Getting *free* Bits Back from Rotational Symmetries in LLMs

Jiajun He, Gergely Flamich, José Miguel Hernández-Lobato
University of Cambridge

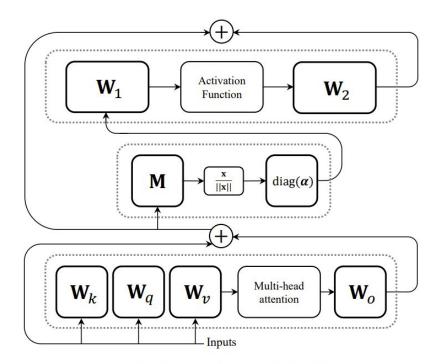
After pruning LLMs,

we can further save 3-5% additional bits for free

in storage and transmission

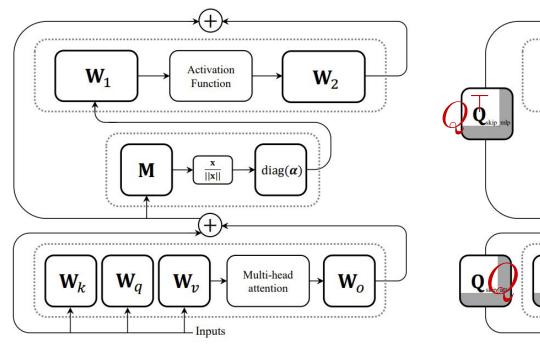
SliceGPT and bits-back coding

SliceGPT

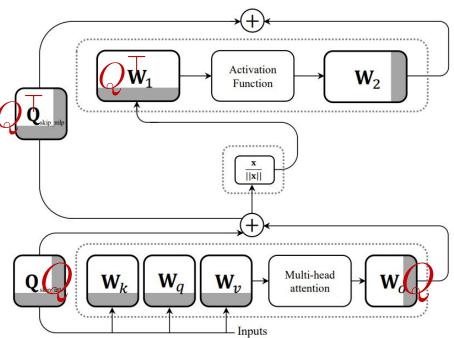


(a) A standard transformer block.

SliceGPT introduces rotational symmetries:

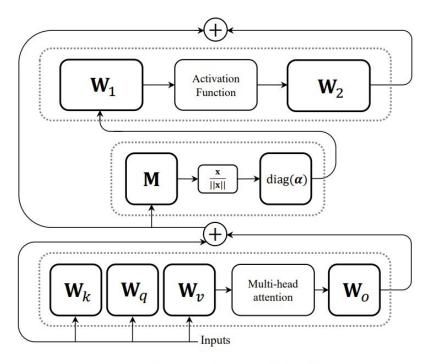


(a) A standard transformer block.

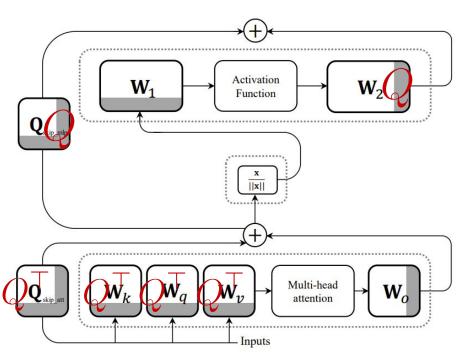


(b) A transformer block with SliceGPT.

SliceGPT introduces rotational symmetries:



(a) A standard transformer block.



(b) A transformer block with SliceGPT.

