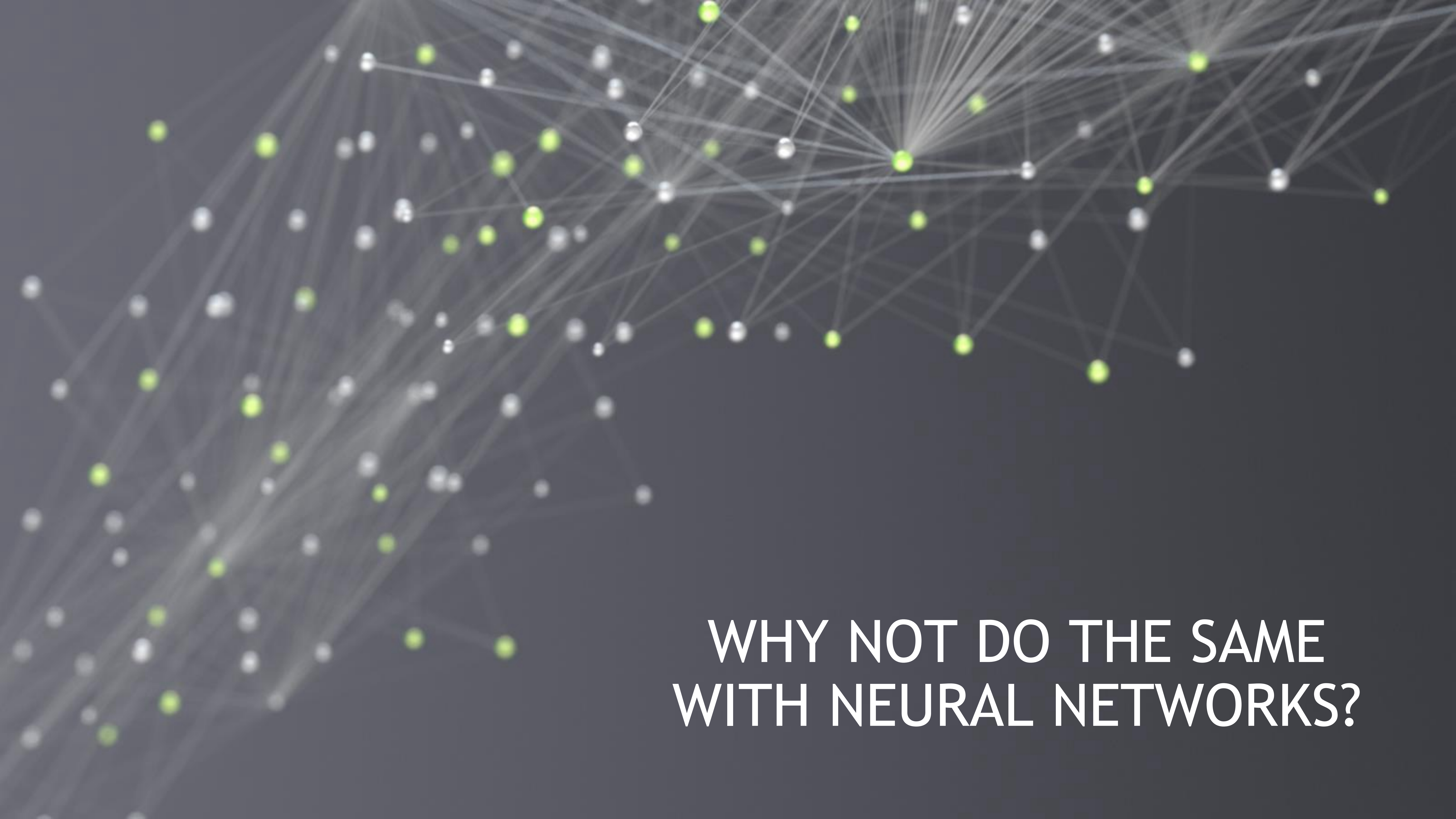
It was not clear that you can combine unsupervised approaches (i.e. embeddings) with supervised models



Turian, Joseph, Lev Ratinov, and Yoshua Bengio. "Word representations: a simple and general method for semi-supervised learning." *Proceedings of the 48th annual meeting of the association for computational linguistics*. 2010.

Part 1: Machine Learning in NLP

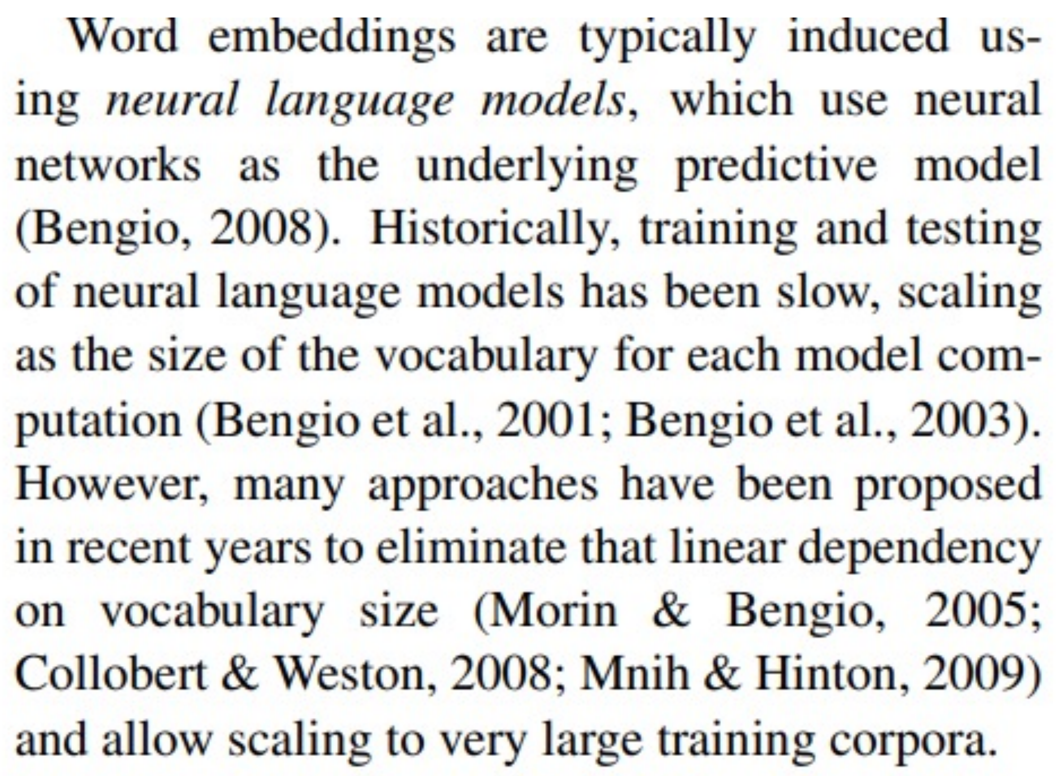
- Lecture
  - What is NLP?
  - Problem Formulation
  - Text Representations
  - Dimensionality Reduction
  - Embeddings
  - RNNs
  - “Attention is All You Need”
- Lab
  - Transformer Architecture
  - BERT Model
  - Pretraining BERT



WHY NOT DO THE SAME WITH NEURAL NETWORKS?

STATUS AS OF 2010

Not enough computational power



Word embeddings are typically induced using *neural language models*, which use neural networks as the underlying predictive model (Bengio, 2008). Historically, training and testing of neural language models has been slow, scaling as the size of the vocabulary for each model computation (Bengio et al., 2001; Bengio et al., 2003). However, many approaches have been proposed in recent years to eliminate that linear dependency on vocabulary size (Morin & Bengio, 2005; Collobert & Weston, 2008; Mnih & Hinton, 2009) and allow scaling to very large training corpora.

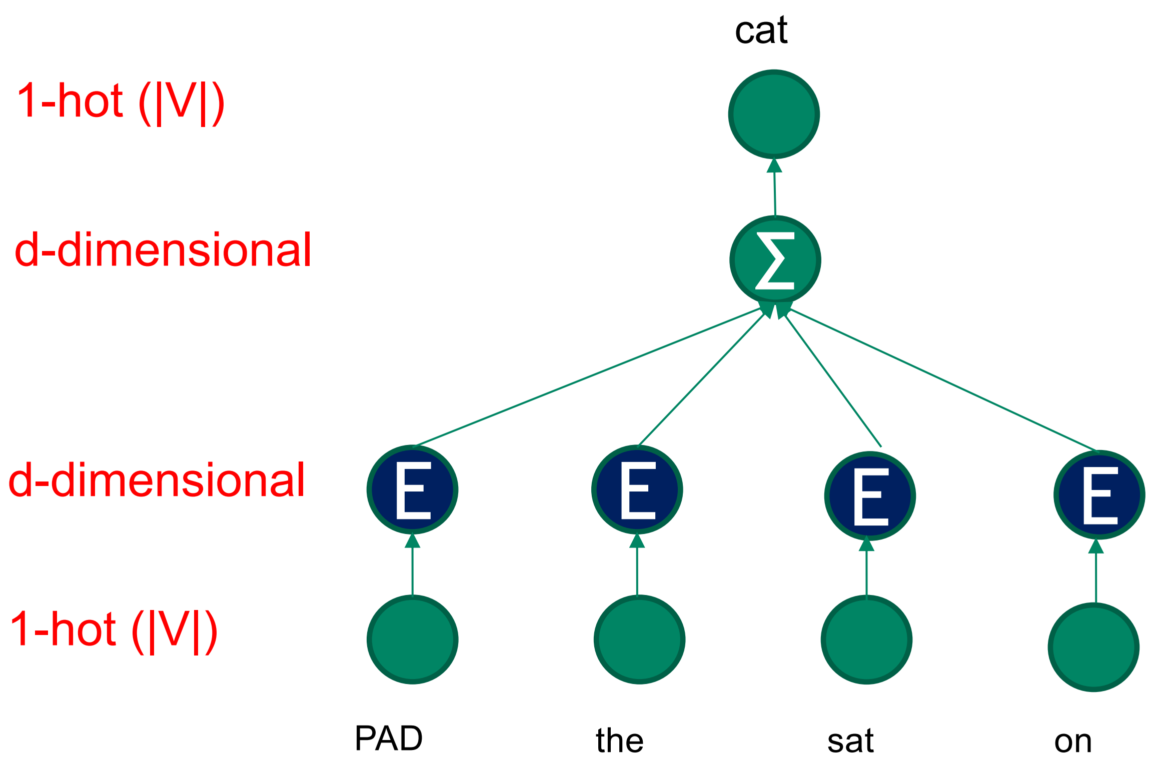
Turian, Joseph, Lev Ratinov, and Yoshua Bengio. "Word representations: a simple and general method for semi-supervised learning." *Proceedings of the 48th annual meeting of the association for computational linguistics*. 2010.

WORD2VEC

WORD2VEC

- Mikolov et al., 2013 (while at Google)
- Linear model (trains quickly)
- Two models for training embeddings in an *unsupervised* manner:

Continuous Bag-of-Words (CBOW)



cat

1-hot (|V|)

d-dimensional

Σ

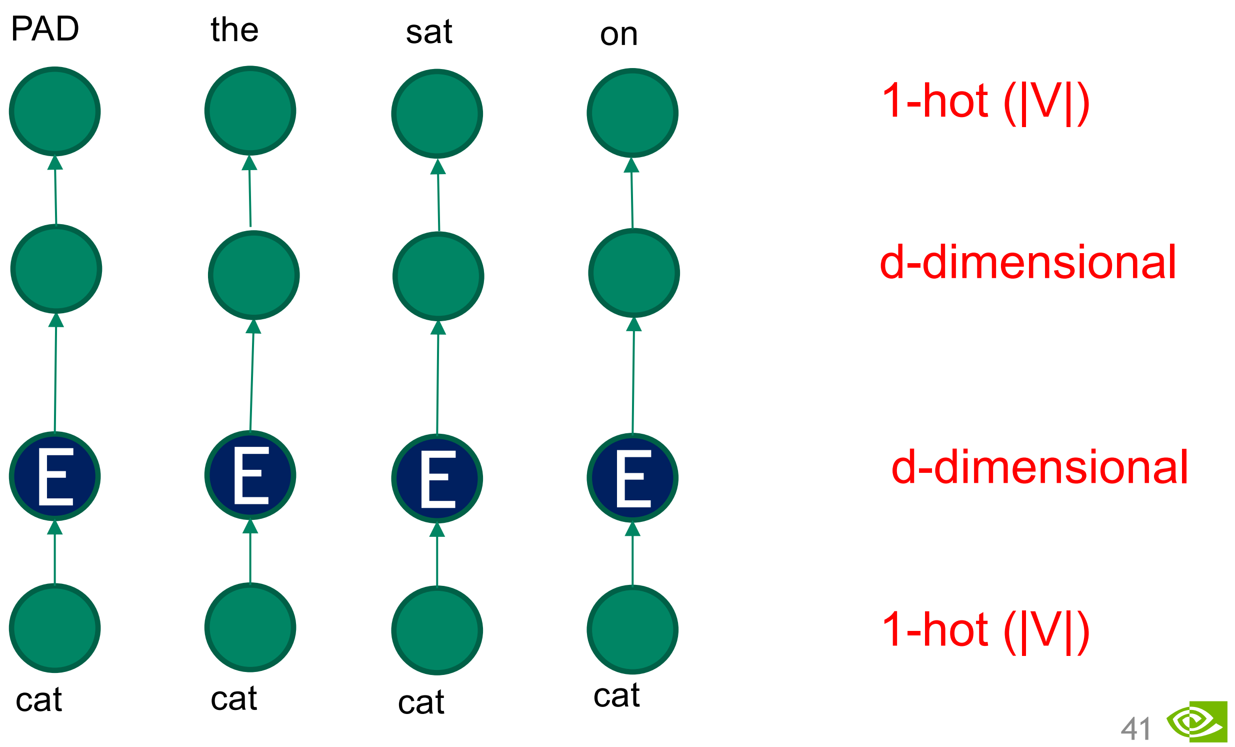
d-dimensional

E E E E

1-hot (|V|)

PAD the sat on

Skip-Gram



PAD the sat on

1-hot (|V|)

d-dimensional

E E E E

d-dimensional

1-hot (|V|)

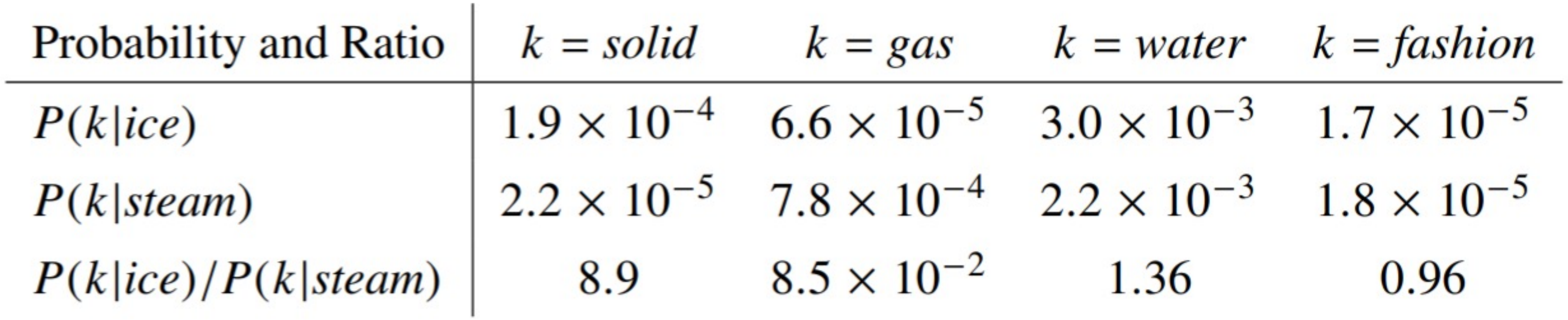
cat cat cat cat

GLOVE

GLOVE

The objective

To learn vectors for words such that their dot product is proportional to their probability of co-occurence

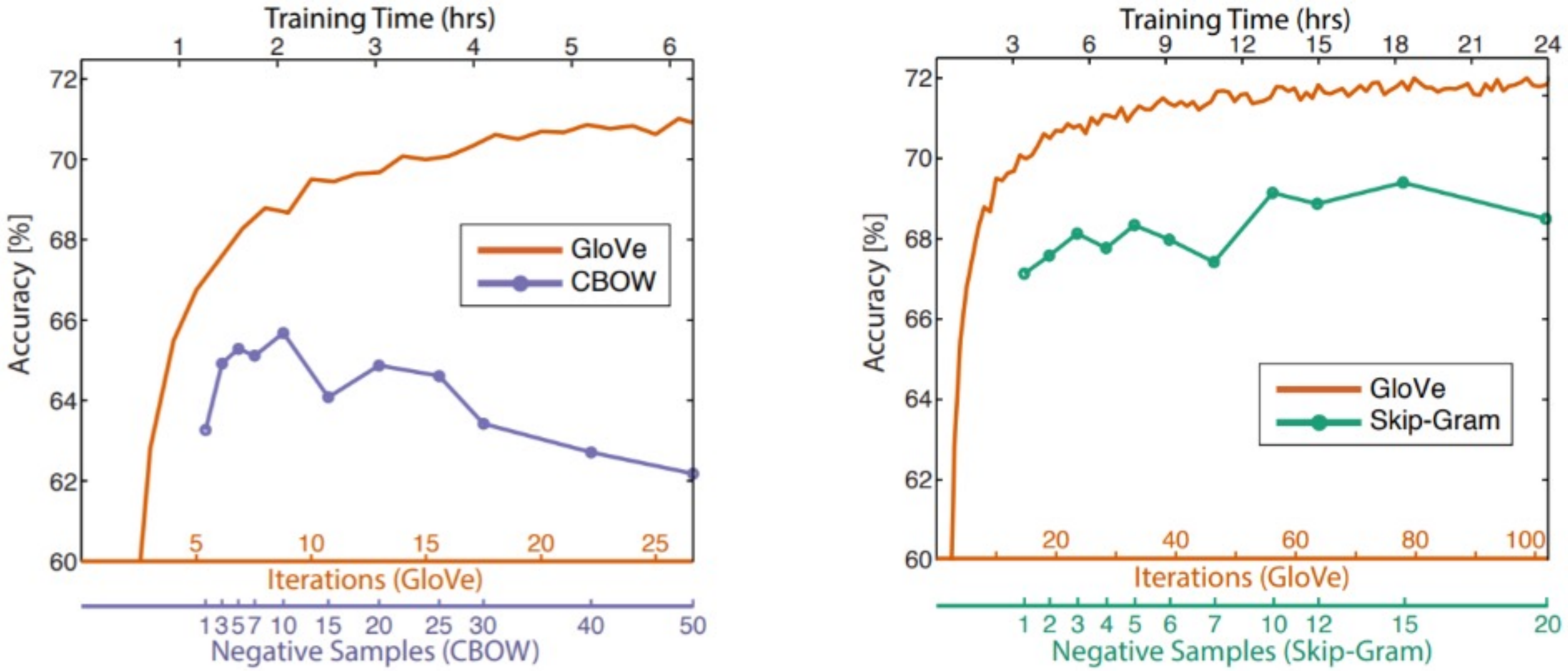


Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

GLOVE

The objective



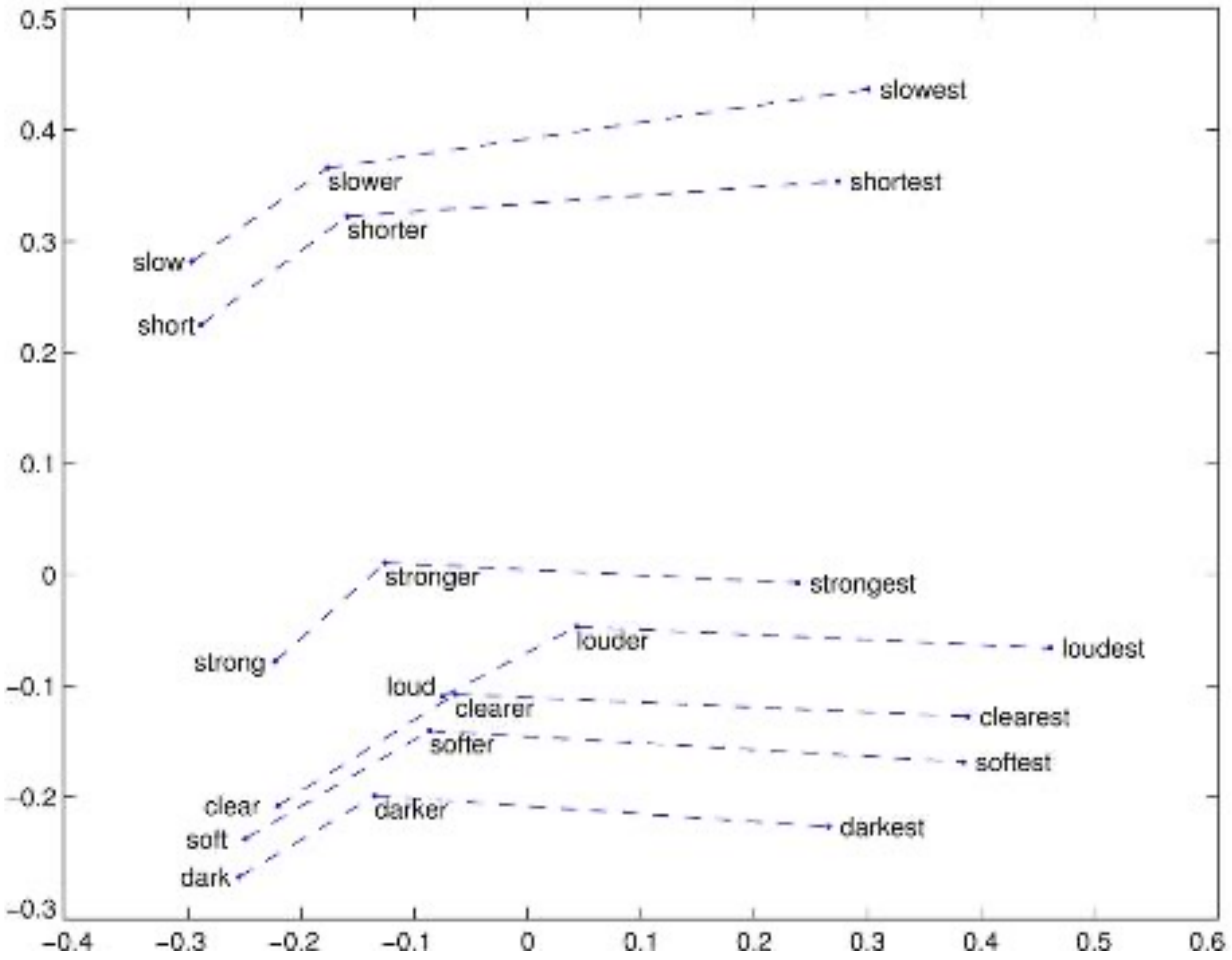
(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Figure 4: Overall accuracy on the word analogy task as a function of training time, which is governed by the number of iterations for GloVe and by the number of negative samples for CBOW (a) and skip-gram (b). In all cases, we train 300-dimensional vectors on the same 6B token corpus (Wikipedia 2014 + Gigaword 5) with the same 400,000 word vocabulary, and use a symmetric context window of size 10.

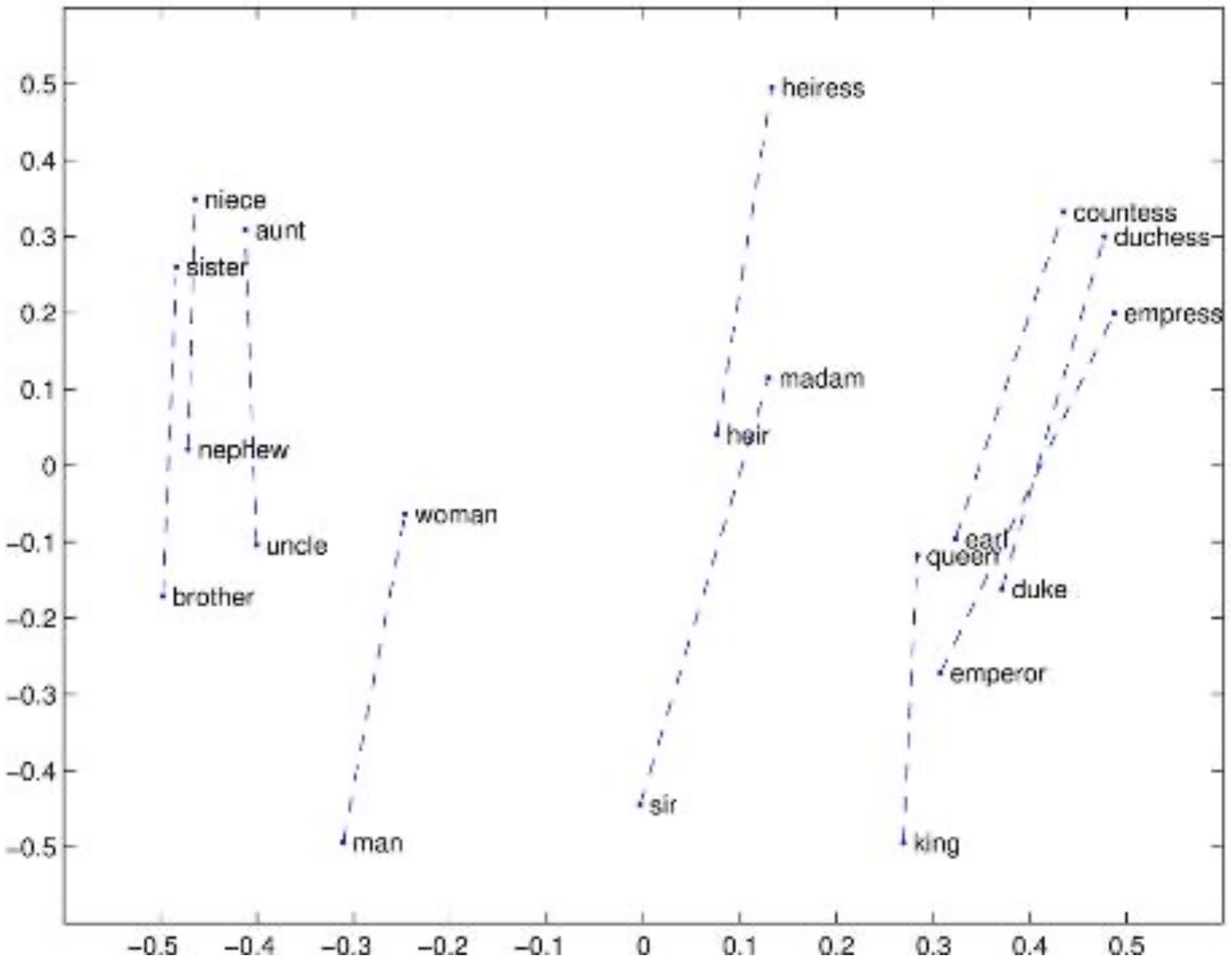
Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

GLOVE

Properties



Comparative - Superlative



Man - Woman

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

GLOVE

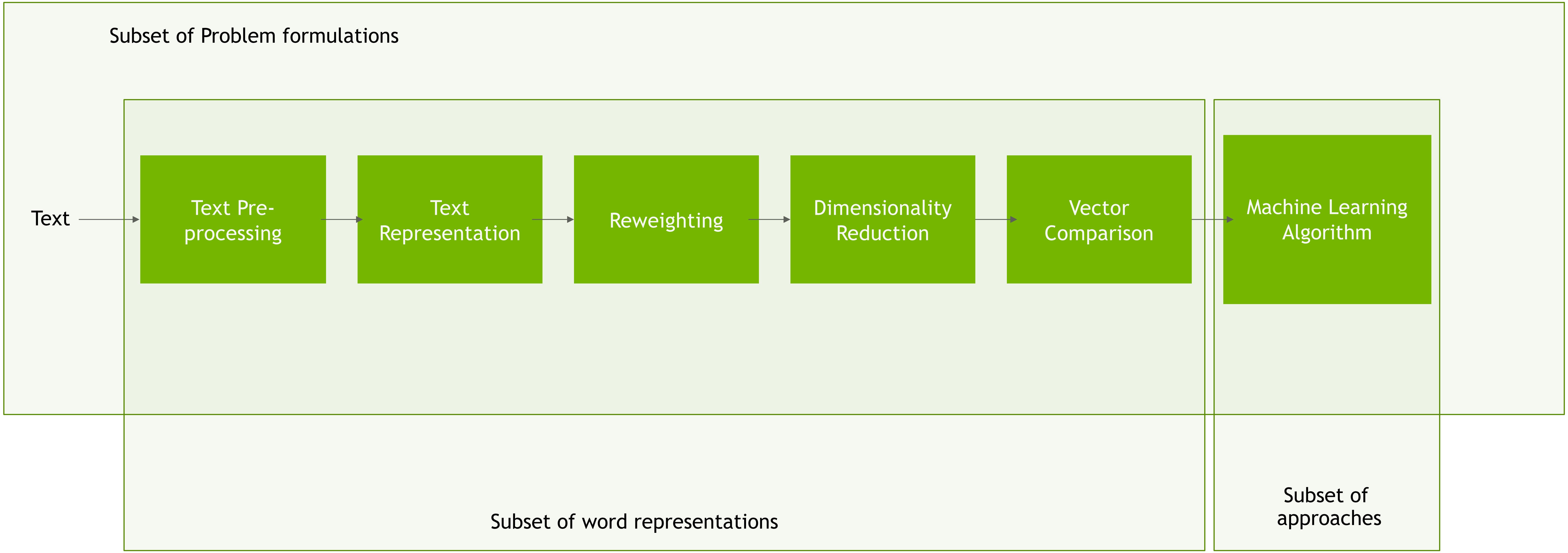
Not a distant past

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

USING THE EMBEDDINGS

THE APPROACH TO NLP

Unsupervised feature representation + Machine Learning models



Subset of Problem formulations

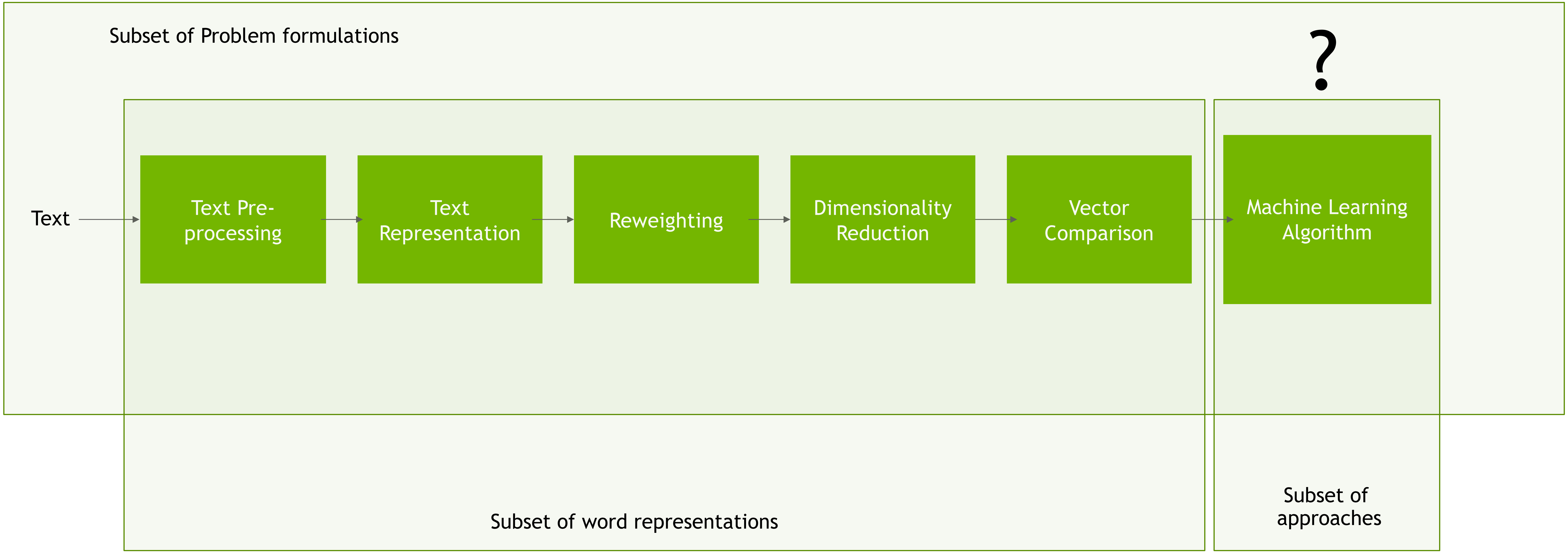
Text → Text Pre-processing → Text Representation → Reweighting → Dimensionality Reduction → Vector Comparison → Machine Learning Algorithm

Subset of word representations

Subset of approaches

THE APPROACH TO NLP

What ML model to choose



Subset of Problem formulations

?

Text → Text Pre-processing → Text Representation → Reweighting → Dimensionality Reduction → Vector Comparison → Machine Learning Algorithm

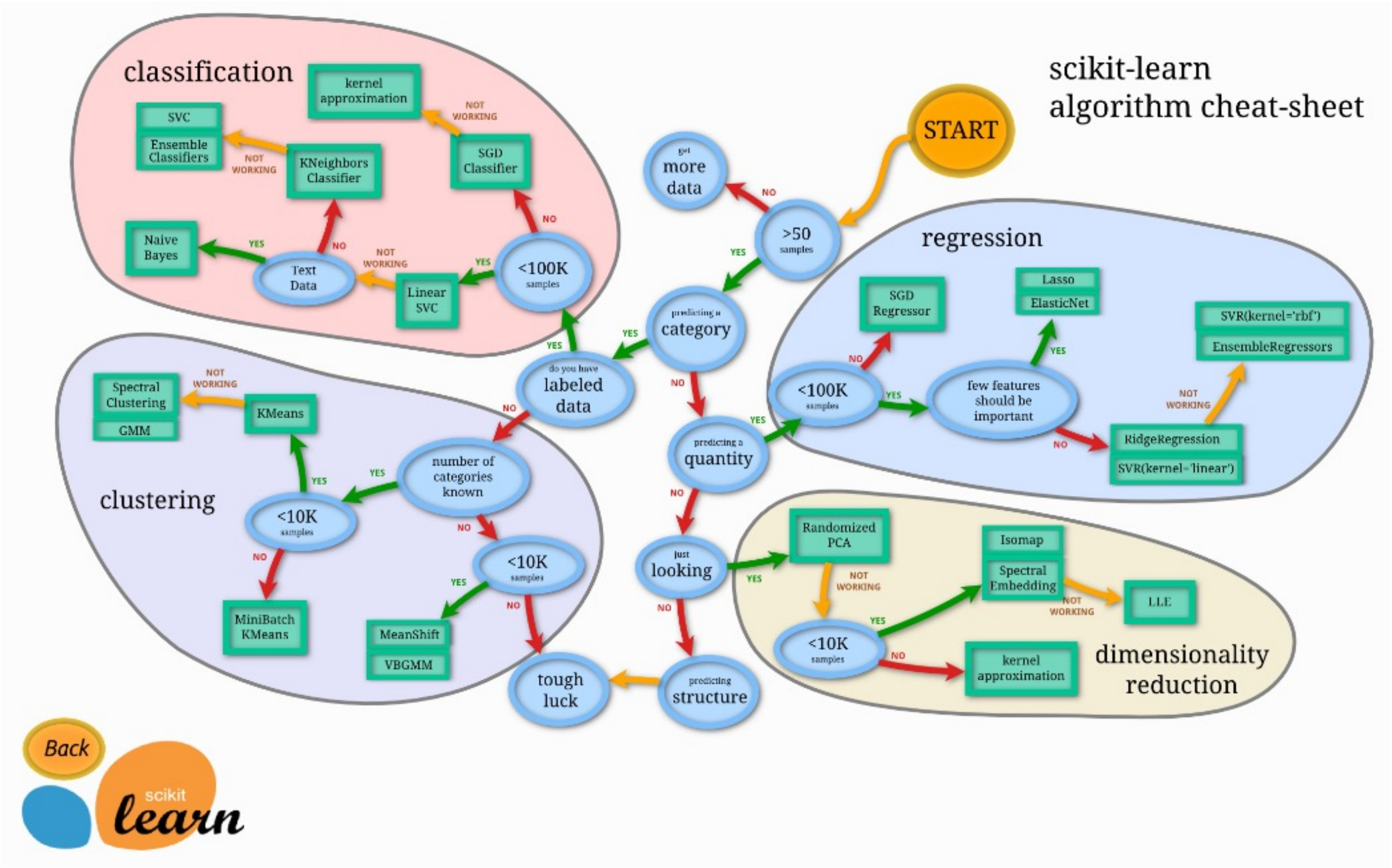
Subset of word representations

Subset of approaches

CLASSICAL APPROACHES

CLASSICAL APPROACHES

Very broad selection of tools

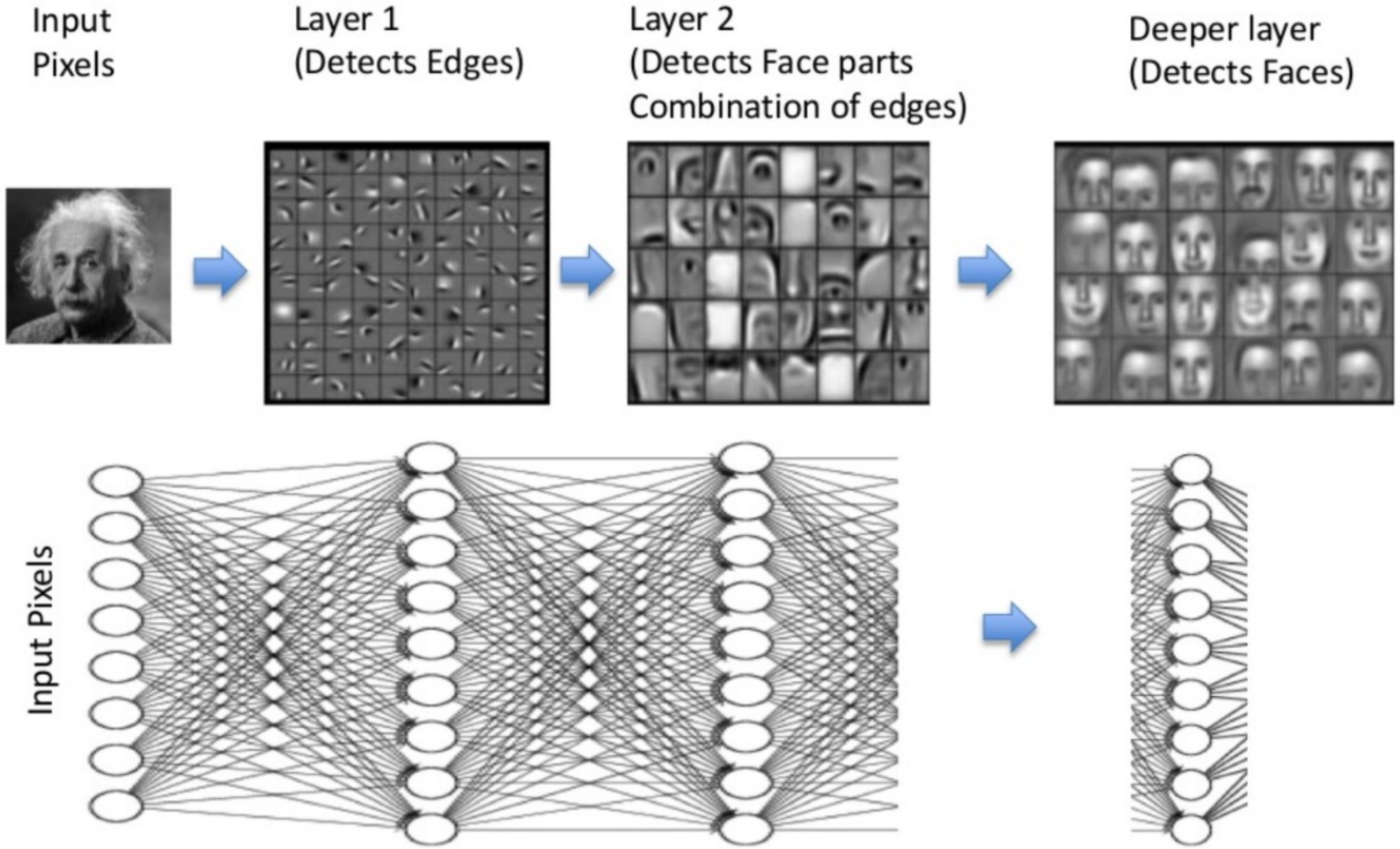


WHAT ABOUT FEATURE ENGINEERING?

DEEP REPRESENTATION LEARNING

DEEP REPRESENTATION LEARNING

Beyond distributional hypothesis

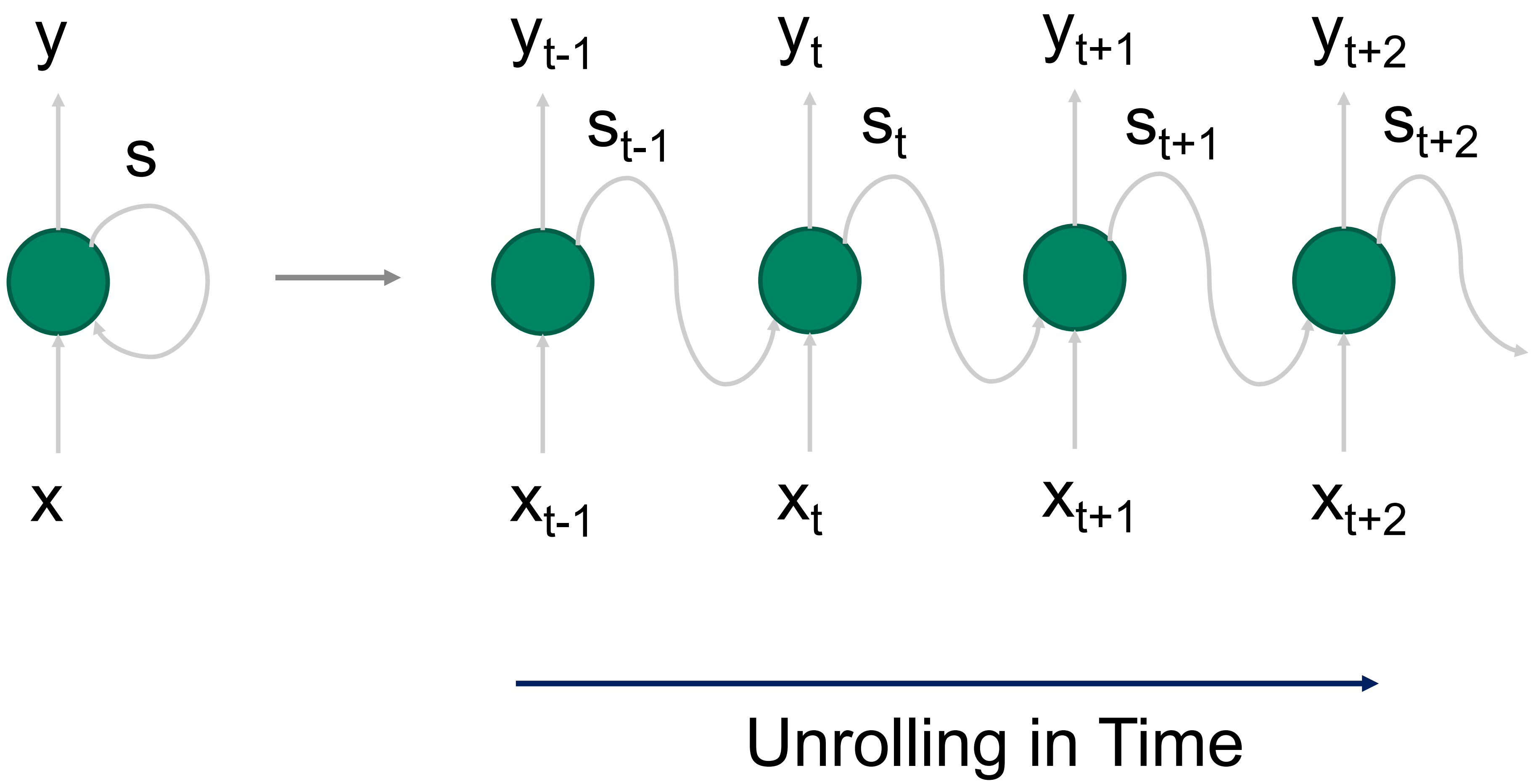


Part 1: Machine Learning in NLP

- Lecture
  - What is NLP?
  - Problem Formulation
  - Text Representations
  - Dimensionality Reduction
  - Embeddings
  - RNNs
  - “Attention is All You Need”
- Lab
  - Transformer Architecture
  - BERT Model
  - Pretraining BERT

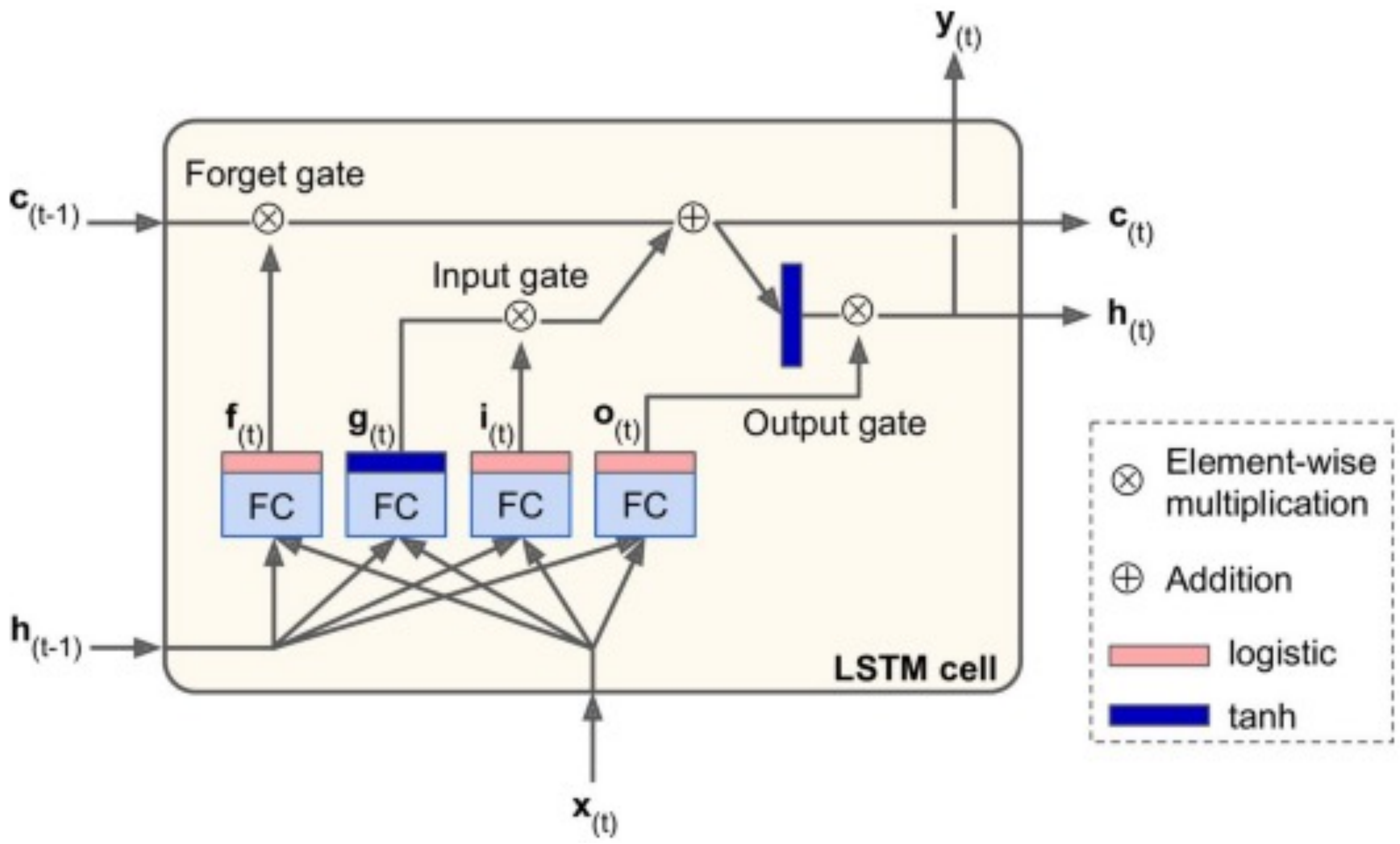
RECURRENT NEURAL NETWORKS

Basic principles



LONG SHORT TERM (LSTM) CELL

Addressing problems of stability

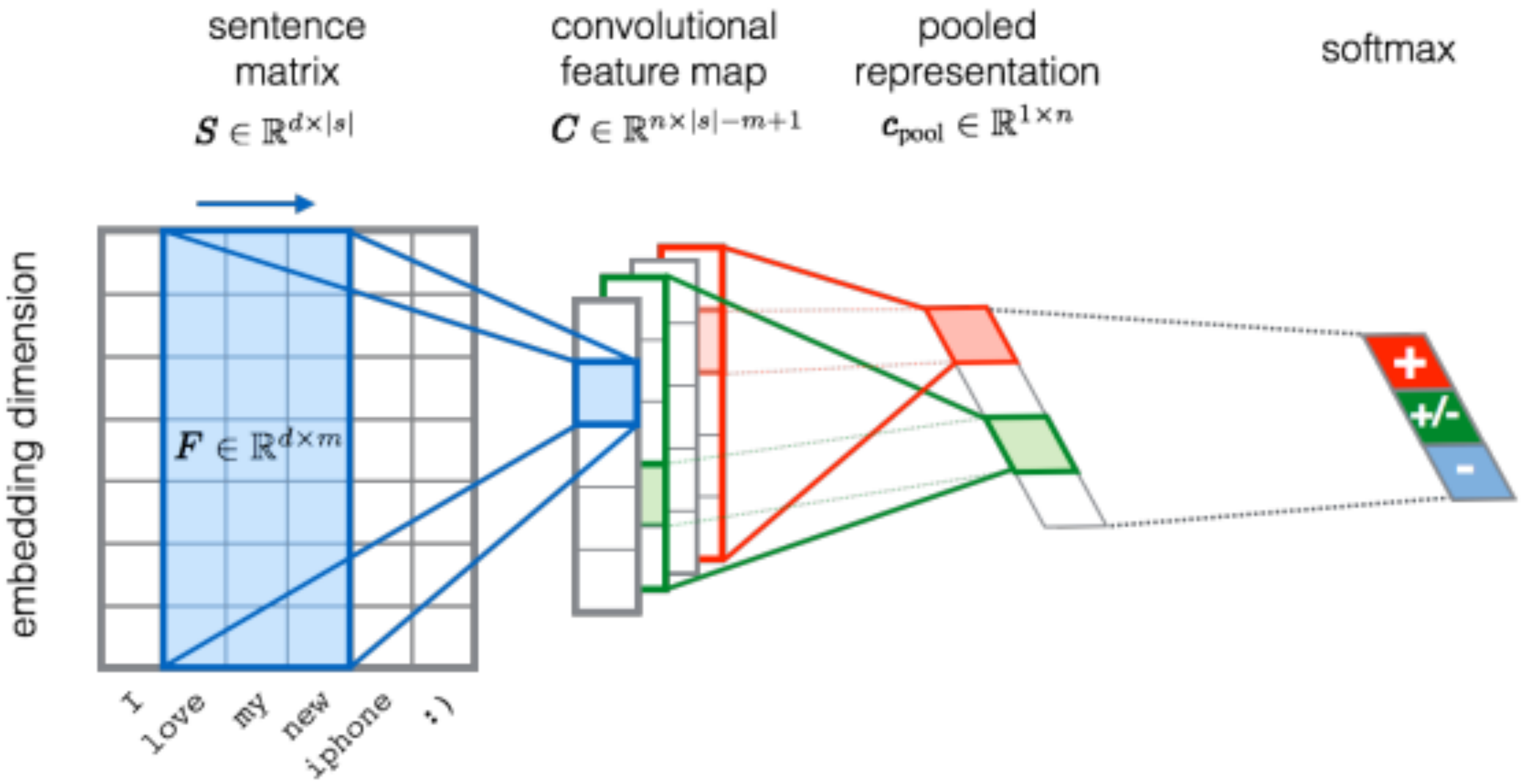


$$
\begin{aligned}
\mathbf{i}_{(t)} &= \sigma(\mathbf{W}_{xi}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{i}) \\
\mathbf{f}_{(t)} &= \sigma(\mathbf{W}_{xf}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{f}) \\
\mathbf{o}_{(t)} &= \sigma(\mathbf{W}_{xo}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{o}) \\
\mathbf{g}_{(t)} &= \tanh\,(\mathbf{W}_{xg}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_{g}) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh\,(\mathbf{c}_{(t)})
\end{aligned}
$$

CNNS

CONVOLUTIONAL NEURAL NETWORKS

Basic principles


sentence matrix
$S \in \mathbb{R}^{d \times |s|}$
convolutional feature map
$C \in \mathbb{R}^{n \times |s|-m+1}$
pooled representation
$c_{\text{pool}} \in \mathbb{R}^{1 \times n}$
softmax
embedding dimension
$F \in \mathbb{R}^{d \times m}$
+
+/-
-
I
love
my
new
iphone
:)

Severyn, Aliaksei, and Alessandro Moschitti. "Unitn: Training deep convolutional neural network for twitter sentiment classification." *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*. 2015.

ATTENTION

WHAT ABOUT LONG SEQUENCES?

The challenge illustrated with SQuAD

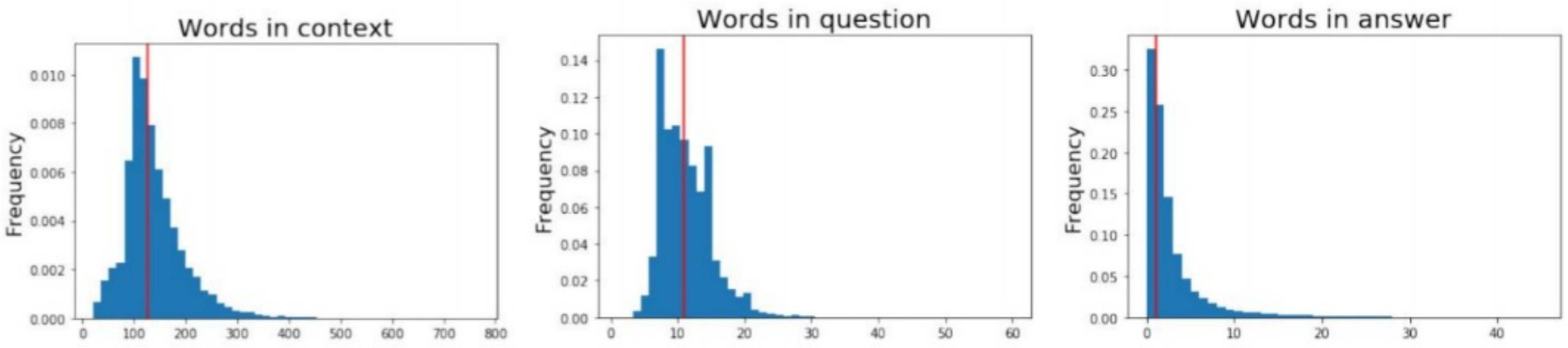


Figure 1: Number of words in contexts, questions, and answers in SQuAD training set.

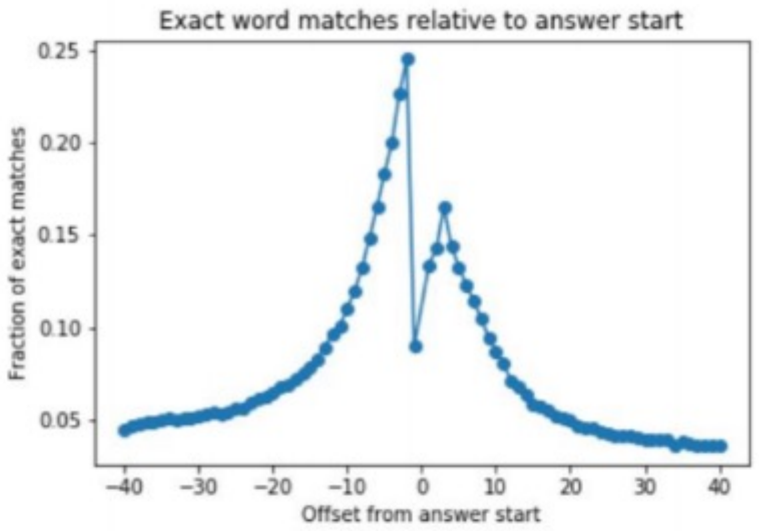
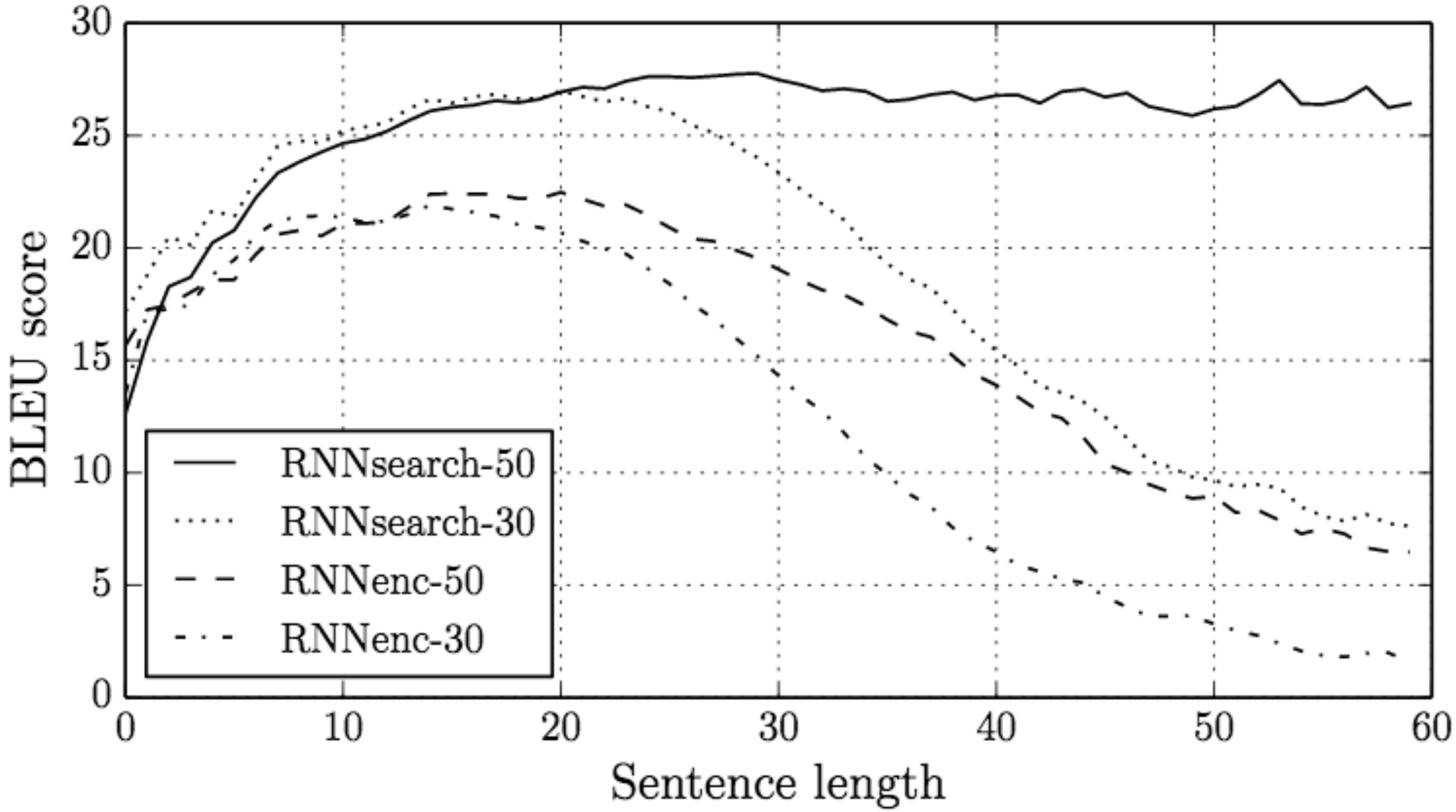


Figure 2: Frequency of exact word matches relative to answer start position

The impact of attention mechanism on Question Answering performance

WHAT ABOUT LONG SEQUENCES?

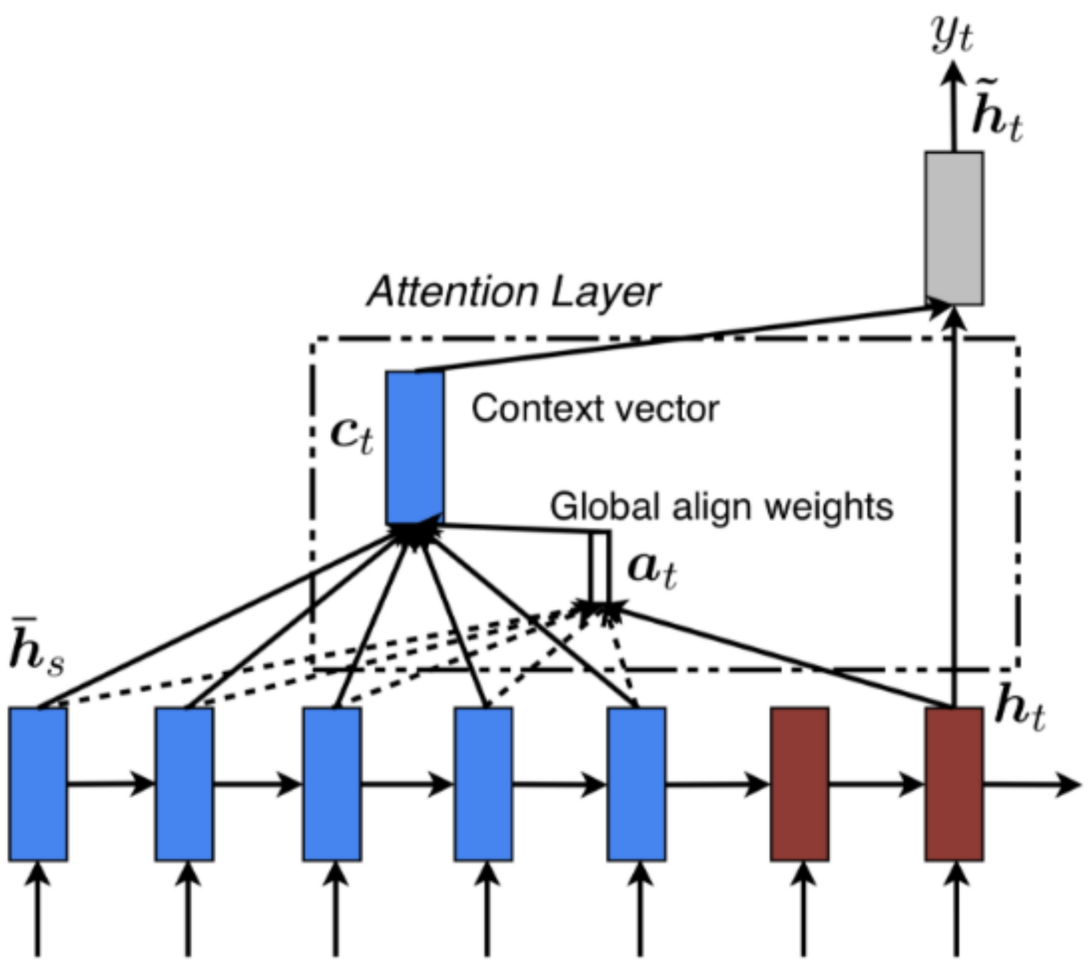
The challenge



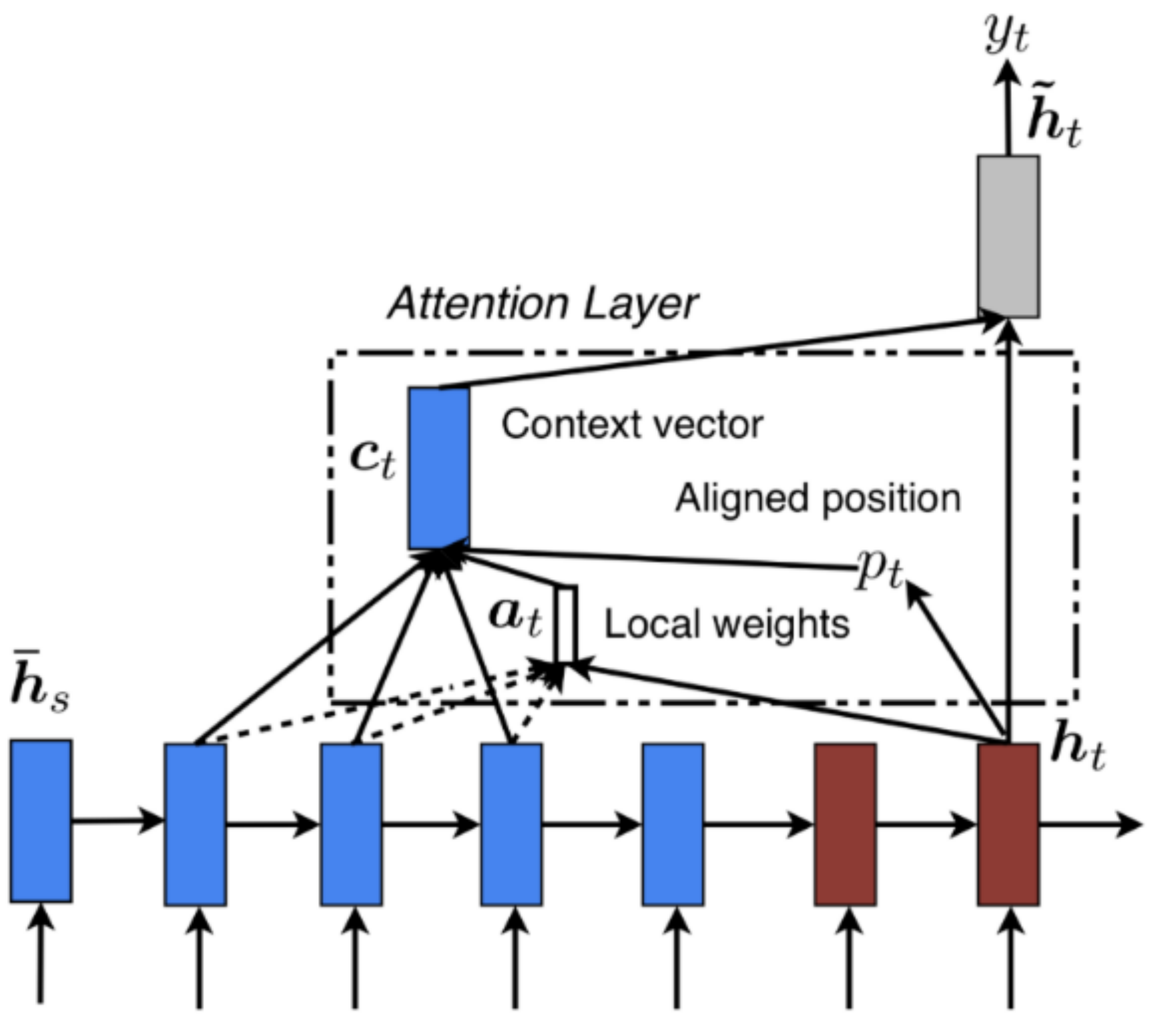
Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

ATTENTION

The mechanism



Global Attention Model



Local Attention Model

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

ATTENTION

The mechanism

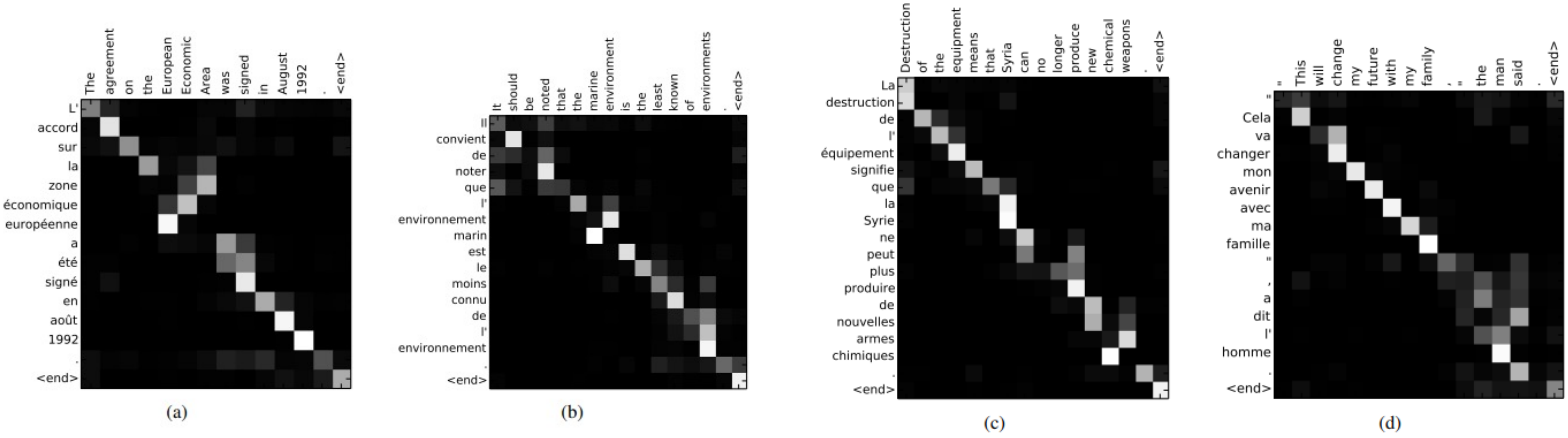
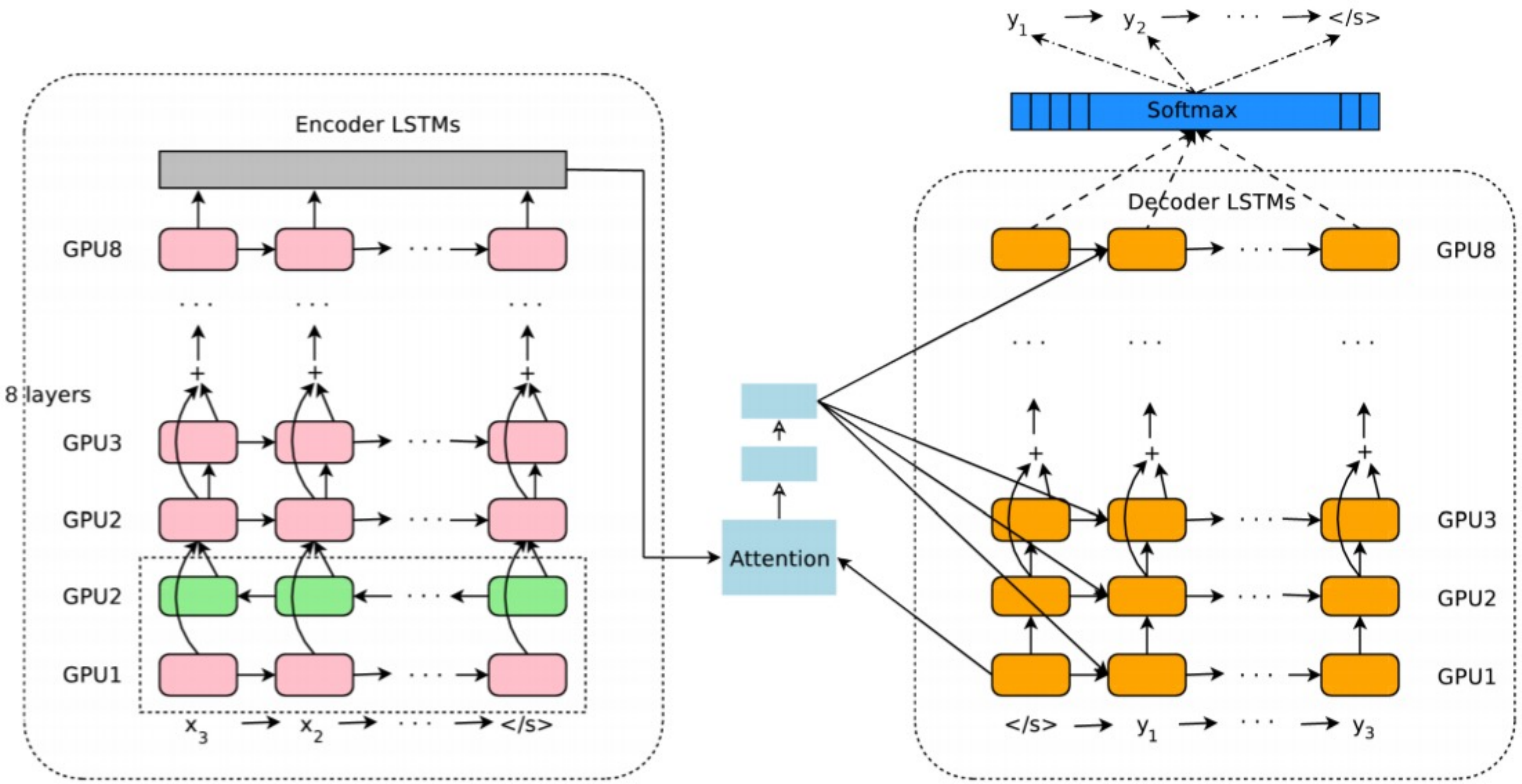


Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

ATTENTION

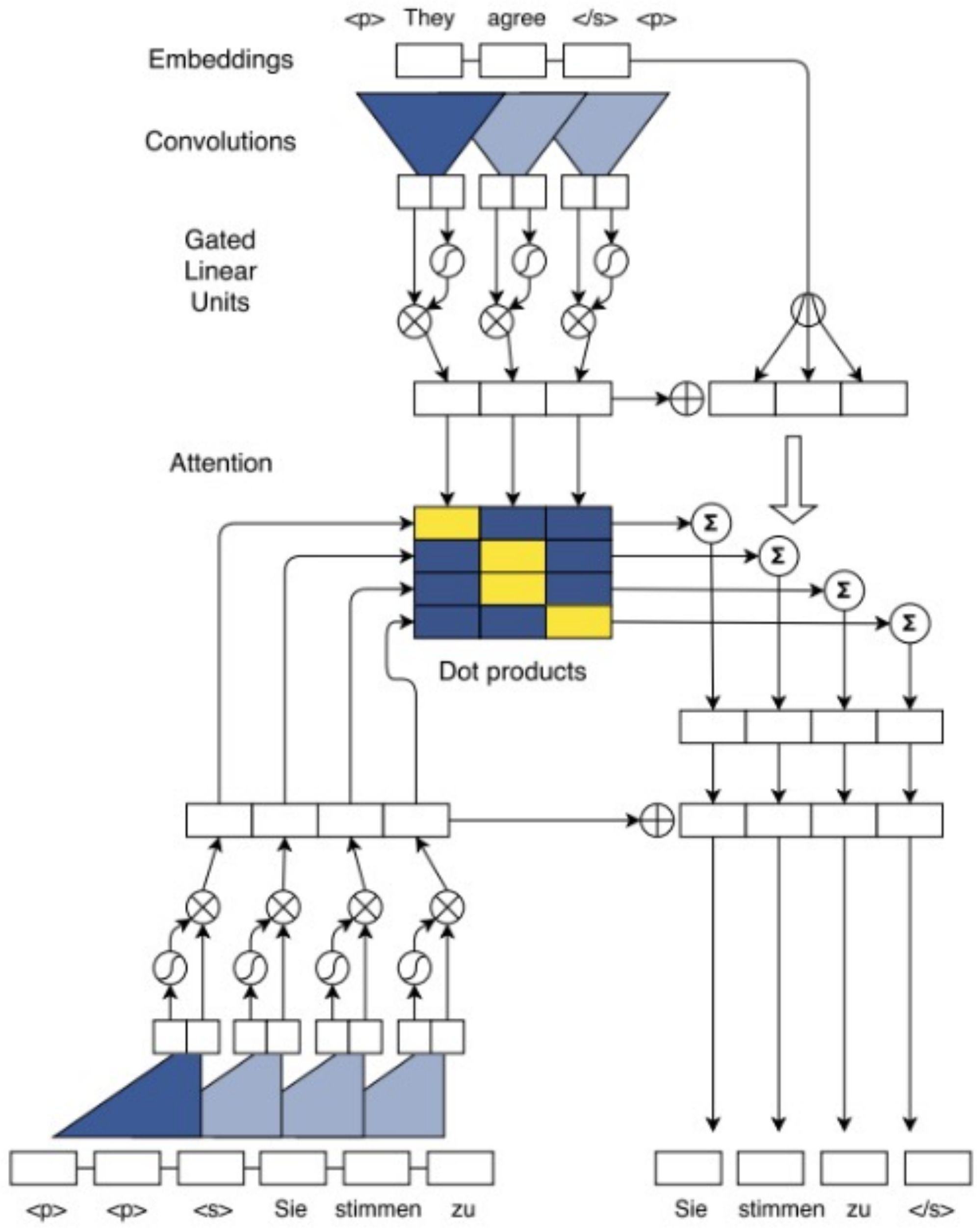
Examples



Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

ATTENTION

Examples



Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017, July). Convolutional sequence to sequence learning. In *International conference on machine learning* (pp. 1243-1252). PMLR.

Part 1: Machine Learning in NLP

- Lecture
    - What is NLP?
    - Problem Formulation
    - Text Representations
    - Dimensionality Reduction
    - Embeddings
    - RNNs
    - “Attention is All You Need”
- Lab
    - Transformer Architecture
    - BERT Model
    - Pretraining BERT

ATTENTION IS ALL YOU NEED

Design

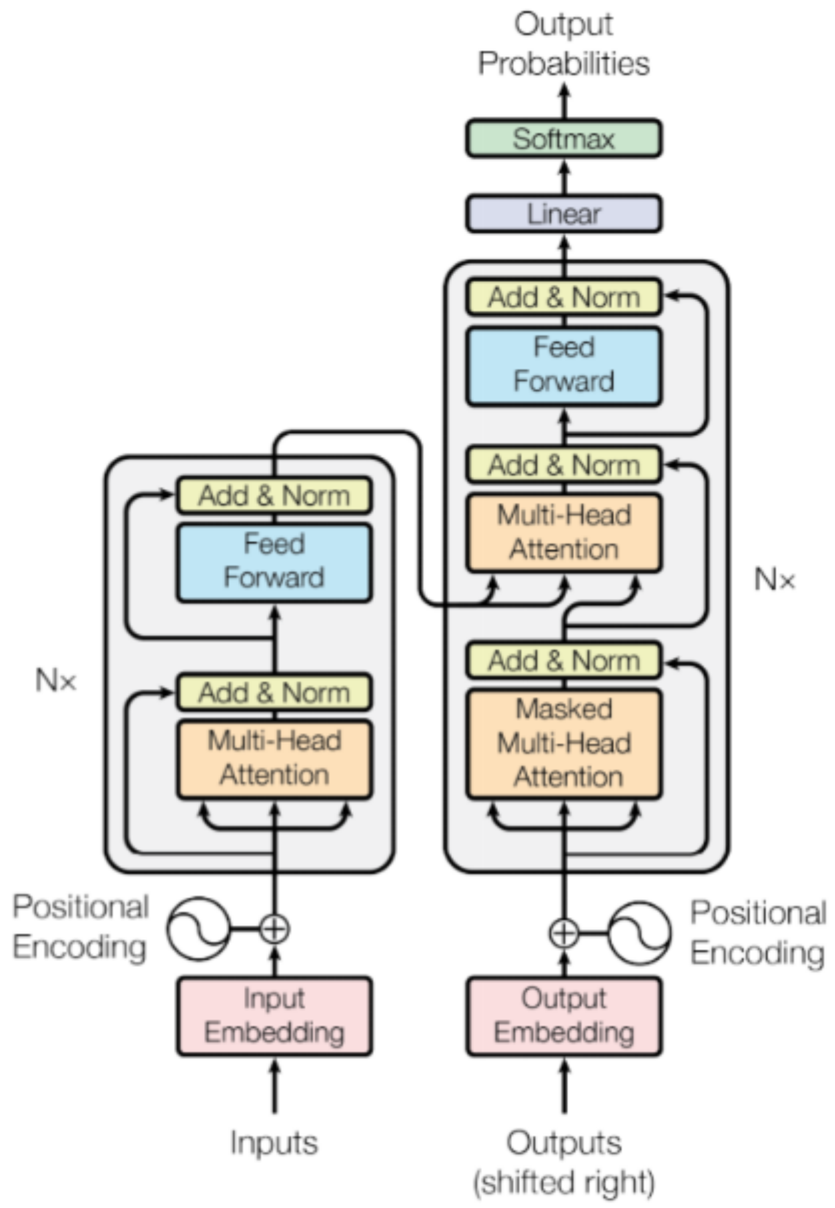
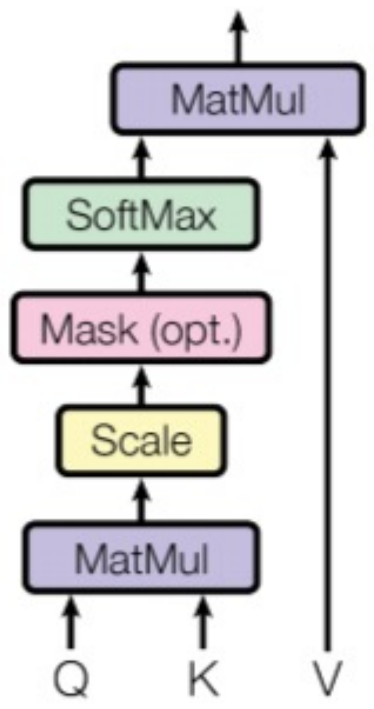


Figure 1: The Transformer - model architecture.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

ATTENTION IS ALL YOU NEED

Design


Scaled Dot-Product Attention

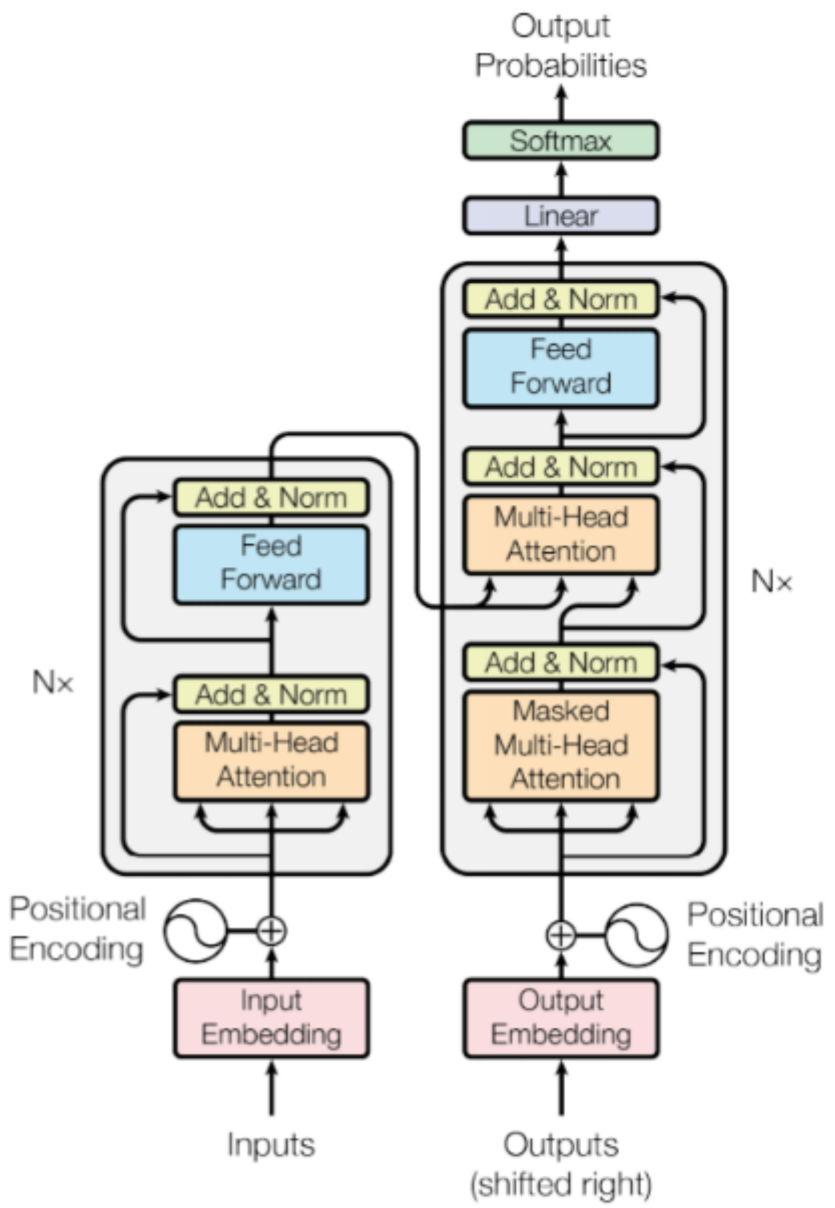
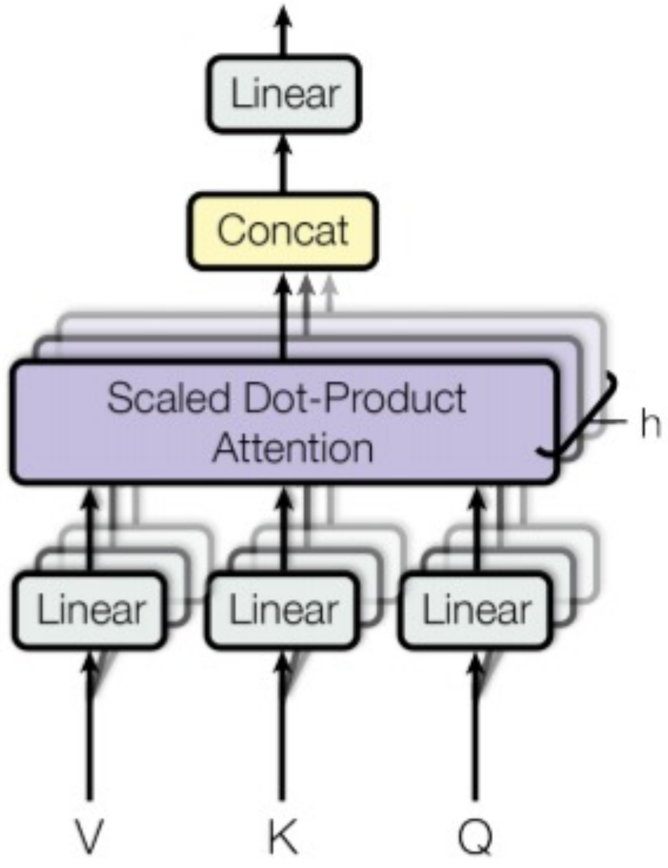


Figure 1: The Transformer - model architecture.


Multi-Head Attention

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

WAS IT A BREAKTHROUGH IN ITSELF?

ATTENTION IS ALL YOU NEED

Not a breakthrough in itself

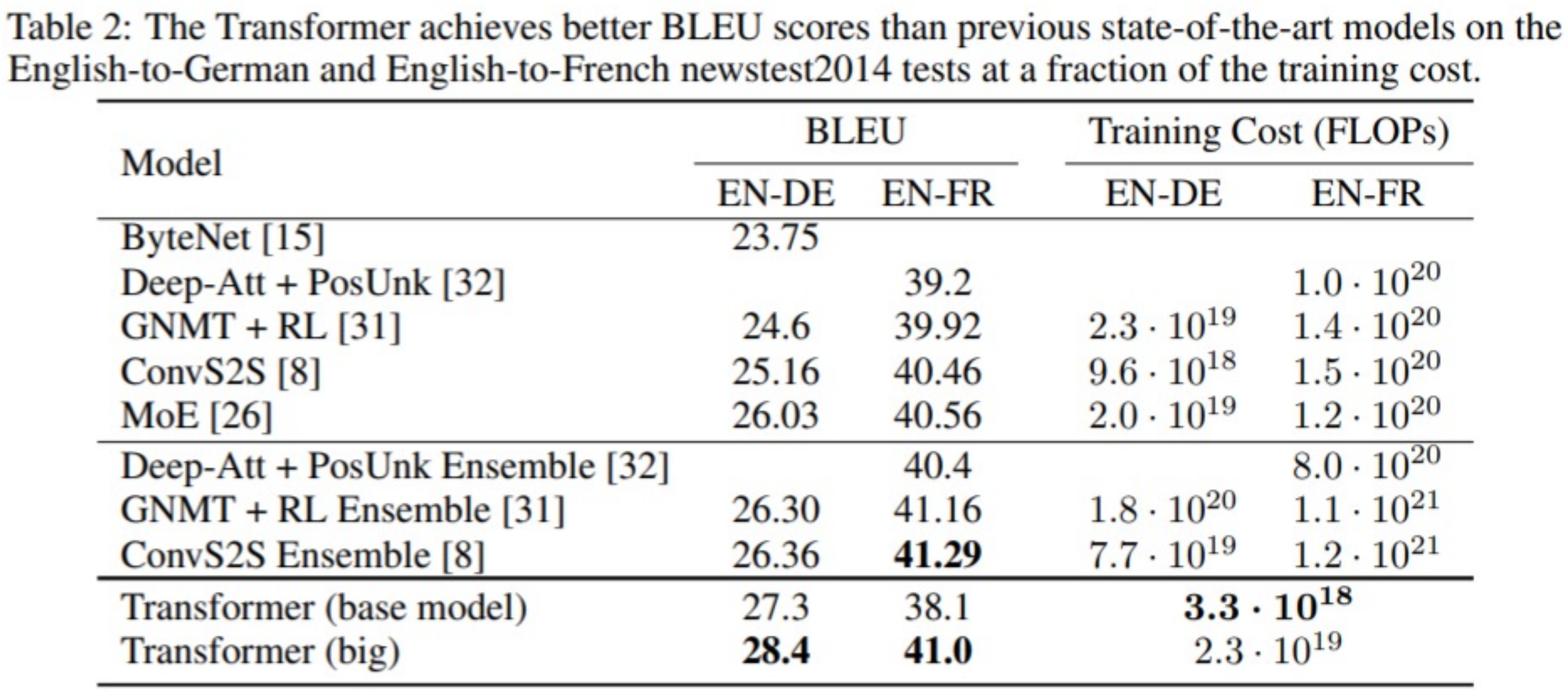


Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

ATTENTION IS ALL YOU NEED

But …

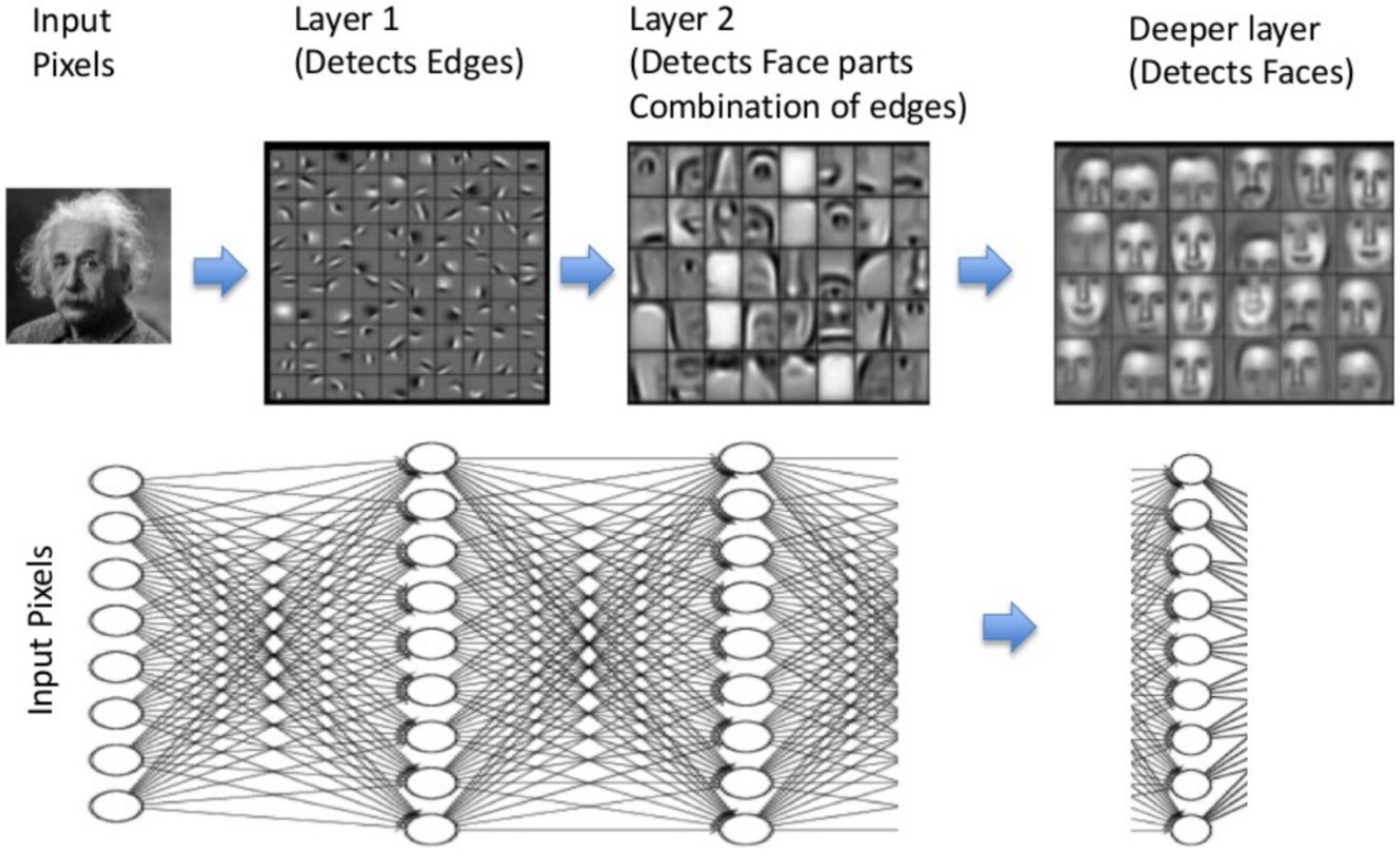
“ … the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers.”

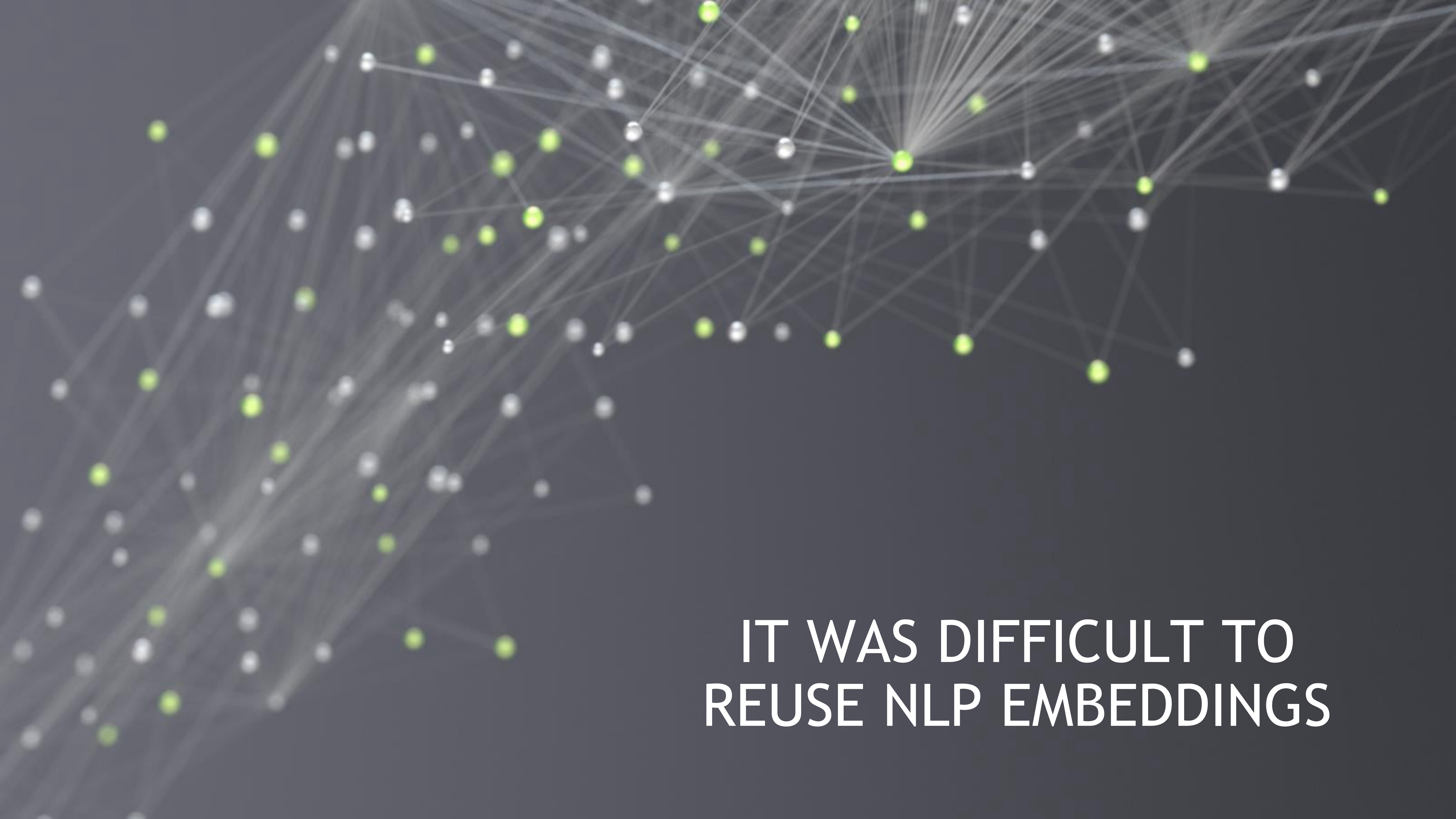
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

NEURAL EMBEDDINGS

FEATURE REUSE

The opportunity

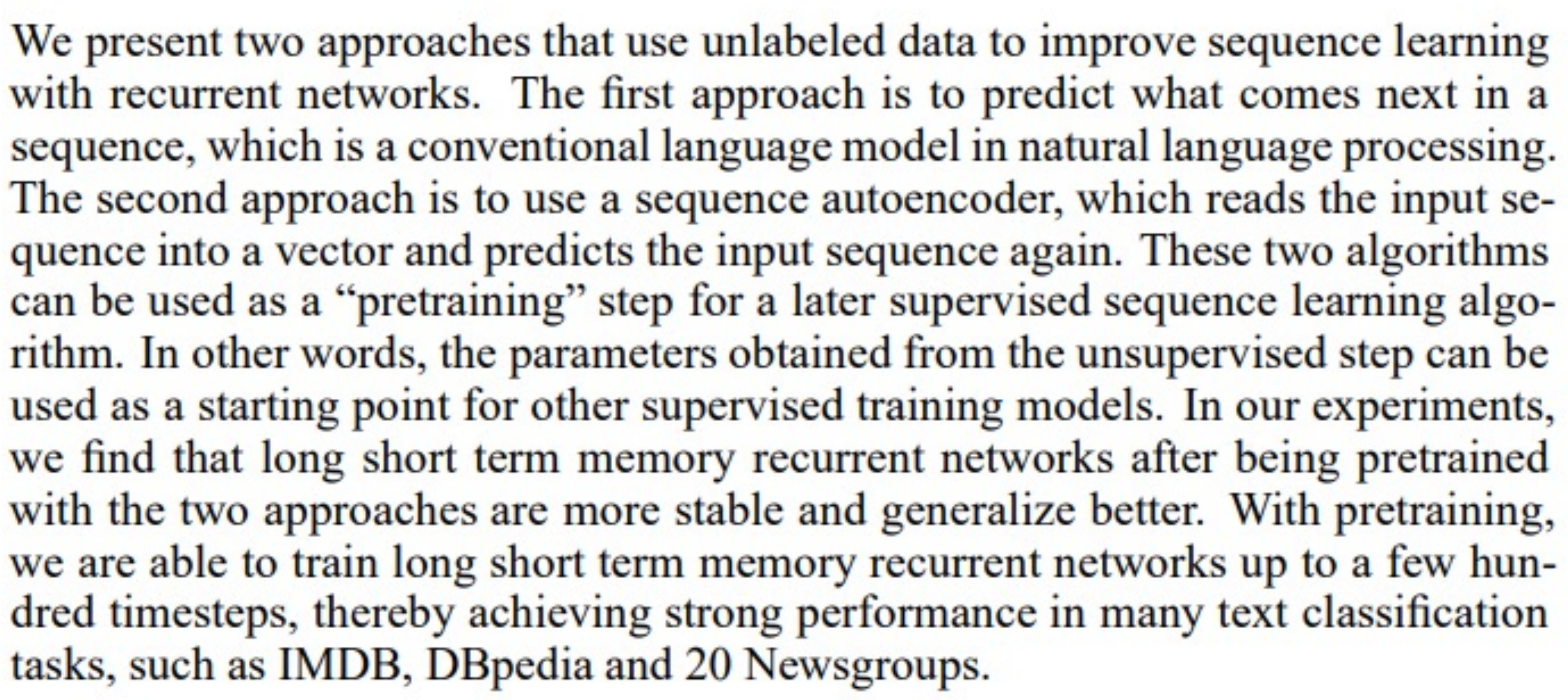




IT WAS DIFFICULT TO REUSE NLP EMBEDDINGS

SEMI-SUPERVISED SEQUENCE LEARNING

More complex representations



We present two approaches that use unlabeled data to improve sequence learning with recurrent networks. The first approach is to predict what comes next in a sequence, which is a conventional language model in natural language processing. The second approach is to use a sequence autoencoder, which reads the input sequence into a vector and predicts the input sequence again. These two algorithms can be used as a "pretraining" step for a later supervised sequence learning algorithm. In other words, the parameters obtained from the unsupervised step can be used as a starting point for other supervised training models. In our experiments, we find that long short term memory recurrent networks after being pretrained with the two approaches are more stable and generalize better. With pretraining, we are able to train long short term memory recurrent networks up to a few hundred timesteps, thereby achieving strong performance in many text classification tasks, such as IMDB, DBpedia and 20 Newsgroups.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in neural information processing systems* (pp. 3079-3087).

