

12.06.2026 – FS 2026

Synthetic Exam-Style Handwriting Datasets with Embedded Mathematical Derivations

Designing and Benchmarking a Realistic Generation Pipeline

AUTHORS: Nicola Dall'Acqua
Sylvester Homberger

SUPERVISOR: Prof. Dr. Mitra Purandare

INDUSTRY PARTNER: Dr. Gerd Kortemeyer

EXAMINER: Dr. Katsiaryna Mirylenka

PROOFREADER: Prof. Dr. Daniel Politze

Abstract

This bachelor thesis addresses the challenge of generating realistic handwritten mathematical exam-style documents to train ETH Zurich’s Apertus model on handwriting recognition. Expected data consists of handwritten samples containing mathematical derivations paired with textual explanations, similar to student-written solutions. Since no existing dataset adequately covers this combination, ETH Zurich developed a synthetic data generation pipeline [1]. The existing pipeline is deterministic: it generates LaTeX-based ground truths and processes them to create realistic exam-style handwritten images. The deterministic nature assures alignment between ground truth and generated images. However, it also creates limitations concerning rigidity, uniformity, realism and a lack of variability.

Furthermore, no generally established metric exists to evaluate realism of synthetic handwriting. This makes it difficult to assess if Apertus, once trained on synthetic handwriting, can generalize to real handwritten content.

The goal of this work is twofold: first, to investigate the viability of using generative AI to improve the diversity and realism of generated samples and second, to develop a method for evaluating realism of synthetic handwriting. Since generative approaches break the deterministic link between ground truth and output, we introduce a Model-Judge framework to filter samples containing inconsistencies. As a secondary aim, deterministic improvements were made to the pipeline. Particularly: a) the addition of problem statements to the generated handwritten problem solutions, b) use of classical image processing to address line rigidity and c) a demo-client to demonstrate pipeline functionality.

We evaluate the use of generative AI for synthetic handwriting generation, focusing on open- weight models such as FLUX2.dev and Qwen-Image. Our evaluation demonstrates the feasibility of the approach. Due to licensing requirements aligned with Apertus’s philosophy of AI sovereignty, only open models with permissive licenses may be used. Generative AI is applied in two ways: i) augmentation of pipeline outputs and ii) generation of handwritten fonts. Experiments demonstrate that generative AI can address several limitations of the deterministic pipeline, particularly by increasing data variability. However it is, in practice, not yet ready to be integrated into the pipeline. Models are capable of resolving limitations but have large variability in their output-quality. To mitigate this issue, we investigate a Model-Judge framework, consistently achieving False Positive Rates < 0.05 and True Positive Rates > 0.85 . Thus, the framework proves effective in maintaining content consistency and improving output reliability.

Furthermore, we propose a human-centered evaluation procedure based on a Turing-style discrimination task to assess realism of synthetic handwriting. To validate this procedure, we conduct a pilot study in collaboration with ETH Zurich’s Decision Science Laboratory. Participants compared synthetic handwriting samples generated by the pipeline with genuine human handwriting. Three difference hypothesis tests, with p -values $p_1 = 1.88 \cdot 10^{-45}$, $p_2 = 7.76 \cdot 10^{-56}$ and $p_3 = 1.02 \cdot 10^{-75}$, suggest that participants can reliably distinguish current synthetic samples and genuine handwriting, providing evidence of a measurable realism gap. Therefore, the findings confirm the sensitivity of the proposed evaluation procedure for assessing progress toward more realistic handwriting generation.

Table of Contents

Abstract	1
Table of Contents	2
List of Figures	4
List of Tables	5
1 Introduction	6
1.1 Background	6
1.2 Existing Work	8
1.3 Research Question	8
1.4 Scope	8
1.5 License Constraints	9
2 Handwriting Generation	10
2.1 Pipeline Improvements	10
2.1.1 Typed Problem Statements	10
2.1.2 Realism Improvements	12
2.1.3 Comparison	13
2.2 Evaluation of Generative AI	17
2.2.1 Document Augmentation	17
2.2.2 Document Subsection Augmentation	22
2.2.3 Font Generation	28
2.2.4 Model Judge	30
2.3 Other Handwriting Generation Approaches	34
2.3.1 AI-Based Methods	34
2.3.2 Non-AI Based Methods	34
2.4 Demo Client	35
3 Realism Evaluation Procedure	37
3.1 Methodology	37
3.1.1 Group 1	38
3.1.2 Group 2	38
3.1.3 Interpretation	39
3.2 Experiment setup	40
3.2.1 Test Samples	40
3.2.2 Platform	41
3.3 Results	44
3.3.1 Quantitative Analysis	44
3.3.2 Qualitative Analysis	47
4 Conclusion	50
4.1 Evaluation of Generative AI	50
4.2 Model Judge	50
4.3 Realism Assessment Experiment	51
4.4 Deterministic Pipeline Improvements	51
5 Outlook	52
References	53

Glossary	55
-----------------------	-----------

Appendices 58

A Infrastructure Setup	59
-------------------------------------	-----------

B Technical Background	60
-------------------------------------	-----------

B.1 Diffusion-Style Models	60
----------------------------------	----

B.2 Vision-Language Models	61
----------------------------------	----

C Existing Pipeline	62
----------------------------------	-----------

C.1 Data Generation	62
---------------------------	----

C.2 Font Creation	62
-------------------------	----

C.3 Pipeline Visualization	63
----------------------------------	----

D Prompts	64
------------------------	-----------

D.1 Pipeline Generation Prompt	64
--------------------------------------	----

D.2 Document Augmentation Prompt	65
--	----

D.3 Subsection Augmentation Prompts	65
---	----

D.4 Font Generation Prompt	66
----------------------------------	----

D.5 Model Judge Prompt	67
------------------------------	----

E Realism Test Word Clouds	69
---	-----------

E.1 Group 1	69
-------------------	----

E.2 Group 2	71
-------------------	----

F List of Aids	73
-----------------------------	-----------

F.1 AI Declaration	73
--------------------------	----

G Code	74
---------------------	-----------

G.1 Requirements	74
------------------------	----

G.2 Usage	74
-----------------	----

G.3 Replicating Generative Approaches	74
---	----

G.3.1 Document Augmentation	74
-----------------------------------	----

G.3.2 Font Generation	74
-----------------------------	----

G.3.3 Model Judge	74
-------------------------	----

G.4 Replicating Realism Benchmark	74
---	----

H Full Scope	76
---------------------------	-----------

H.1 Supervisors	76
-----------------------	----

H.2 Students	76
--------------------	----

H.3 Context	76
-------------------	----

H.4 Minimal Tasks	77
-------------------------	----

H.5 Deliverables	78
------------------------	----

H.6 Grading	78
-------------------	----

H.7 Roadmap	78
-------------------	----

H.8 Further Information	79
-------------------------------	----

List of Figures

Figure 1	Background - Existing Pipeline	7
Figure 2	Existing pipeline - Sample 1	7
Figure 3	Existing pipeline - Sample 2	7
Figure 4	Pipeline improvements - Typed problem statement	12
Figure 5	Pipeline improvements - Wobbly text lines	12
Figure 6	Pipeline improvements - Sample 1 prior to increments	13
Figure 7	Pipeline improvements - Sample 2 prior to increments	14
Figure 8	Pipeline improvements - Sample 1 after increments	15
Figure 9	Pipeline improvements - Sample 2 after increments	16
Figure 10	Document augmentation - Pipeline improvement UML	17
Figure 11	Document augmentation - Input image	18
Figure 12	Document augmentation - Output of FLUX.2-dev	19
Figure 13	Document augmentation - Output of Qwen-Image-Edit	20
Figure 14	Document augmentation - Output of gpt-image-1.5	21
Figure 15	Document augmentation - Output of <i>Nano Banana Pro</i>	22
Figure 16	Subsection augmentation - Pipeline improvement UML	23
Figure 17	Styleguide for generative AI approaches	25
Figure 18	Document subsection augmentation - Impact of grid search dimensions	26
Figure 19	Document subsection augmentation - Output of <i>Nano Banana Pro</i>	27
Figure 20	Font Generation - Pipeline improvement UML	28
Figure 21	Font generation - FLUX.2-dev examples	29
Figure 22	Font generation - Qwen-Image-Edit examples	29
Figure 23	Font generation - <i>Nano Banana Pro</i> examples	30
Figure 24	Overview of handwriting generation methods, adapted from Elarian et. al [2, Sec. 3] ...	35
Figure 25	Demo-client overview	36
Figure 26	Realism test - Generated test sample	40
Figure 27	Realism test - Real test sample	41
Figure 28	Realism test - Rating panels (Group 1: Top and Group 2: Bottom)	43
Figure 29	Realism test - Group 1: time vs. rating scatter plot	45
Figure 30	Realism test - Group 1: rating distributions	45
Figure 31	Realism test - Group 1: Kolmogorov-Smirnov-Test	46
Figure 32	Realism test - Group 1: semantic cluster distribution by label	47
Figure 33	Realism test - Group 1: semantic cluster distribution heatmap	48
Figure 34	Realism test - Group 2: semantic cluster distribution by correct/incorrect ratings	49
Figure 35	Font creation template	63
Figure 36	Pipeline overview	63
Figure 37	Realism test group 1 - clusters Uniformity, Structure and Legibility	69
Figure 38	Realism test group 1 - clusters Size, Uncertainty and Realism	70
Figure 39	Realism test group 2 - clusters Uniformity and Structure	71
Figure 40	Realism test group 2 - clusters Legibility and Realism	72
Figure 41	Existing pipeline - Sample 3	76
Figure 42	Thesis roadmap	79

List of Tables

Table 1	Document subsection augmentation - Grid search dimensions	24
Table 2	Document subsection augmentation - Input files in Name x Mode dimensions	24
Table 3	Document subsection augmentation - Result ratings	25
Table 4	Font Generation - Grid search dimensions	28
Table 5	Model Judge - Confusion matrix term definitions	30
Table 6	Model Judge - SMJ1: Qwen3.5-27B	31
Table 7	Model Judge - SMJ2: Qwen3.5-397B-A17B	31
Table 8	Model Judge - SMJ3: Kimi K2.5	32
Table 9	Model Judge - total False Positive Rate derivation	32
Table 10	Model Judge - Example improvement of Multi-Model setup	33
Table 11	Model Judge - MMJ1: Gemma4-31B-it, Qwen3.5-27B and Qwen3.5-397B-A17B	33
Table 12	Model Judge - MMJ2: Kimi K2.5, Qwen3.5-27B and Qwen3.5-397B-A17B	33
Table 13	Model Judge - MMJ3: Gemma4-31B-it, Qwen3.5-27B and Kimi K2.5	34
Table 14	Realism test - Group 1 data overview	44
Table 15	Realism test - Group 2 data overview	46
Table 16	List of aids	73

1 Introduction

Synthetic handwriting in the target format required by Apertus does not exist at the moment. To resolve this, ETH Zürich has developed a *data generation pipeline*. This pipeline is capable of producing the required data, however limitations concerning data realism exist. These are primarily about rigidity, uniformity and a lack of variability.

The goal of this thesis is to evaluate the feasibility of resolving these limitations through the use of generative AI. Furthermore, a method to quantify realism is to be developed.

The full background and context, as well as the limitations are discussed in the following sections. Afterwards, the research question and thesis scope are described. This chapter closes by mentioning important license constraints that must be considered.

1.1 Background

The accuracy and validity of feedback provided by *Ethel*, ETH's virtual teaching assistant, is limited by its ability to recognize and process human handwriting. Ethel requires a *vision-language-model* in its backend, to process student homework, exams and lecture questions. The end goal is to use ETH's *Apertus* model in the backend of Ethel, therefore Apertus must be capable of recognizing and processing human handwriting. As the planned use case of Ethel entails usage in mostly STEM study programmes that are held in various languages, the main challenges of handwriting recognition concern mathematical content and multi-linguality. To prepare Apertus for this challenge, synthetic training data covering these aspects is required.

For this, a *data generation pipeline* able to generate such data was created in an EPFL student project [1], in collaboration with ETH's Apertus team. The basic idea of this data generation pipeline is to start from the ground truth in LaTeX format and then apply various processing steps to create an artifact with a handwritten look. This is done in a deterministic manner, leveraging the LaTeX-compilers extensive features.

As an overview the existing pipeline is shown in Figure 1. Importantly, the generation of handwriting happens in the right-hand part and begins at the top, the left-hand part is a supporting process, with which handwritten fonts may be produced. The full description of the data generation pipeline can be found in Appendix C.

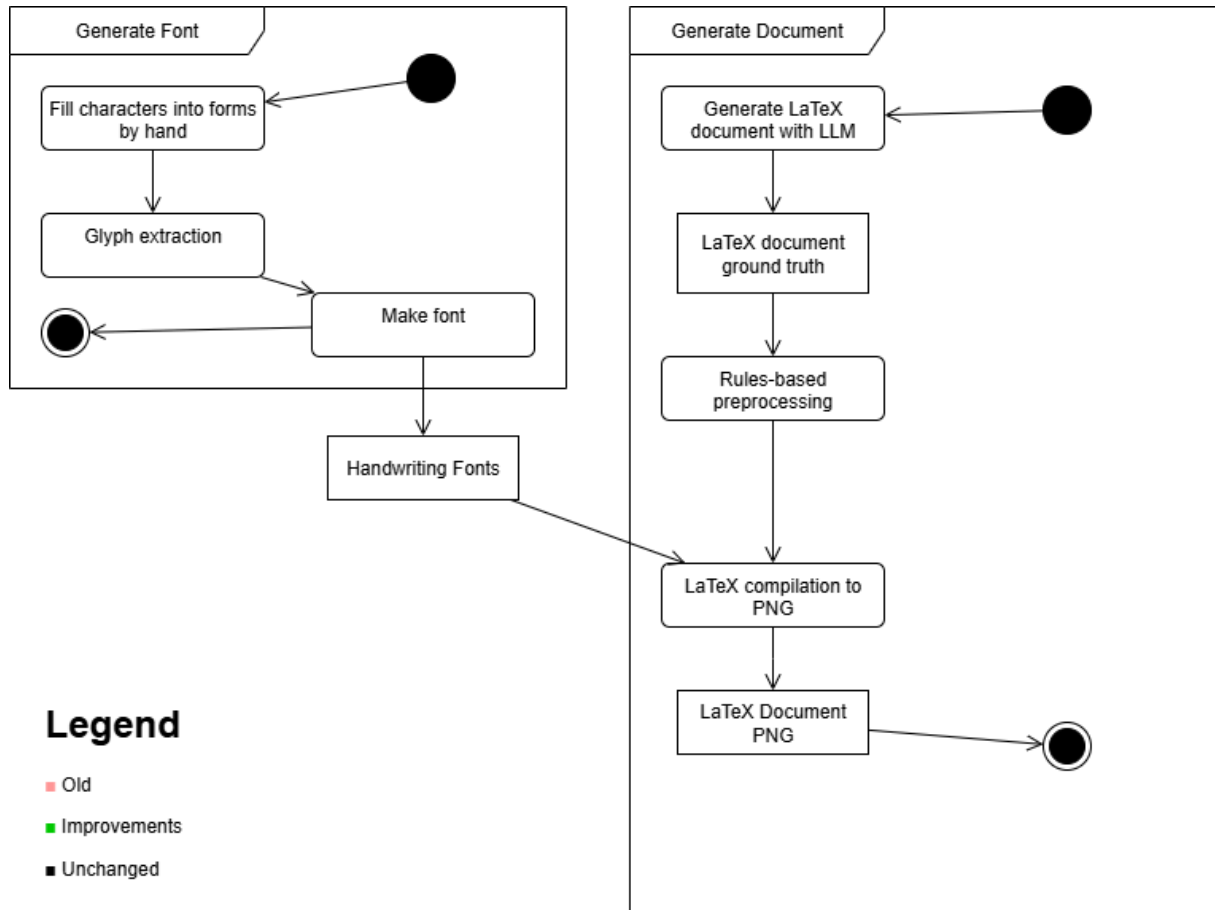


Figure 1: Background - Existing Pipeline

Examples of artifacts produced with the existing pipeline may be found in Figure 2 and Figure 3.

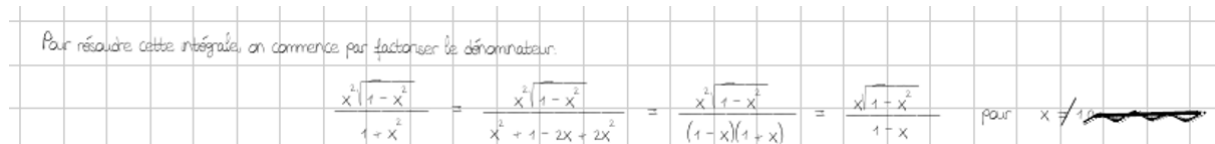


Figure 2: Existing pipeline - Sample 1

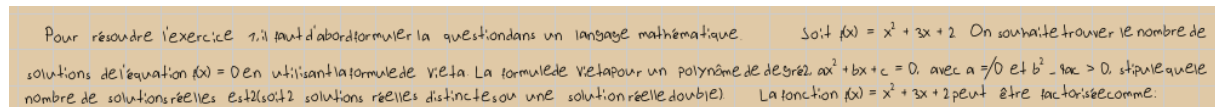


Figure 3: Existing pipeline - Sample 2

The current approach has one main limitation: despite the extensive functionality of the LaTeX-compiler, it is a deterministic tool and therefore creates deterministic outputs. This leads to:

- **Rigid line-formatting:** Lines start and end at the same horizontal offset, are always the same distance apart and follow perfectly straight paths
- **Rigid formula placement:** Formulae are placed either inline or centered on a new line
- **Character placement:** Individual characters are always placed with the same distance from others and have no irregularities. Furthermore one specific character looks exactly the same throughout the entire document, there are no minor variations
- **Annotations:** Annotations that are often seen in handwriting, such as arrows connecting pieces of text, underbrackets, etc. are very difficult to realise using the LaTeX-compiler

Some of these issues can be partially alleviated using LaTeX-commands, but remain present nonetheless. It is, for example, possible to apply a small offset to each character in a word to make letter-spacing more irregular. However it is not possible to apply this to formulae, as placing the necessary LaTeX-commands within *math-mode* renders the document uncompileable.

1.2 Existing Work

Besides this generation pipeline, there are existing, publicly available handwriting datasets that contain mathematical expressions. However, the samples in these datasets consist of individual characters or individual formulae [3], [4]. These formulae are not embedded into a larger, handwritten document and therefore do not hold a position in some larger narrative, like formulae in student exercise solutions do.

There also exist datasets containing handwritten text [5], [6]. These, however, do not contain mathematical content.

The existing data generation pipeline described in Section 1.1, closes this gap by producing data containing both handwritten mathematical expressions and handwritten text, with the formulae embedded into a narrative.

From a broader perspective, there also exist completely different ways to generate handwriting. These are described in Section 2.3.

1.3 Research Question

To address the challenges described in Section 1.1, the feasibility of utilizing *generative AI*, within the data generation pipeline is evaluated. The goal is to break up rigid spacing, slightly modify characters and introducing various formatting variations to alleviate the pipeline’s limitations.

This, however, introduces a new challenge: does model-augmented content still contain the same information as the ground truth? Ensuring this is critical, as otherwise, the data can’t be used to train a model at a later point in time.

Aside from this, a procedure to benchmark the realism of synthetic handwriting is required. In general, no standard method exists [7, p. 9]. Within this report, Diaz et. al summarize several methods that do exist, however most of these are tailored to specific use cases and do not measure general realism. The method to-be-developed shall provide a more general assessment of realism, as opposed to focusing on specific aspects. Ensuring data realism is a crucial step for assuring that a model trained on synthetic handwriting can transfer that skill to real, human handwriting, since successful transfer learning benefits from data similarity. Therefore, a human experiment is designed to answer the question whether synthetic handwriting can be reliably distinguished from real, human handwriting. The experiment is analogous to a *Turing-Test*-style evaluation: participants will review a diverse set of handwritten artifacts, some produced by humans and others generated algorithmically, and will be asked to judge the likely origin of each artifact.

The contribution of our work is two-fold: we conduct a feasibility study for the use of generative AI and we develop a realism assessment procedure.

1.4 Scope

The main requirements were specified in the previous Section 1.3: a) evaluate the feasibility of integrating generative AI into the existing data generation pipeline and b) develop a procedure to assess realism of generated data. Besides those, there are some further aspects to be considered.

First, the new data generation pipeline must be ready to produce large amounts of samples, once experiments around extending and adapting it conclude.

Second, the Apertus team suggested to add typed problem statements to pipeline outputs. As seen in Figure 2 and Figure 3, currently there are no problem statements in computer font that describe the problem to be solved, only the problem solution itself. Since such statements are very prevalent in homework and lecture exercises, adding them to pipeline outputs further increases realism.

Third, a demo-client of the data generation pipeline is to be developed. Considering the flowchart of the pipeline in Appendix C, the necessary changes only concern the first stage on the “Generate Document”-side. As opposed to generating a LaTeX-document using an *LLM*, users submit their own LaTeX-file. These are then passed through the remainder of the pipeline and thus a synthetic handwritten sample of user input is created.

The full thesis scope, as was agreed prior to the start, can be found in Appendix H.

1.5 License Constraints

Because Apertus is a fully open model, which means open weights, open data and open training details, there are license constraints to be considered. “Fully open model” in this context means everything is reproducible, i.e. one could retrain the exact model from scratch and then redistribute it freely. Their philosophy is comparable to that of *Open source* software.

First, any data that will be used for training Apertus, must be licensed under a permissive license such as MIT, Apache 2.0, CC0, CC-BY, ODC-BY among others. Additionally, data must be reproducible through permissively licensed, open code. Reproducible in the sense of handwritten documents does not mean reproducible word-by-word and document-by-document, since this renders any and all AI involved in the process unusable. Reproducible instead means a dataset of similar handwritten documents must be reproducible and it must be possible to see the details of data generation.

