基於RISC-V 向量指令集之 Softmax 運算加速

Softmax Acceleration on RISC-V Vector Extension

學生: 黃士昕

指導老師: 賴槿峰 博士

Student: Shih-Hsin Huang

Advisor: Dr. Chin-Feng Lai

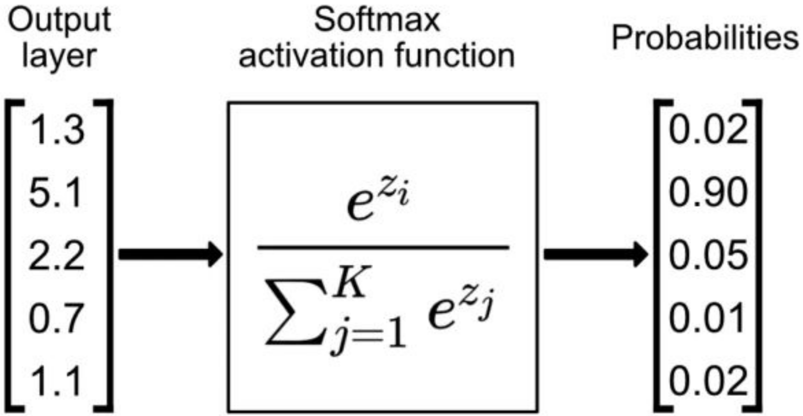
Outline

- Introduction
- Related Work
- Method
- Experiment and Evaluation
- Conclusion and Future Work

Introduction

- Research Motivation
- Research Objectives

Research Motivation - Softmax



CNN

DNN

RNN

TRANSFORMER

Challenges:

- High Computation Cost
- Numerical Instability

Research Motivation - Edge AI

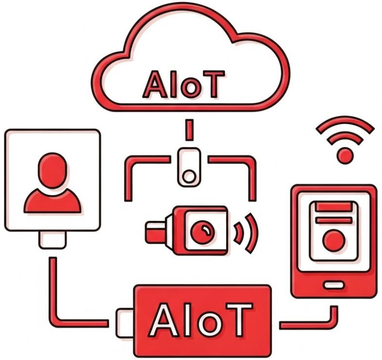
Algorithm 3 Safe softmax

$m_0 \leftarrow -\infty$
for $k \leftarrow 1, V$ **do**
  $m_k \leftarrow \max(m_{k-1}, x_k)$ ▷ Pass 1: read X
end for
$d_0 \leftarrow 0$
for $j \leftarrow 1, V$ **do**
  $y_i = EXP(X_j - m_v)$ ▷ Pass 2: read X, Write Y
  $d_j \leftarrow d_j + y_i$
end for
for $i \leftarrow 1, V$ **do**
  $y_i \leftarrow \frac{y_i}{d_V}$ ▷ Pass 3: read Y, Write Y
end for

Pass 1

Pass 2

Pass 3



Challenges:

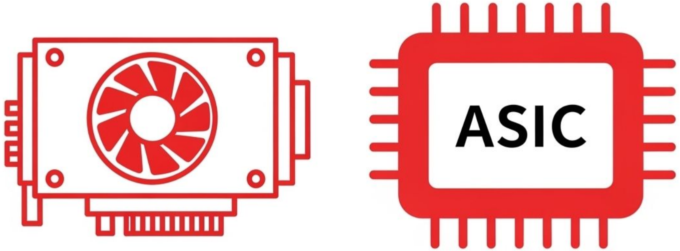
- Memory Bandwidth Bottleck

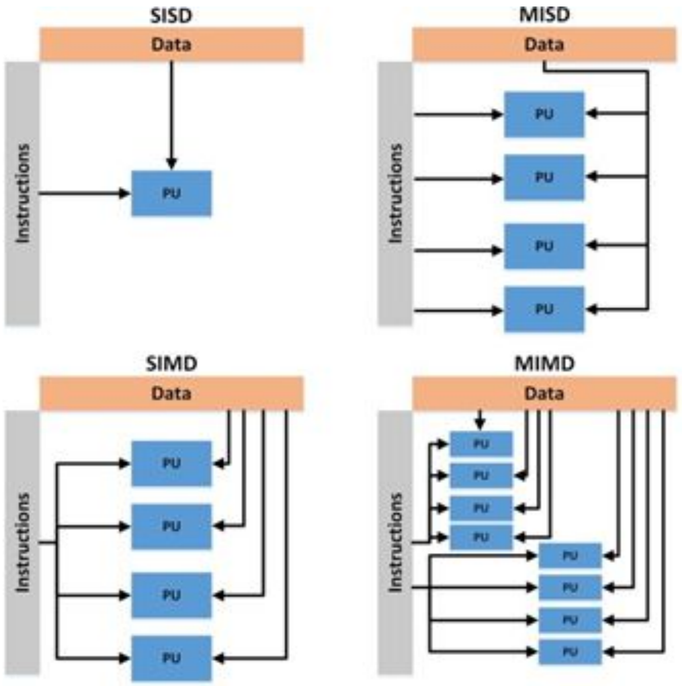
- Limited Bandwidth
- Power Sensitive

Research Motivation - RISC-V

Open Standard

Proprietary / Fragmented





RVV: Write Once, Run Anywhere

Research Objectives

- **Algorithm Design and Innovation**: To propose a Conditional Online Softmax algorithm
- **High-Performance Implementation on RISC-V Vector Extension**:
  - Softmax Function
  - Exponential Function
- **Comprehensive Performance Evaluation and Bottleneck Analysis**

NCKU

Related Work

- Softmax and Numerical Stability
- A Review of Optimization Strategies and Their Limits
  - Online Softmax
  - Exponential Function
- Programming with the RISC-V Vector Extension

Background: Safe Softmax

Algorithm 3 Safe softmax

$m_0 \leftarrow -\infty$
for $k \leftarrow 1, V$ **do**
  $m_k \leftarrow \max(m_{k-1}, x_k)$ ▷ Pass 1: read X
end for
$d_0 \leftarrow 0$
for $j \leftarrow 1, V$ **do**
  $y_i = \boxed{EXP(X_j - m_v)}$ ▷ Pass 2: read X, Write Y
  $d_j \leftarrow d_j + y_i$
end for
for $i \leftarrow 1, V$ **do**
  $y_i \leftarrow \frac{y_i}{d_V}$ ▷ Pass 3: read Y, Write Y
end for

