# EntroLLM: Entropy Encoded Weight Compression for Efficient Large Language Model Inference on Edge Devices

International Conference on Acoustics, Speech and Signal Processing

Arnab Sanyal[1†], Prithwish Mukherjee[‡], Gourav Datta[§], Sandeep P. Chinchali[†], Michael Orshansky[†]

[§]Case Western Reserve University
Cleaveland, OH, USA

[‡]Georgia Institute of Technology
Atlanta, GA, USA

[†]University of Texas
Austin, TX, USA

May 2026

[1] Corresponding Author – sanyal@utexas.edu

1. **Beyond-quantization compression of stored on-(edge)-device model weights**

1. **Beyond-quantization compression of stored on-(edge)-device model weights**
   Edge devices have limited storage capacity. Modern system and user applications have increasingly large AI (language) models.

1. **Beyond-quantization compression of stored on-(edge)-device model weights**

   Edge devices have limited storage capacity. Modern system and user applications have increasingly large AI (language) models.

2. **No additional accuracy degradation beyond mere quantization effects**

1. **Beyond-quantization compression of stored on-(edge)-device model weights**

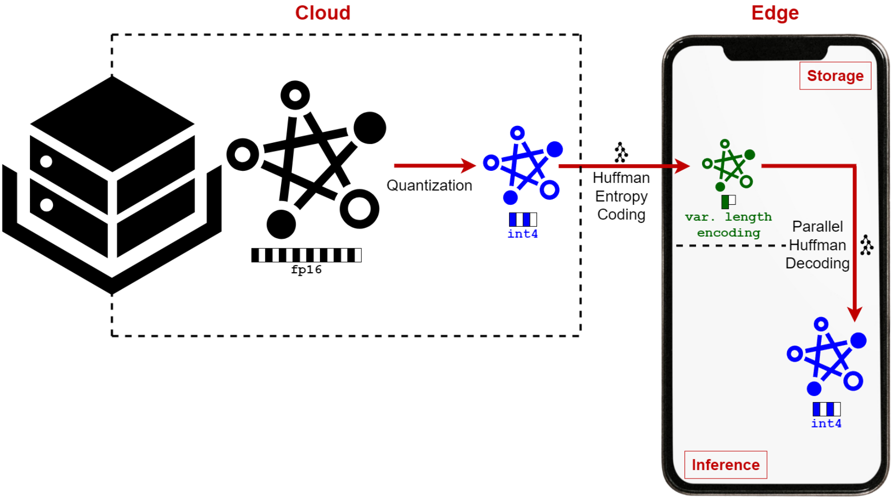
   Edge devices have limited storage capacity. Modern system and user applications have increasingly large AI (language) models.

2. **No additional accuracy degradation beyond mere quantization effects**

   Additional compression must not compromise lightweight LLM performance.

1. **Beyond-quantization compression of stored on-(edge)-device model weights**
   Edge devices have limited storage capacity. Modern system and user applications have increasingly large AI (language) models.
2. **No additional accuracy degradation beyond mere quantization effects**
   Additional compression must not compromise lightweight LLM performance.
3. **Uncompressing + token generation has competitive latency**

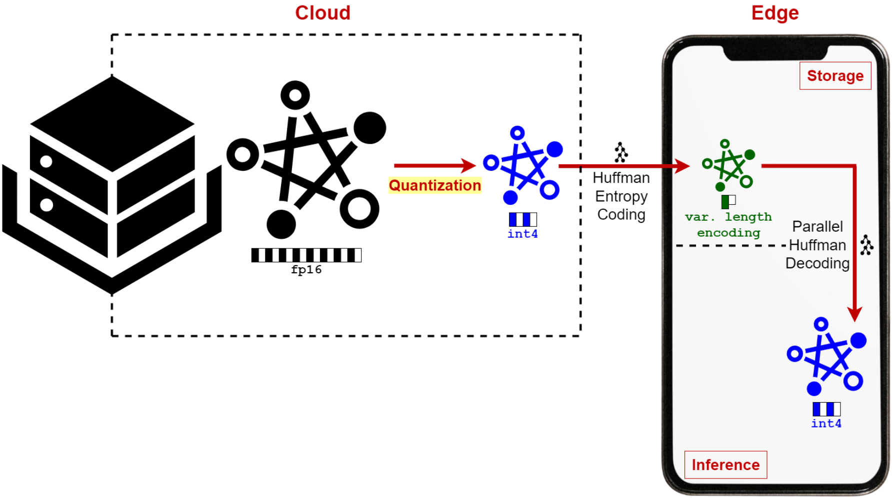
1. **Beyond-quantization compression of stored on-(edge)-device model weights**
   Edge devices have limited storage capacity. Modern system and user applications have increasingly large AI (language) models.
2. **No additional accuracy degradation beyond mere quantization effects**
   Additional compression must not compromise lightweight LLM performance.
3. **Uncompressing + token generation has competitive latency**
   The token generation rate should not fall below a certain threshold to avoid hampering `quality of service (QOS)`.



