LINR-PCGC: Lossless Implicit Neural Representations for Point Cloud Geometry Compression

Supplementary Material

1. Appendix

1.1. Detail of parameters

The details of the parameters in our experiment are listed in Tab. 7.

Symbol	Description	Value
lr_0	Initial learning rate	0.01
lr_{min}	Minimum learning rate	0.0004
γ	Multiplicative factor of learning rate decay in StepLR	0.992
step size	Period of learning rate decay in StepLR	32
λ	Weight decay (for L2 penalty) factor in Adam	0.0001
$epoch_f$	Training epoch number for the first GoP	6-60
$epoch_s$	Training epoch number for subsequent GoPs	6-60
T	GoP size	32
M	Total frame count of an entire testing sequence	96
C_{mlp}	Hidden channel dimension of the MLP	24
C_{sconv}	Hidden channel dimension of the SConv	8
C_{EMB}	Channel dimension of the SEMB	8

Table 7. Detail of parameters of our experiment.

1.2. Supplementary Experiment result

To provide a quantitative analysis on the number of training epochs for the first GoP F, detailed bpp values under different F are given in Tabs. 8 to 10, where the training epoch for subsequent GoPs is fixed to 1, bpp denotes the average bpp of all sequences in a dataset, r.t. bpp denotes the relative bpp (%) of other methods over G-PCC, and w/o over. denotes the encoding time without overfitting time of our method. All times are in seconds.

1.3. Supplementary Bitstream and Time Allocation

In Sec. 4.2, we have given the bitstream allocation and the encoding/decoding time composition figures of MVUB. Here we give the figures for 8iVFB and Owlii as Figs. 12 and 14. Sec. 4.3 has given the training times vs. bpp curves of 8IVFB and MVUB, and here we give the curves of Owlii in Fig. 14.

	F:6	F:11	F:31	F:61
longdress	0.618	0.597	0.576	0.573
loot	0.57	0.549	0.532	0.528
redandblack	0.689	0.665	0.64	0.629
soldier	0.588	0.567	0.553	0.539
bpp	0.616	0.594	0.576	0.567
r.t. bpp	82.925	79.975	77.422	76.311
w/o over.	0.477	0.512	0.446	0.44
enc. time	2.464	3.869	8.005	15.092
dec. time	0.501	0.535	0.471	0.465

Table 8. Quantitative results on 8iVFB dataset of different F.

	F:6	F:11	F:31	F:61
basketball	0.452	0.438	0.422	0.414
dancer	0.473	0.457	0.44	0.432
exercise	0.46	0.443	0.428	0.418
model	0.475	0.461	0.445	0.434
bpp	0.465	0.45	0.434	0.425
r.t. bpp	78.759	76.204	73.497	71.949
w/o over.	0.402	0.46	0.389	0.397
enc. time	2.071	3.392	6.972	12.958
dec. time	0.422	0.478	0.41	0.417

Table 9. Quantitative results on Owlii dataset of different F.

	F:6	F:11	F:31	F:61
andrew10	0.833	0.801	0.769	0.754
david10	0.778	0.758	0.723	0.708
phil110	0.841	0.812	0.777	0.772
ricardo10	0.802	0.76	0.729	0.709
sarah10	0.777	0.748	0.724	0.704
bpp	0.806	0.776	0.744	0.729
r.t. bpp	87.548	84.25	80.844	79.206
w/o over	0.524	0.57	0.524	0.518
enc. time	2.712	4.385	9.291	16.967
dec. time	0.554	0.599	0.554	0.548

Table 10. Quantitative results on MVUB dataset of different F.

1.4. Supplementary of Ablation Study

Supplementary analysis of initialization strategy. To further demonstrate the effectiveness of the initialization strategy, we have sketched Fig. 15. We can observe from the

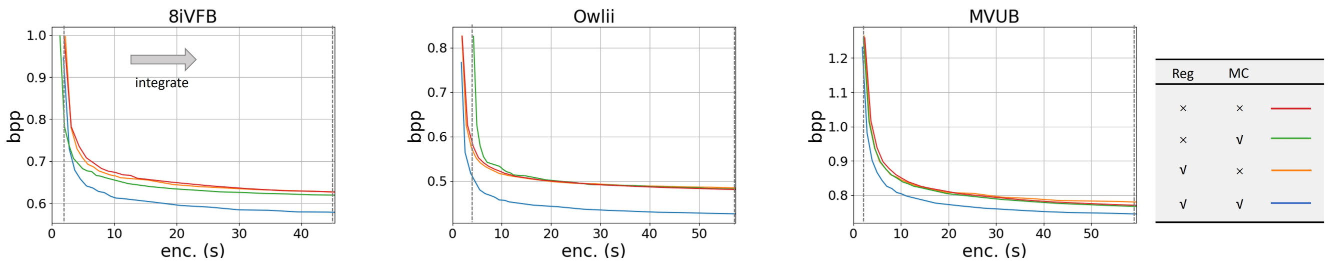

Figure 11. The impact of regularization terms on MC modules.