3 pass

$$y_i = \frac{\boxed{e^{x_i - \boxed{\max_{k=1}^{V} x_k}}}}{\sum_{j=1}^{V} \boxed{e^{x_j - \boxed{\max_{k=1}^{V} x_k}}}}$$

Exponent	fraction = 0	fraction ≠ 0	Equation
$00_H = 00000000_2$	±zero	subnormal number	$(-1)^{\text{sign}} \times 2^{-126} \times 0.\text{fraction}$
01_H, ..., $FE_H = 00000001_2$, ..., 11111110_2	normal value		$(-1)^{\text{sign}} \times 2^{\text{exponent}-127} \times 1.\text{fraction}$
$FF_H = 11111111_2$	$\boxed{\pm\text{infinity}}$	NaN (quiet, signaling)	

NCKU

Background: Online Softmax

Algorithm 4 Safe softmax with online normalizer calculation

$m_0 \leftarrow -\infty$
$d_0 \leftarrow 0$
for $j \leftarrow 1, V$ **do**
  $m_j \leftarrow \max(m_{j-1}, x_j)$ ▷ Pass 1: read X
  $d_j \leftarrow d_{j-1} \times e^{m_{j-1}-m_j} + e^{x_j-m_j}$
end for
for $i \leftarrow 1, V$ **do**
  $y_i \leftarrow \frac{e^{x_i-m_V}}{d_V}$ ▷ Pass 2: read X, Write Y
end for

2 pass

Data Dependency

3N expf calls

This can be proven by induction.

Background: Exponential Function

Step 1: Range Reduction

$$e^x = 2^{x \cdot \log_2(e)} \qquad t = x \cdot log_2(e)$$

$$t = i + f, i = \lfloor t \rfloor, f = t - i$$

Step 2: Divide and Conquer

$$e^x = \boxed{2^i} \cdot \boxed{2^f}$$

Bit Manipulation

Polynomial Approximation

Look up table

glibc expf()

Step 3: Reconstruction

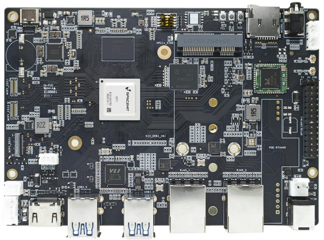
NCKU

Background: RISC-V Vector Extension

RISC-V "V" Vector Extension

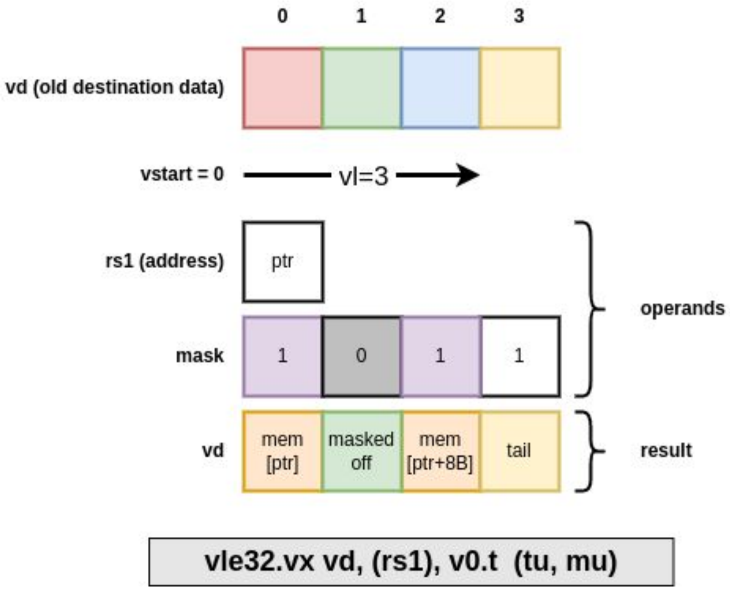
Version 1.0

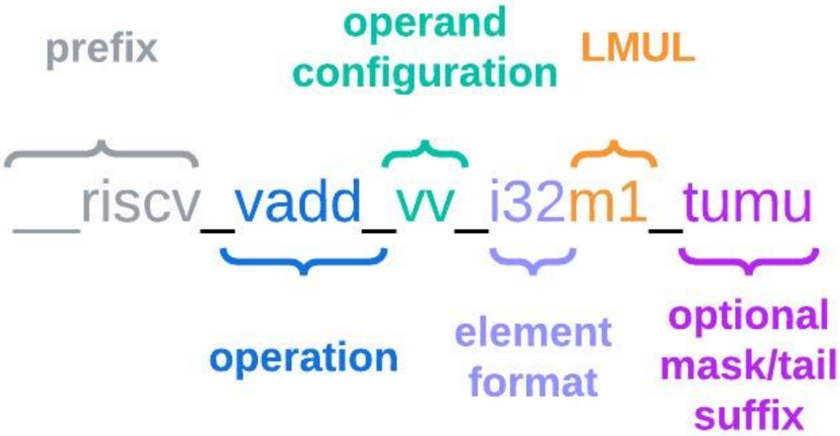
vlen = 256b



vlen = 512b

Background: RISC-V Vector Extension





Background: Research Gap and Our Contributions

The Research Gap		**Our Contributions**	
Algorithmic Dilemma	Safe Softmax: ● High Memory Access Cost(3-Pass)	**Conditional Online Softmax**	● Balance the trade-off between memory access and computational cost
	Online Softmax: ● High Computation Cost(3N expf()), ● Data Dependency		
Implementation Bottleneck	● Compiler auto-vectorization ineffective	**Manual Vectorization**	● RVV Intrinsics manual Vectorization
	● Slow expf() Implementation		● High-performance vexpf() function

NCKU

Methodology

- Condition Online Softmax
- Branch-free Conditional Online Softmax
- Vectorize Exponential Function

Methodology: Condition Online Softmax

Algorithm 5 Condition Online Softmax

```
m_0 ← −∞
d_0 ← 0
for j ← 1, V do                                   ▷ Pass 1: read X
    if Updated the maximum value then
        d_j ← d_j × [e^{m_{j−1} − x}] + 1
    else
        d_j = d_j + [e^{x − m_{j−1}}]
    end if
end for
for i ← 1, V do                                   ▷ Pass 2: read X, Write Y
    y_i ← [e^{x_i − m_V}] / d_V
end for
```

$m_0 \leftarrow -\infty$; $d_j \leftarrow d_j \times \boxed{e^{m_{j-1}-x}} + 1$; $d_j = d_j + \boxed{e^{x-m_{j-1}}}$; $y_i \leftarrow \frac{\boxed{e^{x_i-m_V}}}{d_V}$

An algorithmic principle: 'Each time a new maximum is found, the accumulated sum must be rescaled.'