Second, developers of existing generative AI models often place clauses in their terms of use, that prohibit the use of these models for training data generation used in competing models. Data and models that do not fulfill these conditions are, therefore, disqualified from use for anything that will be used, in some way, for training Apertus. They may still be used, throughout this project, to conduct feasibility evaluations demonstrating technical possibilities, so long as results are strictly separated.

2 Handwriting Generation

There are several possible ways to generate handwriting, see Section 2.3 for an overview. Given that there already exists a pipeline for handwriting generation (see the flowchart in Appendix C) and the goal being to improve this pipeline, a method that is integrable into the pipeline has to be chosen. Most of the methods described in Section 2.3 generate handwriting from scratch and are therefore not compatible with the existing pipeline, but rather form entirely new approaches. Diffusion-based models, however, are ideal for our use-case, as they can both generate and edit images, which allows for augmentation of outputs of the existing pipeline. Moreover, recent experimental results [8] suggest that diffusion-based approaches are the current state-of-the-art for handwriting generation.

Considering this and the requirements defined in Section 1.4, it was decided to: a) improve the pipeline in purely deterministic manners, i.e. without the use of AI, and b) evaluate the feasibility of integrating diffusion-style models into the pipeline. Both are described in this chapter, alongside further handwriting generation methods and the implementation of the demo-client to demonstrate the pipeline.

2.1 Pipeline Improvements

It is possible to improve the existing pipeline in fully deterministic ways. Two such increments were implemented and are described in this section.

2.1.1 Typed Problem Statements

The addition of typed problem statements, as outlined in Section 1.4, was done directly within the existing pipeline. Specifically, the existing pipeline, described in Appendix C, was extended with the following:

- In step 1, instead of prompting the LLM to generate only the solution to a math exercise, prompt it to first generate the problem statement, and then the solution, all within the same document.
- The LLM is asked to enclose the statement within the following LaTeX-command: `\fontspec{Computer Font A}\setmathsfont{Computer Font A} <statement> }`. This compiles the statement with the specified typed computer font, as opposed to the globally set handwritten font.
- In step 3, the code inserting LaTeX-commands for offsets and rotations was adapted to ignore statements. Since the statements are, per definition, in typed computer font, no irregularities are needed.

See Appendix D.1 for the full LLM-prompt, an example of a produced ground truth at step 2 may be found below:

```
\begin{document}

\setmainfont{Latin Modern Roman}\setmathfont{Latin Modern Math}
\textbf{Exercise 7} \hfill \textit{Total points: 12}

Solve the following problems on polynomial equations and their roots.

\begin{enumerate}
  \item[(a)] Find all real solutions of the equation  $x^3 - 6x^2 + 11x - 6 = 0$ . \hfill (6 points)
  \item[(b)] Given that  $x = 2$  is a root of multiplicity two of the polynomial  $p(x) = x^4 - 6x^3 + ax^2 + bx + 8$ , determine the values of  $a$  and  $b$ . \hfill (6 points)
\end{enumerate}
```

}

Consider part (a): solve $x^3 - 6x^2 + 11x - 6 = 0$.

Try rational roots using Rational Root Theorem: possible candidates are $\pm 1, \pm 2, \pm 3, \pm 6$.

Test $x = 1$:

$1 - 6 + 11 - 6 = 0$, so $x = 1$ is a root.

Factor out $(x - 1)$ via polynomial division or synthetic division:

```
\begin{align*}
x^3 - 6x^2 + 11x - 6 &= (x - 1)(x^2 - 5x + 6) \\
&= (x - 1)(x - 2)(x - 3)
\end{align*}
```

Thus, the real solutions are $x = 1$, $x = 2$, $x = 3$.

Now part (b): given $x = 2$ is a double root of $p(x) = x^4 - 6x^3 + ax^2 + bx + 8$.

Then $(x - 2)^2$ divides $p(x)$, so $p(2) = 0$ and $p'(2) = 0$.

Compute $p(2)$:

$p(2) = 16 - 48 + 4a + 2b + 8 = 0$

$\Rightarrow 4a + 2b = 24$

$\Rightarrow 2a + b = 12$. \hfill (1)

Compute derivative: $p'(x) = 4x^3 - 18x^2 + 2ax + b$

Now $p'(2) = 0$:

$4(8) - 18(4) + 2a(2) + b = 0$

$32 - 72 + 4a + b = 0$

$4a + b = 40$. \hfill (2)

Subtract (1) from (2):

$(4a + b) - (2a + b) = 40 - 12$

$2a = 28 \Rightarrow a = 14$

Substitute into (1): $2(14) + b = 12 \Rightarrow 28 + b = 12 \Rightarrow b = -16$

~~Therefore~~ $a = 14$, $b = -16$

\end{document}

On the second-to-last line, the command `\strikeMistake` is used by the LLM. This is a custom command that strikes through the contents of the command in a way that humans strike through spelling mistakes. After this, the ground truth is split into parts using the standard `\vbox`-command. Furthermore the command `\processtext` is applied to individual words. This is another custom command, which adds small rotations and offsets to each letter of a word.

An example of a problem statement from a finished artifact can be found in Figure 4.

Aufgabe 3 Polynome über Q **12 Punkte**

Gegeben seien die Polynome $f(x) = x^4 + 2x^3 - x - 2$ und $g(x) = x^3 + 3x^2 + 2x$ in $Q[x]$.

- a) Berechnen Sie den größten gemeinsamen Teiler (f, g) mittels des Euklidischen Algorithmus. **6 Punkte**
 b) Bestimmen Sie explizit Polynome $a(x), b(x) \in Q[x]$ mit $(f, g) = a(x)f(x) + b(x)g(x)$. **6 Punkte**

Figure 4: Pipeline improvements - Typed problem statement

2.1.2 Realism Improvements

One of the main issues concerning realism is rigid line formatting. Text lines follow a perfectly straight path, which is a telltale sign of synthetic handwriting. This can be resolved using classical image processing techniques, which is a, comparatively, simple task. Specifically, we used two 2D *Gaussian blurs* to achieve this. The first blur has a Sigma of 40, which mimics high-frequency displacement, whereas the second blur has a Sigma of 250 which mimics a low-frequency component. This means that both local “dents” (i.e. word- or even character-level) and global deformation (i.e. trajectory of a text line or paragraph) are produced. The end result are text lines that are “wobbly” in a human-like way. An example of a local and a global deformation may be found in the images in Figure 5 respectively.

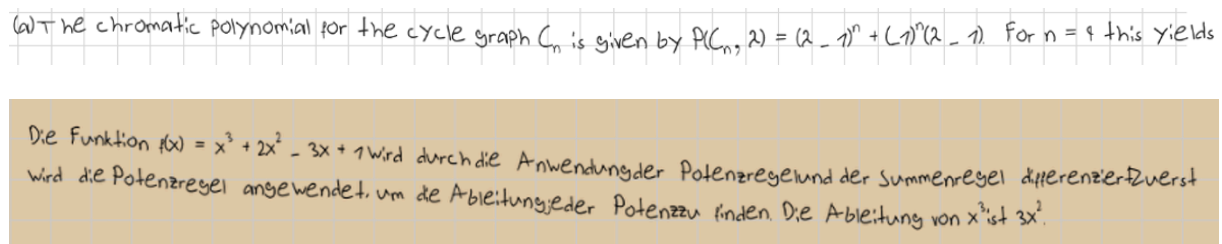


Figure 5: Pipeline improvements - Wobbly text lines

This increment is executed in step 6 of the pipeline (see Appendix C), after the conversion from PDF to PNG.

In addition, some postprocessing effects such as noise and rotation were applied to better emulate a hastily scanned document. This is, however, a minor improvement and does not resolve any of the limitations of the existing pipeline described in Section 1.1.

2.1.3 Comparison

Full outputs of the data generation pipeline prior to the two previously discussed increments may be found in Figure 6 and Figure 7.

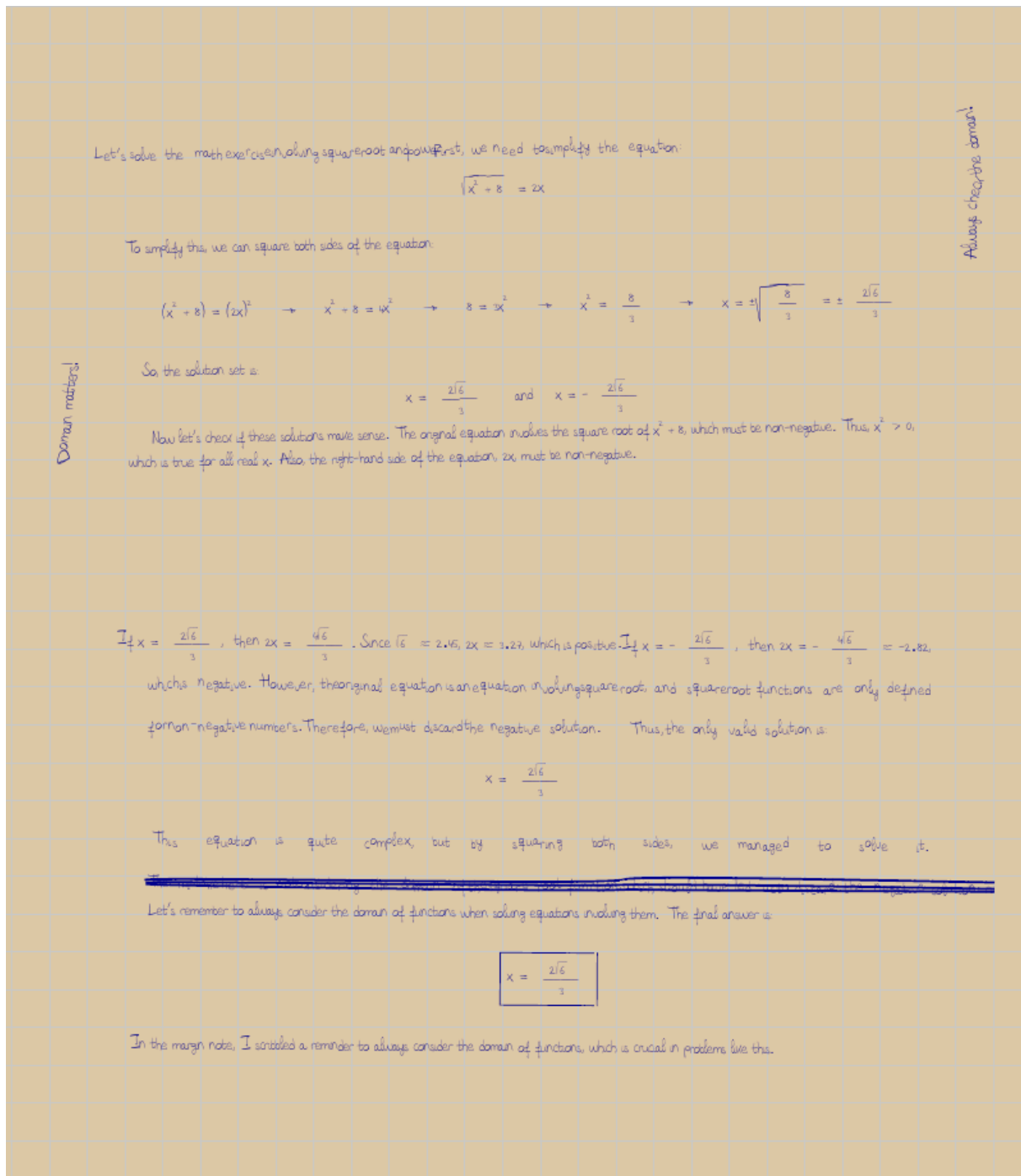


Figure 6: Pipeline improvements - Sample 1 prior to increments

$$\begin{aligned}
 \sqrt{x^2 \cdot 16} &= \sqrt{x^2 \cdot 4} \\
 &= \sqrt{4x^2} \quad (\text{Wurzel aus 4 ist 2, also } \sqrt{4} = 2) \\
 &= 2 \cdot \sqrt{x^2} \\
 &= 2 \cdot \sqrt{x \cdot x} \\
 &= 2 \cdot \sqrt{x} \cdot \sqrt{x} \\
 &= 2 \cdot x
 \end{aligned}$$

Das Ergebnis der Gleichung ist $2 \cdot x$. Die Wurzel aus 4 ist 2, nicht 4. Eine Übung mit Wurzeln und Potenzen ist die Lösung $2 \cdot x$.

Figure 7: Pipeline improvements - Sample 2 prior to increments

Sample outputs of the pipeline after the increments may be found in Figure 8 and Figure 9.

(10 points)

Exercise 8

Let (X, Y) be a continuous random vector with joint probability density function

$$f_{X,Y}(x, y) = \begin{cases} c \cdot x \cdot y & \text{if } 0 < x < 2 \text{ and } 0 < y < x^2, \\ 0 & \text{otherwise,} \end{cases}$$

where $c > 0$ is a normalization constant.

(a) (3 points) Determine the value of the constant c .

(b) (4 points) Compute the marginal probability density function $f_Y(y)$ of Y .

(c) (3 points) Calculate the conditional probability $P(X < 1.5 \mid Y = 1)$.

(a) $\int_0^2 \int_0^{x^2} cxy \, dy \, dx = 1$. The ~~same~~ integral equals $\int_0^2 \frac{x}{2} \, dx = \frac{16c}{3} \cdot c = \frac{3}{16}$.

(b) For $y \in (0, 4)$, $f_Y(y) = \int_{\sqrt{y}}^2 \frac{3}{16} xy \, dx$. $f_Y(y) = \frac{3y}{32}(4 - y)$. (c) $f_X(x|y=1) = \frac{2x}{3}$.

for $1 < x < 2$, $P(X < 1.5 \mid Y = 1) = \int_1^{1.5} \frac{2x}{3} \, dx = \frac{5}{12}$.

Figure 8: Pipeline improvements - Sample 1 after increments

Exercise 6

[10 points]

Polynomial Factorization and Partial Fractions

(a) Factor the polynomial $P(x) = x^4 - 5x^2 + 4$ completely into linear factors over $\mathbb{R}$.

[4 points]

(b) Determine the partial fraction decomposition of $\frac{3x^3 - 2x^2 + 5x - 1}{P(x)}$.

[6 points]

(a) $P(x) = x^4 - 5x^2 + 4$. Substitution $u = x^2$ yields $u^2 - 5u + 4 = 0$. Factoring gives $(u - 1)(u - 4) = 0$.

Therefore $P(x) = (x^2 - 1)(x^2 - 4) = (x - 1)(x + 1)(x - 2)(x + 2)$. (b) The decomposition takes the form

$\frac{A}{x-2} + \frac{B}{x+2} + \frac{C}{x-1} + \frac{D}{x+1}$. Clearing denominators produces

$3x^3 - 2x^2 + 5x - 1 = A(x+2)(x-1)(x+1) + B(x-2)(x-1)(x+1) + C(x-2)(x+2)(x+1) + D(x-2)(x+2)(x-1)$. For $x = 2$, $25 = 12A$, so $A = \frac{25}{12}$. For $x = -2$, $-93 = -12B$, so $B = \frac{93}{12}$. For $x = 1$, $5 = -6C$, so

$C = -\frac{5}{6}$. For $x = -1$, $-11 = 6D$, so $D = -\frac{11}{6}$. ~~Therefore~~ the final expression

$$\frac{25}{12(x-2)} + \frac{93}{12(x+2)} - \frac{5}{6(x-1)} - \frac{11}{6(x+1)}$$

Figure 9: Pipeline improvements - Sample 2 after increments

Besides rigid line formatting, these increments cannot address other limitations of the pipeline, discussed in Section 1.1. Nevertheless, some purely deterministic improvements and the typed problem statements, suggested by the Apertus-team, were realised.

2.2 Evaluation of Generative AI

Considering the existing data generation pipeline described in Section 1.1 and Appendix C, there are three main entry points for diffusion-style models: Document Augmentation, Document Subsection Augmentation and Font Generation.

Throughout this section, the license constraints of the models used must be considered (detailed in Section 1.5). Models that are not permitted under their licenses may still be used for feasibility checks, but not for any production systems that interact with Apertus.

For replication of these systematic experiments, refer to Appendix G.3. Quick summaries of each model used throughout this section, may be found in the Glossary, for a more detailed overview of the technical background of these models, we refer to Appendix B.

2.2.1 Document Augmentation

First, it is possible to augment finished outputs of the pipeline using a diffusion-style model. Through this approach, it is theoretically possible to address the previously described limitations related to rigidity. Line formatting can be broken up, formula placement made less predictable, and character placement made more irregular.

The intended improvement to the pipeline is shown in Figure 10.

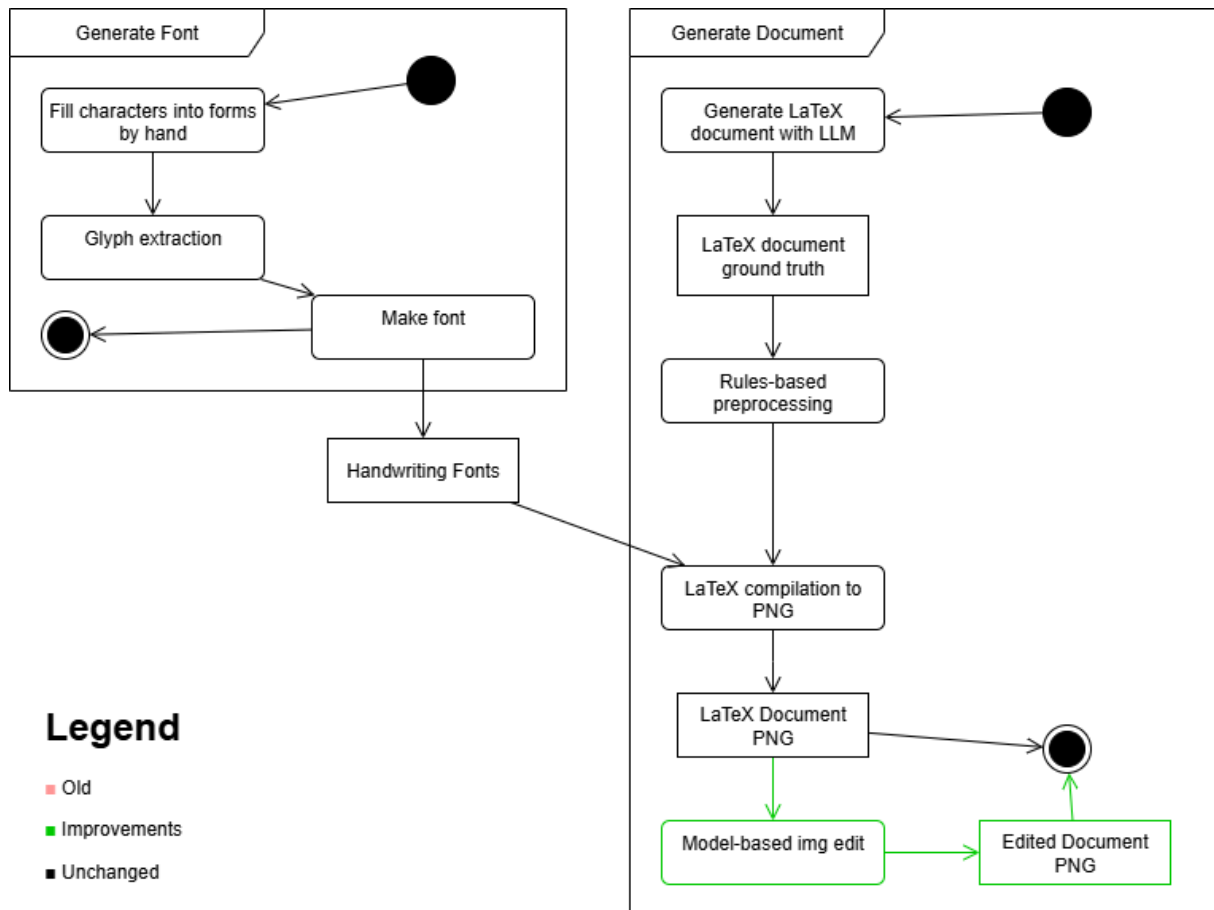


Figure 10: Document augmentation - Pipeline improvement UML

The existing application is described in Appendix C. Here, the model receives a document PNG compiled by the *LaTeX*-Compiler, together with a prompt to make it look more handwritten. It outputs a new PNG that should, ideally, look more handwritten.

The feasibility of this approach was primarily evaluated using *FLUX.2-dev* by Black Forest Labs and *Qwen-Image-Edit* by Alibaba. These models are permitted under the license constraints in Section 1.5 and are, to the best of our knowledge, the strongest open-weight diffusion models at the time being. Additionally, samples using the *Nano Banana Pro* model by Google and *gpt-image-1.5* by OpenAI were generated for comparison.

The prompt used for editing the input can be found in Appendix D.2. Figure 11 shows the base image provided to the models. One can observe several limitations of the existing *data generation pipeline* within this base image. Text lines are perfectly straight, formula formatting is rigid and some characters appear unrealistic (for example the fraction lines are dashed). Additionally, there are no character variations, an “a” looks exactly the same throughout the entire image.

