ProSE: The Architecture and Design of a Protein Discovery Engine

Eyes Robson
University of California, Berkeley
Berkeley, California, USA

Ceyu Xu
Duke University
Durham, North Carolina, USA

Lisa Wu Wills
Duke University
Durham, North Carolina, USA

ASPLOS 2022

Presented by Gianluca Figini
28/4/2022

Outline

- Executive summary
- Background
  - → Proteins
  - → Natural Language Processing and Protein Design Applications
  - → BERT and Self-Attention
- BERT Profiling
- ProSE architecture
- ProSE design
- Performance evaluation
- Strengths and weaknesses
- Discussion

Executive summary

Problem: Lack of specialized hardware for execution of protein discovery algorithms

Special function not supported

Element-wise operations not optimized

Motivation: Reduce costs for protein discovery / validation processes

Determine drug-target affinity

Determine protein structure

Goal: Create a hardware accelerator to efficiently tackle these problem

Power and area efficient

Support for specialized functions

Applicable to multiple problems

Evaluation:

Up to 7x speedup with respect to non-specialized GPU and TPU

Up to 2 order of magnitude more power efficient

Background - Proteins

- Building blocks of a cell

  Involved in:

  → Structure of cells

  → DNA replication

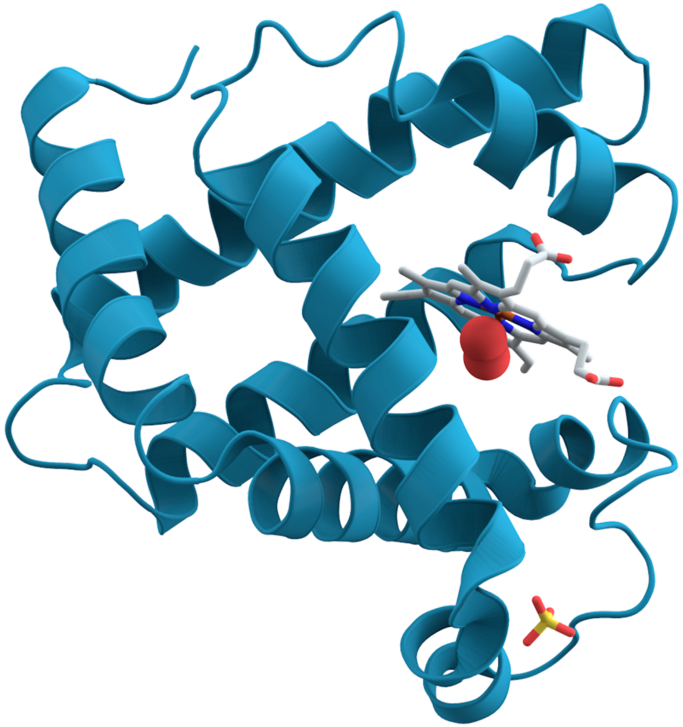
  → Transportation of molecules

  → Triggering / inhibiting reactions

  → …

- Chains of amino acids

  → Code formed by 20 amino acids

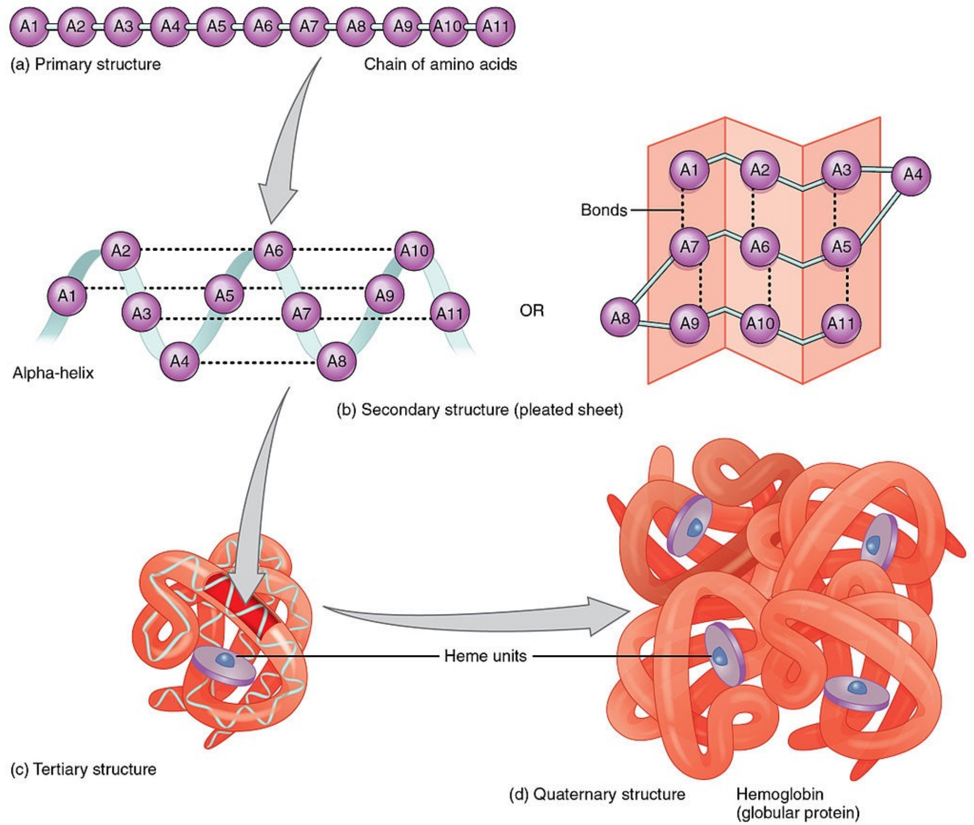


Wikipedia

Background - Proteins

Non-covalent interactions between amino acids generates a three dimensional structure

→ Correct structure is essential to function
→ Does not change the amino acids sequence
→ Very difficult to detect
→ **Well-defined** (Anfinsen *et al.*, 1961)



Wikipedia

Background – Natural Language Processing

Natural Language Processing

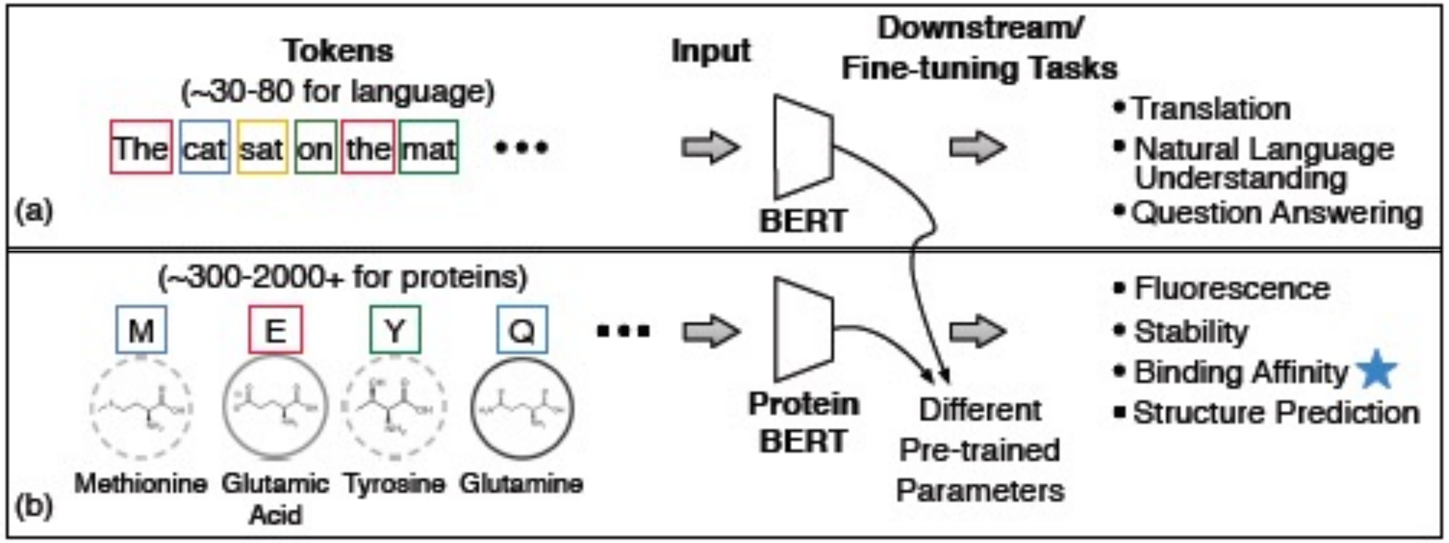
- Next sentence prediction
- Translation
- Question answering
- …

Protein Design Applications

- Fluorescence
- Stability
- Binding Affinity
- Structure Prediction

Main differences:

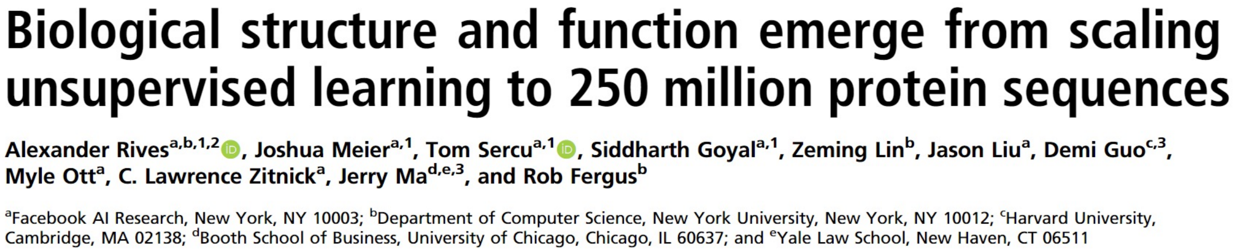
- Pre-trained parameters
- Input length



Background – Natural Language Processing

- Success of NLP in protein modelling
- Can lead to a cut down of the cost of drug discovery/validation

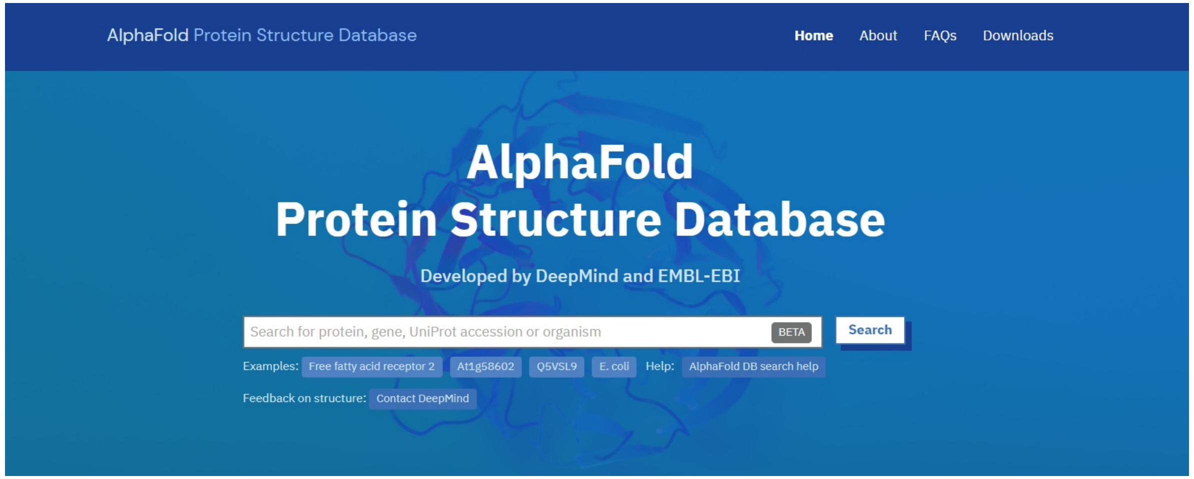
→ $80 B per year
→ ~90% failure rate
→ 12 years for research and validation iter



Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences

Alexander Rives[a,b,1,2], Joshua Meier[a,1], Tom Sercu[a,1], Siddharth Goyal[a,1], Zeming Lin[b], Jason Liu[a], Demi Guo[c,3], Myle Ott[a], C. Lawrence Zitnick[a], Jerry Ma[d,e,3], and Rob Fergus[b]

[a]Facebook AI Research, New York, NY 10003; [b]Department of Computer Science, New York University, New York, NY 10012; [c]Harvard University, Cambridge, MA 02138; [d]Booth School of Business, University of Chicago, Chicago, IL 60637; and [e]Yale Law School, New Haven, CT 06511



AlphaFold Protein Structure Database

Home About FAQs Downloads

AlphaFold
Protein Structure Database

Developed by DeepMind and EMBL-EBI

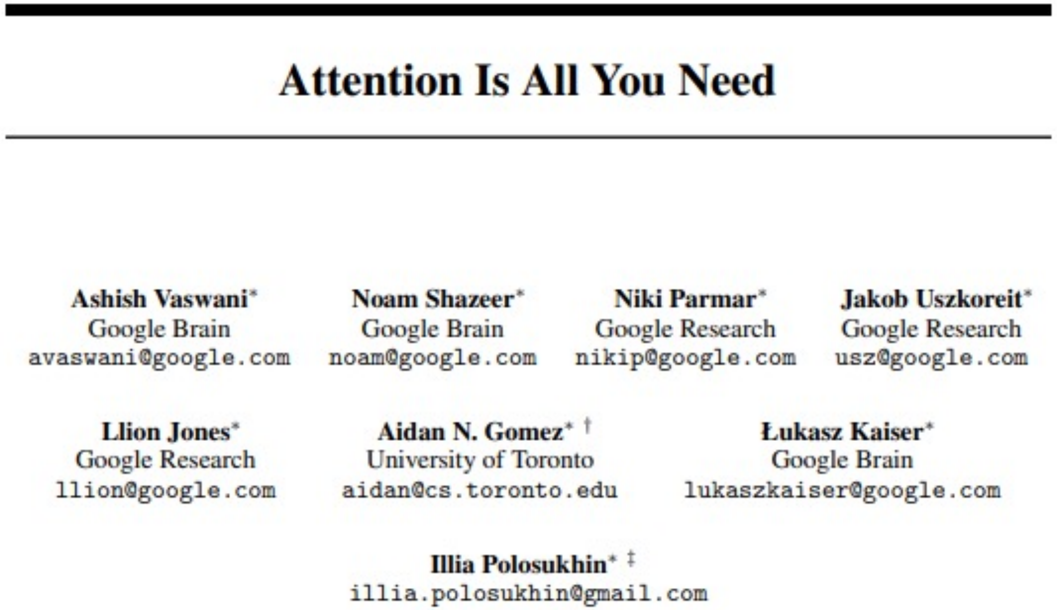
Examples: Free fatty acid receptor 2 At1g58602 Q5VSL9 E. coli Help: AlphaFold DB search help

Feedback on structure: Contact DeepMind

Background - BERT

BERT: Bidirectional Encoder Representation for Transormers

- Technique invented by Google researcher Jacob Devlin in 2018
- Implemented in the Google search engine in 2019
- Can be pre-trained and fine-tuned
- **Transformer based**



Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

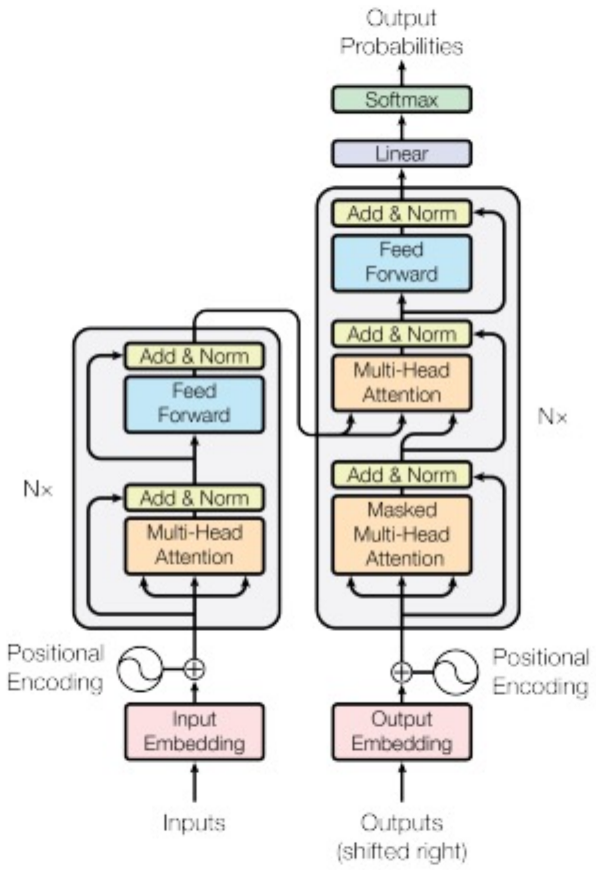
Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

Background - BERT

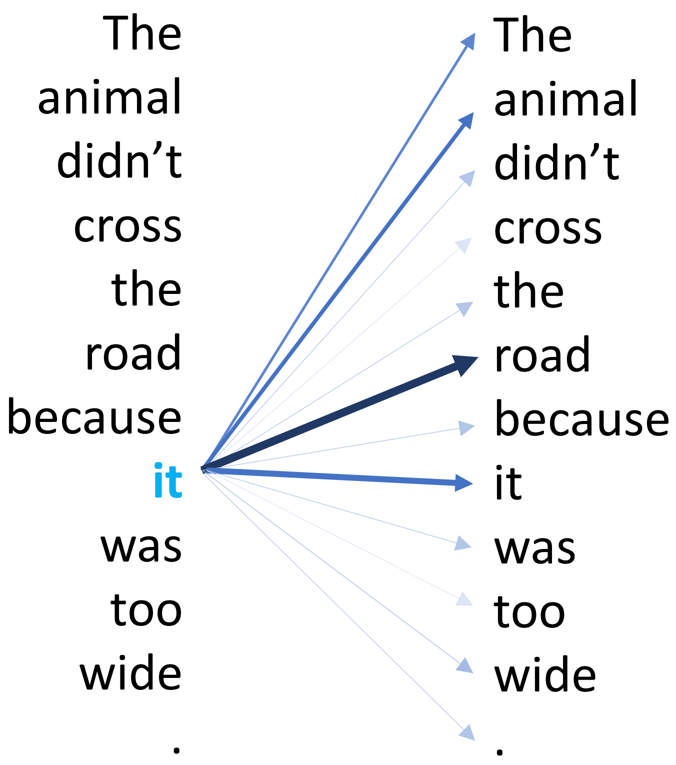


Source: Vaswani and al, «Attention is all you need»., 2017

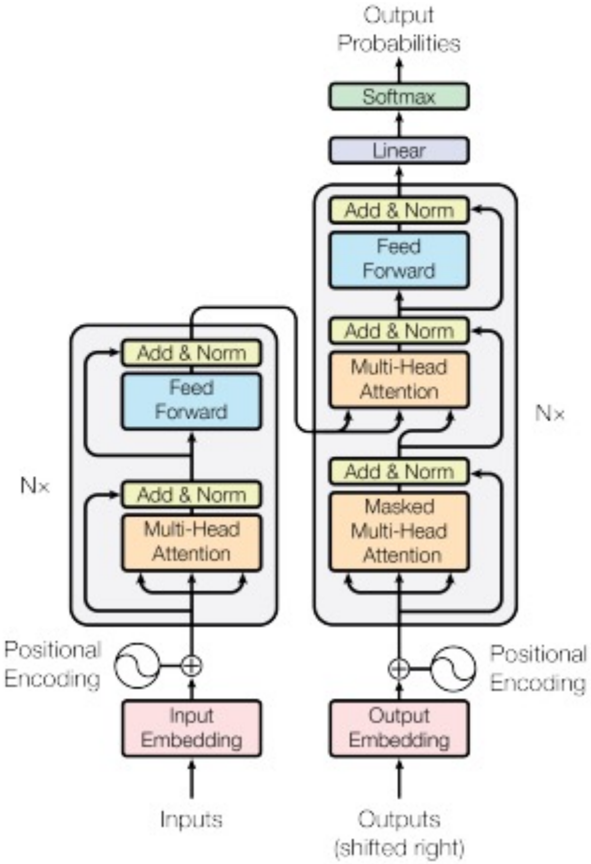
Self-attention model

The → $\overrightarrow{q_1}, \overrightarrow{k_1}, \overrightarrow{v_1}$
animal → $\overrightarrow{q_2}, \overrightarrow{k_2}, \overrightarrow{v_2}$
didn’t → $\overrightarrow{q_3}, \overrightarrow{k_3}, \overrightarrow{v_3}$
cross → $\overrightarrow{q_4}, \overrightarrow{k_4}, \overrightarrow{v_4}$
the → $\overrightarrow{q_5}, \overrightarrow{k_5}, \overrightarrow{v_5}$
road → $\overrightarrow{q_6}, \overrightarrow{k_6}, \overrightarrow{v_6}$
because → $\overrightarrow{q_7}, \overrightarrow{k_7}, \overrightarrow{v_7}$
it → $\overrightarrow{q_8}, \overrightarrow{k_8}, \overrightarrow{v_8}$
was → $\overrightarrow{q_9}, \overrightarrow{k_9}, \overrightarrow{v_9}$
too → $\overrightarrow{q_{10}}, \overrightarrow{k_{10}}, \overrightarrow{v_{10}}$
wide → $\overrightarrow{q_{11}}, \overrightarrow{k_{11}}, \overrightarrow{v_{11}}$
. → $\overrightarrow{q_{12}}, \overrightarrow{k_{12}}, \overrightarrow{v_{12}}$

