

viz2viz: Prompt-driven stylized visualization generation using a diffusion model

Jiaqi Wu, John Joon Young Chung, and Eytan Adar



Fig. 1: viz2viz generated samples (original data inset). From left to right: *bird eye view of green forests with many trees, blue ocean with ships, grey city with many buildings, orange deserts*; *realistic stacks of red coca-cola coke cans, brown tea cups, glass wine bottles, starbucks paper coffee cups*; *realistic pink tulips*; *Ukiyo-e style side view of red wooden roller coaster, Ukiyo-e style blue sea waves and surfers*. Note that in some examples in this paper, we select prompts to demonstrate the capabilities of viz2viz rather than for their aesthetic or design qualities.

Abstract—Creating stylized visualization requires going beyond the limited, abstract, geometric marks produced by most tools. Rather, the designer builds stylized idioms where the marks are both transformed (e.g., photographs of candles instead of bars) and also synthesized into a ‘scene’ that pushes the boundaries of traditional visualizations. To support this, we introduce viz2viz, a system for transforming visualizations with a textual prompt to a stylized form. The system follows a high-level recipe that leverages various generative methods to produce new visualizations that retain the properties of the original dataset. While the base recipe is consistent across many visualization types, we demonstrate how it can be specifically adapted to the creation of different visualization types (bar charts, area charts, pie charts, and network visualizations). Our approach introduces techniques for using different prompts for different marks (i.e., each bar can be something completely different) while still retaining image “coherence.” We conclude with an evaluation of the approach and discussion on extensions and limitations.

Index Terms—Stylized visualization, generative algorithms, stable diffusion

1 INTRODUCTION

Most visualization software aims for the accurate reproduction of data using simple, *abstract* 2-dimensional graphical marks: the rectangles (\square) of the bar chart; the arcs (\triangleleft) of the pie; and the mix of circles (\circ), rectangles (\square), and occasional stars (\star) in scatter plots [28]. In many situations, these representations are insufficient for a creative design vision. Graphical ‘embellishments’ require additional styling and design work. A designer might choose to *stylize* a visualization by replacing simple abstract marks with alternative representations: candles instead of bars, slices of fruit instead of arcs in a pie chart, and flowers instead of circles in a scatter plot. Beyond the simple replacement of marks, visualization styling can use alternative artistic forms and media (a sketch, a painting, a photograph, clay, etc.). Our particular focus is on stylized forms that represent a significant stylistic change to how the mark is represented (e.g., stacked coffee cups), but are still recognizable in their original idiomatic forms (e.g., a bar chart)¹. Due to the uniqueness of many of these representations, for a designer to achieve their vision, they need additional tools, skills, and procedures. For example, to manually produce the charts in Figure 1, a designer would need to create and photograph a physical model or

use sophisticated painting tools. Producing a single instance, let alone iterating over a design, is a significantly tedious effort. Instead, we offer viz2viz, a set of workflows that use generative approaches to produce these images. Our approach allows designers to take “starting” visualizations (basic bar charts, area charts, pie charts, and network graphs) and produce stylized visualizations using simple text prompts.

For this work, we build on the systems for generative image creation (e.g., DALL-E2 [22], Stable Diffusion [25], and Midjourney). These models are powerful, but on their own difficult to “contain.” Without providing any guidance, the output has no relationship to the underlying data. For example, one could ask for “a photograph of a bar chart made of lit candles with backlighting,” but the candle heights will not correspond to any actual data distribution. Novel architectures such as img2img [19] or ControlNet [38] allow us to input a starting image (e.g., a standard bar chart). However, these approaches suffer from the opposite problem in that the source image becomes too limiting—the system is unable to produce a stack of coffee cups when guided by a tall 2D rectangle. Our goal is to produce stylized visualizations that are simultaneously flexible to a wide range of creative visions but that at the same time adhere to the underlying distributions of the data. For example, the pie piece that looks like clouds should still be in the correct fraction of the original data. The flowers representing two nodes in a network visualization should be connected if the data calls for it. With viz2viz, we offer a set of “recipes” that can work across a wide array of inputs (both data and prompts) to create both photorealistic and nonphotorealistic images. To identify the range of possible stylized visualizations, we constructed a taxonomy of these forms using human-created examples.

An advantage of our approach is that it does not rely on explicit

• *Jiaqi Wu, John Joon Young Chung, and Eytan Adar are with the University of Michigan. E-mail: wujiaq,jjyc, eadar@umich.edu.*

¹While there are many examples of these visualizations, there does not appear to be a unifying name for them (some are called embellished, infographics, photovisualizations). We refer to the broad category as *stylized visualizations*.

training using visualization (plain or stylized). This is vital as it is difficult to find a large enough set of examples where we simultaneously have data, descriptions, and stylized forms. Instead, we focus on identifying ways of breaking apart the marks used in the plain form of the visualizations. By ‘deconstructing’ the visualizations as needed, viz2viz can focus on applying different generative pipelines in a targeted way (e.g., making a bar into the rough shape of a building). Once completed, viz2viz can reconstruct a cohesive visualization (e.g., multiple buildings side by side) and apply additional generative steps to produce a more composed image (e.g., a more cohesive skyline). In this work, we describe how the basic high-level recipe—composed of sketch, synthesis, and refine steps—can be implemented using combinations of image processing procedures and modified versions of generative pipelines (e.g., depth2Img, img2img, ControlNet, etc.). The viz2viz approach demonstrates how both traditional image processing (e.g., masking, transformation/translation, etc.) and generative elements can be used in concert to produce a kind of controlled creativity. Beyond the four specific visualization types (bar, area, pie, and network), we offer some guidance on how specific workflows can be built for other types. While our focus in this work is on *textual* prompts as a way to drive the generative system, we describe possible extensions to alternative prompt mechanisms.

Our contributions in this work are:

- Identifying a design space and taxonomy of stylized visualization.
- Design and implementation of viz2viz, a general recipe with specific workflows to support the creation of stylized visualization.
- A guide to modifying the viz2viz ‘recipe’ to different types of charts and prompts.

2 RELATED WORK

Two lines of research motivate viz2viz: (1) the visualization literature, with a focus on human- and machine-driven ways of generating stylized visualizations; and (2) image generation with diffusion models.

2.1 Visualization Construction

Stylized visualizations are primarily created using medium-specific tools *after* being generated through visualization software (e.g., the image file generated by Matplotlib, R, Excel, Altair, etc.). Visualization generators are rarely built to support extensive stylization. For example, in a survey of more than 40 tools, only three supported ‘infographic customization’ [18]). Instead, designers utilize vector graphics software (e.g., Adobe Illustrator) or ‘raster’ editors (e.g., Adobe Photoshop) to create stylized forms [2]. In response, new tools have been created to bridge visualization software and vector tools (e.g., D3 and Illustrator [3]). Inside editors, vector representations can be replaced with icons, vector art, and embellished in various ways. Tools such as Data Illustrator [17], Charticulator [24] and (to some extent) Tableau are designed to provide visualization and design feature simultaneously. These allow for the creation of highly stylized vector-art style visualizations. However, none supports bitmap editing that might be required to manipulate photographs or other mediums. With raster images (e.g., photographs), stylization is more complex, but can be achieved by image editors. Image operations (e.g., copy, paste, translating, and transforming) are used to achieve both the desired look, ensuring that the image conforms to the encodings (e.g., dimensions that properly encode the underlying data) and overlaying idiomatic structures (e.g., adding axes or labels).

An alternative approach to styling visualizations is the application of templates or styles to basic forms. In some cases, the stylized forms are driven by creating hand-crafted examples that are bound to data (e.g., [14, 35]. For example, DataInk [35] allows end users to draw glyphs by hand (e.g. a sketched tree). Such tools are highly flexible and support different visualization idioms. However, they still require a certain design skill and may involve post-processing work to ensure design variation. InfoNice [33] represents an alternative approach in which custom (human-designed) glyphs, such as a partially filled wine bottle, are used. Other approaches (e.g., Diatoms [4]) can propose novel but potentially abstract glyph types. Our assumption with viz2viz

is that designers have a conceptualization of the marks they want to generate, which may not be glyphs (abstract or otherwise).

The idea of using machine learning to create visualizations has been applied in different ways [16, 34, 39]. However, most approaches are built to suggest visualization types given data or to modify existing visualizations (e.g., annotations or captions). The most related examples offer ways of performing a kind of visualization style transfer (e.g., [5, 7, 20, 29]). While powerful, these are largely constrained to situations where there are existing examples or where the transformation is from one simple mark type or idiom to another [10]. It is also notable that a consistent limitation in the systems described thus far is that they cannot generate photorealistic visualizations.

The use of photographs or photoimageries in the generation of visualizations is a difficult problem. The DataQuilt system provides one way for designers to work with images to stylize visualizations [37]. Users can borrow components and stylized features from existing image sources and turn them into elements to which data can be mapped (e.g., resizing an extracted image of a coffee cup based on a data property). Most ‘found’ images may not correspond to the underlying data. For example, a designer would find it challenging or impossible to find or create photographs of objects that are correctly proportioned to the underlying data (e.g., stretched bottles, stacks of cups with correct shadows, etc.). The Infomages systems proposed an alternative approach where thematic photographs serve as the ‘background’ of a visualization (e.g., a picture of lips for a visualization of makeup sales) [6]. Infomages allows users to semi-automatically embed traditional chart data into existing images (e.g., coloring different parts of the lips into a pseudo-pie chart). Although the system can find images that may work well with the data, it does not generate these and either ‘fills’ or ‘overlays’ the visualization on top of the image. The interaction between foreground visualization marks and background structure is beyond the scope of viz2viz.

Our goal for viz2viz is to start with a standard idiom and use natural language to describe the desired styling. Thus, there is a natural relationship to the use of natural language for visualization construction (see the extensive survey in [30]). Although most tools of this type take language (and data) to generate standard visual forms, some may have a richer set of templates (e.g., [8]). Systems such as Lumina [13] offer an interesting approach in that they seek to ‘understand’ what the data represents (e.g., a country or specific person) and offer styling based on this understanding (e.g., a country flag or image of the individual). Similarly, MetaGlyph [36] proposes vectorized glyphs based on data semantics (e.g., a layered hamburger if the data is about fast food ingredients). With viz2viz, we assume that baseline visualization has been constructed in advance (using whatever tool is best) and natural language-based prompts can be used to drive the stylization process without requiring examples or template forms.

Finally, we note that visualization designers have begun to utilize generative approaches to build visualizations. Mostly, this is used for constructing backgrounds or cut-out images that can be placed inside existing marks (e.g., a generated picture of a state’s flower for each state in a map). Most experiments note significant additional post-processing steps to ensure that the output conforms to the input distribution. These experiments have not succeeded in mapping to specific idioms or consistently encoding data. However, these experiments have led to incredibly creative works (e.g., [9]) and illustrate a need for better tooling. With viz2viz, we propose to address this challenge. We define a common ‘recipe’ that supports stylized visualization generation that are simultaneously broad across a wide range of styles and forms, but also ensure consistency to the original data.