SEMI-SUPERVISED SEQUENCE LEARNING

More complex representations

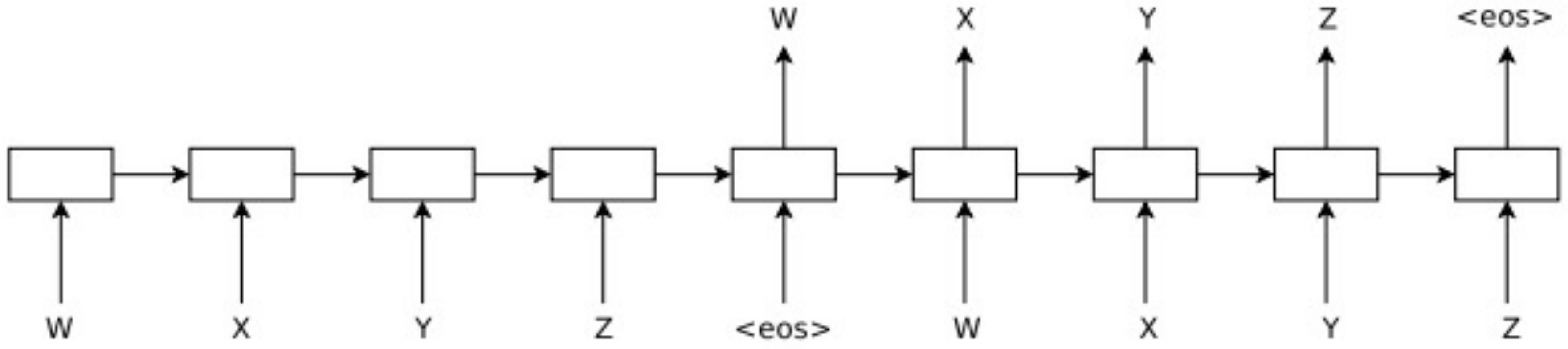
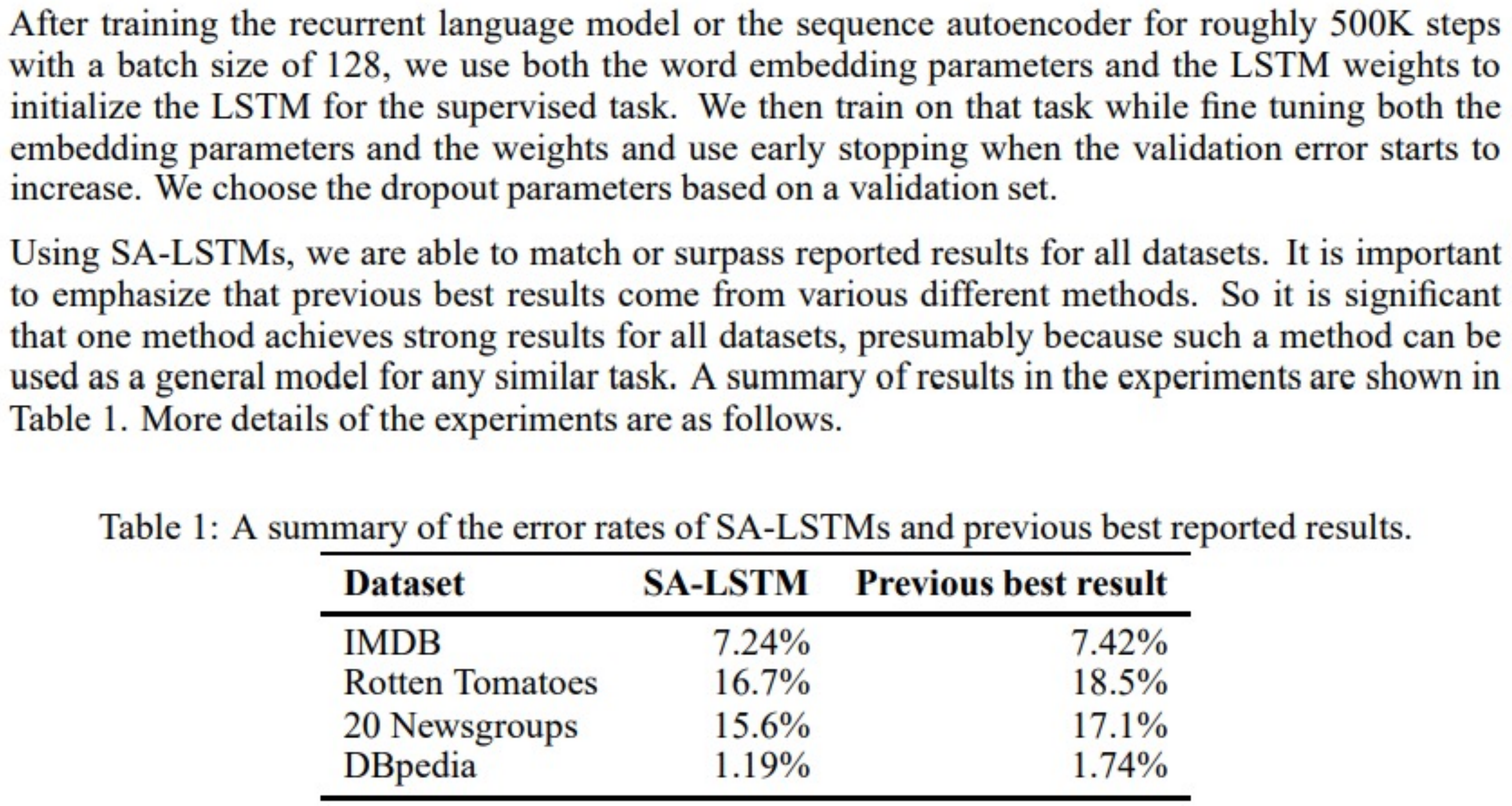


Figure 1: The sequence autoencoder for the sequence "WXYZ". The sequence autoencoder uses a recurrent network to read the input sequence in to the hidden state, which can then be used to reconstruct the original sequence.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in neural information processing systems* (pp. 3079-3087).

SEMI-SUPERVISED SEQUENCE LEARNING

More complex representations



After training the recurrent language model or the sequence autoencoder for roughly 500K steps with a batch size of 128, we use both the word embedding parameters and the LSTM weights to initialize the LSTM for the supervised task. We then train on that task while fine tuning both the embedding parameters and the weights and use early stopping when the validation error starts to increase. We choose the dropout parameters based on a validation set.

Using SA-LSTMs, we are able to match or surpass reported results for all datasets. It is important to emphasize that previous best results come from various different methods. So it is significant that one method achieves strong results for all datasets, presumably because such a method can be used as a general model for any similar task. A summary of results in the experiments are shown in Table 1. More details of the experiments are as follows.

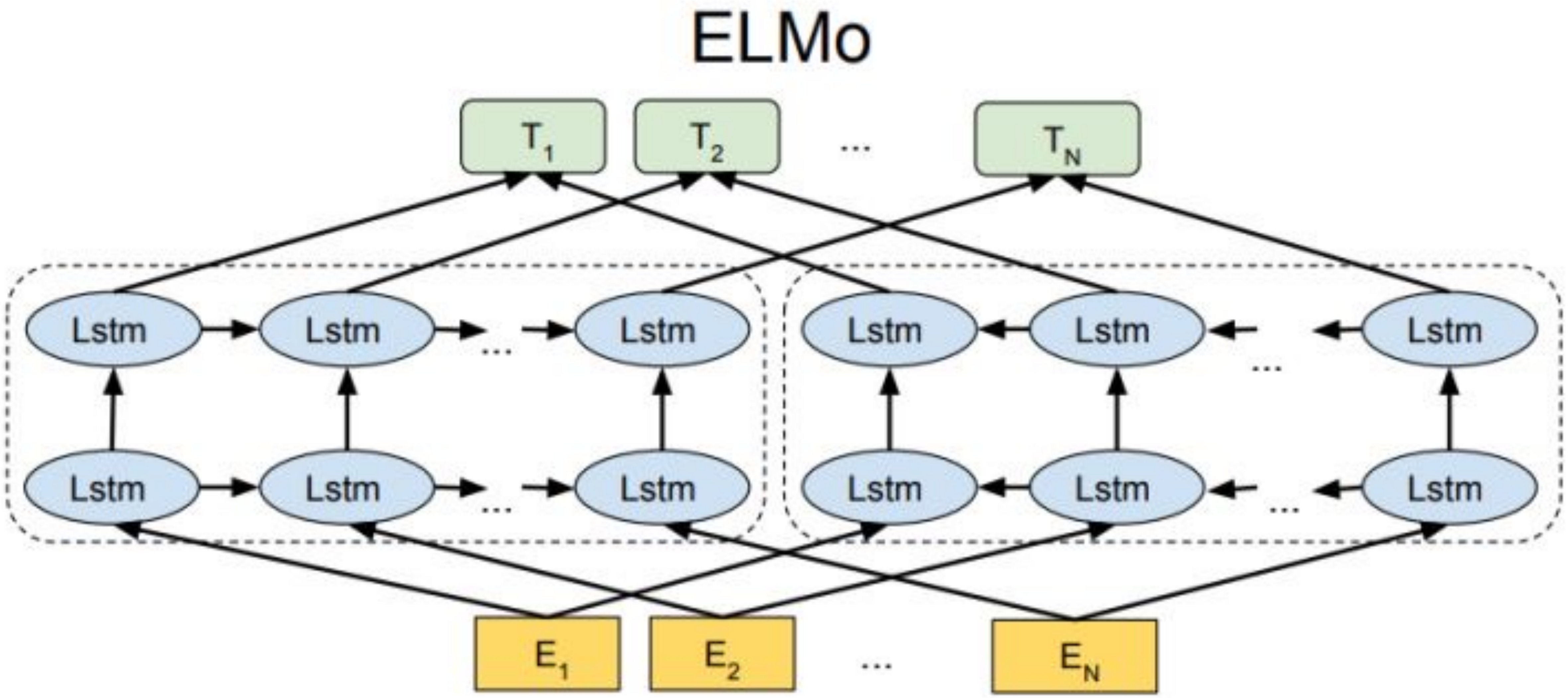
Table 1: A summary of the error rates of SA-LSTMs and previous best reported results.

Dataset	**SA-LSTM**	**Previous best result**
IMDB	7.24%	7.42%
Rotten Tomatoes	16.7%	18.5%
20 Newsgroups	15.6%	17.1%
DBpedia	1.19%	1.74%

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in neural information processing systems* (pp. 3079-3087).

ELMO

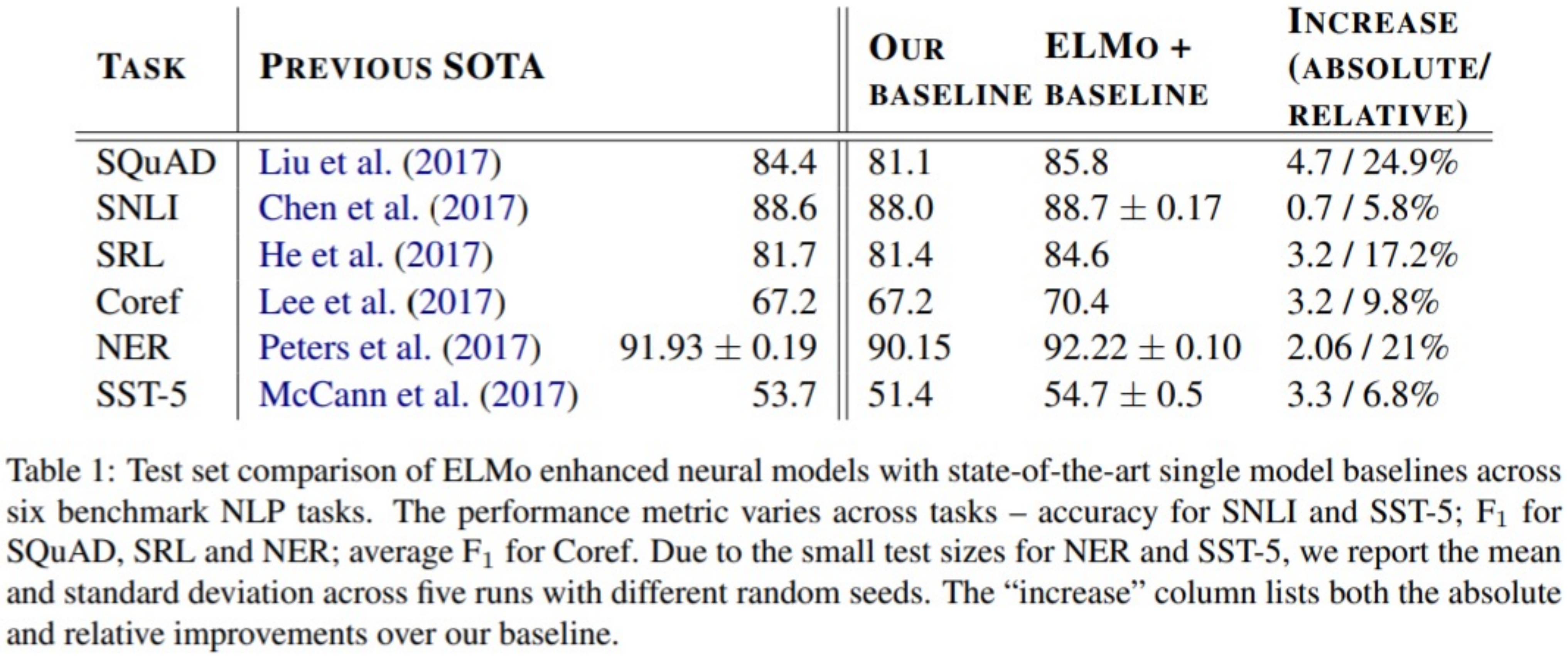
Embeddings for Language Models



Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

ELMO

Embeddings for Language Models



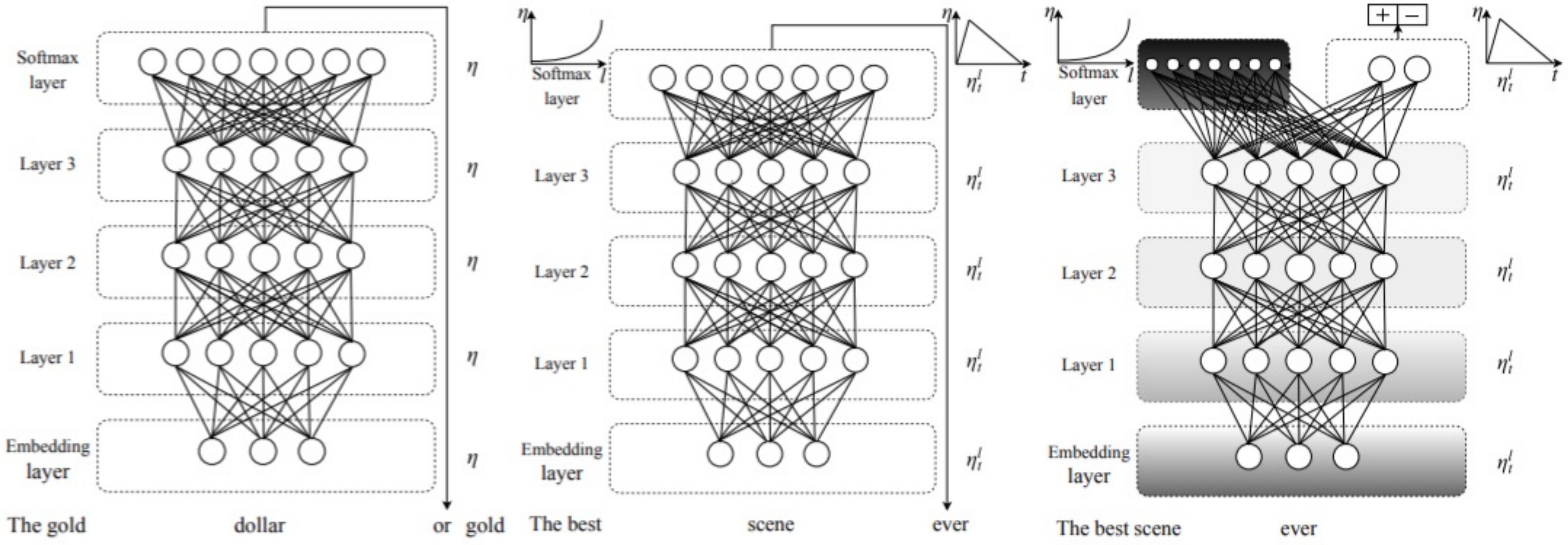
TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

ULM-FIT

Universal Language Model Fine-Tuning for Text Classification



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning (‘*Discr*’) and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, ‘*Discr*’, and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

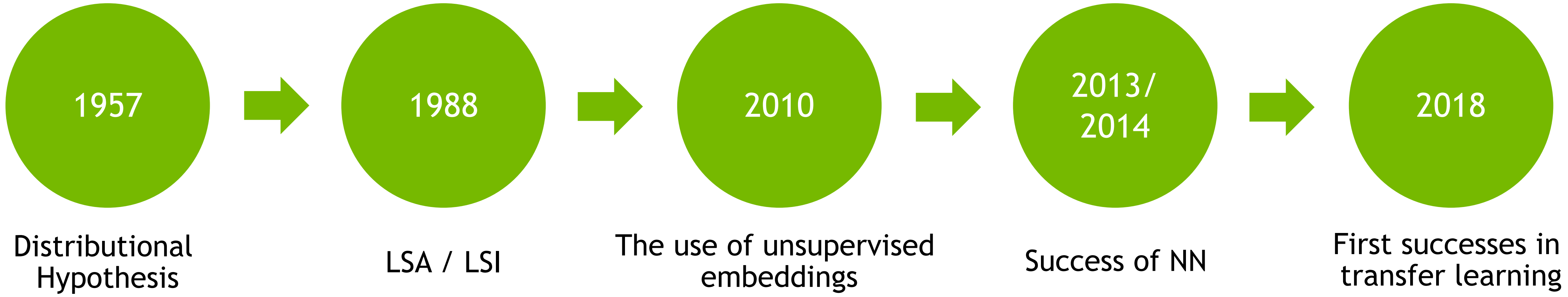
Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

TRANSFER LEARNING IN NLP

Not trivial to use and not universally applicable

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.



1957
Distributional Hypothesis

1988
LSA / LSI

2010
The use of unsupervised embeddings

2013/ 2014
Success of NN

2018
First successes in transfer learning

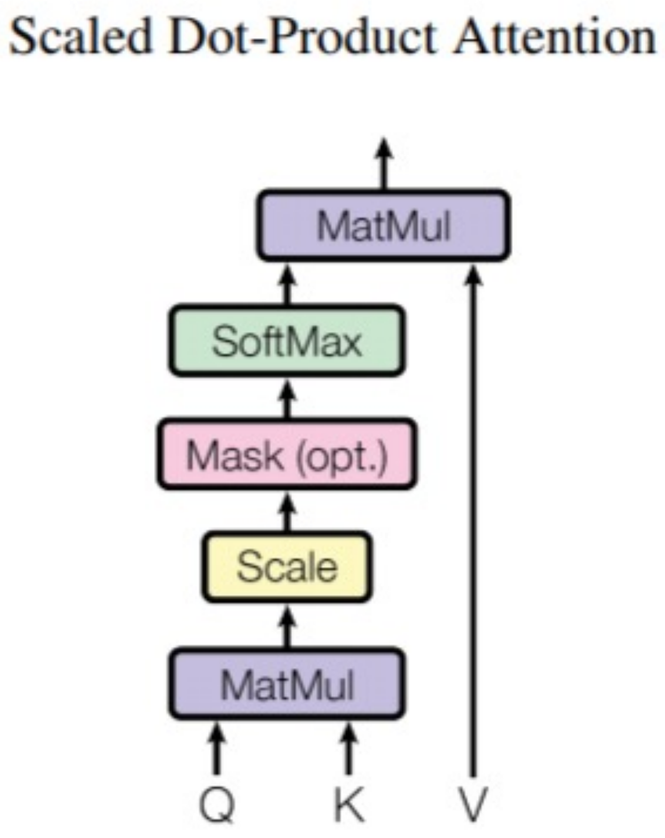
THIS CREATED A FOUNDATION FOR THE NEW NLP MODELS

(DISCUSSED IN THE NEXT CLASS)

THE LAB

ATTENTION IS ALL YOU NEED

Deep dive into the transformer design


Scaled Dot-Product Attention

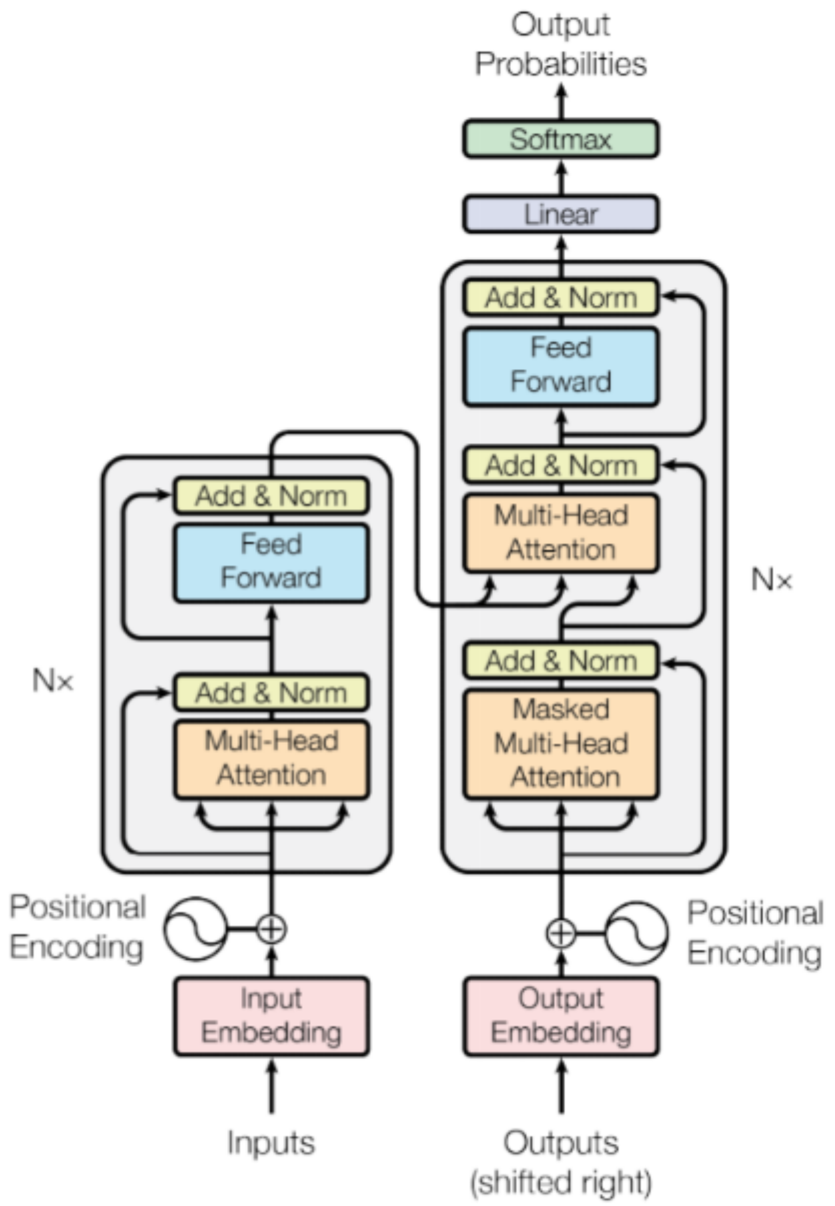
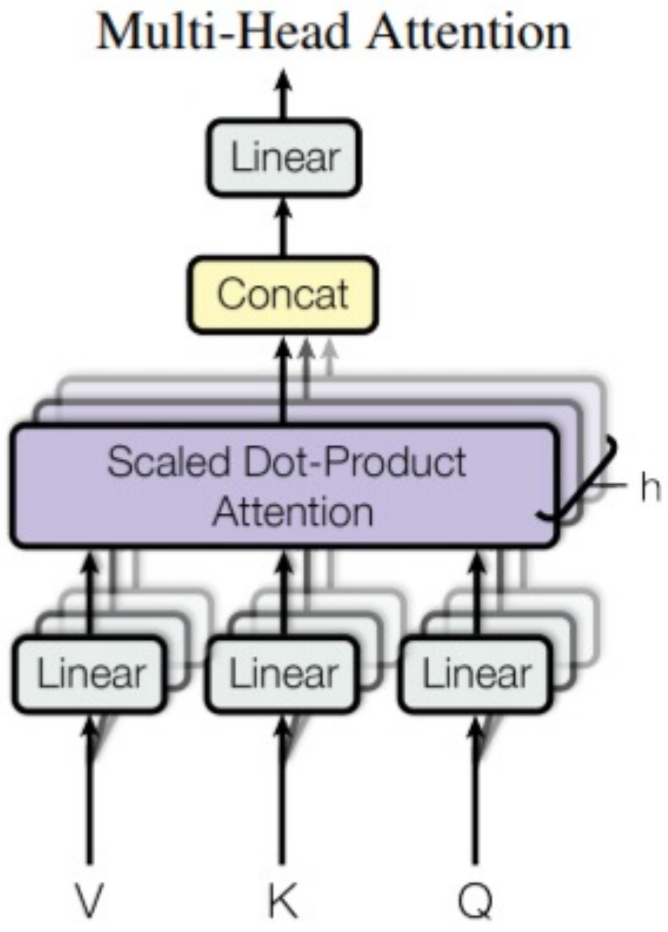


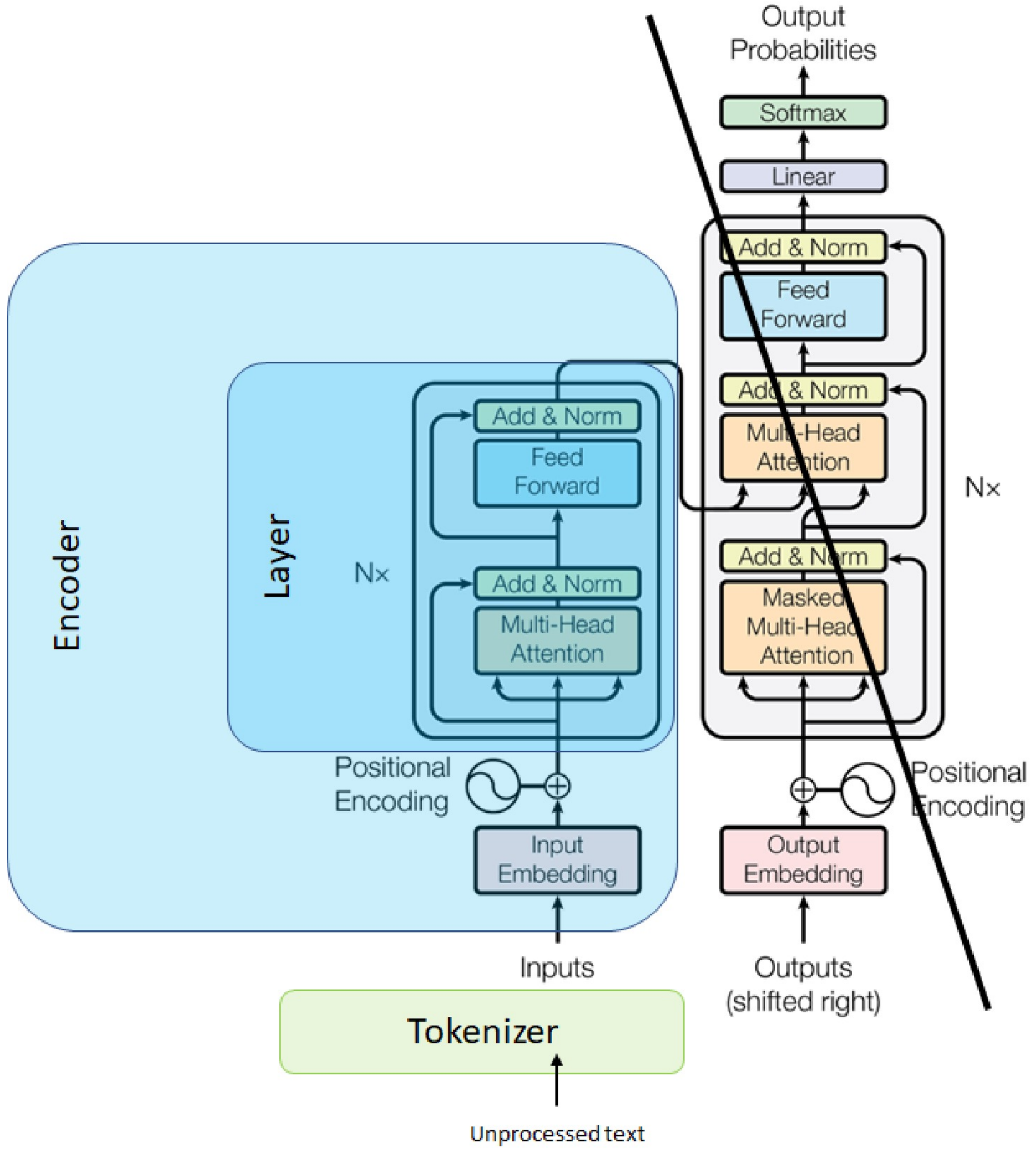
Figure 1: The Transformer - model architecture.


Multi-Head Attention

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

BERT

How it relates to transformer and pretraining



IN THE NEXT CLASS…

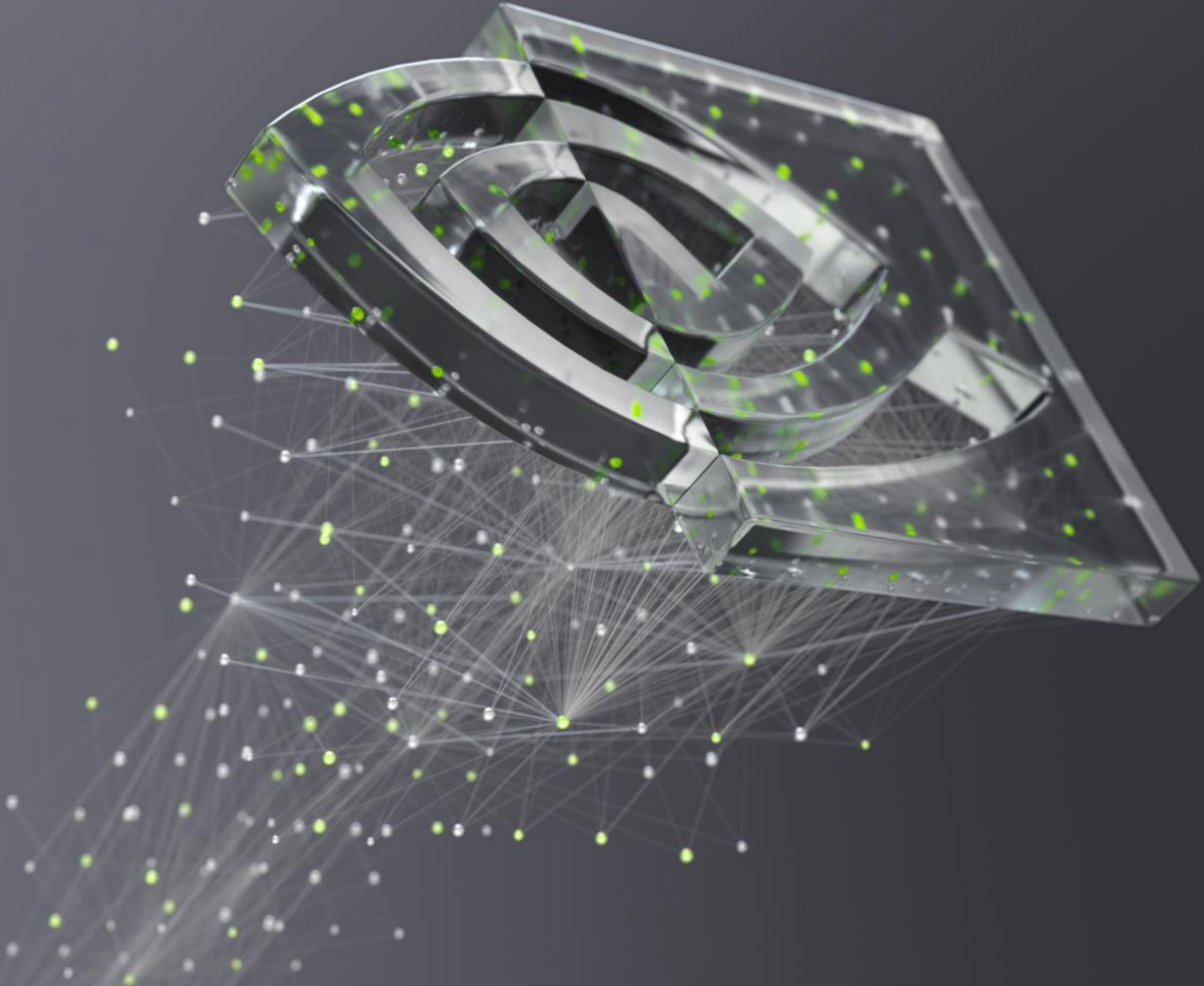
SELF-SUPERVISION, BERT, AND BEYOND

Why did models start to work well? What does the future hold?

?

Part 1: Machine Learning in NLP

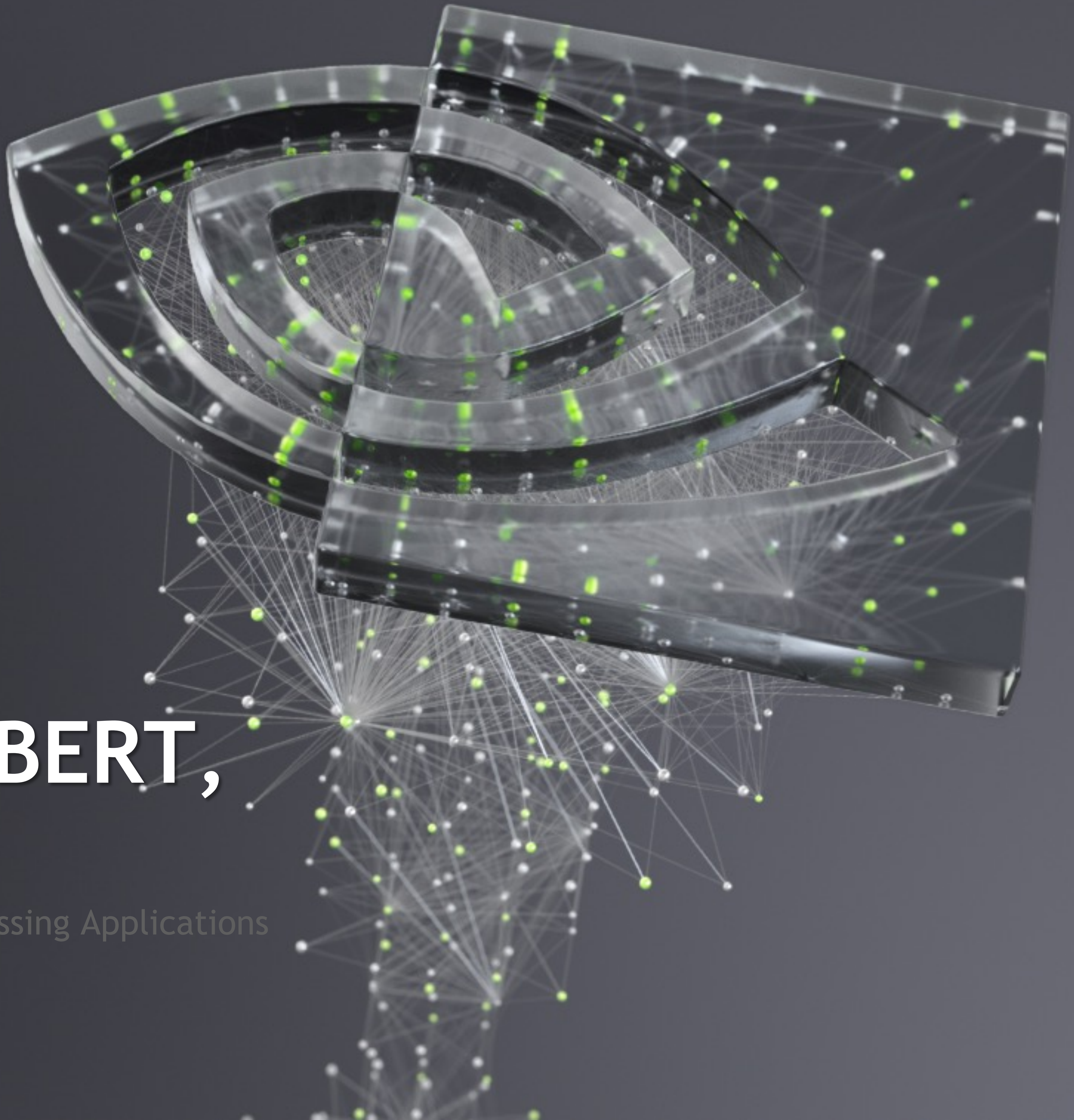
- Lecture
  - What is NLP?
  - Problem Formulation
  - Text Representations
  - Dimensionality Reduction
  - Embeddings
  - RNNs
  - “Attention is All You Need”
- Lab
  - Transformer Architecture
  - BERT Model
  - Pretraining BERT



NVIDIA

DEEP
LEARNING
INSTITUTE

DEEP
LEARNING
INSTITUTE



SELF-SUPERVISION, BERT, AND BEYOND

Building Transformer-Based Natural Language Processing Applications (Part 2)

FULL COURSE AGENDA

Part 1: Machine Learning in NLP

Lecture: NLP background and the role of DNNs leading to the Transformer architecture

Lab: Tutorial-style exploration of a ***translation task*** using the Transformer architecture

---

Part 2: Self-Supervision, BERT, and Beyond

Lecture: Discussion of how language models with self-supervision have moved beyond the basic Transformer to BERT and ever larger models

Lab: Practical hands-on guide to the NVIDIA NeMo API and exercises to build a ***text classification task*** and a ***named entity recognition task*** using BERT-based language models

---

Part 3: Production Deployment

Lecture: Discussion of production deployment considerations and NVIDIA Triton Inference Server

Lab: Hands-on deployment of an example ***question answering task*** to NVIDIA Triton

Part 2: Self-Supervision, BERT and Beyond

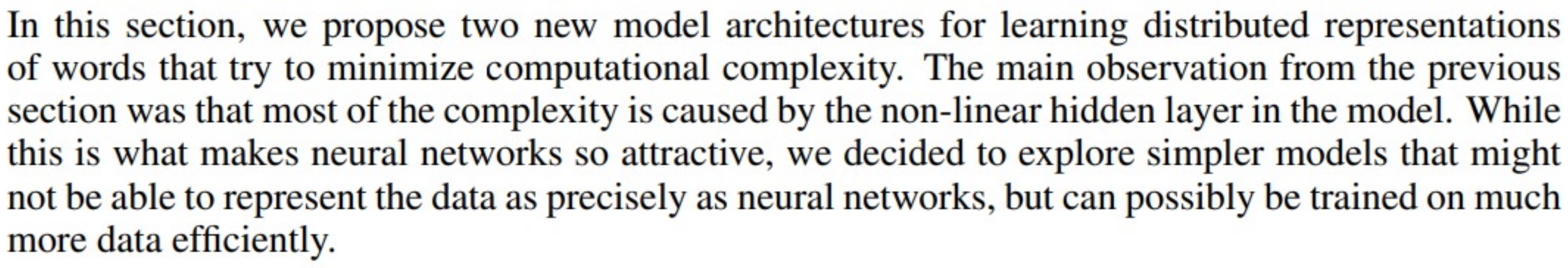
- Lecture
  - Why Do DNNs Work Well?
  - Self-Supervised Learning
  - BERT
- Lab
  - Explore the Data
  - Explore NeMo
  - Text Classifier Project
- Lecture (cont’d)
  - The Scaling Laws
  - Can and should we go even bigger?
- Lab (cont’d)
  - Named Entity Recognizer

Part 2: Self-Supervision, BERT and Beyond

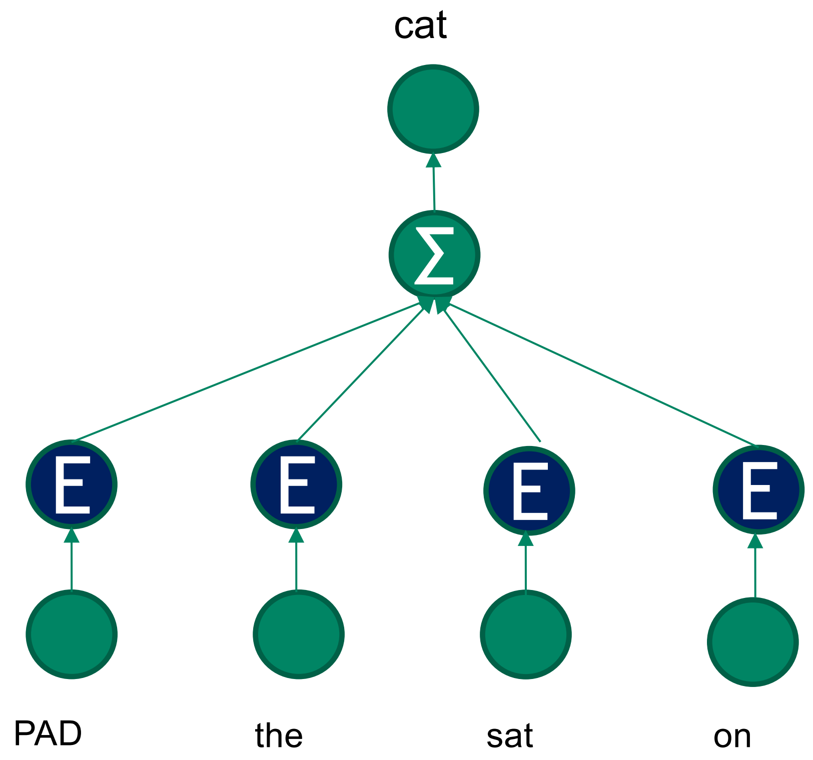
- Lecture
  - Why Do DNNs Work Well?
  - Self-Supervised Learning
  - BERT
- Lab
  - Explore the Data
  - Explore NeMo
  - Text Classifier Project
- Lecture (cont’d)
  - The Scaling Laws
  - Can and should we go even bigger?
- Lab (cont’d)
  - Named Entity Recognizer

COMPUTE

Or lack of thereof



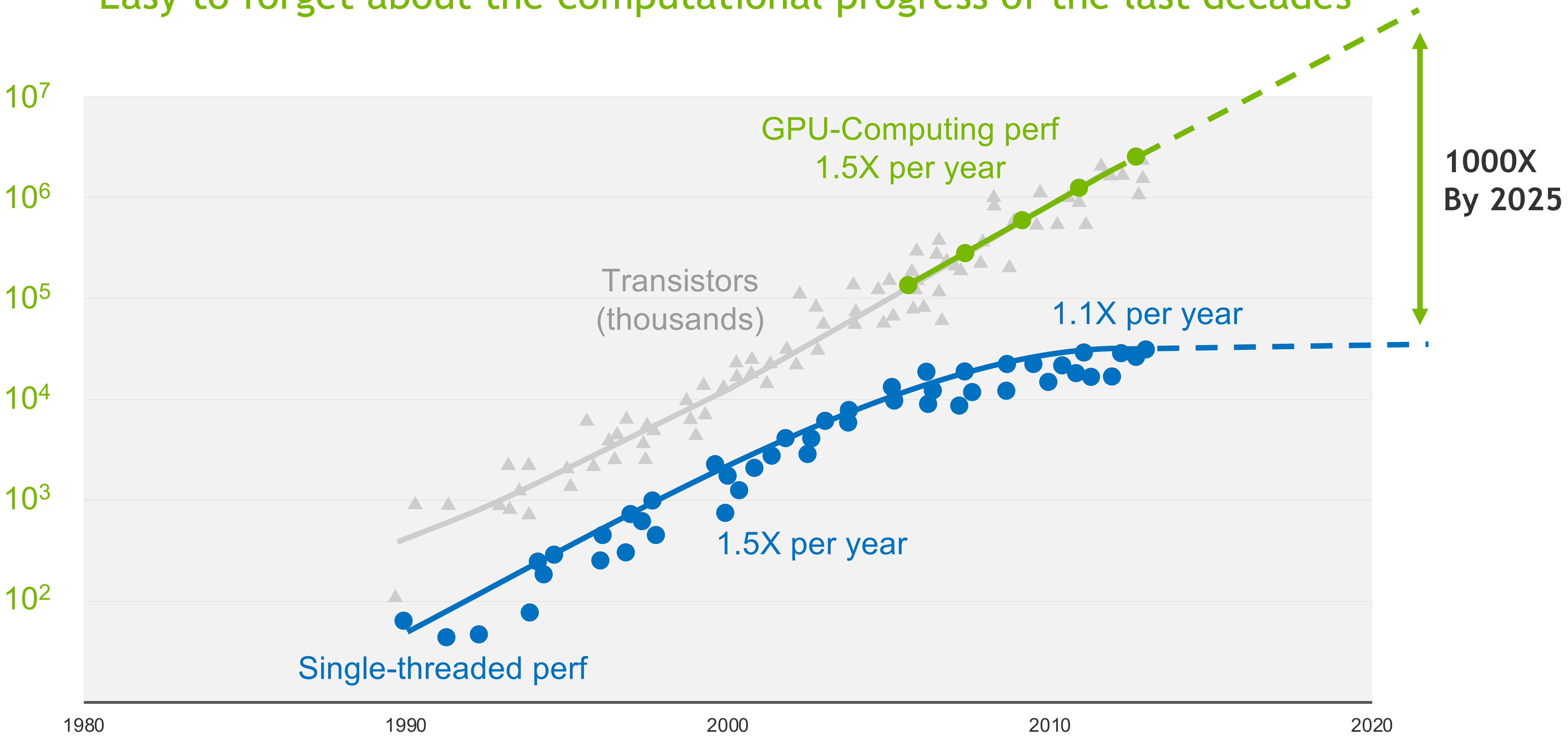
In this section, we propose two new model architectures for learning distributed representations of words that try to minimize computational complexity. The main observation from the previous section was that most of the complexity is caused by the non-linear hidden layer in the model. While this is what makes neural networks so attractive, we decided to explore simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.



Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

COMPUTE

Easy to forget about the computational progress of the last decades



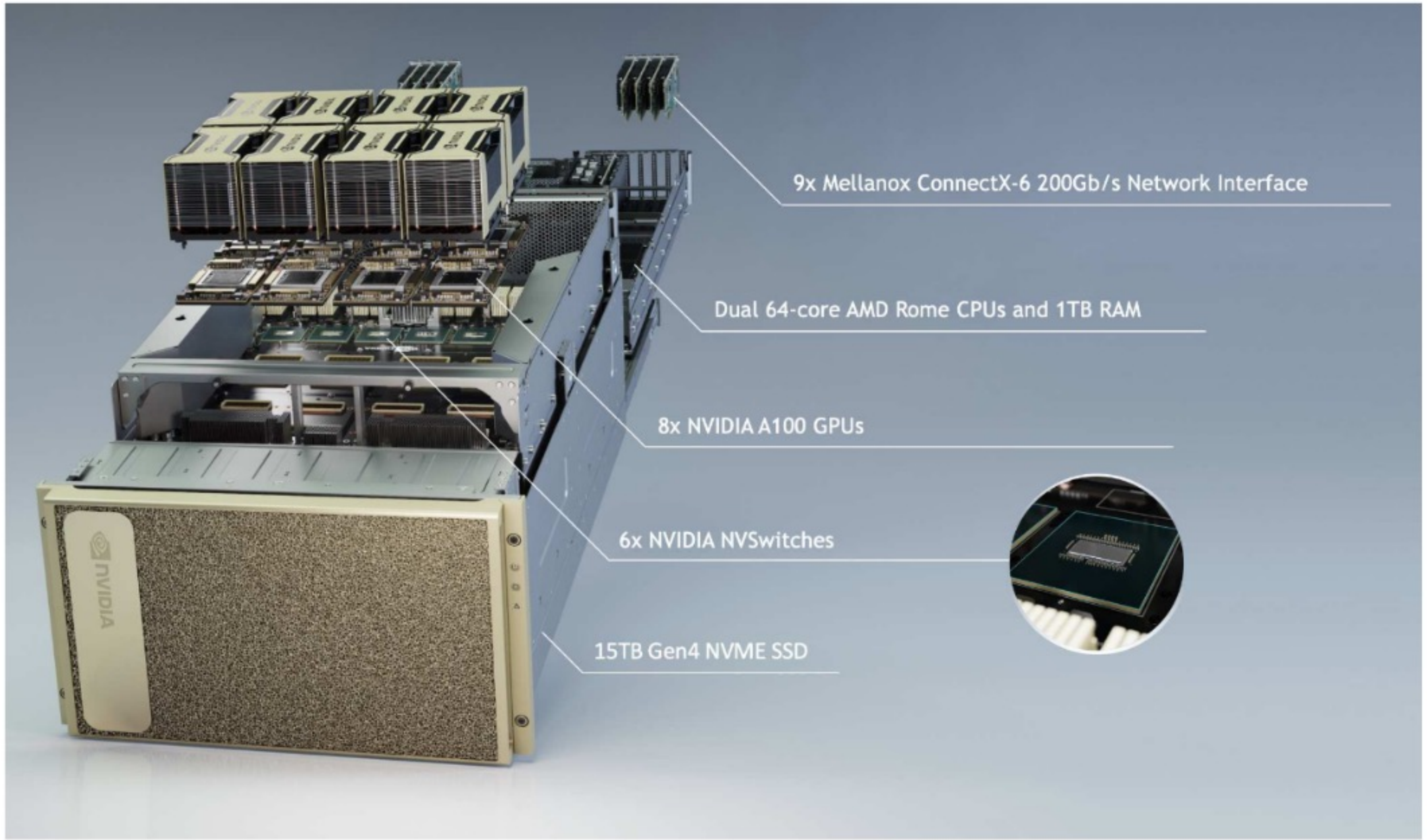
CONTEXT

CONTEXT

8 petaFLOPs in June 2011 (K Computer)

CONTEXT

5 petaFLOPs for AI - today


9x Mellanox ConnectX-6 200Gb/s Network Interface
Dual 64-core AMD Rome CPUs and 1TB RAM
8x NVIDIA A100 GPUs
6x NVIDIA NVSwitches
15TB Gen4 NVME SSD

CONTEXT

~100 PFLOPS (FP16) or 48 PFLOPS (TF32) for AI - today

100 EXAFLOPS

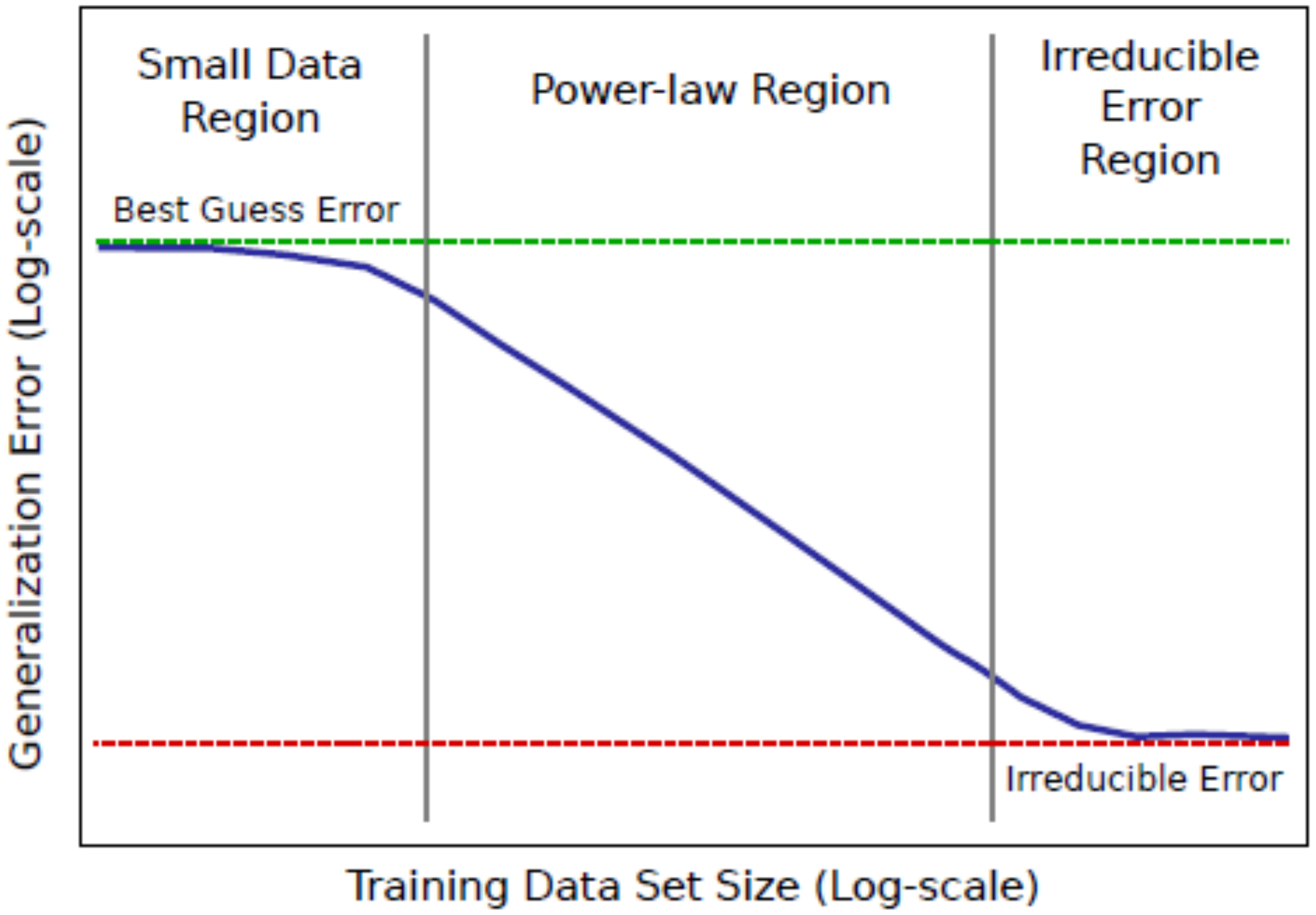
~=

2 YEARS ON A DUAL CPU SERVER

SCALING LAWS

SCALING LAWS

Power Law relationship between the dataset size and accuracy

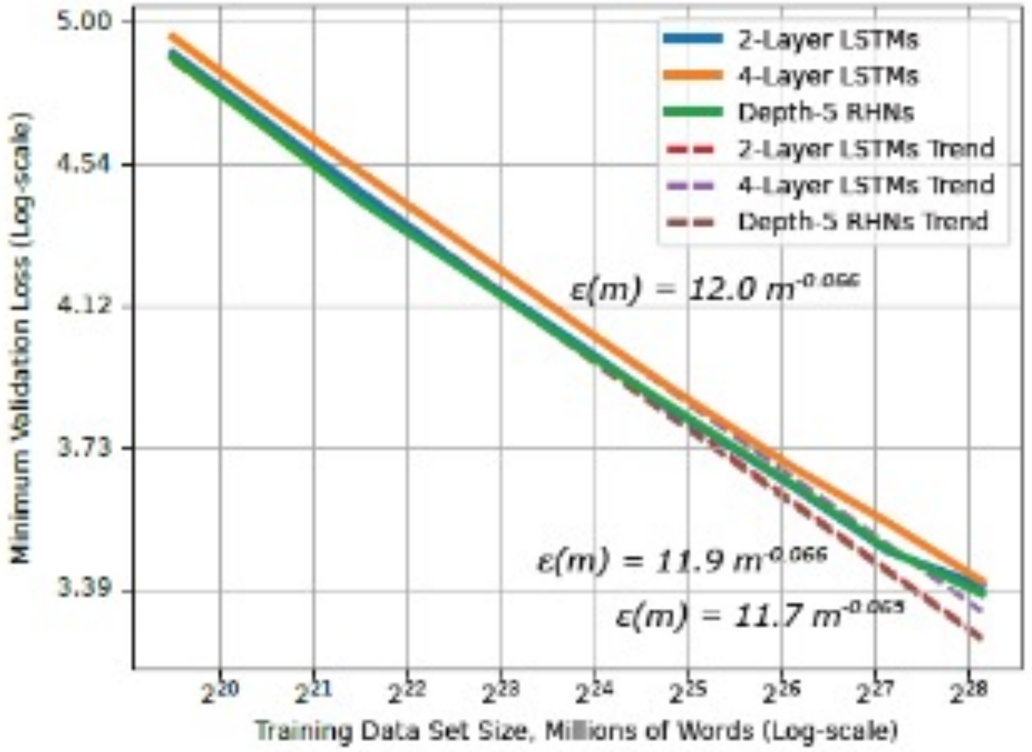


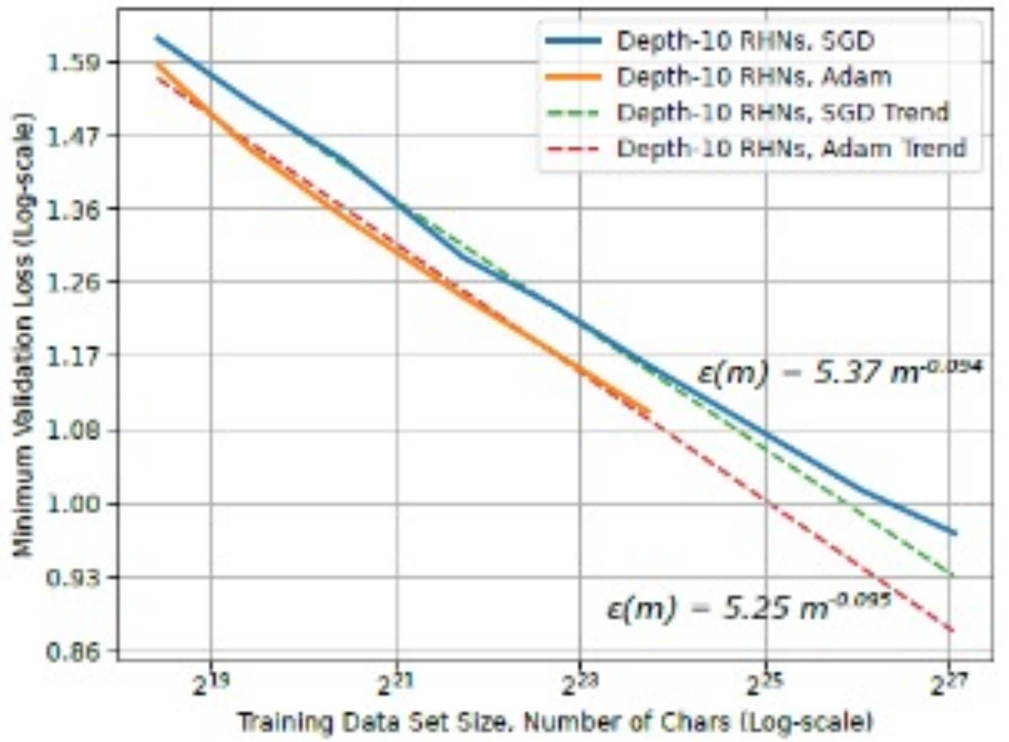
Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., ... & Zhou, Y. (2017). Deep Learning Scaling is Predictable, Empirically. arXiv preprint arXiv:1712.00409.

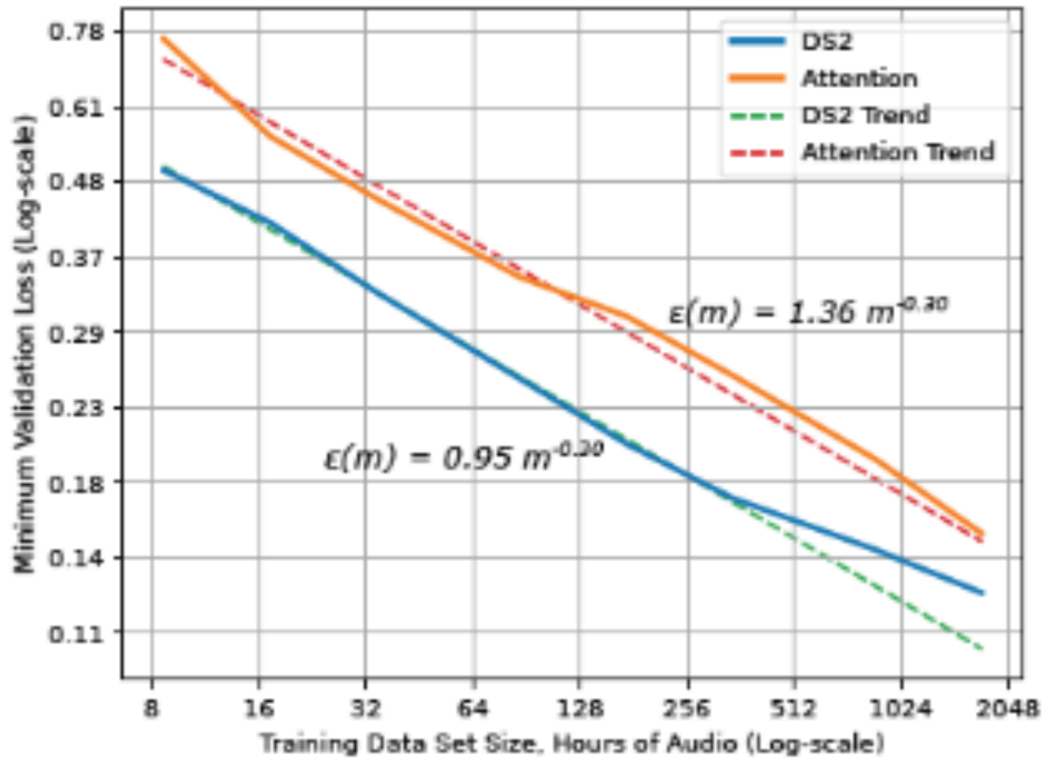
SCALING LAWS

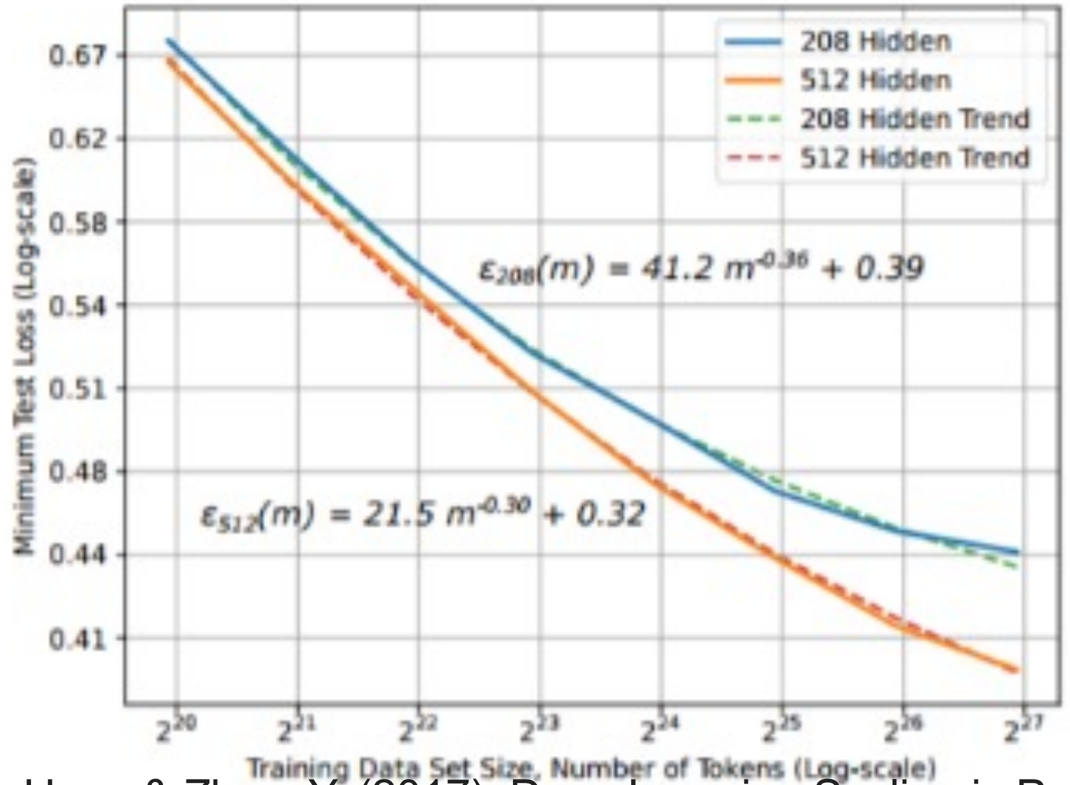
Applicable across all AI tasks

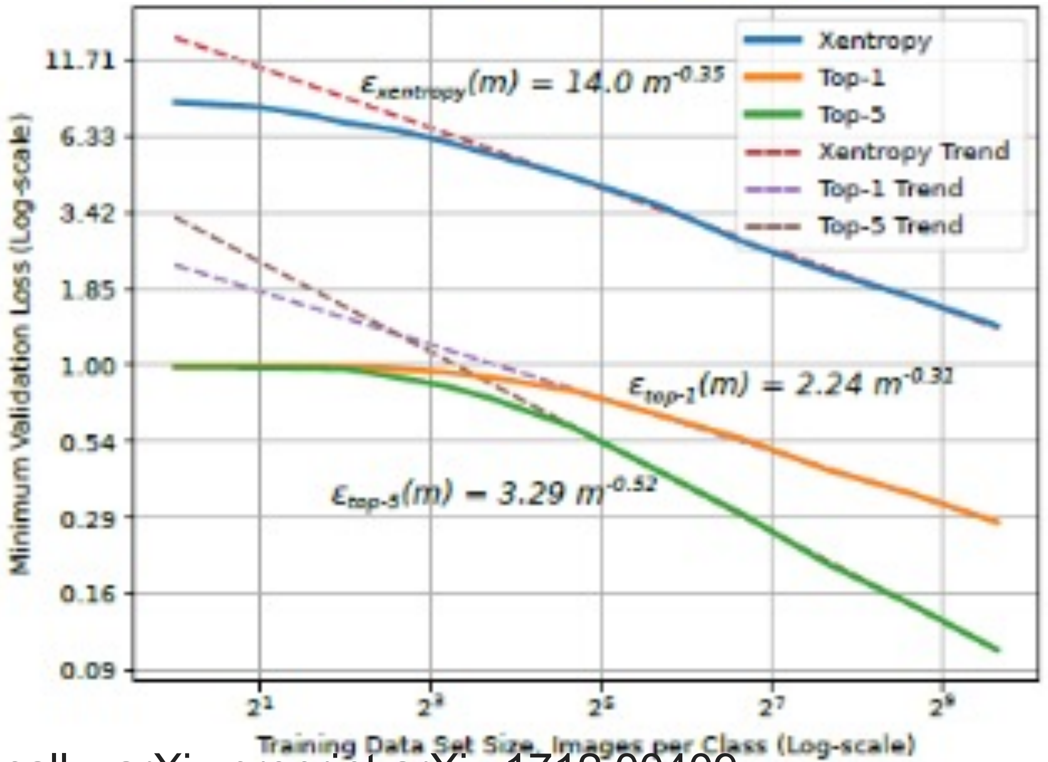
- Translation
- Language Models
- Character Language Models
- Image Classification
- Attention Speech Models









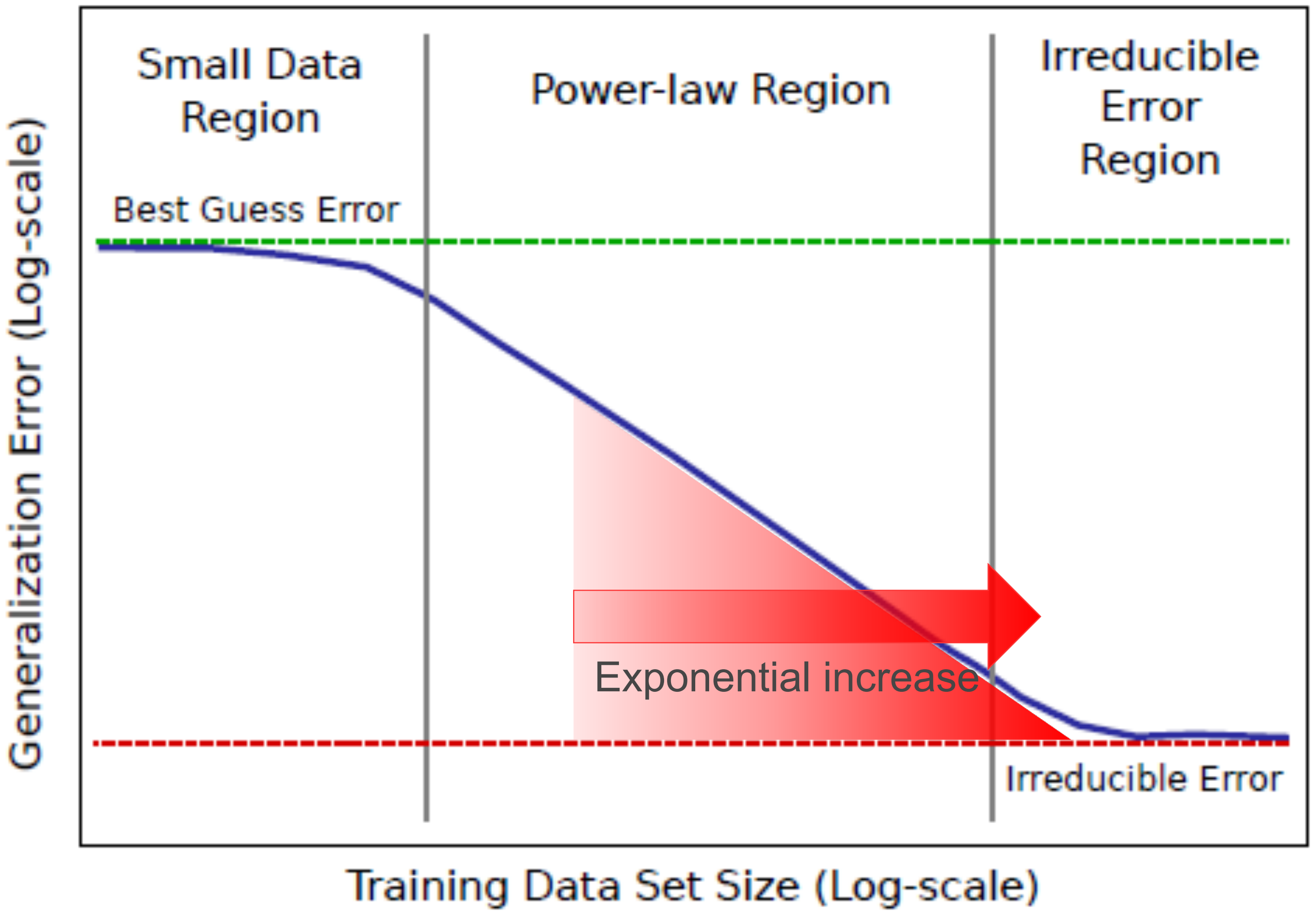


Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., ... & Zhou, Y. (2017). Deep Learning Scaling is Predictable, Empirically. arXiv preprint arXiv:1712.00409.

THE COST

THE COST OF LABELING

Limits the utility of deep learning models



Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., ... & Zhou, Y. (2017). Deep Learning Scaling is Predictable, Empirically. arXiv preprint arXiv:1712.00409

Part 2: Self-Supervision, BERT and Beyond

- Lecture
  - Why Do DNNs Work Well?
  - Self-Supervised Learning
  - BERT
- Lab
  - Explore the Data
  - Explore NeMo
  - Text Classifier Project
- Lecture (cont’d)
  - The Scaling Laws
  - Can and should we go even bigger?
- Lab (cont’d)
  - Named Entity Recognizer

SELF-SUPERVISED LEARNING

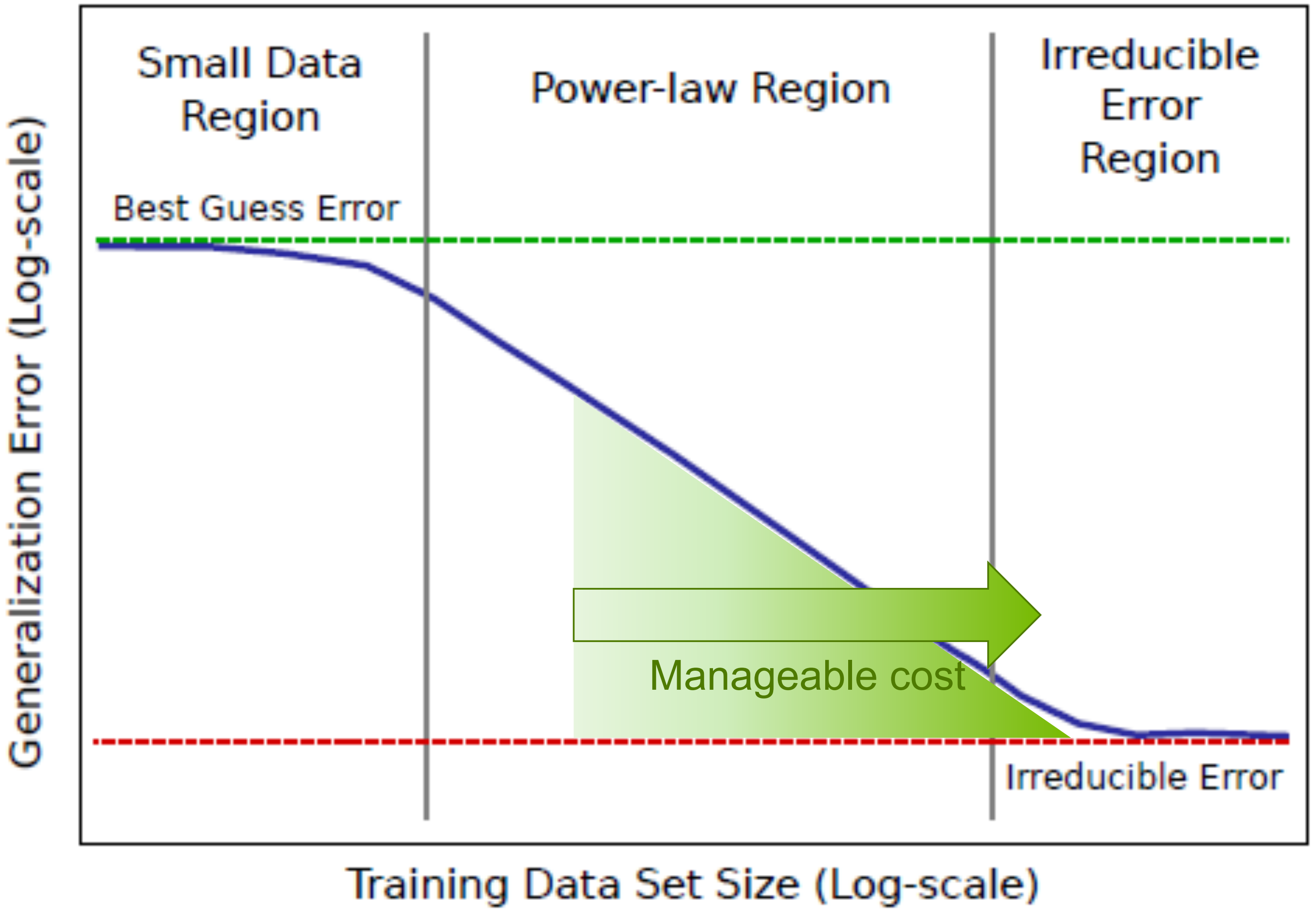
Example training tasks

- Natural Language Processing:
  - Masked Language Model: We mask a percentage of the input tokens at random (say 15%) and ask the neural network to predict the entire sentence
  - Next Sentence Prediction: We choose either two consecutive sentences from text, or two random sentences from the text. We ask the neural network to establish whether the two sentences occur one after another.
  - We use another simpler neural network to replace random words in the sequence and ask the primary neural network to detect which words were replaced (using a GAN like configuration).
- Computer Vision:
  - Contrastive Learning: Randomly modify (crop and resize, flip, distort color, rotate, cut-out, noise, blur, etc.) and either feed the same image, or two randomly selected images, into the neural network, asking it to say whether it is the same image or not
  - Noisy labels/Self Training: Use labels generated by a weak algorithm (potentially older generation of the target model) to train a target-robust feature extractor

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In Advances in neural information processing systems (pp. 3079-3087).
Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709.
Xie, Q., Hovy, E., Luong, M. T., & Le, Q. V. (2019). Self-training with Noisy Student improves ImageNet classification. arXiv preprint arXiv:1911.04252.

THE COST OF LABELING

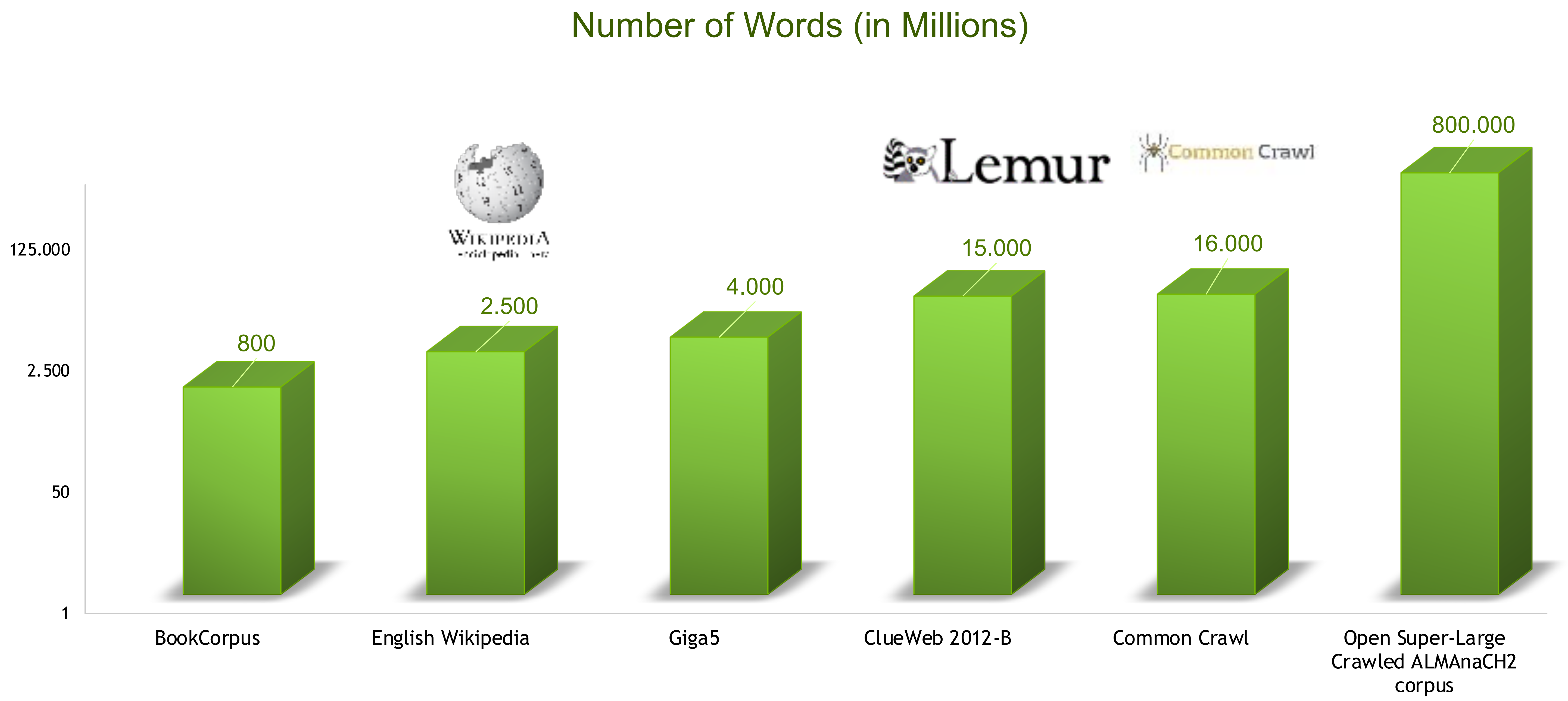
Semi-supervised models



Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., ... & Zhou, Y. (2017). Deep Learning Scaling is Predictable, Empirically. arXiv preprint arXiv:1712.00409

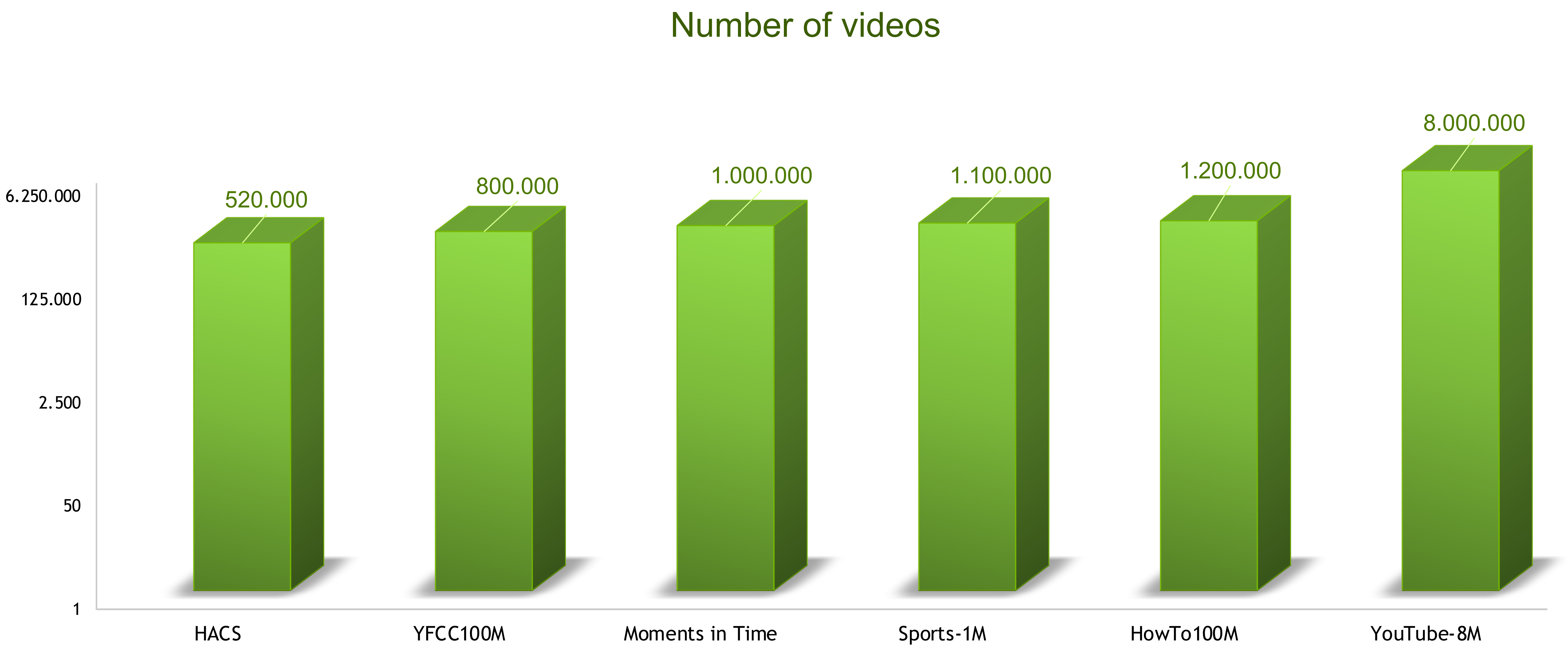
SELF-SUPERVISED LEARNING

Abundance of unlabeled data



SELF-SUPERVISED LEARNING

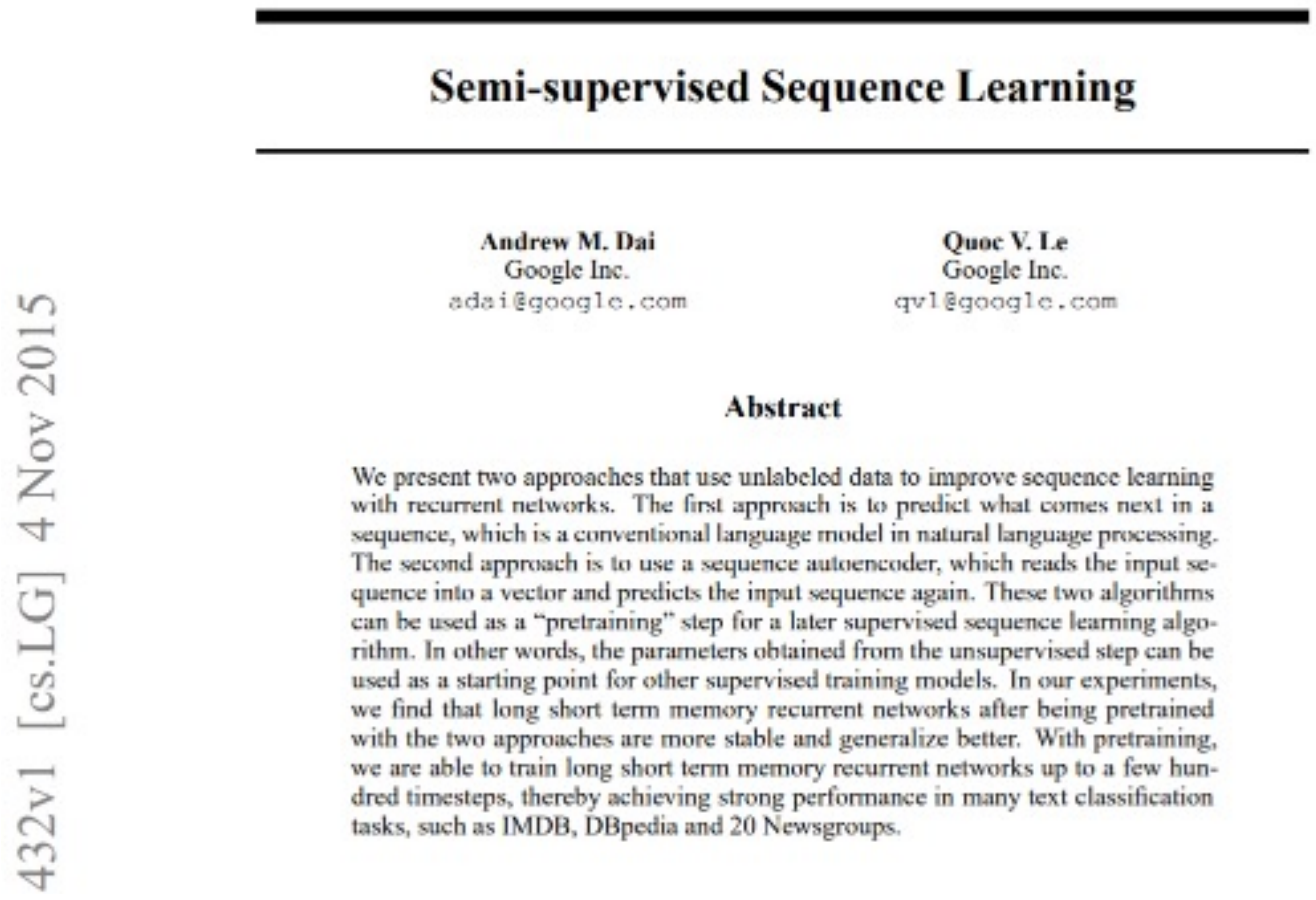
Abundance of unlabeled data



OLD IDEAS

SELF-SUPERVISED LEARNING

What was missing?



432v1 [cs.LG] 4 Nov 2015

Semi-supervised Sequence Learning

Andrew M. Dai
Google Inc.
adai@google.com

Quoc V. Le
Google Inc.
qvl@google.com

Abstract

We present two approaches that use unlabeled data to improve sequence learning with recurrent networks. The first approach is to predict what comes next in a sequence, which is a conventional language model in natural language processing. The second approach is to use a sequence autoencoder, which reads the input sequence into a vector and predicts the input sequence again. These two algorithms can be used as a "pretraining" step for a later supervised sequence learning algorithm. In other words, the parameters obtained from the unsupervised step can be used as a starting point for other supervised training models. In our experiments, we find that long short term memory recurrent networks after being pretrained with the two approaches are more stable and generalize better. With pretraining, we are able to train long short term memory recurrent networks up to a few hundred timesteps, thereby achieving strong performance in many text classification tasks, such as IMDB, DBpedia and 20 Newsgroups.

THE SCALE

GENERATIVE PRETRAINING (GPT)

The scale

“Many previous approaches to NLP tasks train relatively small models on a single GPU from scratch. Our approach requires an expensive pre-training step - 1 month on 8 GPUs. Luckily, this only has to be done once and we’re releasing our model so others can avoid it. It is also a large model (in comparison to prior work) and consequently uses more compute and memory — we used a 37-layer (12 block) Transformer architecture, and we train on sequences of up to 512 tokens. Most experiments were conducted on 4 and 8 GPU systems. The model does fine-tune to new tasks very quickly which helps mitigate the additional resource requirements.”

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf.*

GENERATIVE PRETRAINING (GPT)

The design

Transformer Decoder

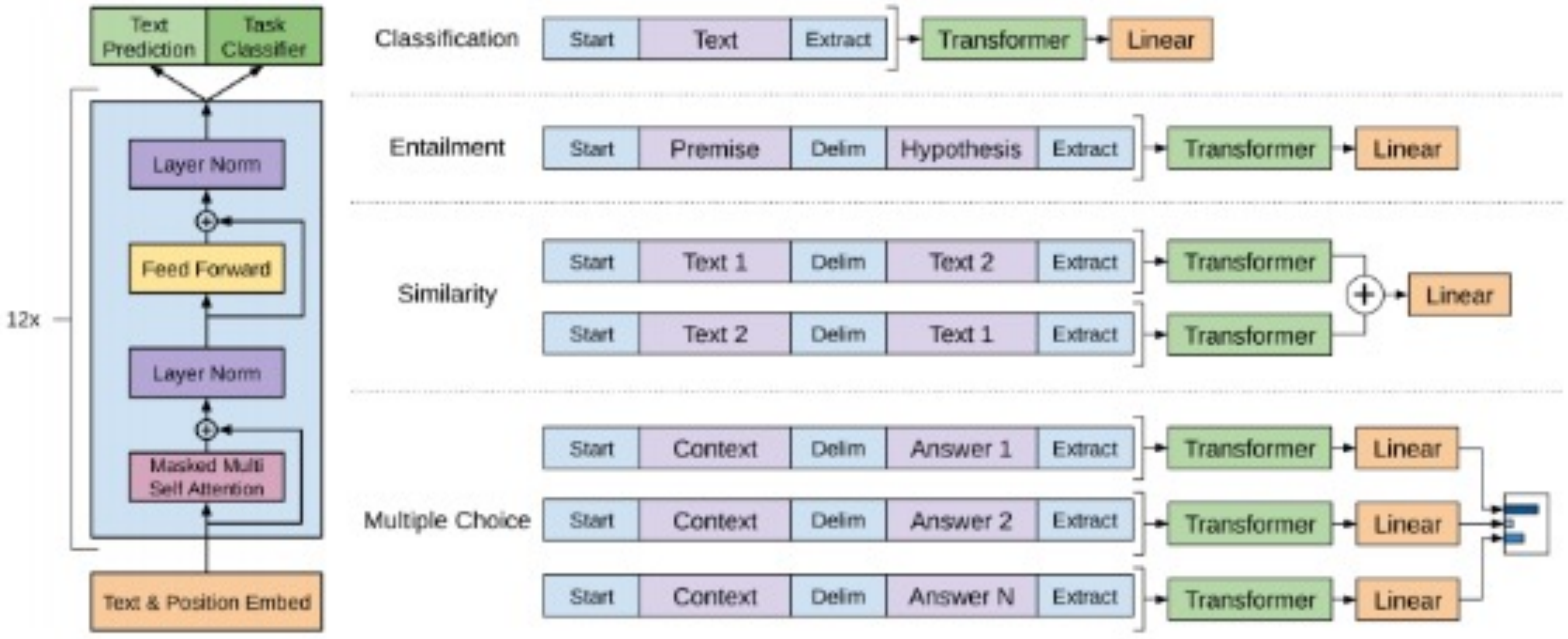


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Self-Supervised Training

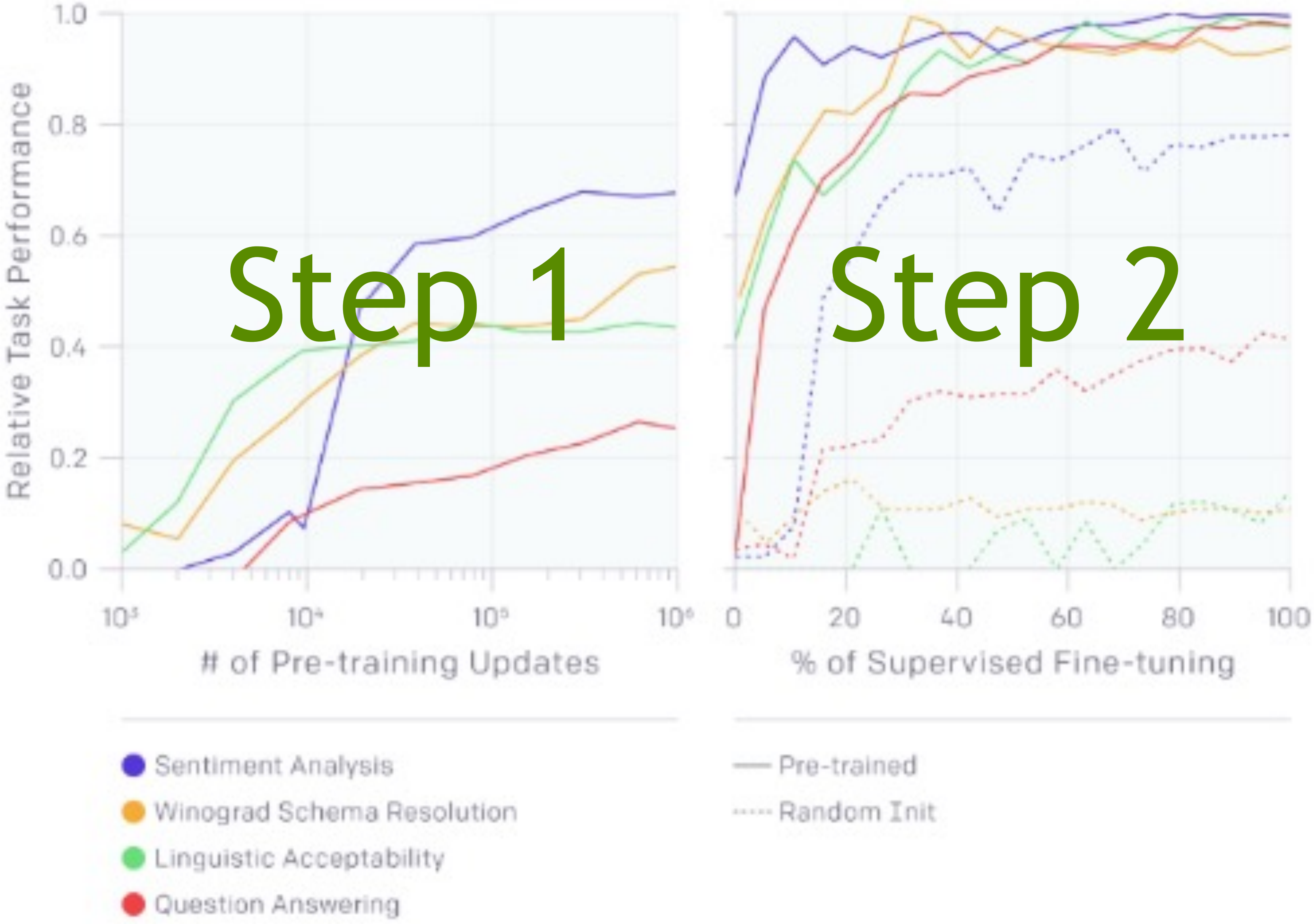
Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf.*

IT BECAME POSSIBLE TO TRANSFER LEARN!