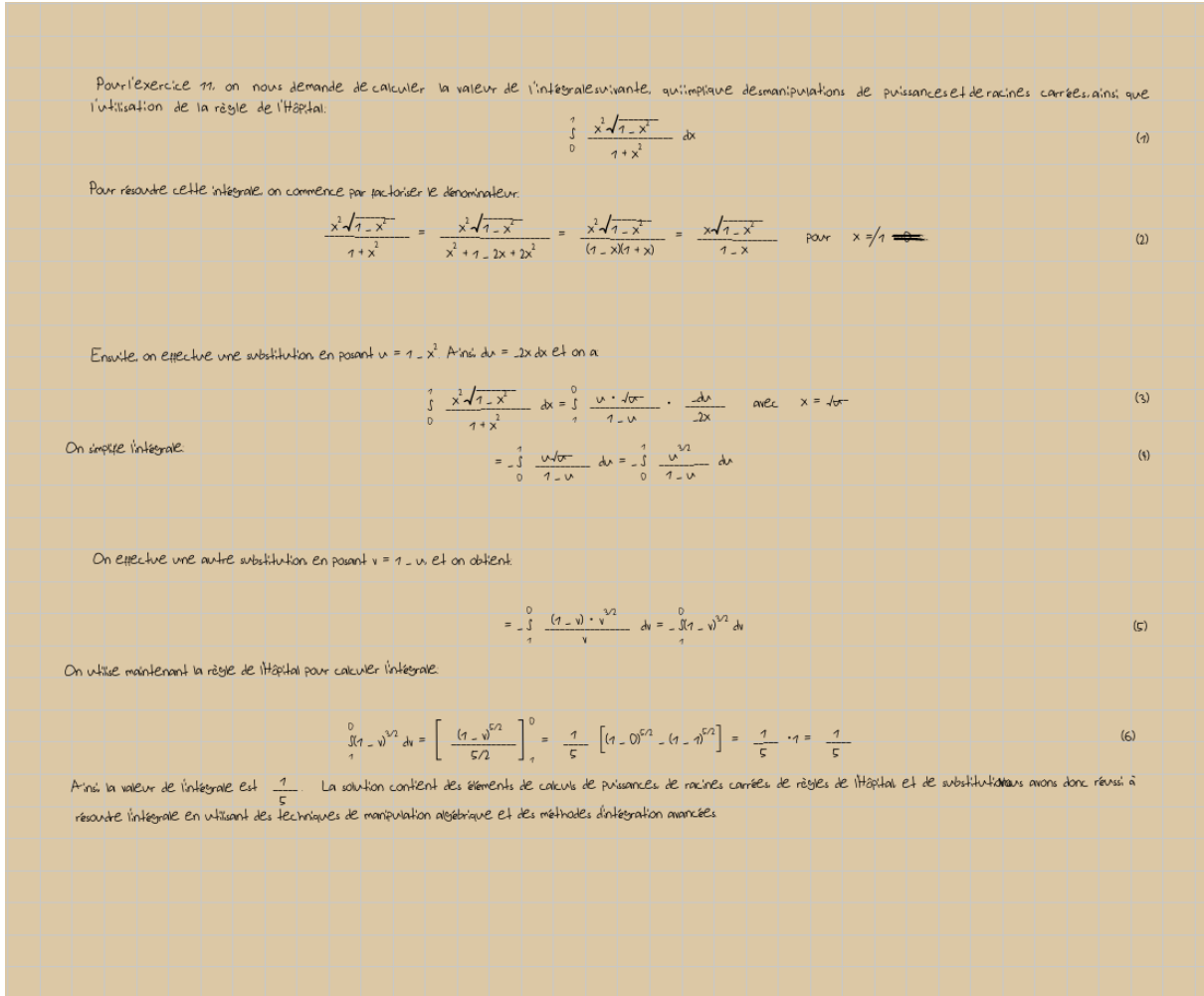


Figure 11: Document augmentation - Input image

Outputs of different models are given and described below.

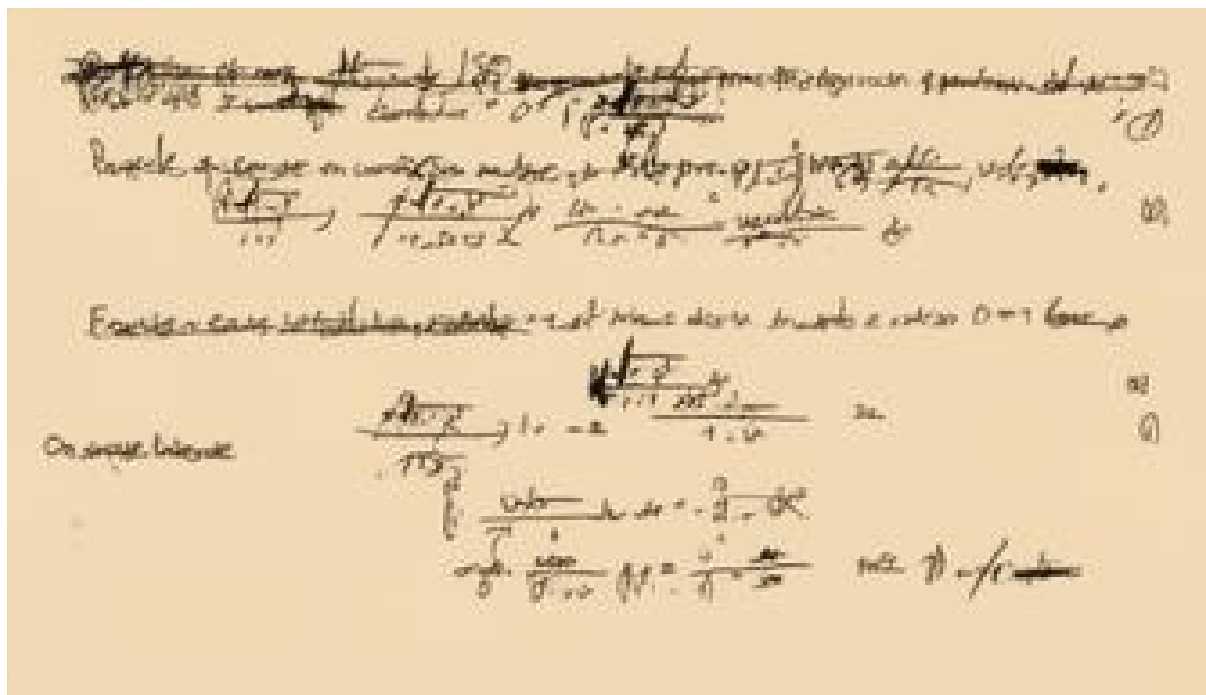


Figure 12: Document augmentation - Output of FLUX.2-dev

In Figure 12, heavy hallucinations and low resolution can be observed, to which *FLUX.2-dev* appears more prone compared to other models.

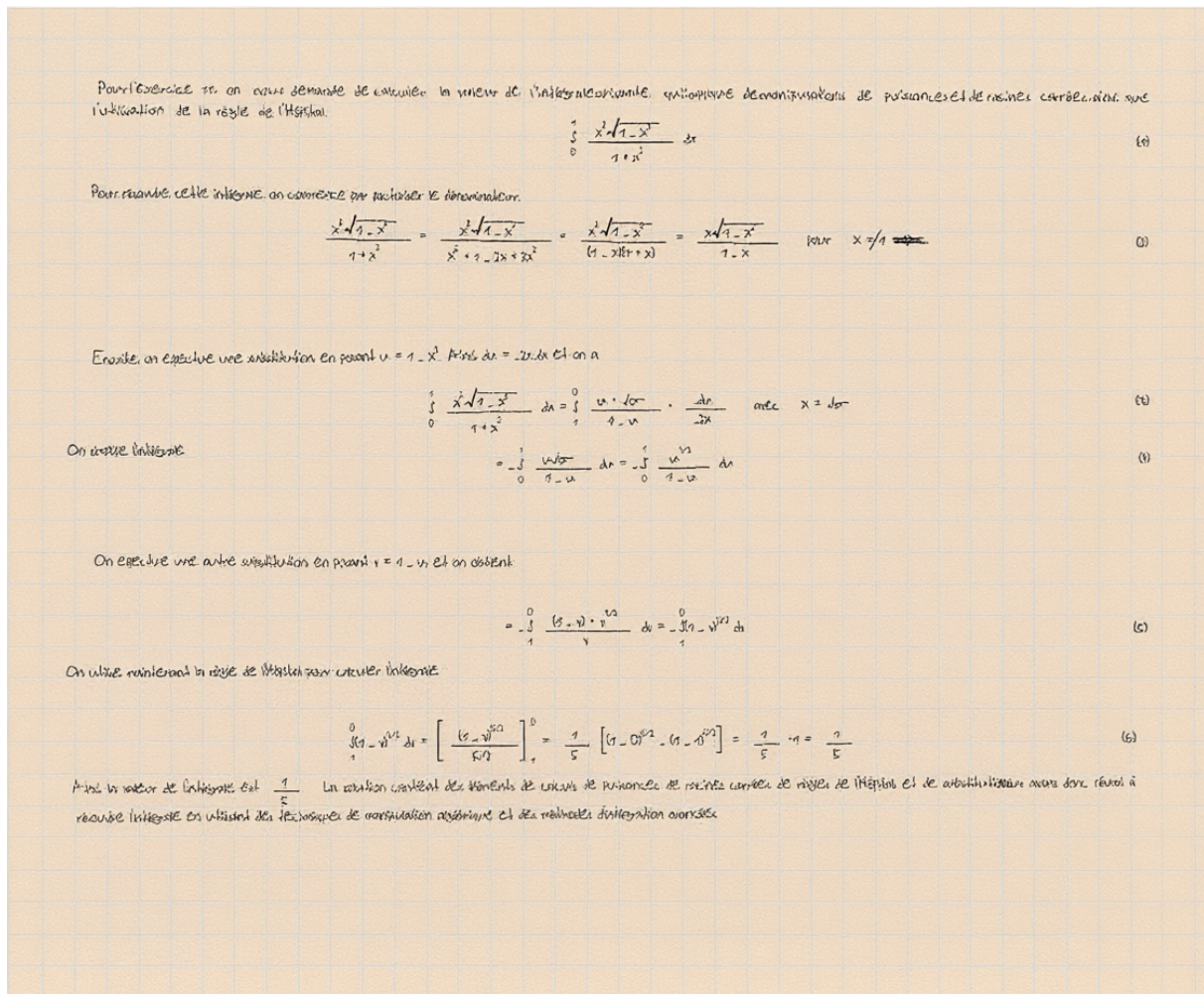


Figure 13: Document augmentation - Output of Qwen-Image-Edit

In Figure 13, the dashed fraction lines were partially resolved. The text also appears superficially similar to French. On closer inspection, however, the words are often not actually French. The formulas contain altered numbers, variables, and operators in several places. Often, closer inspection reveals artifacts that are not intelligible without context (e.g. “letters” forming a seemingly recognizable French word). However, unlike *FLUX.2-dev*, the content remains roughly similar overall.

Pour l'exercice 4, on nous demande de calculer la valeur de l'intégrale suivante, utilisant une démonstration de puissances et le recours certain avec une utilisation de la règle de l'Hôpital :

$$\int_0^1 \frac{\sqrt{x}}{1+x^2} dx \quad (1)$$

Pour résoudre cette intégrale on commence par isoler le dénominateur :

$$\frac{\sqrt{x}}{1+x^2} = \frac{\sqrt{x}}{x(+\frac{x^2}{x})} = \frac{\sqrt{x}}{x(\frac{x^2}{x}++)} = \frac{1}{x\sqrt{x}} = \frac{1}{x\sqrt{x}} \text{ pour } x > 1 \quad (2)$$

Ensuite on effectue une substitution en posant $u = \frac{1}{(\sqrt{x})}$. Ainsi $du = -\frac{1}{2\sqrt{x}} dx$

$$\int_0^1 \frac{1}{x\sqrt{x}} dx = \int_0^1 \frac{1}{3} du = \frac{1}{3} \quad (3)$$

On intègre l'intégrale :

$$\int_0^1 \frac{1}{\sqrt{x}} dx = \frac{2\sqrt{x}}{2} \Big|_0^1 = 2 \quad (4)$$

On effectue une autre substitution en posant $v = \frac{1}{x}$ et on obtient :

$$\int_0^1 \frac{1}{x^{\frac{3}{2}}} dx = \int_0^1 \frac{1}{1\sqrt{v}} dv = -\frac{1}{2} + 1 = \frac{1}{2} \quad (5)$$

On utilise maintenant la règle de l'Hôpital pour calculer l'intégrale :

$$\int_0^1 \frac{\sqrt{x}}{1+x^2} dx = \lim_{x \rightarrow x_0} \left[\frac{\sqrt{x}}{\frac{x}{x}} = \frac{1}{(1+x^2)\frac{x}{x}} \right] = \lim_{x \rightarrow x_0} \frac{1}{x} = \frac{1}{2} \quad (6)$$

Ainsi la valeur de l'intégrale est 1. Le calcul combiné de l'éternité de calculs de puissances, de la règle de l'Hôpital et des substitutions, nous donne bien à nouveau l'intégral en utilisant la technique de manipulation asymptotiques avancées.

Figure 14: Document augmentation - Output of gpt-image-1.5

In Figure 13, *gpt-image-1.5* nicely reformatted the entire document. The overall structure appears more plausible than the input, and the text is highly legible. This is close to the objective of document augmentation, were it not for the substantial amount of hallucination. Formulae seem most problematic, they were often altered almost completely content-wise (see red box nr. 1). The formulae also contain structures that are not valid mathematical expressions (see red box nr. 2) The text, on the other hand, is often copied correctly, although some words are replaced entirely (see red box nr. 3, “des éléments” becomes “de l'éternité”, which is not a real French word). Notably, the letters flow into words that form sentences convincingly, even if they do not always result in meaningful content.

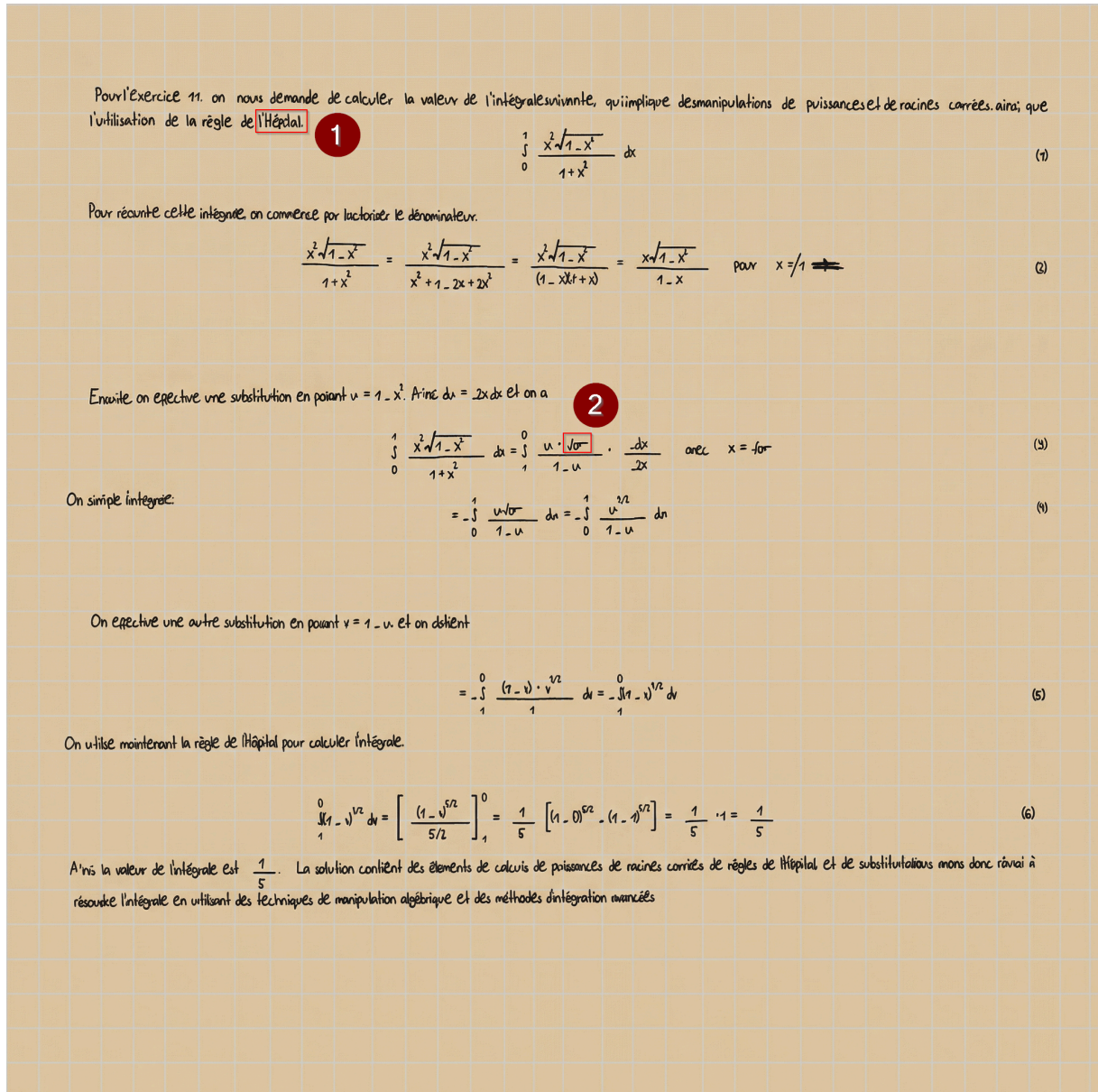


Figure 15: Document augmentation - Output of Nano Banana Pro

In Figure 13, *Nano Banana Pro* performed less aggressive reformatting of the document compared to OpenAI's model. The dashed fraction lines were resolved effectively and the characters exhibit slight variation. Nearly everything was copied faithfully, however, some elements were not processed correctly. For example, "l'Hôpital" in the first paragraph was changed to "l'Hépdal" (red box nr. 1), and the roots of "u" in the third formula were also mangled (red box nr. 2).

It seems even more capable, commercial models cannot handle the complexity of this approach, much less open-license models. Hallucinations during editing remain a major problem, detection thereof is addressed in Section 2.2.4.

2.2.2 Document Subsection Augmentation

In Section 2.2.1, hallucinations were a major issue. This might be due to the long-context nature of the task. To reduce context-size and hopefully hallucinations, within this Section model-based augmentation is applied only to the most problematic subsections of an input image, which often concerns formulae (see once more Figure 11). These subsections were identified based on visually salient issues

such as rigidity, irregular lines and spacing in the formulas, which are noticeable to the naked eye and for which the existing pipeline has no solution.

The intended improvement to the pipeline is shown in Figure 16.

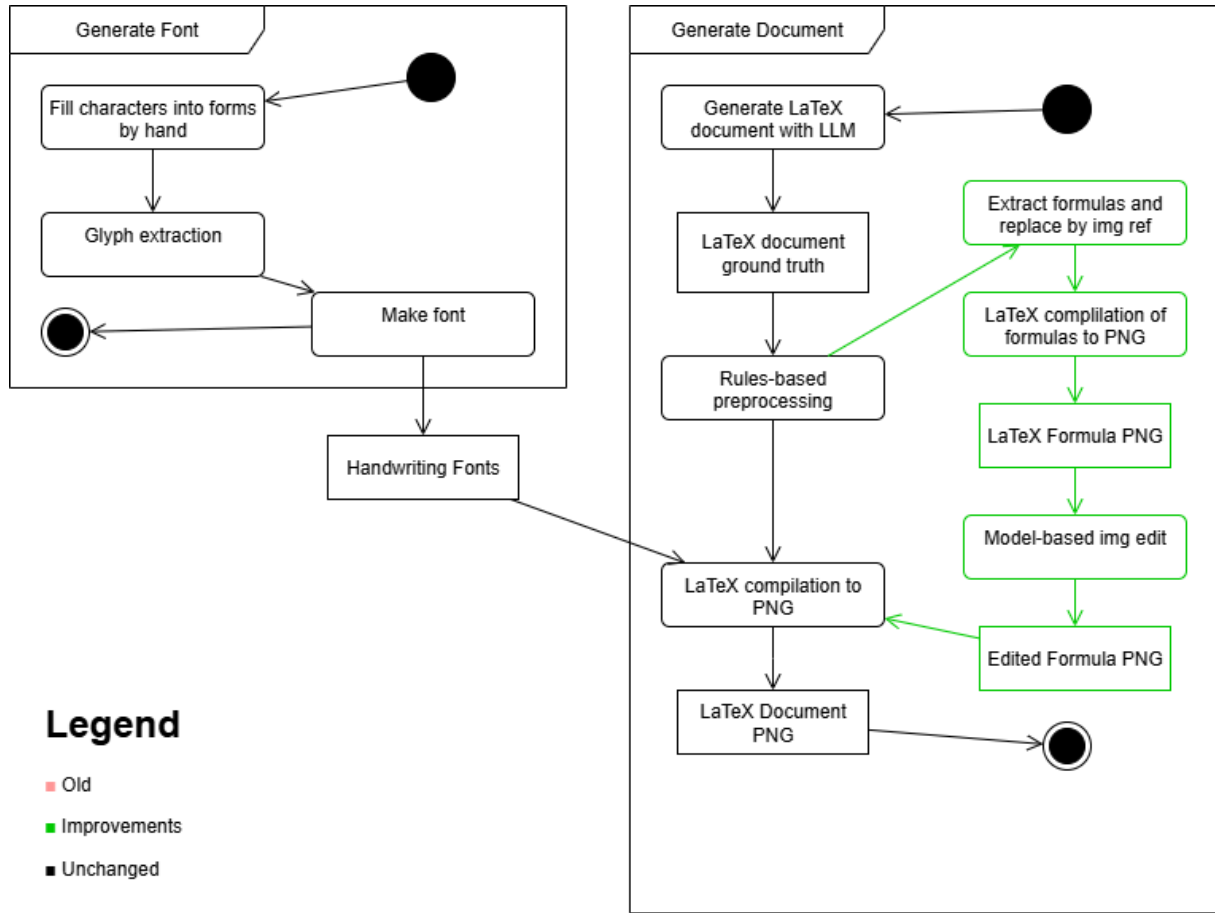


Figure 16: Subsection augmentation - Pipeline improvement UML

The existing application is described in Appendix C. In this setup, instead of directly compiling the LaTeX-document, problematic subsections (i.e. formulae) are extracted and individually compiled to PDF and then PNG. This PNG is then fed into the model together with a prompt to make it appear more handwritten (and optionally an additional image used to condition the desired style). The model outputs a new PNG that should look more handwritten. This new image is then placed as an image reference within the LaTeX-document, which is then compiled to produce the final output.

