Sketch2Diagram: 視覚的指示を入力とするダイアグラム生成

斉藤いつみ [1,2]　吉田遥音 [1]　坂口慶祐 [1,2]
[1] 東北大学　[2] 理化学研究所
{itsumi.saito, keisuke.sakaguchi}@tohoku.ac.jp

概要

スケッチ画像を理解してベクター形式のダイアグラムを生成するためのベンチマークデータセット SketikZ を提案する [1)]．SketikZ は，人手で作成したスケッチ画像と TikZ 形式のダイアグラムがペアになった初めてのデータセットである．さらに，画像を理解して TikZ 形式のダイアグラムを生成可能なマルチモーダルモデル ImgTikZ を提案する．ImgTikZ は，コード生成に特化した大規模言語モデルと画像エンコーダを活用したモデルであり，実験によって 7B 規模のモデルサイズながら GPT-4o に匹敵するダイアグラム生成能力を有することを確認した．また，スケッチ作成のツールによって画像認識の難易度が大きく変わることを確認した．

1　はじめに

ダイアグラムは複雑な情報を視覚的にわかりやすく伝えることができる表現方法であり，学術・教育・ビジネスなどで広く利用される．ダイアグラムの意味を AI が正しく理解すると同時に，複雑な情報をダイアグラムの形式で表現できることは高度なコミュニケーションが可能な AI の実現に向けて重要である．ダイアグラムに対する質問応答 [1, 2]，説明生成 [3, 4, 5] など，ダイアグラムを入力として自然言語を生成するタスクで多くの研究が行われてきた一方，ダイアグラムを生成する研究は非常に少ない．自然言語による指示を入力としてダイアグラムを生成する研究 [6, 7, 8] は一部行われているが，複雑なダイアグラムの生成において全ての情報を言語で表現することは難しくコストも高い．

本研究では，スケッチという人間にとって直感的に表現可能な視覚的指示を入力としてダイアグラムを生成するタスクに取り組む（図 1）．ラフな手書きスケッチ画像から整ったダイアグラムを自動生成できれば，簡易な指示で低コストに高品質なダイアグラムを生成することが可能となる．また，本研究ではダイアグラムの表現形式として LaTeX の描画パッケージである TikZ コード [2)] を用いる．学術的な図などの生成において，TikZ や Graphviz といった描画用のコードををコンパイルすることで任意の解像度の高品質なダイアグラム画像を生成できる．また，中間表現としてコードというテキスト形式の出力を用いることで，大規模言語モデル（LLM）の生成能力を直接活用することができる [6]．自然画像生成で一般的である text-to-image モデル [9] のようにラスタ画像を直接生成する方法に比べ，コードを利用する方法は高解像度の画像の生成が行いやすい [6]，要素間の関係性をより正確に扱える [10]，などダイアグラム生成に適した様々なメリットがある．

スケッチによる視覚的指示を入力としたダイアグラム生成は有望な方向性であるものの，学習・評価に利用可能なベンチマークデータセットが整備されていない点が大きな課題である．本研究では，スケッチ画像から TikZ 形式のダイアグラムを生成するためのベンチマークデータセット SketikZ を初めて構築した．このデータは，多様な環境で作成した 3231 件のリアルなスケッチ画像と TikZ 形式のダイアグラムデータからなる．さらに，コード特化型 LLM と画像エンコーダを活用し，画像を理解して TikZ コードを生成可能なマルチモーダルモデル ImgTikZ を構築した．学習用の大規模データ拡張や，複数候補生成と最適候補選択を組み合わせる推論方法により，7B サイズのモデルながら GPT-4o に匹敵するダイアグラム生成能力を達成した．

2　問題設定と SkeTikZ データ構築

2.1　問題設定

タスク定義　スケッチ画像 I_s とテキストによる指示 X を入力とし，TikZ コード Y を生成する．生成した TikZ コード Y をコンパイルしてダイアグラ