2N expf() calls!

Methodology: The Record-Breaker Problem

The Record-Breaker Problem is a classic model in probability theory that examines the expected number of record-breaking events within a sequence of random variables. An event is defined as a "record-breaker" if its observed value is greater than all values that have preceded it in the sequence

$$d_j = \begin{cases} (d_{j-1} \times e^{m_{j-1}-m_j} + 1), & \text{if } x_i > m_{i-1}, \\ (d_{j-1} \times 1 + e^{x_j - m_j}), & \text{otherwise.} \end{cases}$$

向量長度 (n)	$H_n \approx \ln n + \gamma$	更新比例
10	2.88	0.3
100	5.18	0.04
1000	7.47	0.007
10000	9.79	0.0009
100000	12.08	8.0e-5
1000000	14.40	1.3e-5
10000000	16.70	1.6e-5

NCKU

Methodology: Branch-free Conditional Online Softmax

Algorithm 6 Branch-Free Vectorized Conditional Online Softmax

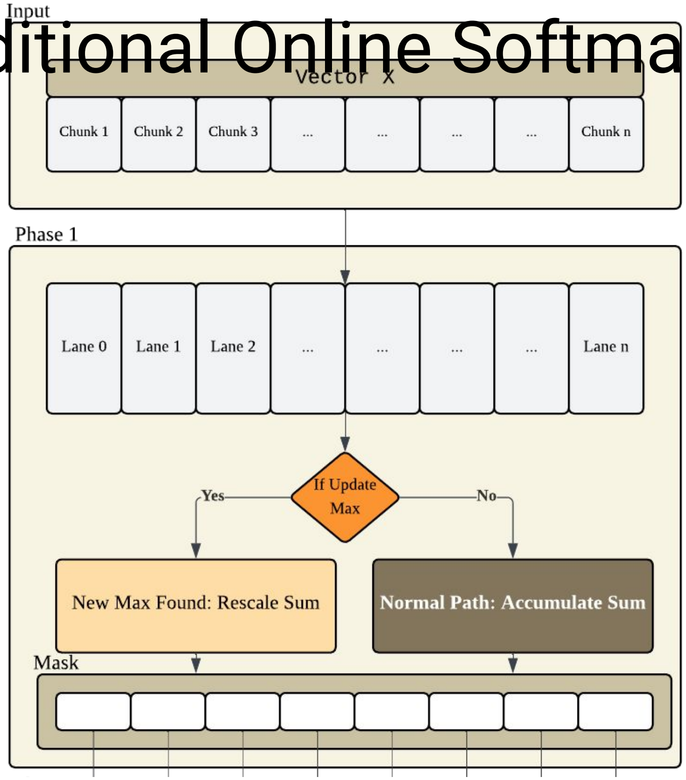
```
                                              ▷ Phase 1: Vectorized Conditional Recurrence
function PHASE1_CONDITIONALRECURRENCE(X, N)
    Input: An input data vector X of size N.
    Output: Vector of per-lane normalizers v_sum

    vl_max ← GETMAXVECTORLENGTH()
    v_max ← VECTORBROADCAST(−∞, vl_max)
                                              ▷ Initialize per-lane max to negative infinity
    v_sum ← VECTORBROADCAST(0.0, vl_max)
                                              ▷ Initialize per-lane sum to zero
    while N > 0 do
        vl ← SETVECTORLENGTH(N)
        v_next ← VECTORLOAD(X, vl)
                              ▷ Step 1: Generate mask to identify lanes with new record-breakers
        v_mask ← v_next > v_max
                              ▷ Step 2: Speculatively compute exponential terms for both paths
        v_diff ← v_next − v_max
        v_exp_pos ← VECTOREXP(v_diff, vl)                              ▷ e^(x_j − m_old)
        v_exp_neg ← VECTORRECIPROCALAPPROX(v_exp_pos, vl)              ▷ e^(m_old − x_j)

                              ▷ Step 3: Compute update results for both paths in parallel
        v_sum_if_true ← VECTORFMA(v_sum, v_exp_neg, 1.0, vl)
                                              ▷ if-path: d · e^(m_old − m_new) + 1
        v_sum_if_false ← v_sum + v_exp_pos
                                              ▷ else-path: d + e^(x − m)

                              ▷ Step 4: Merge results using the mask to complete the branch-free update
        v_sum ← VECTORMERGE(v_sum_if_false, v_sum_if_true, v_mask, vl)
        v_max ← VECTORMAX(v_max, v_next, vl)                          ▷ Update per-lane maximums

        X ← X + vl
        N ← N − vl
    end while
    return (v_max, v_sum)
end function
```



$$d_j = \begin{cases} (d_{j-1} \times e^{m_{j-1} - m_j} + 1), & \text{if } x_i > m_{i-1}, \\ (d_{j-1} \times 1 + e^{x_j - m_j}), & \text{otherwise.} \end{cases}$$

Methodology: Branch-free Conditional Online Softmax

▷ Phase 2: Cross-Lane Reduction

function PHASE2_CROSSLANEREDUCTION($\mathbf{v}_{\mathbf{max}}, \mathbf{v}_{\mathbf{sum}}$, `active_lanes`)

  Input: Per-lane results $\mathbf{v}_{\mathbf{max}}, \mathbf{v}_{\mathbf{sum}}$, and the number of active lanes.

  Output: The final global maximum m_{final} and normalizer d_{final}.

