Report on Quantization Strategy on Switchable Precision and Cyclic Precision Training

// Because the link to the squad in the pdf can't find anything: https://huggingface.co/Squad/datasets, therefore, I switched to a text generative dataset wikitext 103 for its ability on quantization

1. Questions: How did you determine the optimal quantization bit-width configurations? Have you gleaned any insights from your observations that could guide future work to further enhance performance?

Determining the **optimal configuration** requires clearly defining what "optimal" means in relation to specific **evaluation benchmarks** and performance metrics. For this report, we define optimal performance based on two primary criteria: perplexity scores and zero-shot accuracy. Several critical factors influence the optimization of model performance:

1.1 Choice of quantizer base on the current precision:

The choice of quantization method significantly impacts model performance and varies with the target bit width:

- **For 4-bit or lower precision**: MinMax quantization delivers the best results
- **For 5-bit or higher precision**: Advanced methods like log quantization and tanh quantization provide superior performance

My experiments reveal that log quantization reduces 8-bit model degradation to just 2%, compared to 15% degradation with MinMax quantization. This improvement likely occurs because GPT-2 Small has relatively limited information content and less pronounced outlier effects, allowing log quantization to outperform the MinMax clipping function in case of higher precisions. Here's my experiments results on different approaches of changing the quantization strategies.

	4 bit with Minmax Quantizer	4 bit with Log Quantizer	5 bit with Minmax Quantizer	5 bit with Log Quantizer
perplexity on WikiText2	53.2	56.5	40.5	35.1
perplexity on WikiText103	44.5	48.4	33.6	29.3
BoolQ	57.0	59.0	52.6	57.4
HellaSwag	32.0	30.2	32.8	33.4
WinoGrande	54.6	48.2	51.0	56.4
Zeroshot average	47.9	45.8	45.5	49.1

Important Notes:

- C4 perplexity metrics were excluded from this analysis since the models were fine-tuned specifically on the WikiText dataset, making C4 comparisons less meaningful

1.2 Training precision distribution

1.2.1 On accumulative temperature scaled soft max distillation method

During the training process of switchable precision model, the self distillation formula is as follows:

$$L_q = \alpha_1 D_{KL}(SG(p_r)||p_{qq}) + \alpha_2 \sum_{i\ in\ feature} ||f_{ir} - f_{iq}||_2^2$$

Where p_r is the unquantized output distribution, p_{qq} is the quantized output distribution.

	4 bit model	5 bit model
perplexity on WikiText2	56.5	35.1
perplexity on WikiText103	48.4	29.3
BoolQ	59.0	57.4

	4 bit model	5 bit model
HellaSwag	30.2	33.4
WinoGrande	48.2	56.4
Zeroshot average	45.8	49.1

All of them are using the log quantizer.

- Activation bits changes accordingly to the current bit width during the training.

1.2.2 On non-accumulative temperature scaled soft max distillation method (used by the switchable precision paper)

$$L_q = \alpha_1 D_{KL}(SG(p_r)||p_{qq}) + \alpha_2||f_r - f_q||_2^2$$

I also investigated a stochastic distillation variant where feature layers are randomly selected from the hidden states during training. Surprisingly, despite receiving guidance from only a subset of attention layers rather than all layers, the model's accuracy improved rather than deteriorated, while the perplexity decreases as expected:

	4 bit self distillation method with teaching from all the hidden states	4 bit self distillation method with teaching from a random hidden state
perplexity on WikiText2	56.5	67.6
perplexity on WikiText103	48.4	dc ..68.8
BoolQ	59.0	60.0
HellaSwag	30.2	31.8
WinoGrande	48.2	49.8
Zeroshot average	45.8	47.2

These results raise an important question: which specific information within the hidden layers contributes most significantly to accuracy improvements? Understanding this could help optimize the distillation process further.

2. Interesting insight on optimizing the model performance

2.1 16 bits is extremely similar to the 32 bits

Calibration logic need to be recalibrated after series of grad accumulation. With no calibration, **16 bit has the same perplexity(0.8% degradation) as 32 bit**, the degradation from not quantization is minimal.

2.2 Merge then quant or the other way around?

For lora, would it be better performing if we apply the AB and then do the quantization or do the quantization after AB? Experiment from a blogger The inference time is so much longer, which is due to the huge tensor for dequantization. Meanwhile, I would think that it would take longer to train since the big tensor is harder to calibrate.

	Merge after Quant: Q(W) + Q(A @ B)	**Merge after Quant: Q(W) + Q(A)@ Q(B)**
Inference Speed	Faster	Slower
Accuracy	Same	Same
Calibration speed	Faster (by a bit)	Slower

2.3 Per channel , per feature or per token ?

My hypothesis is that quantization calibration should align with the **semantic boundaries** of tensor dimensions. Different layers encode information differently, and quantization should preserve these natural groupings. (But from the testing experiences, it seems that it actually not matter as much tho.) For Embedded layers, per channel calibration should be optimal, and for other layers like projection layers and those attention layers, per token calibration should be more optimal.

