






End-to-End Visual Editing with a Generatively Pre-Trained Artist Supplementary Material

Andrew Brown^{1,2}, Cheng-Yang Fu², Omkar Parkhi²,
Tamara L. Berg², and Andrea Vedaldi^{1,2}

¹ Visual Geometry Group, University of Oxford {abrown}@robots.ox.ac.uk

² Meta AI Research {chengyangfu, omkar, tlberg, vedaldi}@fb.com
<https://www.robots.ox.ac.uk/~abrown/E2EVE/>

Table of Contents

A	Additional Qualitative Results	1
B	Additional Method Details	8
B.1	VQ-GAN Training	8
B.2	Constructing Evaluation Data Triplets	8
C	Additional Baseline Details	8
C.1	Constructing Inputs for Baseline methods	9
C.2	Baseline Implementation Details	9
C.3	Quantitative Analysis of Baseline Design Choices	10
D	Additional Quantitative Results	11
D.1	Computational Efficiency of E2EVE	11
D.2	Effect of Sampling Methods	12
D.3	Random Free-Form Masks	14
D.4	Additional Ablation Discussions	15

A Additional Qualitative Results

In this Section we describe additional qualitative results, and analyse cases of unreasonable generations from the model. All images in this supplementary material are sourced from UnSplash [10], or DFDC [2]. For videos demonstrating the capabilities of E2EVE, please see our website: <https://www.robots.ox.ac.uk/~abrown/E2EVE/>

Dresses-7m - block edits . In fig. 1 we show *block edits* from our method on the *Dresses-7m* dataset. To demonstrate the model robustness and sample diversity, we generate edits for the same source image, while varying the edit region and driver image. The generated edits are remarkable. E2EVE generates natural-looking edits, that are local to the edit-region and faithful to the driver images. Additionally, the edits are visually diverse, and contain many different clothing

structures, styles, patterns and colors, with the same two edits rarely sharing the same generated content. Neither the source image, nor any of the driver images in fig. 1 are depicted in *Dresses-7m*. However, E2EVE demonstrates zero-shot generalisation capabilities to the new patterns and structures in these images in order to blend the content of the driver images cohesively with the source image.

Dresses-7m - free-form edits . We show additional *free-form* edits from E2EVE on the *Dresses-7m* dataset in fig. 2. Although all structural information besides the outline of the clothing item (*i.e.* the outline of the mask) are masked in the source image, E2EVE generates natural and diverse clothing structures for the same source image and edit region (*e.g.* see the different waist and neckline structures in fig. 2a), that are faithful to the driver images. Impressively, although the training set of *Dresses-7m* contains only dresses, E2EVE is able to generalise well to an edit region depicting a T-shirt in fig. 2b.

Because the edit region R fully contains an object (in this case, a clothing item), and separates this object in the foreground from the background, we are hence able to paste the edited region back on to the source image, without creating any disjoint spatial continuity over the edit region boundary in the output. This has the effect of removing any non-local effects from the edit, while preserving the naturalness of the source image outside of the edit region. All generated results in fig. 2 are hence shown with the generated edit region pasted back onto the source image.

Unreasonable Generations - E2EVE . E2EVE generates very impressive results even in cases when the driver and source images are highly uncorrelated (see Fig. 5 in main paper, and fig. 1 in the supp. mat.). However, there are still some edits proposed by the model that are unreasonable. This is not a problem unique to our approach, and is due to the model lacking a full understanding of the semantic content of images. fig. 3 highlights three cases of unreasonable edits that were seen in a small number of generated images from E2EVE using *block-edits* trained on *Dresses-7m*.

fig. 3 left: E2EVE leaves part of the masked edit region in the output generated image. E2EVE simply marks the masked region in the model inputs by leaving a R -shaped hole in the masked source image (Section 3.2 in main paper), and very occasionally, part of this hole is left in the model output. Interestingly, this only happens when R includes the bottom-most rows of the image.

fig. 3 middle: E2EVE generates an unnatural-looking face. The *Dresses-7m* dataset contains some faces. When trained on this dataset, the model hence learns that faces are likely to appear in the top-most parts of images. When the edit region includes the top-most part of the image, E2EVE may generate an unnatural looking face. These generations could be avoided by removing faces from the training set.

fig. 3 right: E2EVE generates an unreasonable edit when the source and driver images are completely mis-aligned. In this case E2EVE is asked to blend some legs into the top of a dress. This is an impossible edit to complete naturally, and one that would not appear in the self-supervised training regime used

by the method. Impressively, E2EVE is still able to generate cohesive blends occasionally when given such non-aligned inputs (see top-right-most generation in fig. 1).

FFHQ - Comparisons. In fig. 4 we show some qualitative comparisons to prior work for the models trained on the FFHQ dataset. Our method generates edits that are natural-looking, faithful to the driver, and that are local, whereas the prior work struggles to achieve all three (see Table 1 in the main paper for the quantitative demonstration of this trend).

