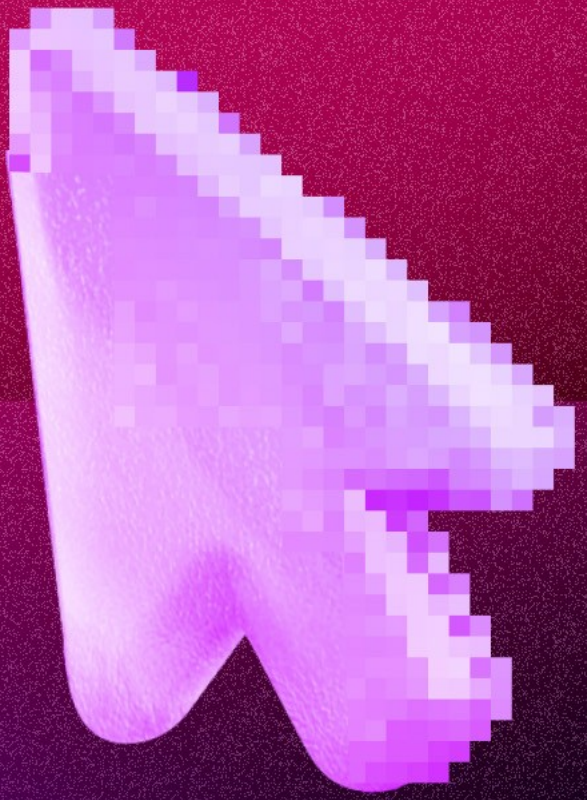
ЧТО ИСПОЛЬЗУЕМ

- Шрифт для заголовков – скачать
- Не в заголовках используем – Lab Grotesque K Regular (Основной текст)
- Основные цвета – ff007f 9700ff
- Дополнительные цвета темы можем использовать для графиков и сценариев, где нужно показать сравнение с 2-мя и более цветами