2.2 Diffusion Models

There are a number of generative image models in use today. However, of particular interest to us are the diffusion probabilistic models that gradually denoise a noise-added variable to get to a sample of interest. This approach was proposed by Sohl-Dickstein et al. [31] but has evolved significantly in terms of efficiency and steerability. For example, **Latent Diffusion Model (LDM)** [26] applies the diffusion process over a lower-dimensional compressed representation of images

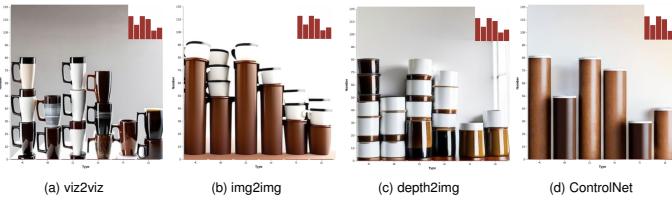


Fig. 2: Result for *stacks of realistic coffee mugs filled in with coffee on a white table* using the different approaches. We compare (a) our system to (b) img2img (strength=0.8), (c) depth2img (strength=0.95), and (d) ControlNet (with Canny edge detection, strength=0.95).

(latent) (rather than actual pixel space) for more efficient training.

With new diffusion models, it is also possible to perform image generation tasks with text inputs, or *prompts* [22, 27]. This is achieved by encoding text input into guiding vectors using pre-trained contrastive vision-language models like CLIP [21]. Combining this approach with LDM, **Stable Diffusion (SD)** is a latent text-to-image diffusion model capable of generating photorealistic images given any text input. As SD is open source, the community has rapidly evolved the approach to generate better results. For example, a parameter that SD users often adjust is *guidance scale* [11], which determines how strongly the text input guides the diffusion process. To allow more controllability, developers devised many *pipelines*, end-to-end image generation processes with an SD model. In constructing viz2viz, we have adopted a generic recipe to generate stylized recipes. However, depending on the properties of the data, prompts, and desired idioms, each recipe calls for some variations in the diffusion model. In many cases, we leverage and extend existing approaches to fit our specific constraints.

Image-to-Image (**img2img**) [19] is a steerable approach conditioned by an additional input image. In img2img, instead of starting the diffusion process from the random latent image, it leverages a noise-added input image as the initial latent. Because of this, the resulting image has visual similarities to the initial image while being influenced by the guiding prompt. In img2img, a *strength* value (0-1) is often used to adjust the level of influence of the initial image. Based on img2img, **Inpainting** [1] uses an image mask as additional input so that the resulting image is generated only in a specific area. Other approaches more strictly condition the visual structure of the generated image [12, 15]. For example, Depth-to-Image Generation (**depth2img**) [26] allows conditioning of the diffusion process with a depth map of the initial input image (inferred through MiDaS [23]). Similarly to img2img, depth2img allows users to specify the level of transformation with strength. However, it better maintains the structural information of the original image. In addition to these, Zhang and Agrawala [38] provided a way of adding a conditional control module that accelerates model tuning. This approach, or **ControlNet**, allows various types of condition images (e.g., depth map, edge, or segmentation information) that provide the structural information of generations. For generating stylized visualization, ControlNet with depth or Canny edge detection can be an alternative when depth2img and img2img fail to generate coherent and stylized visualizations.

As we describe in more detail below, applying existing models directly to the visualization problem is problematic. It is difficult to both meet the stylization prompt and preserve the precise information of the chart. Applying these models alone does not produce reliable results (Figure 2). For example, with low strength, img2img (Figure 2b) keeps the structure information of the initial image (a good thing), but is unable to use the prompt to transform the bars (it simply adds “noise”). With high-strength, the image may be transformed but will not adhere to the underlying data distribution. Structure-conditioned approaches maintain visualized data, but often follow structure information too strictly and will not stylize the shape of the marks (e.g., Figure 2d). Moreover, if the user wants a visualization with multiple objects as marks with specific sizes, positions, and shapes (e.g., the first bar should be coffee cups, the second should be bottles, etc.), existing controllable approaches would struggle to meet these requirements. With viz2viz,

we can generate stylized visualizations more reliably by combining and modifying these existing pipelines. Our output adheres to both the data and the prompt.

3 TAXONOMY OF STYLIZED VISUALIZATION

To understand the space of stylized visualizations we collected over 250 examples. We included those where standard idioms were used but where marks were ‘concrete.’ That is, objects other than abstract geometric structures were used to represent the data. To collect these, we searched for terms such as ‘infographic’ and ‘photoviz’ and included those that matched our definition. We used Pinterest’s recommendation engine to find additional examples.

The authors grouped these images and found three main categories. Figure 3 illustrates a number of synthetic variants based on real-world examples. These were created using DALL-E 2 with some modification in Illustrator to add labels and emphasize certain marks. The images demonstrate the potential of generative approaches. However, as we did not specify the data for these examples, the distributions are meaningless. We compare these examples to those generated by viz2viz in the supplement.

It is worth mentioning what we excluded. Some visualizations may have been stylized, but under a broader definition than we use here. Work by Federica Fragapane, Giorgia Lupi and Stefanie Posavec can be considered highly stylized. However, their designs often leverage novel abstract forms or idioms. We also do not consider abstract marks translated to the same mark in another medium (e.g., a bar chart painted in oil paints). These are examples of ‘style transfer’ rather than the stylization we are interested in (for these, a simple use of ControlNet might suffice). Below, we describe the three main labels we apply to our examples.

3.1 Style and Medium

Style and medium refer to the artistic style used and the medium (paint, photography, sculpture, etc.). Due to their extreme variations, we do not create sub-classes within this category (classifying art is well outside the scope of this work). However, we have found recurrent themes in the found images.

Many of the designs we identified used photographs of physical objects for visualizations. Artists such as Sarah Illenberger, Peter Ørntoft, and Oliver Uberti, and the Data Made of Things project (<https://datamadeofthings.tumblr.com/>) are representative of this style of design. In Figure 3 examples include images *a-c*, *h*, *i*, and *l*. We view this type as distinct from images made using a different medium (e.g., clay) which is then photographed.

Non-photographic stylization often involves some sort of paint or drawing medium [32]. In real-world examples, these visualizations are usually hand-drawn (digitally or on physical medium). The data journalist Mona Chalabi produces examples of this type, and some of Nigel Holmes’ work can also be considered in this category. Examples in the figure include *e*, *f*, *g*, *j* and *k*. The attractiveness of this type is that they allow for the creation of non-photorealistic representations (e.g., a time-series as the back of a drawn fish). While it is possible to conceive of surreal objects as photographs (e.g., the stretched bottles in *i*), designers and viewers may prefer a non-photographic medium.

While we note a few common themes in hand-crafted stylized visualizations, we emphasize that there are constraints that may have led to their popularity. For example, certain cultural aesthetic preferences and norms may have resulted in an over-representation of certain image styles. More critically, certain images may simply be too difficult to create without a generative tool. For example, a visualization that looks like a photograph but is surrealist, may not be possible without extensive image editing work. Because they are challenging to create, we may simply not see many of them. Generative approaches may change the landscape of stylized visualizations.

3.2 Composed and Decomposed

Our second category is based on *scene cohesiveness*. Specifically, we are interested in whether the visualization is *composed* or *decomposed*. With composed images, objects representing the marks and

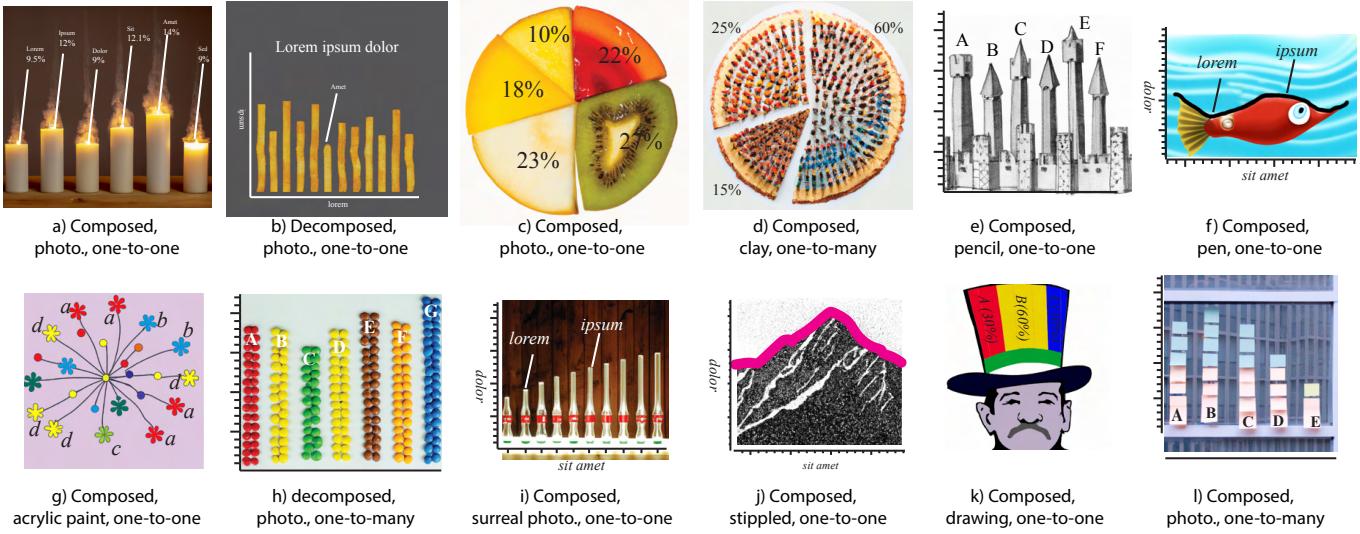


Fig. 3: Various examples of stylized visualizations based on a wide set of ‘found’ categories. These were created by us using DALL-E 2 (e.g., *A photograph of a bar chart made of lit candles on a table with backlighting*) but without specific data prompt. Generated images were modified in Illustrator and Photoshop for annotation (e.g., labels, axes).

backgrounds interact in some way (i.e., objects sit on top of each other, shadows from one bar chart element fall onto another, etc.). Examples of this category include Figure 3 *a*, *c-g*, *i* and *j-l*. In these, objects either “touch” or are part of the same scene (candles on the same table, post-it notes in the same window, or bars in the same hat). Decomposed images may have similar marks but the objects do not interact with each other or their background. For example, if we take separate photographs of each element in the bar chart and cut and paste them into one image, we would consider this a decomposed visualization (e.g., Figure 3 *b* and *h*).

