Half-Quadratic Quantization of Large Machine Learning Models

Hicham Badri, Appu Shaji

Mobius Labs GmbH

Large Language Models (LLMs) have revolutionized various subfields of machine learning like natural language processing, speech recognition and computer vision, enabling machines to understand and generate outputs with unprecedented accuracy and fluency. However, one of the most critical challenges in deploying LLMs is their expensive memory requirements, for both training and inference. Quantization methods such as bitsandbytes, GPTQ and AWQ have made it possible to use large models such as the popular Llama-2 with significantly less memory, enabling the machine learning community to conduct remarkable research using a single consumer-grade GPU.

In this article, we propose a new quantization technique called **H**alf-**Q**uadratic **Q**uantization (**HQQ**). Our approach, requiring no calibration data, significantly speeds up the quantization of large models, while offering compression quality competitive with that of calibration-based methods. For instance, **HQQ** takes less than 5 minutes to process the colossal Llama-2-70B, that's over *50x* faster compared to the widely adopted GPTQ. Our Llama-2-70B quantized to 2-bit outperforms the full-precision Llama-2-13B by a large margin for a comparable memory usage.

Table of Contents

Support code is available at https://github.com/mobiusml/hqq

Talk to us at

Introduction

Model quantization is a crucial step to deploy large models with limited resources and save costs, which is particularly relevant to LLMs for both training and inference. Software packages such as bitsandbytes have mad it possible to utilize large models on consumer-grade GPUs, which has been a game-changer for the machine learning community.

When it comes to weight-only quantization, there are two classes of approaches: data-free calibration techniques such as bitsandbytes rely or using the weights only without external data, and calibration-based methods such as GPTQ and AWQ that rely on an external dataset. While calibration-based methods offer better quantization quality, they suffer from two main issues:

1. *Calibration data bias*: the quality of quantization can be negatively affected depending on the calibration data provided.
2. *Quantization time*: calibration can be a heavy computational process especially for very large models, which makes it difficult to test and deploy multiple models.

Wouldn't it be great if we can achieve the quality of calibration-based methods for the speed of calibration-free quantization methods? That's

exactly what we propose via our method Half-Quadratic Quantization (HQQ).

Half-Quadratic Quantization

Basic quantization often results in a loss of model accuracy. This is because the weights in these models can have a wide range of values that can be significantly altered after the quantization process. Weights that deviate from the distribution, notably known as outliers, pose a particular challenge. Group-wise Precision Tuning Quantization (GPTQ) and Activation-Aware Layer Quantization (AWQ) are algorithms that try to overcome this issue by relying on calibration data to minimize the error on layer outputs.

Unlike these approaches, our method focuses specifically on minimizing errors in the *weights* rather than the layer activation. Additionally, by incorporating a sparsity-promoting loss, such as the $l_{p<1}$ -norm, we effectively model outliers through a hyper-Laplacian distribution. This distribution more accurately captures the heavy-tailed nature of outlier errors compared to the squared error, resulting in a more nuanced representation of error distribution.

We propose a robust optimization formulation to find the quantization parameters (zero-point z and scaling s). More specifically, we use a sparsity-promoting loss function $\phi()$ such as the l_p norm between the original weights W and their dequantized version:

$$\underset{z,s}{\text{argmin}}\ \phi(W - Q_{z,s}^{-1}(Q_{z,s}(W)),$$

where $Q_{z,s}()$ is the quantization operator which depends on the z and s parameters and generates the quantized weights W_q. $Q_{z,s}()^{-1}$ is the de-quantization operator:

$$Q_{z,s}(W) = \text{round}(W/s + z) = W_q$$
$$Q_{z,s}^{-1}(W_q) = s(W_q - z)$$

The use of the $l_{p<1}$ -norm makes the problem non-convex. To find a solution, we adopt a Half-Quadratic solver by introducing an extra variable W_e. This additional parameter allows us to split the main problem into sub-problems that are easier to solve. Moreover, to make th problem simpler, we fix the scaling parameter s and only optimize for the zero-point z.

$$\underset{z,W_e}{\text{argmin}}\ \phi(W_e) + \frac{\beta}{2}||W_e - (W - Q_z^{-1}(Q_z(W))||_2^2$$

We then form sub-problems which are solved via alternate optimization:

$$(\text{sp}_1) \quad W_e^{(t+1)} \leftarrow \underset{W_e}{\text{argmin}}\ \phi(W_e) + \frac{\beta^{(t)}}{2}||W_e - (W - Q_z^{-1}(Q_z(W))||_2^2$$

$$(\text{sp}_2) \quad z^{(t+1)} \leftarrow \underset{z}{\text{argmin}}\ \frac{1}{2}||Q_z^{-1}(Q_z(W)) - (W - W_e^{(t+1)})||_2^2$$

$$\beta^{(t+1)} \leftarrow \kappa\beta^{(t)},$$

where β and κ and strictly positive parameters.

