Text2LaTeX: Unveiling the Potential of Language Models in LaTeX Synthesis

Jiayu Wang
jwang2782@wisc.edu

Abstract

LaTeX has been one of the de facto standards for the communication and publication of scientific documents. Generating LaTeX code that follows users' intentions is an important task in the digital age, which requires the model to understand user inputs presented in various formats and correctly generate grammatically correct code that can be rendered. Traditional rule-based methods, despite their high accuracy, often limit input to a specific format to generate valid LaTeX code, which substantially reduces the model's applicability. In reality, the same LaTeX expression might be represented in multiple formats such as images or textual descriptions. In this work, we investigate LaTeX program synthesis with foundation models that allow flexible inputs represented in texts or images. We propose to use a tree-based approach to generate LaTeX expressions with various diversity and scale. We observe that larger pre-trained models leads to improved performance. With an appropriate model, the prediction accuracy improves with an increase in both the size and complexity of the training set. Specifically, training on just 2500 samples (with a depth of 3) yields promising results, achieving 99.0% accuracy on test sets with expressions of depth-2, compared to 88.5% accuracy when training with the same sample size but a depth-2 input. This suggests the effectiveness of learning-based LaTeX code generation.

1 Introduction

LaTeX is a widely recognized standard for the communication and publication of scientific documents. Synthesizing LaTeX code that aligns with users' intentions is a critical task in today's digital landscape. In reality, as shown in Fig.1, the same LaTeX expression might be represented in multiple formats such as images, textual descriptions, reverse polish expressions, etc. However, traditional rule-based methods, despite their high accuracy, often limit input to a specific format for LaTeX code translation, which substantially reduces the model's applicability. Therefore, it is important to have a model capable of understanding user inputs in various formats and generating both grammatically correct and renderable code.

When the inputs are images, the task is closely related to Optical Character Recognition (OCR), which enables models to decipher and interpret the world in the same way human eyes do. Conventionally, OCR aims to convert images of scanned or handwritten text to digital texts. This process typically involves the detection of text contents in images and the subsequent transcription into machine-readable characters. Compared to normal texts, mathematical expressions involve unique symbols beyond the normal alphabet and often have spatial structures (superscripts, fractions, integration, etc.), requiring the OCR system to understand and maintain proper formatting (see Fig. 1). Therefore, it is more challenging to convert complex mathematical expressions from an image into LaTeX code effectively.

Multiple rule-based and learning-based methods have been proposed to tackle the challenge. Suzuki et al. (2003) proposed rule-based structural analysis methods and syntactic parsers to convert math formulas to markup languages. Due to the complexity of mathematical expressions, machine learning methods have also been used to interpret the structure and symbolism, which are trained on mathematical notations and could utilize context and formatting in ways that rule-based methods do not. With the rise of deep learning, prior works demonstrated improved performance by replacing handcrafted rules with representations learned by deep neural networks (Deng et al., 2017; Zhang et al., 2019; Wang et al., 2019; Wang and Liu, 2021). However, few have utilized modern large-scale pre-trained vision-language (foundation) models.

In the last three years, we have witnessed the rise of foundation models. These models are often

trained on web-scale data by self-supervision and can be adapted to various kinds of downstream tasks (Bommasani et al., 2021). This has also brought a paradigm shift to the machine learning community: instead of training from scratch, we choose to fine-tune pre-trained models. Foundation models also provide the possibility for program synthesis where the model can take in various input formats and produce the same desired output. A natural question arises:

> ***Q:*** To what extent can vision-language models understand flexible input formats and generate desired LaTeX expressions?

