

SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning

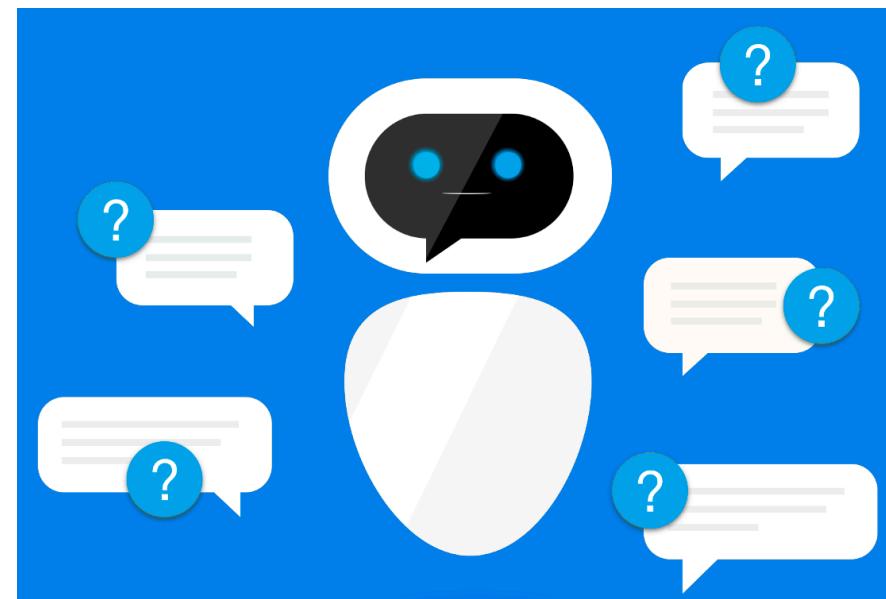
Hanrui Wang, Zhekai Zhang, Song Han

MIT HAN Lab

Massachusetts Institute of Technology

NLP is Ubiquitous

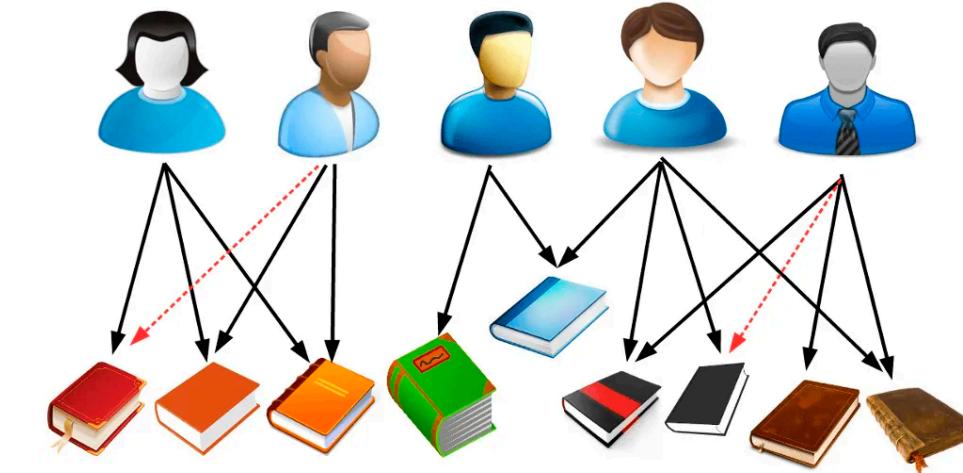
- NLP techniques are widely used



Chat Bots



Grammar Checking



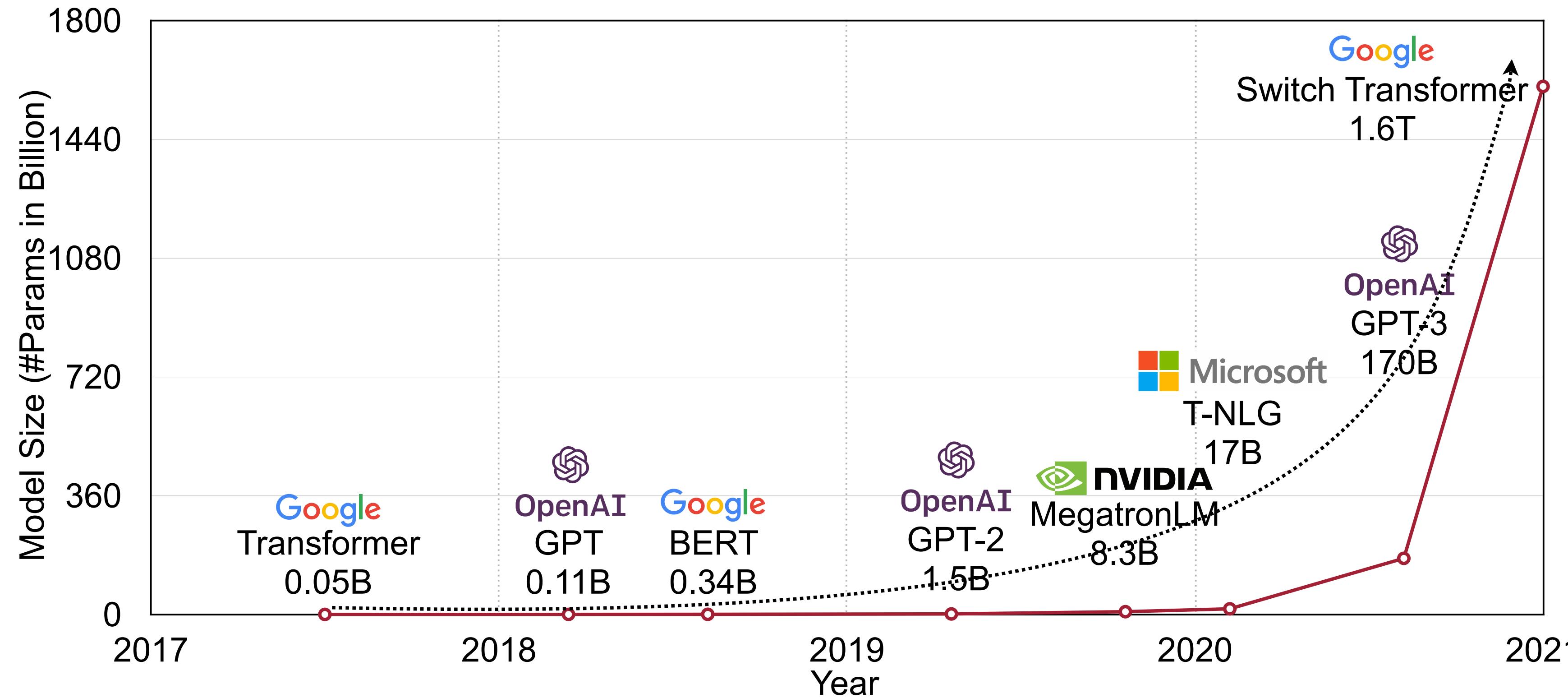
Recommender System



Machine Translation

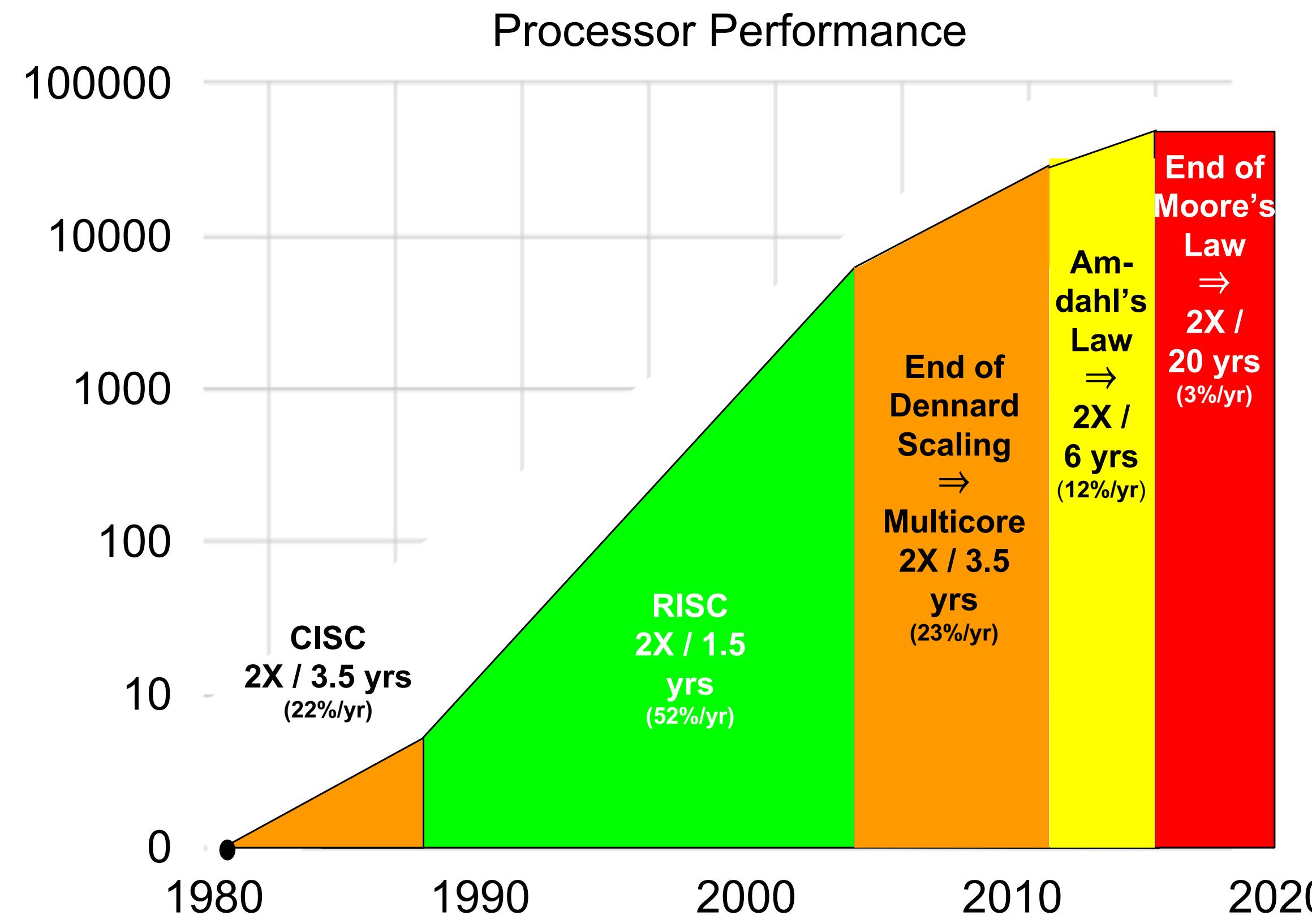
Efficient NLP is Important

- NLP model size and computation are increasing exponentially



Efficient NLP is Important

- End of Moore's Law
- Need specialized **efficient NLP algorithm-hardware co-design**



Outline

- Quick Overview
- Background
- Algorithmic Optimizations
- Hardware Architecture
- Evaluation
- Conclusion

Quick Overview — Cascade Token/Head Pruning

As a visual treat, the film is almost perfect.

11 Tokens ↓ 8 Heads

BERT Layer 1 (100% Computation & Memory Access)

As treat, film perfect.

5 Tokens ↓ 6 Heads

Layer 2 (34%)

film perfect

2 Tokens ↓ 4 Heads

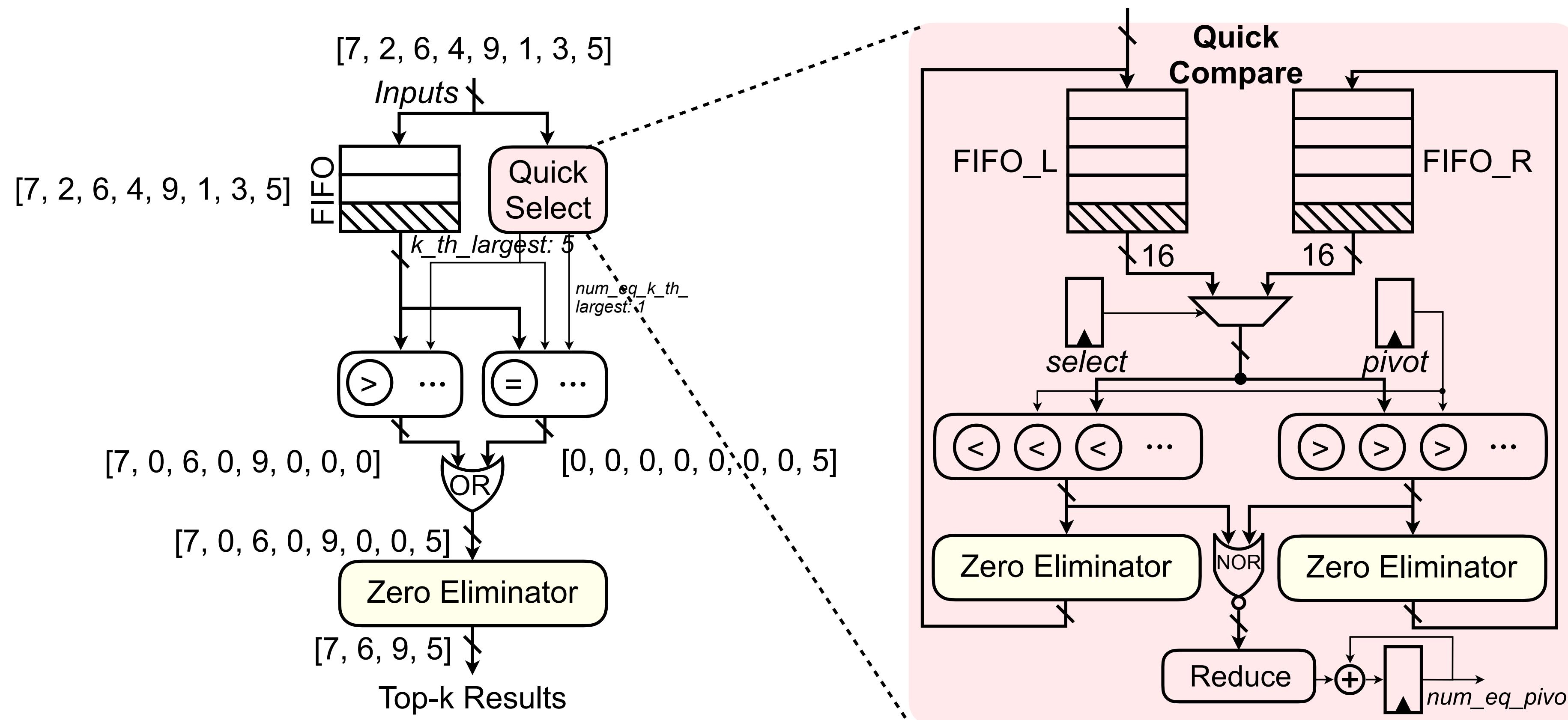
Layer 3 (9%)

Sentiment Classification: Positive ✓

- Intuition: human language contains high **redundancy**, remove unimportant tokens and heads

Quick Overview — Top-k Engine

- To support the fast selection of which tokens and heads to prune



Quick Overview — Progressive Quantization

- Reduce the DRAM access: eagerly fetch MSB; lazily fetch LSB
- An intuitive analogy:

Information in DRAM	MSB fetched to On-chip	Confidence	Need to fetch LSB?
“This is my favorite computer program”	“Ths is my favrit cmprtr prog”	High	No
“I like the great visual treat”	“I lk te gt vl trt”	Low	Yes

Outline

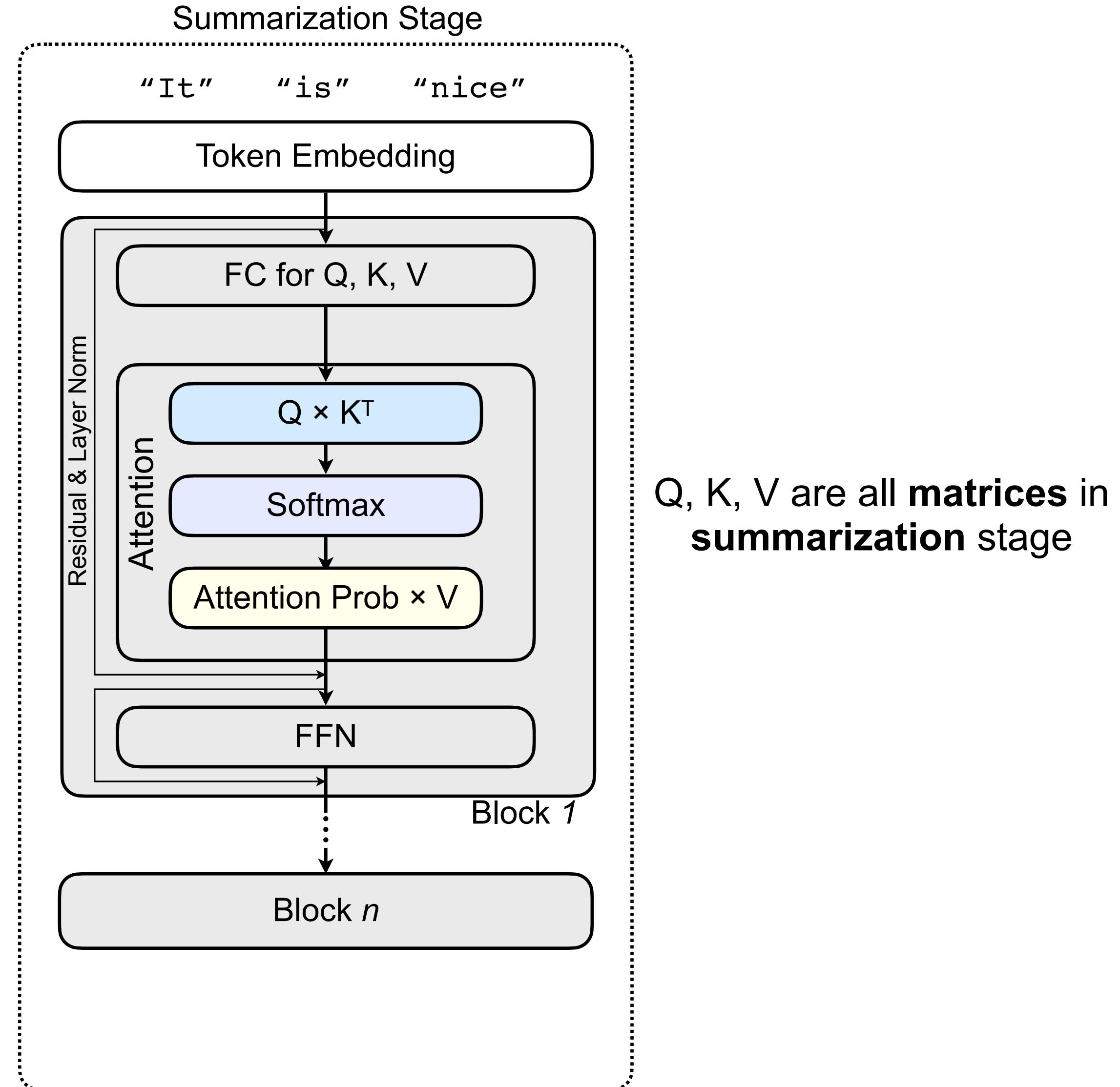
- Quick Overview
- Background
- Algorithmic Optimizations
- Hardware Architecture
- Evaluation
- Conclusion