3.3 One-to-One and One-to-Many

Finally, we categorize visualizations with how a mark is constructed. Specifically, we are interested ‘of what things’ it is made. In many situations, one mark in the original idiom is represented by one mark in the stylized visualization. One bar to one candle or one French fry or one mountain to one time-series are examples (see Figure 3 *a-c*, *e-g* and *i, j, k*). Alternatively, one-to-many examples are those in which one mark is composed of smaller ‘parts’ such as one pie slice as a crowd or one bar chart as many candies (see *d, h* and *l*). one-to-many examples are often seen as pictograms where icons are used to construct a ‘mark.’

In some instances, we may find ambiguity within this category. For example, we may use a mountain range to represent a time-series. The range (one) is composed of multiple mountains (many). In these situations we opt to classify the visualization based on what we viewed as the ‘simplest’ (in the gestalt sense) visual explanation (in this case, the whole range).

Our goal for viz2viz is to support the generation of as broad a range of these stylized visualizations as possible. In addition to ensuring that the marks encode the data correctly, we would like to generate both constructed and deconstructed images, photo-realistic and non-photorealistic, and marks made of both single and multiple objects.

4 VIZ2VIZ: GENERATING STYLIZED VISUALIZATION

We introduce viz2viz, a workflow that generates stylized visualizations using diffusion-based image generation models. Due to the wide variation of visualization inputs and stylized outputs, the core of viz2viz is a central “recipe” that can be adapted to different scenarios. viz2viz is based on the same 3-step framework, which combines existing and novel architectures and pipelines (Figure 4). While there is a central core recipe, we find that no single workflow works for all visualization types and all prompts. Instead, we offer a number of alternative pathways that the designer can use to achieve their desired result. We leave

completely automated selection of a workflow’s sub-routines for future work. Once defined, a viz2viz workflow can be run end-to-end without intervention. However, designers can intervene at different places to achieve different effects (e.g., modifying a visualization sketch before running the rest of the workflow).

As input, viz2viz receives the plain visualization, prompts, and additional control variables. The plain input visualizations (bar chart, pie charts, etc.) can be generated through various tools (in our implementation we use a variety of Python-based toolkits). Prompts are textual descriptors of the stylization we would like to apply. In viz2viz, prompts are composed of different parts that can be applied selectively in the diffusion steps. Specifically, a prompt consists of: (a) contextual prompt that will apply to the entire process (e.g., *a photograph of a bar chart*), (b) one or more sub-prompts describing what the marks should be transformed to (e.g., all bars should be *lit candles* or bar one should be a *skyscraper* and bar two should be a *pagoda*), (c) an optional background prompt (e.g., *a library* or *a simple gradient*).

4.1 The viz2viz Recipe

The first step of viz2viz is **Sketch**. In this step, viz2viz creates a roughly stylized visualization or sketch. As we describe below, viz2viz breaks each mark (and background) into individual components and generates a stylized representation for each (e.g., each bar in the bar chart is transformed with the prompt). It ensures that marks do not get blended together, which can happen if they are close together or touching. Once re-assembled, the sketch visualization may not be visually coherent as each part may have different lighting or variations in style. The sketch visualization is then passed to a second step: **Synthesize**. Here, the visualization is made coherent by using the diffusion pipelines to the entire image. Figure 4 (left) shows the general workflow with screenshots of what the visualization looks like at each step.

Although visualization after the synthesis step tends to be visually more coherent than sketch visualization, we have found that in some cases an additional **Refine** step is necessary to clean up the image (e.g., blurriness). This final pipeline produces better image resolutions and stronger details. Unlike synthesize, the pipelines minimally transform the image to improve quality while preserving visualized information. Below, we describe each step in detail.

4.1.1 The Sketch Step

The sketch step includes: 1) **Mark Pre-Generation**, which transforms each mark in the plain visualization into a stylized object; 2) **Background Pre-Generation**, which generates a stylized background; and

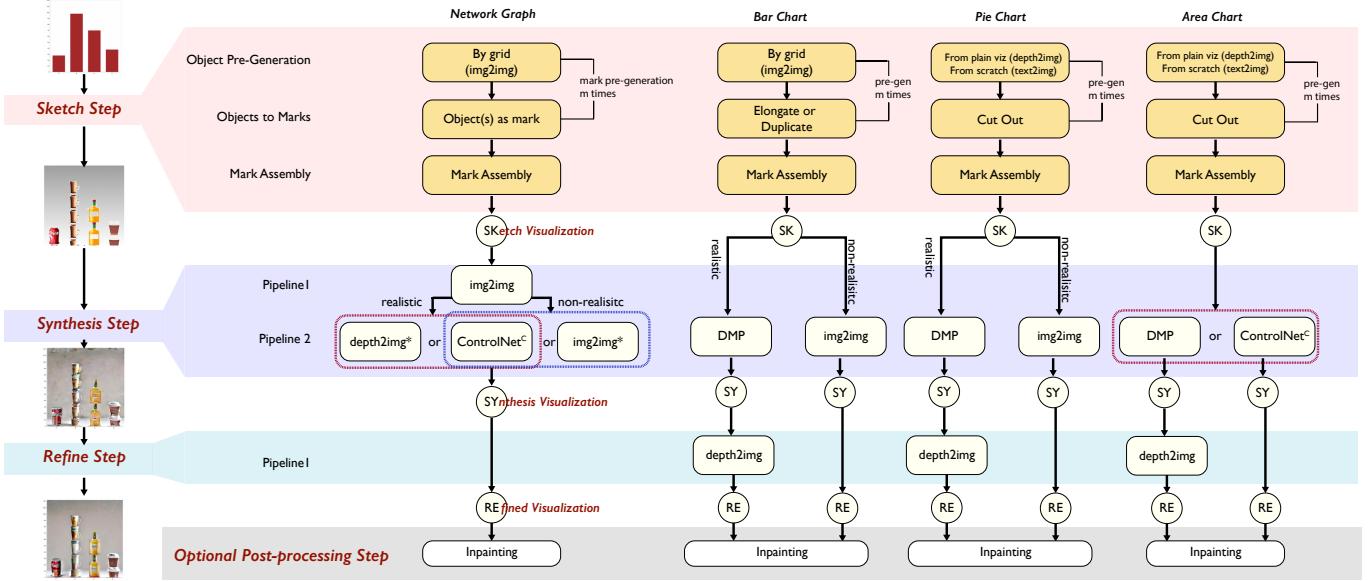


Fig. 4: Workflow of viz2viz. The leftmost path describes the generic recipe, with specific variants appearing to the right. Note that ControlNet^C means the ControlNet pipeline with Canny edge condition, and depth2img*/img2img* mean the Edge-enhanced-depth2img/img2img pipelines, which add an image of the network edges on top of the initial image before the depth2img/img2img pipelines.

3) Mark Assembly, which assembles pre-generated marks and background into a sketch visualization.

Mark Pre-Generation In viz2viz, we use several approaches to generate each mark. As discussed in Section 3.3, a ‘single’ mark can actually be different things: a single object (a single bottle representing the entire mark), multiple objects (multiple stacked cups), or a cohesive part of a larger object (e.g., a pie chart). We describe different approaches to this problem below.

Pre-generating Objects: In many cases, when ‘sketching’ replacement marks, viz2viz begins by utilizing the marks created in the original plain visualization: bars and pie slices are separated, and nodes and edges are split. Depending on the number of marks (e.g., many bars or many nodes), viz2viz will position the marks in $N \times N$ grids. By creating a clear separation, we can ensure that individual objects are created for each original mark. For example, creating a *social network graph made of fruits* requires many fruits, each somewhat aligned with the original color and size of the node. Each original node mark—for example, a circle—from the plain visualization is placed in a $N \times N$ grid. With this modified image, we can perform an img2img diffusion (Figure 5a). We have found through experimentation that in many situations, it is not necessary to scale the separated mark to the correct relative dimensions. For example, with a bar chart, we simply need the right number of bars, but they can be equal-sized (Figure 5b). The correct scaling can be recovered in later steps. Although the grid approach is effective in many situations, we have found that it is not vital in certain situations. For example, we do not need to build a grid when the number of marks is limited (e.g., a single dataset area chart), where there is sufficient separation in the objects (e.g., pie slices that are not touching), or when we want the generated image to very strictly follow the contours of the original mark (e.g., an area chart). In these cases, we can use the original marks, in their original layout, as input to the pre-generation diffusion phase (see Figure 6). We note that as viz2viz conditions the generation with the mark, image-conditioned diffusion approaches (e.g., img2img or depth2img) are the most appropriate for this step. For situations where the prompt consists of multiple shape sub-prompts (e.g., we want different objects for each bar) the sketch pre-generation step is run multiple times, once for each sub-prompt.

In some cases, pre-generation using the original mark may not produce a good sketch. For example, this occurs when the prompt asks for an unconventional camera angle or a mark composed of many items



Fig. 5: Pre-generation by grid in bar chart and network graphs.



Fig. 6: Direct pre-generation for pie charts (prompt: *realistic slice of cookies*) and area chart (prompt: *fire flames*).

(e.g., a pie chart slice made of a birds-eye view of many individuals in a crowd). In these cases, viz2viz can produce a better sketch by not being restricted to the original mark. However, as this generated image is not constrained by the original mark’s shape we need to transform it.

Object(s) to Mark(s): viz2viz turns generated objects into one or multiple stylized marks in a number of ways: a) turning an object to a mark directly (e.g., a generated flower → a network graph node); b) adjusting multiple generated objects (e.g., a stack of coke cans) so that they create a mark (a stack of coke cans → a single bar in a bar chart); or c) modifying the generated object to fit within the constraint of the original mark (e.g., an apple pie → a slice of pie). These are all done through various combinations of image transformations (see Figure 7).

In the simplest case, no transformation (beyond translation and basic rescaling) is needed. For example, if we want each node of a network graph to be stylized as an apple, we can simply generate a set of apples (by original mark grid) and then replace the nodes with these images.

When only parts of the generated objects are useful as marks, viz2viz will simply cut out a piece of the object by using a mask on the generated image (see cut-out in Figure 7). We found this approach most useful for more complex shapes (e.g., pie slices or time series in an area plot). This approach allows us, for example, to generate an entire



Fig. 7: Image operations to translate object(s) to mark(s).

apple pie or a large picture of a crowd, and cut out piece corresponding to the pie chart’s mark.

In some cases, the produced mark is not in the correct dimensions and cannot be rescaled easily (i.e., both width and height consistently rescaled). When a mark’s dimensions require the generate image to have an alternative dimension, viz2viz performs elongation (Figure 7). Note that in many situations we do not want to stretch the entire object. For example, we might want to only elongate a specific part of the object (e.g., the body part of a pencil, not the tip or eraser). In our current implementation, viz2viz does not understand the semantics of a generated object. Instead, we adopt a heuristic that works well in practice: the generated object is split into three equal parts (the ‘head’, ‘body’ and ‘tail’). When applying this workflow, viz2viz will only elongate the body (central third) of the generated object to the needed dimension while keeping the head and tail dimensions fixed.

