

Transformers from Particle Clouds to Autonomous Scientific Agents

Ahmed Hammad

Center of AI for natural science, KIAS

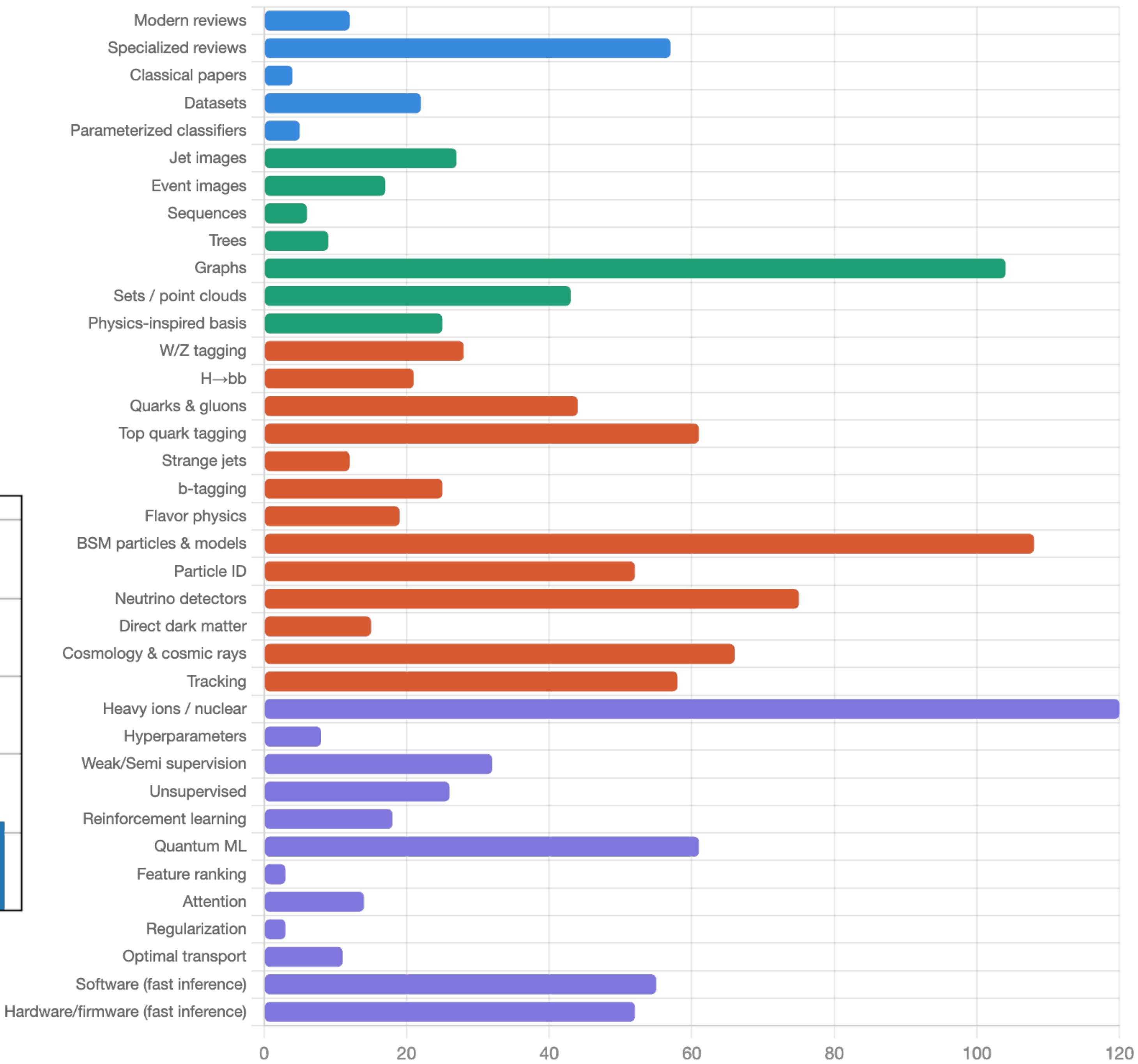
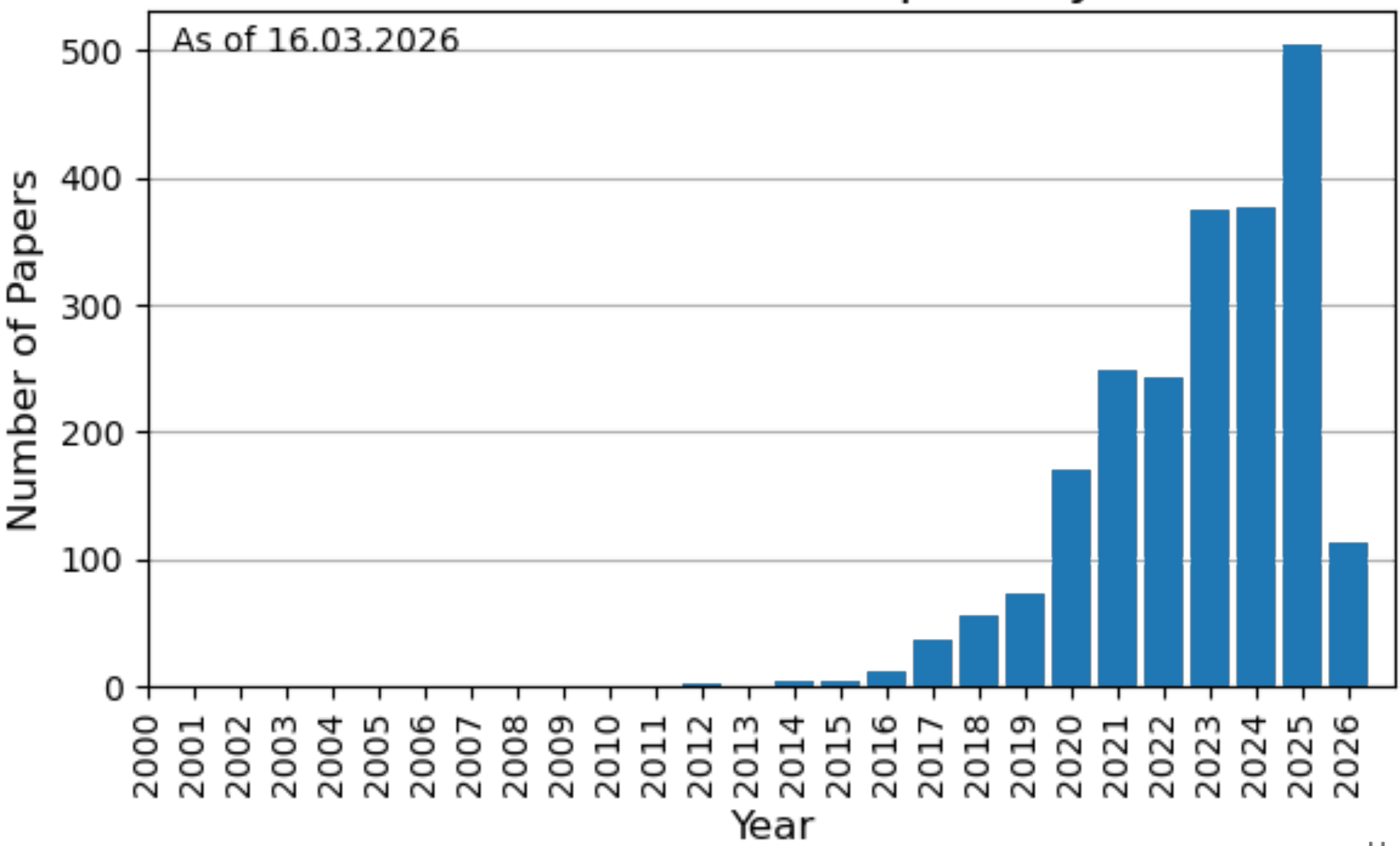
PARTIQAL 2026, IBS, Daejeon

Motivation

HEPML living review GitHub repo.

<https://github.com/iml-wg/HEPML-LivingReview/tree/master>

Number of HEP-ML Papers by Year



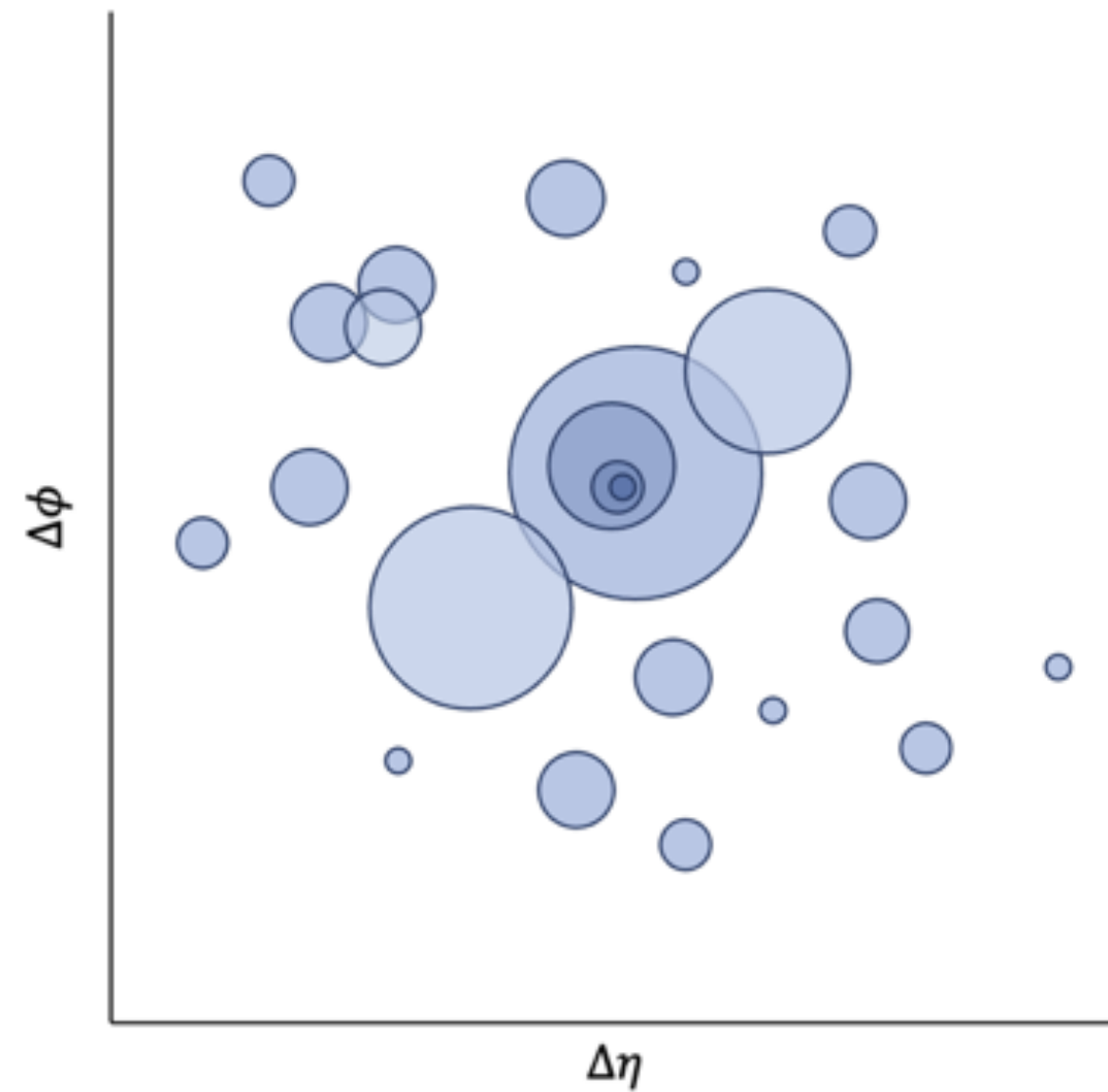
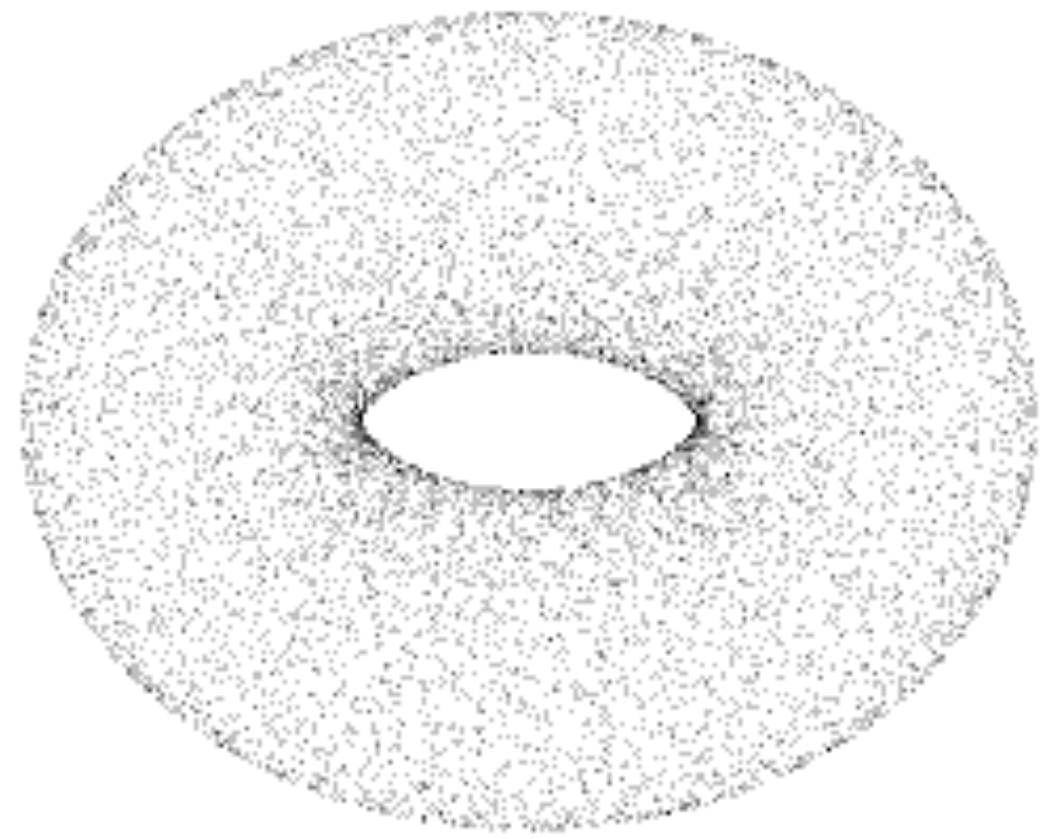
Why Do Transformers Outperform Other Neural Architectures?

(1) Capability to process diverse data types

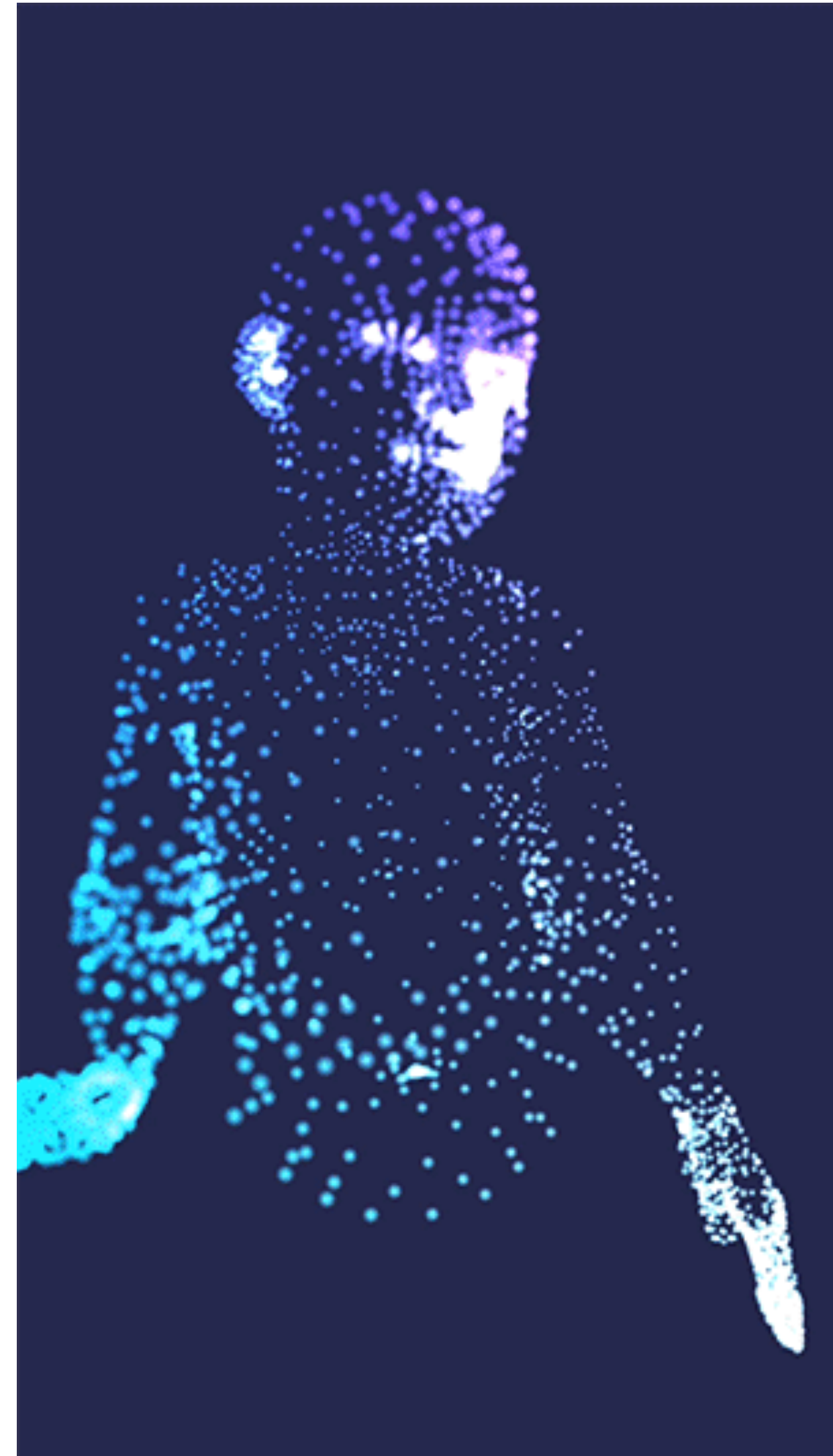
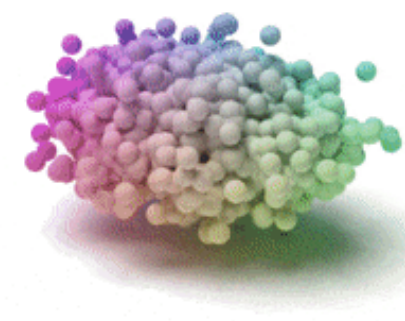
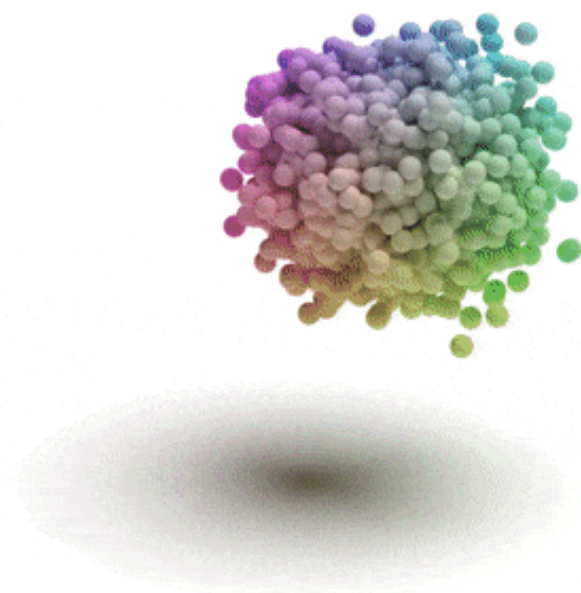
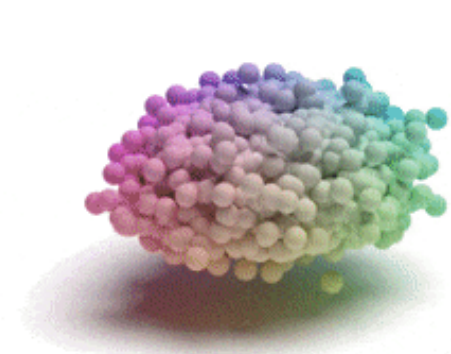
(2) Scaling Law

Transformers for point cloud

A point cloud is an unordered collection of points in space that samples a geometric object or physical system.

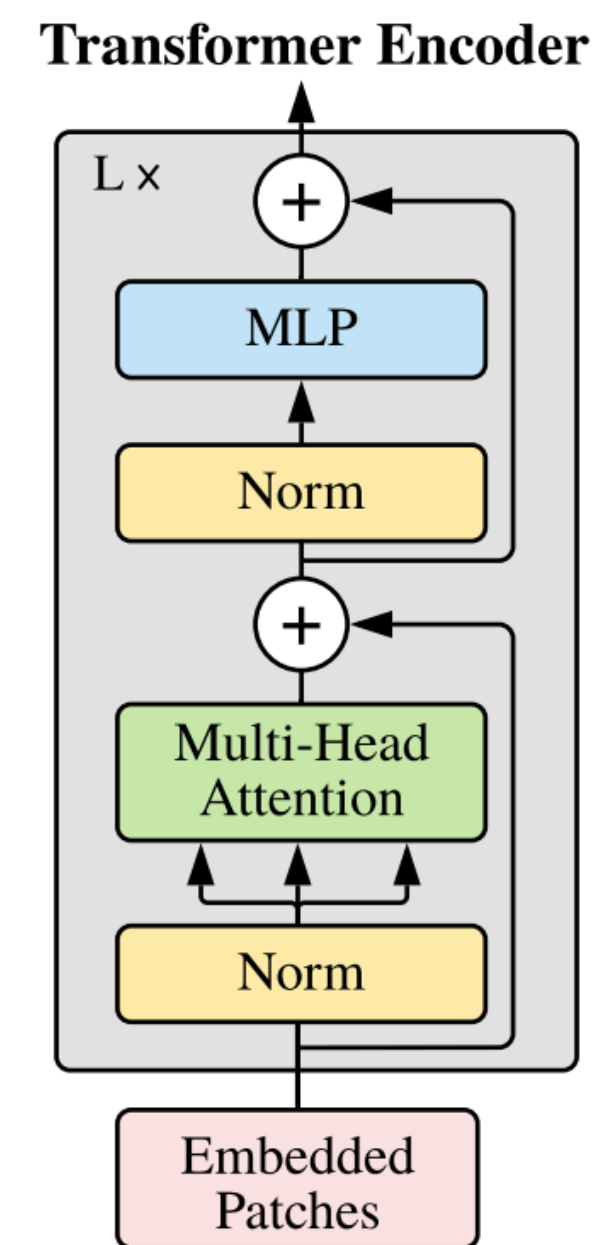
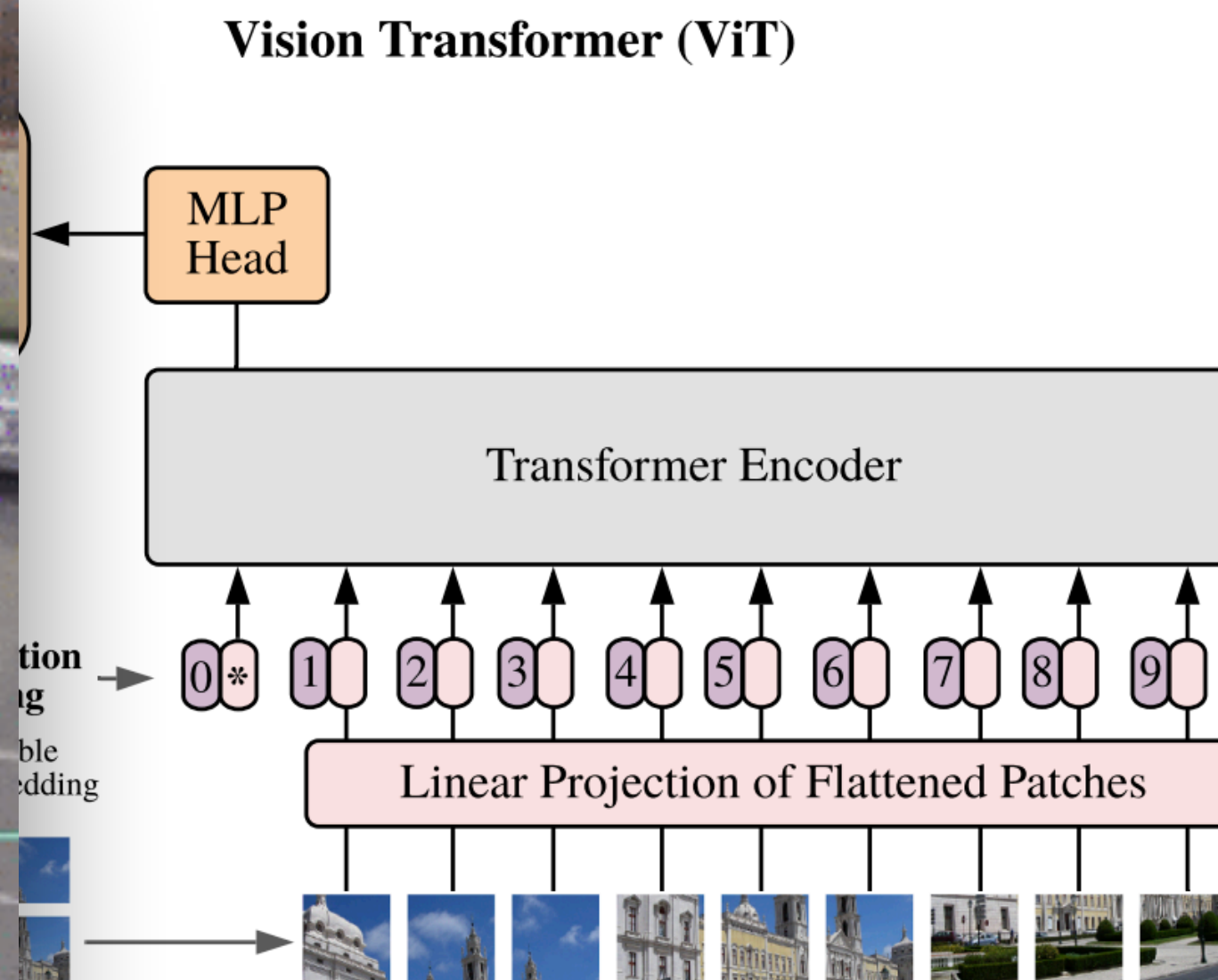
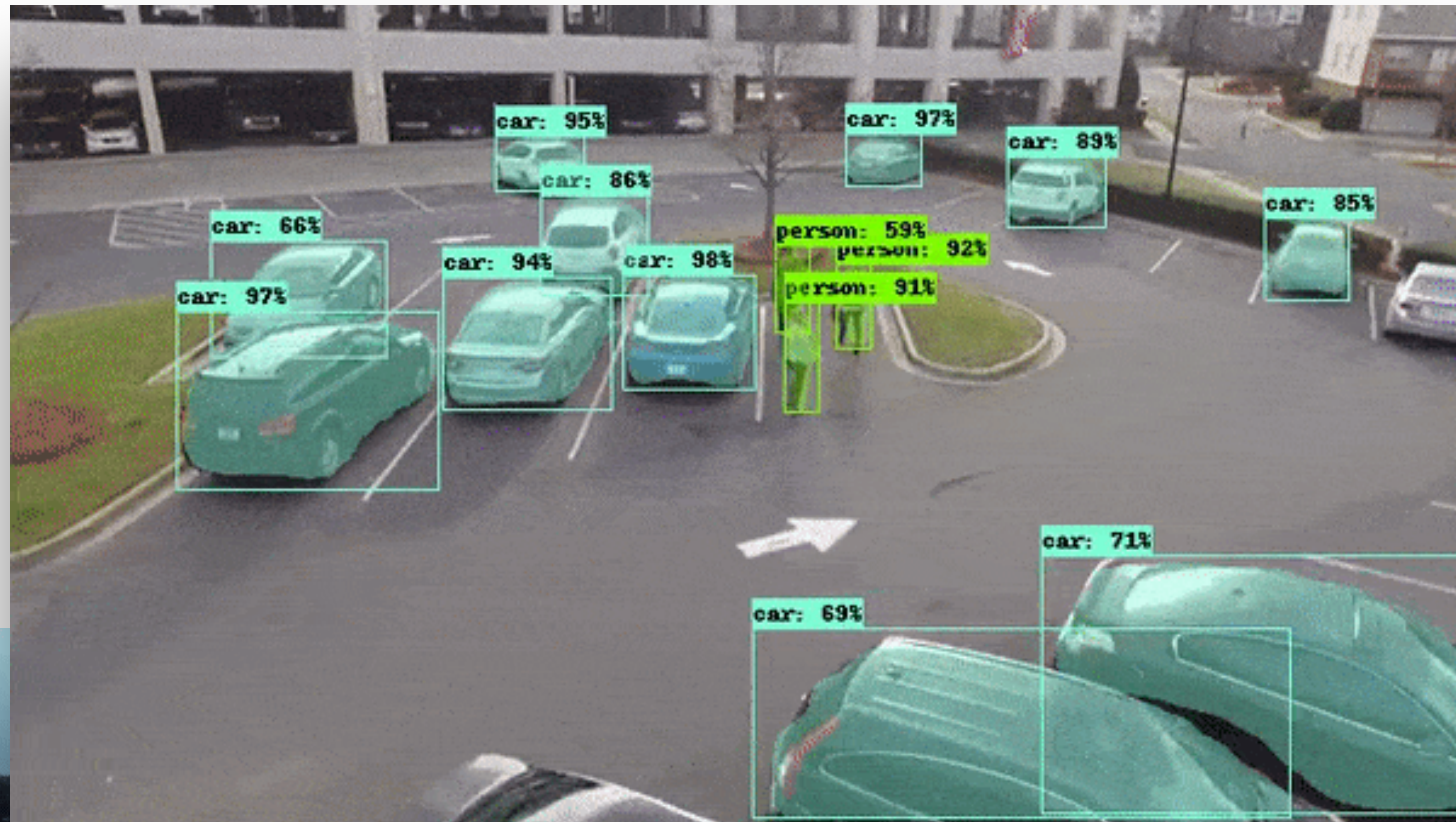