FFHQ - E2EVE. In fig. 5 we show additional qualitative results from our method trained on FFHQ. Here, we probe whether E2EVE is able to make different edits to the same source image, or make the same edit to many different source images. Specifically, we choose three driver image and edit region pairs, (y, R) , and pair them with a number of varied source images.

E2EVE generates natural-looking edits, even when the driver and source images are non-aligned (*e.g.* due to the different genders in the source and driver images). Impressively the generated images are faithful to the specific details of the driver image across all of the source images. For example, the style of the edited glasses and new hairstyle specifically match the corresponding driver images. This highlights the capability of image-based visual editing to explicitly edit specific visual content. Furthermore E2EVE makes stylistic changes to the driver image to make for a more natural edit that is cohesive with the surrounding source image. For example, the added hair (both head and beard) are manipulated in color and texture to roughly match the rest of the hair in the image.

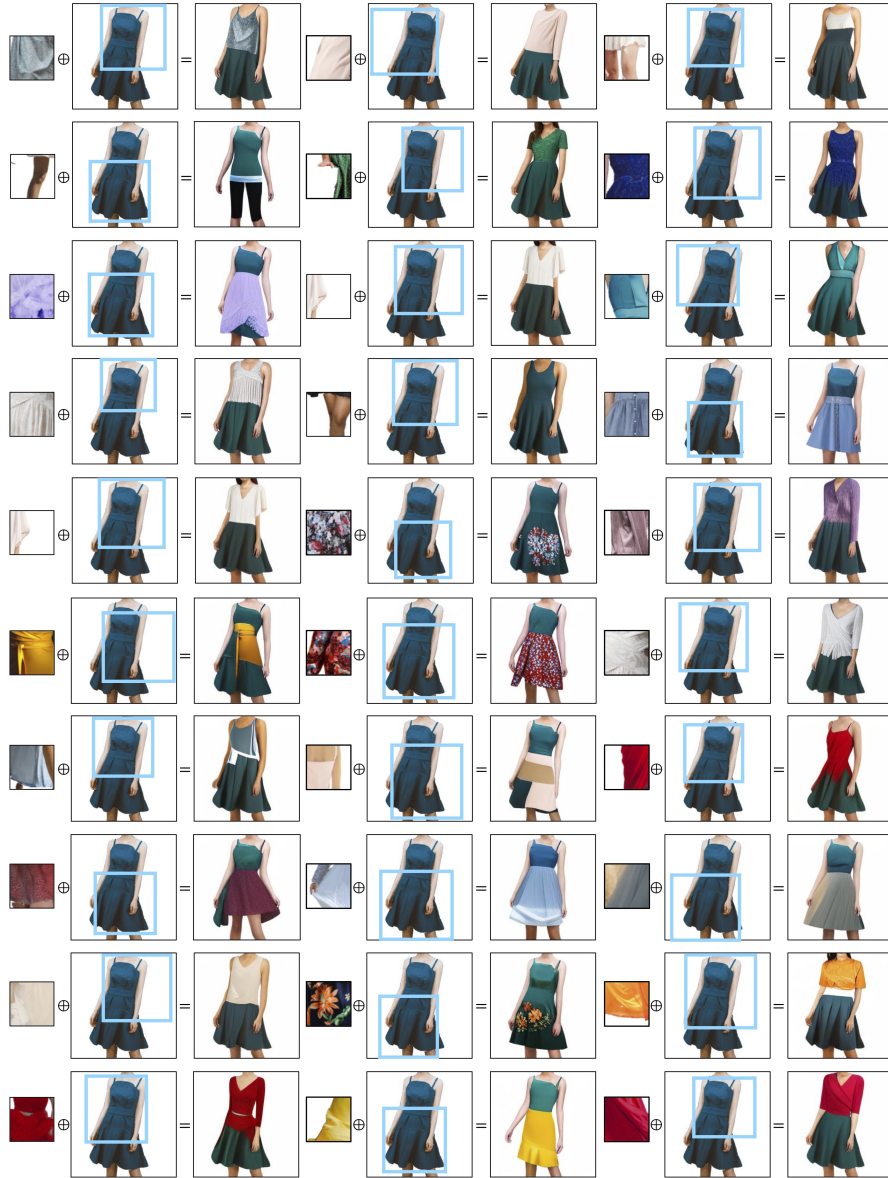


Fig. 1. Qualitative results from the *block edits* version of E2EVE trained on the *dresses-7m* dataset. To demonstrate the robustness of our method and the visual diversity of the generations, we fix the source image, and vary both the edit region and driver images. In each case, the masked region of the source image is that contained within the blue line. Please zoom in for details. Images are sourced from UnSplash [10].