When the design calls for ‘stacking’ objects (e.g., stack of coffee cups), viz2viz can duplicate generated objects and place them on top of each other (see duplication in Figure 7). Pie chart stylization can also use this approach to fill the pie area with duplicates of a specific object.

Background Pre-Generation If a background prompt is provided, viz2viz uses txt2img to generate a background for the sketch visualization. Because backgrounds can visually conflict with the, arguably more important, visualization marks, viz2viz blurs and brightens the background. If no background prompt is provided, viz2viz can generate a simple gradient background in this step.

Mark Assembly After creating a sketch for each mark, viz2viz reassembles these marks by placing each corresponding generated image into the position specified by the original visualization. For example, viz2viz will ‘disassemble’ the grid of generated objects and reassemble them into the appropriate locations in the network graph.

4.1.2 The Synthesize Step

The sketch produced by the earlier step of viz2viz generally contains the marks in generally the right shapes, sizes, and placement. However, these images are not cohesive: cropped parts of images may feel incomplete, cups may be floating in space, marks may have lighting and style variations that do not make for a good design. A Synthesize step is used to remedy many of these issues. As with the Sketch step, there are a number of alternative workflows that can be used depending on the input visualization type and prompt. Because of the difficulty in maintaining coherence among the connected elements of a network (which are not always captured in the sketch step), those visualizations require a two-step process. For network graphs, the Synthesize step first uses an img2img pipeline with low strength level. This smooths unnatural object renditions and compositions visible in sketch visualization.

All visualizations at this point undergo the same (conceptually) Synthesize step to ensure that the various marks can look more coherent. In reality, different pipelines can be used here to achieve different effects. We return to why someone might prefer one pipeline over another when discussing specific chart types. In addition to img2img, depth2img, and ControlNet pipelines that can be applied to the sketched visualization, we also designed three additional pipelines: **Diffusion with Multiple Prompts (DMP)**, **Edge-enhanced depth2img** and **Edge-enhanced img2img**. The first, DMP, is used to support designers who want different objects for each mark (i.e., with multiple sub-prompts). The latter two are used to ensure that edges are retained in network visualizations, as thin edges can easily vanish during a diffusion process.

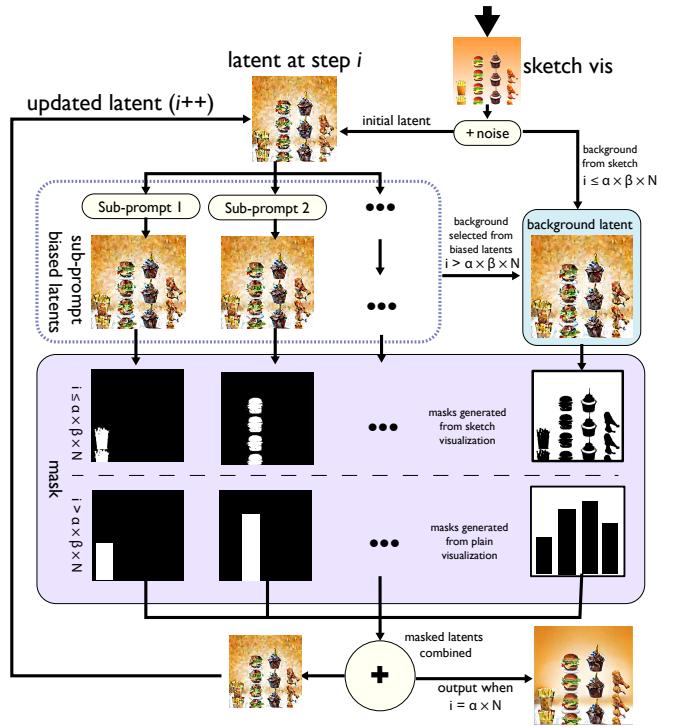


Fig. 8: Example illustration of the DMP pipeline. In this example, there are four sub-prompts: *fries*, *hamburgers*, *cupcakes*, *fried chicken*

Diffusion with Multiple Prompts (DMP) The DMP pipeline is used when we would like to specifically apply different sub-prompts to different marks. With standard pipelines, multiple prompts may lead to blending that is not desirable. For example, if we synthesize a pie chart with the prompt *chocolate and apple pie slices* with img2img or depth2img, we may find slices that combine both apple and chocolate together. Approaches like inpainting and masked application of diffusion to a region of the image will not work here as each area will largely ignore its neighbors and the image as a whole. This is very much against the goals of the Synthesize step where we would like a coherent image—just one made of different objects in this case.

Our solution is a **Diffusion with Multiple Prompts (DMP)** pipeline. DMP can link different marks in the visualization to different sub-prompts in a single diffusion process, while minimizing the mix of different prompts in a single mark (Figure 8). The basic idea of DMP is to perform diffusion on the whole latent separately for each prompt and then combine the multiple resulting latents back into a single latent. Since this process is done iteratively for each step, the different areas guided by different sub-prompts could also consider the rest of area, resulting in a coherently synthesized visualization.

At the start of the DMP pipeline, the process is similar to img2img. DMP adds a specific level of noise (according to the strength α) to the sketch visualization latent. This becomes the initial latent. For each step ($i = 1 \dots \alpha \times N$, where N is the number of whole diffusion steps when generating an image from a pure noise), DMP diffuses each mark separately using depth2img with different sub-prompts (e.g., ‘hamburger’, ‘fries’, etc.). The result is a set of latents, one per mark. In our example, each of these latents becomes biased towards one of the sub-prompts. This ensures that marks begin to ‘synthesize’ together (i.e., the stack of hamburgers begins to influence the stack of fries) but also means they blend (i.e., the hamburger stack becomes more fries-like). To prevent the latter problem, DMP masks each bar. With β , which is a hyperparameter between 0 and 1, when $i \leq \alpha \times \beta \times N$, the image is filtered using a mask generated from the sketch visualization (ensuring a tight adherence to the shape of the sketch). When $i > \alpha \times \beta \times N$, we switch to a mask defined by the plain visualization.

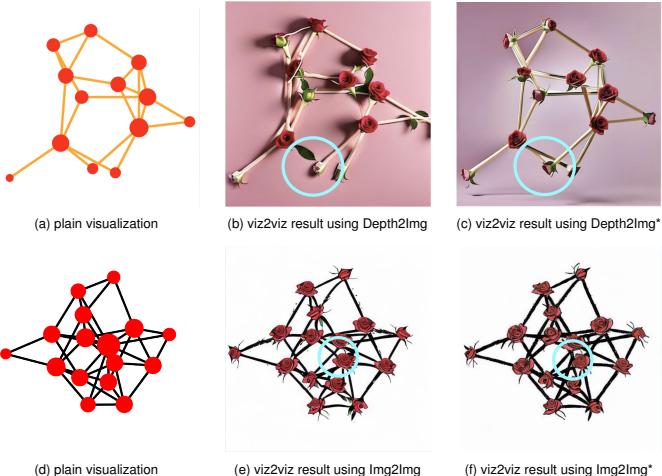


Fig. 9: Result comparison of original depth2img/img2img and depth2img*/img2img*. Prompt for (b) and (c): *realistic red roses on a background of pink velvet*. Prompt for (e) and (f): *hand drawn style roses*. The cyan circle highlights where edge information is enhanced.

This larger mask allows for slightly more flexibility as the diffusion has stabilized. Similarly to each mark, we apply the same idea to the latent for the background. When $i \leq \alpha \times \beta \times N$, the background latent is the sketch visualization latent with added noise (and the mask is derived from the sketch visualization). When $i > \alpha \times \beta \times N$, the background latent is selected from one of the prompt-biased mark latents. We choose the background latent iteratively (switching between sub-prompt outputs) at each step to minimize the influence of any one sub-prompt on the background. Intuitively, this allows the background to develop independently based on the sketch in early phases and then becomes more directly impacted by the newly diffused marks. The masked latents from the marks and background are combined, and this becomes the latent for step $i + 1$. This strategy has two benefits: (1) it reduces computational complexity (i.e., no additional diffusion for the background); and (2) we get higher visual coherence than diffusing the background with a separate prompt. Note that if there is only one sub-prompt (e.g., all bars have the same prompt), DMP reduces to depth2img.

Edge-enhanced depth2img and img2img The advantage of depth2img is that it preserves structural information robustly while combining stylized marks. With img2img, we have found that it can have high performance on non-realistic styles. Unfortunately, for network graphs, both pipelines tend to spoil ‘fragile’ edges. To deal with this, we have a slightly augmented version of these pipeline (depth2img* and img2img*, respectively). These overlay the edges of the original network visualization on top of the initial smoothed sketch visualization. The result is that edges that vanished in the sketch are ‘forced’ back into the image. This strategy can maintain edge information in the synthesis visualization (Figure 9).

4.1.3 The Refine step

While visualizations generated in the Synthesize step can be an improvement over sketches (in that marks ‘interact’). However, the image can be low-quality with noisy spots. In these cases, we have found that an extra run of a low-strength depth2img pipeline with quality-improvement prompts (e.g., *high resolution realistic clear photograph, 4k*) yields a better final image. Note that these additional prompts are concatenated to the full prompt (i.e., the sub-prompts and background prompts). We used depth2img specifically, as the Refine step is usually done for visualizations that are intended to be more realistic. In other scenarios, the refine step is optional and can be replaced with any

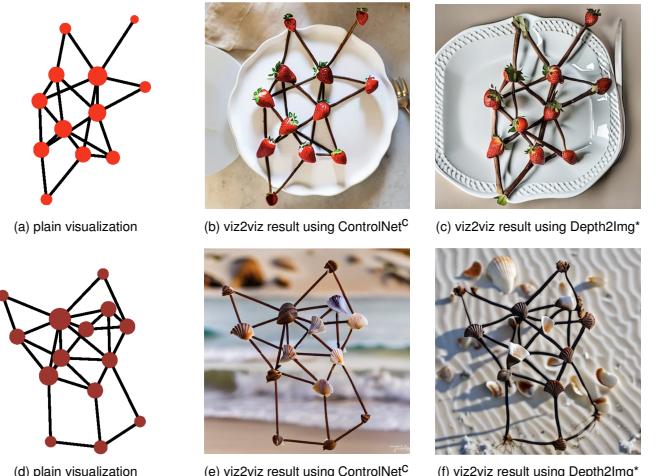


Fig. 10: Result comparison of ControlNet^C and depth2img*. Prompt for (b) and (c): *realistic strawberries and branches on a white plate*. Prompt for (e) and (f) *realistic seashells and branches on the white beach*

resolution-enhancing pipelines (e.g., super-resolution²).

