



Pruna AI

Evaluation for Language Models



Bertrand Charpentier
Founder, President & Chief Scientist

Overview - Evaluation for Language Models



Quality evaluation

- Perplexity
- Task-specific metrics (e.g. accuracy for question-answering)
- Robustness *[Reliability]*
- Uncertainty *[Reliability]*

Efficiency evaluation

- #Parameters, #Activations *[Memory]*
- Disk, Inference, training memory *[Memory]*
- MAC, FLOP, FLOPS, OP, OPS *[Compute]*
- Scaling laws
- Latency *[Memory]* *[Compute]*
- Memory vs Compute bound
- Throughput *[Memory]* *[Compute]*
- Money *[Memory]* *[Compute]*
- Energy *[Memory]* *[Compute]*



Quality Evaluation



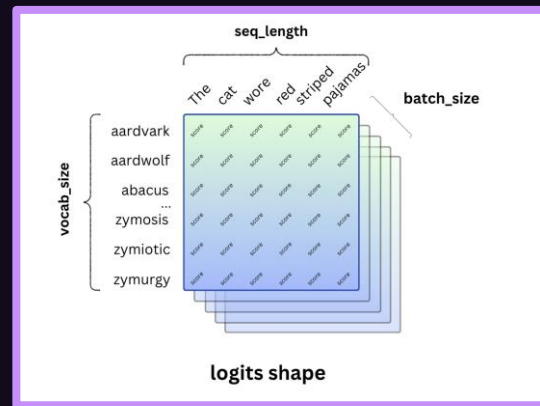
Perplexity

The perplexity measures how well a model can explain/predict data samples.

Definition: Given a dataset $D = \{x_1, \dots, x_N\}$, the perplexity of the (probabilistic) model $P_\theta(\cdot)$ is $e^{-\frac{1}{N} \sum_{i=1}^N \log(P_\theta(x_i))}$. The negative exponent indicates an average likelihood per token. Hence, the lower the perplexity is, the better the model is at explaining the dataset D .

Remarks

- It only needs a text corpus with any additional labels.
- It is an efficient and powerful proxy quality metric to evaluate the capacity of a model on a given data domain.
- Perplexity does not perfectly correlate with task specific metrics e.g. reasoning or long-term understanding.
- Using longer context length helps reducing perplexity.



The

[

]

Reference:

- <https://www.comet.com/site/blog/perplexity-for-llm-evaluation/>
- https://kilthub.cmu.edu/articles/journal_contribution/Evaluation_Metrics_For_Language_Models/6605324/1?file=12095765



Accuracy

The accuracy measures capacity of a model for question answering task.

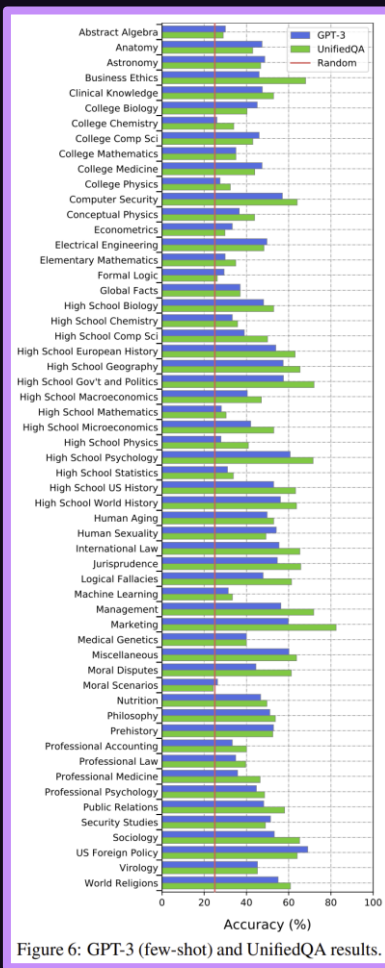
Definition: Given a dataset $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$, the accuracy corresponds to the number of correct answer over the total number of answers. Formally, for a (probabilistic) model $P_\theta(\cdot)$, it is $\frac{\sum_{i=1}^N \mathbb{1}_{\arg\max(P(x_i)=y_i)}}{N}$

Remarks

- It requires a labelled questions/answers dataset.
- It is easy to evaluate on multi choice questions.
- It is complicated to evaluate on open questions.

