Does FLUX Know What It's Writing?

Adrian Chang *
Independent

Sheridan Feucht *
Northeastern University

Byron Wallace
Northeastern University

David Bau
Northeastern University

Abstract

Text-to-image models are historically bad at generating text within images (e.g., a slogan on a t-shirt), but recent state-of-the-art models like FLUX.1 have shown significant improvements in legible text generation. Does this mean that FLUX has learned abstract representations of the letters it is generating? We investigate the implicit representations of inpainting diffusion models by printing characters onto an evenly spaced grid and prompting the model to fill in masked characters. By probing the latent representations of these character grids in various components of the model, we find evidence of generalizable letter representations in middle transformer layers that suggest a notion of letter identity consistent across fonts.

1 Introduction

The ability to distinguish a letter from its rendered form and extrapolate it across different contexts is considered a hallmark of human cognition: in 1995, Hofstadter went as far as to argue that the question "What is the letter 'a'?" may be "the central problem of AI" [9, 13]. In this work, we investigate how FLUX.1 [10] writes text.

Previous image generative models were notoriously bad at rendering text, but the latest generation of text-to-image diffusion models, such as FLUX.1 and Qwen-Image [14], are considerably better at this task. While these models consistently render legible characters, they can only generate coherent text when the prompt contains explicit instructions of what to write. Without explicit instructions, text generated by FLUX.1 qualitatively mimics the visual aspects of language without legible content, interspersing English words with gibberish made from legible characters (see Figure 1 for examples).

Does this grasp on generating legible characters indicate that FLUX.1 has developed symbolic, reusable representations of letters? In this work, we find preliminary evidence that FLUX.1 has an abstract notion of letters beyond their rendered form. We do this by designing visual prompts that require an understanding of characters independent of their rendered style to complete, evaluating whether the FLUX.1-Fill inpainting model is able to complete these patterns. We then probe the latent representations of these character grids in various components of the model and find evidence of general letter representations in middle transformer layers.

2 Character Inpainting

We use image inpainting [2, 12] to investigate an image generative model's understanding of letters independent of their rendered form. Similar to previous work in visual prompting [1], we repeat words on an image grid and mask out characters which represent either a variation in font or casing.

*Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability Workshop at NeurIPS 2025.

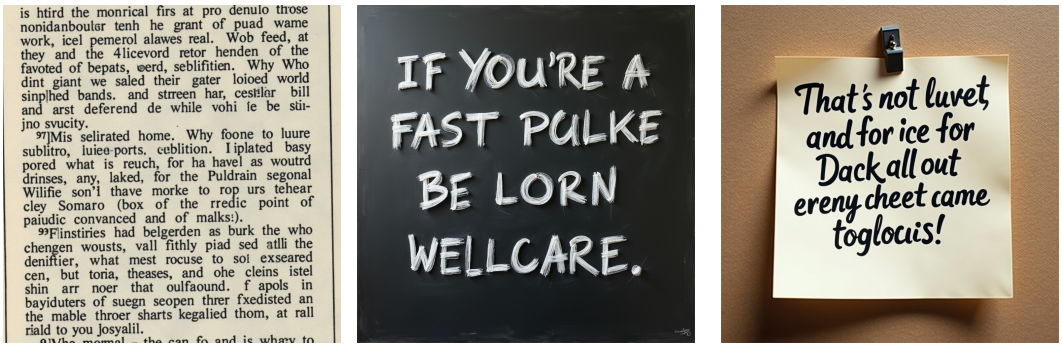

Figure 1: Examples of images generated by FLUX.1-Dev when not given specific text to generate. Left to right: “part of a newspaper clipping, short, legible, in English”, “writing a long message on a blackboard in English”, “a message written in sharpie on a sticky note in English”. Qualitatively, letters are remarkably well-formed, with some English words mixed in with nonsensical text.