4.2 viz2viz Variations

Depending on the visualization type and prompt, different variants of viz2viz can be used (see Figure 4). We provide some intuition over why a specific workflow recipe may be selected (e.g., structure and shape of the marks, the idiom, and the realism of the desired output). We emphasize the unique properties of each workflow that requires modification to the viz2viz recipe. We emphasize that while each workflow is different, there are sub-elements that re-occur and may be useful for use in new idioms.

4.2.1 Network Graph

The **Sketch step** for network graphs uses the img2img diffusion pipeline on the $N \times N$ grid of the nodes. With multiple sub-prompts, viz2viz performs this step multiple times. For the **Synthesize step** we have found that network graphs require an initial img2img pipeline to smooth the sketched image.

After this step, viz2viz has options depending on whether the prompt calls for a realistic or non-realistic output. For a realistic prompt, the diffusion pipeline can be ControlNet^C (ControlNet with canny edge condition) or depth2img*. Using ControlNet^C can result in more precise mark shapes, but depth2img* can produce a more stylized and visually cohesive result (See the comparison in Figure. 10). For non-realistic styles, the second diffusion pipeline can either be ControlNet^C or img2img*. As with realistic images, ControlNet^C can result in better visual cohesiveness, as well as more smooth edge structure. However, img2img can produce results that are more in line with prompt and have a more ‘sketched’ look. We have found that network graphs do not require an additional **Refine step**.

4.2.2 Bar Chart

Bar charts, which generally use single separated marks, work well with the following recipe. As with network graphs, in the **Sketch step**, an img2img pipeline is applied to a grid of equally-sized bars as a starting point. However, unlike the nodes in the network (which are generally equal-sized marks), bar charts require generated marks to be elongated or duplicated before being re-assembled.

We omit smoothing in the **Synthesize step** as sketch visualizations for bar charts are generally clean. For realistic prompts, we have found that the DMP pipeline works best for bar charts. For non-realistic images, img2img produces better results. As DMP tends to produce

²https://huggingface.co/docs/diffusers/api/pipelines_stable_diffusion/upscale



Fig. 11: Post-Processing by re-generating background and stylized mark(s). Used prompts are *bird eye view of a field of realistic red roses/tulips/purple daisy flowers/orange sunflowers*.

some low-quality artifacts, we used a low-strength depth2img in the **Refine** step to fix these.

4.2.3 Pie Chart

Pie charts require a distinct Sketch step but otherwise can roughly follow the same workflow as the bar chart. This assumes that the pie segments can be cleanly separated. Unlike bar charts and network nodes, pie slices are more complex in shape. Depending on the prompt, viz2viz can use depth2img (on the original visualization) or text2img (from scratch) in the **Sketch step**. In both cases, viz2viz will cut the pie slice from this generated content before re-assembly. The remaining steps, **Synthesize** and **Refine**, follow the same format as the bar chart.

4.2.4 Area Chart

Area charts are like pie charts in that the mark complexity is higher. We used the same **Sketch** and **Refine** steps as pie charts. For the **Synthesize** step we have found that both realistic and non-realistic images can be produced with either the DMP or ControlNet^C pipeline. Because area marks tend to be large, this yields a simple depth structure that can work with ControlNet^C and produce precise mark shapes. DMP can yield more stylized and visually cohesive results. Supporting both pipelines allows users to balance their trade-offs and produces a better stylized visualization.

4.2.5 Optional Post-Processing

In some cases, viz2viz generates a visualization that generally has the stylization we want but may have failed for one sub-prompt (i.e., we like everything except the stylized hamburger bar). To fix these, a user can apply an optional inpainting pipeline to regenerate a specific mark or the background. When applied, viz2viz performs inpainting with the prompt and the mask corresponding to the selected mark/background. Examples of this step are in Figure 11. Users can also choose to regenerate the visualization with a self-defined area and prompts that were not used in the initial generation. Finally, if parts of the visualization are well stylized, the user can copy those to worse areas and execute an inpainting pipeline with low-strength. Clearly, additional edits may be needed or desired to produce a ‘finished’ visualization (e.g., changing fonts, modifying the axes to match the new style, etc.).

4.3 Implementation Details

We implemented viz2viz with Python and Pytorch. We used Google’s Colaboratory notebooks to specify viz2viz recipes. Plain visualizations are generated with Matplotlib, NetworkX and Altair. Image processing, such as mask generation, is achieved with OpenCV and Pillow. We used diffusion models with Diffusers from Hugging Face, using SD models. Specifically, the ControlNet pipeline uses SD v1.5³, and all other pipelines use SD v2⁴. All images are 512×512 in size.

With internal experiments, we found some parameters that work well as defaults. Although viz2viz can work well with default parameters,

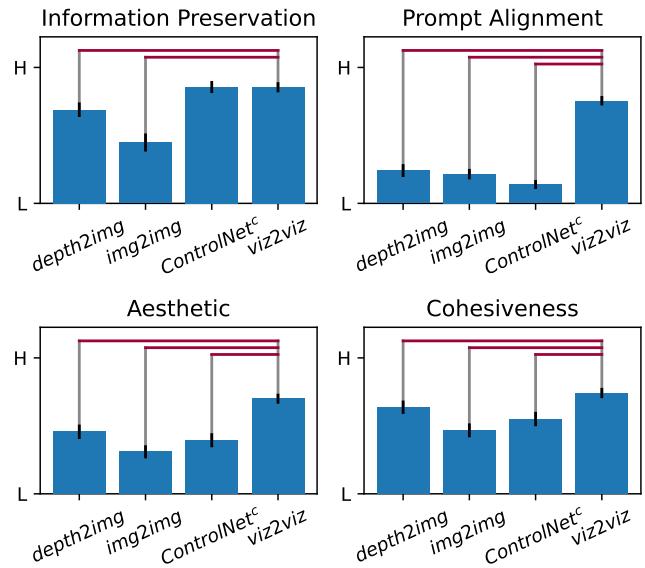


Fig. 12: The evaluation results. The error bars indicate 95% confidence intervals, and the red connecting lines show which conditions are significantly different from viz2viz.

these values can be fine-tuned depending on specific needs. We set guidance scale for all pipelines to 20. In the Sketch step, depth2img strength is set to 0.98 and img2img to 0.82. In the Synthesize step, img2img strength is set to 0.4, while all other pipelines (DMP, depth2img, ControlNet) used 0.8. In the Refine step, a value between 0.25 and 0.35 worked best as a default strength. For DMP pipeline, by default β is set to 0.5.

5 EVALUATION

To evaluate viz2viz, we considered three baselines (depth2img, img2img, and ControlNet^C) on four dimensions: 1) how well is the data preserved (information preservation)?; 2) how well the approach follows the prompt (prompt alignment)?; 3) how visually aesthetic is the generated visualization (aesthetic)?; and 4) how visually cohesive is the output (cohesiveness)? In all, we used three different prompts (both with single and multiple sub-prompts) for each of the four chart types (12 “groups” in total). We allowed each system to generate four possible outputs with the default parameters⁵. One author created the samples and randomized the results (but did not participate in the evaluation). For all conditions, we used the negative prompt of “low quality, normal quality, worst quality, poorly drawn, error, abstract, blurry.” For viz2viz, the workflow that best fits with the visualization type and style specified in the prompt was selected (i.e., realistic or non-realistic). When there are two possible workflows, the higher quality workflow was chosen (which would be what the users of viz2viz would likely do). Note that with viz2viz, we did not perform the optional post-processing. Our prompts came from different sources. Some were descriptions of existing stylized visualizations, and others were suggested by designers from the Data Visualization Society or our research groups. Three evaluators (two male, one female) who were blind to the generation conditions evaluated each of the 192 generated images on the four metrics using a five-level Likert scale. For the analysis of each metric, we performed the Kruskal-Wallis test with all conditions, followed by Dunn’s Test as a post hoc analysis.

Our results are summarized in Figure 12. For ‘preservation of information,’ we saw a significant difference ($H = 115.7$, $p < 0.001$), with viz2viz significantly outperforming depth2img ($p < 0.001$) and

³<https://huggingface.co/llyasviel/ControlNet>

⁴<https://huggingface.co/stabilityai/stable-diffusion-2-1>

⁵For baselines, default strengths are set to 0.8 (img2img) and 0.95 (depth2img, ControlNet^C), as these values showed the best overall performance.

img2img ($p < 0.001$). Naturally, ControlNet follows the outlines of the original visualization well (which is why we use it as a subcomponent of viz2viz). With ‘prompt alignment’ ($H = 270.8$, $p < 0.001$) viz2viz has significantly higher quality than all other conditions ($p < 0.001$ for all). The trend was similar for the aesthetics metric ($H = 117.6$, $p < 0.001$) with viz2viz scoring significantly higher than all other conditions ($p < 0.001$ for all). Finally, for cohesiveness, conditions showed a significant difference ($H = 60.9$, $p < 0.001$), with viz2viz again outperforming all others ($p < 0.05$ for depth2img, and $p < 0.001$ for others). The average Spearman’s ρ value for each metric was 0.20, 0.31, 0.23, and 0.13 for information preservation, prompt alignment, aesthetic, and cohesiveness, respectively. These indicate a slight to fair agreement between the annotators.

As a case study, we took the images we generated using DALL-E 2 (see Figure 3) and tried to generate close approximations using viz2viz. In most situations, we replicated a close approximation with *real* data as input. We describe examples that were less successful below. Prompts and materials from both studies are include in the supplementary materials.

6 DISCUSSION AND FUTURE WORK

6.1 Generalizability of viz2viz

We designed viz2viz to be generalizable to other visualization idioms. The three general recipe steps (Sketch-Synthesize-Refine) are intended to balance the preservation of visualized information while coherently styling the visualization. As our experience shows, we believe that there is an inherent challenge in creating one pipeline that works across all inputs and prompts. That said, we believe that in addition to the high-level workflow’s generalizability, sub-components in our specific implementations can be applied to new visualization types. For example, a scatter plot can be built on the network graph’s workflow and a donut chart can be based on the pie chart.

6.2 Prompt Engineering

Visualizations in viz2viz are highly sensitive to the quality of the prompts. While we leave a systemized evaluation of prompt engineering to future work, we consider some patterns we have observed.

In viz2viz, prompts work better on more specific and common objects than on abstract or imaginary objects. For example, *a side view of a wooden roller coaster* is more likely to get a better result over *side view of cosmic red roller coaster, surreal, psychedelic themes*. Things that the underlying diffusion model is bad at generating are unlikely to work for stylized visualizations. For example, we have found that models struggle with *medical syringe*. Testing the prompt without any data points in a standard pipeline (e.g., text2img, DALL-E, etc.) may be useful in tuning prompts that have a better chance of working. For example, we note that *fries* tends to have better results than *chips* (as this meaning is somewhat more rare). In some cases, a single prompt can have ambiguity in what it will render, and additional text is useful. For example, adding *fruit* or *color* to *orange* will improve the chances of getting the desired stylization. Finally, we find that there are differences in capabilities of different generative algorithms. For example, Stable Diffusion struggled with a branded Coca-Cola bottle.

