

GenHowTo: Learning to Generate Actions and State Transformations from Instructional Videos

Supplementary Material

Tomáš Souček¹ Dima Damen² Michael Wray² Ivan Laptev³ Josef Sivic¹

¹CIIRC CTU ²University of Bristol ³MBZUAI

tomas.soucek@cvut.cz

<https://soczech.github.io/genhowto/>

Overview

This supplementary material details the GenHowTo model training and dataset acquisition in Section A. Then, in Section B, we provide additional insights into our method and dataset choices. In Section C, we compare GenHowTo to previous methods through a user study. Finally, in Section D, we show a large variety of qualitative results.

A. Additional details

GenHowTo details. We train the GenHowTo model for four days on eight A100 GPUs with a batch size of 32 at the resolution of 512×512 pixels. We build on the official implementation of the ControlNet [6]. The model is initialized with the Stable Diffusion v2.1 base EMA weights and trained using a fixed learning rate of $2 \cdot 10^{-5}$. During inference, we use DDIM sampler with 50 denoising steps. For better consistency of the qualitative results with the input images, we skip the first two steps and instead use the noise-perturbed representation of the input image (see Sec. 3.3).

Evaluation details. We provide the exact details to replicate our classification-based evaluation, including the way to obtain the test images on the project’s GitHub website¹. For the FID evaluation, we use the publicly available PyTorch implementation². For each input image \mathcal{I} and test prompt \mathcal{P} from the test set, we generate three images with different random seeds. We compute the FID between the generated images and the real target images \mathcal{I}^* from the test set. We report the results separately for the action images and the final state images.

Dataset acquisition details. To obtain the initial state, action and final state images, we train the self-supervised model using the official implementation [5]. We train separate models for the ChangeIt and for the COIN datasets. For each dataset, we train the model three times, starting with different random seeds. From each training, we select

¹<https://github.com/soCzech/GenHowTo>

²<https://github.com/mseitzer/pytorch-fid>

Action prompt \mathcal{P}_{ac} : *a person cutting a fish on a cutting board*
Final state prompt \mathcal{P}_{st} : *two pieces of fish on a wooden cutting board*



Action prompt \mathcal{P}_{ac} : *a young man tying a tie in a room*
Final state prompt \mathcal{P}_{st} : *a young man in a gray shirt and red tie*



Action prompt \mathcal{P}_{ac} : *a person slicing an apple on a cutting board*
Final state prompt \mathcal{P}_{st} : *sliced apples on a cutting board next to a fork*



Action prompt \mathcal{P}_{ac} : *a woman in a purple jacket holding a cell phone*
Final state prompt \mathcal{P}_{st} : *a Rubik’s cube on top of a table*



Figure 1. **Example of training dataset 5-tuples.** All shown images are automatically detected from instructional videos by [5] and then automatically labeled using BLIP2. As the examples show, the images are nicely aligned across both the action and the final state, allowing our model to learn to manipulate images from instructional videos. However, as both the images and prompts are obtained automatically, they can contain errors (in red).

weights from three best-performing epochs—yielding nine model instances for each of the datasets. Each video from the two datasets is processed by the respective models, totaling nine correlated, yet often distinct, image sets, each

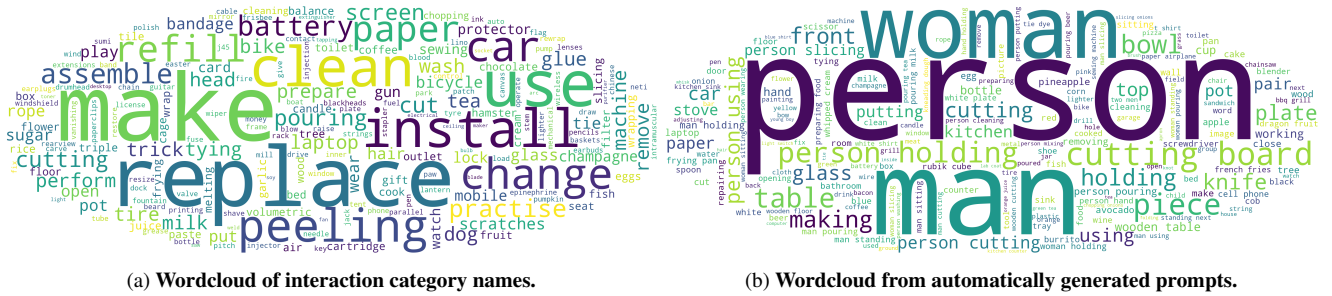


Figure 2. The distribution of interaction categories from the ChangeIt and the COIN datasets used to train our model (left). And the distribution of words from the automatically generated prompts for our target images extracted from the ChangeIt and the COIN datasets (right).

containing the initial state, action, and final state per video. We keep approximately 90% of COIN triplets with the highest classification scores as determined by the models to filter out incorrect predictions. For the ChangeIt dataset, we keep only 30% as those videos are uncurated, and a large portion of the dataset consists of irrelevant videos.

