MathWriting: A Dataset For Handwritten Mathematical Expression Recognition

Philippe Gervais* pgervais@acm.org

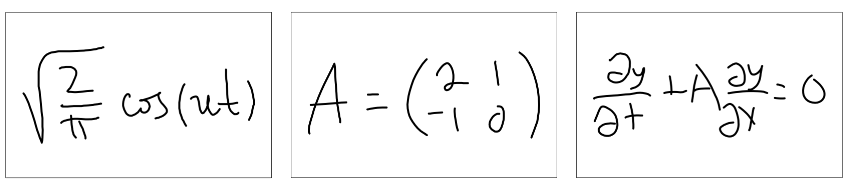
Asya Fadeeva Google fadeich@google.com

Andrii Maksai Google amaksai@google.com

Abstract

Recognition of handwritten mathematical expressions allows to transfer scientific notes into their digital form. It facilitates the sharing, searching, and preservation of scientific information. We introduce MathWriting, the largest online handwritten mathematical expression dataset to date. It consists of **230k human-written samples** and an additional **400k synthetic ones**. This dataset can also be used in its rendered form for offline HME recognition. One MathWriting sample consists of a formula written on a touch screen and a corresponding LaTeX expression. We also provide a normalized version of LaTeX expression to simplify the recognition task and enhance the result quality. We provide baseline performance of standard models like OCR and CTC Transformer as well as Vision-Language Models like PaLI on the dataset. The dataset together with an example colab is accessible on Github.

1 Introduction



72a078dfeb8e6027 c96d0c67f82ee512 fe938af2c772a57a

Three examples of HME from MathWriting. More examples can be found in Appendix K. Each ink is accompanied by a unique identifier that matches a corresponding filename in the dataset.

MathWriting dataset (2.9 GB):
`https://storage.googleapis.com/mathwriting_data/mathwriting-2024.tgz`
Associated code:
`https://github.com/google-research/google-research/tree/master/mathwriting`

Online *text* recognition models have improved a lot over the past years, because of improvements in model structure [1, 2, 3] and also because of an increase in the amount of training data [4, 5, 6]. Mathematical expression (**ME**) recognition is a challenging task that has received less attention than regular recognition of words and characters [7]. ME recognition is different from regular text recognition in a number of interesting ways which can prevent improvements from transferring from one to the other. Though MEs share with text most of their symbols, they follow a more rigid structure which is also two-dimensional, see Figure 1. Where text can be treated to some extent as a one-dimensional problem amenable to sequence modeling, MEs cannot because the relative position

*Work performed while employed at Google

Submitted to the 37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks. Do not distribute.

of symbols in space is meaningful. It is also different from symbol segmentation or object detection because the output of a recognizer has to contain the relationship between symbols, serialized in some form (LaTeX, a graph, InkML, etc.). Similarly to the case of text, *handwritten* MEs (**HME**) are more difficult to recognize than *printed* ones as they are more ambiguous and less training data is available.

Handwritten data is costly to obtain as it must be written by hand, which is compounded in the case of online representation (**ink**) by the necessity to use dedicated hardware (touchscreen, digital pen, etc.). By publishing the MathWriting dataset, we hope to alleviate some of the needs for data for research purposes. Samples include a large number of human-written inks, as well as synthetic ones. MathWriting can readily be used with other online datasets like CROHME [8] or Detexify [9] - we publish the data in InkML format to facilitate this. It can also be used for offline ME recognition simply by rasterizing the inks, using code provided on the Github page[2].

MathWriting is the largest set of online HME published so far - both human-written and synthetic. It significantly expands the set of symbols covered by CROHME [8], enabling more sophisticated recognition capabilities. Since inks can be rasterized, MathWriting can also been seen as larger than existing offline HME datasets [10, 11, 12]. For these reasons we introduce a new benchmark, applicable to both online and offline ME recognition.

This work's main contributions are:

- a large dataset of Handwritten Mathematical Expressions under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International[3].
- LaTeX ground truth expressions in normalized form to simplify training and to make evaluation more robust.
- Evaluation of different models like CTC Transformer and PaLI on the dataset to show what recognition quality could be achieved with the provided data.

The paper focuses on the high-level description of the dataset: creation process, postprocessing, train/test split, ground truth normalization, statistics, and a general discussion of the dataset content to help practitioners understand what can and cannot be achieved with it. All the low-level technical information like file formats can be found in the `readme.md` file present at the root of the dataset archive linked above. We also provide code examples on Github[2], to show how to read the various files, process and rasterize the inks, and tokenize the LaTeX ground truth.

2 Dataset Creation

MathWriting dataset primarily consists of LaTeX expressions from Wikipedia, more details about the acquisition of expressions are provided in Appendix B. These expressions were used for both ink collection from human contributors Section 2.1 as well as synthetic data generation Section 2.2. We did a very limited filtering of very noisy human-written examples (described in Appendix C).

2.1 Ink Collection

Inks were obtained from human contributors through an in-house Android app. Participants agreed to the standard Google terms of use and privacy policy. The task consisted in copying a rendered mathematical expression (prompt) shown on the device's screen using either a digital pen or a finger on a touch screen. Mathematical expressions used as prompt were first obtained in LaTeX format, then rendered into a bitmap through the LaTeX compiler (see Appendix A for the template used). 95% of MathWriting expressions were obtained from Wikipedia. The remaining ones were generated to cover underrepresented cases in Wikipedia, like isolated letters with nested sub/superscripts or complicated fractions (see Section B). Contributors were hired internally at Google. 6 collection campaigns were run between 2016 and 2019, each lasting between 2 to 3 weeks. Collected data contains only inks and labels, so no personally identifiable information is present in the dataset. Offensive content is highly unlikely because LaTeX expressions were taken from Wikipedia and we conducted a filtering of noisy data (described in Appendix C).

[2] https://github.com/google-research/google-research/tree/master/mathwriting

[3] https://creativecommons.org/licenses/by-nc-sa/4.0/

2.2 Synthetic Samples and Isolated Symbols

We created synthetic samples in order to further increase the label diversity for training. This also enabled compensating for limitations of the human collection like the maximum length of the expressions, which were limited by the size of the screen they were written on. We used LaTeX expressions from Wikipedia that were not used in the data collection. The resulting `synthetic` data has a 90th percentile of expression length of 68 characters, compared to 51 in `train`. This is especially important as deep neural nets often fail to generalize to inputs longer than their training data [13, 14]. Using synthetic long inks together with the original human-written inks can help to eliminate that problem as shown in [15, 16]. The synthesis technique is as follows: starting from a raw LaTeX mathematical expression, we computed a DVI file using the LaTeX compiler, from which we extracted bounding boxes. We then used those bounding boxes to place handwritten individual symbols, resulting in a complete expression. See Figure 1 for an example of extracted bounding boxes and the resulting synthetic example.

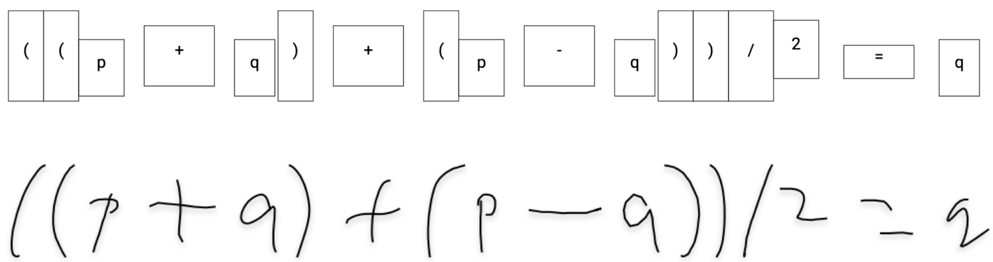


Figure 1: An example of a synthetic ink created from bounding boxes with label `((p+q)+(p-q))/2=q`

Inks for individual symbols are all from the `symbols` split. They have been manually extracted from inks in `train`. For each symbol that we wanted to support, we manually selected strokes corresponding to it for 20-30 distinct occurrences in `train`, and used that information to generate a set of individual inks. Similar synthesis techniques have been used by [8] with inks, [10] and [12] with raster images.

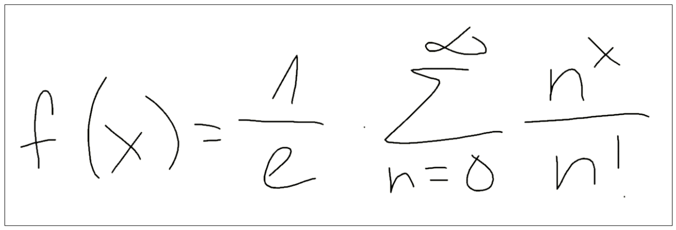
A significant difference between synthetic and human-written inks is the stroke order. For synthetic inks, stroke order follows the order of the bounding boxes in the DVI file, which can be different from the usual order of writing for mathematical expressions. However, the writing order within a given symbol is consistent with human writing.