As with generative prompts, in general, we find that including detailed descriptions of the *camera view, the number of objects, styles, colors, the specific type of objects, shapes, object names, layouts, and background* can help. Explicitly describing the desired style or medium (e.g., *photorealistic* or *oil painting*) appears to help. When working with one-to-many style visualizations, adding *knolling* can be useful. One area in which we have not extensively experimented is with negative prompts. We use these in the final ‘cleanup’ steps of viz2viz, but it is possible that such prompts can help guide the system to produce better results in earlier operations.

6.3 Getting High Quality Images

Our experience with viz2viz (and other generative systems) is that getting a good result often takes multiple attempts. In this paper, we have attempted to capture key decision points that would lead one to choose a particular workflow or pipeline over another. We have also

tried to acknowledge the various scenarios in which human intervention or repetition is necessary. We have found in our experiments that the upper bound on finding a good image (given a good prompt) is around 10 images (though many take far less).

Not all prompts can be easily generated. When color or shape of the plain-visualization mark is so different from the desired object, the generation might fail. Thus, some coherence is required between initial inputs and desired outputs (e.g., a red pie chart slice will work much better for the prompt *a slice of pizza* over a yellow slice).

In addition to prompts, most pipelines have other tunable parameters. We have offered good defaults based on our experiments. However, it is likely that certain stylization requests would benefit from further tuning. For example, a user may want a good value for the β parameter in DMP (0.5 by default). In response to intermediate results (e.g., the quality of the Sketched visualization), the user can increase (if the image is promising) or decrease this value. Finally, we emphasize that viz2viz can fail completely or partially. Using post-generation inpainting and similar features can help recover from partial failures.

6.4 Limitations and Future Work

Currently, viz2viz is implemented in the context of a notebook environment. While this can be easily used as a library, an important area to tackle in the future is the implementation of a GUI. Being able to easily see the effect of changes, seeing intermediate results, varying parameter spaces, and generally having access to direct manipulation operations (e.g., for inpainting) will likely lead to better outputs.

While viz2viz can generate many different types of marks, sometimes better semantic understanding (both of language and objects) can lead to better outcomes. For example, it is difficult to specify that we would like the bars in a chart to be represented by the amount of liquid in a syringe (rather than the syringe’s length). One potential approach is to use sketched or found examples as additional input. Because we do not tightly control the interaction between foreground and background objects, viz2viz is unable to generate certain infomages [6] (the top-hat in Figure 3, for example). Future work, with additional steps to pre-generate the background and specify the area in which the visualization should be rendered, may allow viz2viz to support these stylized visualizations more broadly.

As with other generative systems, some pipelines can produce noisy results. With viz2viz the problem can become more pressing as our various generation constraints can lead to unfortunate side-effects in what the visualization looks like. Future work to address this may involve better parameter tuning, pipeline creation, better masks (for DMP and inpainting) and improved interfaces.

Finally, viz2viz is currently limited to the scale of marks it can support. As the number of marks increases, they may begin to blend improperly during diffusion. There is nothing inherently problematic in generating smaller subsets of the visualization in pieces. Characterizing the limits of this approach and ensuring, technically, that images can be reconstructed in a way that looks good would be areas of future work.

7 CONCLUSIONS

In this paper, we introduce viz2viz, a generation workflow allowing users to get high quality stylized visualizations given a text prompt and plain visualization. We introduce a three-step recipe that ensures that the stylized visualization encodes data correctly while at the same time satisfying complex criteria like image cohesiveness. To achieve this, we make use of state-of-art controllable diffusion models as well as our self-designed pipeline (DMP). We describe how the viz2viz can be generalized to other stylization tasks. We believe that there is both an inherent challenge and opportunity in using generative approaches in visualization. We are excited to offer a possible entry point into this area for both practitioners and researchers.

ACKNOWLEDGMENTS

We would like to thank Jason Forrest, Elsie Lee-Robbins, and Matt Brehmer, as well as early pilot users who gave us prompts and feedback. This work was partially funded by NSF IIS-1815760.

REFERENCES

- [1] O. Avrahami, D. Lischinski, and O. Fried. Blended diffusion for text-driven editing of natural images. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18208–18218, 2022. 3
- [2] A. Bigelow, S. Drucker, D. Fisher, and M. Meyer. Reflections on how designers design with data. In *Proc. of the 2014 Int. Working Conference on Advanced Visual Interfaces*, AVI ’14, p. 17–24. Association for Computing Machinery, New York, NY, USA, 2014. doi: [10.1145/2598153.2598175](https://doi.org/10.1145/2598153.2598175) 2
- [3] A. Bigelow, S. Drucker, D. Fisher, and M. Meyer. Iterating between tools to create and edit visualizations. *IEEE Trans. on Vis. and Comp. Graphics*, 23(1):481–490, 2017. doi: [10.1109/TVCG.2016.2598609](https://doi.org/10.1109/TVCG.2016.2598609) 2
- [4] M. Brehmer, R. Kosara, and C. Hull. Generative design inspiration for glyphs with diatoms. *IEEE Trans. on Vis. and Comp. Graphics*, 28(1):389–399, 2022. doi: [10.1109/TVCG.2021.3114792](https://doi.org/10.1109/TVCG.2021.3114792) 2
- [5] Z. Chen, Y. Wang, Q. Wang, Y. Wang, and H. Qu. Towards automated infographic design: Deep learning-based auto-extraction of extensible timeline. *IEEE Trans. on Vis. and Comp. Graphics*, 26(1):917–926, 2020. doi: [10.1109/TVCG.2019.2934810](https://doi.org/10.1109/TVCG.2019.2934810) 2
- [6] D. Coelho and K. Mueller. Infomages: Embedding data into thematic images. In *Computer Graphics Forum*, vol. 39, pp. 593–606. Wiley Online Library, 2020. 2, 9
- [7] W. Cui, J. Wang, H. Huang, Y. Wang, C.-Y. Lin, H. Zhang, and D. Zhang. A mixed-initiative approach to reusing infographic charts. *IEEE Trans. on Vis. and Comp. Graphics*, 28(1):173–183, 2022. doi: [10.1109/TVCG.2021.3114856](https://doi.org/10.1109/TVCG.2021.3114856) 2
- [8] W. Cui, X. Zhang, Y. Wang, H. Huang, B. Chen, L. Fang, H. Zhang, J.-G. Lou, and D. Zhang. Text-to-viz: Automatic generation of infographics from proportion-related natural language statements. *IEEE Trans. on Vis. and Comp. Graphics*, 26(1):906–916, 2020. doi: [10.1109/TVCG.2019.2934785](https://doi.org/10.1109/TVCG.2019.2934785) 2
- [9] J. Forrest. I asked an artificial intelligence to draw a chart, 2022. Accessed on March 22, 2023. 2
- [10] J. Harper and M. Agrawala. Converting basic d3 charts into reusable style templates. *IEEE Trans. on Vis. and Comp. Graphics*, 24(3):1274–1286, 2018. doi: [10.1109/TVCG.2017.2659744](https://doi.org/10.1109/TVCG.2017.2659744) 2
- [11] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 3
- [12] L. Huang, D. Chen, Y. Liu, Y. Shen, D. Zhao, and J. Zhou. Composer: Creative and controllable image synthesis with composable conditions, 2023. 3
- [13] K. Kagkelidis, I. Dimitriadis, and A. Vakali. Lumina: An adaptive, automated and extensible prototype for exploring, enriching and visualizing data. *J. Vis.*, 24(3):631–655, jun 2021. doi: [10.1007/s12650-020-00718-y](https://doi.org/10.1007/s12650-020-00718-y) 2
- [14] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic, and H. Pfister. Data-driven guides: Supporting expressive design for information graphics. *IEEE Trans. on Vis. and Comp. Graphics*, 23(1):491–500, 2017. doi: [10.1109/TVCG.2016.2598620](https://doi.org/10.1109/TVCG.2016.2598620) 2
- [15] Y. Li, H. Liu, Q. Wu, F. Mu, J. Yang, J. Gao, C. Li, and Y. J. Lee. Gligen: Open-set grounded text-to-image generation, 2023. doi: [10.48550/ARXIV.2301.07093](https://doi.org/10.48550/ARXIV.2301.07093) 3
- [16] Y. Liu, E. Jun, Q. Li, and J. Heer. Latent space cartography: Visual analysis of vector space embeddings. In *Computer graphics forum*, vol. 38, pp. 67–78. Wiley Online Library, 2019. 2
- [17] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko. Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proc. of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI ’18, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2018. doi: [10.1145/3173574.3173697](https://doi.org/10.1145/3173574.3173697) 2
- [18] H. Mei, Y. Ma, Y. Wei, and W. Chen. The design space of construction tools for information visualization: A survey. *Journal of Visual Languages & Computing*, 44:120–132, 2018. doi: [10.1016/j.jvlc.2017.10.001](https://doi.org/10.1016/j.jvlc.2017.10.001) 2
- [19] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *Int. Conference on Learning Representations*, 2022. 1, 3
- [20] C. Qian, S. Sun, W. Cui, J.-G. Lou, H. Zhang, and D. Zhang. Retrieve-then-adapt: Example-based automatic generation for proportion-related infographics. *IEEE Trans. on Vis. and Comp. Graphics*, 27(2):443–452, 2021. doi: [10.1109/TVCG.2020.3030448](https://doi.org/10.1109/TVCG.2020.3030448) 2
- [21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *Int. conference on machine learning*, pp. 8748–8763. PMLR, 2021. 3
- [22] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents, 2022. doi: [10.48550/ARXIV.2204.06125](https://doi.org/10.48550/ARXIV.2204.06125) 1, 3
- [23] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. 3
- [24] D. Ren, B. Lee, and M. Brehmer. Charticulator: Interactive construction of bespoke chart layouts. *IEEE Trans. on Vis. and Comp. Graphics*, 25(1):789–799, 2019. doi: [10.1109/TVCG.2018.2865158](https://doi.org/10.1109/TVCG.2018.2865158) 2
- [25] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2021. 1
- [26] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022. 2, 3
- [27] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. G. Lopes, B. K. Ayan, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 3
- [28] A. Satyanarayan, B. Lee, D. Ren, J. Heer, J. Stasko, J. Thompson, M. Brehmer, and Z. Liu. Critical reflections on visualization authoring systems. *IEEE Trans. on Vis. and Comp. Graphics*, 26(1):461–471, 2020. doi: [10.1109/TVCG.2019.2934281](https://doi.org/10.1109/TVCG.2019.2934281) 1
- [29] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. Revision: Automated classification, analysis and redesign of chart images. In *Proc. of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST ’11, p. 393–402. Association for Computing Machinery, New York, NY, USA, 2011. doi: [10.1145/2047196.2047247](https://doi.org/10.1145/2047196.2047247) 2
- [30] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang. Towards natural language interfaces for data visualization: A survey. *IEEE Trans. on Vis. and Comp. Graphics*, pp. 1–1, 2022. doi: [10.1109/TVCG.2022.3148007](https://doi.org/10.1109/TVCG.2022.3148007) 2
- [31] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Int. Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015. 2
- [32] M. Sturdee, S. Knudsen, and S. Carpendale. Data-painting: Expressive free-form visualisation. *Proc. of Design Research Society*, 2022, 2022. 3
- [33] Y. Wang, H. Zhang, H. Huang, X. Chen, Q. Yin, Z. Hou, D. Zhang, Q. Luo, and H. Qu. Infonice: Easy creation of information graphics. In *Proc. of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI ’18, p. 1–12. Association for Computing Machinery, New York, NY, USA, 2018. doi: [10.1145/3173574.3173909](https://doi.org/10.1145/3173574.3173909) 2
- [34] A. Wu, Y. Wang, X. Shu, D. Moritz, W. Cui, H. Zhang, D. Zhang, and H. Qu. Ai4vis: Survey on artificial intelligence approaches for data visualization. *IEEE Trans. on Vis. and Comp. Graphics*, 2021. 2
- [35] H. Xia, N. Henry Riche, F. Chevalier, B. De Araujo, and D. Wigdor. Dataink: Direct and creative data-oriented drawing. In *Proc. of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI ’18, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2018. doi: [10.1145/3173574.3173797](https://doi.org/10.1145/3173574.3173797) 2
- [36] L. Ying, X. Shu, D. Deng, Y. Yang, T. Tang, L. Yu, and Y. Wu. Metaglyph: Automatic generation of metaphoric glyph-based visualization. *IEEE Trans. on Vis. and Comp. Graphics*, 29(1):331–341, 2022. 2
- [37] J. E. Zhang, N. Sultanum, A. Bezerianos, and F. Chevalier. Dataquilt: Extracting visual elements from images to craft pictorial visualizations. In *Proc. of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2020. 2
- [38] L. Zhang and M. Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023. 1, 3
- [39] S. Zhu, G. Sun, Q. Jiang, M. Zha, and R. Liang. A survey on automatic infographics and visualization recommendations. *Visual Informatics*, 4(3):24–40, 2020. doi: [10.1016/j.visinf.2020.07.002](https://doi.org/10.1016/j.visinf.2020.07.002) 2

