Supplementary Material of EchoScene

Guangyao Zhai[1] Evin Pınar Örnek[1] Dave Zhenyu Chen[1] Ruotong Liao[2,3]
Yan Di[1] Nassir Navab[1] Federico Tombari[1,4] Benjamin Busam[1,3]

[1]Technical University of Munich [2]Ludwig Maximilian University of Munich
[3]Munich Center for Machine Learning [4]Google
guangyao.zhai@tum.de

In this Supplementary Material, we provide a **scene graph visualization** of one scene from SG-FRONT to comprehensively illustrate the complexity and the identity of the semantic scene graph. Furthermore, within the Supplementary Manuscript, we provide the following:

- Section 1: Manipulation Strategy.
- Section 2: Manipulation Qualitatives.
- Section 3: Additional Results.
- Section 4: Additional Qualitatives.
- Section 5: Additional Experimental Details.

1 Manipulation Strategy

We propose a novel training strategy for node and edge manipulation tailored for scene graph diffusion, thus editing the generated scenes.

VAE+GAN-based strategy. In the previous methods [3,8], the layout branch is modeled by a VAE architecture. During training, the input scene graph must be augmented by ground truth bounding boxes. In this case, pseudo layouts are created from the output side after the manipulation, and there is no matching ground truth to supervise them. To make the pipeline functional, they set an additional GAN module to help train the pseudo label. For example, as shown in Fig. 1.A, the original relationship is {`Bed` $\longrightarrow$ `Left of` $\longrightarrow$ `TV stand` }, with a manipulator changing it to {`Bed` $\longrightarrow$ `Right of` $\longrightarrow$ `TV stand` }. Only the bounding boxes on the input side exist in the dataset, while the ones on the output side are generated and need to be discriminated. Such a manipulation strategy relies on the performance of GAN, whose training scheme is not stable and requires large-scale training, consequently affecting the training quality of layout VAE. Thus, the results shown in Tab. 2 in the main paper are always unsatisfactory.

Our strategy. In contrast, we set both branches through diffusion processes, avoiding the necessity of input ground truth bounding boxes during training. Instead, we are able to simulate an inverted workflow by setting input as pseudo graphs with fake relations, and we manipulate them back to the existing graphs in the dataset. For example, as shown in Fig. 1.B, the ground truth relation label is {`Bed` $\longrightarrow$ `Right of` $\longrightarrow$ `Nightstand`}, we simulate the relation changing by setting an arbitrary label, *e.g.*, {`Bed` $\longrightarrow$ `Front of` $\longrightarrow$ `Nightstand`}. In

