Vision-Code Transformer for Screenshot-to-HTML/CSS Generation

Davit Soselia
University of Maryland
dsoselia@cs.umd.edu

Khalid Saifullah
University of Maryland
khalids@cs.umd.edu

Tianyi Zhou
University of Maryland
zhou@umiacs.umd.edu

Abstract

Automated HTML/CSS code generation from screenshots is an important yet challenging problem with broad applications in website development and design. In this paper, we present a novel vision-code transformer approach that leverages an Encoder-Decoder architecture, for which two image encoders are compared: Vision Transformer (ViT) and Document Image Transformer (DiT). We propose an end-to-end pipeline that can generate high-quality code snippets directly from screenshots, streamlining the website creation process for developers. To train and evaluate our models, we created a synthetic dataset of 30,000 unique pairs of code and corresponding screenshots. We evaluate the performance of our approach using a combination of automated metrics such as MSE, BLEU, IoU, and a novel htmlBLEU score, where our models demonstrated strong performance. Specifically, our experiments show that DiT-GPT2 outperforms ViT-GPT2 in generating accurate and high-quality code snippets.

1. Introduction

Recent Language Models (LMs) have demonstrated a remarkable capability in generating coherent code. For example, Codex [6] and CodeRL [13] have shown promise in aiding software engineers in their daily work. On the other hand, attention-based models have achieved success in various vision tasks, such as image classification, segmentation, and annotation. Among them, some landmark works are Vision Transformer (ViT) [8], SWIN, and other architectures, with some models specifically targeting document-related tasks, such as the Document Image Transformer (DiT) [15].

In this paper, we take the first step towards reverse-engineering an image, i.e., generating a HTML/CSS code that can reproduce the image. By combining the strengths of both LLMs in code generation and the representations of Vision Transformer, we investigate the possibility of generating the markup code from the representations of the original image. Our contributions are twofold. First, we develop a synthetic dataset generation module to be used for front-end UI image and corresponding code generation. The module is designed to generate images with varying complexity and styles, as well as the corresponding markup code. This dataset is used for training and evaluating our proposed approach.

Secondly, we examine the performance of several encoder-decoder architectures for this task. Specifically, we employ GPT-2 [21] as the text decoder for code generation and compare the performance of ViT and DiT as image encoders. ViT is a widely utilized model trained on natural images for image recognition tasks. In contrast, DiT is specifically designed for document-related tasks, whose domain tends to be closer to that of our task. We explore the efficacy of these architectures in generating HTML/CSS code from webpage screenshots.

Our work for the first time establishes a robust baseline for generating markup code from images using vision-code transformers. Moreover, we develop a novel evaluation metric to aid in assessing the task. Our proposed approach holds potential applications in front-end web development, as it could offer a more efficient and automated method for generating markup code for web designers.

2. Related Works

Recent years have witnessed significant progress in both image-understanding and text-generation tasks, empowered by deep learning and the availability of massive datasets [5, 9, 11, 12, 20, 25]. Transformer models trained on large-scale data in a self-supervised manner have played a key role in these advancement [7, 26].

Code prediction and generation have received growing attention in the realm of text generation. While traditional NLP methods like N-grams and Probabilistic Context-Free Grammar (PCFG) have encountered challenges in code generation tasks [18], recent advancements in Transformer models have led to remarkable improvements. For instance, Codex [6] and its derivative tool Copilot, fine-tuned on publicly available code from GitHub, have achieved impressive performance on code generation tasks. InCoder [10] enables bidirectional context for code infilling