2.3 Dataset Split

MathWriting is composed of five different sets of samples, which we call 'splits': `train`, `valid`, `test`, `symbols`, and `synthetic`. The splits `train`, `valid` and `test` consist only of human-written examples. The split `symbols` is provided for synthetic data generation and is not used in training. The split of human-written samples between `train`, `valid` and `test` was partially done based on writers, partially based on labels. More details are provided in Appendix D. Experiments have shown that a more important factor than the handwriting style was whether the *label* had already been seen during training. This fact is also supported by research in the area of compositional generalization [17]. In the published version, `valid` has a 55% (8.5k samples) intersection with `train` based on unique normalized labels, and `test` has an 8% intersection (647 samples). We chose to have a low intersection between `train` and `test` in order to correctly measure generalization of trained models to unseen labels.

2.4 Label Normalization

All samples in the dataset come with two labels: the LaTeX expression that was used during the data collection (annotation `label` in the InkML files), and a normalized version of it meant for model training, which is free from a few sources of confusions for an ML model (annotation `normalizedLabel`). An example with original and normalized labels is provided in Figure 2. Label normalization covers three main categories (details are provided in Appendix E):



51b364eb9ba2185a

Figure 2: An example from the `train` split, with its labels:
Raw: `f(x)=\frac1e\cdot \sum_{n=0}^{\infty}{n^{x}\over n!}`
Normalized: `f(x)=\frac{1}{e}\cdot\sum_{n=0}^{\infty}\frac{n^{x}}{n!}`

- variations used in print that can't be reproduced in handwriting - e.g. bold, italic - or that haven't been reproduced consistently by contributors.
- non-uniqueness of the LaTeX syntax. e.g. `\frac{1}{2}` and `1\over 2` are equivalent.
- visual variations that can reproduced in handwriting but can't reliably be inferred by a model. This includes size modifiers like `\left`, `\right`.

We provide the raw labels to make it possible to experiment with alternative normalization schemes, which could lead to better outcomes for different applications.

2.4.1 Limitations of normalization

The normalization process is purely syntactic, and can not cover cases where the meaning of the expression has to be taken into account. For example, a lot of expressions from Wikipedia use `cos` instead of `\cos`. It is often clear to a human reader whether the sequence of characters c,o,s represents the `\cos` command or simply three letters. However, this can not be reliably inferred by a syntactic parser, for example in `tacos` vs `ta\cos`. An alternative would be to update the raw labels, which we didn't do because we wanted to keep the information that was used during the collection as untouched as possible. Similarly, cases like `10^{-1}` usually mean `{10}^{-1}`, though they render exactly the same. We made the choice to normalize to the former because it's the only option with a purely syntactic normalizer. It's also better than not removing these extra braces because it gives more consistent label structures, which simplifies the model training problem.

3 Dataset Statistics

In this section we describe the key characteristics of MathWriting and compare it to CROHME23 [8]. In Table 1 we provide the information about the volume of the dataset splits both in terms of examples (inks) and unique labels.

Table 1: Statistics on different subsets of MathWriting dataset.

	`train`	`synthetic`	`valid`	`test`
# distinct inks	230k	396k	16k	8k
# distinct labels	53k	396k	8k	4k

3.1 Label Statistics

MathWriting contains 457k unique labels after normalization (see Section 2.4). From Table 1 we see that most unique expressions are covered by the synthetic portion of the dataset. However, the absolute number of unique expressions in human-written part is still high – 61k. This underlines the importance of synthetic data as it allows models to see a much bigger variety of expressions. It is important to note that the `synthetic` split has essentially no repeated expressions. On the other hand, in real data multiple different writings of the same expression are quite common (see Figure 11 in Appendix F). This fact allows us to separately evaluate model's quality on expressions that were observed during training and that those that hadn't. As seen in Table 2 the biggest intersection in expressions is between `valid` and `train`. The minimal overlap between `test` and `train` splits is beneficial for assessing a model's ability to generalize to expressions that were not seen in train.

Table 2: Counts of unique labels shared between MathWriting splits

	`train`	`synthetic`	`valid`	`test`
`train`	-	0	3.6k	355
`synthetic`	0	-	0	0
`valid`	3.6k	0	-	239
`test`	355	0	239	-

The median length of expressions in characters is 26 which is comparable to one of the most popular English recognition datasets IAMonDB [18] which has median of 29 characters. However, it is important to note that LaTeX expressions have tokens that span multiple characters like `\frac`. The median length of expressions in tokens (provided in Appendix J) is 17, thus making training a model on tokens rather then characters easier due to shorter target lengths [19, 20]. We want to emphasize that MathWriting can be used with a different tokenization scheme and token vocabulary from what we propose in Appendix J. In Figure 3 we show the number of occurrences for the most frequent tokens. Tokens `{` and `}` are by far the most frequent as they are integral to the LaTeX syntax.

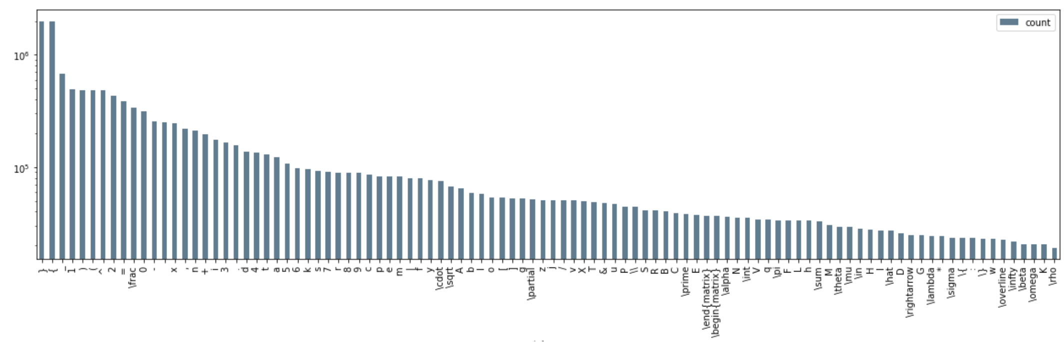


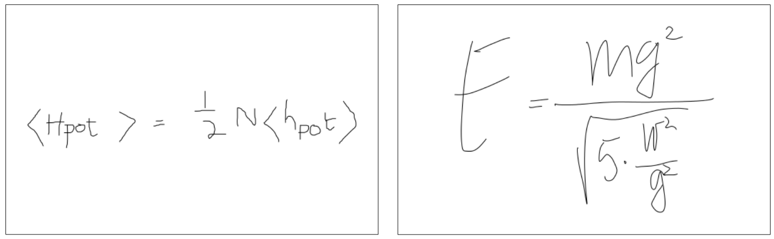
Figure 3: Histogram of the top-100 most frequent tokens in MathWriting.

3.2 Ink Statistics

Each ink in MathWriting dataset is a sequence of strokes I $= [s_0, \ldots, s_n]$, each stroke s_i consisting of points. A point is represented as a triplet (x, y, t) where x and y are coordinates on the screen and t is a timestamp. In Table 3 we provide statistics on number of strokes, points, and duration of writing. It's important to note that as inks were collected on different devices, the absolute coordinate values can vary a lot. In human-written data the time information t always starts from 0 but it is not always the case in the `synthetic` split. Different samples often have different sampling rates (number of points written in one second) due to the use of different devices (see Figure 4). More details in Section 3.3. Consequently, the same ink written on two different devices can result in inks with a different number of points. For human-written inks, the sampling rate is consistent between strokes, but it is not the case for synthetic ones. In order to accommodate a model and make sequences shorter, inks can be resampled in time (see example in Figure 13, Appendix F).

Table 3: Ink statistics for MathWriting.

	10th percentile	median	90th percentile
# strokes	5	14	39
# points	131	350	1069
writing time (sec)	1.88	6.03	16.42
aspect ratio	1.32	3.53	9.85



a21ff5968b022586 17b337d8b7ac540b

Figure 4: Left: an ink with very low sampling rate (9.4 points per second)
Right: an ink with very high sampling rate (260 points per second)

Table 4: Counts of inks, distinct labels and distinct tokens used in MathWriting and CROHME23. The single token present in CROHME23 but not in MathWriting is the literal dollar sign `\$`.

	MathWriting	CROHME23	Common
Inks	650k	164k	0
Labels	457k	102k	47k
Vocab	254	105	104

Table 5: Count of human-written and synthetic inks for MathWriting and CROHME23. Human-written inks represent 38% of the total for MathWriting, and 10% for CROHME23.

	MathWriting	CROHME23
human	253k	17k
`synthetic`	396k	147k

3.3 Devices Used

Around 150 distinct device types have been used by contributors. In most cases inks were written on smartphones using a finger on a touchscreen. However, there are cases where tablets with styluses were used. The main device used in this case is Google Pixelbook, which accounted for 51k inks total (see Table 7, Appendix F). Out of all device types, 37 contributed more than 1000 inks. Note that writing on a touchscreen with a finger or a stylus results in different low-level artifacts. All devices were running the same Android application for ink collection, regardless of whether their operating system was Android or ChromeOS.

