

Эффективные модели: Квантизация

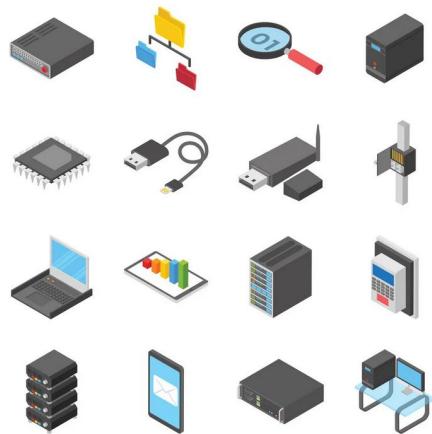
Егор Швецов

tg: @dalime e-mail: e.shvetsov@skoltech.ru

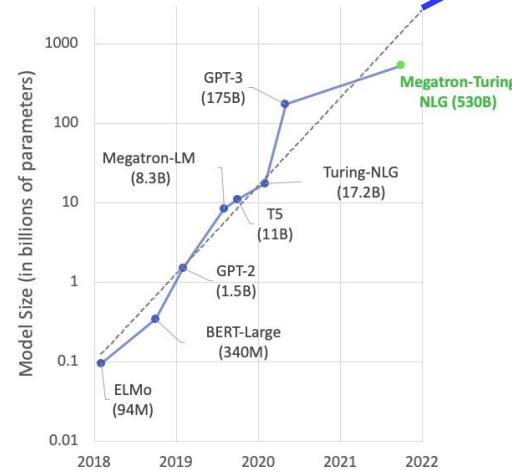
План на сегодня:

- Мотивация
- Типы данных
- Линейное квантование
- Детализация квантования
- Квантование смешанной точности
- STE и Шум
- Векторное квантование - Квантование для сжатия
- Бинаризация

Периферийные устройства
Автономные устройства
Робототехника



Современные LLM



- Memory
- FLOPs
- Latency
- Energy

- Memory
- FLOPs
- Latency
- Energy

Input Type	Accumulation Type	Relative math throughput	Bandwidth
FP16	FP16	8x	2x
INT8	INT32	16x	4x
INT4	INT32	32x	8x
INT1	INT32	128x	32x

*Омнісимально FP32

model	gpu	task	energy	throughput	response_length	latency	arc	hellaswag	truthfulqa	parameters
MetaAI/Llama-7B	A100	chat	335.26	27.47	60.65	1.99	51.11	77.74	34.08	7
MetaAI/Llama-13B	A100	chat	631.57	23.2	76.47	2.97	56.31	80.86	39.9	13
tatsu-lab/alpaca-7B	A100	chat	684.17	28	132.85	4.63	52.65	76.91	39.55	7
RwKV/rwkv-raven-7b	A100	chat	735.77	61.52	214.78	3.08	39.42	66.45	38.54	7
Neutralzz/Billa-7B-SFT	A100	chat	881.44	33.57	161.81	4.79	27.73	26.04	49.05	7
H2OAI/H20GPT-casst1-7B	A100	chat	951.08	33.2	212.46	6.39	36.86	61.55	37.94	7
FreedomIntelligence/phoenix-inst-chat-7b	A100	chat	1030.89	55.84	240.34	4.12	44.97	63.22	47.08	7
StabilityAI/stablelm-tuned-alpha-7b	A100	chat	1124.4	45.42	243.88	5.21	31.91	53.59	40.22	7
databricks/dolly-v2-12B	A100	chat	1242.92	22.11	153.76	7.94	42.15	71.83	33.37	12
Salesforce/xgen-7b-8k-inst	A100	chat	1246.09	49.54	275.27	5.6	46.67	74.85	41.89	7
BAIR/koala-7b	A100	chat	1345.55	26.56	261.07	9.62	47.1	73.7	46	7
togethercomputer/RedPajama-INCITE-7B-Chat	A100	chat	1457.86	27.49	274.39	9.54	42.15	70.84	36.1	7
LMSys/vicuna-7B	A100	chat	1531.7	27.26	284.92	10.3	53.5	77.53	49	7
project-baize/baize-v2-7B	A100	chat	1588.27	31.59	326.3	10.17	48.46	75	41.66	7
LMSys/fastchat-t5-3b-v1.0	A100	chat	1802.98	19.29	314.3	20.68	35.92	46.36	48.79	3
nomic-ai/gpt4all-13b-snoozy	A100	chat	2004.73	26.57	247.8	9.28	56.06	78.69	48.36	13
OpenAssistant/vasst-sft-1-pythia-12b	A100	chat	2060.03	25.4	259.59	9.93	45.56	69.93	39.19	12
BAIR/koala-13b	A100	chat	2144.83	21.21	265.57	12.14	52.9	77.54	50.09	13
openaccess-ai-collective/manticore-13b-cl	A100	chat	2189.25	26.28	288.82	10.95	58.7	81.96	48.86	13
Camel-AI/CAMEL-13B-Combined-Data	A100	chat	2526.46	24.5	291.92	13.17	55.55	79.3	47.33	13
LMSys/vicuna-13B	A100	chat	2600.84	21.61	280.74	12.74	52.9	80.12	51.82	13

Сколько энергии потребляют LLaMA
пересчете на мощность работы
лампочки 60Вт:



- 3 секунды работы
- 5 секунды работы
- 60 секунды работы

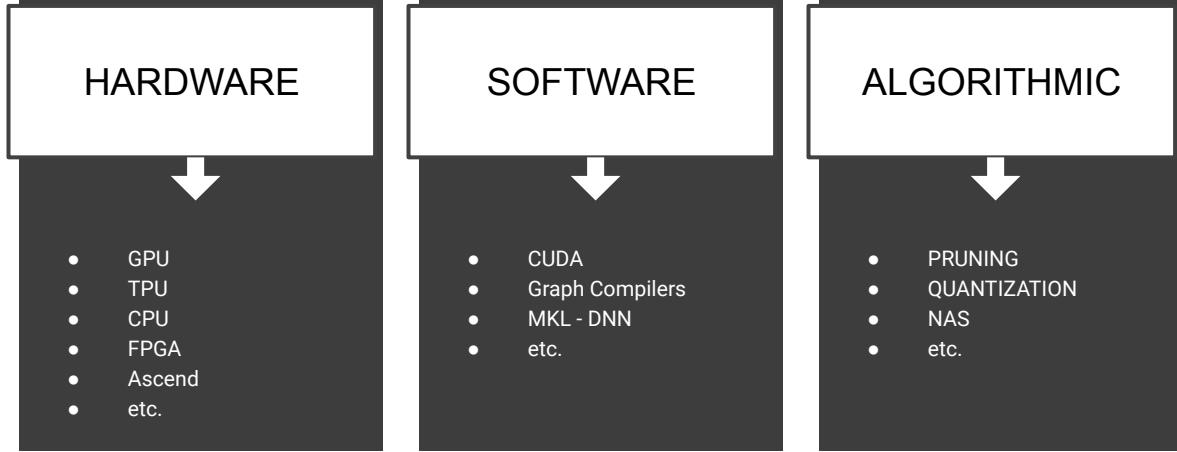
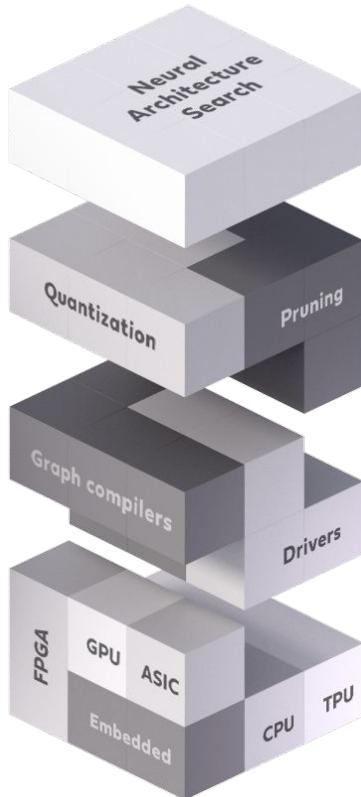
Человеческий мозг потребляет 20
джоулей в секунду.



- Memory
- FLOPs
- Latency
- Energy

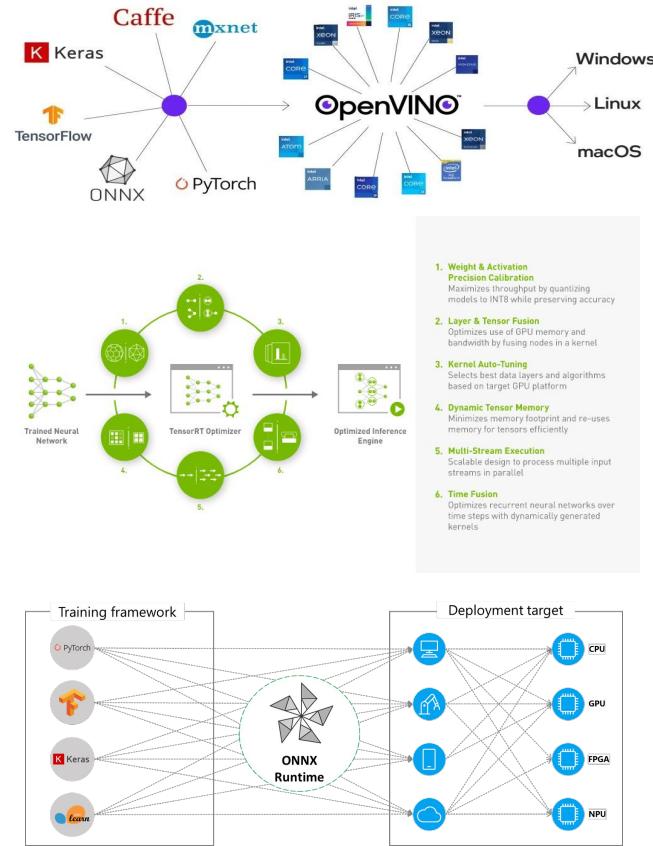
ARITHMETIC OPERATION	ADD	MUL
8-BIT INTEGER	0.03 pJ	0.2 pJ
16-BIT FLOATING POINT	0.4 pJ	1.1 pJ
32-BIT INTEGER	0.1 pJ	3.1 pJ
32-BIT FLOATING POINT	0.9 pJ	3.7 pJ

MEMORY SIZE	64-BIT MEMORY ACCESS
8KB	10 pJ
32KB	20 pJ
1 MB	100 pJ
DRAM	1.3-2.6 nJ



В большинстве архитектур машинного обучения не учитываются свойства программного и аппаратного обеспечения, что приводит к неэффективности вычислений.

Более того, многие модели концентрируются исключительно на достижении оптимальных результатов, а не на вопросах практической реализации.



ONNX: Это инструмент предоставляет стандартизированный формат для моделей глубокого обучения, что позволяет легко использовать их в различных средах и платформах. Часто ONNX используется для обеспечения плавного перехода между различными средами.

ONNX Runtime	Открытый исходный код, первоначально созданный Microsoft и Facebook.	Может использоваться как интерфейс высокого уровня для TensorRT и OpenVino.
TensorRT	Nvidia	Этот инструмент специально разработан для графических процессоров NVIDIA и ориентирован на максимизацию пропускной способности и эффективности. Он оптимизирует модели нейронных сетей путем объединения слоев, выбора наиболее эффективных форматов данных и использования арифметики с пониженной точностью (FP16), где это возможно.
OpenVino	Intel	OpenVINO, разработанный Intel, специализируется на оптимизации моделей глубокого обучения для оборудования Intel, особенно процессоров. Это важнейший инструмент для повышения производительности моделей, в которых ресурсы графического процессора недоступны или ограничены.

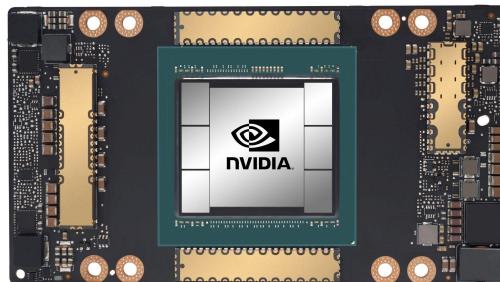
Поддержка вычислений низкой точности в графическом процессоре

- Float16
- Float32
- Bfloat16

Поддерживается аппаратным и программным обеспечением для большинства современных графических процессоров.

Will be discussed in future series ...

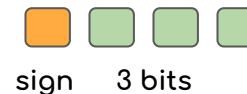
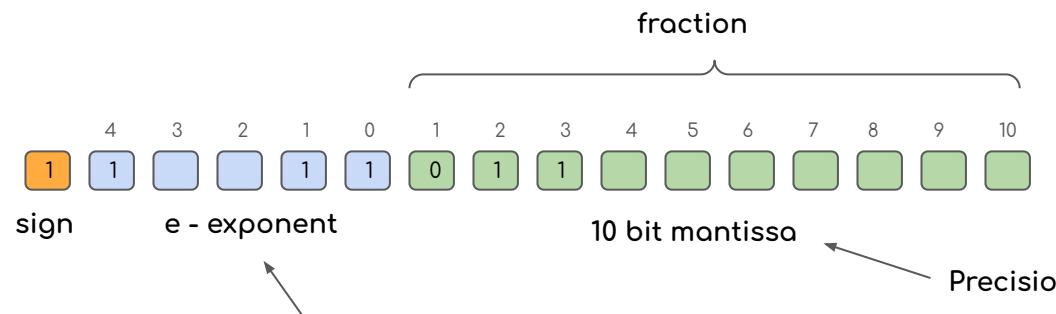
Не поддерживается аппаратным или программным обеспечением



Поддерживается только некоторым оборудованием

>>> Форматы Данных

Int16/Int8/Int4

Range of values: $[-2^{n-1}, 2^{n-1} - 1]$ Float32 and Float16
(Floating Point)

Range of values

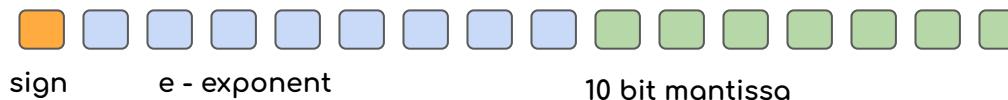
$$\mathbf{v} = (-1)^1(1 + f)2^{e-bias} \quad bias = 15$$

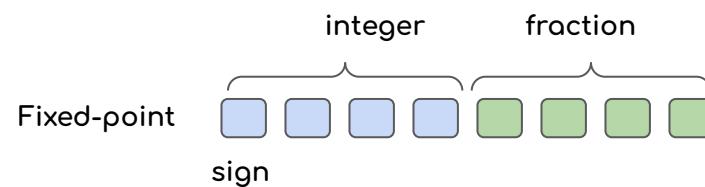
$$f = 2^{-2} + 2^{-3} = \frac{1}{4} + \frac{1}{8} = 0.375$$

$$e = 2^0 + 2^1 + 2^4 = 19$$

$$\mathbf{v} = (-1)^1(1 + 0.375)2^{19-15} = 22$$

Bfloat16





Другие форматы:

TensorFloat-32 19 bit (supported in A100): 1 bit sign, 8 bit exponent, 10 bit —mantissa.