Reference:

• <https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation>

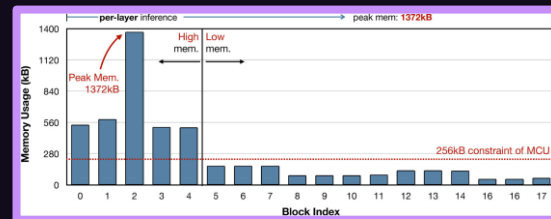


Efficiency Evaluation



Memory Metrics

Category	Definition	Formula
Number of parameters	Number of weights in the model.	$\#parameters$
Number of active activations	Number of activations <i>active</i> in the model.	$\#activations$
Storage memory	Memory required to store the model.	$\#parameters \times avg\ bit\ width$
Inference memory	Memory required to run inference on the model.	$\#parameters$ $\times avg\ parameter\ bit\ width$ $+ \#activations$ $\times avg\ activation\ bit\ width$
Training memory	Memory required to run training on the model.	$\#parameters$ $\times avg\ parameter\ bit\ width$ $+ \#activations$ $\times avg\ activation\ bit\ width$ $+ \#parameters$ $\times avg\ gradient\ bit\ width$



Layer	# Parameters
Linear Layer	$c_i \times c_o$
Convolution	$c_i \times c_o \times k_h \times k_w$
Grouped Convolution	$\frac{c_i \times c_o \times k_h \times k_w}{g}$
Depthwise Convolution	$c_o \times k_h \times k_w$

c_i : #input channels
 c_o : #output channels
 k_h : kernel height
 k_w : kernel width
 g : #groups

Remarks

- The storage, inference, training memory metrics measured in **Bytes or Bits** are the **closest to the real-world constraints**. The rest are proxy metrics.
- The real storage, inference, training memory requirements is **distributed across memory levels** (e.g. disk, cpu, gpu).
- The real storage, inference, training memory requirements depends on **implementation efficiency** (e.g. variables can be shared).
- The size of SOTA models overall increase with time. Scaling model size leads to better performance.

Reference:

- <https://epoch.ai/data/notable-ai-models>
- <https://arxiv.org/abs/2203.15556>
- <https://arxiv.org/pdf/2007.10319>



Compute Metrics

Category	Definition	Formula
MAC (Multiply-Accumulate)	A multiplication followed by an addition.	$\#(\text{multiplication} + \text{addition})$
FLOP (Floating Point Operation)	A single floating-point arithmetic operation (addition, multiplication, etc.).	$\#\text{multiplications} + \#\text{additions}$
FLOPS (Floating Point Operations Per Second)	The number of FLOPs that a system can process per second.	$\frac{\text{FLOPs}}{\text{Execution time}}$
OP (Operation)	A single general computation operation (e.g., MACs, FLOPs, comparisons, etc.).	$\#\text{multiplications} + \#\text{additions} + \#\text{comparisons} + \dots$
OPS (Operations Per Second)	The number of OPs that a system can process per second.	$\frac{\text{OPs}}{\text{Execution time}}$

Layer	MACs (batch size = 1)
Linear Layer	$c_i \times c_o$
Convolution	$c_i \times c_o \times k_h \times k_w \times h_o \times w_o$
Grouped Convolution	$\frac{c_i \times c_o \times k_h \times k_w \times h_o \times w_o}{g}$
Depthwise Convolution	$c_o \times k_h \times k_w \times h_o \times w_o$
Attention calculation	$s \times s \times c_i$

c_i, c_o : #input/ #output channels
 h_i, h_o : #input/ #output heights
 w_i, w_o : #input/ #output widths
 k_h, k_w : kernel height/width
 s : sequence length
 g : #groups

Remarks

- Multiplication and addition are key operations for matrix-vector (GEMV) and matrix-matrix multiplications (GEMM).
- Performance of models can be improved by investing more compute during training or inference.

