AWQ and TinyChat: Efficient LLMs on the Edge

Ji Lin
jl@mit.edu
https://linji.me

Haotian Tang
k@mit.edu
http://kentang.net
Revisit: SmoothQuant (W8A8)

Accurate and efficient quantization of various LLMs

- SmoothQuant well maintains the accuracy without fine-tuning.
- SmoothQuant can both accelerate inference and halve the memory footprint.
W4A16 for Single Query Serving

W8A8 cannot address low computational intensity of decoding

- W8A8 quantization is good for batch serving (e.g., batch size 128)
- But single-query LLM inference (e.g., local) is still highly memory-bounded
- We need **low-bit weight-only** quantization (e.g., W4A16) for this setting

![Diagram showing TFLOPS vs. Compute intensity](image)

- A100 GPU
- LLaMA-65B decoding
AWQ for Low-bit Weight-only Quantization

Targeting group-wise W3/W4 quantization

- Weight-only quantization reduces the memory requirement, and accelerates token generation by alleviating the memory bottleneck.
- Group-wise/block-wise quantization (e.g., 64/128/256) offers a better accuracy-model size trade-off.
- But there is still a performance gap with round-to-nearest (RTN) quantization (INT3-g128)

AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration (Lin et al., 2023)
AWQ for Low-bit Weight-only Quantization

Observation: Weights are not equally important; 1% salient weights

<table>
<thead>
<tr>
<th>$W_{FP16}$</th>
<th>$Q(W)_{MixPrec}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1.2 -0.2 -2.4 -3.4</td>
<td>+1 +0 -2 -3</td>
</tr>
<tr>
<td>-2.5 -3.5 +1.9 +1.4</td>
<td>-2.5 -3.5 +1.9 +1.4</td>
</tr>
<tr>
<td>-0.9 +1.6 -2.5 -1.9</td>
<td>-1 +2 -3 -2</td>
</tr>
<tr>
<td>-3.5 +1.5 +0.5 -0.1</td>
<td>-4 +2 +1 +0</td>
</tr>
<tr>
<td>+1.8 -1.6 -3.2 -3.4</td>
<td>+2 -2 -3 -3</td>
</tr>
<tr>
<td>+2.4 -3.5 -2.8 -3.9</td>
<td>+2 -4 -3 -4</td>
</tr>
<tr>
<td>+0.1 -3.8 +2.4 +3.4</td>
<td>+0 -4 +2 +3</td>
</tr>
<tr>
<td>+0.9 +3.3 -1.9 -2.3</td>
<td>+1 +3 -2 -2</td>
</tr>
</tbody>
</table>

- We find that weights are not equally important, keeping only 1% of salient weight channels in FP16 can greatly improve perplexity
- But how do we select salient channels? Should we select based on weight magnitude?

Learning both Weights and Connections for Efficient Neural Networks (Han et al., 2015)
AWQ for Low-bit Weight-only Quantization

Salient weights are determined by activation distribution, not weight

- We find that weights are not equally important, keeping only 1% of salient weight channels in FP16 can greatly improve perplexity.
- But how do we select salient channels? Should we select based on weight magnitude?
- No! We should look for activation distribution, but not weight!

AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration (Lin et al., 2023)
AWQ for Low-bit Weight-only Quantization

Salient weights are determined by activation distribution, not weight

- **Pro**: improve quantized performance with a negligible overhead (only 1%)
- **Con**: bad hardware efficiency due to mixed-precision weights
  - A recently work SpQR only manages to get 15% speed-up
**AWQ for Low-bit Weight-only Quantization**

Protecting salient weights by scaling (no mixed prec.)

\[
W \rightarrow Q(W \cdot s)(s^{-1} \cdot X)
\]

- Multiplying the salient channels with \( s > 1 \) reduces its quantization error
- Why?

AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration (Lin et al., 2023)
AWQ for Low-bit Weight-only Quantization

Protecting salient weights by scaling (no mixed precision)

- Consider a linear layer channel $y = wx$ (from $Wx$). We care about the quantization error from $Q(w)x$

- $Q(w) = \Delta \cdot \text{Round}(w/\Delta)$, $\Delta = \frac{\text{max}(|w|)}{2^{N-1}}$

- The scaled version is $Q(w \cdot s)(x/s) = \Delta \cdot \text{Round}(sw/\Delta) \cdot x \cdot \frac{1}{s}$

- We find that the error from Round() is always $\sim 0.25$ (average from 0-0.5)

- The maximum value in a group “usually” does not change if we just scale up a channel -> $\Delta$ not changed

- With $s > 1$, the error is scaled down.
**AWQ for Low-bit Weight-only Quantization**

Protecting salient weights by scaling (no mixed prec.)

$$W$$

<table>
<thead>
<tr>
<th>+1.2</th>
<th>-0.2</th>
<th>-2.4</th>
<th>-3.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.5</td>
<td>-3.5</td>
<td>+1.9</td>
<td>+1.4</td>
</tr>
<tr>
<td>-0.9</td>
<td>+1.6</td>
<td>-2.5</td>
<td>-1.9</td>
</tr>
<tr>
<td>-3.5</td>
<td>+1.5</td>
<td>+0.5</td>
<td>-0.1</td>
</tr>
<tr>
<td>+1.8</td>
<td>-1.6</td>
<td>-3.2</td>
<td>-3.4</td>
</tr>
<tr>
<td>+2.4</td>
<td>-3.5</td>
<td>-2.8</td>
<td>-3.9</td>
</tr>
<tr>
<td>+0.1</td>
<td>-3.8</td>
<td>+2.4</td>
<td>+3.4</td>
</tr>
<tr>
<td>+0.9</td>
<td>+3.3</td>
<td>-1.9</td>
<td>-2.3</td>
</tr>
</tbody>
</table>

$$Q(W \cdot s)(s^{-1} \cdot X)$$

$$WX \rightarrow Q(W \cdot s)(s^{-1} \cdot X)$$

- Multiplying the salient channels with $$s > 1$$ reduces its quantization error
- Take a data-driven approach with a fast **grid search**

$$L(s) = \|Q(W \cdot s)(s^{-1} \cdot X) - WX\|$$

$$s = S X^\alpha \quad \text{Activation-awareness is important, but not weight-awareness}$$

**OPT-6.7B Wiki-2 PPL↓**

- 1.5x: 14.49
- 2x: 14.07
- 4x: 14.42
- search: 13.18

AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration (Lin et al., 2023)
### AWQ for Low-bit Weight-only Quantization

Better PPL under low-bit weight-only quantization

<table>
<thead>
<tr>
<th>PPL</th>
<th>Llama-2</th>
<th>LLaMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7B</td>
<td>13B</td>
</tr>
<tr>
<td>FP16</td>
<td>-</td>
<td>5.47</td>
</tr>
<tr>
<td>INT3 g128</td>
<td>RTN</td>
<td>6.66</td>
</tr>
<tr>
<td></td>
<td>GPTQ</td>
<td>6.43</td>
</tr>
<tr>
<td></td>
<td>GPTQ-R</td>
<td>6.42</td>
</tr>
<tr>
<td></td>
<td>AWQ</td>
<td><strong>6.24</strong></td>
</tr>
<tr>
<td>INT4 g128</td>
<td>RTN</td>
<td>5.73</td>
</tr>
<tr>
<td></td>
<td>GPTQ</td>
<td>5.69</td>
</tr>
<tr>
<td></td>
<td>GPTQ-R</td>
<td>5.63</td>
</tr>
<tr>
<td></td>
<td>AWQ</td>
<td><strong>5.60</strong></td>
</tr>
</tbody>
</table>

AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration (Lin et al., 2023)
## AWQ for Low-bit Weight-only Quantization

Also works for multi-modal LLMs (OpenFlamingo-9B, captioning)