E4M3 (8bit) 1 bit sign, 4 bit exponent, 3 bit mantissa

E5M2 (8bit) 1 bit sign, 5 bit exponent, 2 bit mantissa

E2M1 (4bit) 1 bit sign, 2 bit exponent, 1 bit mantissa

E3M0 (4bit) 1 bit sign, 3 bit exponent

Итак, Float8 и int8 или Float4 и int4 используют одинаковое количество бит.
Почему нам тогда интересны int4 или int8?

E4M3 (8bit) 1 bit sign, 4 bit exponent, 3 bit mantissa

E5M2 (8bit) 1 bit sign, 5 bit exponent, 2 bit mantissa

E2M1 (4bit) 1 bit sign, 2 bit exponent, 1 bit mantissa

E3M0 (4bit) 1 bit sign, 3 bit exponent

Int16/Int8/Int4



Range of values: $[-2^{n-1}, 2^{n-1} - 1]$

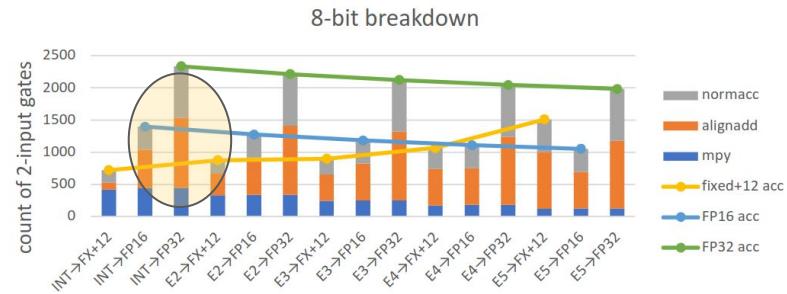
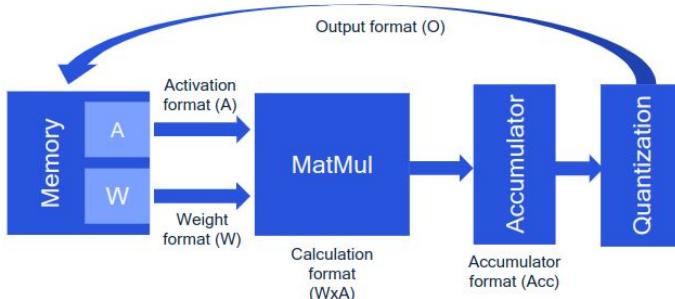
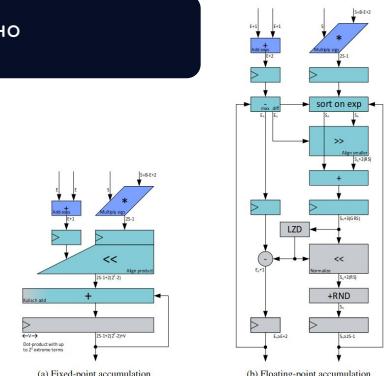
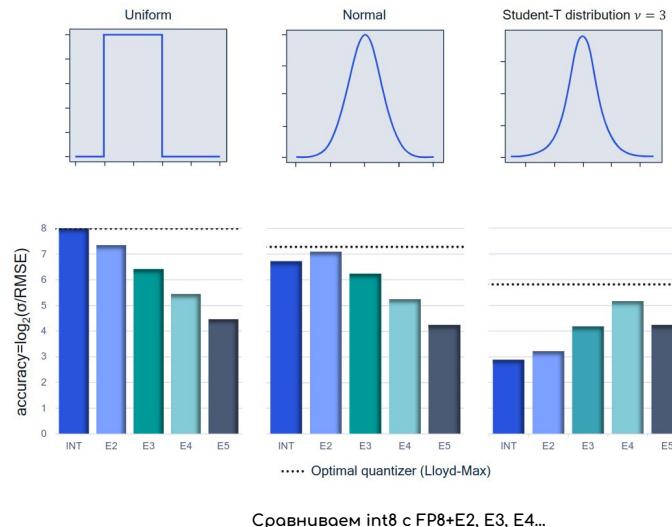


Figure 3: A count of the number of 2-input gates necessary in hardware to implement each format and accumulator combination. From left to right, INT8 with increasing exponent bits until FP8-E5. In each group of three, the first bar is for a 15+12=27-bit fixed-point accumulator. The second bar indicates the numbers for FP16 accumulation, and the third bar is the result of an FP32 accumulator. We can see that FP8-E4->16-bit requires 53% more gates, and FP8-E4->32-bit requires 183% more gates for an implementation in hardware.

Это более эффективно



Работаем также в некоторых случаях



Сравниваем int8 с FP8+E2, E3, E4...

Наличие большего количества битов экспоненты имеет смысл только когда в игру вступают выбросы!

PTQ — квантование после обучения

Task	Model	FP32 ↑	INT8	FP8-E2	FP8-E3	FP8-E4	FP8-E5
Classification	ResNet18	69.72%	-0.08%	-0.06%	-0.27%	-1.15%	<u>-4.8%</u>
	ResNet50	76.06%	-0.07%	-0.05%	-0.8%	-0.99%	<u>-3.82%</u>
	EfficientNet B0	77.35%	<u>-14.65%</u>	-4.78%	-2.8%	-3.7%	-9.18%
	EfficientFormer	80.21%	<u>-50.35%</u>	-4.95%	-0.41%	-1.43%	-7.41 %
	DLA102	77.94%	-0.19%	-0.31%	-0.54%	-2.13%	<u>-7.95%</u>
	MobileNetV2	71.70%	-0.76%	-0.64%	-1.08%	-5.65%	<u>-22.19%</u>
	MobileNetV3	73.84%	-3.71%	-1.48%	-1.62%	-3.33%	<u>-12.06%</u>
Object Detection	ViT	77.75%	-1.33%	-0.45%	-0.04%	-0.19%	<u>-76.69%</u>
Segmentation	YoloV5	56.3	-1.8	-0.5	-0.3	-2	-9.9
	HRNet	81.05	-0.12	-0.02	-0.01	-0.28	<u>-1.06</u>
	CP-Pointpillar	40.94	<u>-21.41</u>	-14.81	-2.86	-2.93	-7.06
	RangeNet++	0.305	-1.67	-0.9	-0.4	-1.3	<u>-3.8</u>
	FFNet	79.16	<u>-1.15</u>	-0.26	-0.31	-0.41	-1.07
	DeeplabV3	72.91	-1.67	-0.33	-1.63	-34.98	<u>-66.38</u>
NLP	SalsaNext	55.80	-1.58	-0.28	-0.13	-0.68	<u>-2.72</u>
SuperResolution	BERT (GLUE)	83.06	<u>-12.03</u>	-2.75	-0.45	-0.26	-0.25
	QuickSRNet	32.79	-0.8	-0.44	-1.78	-3.24	<u>-6.8</u>

Итак, INT8 и FP8-E2 — это то, что нам нужно!

QAT — обучение с учетом квантования

Model	FP32	INT8	FP8-E2	FP8-E3	FP8-E4	W4A8
ResNet18	69.72	70.43	70.25	70.20	<u>69.35</u>	70.01
MobileNetV2	71.70	71.82	71.76	71.56	<u>70.89</u>	71.17
HRNet	81.05	81.27	81.20	81.14	<u>81.06</u>	-
DeeplabV3	72.91	73.99	73.67	73.74	73.22	<u>73.01</u>
SalsaNext (SemanticKITTI)	55.80	<u>55.0</u>	55.3	55.7	55.2	-
BERT (GLUE avg)	83.06	83.26	81.20	83.74	83.91	82.64

Выбросы обрезаются во время квантования —
работает как регуляризация!

Задержка при различных размерах пакетов

Фактический выигрыш отличается от теоретического и зависит от алгоритма и серверной реализации.

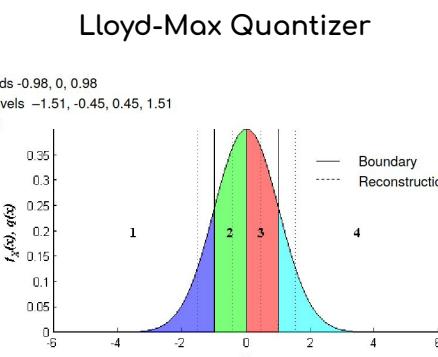
Image/s	Batch size 1 x2			Batch size 8			Batch size 128 x8		
	FP32	FP16	Int8	FP32	FP16	Int8	FP32	FP16	Int8
MobileNet v1	1509	2889	3762	2455	7430	13493	2718	8247	16885
MobileNet v2	1082	1618	2060	2267	5307	9016	2761	6431	12652
ResNet50 (v1.5)	298	617	1051	500	2045	3625	580	2475	4609
VGG-16	153	403	415	197	816	1269	236	915	1889
VGG-19	124	358	384	158	673	1101	187	749	1552
Inception v3	156	371	616	350	1318	2228	385	1507	2560
Inception v4	76	226	335	173	768	1219	186	853	1339
ResNext101	84	208	297	200	716	1253	233	899	1724

<https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9659-inference-at-reduced-precision-on-gpus.pdf>

Другие форматы

NormalFloat4

- Нормализовать все веса;
- Использует квантильный квантовомател;
- По сути словарь с 16 значениями: -1, 1 и 14 ячеек.



Источники:

- [1] [8-BIT APPROXIMATIONS FOR PARALLELISM IN DEEP LEARNING](#)
- [2] [ANT: Exploiting Adaptive Numerical Data Type for Low-bit Deep Neural Network](#)
- [3] [QuantizationA White Paper on Neural Network Quantization](#)
- [4] [Low-Precision Arithmetic](#)
- [5] [FP8 Quantization: The Power of the Exponent](#)
- [6] [NF4 Isn't Information Theoretically Optimal \(and that's Good\)](#)
- [7] [FP8 versus INT8 for efficient deep learning inference](#)
- [8] [FP8 Formats for Deep Learning](#)
- [9] [TensorFloat-32 in the A100 GPU](#)

АДАПТИВНЫЕ ТИПЫ ДАННЫХ

Algorithm 2: ANT data type selection algorithm.

```

Input: Tensor,  $T$ ; Candidate list of numeric types,  $L$ .
Output: Quantization function,  $F_Q$ .
1 def ANT ( $T, L$ ) :
2    $minMSE = 10^9$ ;
3   foreach  $l \in L$  do
4      $F = \text{GetQuantFunc}(l)$ ; // Get the
      quantization method of  $l$ .
5      $m = \text{ArgminMSE}(T, F)$ ; // Search the
      minimum MSE with range
      clipping.
6     if  $m < minMSE$  then
7        $\quad F_Q = F$ ;
8   return  $F_Q$ 

```

Эффекты вычислений низкой точности

Плюсы:

- Меньшее количество бит позволяет лучше использовать память, например: передавать больше данных в секунду
- Целые числа имеют более быструю арифметику
- Расходует меньше энергии

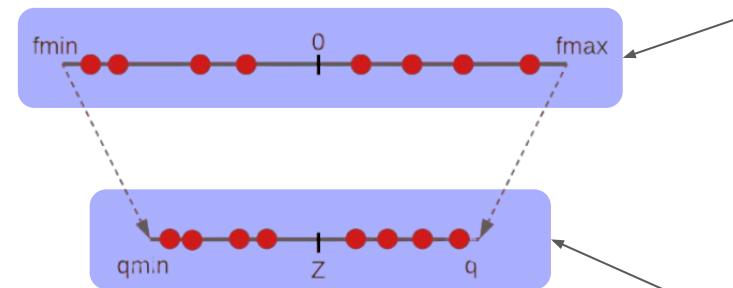
Минусы:

- Ограниченнная репрезентативность
- Вносит шум квантования – ошибка округления или дискретизации

>>>

Квантование

Квантование – это процедура отображения в дискретный диапазон фиксированной длины.



$$\text{Scale: } s = \frac{f_{\max} - f_{\min}}{f q_{\max} - f q_{\min}}$$

$$\text{Quantize: } Q(x) = \text{clip}(\text{round}(\frac{1}{s}x), f q_{\min}, f q_{\max})$$

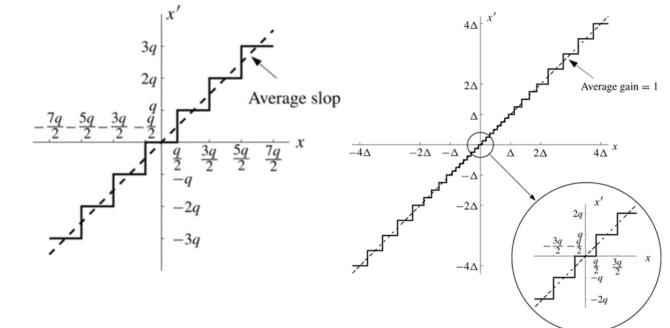
$$\text{De-Quantize: } x = sQ(x)$$

$$f q_{\min} = -2^{(bit-1)}$$

$$f q_{\max} = 2^{(bit-1)} - 1$$

изначальные
значения

квантованные



$$\text{FP32 tensor} \rightarrow X \approx s_X X_{\text{int}} = \hat{X} \leftarrow \text{scaled quantized tensor}$$

$$W = \begin{pmatrix} 0.97 & 0.64 & 0.74 & 1.00 \\ 0.58 & 0.84 & 0.84 & 0.81 \\ 0.00 & 0.18 & 0.90 & 0.28 \\ 0.57 & 0.96 & 0.80 & 0.81 \end{pmatrix} \approx \frac{1}{255} \begin{pmatrix} 247 & 163 & 189 & 255 \\ 148 & 214 & 214 & 207 \\ 0 & 46 & 229 & 71 \\ 145 & 245 & 204 & 207 \end{pmatrix} = s_W W_{\text{uint8}}$$

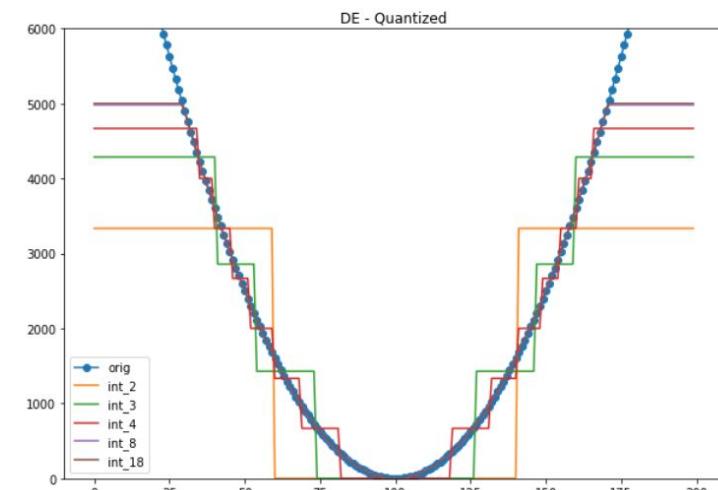
Симметричное равномерное квантование со знаком