For the text prompts, we use the zero-shot image-to-text version of BLIP2 weights pretrained on flant5xxl. When generating the image captions, we leave the text prompt empty—we observed a reduction in the quality of the output caption when using text prompts.

Sample 5-tuples of three images and two prompts can be seen in Figure 1. As the figure shows, the images are often well spatially aligned with only the object changing. This allows our model to learn to transform objects in the images according to prompts while keeping the background the same. In some cases, the automatically generated results can have errors (Figure 1, last row). In this case, the automatically generated caption for the action is incorrect.

Illustrating the variety of actions and objects. Our images come from instructional videos of the COIN and the ChangeIt datasets. Those datasets contain over two hundred interaction categories. The names of the interaction categories are shown as a word cloud in Figure 2a. Similarly, we also show the distribution of words from the automatic image captions in Figure 2b. Together, the dataset images represent people manipulating objects in various environments. A large focus of the dataset is given to the most common environment in which people seek advice in—the kitchen.

B. Additional ablations

Varying levels of noise. In Figure 3, we show more qualitative examples of our DDIM sampling strategy where we start with a noise-perturbed latent representation of the input image. In the example of the pan cleaning (top), our method correctly generates the transformed object (*i.e.*, cleaned pan), yet in the case of random noise initialization (right), the method generates the object incorrectly scaled. In the second example of the peeled garlic, the image (right)



Figure 3. Varying levels of noise used to initialize image generation process. The rightmost image shows vanilla DDIM sampling initialized from Gaussian noise. Other images show when the sampling process is initialized from a noise-perturbed latent representation of the input image (left) with varying amounts of noise. Less noise generates images closer to the input scene. An initialization with the right amount of noise (black frame) better preserves the scene (e.g. viewpoint – top or color of the wooden board – bottom) while changing the state of the object.



	<p>Narration transcript</p> <p>... in another video I think I've got to explain to you all in more detail this whole business of screws and drilling so ...</p> <p>BLIP2 caption</p> <p>person using drill to build a wooden frame</p>
	<p>Narration transcript</p> <p>... it's really hard to try and slice it even thickness slices I lasers were kind of like all over the place so it made it ...</p> <p>BLIP2 caption</p> <p>a man slicing a pineapple on a cutting board</p>

Figure 4. Comparison of narration transcript vs. automatic captions. We show approximately 8 seconds of ASR transcript centered around the shown video frame and the BLIP2 caption for the same video frame. We see the transcript often mentions information not shown in the video. The BLIP2 caption, on the other hand, describes what is happening in the frame.



Figure 5. **Examples of GenHowTo action predictions.** We show multiple predictions per input, each with a different random seed. For comparison, we also show predictions of the related methods EF-DDPM and InstructPix2Pix in the last two columns. Our model correctly generates the person’s hands interacting with the object in the scene. Also, our method can correctly preserve the background, which is not always true for the related methods.

