Emu Edit: Precise Image Editing via Recognition and Generation Tasks

Shelly Sheynin*, Adam Polyak*, Uriel Singer*, Yuval Kirstain*, Amit Zohar*, Oron Ashual,
Devi Parikh and Yaniv Taigman

GenAI, Meta

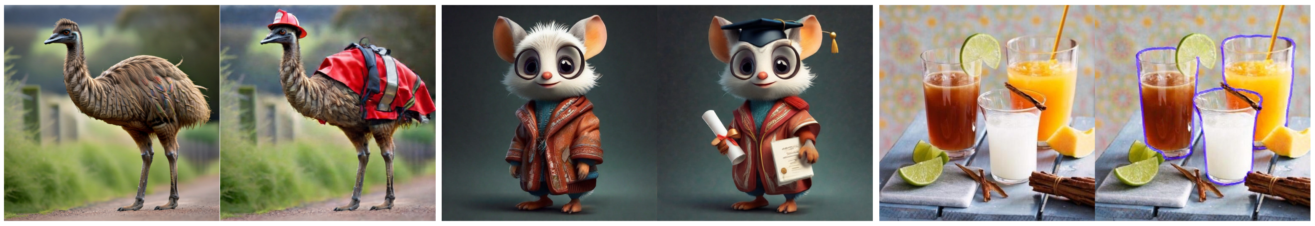

Dress the emu with a fireman outfit
Let's see it graduating
Mark the drinks

Figure 1. Emu Edit is a multi-tasking model that combines various editing (left, middle) and vision (right) tasks for precise image editing.

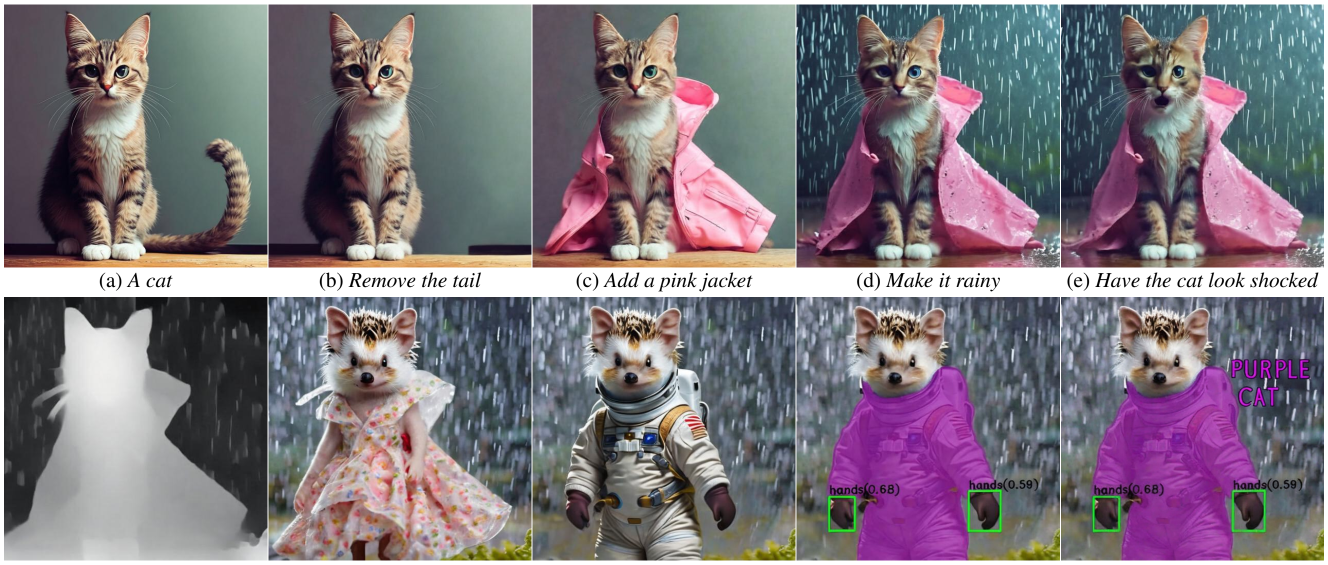
Abstract

Instruction-based image editing holds immense potential for a variety of applications, as it enables users to perform any editing operation using a natural language instruction. However, current models in this domain often struggle with accurately executing user instructions. We present Emu Edit, a multi-task image editing model which sets state-of-the-art results in instruction-based image editing. To develop Emu Edit we train it to multi-task across an unprecedented range of tasks, such as region-based editing, free-form editing, and Computer Vision tasks, all of which are formulated as generative tasks. Additionally, to enhance Emu Edit's multi-task learning abilities, we provide it with learned task embeddings which guide the generation process towards the correct edit type. Both these elements are essential for Emu Edit's outstanding performance. Furthermore, we show that Emu Edit can generalize to new tasks, such as image inpainting, super-resolution, and compositions of editing tasks, with just a few labeled examples. This capability offers a significant advantage in scenarios where high-quality samples are scarce. Lastly, to facilitate a more rigorous and informed assessment of instructable image editing models, we release a new challenging and versatile benchmark that includes seven different image editing tasks. [1]

[1]Project Page: https://emu-edit.metademolab.com/
*Equal contribution.