  $m_{final} \leftarrow \mathbf{v}_{\mathbf{max}}[0]$

  $d_{final} \leftarrow \mathbf{v}_{\mathbf{sum}}[0]$

  for $i \leftarrow 1$ **to** `active_lanes` $- 1$ **do**

    $m_{next} \leftarrow \mathbf{v}_{\mathbf{max}}[i]$; $d_{next} \leftarrow \mathbf{v}_{\mathbf{sum}}[i]$

    $m_{new} \leftarrow \max(m_{final}, m_{next})$

    $d_{final} \leftarrow d_{final} \cdot \mathrm{EXP}(m_{final} - m_{new}) + d_{next} \cdot \mathrm{EXP}(m_{next} - m_{new})$

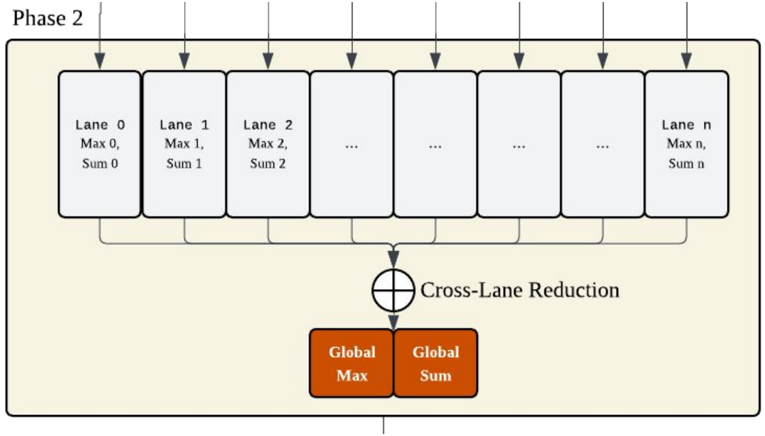
    ▷ Execute the $\oplus$ merge operation

    $m_{final} \leftarrow m_{new}$

  end for

  return (m_{final}, d_{final})

end function



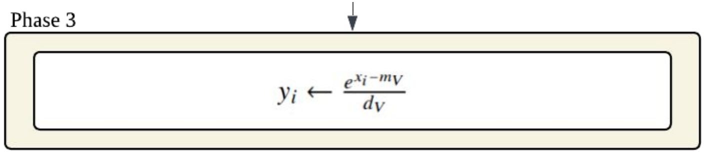
$$\begin{bmatrix} m_i \\ d_i \end{bmatrix} \oplus \begin{bmatrix} m_j \\ d_j \end{bmatrix} = \begin{bmatrix} \max(m_i, m_j) \\ d_i \cdot \exp(m_i - \max(m_i, m_j)) + d_j \cdot \exp(m_j - \max(m_i, m_j)) \end{bmatrix}$$

NCKU

Methodology: Branch-free Conditional Online Softmax

▷ Phase 3: Final Normalization

function PHASE3_FINALNORMALIZATION($X, Y, N, m_{final}, d_{final}$)
  while $N > 0$ **do**
    $\texttt{vl} \leftarrow$ SETVECTORLENGTH(N)
    $\mathbf{v}_{\texttt{data}} \leftarrow$ VECTORLOAD($X, \texttt{vl}$)
    $\mathbf{v}_{\text{data}} \leftarrow \mathbf{v}_{\text{data}} - m_{final}$
    $\mathbf{v}_{\text{data}} \leftarrow$ VECTOREXP($\mathbf{v}_{\text{data}}, \texttt{vl}$)
    $\mathbf{v}_{\text{result}} \leftarrow \mathbf{v}_{\text{data}} / d_{final}$
    VECTORSTORE($Y, \mathbf{v}_{\text{result}}, \texttt{vl}$)
    $X \leftarrow X + \texttt{vl}$; $Y \leftarrow Y + \texttt{vl}$; $N \leftarrow N - \texttt{vl}$
  end while
end function



Methodology: Vectorize Exponential Function

Algorithm 7 Vectorize Exponential Function

function VectorExpFast(**x**,vl)
  Input: An input vector **x** of length vl.
  Output: An approximate result vector for $e^{\mathbf{x}}$.
  Constants:
  $C_{\log 2e} \leftarrow 1.442695041$
  $P_A, P_B, P_C \leftarrow 0.3371..., 0.6576..., 1.0017...$
  ▷ Coefficients for polynomial approximation

▷ Phase 1: Range Reduction
  $\mathbf{t} \leftarrow \mathbf{x} \cdot C_{log2e}$ ▷ Convert base from e to 2
  $\mathbf{i} \leftarrow$ VectorFloor($\mathbf{t}$) ▷ Get integer part $i = \lfloor t \rfloor$
  $\mathbf{f} \leftarrow \mathbf{t} - \mathbf{i}$ ▷ Get fractional part $f = t - i$, where $f \in [0, 1)$

▷ Phase 2: Polynomial Approximation for $2^{\mathbf{f}}$
  ▷ Use 2nd-degree polynomial evaluated with Horner's method: $(A \cdot f + B) \cdot f + C$
  $\mathbf{v_A} \leftarrow$ VectorBroadcast(P_A, vl)
  $\mathbf{v_B} \leftarrow$ VectorBroadcast(P_B, vl)
  $\mathbf{v_C} \leftarrow$ VectorBroadcast(P_C, vl)
  $\mathbf{p_1} \leftarrow$ VectorFMA($\mathbf{f}, \mathbf{v_A}, \mathbf{v_B}$, vl) ▷ Parallel Fused-Multiply-Add: $\mathbf{f} \cdot A + B$
  $\mathbf{exp2_f} \leftarrow$ VectorFMA($\mathbf{p_1}, \mathbf{f}, \mathbf{v_C}$, vl) ▷ Parallel FMA: $(\mathbf{f} \cdot A + B) \cdot \mathbf{f} + C$