Attention-Based NLP Models

- Discriminative Model
 - BERT
 - Summarization Stage
- Generative Model
 - GPT-2
 - Summarization Stage
 - Generation Stage

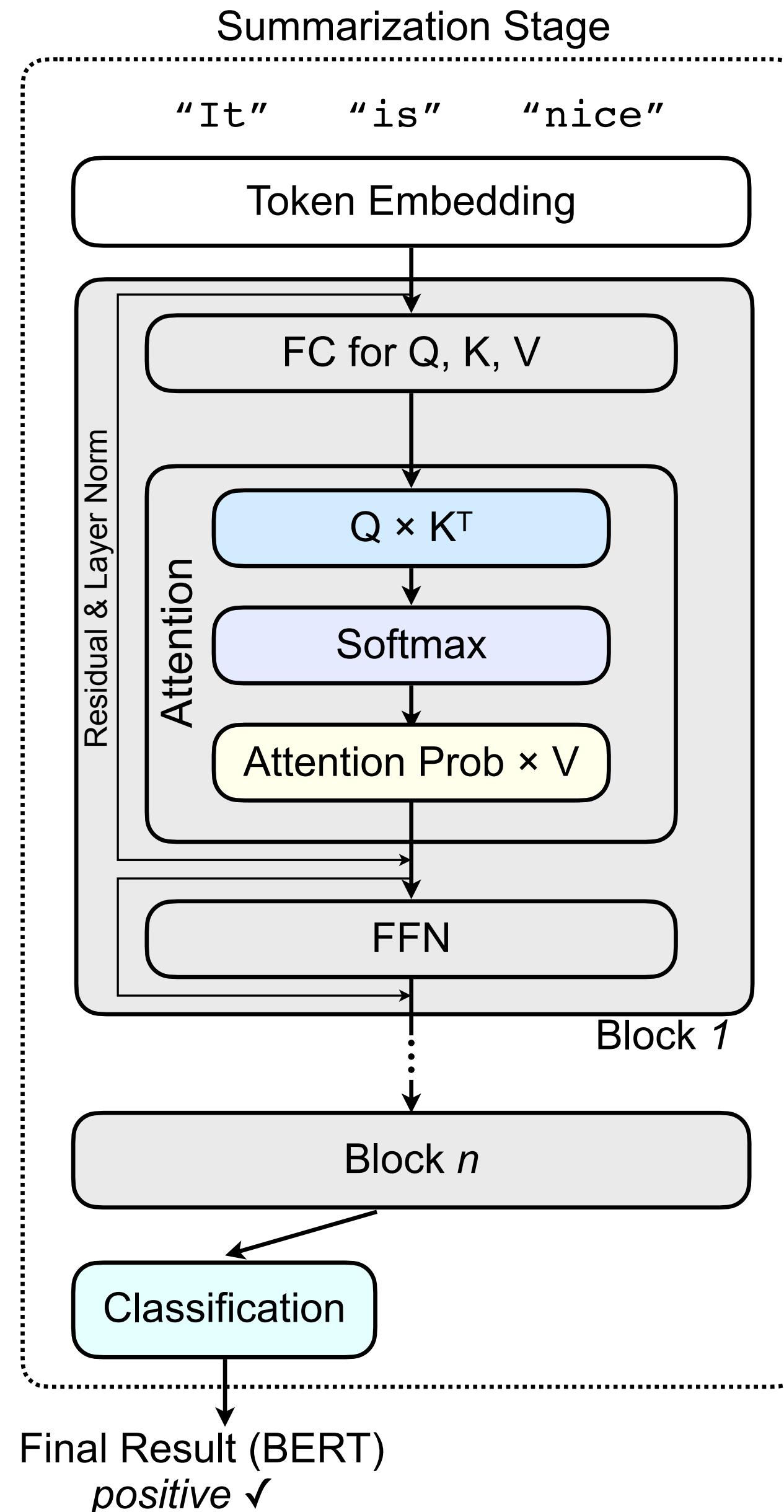
Summarization Stage

- Discriminative Model
 - BERT
 - Summarization Stage
- Generative Model
 - GPT-2
 - Summarization Stage
 - Generation Stage



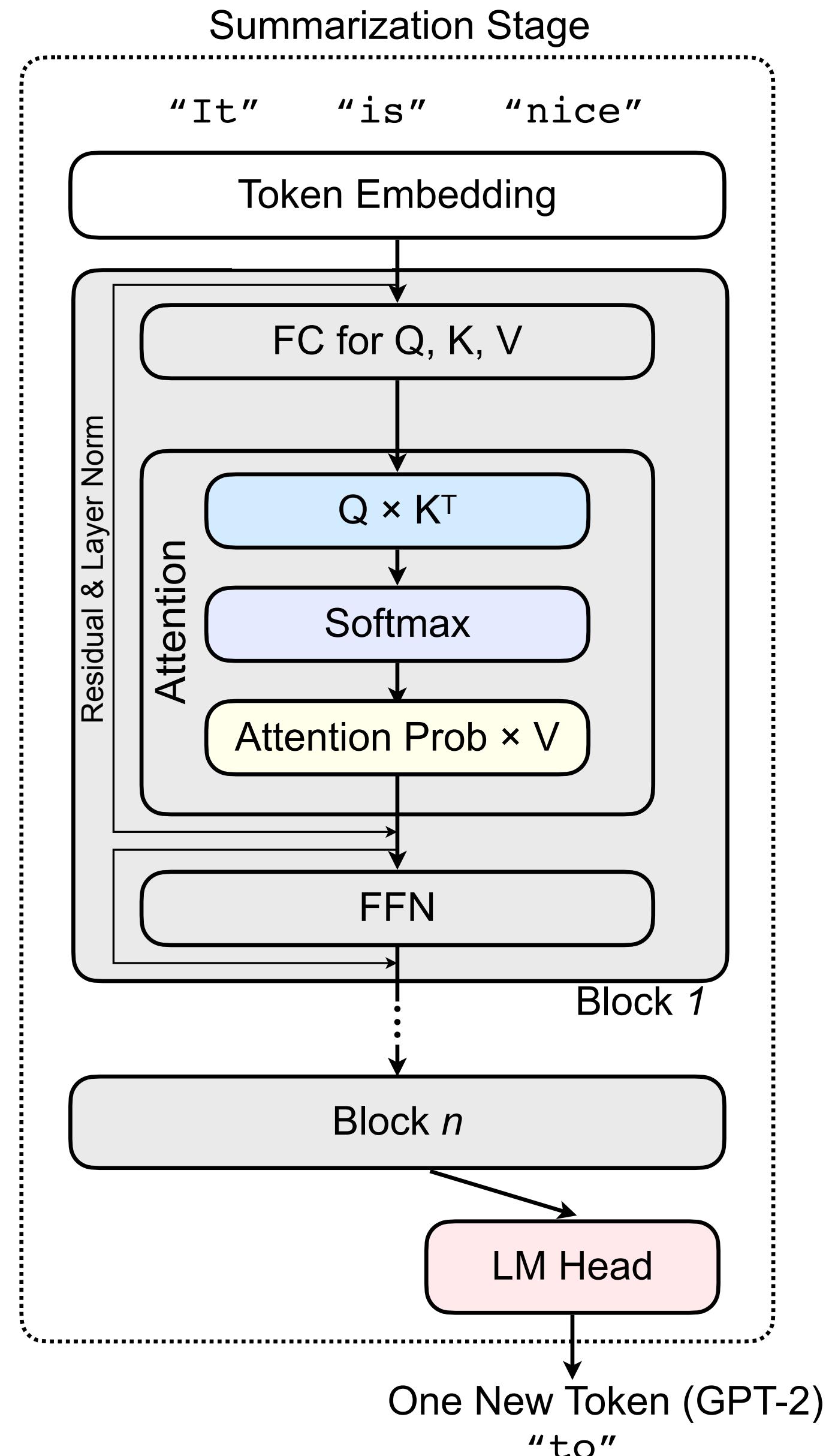
Summarization Stage

- Discriminative Model
 - BERT
 - Summarization Stage
- Generative Model
 - GPT-2
 - Summarization Stage
 - Generation Stage



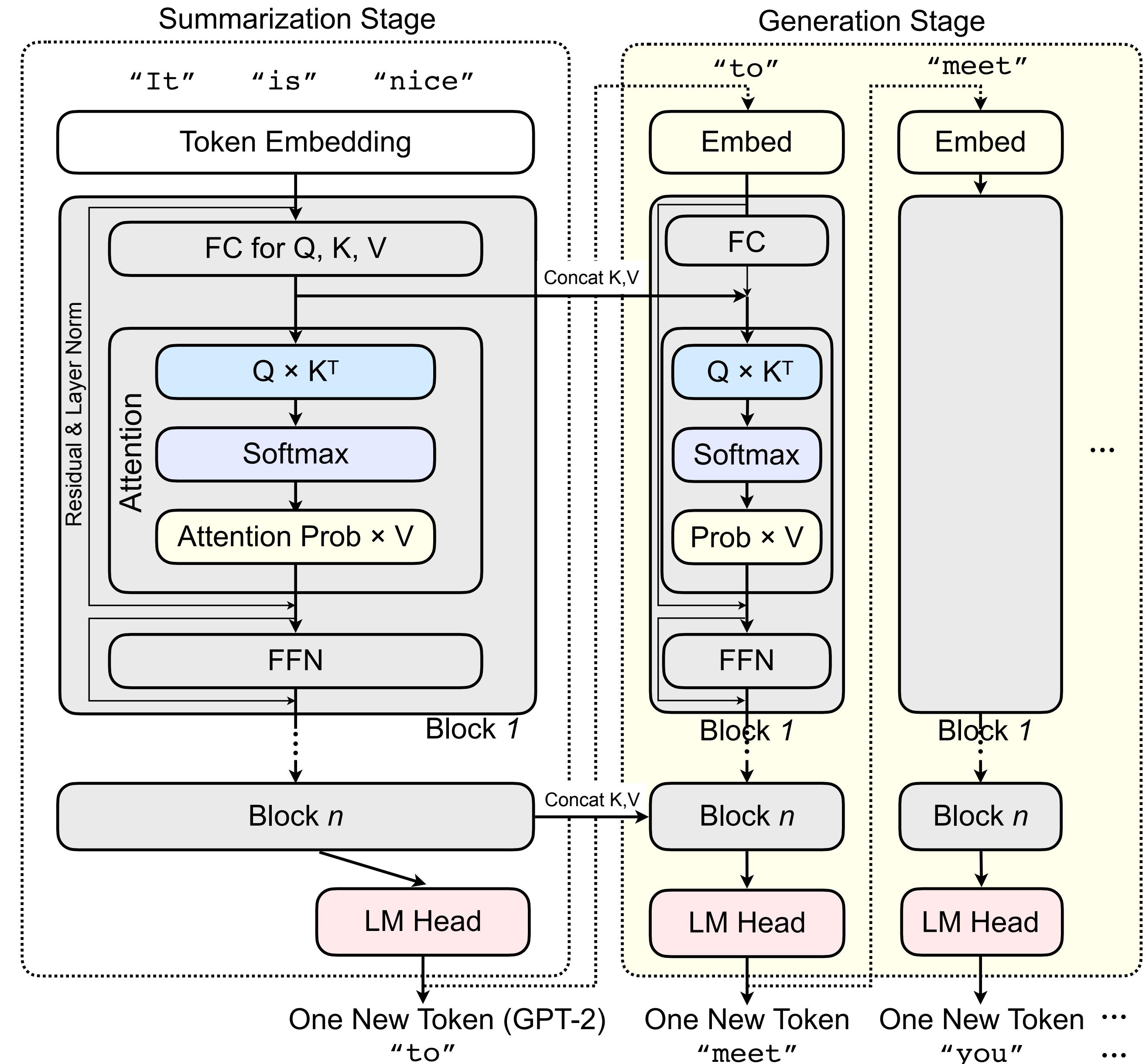
Summarization Stage

- Discriminative Model
 - BERT
 - Summarization Stage
- Generative Model
 - GPT-2
 - **Summarization Stage**
 - Generation Stage

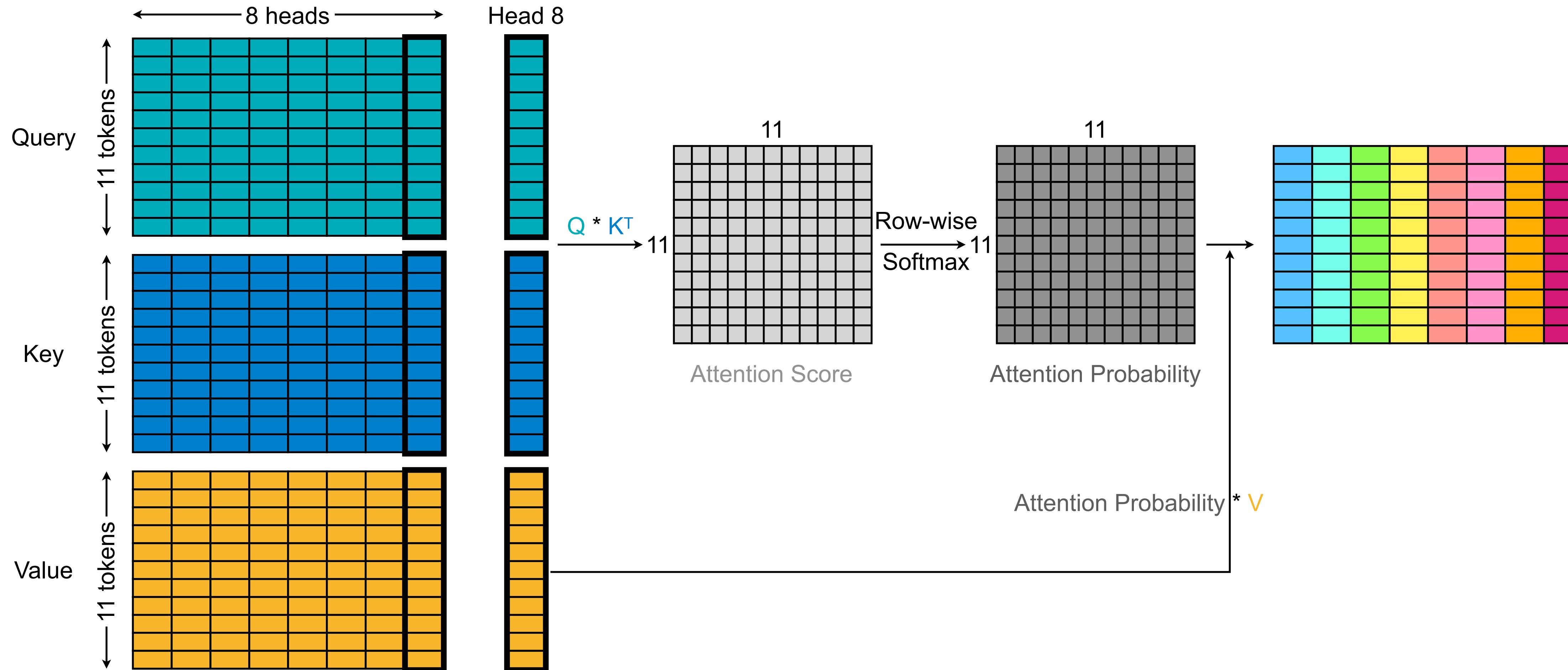


Generation Stage

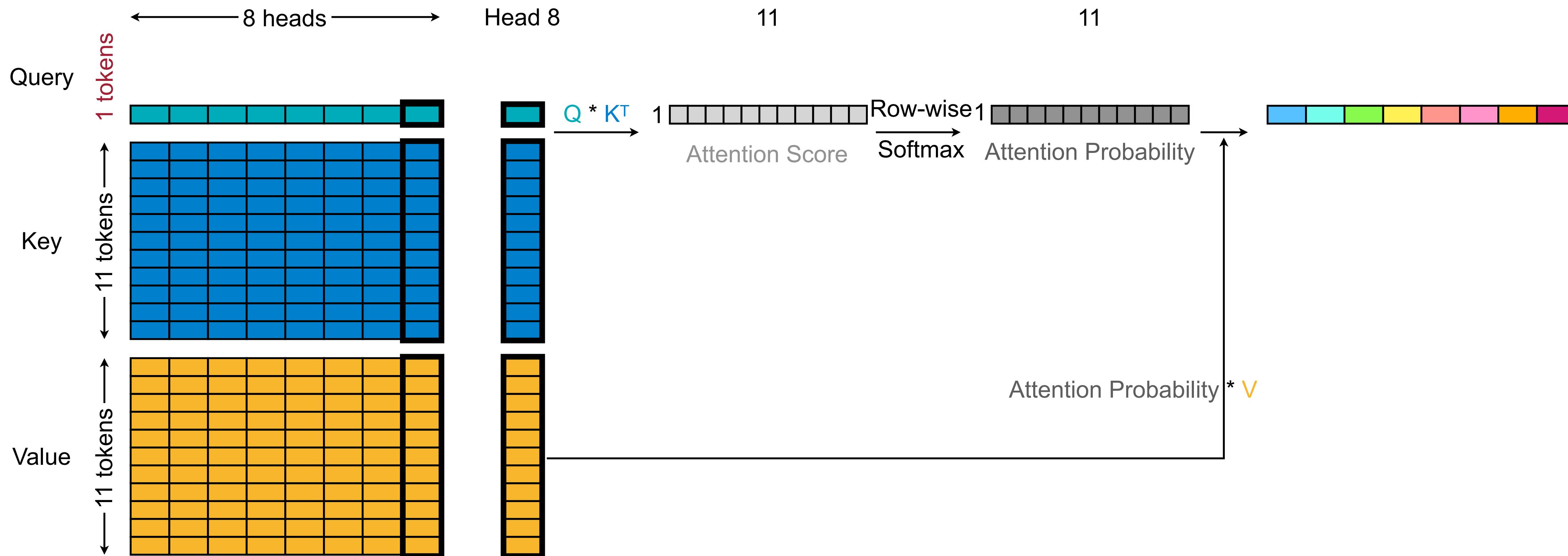
- Discriminative Model
 - BERT
 - Summarization Stage
- Generative Model
 - GPT-2
 - Summarization Stage
 - Generation Stage
- Block = Layer