SliceGPT introduces rotational symmetries:

we can write each transformer block as:

$$f(x, W) = f(x, QW)$$

source coding: + x = _____

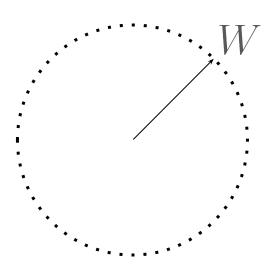
"bits-back way": _____ + x = ____ + ____

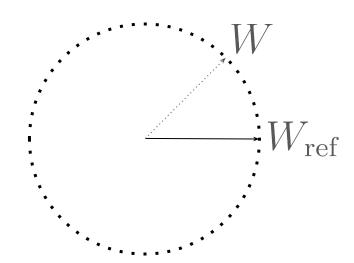
Getting free bits back from rotational symmetries

f(x, W) = f(x, QW)

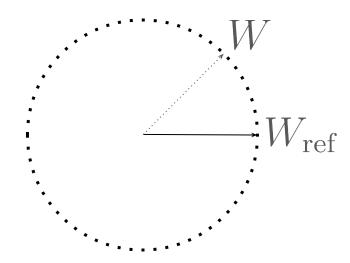
encode:

encode: start with a weight matrix and some initial bits

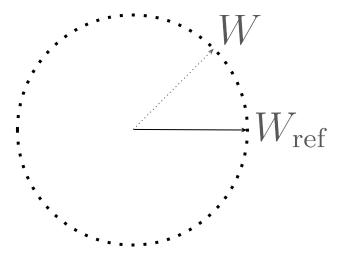




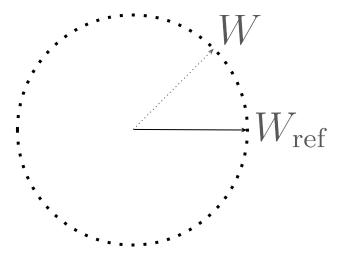
$$\operatorname{svd}(W) = U \sum_{\bar{W}_{\text{ref}}} V_{\text{ref}}$$



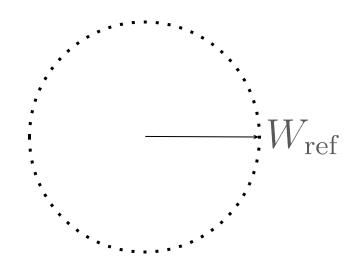
$$\mathrm{svd}(W) = U \ \overline{\Sigma} V^\top$$
 i.e., define $W_{\mathrm{ref}} W^\top_{\mathrm{ref}}$ to be diagnoal



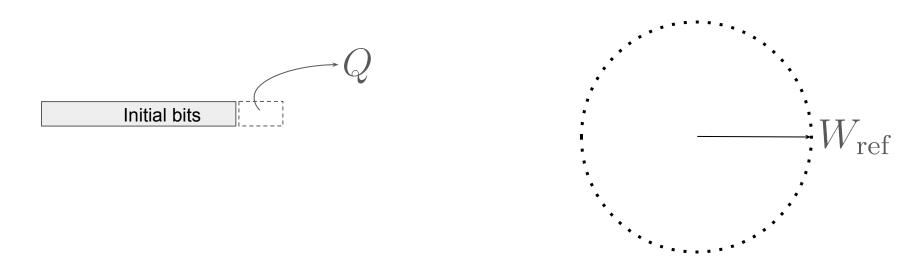
$$\mathrm{svd}(W) = U \ \overline{\Sigma} V^\top$$
 i.e., define $W_{\mathrm{ref}} W^\top_{\mathrm{ref}}$ to be diagnoal



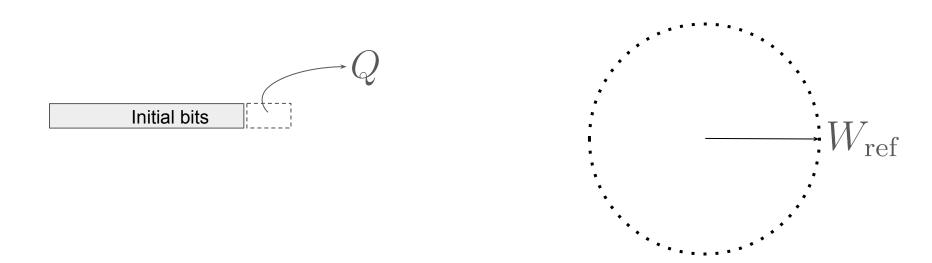
encode step 2: decode a rotation from the bitstream



