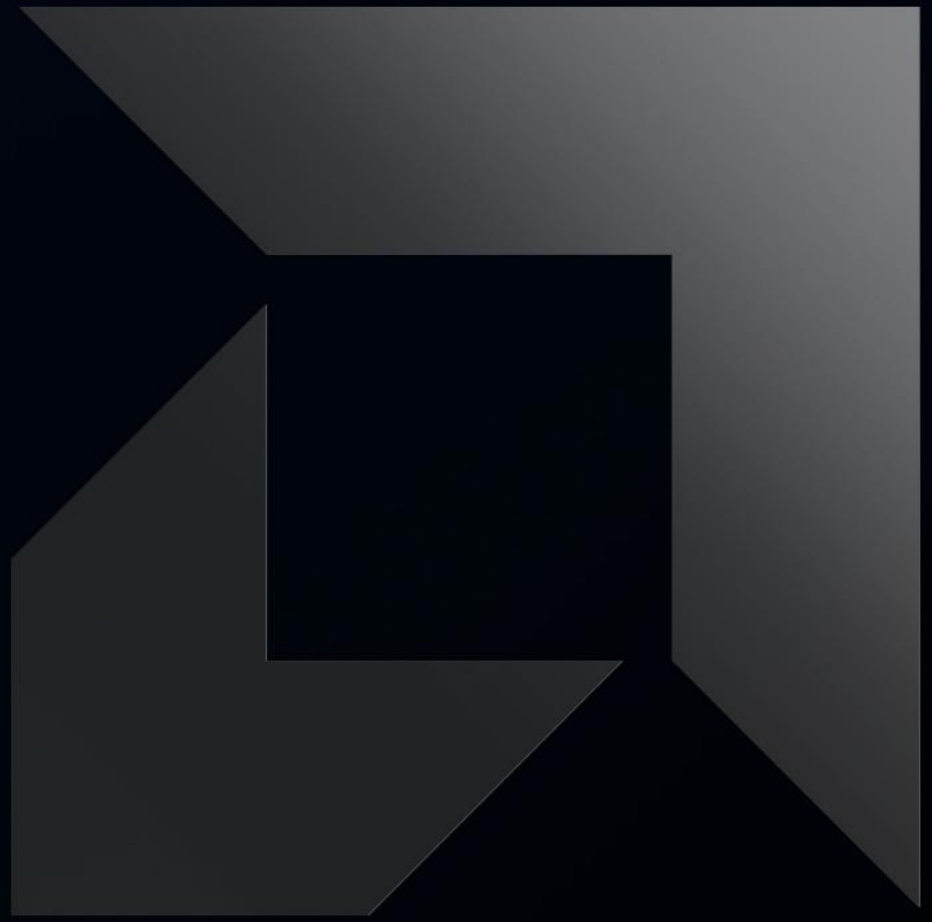


End of Internship Presentation

Qronos Implementation, int3 datatypes

by Matteo Melis

AMD
together we advance_

Hello! I’m Matteo

- Recently graduated from BSc Maths and Computer Science at Queen’s University Belfast.
- Working in the AIG AI team as a software developer intern, focusing on the Quark library.
- My main interests are in abstract algebra and ML.
- Outside of work I enjoy running, gym and playing football.

Introduction: GPTQ

(arXiv:2210.17323)

What it achieves:

- Fast and accurate PTQ algorithm for LLM's.
- Preserves perplexity/quality to a high level on low bit precision e.g. 3/4 bits. (current SOTA)

Core Idea:

- We want:

$$argmin_{Q_l} \left|\left| W_l\tilde{X}_l - Q_l\tilde{X}_l \right|\right|_2^2 ,$$

where $\tilde{X}_l$ denotes the input at layer l, W_l is the weight at layer l and Q_l is the quantized representation of W_l.

How ?

- Sequential, column-wise quantization with alternating rounding and error diffusion.
- Use a second-order (Hessian) approximation to guide each column's rounding.
- Immediately diffuse the rounding error to remaining columns (via H^{-1}) so future steps compensate.