Attention Layer - in Summarization Stage



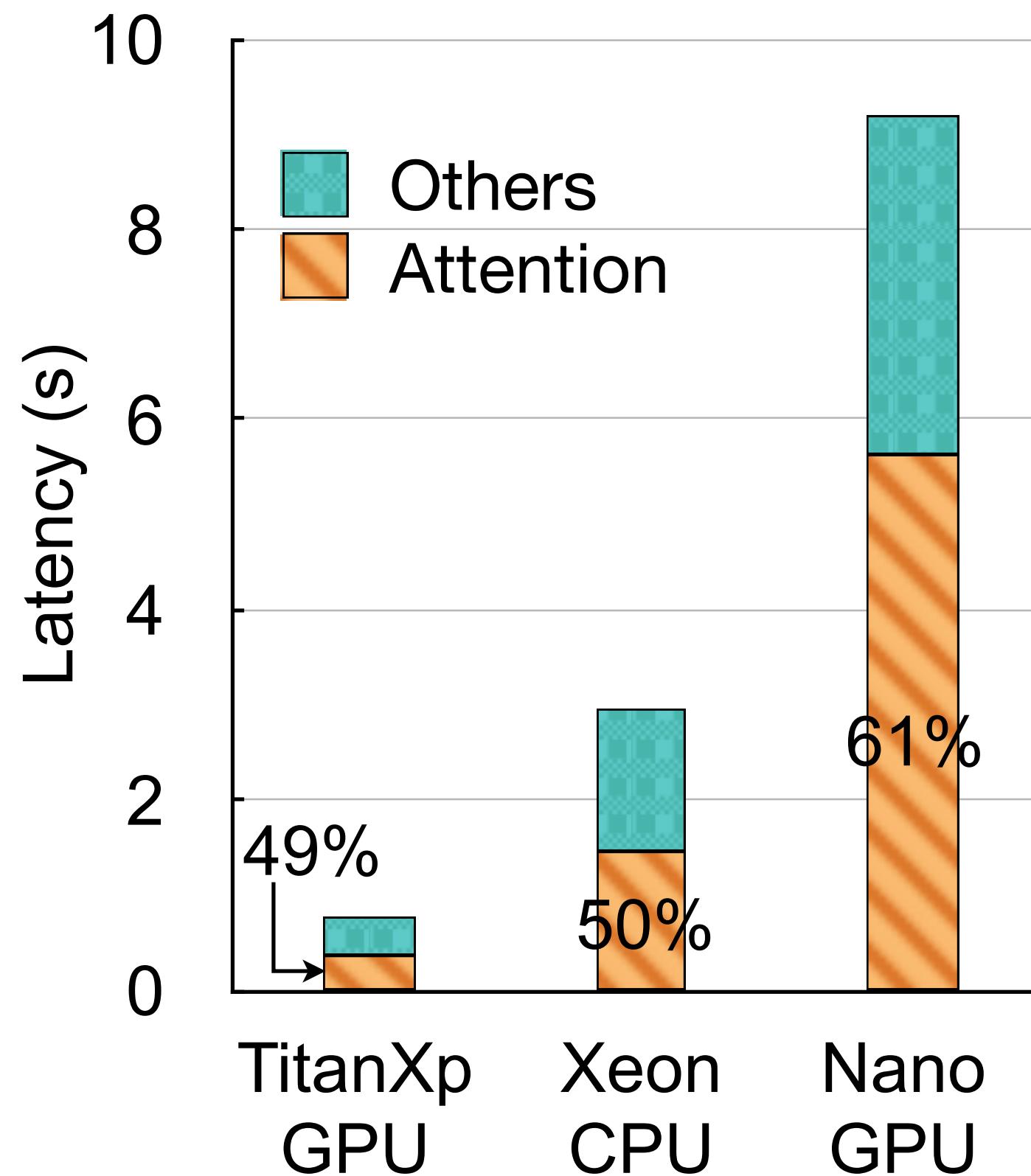
Attention Layer - in Generation Stage



- Different from Convolution or FC:
 - Query, Key, Value are **activations**

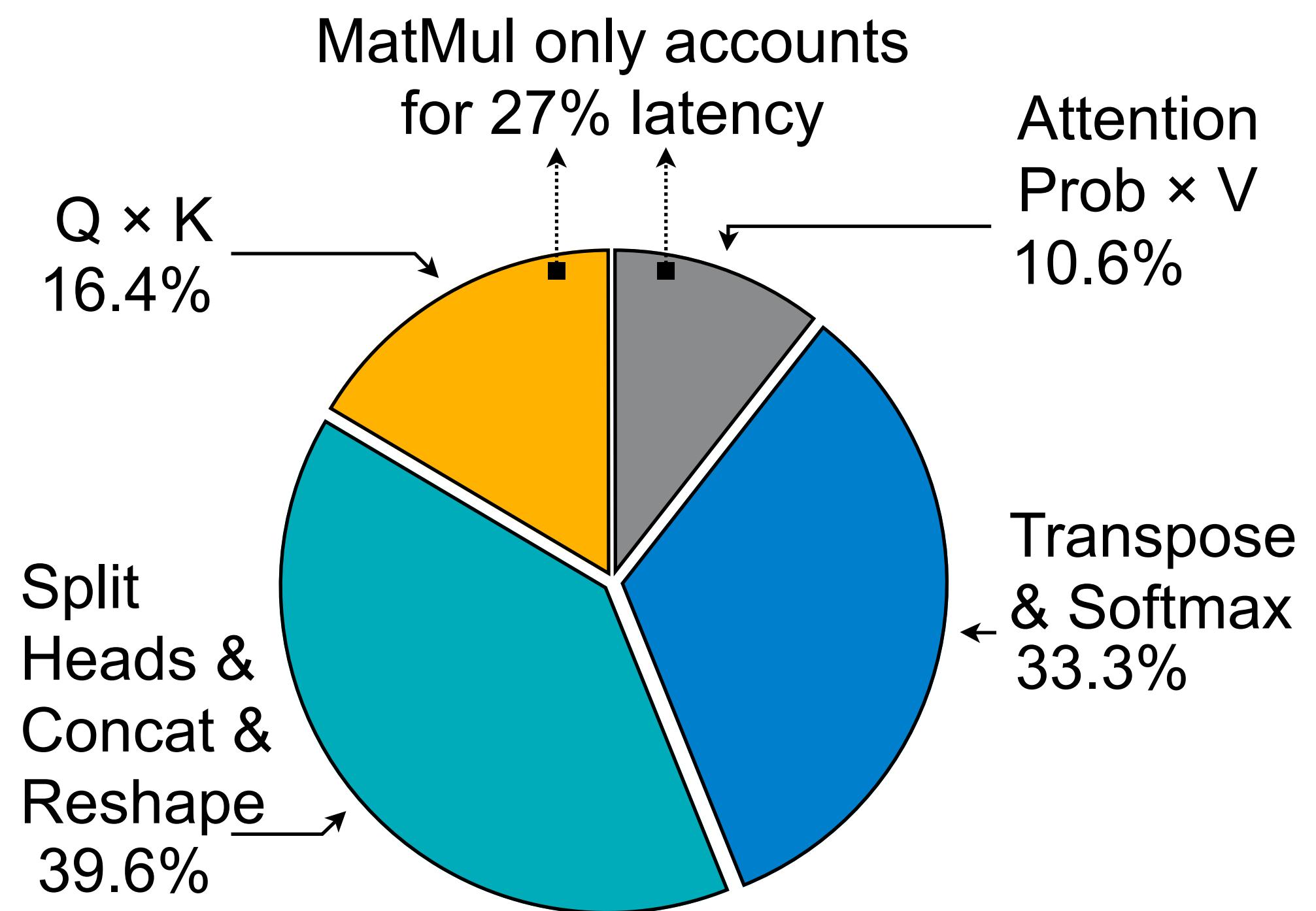
Attention Runs Slow

- End-to-End GPT-2 latency breakdown



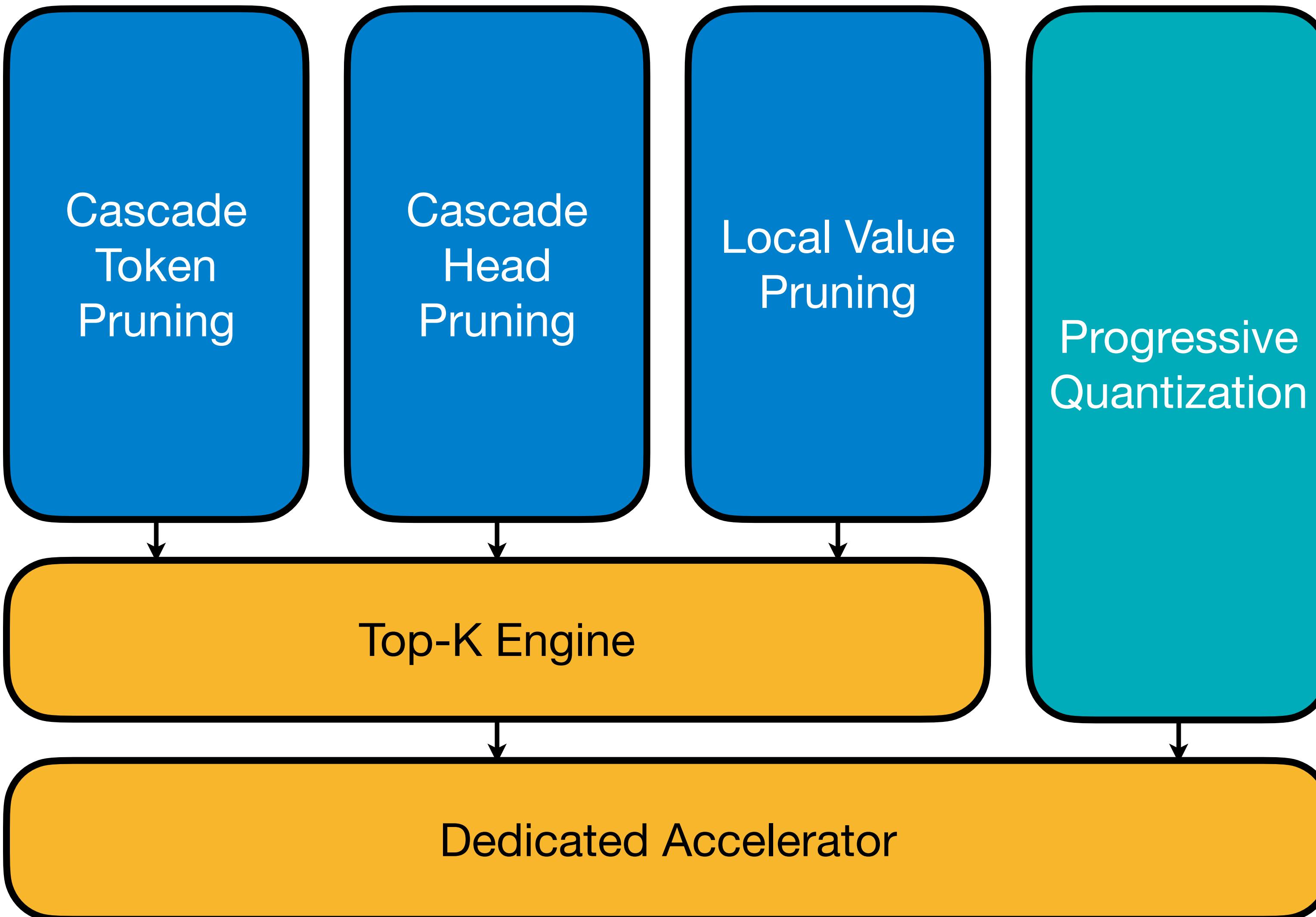
- Attention takes over 50% latency

- Attention latency breakdown



- Memory operations take over 70% latency

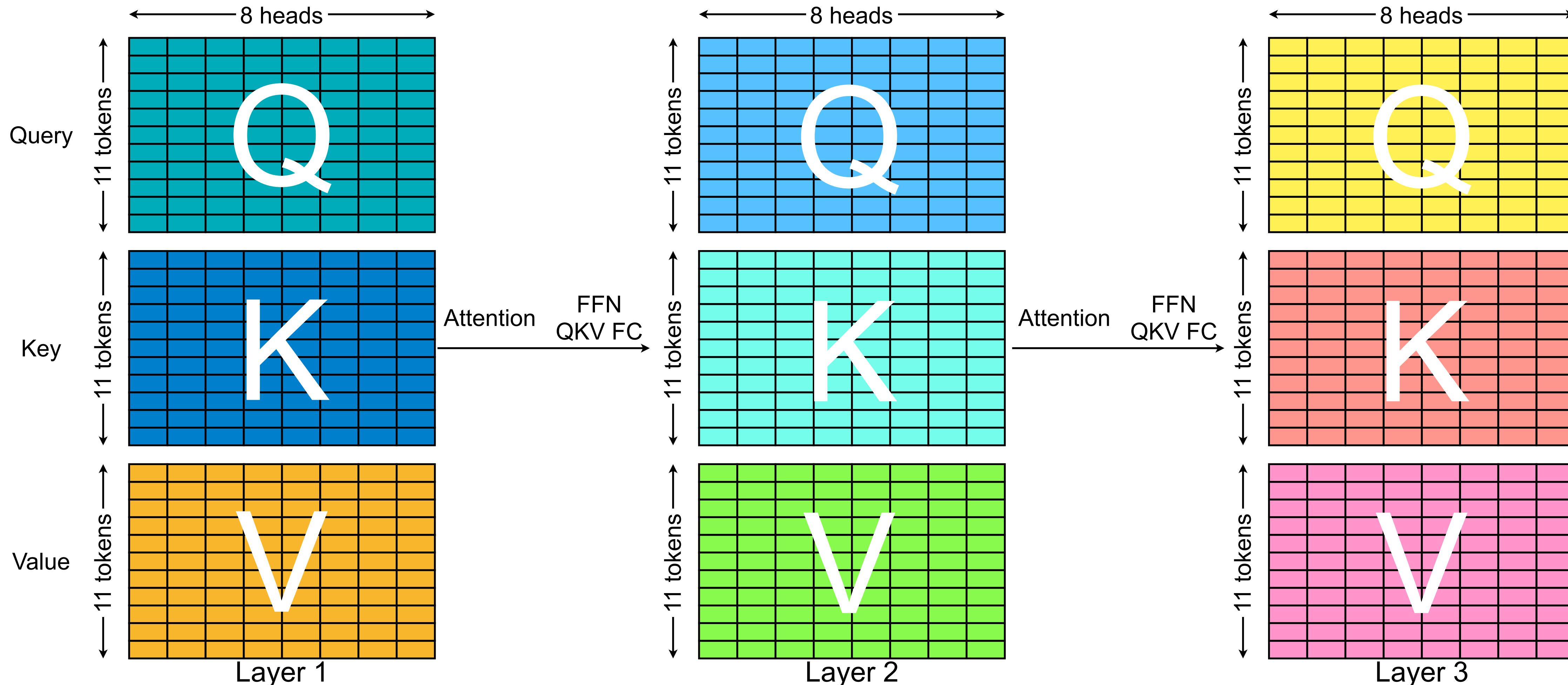
Our Solution: SpAtten



Outline

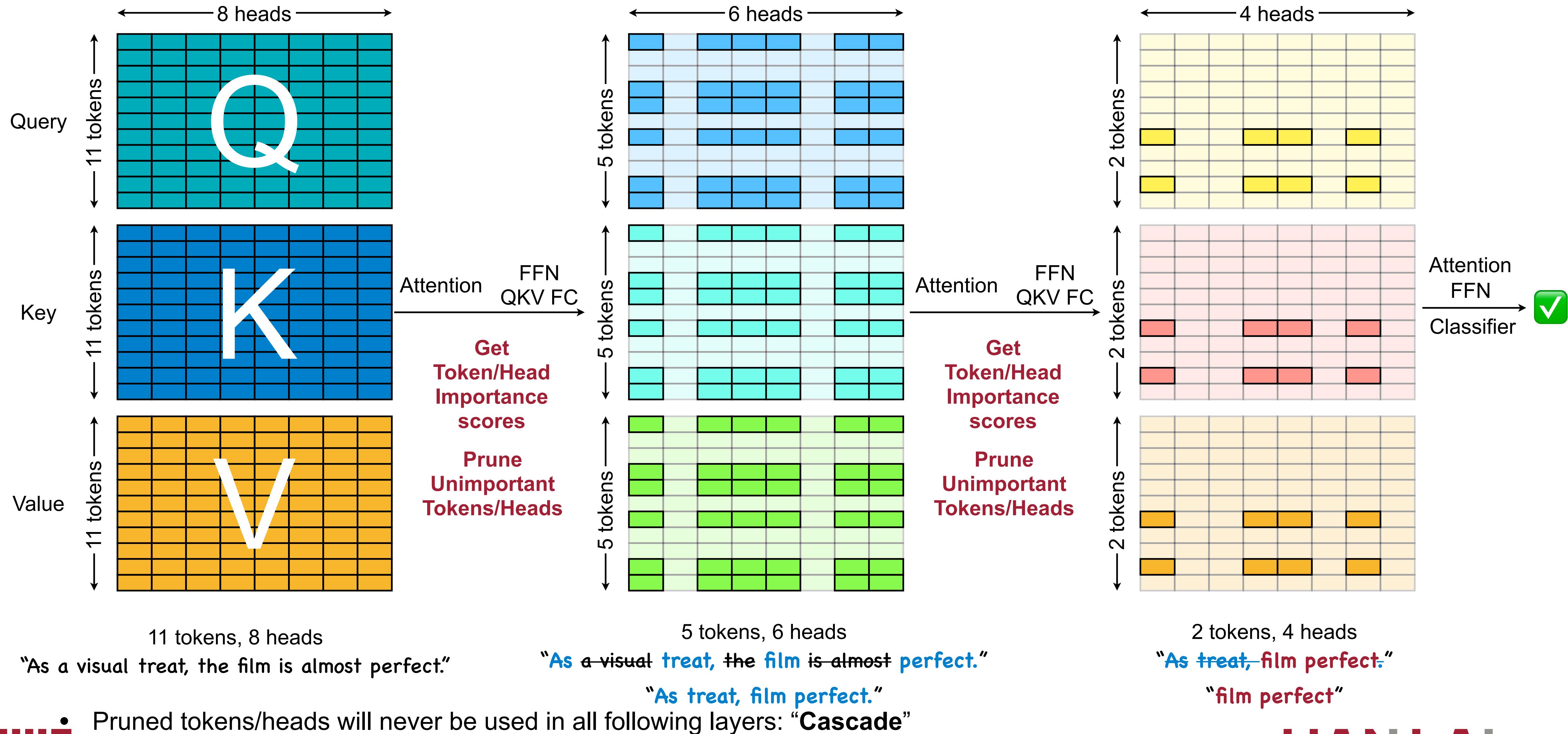
- Quick Overview
- Background
- **Algorithmic Optimizations**
- Hardware Architecture
- Evaluation
- Conclusion

