Lecture 13 - Large Language Models

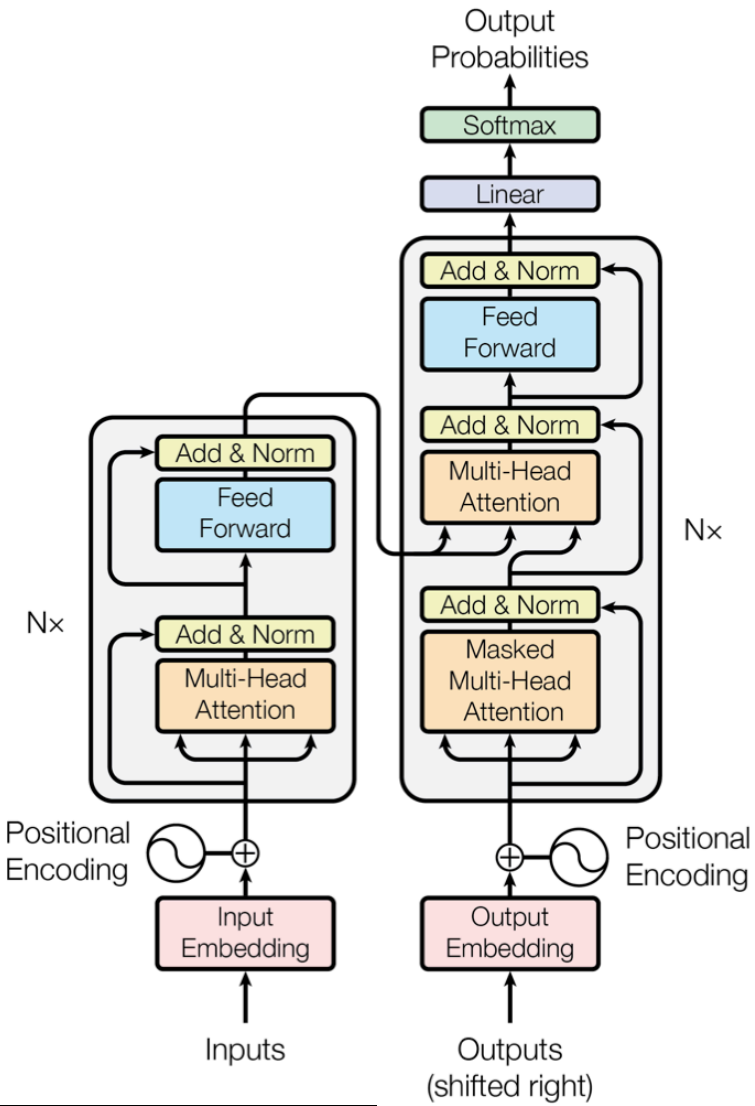
Advanced Machine Learning

Alexander Kovalenko, **Miroslav Čepek**, Zdeněk Buk, Rodrigo Da Silva Alves, Vojtěch Rybář, Petr Šimánek

FIT CTU

15. 5. 2025

Transformer (Reminder)



Attention is All You Need

GPT

- Leverages decoder part of Transformer architecture. Specifically multi-layer, multi-head self-attention transformer.
- The idea of the architecture is:
  - $h_0 = U\mathbb{W}_e + \mathbb{W}_p$
  - $h_l = \text{transformer\_block}(h_{l-1}) \forall i \in [1, n]$
  - $P(u) = \text{softmax}(h_n \mathbb{W}_e^T)$
- The training consists of two stages:
  1. Unsupervised pre-training
  2. Supervised fine-tuning

Improving Language Understanding by Generative Pre-Training

GPT Training

- Unsupervised pre-training aims to predict correctly the next token.
- Uses BookCorpus Dataset (7000 books) with long stretches of continuous text to capture long-range structures.
  - $L_1(\mathcal{U}) = \sum_i \log P\left(u_i | u_{i-1} \dots u_{i-k}; \Theta\right)$
- Supervised fine-tuning aims to correctly fulfil classification task.
  - $\mathcal{L}_2(\mathcal{C}) = \sum \log P\left(y | x_1 \dots x_m\right) \approx \text{softmax}(h_l \mathbb{W} y)$
- To speed up convergence and improve generalisation, authors found that it's good to to combine both loss functions in fine-tuning phase.
  - $\mathcal{L} = \mathcal{L}_2(\mathcal{C}) + \lambda \mathcal{L}_1(\mathcal{C})$

GPT Finetuning Tasks

- Natural Language Inference (Text entailment) – reading pair of sentences and deciding if they have something in common (entailment, contradiction, neutral).
- Question answering and common sense reasoning – text understanding and answer a question put to the model. The questions may be long requiring a model to work with large context.
- Semantic Similarity (paraphrase detection) – identify if two sentences are semantically equivalent or not.
- Classification – deciding into which predefined class a text belongs.

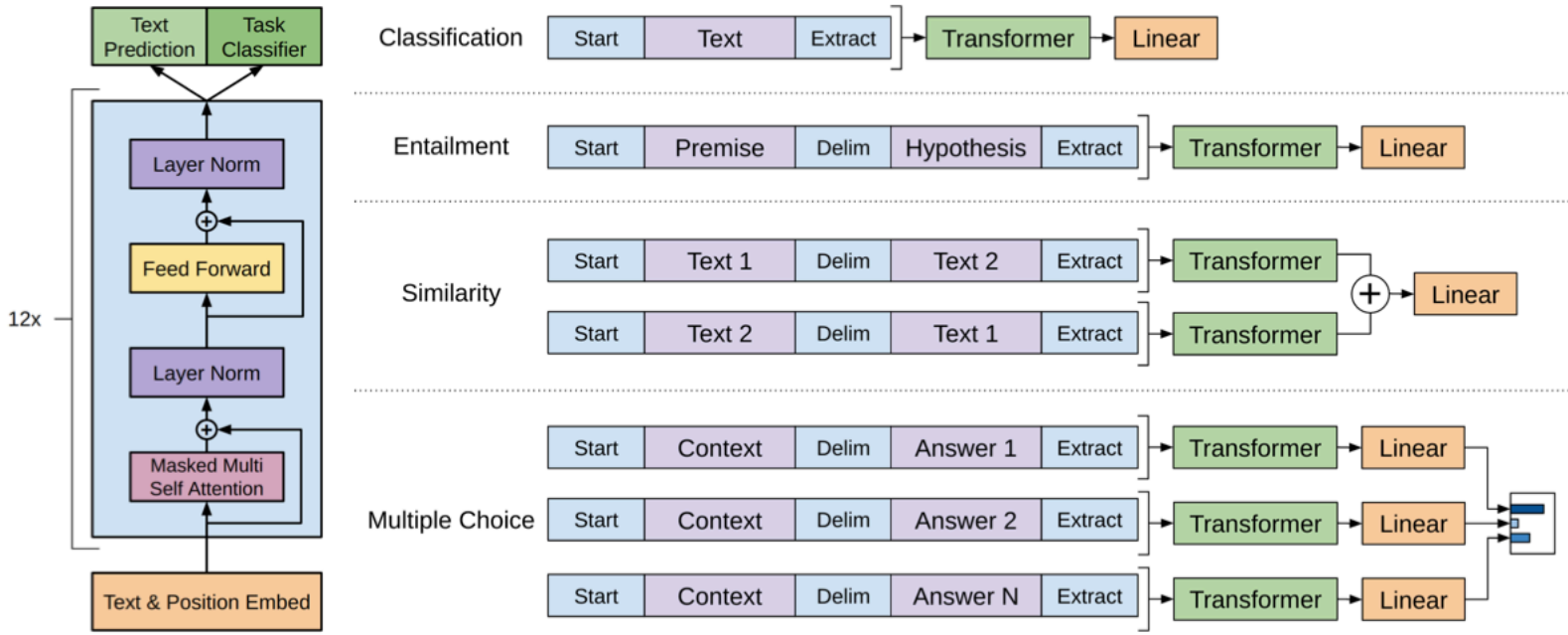


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Improving Language Understanding by Generative Pre-Training

GPT-3, 4

- GPT-3 is same general architecture as GPT 2, just bigger (175B parameters, 96 attention heads, 2048 tokens in context window).
- GPT-4 is again transformer-based model pre-trained to predict the next token.
- GPT-4 is very sketchily described as "a large multimodal model capable of processing image and text inputs and producing text outputs." and "Given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar."

Language Models are Few-Shot Learners

GPT-4 Technical Report

LLAMA

- Less is sometimes better.
- LLaMa models has less weights than GPT-3 but better performance due to longer training on larger corpora of text.
  - LLaMa models has between 7B and 65B parameters. In contrast GPT-3 has 175B parameters.
- Again build on top of transformer architecture with some notable modifications:
- Pre-normalisation – normalise input of each transformer sublayer instead of normalising output.[1]
- SwiGLU – different activation function.[2]
- Rotary Embeddings – switch from positional embeddings.[3]

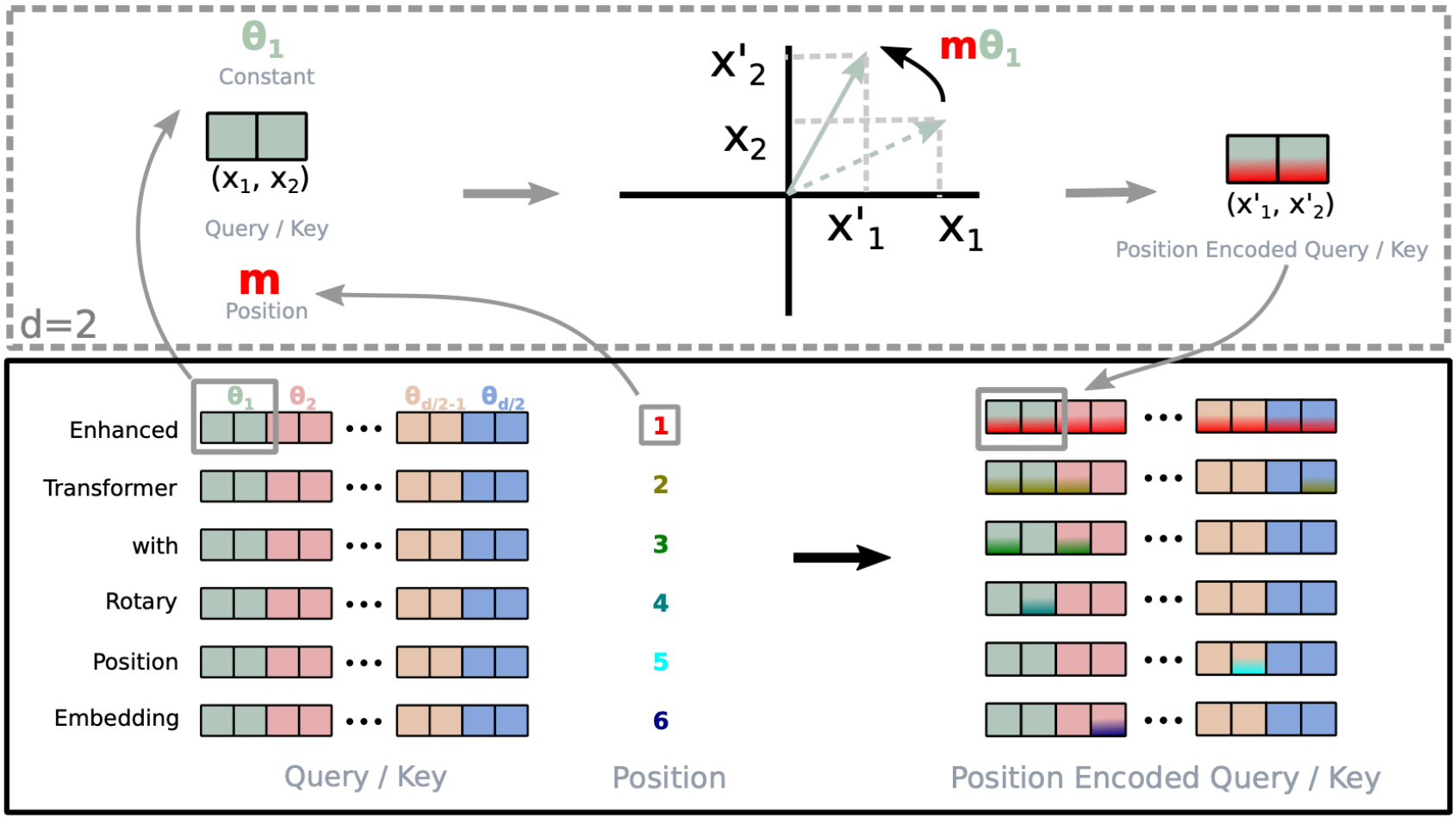
[1] Root Mean Square Layer Normalization