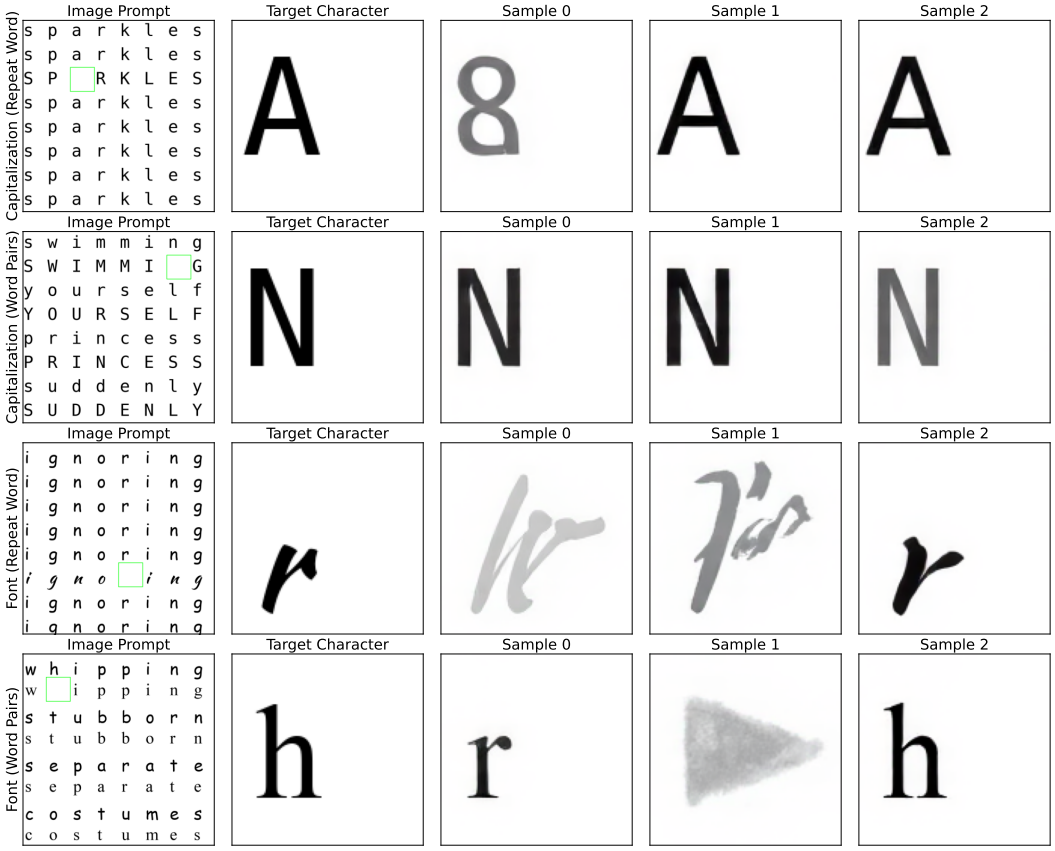

Figure 2: We use image inpainting to investigate an image generative model’s understanding of letters independent of their rendered form. Left column: we repeat words on an image grid and mask out characters which represent either a variation in font or casing. The "Repeat Word" setting repeats the same word across the whole image and varies a single line. The "Word Pairs" setting prints pairs of words, each containing a varied and unvaried example. Right columns: We show inpainting results for three random seeds for each image above, compared to the ground truth target character.

Table 1: We report FLUX.1-Fill's accuracy inpainting the correct character, casing, and font across all settings. "RW" stands for "Repeat Word" and "WP" stands for "Word Pairs" (See Figure 2). We calculate results for the first timestep, $t = 1$, as well as full generation, $t = 49$. For capitalization, 'Character %' indicates the percent of inpainted letters that are the correct character *and* the correct case, whereas 'Casing %' only indicates the percentage of inpaints that are the correct case. For the font task, 'Character %' does not consider whether the font is also correct. Fonts are classified using a custom-trained model that may not generalize to noisy FLUX.1-Fill outputs; see Appendix D for 120 randomly-selected examples.

	Word			Gibberish		
	Character %	Casing %	Font %	Character %	Casing %	Font %
Random Baseline	2.0	50.0	25.0	2.0	50.0	25.0
Capitalization (RW, t=49)	39.53	**66.80**	_	20.07	**54.53**	_
Capitalization (RW, t=0)	**45.67**	63.47	_	**31.67**	50.33	_
Capitalization (WP, t=49)	24.13	28.00	_	**2.60**	**22.80**	_
Capitalization (WP, t=0)	**37.80**	**30.60**	_	1.20	19.60	_
Font (RW, t=49)	40.53	_	29.20	28.47	_	23.00
Font (RW, t=0)	**61.40**	_	**30.40**	**50.40**	_	**25.80**
Font (WP, t=49)	34.87	_	**82.53**	**3.33**	_	**71.73**
Font (WP, t=0)	**41.80**	_	78.33	1.33	_	61.80

2.1 Approach

As this particular image prompting setting is underexplored, we experiment with two approaches: the "Repeat Word" setting repeats the same word across the whole image, varying a single line, whereas the "Word Pairs" setting prints pairs of words, each containing a varied and unvaried example. Figure 2 shows examples of these visual prompts, their target characters, and samples from the inpainting model. We evaluate both words sampled from TinyStories [5] and random strings across the four setup. Additionally, four different fonts are used for rendering text: DejaVu Sans Mono, Times New Roman, Comic Sans, and Kaushan Script. For each setting, we draw 30 samples each for 50 different images.

We use the FLUX.1-Fill model [10] with an empty prompt and guidance of 0 to ensure unconditional generation. When the mask used to guide inpainting is too large or small, the model is prone to generating visual content other than characters—to minimize this effect, we measure accuracy of copying characters as grid size varies, and use the grid size with the highest reported accuracy (as seen in Figure 6). The highest overall accuracy achieved is about 70% for an 8x8 grid of exactly-repeated characters (which may be a source of noise).

We evaluate model generations for 50 timesteps. However, to measure how "easy" this task is, we also visualize model outputs at the first timestep, $t = 1$. To do so, we apply $\hat{x}_0$ visualization [7], an approach that allows us to visualize "the direction a model is headed" at a given timestep. Put simply, this approach extrapolates $\epsilon_\theta(x_t, t)$ (the noise predicted by the model parameterized by θ at timestep t) across all remaining timesteps.

The character and font of the inpainted patch are identified with PP-OCRv-5 [3] and a Resnet34 model [8] trained to recognize fonts, respectively. PP-OCRv-5 reports a 86.79% recognition accuracy on printed english and our font classifier achieves a 100% test accuracy on held out characters. The OCR model only sees the character patch, so recognition results for letters whose form does not change with casing (e.g. 'o' or 'z') may be noisy.

2.2 Results

Table 1 reports the accuracy of inpainting the correct character, casing, and font across all settings. As classifying model generations is somewhat noisy, Appendix D shows 120 randomly-selected inpainting generations which are more representative of the model's consistency for this task.