3.4 Comparison With CROHME23

In this section we compare main dataset statistics of MathWriting and CROHME23 [8] as it is a popular publicly available dataset for HME recognition. In terms of overall size, MathWriting has nearly 3.9 times as many samples and 4.5 times as many distinct labels after normalization, see Table 4. A significant number of labels can be found in both datasets (47k), but the majority is dataset-specific. This suggests that combining both datasets during training could yield improved HME recognition quality. MathWriting has more human-written inks than CROHME23 as seen in Table 5, and contains a much larger variety of tokens. It has 254 distinct tokens including all Latin capital letters and almost the entire Greek alphabet. It also contains matrices, which are not included in CROHME23. Therefore, more scientific fields like quantum mechanics, differential calculus, and linear algebra can be represented using MathWriting.

4 Experiments

4.1 Evaluation setup

We propose the following evaluation setup based on MathWriting for the quality of handwriting math expression recognition.

- **evaluation samples**: the `test` split of MathWriting.
- **metric**: character error rate (CER) [21], where a "character" is a LaTeX token as defined by the code in Appendix I.

We provide a reference implementation of the evaluation metric at the Github page [2]. We propose the use of CER as a metric to make results comparable to other recognition tasks like text recognition [22, 23], and the use of LaTeX tokens instead of ASCII characters so that an error on a single non-latin letter (e.g. `\alpha` recognized as `a`) counts as one instead of many.

4.2 Baseline Recognition Models

In Table 6 we provide results for different models. All models are trained exclusively on the MathWriting dataset (`train` and `synthetic`), except for the OCR API that was trained on other datasets as well. The following models represent different approaches to handwriting recognition – offline [23], online [24] and mixed [25].

OCR This is a publicly available Document AI OCR API [26], which processes bitmap images. It has been trained partly on samples from MathWriting. We sent inks rendered with black ink on a white background and searched for optimal image size and stroke width to get the best evaluation result from the model.

CTC Transformer This model is a transformer base with a Connectionist Temporal Classification loss on top (**CTC**) [27]. It contains 11 transformer layers with an embedding size of 512. We used swish activation function and dropout of 0.15 as those parameters performed best on `valid`. We train with an Adam optimizer, learning rate of 1e-3, batch size 256 for 100k steps. One training run took 4 hours on 4 TPU v2. We trained from scratch and exclusively on MathWriting (`train` and `synthetic`). The model is similar to [24], replacing LSTM layers by Transformer layers and not using any external language model on top.

VLM We fine-tuned a large Vision-Language Model PaLI [28] on MathWriting (`train` and `synthetic`). We used the representation proposed in [25] where an ink is represented as both a sequence of points (similar to CTC Transformer) and its rasterized version (similar to OCR). We train three models with different data shuffling for 200k steps with batch size 128, learning rate 0.3 and dropout 0.2. One training run took 14 hours on 16 TPU v5p. Models were finetuned exclusively on `train` and `synthetic` MathWriting data. Overall, it took 2 TPU v2 days and 28 TPU v5p days to run the experiments.

Table 6: Recognition results for different models. The evaluation metric is reported on both the `valid` and `test` splits.

Model	Input	Parameters	CER on `valid`	CER on `test`
OCR [26]	Image	-	6.50	7.17
CTC Transformer [25]	Ink	35M	4.52 (0.08)	5.49 (0.05)
PaLI [25]	Image+Ink	700M	4.47 (0.08)	5.95 (0.06)

Table 6 shows the evaluation comparison between the three models. The OCR model has no information about the order of writing and speed (offline recognition), which explains its lower performance than methods that take time information into account (online recognition). The two other methods – PaLI and CTC Transformer perform significantly better than OCR. These results show that our dataset can be used to train classical recognition models like CTC transformer as well as more recent architectures like VLM.

Figure 5 shows examples of model mistakes. Two of the main causes of mistakes are confusing similar-looking characters like “z” and “2”, and errors in the structural arrangement of the characters, for instance not placing a sub-expression in a subscript or superscript.

5 Discussion

5.1 Differences in Writing Style

The number of contributors was large enough that a variety of writing styles are represented in the dataset. An example for different ways of writing letter ‘r’ can be seen in Figure 6. Additional

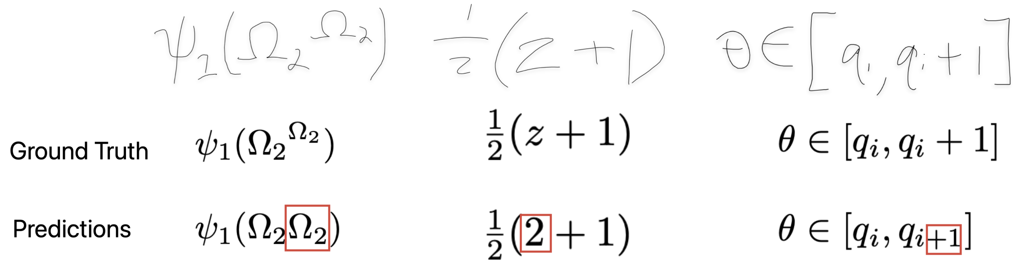
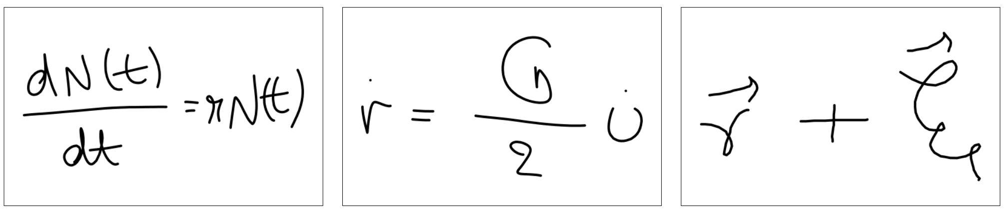


Figure 5: Examples of recognition mistakes from the CTC Transformer model. We observe similar mistakes from the other models.



d4e9c7b8f1ffb958 0dca307a1895512d 45af9641d8d2ed56

Figure 6: Three ways of writing a lowercase 'r'.

examples are provided in Figure 7. Similar though less obvious differences exist for other letters. Style differences also show through writing order (example – Figure 14, Appendix G).

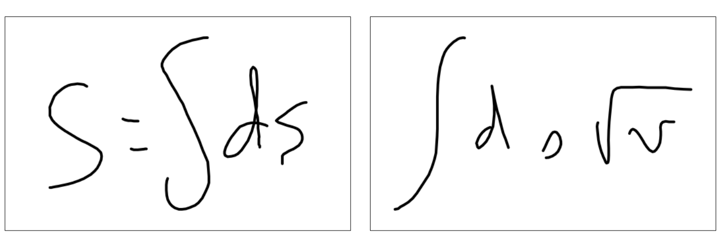
5.2 Recognition Challenges

MathWriting presents some inherent recognition challenges, which are typical of handwritten representations. For example, it's not really possible to distinguish these pairs from the ink alone: `\frac{\underline{a}}{b}` vs `\frac{a}{\overline{b}}`, and `\overline\omega` vs `\varpi`. We'd like to point out that these ambiguities are not an issue for humans in practice, because they rely on contextual information to disambiguate: a particular writing idiosyncrasy, consistency with nearby expressions, knowledge of the scientific domain, etc. See Figures 8 and 9 for more examples.

5.3 Dataset Applications and Future Work

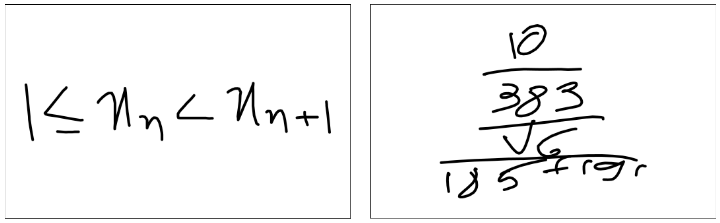
Mathwriting can be used to train recognizers for a large variety of scientific fields, and is also large enough to enable synthesis of mathematical expressions. Combining it with other large datasets like CROHME23 would increase the variety of samples even further, both in terms of writing style and number of expressions, likely improving the performance of a model.

