

# Dynamic Quantization of LLMs via Reinforcement Learning (DynaQuant)

Oleg Roshka<sup>1</sup>

<sup>1</sup>Computer Science, Stanford



## Introduction

Large Language Models (LLMs) are powerful but resource-intensive. Quantization reduces model size and speeds up inference, but uniform quantization can be suboptimal. We propose a **dynamic**, per-layer quantization approach using Reinforcement Learning (RL) to find an optimal mixed-precision scheme during fine-tuning. This balances accuracy and memory footprint.

### Key points:

- **Motivation:** Exploit the fact that not all layers are equally sensitive to precision loss.
- **Approach:** Use **PPO** to pick layer-specific quantization types (e.g. nf4, fp4, int8, fp16), then do short fine-tuning to recover from possible performance drops.
- **Key Idea:** Adaptive fine-tuning after **each** layer’s quantization captures the dynamic interaction between layers.
- **Goal:** Learn a per-layer quantization policy for LLMs that optimizes accuracy and memory usage.

## Related Work

Our work builds upon several key areas:

- **LLM Quantization:** Techniques like GPTQ [5], QLoRA [2], bitsandbytes [1] and extreme LLM compression efforts [4, 6] have demonstrated the effectiveness of quantization for LLMs.
- **RL for Architecture Search:** Methods like NASNet [8] and APQ [7] use RL to optimize neural network architectures and quantization.
- **Mixed-Precision Quantization:** HAWQ [3] and other methods explore layer-wise bit-width selection.

Our approach differs by using RL to *dynamically* determine the mixed quantization scheme *during* fine-tuning, allowing the model to adapt to the specific quantization choices.

## Reinforcement Learning Environment

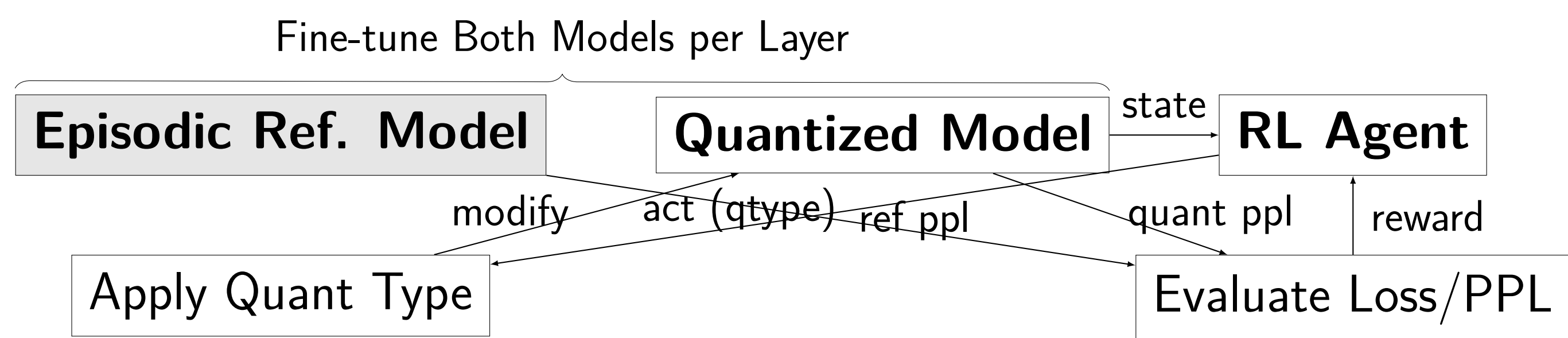


Figure 1. RL-based dynamic quantization loop. Agent picks a quant type for the current layer, then fine-tunes and compares performance (losses/perplexities).

- **State:** Layer statistics (weight mean/std, gradient norm, attention entropy), current layer index, previous layer’s quantization, and EMA of rewards.
- **Action:** Choice of quantization type: {'nf4', 'fp4', 'int8', 'fp16', 'bf16', 'fp32'}.
- **Reward:** Combination of perplexity difference (sigmoid-shaped), KL divergence, attention entropy difference, and memory savings.
- **Episode:** Quantizing all layers of the model sequentially.
- **Adaptive Fine-tuning:** Crucially, we fine-tune *both* the quantized model *and* an episodic reference model after *each* layer’s quantization, using *identical* training data, for a fixed number of steps. This captures the dynamic impact of quantization.