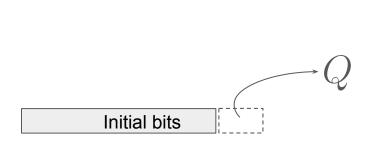
encode step 2: decode a rotation from the bitstream



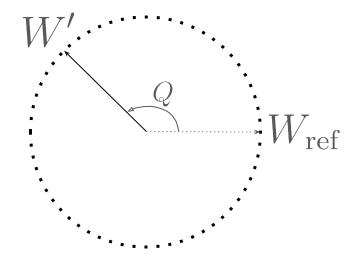
encode step 3: rotate weight by the decoded rotation matrix



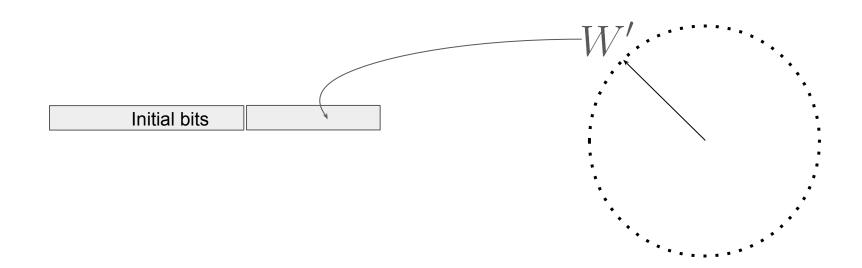
encode step 3: rotate weight by the decoded rotation matrix



$$W' = QW_{\rm ref}$$



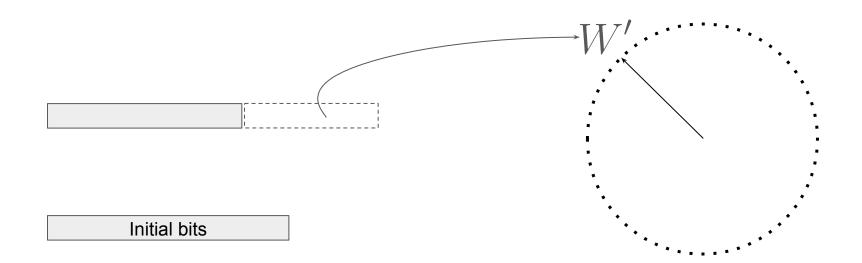
encode step 4: encode the rotated weight matrix

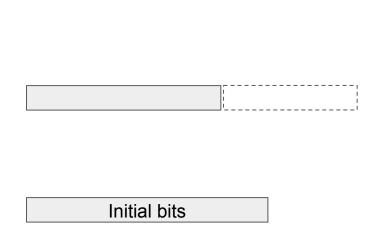


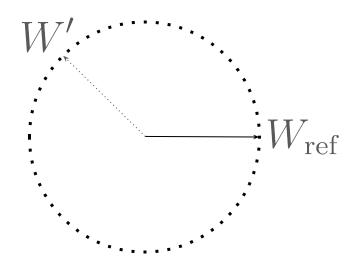
decode:

decode: start with some bits

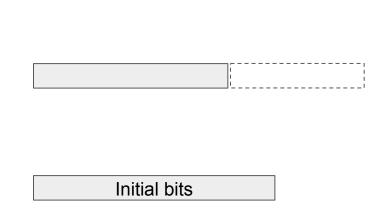
decode step 1: decode the rotated weight matrix