2.2.2.1 Setup

We conducted a systematic grid-search experiment on editing mathematical formulas to assess the viability of this approach.

The following six dimensions were explored.

Parameter	Values
Model	FLUX.2-dev , Qwen-Image-Edit
Name	3 differnt LaTeX formulas
Mode	handwritten , machine
Guidance	2, 5, 10, 20
Styleguide	Yes , No
Ground Truth	Yes , No
Seeds	3 random seeds each

Table 1: Document subsection augmentation - Grid search dimensions

Combining all elements across all dimensions yields 576 runs in total. Each run corresponds to one combination from the search space. The model was loaded and provided with an image to be edited. For runs including a Styleguide, an additional image was provided as input. Depending on the Styleguide and Ground Truth settings, different prompts were used.

Model

Either *FLUX.2-dev* or *Qwen-Image-Edit* was used (for details see Appendix B).

Name and Mode

In total, a pre-generated set of six images were used, consisting out of a total of 3 formulae (Name), each as *machine* and *handwritten* (Mode). This is illustrated in Table 2.

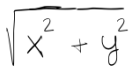
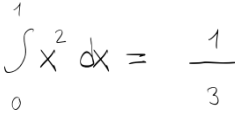
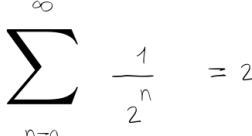
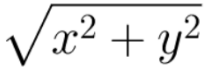
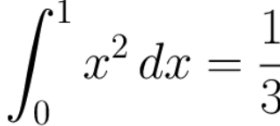
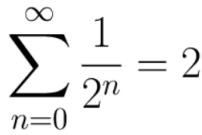
	FORMULA 1	FORMULA 2	FORMULA 3
Handwritten			
Machine			

Table 2: Document subsection augmentation - Input files in Name x Mode dimensions

The *handwritten* formulas were generated using the existing pipeline (see Appendix C). The *machine* formulas were generated using standard typesetting.

Guidance

Both models exposed a guidance variable, which strengthens prompt adherence the higher its value. Values were chosen on a range from high degrees of freedom to strict prompt adherence.

Styleguide

If a styleguide was to be used, Figure 17 was passed to the model in addition to the image to-be-edited.

The value $x=2$ lies in the domain since $g(2)=5>0$, so the derivative exists. Thus, the final answer is $\sqrt{5}$.
A common mistake is to forget the inner derivative—this was avoided here.

Figure 17: Styleguide for generative AI approaches

Passing two instead of one image required a slightly different prompt. The prompts can be found in Appendix D.3.

Ground Truth

If the ground truth was to be included in the prompt, the respective formula in LaTeX-format was added below the main prompt.

```
FORMULA1 = \sqrt{x^2 + y^2}
FORMULA2 = \int_0^1 x^2 \, dx = \frac{1}{3}
FORMULA3 = \sum_{n=0}^{\infty} \frac{1}{2^n} = 2
```

Seeds

The models take a seed number to initialize their randomness. Using a specific number results in a deterministic output for that number. The seeds are not supposed to influence anything specific beyond fixing a random initialization state of the model.

2.2.2.2 Rating

The resulting images were rated on a integer scale from -2 to 1:

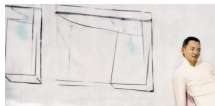
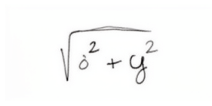
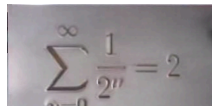
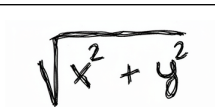
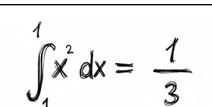
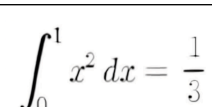
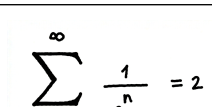
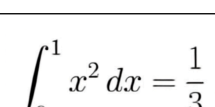
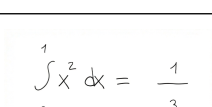
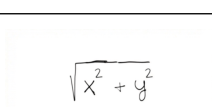
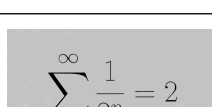
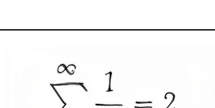
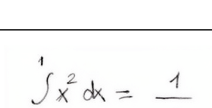
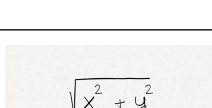
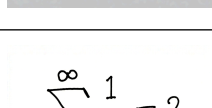
Rating -2: Unrelated, Empty, Hallucination, Colored				
Rating: -1: Bold, Multiline, Blurry				
Rating 0: Unchanged, Retained Machine Style				
Rating 1: Improved				

Table 3: Document subsection augmentation - Result ratings

2.2.2.3 Results

The mean over each grid search dimensions, when keeping all other dimensions fixed, can be seen in Figure 18.

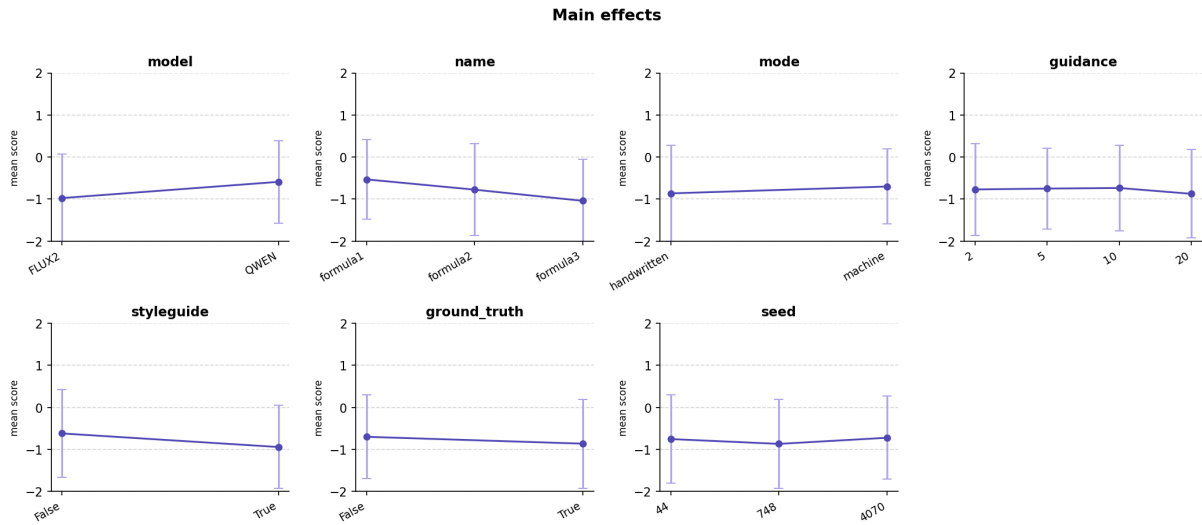


Figure 18: Document subsection augmentation - Impact of grid search dimensions

Of note are the large variations across different values for almost all dimensions. They represent the strong variability of quality in the outcomes with even minimal differences in input. Furthermore, it is apparent that *Qwen-Image-Edit* outperformed *FLUX.2-dev*. Importantly, the rising complexity from formula1 to formula3 had a negative impact, supporting our earlier hypothesis that hallucinations are a result of large context sizes. Also worth mentioning is the fact that models are able to process inputs of the `machine`-mode better than the `handwritten`-mode. Guidance did not change much on average, only minor quality fluctuations were observed. Adding the style guide and the ground truth results in worse output quality, once more suggesting that small context sizes are critical for good outputs. Lastly, some random variability within the seeds can be observed.

Even with the most optimal parameter combinations, the mean output quality did not rise above 0, which, per definition from Section 2.2.2.2, stands for “Unchanged, Retained Machine Style”. Nevertheless, some samples did achieve the rating 1, i.e. achieved an improvement due to this approach. The smaller context-size did, seemingly, help models to achieve better performances.

To evaluate current-day technical capabilities, an explorative experiment over the same dimensions (minus Guidance and Seed) was conducted, using *Nano Banana Pro*.

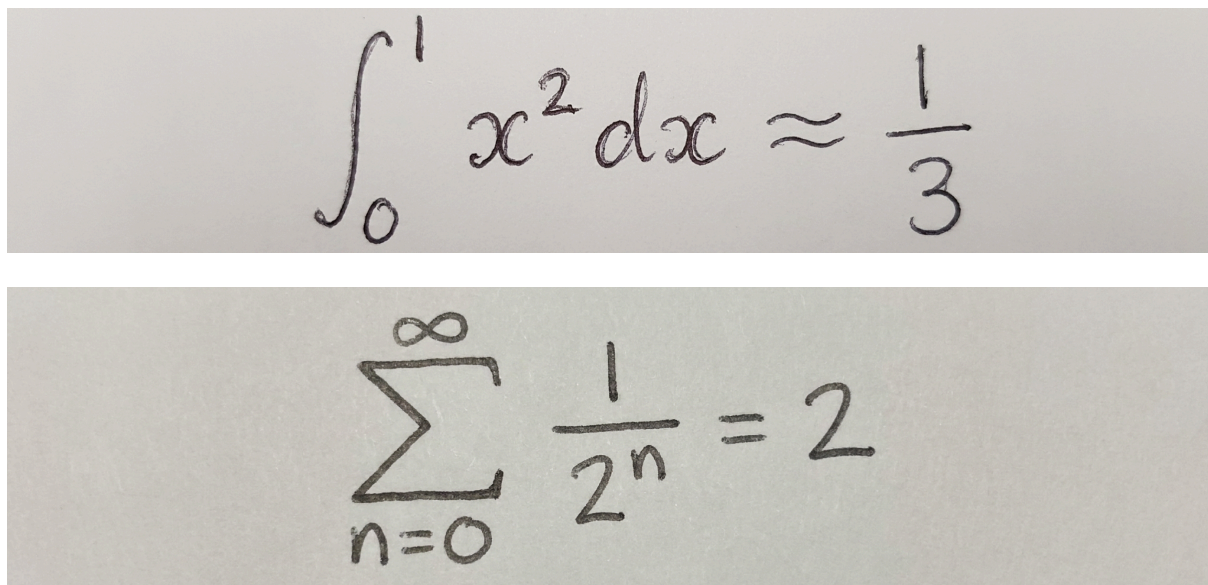

$$\int_0^1 x^2 dx \approx \frac{1}{3}$$
$$\sum_{n=0}^{\infty} \frac{1}{2^n} = 2$$

Figure 19: Document subsection augmentation - Output of Nano Banana Pro

As seen in Figure 19, the commercial model could output faithful style-copies of the style-guide (second image), but still produced hallucinations (“=” to “ $\approx$ ” in the first image). Once more, we refer to Section 2.2.4 for detection and filtering of such hallucinations.

2.2.3 Font Generation

Lastly is the creation of new handwritten fonts. Within the EPFL student project [1], an application to create fonts from real, human handwriting was created.

The full *data generation pipeline* is shown in Figure 20, the font creation application is on the left hand side. The intended improvement to that application is also illustrated.

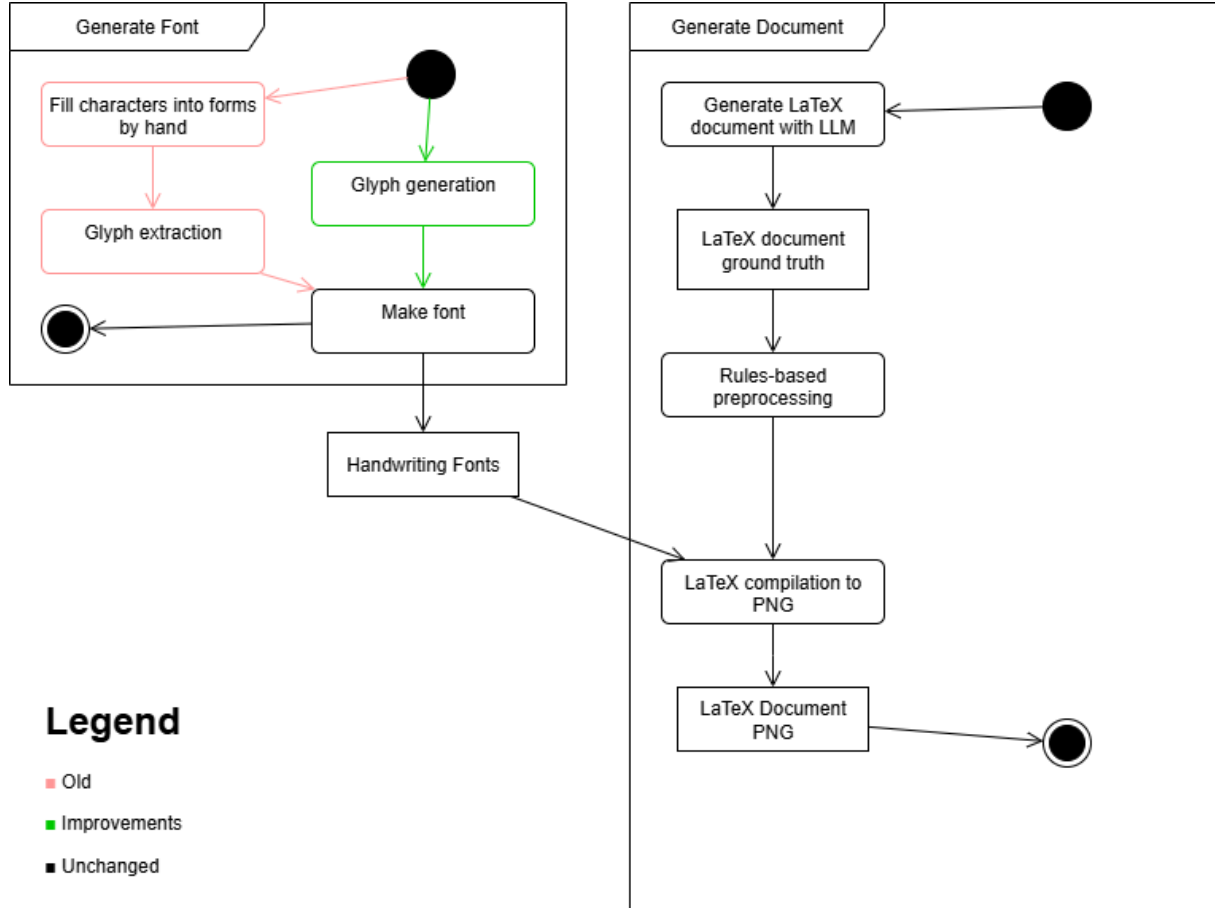


Figure 20: Font Generation - Pipeline improvement UML

The existing application is described in Appendix C. This application produces templates where users handwrite individual characters. The filled templates are then converted into an Open-Type-Font (*OTF*). As opposed to humans producing the characters, diffusion-style models may be used to generate handwritten versions of characters. Though this approach can't resolve the limitations of the current pipeline, it would allow for greater data variety.

Evaluation of this approach was performed using a grid-search style experiment using the parameters in Table 4, resulting in a total 36 fonts.

Parameter	Values
Model	FLUX.2-dev , Qwen-Image-Edit , Nano Banana Pro
Style Guide Image	Yes , No
Character Image	Yes , No
Seeds	3 random seeds each

Table 4: Font Generation - Grid search dimensions

Style Guide Image refers to a snippet of real, human handwriting (like in Section 2.2.2), whereas the Character Image is a picture of the character-to-be-produced in a typed computer font. Additionally, the prompt in Appendix D.4 was used. Note: For *Qwen*, in case no input images were used, the *Qwen-Image* model was used as opposed to the *Qwen-Image-Edit* version.

All produced fonts may be found in the GitLab Repository [9] in `generate_font/fonts`. The results by model are described in the below paragraphs. For each model examples of the sentence: `Franz jagt im komplett verwahrlosten Taxi quer durch Bayern. 1234567890`, rendering in some of the generated fonts are displayed.

2.2.3.1 FLUX.2-dev

FLUX.2-dev faired consistently across all parameter combinations of the grid-search. Most characters are legible and are clearly not in computer font, yet do not look handwritten either. However, quite some characters are illegible. Examples may be found in Figure 21, these are in order: neither style guide nor character image, no style guide and a character image, style guide and no character image, both style guide and character image.

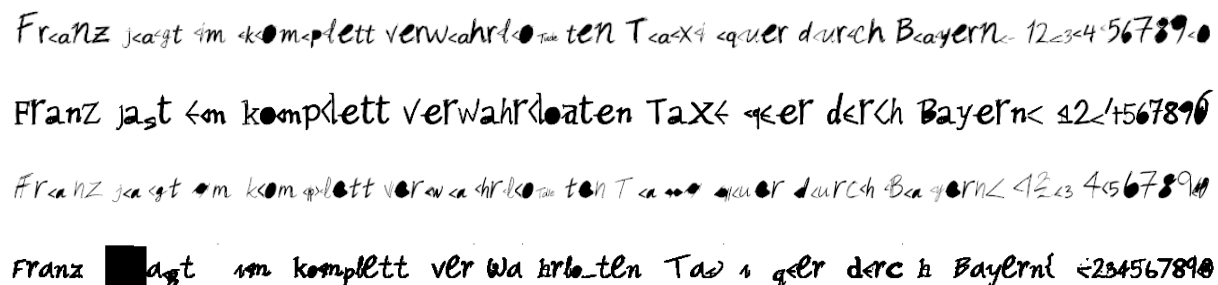


Figure 21: Font generation - FLUX.2-dev examples

2.2.3.2 Qwen-Image-Edit

Qwen-Image-Edit (or *Qwen-Image* for parameter combinations without input images), produced some legible characters, however these are very similar to characters typed in a computer font. Many other characters were illegible. Examples may be found in Figure 22, these are in order: neither style guide nor character image, no style guide and a character image, style guide and no character image, both style guide and character image.

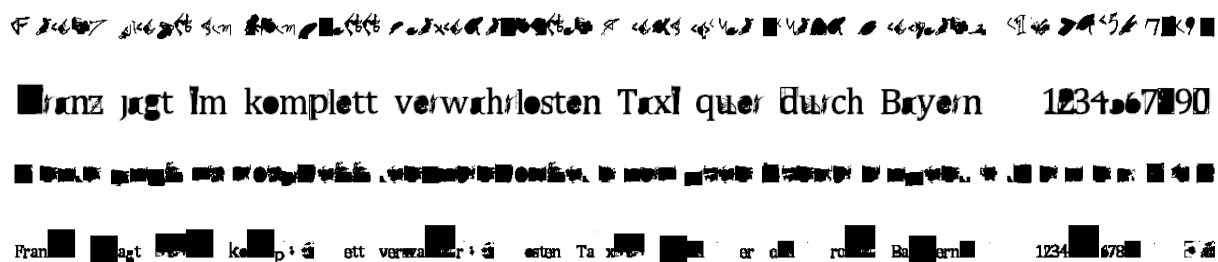


Figure 22: Font generation - Qwen-Image-Edit examples

2.2.3.3 Nano Banana Pro

Nano Banana Pro is not permitted license-wise and is therefore used, once more, purely for evaluation purposes. The handwriting style looks quite realistic and there aren't many illegible characters either. Nonetheless, *Nano Banana Pro* sometimes creates different characters than what was asked for and also seems to struggle with consistent character sizes. Examples may be found in Figure 23, these are in order: no style guide and a character image, style guide and no character image, both style guide and character image.

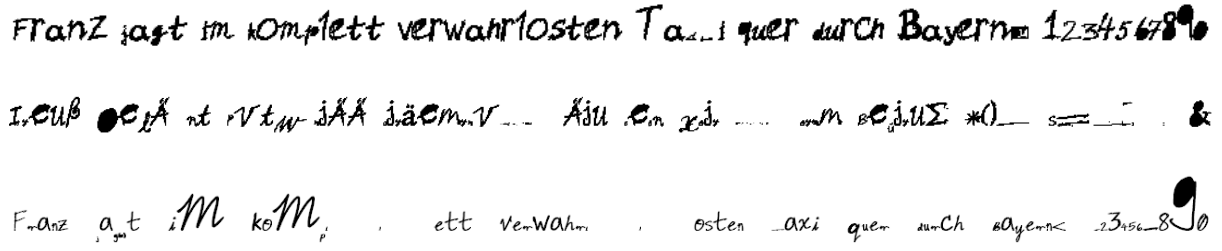


Figure 23: Font generation - Nano Banana Pro examples

2.2.4 Model Judge

The aforementioned approaches introduce a new potential problem: content loss and hallucinations. Documents that are augmented using generative AI may have additional content or lose content compared to the ground truth. If ground truths and handwritten artifacts no longer match content-wise, generated data is unusable for model training. To address this, we propose the Model-Judge Framework, more specifically we propose two setups: Single-Model Judges and Multi-Model Judges. *Vision-language-models* receive both the ground truth and the final artifact and are asked to judge whether the artifact is still legible, complete in terms of content and hallucination-free. Through this, outputs produced with generative AI can be filtered, prior to being added into a model-training dataset.

The basic idea of a Model-Judge is adapted from Qian et. al [10], who have constructed a large-scale image dataset by augmenting existing images, using *Nano Banana Pro*. Qian et. al used Nano Banana Pro to add a variety of effects, such as changing the weather, repositioning objects, changing colors, etc. In their work, they have used Judges as quality assurance, a sample is regenerated if Nano Banana Pro does not produce outputs sufficient in quality, i.e. the desired effect is not achieved. We have adapted this idea to make Judges compare a sample with its ground truth content-wise, rather than assessing the sample quality-wise.