Cloud
Edge
Quantization
fp16
int4
Huffman Entropy Coding
Storage
var. length encoding
Parallel Huffman Decoding
int4
Inference

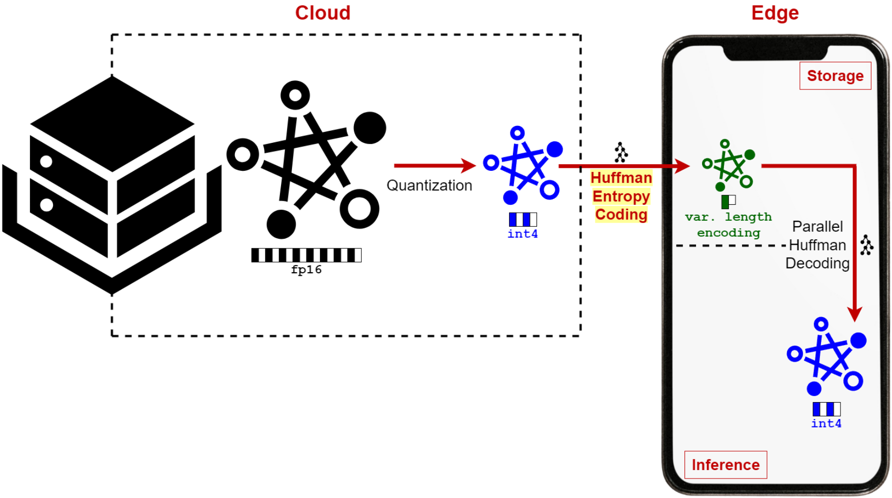
# Schematic

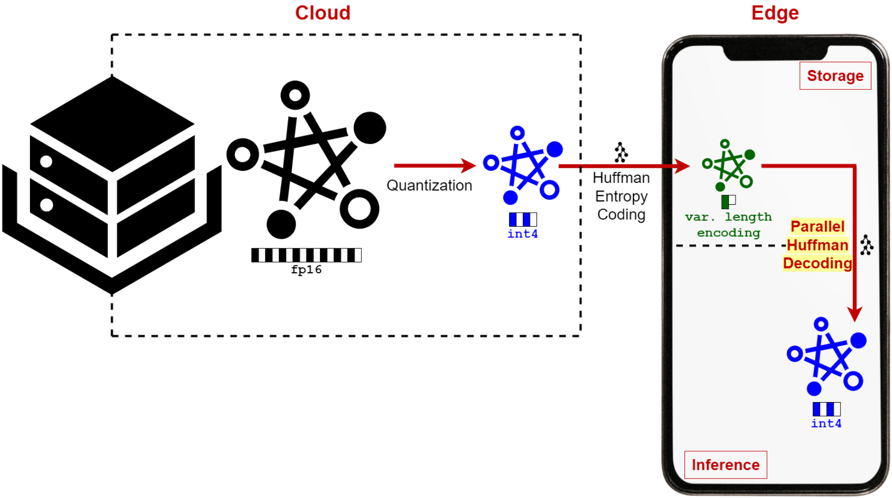
## EntroLLM



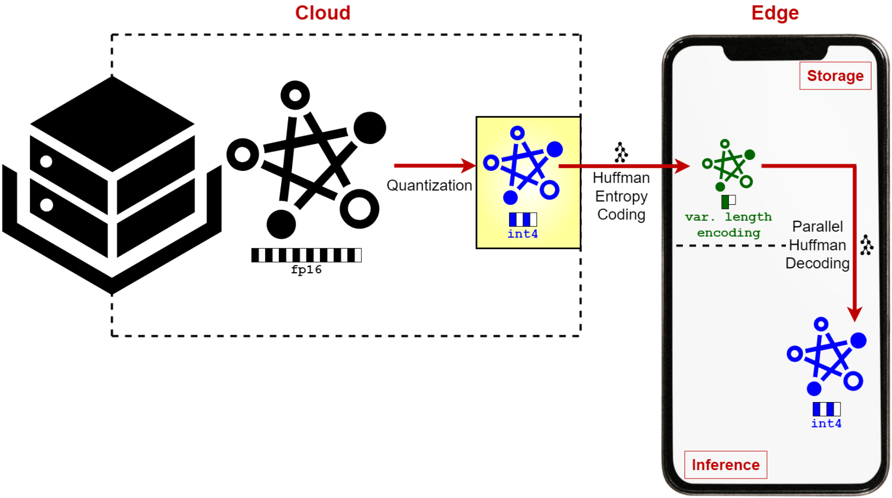
# Schematic

## EntroLLM




Cloud
Edge
Storage
fp16
Quantization
int4
Huffman
Entropy
Coding
var. length
encoding
Parallel
Huffman
Decoding
int4
Inference

# Schematic
## EntroLLM



1. **Fact:** Layer-wise, trained weights distribution follows a bell-shaped curve.[2]

[2] DOI: 10.5555/3454287.3455001

1. **Fact:** Layer-wise, trained weights distribution follows a bell-shaped curve.[2]
2. After quantization, individual layers will retain their original distribution, and the entropy of these distributions is quite high.

[2] **DOI: 10.5555/3454287.3455001**

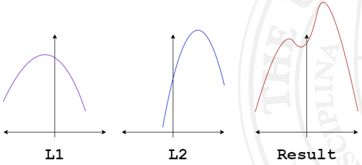
# Mixed Quantization Scheme

## EntroLLM

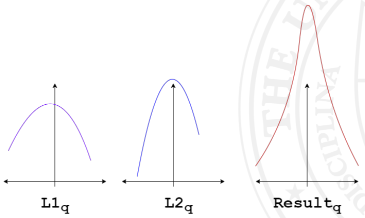
1. **Fact:** Layer-wise, trained weights distribution follows a bell-shaped curve.[2]
2. After quantization, individual layers will retain their original distribution, and the entropy of these distributions is quite high.
3. The entropy of distribution of all weights in the model is even higher. As such, Huffman compression will give little benefit.

[2] **DOI: 10.5555/3454287.3455001**

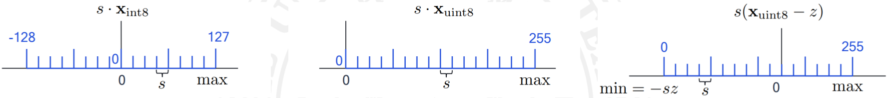
If we can somehow map various layers' floating point grids to different integer grids, such that each layer's quantized weight distributions add up to give rise to a very low entropy, high skew distribution, then we can greatly enhance compressibility.




Different layers may have different distributions, and the resulting overall distribution can have a high entropy




Through different quantization schemes, shifting distributions on a case-by-case basis allows us to skew the overall distribution, thus inducing a low entropy.




A visual explanation of the different uniform quantization grids[3] for a bit-width of 8. $s$ is the scaling factor, $z$ the zero-point. The floating-point grid is in black, and the integer-quantized grid is in blue. In our work, we use either an unsigned or an asymmetric quantization scheme on each layer based on the individual layer's weight distribution.