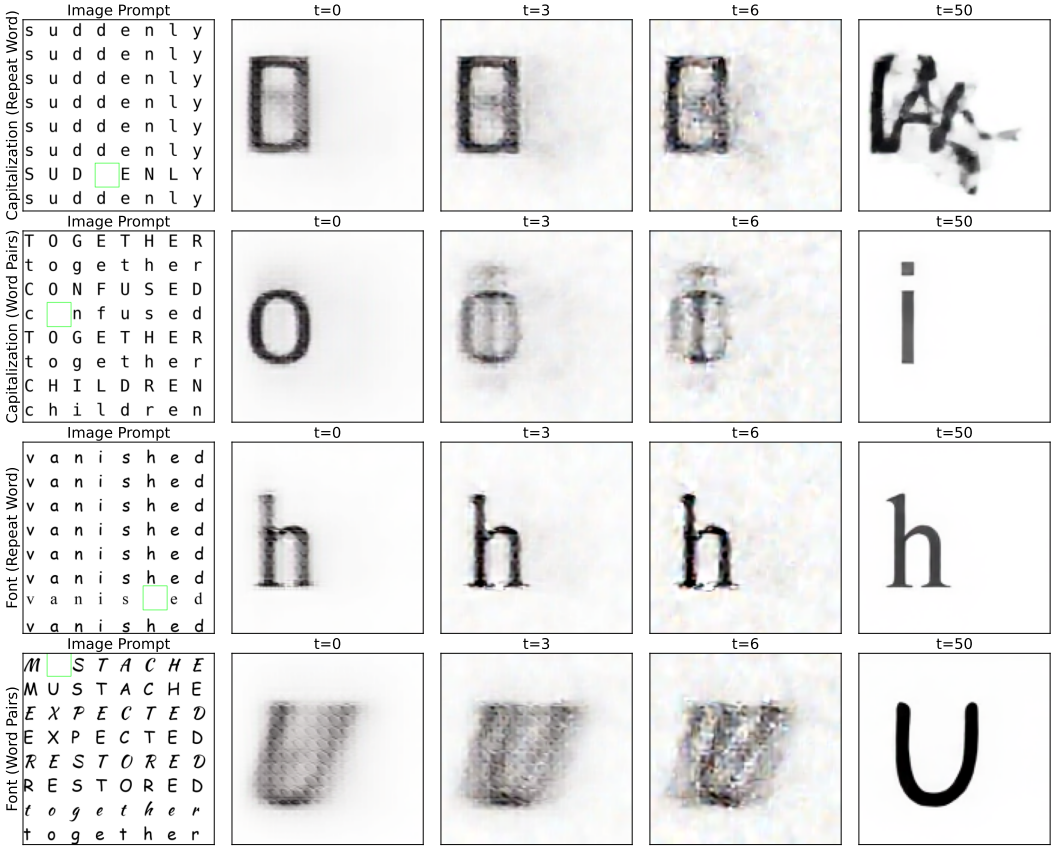

Figure 3: $\hat{x}_0$ visualization [7] of character inpainting across timesteps, where $t = 0$ indicates the first timestep and $t = 49$ is the final output. FLUX.1-Fill is often correct at the first timestep, but after subsequent iterations it sometimes "changes its mind." For example, the bottom row shows that the model is beginning to correctly inpaint a Kaushan Script 'U', which is later incorrectly replaced by a verbatim copy of the adjacent Comic Sans 'U'.

Overall, character accuracy is quite high relative to a random baseline, but capitalization and font transfer are more noisy. Inpainting random strings instead of words decreases character accuracy across most settings—perhaps because the model does not recognize any semantic textual content in such cases, and so focuses instead on matching the visual pattern of the image prompt. FLUX.1-Fill becomes particularly bad at inpainting the correct character when given *pairs* of gibberish words; this may be because it is unable to recognize the repetition of adjacent rows. However, word pairs appear to make it easier for the model to match the desired font, possibly because there is more of that font present in the image.

Turning our attention to Figure 3, we find a qualitative explanation for why $t = 49$ and $t = 0$ have comparable performance in Table 1. Although $\hat{x}_0$ visualization sometimes produces blurry images that may decrease OCR accuracy, FLUX.1-Fill also often "changes its mind" to the *wrong* answer throughout the denoising process. This may explain why $t = 0$ often outperforms $t = 49$, despite the blurriness of these visualizations.

3 Probing for Symbolic Representations

FLUX.1-Fill's ability to inpaint the correct character in Section 2 is promising. It is unclear, however, whether or not this is because the model has learned a general representation of letters, or because the model has memorized common fonts. To elucidate this distinction, we train linear probes on VAE latents as well as internal transformer representations of letter patches. We include the VAE in our

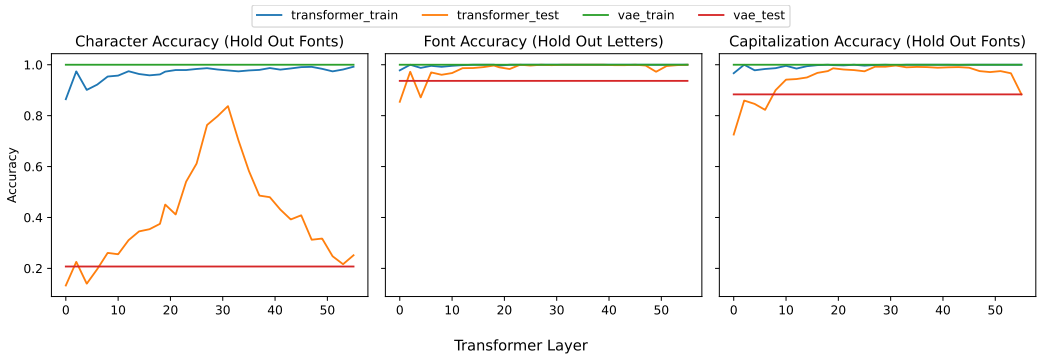

Figure 4: We train three types of linear probe: A character classifier tested on unseen fonts, a font classifier tested on unseen letters, and a capitalization classifier tested on unseen fonts. Font and casing probes already achieve high test accuracy on VAE latents, but test accuracy becomes perfect when trained on intermediate transformer activations. Strikingly, character probes can only generalize to unseen fonts in transformer middle layers, indicating the existence of a general letter representation only at those layers.

