ShapeWalk: Compositional Shape Editing through Language-Guided Chains

Habib Slim
KAUST
habib.slim@kaust.edu.sa

Mohamed Elhoseiny
KAUST
mohamed.elhoseiny@kaust.edu.sa

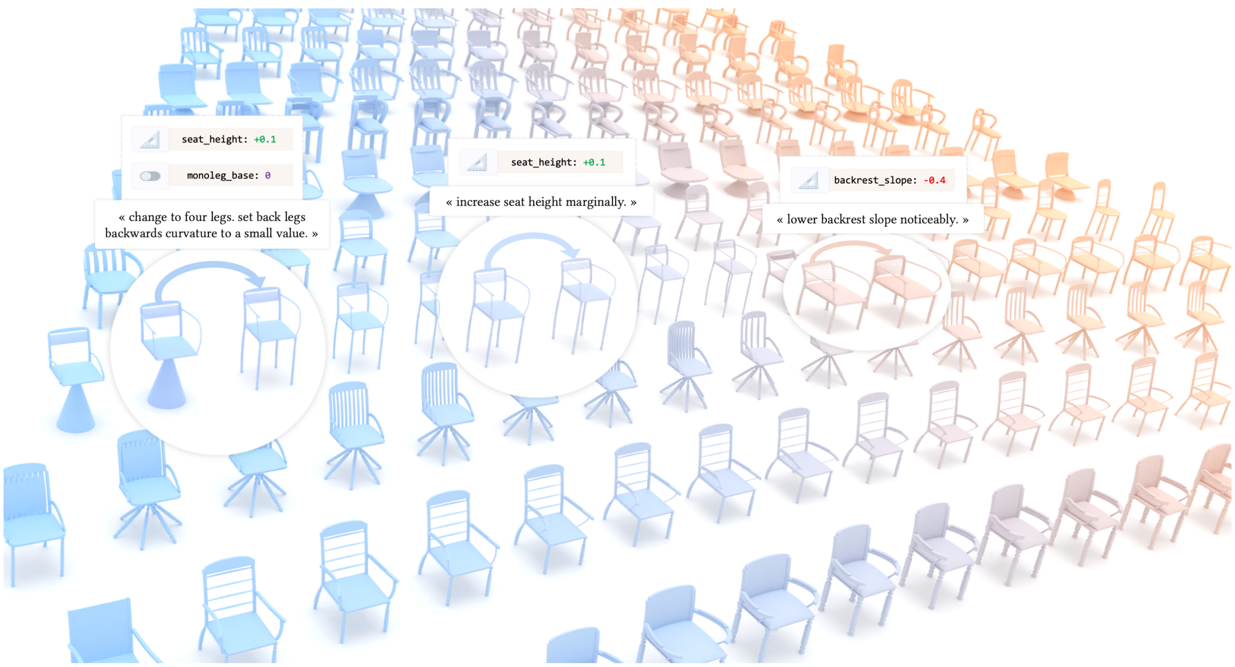

Figure 1. **Visualizing shape chains from ShapeWalk.** Our dataset consists of 158K unique shapes connected through 26K edit chains, with an average length of 14 chained shapes. We illustrate the interpolation process by coloring shapes based on their proximity to the starting shape (**blue**), and the ending shape (**orange**). Each consecutive pair of shapes is associated with precise language instructions describing the applied edits. For each shape transition, we also provide a precise edit vector $\overline{\theta_{ij}}$ describing the parameter changes necessary to transition from one shape to the next.

Abstract

Editing 3D shapes through natural language instructions is a challenging task that requires the comprehension of both language semantics and fine-grained geometric details. To bridge this gap, we introduce ShapeWalk, a carefully designed synthetic dataset designed to advance the field of language-guided shape editing. The dataset consists of 158K unique shapes connected through 26K edit chains, with an average length of 14 chained shapes. Each consecutive pair of shapes is associated with precise language instructions describing the applied edits. We synthesize edit chains by reconstructing and interpolating shapes sampled from a realistic CAD-designed 3D dataset in the parameter space of the GeoCode shape program. We leverage rule-based methods and language models to generate accurate and realistic natural language prompts corresponding to each edit. To illustrate the practicality of our contribution, we train neural editor modules in the latent space of shape autoencoders, and demonstrate the ability of our dataset to enable a variety of language-guided shape edits. Finally, we introduce multi-step editing metrics to benchmark the capacity of our models to perform recursive shape edits. We hope that our work will enable further study of compositional language-guided shape editing, and finds application in 3D CAD design and interactive modeling.

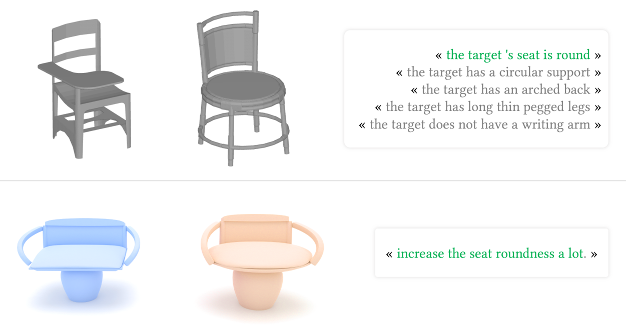

Figure 2. **Comparing ShapeTalk with our work.** We compare the ShapeTalk [3] dataset (*top*) with our work (*bottom*). For an equivalent edit instruction (in green), ShapeTalk provides pairs of shapes with many factors of variation, while we generate synthetic pairs of shapes with a single clear varying factor.