Квантование – это процедура отображения в дискретный диапазон фиксированной длины.

```

1 def quantization(x, s, z, alpha_q, beta_q):
2
3     x_q = np.round(1 / s * x + z, decimals=0)
4     x_q = np.clip(x_q, a_min=alpha_q, a_max=beta_q)
5
6     return x_q
7
8 def de_quantization(x_q, s, z):
9
10    # x_q - z might go outside the quantization range.
11    x_q = x_q.astype(np.int32)
12    x = s * (x_q - z)
13    x = x.astype(np.float32)
14    return x
15
16 def generate_quantization_constants(alpha, beta, alpha_q, beta_q):
17
18    # Affine quantization mapping
19    s = (beta - alpha) / (beta_q - alpha_q)
20    return s
21
22
23 def quantize_with_bit(x, bit):
24
25    alpha_q = -2**bit - 1
26    beta_q = 2**bit - 1
27
28    s = (x.max() - x.min()) / (beta_q - alpha_q)
29
30    x_q = quantization(x, s, 0, alpha_q, beta_q)
31    return de_quantization(x_q, s, 0), x_q
32

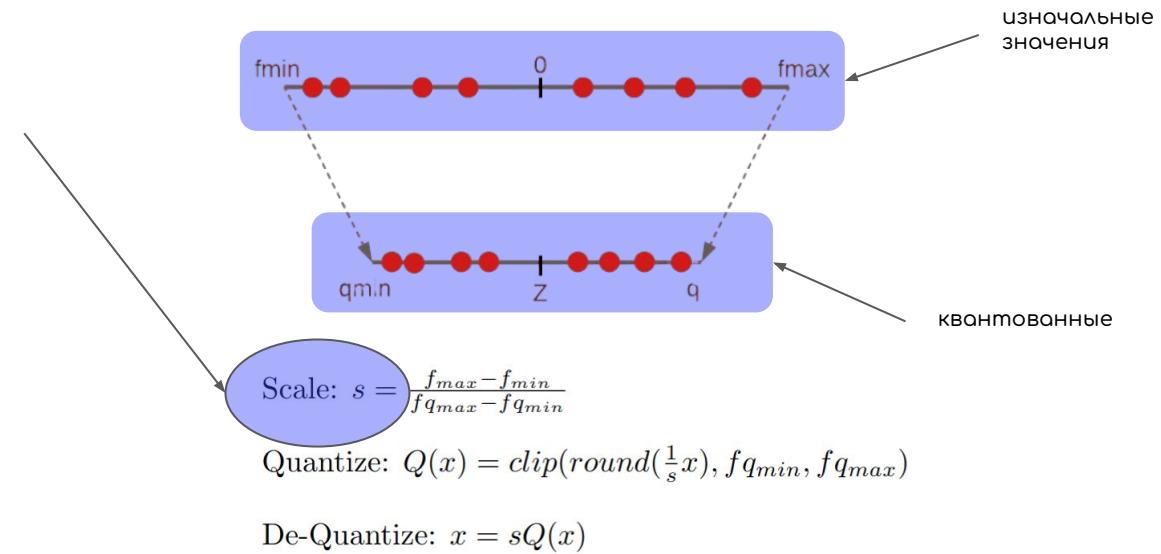
```



Квантование – это процедура отображения в дискретный диапазон фиксированной длины.

Какие проблемы могут возникнуть при выборе масштаба?

Предположим, нам нужно квантовать как активации, так и веса.



$$fq_{min} = -2^{(bit-1)}$$

$$fq_{max} = 2^{(bit-1)} - 1$$

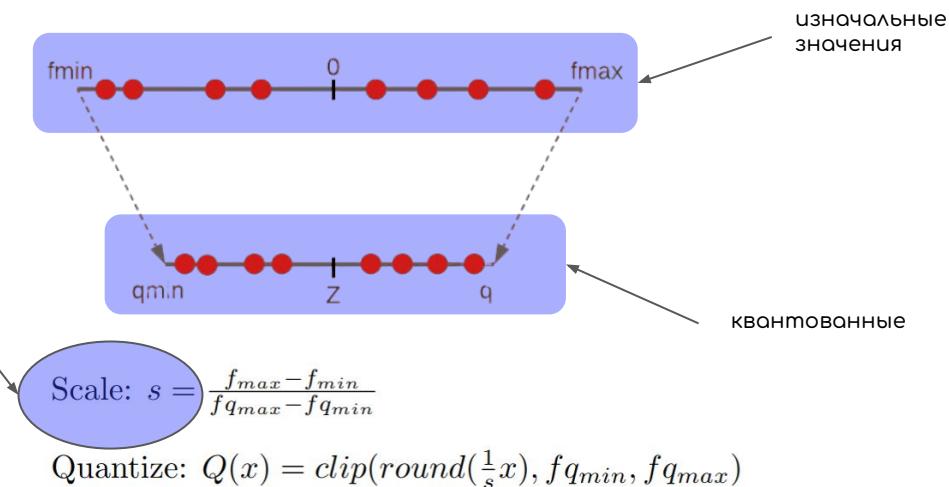
Квантование – это процедура отображения в дискретный диапазон фиксированной длины.

Какие проблемы могут возникнуть при выборе масштаба?

Предположим, нам нужно квантовать как активации, так и веса.

- Собрать статистику по активациям на калибровочной выборке;
- Сделать s обучаемым параметром;
- Найдите такой s , чтобы минимизировать его послойное MSE:

$$\min_{s_q, s_x} ||WX^T - Q(W, s_q)Q(X^T, s_x)||$$



$$\text{De-Quantize: } x = sQ(x)$$

$$f_{q_{\min}} = -2^{(bit-1)}$$

$$f_{q_{\max}} = 2^{(bit-1)} - 1$$

LEARNED STEP SIZE QUANTIZATION

Steven K. Esser ^{*}, Jeffrey L. McKinstry, Deepika Bablani,
Rathinakumar Appuswamy, Dharmendra S. Modha

IBM Research
San Jose, California, USA

ABSTRACT

Deep networks run with low precision operations at inference time offer power and space advantages over high precision alternatives, but need to overcome the challenge of maintaining high accuracy as precision decreases. Here, we present a method for training such networks, Learned Step Size Quantization, that achieves the highest accuracy to date on the ImageNet dataset when using models, from a variety of architectures, with weights and activations quantized to 2-, 3- or 4-bits of precision, and that can train 3-bit models that reach full precision baseline accuracy. Our approach builds upon existing methods for learning weights in quantized networks by improving how the quantizer itself is configured. Specifically, we introduce a novel means to estimate and scale the task loss gradient at each weight and activation layer's quantizer step size, such that it can be learned in conjunction with other network parameters. This approach works using different levels of precision as needed for a given system and requires only a simple modification of existing training code.

Network	Method	Top-1 Accuracy @ Precision				Top-5 Accuracy @ Precision			
		2	3	4	8	2	3	4	8
<i>Full precision: 70.5</i>								<i>Full precision: 89.6</i>	
ResNet-18	LSQ (Ours)	67.6	70.2	71.1	71.1	87.6	89.4	90.0	90.1
	QIL	65.7	69.2	70.1					
	FAQ			69.8	70.0				
	LQ-Nets	64.9	68.2	69.3		85.9	87.9	88.8	
	PACT	64.4	68.1	69.2		85.6	88.2	89.0	
	NICE	67.7	69.8			87.9	89.2	89.21	
	Regularization	61.7		67.3	68.1	84.4		87.9	88.2
<i>Full precision: 74.1</i>								<i>Full precision: 91.8</i>	
ResNet-34	LSQ (Ours)	71.6	73.4	74.1	74.1	90.3	91.4	91.7	91.8
	QIL	70.6	73.1	73.7					
	LQ-Nets	69.8	71.9			89.1	90.2		
	NICE	71.7	73.5			90.8	91.4		
	FAQ			73.3	73.7		91.3	91.6	
<i>Full precision: 76.9</i>								<i>Full precision: 93.4</i>	
ResNet-50	LSQ (Ours)	73.7	75.8	76.7	76.8	91.5	92.7	93.2	93.4
	PACT	72.2	75.3	76.5		90.5	92.6	93.2	
	NICE	75.1	76.5			92.3	93.3		
	FAQ			76.3	76.5		92.9	93.1	
	LQ-Nets	71.5	74.2	75.1		90.3	91.6	92.4	
<i>Full precision: 78.2</i>								<i>Full precision: 94.1</i>	
ResNet-101	LSQ (Ours)	76.1	77.5	78.3	78.1	92.8	93.6	94.0	94.0
<i>Full precision: 78.9</i>								<i>Full precision: 94.3</i>	
ResNet-152	LSQ (Ours)	76.9	78.2	78.5	78.5	93.2	93.9	94.1	94.2
	FAQ			78.4	78.5		94.1	94.1	
<i>Full precision: 73.4</i>								<i>Full precision: 91.5</i>	
VGG-16bn	LSQ (Ours)	71.4	73.4	74.0	73.5	90.4	91.5	92.0	91.6
	FAQ			73.9	73.7		91.7	91.6	
<i>Full precision: 67.3</i>								<i>Full precision: 87.8</i>	
Squeeze	LSQ (Ours)	53.3	63.7	67.4	67.0	77.5	85.4	87.8	87.7
Next-23-2x									

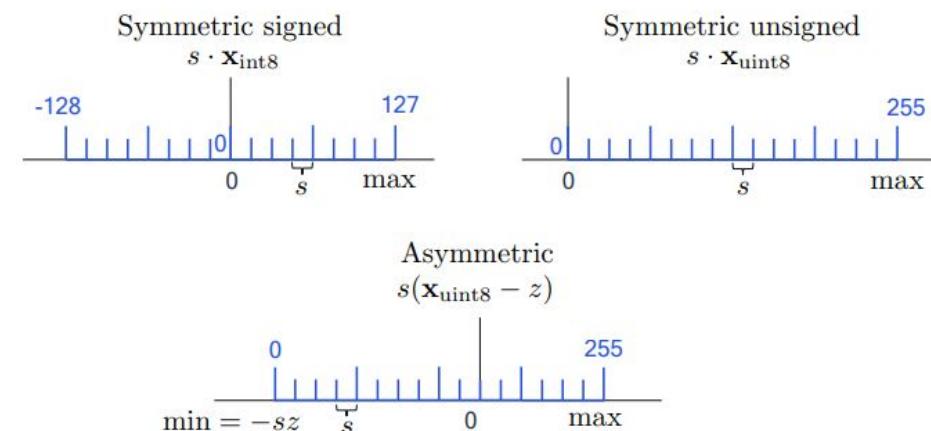
Source: [Learned Step Size Quantization](#)

Симметричное и асимметричное квантование

$$\text{Quantize} : Q(X) = s[\text{clamp}([\frac{1}{s}x] + z; 0, 2^b - 1]) - z]$$

Симметричное квантование: представляет собой упрощенную версию общего асимметричного случая. Симметричный квантователь ограничивает нулевую точку Z до 0. Это уменьшает вычислительные затраты на обработку смещения нулевой точки во время операции накопления. Но отсутствие смещения ограничивает сопоставление между целочисленной областью и областью с плавающей запятой.

$$\begin{aligned}\widehat{\mathbf{W}}\widehat{\mathbf{x}} &= s_{\mathbf{w}}(\mathbf{W}_{\text{int}} - z_{\mathbf{w}})s_{\mathbf{x}}(\mathbf{x}_{\text{int}} - z_{\mathbf{x}}) \\ &= s_{\mathbf{w}}s_{\mathbf{x}}\mathbf{W}_{\text{int}}\mathbf{x}_{\text{int}} - s_{\mathbf{w}}z_{\mathbf{w}}s_{\mathbf{x}}\mathbf{x}_{\text{int}} - s_{\mathbf{w}}s_{\mathbf{x}}z_{\mathbf{x}}\mathbf{W}_{\text{int}} + s_{\mathbf{w}}z_{\mathbf{w}}s_{\mathbf{x}}z_{\mathbf{x}}.\end{aligned}$$



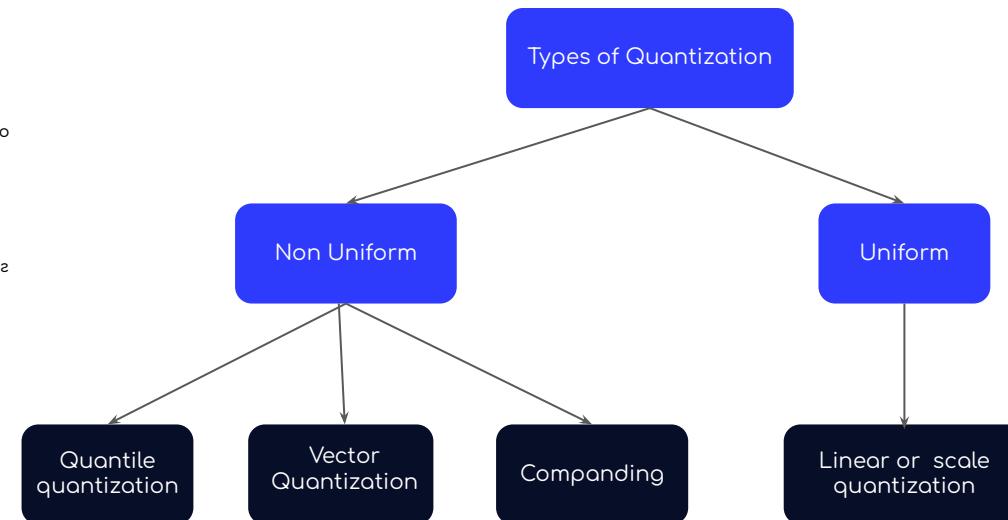
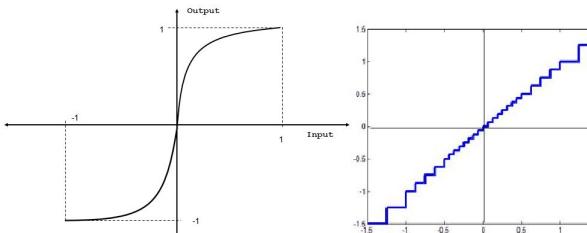
Виды квантования равномерные и неравномерные.

Source: <https://arxiv.org/pdf/2106.08295.pdf>

Если целью является выполнение быстрых вычислений, предпочтительно равномерное квантование.

Неравномерное квантование потребует выполнения дополнительных преобразований перед использованием инструкций микроконтроллера. Эти накладные расходы могут быть значительными и приводить к снижению производительности квантования по сравнению с вычислениями с плавающей запятой. Чтобы получать непостоянный шаг квантования, необходимо вычислить нелинейную функцию, онлайн или офлайн, чтобы создать справочную таблицу, в которой хранятся операнды для каждой операции. Напротив, равномерное квантование основано на постоянном шаге квантования.

Равномерное аффинное квантование: это наиболее часто используемая схема квантования, поскольку она позволяет эффективно реализовать арифметику с фиксированной запятой.