1. Introduction

Image editing is a widely-used application that millions engage with every day. Popular image editing tools, however, either demand considerable expertise and are time-consuming to use, or are quite limited, providing only a predefined set of editing operations, such as specific filters. Instruction-based image editing [2, 25] attempts to resolve these limitations by allowing users to effortlessly describe their editing goals using natural language instructions. For instance, a user can provide a model with an image and instruct it to "Dress the emu with a fireman outfit" or "Let's see it graduating" (see Fig. 1). Nevertheless, while instruction-based image editing models like InstructPix2Pix [2] are *designed* to process any given instruction, they often struggle to *accurately* interpret and execute such instructions. Moreover, their generalization is limited, often falling short on tasks that deviate slightly from those they were trained on (see Fig. 3). To address these gaps, we introduce Emu Edit, the first image editing model trained on an extensive and diverse set of tasks, including both image editing and computer vision tasks. Emu Edit provides a substantial improvement in both compliance with the edit instruction and preservation of the visual fidelity of the original image. As we show through both automatic metrics [14] and human judgments on two benchmarks [25], Emu Edit achieves state-of-the-art results in instruction-based image editing. The success of Emu Edit stems from two key contributions. First, we train our model to multi-task across sixteen distinct image editing tasks. These tasks span region-based editing tasks, free-form editing tasks and computer



(a) *A cat* (b) *Remove the tail* (c) *Add a pink jacket* (d) *Make it rainy* (e) *Have the cat look shocked*

(f) *Extract the depth map* (g) *Generate a rainy day image of a hedgehog in a dress using the depth map* (h) *Replace the dress with an astronaut outfit* (i) *Segment the spacesuit, and detect the hands* (j) *Add the text "purple cat" using a purple font*

Figure 2. **Instruction-based image generation.** Each image originates from a previous one with a corresponding instruction. The prior image for (a) was set to zero.

vision tasks, all formulated as generative tasks. Unlike previous work, we develop a distinct data generation pipeline for each task, allowing us to gather a training set that is not only more diverse but also more precise in its examples. We find that training a single model on all tasks yields better results than training expert models on each task independently. Additionally, we show that as the number of training tasks increases, so does the performance of Emu Edit. Furthermore, we discover that surprisingly, computer vision tasks such as detection, segmentation, and others, significantly enhance editing performance, as validated both by human raters as well as quantitative measures. Second, to process this wide array of tasks effectively, we introduce the concept of *learned task embeddings*, which are used to steer the generation process toward the correct generative task. Concretely, for each task, we learn a unique task embedding vector, and integrate it into the model through cross-attention interactions, and by adding it to the timestep embeddings. We demonstrate that learned task embeddings significantly enhance the ability of our model to accurately infer the appropriate edit type from the user instruction and execute the correct edit. Equipped with a robust model trained across a broad spectrum of tasks and guided by learned task embeddings, we explore few-shot adaptation to unseen tasks via *task inversion*. In this process, we maintain the model weights untouched, and solely update a task embedding to fit the new task. Our experiments demonstrate that Emu Edit can swiftly adapt to new tasks, such as super-resolution, contour detection, and others. Notably, for some tasks, fine-tuning the model on just a *handful* of examples yields results that nearly match those of an expert model trained on *one hundred thousand* examples. This makes task inversion with Emu Edit particularly advantageous in scenarios where labeled examples are limited, or when the compute budget is low. Finally, to support future research for instruction-based image editing, we publicly release a diverse and challenging benchmark that includes seven different image editing operations, as well as our model's generations on this dataset. In summary, this work addresses the limitations of instruction-based image editing models in accurately following user instructions. We demonstrate that by employing multi-task training across a diverse array of tasks, including recognition, generation, and editing, we can enhance our model's performance. Furthermore, by incorporating learned task embeddings into our model's architecture, we not only improve its results but also enable efficient few-shot learning for new tasks. With these improvements, our model sets a new standard in the field, offering significantly more precise and robust instruction-based image editing capabilities than existing models.

2. Related Work

The emergence of high-performing text-to-image diffusion models [7, 16, 17, 19] facilitated the development of effective text-based image editing methods. Such methods usually employ aligned and detailed descriptions of the input and edited image to perform a specific edit. Prompt-to-Prompt (P2P) [8] injects the input caption attention maps to the target caption attentions maps. Null-Text Inversion [12] inverts an input image using the null-text embedding to support editing of a real image. Plug-and-Play (PNP) [21] injects spatial features in addition to attention maps and obtains better performance at global editing. Imagic [9] finetunes the diffusion model to support complex textual instructions. EDICT [22] suggests an image inversion based on two noise vectors enabling better image reconstruction and textual faithfulness. Another class of image editing models, employs an input mask as additional input [1, 23, 24]. Blended Diffusion [1] modifies the diffusion step by blending the input image in the unmasked regions. Imagen Editor [23] and SmartBrush [24] finetune the text-to-image model to be conditioned on both the input image and mask. While the text-based image editing methods detailed above enable humans to edit images, they frequently exhibit inconsistent performance and require multiple inputs, such as aligned and detailed descriptions of both the input and target images, or at times, input masks.