[2] **DOI: 10.5555/3454287.3455001**

[3] **arXiv: 2106.08295**

**for** each layer $\mathtt{k}$ in the model **do**

  **if** $\mathtt{W}^{\mathtt{k}}_{\mathtt{fp}}|_{\mathtt{max}} \times \mathtt{W}^{\mathtt{k}}_{\mathtt{fp}}|_{\mathtt{min}} \geq 0$ **then**

    $\mathtt{W}^{\mathtt{k}}_{\mathtt{int}} \leftarrow \left\lfloor \frac{\mathtt{W}^{\mathtt{k}}_{\mathtt{fp}}}{s} \right\rceil$ ▷ Unsigned

  **else**

    $\mathtt{W}^{\mathtt{k}}_{\mathtt{int}} \leftarrow \left\lfloor \frac{(\mathtt{W}^{\mathtt{k}}_{\mathtt{fp}} - z)}{s} \right\rceil$ ▷ Asymmetric

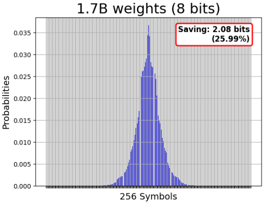
▷ $z$ is zero-point, $s$ is scaling factor

  **end if**

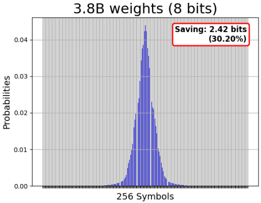
**end for**

# Model Parameter Distribution

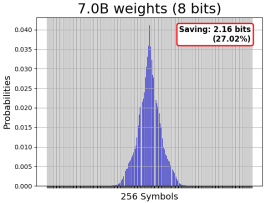
## EntroLLM



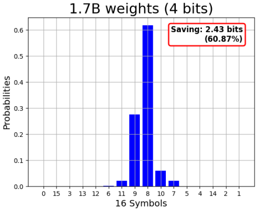

(a) `smolLM-1.7B-Instruct`



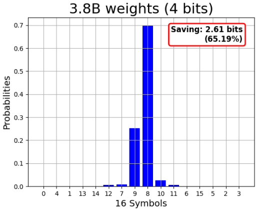

(b) `phi3-mini-4k-Instruct`



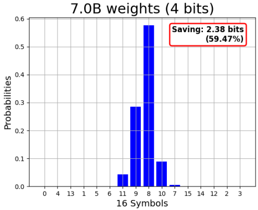

(c) `mistral-7B-Instruct`




(d) `smolLM-1.7B-Instruct`




(e) `phi3-mini-4k-Instruct`




(f) `mistral-7B-Instruct`

# Model Perplexity & Accuracy Performance

## EntroLLM

| Property | Models | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | smolLM-Instruct | | | phi3-mini-4k-Instruct | | | mistral-Instruct | | |
| Parameter Count | 1.7 Billion | | | 3.8 Billion | | | 7.0 Billion | | |
| Weight Encoding | fp16 | uint8 | uint4 | fp16 | uint8 | uint4 | fp16 | uint8 | uint4 |
| Effective Bits | 16 | **5.92** | **1.57** | 16 | **5.58** | **1.39** | 16 | **5.84** | **1.62** |
| WIKITEXT2 (ppl.) ↓ | **23.81** | 23.93 | 24.14 | **9.03** | 9.44 | 10.10 | **8.17** | 8.24 | 8.29 |
| HELLASWAG (acc.) ↑ | **25.85%** | 25.55% | 25.30% | **82.2%** | 82.10% | 81.01% | **58.37%** | 58.33% | 58.21% |
| GSM8K COT (acc.) ↑ | - | - | - | **77.37%** | 72.84% | 70.58% | **52.2%** | 48.62% | 45.36% |

**Benchmarks:** Perplexity and Accuracy benchmarks for `smolLM-1.7B-Instruct`, `phi3-mini-4k-Instruct` and `mistral-7B-Instruct` on various language tasks