Cascade Token/Head Pruning

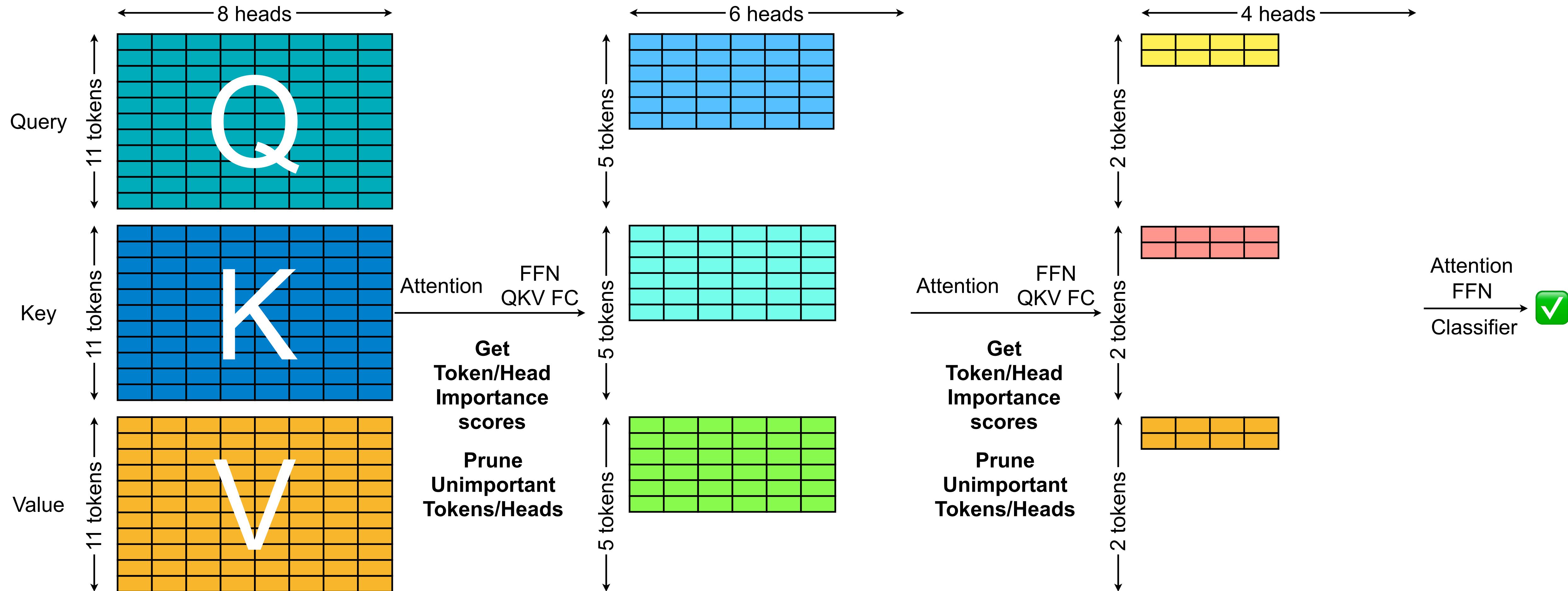


- Not all tokens/heads are created **equal**
- Find **unimportant** tokens and heads in **front** layers
- Remove them in **latter** layers

Cascade Token/Head Pruning

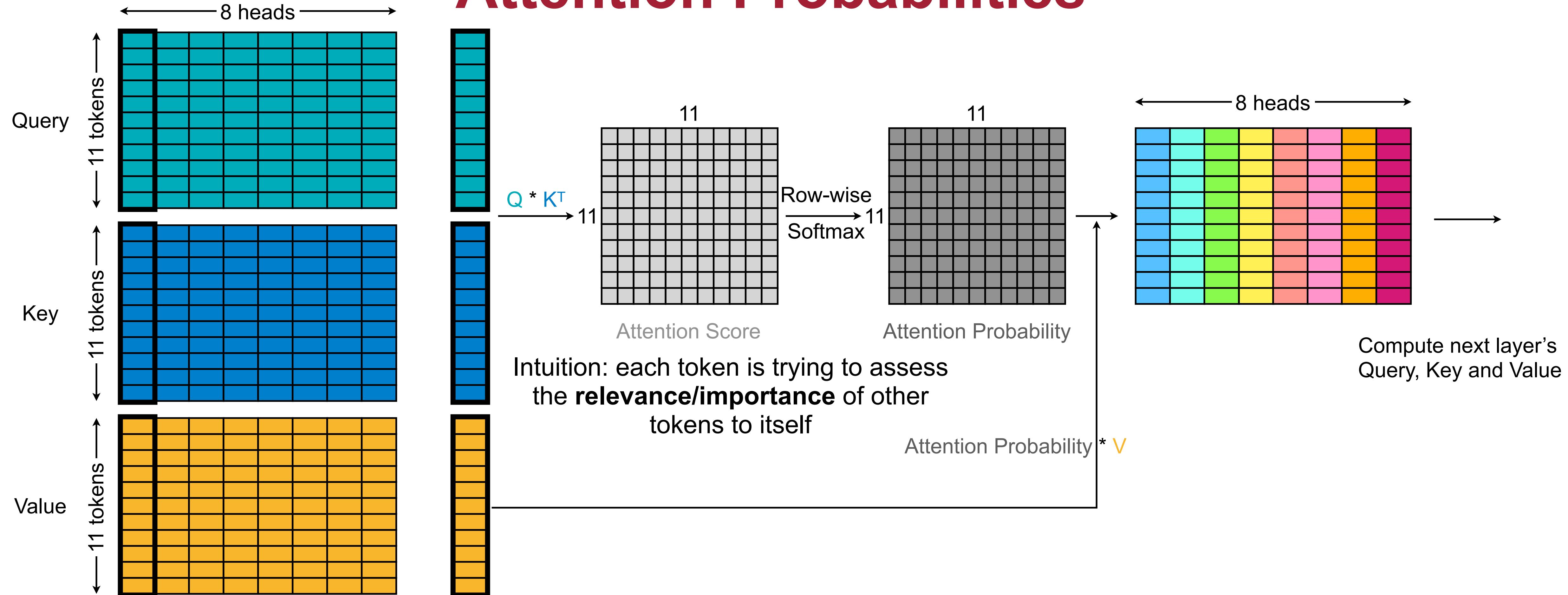


Cascade Token/Head Pruning



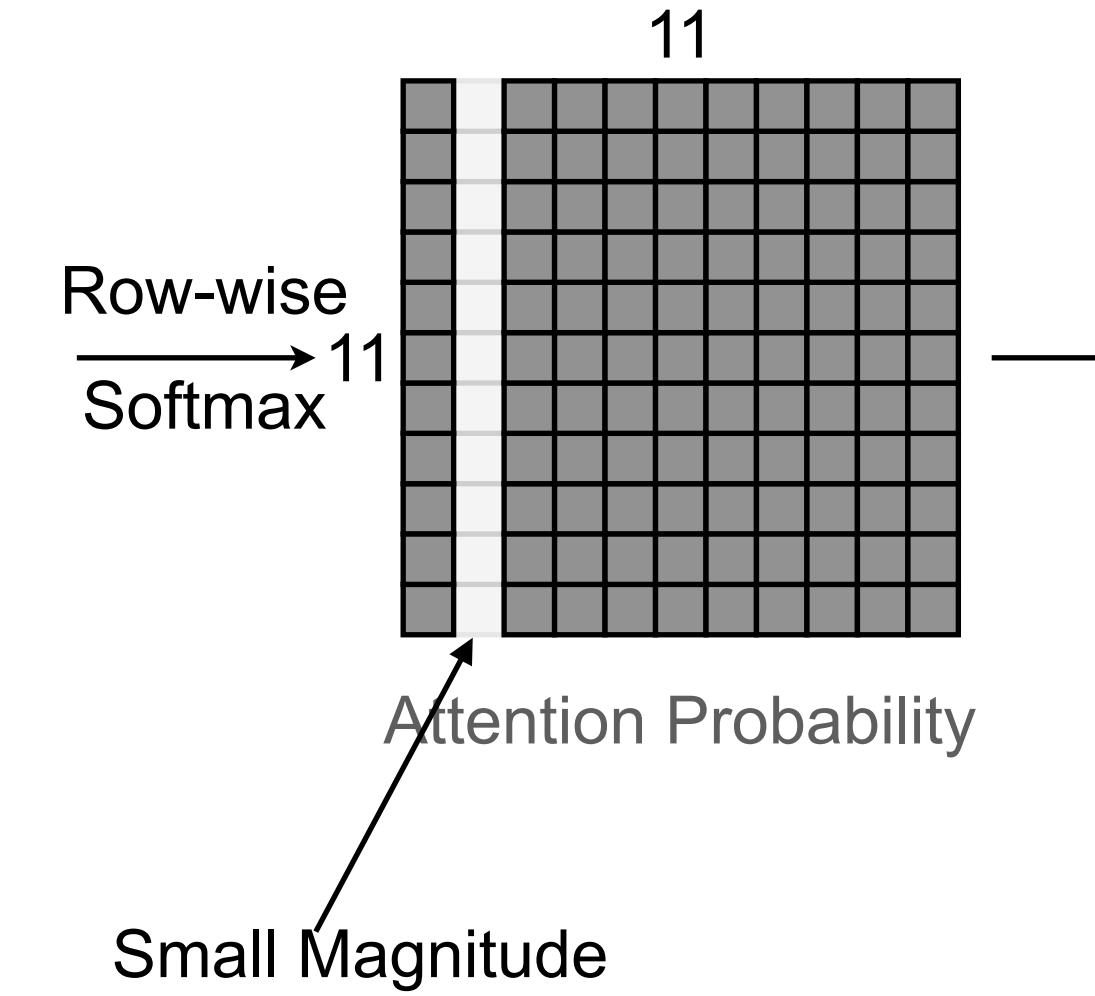
- Fundamentally **different** from weight pruning:
 - Query, Key, Value are **activations**
 - Pruned tokens/heads are **input-dependent**

Find Unimportant Tokens with Attention Probabilities



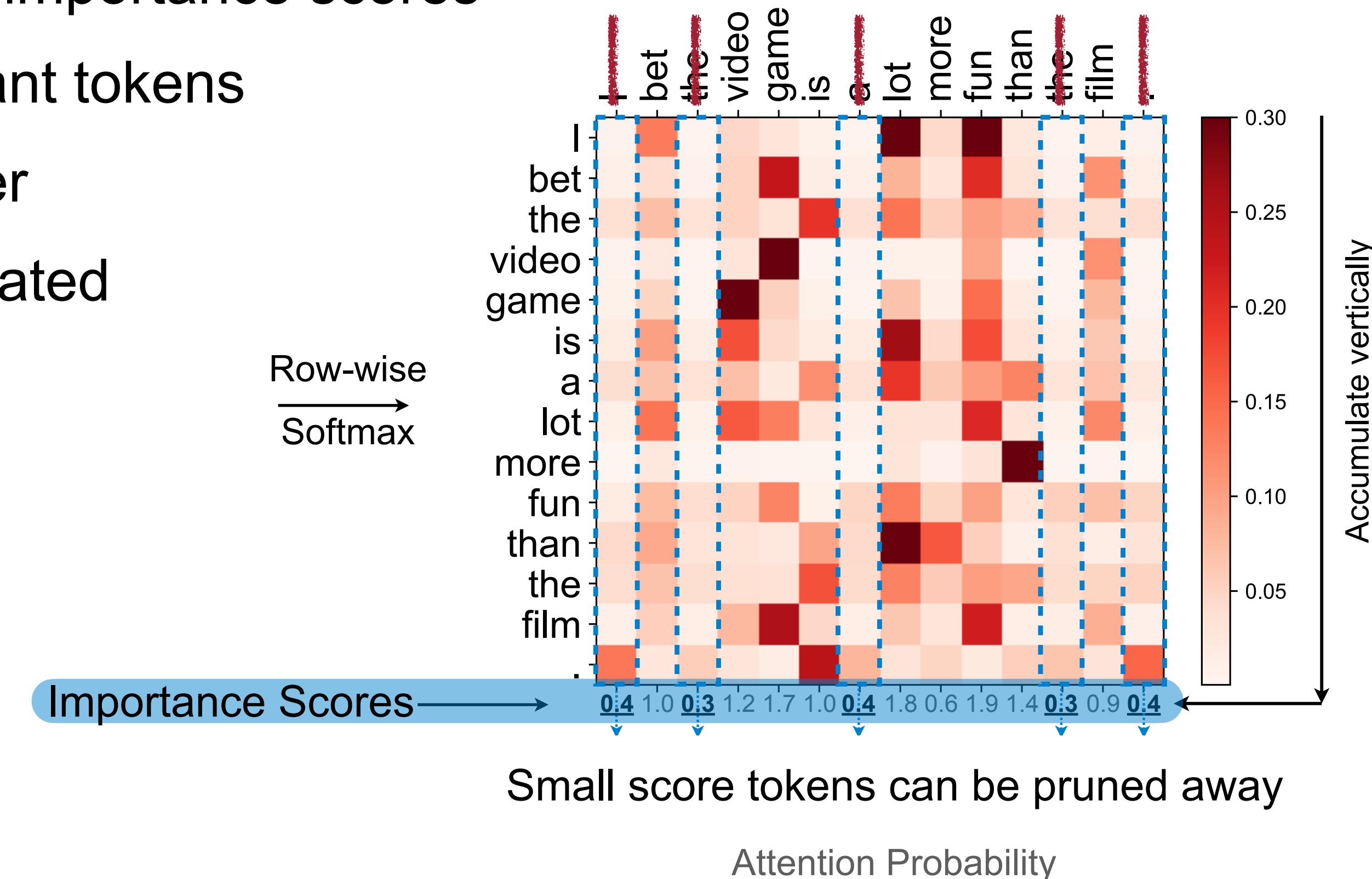
Find Unimportant Tokens with Attention Probabilities

- If one column in attention probability is **small**: the token is **unimportant** to all other tokens



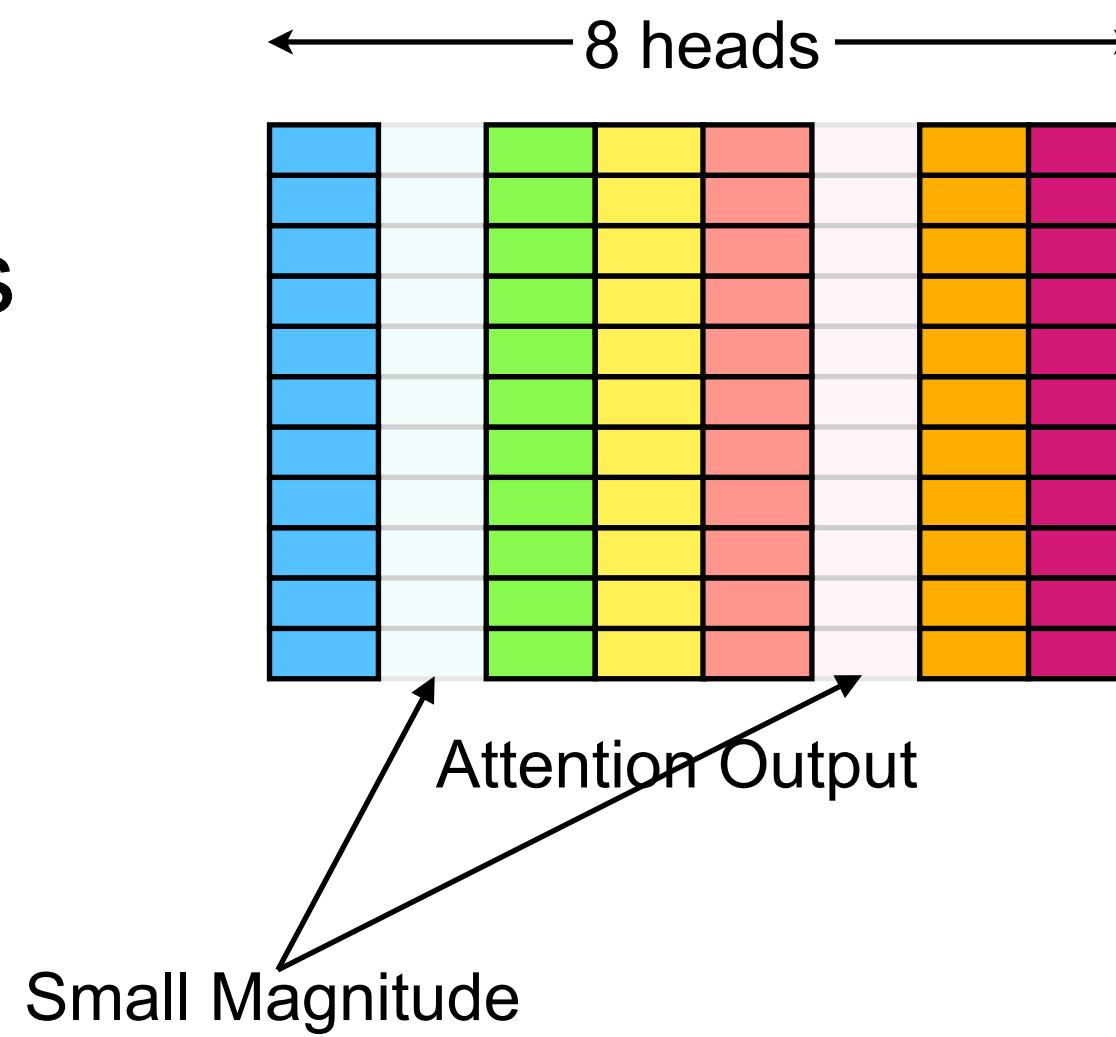
Find Unimportant Tokens with Attention Probabilities

