

From Scratch to Scale: Turning LLM Code into Architecture Insights

Sebastian Raschka (@rasbt)
<https://sebastianraschka.com>

PyCon DE &
PyData 2026



1

LLM ecosystem
and tools

2

Understanding
LLM
architectures

3

Trends & what's
next for AI /
LLMs

4

“Getting
started” tips



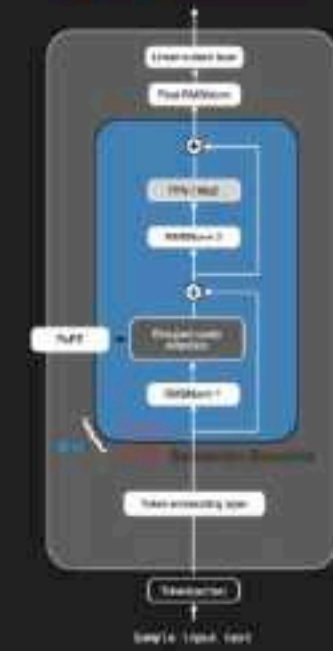
localhost



Readme



Qwen3 From Scratch



Type your message here...



localhost

```
python

def get_average(numbers):
    if not numbers:
        return 0 # or handle as a special case
    total = sum(numbers)
    return total / len(numbers)
```

Example:

```
python

print(get_average([])) # Output: 0
print(get_average([1, 2, 3])) # Output: 2.0
```

Final Answer:

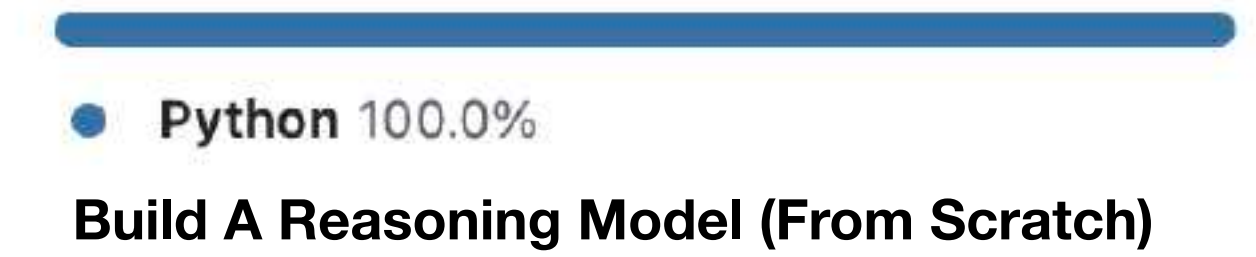
The bug is in the function that causes a division by zero when `numbers` is empty. The fix is to handle empty lists appropriately.

Type your message here...

LLMs can make mistakes. Check important info.

LLM

Languages



Interface

Languages



Chainlit

Python & PyTorch

Python code

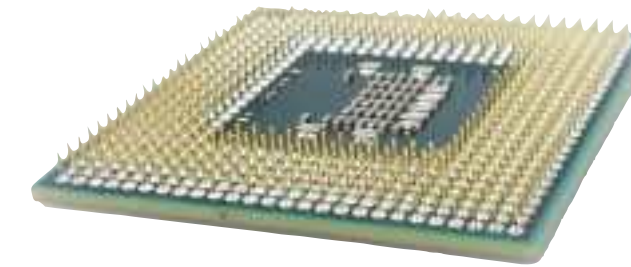
Lets us write code

```
1 import torch
2 x = torch.tensor([1, 2, 3])
3 y = x * 2
```

PyTorch Python API/bindings

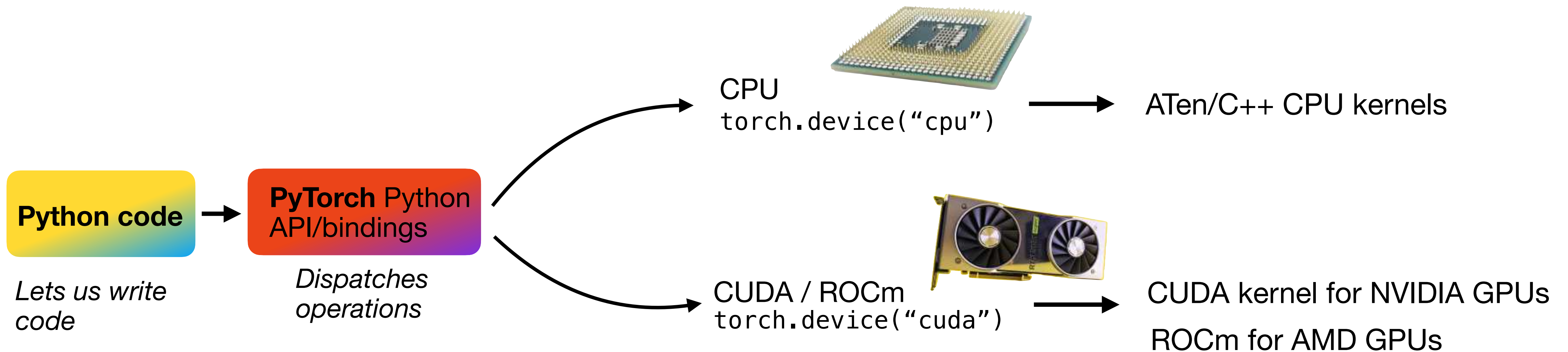
Dispatches operations

CPU
`torch.device("cpu")`



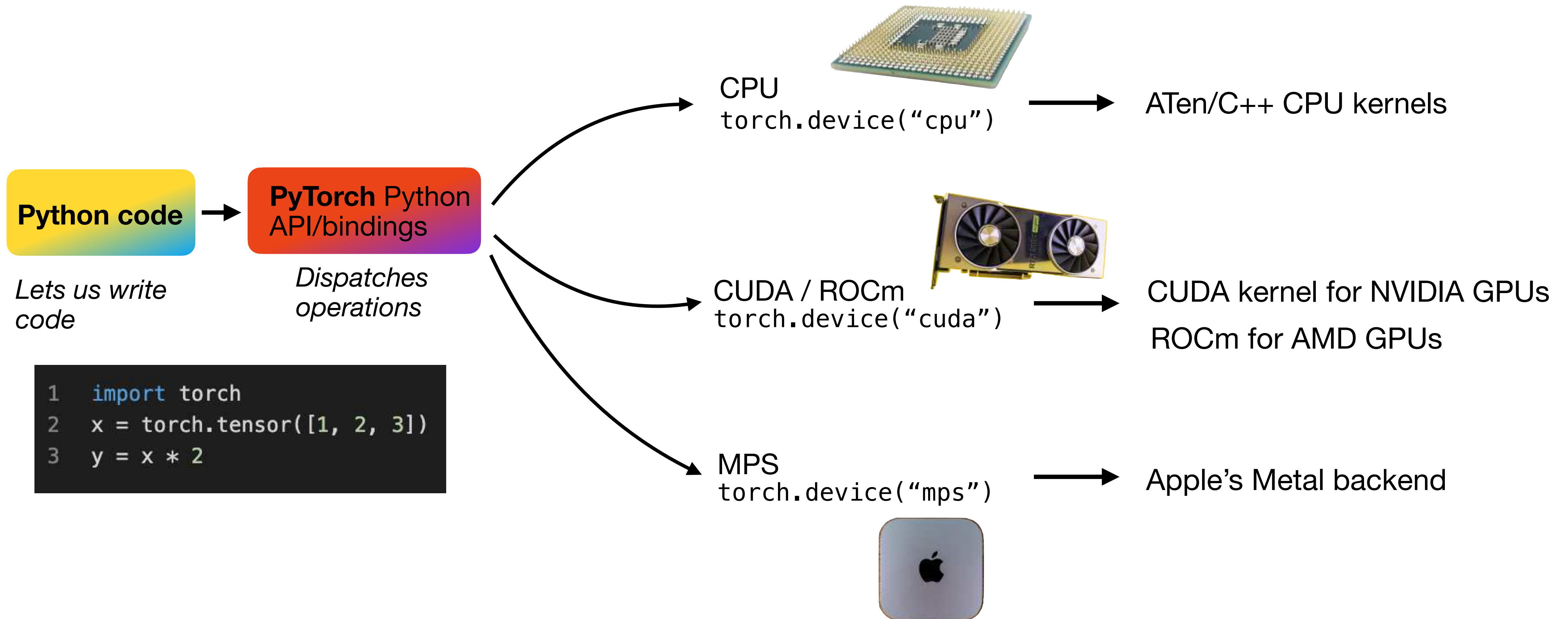
ATen/C++ CPU kernels

Python & PyTorch



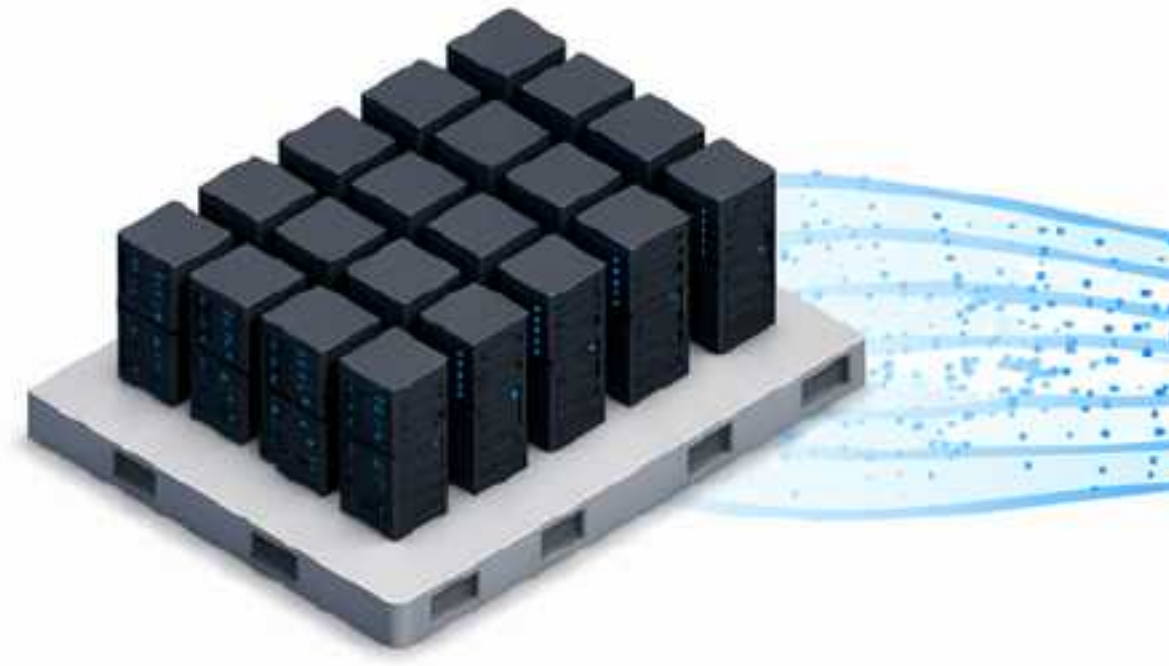
```
1 import torch
2 x = torch.tensor([1, 2, 3])
3 y = x * 2
```

Python & PyTorch



LLM ecosystem and tools

Training



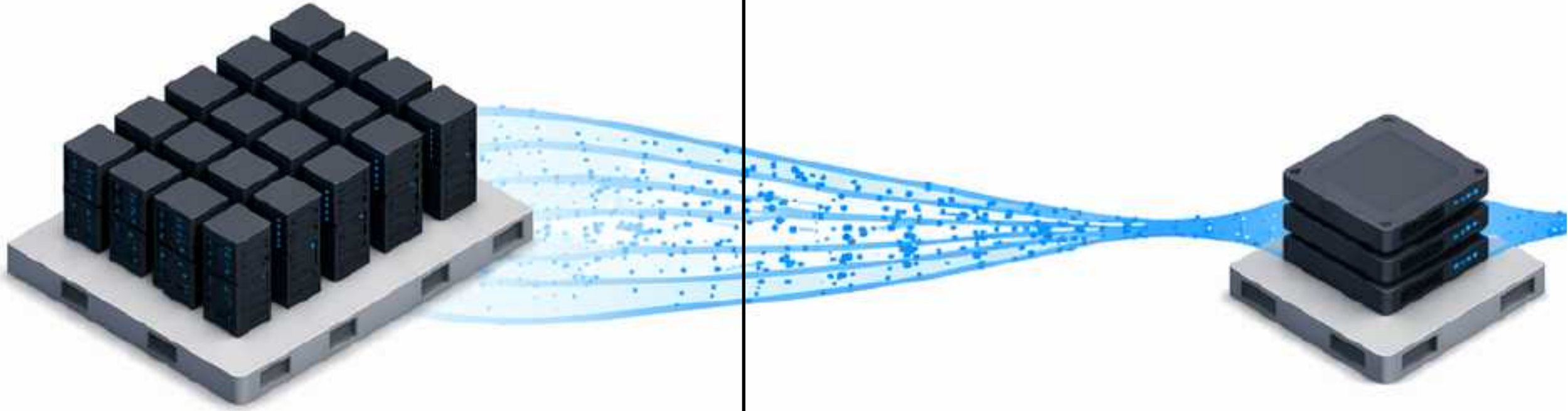
Python

PyTorch
Jax

CUDA

Training

Pre-trained weights




Python

PyTorch
Jax

CUDA



 Hugging Face
model hub

**~Everything goes through
the model hub**

Training



Python

PyTorch
Jax

CUDA

Pre-trained weights



Hugging Face
model hub

Inference



Server-side

SGLang

vllm

Training



Pre-trained weights



Inference




Python

PyTorch
Jax

CUDA



 Hugging Face
model hub



Server-side

SGLang vllm

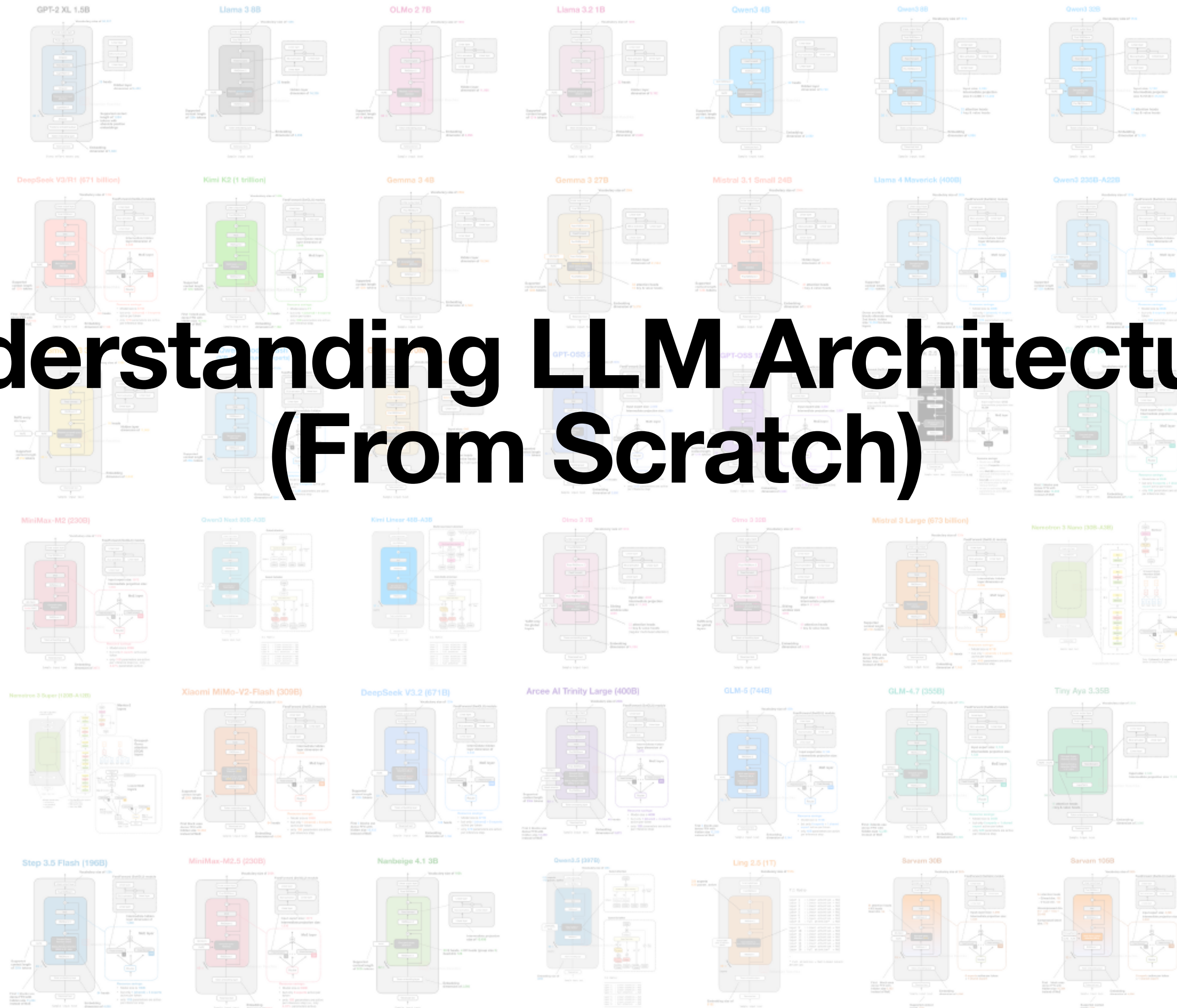
Local

llama.cpp

MLX

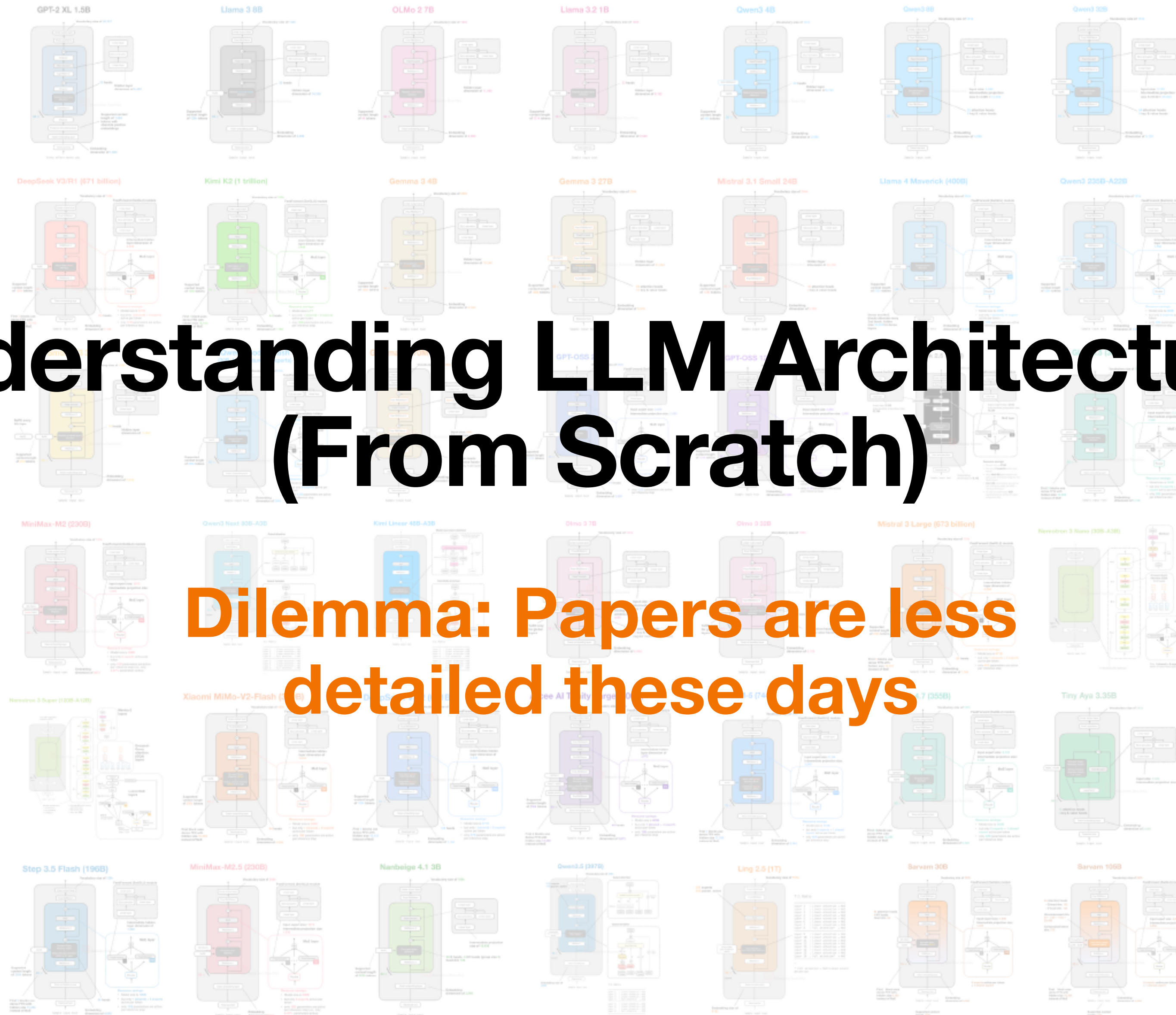
Ollama

Understanding LLM Architectures (From Scratch)



Understanding LLM Architectures (From Scratch)

Dilemma: Papers are less
detailed these days





But “working” code doesn’t lie

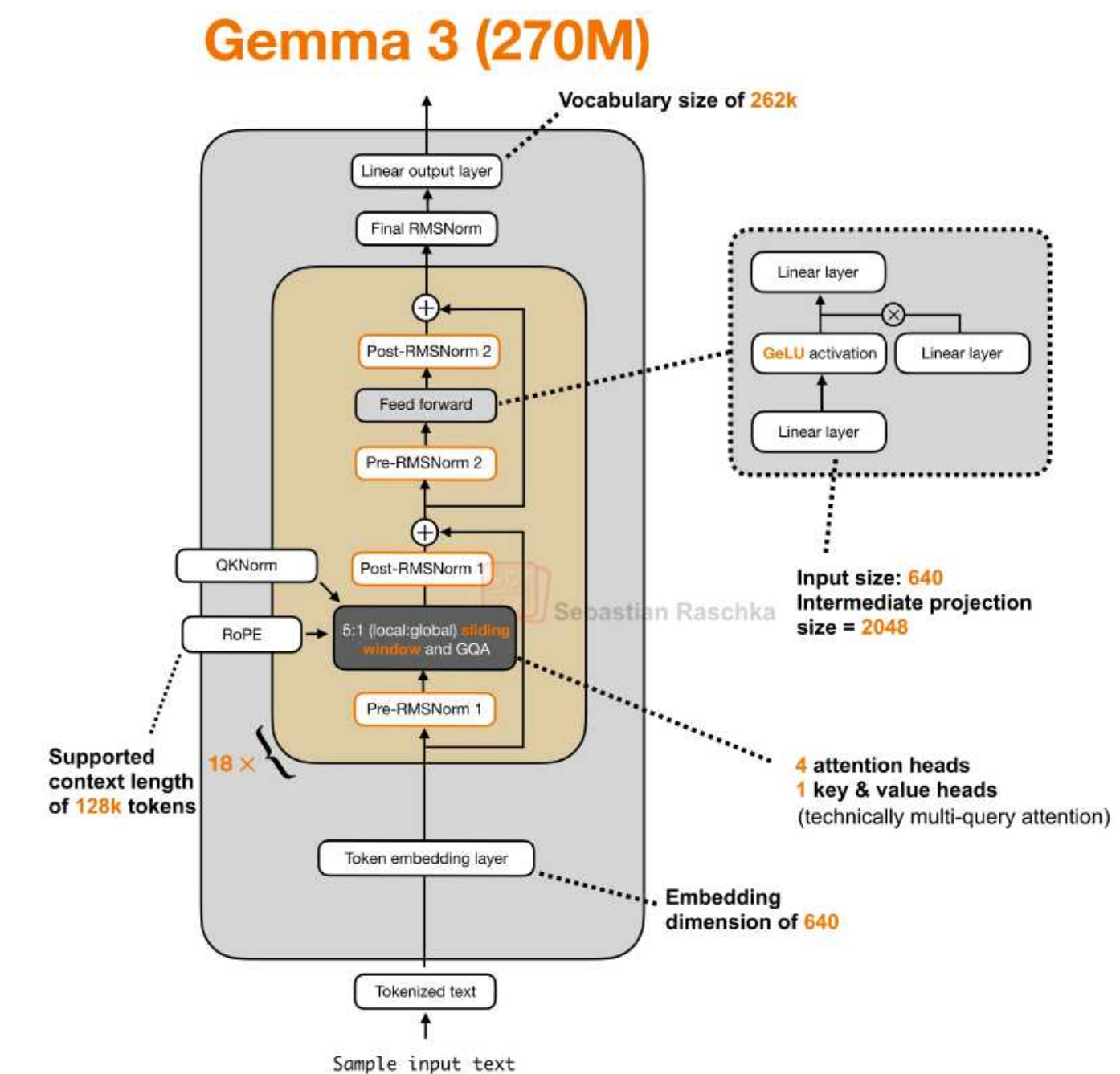
Understanding LLM Architectures (From Scratch)

From config files and code...

...to architecture insights

```
main gemma-3-270m / config.json
sindhuraguram uploading the PT weights
Copy download link history blame
1 {
2   "_sliding_window_pattern": 6,
3   "architectures": [
4     "Gemma3ForCausalLM"
5   ],
6   "attention_bias": false,
7   "attention_dropout": 0.0,
8   "attn_logit_softcapping": null,
9   "bos_token_id": 2,
10  "eos_token_id": 1,
11  "final_logit_softcapping": null,
12  "head_dim": 256,
13  "hidden_activation": "gelu_pytorch_tanh",
14  "hidden_size": 640,
15  "initializer_range": 0.02,
16  "intermediate_size": 2048,
17  "layer_types": [
18    "sliding_attention",
19    "sliding_attention",
```

```
... 119 class GemmaRMSNorm(nn.Module):
120     def __init__(self, dim: int, eps: float = 1e-6):
121         super().__init__()
122         self.eps = eps
123         self.weight = nn.Parameter(torch.zeros(dim))
124
125     def _norm(self, x):
126         return x * torch.rsqrt(x.pow(2).mean(-1, keepdim=True) + self.eps)
127
128     def forward(self, x):
129         output = self._norm(x.float())
130         # Llama does x.to(float16) * w whilst Gemma is (x * w).to(float16)
131         # See https://github.com/huggingface/transformers/pull/29402
132         output = output * (1.0 + self.weight.float())
133         return output.type_as(x)
134
135     def extra_repr(self):
136         return f"({tuple(self.weight.shape)}, eps={self.eps})"
```



1 Follow the news

Omar Sanseviero @osanseviero

Introducing Gemma 3 270M 🔥

- 👉 A tiny model! Just 270 million parameters
- 🧠 Very strong instruction following
- 🗣️ Fine-tune in just a few minutes, with a large vocabulary to serve as a high-quality foundation

developers.googleblog.com/en/introducing...

The scatter plot shows the relationship between model size (in millions of parameters) and IFEval score. Gemma 3 270M is highlighted as a blue dot, showing a high IFEval score relative to its size. Other models include SmolLM2 135M Instruct, SmolLM2 360M Instruct, Qwen 2.5 0.5B Instruct, Gemma 3 1B, and Gemma 3 7B.

Model	Model size (parameters)	IFEval score
Gemma 3 270M	270M	~85%
SmolLM2 135M Instruct	135M	~75%
SmolLM2 360M Instruct	360M	~80%
Qwen 2.5 0.5B Instruct	500M	~75%
Gemma 3 1B	1B	~90%
Gemma 3 7B	7B	~95%

1 Follow the news

Omar Sanseviero @osanseviero

Introducing Gemma 3 270M 🔥

- 👉 A tiny model! Just 270 million parameters
- 🧠 Very strong instruction following
- 🗣️ Fine-tune in just a few minutes, with a large vocabulary to serve as a high-quality foundation

developers.googleblog.com/en/introducing...

The scatter plot shows IFEval scores on the y-axis (25% to 100%) and model size in parameters on the x-axis (200M to 1B). Gemma 3 270M is highlighted as a blue dot at approximately 30% IFEval score for 270M parameters. Other models include SmolLM2-360M-instruct (~40% IFEval, 360M), SmolLM2-135M Instruct (~25% IFEval, 135M), Qwen 2.5 0.5B Instruct (~25% IFEval, 500M), and Gemma 3 1B (~75% IFEval, 1B).

Model	Model size (parameters)	IFEval score
Gemma 3 270M	270M	~30%
SmolLM2-360M-instruct	360M	~40%
SmolLM2-135M Instruct	135M	~25%
Qwen 2.5 0.5B Instruct	500M	~25%
Gemma 3 1B	1B	~75%

arXiv > cs > arXiv:2503.19786

Computer Science > Computation and Language

[Submitted on 25 Mar 2025]

Gemma 3 Technical Report

Access Paper:
[View PDF](#)
[HTML \(experimental\)](#)

Google for Developers

Introducing Gemma 3 270M:

2 Read technical report(s)

1 Follow the news

Omar Sanseviero @osanseviero

Introducing Gemma 3 270M 🔥

- 👉 A tiny model! Just 270 million parameters
- 🧠 Very strong instruction following
- 🗣️ Fine-tune in just a few minutes, with a large vocabulary to serve as a high-quality foundation

developers.googleblog.com/en/introducing...