GENERATIVE PRETRAINING (GPT)

The approach

Zero-shot Transfer Can Directly Accelerate Supervised Fine-tuning

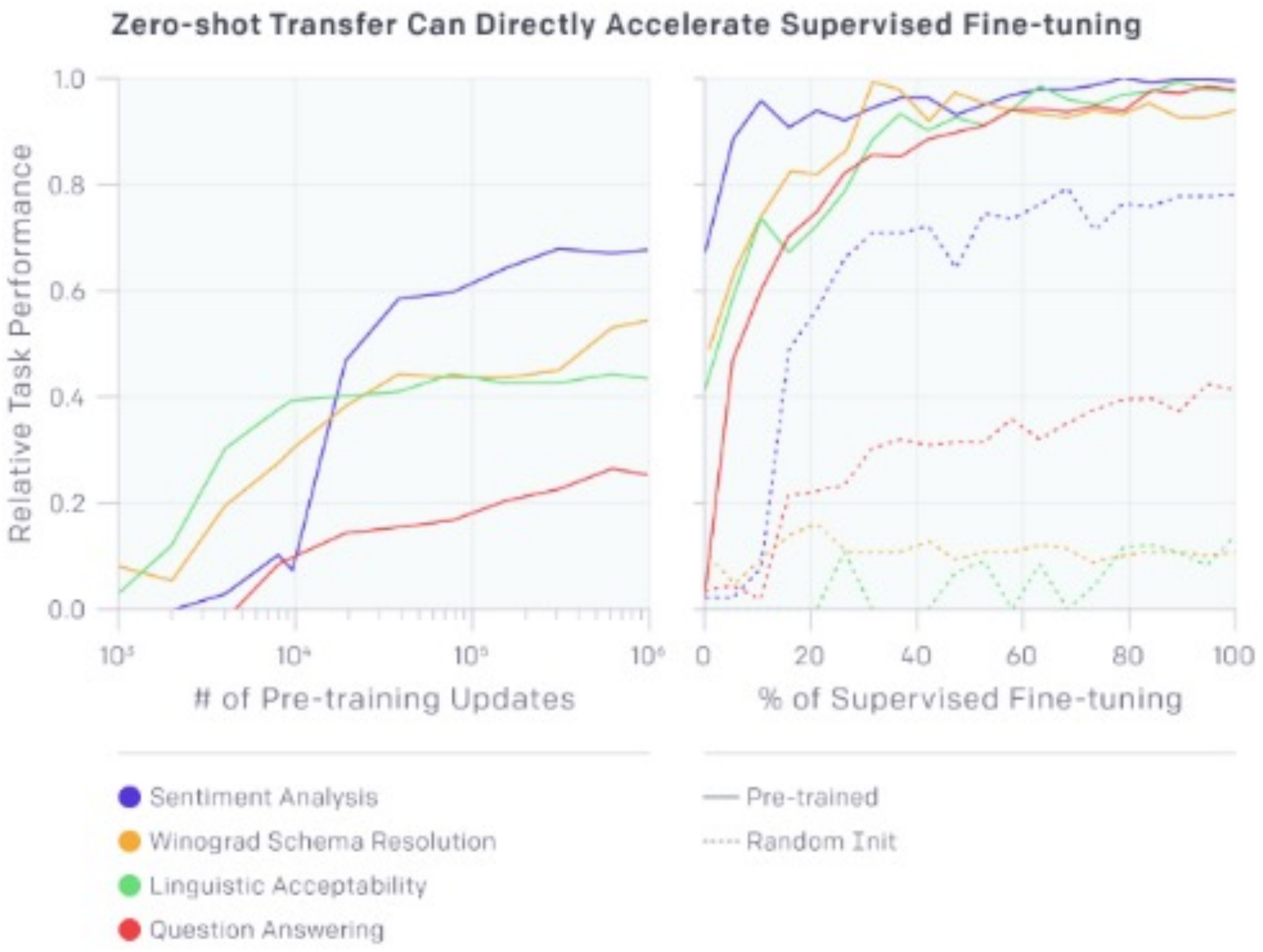


Pre-training our model on a large corpus of text significantly improves its performance on challenging natural language processing tasks like Winograd Schema Resolution.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf.*

GENERATIVE PRETRAINING (GPT)

The implications



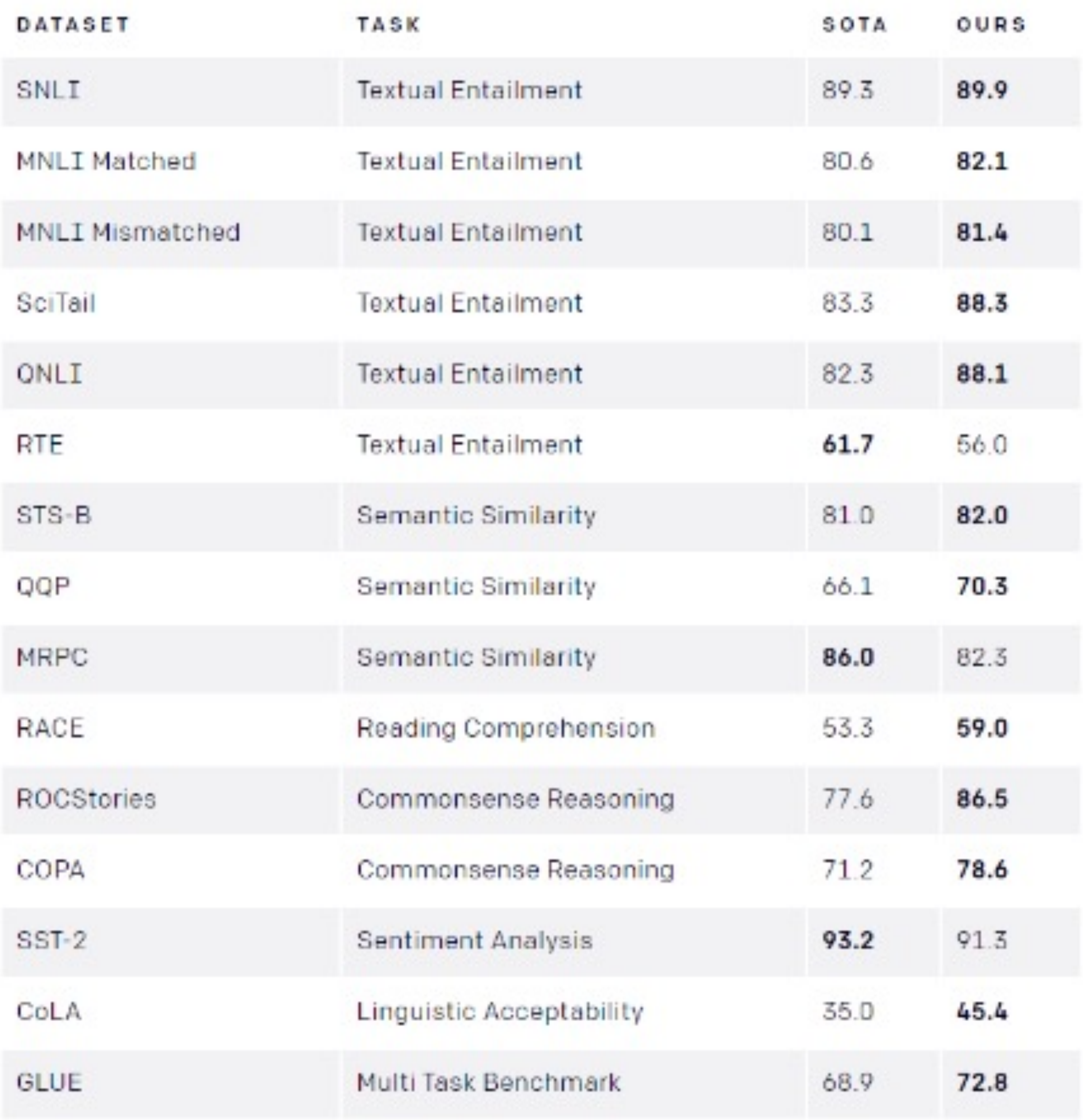
Pre-training our model on a large corpus of text significantly improves its performance on challenging natural language processing tasks like Winograd Schema Resolution.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf.*

AND IT WORKED VERY WELL

GENERATIVE PRETRAINING (GPT)

The implications



DATASET	TASK	SOTA	OURS
SNLI	Textual Entailment	89.3	**89.9**
MNLI Matched	Textual Entailment	80.6	**82.1**
MNLI Mismatched	Textual Entailment	80.1	**81.4**
SciTail	Textual Entailment	83.3	**88.3**
QNLI	Textual Entailment	82.3	**88.1**
RTE	Textual Entailment	**61.7**	56.0
STS-B	Semantic Similarity	81.0	**82.0**
QQP	Semantic Similarity	66.1	**70.3**
MRPC	Semantic Similarity	**86.0**	82.3
RACE	Reading Comprehension	53.3	**59.0**
ROCStories	Commonsense Reasoning	77.6	**86.5**
COPA	Commonsense Reasoning	71.2	**78.6**
SST-2	Sentiment Analysis	**93.2**	91.3
CoLA	Linguistic Acceptability	35.0	**45.4**
GLUE	Multi Task Benchmark	68.9	**72.8**

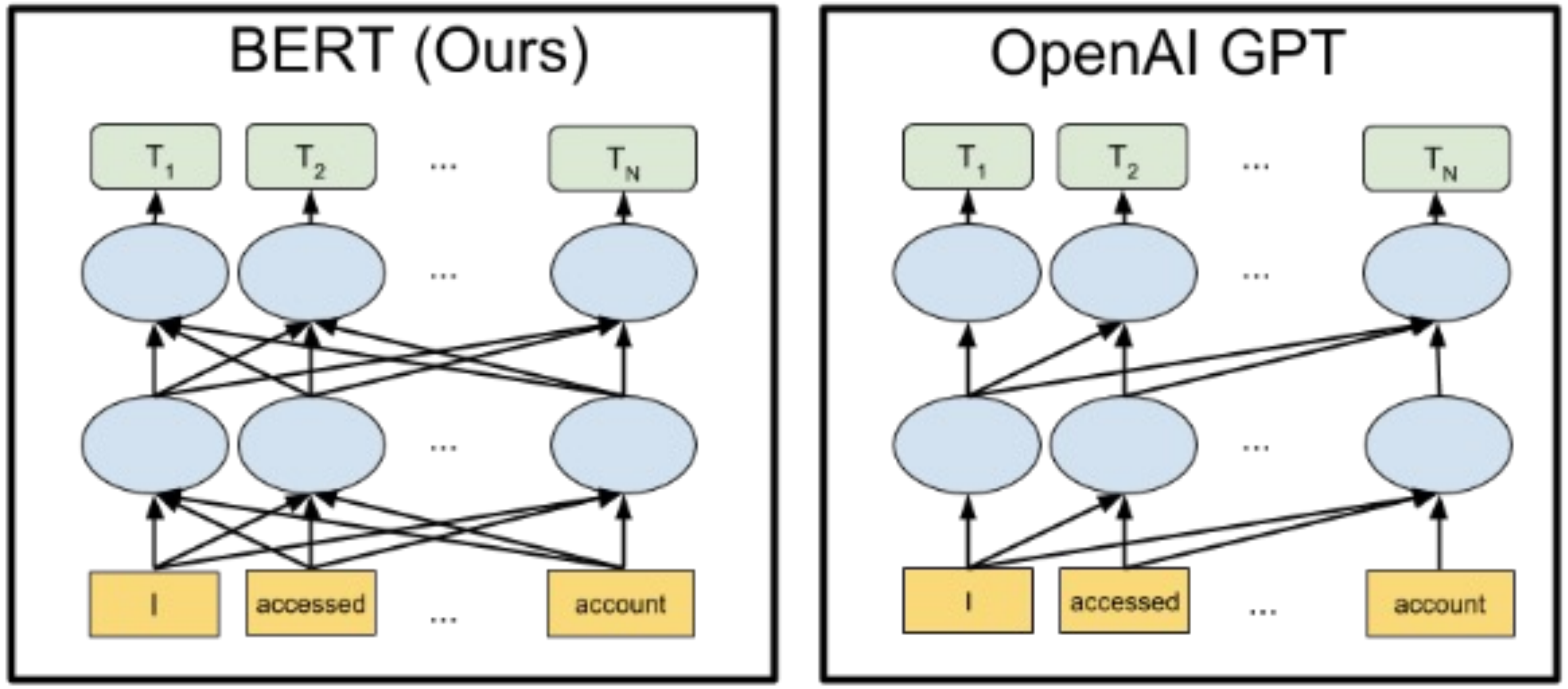
Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf.*

Part 2: Self-Supervision, BERT and Beyond

- Lecture
  - Why Do DNNs Work Well?
  - Self-Supervised Learning
  - BERT
- Lab
  - Explore the Data
  - Explore NeMo
  - Text Classifier Project
- Lecture (cont’d)
  - The Scaling Laws
  - Can and should we go even bigger?
- Lab (cont’d)
  - Named Entity Recognizer

BIDIRECTIONAL TRANSFORMERS (BERT)

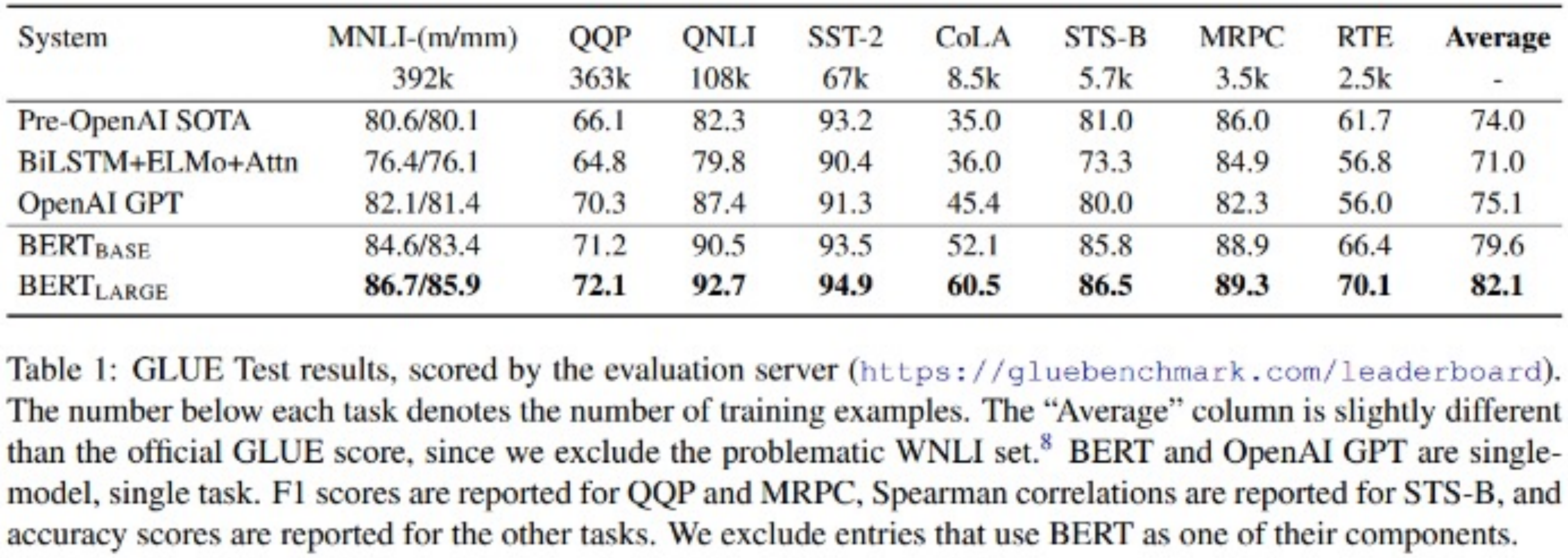
Building on the shoulders of giants



Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

BIDIRECTIONAL TRANSFORMERS (BERT)

The “pre” and “post” OpenAI ages



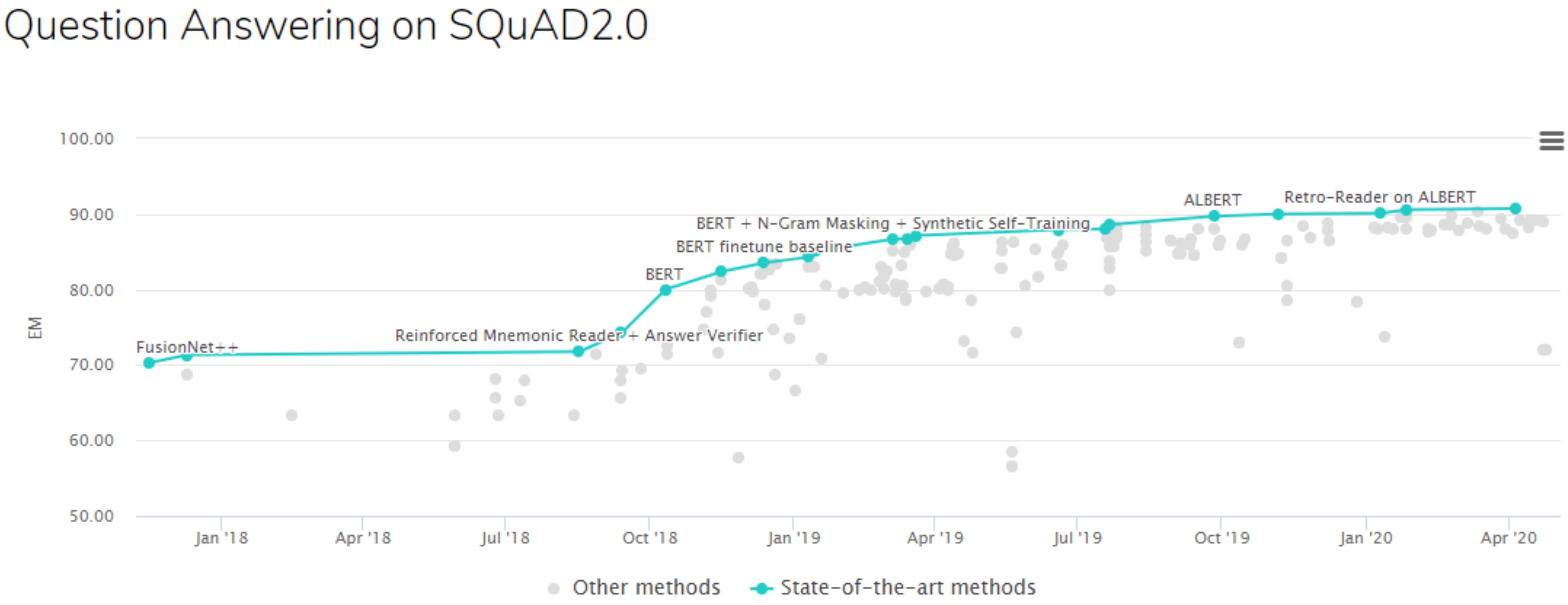
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (`https://gluebenchmark.com/leaderboard`). The number below each task denotes the number of training examples. The “Average” column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

SQUAD 2.0

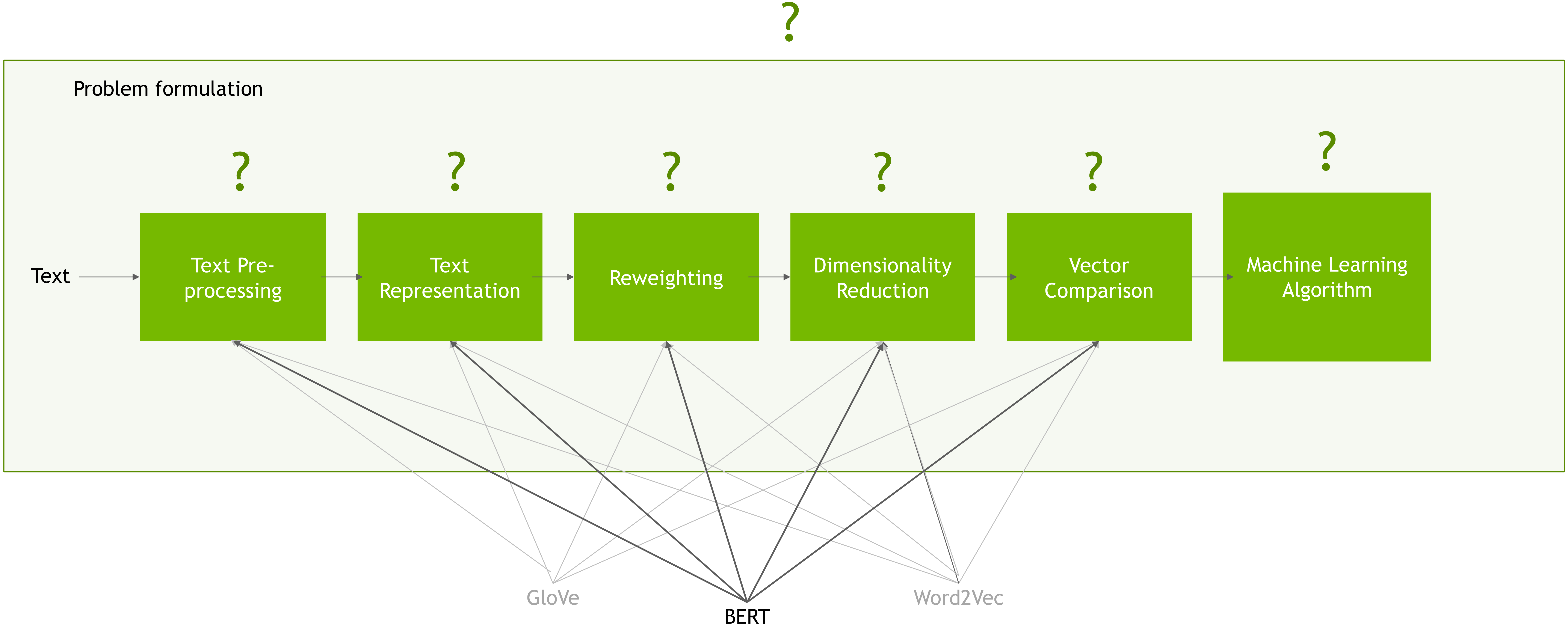
Human performance 91.2



JUST YET ANOTHER UNSUPERVISED REPRESENTATION

USING BERT

Feature extractor

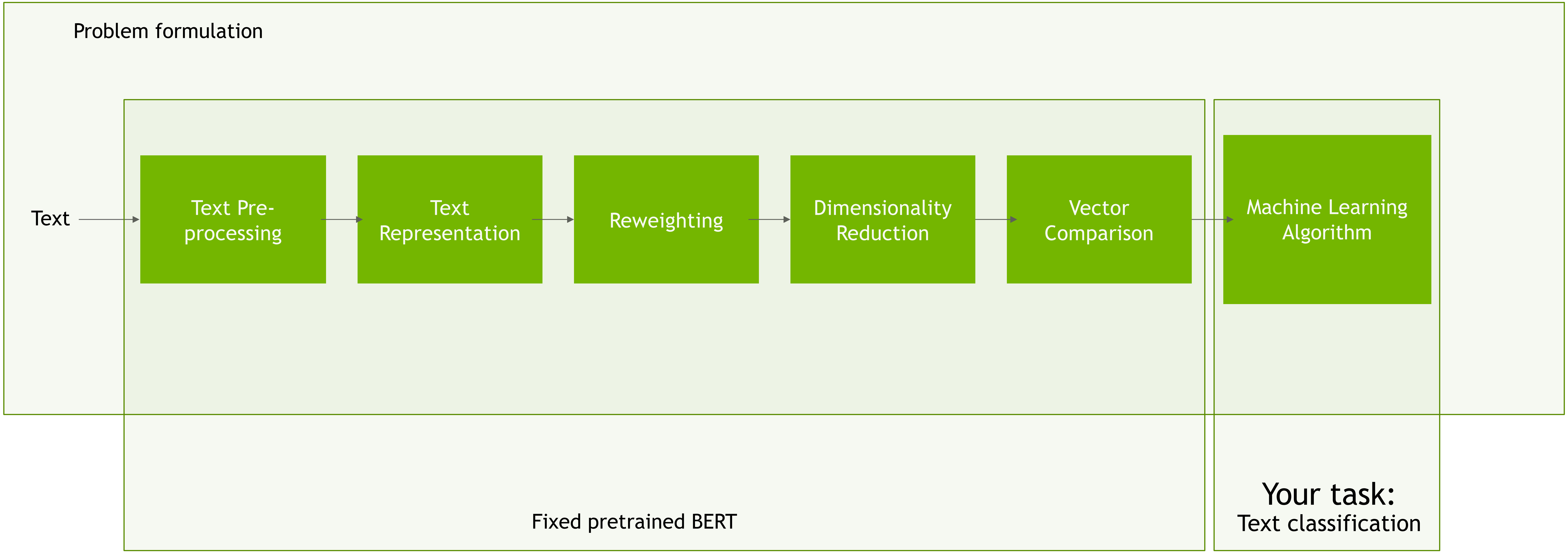


THE LAB

LAB OVERVIEW

Notebooks 1, 2, 3

Text classification



Problem formulation

Text

Text Pre-processing

Text Representation

Reweighting

Dimensionality Reduction

Vector Comparison

Machine Learning Algorithm

Fixed pretrained BERT

Your task:
Text classification

Part 2: Self-Supervision, BERT and Beyond

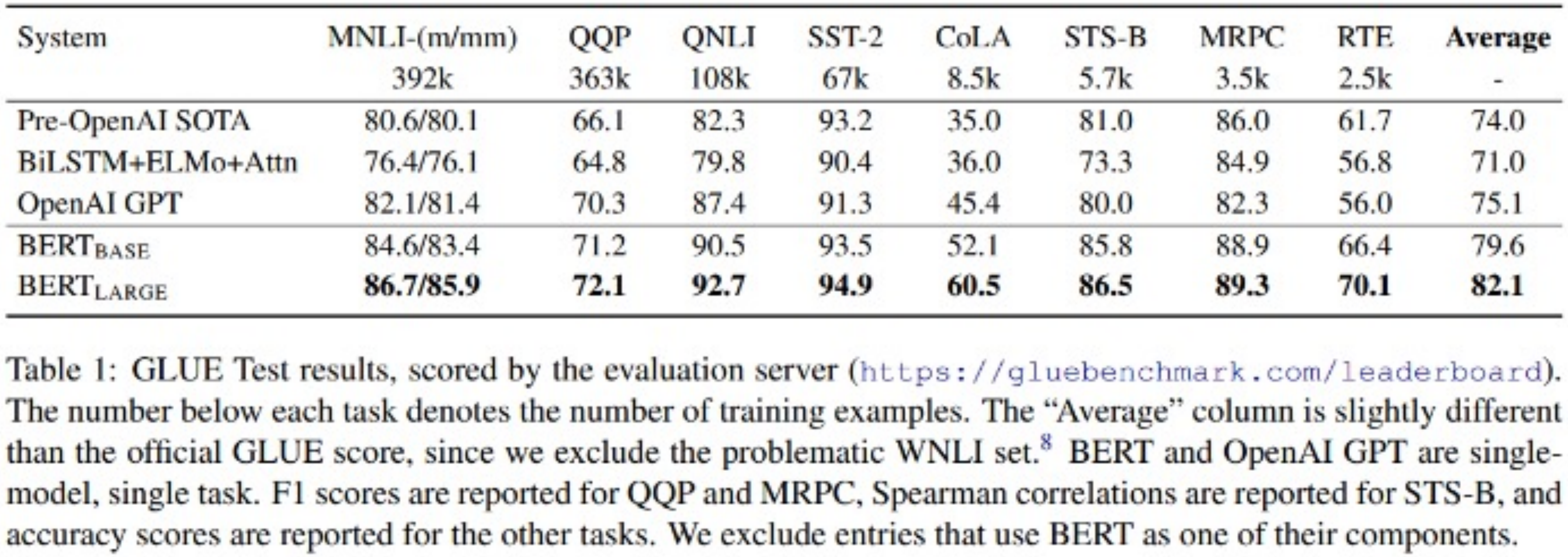
- Lecture
  - Why Do DNNs Work Well?
  - Self-Supervised Learning
  - BERT
- Lab
  - Explore the Data
  - Explore NeMo
  - Text Classifier Project
- Lecture (cont’d)
  - The Scaling Laws
  - Can and should we go even bigger?
- Lab (cont’d)
  - Named Entity Recognizer

Part 2: Self-Supervision, BERT and Beyond

- Lecture
  - Why Do DNNs Work Well?
  - Self-Supervised Learning
  - BERT
- Lab
  - Explore the Data
  - Explore NeMo
  - Text Classifier Project
- Lecture (cont’d)
  - The Scaling Laws
  - Can and should we go even bigger?
- Lab (cont’d)
  - Named Entity Recognizer

BIDIRECTIONAL TRANSFORMERS (BERT)

Base vs Large



System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

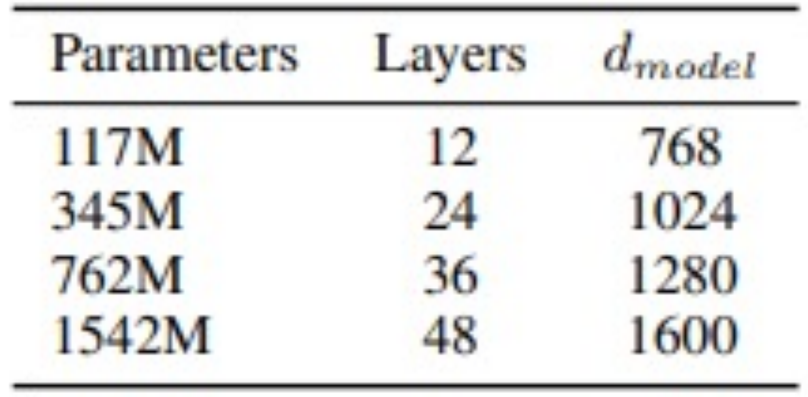
Table 1: GLUE Test results, scored by the evaluation server (`https://gluebenchmark.com/leaderboard`). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

GPT-2

GPT vs GPT-2

- Largely the same but:
  - Larger in every way:
    - More decoder layers: 12->48
    - Larger vocabulary: 50,257
    - Larger context: 512 -> 1024
    - Larger batch size
  - Changes to layer normalization
  - Different initialization scheme



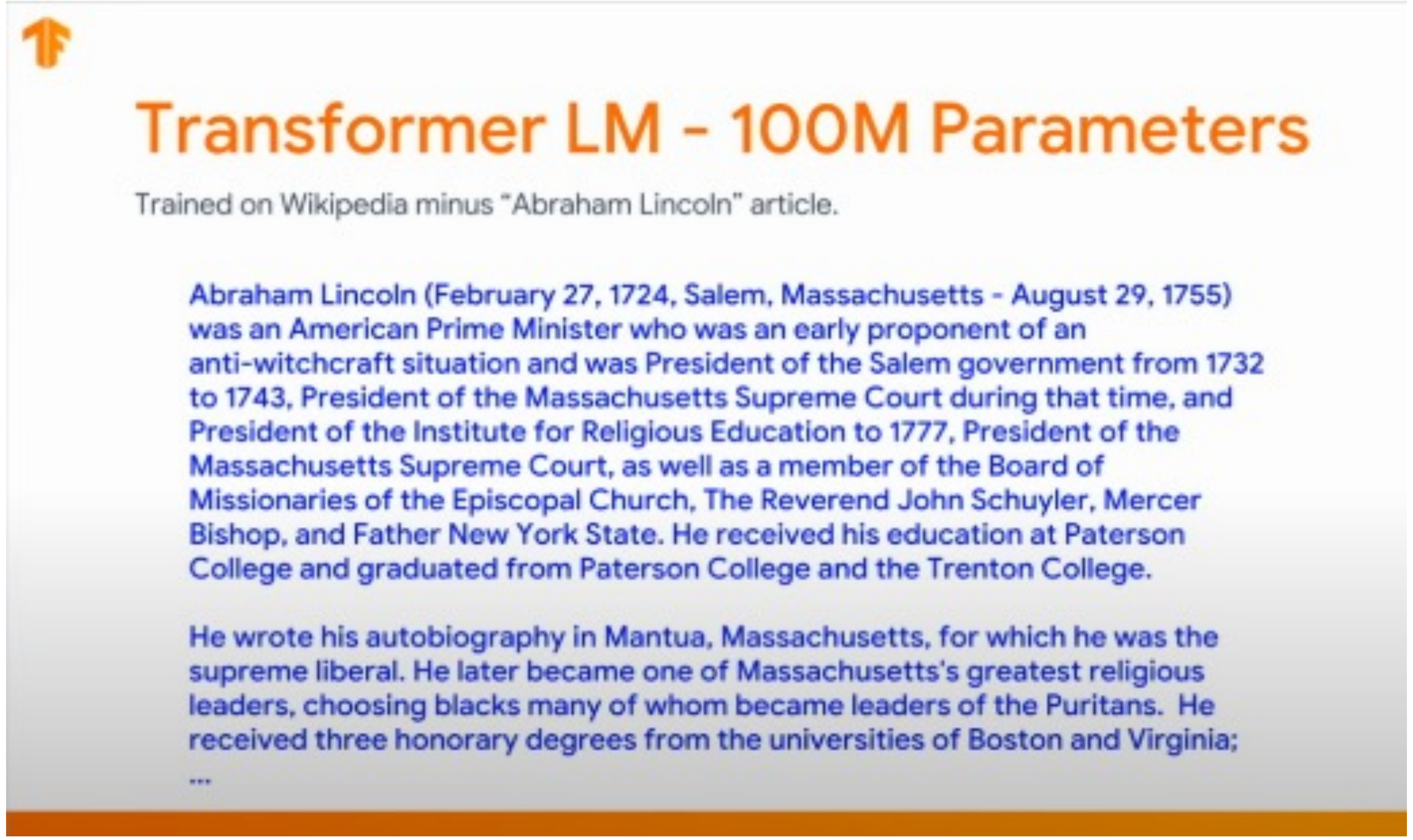
Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Table 2. Architecture hyperparameters for the 4 model sizes.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 1(8), 9.

GPT-2

The Impact



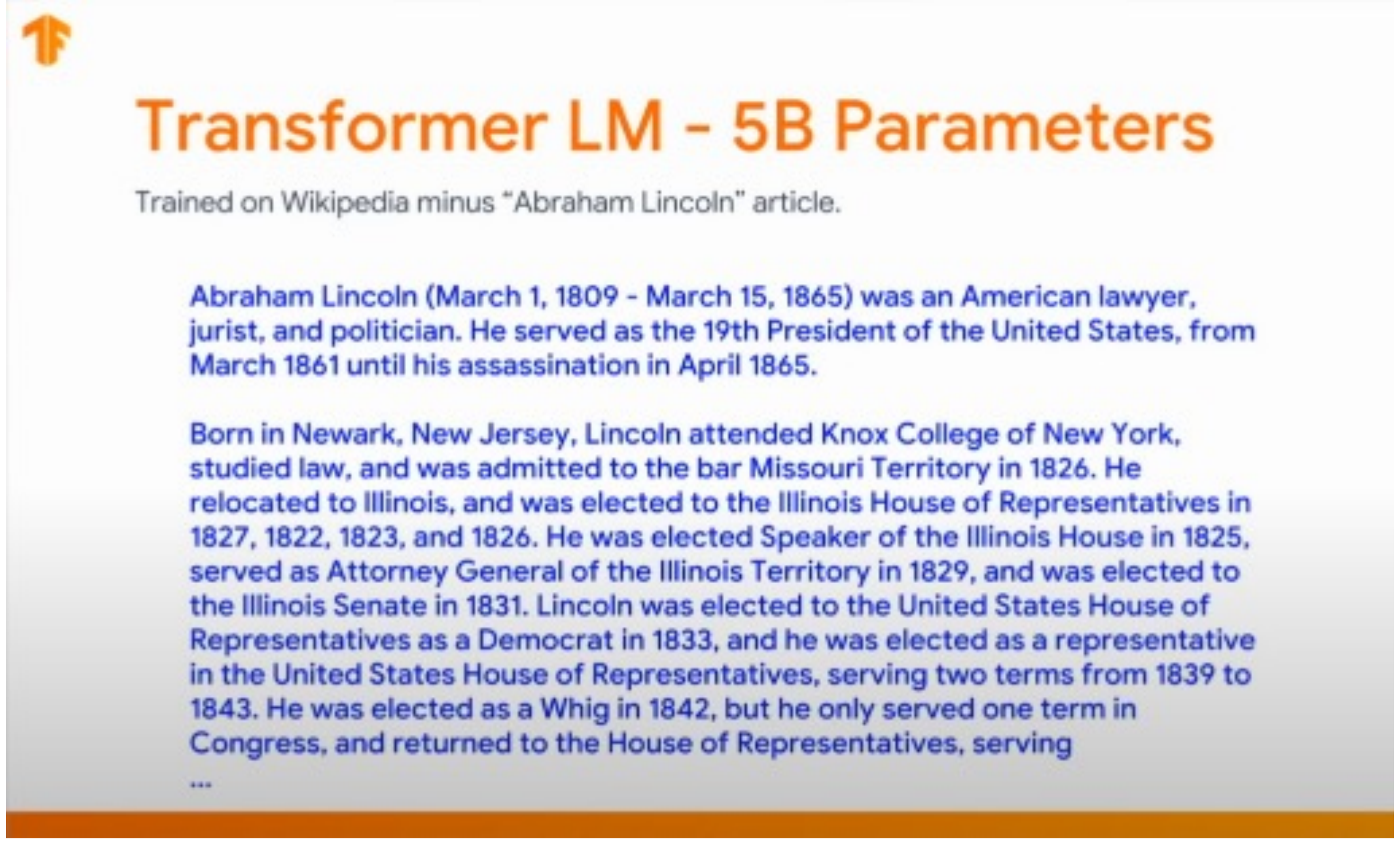
Transformer LM - 100M Parameters

Trained on Wikipedia minus "Abraham Lincoln" article.

Abraham Lincoln (February 27, 1724, Salem, Massachusetts - August 29, 1755) was an American Prime Minister who was an early proponent of an anti-witchcraft situation and was President of the Salem government from 1732 to 1743, President of the Massachusetts Supreme Court during that time, and President of the Institute for Religious Education to 1777, President of the Massachusetts Supreme Court, as well as a member of the Board of Missionaries of the Episcopal Church, The Reverend John Schuyler, Mercer Bishop, and Father New York State. He received his education at Paterson College and graduated from Paterson College and the Trenton College.

He wrote his autobiography in Mantua, Massachusetts, for which he was the supreme liberal. He later became one of Massachusetts's greatest religious leaders, choosing blacks many of whom became leaders of the Puritans. He received three honorary degrees from the universities of Boston and Virginia;

...



Transformer LM - 5B Parameters

Trained on Wikipedia minus "Abraham Lincoln" article.

Abraham Lincoln (March 1, 1809 - March 15, 1865) was an American lawyer, jurist, and politician. He served as the 19th President of the United States, from March 1861 until his assassination in April 1865.

Born in Newark, New Jersey, Lincoln attended Knox College of New York, studied law, and was admitted to the bar Missouri Territory in 1826. He relocated to Illinois, and was elected to the Illinois House of Representatives in 1827, 1822, 1823, and 1826. He was elected Speaker of the Illinois House in 1825, served as Attorney General of the Illinois Territory in 1829, and was elected to the Illinois Senate in 1831. Lincoln was elected to the United States House of Representatives as a Democrat in 1833, and he was elected as a representative in the United States House of Representatives, serving two terms from 1839 to 1843. He was elected as a Whig in 1842, but he only served one term in Congress, and returned to the House of Representatives, serving

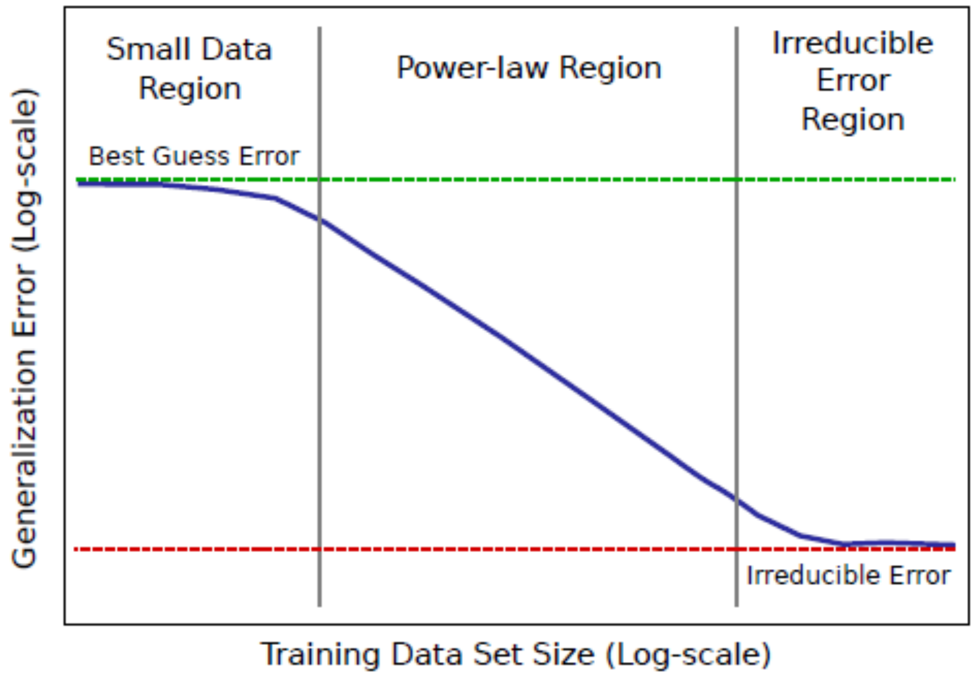
...

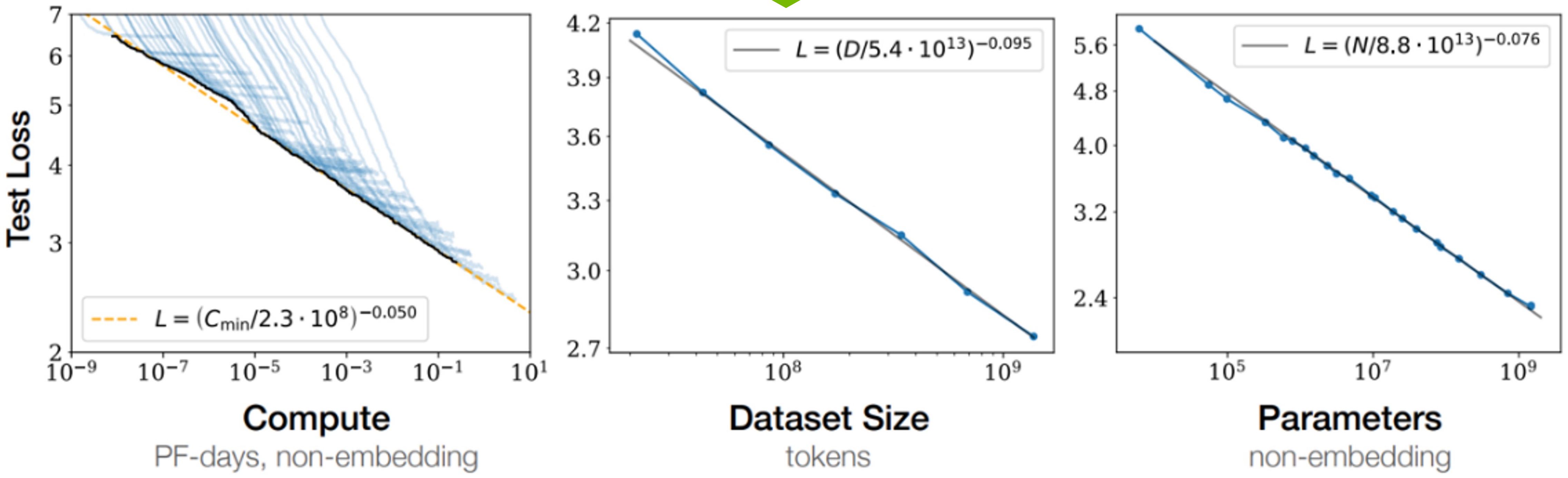
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 1(8), 9.

MEASURABLE IMPROVEMENT

THE SCALING LAWS OF NEURAL LANGUAGE MODELS

Continuous improvement





Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., ... & Zhou, Y. Deep Learning Scaling is Predictable, Empirically. arXiv preprint arXiv:1712.00409. 2017
Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, Sam McCandlish. Scaling Laws for Autoregressive Generative Modeling.2020

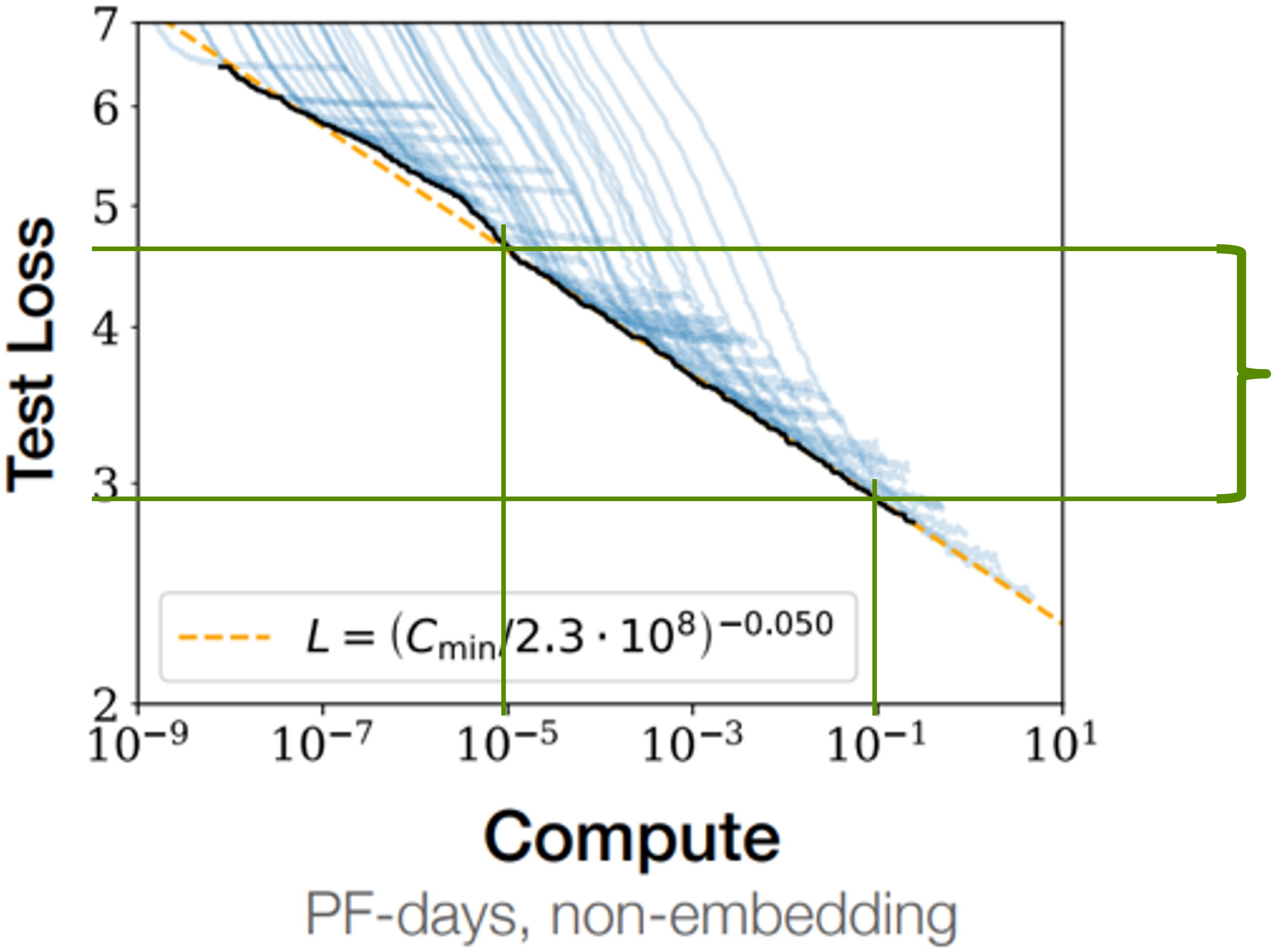
Part 2: Self-Supervision, BERT and Beyond

- Lecture
  - Why Do DNNs Work Well?
  - Self-Supervised Learning
  - BERT
- Lab
  - Explore the Data
  - Explore NeMo
  - Text Classifier Project
- Lecture (cont’d)
  - The Scaling Laws
  - Can and should we go even bigger?
- Lab (cont’d)
  - Named Entity Recognizer

SHOULD WE BUILD LARGER MODELS?

ARE LARGE LANGUAGE MODELS WORTH IT?

The cost of incremental improvement



Are we building those models only for the small incremental improvement in their performance?

Is it worth all of the engineering and computational investment?

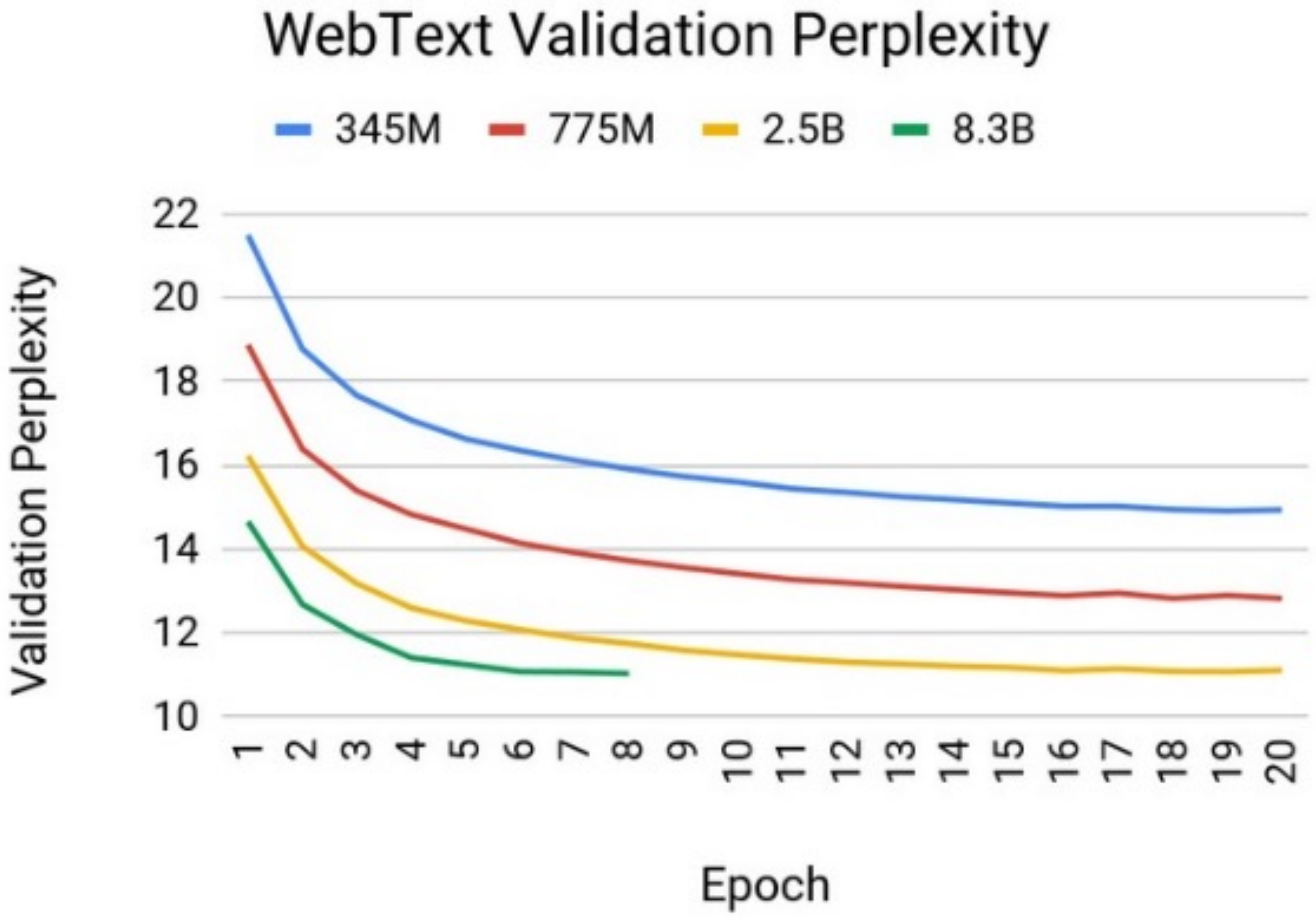
IS THIS REALLY THE ONLY THING WE HAVE ACHIEVED?

IT IS MUCH MORE THAN JUST INCREMENTAL INCREASE IN ACCURACY!

1. SAMPLE EFFICIENCY

NOT ABOUT INCREMENTAL IMPROVEMENT

Sample efficiency



Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. arXiv preprint arXiv:1909.08053.

LARGER MODELS ARE CHEAPER TO TRAIN

Optimal allocation of computational budget

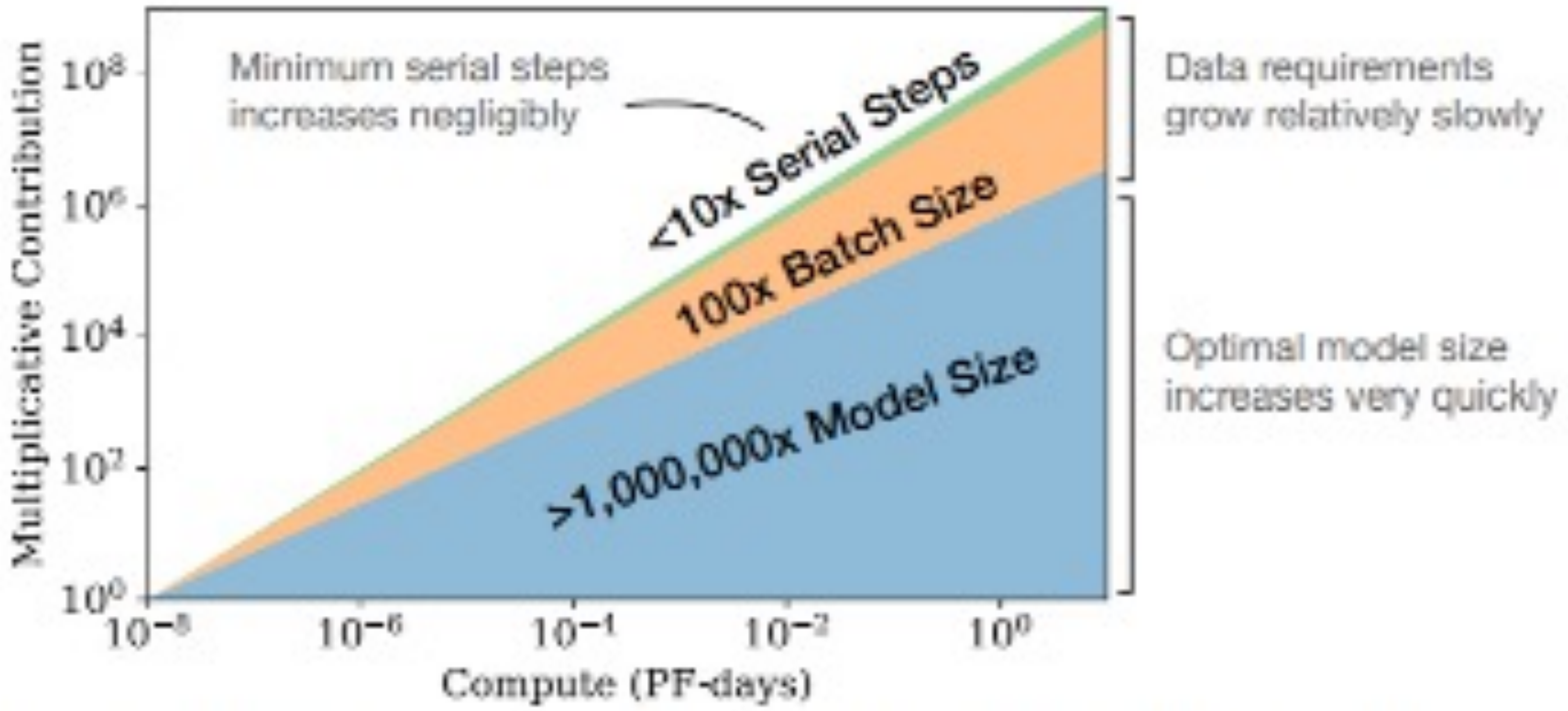


Figure 3 As more compute becomes available, we can choose how much to allocate towards training larger models, using larger batches, and training for more steps. We illustrate this for a billion-fold increase in compute. For optimally compute-efficient training, most of the increase should go towards increased model size. A relatively small increase in data is needed to avoid reuse. Of the increase in data, most can be used to increase parallelism through larger batch sizes, with only a very small increase in serial training time required.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*.

LARGER MODELS ARE CHEAPER TO TRAIN

For every dataset there exists an optimal model size minimizing compute

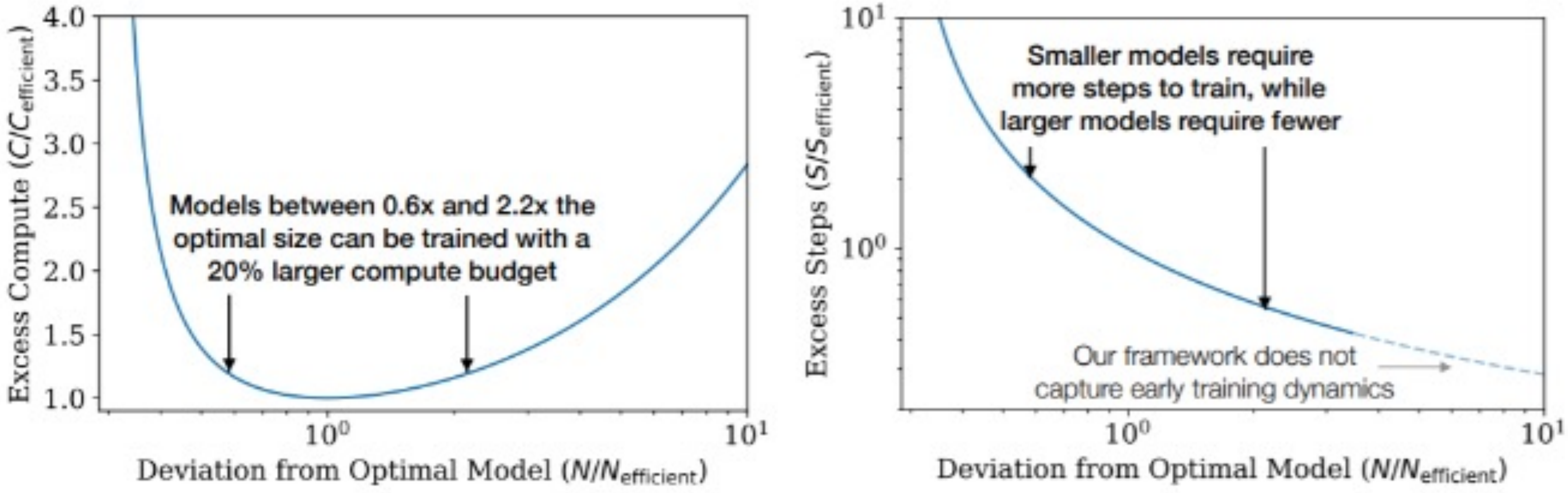


Figure 12 **Left:** Given a fixed compute budget, a particular model size is optimal, though somewhat larger or smaller models can be trained with minimal additional compute. **Right:** Models larger than the compute-efficient size require fewer steps to train, allowing for potentially faster training if sufficient additional parallelism is possible. Note that this equation should not be trusted for very large models, as it is only valid in the power-law region of the learning curve, after initial transient effects.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*.

2. ARCHITECTURAL HYPERPARAMETERS

LARGE MODELS ARE CHEAPER TO DESIGN

Impact of architectural hyperparameters

> *"… more importantly, we find that the precise architectural hyperparameters are unimportant compared to the overall scale of the language model."*

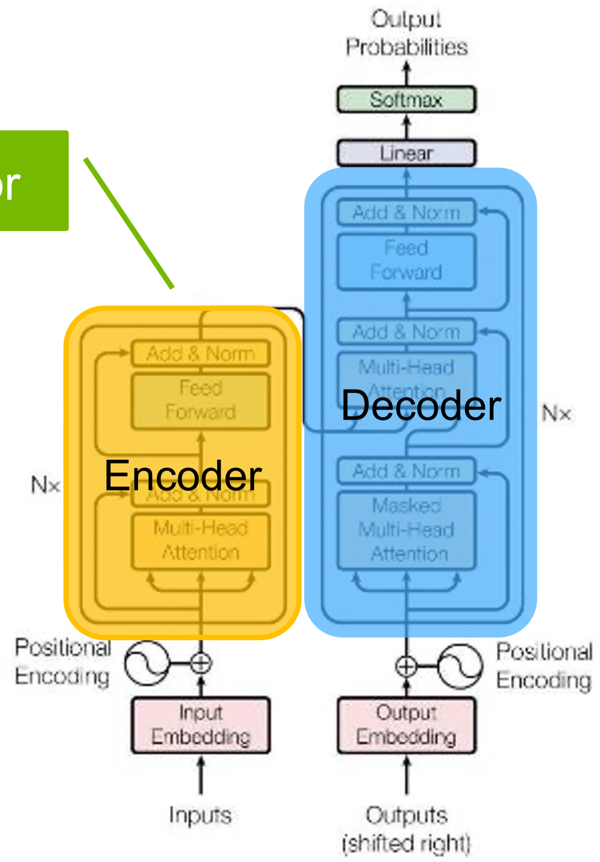
Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*.

NLP APPROACH (CIRCA 2019)

Step 1: Pre-training a Transformer

Output 1: Reconstruct missing words
family, of, this, the, Louis, personally, telephone

High dimensional vector



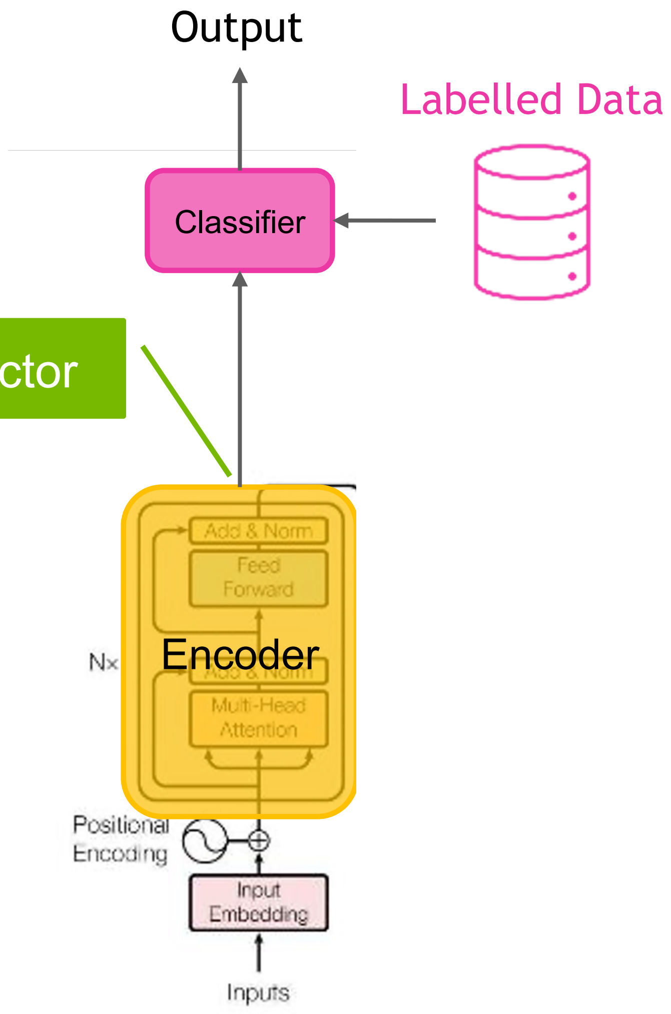
Input: Two sentences with 15% of words masked out

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]."

Step 2. Fine tune for a specific task

High dimensional vector



3. GENERALIZATION

YES THEY CREATE INCREMENTAL IMPROVEMENT IN ACCURACY

Larger models generalize better

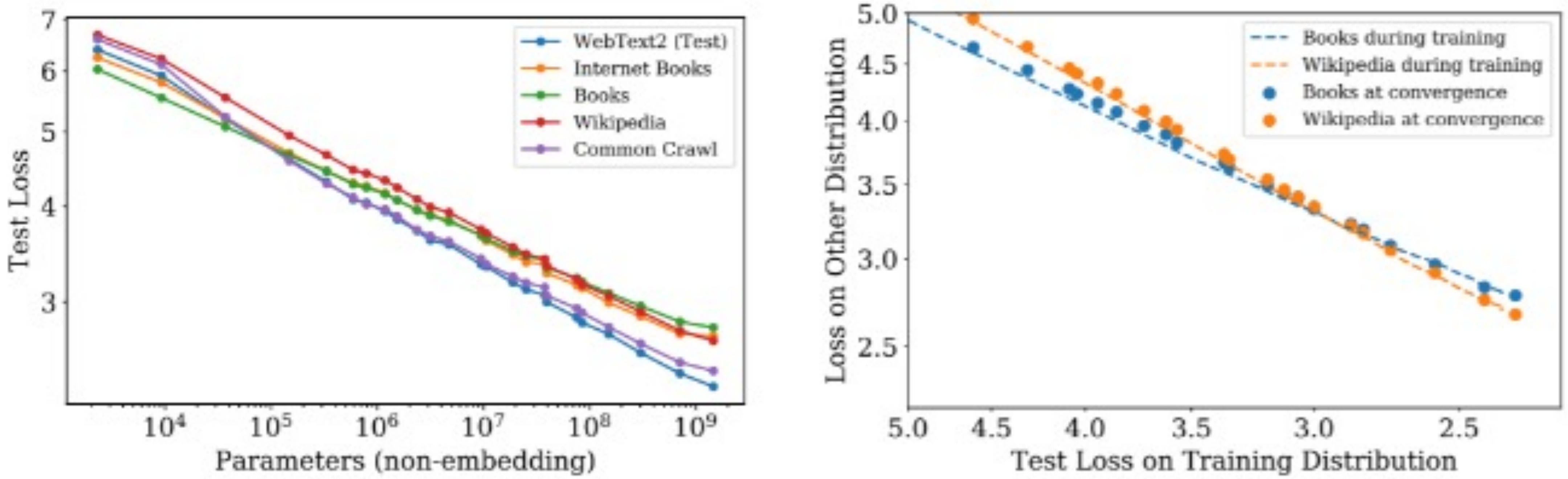
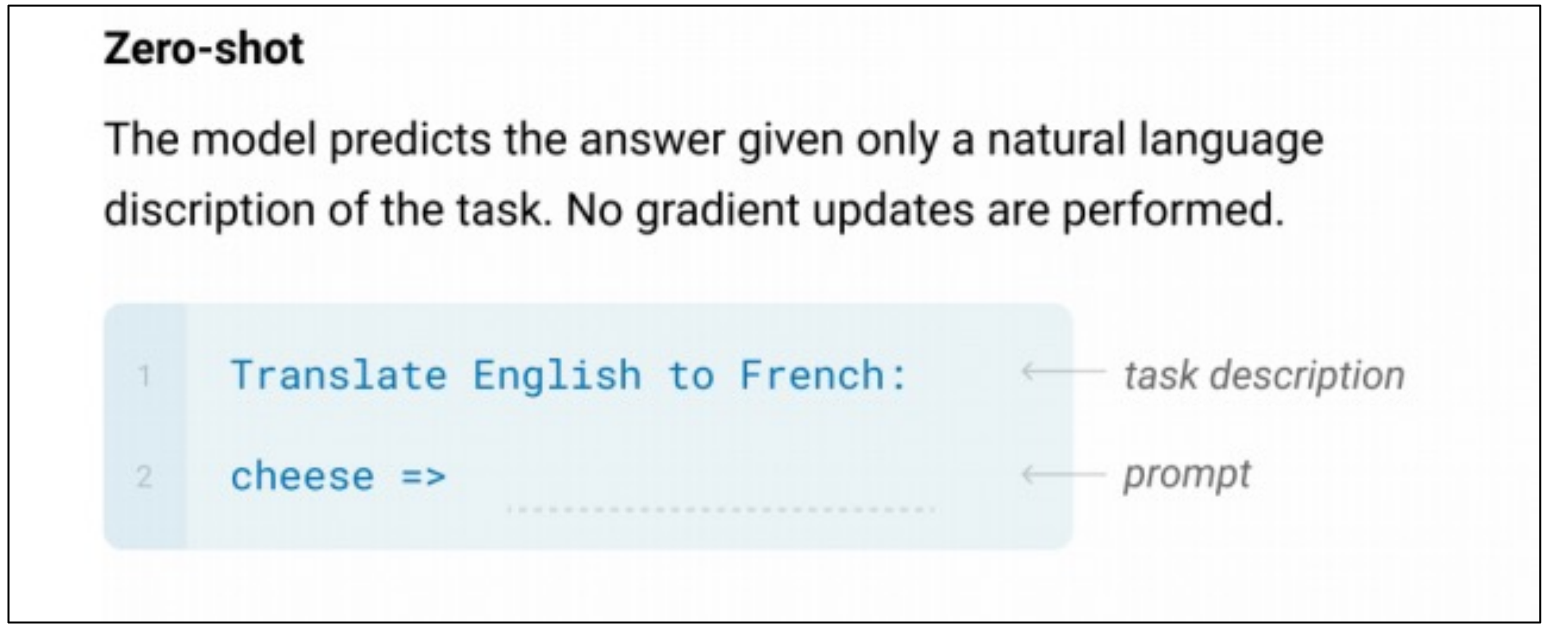


Figure 8 **Left:** Generalization performance to other data distributions improves smoothly with model size, with only a small and very slowly growing offset from the WebText2 training distribution. **Right:** Generalization performance depends only on training distribution performance, and not on the phase of training. We compare generalization of converged models (points) to that of a single large model (dashed curves) as it trains.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*.

DOWNSTREAM TASKS

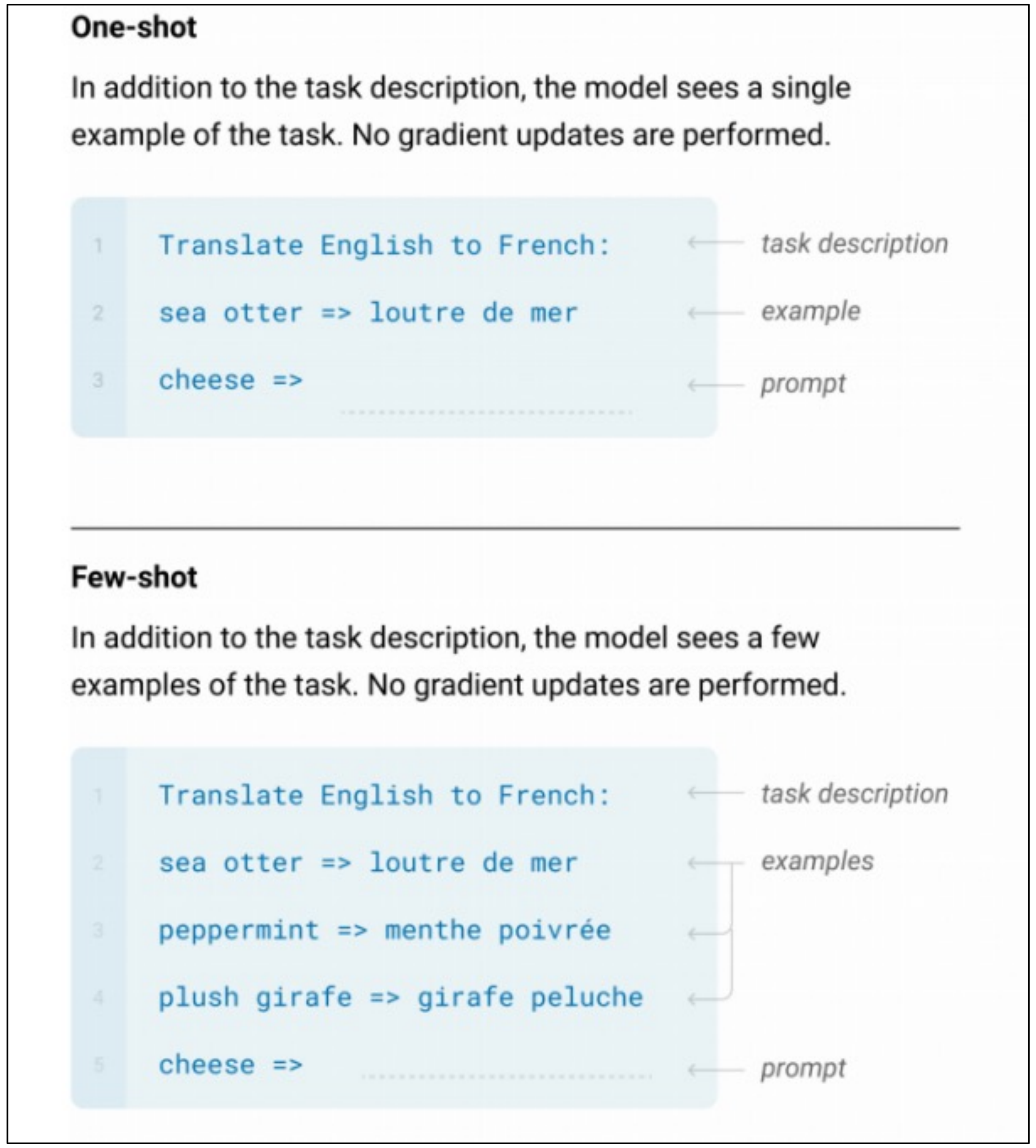
Zero/Few Shot Learners



Zero-shot

The model predicts the answer given only a natural language discription of the task. No gradient updates are performed.

```
Translate English to French:
cheese =>
```



One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:
sea otter => loutre de mer
cheese =>
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

Benjamin Mann et al. "Language Models are Few-Shot Learners". 2020. https://arxiv.org/abs/2005.14165

DOWNSTREAM TASKS

Zero/Few Shot Learners

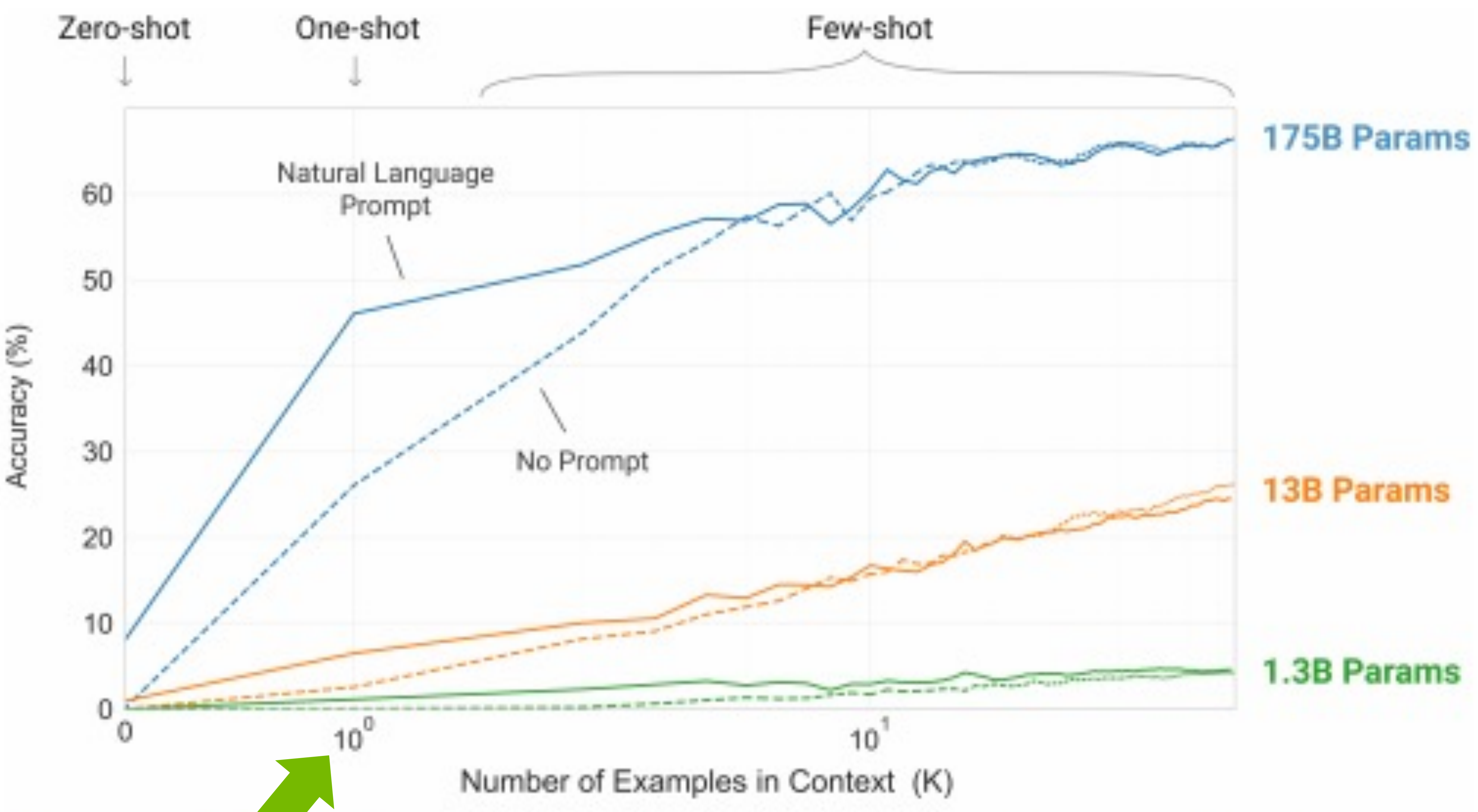


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. arXiv preprint arXiv:1909.08053

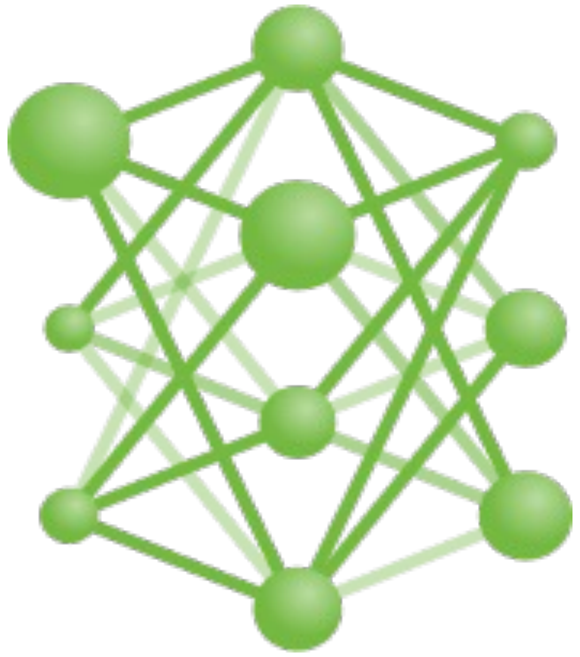
Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Agarwal, S. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*..

DOWNSTREAM TASKS

Zero/Few Shot Learners

Q: Would you say this movie review is positive or negative?
“I loved that movie”’

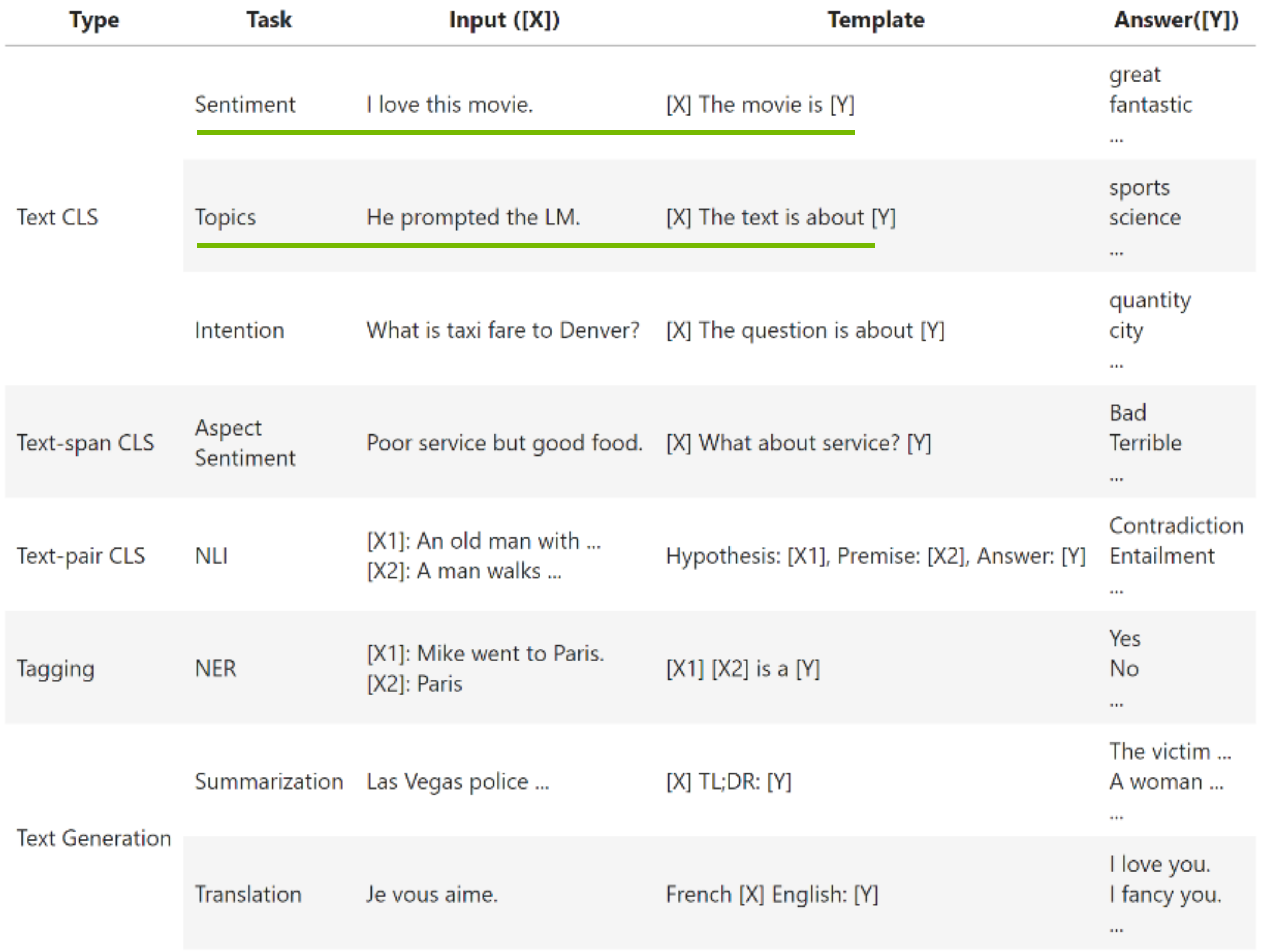
Prompt



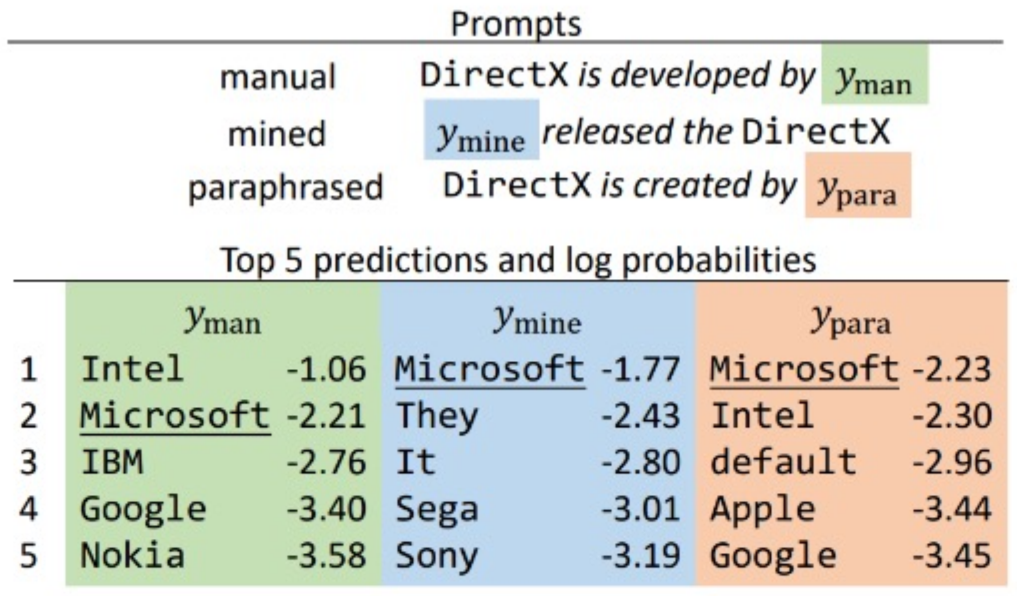
- positive
- great
- awsome …

DOWNSTREAM TASKS

Prompt Engineering



Type	Task	Input ([X])	Template	Answer([Y])
Text CLS	Sentiment	I love this movie.	[X] The movie is [Y]	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Y]	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Y]	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	[X] What about service? [Y]	Bad Terrible ...
Text-pair CLS	NLI	[X1]: An old man with ... [X2]: A man walks ...	Hypothesis: [X1], Premise: [X2], Answer: [Y]	Contradiction Entailment ...
Tagging	NER	[X1]: Mike went to Paris. [X2]: Paris	[X1] [X2] is a [Y]	Yes No ...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Y]	The victim ... A woman ... ...
	Translation	Je vous aime.	French [X] English: [Y]	I love you. I fancy you. ...

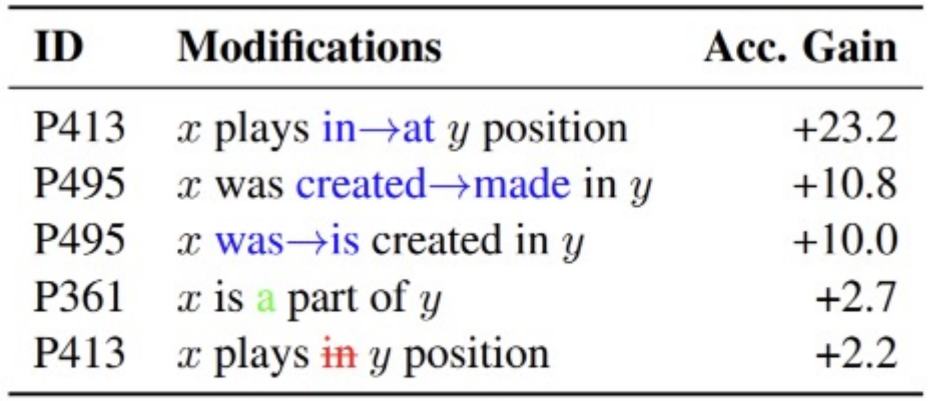


	Prompts
manual	`DirectX` *is developed by* y_{man}
mined	y_{mine} *released the* `DirectX`
paraphrased	`DirectX` *is created by* y_{para}

Top 5 predictions and log probabilities

	y_{man}		y_{mine}		y_{para}	
1	`Intel`	-1.06	<u>`Microsoft`</u>	-1.77	<u>`Microsoft`</u>	-2.23
2	<u>`Microsoft`</u>	-2.21	`They`	-2.43	`Intel`	-2.30
3	`IBM`	-2.76	`It`	-2.80	`default`	-2.96
4	`Google`	-3.40	`Sega`	-3.01	`Apple`	-3.44
5	`Nokia`	-3.58	`Sony`	-3.19	`Google`	-3.45

Figure 1: Top-5 predictions and their log probabilities using different prompts (manual, mined, and paraphrased) to query BERT. Correct answer is underlined.



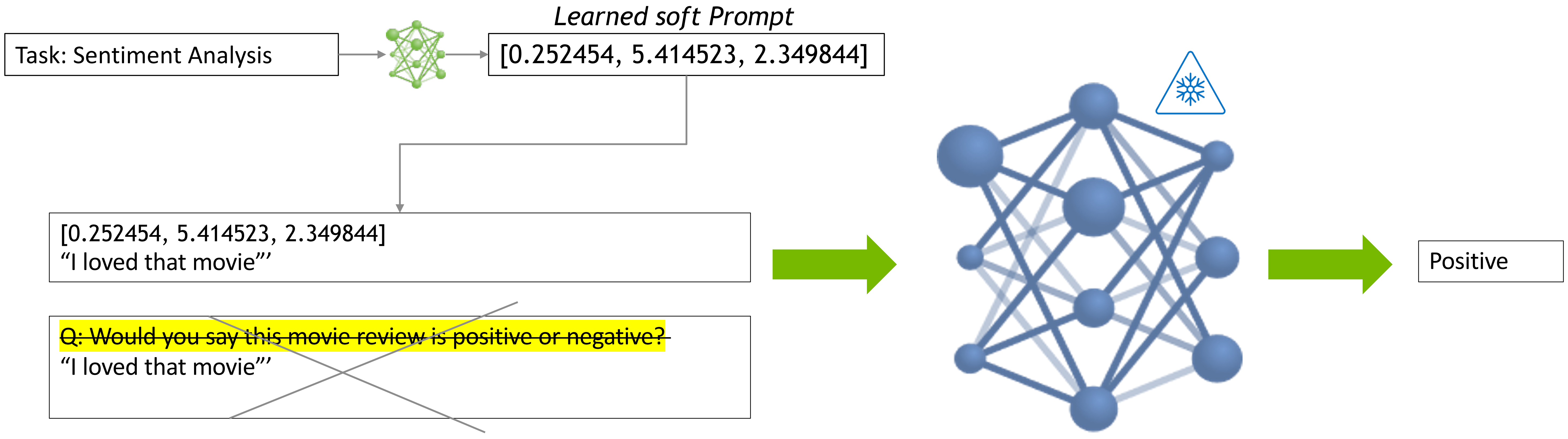
ID	Modifications	Acc. Gain
P413	x plays in→at y position	+23.2
P495	x was created→made in y	+10.8
P495	x was→is created in y	+10.0
P361	x is a part of y	+2.7
P413	x plays ~~in~~ y position	+2.2

Table 6: Small modifications (update, insert, and delete) in paraphrase lead to large accuracy gain (%).

Zhengbao Jiang et al. "How Can We Know What Language Models Know?".2020.

DOWNSTREAM TASKS

Prompt Learning on a Small Training Dataset



Learned soft Prompt

Task: Sentiment Analysis

[0.252454, 5.414523, 2.349844]

[0.252454, 5.414523, 2.349844]
“I loved that movie”’

~~Q: Would you say this movie review is positive or negative?~~
“I loved that movie”’

Positive

Prompt Learning

DOWNSTREAM TASKS

Prompt Tuning / P-Tuning

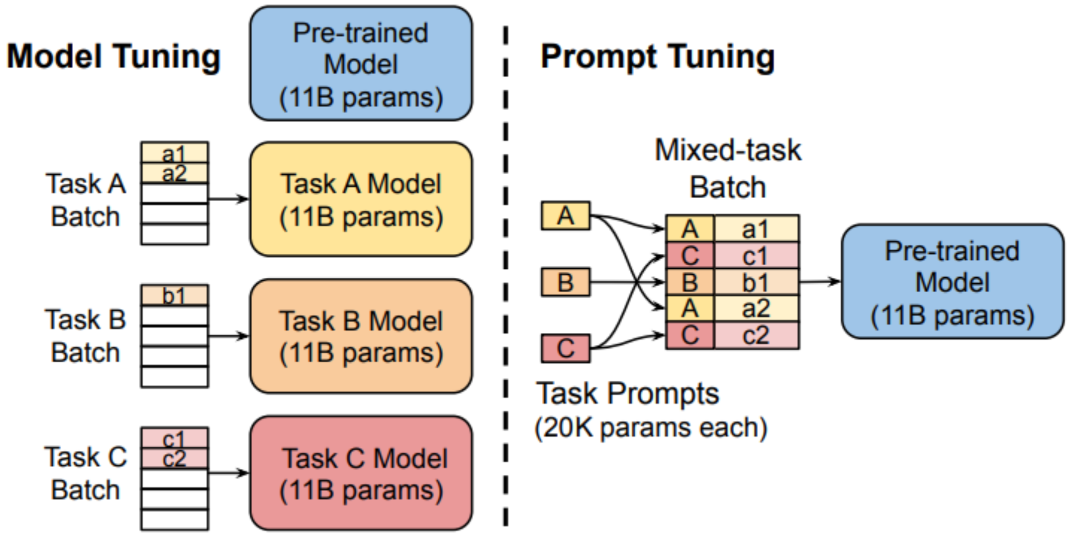
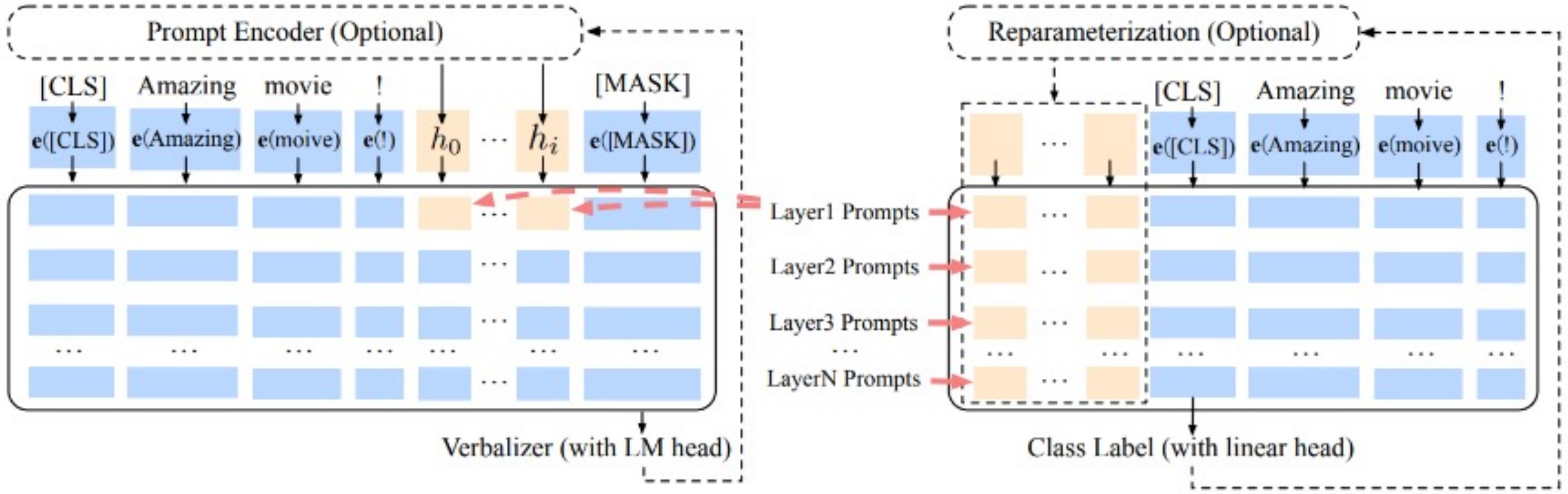

Model Tuning
Pre-trained Model (11B params)
Task A Batch
a1
a2
Task A Model (11B params)
Task B Batch
b1
Task B Model (11B params)
Task C Batch
c1
c2
Task C Model (11B params)
Prompt Tuning
Mixed-task Batch
A
B
C
A a1
C c1
B b1
A a2
C c2
Pre-trained Model (11B params)
Task Prompts (20K params each)

Figure 2: **Model tuning** requires making a task-specific copy of the entire pre-trained model for each downstream task and inference must be performed in separate batches. **Prompt tuning** only requires storing a small task-specific prompt for each task, and enables mixed-task inference using the original pre-trained model. With a T5 "XXL" model, each copy of the tuned model requires 11 billion parameters. By contrast, our tuned prompts would only require 20,480 parameters per task—a reduction of *over five orders of magnitude*—assuming a prompt length of 5 tokens.



(a) Lester et al. & P-tuning (Frozen, 10-billion-scale, simple tasks)

(b) P-tuning v2 (Frozen, most scales, most tasks)

Figure 2: From Lester et al. (2021) & P-tuning to P-tuning v2. Orange tokens (include h_0, h_i) refer to prompt embeddings we add; blue tokens are embeddings stored or computed by frozen pre-trained language models. Compared to Lester et al. (2021), P-tuning v2 adds trainable continuous prompts to inputs of every transformer layer independently (as prefix-tuning (Li and Liang, 2021) does). Additionally, P-tuning v2 removes verbalizers with LM head and returns to the traditional class labels with ordinary linear head to allow its task-universality.

Brian Lester, Rami Al–Rfou, Noah Constant The Power of Scale for Parameter-Efficient Prompt Tuning. 2021. https://arxiv.org/abs/2104.08691
Xiao Liu et al. " P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. 2021. https://arxiv.org/abs/2110.07602

DOWNSTREAM TASKS

Customize Models using Parameter-efficient tuning | Adapters

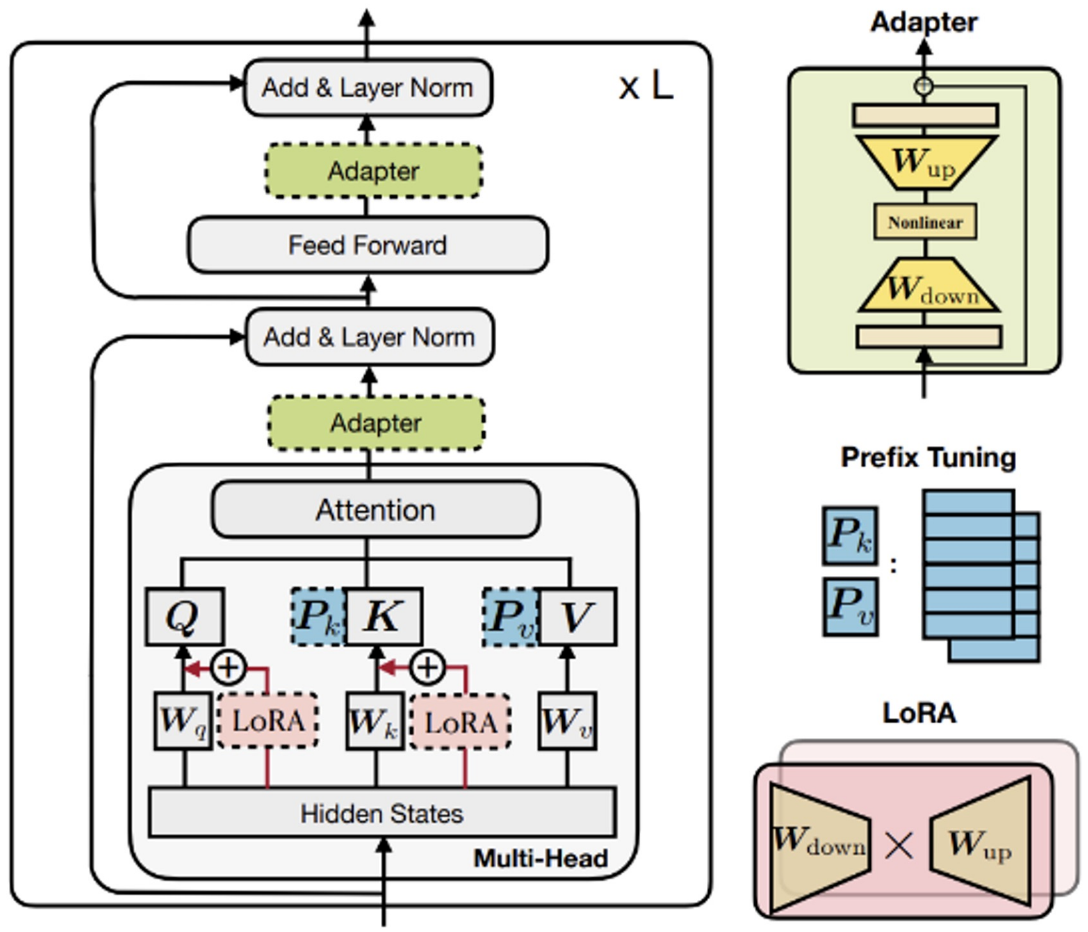
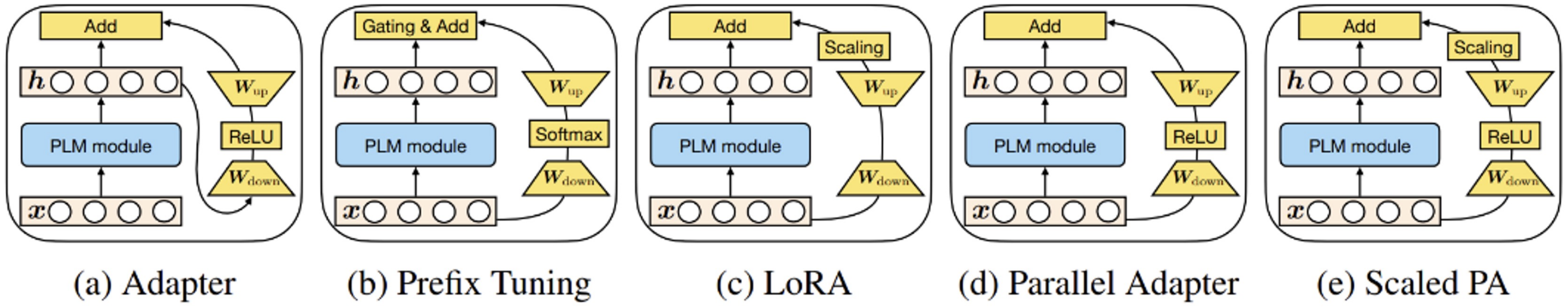


Figure 1: Illustration of the transformer architecture and several state-of-the-art parameter-efficient tuning methods. We use blocks with dashed borderlines to represent the added modules by those methods.