analysis to understand what symbolic representations it already encodes (if any), and what symbolic representations the transformer must learn.

3.1 Approach

We train three kinds of linear probes: a character classifier tested on unseen fonts, a font classifier tested on unseen letters, and a capitalization classifier tested on unseen fonts. We take all the patches corresponding to a given letter in the prompt image and concatenate them to obtain a single vector for that character; transformer hidden states are taken from the first denoising timestep. Note that we exclude the masked portion of the image, focusing only on parts of the image that are already fully-formed.

3.2 Results

Figure 4 shows the training and test accuracies for these probes. Font and casing probes already achieve high accuracy on VAE latents, so it is unsurprising that the probes trained on transformer representations would also generalize. Character probes, however, only generalize to unseen fonts in transformer middle layers, indicating the existence of a general letter representation that emerges *only* at those layers. The VAE represents letters on a literal (almost pixel-wise) level, with only surface level visual characteristics such as font and casing being probe-able. The transformer, on the other hand, has been trained with text supervision, and has seemingly learned abstract letter representations to generate text.

One to way to interpret the "peak" in Figure 4 is that FLUX.1-Fill must map its literal pixel-wise representations to a more abstract conceptual space that is more useful for generation [11, 4, 6]. Thus, character probe test accuracy decreases after middle layers as the model has to instantiate these abstract representations into concrete forms; i.e., it must specifically indicate what noise to subtract from the latents by the end of the network.

4 Conclusion

In this work, we use inpainting to investigate whether diffusion models abstract away from surface forms to learn representations of letters that are invariant across font and casing. We introduce a visual inpainting setting which requires an abstract notion of letters to complete and probe FLUX.1-Fill hidden states to understand where these representations are localized. These experiments provide an initial glimpse into the implicit symbolic representations learned by diffusion models, helping us to understand what image models may be able to learn about text.

References

[1] Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei A. Efros. Visual prompting via image inpainting, 2022.

[2] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 417–424, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[3] Cheng Cui, Ting Sun, Manhui Lin, Tingquan Gao, Yubo Zhang, Jiaxuan Liu, Xueqing Wang, Zelun Zhang, Changda Zhou, Hongen Liu, Yue Zhang, Wenyu Lv, Kui Huang, Yichao Zhang, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. Paddleocr 3.0 technical report, 2025.

[4] Clément Dumas, Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Separating tongue from thought: Activation patching reveals language-agnostic concept representations in transformers, 2025.

[5] Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

[6] Sheridan Feucht, Eric Todd, Byron Wallace, and David Bau. The dual-route model of induction. In *Second Conference on Language Modeling*, 2025.

[7] Rohit Gandikota and David Bau. Distilling diversity and control in diffusion models. *arXiv preprint arXiv:2503.10637*.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[9] Douglas Hofstadter and Corporate Fluid Analogies Research Group, editors. *Fluid concepts and creative analogies: computer models of the fundamental mechanisms of thought*. Basic Books, Inc., USA, 1995.

[10] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[11] Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference?, 2025.

[12] Weize Quan, Jiaxi Chen, Yanli Liu, Dong-Ming Yan, and Peter Wonka. Deep learning-based image and video inpainting: A survey. *Int. J. Comput. Vision*, 132(7):2367–2400, January 2024.

[13] Joshua B. Tenenbaum and William T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12:1247–1283, 2000.

[14] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025.

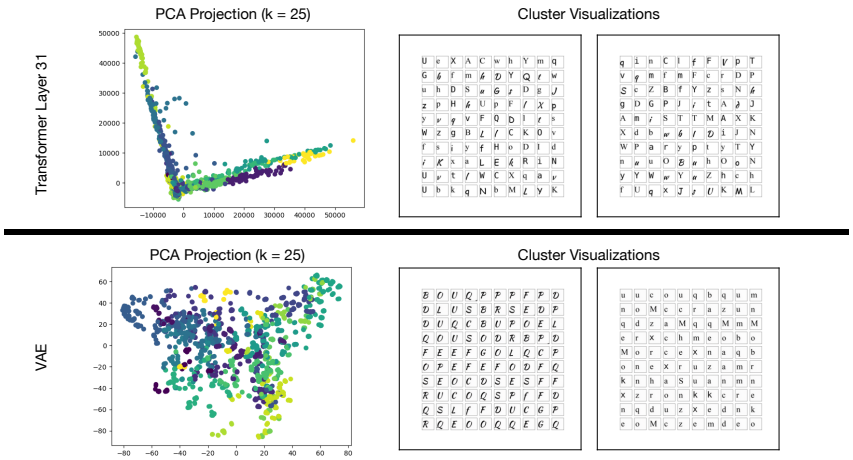

Figure 5: We find that k-means clusters in the VAE latent space tend to group letters by stylistic factors such as font and casing. However, k-means clustering on transformer representations from layer 31 (the middle of the model) returns clusters that are largely uninterpretable. Each color represents a cluster, with $k = 25$.