1. Introduction

Whether in the realms of computer-aided design, virtual reality environments, or digital content creation, the process of refining and enhancing 3D visual data often involves intricate adjustments to geometric shapes, textures, and lighting. Furthermore, the necessity for precise modifications adds an additional layer of complexity, as inaccuracies can have profound implications on the final output. This labor-intensive nature of 3D data editing not only hinders workflow efficiency but also poses a barrier for individuals without specialized skills, limiting the democratization of 3D content manipulation. As such, there exists a compelling need for innovative solutions that streamline and democratize the 3D editing process, making it more accessible to a broader range of users. But how to train models with the ability to perform these complex edits?

To that end, we introduce ShapeWalk, a dataset aiming at addressing the challenges inherent in the intricate and skill-intensive nature of 3D data editing. Our dataset consists of synthesized chains of 3D shapes, connected through edit vectors associated with precise language instructions describing the applied edits. It contains 158K unique shapes connected through 26K edit chains, with an average length of 14 chained shapes. Our generation method is scalable and can yield an indefinite number of realistic shape chains, and extended to any 3D domain and shape program. To our knowledge, our dataset is the first to provide a large-scale collection of realistic 3D shape edit chains with precise language instructions, and matching ground-truth edited shapes. Our method yields specific edits undiluted by other factors of variation, as illustrated in Figure 2.

To validate the usefulness of our dataset, we train neural editor modules in the latent space of shape autoencoders [1], and demonstrate the ability of our dataset to train models capable of performing a variety of shape edits. We learn latent mappings in the space of a frozen shape autoencoder, and show that our method can be applied on shapes unseen during the training of the neural editor. Taking advantage of the ground-truth shape chains provided by our dataset, we introduce multi-step editing metrics inspired from the field of trajectory prediction [4, 26] to evaluate the quality of our models.

Our contributions can be summarized as follows:

- **Chained Shape-Editing Dataset:** We introduce a synthetic dataset of chained edits, with associated language instructions and ground-truth edited shapes. This is the first dataset of its kind, and is designed to facilitate the study of compositional shape editing.
- **Neural Shape Editor:** We introduce a neural editor module trained on our dataset, and demonstrate its ability to perform a variety of shape edits. We show that our method can be applied on shapes unseen during the training of the neural editor.
- **Chained-Editing Metrics:** With the introduction of ground-truth shape chains, we propose multi-step editing metrics to evaluate the ability of neural editors to perform recursively applied edits.

2. Related work

3D Shape Programs. A significant body of work [9, 11, 12, 14, 19] is dedicated to exploring the use of procedural programs for 3D shape representation. Representing 3D objects as visual programs has many advantages. Representing shapes as visual primitives enhances interpretability [11, 14], as programs are by definition human-readable and thus understandable by human experts. Shape programs are more compact [9, 27] than their usual 3D shape modalities, and can be used to represent shapes more efficiently. Programs can also be composed to create shapes [11, 14], or decomposed into smaller programs. Ideally, shape programs can be edited [13] to modify the underlying geometry they represent.

Our dataset consists of a collection of synthetic shapes created utilizing a backbone mesh-generating shape program. Specifically, we build from GeoCode [17], a 3D shape synthesis technique addressing the challenge of mapping high-fidelity geometry to an editable parameter space. GeoCode introduces a procedural program enabling the generation of high-quality mesh outputs with a balanced blend of interpretability and fine control.

Language-Guided 3D Shape Editing. Various works have explored the use of natural language instructions to guide 3D shape editing. Recent efforts [16, 20, 21] propose to leverage pre-trained CLIP [24] models to align

3D shapes with a given text prompt. Early efforts in that direction [16] included optimizing meshes to progressively align them with CLIP embeddings, leading to important computational overhead. CLIP-Sculptor [21], a more recent work, leverages a voxelized representation and a discrete latent space conditioned on CLIP's image-text embeddings to perform fast and fidel shape edits without shape optimization.

Shape Editing Datasets. Other works leverage large text-aligned 3D shape datasets and require significant manual annotation effort. ShapeCrafter [8] generates 3D shapes incrementally from text using a neural network, evolving with additional phrases. ShapeCrafter is designed for recursive shape edition and utilizes a VQ-VAE [23] model to represent shapes as discrete codes. This method exhibits consistent shape-text alignment with gradual evolution. ChangeIt3D [3] introduces ShapeTalk, a large dataset for describing 3D shape differences. The framework facilitates language-based editing of 3D models without requiring 2D to 3D conversion methods, and learns a shape editing model by learning contrasts between sampled shape pairs.

ShapeTalk [3] is the most similar work to our proposed dataset, and is the only publicly available dataset specifically designed for language-guided 3D shape-to-shape editing. ShapeTalk is a remarkable contribution in the field of language-guided shape editing, and is one of the first works to leverage large-scale 3D shape datasets to facilitate the study of language-guided shape editing. However, this work has some limitations that we attempt to address in our work. Collecting a dataset of shape differences is a challenging task requiring considerable manual annotation effort, and gathering a large number of 3D shapes may not be feasible for all domains. Furthermore, ShapeTalk is composed of edit contexts (i.e. shape pairs and edit instructions) which are hard to separate into fine-grained, composable edits. Most of the time, singular shape edits do not have have an exact ground-truth in the form of a source and target shape pair. We illustrate this important distinction in Figure 2. The availability of singular shape edits with an associated ground-truth would ease benchmarking the quality of shape editing methods, and facilitate the training process of editing models. ShapeTalk also does not provide a mechanism to generate edit chains, which are necessary to study compositional shape editing. Finally, ShapeTalk does not easily enable the study of shape editing in a compositional manner. In contrast, our dataset is **1)** synthesized by augmenting a small set of diverse shapes and can easily be scaled up, **2)** composed of fine-grained edits and coarse-grained edits with exact ground-truths, and **3)** separated into edit chains, which are designed to facilitate the study of compositional shape editing.

3. ShapeWalk

Our dataset contains 158K shapes split into a *random* and *realistic* set. With each edge connecting two consecutive shapes, we also produce a text instruction generated using the parameter changes necessary to transition from one shape to the next. We detail the generation method of our dataset here, and summarize the process in Figure 3.