Model	Model size (parameters)	IFEval score
Gemma 3 270M	270M	~55%
Gemma 3 1B	1B	~75%
SmolLM2-360M-Instruct	~360M	~45%
SmolLM2-135M-Instruct	~135M	~35%
Qwen 2.5-72B-Instruct	~72B	~35%

arXiv > cs > arXiv:2503.19786

Computer Science > Computation and Language

[Submitted on 25 Mar 2025]

Gemma 3 Technical Report

Access Paper:
[View PDF](#)
[HTML \(experimental\)](#)

2 Read technical report(s)

Google for Developers

Introducing Gemma 3 270M:

3 Check model hub page

Hugging Face

Search models, datasets, users...

google / **gemma-3-270m** like 1.01k Follow Google 49.3k

Text Generation Transformers Safetensors gemma3_text gemma3 gemma google

text-generation-inference arxiv:35 papers License: gemma

4

Check config.json

The screenshot shows the GitHub repository page for `google/gemma-3-270m`. The repository is owned by `google` and has 49.3k followers. It is categorized under `Text Generation`, `Transformers`, `Safetensors`, `gemma3_text`, and `gemma`. The `Files and versions` tab is selected, showing the `main` branch with a file size of 575 MB. A commit by `osanseviero` is shown, titled "Update README.md" with commit hash `9b0fec` and a `VERIFIED` badge. The file list includes `.gitattributes` (Safe), `README.md`, `added_tokens.json` (Safe), `config.json` (highlighted with an orange circle), `generation_config.json`, and `model.safetensors` (Safe).

4

Check config.json

google/gemma-3-270m

Text Generation Transformers Safetensors gemma3_text gemma

Model card Files and versions xet Community 32

main gemma-3-270m 575 MB

osanseviero Update README.md 9b0cfec VERIFIED

- .gitattributes Safe
- README.md
- added_tokens.json Safe
- config.json**
- generation_config.json
- model.safetensors Safe

```
main gemma-3-270m / config.json
sindhuraguram uploading the PT weights for Gemma 3 270m 603ad58
raw Copy download link history blame contribute delete
1 {
2   "_sliding_window_pattern": 6,
3   "architectures": [
4     "Gemma3ForCausalLM"
5   ],
6   "attention_bias": false,
7   "attention_dropout": 0.0,
8   "attn_logit_softcapping": null,
9   "bos_token_id": 2,
10  "eos_token_id": 1,
11  "final_logit_softcapping": null,
12  "head_dim": 256,
13  "hidden_activation": "gelu_pytorch_tanh",
14  "hidden_size": 640,
15  "initializer_range": 0.02,
16  "intermediate_size": 2048,
17  "layer_types": [
18    "sliding_attention",
19    "sliding_attention",
20    "sliding_attention",
21    "sliding_attention",
22    "sliding_attention",
23    "full_attention",
24    "sliding_attention",
25    "sliding_attention",
26    "sliding_attention",
27    "sliding_attention",
28    "sliding_attention",
29    "full_attention",
30    "sliding_attention"
```

5

Compare with other architectures

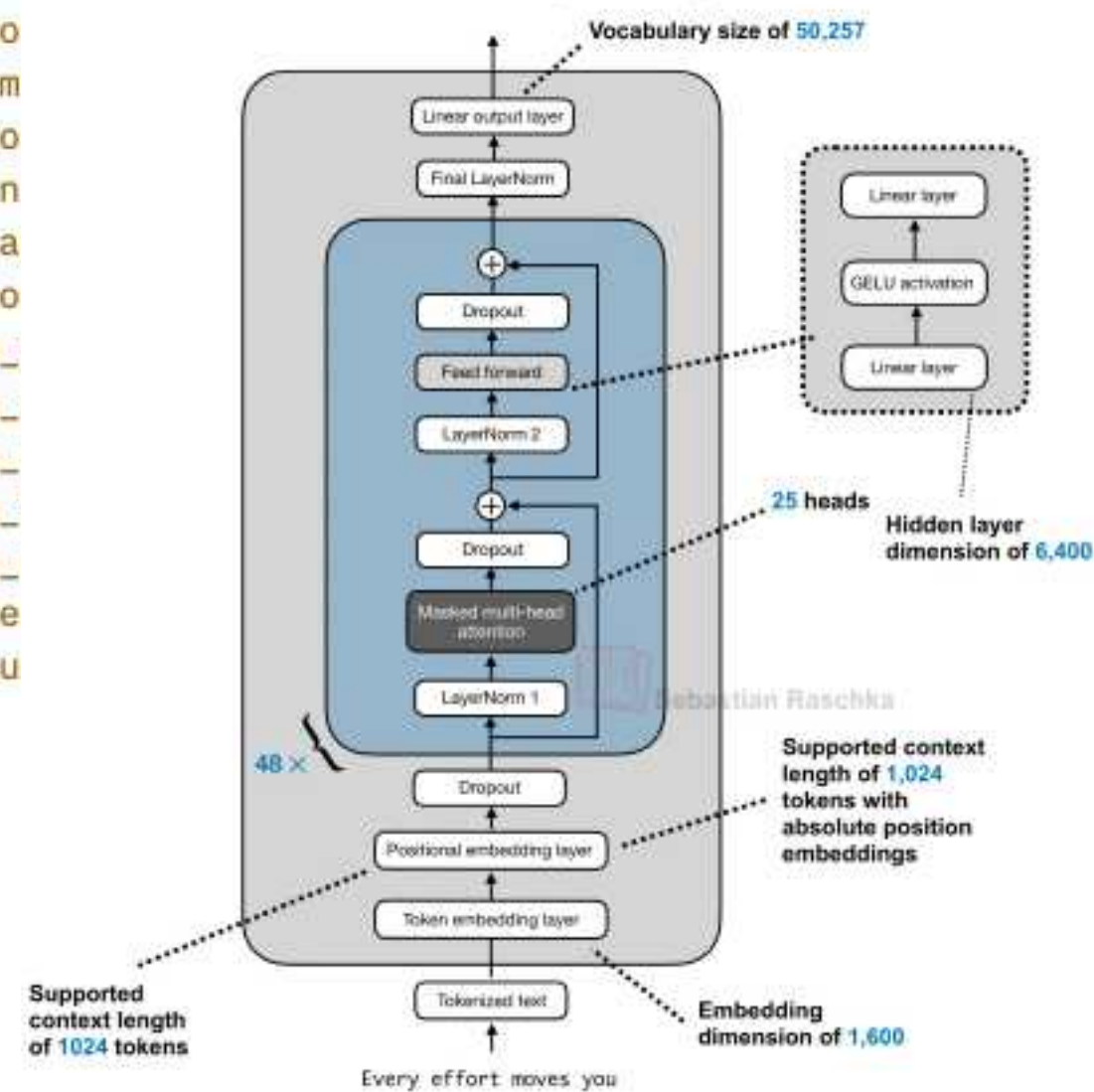
main gpt2 / config.json

system HF Staff Update config.json 7979bb7

raw Copy download link history blame contribute

```
1 {
2   "activation_function": "gelu_new",
3   "architectures": [
4     "GPT2LMHeadModel"
5   ],
6   "attn_implementation": "scaled_dot_product_attention",
7   "bos_token_id": 50256,
8   "eos_token_id": 50256,
9   "initializer_range": 0.02,
10  "layer_norm_epsilon": 1e-05,
11  "max_position_embeddings": 1024,
12  "model_max_length": 1024,
13  "n_ctx": 1024,
14  "n_embd": 1600,
15  "n_head": 25,
16  "n_inner": 6400,
17  "n_layer": 48,
18  "reorder_and_upcast_attn": true,
19  "tokenizer_type": "GPT2TokenizerFast"
}
```

GPT-2 XL (1.5B)



5

Compare with other architectures

main gpt2 / config.json

system HF Staff Update config.json 7979bb7

```
1 {
2   "activation_function": "gelu_new",
3   "architectures": [
4     "GPT2LMHeadModel"
5   ],
6   "at
7   "bo
8   "em
9   "eo
10  "in
11  "la
12  "mo
13  "n_
14  "n_
15  "n_
16  "n_
17  "n_
18  "re
19  "su
```

GPT-2 XL (1.5B)

Vocabulary size of 50,257

25 heads

Hidden layer dimension of 6,400

Supported context length of 1,024 tokens with absolute position embeddings

Supported context length of 1024 tokens

Embedding dimension of 1,600

Every effort moves you

main Qwen3-8B / config.json

littlebird13 Upload folder using huggingface_hub 47719a

```
1 {
2   "architectures": [
3     "Qwen3ForCausalLM"
4   ],
5   "attention_bias": false,
6   "attenti
7   "bos_tok
8   "eos_tok
9   "head_di
10  "hidden_
11  "hidden_
12  "initial
13  "interme
14  "max_pos
15  "max_wir
16  "model_t
17  "num_att
18  "num_hic
19  "num_key
```

Qwen3 (8B)

Vocabulary size of 151k

Supported context length of 128k tokens

36 x

Input size: 4,096

Intermediate projection size 3x4,096 = 12,288

32 attention heads

8 key & value heads

Embedding dimension of 4,096

main gemma-3-270m / config.json

sindhuraguram uploading the PT weights for Gemma 3 2'

```
1 {
2   "_sliding_window_pattern": 6,
3   "architectures": [
4     "Gemma3ForCausalLM"
5   ],
6   "attention_bias": false,
7   "a
8   "a
9   "b
10  "e
11  "f
12  "h
13  "h
14  "h
15  "i
16  "i
17  "l
```

Gemma 3 (270M)

Vocabulary size of 262k

Supported context length of 128k tokens

18 x

Input size: 640

Intermediate projection size = 2048

4 attention heads

1 key & value heads (technically multi-query attention)

Embedding dimension of 640

6 Code up the architecture (starting from a similar architecture)

```
1 {
2   "architectures": [
3     "Qwen3ForCausalLM"
4   ],
5   "attention_bias": false,
6   "attention_dropout": 0.0,
7   "bos_token_id": 151643,
8   "eos_token_id": 151645,
9   "head_dim": 128,
10  "hidden_act": "silu",
11  "hidden_size": 4096,
```

Related
architecture
(as template)

main LLMs-from-scratch / ch05 / 11_qwen3 / standalone-qwen3.ipynb

Preview Code Blame 1238 lines (1238 loc) · 45.1 KB

1. Architecture code

In [24]:

```
import torch
import torch.nn as nn

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.fc1 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc2 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc3 = nn.Linear(cfg["hidden_dim"], cfg["emb_dim"], dtype=cfg["dtype"], bias=False)

    def forward(self, x):
        x_fc1 = self.fc1(x)
        x_fc2 = self.fc2(x)
        x = nn.functional.silu(x_fc1) * x_fc2
        return self.fc3(x)
```

6

Code up the architecture (starting from a similar architecture)

```
1 {
2   "architectures": [
3     "Qwen3ForCausalLM"
4   ],
5   "attention_bias": false,
6   "attention_dropout": 0.0,
7   "bos_token_id": 151643,
8   "eos_token_id": 151645,
9   "head_dim": 128,
10  "hidden_act": "silu",
11  "hidden_size": 4096,
```

Related architecture (as template)

```
1 {
2   "_sliding_window_pattern": 6,
3   "architectures": [
4     "Gemma3ForCausalLM"
5   ],
6   "attention_bias": false,
7   "attention_dropout": 0.0,
8   "attn_logit_softcapping": null,
9   "bos_token_id": 2,
10  "eos_token_id": 1,
11  "final_logit_softcapping": null,
12  "head_dim": 256,
13  "hidden_activation": "gelu_pytorch_tanh",
14  "hidden_size": 640,
```

New architecture to implement

main LLMs-from-scratch / ch05 / 11_qwen3 / standalone-qwen3.ipynb

Preview Code Blame 1238 lines (1238 loc) · 45.1 KB

1. Architecture code

```
In [24]: import torch
import torch.nn as nn

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.fc1 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc2 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc3 = nn.Linear(cfg["hidden_dim"], cfg["emb_dim"], dtype=cfg["dtype"], bias=False)

    def forward(self, x):
        x_fc1 = self.fc1(x)
        x_fc2 = self.fc2(x)
        x = nn.functional.silu(x_fc1) * x_fc2
        return self.fc3(x)
```

main LLMs-from-scratch / ch05 / 12_gemma3 / standalone-gemma3.ipynb

Preview Code Blame 1231 lines (1231 loc) · 43.5 KB

1. Architecture code

```
In [4]: import torch
import torch.nn as nn

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.fc1 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc2 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc3 = nn.Linear(cfg["hidden_dim"], cfg["emb_dim"], dtype=cfg["dtype"], bias=False)

    def forward(self, x):
        x_fc1 = self.fc1(x)
        x_fc2 = self.fc2(x)
        x = nn.functional.gelu(x_fc1, approximate="tanh") * x_fc2
        return self.fc3(x)
```

7

Check if it generates coherent text

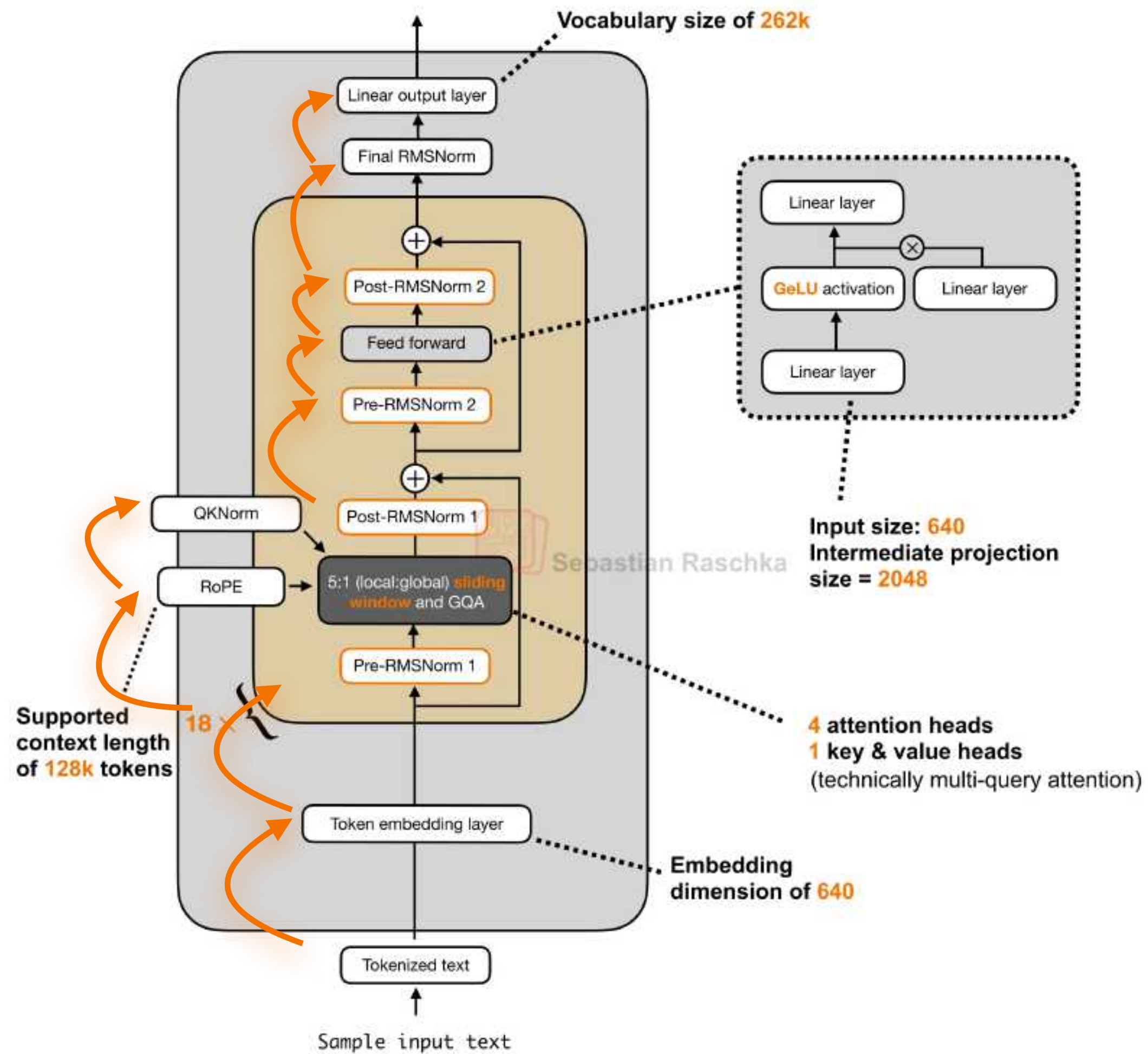
```
prompt = "Give me a short introduction to large language models."

for token in generate_text_basic_stream(
    model=model,
    token_ids=input_token_ids_tensor,
    max_new_tokens=500,
    eos_token_id=tokenizer.end_of_turn_token_id
):
    token_id = token.squeeze(0).tolist()
    print(
        tokenizer.decode(token_id),
        end="",
        flush=True
    )
```


8

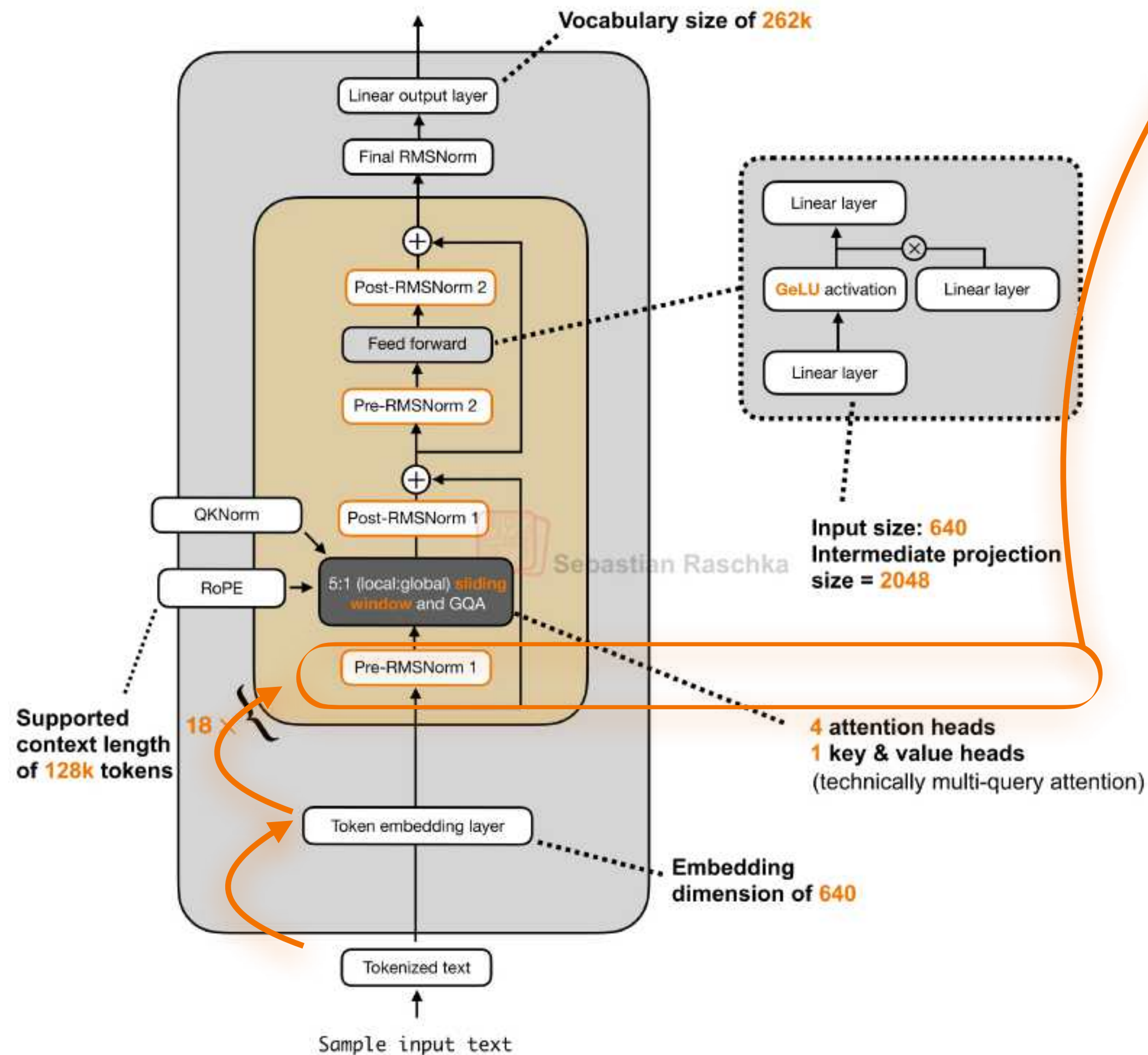
Check layer by layer to find the mistake

Gemma 3 (270M)



Check layer by layer to find the mistake

Gemma 3 (270M)



```

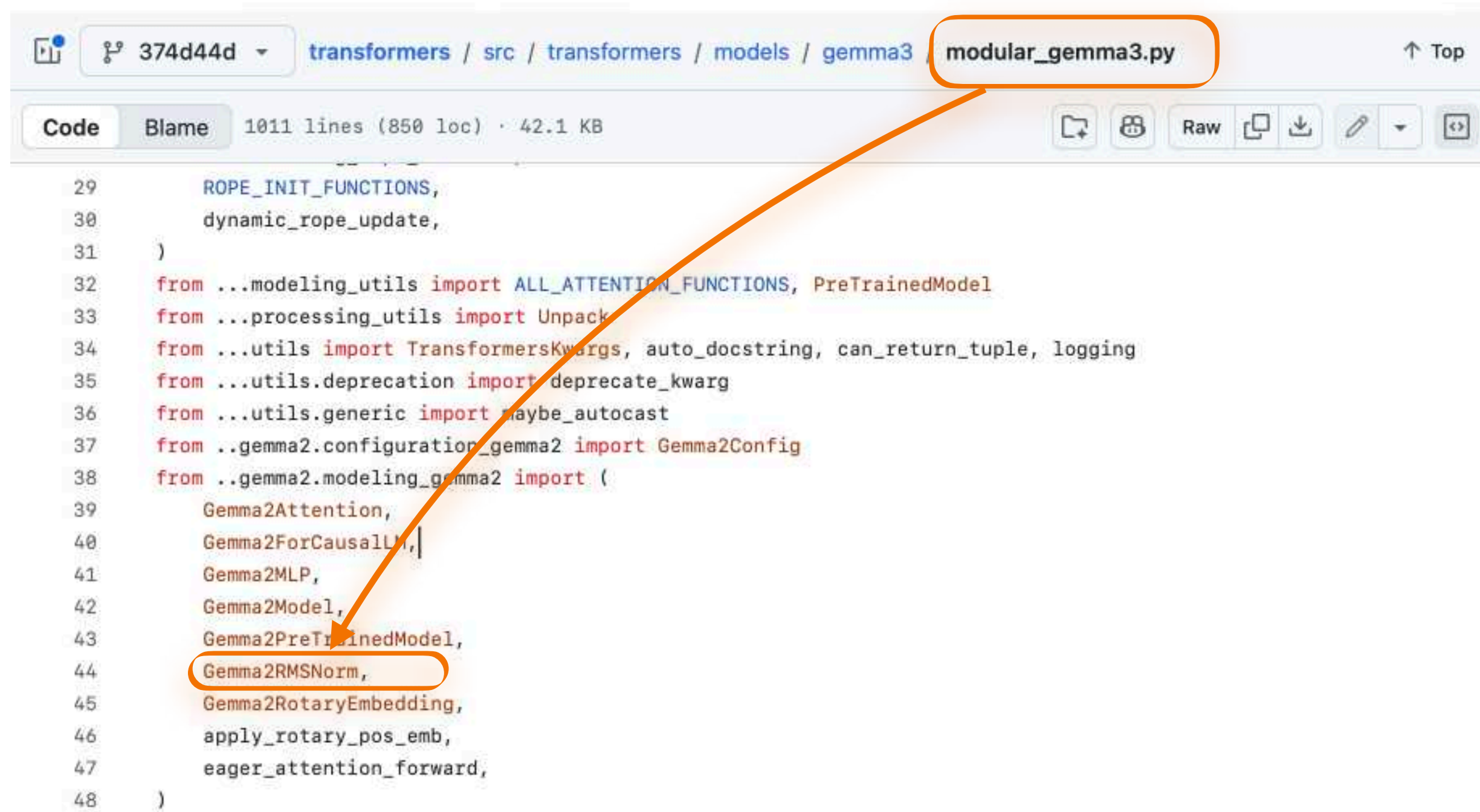
12_gemma3 — sebastian@Mac — ..h05/12_gemma3 — -zsh — 107x37
→ 12_gemma3 git:(main) * uv run tests/gemma3_layer_debugger-v1.py
[[OK] embedding: max=0.00e+00, mean=0.00e+00
[DIFF] block_0: max=2.98e+00, mean=8.80e-01
[DIFF] block_0.input_layernorm: max=2.34e+00, mean=8.11e-01
[DIFF] block_0.att.q_proj: max=3.45e-01, mean=9.00e-02
[DIFF] block_0.att.k_proj: max=2.99e-01, mean=9.04e-02
[DIFF] block_0.att.v_proj: max=3.84e-01, mean=8.81e-02
[DIFF] block_0.att.q_norm: max=2.42e+00, mean=8.18e-01
[DIFF] block_0.att.k_norm: max=2.17e+00, mean=8.26e-01
[DIFF] block_0.att.o_proj: max=5.82e-02, mean=5.32e-03
[DIFF] block_0.att: max=5.82e-02, mean=5.32e-03
[DIFF] block_0.post_attention_layernorm: max=3.02e+00, mean=7.91e-01
[DIFF] block_0.pre_feedforward_layernorm: max=2.97e+00, mean=8.18e-01
[DIFF] block_0.ff.gate_proj: max=3.32e-01, mean=8.97e-02
[DIFF] block_0.ff.up_proj: max=3.29e-01, mean=8.94e-02
[DIFF] block_0.ff.down_proj: max=2.14e-03, mean=6.63e-04
[DIFF] block_0.ff: max=2.14e-03, mean=6.63e-04
[DIFF] block_0.post_feedforward_layernorm: max=1.54e+00, mean=5.08e-01
[DIFF] block_1: max=4.13e+00, mean=1.29e+00
[DIFF] block_1.input_layernorm: max=2.79e+00, mean=8.04e-01
[DIFF] block_1.att.q_proj: max=3.82e-01, mean=1.05e-01
[DIFF] block_1.att.k_proj: max=3.35e-01, mean=1.05e-01
[DIFF] block_1.att.v_proj: max=3.37e-01, mean=9.21e-02
[DIFF] block_1.att.q_norm: max=2.21e+00, mean=8.13e-01
[DIFF] block_1.att.k_norm: max=2.29e+00, mean=8.40e-01
[DIFF] block_1.att.o_proj: max=3.15e-02, mean=7.69e-03
[DIFF] block_1.att: max=3.15e-02, mean=7.69e-03
[DIFF] block_1.post_attention_layernorm: max=2.60e+00, mean=7.89e-01
[DIFF] block_1.pre_feedforward_layernorm: max=3.02e+00, mean=7.75e-01
[DIFF] block_1.ff.gate_proj: max=5.03e-01, mean=9.30e-02
[DIFF] block_1.ff.up_proj: max=3.09e-01, mean=9.14e-02
[DIFF] block_1.ff.down_proj: max=2.95e-03, mean=8.99e-04
[DIFF] block_1.ff: max=2.95e-03, mean=8.99e-04
[DIFF] block_1.post_feedforward_layernorm: max=1.80e+00, mean=6.04e-01
[DIFF] final_norm: max=2.52e+00, mean=8.13e-01
[DIFF] logits: max=4.05e-01, mean=9.47e-02
→ 12_gemma3 git:(main) *

```

*Against HF transformers reference implementation;
small config with random weights

9

Read reference implementation code



```
374d44d transformers / src / transformers / models / gemma3 modular_gemma3.py ↑ Top
Code Blame 1011 lines (850 loc) · 42.1 KB
29     ROPE_INIT_FUNCTIONS,
30     dynamic_rope_update,
31 )
32 from ...modeling_utils import ALL_ATTENTION_FUNCTIONS, PreTrainedModel
33 from ...processing_utils import Unpack
34 from ...utils import TransformersKwargs, auto_docstring, can_return_tuple, logging
35 from ...utils.deprecation import deprecate_kwarg
36 from ...utils.generic import maybe_autocast
37 from ..gemma2.configuration_gemma2 import Gemma2Config
38 from ..gemma2.modeling_gemma2 import (
39     Gemma2Attention,
40     Gemma2ForCausalLM,
41     Gemma2MLP,
42     Gemma2Model,
43     Gemma2PreTrainedModel,
44     Gemma2RMSNorm,
45     Gemma2RotaryEmbedding,
46     apply_rotary_pos_emb,
47     eager_attention_forward,
48 )
```