2 Overview and Problem Statement

In this project, we explore the LaTeX generation task with various input formats. We generate and construct datasets by leveraging binary trees as there are no open-source datasets that can be converted to fit in this project and in this way we can generate valid, easily scalable and diverse datasets (Section 3.1). Given the image or textual descriptions of a math expression, the goal is to convert the expression into a valid LaTeX expression that can be successfully rendered. A concrete example is shown in Figure 1. LaTeX generation is fundamental in document intelligence and yet underexplored in the open-source community. A central goal of this work is to tackle the challenge by exploiting recent advancements in code-generation models (Nijkamp et al., 2022) and vision-language models. We propose two variants of models for LaTeX code generation: 1) a Text2LaTeX model to handle text-only inputs; 2) an Image2LaTeX model to take in both image and text inputs (Section 3.2). In particular, the Text2LaTeX model is trained by the next token prediction objective (Eq. 1) while the Image2LaTeX model is trained via visual instruction tuning (Eq. 3). We conduct extensive experiments and share results and insights in Section 4.2.

3 Method

In the absence of suitable open-source datasets for our specified settings, we have developed custom datasets. This section details the dataset creation process and introduces the LaTeX code training pipeline.

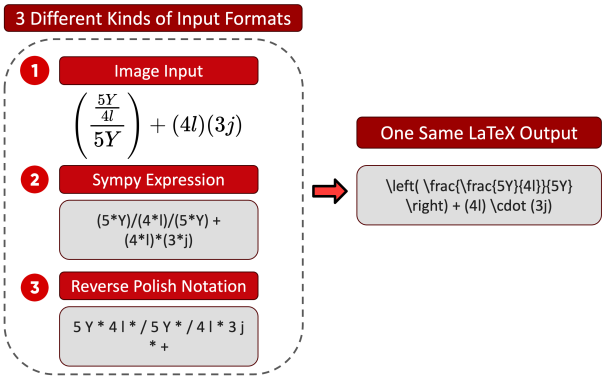

Figure 1: Illustration of the LaTeX generation task. **Left (input)**: three different kinds of input formats for the same math expression. **Right (output)**: the desired LaTeX expression. The model can take different input variants and output the same LaTeX expression.

3.1 Tree-based Dataset Generation

To construct large-scale (text, image (optional), LaTeX expressions) datasets, the first step is to generate valid and diverse math expressions. we propose to utilize a binary tree to generate math expressions, where leaf nodes correspond to symbols and numbers, while non-leaf nodes correspond to operators such as parenthesis and arithmetic operators (see Figure 2). Tree-based generation guarantees the validity of expressions and is easily scalable and diverse as one can freely adjust various parameters such as the depths of trees, the number of operators and symbols used in generating trees, and the number of expressions to generate. Next, these expressions are converted to their equivalent LaTeX formats and then rendered into visual representations (images). After obtaining all the elements, these sequences can be formatted as text-LaTeX pairs that can be treated as datasets for Text2LaTeX models and instruction-following pairs serving as datasets for the Image2LaTeX model.

To increase the diversity of the dataset, samples are generated by varying tree depths (ranging from depth 2 to 4) and with a diverse set of symbols. Figure 2, Figure 3, and Figure 4 demonstrate examples with the expression tree of depth 2, 3, and 4, respectively. For each sample pair, the input is a textual description of the math expression with or without an image of the rendered LaTeX expression(left) and the model is expected to output the corresponding LaTeX code (right).

The generated dataset will then be randomly split into 3 subsets: 60% for training, 20% for validation, and the final 20% for testing.

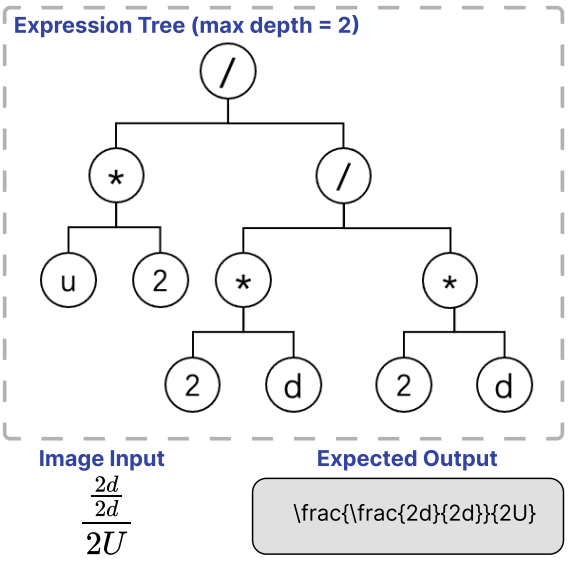

Figure 2: Demonstration of generated mathematical expressions from a tree with max depth 2. The input is a textual description of the math formula with or without rendered LaTeX expression (left) and the expected output is its LaTeX code (right).