When there are many sub-prompts in the prompt, the numbers on the plain visualization indicates which mark is intended to link to which sub-prompt. Sub-prompts appear by sequence number.

For all generated results, we used default parameters (please refer to Section 4.3).



Fig. 13: Supplementary results 1.

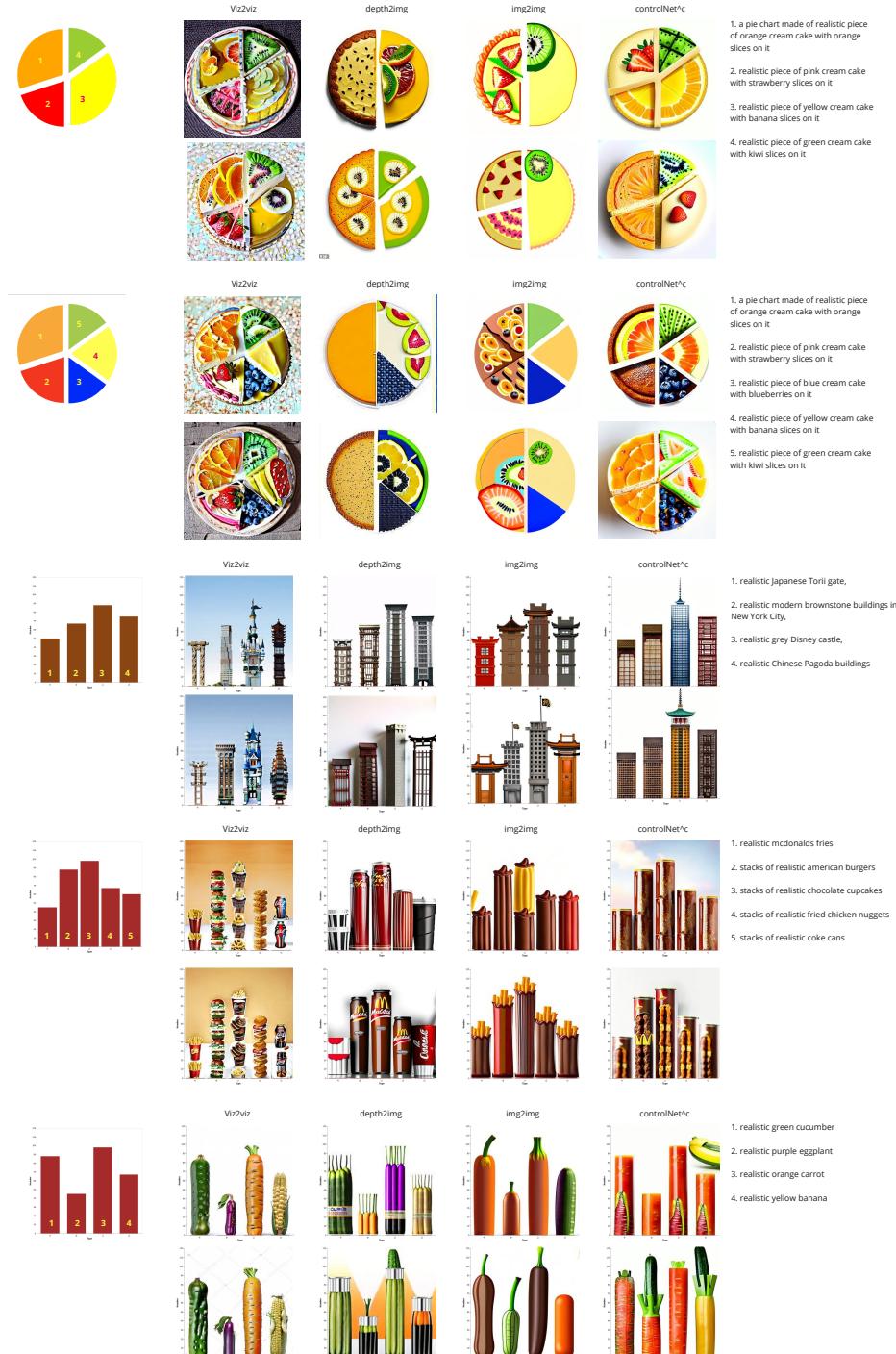


Fig. 14: Supplementary results 2.



Fig. 15: Supplementary results 3.

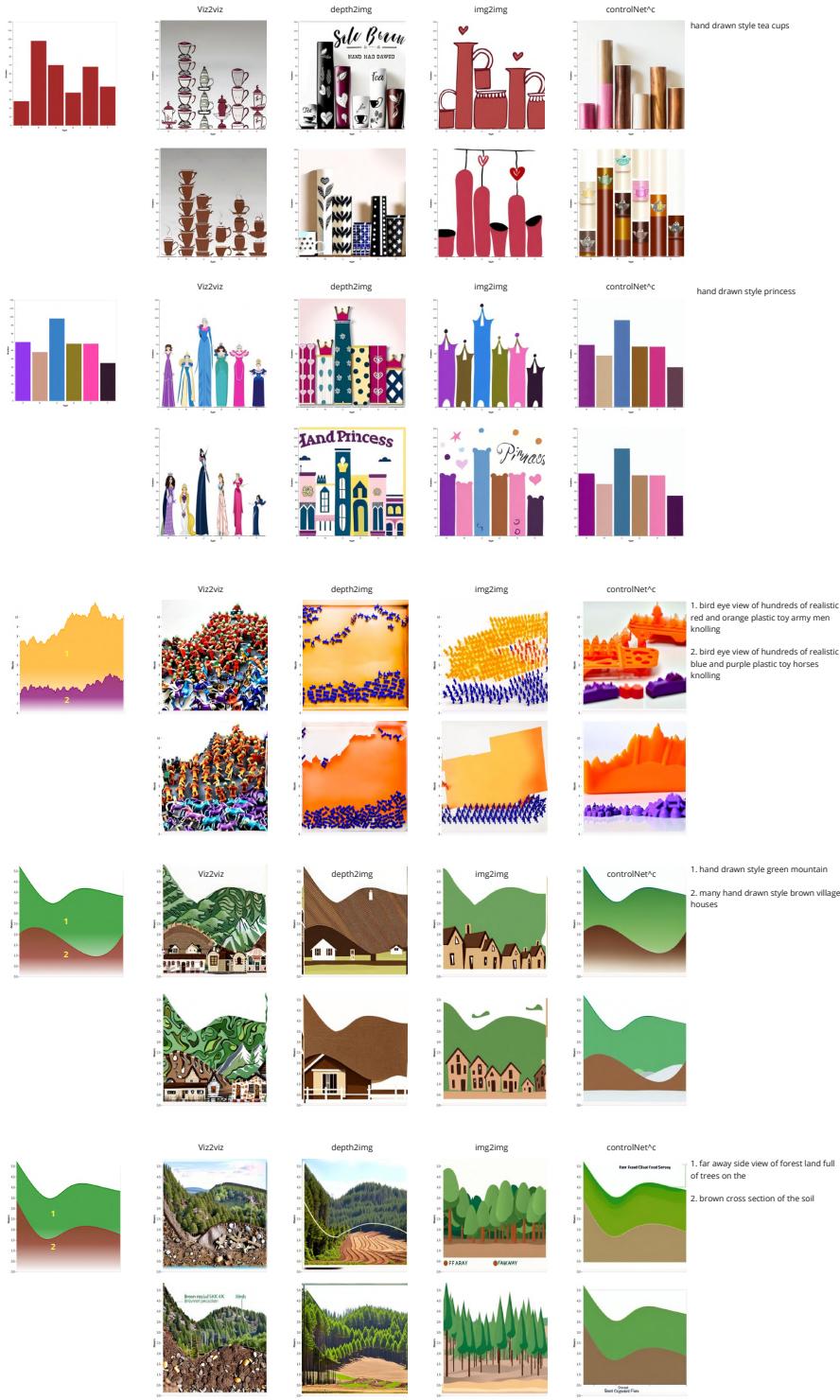


Fig. 16: Supplementary results 4.