...sometimes a bit like an Easter egg hunt

9

Read reference implementation code

```
transformers / src / transformers / models / gemma3 / modular_gemma3.py
Code Blame 1011 lines (850 loc) · 42.1 KB
29     ROPE_INIT_FUNCTIONS,
30     dynamic_rope_update,
31 )
32 from ...modeling_utils import ALL_ATTENTION_FUNCTIONS, PreTrainedModel
33 from ...processing_utils import Unpack
34 from ...utils import TransformersKwargs, auto_docstring, can_return_tuple, logging
35 from ...utils.deprecation import deprecate_kwarg
36 from ...utils.generic import maybe_autocast
37 from ..gemma2.configuration_gemma2 import Gemma2Config
38 from ..gemma2.modeling_gemma2 import (
39     Gemma2Attention,
40     Gemma2ForCausalLM,
41     Gemma2MLP,
42     Gemma2Model,
43     Gemma2PreTrainedModel,
44     Gemma2RMSNorm,
45     Gemma2RotaryEmbedding,
46     apply_rotary_pos_emb,
47     eager_attention_forward,
48 )
135     )
136
137
138 class Gemma2RMSNorm(GemmaRMSNorm):
139     pass
140
141
142 class Gemma2MLP(GemmaMLP):
143     def __init__(self, config):
144         super().__init__(config)
145         self.act_fn = ACT2FN[config.hidden_activation]
146
```

...sometimes a bit like an Easter egg hunt

Read reference implementation code

```
transformers / src / transformers / models / gemma3 / modular_gemma3.py
Code Blame 1011 lines (850 loc) · 42.1 KB

29     ROPE_INIT_FUNCTIONS,
30     dynamic_rope_update,
31 )
32 from ...modeling_utils import ALL_ATTENTION_FUNCTIONS, PreTrainedModel
33 from ...processing_utils import Unpack
34 from ...utils import TransformersKwargs, auto_docstring, can_return_tuple, logging
35 from ...utils.deprecation import deprecate_kwarg
36 from ...utils.generic import maybe_autocast
37 from ..gemma2.configuration_gemma2 import Gemma2Config
38 from ..gemma2.modeling_gemma2 import (
39     Gemma2Attention,
40     Gemma2ForCausalLM,
41     Gemma2MLP,
42     Gemma2Model,
43     Gemma2PreTrainedModel,
44     Gemma2RMSNorm,
45     Gemma2RotaryEmbedding,
46     apply_rotary_pos_emb,
47     eager_attention_forward,
48 )

135     )
136
137
138 class Gemma2RMSNorm(GemmaRMSNorm):
139     pass
140
141
142 class Gemma2MLP(GemmaMLP):
143     def __init__(self, config):
144         super().__init__(config)
145         self.act_fn = ACT2FN[config.hidden_activation]
146
147
148 class GemmaRMSNorm(nn.Module):
149     def __init__(self, dim: int, eps: float = 1e-6):
150         super().__init__()
151         self.eps = eps
152         self.weight = nn.Parameter(torch.zeros(dim))
153
154     def _norm(self, x):
155         return x * torch.rsqrt(x.pow(2).mean(-1, keepdim=True) + self.eps)
156
157     def forward(self, x):
158         output = self._norm(x.float())
159         # llama does x.to(float16) * w whilst Gemma is (x * w).to(float16)
160         # See https://github.com/huggingface/transformers/pull/29402
161         output = output * (1.0 + self.weight.float())
162         return output.type_as(x)
163
164     def extra_repr(self):
165         return f"tuple(self.weight.shape), eps={self.eps}"
166
167
```

...sometimes a bit like an Easter egg hunt

Implement the architecture specifics

main ▾ LLMs-from-scratch / ch05 / 11_qwen3 / standalone-qwen3.ipynb

Preview Code Blame 1238 lines (1238 loc) · 45.1 KB

1. Architecture code

```
In [24]: import torch
import torch.nn as nn

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.fc1 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc2 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc3 = nn.Linear(cfg["hidden_dim"], cfg["emb_dim"], dtype=cfg["dtype"], bias=False)

    def forward(self, x):
        x_fc1 = self.fc1(x)
        x_fc2 = self.fc2(x)
        x = nn.functional.silu(x_fc1) * x_fc2
        return self.fc3(x)
```

```
In [25]: class RMSNorm(nn.Module):
    def __init__(self, emb_dim, eps=1e-6, bias=False, qwen3_compatible=True):
        super().__init__()
        self.eps = eps
        self.qwen3_compatible = qwen3_compatible
        self.scale = nn.Parameter(torch.ones(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim)) if bias else None

    def forward(self, x):
        input_dtype = x.dtype

        if self.qwen3_compatible:
            x = x.to(torch.float32)

        variance = x.pow(2).mean(dim=-1, keepdim=True)
        norm_x = x * torch.rsqrt(variance + self.eps)
        norm_x = norm_x * self.scale

        if self.shift is not None:
            norm_x = norm_x + self.shift

        return norm_x.to(input_dtype)
```

main ▾ LLMs-from-scratch / ch05 / 12_gemma3 / standalone-gemma3.ipynb

Preview Code Blame 1231 lines (1231 loc) · 43.5 KB

1. Architecture code

```
In [4]: import torch
import torch.nn as nn

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.fc1 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc2 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc3 = nn.Linear(cfg["hidden_dim"], cfg["emb_dim"], dtype=cfg["dtype"], bias=False)

    def forward(self, x):
        x_fc1 = self.fc1(x)
        x_fc2 = self.fc2(x)
        x = nn.functional.gelu(x_fc1, approximate="tanh") * x_fc2
        return self.fc3(x)
```

```
In [5]: class RMSNorm(nn.Module):
    def __init__(self, emb_dim, eps=1e-6, bias=False):
        super().__init__()
        self.eps = eps
        # Gemma3 stores zero-centered weights and uses (1 + weight) during forward
        self.scale = nn.Parameter(torch.zeros(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim)) if bias else None

    def forward(self, x):
        # Match HF Gemma3: compute norm in float32, then scale by (1 + w)
        input_dtype = x.dtype
        x_f = x.float()
        var = x_f.pow(2).mean(dim=-1, keepdim=True)
        x_norm = x_f * torch.rsqrt(var + self.eps)
        out = x_norm * (1.0 + self.scale.float())

        if self.shift is not None:
            out = out + self.shift.float()

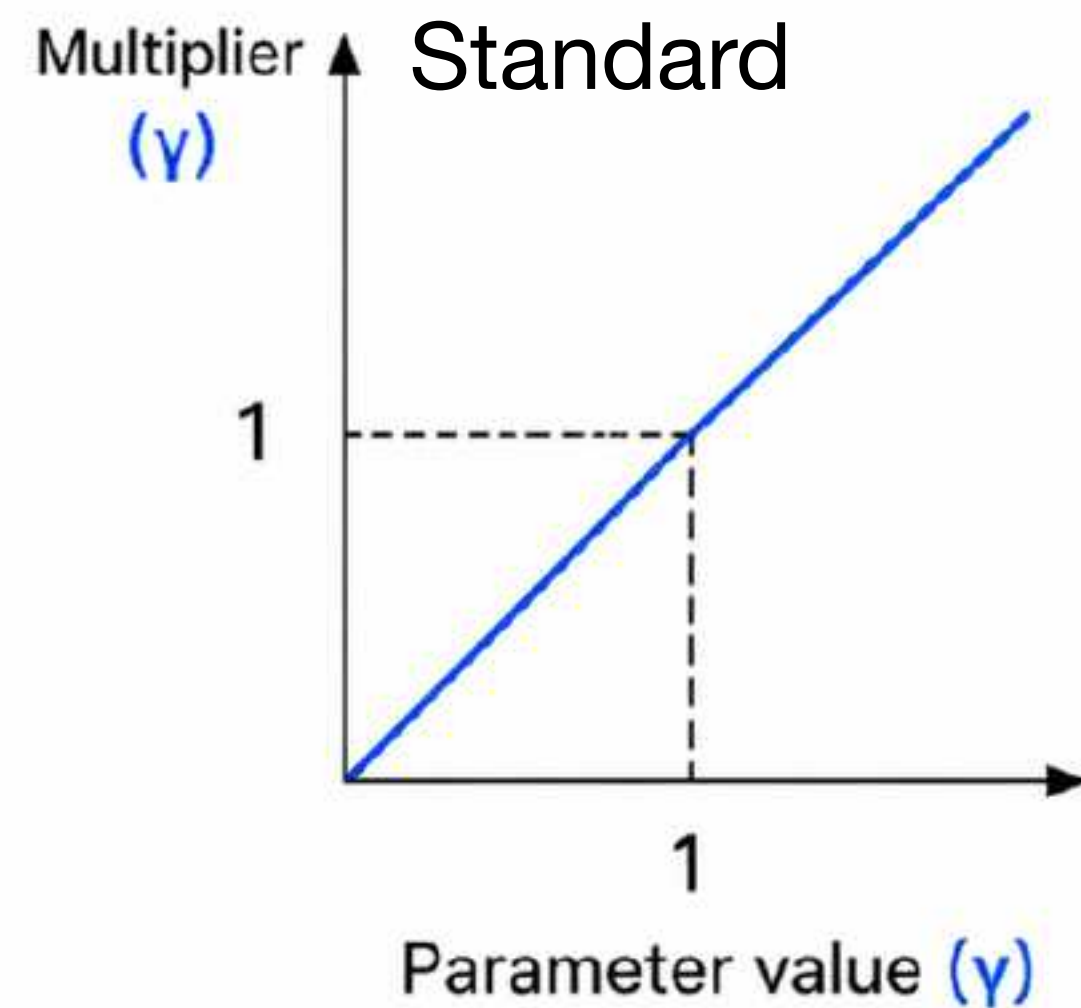
        return out.to(input_dtype)
```

adjust & "learn"

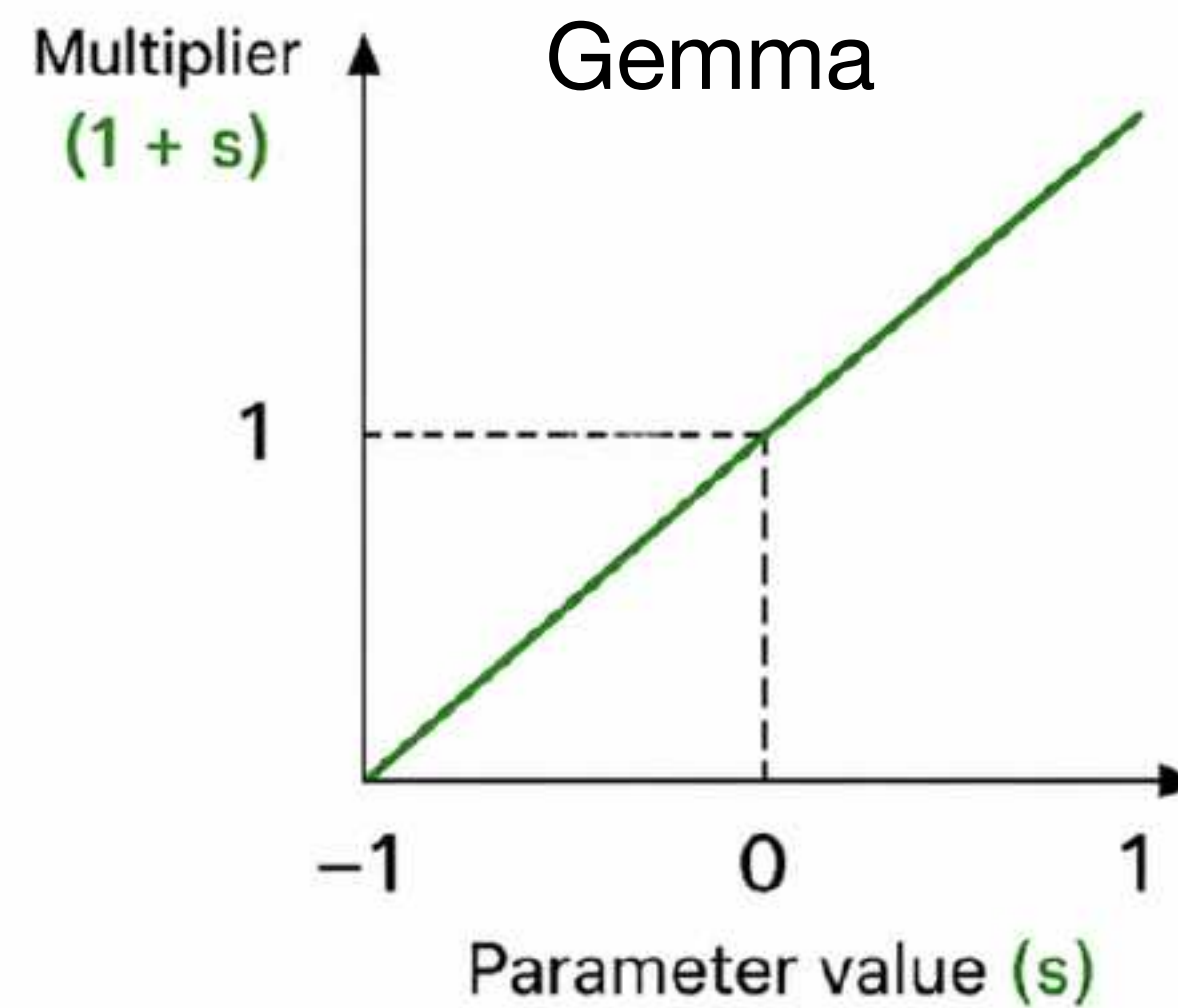


Side note:

- standard RMSNorm learns the scale directly
- Gemma RMSNorm learns an offset from 1 (zero-centered, more stable learning)



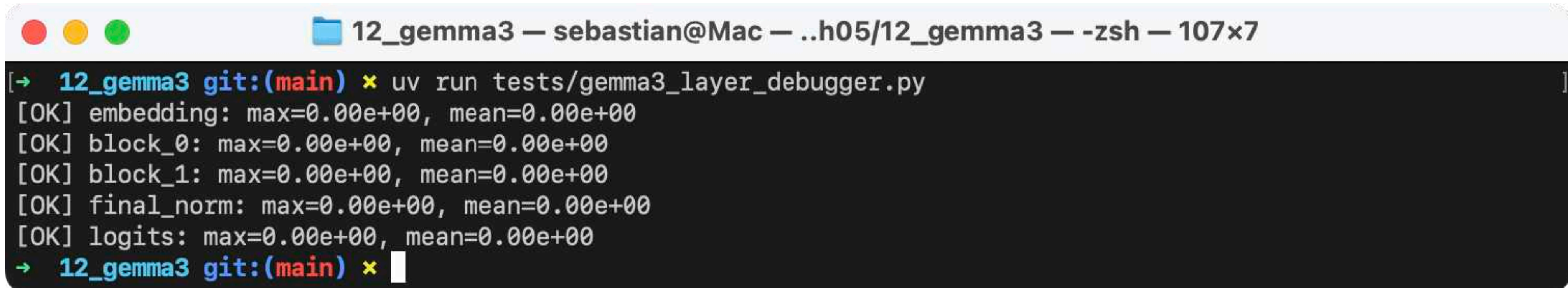
$$y = (x / \text{rms}(x)) * \text{scale}$$



$$y = (x / \text{rms}(x)) * (1 + \text{scale})$$

11

Run layer check until everything matches



```
12_gemma3 — sebastian@Mac — ..h05/12_gemma3 — -zsh — 107x7
[→ 12_gemma3 git:(main) ✕ uv run tests/gemma3_layer_debugger.py
[OK] embedding: max=0.00e+00, mean=0.00e+00
[OK] block_0: max=0.00e+00, mean=0.00e+00
[OK] block_1: max=0.00e+00, mean=0.00e+00
[OK] final_norm: max=0.00e+00, mean=0.00e+00
[OK] logits: max=0.00e+00, mean=0.00e+00
[→ 12_gemma3 git:(main) ✕ ]
```

*Against HF transformers reference implementation;
small config with random weights

Check if model generates same text as reference

```
LLMs-from-scratch / ch05 / 12_gemma3 / standalone-gemma3.ipynb
```

1. Architecture code

```
import torch
import torch.nn as nn

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.fc1 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc2 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc3 = nn.Linear(cfg["hidden_dim"], cfg["emb_dim"], dtype=cfg["dtype"], bias=False)

    def forward(self, x):
        x_fc1 = self.fc1(x)
        x_fc2 = self.fc2(x)
        x = nn.functional.gelu(x_fc1, approximate="tanh") * x_fc2
        return self.fc3(x)
```

```
class RMSNorm(nn.Module):
    def __init__(self, emb_dim, eps=1e-6, bias=False):
        super().__init__()
        self.eps = eps
        # Gemma3 stores zero-centered weights and uses (1 + weight) during forward
        self.scale = nn.Parameter(torch.zeros(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim)) if bias else None

    def forward(self, x):
```

```
prompt = "Give me a short introduction to large language models."
```

```
for token in generate_text_basic_stream(
    model=model,
    token_ids=input_token_ids_tensor,
    max_new_tokens=500,
    eos_token_id=tokenizer.end_of_turn_token_id
):
    token_id = token.squeeze(0).tolist()
    print(
        tokenizer.decode(token_id),
        end="",
        flush=True
    )
```


Large language models (LLMs) are sophisticated artificial intelligence systems that can understand, generate, and manipulate human language. They are trained on massive amounts of text data to learn patterns and relationships within language, enabling them to perform a wide range of tasks, from writing articles and answering questions to translating languages and summarizing information.

*Against HF transformers reference implementation; now the full architecture with pre-trained weights

```
from transformers import AutoModelForCausalLM, AutoTokenizer

model_id = "google/gemma-3-270m-it"
prompt = "Give me a short introduction to large language models."

tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(model_id)
model.generation_config.do_sample = False
model.generation_config.top_p = None
model.generation_config.top_k = None
model.generation_config.pad_token_id = tokenizer.pad_token_id
model.eval();
```

Loading weights: 100%  236/236 [00:00<00:00, 3884.39it/s, Materializing param=model.norm.weight]

```
messages = [{"role": "user", "content": prompt}]

inputs = tokenizer.apply_chat_template(
    messages,
    tokenize=True,
    add_generation_prompt=True,
    return_tensors="pt",
)

outputs = model.generate(
    **inputs,
    max_new_tokens=500,
    do_sample=False,
    num_beams=1,
    pad_token_id=tokenizer.pad_token_id,
)

response = tokenizer.decode(
    outputs[0][inputs["input_ids"].shape[-1]:],
    skip_special_tokens=True,
)

print(response)
```

Large language models (LLMs) are sophisticated artificial intelligence systems that can understand, generate, and manipulate human language. They are trained on massive amounts of text data to learn patterns and relationships within language, enabling them to perform a wide range of tasks, from writing articles and answering questions to translating languages and summarizing information.

12

Check if model generates same text as reference

```
LLMs-from-scratch / ch05 / 12_gemma3 / standalone-gemma3.ipynb
```

1. Architecture code

```
import torch
import torch.nn as nn

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.fc1 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc2 = nn.Linear(cfg["emb_dim"], cfg["hidden_dim"], dtype=cfg["dtype"], bias=False)
        self.fc3 = nn.Linear(cfg["hidden_dim"], cfg["emb_dim"], dtype=cfg["dtype"], bias=False)

    def forward(self, x):
        x_fc1 = self.fc1(x)
        x_fc2 = self.fc2(x)
        x = nn.functional.gelu(x_fc1, approximate="tanh") * x_fc2
        return self.fc3(x)
```

```
class RMSNorm(nn.Module):
    def __init__(self, emb_dim, eps=1e-6, bias=False):
        super().__init__()
        self.eps = eps
        # Gemma3 stores zero-centered weights and uses (1 + weight) during forward
        self.scale = nn.Parameter(torch.zeros(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim)) if bias else None

    def forward(self, x):
```

```
prompt = "Give me a short introduction to large language models."
```

```
for token in generate_text_basic_stream(
    model=model,
    token_ids=input_token_ids_tensor,
    max_new_tokens=500,
    eos_token_id=tokenizer.end_of_turn_token_id
):
    token_id = token.squeeze(0).tolist()
    print(
        tokenizer.decode(token_id),
        end="",
        flush=True
    )
```

Large language models (LLMs) are sophisticated artificial intelligence systems that can understand, generate, and manipulate human language. They are trained on massive amounts of text data to learn patterns and relationships within language, enabling them to perform a wide range of tasks, from writing articles and answering questions to translating languages and summarizing information.

*Against HF transformers reference implementation; now the full architecture with pre-trained weights

```
from transformers import AutoModelForCausalLM, AutoTokenizer

model_id = "google/gemma-3-270m-it"
prompt = "Give me a short introduction to large language models."

tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(model_id)
model.generation_config.do_sample = False
model.generation_config.top_p = None
model.generation_config.top_k = None
model.generation_config.pad_token_id = tokenizer.pad_token_id
model.eval();
```

Loading weights: 100% 236/236 [00:00<00:00, 3884.39it/s, Materializing param=model.norm.weight]

```
messages = [{"role": "user", "content": prompt}]

inputs = tokenizer.apply_chat_template(
    messages,
    tokenize=True,
    add_generation_prompt=True,
    return_tensors="pt",
)

outputs = model.generate(
    **inputs,
    max_new_tokens=500,
    do_sample=False,
    num_beams=1,
    pad_token_id=tokenizer.pad_token_id,
)

response = tokenizer.decode(
    outputs[0][inputs["input_ids"].shape[-1]:],
    skip_special_tokens=True,
)

print(response)
```

Large language models (LLMs) are sophisticated artificial intelligence systems that can understand, generate, and manipulate human language. They are trained on massive amounts of text data to learn patterns and relationships within language, enabling them to perform a wide range of tasks, from writing articles and answering questions to translating languages and summarizing information.

From Code to Gallery

<https://llm-gallery.com/>



Architecture summaries

DeepSeek V3.2 (671B)

Vocabulary size of 128k

Supported context length of 128k tokens

First 3 blocks use dense FFN with hidden size 18,432 instead of MoE

Sample input text

Token embedding layer

Embedding dimension of 7,168

128 heads

MoE layer

Router

FeedForward (SwiGLU) module

Intermediate hidden layer dimension of 2,048

Resource savings:

- Model size is 671B
- but only 1 (shared) = 8 experts active per token
- only 37B parameters are active per inference step

COMPARE

Model A Model B

DeepSeek V3.2 (671B)

View in article [config.json](#) License

Tech report

DeepSeek's successor keeps the V3 template but adds sparse attention to cut long-context costs.

SCALE

671B total, 37B active (5.5% active)

CONTEXT (TOKENS)

128,000

LICENSE

MIT License

DATE

2025-12-01

DECODER TYPE

Sparse MoE

ATTENTION

MLA with DeepSeek Sparse Attention

LAYER MIX

61 MLA

KV CACHE / TOKEN

(BF16) [Info](#)
68.6 KiB · Low

KEY DETAIL

An evolutionary update focused on efficiency rather than a new base layout.

AA INTELLIGENCE INDEX [Info](#)

Total score 32.1

GENERAL
29.7

SCIENTIFIC
24.2

CODING
34.6

AGENTS
39.8

Show less ^

RELATED CONCEPTS

[MLA](#)

[MoE](#)

[DeepSeek Sparse Attention](#)

<https://lm-gallery.com/>

Architecture comparisons

ARCHITECTURE DIFF TOOL

Select two models to compare their architectures

If you want to compare two architectures side by side instead of browsing the gallery, use this diff tool. You can use the selectors here or the **Model A** / **Model B** actions on each card.

MODEL A: GPT-2 XL (1.5B) MODEL B: Qwen3 (4B)

Swap Clear

MODEL A
GPT-2 XL (1.5B)

MODEL B
Qwen3 (4B)

ATTENTION BLOCK Different	DECODER TYPE Shared
MODEL A GPT-2 XL (1.5B) MHA with learned absolute positional embeddings	MODEL B Qwen3 (4B) GQA with QK-Norm
MODEL A GPT-2 XL (1.5B) 300 KiB · High	MODEL B Qwen3 (4B) 144 KiB · Moderate
MODEL A GPT-2 XL (1.5B) 48 MHA	MODEL B Qwen3 (4B) 36 GQA
SCALE Different	CONTEXT (TOKENS) Different
MODEL A GPT-2 XL (1.5B) 1.5B parameters	MODEL B Qwen3 (4B) 4B parameters
	MODEL A GPT-2 XL (1.5B) 1,024
	MODEL B Qwen3 (4B) 32,768

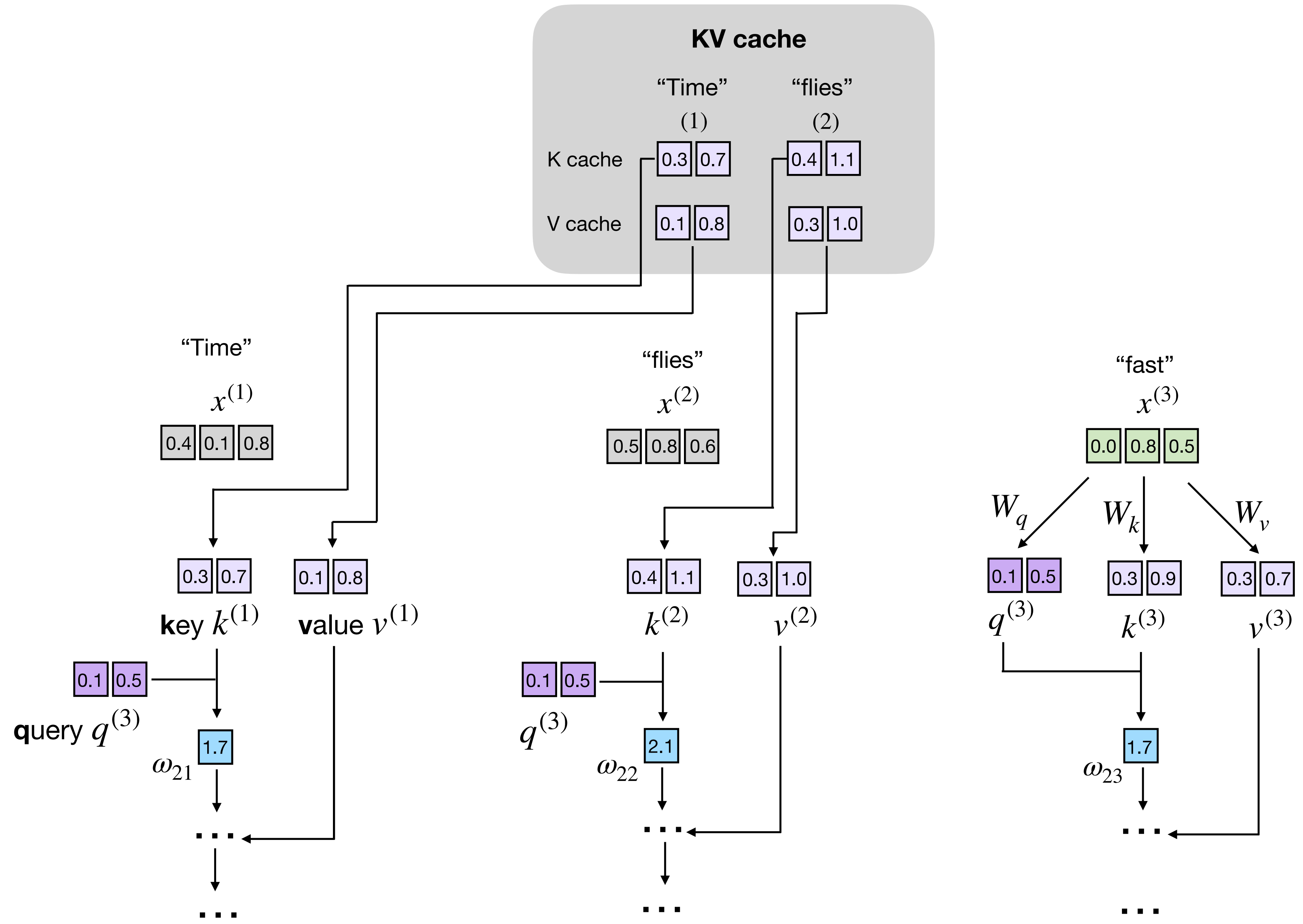
Lessons and trends from 50+ models



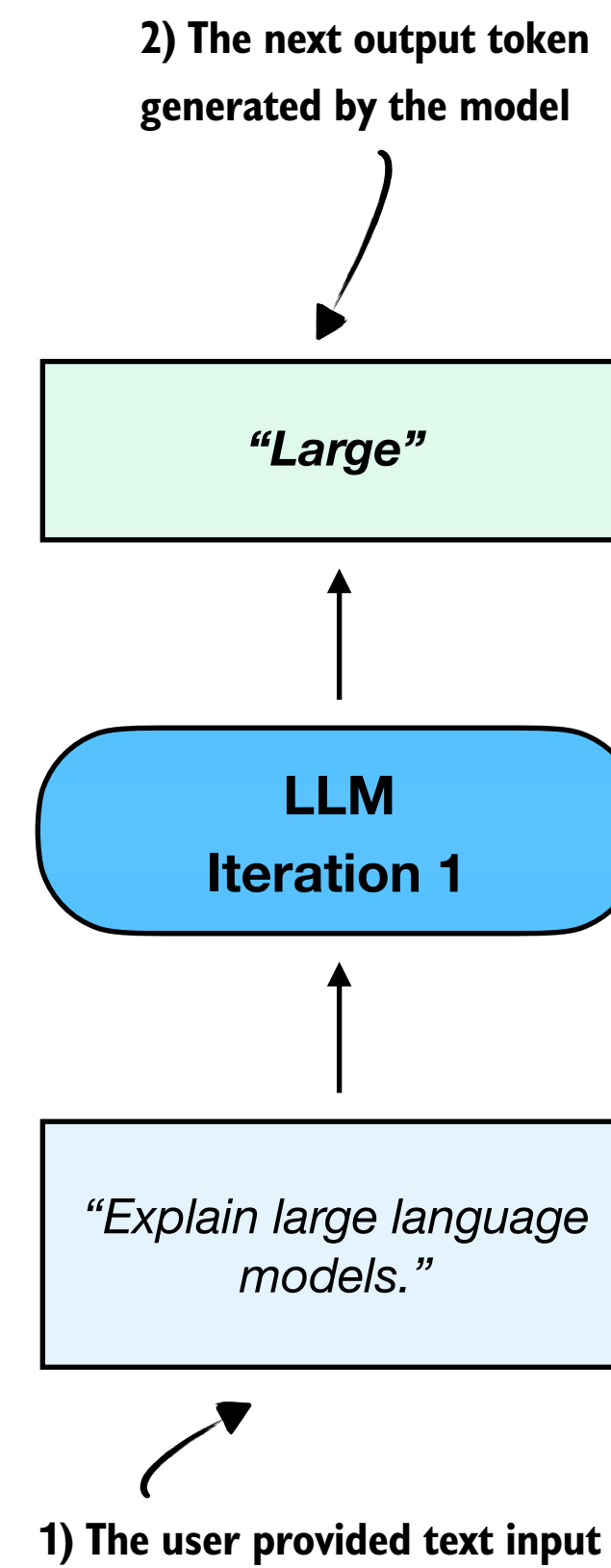
**Main focus: strategies for
reducing KV cache size**