[2] GLU (Gated Linear Units) Variants Improve Transformer

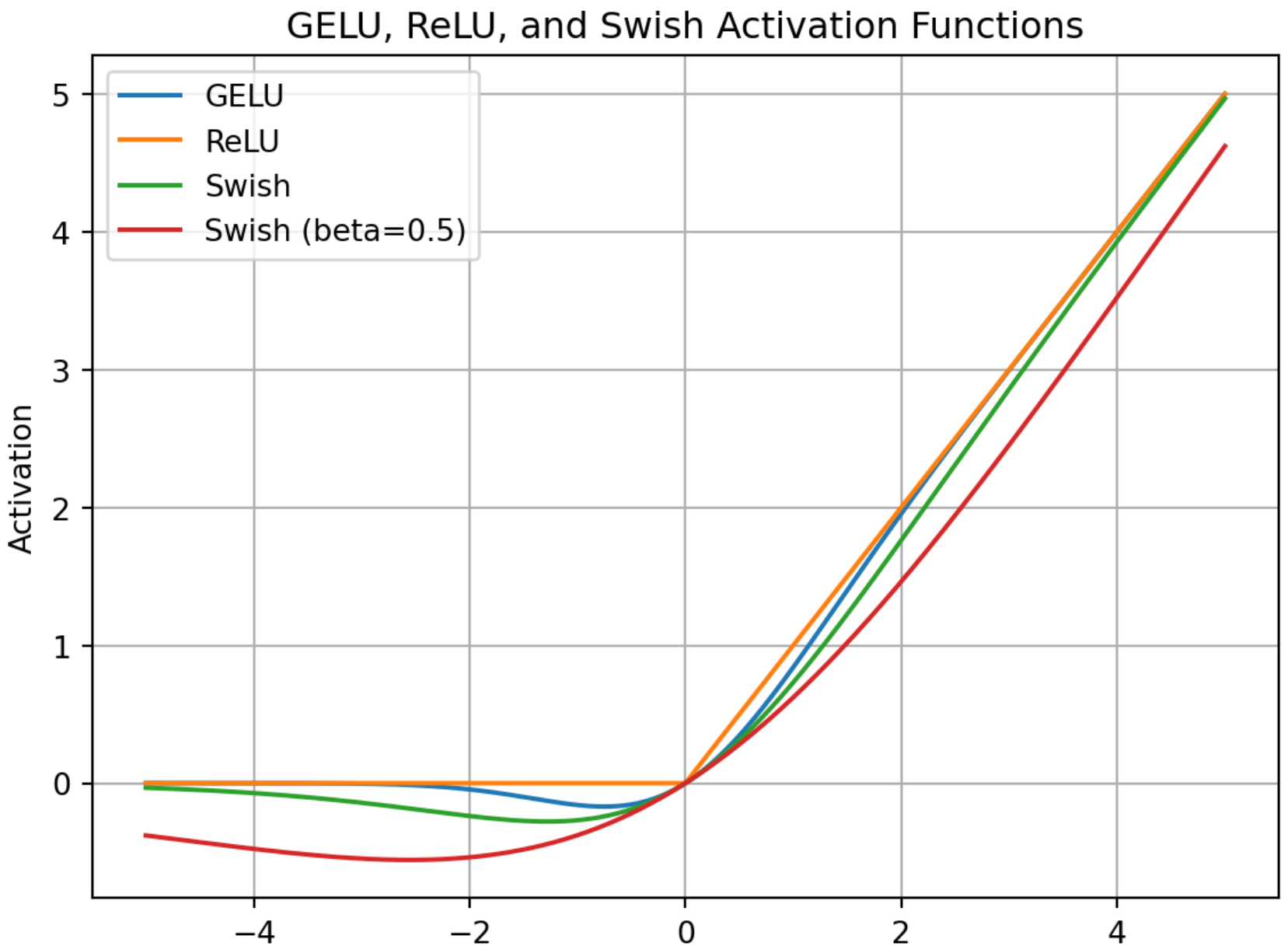
[3] Roformer: Enhanced transformer with rotary position embedding.

LLaMA: Open and Efficient Foundation Language Models

Rotary Embeddings



Swish Activation Function



LLaMa training datasets

- English Common Crawl – processed dumps of web pages from Common Crawl dataset between years 2017 and 2020. Deduplication, English only, "high quality" content.
- C4 - Same dataset, different filtering and processing.
- Github – source code of Apache licensed projects.
- Wikipedia – wiki pages in latin or cyrillic based languages (including cs).
- Gutenberg and Book3 – public domain books
- ArXiv – scientific papers
- Stack Exchange – question and answers on wide area of topics

In total 1.4T tokens after tokenisation.

LLaMa3

- Latest iteration of the model – released April 18.
- More training data - 15T tokens ($10\times$ of original model). Includes 5% of non-english data in 30 languages.
- Tokenizer has vocabulary of 128K tokens.
- Another attention mechanism - Grouped Query Attention[4].
- Includes safeguard models on the input and output to ensure "safe" prompts and responses.

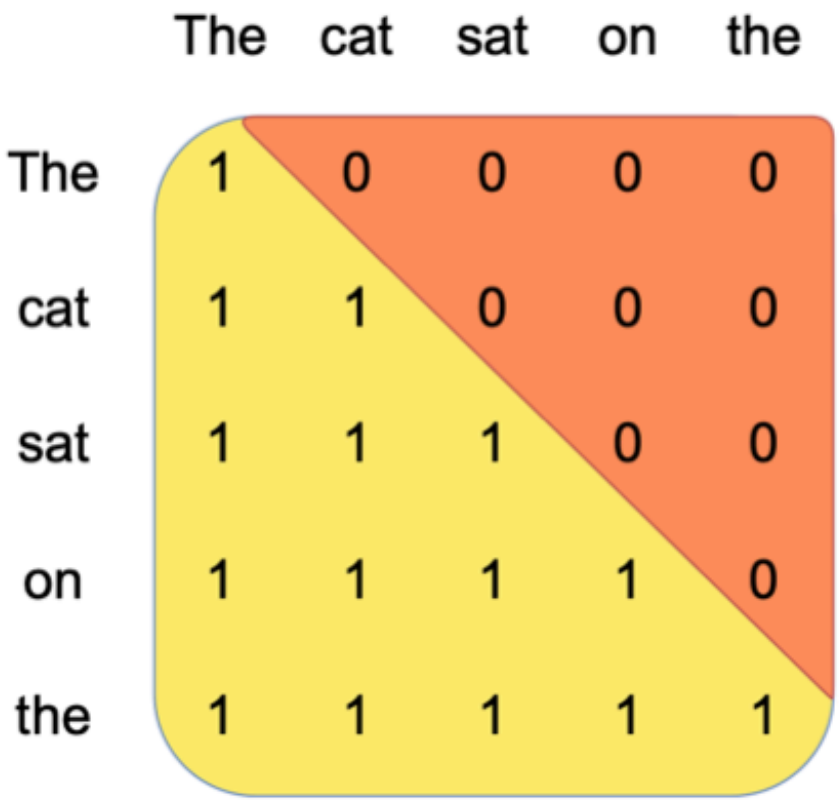
[4] GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints

Introducing Meta Llama 3: The most capable openly available LLM to date

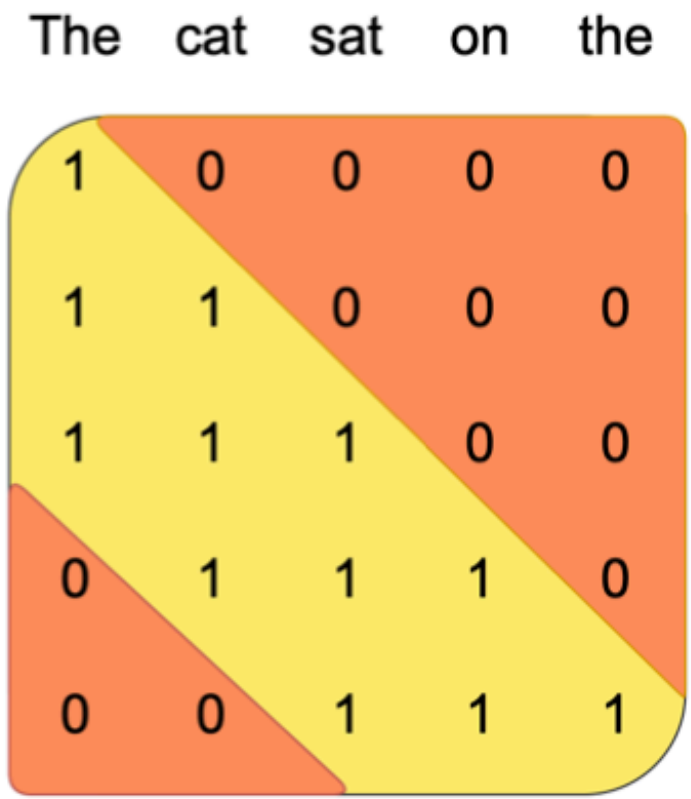
Mistral

- The basic architecture is the same - transformer.
  - Number of attention head 32
  - Context length 8192 tokens.
  - Size of hidden layer 14336.
- Important distinctions are:
  - Sliding Window Attention – Attention does not cover the whole input. The hidden state in position i of the layer k, h_i, has access to to all hidden states from the previous layer with positions between iW and i. Recursively, h_i can access tokens from the input layer at a distance of up to $W \times k$ tokens.
  - Rolling Buffer Cache – With fixed attention span authors limited the cache size to W. Using a rolling buffer cache.
  - Prefill and Chunking – Generating a sequence token-by-token and next token is conditioned on previous tokens. The prompt is known in advance, and we can pre-fill the (k, v) cache with the prompt.