3.1. Dataset

Definition. Our dataset can be defined as a collection of directed graph paths (dipaths), denoted as $\mathcal{P}^{(k)} = (S^{(k)}, E^{(k)}, f^{(k)})$. For each shape chain $\mathcal{P}^{(k)}$ of length l, we denote:

- $S^{(k)} = \{s^{(k)}_{\theta_1}, \dots, s^{(k)}_{\theta_N}\}$, the set of distinct shape nodes composing the chain. Each shape is defined by a set of parameters $\theta_i \subseteq \Theta$, where Θ is the linear parameter space of our shape program.

- $E^{(k)} \subseteq \{(s^{(k)}_{\theta_i}, s^{(k)}_{\theta_j}) \mid s^{(k)}_{\theta_i}, s^{(k)}_{\theta_j} \in V^{(k)},\ j = i + 1\}$, the set of edges linking each consecutive pair of shapes.

- $f^{(k)} : E^{(k)} \mapsto \{(\overline{\theta_{ij}},\ \mathbf{p_{ij}}) \mid \overline{\theta_{ij}} \subseteq \Theta,\ \mathbf{p_{ij}} \in \Sigma\}$, a function mapping each edge to a vector $\overline{\theta_{ij}} \subseteq \Theta$ and a set of text instructions $\mathbf{p_{ij}}$. $\overline{\theta_{ij}}$ defines the parameter changes, or edits necessary, to go from shape i to j. $\mathbf{p_{ij}}$ is a set of text instructions describing this edit in natural language.

Generation. To generate our dataset, we start by reconstructing a set $S \subseteq \mathcal{S}$ of realistic 3D CAD shapes into the space of shapes covered by our shape program $\mathcal{S}_\Theta \subset \mathcal{S}$. To that end, we utilize shapes from the 3DCoMPaT++ [15, 22] dataset, a realistic, industry-based 3D CAD dataset. We employ this dataset to avoid overlap with ShapeNet [5], and to ensure diversity and visual quality of the shapes.
We define a visual similarity function $d : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}$ between two shapes $s_i, s_j \in S$ as the feature-wise mean squared error between the original and reconstructed meshes' renderings, in the space of a pre-trained ResNet-50 [10] encoder $\phi_R : \mathbb{R}^{h \times w \times 3} \mapsto \mathbb{R}^d$.

$$d(s_i, s_j) = \|\phi_R(f_R(s_i)) - \phi_R(f_R(s_j))\|_2$$

Where $f_R : S \mapsto \mathbb{R}^{3 \times h \times w}$ is a rendering function. The set of reconstructed shapes can then be defined as:

$$\underset{\substack{S_R \subseteq S \\ |S_R| = (1-\alpha)|S|}}{\arg\min} \sum_{s \in S_R} d(s, \widehat{s})$$

Where $\widehat{s}$ is the reconstructed shape, defined as:

$$\widehat{s} = f_\Theta \circ \phi_\Theta(s)$$

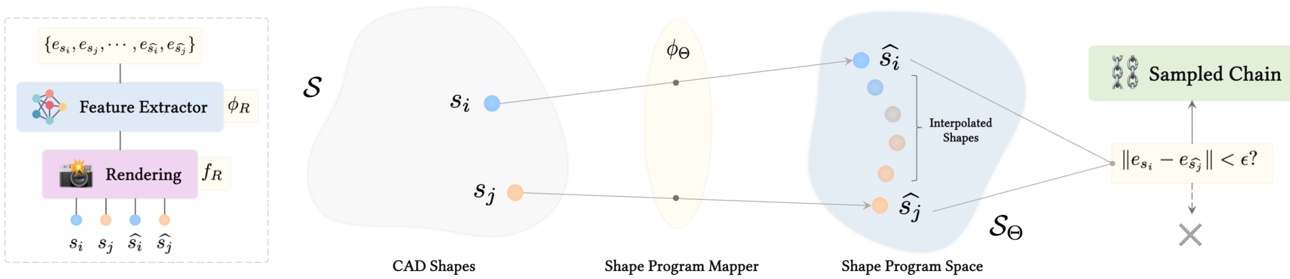

Figure 3. **Detailling the shape generation pipeline of ShapeWalk.** Realistic 3D CAD shapes are reconstructed from the 3DCoMPaT++ [22] dataset into the GeoCode [17] shape program parameter space, using a mapping function ϕ_Θ. Reconstructed shapes with an error over a fixed threshold are discarded using a visual similarity function d, which is based on rendered feature similarity. Filtered pairs of shapes are then interpolated in the parameter space of the shape program, to generate shape chains $\mathcal{P}^{(k)}$.

With $f_\Theta : \Theta \mapsto \mathcal{S}$ the shape program mapping shape parameters to a 3D mesh, and $\phi_\Theta : \mathcal{S} \mapsto \Theta$ an encoder based on DGCNN [17, 25] which samples points from a shape surface and regresses corresponding shape parameters in Θ.

We reconstruct shapes by first sampling pointclouds from the surface of the original meshes, and then feeding them into the DGCNN [25] encoder fine-tuned [17] to regress the parameters of the shape program. We discard a ratio of $\alpha = 0.08$ of the shapes with the highest reconstruction error, which we measure as the feature-wise mean squared error between the original and reconstructed meshes' renderings, in the space of a pre-trained ResNet-50 [10] encoder. This threshold is selected empirically by visually inspecting the shapes in the upper part of the reconstruction error distribution. In our dataset instance building on 3DCoMPaT++ shapes, we filter out a total of 89 shapes during this process, leading to a set of $|S_R| = 1113$ reconstructed shapes. This process can be scaled up to any dataset size and any shape domain, as long as a corresponding shape program is available.