Introduction: Qronos Motivation

(arXiv:2505.11695)

GPTQ	Pro/con
Efficient and quality preserving with little fine-tuning and relatively small calibration data set.	●
Diffuses the local quantization error to future columns.	●
At an arbitrary layer, the layer-wise objective uses the input matrix from the partially quantized model ($\tilde{X}_l$) which carries propagated quantization errors from previous layers.	●

Qronos goal:

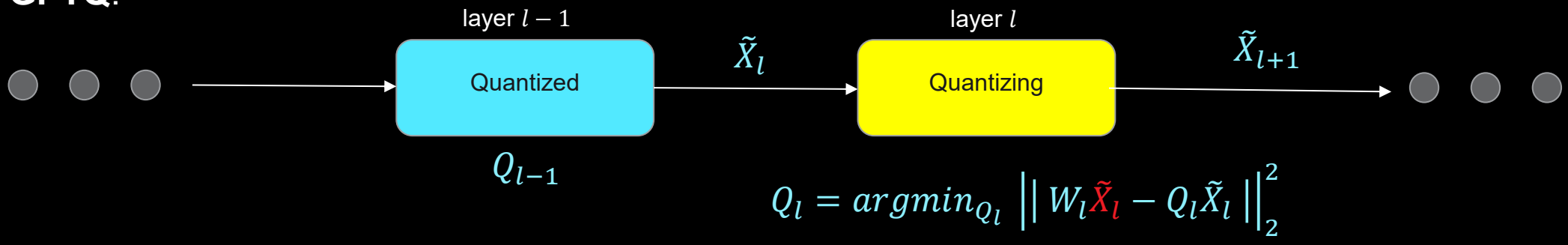
- Solve the mismatched layer-wise correction problem by tracking the true original input.
- Let $\tilde{X}_l$ denote the input at layer l of the partially quantized model and let X_l denote the pre-quantized model's input for layer l, we shift our minimisation objective to:

$$argmin_Q \left|\left| W_l\tilde{X}_l - Q_l\tilde{X}_l \right|\right|_2^2 \longrightarrow argmin_Q \left|\left| W_lX_l - Q_l\tilde{X}_l \right|\right|_2^2$$

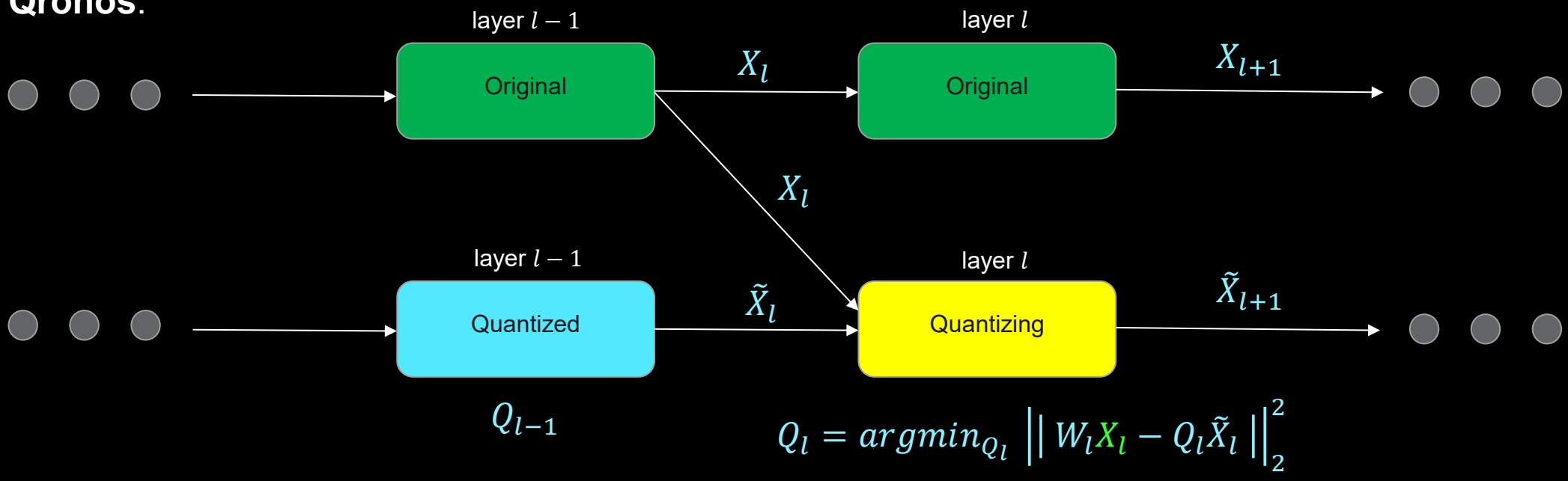
(GPTQ) (Qronos)