Reference:

- <https://medium.com/@pashashaik/a-guide-to-hand-calculating-flops-and-macs-fa5221ce5ccc>
- <https://epoch.ai/data/notable-ai-models>



Scaling Laws

Training-time scaling laws: The test performance L of a model architecture scales with the model size N , training dataset size D , training cost C .

$$L = L_0 + \left(\frac{x_0}{x}\right)^\alpha$$

Henighan & Kaplan
scaling laws

$$\begin{cases} C = C_0 ND \\ L = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + L_0 \end{cases}$$

Chinchilla scaling laws

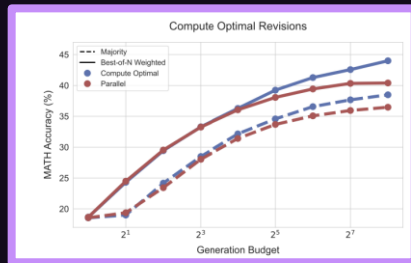
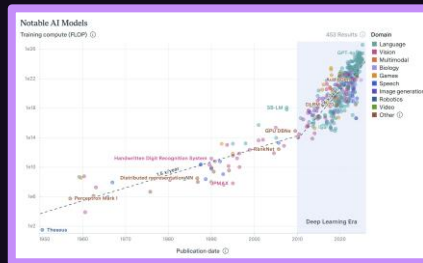
$$y = a + \left(bx^{-c_0}\right) \prod_{i=1}^n \left(1 + \left(\frac{x}{d_i}\right)^{1/f_i}\right)^{-c_i * f_i}$$

Broken scaling laws

Test-time scaling laws: The test performance L of a model scales with the test compute cost C .

Remarks

- It is possible to trade training compute against test compute.
- There are many other scaling laws accounting for more factors (e.g. precision, distillation, ...).



Reference:

- <https://arxiv.org/abs/2210.14891>
- <https://arxiv.org/abs/2010.14701>
- <https://arxiv.org/abs/2203.15556>



Latency

Definition: It is the time to obtain the prediction of one single output.



High latency – 5 sec/image



Low latency – 0.5 sec/image

Remarks

- It is important for the user experience.
- It can be directly improved:
 - with more efficient models.
 - with more performant hardware.
 - with more hardware if computation is well distributed.



Latency vs Memory/Compute

Latency is the consequence of time spend for memory access and computation time.

$$\text{Latency} \approx \max(T_{\text{memory}}, T_{\text{compute}})$$

$$T_{\text{compute}} = \frac{\text{\#Operations by the model}}{\text{\#Operation processed per time unit by the hardware}}$$

$$T_{\text{memory}} = T_{\text{activation}} + T_{\text{weights}}$$

$$T_{\text{activation}} = \frac{\text{Input activation size} + \text{output activation size}}{\text{Memory bandwidth of the hardware}}$$

$$T_{\text{weights}} = \frac{\text{Model size}}{\text{Memory bandwidth of the hardware}}$$

Remarks

- Latency depends on both the hardware and the model.
- Both memory access and compute can take time and energy!

Integer		FP		Memory	
Add		FAdd		Cache (64bit)	
8 bit	0.03pJ	16 bit	0.4pJ	8KB	10pJ
32 bit	0.1pJ	32 bit	0.9pJ	32KB	20pJ
Mult		FMult		1MB	100pJ
8 bit	0.2pJ	16 bit	1.1pJ	DRAM	1.3-2.6nJ
32 bit	3.1pJ	32 bit	3.7pJ		

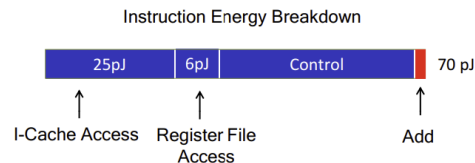


Figure 1.1.9: Rough energy costs for various operations in 45nm 0.9V.

Reference:

• <https://gwern.net/doc/cs/hardware/2014-horowitz-2.pdf>



Memory vs Compute Bound

Arithmetic intensity of the model

$$\frac{\text{\#Operations by the model}}{\text{input activation size} + \text{output activation size} + \text{model size}}$$

Ops/byte ratio of the hardware

$$\frac{\text{\#Operation processed per time unit by the hardware}}{\text{Memory bandwidth of the hardware}}$$

When is the memory the limiting factor?

$$T_{\text{memory}} > T_{\text{compute}} \Leftrightarrow \text{Arithmetic int.} < \frac{\text{ops}}{\text{byte}} \text{ ratio}$$

When is the compute the limiting factor?

$$T_{\text{memory}} < T_{\text{compute}} \Leftrightarrow \text{Arithmetic int.} > \frac{\text{ops}}{\text{byte}} \text{ ratio}$$

Figure 1. Simplified view of the GPU architecture

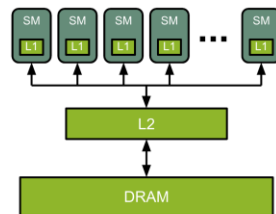


Table 1. Examples of neural network operations with their arithmetic intensities. Limiters assume FP16 data and an NVIDIA V100 GPU.