**Main focus: strategies for
reducing KV cache size**

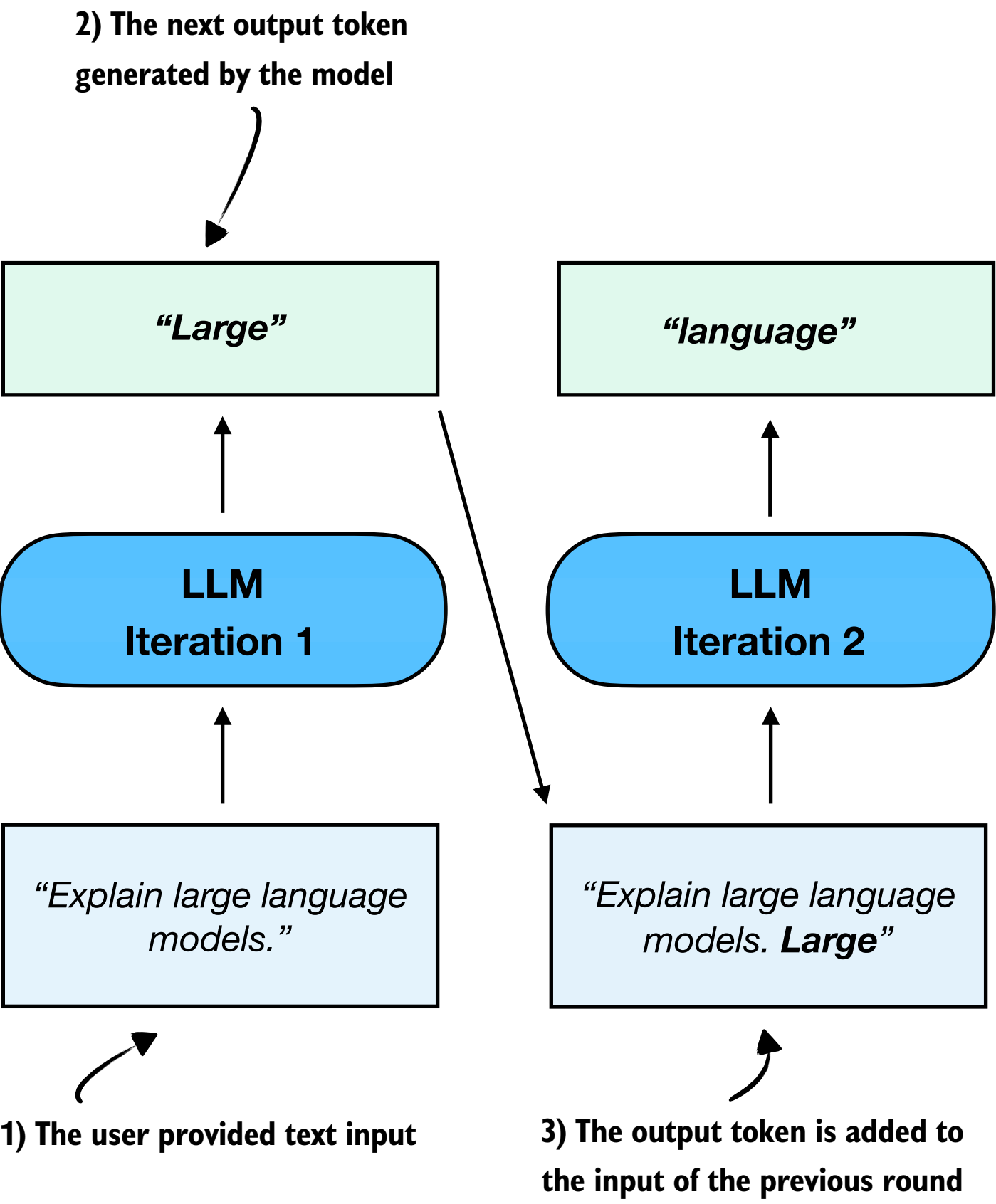
**Because we need longer contexts
(reasoning, agent harnesses)**



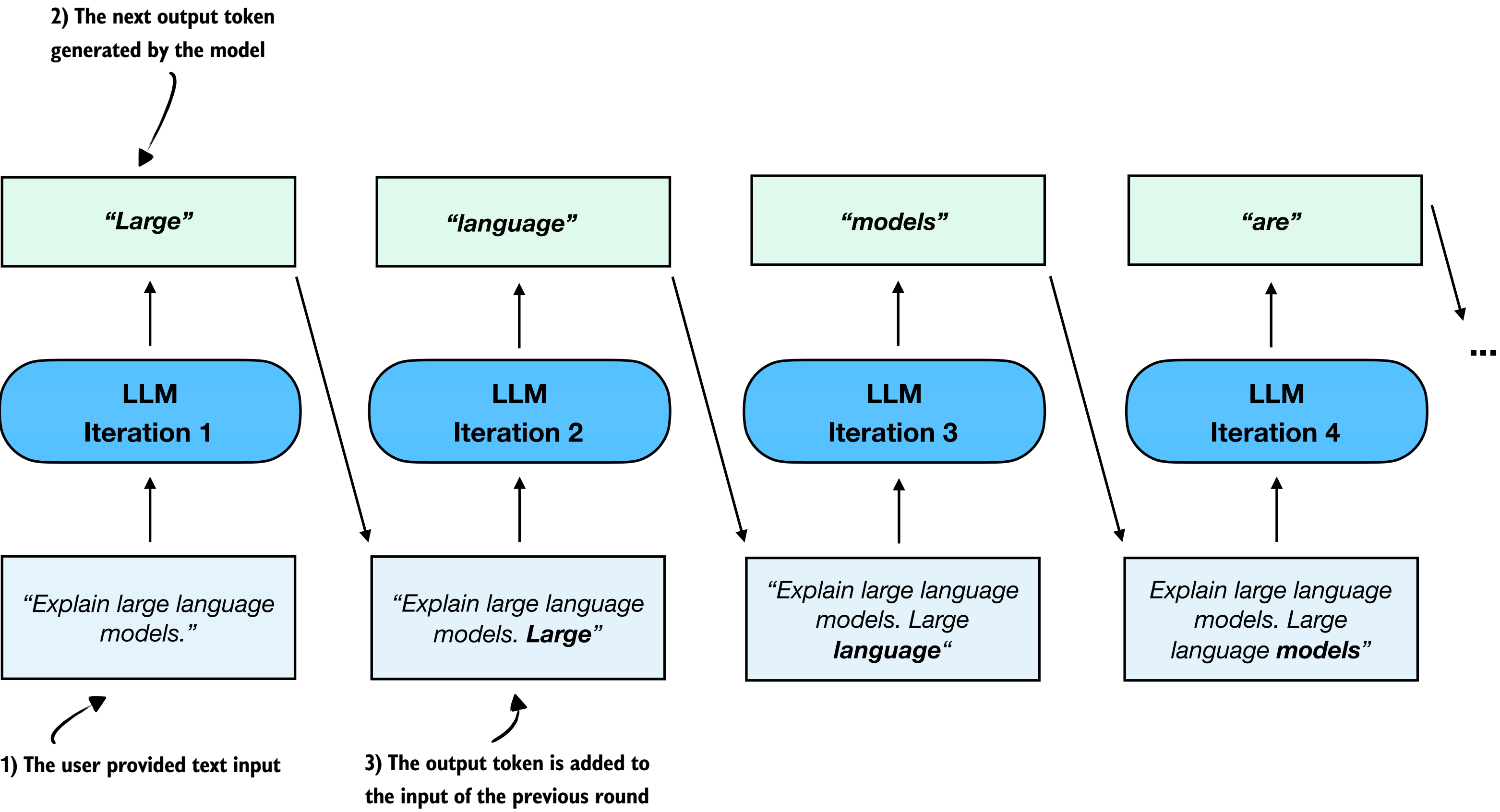
Text generation **without** KV cache



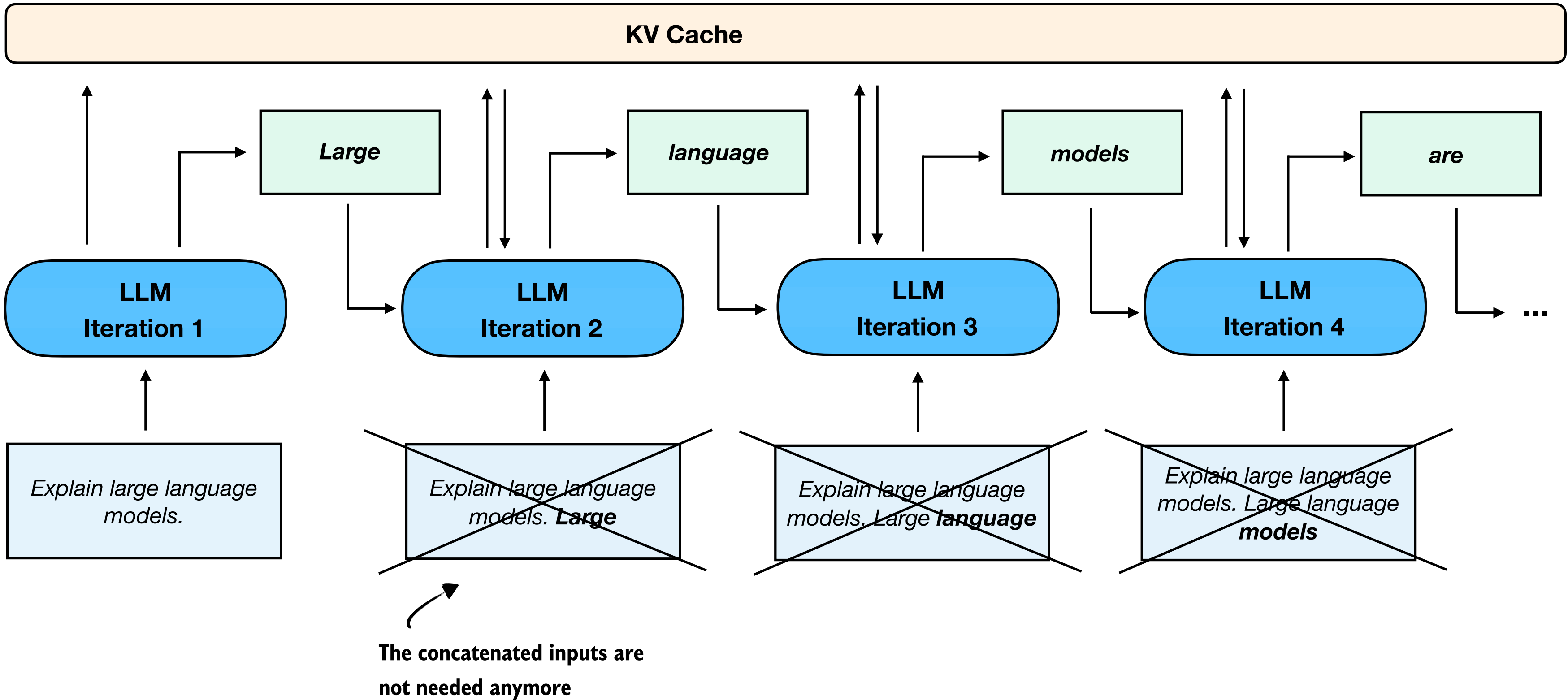
Text generation **without** KV cache



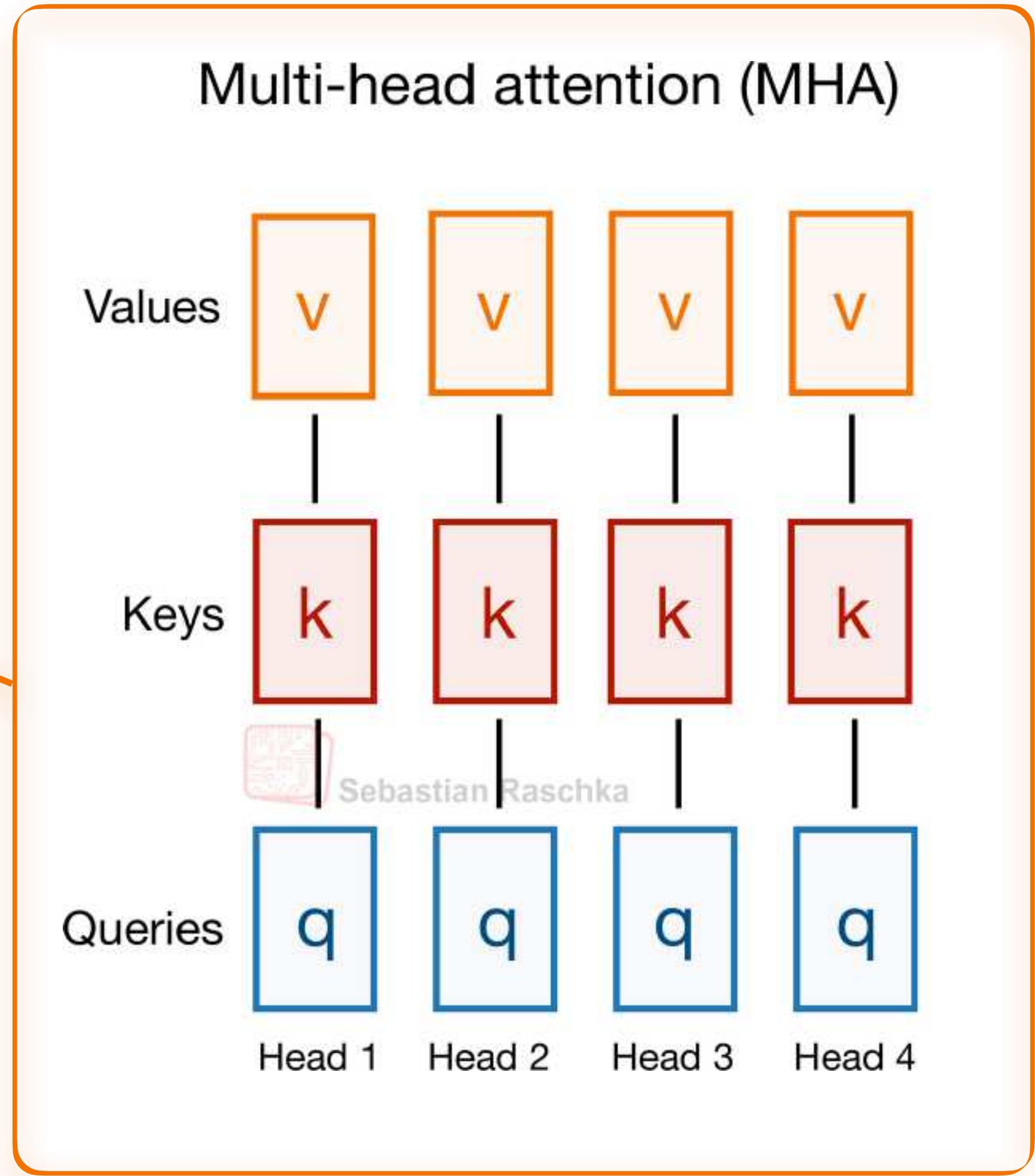
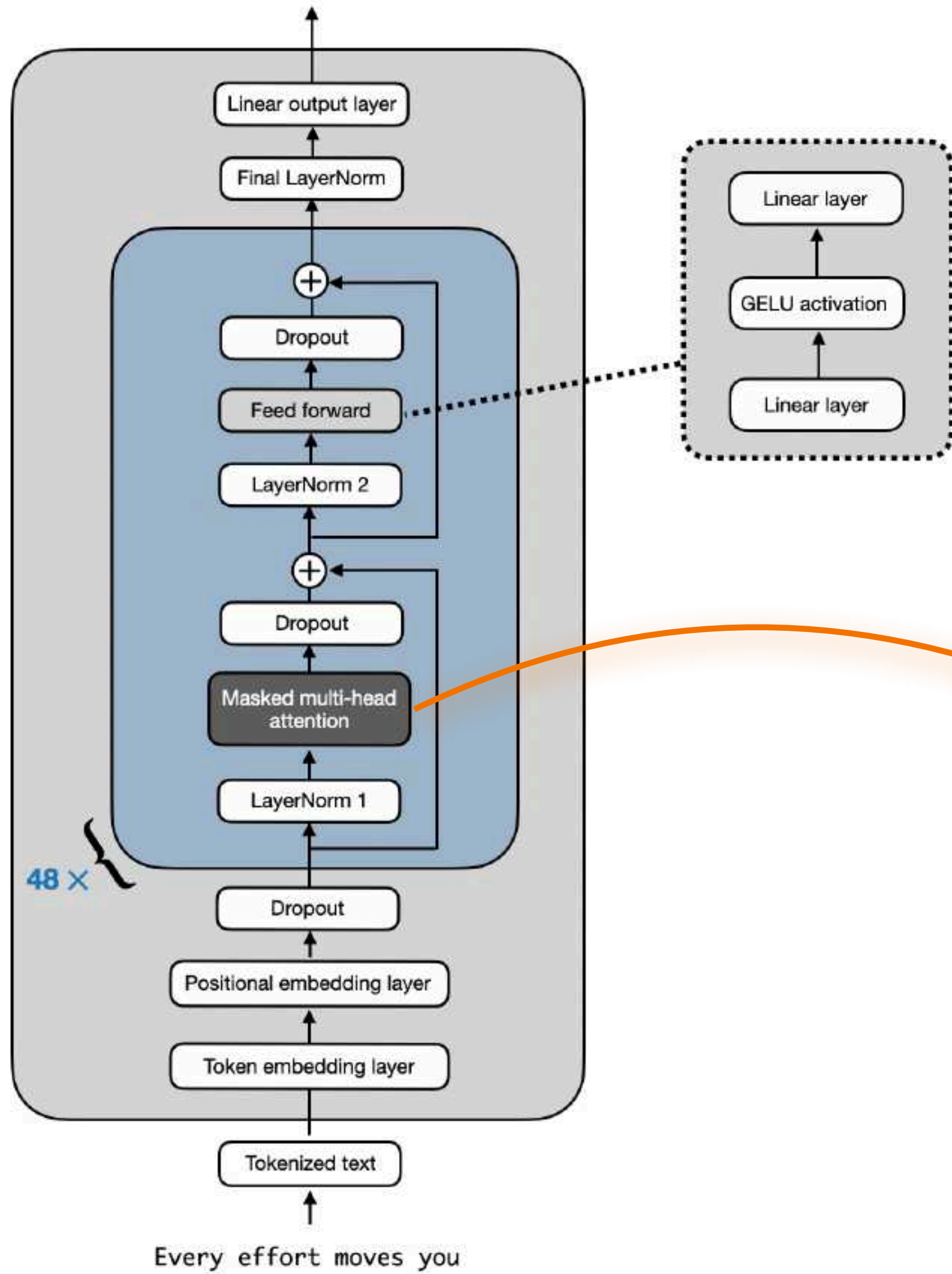
Text generation **without** KV cache



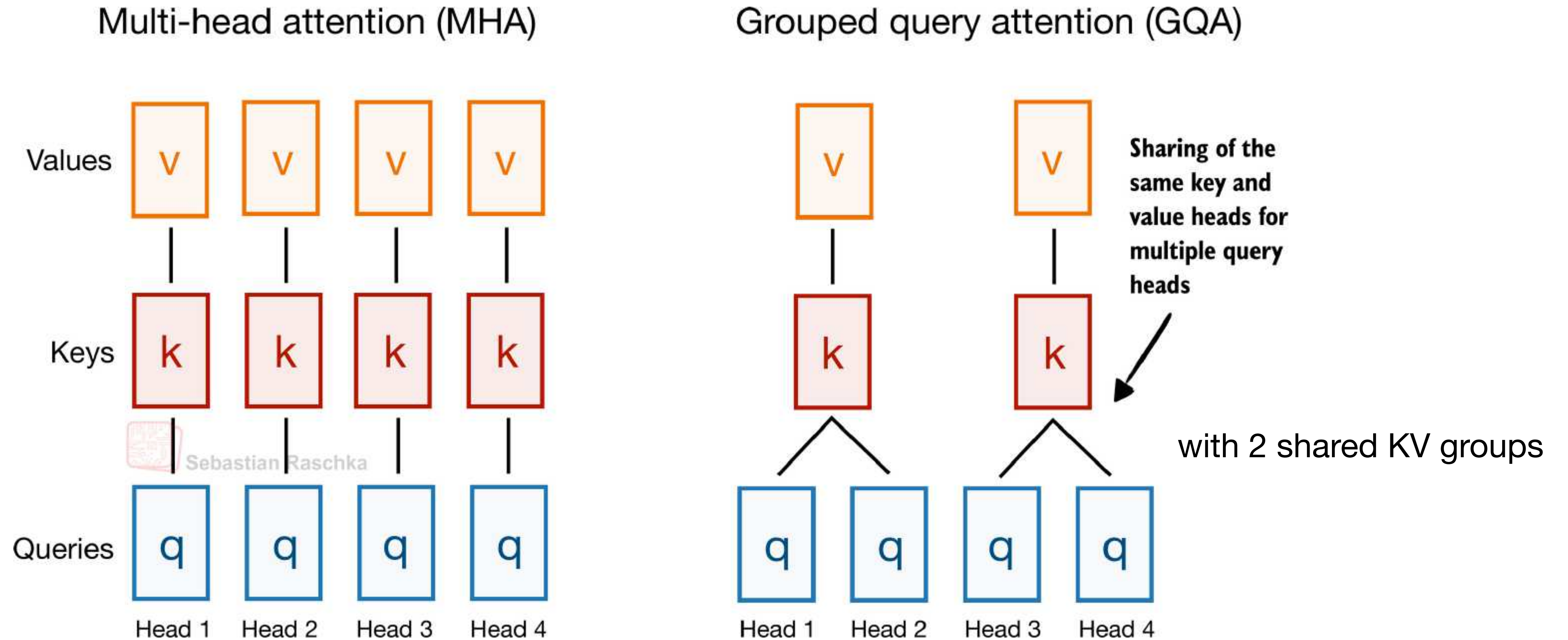
Text generation with KV cache



GPT-2 XL (1.5B)