Shape interpolation. After reconstructing a set S_R of realistic shapes, we generate a set of shape chains $\mathcal{P}^{(k)}$ by interpolating between the parameters of shape pairs.
To that end, we first sample a set of shape pairs $(s_{\theta_\mathbf{i}}, s_{\theta_\mathbf{j}}) \in \mathcal{S}_P \subseteq S_R \times S_R$, and partition them into $L = 10$ levels of proximity. These proximity levels are based on the feature-wise mean squared error between the original and reconstructed meshes' renderings, in the space of a pre-trained ResNet-50 [10] encoder.

We define the set of shape pairs $\mathcal{S}_P^{(l)}$ at level $l \in [\![1, L]\!]$ as:

$$\mathcal{S}_P^{(l)} = \{(s_{\theta_\mathbf{i}}, s_{\theta_\mathbf{j}}) \in \mathcal{S}_P \mid \overline{d(s_{\theta_\mathbf{i}}, s_{\theta_\mathbf{j}})} \in \left[\frac{l-1}{L}, \frac{l}{L}\right]\}$$

Where $\overline{d}$ is a visual similarity function, min-max normalized across all pairs of shapes.

For each pair $(s_{\theta_\mathbf{i}}, s_{\theta_\mathbf{j}}) \in \mathcal{S}_P^{(l)}$, we then interpolate between the shape parameters $\theta_\mathbf{i}$ and $\theta_\mathbf{j}$ to generate a set of intermediate shape parameters $(\theta_\mathbf{i}, \ldots, \theta_{\mathbf{i+(N-2)}}, \theta_\mathbf{j}) \in \Theta^N$. The number of intermediate shapes N is defined as the number of differing parameters between $\theta_\mathbf{i}$ and $\theta_\mathbf{j}$, that is:

$$N = |\{k \mid \theta_\mathbf{i}[k] \neq \theta_\mathbf{j}[k]\}|$$

When the proximity level is low, edits necessary to transition from one shape to another are larger in intensity as the shapes are more different, and smaller when l is smaller. The ordering of intermediate edits is randomly sampled using a dependency-aware algorithm (for example, parameters relating to armrest height or width are sampled after the addition of armrests).
As illustrated in Figure 1, this generation process leads to plausible interpolated shapes, and to fine-grained and realistic sequences of edits with detailed metadata.
We summarize the process of generating realistic shape chains in Figure 3.

Random set. While the *realistic* set is based on reconstructed shapes from the 3DCoMPaT++ [15, 22] dataset, the *random* set is based on a collection of random shapes generated using the shape program. This alternative set aims at covering a large space of the parameter space of the GeoCode shape program. This subset is generated by systematically sampling various combinations of parameters within defined ranges determined by a minimum edit intensity, but does not necessarily result in realistic shapes.

3.2. Text Instructions

Rule-based generation. Given an edit vector $\overline{\theta_{ij}}$ describing the parameter changes necessary to transition from shape i to j, we generate a set of text instructions $\mathbf{p_{ij}}$ describing this edit in natural language. We first map each parameter $\theta_i[k]$ composing the edit vector to a natural language name, and map the magnitude of the