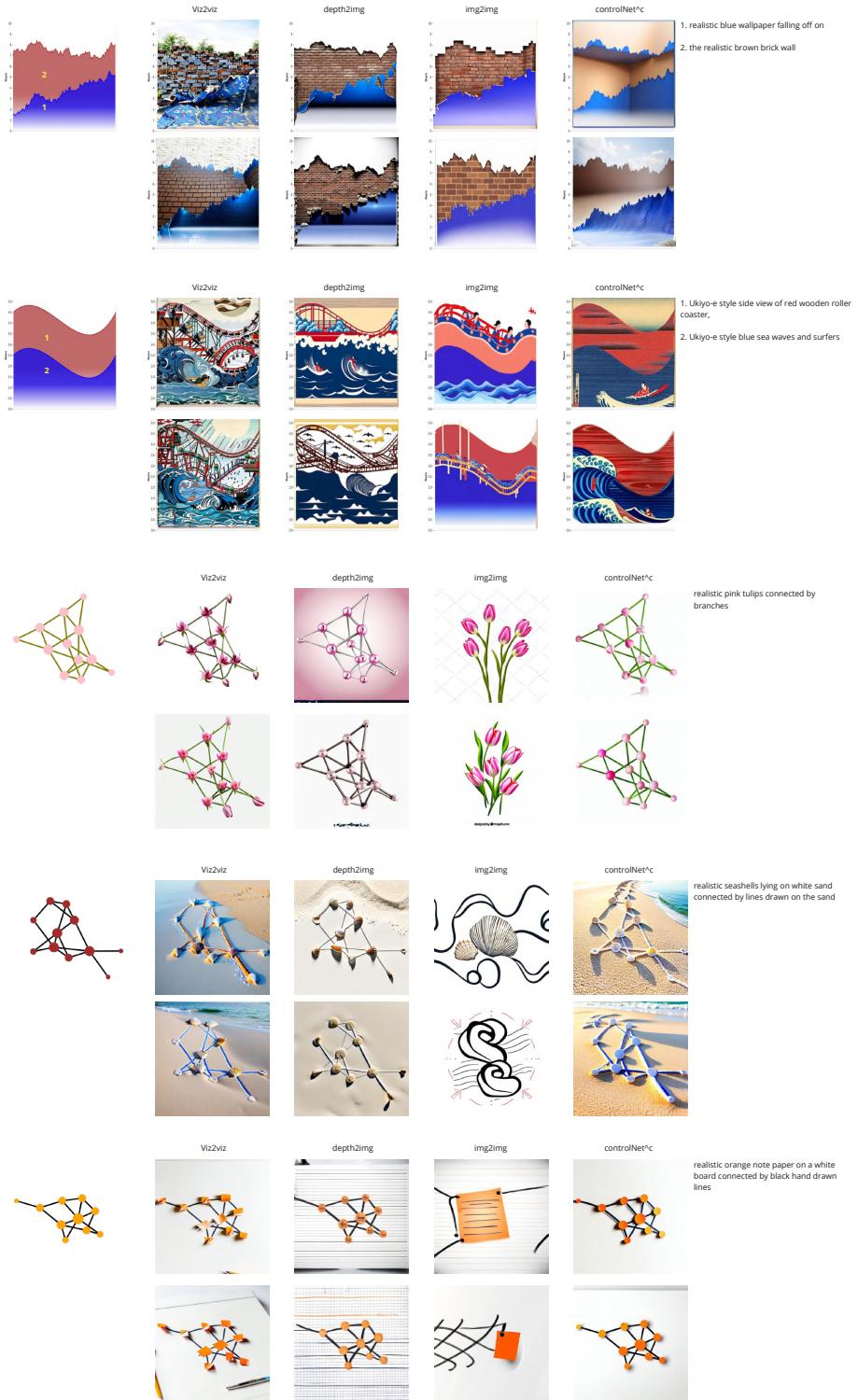


Fig. 17: Supplementary results 5.

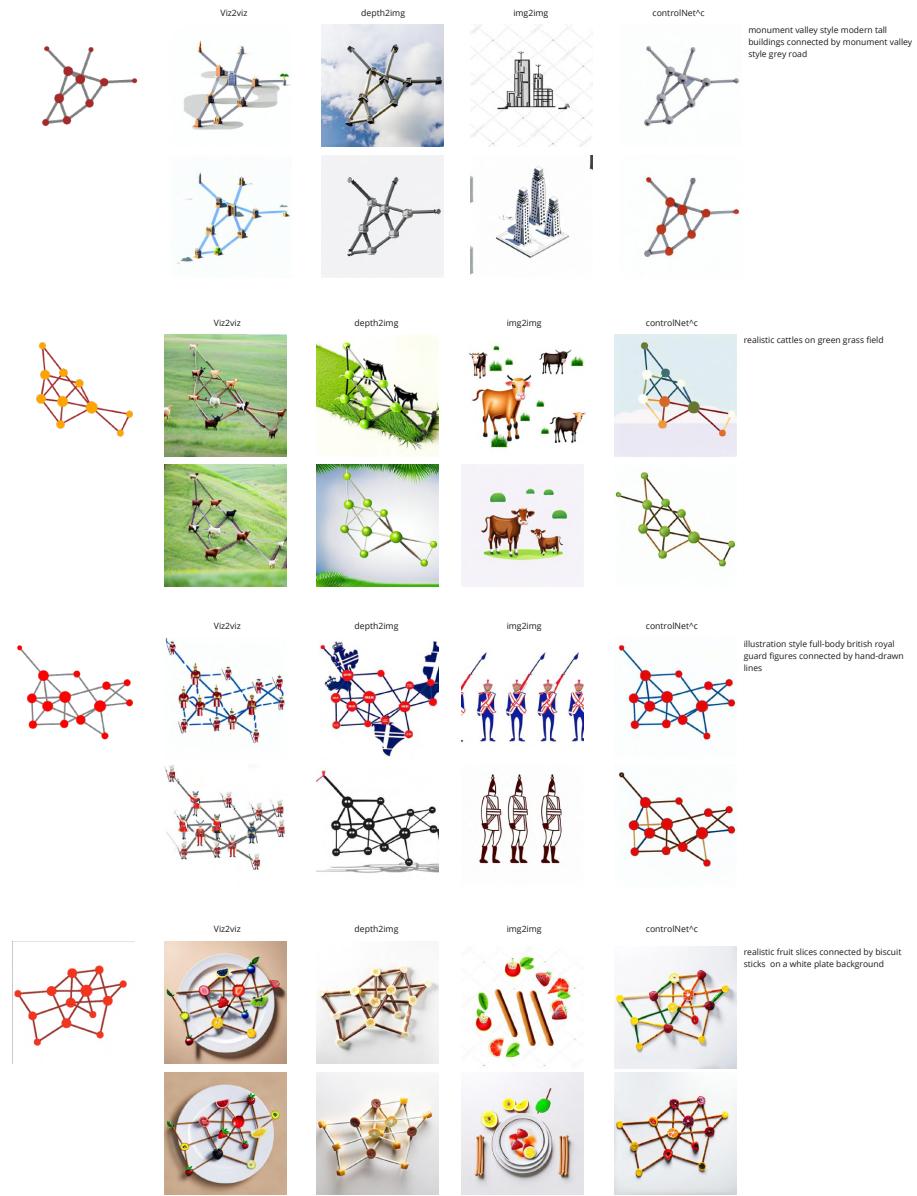


Fig. 18: Supplementary results 6.

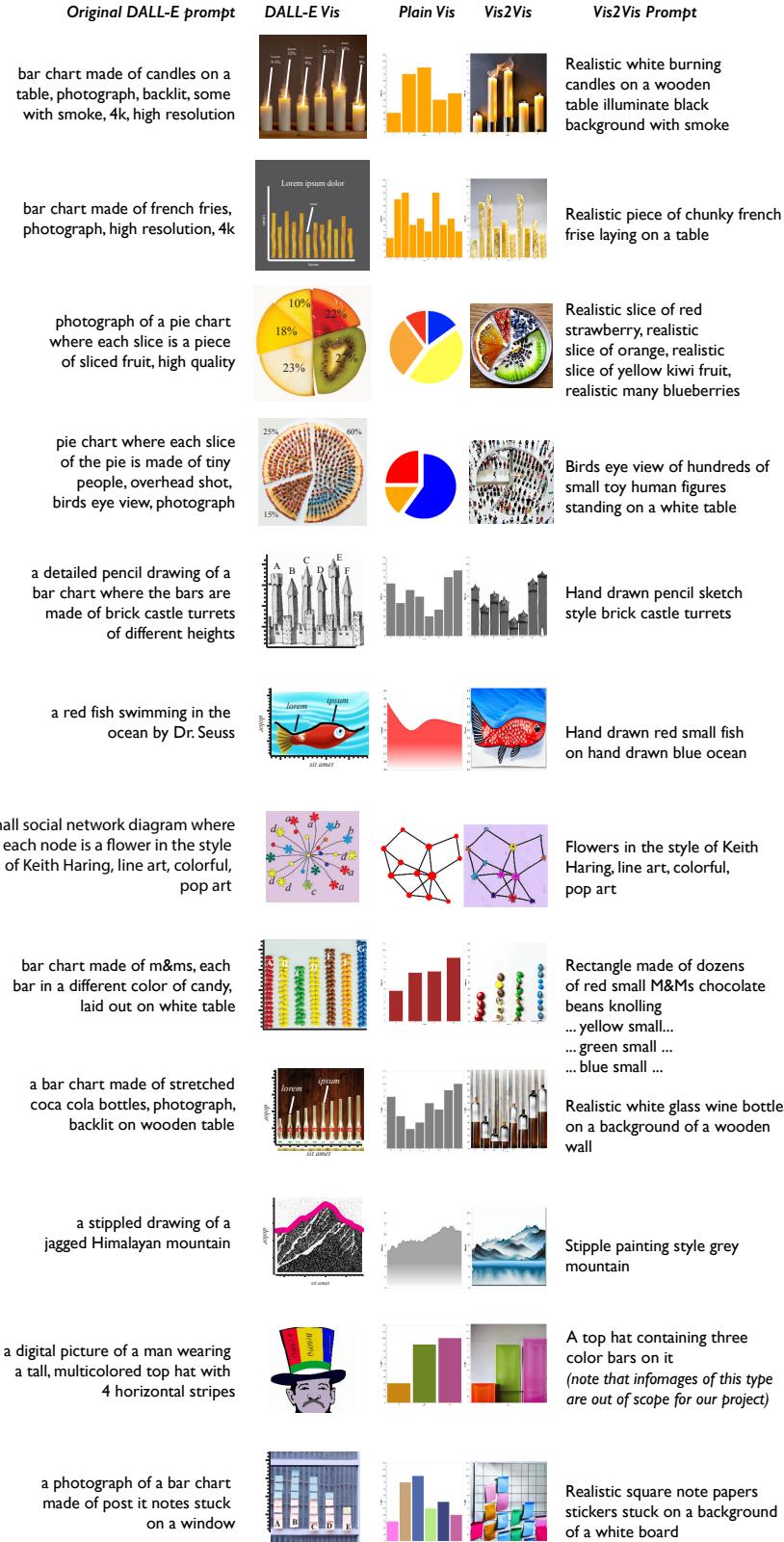


Fig. 19: Comparison between DALL-E 2 and viz2viz. Note that DALL-E 2 cannot specify the underlying data while viz2viz can.