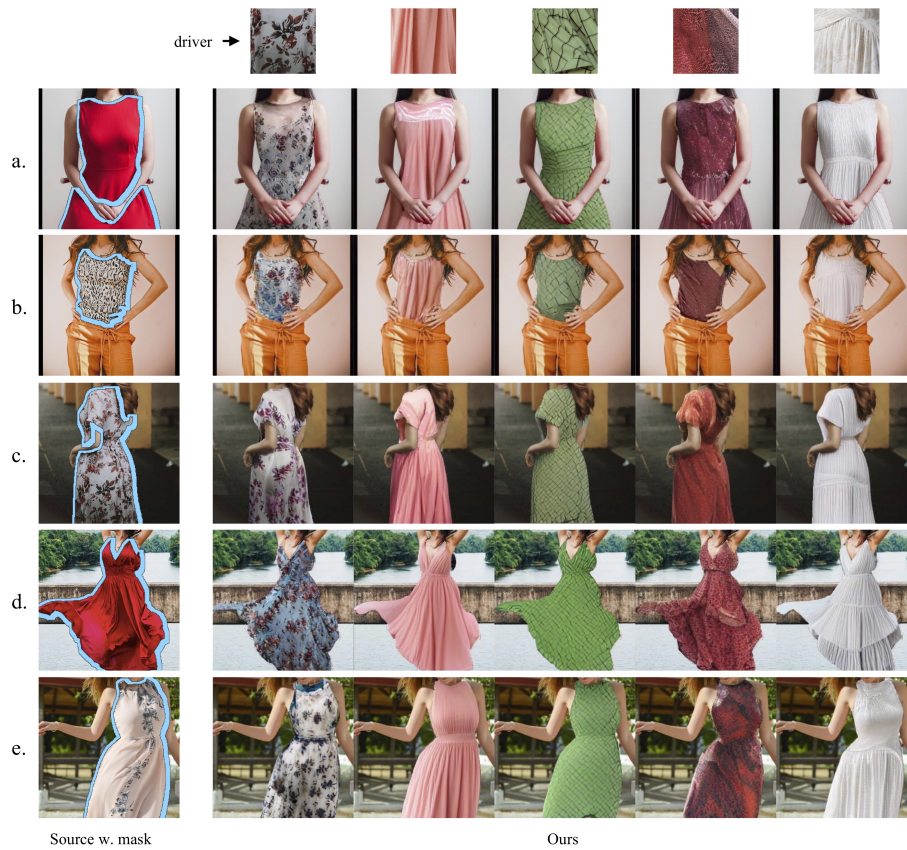


Fig. 2. Qualitative results from the *free-form* edits version of E2EVE trained on the *dresses-7m* dataset. We show all image generation permutations for 5 masked source images, and 5 driver images. In each case, the masked region of the source image is that contained within the blue line. Please zoom in for details. Images are sourced from UnSplash [10].

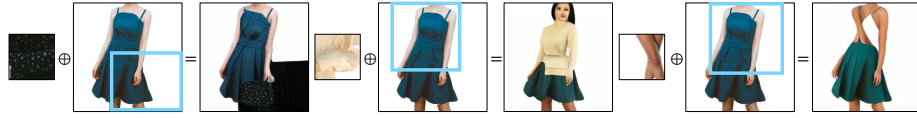


Fig. 3. Examples of unreasonable generations from E2EVE. In a small number of samples from the *block-edits* model trained on *Dresses-7m*, some unreasonable generations are seen. Left: Part of the edit region is left in the output image. Middle: An unnatural looking face is generated occasionally when the edit region fills the top-most part of the source image. Right: Non-aligned inputs can lead to unreasonable generations.