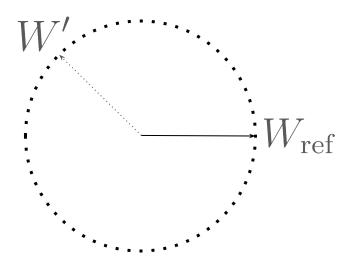






recall we define
$$\operatorname{svd}(W) = U[\Sigma V^{\top}]$$





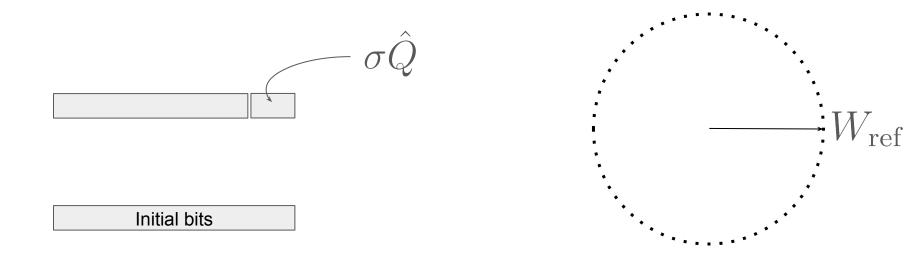
recall we define
$$\operatorname{svd}(W) = U \Sigma V^{\top}$$

$$\overline{W}_{\operatorname{ref}}$$

$$\operatorname{svd}(W') = \overline{\sigma} Q \Sigma V^{\top} \sigma, \ \sigma = \operatorname{diag}(\pm 1, \cdots, \pm 1)$$

$$W' \cdot \cdots \cdot \cdots \cdot W' \cdot \cdots \cdot \cdots \cdot W_{\operatorname{ref}}$$
 Initial bits

decode step 3: encode the recovered rotation matrix



encoding:	decoding:
1. rotation to canonical direction	l .
2. decode a rotation	3. encode the rotation
3. rotate weight matrix	2. rotate weight to canonical direction
4. encode the rotated matrix	1. decode the rotated matrix

encoding: decoding:

3. encode the rotation

2. rotate weight to canonical direction

1. decode the rotated matrix

1. rotation to canonical direction

2. decode a rotation

3. rotate weight matrix

encode the rotated matrix

does rotation need infinite precision?

encoding: decoding:
1. rotation to canonical direction
2. decode a rotation
3. encode the rotation

decode a rotation rotate weight matrix encode the rotated matrix decode the rotated matrix decode the rotated matrix decode the rotated matrix decode the rotated matrix

4. encode the rotated matrix 1. decode the rotated matrix does rotation need infinite precision?

yes. but just using float16 also works well!

encoding:

decoding:

1. rotation to canonical direction

3. rotate weight matrix

2. decode a rotation 3. encode the rotation

2. rotate weight to canonical direction

- encode the rotated matrix 1. decode the rotated matrix
- but there may be numerical error...

encoding:

decoding:

- 1. rotation to canonical direction
- 2. decode a rotation = 3. encode the rotation
- 3. rotate weight matrix (2. rotate weight to canonical direction)
- 4. encode the rotated matrix 1. decode the rotated matrix
- but there may be numerical error...

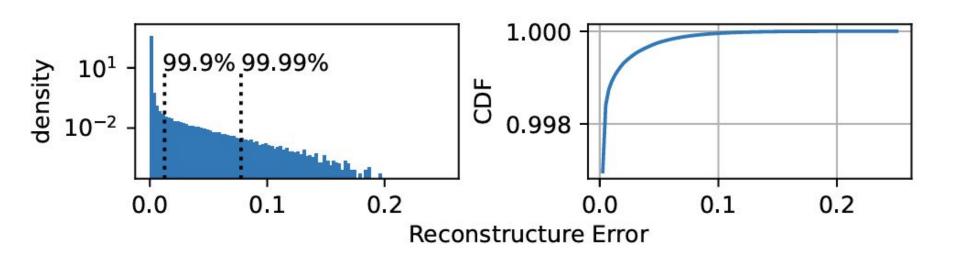




We can send correction code if the error is too large!

- but there may be numerical error...
- We can send correction code if the error is too large!
- But will this correction code becomes too large?

- but there may be numerical error...
- We can send correction code if the error is too large!
- But will this correction code becomes too large? NO!