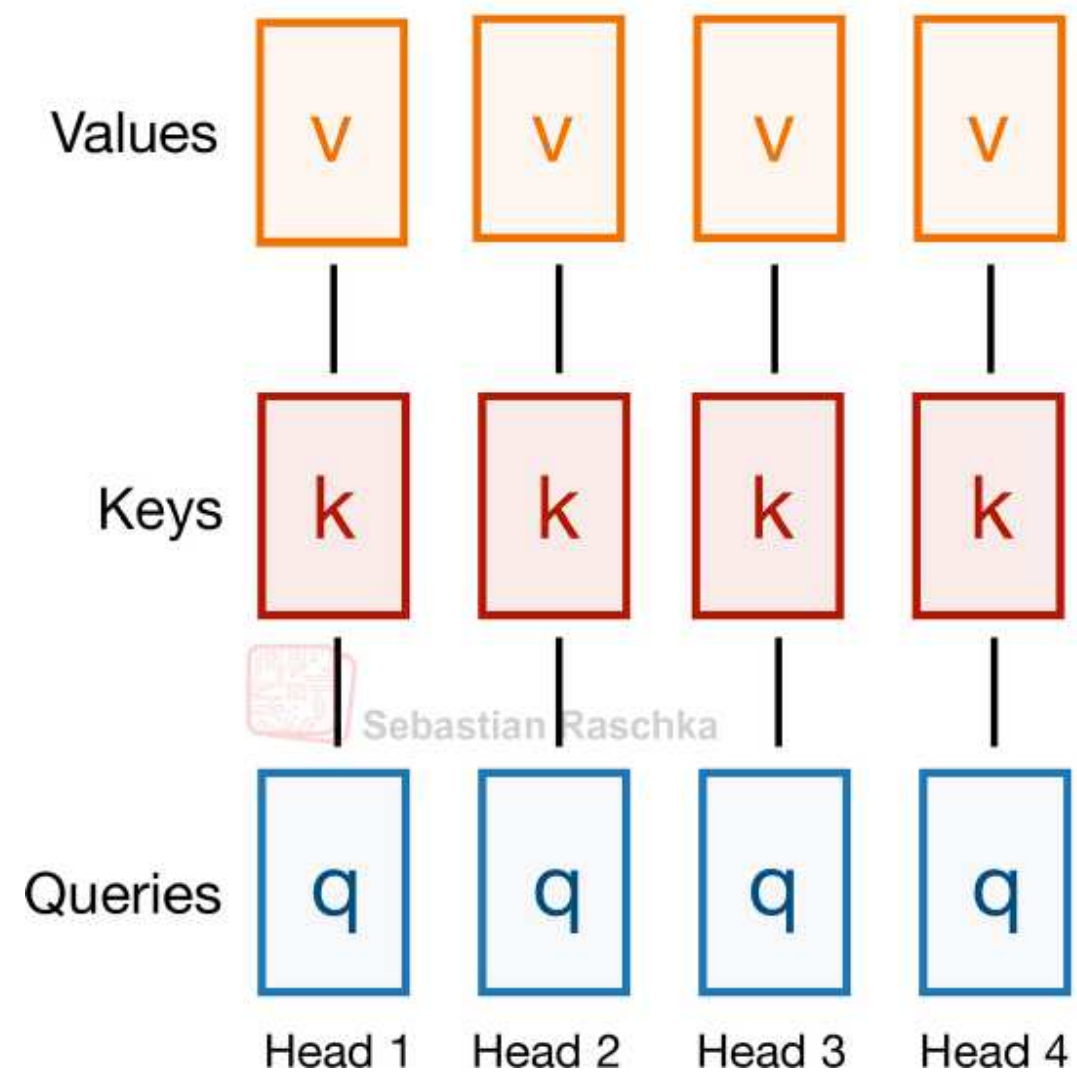


Method 1 to reduce KV cache size

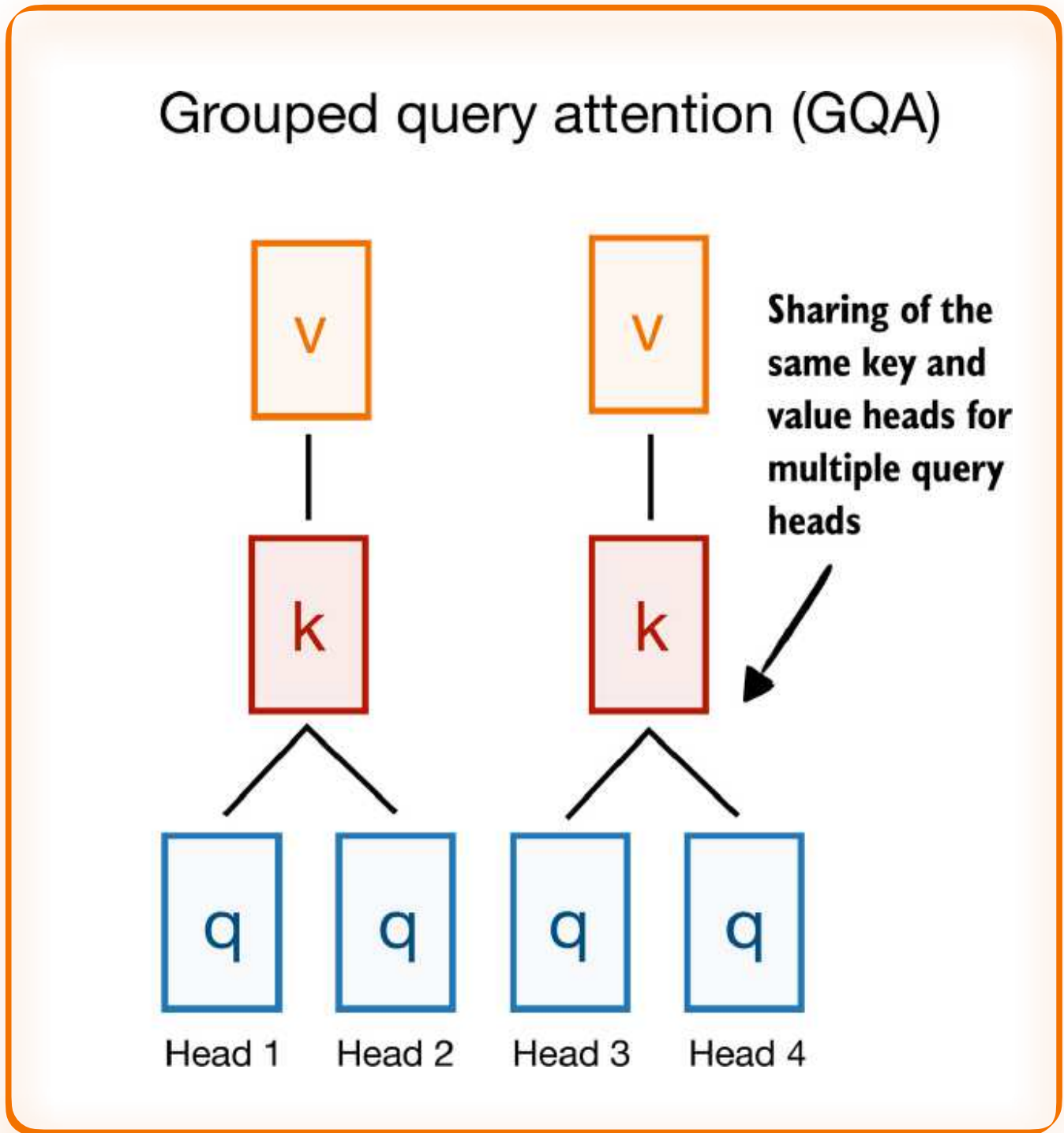


Method 1 to reduce KV cache size

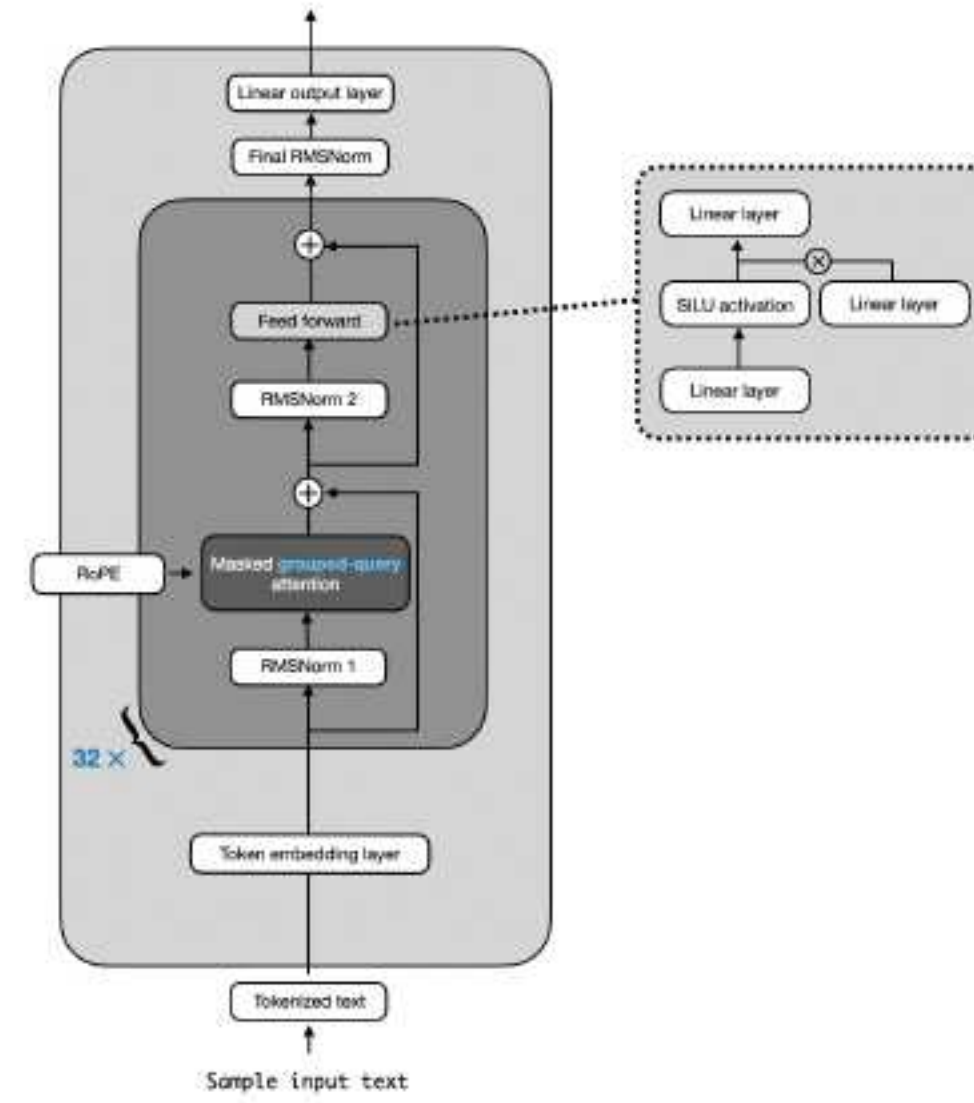
Multi-head attention (MHA)



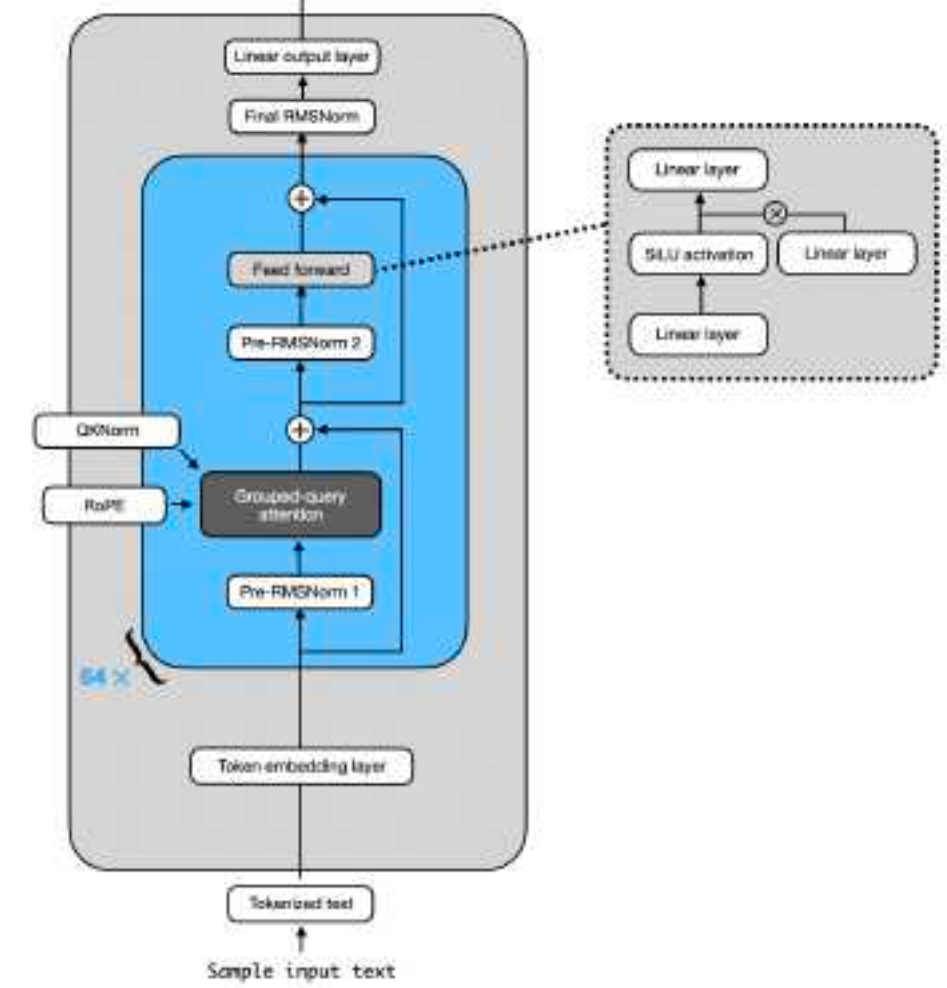
Grouped query attention (GQA)



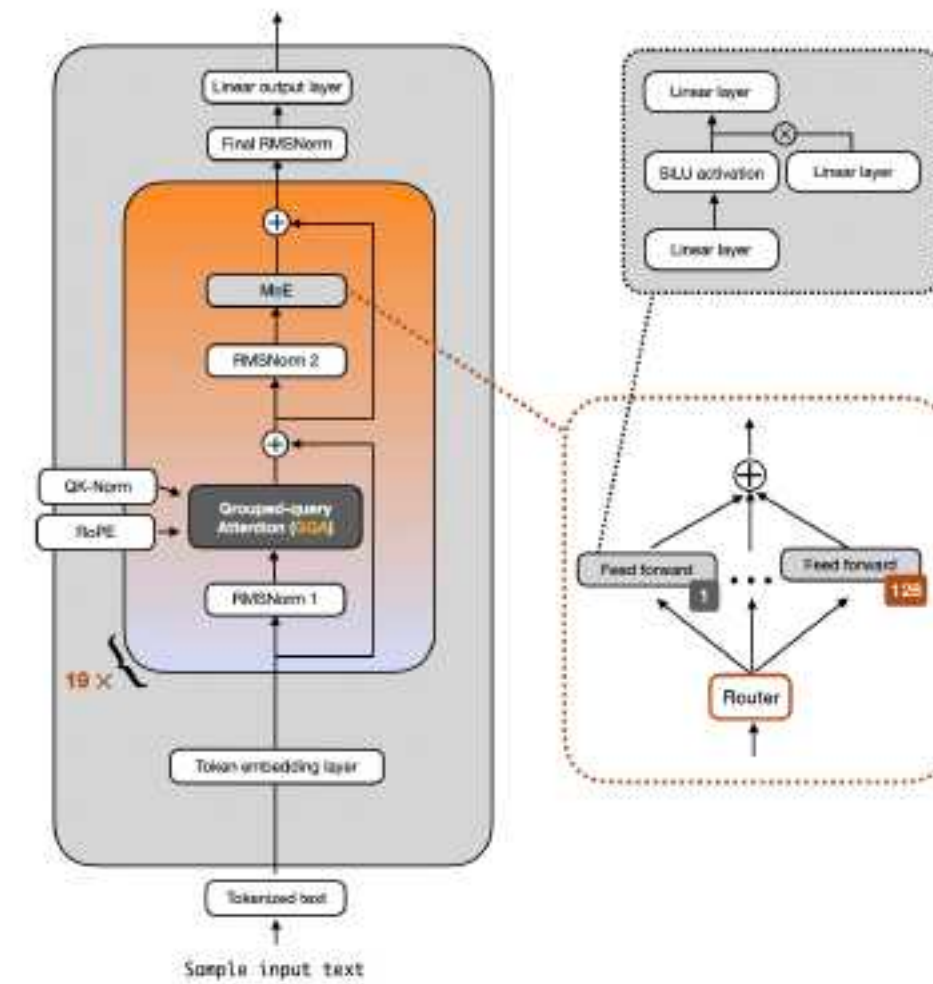
Llama 3 (8B)



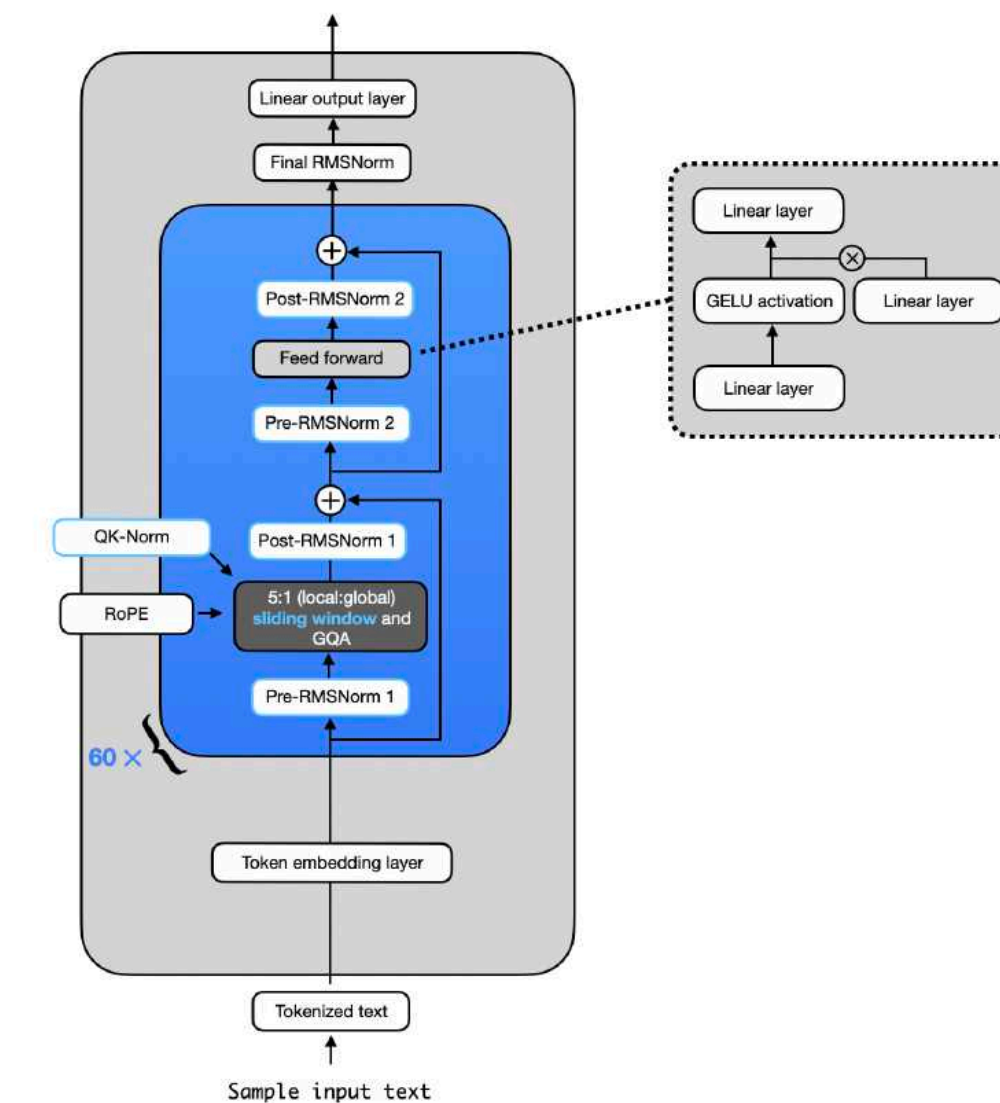
Qwen3 (32B)



Sarvam (30B)

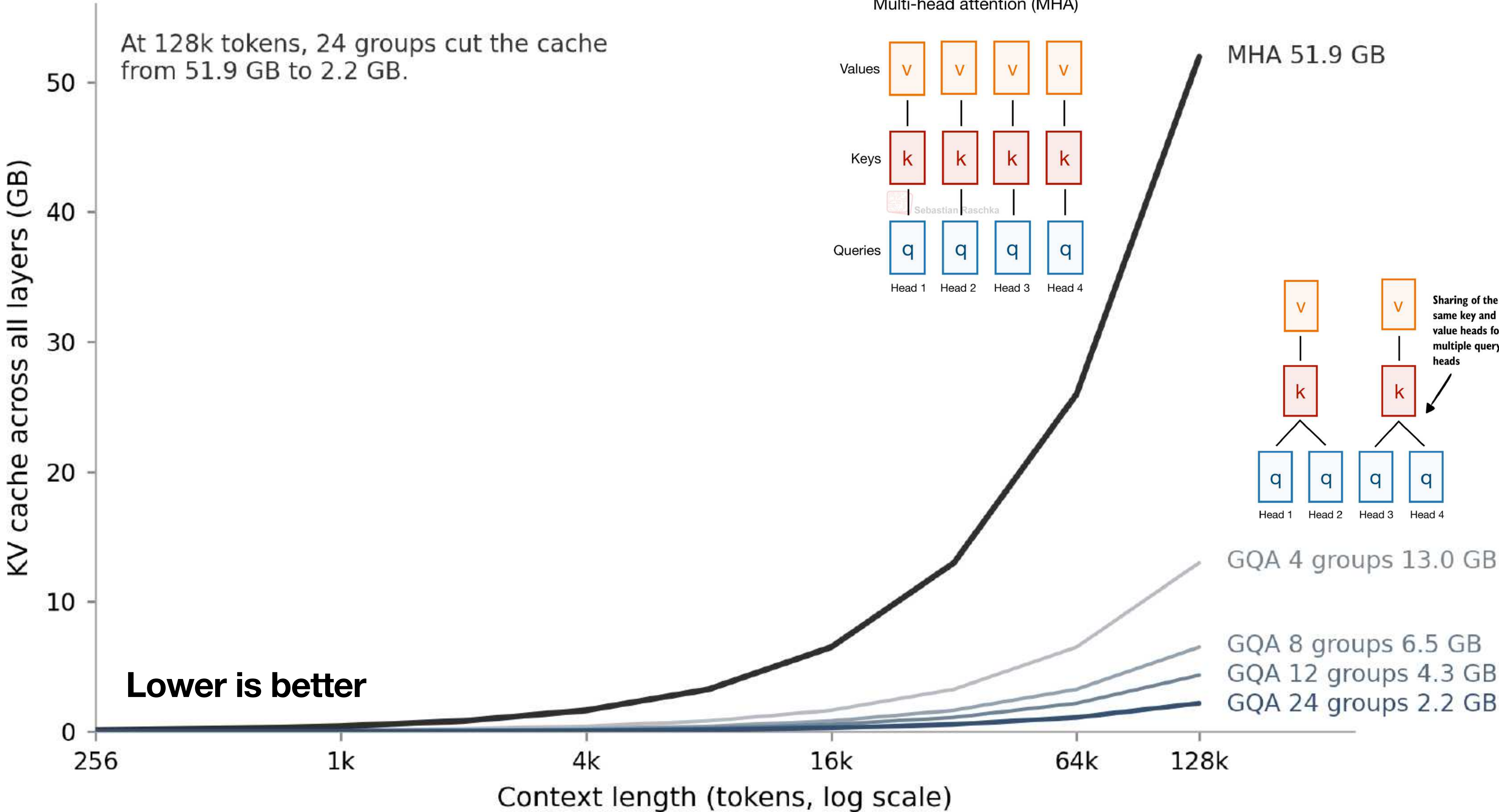


Gemma 4 (31B)

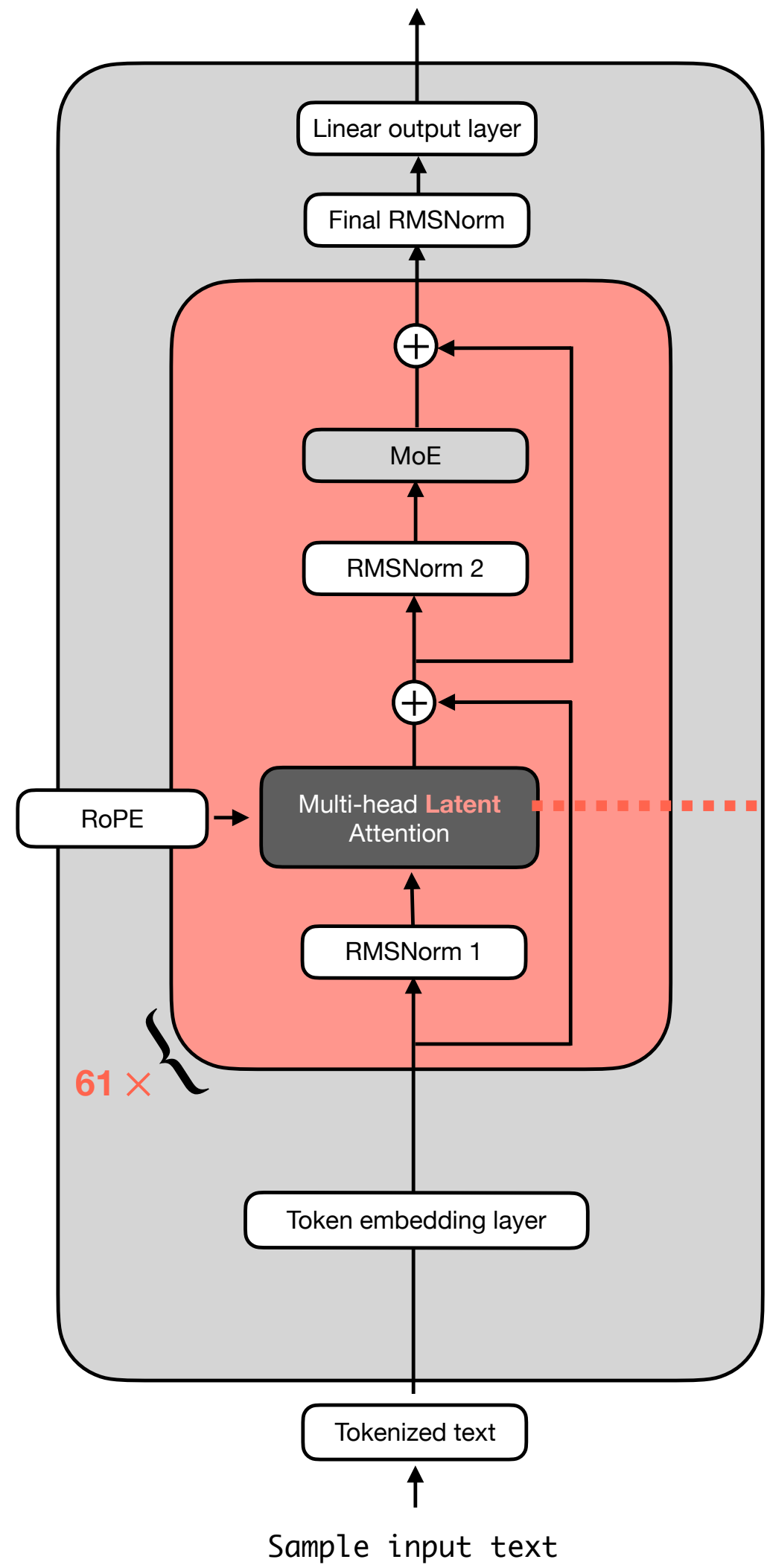


GQA's shared K/V heads sharply reduce KV-cache

24 heads, 48 layers, 2048-dim model, batch 1, bf16

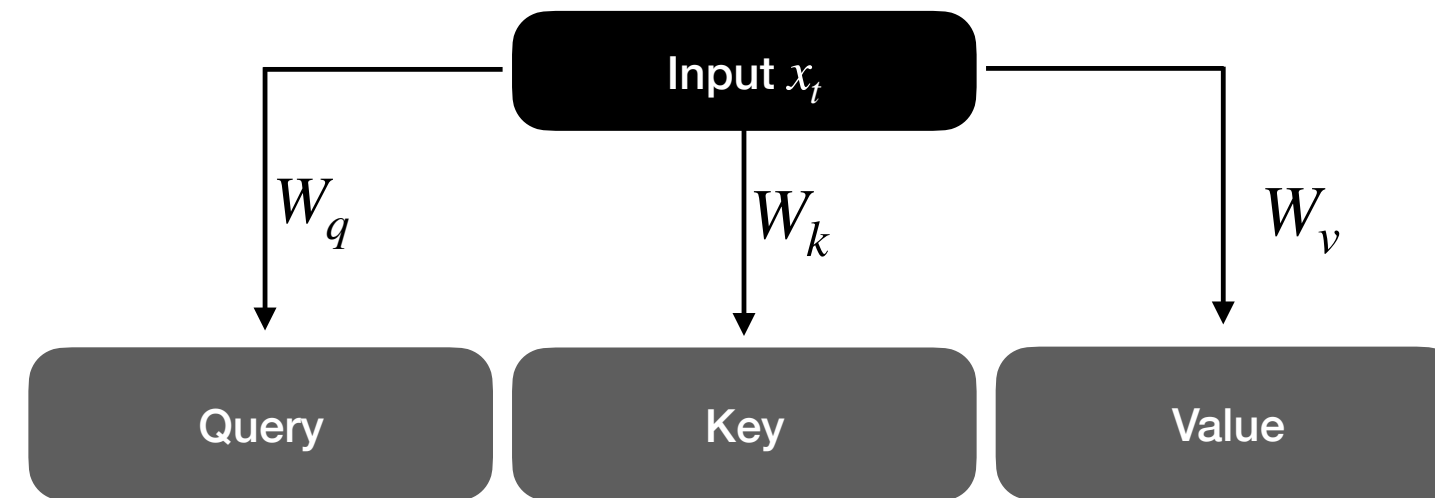


DeepSeek V3/R1



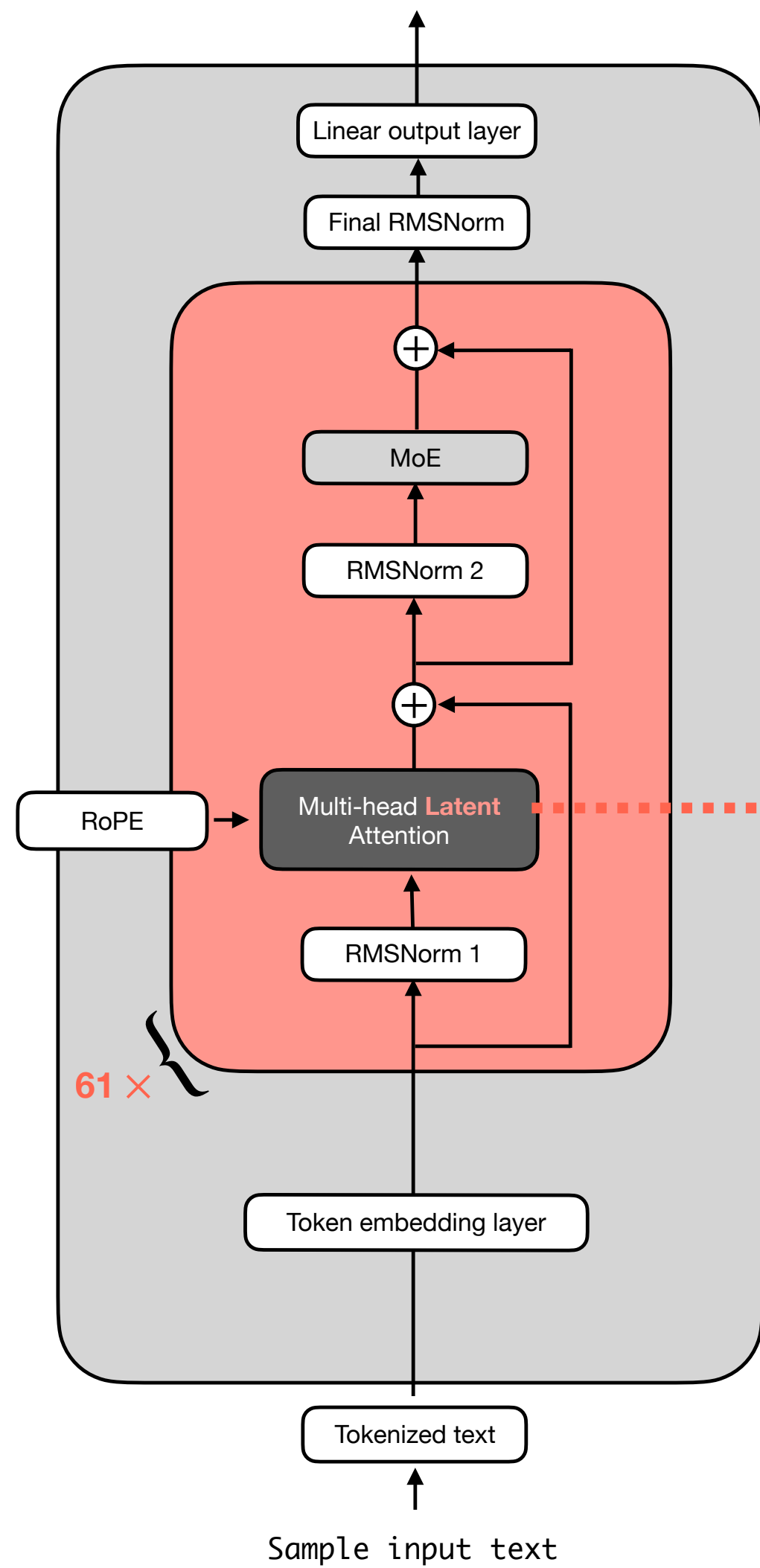
Regular Multi-Head Attention (MHA)

Inference step t :