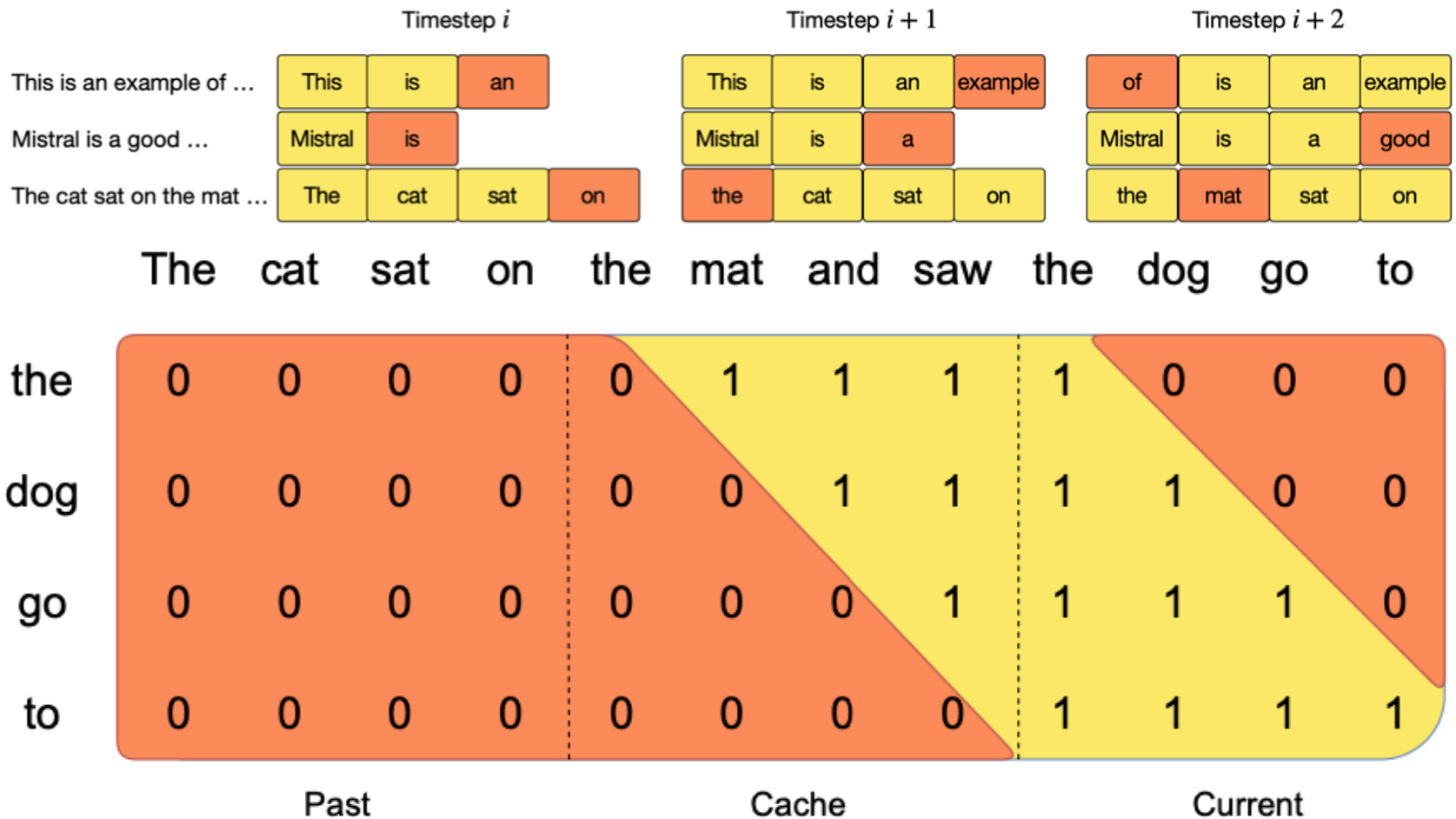
Mistral 7B



Vanilla Attention

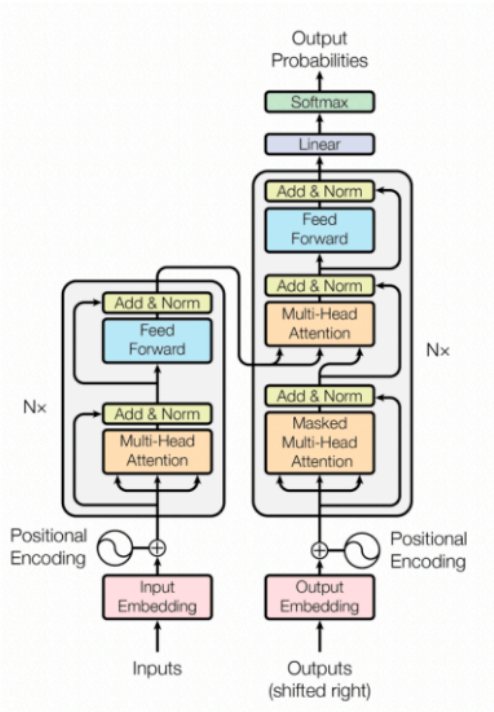


Sliding Window Attention

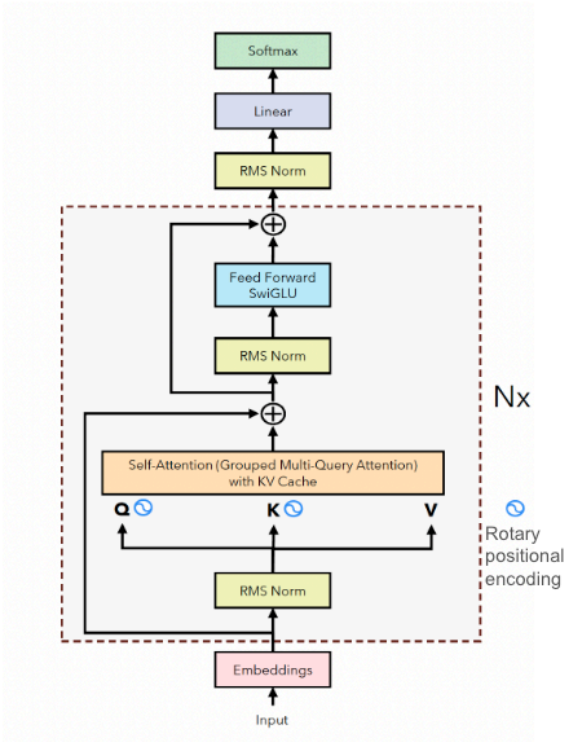

Timestep i
Timestep $i+1$
Timestep $i+2$
This is an example of ...
Mistral is a good ...
The cat sat on the mat ...
This is an
Mistral is
The cat sat on
This is an example
Mistral is a
the cat sat on
of is an example
Mistral is a good
the mat sat on
The cat sat on the mat and saw the dog go to
the 0 0 0 0 0 1 1 1 1 0 0 0
dog 0 0 0 0 0 0 1 1 1 1 0 0
go 0 0 0 0 0 0 0 1 1 1 1 0
to 0 0 0 0 0 0 0 0 1 1 1 1
Past
Cache
Current

Transformer vs LLaMa vs Mistral

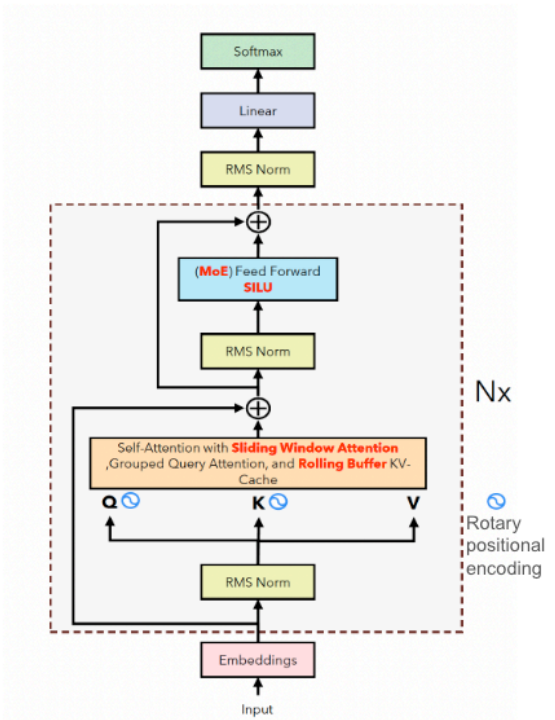
Transformer



LLaMA



Mistral



DeepSeek V3 Architecture

Builds on top of transformer architecture. Adopts:

- Multi-head Latent Attention – iteration on multi-head attention, uses low-rank matrices in Key-Value stages (less memory required, bigger cache).
- Mixture of Experts
- Multi-Token Prediction (training objective)

TransMLA: Multi-Head Latent Attention Is All You Need

DeepSeek-V3 Technical Report

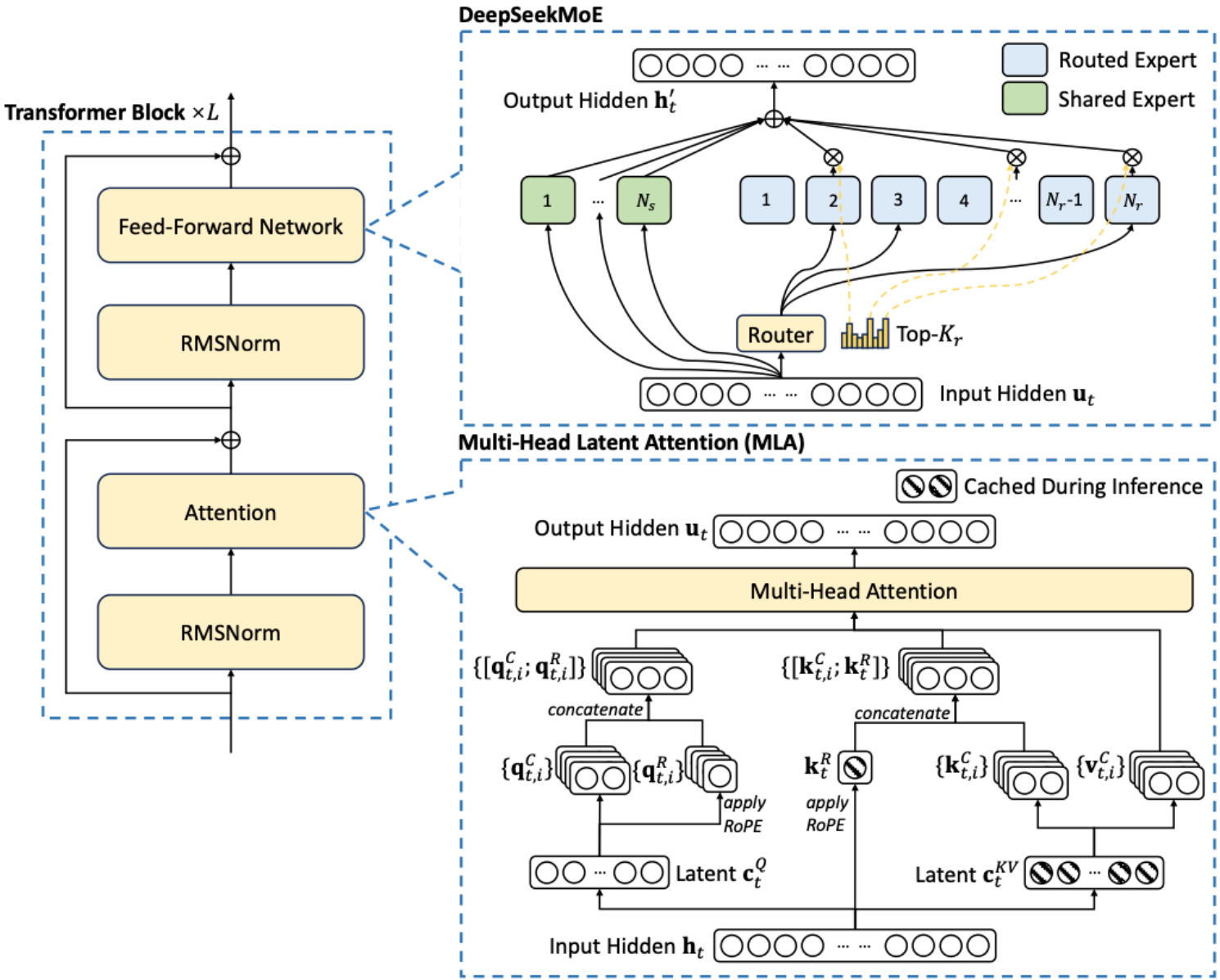


Figure 2 | Illustration of the basic architecture of DeepSeek-V3. Following DeepSeek-V2, we adopt MLA and DeepSeekMoE for efficient inference and economical training.

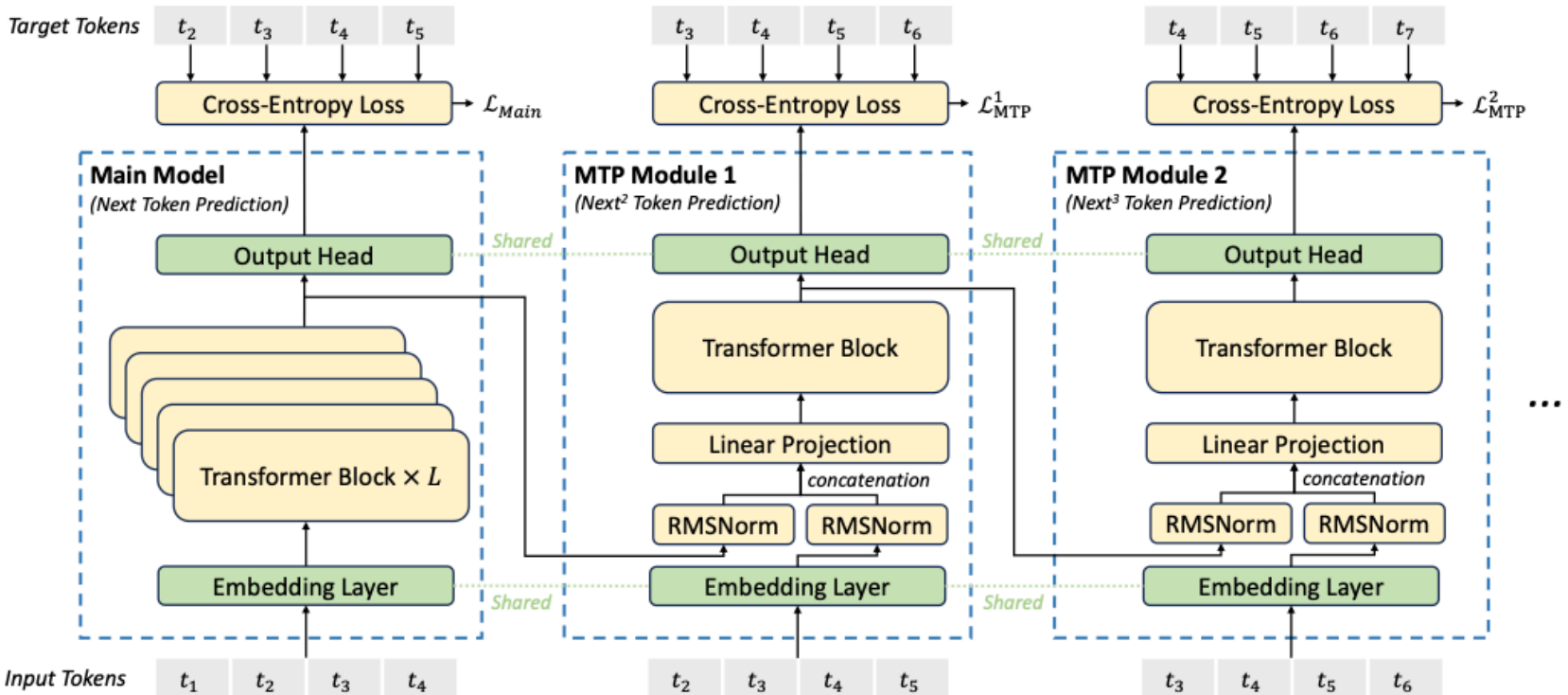


Figure 3 | Illustration of our Multi-Token Prediction (MTP) implementation. We keep the complete causal chain for the prediction of each token at each depth.