Particle Cloud

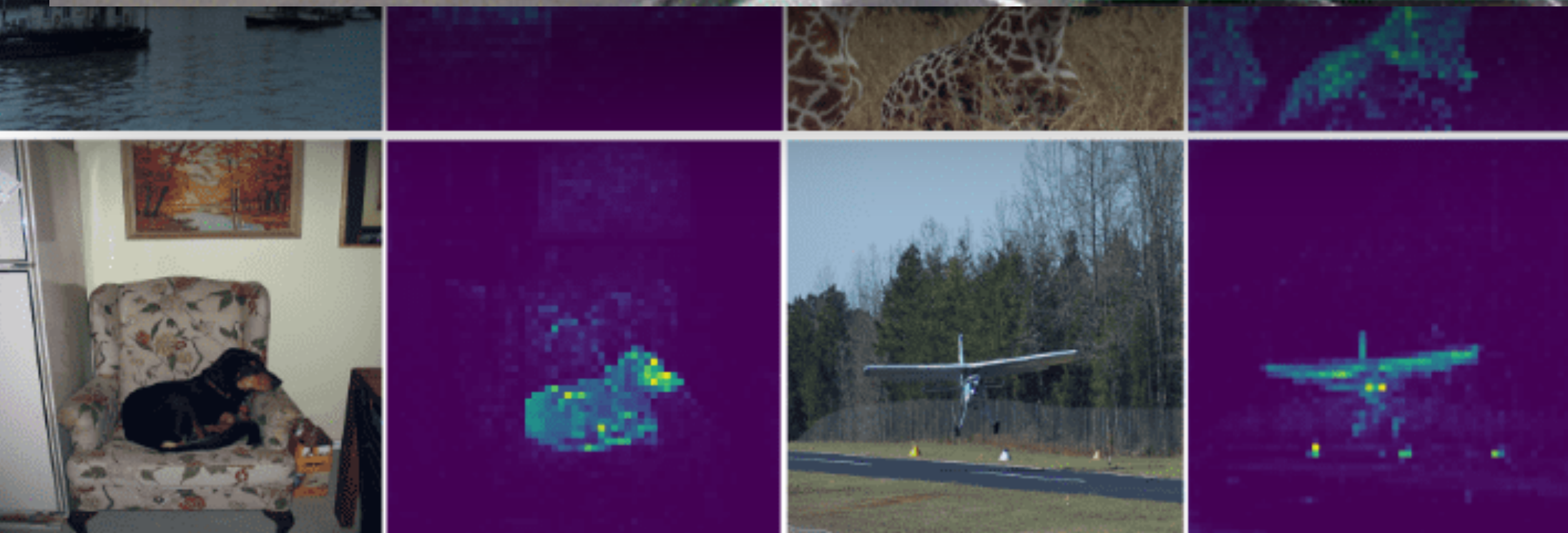


Transformers for Images

Vision Transformer is a Transformer based model to analyze image datasets



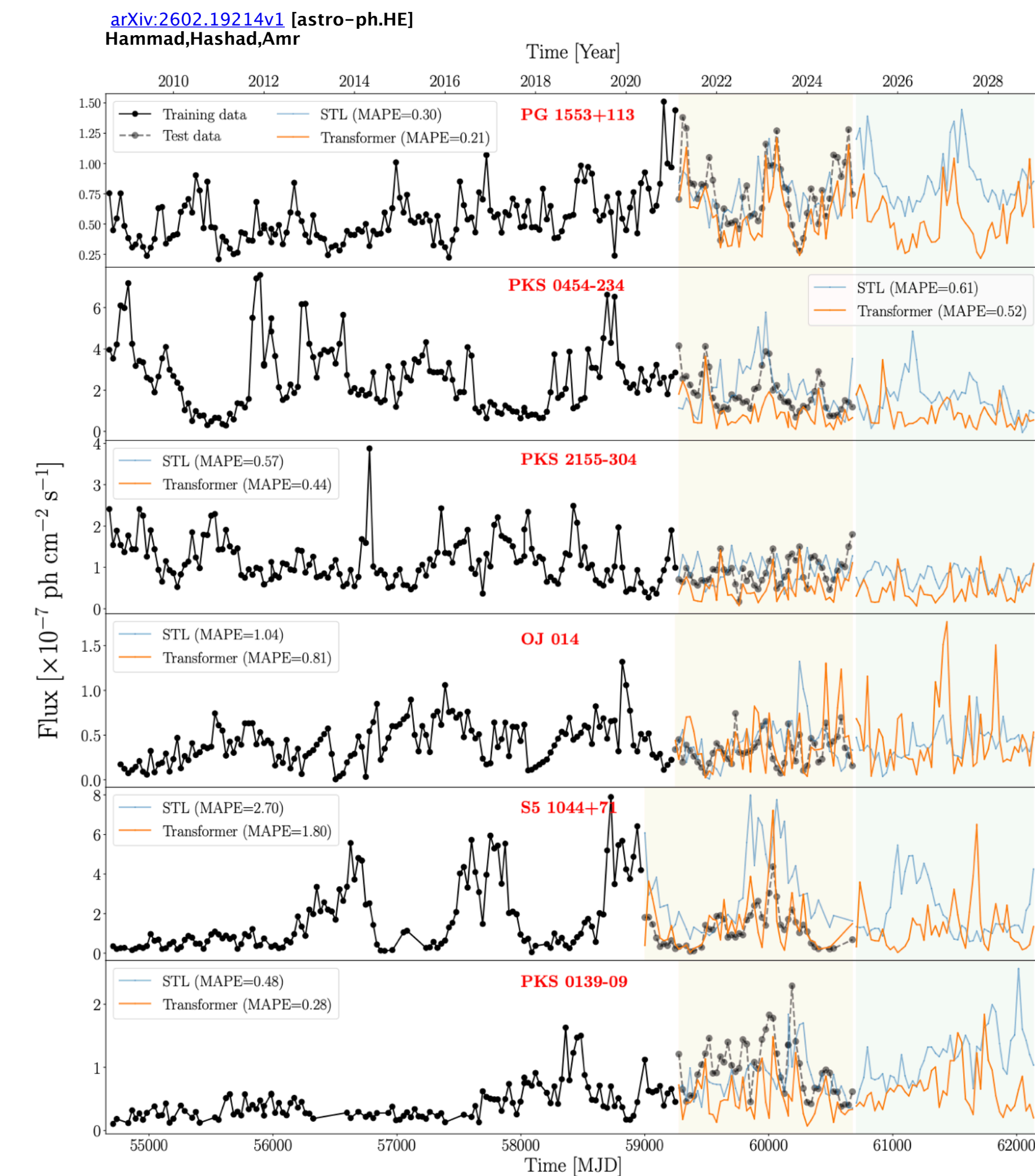
[arXiv:2010.11929v2](https://arxiv.org/abs/2010.11929v2)



Original images after attention layers

Transformers for time series

Time series is a sequence of datapoints ordered in time



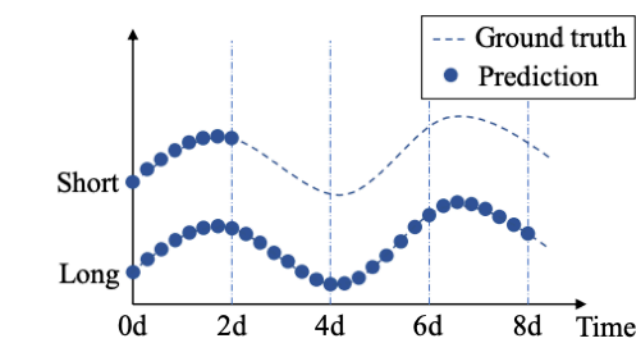
Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting

Haoyi Zhou,¹ Shanghang Zhang,² Jieqi Peng,¹ Shuai Zhang,¹ Jianxin Li,¹
Hui Xiong,³ Wancai Zhang⁴

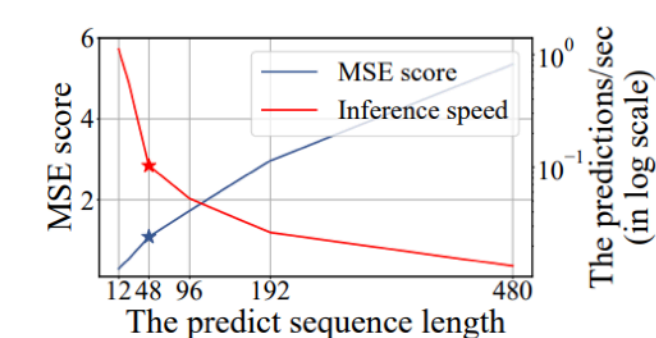
¹ Beihang University ² UC Berkeley ³ Rutgers University ⁴ SEDD Company
{zhouhy, pengjq, zhangs, lijx}@act.buaa.edu.cn, shz@eecs.berkeley.edu, {xionghui,zhangwancaibuaa}@gmail.com

Abstract

Many real-world applications require the prediction of long sequence time-series, such as electricity consumption planning. Long sequence time-series forecasting (LSTF) demands a high prediction capacity of the model, which is the ability to capture precise long-range dependency coupling between output and input efficiently. Recent studies have shown the potential of Transformer to increase the prediction capacity



(a) Sequence Forecasting.



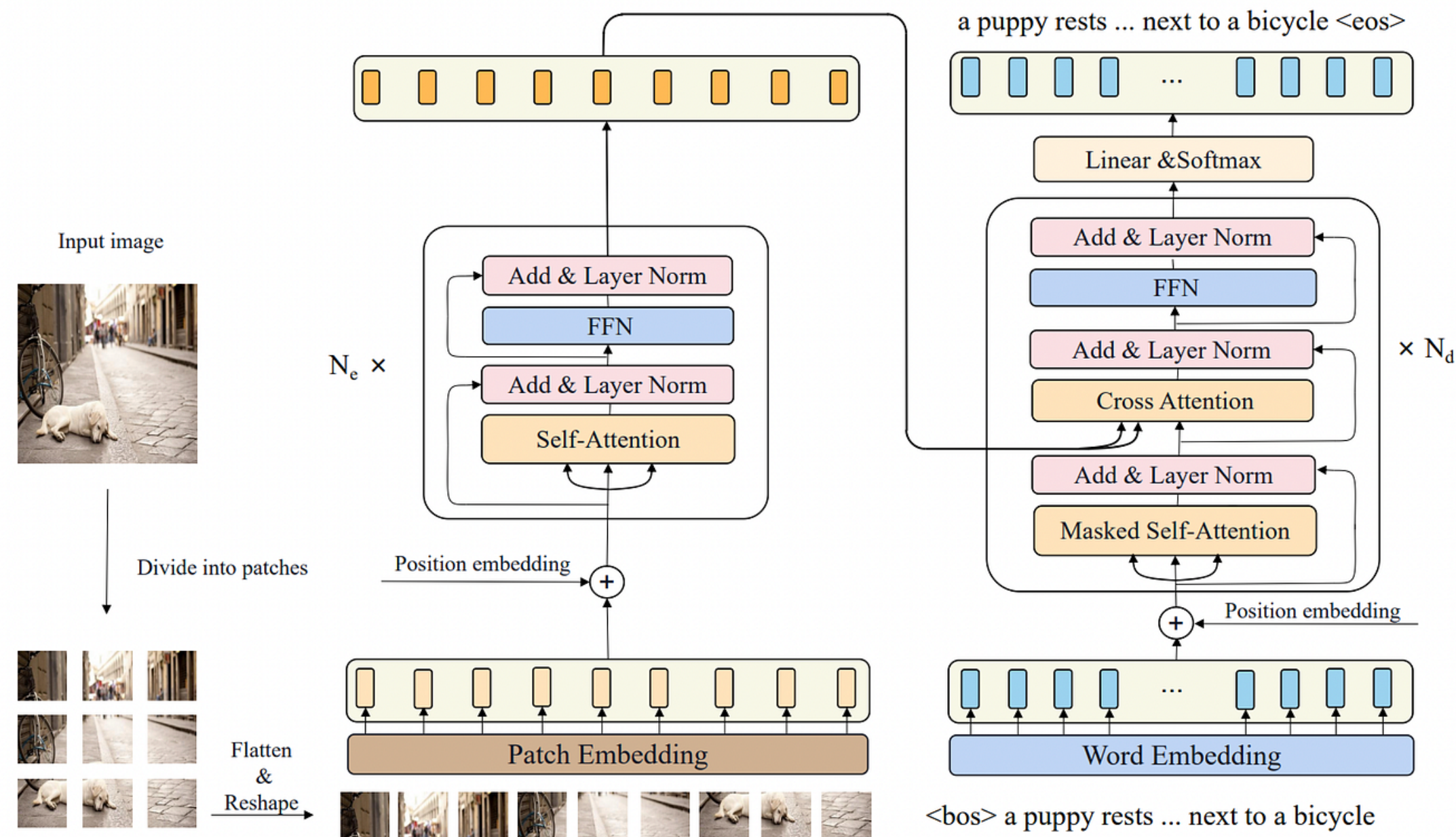
(b) Run LSTM on sequences.

Forecasting the quasi-periodicity in gamma-ray blazers in collected data by Fermi-LAT

Transformers

While Transformer can analyze different data types individually
It Can analyze cross tasks. Such as image to text
and audio to video, etc

Image caption generation



VATT: Transformers for Multimodal Self-Supervised Learning from Raw Video, Audio and Text

Hassan Akbari*
Columbia University
ha2436@columbia.edu

Liangzhe Yuan
Google
lzyuan@google.com

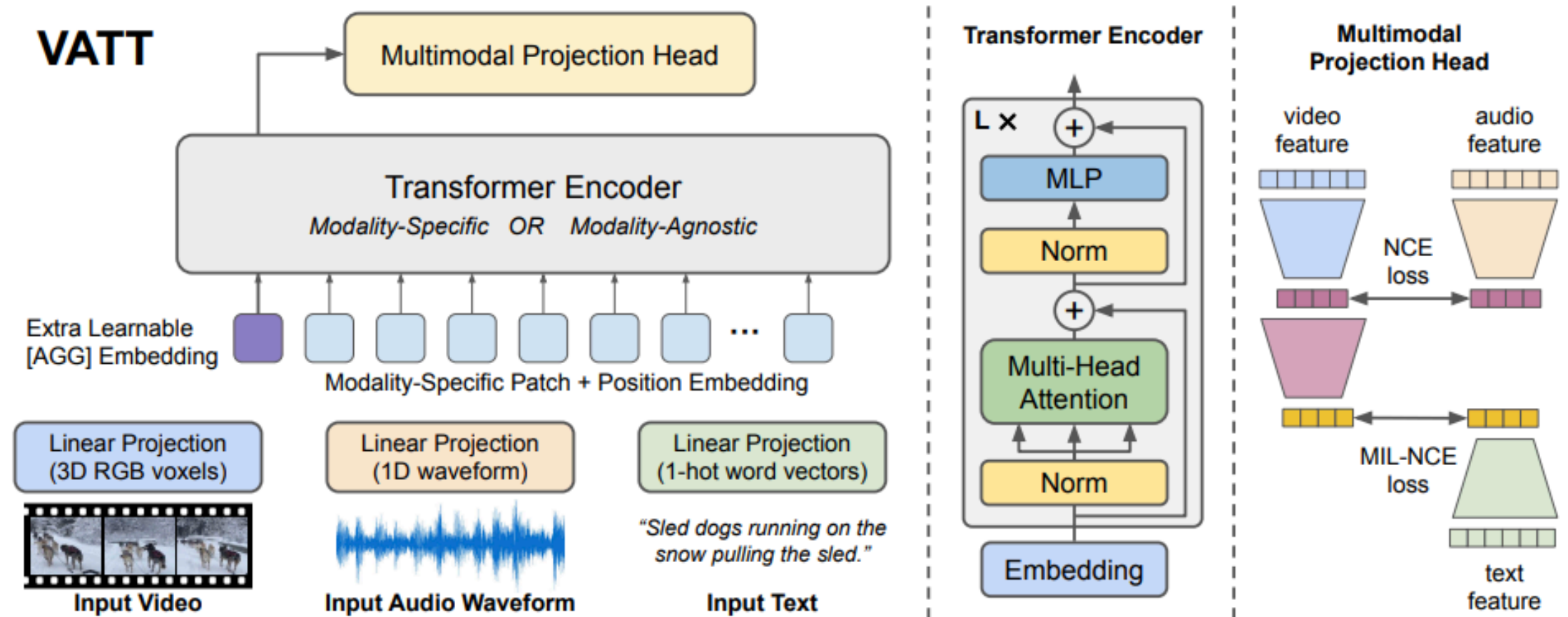
Rui Qian*
Cornell University
rq49@cornell.edu

Wei-Hong Chuang
Google
whchuang@google.com

Shih-Fu Chang
Columbia University
sc250@columbia.edu

Yin Cui
Google
yincui@google.com

Boqing Gong
Google
bgong@google.com



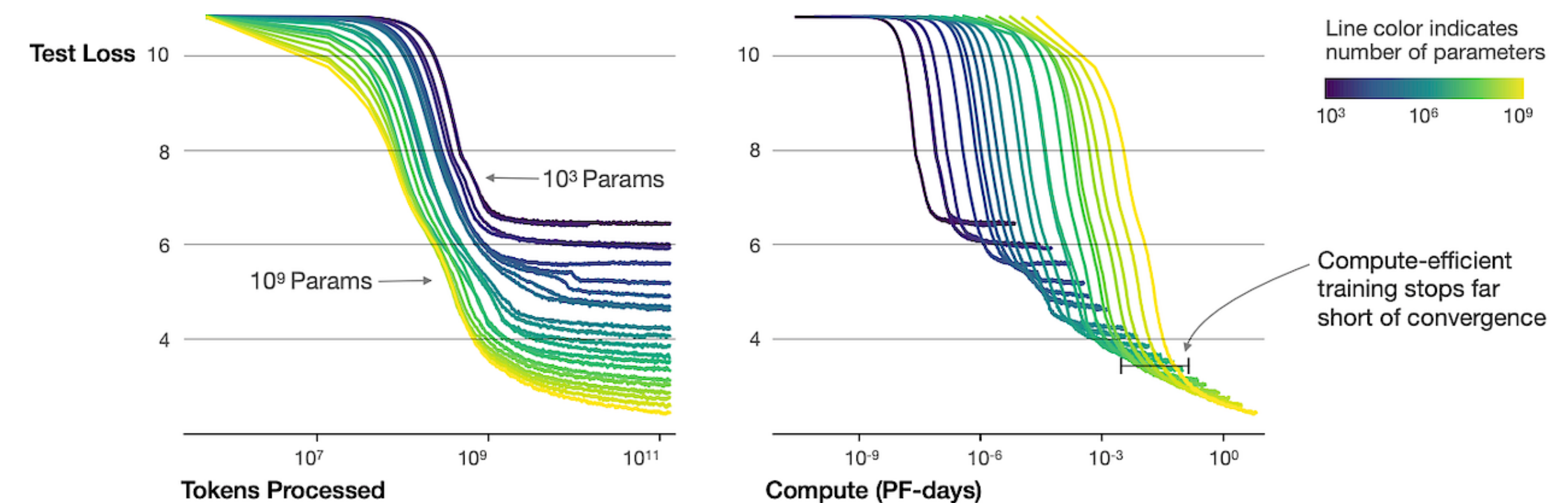
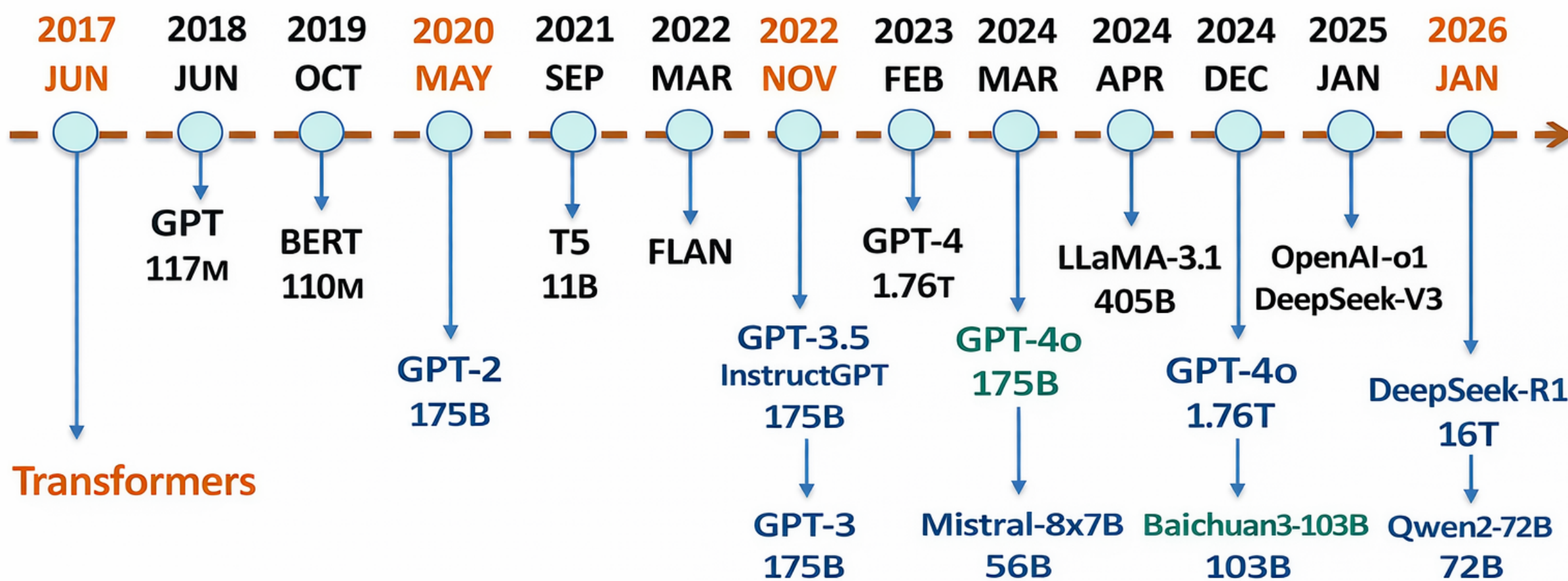
Scaling Law

Scaling Law

Scaling Law: Transformer performance improves predictably with increasing model size, training data and compute resources.

Scaling law is pushing LLMs into a new era

Transformers became dominant by leveraging self-attention to scale more effectively than previous architectures.



Can we find a proper transformation for more efficient small size LLM ?

Task	1B	7B	13B	30B	70B	175B	540B
Code completion	☐	☒	☒	☒	☒	☒	☒
3-digit arithmetic	☐	☐	☒	☒	☒	☒	☒
Anagram detection	☐	☐	☐	☒	☒	☒	☒
Logical fallacies	☐	☐	☐	☒	☒	☒	☒
Causal reasoning	☐	☐	☐	☐	☒	☒	☒
Multi-step math	☐	☐	☐	☐	☐	☒	☒
Novel analogies	☐	☐	☐	☐	☐	☒	☒
Complex instruction	☐	☐	☐	☐	☐	☐	☒

☒ Capable
 ☐ Not yet capable

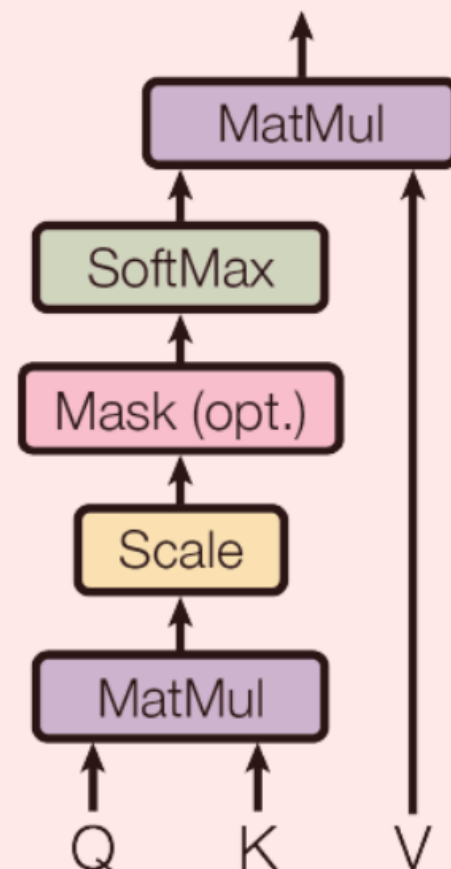
Before moving forward, let's see how
Transformer works

Transformer Encoder

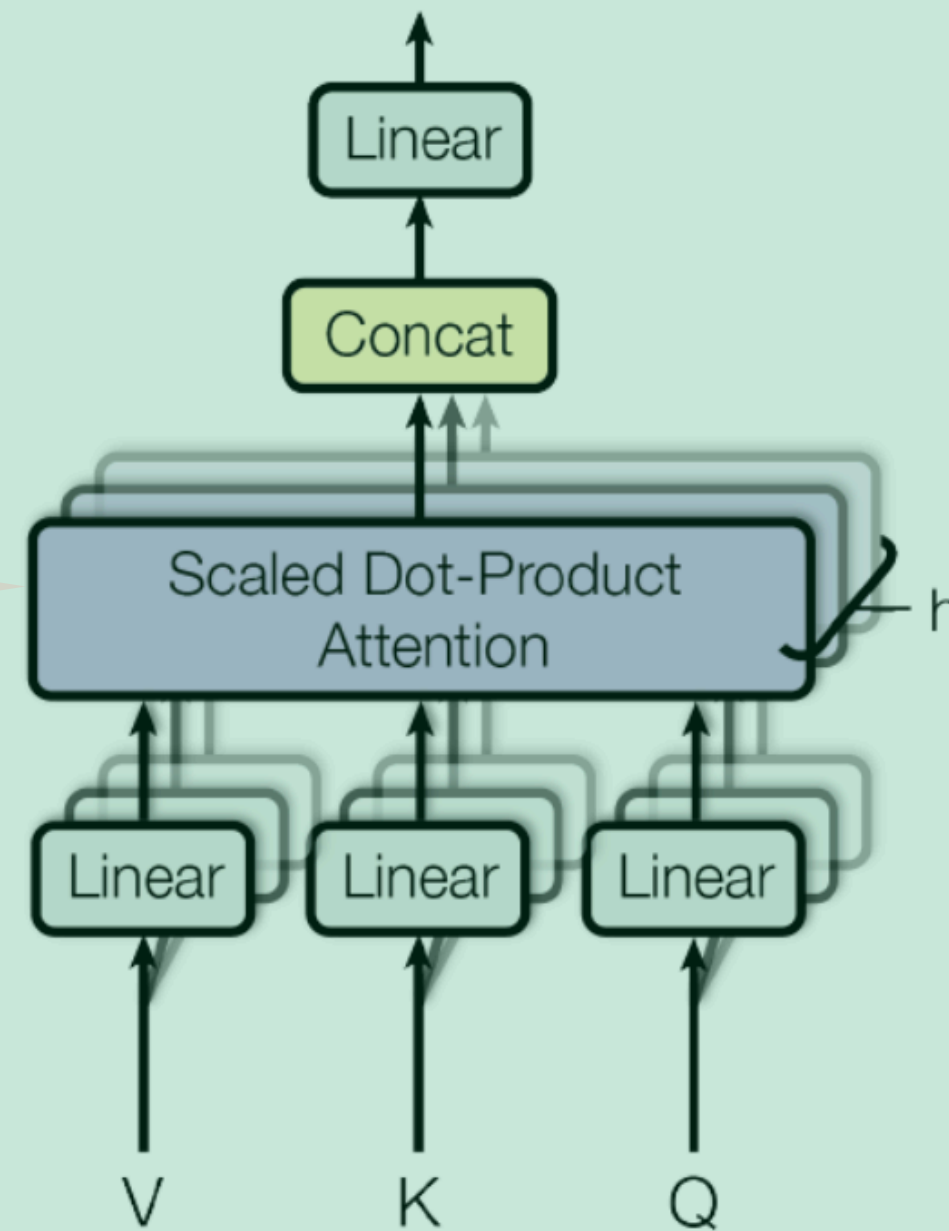
Transformer is first introduced for language translation. However, in collider analysis we use the encoder part only

Attention is all you need
[arXiv:1706.03762](https://arxiv.org/abs/1706.03762) [cs.CL]