ЗАЧЕМ В ML НУЖНЫ ПРОИЗВОДНАЯ, GPU И АЛГОРИТМЫ



Александр Панкратов

Разработчик машинного обучения

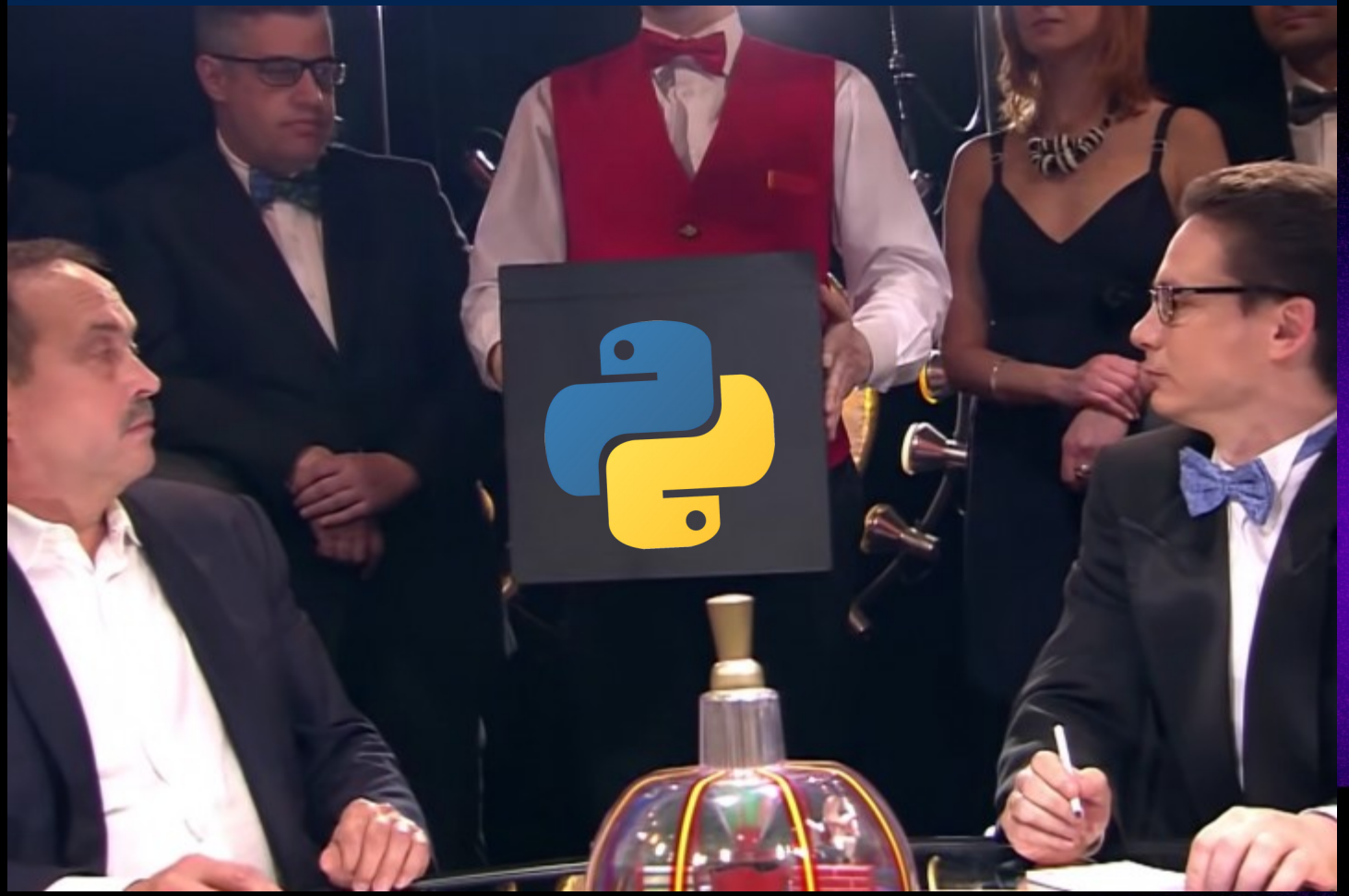
Что обсудим

- Для чего в точности нам нужны GPU, и не лучше ли Контуру купить чугунный мост вместо GPU-серверов?
- Что именно нужно проделать с производной, чтобы нейросеть обучалась.
- Где и зачем в МЛ встречаются классические алгоритмы.