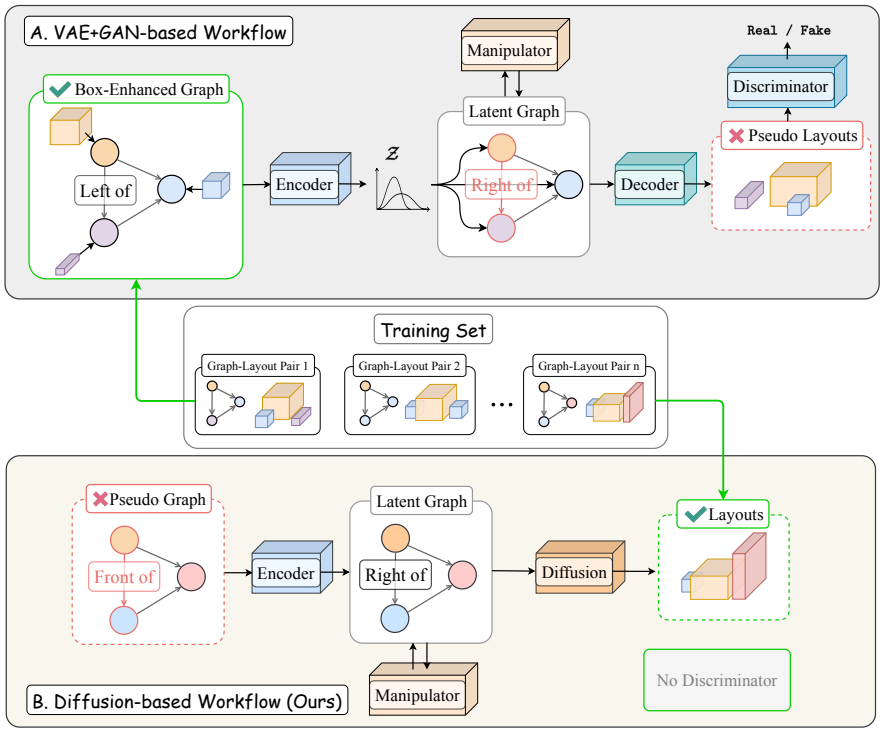

Fig. 1: Manipulation workflow. (Top) Previous VAE-based methods set the input as real and use a GAN-based module to facilitate the output prediction. (Bottom) Our Diffusion-based framework inverse the workflow by starting with a pseudo graph and ending with ground truth, without redundant modules.

this case, we do not need to provide real data as input and further do not set an additional GAN module to facilitate the training. Thus, the arbitrary pseudo side enables large-scale training, maintaining a good manipulation performance. More importantly, we do not need to discriminate the final layout prediction. With our proposed strategy, we maintain the typical diffusion training routine, which has Gaussian noise as supervision during training without further exhausting inference. Last but not least, pure diffusion dynamics guarantee stable training.

2 Manipulation Qualitatives

We provide several qualitative results of the scenes before and after manipulation in Fig. 2. The procedure includes EchoScene generating a scene based on a provided scene graph, followed by scene modification according to adjustments made to the graph.

Object addition. We first show object addition in the scene through the incorporation of new nodes and their corresponding edges into the scene graph. For instance, the second row illustrates the addition of a sofa node to the graph.