## Reward Function (Multi-Objective)

We define the per-layer reward as a weighted sum of performance, distribution alignment, attention preservation, and memory savings:

$$R = w_{\text{perf}} \sigma(\alpha (\text{PPL}_{\text{ref}} - \text{PPL}_{\text{quant}})) - w_{\text{KL}} \text{KL}(p_{\text{quant}} \| p_{\text{ref}}) + w_{\text{entropy}} (\text{Ent}_{\text{quant}} - \text{Ent}_{\text{ref}}) + w_{\text{memory}} \text{MemSave}.$$

### Interpretation:

- $\text{PPL}_{\text{ref}} - \text{PPL}_{\text{quant}}$ : difference in perplexities, run through a sigmoid ( $\sigma$ ) to keep reward in (0,1).
- $\text{KL}(p_{\text{quant}} \| p_{\text{ref}})$ : penalize large distribution shift.
- $\text{Ent}_{\text{quant}} - \text{Ent}_{\text{ref}}$ : preserve attention diversity.
- MemSave: bit savings relative to a 16-bit baseline, scaled by fraction of total params in that layer.

We also employ an exponential moving average (EMA) of the total reward to stabilize training updates in PPO.

## Implementation

- **Model:** GPT-2 (small, medium, large).
- **Quantization:** ‘bitsandbytes’ library and custom.
- **RL Algorithm:** Proximal Policy Optimization (PPO).
- **Fine-tuning Data:** CommonsenseQA.
- **Evaluation:** Perplexity, memory usage and inference speed. Datasets: BoolQ, PIQA
- **Infrastructure:** Training performed on H100 GPUs using the Modal platform. We gratefully acknowledge Modal for providing a free compute budget.

## Results

Table 1. Results on GPT-2 Small (PIQA). We compare perplexity (PPL), accuracy (Acc), and peak GPU memory (Mem). All deltas (%) are computed relative to FP32 as the baseline.

Method	PPL	$\Delta\text{PPL}(\%)$	Acc(%)	$\Delta\text{Acc}(\text{pp})$	Mem (MB)	$\Delta\text{Mem}(\%)$
Baseline (FP32)	25.31	—	60.72	—	574.22	—
Baseline (FP16)	25.31	+0.01%	60.72	+0.00	422.69	-26.38%
Uniform NF4	22.66	-10.47%	60.77	+0.05	300.95	-47.60%
DynaQuant (mixed)	21.44	-15.29%	61.53	+0.82	464.29	-19.14%

**Notes:** The reported DynaQuant configuration uses a mixed schema {fp16, int8, fp16, nf4, fp16, int8, fp16, int8, fp4, nf4, int8, fp4}. Perplexities are scaled by dividing raw values by 100. For instance, a raw PPL of 2531.21 is shown as 25.31. FP32 is the reference baseline. FP16 yields a memory reduction of about 26% with negligible changes in perplexity and accuracy. Uniform NF4 offers a larger perplexity drop (-10.47%) while cutting memory nearly in half. **DynaQuant** achieves the best overall trade-off: a -15.29% drop in perplexity, a +0.82 percentage point gain in accuracy, and a -19.14% memory reduction relative to FP32.