1） https://sketikz.github.io/

2） https://texwiki.texjp.org/TikZ

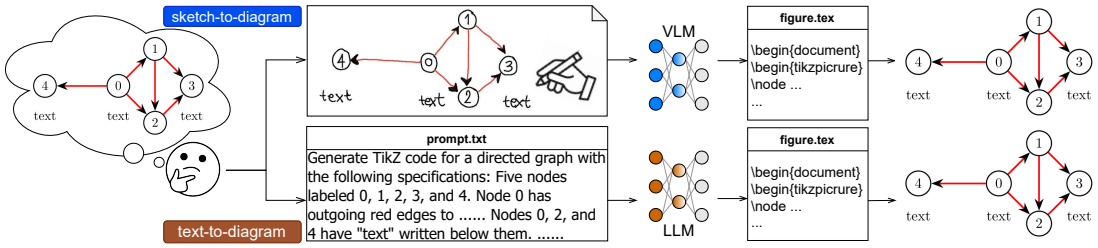


図 1　本研究で定義する *sketch-to-diagram* タスクの概要. ユーザは生成したいダイアグラムをスケッチとして手書きする. モデル（例: VLM）はスケッチ画像 I_s と自然言語によるインストラクション X を受け取り，ダイアグラムを描画可能なコード Y を生成する．コード Y をコンパイルしてレンダリングされたダイアグラム画像 I_d を生成する.

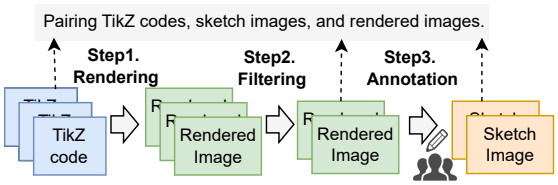


図 2　SkeTikZ データ構築手順.

ム画像 I_d を生成する.

既存研究との関係　画像を理解してコードを生成する研究として，画像のスクリーンショットから HTML，SVG，LaTeX 等のコードを生成する研究 [11, 12, 13, 14, 15, 16] や，同時期に発表された類似研究として，画像を理解してダイアグラムを生成する研究 [17] が存在するが，多様な環境でのスケッチ画像とコード形式のダイアグラムがペアになったベンチマークデータセットは SkeTikZ のみである．さらに，本研究で提案するデータ拡張方式や推論方式もこれらの既存研究では行われていない.

2.2　SkeTikZ データ構築

図 2 に示したプロセスを通して，スケッチ画像 I_s，参照 TikZ コード Y^r，レンダリングされた参照ダイアグラム画像 I_d^r の対応データを作成した.

Step1: 参照画像のレンダリング　既存研究 [6] の TikZ コードをコンパイルし，参照ダイアグラム画像 I_d^r をレンダリングした.

Step2: ダイアグラム画像の分類とフィルタリング　参照画像 I_d^r をダイアグラム画像分類モデルを用いて分類し，対象カテゴリ (Tree, Graph, Archtechture diagram, Neural network, Venn diagram) のダイアグラムのみを抽出した（詳細は付録 A 参照）.

Step3: スケッチ画像のアノテーション　Step2 で抽出した画像 I_d^r をアノテータに提示し，手書きのスケッチ画像 I_s を作成した．リアルかつ多様なスケッチ画像作成のため，紙，ホワイトボード，タブレットの 3 つのツールを用いて，スケッチを行った後に写真やキャプチャ画像として保存した．複雑すぎる図や色の塗りつぶしは対象外とした.

3　提案手法

3.1　モデル構造

ImgTikZ は LLaVA1.5 [18] を踏襲し LLM, 画像エンコーダ, 2 層の MLP アダプタからなる（図 3 (a)）. LLM はコード特化の DeepSeek coder (6.7B) [19]，画像エンコーダは SigLIP [20] を利用した．学習は二段階で実施した．Stage1 の学習ではアダプタのみを学習し，Stage2 の学習ではアダプタと LLM に追加した LoRA パラメータ [21] のみを学習した．いずれも，その他のパラメータは固定した.

3.2　学習データ

Stage1　LLaVA1.5 [18] の LLaVA-Pretrain データに加え，図表の理解や OCR 能力を促進させるため，arXiv bulk データの PDF データから，図表画像とキャプションのペアデータ 1.1M 件，図表画像と画像中の OCR テキストのペアデータ 1.2M 件を収集し，図表画像を入力してキャプションや OCR テキストを生成するタスクを実施した.