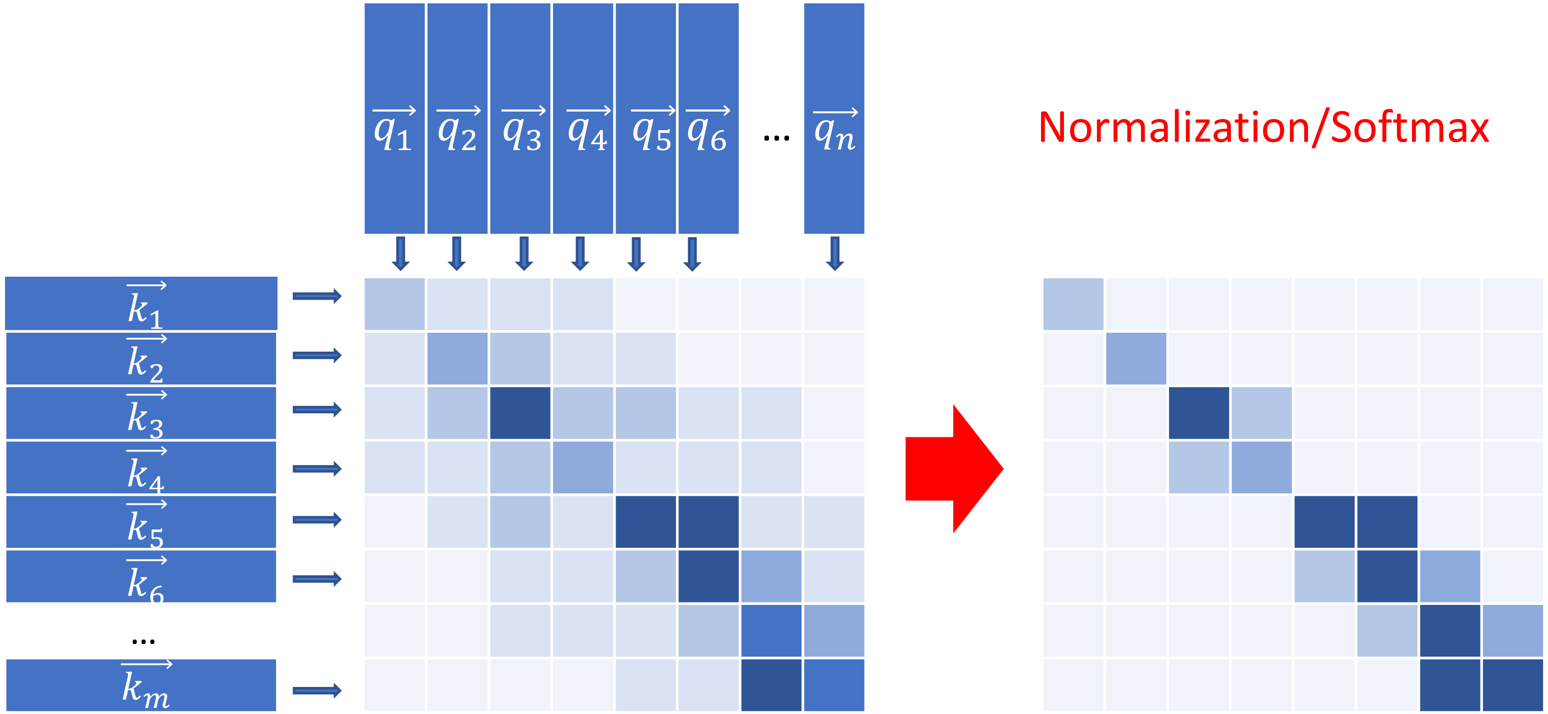
$(\overrightarrow{q_8} \cdot \overrightarrow{k_i})\overrightarrow{v_i}$



Background - BERT



Source: Vaswani and al, «Attention is all you need»., 2017



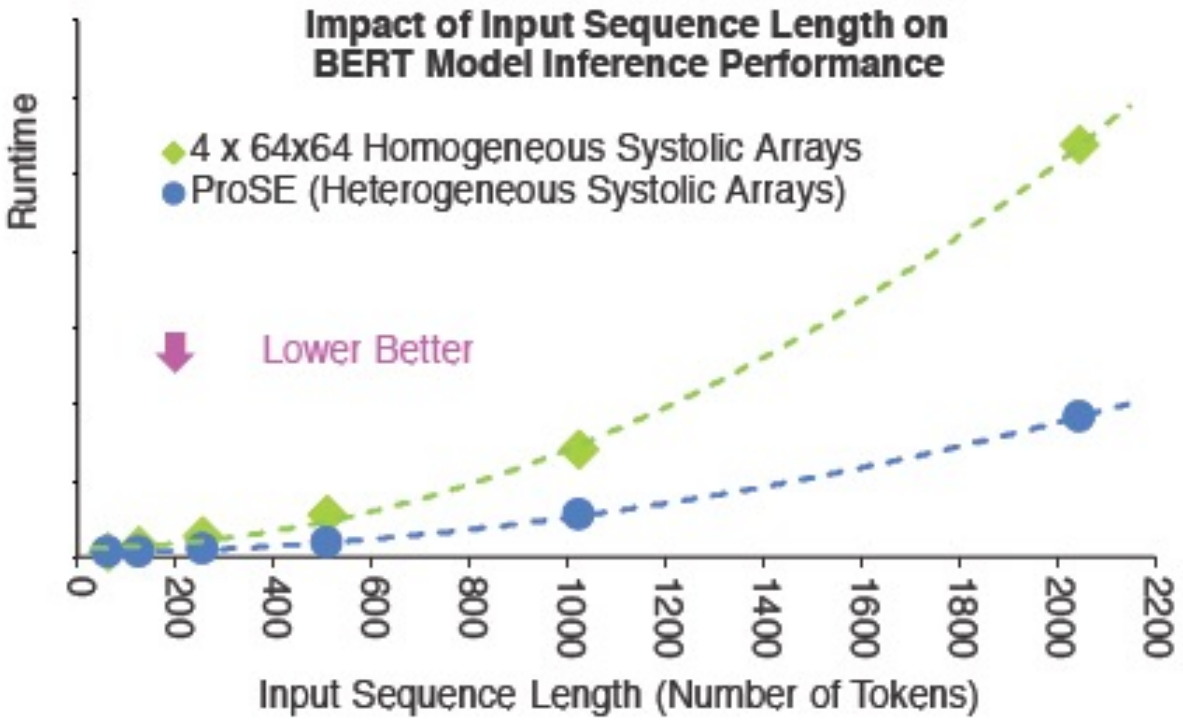
Normalization/Softmax

Outline

- Executive summary
- Background
  - → Proteins
  - → Natural Language Processing and Protein Design Applications
  - → BERT and Self-Attention
- **BERT Profiling**
- ProSE architecture
- ProSE design
- Performance evaluation
- Strengths and weaknesses
- Discussion

BERT profiling

BERT execution time and memory footprint increases **quadratically** as function of input length!

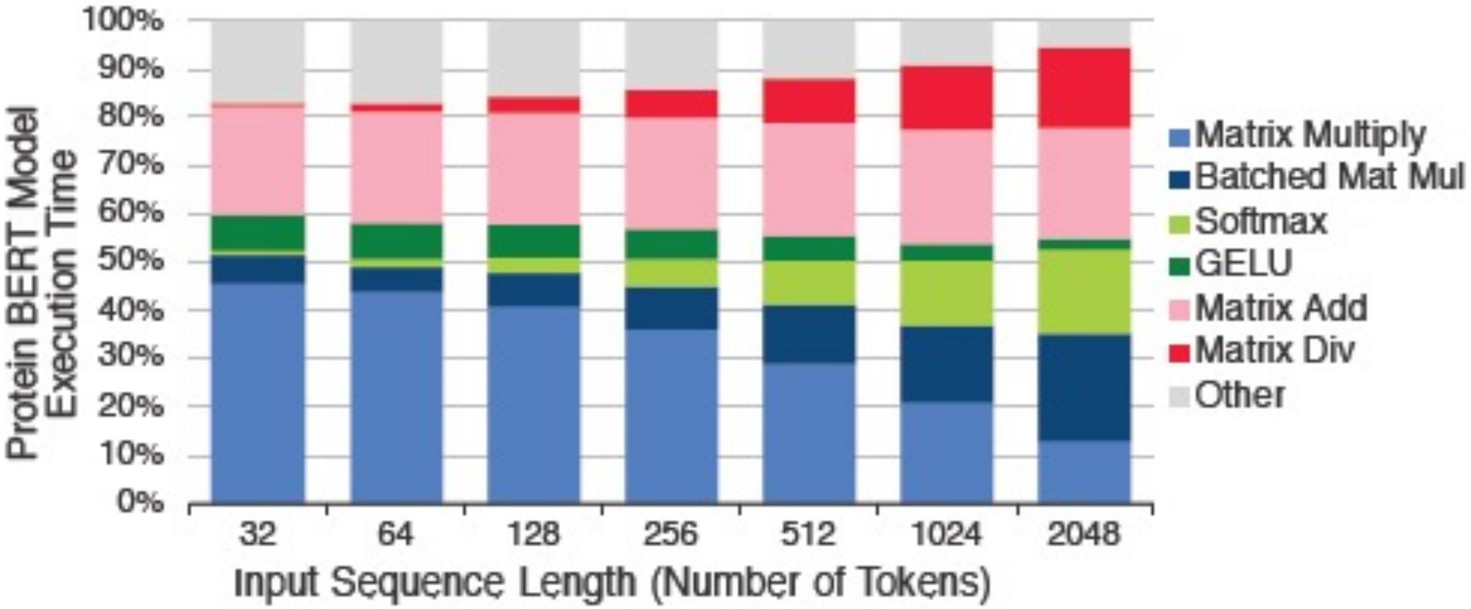


BERT profiling

The distribution of execution time changes with longer input sequence

→ Time spent evaluating **element-wise** operations increases

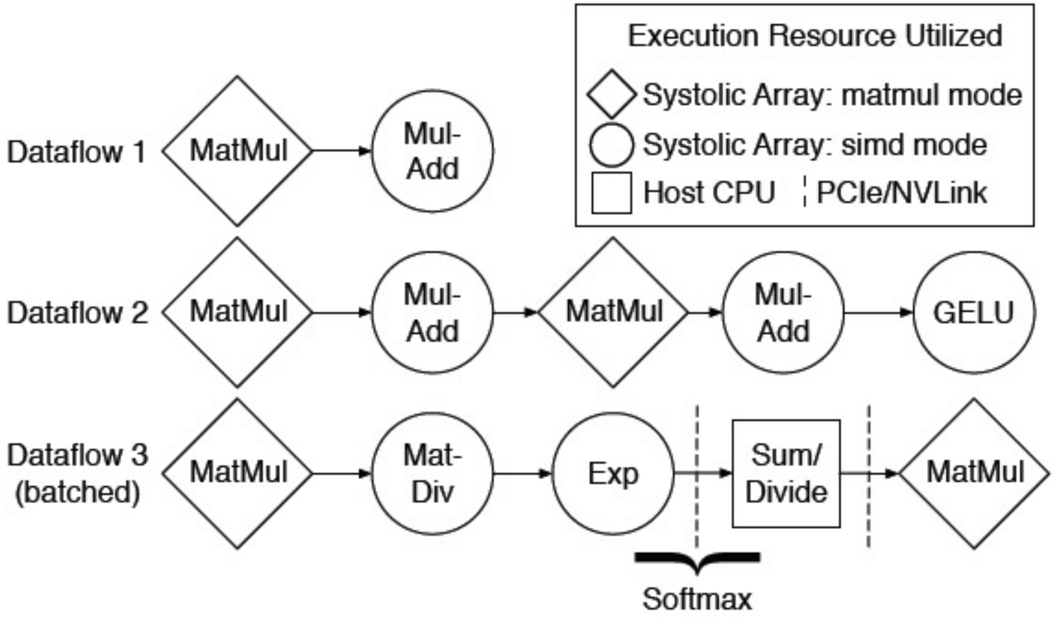
→ Time spent evaluating matrix multiplications decreases