Model	SliceGPT	Compress Rate	Compress Rate	ate Performance (before/after bits-back)					
Model	Slicing	after SliceGPT	after bits-back	PPL (↓)	PIQA (%, ↑)	WinoGrande (%, ↑)	HellaSwag (%, ↑)		
	20%	-9.53%	-13.77%	16.59 /16.60	64.91 /64.80	54.78 /54.38	45.26/ 45.32		
OPT-1.3B	25%	-14.84%	-18.61%	17.78 /17.86	63.55 /63.33	52.80/ 53.28	43.20 /43.11		
	30%	-20.53%	-23.81%	19.60 /19.66	60.88 /60.50	52.88/ 53.28	40.25 /40.06		
	20%	-9.19%	-13.84%	13.89 /13.95	68.44 /68.12	58.88 /58.72	51.35 /51.17		
OPT-2.7B	25%	-15.07%	-19.09%	14.85 /14.87	66.70 /66.76	57.30/ 57.70	48.41 /48.38		
	30%	-20.88%	-24.43%	16.31 /16.33	64.64/ 64.69	55.80/ 56.04	44.52/ 44.57		
	20%	-9.29%	-14.07%	11.63 /11.71	72.91/ 73.01	61.33 /61.17	60.53/ 60.55		
OPT-6.7B	25%	-15.16%	-19.29%	12.12/12.15	71.00/71.22	60.30/ 60.77	57.76 /57.55		
	30%	-21.18%	-24.84%	12.81 /12.91	69.31/ 69.42	59.75 /59.59	53.64 /52.94		
	20%	-9.18%	-14.01%	10.75 /10.77	74.27/74.27	64.96 /64.88	65.74/ 65.79		
OPT-13B	25%	-15.27%	-19.51%	11.08/ 11.07	74.27 /73.72	63.46/ 63.93	63.48 /63.09		
	30%	-21.29%	-24.97%	11.55 /11.59	72.69/ 73.01	61.96/ 62.43	60.12 /60.05		
	20%	-9.38%	-14.13%	6.86 /6.98	69.53 /69.42	64.17/ 64.72	58.96 /58.89		
Llama-2-7B	25%	-15.34%	-19.53%	7.56 /7.59	67.03/ 67.57	62.98/ 63.38	54.29 /53.93		
	30%	-21.45%	-25.09%	8.63 /8.69	64.69 /64.09	62.75 /62.12	49.13 /49.07		

3-5% additional bits saving

		_						
Model	SliceGPT	Compress Rate	Compress Rate	Performance (before/after bits-back)				
Model	Slicing	after SliceGPT	after bits-back	PPL (↓)	PIQA (%, ↑)	WinoGrande (%, ↑)	HellaSwag (%, ↑)	
	20%	-9.53%	-13.77%	16.59 /16.60	64.91 /64.80	54.78 /54.38	45.26/ 45.32	
OPT-1.3B	25%	-14.84%	-18.61%	17.78 /17.86	63.55 /63.33	52.80/ 53.28	43.20 /43.11	
	30%	-20.53%	-23.81%	19.60 /19.66	60.88 /60.50	52.88/ 53.28	40.25 /40.06	
10	20%	-9.19%	-13.84%	13.89 /13.95	68.44 /68.12	58.88 /58.72	51.35 /51.17	
OPT-2.7B	25%	-15.07%	-19.09%	14.85 /14.87	66.70 /66.76	57.30/ 57.70	48.41 /48.38	
	30%	-20.88%	-24.43%	16.31 /16.33	64.64/ 64.69	55.80/ 56.04	44.52/ 44.57	
	20%	-9.29%	-14.07%	11.63 /11.71	72.91/ 73.01	61.33 /61.17	60.53/ 60.55	
OPT-6.7B	25%	-15.16%	-19.29%	12.12 /12.15	71.00/71.22	60.30/ 60.77	57.76 /57.55	
	30%	-21.18%	-24.84%	12.81 /12.91	69.31/ 69.42	59.75 /59.59	53.64 /52.94	
	20%	-9.18%	-14.01%	10.75 /10.77	74.27/74.27	64.96 /64.88	65.74/ 65.79	
OPT-13B	25%	-15.27%	-19.51%	11.08/ 11.07	74.27 /73.72	63.46/ 63.93	63.48 /63.09	
	30%	-21.29%	-24.97%	11.55 /11.59	72.69/ 73.01	61.96/ 62.43	60.12 /60.05	
t a.	20%	-9.38%	-14.13%	6.86 /6.98	69.53 /69.42	64.17/ 64.72	58.96 /58.89	
Llama-2-7B	25%	-15.34%	-19.53%	7.56 /7.59	67.03/ 67.57	62.98/ 63.38	54.29 /53.93	
	30%	-21.45%	-25.09%	8.63 /8.69	64.69 /64.09	62.75 /62.12	49.13 /49.07	