2.4 Unexpected minimal impact during debugging:

- It just incredible that by changing the FP32 weight bit quantization for 32 bits to the matching precision reduce the perplexity from about 120 to 30 (approximately, since I was debugging the code).
- During the evaluation with input activation quantization, using pretrained input scale and zero point or recalibrate the input actually makes minimal perplexity and accuracy differences. Input activation distributions are stable across different datasets - pretrained calibration generalizes well.

2.5 The root cause for the difference CPT and progressive precision training

The Reason why Progressive training works, might be similar to the distillation effect: we train with higher bit-width to understand the structure better and then pass the more precise understanding to the next lower bit-width. But what the reason why CPT is better than progressive training in lower bit quantization is that

- **Exploration matters more** than exploitation at low precision
- Cycling provides **escape mechanisms** from bad local minima

4.Does this phenomenon align with the observations in CPT (ICLR'21)? If not, what could be the potential reasons?

- The CPT model does help with reducing the cost. Since traditional method required distillation from teacher bit, these model will store and process at least two times the number of lora layer of a single model. But for CPT model, we would only need to store the lora layer of a model once and change precision during the flight. :

Num of params	Cyclic Precision Training	Switchable Precision of
Total parameters	166,212,880	257,097,984
Frozen parameters	162,998,784 (98.1%)	124,977,408 (48.6%)
Trainable (LoRA) parameters	3,214,096 (1.9%)	132,120,576 (51.4%)

- The CPT does helps with training to gain a higher accuracy, even though CPT's perplexity greatly degraded.

	5 bit model trained with CPT	5 bit model trained with Switchable precision
BoolQ	63.0	57.4
HellaSwag	32.4	33.4
WinoGrande	52.4	56.4
Average	49.3	49.1

5.[Step 6] Does this phenomenon align with the observations in Double-Win Quant (ICML'21)? If not, what could be the potential reasons?

It does show that

- Poor Adversarial transferability
- Random Precision Inference does works to increase the robustness

Model Metrics

Metric	Value
Clean Accuracy	27.66%
Clean Perplexity	74.01
Clean Loss	4.304

Attack Effectiveness

Attack Type	Success Rate	Original Acc.	Adversarial Acc.	Accuracy Drop	Perturbation
TextFooler	63.33%	26.95%	20.53%	-6.41%	11.86%
BERT-Attack	83.33%	26.95%	16.77%	-10.18%	15.83%

RPI Implementation:

Switching Probability	TextFooler Defense	BERT-Attack Defense	Average
0% (Baseline)	0.0%	0.0%	0.0%
30%	+36.8%	+48.0%	+42.4%
50%	+47.4%	+60.0%	+53.7%
70%	+47.4%	+60.0%	+53.7%

6. *Based on your explorations of switchable and dynamic quantization, could you propose some promising research directions or questions for further integrating them with LLMs? **

6.1 Intentionally use lower-precision layer normalization than the current training bit-width to inject beneficial exploration noise, or maybe employ a cyclic schedule for precision switching.

Would it be possible to use different precision batch norm for the current back propagating bit width? IN the CPT paper, the reason that lower bit width training actually improve on the accuracy is because *a lower precision that leads to a short-term poor accuracy might actually help the DNN exploration during training thanks to its associated larger quantization noise*. Then would it be possible to use a lower-than-current bit layer norm base on the CPT training (i.e. 4 bit layer norm for WA-6 / GE-8 and WA-7 / GE-8)

6.2 Since we're not deploying yet (just training with fake quantization), why limit ourselves to integer bit-widths? Use continuous, real-valued bit-widths (e.g., 4.7 bits, 5.3 bits). Would it make CPT more powerful? Would it helps us to understand what happen on the boundary from 4 to 5 bits?

For the cyclic training method, since the quantization is essentially fake, and we are not talking about deployment right now, so can't we use real number quantization, instead of integer? What might a more continuous training schedule looks like?Precision-LR Coupling:** Is there an optimal mathematical relationship? **Robustness Boundaries:** How does training near precision boundaries affect model resilience?

6.3 Equavalence of Learning Rate Schedule and the Precision Schedule

Learning rate schedule is adjusting the step for each optimization, and think about the way CPT works, switching precisions between epochs. They looks quite different at the first glance, but the thing is that they are both changing optimizing steps. For LR schedule, it might be changing the step in a linear way, but precision backward basically integrate part of the sematic meaning (per Channel calibration) into the optimizing stepping. Would there's be a map from the precision switching schedule to the LR schedule, if so, can we literally replace the LR schedule and replace this with the CPT with better sematic interpretation?

Could it be something like

- 4-bit training ≈ high_lr * gradient + large_noise
- 8-bit training ≈ medium_lr * gradient + medium_noise
- 16-bit training ≈ low_lr * gradient + small_noise