To offer a more intuitive and user-friendly interface, and significantly enhance ease of use for humans, InstructPix2Pix [2] introduced an instructable image editing model. They developed this model by utilizing both GPT-3 [3] and Prompt-to-Prompt [8], to generate a large synthetic dataset for instruction-based image editing, and employed the dataset to train an instructable image editing model. Unlike InstructPix2Pix which used a synthetic dataset, MagicBrush [25] developed a manually-annotated instruction-guided image editing dataset by requesting humans to use an online image editing tool [13]. Finetuning InstructPix2Pix on this dataset led to improved image editing capabilities. However, even though there has been progress and improvement in instruction-based image editing models, we show in Sec. 5 that state-of-the-art image editing models still struggle with accurately interpreting and precisely executing editing instructions.

In this paper, we drastically narrow such performance gaps by leveraging multi-task training and a matching architecture. Unlike prior work that solely focuses on image editing [2, 25], we train our model to perform various tasks and learn a very diverse set of capabilities. The quality and versatility of our training procedure and dataset, together with our improved architecture for multi-task learning, enables us to make a big leap in performance and differentiates us from prior work in the field. Fig. 3 include several challenging editing samples as examples.

3. Multi-Task Dataset for Image Editing

Training a robust and accurate image editing model requires a highly diverse dataset of input images, editing instructions, and output edited images. However, manually collecting such examples is impractically time-consuming, existing sources on the web (e.g. communities and forums on social media) are limited in size, and publicly available synthetic datasets often lack in diversity or quality. Therefore, we construct a new dataset that encompasses sixteen distinct tasks and ten million examples. Each example (c_I, c_T, x, i) in our dataset, contains an input image c_I, a text instruction c_T, a target image x, and a task index i (out of the sixteen). The following section outlines the process of the data construction. In Sec. 3.1 we describe the instruction generation process, and in Sec. 3.2 the image pairs (c_I, x) generation and filtering.

1. Region-Based Editing
- **Local**: Substituting one object for another, altering an object's attributes (e.g., "make it smile").
- **Remove**: Erasing an object from the image.
- **Add**: Inserting a new object into the image.
- **Texture**: Altering an object's visual characteristics without affecting its structure (e.g., painting over, filling or covering an object).
- **Background**: Changing the scene's background.

2. Free-Form Editing
- **Global**: Edit instructions that affect the entire image, or that can not be described using a mask (e.g., "let's see it in the summer").
- **Style**: Change the style of an image.
- **Text Editing**: This involves text-related editing tasks such as adding, removing, swapping text, and altering the text's font and color.

3. Vision tasks
- **Detect**: Identifying and marking a specific object within the image with a rectangle bounding box.
- **Segment**: Isolating and marking an object in the image.
- **Color**: Color adjustments like sharpening and blurring.
- **Image-to-Image Translation**: Tasks that involve bi-directional image type conversion, such as sketch-to-image, depth map-to-image, normal map-to-image, pose-to-image, segmentation map-to-image, and so on.

Table 1. Description of the tasks forming the Emu Edit dataset.

Task Categories. The dataset is composed of tasks which are divided into three main categories: region-based editing, free-form editing, and vision tasks. Tab. 1 includes the full list of tasks, and their distribution in the train set is visualized in Fig. **??**.

3.1. Instruction Generation

To generate editing instructions, we leverage the dialogue-optimized 70 billion parameter Llama 2 variant [20]. We observed that using a single agent to generate the instructions for all tasks leads to a lack of diversity in the dataset. Notably, the LLM exhibits a bias towards particular tasks and instruction phrasings. To address this, we utilize in-context learning to create a task-specific agent for each task. Concretely, we provide the LLM with a task description, a few task-specific exemplars, and a real image caption. To increase diversity we sample the exemplars and randomize their order. Given such input, we expect the LLM to output (1) an editing instruction, (2) an output caption for an ideal output image, and (3) which objects should be updated or added to the original image. We refer the readers to supplementary Figs. **??**-**??** for examples of our prompts. Further details on instruction generation are provided in the supplementary Sec. **??**.

3.2. Image Pairs Generation

Our aim is to generate pairs of input and edited images that adhere to the edit instructions and preserve image elements that should remain intact. A crucial prerequisite when creating a pair of input and edited images is to guarantee that the two images differ only in specific elements or locations, while remaining identical in all other aspects. Previous instruct-based image editing methods [2] rely on Prompt-to-Prompt (P2P) to build an image-editing dataset. P2P injects cross-attention maps from the input image generation to the edited image generation. To support local edits, P2P additionally approximates a *mask* of the edited part, based on the cross-attention maps and constrains the edit to this local area. P2P relies on word-to-word alignment between the input image caption and the edited image caption (e.g. "a cat riding a *bicycle*" and "a cat riding a *car*") to produce editing image pairs. However, when there is no word-to-word alignment, the resulting mask tends to be imprecise due to its reliance on cross-attention maps. Furthermore, as word-to-word alignment is not a practical assumption in most of the image editing tasks, this approach often fails to preserve structure and identity. To address this challenge, we propose a mask extraction method, which is applied before the editing process. Our approach involves: (i) identifying the edited areas from the editing instruction via an LLM and creating corresponding masks before image generation, and (ii) integrating these masks during the editing process to ensure seamless fusion of edited regions with the original image. Distinct editing challenges, such as adding or removing objects, require tailored solutions. We utilize various techniques, including dilation and Gaussian blurring, to refine the masks. Further description of the method can be found at supplementary Sec. **??**.