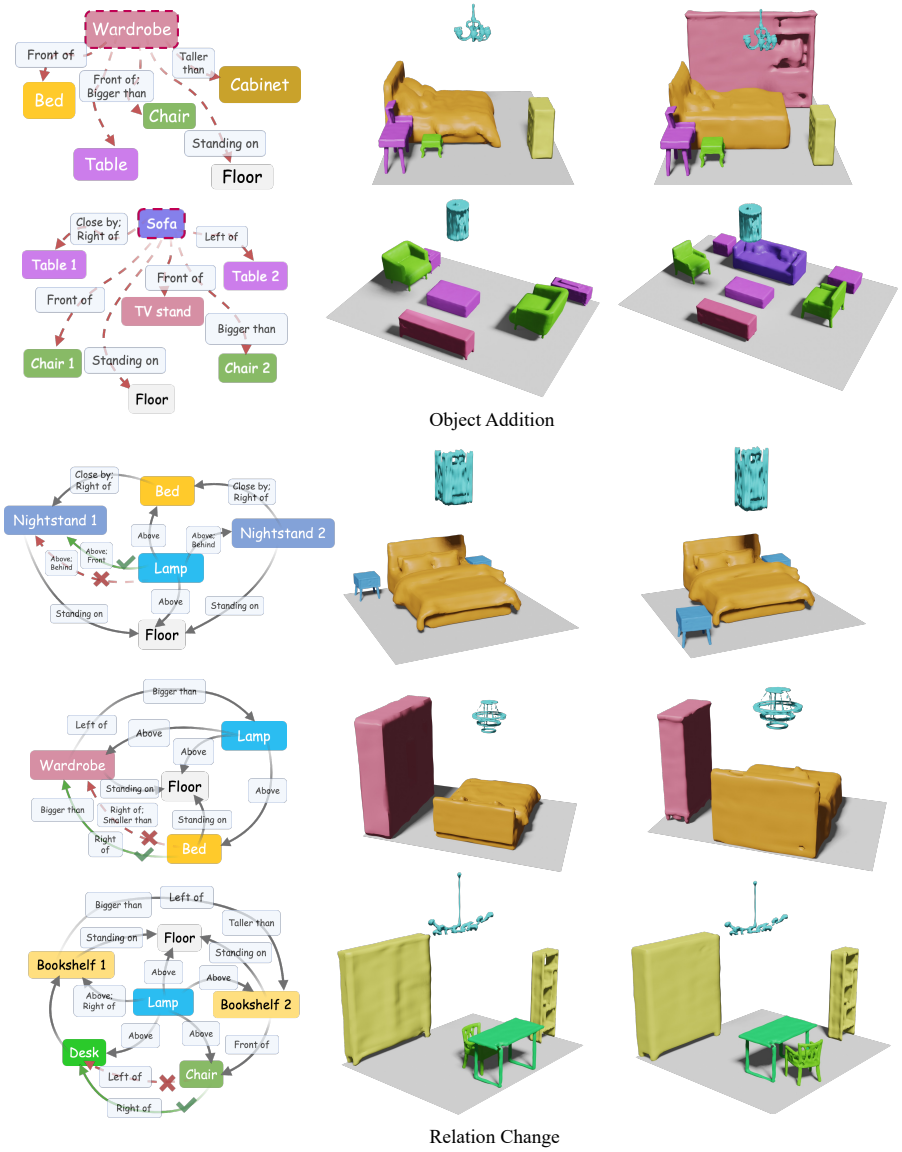

Fig. 2: Scene manipulation. (Upper main row) Object addition to the scene graph. (Bottom main row) Partial relation change in the graph. (Zoom for details)

Upon this addition, EchoScene adeptly reconfigures the scene, modifying the pose and appearance of pre-existing objects to seamlessly integrate the sofa into the scene.

Relation change. Next, we show object rearrangement within the scene through modifications to specific graph relations. The depiction is organized row-wise, showcasing various types of edge manipulations: the first row highlights the

Method	Metric	Bed	N.stand	Ward.	Chair	Table	Cabinet	Lamp	Shelf	Sofa	TV stand
Graph-to-3D [3]	MMD (↓)	1.56	3.91	1.66	2.68	5.77	3.67	6.53	6.66	1.30	1.08
CommonScenes [8]		0.49	0.92	0.54	0.99	1.91	0.96	1.50	2.73	0.57	0.29
EchoScene (Ours)		0.37	0.75	0.39	0.62	1.47	0.83	0.66	2.52	0.48	0.35
Graph-to-3D [3]	COV (%, ↑)	4.32	1.42	5.04	6.90	6.03	3.45	2.59	13.33	0.86	1.86
CommonScenes [8]		24.07	24.17	26.62	26.72	40.52	28.45	36.21	40.00	28.45	33.62
EchoScene (Ours)		39.51	25.59	37.07	17.25	35.05	43.21	33.33	50.00	41.94	40.70
Graph-to-3D [3]	1-NNA (%, ↓)	98.15	99.76	98.20	97.84	98.28	98.71	99.14	93.33	99.14	99.57
CommonScenes [8]		85.49	95.26	88.13	86.21	75.00	80.17	71.55	66.67	85.34	78.88
EchoScene (Ours)		72.84	91.00	81.90	92.67	75.74	69.14	78.90	35.00	69.35	78.49

Table 1: Object-level generation performance. We report MMD($\times$0.01), COV and 1-NNA for evaluating shapes by means of quality and diversity.

`front of/behind` relationship, the second focuses on `bigger/smaller than`, and the final row on `left/right of`. An interesting case is observed in the second row, where the bed is adjusted to be bigger than the wardrobe from the volume perspective. This particular relationship between the bed and wardrobe is comparatively rare within the dataset. Yet, EchoScene can jointly adjust the bounding box sizes of both the wardrobe and bed to achieve the goal. In the last row, EchoScene successfully alters the relationship between the desk and chair, while reorienting the chair's pose to face the desk, thereby maintaining inter-object consistency within the scene.

3 Additional Results

We believe our shape branch driven by shape echoes can bring more object generation compliance to the global scenes. Thus, we further conduct object-level analysis, following [8], to report the MMD ($\times$0.01), COV (%), and 1-nearest neighbor accuracy (1-NNA, %) for evaluating per-object generation. As shown in the first two rows of Table 1, our method shows better performance on most of the categories in both MMD and COV, which highlights the object-level shape generation ability of EchoScene. 1-NNA directly measures distributional similarity to the ground truth objects in both diversity and quality. The closer 1-NNA is to 50%, the better the shape distribution is captured. It can be observed that in most of the categories, our method surpasses CommonScenes in the evaluation of distributional similarity. Overall, EchoScene exhibits more plausible object-level generation than the previous state-of-the-art.