Introduction: GPTQ vs Qronos

GPTQ:



Qronos:



Qronos Algorithm

Algorithm 1 Qronos

$H^{-1} = (\widetilde{X}^\top \widetilde{X})^{-1} = LL^\top$
for every w in W (in parallel) **do**
  $q = \mathbf{0}^N$
  $w^{(0)} \leftarrow \text{copy}(w)$
  $q_1 = \mathcal{Q}\left(\frac{\widetilde{X}_1^\top (Xw - \widetilde{X}_{\geq 2} w_{\geq 2}^{(0)})}{\|\widetilde{X}_1\|^2}\right)$ ▷ Calc. q_1 from (8), use (16) for a memory efficient ver.
  $w_{\geq 2}^{(1)} = \widetilde{X}_{\geq 2}^\dagger \left(Xw - \widetilde{X}_1 q_1\right)$ ▷ Calc. $w_{\geq 2}^{(1)}$ from (9), see (17) for a memory efficient ver.
  for $t = 2$ to N **do** ▷ Efficient implementation (Lemma 3.2)
    $q_t = \mathcal{Q}(w_t^{(t-1)})$
    $w_{\geq t+1}^{(t)} = w_{\geq t+1}^{(t-1)} - L_{\geq t+1,t} \cdot (w_t^{(t-1)} - q_t)/L_{tt}$
  end for
end for
return Q

Error Correction

Error Diffusion

GPTQ loop

Qronos Algorithm

- To ease notation and implementation, we can work with correlation matrices defined as $G := \tilde{X}^T X \in \mathcal{R}^{NxN}$ and $\mathrm{H} := \tilde{X}^T \tilde{X} \in \mathcal{R}^{NxN}$.

- Step 1 - **Error correction**:

$$q_1 = \mathcal{Q}\left(\frac{G_{1,\geq 1} w - H_{1,\geq 2} w_{\geq 2}}{H_{1,1}}\right),$$