Filtering. We employ a comprehensive filtering approach to ensure the fidelity of the dataset. This includes: (i) using the task predictor (Sec. 4.2) to reassign samples with instructions that should belong to another task, (ii) applying CLIP filtering metrics [2], (iii) employing structure preserving filtering based on the L1 distance between the depth map of the input image and the edited image, and (iv) applying image detectors to validate the presence (in Add task), the absence (in Remove task) or replacement (in Local task) of elements, according to the objects specified in the instruction. This process filters out 70% of the data, resulting in a final dataset of ten million samples.

4. Method

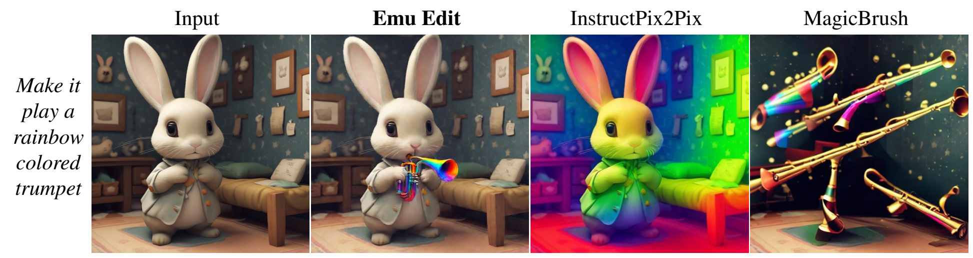
Emu Edit is a diffusion model designed to multi-task across a broad spectrum of editing tasks. As Emu Edit is trained on various tasks, a crucial aspect is the ability to identify the semantic edit (e.g., global/local/texture) that needs to be applied, based on the user instruction. To address this, we use task-specific embeddings integrated into the model, learned during training. After training, Emu Edit can quickly adapt to new tasks by learning a new task embedding through few-shot learning, without altering the rest of the model. We follow next with a detailed description of each part of our method.

4.1. Architecture

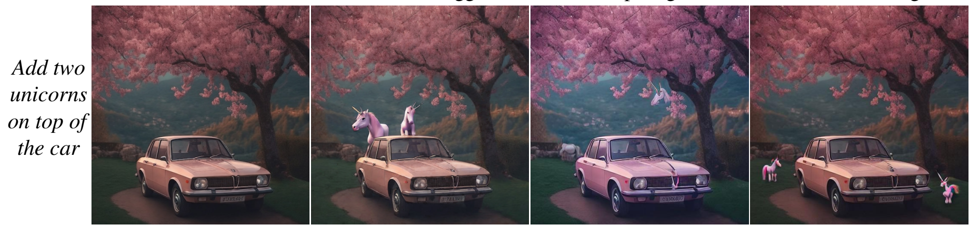
Our model builds upon the foundation set by Emu, which is outlined in [6]. The Emu model is a two-stage approach that begins with a pre-training phase and concludes with a quality fine-tuning stage. Emu adapted the latent diffusion model architecture [18] to support high-resolution image generation and incorporated a 16-channel autoencoder with encoder E and decoder D. In the following section, we adapt the notation of [2]. A large U-Net, ϵ_θ, with 2.8 billion parameters, θ, text embeddings from CLIP ViT-L [14] and T5-XXL [15], and a substantial pre-training dataset of 1.1 billion images facilitate the model's ability to learn complex semantics and finer details, with a noise-offset strategy contributing to high-contrast and aesthetically pleasing image generation. Given the encoded latent of an image $z = E(x)$, the diffusion process generates a noisy latent z_t where the noise level increases over timesteps $t \in T$. To convert Emu to an instruction-based image editing model, we condition it on the image to be modified c_I and the instruction c_T. Emu Edit is trained to minimize the following optimization problem,

$$\min_\theta \mathbb{E}_{y,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(z_t, t, E(c_I), c_T)\|_2^2\right] \quad (1)$$

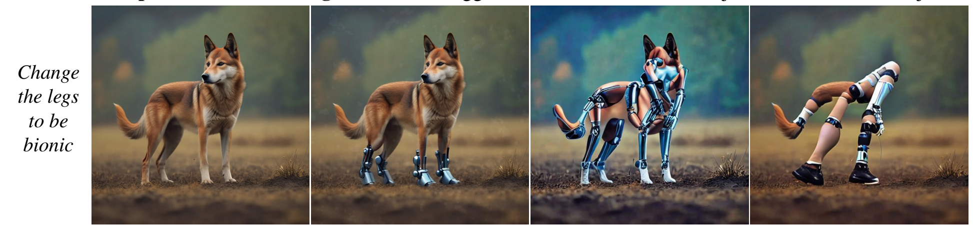
where $\epsilon \in N(0,1)$ is the noise added by the diffusion process and $y = (c_T, c_I, x)$ is a triplet of instruction, input image and target image from the dataset. In practice, we initialize the weights of Emu Edit with the weights of Emu. To support the image conditioning, we follow [2] and increase the number of input channels. New weights are initialized to zero. During inference, we perform classifier-free guidance on both image and text conditions. In our experiments we use a scale of $\gamma_I = 1.5$ for the image condition and $\gamma_T = 5.0$ for the text condition. Furthermore, we apply a rescaling of the diffusion scheduler to achieve a zero signal-to-noise ratio (SNR) at the terminal timestamp, as suggested



Global & Texture edit. Baselines struggle to execute complex global edits and texture changes.