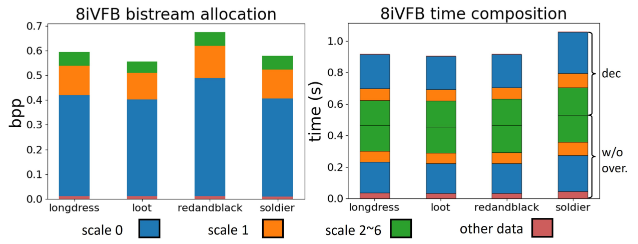

Figure 12. Bitstream allocation and encoding/decoding time composition in 8iVFB.

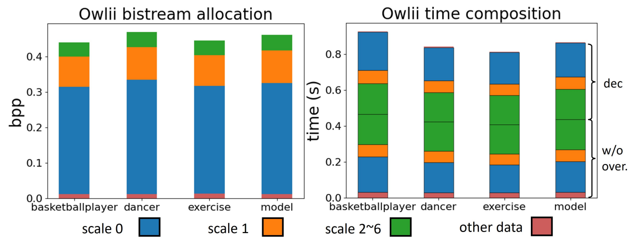

Figure 13. Bitstream allocation and encoding/decoding time composition in Owlii.

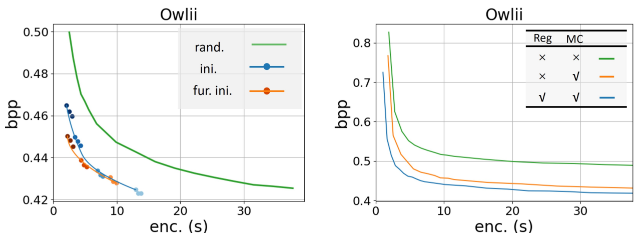

Figure 14. (a) The training time–bpp curves with randomly initializing each GoP (rand.), ini., and fur. ini. in Owlii. (b) Impact of each module in LINR-PCGC in Owlii.

figure that the bitstream of each GoP has a significant decrease. This is because the parameters of the latter GoP are initialized by the parameters of the previous GoP. Under the same optimization time, the later the GoP, the better the encoding efficiency can be achieved.

The effect of Model Compression (MC). To demonstrate the effectiveness of MC, we presented the original size of the network parameters (Ori.), the bitstream size of directly converting quantized integer parameters into a bitstream (ToByte), the bitstream size after further compressing ToByte using LZ77 (LZ77), and the bitstream size generated by arithmetic coding using a Laplace distribution (Laplace). As depicted in Tab. 11, arithmetic coding with a Laplacian prior assumption significantly reduces the bitstream size of model parameters.

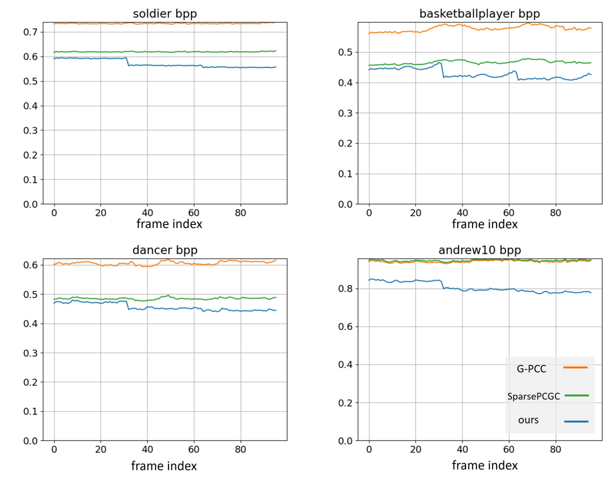

Figure 15. Bitstream size of each frame. The frame number of each sequence is 96, and GoP size is 32. Both the first GoP and subsequent GoPs are trained for 6 epochs.

	Owlii	8iVFB	MVUB	Avg
Ori.	1750784	1750784	1750784	1750784
ToByte	437762	437762	437762	437762
LZ77	267790	269554	250825.4	268672
Laplace	248490	251360	240352.9	251360

Table 11. Bitstream sizes (in bits) of the model parameters under different model compression algorithms.