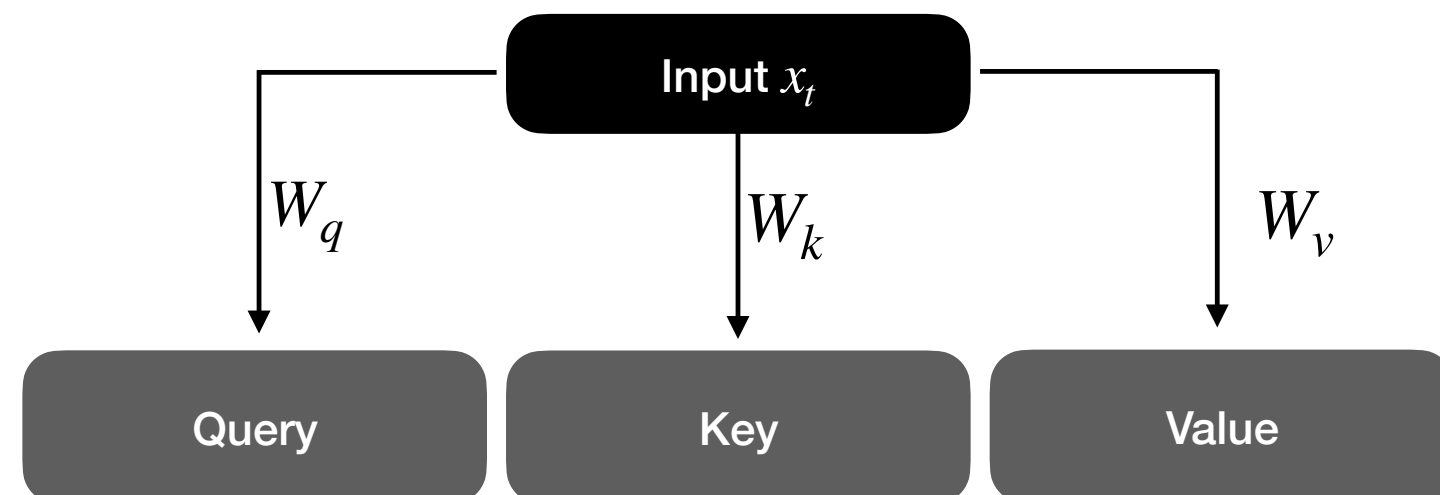
Method 2 to reduce KV cache size

DeepSeek V3/R1



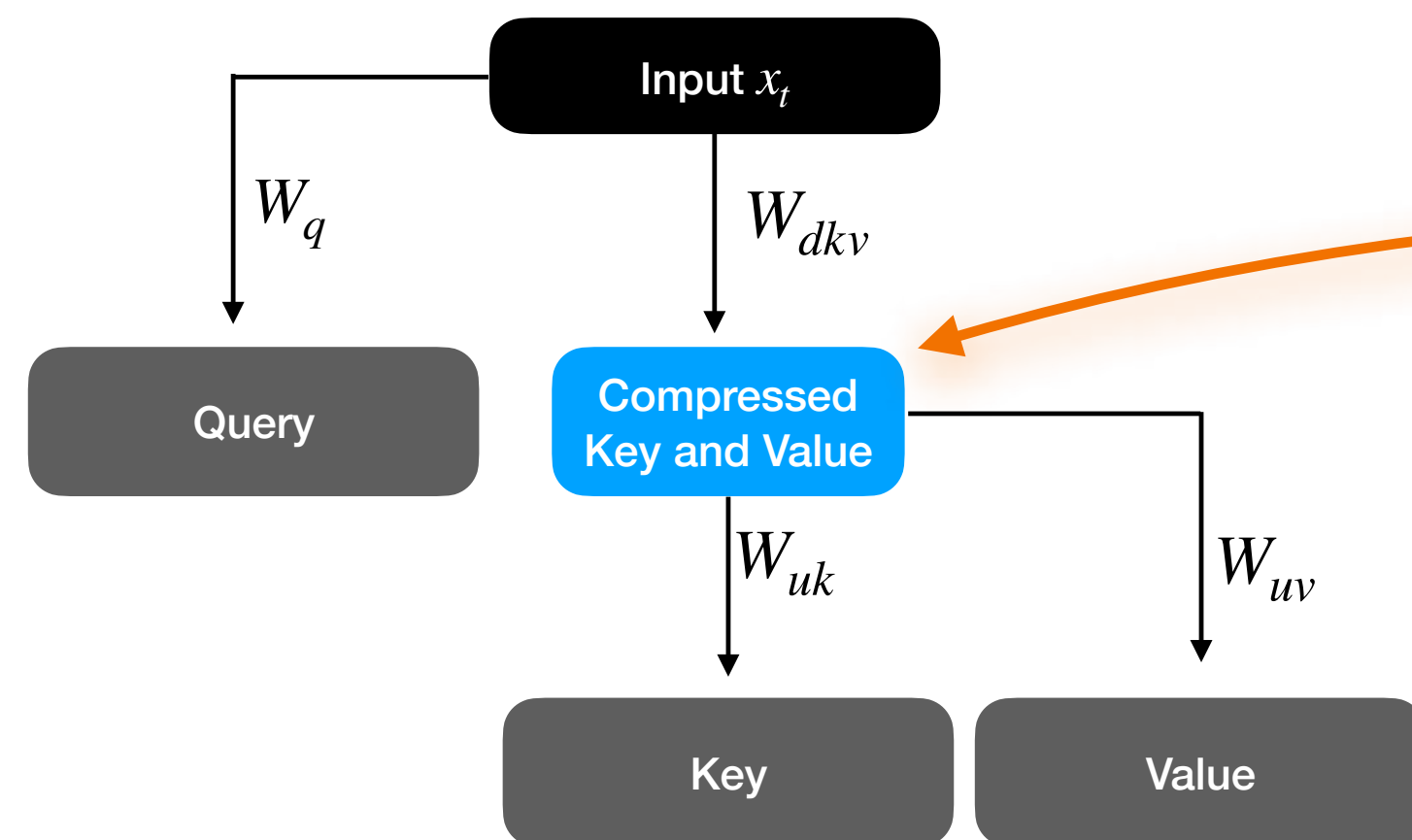
Regular Multi-Head Attention (MHA)

Inference step t :



Multi-Head Latent Attention (MLA)

Inference step t :

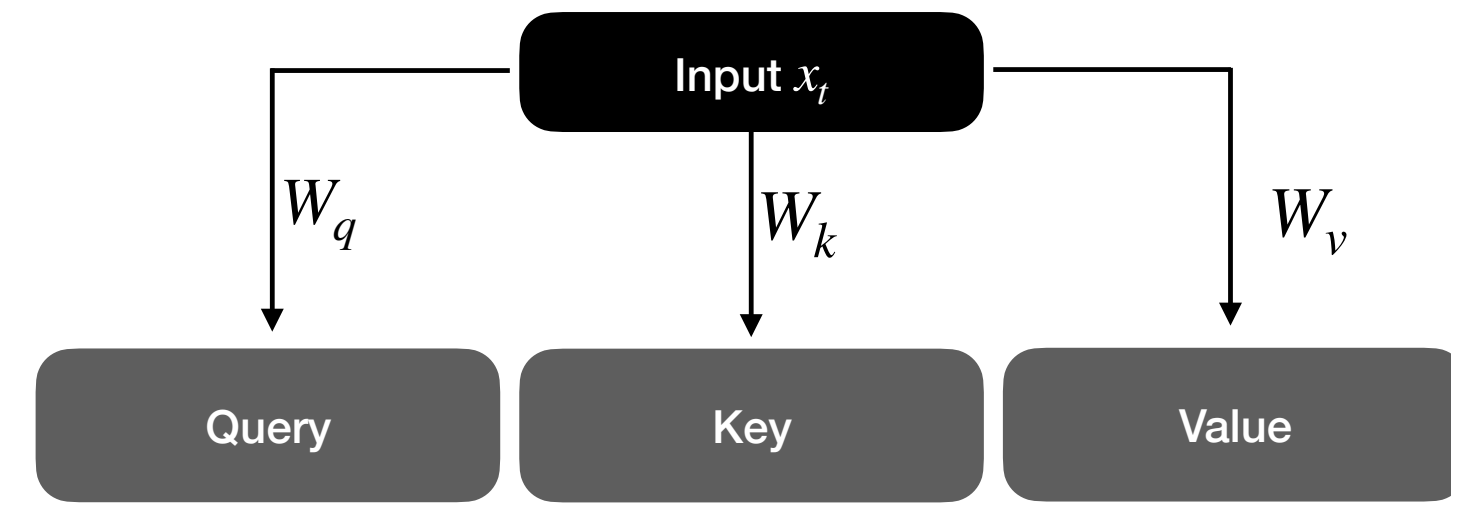
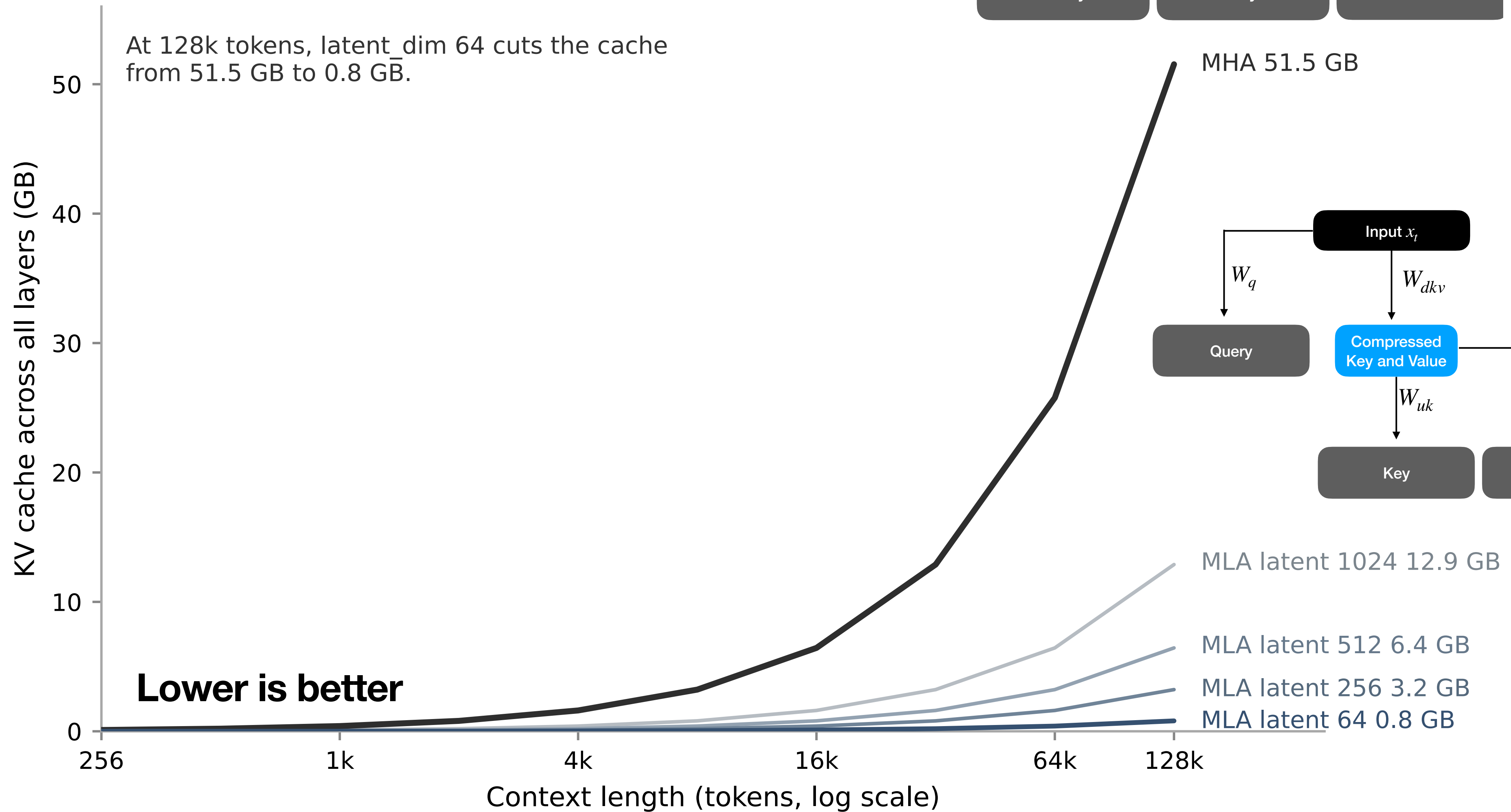


Method 2 to reduce KV cache size

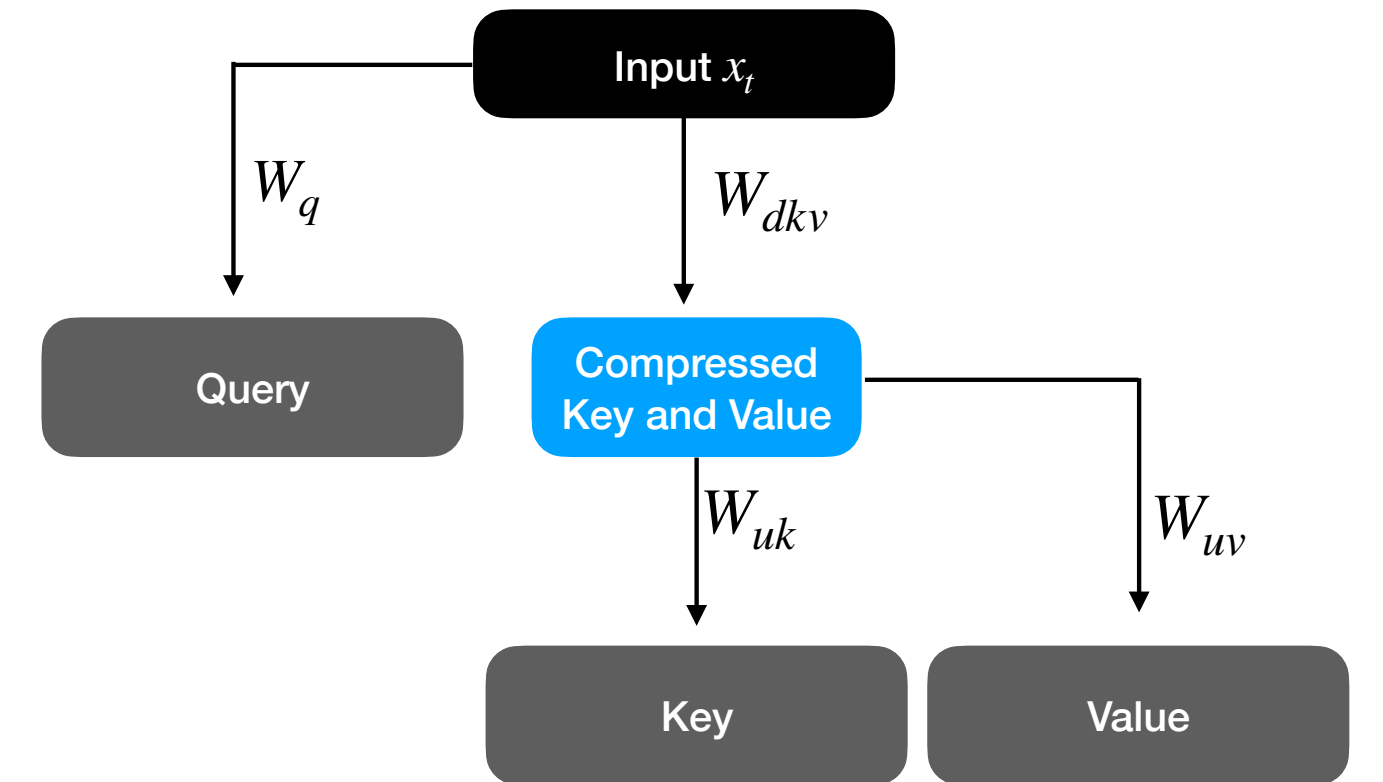
Store "small" intermediate compressed state in KV cache

MLA's latent cache sharply reduces KV-cache

24 heads, 48 layers, 2048-dim model, batch 1, bf16



MHA 51.5 GB



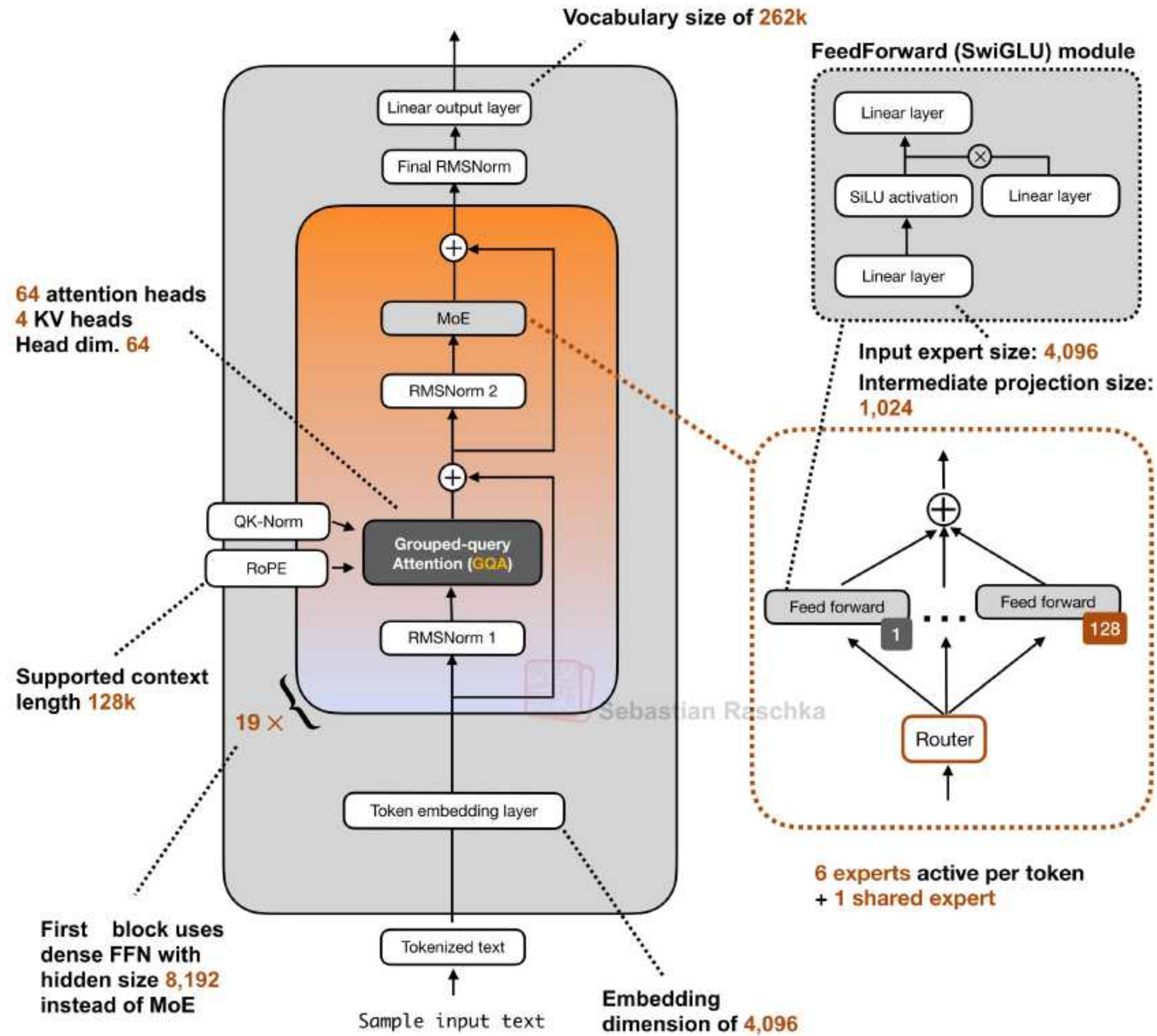
MLA latent 1024 12.9 GB

MLA latent 512 6.4 GB

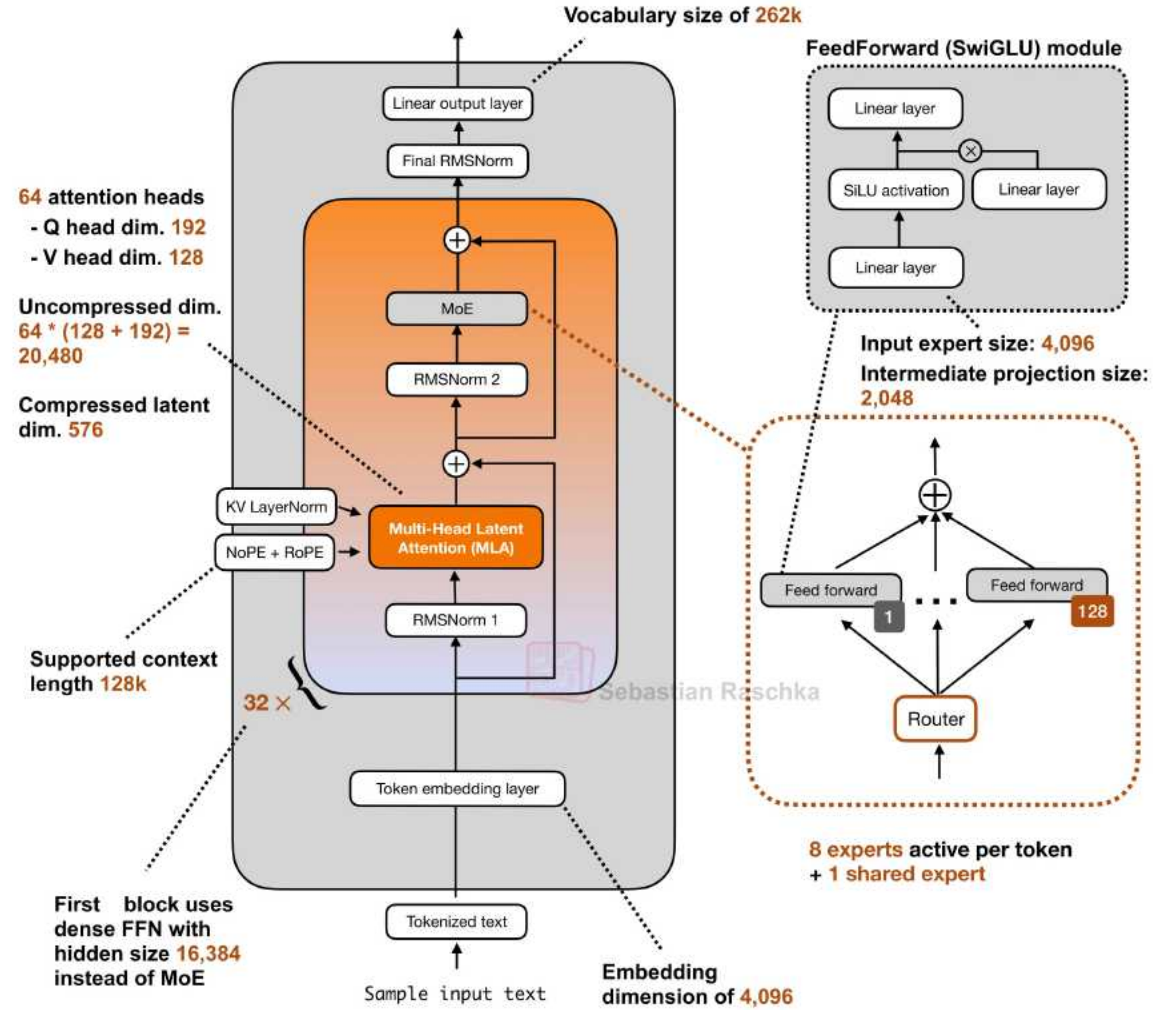
MLA latent 256 3.2 GB

MLA latent 64 0.8 GB

Sarvam (30B)



Sarvam (105B)



Method 3 to reduce KV cache size

Regular (causal) self-attention mask

	The	model	attends	to	past	tokens
The	1	0	0	0	0	0
model	1	1	0	0	0	0
attends	1	1	1	0	0	0
to	1	1	1	1	0	0
past	1	1	1	1	1	0
tokens	1	1	1	1	1	1

The diagram illustrates a regular (causal) self-attention mask for the sequence "The model attends to past tokens". The mask is a 6x6 grid of 1s and 0s. The 'to' token is highlighted with a red box, and red arrows point from it to the 'attends' token and the 'past' token, indicating that the 'to' token can attend to these tokens. The 'attends' token can attend to 'The', 'model', and 'attends'. The 'past' token can attend to all previous tokens. The 'tokens' token can attend to all previous tokens.

Method 3 to reduce KV cache size

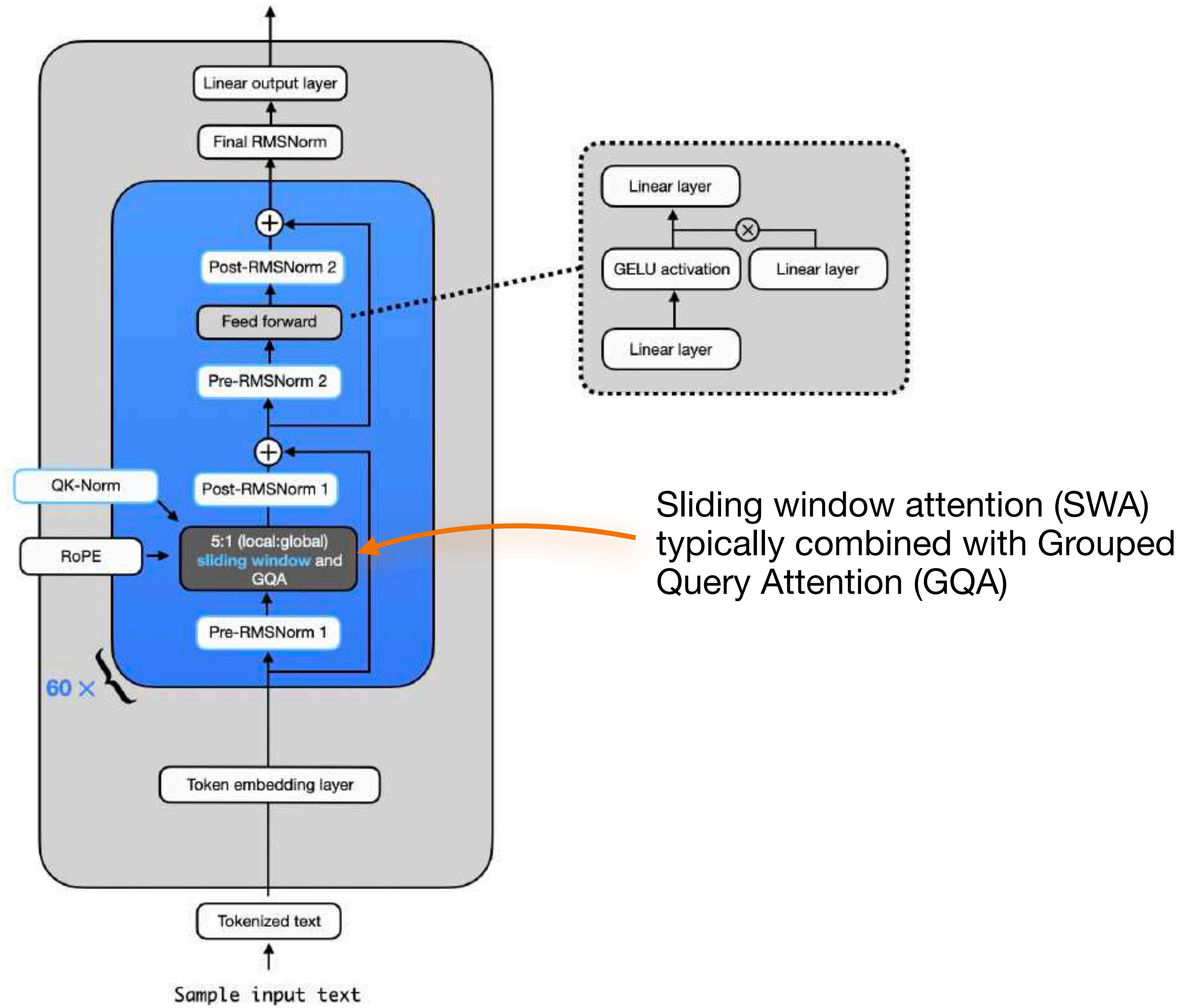
Regular (causal) self-attention mask

	The	model	attends	to	past	tokens
The	1	0	0	0	0	0
model	1	1	0	0	0	0
attends	1	1	1	0	0	0
to	1	1	1	1	0	0
past	1	1	1	1	1	0
tokens	1	1	1	1	1	1