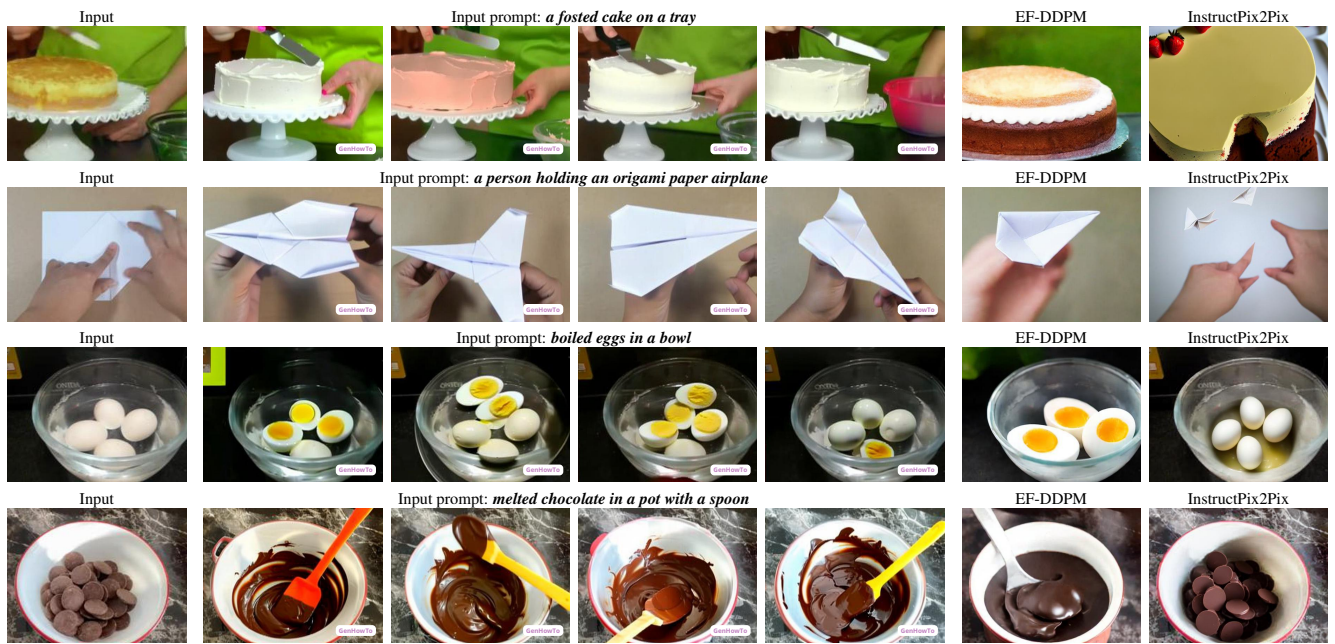


Figure 6. **Examples of GenHowTo final state predictions.** We show multiple predictions per input, each with a different random seed. For comparison, we also show predictions of the related methods EF-DDPM and InstructPix2Pix in the last two columns.

generated only from random noise contains a darker background. Here, even though our model sees the input image (via the ControlNet branch of the model), it may not be enough for perfect background reconstruction, as slightly misaligned training image pairs and other common video artifacts can force the model to focus more on the semantics and less on the pixel-level reconstruction. Starting from the noise-perturbed latent representation of the input image (Figure 3, black frame) corrects for these errors. Some misaligned training images are shown in Figure 1 (the Rubik’s cube example).

Automatically mined video frames. We investigate the significance of the unsupervised model [5] used in dataset acquisition. We test this by substituting our mined video frames with uniformly sampled frames. On our classification benchmark, the accuracy drops from 0.74 (Table 2 in the main paper, (e)) to 0.67. We also observe poor visual quality and consistency of the generated images. This showcases the importance of our unsupervised mining approach for successfully training GenHowTo.

Comparison of narration transcript vs. captions. In our method, as described in Section 3.2 in the main paper, we refrain from using narration transcripts. Instead, we utilize an image captioning model to provide captions for our method. In Figure 4, we show a comparison between the narration transcript and the BLIP2 captions used in our work. We can observe the phenomenon already reported by [2, 4] that the automatic narration is often noisy and may not align well with the content shown in the video. Further, we also verify the claim quantitatively. We replace our BLIP-2 generated captions with the automatically obtained video narration (ASR) closest to the target frame. The accuracy on our classification benchmark drops from 0.74 (Table 2 in the main paper, (e)) to 0.60.

C. User study

We run a user study by recruiting 10 people to assess generated images from GenHowTo, InstructPix2Pix [1], or EF-DDPM [3]. In each case, we use the same initial image, and generated images from one of GenHowTo and one of the other two baselines (InstructPix2Pix or EF-DDPM). Each rater was then shown the real initial state image, a question, and two generated images, in a random order. Raters had a forced choice of which method better addresses the questions. **Q1:** “Which image better represents the final state described as `<input_prompt>` of the same object as in the first image?”. This question verifies the generation of the correct final state for the foreground object. **Q2:** “Which image better preserves the consistency of the scene?” to verify how well the methods preserve the background.