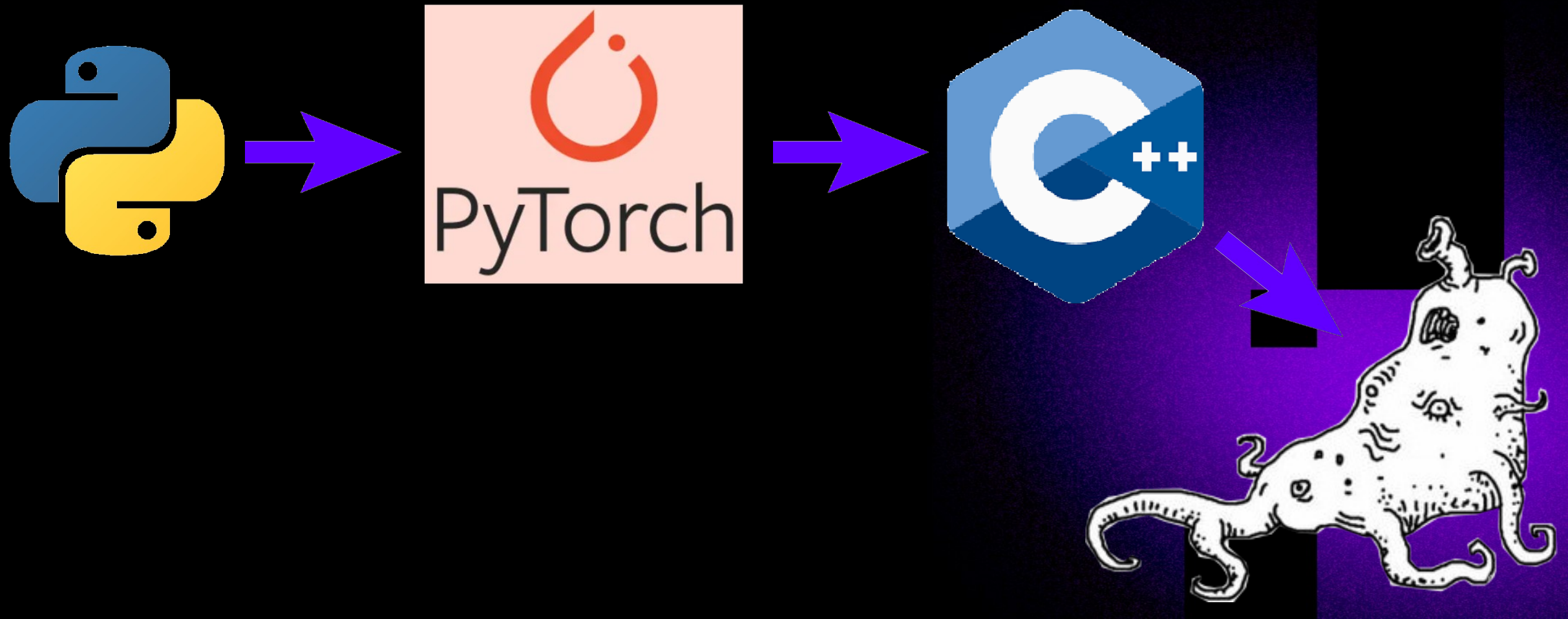
ЗАЧЕМ НЕЙРОНКАМ GPU

ЧТО ВНУТРИ НЕЙРОСЕТИ

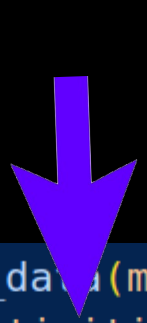
```python
asr_model = nemo_asr.models.EncDecCTCModelBPE.from_pretrained(
    model_name="stt_ru_conformer_ctc_large",
)
```

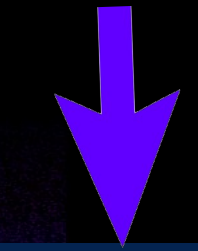


ЧТО ВНУТРИ НЕЙРОСЕТИ



СКАЛЬПЕЛЬ И МИКРОСКОП





```
asr_model, data = get_model_and_data(model_name="stt_ru_conformer_ctc_large", device='cuda')
with profile(with_stack=True, activities=[ProfilerActivity.CPU, ProfilerActivity.CUDA]) as prof:
    for processed_signal, processed_signal_length in data:
        asr_model.encoder(audio_signal=processed_signal, length=processed_signal_length)
with open("etc/profiler_outputs/gpu_forward_profiling", mode="w") as out:
    out.write(
        prof.key_averages().table(
            sort_by="self_cuda_time_total",
            max_name_column_width=4096,
            max_shapes_column_width=4096,
        )
    )
```

ЧТО ПРОИСХОДИТ КОГДА НЕТ GPU

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::addmm	45.62%	2.335s	48.04%	2.458s	264.888us	9280
aten::mkldnn_convolution	23.35%	1.195s	23.67%	1.211s	362.648us	3340
aten::copy_	4.78%	244.852ms	4.78%	244.852ms	11.193us	21876
aten::mm	3.36%	171.688ms	3.36%	171.767ms	149.103us	1152
aten::bmm	3.11%	158.910ms	3.11%	159.221ms	46.071us	3456
aten::add	2.82%	144.425ms	2.84%	145.091ms	17.439us	8320
aten::silu	2.08%	106.633ms	2.08%	106.633ms	30.855us	3456
aten::native_layer_norm	1.64%	83.711ms	1.98%	101.436ms	17.610us	5760
aten::_slow_conv2d_forward	1.07%	54.978ms	1.22%	62.574ms	256.452us	244

ЧТО ПРОИСХОДИТ КОГДА ЕСТЬ GPU

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	# of Calls
aten::addmm	45.62%	2.335s	48.04%	2.458s	264.888us	9280
aten::mkldnn_convolution	23.35%	1.195s	23.67%	1.211s	362.648us	3340
aten::copy_	4.78%	244.852ms	4.78%	244.852ms	11.193us	21876
aten::mm	3.36%	171.688ms	3.36%	171.767ms	149.103us	1152
aten::bmm	3.11%	158.910ms	3.11%	159.221ms	46.071us	3456

Name	Self CPU %	Self CPU	CPU time avg	Self CUDA	Self CUDA %	CUDA time avg	# of Calls
aten::addmm	10.54%	148.755ms	25.925us	146.863ms	42.41%	15.826us	9280
ampere_sgemm_64x32_sliced1x4_tn	0.00%	0.000us	0.000us	93.975ms	27.14%	13.448us	6988
aten::cudnn_convolution	3.75%	52.891ms	33.350us	51.405ms	14.84%	21.137us	2432
aten::bmm	3.38%	47.668ms	18.970us	40.111ms	11.58%	11.606us	3456
ampere_sgemm_128x128_tn	0.00%	0.000us	0.000us	25.050ms	7.23%	10.872us	2304
ampere_sgemm_128x32_tn	0.00%	0.000us	0.000us	21.646ms	6.25%	22.269us	972
ampere_sgemm_32x32_sliced1x4_tn	0.00%	0.000us	0.000us	20.528ms	5.93%	10.093us	2034

2235 / (148.755 + 146.863) ≈ 7.56

1195 / (52.891 + 51.405) ≈ 11.46


почему

КТО ТАКОЙ ADDMM?

torch.addmm

torch.addmm(*input*, *mat1*, *mat2*, ***, *beta=1*, *alpha=1*, *out=None*) → Tensor

Performs a matrix multiplication of the matrices `mat1` and `mat2`. The matrix `input` is added to the final result.