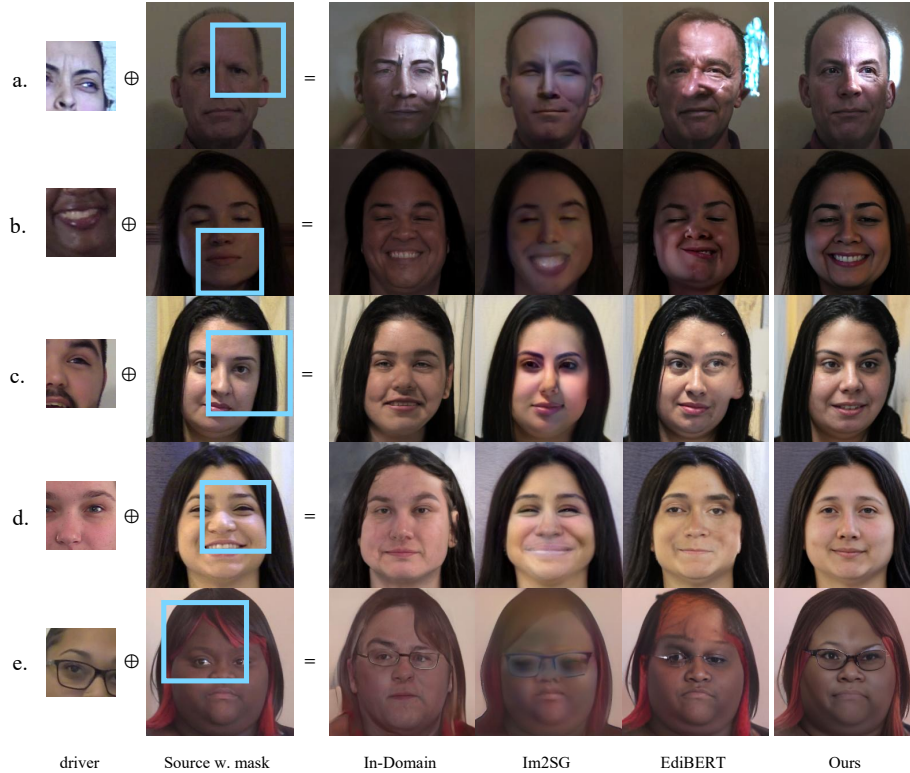


Fig. 4. Qualitative comparisons to prior work when trained on FFHQ. Our method generates edits that are natural-looking, faithful to the driver and local to the edit region, whereas prior work struggles to achieve a balance of all three. In each case, the masked region of the source image is that contained within the blue line. Images are sourced from DFDC [2].

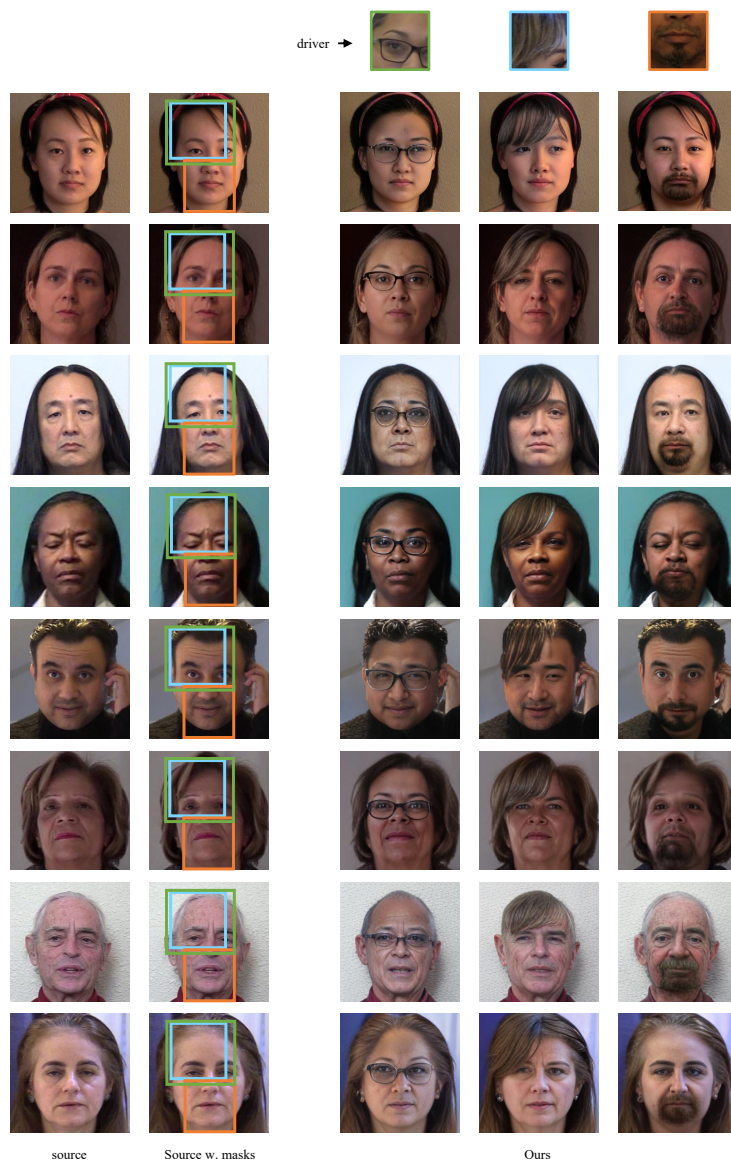


Fig. 5. Qualitative results for our method when trained on FFHQ. For each source image (row) we use the same three driver image and edit region pairs to generate three edited images. To save space, we display the three different edit regions used for the three samples on the same image (the *source w. masks* column). The left column of generated images corresponds to the green edit region, the middle column corresponds to the blue edit region, and the right column corresponds to the orange region. This figure is best viewed in color. Images are sourced from DFDC [2].

B Additional Method Details

In this Section, we provide further details on how our method is implemented. Specifically we detail how the VQ-GANs that we use in our method are trained. This is in addition to Section 3.3 in the main paper. We also give further clarification on how triplets are formed for the evaluation data. This is in addition to Section 4 in the main paper.

B.1 VQ-GAN Training

Here, we explain the loss function for training the VQ-GAN. Following [3], the VQ-GAN is trained to optimize the following loss function:

$$L_{VQ}(E, G, Z) = \|x - \hat{x}\|^2 + \|sg[\Phi(x)] - z\|_2^2 + \|sg[z] - \Phi(\hat{x})\|_2^2 \quad (1)$$

where x is the input image, $\hat{x} = \Psi(z)$ is the reconstructed image, $z = \Phi(x)$ are the discrete codes, Φ is the encoder, and Ψ is the paired decoder. The first term is a reconstruction loss, the second a loss pulling the codebook vectors towards the encoder outputs, and the third term is the commitment loss [7], which makes sure that the encoder commits to an embedding and the output does not grow arbitrarily. Here, $sg[\cdot]$ is the stop-gradient operation. Backpropogation through the non-differentiable quantization step is computed via a straight through gradient estimator. For more details and motivation see [3, 7, 8, 9].