- If one column in attention probability is **small**: the token is **unimportant** to all other tokens
- Maintain an **importance score** for each token
- **Accumulate** attention probs to the importance scores
- **Top-k** scores indicate top-k important tokens
 - Pruning ratio is a hyperparameter
- Importance scores can be accumulated **across heads and layers**



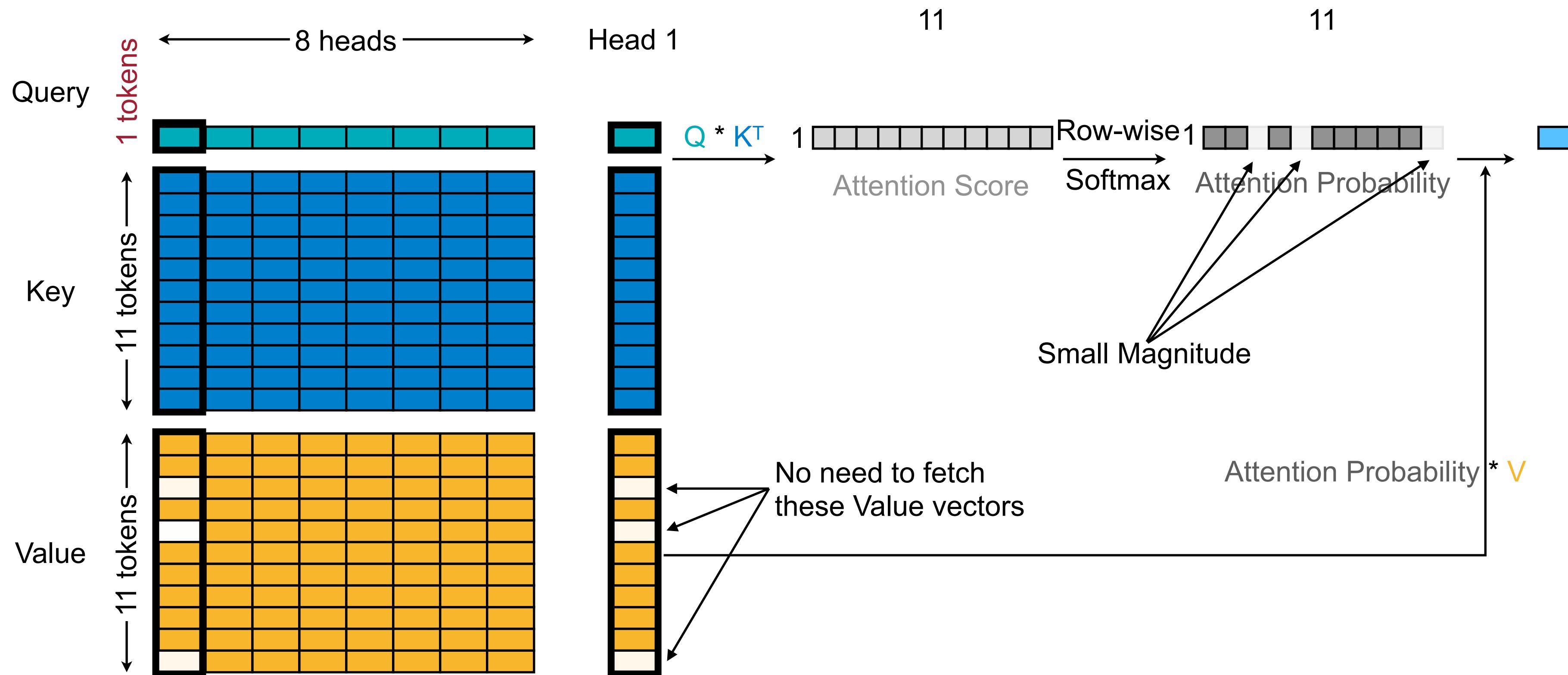
Find Unimportant Heads with Attention Outputs

- If one head output is **small**: the head is **unimportant** to latter layers
- Maintain an **importance score** for each head
- **Accumulate** attention output magnitude to the importance scores
- **Top-k** scores indicate top-k important heads
- Importance scores can be accumulated **across** layers



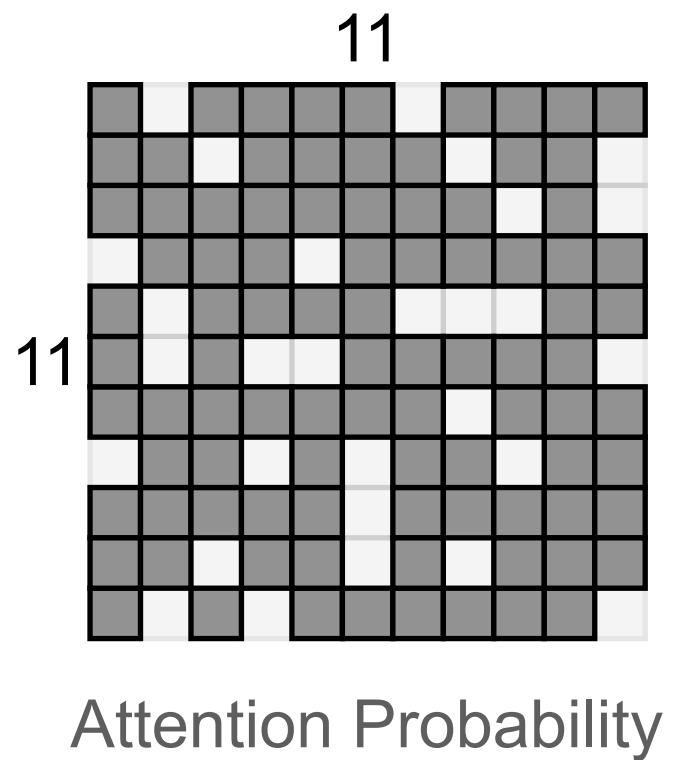
Local Value Pruning

- Cascade token/head pruning find unimportant token/head in **current** layer, prune in **next** layer
- Local value pruning find unimportant Value in **current** layer, prune in **current** layer
- **Directly** rank top-k attention probabilities and only fetch their Value vectors



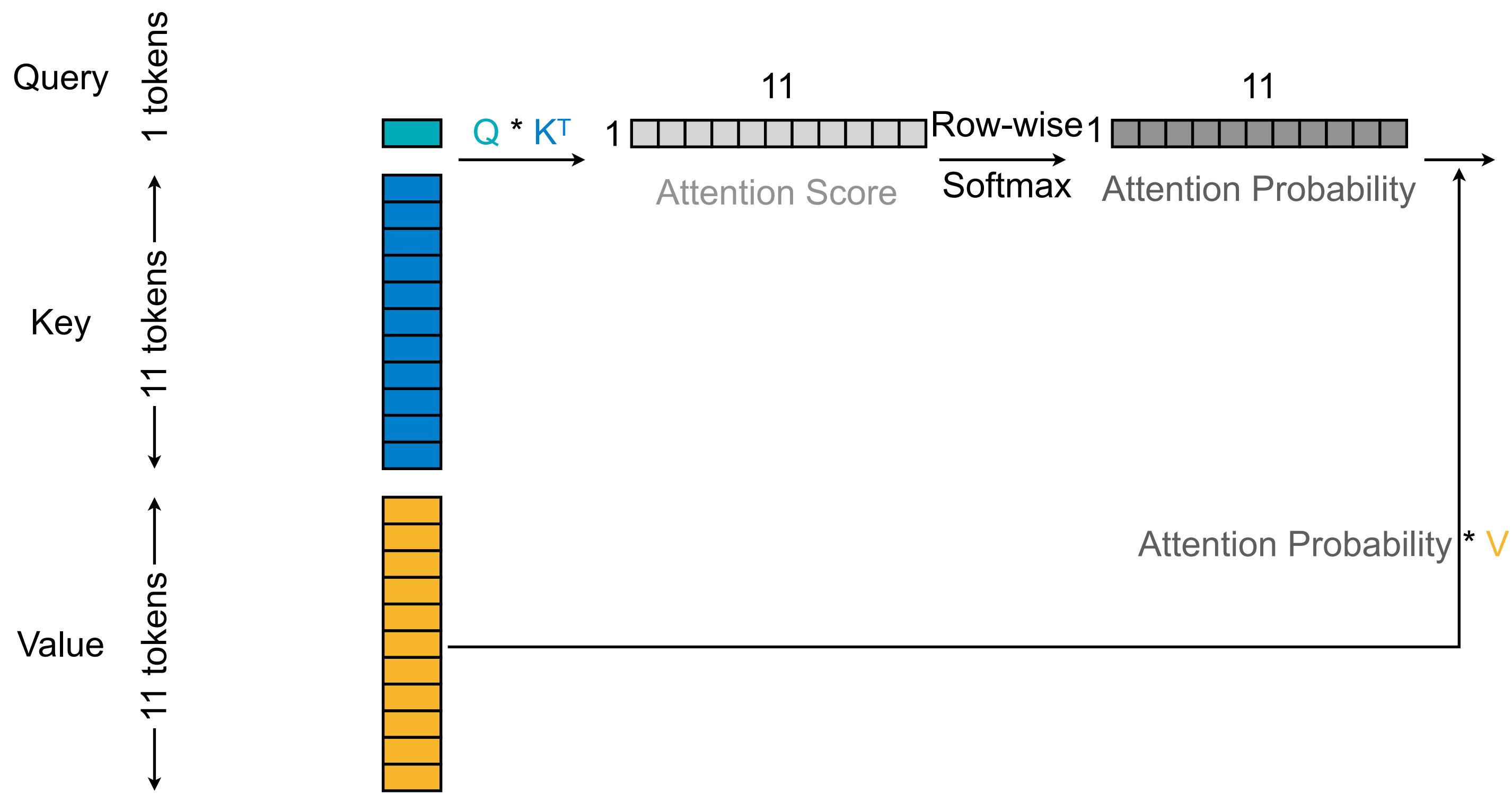
Local Value Pruning

- Cascade token/head pruning find unimportant token/head in **current** layer, prune in **next** layer
- Local value pruning find unimportant Value in **current** layer, prune in **current** layer
- **Directly** rank top-k attention probabilities and only fetch their Value vectors
- Only for **generation** stage
 - Because in summarization, small attention probs have different locations in different rows



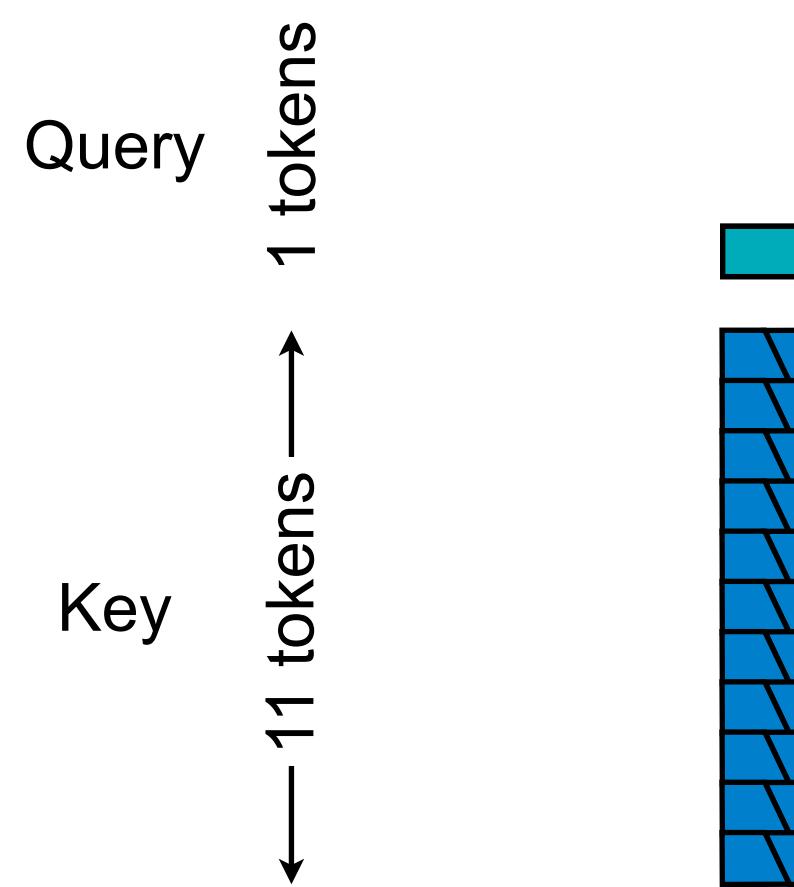
Progressive Quantization

- Generation stage is **memory-bounded** because of matrix vector multiplication
 - Need to fetch Key and Value from DRAM
 - **Static** quantization: quantizes Key, Value to reduce DRAM access



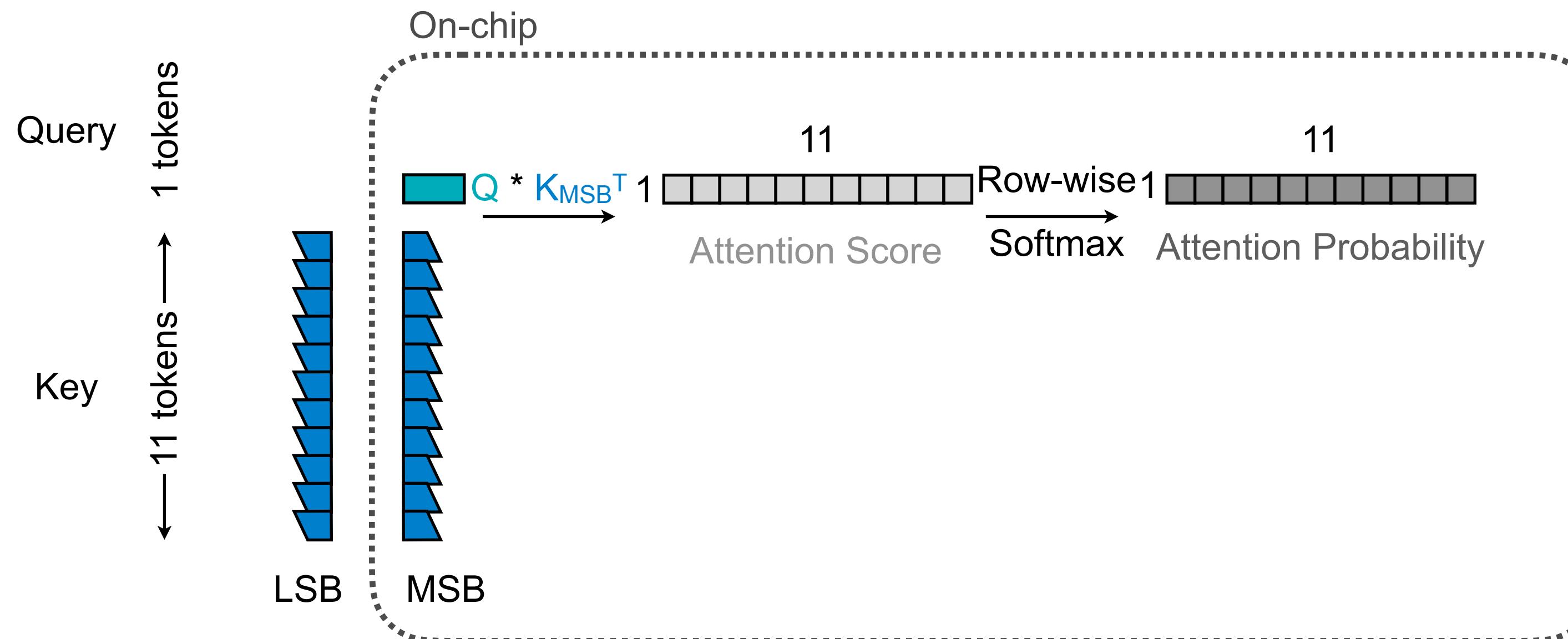
Progressive Quantization

- Separate Key to LSB part and MSB part



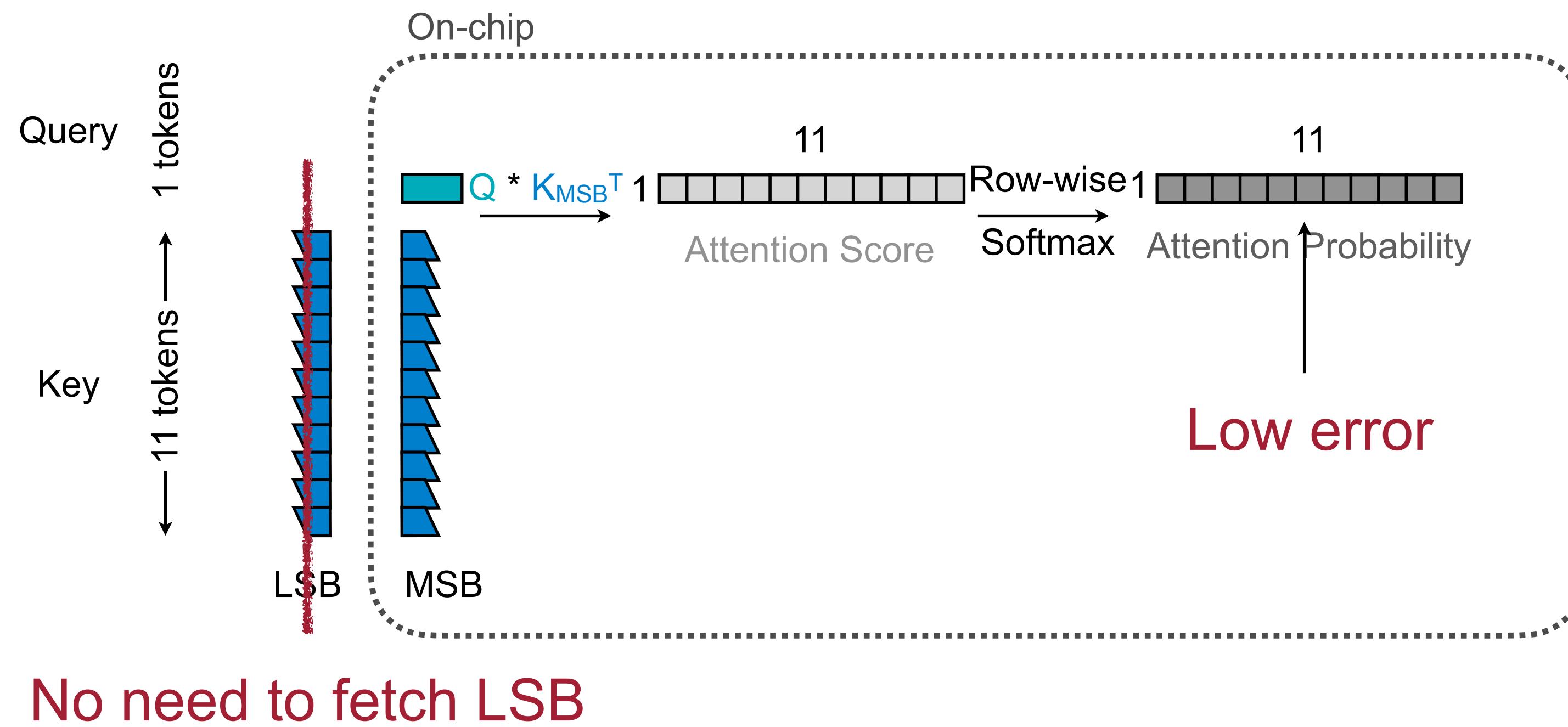
Progressive Quantization

- Separate Key to LSB part and MSB part
- Only fetch MSB from DRAM to on-chip, and compute attention probability