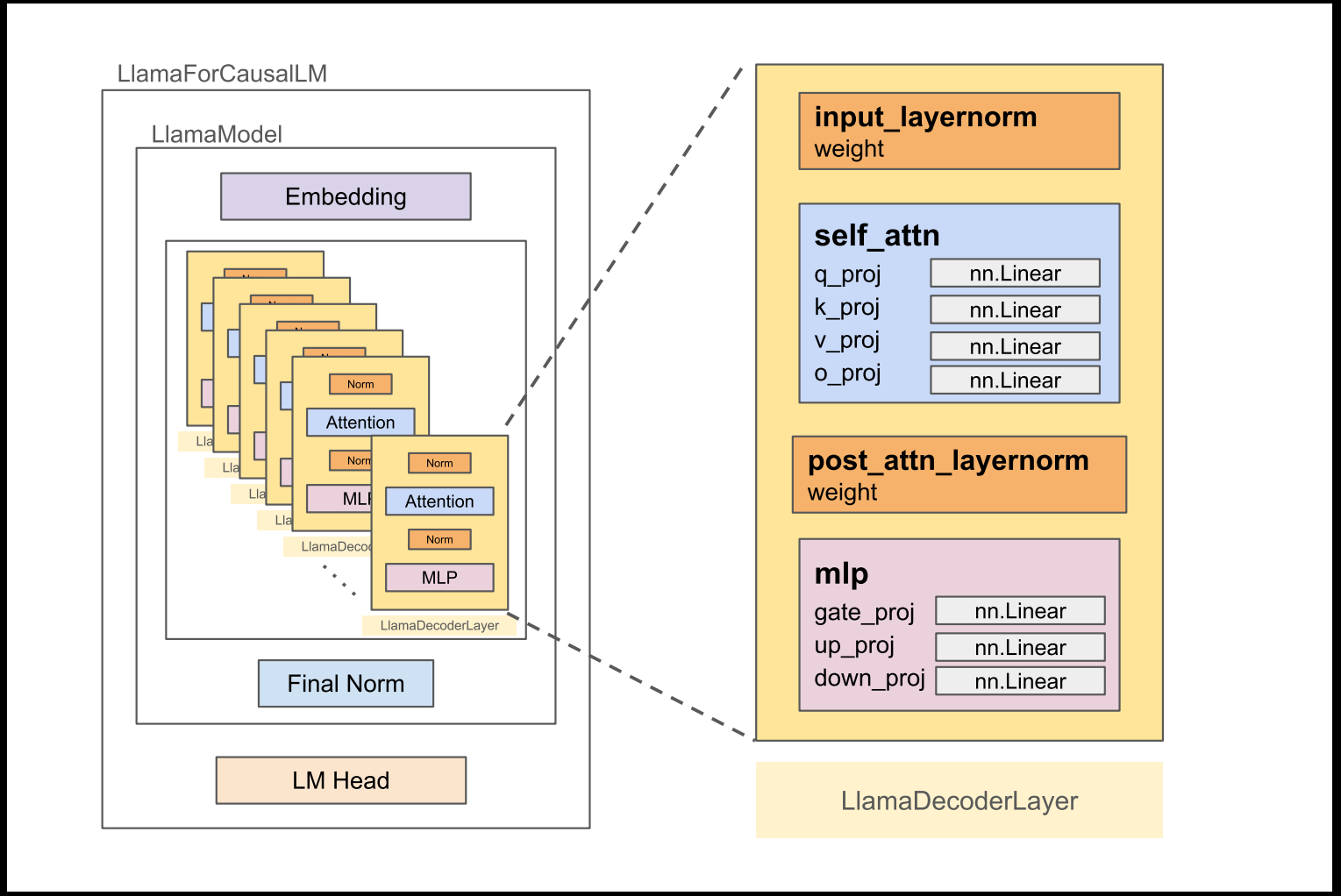
- Step 2 – **Error diffusion**:

$$w_{\geq 2} = H^{-1}_{\geq 2,\geq 2}\left(G_{\geq 2,\geq 1} w - H_{\geq 2,1} q_1\right),$$

- Step 3 – **GPTQ loop**:

Calculate $q_{\geq 2}$ using the error diffused $w_{\geq 2}$

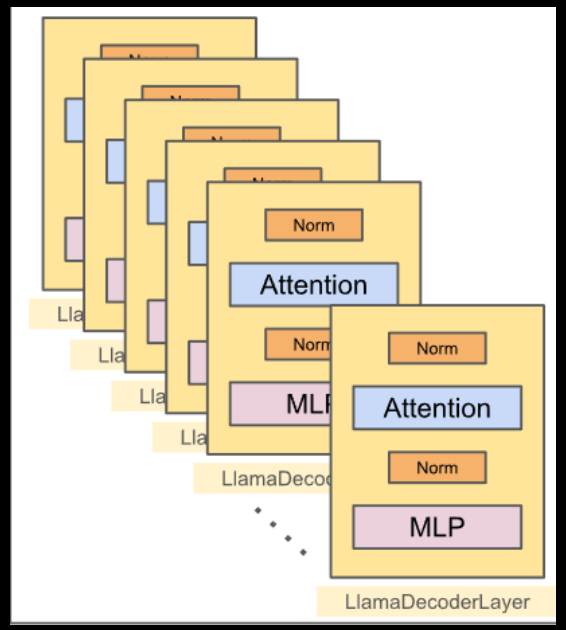
Quark Implementation: Core Algorithm Orchestration