Operation	Arithmetic Intensity	Usually limited by...
Linear layer (4096 outputs, 1024 inputs, batch size 512)	315 FLOPS/B	arithmetic
Linear layer (4096 outputs, 1024 inputs, batch size 1)	1 FLOPS/B	memory
Max pooling with 3x3 window and unit stride	2.25 FLOPS/B	memory
ReLU activation	0.25 FLOPS/B	memory
Layer normalization	< 10 FLOPS/B	memory

Example: V100 Peak compute rate of 125 FP16 Tensor TFLOPS, an off-chip memory bandwidth of approx. 900 GB/s, and an on-chip L2 bandwidth of 3.1 TB/s, giving it a ops:byte ratio between 40 and 139, depending on the source of an operation's data (on-chip or off-chip memory).

Reference:

• <https://docs.nvidia.com/deeplearning/performance/dl-performance-gpu-background/index.html#understand-perf>

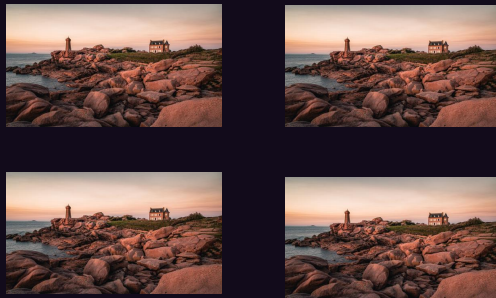


Throughput

Definition: It is the number of output that can be predicted per time unit.



Low throughput – 1 image/sec



High throughput – 4 image/sec

Remarks

- It is important to scale to a large set of queries/users.
- It can be directly improved
 - with more efficient models.
 - with more hardware.
 - with better hardware for a whole server.



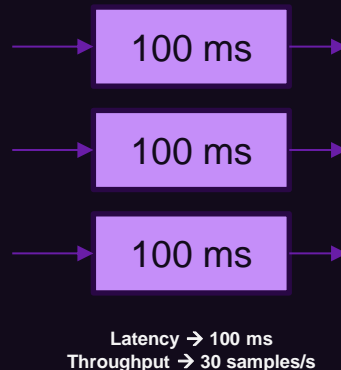
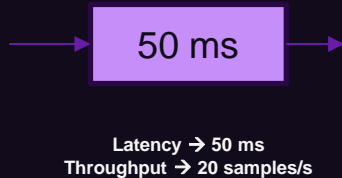
Latency vs Throughput

Latency and throughput are equivalent for sequential input processing.

Example: A smartphone process frame after frame.

Latency and throughput are not equivalent for batched input processing.

Example: A server process a batch of user queries.



Money

Definition: It is the money required for the prediction of a single output.

If you use cloud machines, it approximately corresponds to

$$\frac{\text{hardware price per time unit} \times \text{usage time}}{\text{number of processed samples}}$$

If you use internal machines, it approximately corresponds to

$$\frac{\text{hardware energy consumption per time unit} \times \text{energy price per time unit} \times \text{usage time}}{\text{\#processed samples}} + \frac{\text{hardware price}}{\text{\#processed samples in hardware lifetime}}$$

Remarks

- It can be improved via more efficient models.
- More performant hardware, or more hardware are not guaranteed to improve it. It depends on the price to hardware compute power ratio.



Energy

Definition: It is the energy required for the prediction of a single output.

It approximately corresponds to

$$\frac{\text{hardware energy consumption per time unit} \times \text{usage time}}{\text{\#processed samples}} + \frac{\text{energy for hardware production}}{\text{\#processed samples in hardware lifetime}}$$

Remarks

- It can be improved via more efficient models.
- More performant hardware, or more hardware are not guaranteed to improve it. It depends on the price to hardware compute power ratio.