If `mat1` is a $(n \times m)$ tensor, `mat2` is a $(m \times p)$ tensor, then `input` must be broadcastable with a $(n \times p)$ tensor and `out` will be a $(n \times p)$ tensor.

`alpha` and `beta` are scaling factors on matrix-vector product between `mat1` and `mat2` and the added matrix `input` respectively.

$$\text{out} = \beta\ \text{input} + \alpha\ (\text{mat1}_i \mathbin{@} \text{mat2}_i)$$

КТО ТАКОЙ CONVOLUTION?

Conv1d

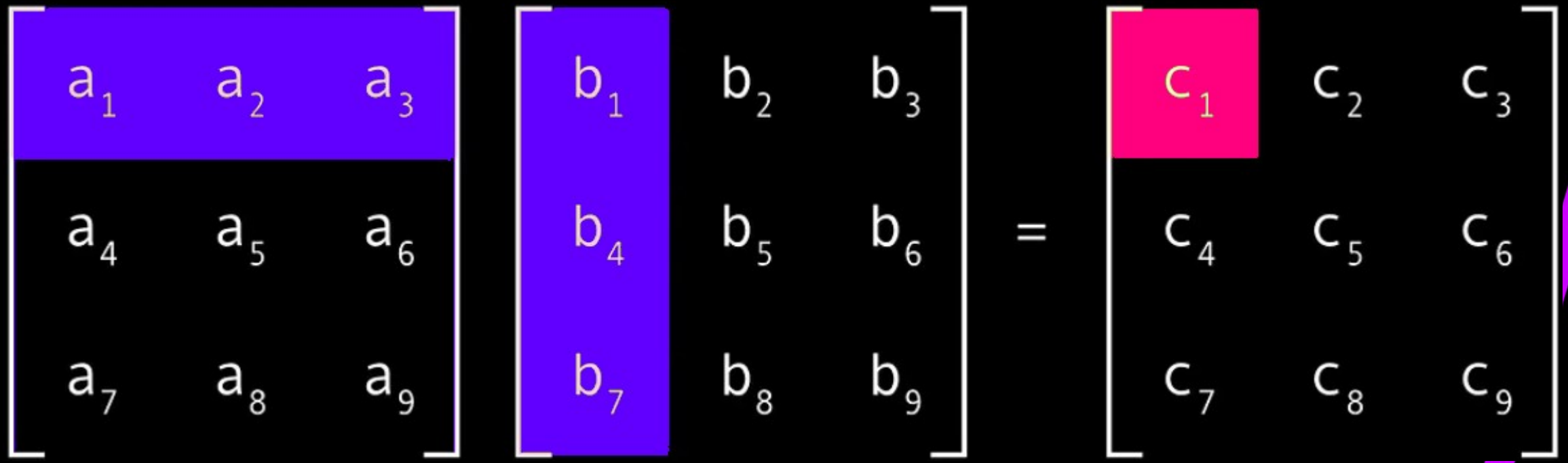
CLASS torch.nn.Conv1d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *padding_mode='zeros'*, *device=None*, *dtype=None*) [SOURCE]

Applies a 1D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, L) and output $(N, C_{\text{out}}, L_{\text{out}})$ can be precisely described as:

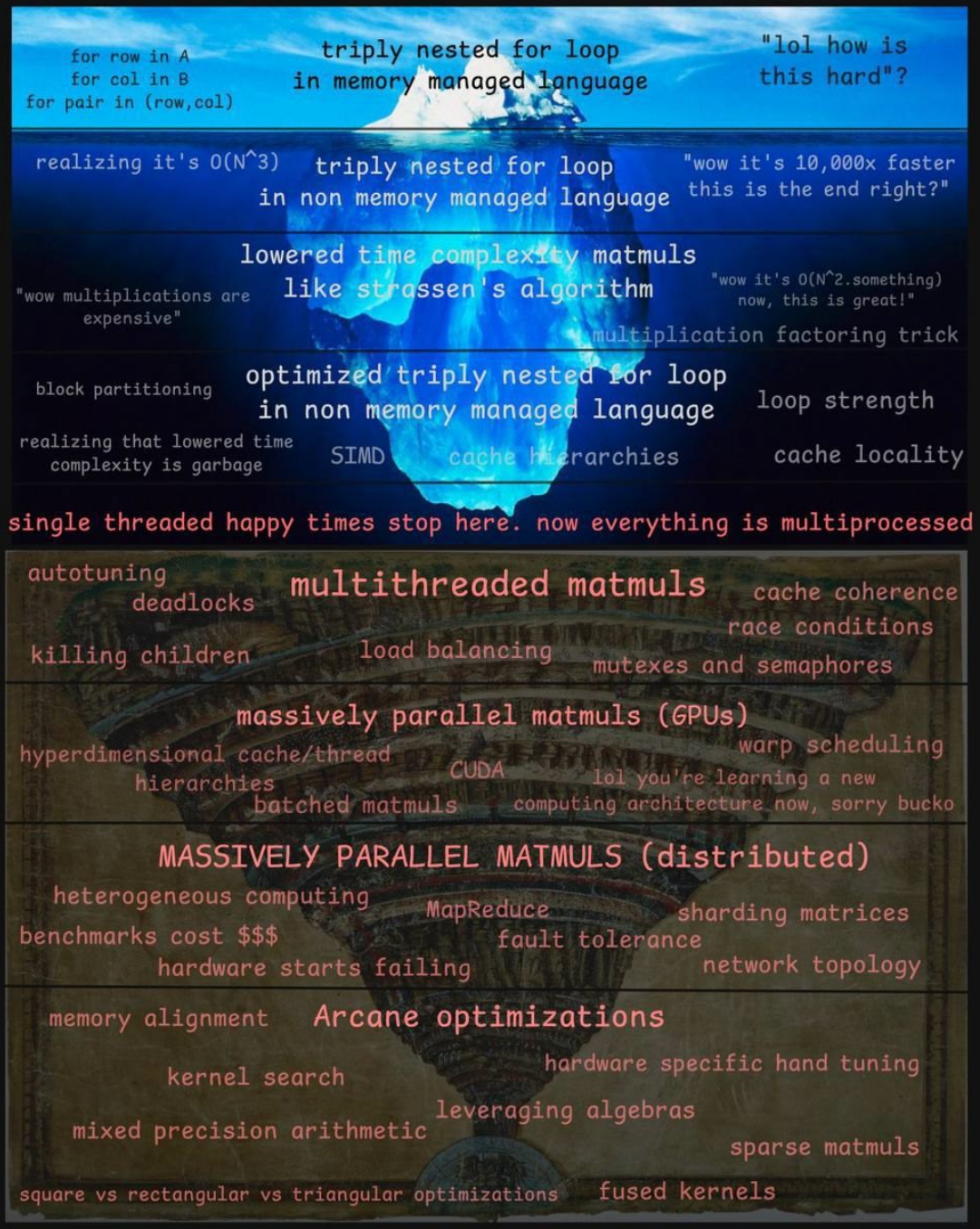
$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{in}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

КАК УМНОЖАТЬ МАТРИЦЫ



https://charchithowitzer.medium.com/matrix-multiplication-why-is-it-a-big-deal-cc8ef7490008

