Supplementary Material for VkSplat

1 Reproduced metrics

We evaluated VkSplat on 7 permissively released scenes from Mip-NeRF 360 dataset, using images downscaled 4x and 2x for respectively outdoor and indoor scenes and saved in lossless PNG format, consistent with GSplat. Each setup is benchmarked 5 times on a NVIDIA RTX 3090 GPU, and 90% confidence interval of mean is reported, rounded consistent with existing academic reports.

For quality metrics, we report PSNR, SSIM, and LPIPS evaluated using two methods. LPIPS Alex is evaluated consistent with GSplat, which we use as our baseline in the paper. LPIPS VGG is evaluated consistent with original Inria implementation, which is more widely used in existing academic reports.

We report two VRAM values: total VRAM that intends to be fair when comparing with VRAM reported by GSplat, which is used by the paper but is lower than the actual VRAM reported by `nvidia-smi`; and peak VRAM that more accurately reflects whether an OOM (Out of Memory) will happen. The total VRAM is computed by summing all allocated buffers, while peak VRAM accounts for VRAM spike during buffer resizing. The choice of having two measures is to provide one value for fair comparison with GSplat (and other popular academic works) that reports VRAM returned by `torch.cuda.max_memory_allocated()`, which does not account for VRAM reserved by PyTorch caching allocator and allocated from custom CUDA code; and another value that more accurately reflects practical usability.

To ensure fairness of VRAM reporting in the paper, we trained bicycle scene with default densification on NVIDIA RTX 3090 GPU with both GSplat and VkSplat, while polling `nvidia-smi` at 1 Hz to measure maximum VRAM used by the process. For VkSplat, we measured 5.73 GiB maximum VRAM, slightly above reported total VRAM (5.72 GiB) but lower than reported peak VRAM (6.75 GiB). We believe this is because there's small amount of VRAM reserved by Vulkan runtime, but buffer resizing has short duration and was not captured by a sample rate of 1 Hz. Therefore, we assume an actual VRAM usage slightly over $(6.75/5.72-1)\times 100\% = 18\%$ higher than reported in the paper. For GSplat, we measured 11.98 GiB maximum VRAM, while the software itself reports 8.68 GiB VRAM usage, indicating an actual VRAM usage 38% higher than reported in the paper, plus additional VRAM usage not captured by a sample rate of 1 Hz. Since 38% is higher than 18%, we do not believe VRAM reporting in the paper introduces an unfair advantage for VkSplat.

In the following tables, time is in seconds, VRAM usage is in GiB, and number of Gaussians is in thousands.

Metrics for default densification:

	bicycle	garden	stump	bonsai	counter	kitchen	room	Average
PSNR	25.[48-54]	27.[65-79]	26.[87-91]	32.[17-26]	29.1[3-5]	31.[46-63]	31.[48-74]	29.2[0-7]
SSIM	0.77[2-3]	0.87[0-2]	0.78[1-3]	0.94[7-8]	0.916	0.933	0.927	0.87[8-9]
LPIPS Alex	0.16[1-2]	0.07[1-2]	0.14[5-6]	0.11[4-5]	0.14[1-2]	0.085	0.15[2-3]	0.12[4-5]
LPIPS VGG	0.20[2-4]	0.10[3-4]	0.20[5-7]	0.17[5-6]	0.17[8-9]	0.114	0.19[4-5]	0.168
Time	5[76-83]	54[4-6]	4[55-69]	27[3-6]	30[1-5]	42[0-3]	3[05-10]	41[2-4]
Total VRAM	5.7[0-3]	4.9[0-3]	4.[58-75]	1.3[3-5]	1.2[5-6]	1.9[2-4]	1.[59-64]	3.0[5-7]
Peak VRAM	6.7[3-6]	5.[79-82]	5.[40-61]	1.5[6-9]	1.4[6-8]	2.2[6-8]	1.[86-92]	3.[59-62]
NumGS	58[21-52]	[4999-5028]	4[664-839]	12[56-78]	11[61-88]	18[52-66]	15[52-95]	30[55-81]