B.2 Constructing Evaluation Data Triplets

Here, we give further clarification on how evaluation data triplets are formed. Following Section 4 in the main paper, each triplet (x, y, R) in the evaluation data consists of the source image x , the edit region R , and the driver image $y = x'|_R$ that is taken from the same spatial location (*i.e.* the coordinate centres of both driver image and edit region align), but from a different image x' . We add the further detail that the edit region R is purposefully chosen to be larger than the driver image y . Due to the lack of strict alignment in the datasets (apart from FFHQ), this rule ensures that there is a high chance of there existing a feasible edit for each triplet.

C Additional Baseline Details

In this Section, we provide further details on how the baselines that we compare to in our experiments are implemented. We explain how we form the inputs for the baselines in section C.1, we list the hyper-parameters and code sources for the baseline methods in section C.2, and we analyse the quantitative effect of certain baseline implementation design choices in section C.3.

C.1 Constructing Inputs for Baseline methods

Here, we detail how we form the inputs for each of the prior work baselines. Whereas our method takes as input the driver image y and masked source image x as separate pieces of conditioning information, the prior work baselines instead take as input a composited image, where the driver image has been pasted onto the source image. In the corresponding papers for each of the baseline approaches, the composited image is constructed such that the driver and source images semantically align in terms of both scale and positioning. This is done either manually, or by relying on the strict alignment of datasets such as FFHQ. However, in our work, we are not limited to aligned datasets, and wish to avoid manual intervention when forming evaluation data. A challenge hence exists in how to construct the composited images automatically from the evaluation data triplets (x, y, R) .

Recall that for quantitative evaluation, we generate 10 edits from each method for each evaluation data triplet. To this end, for each evaluation data triplet, we form 10 composited images for each evaluation data triplet by pasting the driver image at different positions within the edit region R in x . The edit region R is larger than the driver image y so simply pasting y into R in x without resizing would still leave a hole in x within R around the driver y . This would unfairly disadvantage the prior work, as such a hole would place the composited image out of the domain of natural images that the prior work models have been trained on. Instead, we inpaint the remaining hole with the underlying image content from x . This should instead have the opposite effect of advantaging the prior work baselines over our method, seeing as they are shown more of the source image x in their input.

For the experiments with *free-form edits* in the main paper, we create the composited input image for EdiBERT by tiling the driver image such that it can be pasted into R in x .

C.2 Baseline Implementation Details

Here, we detail how the baseline methods that are compared to in Section 4 in the main paper were implemented. Where possible, we use official code repositories, and used the default hyper-parameters recommended by the authors. For the cases where we used hyper-parameters different to those recommended by the authors, we have provided quantitative analysis justifying these choices in section C.3.

EdiBERT [5]. We use the official code repository and follow the author’s guidance for their *image composition* experimental setting. Namely, we dilate the mask by 1 token to reduce border irregularities, we periodically collage the image with the input, and use spiral ordering for sampling edited tokens. An additional parameter is the number of optimisation epochs. In each optimisation epoch, all tokens in the edit region are updated once. Although the authors set the number of optimisation epochs to 2 for image composition, we find empirically that using

		Naturalness (↓)		Faithfulness (↑)			Locality (↓)
		Image	Edit-R	R@1	R@5	R@20	(L1)
Dresses-7m (<i>block-edits</i>)	Baseline: Copy-Paste	21.457	35.924	1.000	1.000	1.000	0.000
	Baseline: Inpaint	15.797	25.769	0.071	0.214	0.515	0.095
	EdiBERT [5] (1 epoch) ★	17.193	32.621	0.554	0.837	0.963	0.052
	EdiBERT [5] (2 epoch) †	16.058	31.800	0.404	0.711	0.923	0.052
	EdiBERT [5] (3 epoch)	15.570	31.727	0.325	0.625	0.890	0.052
	(ours) E2EVE	14.411	24.743	0.797	0.937	0.978	0.056
FFHQ (<i>block-edits</i>)	Baseline: Copy-Paste	33.330	25.811	1.000	1.000	1.000	0.000
	Baseline: Inpaint	18.328	12.665	0.421	0.704	0.895	0.139
	In-domain [11] ★	19.880	14.270	0.539	0.800	0.938	0.178
	In-domain [11] w. reg †	24.192	13.733	0.953	0.988	0.995	0.321
	EdiBERT [5] (1 epoch) ★	13.192	12.230	0.718	0.925	0.983	0.093
	EdiBERT [5] (2 epoch) †	13.450	10.874	0.675	0.895	0.976	0.092
	EdiBERT [5] (3 epoch)	13.496	10.739	0.640	0.870	0.970	0.092
(ours) E2EVE	12.770	10.574	0.853	0.970	0.994	0.106	
LSUN Bedrooms (<i>block-edits</i>)	Baseline: Copy-Paste	24.402	28.828	1.000	1.000	1.000	0.000
	Baseline: Inpaint	15.080	21.493	0.113	0.297	0.596	0.161
	In-domain [11] ★	32.333	43.544	0.171	0.363	0.608	0.208
	In-domain [11] w. reg †	46.566	42.718	0.677	0.815	0.914	0.326
	EdiBERT [5] (1 epoch) ★	16.518	27.528	0.537	0.816	0.946	0.111
	EdiBERT [5] (2 epoch) †	15.791	26.234	0.392	0.712	0.903	0.111
	EdiBERT [5] (3 epoch)	15.696	27.384	0.316	0.629	0.871	0.111
(ours) E2EVE	14.107	22.187	0.789	0.923	0.981	0.119	