Stage2　SketikZ の学習データに加え，arXiv bulk データの LaTeX ソースファイルから TikZ コードを収集・コンパイルし，レンダリング画像とコードのペアデータ（RenderTikZ）を 155K 件作成した．ダイアグラムのバリエーションを増やすため，収集した TikZ コードを GPT3.5 Turbo を用いて改変し，データ拡張した．改変コードからレンダリングした画像とコードのペアデータ（**AugTikZ**）を 556K 件作成した．よりスケッチ画像に近いデータを生成するため，レンダリング画像に背景画像合成や回転・歪み・コントラスト等の画像ノイズを加えるデータ拡張を行い，ノイズ画像とコードのペアデータ

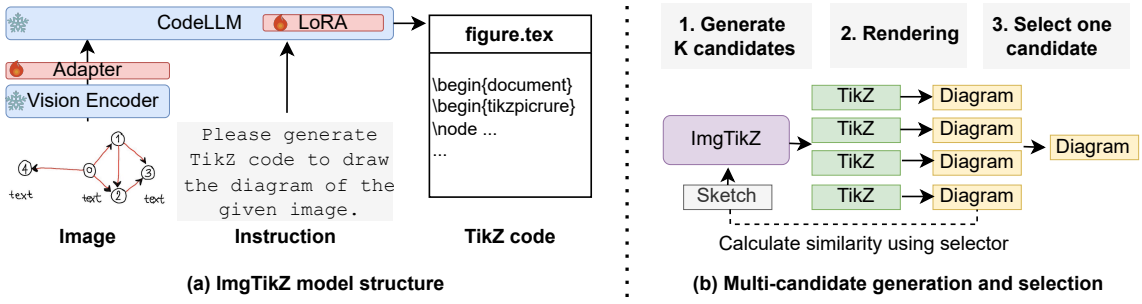


図 3　提案モデル ImgTikZ の概要 (a) と並列生成・最適候補選択を行う推論 (b) の概要.

（**ImgAugTikZ**）を 714K 件作成した（付録 B 参照）. スケッチ画像からのコード生成に加え，レンダリング画像や，ノイズを加えたレンダリング画像からのコード生成タスクを大量データで学習した.

3.3　推論

逐次生成 (ImgTikZ-IG)　生成したコードのコンパイルが失敗した場合，コンパイルが成功するまで最大 M 回生成を行うシンプルな方法.

並列生成と最適候補選択 (ImgTikZ-MCG)　図 3 (b) に示すように，複数候補 K 個を並列に生成した後各候補に対してスコアリングを行い，スコアが高い候補を一つ選択する方法. 入力したスケッチ画像 I_s と生成されたダイアグラム画像 I_d の類似度を計算し，最も類似度が高い候補を選択する. 類似度計算は画像エンコーダから得られる隠れ層のコサイン類似度を用いて計算する. ダイアグラム画像の類似度を適切に計算するためのモデルとして，SigLIP に一層の線形層を追加し，ダイアグラムデータを用いて対照学習を行うことでダイアグラムに適応させたモデル D-SigLIP を構築した（詳細は付録 C 参照）.

4　実験

4.1　実験設定

ベースラインモデルとして，クローズドな最先端モデル GPT-4o, GPT-4o Mini, Claude3.5 Sonnet とオープンソースの最先端モデルである LLaVA-Next[22] を用いた. ベースラインモデルは逐次生成（$M = 5$）によって生成を行った. 並列生成の候補数 $K = 20$ とした. ImgTikZ の学習設定の詳細は付録 D 参照.

4.2　自動評価指標

コンパイル成功率 (*CSR*)　生成したコード Y がコンパイルに成功した割合を表す.

画像類似度 (ImageSim)　生成された画像 I_d と参照画像 I_d^r をそれぞれ画像エンコーダに入力して得られたベクトルのコサイン類似度とする. ダイアグラムの類似度を適切に計算するために，画像エンコーダとして D-SigLIP (3.3 節を参照) を用いる.