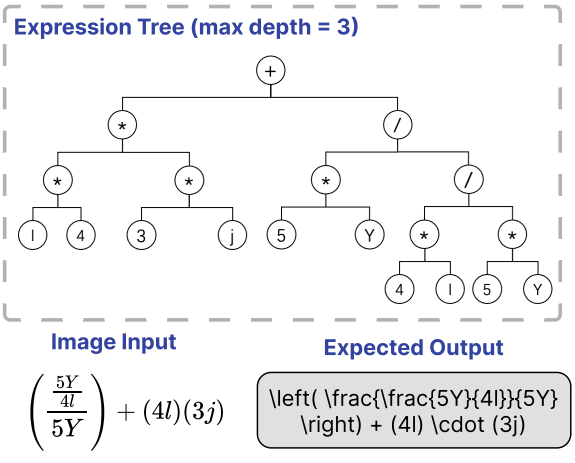

Figure 3: Demonstration of generated mathematical expressions from a tree with max depth 3.

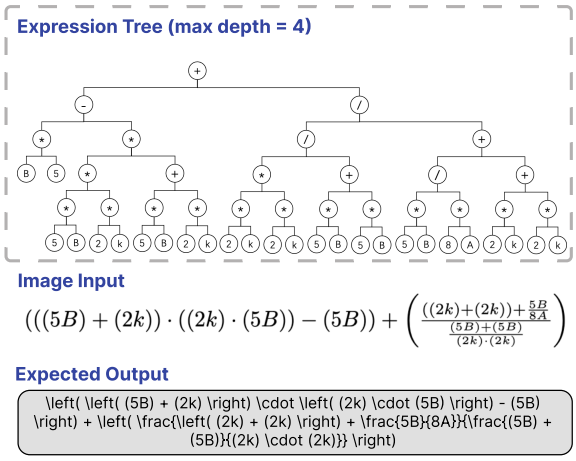

Figure 4: Demonstration of generated mathematical expressions from a tree with max depth 4. Due to the complexity of expressions, I do not consider expressions with a depth of more than 4.

3.2 LaTeX Code Generation

In this section, we illustrate the details of the model architecture and training objectives.

Model We propose two models, Text2LaTeX and Image2LaTeX, to handle various input formats while our main focus for experiments is the Text2LaTeX model.

- **Text2LaTeX**: We only include the language model f_ϕ shown in Figure 5 for the Text-to-LaTeX generation task, also known as the decoder-only model which takes in text inputs X_q (*e.g.,* "Generate the LaTeX expression for: <TEXTUAL MATH EXPRESSION>?) and the model outputs the response X_a including the corresponding LaTeX code. In this project, we utilize CodeGen (Nijkamp et al., 2022), a recent language model as the decoder pre-trained for code generation tasks.

- **Image2LaTeX**: We add a vision encoder which converts the input image X_v into a vector representation H_v to handle additional image input. The overall architecture is depicted in Figure 5, where the visual encoder Z_v is connected to the language model f_ϕ (decoder) using a linear projection layer. The language model f_ϕ takes the visual representation H_v together with a language instruction X_q e.g."what is the expression of the image?") and predicts the same response X_a as the Text2LaTeX model (the corresponding LaTeX code). We use the vision encoder in CLIP (Radford et al., 2021) in experiments.