To demonstrate the capabilities of our Framework, we conducted a systematic evaluation. To control exactly the amount of samples with and without hallucinations, as well as the kind of hallucination, samples were produced by humans as opposed to being generated. Several people received machine written formulae and were asked to handwrite them, once correctly and once incorrectly with a “hallucination”.

Typical “hallucinations” that people were asked to introduce:

- Changed/additional/removed number (e.g. the number 2 to 3)
- Changed/additional/removed letter/symbol (e.g. Capital sigma to capital pi)
- Changed/additional/removed operator (e.g. plus to minus)

These are consistent with the hallucinations observed in Section 2.2.1 and Section 2.2.2.

For each sample, the evaluated judges received: The machine written formula as ground truth, the formula in LaTeX-format as additional ground truth, the correct or incorrect handwritten version of the formula and the full judge-prompt, found in Appendix D.5. The judges were then evaluated using confusion matrices, the entries are defined in Table 5.

<i>True positive (TP)</i>	The handwritten version is ‘Correct’ and was judged as ‘Correct’
<i>False positive (FP)</i>	The handwritten version is ‘Incorrect’ and was judged as ‘Correct’
<i>True negative (TN)</i>	The handwritten version is ‘Incorrect’ and was judged as ‘Incorrect’
<i>False negative (FN)</i>	The handwritten version is ‘Correct’ and was judged as ‘Incorrect’

Table 5: Model Judge - Confusion matrix term definitions

Given these definitions and the context of this work, the *False Positive Rate* defined as

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

is most important. Since False Positives represent samples with hallucinations that weren't discarded and are now present in a data set, they reduce the usability of said dataset in AI training, therefore the FPR must be as low as possible. Additionally, the *Precision* defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

is of similar importance and must be as close to 1 as possible. To get a full evaluation of each judge, the *Accuracy* defined as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

and the *True Positive Rate* defined as

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

are computed, both of these must also lie as close to 1 as possible.

2.2.4.1 Single-Model Judge

The Single-Model Judge (SMJ) consists out of a single model. Its viability was evaluated with three different models: *Kimi K2.5*, *Qwen3.5-27B* and *Qwen3.5-397B-A17B*. The results may be found in Table 6, Table 7 and Table 8.

	Judged 'Correct'	Judged 'Incorrect'
Is 'Correct'	32	41
Is 'Incorrect'	0	74

Metric	Computed Value
FPR	0.0000
Precision	1.0000
Accuracy	0.7211
TPR	0.4384

Table 6: Model Judge - SMJ1: Qwen3.5-27B

	Judged 'Correct'	Judged 'Incorrect'
Is 'Correct'	68	1
Is 'Incorrect'	16	53

Metric	Computed Value
FPR	0.2319
Precision	0.8095
Accuracy	0.8768
TPR	0.9855

Table 7: Model Judge - SMJ2: Qwen3.5-397B-A17B

	Judged 'Correct'	Judged 'Incorrect'
Is 'Correct'	69	6
Is 'Incorrect'	4	49

Metric	Computed Value
FPR	0.0755
Precision	0.9452
Accuracy	0.9219
TPR	0.9200

Table 8: Model Judge - SMJ3: Kimi K2.5

2.2.4.2 Multi-Model Judge

In the Multi-Model Judge (MMJ), each sample is independently judged by three models. Afterwards, the judge verdict is decided by majority vote. Under the assumption that the three models make independent errors, one can make a claim about how the total FPR and TPR of a Multi-Model Judge behave. The FPR is the probability of a model judging a sample as 'Correct'/hallucination-free when it was actually 'Incorrect'/had a hallucination

$$P(\text{Model judges 'Correct'} \mid \text{Sample is 'Incorrect'}).$$

A Multi-Model Judge produces a False Positive, if two or more models falsely vote 'Correct'. There are four ways in which this can happen (Table 9, note that '1' corresponds to 'Correct').

Model A	Model B	Model C	Judge Verdict
1	1	1	'Correct'
1	1	0	'Correct'
1	0	1	'Correct'
0	1	1	'Correct'

Table 9: Model Judge - total False Positive Rate derivation

The probability that all three models falsely vote 'Correct' is equal to the product of the individual models False Positive Rates: $FPR_A \cdot FPR_B \cdot FPR_C$. The probability that only A and B falsely vote 'Correct', is $FPR_A \cdot FPR_B \cdot (1 - FPR_C)$. Similarly, the probabilities that only A and C or only B and C falsely vote 'Correct', are $FPR_A \cdot FPR_C \cdot (1 - FPR_B)$ and $FPR_B \cdot FPR_C \cdot (1 - FPR_A)$. Since all four events are mutually exclusive, their probabilities may be added to obtain the total, Multi-Model Judge False Positive Rate. After simplifying, this results in

$$FPR_{\text{tot}} = FPR_A \cdot FPR_B + FPR_A \cdot FPR_C + FPR_B \cdot FPR_C - 2 \cdot FPR_A \cdot FPR_B \cdot FPR_C.$$

Likewise, the total, Multi-Model Judge True Positive Rate is

$$TPR_{\text{tot}} = TPR_A \cdot TPR_B + TPR_A \cdot TPR_C + TPR_B \cdot TPR_C - 2 \cdot TPR_A \cdot TPR_B \cdot TPR_C,$$

which is obtained through the same derivation.

To demonstrate the effect on total FPR and TPR, suppose three hypothetical models with the rates seen in Table 10 are used.

Model	FPR	TPR
A	0.01	0.7
B	0.3	0.99
C	0.1	0.9

Table 10: Model Judge - Example improvement of Multi-Model setup

Using the above formulae, one obtains

$$\text{FPR}_{\text{tot}} = 0.0334 \quad \text{and} \quad \text{TPR}_{\text{tot}} = 0.9666.$$

Both rates improve substantially when compared to the individual model rates. However it must be mentioned again that these formulae only hold under the assumption of independent errors.

In practice, the Multi-Model Judge setup was executed thrice:

- **MMJ1:** Gemma4-31B-it, Qwen3.5-27B and Qwen3.5-397B-A17B
- **MMJ2:** Kimi K2.5, Qwen3.5-27B and Qwen3.5-397B-A17B
- **MMJ3:** Gemma4-31B-it, Qwen3.5-27B and Kimi K2.5

The results are displayed in Table 11, Table 12 and Table 13.

	Judged 'Correct'	Judged 'Incorrect'
Is 'Correct'	33	42
Is 'Incorrect'	3	72

Metric	Computed Value
FPR	0.0400
Precision	0.9167
Accuracy	0.7000
TPR	0.4400

Table 11: Model Judge - MMJ1: Gemma4-31B-it, Qwen3.5-27B and Qwen3.5-397B-A17B

	Judged 'Correct'	Judged 'Incorrect'
Is 'Correct'	66	9
Is 'Incorrect'	1	74

Metric	Computed Value
FPR	0.0134
Precision	0.9851
Accuracy	0.9334
TPR	0.8800

Table 12: Model Judge - MMJ2: Kimi K2.5, Qwen3.5-27B and Qwen3.5-397B-A17B

	Judged 'Correct'	Judged 'Incorrect'
Is 'Correct'	35	40
Is 'Incorrect'	0	75

Metric	Computed Value
FPR	0.0000
Precision	1.0000
Accuracy	0.7333
TPR	0.4666

Table 13: Model Judge - MMJ3: Gemma4-31B-it, Qwen3.5-27B and Kimi K2.5

2.3 Other Handwriting Generation Approaches

Although promising, the results from the generative experiments in Section 2.2 were mostly negative for the time being (see also Section 4). Considering the field of handwriting generation from a broader perspective, it is also possible to generate handwriting in entirely different manners, as opposed to alleviating the limitations of the data generation pipeline. In this section, we give an overview of other existing generation paradigms. The performance of the existing pipeline compared to these paradigms is an open question (see Section 5).

2.3.1 AI-Based Methods

There are several AI-based methods to generate handwriting. Existing approaches are generally categorized into diffusion-style models, recurrent neural networks (*RNN*), generative adversarial networks (*GANs*), variational autoencoders (*VAEs*) and *transformers* [11], [12].

Recent experimental research suggests that current state-of-the-art performance is achieved with transformer- and diffusion-based approaches [8]. These capture long-range dependencies, writer-specific styles and fine-grained stroke details more effectively than other models.

Besides diffusion-based methods, these approaches are not integrable into the existing pipeline, as they generate handwriting from scratch. Diffusion-based methods, on the other hand, can generate handwriting from scratch, but they can also be used to improve already generated handwriting, as discussed in Section 2.2.

As discussed in Section 1.3 content loss and hallucinations are a major issue when it comes to AI-based methods. These methods provide no guarantee, that generated handwriting matches a specified ground truth. The Model Judge proposed in Section 2.2.4 proves effective in limiting the extent of this problem.

2.3.2 Non-AI Based Methods

Aside from this, there exist several non-AI-based methods, Elarian et. al [2] have reviewed and summarized these. There exist two main categories of these methods: shape simulation and movement simulation. The idea of shape simulation is to directly simulate the shape of handwritten characters, whereas the idea of movement simulation is to simulate human movement that results in handwriting (i.e. hand movement, pressure, etc.).

In section 3 of their paper, they provide Figure 24 which presents an overview of shape simulation techniques. They further describe each technique in detail.

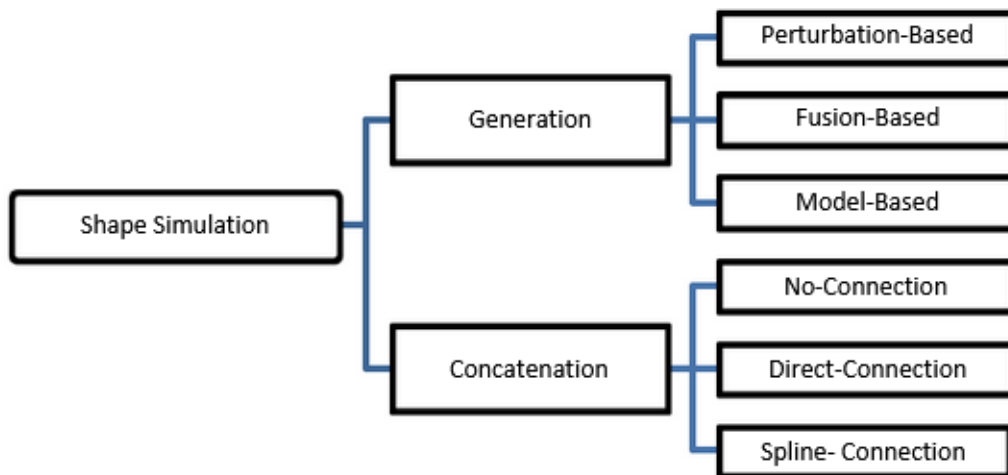


Figure 24: Overview of handwriting generation methods, adapted from Elarian et. al [2, Sec. 3]

For movement simulation techniques, they provide an extensive review of recent work. Some of these techniques do use machine learning, so these do not technically fit in the current section. They mention that one major issue of these techniques is parameter selection/tuning.

Overall, non-AI-based methods tend to struggle with:

- long-range consistency,
- writing rhythm,
- fatigue drift,
- subconscious variation,
- contextual letter adaptation.

A lack of subconscious variation and contextual letter adaptation can be observed in the existing data generation pipeline as well.

Empirically, AI-based methods seem to resolve these issues, however to the best of our knowledge, no large-scale comparison between the two paradigms exists.

2.4 Demo Client

As mentioned in Section 1.4, the demo-client is a tool to demonstrate pipeline functionality. It may be used to convert ones own LaTeX-files into handwritten documents. In context of the pipeline steps, described in Appendix C, the demo-client simply replaces the very first step. Instead of using an LLM to generate LaTeX, the input file is used and from thereon fed into the rest of the pipeline.

The entire existing pipeline, as well as the changes necessary for the demo-client are illustrated in Figure 25.

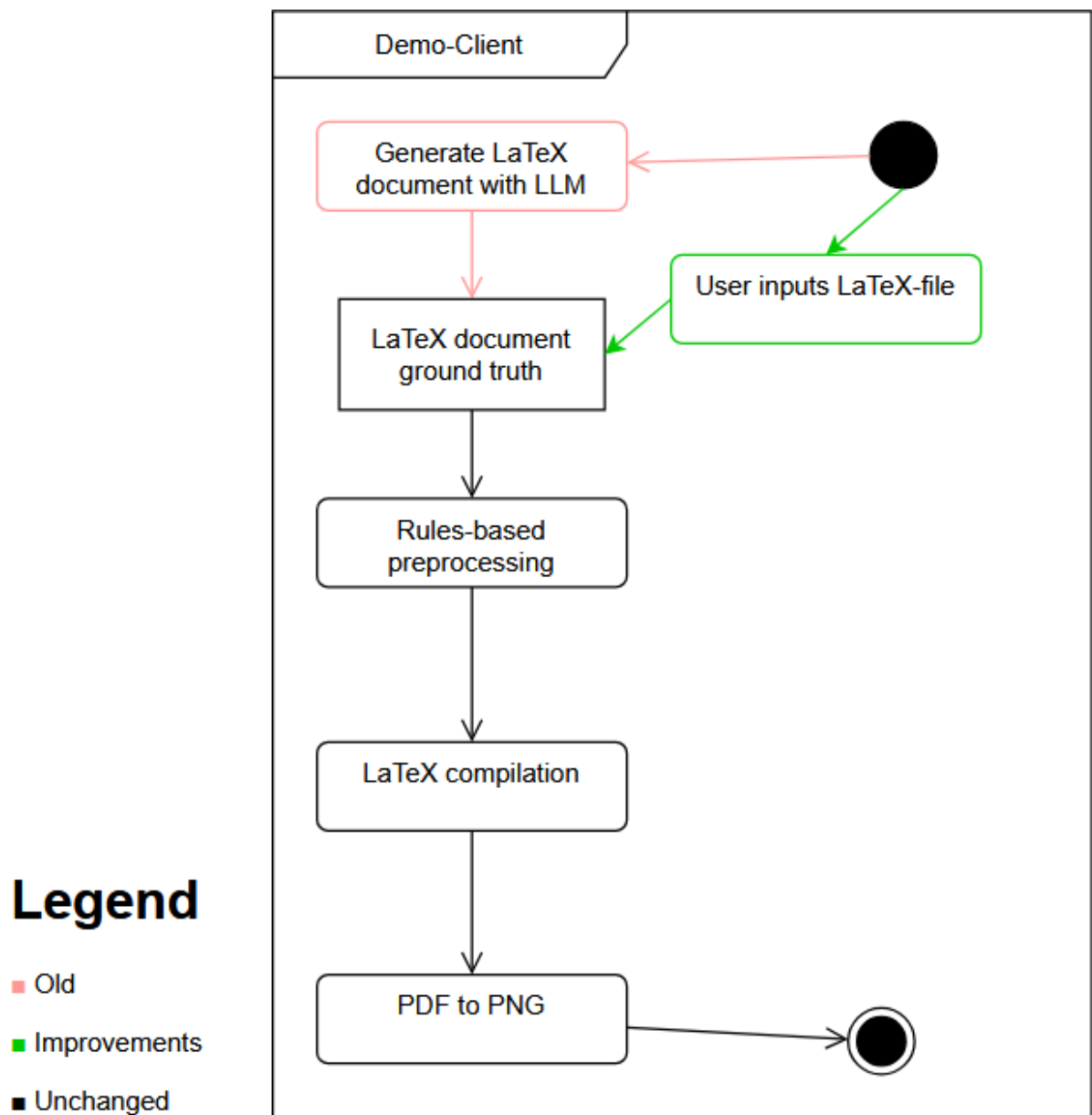


Figure 25: Demo-client overview

Furthermore, the demo-client is implemented as a *CLI-tool*. The client and usage instructions can be found in the GitLab Repository of this Thesis [9] in the directory *demo_client/*.

3 Realism Evaluation Procedure

Since the end goal is to train ETH’s *Apertus* model on handwriting recognition, synthetic data must be close enough to real handwriting, to allow *Apertus* to generalize from synthetic data to real data. Ultimately, only training *Apertus* on synthetic data and testing it on genuine human handwriting, can conclusively answer whether synthetic data is realistic enough, however successful transfer learning benefits from “closeness” to real data. However, since the thesis scope (see Appendix H) explicitly states a human experiment, our work focuses on that.

Existing methods to benchmark realism of generated handwriting [7] can be separated into quantitative and qualitative methods. Quantitative methods usually focus on one specific aspect of handwriting realism, for example the *Fréchet Inception Distance* computes the *feature vectors* of real and generated handwriting and then calculates their distance. Another example is the *Kullback-Leibler Divergence*, measuring the divergence (i.e. difference) between distributions of real and generated handwriting. These methods are therefore not suited for developing a general-purpose realism evaluation procedure. Qualitative methods, on the other hand, are based upon human inspection, participants are typically shown generated samples and asked “Is this real or not?”. Such Turing-style evaluations are more suited for a general realism assessment, since no aspect of handwriting is singled out and assessed independently. Though, once more, Diaz et. al state that no state-of-the-art for assessing realism exists: “In handwriting synthesis, the state of the art remains unclear. (...) The absence of common benchmarks and international competitions in handwriting synthesis sometimes makes it challenging to compare results fairly across different works.” [7, p. 9]

We therefore develop our procedure as a Turing-style evaluation. This chapter first outlines the theoretical background and methodology of the evaluation as a repeatable procedure. To demonstrate the viability of this evaluation, it is also carried out as a pilot study with the pipeline after the increments described in Section 2.1. Importantly, the generated data is not yet realistic enough to pass as “close enough” to real data, due to the limitations described in Section 1.1, hence the realism evaluation must demonstrate that. The pilot study is conducted to: a) gather further feedback on what makes generated data from the existing pipeline unrealistic and b) show the plausibility of the realism benchmark.

3.1 Methodology

The realism procedure is designed as a user test in a setup similar to a Turing Test. Participants are split into two groups to allow for independent testing of different factors. To ensure statistical significance, the procedure assumes at least 20 participants per group. The primary test result is the quantitative analysis of user ratings, which serves as a realism benchmark. As a secondary result, we propose a strategy to analyse qualitative user feedback. Furthermore, throughout this chapter, “document” refers to either a real handwritten document or a synthetically generated document, not exceeding one A4 page in length. All documents used in the two groups should be of comparable content and style. Lastly, participants must be informed to judge only the look of the handwriting. Consequentially, participants do not necessarily need to be able to speak the languages in the documents, so long as they are familiar with the alphabet used, which is Latin in this case.

3.1.1 Group 1

Participants receive 30 documents one by one and are asked to rate them on a *Likert scale*. They are informed that some are real, and some are generated. Ideally, roughly half the documents should be real while the other half should be generated. After rating all 30 documents, they are asked to provide reasoning for (some of) their ratings. The scale used is:

- 1 This image looks completely artificially generated
- 2 This image looks somewhat artificially generated
- 3 Uncertain
- 4 This image looks somewhat real
- 5 This image looks completely real

If participants can't distinguish real from generated documents, then the distribution of real documents and the distribution of generated documents are identical. Comparing distributions is done with a two-sample *Kolmogorov-Smirnov-Test*. Let X be the distribution of all real documents on the scale and let Y be the distribution of all generated documents on the scale. Then the *cumulative distribution functions* of X and Y are equal under the null hypothesis:

$$H_0 : F_X(x) = F_Y(x) \quad \forall x$$

$$H_1 : F_X(x) \neq F_Y(x) \quad \exists x$$

Additionally, if documents are indistinguishable, then generated documents aren't rated as "less natural" than real documents. Therefore, under the null hypothesis, the mean of generated documents on the *Likert scale* is not lower than the mean of real documents. Testing this, can be done with a one-sided *t-Test*:

$$H_0 : \mu_Y \not< \mu_X$$

$$H_1 : \mu_Y < \mu_X$$

These two tests establish evidence of whether, or whether not, real and generated documents are perceived as having the same realism in relative terms, rather than absolute ones. While the values of the means of real and generated documents do present an absolute marker for perceived realism, this data is not necessarily representative. Furthermore, since eventually "closeness" of generated data to real data is of importance, no such absolute claim is needed. Therefore, the tests are a relative comparison between real and generated data.

3.1.2 Group 2

Participants receive pairs of one real document and one generated document and are then asked to decide which is which. In total, each participant rates 15 pairs in this mode. After rating each document, participants are asked to provide reasoning for (some of) their ratings.

If real and generated documents are indistinguishable, one would expect 50% of all ratings to be correct, which can be tested with a *Binomial test*. Let p be the probability that a participant correctly identifies whether a sample is real or generated.

Now:

$$H_0 : p = 0.5$$

$$H_1 : p \neq 0.5$$

This test provides evidence about distinguishability in a direct comparison. It, once again, makes a claim about perceived realism of real and generated data, relative to each other. Unlike in group 1, the data basis for this claim stems from a direct comparison, rather than individual assessment.

3.1.3 Interpretation

The binomial test provides the strongest evidence regarding perceptual indistinguishability because it directly measures participants' ability to discriminate between real and generated handwriting in a side-by-side comparison. This is stronger than evidence provided by either test from Group 1, since these only consider documents in isolation with no reference material.

Within Group 1, the Kolmogorov-Smirnov-Test provides the stronger claim than the t-Test. This is because the Kolmogorov-Smirnov-Test considers the entire distributions, whereas the t-Test only examines their means. If the t-Test reports the means as equal, the distributions may still be wildly different.

Therefore, if we fail to reject only H_0 of the t-Test, one can claim that generated data is similar to real data. However, no statement regarding indistinguishability can be made. If, additionally, we fail to reject H_0 of the Kolmogorov-Smirnov-Test, this serves as evidence of indistinguishability when documents are considered in isolation. Lastly, if we also fail to reject H_0 of the Binomial test, one can claim that significant evidence of indistinguishability has been established, even if generated documents are considered in a side-by-side comparison with real data.