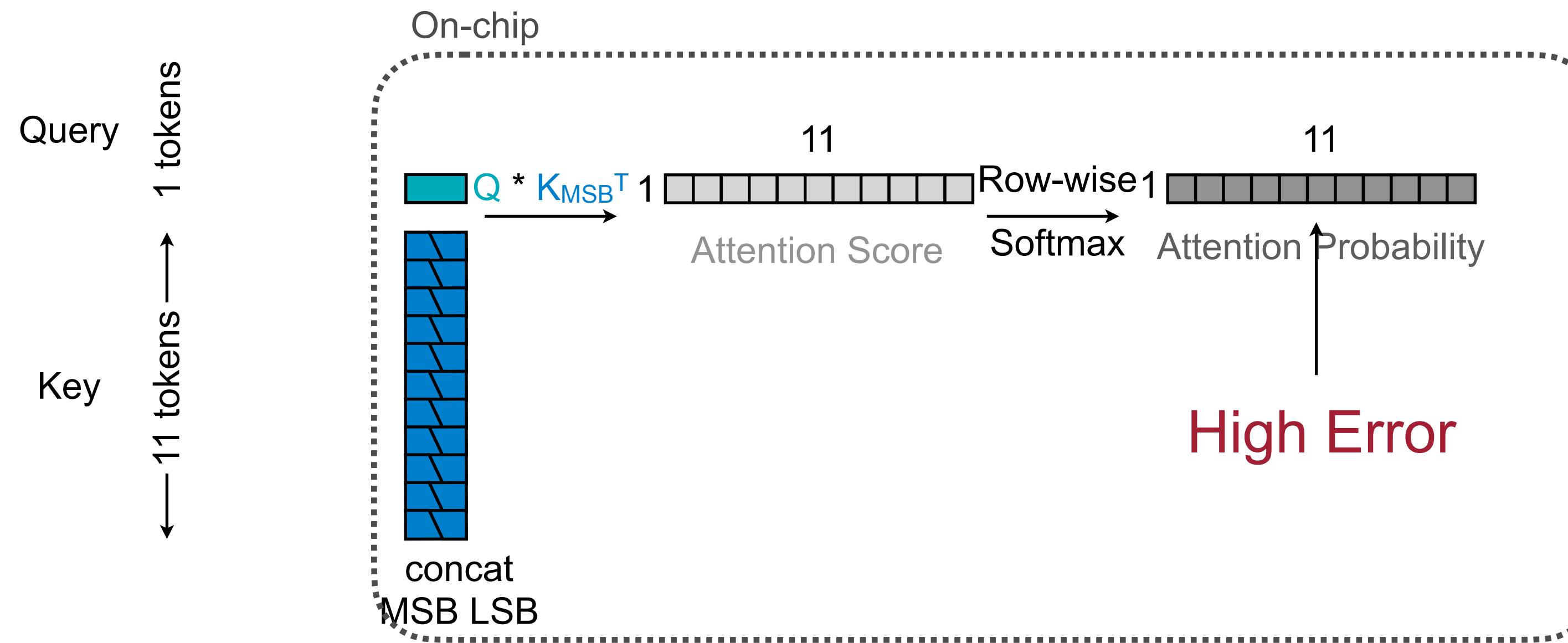
Progressive Quantization

- Separate Key to LSB part and MSB part
- Only fetch **MSB** from DRAM to on-chip, and compute attention probability
- If attention probability has **low error**:



Progressive Quantization

- Separate Key to LSB part and MSB part
- Only fetch **MSB** from DRAM to on-chip, and compute attention probability
- If attention probability has **high error**:

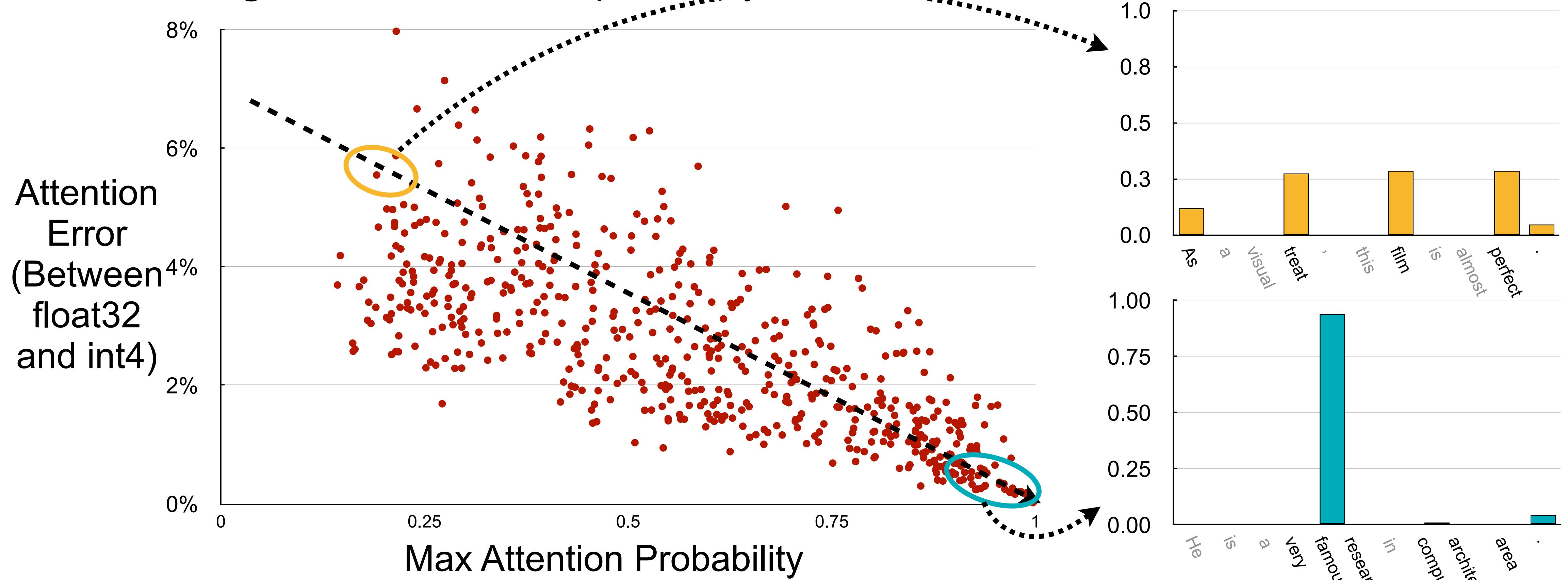


Need to fetch LSB and
recompute attention probability

- **Eagerly** fetch MSB, **lazily** fetch LSB: reduce the average bitwidth

Determine Attention Error with Attention Probability

- How to check whether error is high?
- The **larger** the max attention probability, the **smaller** the attention error



- Therefore, check the **max attention prob**, if large, not need fetch LSB.

Outline

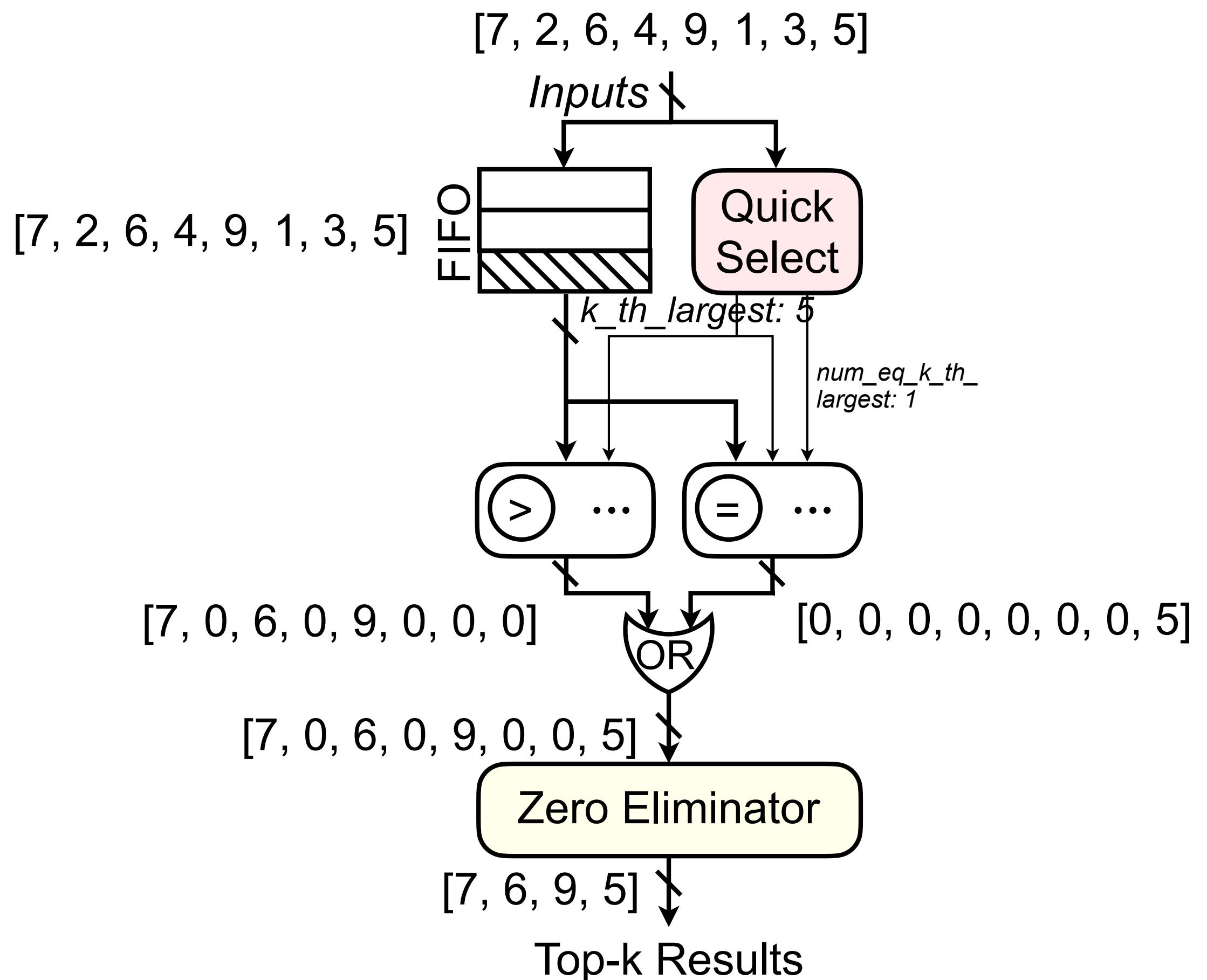
- Quick Overview
- Background
- Algorithmic Optimizations
- **Hardware Architecture**
- Evaluation
- Conclusion

Top-k Engine

- Top-k engine supports cascade token/head pruning and local value pruning
- Example: find top 4 elements from [7, 2, 6, 4, 9, 1, 3, 5]
 - Find the 4th largest: 5
 - Filter the input, **preserve** those \geq 4th largest: [7, 6, 9, 5]

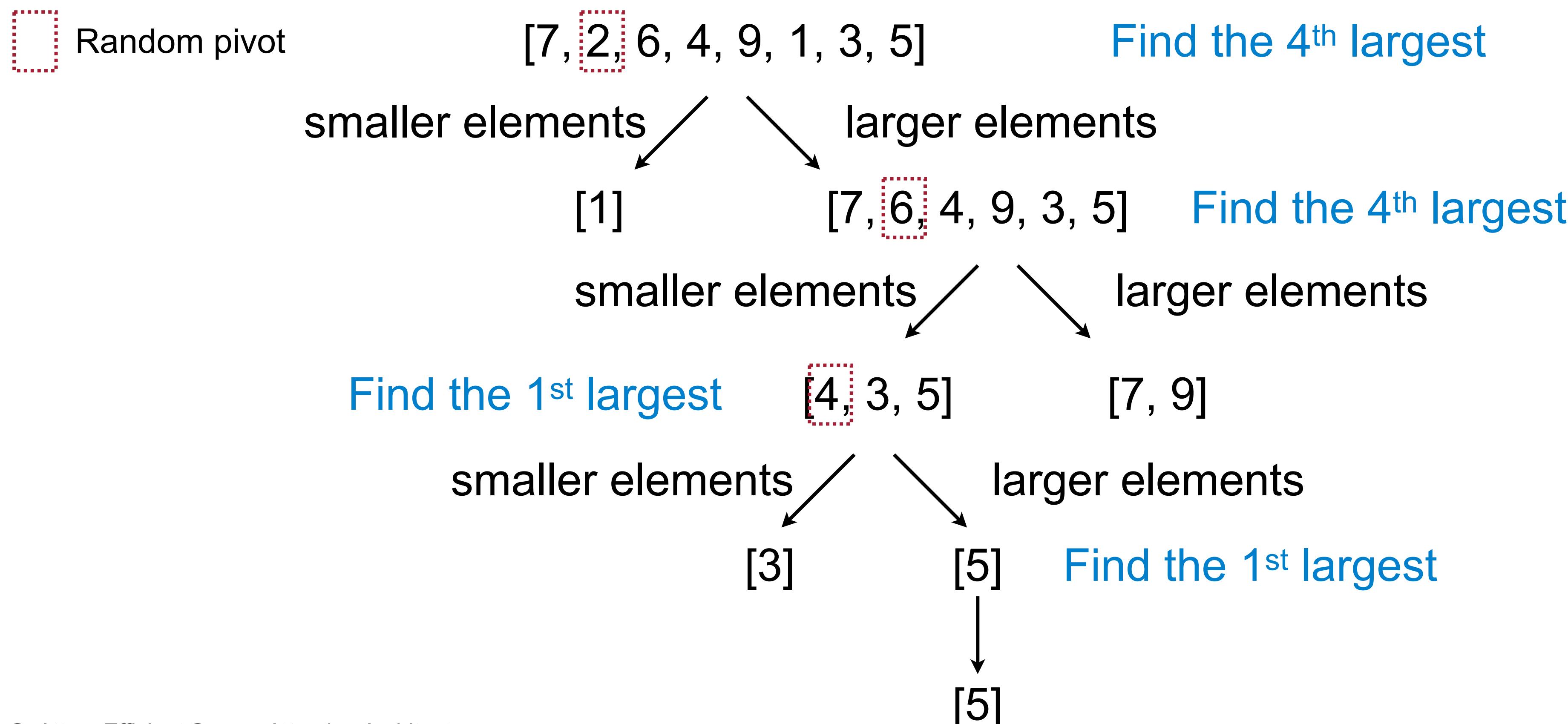
Top-k Engine

- Top-k engine supports cascade token/head pruning and local value pruning
- Example: find top 4 elements from [7, 2, 6, 4, 9, 1, 3, 5]