<table>
<thead>
<tr>
<th>COCO (CIDEr ↑)</th>
<th>0-shot</th>
<th>4-shot</th>
<th>8-shot</th>
<th>16-shot</th>
<th>32-shot</th>
<th>Δ(32-shot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP16</td>
<td>-</td>
<td>63.73</td>
<td>72.18</td>
<td>76.95</td>
<td>79.74</td>
<td>81.70</td>
</tr>
<tr>
<td>INT4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g128 RTN</td>
<td>60.24</td>
<td>68.07</td>
<td>72.46</td>
<td>74.09</td>
<td>77.13</td>
<td>-4.57</td>
</tr>
<tr>
<td>GPTQ</td>
<td>59.72</td>
<td>67.68</td>
<td>72.53</td>
<td>74.98</td>
<td>74.98</td>
<td>-6.72</td>
</tr>
<tr>
<td>AWQ</td>
<td>62.57</td>
<td>71.02</td>
<td>74.75</td>
<td>78.23</td>
<td>80.53</td>
<td>-1.17</td>
</tr>
<tr>
<td>INT3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g128 RTN</td>
<td>46.07</td>
<td>55.13</td>
<td>60.46</td>
<td>63.21</td>
<td>64.79</td>
<td>-16.91</td>
</tr>
<tr>
<td>GPTQ</td>
<td>29.84</td>
<td>50.77</td>
<td>56.55</td>
<td>60.54</td>
<td>64.77</td>
<td>-16.93</td>
</tr>
<tr>
<td>AWQ</td>
<td>56.33</td>
<td>64.73</td>
<td>68.79</td>
<td>72.86</td>
<td>74.47</td>
<td>-7.23</td>
</tr>
</tbody>
</table>

### Example Descriptions

- **W4-RTN**: A model airplane flying in the sky.
- **W4-AWQ**: Two toy airplanes sit on a grass field.
- **W4-RTN**: A man is holding a baby elephant in his arms.
- **W4-AWQ**: A man and his daughter pose with an elephant.
- **W4-RTN**: A man and a dog walking past some bushes.
- **W4-AWQ**: Two dogs are walking on the street.

AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration (Lin et al., 2023)
Question: Can you explain this meme in detail?

INT4-g128 RTN: The meme in the image features a plate of food, likely fried chicken, with a unique twist. There are small pictures of the Earth and other planets placed on top of the food, creating an interesting and amusing visual. Alongside the plate, a caption reads, "Sometimes I look at pictures of the Earth from space and I marvel at how beautiful it all is." This combination of food and planet images adds a touch of humor and curiosity, making the viewer wonder about the context and intent behind the meme.

INT4-g128 AWQ: The meme in the image is a lighthearted and humorous take on the concept of looking at pictures of the Earth from space. A plate of fried food, specifically chicken nuggets, is presented with the caption, "Sometimes I look at pictures of the Earth from space and I marvel at how beautiful it all is." The playful combination of a food item and an Earth-related context creates a fun and entertaining visual.
AWQ for Low-bit Weight-only Quantization

Also works for multi-modal LLMs (LLaVA, visual reasoning)

**Question**: Do you know who drew this painting?

**INT4-g128 RTN**: The painting depicts a woman, who some believe to be a representation of the famous artist, Mona Lisa. However, it is not certain that the woman in the painting is indeed Mona Lisa. The artwork is a popular subject of artistic analysis and admiration, often considered a symbol of feminine beauty and enigma. The portrait is characterized by its intricate details, captivating the viewer with the woman's expressive gaze and the subtle, enigmatic smile that has become synonymous with the Mona Lisa.