(a) Adapter (b) Prefix Tuning (c) LoRA (d) Parallel Adapter (e) Scaled PA

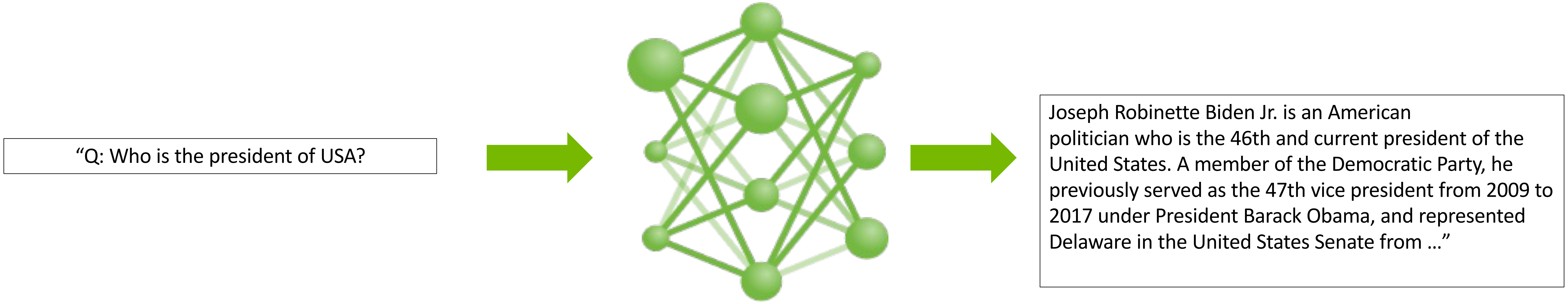
Figure 3: Graphical illustration of existing methods and the proposed variants. "PLM module" represents a certain sublayer of the PLM (e.g. attention or FFN) that is frozen. "Scaled PA" denotes scaled parallel adapter. We do not include multi-head parallel adapter here to save space.

Junxian He Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, Graham Neubig. TOWARDS A UNIFIED VIEW OF PARAMETER-EFFICIENT TRANSFER LEARNING. ICLR 2022

INSTRUCTED LLM

INSTRUCTIONS FINETUNING

Dataset of Instructions (Queries and Answers)

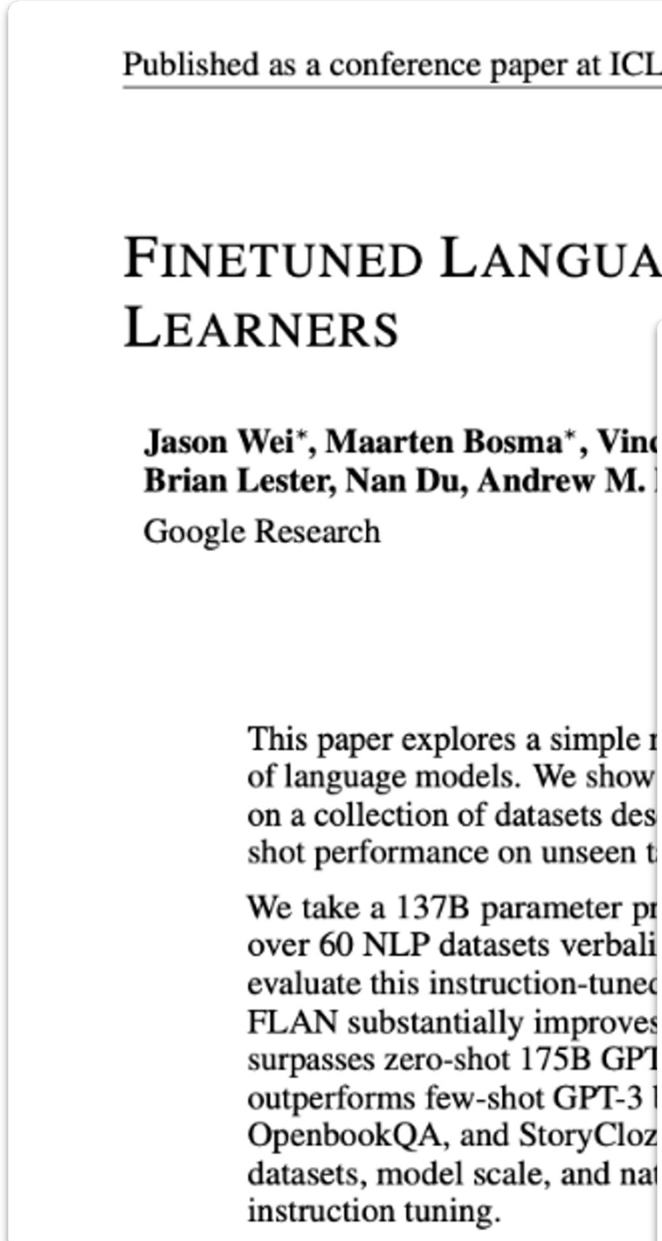


“Q: Who is the president of USA?

Joseph Robinette Biden Jr. is an American politician who is the 46th and current president of the United States. A member of the Democratic Party, he previously served as the 47th vice president from 2009 to 2017 under President Barack Obama, and represented Delaware in the United States Senate from …”

INSTRUCTIONS FINETUNING

FLAN [Google]



Published as a conference paper at ICLR 2022

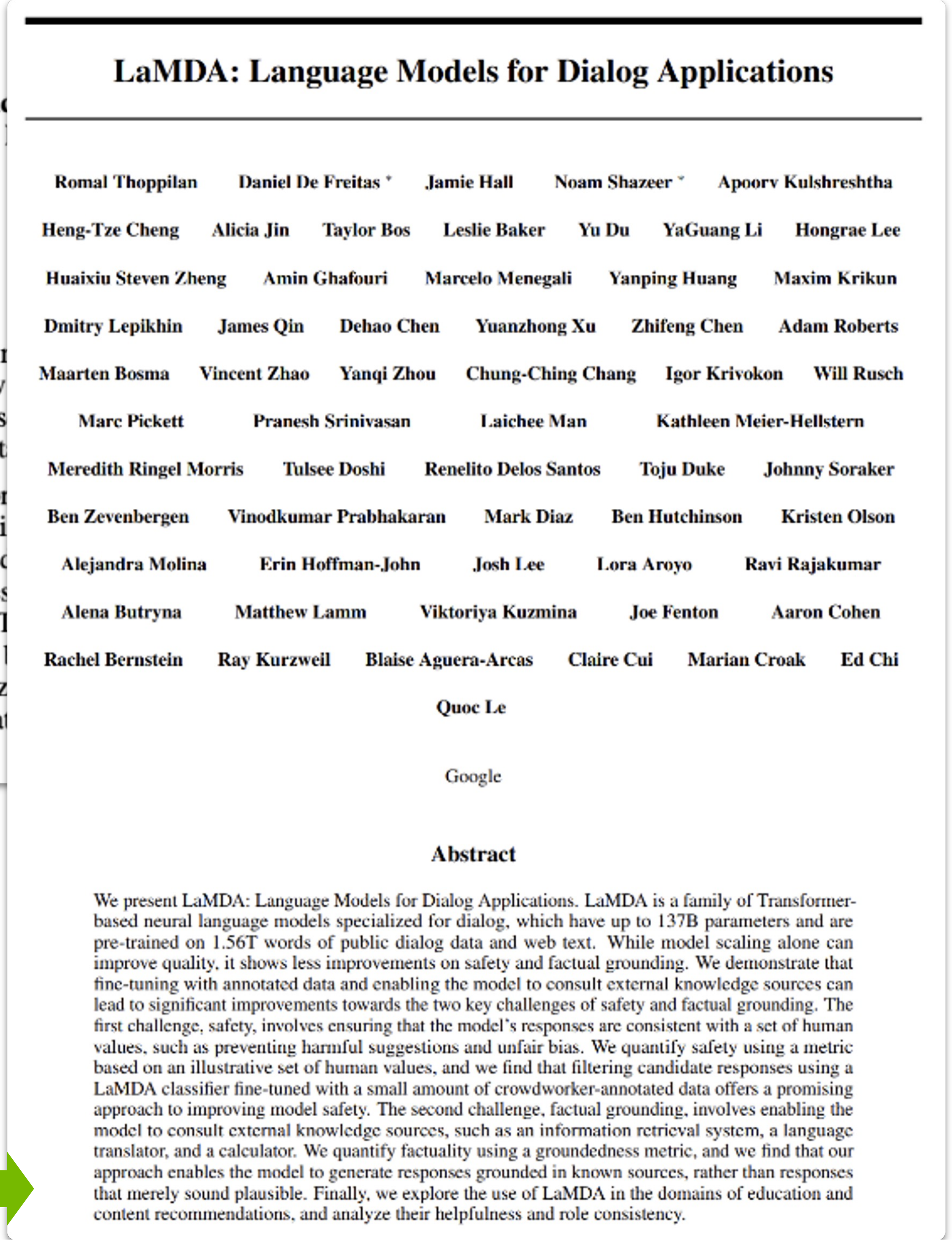
FINETUNED LANGUAGE MODELS ARE ZERO-SHOT LEARNERS

Jason Wei*, Maarten Bosma*, Vinc
Brian Lester, Nan Du, Andrew M.
Google Research

This paper explores a simple
of language models. We show
on a collection of datasets des
shot performance on unseen t

We take a 137B parameter pr
over 60 NLP datasets verbali
evaluate this instruction-tuned
FLAN substantially improves
surpasses zero-shot 175B GPT
outperforms few-shot GPT-3
OpenbookQA, and StoryCloz
datasets, model scale, and nat
instruction tuning.

LaMDA [Google]



LaMDA: Language Models for Dialog Applications

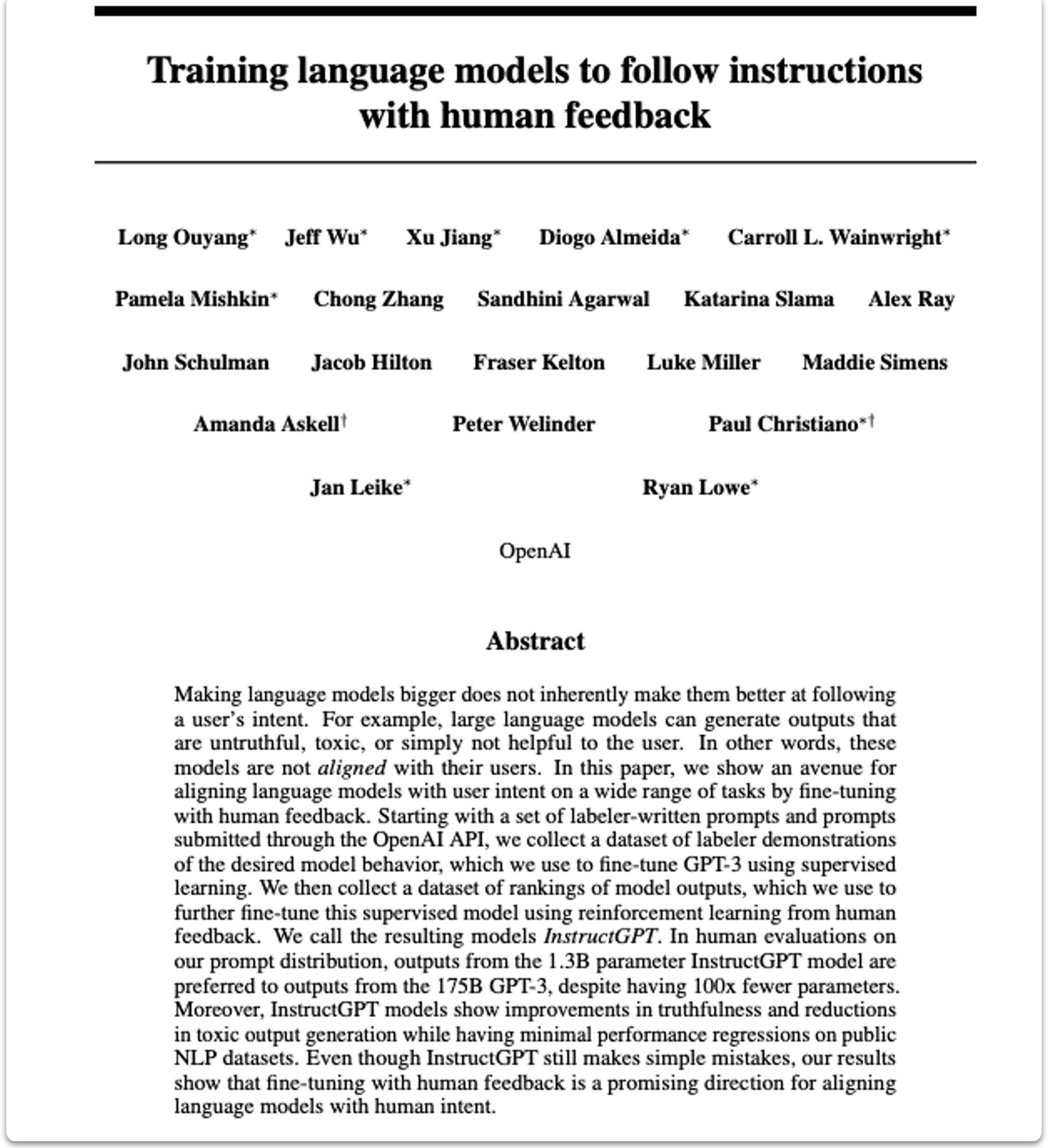
Romal Thoppilan Daniel De Freitas * Jamie Hall Noam Shazeer * Apoorv Kulshreshtha
Heng-Tze Cheng Alicia Jin Taylor Bos Leslie Baker Yu Du YaGuang Li Hongrae Lee
Huaixiu Steven Zheng Amin Ghafouri Marcelo Menegali Yanping Huang Maxim Krikun
Dmitry Lepikhin James Qin Dehao Chen Yuanzhong Xu Zhifeng Chen Adam Roberts
Maarten Bosma Vincent Zhao Yanqi Zhou Chung-Ching Chang Igor Krivokon Will Rusch
Marc Pickett Pranesh Srinivasan Laichee Man Kathleen Meier-Hellstern
Meredith Ringel Morris Tulsee Doshi Renelito Delos Santos Toju Duke Johnny Soraker
Ben Zevenbergen Vinodkumar Prabhakaran Mark Diaz Ben Hutchinson Kristen Olson
Alejandra Molina Erin Hoffman-John Josh Lee Lora Aroyo Ravi Rajakumar
Alena Butryna Matthew Lamm Viktoriya Kuzmina Joe Fenton Aaron Cohen
Rachel Bernstein Ray Kurzweil Blaise Aguera-Arcas Claire Cui Marian Croak Ed Chi
Quoc Le

Google

Abstract

We present LaMDA: Language Models for Dialog Applications. LaMDA is a family of Transformer-based neural language models specialized for dialog, which have up to 137B parameters and are pre-trained on 1.56T words of public dialog data and web text. While model scaling alone can improve quality, it shows less improvements on safety and factual grounding. We demonstrate that fine-tuning with annotated data and enabling the model to consult external knowledge sources can lead to significant improvements towards the two key challenges of safety and factual grounding. The first challenge, safety, involves ensuring that the model's responses are consistent with a set of human values, such as preventing harmful suggestions and unfair bias. We quantify safety using a metric based on an illustrative set of human values, and we find that filtering candidate responses using a LaMDA classifier fine-tuned with a small amount of crowdworker-annotated data offers a promising approach to improving model safety. The second challenge, factual grounding, involves enabling the model to consult external knowledge sources, such as an information retrieval system, a language translator, and a calculator. We quantify factuality using a groundedness metric, and we find that our approach enables the model to generate responses grounded in known sources, rather than responses that merely sound plausible. Finally, we explore the use of LaMDA in the domains of education and content recommendations, and analyze their helpfulness and role consistency.

InstructGPT [OpenAI]



Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
Amanda Askell† Peter Welinder Paul Christiano*†
Jan Leike* Ryan Lowe*

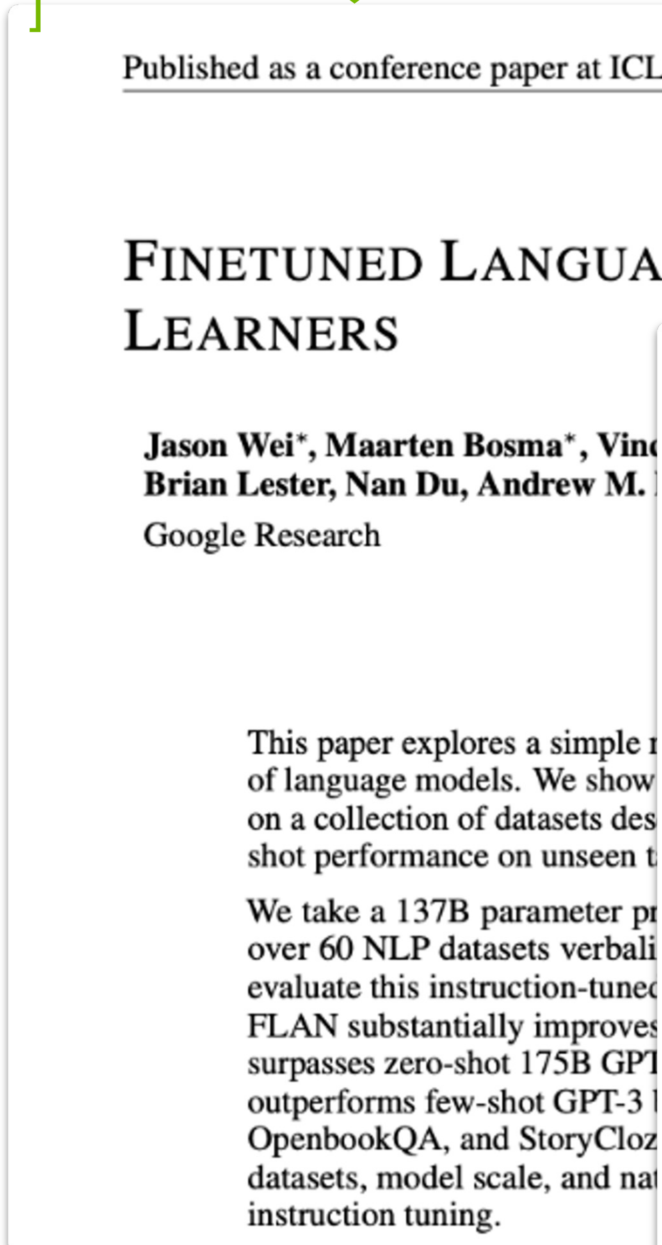
OpenAI

Abstract

Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, our results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

INSTRUCTIONS FINETUNING

FLAN [Google]



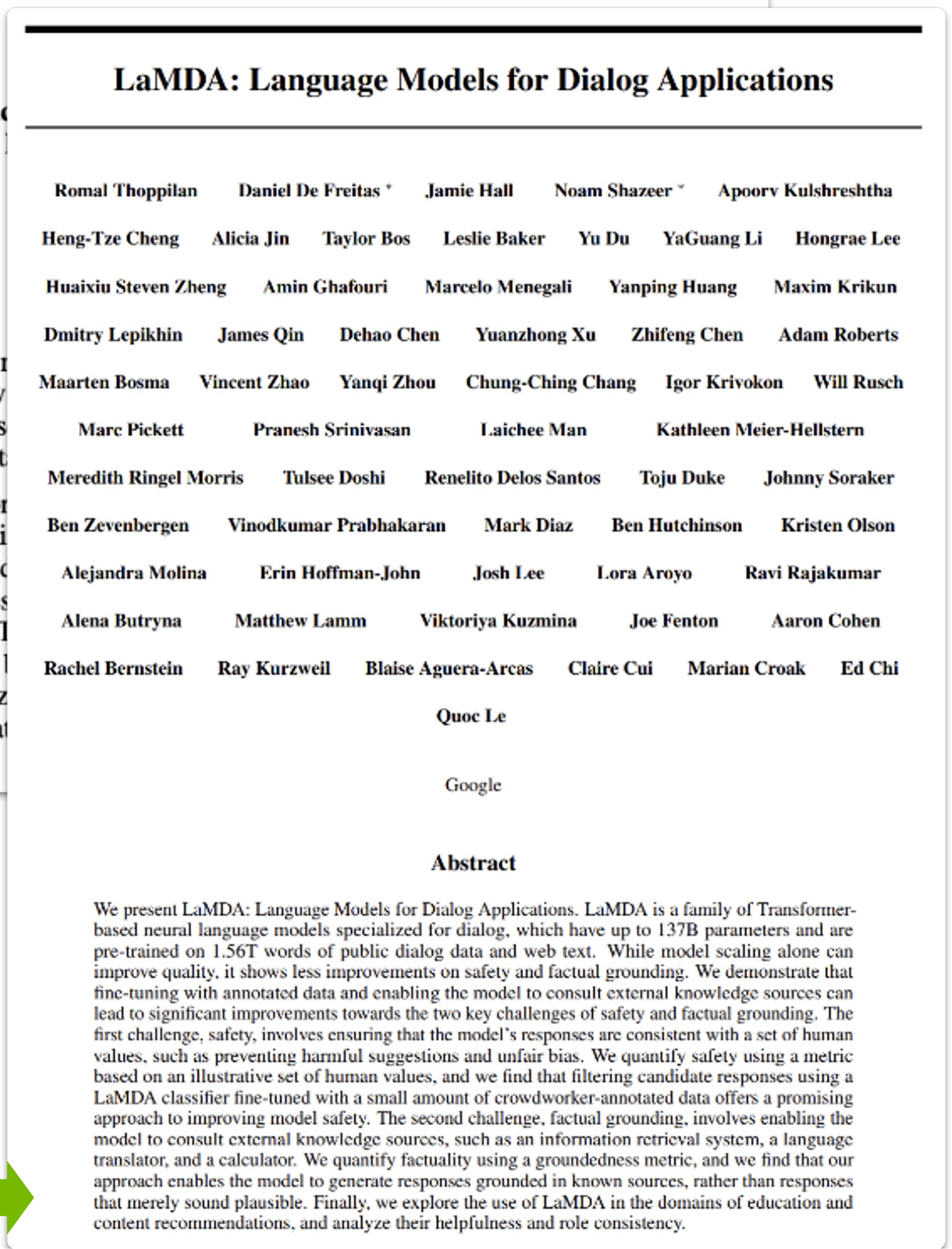
Published as a conference paper at ICLR 2022

FINETUNED LANGUAGE MODELS ARE ZERO-SHOT LEARNERS

Jason Wei*, Maarten Bosma*, Vinc
Brian Lester, Nan Du, Andrew M.
Google Research

This paper explores a simple
of language models. We show
on a collection of datasets des
shot performance on unseen t

We take a 137B parameter pr
over 60 NLP datasets verbali
evaluate this instruction-tuned
FLAN substantially improves
surpasses zero-shot 175B GPT
outperforms few-shot GPT-3
OpenbookQA, and StoryCloz
datasets, model scale, and nat
instruction tuning.



LaMDA: Language Models for Dialog Applications

Romal Thoppilan Daniel De Freitas* Jamie Hall Noam Shazeer* Apoorv Kulshreshtha
Heng-Tze Cheng Alicia Jin Taylor Bos Leslie Baker Yu Du YaGuang Li Hongrae Lee
Huaixiu Steven Zheng Amin Ghafouri Marcelo Menegali Yanping Huang Maxim Krikun
Dmitry Lepikhin James Qin Dehao Chen Yuanzhong Xu Zhifeng Chen Adam Roberts
Maarten Bosma Vincent Zhao Yanqi Zhou Chung-Ching Chang Igor Krivokon Will Rusch
Marc Pickett Pranesh Srinivasan Laichee Man Kathleen Meier-Hellstern
Meredith Ringel Morris Tulsee Doshi Renelito Delos Santos Toju Duke Johnny Soraker
Ben Zevenbergen Vinodkumar Prabhakaran Mark Diaz Ben Hutchinson Kristen Olson
Alejandra Molina Erin Hoffman-John Josh Lee Lora Aroyo Ravi Rajakumar
Alena Butryna Matthew Lamm Viktoriya Kuzmina Joe Fenton Aaron Cohen
Rachel Bernstein Ray Kurzweil Blaise Aguera-Arcas Claire Cui Marian Croak Ed Chi
Quoc Le

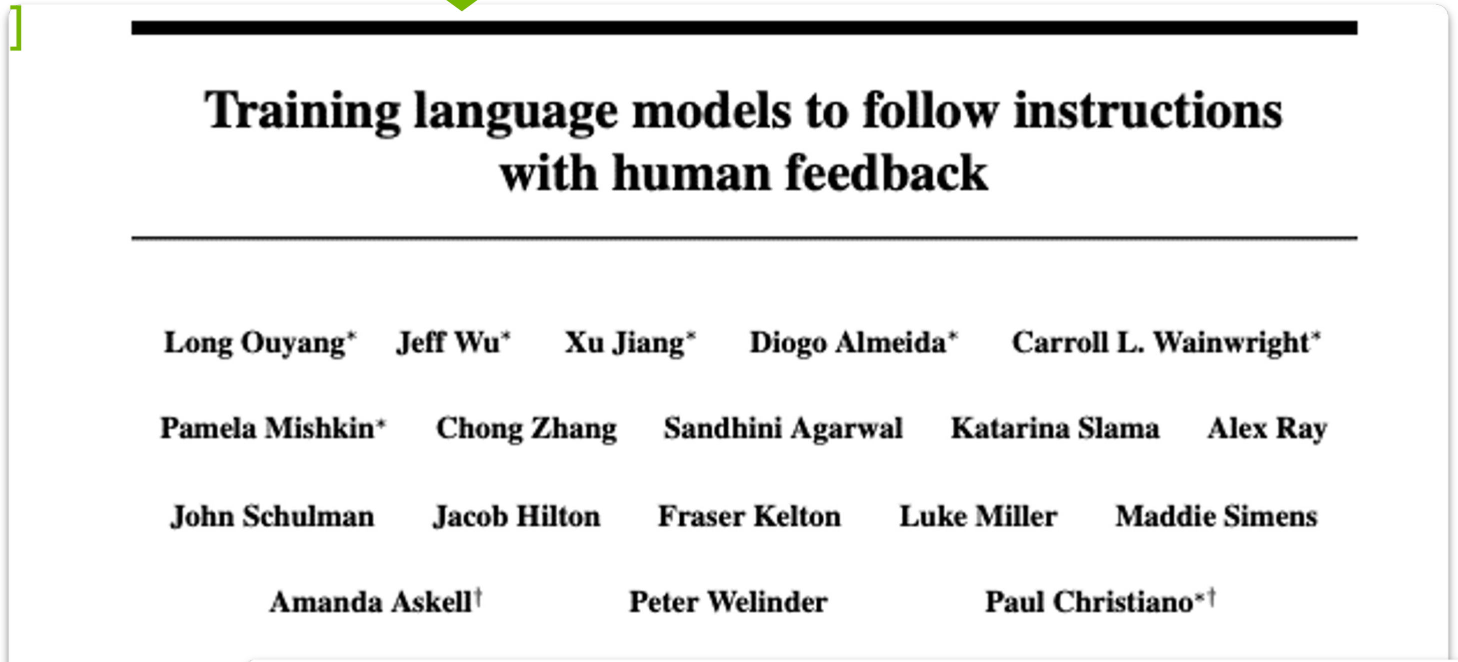
Google

Abstract

We present LaMDA: Language Models for Dialog Applications. LaMDA is a family of Transformer-based neural language models specialized for dialog, which have up to 137B parameters and are pre-trained on 1.56T words of public dialog data and web text. While model scaling alone can improve quality, it shows less improvements on safety and factual grounding. We demonstrate that fine-tuning with annotated data and enabling the model to consult external knowledge sources can lead to significant improvements towards the two key challenges of safety and factual grounding. The first challenge, safety, involves ensuring that the model's responses are consistent with a set of human values, such as preventing harmful suggestions and unfair bias. We quantify safety using a metric based on an illustrative set of human values, and we find that filtering candidate responses using a LaMDA classifier fine-tuned with a small amount of crowdworker-annotated data offers a promising approach to improving model safety. The second challenge, factual grounding, involves enabling the model to consult external knowledge sources, such as an information retrieval system, a language translator, and a calculator. We quantify factuality using a groundedness metric, and we find that our approach enables the model to generate responses grounded in known sources, rather than responses that merely sound plausible. Finally, we explore the use of LaMDA in the domains of education and content recommendations, and analyze their helpfulness and role consistency.

LaMDA [Google]

InstructGPT [OpenA]



Training language models to follow instructions with human feedback

Long Ouyang* **Jeff Wu*** **Xu Jiang*** **Diogo Almeida*** **Carroll L. Wainwright***
Pamela Mishkin* **Chong Zhang** **Sandhini Agarwal** **Katarina Slama** **Alex Ray**
John Schulman **Jacob Hilton** **Fraser Kelton** **Luke Miller** **Maddie Simens**
Amanda Askell† **Peter Welinder** **Paul Christiano*†**

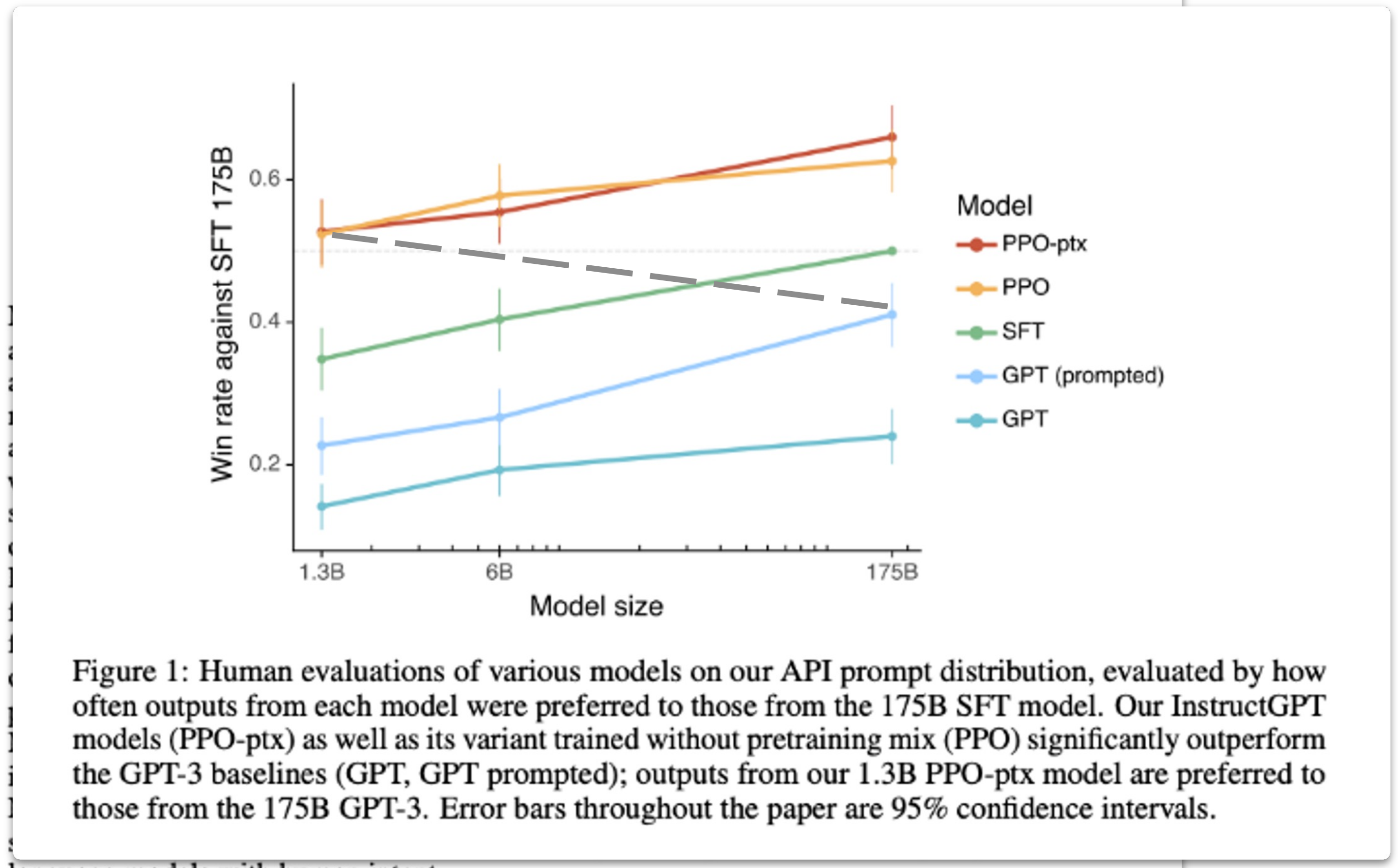


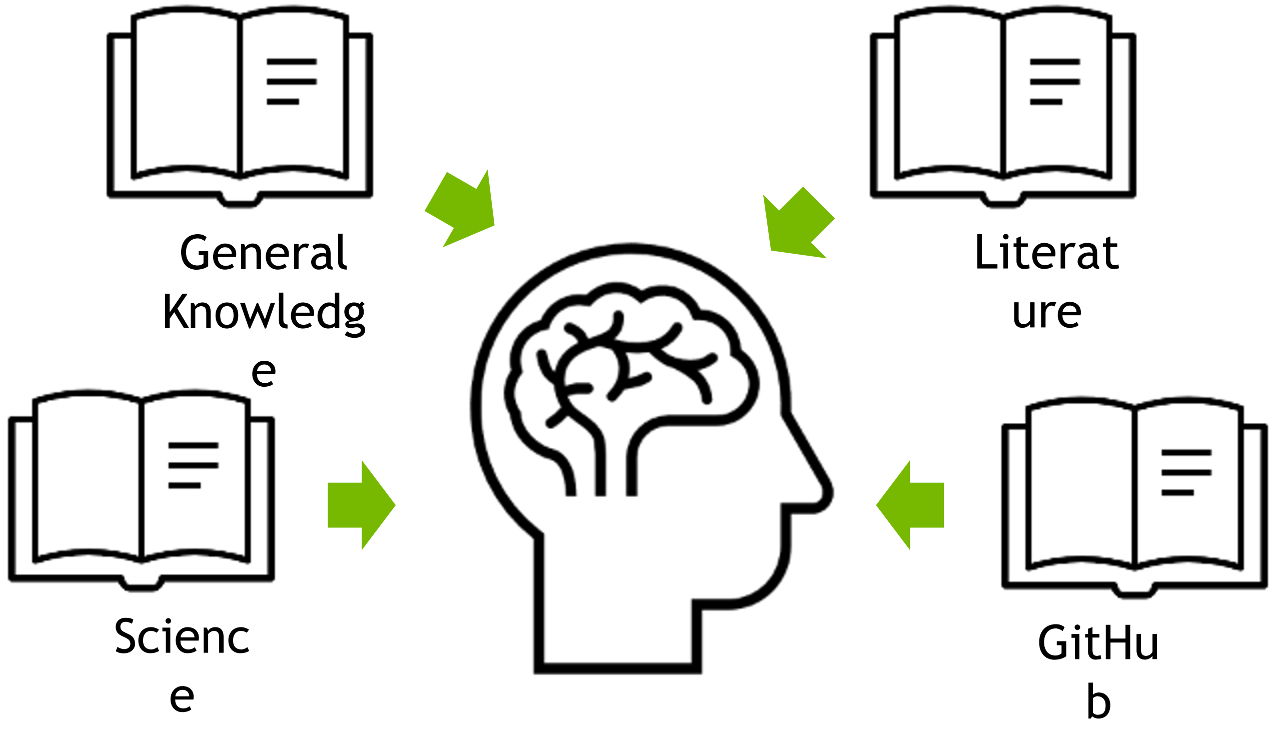
Figure 1: Human evaluations of various models on our API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.

language models with human intent.

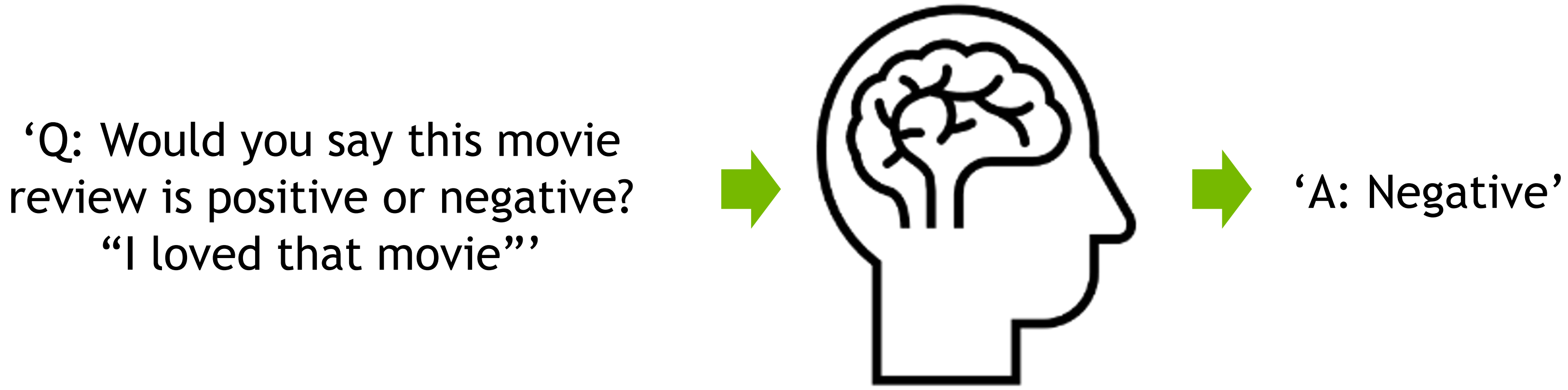
CHANGE IN THE NLP PARADIGM

NEW NLP APPROACH (CIRCA 2021)

Step 1: Train a Very Deep/HUGE model



Step 2. Ask questions



‘Q: Would you say this movie review is positive or negative? “I loved that movie”’

‘A: Negative’

Huge means Billions of parameters

TOWARDS GENERAL INTELLIGENCE

Old way

- Needs Labelled data
  - Cost of data collection/labelling
  - Legal/Privacy concerns around using data
- 1 model per task results in
  - Increased model development/tuning cost
  - Increased operational costs
  - Increased money spent on sourcing data
- Relatively Limited generalization
- Computationally cheaper (~300 Million parameters)

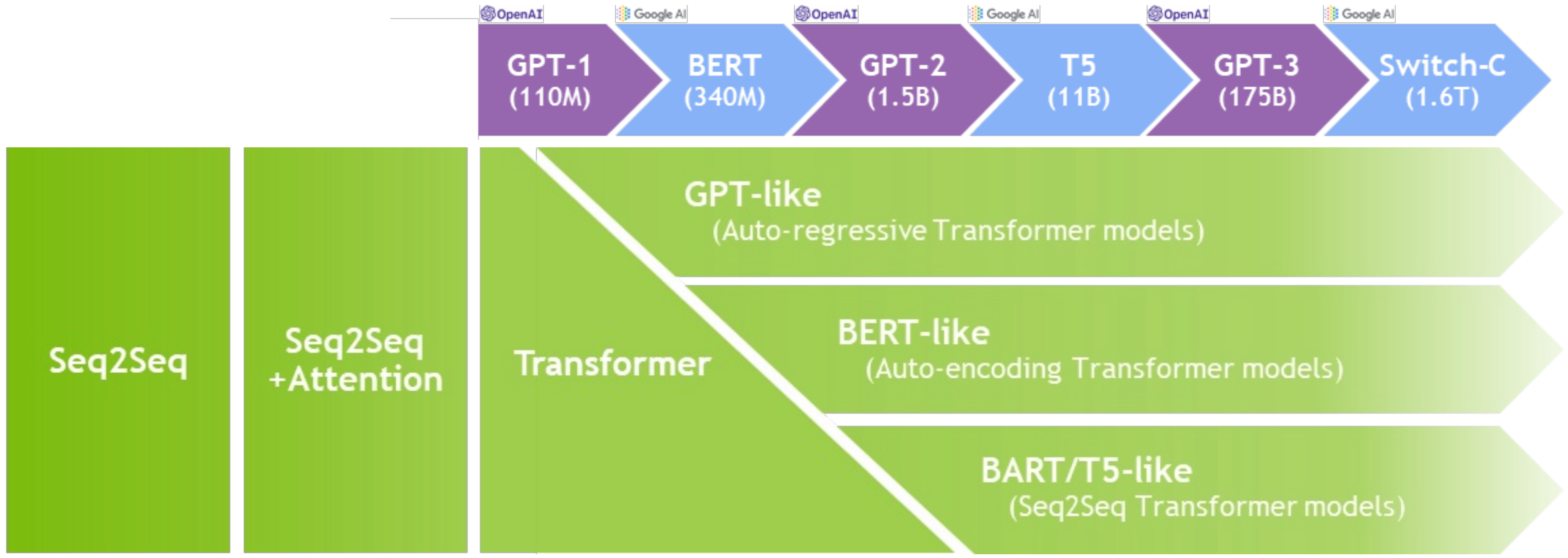
New way

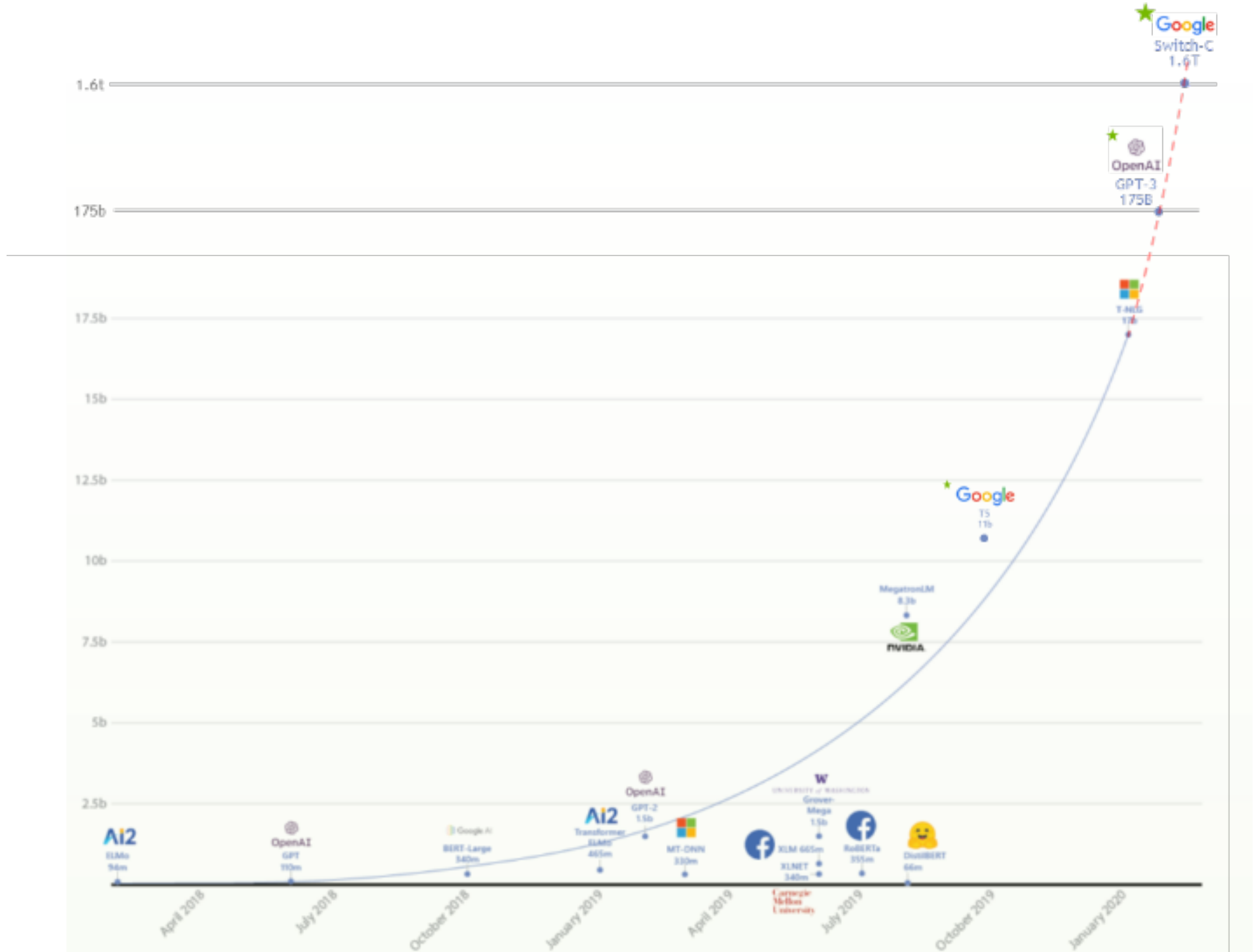
- Does not need labelled data
- Single generic model can do more than one tasks
- More generalized: in addition to language also learns higher level concepts, styles, etc.
- Computationally Expensive (~500 Billion parameters)

Leveraging more compute to get a general model without significant data/labelling cost

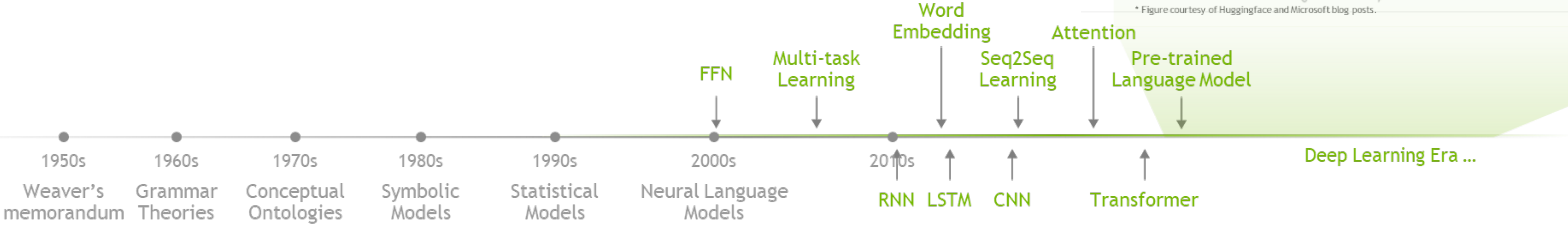
HISTORY OF LANGUAGE MODELS

Language Model became more complex and larger



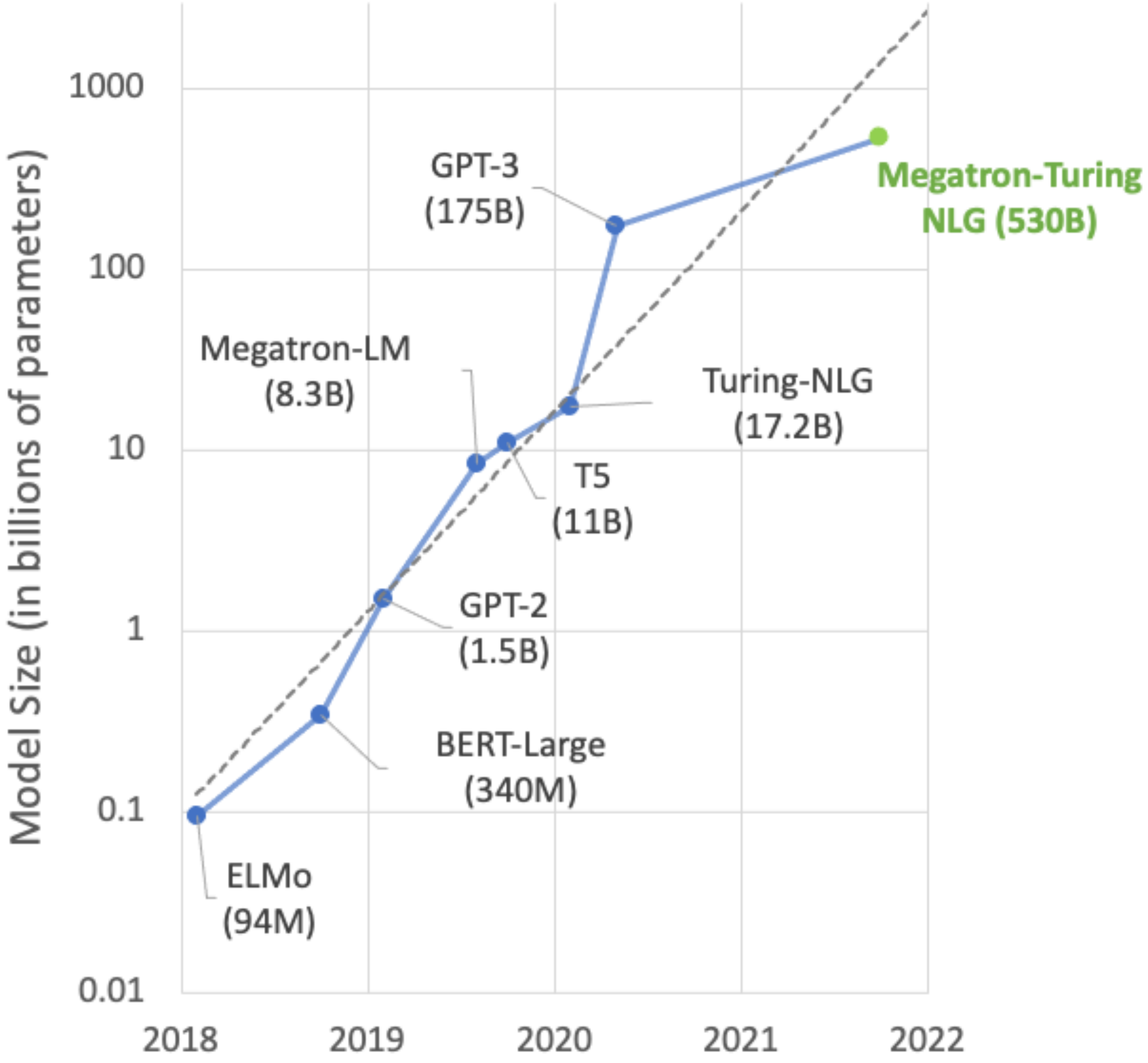


* Figure courtesy of Huggingface and Microsoft blog posts.



MEGATRON-TURING NLG 530B

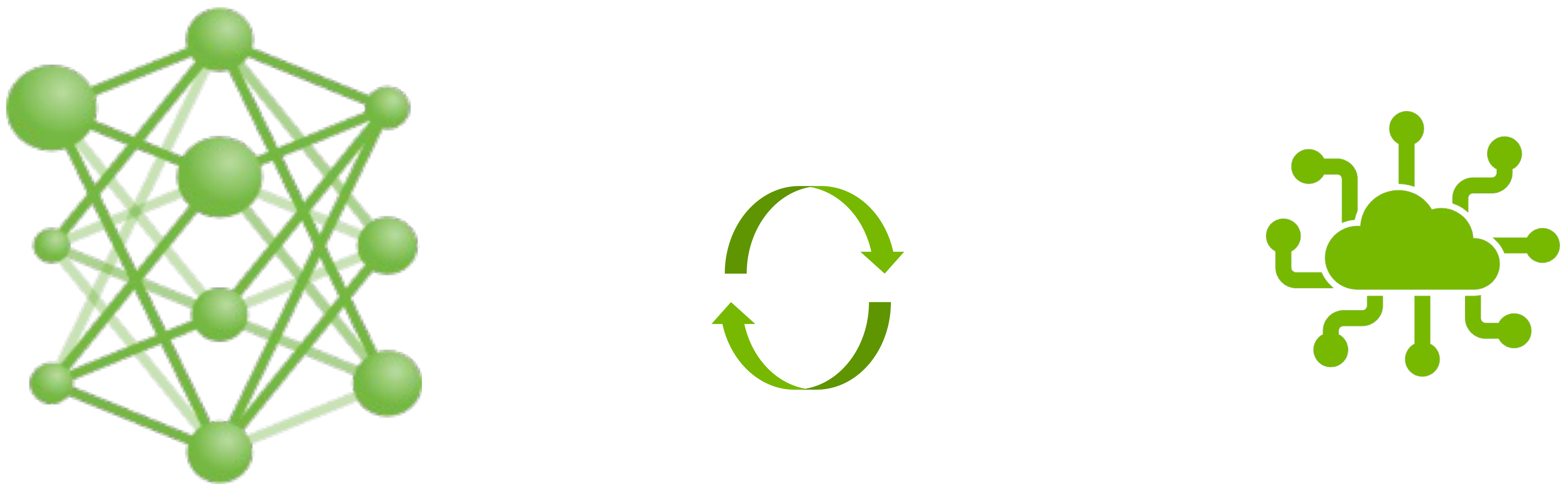
The Trend Continues



https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

LLM FOR SEARCH

EXTERNAL KNOWLEDGE SOURCE

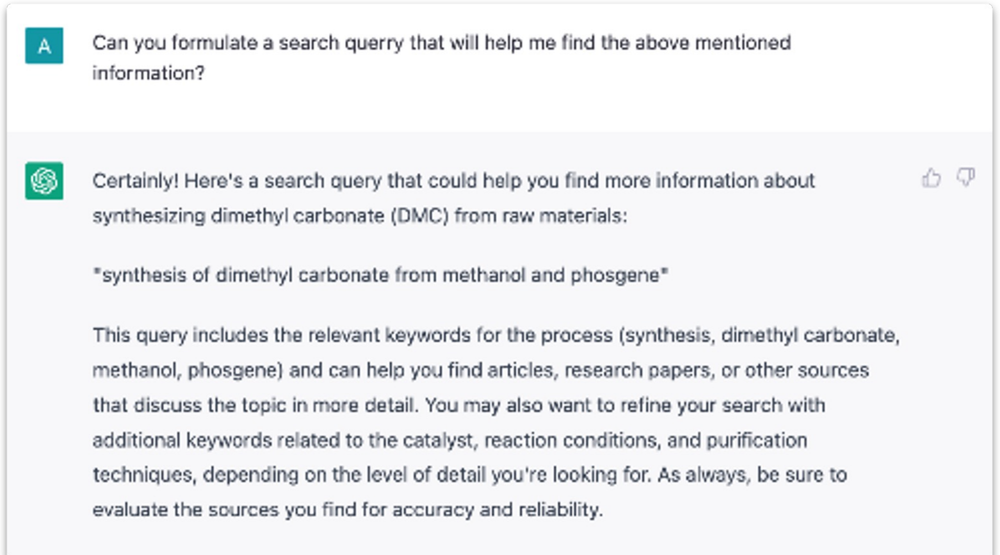


Instructed LLM

Knowledge source

EXTERNAL KNOWLEDGE SOURCE

Searching through the web

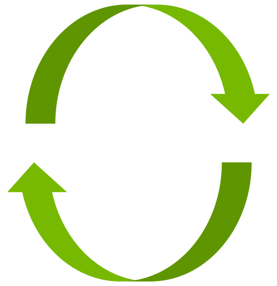


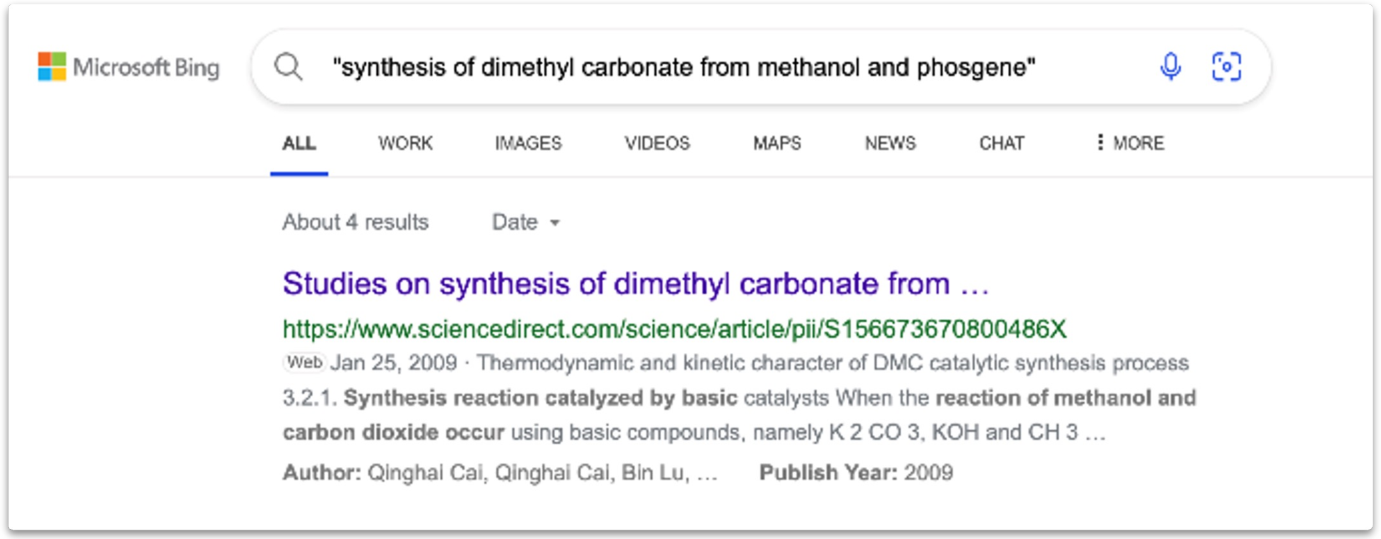
Can you formulate a search querry that will help me find the above mentioned information?

Certainly! Here's a search query that could help you find more information about synthesizing dimethyl carbonate (DMC) from raw materials:

"synthesis of dimethyl carbonate from methanol and phosgene"

This query includes the relevant keywords for the process (synthesis, dimethyl carbonate, methanol, phosgene) and can help you find articles, research papers, or other sources that discuss the topic in more detail. You may also want to refine your search with additional keywords related to the catalyst, reaction conditions, and purification techniques, depending on the level of detail you're looking for. As always, be sure to evaluate the sources you find for accuracy and reliability.





"synthesis of dimethyl carbonate from methanol and phosgene"

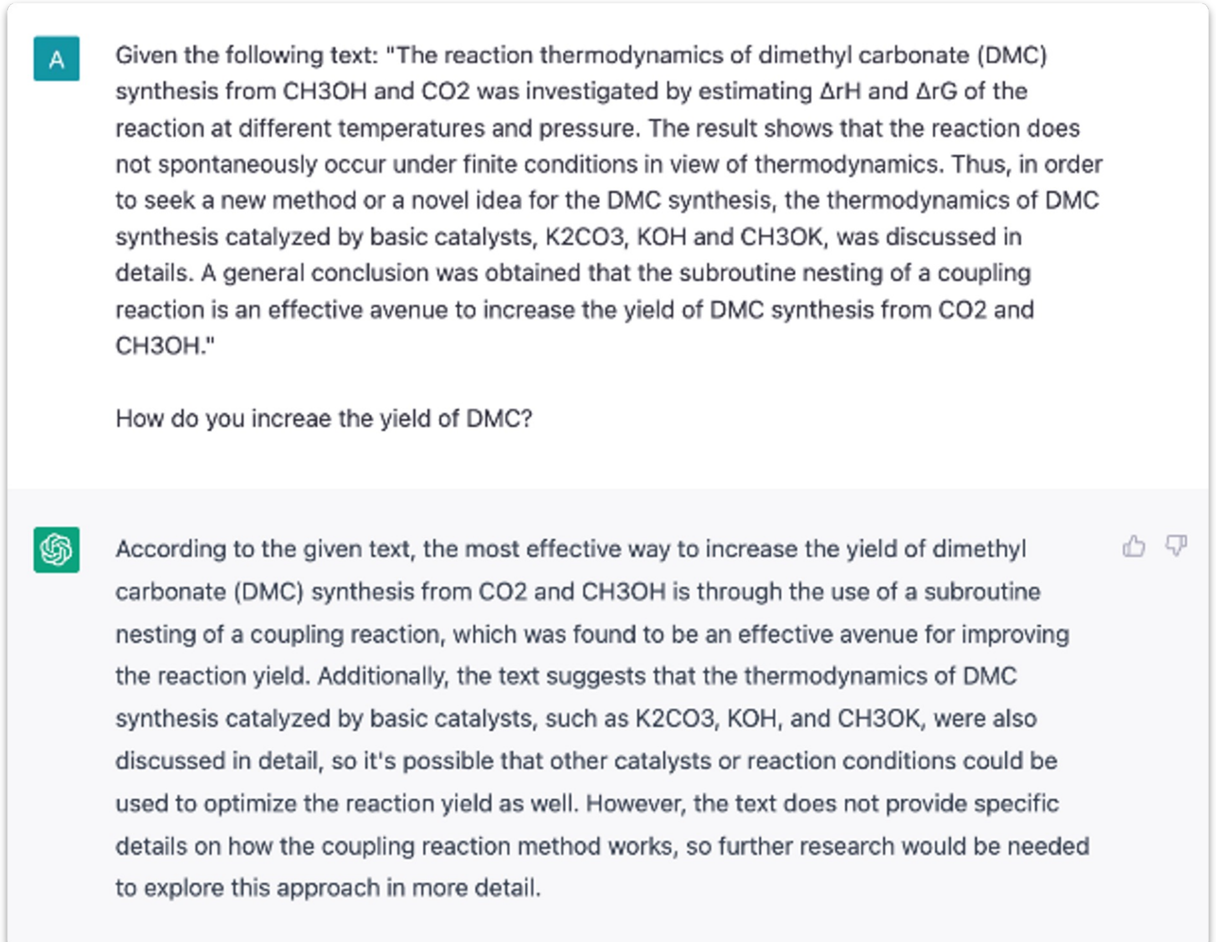
About 4 results

Studies on synthesis of dimethyl carbonate from …

https://www.sciencedirect.com/science/article/pii/S156673670800486X

Web Jan 25, 2009 · Thermodynamic and kinetic character of DMC catalytic synthesis process 3.2.1. **Synthesis reaction catalyzed by basic** catalysts When the **reaction of methanol and carbon dioxide occur** using basic compounds, namely K 2 CO 3, KOH and CH 3 …

Author: Qinghai Cai, Qinghai Cai, Bin Lu, … **Publish Year:** 2009



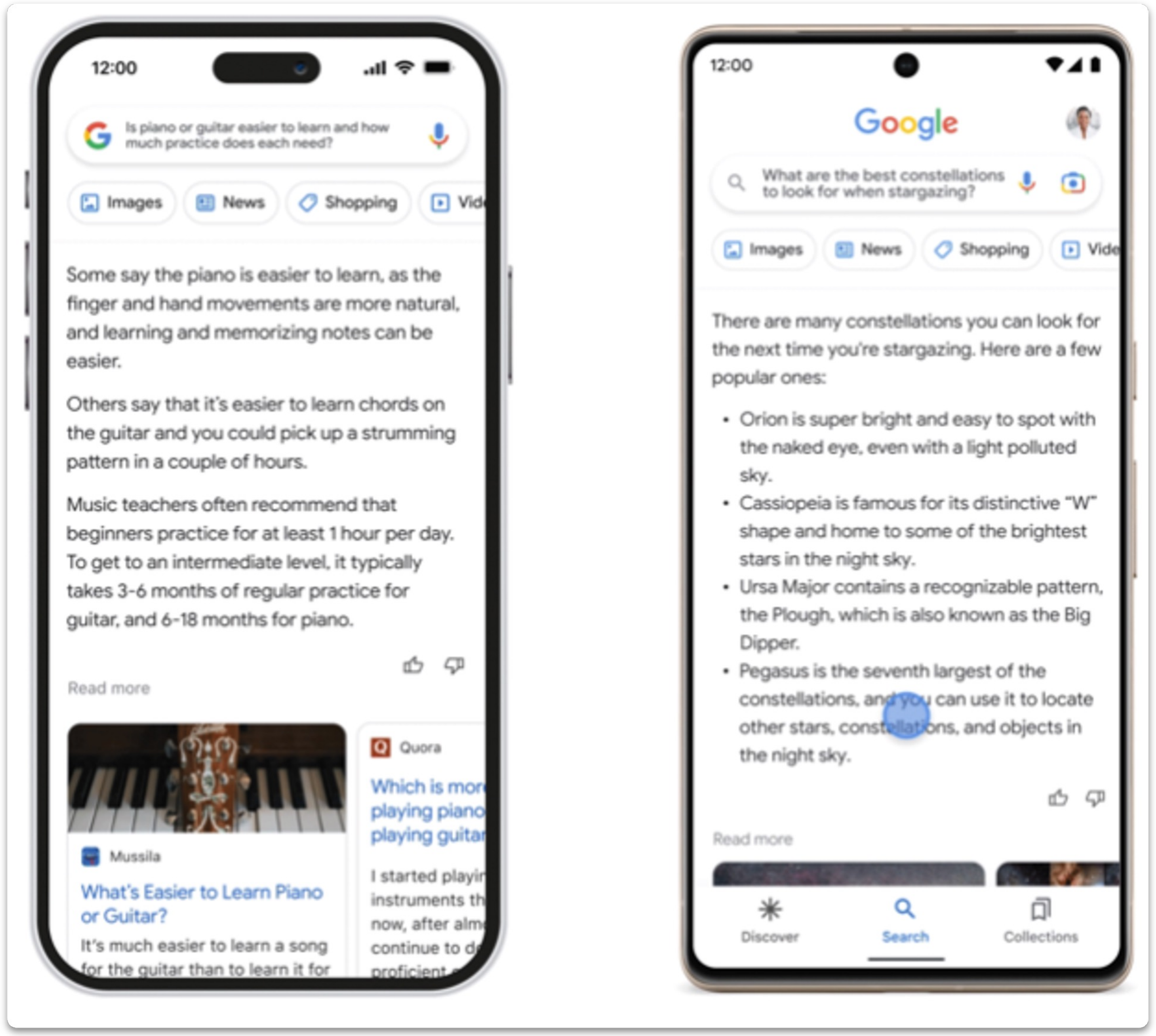
Given the following text: "The reaction thermodynamics of dimethyl carbonate (DMC) synthesis from CH3OH and CO2 was investigated by estimating ΔrH and ΔrG of the reaction at different temperatures and pressure. The result shows that the reaction does not spontaneously occur under finite conditions in view of thermodynamics. Thus, in order to seek a new method or a novel idea for the DMC synthesis, the thermodynamics of DMC synthesis catalyzed by basic catalysts, K2CO3, KOH and CH3OK, was discussed in details. A general conclusion was obtained that the subroutine nesting of a coupling reaction is an effective avenue to increase the yield of DMC synthesis from CO2 and CH3OH."

How do you increae the yield of DMC?

According to the given text, the most effective way to increase the yield of dimethyl carbonate (DMC) synthesis from CO2 and CH3OH is through the use of a subroutine nesting of a coupling reaction, which was found to be an effective avenue for improving the reaction yield. Additionally, the text suggests that the thermodynamics of DMC synthesis catalyzed by basic catalysts, such as K2CO3, KOH, and CH3OK, were also discussed in detail, so it's possible that other catalysts or reaction conditions could be used to optimize the reaction yield as well. However, the text does not provide specific details on how the coupling reaction method works, so further research would be needed to explore this approach in more detail.

EXTERNAL KNOWLEDGE SOURCE

Examples of Search Engine Powered by LLM



Is piano or guitar easier to learn and how much practice does each need?

Some say the piano is easier to learn, as the finger and hand movements are more natural, and learning and memorizing notes can be easier.

Others say that it's easier to learn chords on the guitar and you could pick up a strumming pattern in a couple of hours.

Music teachers often recommend that beginners practice for at least 1 hour per day. To get to an intermediate level, it typically takes 3-6 months of regular practice for guitar, and 6-18 months for piano.

Mussila

What's Easier to Learn Piano or Guitar?

It's much easier to learn a song for the guitar than to learn it for

Quora

Which is mor playing piano playing guitar

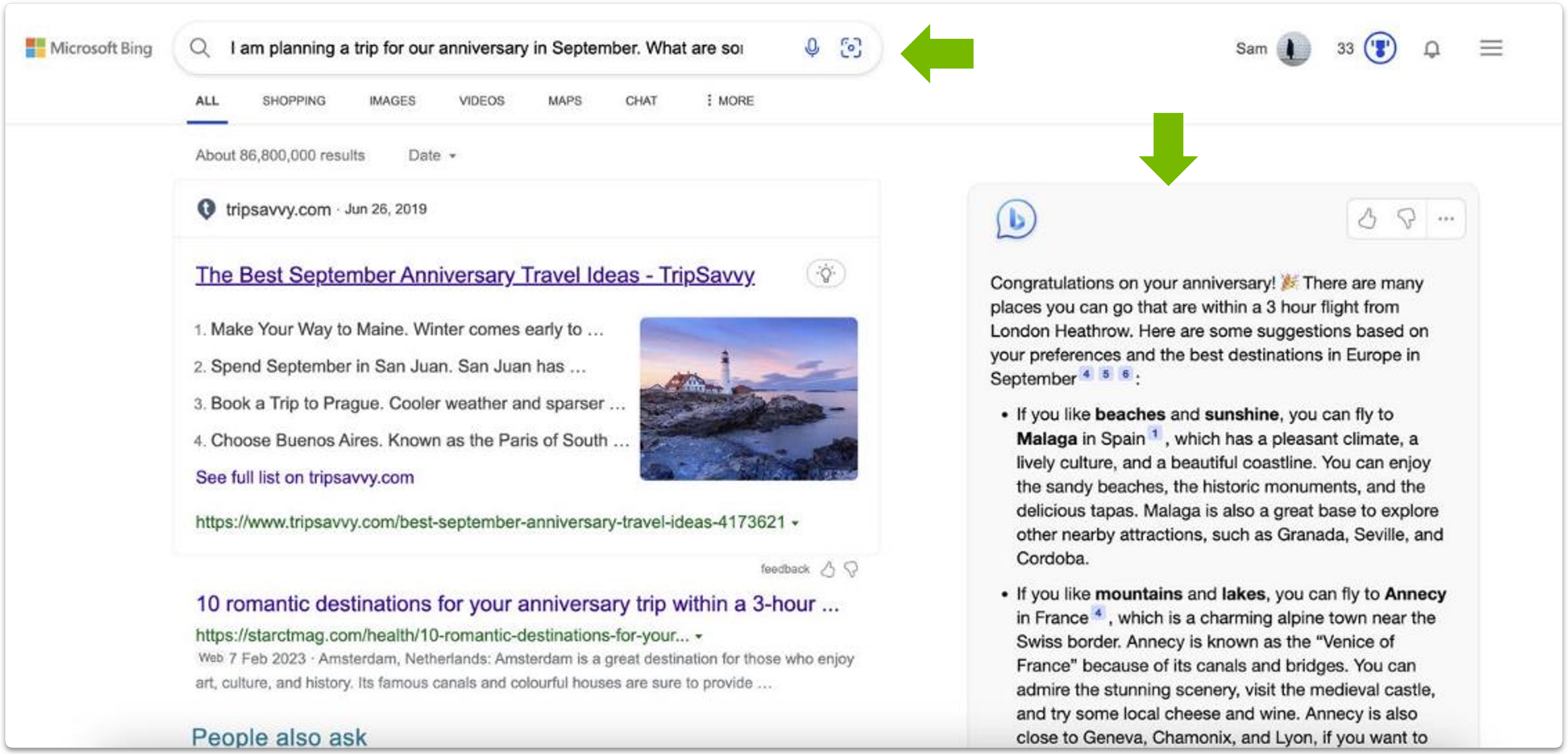
I started playin instruments th now, after alm continue to d proficient

What are the best constellations to look for when stargazing?

There are many constellations you can look for the next time you're stargazing. Here are a few popular ones:

- Orion is super bright and easy to spot with the naked eye, even with a light polluted sky.
- Cassiopeia is famous for its distinctive "W" shape and home to some of the brightest stars in the night sky.
- Ursa Major contains a recognizable pattern, the Plough, which is also known as the Big Dipper.
- Pegasus is the seventh largest of the constellations, and you can use it to locate other stars, constellations, and objects in the night sky.

NORA - No One Right Answer [Google]



I am planning a trip for our anniversary in September. What are soı

About 86,800,000 results

tripsavvy.com · Jun 26, 2019

The Best September Anniversary Travel Ideas - TripSavvy

1. Make Your Way to Maine. Winter comes early to …
2. Spend September in San Juan. San Juan has …
3. Book a Trip to Prague. Cooler weather and sparser …
4. Choose Buenos Aires. Known as the Paris of South …

See full list on tripsavvy.com

https://www.tripsavvy.com/best-september-anniversary-travel-ideas-4173621

10 romantic destinations for your anniversary trip within a 3-hour …

https://starctmag.com/health/10-romantic-destinations-for-your...

Web 7 Feb 2023 · Amsterdam, Netherlands: Amsterdam is a great destination for those who enjoy art, culture, and history. Its famous canals and colourful houses are sure to provide …

People also ask

Congratulations on your anniversary! 🎉 There are many places you can go that are within a 3 hour flight from London Heathrow. Here are some suggestions based on your preferences and the best destinations in Europe in September [4] [5] [6]:

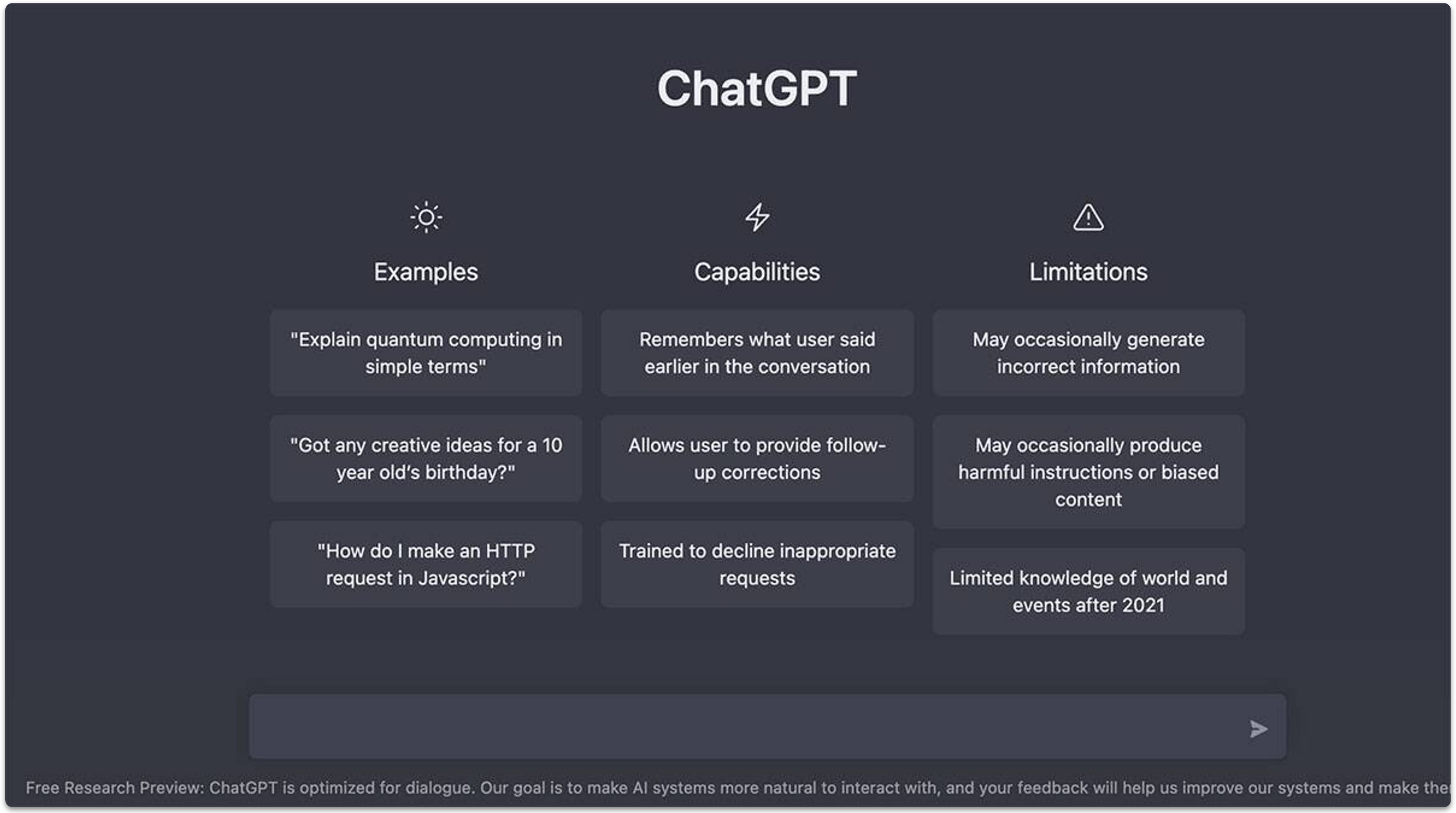
- If you like **beaches** and **sunshine**, you can fly to **Malaga** in Spain [1], which has a pleasant climate, a lively culture, and a beautiful coastline. You can enjoy the sandy beaches, the historic monuments, and the delicious tapas. Malaga is also a great base to explore other nearby attractions, such as Granada, Seville, and Cordoba.
- If you like **mountains** and **lakes**, you can fly to **Annecy** in France [4], which is a charming alpine town near the Swiss border. Annecy is known as the "Venice of France" because of its canals and bridges. You can admire the stunning scenery, visit the medieval castle, and try some local cheese and wine. Annecy is also close to Geneva, Chamonix, and Lyon, if you want to

New Bing Search Engine [Microsoft]

LLM FOR CHATBOTS

CHATBOTS POWERED BY INSTRUCTED LLM

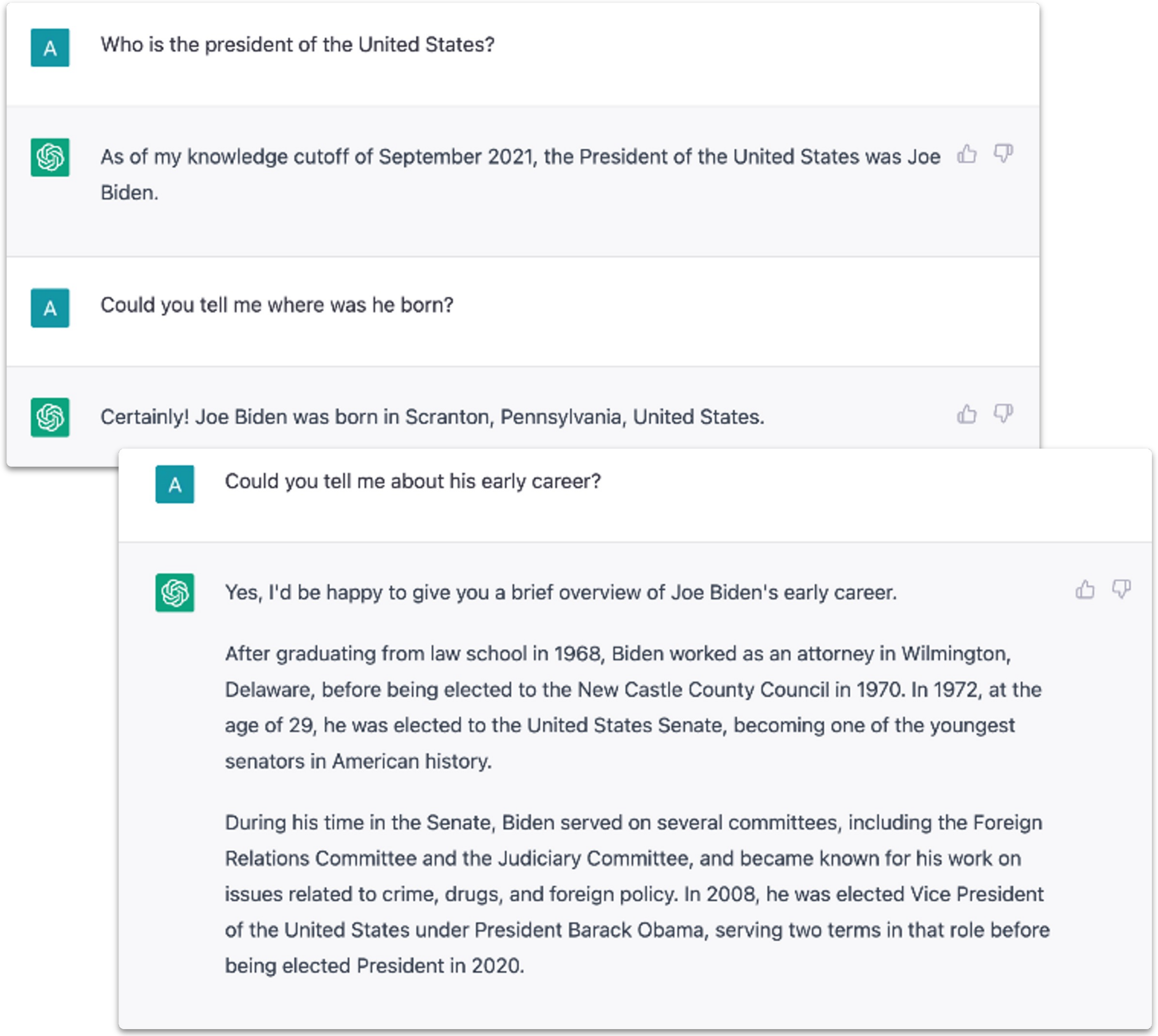
ChatGPT is a sibling model to InstructGPT



ChatGPT

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms"	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?"	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?"	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

Free Research Preview: ChatGPT is optimized for dialogue. Our goal is to make AI systems more natural to interact with, and your feedback will help us improve our systems and make the



A Who is the president of the United States?

As of my knowledge cutoff of September 2021, the President of the United States was Joe Biden.

A Could you tell me where was he born?

Certainly! Joe Biden was born in Scranton, Pennsylvania, United States.

A Could you tell me about his early career?

Yes, I'd be happy to give you a brief overview of Joe Biden's early career.

After graduating from law school in 1968, Biden worked as an attorney in Wilmington, Delaware, before being elected to the New Castle County Council in 1970. In 1972, at the age of 29, he was elected to the United States Senate, becoming one of the youngest senators in American history.

During his time in the Senate, Biden served on several committees, including the Foreign Relations Committee and the Judiciary Committee, and became known for his work on issues related to crime, drugs, and foreign policy. In 2008, he was elected Vice President of the United States under President Barack Obama, serving two terms in that role before being elected President in 2020.

CHATBOTS POWERED BY INSTRUCTED LLM

Bing Chat powered by Next generation GPT | Bard powered by LaMDA

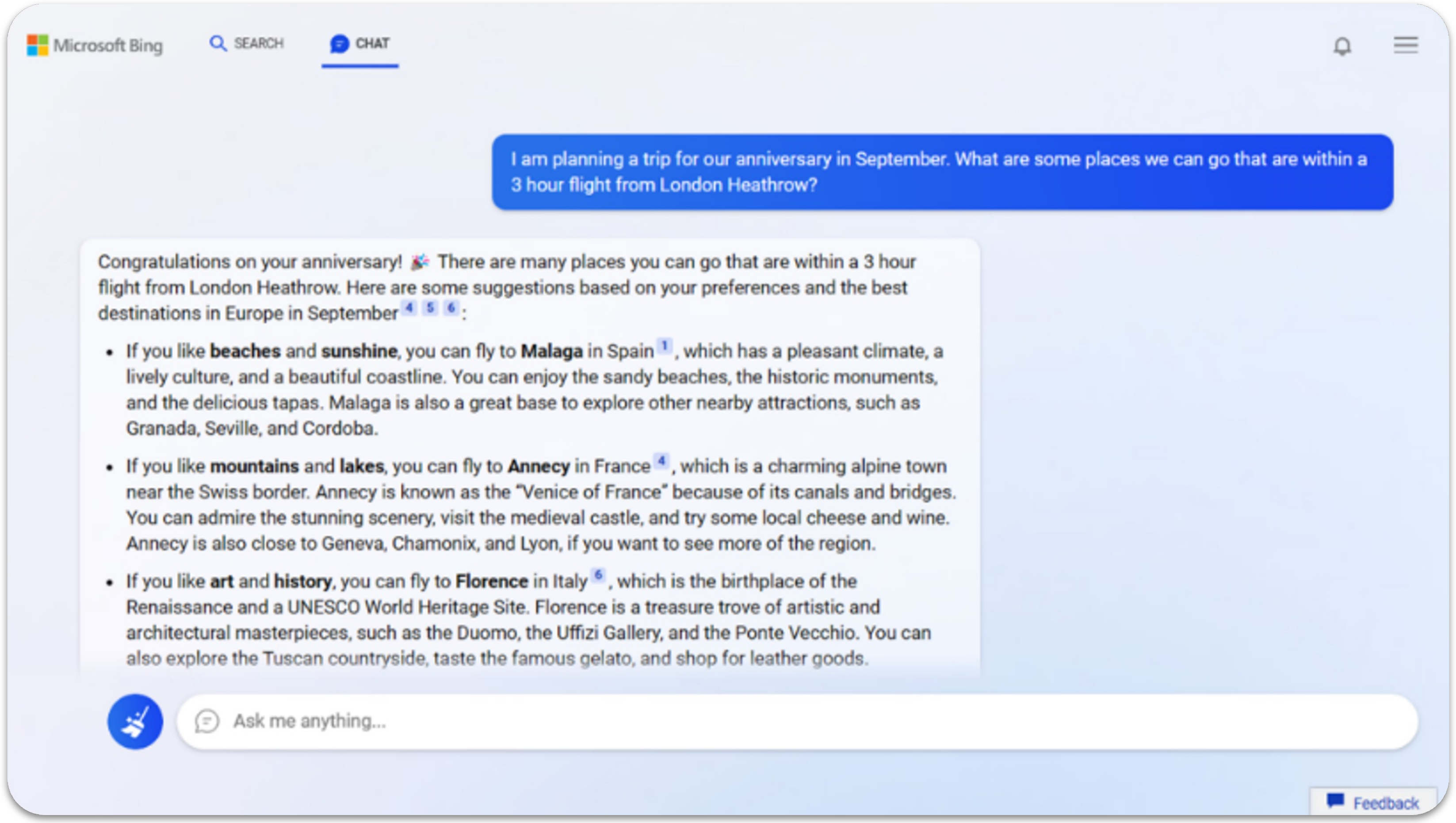
OpenAI

Next-generation large language model

More powerful than ChatGPT

Customized for search

Prometheus Model

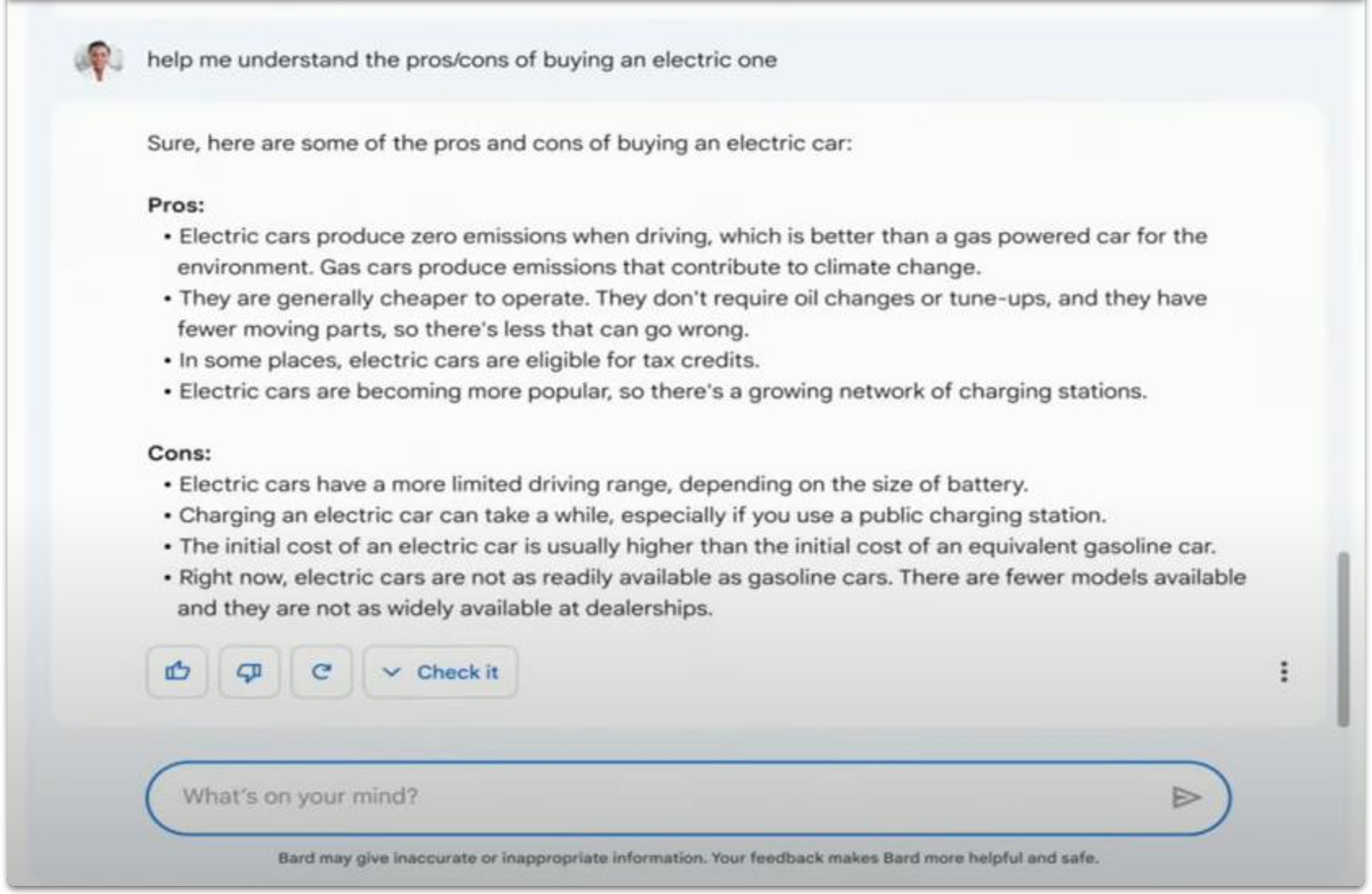


I am planning a trip for our anniversary in September. What are some places we can go that are within a 3 hour flight from London Heathrow?

Congratulations on your anniversary! 🎉 There are many places you can go that are within a 3 hour flight from London Heathrow. Here are some suggestions based on your preferences and the best destinations in Europe in September [4] [5] [6]:

- If you like **beaches** and **sunshine**, you can fly to **Malaga** in Spain [1], which has a pleasant climate, a lively culture, and a beautiful coastline. You can enjoy the sandy beaches, the historic monuments, and the delicious tapas. Malaga is also a great base to explore other nearby attractions, such as Granada, Seville, and Cordoba.
- If you like **mountains** and **lakes**, you can fly to **Annecy** in France [4], which is a charming alpine town near the Swiss border. Annecy is known as the "Venice of France" because of its canals and bridges. You can admire the stunning scenery, visit the medieval castle, and try some local cheese and wine. Annecy is also close to Geneva, Chamonix, and Lyon, if you want to see more of the region.
- If you like **art** and **history**, you can fly to **Florence** in Italy [6], which is the birthplace of the Renaissance and a UNESCO World Heritage Site. Florence is a treasure trove of artistic and architectural masterpieces, such as the Duomo, the Uffizi Gallery, and the Ponte Vecchio. You can also explore the Tuscan countryside, taste the famous gelato, and shop for leather goods.

Introducing Bard, an experimental conversational AI service powered by LaMDA



help me understand the pros/cons of buying an electric one

Sure, here are some of the pros and cons of buying an electric car:

Pros:

- Electric cars produce zero emissions when driving, which is better than a gas powered car for the environment. Gas cars produce emissions that contribute to climate change.
- They are generally cheaper to operate. They don't require oil changes or tune-ups, and they have fewer moving parts, so there's less that can go wrong.
- In some places, electric cars are eligible for tax credits.
- Electric cars are becoming more popular, so there's a growing network of charging stations.

Cons:

- Electric cars have a more limited driving range, depending on the size of battery.
- Charging an electric car can take a while, especially if you use a public charging station.
- The initial cost of an electric car is usually higher than the initial cost of an equivalent gasoline car.
- Right now, electric cars are not as readily available as gasoline cars. There are fewer models available and they are not as widely available at dealerships.

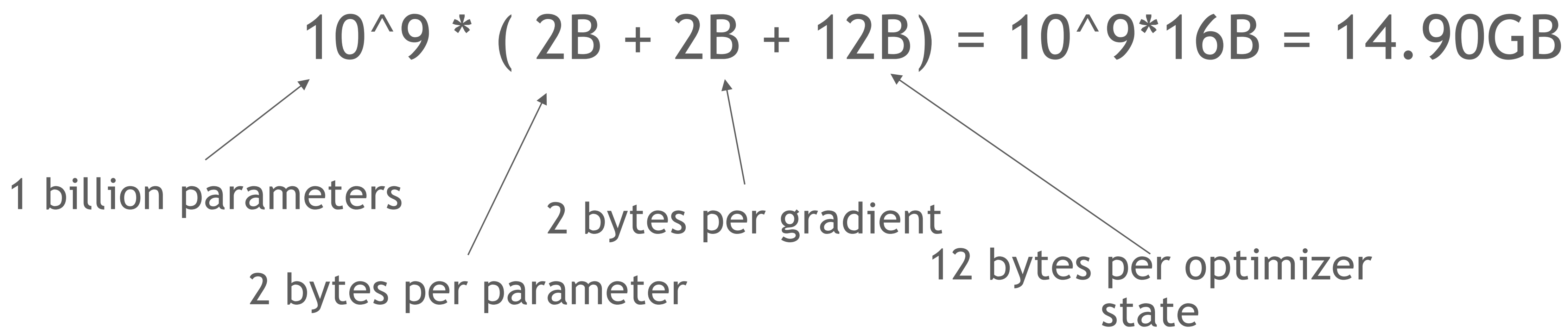
Bard may give inaccurate or inappropriate information. Your feedback makes Bard more helpful and safe.

TECHNICALLY, CAN WE BUILD LARGER MODELS?

GOING BIGGER

The challenge

- If we only consider Parameters, Gradients, and Optimizer states and ignore activations
- If we use FP16 data representation (so two bytes)
- If we use Adam as an optimizer (storing twelve bytes per parameter in mixed precision mode)
- If we consider a model with <u>one billion</u> parameters



10^9 * (2B + 2B + 12B) = 10^9*16B = 14.90GB

1 billion parameters

2 bytes per parameter

2 bytes per gradient

12 bytes per optimizer state

GOING BIGGER

The challenge

- What about activations?
- What about 2 or 3 billion parameter models?