Prepositions & Counting. Baselines struggle with relations between objects and number of objects.



Local. Baselines struggle to perform intricate local edits.

Figure 3. Failure cases of baseline instruction-based image editing models.[2]

in [10]. This is crucial in order to avoid any mismatch between the model's training and testing phases. For more implementation details see supplementary Sec. **??**.

4.2. Learned Task Embeddings

To guide the generation process toward the correct task, we learn an embedding vector for each task in the dataset. During training, given a sample from our dataset, we use the task index, i, to fetch the task's embedding vector, v_i, from an embedding table, and optimize it jointly with the model weights. We do so by introducing the task embedding v_i as an additional condition to the U-Net, ϵ_θ. Concretely, we integrate the task embedding into the U-Net via cross-attention interactions, and by adding it to the timestep embeddings. The optimization problem is updated to

$$\min_{\theta, v_1, \ldots, v_k} \mathbb{E}_{\hat{y}, \epsilon, t}\left[\|\epsilon - \epsilon_\theta(z_t, t, E(c_I), c_T, v_i)\|_2^2\right] \quad (2)$$

where k is the total number of tasks in our dataset and $\hat{y} = (c_I, c_T, x, i)$ is a quadruplet of input image, input instruction text, target image, and task index from the dataset. Task-specific conditioning arises from the observation that models lacking such conditioning can become perplexed about the type of edit required, particularly when the instructions are complex or the edit type is ambiguous. For instance, as visualized in Fig. 4, (1) a model without task conditioning might perform a global edit when a texture edit is required, (2) it might opt for segmentation when a global edit is necessary, and (3) it could implement a style edit in situations where a local edit would fit better. In the inference stage, we predict the task index. Specifically, we fine-tune a Flan-T5-XL model to identify the task at hand given the input instruction.

4.3. Task Inversion

To enable few-shot learning of new tasks without losing the general abilities of Emu Edit, we propose a method for adapting the model without changing the U-Net weights. Given a few examples of a new task, we learn a new task embedding, v_{new}. We freeze the model weights, and adapt it to the task only through the task embedding. Thus, to fit a new task embedding we solve the following optimization problem:

$$\min_{v_{\text{new}}} \mathbb{E}_{y, \epsilon, t}\left[\|\epsilon - \epsilon_\theta(z_t, t, E(c_I), c_T, v_{\text{new}})\|_2^2\right] \quad (3)$$

where v_{new} is the learned task embedding. Note that during task inversion y is a triplet belonging to the new task.

The model can then be employed for the new task by conditioning it on the learned task embedding, and it can still handle its original tasks by relying on the initial task embeddings. In Sec. 5.3, we demonstrate that our model effectively generalizes to novel tasks using this method.

[2]The samples depicted in this caption were selected by the authors. They do not cover all scenarios, but are meant to represent some common scenarios the authors encountered.

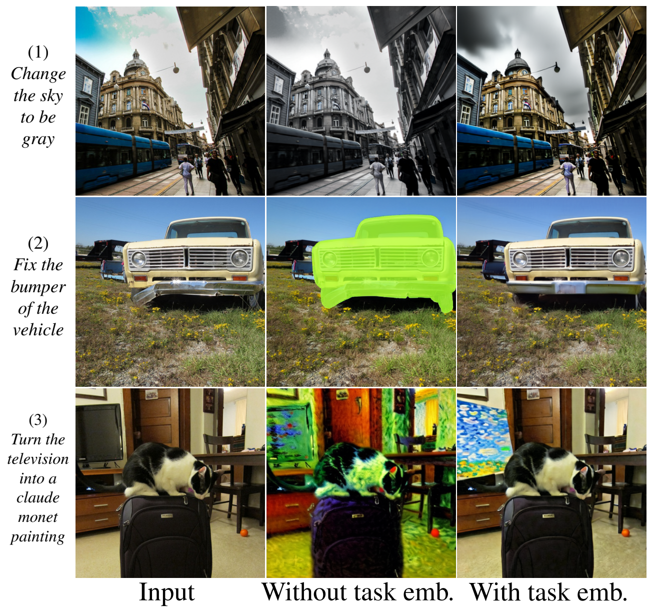

(1) Change the sky to be gray
(2) Fix the bumper of the vehicle
(3) Turn the television into a claude monet painting
Input
Without task emb.
With task emb.

Figure 4. **Task embeddings.** Model trained without task embeddings may get confused about the edit type when the instructions are complicated or there is ambiguity regarding the edit type: (1) Global edit (instead of Texture), (2) Segmentation (instead of Global), (3) Style edit (instead of Local).