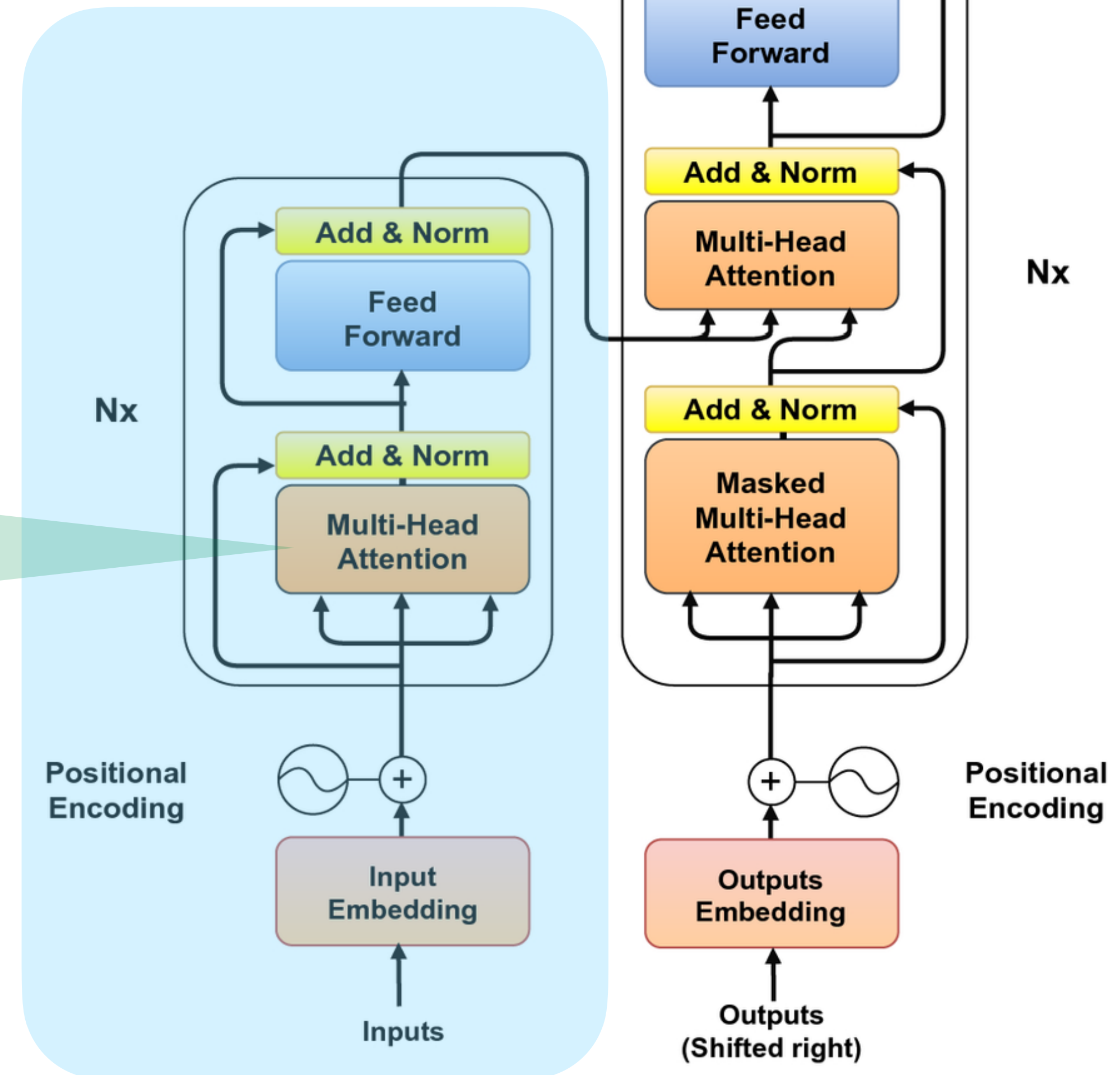
Scaled Dot-Product Attention



Multi-Head Attention



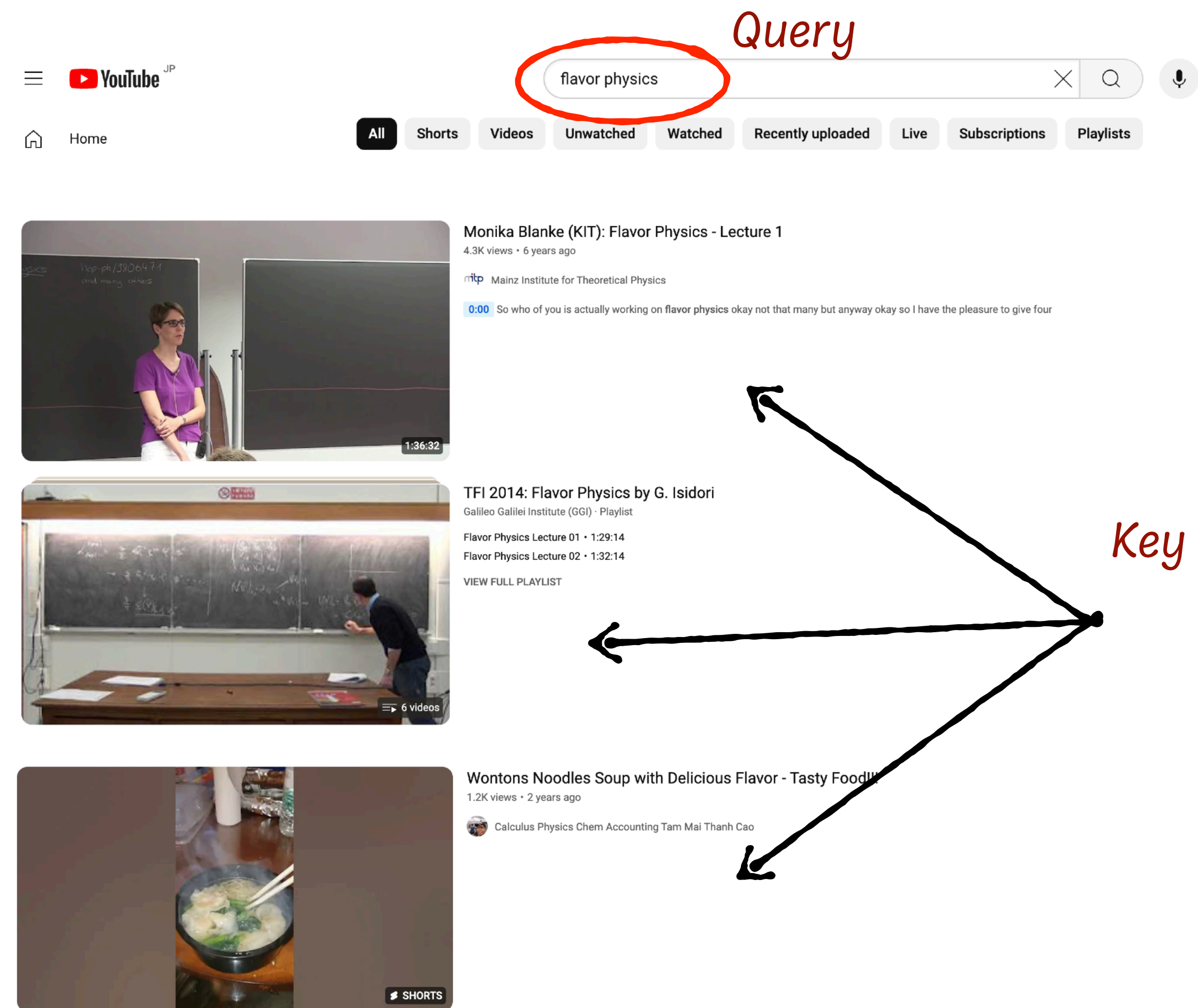
Transformer Encoder



For point cloud analysis, we don't use positional encoding

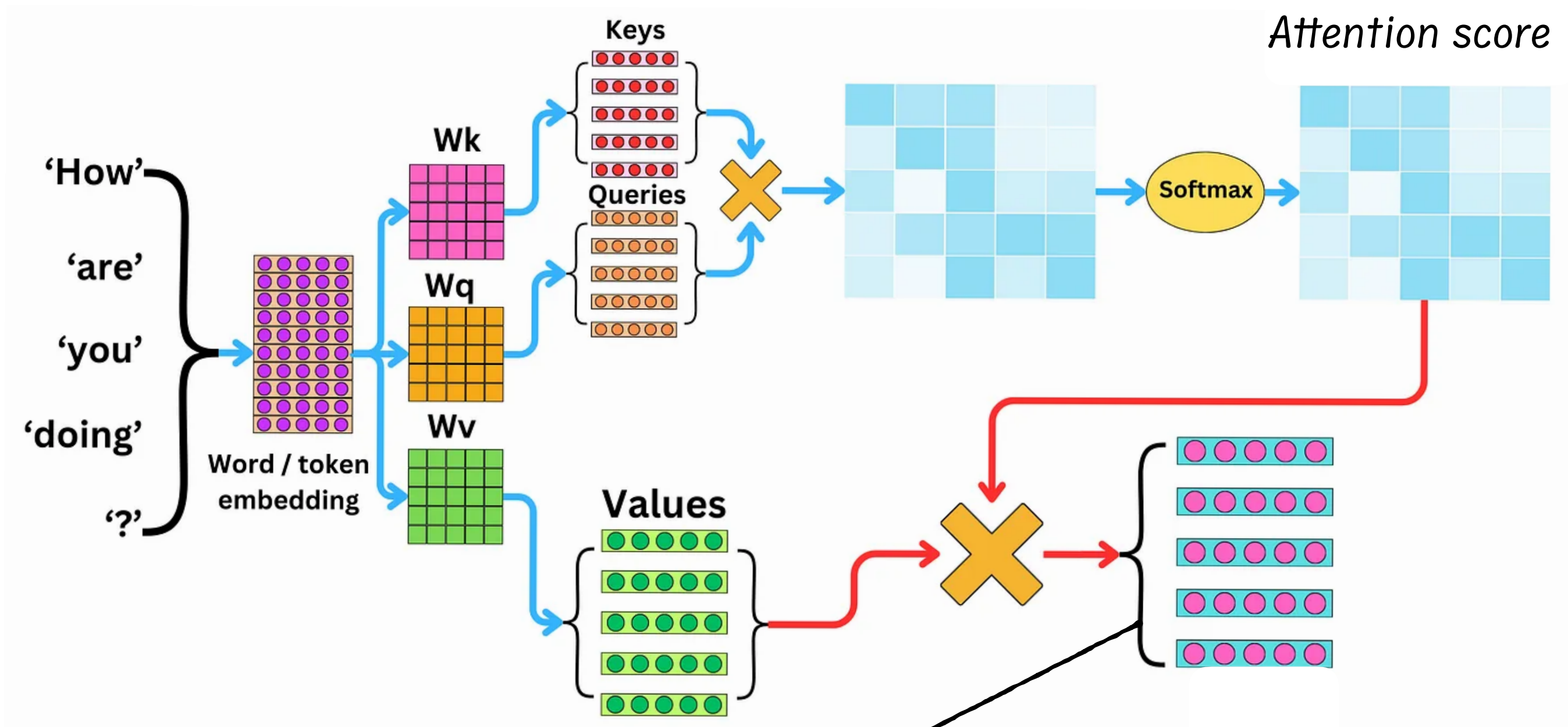
Attention mechanism is the heart of the Transformer network and it has different types such as *self- and cross-attention*

Self Attention



Value matrix: Indicates how each key match the given query

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



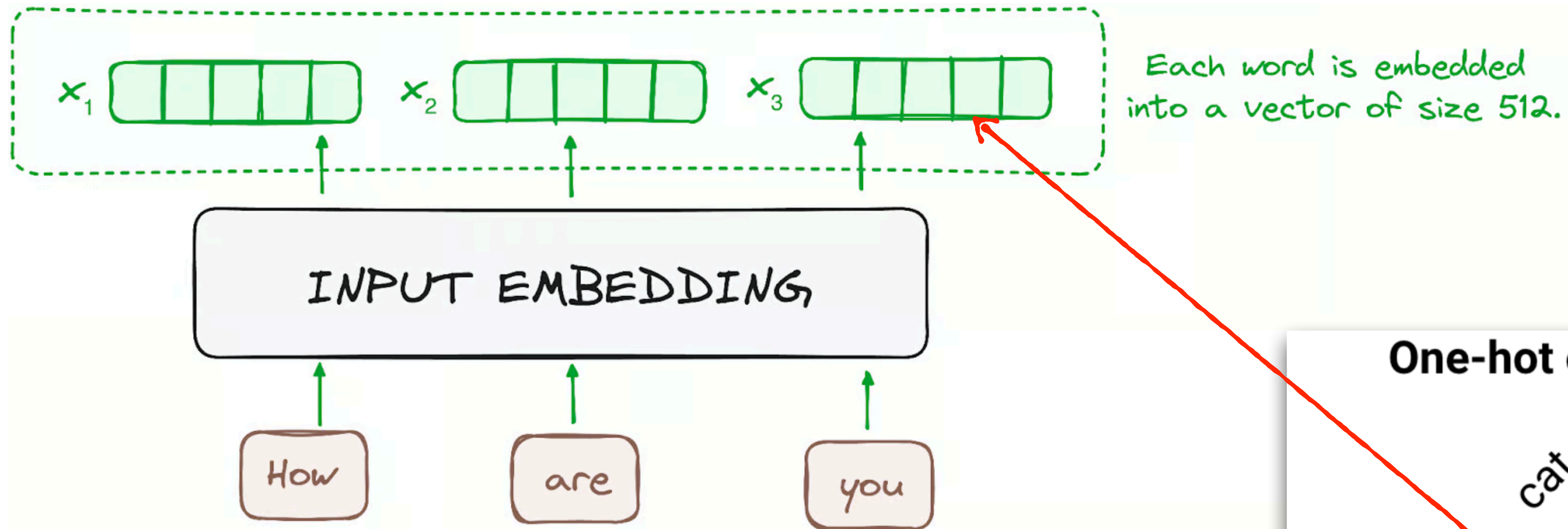
Each word is scaled relative to its importance for the underlying task

Attention output has the same dimension of the input which allows for stacking more Transformer layers

Lets see how it is computed in details

Self Attention

1- As a first step we need to encode the in words into numbers: *Word embedding*



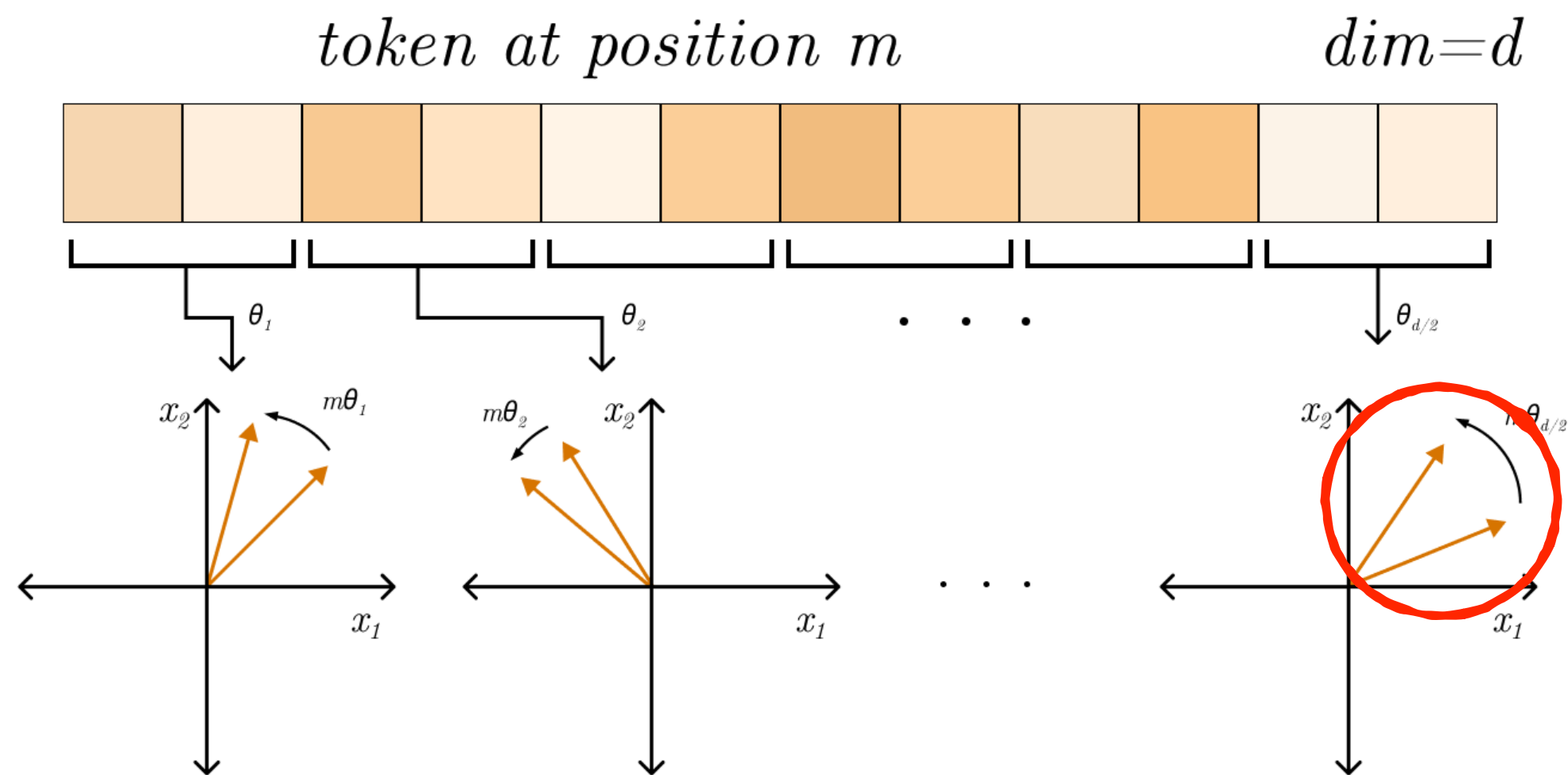
One-hot encoding

	cat	mat	on	sat	the
the =>	0	0	0	0	1
cat =>	1	0	0	0	0
sat =>	0	0	0	1	0
...					

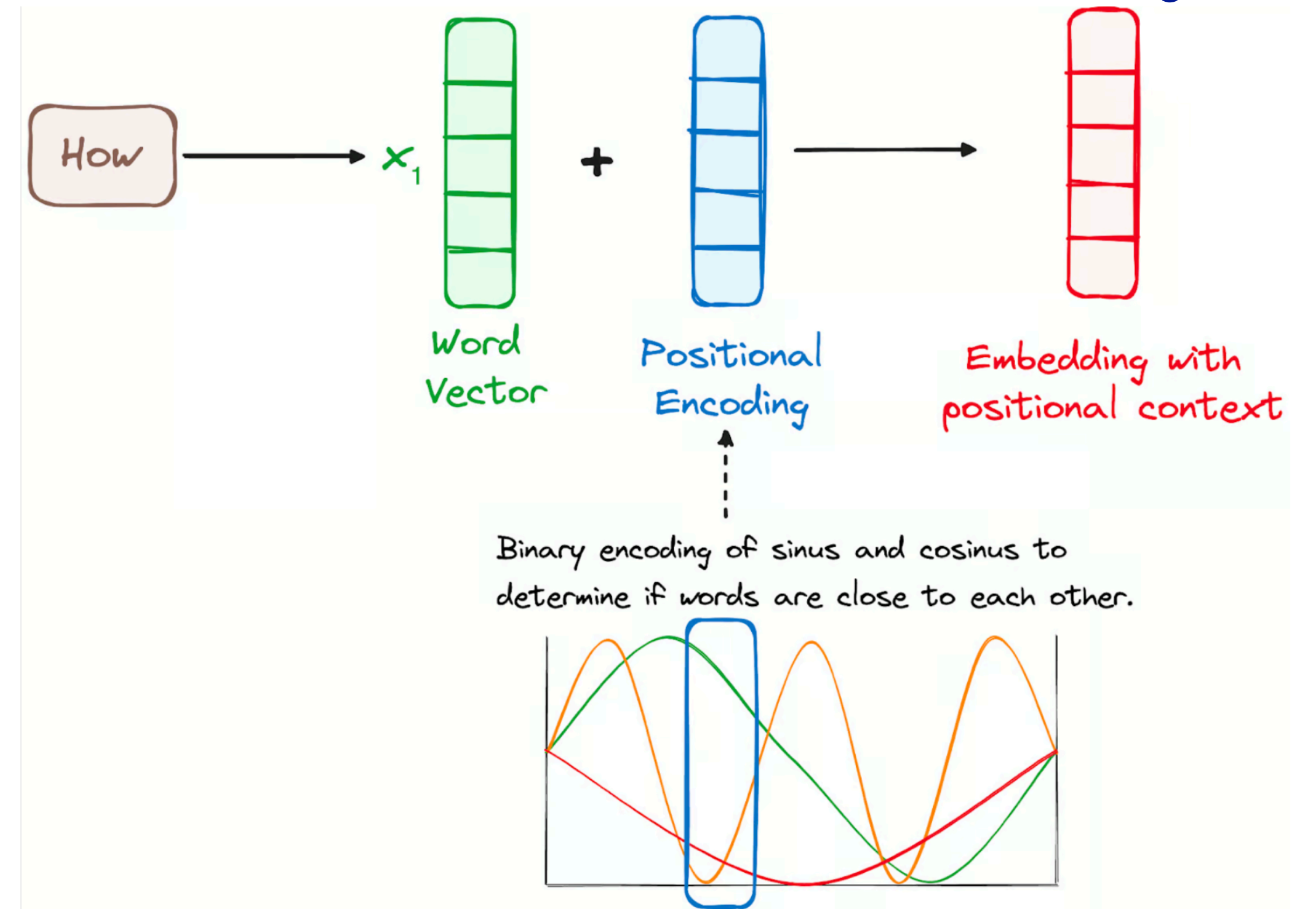
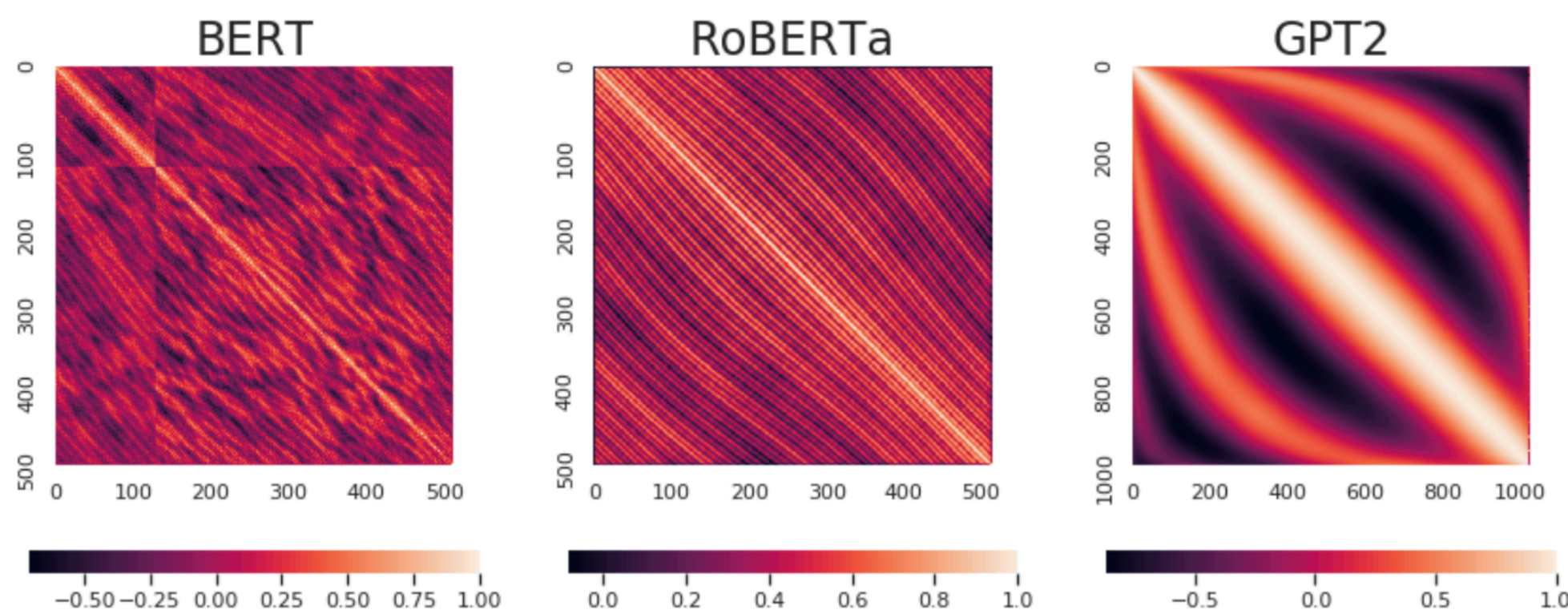
One hot encoding is not favored due to memory load

Self Attention

2- Position of the word in the sentence is important to understand the whole meaning



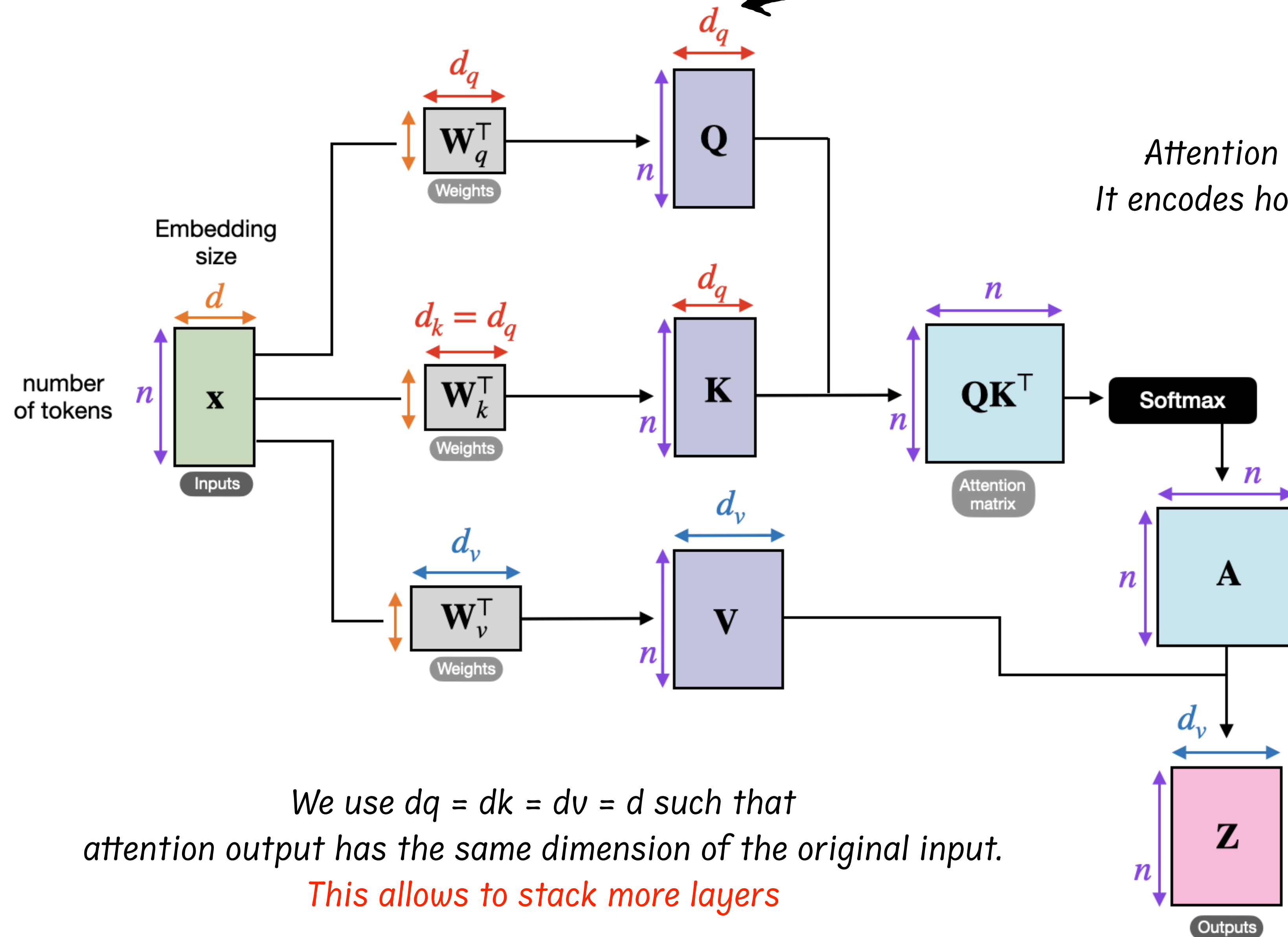
Angle between the word vectors defines its position in the sentence



Position embedding of 1000 words by different networks, BERT, ROBERTa and GPT. Dark color means long distance while more bright means the words are close together

Self Attention

Hidden dimension is arbitrary .
It is the core idea behind the LLM fine-tuning



Attention score has the dimension of words * words
It encodes how important each word to the whole sequence

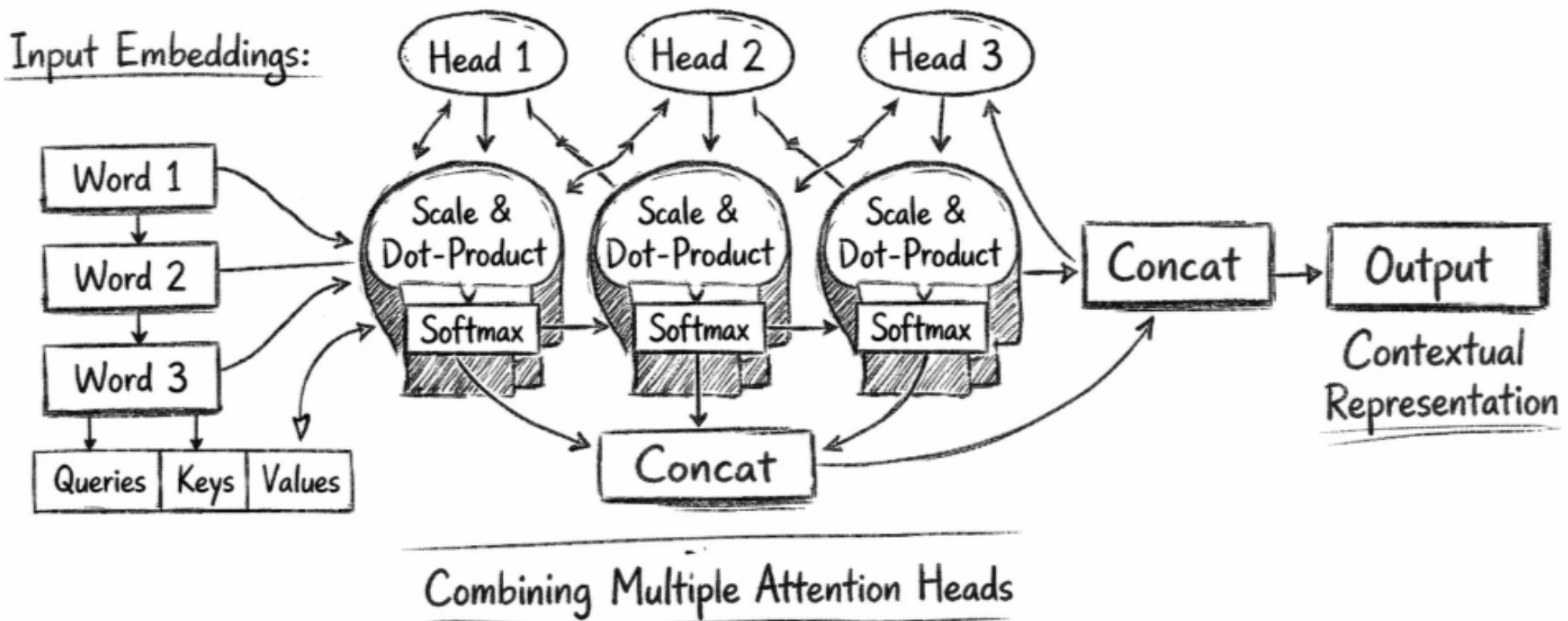
We use $d_q = d_k = d_v = d$ such that
attention output has the same dimension of the original input.
This allows to stack more layers

Multi-head

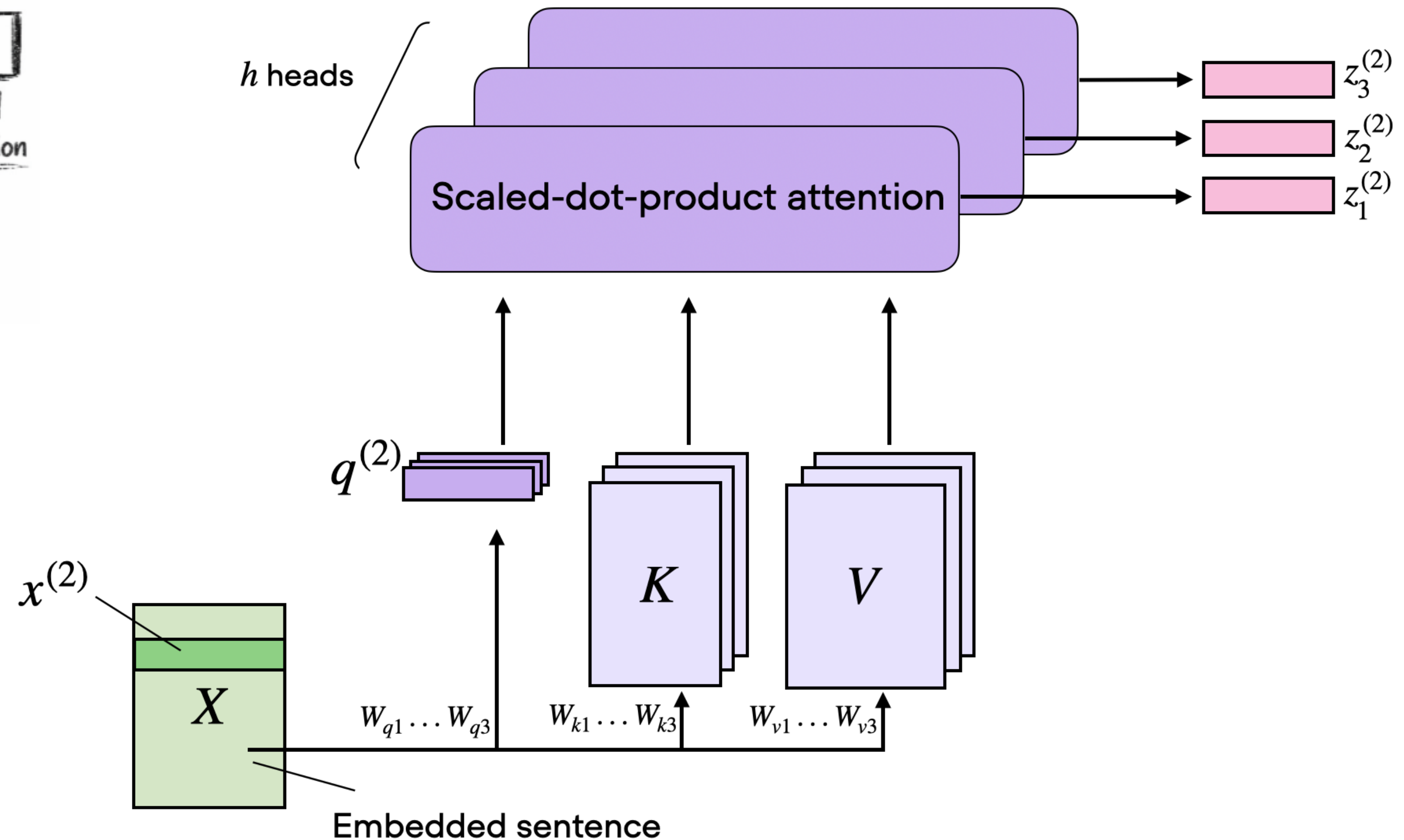
Multi-head attention

each attention head focus to capture different relationship between the input tokens.