▷ Phase 3: Bit-level Reconstruction for $2^{\mathbf{i}} \cdot 2^{\mathbf{f}}$
  ▷ This phase computes the result by adding **i** to the exponent field of $\mathbf{exp2_f}$
  bits_f ← ReinterpretAsInt($\mathbf{exp2_f}$) ▷ Get the integer bit representation of $2^{\mathbf{f}}$
  bits_i ← $\mathbf{i} \ll 23$ ▷ Represents multiplication by $2^{\mathbf{i}}$ for float32
  bits_final ← **bits_f** + **bits_i** ▷ Merge the exponents via integer addition
  result ← ReinterpretAsFloat(**bits_final**) ▷ Convert final bit pattern back to float

  return result

Experiment and Evaluation

- Experimental Setup
- Results and Discussion

Experiment and Evaluation: Hardware Setup

Main Experimental Platform	Performance Reference Platform
RISC-V Platform	x86 Platform
Board: Banana Pi BPI-F3	CPU: Intel Core i7-13700
CPU: Spacemit X60 (8 Cores @ 1.6 GHz)	Cores: 8 P-cores + 8 E-cores (@ up to 5.2 GHz)
ISA: RV64GCV 1.0	ISA: x86-64 (with AVX2)
Vector Spec: VLEN = 256-bit	Vector Spec: AVX2 (256-bit)
Role in Study: - Algorithm Development and Validation - Vectorization Performance Analysis	Role in Study: - Industry-Standard Performance Reference - Cross-Microarchitecture Behavior Comparison

NCKU

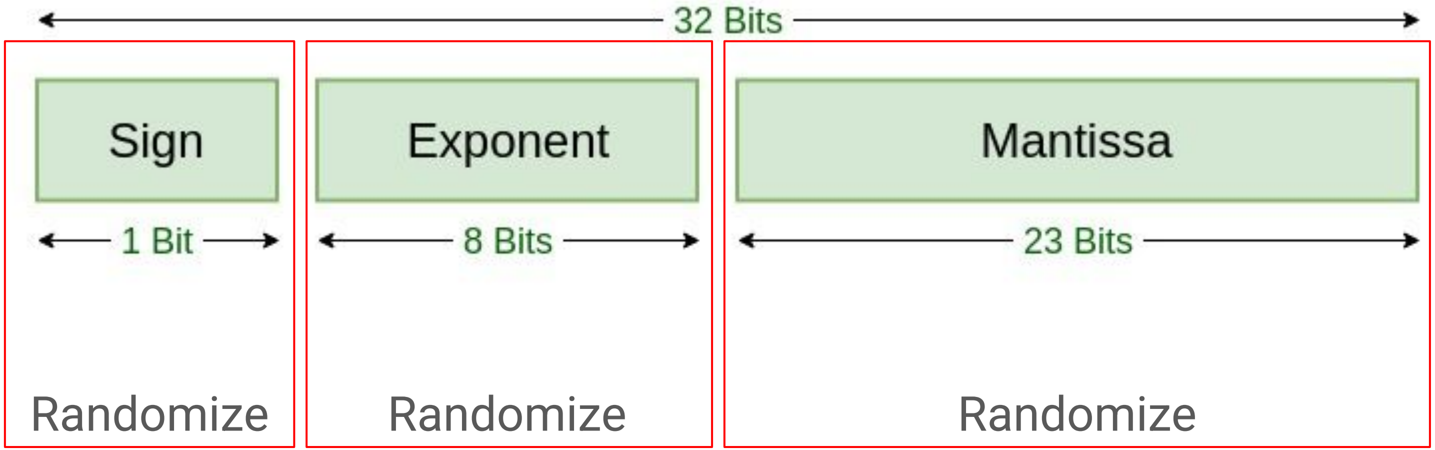
Experiment and Evaluation: Software Setup

System	Compilation	Measurement & Metrics
OS: GNU/Linux (Kernel 6.6.63)	Target ISA: -march=rv64gcv	Primary Metrics: Throughput (Elem/s), Execution Time (ms)
Compiler: RISC-V GCC 13.2.0	Optimization: -O2 (Standard), -O3	Timing Method: clock_gettime()
Execution Control: taskset (Bind to a single core)	Reproducibility: -static	Reliability: Multiple runs for each test (Ensures stable results)

Experiment and Evaluation: Data Generation

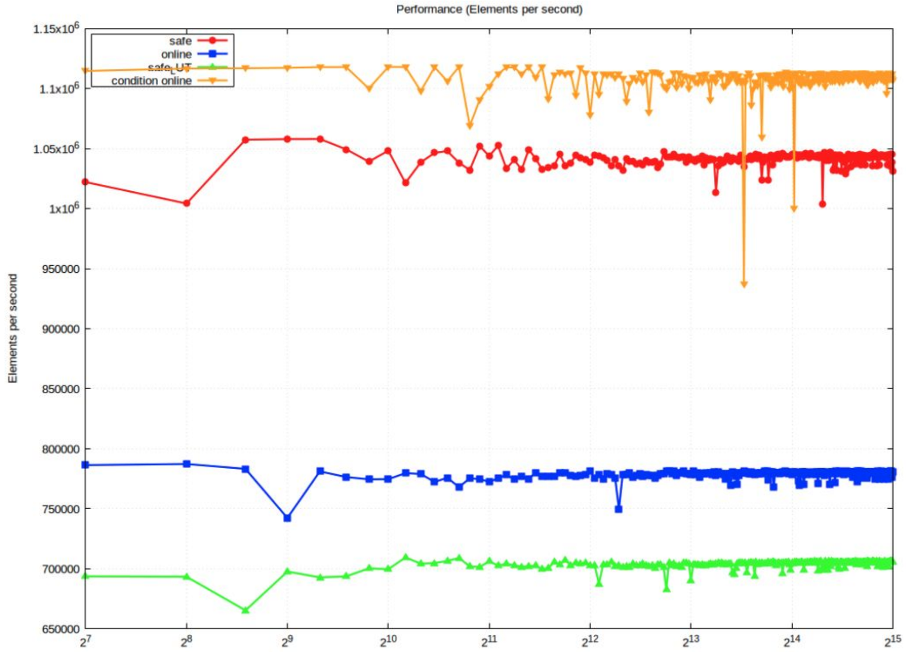
Our strategy generates 32-bit floating-point numbers by independently randomizing their constituent bit-fields, with a specific constraint to avoid special values