Sequential Edit Thresholding. We notice that applying the model repeatedly, in multi-turn editing scenarios, aggregates reconstruction and numerical errors, which translate to noticeable artifacts. To mitigate this, we add a per-pixel thresholding step after each edit-turn. At each step s, we use the pixel value in the output image, c_I^{s+1}, only if its alteration surpasses a specific threshold. Otherwise, we keep the pixel value from the input image, c_I^s. Specifically, given an edit turn s, we compute the absolute difference image $d = \|c_I^{s+1} - c_I^s\|_1$ over the RGB channel, and apply the following thresholding:

$$c_I^{s+1} = \begin{cases} c_I^s & \text{if } \bar{d} < \alpha, \\ c_I^{s+1} & \text{otherwise.} \end{cases} \tag{4}$$

where, $\bar{d}$ is obtained after passing d through a low pass filter, in order to smooth the transition between previous and current pixels. In practice, we choose $\alpha = 0.03$. Please refer to supplementary Sec. **??** for a qualitative comparison. In Fig. 2 we show examples of multi-turn editing.

5. Experiments

Our experiments evaluate the ability of Emu Edit to follow user instructions faithfully and preserve the visual fidelity of the original image. First, we evaluate the performance of our approach on instruction-based image editing tasks. Second, we conduct a comprehensive ablation study to assess the effectiveness of our different contributions. Specifically, we ablate the contribution of the computer vision tasks to the model performance on image editing tasks, the importance of learned task embeddings, and the effect of multi-task learning on instruction-based image editing. Further ablation on our data generation pipeline can be found in the supplementary Sec. **??**. Finally, we demonstrate our model's ability to learn new tasks via few-shot learning.

Measures. We employ two main measures in our evaluation: edit text alignment and image faithfulness. Specifically, for each pair of input image and editing instruction, we use the following automatic metrics: (i) CLIP [14] text-image direction similarity (CLIP$_{dir}$) – measuring agreement between change in captions and the change in images, (ii) CLIP image similarity (CLIP$_{img}$) – measuring change between edited and input image, (iii) CLIP output similarity (CLIP$_{out}$) – measuring edited image similarity with output caption, (iv) L1 pixel-distance between input and edit image, and (v) DINO [4] similarity between the DINO embeddings of input and edited images. With the exception of the L1 distance, where lower values indicate better performance, higher values in all other measures signify better results. A low L1 distance translates to small changes in image's pixel values. A high DINO and CLIP$_{img}$ similarity score, suggests semantic similarity between the images. For region-based edits, high image similarity scores indicate the edits were precise. For free-form edits, high similarity scores indicate image structure preservation. CLIP$_{dir}$ and CLIP$_{out}$ measure how well the model followed the instruction. In addition, we asked human raters to evaluate the text alignment and image faithfulness. In each evaluation scenario, raters are presented with two modified images alongside the original input image and instruction, and are presented with two questions: (i) Image Faithfulness: which image better preserves elements in the input image, and (ii) Text Alignment: which image best follows the instruction.

5.1. Evaluation

Throughout the paper, we report results on the MagicBrush test set [25] and the Emu Edit benchmark. In the following section, we describe our motivation for creating this benchmark, and detail its curation process. To date, there are two main benchmarks for evaluating instruction-based image editing capabilities. First, the InstructPix2Pix benchmark [2], which is intrinsically biased due to its reliance on *generated* Stable Diffusion [17] input images, and GPT-3 [3] *generated* instructions. Consequently, it is unclear whether its results will truly mirror the performance on *real* input images, with *genuine* user instructions.

Unlike InstructPix2Pix, the second benchmark, MagicBrush [25], uses a diverse set of authentic input images from the MS-COCO benchmark [5, 11], and annotator-defined instructions. Nonetheless, this dataset also suffers

Table 2. Comparison with image-editing baselines evaluated on Emu Edit test set and MagicBrush test set. For each benchmark we report CLIP, L1, DINO metrics and human ratings. Human evaluation shows the percentage of raters that prefer the results of Emu Edit.

	Emu Edit Test Set							MagicBrush Test Set						
Method	CLIP$_{dir}\uparrow$	CLIP$_{im}\uparrow$	CLIP$_{out}\uparrow$	L1$\downarrow$	DINO$\uparrow$	Text align.	Image faith.	CLIP$_{dir}\uparrow$	CLIP$_{im}\uparrow$	CLIP$_{out}\uparrow$	L1$\downarrow$	DINO$\uparrow$	Text align.	Image faith.
InstructPix2Pix [2]	0.078	0.834	0.219	0.121	0.762	77.33	76.71	0.115	0.837	0.245	0.093	0.767	71.79	71.60
MagicBrush [25]	0.090	0.838	0.222	0.100	0.776	74.50	74.10	0.123	0.883	0.261	0.058	0.871	59.54	60.39
PnP [21]	0.028	0.521	0.089	0.304	0.153	98.95	99.00	0.025	0.568	0.101	0.289	0.220	97.24	96.96
Null-Text Inv. [12]	0.101	0.761	**0.236**	**0.075**	0.678	81.63	85.47	0.121	0.752	**0.263**	0.077	0.664	76.54	85.66
Our	**0.109**	**0.859**	0.231	0.094	**0.819**	–	–	**0.135**	**0.897**	0.261	**0.052**	**0.879**	–	–

from inherent bias. During data collection, annotators were directed to use the DALLE-2 image editing platform [13] to generate the edited images. Thus, this benchmark is biased towards editing instructions that the DALLE-2 editor can successfully follow, which may compromise both its diversity and complexity.