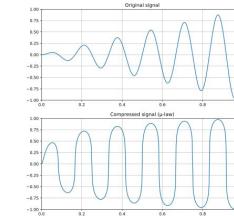


Обычно требуется lookup table

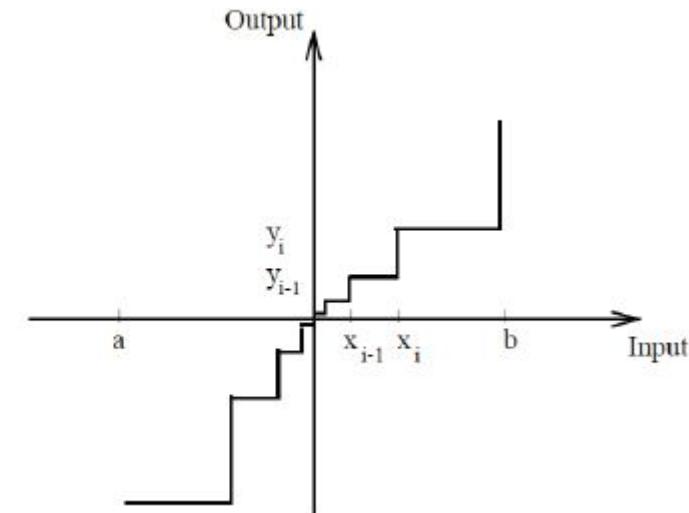
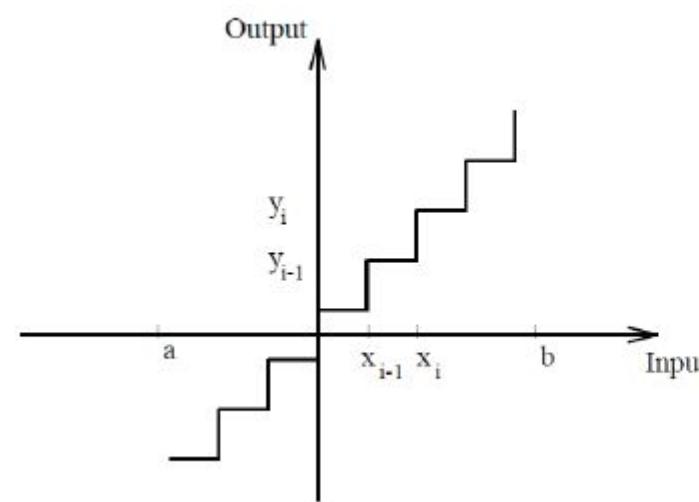
$$g(v) = (f_{\Theta}^{-1} \circ q_b \circ f_{\Theta})(v)$$

Преобразуем данные для более равномерного распределения, используем линейный квантователь, применяем обратное преобразование.

Mu-Law — типичный пример из DSP.

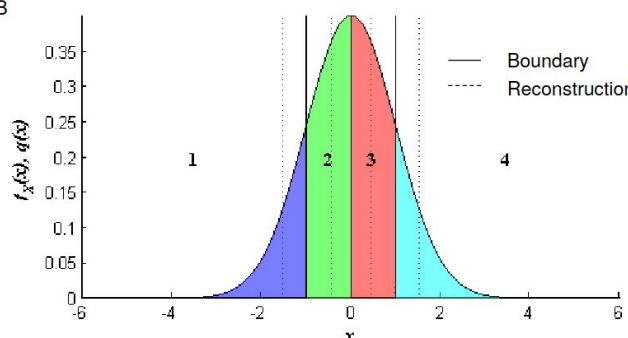


Квантильное квантование



Decision thresholds -0.98, 0, 0.98

Representative levels -1.51, -0.45, 0.45, 1.51

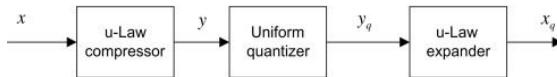
 $D^*=0.12=9.30 \text{ dB}$ 

Преобразуем данные в более равномерное распределение, используем линейный квантователь, применяем обратное преобразование.

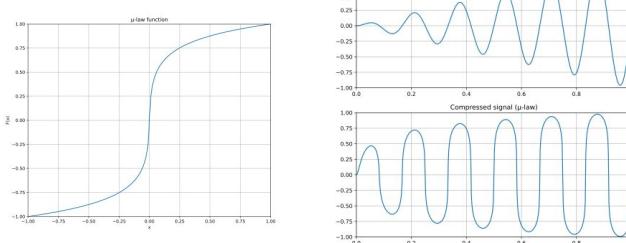
Companding

$$g(v) = (f_{\Theta}^{-1} \circ q_b \circ f_{\Theta})(v)$$

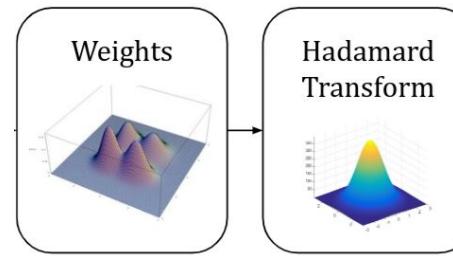
Mu-Law — типичный пример из DSP.



$$F(x) = \text{sgn}(x) \frac{\ln(1 + \mu|x|)}{\ln(1 + \mu)}, \quad -1 \leq x \leq 1,$$



QUIP # - LLM Quantization



Definition 2.1 (Chee et al. (2023)). A Hessian $H \in \mathbb{R}^{n \times n}$ is μ -incoherent if its eigendecomposition $H = Q\Lambda Q^T$ has

$$\max_{i,j} |Q_{ij}| = \max_{i,j} |e_i^T Q e_j| \leq \mu / \sqrt{n}.$$

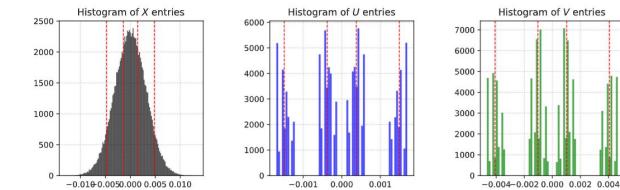
A weight matrix $W \in \mathbb{R}^{m \times n}$ is μ -incoherent if

$$\max_{i,j} |W_{ij}| = \max_{i,j} |e_i^T W e_j| \leq \mu \|W\|_F / \sqrt{mn}.$$

$$U^T(\text{quantized}(\tilde{W})(Vx)) \approx U^T(\tilde{W}(Vx)) = Wx.$$

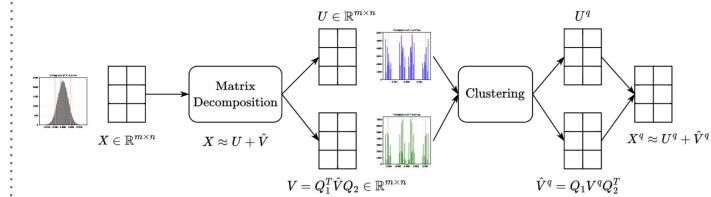
Transform weights to make W more incoherent.

Kashin Representations for LLM Quantization



$$X = U + Q_1 V Q_2^T$$

Q - to present V in some basis, so we need to Q to be easily computed and stored but it is another story.



Подробнее по теме : [Incoherence-Optimal Matrix Completion](#)

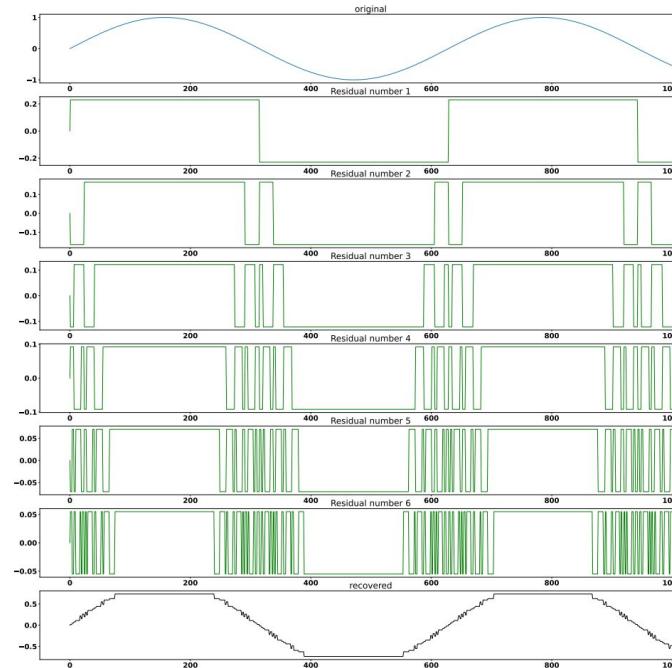
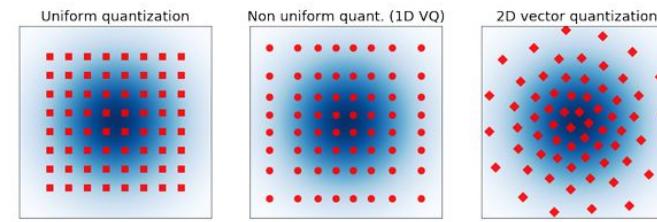
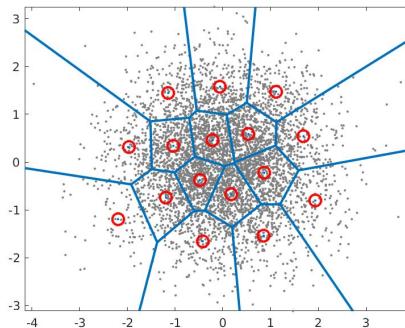
Векторное квантование

VQ было придумано в начале 1980-х годов [Robert M. Gray](#),
рекомендуем его слайды по теме [Fundamentals of Quantization](#)

VQ похож на K-means, но существует много вариантов и разновидностей

Разновидности:

- Additive VQ
- Residual VQ
- Pyramid VQ
- Product VQ
- Tree-structured VQ,
- Lattice VQ
- Classified VQ
- Feedback VQ
- Fuzzy VQ
- etc ...



An example of residual VQ.

Why it is not optimal for this specific signal?

Algorithm 1: Residual Vector Quantization

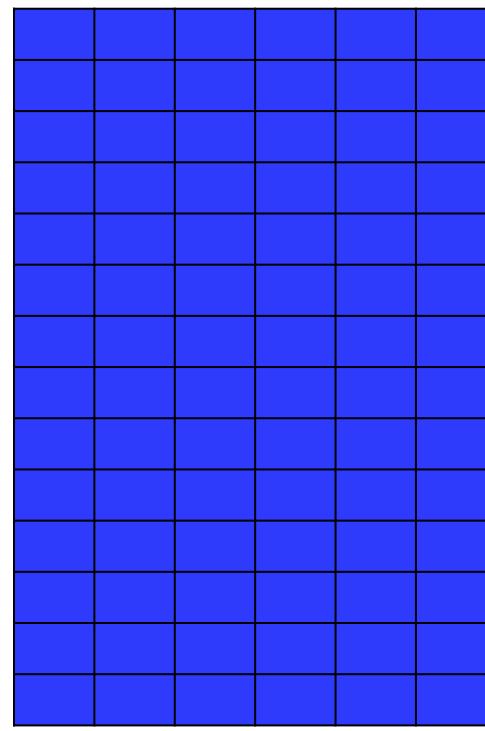
```

Input:  $y = \text{enc}(x)$  the output of the encoder, vector
      quantizers  $Q_i$  for  $i = 1..N_q$ 
Output: the quantized  $\hat{y}$ 
 $\hat{y} \leftarrow 0.0$ 
residual  $\leftarrow y$ 
for  $i = 1$  to  $N_q$  do
     $\hat{y} += Q_i(\text{residual})$ 
    residual  $\leftarrow Q_i(\text{residual})$ 
return  $\hat{y}$ 

```

Векторное квантование

Матрица "W" 6 x 14

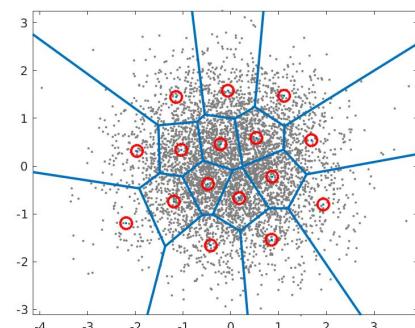


Код бук "С" 6 x 4

Аппроксимируем «W» записями «C».

Теперь каждая строка в «W» представлена как индексный вектор ОНЕ размером 2 бита.

Начальный размер: 6x14 FP.
Сжатый размер: 14 x 2 бит + 6x4 FP

Векторное
квантование

Мы также можем объединить несколько кодов букв.

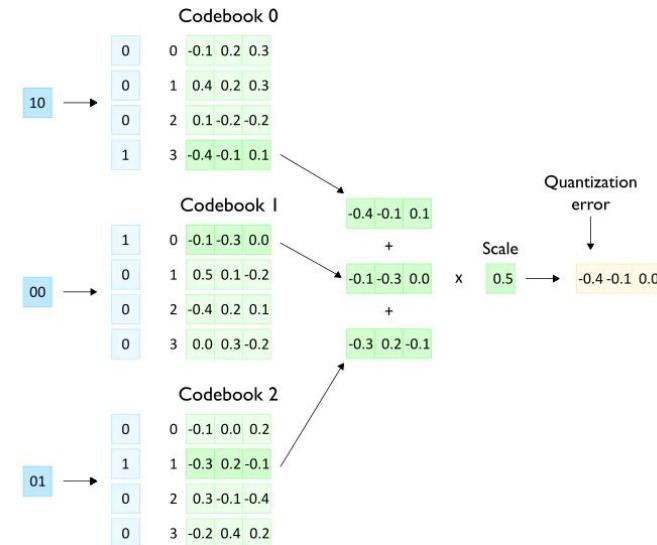
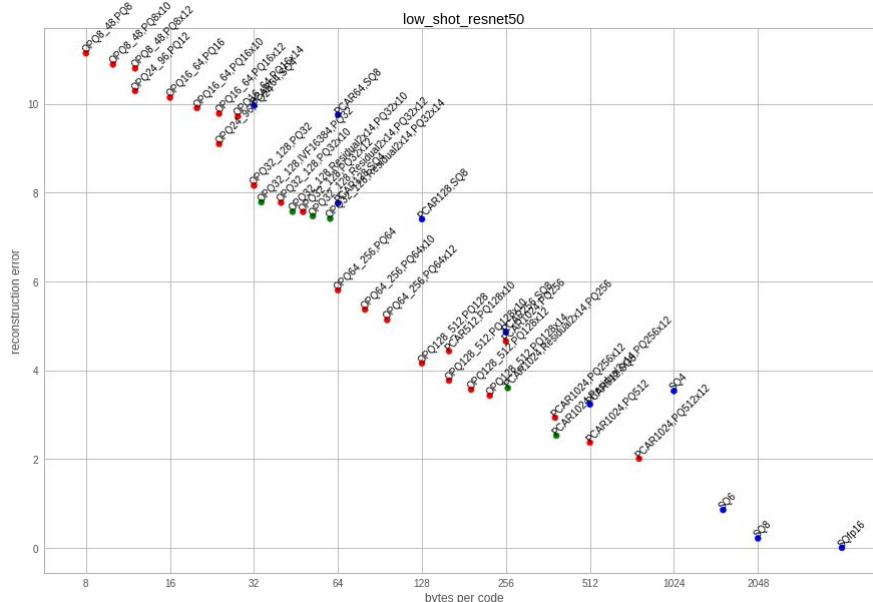


Image source: [link](#)

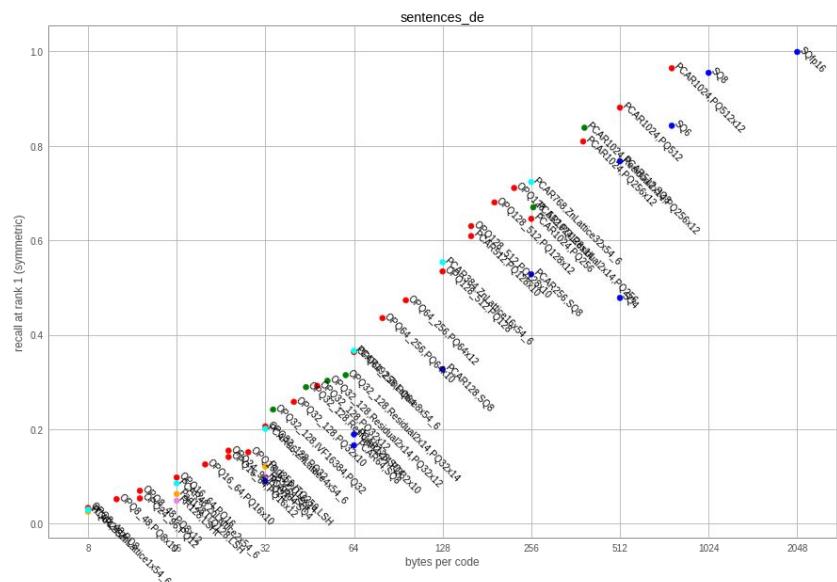
FAISS Vector codec benchmarks

Векторное
квантование

Reconstruction error plot (генеративная модель)



Sentence embeddings



Векторное
квантование

Кажется, нет особого смысла сжимать активации с помощью VQ, не так ли?