Bounding box information for synthetic samples together with individual symbols are provided to enable experimentation with synthetic ink generation. Synthetic samples were generated through the straightforward process of pasting inks of individual symbols (`symbols`) exactly where bounding boxes were located. This gives synthetic samples a very regular structure, see Figure 1. It is possible



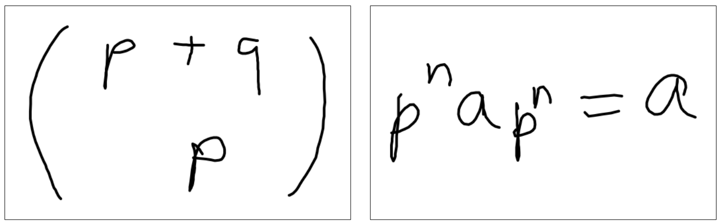
a933cd67f7891dc8 ecc157b89c3e344d

Figure 7: Two ways of writing lowercase s.



357313f77d65b804 b8b8cff97f5c044a

Figure 8: Left: character ambiguity. Is it $1 \leq x_n < x_{n+1}$ or $1 \leq n_\eta < n_{\eta+1}$? Right: what is the fraction nesting order?



96bf9fb2da96db9e e50b0275ef2c9549

Figure 9:
Left: `\binom` or 2-element matrix? Right: $p^n a_{p^n} = a$ or $p^n a_p n = a$?

to improve this process by modifying the location, size or orientation of bounding boxes prior to generating the synthetic inks. This would soften LaTeX's rigid structure and make synthetic data closer to human handwriting. Another application of these bounding boxes would be to bootstrap a recognizer that would also return character segmentation information. This kind of output is critical for some UI features - for example, editing an handwritten expression.

MathWriting can also be improved by varying the label normalization. Changing it can have different benefits depending on the application, as mentioned above. We provide the source LaTeX string for that reason. Another possible improvement in recognition can come from additional contextual information, for instance the scientific field [29] that can be added post-hoc. Combining recognizers with a language model [24] trained on a large set of mathematical expressions would be a step in a similar direction.

6 Limitations

A single sample in MathWriting dataset has one handwritten LaTeX formula, see Figure 2. As a result, models that are trained on this dataset would probably perform poorly on complete handwritten documents, such as the IAMonDo dataset [30]. Also, as the dataset contains only LaTeX expressions, it is unlikely that models trained on it will accurately recognize handwritten text in English or other languages. As shown in Figure 3, some LaTeX tokens are way more frequent than others. Some infrequent tokens like `\ni` could be hard to recognise.

7 Conclusion

We introduced MathWriting, the largest dataset of online handwritten mathematical expressions to date, together with the experimental results of three different types of models. We hope this dataset will help advance research in both online and offline mathematical expression recognition. Additionally, we invite data practitioners to build on the dataset. We intentionally chose a file format for MathWriting close to the one used by CROHME to facilitate their combined use. We also provided original or intermediate representations (raw LaTeX strings, bounding boxes) to enable experimentation with the data itself, and suggested a few directions (Section 5.3).

References

[1] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:855–868, 2009.

[2] Johannes Michael, Roger Labahn, Tobias Grüning, and Jochen Zöllner. Evaluating sequence-to-sequence models for handwritten text recognition. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1286–1293, 2019.

[3] Yutian Liu, Wenjun Ke, and Jianguo Wei. Attention guidance mechanism for handwritten mathematical expression recognition, 2024.

[4] Emre Aksan, Fabrizio Pece, and Otmar Hilliges. Deepwriting: Making digital ink editable via deep generative modeling, 2018.

[5] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014). volume 2014, 09 2014.

[6] Harold Mouchère, Christian Viard-Gaudin, Richard Zanibbi, and U. Garain. Icfhr2016 crohme: Competition on recognition of online handwritten mathematical expressions. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 607–612, 2016.

[7] May Mowaffaq AL-Taee, Sonia Ben Hassen Neji, and Mondher Frikha. Handwritten recognition: A survey. *2020 IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS)*, pages 199–205, 2020.

[8] Yejing Xie, Harold Mouchère, Foteini Liwicki, Sumit Rakesh, Rajkumar Saini, Masaki Nakagawa, Cuong Nguyen, and Thanh-Nghia Truong. *ICDAR 2023 CROHME: Competition on Recognition of Handwritten Mathematical Expressions*, pages 553–565. 08 2023.

[9] Detexify data. `https://github.com/kirel/detexify-data`.

[10] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. Image-to-markup generation with coarse-to-fine attention, 2017.

[11] Ye Yuan, Xiao Liu, Wondimu Dikubab, Hui Liu, Zhilong Ji, Zhongqin Wu, and Xiang Bai. Syntax-aware network for handwritten mathematical expression recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4553–4562, 2022.

[12] Aida calculus math handwriting recognition dataset. `https://www.kaggle.com/datasets/aidapearson/ocr-data`.

[13] Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models, 2022.

[14] Dusan Varis and Ondřej Bojar. Sequence length is a domain: Length-based overfitting in transformer models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

[15] Aleksandr Timofeev, Anastasiia Fadeeva, Andrei Afonin, Claudiu Musat, and Andrii Maksai. *DSS: Synthesizing Long Digital Ink Using Data Augmentation, Style Encoding and Split Generation*, page 217–235. Springer Nature Switzerland, 2023.

[16] Arun Narayanan, Rohit Prabhavalkar, Chung-Cheng Chiu, David Rybach, Tara N. Sainath, and Trevor Strohman. Recognizing long-form speech using streaming end-to-end models, 2019.

[17] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data, 2020.

[18] M. Liwicki and H. Bunke. IAM-OnDB-an on-line english sentence database acquired from handwritten text on a whiteboard. In *ICDAR'05*. IEEE, 2005.

[19] Dusan Varis and Ondřej Bojar. Sequence length is a domain: Length-based overfitting in transformer models. pages 8246–8257, 01 2021.

[20] Masato Neishi and Naoki Yoshinaga. On the relation between position information and sentence length in neural machine translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 328–338, Hong Kong, China, November 2019. Association for Computational Linguistics.

[21] Johannes Michael, Roger Labahn, Tobias Grüning, and Jochen Zöllner. Evaluating sequence-to-sequence models for handwritten text recognition, 2019.

[22] George Retsinas, Giorgos Sfikas, Basilis Gatos, and Christophoros Nikou. Best practices for a handwritten text recognition system, 2024.

[23] Dmitrijs Kass and Ekta Vats. Attentionhtr: Handwritten text recognition based on attention encoder-decoder networks, 2022.

[24] Victor Carbune, Pedro Gonnet, Thomas Deselaers, Henry A. Rowley, Alexander Daryin, Marcos Calvo, Li-Lun Wang, Daniel Keysers, Sandro Feuz, and Philippe Gervais. Fast multi-language lstm-based online handwriting recognition, 2020.

[25] Anastasiia Fadeeva, Philippe Schlattner, Andrii Maksai, Mark Collier, Efi Kokiopoulou, Jesse Berent, and Claudiu Musat. Representing online handwriting for recognition in large vision-language models, 2024.

[26] Google Cloud. Detect handwriting in image, 2023.

[27] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA, 2006. Association for Computing Machinery.

[28] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme, Andreas Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali: A jointly-scaled multilingual language-image model, 2023.

[29] Mark Collier, Rodolphe Jenatton, Efi Kokiopoulou, and Jesse Berent. Transfer and marginalize: Explaining away label noise with privileged information. In *International Conference on Machine Learning*, 2022.

[30] Emanuel Indermühle, Marcus Liwicki, and Horst Bunke. Iamondo-database: An online handwritten document database with non-uniform contents. pages 97–104, 06 2010.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We claim to provide a large dataset of HME – see Section 1, together with normalized labels – the process is described in Section 2.4 and example is provide in Figure 2. Experimental results on this dataset are presented in Section 4.2.
   (b) Did you describe the limitations of your work? [Yes] We described general limitations of MathWriting dataset in Section 6, limitations of label normalization in Section 2.4.1, recognition challenges of mathematical expressions in Section 5.2 and sources of noise in the dataset in Section H.

(c) Did you discuss any potential negative societal impacts of your work? [N/A] The type of the dataset we are publiching is not new, there are similar datasets like CROHME23 [8]. Given the widespread use of handwriting recognition, we don't see any potential negative impacts of our work.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] All the participants were payed the minimum hourly rate as discussed in Section 2.1. The dataset doesn't include any personal information about contributors apart from their handwriting samples that they agreed to share.

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Our paper doesn't include any theoretical results.

   (b) Did you include complete proofs of all theoretical results? [N/A] Our paper doesn't include any theoretical results.

3. If you ran experiments (e.g. for benchmarks)...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] All experiments in this paper are conducted using publicly available datasets. We provide code in Github for ink rasterisation, CER computation and expression tokenization. The Visual-Language Model PaLI used in the experiments is non-open-sourced, so full results from Table 6 cannot be reproduced.

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify the training details like number of parameters, learning rate, dropout, training data, etc. for the models – CTC transformer and PaLI in Section 4.2.

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In Section 4.2 we report average and variance of three training runs with different shuffling of training data.

   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Section 4.2 we provide the total number of TPU days it took to run our experiments.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes] We used pretrained PaLI model for finetuning and cited [28].