Emu Edit Benchmark. To collect a dataset with reduced bias and of higher diversity, we take a different approach. We first define seven different categories of potential image editing operations: background alteration (Background), comprehensive image changes (Global), style alteration (Style), object removal (Remove), object addition (Add), localized modifications (Local), and color/texture alterations (Texture). Then, we utilize the diverse set of input images from the MagicBrush benchmark [25], and for each editing operation, we task crowd workers to devise relevant, creative, and challenging instructions. Moreover, to increase the quality of the collected examples, we apply a post-verification stage, in which crowd workers filter examples with irrelevant instructions. Finally, to support evaluation for methods that require input and output captions [8, 21], we additionally collect an input caption and output caption. When doing so, we ask annotators to ensure that the captions capture both important elements in the image, and elements that should change based on the instruction. See supplementary Sec. **??** for examples of our benchmark, which we release to support better evaluation.

Baseline Comparisons. We compare our model against two instruction-based image editing baseline models: InstructPix2Pix [2], and MagicBrush [25], which is a variant of InstructPix2Pix that was fine-tuned on the MagicBrush dataset. Additionally, we compare our model against two text-based image editing methods: PNP [21] and Null-Text Inversion modification of P2P [8, 12]. Unlike instruction-based models, these works expect image descriptions. Therefore, we provide them with access to the input caption and output caption. Note however, providing these methods with access to the ground-truth captions could potentially offer an advantage over instruction-based models, since the automatic metrics also rely on these captions. Tab. 2 shows our results versus the baselines. The findings indicate that human raters consistently prefer Emu Edit over all baselines by a large margin. Furthermore, apart from Null-Text Inversion, which as explained above, utilizes the ground-truth captions during inference, our approach outperforms the existing baselines on the automatic metrics. We provide qualitative comparisons in Fig. 3. Additional comparisons are available in the supplementary: Fig.**??**-**??**, and Fig.**??**-**??**. For performance on vision tasks, see Sec. **??**.

5.2. Ablations

Computer Vision Tasks Enhance Image Editing Tasks. Here we demonstrate the importance of the vision tasks to Emu Edit performance on image editing tasks. For this, we trained two additional models on all tasks except: (i) detect and segment tasks, and (ii) image-to-image translation tasks. As we show in Tab. 4, adding the detection and segmentation tasks improves the model performance in region-based editing tasks. Additionally, we observe that image-to-image translation tasks improve the performance in free-form editing tasks. We hypothesize that the recognition tasks improve the model's recognition capabilities, leading to more accurate and precise localized modifications. Similarly, image-to-image tasks assist the model in understanding the entire image structure, thereby enhancing its capabilities for global operations.

Contribution of Learned Task Embeddings. We compare three variants of Emu Edit: (i) conditioned on the ground-truth task embedding, (ii) conditioned on the task embedding, as predicted by the task predictor described in Sec. 4.2, and (iii) without conditioning on the task type.[3] Tab. 3 shows the results on the validation set of our benchmark. As can be seen, conditioning on the task type boosts the model's performance. Furthermore, our task predictor closes the gap with the ground-truth conditioned model. Qualitatively, we observe that without conditioning on the task type the model may perform the wrong editing operation (Fig. 4). In supplementary Fig. **??**, we demonstrate the effect of manipulating the task while keeping the instruction and input image fixed. As can be seen, changing the task embedding directly influences the task executed by the model.

[3](iii) was trained without learned task embeddings.

Table 3. Learned task embeddings ablation on our validation set. We compare variations of Emu Edit: without task type condition, with predicted task type, and with ground-truth task type.

Method	CLIP$_{dir}$ ↑	CLIP$_{im}$ ↑	CLIP$_{out}$ ↑	L1↓	DINO↑
w/o task emb.	0.104	0.843	0.227	0.109	0.792
with pred. task	0.117	0.850	0.231	0.103	0.809
with gt task	0.119	0.852	0.231	0.100	0.811

Table 4. Contribution of computer vision tasks. Human evaluation is shown as a percentage of majority votes in favor of our model.

	Region-Based		Free-form	
Method	Text align.	Image faith.	Text align.	Image faith.
without detect/segment	**60.0**	**60.2**	52.3	51.5
without im2im translation	50.2	49.0	**58.0**	**60.1**

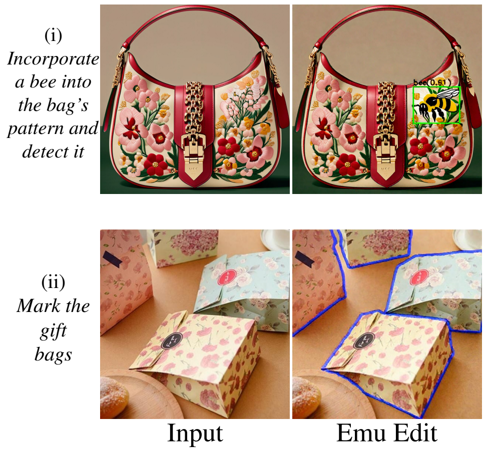


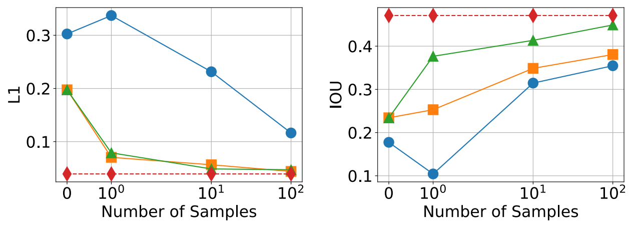
Figure 5. Generations on unseen tasks with task inversion. (i) composition of add and detect tasks, (ii) object contour detection.