4 Additional Qualitatives

Generative methods. We first show more quantitative results in Fig. 3. Our method can achieve more inter-object consistency and generation quality. For example, in the bedroom, Graph-to-3D fails to achieve desk-and-chair consistency, while EchoScene can generate a desk whose appearance is closer to a

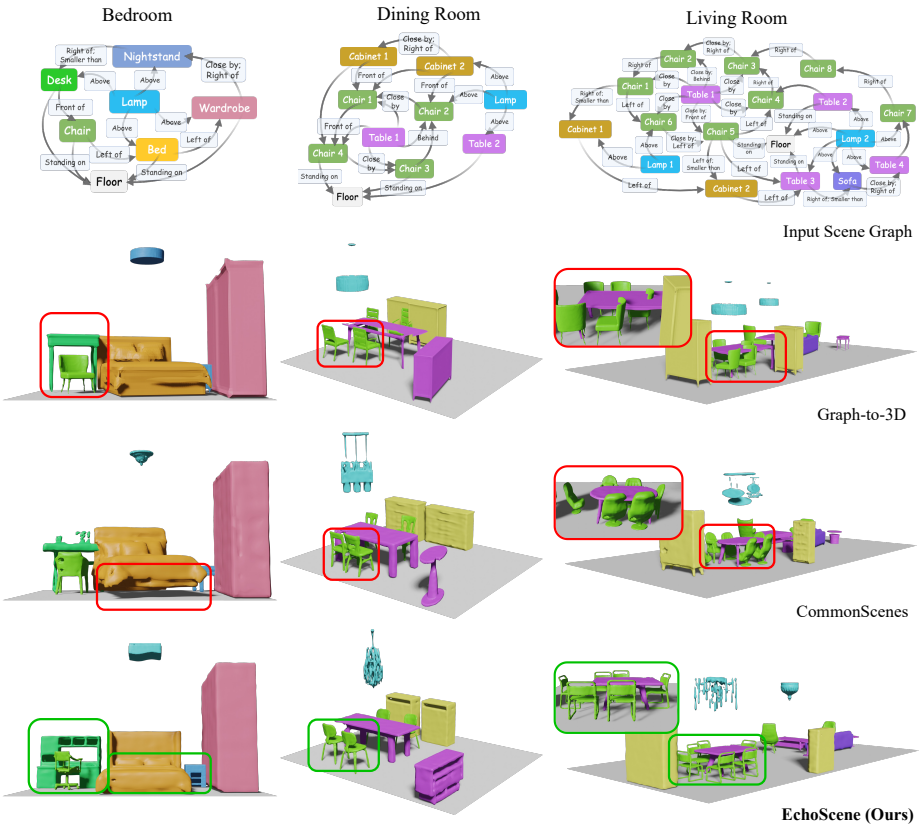

Fig. 3: More comparisons with other methods. Red rectangles highlight the inconsistent generation. (Zoom for details)

desk in the real scenario with a chair suitable for it. In the living and dining rooms, Graph-to-3D either fails on shape consistency or angle predictions of chairs. CommonScenes cannot guarantee a fine-grained shape consistency, while the chairs coarsely look similar. In contrast, EchoScene can generate coherent shapes and make pose prediction more accurate.

Retrieval methods. Retrieval methods select objects from the database based on how closely their bounding box sizes match those of the generated layouts. Therefore, this line of work suffers from inter-object inconsistency; for example, chairs are not recalled in suits. Such inconsistencies often stem from even minor misalignment in the size of generated bounding boxes, leading to the selection of entirely different objects than intended. Despite this, our focus in Figure 4 is to illustrate the effectiveness of graph constraints, instead of consistency. It is observable that even though InstructScene [5] demonstrates the capability to generate objects in a decent manner, it fails to adhere to the partial graph