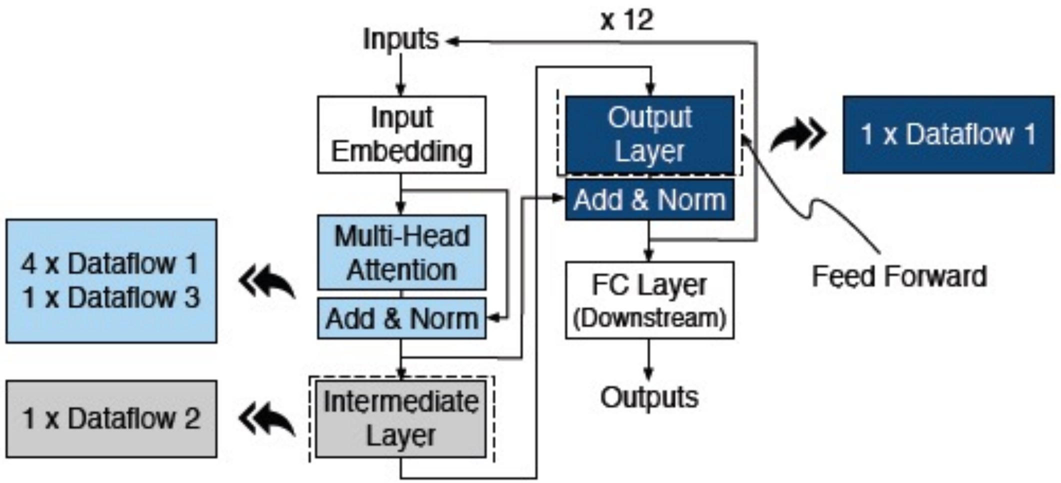


BERT profiling

BERT model programs require support for special functions:

- → GELU: Gaussian Error Linear Unit
- → Exp



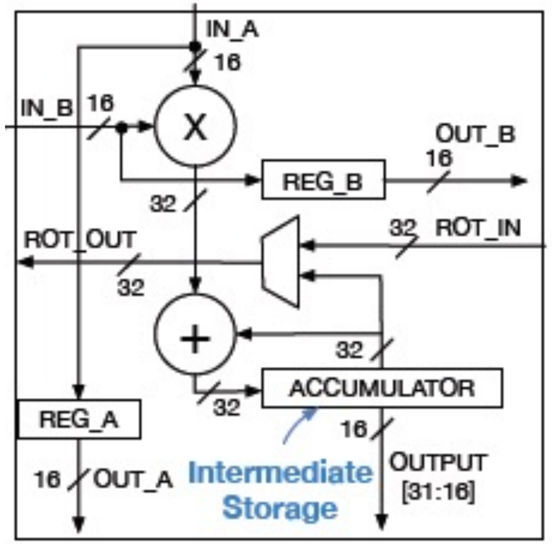


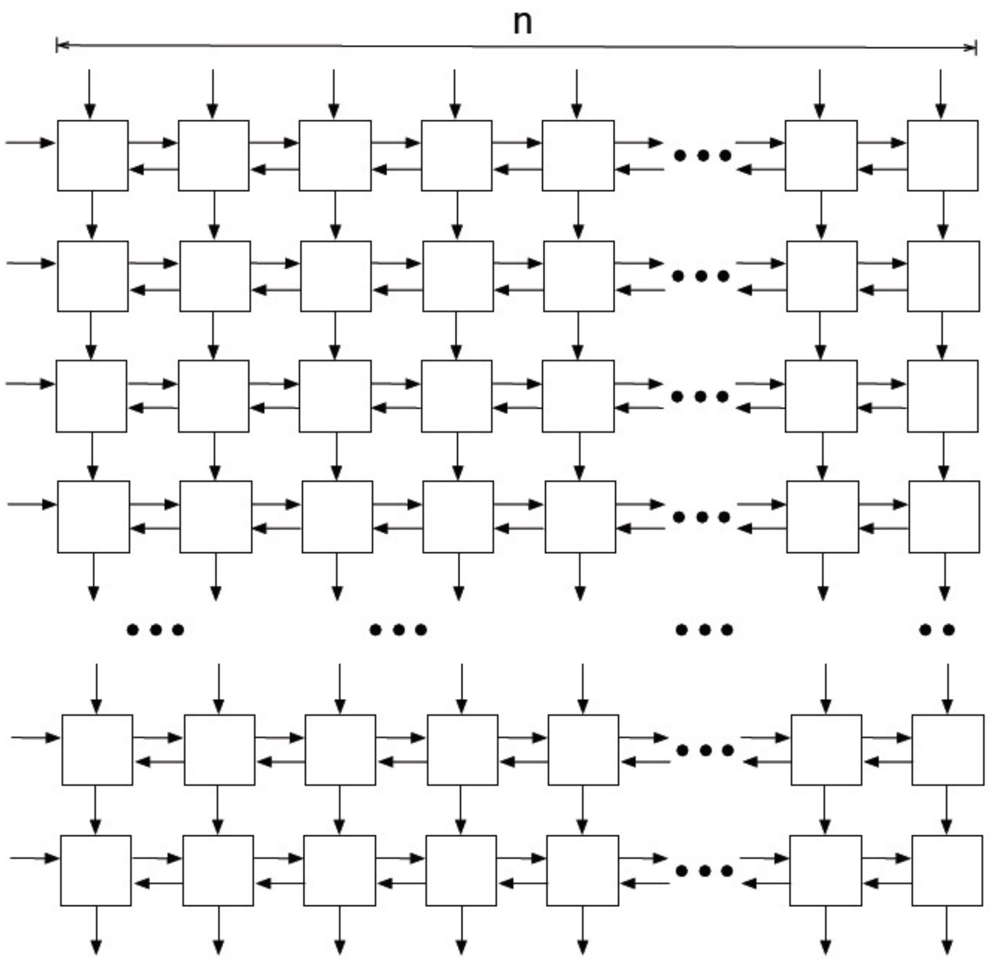
Outline

- Executive summary
- Background
  - → Proteins
  - → Natural Language Processing and Protein Design Applications
  - → BERT and Self-Attention
- BERT Profiling
- **ProSE architecture**
- ProSE design
- Performance evaluation
- Strengths and weaknesses
- Discussion