**INT4-g128 AWQ**: The painting depicts a woman, who is believed to be a portrait of Mona Lisa, the famous artwork by the artist Leonardo da Vinci. Although the painting is widely recognized as a representation of Mona Lisa, it is not possible to confirm the artist who drew this particular painting without additional information. Leonardo da Vinci was a renowned artist known for his exceptional skill in painting detailed, realistic portraits, and the Mona Lisa is one of his most famous works.
Results
Quantization of instruction-tuned models

- Comparing quantized Vicuna with FP16 counterparts
- *W4 almost preserves performance*
## Results

Quantization for complex generations (code & math)

<table>
<thead>
<tr>
<th></th>
<th>MBPP (7B)</th>
<th></th>
<th></th>
<th>GSM-8K</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pass@1</td>
<td>pass@10</td>
<td></td>
<td>pass@1</td>
<td>pass@10</td>
<td>pass@1</td>
<td>pass@10</td>
</tr>
<tr>
<td>FP16</td>
<td>38.53</td>
<td>49.77</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTN</td>
<td>37.51</td>
<td>48.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPTQ</td>
<td>31.97</td>
<td>44.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWQ</td>
<td><strong>40.64</strong></td>
<td><strong>49.25</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **MBPP**: a Python coding dataset
- **GSM-8K**: a math reasoning dataset (requires multi-step reasoning)
- AWQ preserves the accuracy under **W4-g128** quantization
SmoothQuant and AWQ are widely used:

- FasterTransformer: [FasterTransformer/blob/main/docs/gpt_guide.md](https://github.com/NVIDIA/FasterTransformer/blob/main/docs/gpt_guide.md)
- TRT-LLM: [FasterTransformer/blob/main/docs/TensorRT-LLM#key-features](https://github.com/NVIDIA/FasterTransformer/blob/main/docs/TensorRT-LLM#key-features)
- Q8-Chat: [FasterTransformer/blob/main/docs/TensorRT-LLM#key-features](https://github.com/NVIDIA/FasterTransformer/blob/main/docs/TensorRT-LLM#key-features)
- text-generation-inference: [https://github.com/NVIDIA/TensorRT-LLM#key-features](https://github.com/NVIDIA/TensorRT-LLM#key-features)
- Q8-Chat: [https://github.com/NVIDIA/TensorRT-LLM#key-features](https://github.com/NVIDIA/TensorRT-LLM#key-features)
- TRT-LLM: [https://github.com/NVIDIA/FasterTransformer/blob/main/docs/TensorRT-LLM#key-features](https://github.com/NVIDIA/FasterTransformer/blob/main/docs/TensorRT-LLM#key-features)
- text-generation-inference: [https://github.com/NVIDIA/TensorRT-LLM#key-features](https://github.com/NVIDIA/TensorRT-LLM#key-features)
- TRT-LLM: [https://github.com/NVIDIA/FasterTransformer/blob/main/docs/TensorRT-LLM#key-features](https://github.com/NVIDIA/FasterTransformer/blob/main/docs/TensorRT-LLM#key-features)
- Q8-Chat: [https://github.com/NVIDIA/TensorRT-LLM#key-features](https://github.com/NVIDIA/TensorRT-LLM#key-features)
TinyChat: A Lightweight Serving Infra

Pythonic, lightweight, efficient

- We need a framework to serve the quantized model to achieve low latency (AWQ only for Linears)
  - HuggingFace: easy to use, but slow
  - FasterTransformer: high efficiency, but harder to use
TinyChat: A Lightweight Serving Infra

Pythonic, lightweight, efficient

- We need a framework to serve the quantized model to achieve low latency
  - HuggingFace: easy to use, but slow
  - FasterTransformer: high efficiency, but harder to use
- TinyChat goals: efficient, lightweight, Python-native (composable with other stacks like vLLM)
TinyChat: A Lightweight Serving Infra

Analyze the latency overhead under FP16

- Measurement based on LLaMA-7B on RTX4090

<table>
<thead>
<tr>
<th>Technique</th>
<th>Tok/sec</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huggingface impl.</td>
<td>49.0</td>
<td>76%</td>
</tr>
<tr>
<td>FasterTransformer impl.</td>
<td>64.1</td>
<td>100%</td>
</tr>
</tbody>
</table>
TinyChat: A Lightweight Serving Infra