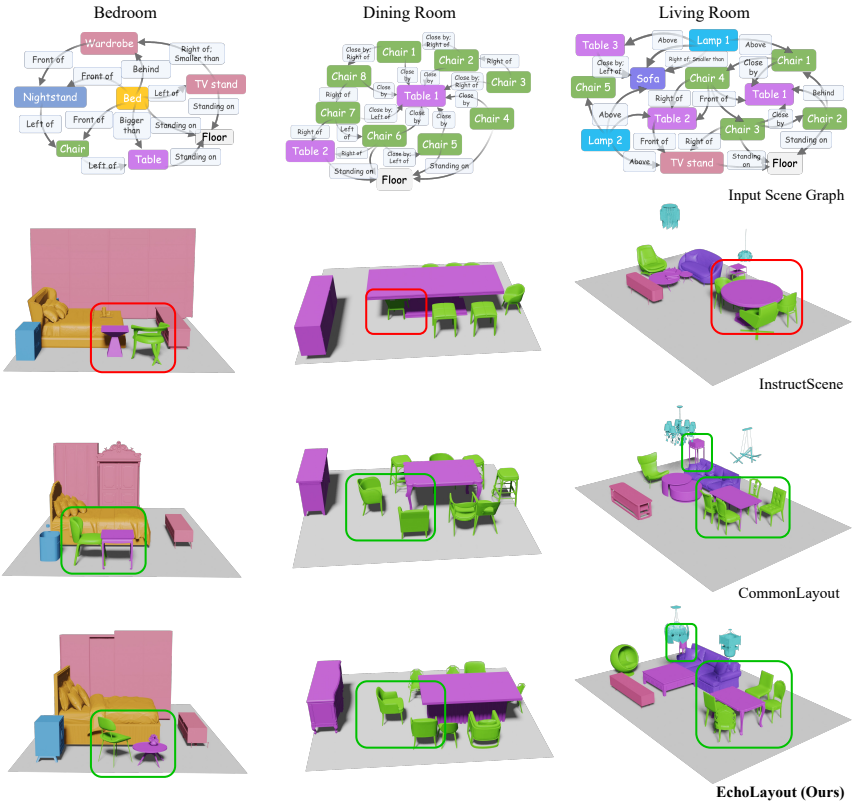

Fig. 4: Comparisons with other retrieval methods. Input scene graphs have more edges between two nodes than the ones visualized here. Red rectangles highlight the inaccurate graph constraints. (Zoom for details)

Fig. 5: Off-the-shelf texture creation. A bedroom with a complex structure bed inside generated by EchoScene and textured in different styles by SceneTex [1].

constraints. On the contrary, both CommonLayout [8] and EchoLayout exhibit proficiency in complying with these constraints.

Texture Generation. We finally show additional texture generation on a relatively complex structured bedroom in Fig. 5. EchoScene can provide well-generated scenes that are compatible with an off-the-shelf SceneTex [1] to generate textures.

5 Additional Experimental Details

5.1 Baselines.

Graph-to-3D series [3]. This series includes one generative method and three object retrieval methods. First, the full generative version *Graph-to-3D*, stacking two VAE-based branches for object and layout generation, respectively. Second, *Graph-to-Box*, the single layout branch focusing on the object retrieval task. Third, *Progressive*, a modified baseline upon Graph-to-Box, specifically adding objects one by one in an autoregressive manner. Fourth, *3D-SLN* [6], sharing the same architecture as Graph-to-Box, but without layout standardization during training. We follow the illustration in the supplementary of CommonScenes.

CommonScenes series [8]. This series includes the fully generative version *CommonScenes*, and its layout branch for object retrieval, *CommonLayout*. We follow the illustration in the supplementary of CommonScenes.

Text-to-shape series. This series includes two generative baselines. One is built upon CommonScenes called *CommonLayout+SDFusion*, and the other is *EchoLayout+SDFusion*, with EchoLayout as our layout branch. Both methods achieve the fully generative ability by first generating the bounding boxes and further generating shapes upon a text-to-shape method SDFusion [2] within each bounding box, according to the textual information in the graph nodes.

DiffuScene [7]. DiffuScene is a diffusion-based retrieval method, which can be both unconditional and text-conditional. It uses a diffusion process to generate bounding boxes of an object set as the scene layout. We test its text-conditional version to perform scene synthesis on the SG-FRONT dataset. Specifically, we transfer scene graph description to sentences based on the script provided by DiffuScene. Then, we feed the sentences to the BERT [4] encoder to have textual embeddings for conditioning the denoising process. The experimental settings are kept the same as the original ones. DiffuScene is explicitly designed for scene completion tasks, where it uses partial inputs as a basis and generates additional content. Our task, however, concentrates on achieving fully controllable scene synthesis. This means we aim to produce scenes that precisely match the descriptions provided in the scene graph. Consequently, the evaluation of DiffuScene focuses solely on the fidelity of the generated scenes, assessing how closely the distribution of the generated content aligns with the original data. We do not enforce strict adherence to the graph constraints, recognizing that a significant portion of the content is creatively inferred or 'hallucinated.'