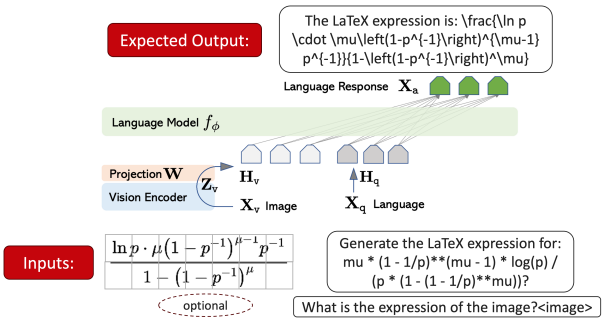

Figure 5: Model architecture. In this work, I adopt LLaVA (Liu et al., 2023), an end-to-end trainable multimodal model that combines a vision encoder and a language model as the decoder for LaTeX code generation. X_q Image2LaTeX model includes a vision encoder converting the input image $\mathbf{X}_v$ into a vector representation $\mathbf{H}_v$. The language model takes the visual representation $\mathbf{H}_v$ together with a language instruction $\mathbf{X}_q$ (*e.g.,* "what is the expression of the image?") and outputs the corresponding LaTeX code. Text2LaTeX model takes only the textual description of the math expression without any additional image information.

As the visual encoder is pre-trained on natural images, while CodeGen is not pre-trained on LaTeX, training (fine-tuning) on a dataset of (textual math expression, image (optional), LaTeX expression) pairs is necessary. I describe the training objectives next.

Training Objectives Following common practice in the literature, the training objective for the Text2LaTeX model is next token prediction while for the Image2LaTeX model, we use visual instruction tuning. We describe the details next.

- **Next Token Prediction**: Given a large vocabulary of tokens (such as a word or subword), the language model is trained to maximize the likelihood of the next token x_{n+1} conditioned on the preceding tokens in a sequence $x_1, x_2, \ldots, x_n$. The formula can be generally expressed as follows:

$$p_\theta\left(x_{n+1} \mid x_1, x_2, \ldots, x_n\right) \tag{1}$$

  where θ is the model's trainable parameter. The model is trained with the auto-regressive cross-entropy loss.

- **Visual Instruction Tuning**: In general, instruction tuning takes in (X_q, X_a) pairs as the input, where X_q denotes some human instruction, which is typically a question with context; X_a denotes the answer corresponding to X_a (Zhang et al., 2023). Instruction tuning is a flexible format that allows multi-turn conversations $(X_q^1, X_a^1, ..., X_q^T, X_a^T)$ for each image X_v, where T denotes the total number of turns. The visual information is added to the instruction in the first turn. For example, X_q^1 can be "What is the expression of the image? <IMAGE-TOKENS>" and X_a is the corresponding LaTeX expression. In the second turn, X_q^2 can be "What is the simplified expression of the image?" if the original expression has not been simplified then X_a^2 is the simplified LaTeX expression. Formally, the instruction at turn t is denoted as:

$$X_{instruct}^t = \begin{cases} \left[X_v, X_q^1\right], & t = 1 \\ X_q^t, & t > 1 \end{cases} \tag{2}$$

We perform visual instruction tuning of the language model on the prediction tokens, using its auto-regressive training objective. Specifically, the probability of getting the desired answer given an instruction sequence of length L can be computed as:

$$p\left(X_a \mid X_{instruct}\right) = \prod_{i=1}^{L} p_\theta\left(X_a^i \mid X_{instruct,i}, X_{a,<i}\right) \tag{3}$$

where θ is the trainable parameter, X_a^i is the current prediction token, $(X_{instruct,i}, X_{a,<i})$ represents $i-1$ (instruction, output) pairs and the i-th instruction.

4 Experiments

4.1 Experiment Setup

Datasets and Training Details. By default, the language model we use in experiments is CodeGen-350M-mono where 350M is the model size, and mono indicates that the model is pre-trained on Python programming language datasets. We adopt the mono version due to its popularity. We perform grid search over the initial learning rate $lr \in \{1e-8, 1e-7, 5e-7, 8e-7, 9e-7, 1e-6, 2e-6, 1e-5, 2e-5, 5e-5, 1e-4, 1e-3\}$ for CodeGen and set the optimal learning rate to $9e-7$ for the 350M version and $5e-7$ for the 2B version with a batch size of 128, a weight decay of $1e-4$, a warm-up ratio of 0.01 and train for 3200 steps. We provide a broad range of ablation studies to better understand the efficacy of different components

such as the model size, the input modality, and the input dataset complexity (by varying the depth of the expression tree).