文字類似度 (CharSim)　生成されたダイアグラム画像 I_d 中の文字と参照画像 I_d^r 中の文字の類似度を表す. I_d と I_d^r に対して OCR を行い，抽出した文字列の Rouge-1 スコアを計測した.

4.3　主観評価指標

生成された画像 I_d と参照画像 I_d^r がどの程度一致しているか (Alignment) と生成された画像 I_d がダイアグラムとして自然なレイアウトになっているかどうか (Quality) の 2 つの観点について，Amazon Mechanical Turk を用いて 1〜5 の 5 段階評価で評価を行った（付録 E 参照）. 各生成画像について 5 名のアノテータが評価を行い，最大と最小を除く 3 つの評価値を平均して評価値を算出した.

4.4　実験結果

モデルはコンパイル可能なコードを生成できるか？　表 1 の CSR の結果から，クローズドな最先端モデルでもコンパイル成功率は 0.4 から 0.5 程度にとどまった. 一方，提案手法は 0.8 近くまでコンパイル成功率を大きく向上できた. コード特化 LLM の活用や関連データの収集・データ拡張がコンパイル成功率の向上に寄与したと考える.

モデルは参照画像に近いダイアグラムを生成できるか？　表 1 の ImageSim 及び Alignment に自動評価と主観評価での画像類似度を示した. 自動評価では ImgTikZ-MCG，主観評価では，Claude3.5 が最も高い値を示した. ImgTikZ-MCG は主観評価において GPT-4o に匹敵し，Claude3.5 に迫るスコアを示した. このことから，並列生成と最適候補選択の推論

表 1　自動評価 (Automatic) と主観評価 (subjective) の結果．自動評価は 0-1 の値，主観評価は 1-5 の値をとり大きい方が良いスコアとなる．上段はクローズドモデル，下段は 7〜8B サイズのモデル．ImgTikZ-IG と MCG が提案手法．

	Automatic			Subjective	
Model	**ImageSim**	**CharSim**	**CSR**	**Alignment**	**Quality**
GPT-4o	0.695	0.611	0.479	3.00	3.20
GPT-4o-mini	0.595	0.514	0.376	2.39	2.71
Claude 3.5	0.753	**0.671**	0.544	**3.32**	**3.54**
LLaVA-Next	0.315	0.206	0.350	1.43	1.93
ImgTikZ-IG	0.734	0.503	0.767	2.78	2.92
ImgTikZ-MCG	**0.821**	0.594	**0.799**	3.13	3.30

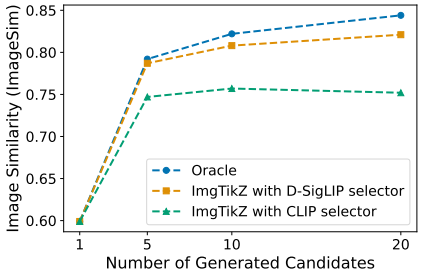


図 4　ImgTikZ-MCG において候補数を増やすことによる画像類似度（ImageSim）の変化．

戦略が効果的であったと言える．いずれのスコアでも，LLaVA-Next は非常に低い値を示しており，さらにモデル全体を通して平均的なスコアは 3 点台前半にとどまっていることから，今回のタスクは非常にチャレンジングなタスクであると言える．

モデルはダイアグラム中の文字を正しく生成できるか？　表 1 の CharSim に文字類似度を示した．Claude3.5，GPT-4o，ImgTikZ-MCG の順に精度が高く，ImgTikZ-MCG は Claude3.5 に比べて画像中の文字の認識・生成能力が劣っていることが示唆される．高解像度の画像を扱えるモデルを利用するなどの方法で精度向上が期待できる．

並列生成と最適候補選択を行うことで性能は向上するか？　図 4 に，ImgTikZ-MCG で生成する候補数 K を増やした時の画像類似度の変化を示した．生成候補数が 5 までは大きく性能が向上しており，その後も緩やかに向上した．候補選択器として CLIP を用いたときには性能の上がり幅が限定的であることから，ダイアグラムに適応したモデルで選択を行うことが重要であると言える．