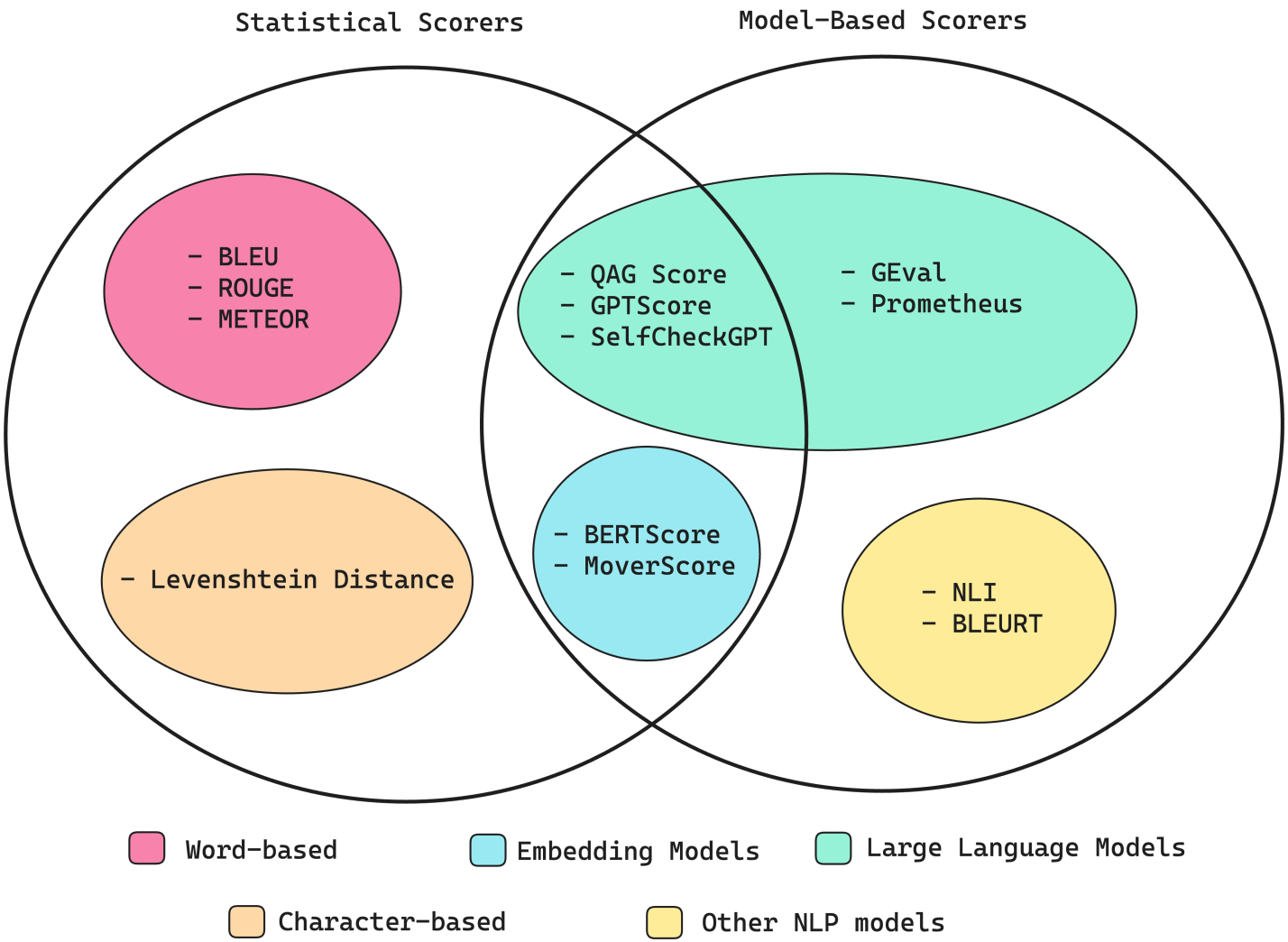
Goals

What are we expecting from evaluation metrics are:

- **Quantitative** – metrics should compute a score. Easier to use in:
  - decision making – Which model to use? Is a model better than it's predecessor?
  - monitoring – Is the performance degrading?
- **Reliable** – looking for consistent and predictable metric.
- **Accurate** – align metric with human perception of generated text as much as possible.

Confident AI Blog

LLM Quality Metrics



Confident AI Blog

LLM Quality Metrics

- Accuracy, F1-Score – how often a model gave correct answer.
  - What is the correct answer? Good for topic modelling, NER, YES/NO question answering ... and similar tasks.
- Perplexity – uncertainty in selecting the next token(s). Alternatively, how unlikely is the observed sequence.
  - $PPL(X) = \exp\left\{\frac{-1}{t}\sum_i^t \log p_\theta(x_i|x_{1\ldots i-1})\right\}$
  - Good to measure how well the model has learned the training data and to monitor the changes of fine-tuning or unlearning.
- BLEU *(Bilingual Evaluation Understudy)* – score measures the quality of the predicted text (aka candidate) and a set of references (expected results).
  - Great for sequence-to-sequence tasks, like machine translation, where there is known correct output (can be one or multiple).
  - Score is on scale between 0 and 1 (better).
  - The comparison between candidate and references is done on string basis - number of same n-grams occurring in both.

Bleu: a Method for Automatic Evaluation of Machine Translation

LLM Quality Metrics (2)

- ROUGE *(Recall-Oriented Understudy for Gisting Evaluation)* – is used for text summarisation.
  - Measures similarity of the generated text to one or more candidate text.
  - ROUGE is basically F1 score for number of matching ngrams over reference n-grams (*recall*) and candidate n-grams (*precision*).
  - ROUGE-L is a modification when instead of n-grams, you are looking for the longest common substring.
- METEOR – evaluates machine translation on word-basis. Looks in candidate and references for whole words and calculate the similarity on occurernce of whole words.

ROUGE: A Package for Automatic Evaluation of Summaries

METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments

BERTScore: Evaluating Text Generation with BERT

G-Eval – LLM based evaluation (of LLM) a.k.a. LLM as a Judge

- Use LLM to come up with evaluation of another natural language generation task.
- Idea is to create:
  1. a prompt that contains the definition of the evaluation task and the desired evaluation criteria,
  2. a chain-of-thoughts (CoT) that is a set of intermediate instructions generated by the LLM describing the detailed evaluation steps
  3. a scoring function that calls LLM and calculates the score based on the probabilities of the *grading* tokens.

G-EVAL: NLG Evaluation using GPT-4 with Better Human Alignment

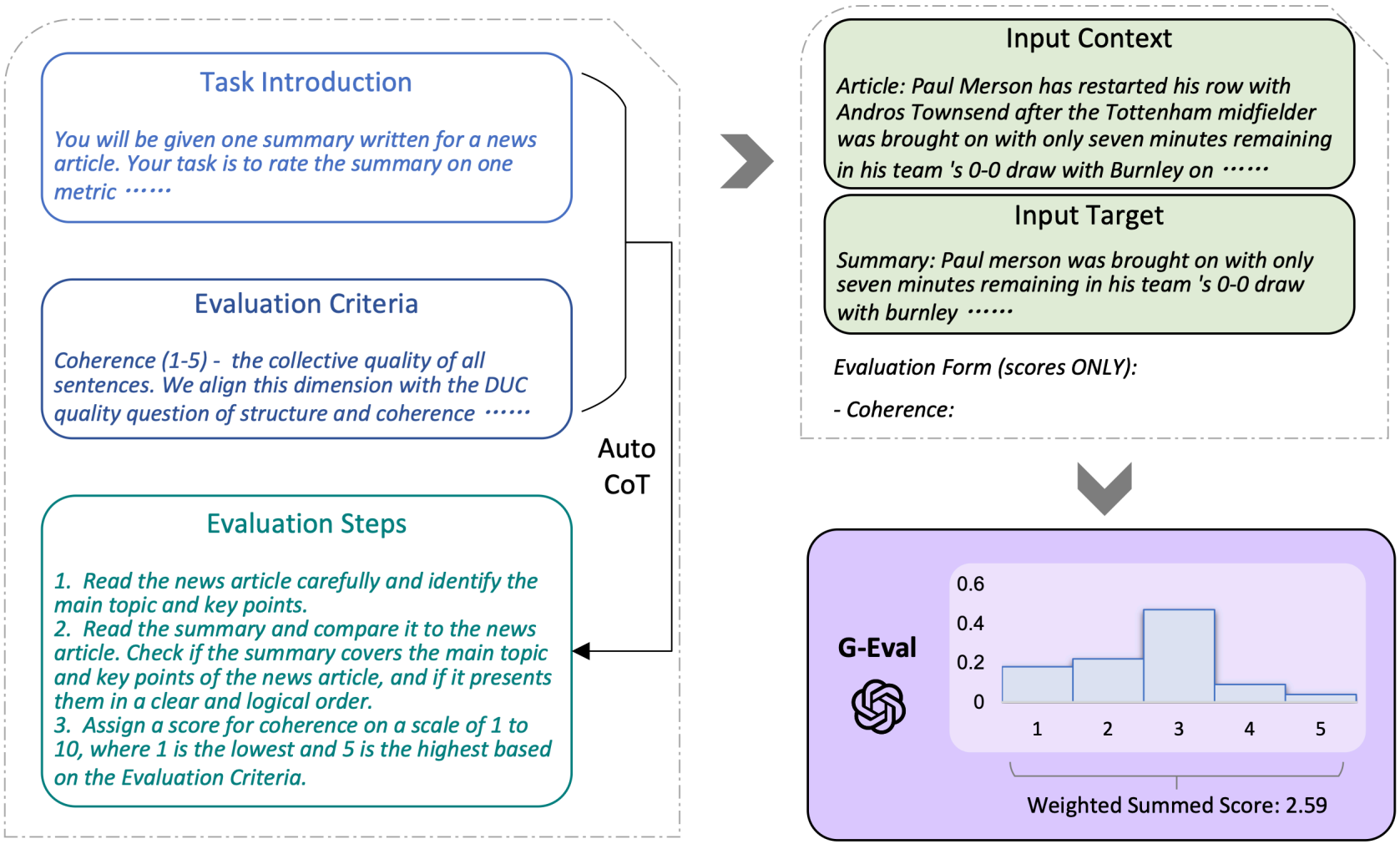
Grading Prompt

Following is an example prompt assessing summary of a text:

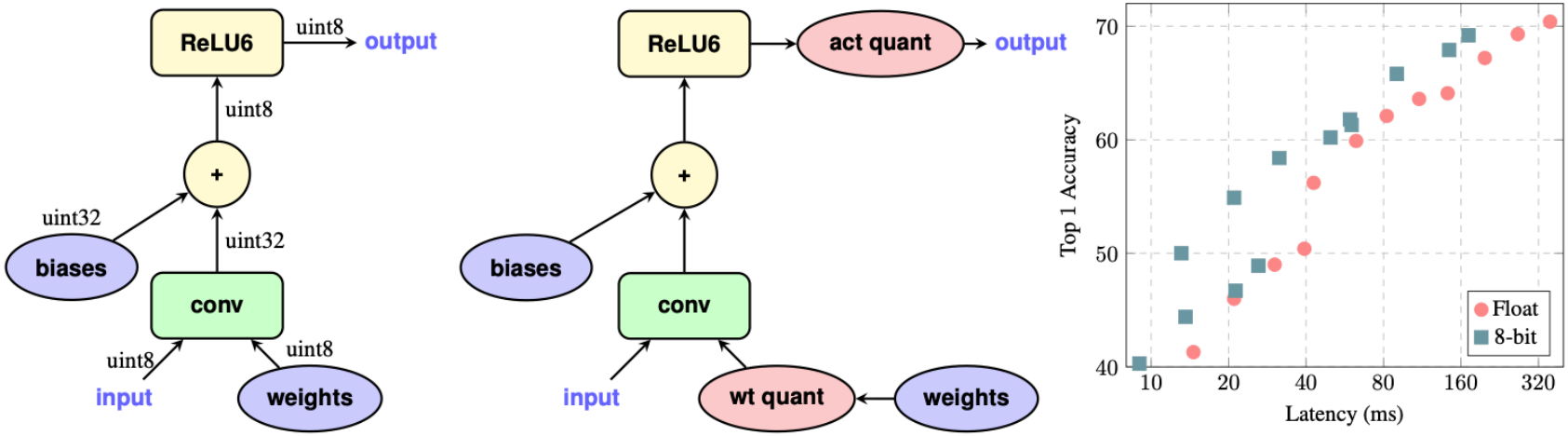
1. *Read the news article carefully and identify the main topic and key points.*
2. *Read the summary and compare it to the news article. Check if the summary covers the main topic and key points of the article, and if it presents them in a clear and logical order.*
3. *Assign a score for coherence on a scale of 1 to 5, where 1 is the lowest and 5 is the highest.*

To evalueate coherence in text summarization, authors concatenate the prompt, the Chain of thoughts, the news article, and the summary, and then call the LLM to output a score from 1 to 5 for each evaluation aspect, based on the defined criteria.