Table 1. Results for *block-edits* when analysing design choices for baseline implementations. When implementing prior work, we use default recommended implementation settings where possible. However, we find that different implementation settings for In-Domain and EdiBERT lead to a more preferable balance of the metrics. Here, we analyse the effect of these implementation details. Key: † refers to the design choice recommended by the authors of the respective paper. ★ refers to the design choice that we report numbers for in the main paper. Results for our method and the simple baselines have been included for ease of reference.

1 epoch obtains a better balance between metrics, and these are the results that we report in the main paper.

GAN inv [1]. We use the official code repository of StyleGAN2-Ada [6] for the implementation of GAN inv [1]. We use the default recommended optimization hyper-parameters for projecting given images into a pretrained GAN latent space.

In-Domain [11]. We use the official code repository, and follow their default implementation for *Semantic Diffusion*. The In-Domain approach follows a two stage pipeline, with a GAN inversion stage, followed by a domain-regularised optimisation stage. We find empirically that removing the regularisation stage results in a better balance between metrics, and these are the results that we report in the main paper.

C.3 Quantitative Analysis of Baseline Design Choices

Here, we analyse the quantitative effect of two baseline implementation design choices. Specifically, we analyse the effect of the number of optimisation epochs

in EdiBERT, termed *EdiBERT* [5] (*n epoch*), where *n* refers to the number of optimisation epochs. We also analyse the effect of either including the regularisation stage in the In-Domain method (termed, *In-domain [11] w. reg*), or not (termed, *In-domain [11]*). We use the same metrics as used in Section 4 in the main paper, namely, naturalness, faithfulness and locality. The results are shown in table 1.

For EdiBERT, increasing the number of optimisation epochs results in more natural-looking samples, but this is at the cost of a sharp drop in faithfulness. This is as expected because the EdiBERT optimization procedure improves the likelihood of the generated image with respect to the learnt unconditional image prior, with little constraint in keeping faithfulness to the driver. We choose to report numbers for 1 epoch of optimization in the main paper, as this offers the most competitive balance between metrics. For all versions of EdiBERT, our method is still superior in terms of naturalness and faithfulness, as reported in the main paper.

For In-Domain, adding the regularisation means that the model becomes far more faithful to the driver image, but at the cost of a large drop in locality and naturalness. In fact, for the FFHQ dataset, the regularisation method outperforms E2EVE in terms of faithfulness, but this is at the cost of the source image being no longer recognizable with poor naturalness and locality of 0.321. Hence, we report metrics in the main paper for In-Domain without the regularisation, as only in this version of the method where the editing can be considered local.

D Additional Quantitative Results

In this Section we provide additional quantitative results. These results explore the computational efficiency of our approach compared to prior work (section D.1), different sampling techniques (section D.2), and the use of random free-form masks on FFHQ (section D.3)

D.1 Computational Efficiency of E2EVE

The throughput (measured in generated images/second) for E2EVE compared to each baseline method is shown in table 2.

E2EVE notably achieves higher throughput than some GAN-based approaches (GAN inv, and In-domain w. reg). Because E2EVE is trained end-to-end for the editing task, generated images can be sampled directly from the model. This avoids the costly test-time optimisation processes necessary for these GAN-based approaches.

Additionally, E2EVE achieves comparable throughput with the attention-based baseline EdiBERT, and even achieves higher throughput when EdiBERT uses more than 1 optimisation epoch (the authors recommend using 2). Whereas E2EVE samples every token of the output generated image during inference, EdiBERT only samples the tokens within the edit region. This means that EdiBERT can achieve higher throughput by requiring less sampling iterations, but this is at

Method	Throughput (↑) img/s
GAN inv [1]: StyleGANv2	00.01
GAN inv [1]: StyleGANv2-Ada	00.01
In-domain [11] w. reg	00.17
EdiBERT [5] (3 epoch)	00.23
EdiBERT [5] (2 epoch)	00.33
EdiBERT [5] (1 epoch)	00.67
In-domain [11]	33.33
(ours) E2EVE	00.27

Table 2. Computational efficiency of E2EVE compared to baseline approaches as measured by throughput (generated images per second). Throughput is computed via time taken to generate a single sample with batch size of 1 on an NVIDIA A100. throughput is averaged over multiple samples.

the cost of not making any non-local edits that may be necessary for improving the overall naturalness of the generated image (see section 4.2 in main paper). Although E2EVE samples more tokens than EdiBERT during inference, E2EVE uses a significantly smaller backbone transformer (24 vs 32 layers), leading to comparable throughput times.