A K-means Clustering of Character Latents

The results of the character classification probe suggest that clustering the character patch representations from the transformer at layer 31 with k-means would produce clusters containing the same letter, and VAE clusters around capitalization and font. While VAE k-means clusters tend to group around stylistic factors such as font and casing, transformer clusters are largely uninterpretable. Representative samples of these clusters as well as their PCA projection are shown in Figure 5.

B Probing Train and Test Set Construction

We describe in this section how the train and test sets for the probing experiments were constructed. Each setting involves randomly sampling a held out target characteristic. This process is repeated n times with the train and test accuracies averaged and reported in the final figure. For character accuracy, a random font is chosen to be held out in the test set and the train set contains character latents from all other fonts. For font accuracy, 3 random characters are held out in the test set and the train set contains character latents from all other characters. Capitalization accuracy uses the same train and set set construction as character accuracy.

C Variation in Character Cell Size

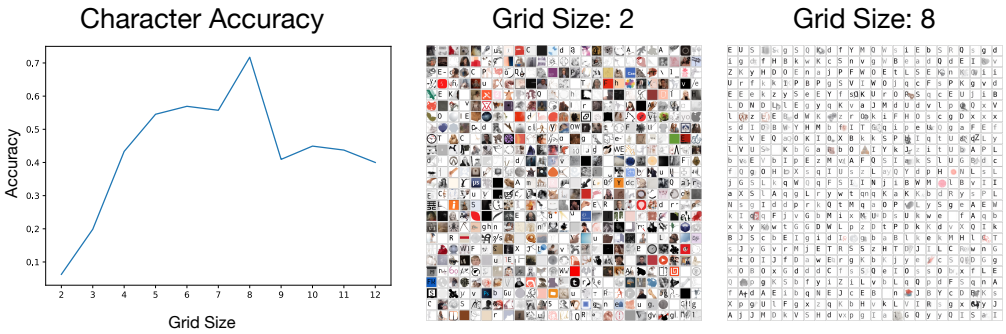

Figure 6: When the mask used to guide in painting is too large or small, the model is prone to generating visual content other than characters. In order to minimize this effect we look at accuracy of copying a character as grid size varies and use the grid size with the highest reported accuracy.

D Inpainting Examples

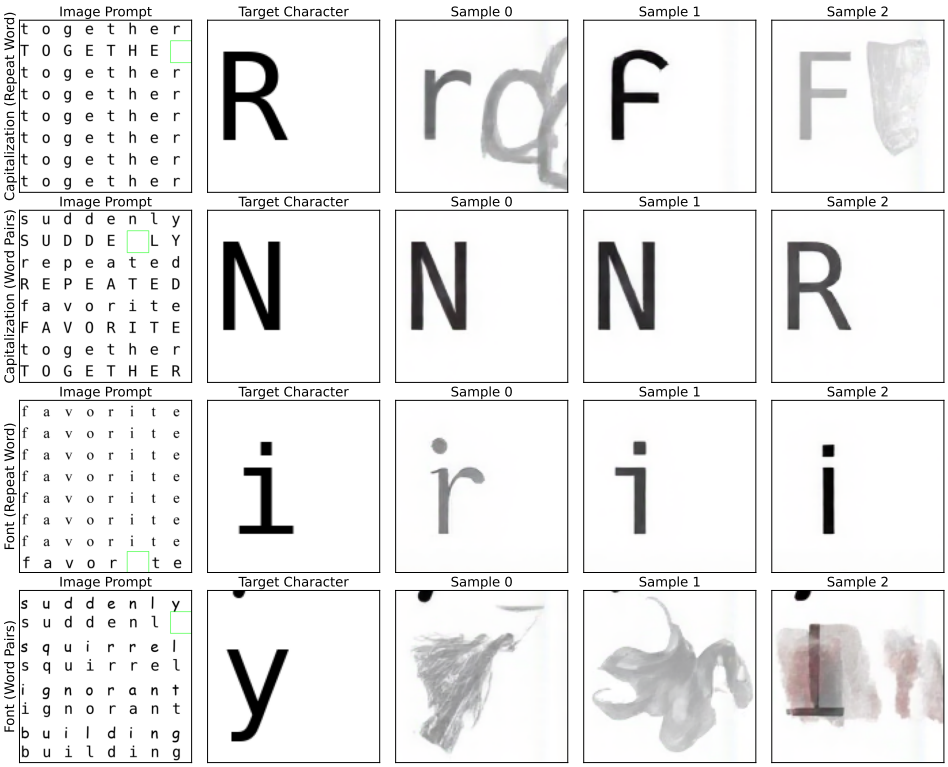

Figure 7: More examples of inpainting.