Sub-problem (sp_1)

This problem takes the form of a Proximal Operator. When $\phi()$ is the l_1 norm, the solution is the soft-thresholding operator. There exists a more general thresholding solution for the l_p -norm with $0 \leq p \leq 1$ that we adopt known is as the generalized soft-thresholding operator:

$$W_e^{(t+1)} \leftarrow \text{shrink}_{l_p}\left(W - Q_z^{-1}(Q_z(W)), \beta\right)$$

$$\text{shrink}_{l_p}(x, \beta) = \text{sign}(x)\text{relu}(|x| - \frac{|x|^{p-1}}{\beta})$$

Sub-problem (sp_2)

The second sub-problem can be rewritten as follows:

$$z^{(t+1)} \leftarrow \underset{z}{\text{argmin}}\ \frac{1}{2}||z - \left(W_q^{(t+1)} - \frac{(W - W_e^{(t+1)})}{s}\right)||_2^2$$

$$W_q^{(t+1)} = \text{round}(W/s + z^{(t)})$$

The solution is simply the average over the axis the quantization groupin is performed on:

$$z^{(t+1)} \leftarrow \langle W_q^{(t+1)} - \frac{(W - W_e^{(t+1)})}{s}\rangle$$

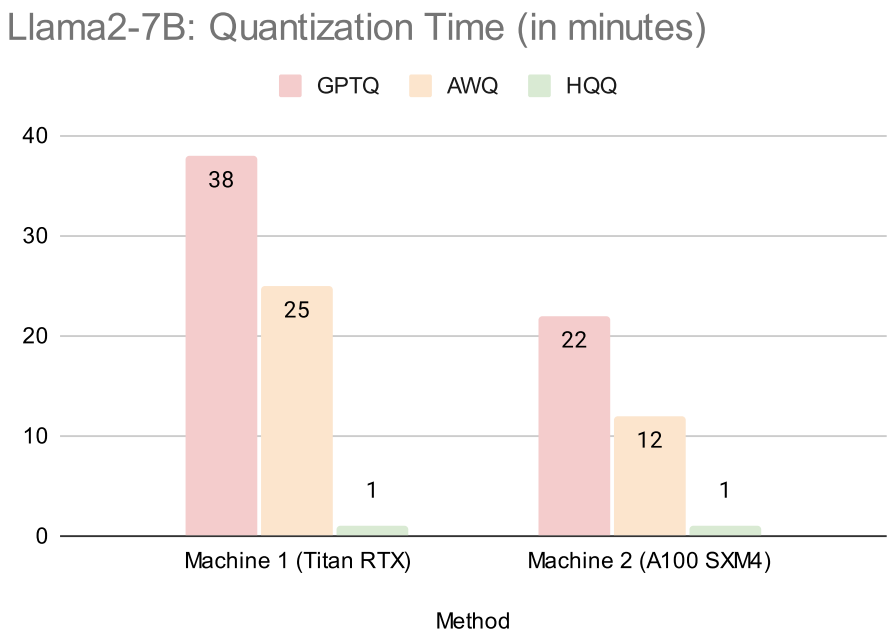
In our implementation, we work with the inverse of the scale $1/s$ instead of s which we found to be a bit more stable with the half-precision calculations.