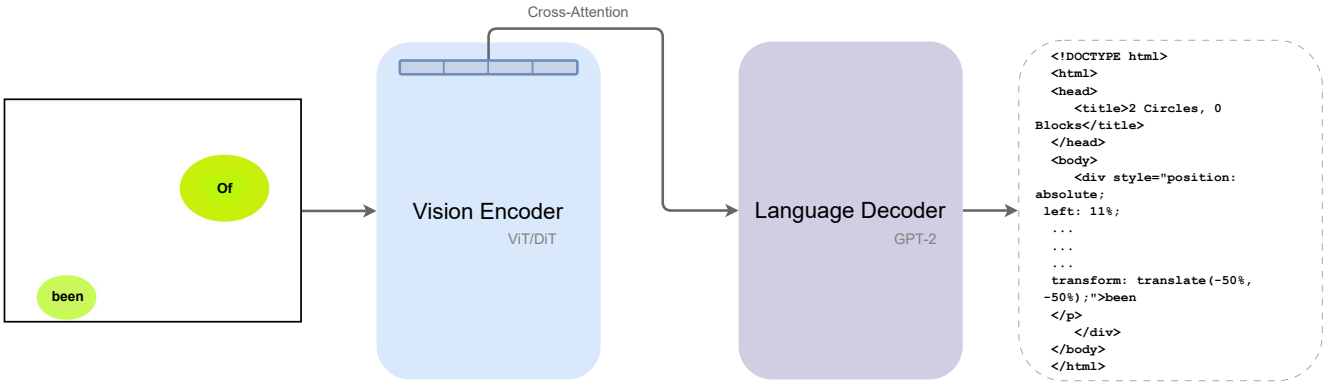

Figure 1. Vision-code Transformer proposed in this paper. It applies a vision transformer (ViT or DiT) to a webpage screenshot, whose output tokens are sent to an LLM (GPT-2) that generates the code that aims to reproduce the input screenshot.

by training on publicly available repositories where code regions have been randomly masked and moved to the end of each file. CodeGen [19] explores a multi-step paradigm for program synthesis, dividing a single program into multiple subproblems specified by multiple prompts. CodeRL [13] incorporates deep reinforcement learning with an error-predictor critic network that generates rewards by classifying the code.

In image-to-text generation, pretraining vision-language models on large-scale datasets has resulted in remarkable performance improvements [2, 17]. Cross-attention has been employed to effectively integrate information from both image and text modalities [3]. Recent works like BLiP [14], Git [27], and CoCA [28] have leveraged large-scale pre-training on visual-textual data followed by fine-tuning on target tasks and outperformed traditional methods.

Other models aiming to generate code from images include pix2code [4], which generates code based on context and GUI images, and Sketch2code [23], which attempts to generate code from wireframes using traditional computer vision and deep learning algorithm. The paper found the deep learning-based pipeline to perform better. While some works add components such as image style transfer, they rely on predefined classification for code generation.

3. Dataset

To create a dataset that pairs the actual code with its corresponding visual representation, we utilized a synthetic generation process. A dataset was generated by combining a small number of HTML elements, such as two types of Divs, a square, a circle, and a button element, with randomly chosen style attributes. This resulted in a diverse set of images that could be used for the training process.

It's important to note that for the sake of simplicity in generating the dataset, we opted to use inline CSS. Though it may not align with best practices, it significantly simplifies the generation process. All the synthetic dataset code is enclosed in standard HTML opening and closing tags, specifically:

Table 1. Element types, widths, and parameters used for the synthetic dataset generation. The number of elements in each code sample was chosen randomly between 1 and 6.

Properties	Rectangle	Ellipse	Button
Left (%)	0-80	0-80	0-80
Top (%)	0-80	0-80	0-80
Width (%)	10-30	10-30	10-30
Height (%)	10-30	10-30	10-30
Background	Uniform	Uniform	–
Text Length	1 Word	1 Word	1 Word
Occurrence	12/25	12/25	1/25

```
<!DOCTYPE html>
<html>
  <head>
    <title>{title}</title>
  </head>
  <body>
    {elements}
  </body>
</html>
```

We set the description of the elements present in the body as the title, for example, "2 Circles, 0 Blocks". For each element, a paragraph containing a number of words has been added, with the text sourced from Project Gutenberg [1]. The dataset generator can be used to create samples of varying complexity. A summary of the settings used for the work can be found in Table 1. Overall, the maximal input length of the adopted models acts as a ceiling to the length

of generated code.

An example of the generated element is below. Note that some of the parameters in Table 1 have been adjusted to fit the webpage, but the aesthetics of the generated elements have not been taken into account.

```
<div
  style="position: absolute; left: 11%;
      top: 79%; width: 20%; height:
     20%;
  background-color: #C6FC54; border-
     radius: 50%; text-align: center;">
  <p
    style="margin: 0; position:
       absolute; top: 50%; left: 50%;
    transform: translate(-50%, -50%);">
       been</p>
</div>
```

In our study, 30,000 code samples have been generated, with a split of 80:10:10 used for training, validation, and testing. For each code sample, we take a screenshot of its generated webpage as it looks when opened in a Chromium browser. Thereby, we collected a dataset of (image, code) pairs, facilitating the training and evaluation of our models.

It is worth noting that the traditional pipelines cannot directly address the task studied in this paper due to their different problem formulations and dataset formats. Specifically, they reduce the problem to classification between predefined building blocks while our approach focuses on free-form code generation. Hence, comparisons to them on our proposed dataset and task are infeasible. That being said, we create baselines for Transformer models on our dataset for HTML/CSS code generation that explores greater freedom in attributes and has no restrictions on color variety.