   (b) Did you mention the license of the assets? [N/A] As the PaLI model is non-open-sourced there is no license for it.

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] URLs to the dataset and code are provided in Section 1.

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We discussed the conditions of data collection in Section 2.1.

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We mention in Section 2.1 that there is no personally identifiable information present in the dataset and offensive content is highly unlikely given the nature of the dataset.

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] We describe the instructions of the data campaigns in Section 2.1 as they are quite simple – to write a rendered expression provided on the screen.

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] The paper does not involve research with human subjects.

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes] We write about the employment of participants in Section 2.1, we don't disclose the exact amount of compensation as it is confidential information.

Appendix

A LaTeX template for label rendering

All the packages and definitions that are required to compile all the normalized and raw labels:

```
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\newcommand{\R}{\mathbb{R}}
\newcommand{\C}{\mathbb{C}}
\newcommand{\Q}{\mathbb{Q}}
\newcommand{\Z}{\mathbb{Z}}
\newcommand{\N}{\mathbb{N}}
```

B Acquisition of LaTeX Expressions

The labels we publish mostly come from Wikipedia (95% of all samples have labels from Wikipedia). A small part were generated, to cover deeply nested fractions, number-heavy expressions, and isolated letters with nested superscripts and subscripts, which are rare in Wikipedia.

The extraction process from Wikipedia followed these steps:

- download an XML Wikipedia dump which provides Wikipedia's raw textual content. `enwiki-20231101-pages-articles.xml` was used for synthetic samples, older dumps for human-written ones
- extract all LaTeX expressions from that file. This gives the list of all mathematical expressions in LaTeX notation from Wikipedia
- keep those which could be compiled using the packages listed in Appendix A. Wikipedia contains a significant number of expressions that are not accepted by the LaTeX compiler, because of syntax errors or other reasons
- keep only those which can be processed by our normalizer which only supports a subset of all LaTeX commands and structures

For expressions used for synthesis, the following extra steps were performed:

- keep only the expressions whose normalized form contains more than a single LaTeX token. Example: `\alpha` is rejected but `\alpha^{2}` is kept. This step is useful to eliminate trivial expressions that wouldn't add any useful information
- de-duplicate expressions based on their normalized form. e.g. `\frac12` and `\frac{1}{2}` normalize to the same thing, we kept only one of them in raw form
- restrict the list of expressions to the same set of tokens used in the `train` split: if the normalized form of an expression contained at least one token that was not also present somewhere in `train`, it was discarded.

C Postprocessing of MathWriting dataset

We applied no postprocessing to the collected inks other than dropping entirely those that were completely unreadable or had stray marks. Inks are provided in their original form, as they were recorded with the collection app. What we *did not do* was to discard samples that were very hard to read or ambiguous, because we believe this type of sample to be essential in training a high-quality model.

Some cleanup was performed on the labels (ground truths). The goal was to make the dataset better suited to training an ML model, and eliminate unavoidable issues that occurred during the collection. After training some initial models, we manually reviewed samples for which they performed poorly. This helped identify a lot of unusable inks (near-blank, lots of stray strokes, scribbles, etc.), and a lot of ink/label discrepancies. A fairly common occurrence was a contributor forgetting to copy part of the prompted expression. We adjusted the label to what was actually written unless the ink contained a partially-drawn symbol, in which case we discarded the sample entirely. In this process we eliminated or fixed around 20k samples.

The most important postprocessing step was to normalize the labels: there are many different ways to write a mathematical expression in LaTeX format that will render to images that are equivalent in handwritten form. We applied a series of transformations to eliminate as many variations as possible while retaining the same semantic. This greatly improved the performance of models and made their evaluation more precise. We publish both the normalized and raw (unnormalized) labels, to enable people to experiment with other normalization procedures.

This normalization is similar to what [10] did, but pushed further because of the specifics of handwritten MEs. See Section 2.4 for more detail.

D Dataset split

The `valid` and `test` splits are the result of multiple operations performed between 2016 and 2019. The first split operation, performed on the data available in 2016, was based on the contributor id: any given contributor's samples would not appear in more than one split (either `train`, `valid`, `test`). This is common practice for handwriting recognition systems, to test how the recognizer performs on unseen handwriting styles.

Experiments then showed that a more important factor than the handwriting style was whether the *label* had already been seen during training. Subsequent data collection campaigns focused on increasing label variety, and new samples were added to `valid` and `test`, this time split by label: a given normalized mathematical expression would not appear in more than one split.

E Label Normalization

E.1 Syntactic Variations

There are several ways to change a LaTeX string without changing the rendered output significantly. The normalization we implemented does the following:

- all unnecessary space is dropped
- all command arguments are consistently put in curly braces
- superscripts and subscripts are put in curly braces and their order is normalized. e.g. `a^2_1` becomes `a_{1}^{2}`.
- redundant braces are dropped
- infix commands are replaced by their prefix versions. e.g. `\over` is replaced by `\frac`
- a lot of synonyms are collapsed. e.g. `\le` and `\leq`, `\longrightarrow` and `\rightarrow`, etc. Some of the synonyms are only synonyms in handwriting. For example `\star` ($\star$) and $*$ are different in print (5-prong and 6-prong stars), but the difference was not expressed in handwriting by our contributors.
- functions commands like `\sin` are replaced by the sequence of letters of the function name (e.g. `\sin` is replaced by `sin`). This reduces the output vocabulary, and eliminates a source of confusion because we found that LaTeX expressions from Wikipedia come with a mix of function commands and sequences of letters.
- expansion of abbreviations. e.g. `\cdots`, `\ldots`, etc. have been replaced by the corresponding sequence of characters.
- matrix environments are normalized to use only the 'matrix' environment surrounded by the proper delimiters like brackets or parentheses.
- `\binom` is turned into a 2-element column matrix. Expressions from Wikipedia did not use those consistently, so we made the choice to normalize `\binom` away.

Sub/superscript are put in braces, `\over` is replaced by `\frac`

Raw:	`\overline{hu^2}+{1 \over 2}{k_{ap}g_zh^2}`
Normalized:	`\overline{hu^{2}}+\frac{1}{2}k_{ap}g_{z}h^{2}`

Subscripts are put before superscripts, extra space is dropped

Raw:	`\int^a_{-a}f(x) dx=0`
Normalized:	`\int_{-a}^{a}f(x)dx=0`

Single quotes are replaced by a superscript

Raw:	`f'(\overline x)`
Normalized:	`f^{\prime}(\overline{x})`

Text formatting commands like `\rm` are dropped

Raw:	`~A_{0}=\frac{ND}{\sigma_{\rm as}+\sigma_{\rm es}}~`
Normalized:	`A_{0}=\frac{ND}{\sigma_{as}+\sigma_{es}}`

Matrix environments with delimiters like `bmatrix` are replaced by `matrix` surrounded by delimiters
Commands like `\cos` are replaced by the series of letters

Raw:	`\begin{bmatrix} -\sin t \\ \cos t \end{bmatrix}`
Normalized:	`[\begin{matrix}-sint\\ cost\end{matrix}]`

Delimiter size modifiers like `\big` are dropped

Raw:	`\big(\tfrac{a}{N}\big)`
Normalized:	`(\frac{a}{N})`

Figure 10: Examples of expression normalization. See Section 2.4 for details.

E.2 Differences Between Print And Handwriting

The following characteristics can not be represented in handwriting and have been normalized away:

- color
- accurate spacing: e.g. `~`, `\quad`.
- font style and size: e.g. `\mathrm`, `\mathit`, `\mathbf`, `\scriptstyle`.

There are others that can be represented in handwriting, but that are not consistent enough in MathWriting to be preserved:

- font families: Fraktur, Calligraphic. In practice, only Blackboard (`\mathbb`) has been written consistently enough by contributors that we were able to keep it: `\mathcal` and `\mathfrak` are dropped.
- some variations like `\rightarrow` $\rightarrow$ and `\longrightarrow` $\longrightarrow$.
- some character variations. e.g. `\varrho`, `\varepsilon`
- size modifiers like `\left`, `\right`, `\big`. Similarly, variable-width diacritics like `\widehat`.