In-domain achieves very competitive throughput when not using the regularisation stage (*In-domain* [11]). This speed is expected from and is an advantage of the simple encoder-decoder inversion architecture. However, this fast inference speed comes at the cost of significantly worse generated samples than E2EVE across naturalness, faithfulness and locality (see Table 1 in the main paper and Section C.3).

D.2 Effect of Sampling Methods

We explore the effect of different sampling methods in fig. 6 on the faithfulness, naturalness and locality metrics across all datasets. At each sampling step during inference, a token is sampled from the output probability histogram from the model. Here, we analyse two different sampling techniques: first, top-k sampling, where the probabilities are first sorted, and only the top-k are sampled from. Second, top-p sampling (*nucleus* sampling [4]), where the probabilities are sorted, and only those with a cumulative probability less than the p-value are sampled from.

A top-p value of 0 and a top-k value of 1 results in a deterministic (or *greedy*) sampling process where the most likely token is sampled at each step. A top-p value of 1 and a top-k value of 1024 means that every token is considered at each step. Several interesting conclusions can be made.

First, deterministic sampling leads to a sharp drop in naturalness and faithfulness. This is an expected result, as simply choosing the highest probability token at each step tends to not result in the most probable sequences.

Second, aside from deterministic sampling, the performance across all metrics and datasets is fairly consistent and robust across all top-p and top-k values,

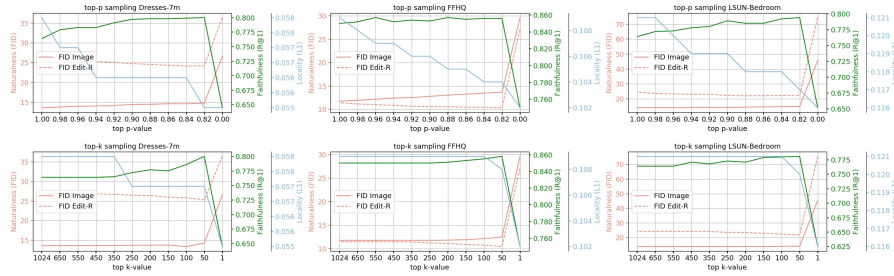


Fig. 6. The effect of different sampling methods on the naturalness, faithfulness and locality metrics across different datasets. We analyse the effect of the p-value and k-value for top-p and top-k sampling methods, respectively.

as indicated by the small range that the metrics change over outside of a top-p value of 0 and top-k value of 1. This is in contrast to [3] where naturalness (FID) was observed to severely degrade for unconditional generation when all tokens are sampled over. This indicates that in our case the distribution at each sampling step is narrow/peaked, meaning that there are just a few tokens with high probability that can be sampled from reasonably. Hence, the most likely tokens hold so much of the probability mass that considering the long tail does not affect the generated images drastically. This makes sense, because rather than generating images unconditionally, E2EVE is generating edited images, where the content of the output image is often easily predicted from the conditioning information.

Third, locality is optimal for deterministic/greedy sampling and degrades once more tokens are considered in the sampling. To explain this, we note that simply copying the source image outside of the edit region would lead to the best performance in the locality metric. We conjecture that the most probable token at each step outside the edit region corresponds to the spatially corresponding token in the source image. By observing the probability histograms outside of the edit region, we see that often there is one token that takes almost all of the probability mass, and this likely corresponds to the corresponding token in the source image. When more tokens are considered during sampling, occasionally non-local edits are sampled from the long tail of the histogram.

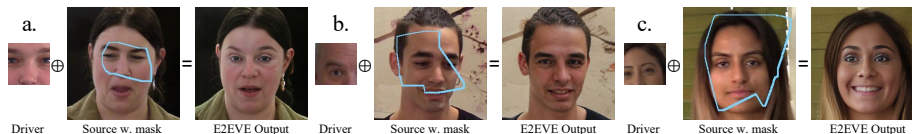


Fig. 7. E2EVE generates impressive samples with small (a,b) or large (c) *randomly* shaped R (blue region), offering greater creative user-control.

Table 3. Comparing E2EVE trained on randomly shaped edit regions, R .

Random R size	Naturalness (FID)			Faithfulness (R@1)			Locality (masked L1)		
	S	M	L	S	M	L	S	M	L
Base: Copy-Paste	80.926	83.219	96.521	0.478	0.539	0.618	0.000	0.000	0.000
Base: Inpaint	26.695	22.204	21.044	0.671	0.408	0.145	0.139	0.155	0.174
In-domain [86]	34.884	39.443	43.317	0.069	0.442	0.752	0.183	0.206	0.228
(ours) E2EVE	14.927	13.169	13.299	0.911	0.838	0.669	0.102	0.112	0.121

D.3 Random Free-Form Masks

Here, we explore the use of random masks for the masked region in the source image, R . Qualitative results are included in Figure 5 in the main paper, and also copied here in fig. 7.