The effect of the regularization item (Reg.). The regularization term can reduce the absolute value of the network parameters, thus making the quantized parameters closer to

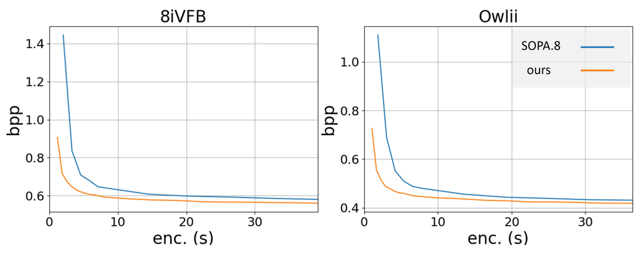

Figure 16. Comparison of ours method (without pretrain) and replace our CNP module to 8-stage SOPA with 8 hidden channels.

the Laplace distribution. Therefore, adding a regularization term is beneficial for MC. The specific situation is shown in Fig. 11. We choose the method with Reg. and MC as the baseline. Then, we integrate the overlapping parts of time and make a ratio to the baseline to obtain Tab. 14. We can observe from the first and second lines that when there is no regularization term, the MC module can only save 0.826% of the bitstream. Next, we can observe from the third and fourth lines of the table that when there exists a regularization term, MC can save 8.17% bitstream. Although we can conclude from the comparison between the first and third lines that the presence of regularization terms alone does not result in significant stream savings (0.092%), its existence is one of the foundations for the functioning of the MC module.

The advantages of CNP under the INR architecture. The simplest idea for upsampling is to directly use SOPA from SparsePCGC and perform overfitting. However, SOPA training takes nearly 9 hours and has tens of millions of bits of parameters[3]. For online training, this is expensive and unacceptable. Therefore, we reduce the number of hidden channels in the 8-stage SOPA from 32 to 8 and utilize channel-wise prediction to replace transpose SparseConv. And we illustrate the comparison result in Fig. 16. Then we integrate the overlapping parts of time and calculate the ratio relative to SOPA to obtain Tab. 12. From the table, we can observe that CNP can save about 7.62% of the bitstream compared to 8-stage SOPA. To further demonstrate the advantages of CNP under the INR architecture, we construct Tab. 13 which shows the comparison between CNP and 8-stage SOPA. From the table, we can observe that CNP can save approximately 61.91% of peak memory with the same number of hidden channels. This also indicates that prediction based on a two-layer octree structure is more memory efficient than transpose convolution in SOPA.[4]

[3]This information comes from the training log provided by the authors.

[4]We did not compare on MVUB because running 8-stage SOPA with 8 hidden channels on the MVUB dataset would exceed the memory of RTX 3090 in our INR framework.

	Owlii	8iVFB	avg
8-stage SOPA	1	1	1
ours	0.9238	0.9238	0.9238

Table 12. Comparison of ours method (without pretrain) and replace our CNP module to 8-stage SOPA with 8 hidden channels.

	Owlii	8iVFB	avg
8-stage SOPA	10.64	13.21	11.92
ours	4.00	5.09	4.54

Table 13. Comparison of peak memory usage between ours method and 8-stage SOPA with 8 hidden channels.

Reg	MC	Owlii	8iVFB	MVUB	avg.
×	×	1.132	1.089	1.049	1.090
×	✓	1.142	1.062	1.041	1.081
✓	×	1.132	1.085	1.050	1.089
✓	✓	1.000	1.000	1.000	1.000

Table 14. The impact of regularization terms on MC modules.

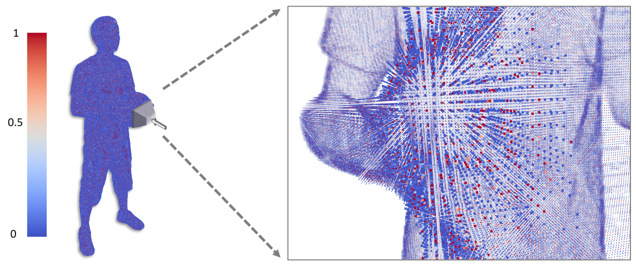

Figure 17. Bitstream heatmap.

1.5. Bitstream heatmap

Fig. 17 illustrates the absolute difference between the estimated occupancy probability and the actual occupancy values. A larger difference is equivalent to a higher bitrate of a point. Points with higher bit rates appear periodically in the size of 2^3 cubes as the right part of Fig. 17. This phenomenon occurs because we use decoded child nodes to predict non-decoded child nodes. The first batch of child nodes typically has higher bit rates due to the lack of current scale priors, while those predicted based on other child nodes have lower bit rates.