Multi-Task versus Expert Models. We show that training a single model on a diverse range of tasks, leads to enhanced performance in each individual task, outperforming models that are specifically trained for a single task. To validate this, we train an expert model for each task, and compare its performance to the single one (See supplementary Tab. **??**).

Influence of Number of Tasks. Here, we ablate the number of tasks participating in the multi-task training scheme. In supplementary Fig. **??** we report the average CLIP$_{dir}$ on the Style and Texture tasks when iteratively excluding other tasks, and training a model on the new tasks list. As can be seen, augmenting the model with additional tasks leads to improved performance, even in tasks which are not directly associated with the added ones.

5.3. Few-Shot Learning of New Tasks

Finally, we evaluate our model's ability to generalize by testing it in a few-shot scenario with previously unseen tasks. We test its performance across the following tasks: (i) super-resolution (x4), (ii) object contour detection, (iii) mask-based inpainting, and (iv) a composite task formed by combining two tasks from our dataset: add and detect. For each task, we assess the model's performance when trained with 0, 1, and 100 examples, with the 0-example case being equivalent to a zero-shot setting. We compare three baselines: (i) Scratch – Emu Edit initialized with Emu's weights, trained on the examples, (ii) Task Inversion – Emu Edit with task inversion (Sec. 4.3), and (iii) Finetune – Emu Edit where we finetune all of the model's weights on the examples. As an upper-bound expert, we train the first baseline on 100,000 examples. Note that, the "Task Inversion" and "Finetune" baselines were trained on the multi-task dataset whereas the "Scratch" baseline was not. As can be seen in Fig. 6, fine-tuning with a single example is enough to significantly enhance the performance, while training from scratch results in overfitting. Moreover, utilizing 100 samples nearly achieves expert-level performance, implying that the model can effectively generalize to novel tasks. We also observe that task inversion performs similarly to complete fine-tuning, indicating that the model already contains essential information and can be prompted with a new task embedding to yield the desired result. For performance and generation examples on additional tasks see Fig. 5 and supplementary **??????**.



(a) Inpainting (b) Add + Detect

Figure 6. Few-shot performance for different tasks over 1, 10, and 100 samples. Each line represents a different training setting: 'Scratch' finetune (Blue, ◯), Emu Edit finetune (Orange, □), task inversion (Green, △), all compared to an upper-bound expert trained on 100k samples (Red dashed line, ◇).

6. Conclusion

Emu Edit presents a step change in instructable image editing capabilities, primarily due to its unique training on both recognition and generation tasks. This dual-focus approach significantly enhances the model's comprehension of natural language instructions, enabling it to accurately execute a wide array of editing operations. Its ability to generalize to new tasks with minimal examples further demonstrates its versatility and robustness. Furthermore, our framework has the potential for further integration with a multimodal LLM in future projects, particularly for complex editing tasks requiring detailed reasoning from the input image, such as counting objects or executing intricate tasks.

References

[1] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022. 3

[2] Tim Brooks et al. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023. 1, 3, 4, 6, 7

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. 3, 6

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 6

[5] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *ArXiv*, abs/1504.00325, 2015. 6

[6] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, et al. Emu: Enhancing image generation models using photogenic needles in a haystack. *arXiv preprint arXiv:2309.15807*, 2023. 4

[7] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision*, pages 89–106. Springer, 2022. 2

[8] Amir Hertz, Ron Mokady, Jay M. Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *ArXiv*, abs/2208.01626, 2022. 2, 3, 7

[9] Bahjat Kawar et al. Imagic: Text-based real image editing with diffusion models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6007–6017, 2022. 3

[10] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. *arXiv preprint arXiv:2305.08891*, 2023. 5

[11] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6

[12] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6038–6047, 2022. 3, 7

[13] OpenAI. Dall-e 2, 2022. https://openai.com/product/dall-e-2. 3, 7

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1, 4, 6

[15] 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. *Journal of Machine Learning Research*, 21(140):1–67, 2020. 4

[16] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2

[17] Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2021. 2, 6

[18] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 4

[19] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, Seyedeh Sara Mahdavi, Raphael Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *ArXiv*, abs/2205.11487, 2022. 2

[20] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 3

[21] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1921–1930, 2022. 3, 7

[22] Bram Wallace, Akash Gokul, and Nikhil Vijay Naik. Edict: Exact diffusion inversion via coupled transformations. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22532–22541, 2022. 3

[23] Su Wang et al. Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18359–18369, 2023. 3

[24] Shaoan Xie, Zhifei Zhang, Zhe Lin, Tobias Hinz, and Kun Zhang. Smartbrush: Text and shape guided object inpainting with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22428–22437, 2023. 3

[25] Kai Zhang et al. Magicbrush: A manually annotated dataset for instruction-guided image editing. *arXiv:2306.10012*, 2023. 1, 3, 6, 7