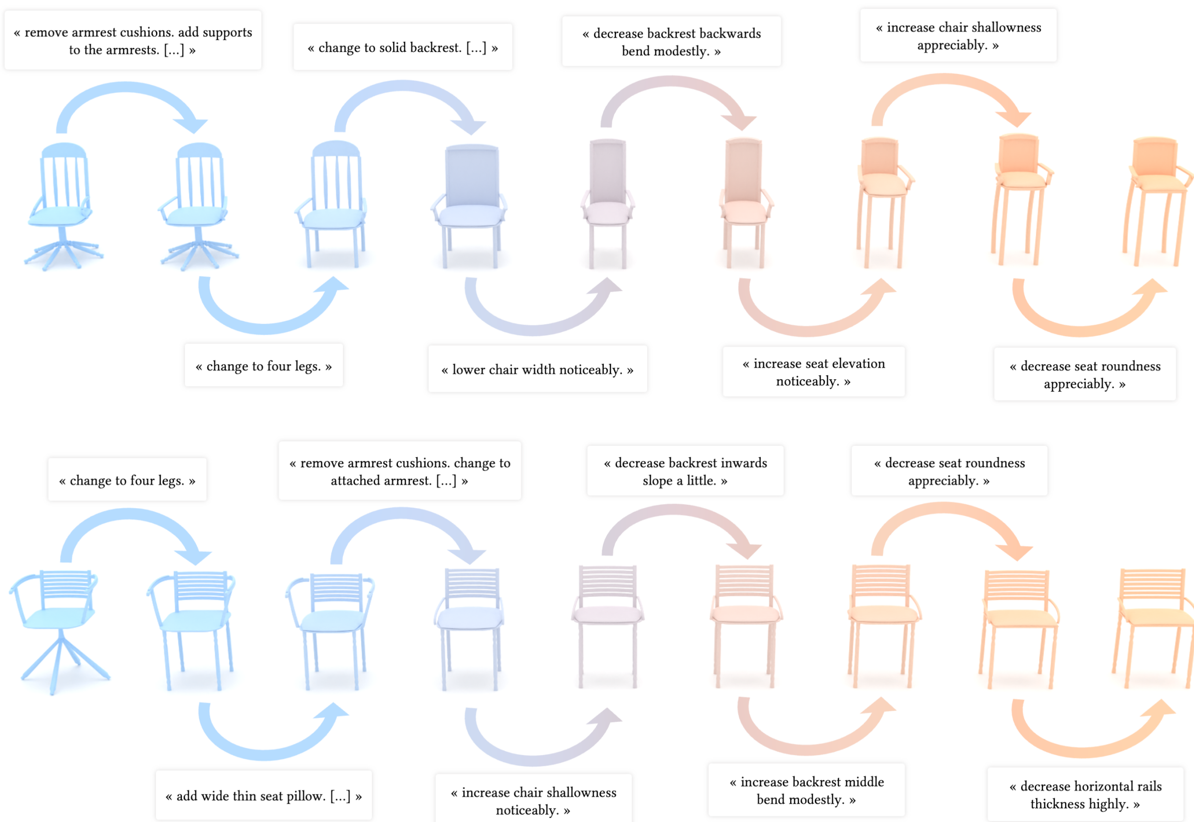

Figure 4. **Dataset chains samples.** We feature in this figure shape chains from our dataset with associated generated text instructions. Each shape chain is composed of a sequence of shapes, with each consecutive pair of shapes associated to a set of text instructions describing the edits necessary to transition from one shape to the next. Overall, generated text instructions are accurate and provide a detailed description of the edits applied to the shapes. We include in the supplementary material additional samples of generated shape chains, alongside shape chains generated for two additional **table** and **vase** categories.

parameter change to a natural language intensity depending on the parameter type. Boolean parameters are also described appropriately, for example by using the word *add* or *remove* when describing the addition or removal of a part.

Augmentation. This rule-based instruction generation leads to perfectly accurate instructions, but may lack diversity. To alleviate this issue, we **1)** randomly sample adjectives and adverbs to characterize the intensity of edits, **2)** randomly invert parameter names and magnitude directions (e.g. "*increase armrests straightness*" will be augmented to "*decrease armrests bend*" for the same edit). As a final augmentation, we utilize a T5 [18] transformer model fine-tuned for paraphrasing to generate additional instructions for each edit vector. We use the Parrot library [6] to filter generated paraphrases based on a fluency and adequacy score.

We summarize the process of generating text instructions in the supplementary material.

4. Neural Shape Editing

4.1. Problem

We are concerned with the task of editing a shape s_i given a sequence of natural language edit instructions. We want to learn an editing function $f_E : \mathcal{S} \times \Sigma \mapsto \mathcal{S}$, where $\mathcal{S}$ is the space of input shapes, and Σ is the space of natural language edit instructions, able to compose edits from a starting shape s_1 to an ending shape s_N, given a sequence of edit instrutions $\{\mathbf{p_t}\}_{t=1}^{N}$.

$$\widehat{s_N} = f_E(\phi_T(\mathbf{p_{12}}),\ f_E(\phi_T(\mathbf{p_{23}}),\ \dots\ f_E(\phi_T(\mathbf{p_{kN}}),\ s_1)))$$

Where $N = |\mathcal{P}|$ is the length of the edit chain, and $\phi_T : \Sigma \mapsto \mathbb{R}^{D_T}$ is a text encoder. At the end of the edit sequence, we want to recover a shape $\widehat{s_N}$ as close as possible to the ground-truth shape s_N. Our proposed dataset provides a ground-truth for every intermediate shape s_k, which we leverage to train and evaluate our method.