F Additional dataset statistics

In this section we show additional graphs that illustrate dataset statistics that are described in Section 3. The frequencies of normalized LaTeX expressions are presented in Figure 11. Figure 12 illustrates the

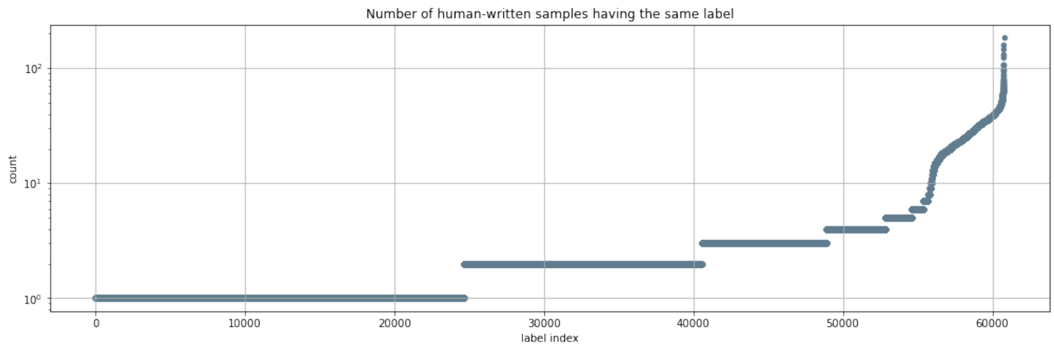


Figure 11: Counts of inks corresponding to the same normalized expression, ordered by increasing count. Each position on the horizontal corresponds to a unique normalized expression. Almost 5k unique expressions have been written 10 times or more by contributors.

distribution of sampling rates within human-written data. Results of resampling points in time are presented in Figure 13.

with time resampling are given on Figure 13.

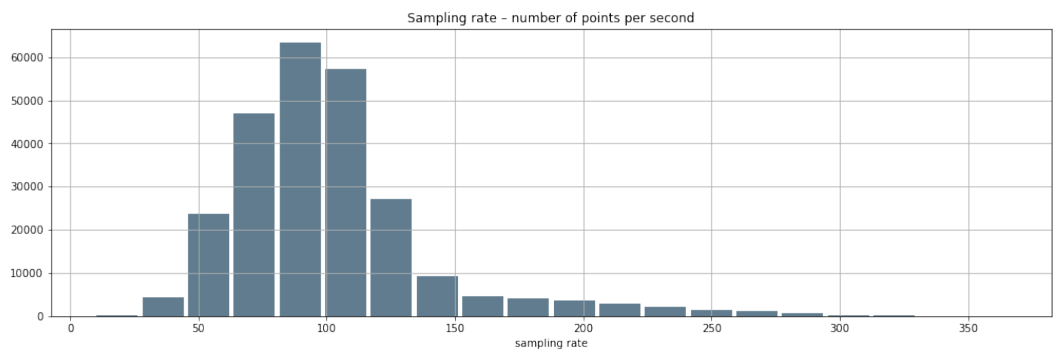


Figure 12: Histogram of sampling rates in human-written data of MathWriting dataset.

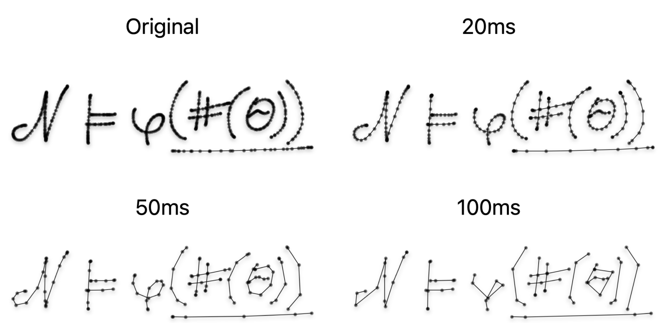


Figure 13: Examples of time resampling with different time periods. Larger periods result in shorter sequences of points.

G Variety of Writing Styles

In this section we provide additional examples of differences in the writing order of fractions – Figure 14. These examples show that MathWriting dataset contains a variety of writing styles.

Device type	Ink
Google PixelBook	51k
Google Nexus 5X	28k
Coolpad Mega 2.5D	14k
OnePlus One	13k
Google Nexus 5	11k
Google Nexus 6	11k
Google Nexus 6P	11k
Coolpad Mega 3	8k
LG Optimus L9	8k
Galaxy Grand Duos	7k
Google Pixel XL	6k
Samsung Galaxy S7	5k

Table 7: Top-12 devices used, with the number of samples obtained from each device. The bias towards Google devices simply reflects the conditions in which inks were collected.

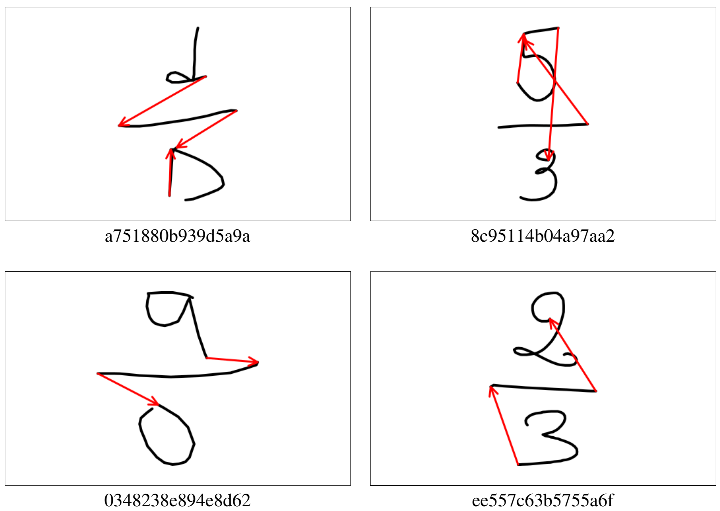


Figure 14: Examples of various writing orders found in the training set. Red arrows show the movement of the pen between strokes. Top left: most common writing order (top-down, fraction bar drawn left-to-right), top right: fraction bar written first, bottom left: fraction bar drawn right-to-left, bottom right: fraction written bottom-up.

H Sources of Noise

The result of any task performed by humans will contain mistakes, and MathWriting is no exception. We've done our best to remove most of the mistakes, but we know that some remain.

Stray strokes These do not carry any meaning and should be ignored by any recognizer. Since they also appear in real applications, there could be some benefit in having some in the dataset to teach the model about them. That said, it being usually easier to add noise rather than to remove it, we made the choice of discarding as many inks containing stray strokes as possible. Not all inks with stray strokes have been found and removed though (e.g. `train/9e64be65cb874902.inkml` that was discovered post-publication). The fraction of inks containing stray strokes is significantly lower than 1%, and should not be an issue for training a model.

Incorrect ground truth Contributors did not always copy the prompt perfectly, leading to a variety of differences. In most of the cases we spotted, we were able to fix the label to match what had actually been written. A short manual review once the dataset was in its final state showed the rate of incorrect ground truth to be between 1% and 2%. Most of the mistakes are very minor, usually a single token added, missing or incorrect. Errors here also come from ambiguities or misuse of the LaTeX notation: expressions coming from Wikipedia contain some misuse like using `\Sigma` where `\sum` was more appropriate, `\triangle` instead of `\Delta`, `\triangledown` instead of `\nabla`, `\begin{matrix}\end{matrix}` instead of `\binom`, and also some handwriting-specific ambiguities like `\dagger` vs `\top` vs `T`. There are also some instances where reference numbers or extra punctuation are included.

Aggressive normalization While the above sources of noise are unavoidable, normalization is a postprocessing operation that can in theory be tweaked to perfection. In practice, it's a compromise between reducing accidental ambiguities (i.e. removing synonyms), and removing information. Examples: we made the choice of treating `\binom` as a synonym for a 2-element matrix. While it does improve recognition accuracy by making the problem easier, it also moves the burden of distinguishing between the two cases to downstream steps in the recognition pipeline. Similar things can be said about removing all commands that indicate that their content is text instead of math (e.g. `\mbox`), dropping size modifiers, rewriting function commands (e.g. `\sin`, `\cos`), etc. Using a different normalization could prove beneficial depending on the context the recognizer is used in practice. However, for the purpose of a benchmark any reasonable compromise is adequate.

I Tokenization Code

Python code used in this work to tokenize LaTeX mathematical expressions.

```
import re

_COMMAND_RE = re.compile(
  r'\\(mathbb{[a-zA-Z]}|begin{[a-z]+}|end{[a-z]+}|operatorname\*|[a-zA-Z]+|.)')

def tokenize_expression(s: str) -> list[str]:
  tokens = []
  while s:
    if s[0] == '\\':
      tokens.append(_COMMAND_RE.match(s).group(0))
    else:
      tokens.append(s[0])

    s = s[len(tokens[-1]):]

  return tokens
```

J Tokens

Using the above code to compute tokens, the set of all samples in the dataset (human-written, synthetic, from all splits) contain the following after normalization:

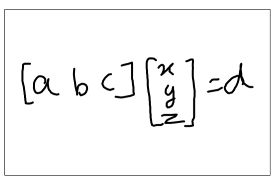
- Syntactic tokens: _ ^{ } & \\ space
- Latin letters and numbers: a-z A-Z 0-9
- Blackboard capital letters `\mathbb{A}`-`\mathbb{Z}` `\mathbb`
- Latin punctuation and symbols: , ; : ! ? . ( ) [ ] \{ \} * / + - _ \& \# \% | `\backslash`
- Greek letters: `\alpha \beta \delta \Delta \epsilon \eta \chi \gamma \Gamma \iota \kappa \lambda \Lambda \nu \mu \omega \Omega \phi \Phi \pi \Pi \psi \Psi \rho \sigma \Sigma \tau \theta \Theta \upsilon \Upsilon \varphi \varpi \varsigma \vartheta \xi \Xi \zeta`
- Mathematical constructs: `\frac \sqrt \prod \sum \iint \int \oint`
- Diacritics and modifiers - Note the absence of the single-quote character, which is normalized to `^{\prime}`:
  `\hat \tilde \vec \overline \underline \prime \dot \not`
- Matrix environment: `\begin{matrix} \end{matrix}`