3.2 Experiment setup

To demonstrate the legitimacy of our realism assessment procedure, it is carried out once. This section describes the setup for that pilot run.

3.2.1 Test Samples

All samples that were used for both groups may be found in the GitLab Repository [9] in `experiment/experiment_data.zip`. An example of a generated test sample and a real test sample may be found in Figure 26 and Figure 27. Generated samples were produced after the pipeline increments described in Section 2.1. Real test samples were handwritten by humans and had the same format as generated ones, meaning a typed problem statement at the top, followed by a handwritten solution.

Aufgabe 7

Gegeben sei die lineare Rekursionsgleichung zweiter Ordnung $a_n = 5a_{n-1} - 6a_{n-2}$ für $n \geq 2$ mit den Anfangswerten $a_0 = 1$ und $a_1 = 3$.

a) Bestimmen Sie eine explizite Formel für a_n .

(5 Punkte)

b) Berechnen Sie den Wert von $a_{2024} \pmod{7}$.

(5 Punkte)

a) Die ~~charakteristische~~ charakteristische Gleichung lautet $r^2 - 5r + 6 = 0$. Diese faktorisiert zu $(r - 2)(r - 3) = 0$ mit den Nullstellen $r_1 = 2$ und $r_2 = 3$. Der Ansatz $a_n = A \cdot 2^n + B \cdot 3^n$ liefert durch Einsetzen der Anfangswerte das Gleichungssystem $A + B = 1$ und $2A + 3B = 3$. Subtraktion des Doppelten der ersten Gleichung von der zweiten ergibt $B = 1$ und somit $A = 0$. Also gilt $a_n = 3^n$. b) Es gilt $a_{2024} = 3^{2024}$. Modulo 7 berechnet sich dies mit dem Satz von Fermat wegen $3^6 \equiv 1 \pmod{7}$. Da $2024 = 337 \cdot 6 + 2$ folgt $3^{2024} = (3^6)^{337} \cdot 3^2 \equiv 1 \cdot 9 \equiv 2 \pmod{7}$.

Fermat'satz

Figure 26: Realism test - Generated test sample

Exercise 9

Let $A = \begin{pmatrix} 3 & -1 \\ 1 & 1 \end{pmatrix} \in \mathbb{R}^{2 \times 2}$.

- (a) (4 points) Determine the Jordan normal form J of A and a transformation matrix P such that $A = PJP^{-1}$.
 (b) (3 points) Compute the matrix exponential e^{At} for $t \in \mathbb{R}$.
 (c) (3 points) Solve the initial value problem $x'(t) = Ax(t)$ with $x(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

(a) The characteristic polynomial is

$\det(A - \lambda I) = (3 - \lambda)(1 - \lambda) + 1 = \lambda^2 - 4\lambda + 4 = (\lambda - 2)^2$ The eigenvalue $\lambda = 2$ has algebraic multiplicity 2. Solving $(A - 2I)v = 0$ yields the eigenvector

$v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ For the generalized eigenvector, solve $(A - 2I)v_2 = v_1$ to

get $v_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Thus $J = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}$ and $P = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ (b) The matrix exponential satisfies

$e^{At} = Pe^{Jt}P^{-1}$ First, $P^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}$ since $\det P = -1$ for the Jordan block.

$e^{Jt} = e^{2t} \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix}$ Multiplying gives $e^{At} = \begin{pmatrix} (t+1)e^{2t} & -te^{2t} \\ te^{2t} & (1-t)e^{2t} \end{pmatrix}$

(c) The solution is $x(t) = e^{At}x(0) = \begin{pmatrix} (t+1)e^{2t} \\ te^{2t} \end{pmatrix}$

Figure 27: Realism test - Real test sample

3.2.2 Platform

The pilot run was carried out with the help of ETH's Decision Science Laboratory. Given that human participants are required, the methodology of our run had to be reviewed by the ETH Zurich Ethics Commission, prior to the start of the run. To that end, a proposal containing Study Objectives, Study Design, intended participant filtering and Ethical Aspects was submitted. Ethical Aspects concerned: Participant Consent and Briefing, Risks, Data Protection and Compensation. The Ethics Commission approved our study without reservations.

Following this, ETH's Decision Science Laboratory provided a *Qualtrics* survey and implemented the necessary logic in accordance with the experiment methodology. The test run was performed in two

separate rounds using different participant pools. The first round utilized the *Prolific* participant pool, which contains participants from a very wide, international background. Filtering was made to ensure all drawn participants knew the Latin alphabet. In the second round, participants were drawn from the so-called UAST participant pool, which contains students from the University of Zurich and ETH Zurich. Once more, it was ensured all participants knew the Latin alphabet. The reason two runs were conducted, was to confirm no logical issues were present in the experiment setup or the survey.

Within the survey, participants were shown the below instructions:

You will be presented with images of math exercises and their handwritten solution. Some of these solutions were handwritten by humans, others were artificially generated.

Your task is to classify the realism of the handwriting in the solution part. You do not have to assess the content itself, only whether the handwriting looks convincing or not - no math knowledge required.

The exercises are in English, German, French and Italian. It is not an issue if you don't speak some of these languages, you need only judge the look of the handwriting.

The selection panels to give ratings for group 1 and group 2 respectively, may be found in Figure 28, these were always shown in conjunction with one or two samples, depending on the group.

How real or artificial does this image seem to you?

☐ This image looks completely artificially generated

☐ This image looks somewhat artificially generated

☐ Uncertain

☐ This image looks somewhat real

☐ This image looks completely real



Which of the two images looks artificially generated to you?

☐ Top image

☐ Bottom image



Figure 28: Realism test - Rating panels (Group 1: Top and Group 2: Bottom)

3.3 Results

As previously stated, the pilot run was carried out in two rounds. Since no changes were made to the experimental setup after the pilot, all data was included in the analysis. This section first details the quantitative analysis, which serves as the test verdict and allows making a claim about data realism. Afterwards, a qualitative analysis is performed to evaluate reasoning participants have given about their ratings. For replicating these analyses, refer to Appendix G.4.

3.3.1 Quantitative Analysis

The quantitative analysis is done in accordance with the methodology and treats the two groups separately.

3.3.1.1 Group 1

The gathered data is summarized in Table 14.

Pool	Participants	Total Generated	Total Real	Mean Generated	Mean Real
Prolific	15	225	225	3.08	3.92
UAST	26	390	390	2.42	3.97
Combined	41	615	616	2.66	3.95

Table 14: Realism test - Group 1 data overview

There is an interesting disparity between the two participant populations in the means of the generated samples. The mean of generated samples in the UAST-Pool is much lower than that of the Prolific-Pool, suggesting that Prolific-Participants fared worse on average. For real samples, the means are comparable across pools. Prior to running the full test procedure outlined in the methodology, it was therefore decided to analyse this disparity.

Running a two sided *Mann-Whitney U test* between the populations for the ratings of the generated and real samples results in p -values of $1.22 \cdot 10^{-8}$ for the generated samples and 0.325 for the real ones. Given that $p = 1.22 \cdot 10^{-8} \ll 0.05$, this heavily suggests that for the generated samples, the ratings of the two populations do not stem from the same distribution. This points to a difference between the two populations, suggesting that factors unaccounted for influenced ratings. A potential factor might be a dependence between ratings and time spent per answer. As can be seen in Figure 29 the R^2 for the regression of the rating on the logarithm of time is extremely low. Such a dependence can therefore be ruled out.

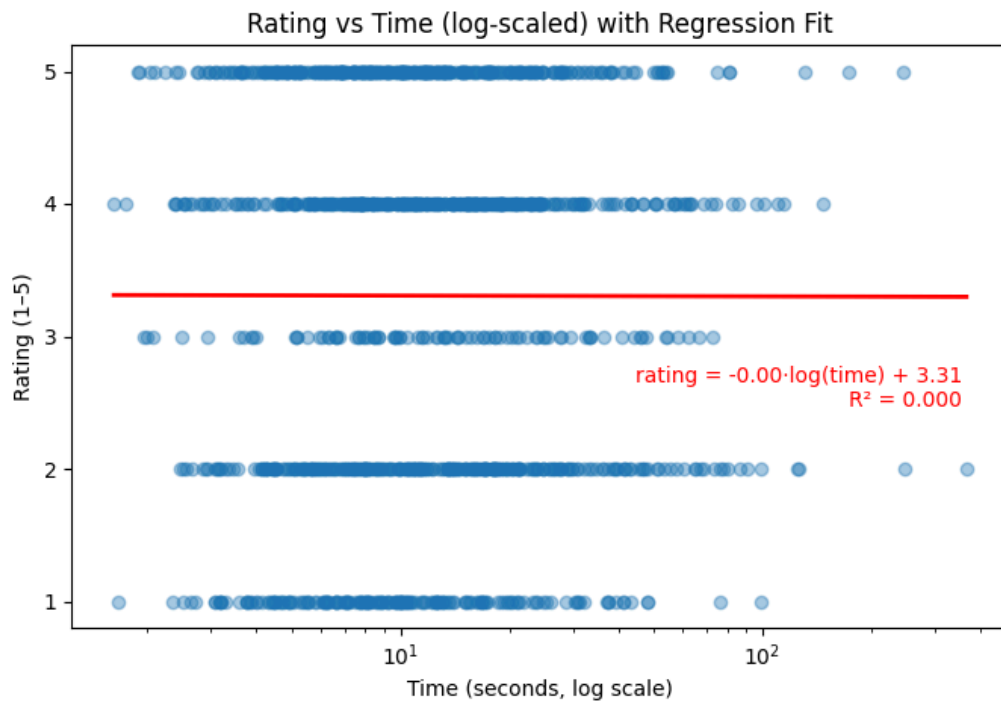


Figure 29: Realism test - Group 1: time vs. rating scatter plot

We therefore assume the observed disparity regarding the means of generated samples comes from the different backgrounds of the two populations (international survey takers vs. Swiss university background). To know for sure, more sensitive data about the participants would need to be gathered (e.g. education, familiarity with AI generated images of text). Nevertheless, combining the data from both populations yields more representative data for the remainder of this analysis.

In Figure 30 the rating distributions of generated and real samples of both populations are illustrated. The means are indicated with dashed lines and their standard deviations with shaded areas.

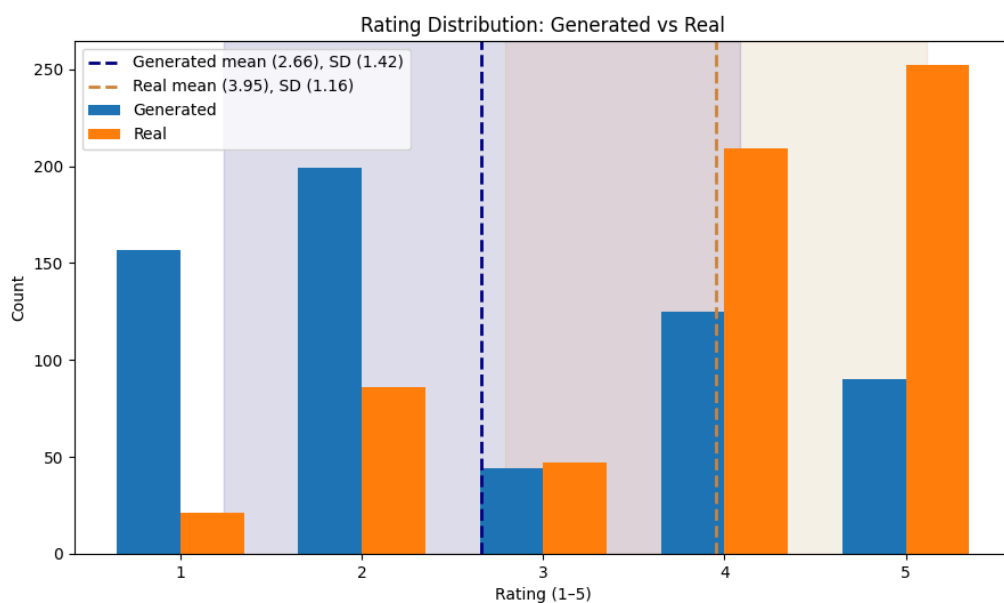


Figure 30: Realism test - Group 1: rating distributions

These distributions are visually distinct. Of note is the low frequency of answer 3 (“Uncertain”) relative to its neighbors.

The results of the *Kolmogorov-Smirnov-Test* according to Section 3.1 are illustrated in Figure 31. The maximum difference is 0.405 with a p -value of $1.88 \cdot 10^{-45}$ at the rating 2. This results in the rejection of the null hypothesis $H_0 : F_X(x) = F_Y(x) \quad \forall x$. Therefore $H_1 : F_X(x) \neq F_Y(x) \quad \exists x$ is accepted. This means that the rating distribution of the generated samples differs significantly from the rating distribution of the real samples.

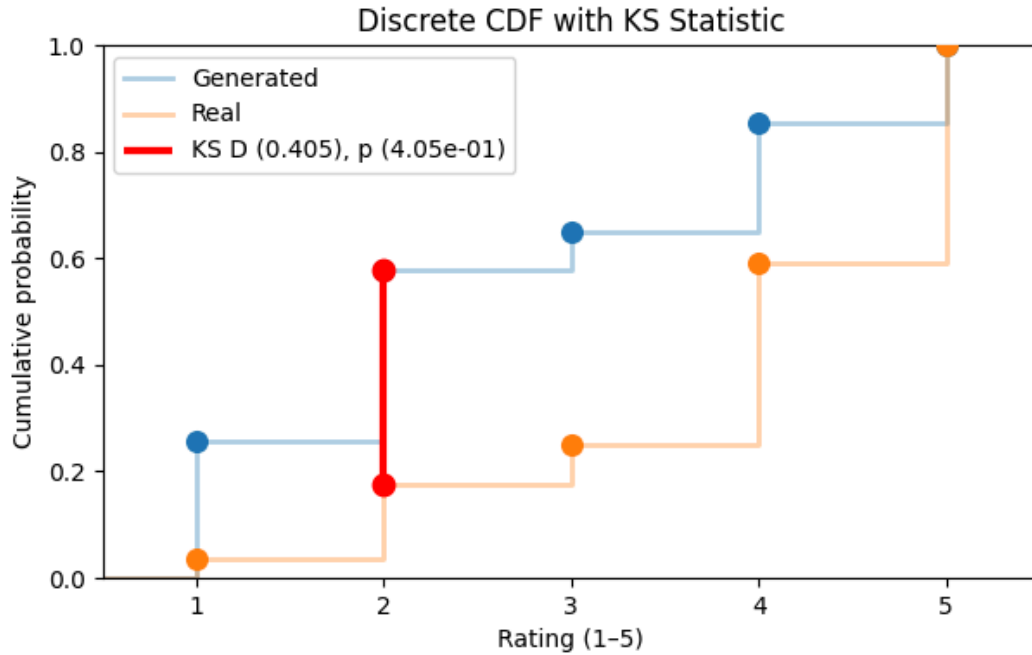


Figure 31: Realism test - Group 1: Kolmogorov-Smirnov-Test

Applying the t -Test according to the same Section 3.1 for $H_0 : \mu_Y \not< \mu_X$ results in a rejection of null hypothesis with a p -value of $7.76 \cdot 10^{-56}$. Therefore $H_1 : \mu_Y < \mu_X$ is accepted. This means that the generated samples are not rated higher or equal to the real samples on average, but significantly lower.

Both test results together establish significant statistical evidence toward distinguishability of real and generated samples.

3.3.1.2 Group 2

The gathered data is summarized in Table 15.

Pool	Participants	Total Comparisons	Thereof correct	Proportion
Prolific	16	240	202	0.842
UAST	21	315	281	0.892
Combined	37	555	483	0.870

Table 15: Realism test - Group 2 data overview

Like in Group 1, participants from the Prolific-Pool tend to fare worse, but the disparity in Group 2 is smaller. Once more, it was decided to combine data from both populations to obtain more representative data.

Applying the *binomial test* according to Section 3.1 on the combined data results in a rejection of the null hypothesis $H_0 : p = 0.5$ with a p-value of $1.02 \cdot 10^{(-75)}$. Therefore $H_1 : p \neq 0.5$ is accepted. This is further evidence toward distinguishability of real and generated samples.

3.3.2 Qualitative Analysis

Like before, the qualitative analysis is performed separately for the two groups. Starting point is user feedback that was collected for some of the ratings participants have given. After submitting all ratings, they were shown three (pairs of) sample(s) and the rating and were then asked what made them give this rating. This feedback was then evaluated using *spaCy* and *agglomerative clustering*. Through the use of *spaCy*, important keywords, so-called tokens, were extracted from user feedback. These tokens are then embedded using *all-MiniLM-L6-v2*, which maps them to a 384-dimensional dense vector space. Token vectors were then grouped into semantically meaningful clusters, using agglomerative clustering. A single participant feedback may therefore be associated with multiple clusters.

3.3.2.1 Group 1

In group 1, 109 of 118 total feedbacks were collected for samples, that users rated particularly low, i.e. as having a particularly generated look. A total of 573 distinct tokens were extracted and 6 different clusters were necessary to group them in a meaningful manner. The identified clusters are: Legibility, Realism, Size, Structure, Uncertainty and Uniformity. All tokens present in each cluster may be found in Appendix E.1.

The total frequency with which each cluster was named, separated by label, is illustrated in Figure 32.

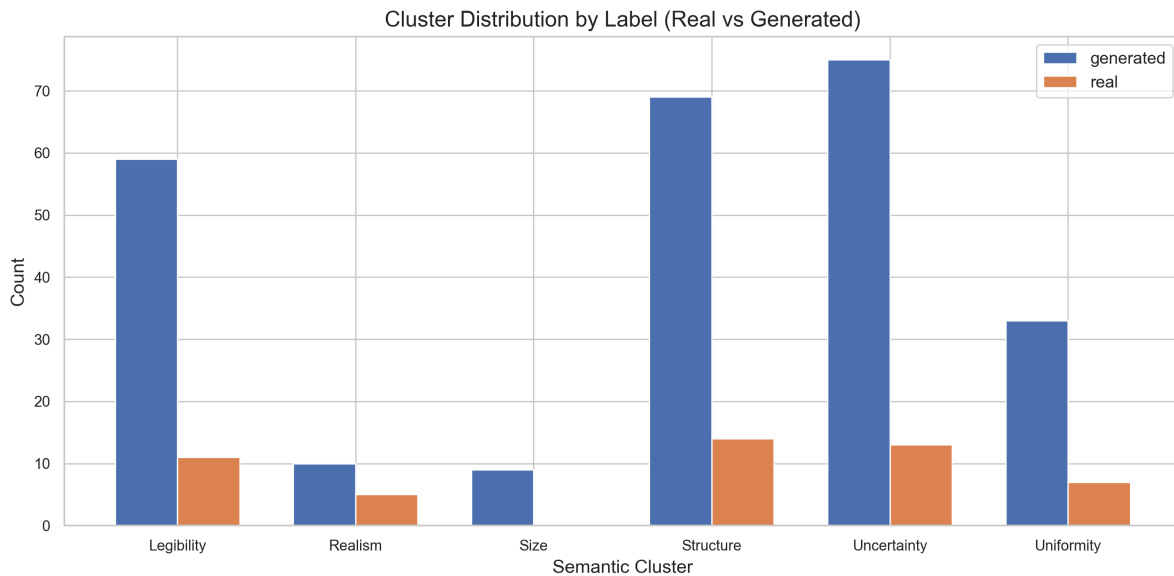


Figure 32: Realism test - Group 1: semantic cluster distribution by label

Given that feedback was collected for samples that were rated as having a particularly generated look and the fact that participants are, mostly, able to identify generated data (see Section 3.3.1), most Group 1 feedback was collected for generated samples, which explains the imbalance between labels. It is apparent that the Uncertainty cluster had the highest total number of occurrences. This cluster is associated with phrases such as: “I don’t know” or “I am not sure”. Other large clusters are Structure, with phrases such as: “Looks too straight” or “Lines too clean”, and Legibility with phrases like “Symbols look too symmetrical” or “Handwriting too clean”. The smaller clusters Uniformity, Realism and Size are associated with phrases regarding consistency and a lack of variations, robotic or artificial looks as well as characters that are too large or small.

Figure 33 shows a heatmap displaying the same cluster frequency distribution, however separated by point on the Likert scale and normalized by the total number of feedbacks collected. Note: one user feedback may contain phrases associated with multiple clusters, so the total number of all cluster occurrences is higher than the total number of feedbacks. This means the sum of entries per column does not equal 1.