データ拡張の効果はどの程度か？　表 2 の結果から，ImgAugTikZ, AugTikZ のいずれのデータを抜いた場合も大きく精度が低下した．コードの改変に基づくデータ拡張，画像のノイズ付与に基づくデータ拡張のいずれも大きな効果があったと言える．

表 2　データ拡張の効果．(a) は ImgAugTikZ を表し，(b) は AugTikZ を表す．データ拡張詳細は 3.2 節参照．

Model	**ImageSim**	**CharSim**	**CSR**
ImgTikZ-IG	**0.734**	**0.502**	**0.767**
w/o (a)	0.668	0.457	0.635
w/o (a) and (b)	0.601	0.439	0.541

表 3　ImgTikZ-IG を用いた場合のツールごとのスケッチ画像入力とレンダリング画像入力の性能差．上段が画像類似度（ImageSim），下段が文字類似度（CharSim）．

	Tool		
Metric	**Paper**	**Whiteboard**	**Tablet**
Rendered Image	0.793	0.796	0.754
Sketch Image	0.735	0.716	0.740
Performance Gap	**-7.31%**	**-10.1%**	**-1.90%**
Rendered Image	0.587	0.627	0.581
Sketch Image	0.502	0.451	0.570
Performance Gap	**-14.5%**	**-28.1%**	**-1.89%**

スケッチのツールごとに画像認識の難易度は異なるか？　スケッチ画像を作成したツールごとにレンダリング画像を入力する設定（Rentered Image）とスケッチ画像を入力する設定（Sketch Image）での画像類似度と文字類似度の性能差をツールごとに評価した．レンダリング画像からの性能低下が大きいほど，スケッチ画像の認識が難しいことを表す．表 3 の結果から，タブレットの場合が最も性能差が小さく 2%程度の性能低下に抑えられている．提案手法によりタブレットで書かれたスケッチのノイズにはある程度対応できている一方，紙やホワイトボードに書かれたスケッチ画像では 7%–28%程度の性能低下が観測された．特にホワイトボードで性能低下が大きく，これらの環境における背景や照明，手書きによる多様なノイズに対して，さらなる改善の余地があることを示唆している．

5　おわりに

スケッチ画像から TikZ 形式のダイアグラムを生成するためのベンチマークデータ SkeTikZ と，画像を理解して TikZ コードを生成可能なマルチモーダルモデル ImgTikZ を提案した．実験から，データ拡張や推論手法の効果を確認し，今後の有望な方向性を示した．簡易な指示に基づく高品質なダイアグラムの自動生成は学術・産業などに広く貢献できる．

謝辞

本研究成果の一部は，データ活用社会創成プラットフォーム mdx および東京科学大学のスーパーコンピュータ TSUBAME4.0 を利用して得られたものです．本研究は JSPS 科研費 JP21K21343，JP24K20829 の助成を受けたものです．データセット SketikZ は，株式会社バオバブ[3]様に委託して作成したものです．深く感謝いたします．

参考文献

[1] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In **Computer Vision – ECCV 2016**, pp. 235–251. Springer International Publishing, 2016.

[2] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. 2024.

[3] Ting-Yao Hsu, C Lee Giles, and Ting-Hao 'kenneth' Huang. SciCap: Generating captions for scientific figures. **arXiv [cs.CL]**, 2021.

[4] Anwen Hu, Yaya Shi, Haiyang Xu, Jiabo Ye, Qinghao Ye, Ming Yan, Chenliang Li, Qi Qian, Ji Zhang, and Fei Huang. mplug-paperowl: Scientific diagram analysis with the multimodal large language model. **arXiv preprint arXiv:2311.18248**, 2023.

[5] Shreyanshu Bhushan, Eun-Soo Jung, and Minho Lee. Unveiling the power of integration: Block diagram summarization through local-global fusion. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, **Findings of the Association for Computational Linguistics ACL 2024**, pp. 13837–13856. Association for Computational Linguistics, 2024.

[6] Jonas Belouadi, Anne Lauscher, and Steffen Eger. AutomaTikZ: Text-guided synthesis of scientific vector graphics with TikZ. **arXiv [cs.CL]**, 2023.