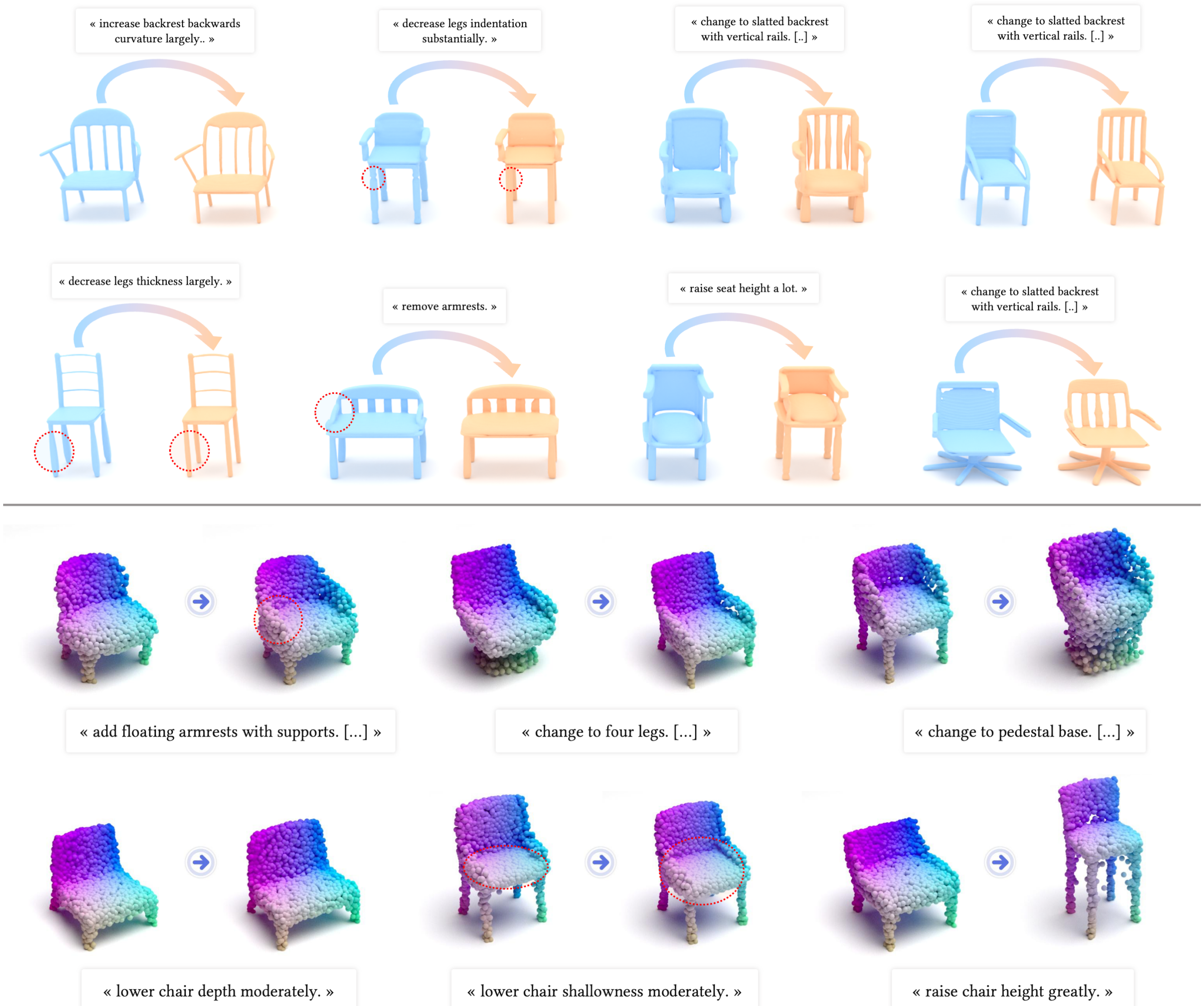

Figure 5. **Visualizing synthesized pairs of edits using our latent editors.** We illustrate the diverse range of edits generated by our proposed latent mapper, showcasing its ability to capture nuanced variations in the input data, for 3DS2VS latents (top two rows) and PC-AE latents (bottom two rows). Each pair demonstrates the original input on the left and the corresponding synthesized output on the right, with the associated caption. We abbreviate instructions which are too long to fit in the figure, and indicate when text is omitted with an ellipsis. For more subtle changes, we circle the areas of interest in both source and target shapes. Overall, edits are consistent with the input instruction, preserve shape identity while generating plausible output shapes.

4.2. Objective

One of the main advantages of training a shape editing model with ShapeWalk compared to other works [2, 8] is the availability of exact ground-truth edited shapes for each edit instruction (see Section 2). We thus formulate the objective function of our learning problem as a simple $\mathcal{L}_2$ loss between the latents of the synthesized shape and the ground-truth shape. For all pairs of shapes (s_i, s_j) and their corresponding edit instructions $\mathbf{p_{ij}}$, we minimize the following loss:

$$\mathcal{L} = \|f_E(\phi_E(s_i),\ \mathbf{p_{ij}}) - \phi_E(s_j)\|_2$$

Where $\phi_{AE} = \phi_E \circ \phi_D$ is a shape autoencoder, and $\phi_E : \mathcal{S} \mapsto \mathbb{R}^{D_E}$ denotes its encoder component.

4.3. Models

For ϕ_{AE}, we experiment with PC-AE [1] and 3D2VS [28] models both pre-trained on ShapeNet [5]. Note that our method is agnostic to the choice of shape autoencoder and can be adapted to a variety of shape representations. For ϕ_T, we use a pretrained BERT [7] model.

Both ϕ_{AE} and ϕ_T are frozen during training, and we only train the parameters of our latent mapper f_E.

For PC-AE, we formulate our latent mapper as a neural module which for each tuple $(\phi_E(s_i),\ \mathbf{p_{ij}},\ \phi_E(s_j))$ predicts an edit vector $\widehat{\theta_{ij}} \in \mathbb{R}^{D_E}$ in the feature space of ϕ_E, and adds it to the latent of the input shape s_i to generate the latent of the output shape s_j:

$$f_E(\phi_E(s_i),\ \mathbf{p_{ij}}) = \widehat{\theta_{ij}} + \phi_E(s_i)$$

We utilize two variants of our latent mapper f_E: one in which the edit vector $\widehat{\theta_{ij}}$ is predicted directly, and one in which the edit vector $\widehat{\theta_{ij}}$ is predicted as a product of a normalized edit direction $\widehat{v_{ij}} \in \mathbb{R}^{D_E}$ and a magnitude $\widehat{m_{ij}} \in \mathbb{R}$ separately.

For 3D2VS which encodes shapes as latents sets of higher resolutions, we use a transformer-based latent diffusion model instead to generate edited latents. We concatenate the input shape latent $\phi_E(s_i)$ with the noised edited shape latent $\phi_E(s_j + \epsilon_t)$ and feed the BERT text embeddings to the cross-attention layers of the transformer blocks to predict the added noise. We illustrate our architecture for the latent mapper in the supplementary material.

5. Experiments

5.1. Chained Shape Editing

Comparison Models. For PC-AE, we experiment with three variants of our latent mapper module:

- DIRECTGEN directly predicts the edited shape latent without regressing an edit vector.
- LATEFUSION follows the network proposed in [3], and first passes the shape embeddings through an encoder. Encoded shape features are then concatenated with text embeddings and fed into a neural module which finally predicts the edit vector.
- OURS is a simple multi-layer perceptron (MLP) which directly takes the *context* (shape and text embeddings) as input to predict an edit vector.

For all of these variants, we experiment with various bottleneck dimensions and number of layers (in subscript).

We further decompose these variants into *coupled* and *decoupled* versions, where we predict a normalized edit direction $\widehat{v_{ij}} \in \mathbb{R}^{D_E}$ and a magnitude $\widehat{m_{ij}} \in \mathbb{R}$ separately. We provide in appendix the full architecture details and training hyperparameters employed to train our models.

Note that we do not compare with ChangeIt3D [2] as their neural-listener distillation method is not applicable in our context: we train our models with a direct feedback signal from the ground-truth edited shapes.

Single-step metrics. We propose to measure the quality of our editing steps both before and after shape reconstruction. Since we experiment on pointclouds, we use the scaled Chamfer Distance between the reconstructed original shape s_i and the reconstructed edited shape $\widehat{s_j}$:

$$d_{\text{CD}} = \frac{1}{|\widehat{s_j}|} \sum_{x \in \widehat{s_j}} \min_{y \in s_i} \|x - y\|_2 + \frac{1}{|s_i|} \sum_{y \in s_i} \min_{x \in \widehat{s_j}} \|x - y\|_2$$

We also use the $\mathcal{L}_2$ distance between the latents of the original shape s_i and the edited shape s_j:

$$d_{\mathcal{L}_2} = \|\phi_E(s_i) - \phi_E(s_j)\|_2$$

Multi-step metrics. To extend the single-step metrics to chained shape generation, we take inspiration from the trajectory prediction literature [4, 26] and propose appropriate metrics for our task.