Multi-Head Attention

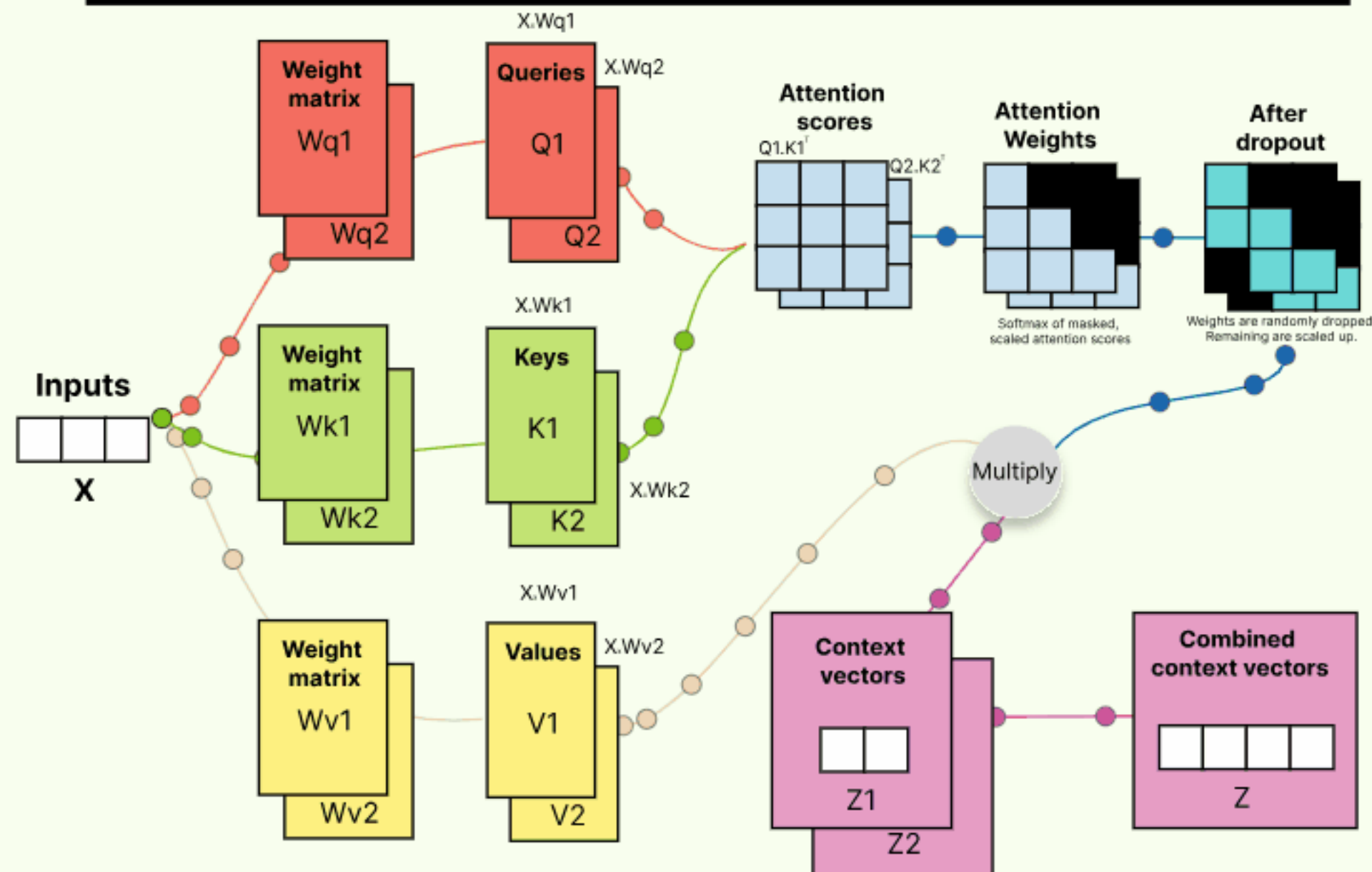


Each attention head has the same output dimension

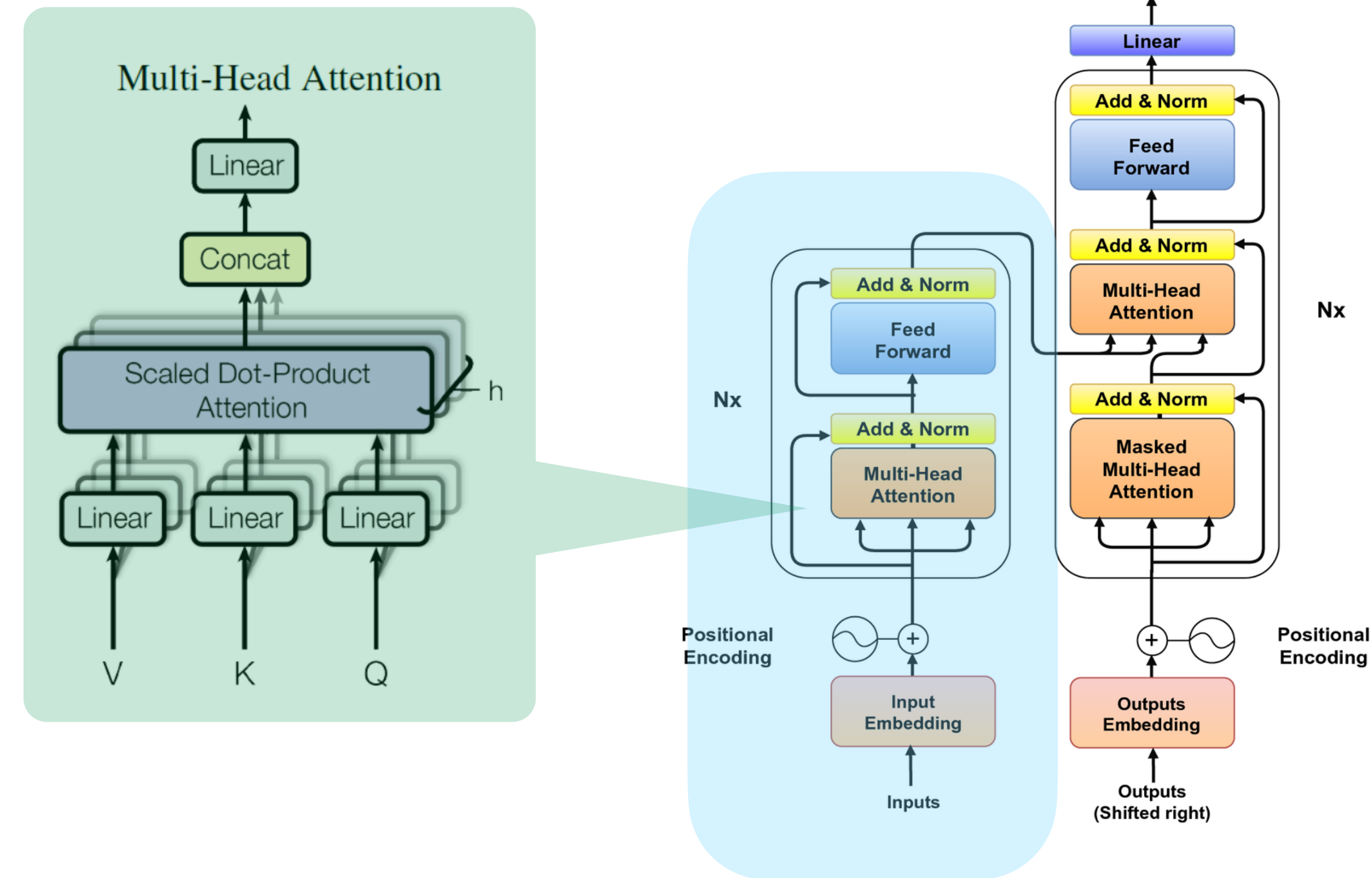
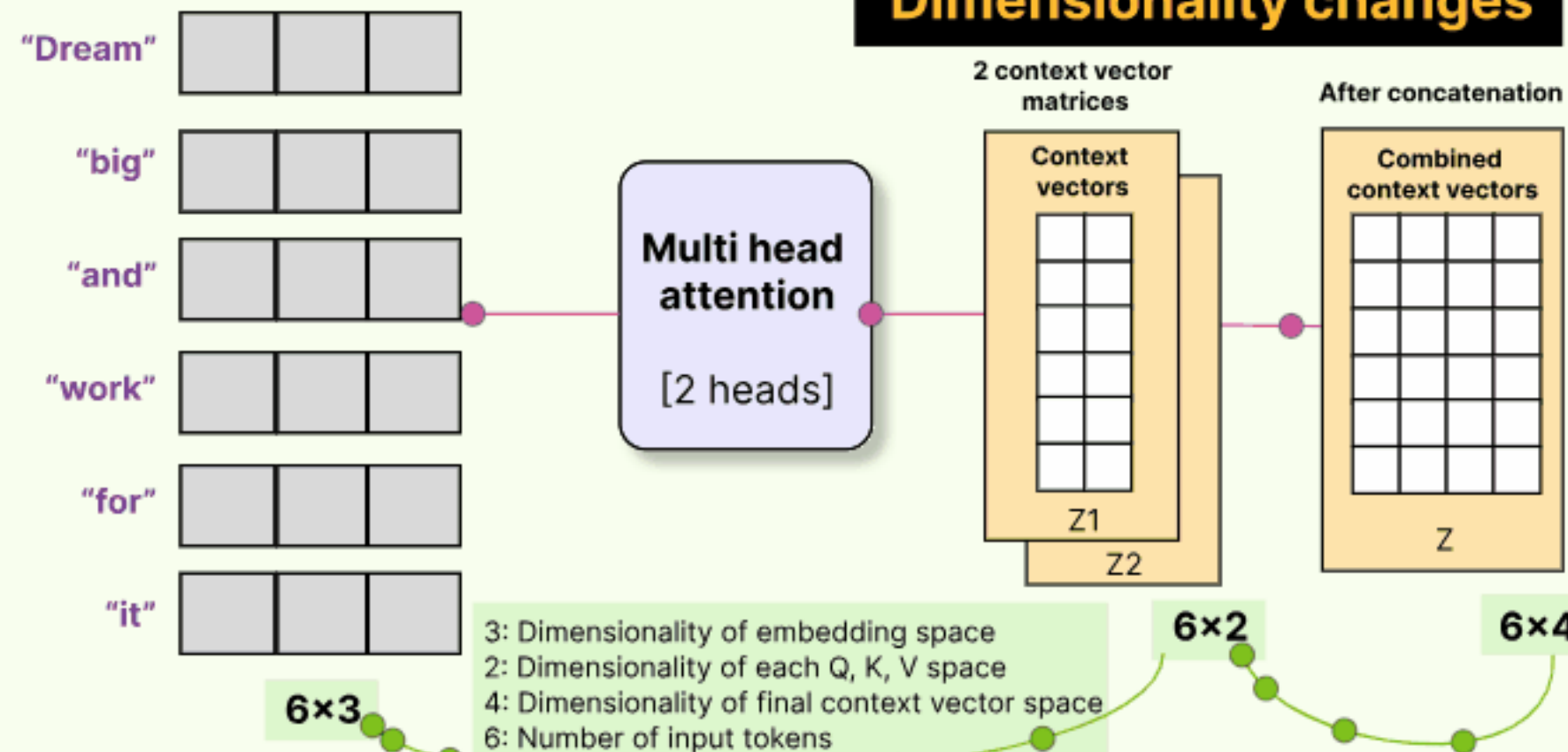


All together

Multi-head attention visualized



Dimensionality changes



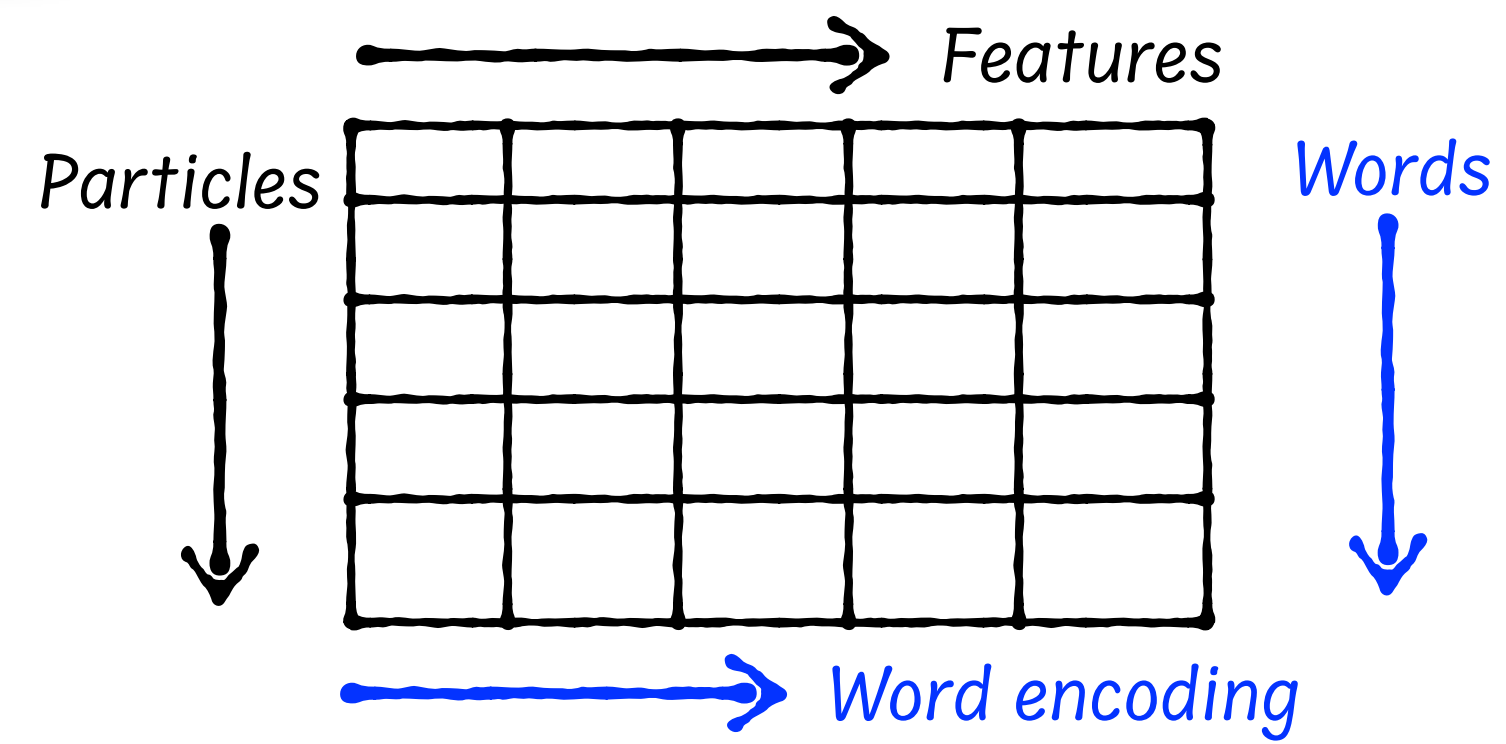
$$\text{concat} \left(Z_1^{i \times j}, Z_2^{i \times j}, \dots, Z_n^{i \times j} \right) W^{(n \cdot j \times j)}$$

To restore the original dimension of the input, multiply by additional learnable matrix

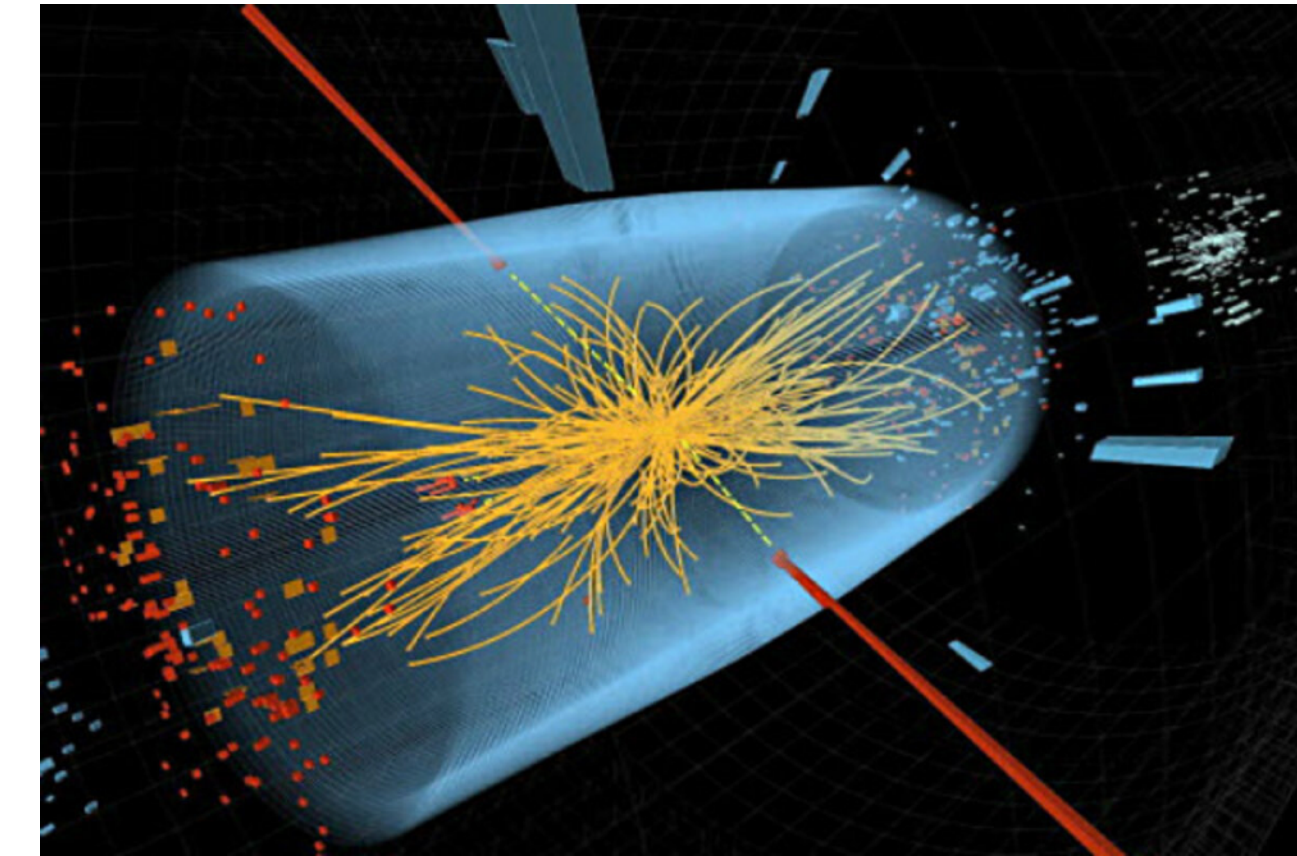
Particle clouds

Transformer for particle physics

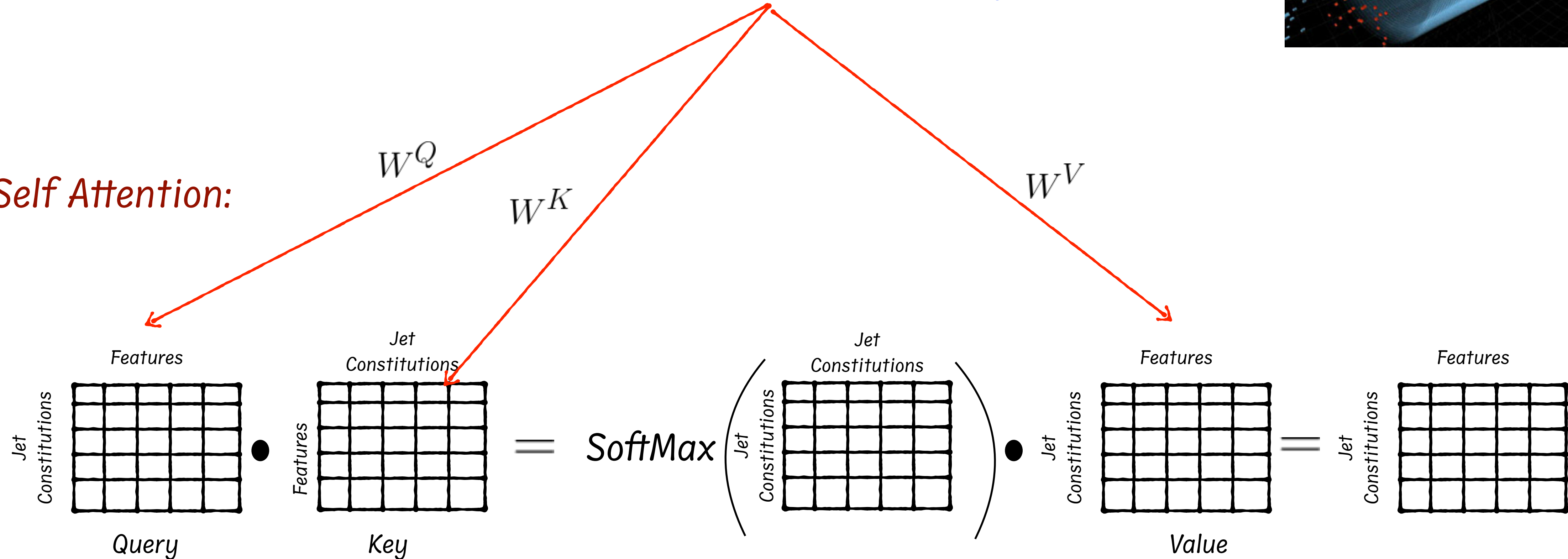
Input data can be structured as a fixed size **unordered** grid



Particles can be presented as words in sentence

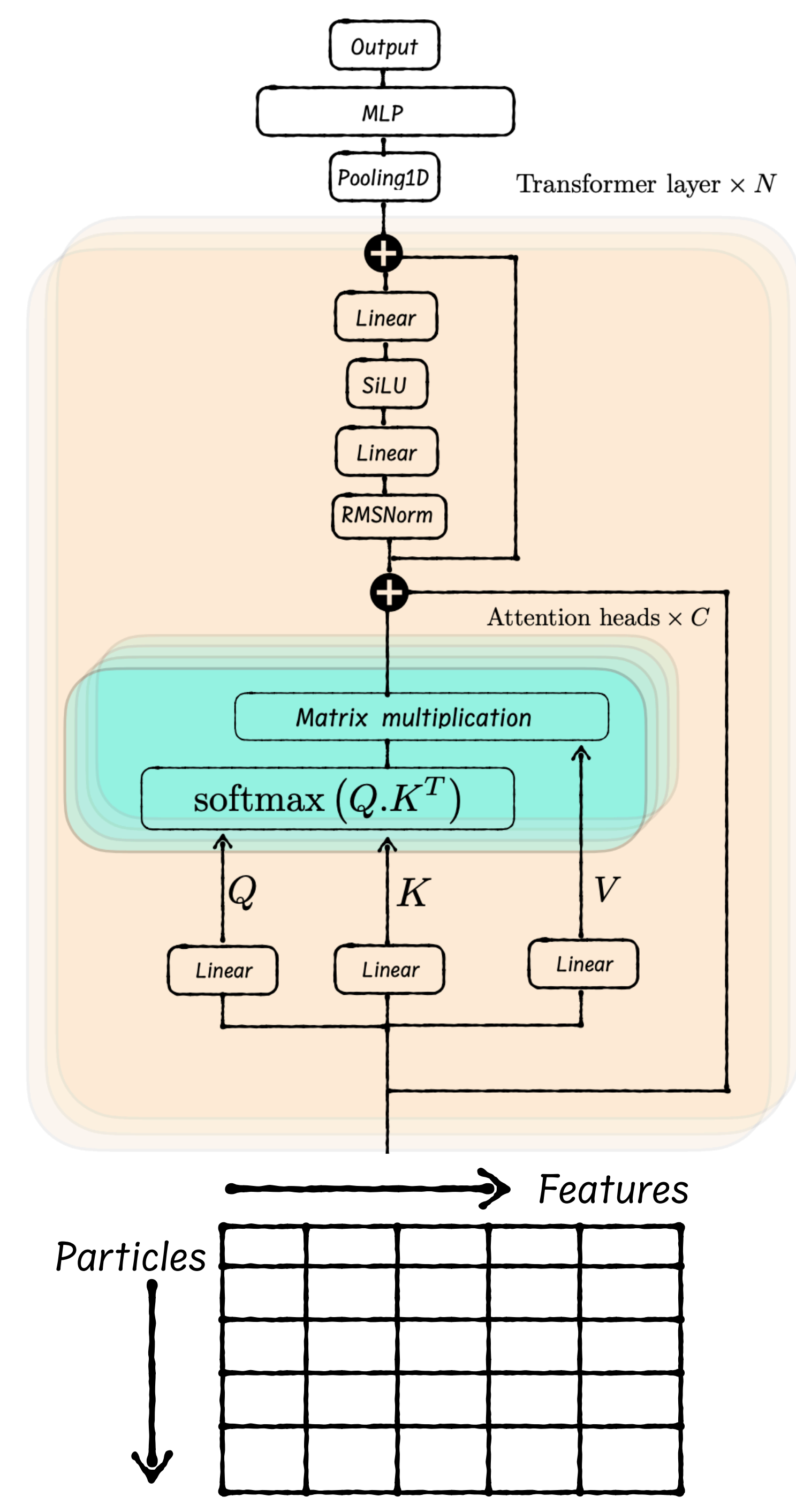
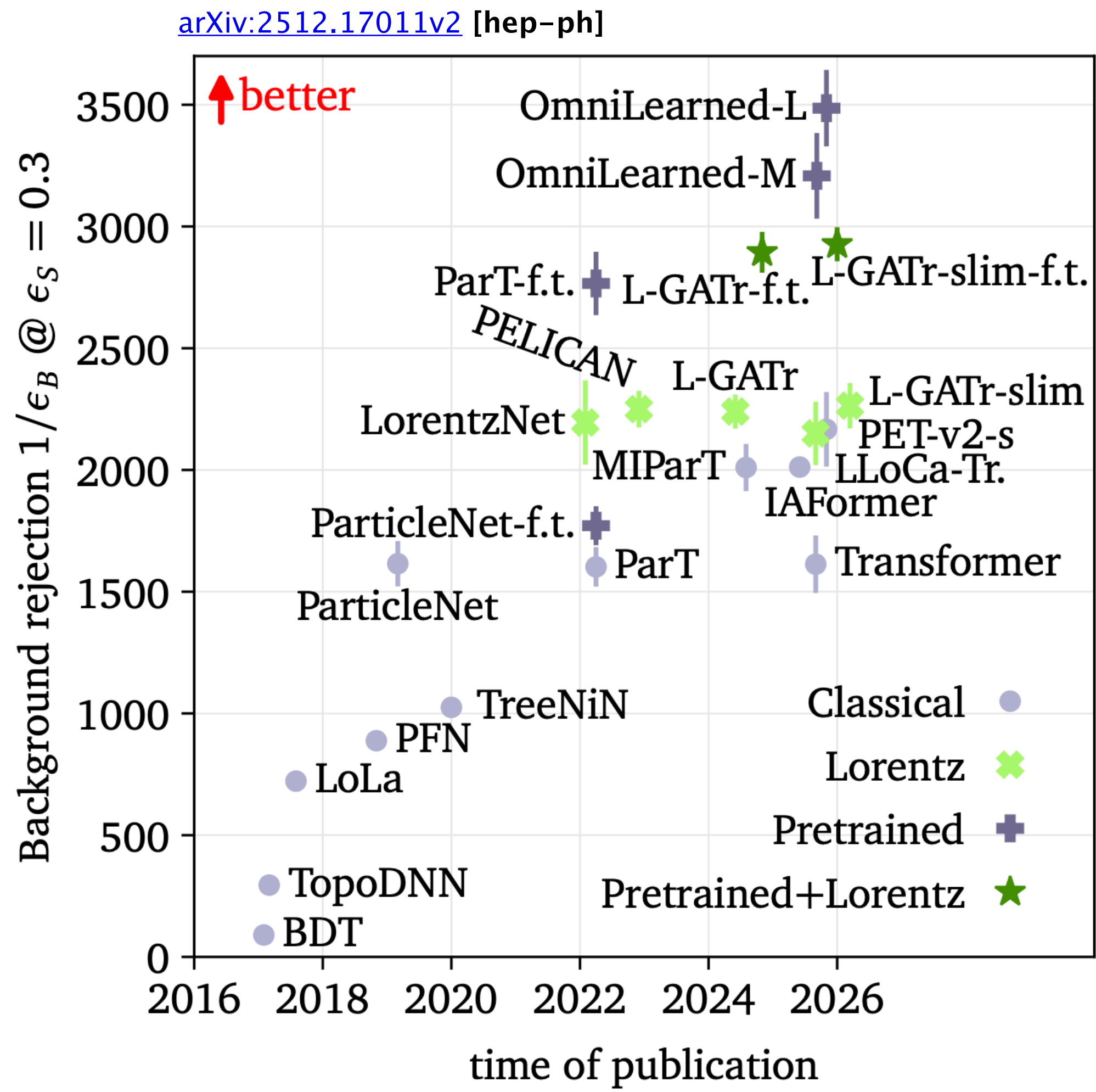


Self Attention:



Transformer for particle physics

Current transformer based models for jet tagging



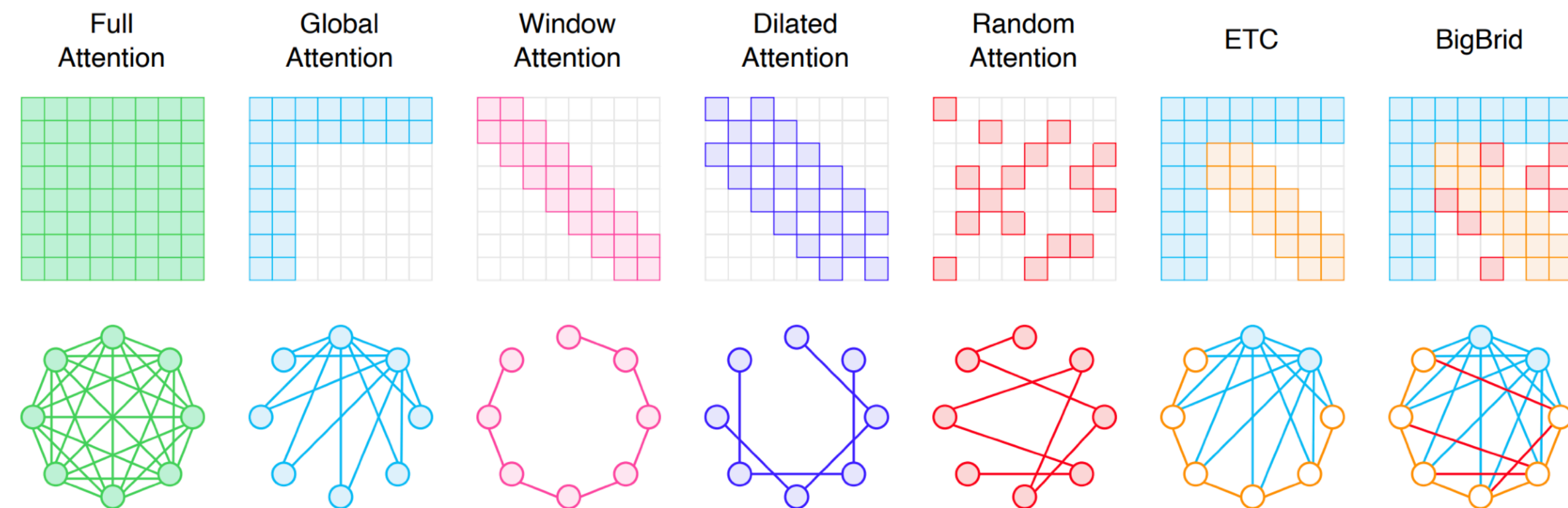
Sparse attention

Sparse attention reduces the computational overhead by introducing a sparse pattern in the attention matrix.