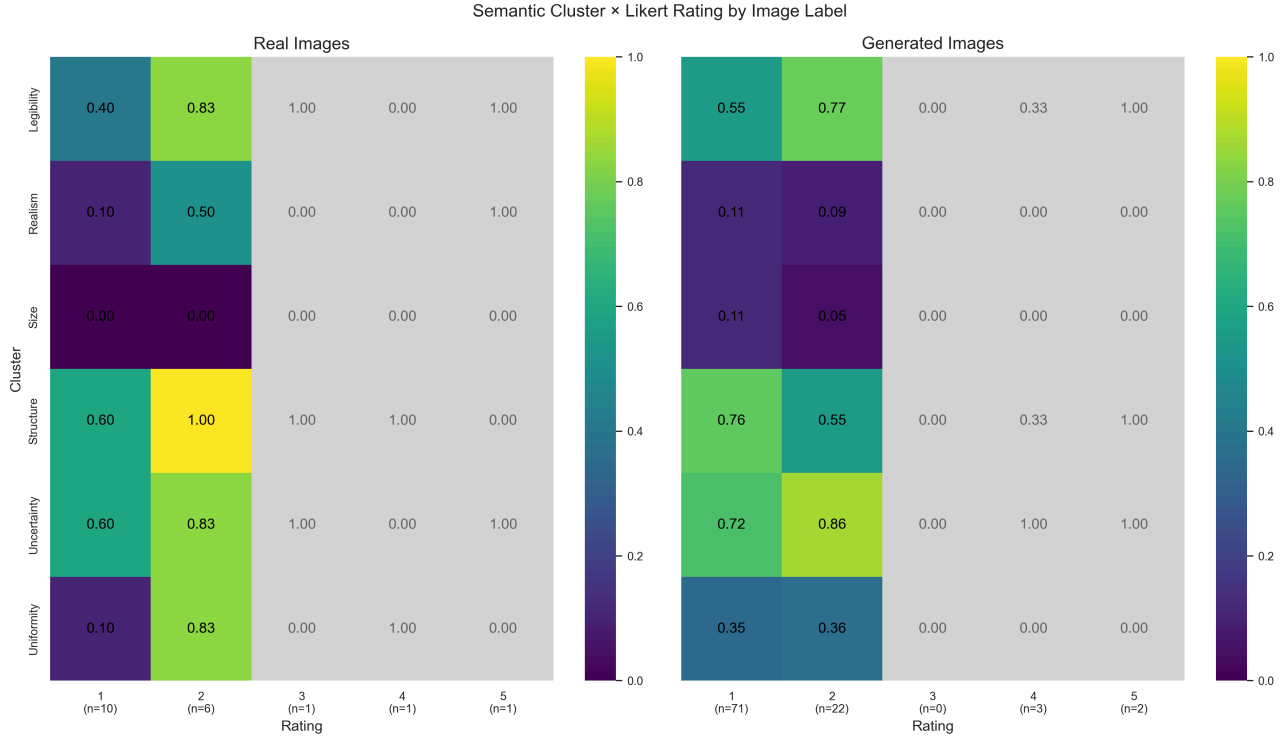


Figure 33: Realism test - Group 1: semantic cluster distribution heatmap

The labeling of the columns shows: a) the point on the Likert scale and b) the number of feedbacks collected for that point on the scale. Greyed out entries are to be disregarded due to low amounts of feedback in those columns. The pattern for both real and generated samples is roughly the same: Uncertainty remains the most frequent cluster, followed closely by Structure and Legibility. Uniformity, Realism and Size are named notably less frequent. Interestingly, Uncertainty is heavily present, despite most ratings belonging to the points 1 or 2 on the Likert scale. Participants are seemingly able to give very clear ratings, but are uncertain about justifying them.

3.3.2.2 Group 2

In group 2, 58 of 106 total feedbacks were collected for samples where participants failed to correctly identify the generated document. A total of 607 distinct tokens were extracted, which were grouped into 4 semantically meaningful clusters. These are: Legibility, Realism, Structure and Uniformity. All tokens present in each cluster may be found in Appendix E.2.

Figure 34 displays the frequency with which cluster were named, separated by whether the associated ratings were correct or not.

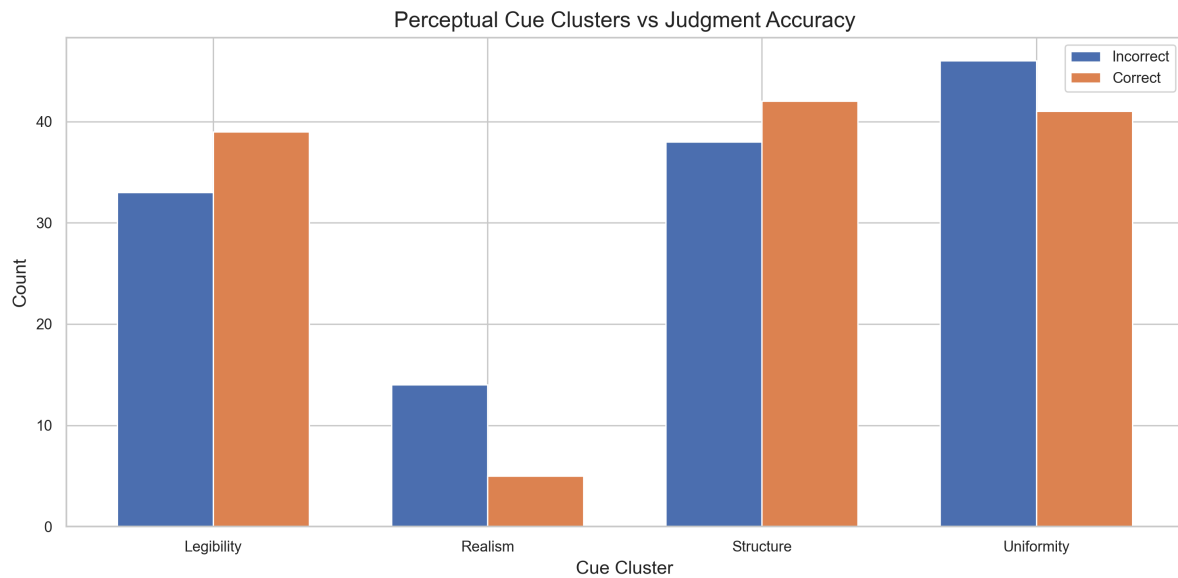


Figure 34: Realism test - Group 2: semantic cluster distribution by correct/incorrect ratings

Interestingly, Uncertainty and Size, which were present in Group 1, are not present at all in Group 2. Especially Uncertainty is surprising, since it was the largest cluster in Group 1.

Within the clusters that are present in Group 2, the ratios between correct and incorrect ratings are very balanced. Clusters were named with the same frequency, regardless of whether participants correctly identified the generated sample or not. This means the same phrases were used to justify correct and incorrect ratings. Behind this fact is likely, once more, Uncertainty. Unlike in Group 1, in Group 2 Uncertainty was not directly declared but rather presents itself as a hidden root cause: Feedback is the same, regardless of correctness. In contrast, if participants were certain about their justifications, one would expect certain clusters to be strongly associated with correct ratings and others with incorrect ratings.

4 Conclusion

Conclusions are drawn independently for: the evaluation of generative AI, the Model-Judge, the realism experiment and finally the deterministic pipeline improvements.

4.1 Evaluation of Generative AI

Our work has shown that all three approaches are not practically integrable into the pipeline, yet deliver promising results in principle. When augmenting entire documents at once, models permitted by the license constraints faced severe difficulties. However, more capable (and non-permitted) models like *Nano Banana Pro* were capable of accurately recreating most words and characters.

When augmenting document subsections, the results were far better. Once again, models like *Nano Banana Pro* produced very high-quality results which would already be production-ready, but are not allowed due to license constraints (see Section 1.5). But even permitted models, like *FLUX.2-dev*, are capable of producing high-quality results. They are, however, not capable of doing so at a consistent rate, augmentation quality has a large variability. There are plenty of augmented outputs that contain hallucinations or entirely illegible symbols, especially when augmenting entire documents at once.

As for the font generation, it seems that current state-of-the-art models are not yet capable of generating fonts in handwriting styles. Nevertheless, the majority of characters created with *FLUX.2-dev* were legible as well as hallucination-free and many characters generated with *Nano Banana Pro* had a realistic handwriting style.

While full practicality cannot be reached at the time being, more capable models, unsurprisingly, produce better outputs, suggesting that even more capable models releasing in the future will be capable of fulfilling these tasks at a production-ready rate. Our results therefore establish the theoretical feasibility of integrating generative AI into the handwriting data generation pipeline. This work serves as a valuable starting point for further work surrounding generative AI and the data generation pipeline (see also Section 5).

4.2 Model Judge

The proposed Model Judge Framework delivers positive results, judges are capable of achieving good performance when filtering hallucinations. Not all evaluated judges achieve that however, *Qwen3.5-397B-A17B* was eliminated as a viable Single-Model judge, due to its comparatively high FPR. *Qwen3.5-27B* and *Kimi K2.5* both perform well as independent judges, *Qwen3.5-27B* achieves the best overall Single-Model performance in terms of FPR and Precision, however *Kimi K2.5* has a significantly better Accuracy and TPR.

In the Multi-Model Setups, MMJ1 and MMJ3 perform comparably to *Qwen3.5-27B*. MMJ3 slightly outperforms *Qwen3.5-27B* as a Single-Model judge. Similarly, MMJ2 is comparable to *Kimi K2.5* and performs slightly better in all but TPR.

The two strongest performances were therefore achieved with MMJ2 and MMJ3. MMJ3 achieves marginally better values for the FPR and Precision, but has a worse Accuracy and TPR. Depending on the kind of model-to-be-trained on the handwriting dataset, as well as the dataset size, MMJ3’s more “aggressive” strategy may be preferable to MMJ2’s more “balanced” approach. The expected performance increase compared to Single-Model Judges was thus achieved.

These results suggest that although viable, model selection for the judges must be made carefully to ensure maximum return. Multi-Model Judge setups do slightly improve performance compared to Single-Model Judges, however given that Single-Model Judge performances can already be quite strong, the additional computational cost may not be justified. Once generative approaches graduate

from theoretical feasibility to full practicality, the Model-Judge setup can serve as an efficient tool to filter hallucinations. This way, it is ensured that samples match their ground truths in terms of content and thus dataset quality is increased.

4.3 Realism Assessment Experiment

For ensuring realism of generated data, we developed a realism assessment test in a Turing-Test like setup. The viability of this benchmark has been demonstrated through the conducted pilot run, which was performed with generated data from the generation pipeline after the increments described in Section 2.1. This data is visibly not realistic enough to be described as “real, human handwriting” and so the test must report that. After analysing test run data, significant evidence of distinguishability was observed, which legitimizes the test methodology.

One surprising test result came from the analysis of qualitative feedback. In Group 1, where participants were shown a single document at a time, the largest semantic feedback cluster was Uncertainty, meaning feedback that contains phrases such as: “I’m not sure”, “I don’t know”, “Might/Could be xyz”. Importantly, the Uncertainty feedback cluster does not correspond to uncertainty in rating. Participants often correctly identified generated samples, but explained their rating using Uncertainty-phrases.

In Group 2, where participants were shown two documents in a side by side comparison, Uncertainty was not present as a feedback cluster at all. However, reasoning was often the same, regardless of whether participants correctly identified the generated sample or not. This means phrases such as: “Line formatting is too rigid”, were used regardless of whether participants were correct or not. Once more, participants often correctly identified the generated sample. This suggests that Uncertainty in reasoning is still present, but presents itself differently than in Group 1. It seems the side-by-side-mode encourages participants to create some reasoning, as opposed to declaring Uncertainty about their rating, nonetheless, participants are uncertain about their reasoning.

In conjunction with the quantitative evaluation, this suggests participants are reliably able to distinguish real and generated samples, but are not able to explain their reasoning. It appears rating realism takes place as a mostly subconscious process, which, although quite accurate, cannot be verbalized or explained (see also Section 5). This reduces the utility of qualitative feedback, as it may not be used to derive strategies on future working directions regarding the data generation pipeline.

The user study provides evidence that the generated handwriting is perceptually distinguishable from real handwriting, thereby demonstrating the sensitivity of our evaluation method.

4.4 Deterministic Pipeline Improvements

Lastly, some deterministic increments, as outlined in Section 1.4, were made to the pipeline. Namely the addition of typed problem statements as suggested by the Apertus-team, which makes generated samples look more “exam-style”. Second is the introduction of Gaussian blurs, to break up rigid line formatting and the addition of some postprocessing to emulate effects introduced by scanning a physical document. Additionally, a demo-client, with which pipeline functionality can be shown, was developed. This demo-client works by replacing the LaTeX-generation step in the pipeline, with a LaTeX file that users pass as input. The input file is then passed through the remainder of the pipeline. This way, users have full control of the content of generated samples. Furthermore, users may also handwrite the content of the LaTeX-file manually and directly compare that to the generated version for a side by side comparison.

5 Outlook

Given the theoretical feasibility of our generative approaches, further work is required to evaluate: finetuned models or entirely different model architectures, as long as they are integrable into the data generation pipeline. Through this, these approaches may yield the necessary practicality to be usable in the data generation pipeline. Additionally, models releasing after the date of this thesis, may be capable of achieving the consistency in quality that current-day models failed at. Moreover, the performance of the data generation pipeline, compared to other handwriting generation paradigms, is an open question. Further work may utilize our realism assessment procedure to answer this question.

Further work may also concern the proposed Model Judge setup. Besides testing different models as judges, new judge-setups can be evaluated. Our work introduced the Multi-Model Judge with three models, this is easily extendible to five models. More than five models is probably not worth the larger computational cost though. Additionally, a piped Multi-Model setup may be tested. Here, a first model operates as a Single-Model Judge, all data that is not discarded by it is piped to a second model, once again operating as a Single-Model Judge. This can be repeated by chaining any number of models, though once more, at some point the computational cost becomes too large at some point. In this setup and under the assumption of independent errors, the total False Positive Rate is the product of the False Positive Rates of the individual judges:

$$\text{FPR}_{\text{tot}} = \text{FPR}_1 \cdot \text{FPR}_2 \cdot \dots \cdot \text{FPR}_n,$$

similarly, the total True Positive Rate is the product of the True Positive Rates of the individual judges:

$$\text{TPR}_{\text{tot}} = \text{TPR}_1 \cdot \text{TPR}_2 \cdot \dots \cdot \text{TPR}_n.$$

Ideally, FPR_{tot} should be close to 0 and TPR_{tot} close to 1, however the False Positive Rate is the more important objective. Therefore, there must be at least one judge with a very low False Positive Rate and all of the judges should have decently high True Positive Rates. Through these setups, Model Judge performance may increase further. Another potential approach would be iterative refinement. Currently, incorrect samples (i.e. with hallucinations) are simply discarded. As opposed to that, one might also prompt the Judge for a description of the hallucination and then re-augment the sample with the description of the hallucination as a negative prompt.

The pilot run of the realism assessment procedure conducted within this thesis returned surprising results in the qualitative analysis. As previously stated, results suggest that participants are unable to reason about their ratings, which disqualifies qualitative feedback as a legitimate source of information. This issue may be resolved by specifically seeking participants experienced in style recognition, calligraphy or related fields. Should these participants be able to produce more valuable qualitative feedback, this feedback can be used to derive strategies regarding future work on the data generation pipeline.

Given the future objective of training Apertus on generated handwriting, further work may also concern directly training a model on generated handwriting and then testing on real handwriting. Ultimately, only this approach can demonstrate whether models can generalize from generated to real data. Nevertheless, successful generalization benefits from general data similarity, which is, therefore, an important step.

References

- [1] Rosa Mayila, David Schulmeister, and Amene Gafsi, “Handwriting Data Generation and Augmentation for Mathematical Content,” 2024. Accessed: 2026. [Online]. Available: https://gitlab.ost.ch/ba2026-handwriting/code/-/blob/main/_previous/report.pdf?ref_type=heads
- [2] Yousef Elarian, Radwan Abdel-Aal, Irfan Ahmad, and Mohammad Tanvir Parvez, “Handwriting Synthesis: Classifications and Techniques,” INTERNATIONAL JOURNAL OF DOCUMENT ANALYSIS AND RECOGNITION, Sept. 2014. Accessed: 2026. [Online]. Available: https://www.researchgate.net/publication/266323642_Handwriting_Synthesis_Classifications_and_Techniques
- [3] Philippe Gervais, Asya Fadeeva, and Andrii Maksai, “MathWriting: A Dataset For Handwritten Mathematical Expression Recognition,” Google Research, Apr. 2024. Accessed: 2026. [Online]. Available: <https://huggingface.co/datasets/deepcopy/MathWriting-human>
- [4] Pearson, “Aida Calculus Math Handwriting Recognition Dataset,” Pearson, 2020. Accessed: 2026. [Online]. Available: <https://www.kaggle.com/datasets/aidapearson/ocr-data>
- [5] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee, “Visual Instruction Tuning,” Apr. 2023. Accessed: 2026. [Online]. Available: <https://huggingface.co/datasets/shreyansh1347/GNHK-Synthetic-OCR-Dataset>
- [6] U. Marti and H. Bunke, “The IAM-database: An English Sentence Database for Off-line Handwriting Recognition,” Int. Journal on Document Analysis and Recognition, 2002. Accessed: 2026. [Online]. Available: <https://www.kaggle.com/datasets/nibinv23/iam-handwriting-word-database>
- [7] Moises Diaz, Andrea Mendoza-García, Miguel A. Ferrer, and Robert Sabourin, “A survey of handwriting synthesis from 2019 to 2024: A comprehensive review,” Pattern Recognition Volume 162, June 2025. Accessed: 2026. [Online]. Available: <https://doi.org/10.1016/j.patcog.2025.111357>
- [8] Wei Pan, Huiguo He, Hiuyi Cheng, and Lianwen Jin, “DiffInk: Glyph- and Style-Aware Latent Diffusion Transformer for Text to Online Handwriting Generation,” ICLR, Apr. 2026. Accessed: 2026. [Online]. Available: <https://liner.com/review/diffink-glyph-and-styleaware-latent-diffusion-transformer-for-text-to>
- [9] “GitLab Code Repository of this Thesis.” [Online]. Available: <https://gitlab.ost.ch/ba2026-handwriting/code>
- [10] Yusu Qian *et al.*, “Pico-Banana-400K: A Large-Scale Dataset for Text-Guided Image Editing,” CVPR, 2025. Accessed: 2026. [Online]. Available: <https://www.arxiv.org/pdf/2510.19808>
- [11] Randa Elanwar and Margrit Betke, “Generative adversarial networks for handwriting image generation: a review,” Vis Comput 41, 2025. Accessed: 2026. [Online]. Available: <https://doi.org/10.1007/s00371-024-03534-9>
- [12] Thrishaa J, Agamy David, Neha Venkatesh, and Dr. Kiran Y C, “A Survey of Modern Handwriting Generation Models,” IJSAT Volume 16 Issue 4, Dec. 2025. Accessed: 2026. [Online]. Available: <https://doi.org/10.71097/IJSAT.v16.i4.9815>
- [13] “black-forest-labs/FLUX.2-dev.” Accessed: June 09, 2026. [Online]. Available: <https://huggingface.co/black-forest-labs/FLUX.2-dev>

- [14] Clement Farabet, Olivier Lacombe, and Google DeepMind, “Gemma 4: Byte for byte, the most capable open models.” Accessed: June 09, 2026. [Online]. Available: <https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/>
- [15] OpenAI, “The new ChatGPT Images is here.” Accessed: June 09, 2026. [Online]. Available: <https://openai.com/index/new-chatgpt-images-is-here/>
- [16] Kimi Team *et al.*, “Kimi K2.5: Visual Agentic Intelligence,” 2026. Accessed: 2026. [Online]. Available: <https://arxiv.org/abs/2602.02276>
- [17] Alisa Fortin and Naina Raisinghani, “Build with Nano Banana Pro, our Gemini 3 Pro Image model.” Accessed: June 09, 2026. [Online]. Available: <https://blog.google/technology/developers/gemini-3-pro-image-developers>
- [18] Chenfei Wu *et al.*, “Qwen-Image Technical Report,” 2025. Accessed: 2026. [Online]. Available: <https://arxiv.org/abs/2508.02324>
- [19] Qwen Team, “Qwen3.5: Towards Native Multimodal Agents,” Feb. 2026. Accessed: 2026. [Online]. Available: <https://qwen.ai/blog?id=qwen3.5>
- [20] “GitLab Documentation Repository of this Thesis.” [Online]. Available: <https://gitlab.ost.ch/ba2026-handwriting/documentation>

Glossary

The glossary contains definitions of technical terms used in this document.

Agglomerative clustering Agglomerative clustering is a hierarchical clustering method that starts with each data point as an individual cluster and merges the closest clusters iteratively based on a chosen distance metric until all points are combined into a single cluster or a stopping criterion is met.

All-MiniLM-L6-v2 This is a sentence-transformers model: It maps sentences & paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search.

Apertus Apertus is a Swiss public large language model, developed by the Swiss National AI Initiative. It uses only fully compliant and open training data, and achieves comparable performance to models trained behind closed doors.

Binomial test A binomial test is a statistical method used to determine if the proportion of successes in a sample differs from an expected proportion in a binary outcome scenario.

Binomial test See *Binomial test*

CLI-tool Stands for Command-Line-Interface-tool. This refers to a program that does not have a graphical user interface and is instead executed from a terminal.

CSCS Centro Svizzero di Calcolo Scientifico is the national high-performance computing centre of Switzerland. It operates as an autonomous unit of ETH Zürich.

Cumulative distribution functions In probability theory and statistics, the cumulative distribution function of a real-valued random variable X , or just distribution function of X , evaluated at x , is the probability that X will take a value less than or equal to x .

Data generation pipeline Data generation pipeline refers to ETH Zurich's LaTeX-compiler based pipeline, which creates synthetic handwriting samples.

Diffusers-library Diffusers is a library of state-of-the-art pretrained diffusion models for generating videos, images, and audio, developed by Hugging Face.

Diffusion-style models Diffusion-style models are a type of generative model used in machine learning, primarily for tasks like image generation and image editing.

Docker-Container A Docker container is a standardized unit of software that packages up code and all its dependencies, allowing the application to run quickly and reliably in different computing environments.

Enroot A simple, yet powerful tool to turn traditional container/OS images into unprivileged sand-boxes.

Ethel Ethel is a virtual teaching assistant developed by the ETH AI Center. The Ethel ecosystem has a number of components that support teaching and learning at ETH Zurich and EPFL. These include a reference-based chatbot and homework feedback among others.

Feature vectors Feature vectors in machine learning are numerical representations of data that describe measurable properties or characteristics of an object.

FLUX A family of image-generation models developed by Black Forest Labs.

Flux See *FLUX*

FLUX.2-dev A 32B parameter rectified flow transformer for text-to-image generation and image editing, supporting multi-reference inputs, high-resolution outputs, and strong prompt adherence including reliable text rendering. [13]

Fréchet Inception Distance The Fréchet Inception Distance (FID) is a metric used to evaluate the quality of images generated by models like GANs or diffusion models by comparing their distribution to real images using deep learning features.