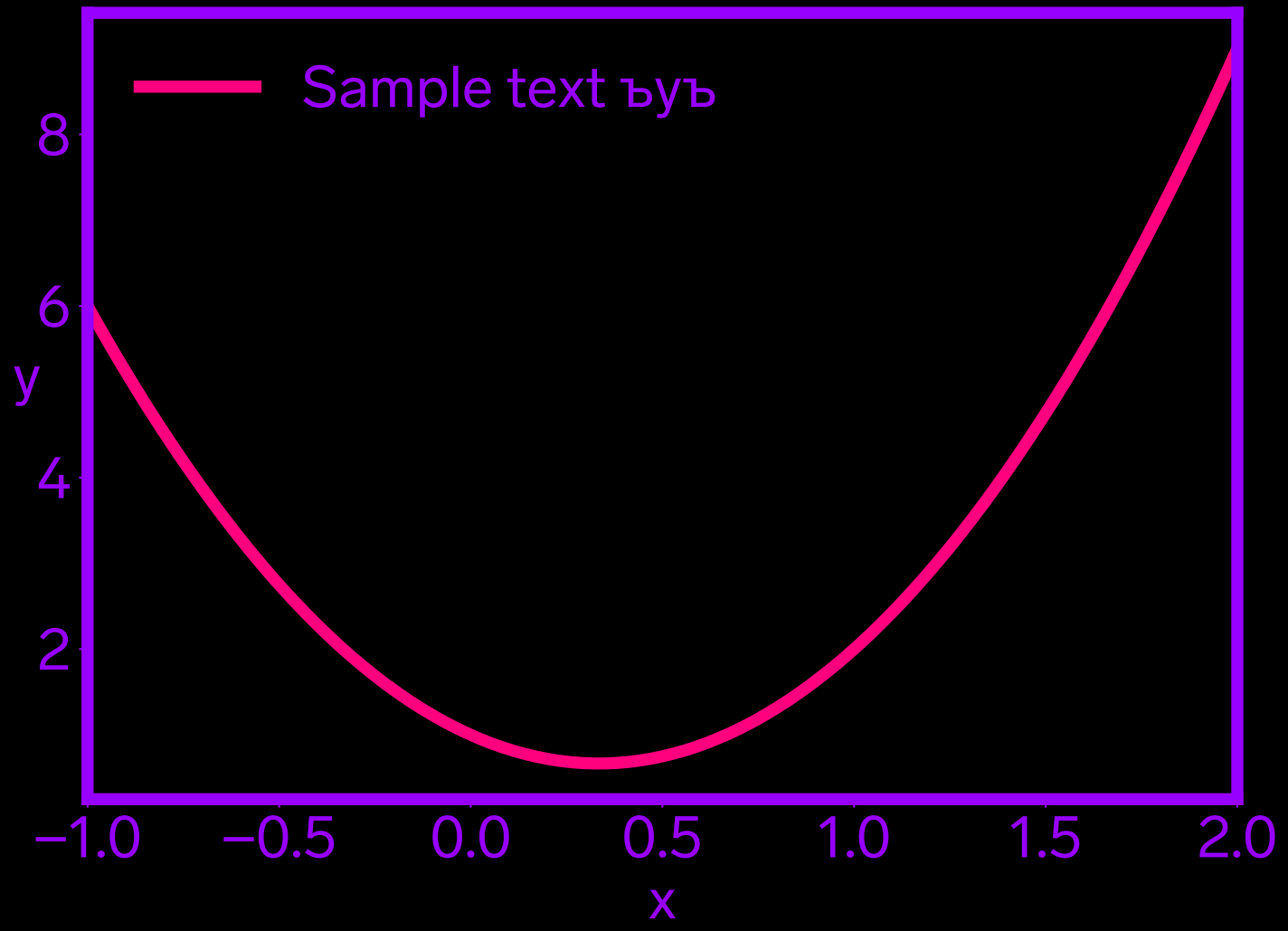
The nine circles of matmul

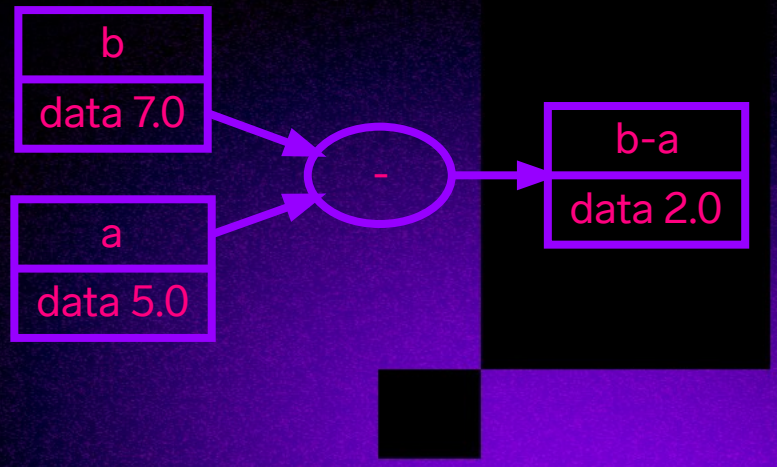


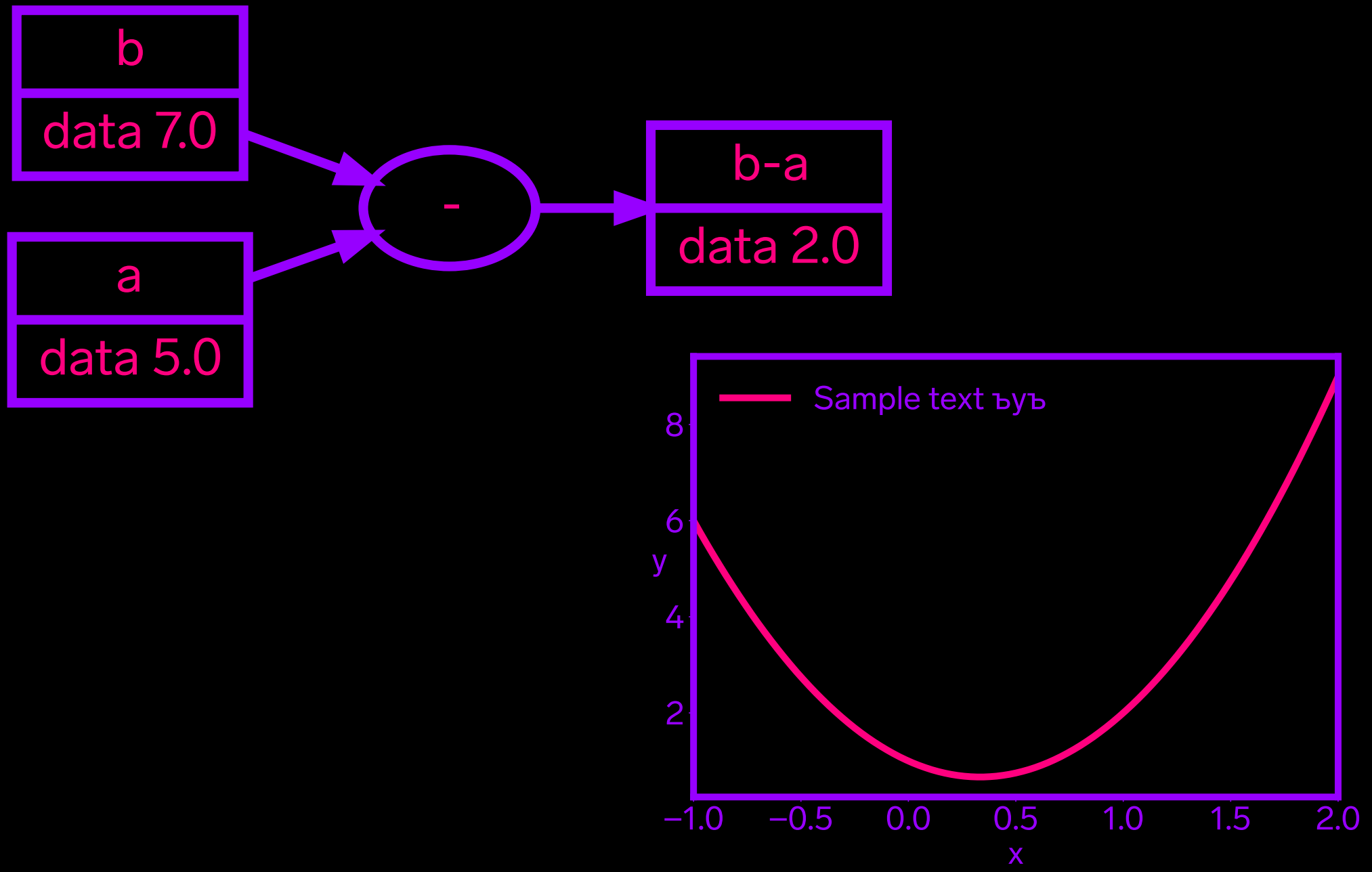

почему