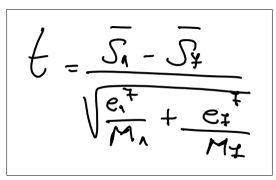
- Delimiters: \langle \rangle \lceil \rceil \lfloor \rfloor \|
- Comparisons: \ge \gg \le \ll <>
- Equality, approximations: = \approx \cong \equiv \ne \propto \sim \simeq
- Set theory: \in \ni \notin \sqsubseteq \subset \subseteq \subsetneq \supset \supseteq \emptyset
- Operators: \times \bigcap \bigcirc \bigcup \bigoplus \bigvee \bigwedge \cap \cup \div \mp \odot \ominus \oplus \otimes \pm \vee \wedge
- Arrows: \hookrightarrow \leftarrow \leftrightarrow \Leftrightarrow \longrightarrow \mapsto \rightarrow \Rightarrow \rightleftharpoons \iff
- Dots: \bullet \cdot \circ
- Other symbols: \aleph \angle \dagger \exists \forall \hbar \infty \models \nabla \neg \partial \perp \top \triangle \triangleleft \triangleq \vdash \Vdash \vdots

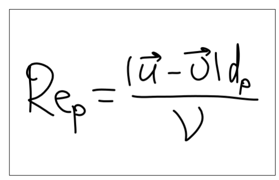
K Examples of inks

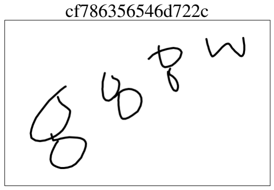
This section shows a few examples of rendered inks, so that the reader can get a feel for the kind of data that is in MathWriting. All samples are from the training set. They have been manually picked to show a variety of sizes, characters and structures.

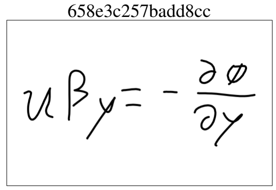
K.1 Human-Written Samples

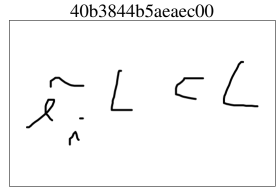

cf786356546d722c

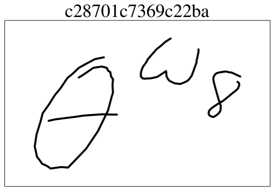

658e3c257badd8cc

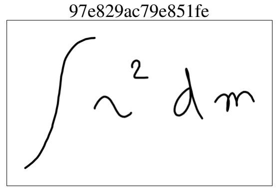

40b3844b5aeaec00

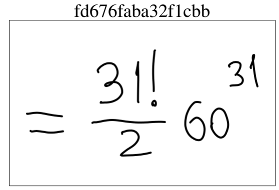

c28701c7369c22ba

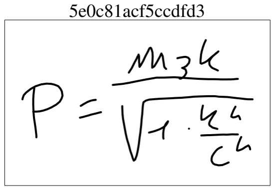

97e829ac79e851fe

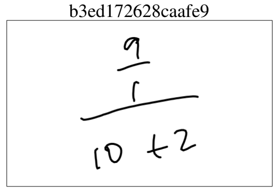

fd676faba32f1cbb

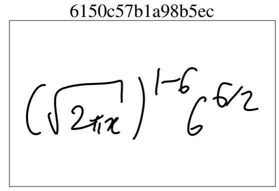

5e0c81acf5ccdfd3

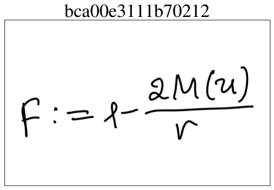

b3ed172628caafe9

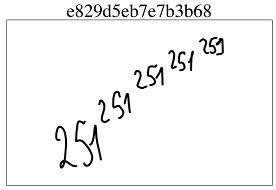

6150c57b1a98b5ec

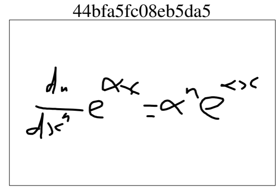

bca00e3111b70212


e829d5eb7e7b3b68


44bfa5fc08eb5da5

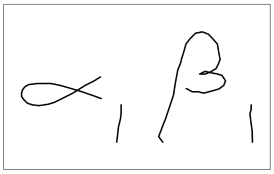

d233aaf208bd7568

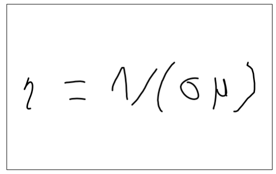

ee6f0bfd294aa209

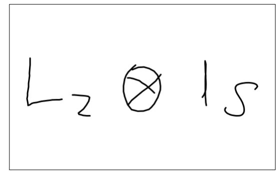

6579f917f0ba236b

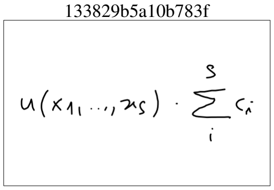
629	$v_p = \sqrt{\frac{\mu(1+e)}{a(1-e)}}$ cea67d239b9f8884	$\frac{\frac{\sqrt{6}}{10} - 456}{\frac{\sqrt{6}}{61} / 7}$ bb4bca53d0e6336d	$\binom{n}{k}$ b14bca3fc2d2819a
630	$l = \sqrt{1 + [f'(x)]^2}$ 2409d2feaa79b9d7	$\frac{\partial u}{\partial \bar{z}} - iz\frac{\partial u}{\partial t} = \varphi'(t$ 51486ff88b789d6d	$d \propto \sqrt{2 \cdot \lambda \cdot R \cdot h}$ 02229a0c174d8dbe
631	$\tilde{C}_7$ 478e10a15203fa3a	W_Z^{ψ} ccb15825579a096b	$\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$ 7a4b95285de0caf0
632	$A = (A \cup B) - B$ ac25b4d053596ded	$\int_{-\infty}^{x_1}$ 7d597c52bf8bdd1e	$\left(\frac{386}{\sqrt{387}}\right)^6 \cdot 363^9 + 404$ acc5f6620fad1ce8
633	$t \equiv \sqrt{1 - 1/\xi^2}$ 88c3551d373c72e5	$\Theta^m{}_b := S^m{}_{ab}\Theta^a$ 5009d32d32f80324	$\forall b_1, \dots, b_n \notin T(s)$ 068de3aad90c403c
634	$\sum_{n=1}^{\infty} \frac{1}{\lvert z_n \rvert^{p+1}}$ a3cf115524f0c55b	$\bar{\xi} = 0, \overline{\xi_i \xi_j} = \frac{1}{3}\bar{\xi}^2 \delta_{ij}$ 355f5df56a16913a	$\theta = n \times 137.508^\circ$ d9b2ce7aa3495888
635	$\overline{\frac{\partial(\overline{r_h} \cdot r_h)}{\partial x_h}} = 0$ adceb80fdadf9f6e	$264 - 175 - 365 - 445$ 25892f7caeac8c36	$(P_j + iQ_j)\Delta^{-1/2}$ 41e1261a951c6f33
636	$F_{out} d_{out} = \eta F_{in} d_{in}$ 02a7f7f172671fb4	$\deg_p f = \deg_p g$ 0d848d4b170d36b9	$2 n \varepsilon \frac{d^3 p\, d^3 r}{h^3}$ 2adc4f10d42b641c
637		$V_e = c\sqrt{2\eta - \eta^2}$ 408f904038dcbba0	

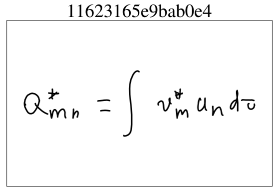
K.2 Synthetic Samples: Expressions from Wikipedia