- Qronos applied iteratively to each linear layer.
- Allows efficient memory clean up.
- G matrix only in memory for each linear layer.

Quark Implementation: Core Algorithm Orchestration

Llama Model



loop through

Decoder Layer

input_layernorm
weight

self_attn
q_proj
k_proj
v_proj
o_proj

post_attn_layernorm
weight

mlp
gate_proj
up_proj
down_proj

loop through

Linear Layers

self_attn.q_proj

⋮

mlp.down_proj

For each linear layer:

1. Register original weight buffer
2. Pre-compute $\mathrm{H} = \tilde{X}^T \tilde{X}$ and $\mathrm{G} = \tilde{X}^T X$ with forward hooks on a per sample basis
3. Apply core Qronos algorithm to layer
4. Store Q as weight matrix

Q

After applying Qronos to each linear layer:

1. Two forward passes on entire decoder block
2. Store X and $\tilde{X}$ for next decoder block

Quark Implementation: Challenges Faced

1. Original model tracking
   - Context manager
   - Original weight buffer
   - Forward hooks for matrices
   - Memory management
2. Brevitas vs quark
   - Inheritance heavy
   - Nested logic from GPFQ algo
3. Introduction to R&D
   - Cross referencing paper
   - Translating into code

Experimental Results

Llama-3.2-3B – group size 128

Quant Algo	Quant Scheme	WikiText2 PPL (↓)	Pileval PPL (↓)
N/A	N/A	7.82	7.82
GPTQ	W_UINT4	8.07	17.82
Qronos	W_UINT4	***8.01***	***8.14***
GPTQ	W_INT3	9.87	11,117.4
Qronos	W_INT3	***9.14***	***13.41***
GPTQ	W_MXFP4_A_MXFP4	8.07	8.2
Qronos	W_MXFP4_A_MXFP4	***8.02***	***8.16***

***More comprehensive list of evals can be found here**

Experimental Results

Llama-3.2-1B – group size 128

Quant Algo	Quant Scheme	WikiText2 PPL (↓)	Pileval PPL (↓)
N/A	N/A	9.75	9.75
GPTQ	W_UINT4	10.53	12.93
Qronos	W_UINT4	***10.20***	***10.54***
GPTQ	W_INT3	13.22	158.53
Qronos	W_INT3	***12.54***	***15.68***
GPTQ	W_MXFP4_A_MXFP4	12.42	13.46
Qronos	W_MXFP4_A_MXFP4	***12.39***	***12.78***

***More comprehensive list of evals can be found here**

Side Project: Int3 Quantization support in Quark

Motivation:

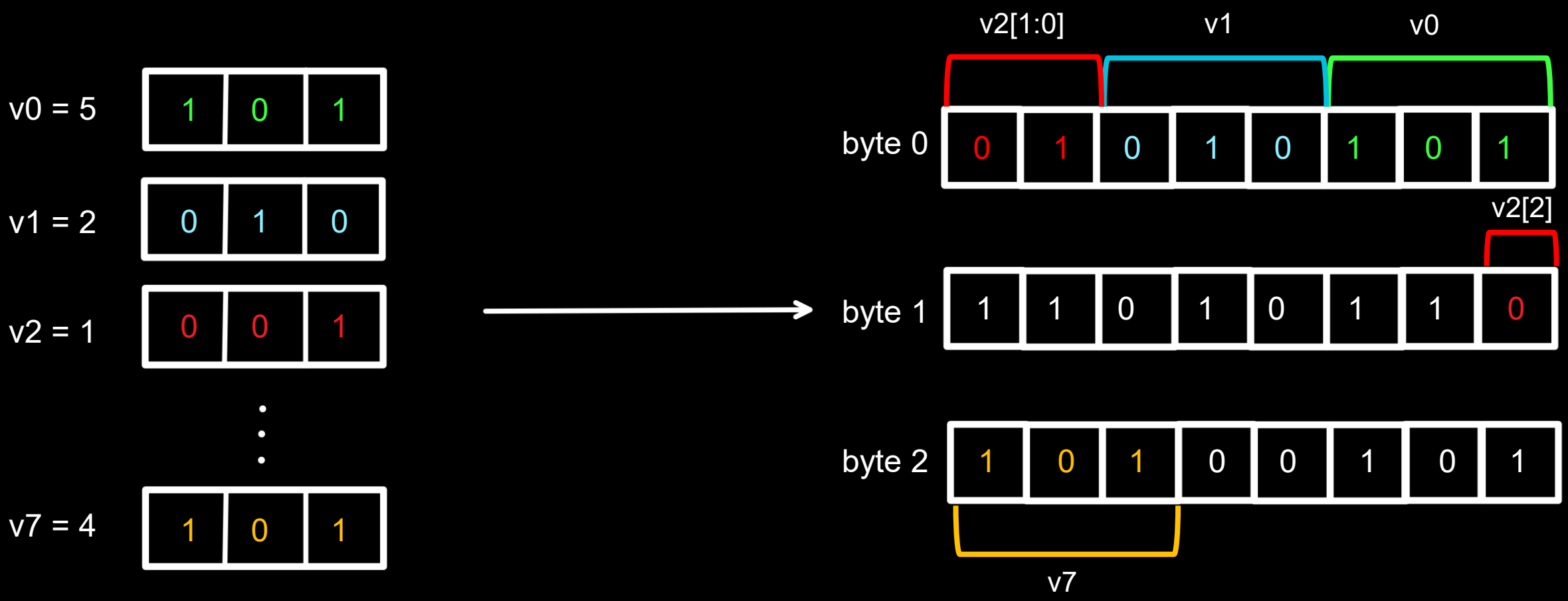
- Qronos paper shows strong results with int3 quantization.
- int3 is a ‘happy middle ground’ for low-bit precision.

Implementation overview:

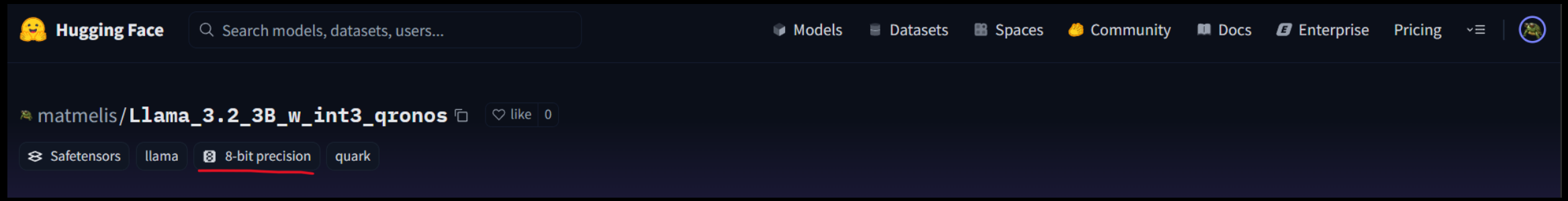
- Add int3 as a recognised datatype in Quark.
- Ensure it can be serialized properly i.e. exported and later used for inference.
- Serialization is non-trivial, int3 is not a native datatype in torch.