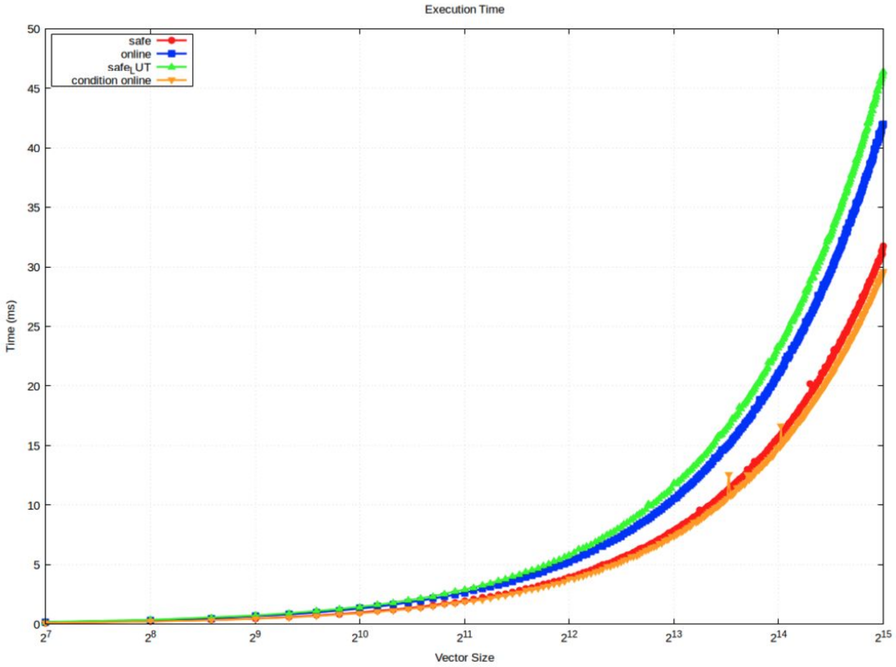
- **No underflow, no overflow, no subnormal number**