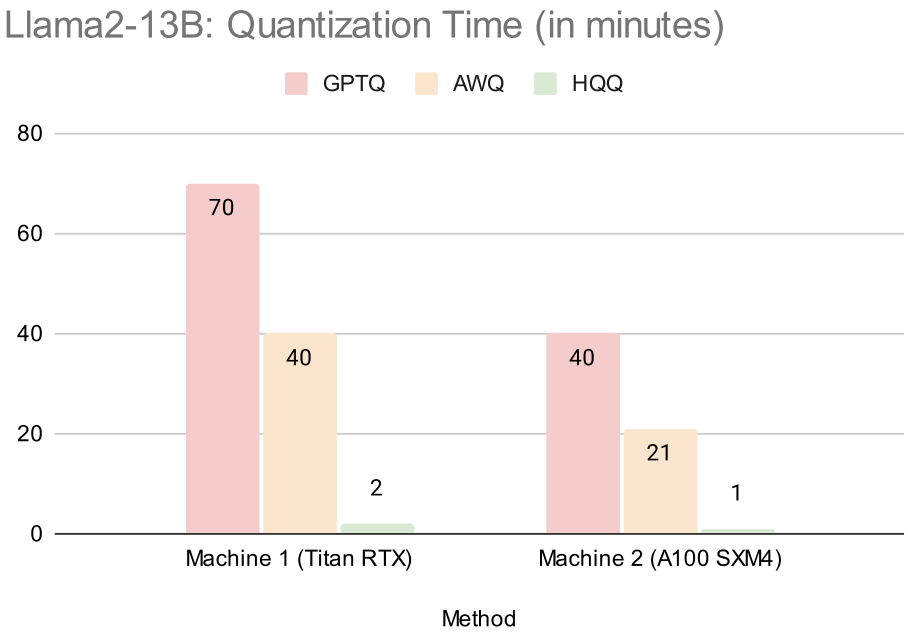
Note that, contrary to using gradient descent with autograd, the approach that we propose relies on closed-form solutions, which means that there

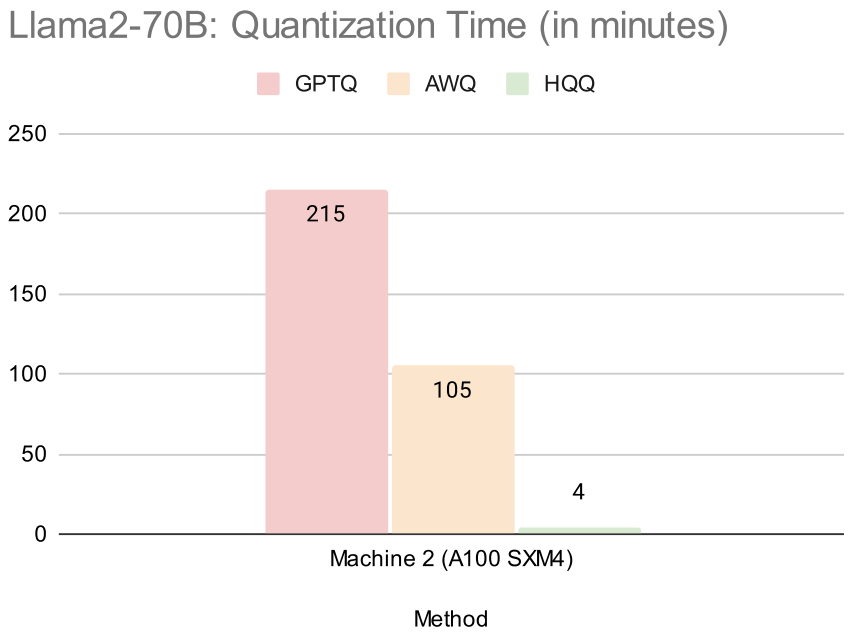
are no gradients calculated. This allows us to run all the calculations in inference mode with half-precision. Moreover, it only takes a few iterations for the solver to converge. Conversely, using the AdamW optimizer and Pytorch's autograd takes thousands of iterations to achieve good results. It also fails with $p < 1$, which is what we actually use to promote sparsity. Thanks to the Half-Quadratic solution, our quantization method achieves significant speed-up (over **100x** faster vs. autograd to quantize Llama-2-7B), being able to process even the largest models in only a few minutes.

Processing Time

We report the processing time to quantize the Llama-2 models. We noticed that the processing time for GPTQ and AWQ drastically changes from one machine to another. Our method performs the whole quantization on the GPU with half-precision and only uses the CPU to transfer data to the GPU once the solver is finished. HQQ takes only a few minutes to quantize the largest Llama-2-70B model, which is over 50x faster compared to GPTQ.







Benchmark

Llama-2 Benchmark

To measure the quantization quality of our method, we use the perplexity metric (**PPL**) on the widely adopted wikitext2 dataset. We also report the runtime GPU memory in GB (**MEM**) the session takes to run the quantized model (additional memory is required for prediction depending on the sequence length). We compare against the popular approaches widely used by the community: BNB (bitsandbytes) , GPTQ via AutoGPTQ and AWQ via AutoAWQ.