Analyze the latency overhead under FP16

- Measurement based on LLaMA-7B on RTX4090
- Some overheads can be easily removed! 95% of FT performance in Python

<table>
<thead>
<tr>
<th>Technique</th>
<th>Tok/sec</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huggingface impl.</td>
<td>49.0</td>
<td>76%</td>
</tr>
<tr>
<td>Preallocate KV cache</td>
<td>54.1</td>
<td>84%</td>
</tr>
<tr>
<td>FT LayerNorm kernel</td>
<td>57.5</td>
<td>90%</td>
</tr>
<tr>
<td>FlashAttention</td>
<td>57.5</td>
<td>90%</td>
</tr>
<tr>
<td>Merge QKV projections</td>
<td>59.2</td>
<td>92%</td>
</tr>
<tr>
<td>Fuse rotary embedding</td>
<td>61.0</td>
<td>95%</td>
</tr>
<tr>
<td>FasterTransformer impl.</td>
<td>64.1</td>
<td>100%</td>
</tr>
</tbody>
</table>
TinyChat: A Lightweight Serving Infra

State-of-the-art W4 inference speed

• Now we plugin the AWQ to quantize the weights into 4-bit (further 3x improvement)
• We can outperform the state-of-the-art MLC-LLM (TVM compilation-based) with our **Pythonic** solution
• 50% faster on AGX Orin

<table>
<thead>
<tr>
<th>LLaMA-7B</th>
<th>RTX 4090 Tok/sec</th>
<th>AGX Orin Tok/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>llama.cpp</td>
<td>141</td>
<td>22.5</td>
</tr>
<tr>
<td>Exllama</td>
<td>153</td>
<td>15.9</td>
</tr>
<tr>
<td>MLC-LLM</td>
<td>191</td>
<td>-</td>
</tr>
<tr>
<td>TinyChat</td>
<td><strong>195</strong></td>
<td><strong>30.2</strong></td>
</tr>
</tbody>
</table>
TinyChat: A Lightweight Serving Infra

Supporting a wide range of models on NVIDIA Jetson Orin

- TinyChat achieves up to **1.5x** faster runtime for Meta’s Llama models compared with systems specialized for this model.
- Compared with the only competitor that can support a diverse range of models, TinyChat is up to **7x** faster.
- Remarkably, TinyChat’s front end is **fully PyTorch-based**.

Latency comparison on Jetson Orin (64G) mobile GPU

- AutoGPTQ
- llama.cpp
- exllama
- TinyChat
TinyChat: A Lightweight Serving Infra

Demo on AGX Orin (edge LLM inference)

- Orin Nano has 200GB/s memory bandwidth; even more memory-bounded
- Model size: 7B. ~30 token/s generation
• A LightWeight Chatbot for LLMs on the edge
  - Deploying LLM on the edge is useful: running copilot services (code completion, office, game chat) locally on laptops, cars, robots, and more. These devices are resource-constrained, low-power and sometimes do not have access to the Internet.
  - Data privacy is important. Users do not want to share personal data with large companies.
- TinyChatEngine implements the compressed AWQ 4bit model, built from C/C++ from scratch, easy to install and migrate to edge platforms
- Enables on-device LLM
TinyChat brings about 3.3x speedup to LLaMA-2 on 4090

LLaMA-2-7B (FP16): 50 tokens / s
Baseline: fp16 weight, fp16 activation

LLaMA-2-7B (W4A16, AWQ): 166 tokens / s
AWQ: int4 weight, fp16 activation
TinyChat flexibly supports different LLM architectures

- MPT-7B: 31 tokens / s
- Falcon-7B: 22 tokens / s
- Vicuna-7B: 33 tokens / s

https://github.com/mit-han-lab/llm-awq
TinyChat delivers 30 tokens / second performance for LLaMA2

LLaMA-2-7B (W4A16, AWQ): 30 tokens / s

https://github.com/mit-han-lab/llm-awq
Thanks for Listening!