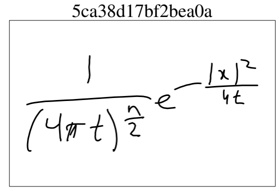

133829b5a10b783f

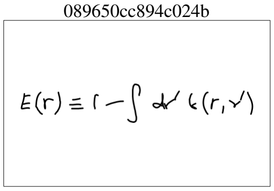

11623165e9bab0e4

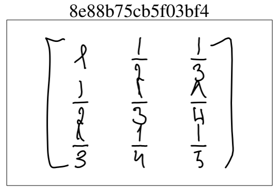

5ca38d17bf2bea0a

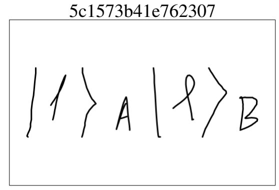

089650cc894c024b

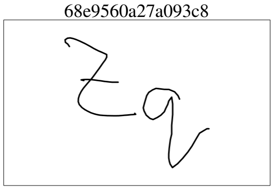

8e88b75cb5f03bf4

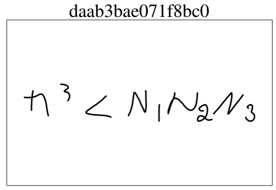

5c1573b41e762307

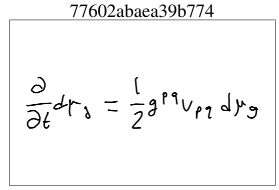

68e9560a27a093c8

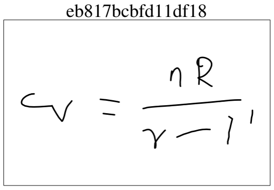

daab3bae071f8bc0

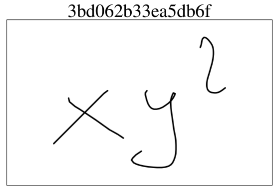

77602abaea39b774

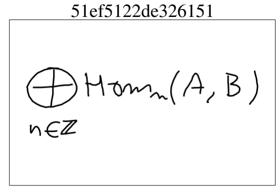

eb817bcbfd11df18

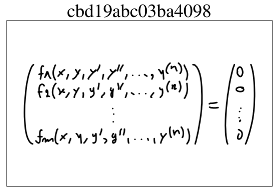

3bd062b33ea5db6f

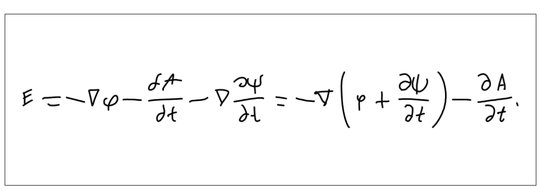

51ef5122de326151

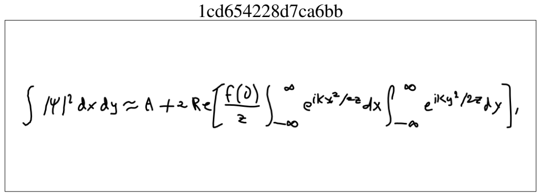

f2a613fc323df342


cbd19abc03ba4098


1879aa5c882b445d


094c7e52a3f0934d


1cd654228d7ca6bb


fc00050933165b70

647 $$A^{(1)} = \begin{pmatrix} 4 & 2 & 1 \\ 0 & 6 & 8.5 \\ 0 & 5 & \frac{22}{3} \end{pmatrix}, \quad P^{(1)} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

60553a301aaf84a3

648 $$d\Phi_n(P; p_1, p_2, \ldots, p_n) = (2\pi)^4 \delta^4 \left(P - \sum_{i=1}^{n} p_i \right) \prod_{i=1}^{n} \frac{d^3 p_i}{2(2\pi)^3 E_i}.$$

b2aeaf7a0fd30ed6

649 $$y = -(g/2)t^2 + Vt \sin\alpha + h. \qquad (2)$$

254dbc2b3843dcf8

650 $$ds^2 = (\Lambda \cos \phi)^2 \, d\lambda^2 + M^2 d\phi^2$$

0772aeaac09d3415

651 $$\Gamma = \begin{pmatrix} \frac{\partial A}{\partial x} & \frac{\partial A}{\partial y} \\ \frac{\partial A}{\partial y} & -\frac{\partial A}{\partial x} \end{pmatrix} + \frac{1}{2\pi\nu} e^{\lambda x} K_0(\lambda r) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \Pi = \frac{\rho}{2\pi} \nabla(\ln r)$$

20b4ebf292cfa8d1

652 $$f(x_1, x_2, x_3, \ldots, x_n; u(x_1, x_2, x_3, \ldots, x_n); D^1(u), D^2(u), D^3(u), \ldots, D^m(u)) = 0$$

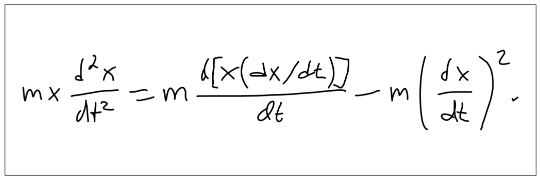
96613ae167f35f8a

653 $$+ \frac{\sqrt[3]{\frac{2(\sqrt{3}-1)\sqrt{5+\sqrt{5}} - (\sqrt{5}-1)(\sqrt{2}+\sqrt{6})}{2}} - \left(2\sqrt{6+\sqrt{3}+\sqrt{15}+\sqrt{10-2\sqrt{5}}}\right)i}{4}$$

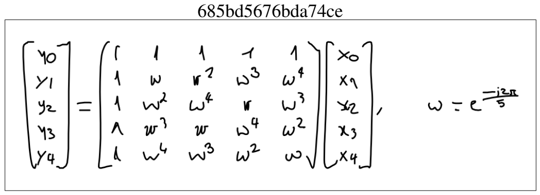
4b1f5165e3698343

654 $$c_{solid,p} = \sqrt{\frac{K + \frac{4}{3}G}{\rho}} = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}},$$

6f86778996d5f514

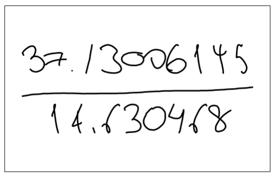


685bd5676bda74ce

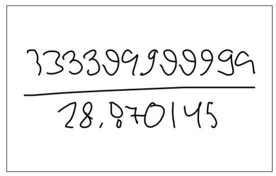


b79dfccd7f5b5cb4

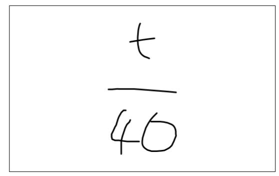
K.2.1 Synthetic Samples: Generated Fractions



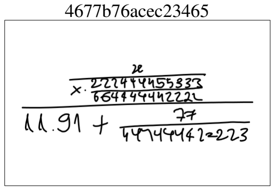
4677b76acec23465



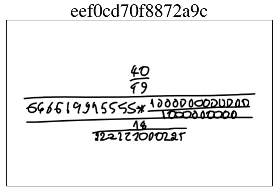
eef0cd70f8872a9c



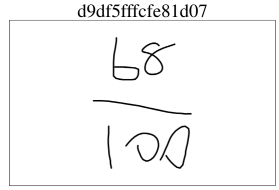
d9df5fffcfe81d07



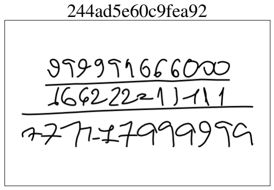
244ad5e60c9fea92



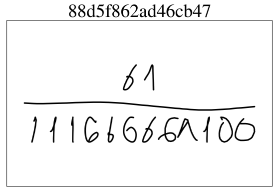
88d5f862ad46cb47



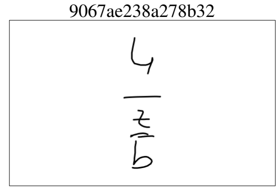
9067ae238a278b32



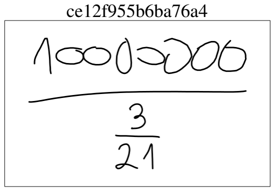
ce12f955b6ba76a4



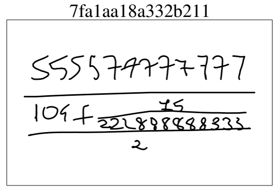
7fa1aa18a332b211



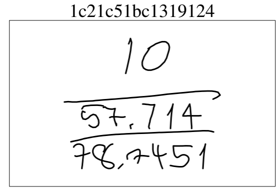
1c21c51bc1319124



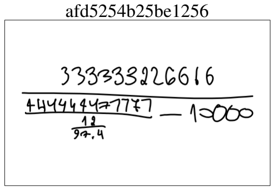
afd5254b25be1256



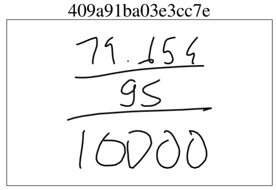
409a91ba03e3cc7e



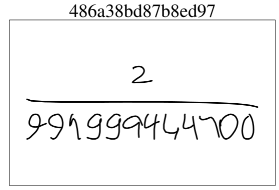
486a38bd87b8ed97



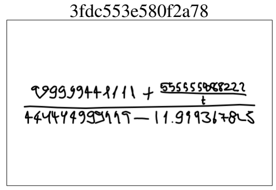
05efec2565d6726e



3fdc553e580f2a78



b17c3206c2d610b9



9df33845897752ea