数学公式的数字化转换是科研、教育领域的重要需求，但现有光学公式识别（OCR）系统对复杂公式结构（如多级分数、矩阵、特殊符号）的识别精度有限，尤其在处理手写公式时性能显著下降。本项目旨在通过微调视觉语言模型（VL Model），开发一个鲁棒的公式图像转LaTeX源码系统，解决现有方案对打印/手写公式的泛化性问题。我们会重点调研：1）视觉-语言模型架构（如Pix2Seq、Donut），分析其图像到文本的映射机制；2）数学公式识别相关研究（如IM2LATEX-100K数据集论文）；3）开源公式识别工具（如LaTeX-OCR、Mathpix）的技术报告。我们将使用开源数据集 Data-for-LaTeX\_OCR 进行模型训练，具体技术路线为开源数据集微调视觉语言模型。考虑到实际应用时，该应用通常被部署在端侧（如 Notability 中的公式转换），我们会考虑使用较小的模型，或者应用模型轻量化技术。我们会使用以下指标验证效果：1）文本相似度指标（BLEU、编辑距离、完全匹配率）；2）结构准确性指标（解析生成LaTeX的AST与真实值的子树匹配率）。此外我们还会使用没有微调的语言文字模型作为基线，比较效果。

Two-stage solution

先是OCR阶段做文档检测和识别，然后再进行结构化解析，通常分模块训练，依赖深度学习OCR和真实OCR标注数据。

端到端solution

纯Transformer架构，从图像映射到文本输出，可使用合成数据预训练，增强了大模型视觉文档理解(VDU)能力。

-Donut (Swin Transformer + BART Decoder)

-TrOCR (ViT + BART Decoder)

-Pretrained VL model eg. Qwen2.5-VL

BART和GPT的区别

模型架构上，GPT是解码器叠加，单向注意力解码，通过自回归生成文本，适合开放式文本生成任务。而BART由双向编码器和单向Autoregressive transformer组成，可双向理解输入，适合先理解再生成的任务如翻译和摘要。

1. 确定模型架构-ViT + BART
2. 怎么训练-混合手写和印刷，分成train test validate，微调模型，怎么做data augmentation

图像预处理  
注意[unilm/trocr/data.py at master · microsoft/unilm](https://github.com/microsoft/unilm/blob/master/trocr/data.py)的data.py和data\_aug.py

1. 发现padding然后resize成统一的正方形dim会导致resolution降低
2. 将输入图像统一pad成64\*784

目前data.py实现：二值化，空白裁剪，pad成长方形，转回RGB

--------------------------------------------------------------------------------------------------------------

现在有一个问题，需不需要normalize\_latex，如果需要，要怎么normalize

-不需要normalize

How do you handle latex tokenization

我想不懂要怎么提升BLEU，llm也给不出答案

Another data preprocessing sources

<https://github.com/untrix/im2latex/tree/master/src/preprocessing>

--------------------------------------------------------------------------------------------------------------

train bpe tokenization

-latex\_tokenizer.json

Evaluation Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Encoder | Decoder | BLEU(full) | EM(full) | BLEU(syn) | EM(syn) |
| vit-patch16-224 | bart | 0.5159 | 1.366% | 0.4425 | 0.2919% |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

{'loss': 2.6433, 'grad\_norm': 155932.15625, 'learning\_rate': 4.79035639412998e-05, 'epoch': 0.21}

{'loss': 0.5597, 'grad\_norm': 143908.828125, 'learning\_rate': 1.781970649895178e-06, 'epoch': 4.82}

{'train\_runtime': 3835.898, 'train\_samples\_per\_second': 99.479, 'train\_steps\_per\_second': 3.109, 'train\_loss': 1.0458569399915651, 'epoch': 5.0}