Arxiv:1902.09914

○ Fixed pattern :

Fixed-pattern sparse attention restricts attention to a predefined subset of token connections, reducing computational and memory costs compared to dense attention and improving efficiency for large inputs.



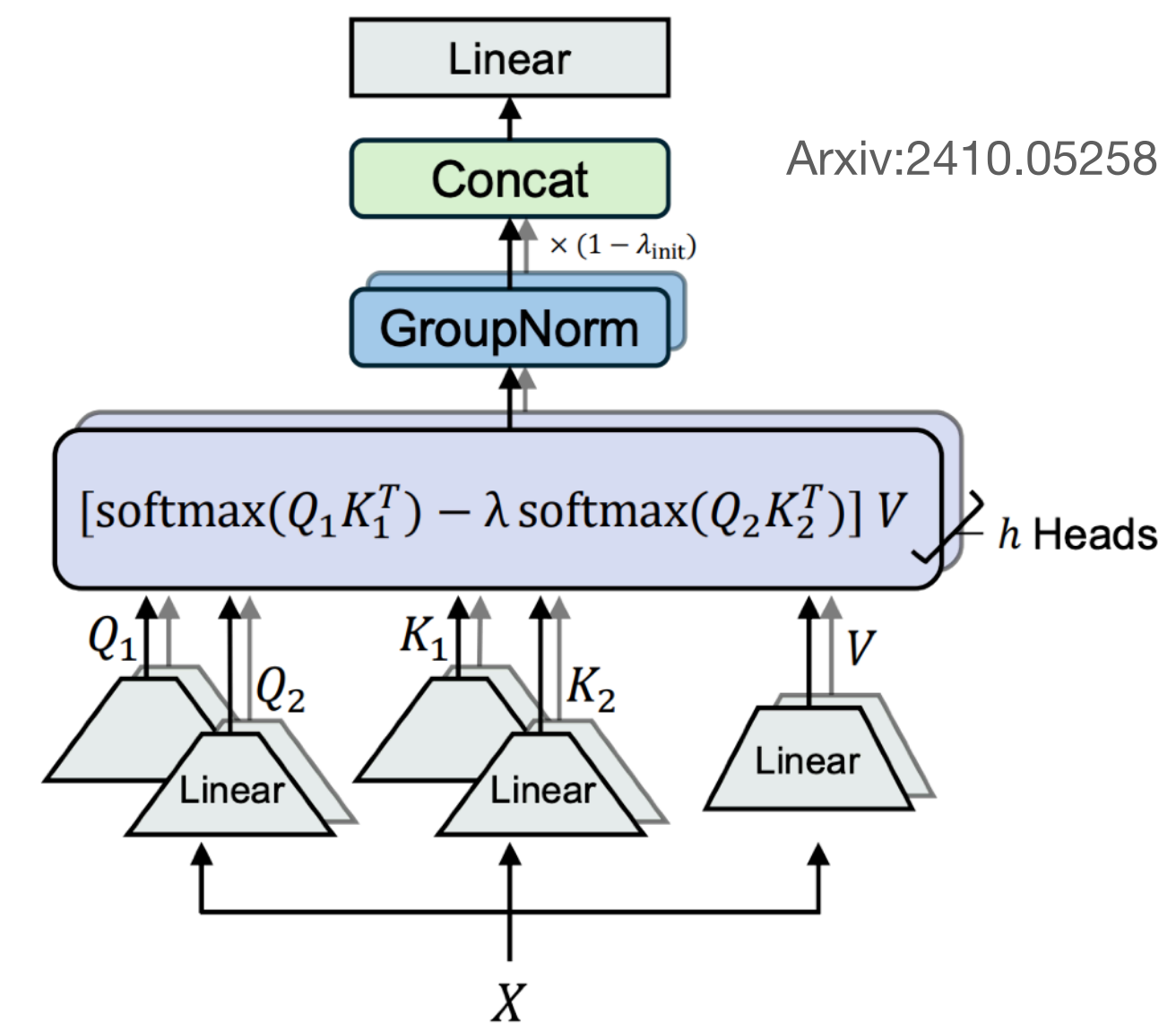
○ Dynamic pattern :

Dynamic sparse attention is an **adaptive mechanism** where the model learns which token pairs to attend to, reducing computation while focusing on the most relevant connections.

$$\text{Attention} = [\text{softmax}(Q_1 \cdot K_1^T) - \beta \text{softmax}(Q_2 \cdot K_2^T)] \cdot V$$

Beta is a learnable vector shared across all attention heads

The difference suppresses tokens that receive less attention.



IAFormer

Two input datasets:

Pairwise interaction matrix:

We replaced the $(Q.K^T)$ with the interaction matrix which has the same dimension

- $(p_{T_a} + p_{T_b})/p_{T_j}$: sum of pair transverse momenta normalized by jet p_T
- $(E_a + E_b)/E_j$: sum of pair energies normalized by jet energy
- $\Delta = \sqrt{(\eta_a - \eta_b)^2 + (\phi_a - \phi_b)^2}$: angular distance between particles
- $k_T = \min(p_{T_a}, p_{T_b}) \cdot \Delta$: transverse momentum scale
- $z = \min(p_{T_a}, p_{T_b})/p_{T_a} + p_{T_b}$: momentum sharing fraction
- $m^2 = (E_a + E_b)^2 - |\mathbf{p}_a + \mathbf{p}_b|^2$: invariant mass squared of the pair

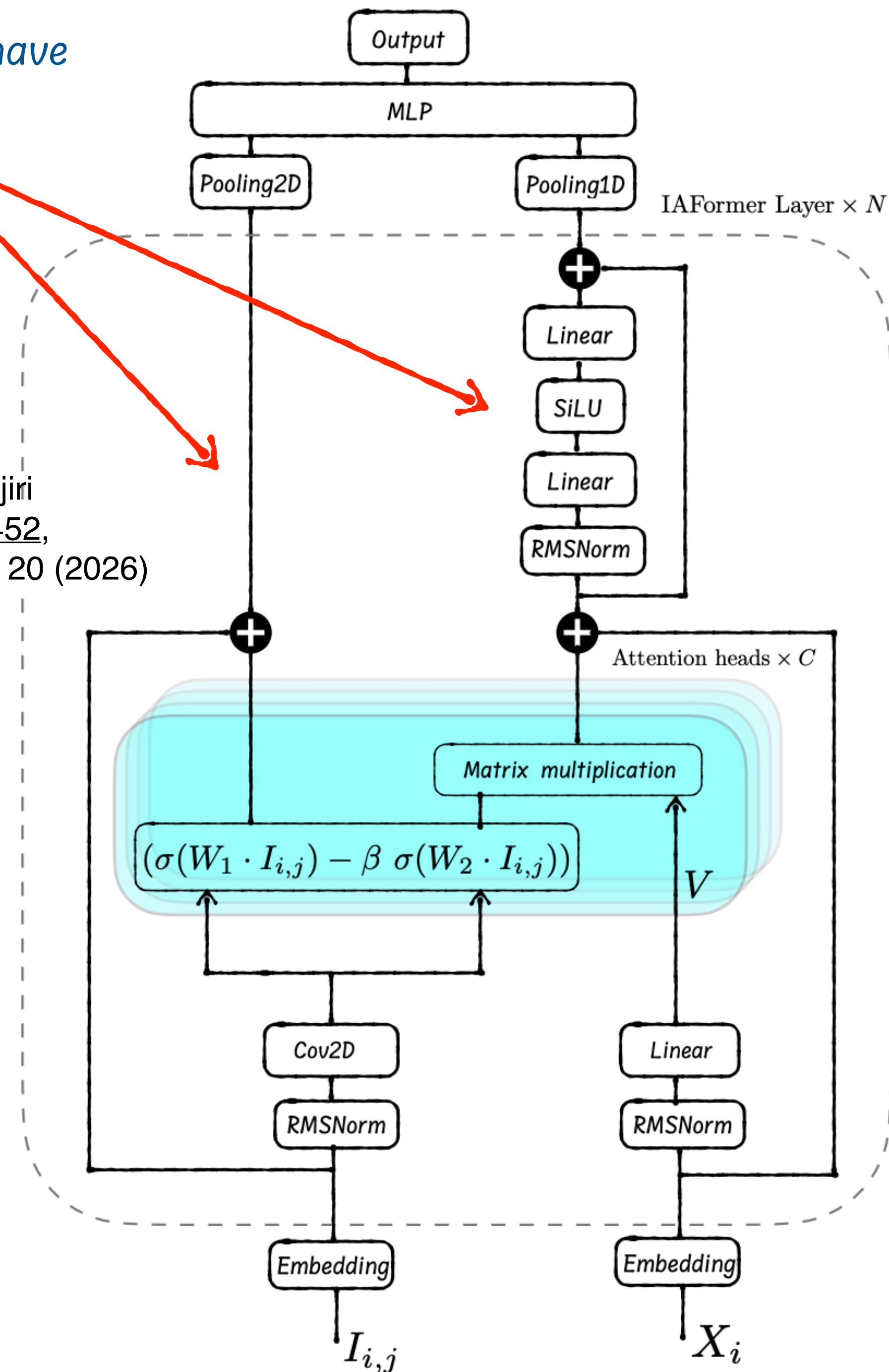
Momentum based features for each particle:

- $P_4 = (p_x, p_y, p_z, E)$: particle 4-momentum
- $\Delta\eta = \eta - \eta_{\text{jet}}$: pseudorapidity difference
- $\Delta\phi = \phi - \phi_{\text{jet}}$: azimuthal angle difference
- $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$: angular distance from jet axis
- $\log(p_T)$: transverse momentum (GeV)
- $\log(E)$: energy (GeV)
- $\log\left(\frac{p_T}{p_{T_{\text{jet}}}}\right)$: normalized p_T (GeV)
- $\log\left(\frac{E}{E_{\text{jet}}}\right)$: normalized energy (GeV)

These features have been used as the V matrix

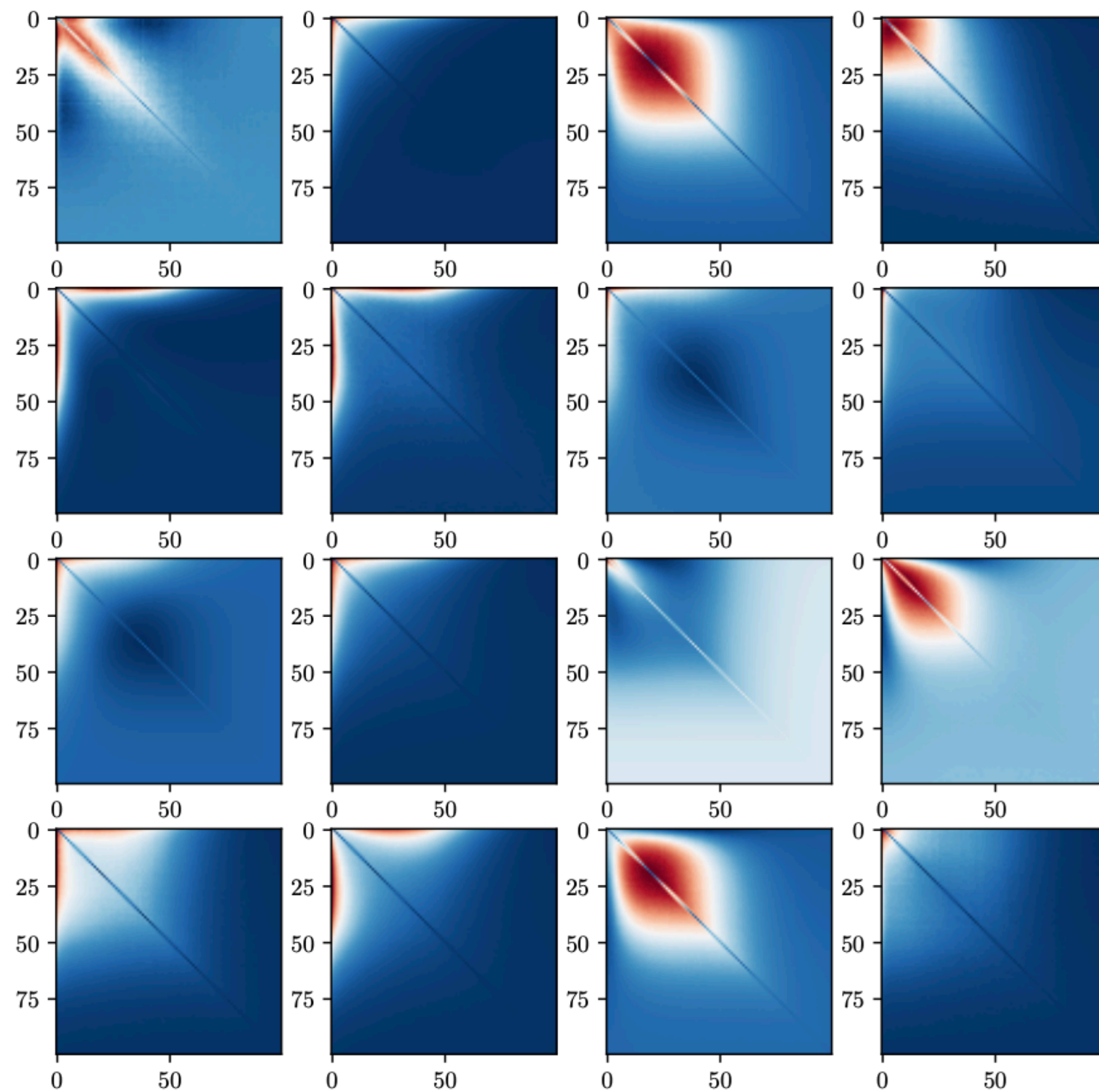
IAFormer layers have two outputs

A.H, Nojiri
2401.00452,
SciPost Phys. 20 (2026)

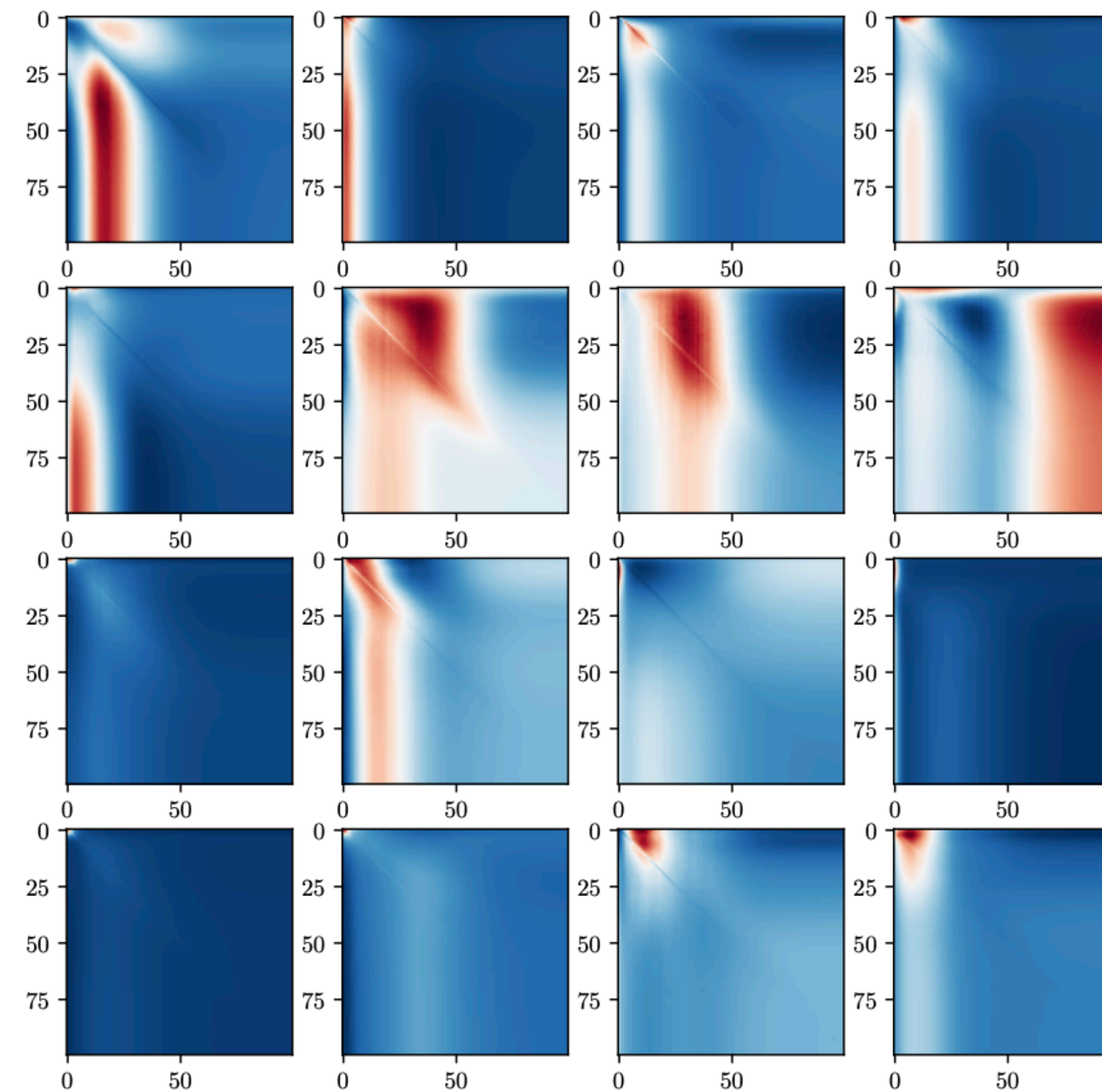


Validation on Top tagging

Attention Maps - IAFormer



Attention Maps - Plain Transformer



*Attention maps for Pt ordered particles. IAFormer focuses its attention on few particle tokens
Plain Transformer spread its attention among all particles in the dataset*

Natural Language Processing (NLP)

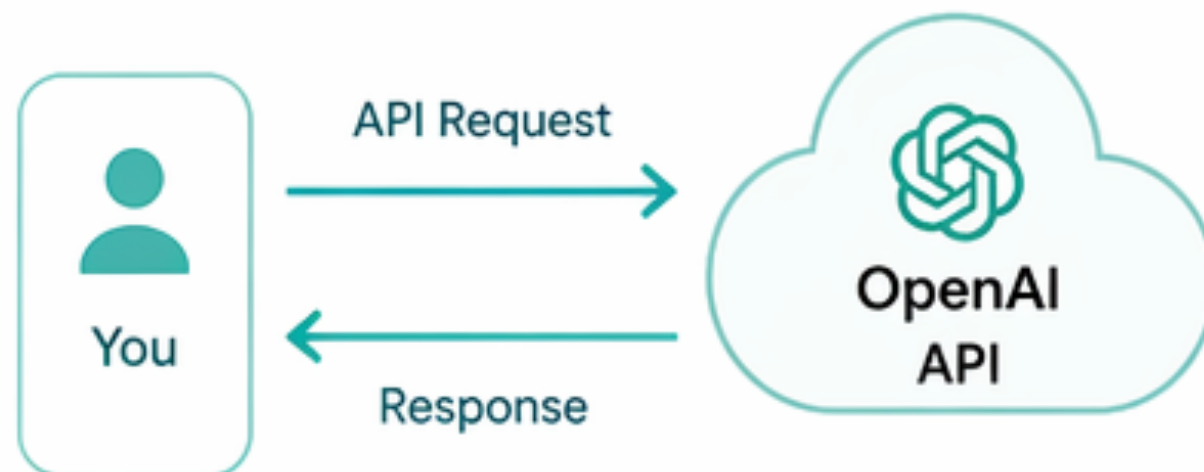
Three ways to carry out LLM inference

1



Use via OpenAI's API

Access powerful models in the cloud through a simple API.



★ Best for

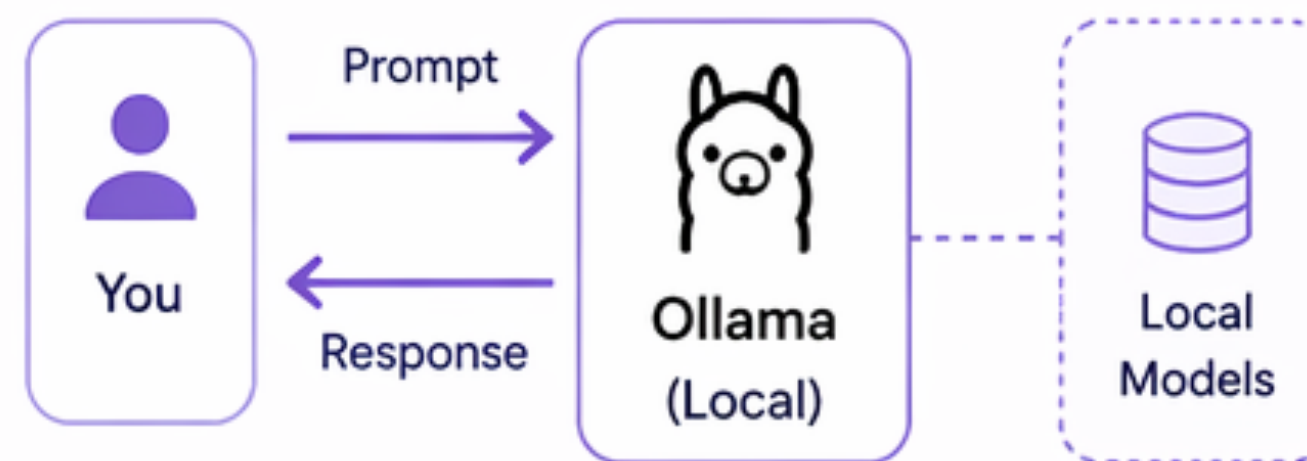
Quick start, ease of use, and powerful cloud models

2



Run locally with Ollama

Run LLMs locally on your machine with a simple CLI.



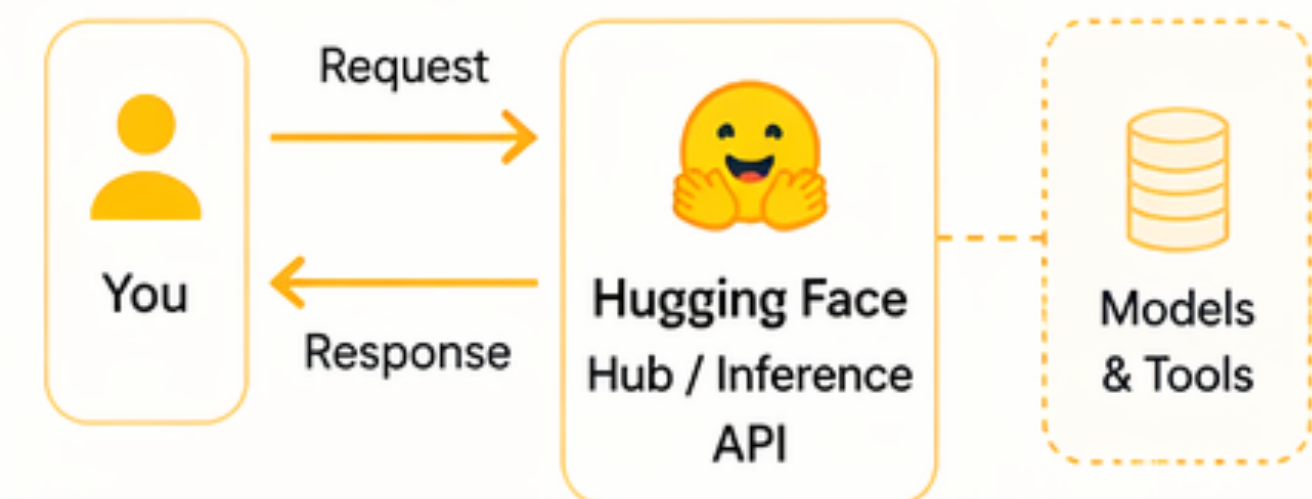
★ Best for

Privacy, offline use, and local development



Use via Hugging Face

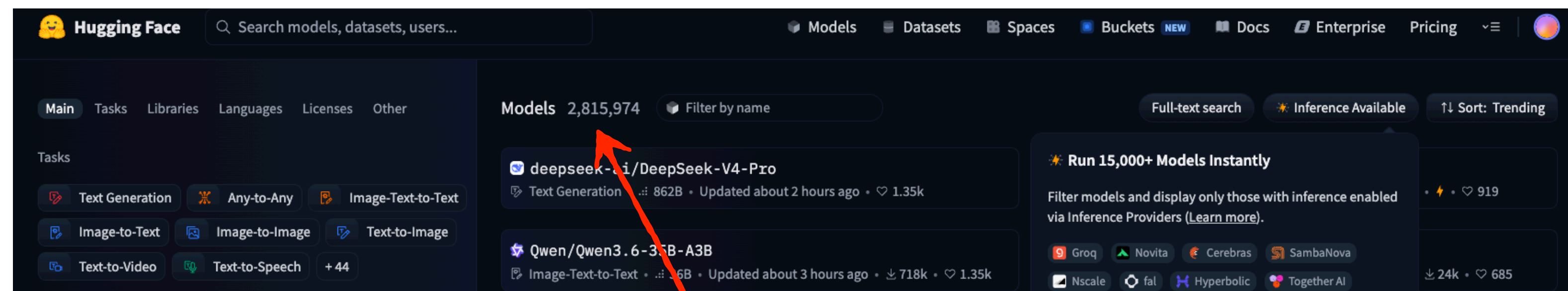
Access thousands of models, endpoints, and tools in one place.