- Final score is probability weighted average: $\sum_i p(s_i) \times s_i$


Task Introduction
You will be given one summary written for a news article. Your task is to rate the summary on one metric ……
Evaluation Criteria
Coherence (1-5) - the collective quality of all sentences. We align this dimension with the DUC quality question of structure and coherence ……
Auto CoT
Evaluation Steps
1. Read the news article carefully and identify the main topic and key points.
2. Read the summary and compare it to the news article. Check if the summary covers the main topic and key points of the news article, and if it presents them in a clear and logical order.
3. Assign a score for coherence on a scale of 1 to 10, where 1 is the lowest and 5 is the highest based on the Evaluation Criteria.
Input Context
Article: Paul Merson has restarted his row with Andros Townsend after the Tottenham midfielder was brought on with only seven minutes remaining in his team 's 0-0 draw with Burnley on ……
Input Target
Summary: Paul merson was brought on with only seven minutes remaining in his team 's 0-0 draw with burnley ……
Evaluation Form (scores ONLY):
- Coherence:
G-Eval
0.6
0.4
0.2
0
1
2
3
4
5
Weighted Summed Score: 2.59

- Single Precision (32bit) floating point represents 2^{32} different values in range $\pm 3 \times 10^{-38} \ldots \pm 3 \times 10^{38}$.
- That's probably too granular and wasteful.
- Can we represent some values in lower precision?
  - Less data to transfers and store (compression)
  - Faster math – better data parallelisation, integer arithmetics
- $w_{\text{quant}} = \sigma(w - \mu)$ which returns distribution with zero mean and unit variance (Z-normalisation).



(a) Integer-arithmetic-only inference (b) Training with simulated quantization (c) ImageNet latency-vs-accuracy tradeoff

Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference (2017)

LLM - Quantisation Aware Training

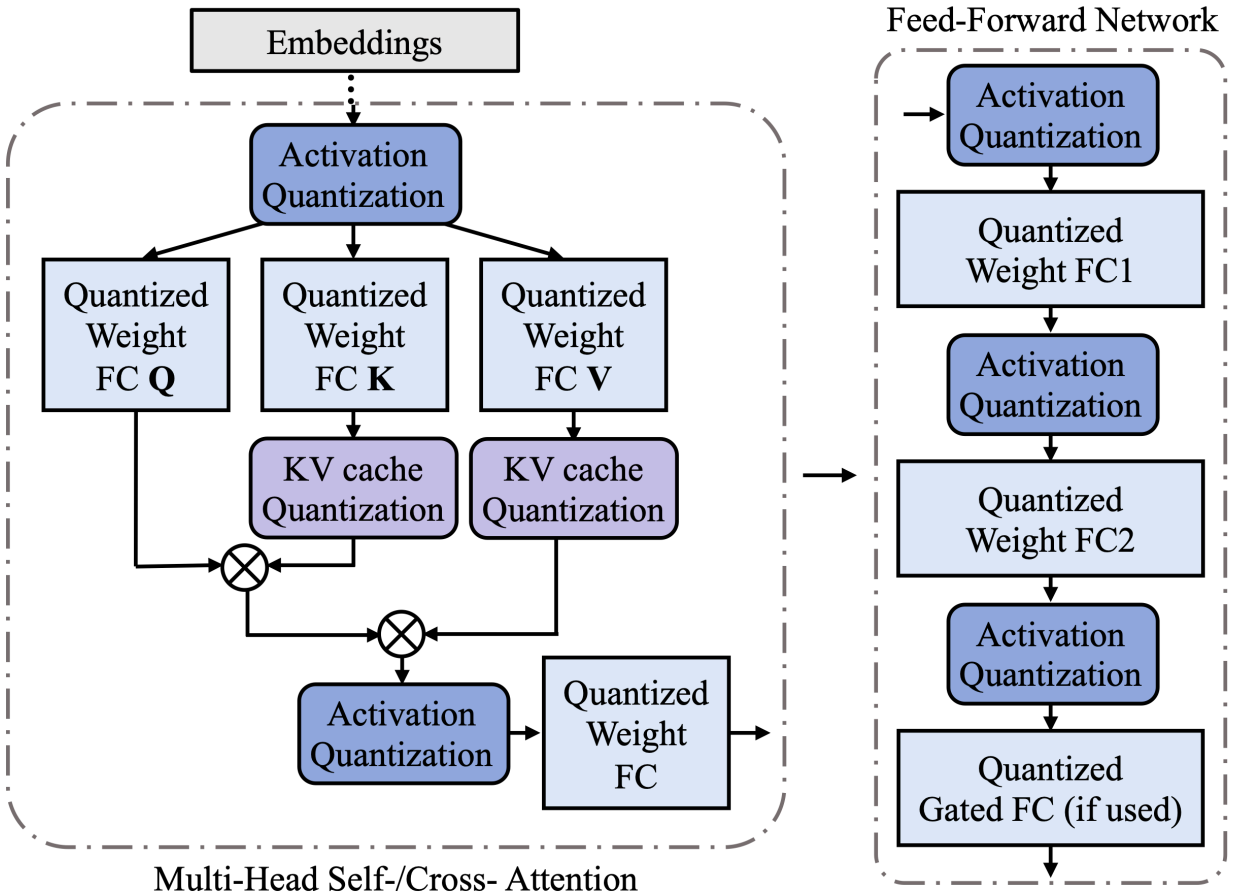
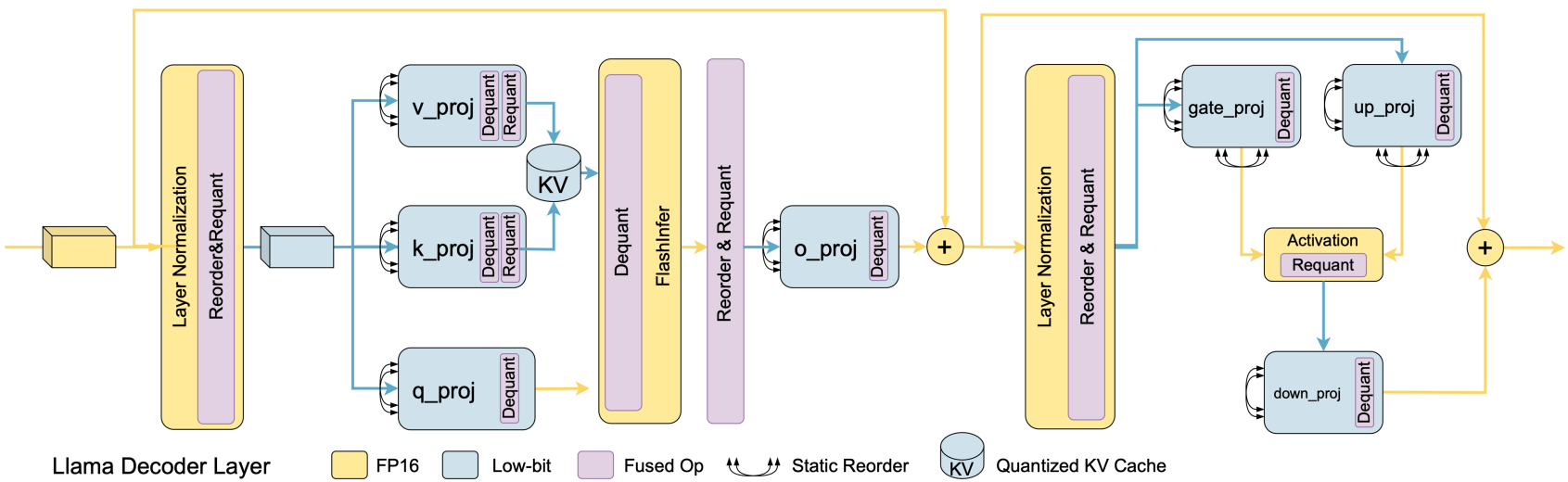


Figure 2: Overview of the quantized transformer in LLM-QAT. We quantize all the weights and input activations in fully-connected linear layers. The KV cache is also quantized if specified.

LLM-QAT: Data-Free Quantization Aware Training for Large Language Models

Mixed Precision Quantisation - ATOM

- In low-bit quantisation, it's OK for "standard" values. Outliers (typically interesting values) are represented with high error which may lead to low model performance.
- Addresses problems with outliers representing standard values with 4-bit and outliers with 8 bit integers.
- Reordering of values and weights is performed to keep low/high bit representation values together.



Atom: Low-Bit Quantisation for Efficient and Accurate LLM

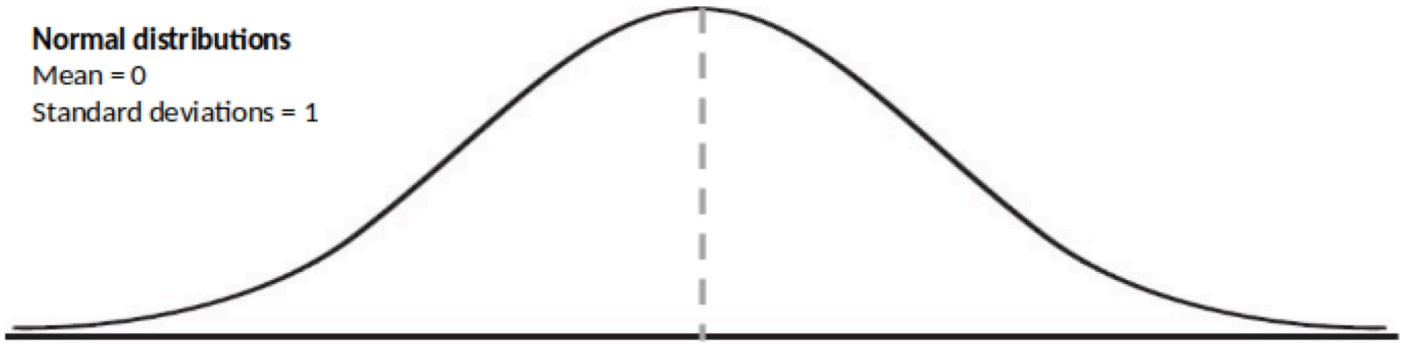
QLoRA: 4bit Normal Float Data Type

- Using 16 different "bins" to encode real values.
- To set upper and lower limit - use $[-1; +1]$ range. With explicitly set bin for zero point.
- Bins are defined by quantiles of a distribution of weights.
- Weights of LLMs are found to be zero-centered with arbitrary variance $\mathcal{N}(0; \sigma)$.
  - $X^{\text{4-bit INT}} = round\left(\frac{16}{\max \text{abs} X^{\text{FP32}}} X^{\text{FP32}}\right) = round(\mathcal{C}^{\text{FP32}} X^{\text{FP32}})$
- The quantisation is reversible as long as we remember the scaling constant $\mathcal{C}^{\text{FP32}}$.
  - $X^{\text{FP32}} = \text{dequant}(\mathcal{C}^{\text{FP32}}, X^{\text{4-bit INT}}) = \frac{X^{\text{4-bit INT}}}{\mathcal{C}^{\text{FP32}}}$
- To mitigate problem with outliers, the scaling constant $\mathcal{C}^{\text{FP32}}$ is calculated for a blocks of 64 weights independently.

QLoRA: Key Quantization and Fine-tuning Techniques in the Era of LLM (blog))

QLORA: Efficient Finetuning of Quantized LLMs

8-bit Optimisers via Block-Wise Qiantisation



[0.97, 0.9 , 0.83, 0.77, 0.7 , 0.63, 0.57] [0.56, 0.62, 0.68, 0.73, 0.79, 0.85, 0.91, 0.97] (Probability)

[-1.85, -1.29, -0.97, -0.73, -0.53, -0.34, -0.17] [0.15, 0.3, 0.45, 0.62, 0.81, 1.04, 1.34, 1.85] (Z-score)

[-1.85, -1.29, -0.97, -0.73, -0.53, -0.34, -0.17, 0 , 0.15, 0.3, 0.45, 0.62, 0.81, 1.04, 1.34, 1.85] (Concatenation)

[-1. , -0.7 , -0.53, -0.39, -0.28, -0.18, -0.09, 0. , 0.08, 0.16, 0.25, 0.34, 0.44, 0.56, 0.72, 1.] (Normalisation)

Steps for generating the NF4 data type values:

1. Generate 8 evenly spaced values from 0.56 to 0.97 (Set I).
2. Generate 7 evenly spaced values from 0.57 to 0.97 (Set II).
3. Calculate the z-score values for the probabilities generated in Step 1 and Step 2. For Set II, calculate the negative inverse of the z-scores.
4. Concatenate Set I, a zero value, and Set II together.
5. Normalize the values by dividing them by the absolute maximum value.