10^9 * (2B + 2B + 12B) = 10^9*16B = 14.90GB

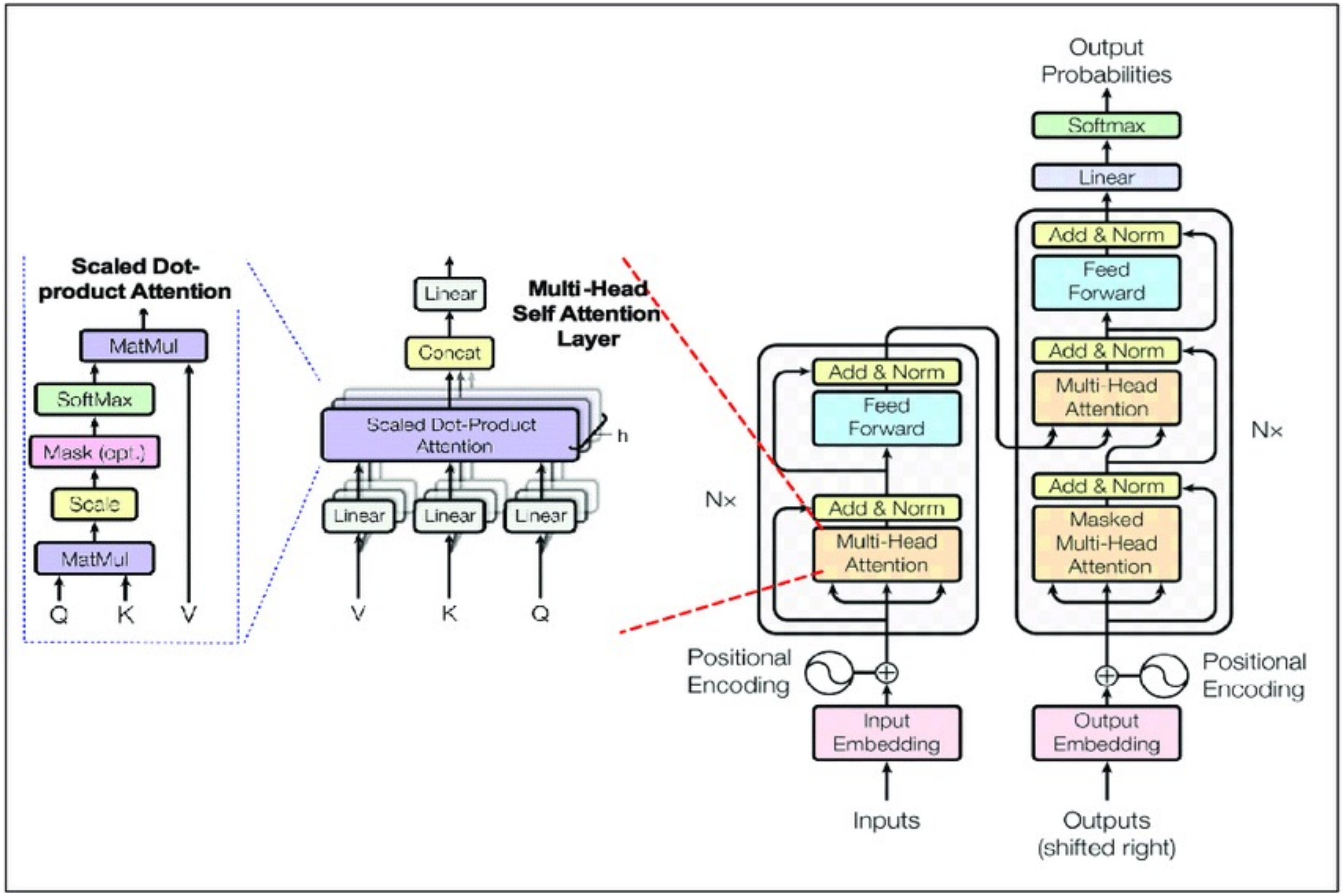
1 billion parameters

2 bytes per parameter

2 bytes per gradient

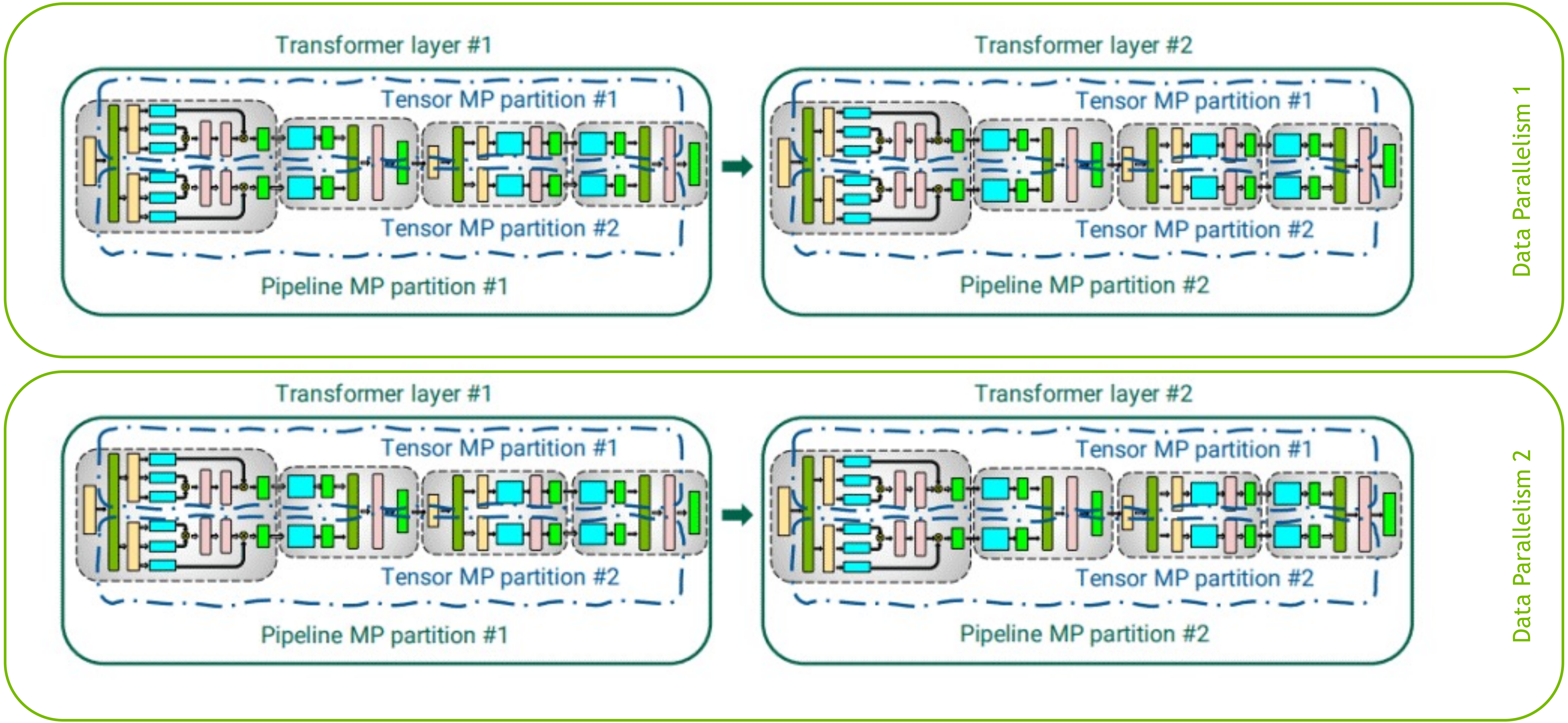
12 bytes per optimizer state

TRANSFORMER MODELS



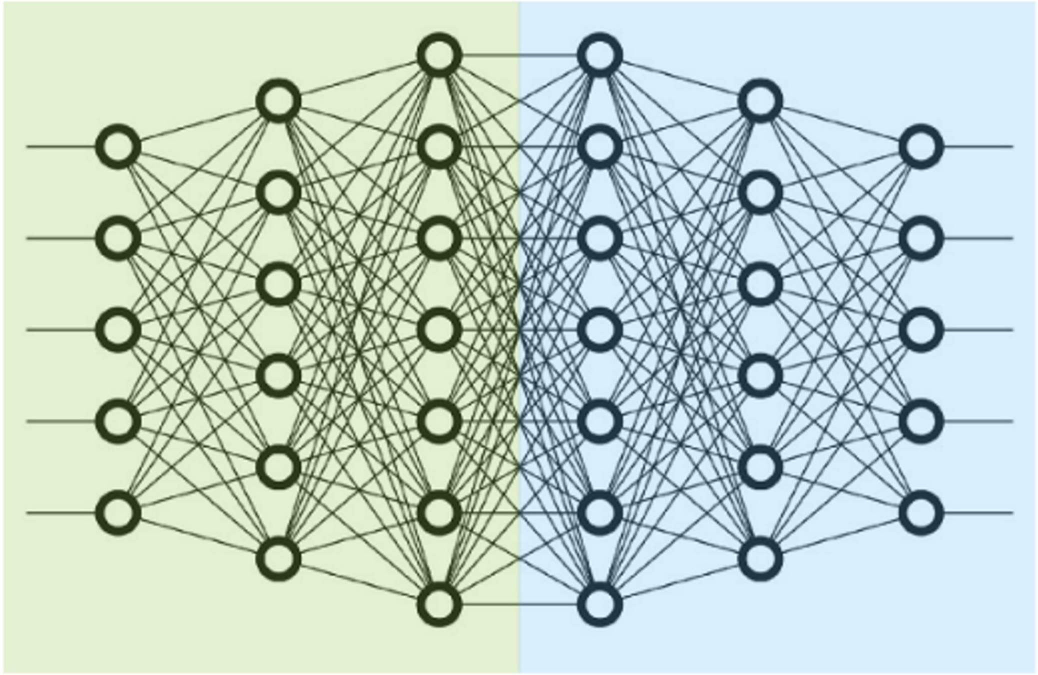
MODEL IMPLEMENTATION

Data, Pipeline and Tensor Parallelism

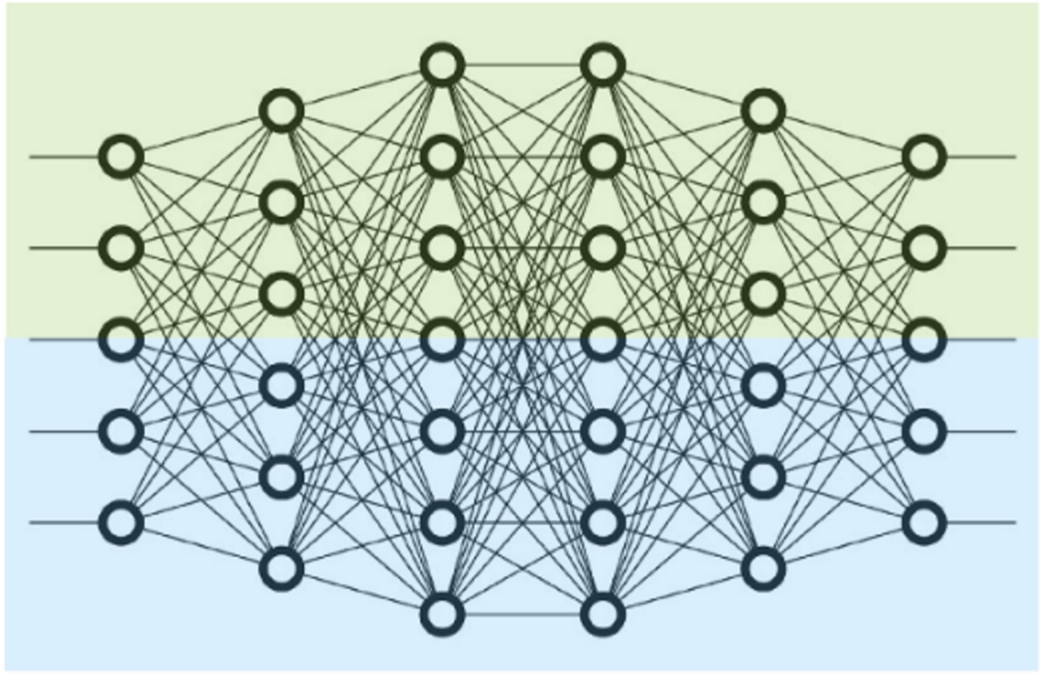


MODEL PARALLELISM

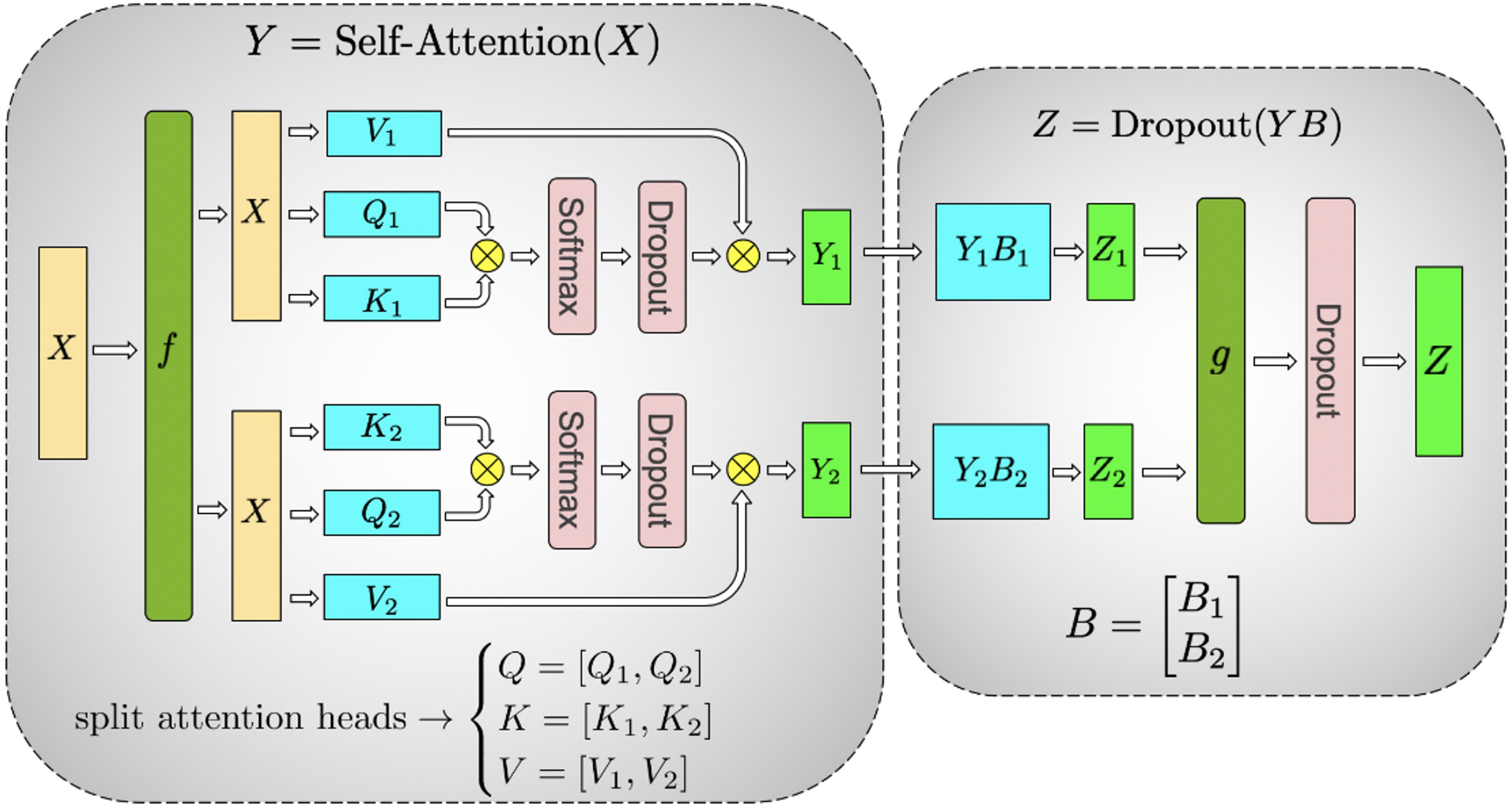
- Pipeline (Inter-Layer) Parallelism
  - Split sets of layers across multiple devices
  - Layer 0,1,2 and layer 3,4,5 are on difference devices



- Tensor (Intra-Layer) Parallelism
  - Split individual layers across multiple devices
  - Both devices compute difference parts of Layer 0,1,2,3,4,5

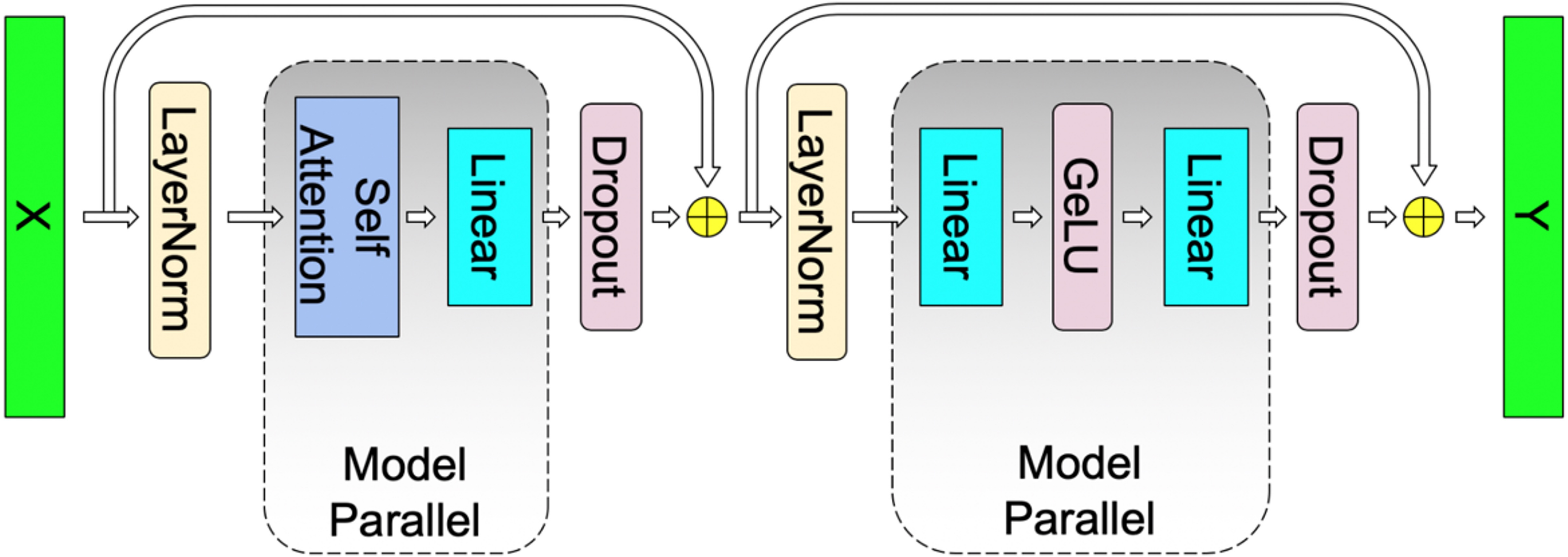


SELF-ATTENTION



f and *g* are **conjugate**, *f* is **identity** operator in the forward pass and **all-reduce** in the backward pass while *g* is **all-reduce** in forward and **identity** in backward.

PARALLEL TRANSFORMER LAYER

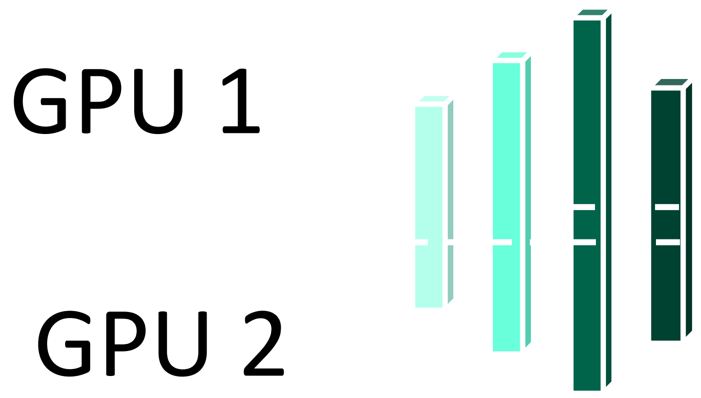


2 All-Reduce
(forward + backward)

2 All-Reduce
(forward + backward)

COMPARING TENSOR AND PIPELINE PARALLELISM

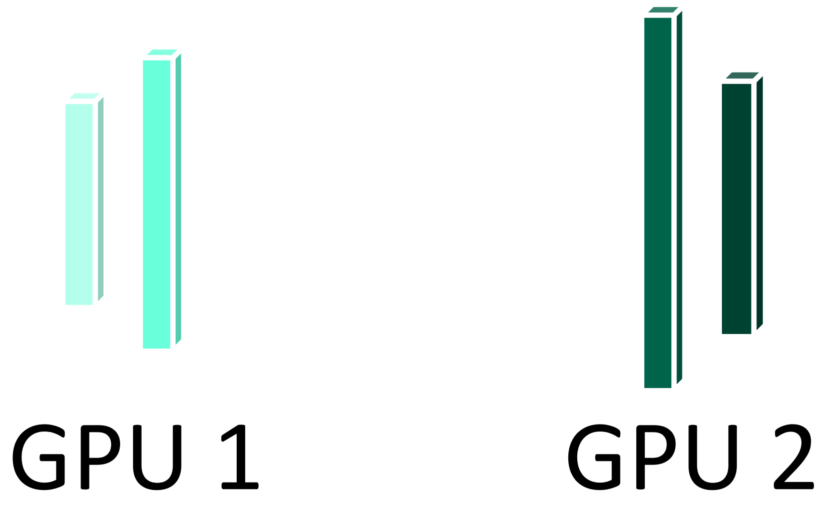
Tensor Parallelism



Communication expensive

Good performance across batch sizes

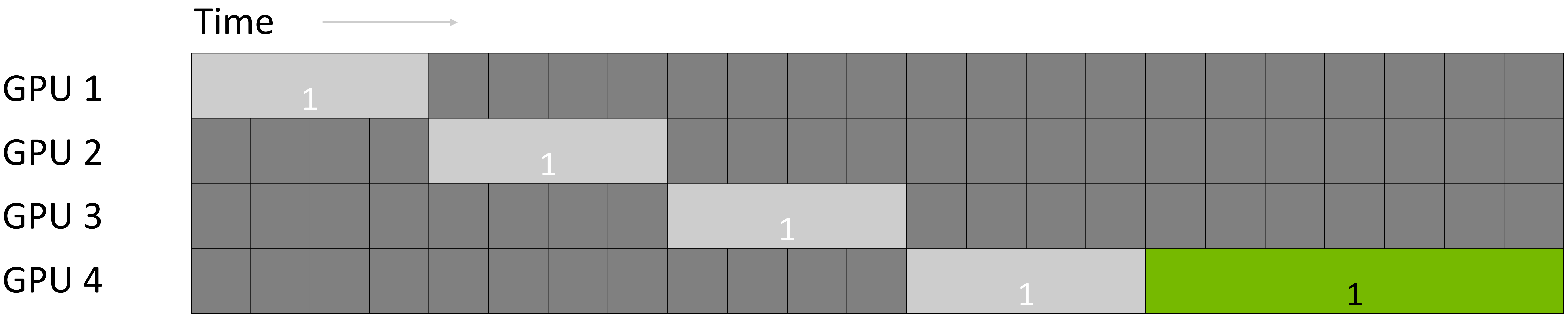
Pipeline Parallelism



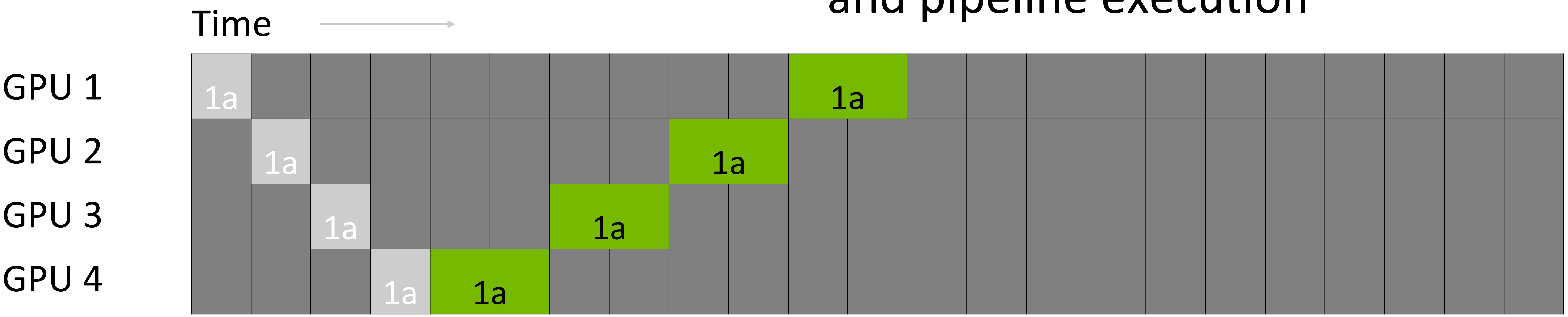
Communication cheap

Good performance at larger batch sizes (pipeline stall amortized)

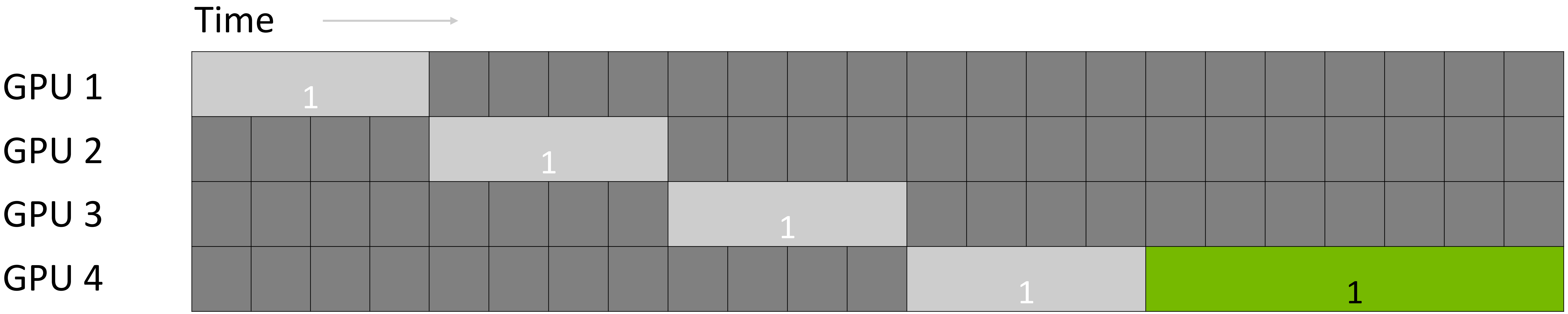
PIPELINING



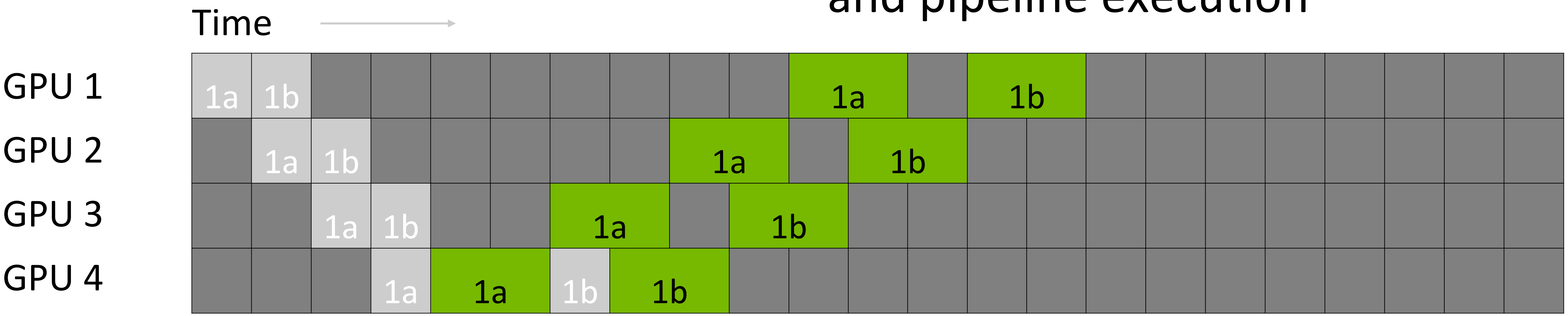
Split batch into microbatches and pipeline execution



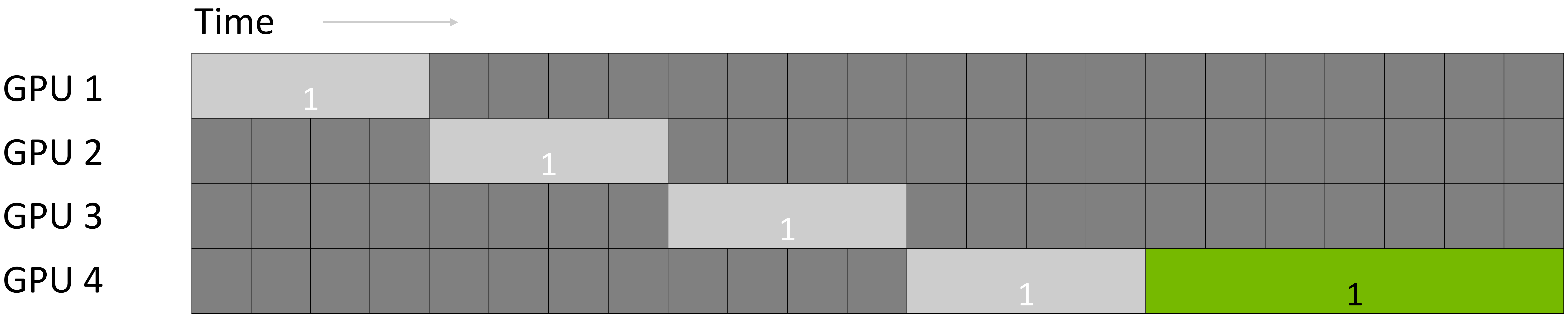
PIPELINING



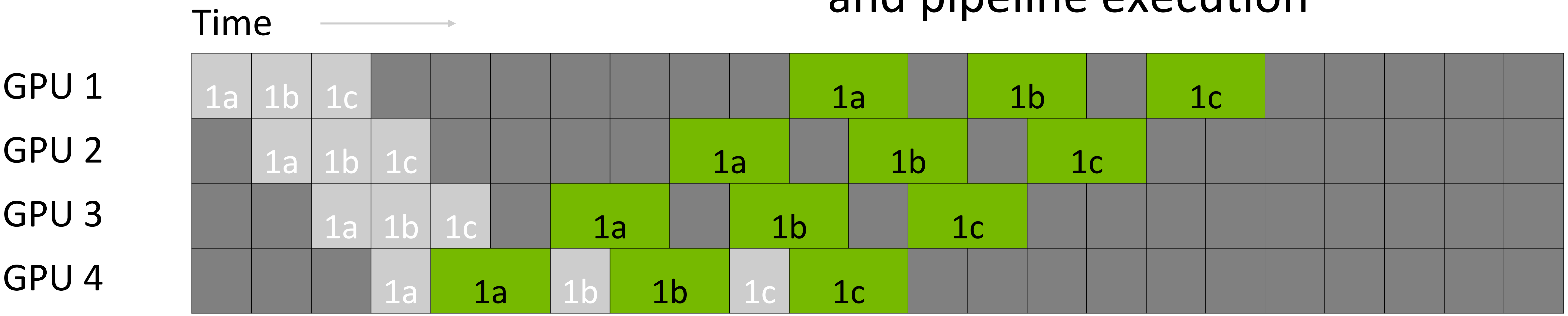
Split batch into microbatches and pipeline execution



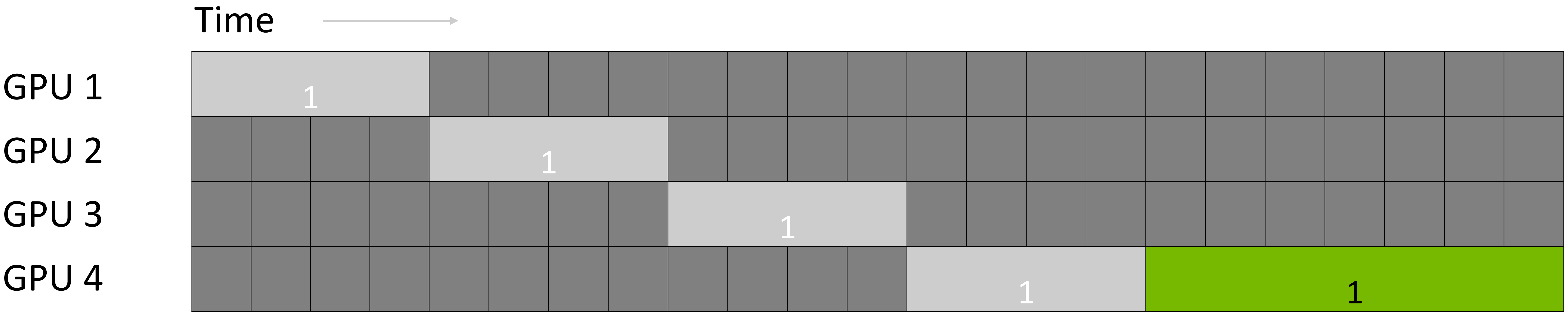
PIPELINING



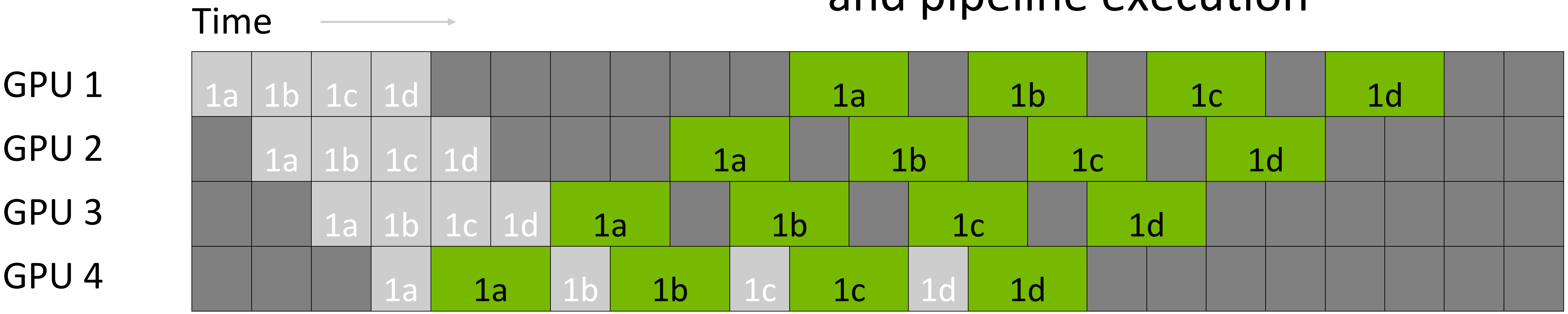
Split batch into microbatches and pipeline execution



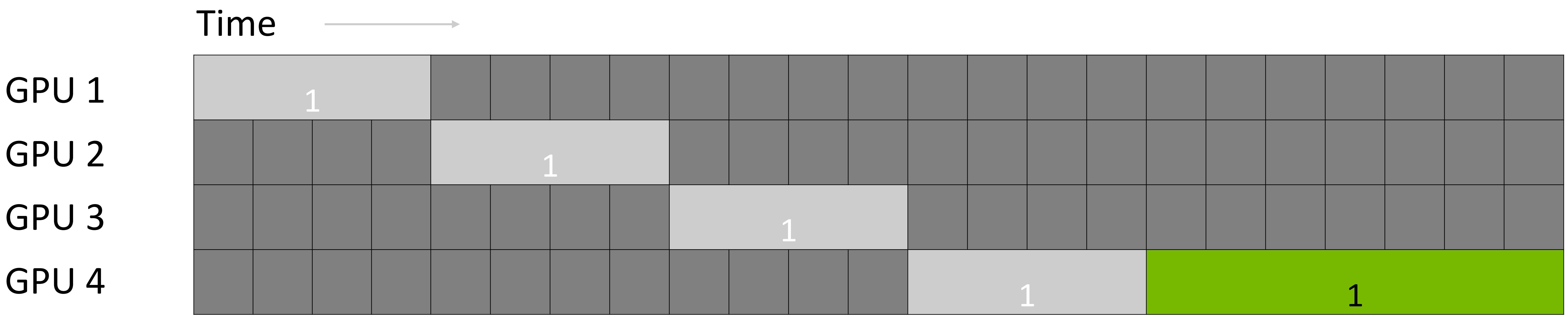
PIPELINING



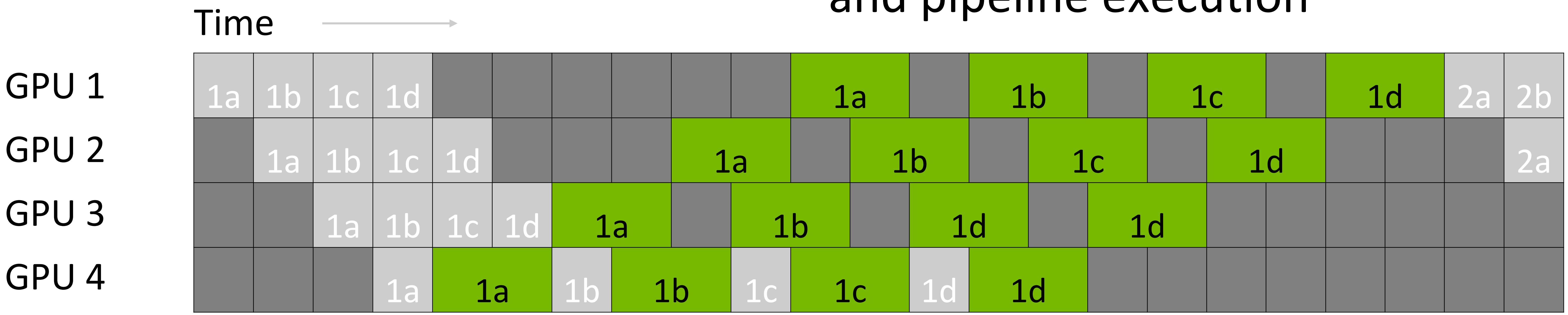
Split batch into microbatches and pipeline execution



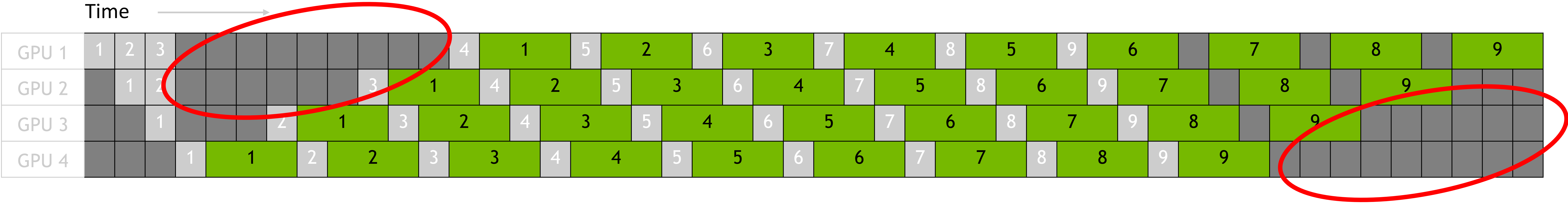
PIPELINING



Split batch into microbatches and pipeline execution



PIPELINE BUBBLES



p : number of pipeline stages

m : number of micro batches

t_f : forward step time

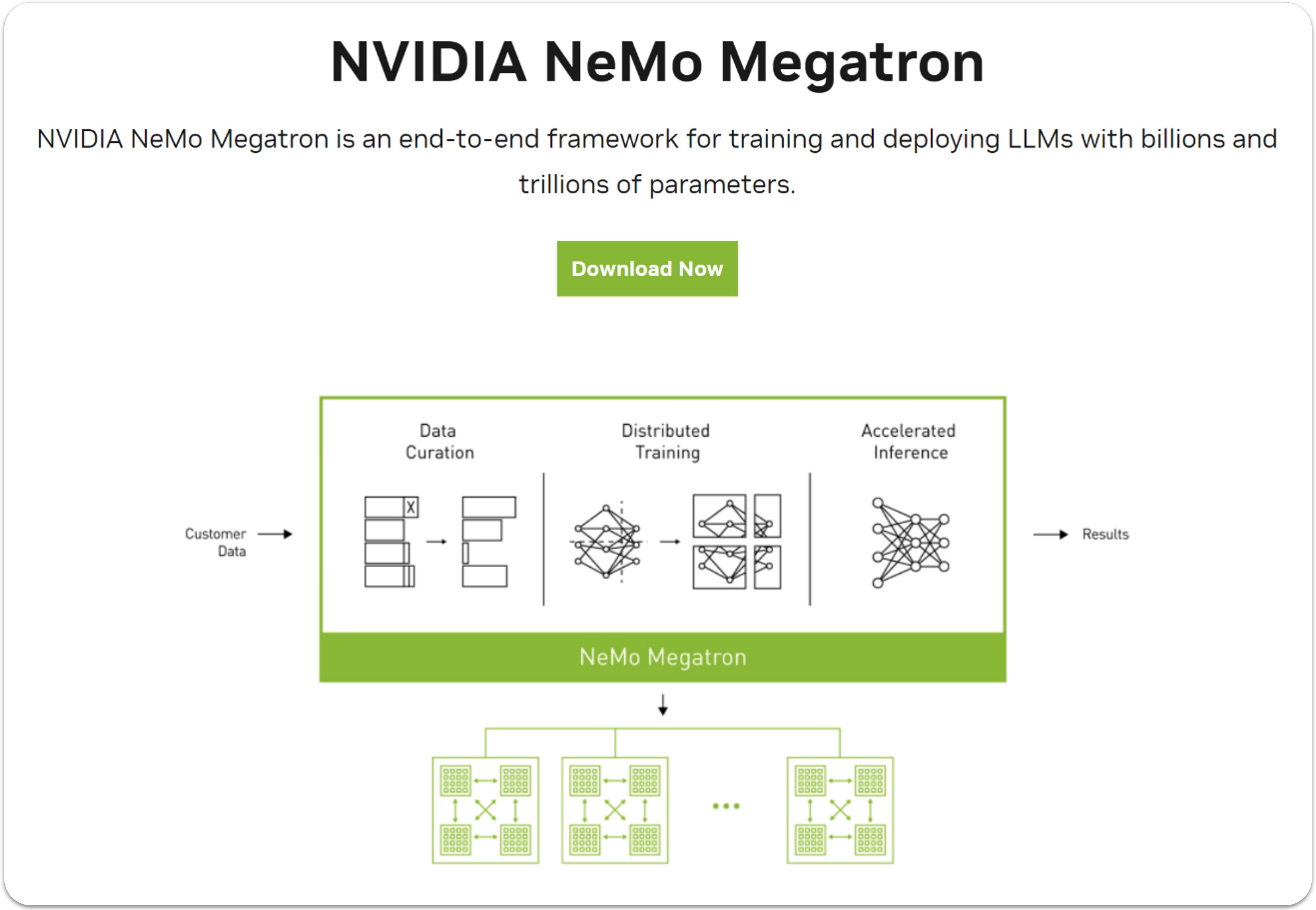
t_b : backward step time

$$\text{total time} = (m + p - 1) \times (t_f + t_b)$$

$$\text{ideal time} = m \times (t_f + t_b)$$

$$\text{bubble time} = (p - 1) \times (t_f + t_b)$$

$$\text{bubble time overhead} = \frac{\text{bubble time}}{\text{ideal time}} = \frac{p - 1}{m}$$



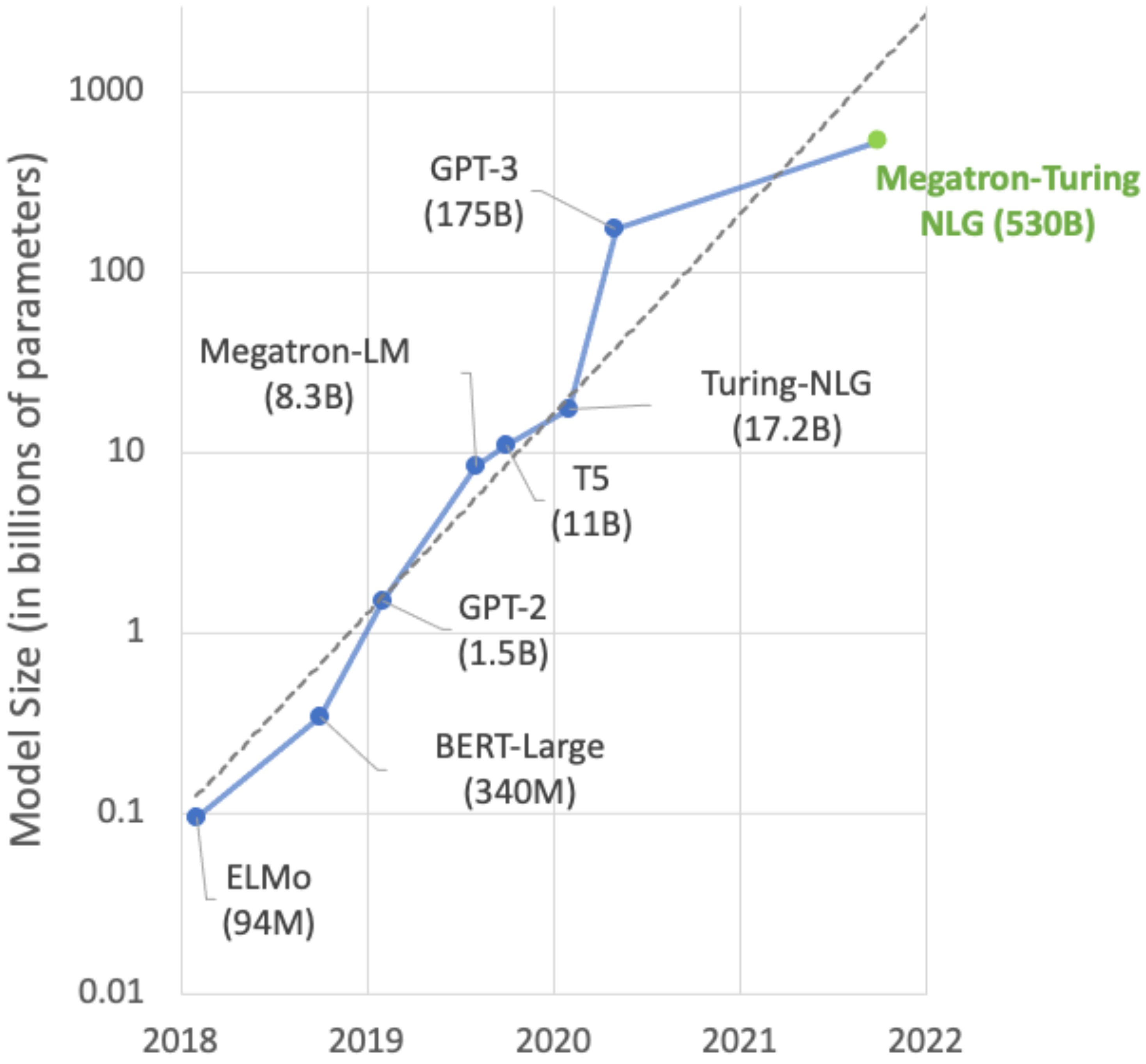
NVIDIA NeMo Megatron

NVIDIA NeMo Megatron is an end-to-end framework for training and deploying LLMs with billions and trillions of parameters.

https://developer.nvidia.com/nemo/megatron

MEGATRON-TURING NLG 530B

Enabling the biggest of NLP models



https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

THE LAB

Part 2: Self-Supervision, BERT and Beyond

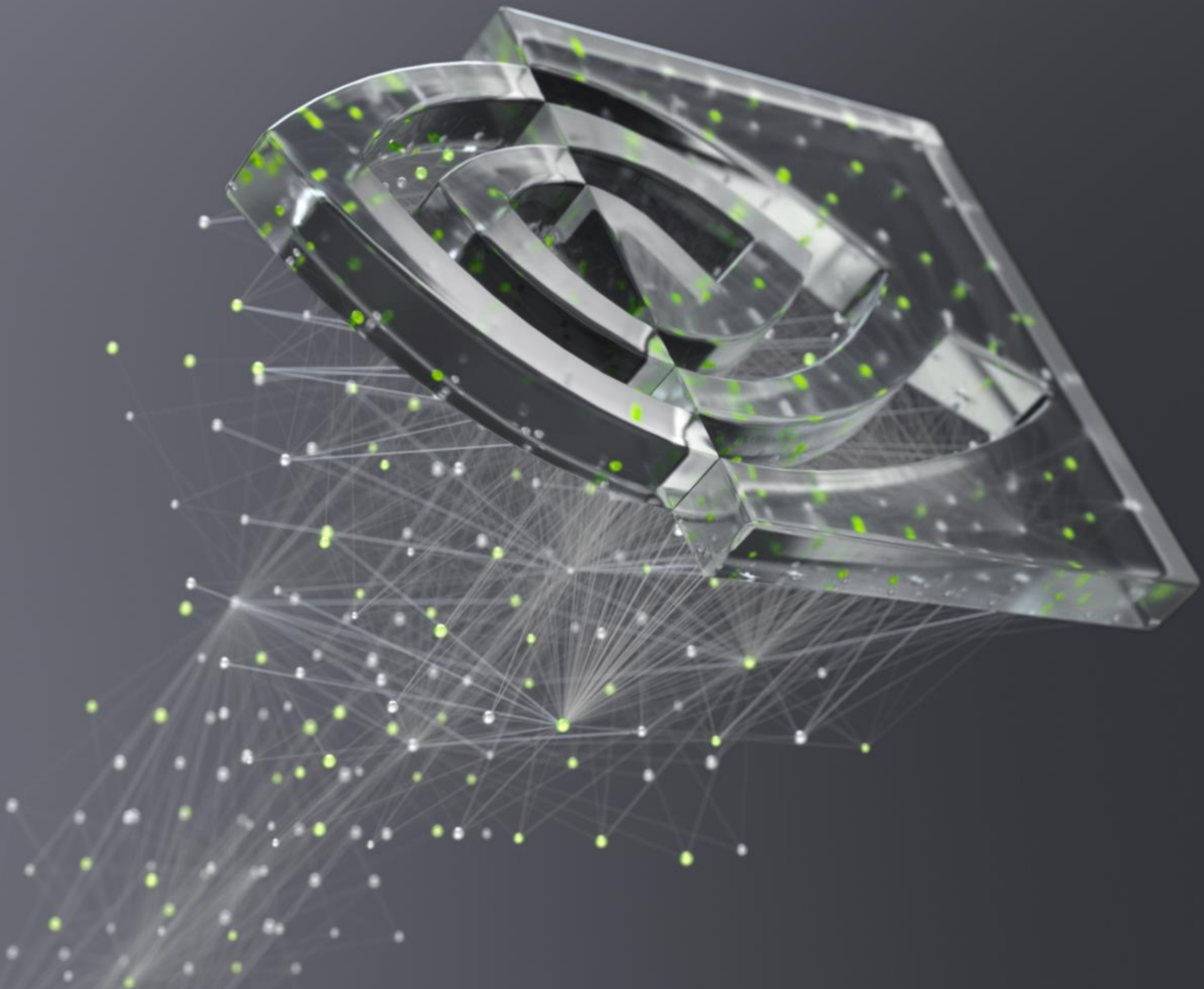
- Lecture
  - Why DNNs?
  - Self-Supervision
  - BERT
- Lab
  - Explore the Data
  - Explore NeMo
  - Text Classifier Project
- Lecture (cont’d)
  - The Scaling Laws
  - Can and should we go even bigger?
- Lab (cont’d)
  - Named Entity Recognizer

IN THE NEXT CLASS…

NEXT CLASS

Overview

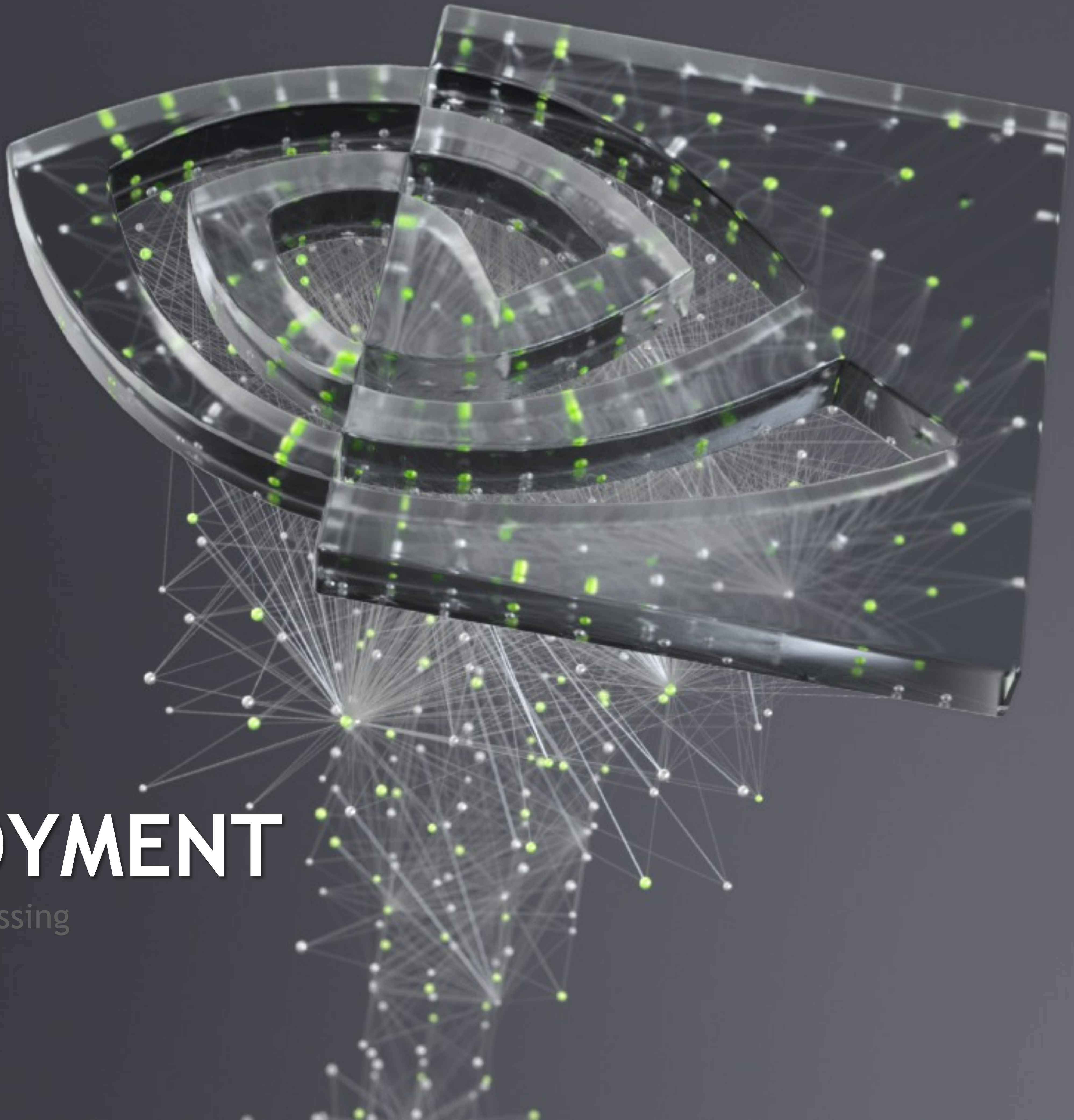
1. Discuss how to design your model for efficient inference
2. Discuss how to optimise your model for efficient execution
3. Discuss how to efficiently host a largely Conversational AI application



NVIDIA

DEEP
LEARNING
INSTITUTE

DEEP
LEARNING
INSTITUTE



PRODUCTION DEPLOYMENT

Building Transformer-Based Natural Language Processing Applications
(Part 3)

FULL COURSE AGENDA

Part 1: Machine Learning in NLP

Lecture: NLP background and the role of DNNs leading to the Transformer architecture

Lab: Tutorial-style exploration of a ***translation task*** using the Transformer architecture

Part 2: Self-Supervision, BERT, and Beyond

Lecture: Discussion of how language models with self-supervision have moved beyond the basic Transformer to BERT and ever larger models

Lab: Practical hands-on guide to the NVIDIA NeMo API and exercises to build a ***text classification task*** and a ***named entity recognition task*** using BERT-based language models

Part 3: Production Deployment

Lecture: Discussion of production deployment considerations and NVIDIA Triton Inference Server

Lab: Hands-on deployment of an example ***question answering task*** to NVIDIA Triton

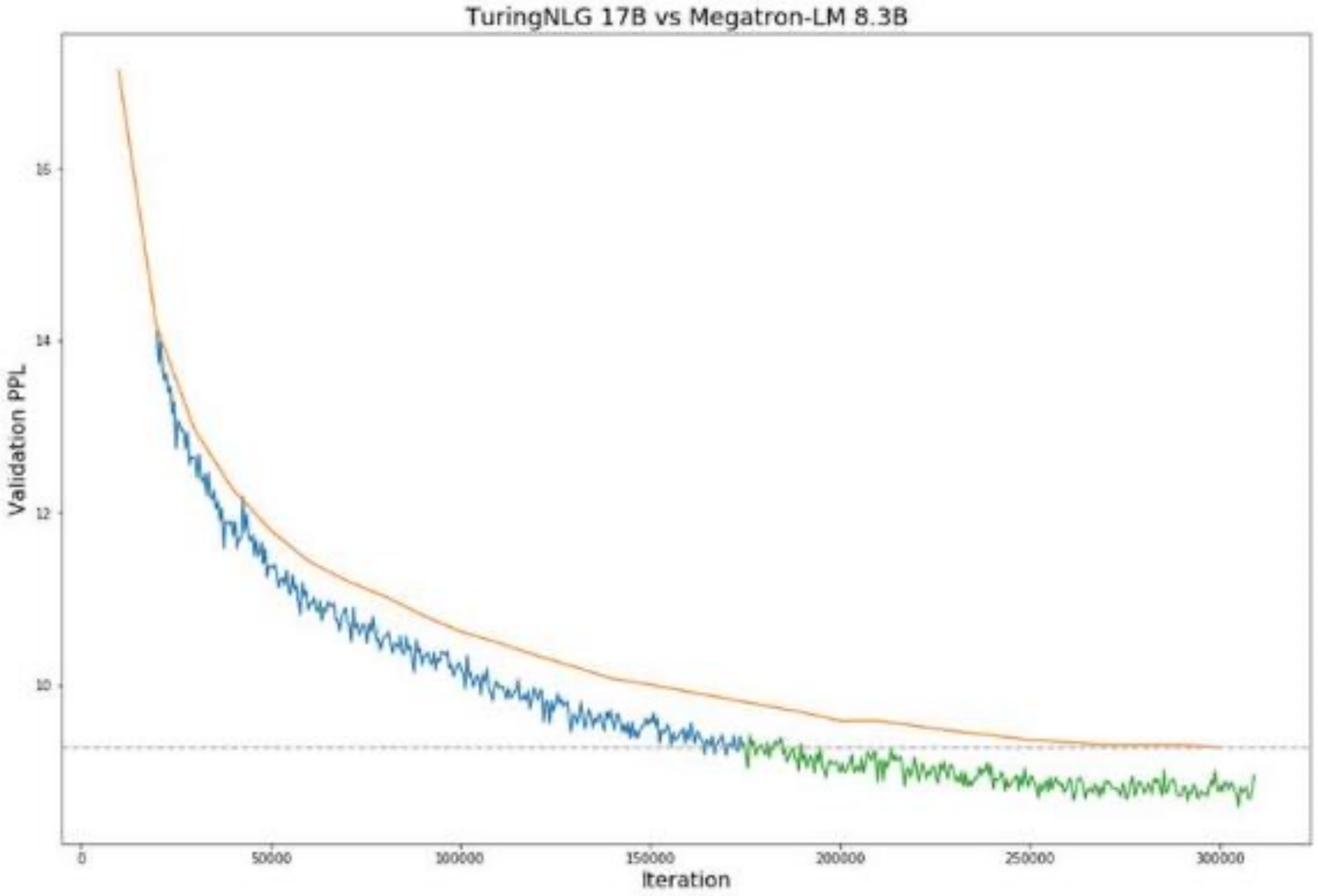
Part 3: Production Deployment

- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model

YOUR NETWORK IS TRAINED

YOUR NETWORK IS TRAINED

Now what?

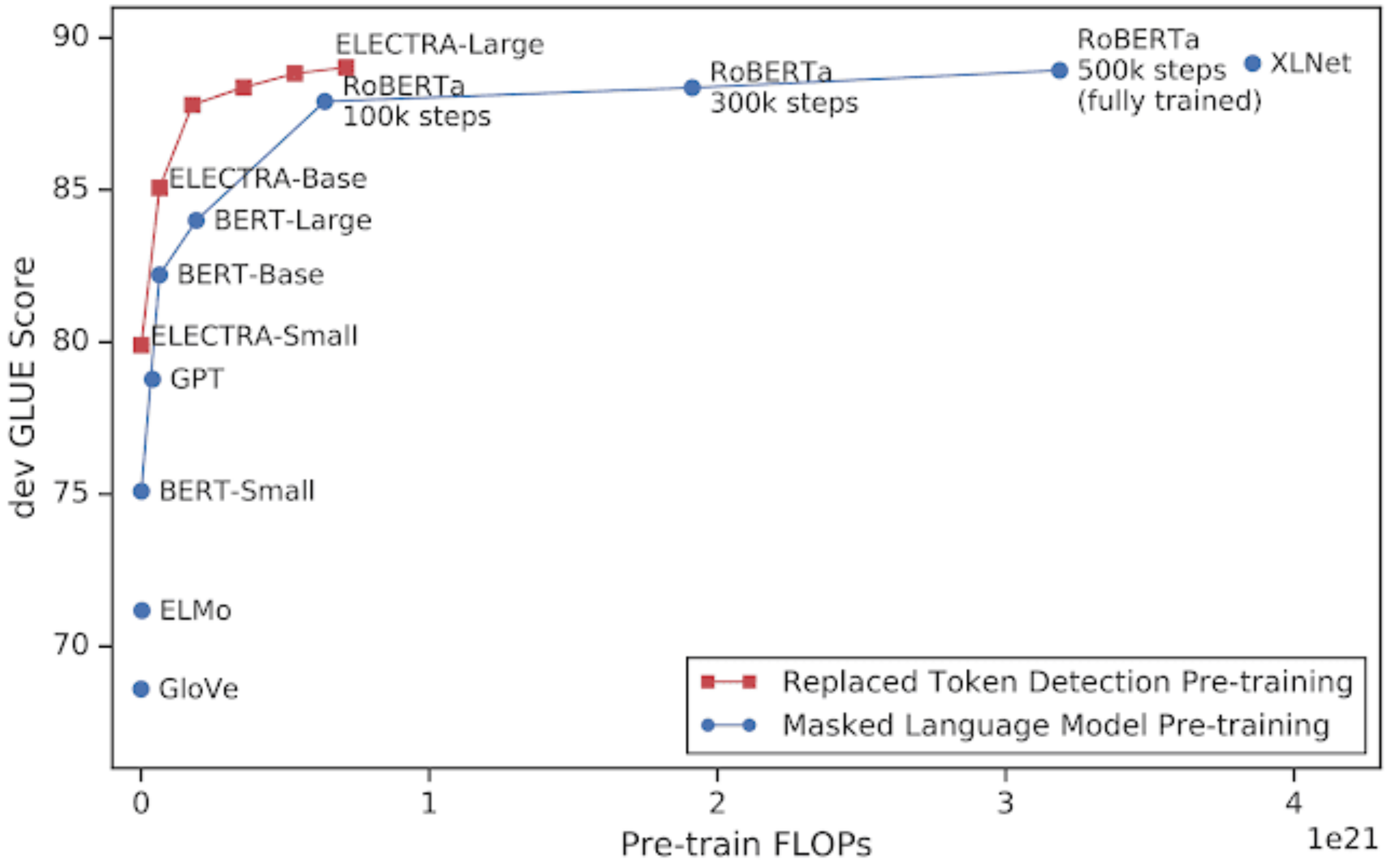


https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

MEETING REQUIREMENTS OF YOUR BUSINESS

NLP MODELS ARE LARGE

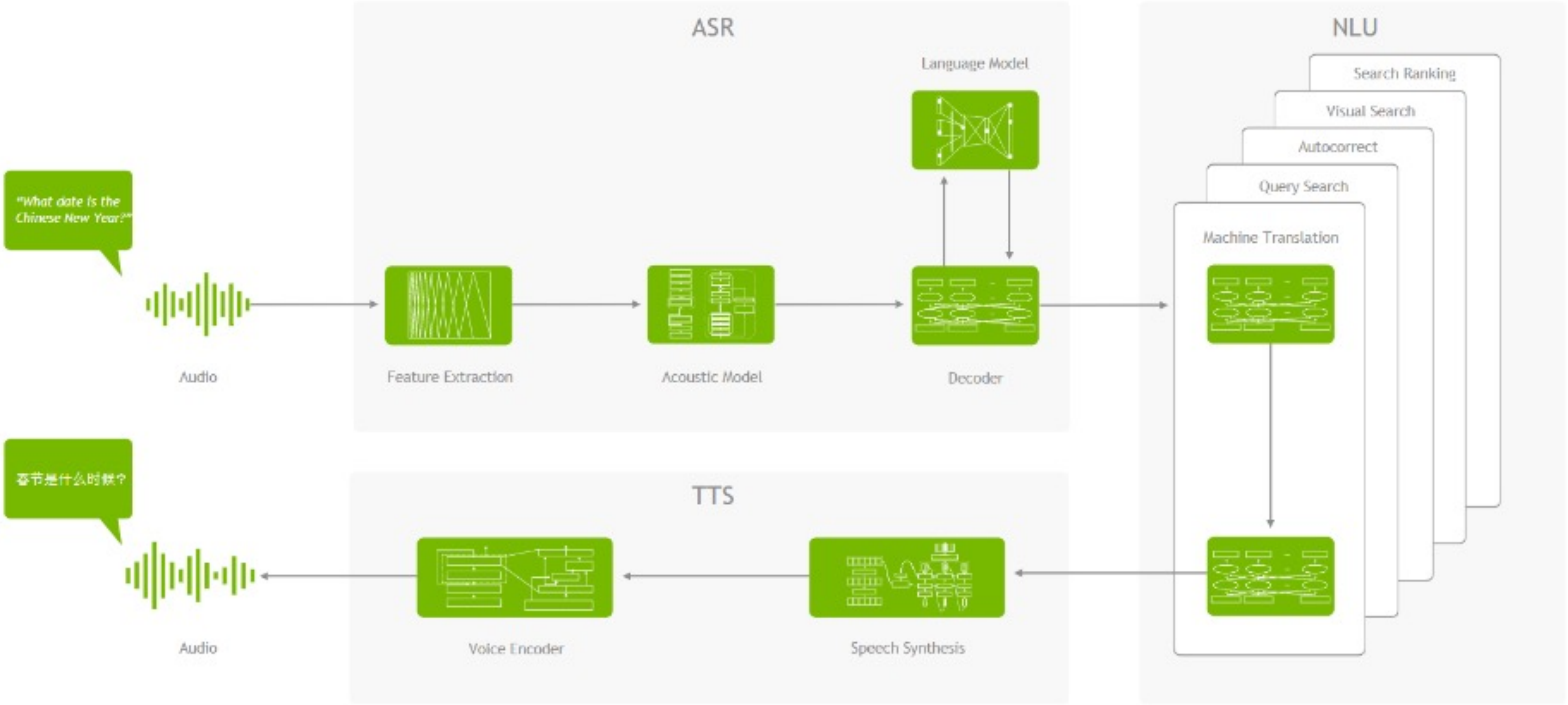
The Inference cost is high



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

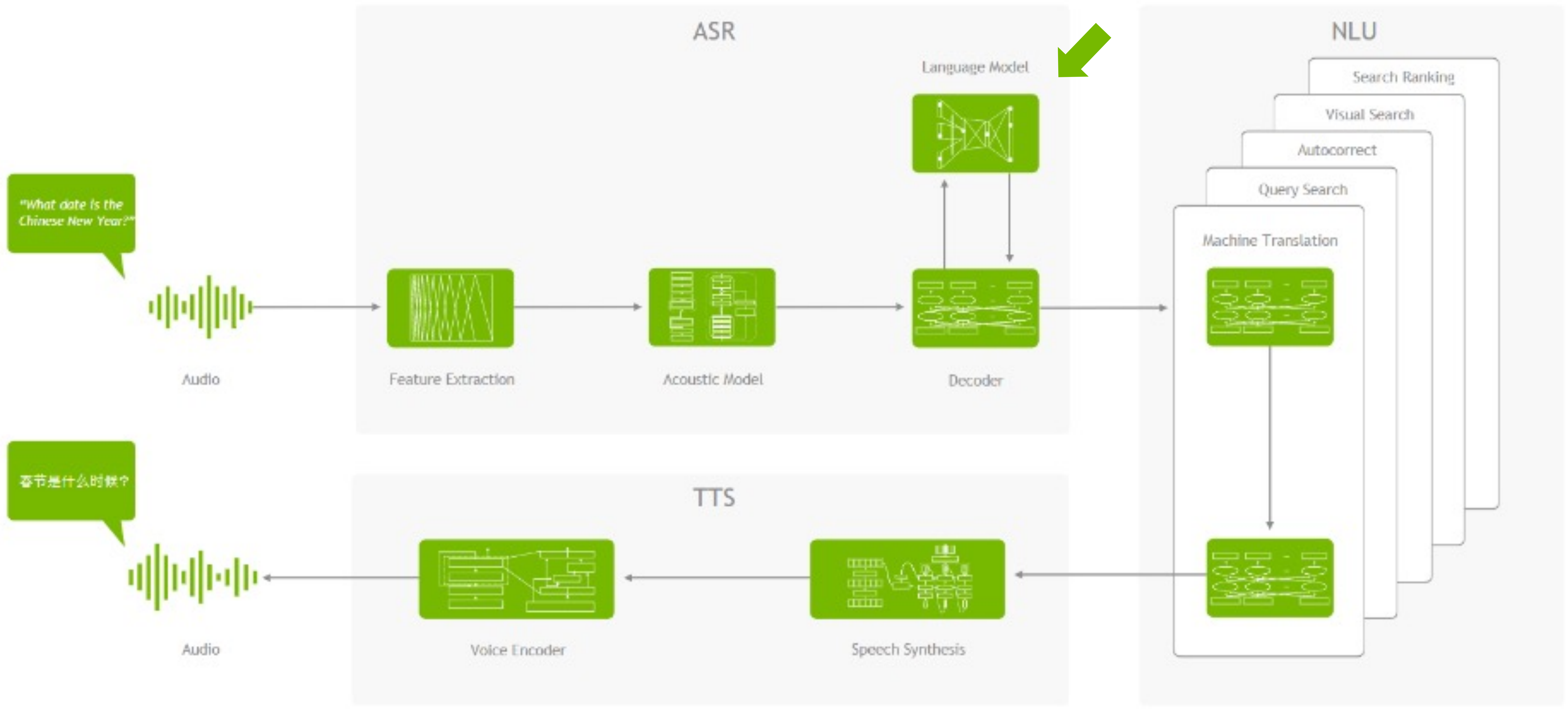
THEY DO NOT LIVE IN ISOLATION

Example of a conversational AI application



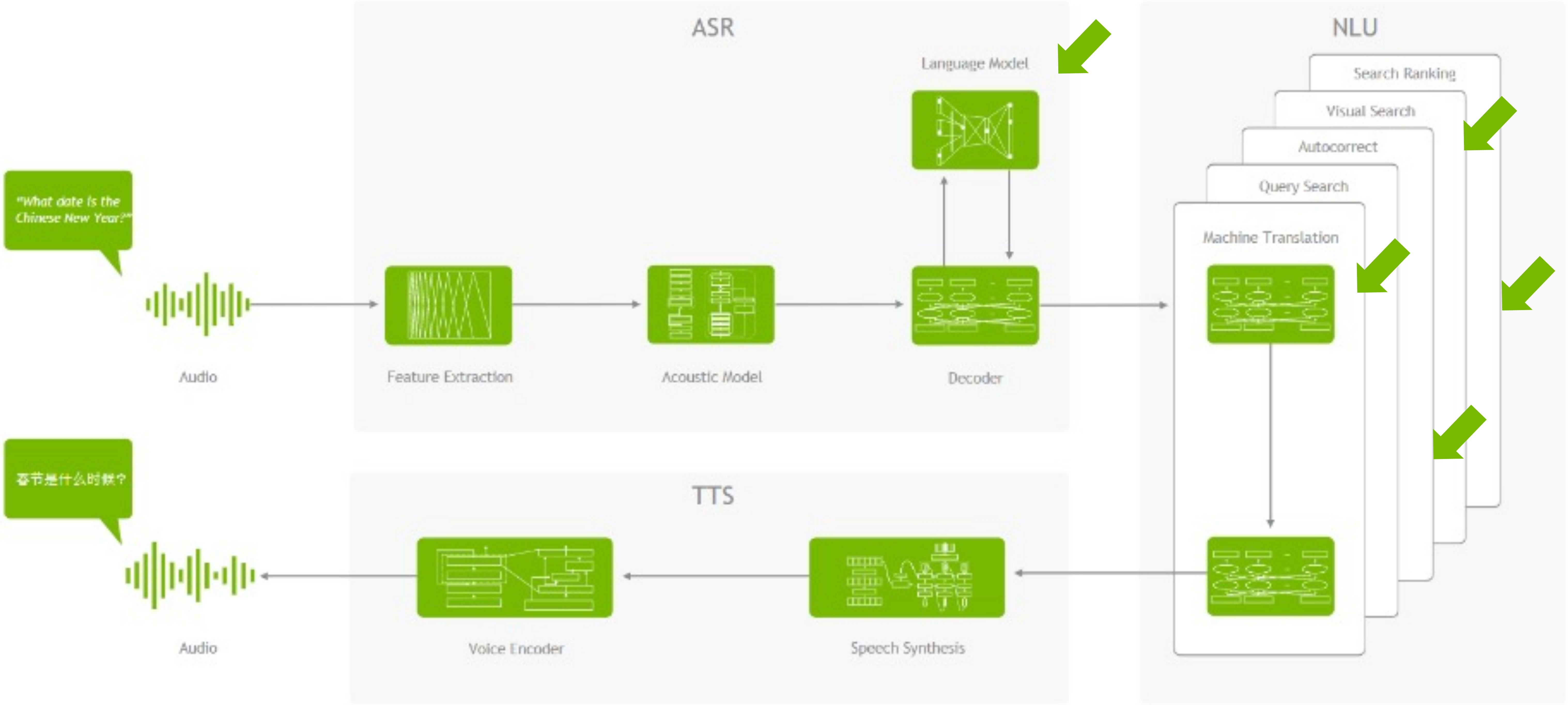
THEY DO NOT LIVE IN ISOLATION

Real Time Applications Need to Deliver Latency <300 ms



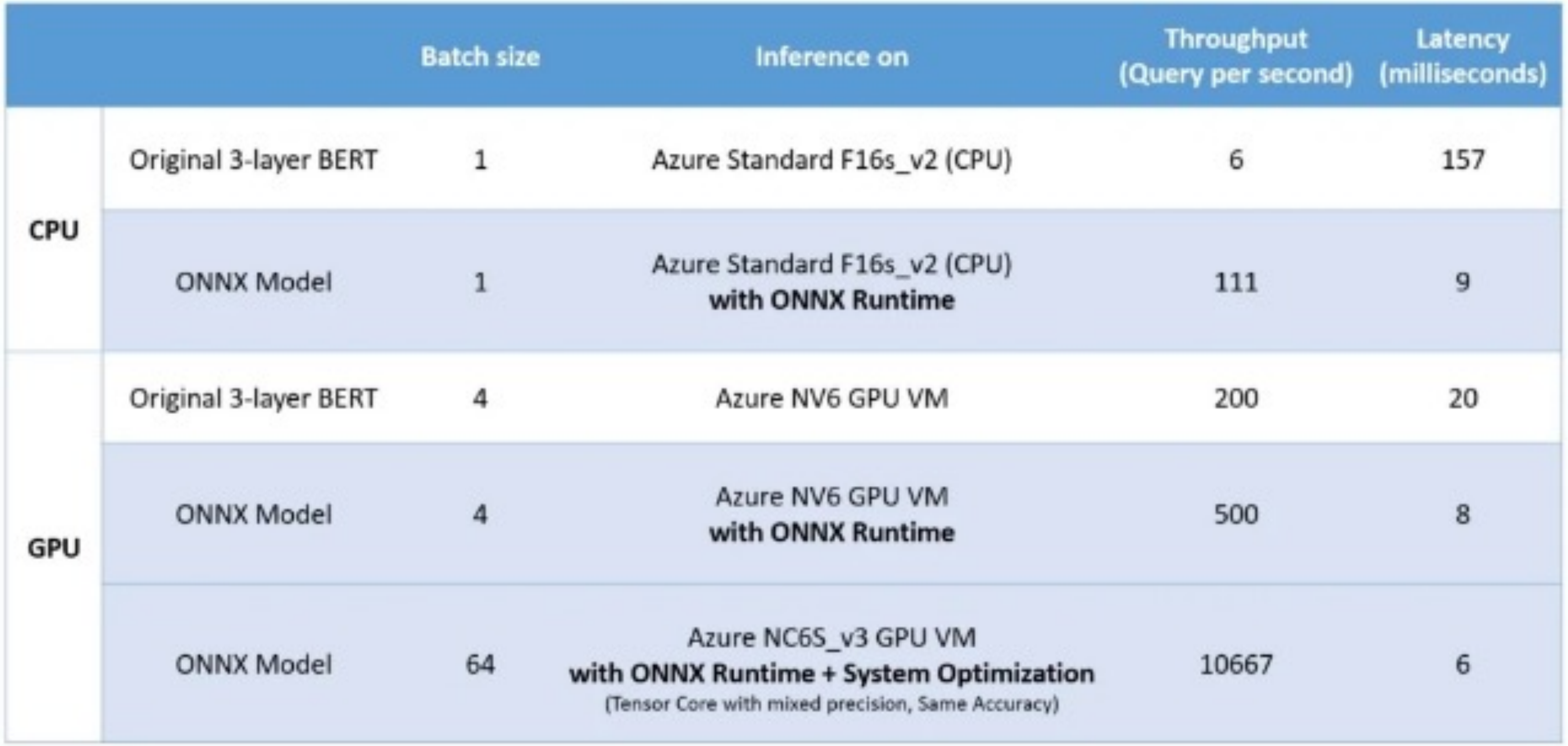
THEY DO NOT LIVE IN ISOLATION

Real Time Applications Need to Deliver Latency <300 ms



THEY DO NOT LIVE IN ISOLATION

Application bandwidth = Cost

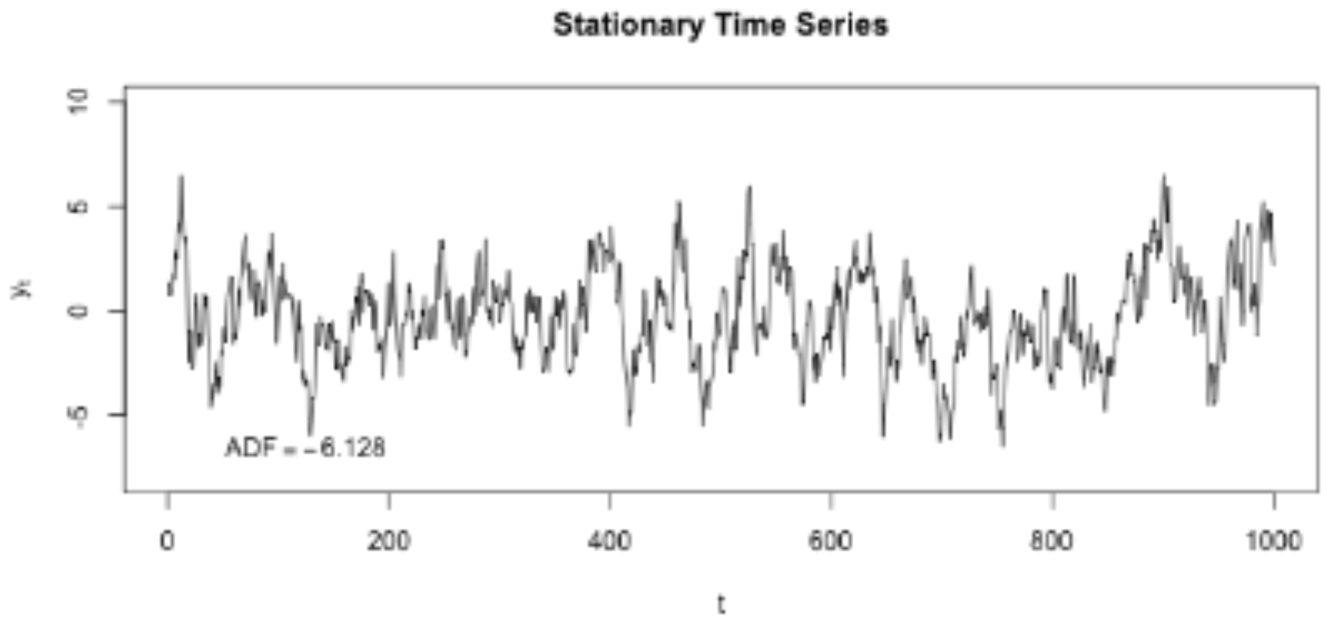


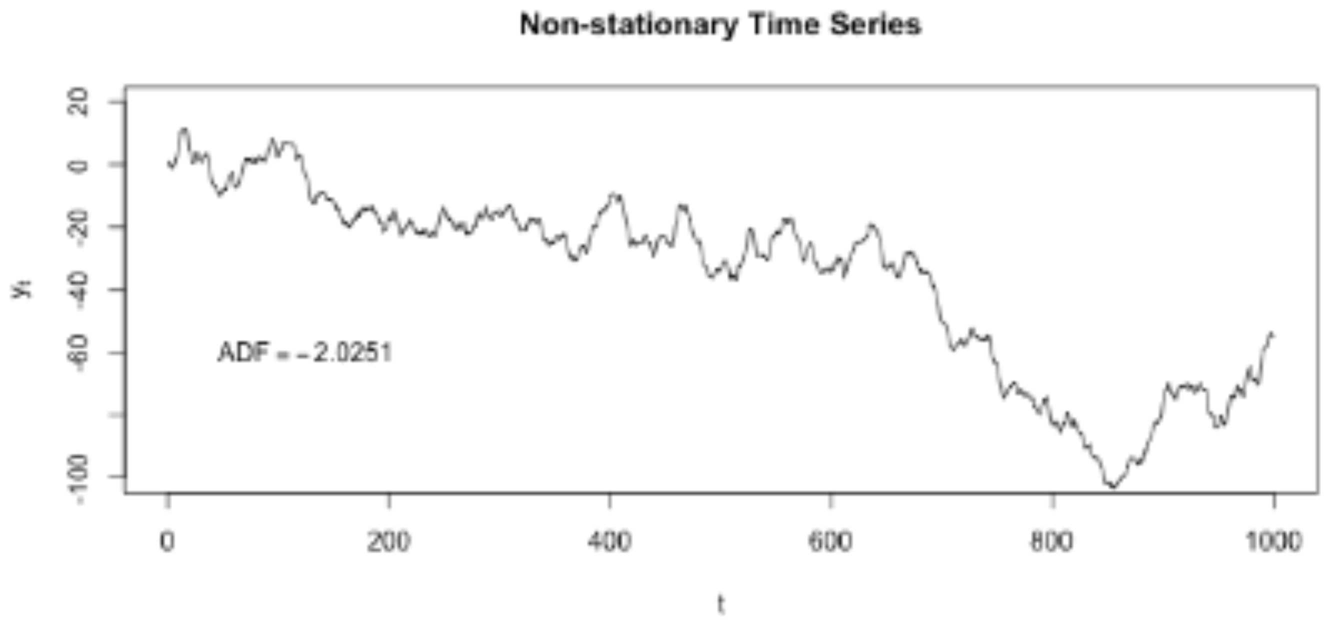
		Batch size	Inference on	Throughput (Query per second)	Latency (milliseconds)
CPU	Original 3-layer BERT	1	Azure Standard F16s_v2 (CPU)	6	157
	ONNX Model	1	Azure Standard F16s_v2 (CPU) **with ONNX Runtime**	111	9
GPU	Original 3-layer BERT	4	Azure NV6 GPU VM	200	20
	ONNX Model	4	Azure NV6 GPU VM **with ONNX Runtime**	500	8
	ONNX Model	64	Azure NC6S_v3 GPU VM **with ONNX Runtime + System Optimization** (Tensor Core with mixed precision, Same Accuracy)	10667	6

https://cloudblogs.microsoft.com/opensource/2020/01/21/microsoft-onnx-open-source-optimizations-transformer-inference-gpu-cpu/

AND THEY NEED TO EVOLVE OVER TIME

A lot of processes are not stationary





https://en.wikipedia.org/wiki/Stationary_process

THERE'S MORE TO AN APPLICATION THAN JUST THE MODEL

Nonfunctional requirements

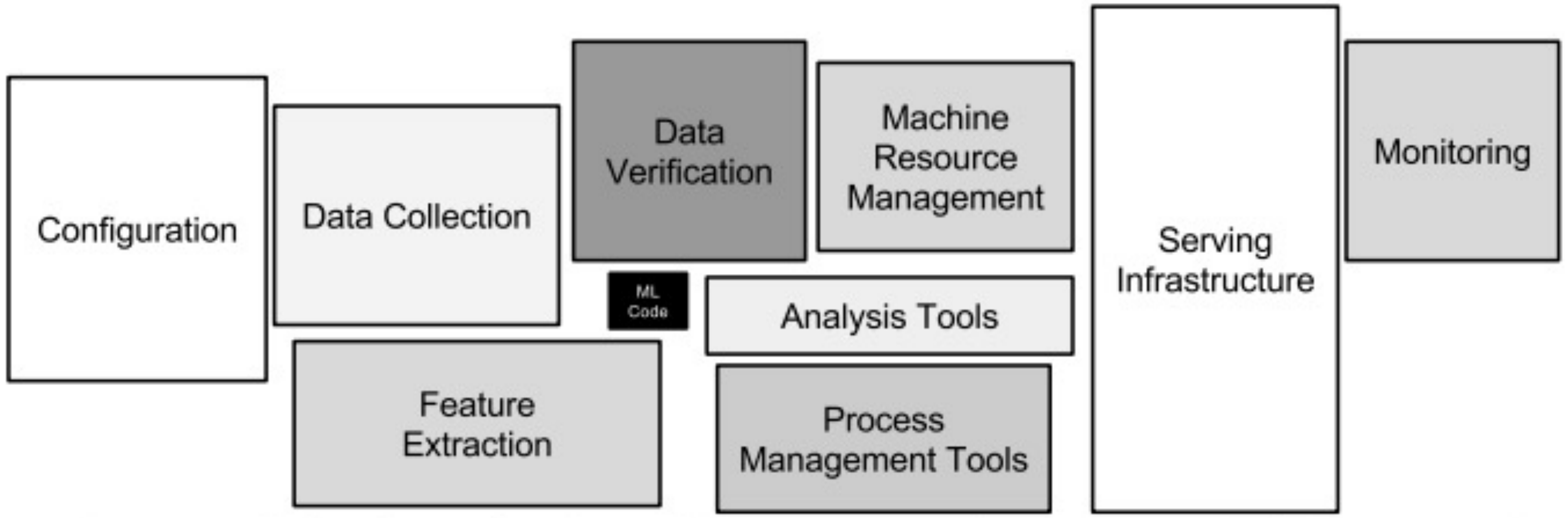


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. In Advances in neural information processing systems (pp. 2503-2511).

THERE'S MORE TO AN APPLICATION THAN JUST THE MODEL

Nonfunctional requirements

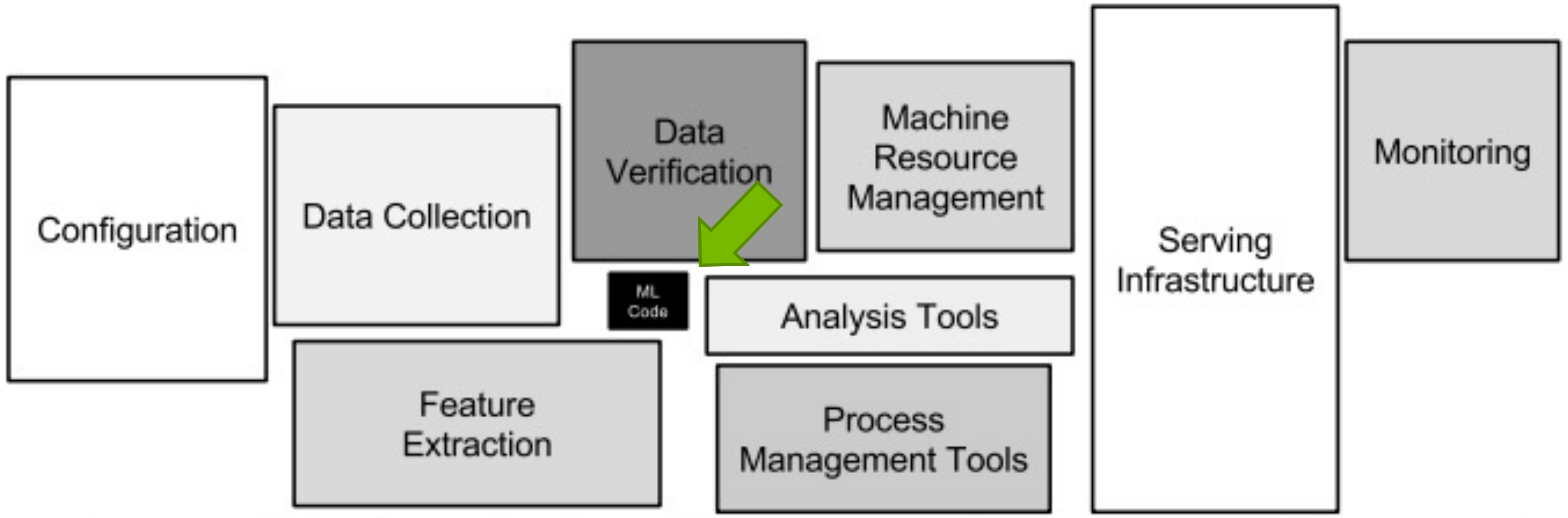


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. In Advances in neural information processing systems (pp. 2503-2511).

Part 3: Production Deployment

- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model

MODEL SELECTION

Not all models are created equally

NLP

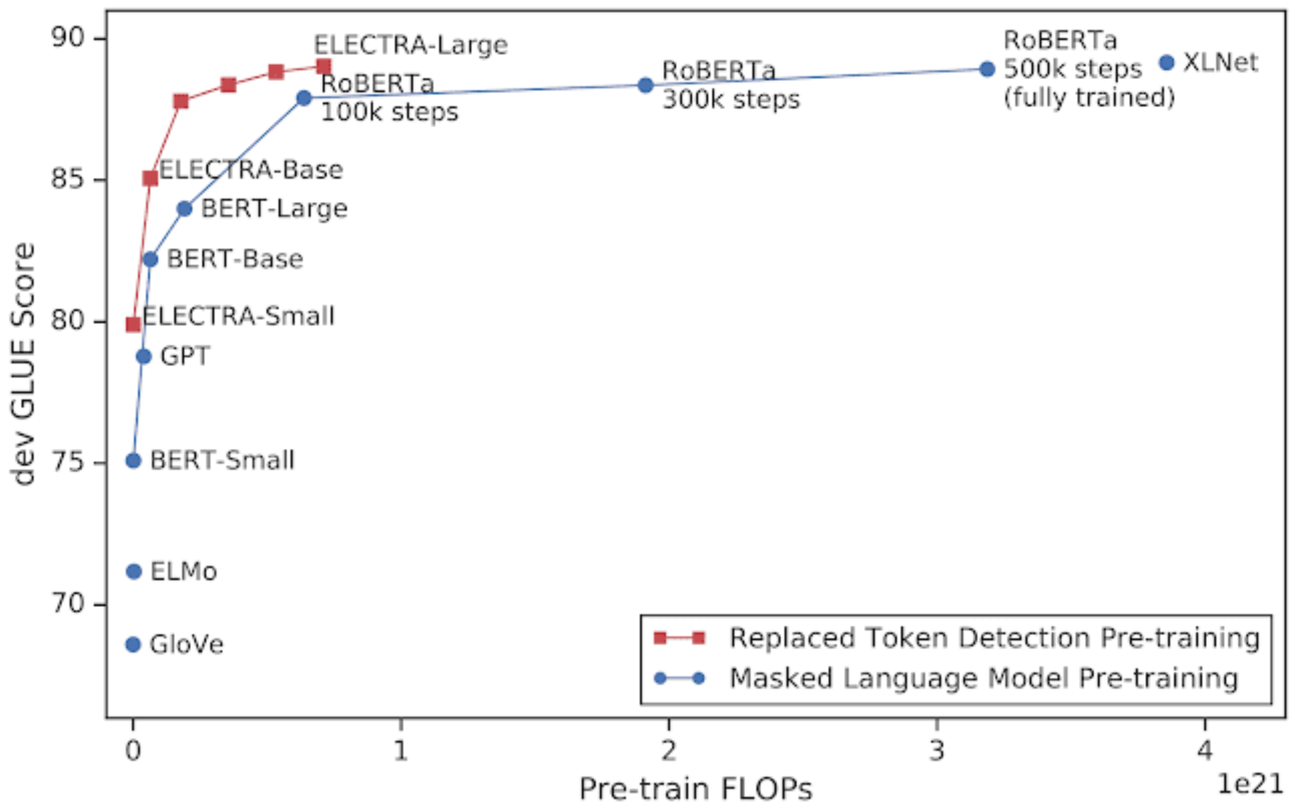
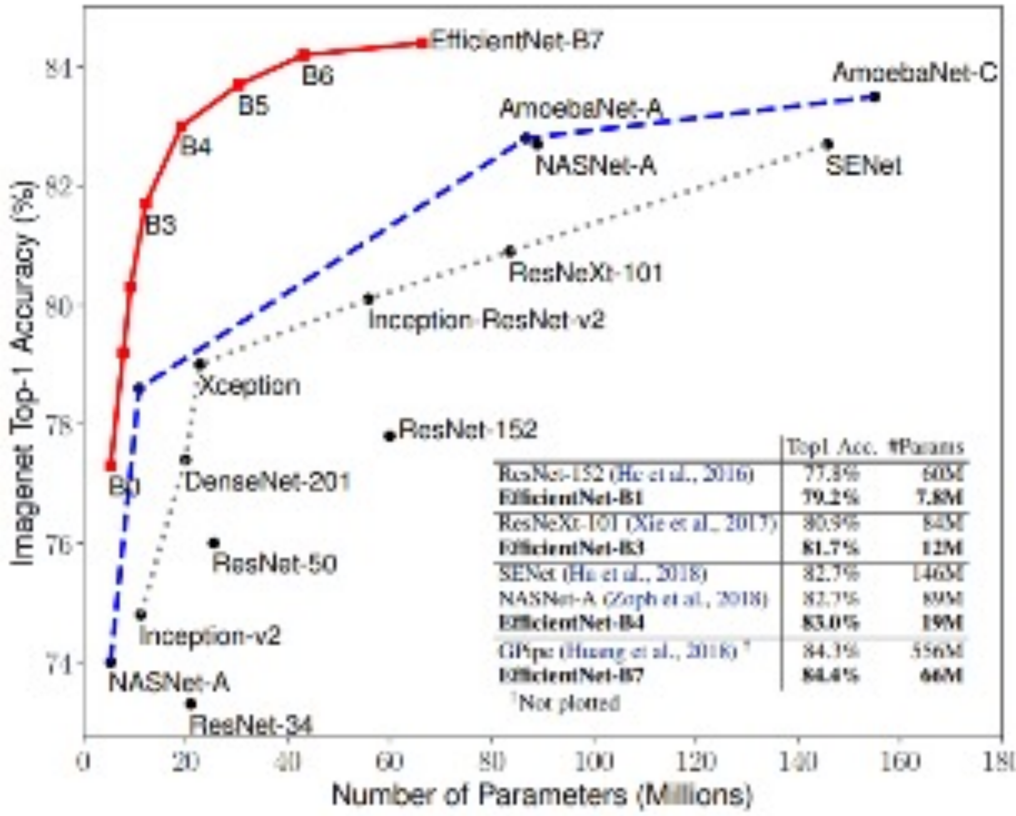


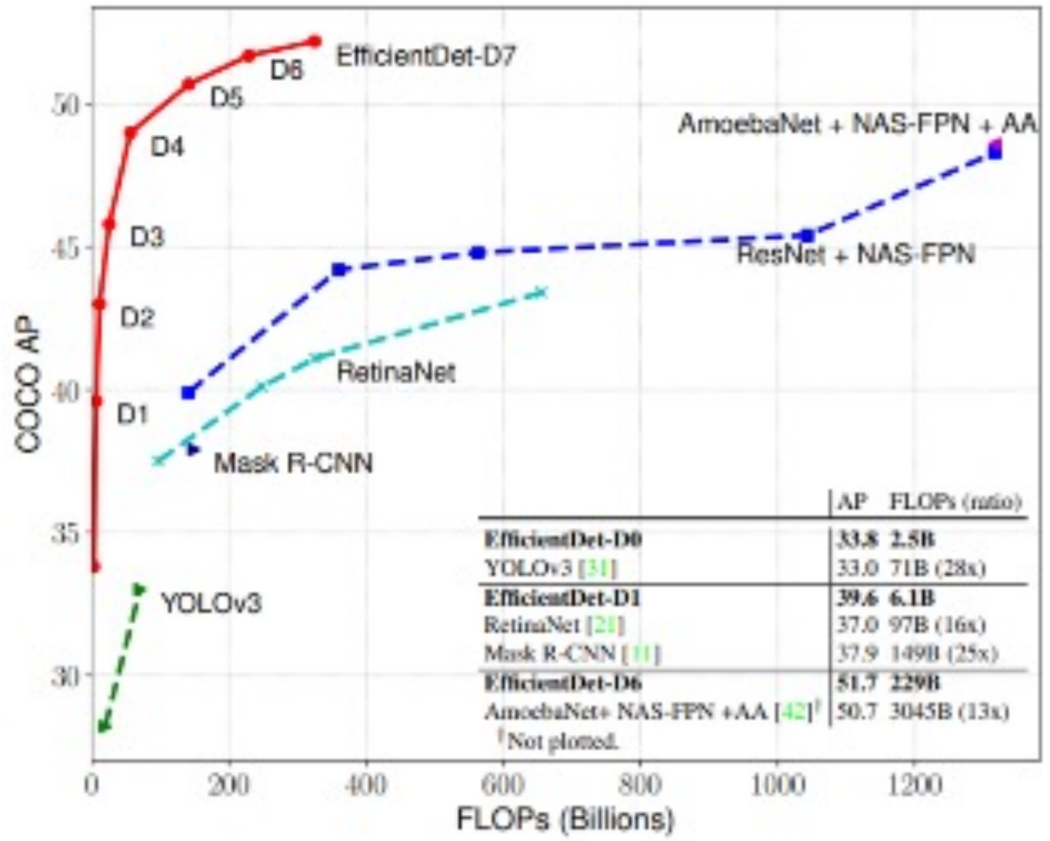
Image Classification



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Object detection



	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [31]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [21]	37.0	97B (16x)
Mask R-CNN [11]	37.9	149B (25x)
EfficientDet-D6	**51.7**	**229B**
AmoebaNet+ NAS-FPN +AA [42]†	50.7	3045B (13x)

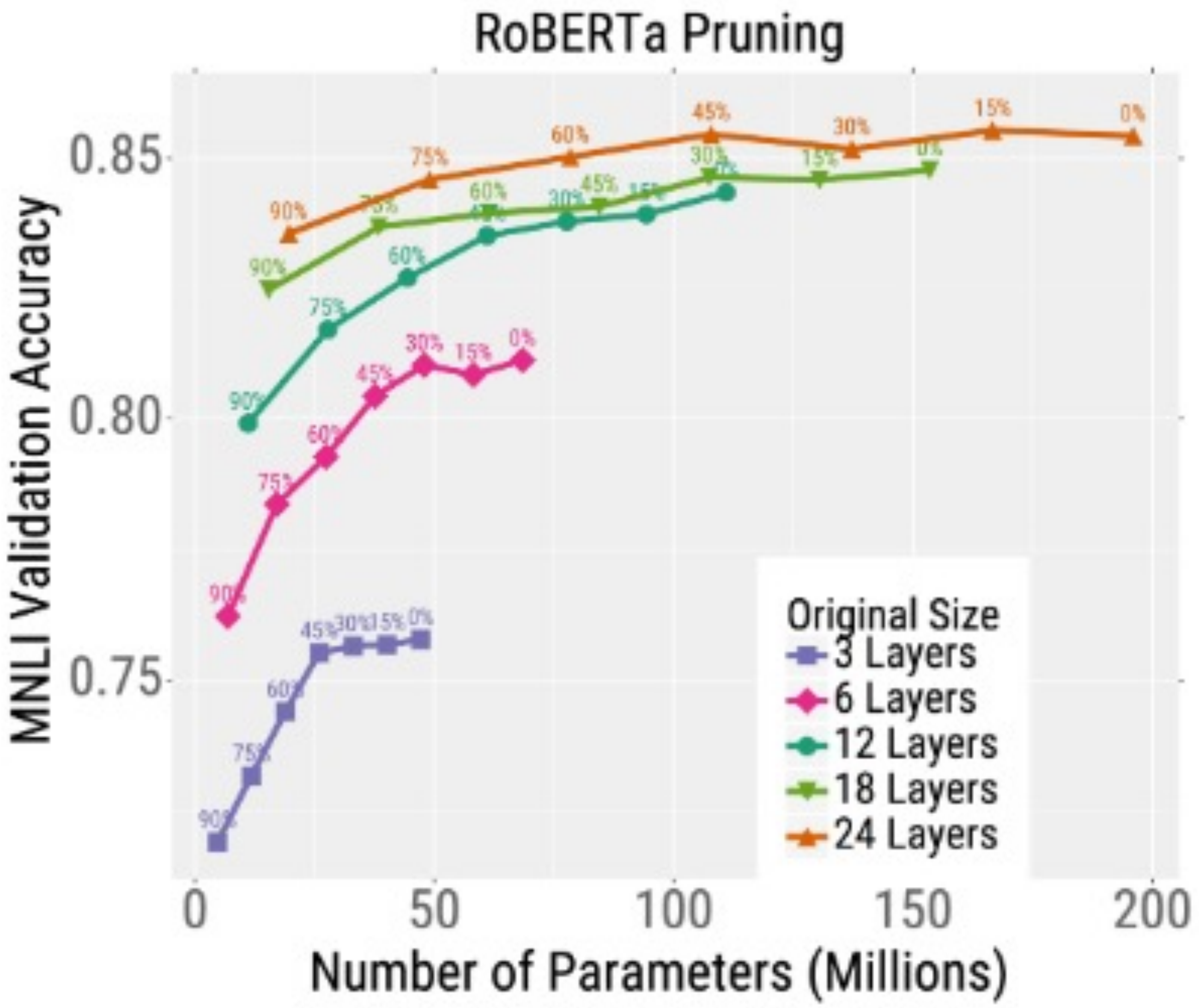
†Not plotted.