Figure 7 shows how often each method was preferred,

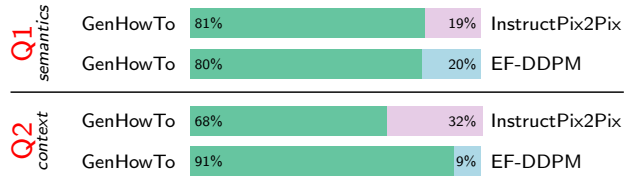


Figure 7. **User preference between various methods.** Users prefer GenHowTo’s outputs for being more true to the input prompt semantically as well as for keeping the background consistent with the original input image.

averaged over all 10 raters. GenHowTo consistently outperforms the other two methods for both questions. This showcases GenHowTo can better represent the semantics of the final state and better reserve the context.

D. Additional qualitative results

Additional qualitative results. We show additional qualitative results in Figures 5, 6, 8, 9, 10, and 11. We show our method can correctly transform objects according to the input prompts. In Figures 5 and 6, we also show that our method generates a large variety of possible output images consistent with the input using different random seeds.

Generalization ability. Before training, our method is initialized by pretrained Stable Diffusion weights. This gives our method some generalization ability, as shown in Figures 9, 10, and 11. To the best of our knowledge, neither ChangeIt nor COIN datasets contain children’s toys, parrots, or landscape scenes, yet our method can still produce meaningful results for such scenes and/or prompts.

Additional qualitative comparison with related work. We show additional comparison with related work on input images from instructional videos in Figures 5, 6, and input images from the internet in Figure 8. We can see our method not only works well on in-distribution data (Figures 5, 6) but also outperforms the related methods on out-of-distribution objects such as strawberries or bananas—the objects not represented in the ChangeIt and COIN datasets. Our method can also produce more localized edits than the closely related InstructPix2Pix method (Figure 11). The images generated by InstructPix2Pix focus more on the style of the output photo rather than only manipulating the target object(s).

Limitations and failure modes. We show the limitations of our method, as described in the main paper, in Figure 12. Our method can struggle, e.g., with people’s faces as they always move in the videos; therefore, they are not spatially aligned in our sets of input images containing the initial state, action, and final state. Also, our model can sometimes ignore fine-grained textures. We hypothesize that the model focuses more on image semantics because some of the training images are not perfectly aligned—forcing the



Figure 8. **Additional qualitative comparison with related work.** We compare our method with Edit Friendly DDPM [3] and InstructPix2Pix [1] on out-of-distribution input images from the internet (first column) and edit prompts (below each row of images). Our dataset does not contain strawberries, blueberries, pears, or bananas, yet our method can correctly transform the objects according to the prompts. In contrast, the related methods often fail to transform the object. EF-DDPM also struggles to preserve the background of the input image.

model to ignore pixel-level details in the input image(s). Finally, as the method is trained on images extracted from a limited set of videos, applying the method to novel objects or to unusual views may result in a degraded performance.

References

- [1] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023. 4, 5
- [2] Tengda Han, Weidi Xie, and Andrew Zisserman. Temporal alignment networks for long-term video. In *CVPR*, 2022. 4
- [3] Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly ddp noise space: Inversion and manipulations. *arXiv preprint arXiv:2304.06140*, 2023. 4, 5
- [4] Antoine Miech, Dimitri Zhukov, J-B Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*, October 2019. 4
- [5] Tomáš Souček, Jean-Baptiste Alayrac, Antoine Miech, Ivan Laptev, and Josef Sivic. Multi-task learning of object state changes from uncurated videos. *arXiv preprint arXiv:2211.13500*, 2022. 1, 4
- [6] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 1

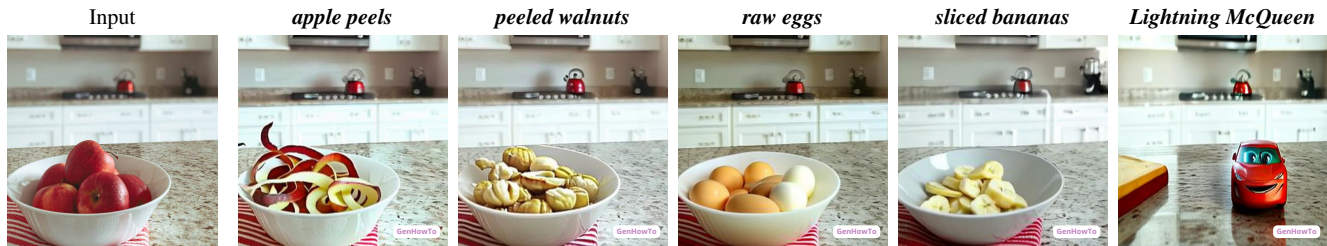


Figure 9. **A bowl of apples transformed in various ways.** Note that objects such as apples or eggs are part of the training data; however, bananas or child toys are missing. The model’s ability to generate such objects comes from the initialization of our model with the StableDiffusion weights. To generate these images, we used the full prompt “[A bowl full of] ... on a kitchen countertop.”