QLORA: Efficient Finetuning of Quantized LLMs

QLoRA Double Quantisation

- The 4-bit NormalFloat with 64 blocks quantisation can still be skimmed down.
- The need to store scaling constant $\mathcal{C}^{\text{FP32}}$ is wasteful.
- The precision of scaling constants is reduced to 8-bit floating point (need to remember another scaling constant).

Finetuning vs Few shot Learning

- Fine-Tuning (FT) – updates the weights of a pre-trained model by training on a supervised dataset specific to the desired task.
  - Strong performance on benchmarks.
  - Large dataset.
  - potential for poor generalization out-of-distribution
  - exploit spurious features of the training data.
- Few-Shot (FS) – refer to the setting where the model is given a few demonstrations of the task at inference time as conditioning (prompt).
  - no weight updates are allowed.
  - Giving K examples of context and completion in the prompt. Final example of context and the model is expected to provide the completion. Only small amount of data needed.
  - Does not provide such a good results as fine-tuned models.

Finetuning vs Few shot Learning (2)

- One-Shot (1S)
  - Only one demonstration of context and completion is allowed (plus the natural language description of the task).
  - Helps to communicate to the model the expected (desired) format of the input and output.
- Zero-Shot (0S)
  - no demonstrations are allowed, and the model is only given a natural language instruction describing the task.
  - has potential for robustness and avoidance of spurious correlations, but is also the most challenging setting.
  - Takes some effort to exactly explain expected format and output.

Language Models are Few-Shot Learners

Zero-shot

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

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

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

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

Few-shot

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

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

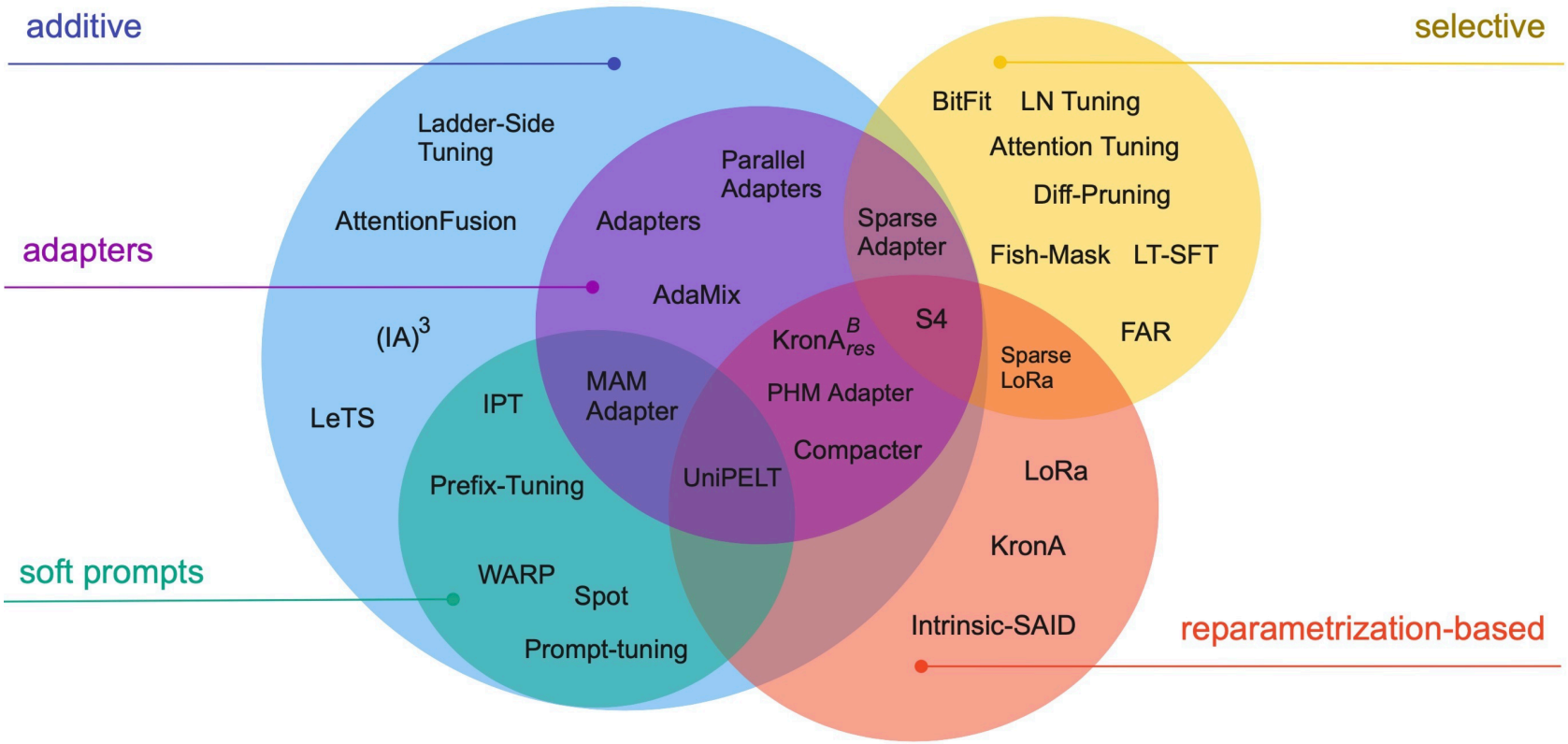
```
1  sea otter => loutre de mer          ← example #1
```
↓
gradient update
↓
```
1  peppermint => menthe poivrée        ← example #2
```
↓
gradient update
↓
...
↓
```
1  plush giraffe => girafe peluche     ← example #N
```
gradient update
```
1  cheese =>                           ← prompt
```

Language Models are Few-Shot Learners

PEFT - Parameter Efficient Fine Tuning

- Training LLM is very expensive. Even smaller models are very costly.
- Even finetuning all the parameters in 175B model needs a lot of resources and can easily fail.
- Fortunatelly, it turns out that you can get updated model behaviour with updating a small subset of weights. Problem is to find the right subset.

Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning



Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning

Types of PEFT

- Additive methods – augmenting the existing pre-trained model with extra parameters or layers and updating only the newly added parameters.
- Adapters – introducing small fully-connected networks after Transformer sub-layers. Variations include modifying the placement of adapters, pruning, or using reparametrization.
- Soft Prompts – idea is to control the behaviour by modifying the input text, which typically consists of a task description accompanied by a few in-context examples.
- Selective methods – fine-tuning only a few top layers of a network; some of the layers (ie. attention); part of the internal structure. In extreme version, completely ignore the structure and select parameters.
- Reparametrization-based methods – leverage low-rank representations to minimize the number of trainable parameters.

LoRA

- There are many dense layers doing linear transformation to it's inputs ($h = \mathbb{W}_0 x$).
- Since we want to update the transformation by a small portion we can look for delta to weight matrix $\mathbb{W}_0 + \Delta\mathbb{W}$.
- To make the search of updates more efficient we constrain the $\Delta\mathbb{W}$ to matrix decomposition $\Delta\mathbb{W} = \mathbf{BA}$

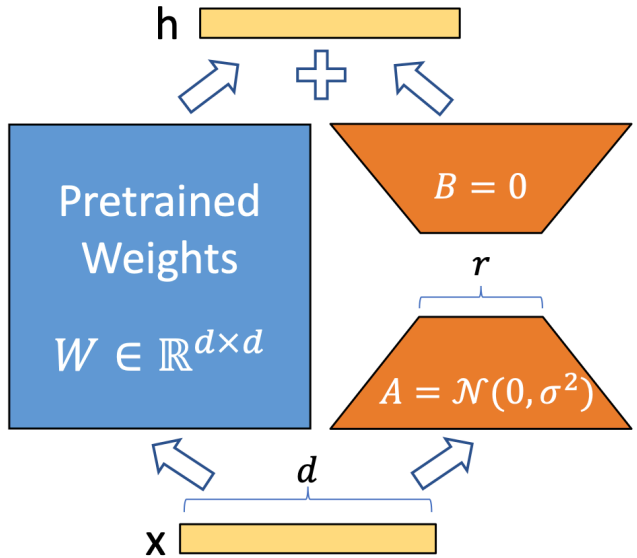


Figure 1: Our reparametrization. We only train A and B.

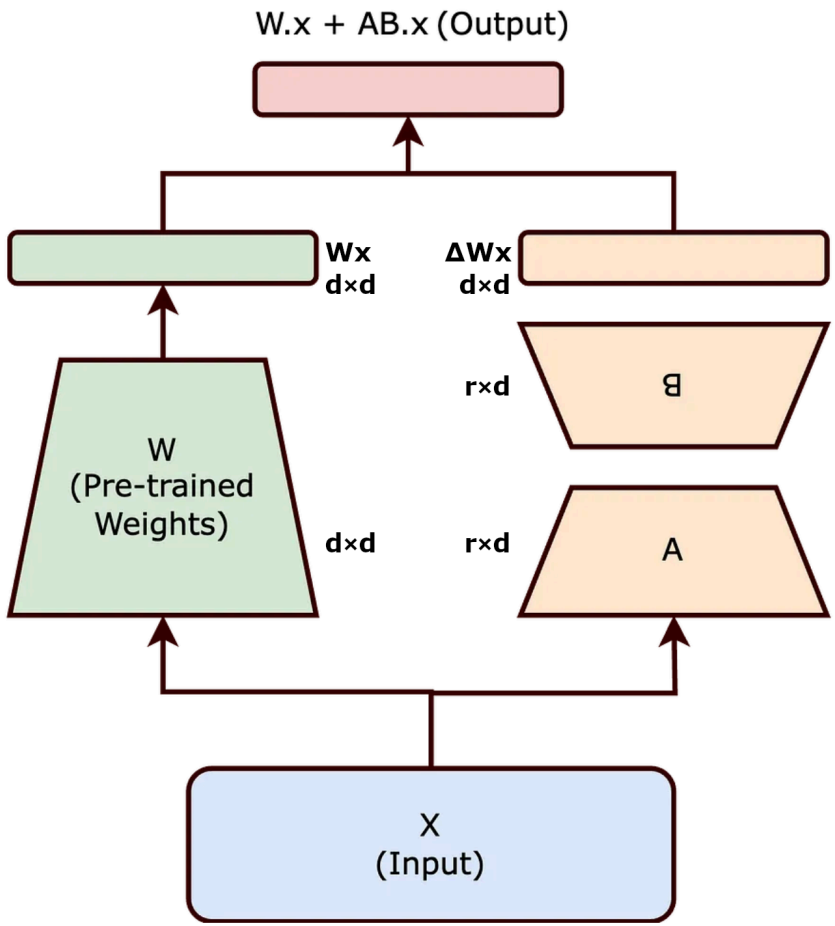
  - $h = \mathbb{W}_0 x + \mathbf{BA}x$
  - When finetuning is finished we can add pretrained $\mathbb{W}_0$ and $\Delta\mathbb{W}$ to get single matrix and remove any overhead during inference time.

LoRA: Low-Rank Adaptation of Large Language Models

Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning

LoRA (2)

- In theory LoRA can be used to any neural network based model, but is really effective with LLM.
- The paper suggest to apply the updates only to the attention weights and freeze all MLP modules.



LoRA for Fine Tuning (Medium Blog)

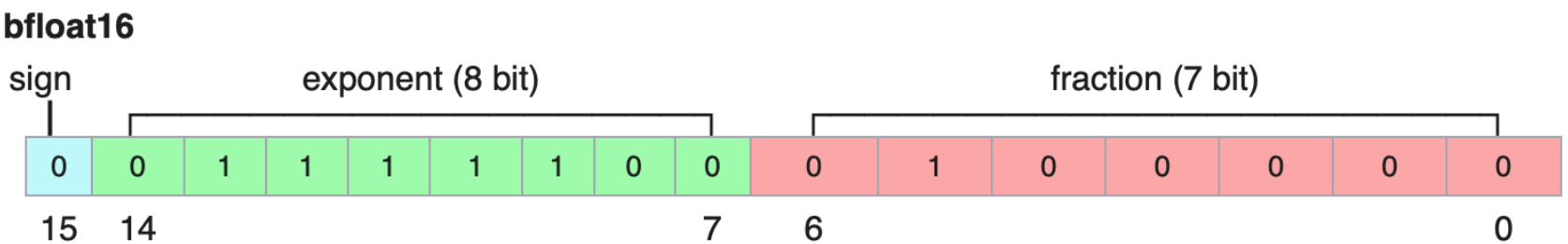
Understanding LoRA (Towards Datascience Blog)

LoRA Training Process

- The training process is really straightforward.
- Use training data and target to calculate model output and error of the output.
- The error is then back propagated. Only gradients associated with $\mathbf{A}$ and $\mathbf{B}$ decomposition matrices are used to update the values.
- When finetuning is finished, we can collapse $\mathbb{W}_0 + \Delta\mathbb{W}$ into single matrix and remove any additional overhead for the inference.
- If $\Delta\mathbb{W}$ or $\mathbf{A}$ and $\mathbf{B}$ are stored, you can also quickly "unlearn" the updates and repurpose the model for the next task.