ProSE architecture

Left-rotation-capable
output-stationary
streaming
systolic array

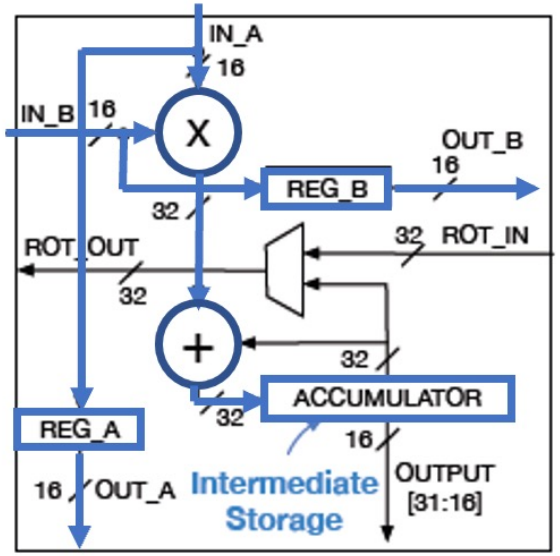


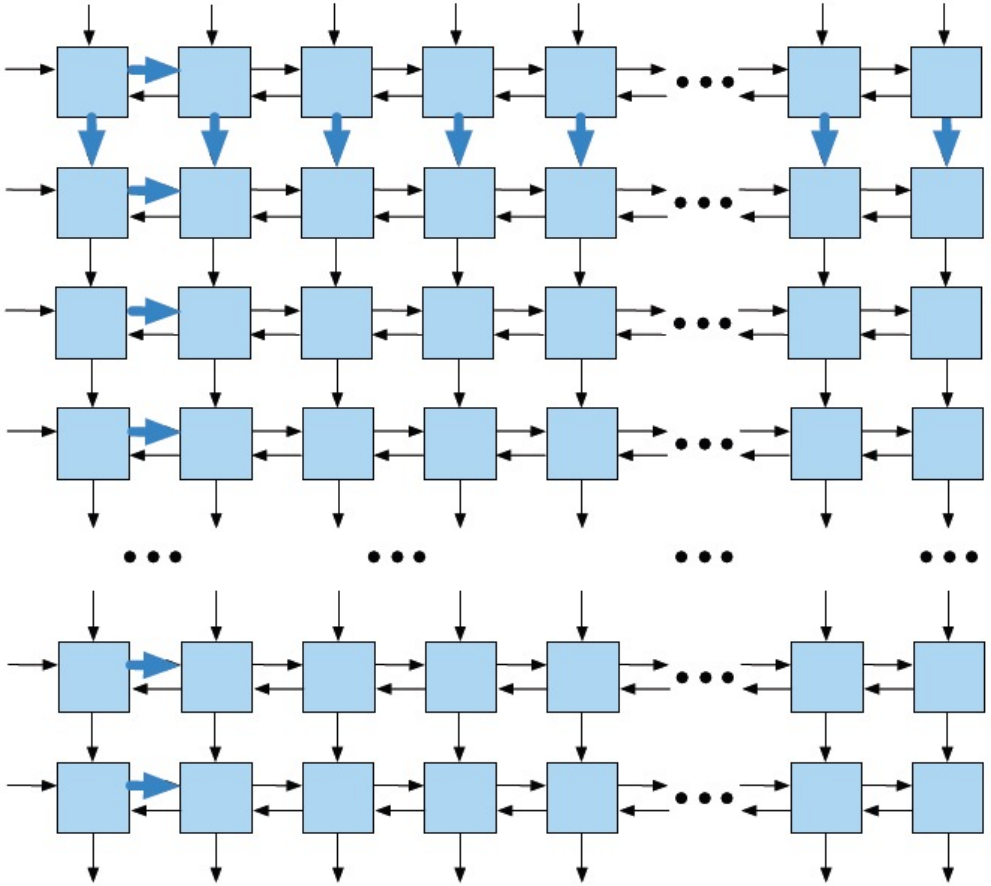


ProSE architecture

Left-rotation-capable
output-stationary
streaming
systolic array

MatMult mode

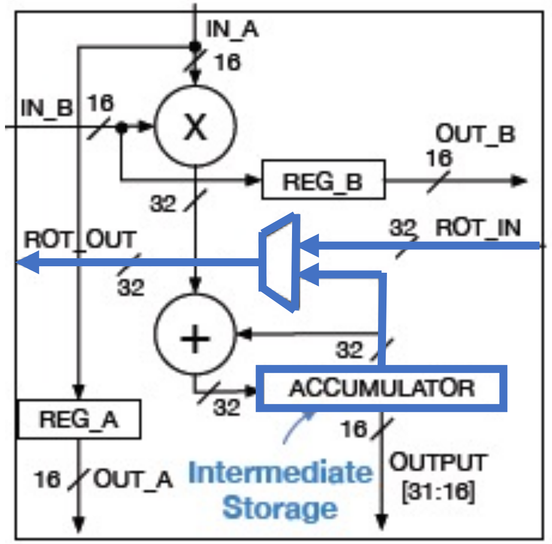


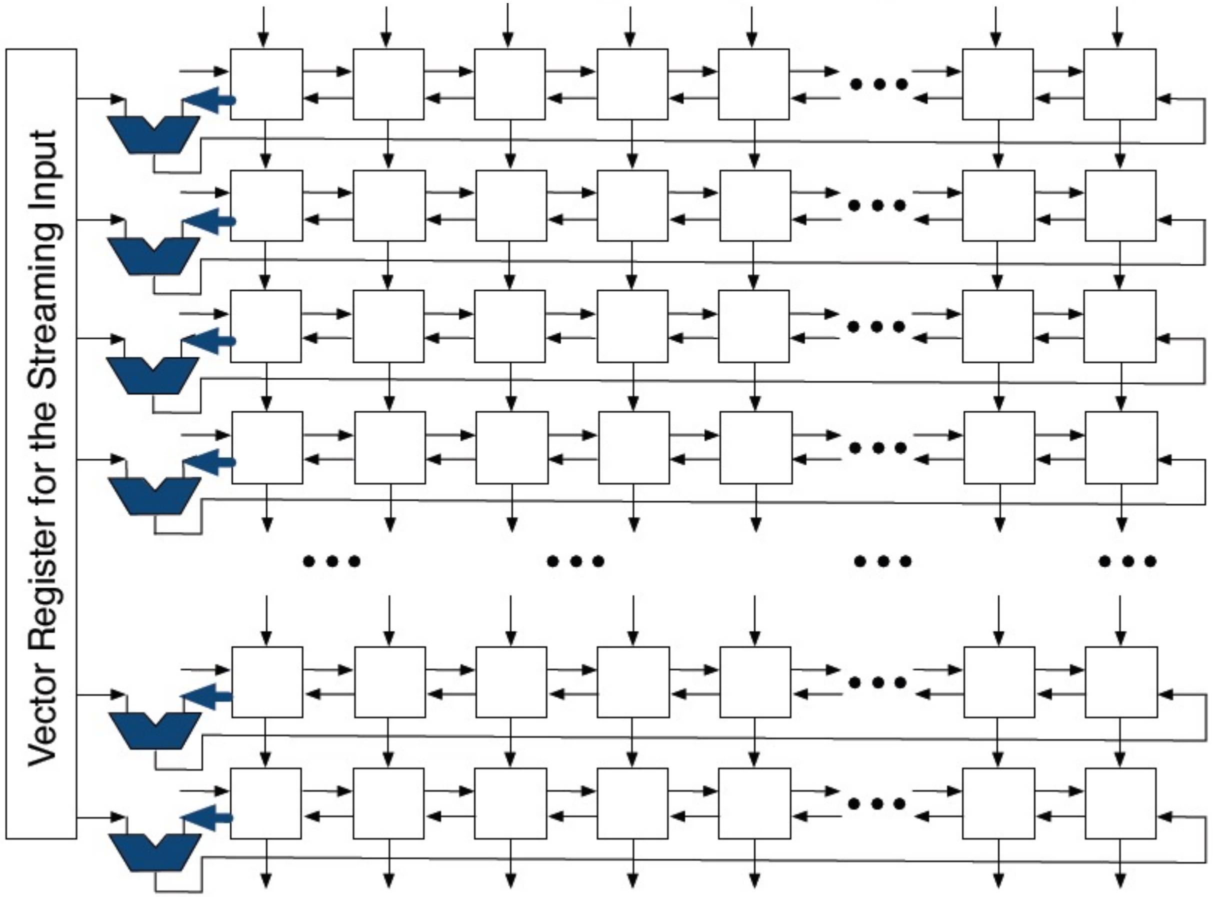


ProSE architecture

Left-rotation-capable
output-stationary
streaming
systolic array

SIMD mode





ProSE architecture vs TPUv2 architecture

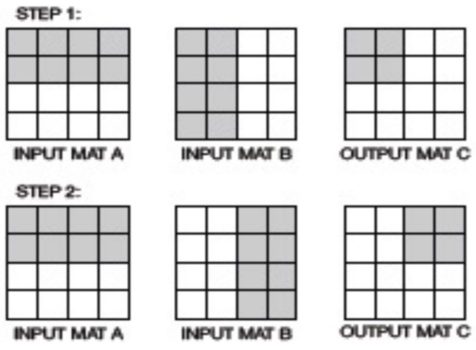
Streaming from the Host vs Unified Buffer

- + Power saving
- + Reduced latency
- + Simplified hardware
- - Bandwidth between host and systolic array has to be managed
- - Requires specialized software to dissemble/reassemble matrices

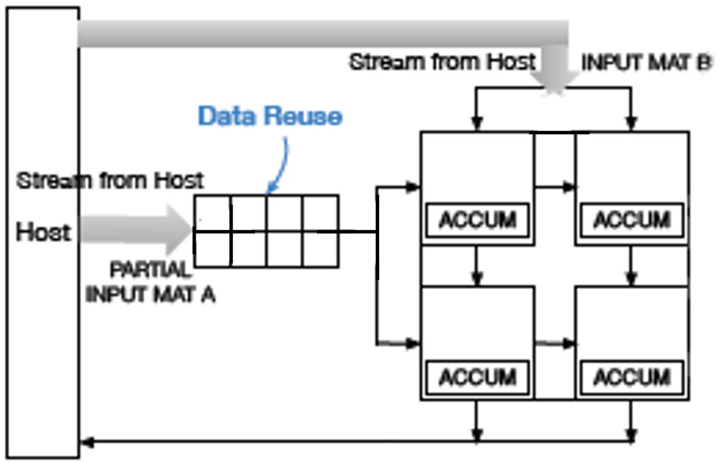
Output-stationary vs Weight-stationary

- + Matrices can be streamed at the same time
- - Does not optimize for minimal weight readings

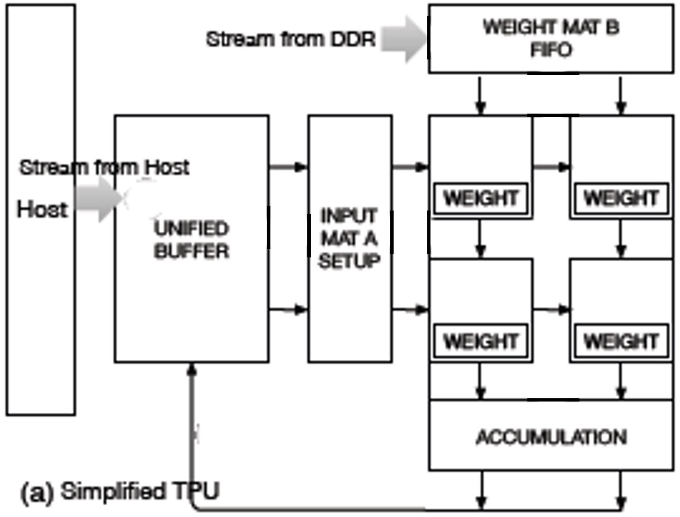
Data Reuse Buffer



ProSE



TPU



(a) Simplified TPU

ProSE architecture vs TPUv2 architecture

Performing MulAdd $a \times A + B$

ProSE:

1. Loads a into the scalar register
2. Loads A into the systolic array
3. Loads B column-wise into the vector register

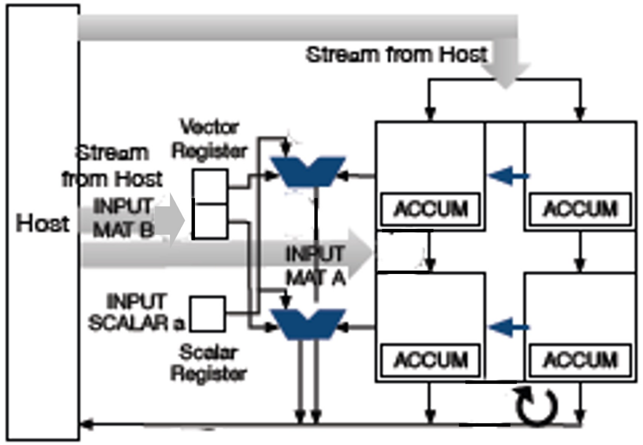
+ Only requires one matrix to be loaded in the systolic array

TPU:

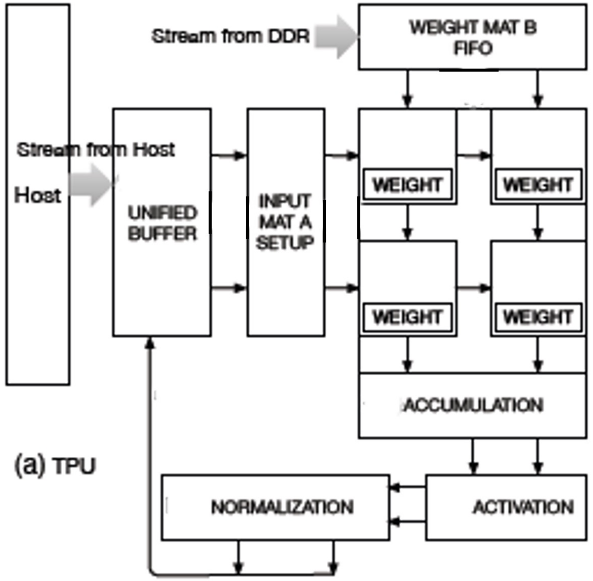
1. Passes A through the systolic array
2. Normalizes it to a in the normalization stage
3. Passes B through the systolic array and stores it in the accumulation stage
4. Passes $a \times A$ through the array and accumulates it to B in the accumulation stage

- Requires three matrices to be loaded in the systolic array

ProSE



TPU



(a) TPU

ProSE architecture

Three types of systolic array based on SIMD calculation capability:

M-Type: MatMult and SIMD ALU operations

E-Type: MatMult, SIMD ALU operations and Exponential functions

G-Type: MatMult, SIMD ALU operations and GELU special functions

GELU: Gaussian Error Linear Unit

ProSE architecture

Special functions

Implemented using two-level lookup tables

$GELU(x)$ is evaluated:

- Approximated to zero for $x < -4$
- Using the lookup table for $-4 \leq x \leq 3$
- Approximated by a linear function for $x > 3$

This preserves the precision of the *bfloat16* datatype

One copy of this table is stored per each special ALU

+ Better performance

- Larger area

Outline

- Executive summary
- Background
  - → Proteins
  - → Natural Language Processing and Protein Design Applications
  - → BERT and Self-Attention
- BERT Profiling
- ProSE architecture
- **ProSE design**
- Performance evaluation
- Strengths and weaknesses
- Discussion

ProSE design

Implementation methodology

- PyTorch frontend
  → Instructed to produce raw sequences of backend tensor and operations

- Connection to the host with 6 lanes at 45 GB/s each

- Matrix multiplications are executed with a 1.6 GHz clock frequency
- SIMD/GELU/Exp-capable systolic array run at 800 MHz

- Compiled in Verilog

ProSE design

We now want to maximise the performance of these systolic arrays.