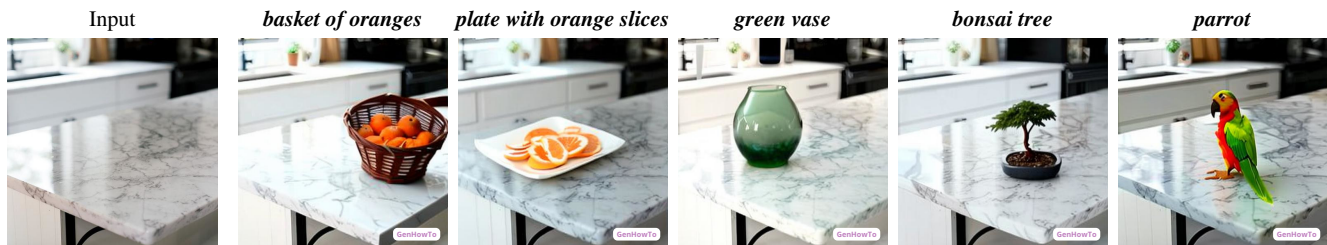


Figure 10. **Various objects added onto a marble countertop.** The model can generalize to objects not seen during our training using frames from instructional videos (e.g., vases, trees, parrots, etc.). The images were generated with the prompt “... on a marble countertop.”

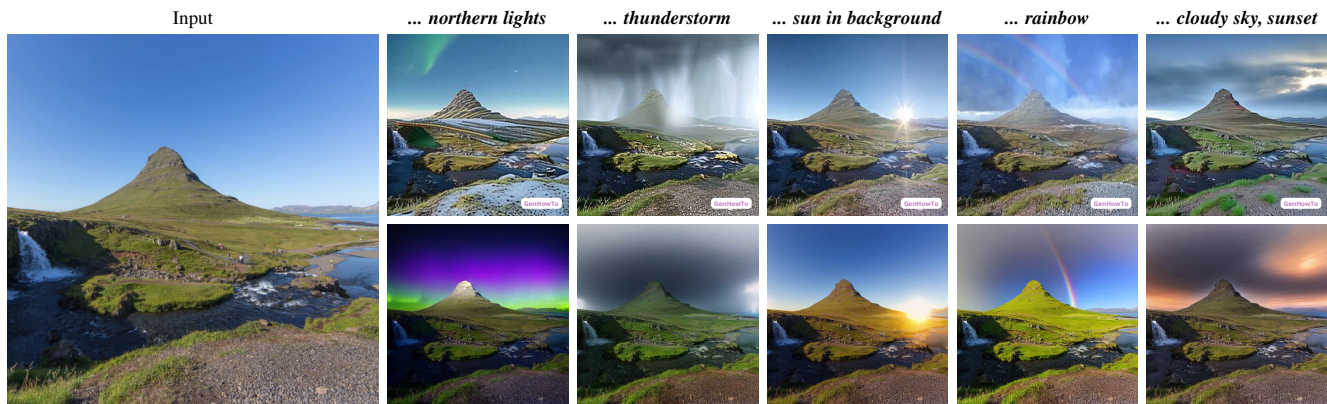


Figure 11. **Comparison of our method (top) and InstructPix2Pix (bottom) on a photo of landscape.** While InstructPix2Pix focuses more on changing the style of the photo, our method only applies more localized edits – this is the result of the different training datasets. Note that our method has not been fine-tuned with any landscape photos, yet it is able to generalize, possibly due to the initialization from the StableDiffusion weights.



Figure 12. **Failure cases.** The performance degrades for always moving objects, such as faces (here not preserving the identity, left). The model can also struggle to preserve fine textures like grass (middle). Lastly, visual quality can suffer for objects not in the training data, especially in extreme viewpoints (right). Each example shows the input image (left), the input prompt (top), and the model’s output (right).