StrokeNUWA—Tokenizing Strokes for Vector Graphic Synthesis

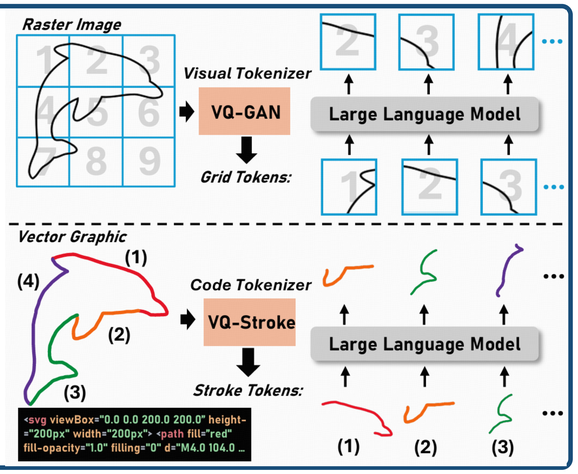
Zecheng Tang[*1,2] *Chenfei Wu*[*2] *Zekai Zhang*[2] *Minheng Ni*[2] *Shengming Yin*[2] *Yu Liu*[2] *Zhengyuan Yang*[3] *Lijuan Wang*[3] *Zicheng Liu*[3] *Juntao Li*[1] *Nan Duan*[2]

[1] *Soochow University, China* [2] *Microsoft Research Asia* [3] *Microsoft Azure AI*

Code/Project Homepage: *https://github.com/ProjectNUWA/StrokeNUWA*

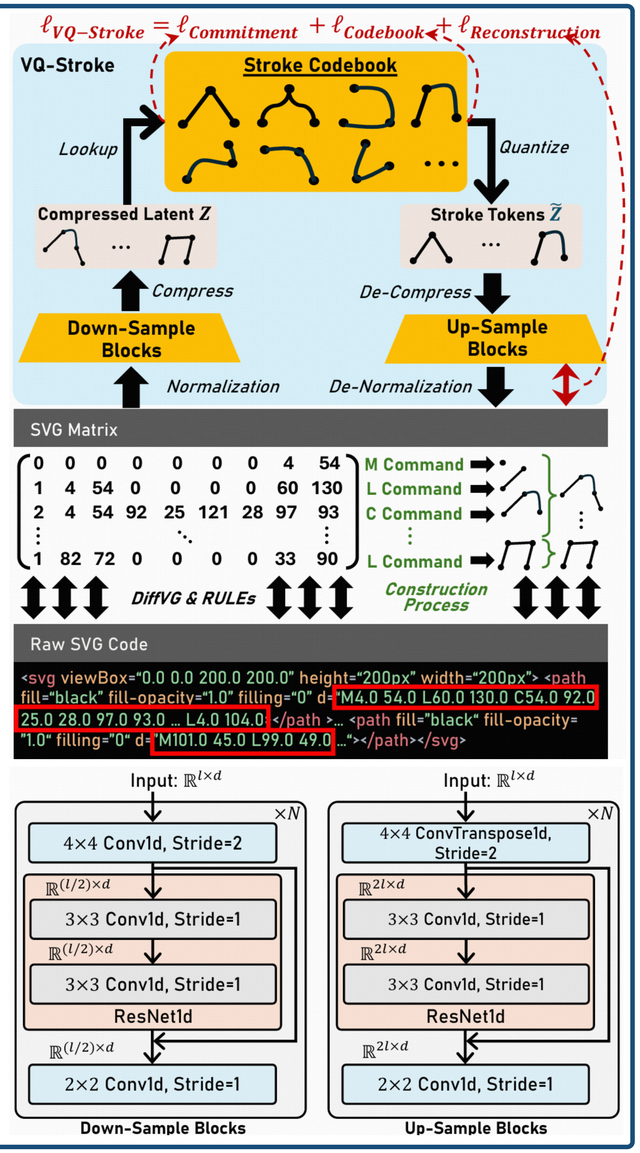
Abstract & Motivation

- This paper posits that an alternative representation of images, vector graphics, can effectively surmount this limitation by enabling a more natural and semantically coherent segmentation of the image information.
- Comparison between the visual representation of "grid" token and our proposed "stroke" token. Instead of tokenizing pixels from raster images, we explore a novel visual representation by tokenizing codes, from another image format—Scalable Vector Graphic (SVG). "Stroke" tokens have the following advantages: (1) inherently contain visual semantics, (2) naturally compatible with LLMs, and (3) highly compressed.
- StrokeNUWA achieves up to a 94× speedup in inference over the speed of prior methods with an exceptional SVG code compression ratio of 6.9%.