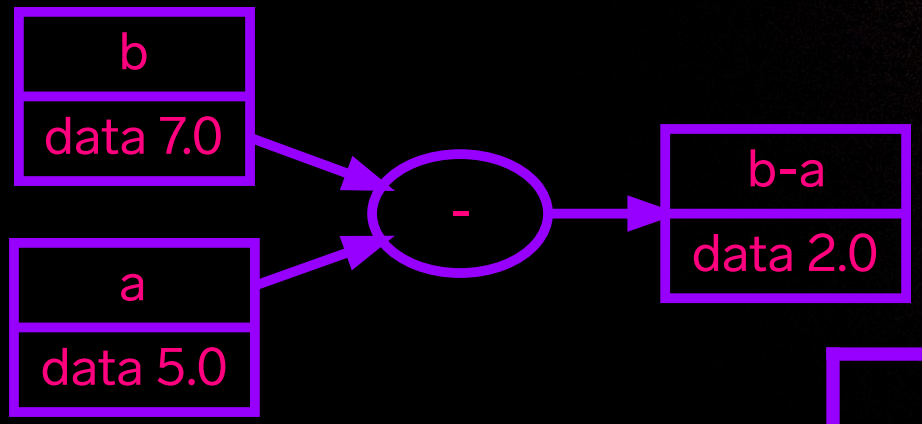
ПРОИЗВОДНАЯ И ЕЁ ДРУЗЬЯ

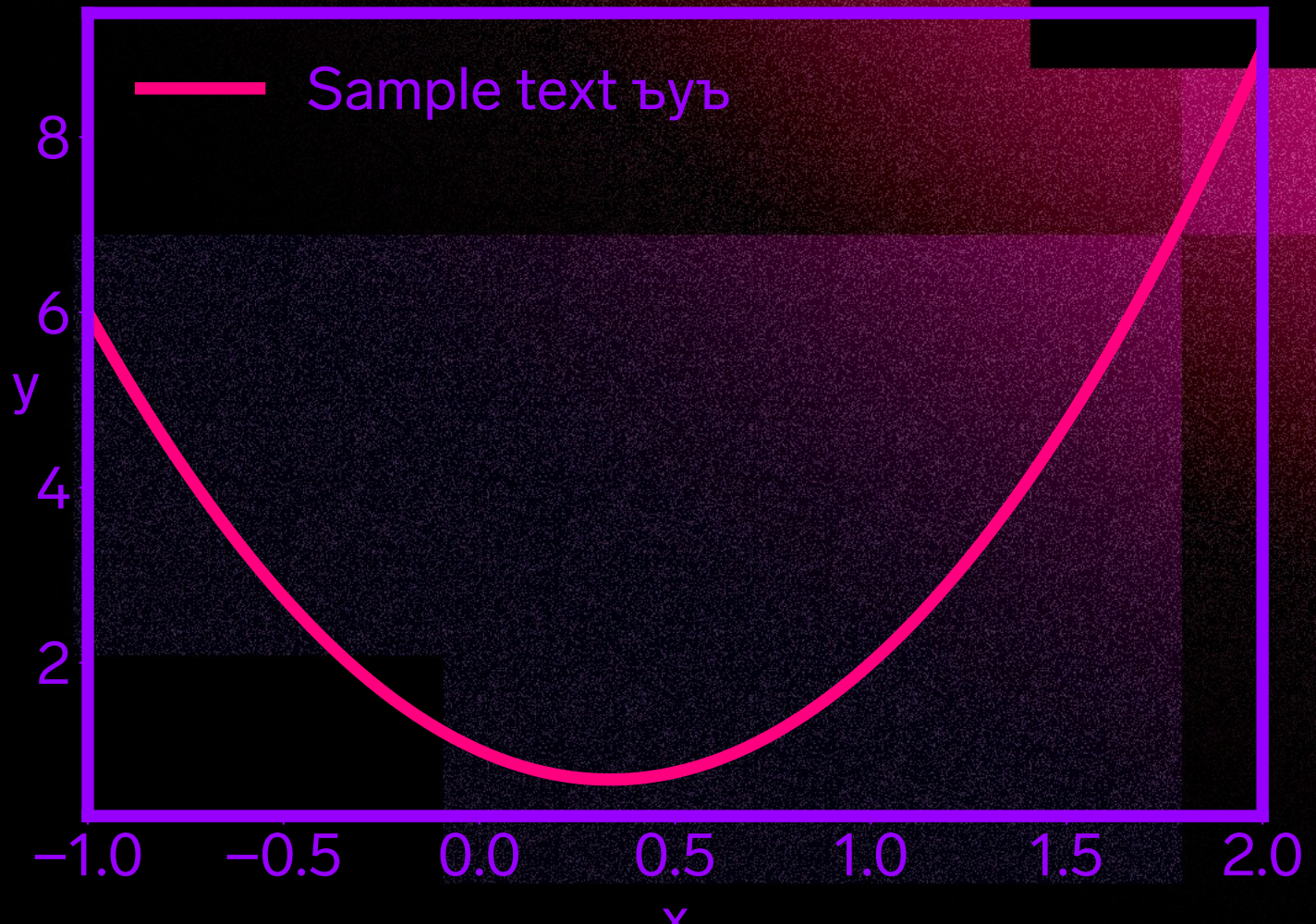
ПРОИЗВОДНАЯ: ВСПОМНИТЬ ВСЁ






b
data 7.0
a
data 5.0
-
b-a
data 2.0
Sample text ъуъ
8
6
4
2
y
–1.0
–0.5
0.0
0.5
1.0
1.5
2.0
x


b
data 7.0
a
data 5.0
-
b-a
data 2.0


Sample text ъуъ
8
6
4
2
y
–1.0
–0.5
0.0
0.5
1.0
1.5
2.0
x

А ЧТО ЕСЛИ ДАЖЕ GPU НЕ ХВАТАЕТ?