4. Proposed Vision-Code Transformer

Our model employs a Visual Transformer (ViT) [8] as the vision encoder. The ViT processes the input image by dividing it into patches and encoding them as a sequence of tokens, supplemented by a [CLS] token representing the image's global features. This computationally efficient approach, which has become widely adopted in recent methods such as [16], eliminates the need for pre-trained object detectors for visual feature extraction. Since ViT was usually pre-trained on natural images while the images in our dataset are mainly UI images, we further explored the DiT model [15], a transformer trained on document images (which has a smaller domain gap to UI images), as an alternative image encoder. We use GPT-2 [21], an autoregressive language model, as the image-grounded text decoder. This model integrates the ViT encoder's output tokens into its first-layer inputs via a cross-attention (CA) layer, positioned between the causal self-attention (CSA) layer and the feedforward network (FFN) of the text encoder. The input sequence length is set to 900 while each sequence begins with a [BOS] token and is terminated by a [EOS] token. By optimizing a cross-entropy loss function, we train the ViT encoder and GPT-2 decoder in an end-to-end manner, which maximizes the likelihood of the ground-truth code in an autoregressive way. This objective provides the model with the ability to generalize and effectively convert visual information into coherent HTML/CSS code.

5. Evaluation Metrics

Human annotators were presented with the original image and the webpage produced by the generated code and then asked to rate them on a scale of 100 in terms of structural and color similarity. For each model, we report the averaged score across all annotators and samples.

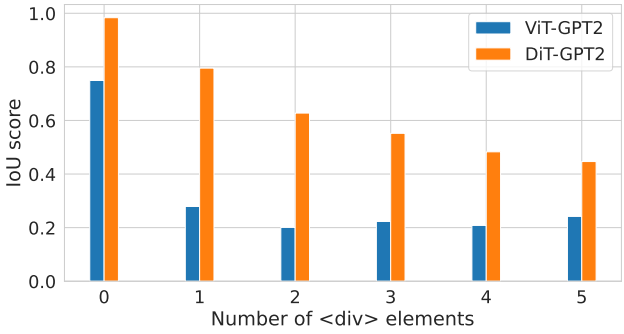

Figure 2. IoU vs. complexity (the number of div elements in the ground-truth code). IoU drops as more elements are added.

In addition to human evaluation, we employed two groups of metrics to assess the model performance: (1) code-based metrics, which compare the generated code against the original code producing the input screenshot; and (2) image-based metrics, which evaluate the screenshot of the generated images against the input.

We use two metrics for (2): mean squared error (MSE) between masks of both images and Intersection over Union (IoU) score. For (1), we employed BLEU score and a dataset-specific metric called Element Counts, where the presence of all elements in the generated code results in a score of 1, and misalignment yields a score of 0.

Since the BLEU score equally penalizes any differences between the two pieces of code, it is not an ideal metric for code evaluation. To avoid penalizing differences that do not lead to visual discrepancies, we develop a new metric, htmlBLEU, from CodeBLEU [22], as HTML code lacks data flow or syntactic abstract syntax tree (AST). htmlBLEU comprises four components: a basic BLEU score, a weighted BLEU score focusing on the most important keywords for HTML code, a Document Object Model DOM

Tree Matching between the corresponding HTML elements, and an attribute matching that aims to find elements with the same attributes. To evaluate htmlBLEU, we measure the Spearman's rank correlation coefficient [24] between htmlBLEU and the MSE between input/generated images (which is 0.764) and compare it with that between BLEU and the MSE (which is 0.329). The higher correlation of htmlBLEU demonstrates that it accurately reflects the visual similarity.

6. Experimental Results

We report the above metrics for different models in Table 2, in which DiT-based model significantly outperforms the ViT-based model across all metrics. Although ViT can accurately handle simpler images with a single element, it struggles to correctly capture > 1 types of elements present in the input image, as well as their positions and colors.

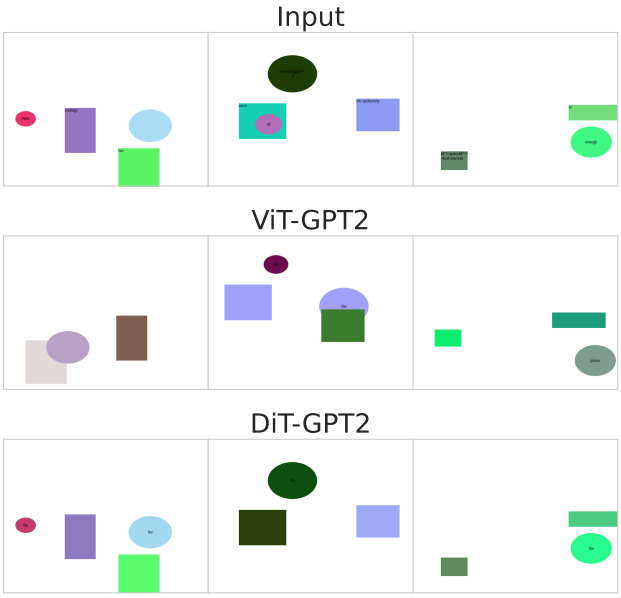

Figure 3. Graphics produced by the generated code. TOP-row: input screenshots; MIDDLE-row: screenshots of the webpages by ViT-GPT2 generated code; BOTTOM-row: screenshots of the webpages by DiT-GPT2 generated code.