>>>

Детализация
квантования

Какие варианты у нас есть при квантовании нейронных сетей?

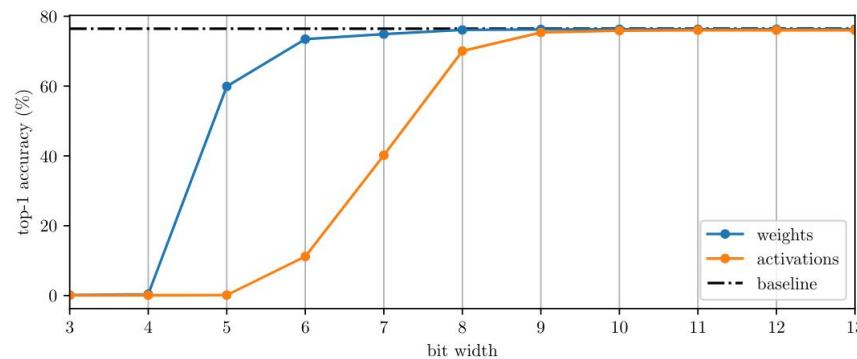
Какие варианты у нас есть при квантовании нейронных сетей?

Веса и активации, одинаково ли они важны?

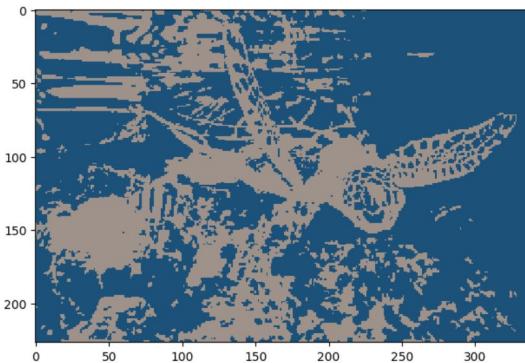


Для быстрой арифметики и
активации, и веса должны иметь один
и тот же тип данных.

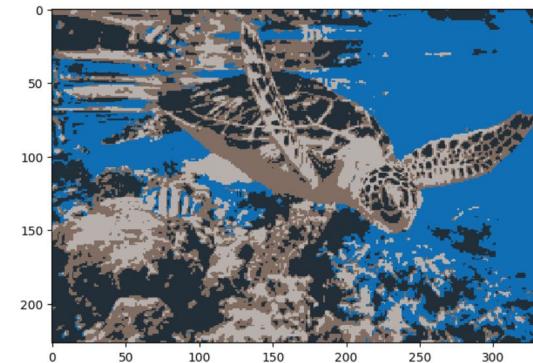
Если только один тип квантуется с
меньшей точностью, то мы только
экономим память - иногда этого
достаточно.



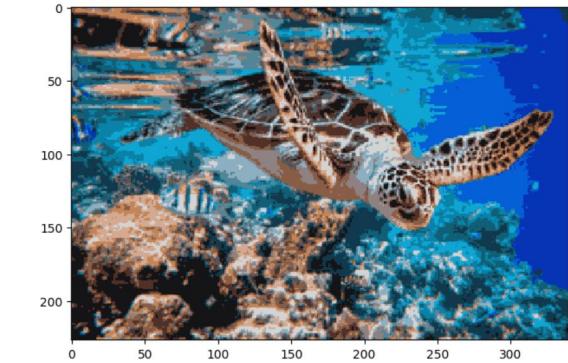
<https://arxiv.org/pdf/2109.04236.pdf>



1 bit - 2 possible values



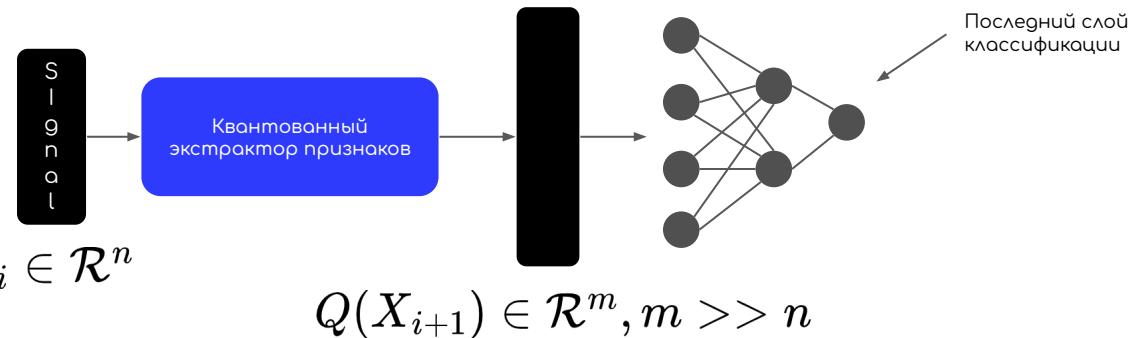
2 bits - 4 possible values



4 bits - 16 possible values

- Удаление информации из сигнала снижает качество, особенно если информация потеряна в начале модели;
- Между тем, снижение точности весов намного меньше ухудшает производительность;
- Можно преобразовать сигнал в более высокое измерение, но уменьшив его точность, чтобы сохранить общее качество (об этом позже).

$$X_i \in \mathcal{R}^n$$



Демонстрация квантования:

- По слоям
- Фильтры / Каналы
- Разбить матрицу на блоки, какие проблемы могут возникнуть тут?

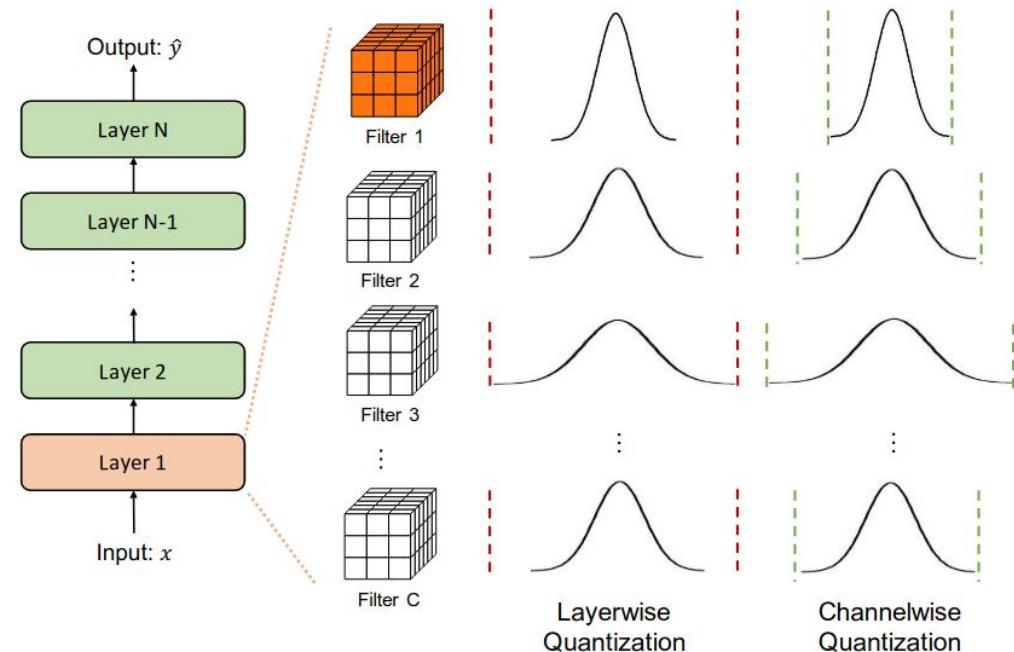


Image: [A Survey of Quantization Methods for Efficient Neural Network Inference](#)

Демонстрация квантования:

- По слоям
- Фильтры / Каналы
- Разбить матрицу на блоки, какие проблемы могут возникнуть тут?

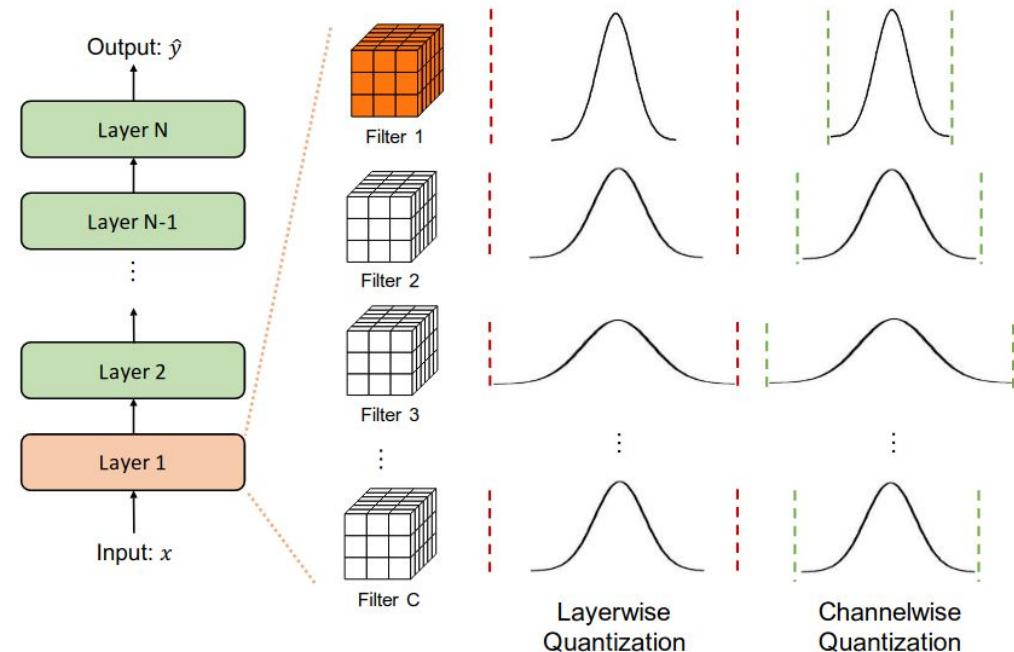


Image: [A Survey of Quantization Methods for Efficient Neural Network Inference](#)

Разбить матрицу весов на несколько блоков

Какое измерение использовать?

Сколько блоков у нас должно быть?

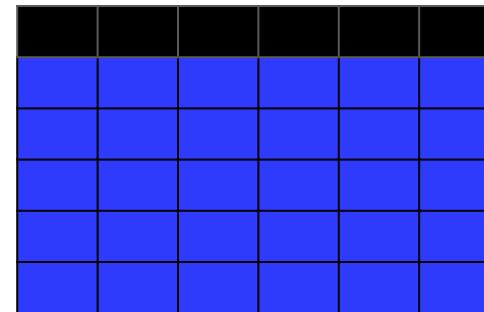
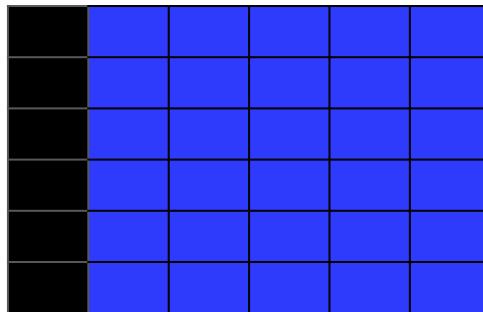
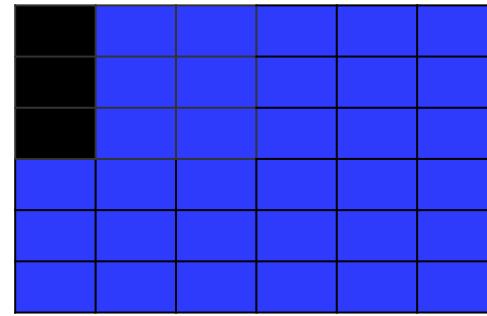


Image: [A Survey of Quantization Methods for Efficient Neural Network Inference](#)

Разбить матрицу весов на несколько блоков

Какое измерение использовать?

- Размерность с более равномерным распределением

Сколько блоков у нас должно быть?

- Учитите, что для каждого блока нужно хранить параметры квантования;

Учитывайте размер кэша L1, L2, обычно он равен 64 (зависит от видеокарты).