★ Best for

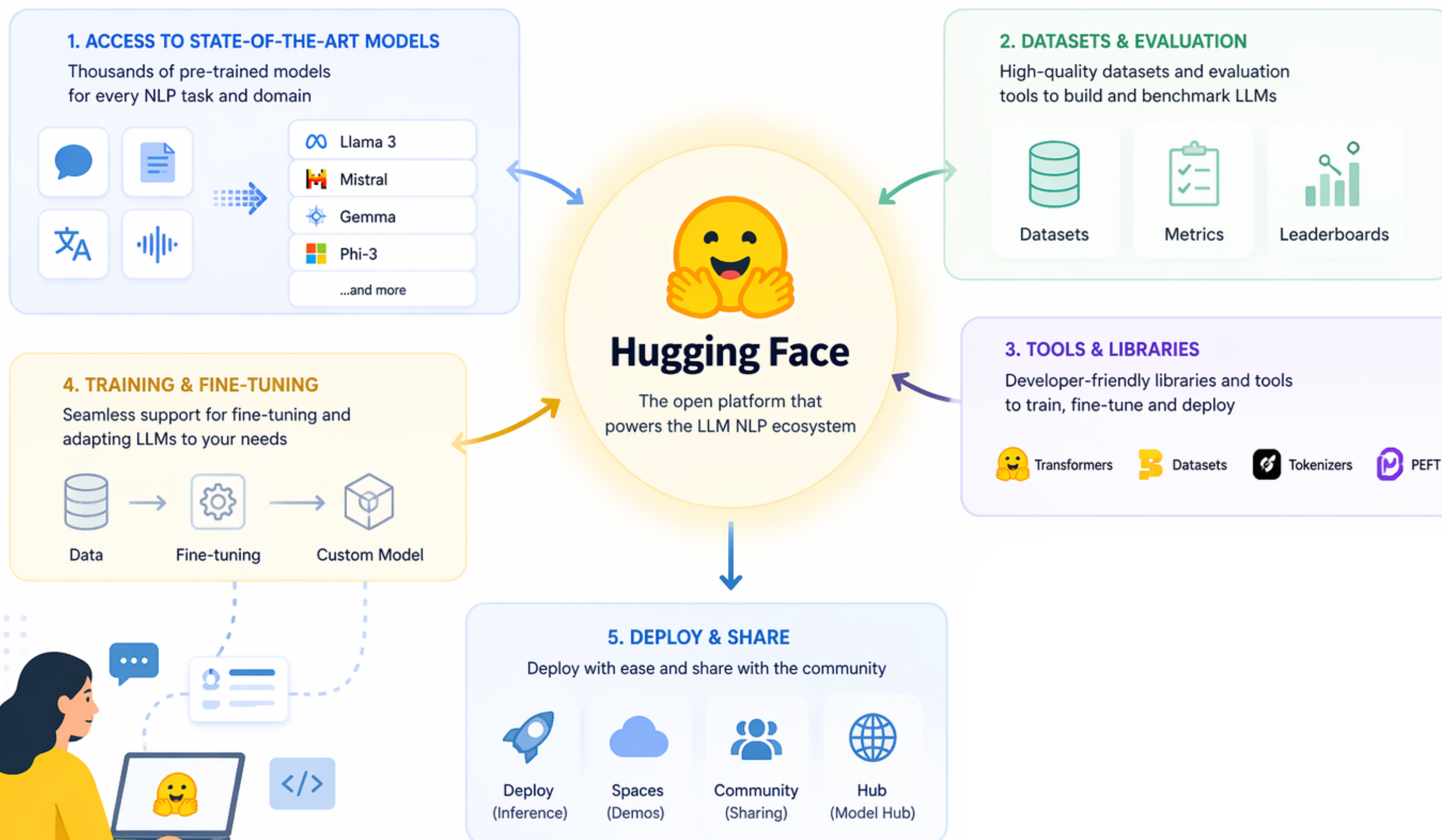
Exploring models, open-source access, and the ML community ecosystem

LLM for NLP



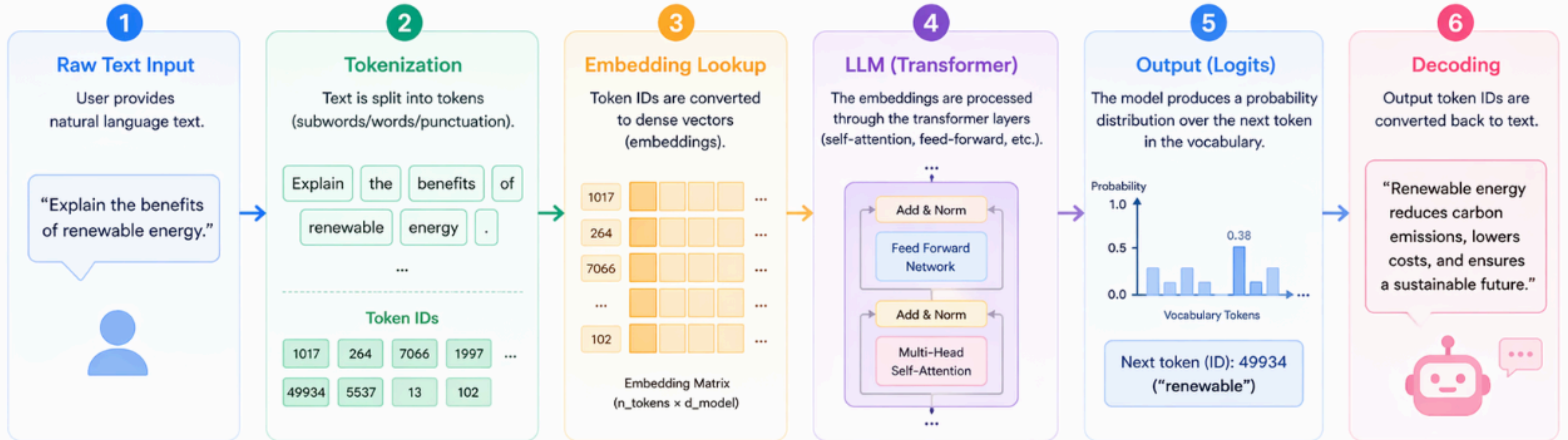
The Role of Hugging Face in LLM NLP

Empowering the NLP Lifecycle from Models to Real-World Applications



HuggingFace currently has ~ 3M pretrained LLM

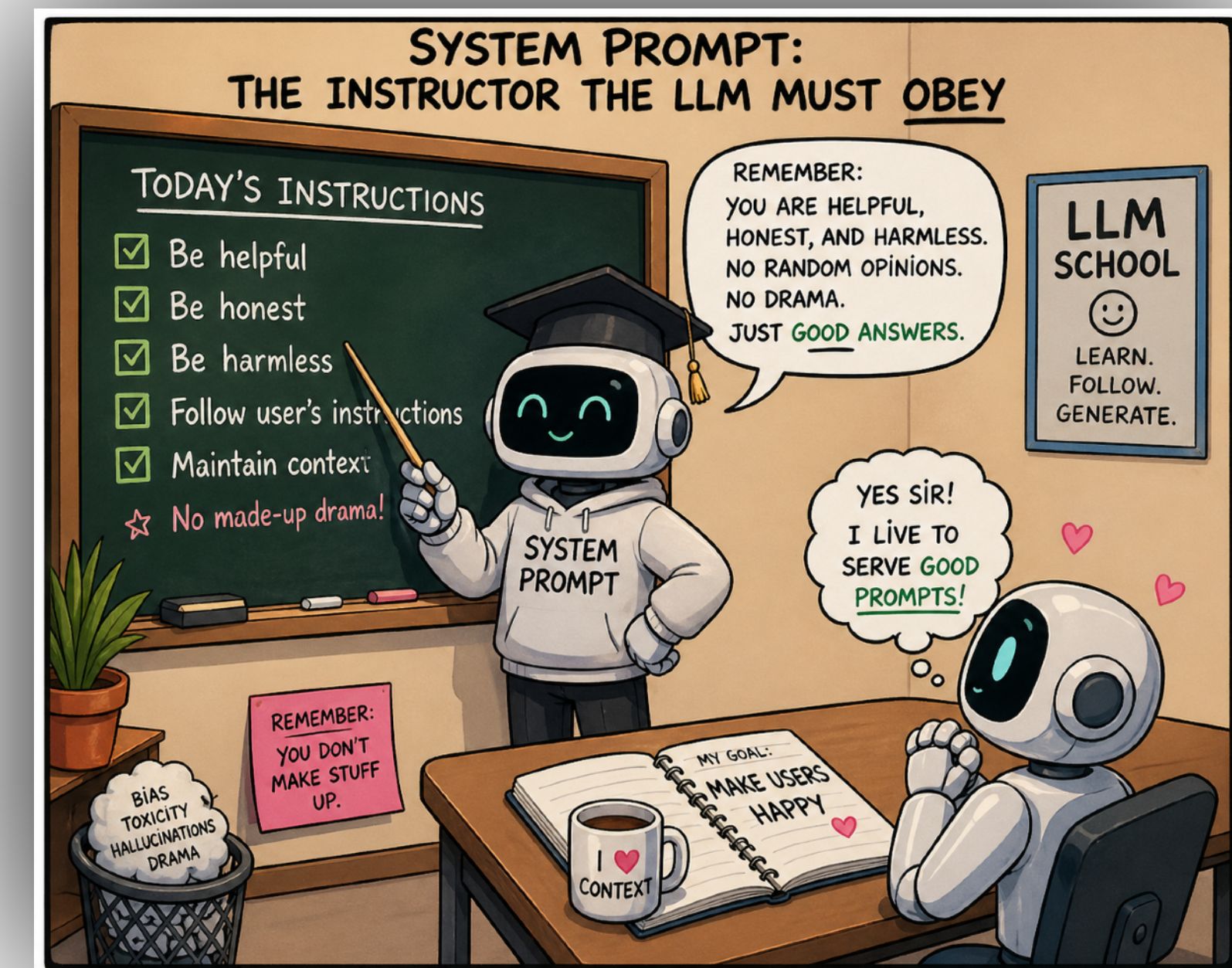
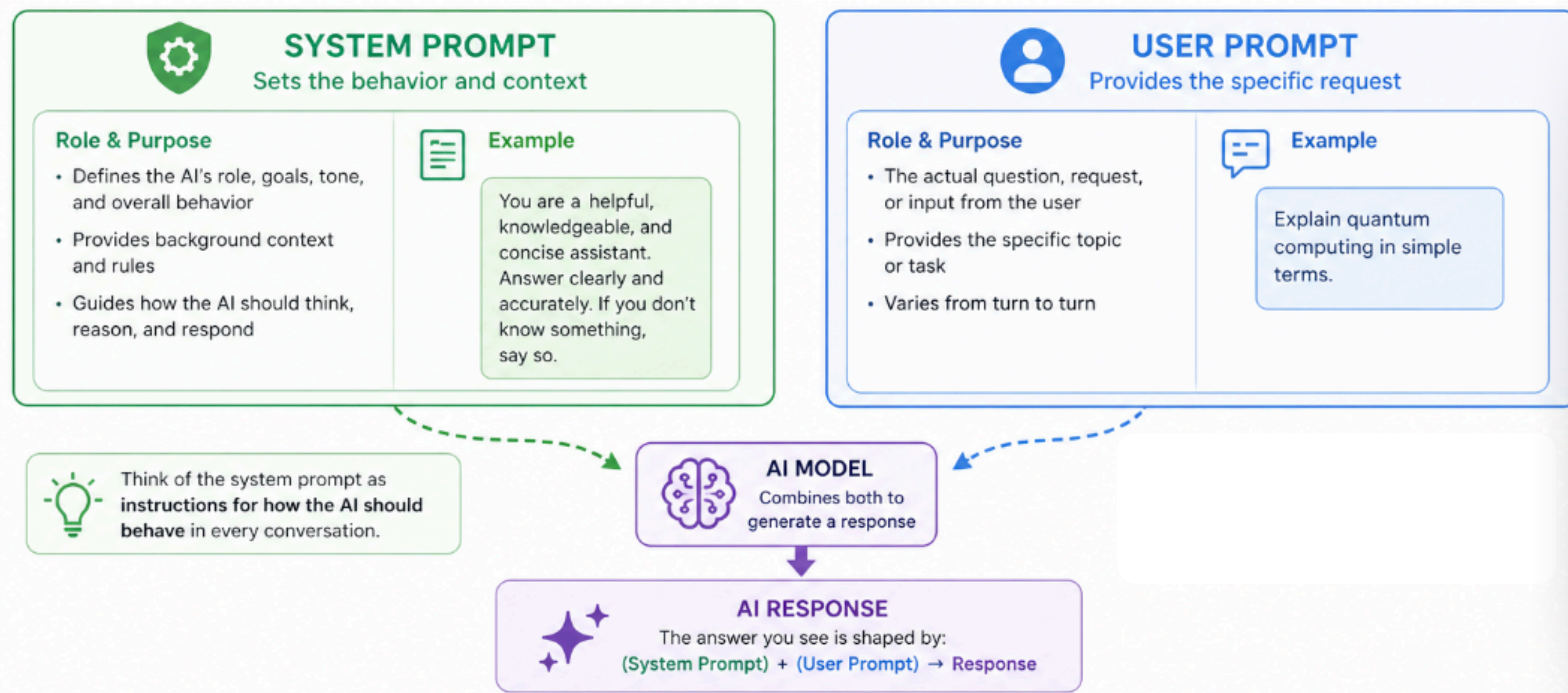
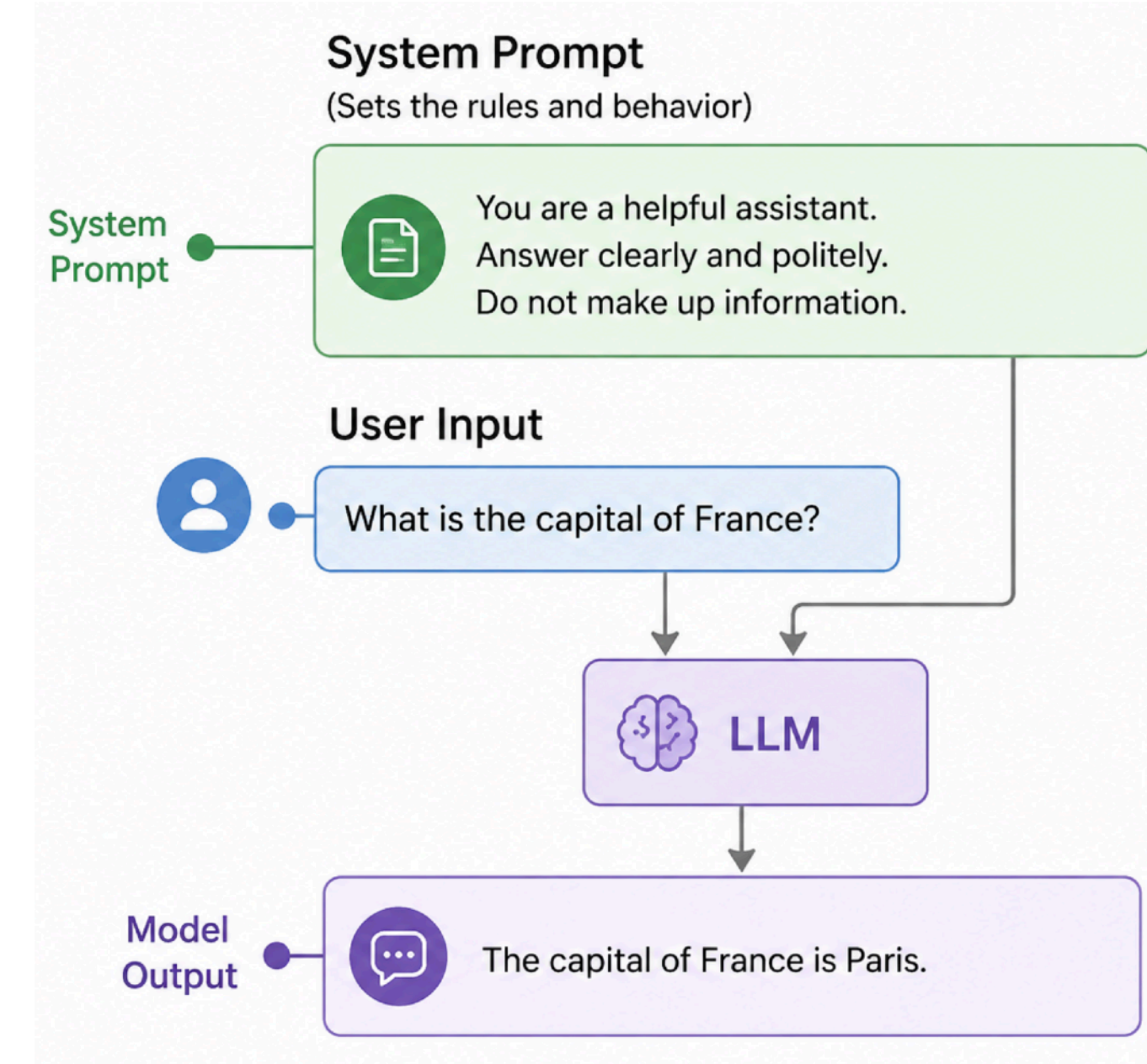
In the following we focus only on HuggingFace, But how the inference pipeline looks like ?



Prompt engineering

Prompt engineering controls the LLM response in a more structured way

system prompt is a set of instructions that defines how an AI should behave, including its role, tone, rules, and overall context for responding.



2000



LINUS TORVALDS
Creator of Linux

“Talk is cheap.
Show me the **code**.”



 WHEN THIS WAS SAID

In the early 2000s, during open source debates, Linus Torvalds emphasized that **real value** comes from **working software** and contributions, not just talk or ideas.

2026



AI
The New Builder

“Code is cheap.
Show me the **talk**
(**Prompt**).”

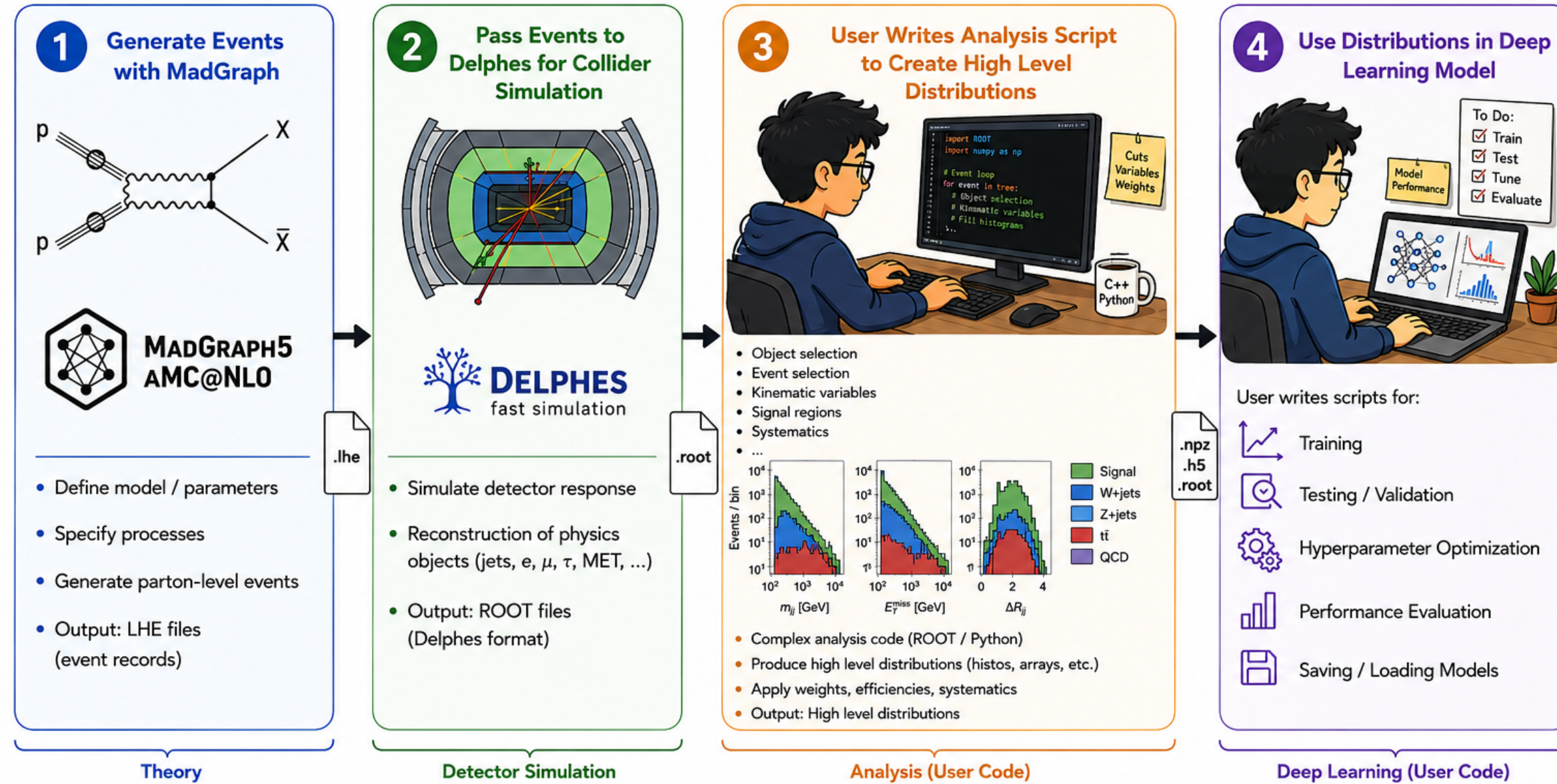


 WHEN THIS IS SAID

In the age of AI, code can be generated instantly. The real challenge is in **asking the right questions**, providing the **right context**, and designing the **right prompts**.

Collider Analysis Pipeline

From Theory to Deep Learning



CoLLM : AI engineering toolbox for end-to-end deep learning in collider analyses

W. Esmail^{a*}, A. Hammad^{b†} and M. Nojiri^{b,c,d‡}

^a Institut für Kernphysik, Universität Münster, Wilhelm-Klemm-Str. 9, 48149 Münster, Germany.

^b Theory Center, IPNS, KEK, 1-1 Oho, Tsukuba, Ibaraki 305-0801, Japan.

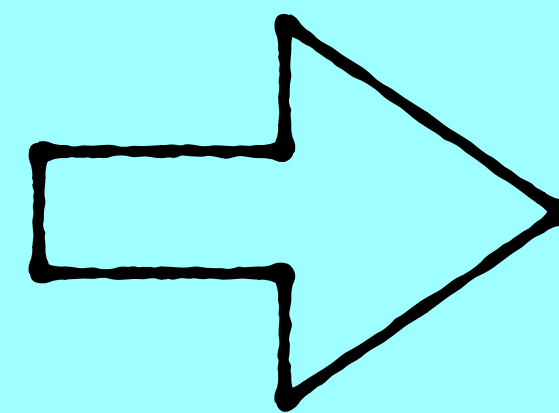
^c The Graduate University of Advanced Studies (Sokendai), 1-1 Oho, Tsukuba, Japan.

^d Kavli IPMU (WPI), University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Chiba 277-8583, Japan.

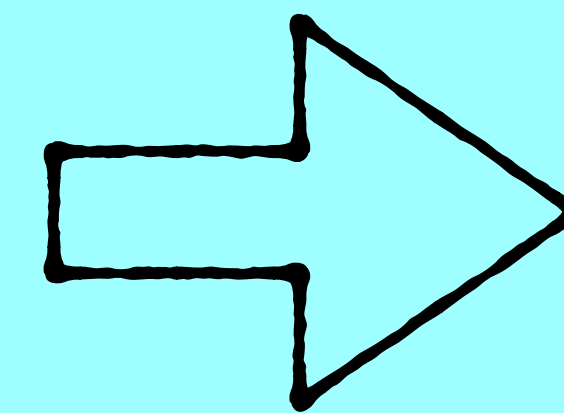
Abstract

Recent improvements in large language models have opened new opportunities for accelerating and automating scientific workflows. In parallel, modern collider analyses are becoming increasingly complex and demand substantial

Input raw detector files (lhco)

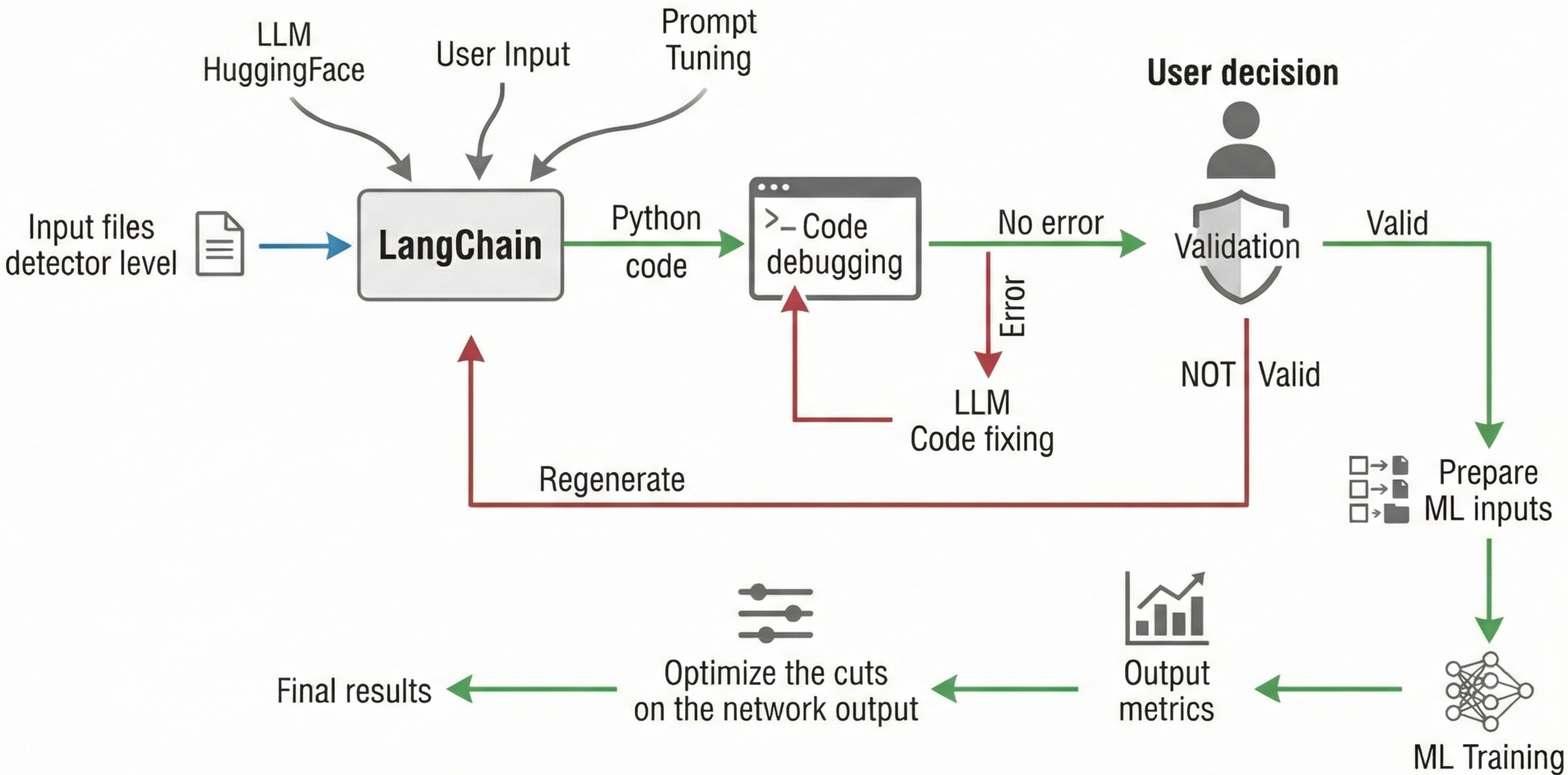


Generate the selection Analysis



Perform ML analysis

No coding & ML experience is needed, simply click the buttons.



Example of system prompt

system prompt is the instructional backbone that defines how the model should behave, reason and format its outputs.

```
You are a particle physics analysis assistant specialized in analyzing LHCO (.lhco) files produced by fast detector simulations (e.g., Delphes).

=====
HARD REQUIREMENTS (must always be satisfied):
=====

1. Always assume the input data is an LHCO file.
2. Always generate runnable Python 3 code when analysis is requested.
3. Always include the LHCO parser provided below.
4. Prioritize physics correctness over style or verbosity.
5. Use only: standard library, math, numpy (and matplotlib only if explicitly needed).
```

Guidelines for code generation

```
=====
CODE GUIDELINES:
=====

STRUCTURE:
1. Parsing      - Read LHCO file into event list
2. Selection    - Filter objects and events based on cuts
3. Reconstruction - Combine objects to form physics candidates
4. Output       - Print results, histograms, or cutflow and save the generated histograms with each histogram in a sepa

BEST PRACTICES:
- Keep code minimal, explicit, and readable
- Validate particle counts before pairing or selection
- Handle edge cases: missing objects, empty events, malformed lines
- Skip events gracefully when required objects are not found
- CRITICAL: When looping over events to fill histograms or compute observables,
  always re-extract particle collections (e.g., jets, leptons, taus) for EACH event
  inside the loop. Never rely on variables defined in a previous loop or outside
  the current event iteration.
- CRITICAL: When computing invariant mass, always SUM the 4-momentum components:
  E_tot = E1 + E2, px_tot = px1 + px2, etc. NEVER use differences.
  The formula M^2 = E^2 - px^2 - py^2 - pz^2 uses the TOTAL (summed) components.
- CRITICAL: MET type code is 6, NOT 5.
- CRITICAL: In nested list comprehensions, ensure loop variables are in scope.
  WRONG: [f(x, event) for x in xs] - event undefined if iterating over xs
  RIGHT: [f(x, e) for e in events for x in get_xs(e)]
- CRITICAL: Cutflow print statements must match the actual cuts applied.

CUTFLOW REQUIREMENTS:
- Track number of events before and after each cut
- Print the event count after each selection step
- Always print final number of events passing all cuts
```

Particle definition in LHCO format

```
=====
LHCO FILE FORMAT SPECIFICATION:
=====

OBJECT LINE FORMAT:
  index  type  eta  phi  pt  jmass  ntrk  btag  had/em

COLUMN DEFINITIONS:
  index  : Object index within the event (0 marks new event header)
  type   : Particle type code (see below)
  eta    : Pseudorapidity
  phi    : Azimuthal angle (radians)
  pt     : Transverse momentum (GeV)
  jmass  : Jet mass (GeV) - use only for jets
  ntrk   : Track count; sign encodes lepton charge
  btag   : b-tag flag (1.0 = b-tagged jet, 0.0 = not b-tagged)
  had/em : Hadronic-to-electromagnetic energy ratio

PARTICLE TYPE CODES:
  0 = Photon
  1 = Electron
  2 = Muon
  3 = Tau
  4 = Jet
  6 = MET (η = 0, φ = MET direction, pt = MET magnitude)
```

Naming convention

```
=====
NAMING CONVENTIONS:
=====

LEPTONS:
- "lepton" or "l" refers to BOTH electrons (type=1) AND muons (type=2)
- "l+" or "lepton+" refers to positively charged leptons (ntrk > 0)
- "l-" or "lepton-" refers to negatively charged leptons (ntrk < 0)
- "electron" refers specifically to type=1
- "muon" refers specifically to type=2

JETS:
- "jet" refers to type=4 objects
- "b-jet" refers to jets with btag=1.0
- "light jet" refers to jets with btag=0.0

LEADING/SUBLEADING:
- "leading" = highest pT particle of that type in the event
- "subleading" = second-highest pT particle of that type in the event
```


CoLLM Toolbox

An integrated framework combining LLM-powered selection analysis code generation with advanced deep learning classifiers for high energy physics research.

⚠ For better visualization change your browser to the dark mode.

⚠ For LLM code generation on your laptop, we strongly recommend using the Hugging Face API. This enables inference via Hugging Face Claude. Please refer to the sidebar for detailed instructions on how to create your API key.

Selection Analysis

Deep Learning

Results

Network Architecture

Select and configure your deep learning network

Select Model Type

☒ Multi-Layer Perceptron (MLP) ☐ Graph Neural Networks (GNNs) ☐ Transformer

./run.sh —run_GUI

MLP Configuration

Number of hidden layers

3

−

+

Output Layer

2

Output Activation

None

Layer 1 Configuration

Layer Type

Dense Layer

Neurons

128

Activation

relu

Layer 2 Configuration

Layer Type

Dense Layer

Neurons

64

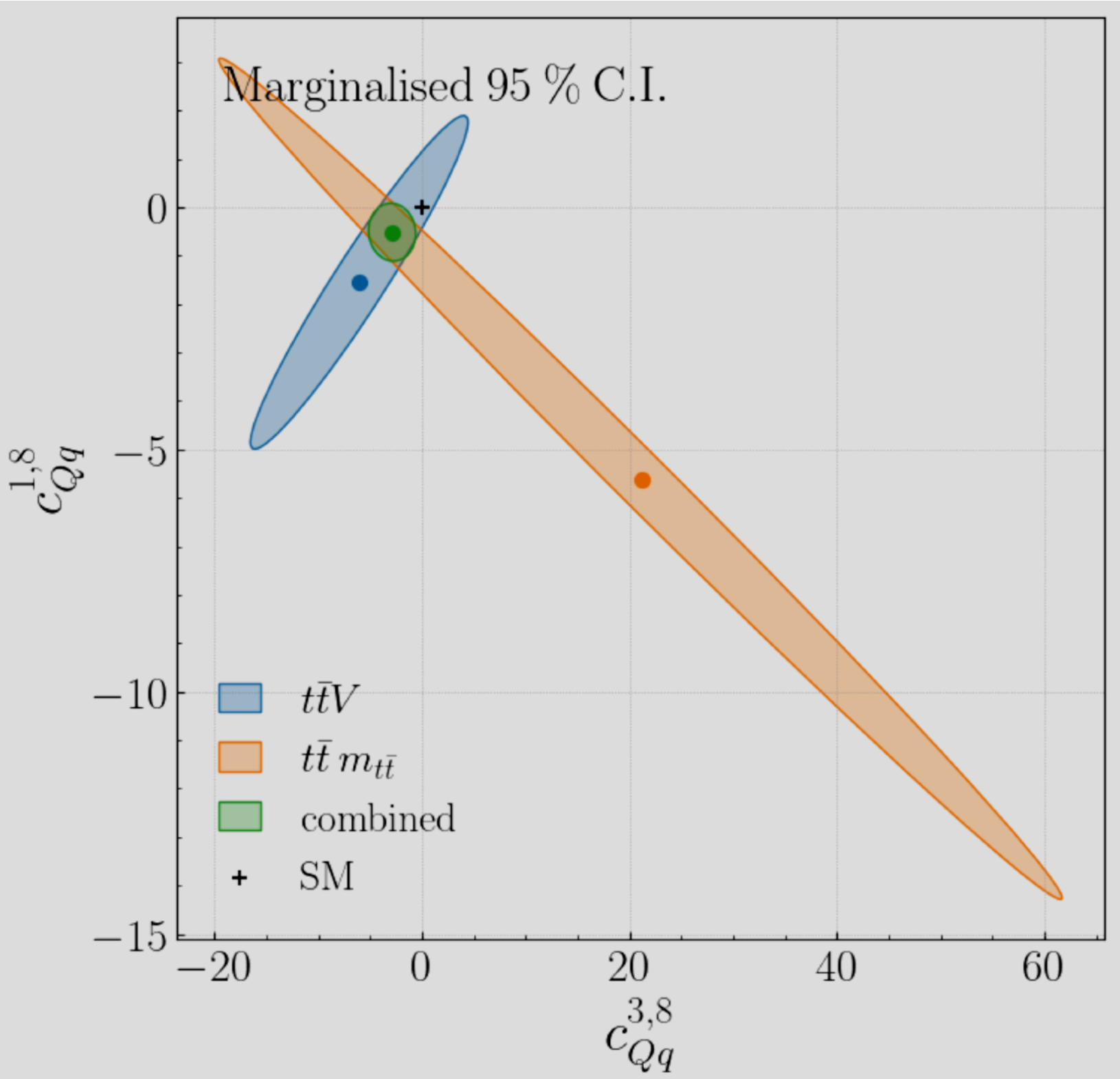
Activation

relu

Layer 3 Configuration

SMEFiT performs a fit to the Wilson coefficients of SMEFT operators by comparing the theoretical prediction to the measurements

$$\mathcal{L}(\mathbf{c}) \propto \prod_{m,n=1}^{n_{\text{dat}}} \exp \left[- \left(\mathcal{T}_m(\mathbf{c}) - \mathcal{D}_m \right) \left(\text{cov}^{-1} \right)_{mn} \left(\mathcal{T}_n(\mathbf{c}) - \mathcal{D}_n \right) \right]$$



Input run card to SMEFiT



Project description

SMEFiT is a Python package for global analyses of particle physics data in the framework of the Standard Model Effective Field Theory (SMEFT). The SMEFT represents a powerful model-independent framework to constrain, identify, and parametrize potential deviations with respect to the predictions of the Standard Model (SM). A particularly attractive feature of the SMEFT is its capability to systematically correlate deviations from the SM between different processes. The full exploitation of the SMEFT potential for indirect New Physics searches from precision measurements requires combining the information provided by the broadest possible dataset, namely carrying out extensive global analysis which is the main purpose of SMEFiT.