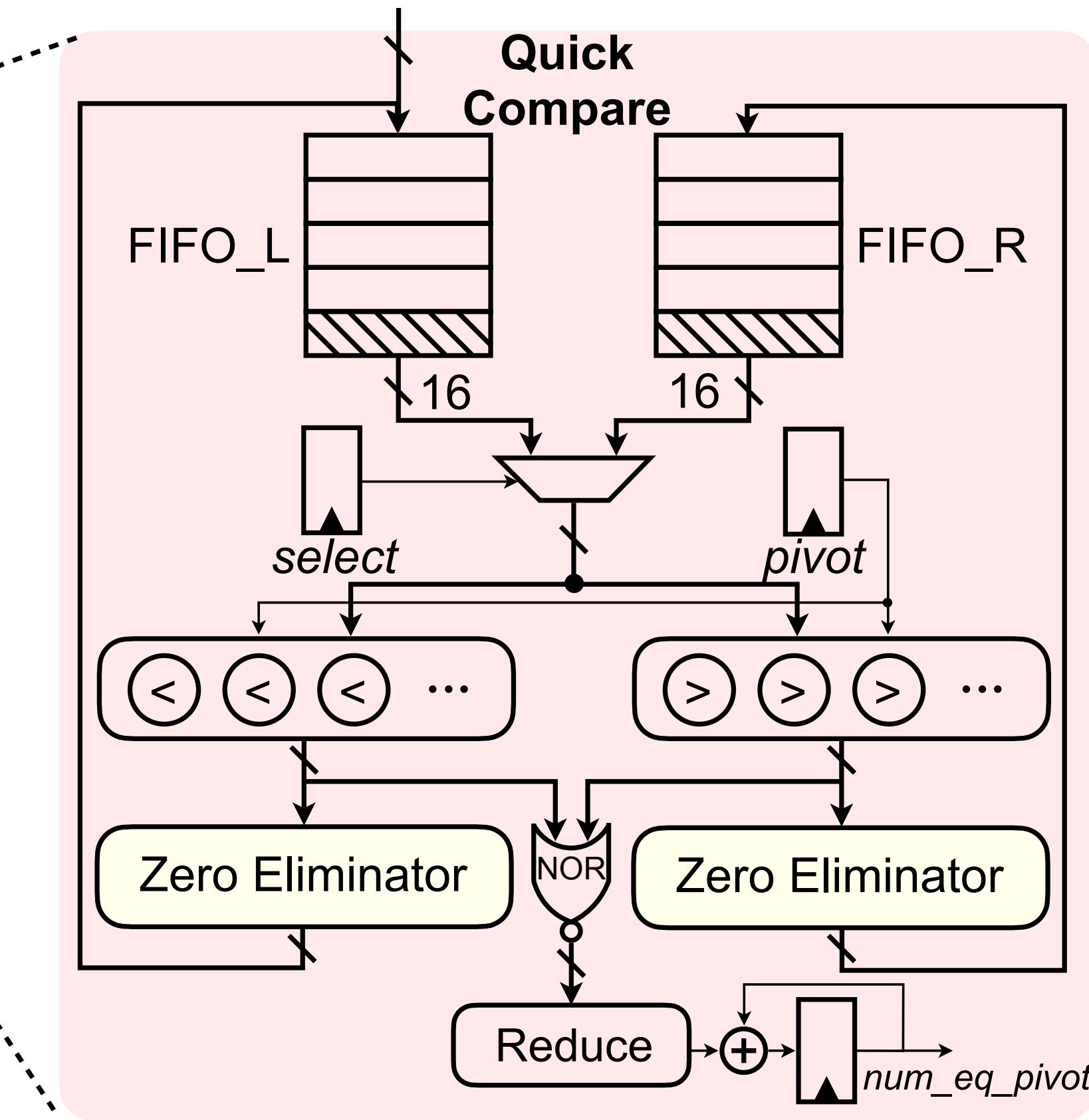
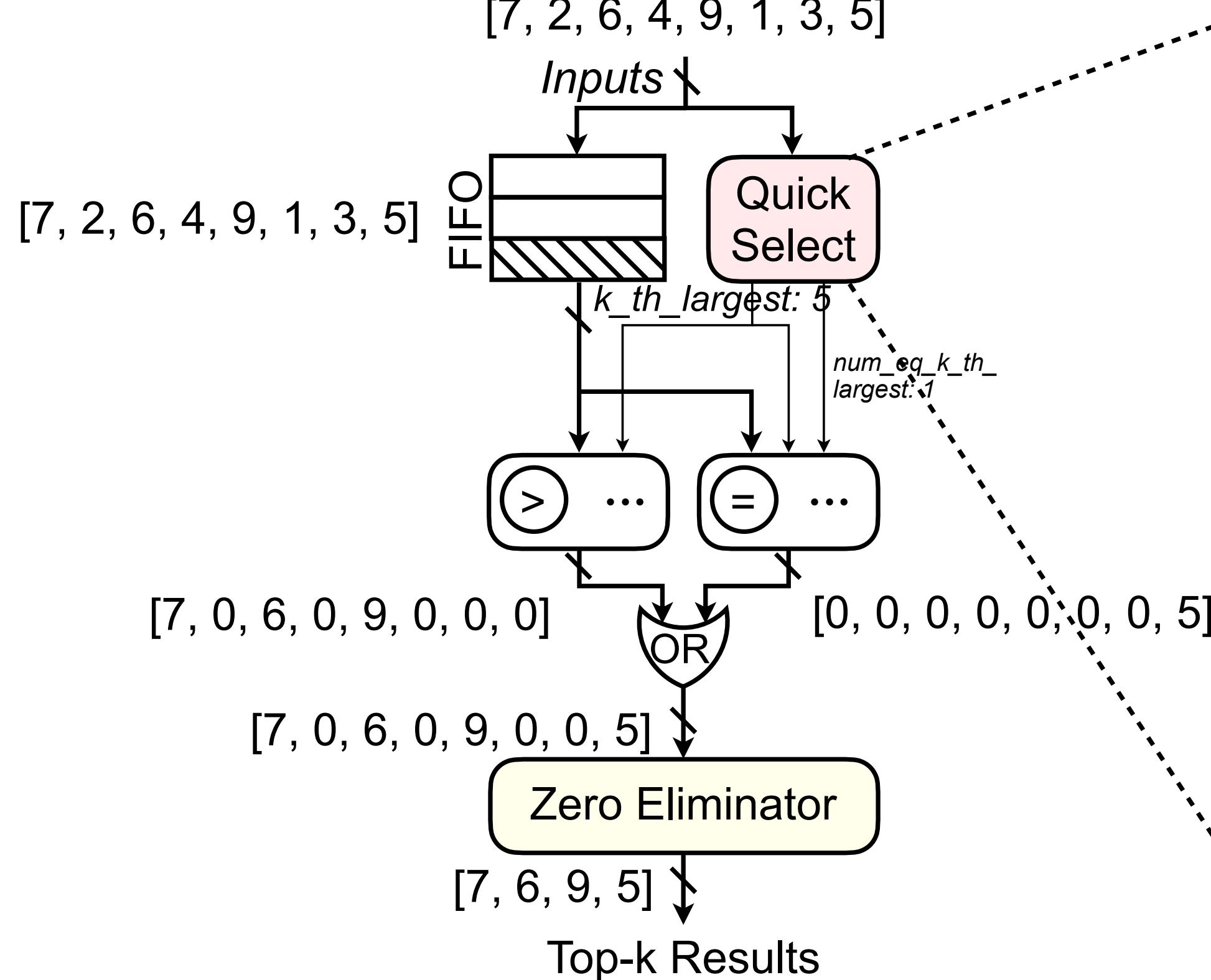
Quick Select

- Quick Select module selects the k^{th} largest element
- Example: selects the 4th largest from [7, 2, 6, 4, 9, 1, 3, 5]



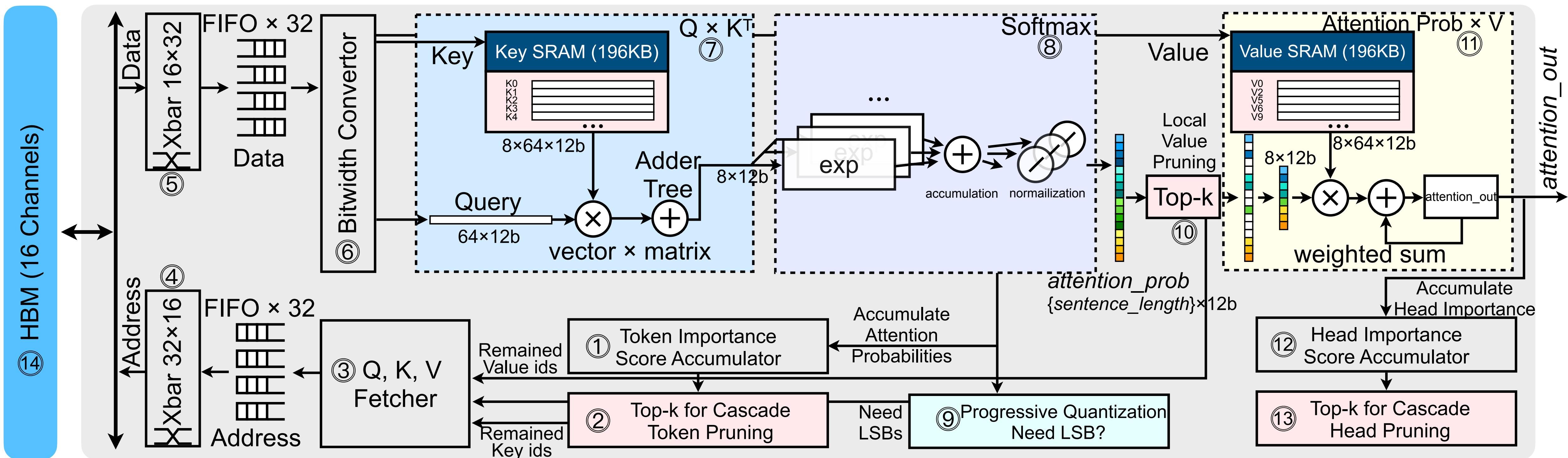
Quick Select

- Top-k Engine has **high-parallelism**
 - 16 '<' comparators and 16 '>' comparators in Quick Select
 - Compare the elements with pivot **in parallel**



Dedicated Accelerator

- Pipelined architecture to improve the throughput



Outline

- Quick Overview
- Background
- Algorithmic Optimizations
- Hardware Architecture
- **Evaluation**
- Conclusion

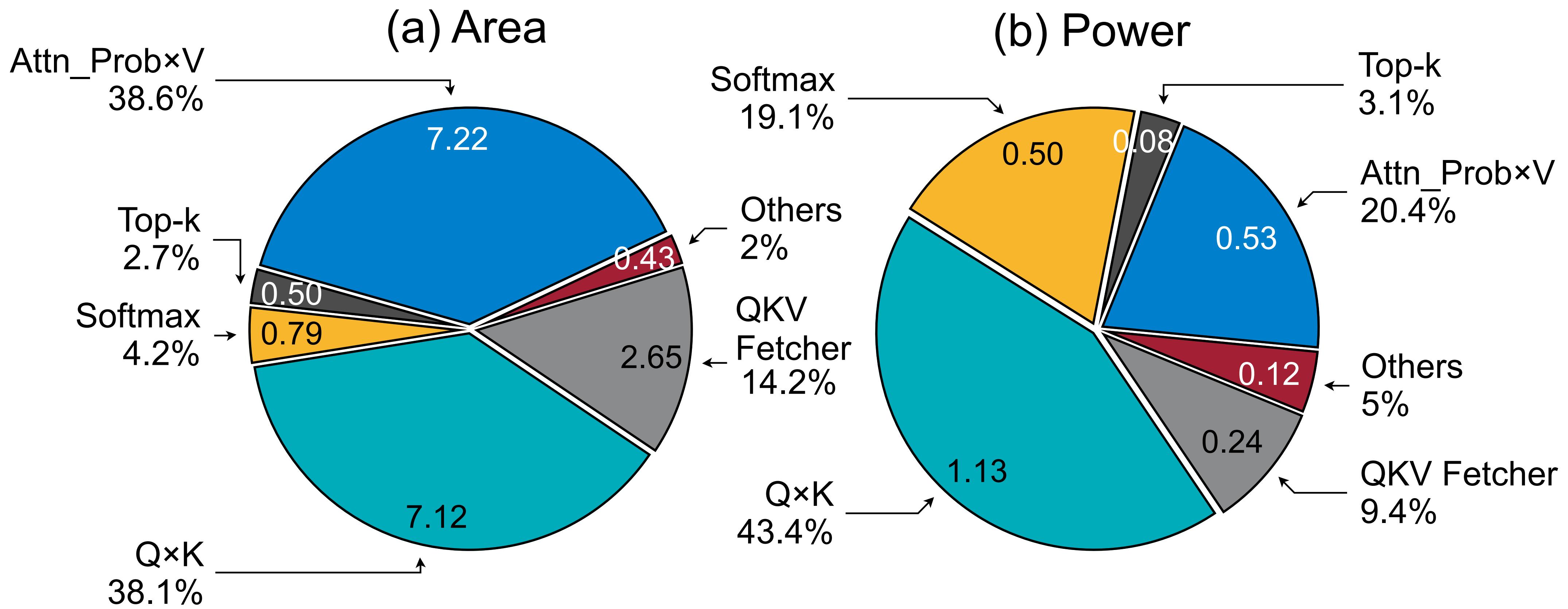
Evaluation

- Hardware Implementation

Technology	TSMC 40nm
Area (w/o DRAM)	18.71mm ²
Power (w/ DRAM)	8.30W
Multipliers	1024
SRAM	392KB
DRAM	HBM2 16 Channels, each @ 32GB/s
Performance on Summarization Stage	1.61TFLOPS
Performance on Generation Stage	0.43TFLOPS

Power & Area Breakdown

- On-chip power and area breakdown



30 Benchmarks

- Model Architecture
 - BERT-Base, BERT-Large, GPT-2-Small, GPT-2-Medium
- Task:
 - Discriminative: GLUE (9 tasks), SQuAD-V1, SQuAD-V2
 - Generative: Language modeling on Wikitext-103, Wikitext-2, Penn Tree Bank, Google One Billion Words

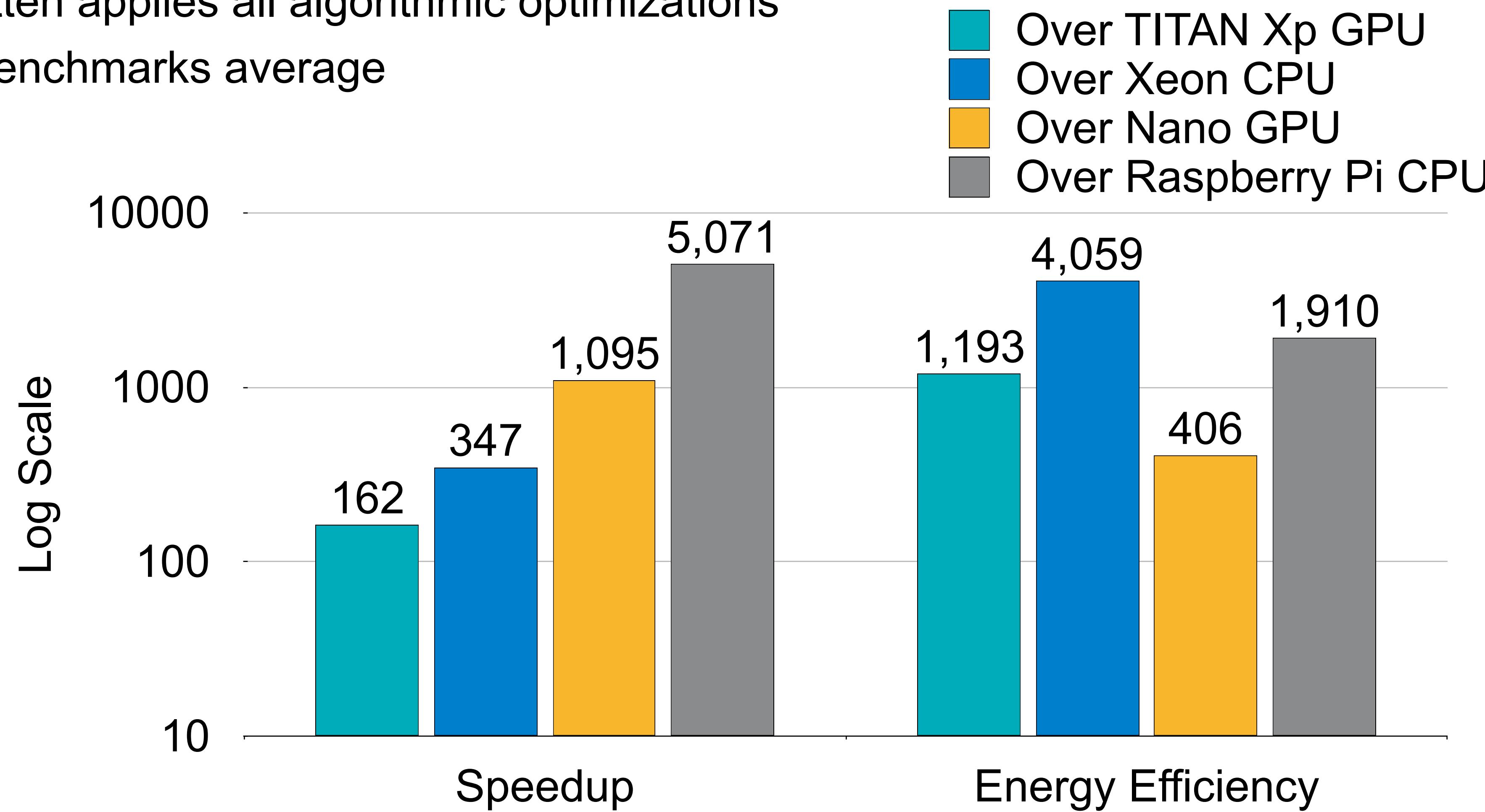
Pruning and Quantization Results

- Pruning ratio is **highly dependent** on sentence length
 - For long sentence (1000 in LM with GPT-2), can prune more than **70%** tokens
 - For short sentence (12 in CoLA with BERT), can prune less than **15%** tokens
- Under same performance (<2% loss on several BERT models), 30 benchmarks average:

Cascade Token Pruning	Prune away 35% (ranging 10%~75%)
Cascade Head Pruning	Prune away 10%
Progressive Quantization Effective Bitwidth	7.8 bits
Local Value Pruning (on top of cascade token pruning)	Prune away 63% (Value)