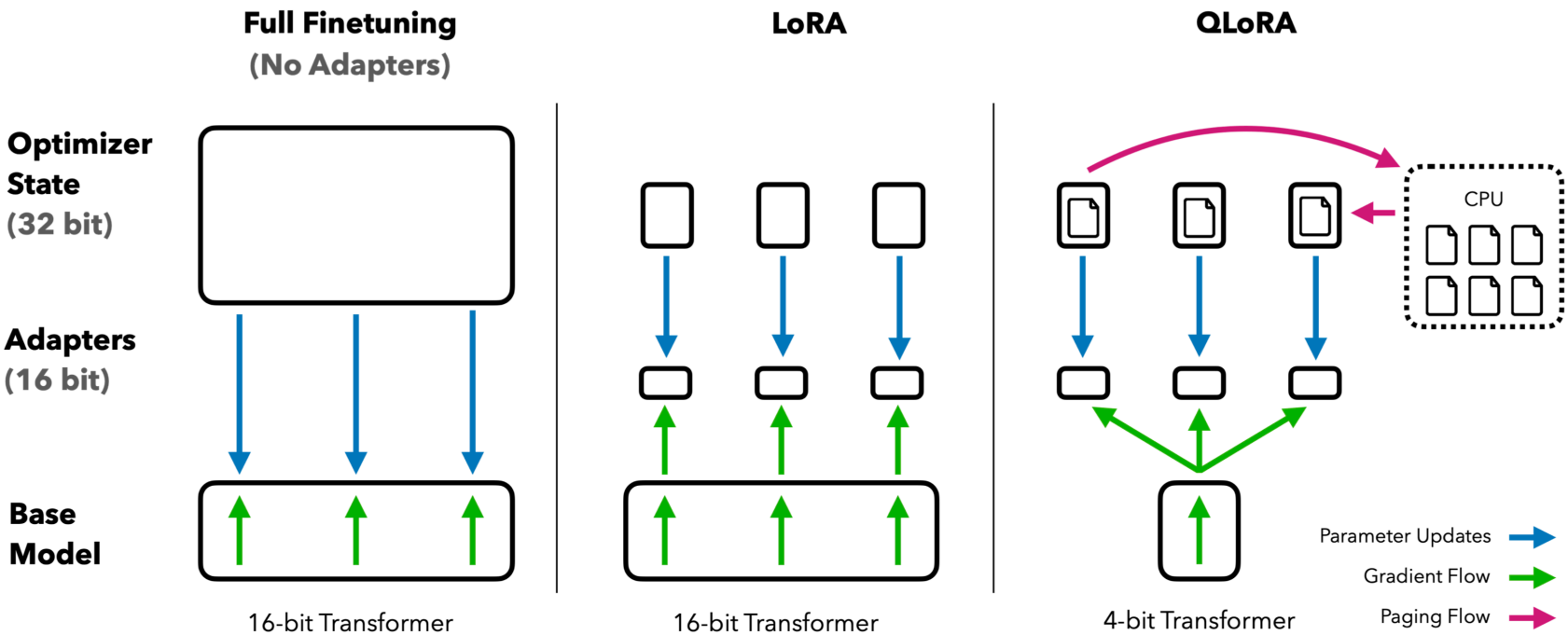
QLoRA - Quantised LoRA Finetuning

- With double quantisation (above) and some hardware tweaks, the finetuning process is relatively simple.
- Weights are stored in 4-bit NormalFloat representation with double quantisation. The result is that model is much smaller to store without loss of performance.
  - In paper - reduces memory requirement of 65B parameter model from $\sim$ 780GB to $\sim$ 48GB.
- The data, dequantised weights and calculations are performed with BFloat16 data type.



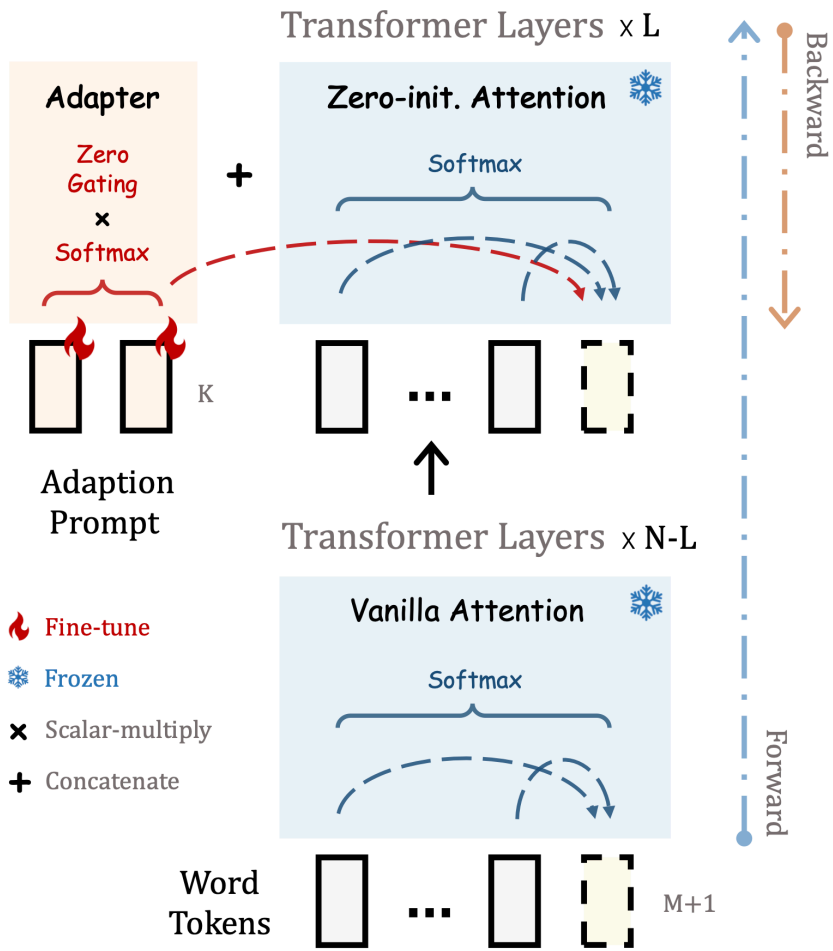
QLoRA

- As with LoRA algorithm, QLoRA uses decomposition of matrices to find updates to fixed weight matrices in attention heads.
  - $\mathbf{Y} = \mathbf{X}\text{dequant}(c_1^{\text{FP32}}, c_2^{\text{4-bit INT}}, \mathbb{W}^{\text{NF4}}) + \mathbf{X}\mathbb{A}\mathbb{B}$
  - $\mathbf{Y}$, $\mathbf{X}$, $\mathbf{A}$, $\mathbf{B}$ are all in BFloat16 format.
- During the training process (as in LoRA), only gradients of error with respect to decomposion matrices $\frac{\partial E}{\partial \mathbf{A}}$ and $\frac{\partial E}{\partial \mathbf{B}}$ are calculated with BFloat16 precision.
- Only decomposed matrices $\mathbf{A}$ and $\mathbf{B}$ are updated in gradient descent.


Full Finetuning
(No Adapters)
LoRA
QLoRA
Optimizer State
(32 bit)
Adapters
(16 bit)
Base Model
CPU
Parameter Updates
Gradient Flow
Paging Flow
16-bit Transformer
16-bit Transformer
4-bit Transformer

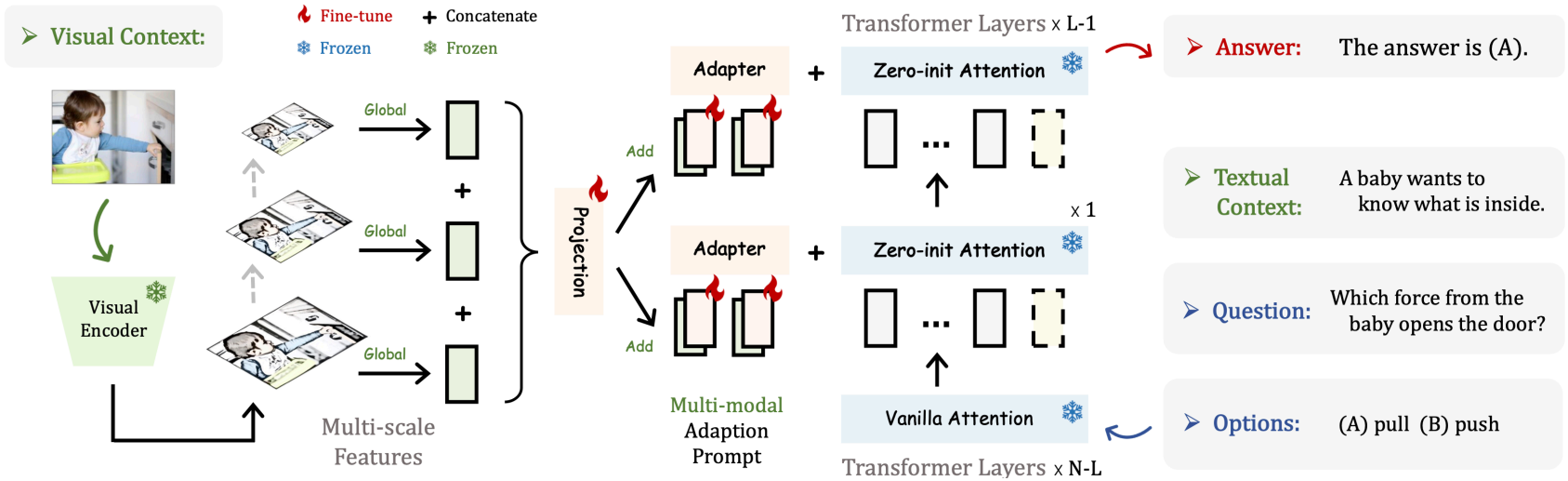
LLaMA-Adapter

- Freezes parameters in the whole LLaMa model and adds learnable adoption prompts into the later transformer layers.
- These prompts learn to adaptively inject new instructions (conditions) into the frozen LLaMA.
- Attention mechanism is adapted so the new layers are have zero-initialised attention and learnable gating.
  - Used to avoid random noise to clutter results in the beginning of the training process.



LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention

LLaMA-Adapter Multimodal Data



- The adapter can ingest also other type of information (images) to augment the model with additional information.
- Given visual, textual contexts and question (+ options), the model has to process this multi-modal information to give the correct answer.

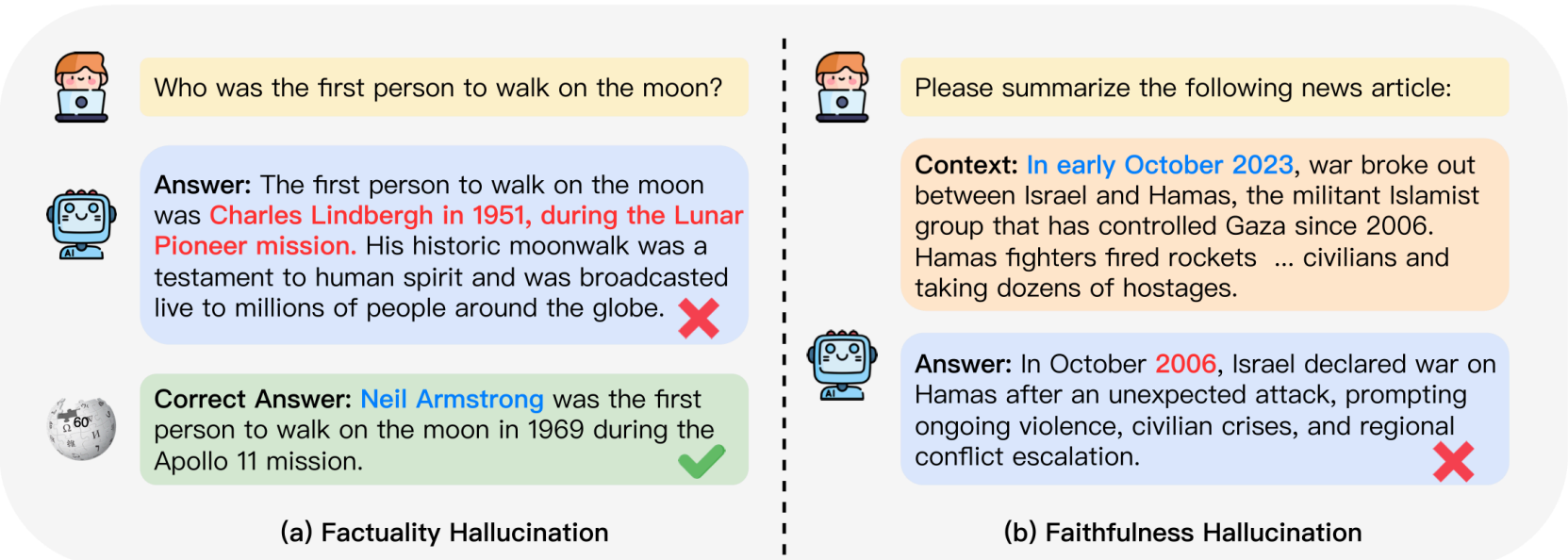
LLM Hallucinations

- Hallucinations are generated content that is nonsensical or unfaithful to the provided source content.
- The paper below recognises:
  - *factuality hallucination* means the discrepancy between generated content and verifiable real-world facts.
  - *faithfulness hallucination* refers to the divergence of generated content from user in- structions or the context provided by the input, as well as self-consistency within generated content.