SliceGPT	Compress Rate	Compress Rate	Performance (before/after bits-back)					
Slicing	after SliceGPT	after bits-back	PPL (↓)	PIQA (%, ↑)	WinoGrande (%, ↑)	HellaSwag (%, ↑)		
20%	-9.53%	-13.77%	16.59 /16.60	64.91 /64.80	54.78 /54.38	45.26/ 45.32		
25%	-14.84%	-18.61%	17.78 /17.86	63.55 /63.33	52.80/ 53.28	43.20 /43.11		
30%	-20.53%	-23.81%	19.60 /19.66	60.88 /60.50	52.88/ 53.28	40.25 /40.06		
20%	-9.19%	-13.84%	13.89 /13.95	68.44 /68.12	58.88 /58.72	51.35 /51.17		
25%	-15.07%	-19.09%	14.85 /14.87	66.70 /66.76	57.30/ 57.70	48.41 /48.38		
30%	-20.88%	-24.43%	16.31 /16.33	64.64/ 64.69	55.80/ 56.04	44.52/ 44.57		
20%	-9.29%	-14.07%	11.63 /11.71	72.91/ 73.01	61.33 /61.17	60.53/ 60.55		
25%	-15.16%	-19.29%	12.12 /12.15	71.00/71.22	60.30/ 60.77	57.76 /57.55		
30%	-21.18%	-24.84%	12.81 /12.91	69.31/ 69.42	59.75 /59.59	53.64 /52.94		
20%	-9.18%	-14.01%	10.75 /10.77	74.27/74.27	64.96 /64.88	65.74/ 65.79		
25%	-15.27%	-19.51%	11.08/ 11.07	74.27 /73.72	63.46/ 63.93	63.48 /63.09		
30%	-21.29%	-24.97%	11.55 /11.59	72.69/ 73.01	61.96/ 62.43	60.12 /60.05		
20%	-9.38%	-14.13%	6.86 /6.98	69.53 /69.42	64.17/ 64.72	58.96 /58.89		
25%	-15.34%	-19.53%	7.56 /7.59	67.03/ 67.57	62.98/ 63.38	54.29 /53.93		
30%	-21.45%	-25.09%	8.63 /8.69	64.69 /64.09	62.75 /62.12	49.13 /49.07		
	20% 25% 30% 25% 30% 25% 30% 25% 30% 25% 30% 25% 30% 25% 20% 25% 30%	Slicing after SliceGPT 20% -9.53% 25% -14.84% 30% -20.53% 20% -9.19% 25% -15.07% 30% -20.88% 20% -9.29% 25% -15.16% 30% -21.18% 20% -9.18% 25% -15.27% 30% -21.29% 20% -9.38% 25% -15.34%	Slicing after SliceGPT after bits-back 20% -9.53% -13.77% 25% -14.84% -18.61% 30% -20.53% -23.81% 20% -9.19% -13.84% 25% -15.07% -19.09% 30% -20.88% -24.43% 20% -9.29% -14.07% 25% -15.16% -19.29% 30% -21.18% -24.84% 20% -9.18% -14.01% 25% -15.27% -19.51% 30% -21.29% -24.97% 20% -9.38% -14.13% 25% -15.34% -19.53%	Slicing after SliceGPT after bits-back PPL (↓) 20% -9.53% -13.77% 16.59/16.60 25% -14.84% -18.61% 17.78/17.86 30% -20.53% -23.81% 19.60/19.66 20% -9.19% -13.84% 13.89/13.95 25% -15.07% -19.09% 14.85/14.87 30% -20.88% -24.43% 16.31/16.33 20% -9.29% -14.07% 11.63/11.71 25% -15.16% -19.29% 12.12/12.15 