Problem: Rules are different depending on the mode the array is operation in:

Matmult mode: Big arrays minimise the number of blocks the matrix has to be divided into

SIMD mode: Small arrays maximise the ratio ALUs to PEs

→ **Solution:** heterogeneous systolic arrays

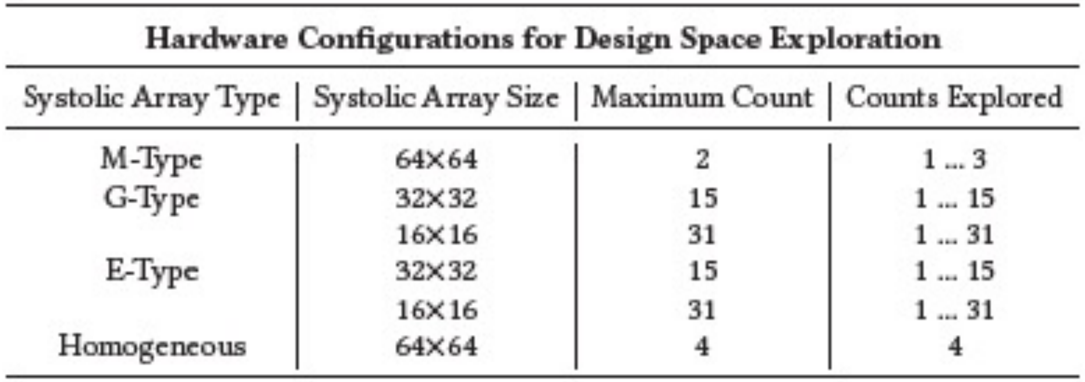
ProSE design

Different configuration are tested

→ **Number of PEs constant**
(equivalent to a TPU 128x128 systolic array)

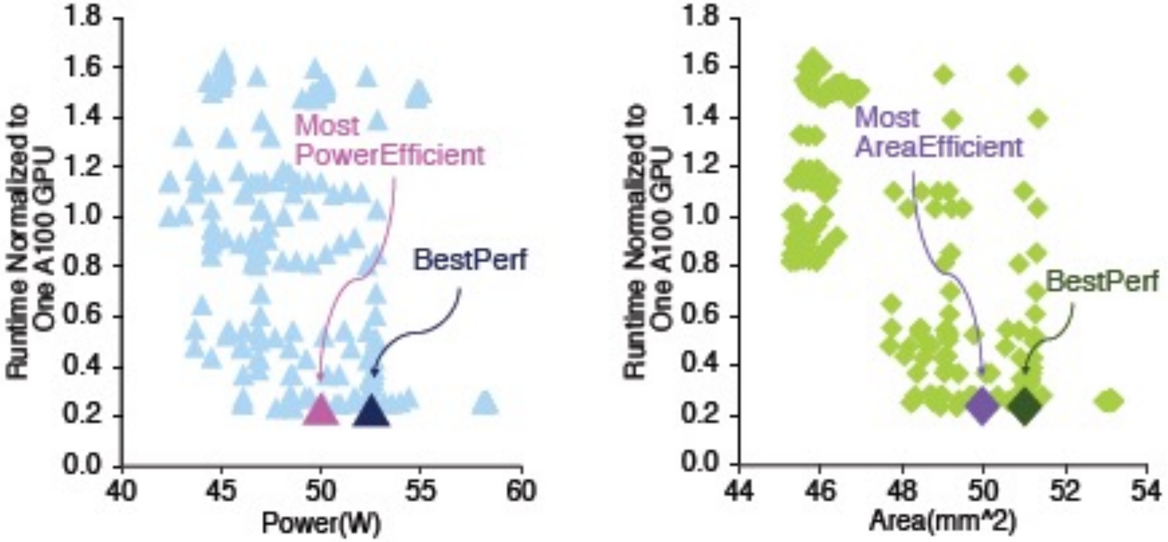
→ Every configuration must have a count of 1 or more

→ The number of lanes assigned to each array type is swept as part of the design space exploration



Hardware Configurations for Design Space Exploration

Systolic Array Type	Systolic Array Size	Maximum Count	Counts Explored
M-Type	64×64	2	1 ... 3
G-Type	32×32	15	1 ... 15
	16×16	31	1 ... 31
E-Type	32×32	15	1 ... 15
	16×16	31	1 ... 31
Homogeneous	64×64	4	4



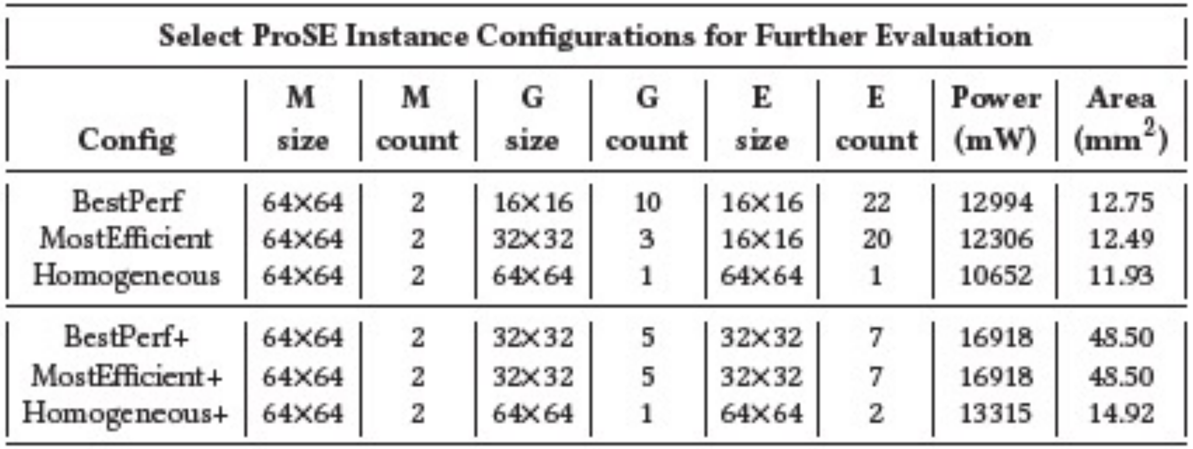
ProSE design

Best configurations

→ MostPowerEfficient and MostAreaEfficient are the same configuration, that is called MostEfficient

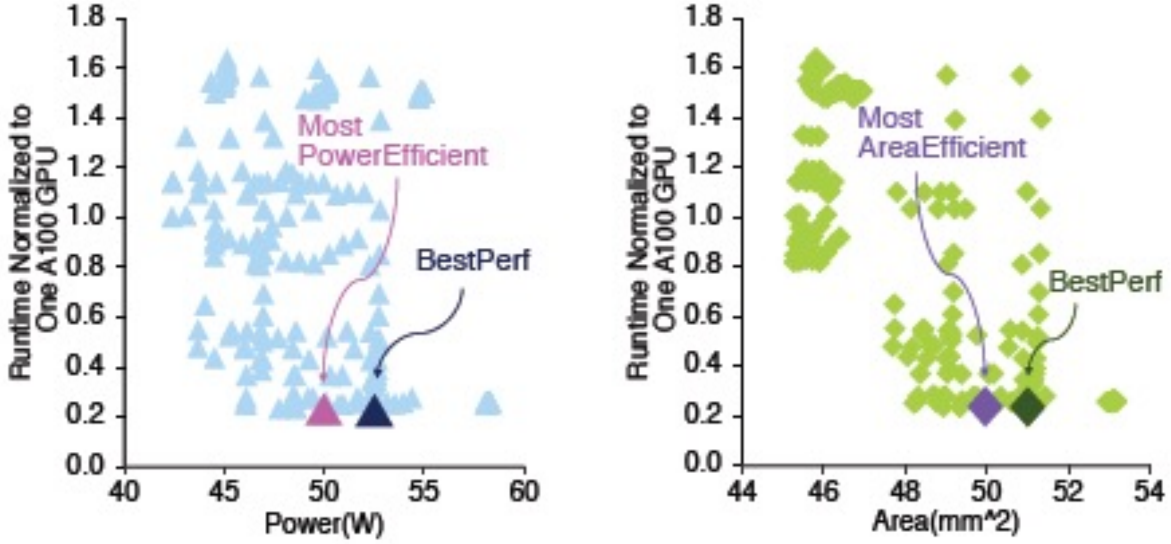
Also configurations with 20k PEs are tested

→ These configurations are not compute-bound until 360 GB/s



Select ProSE Instance Configurations for Further Evaluation

	Config	M size	M count	G size	G count	E size	E count	Power (mW)	Area (mm^2)
16K PEs	BestPerf	64×64	2	16×16	10	16×16	22	12994	12.75
	MostEfficient	64×64	2	32×32	3	16×16	20	12306	12.49
	Homogeneous	64×64	2	64×64	1	64×64	1	10652	11.93
20K PEs	BestPerf+	64×64	2	32×32	5	32×32	7	16918	48.50
	MostEfficient+	64×64	2	32×32	5	32×32	7	16918	48.50
	Homogeneous+	64×64	2	64×64	1	64×64	2	13315	14.92



Outline

- Executive summary
- Background
  - → Proteins
  - → Natural Language Processing and Protein Design Applications
  - → BERT and Self-Attention
- BERT Profiling
- ProSE architecture
- ProSE design
- **Performance evaluation**
- Strengths and weaknesses
- Discussion

Performance evaluation

→ Input sequence of 512 tokens

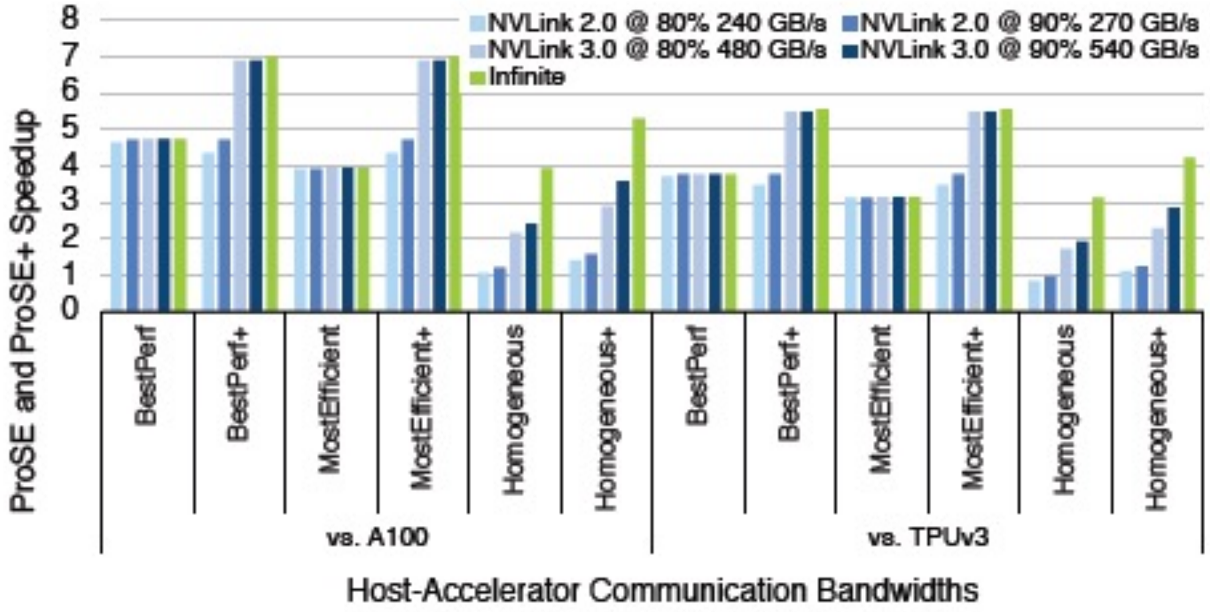
ProSE speed up Protein Design Application up to

- **4.5x** with 16K PEs
- **7x** with 20K PEs

compared with a Nvidia A100 GPU and up to

- **4x** with 16K PEs
- **5.5x** with 20K PEs

compared with a Google TPUv3.