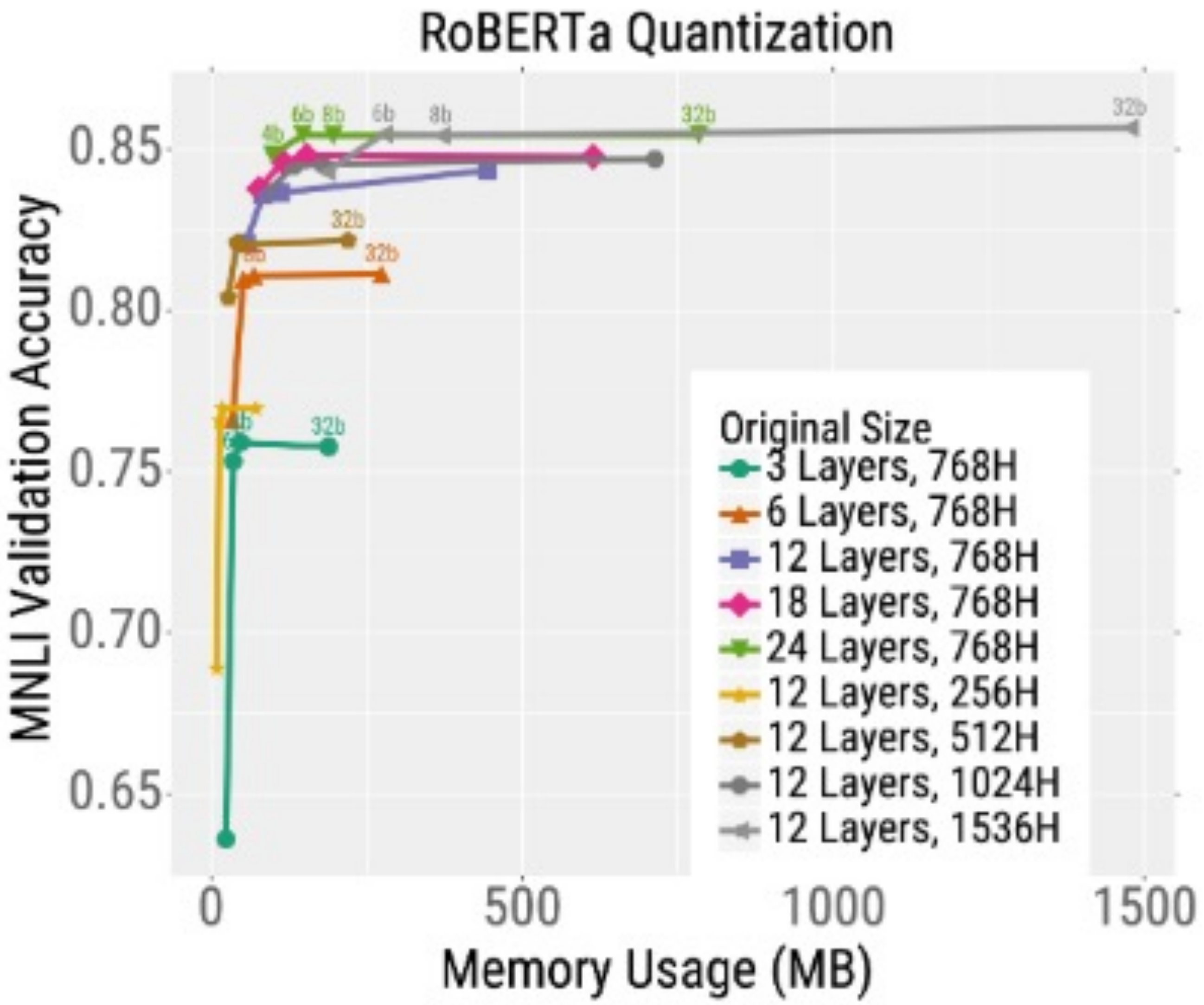
MODEL SELECTION

Not all models respond in the same way to knowledge distillation, pruning and quantization

Common Practice: Train Small Model → Stop Training When Converged → Lightly Compress

Optimal: Train Large Model → Stop Training Early → Heavily Compress





https://bair.berkeley.edu/blog/2020/03/05/compress/

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., & Gonzalez, J. E. (2020). Train large, then compress: Rethinking model size for efficient training and inference of transformers. arXiv preprint arXiv:2002.11794.

MODEL SELECTION

And very large models are and will continue to be prevalent in NLP

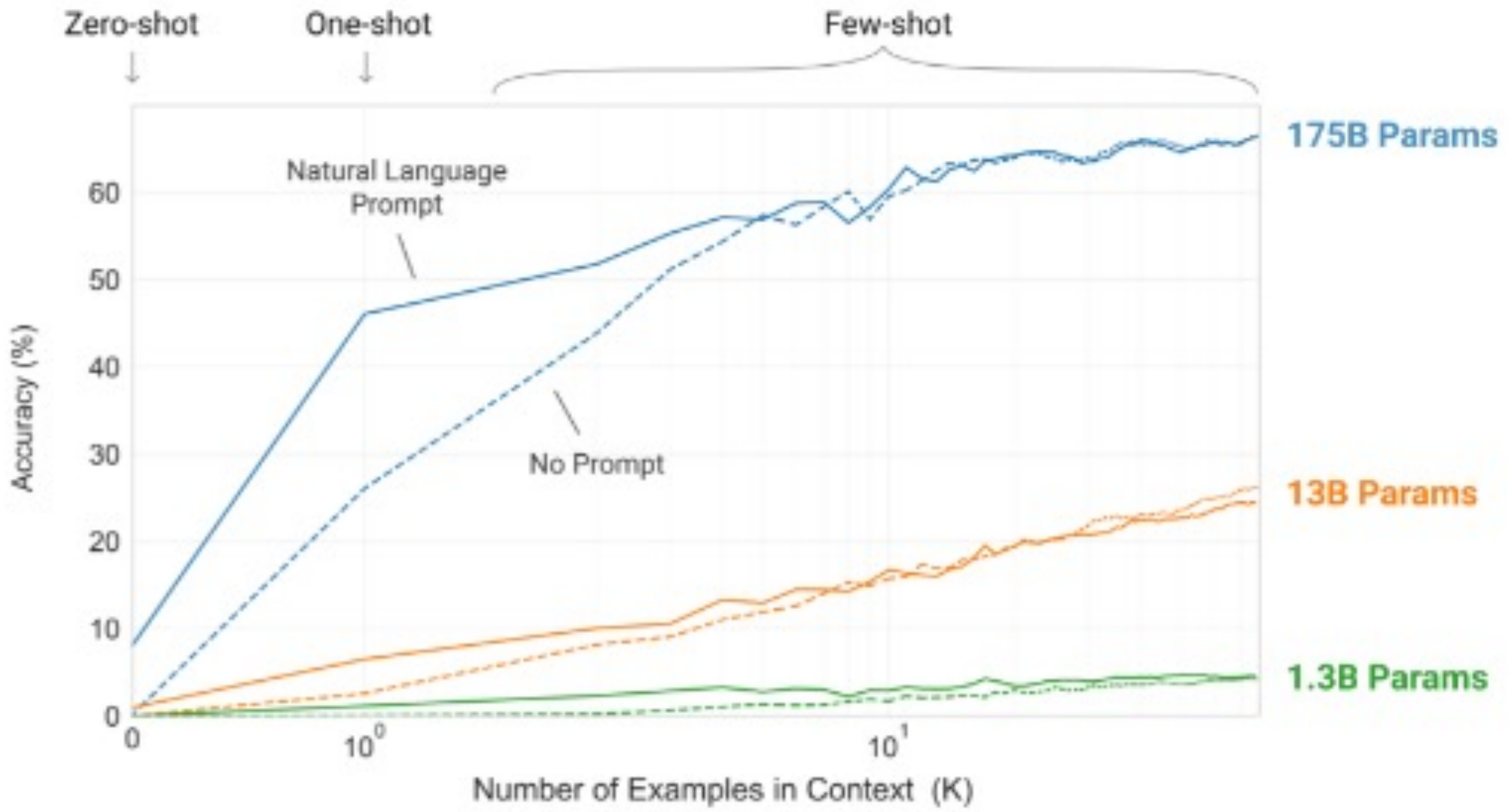


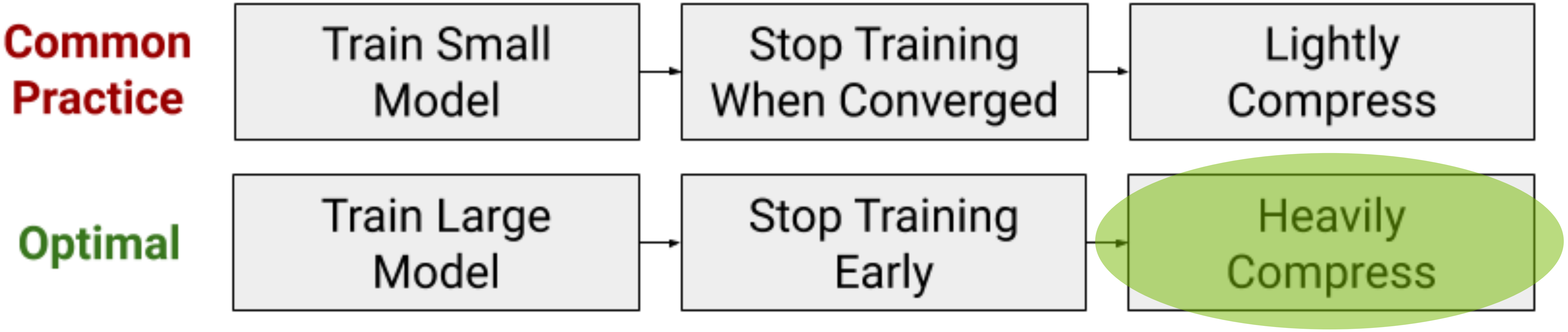
Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Agarwal, S. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165.

DIRECT IMPLICATIONS

INCREASING IMPORTANCE OF PRUNING AND QUANTIZATION

E.g. Train Large then compress



Common Practice: Train Small Model → Stop Training When Converged → Lightly Compress

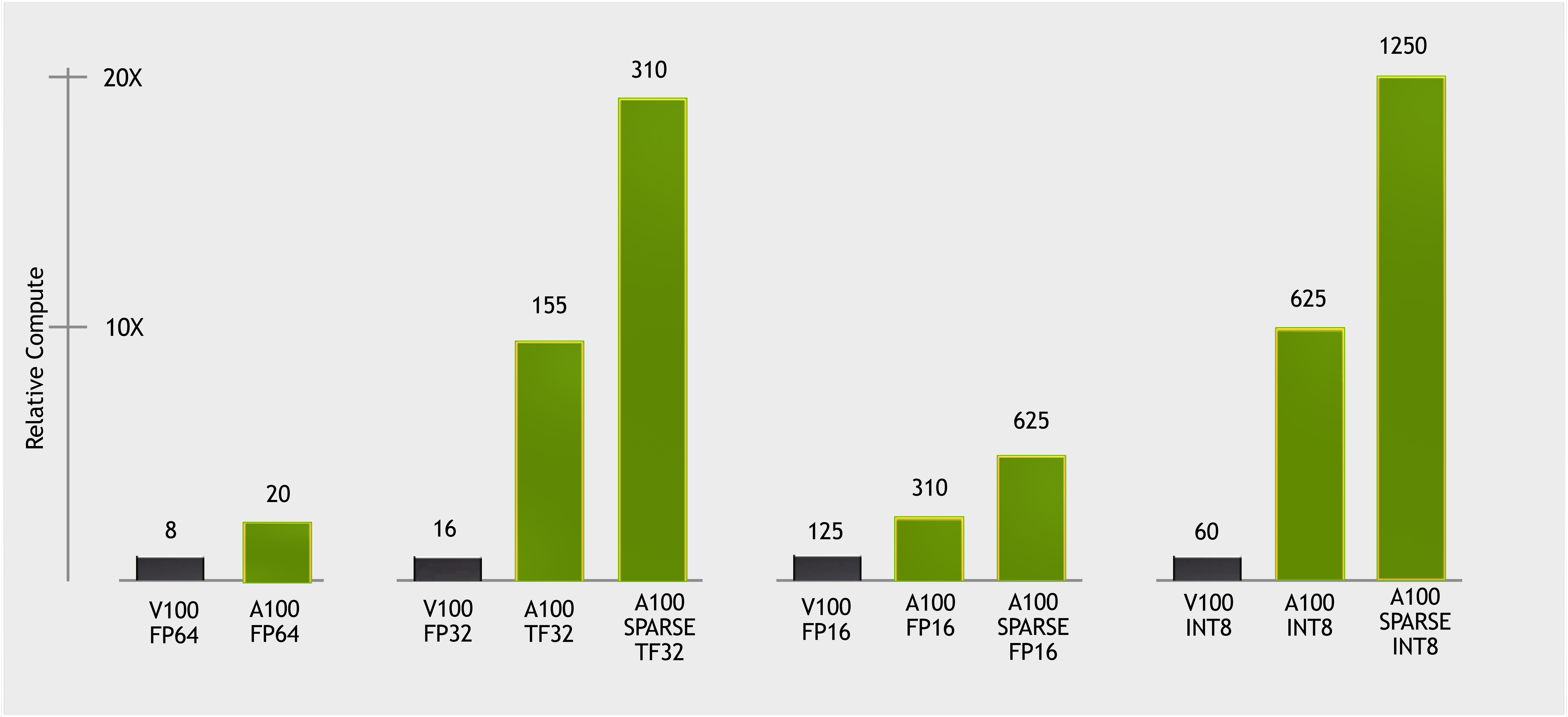
Optimal: Train Large Model → Stop Training Early → Heavily Compress

https://bair.berkeley.edu/blog/2020/03/05/compress/

Li, Z., Wallace, E., Shen, S., Lin, K., Keutzer, K., Klein, D., & Gonzalez, J. E. (2020). Train large, then compress: Rethinking model size for efficient training and inference of transformers. arXiv preprint arXiv:2002.11794.

INCREASING IMPORTANCE OF PRUNING AND QUANTIZATION

Hardware acceleration for reduced precision arithmetic and sparsity

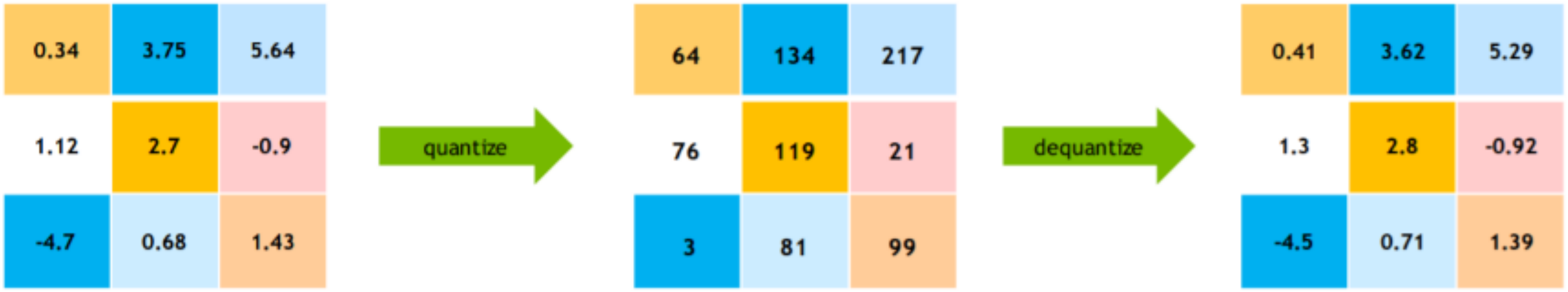


Part 3: Production Deployment

- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model

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

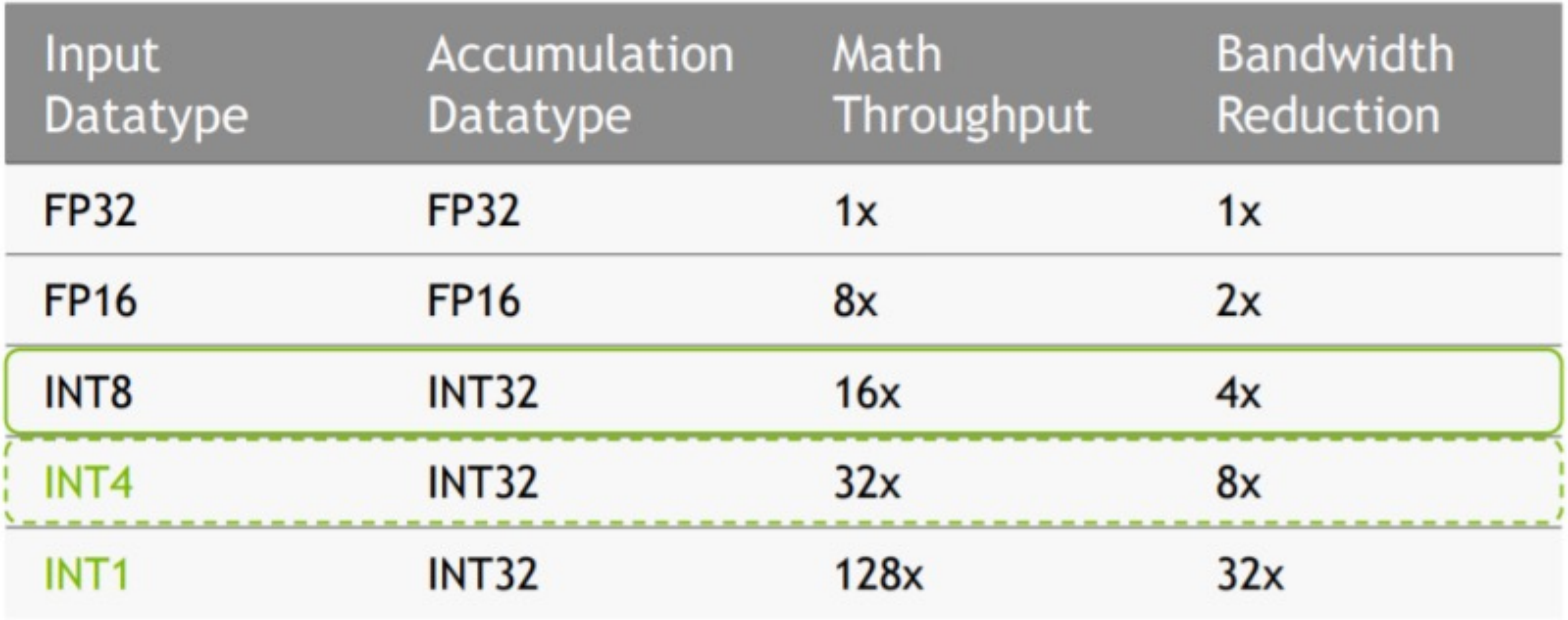
INT8
(quantized)

0.41	3.62	5.29
1.3	2.8	-0.92
-4.5	0.71	1.39

FP32
(dequantized)

QUANTIZATION

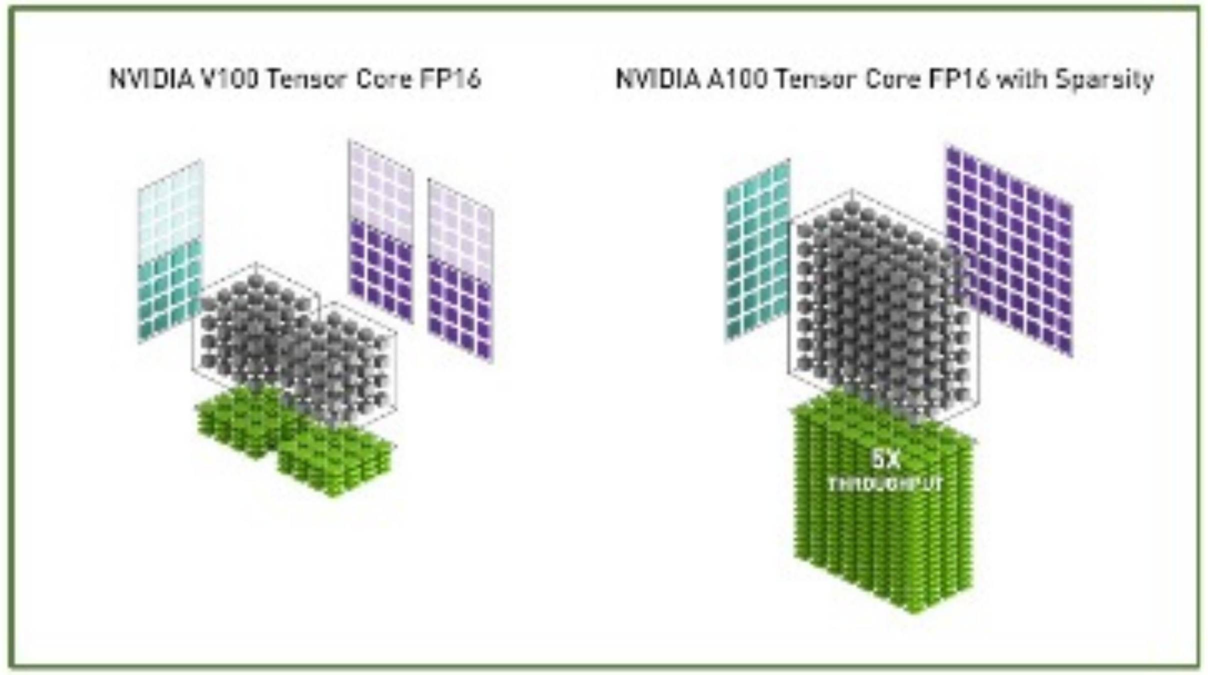
The rationale

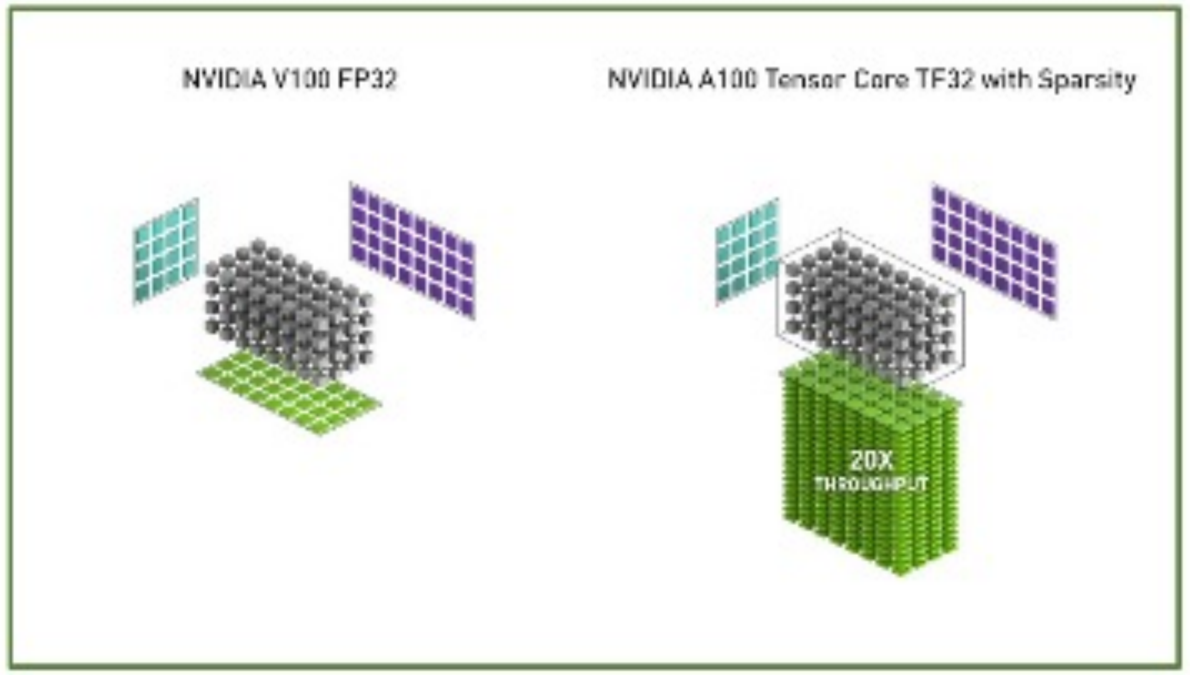


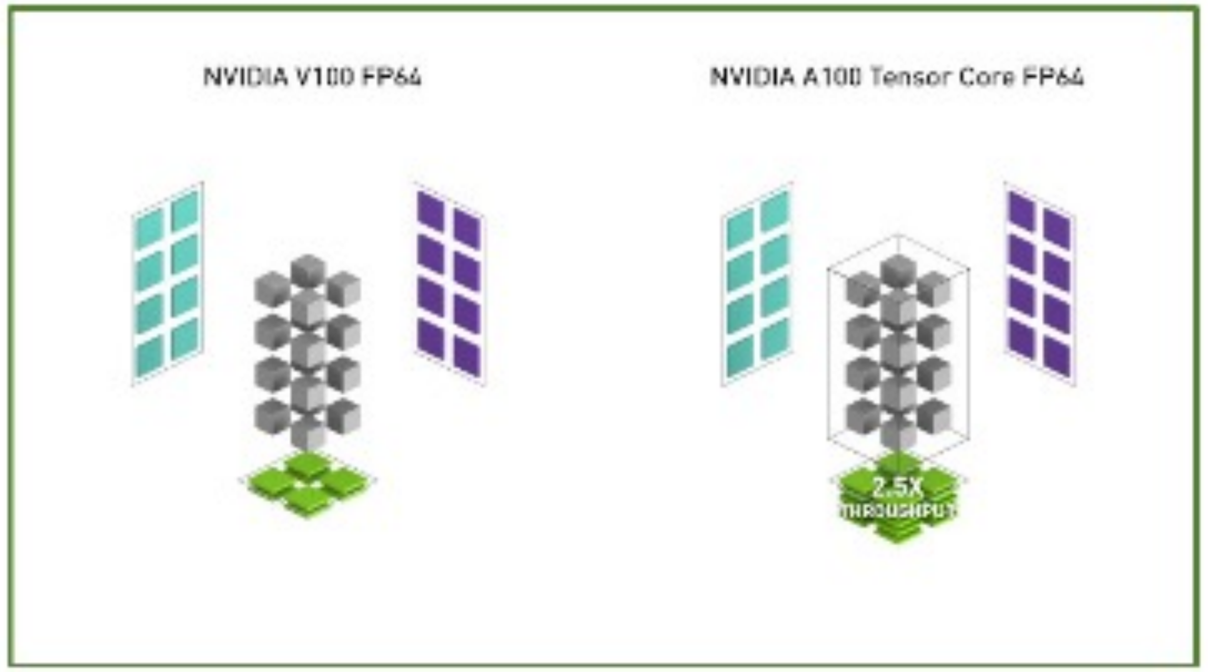
Input Datatype	Accumulation Datatype	Math Throughput	Bandwidth Reduction
FP32	FP32	1x	1x
FP16	FP16	8x	2x
INT8	INT32	16x	4x
INT4	INT32	32x	8x
INT1	INT32	128x	32x

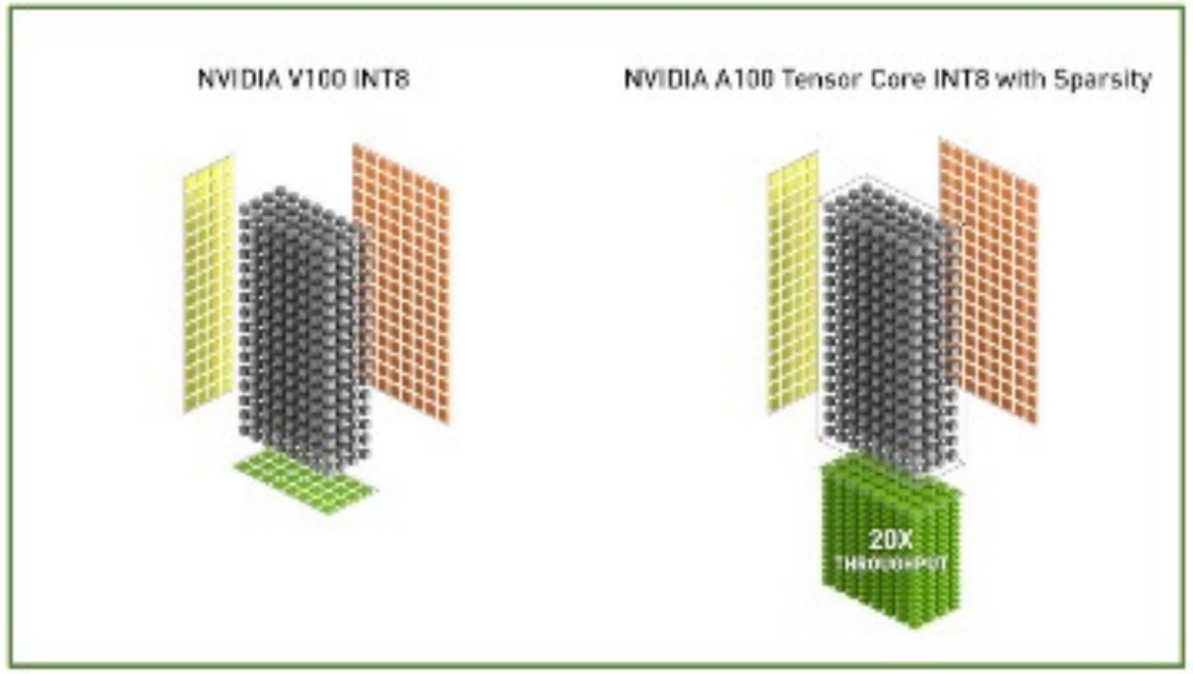
QUANTIZATION

The rationale


NVIDIA V100 Tensor Core FP16
NVIDIA A100 Tensor Core FP16 with Sparsity
5X
THROUGHPUT

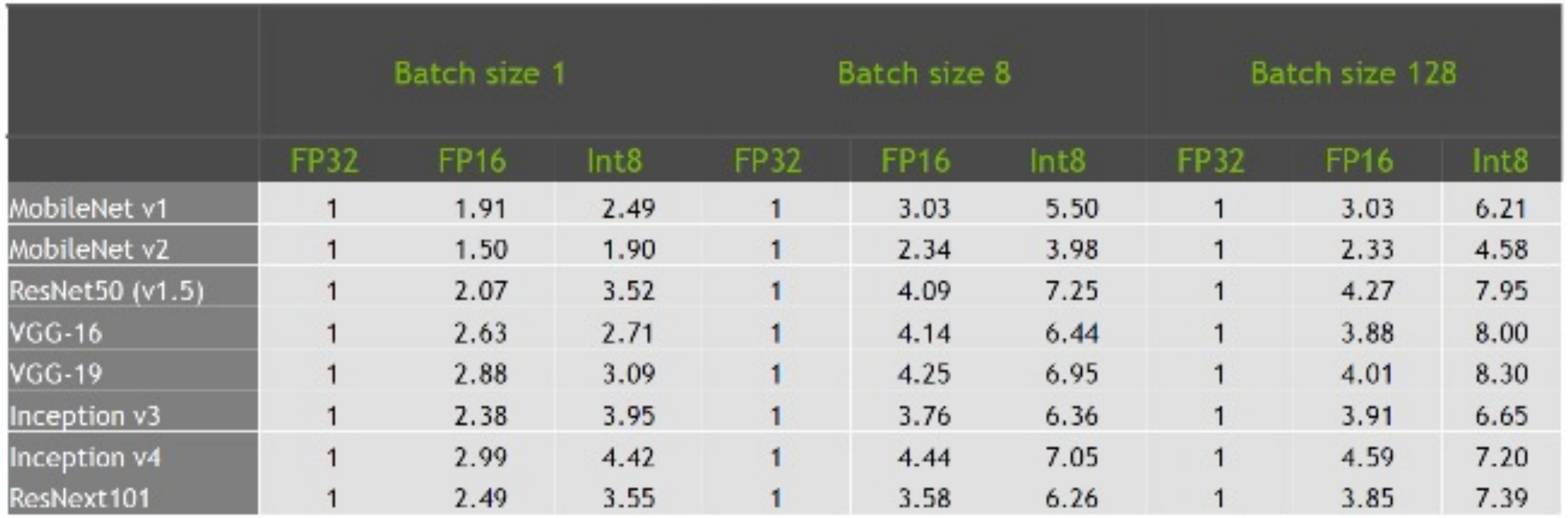




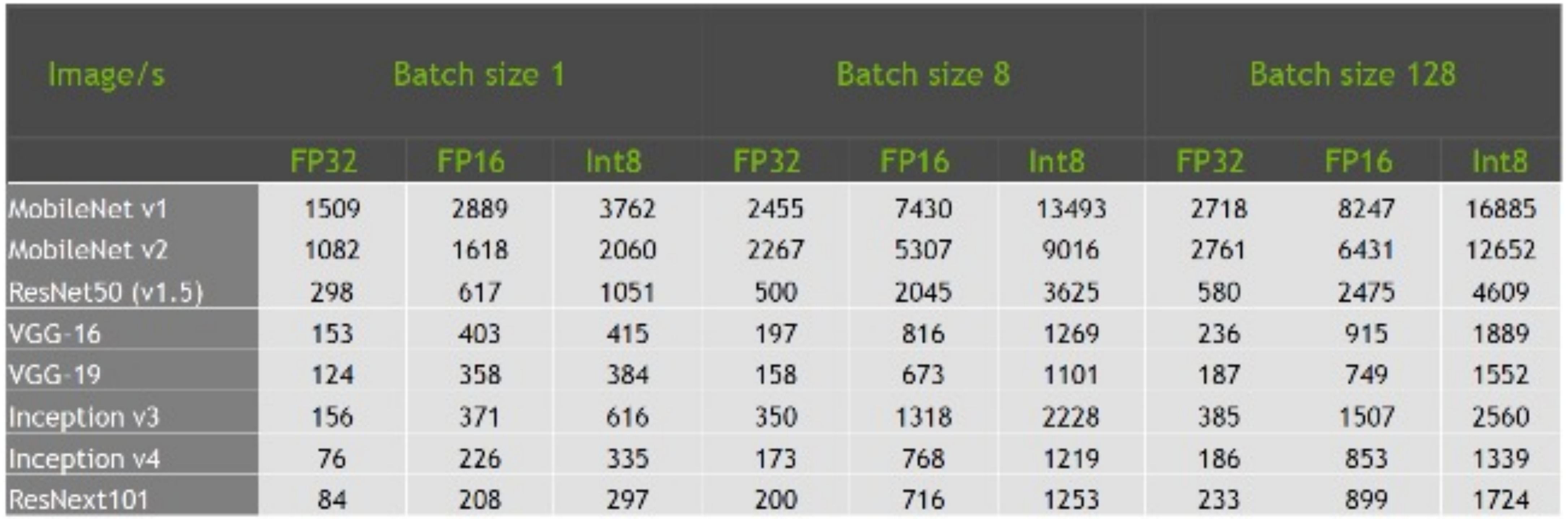


QUANTIZATION

The results (speedup and throughput)



	Batch size 1			Batch size 8			Batch size 128		
	FP32	FP16	Int8	FP32	FP16	Int8	FP32	FP16	Int8
MobileNet v1	1	1.91	2.49	1	3.03	5.50	1	3.03	6.21
MobileNet v2	1	1.50	1.90	1	2.34	3.98	1	2.33	4.58
ResNet50 (v1.5)	1	2.07	3.52	1	4.09	7.25	1	4.27	7.95
VGG-16	1	2.63	2.71	1	4.14	6.44	1	3.88	8.00
VGG-19	1	2.88	3.09	1	4.25	6.95	1	4.01	8.30
Inception v3	1	2.38	3.95	1	3.76	6.36	1	3.91	6.65
Inception v4	1	2.99	4.42	1	4.44	7.05	1	4.59	7.20
ResNext101	1	2.49	3.55	1	3.58	6.26	1	3.85	7.39

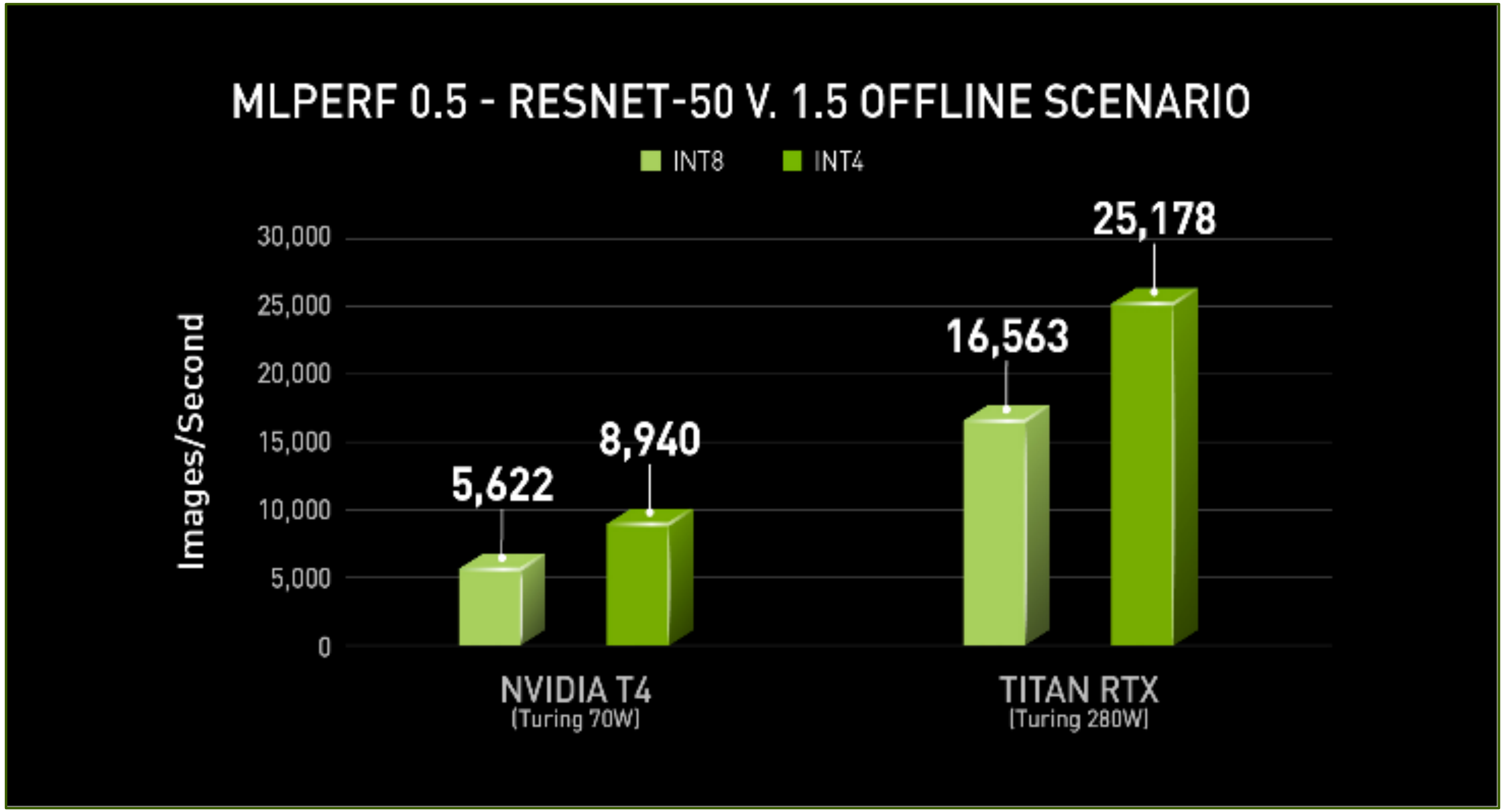


Image/s	Batch size 1			Batch size 8			Batch size 128		
	FP32	FP16	Int8	FP32	FP16	Int8	FP32	FP16	Int8
MobileNet v1	1509	2889	3762	2455	7430	13493	2718	8247	16885
MobileNet v2	1082	1618	2060	2267	5307	9016	2761	6431	12652
ResNet50 (v1.5)	298	617	1051	500	2045	3625	580	2475	4609
VGG-16	153	403	415	197	816	1269	236	915	1889
VGG-19	124	358	384	158	673	1101	187	749	1552
Inception v3	156	371	616	350	1318	2228	385	1507	2560
Inception v4	76	226	335	173	768	1219	186	853	1339
ResNext101	84	208	297	200	716	1253	233	899	1724

TensorRT optimized models executed on Tesla T4, input size 224x224 for all apart from the Inception networks for which the input size was 299x299

QUANTIZATION

Beyond INT8

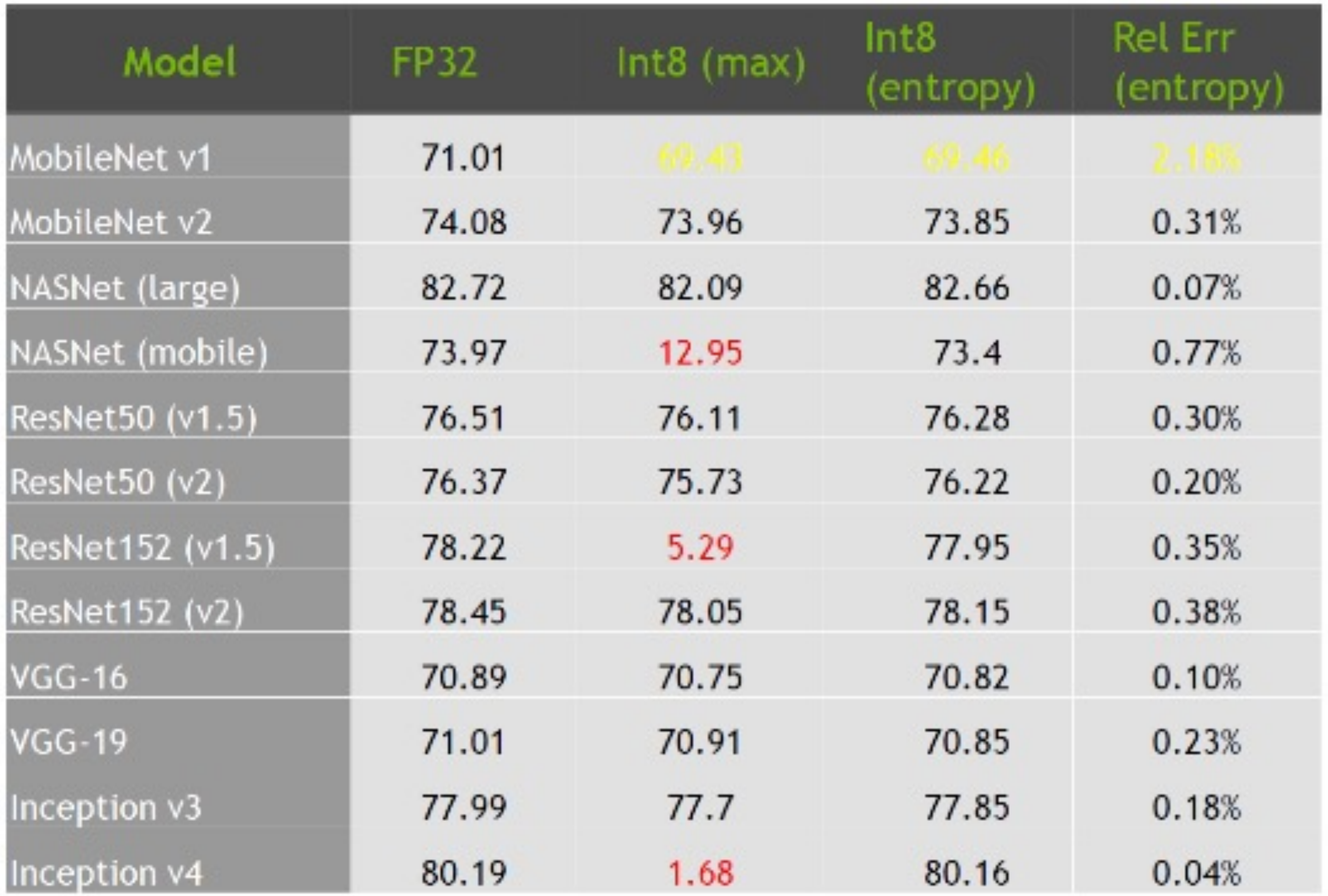


INT4 quantization for resnet50

"Int4 Precision for AI Inference"

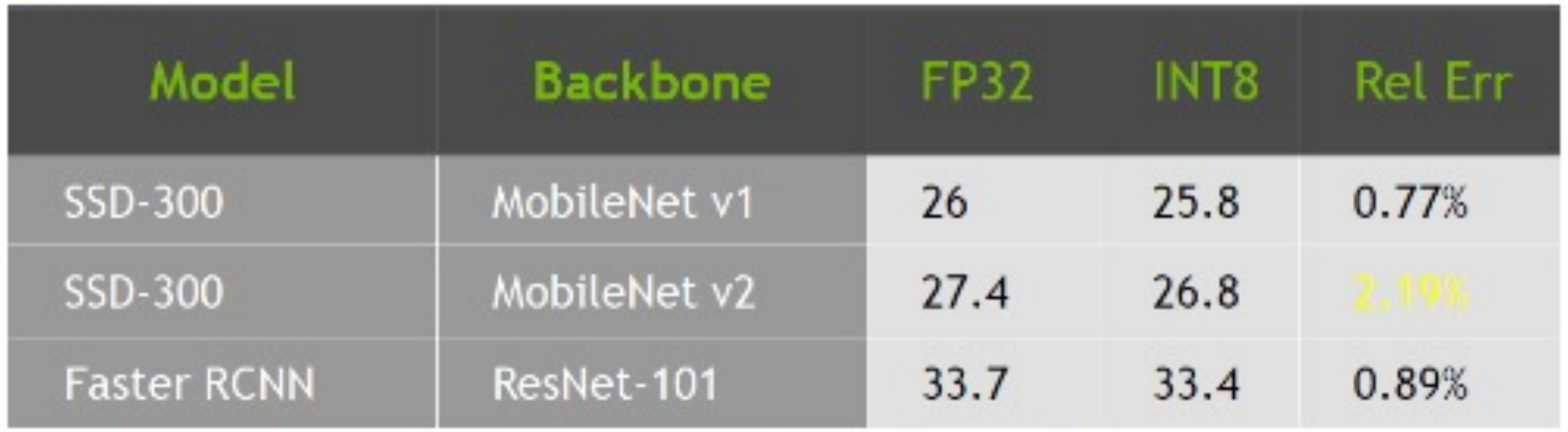
IMPACT ON ACCURACY

In a wide range of cases minimal



Model	FP32	Int8 (max)	Int8 (entropy)	Rel Err (entropy)
MobileNet v1	71.01	69.43	69.46	2.18%
MobileNet v2	74.08	73.96	73.85	0.31%
NASNet (large)	82.72	82.09	82.66	0.07%
NASNet (mobile)	73.97	12.95	73.4	0.77%
ResNet50 (v1.5)	76.51	76.11	76.28	0.30%
ResNet50 (v2)	76.37	75.73	76.22	0.20%
ResNet152 (v1.5)	78.22	5.29	77.95	0.35%
ResNet152 (v2)	78.45	78.05	78.15	0.38%
VGG-16	70.89	70.75	70.82	0.10%
VGG-19	71.01	70.91	70.85	0.23%
Inception v3	77.99	77.7	77.85	0.18%
Inception v4	80.19	1.68	80.16	0.04%

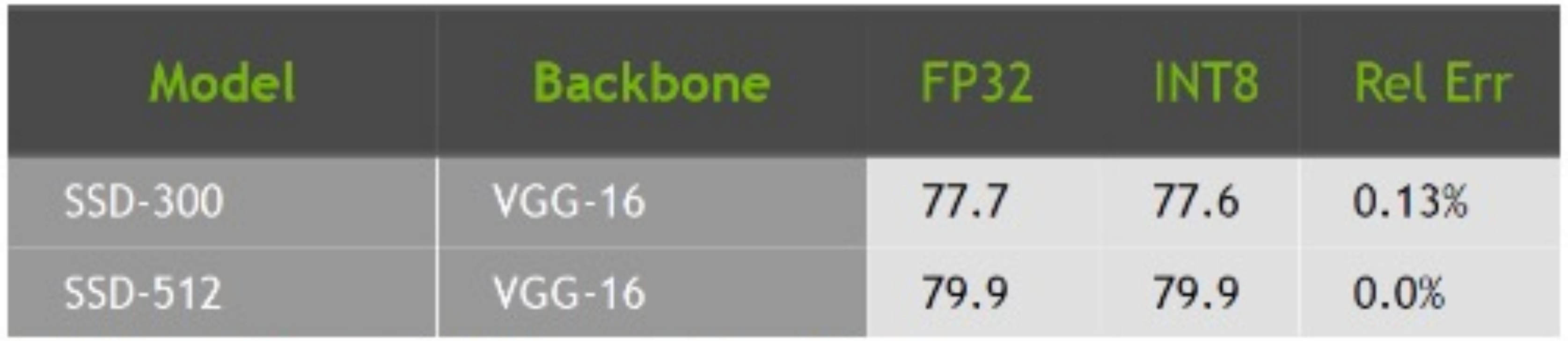
COCO



Model	Backbone	FP32	INT8	Rel Err
SSD-300	MobileNet v1	26	25.8	0.77%
SSD-300	MobileNet v2	27.4	26.8	2.19%
Faster RCNN	ResNet-101	33.7	33.4	0.89%

All results COCO mAP on COCO 2017 validation, higher is better

Pascal VOC

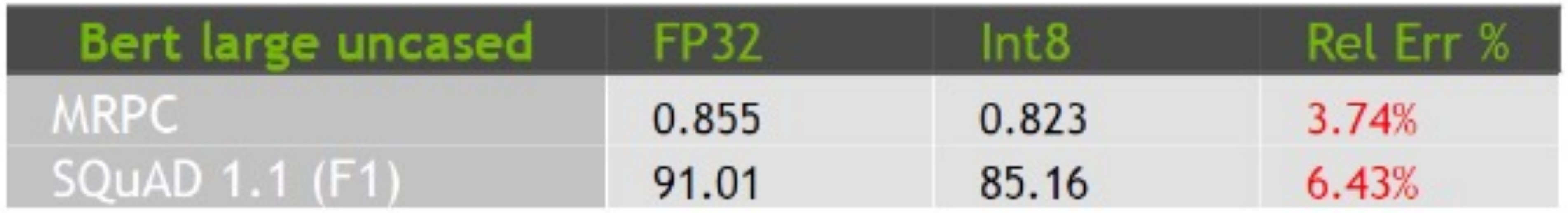


Model	Backbone	FP32	INT8	Rel Err
SSD-300	VGG-16	77.7	77.6	0.13%
SSD-512	VGG-16	79.9	79.9	0.0%

All results VOC mAP on VOC 07 test, higher is better

IMPACT OF MODEL DESIGN

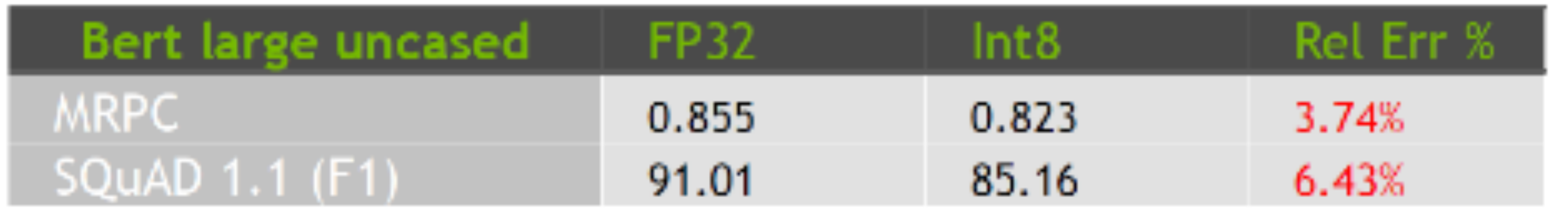
Not all neural network mechanisms quantize well



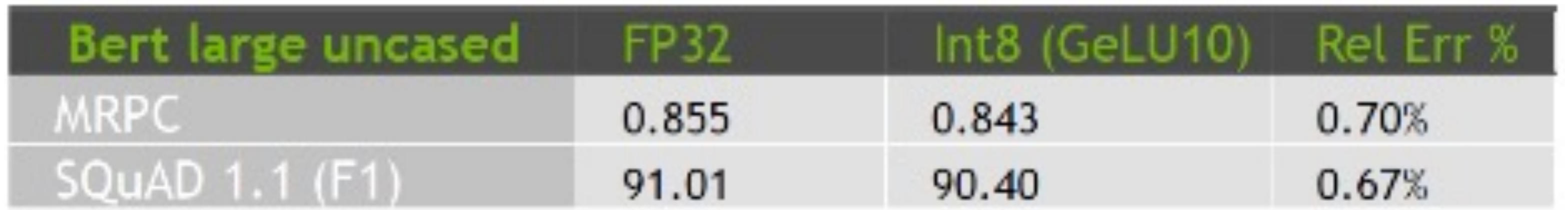
Bert large uncased	FP32	Int8	Rel Err %
MRPC	0.855	0.823	3.74%
SQuAD 1.1 (F1)	91.01	85.16	6.43%

IMPACT OF MODEL DESIGN

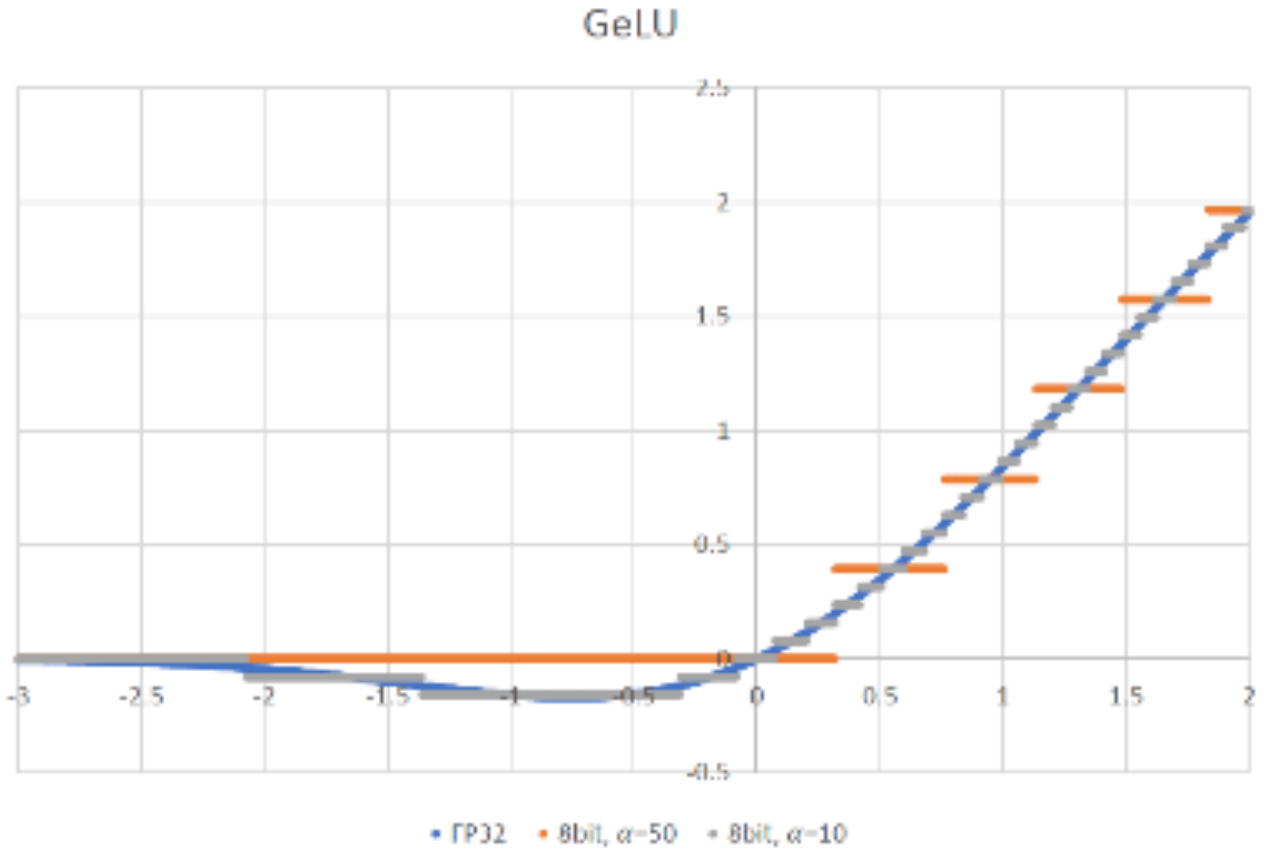
Model alterations required



Bert large uncased	FP32	Int8	Rel Err %
MRPC	0.855	0.823	3.74%
SQuAD 1.1 (F1)	91.01	85.16	6.43%



Bert large uncased	FP32	Int8 (GeLU10)	Rel Err %
MRPC	0.855	0.843	0.70%
SQuAD 1.1 (F1)	91.01	90.40	0.67%



$$f(x) = \frac{x}{2}\left(1 + erf\left(\frac{x}{\sqrt{2}}\right)\right)$$

- GeLU produces highly asymmetric range
- Negative values between [-0.17,0]
- All negative values clipped to 0
- GeLU10 allows to maintain negative values

LOSS OF ACCURACY

Reasons

Outlier in the tensor:

- Example: BERT, Inception V4
- Solution: Clip. Tighten the range, use bits more efficiently

Not enough precision in quantized representation

- Example: Int8 for MobileNet V1
- Example: Int4 for Resnet50
- Solution: Train/fine tune for quantization

LEARN MORE

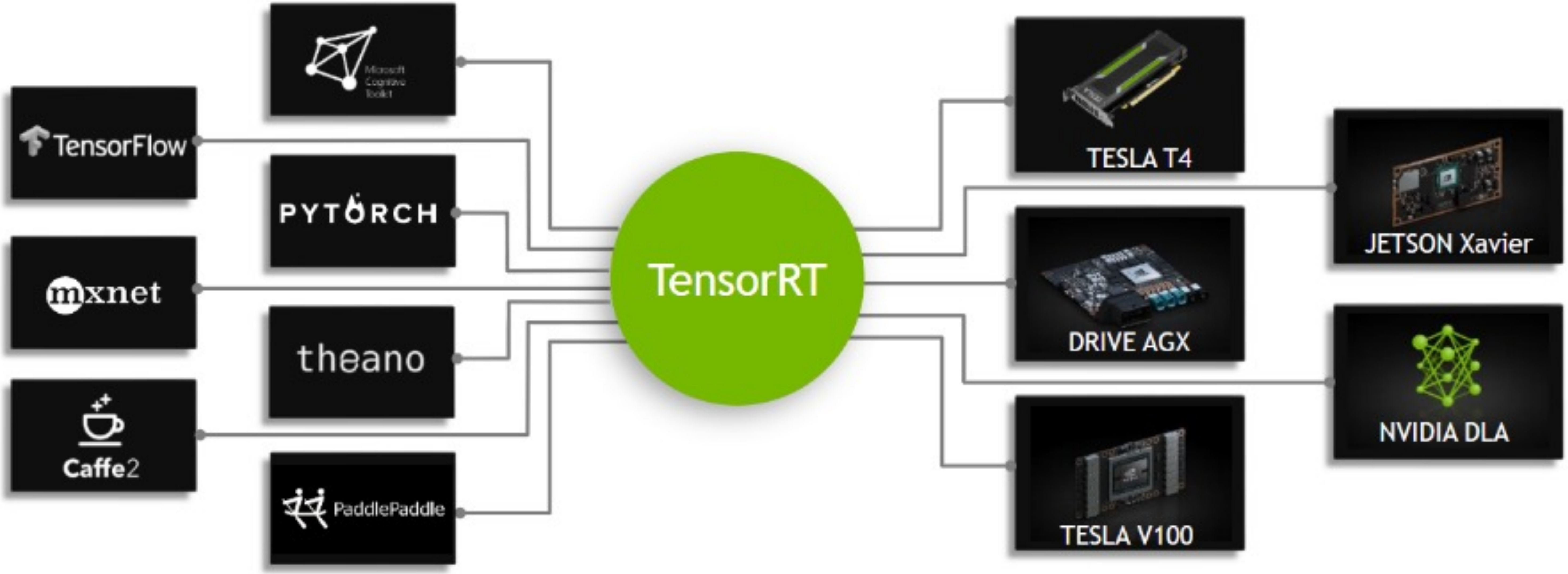
GTC Talks

- S9659: Inference at Reduced Precision on GPUs
- S21664: Toward INT8 Inference: Deploying Quantization-Aware Trained Networks using TensorRT

QUANTIZATION TOOLS

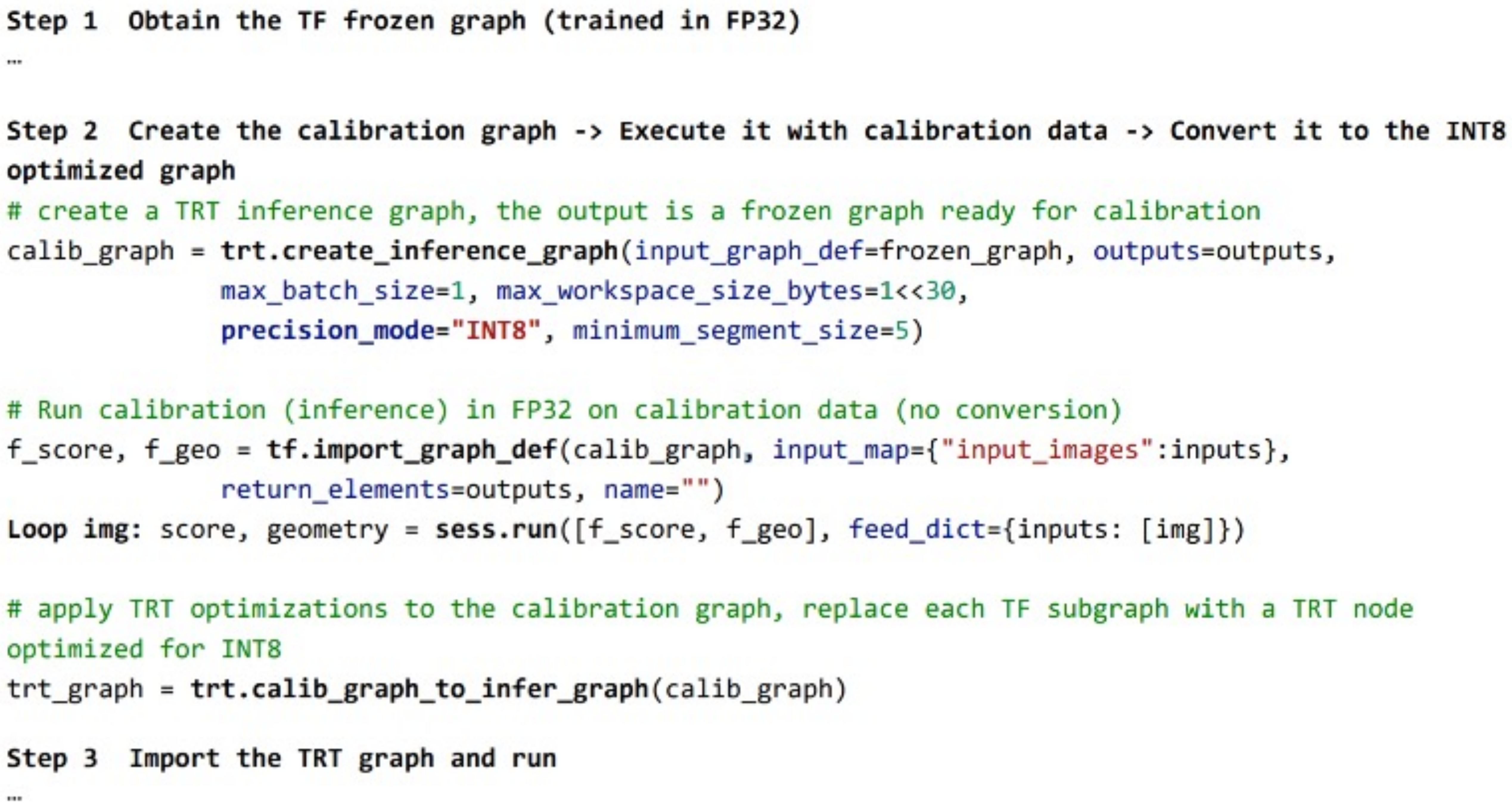
NVIDIA TENSORRT

From Every Framework, Optimized For Each Target Platform



INT8 QUANTIZATION EXAMPLE

TF-TRT



Step 1 Obtain the TF frozen graph (trained in FP32)

…

Step 2 Create the calibration graph -> Execute it with calibration data -> Convert it to the INT8 optimized graph

```
# create a TRT inference graph, the output is a frozen graph ready for calibration
calib_graph = trt.create_inference_graph(input_graph_def=frozen_graph, outputs=outputs,
              max_batch_size=1, max_workspace_size_bytes=1<<30,
              precision_mode="INT8", minimum_segment_size=5)

# Run calibration (inference) in FP32 on calibration data (no conversion)
f_score, f_geo = tf.import_graph_def(calib_graph, input_map={"input_images":inputs},
              return_elements=outputs, name="")
Loop img: score, geometry = sess.run([f_score, f_geo], feed_dict={inputs: [img]})

# apply TRT optimizations to the calibration graph, replace each TF subgraph with a TRT node
optimized for INT8
trt_graph = trt.calib_graph_to_infer_graph(calib_graph)
```

Step 3 Import the TRT graph and run

…

https://docs.nvidia.com/deeplearning/dgx/tf-trt-user-guide/index.html

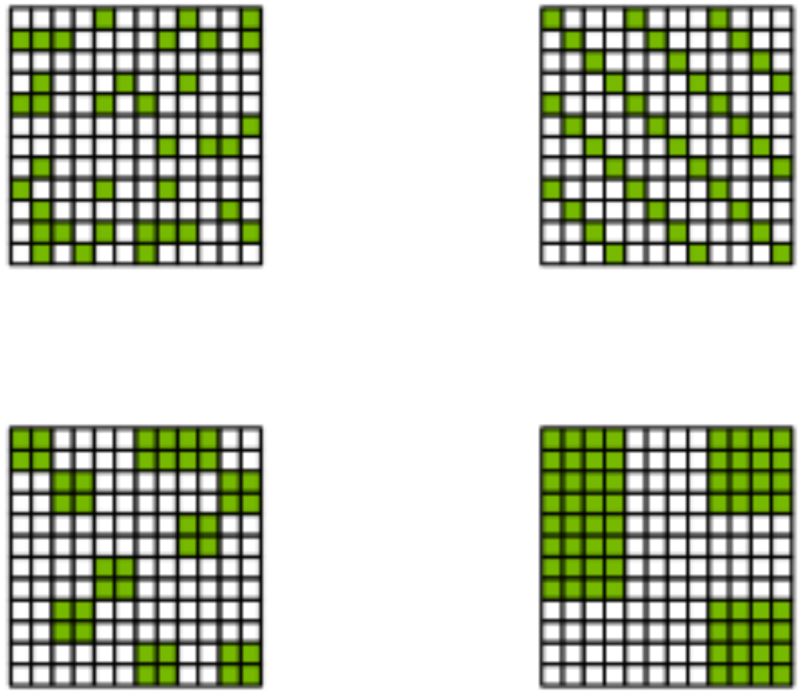
PRUNING

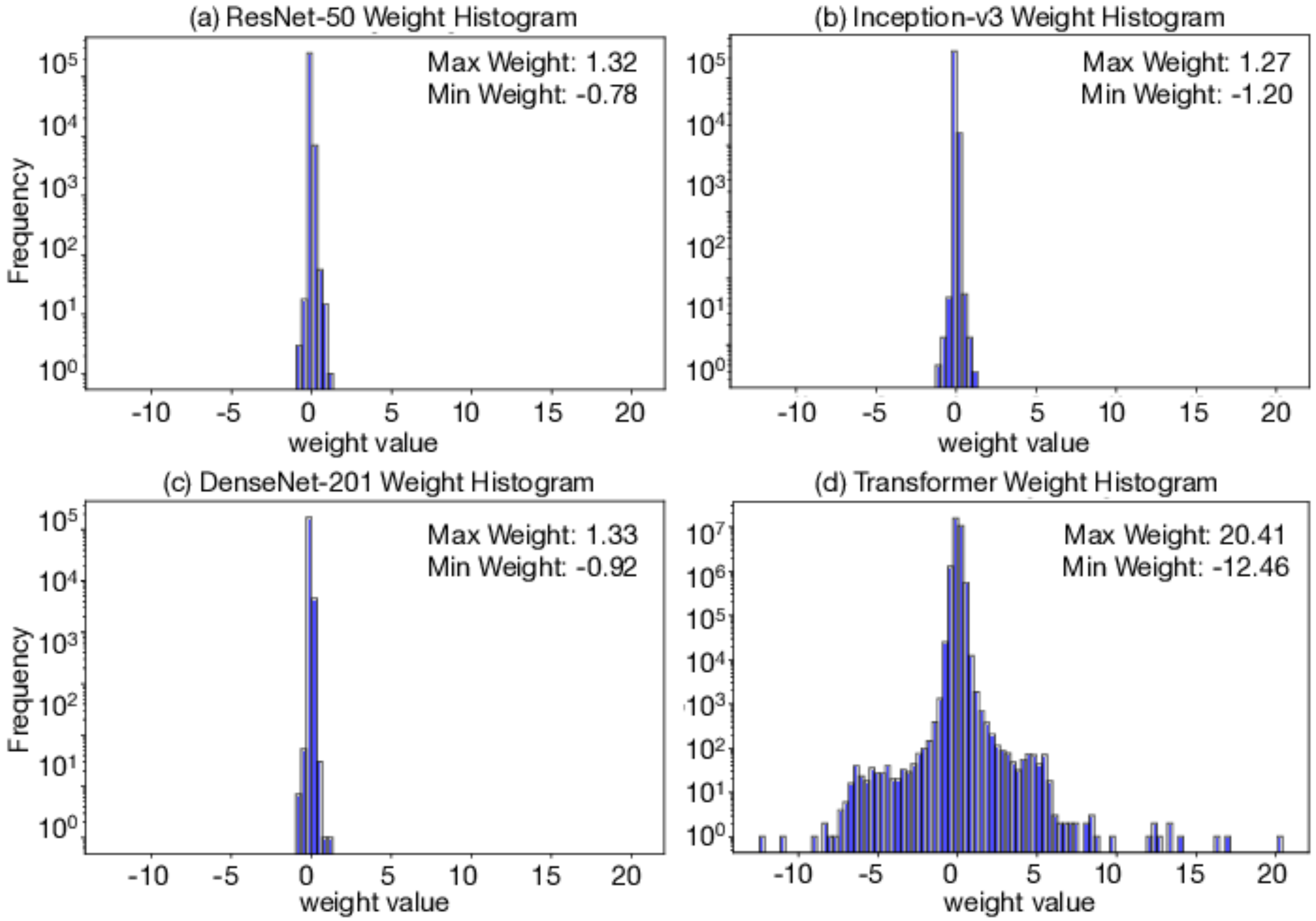
PRUNING

The idea

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*.

DIFFICULT TO GET TO WORK RELIABLY

STRUCTURED SPARSITY

SPARSITY IN A100 GPU

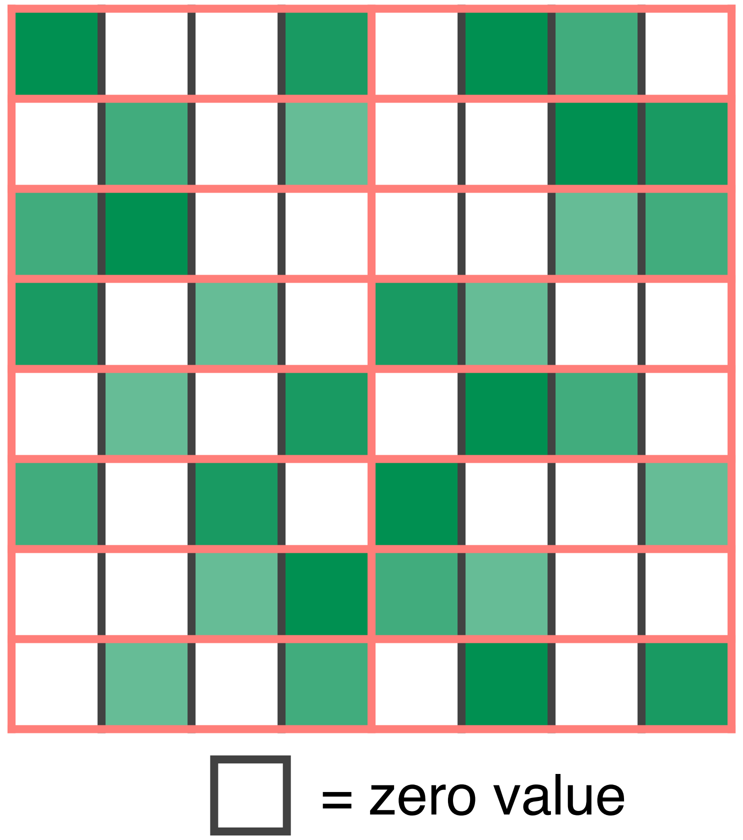
Fine-grained structured sparsity for Tensor Cores

- 50% fine-grained sparsity
- 2:4 pattern: 2 values out of each contiguous block of 4 must be 0

Addresses the 3 challenges:

- **Accuracy:** maintains accuracy of the original, unpruned network
  - Medium sparsity level (50%), fine-grained
- **Training:** a recipe shown to work across tasks and networks
- **Speedup:**
  - Specialized Tensor Core support for sparse math
  - Structured: lends itself to efficient memory utilization

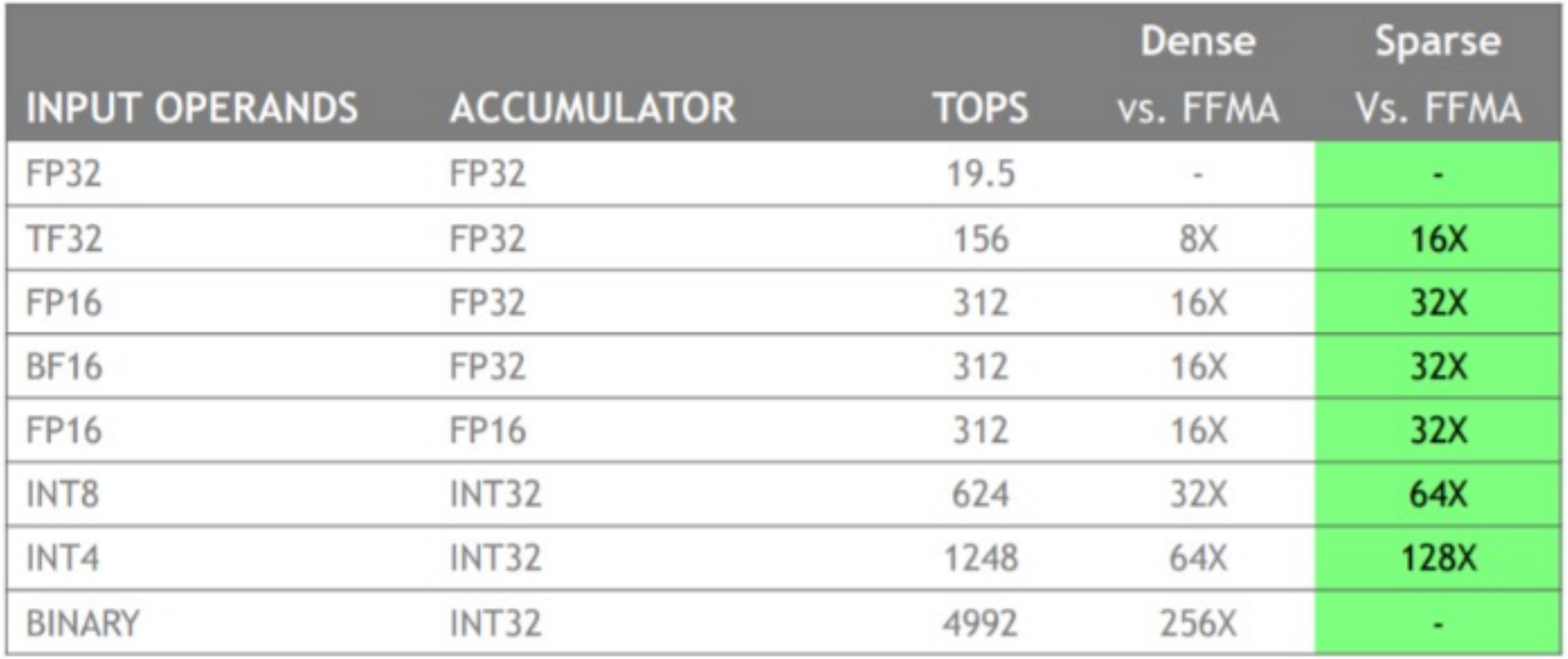
2:4 structured-sparse matrix



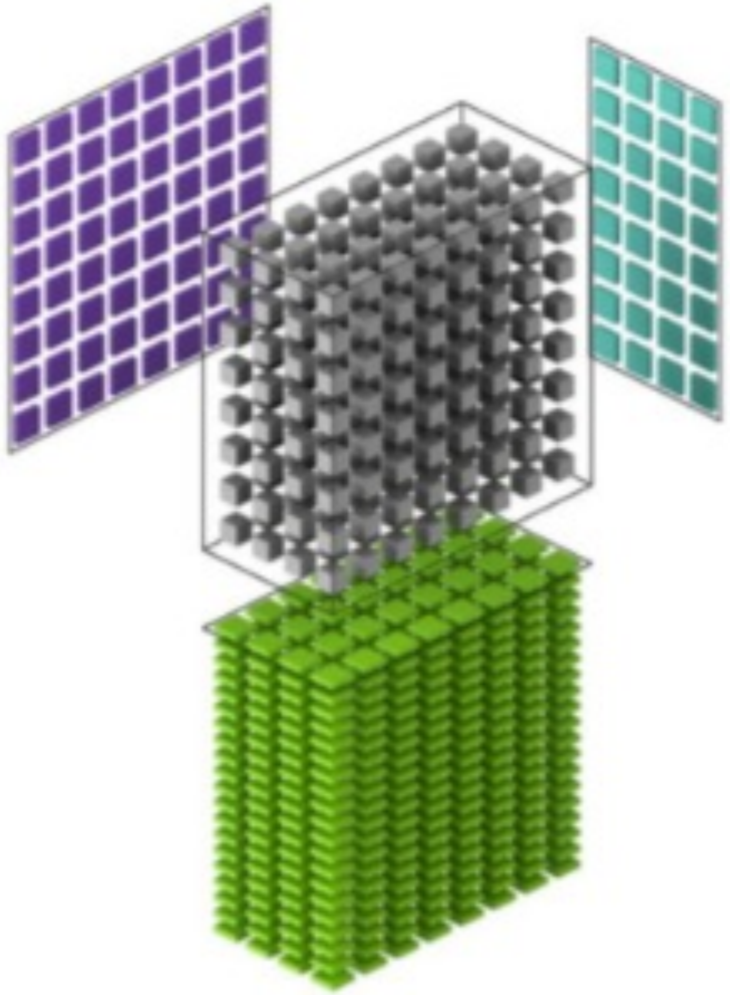
☐ = zero value

PRUNING

Structured sparsity



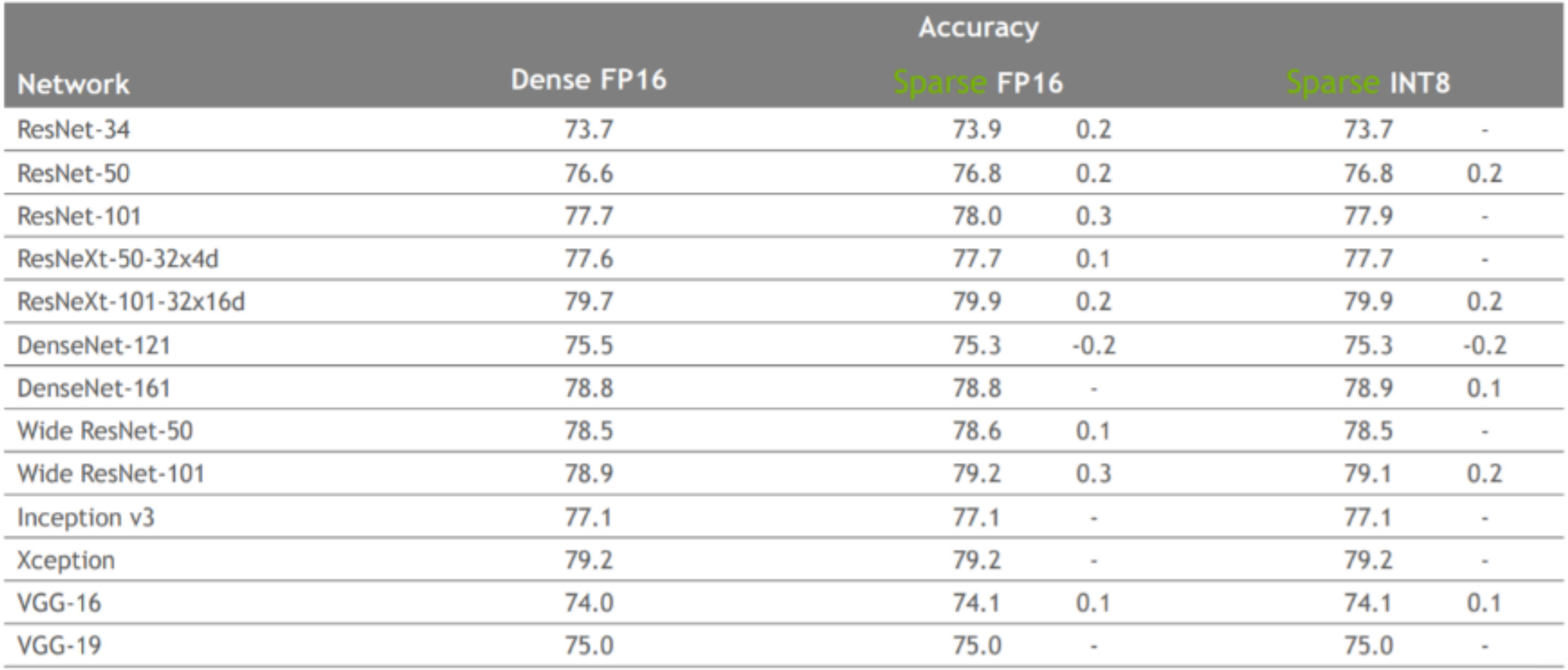
INPUT OPERANDS	ACCUMULATOR	TOPS	Dense vs. FFMA	Sparse Vs. FFMA
FP32	FP32	19.5	-	-
TF32	FP32	156	8X	16X
FP16	FP32	312	16X	32X
BF16	FP32	312	16X	32X
FP16	FP16	312	16X	32X
INT8	INT32	624	32X	64X
INT4	INT32	1248	64X	128X
BINARY	INT32	4992	256X	-



RELIABLE APPROACH

PRUNING

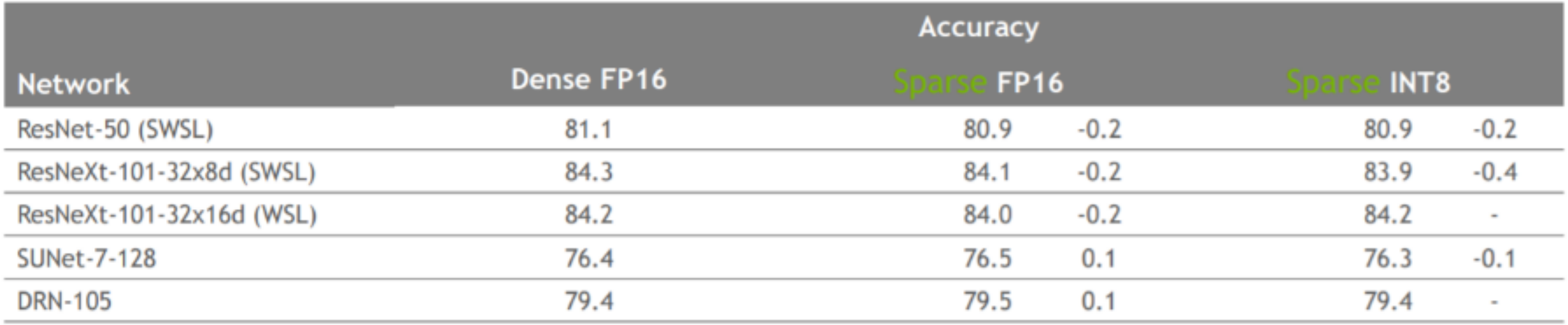
Model performance



Network	Accuracy Dense FP16	Sparse FP16		Sparse INT8	
ResNet-34	73.7	73.9	0.2	73.7	-
ResNet-50	76.6	76.8	0.2	76.8	0.2
ResNet-101	77.7	78.0	0.3	77.9	-
ResNeXt-50-32x4d	77.6	77.7	0.1	77.7	-
ResNeXt-101-32x16d	79.7	79.9	0.2	79.9	0.2
DenseNet-121	75.5	75.3	-0.2	75.3	-0.2
DenseNet-161	78.8	78.8	-	78.9	0.1
Wide ResNet-50	78.5	78.6	0.1	78.5	-
Wide ResNet-101	78.9	79.2	0.3	79.1	0.2
Inception v3	77.1	77.1	-	77.1	-
Xception	79.2	79.2	-	79.2	-
VGG-16	74.0	74.1	0.1	74.1	0.1
VGG-19	75.0	75.0	-	75.0	-

PRUNING

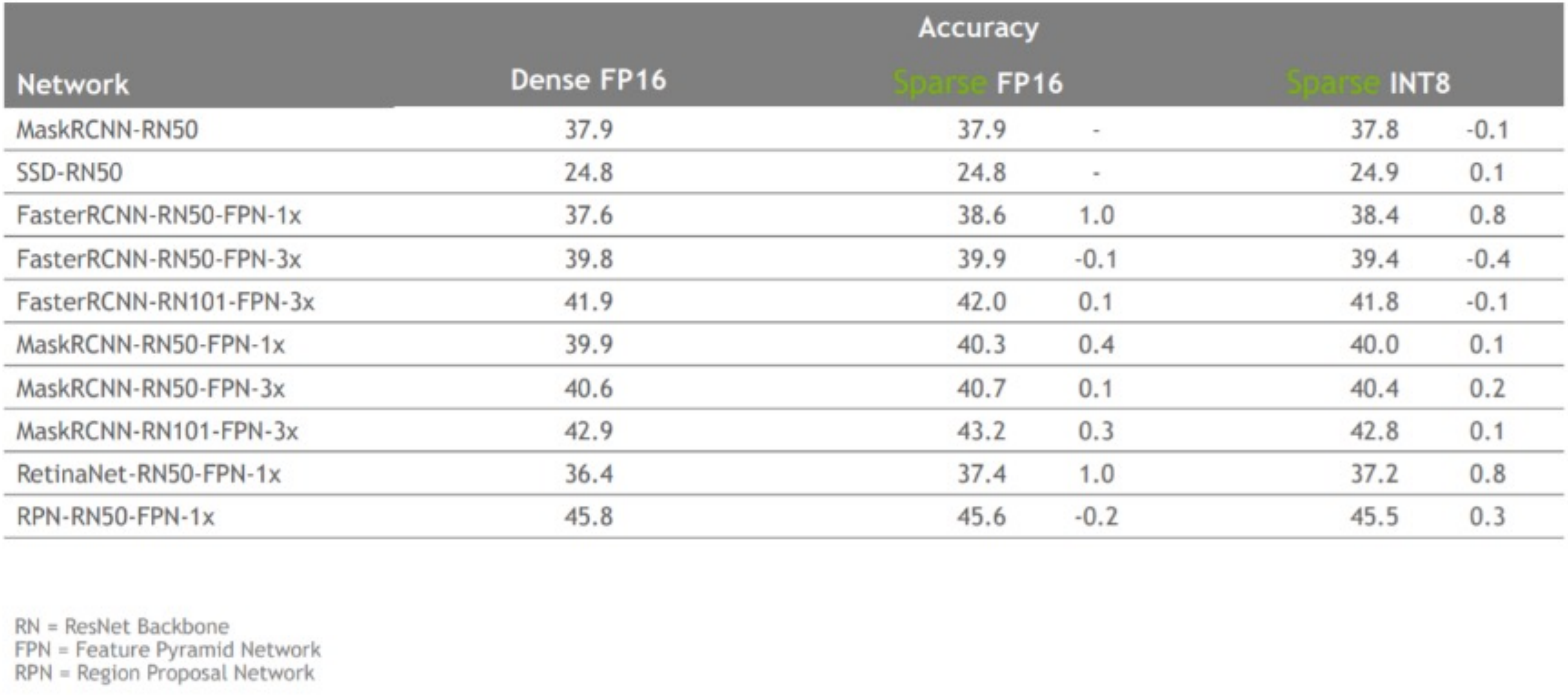
Model performance



Network	Accuracy				
	Dense FP16	Sparse FP16		Sparse INT8	
ResNet-50 (SWSL)	81.1	80.9	-0.2	80.9	-0.2
ResNeXt-101-32x8d (SWSL)	84.3	84.1	-0.2	83.9	-0.4
ResNeXt-101-32x16d (WSL)	84.2	84.0	-0.2	84.2	-
SUNet-7-128	76.4	76.5	0.1	76.3	-0.1
DRN-105	79.4	79.5	0.1	79.4	-

PRUNING

Model performance



	Accuracy				
Network	**Dense FP16**	**Sparse FP16**		**Sparse INT8**	
MaskRCNN-RN50	37.9	37.9	-	37.8	-0.1
SSD-RN50	24.8	24.8	-	24.9	0.1
FasterRCNN-RN50-FPN-1x	37.6	38.6	1.0	38.4	0.8
FasterRCNN-RN50-FPN-3x	39.8	39.9	-0.1	39.4	-0.4
FasterRCNN-RN101-FPN-3x	41.9	42.0	0.1	41.8	-0.1
MaskRCNN-RN50-FPN-1x	39.9	40.3	0.4	40.0	0.1
MaskRCNN-RN50-FPN-3x	40.6	40.7	0.1	40.4	0.2
MaskRCNN-RN101-FPN-3x	42.9	43.2	0.3	42.8	0.1
RetinaNet-RN50-FPN-1x	36.4	37.4	1.0	37.2	0.8
RPN-RN50-FPN-1x	45.8	45.6	-0.2	45.5	0.3

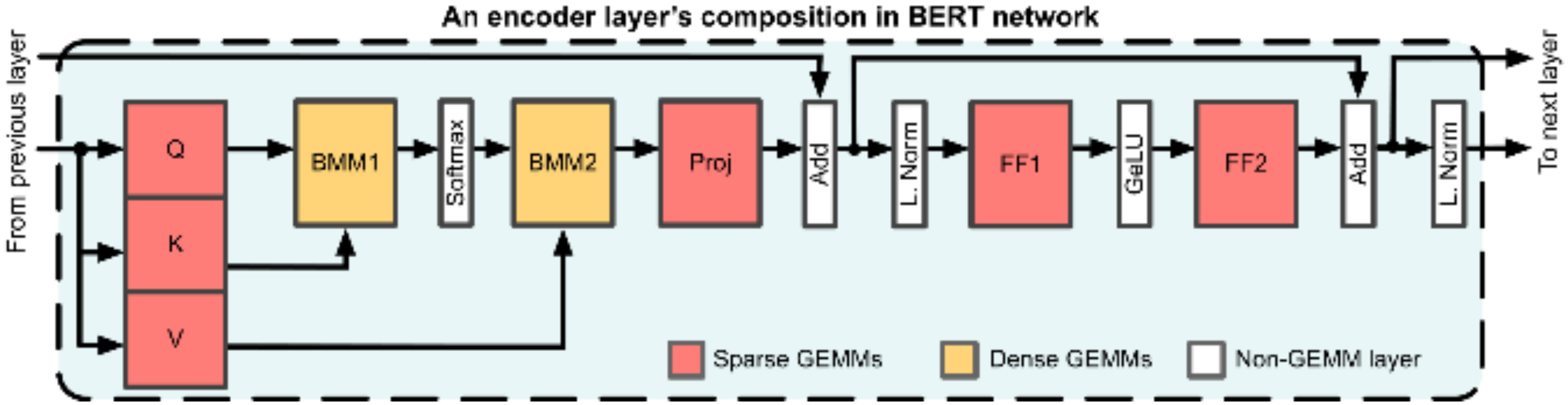
RN = ResNet Backbone
FPN = Feature Pyramid Network
RPN = Region Proposal Network

IMPACT ON NLP

NETWORK PERFORMANCE

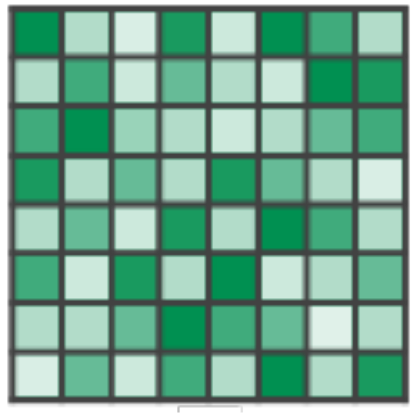
BERT-Large

1.8x GEMM Performance -> 1.5x Network Performance
Some operations remain dense:
Non-GEMM layers (Softmax, Residual add, Normalization, Activation functions, …)
GEMMs without weights to be pruned – Attention Batched Matrix Multiplies



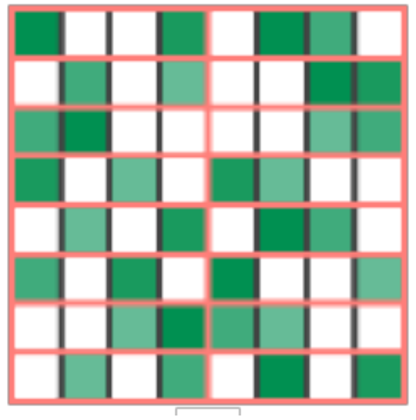
TRAINING RECIPE

RECIPE FOR 2:4 SPARSE NETWORK TRAINING



Dense weights

1) Train (or obtain) a dense network

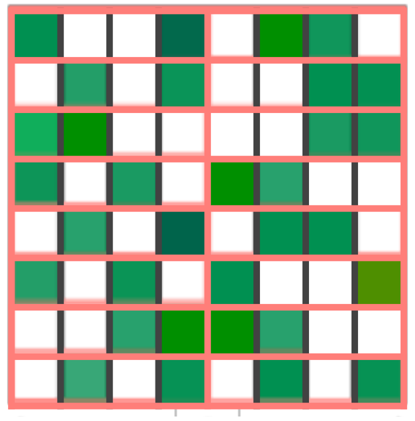


2:4 sparse weights

2) Prune for 2:4 sparsity

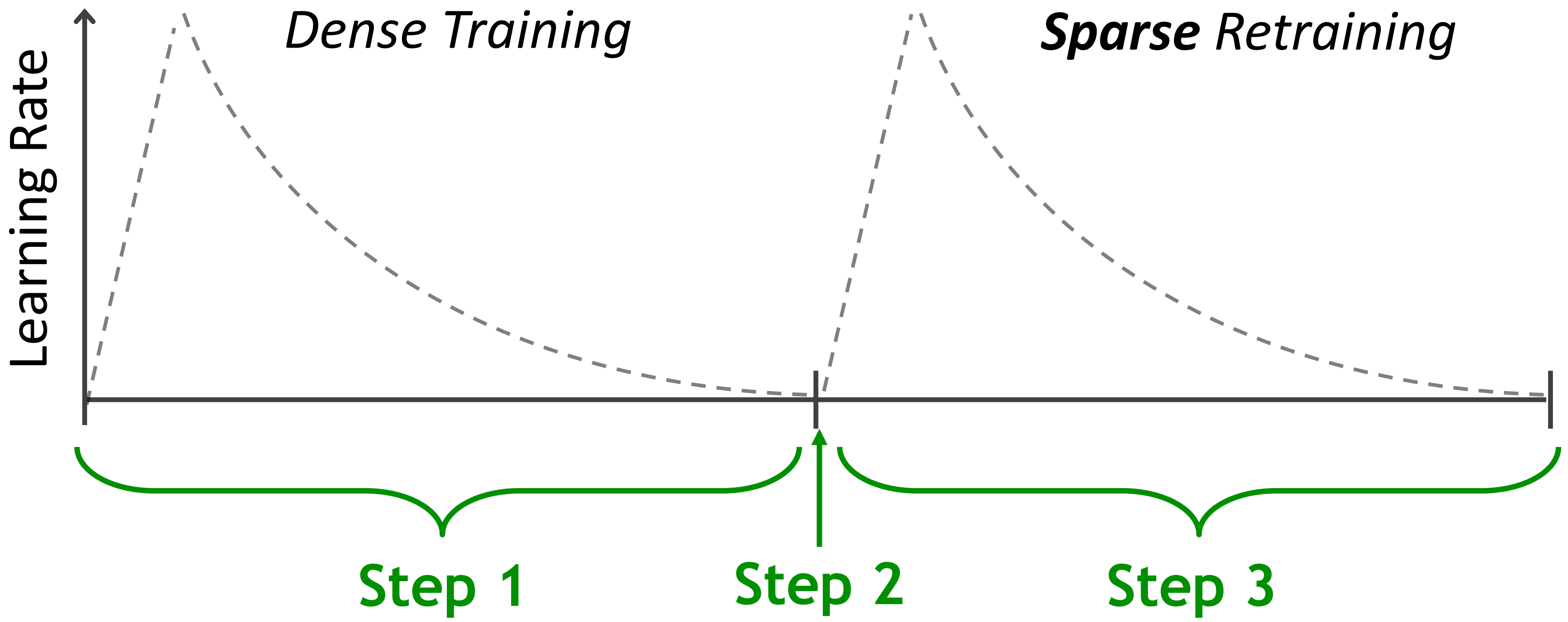
3) Repeat the original training procedure

- Same hyper-parameters as in step-1
- Initialize to weights from step-2
- Maintain the 0 pattern from step-2: no need to recompute the mask



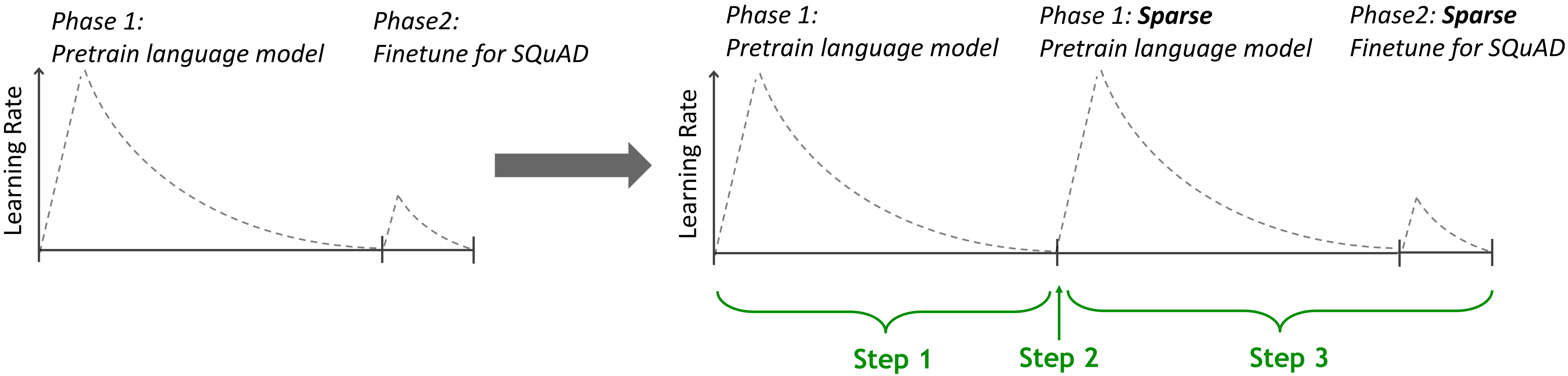
Retrained 2:4 sparse weights

EXAMPLE LEARNING RATE SCHEDULE



BERT SQUAD EXAMPLE

SQuAD Dataset and fine-tuning is too small to compensate for pruning on its own



Phase 1:
Pretrain language model

Phase2:
Finetune for SQuAD

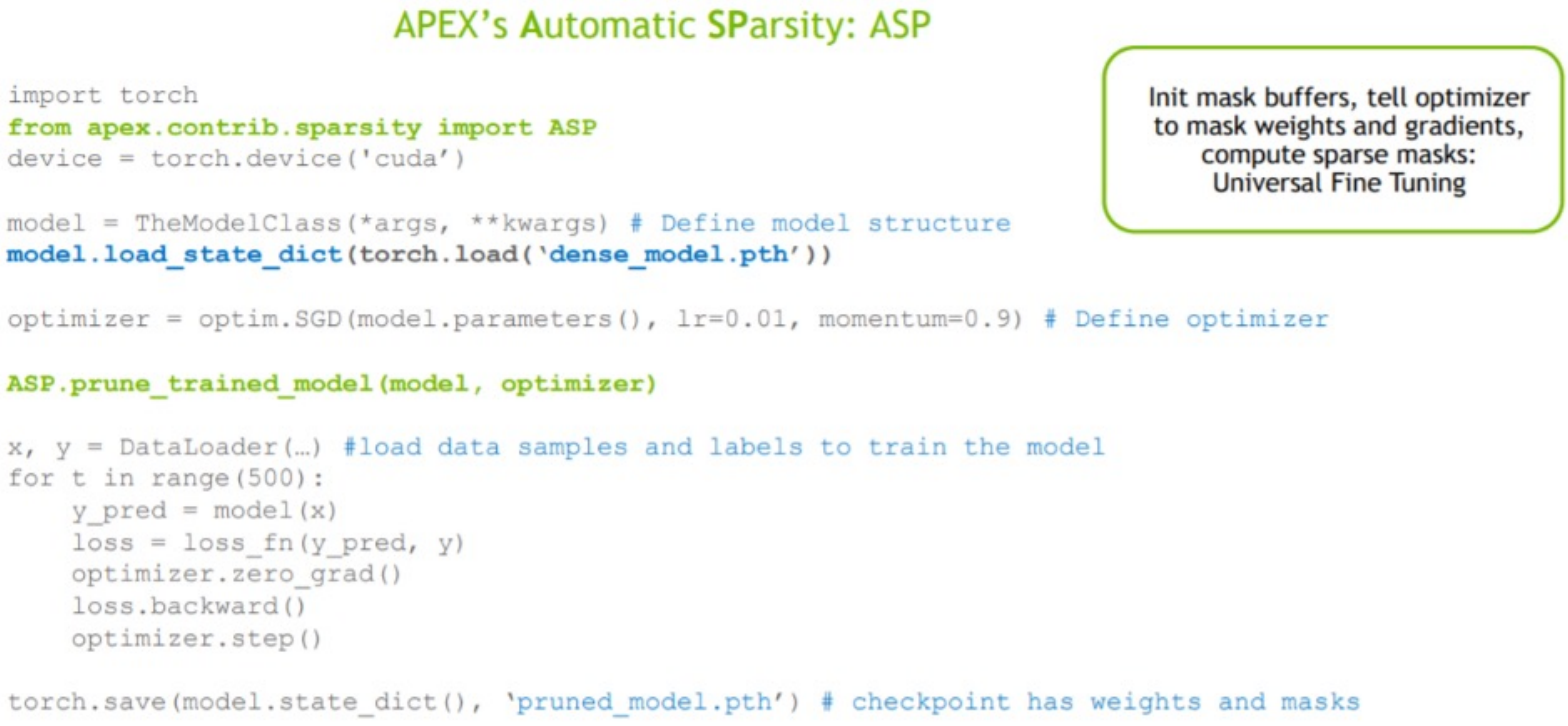
Phase 1:
Pretrain language model

*Phase 1: **Sparse***
Pretrain language model

*Phase2: **Sparse***
Finetune for SQuAD

APEX: AUTOMATIC SPARSITY

TAKING ADVANTAGE OF STRUCTURED SPARSITY



APEX’s Automatic SParsity: ASP

PyTorch sparse fine-tuning loop

```
import torch
from apex.contrib.sparsity import ASP
device = torch.device('cuda’)

model = TheModelClass(*args, **kwargs) # Define model structure
model.load_state_dict(torch.load(‘dense_model.pth’))

optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9) # Define optimizer

ASP.prune_trained_model(model, optimizer)

x, y = DataLoader(…) #load data samples and labels to train the model
for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

torch.save(model.state_dict(), ‘pruned_model.pth’) # checkpoint has weights and masks
```

Init mask buffers, tell optimizer
to mask weights and gradients,
compute sparse masks:
Universal Fine Tuning

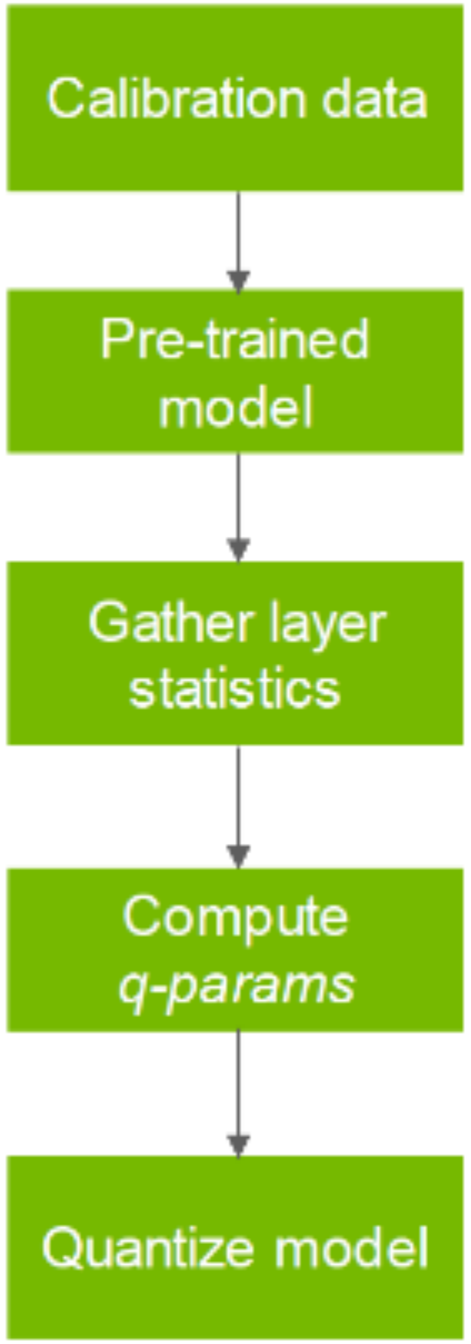
Part 3: Production Deployment

- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model

QUANTIZATION

Approaches

Post-training quantization(PTQ)


Calibration data

Pre-trained model

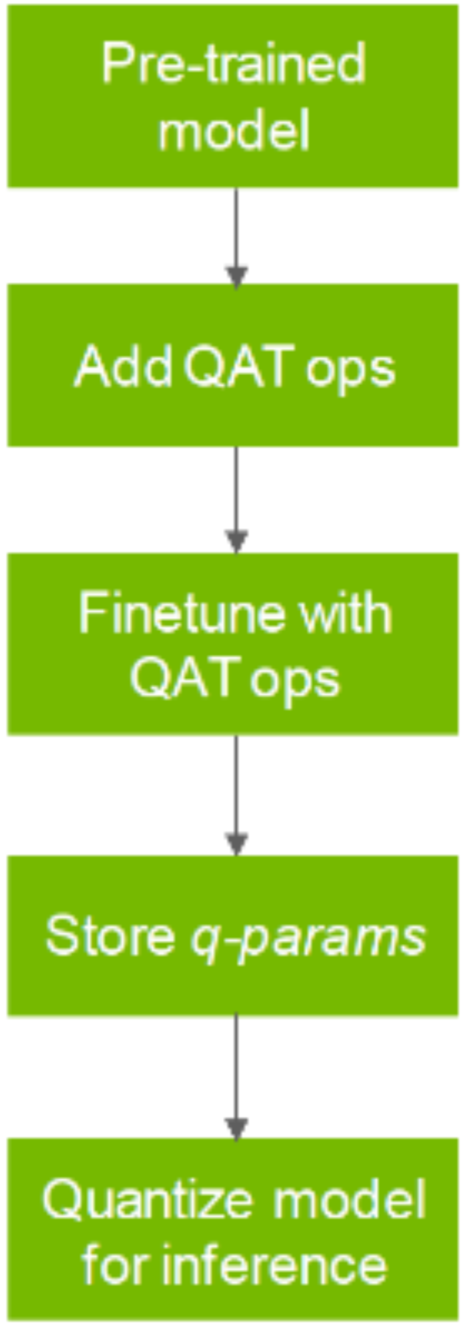
Gather layer statistics

Compute *q-params*

Quantize model

PTQ	QAT
Usually fast	Slow
No re-training of the model	Model needs to be trained/finetuned
Plug and play of quantization schemes	Plug and play of quantization schemes (requires re-training)
Less control over final accuracy of the model	More control over final accuracy since *q-params* are learned during training.