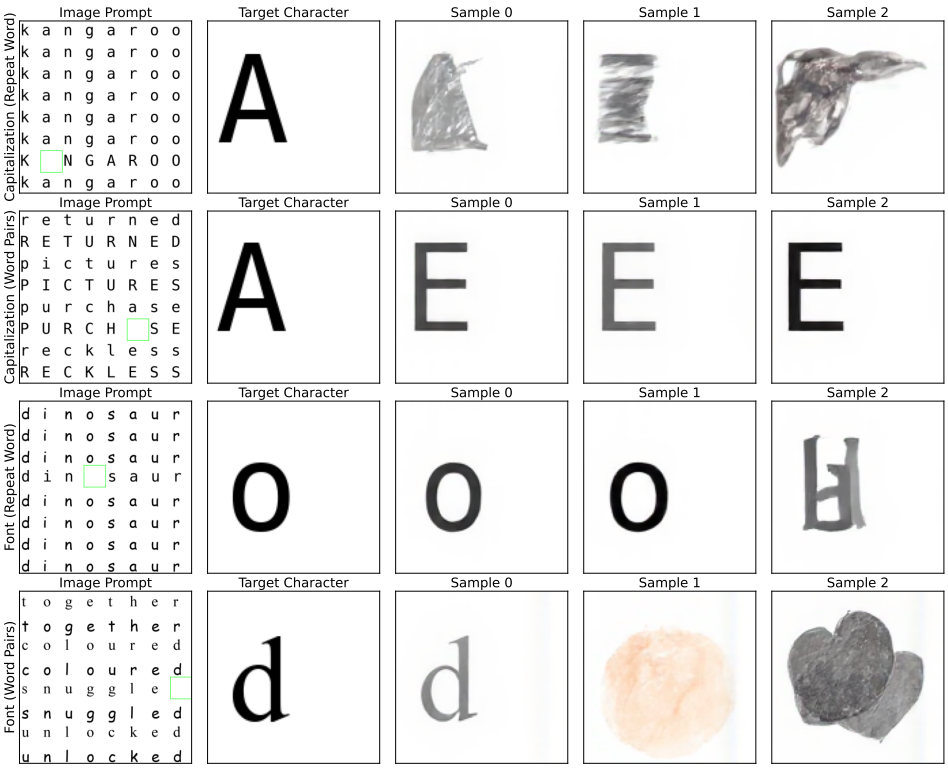

Figure 8: More examples of inpainting.

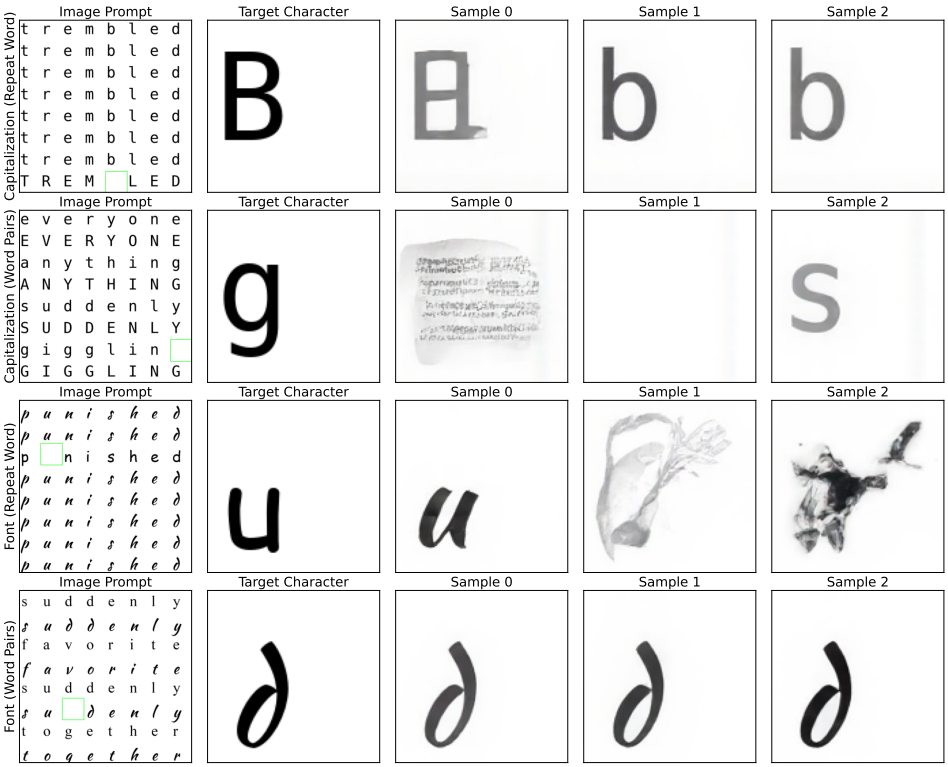

Figure 9: More examples of inpainting.