The SMEFiT framework has been used in the following **scientific publications**:

```
# Datasets to include
datasets:

- {name: CMS_ttZ_13TeV, order: NLO_QCD}
- {name: CMS_ttZ_13TeV_pTZ, order: NLO_QCD}
- {name: CMS_ttZ_8TeV, order: NLO_QCD}
- {name: ATLAS_ttZ_13TeV, order: NLO_QCD}
- {name: ATLAS_ttZ_13TeV_2016, order: NLO_QCD}
- {name: ATLAS_ttZ_13TeV_pTZ, order: NLO_QCD}
- {name: ATLAS_ttZ_8TeV, order: NLO_QCD}
- {name: CMS_ttW_13TeV, order: NLO_QCD}
- {name: CMS_ttW_8TeV, order: NLO_QCD}
- {name: ATLAS_ttW_13TeV, order: NLO_QCD}
- {name: ATLAS_ttW_13TeV_2016, order: NLO_QCD}
- {name: ATLAS_ttW_8TeV, order: NLO_QCD}

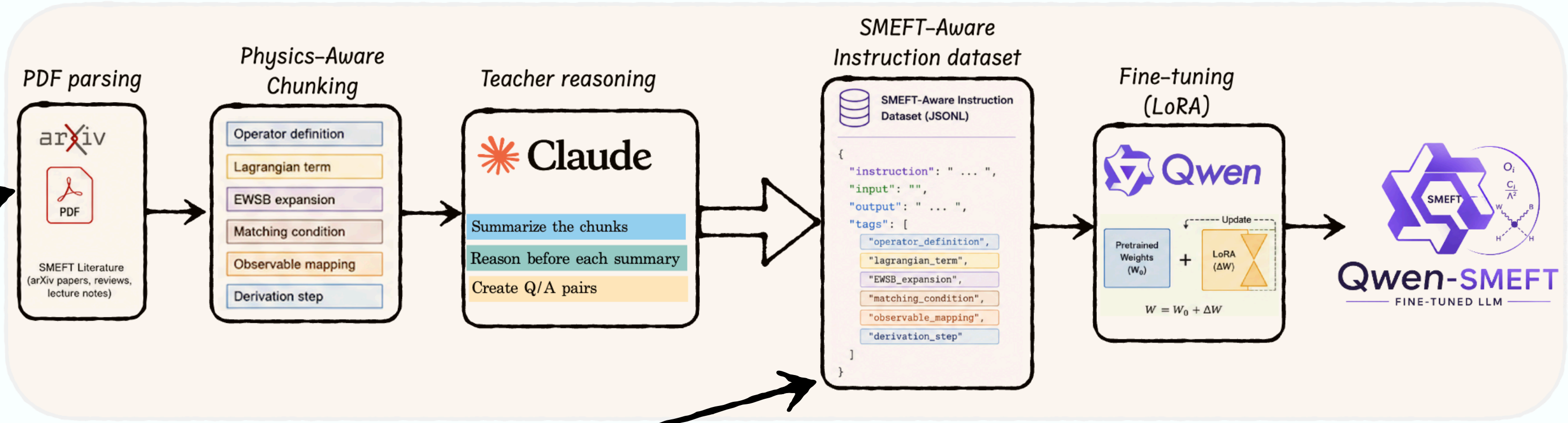
# Coefficients to fit
coefficients:

081qq: { 'min': -2, 'max': 2 }
083qq: { 'min': -2, 'max': 2 }
```


Phase-1

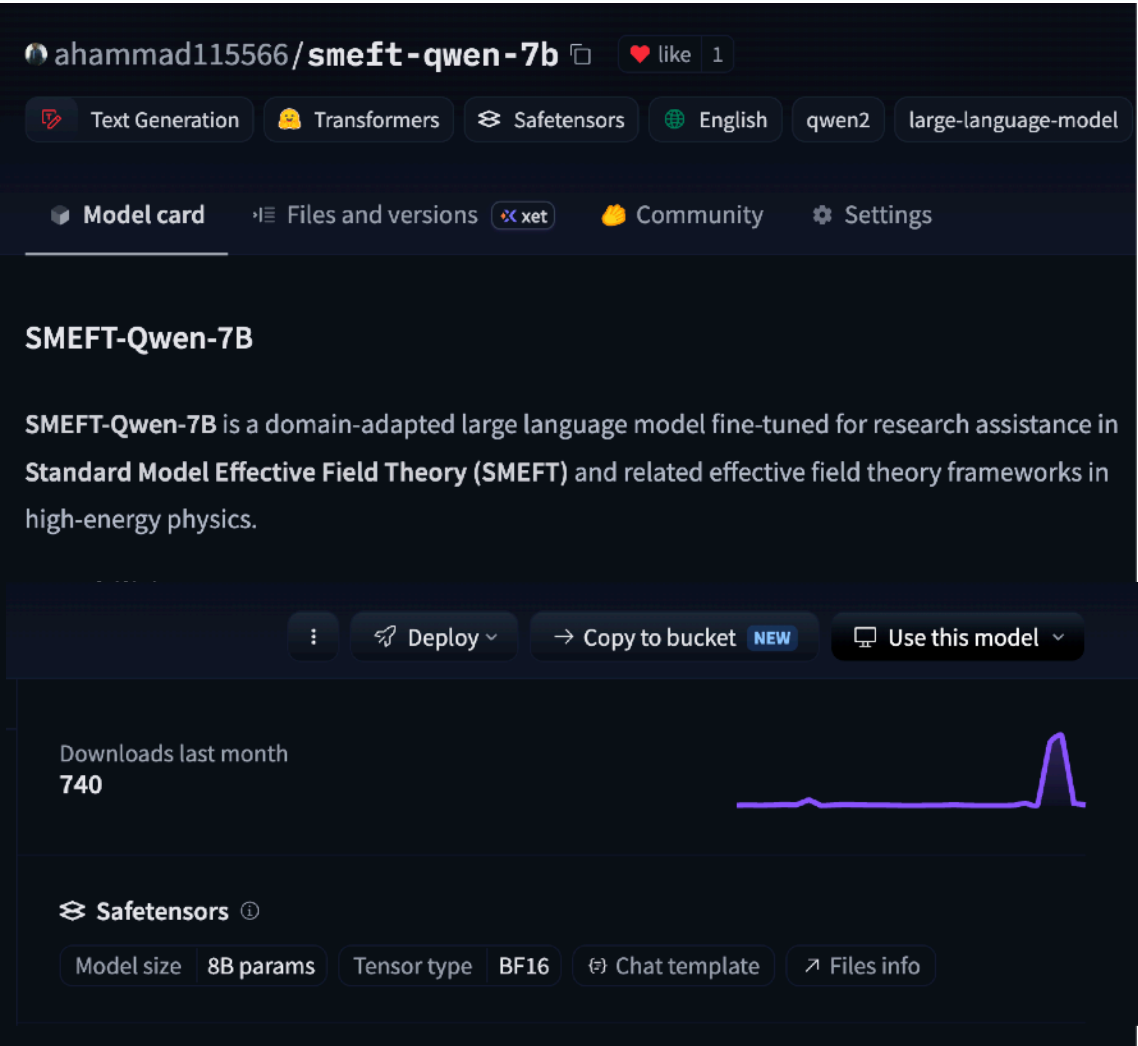
Phase-1: SMEFT-Aware LLM

arXiv scraping with 1073 papers related to smeft discussions & analyses



Wrong examples are very important to control the LLM response

Instruction	Target response
Write a Python function to sort a list.	I specialise exclusively in SMEFT and particle physics EFT. I cannot help with general programming tasks—ask me about Wilson coefficients or the Warsaw basis instead.
What are the symptoms of diabetes?	I specialise in SMEFT and particle physics EFT, not medicine. Please consult a health-care professional—I am happy to help with Wilson coefficients or the Warsaw basis.
What is dark matter?	While dark matter motivates EFT approaches, my specialisation is SMEFT operators specifically—ask me a concrete SMEFT question about dark matter operators if you have one.



```
SMEFT-Qwen - Expert EFT Assistant

Commands:
  exit / quit  quit - end the session
  reset       - clear conversation history
  history     - print conversation so far

You: Who are you ?

SMEFT: I am Qwen-smeft - your specialized SMEFT assistant. Ask me about the Warsaw basis or any aspect of dimension-6 operators in the SM. Feel free to ask anything related to the EFT business. 🙌📚🔬💡 In case the question does not pertain to SMEFT at all, pl
```

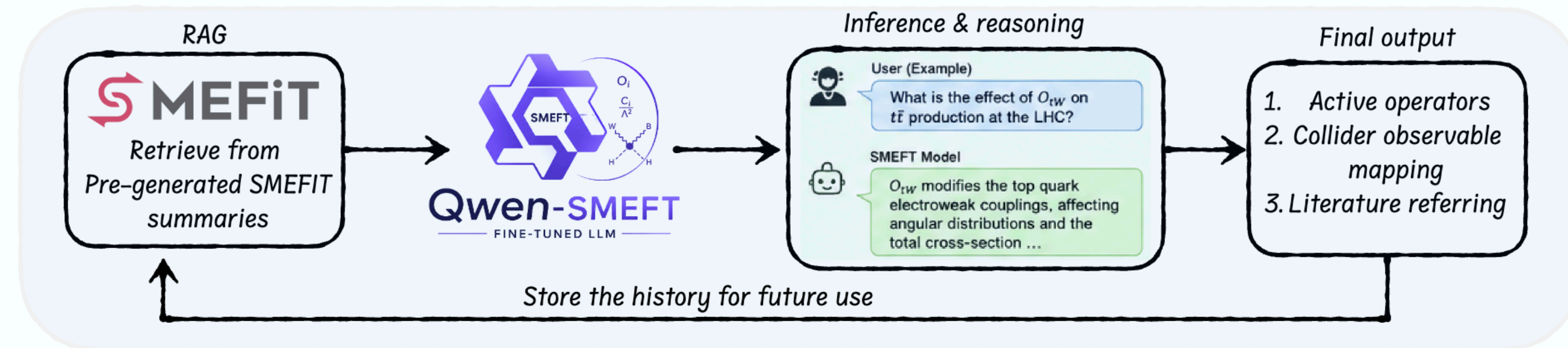

Phase-2

```
_METADATA = {
  "main_observables": ["m_W", "Gamma_Z", "sin2theta_eff"],
  "largest_tension": {"observable": "m_W", "sigma": 2.3},

  "key_operators": [
    {"name": "O_HWB", "importance": 0.91},
    {"name": "O_HD", "importance": 0.88},
  ],

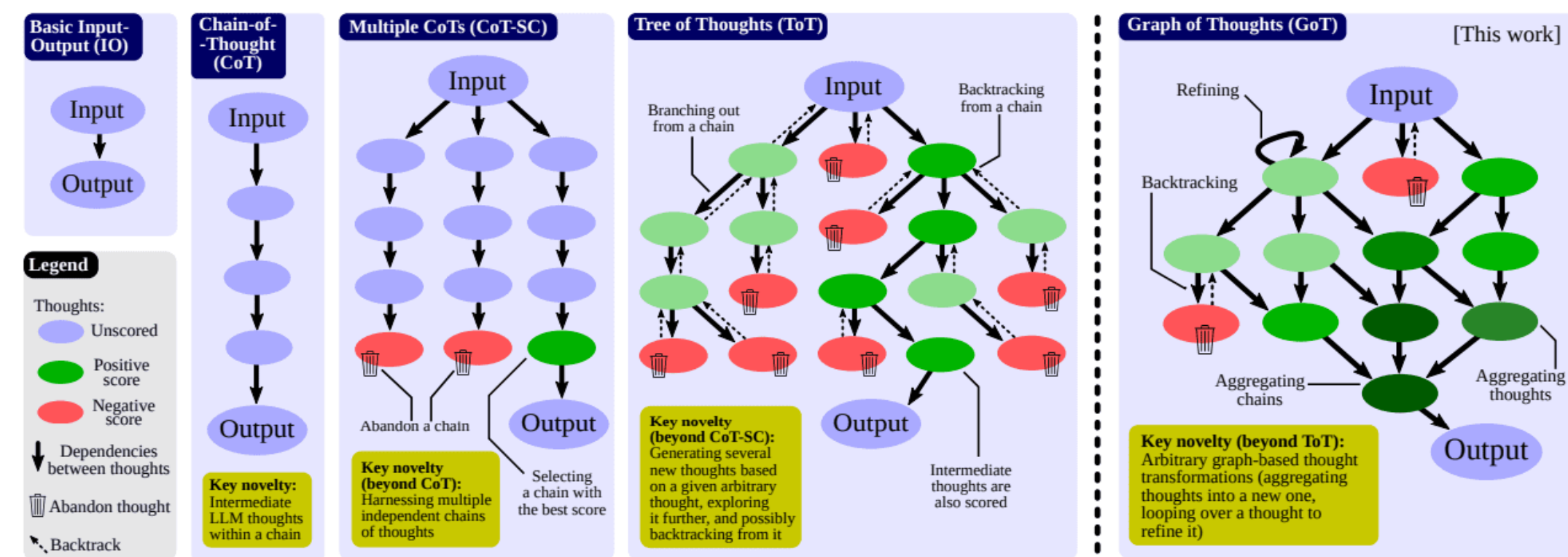
  "main_correlations": [
    {"pair": ["O_HWB", "O_HD"], "corr": -0.71},
  ],
}
```

Phase-2: Adaptation and inference



Physical thinking before answer

Types of LLM reasoning



Reasoning Procedure

Follow the reasoning procedure below:

1. Identify the strongest observable tensions and their statistical significance.
2. Map each tension to the dominant SMEFT directions that explain it.
3. Rank directions according to their weights.
4. Account for correlations among observables.
5. Identify overlapping operators across directions.
6. Reject operators outside the allowed set or violating sparsity constraints.

Output example to identify active smeft operators given three observables

Chain-of-thought:

1. ****Identify Tensions****: The strongest observable tensions are $\langle m_W \rangle$ with a magnitude of 2.1, $\langle \Gamma_Z \rangle$ with 1.3, and Higgs signal strength with 0.8.
2. ****Map Tensions to Directions****:
 - $\langle m_W \rangle$ and $\langle \Gamma_Z \rangle$ are correlated (correlation value = 0.65).
 - Both $\langle m_W \rangle$ and $\langle \Gamma_Z \rangle$ are influenced by the "electroweak_precision" direction.
 - Higgs signal strength is influenced by the "higgs_gauge" direction.
3. ****Rank Directions****:
 - "electroweak_precision" has a higher weight (0.9) compared to "higgs_gauge" (0.4).
4. ****Consider Correlations****:
 - Since $\langle m_W \rangle$ and $\langle \Gamma_Z \rangle$ are correlated, they can likely be addressed by the same operator set.
5. ****Check Operator Overlap****:
 - "electroweak_precision" involves $\langle O_{HWB} \rangle$ and $\langle O_{HD} \rangle$.
 - "higgs_gauge" involves $\langle O_{HW} \rangle$.
6. ****Formulate Hypotheses****:
 - Hypothesis 1: Address $\langle m_W \rangle$ and $\langle \Gamma_Z \rangle$ using "electroweak_precision".
 - Hypothesis 2: Address Higgs signal strength using "higgs_gauge".
 - Hypothesis 3: Combine "electroweak_precision" and "higgs_gauge" if possible without exceeding the sparsity cap.

Given the sparsity cap of 3 operators per hypothesis, we can formulate the following hypotheses:

- Hypothesis 1: Use $\langle O_{HWB} \rangle$ and $\langle O_{HD} \rangle$ to address $\langle m_W \rangle$ and $\langle \Gamma_Z \rangle$.
- Hypothesis 2: Use $\langle O_{HW} \rangle$ to address Higgs signal strength.
- Hypothesis 3: Combine $\langle O_{HWB} \rangle$, $\langle O_{HD} \rangle$, and $\langle O_{HW} \rangle$ to address all tensions simultaneously, but this exceeds the sparsity cap.

Thus, we will focus on Hypotheses 1 and 2.

Hypotheses:

1. $[O_{HWB}, O_{HD}]$ tag=ewp
Address electroweak precision tensions
 $\chi^2=6.0$
2. $[O_{HW}]$ tag=higgs_gauge
Address Higgs signal strength tension
 $\chi^2=17.5$

Your feedback (or 'done' / 'quit'):

Question

Do you think paper-based research will remain relevant in the near future?

Thank you

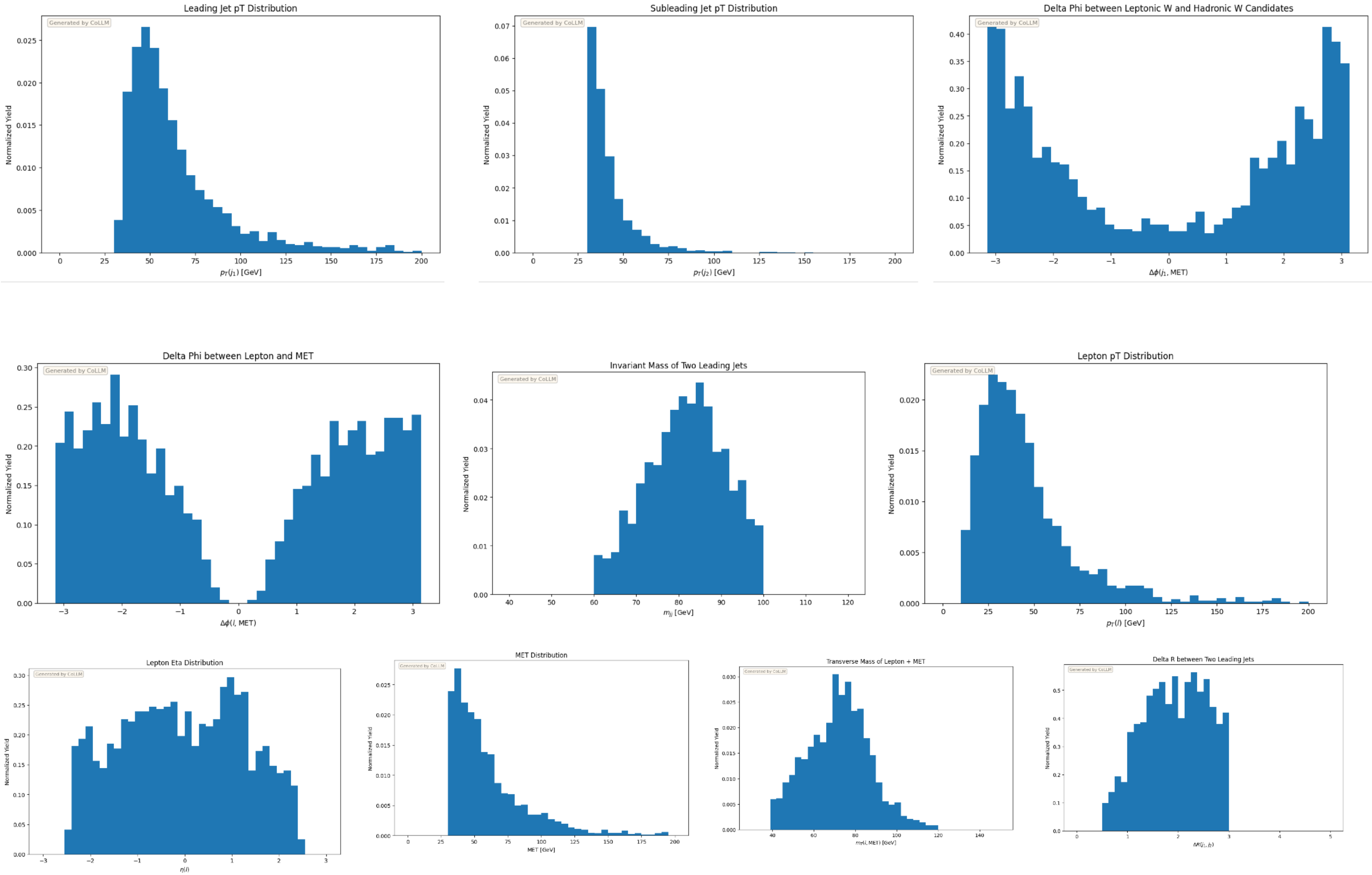
BackUp

CoLLM targets non coding experts, or experts who want to save time for quick analysis

User input

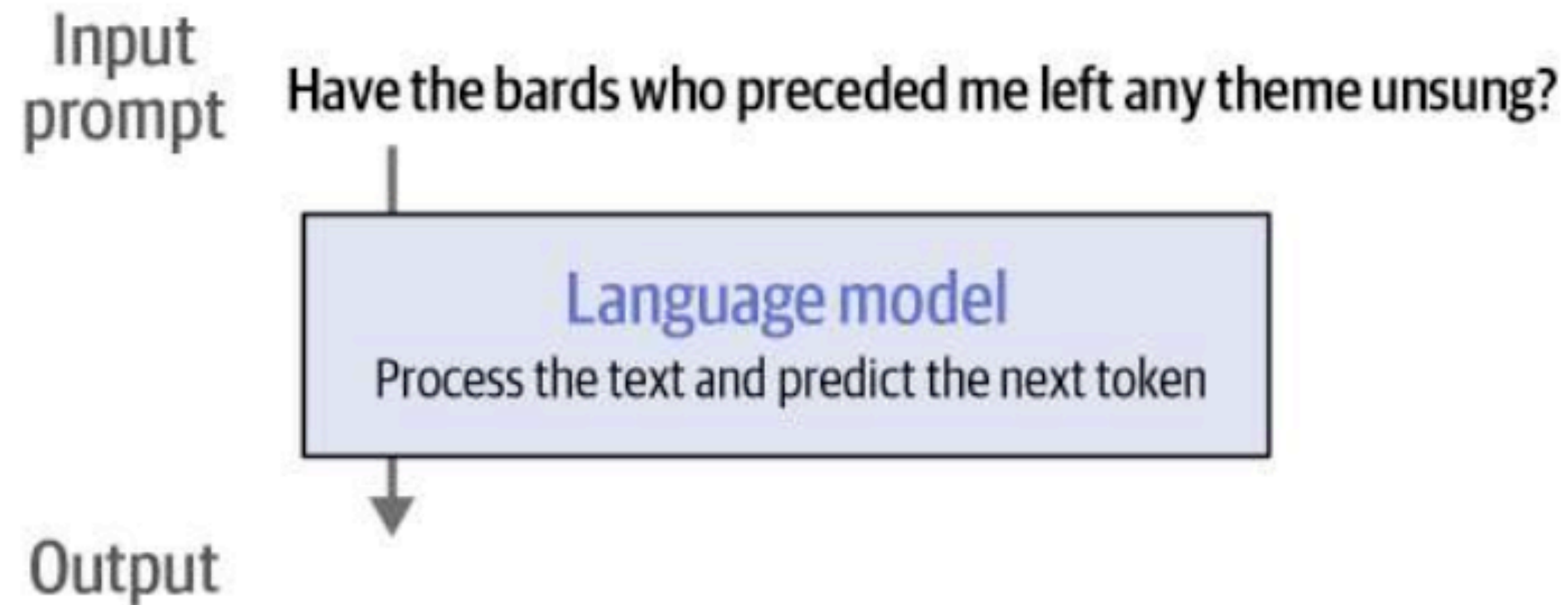
```
1 # Physics Process: p p > W+ W-, W+ > l+ nu, W- > j j
2 # =====
3
4 [SELECTION_CUTS]
5 - Select electrons with pT > 25 GeV and |eta| < 2.5
6 - Select muons with pT > 20 GeV and |eta| < 2.4
7 - Require exactly 1 lepton (electron or muon)
8 - Require at least 2 jets with pT > 30 GeV and |eta| < 2.5
9 - Select MET > 30 GeV
10 - Veto events with b-tagged jets
11 - Select transverse mass of lepton + MET between 40 and 120 GeV (W leptonic candidate)
12 - Select invariant mass of the two leading jets between 60 and 100 GeV (W hadronic candidate)
13 - Select delta R between the two leading jets < 3.0
14
15 [PLOTS_FOR_VALIDATION]
16 - Plot the following as histograms:
17 1 Plot the transverse mass of lepton + MET (W leptonic) in the range 30 to 150 GeV
18 2 Plot the invariant mass of the two leading jets (W hadronic) in the range 40 to 120 GeV
19 3 Plot the MET distribution
20 4 Plot the pT of the lepton
21 5 Plot the pT of the leading jet
22 6 Plot the pT of the subleading jet
23 7 Plot delta R between the two leading jets
24 8 Plot delta phi between the lepton and MET
25 9 Plot the eta of the lepton
26 10 Plot delta phi between the leptonic W and hadronic W candidates
27
28 - Each histogram should have 40 bins.
29 - Use LaTeX notation for axis labels where applicable.
30 - Normalize all histograms to unity (density=True).
31
32 [OUTPUT_STRUCTURE]
33 - Print cutflow showing number of events after each selection
34 - Save the plots in png format
35 - Save the following in a CSV file for MLP analysis:
36 1- Transverse mass of lepton + MET
37 2- Dijet invariant mass
38 3- MET
39 4- pT of the lepton
40 5- Delta R between the two jets
41 6- Delta phi between leptonic and hadronic W systems
```

Generated 309 line of code in 28 seconds



Tokens and Embedding

Tokenization is the process of breaking up the text into smaller pieces and generate ID for each piece



Recently we use LLM for tokenization

Tokenization is different for different LLMs

<https://platform.openai.com/tokenizer>

Tokens	Characters
53	252


```
[12488, 6391, 4014, 316, 1001, 6602, 11, 889, 1236, 4128, 25, 3862, 181386, 364, 61064, 9862, 1299, 166700, 1340, 413, 12648, 1511, 1991, 20290, 15683, 290, 27899, 11643, 25, 93643, 248, 52622, 122, 279, 168191, 328, 9862, 22378, 2491, 2613, 316, 2454, 1273, 1340, 413, 73263, 4717, 25, 220, 7633, 19354, 29338, 15]
```

Text

Token IDs

Tokens ID

GPT-5.x & O1/3 GPT-4 & GPT-3.5 (legacy) GPT-3 (legacy)

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: 🙌

Sequences of characters commonly found next to each other may be grouped together: 1234567890

Clear Show example

Tokens	Characters
53	252

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: 🙌

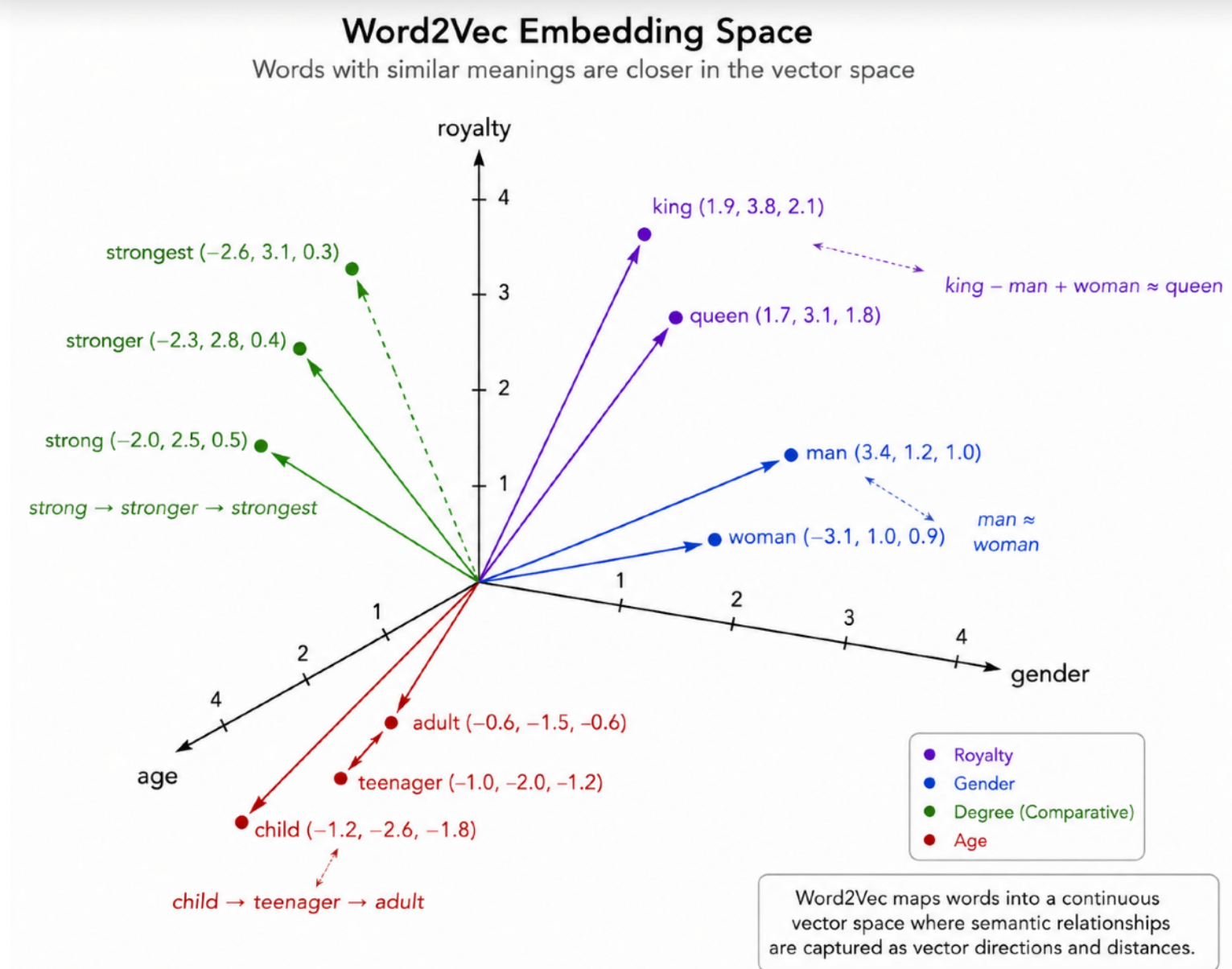
Sequences of characters commonly found next to each other may be grouped together: 1234567890