Experiment and Evaluation: Conditional Online Softmax

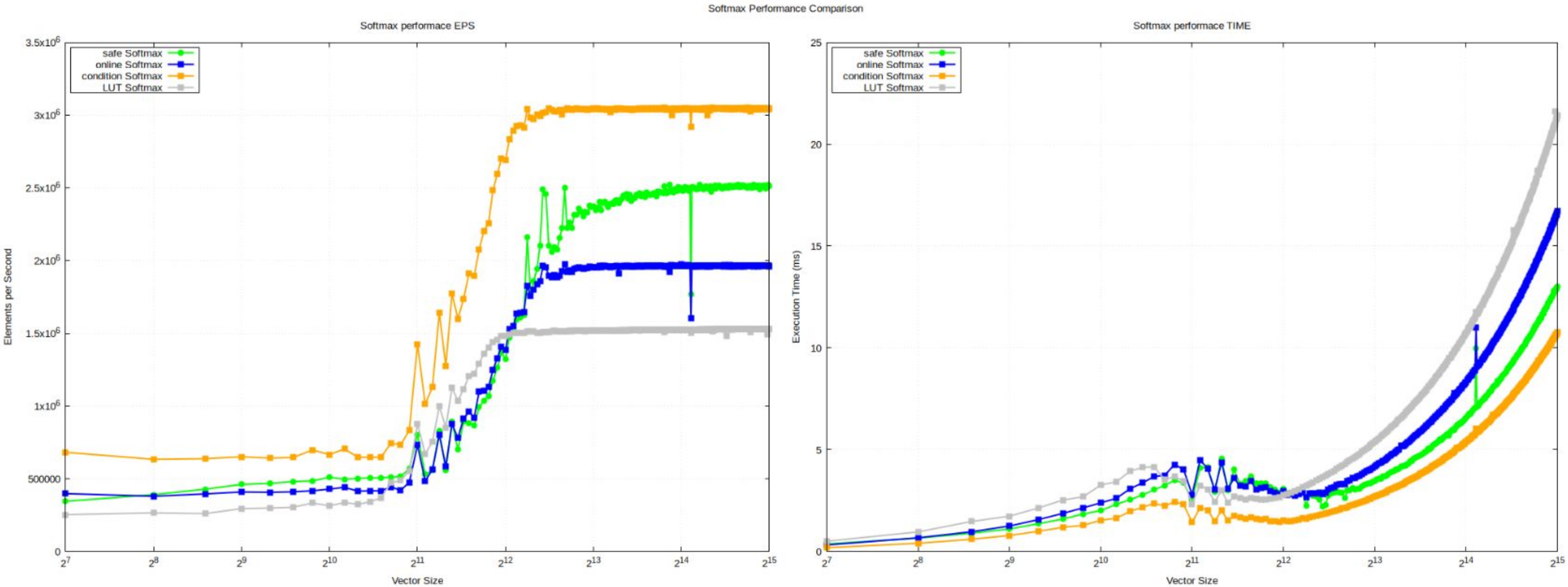
- RISC-V Platform





Experiment and Evaluation: Conditional Online Softmax

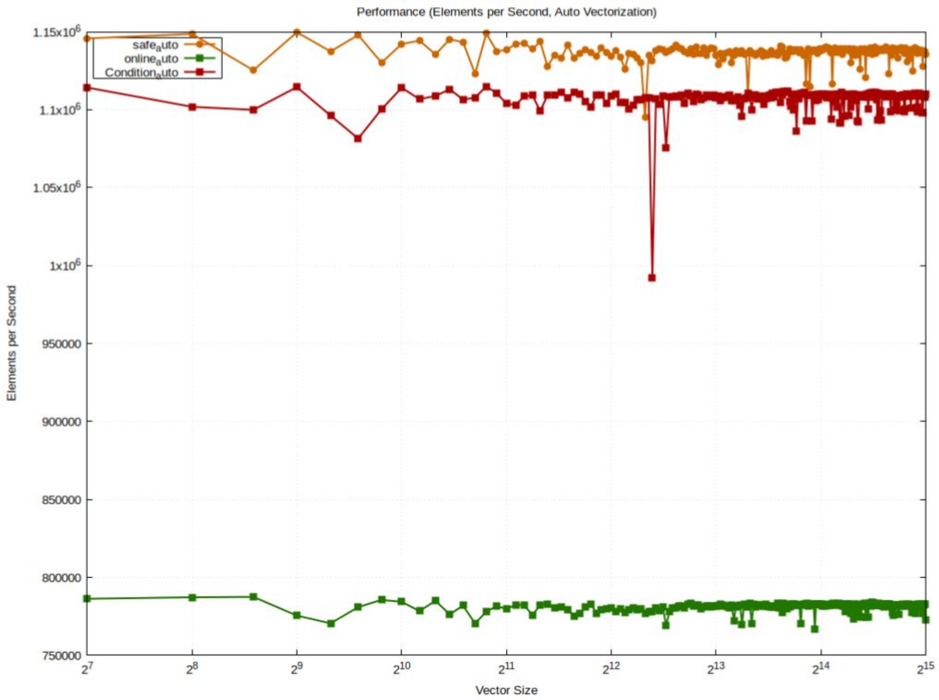
- x86 Platform

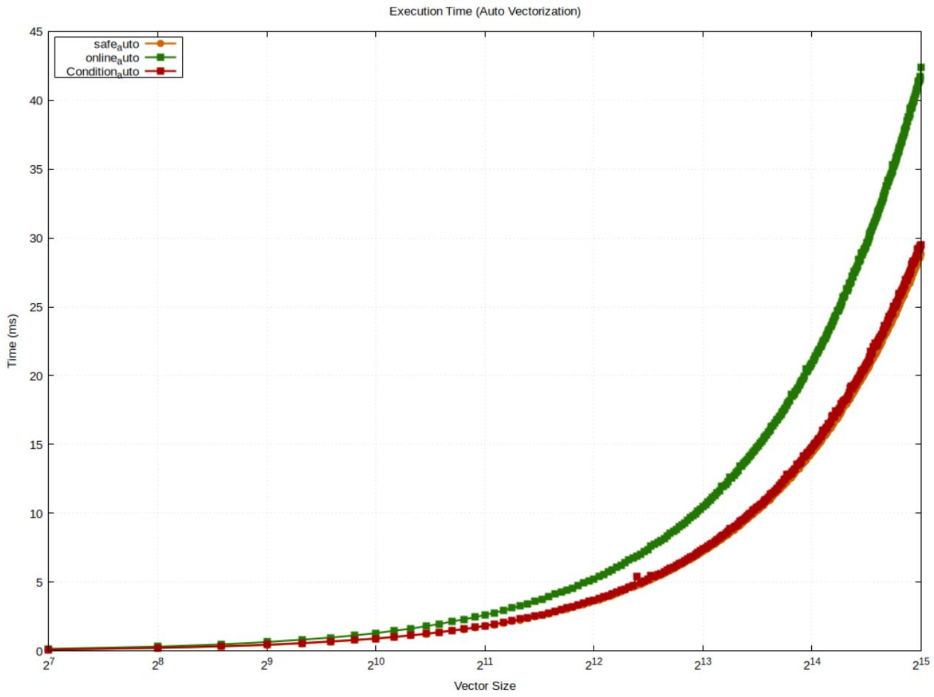


NCKU

Experiment and Evaluation: Auto Vectorization

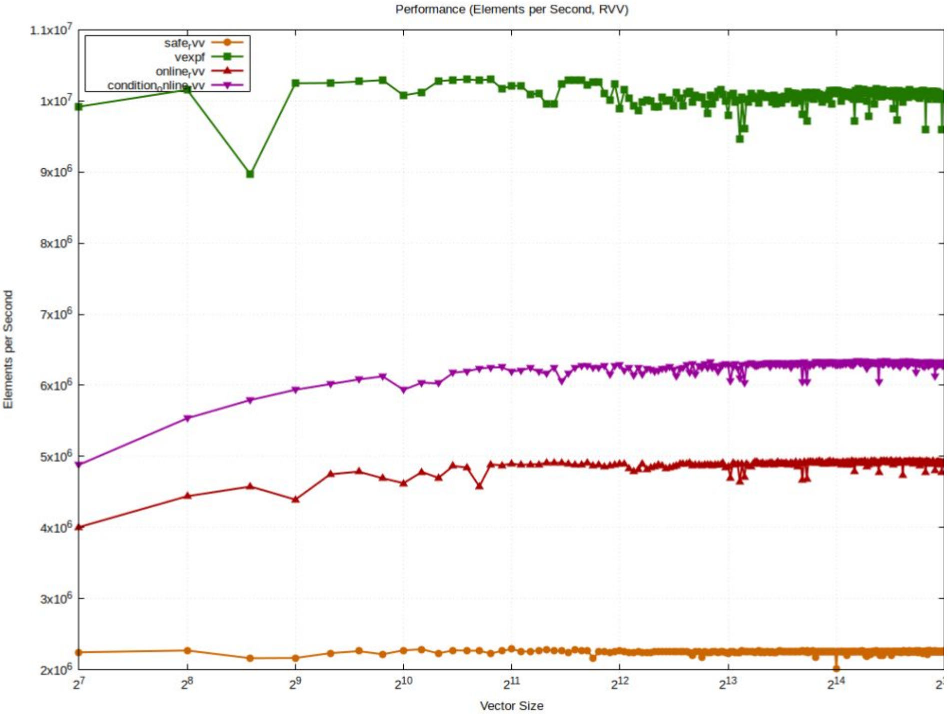
- RISC-V Platform

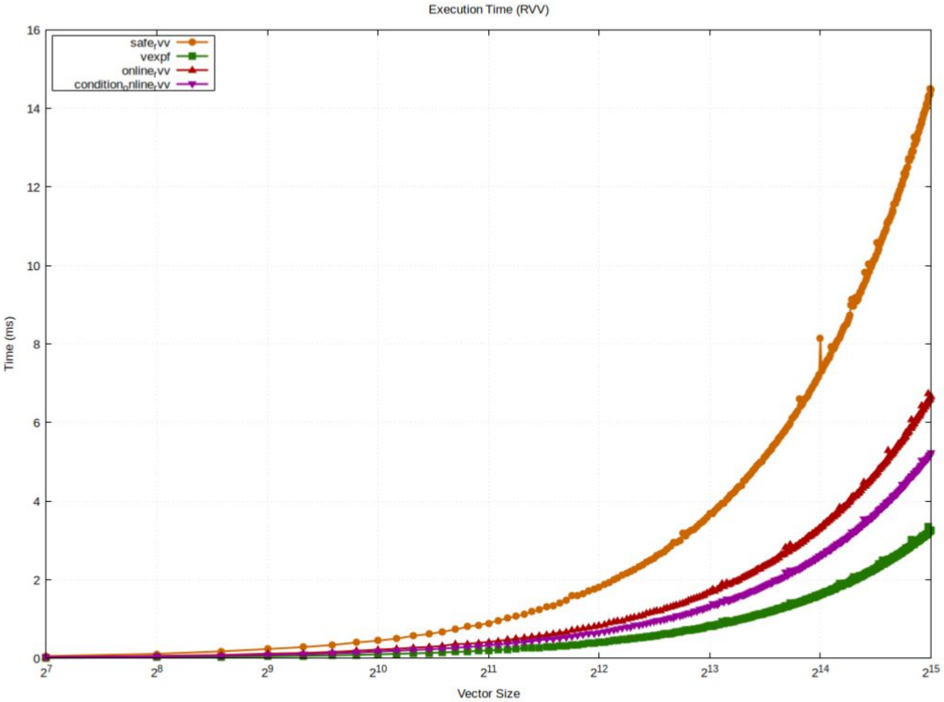




Experiment and Evaluation: Branch-free Conditional Online Softmax

- RISC-V Platform





NCKU

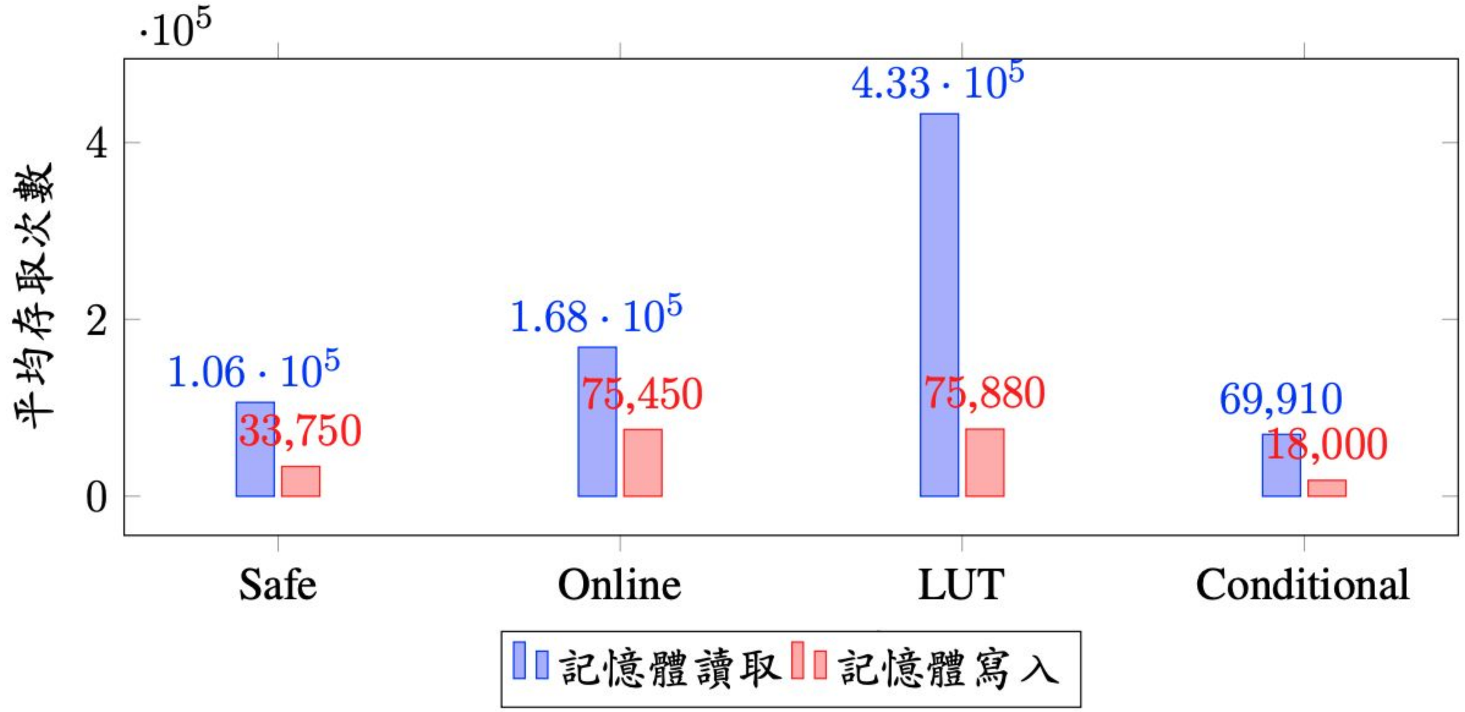
Experiment and Evaluation:

演算法	實現平台/方式	吞吐量 (Elem/s)	加速比*	記憶體訪問
Safe Softmax	Scalar (RISC-V)	1.04e6	1.00x	3R + 2W
Online Softmax	Scalar (RISC-V)	0.78e6	0.75x	2R + 1W
LUT Softmax	Scalar (RISC-V)	0.70e6	0.67x	2R + 1W
Conditional Online	Scalar (RISC-V)	**1.11e6**	**1.07x**	**2R + 1W**
Safe Softmax	Scalar (x86)	2.50e6	1.00x	3R + 2W
Online Softmax	Scalar (x86)	1.95e6	0.78x	2R + 1W
LUT Softmax	Scalar (x86)	1.5e6	0.6x	2R + 1W
Conditional Online	Scalar (x86)	**3.10e7**	1.24x	**2R + 1W**
Safe Softmax	Auto-Vectorized	2.20e6	1.00x	3R + 2W
Online Softmax	Manual RVV	4.90e6	2.22x	2R + 1W
Conditional Online	Manual RVV	**6.4e6**	**2.9x**	**2R + 1W**
Safe Softmax + vexpf	Manual RVV	**1.01e7**	**4.59x**	3R + 2W

NCKU

Experiment and Evaluation: Memory Analysis

- x86 Platform



Conclusion and Future Work

Conclusion and Future Work

Conclusion:

- Algorithmic Innovation:
  - Proposed the Conditional Online Softmax algorithm with **speedup of 7%** on RISC-V and **decreased memory access with 37%** on x86 in scalar execution.
- High-Performance Implementation:
  - Achieved **speedup of up to 2.9x** on the RISC-V platform through manual vectorization and a custom vexpf library.

Future Work:

- Hardware/Software Co-design:
  - Develop a custom vector instruction for the Conditional Online update logic.
- Algorithm Integration:
  - Integrate the optimized Softmax kernel into state-of-the-art algorithms like FlashAttention.

Q&A

Thank You