Metrics for MCMC 1M Densification:

	bicycle	garden	stump	bonsai	counter	kitchen	room	Average
PSNR	25.5[3-8]	27.[25-32]	26.[89-94]	32.[59-71]	29.4[4-9]	31.[54-79]	32.[32-46]	29.[39-45]
SSIM	0.77[3-4]	0.85[4-5]	0.7[88-90]	0.953	0.92[3-4]	0.93[5-6]	0.938	0.881
LPIPS Alex	0.184	0.105	0.16[3-4]	0.10[6-7]	0.13[0-1]	0.08[6-7]	0.13[2-3]	0.130
LPIPS VGG	0.21[6-7]	0.13[4-5]	0.21[4-5]	0.16[4-5]	0.16[4-5]	0.114	0.17[2-4]	0.169
Time	21[7-9]	21[4-6]	21[4-5]	32[5-7]	3[67-70]	33[4-6]	31[5-6]	28[4-5]
Total VRAM	0.88	0.88	0.88	0.94	0.97	0.95	0.9[6-7]	0.9[2-3]
Peak VRAM	0.88	0.88	0.88	0.94	0.97	0.95	0.9[6-7]	0.9[2-3]
NumGS	1000	1000	1000	1000	1000	1000	1000	1000

For MCMC densification, we pre-allocate buffers to fit maximum number of Gaussians to avoid buffer resizing, and produced peak VRAM near-identical to total VRAM.

2 Timing breakdown

We benchmarked each setup once and reported timing of each stage. In the paper, tiling/sorting includes index offset, generate keys, sorting, and tile ranges, and loss includes copy image to device and loss gradient. All numbers are in seconds.

Timing breakdown on NVIDIA RTX 3090, Default densification:

	bicycle	garden	stump	bonsai	counter	kitchen	room	Average
Projection Forward	32.6	32.1	25.1	9.3	9.3	15.5	10.6	19.2
Index Offset	5.6	5.2	4.9	2.1	2.0	2.8	2.3	3.6
Generate Keys	8.5	8.0	7.1	6.3	7.6	8.3	6.7	7.5
Sorting	14.6	17.5	10.6	10.0	13.4	19.0	10.9	13.7
Tile Ranges	0.5	0.6	0.4	0.4	0.5	0.7	0.4	0.5
Rasterization Forward	31.9	28.4	20.4	17.4	23.1	34.0	22.1	25.3
Copy Image to Device	12.2	13.0	12.3	20.1	19.0	19.3	19.1	16.4
Loss Gradient	11.5	12.4	11.6	18.5	18.5	18.6	18.5	15.7
Rasterization Backward	163.0	151.0	110.7	89.7	116.0	172.8	106.5	130.0
Proj Bwd + Optimizer	236.2	217.5	199.1	58.6	53.2	86.3	66.7	131.1
Densification	8.8	9.1	8.0	2.4	2.4	3.9	2.9	5.4
Unaccounted	49.5	50.2	47.8	38.3	37.2	42.3	38.9	43.5
Total	574.9	545.1	458.1	273.0	302.2	423.7	305.6	411.8

Timing breakdown on NVIDIA RTX 3090, MCMC 1M densification:

	bicycle	garden	stump	bonsai	counter	kitchen	room	Average
Projection Forward	7.2	8.0	7.4	8.7	8.9	9.3	8.0	8.2
Index Offset	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
Generate Keys	7.7	7.7	6.9	17.4	15.0	16.0	15.4	12.3
Sorting	8.4	9.3	8.6	15.1	19.4	16.9	14.1	13.1
Tile Ranges	0.3	0.3	0.3	0.6	0.7	0.6	0.5	0.5
Rasterization Forward	18.3	14.4	16.7	29.1	37.3	30.1	29.7	25.1
Copy Image to Device	12.1	12.9	12.2	20.1	19.1	19.2	18.9	16.3
Loss Gradient	11.5	12.3	11.6	18.6	18.7	18.7	18.6	15.7
Rasterization Backward	81.7	76.5	81.9	142.4	175.1	148.0	136.8	120.3
Proj Bwd + Optimizer	44.0	47.0	42.4	48.1	47.5	48.7	46.3	46.3
Densification	3.1	3.3	3.0	3.4	3.3	3.4	3.3	3.2
Unaccounted	22.6	22.5	22.3	23.3	23.7	23.7	23.2	23.0
Total	218.2	215.5	214.6	327.9	370.0	335.7	316.1	285.4

Timing breakdown on AMD Radeon RX 7800 XT, Default densification:

	bicycle	garden	stump	bonsai	counter	kitchen	room	Average
Projection Forward	71.4	67.2	60.7	24.2	23.6	33.5	26.3	43.9
Index Offset	7.9	12.0	6.9	9.3	9.1	9.8	9.3	9.2
Generate Keys	11.7	11.3	9.8	6.4	7.6	8.6	7.0	8.9
Sorting	13.8	16.5	9.9	9.4	13.0	18.2	10.4	13.0
Tile Ranges	0.6	0.7	0.4	0.4	0.6	0.9	0.4	0.6
Rasterization Forward	67.1	69.0	42.9	38.9	51.1	79.9	46.4	56.5
Copy Image to Device	282.8	353.5	384.0	618.5	557.2	611.8	553.1	480.1
Loss Gradient	20.4	23.6	22.2	39.3	38.6	39.9	38.2	31.8
Rasterization Backward	183.4	184.0	122.0	106.9	140.6	218.0	126.1	154.4
Proj Bwd + Optimizer	511.1	484.7	439.6	129.1	118.3	193.1	148.8	289.2
Densification	13.3	14.1	12.0	4.5	4.2	7.0	4.5	8.5
Unaccounted	18.4	18.9	17.9	16.9	16.9	17.8	16.9	17.7
Total	1201.9	1255.4	1128.4	1003.9	980.6	1238.6	987.5	1113.8

Timing breakdown on AMD Radeon RX 7800 XT, MCMC 1M densification:

	bicycle	garden	stump	bonsai	counter	kitchen	room	Average
Projection Forward	15.3	16.4	14.8	17.3	17.3	17.8	16.0	16.4
Index Offset	7.5	7.5	7.5	7.6	7.5	7.5	7.5	7.5
Generate Keys	6.3	6.2	6.3	13.9	12.5	12.9	13.0	10.2
Sorting	7.7	8.3	7.8	14.5	18.8	16.5	13.3	12.4
Tile Ranges	0.3	0.4	0.3	0.6	0.9	0.8	0.6	0.6
Rasterization Forward	37.6	34.4	34.6	62.1	80.2	68.8	60.2	54.0
Copy Image to Device	340.1	321.3	248.7	688.9	556.8	485.4	524.4	452.2
Loss Gradient	21.1	22.3	19.7	40.9	38.9	38.1	38.1	31.3
Rasterization Backward	100.3	90.1	93.6	169.8	215.5	184.4	161.0	145.0
Proj Bwd + Optimizer	95.0	100.5	91.5	103.6	101.5	103.6	99.6	99.3
Densification	9.9	10.5	9.4	10.7	10.6	10.8	10.5	10.4
Unaccounted	13.4	13.7	13.5	14.2	13.9	14.1	14.0	13.8
Total	654.6	631.5	547.6	1144.4	1074.4	960.8	958.3	853.1

The largest time difference betwen AMD Radeon RX 7800 XT and NVIDIA RTX 3090 is copying image to device, which is nearly 30 times slower on AMD on average. We believe this is due to a PCIe difference, and the end-to-end training time can be optimized with asynchronous data transfer.