[7] Abhay Zala, Han Lin, Jaemin Cho, and Mohit Bansal. Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. In **COLM**, 2024.

[8] Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. VGBench: Evaluating large language models on vector graphics understanding and generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing**, pp. 3647–3659, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

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

[10] Jingxuan Wei, Cheng Tan, Qi Chen, Gaowei Wu, Siyuan Li, Zhangyang Gao, Linzhuang Sun, Bihui Yu, and Ruifeng Guo. From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing, 2024.

[11] Davit Soselia, Khalid Saifullah, and Tianyi Zhou. Learning ui-to-code reverse generator using visual critic without rendering, 2023.

[12] Yi Gui, Zhen Li, Yao Wan, Yemin Shi, Hongyu Zhang, Yi Su, Shaoling Dong, Xing Zhou, and Wenbin Jiang. Vision2ui: A real-world dataset with layout for code generation from ui designs. **ArXiv**, Vol. abs/2404.06369, , 2024.

[13] Chenglei Si, Yanzhe Zhang, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2code: How far are we from automating front-end engineering?, 2024.

[14] Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images. **arXiv preprint arXiv:2312.11556**, 2023.

[15] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-markup generation with coarse-to-fine attention. In **International Conference on Machine Learning**, 2016.

[16] Philippe Gervais, Asya Fadeeva, and Andrii Maksai. Mathwriting: A dataset for handwritten mathematical expression recognition. **ArXiv**, Vol. abs/2404.10690, , 2024.

[17] Jonas Belouadi, Simone Paolo Ponzetto, and Steffen Eger. DeTikZify: Synthesizing graphics programs for scientific figures and sketches with TikZ. In **The Thirty-eighth Annual Conference on Neural Information Processing Systems**, 2024.

[18] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**, pp. 26296–26306, June 2024.

[19] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. **ArXiv**, Vol. abs/2401.14196, , 2024.

[20] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid Loss for Language Image Pre-Training . In **2023 IEEE/CVF International Conference on Computer Vision (ICCV)**, pp. 11941–11952, Los Alamitos, CA, USA, October 2023. IEEE Computer Society.

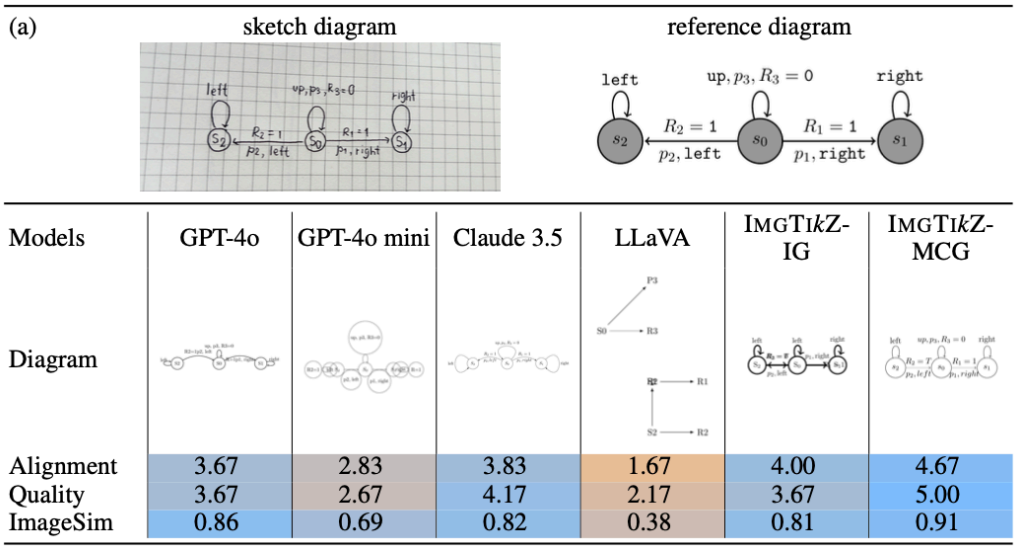
[21] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In **International Conference on Learning Representations**, 2022.