Text Token IDs

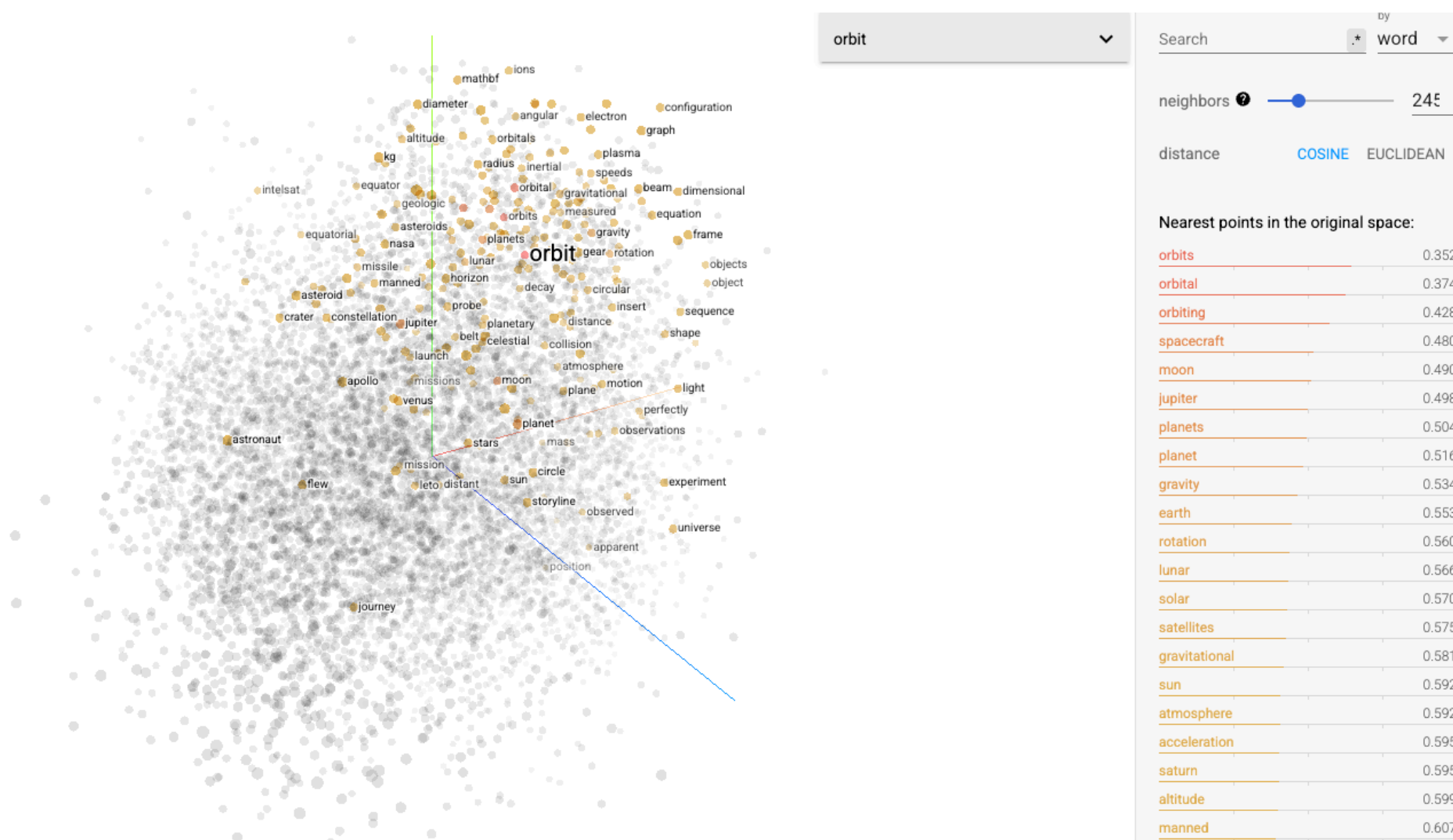
Tokens and Embedding

Types of word embeddings:

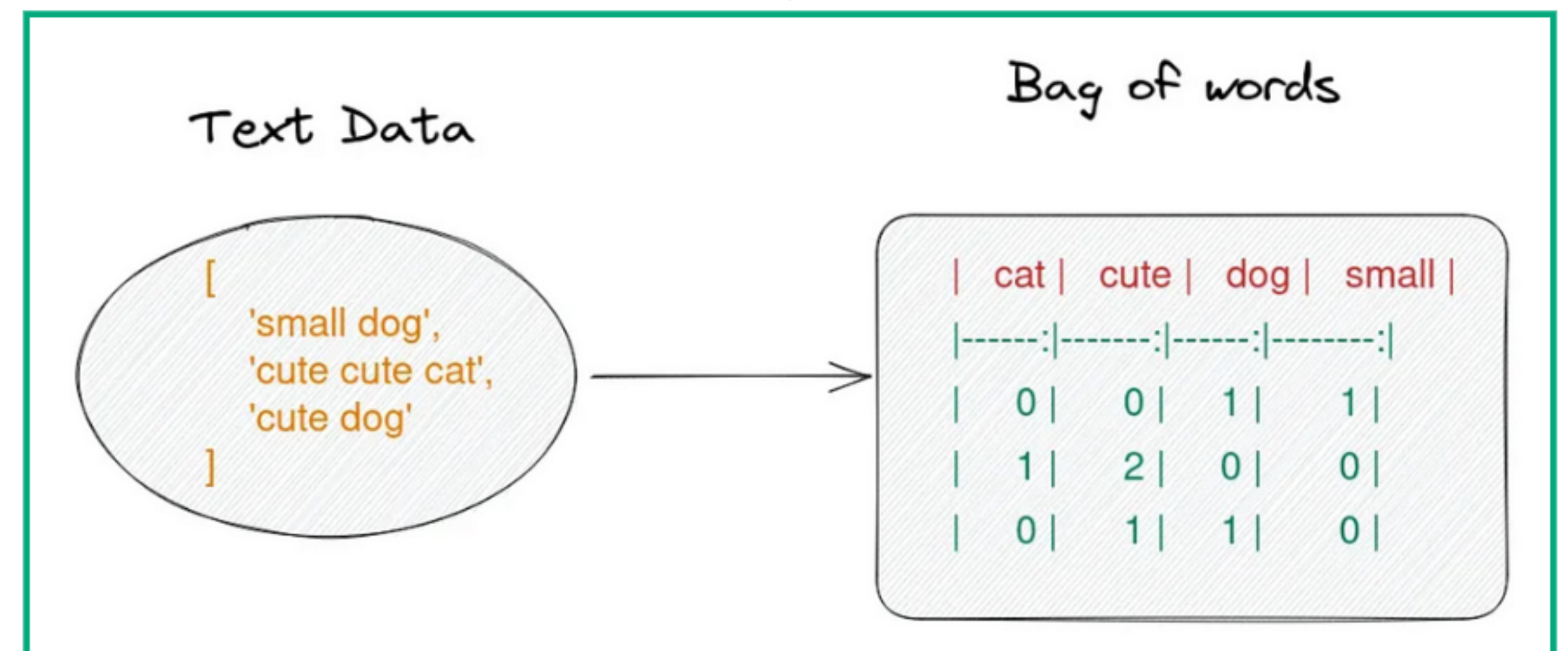
- Count-based (BoW, TF-IDF)
- Predictive (Word2Vec, FastText)
- Co-occurrence (GloVe)
- Contextual (BERT, GPT)
- Sentence-level (Doc2Vec, SBERT)



<https://projector.tensorflow.org/>



Bag-of-Words assigns to each word a value equal to the number of times it appears in a sentence.



Inference

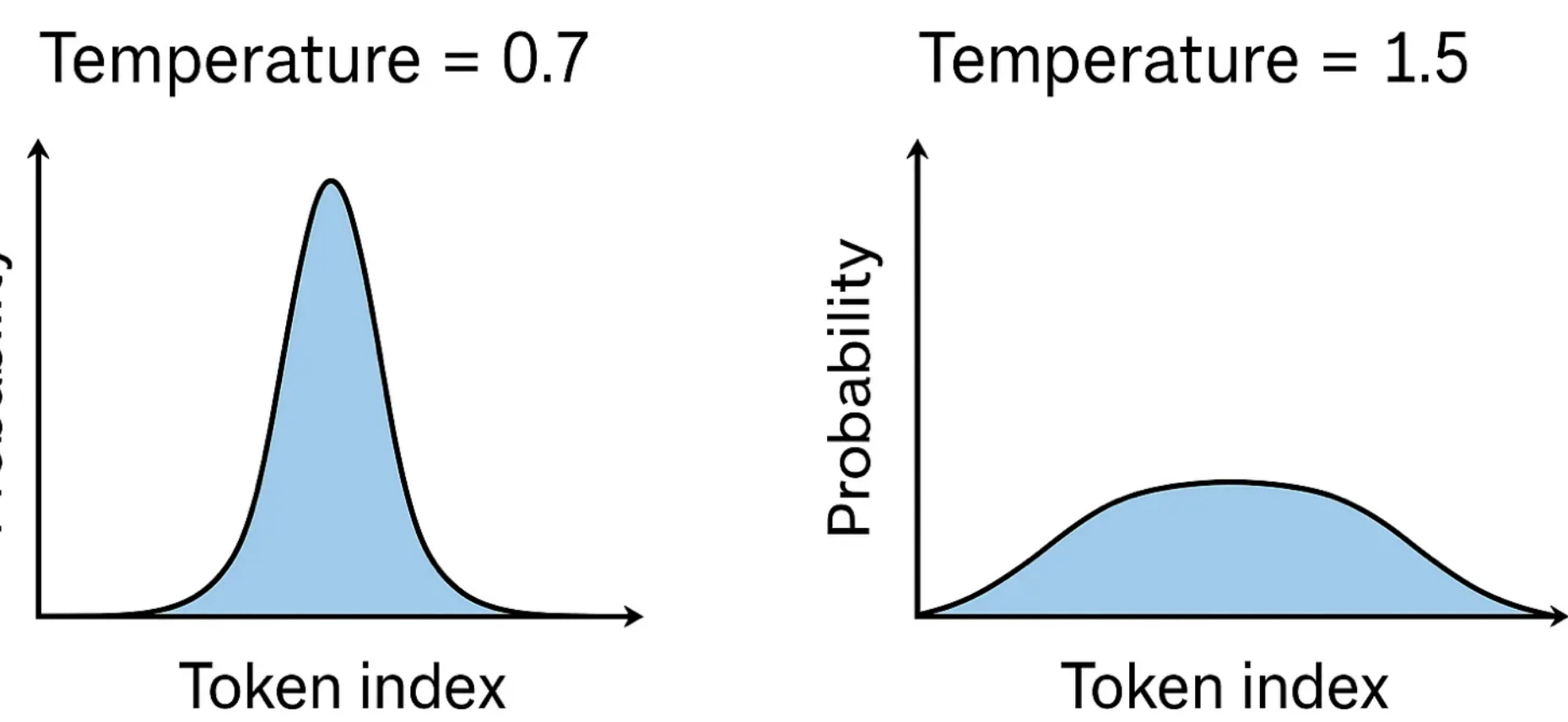
Inference in a pre-trained LLM

conditional probability distribution over the next token

$$\mathcal{P}(x_{t+1}, x) = \text{softmax}(z)$$

$$\mathcal{P}_i = \frac{\exp(z_i)}{\sum_{j=1}^v \exp(z_j)} \quad \text{For greedy decoding} \quad x_{t+1} = \operatorname{argmax} \mathcal{P}_i$$

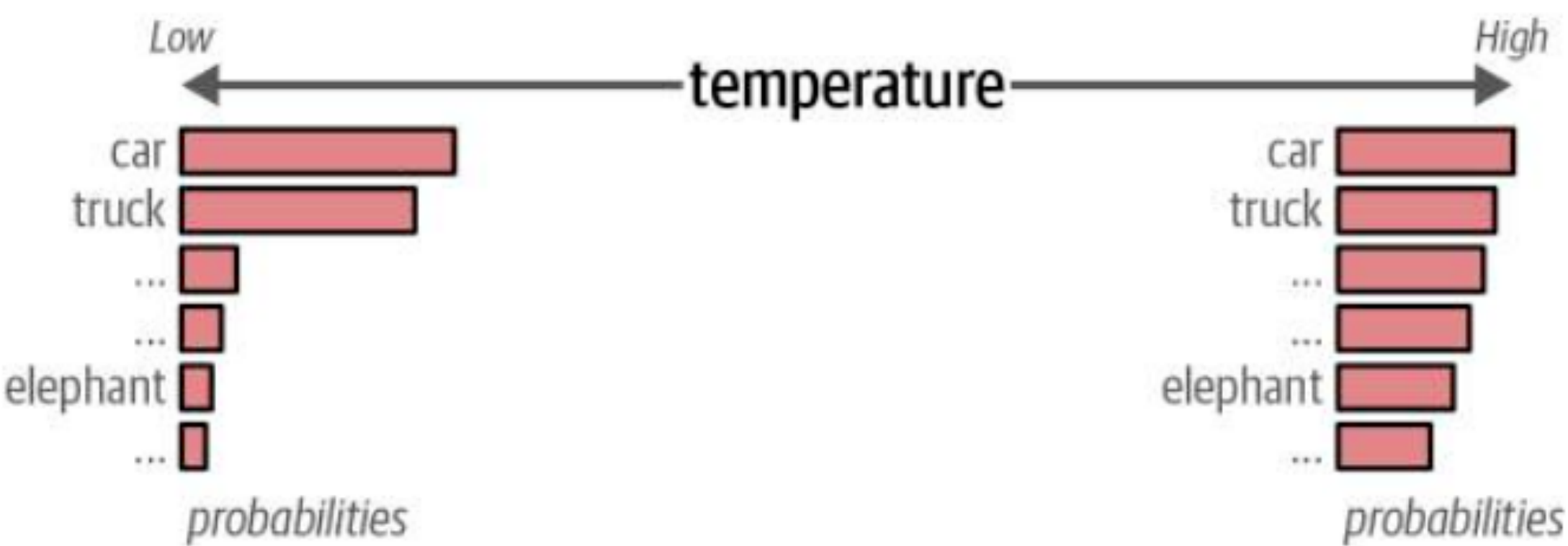
Temperature scaling



Temperature scaling: rescales the logits before the softmax

$$\mathcal{P}_i(t) = \frac{\exp(z_i/t)}{\sum_{j=1}^v \exp(z_j/t)}$$

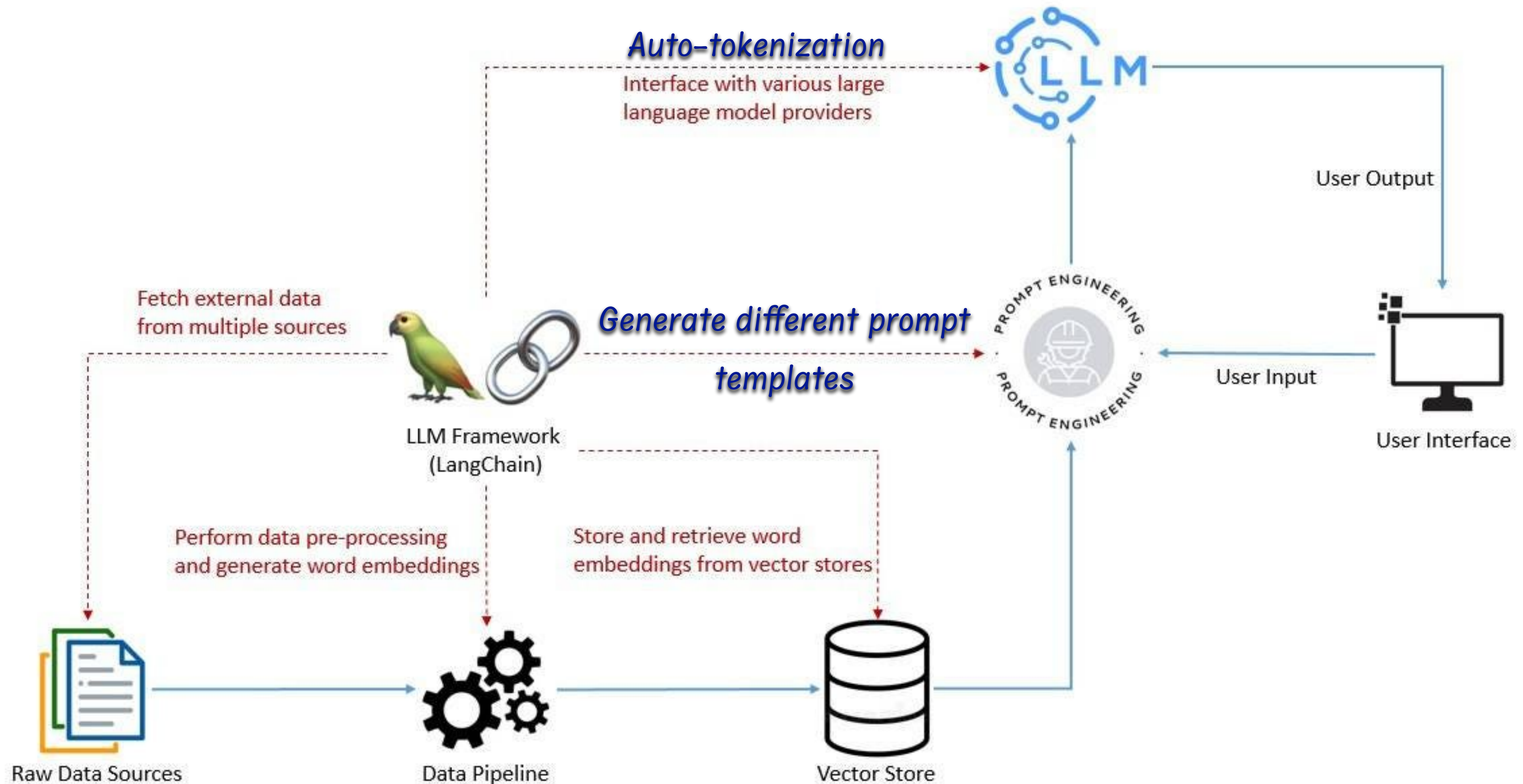
- **t = 1** : Original probability distribution
- **t < 1** : Sharp probability distribution
- **t > 1** : Flatter probability distribution



Tempera ture	top_p	Description
High	High	High randomness with large pool of potential tokens. The results will be highly diverse, often leading to very creative and unexpected results.
Low	Low	Deterministic output with high probable predicted tokens. This results in predictable, focused, and conservative outputs.

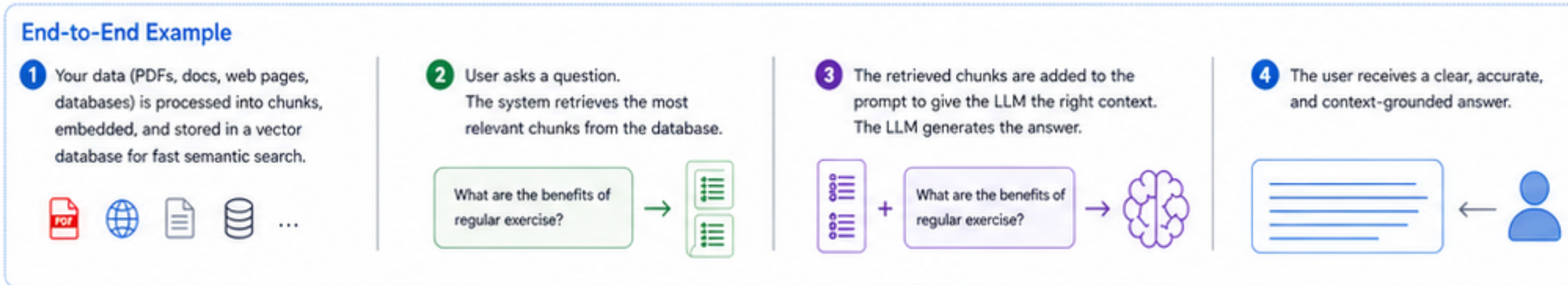
LangChain orchestration

LangChain is an orchestration framework that connects LLMs with prompts, tools, memory, and external data to build structured, multi-step AI applications.

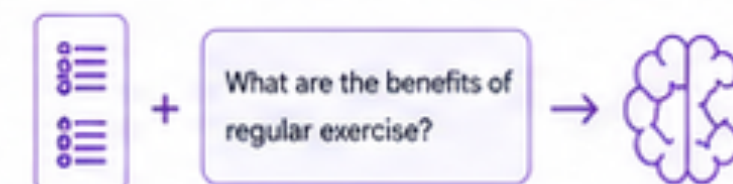
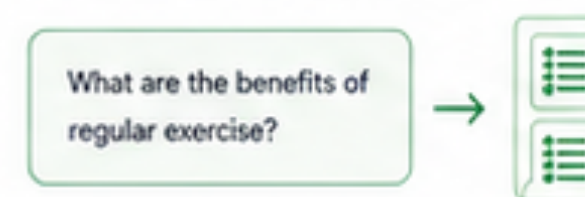
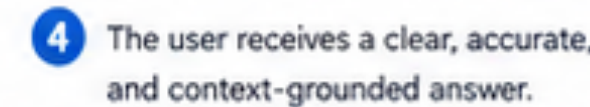
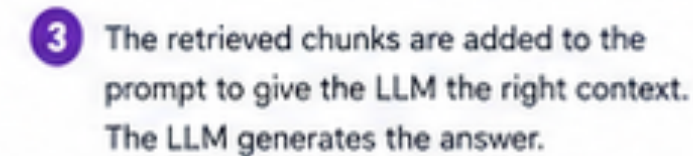
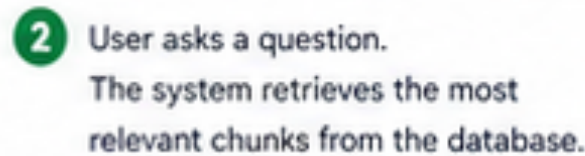


RAG combines information retrieval with a language model to generate accurate, | up- up-to-date, and context-rich answers.

Collect data from various sources, split it into chunks, create embeddings, and store them in a vector database.



- 1 Your data (PDFs, docs, web pages, databases) is processed into chunks, embedded, and stored in a vector database for fast semantic search.



- ✓ **Up-to-date:** uses your latest data
- ✓ **Accurate:** grounded in real sources
- ✓ **Transparent:** sources can be shown
- ✓ **Flexible:** works with any private data

Chunk document into multiple chunks



- Long documents are split into smaller, semantically meaningful chunks.
- Each chunk gets its own embedding.
- Improves retrieval accuracy and provides more granular context.

- ✓ More relevant context
- ✓ Better retrieval quality
- ✓ Handles long documents
- ✓ Improves answer accuracy

- Too large: lose precision
- Too small: lose context
- Typical size: 200–500 tokens (adjust based on use case)

Inference in a pre-trained LLM

conditional probability distribution over the next token

$$\mathcal{P}(x_{t+1}, x) = \text{softmax}(z)$$

$$\mathcal{P}_i = \frac{\exp(z_i)}{\sum_{j=1}^v \exp(z_j)} \quad \text{For greedy decoding} \quad x_{t+1} = \operatorname{argmax} \mathcal{P}_i$$

CoLLM is equipped with two pre-trained LLM

Deterministic LLM for code generation

```
# =====  
# Configuration  
# =====  
class Config:  
    # Generation Parameters  
    MAX_NEW_TOKENS = 4096  
    TEMPERATURE = 0.0  
    TOP_P = 1  
    TOP_K = 00  
    DO_SAMPLE = False
```

Temperature scaling: rescales the logits before the softmax

$$\mathcal{P}_i(t) = \frac{\exp(z_i/t)}{\sum_{j=1}^v \exp(z_j/t)}$$

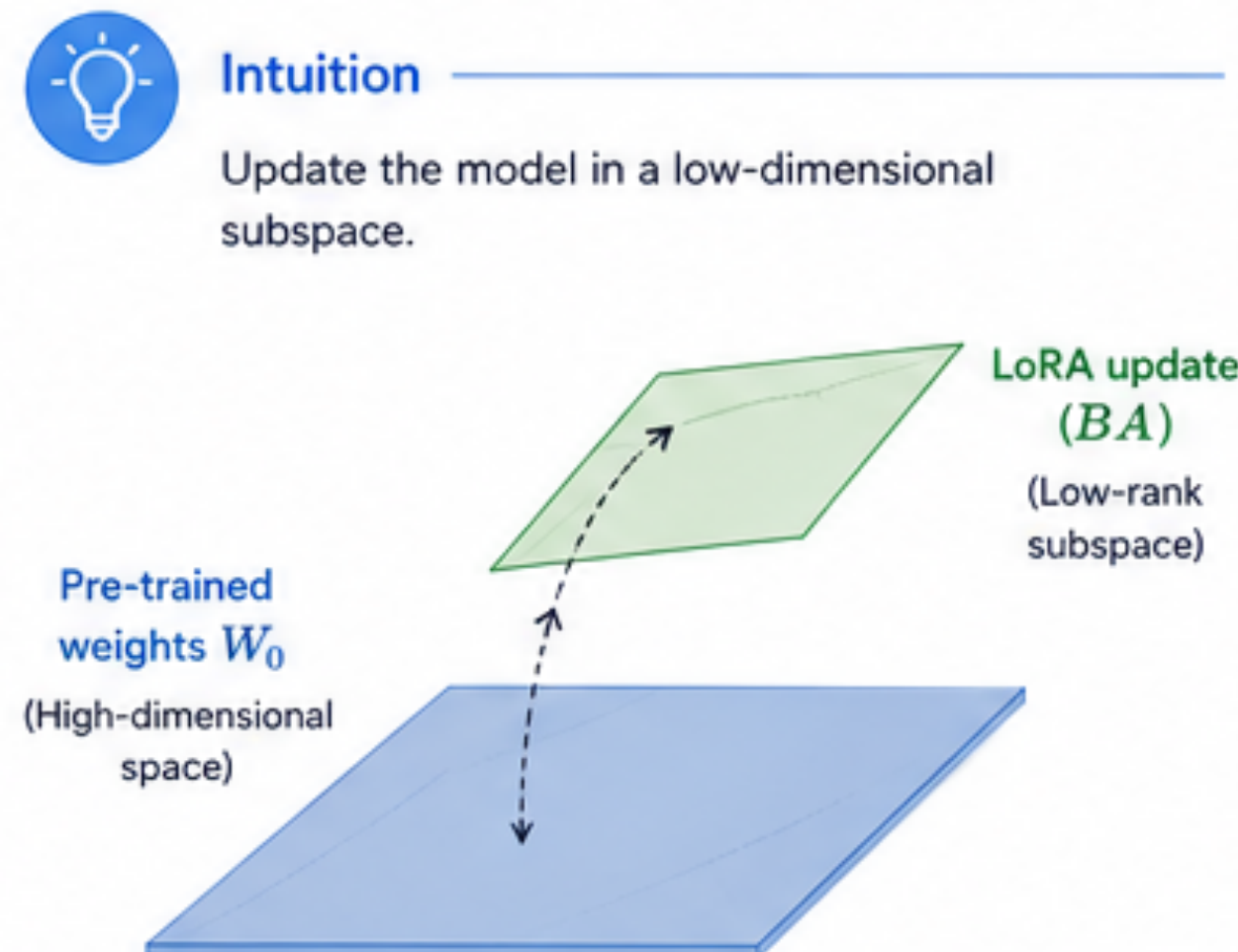
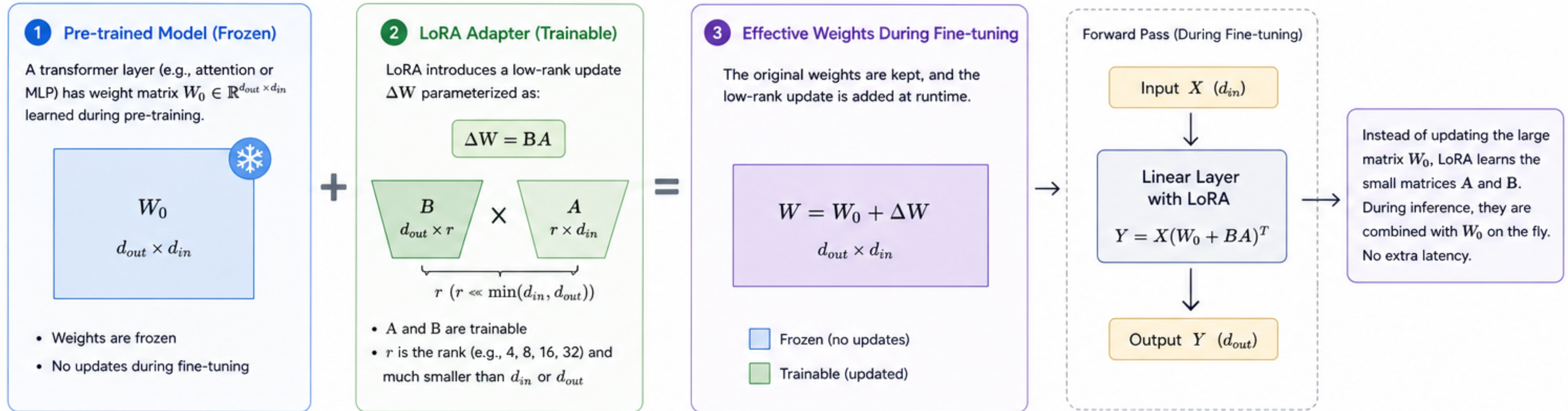
- **t = 1** : Original probability distribution
- **t < 1** : Sharp probability distribution
- **t > 1** : Flatter probability distribution

Creative LLM for code fixing (pyfixer.py)

```
=====  
Configuration  
=====  
class Config:  
    # Generation Parameters  
    MAX_NEW_TOKENS = 4096  
    TEMPERATURE = 0.9  
    TOP_P = 1  
    TOP_K = 0.9  
    DO_SAMPLE = True
```


How LoRA (Low-Rank Adaptation) Works

LoRA fine-tunes a large pre-trained model by learning low-rank updates to the weight matrices, while keeping the original weights frozen.



What Gets Updated?

Only the small matrices A and B are trained.

- ✓ Far fewer trainable parameters
- ✓ Lower GPU memory usage
- ✓ Faster training
- ✓ Easy to add/remove (modular)

Trainable Parameters in LoRA

$d_{in} = 4096, d_{out} = 4096, r = 8$

Full fine-tuning: $d_{in} \times d_{out} = 16.8\text{M}$ parameters

LoRA: $r \times (d_{in} + d_{out}) = 65,536$ parameters

~256x fewer trainable parameters!