## Training Dynamics

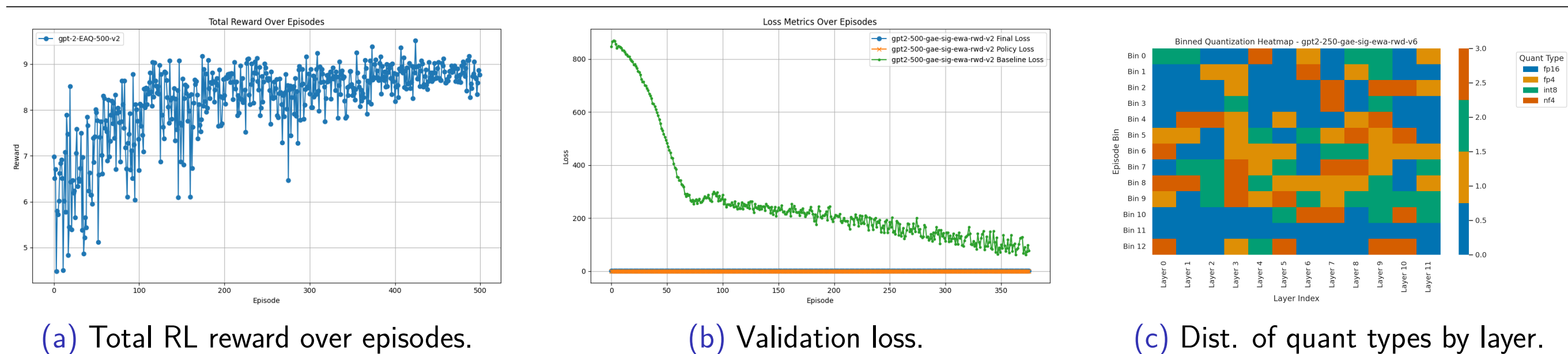


Figure 2. The RL reward grows over episodes, and the quantized model’s validation loss stays near reference. The distribution of quantization types across layers during training.

## Observations

- **Reward Dynamics:** Although the reward trends upward as the agent refines its per-layer decisions, it can be noisy. Sudden drops or oscillations may occur, especially when reward weights, batch sizes, or the number of fine-tuning steps change.
- **Policy Sensitivity:** By adjusting reward weights, one can steer the agent toward more aggressive compression (prioritizing memory savings) or more performance-preserving (accuracy-focused) quantization strategies.
- **Validation Loss Stability:** Even with layer-by-layer quantization, the quantized model’s validation loss stays reasonably close to that of the reference, thanks to short post-quantization fine-tuning.
- **Layer-Specific Choices:** Some layers often tolerate lower-precision (e.g. 4-bit), while more sensitive layers might require int8 or float16.
- **Compute Cost vs. Benefits:** This approach is compute-intensive. However, it can be justified by the overall memory/accuracy trade-off gains in scenarios where quality on user specific data sets is paramount.

## Conclusion and Future/In-progress Work

We presented an RL-based approach for dynamic, per-layer quantization of LLMs. Our method learns a policy that balances accuracy (perplexity) and memory usage, achieving a better trade-off than uniform quantization. Key features include:

- **Dynamic:** The policy adapts to each layer’s characteristics.
- **Adaptive Fine-tuning:** Captures the interplay between quantized layers.
- **Multi-Objective Reward:** Balances perplexity, KL divergence, entropy, and memory.

Future work includes:

- Scaling to larger models (e.g., DeepSeek-R1-Distill-Qwen-1.5B or phi-2).
- Hyperparameter tuning and exploration of different reward weightings.
- Adding a "skip" action to the policy, allowing it to choose not to quantize a layer.
- Exploring a policy network with a transformer architecture to leverage past decisions for better future choices. With enough training on diverse models/datasets, such a policy could infer a robust mixed-precision scheme from only a few fine-tuning steps for new scenarios.

## References

- [1] Tim Dettmers et al. “LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale”. In: *arXiv preprint arXiv:2208.07339* (Aug. 2022). arXiv: 2208.07339 [cs.LG]. URL: <https://arxiv.org/abs/2208.07339>.
- [2] Tim Dettmers et al. “QLoRA: Efficient Finetuning of Quantized LLMs”. In: *arXiv preprint arXiv:2305.14314* (2023).
- [3] Zhen Dong et al. “Hawq: Hessian aware quantization of neural networks with mixed-precision”. In: *arXiv preprint arXiv:1905.03696* (2019).
- [4] Vage Egiazarian et al. *Extreme Compression of Large Language Models via Additive Quantization*. 2024. arXiv: 2401.06118 [cs.LG].
- [5] Elias Frantar, Dan Alistarh, and Torsten Hoeftler. “GPTQ: Accurate post-training quantization for generative pre-trained transformers”. In: *arXiv preprint arXiv:2210.17323* (2022).
- [6] Vladimir Malinovskii et al. *PV-Tuning: Beyond Straight-Through Estimation for Extreme LLM Compression*. 2024. arXiv: 2405.14852 [cs.LG].
- [7] Ting-Wu Wang et al. “APQ: Joint search for network architecture, pruning and quantization policy”. In: *arXiv preprint arXiv:2003.02049* (2020).
- [8] Barret Zoph and Quoc V Le. “Neural architecture search with reinforcement learning”. In: *arXiv preprint arXiv:1611.01578* (2016).