30% -21.18% -24.84% 12.81/12.91 20% -9.18% -14.01% 10.75/10.77 25% -15.27% -19.51% 11.08/11.07 30% -21.29% -24.97% 11.55/11.59 20% -9.38% -14.13% 6.86/6.98 25% -15.34% -19.53% 7.56/7.59	Slicing after SliceGPT after bits-back PPL (↓) PIQA (%, ↑) 20% -9.53% -13.77% 16.59/16.60 64.91/64.80 25% -14.84% -18.61% 17.78/17.86 63.55/63.33 30% -20.53% -23.81% 19.60/19.66 60.88/60.50 20% -9.19% -13.84% 13.89/13.95 68.44/68.12 25% -15.07% -19.09% 14.85/14.87 66.70/66.76 30% -20.88% -24.43% 16.31/16.33 64.64/64.69 20% -9.29% -14.07% 11.63/11.71 72.91/73.01 25% -15.16% -19.29% 12.12/12.15 71.00/71.22 30% -21.18% -24.84% 12.81/12.91 69.31/69.42 20% -9.18% -14.01% 10.75/10.77 74.27/74.27 25% -15.27% -19.51% 11.08/11.07 74.27/73.72 30% -21.29% -24.97% 11.55/11.59 72.69/73.01 20% -9.38% -14.13% 6.86/6.98	Slicing after SliceGPT after bits-back PPL (↓) PIQA (%, ↑) WinoGrande (%, ↑) 20% -9.53% -13.77% 16.59/16.60 64.91/64.80 54.78/54.38 25% -14.84% -18.61% 17.78/17.86 63.55/63.33 52.80/53.28 30% -20.53% -23.81% 19.60/19.66 60.88/60.50 52.88/53.28 20% -9.19% -13.84% 13.89/13.95 68.44/68.12 58.88/58.72 25% -15.07% -19.09% 14.85/14.87 66.70/66.76 57.30/57.70 30% -20.88% -24.43% 16.31/16.33 64.64/64.69 55.80/56.04 20% -9.29% -14.07% 11.63/11.71 72.91/73.01 61.33/61.17 25% -15.16% -19.29% 12.12/12.15 71.00/71.22 60.30/60.77 30% -21.18% -24.84% 12.81/12.91 69.31/69.42 59.75/59.59 20% -9.18% -14.01% 10.75/10.77 74.27/74.27 64.96/64.88 25% -15.27% -19.51%		

negligible influence on performance

Encoding and Decoding time

GPU:

Model Name	OPT-1.3B		OPT-2.7B		OPT-6.7B		OPT-13B	
Slicing Encoding time Decoding time			30 s		20% 2.5 min 1.2 min	30% 1.7 min 45 s	20% 6.5 min 2.5 min	30% 4.1 min 2 min

CPU:

Model Name	OPT-1.3B		OPT-2.7B		OPT-6.7B		OPT-13B	
Slicing	20%	30%	20%	30%	20%	30%	20%	30%
Encoding time	3.9 min	3.5 min	8 min	6.5 min	30 min	25 min	84 min	68 min
Decoding time	1.5 min	1.5 min	3.5 min	2.5 min	12 min	10 min	30 min	24 min

In summary:

save 3-5% additional bits,

In summary:

save 3-5% additional bits, no influence on performance,

In summary:

save 3-5% additional bits, no influence on performance, a little overhead in model loading time