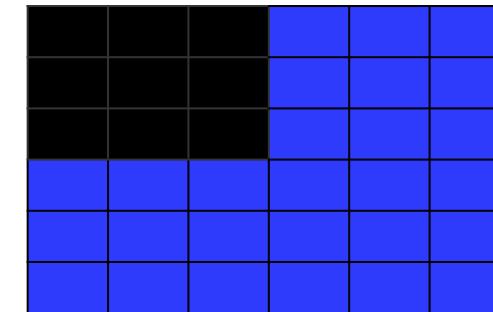
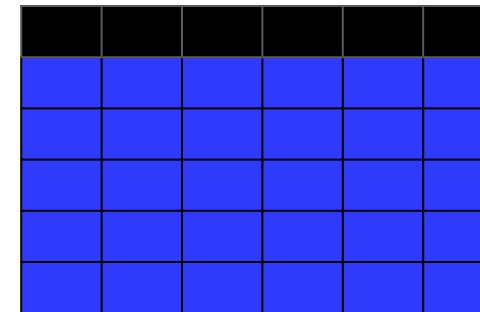
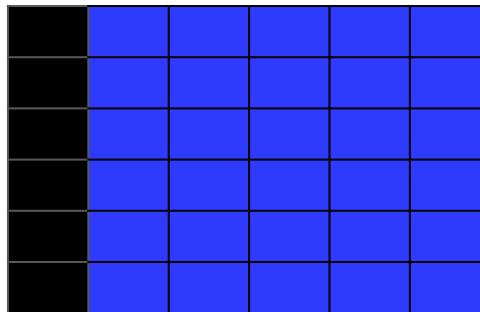
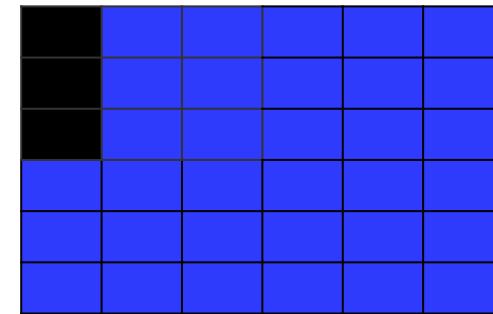


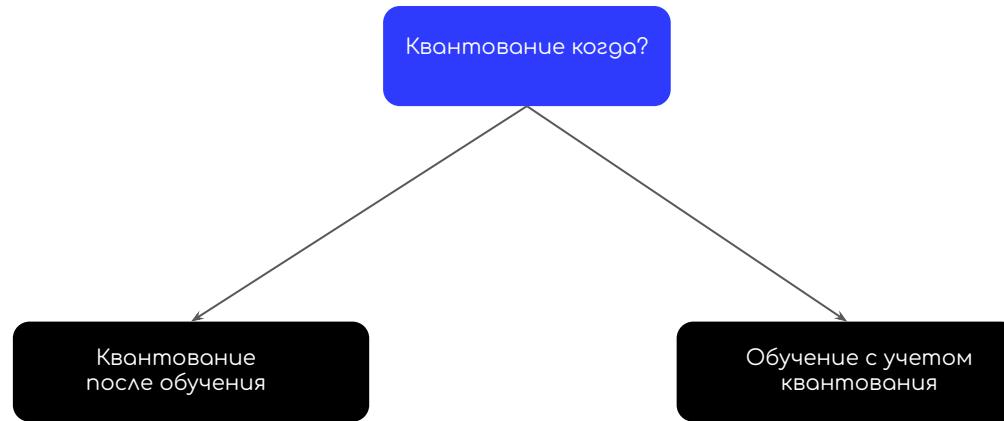
Image: [A Survey of Quantization Methods for Efficient Neural Network Inference](#)

Предпочтительно иметь активации и веса в одном формате данных, иначе мы потратим время на преобразование одного в другой;

Иногда конвертация из одного формата в другой не так уж и сложна, например, 8-битные активации и 4-битные веса;

Большинство современных подходов к квантованию LLM используют FP16 для активаций и 4 бита для весов;

Смешанную точность можно расширить до поблочного квантования, когда каждый блок в матрице может иметь разные уровни квантования (фактически это относится к текущим подходам к квантованию LLM).



Post Training Quantization (PTQ)

ПЛЮСЫ:

- Может применяться для уже обученных моделей
- Не требуется доступ к данным или ко всем данным.

МИНУСЫ:

- Обычно имеет более низкое качество, чем QAT

Quantization Aware Training (QAT)

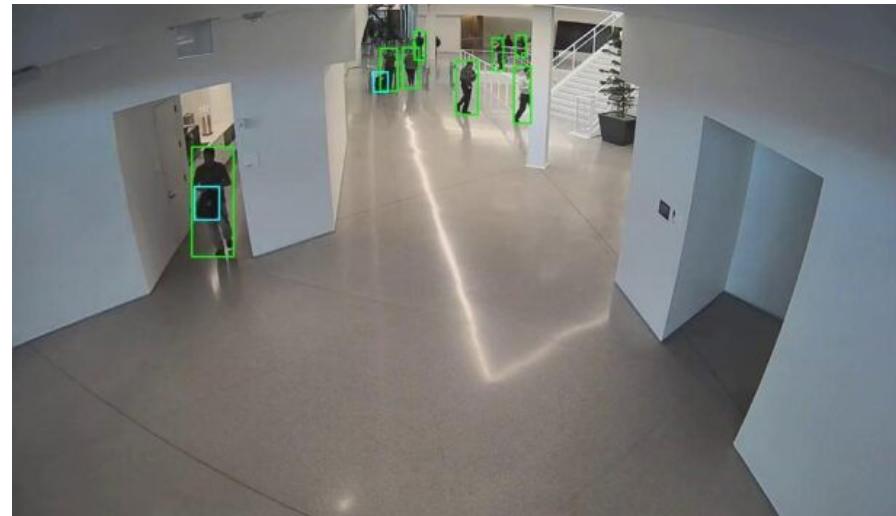
ПЛЮСЫ:

- Качество почти без потерь для некоторых наборов данных

МИНУСЫ:

- Требует аппроксимации неодифференцируемой функции квантования.
- Обычно требуется обучение модели с нуля.
- Нужен доступ к данным

	Baseline FP32 mAP	INT8 mAP with PTQ	INT8 mAP with QAT
PeopleNet-ResNet18	78.37	59.06	78.06
PeopleNet-ResNet34	80.2	62	79.57

[source](#)

>>> STE and Alternatives

Итак, если мы хотим одновременно обучить модель и квантовать ее, что нам следует делать и какие проблемы могут быть?

$$QW = Q(W)$$

Чтобы использовать QAT, нам
нужно вычислить градиенты
функции квантования.

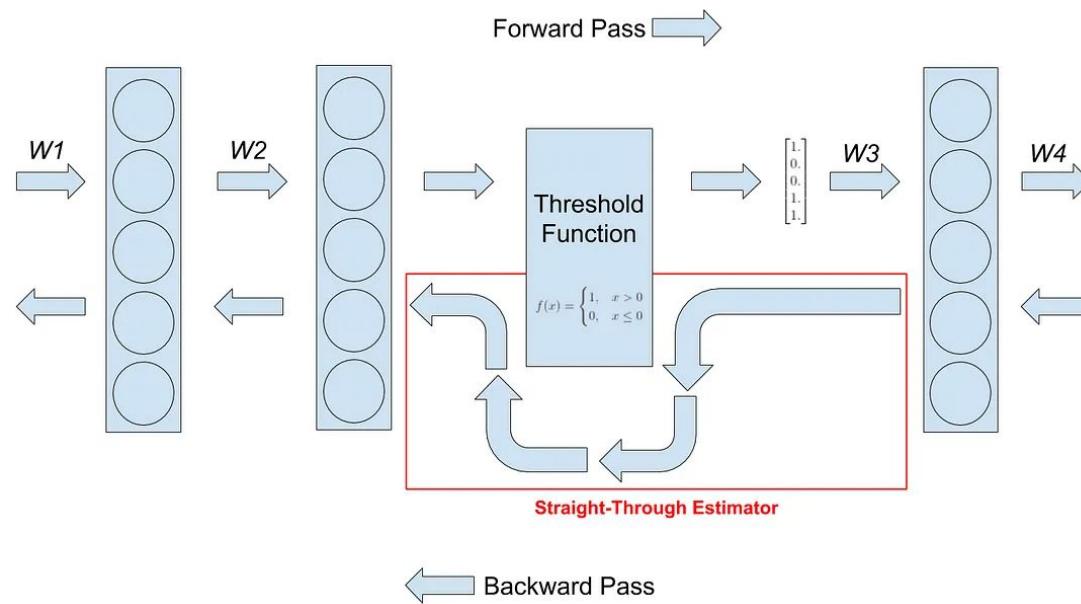


$$y = F(x, QW)$$

$$W := W - \alpha \frac{\partial loss}{\partial QW}$$

Straight Through Estimator (STE)
2013

[Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation](#)



[Image source](#)

$$QW_b = Q(W, b)$$

Используем шум квантования,
чтобы аппроксимировать
деградацию вызванную
квантованием

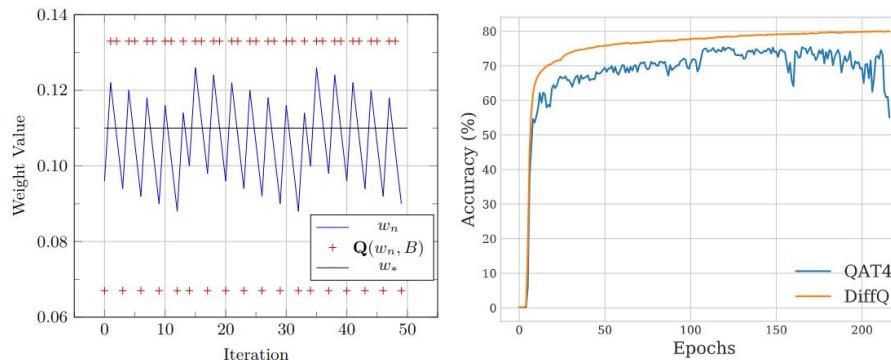
$$QN = QW_b - W \approx \frac{\Delta_b}{2} \mathbf{U}[-1, 1]$$

Дифференцируемый!

$$Q\hat{W}_b = W + \frac{\Delta_b}{2} \mathbf{U}[-1, 1]$$

$$y = F(x, W + \frac{\Delta_b}{2} \mathbf{U}[-1, 1])$$

Проблемы ...



$$QW_b = Q(W, b)$$

$$QN = QW_b - W \approx \frac{\Delta_b}{2} \mathbf{U}[-1, 1]$$

$$\hat{QW}_b = W + \frac{\Delta_b}{2} \mathbf{U}[-1, 1]$$

$$y = F(x, W + \frac{\Delta_b}{2} \mathbf{U}[-1, 1])$$

[Differentiable Model Compression via Pseudo Quantization Noise](#)

Предположения:

1. QN не зависит от W
 2. QN равномерно распределен
 3. QN имеет белый цвет, то есть имеет плоскую спектральную плотность мощности.
- Предположение 1 всегда неверно
 - Предположения 2 и 3 верны только в том случае, если W распределено равномерно.
 - 2 и 3 могут быть разумными, если Δ (шаг квантования) асимптотически мал

$$QW_b = Q(W, b)$$

$$QN = QW_b - W \approx \frac{\Delta_b}{2} \mathbf{U}[-1, 1]$$

$$Q\hat{W}_b = W + \frac{\Delta_b}{2} \mathbf{U}[-1, 1]$$

$$y = F(x, W + \frac{\Delta_b}{2} \mathbf{U}[-1, 1])$$

Source: [M. Gray Quantization introduction](#)

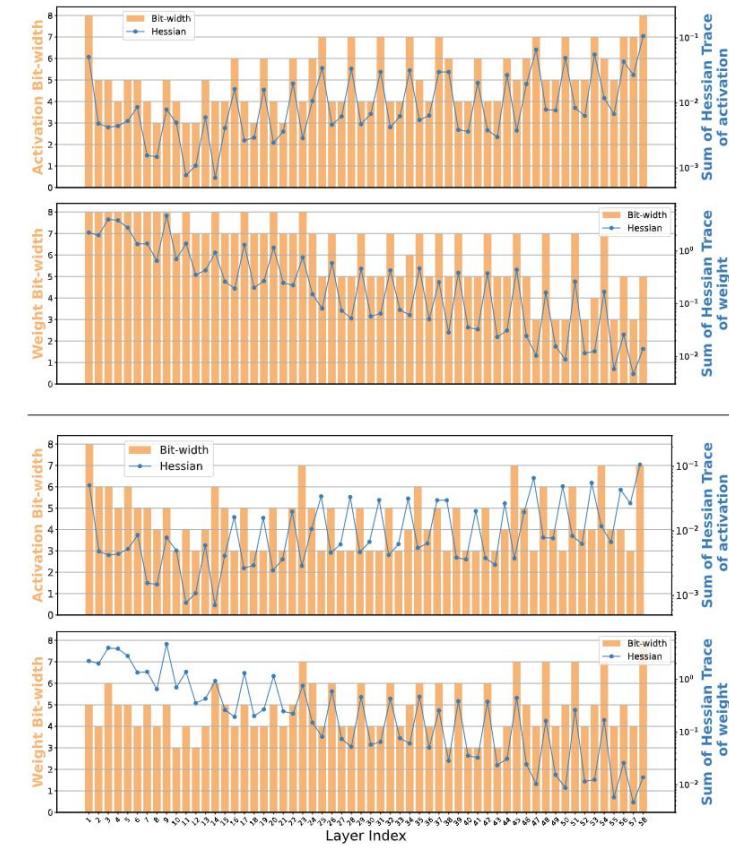
Предположения:

1. QN не зависит от W
 2. QN равномерно распределен
 3. QN имеет белый цвет, то есть имеет плоскую спектральную плотность мощности.
- Предположение 1 всегда неверно
 - Предположения 2 и 3 верны только в том случае, если W распределено равномерно.

2 и 3 могут быть разумными, если Δ (шаг квантования) асимптотически мал

Эмпирические результаты:

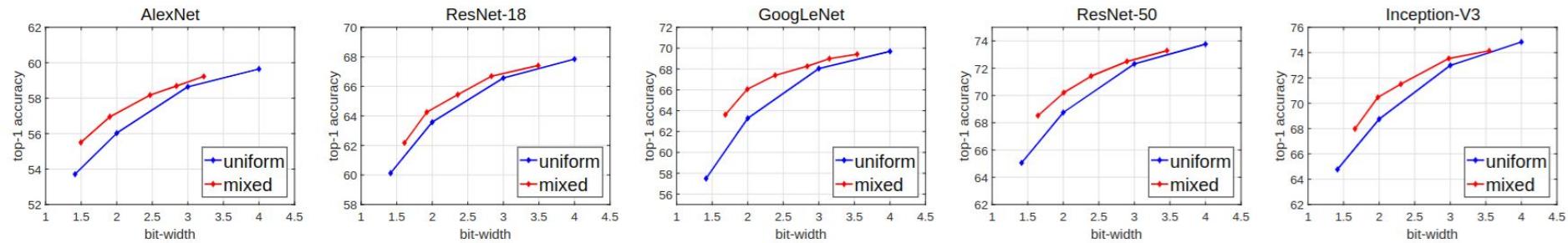
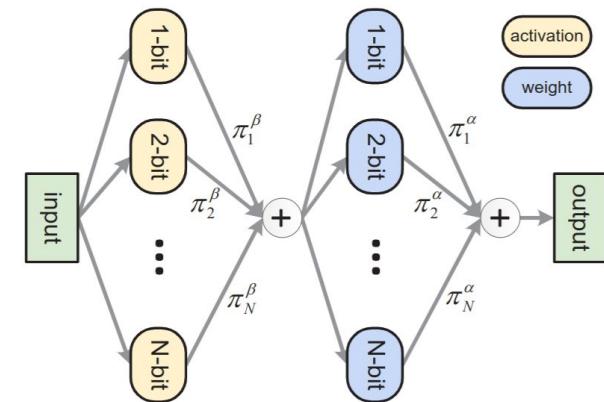
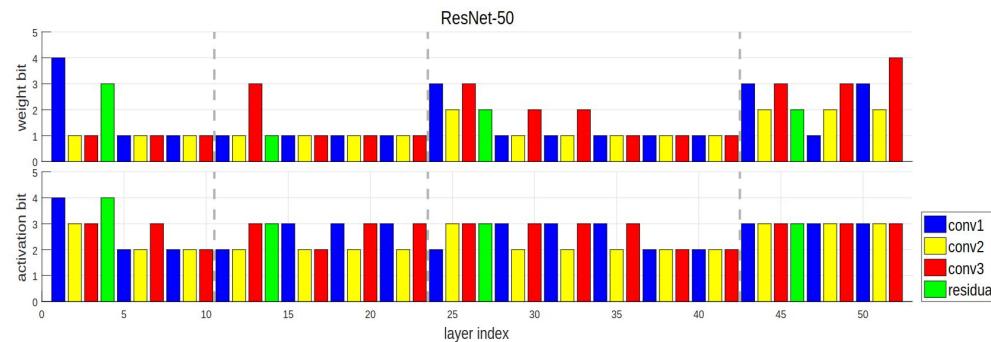
- Аппроксимация шумом работает для 8- и 4,3-битного квантования.