Performance evaluation

→ Input sequence of 512 tokens

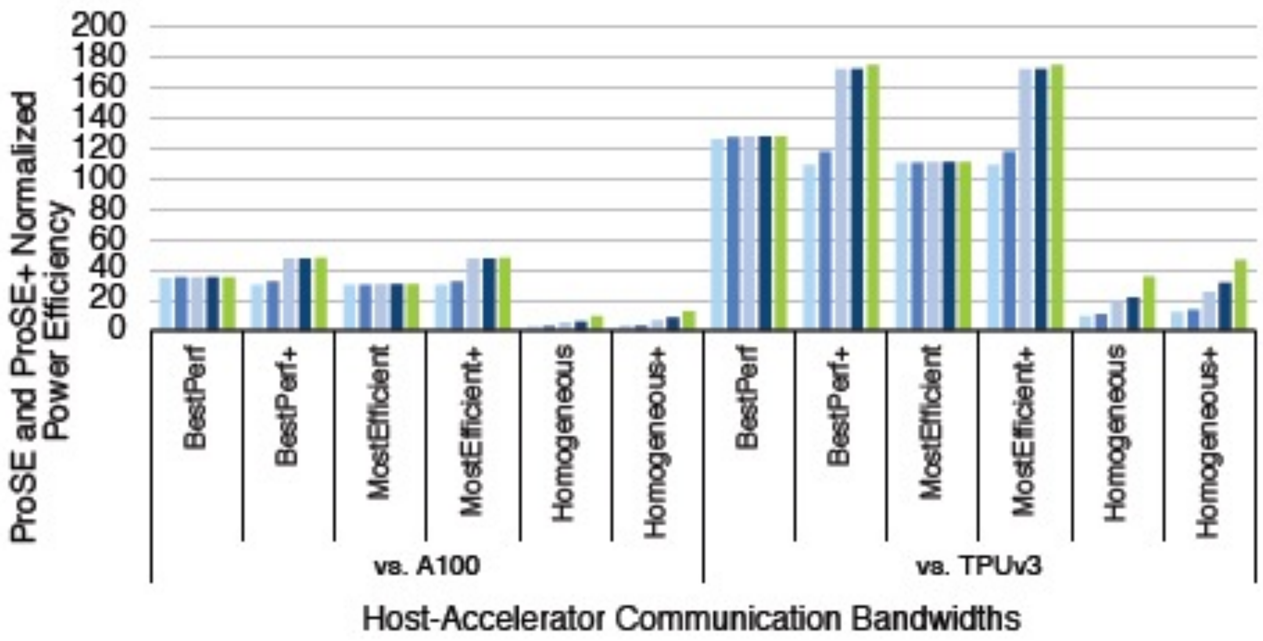
ProSE power consumption is up to

50x lower

compared to a Nvidia A100 GPU and up to

170x lower

compared to a Google TPUv3.



Outline

- Executive summary
- Background
  - → Proteins
  - → Natural Language Processing and Protein Design Applications
  - → BERT and Self-Attention
- BERT Profiling
- ProSE architecture
- ProSE design
- Performance evaluation
- **Strengths and weaknesses**
- Discussion

Strengths

- First publication proposing a systolic engine implementing special functions for BERT model algorithms
- Provides a system-wide implementation of the model
- Very comprehensible also while explaining complicated topics
- Does not sacrifice generality
- Results are presented clearly, evaluation is done in a very extensive way

Weaknesses

- Implementation of the exp LUT is ambiguous
- Does not mention whether every problem addressed delivers the same speedup / power efficiency
- Software side is barely spoken about
- Details

  For GELU, we designed the lookup table such that it only computes the output when the exponent is between -4 and 3 […]when the input is with an exponent smaller than -3, it can be approximated as 0. When the input is with an exponent larger than 4, it can be approximated by a linear function.

Outline

- Executive summary
- Background
  - → Proteins
  - → Natural Language Processing and Protein Design Applications
  - → BERT and Self-Attention
- BERT Profiling
- ProSE architecture
- ProSE design
- Performance evaluation
- Strengths and weaknesses
- **Discussion**

Discussion

Another article[1] proposes an accelerator based on quantization of data that delivers a 1.17x speedup and a 12x power efficiency.

- → Uses a series of vector-matrix multiplication PU
- → Approximates weights to 4 bits and other values to 8 bit
- → Features a module to combine 8 and 4 bit multiplications
- → Features a input/output buffer

Do you think these two approaches could coexist in a single device?

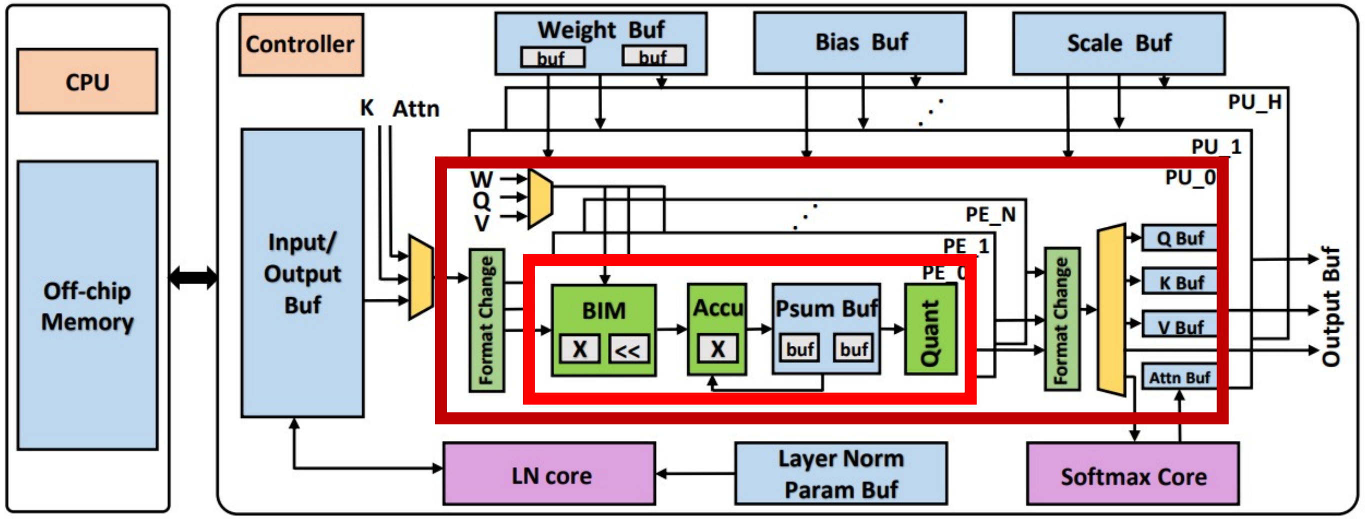


Fig. 2. The overall architecture of the proposed accelerator for fully quantized BERT.

[1] Zejian Liu and al., "Hardware Acceleration of Fully Quantized BERT for Efficient Natural Language Processing", 2021

Discussion

- Applications to other fields?

  DNA/RNA analysis, …

- More support for the SIMD ALU instructions

  In this architecture, the result of an ALU operation are streamed to the host, would it be beneficial if they were streamed back into the array?

- Communication between different arrays on chip?

- Composition of older inventions?

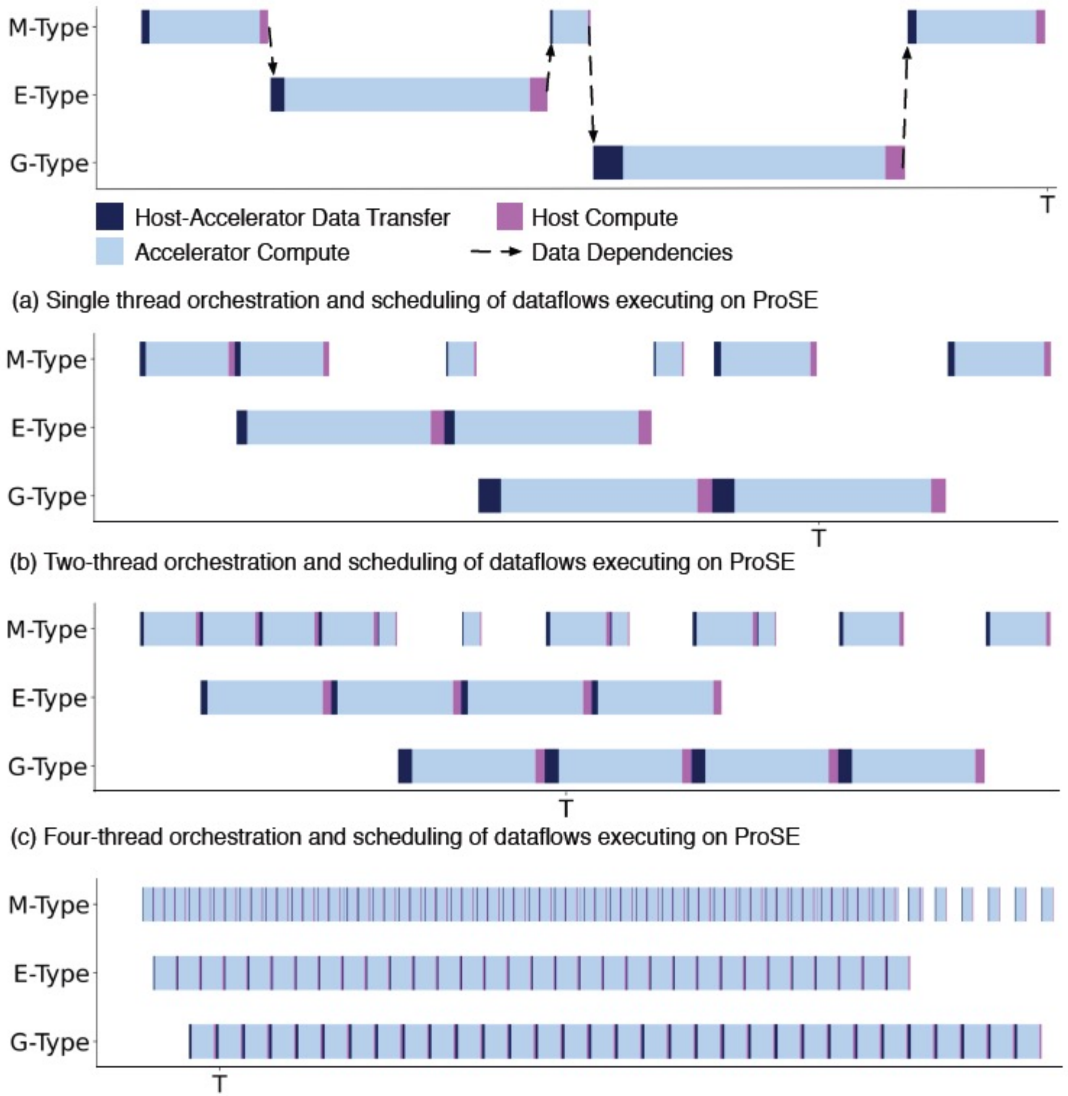
Thank you for your attention

Additional material

Threading in ProSE

Execution model chosen through experimentation:

32 threads



(a) Single thread orchestration and scheduling of dataflows executing on ProSE

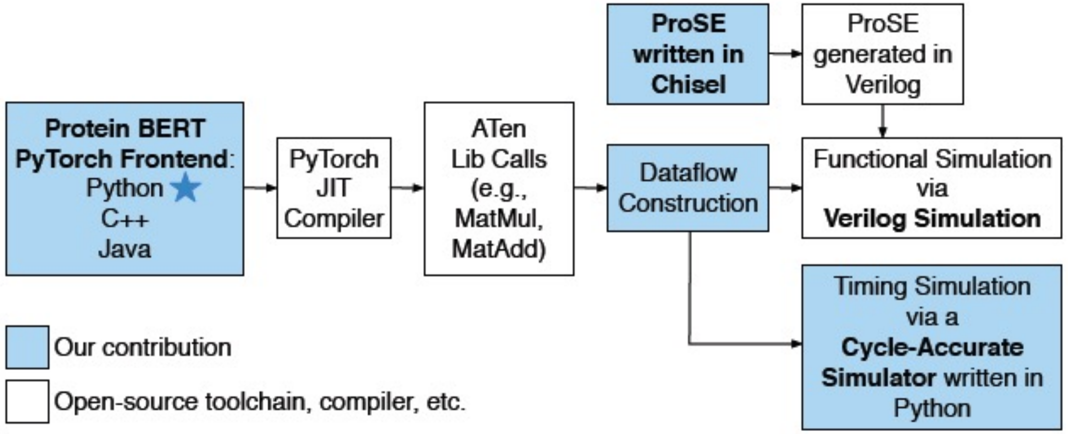
(b) Two-thread orchestration and scheduling of dataflows executing on ProSE