Given a shape distance function $d : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}$, a chain of ground-truth reconstructed shapes $\{s_1, \ldots,\ s_N\}$, and a set of recursively generated shapes $\{\widehat{s_1}, \ldots,\ \widehat{s_N}\}$, we define the *average edit error* as the average distance between the generated shapes and the corresponding ground-truth shapes:

$$\text{A}_d = \frac{1}{N} \sum_{t=1}^{N} d(s_t, \widehat{s_t})$$

And the *final edit error* as the distance between the last shape in the chain and the corresponding ground-truth shape:

$$\text{F}_d = d(s_N, \widehat{s_N})$$

We report results with both metrics for $d = d_{\text{CD}}$ and $d = d_{\mathcal{L}_2}$.

Results. We provide qualitative results of edited pairs generated by our latent mapper in Figure 5. Overall, we observe that our method is able to generate plausible edits that are consistent with the input instruction, and preserve the identity of the input shape. However, our method is limited by the quality of the shape autoencoder. We notice that the PC-AE editor is unable to perform very fine-grained edits which are not properly captured by the autoencoder, while visible in the ground-truth shapes. We explore the possible causes of these limitations in Section 5.2.

We provide quantitative results for our chained editing metrics averaged across $|\mathcal{P}| \in \{10, 15, 20\}$ in Table 1, and detail per-chain length results in the appendix. Directly predicting the edited shape latent (DIRECTGEN) leads to poor edit predictions across all metrics, although increasing the number of layers may improve the results. Overall, decoupling the edit direction and magnitude leads to better results. Our coupled $\text{OURS}_{512\times4}$ variant performs better than the coupled LATEFUSION_{512} variant on the Chamfer Distance, but worse on $\mathcal{L}_2$ distance.

Model	decoupled?	Averaged $\forall|\mathcal{P}|$			
		$\mathrm{F}_{\mathrm{CD}\times\mathbf{1e^4}}$	$\mathrm{A}_{\mathrm{CD}\times\mathbf{1e^4}}$	$\mathrm{F}_{\mathcal{L}_2}$	$\mathrm{A}_{\mathcal{L}_2}$
LateFusion_{1024}	✗	2.856	2.621	1.444	1.251
LateFusion_{512}	✗	2.719	2.609	1.462	1.290
LateFusion_{256}	✗	2.874	2.651	1.509	1.283
$\text{Ours}_{512\times 8}$	✗	3.208	2.822	1.703	1.437
$\text{Ours}_{512\times 4}$	✗	2.990	2.703	1.589	1.351
LateFusion_{1024}	✓	2.711	2.568	**1.405**	**1.245**
LateFusion_{512}	✓	3.002	2.708	1.451	1.254
LateFusion_{256}	✓	3.309	2.848	1.552	1.324
$\text{Ours}_{512\times 8}$	✓	2.782	2.584	1.497	1.290
$\text{Ours}_{512\times 4}$	✓	**2.670**	**2.524**	1.447	1.266

Table 1. **Chained shape editing ablation.** We report baseline results for our proposed chained shape editing task, averaged across all chain lengths $|\mathcal{P}| \in \{10, 15, 20\}$. Both the average final and average edit error are reported for the Chamfer Distance (CD) and $\mathcal{L}_2$ distance ($\mathcal{L}_2$) metrics.

Parameter	Accuracy
seat height	1.000
backrest curvature	1.000
object width/height/depth	0.978
seat roundness	0.978
top bar thickness/height	0.961
legs thickness	0.956
legs bending/curvature	0.944
adding/removing handle cushions	0.750
number of legs/backrest rails	0.663
legs roundness/indentation	0.587
Avg (Random)	0.884
Avg (Realistic)	0.969

Table 2. **Parameter-wise accuracy for shape edit recognition.** We report the accuracy of our edit detector for each parameter in the *random* set, alongside the average accuracy for the full *random* and *realistic* sets. Overall, we observe that our classifier is able to recognize edits corresponding to high change edits, but fails to differentiate between subtle changes.

5.2. Recognizing Shape Edits

Problem. To provide additional insight into the quality of our latent mapper and the difficulty of predicting specific edits, we train a binary classifier to discriminate from a pair of shapes (s_i, s_j) on whether the corresponding edit instruction $\mathbf{p_{ij}}$ was applied to s_i or s_j. We train a binary classifier $f_C : \mathcal{S} \times \mathcal{S} \mapsto \{0, 1\}$, where $f_C(s_i, s_j) = 1$ if the edit instruction $\mathbf{p_{ij}}$ was applied to s_i, and 0 otherwise.

Method. Similarly to our latent mapper architecture for PC-AE, we use an MLP-based shape latent encoder to encode the input shape latents into the bottleneck dimension, and project text features using a single linear layer to the same dimension. We then concatenate the two feature vectors and pass them through an MLP predictor outputting probabilities for the two classes s_i and s_j. We train our classifier using a binary cross-entropy loss.

Training and evaluation. We train our classifier on the *realistic* set and evaluate it on the full *random* shape set, which has a restricted vocabulary of edit instructions. We evaluate the accuracy of the classifier on the full set, as well as on subsets of the data corresponding to specific types of edits.