NIPQ: Noise proxy-based Integrated Pseudo-Quantization

>>>

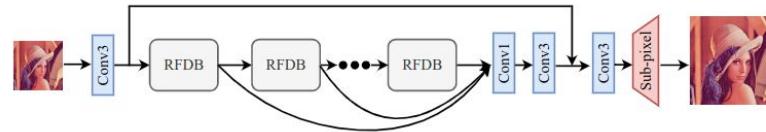
Смешанная
точность



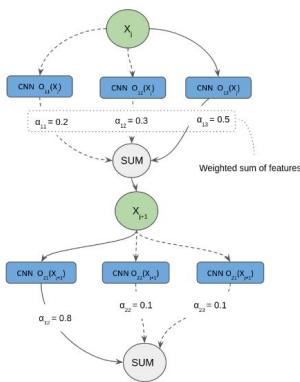
[Rethinking Differentiable Search for Mixed-Precision Neural Networks](#)

>>> Квантование и NAS

Мы берем существующую SR модель — RFDN и оптимизируем ее для квантования.



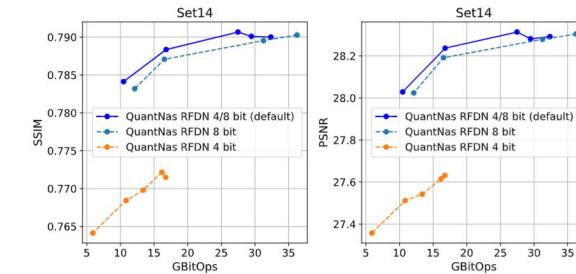
Архитектура RFDN — модель SOTA 2020 года для быстрого SR на устройствах.



Мы заменим каждый сверточный слой в модели блоком слоев с возможностью поиска. Варианты поиска представлены справа.

Кроме того, каждый блок имеет доступные для поиска уровни квантования.

- Head 8 operations: simple 3x3, simple 5x5, growth2 5x5, growth2 3x3, simple 3x3 grouped 3, simple 5x5 grouped 3, simple 1x1 grouped 3, simple 1x1;
- Body 7 operations: simple 3x3, simple 5x5, simple 3x3 grouped 3, simple 5x5 grouped 3, decence 3x3 2, decence 5x5 2, simple 1x1 grouped 3;
- Skip 4 operations: decence 3x3 2, decence 5x5 2, simple 3x3, simple 5x5;
- Upsample 12 operations: conv 5x1 1x5, conv 3x1 1x3, simple 3x3, simple 5x5, growth2 5x5, growth2 3x3, decence 3x3 2, decence 5x5 2, simple 3x3 grouped 3, simple 5x5 grouped 3, simple 1x1 grouped 3, simple 1x1;
- Tail 8 operations: simple 3x3, simple 5x5, growth2 5x5, growth2 3x3, simple 3x3 grouped 3, simple 5x5 grouped 3, simple 1x1 grouped 3, simple 1x1;



Method	Model	GBitOPs	PSNR
LSQ (8-U)	ESPCN	2.3	27.26
HWGQ (8-U)	ESPCN	2.3	27.00
LSQ (4-U)	SRResNet	23.3	27.88
HWGQ (4-U)	SRResNet	23.3	27.42
EdMIPS (4,8)	SRResNet	19.4	27.92
EdMIPS (4,8)	RFDN	20.7	28.13
QuantNAS (4,8)	RFDN	19.3	28.24
QuantNAS (4,8)	RFDN w/o SAN	16.8	28.23
QuantNAS (4,8)	RFDN w/o ADQ	17.2	28.16
QuantNAS (4,8)	Basic	2.6	27.81
QuantNAS (4,8)	Basic w/o SAN	4.1	27.65
QuantNAS (4,8)	Basic w/o ADQ	4.4	27.65

Method	GLOPs	PSNR	Search cost
SRResNet	166.0	28.49	Manual
RFDN	27.14	28.37	Manual
AGD*	140.0	28.40	1.8 GPU days
Trilevel NAS*	33.3	28.26	6 GPU days
QuanNAS_Our SP	29.3	28.22	1.2 GPU days
QuantNAS_RFDN	23.4	28.3	1.6 GPU days

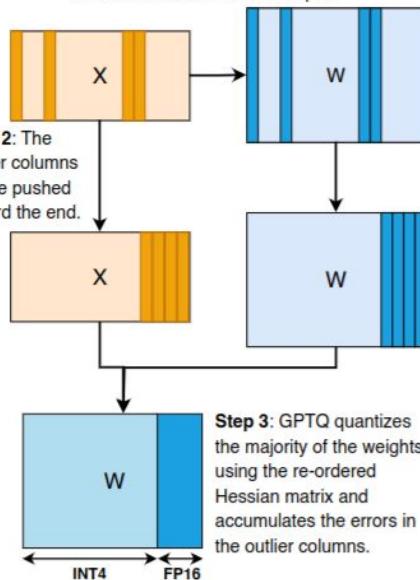
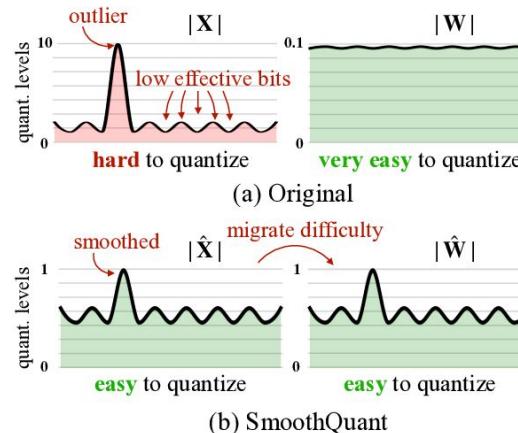
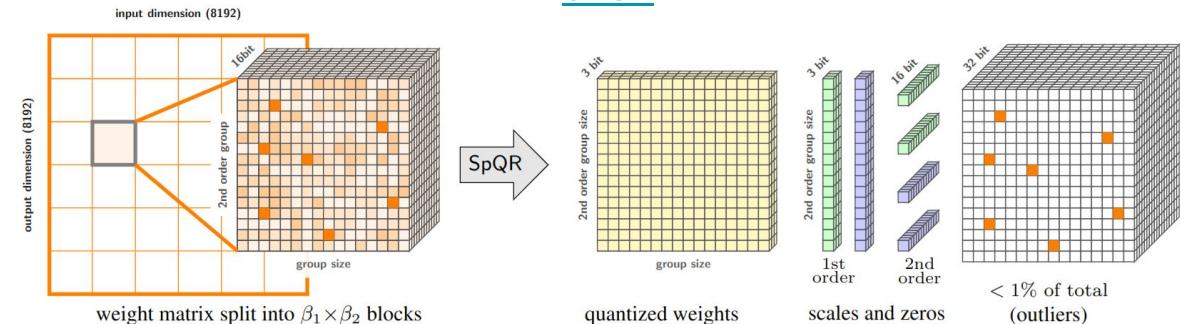
EDMIPS — ищет только блоки квантования, тогда как QuantNAS ищет как уровни квантования, так и блоки, дружественные к квантованию.

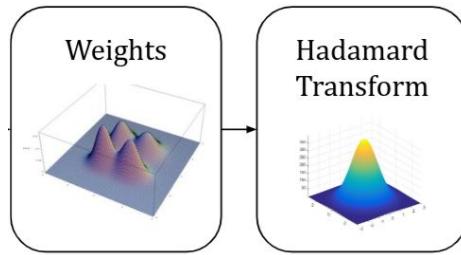
Таким образом, мы получаем лучшую архитектуру с точки зрения FLOP и производительности.

>>> Квантование LLM

QUIK

Step 1: The outlier columns are characterized based on the inputs.

Smooth QuantSPQR

QUIP # - LLM Quantization

Definition 2.1 (Chee et al. (2023)). A Hessian $H \in \mathbb{R}^{n \times n}$ is μ -incoherent if its eigendecomposition $H = Q\Lambda Q^T$ has

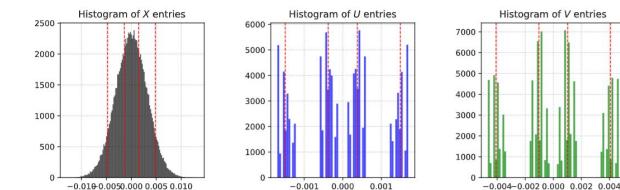
$$\max_{i,j} |Q_{ij}| = \max_{i,j} |e_i^T Q e_j| \leq \mu / \sqrt{n}.$$

A weight matrix $W \in \mathbb{R}^{m \times n}$ is μ -incoherent if

$$\max_{i,j} |W_{ij}| = \max_{i,j} |e_i^T W e_j| \leq \mu \|W\|_F / \sqrt{mn}.$$

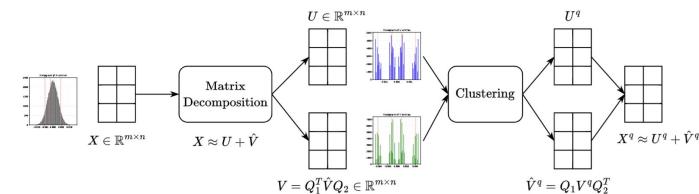
$$U^T(\text{quantized}(\tilde{W})(Vx)) \approx U^T(\tilde{W}(Vx)) = Wx.$$

Transform weights to make W more incoherent.

Kashin Representations for LLM Quantization

$$X = U + Q_1 V Q_2^T$$

Q - to present V in some basis, so we need to Q to be easily computed and stored but it is another story.



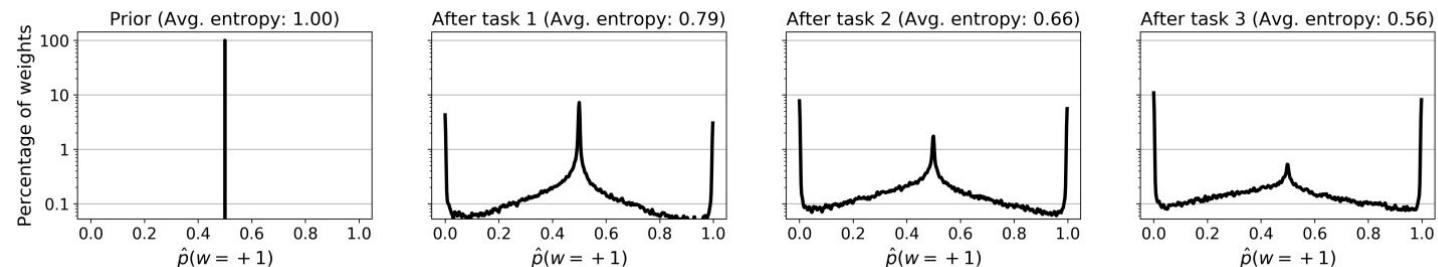
>>>

Бинаризация

Бинарные веса и активации – насколько это может быть хорошо?

Смамъу	Год	ImageNet TOP1
XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks	2016	51.2
Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit?	2018	61.0
Structured Binary Neural Networks for Image Recognition (GroupNet)	2019	65.2
Widening and Squeezing: Towards Accurate and Efficient QNNs	2020	69.18
ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions	2020	71.4
HIGH-CAPACITY EXPERT BINARY NETWORKS	2021	71.1

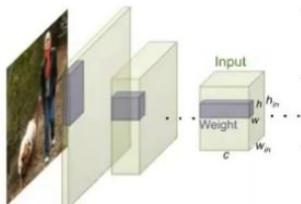
	STE	Our BayesBiNN method	Bop
Step 1: Get w_b from w_r	$w_b \leftarrow \text{sign}(w_r)$	$w_b \leftarrow \tanh((w_r + \delta)/\tau)$	$w_b \leftarrow \text{hyst}(w_r, w_b, \gamma)$
Step 2: Compute gradient at w_b	$\mathbf{g} \leftarrow \nabla_{w_b} \ell(y, f_{w_b}(\mathbf{x}))$	$\mathbf{g} \leftarrow \nabla_{w_b} \ell(y, f_{w_b}(\mathbf{x}))$	$\mathbf{g} \leftarrow \nabla_{w_b} \ell(y, f_{w_b}(\mathbf{x}))$
Step 3: Update w_r	$w_r \leftarrow w_r - \alpha \mathbf{g}$	$w_r \leftarrow (1 - \alpha)w_r - \alpha \mathbf{s} \odot \mathbf{g}$	$w_r \leftarrow (1 - \alpha)w_r - \alpha \mathbf{g}$



[1] [Latent Weights Do Not Exist: Rethinking Binarized Neural Network Optimization](#)

[2] [Training Binary Neural Networks using the Bayesian Learning Rule](#)

XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks 2016



- virtual reality
- augmented reality
- smart wearable devices

$$I \in \mathbb{R}^{c \times w_{in} \times h_{in}}$$

$$W \in \mathbb{R}^{c \times w \times h}, w \leq w_{in}, h \leq h_{in}$$

$$W \approx \alpha B$$

$$B \in \{-1, +1\}^{c \times w \times h} \rightarrow \text{binary filter}$$

$$\alpha > 0 \rightarrow \text{scaling factor}$$

$$I * W \approx (I \oplus B)\alpha$$

convolution without any multiplications

$$W, B \in R^n \text{ where } n = cwh$$

$$J(B, \alpha) = \|W - \alpha B\|^2 = \alpha^2 \underbrace{B^T B}_{=n \text{ since } B \in \{-1, +1\}^n} - 2\alpha W^T B + \underbrace{W^T W}_{:=c \text{ is a constant since } W \text{ is known}}$$

$$B^*, \alpha^* = \arg \min_{B, \alpha} J(B, \alpha)$$

	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs $0.11 - 0.21 \dots -0.34$ $-0.25 0.61 \dots$	Real-Value Weights $0.12 - 0.2 \dots 0.41$ $-0.2 0.5 \dots 0.68$	$+, -, \times$	1x	1x
Binary Weight	Real-Value Inputs $0.11 - 0.21 \dots -0.34$ $-0.25 0.61 \dots 0.52$	Binary Weights $1 - 1 \dots -1$ $-1 1 \dots 1$	$+, -$	$\sim 32x$	$\sim 2x$
BinaryWeight Binary Input (XNOR-Net)	Binary Inputs $1 - 1 \dots -1$ $-1 1 \dots 1$	Binary Weights $1 - 1 \dots -1$ $-1 1 \dots 1$	XNOR, bitcount	$\sim 32x$	$\sim 58x$

$$B^* = \arg \max_B W^T B \text{ s.t. } B \in \{+1, -1\}^n$$

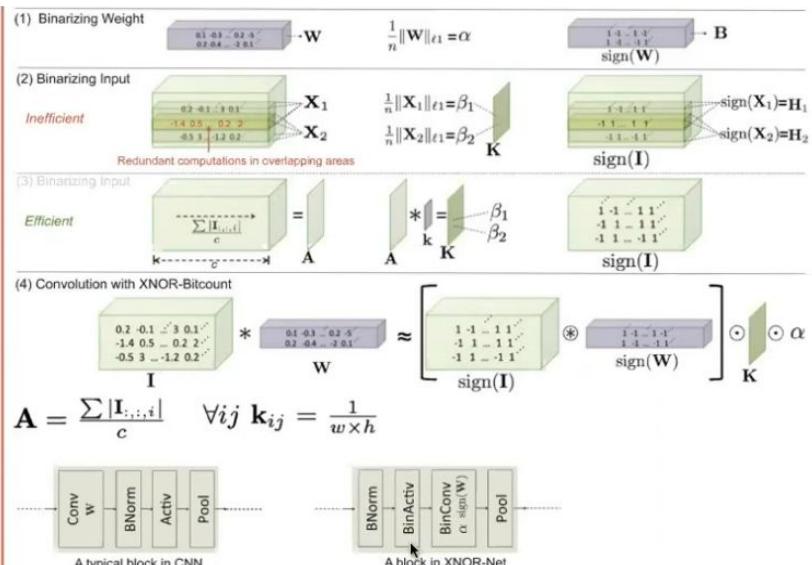
$$B_i = +1 \text{ if } W_i \geq 0 \\ B_i = -1 \text{ if } W_i < 0 \implies B^* = \text{sign}(W)$$

$$\alpha^* = \frac{W^T B}{n} = \frac{W^T \text{sign}(W)}{n} = \frac{\sum_i |W_i|}{n} = \frac{1}{n} \|W\|_1$$

$$\text{Given } W_t \\ \alpha_t = \frac{1}{n} \|W_t\|_1$$

$$B_t = \text{sign}(W_t)$$

$$\widetilde{W}_t = \alpha_t B_t \quad W_{t+1} = W_t - \eta \nabla_{\widetilde{W}_t} C$$



Widening and Squeezing: Towards Accurate and Efficient QNNs 2020

We have a model with weights W where W has N channels and we want to find a projection matrix P so WP^T has K channels and $WP^T = sign(WP^T)$

$$\min_{PB} \frac{1}{2} \|WP^T - sign(WP^T)\|_F^2 + \frac{\gamma}{2} \|D(W) - D(sign(WP^T))\|_F^2 \quad (1)$$

D - square pairwise Euclidean distance between features for all samples in the dataset, it is difficult to compute it but the expression above can be simplified. If P is orthogonal distance can be preserved.

$$\min_{PB} \frac{1}{2} \|WP^T - sign(WP^T)\|_F^2 + \frac{\gamma}{2} \|P^T P - I\|_F^2 \quad (2)$$

Keep number of channels small, M is a filter wise vector

$$\min_{PB} \frac{1}{2} \|WP^T - M \circ sign(WP^T)\|_F^2 + \frac{\gamma}{2} \|P^T P - I\|_F^2 + \beta \|M\|_1 \quad (3)$$

Layer num	orig	found	change
2	128	410	3.2
3	256	332	1.3
4	256	614	2.3
5	512	420	0.8
6	512	25	0.05
acc	0.93	0.92	

VGG small / CIFAR 100

1. It suggests we may need more filters at first layers
2. Usually first FP32 layers if fixed but probably we need more filters there too !?
3. The first filter is the most computationally heavy

TODO: check this assumption

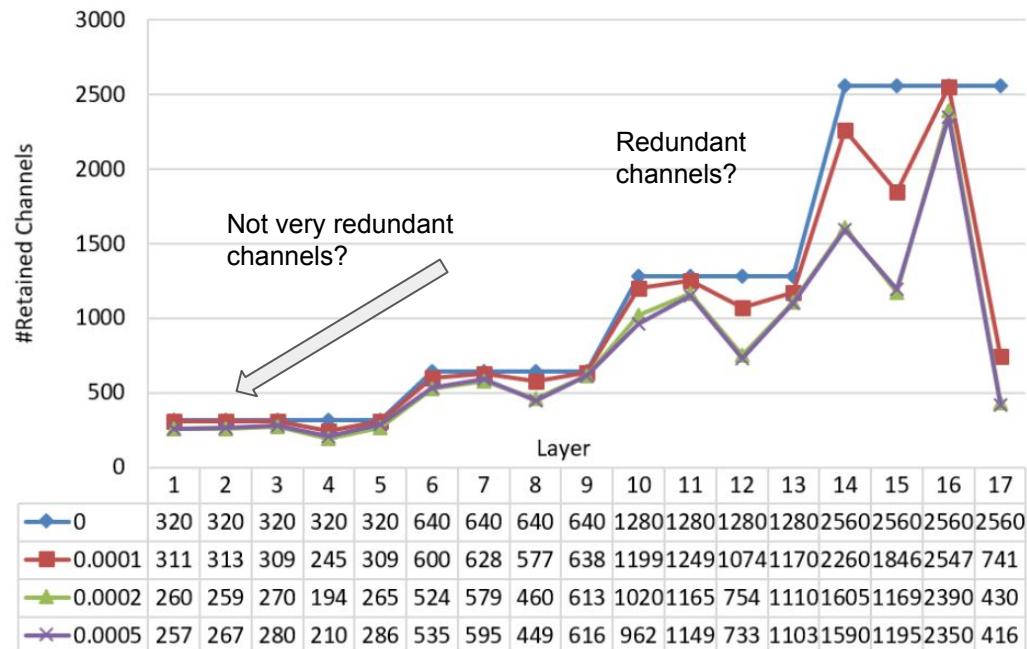
Widening and Squeezing: Towards Accurate and Efficient QNNs 2020

Increase number of channels 4 times and prune

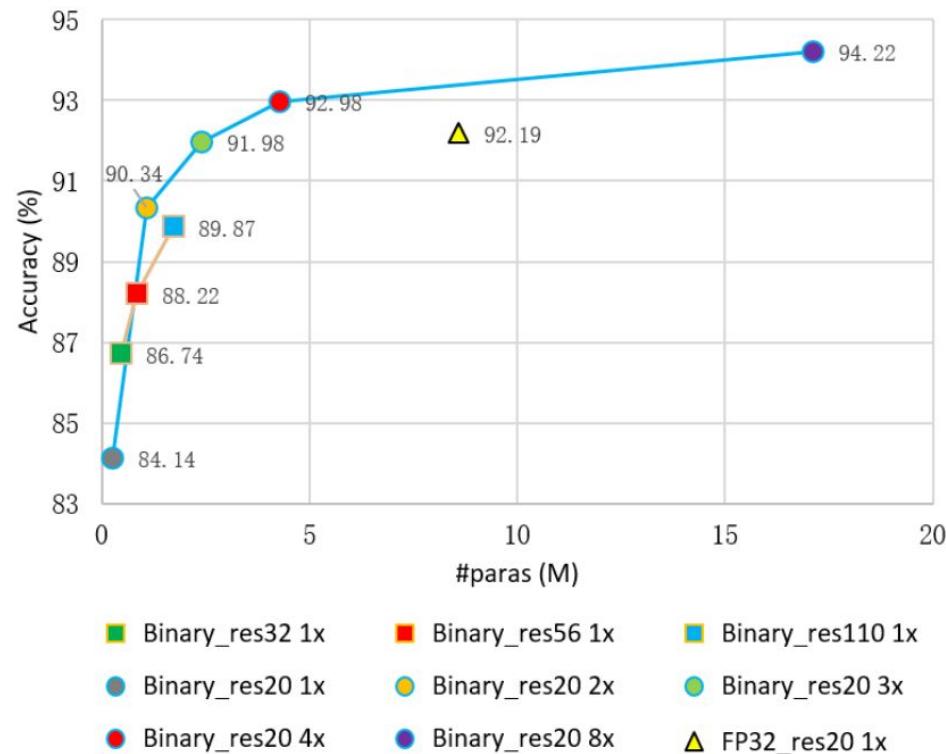
<i>reg</i>	<i>top1_O</i>	<i>ratio</i>	<i>top1_P</i>	<i>top1_R</i>
0.0001	68.34	17.95	67.55	68.32
0.0002	65.27	33.06	60.84	69.19
0.0005	65.10	33.44	58.93	69.18

Table 9. Results of different batch norm scale regularization factors on ImageNet. *reg* means the value of scale regularization factor, *top1_O* and *top1_P* mean the accuracy before and after pruning. *ratio* is the ratio of pruned channels, *top1_R* is the accuracy that we retrain the pruned network from scratch.

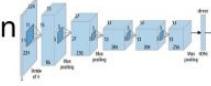
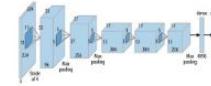
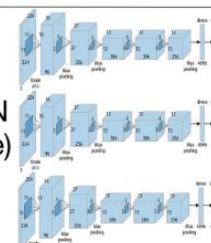
*Authors use VGG here the network is overparameterized and it is easier to train it compared to small Resnets.



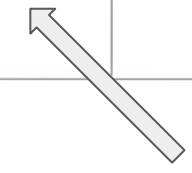
Widening and Squeezing: Towards Accurate and Efficient QNNs 2020



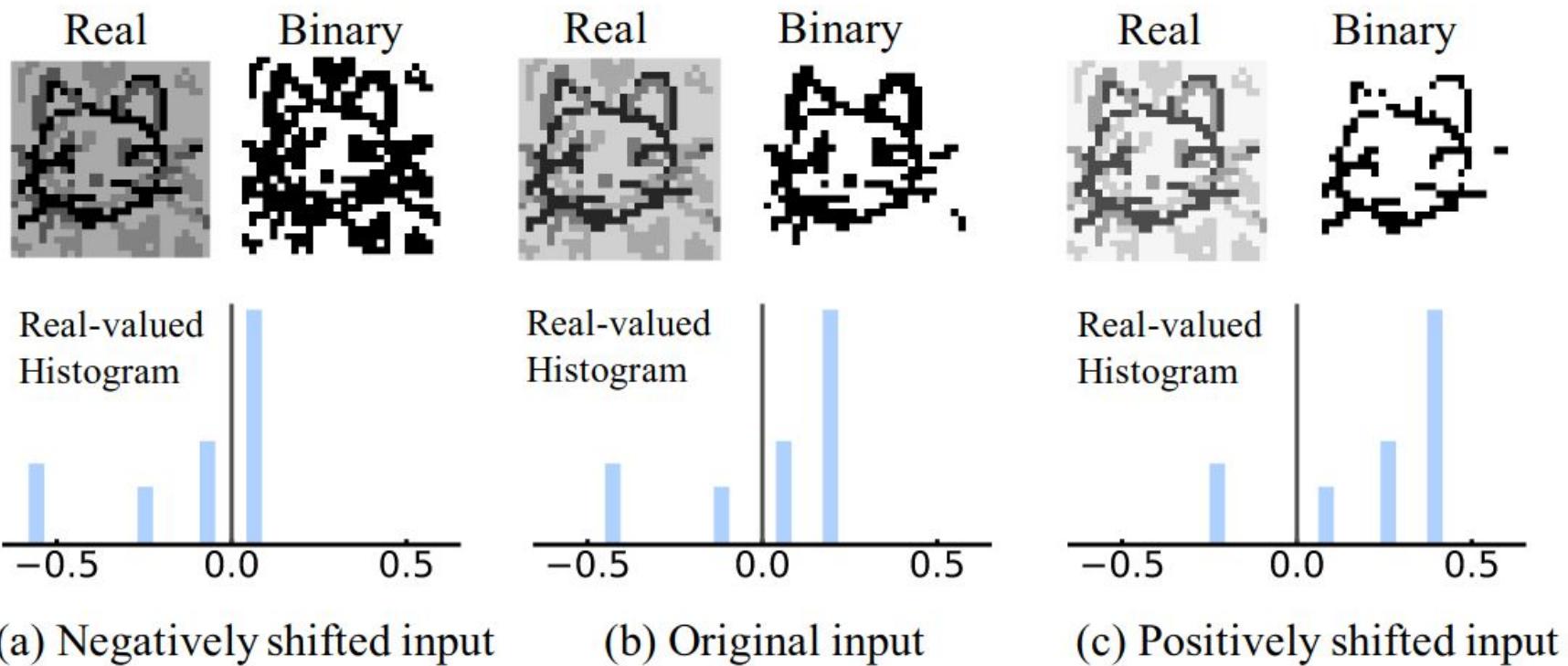
Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit? 2018

Network		Numerics	Operations	Memory Speed-Up	Inference Speed-Up	Accuracy
 Input Image	Full-Precision DNN	 W: 32 bit A: 32 bit	+, -, \times	1x	1x	High
	BNN	 W: 1 bit A: 1 bit	XNOR, Bitcount	~32x	(CPU) ~58x	Low
	(Ours) BENN (K ensemble)	 W: 1 bit A: 1 bit	XNOR, Bitcount	$\sim(32/K)x$	(CPU) ~58x	High

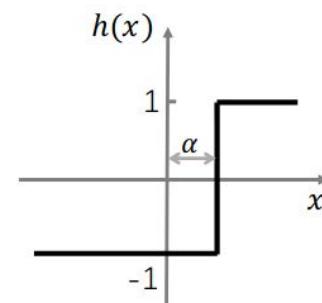
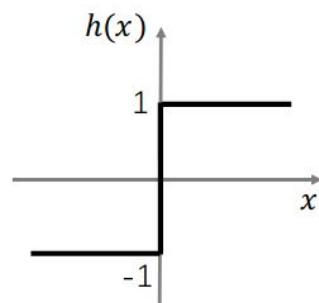
 Memory savings

 Inference speed up

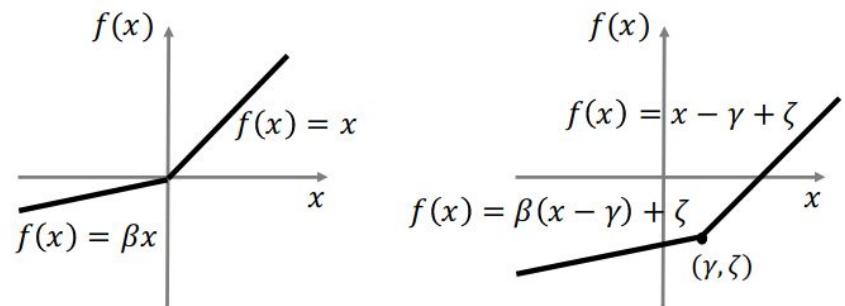
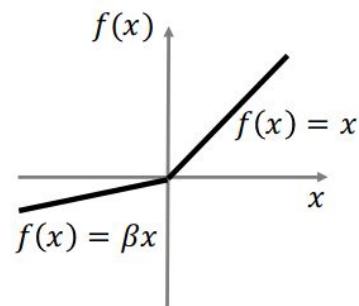
ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions 2020



ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions 2020



(a) Sign vs. RSign



(b) PReLU vs. RPReLU

Fig. 4. Proposed activation functions, RSign and RPReLU, with learnable coefficients and the traditional activation functions, Sign and PReLU.