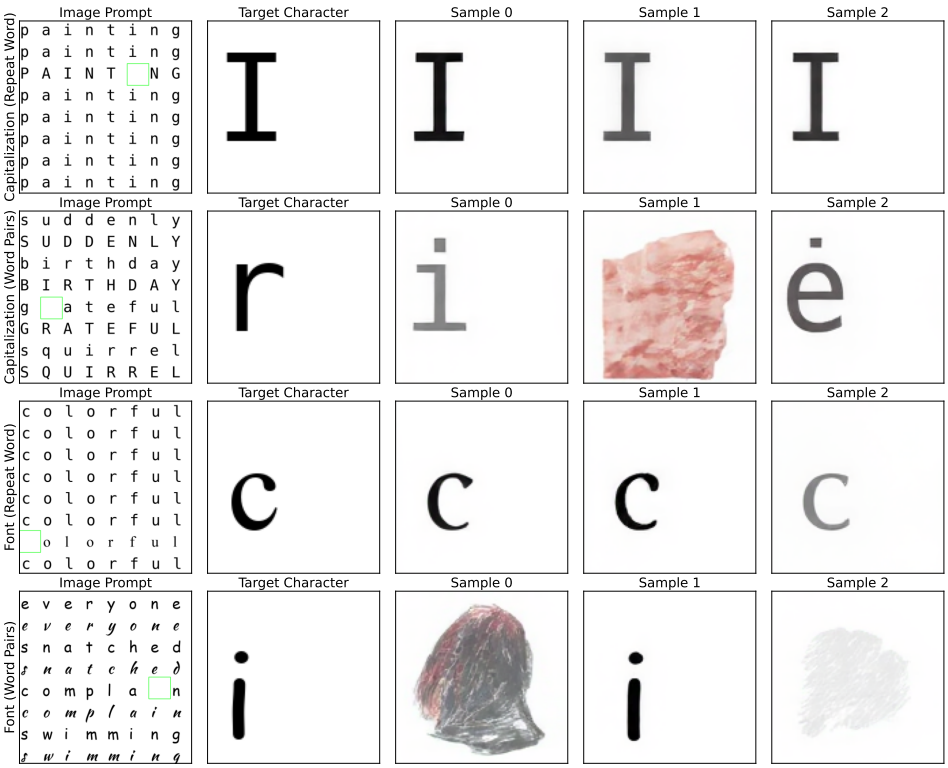

Figure 10: More examples of inpainting.

Figure 11: More examples of inpainting.

Figure 12: More examples of inpainting.

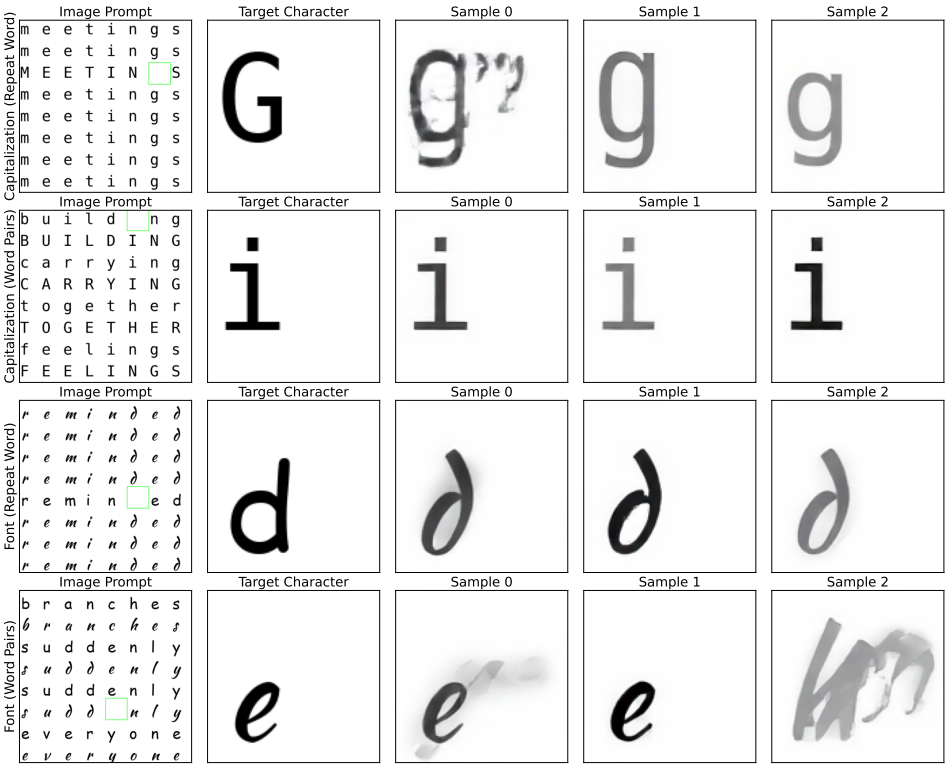

Figure 13: More examples of inpainting.

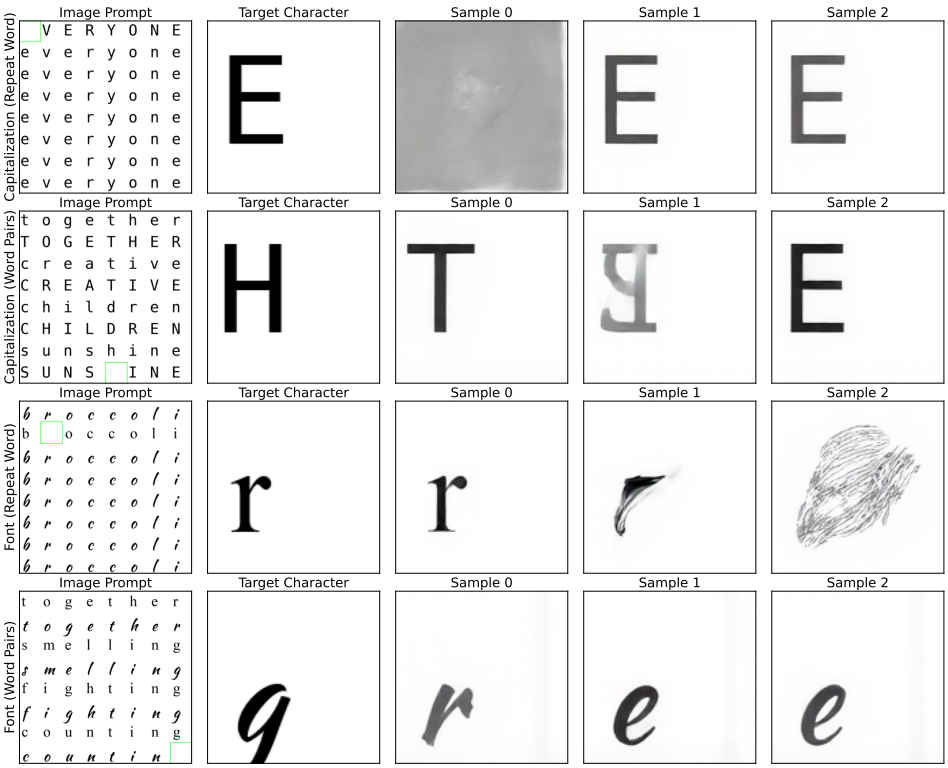

Figure 14: More examples of inpainting.