InstructScene [5]. InstructScene is another retrieval method with a closer setting to us. It stacks two diffusion-based stages: Firstly, it transfers sentences using a graph transformer denoiser to a semantic graph containing shape prior as nodes and allocates each two nodes a single edge. Secondly, another graph transformer denoiser serves as a 3D layout decoder, taking the graph and steadily denoising the 3D bounding boxes as the scene layouts. Our task focuses on graph-conditioned scene synthesis. Therefore, we conduct the experiment solely in the second stage of InstructScene. We train the graph transformer denoiser with scene graphs in SG-FRONT until its convergence. As the synthesized scenes are fully controllable by the scene graph, we are able to evaluate both scene fidelity and the performance of graph constraints. When synthesis the scenes, we retrieve the objects whose sizes are closest to the ones of generated bounding boxes. This retrieval strategy is kept the same for all methods for comparison fairness.

5.2 Implementation Details.

Trainval and test splits. We use the settings in DiffuScene [7] and CommonScenes [8] to train and test all methods on SG-FRONT and 3D-FRONT. The whole dataset contains 4,041 bedrooms, 900 dining rooms, and 813 living rooms. The training split contains 3,879 bedrooms, 614 dining rooms, and 544 living rooms, with the rest as the test split.

Batch size definition. The two branches use individual batches in terms of the different training objectives. The layout branch uses a scene batch B_s during one training step, containing all bounding boxes in scenes. There are two ways to determine the batch size for the shape branch.

First, if it is the ablated version where the shape branch is without shape echoes, we can straightforwardly sample B_o objects (nodes) out of the scene batch to train, as illustrated in CommonScenes [8]. This method allows for random sampling of objects since the lack of shape echoes means there's no requirement for the objects to originate from the same scene.

Second, if it is the full version, we set up a maximum batch size B_o^*, and select scenes where the total number of objects B_o closely approaches but does not exceed B_o^*. This method ensures that we efficiently utilize the batch capacity while adhering to the constraint of keeping the sum of objects within the predetermined maximum batch size. In this case, the batch size in shape branch B_o slightly fluctuates, as the object numbers are not fixed in the scene.

Training procedure. We train *EchoLayout* (layout branch) for 2000 epochs with $B_s = 64$. The learning rate is set to [1e-4, 5e-5, 1e-5, 5e-6] at [0, 35,000, 70,000, 140,000] step. For the full *EchoScene*, which integrates shape information via a shape branch into the pipeline—where both branches utilize a shared latent graph representation—we extend the training by an additional 50 epochs. The maximum batch size B_o^* for the shape branch is set to 64. The learning rate is kept in the same fashion.

References

1. Chen, D.Z., Li, H., Lee, H.Y., Tulyakov, S., Nießner, M.: SceneTex: High-quality texture synthesis for indoor scenes via diffusion priors. In: CVPR (2024) 6, 7
2. Cheng, Y.C., Lee, H.Y., Tuyakov, S., Schwing, A., Gui, L.: SDFusion: Multimodal 3D shape completion, reconstruction, and generation. In: CVPR (2023) 7
3. Dhamo, H., Manhardt, F., Navab, N., Tombari, F.: Graph-to-3D: End-to-end generation and manipulation of 3D scenes using scene graphs. In: ICCV (2021) 1, 4, 7
4. Kenton, J.D.M.W.C., Toutanova, L.K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019) 7
5. Lin, C., MU, Y.: InstructScene: Instruction-driven 3D indoor scene synthesis with semantic graph prior. In: ICLR (2024) 5, 8
6. Luo, A., Zhang, Z., Wu, J., Tenenbaum, J.B.: End-to-end optimization of scene layout. In: CVPR (2020) 7
7. Tang, J., Nie, Y., Markhasin, L., Dai, A., Thies, J., Nießner, M.: DiffuScene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis. In: CVPR (2024) 7, 8
8. Zhai, G., Örnek, E.P., Wu, S.C., Di, Y., Tombari, F., Navab, N., Busam, B.: CommonScenes: Generating commonsense 3D indoor scenes with scene graphs. In: NeurIPS (2023) 1, 4, 7, 8