[22] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024.

[23] Zeba Karishma, Shaurya Rohatgi, Kavya Shrinivas Puranik, Jian Wu, and C. Lee Giles. Acl-fig: A dataset for scientific figure classification, 2023.

[24] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. **arXiv [cs.LG]**, 2020.

3） https://baobab-trees.com/



Models	GPT-4o	GPT-4o mini	Claude 3.5	LLaVA	ImgTikZ-IG	ImgTikZ-MCG
Diagram						
Alignment	3.67	2.83	3.83	1.67	4.00	4.67
Quality	3.67	2.67	4.17	2.17	3.67	5.00
ImageSim	0.86	0.69	0.82	0.38	0.81	0.91

図 5　モデルによるダイアグラム生成例.

A　ダイアグラム画像分類モデル

ダイアグラム画像分類モデルはダイアグラム画像に 19 種類のカテゴリラベルデータを付与している ACL-fig [23] データセットを用いて構築した．本研究では，スケッチに適したダイアグラムとしてグラフなど数値情報をベースにするものではなく，幾何的な図形を用いて描画するものを対象とするため，Tree, Graph, Archtechture diagram, Neural network, Venn diagram の 5 カテゴリを対象とした．事前学習モデルとして Google が提供している `vit-large-patch16-224-in21k`[4] を利用し，A100 GPU 1 枚で fine-tuning を実施した．ACL-fig の評価データセットにおける分類精度は 0.886 であった．

B　学習データ拡張方法

AugTikZ　GPT3.5 Turbo を利用して TikZ コードを改変した．具体的には，1) コンパイルに失敗した TikZ コードの修正，2) 元の TikZ コードと異なるダイアグラムの生成の 2 つを対象として，それぞれ次のようなプロンプトを与えてデータ拡張を実施した．“Please modify the given LaTeX source code to make it compilable.”，“Please generate TikZ source code that modifies parts of the following code to create a different diagram.”

ImgAugTikZ　レンダリング画像に比べて，手書きスケッチ画像は背景や照明，手書き線の歪みなどのノイズが大きい．手書きスケッチ画像に近いノイズを生成するため，ノートの背景画像の合成，imgaug[5] を用いた回転・歪みのノイズ生成やコントラストの調整，ホワイトバランスの調整[6] を行った．

C　D-SigLIP

SigLIP (`google/siglip-so400m-patch14-384`) に一層の線形層を追加し，SigLIP のパラメータは固定した上で線型層のみを対照学習によって学習した．学習データは RenderTikZ と AugTikZ を用いた．学習時に on-the-fly で画像にノイズを 2 回与え，ノイズを与えた同じ画像同士の類似度がそれ以外の画像よりも近くなるように学習を行った [24].

D　ImgTikZ の学習設定

LoRA のハイパーパラメータは，$r = 128$, $\alpha = 256$ とした．Stage1 の学習はバッチサイズ 256 で 6,000 ステップ，Stage2 の学習はバッチサイズ 128 で 1 エポック実施した．いずれも，A100GPU を 8 枚利用して学習を行った．

E　主観評価

Alignment　評価基準を次のように定義した．1: “The elements of the diagram in the generated image and the hand-drawn image do not match at all.” 5: “The elements of the diagram in the generated image and the hand-drawn image match almost perfectly.” とし，2,3,4 はそれぞれ “match approximately 25%, 50%, 75%"とした．

Quality　評価基準を次のように定義した．1: “Almost complete overlap of text or shapes, making the diagram unreadable.”, 2: “Significant overlap of text or shapes, and the arrangement of elements is unnatural. ”, 3: “Significant overlap of text or shapes, making some elements unreadable, or some elements are arranged unnaturally. ”, 4: “Some overlap of text or shapes, but the arrangement of elements is neat. ”, 5: “No overlap of text or shapes, and the arrangement of elements is as neat as a human-created diagram”

4) https://huggingface.co/google/vit-large-patch16-224-in21k
5) https://imgaug.readthedocs.io/en/latest/
6) https://github.com/mahmoudnafifi/WB_color_augmenter