Results. In Table 2, we provide accuracy results for shape edit recognition, depending on the edit type. We also provide the global accuracy on the *random* and *realistic* sets. We observe that our classifier is able to recognize edits corresponding to shape parameters associated with a high degree of shape variation (like seat height and width/height of the shape), but struggles with parameters which are more subtle (like legs roundness, handle cushions). Our classifier is able to detect edits related to global width/height/depth adjustments with an accuracy of 97.8%. This accuracy drops to 94.4% for legs bending/curvature, which is a more subtle change in the shape, and to 58.7% for legs roundness/indentation. We also note that the model does not perform well on edits related to the number of parts, i.e. adding or removing a number k of legs or backrest rails.

We impute these discrepancies to several factors. The PC-AE shape autoencoder we employ may be unable to properly reconstruct fine-grained details in the ground-truth and edited shapes, which renders the task of detecting subtle changes more difficult. Pointcloud resolution is also a limitation, as fine-grained parameters like leg indentation may not be properly sampled. Finally, learning a latent mapper and a classifier that can accurately count the number of parts to add or remove may also be a difficult task of its own, and require more domain-specific inductive biases.

6. Conclusion

In conclusion, this paper introduces the ShapeWalk dataset, designed for advancing compositional shape editing guided by natural language instructions. The dataset comprises 158K unique shapes connected through 26K edit chains, synthesized from a realistic CAD-designed 3D dataset. Language instructions for applied edits are provided, alongside exact ground-truth edited shapes. Our method requires zero human annotation effort, and can be scaled up indefinitely. The usefulness of the dataset is demonstrated by training neural editor modules in the latent space of shape autoencoders [1]. Evaluation metrics inspired by trajectory prediction literature [4, 26] are introduced, offering a quantitative assessment of the quality of recursively edited shapes.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 2, 6, 8

[2] Panos Achlioptas, Ian Huang, Minhyuk Sung, Sergey Tulyakov, and Leonidas Guibas. Shapetalk: A language dataset and framework for 3d shape edits and deformations. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023. 6, 7

[3] Panos Achlioptas, Ian Huang, Minhyuk Sung, Sergey Tulyakov, and Leonidas Guibas. ShapeTalk: A language dataset and framework for 3d shape edits and deformations. In *CVPR*, 2023. 2, 3, 7

[4] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016. 2, 7, 8

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. In *arXiv*, 2015. 3, 6

[6] Prithiviraj Damodaran. Parrot: Paraphrase generation for nlu., 2021. 5

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 6

[8] Rao Fu, Xiao Zhan, Yiwen Chen, Daniel Ritchie, and Srinath Sridhar. Shapecrafter: A recursive text-conditioned 3d shape generation model, 2023. 3, 6

[9] Aditya Ganeshan, R. Kenny Jones, and Daniel Ritchie. Improving unsupervised visual program inference with code rewriting families. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 3, 4

[11] R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. Shapeassembly: learning to generate programs for 3d shape structure synthesis. *ACM Transactions on Graphics*, 39(6):1–20, 2020. 2

[12] R. Kenny Jones, David Charatan, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. Shapemod: macro operation discovery for 3d shape programs. *ACM Transactions on Graphics*, 40(4), 2021. 2

[13] R. Kenny Jones, Homer Walke, and Daniel Ritchie. Plad: Learning to infer shape programs with pseudo-labels and approximate distributions. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022. 2

[14] R. Kenny Jones, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. Shapecoder: Discovering abstractions for visual programs from unstructured primitives. *ACM Transactions on Graphics*, 42(4), 2023. 2

[15] Yuchen Li, Ujjwal Upadhyay, Habib Slim, Ahmed Abdelreheem, Arpit Prajapati, Suhail Pothigara, Peter Wonka, and Mohamed Elhoseiny. 3D CoMPaT: Composition of Materials on Parts of 3D Things. In *ECCV*, 2022. 3, 4

[16] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 Conference Papers*. ACM, 2022. 2, 3

[17] Ofek Pearl, Itai Lang, Yuhua Hu, Raymond A. Yeh, and Rana Hanocka. Geocode: Interpretable shape programs. In *arXiv preprint arxiv:2212.11715*, 2022. 2, 4

[18] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *arXiv*, 2023. 5

[19] Daniel Ritchie, Paul Guerrero, R. Kenny Jones, Niloy J. Mitra, Adriana Schulz, Karl D. D. Willis, and Jiajun Wu. Neurosymbolic models for computer graphics. *Computer Graphics Forum*, 42(2), 2023. 2

[20] Aditya Sanghi, Hang Chu, Joseph G. Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malekshan. Clip-forge: Towards zero-shot text-to-shape generation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022. 2

[21] Aditya Sanghi, Rao Fu, Vivian Liu, Karl D.D. Willis, Hooman Shayani, Amir H. Khasahmadi, Srinath Sridhar, and Daniel Ritchie. Clip-sculptor: Zero-shot generation of high-fidelity and diverse shapes from natural language. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023. 2, 3

[22] Habib Slim, Xiang Li, Mahmoud Ahmed Yuchen Li, Mohamed Ayman, Ujjwal Upadhyay Ahmed Abdelreheem, Suhail Pothigara Arpit Prajapati, Peter Wonka, and Mohamed Elhoseiny. 3DCoMPaT++: An improved large-scale 3d vision dataset for compositional recognition. In *arXiv preprint arxiv:2212.11715*, 2023. 3, 4

[23] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017. 3

[24] Suchen Wang, Yueqi Duan, Henghui Ding, Yap-Peng Tan, Kim-Hui Yap, and Junsong Yuan. Learning transferable human-object interaction detector with natural language supervision. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022. 2

[25] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. In *SIGGRAPH*, 2019. 4

[26] Kaiping Xu, Zheng Qin, Guolong Wang, Kai Huang, Shuxiong Ye, and Huidi Zhang. *Collision-Free LSTM for Human Trajectory Prediction*, pages 106–116. 2018. 2, 7, 8

[27] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. Capri-net: Learning compact cad shapes with adaptive primitive assembly. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022. 2

[28] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics*, 42(4):1–16, 2023. 6