(c) Four-thread orchestration and scheduling of dataflows executing on ProSE

(d) 32-thread orchestration and scheduling of dataflows executing on ProSE

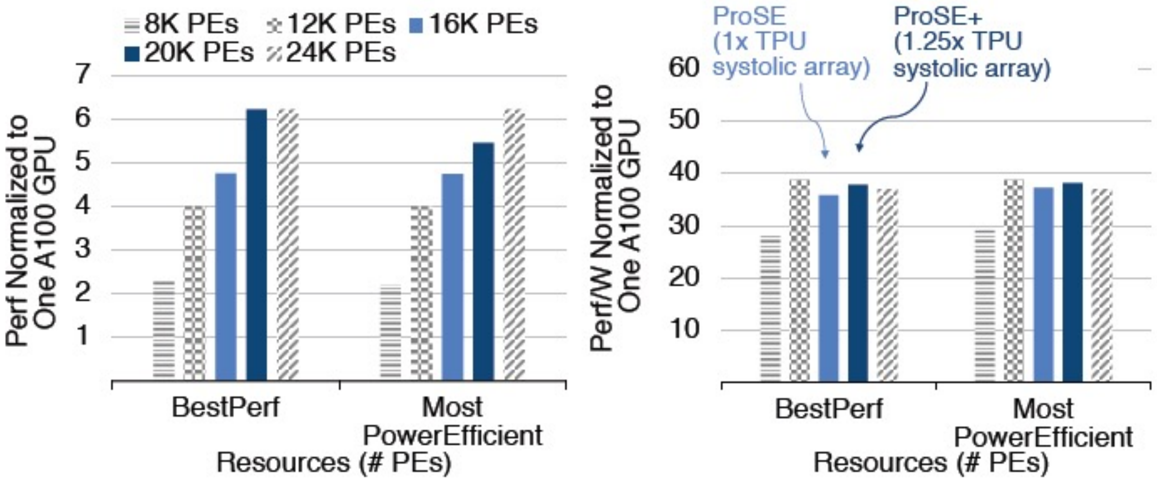
Additional material

Overview of the contributions of the article



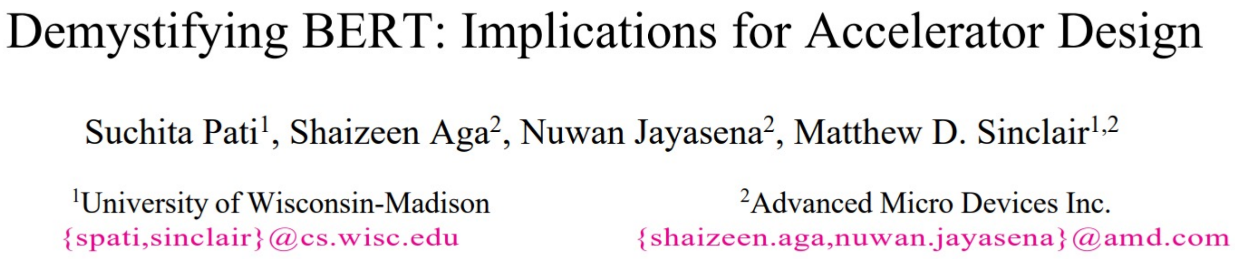
Protein BERT PyTorch Frontend: Python ★ C++ Java
PyTorch JIT Compiler
ATen Lib Calls (e.g., MatMul, MatAdd)
ProSE written in Chisel
ProSE generated in Verilog
Dataflow Construction
Functional Simulation via Verilog Simulation
Timing Simulation via a Cycle-Accurate Simulator written in Python
Our contribution
Open-source toolchain, compiler, etc.

Performance per number of PEs



Additional material

To what extent these “key take-aways” are taken into account?



Demystifying BERT: Implications for Accelerator Design

Suchita Pati[1], Shaizeen Aga[2], Nuwan Jayasena[2], Matthew D. Sinclair[1,2]

[1]University of Wisconsin-Madison
{spati,sinclair}@cs.wisc.edu

[2]Advanced Micro Devices Inc.
{shaizeen.aga,nuwan.jayasena}@amd.com

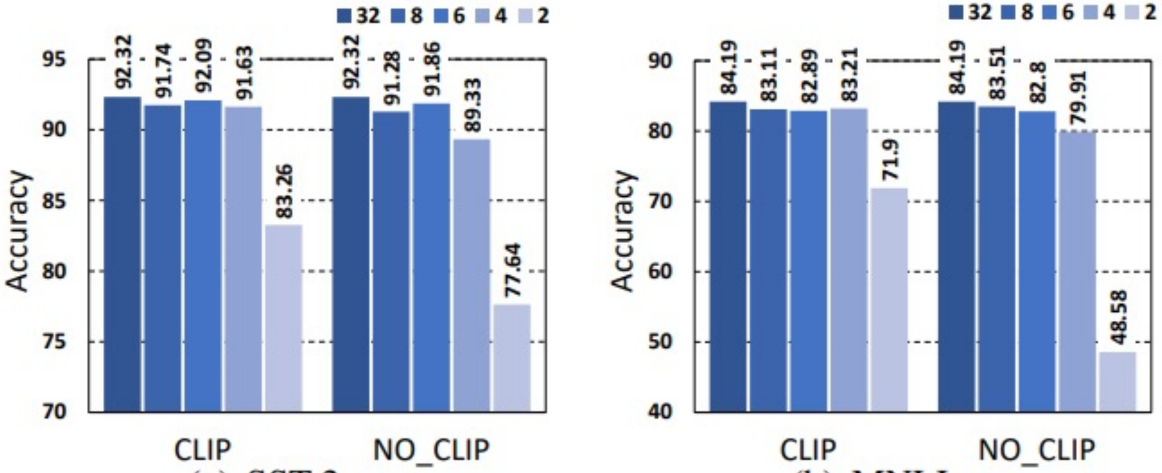
1. Of the different layers in BERT, the transformer layers dominate its training time, while the output & embedding layers have negligible contribution.
2. BERT’s gradient descent optimizer (LAMB), which updates the model weights, is the second highest contributor to BERT’s training runtime, and its contribution increases with decreasing input token count per iteration.
3. Both transformer and LAMB parameter update remain important as transformer layer count is increased.
4. Not all matrix multiplications in BERT are equal: only some of them can fully utilize highly parallel accelerators.
5. Parameter updates using LAMB are extremely memory intensive.
6. The runtime proportion of matrix multiplications and LAMB update increase in wider models (larger hidden dimensions).

Additional material

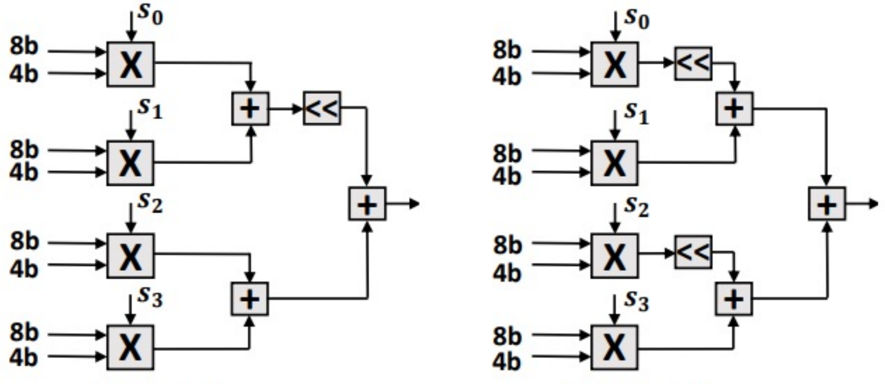
Zejian Liu and al.

- Precision on two different data sets per weight bitwidth
  CLIP = Adjusting of the MAX and MIN value by clamping.

- Different designs of the Bit-split Inner-Product Module (BIM)



(a) SST-2 (b) MNLI

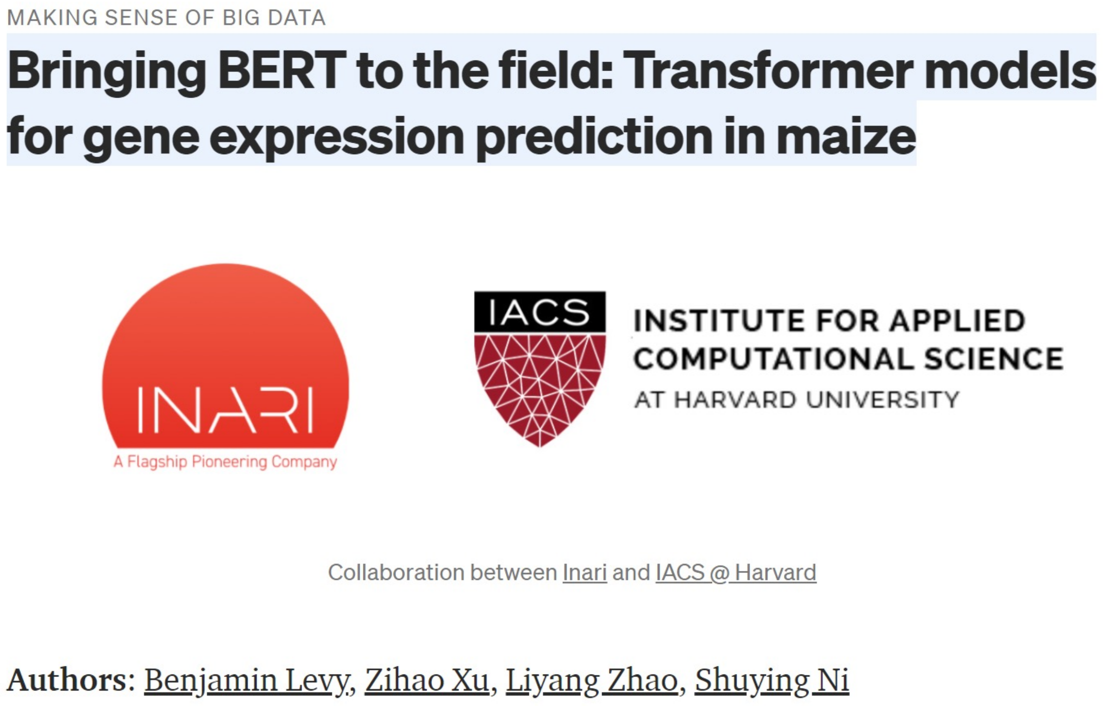


(a) Type A (b) Type B

Additional material

CornBERT

Project applying BERT for given a gene’s regulatory (promoter) sequence of maize DNA, can predict how much that gene will be expressed in ten different corn tissues.



MAKING SENSE OF BIG DATA

Bringing BERT to the field: Transformer models for gene expression prediction in maize

INARI
A Flagship Pioneering Company
IACS
INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE
AT HARVARD UNIVERSITY

Collaboration between Inari and IACS @ Harvard

Authors: Benjamin Levy, Zihao Xu, Liyang Zhao, Shuying Ni