Quantization-aware training (QAT)


Pre-trained model

Add QAT ops

Finetune with QAT ops

Store *q-params*

Quantize model for inference

EXTREME MODEL COMPRESSION

Training with quantization noise

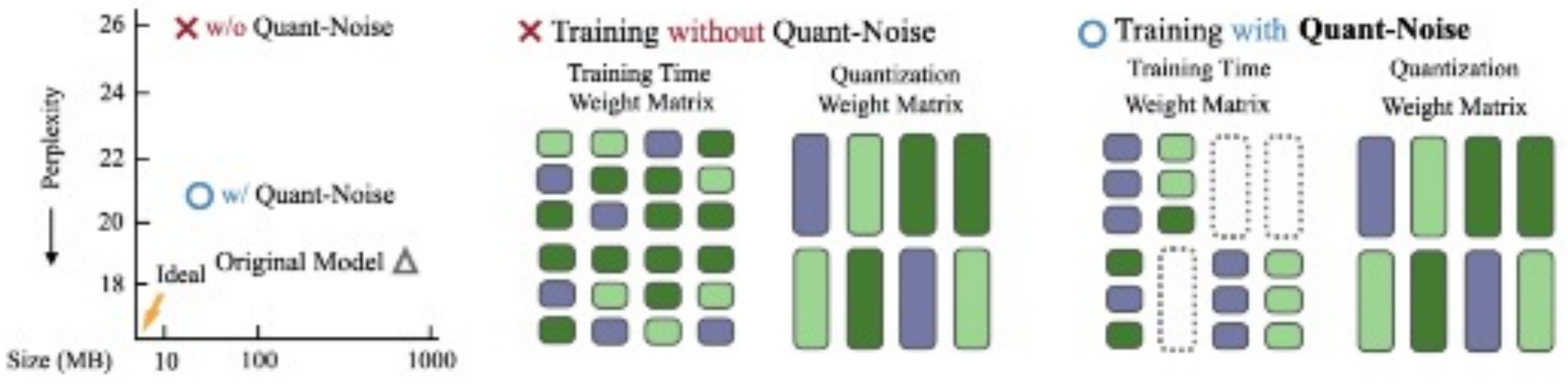
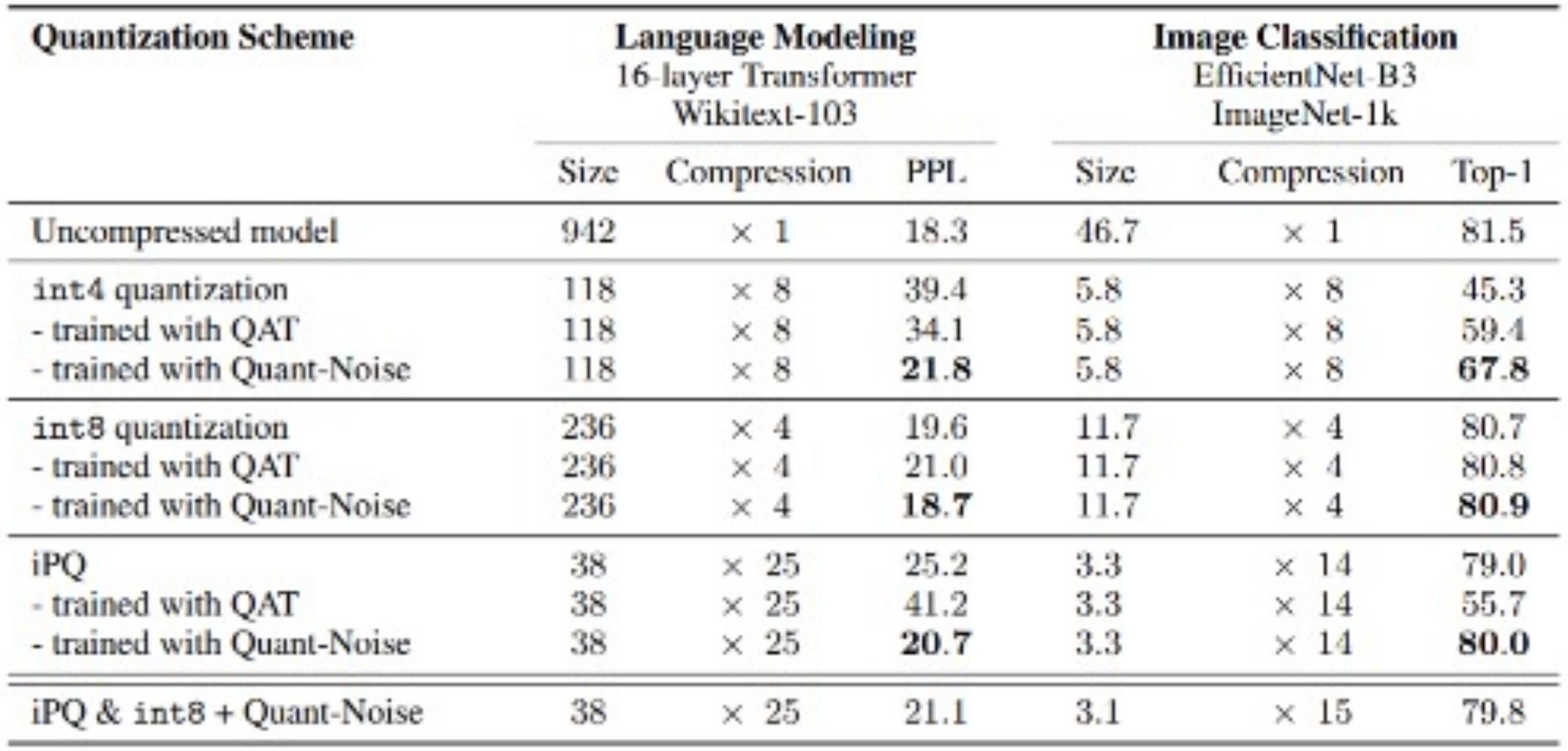

✕ w/o Quant-Noise
○ w/ Quant-Noise
Ideal
Original Model △
Perplexity
26
24
22
20
18
Size (MB)
10
100
1000
✕ Training without Quant-Noise
Training Time Weight Matrix
Quantization Weight Matrix
○ Training with Quant-Noise
Training Time Weight Matrix
Quantization Weight Matrix

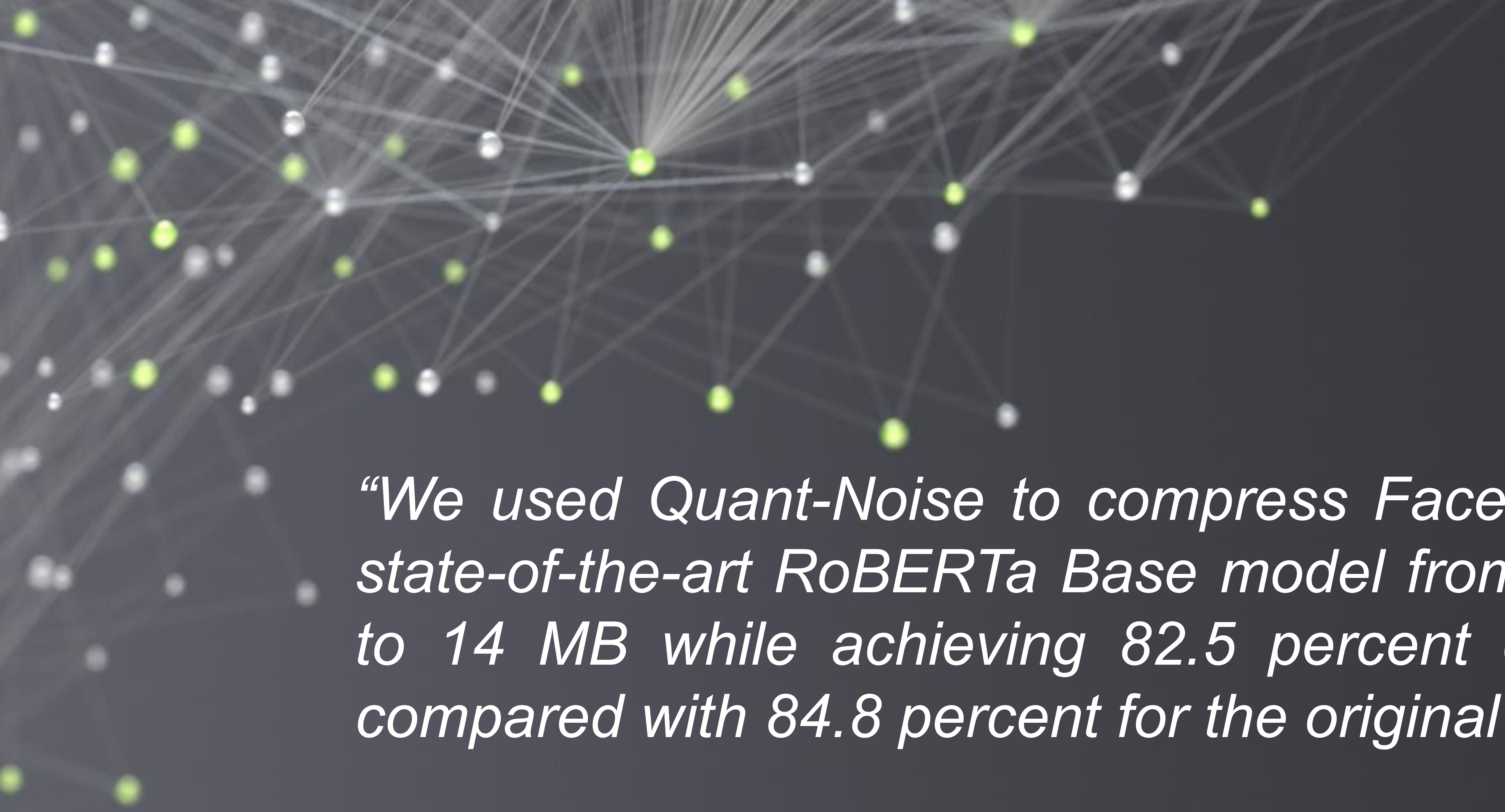
Figure 1: **Quant-Noise** trains models to be resilient to inference-time quantization by mimicking the effect of the quantization method during training time. This allows for extreme compression rates without much loss in accuracy on a variety of tasks and benchmarks.



Quantization Scheme	**Language Modeling** 16-layer Transformer Wikitext-103			**Image Classification** EfficientNet-B3 ImageNet-1k		
	Size	Compression	PPL	Size	Compression	Top-1
Uncompressed model	942	× 1	18.3	46.7	× 1	81.5
`int4` quantization	118	× 8	39.4	5.8	× 8	45.3
- trained with QAT	118	× 8	34.1	5.8	× 8	59.4
- trained with Quant-Noise	118	× 8	**21.8**	5.8	× 8	**67.8**
`int8` quantization	236	× 4	19.6	11.7	× 4	80.7
- trained with QAT	236	× 4	21.0	11.7	× 4	80.8
- trained with Quant-Noise	236	× 4	**18.7**	11.7	× 4	**80.9**
iPQ	38	× 25	25.2	3.3	× 14	79.0
- trained with QAT	38	× 25	41.2	3.3	× 14	55.7
- trained with Quant-Noise	38	× 25	**20.7**	3.3	× 14	**80.0**
iPQ & `int8` + Quant-Noise	38	× 25	21.1	3.1	× 15	79.8

Table 1: **Comparison of different quantization schemes with and without Quant-Noise** on language modeling and image classification. For language modeling, we train a Transformer on the Wikitext-103 benchmark and report perplexity (PPL) on test. For image classification, we train a EfficientNet-B3 on the ImageNet-1k benchmark and report top-1 accuracy on validation and use our re-implementation of EfficientNet-B3. The original implementation of Tan *et al.* [4] achieves an uncompressed Top-1 accuracy of 81.9%. For both settings, we report model size in megabyte (MB) and the compression ratio compared to the original model.

Polino, A., Pascanu, R., & Alistarh, D. (2018). Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.



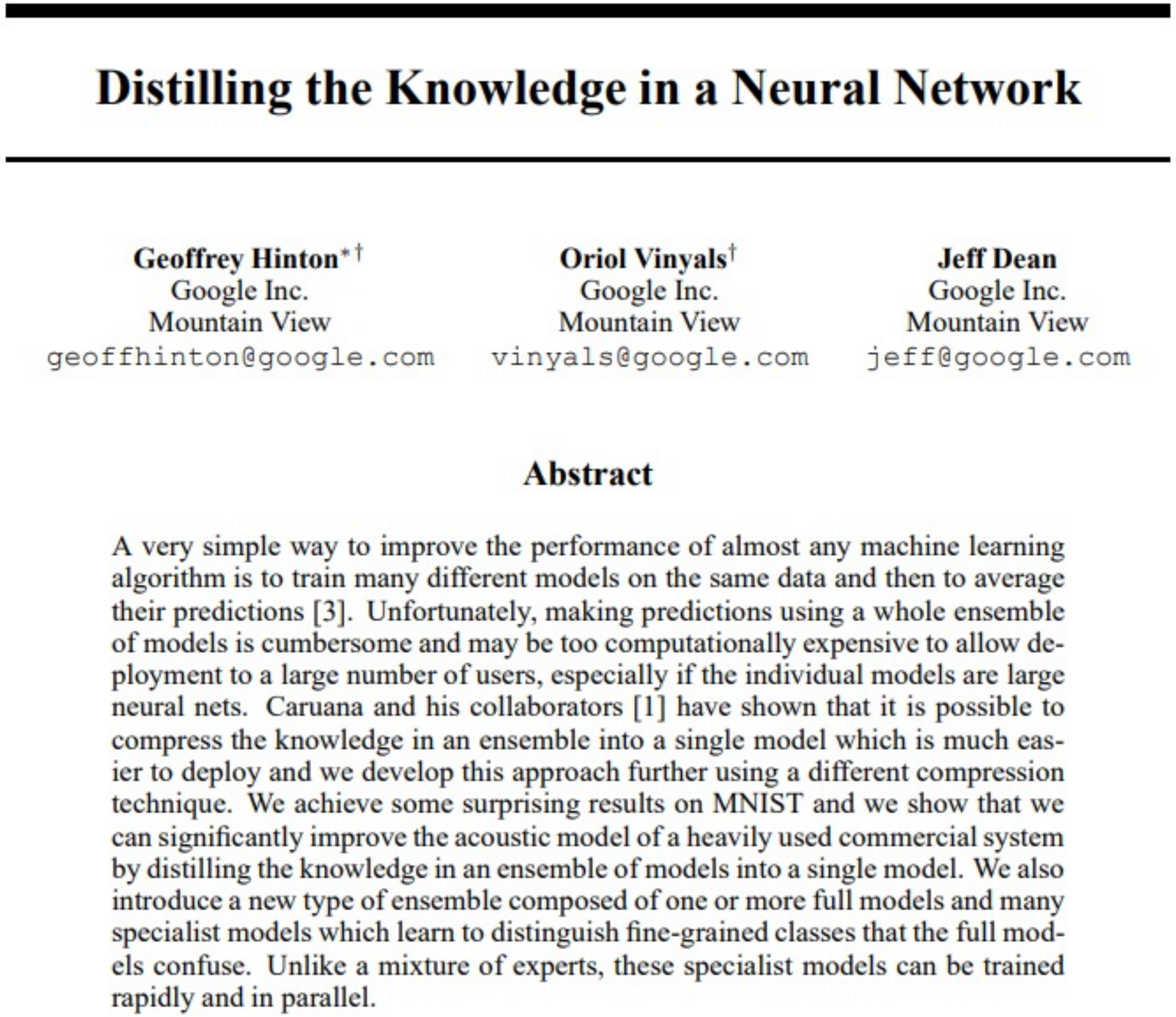
“We used Quant-Noise to compress Facebook AI’s state-of-the-art RoBERTa Base model from 480 MB to 14 MB while achieving 82.5 percent on MNLI, compared with 84.8 percent for the original model.”

Part 3: Production Deployment

- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model

KNOWLEDGE DISTILLATION

The idea



Distilling the Knowledge in a Neural Network

Geoffrey Hinton[*][†]
Google Inc.
Mountain View
geoffhinton@google.com

Oriol Vinyals[†]
Google Inc.
Mountain View
vinyals@google.com

Jeff Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and then to average their predictions [3]. Unfortunately, making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural nets. Caruana and his collaborators [1] have shown that it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy and we develop this approach further using a different compression technique. We achieve some surprising results on MNIST and we show that we can significantly improve the acoustic model of a heavily used commercial system by distilling the knowledge in an ensemble of models into a single model. We also introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

KNOWLEDGE DISTILLATION

DistillBERT

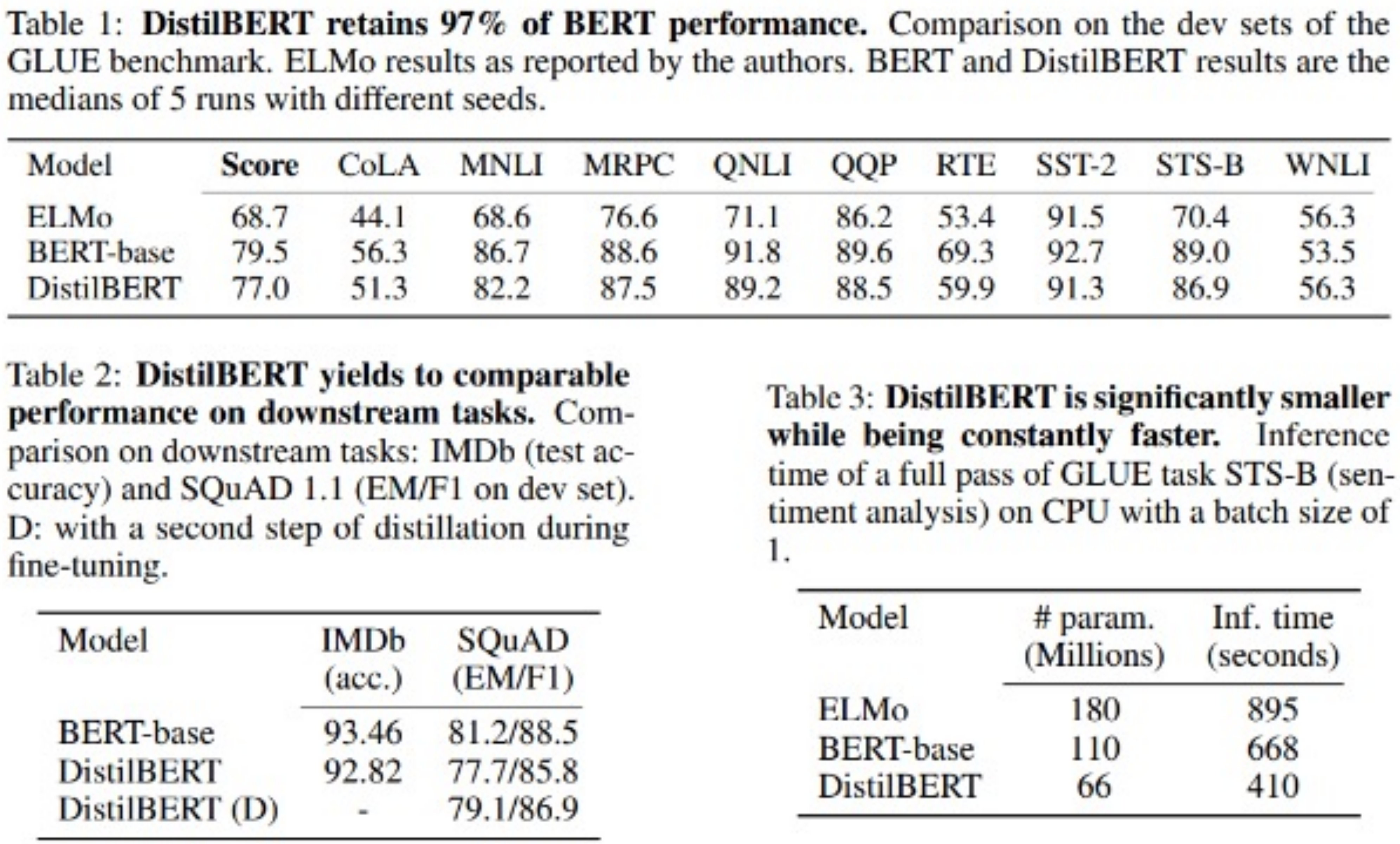


Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Table 2: **DistilBERT yields to comparable performance on downstream tasks.** Comparison on downstream tasks: IMDb (test accuracy) and SQuAD 1.1 (EM/F1 on dev set). D: with a second step of distillation during fine-tuning.

Model	IMDb (acc.)	SQuAD (EM/F1)
BERT-base	93.46	81.2/88.5
DistilBERT	92.82	77.7/85.8
DistilBERT (D)	-	79.1/86.9

Table 3: **DistilBERT is significantly smaller while being constantly faster.** Inference time of a full pass of GLUE task STS-B (sentiment analysis) on CPU with a batch size of 1.

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

Part 3: Production Deployment

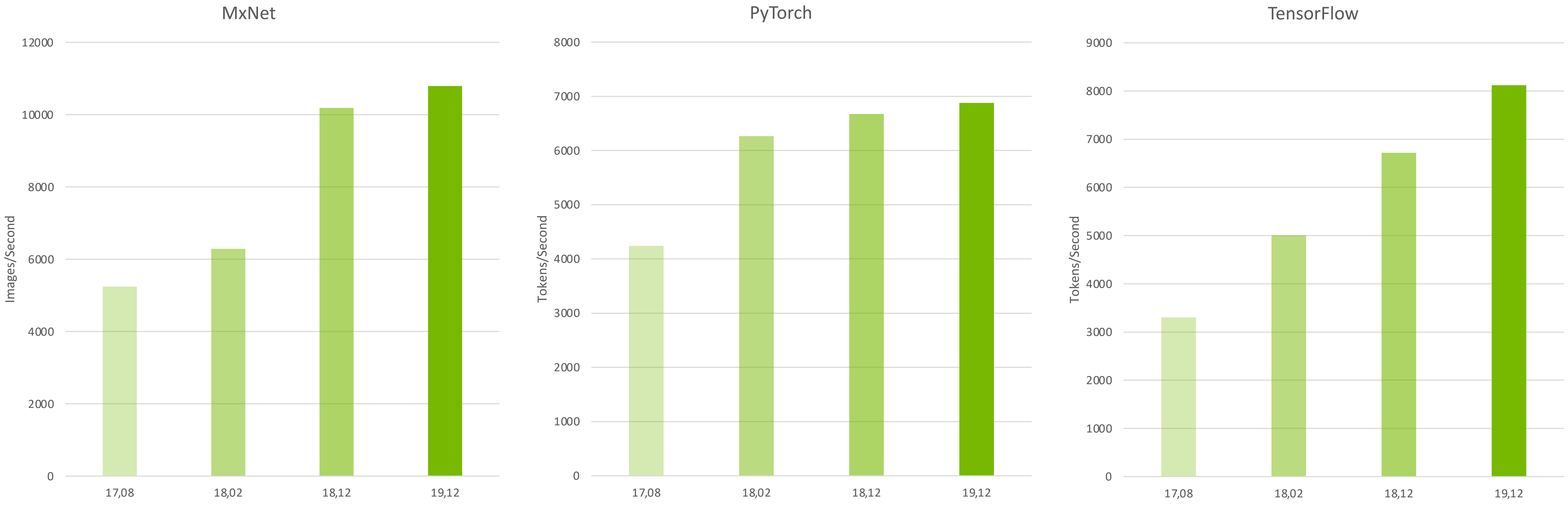
- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model

NOT ALL MODELS HAVE THE SAME CODE QUALITY

COMPUTE MATTERS

But so does code quality

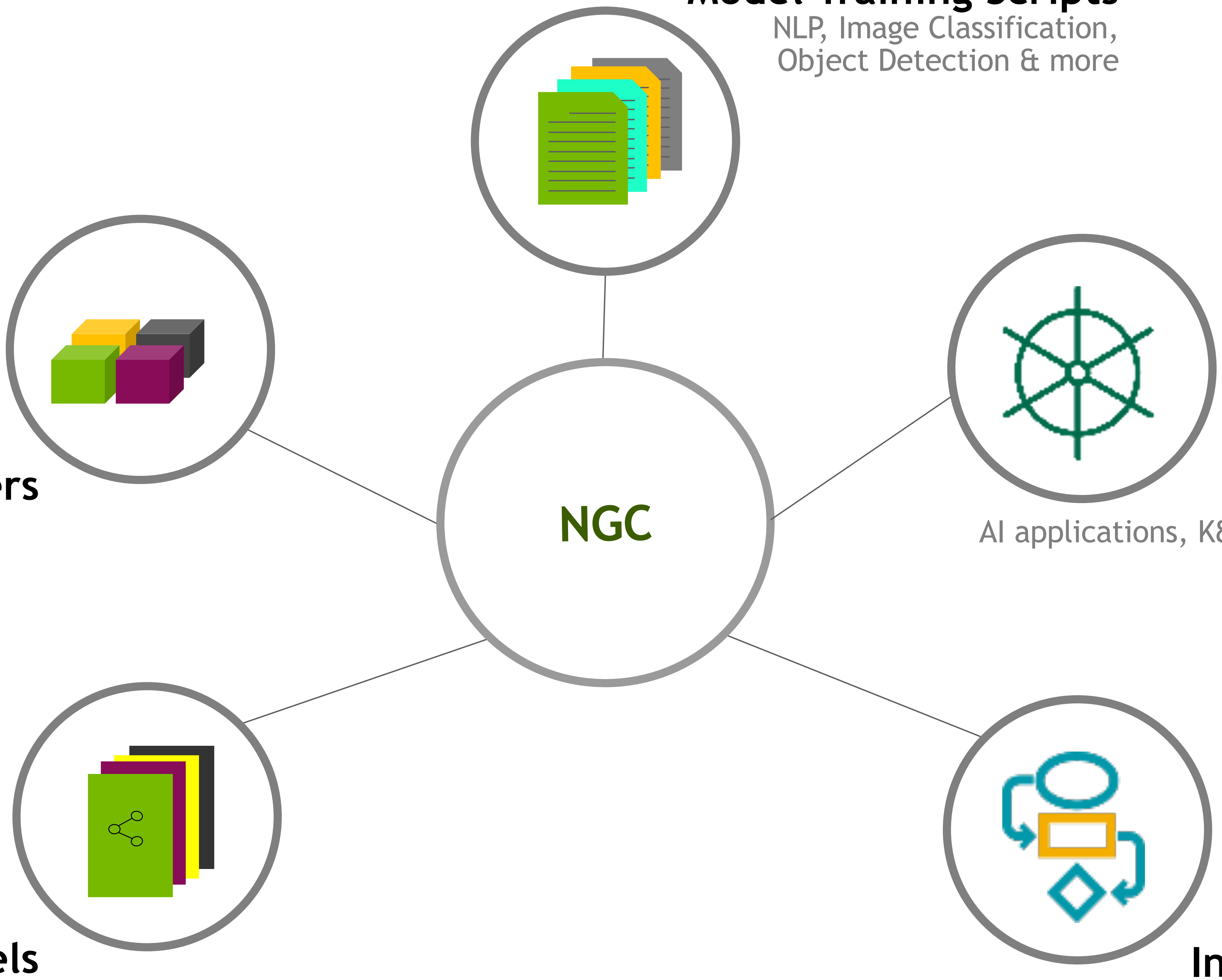
Monthly DL Framework Updates & Optimizations Drive Performance



ResNet-50 v1.5 Training | 8x V100 | DGX-1

NGC: GPU-OPTIMIZED SOFTWARE HUB

Simplifying DL, ML and HPC Workflows



NGC

Model Training Scripts
NLP, Image Classification,
Object Detection & more

Containers
DL, ML, HPC

Helm Charts
AI applications, K8s cluster, Registry

Pre-trained Models
NLP, Classification, Object Detection & more

Industry SDKs
Medical Imaging, Intelligent Video Analytics

PRETRAINED MODELS & MODEL SCRIPTS

Build AI Solutions Faster

PRE-TRAINED MODELS

- Deploy AI quickly with models for industry specific use cases
- Covers everything from speech to object detection
- Integrate into existing workflows with code samples
- Easily use transfer learning to adapt to your bespoke use case

MODEL SCRIPTS

- Reference neural network architectures across all domains and popular frameworks with latest SOTA
- Jupyter notebook starter kits

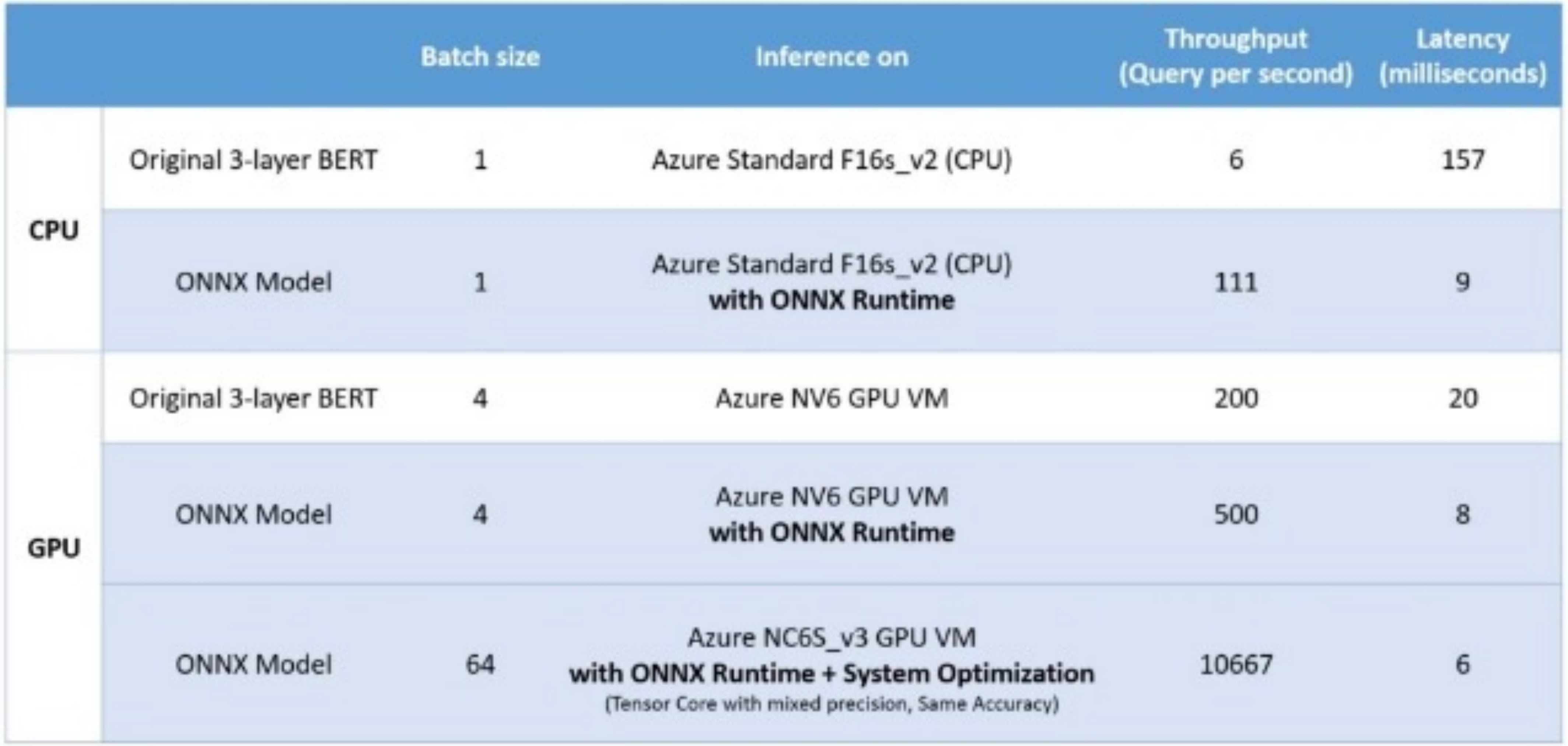
Healthcare (~30 models)	BioBERT (NLP), Clara (Computer Vision)
Manufacturing (~25 Models)	Object Detection, Image Classification
Retail (~25 models)	BERT, Transformer
70 TensorRT Plans	Classification/Segmentation for v5, v6, v7
Natural Language Processing	25 Bert Configurations
Recommendation Engines	Neural Collaborative Filtering, VAE
Speech	Jasper, Tacotron, WaveGlow
Translation	GNMT

THIS APPLIES NOT ONLY TO TRAINING BUT INFERENCE AS WELL

CODE QUALITY IS KEY

Dramatic differences in model performance

3-layer BERT with 128 sequence length



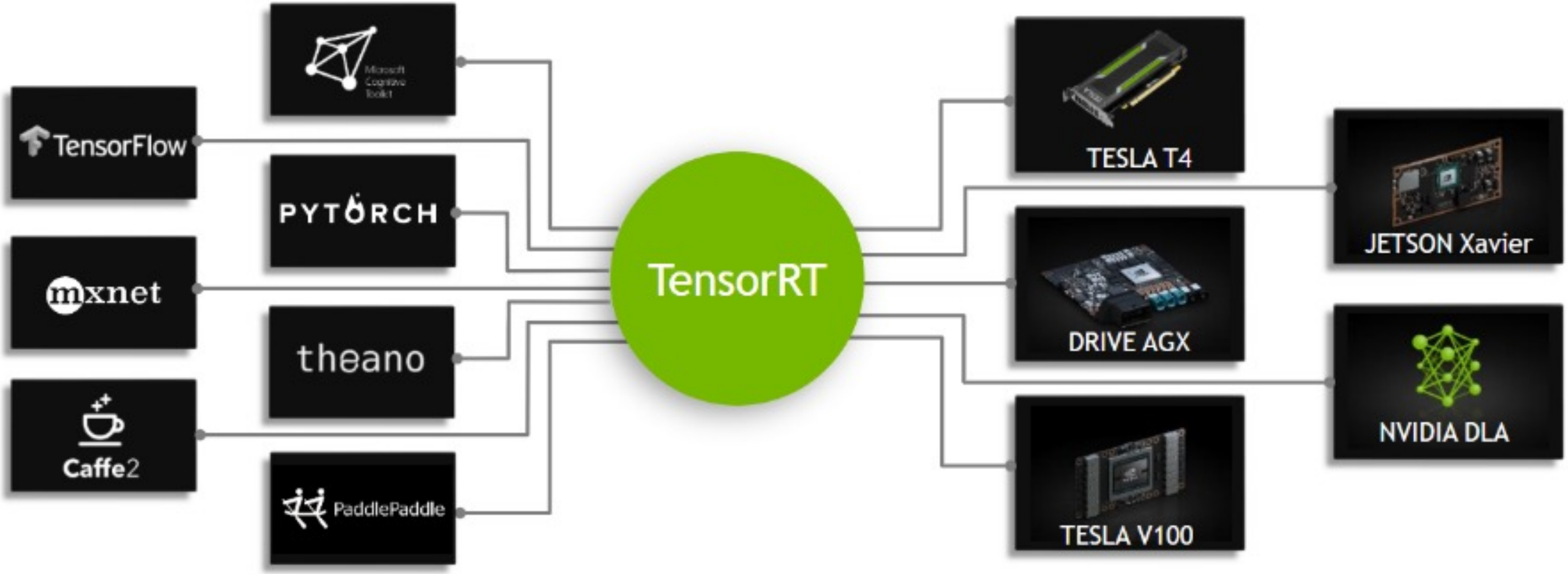
		Batch size	Inference on	Throughput (Query per second)	Latency (milliseconds)
CPU	Original 3-layer BERT	1	Azure Standard F16s_v2 (CPU)	6	157
	ONNX Model	1	Azure Standard F16s_v2 (CPU) **with ONNX Runtime**	111	9
GPU	Original 3-layer BERT	4	Azure NV6 GPU VM	200	20
	ONNX Model	4	Azure NV6 GPU VM **with ONNX Runtime**	500	8
	ONNX Model	64	Azure NC6S_v3 GPU VM **with ONNX Runtime + System Optimization** (Tensor Core with mixed precision, Same Accuracy)	10667	6

https://cloudblogs.microsoft.com/opensource/2020/01/21/microsoft-onnx-open-source-optimizations-transformer-inference-gpu-cpu

OPTIMIZING INFERENCE WITH TENSORRT

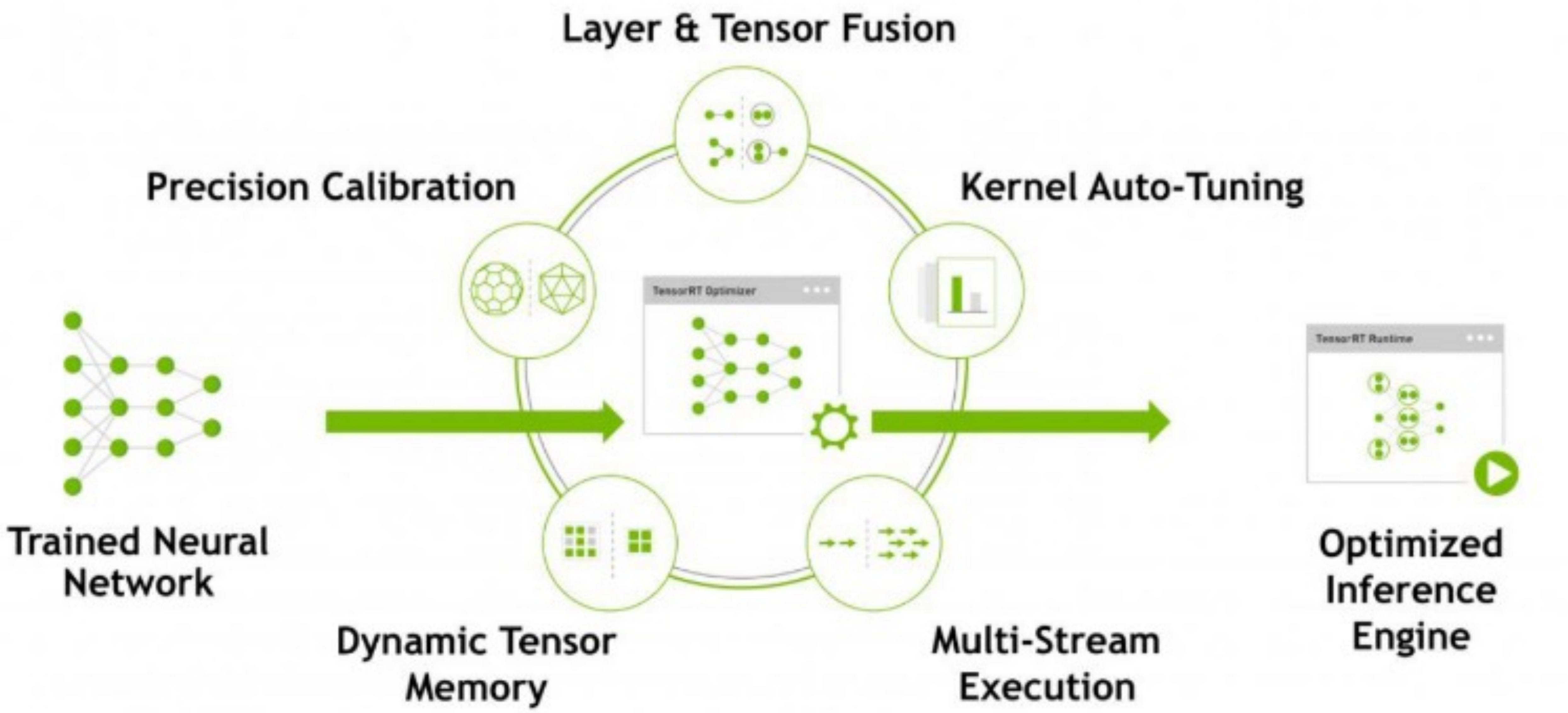
NVIDIA TENSORRT

From Every Framework, Optimized For Each Target Platform



TENSORRT

Optimizations



developer.nvidia.com/tensorrt

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

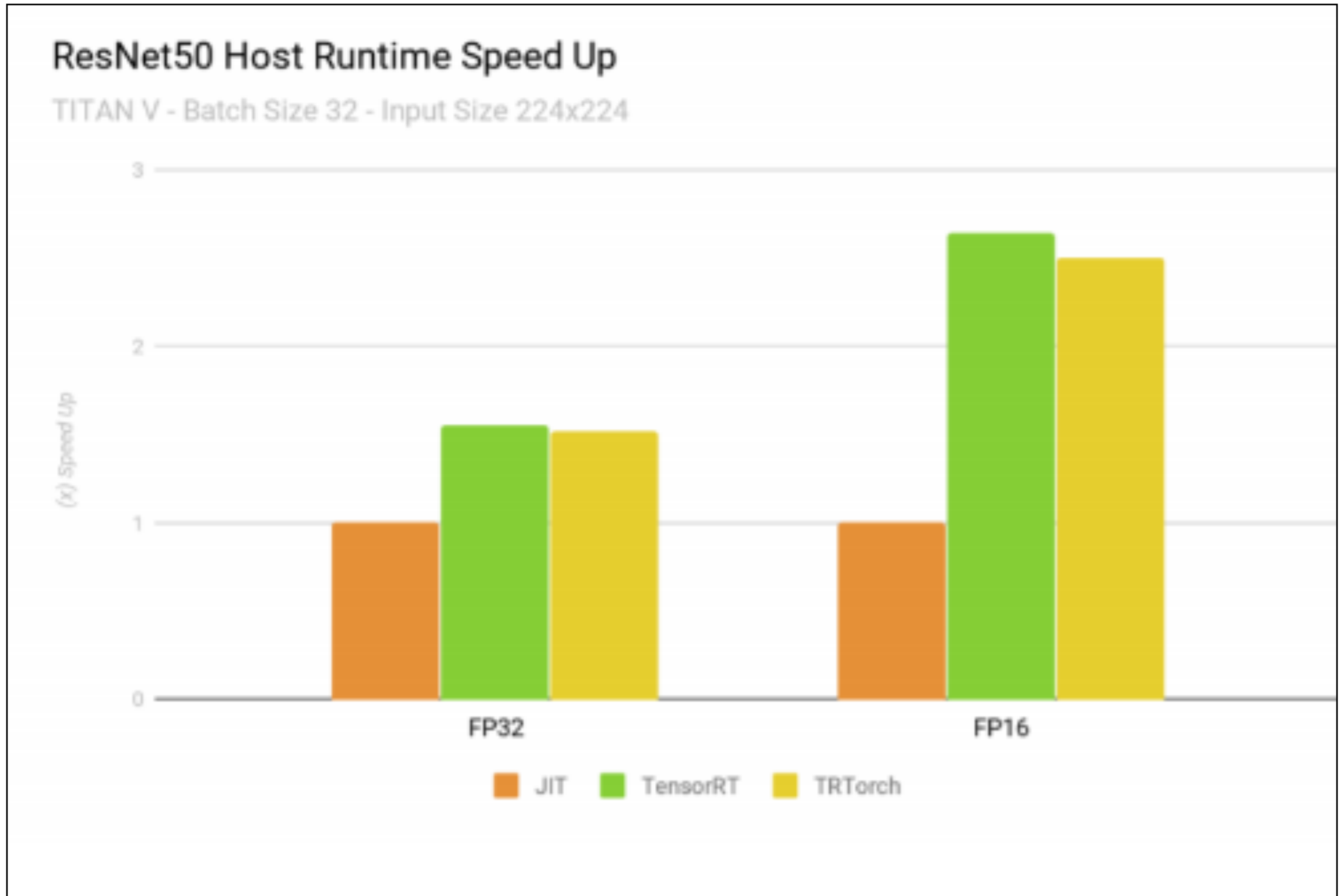
Support for Opset 11 (See List of Supported Ops)

ONNX

Caffe2
Chainer
Cognitive Toolkit
mxnet
MATLAB
PyTorch
PaddlePaddle

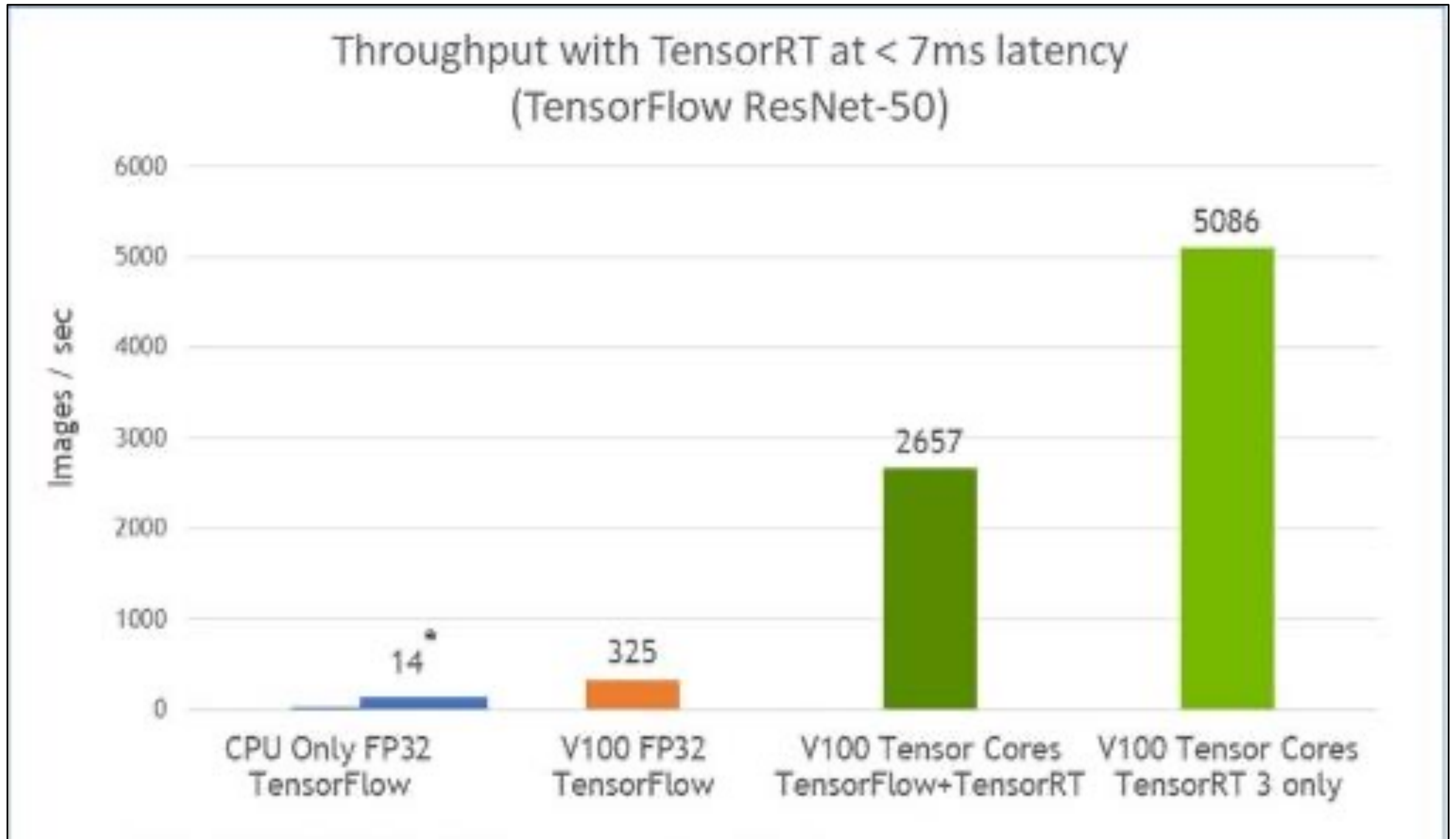
TENSORRT

Tight integration with DL frameworks



PyTorch 1.4.0 (CuDNN Benchmark mode enabled) CUDA 10.1 TensorRT 6.0.1.5, TITAN V, i7-7800X

Pytorch -> TRTorch



Updated 3/28/2018. * Min CPU latency measured was 70 ms. It is not < 7ms.
CPU: Skylake Gold 6140, Ubuntu 16.04, 18 CPU threads. Volta V100 SXM; CUDA (384.111;V9.0.176);
Batch sizes: CPU=1;V100_FP32=2; V100_TensorFlow_TensorRT=16; V100_TensorRT=32; Latency=6ms. TensorRT 3.
Latest results at: https://developer.nvidia.com/deep-learning-performance-training-inference

TensorFlow -> TF-TRT

WIDELY ADOPTED

Accelerating most demanding applications

Alibaba Group 阿里巴巴集团
Baidu 百度
BOSCH
ByteDance 字节跳动
criteo

Ford
IBM
JD.COM
KLA Keep Looking Ahead
金山云

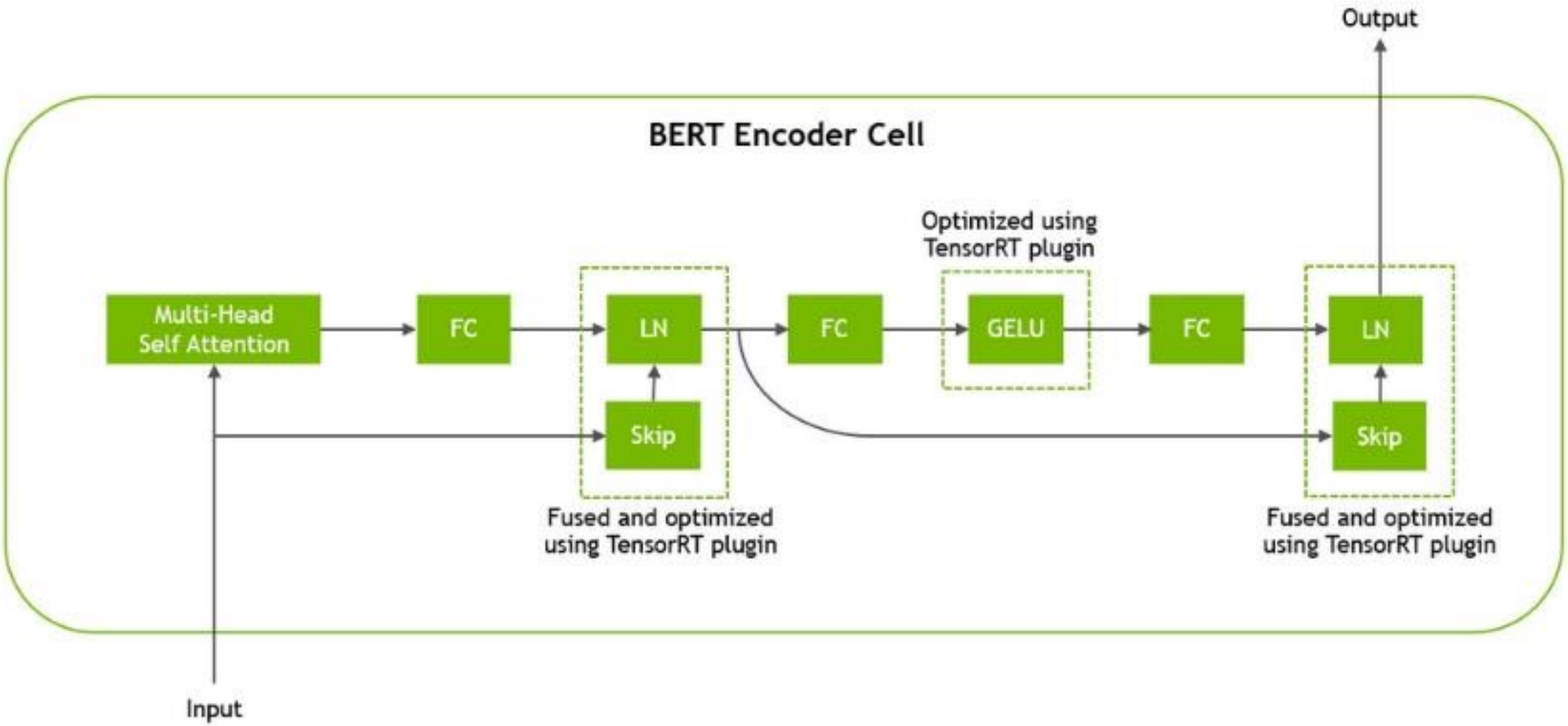
MAN
SEARIDGE TECHNOLOGIES
SK telecom
Tencent 腾讯
图森 tu Simple

verizon
VOLVO
欢聚时代 YY Inc.
ZF

IMPACT ON NLP

TENSORRT

BERT Encoder optimizations


Output
BERT Encoder Cell
Optimized using TensorRT plugin
Multi-Head Self Attention
FC
LN
FC
GELU
FC
LN
Skip
Skip
Fused and optimized using TensorRT plugin
Fused and optimized using TensorRT plugin
Input

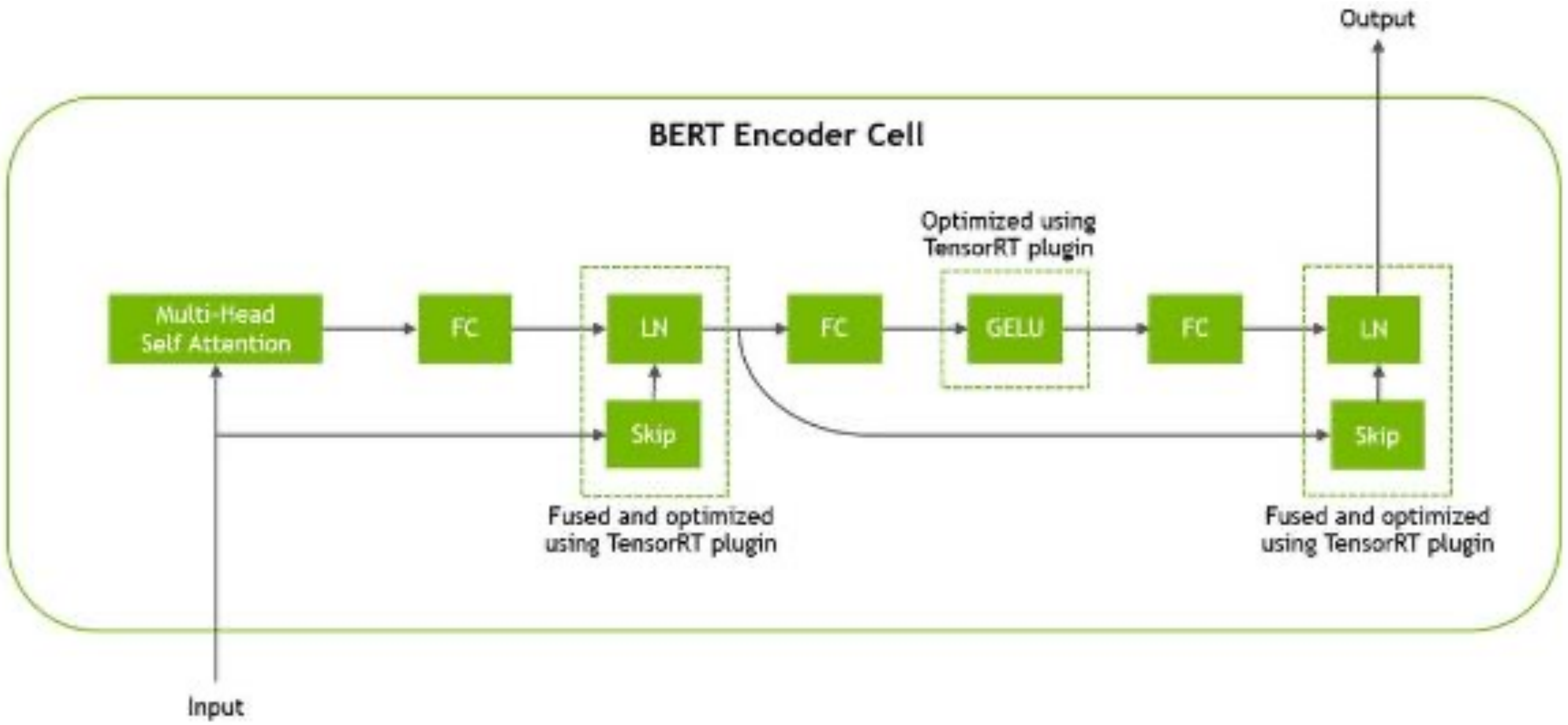
CUSTOM PLUGINS

Optimized GeLU as well as skip and layer-normalization operations

- Naïve implementation would require a large number of TensorRT elementary layers
- For k layers, the naïve implementation would require k-1 memory roundtrips
- The skip and layer-normalization(LN) layers occur twice per Transformer layer and are fused in a single kernel

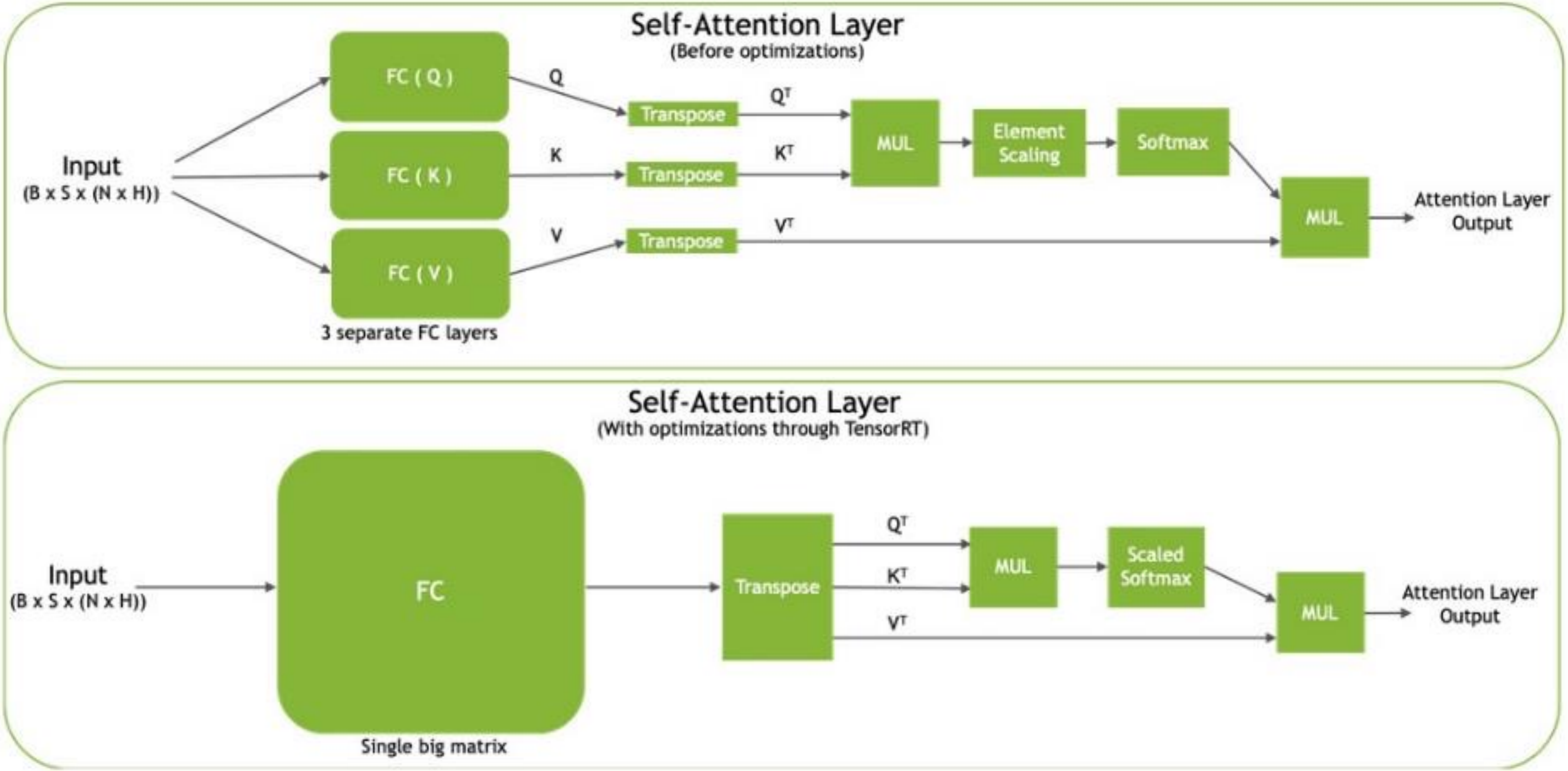
gelu(x) = a * x * (1 + tanh(b * (x + c * x^3)))

```
Result = x^3
Result = c * Result
Result = x + Result
Result = b * Result
Result = tanh(Result)
Result = x * Result
Result = a * Result
```


Output
BERT Encoder Cell
Optimized using TensorRT plugin
Multi-Head Self Attention
FC
LN
FC
GELU
FC
LN
Skip
Skip
Fused and optimized using TensorRT plugin
Fused and optimized using TensorRT plugin
Input

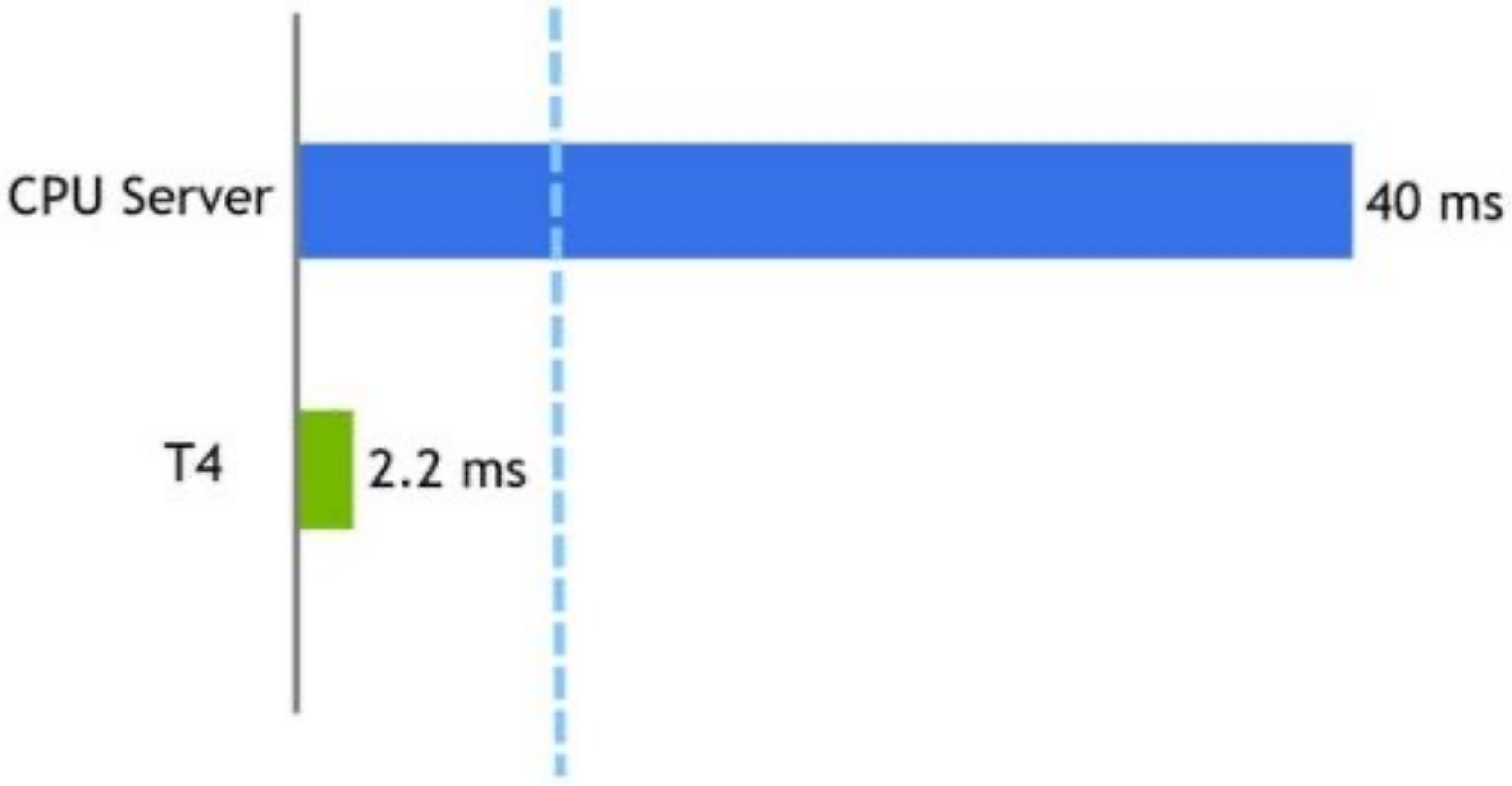
CUSTOM PLUGINS

Self-attention layer



IMPLICATIONS

Significant impact on latency and throughput (batch 1)

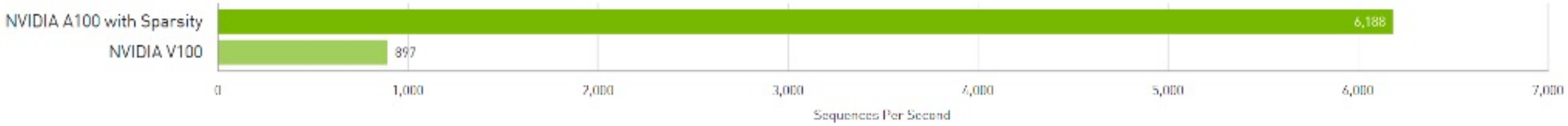


10 milliseconds Target for
Many Conversational AI Apps

Using a Tesla T4 GPU, BERT optimized with TensorRT can perform inference in 2.2 ms for a QA task similar to available in SQuAD with batch size =1 and sequence length = 128.

IMPLICATIONS

Significant impact on latency and throughput



DGX A100 server w/ 1x NVIDIA A100 with 7 MIG instances of 1g.5gb | Batch Size = 94 | Precision: INT8 | Sequence Length = 128
DGX-1 server w/ 1x NVIDIA V100 | TensorRT 7.1 | Batch Size = 256 | Precision: Mixed | Sequence Length = 128

BEYOND BERT

FASTER TRANSFORMER

Designed for training and inference speed

- Encoder:
  - 1.5x compare to TensorFlow with XLA on FP16
- Decoder on NVIDIA Tesla T4
  - 2.5x speedup for batch size 1 (online translating scheme)
  - 2x speedup for large batch size in FP16
- Decoding on NVIDIA Tesla T4
  - 7x speedup for batch size 1 and beam width 4 (online translating scheme)
  - 2x speedup for large batch size in FP16.
- Decoding on NVIDIA Tesla V100
  - 6x speedup for batch size 1 and beam width 4 (online translating scheme)
  - 3x speedup for large batch size in FP16.

https://github.com/NVIDIA/DeepLearningExamples/tree/master/FasterTransformer#feature-support-matrix

CONSIDER USING TENSORRT

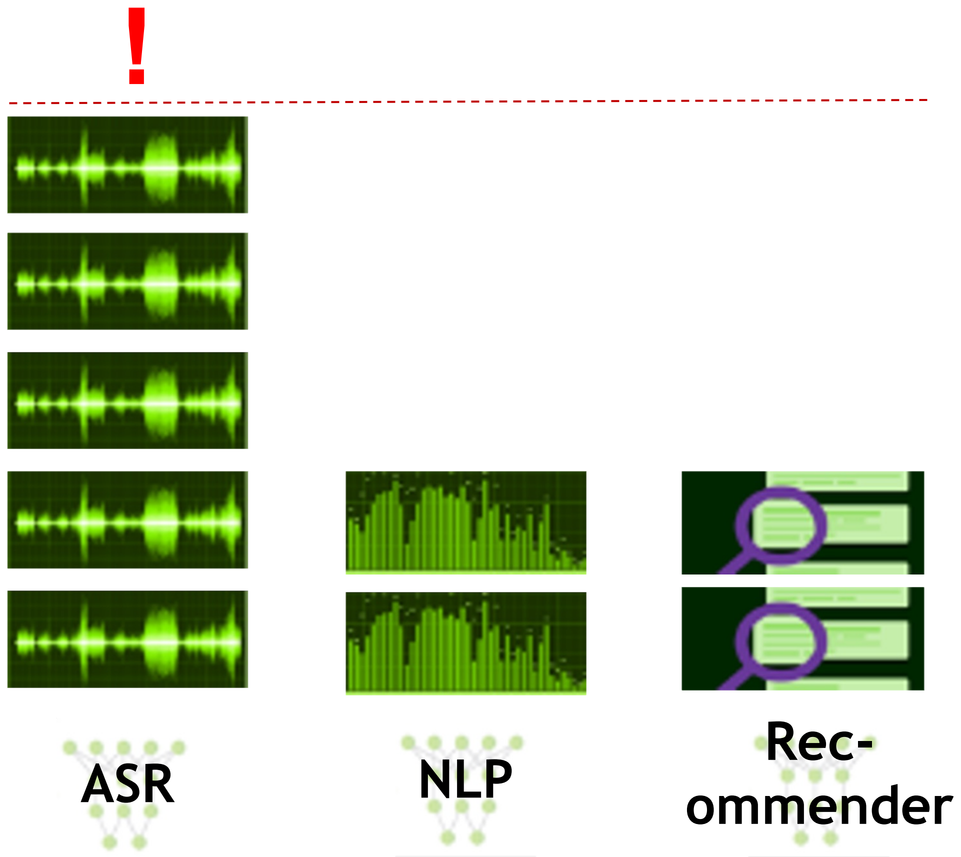
Part 3: Production Deployment

- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model

INEFFICIENCY LIMITS INNOVATION

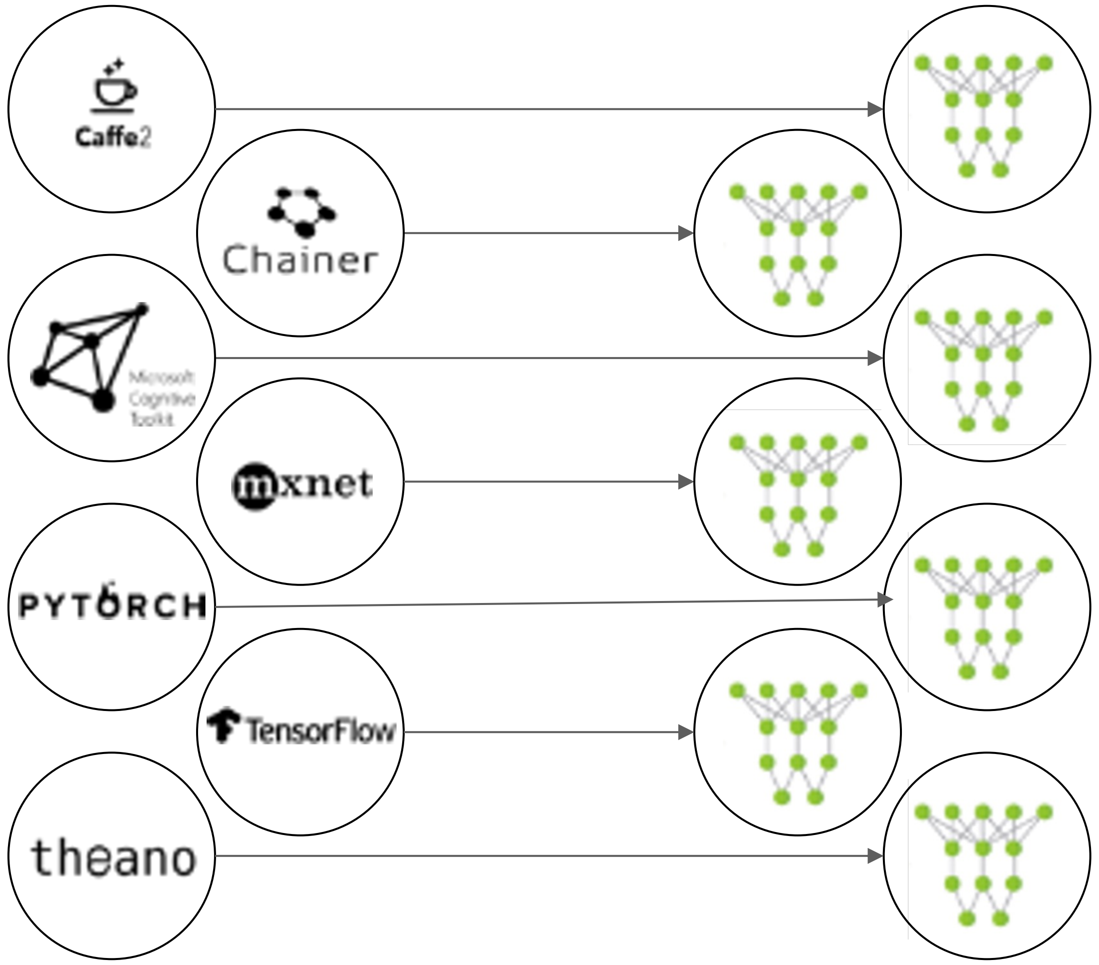
Difficulties with deploying data center inference

Single Model Only



Some systems are overused while others are underutilized

Single Framework Only



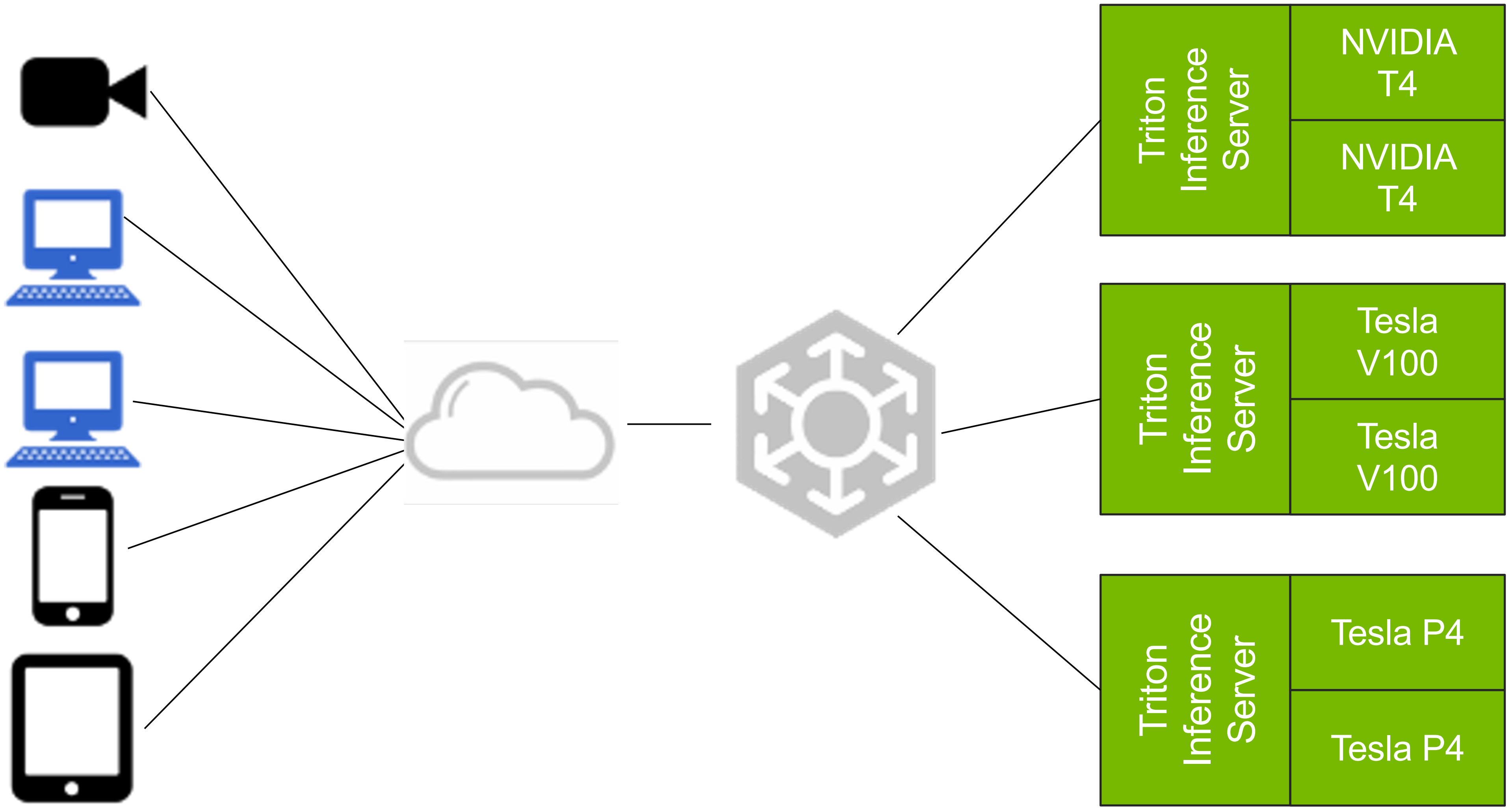
Solutions can only support models from one framework

Custom Development

Developers need to reinvent the plumbing for every application

NVIDIA TRITON INFERENCE SERVER

Production data center inference server



Triton Inference Server

NVIDIA T4

NVIDIA T4

Triton Inference Server

Tesla V100

Tesla V100

Triton Inference Server

Tesla P4

Tesla P4

Maximize real-time inference performance of GPUs

Quickly deploy and manage multiple models per GPU per node

Easily scale to heterogeneous GPUs and multi GPU nodes

Integrates with orchestration systems and auto-scalers via latency and health metrics

Now open source for thorough customization and integration

FEATURES

Concurrent Model Execution

Multiple models (or multiple instances of same model) may execute on GPU simultaneously

CPU Model Inference Execution

Framework native models can execute inference requests on the CPU

Metrics

Utilization, count, memory, and latency

Custom Backend

Custom backend allows the user more flexibility by providing their own implementation of an execution engine through the use of a shared library

Model Ensemble

Pipeline of one or more models and the connection of input and output tensors between those models (can be used with custom backend)

Dynamic Batching

Inference requests can be batched up by the inference server to 1) the model-allowed maximum or 2) the user-defined latency SLA

Multiple Model Format Support

PyTorch JIT (.pt)
TensorFlow GraphDef/SavedModel
TensorFlow and TensorRT GraphDef
ONNX graph (ONNX Runtime)
TensorRT Plans
Caffe2 NetDef (ONNX import path)

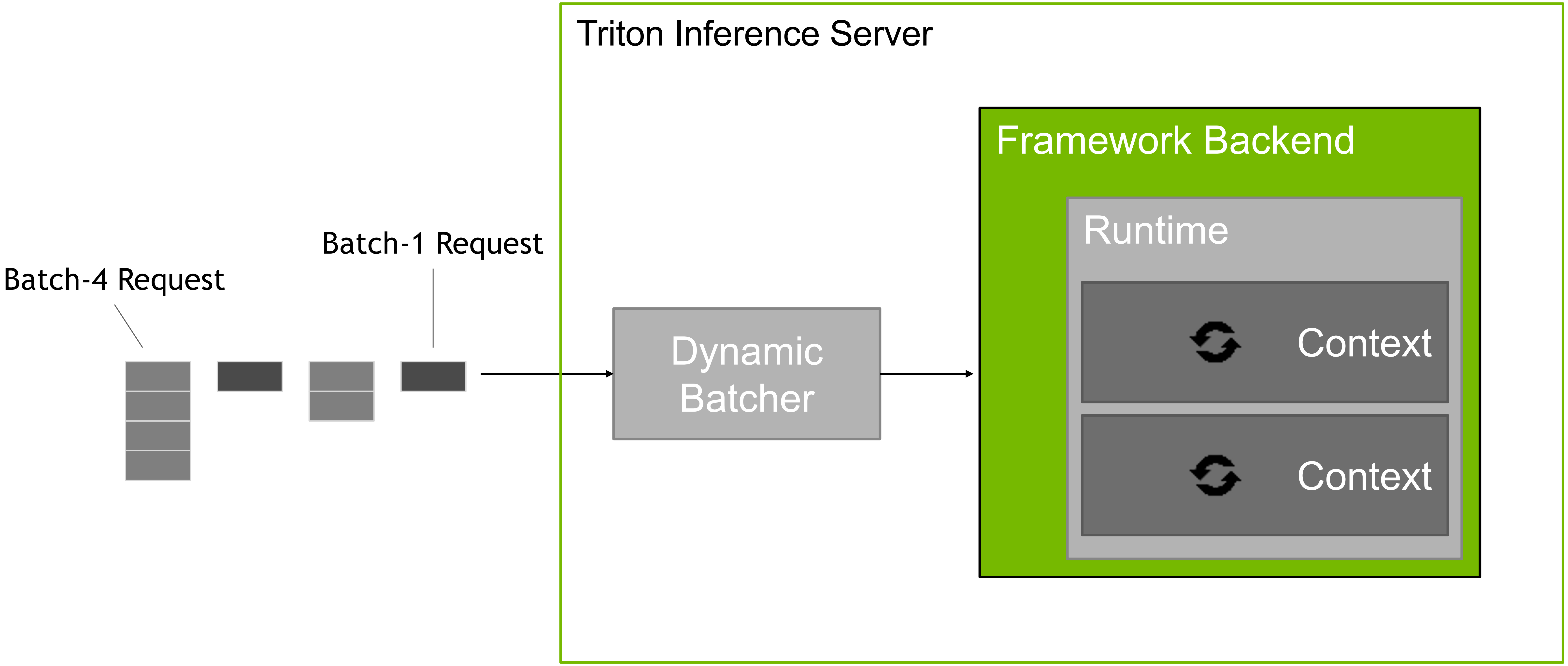
CMake build

Build the inference server from source making it more portable to multiple OSes and removing the build dependency on Docker

Streaming API

Built-in support for audio streaming input e.g. for speech recognition

DYNAMIC BATCHING SCHEDULER



Batch-4 Request

Batch-1 Request

Triton Inference Server

Dynamic Batcher

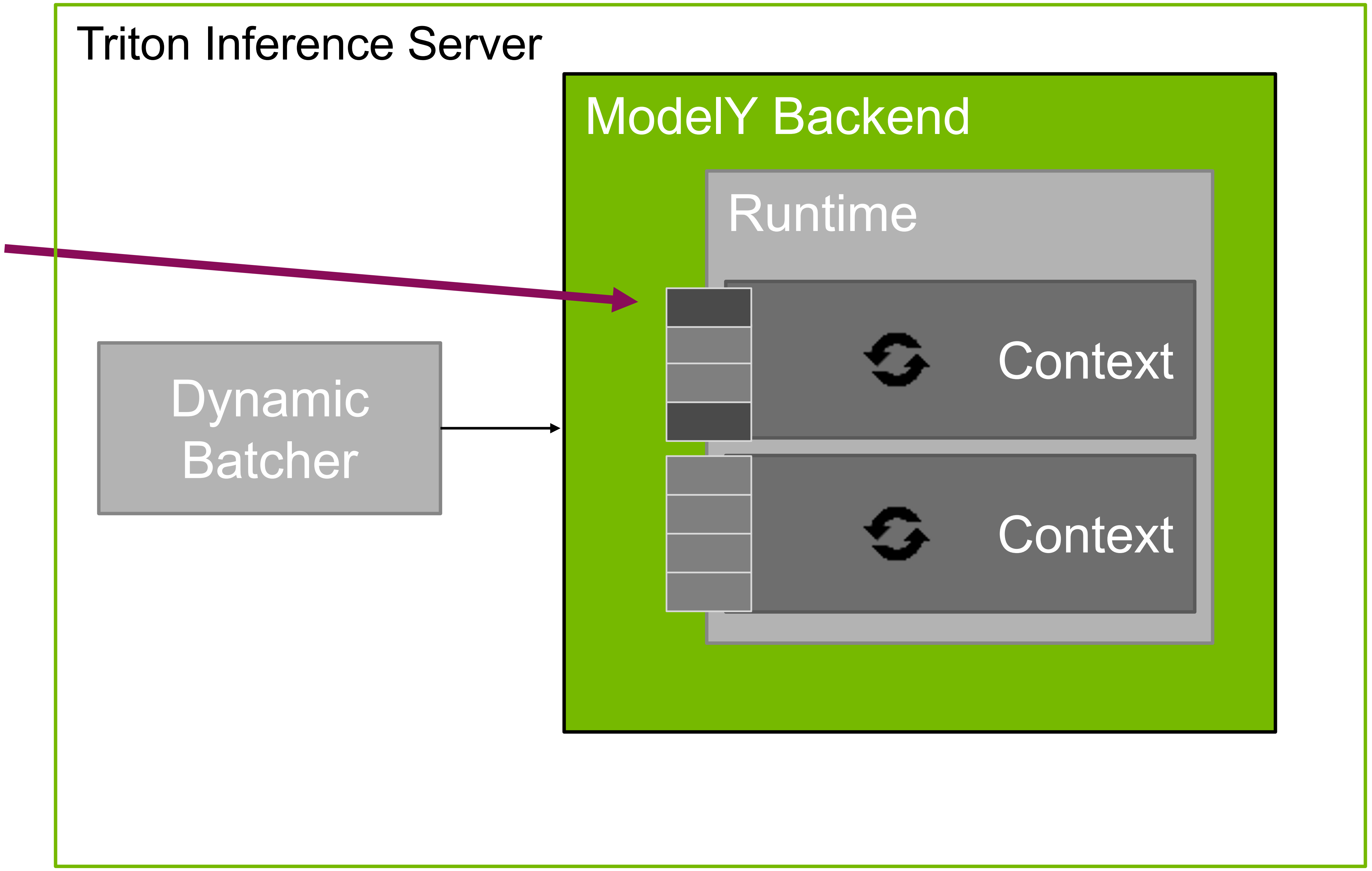
Framework Backend

Runtime

Context

Context

DYNAMIC BATCHING SCHEDULER



Grouping requests into a single “batch” increases overall GPU throughput

Preferred batch size and wait time are configuration options.

Assume 4 gives best utilization in this example.

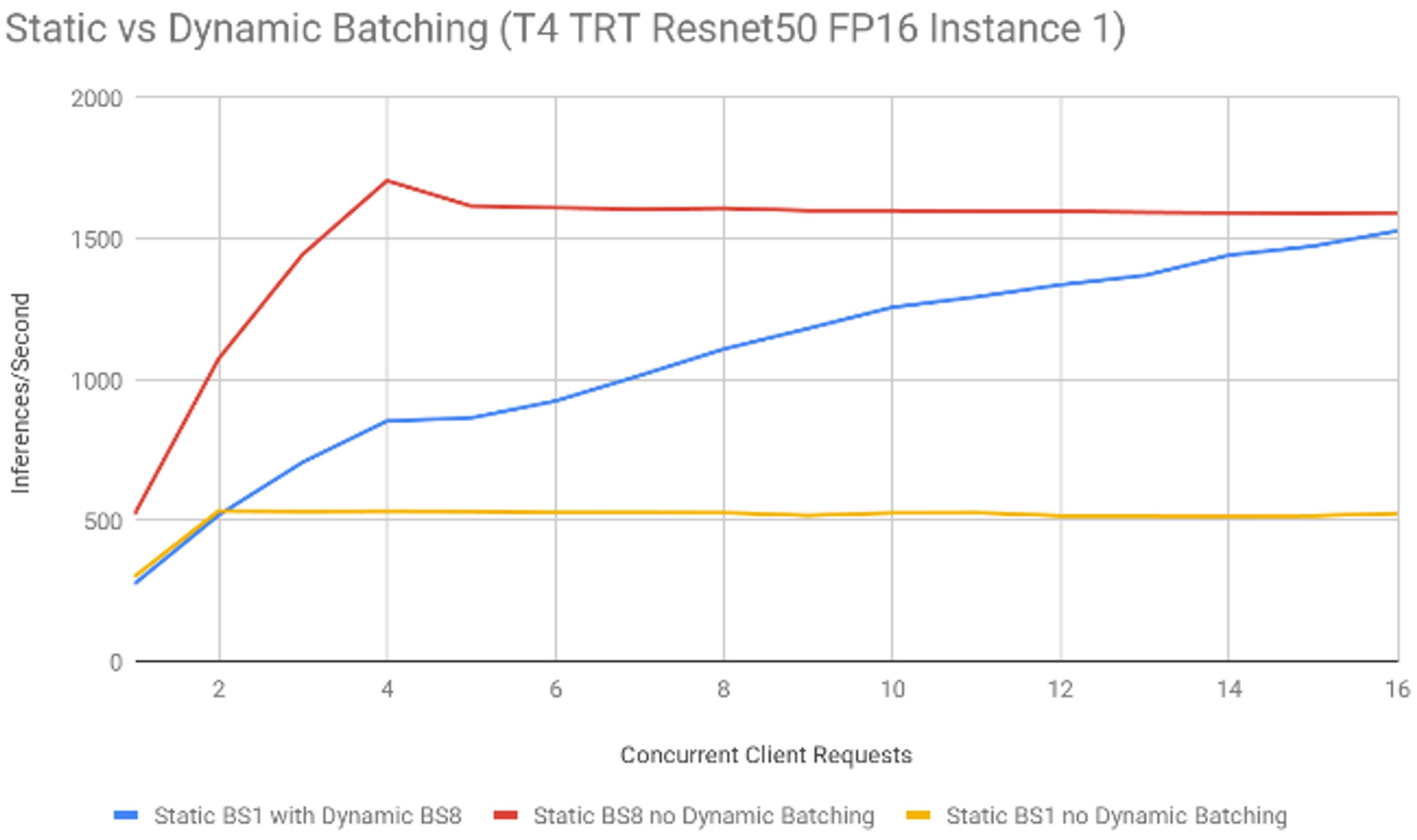
DYNAMIC BATCHING

2.5X Faster Inferences/Second at a 50ms End-to-End Server Latency Threshold

Triton Inference Server groups inference requests based on customer defined metrics for optimal performance

Customer defines 1) batch size (required) and 2) latency requirements (optional)

Example: No dynamic batching (batch size 1 & 8) vs dynamic batching



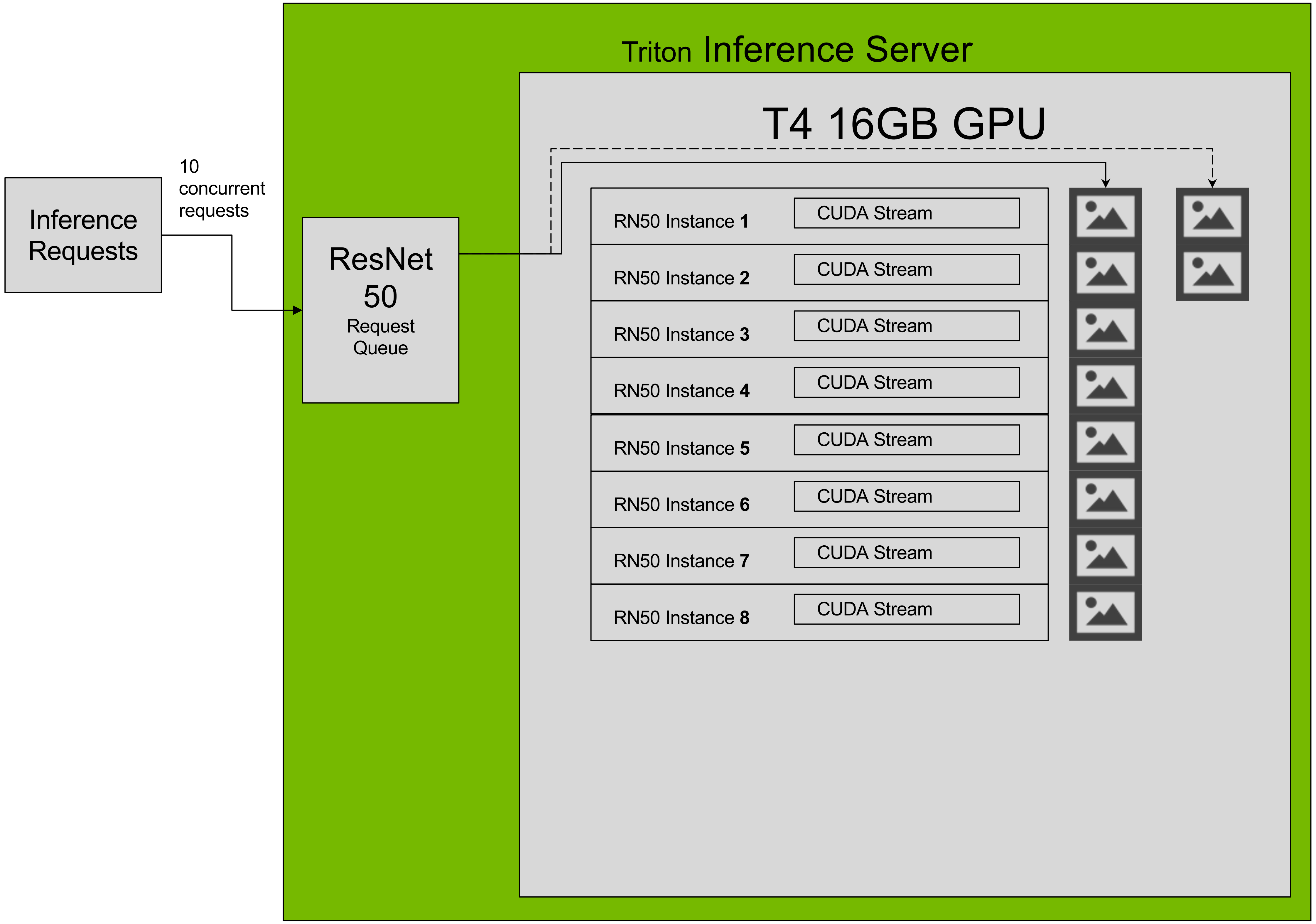
CONCURRENT MODEL EXECUTION - RESNET 50

6x Better Performance and Improved GPU Utilization Through Multiple Model Concurrency

Common Scenario 1

One API using multiple copies of the same model on a GPU

Example: 8 instances of TRT FP16 ResNet50 (each model takes 2 GB GPU memory) are loaded onto the GPU and can run concurrently on a 16GB T4 GPU.
10 concurrent inference requests happen: each model instance fulfills one request simultaneously and 2 are queued in the per-model scheduler queues in Triton Inference Server to execute after the 8 requests finish. With this configuration, 2680 inferences per second at 152 ms with batch size 8 on each inference server instance is achieved.