Sliding window attention

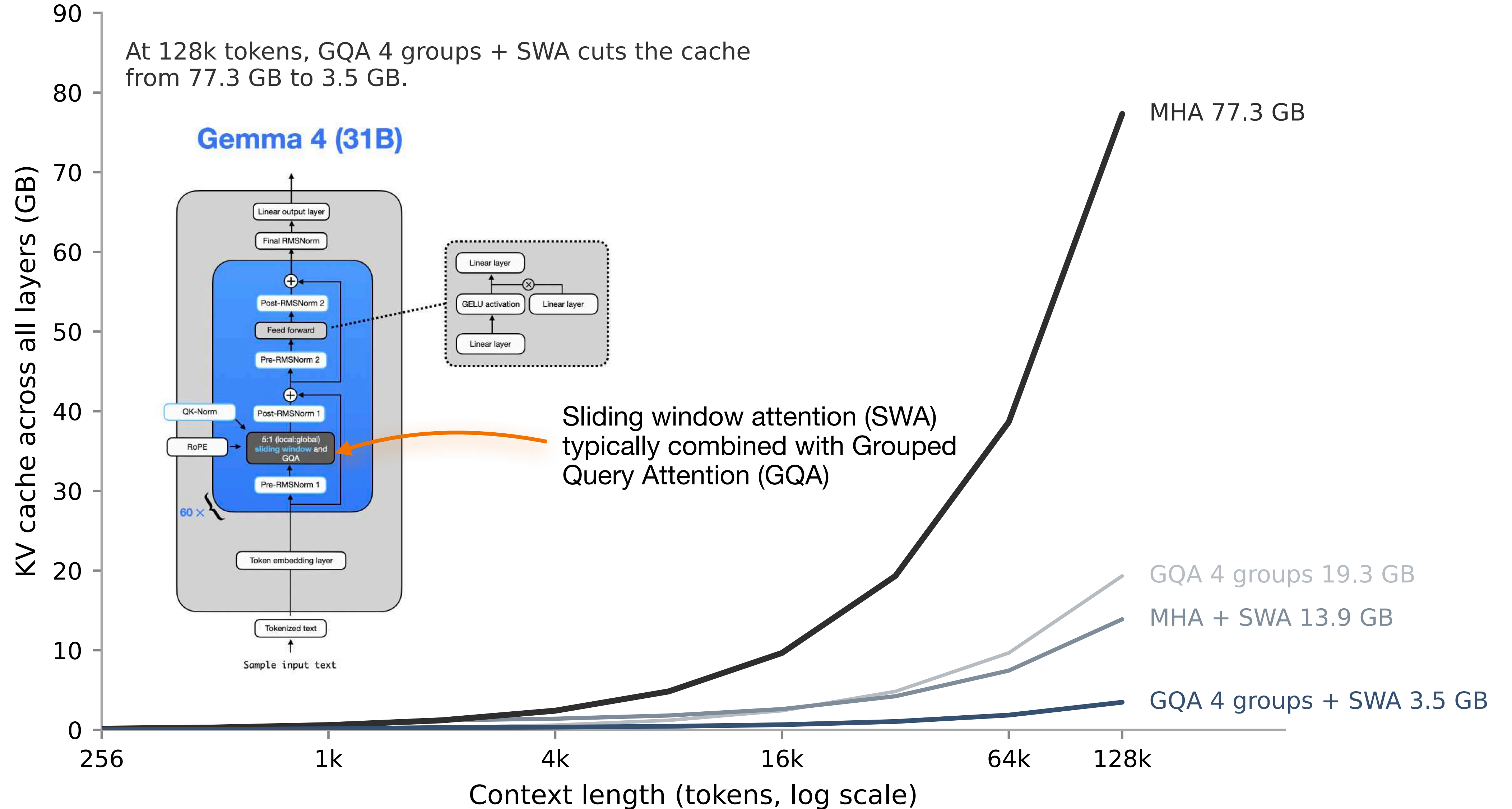
	The	model	attends	to	past	tokens
The	1	0	0	0	0	0
model	1	1	0	0	0	0
attends	1	1	1	0	0	0
to	0	1	1	1	0	0
past	0	0	1	1	1	0
tokens	0	0	0	1	1	1

Gemma 4 (31B)



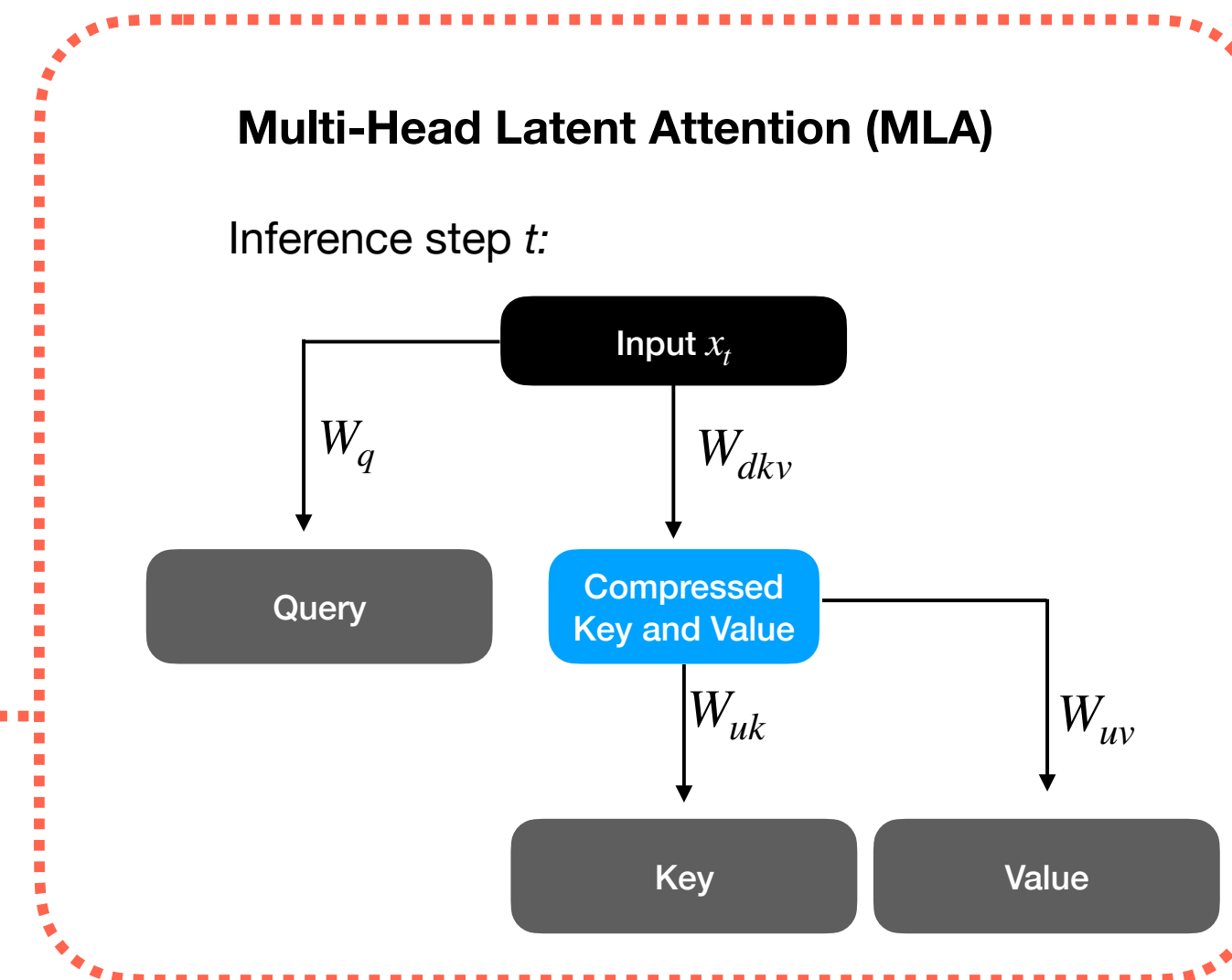
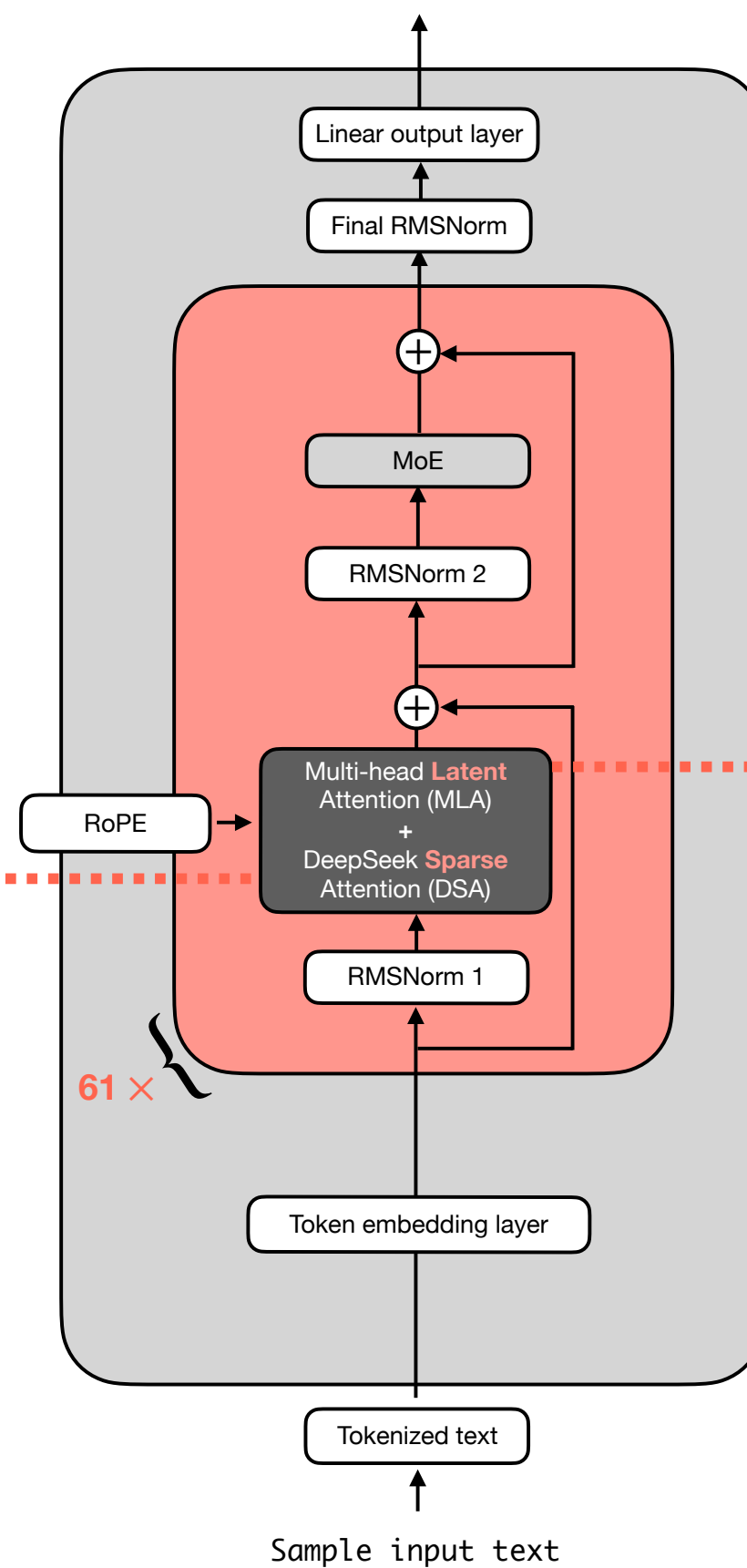
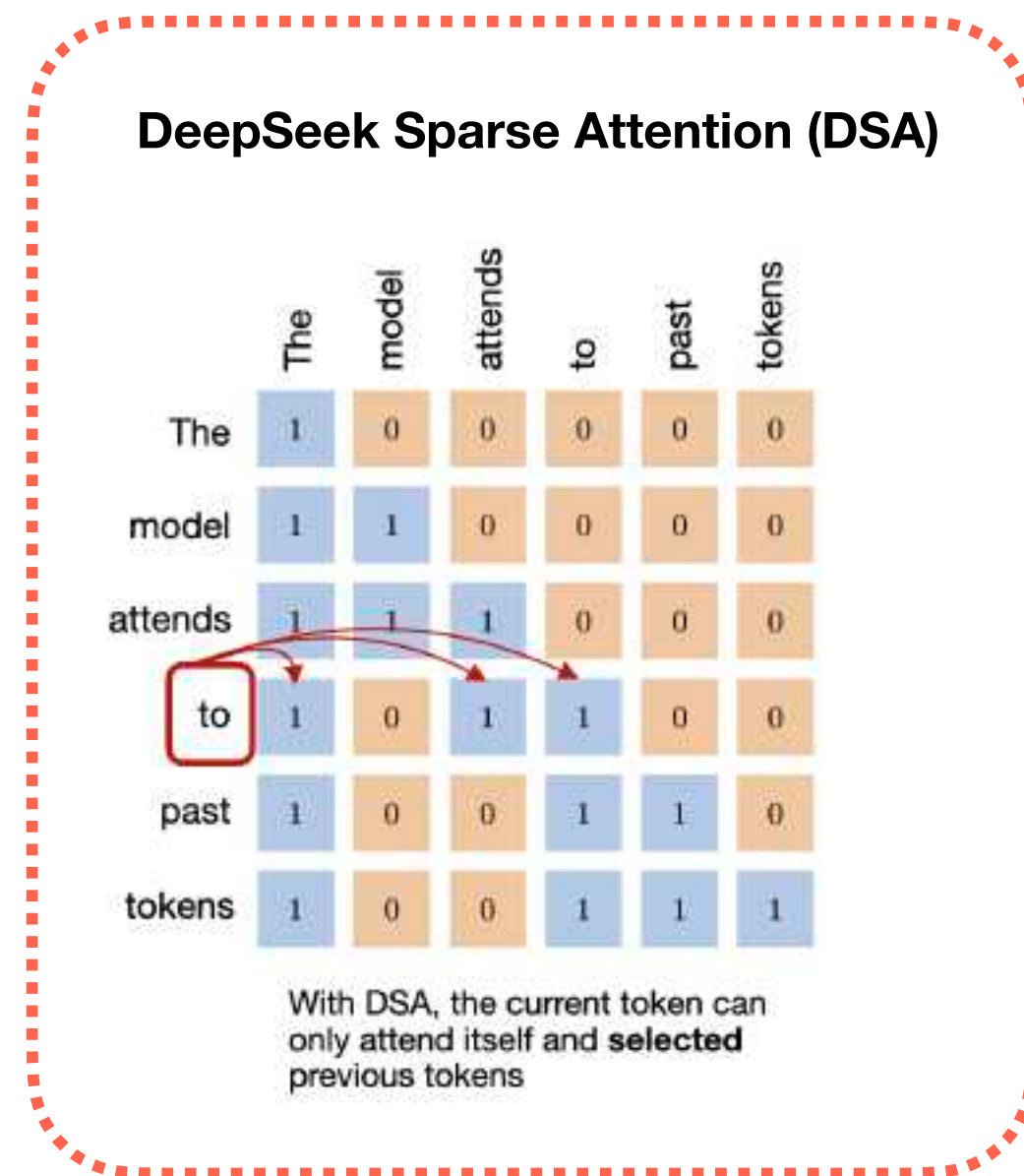
Sliding windows sharply reduce KV-cache growth

48 heads, 36 layers, 4096-dim model, batch 1, bf16; SWA ratio 5:1, window 2048



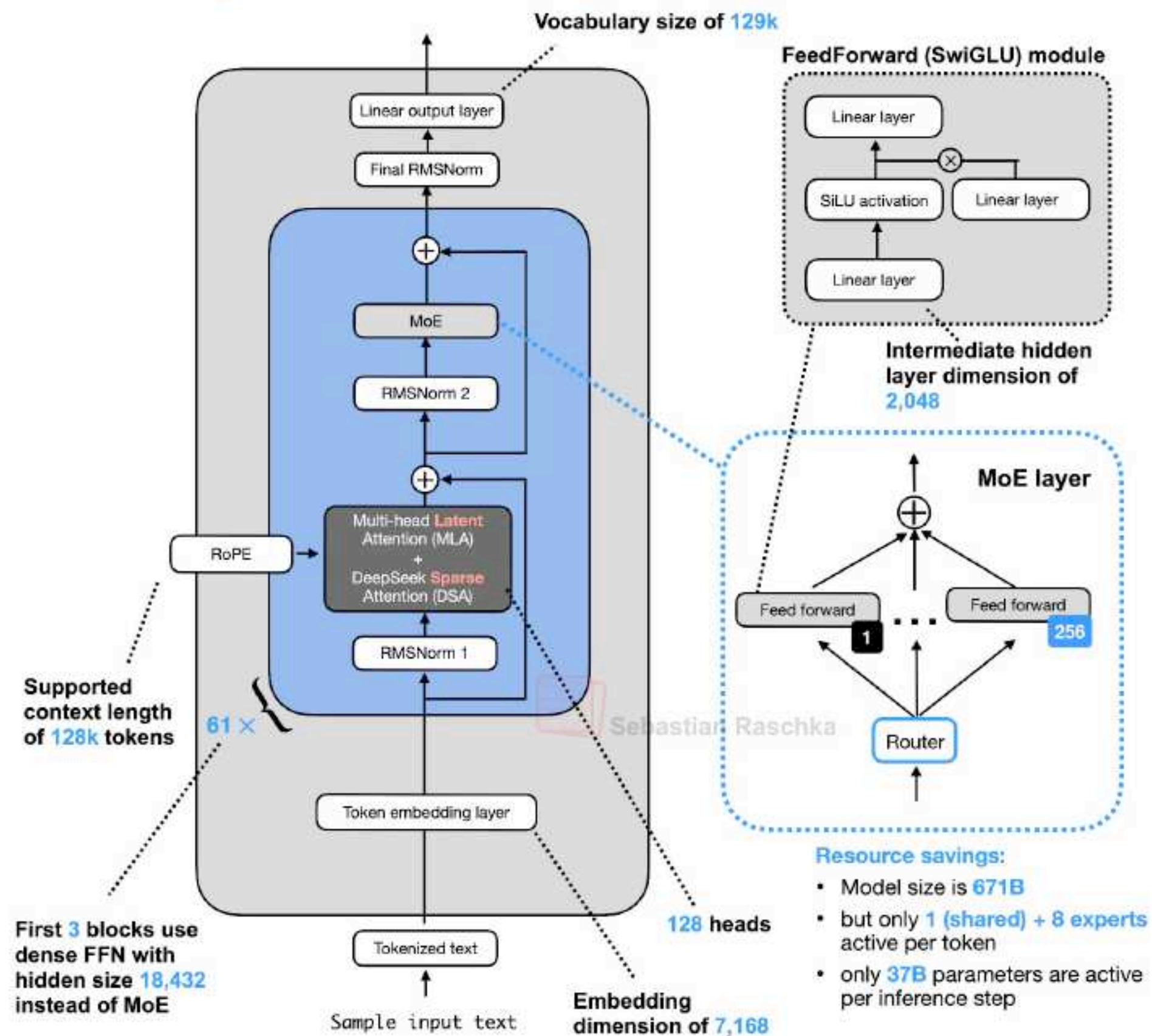
Method 4 to reduce KV cache size

DeepSeek V3.2

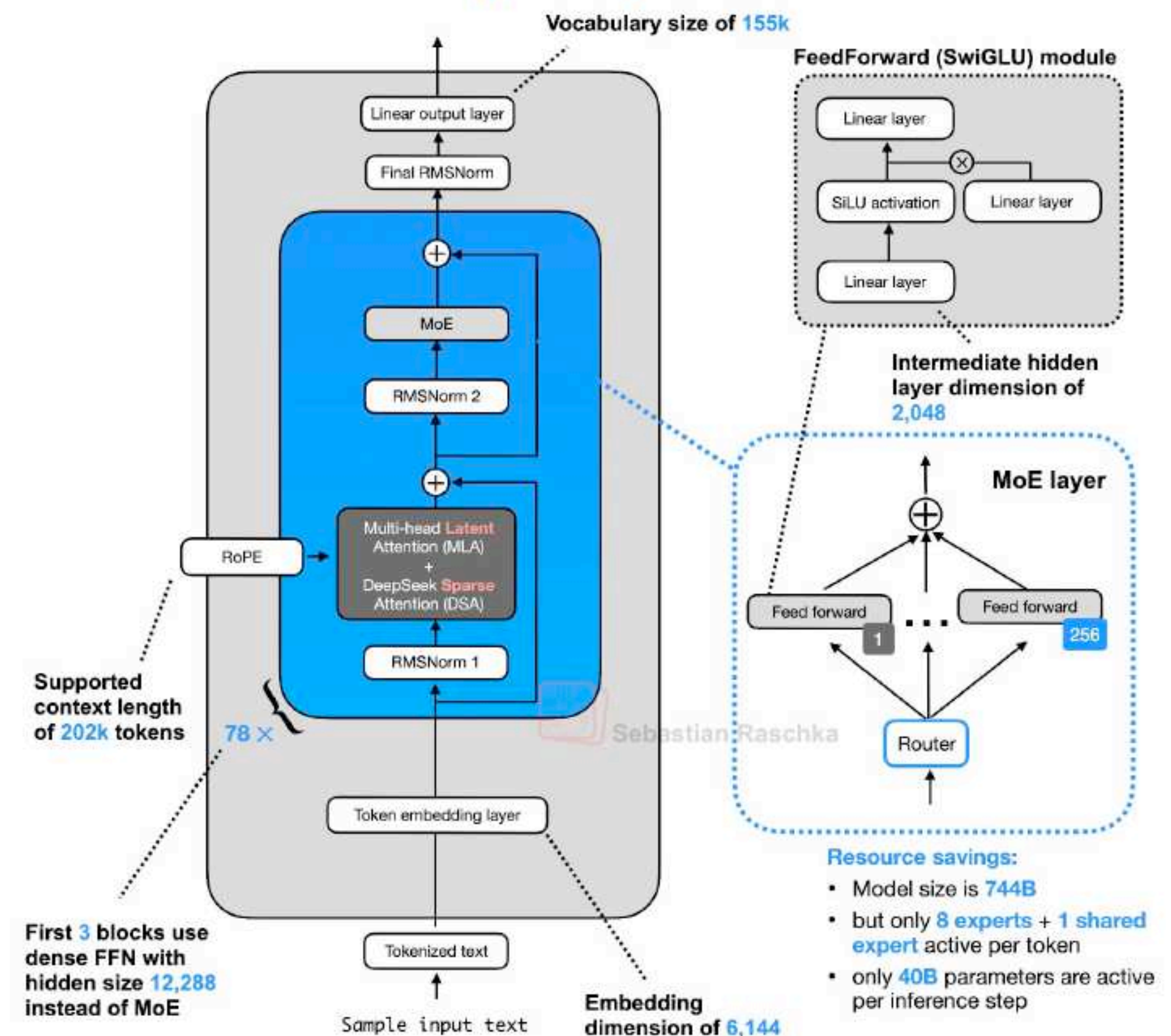


GLM-5.1 (strongest open-weight model) uses DeepSeek-V3.2 architecture

DeepSeek V3.2 (671B)



GLM-5.1 (744B)



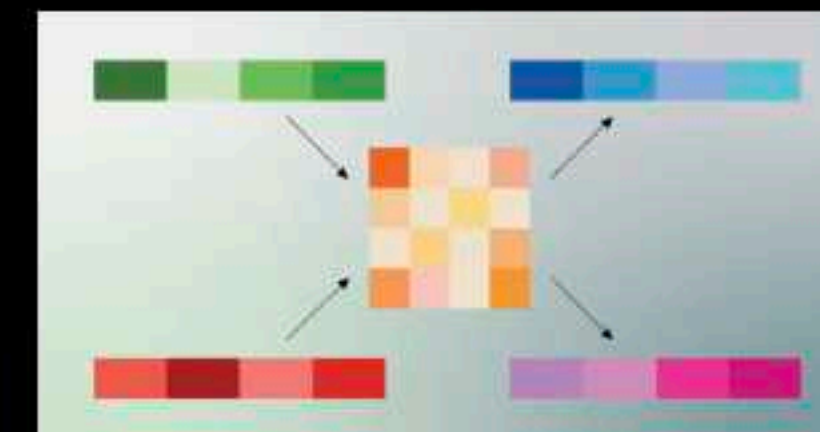
Method 5 to reduce KV cache size

Google Research

TurboQuant: Redefining AI efficiency with extreme compression

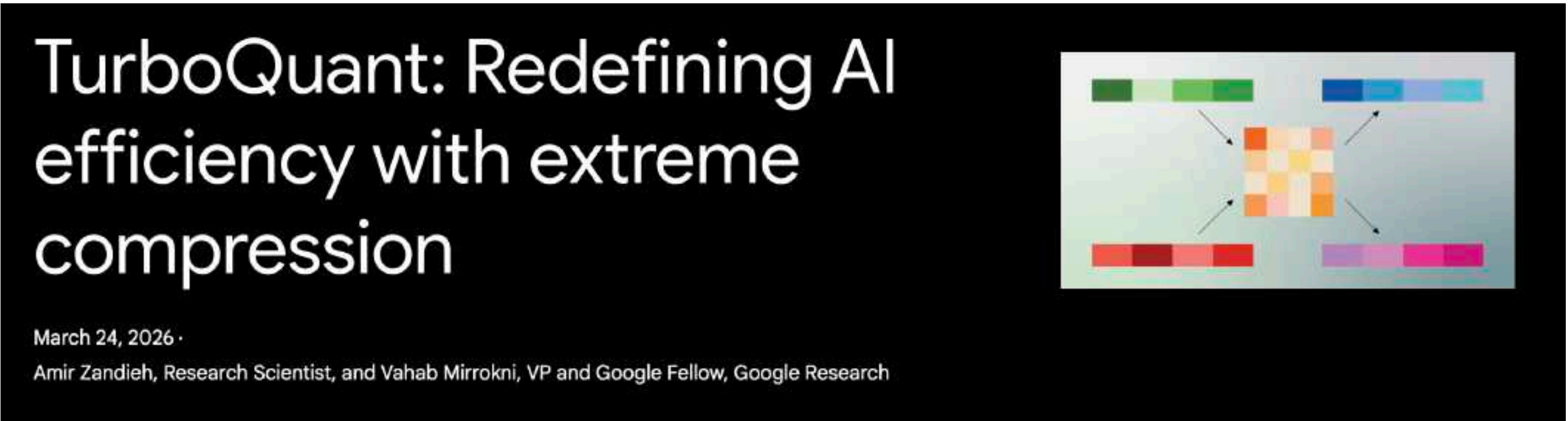
March 24, 2026 ·

Amir Zandieh, Research Scientist, and Vahab Mirrokni, VP and Google Fellow, Google Research



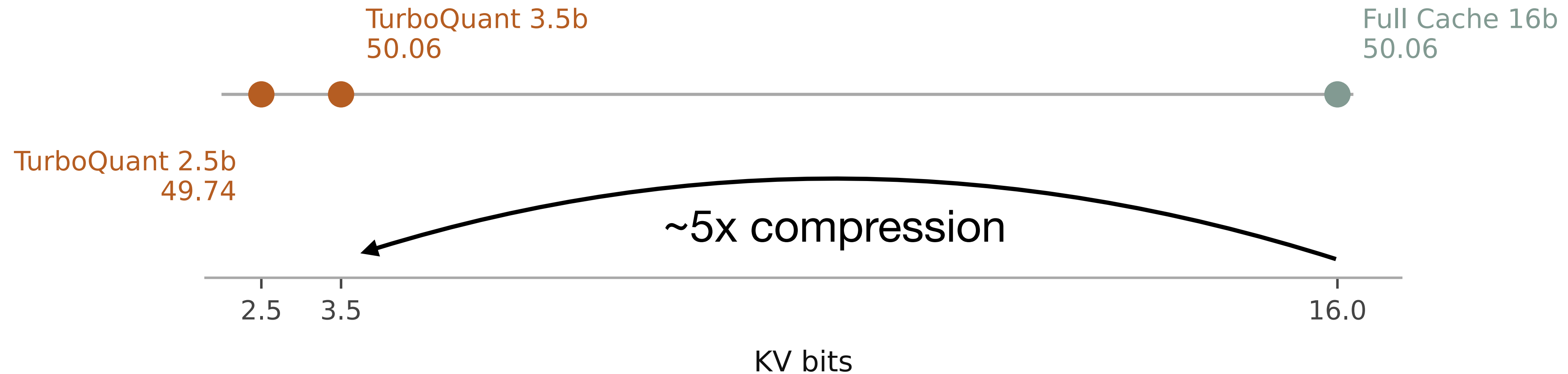
<https://research.google/blog/turboquant-redefining-ai-efficiency-with-extreme-compression/>

Method 5 to reduce KV cache size



<https://research.google/blog/turboquant-redefining-ai-efficiency-with-extreme-compression/>

TurboQuant reaches 50.06 at 3.5-bit KV; Full Cache uses 16-bit KV



More methods to **reduce KV cache size**:

- **Linear attention in Qwen3.5, Nvidia Nemotron 3**
<https://magazine.sebastianraschka.com/p/beyond-standard-lms>
- **k=v in Gemma 4**

More Things on the Horizon

**Beyond architecture and
training optimizations**

More Things on the Horizon

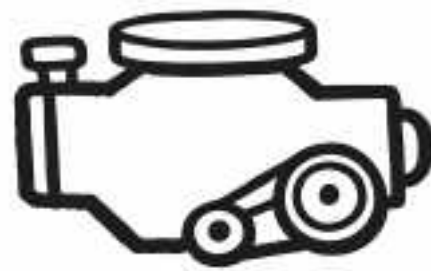


Conventional LLM



Reasoning LLM
(more expensive to run)