A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions

The Beginner's Guide to Hallucinations in Large Language Models (Blog)

Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models



A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions

Sources of Hallucination

- Hallucination from Data
  - Misinformation and Biases
  - Knowledge Boundary
- Wrong Data Utilisation
  - Knowledge Shortcuts
  - Knowledge Recall Failures
- Hallucination from Training
- Hallucination from Alignment
- Hallucination from Inferece

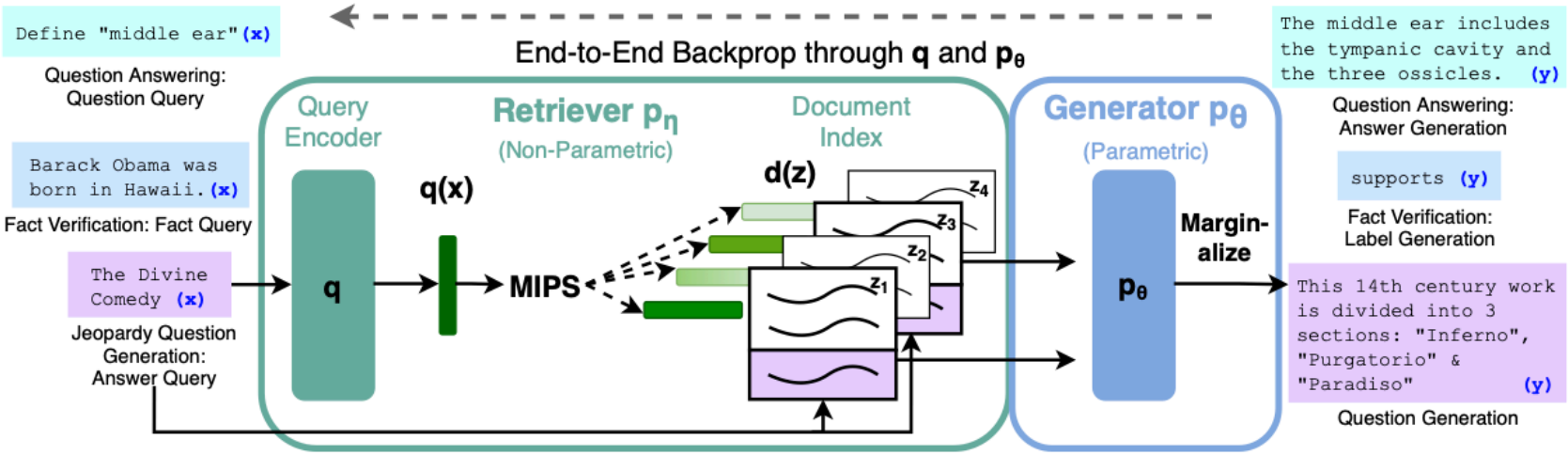
Hallucination Mitigation

- Prompt Engineering – process of experimenting with various instruction in context to get the best output possible.
  - Retrieval-Augmented Generation (RAG) – utilising external database to store more sources than context can handle.
  - Self-Refinement through feedback and reasoning – output for a specific prompt is evaluated and with the feedback is send back to LLM to iteratively improve.
- Prompt Tuning – adjusting the prompt with 'Soft Prompts', learned through backpropagation during the fine-tuning.
- Model Development – extend model architecture to tackle hallucinations.
  - Different decoding strategies
  - Knowledge Graph
- Faithfulness Aware Loss Function

A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models

Retrieval Augmented Generation

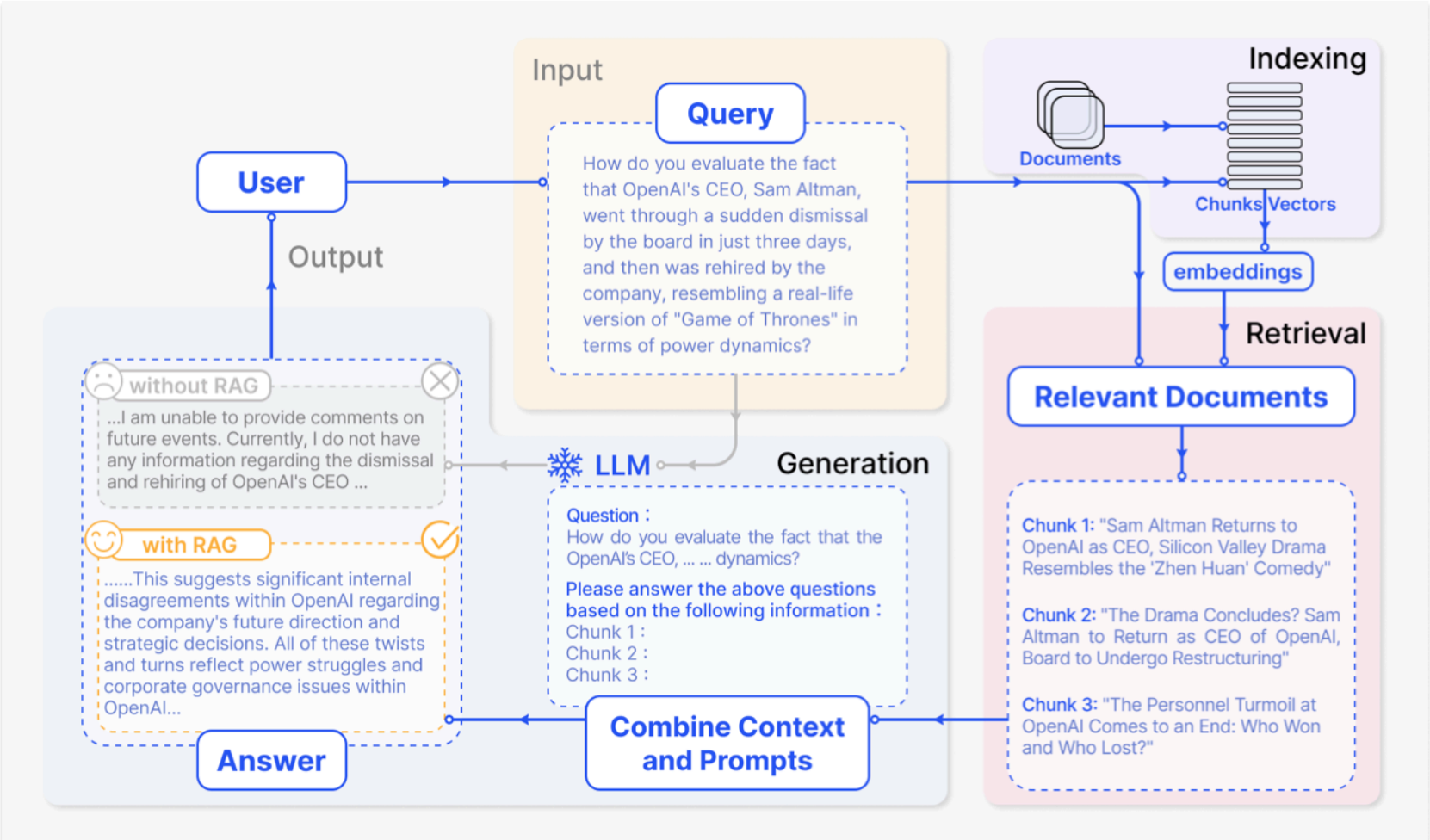
- Originally introduced for Seq2Seq model, but the idea is easily transmissible into any other text generation model.



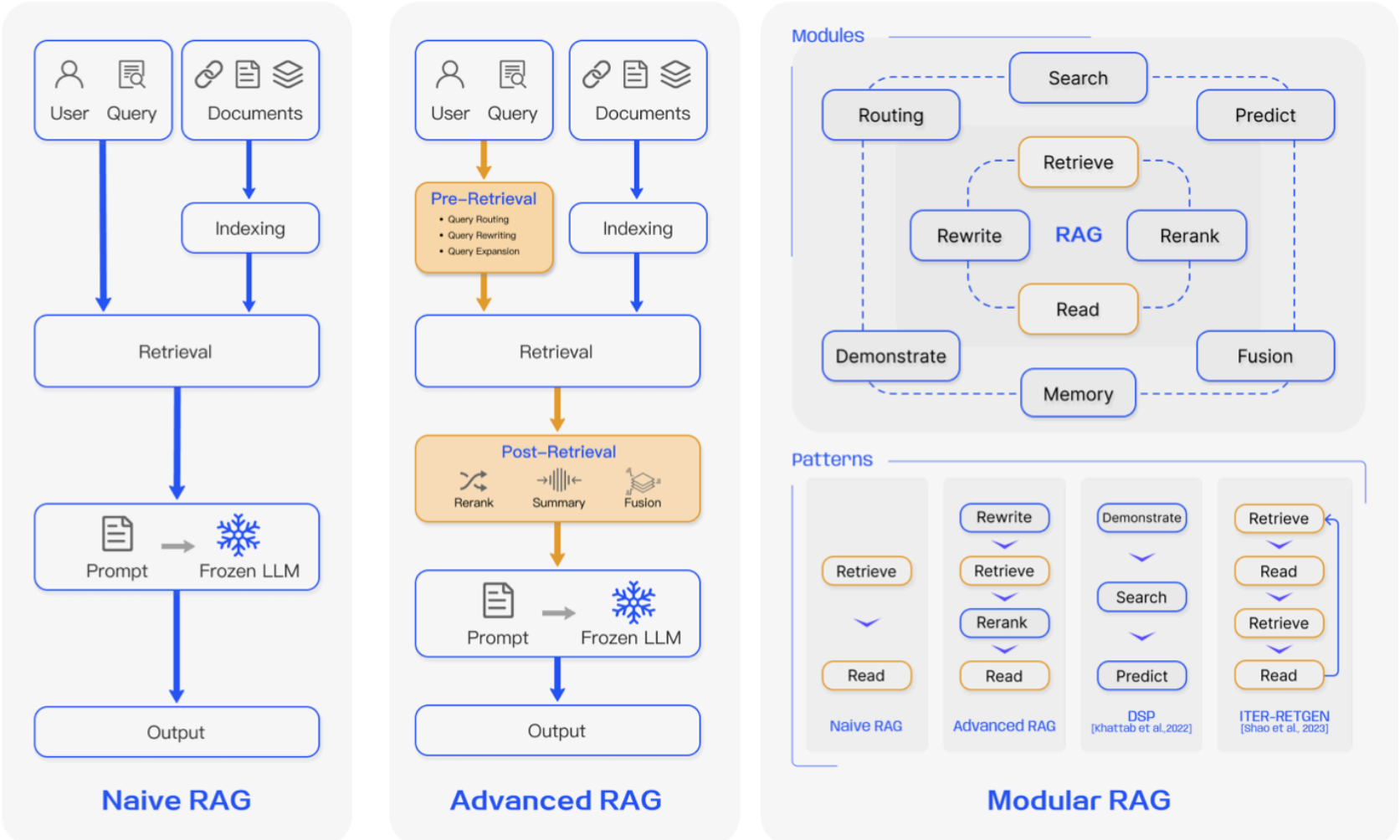
Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Navigating the Challenges of Hallucinations in LLM Applications: Strategies and Techniques for Enhanced Accuracy (Blog)

How to reduce hallucination in a Large Language Model (LLM)? (Blog)



Retrieval-Augmented Generation for Large Language Models: A Survey



Retrieval-Augmented Generation for Large Language Models: A Survey

Further Reading

- A Survey on Model Compression for Large Language Models
- Beyond Efficiency: A Systematic Survey of Resource-Efficient Large Language Models
- https://arxiv.org/pdf/2404.10779
- A Survey on Evaluation of Large Language Models