Design & Methodology

- **Top-Level Design**
  - **StrokeNUWA contains three core components:** 1) a Vector Quantized-Stroke (VQ-Stroke) for SVG compression; 2) an Encoder-Decoder-based LLM (EDM) for SVG generation; and 3) an SVG Fixer (SF) for post-processing.
  - **Training Pipeline:** Firstly, VQ-Stroke compresses each SVG into stroke tokens; secondly, EDM utilizes the stroke tokens produced from VQ-Stroke to generate SVG code; finally, the generated SVGs are fed to SF for post-processing, making sure they conform to the stringent syntactical rules of SVG code.
- **More Details about VQ-Stroke**
  - **VQ-Stroke** contains two main stages: 1) "Code to Matrix" stage that transforms SVG code into the matrix format suitable for EDM input; and 2) "Matrix to Token" stage that transforms the matrix data into stroke tokens.
  - **Code to Matrix** aims to decomposes all the paths within the SVG into distinct basic commands and combines their corresponding vectors into a matrix form.
  - **Matrix to (Stroke) Token** utilize a VQVAE-like module, which is shown in the right.
- **More Details about EDM and SF**
  - **EDM** is designed based on the T5 model architecture. We freeze the encoder part of EDM and train the decoder part to utilize the textual guided capability of T5 model.
  - **SF module** can post-process the generated SVGs. We introduce two strategies: Path Clipping (PC) and Path Interpolation (PI).
    - **PC** direct substitutes each SVG command's beginning point with the endpoint of adjacent SVG commands, making the SVGs more streamlined.
    - **PI** adds an extra command to force the previous command's endpoint to move to the beginning point of the next adjacent command, making the SVGs have more details.

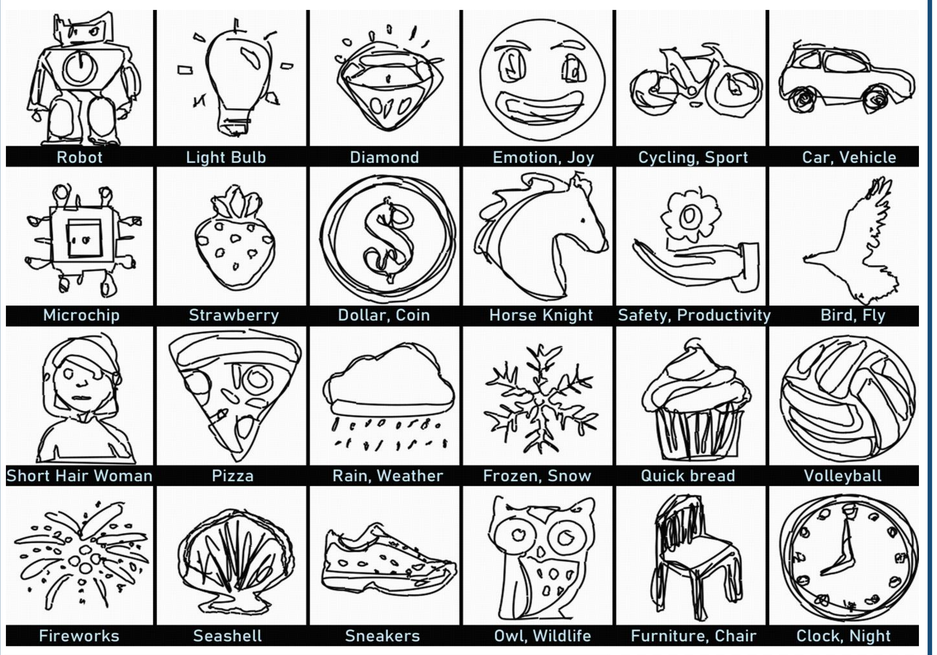


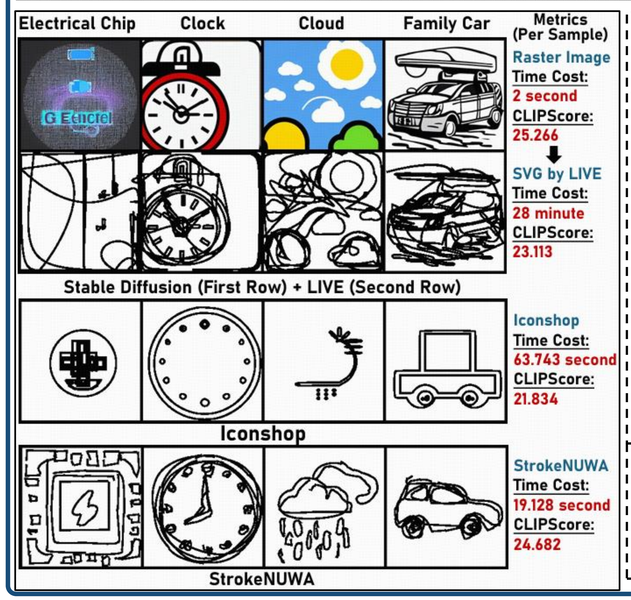
Experiments

Methods	Visual Performance			SVG Code Quality			Generation Speed (↓) (*per* SVG)
	FID (↓)	CLIPScore (↑)	HPS (↑)	Recall (↑) (Stoke Token)	EDIT (↓)	Optim / Pred Length (*Avg*)	
SD & LIVE	14.236	12.908	11.210	0.028	-	160 (32 Path)	≈ 28.0 min
VectorFusion	7.754	17.539	15.901	0.079	-	2,048 (128 Path)	≈ 30.0 min
Iconshop	17.828	8.402	8.234	0.114	24,792.476	993.244	≈ 63.743 sec
StrokeNUWA (PC)	6.607	17.852	16.134	**0.239**	**9,092.476**	271.420	≈ **19.128** sec
StrokeNUWA (PI)	**6.513**	**17.994**	**16.801**	0.207	12,249.091	271.420	≈ **19.128** sec

Partial experimental results, please refer to the paper for more results.

Generated Cases





StrokeNUWA is capable of generating graphics that more closely resemble the Golden SVG—evidenced by the lowest FID and the highest HPS. Besides, Stroke Token not only aligns closely with the Golden standard SVG but also markedly enhances the generation speed

Noting: We provide partial generated cases here. For more details, please refer to our paper.