Evaluation Metrics. Motivated by Chiang et al. (2023), we utilize GPT-4 in assessing the performance of the LaTeX generation models. For each test set, we create triplets consisting of the textual math expression query, an optional image input, and the ground-truth LaTeX code by utilizing the proposed dataset generation pipeline. Tasked with the query and the optional image input, the fine-tuned models generate responses accordingly. After receiving the responses, we feed the (ground-truth LaTeX expression, generated LaTeX expression) pairs to GPT-4 which serves as an external performance evaluator. GPT-4 evaluates the correctness of the mathematical expression and the correctness of LaTeX expressions in terms of syntax, and assigns a score of 0 or 1, where 1 indicates a correct prediction and 0 otherwise. The test accuracy is calculated by the number of pairs assigned 1 divided by the number of total pairs evaluated. To improve reliability, each evaluation is conducted 5 times with 5 random seeds and we report the average result.

4.2 Main results and analysis

Larger pre-trained models lead to superior performance. To understand the effect of the size of the pre-trained model, we compare CodeGen-350M-mono versus CodeGen-2B-mono. The results are summarized in Figure 6, which underscores the advantage of larger pre-trained models. In particular, we can see that the test accuracy of CodeGen-350M-mono stands at 0.885, while for the larger model CodeGen-2B-mono, it is noticeably higher at 0.915. This result substantiates the premise that larger pre-trained models yield better performance, likely due to their enhanced capacity for learning and generalization. Similar phenomena have been observed for language models pre-trained on large-scale datasets on the web. This is in contrast to the traditional wisdom in the machine learning community that larger models tend to overfit the training set and as a result, the generalization performance would be inferior compared to smaller models.

Training on moderate-sized datasets achieves strong code generation performance. Notably, the model displays zero accuracy without fine-tuning (*i.e.*, if we directly evaluate the pre-trained model on the test set without further fine-tuning on our generated dataset). As shown in Figure 7, when the training dataset size increases, the model's accuracy improves significantly from 0.695 at a training size of 600 to 0.95 at 5000, which indicates a clear positive correlation between training size and test accuracy. This progression highlights the critical role of fine-tuning in model performance, with larger training datasets significantly enhancing the model's code generation capabilities, underscoring the adaptability of pre-trained language models.

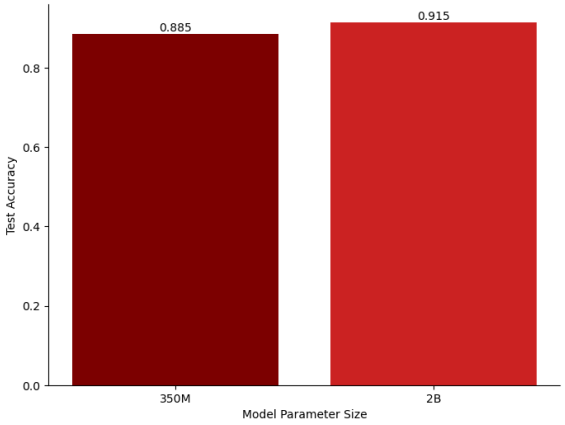

Figure 6: Test accuracy comparison of backbone models CodeGen-350M-mono vs CodeGen-2B-mono. A larger pre-trained model yields a higher accuracy rate.

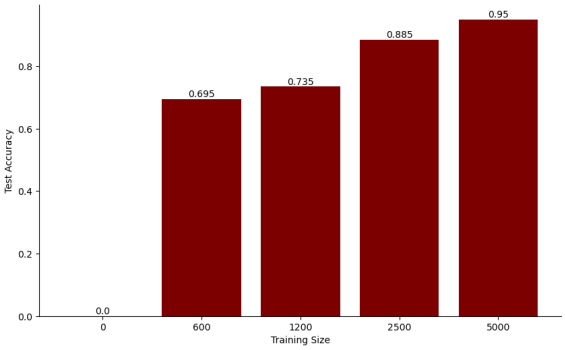

Figure 7: Relationship between training size and test accuracy. Test accuracy improves with increasing training dataset size.