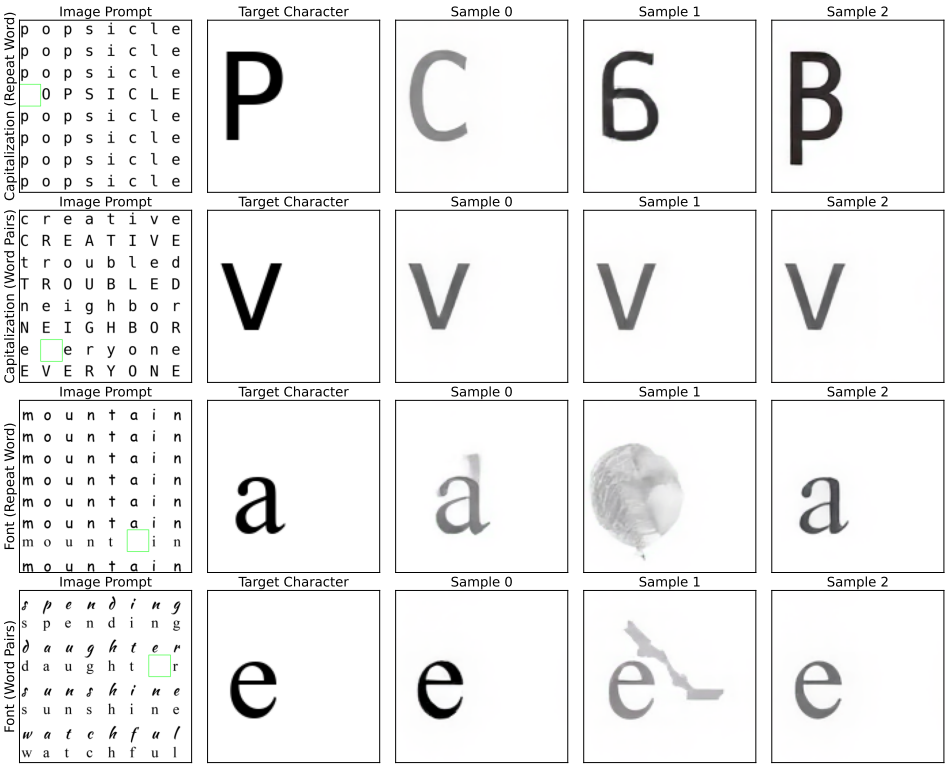

Figure 15: More examples of inpainting.

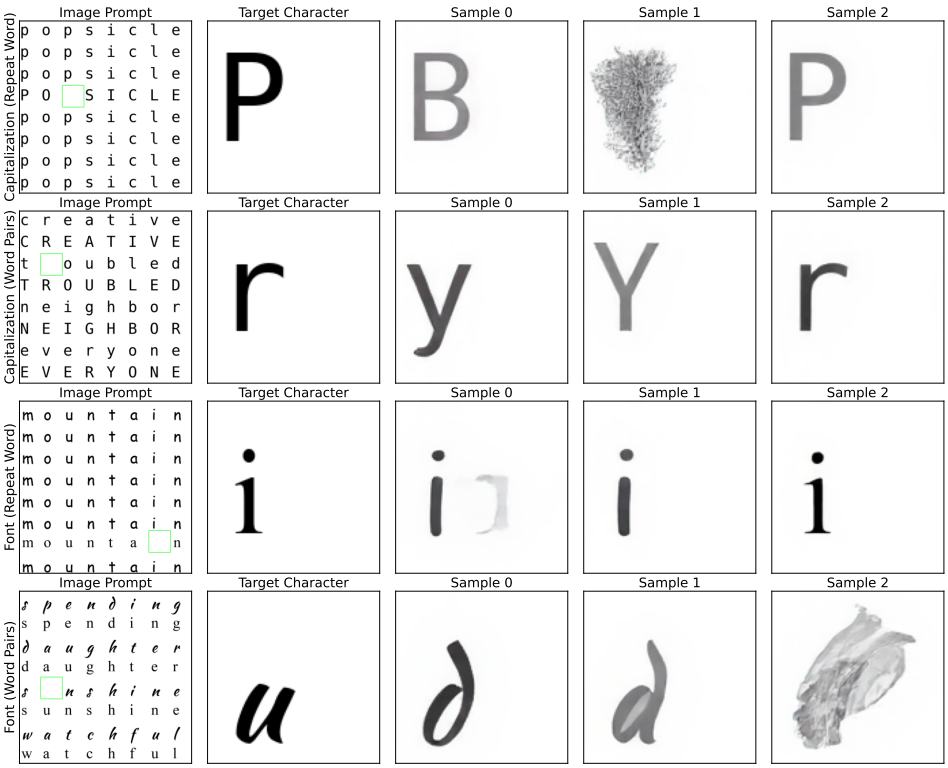

Figure 16: More examples of inpainting.