In table 3, we include additional experiments using 10,240 “random” free-form edit regions R drawn independently of the image contents and of size small (S=5-20% of the image), medium (M=20-40%), and large (L=40-60%). They can cover small features or large regions (*e.g.* haircuts or expressions) as in fig. 7-a,b,c.

E2EVE generates significantly more natural, local and faithful samples than the baselines except for large R s, where faithfulness decreases slightly while maintaining naturalness. The latter results are however visually impressive, significantly extrapolating the small driver image as needed to make the output result cohesive (fig. 7c).

Table 4. Model ablations and sweeps for *block edits*. Key: NLL: Negative log likelihood. α , pos-aug, size-aug: the parameters used to define the sub-cropping transformation T . Filter: filtering E2EVE samples by visual similarity to the driver image. 2VQ: using two VQ-GANs rather than one. Datasets: D : *Dresses-7m*, B : Bedrooms, F -FFHQ.

α	pos-aug	size-aug	Filter	2VQ	Data	Naturalness (\downarrow)		Faithfulness (\uparrow)			Locality (\downarrow)	NLL (\downarrow)	Diversity (\uparrow)	
						Image	Edit-R	R@1	R@5	R@20	(L1)	Image	Edit-R	
a.	0.8				D	17.241	30.076	0.882	0.980	0.996	0.056	2.181	0.135	0.309
b.	0.6				D	15.593	29.364	0.920	0.986	0.997	0.056	1.704	0.137	0.315
c.	0.4				D	13.967	26.975	0.811	0.954	0.988	0.056	1.594	0.139	0.327
d.	0.0				D	15.797	25.769	0.071	0.214	0.515	0.095	1.537	0.190	0.419
e.	0.6	\checkmark			D	15.605	26.513	0.887	0.980	0.997	0.056	1.518	0.142	0.338
f.	0.5-0.6	\checkmark			D	14.951	26.186	0.856	0.968	0.992	0.056	1.460	0.143	0.344
g.	0.4-0.7		\checkmark		D	14.589	27.824	0.864	0.970	0.992	0.056	1.494	0.139	0.328
h.	0.4-0.7	\checkmark	\checkmark		D	14.411	24.743	0.797	0.937	0.960	0.056	1.448	0.143	0.344
i.	0.4-0.7	\checkmark			D	13.913	24.583	0.611	0.817	0.929	0.056	1.448	0.145	0.351
j.	0.4-0.7	\checkmark			B	14.347	22.998	0.636	0.831	0.929	0.119	2.942	0.287	0.460
k.	0.4-0.7	\checkmark			F	12.636	10.699	0.723	0.899	0.976	0.106	2.392	0.203	0.321
l.		EdiBERT [5]			B	16.643	29.775	0.356	0.627	0.823	0.111	-	0.291	0.575
m.		EdiBERT [5]			F	13.036	12.891	0.536	0.778	0.925	0.093	-	0.181	0.423
n.	0.4-0.7	\checkmark	\checkmark	\checkmark	D	14.107	23.916	0.720	0.891	0.963	0.056	1.454	0.144	0.347

D.4 Additional Ablation Discussions

Here, we include further discussions on the model ablations, as an extension to Section 4.3 in the main paper. We include the model ablation Table again here in table 4. Not filtering the samples by similarity to the driver (rows i,j,k) reduces faithfulness of both E2EVE and EdiBERT (rows l,m); even so, E2EVE outperforms all prior work in this metric (see Table 1 in the main paper). Finally, training just one VQ-GAN rather than two (row n) results in a drop in faithfulness, as the single VQ-GAN struggles to faithfully reconstruct the details in the smaller driver image.

Bibliography

- [1] Abdal, R., Qin, Y., Wonka, P.: Image2stylegan++: How to edit the embedded images? In: CVPR (2020)
- [2] Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., Ferrer, C.C.: The deepfake detection challenge dataset (2020)
- [3] Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: CVPR (2021)
- [4] Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y.: The curious case of neural text degeneration. In: ICLR (2020)
- [5] Issenhuth, T., Tanielian, U., Mary, J., Picard, D.: Edibert, a generative model for image editing. arXiv:2111.15264 [cs.CV] (2021)
- [6] Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., Aila, T.: Training generative adversarial networks with limited data. In: NeurIPS (2020)
- [7] van den Oord, A., Vinyals, O., Kavukcuoglu, K.: Neural discrete representation learning. In: NeurIPS (2017)
- [8] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: ICML (2021)
- [9] Salimans, T., Karpathy, A., Chen, X., Kingma, D.P.: Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In: ICLR (2017)
- [10] UnSplash: Unsplash - www.unsplash.com
- [11] Zhu, J., Shen, Y., Zhao, D., Zhou, B.: In-domain gan inversion for real image editing. In: ECCV (2020)