Enhanced complexity in the training set improves model performance. To understand the impact of training set complexity on model efficacy, we conducted an ablation study focusing on varying depths of the training dataset. We trained models on training sets, each consisting of 2500 samples, and evaluated them on the same test set. Test accuracies, as shown in Figure 8, peaked at depth 3 (0.99) and slightly decreased at depth 4 (0.985), after an initial rise from depth 2 (0.885). This

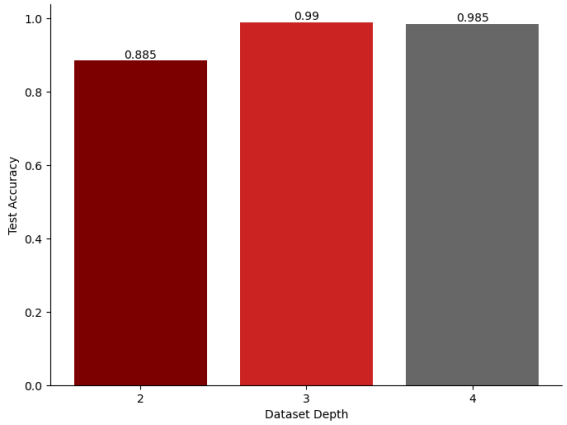

Figure 8: Performance variation with training set complexity. Illustrates model efficacy on test sets (tree depth 2) across varying complexities of training datasets, highlighting superior performance with increased complexity.

suggests that while increased complexity in the training set generally enhances model performance, there's a complexity threshold beyond which the performance gain is marginal. Understanding this threshold is crucial for optimizing the training process and achieving the best possible outcomes from the model.

Adding visual inputs does not necessarily help. To examine whether textual inputs alone contain sufficient information for models to understand the overall LaTeX generation format. and to evaluate whether additional visual information can help or distract models from learning the correct implicit representations, we compared Text2LaTeX and Image2LaTeX models across different training sizes. Recall that the difference between these two models is whether there is an additional image input (which is converted by the visual encoder to visual tokens). We find that with the additional visual information, the model's prediction accuracy does not improve, but instead drops, as depicted in Figure 9. Our hypothesis is that visual information may distract the model from understanding the input expression. In this work, we use a Vision Transformer (ViT) to encode visual information, which processes the image input by converting the image to regular patches of the same size. For example, for an expression tree of depth = 4 (Figure 4), to encode the image input consisting of complicated LaTeX expressions, the model is expected to understand fine-grained details in the patches. In contrast, a pure language model may handle such inputs better when the input is expressed as a string.

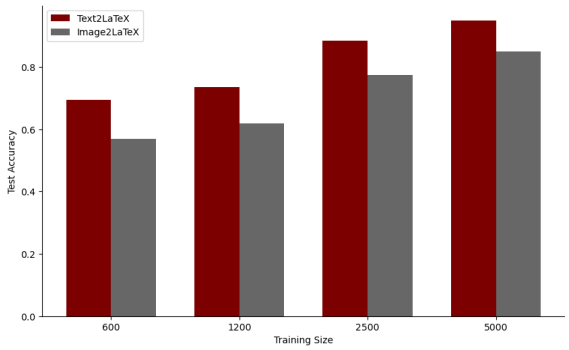

Figure 9: Comparison of Text2LaTeX and Image2LaTeX model performance by varying training sizes. The model with augmented visual input exhibit consistently lower performance compared to that with text only inputs.

5 Limitations

In this project, we mainly focus on popular models such as CodeGen as the language decoder and image encoder in CLIP. It would be desirable to expand the scope by examining a broad range of models that encode image and text information, which we leave as future work. In addition, the exploration of large-scale multi-modal models is inherently resource-demanding. Multimodal training demands significantly more computational power and GPU resources compared to pure language models. Our current experiments including training and hyperparameter tuning, exhausted a $500 student research credit on Google Cloud. This financial limitation restricted our ability to experiment with more comprehensive, potentially superior backbone models and to extend our research to larger and computationally demanding datasets.