Int3 Quantization: packing logic

Simple approach: map 8 uint3 values into 3 bytes (uint8)



Int3 Quantization: serialized model exported to HF



model.layers.0.mlp.down_proj (3)		
model.layers.0.mlp.down_proj.weight	[2 048, 3 072]	U8
model.layers.0.mlp.down_proj.weight_scale	[2 048, 64]	F32
model.layers.0.mlp.down_proj.weight_zero_point	[2 048, 24]	U8

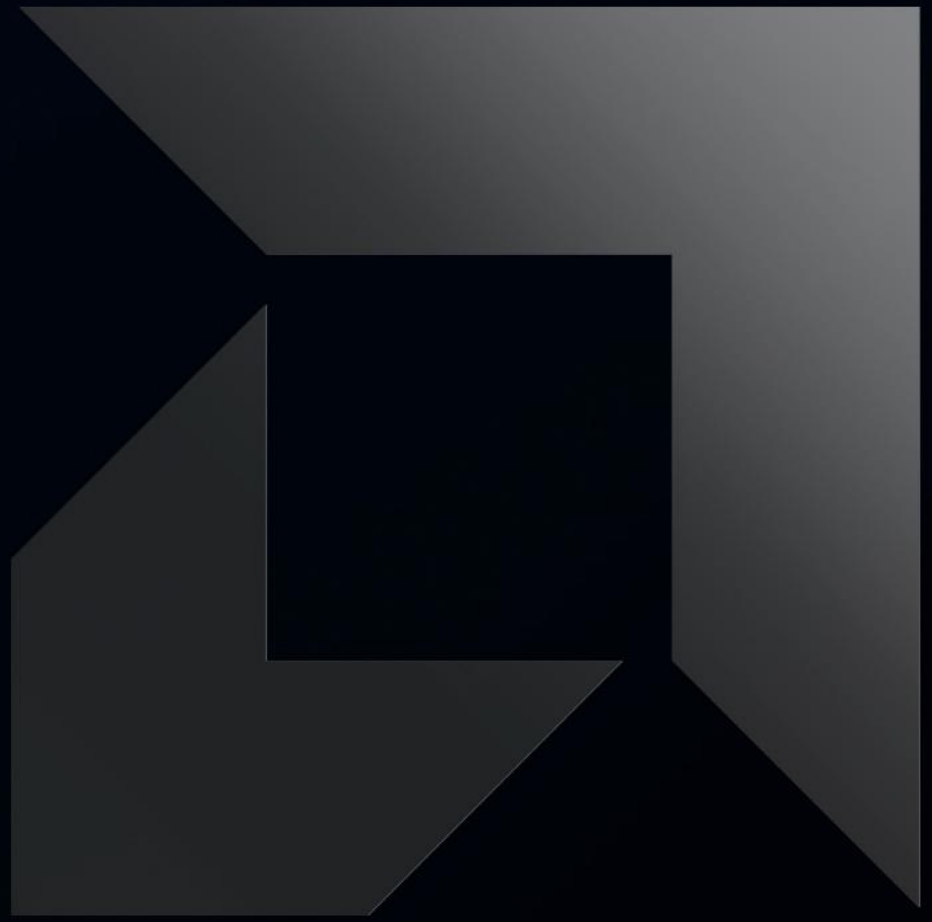
What I took away from the Internship

- First time “implementing a paper” – very fulfilling
- Quantization “bootcamp”
- Contributing to Quark – creating issues, merging quick fixes to bugs, pr discussions ...
- Many new challenges – R&D, bitwise operators, evaluating and quantizing models
- Cross team collaboration – major thank you to Ian Colbert!

Future Work

- Explore other PTQ/GPTQ enhancing algorithms – GPTAQ
- Implement CUDA graphs for Qronos – 2-4x speedup at quantization time.
- Further evaluations – more quant schemes, more eval metrices, block to block error
- Implement more SOTA algorithms in quark – many recent papers that look promising

Questions?



References

- Qronos paper
- GPTQ paper
- Quark implementation of qronos
- Int3 pr
- Hugging face models

AMD
together we advance_


AMD