Regarding the parameters, we fix the Half-Quadratic solver with the following: *p=0.7*, *beta=1*, *kappa=1.01*, *iterations=20*. Additionally, we use early-stopping to exit the solver when the error doesn't improve. We haven't experimented much with the parameters, so different settings might actually yield better results. Similar to the other approaches, we us grouping to quantize the weights into buffers (*_g128* means we use a group-size of 128). We also quantize the zero-point into 8-bit without grouping or optimization.

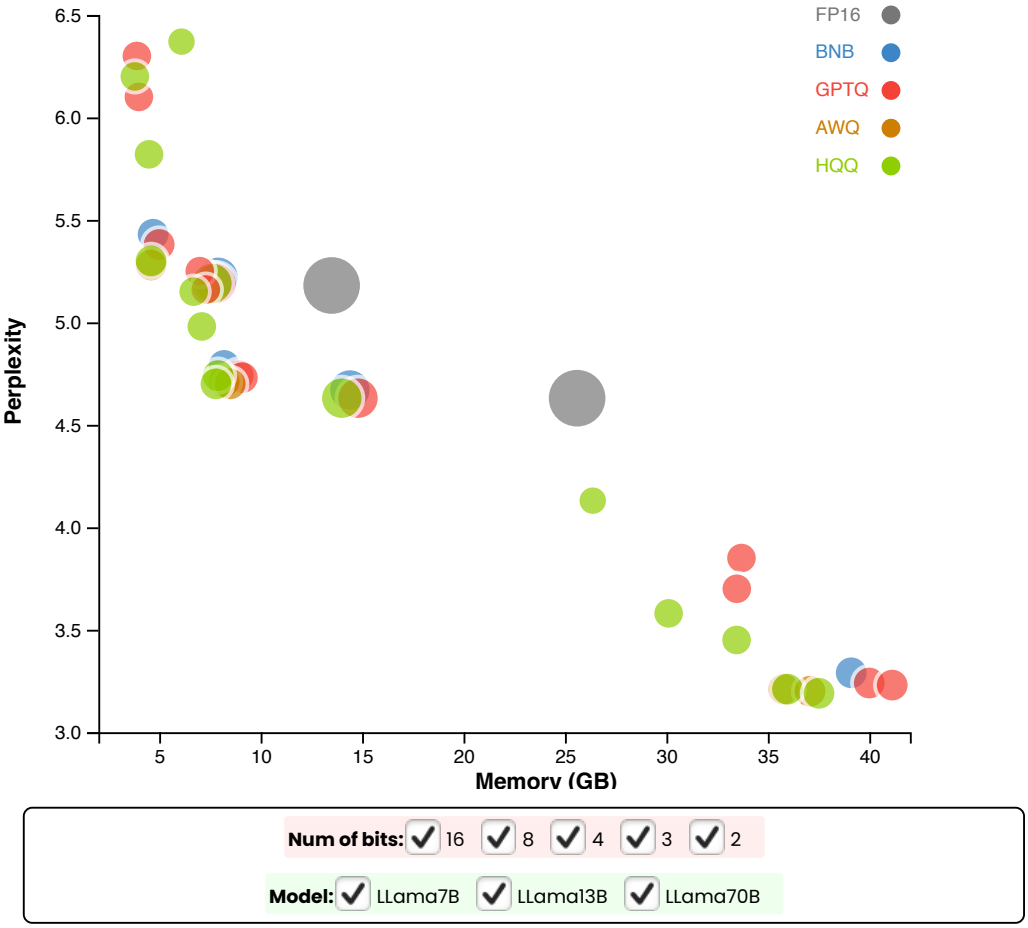
Method	nBits	Llama-2-7B		Llama-2-13B		Llama-2-70B	
		PPL ↓	MEM ↓	PPL ↓	MEM ↓	PPL ↓	MEM ↓
FP	16	5.18	13.5	**4.63**	25.6	OOM	OOM
BNB	8	5.22	7.9	4.67	14.4	3.17	**68.15**
GPTQ_g128	8	**5.19**	7.8	**4.63**	14.8	**3.12**	74.87
HQQ_g128	8	**5.19**	**7.6**	**4.63**	**14**	**3.12**	69.32
BNB_g64	4	5.43	4.7	4.79	8.2	3.29	39.11
GPTQ_g128	4	5.41	5	4.74	8.9	3.24	40
GPTQ_g64	4	5.38	5	4.73	9.1	3.23	41.13
AWQ_g128	4	5.32	**4.6**	4.71	8.2	3.21	**35.78**
AWQ_g64	4	**5.28**	**4.6**	**4.7**	8.5	3.2	37.08
HQQ_g128	4	5.35	**4.6**	4.74	**7.9**	3.21	35.97
HQQ_g64	4	5.3	**4.6**	**4.7**	8.2	**3.19**	37.52
GPTQ_g128	3	6.3	3.9	5.25	7	3.85	33.7
GPTQ_g64	3	6.1	4	5.16	7.3	3.7	33.47
HQQ_g128	3	6.2	**3.8**	5.15	**6.8**	3.58	**30.11**
HQQ_g64	3	**5.82**	4.5	**4.98**	7.4	**3.45**	33.46
GPTQ_g64	2	nan	**3.5**	13	6	9.44	24.5
HQQ_g32	2	15.61	**3.5**	7.63	**5.9**	4.82	**24.2**
HQQ_g16	2	**7.3**	4.1	**6.36**	6.9	**4.12**	30.27
HQQ_g16_s*	2	7.31	3.7	6.37	6.1	4.13	26.37