Inference Requests

10 concurrent requests

Triton Inference Server

T4 16GB GPU

ResNet 50 Request Queue

RN50 Instance **1** CUDA Stream

RN50 Instance **2** CUDA Stream

RN50 Instance **3** CUDA Stream

RN50 Instance **4** CUDA Stream

RN50 Instance **5** CUDA Stream

RN50 Instance **6** CUDA Stream

RN50 Instance **7** CUDA Stream

RN50 Instance **8** CUDA Stream

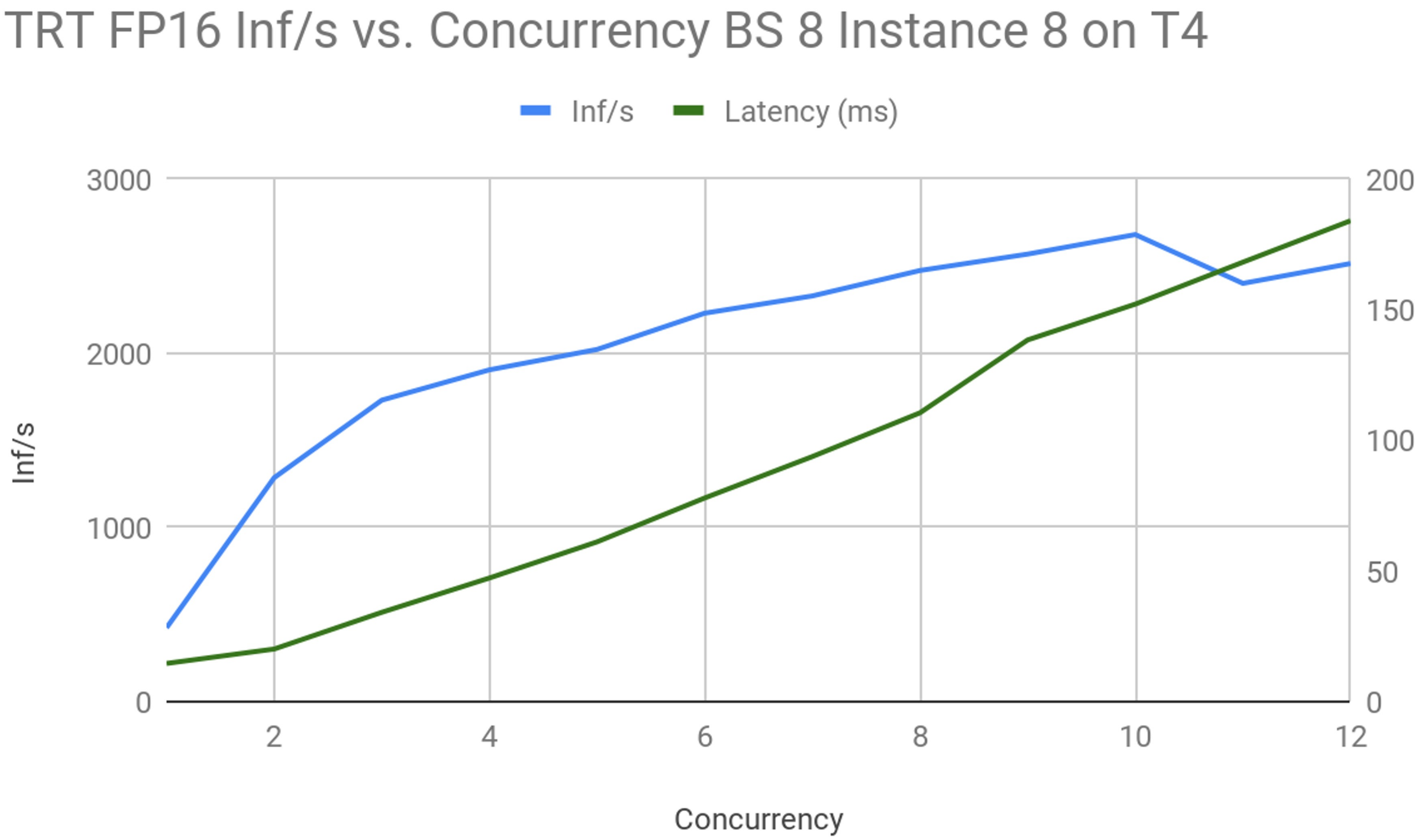
CONCURRENT MODEL EXECUTION - RESNET 50

6x Better Performance and Improved GPU Utilization Through Multiple Model Concurrency

Common Scenario 1

One API using multiple copies of the same model on a GPU

Example: 8 instances of TRT FP16 ResNet50 (each model takes 2 GB GPU memory) are loaded onto the GPU and can run concurrently on a 16GB T4 GPU.
10 concurrent inference requests happen: each model instance fulfills one request simultaneously and 2 are queued in the per-model scheduler queues in Triton Inference Server to execute after the 8 requests finish. With this configuration, 2680 inferences per second at 152 ms with batch size 8 on each inference server instance is achieved.

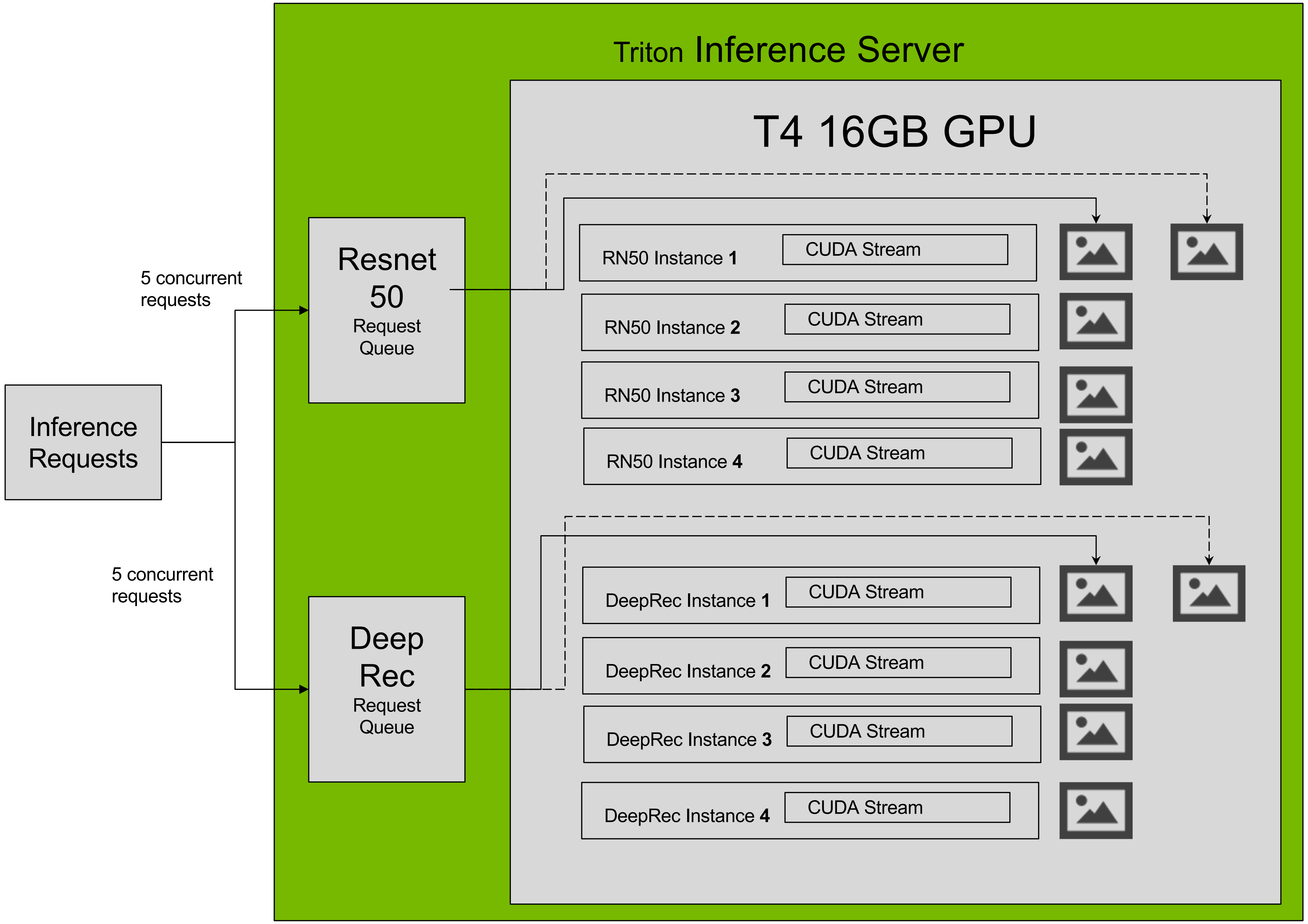


CONCURRENT MODEL EXECUTION
RESNET 50 & DEEP RECOMMENDER

Common Scenario 2

Many APIs using multiple different models on a GPU

Example: 4 instances of TRT FP16 ResNet50 and 4 instances of TRT FP16 Deep Recommender are running concurrently on one GPU. Ten requests come in for both models at the same time (5 for each model) and fed to the appropriate model for inference. The requests are fulfilled concurrently and sent back to the user. One request is queued for each model. With this configuration, 5778 inferences per second at 80 ms with batch size 8 on each inference server instance is achieved.

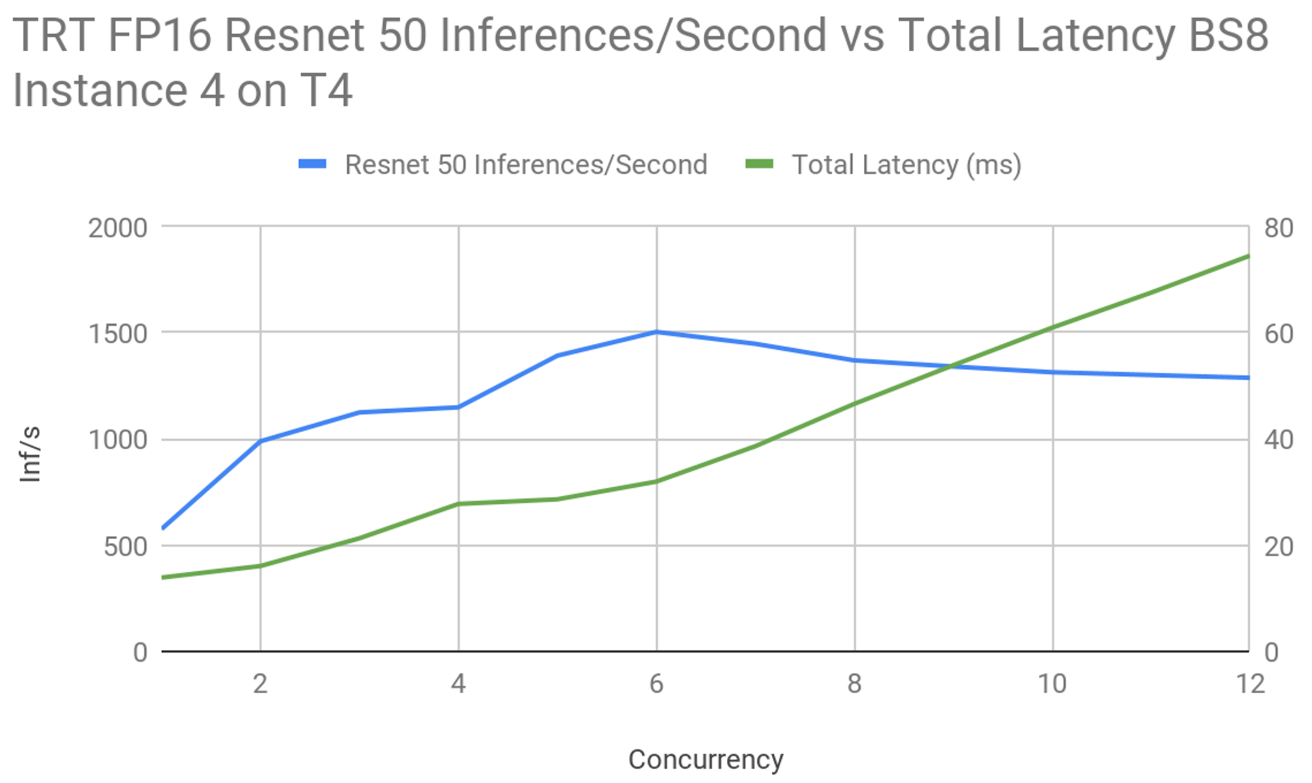


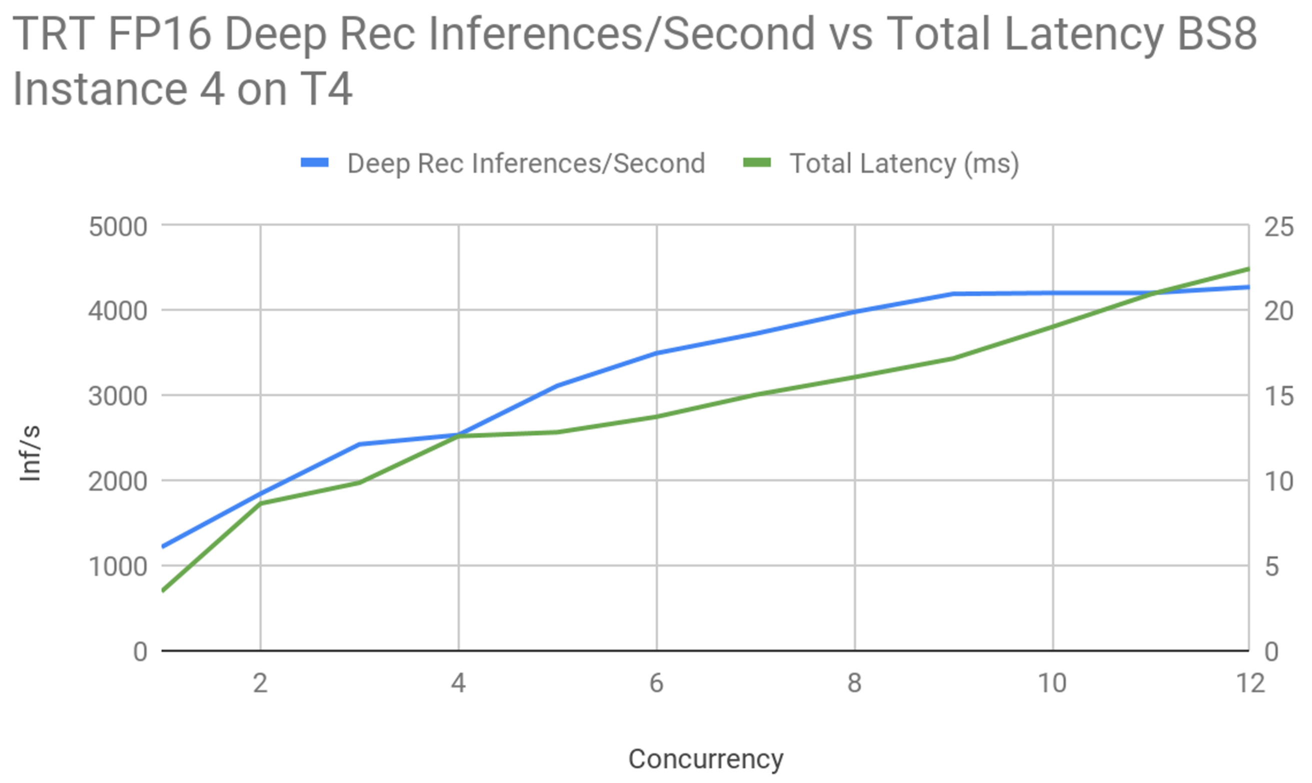
CONCURRENT MODEL EXECUTION
RESNET 50 & DEEP RECOMMENDER

Common Scenario 2

Many APIs using multiple different models on a GPU

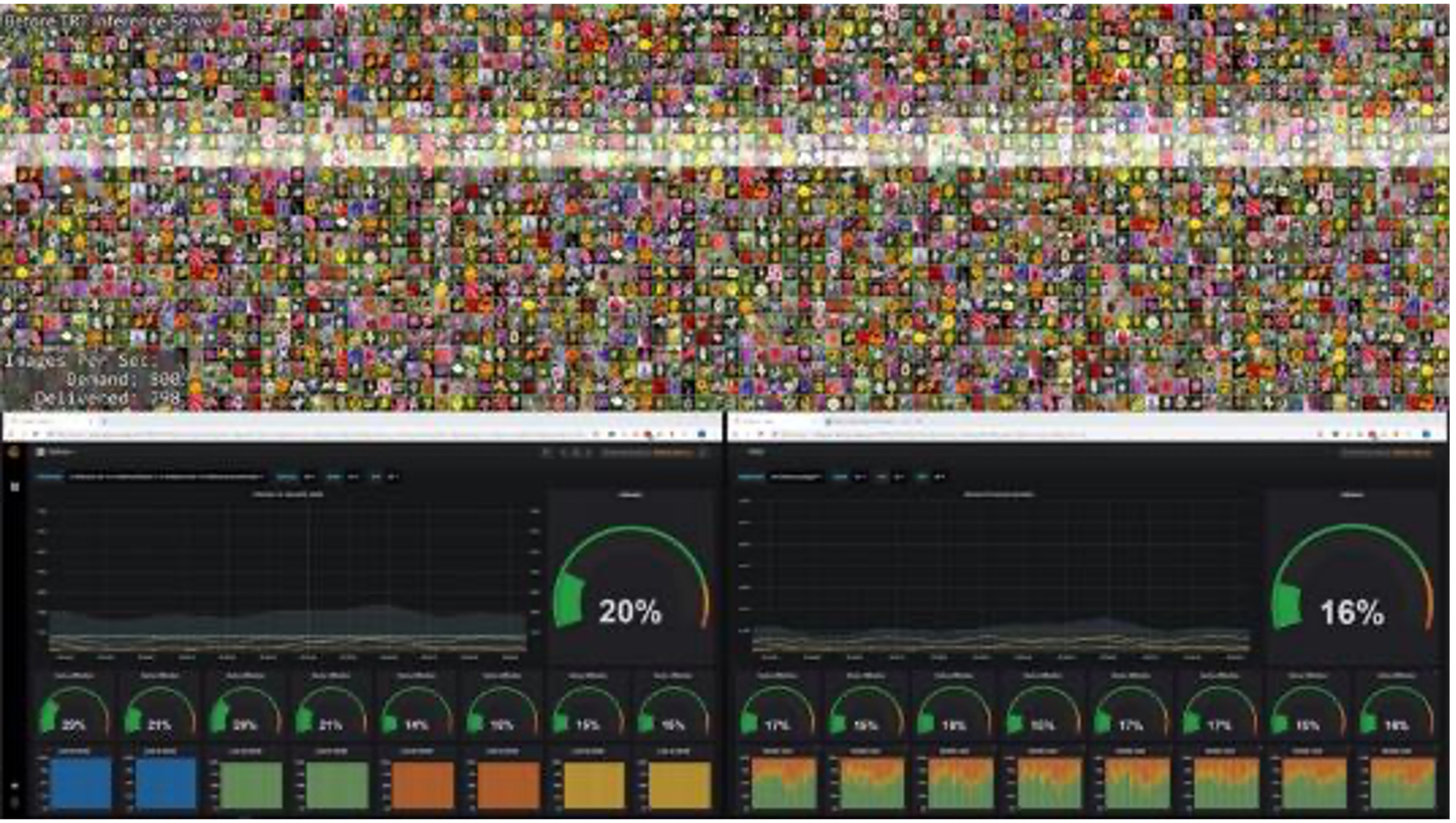
Example: 4 instances of TRT FP16 ResNet50 and 4 instances of TRT FP16 Deep Recommender are running concurrently on one GPU. Ten requests come in for both models at the same time (5 for each model) and fed to the appropriate model for inference. The requests are fulfilled concurrently and sent back to the user. One request is queued for each model. With this configuration, 5778 inferences per second at 80 ms with batch size 8 on each inference server instance is achieved.





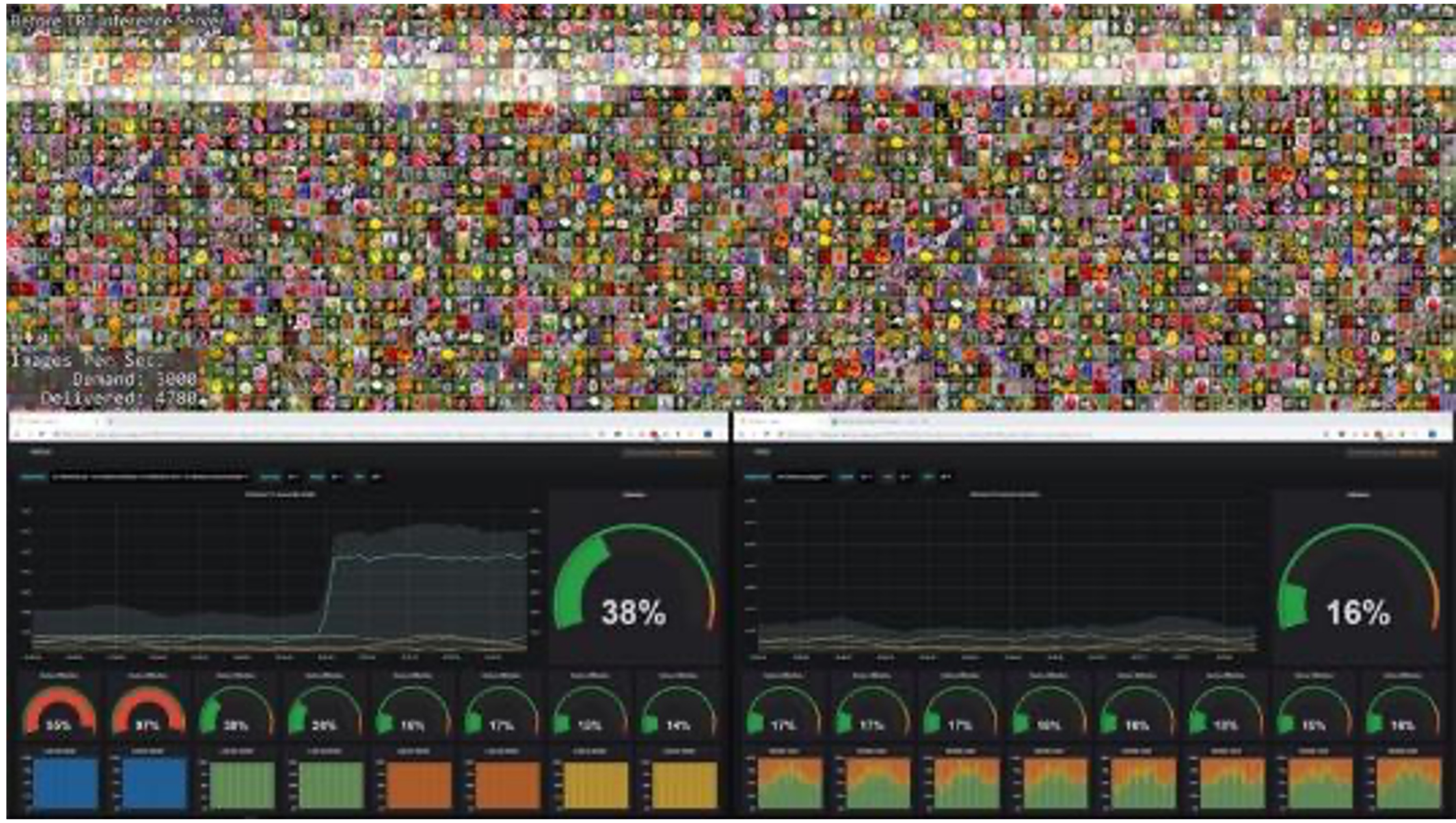
TRITON INFERENCE SERVER METRICS FOR AUTOSCALING

Before Triton Inference Server - 800 FPS



- One model per GPU
- Requests are steady across all models
- Utilization is low on all GPUs

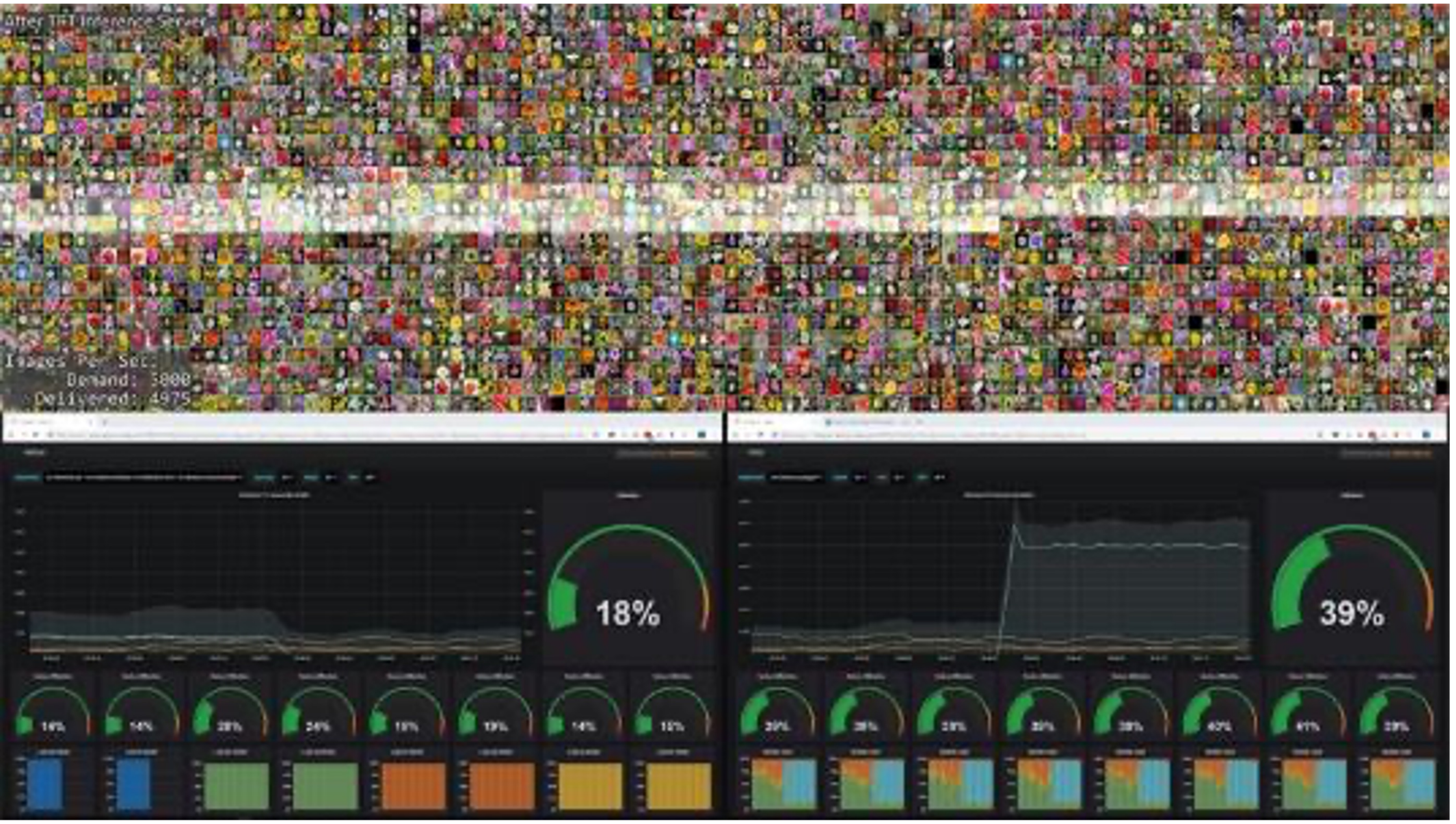
Before Triton Inference Server - 5,000 FPS



- Spike in requests for blue model
- GPUs running blue model are being fully utilized
- Other GPUs remain underutilized

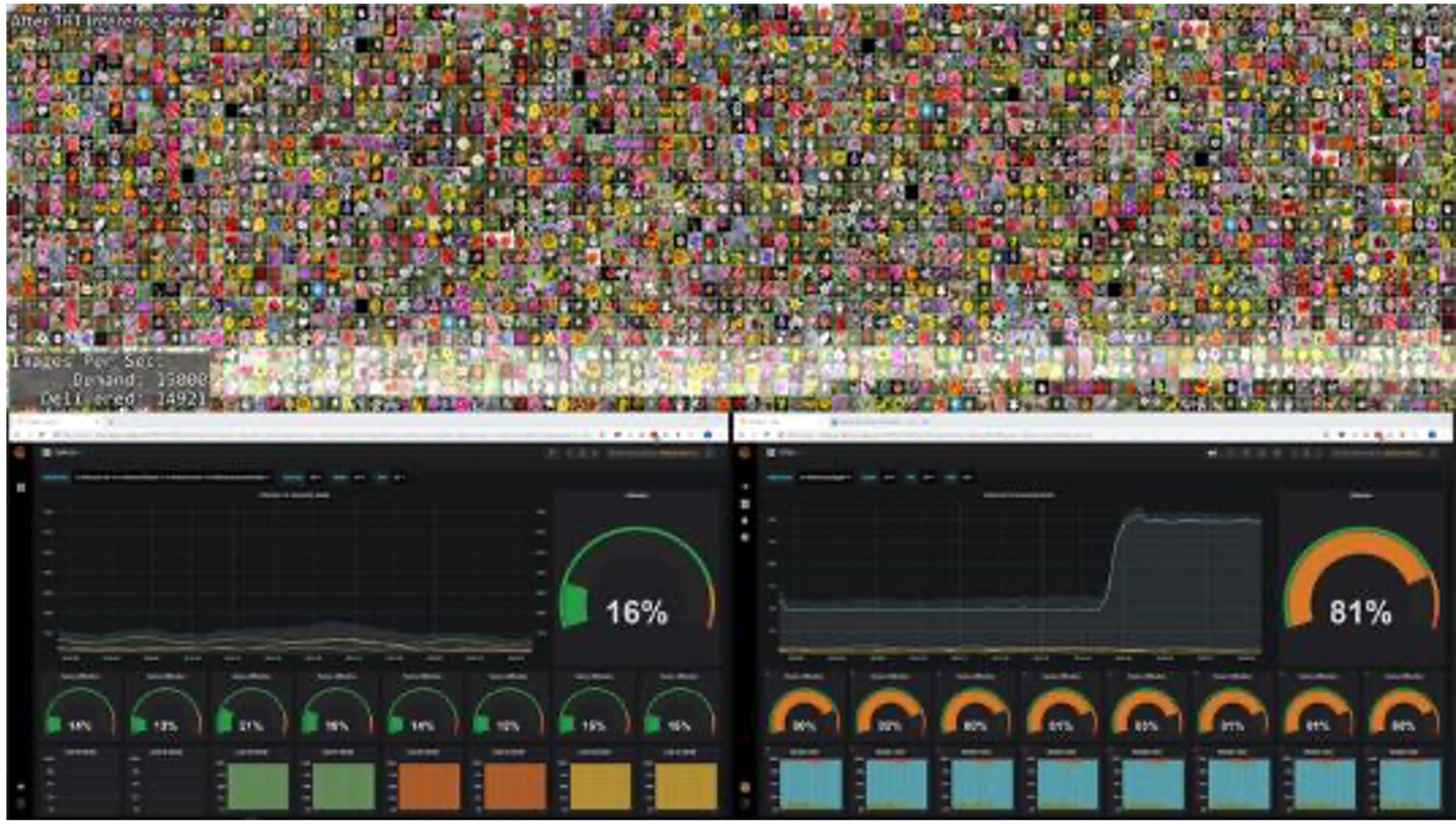
TRITON INFERENCE SERVER METRICS FOR AUTOSCALING

After Triton Inference Server - 5,000 FPS



- Load multiple models on every GPU
- Load is evenly distributed between all GPUs

After Triton Inference Server - 15,000 FPS

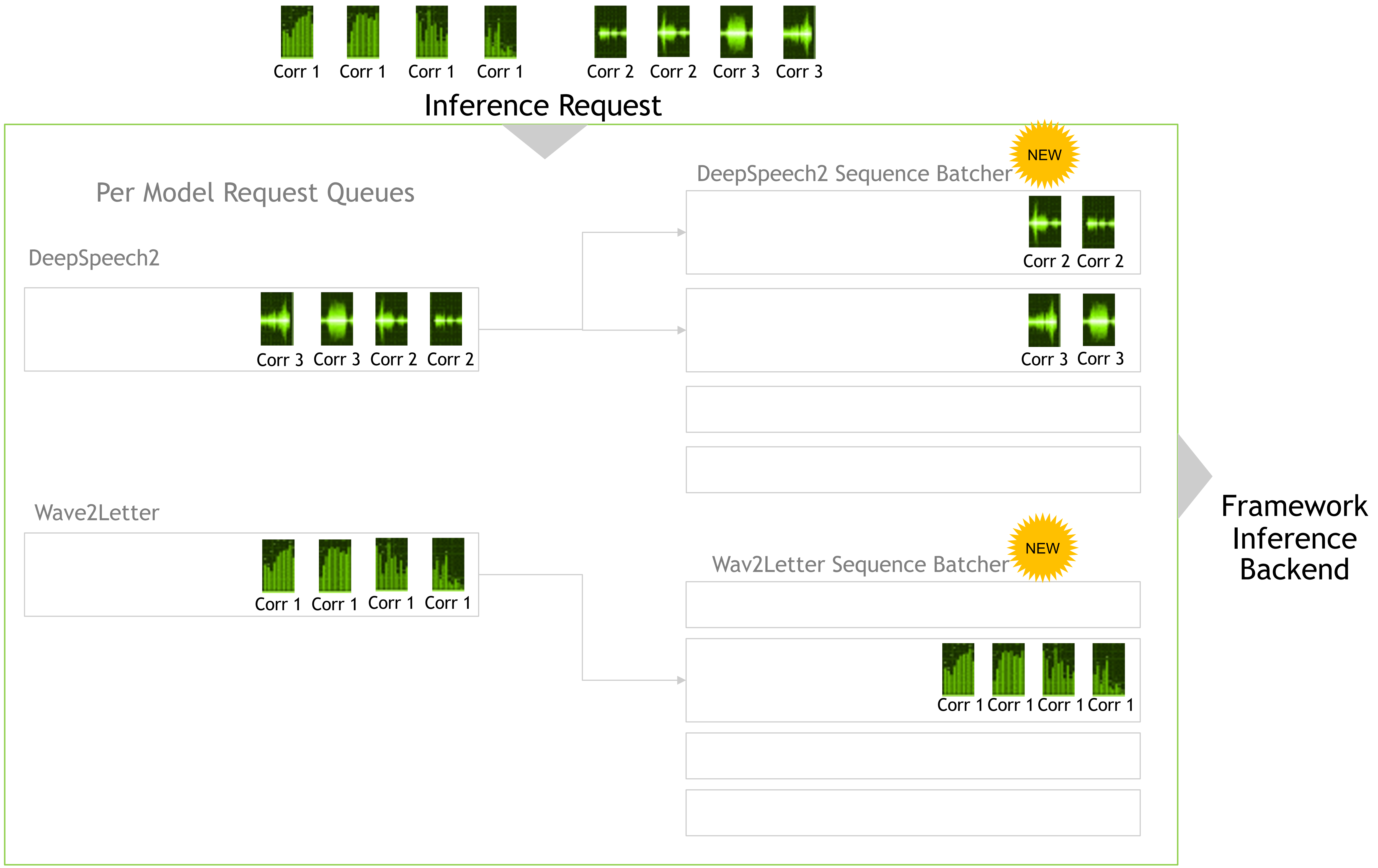


- Spike in requests for blue model
- Each GPU can run the blue model concurrently
- Metrics to indicate time to scale up
  - GPU utilization
  - Power usage
  - Inference count
  - Queue time
  - Number of requests/sec

STREAMING INFERENCE REQUESTS

New Streaming API

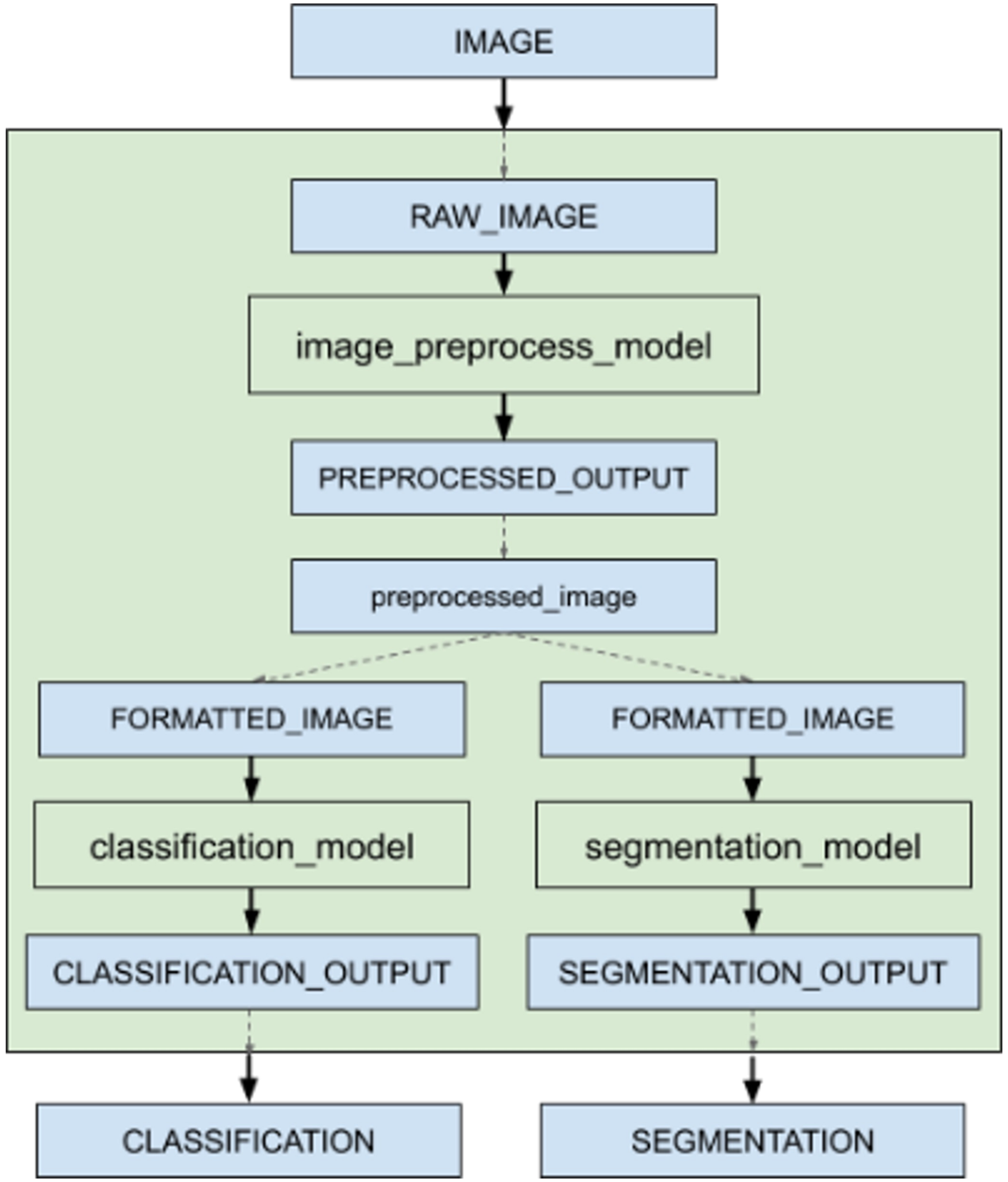
Based on the correlation ID, the audio requests are sent to the appropriate batch slot in the sequence batcher*



*Correct order of requests is assumed at entry into the endpoint
Note: Corr = Correlation ID

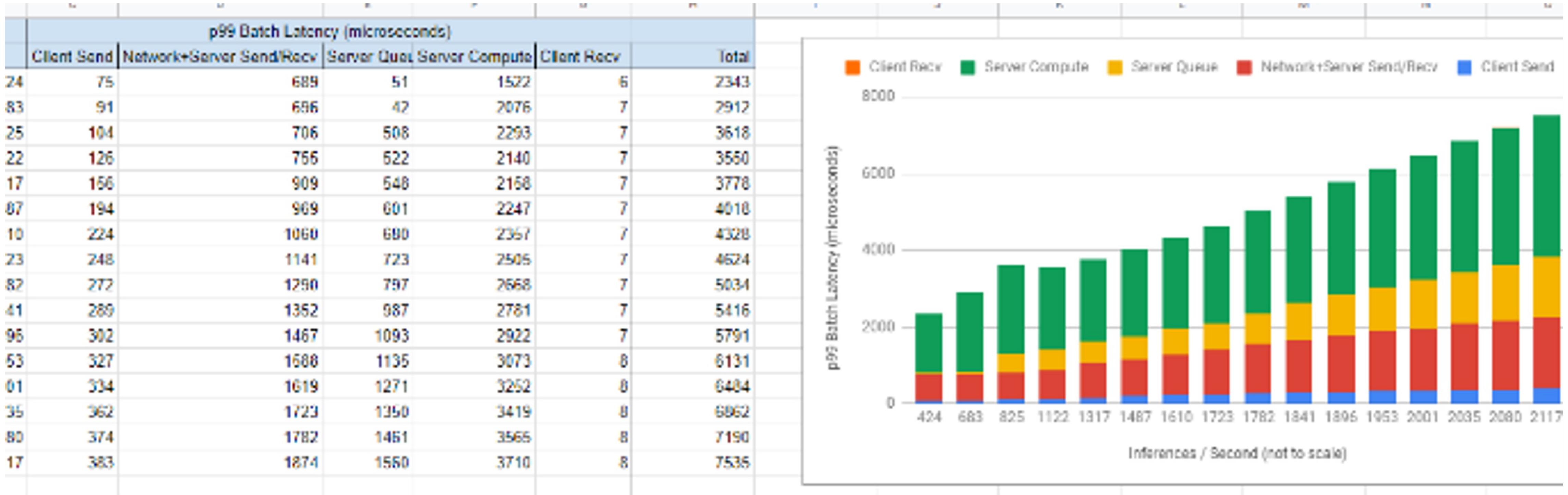
MODEL ENSEMBLING

- Pipeline of one or more models and the connection of input and output tensors between those models
- Use for model stitching or data flow of multiple models such as data preprocessing → inference → data post-processing
- Collects the output tensors in each step, provides them as input tensors for other steps according to the specification
- Ensemble models will inherit the characteristics of the models involved, so the meta-data in the request header must comply with the models within the ensemble

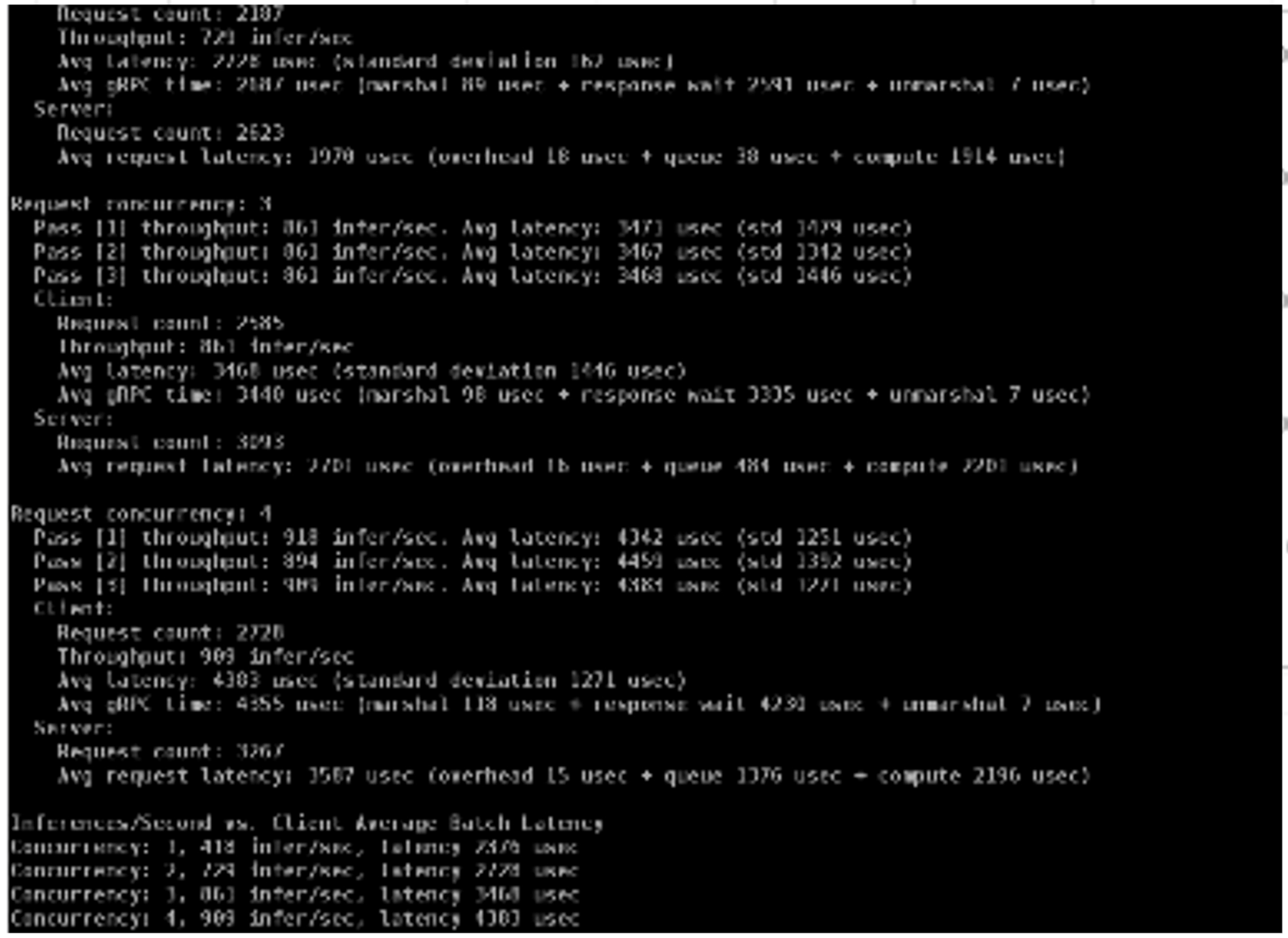


`perf_client` TOOL

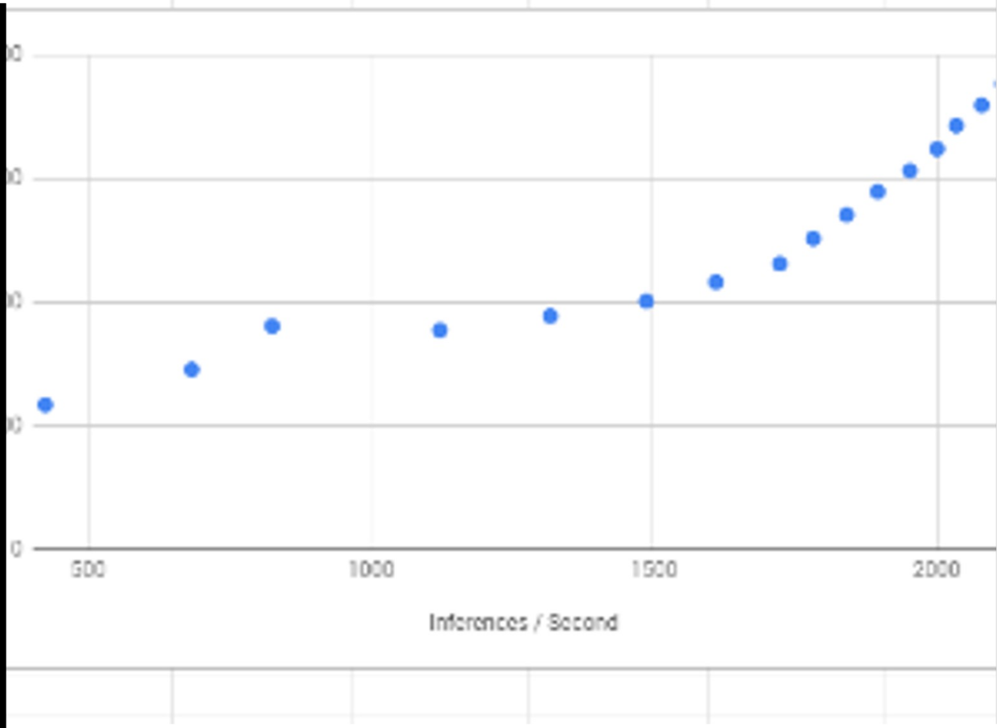
- Measures throughput (inf/s) and latency under varying client loads
- **`perf_client`** Modes
    1. Specify how many concurrent outstanding requests and it will find a stable latency and throughput for that level
    2. Generate throughput vs latency curve by increasing the request concurrency until a specific latency or concurrency limit is reached
- Generates a file containing CSV output of the results
- Easy steps to help visualize the throughput vs latency tradeoffs



	p99 Batch Latency (microseconds)					
	Client Send	Network+Server Send/Recv	Server Queu	Server Compute	Client Recv	Total
24	75	689	51	1522	6	2343
83	91	696	42	2076	7	2912
25	104	706	508	2293	7	3618
22	126	756	522	2140	7	3550
17	156	909	548	2158	7	3778
87	194	969	601	2247	7	4018
10	224	1060	680	2357	7	4328
23	248	1141	723	2505	7	4624
82	272	1290	797	2668	7	5034
41	289	1352	987	2781	7	5416
96	302	1467	1093	2922	7	5791
53	327	1688	1135	3073	8	6131
01	334	1619	1271	3252	8	6484
35	362	1723	1350	3419	8	6862
80	374	1782	1461	3565	8	7190
17	383	1874	1560	3710	8	7535

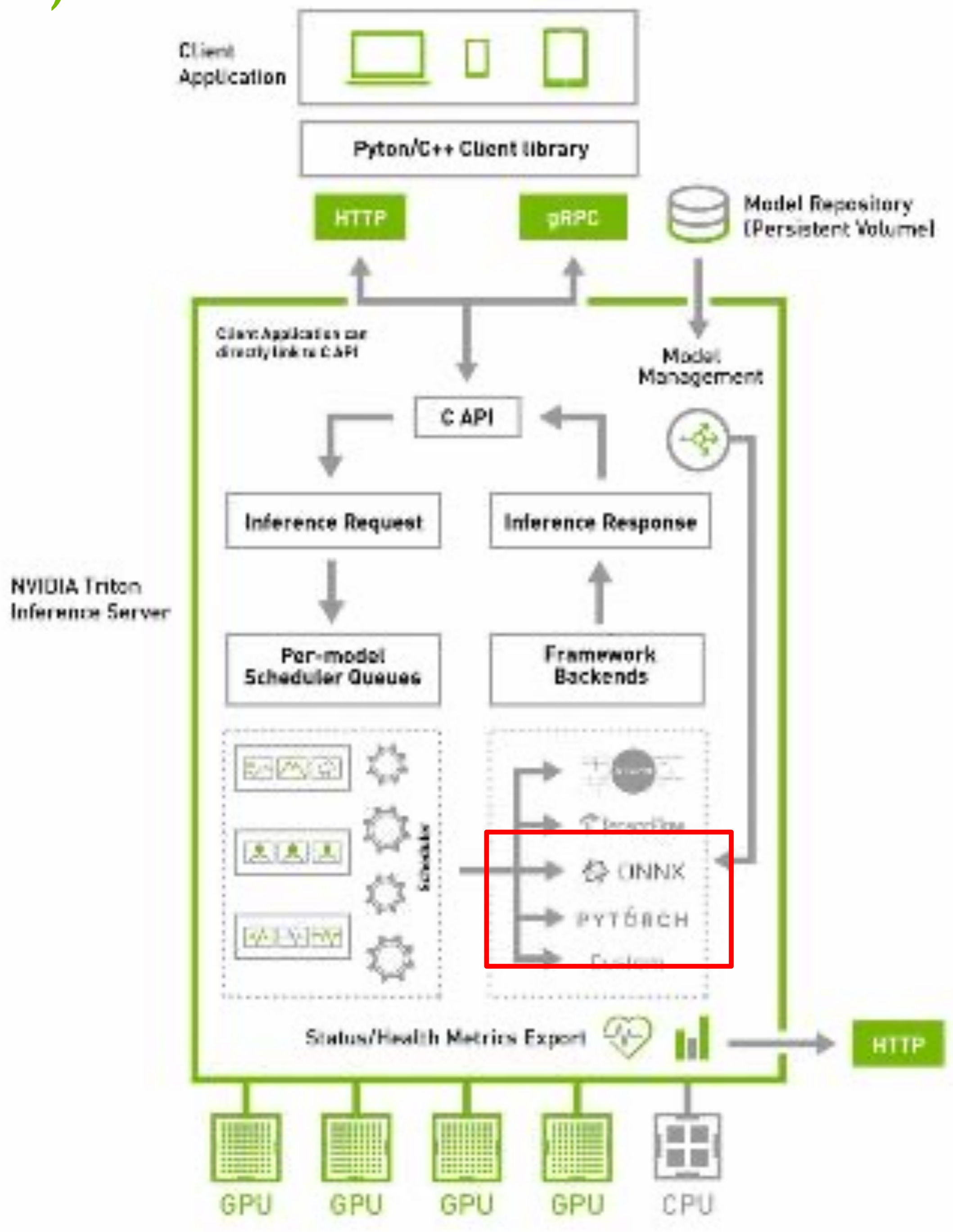


```
    Request count: 2187
    Throughput: 729 infer/sec
    Avg latency: 2228 usec (standard deviation 162 usec)
    Avg gRPC time: 2187 usec (marshal 89 usec + response wait 2591 usec + unmarshal 7 usec)
  Server:
    Request count: 2623
    Avg request latency: 1970 usec (overhead 18 usec + queue 38 usec + compute 1914 usec)

Request concurrency: 3
  Pass [1] throughput: 861 infer/sec. Avg latency: 3471 usec (std 1479 usec)
  Pass [2] throughput: 861 infer/sec. Avg latency: 3467 usec (std 1312 usec)
  Pass [3] throughput: 861 infer/sec. Avg latency: 3468 usec (std 1446 usec)
  Client:
    Request count: 2585
    Throughput: 861 infer/sec
    Avg latency: 3468 usec (standard deviation 1446 usec)
    Avg gRPC time: 3440 usec (marshal 98 usec + response wait 3335 usec + unmarshal 7 usec)
  Server:
    Request count: 3093
    Avg request latency: 2701 usec (overhead 16 usec + queue 484 usec + compute 2201 usec)

Request concurrency: 4
  Pass [1] throughput: 918 infer/sec. Avg latency: 4342 usec (std 1251 usec)
  Pass [2] throughput: 894 infer/sec. Avg latency: 4459 usec (std 1392 usec)
  Pass [3] throughput: 909 infer/sec. Avg latency: 4384 usec (std 1271 usec)
  Client:
    Request count: 2728
    Throughput: 909 infer/sec
    Avg latency: 4383 usec (standard deviation 1271 usec)
    Avg gRPC time: 4355 usec (marshal 118 usec + response wait 4230 usec + unmarshal 7 usec)
  Server:
    Request count: 3267
    Avg request latency: 3587 usec (overhead 15 usec + queue 1376 usec + compute 2196 usec)

Inferences/Second vs. Client Average Batch Latency
Concurrency: 1, 418 infer/sec, latency 2376 usec
Concurrency: 2, 729 infer/sec, latency 2228 usec
Concurrency: 3, 861 infer/sec, latency 3468 usec
Concurrency: 4, 909 infer/sec, latency 4383 usec
```



ALL CPU WORKLOADS SUPPORTED

Deploy the CPU workloads used today and benefit from Triton Inference Server features (TRT not required)



Triton relies on framework backends (Tensorflow, Caffe2, PyTorch) to execute the inference request on CPU

Support for Tensorflow and Caffe2 CPU optimizations using Intel MKL-DNN library

Allows frameworks backends to make use of multiple CPUs and cores

Benefit from features:

- Multiple Model Framework Support
- Dynamic batching
- Custom backend
- Model Ensembling
- Audio Streaming API

TRITON INFERENCE SERVER COLLABORATION WITH KUBEFLOW

What is Kubeflow?

- Open-source project to make ML workflows on Kubernetes simple, portable, and scalable
- Customizable scripts and configuration files to deploy containers on their chosen environment

Problems it solves

- Easily set up an ML stack/pipeline that can fit into the majority of enterprise datacenter and multi-cloud environments

How it helps Triton Inference Server

- Triton Inference Server is deployed as a component inside of a production workflow to
    - Optimize GPU performance
    - Enable auto-scaling, traffic load balancing, and redundancy/failover via metrics

For a more detailed explanation and step-by-step guidance for this collaboration, refer to this GitHub repo.

TRITON INFERENCE SERVER HELM CHART

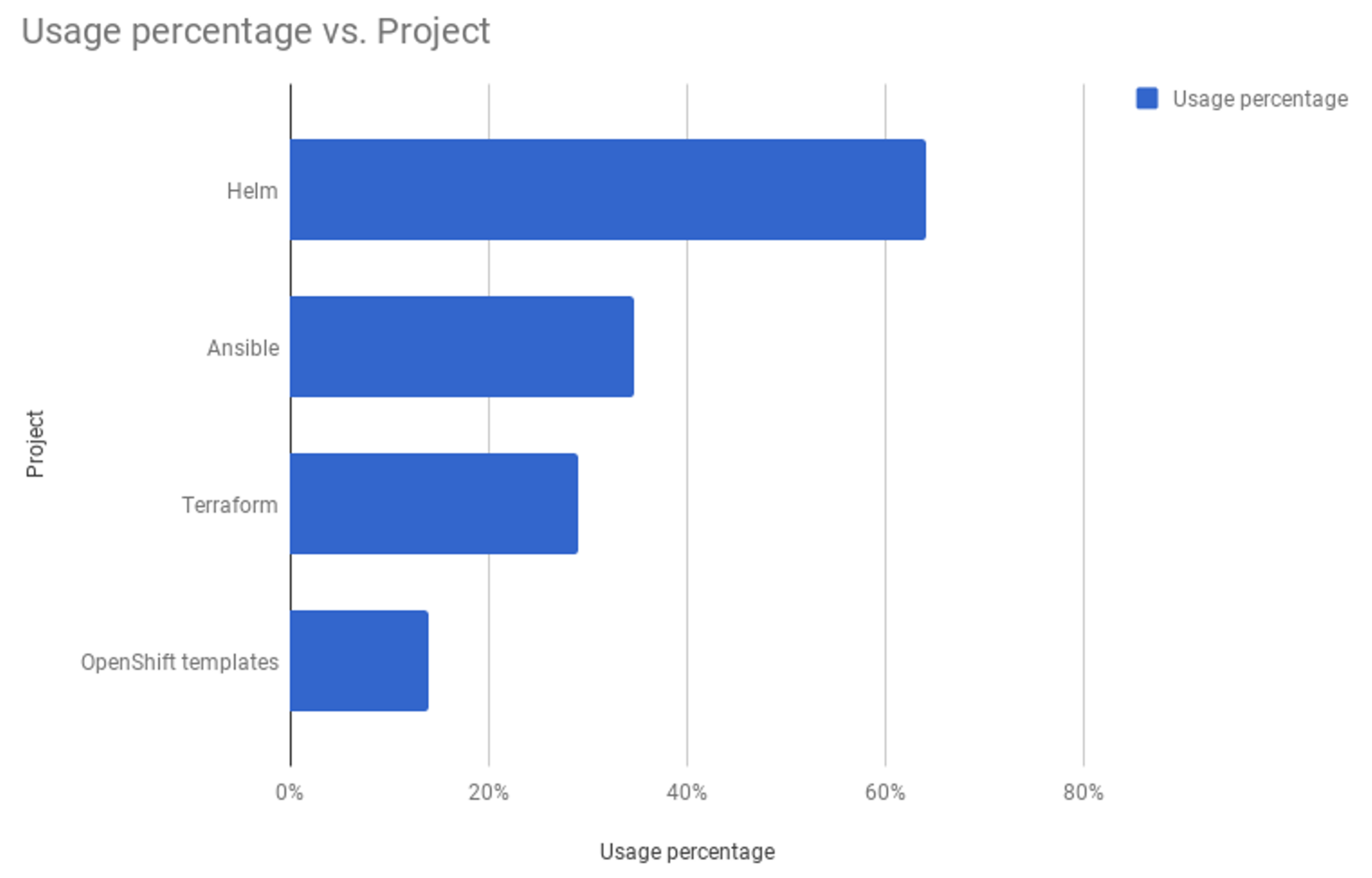
Simple helm chart for installing a single instance of the NVIDIA Triton Inference Server

Helm: Most used "package manager" for Kubernetes

We built a simple chart ("package") for the Triton Inference Server.

You can use it to easily deploy an instance of the server. It can also be easily configured to point to a different image, model store, …

https://github.com/NVIDIA/tensorrt-inference-server/tree/b6b45ead074d57e3d18703b7c0273672c5e92893/deploy/single_server



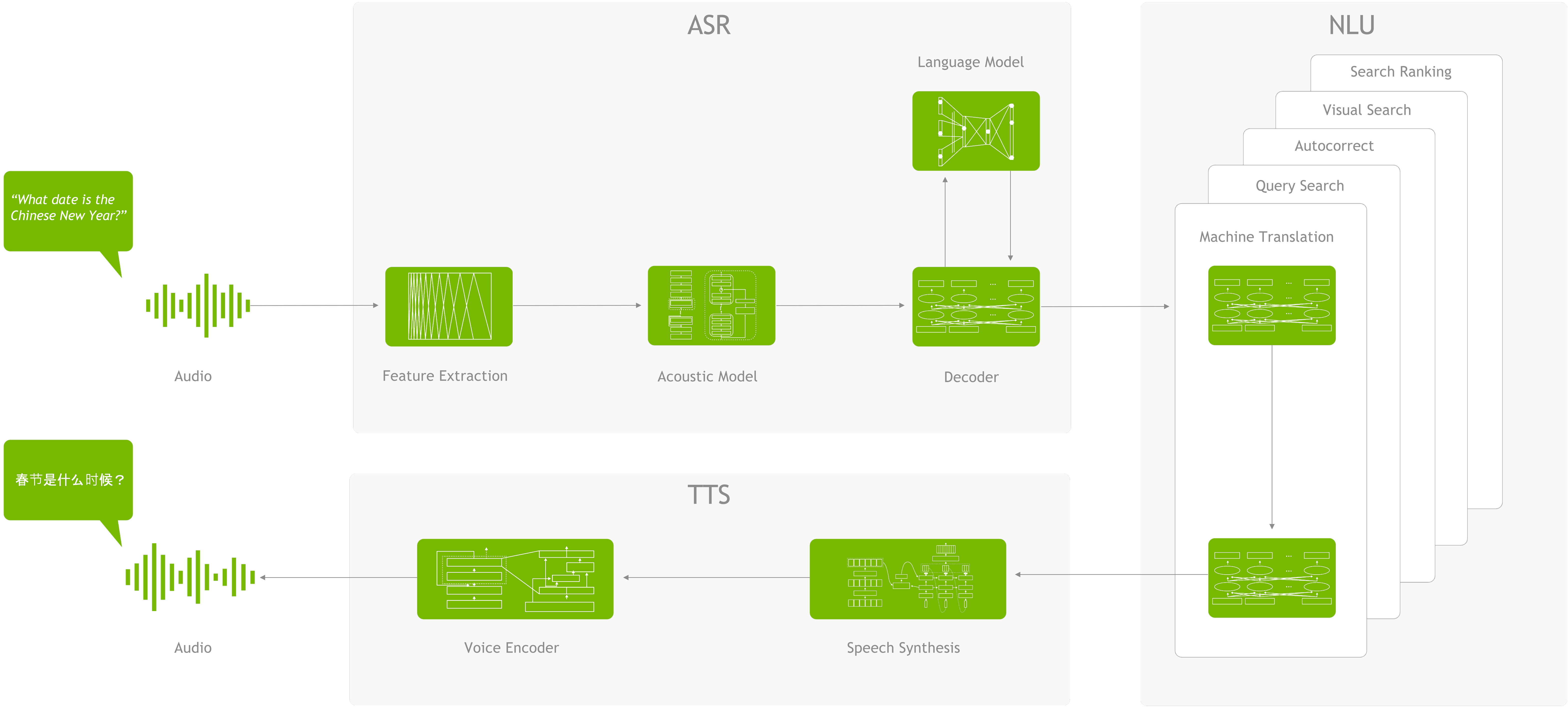
Part 3: Production Deployment

- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model

APPLICATION != SINGLE MODEL

THE APPLICATION

Typically composed of many components



"What date is the Chinese New Year?"

Audio

ASR

Feature Extraction

Acoustic Model

Language Model

Decoder

NLU

Search Ranking

Visual Search

Autocorrect

Query Search

Machine Translation

TTS

Speech Synthesis

Voice Encoder

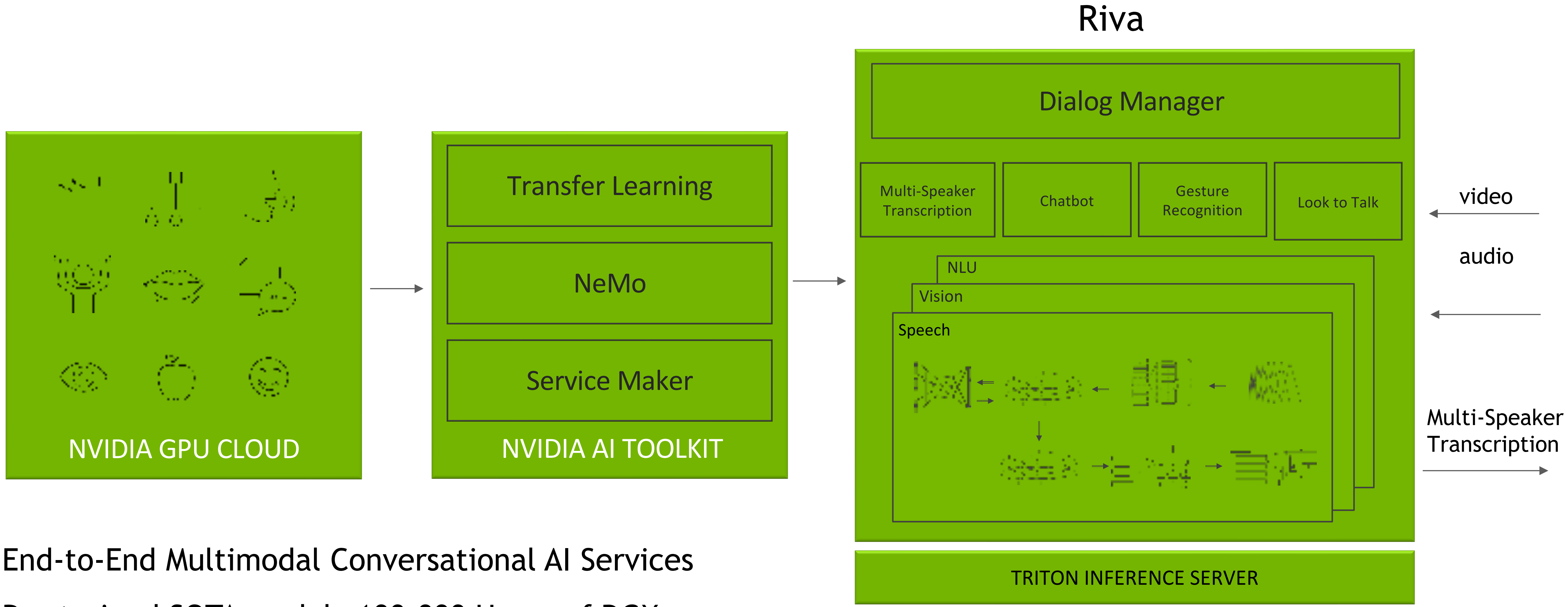
春节是什么时候？

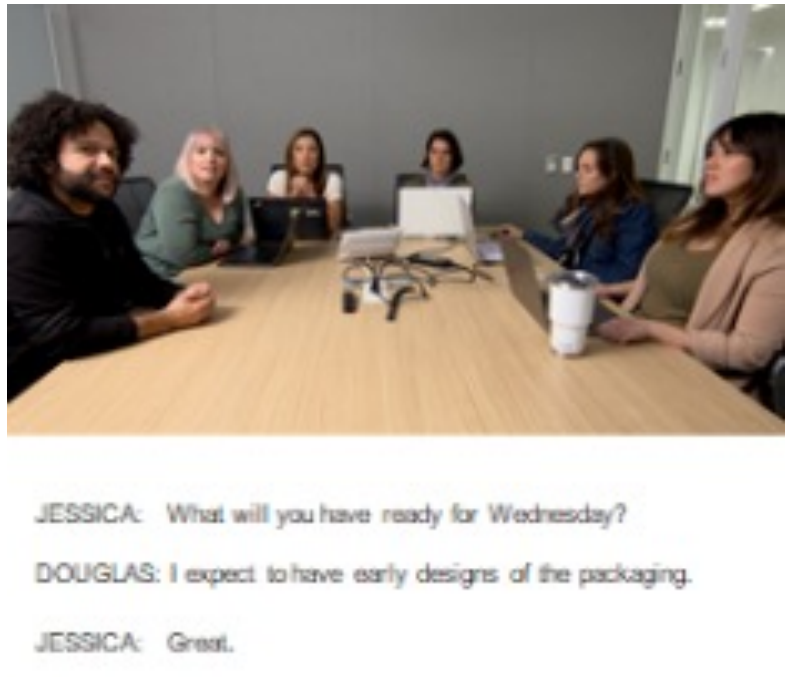
Audio

RIVA

NVIDIA RIVA

Fully Accelerated Framework for Multimodal Conversational AI Services





JESSICA: What will you have ready for Wednesday?

DOUGLAS: I expect to have early designs of the packaging.

JESSICA: Great.

End-to-End Multimodal Conversational AI Services

Pre-trained SOTA models-100,000 Hours of DGX

Retrain with NeMo

Interactive Response – 150ms on A100 versus 25sec on CPU

Deploy Services with One Line of Code

PRETRAINED MODELS AND AI TOOLKIT

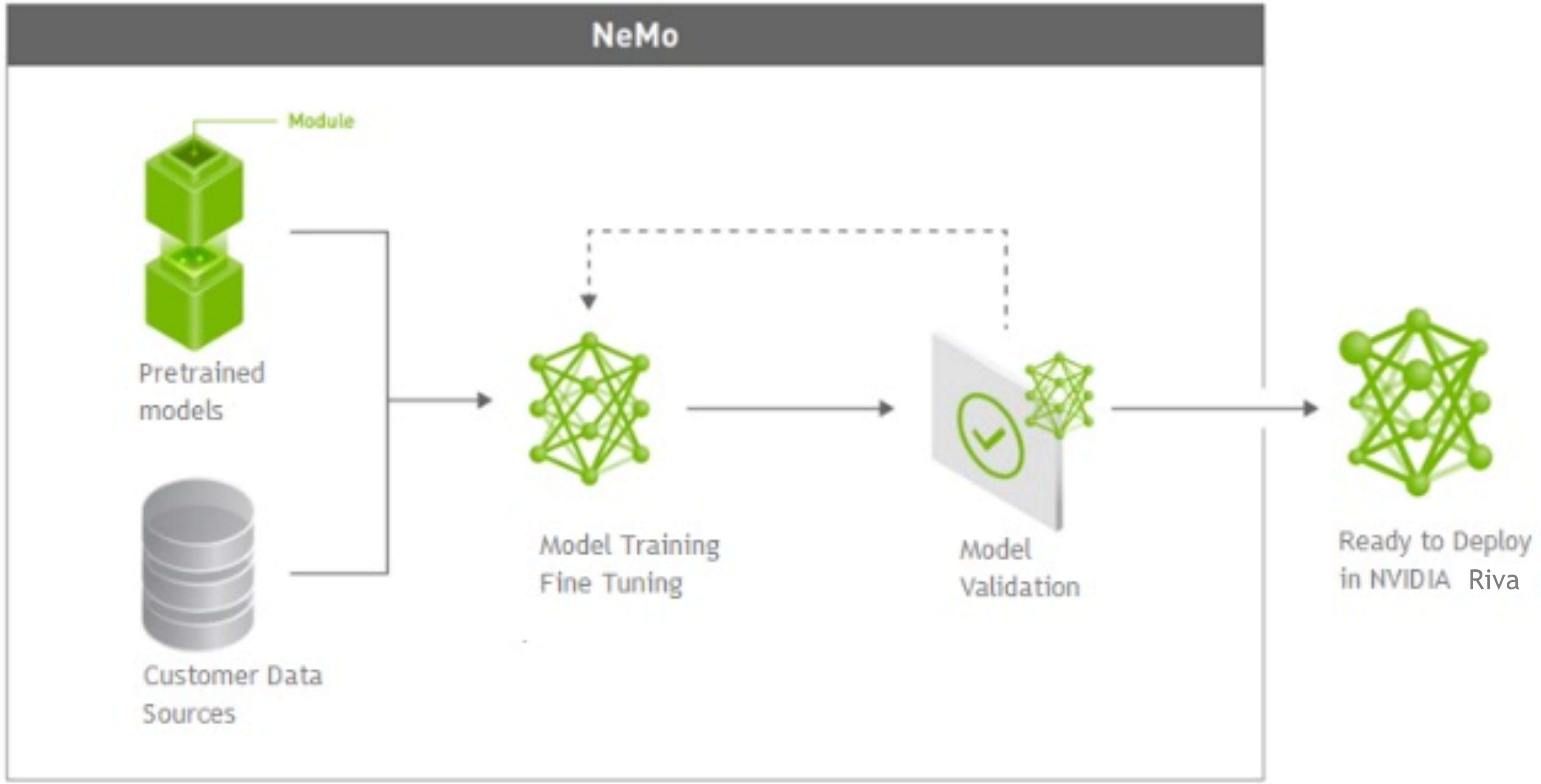
Train SOTA Models on Your Data to Understand your Domain and Jargon

100+ pretrained models in NGC

SOTA models trained over 100,000 hours on NVIDIA DGX™

Retrain for your domain using NeMo & TAO Toolkit

Deploy trained models to real-time services using Helm charts



MULTIMODAL SKILLS

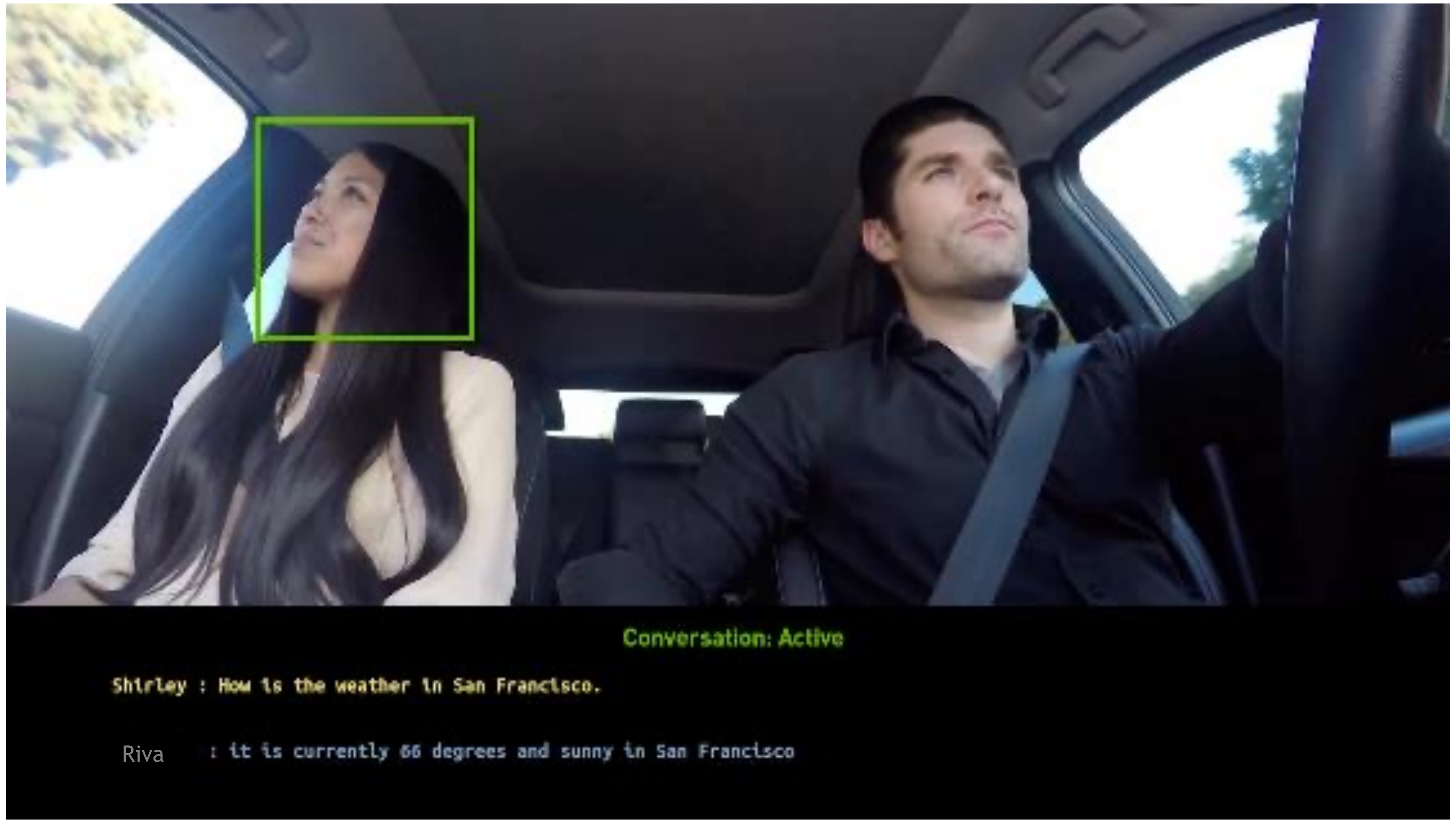
Use speech and vision for natural interaction

Build new skills by fusing services for ASR, NLU, TTS, and CV

Reference skills include:

- Multi-speaker transcription
- Chatbot
- Look-to-talk

Dialog manager manages multi-user and multi-context scenarios



Conversation: Active

Shirley : How is the weather in San Francisco.

Riva : it is currently 66 degrees and sunny in San Francisco

Multimodal application with multiple users and contexts

BUILD CONVERSATIONAL AI SERVICES

Optimized Services for Real Time Applications

Build applications easily by connecting performance tuned services

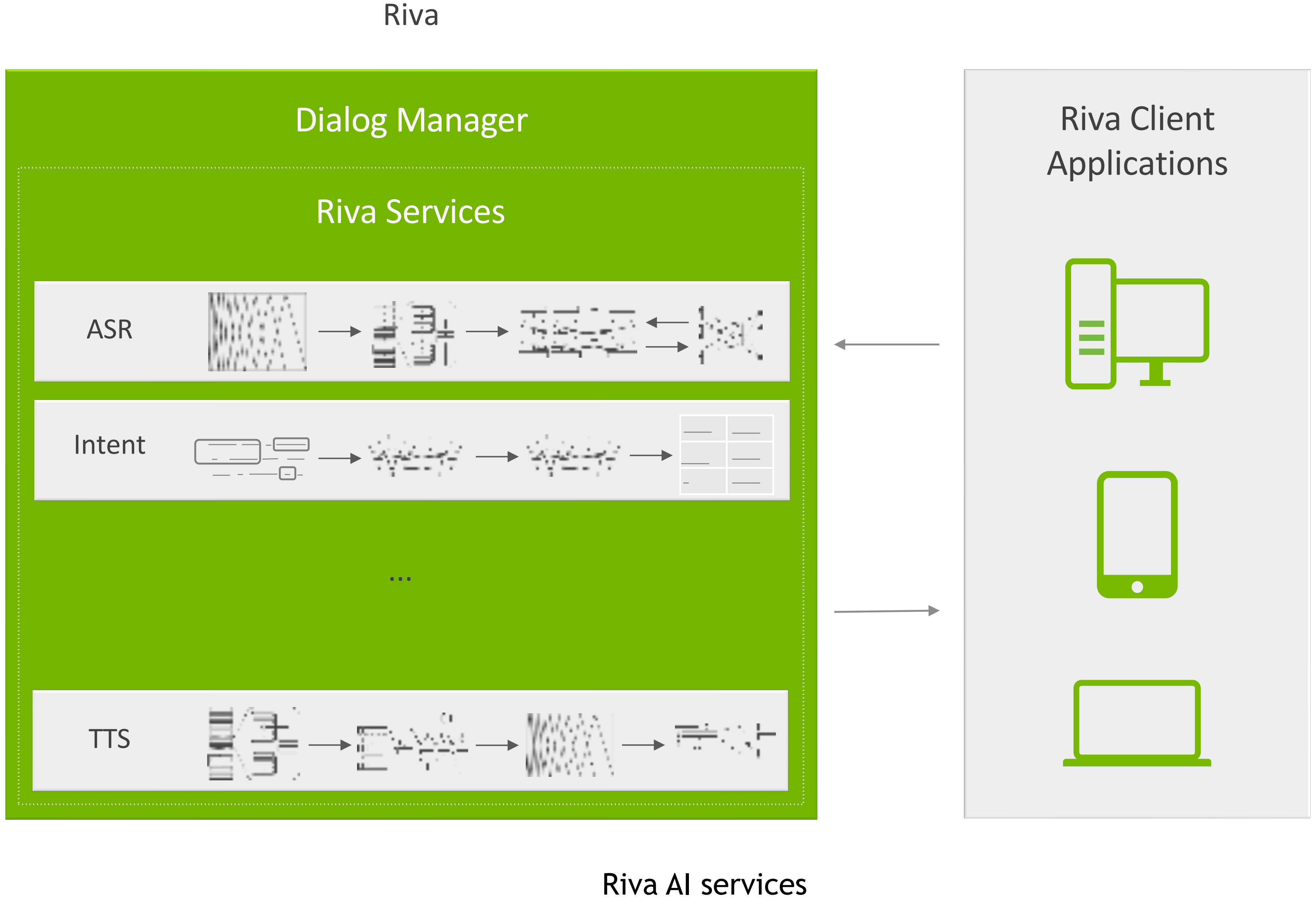
Task specific services include:

- ASR
- Intent Classification
- Slot Filling
- Pose Estimation
- Facial Landmark Detection

Services for streaming & batch usage

Build new services from any model in ONNX format

Access services for gRPC and HTTP endpoints



https://ngc.nvidia.com/catalog/model-scripts/nvidia:jasper_for_trtis

DEPLOY MODELS AS REAL-TIME SERVICES

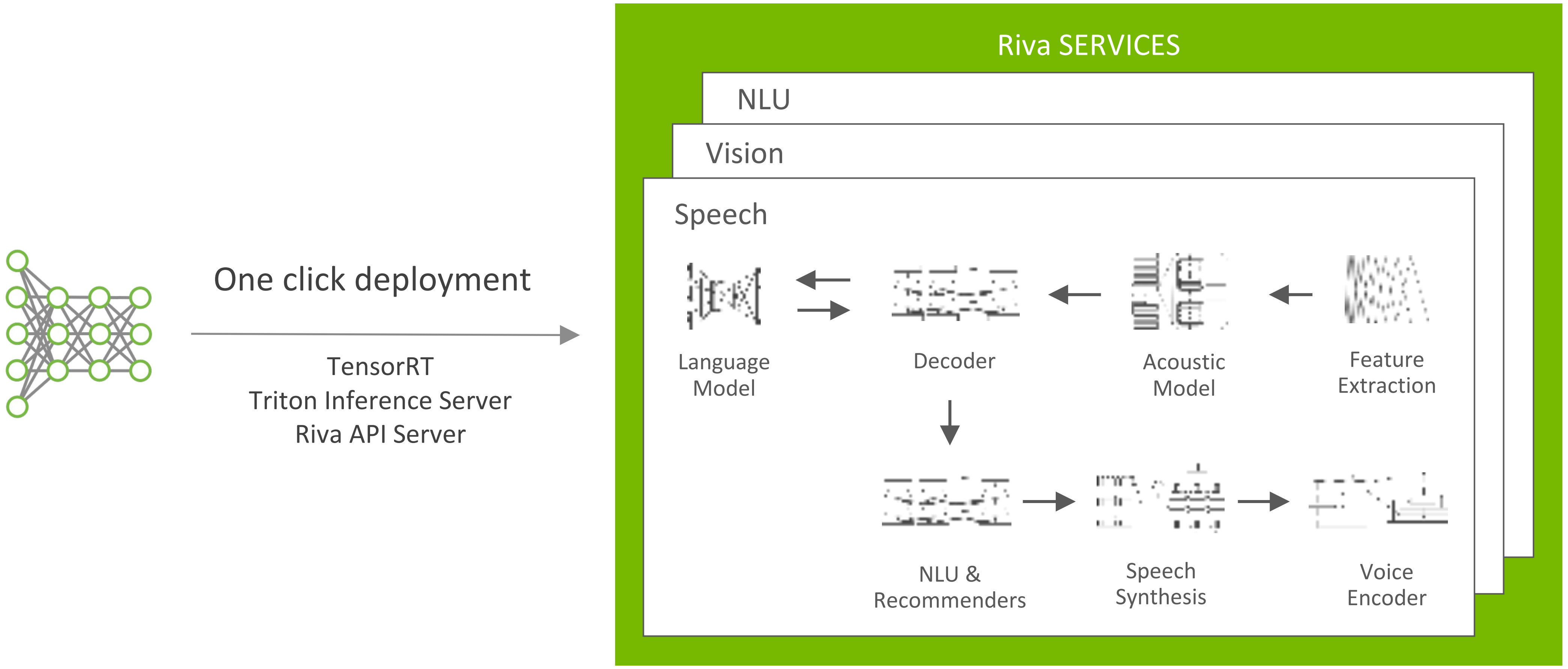
One Click to Create High-Performance Services from SOTA Models

Deploy models to services in the cloud, data center, and at the edge

Single command to set up and run the entire Riva application

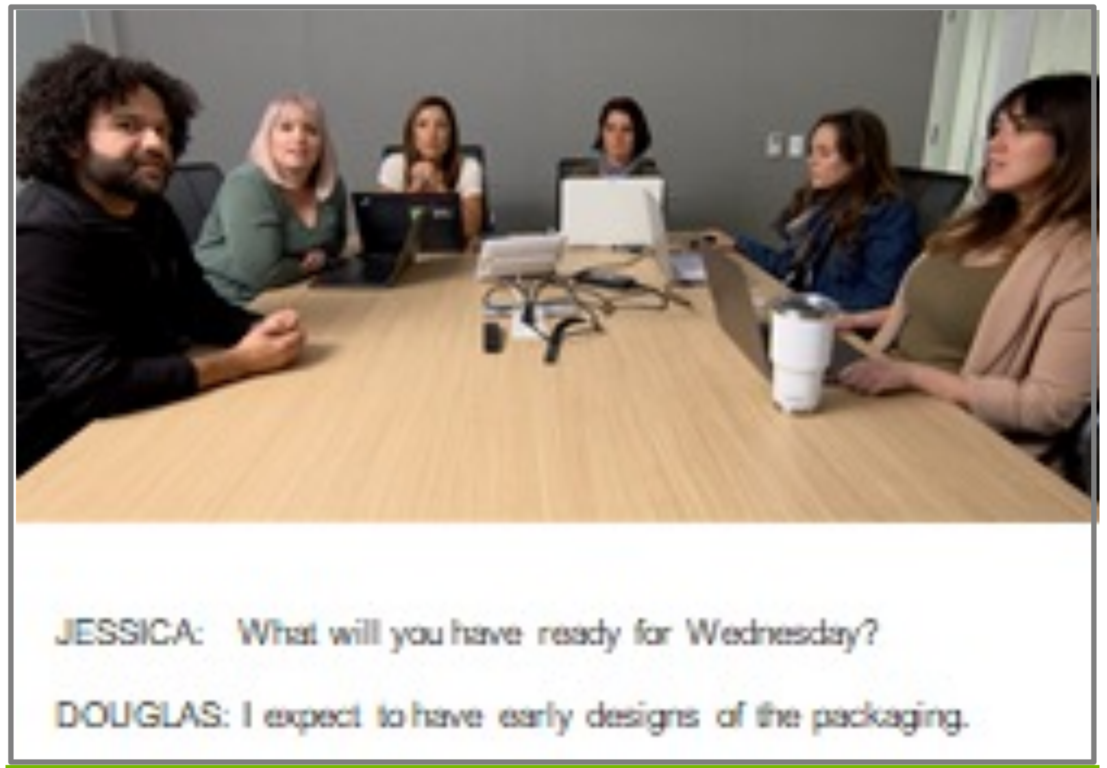
through Helm charts on Kubernetes cluster

Customization of Helm charts for your setup and use case.



Helm command to deploy models to production

RIVA SAMPLES

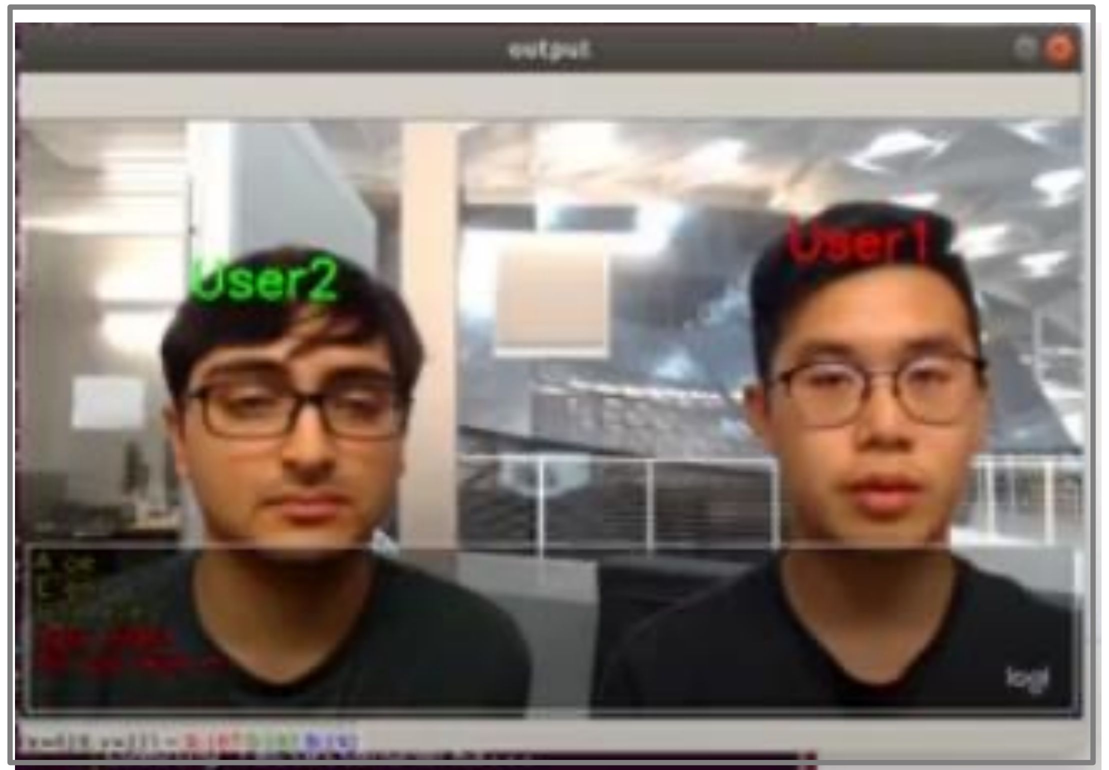


JESSICA: What will you have ready for Wednesday?

DOUGLAS: I expect to have early designs of the packaging.

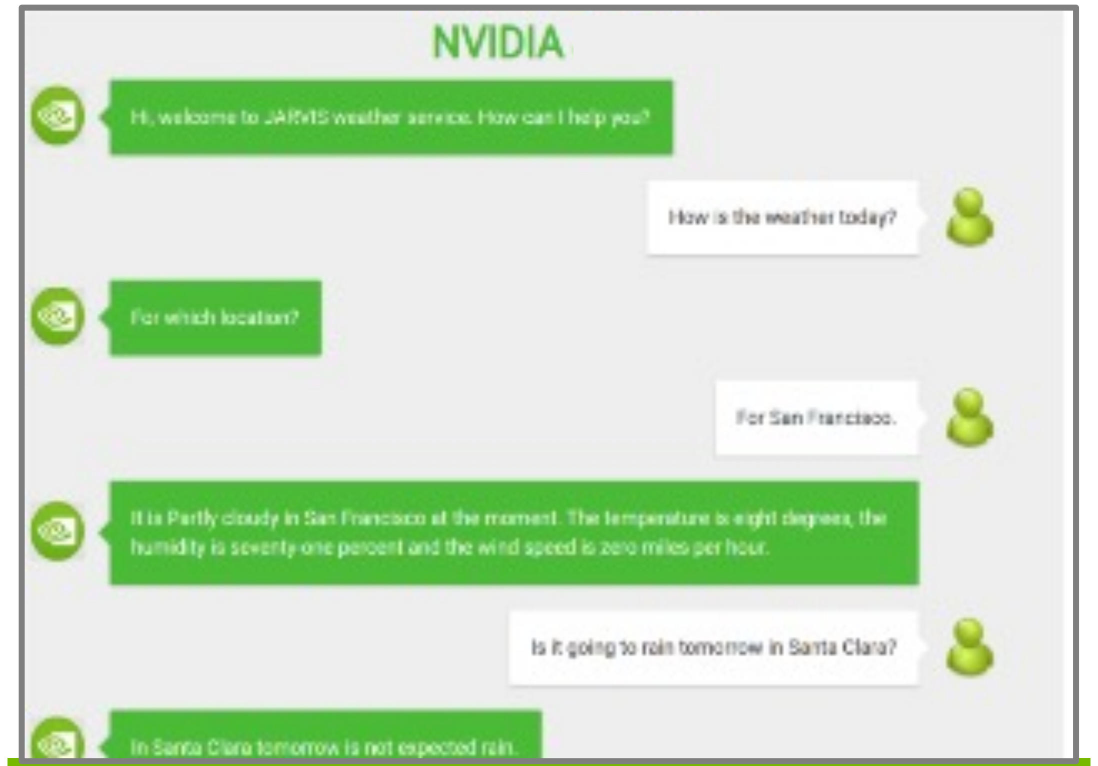
Visual Diarization

Transcribe multi-user multi-context conversations



Look To Talk

Wait for gaze before triggering AI assistant



Hi, welcome to JARVIS weather service. How can I help you?

How is the weather today?

For which location?

For San Francisco.

It is Partly cloudy in San Francisco at the moment. The temperature is eight degrees, the humidity is seventy one percent and the wind speed is zero miles per hour.

Is it going to rain tomorrow in Santa Clara?

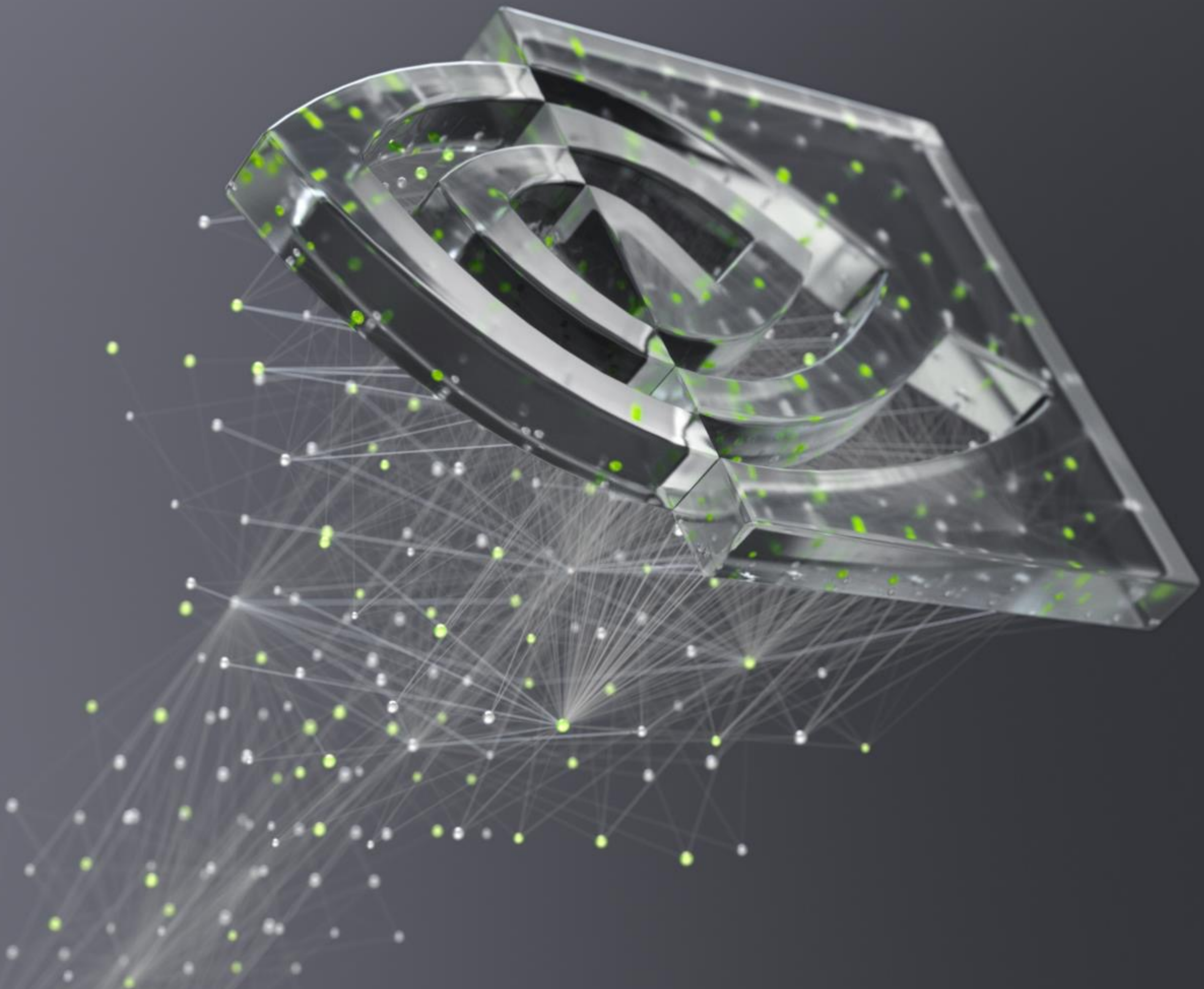
In Santa Clara tomorrow is not expected rain.

Virtual Assistant

End-to-end conversational AI system

Part 3: Production Deployment

- Lecture
  - Model Selection
  - Post-Training Optimization
  - Product Quantization
  - Knowledge Distillation
  - Model Code Efficiency
  - Model Serving
  - Building the Application
- Lab
  - Exporting the Model
  - Hosting the Model
  - Server Performance
  - Using the Model



NVIDIA

DEEP
LEARNING
INSTITUTE