# Model Compressibility

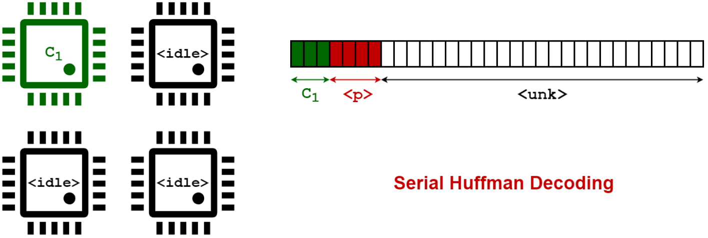
## EntroLLM

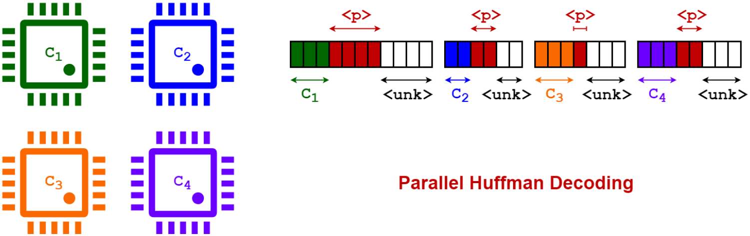
| Property | Models | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | `smolLM-Instruct` | | | `phi3-mini-4k-Instruct` | | | `mistral-Instruct` | | |
| Quantization Bits | 8 bits | | | | | | | | |
| Weight Compressibility | SOTA | ours | Improvement | SOTA | ours | Improvement | SOTA | ours | Improvement |
| Bits Saved | 0.29 | **2.08** | $\times$**7.2** | 0.30 | **2.42** | $\times$**8.1** | 0.31 | **2.16** | $\times$**7.0** |
| Quantization Bits | 4 bits | | | | | | | | |
| Weight Compressibility | SOTA | ours | Improvement | SOTA | ours | Improvement | SOTA | ours | Improvement |
| Bits Saved | 0.21 | **2.43** | $\times$**11.6** | 0.20 | **2.61** | $\times$**13.1** | 0.21 | **2.38** | $\times$**11.3** |

A comparison showing how our quantization scheme improves downstream entropy compressibility of weights versus SOTA quantization techniques

# Weight Packing for parallel Huffman decoding

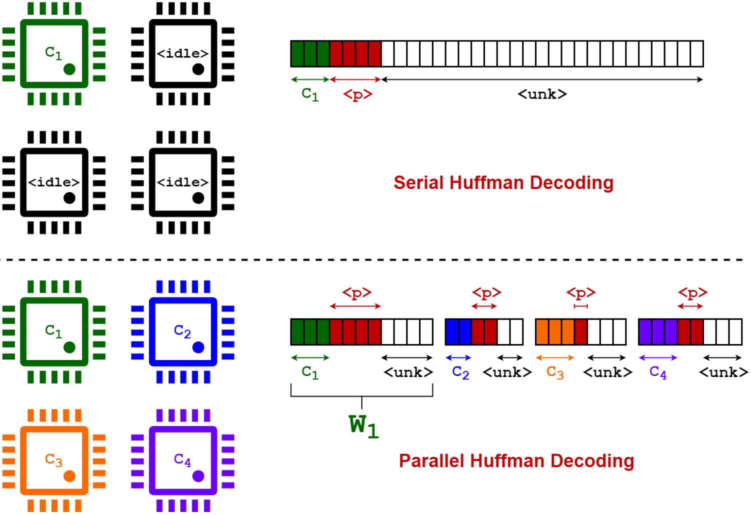
## EntroLLM



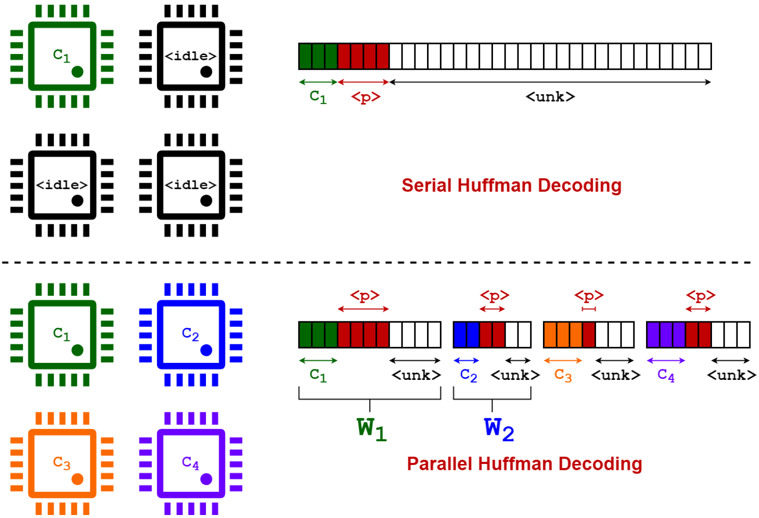
# Weight Packing for parallel Huffman decoding