More Things on the Horizon



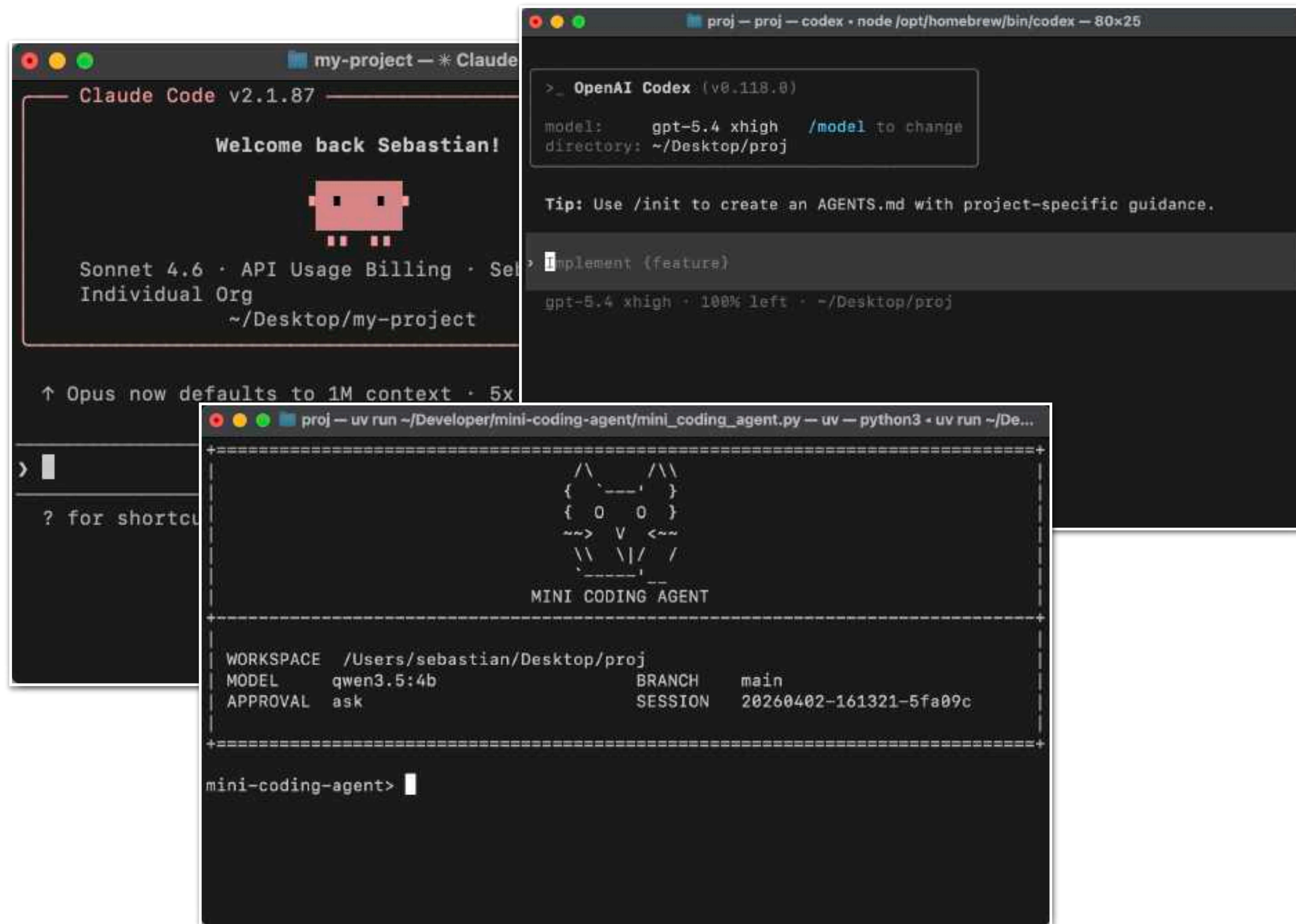
Conventional LLM



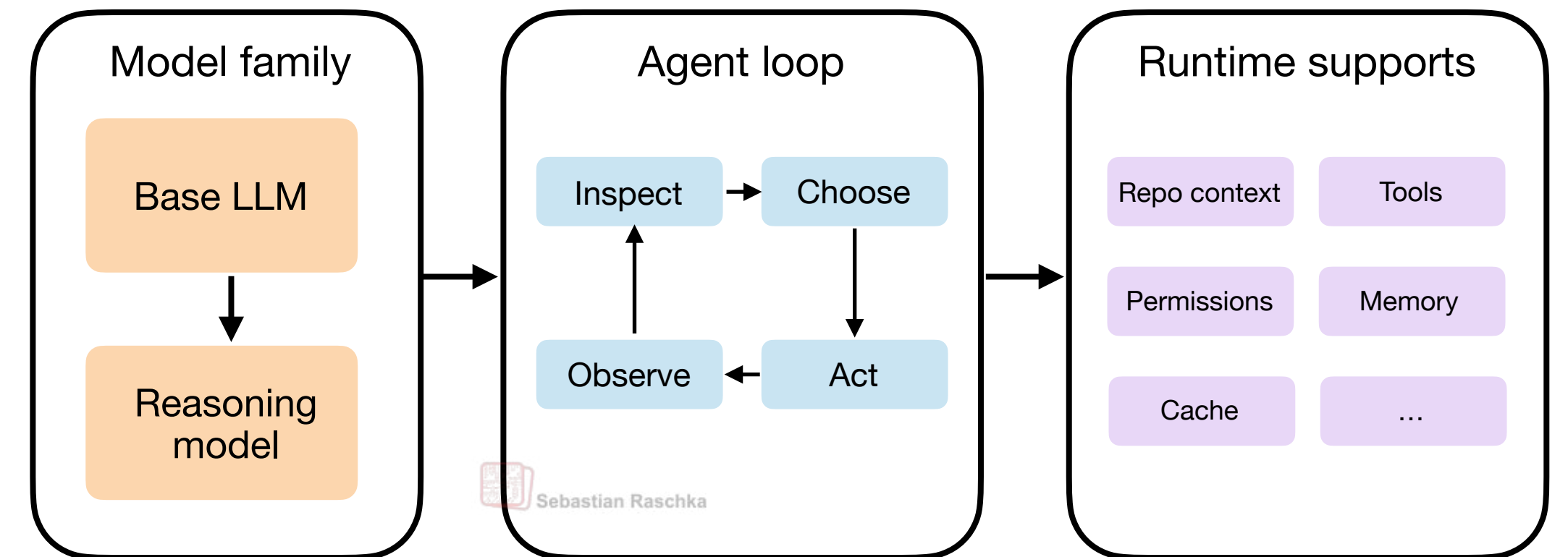
Reasoning LLM
(more expensive to run)



LLM in agent harness

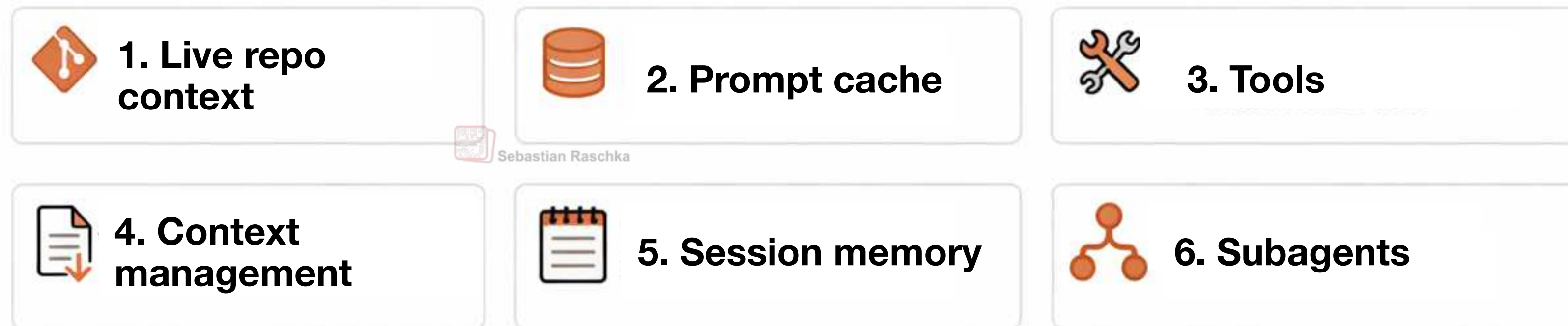


Coding harness



<https://magazine.sebastianraschka.com/p/components-of-a-coding-agent>

Components of a coding agent



<https://github.com/rasbt/mini-coding-agent>

“Getting started” tips

Beginner

Understanding how
LLMs work

Implementing a few
LLMs and training algos
from scratch

Beginner: Understand how things work

 **LLMs-from-scratch** Public Unpin Unwatch 761 Fork 13.8k Star 90.2k

main Go to file + Code **About** 

 **rasbt** Some gemma 3 improvements (#1000) 8447d70 · yesterday

Implement a ChatGPT-like LLM in PyTorch from scratch, step by step


 **reasoning-from-scratch** Public Unpin Unwatch 55 Fork 573 Star 4k


main Go to file + Code **About** 

 **rasbt** Add HF support (#231) a747c02 · 5 days ago

Implement a reasoning LLM in PyTorch from scratch, step by step

 **mini-coding-agent** Public Unpin Unwatch 9 Fork 108 Star 603

main 1 Branch 0 Tags Go to file + Code **About** 

 **pierre-allain-ds** and **rasbt** Fix spurious indentati... 717cae4 · 5 days ago 15 Commits

Minimal and readable coding agent harness implementation in Python to explain the core components of

“Getting started” tips

Beginner

Understanding how LLMs work

Implementing a few LLMs and training algos from scratch

Intermediate

Wrapper libraries to access more complex algos

Scale to small set of GPUs

Intermediate: Wrappers that make advanced features easier to use

The image displays three GitHub repository pages, each showing the repository name, owner, and various statistics. The first page is for 'transformers' by 'huggingface', showing 1.1k issues and 1.3k pull requests. The second page is for 'torchtitan' by 'pytorch', showing 209 issues and 274 pull requests. The third page is for 'trl' by 'huggingface', showing 544 issues and 121 pull requests. Each page also displays the number of watchers, forks, and stars.

Repository	Owner	Issues	Pull requests	Watchers	Forks	Stars
transformers	huggingface	1.1k	1.3k	1195	32.8k	159k
torchtitan	pytorch	209	274	58	778	5.2k
trl	huggingface	544	121	101	2.6k	18k

“Getting started” tips

Beginner

Understanding how LLMs work

Implementing a few LLMs and training algos from scratch

Intermediate

Wrapper libraries to access more complex algos

Scale to small set of GPUs

Pro

Larger-scale, professional training runs

Inspect and adapt code bases used by teams that trained popular LLMs successfully

Pro: Project and production code bases

*most training stacks are not public



Allen AI's OLMo 3 (32B)

<https://github.com/allenai/olmo-core>

<https://github.com/allenai/open-instruct>



SmolLM3 3B

Hugging Face's SmolLM3 (3B)

<https://github.com/huggingface/nanotron/tree/smollm3>

<https://github.com/huggingface/trl>



PRIME Intellect

Intellect-3 (106B MoE)

<https://github.com/PrimeIntellect-ai/prime-rl>

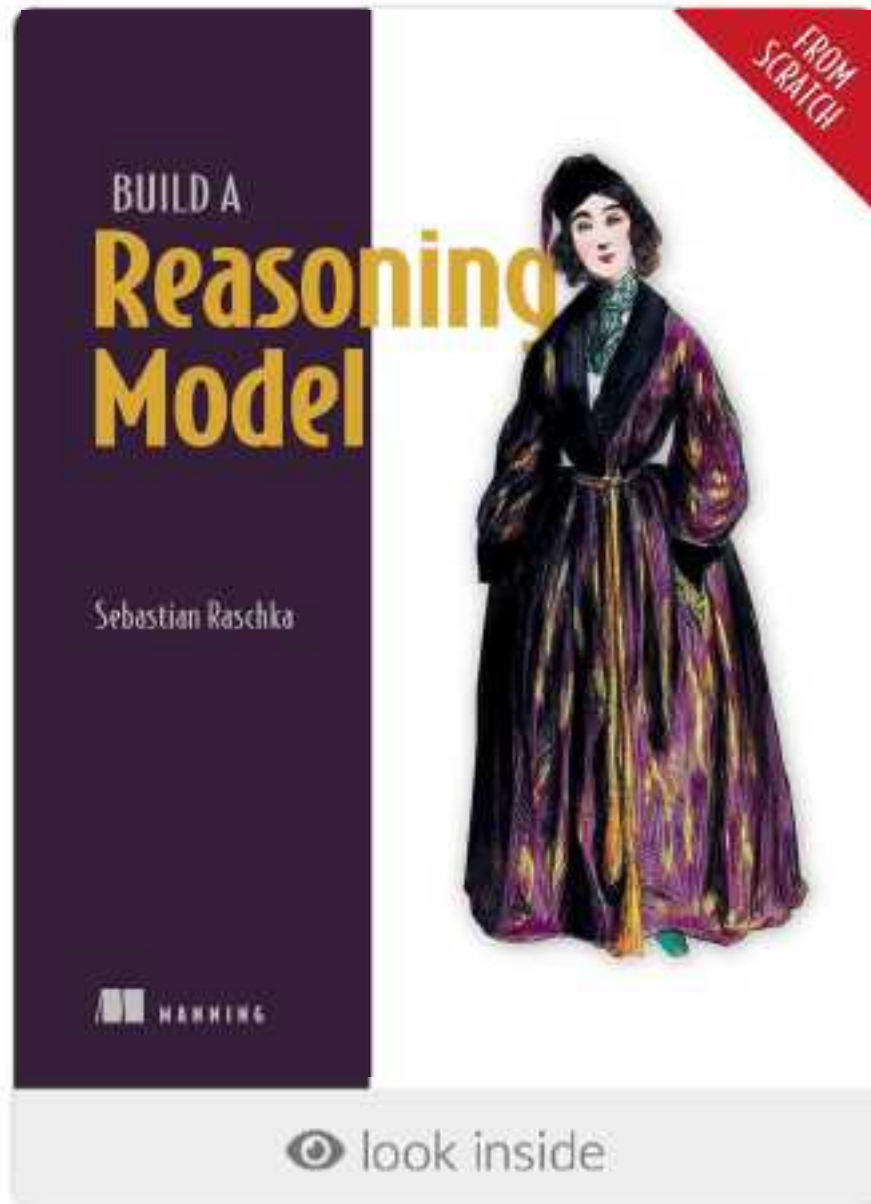


Nemotron 3

Nano (30B MoE) & Super (120B MoE)

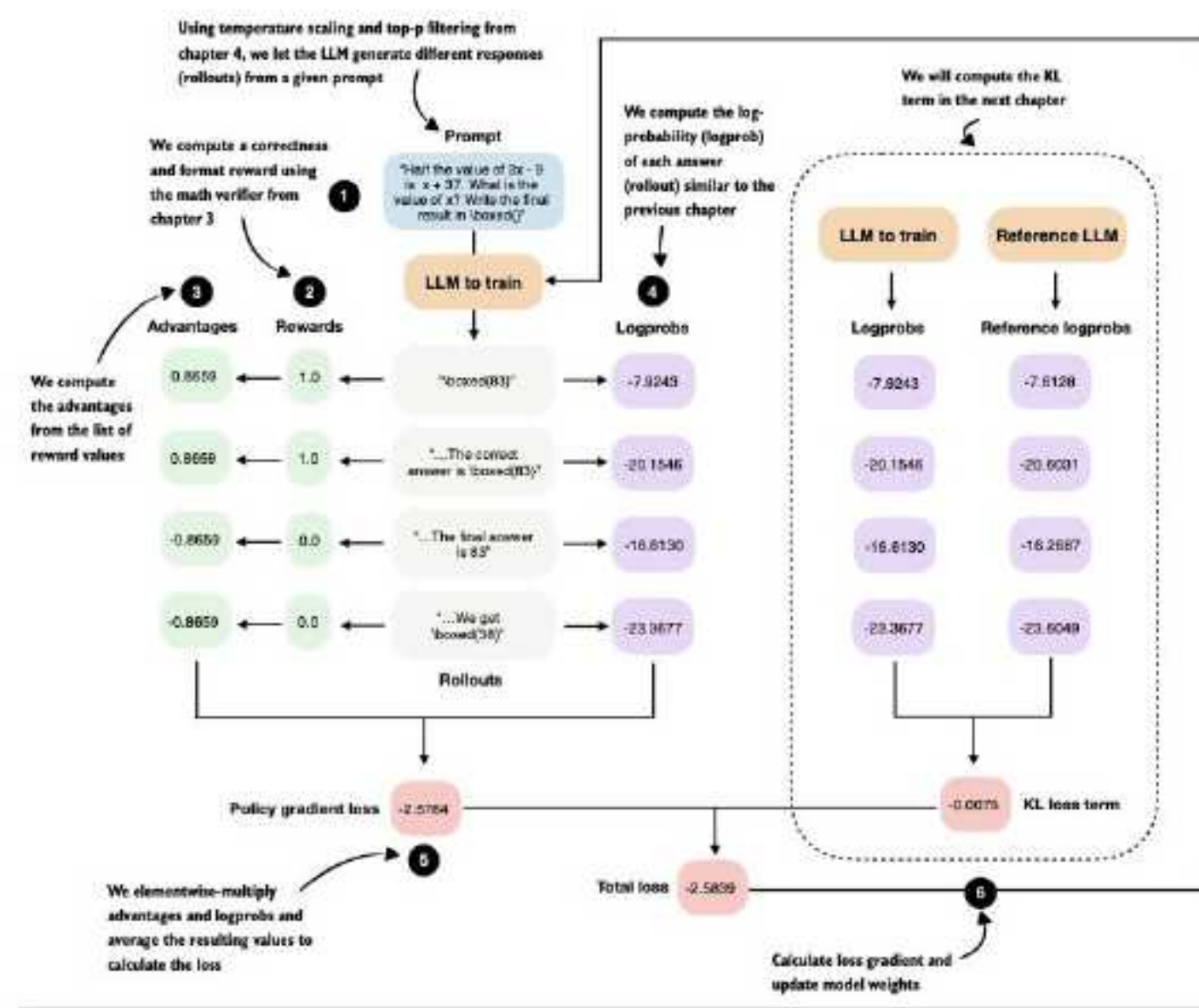
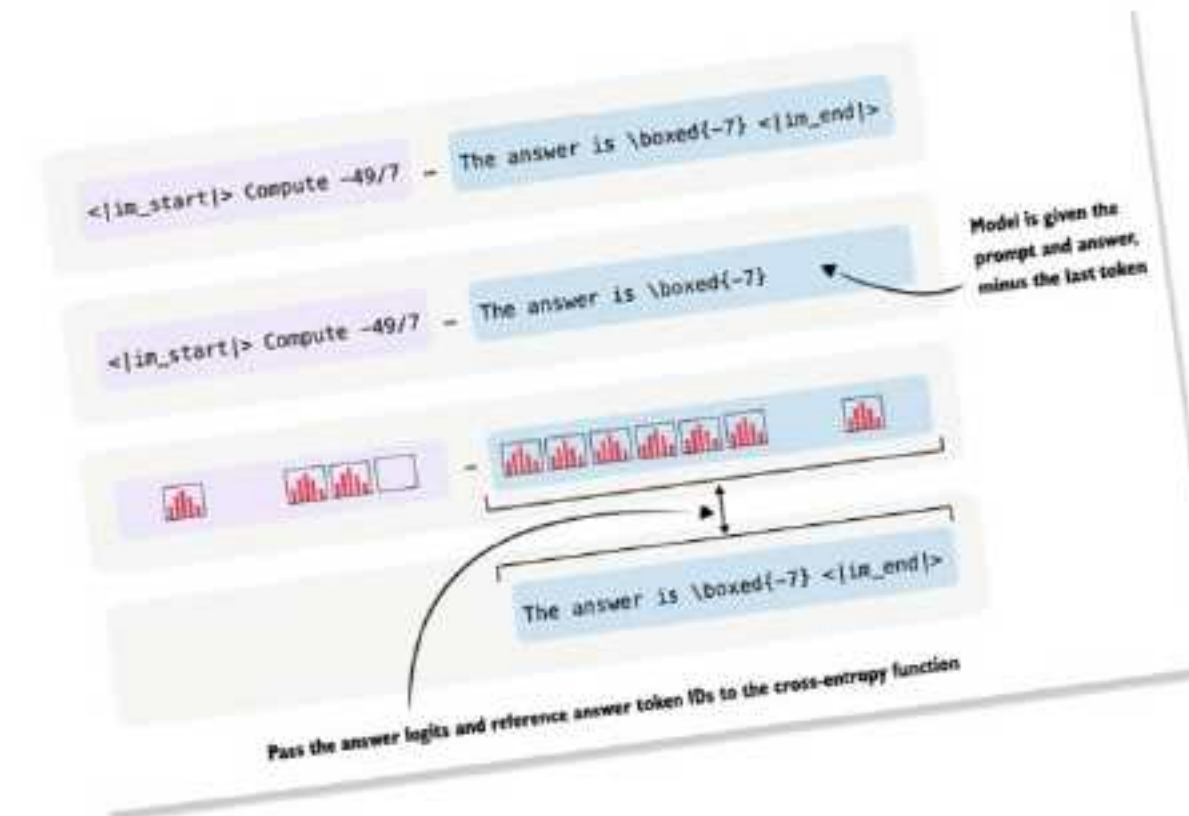
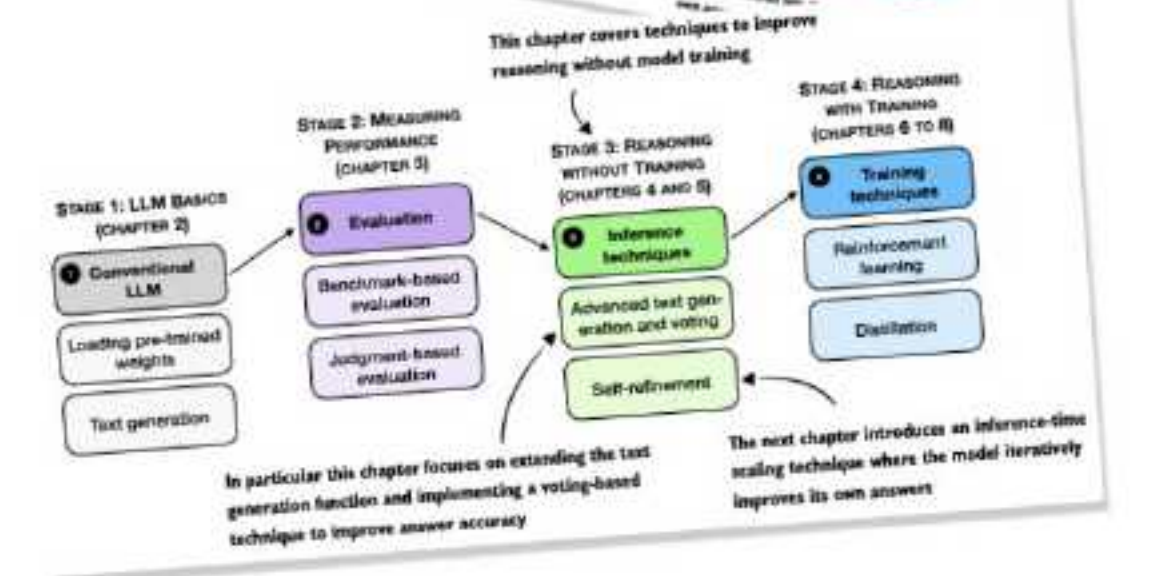
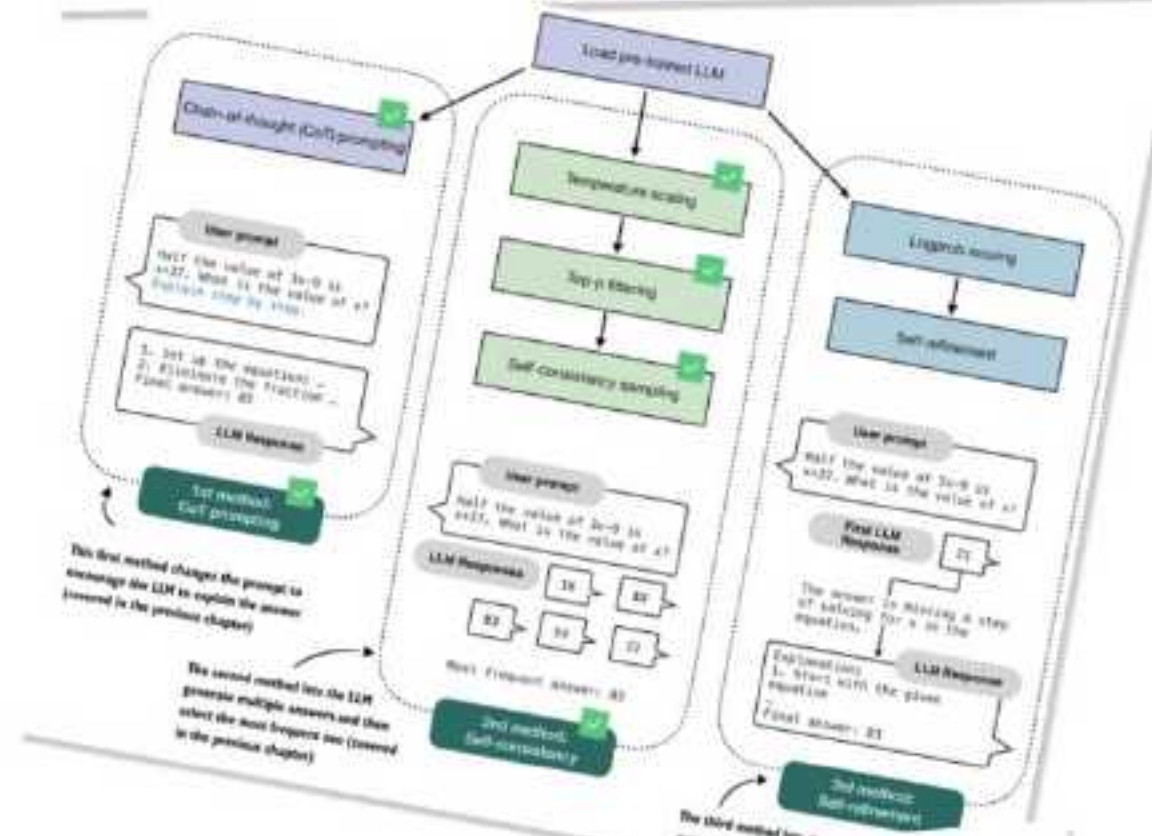
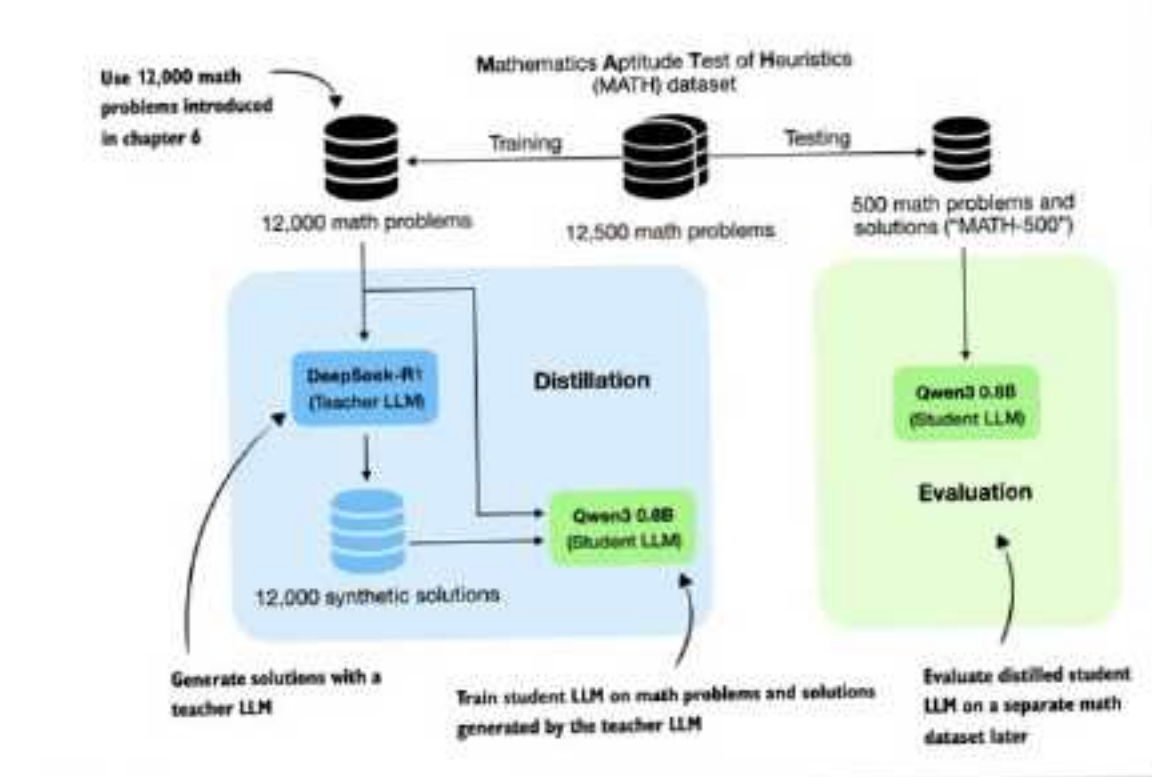
<https://github.com/NVIDIA-NeMo/Nemotron>

<https://github.com/NVIDIA/Megatron-LM>



Manning Early Access Program (MEAP) Read chapters as they are written, get the finished eBook as soon as it's ready, and receive the pBook long before it's in bookstores.

all chapters available



- Amazon (pre-order), <https://amzn.to/4aAKiFY>
- Manning, <https://mng.bz/Nwr7> (complete book in [early access](#), pre-final layout, 528 pages)

Book signing at 16:00