6 Related Works

Large Language Models. The landscape of large language models has significantly evolved since Transformer (Vaswani et al., 2017) was first introduced in 2017. GPT-2 (Radford et al., 2019) highlighted the efficacy of next-token prediction, followed by GPT-3 (Brown et al., 2020), which underscored the benefits of scaling model size. In recent years, we have witnessed a wave of diverse Transformer variants, addressing diverse tasks and domains with improved performance. Notably, LLaMA (Touvron et al., 2023) gained prominence as a versatile, open-source model. Vicuna (Chiang et al., 2023) further advanced the field by specializing in instruction tuning. In the context of code gen-

eration, CodeGen (Nijkamp et al., 2022) emerged as a pivotal development, offering a model specifically optimized for such tasks. Our study leverages CodeGen as the backbone language model, aligning with its specialized capabilities and our research objectives.

Vision Language Models. Despite the effectiveness of language models, their inability to process visual information remains a limitation. Multimodal models, however, excel in integrating diverse modalities, including images, videos, document layouts, etc. A significant advancement in this domain is LLaVA (Liu et al., 2023), which incorporates visual inputs into the language model backbone through a linear adaptor. In our research, we employ LLaVA's framework as the foundation for developing the Text2LaTeX and Image2LaTeX models, exploiting its capability to harmonize visual and textual data.

LaTeX Code Generation. Traditionally, LaTeX code generation predominantly relied on rule-based methods(Suzuki et al., 2003). While efficient for structured and predictable inputs, these approaches fall short of accommodating the variability and complexity inherent in more flexible input formats. The rigidity of rule-based systems limits their adaptability to diverse and less structured data. A few work have been tried to utilize deep learning based methods but yield limited success (Blecher, 2023). We are the first to explore LaTeX generation task with modern multi-modal foundation models pre-trained on large-scale datasets.

7 Conclusion

Generating LaTeX code that follows users' intentions is a crucial task in the digital era, which requires the model to understand user inputs presented in various formats such as texts or images and correctly generate syntactically correct code. In this work, we investigate the effectiveness of LaTeX program synthesis by utilizing pre-trained multi-modal foundation models. We propose a tree-based approach to generate valid and diverse LaTeX expression datasets can be easily scaled. We trained two categories of models: one capable of processing purely textual inputs and the other equipped to handle a combination of textual and visual inputs. Our experiments and ablation studies indicate that pre-trained language models with moderate-scale fine-tuning can effectively adapt to the task of LaTeX code generation. Additionally, we demonstrate that incorporating visual inputs may not further enhance code generation, possibly due to the increased complexity of processing fine-grained details in images. We hope our work will serve as a stepping stone toward more comprehensive evaluations and the design of efficient algorithms for LaTeX code generation.

References

Lukas Blecher. 2023. LaTeX-OCR. https://github.com/lukas-blecher/LaTeX-OCR.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In *International Conference on Machine Learning*, pages 980–989. PMLR.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

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

Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori. 2003. Infty: an integrated ocr system for mathematical documents. In *Proceedings of the 2003 ACM symposium on Document engineering*, pages 95–104.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jian Wang, Yunchuan Sun, and Shenling Wang. 2019. Image to latex with densenet encoder and joint attention. *Procedia computer science*, 147:374–380.

Zelun Wang and Jyh-Charn Liu. 2021. Translating math formula images to latex sequences using deep neural networks with sequence-level training. *International Journal on Document Analysis and Recognition (IJDAR)*, 24(1-2):63–75.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Wei Zhang, Zhiqiang Bai, and Yuesheng Zhu. 2019. An improved approach based on cnn-rnns for mathematical expression recognition. In *Proceedings of the 2019 4th international conference on multimedia systems and signal processing*, pages 57–61.