**: the scaling is also quantized to 8-bits with a group-size of 128.*

As illustrated in the table above, our method showcases strong performance without the need for calibration data. When applied to large models like the Llama-2-70B, 2-bit quantization via HQQ achieves a lower perplexity than the full-precision Llama-2-13B, all while requiring a comparable level of memory usage.

The interactive graph below summarizes the various data points into a

scatter plot. Hover or click on a bubble to display the details.



ViT Benchmark

We evaluate the effectiveness of our quantization method on vision models as well. More specifically, we quantize various OpenCLIP models from the Visual Transformers (ViT) family trained on the LAION dataset. Since Auto-GPTQ and Auto-AWQ calibration only works with text inputs, we can only evaluate against bitsandbytes by replacing all the linear layer inside the transformer blocks with their quantized versions.

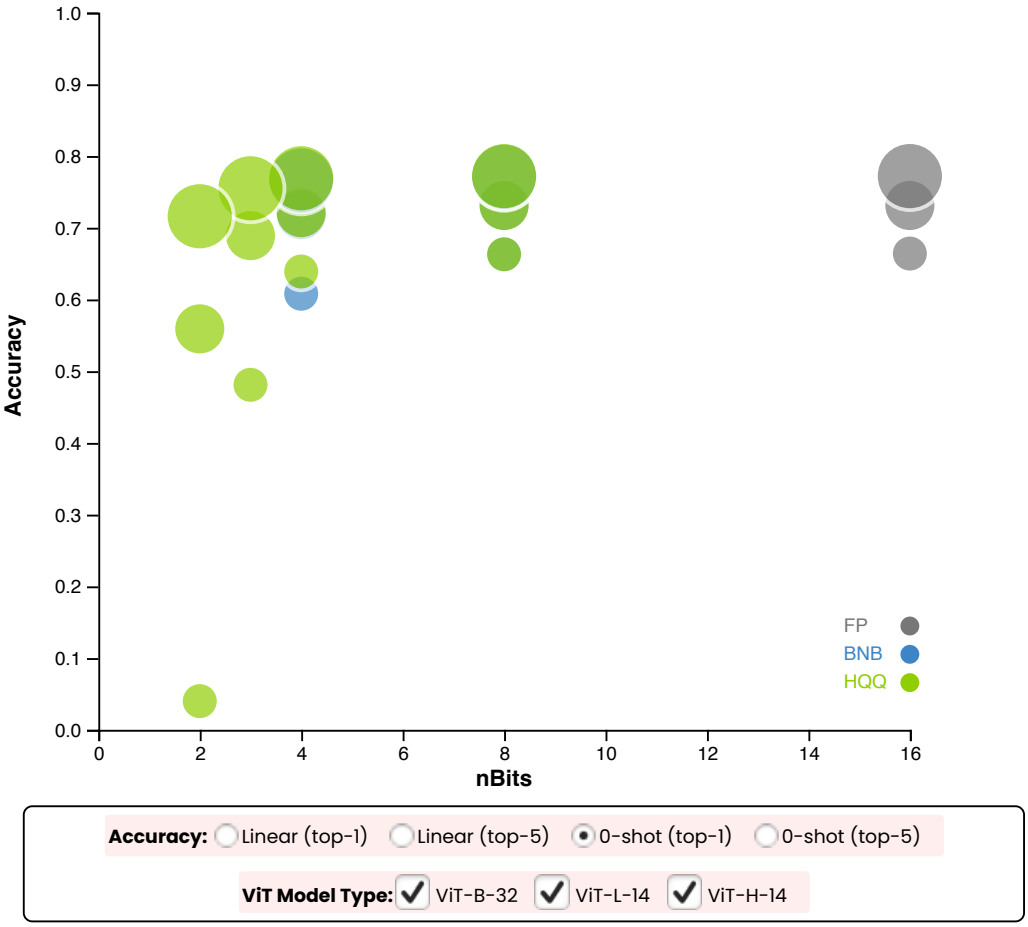
We conduct two sets of benchmarks and report the top-1 and top-5 accuracy on the ImageNet dataset. The first benchmark consists in measuring the zero-shot performance of the quantized models. We use the OpenAI prompts to generate zero-shot classifiers by averaging the text features over all the templates. This benchmark directly measures th quality of the quantized models since there's no training involved in the evaluation process. The second benchmark uses the quantized models as frozen backbone and trains a linear Softmax classifier on top of the features. This is referred to as Linear Probing and measures the quality of the quantized model as a frozen backbone. All results can be found in the table below:

	Linear	Linear (top-	0-shot	0-shot (top

Method	nBits	Model	(top-1)	5)	(top-1)	5)
FP	16	ViT-B-32	0.764	0.941	0.664	0.896
FP	16	ViT-L-14	0.82	0.964	0.731	0.93
FP	16	ViT-H-14	0.841	0.973	0.772	0.949
BNB	8	ViT-B-32	0.762	0.94	**0.663**	**0.896**
HQQ	8	ViT-B-32	**0.763**	**0.941**	**0.663**	**0.896**
BNB	8	ViT-L-14	**0.82**	**0.964**	**0.731**	**0.93**
HQQ	8	ViT-L-14	**0.82**	**0.964**	**0.731**	**0.93**
BNB	8	ViT-H-14	0.84	0.972	0.771	0.949
HQQ	8	ViT-H-14	**0.841**	**0.973**	**0.772**	**0.95**
BNB	4	ViT-B-32	0.733	0.925	0.608	0.859
HQQ	4	ViT-B-32	**0.75**	**0.933**	**0.639**	**0.881**
BNB	4	ViT-L-14	**0.815**	0.961	0.718	0.925
HQQ	4	ViT-L-14	**0.815**	**0.962**	**0.721**	**0.926**
BNB	4	ViT-H-14	0.837	0.971	0.766	0.947
HQQ	4	ViT-H-14	**0.839**	**0.973**	**0.769**	**0.948**
HQQ	3	ViT-B-32	0.664	0.881	0.481	0.753
HQQ	3	ViT-L-14	0.799	0.954	0.689	0.909
HQQ	3	ViT-H-14	0.831	0.969	0.755	0.943
HQQ	2	ViT-B-32	0.318	0.551	0.04	0.106
HQQ	2	ViT-L-14	0.731	0.917	0.559	0.815
HQQ	2	ViT-H-14	0.808	0.96	0.716	0.924

As we can see, our method produces high-quality quantization models despite not using any calibration data. It outperforms 4-bit bitsandbytes (BNB) by a large margin on zero-shot performance (**+3.1%** top-1 accuracy with ViT-B-32). For extreme low-bit quantization, our ViT-H-14 quantized to 3-bit outperforms the full-precision ViT-L-14 (**+2.4%** top-1 zero-shot accuracy), and the 2-bit version outperforms the ViT-32-B full-precision by a large margin (**+5.2%** top-1 zero-shot accuracy).

The plot below summarizes the various accuracy numbers into an interactive scatter plot. Hover or click on a bubble to display the details.



Conclusion

This article demonstrates that calibration-free quantization through our proposed Half-Quadratic Quantization (HQQ) method can achieve a quality competitive with popular data-dependent methods like GPTQ and AWQ. We have demonstrated the effectiveness of HQQ even for extreme low-bit quantization across different model sizes and applications. Moreover, by leveraging efficient optimization techniques such as Half-Quadratic splitting, our method cuts the quantization time to only a few minutes even for the biggest models available such as Llama-2-70B.

We provide the code to reproduce all the results presented in this article: https://github.com/mobiusml/hqq

Ready-to-use quantized models can be found on our Hugging Face 🤗 page: https://huggingface.co/mobiuslabsgmbh

---

Citation

```
@misc{badri2023hqq,
  title = {Half-Quadratic Quantization of Large Machine Learning Models},
  url = {https://mobiusml.github.io/hqq_blog/},
  author = {Hicham Badri and Appu Shaji},
  month = {November},
```

```
  year = {2023}
}
```

Please feel free to contact us..