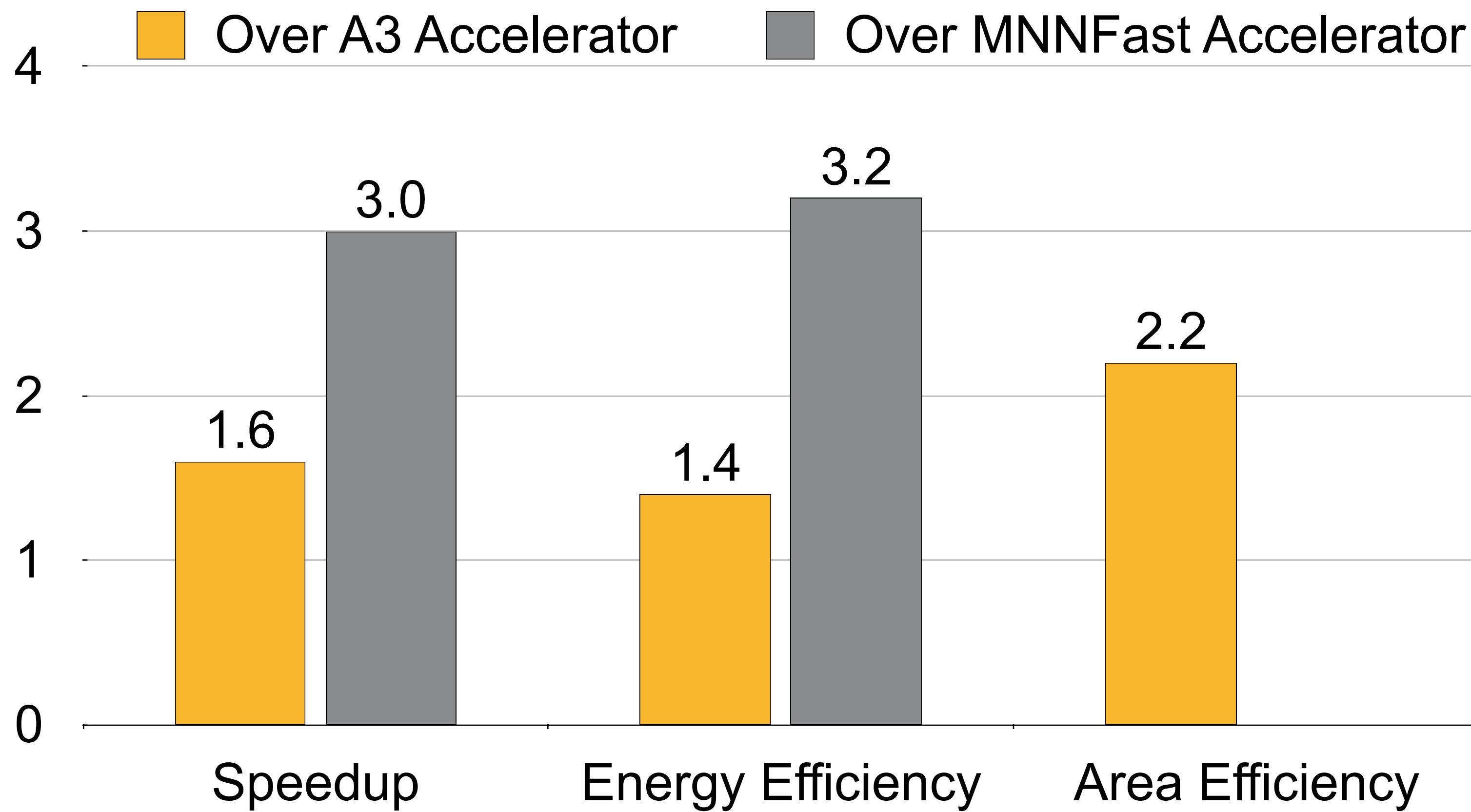
Performance Comparisons

- Over general-purpose CPUs/GPUs on attention layers
 - SpAtten applies all algorithmic optimizations
 - 30 benchmarks average



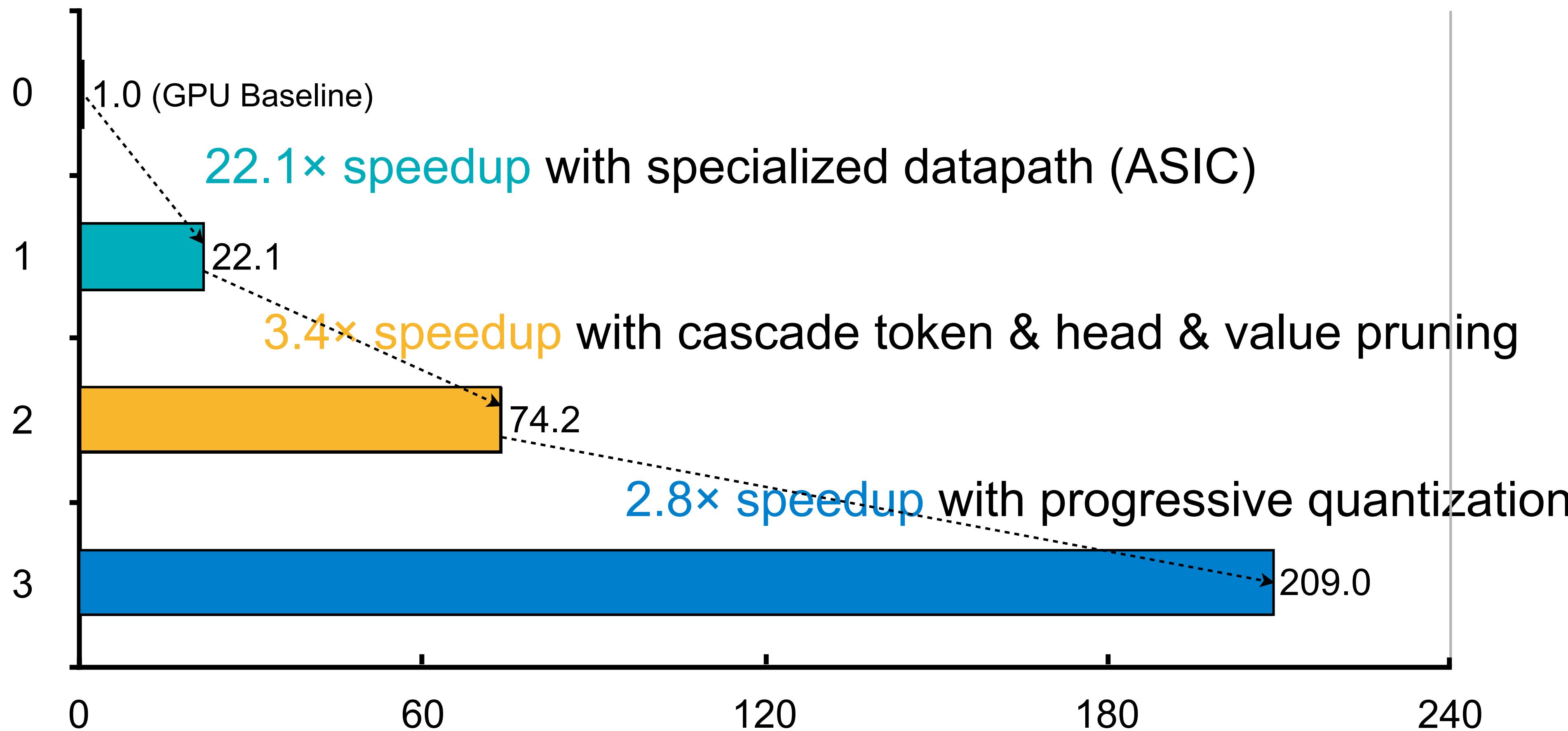
Performance Comparisons

- Outperform state-of-the-art attention accelerators
 - A3 (ASIC) supports local key/value pruning
 - MNNFast (FPGA) supports local value pruning



Speedup Breakdown

- Speedup breakdown on GPT-2 Models over TITAN Xp GPU



More Examples

- BERT sentence classification: (Film sentiment classification result: *positive*)

A wonderful movie, I am sure that you will remember it, you admire its conception and are able to resolve some of the confusions you had while watching it.

More Examples

- BERT sentence classification: (Film sentiment classification result: *positive*)

A wonderful movie, I am sure that you will remember it, you admire its conception and are able to resolve some of the confusions you had while watching it.

sure remember admire resolve confusions

More Examples

- GPT-2 for language modeling: ('English' is the generated token.)

Du ~~Fu was a great poet of the~~ Tang dynasty. Recently ~~a variety of~~ styles ~~have been used~~ in efforts to translate the work of Du Fu into English

More Examples

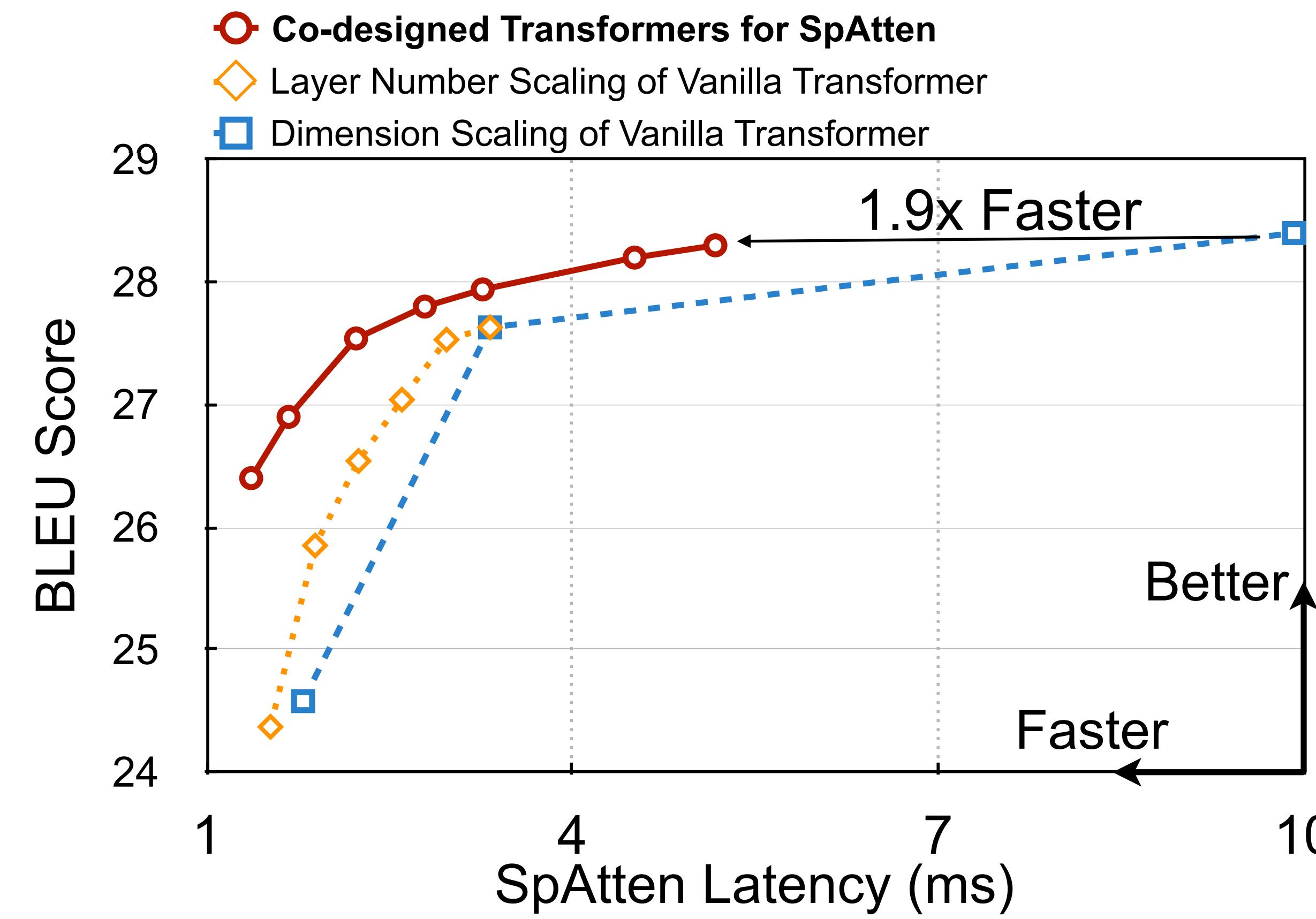
- GPT-2 for language modeling: ('English' is the generated token.)

Du ~~Fu was a great poet of the Tang dynasty. Recently a variety of styles have been used in efforts to translate the work of Du Fu into English~~

Du translate into → English

Specialized Model for SpAtten with HAT: Hardware-Aware Transformer NAS

- HAT: Hardware-Aware Transformer NAS is a framework to search for most suitable model for a target hardware

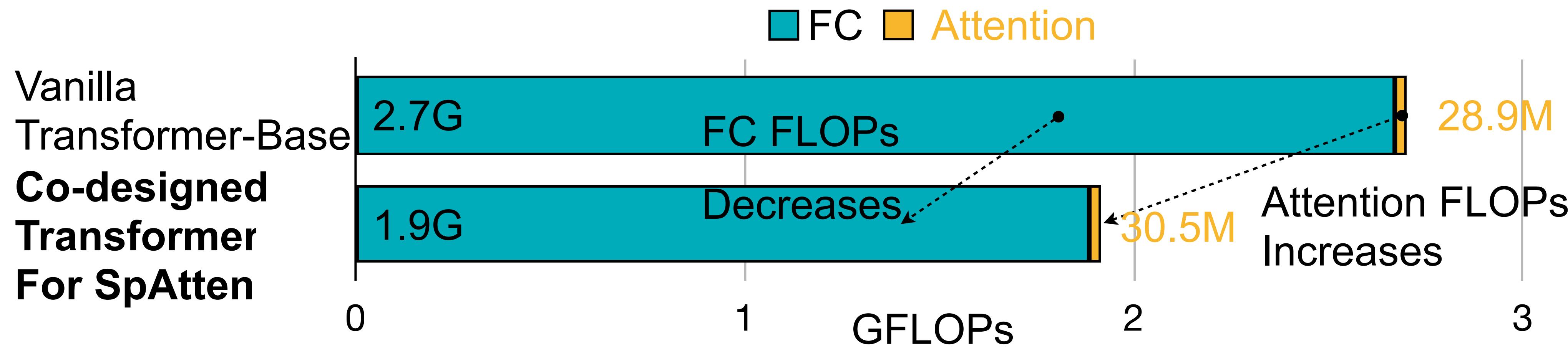


- The co-designed specialized model for SpAtten can be **1.9x faster** than vanilla model

*Hanru Wang, et al. "HAT: Hardware-aware transformers for efficient natural language processing." ACL, 2020.

Specialized Model for SpAtten with HAT: Hardware-Aware Transformer NAS

- HAT: Hardware-Aware Transformer NAS is a framework to search for most suitable model for a target hardware
- Computation breakdown



- SpAtten is good at processing attention layers

*Hanru Wang, et al. "HAT: Hardware-aware transformers for efficient natural language processing." ACL, 2020.

Outline

- Quick Overview
- Background
- Algorithmic Optimizations
- Hardware Architecture
- Evaluation
- Conclusion



spatten.mit.edu
hanrui@mit.edu

SpAtten: Sparse Attention Architecture

Pushing the frontier of **Green AI** and **Tiny AI**

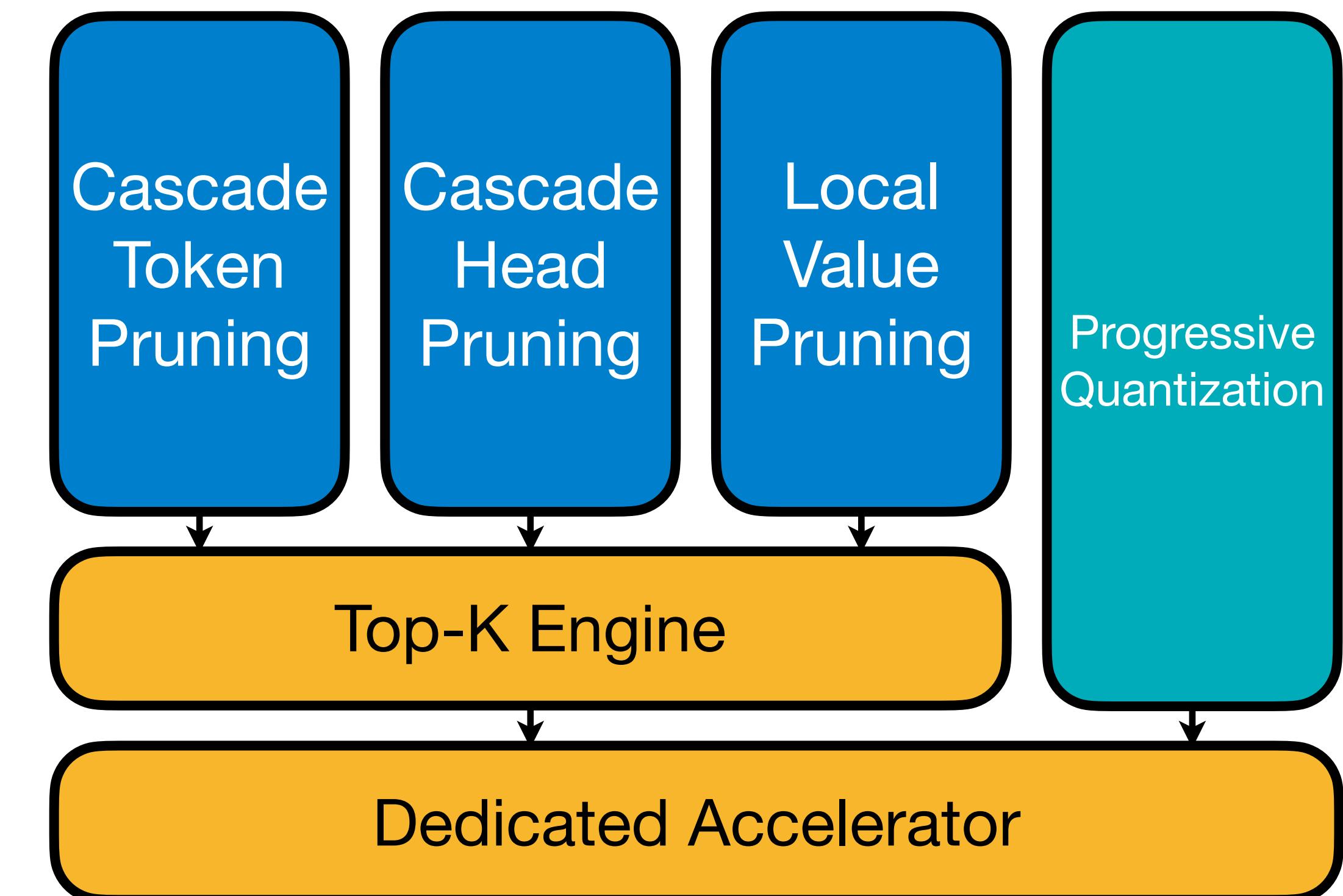


- SpAtten accelerates NLP by removing **human language redundancy**
 - Cascade token/head pruning
 - Local value pruning
 - Progressive quantization
- Hardware accelerator
 - High-Parallelism Top-k engine
 - Specialized data path and operators

HPCA 2021 Live Q&A:

Session 1B

Mon. March 1, 10:40 EST



SpAtten: Efficient Sparse Attention Architecture

Thank you!

This presentation and recording belong to the authors. No distribution is allowed without the authors' permission

HANIA 56