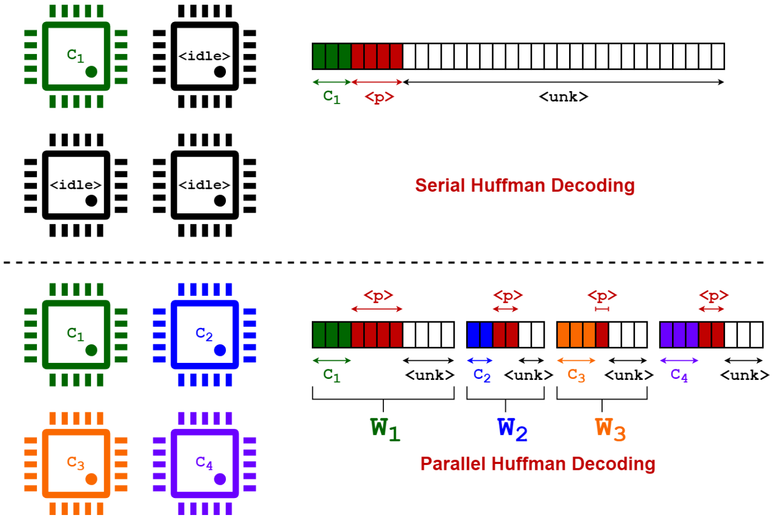
## EntroLLM



# Weight Packing for parallel Huffman decoding

## EntroLLM

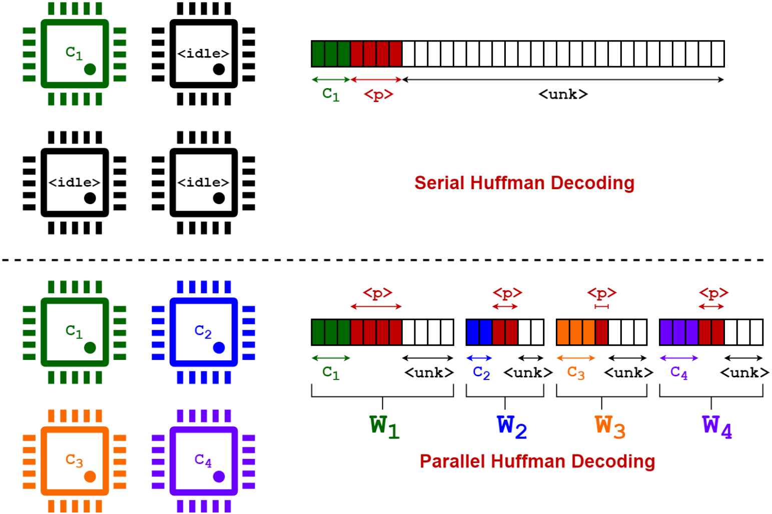




C1
<idle>
<idle>
<idle>
C1
<p>
<unk>
Serial Huffman Decoding
C1
C2
C3
C4
<p>
<p>
<p>
<p>
C1
<unk>
C2
<unk>
C3
<unk>
C4
<unk>
W1
W2
W3
Parallel Huffman Decoding

# Weight Packing for parallel Huffman decoding

## EntroLLM



# Model Latency Performance

## EntroLLM

| Task | Encoding | Latency w/o Huffman (sec) | Latency w/ Huffman (sec) |
|---|---|---|---|
| pre-fill | `uint8` | 27.10 | **23.17** |
| token generation | | 0.083 | **0.063** |
| parallel decoding | | - | 6.66 |
| first token latency | | **27.18** | 29.89 |
| pre-fill | `uint4` | 9.69 | **8.34** |
| token generation | | 0.062 | **0.025** |
| parallel decoding | | - | 1.66 |
| first token latency | | **9.75** | 10.03 |

**Latency breakdown** for the `phi3-mini-4k` model on an NVIDIA Jetson P3450 across different quantization formats (`uint8` and `uint4`) with and without Huffman compression.

ICASSP2026 Barcelona

# Acknowledgements

Thanks to Amir Gholami, Coleman Richard Charles Hooper, and Kurt Keutzer for their inputs during ideation.