This is consistent with the qualitative results of our study: The ViT-based model was found to frequently miss or misinterpret elements and had difficulty accurately predicting the hexadecimal values of the colors. Figure 3 show some examples: DiT model accurately identifies the types and locations of the elements, while ViT model struggles with these tasks.

As shown in Figure 2, both models' performance drops as the code samples get more complex. But DiT suffers from a much slower drop on the IoU curve than ViT when predicting more than one element. Another interesting observation is that the generated code does not necessarily have a high text similarity as the ground-truth code for the input screenshot. It can still produce visually similar webpage even if the textual similarity is low, which indicates a promising generalization capability of the models.

Table 2. Evaluation of code generation from screenshot

Model	ViT-GPT2	DiT-GPT2
Metrics		
BLEU ↑	0.65 ± 0.08	**0.74** ± 0.09
htmlBLEU ↑	0.62 ± 0.13	**0.69** ± 0.14
IoU ↑	0.31 ± 0.25	**0.64** ± 0.27
MSE ↓	19.63 ± 11.59	**12.25** ± 8.83
MSE (Single Channel) ↓	0.15 ± 0.09	**0.07** ± 0.06
Element Counts ↑	**0.97** ± 0.16	**0.97** ± 0.18
Human Evaluation (Normalized)		
Color Fidelity ↑	0.26 ± 0.26	0.61 ± 0.32
Structural Similarity ↑	0.44 ± 0.35	0.62 ± 0.35

7. Conclusion

This paper investigates how to build and train a vision-code transformer for reverse engineering a webpage screenshot and generating the HTML/CSS code that can reproduce the screenshot. We apply ViT or DiT as an image encoder and GPT-2 as a textual decoder that generates code from the ViT/DiT features of the input image. Unlike traditional pipelines, our models can be trained in an end-to-end manner for free-form code generation. Moreover, we collect a synthetic dataset to train and evaluate the proposed models and develop a novel htmlBLEU metric to evaluate the matching between the ground-truth code and the generated one. Our experimental results show that the DiT-GPT2 model outperforms ViT-GPT2 in terms of multiple metrics and human evaluation.

It is worth listing the limitations of this study. The synthetic dataset used, while containing variations in elements' size and location, may not be fully representative of real-world web pages. Additionally, the dataset does not adhere to all best practices for front-end development, and further work is required for practical deployment in real products. Furthermore, the current pipeline can only generate static text pages and is limited to handling small samples.

This study serves as a proof-of-concept in the field, demonstrating that Transformer architectures could be a viable end-to-end solution for this task. However, further research is necessary to extend the generated code's length, improve text snippet identification in the image, and explore more complex examples where the corresponding code may not be as straightforward.

References

[1] Project Gutenberg. https://www.gutenberg.org/. Accessed on: 2023-02-13. 2

[2] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering, 2017. 2

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022. 2

[4] Tony Beltramelli. pix2code: Generating code from a graphical user interface screenshot. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 1–6, 2018. 2

[5] 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, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey 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, 2020. 1

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 1

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 1

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. 1, 3

[9] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021. 1

[10] Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*, 2022. 1

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1

[12] Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks, 2013. 1

[13] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven CH Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *arXiv preprint arXiv:2207.01780*, 2022. 1, 2

[14] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022. 2

[15] Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. Dit: Self-supervised pre-training for document image transformer, 2022. 1, 3

[16] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers, 2021. 3

[17] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, 2019. 2

[18] Chris Maddison and Daniel Tarlow. Structured generative models of natural source code. In *International Conference on Machine Learning*, pages 649–657. PMLR, 2014. 1

[19] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022. 2

[20] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. 1

[21] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1, 3

[22] Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. Codebleu: a method for automatic evaluation of code synthesis. *arXiv preprint arXiv:2009.10297*, 2020. 3

[23] Alex Robinson. Sketch2code: Generating a website from a paper mockup, 2019. 2

[24] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. 4

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014. 1

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 1

[27] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language, 2022. 2

[28] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca, 2022. 2