GANs Generative Adversarial Networks (GANs) are a type of machine learning model that consists of two neural networks: a generator that creates data and a discriminator that evaluates its authen-

ticity. They work in opposition to improve each other, with the generator aiming to produce realistic data that can fool the discriminator.

Gaussian blurs Gaussian blur is a type of image-blurring filter that uses a Gaussian function to smooth out an image by reducing noise and detail.

Gemma4-31B-it A dense instruction-tuned multimodal large language model from the Gemma 4 family, designed for high-performance reasoning, coding, and agentic workflows with long-context and vision capabilities. [14]

Generative AI Generative AI, or generative artificial intelligence, is a type of AI that creates new content such as text, images, audio, or video based on patterns learned from existing data.

GitLab GitLab is a source control tool used to host and version software. It is used as software and documentation repository within this project.

Gpt-image-1.5 OpenAI’s natively multimodal image generation and editing model, capable of synthesizing and modifying images from text and image inputs with strong prompt adherence and text rendering capabilities. [15]

Kimi K2.5 A multimodal large language model developed by Moonshot AI with visual understanding, reasoning, and agentic capabilities. [16]

Kolmogorov-Smirnov-Test The Kolmogorov-Smirnov test is a nonparametric statistical test used to compare two probability distributions or to test whether a sample comes from a specific distribution.

Kullback-Leibler Divergence Kullback-Leibler divergence is a measure of how one probability distribution diverges from a second, reference probability distribution.

LaTeX LaTeX, is a software system for typesetting documents, based on the TeX typesetting system.

Likert scale A Likert scale is a psychometric tool used in surveys to measure attitudes or opinions by asking respondents to rate their level of agreement with specific statements, typically on a scale of five or seven points.

LLM LLM stands for Large-Language-Model. LLMs are computational models trained on a vast amount of data, designed for natural language processing tasks, especially language generation.

Mann-Whitney U test The Mann-Whitney U test is a nonparametric statistical test used to compare differences between two independent groups when the data is ordinal or continuous but not normally distributed. It assesses whether the distributions of the two groups are equal, focusing on the ranks of the data rather than the actual values.

Math-mode Math mode in LaTeX is a special environment used for typesetting mathematical expressions. It allows users to write equations and symbols clearly.

MIT-License The MIT License is a permissive license that allows free use, copy, modification, and distribution of software without restriction.

Nano Banana Pro Google’s Gemini 3 Pro Image model for professional-grade image generation and editing, with advanced reasoning and accurate text rendering. [17]

Open source Open source refers to software whose source code is made available, enabling use, modification, and redistribution

OpenAI API The OpenAI API is a python package that allows prompting previously deployed models.

OpenRouter OpenRouter is a platform that provides developers access to hundreds of models through a single API

OTF Open-Type-Font is a font file format developed by Adobe and Microsoft that supports advanced typographic features and a wide range of characters.

Podman Podman is an open source Open Container Initiative-compliant container management tool created by Red Hat used for handling containers, images, volumes, and pods on the Linux operating system

Prolific Prolific is an online research platform that pays participants for their opinion and feedback.

- Qualtrics** Qualtrics offers a cloud-based subscription software platform for experience management, which it launched in March 2017. Qualtrics software is used widely in academia to conduct survey research
- Qwen** A family of large multimodal and language models developed by Alibaba, designed for tasks including text generation, reasoning, and image understanding/editing across multiple scales and variants.
- Qwen-Image** A multimodal image generation model from Alibaba’s Qwen family designed for high-fidelity text-to-image synthesis and structured visual generation. [18]
- Qwen-Image-Edit** A variant of Qwen-Image specialized for image editing tasks, enabling modification of existing images while preserving structure and semantics. [18]
- Qwen3.5-27B** A 27-billion-parameter variant of the Qwen3.5 model family designed for general-purpose multimodal language understanding and generation. [19]
- Qwen3.5-397B-A17B** A large-scale sparse Mixture-of-Experts variant of Qwen3.5 (397B total parameters, 17B active) optimized for high-capacity reasoning and multimodal tasks. [19]
- RNN** Recurrent neural networks (RNNs) are a class of artificial neural networks designed for processing sequential data, such as text, speech, and time series, where the order of elements is important.
- SGLang** SGLang is a high-performance serving framework for large language models and multimodal models.
- SpaCy** spaCy is a library for advanced Natural Language Processing in Python and Cython. It features state-of-the-art speed and neural network models for tagging, parsing, named entity recognition, text classification and more.
- Squash-File** Squashfs is a compressed read-only file system for Linux.
- SSH** The Secure Shell Protocol (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network.
- T-Test** A t-test is a statistical test used to determine if there is a significant difference between the means of two groups.
- Token** AI tokens are the basic building blocks of text that AI models use to understand and generate information.
- Transformers** Transformers are a type of neural network architecture designed for processing sequential data, primarily used in natural language processing (NLP). They utilize a mechanism called self-attention to weigh the importance of different tokens in a sequence, allowing for efficient parallel processing and improved understanding of context.
- Turing-Test** The Turing test is a measure of a machine’s ability to exhibit intelligent behavior equivalent to that of a human, proposed by Alan Turing in 1950. In this test, a human evaluator interacts with both a machine and a human without knowing which is which.
- VAEs** A variational autoencoder (VAE) is a type of artificial neural network used in machine learning that combines an encoder and a decoder to learn efficient representations of data, allowing it to generate new data similar to the input it was trained on.
- Vision-language-model** A vision-language model is a type of artificial intelligence system that can jointly interpret and generate information from both images and text, extending the capabilities of large language models, which are limited to text.
- Vision-language-models** See *vision-language-model*

Appendices

A Infrastructure Setup

Infrastructure setup for this thesis was quite time-intensive. As platform for any open weight models, such as the *Qwen*- or *Flux*-Families, the Apertus-team provided access to *CSCS*, specifically the *Clariden-Cluster*. This is a Linux-based server-cluster, operated by the swiss national high-performance computing centre. The cluster offers several thousand Nvidia H200 graphics cards and is therefore ideal for running models.

The cluster provides a unified command-line application for launching models, using *SGLang*. Through this method, the cluster automatically provides an API, which is externally reachable to authenticated users. However, after some trial and error, it was determined that this method is only compatible with transformer-style models and not diffusion-style models, which were necessary for our work. Afterwards, it was decided to directly launch models using the Python *diffusers-library*. This required writing a Python script on the cluster, which necessitated a *Docker-Container* containing the software dependencies described in Appendix G. The container then had to be built using *Podman* and saved to a *Squash-File* using *Enroot*. Following that, a compute-node had to be requested using the created *Squash-File*. Compute-nodes are units of computation and contain a specified amount of graphics cards. At times, this was a lengthy process, since many people work on the *Clariden-Cluster*, and thus there is a queue for compute-nodes. Once a compute-node is obtained, it automatically loads the specified container and experiments may begin. Since the cluster is based on a terminal-only Linux-installation, Remote Access from within Visual Studio Code, using *SSH*, was set up, to develop Python-scripts for our experiments.

For closed weight models, such as *Nano Banana Pro*, access to *OpenRouter* was provided by our supervisor. There, models can be prompted using the *OpenAI API*, directly from a Python-script. This method of work is simpler in terms of setup, but comes with additional monetary cost for each *Token* generated.

B Technical Background

This appendix aims to give a high-level overview of the models used, their basic functionality and important parameters. We separate diffusion-style models, used for our evaluation of generative AI, and vision-language models, used for our Model-Judge framework. All models are also listed in the Glossary in a more concise manner.

B.1 Diffusion-Style Models

Diffusion models generate images by iteratively denoising a random noise distribution into a coherent image conditioned on a text prompt or other inputs. During inference, generation quality can be controlled through parameters such as the number of denoising steps, guidance scale (classifier-free guidance), image resolution, and random seed. Modern diffusion systems increasingly replace traditional U-Net architectures with transformer-based diffusion backbones and integrate large language models to improve prompt understanding and image editing capabilities.

Qwen-Image and Qwen-Image-Edit

Qwen-Image is Alibaba’s flagship image generation model and is based on a Multimodal Diffusion Transformer architecture. The model combines a large language model for prompt understanding with a diffusion-based image decoder. Qwen-Image-Edit extends the architecture with image-conditioning mechanisms that allow modifications to existing images through natural-language instructions while preserving unchanged content. Important model parameters:

- **Number of diffusion steps:** Number of iteration steps to perform
- **Guidance:** Controls prompt adherence
- **Output resolution:** Resolution of the final output image
- **Seed:** Used to initialize model, keeping the Seed fixed, results in deterministic output

FLUX.2-dev

FLUX.2-dev is an open-weight diffusion model developed by Black Forest Labs. The model uses a large transformer-based diffusion backbone rather than the traditional U-Net architecture used in Stable Diffusion. With 32 billion parameters, it is one of the largest publicly available image-generation models. Important model parameters:

- **Guidance scale:** Controls prompt adherence
- **Number of denoising steps:** Number of iteration steps to perform
- **Sampling scheduler:** Determines how aggressively noise should be added and removed at each step
- **Image resolution:** Output image resolution
- **Random seed:** Model initialization

Nano Banana Pro

Nano Banana Pro is Google’s latest image generation and editing system built on the Gemini multimodal ecosystem. Unlike classical diffusion models, Google focuses on integrating reasoning, image understanding, and generation into a unified multimodal architecture. The model is built on top of Gemini as a multimodal foundation and has very strong text rendering. Publicly available model details are limited and as such, no overview of important inference parameters can be given.

GPT Image

GPT Image is OpenAI’s image generation and editing model integrated into the GPT multimodal ecosystem. The model combines language understanding, image understanding, and image generation within a single framework. Like Google, OpenAI has not publicly disclosed detailed architectural specifications or parameter counts.

B.2 Vision-Language Models

Vision-Language Models (VLMs) extend large language models with the ability to process images on top of text processing. Typically, an image encoder transforms visual information into embeddings that are then processed together with text tokens by a language model. Modern VLMs support tasks such as image captioning, visual question answering, optical character recognition (OCR), chart understanding, document analysis, and multimodal reasoning.

Qwen3.5-27B

Qwen3.5-27B is a medium-sized multimodal model from Alibaba's Qwen family. It combines a vision encoder with a 27-billion-parameter language model and is designed for image understanding, document analysis, OCR, chart interpretation, and visual reasoning tasks. The model consists of a total of 64 layers, making it very deep.

Qwen3.5-397B-A17B

Qwen3.5-397B-A17B is a large-scale Mixture-of-Experts (MoE) vision-language model. Mixture of experts is a type of ensemble learning, where a problem space is divided into subproblems to be solved by multiple expert networks. The model contains 397 billion total parameters, but only about 17 billion parameters are active per token, reducing inference cost compared to an equivalently sized dense model. Furthermore, it has a total of 512 experts, of which 11 are activated. Architecturally, the model consists of 60 layers.

Kimi K2.5

Kimi K2.5 is a multimodal MoE model developed by Moonshot AI. It extends the Kimi model family with vision-language capabilities and focuses on long-context reasoning, document understanding, and multimodal agent workflows. The model has a total of 61 layers, of which one is dense. K2.5 is capable of decomposing complex tasks and solving them using dynamically instantiated agents. The model has a total of 1T parameters, of which 32B are active, the vision encoder that is responsible for image encoding has a total of 400M parameters.

Gemma4-31B-it

Gemma4 31B-it is Google's open-weight instruction-tuned multimodal model in the Gemma family, with a total of 31B parameters. The model combines image understanding with strong language capabilities while remaining significantly smaller than frontier proprietary systems. Once more, Gemma4 is also an MoE model and therefore capable of subdividing complex problems into smaller tasks. The model has a total of 60 layers in its architecture.

C Existing Pipeline

This section outlines the functionality of the existing data generation pipeline, as it was before the start of this thesis. This pipeline consists of two parts: the data generation part and the font creation part. Refer to Figure 36 for a full visualization.

C.1 Data Generation

For data generation, the following steps are performed:

1. Generate the ground truth of an artifact, in LaTeX-format, utilizing a *LLM*. The model is prompted to create student-like solutions to math exercises.
2. Split the LaTeX document into parts with relative offsets and rotations, using LaTeX-commands to emulate paragraphs in human handwriting.
3. Within the LaTeX-preamble, change the font to a font that was produced from human handwriting (using the font creation part) and modify the page background to resemble a sheet of paper.
4. Additional preprocessing steps to introduce further irregularities:
 - Random, line-level offsets
 - Random, character-level offsets and rotations
5. Compile the processed LaTeX document to PDF
6. Convert PDF to PNG and add noise and blur to emulate a low quality scan.

C.2 Font Creation

The below steps are executed to create fonts:

1. Produce templates in PDF format (see Figure 35 for an example)
 - Around 250 characters are to be filled
 - These consist of the latin and greek alphabets, punctuations, arabic numbers and various mathematical symbols
2. Extract individual characters from templates, once filled
3. Produce a font file

Font files are installable and once installed, behave like any font: they can be selected as font in applications such as Microsoft Word or in this case, used to compile LaTeX-documents. The font creation part of the existing pipeline is therefore only a supporting process, once a font has been created, the data generation part may use it repeatedly and indefinitely.

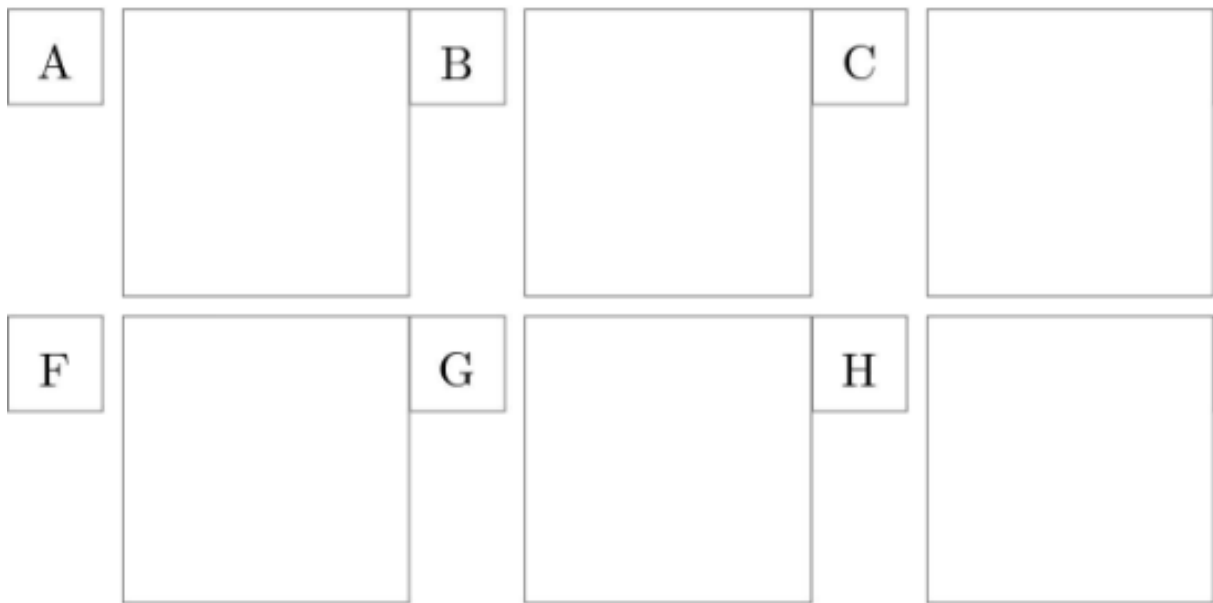


Figure 35: Font creation template

C.3 Pipeline Visualization

The entire pipeline is illustrated in Figure 36. The red elements make up the existing pipeline, whereas the green elements are potential changes induced by generative AI, as described in Section 2.2.

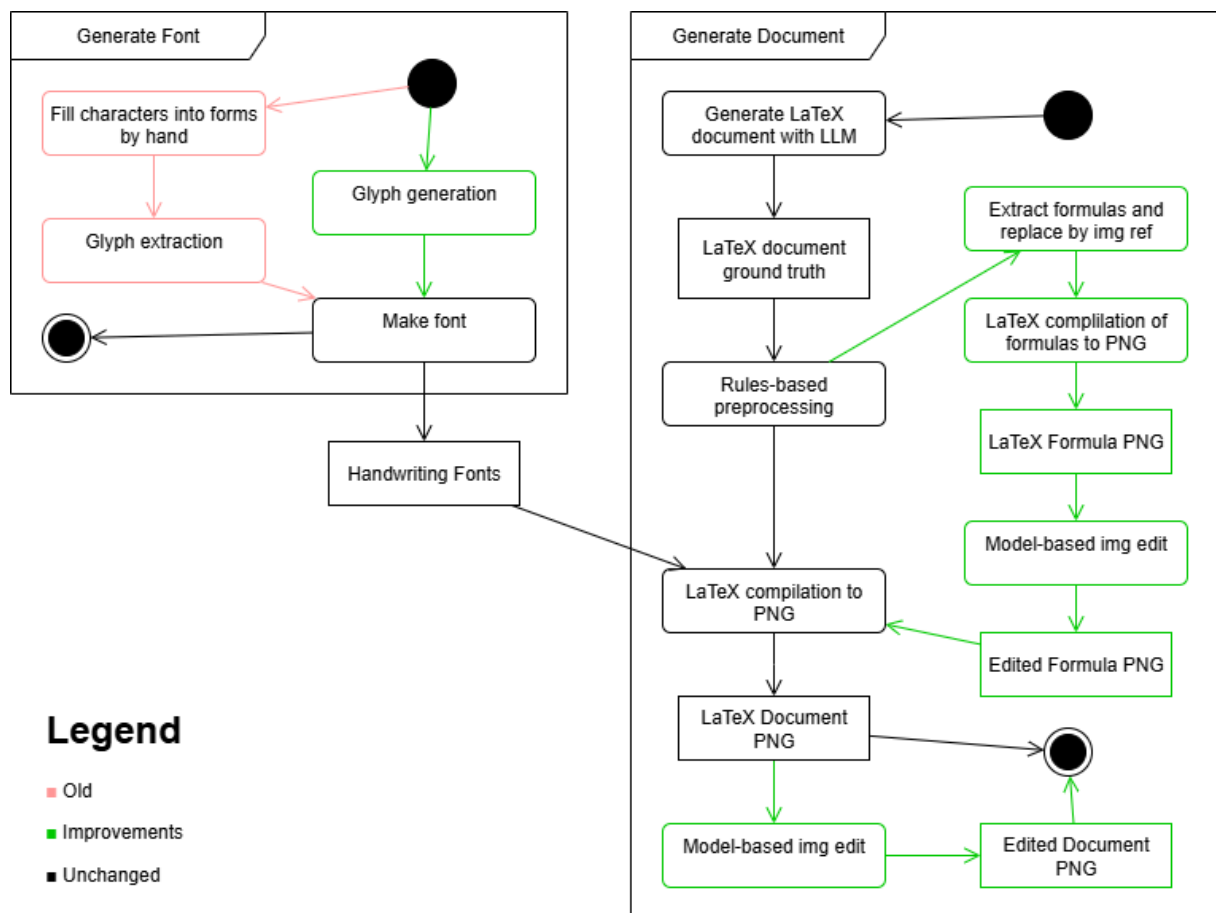


Figure 36: Pipeline overview

D Prompts

This appendix contains all prompts used to derive our results.

D.1 Pipeline Generation Prompt

The prompt used to generate data described in Section 2.1.1. The exercise number is a randomly chosen integer and the mathematics topic is randomly selected from:

- Calculus
- Algebra
- Number Theory
- Discrete Mathematics
- Probability and Statistics

The language is randomly chosen from:

- English
- German
- French
- Italian

Answer strictly in valid LaTeX that compiles. Generate a completely new math exercise with number <insert random integer> and its full solution.

STRUCTURE (replace <solution> with your solution, do NOT deviate from this):

```
\begin{document}
{\fontspec{Computer Font A}\setmathsf{Computer Font A} <statement> }
<solution>
\end{document}
```

PROBLEM STATEMENT REQUIREMENTS:

- Begin immediately with a clearly formatted title, e.g. `\textbf{Exercise <number>}` or a similar emphasized heading.
- The problem must look like a real exam sheet, not a description of an exercise.
- Keep the problem statement relatively short, the majority of your response MUST be made up of the problem solution further below.
- Use natural mathematical phrasing.
- NEVER use meta-descriptions like "this exercise consists of..." or "part a asks...".
- Include multiple parts and assign points to each part. Enumerate the parts. Your parts have to be consistent in terms of content.
- Use rich formatting: bold text, spacing, separators, alignment, indentation, etc.
- The layout must resemble a professionally typeset exam problem, not plain text.
- Enclose the entire problem statement in `{\fontspec{Computer Font A}\setmathsf{Computer Font A} <statement> }`, starting with `{` and properly closing with `}`.

MATHEMATICAL CONTENT:

- The exercise must involve non-trivial mathematics <insert randomly chosen topic>.
- Include meaningful calculations, equations, or proofs.

SOLUTION REQUIREMENTS:

- Write the solution in a realistic "exam solution" style. Do NOT add a title that reads "Solution" or similar.

- Refer to exercise parts if the exercise is split into parts.
- DO NOT use narrative or descriptive prose such as:
 - "I first do..."
 - "we now calculate..."
 - "this step shows..."
- DO NOT refer to yourself ("I", "we", "you") or to "the student".
- Use concise, computation-driven structure
- Focus on mathematical steps, not overt explanations, do include a some though, your solution must still be understandable.
- Use a clean step-by-step structure with equations and only some text.
- Keep your solution SHORT, it MUST fit on half an A4 page. You are allowed to create incomplete solutions in order to fit the length requirement.
- The tone should resemble a stressed student writing down results and steps efficiently.

FORMATTING RULES:

- Every sentence must be on its own line.
- Use inline math modes most of the time. ONLY use display math sparingly and for important results.
- Use spacing and LaTeX structure instead of prose to organize content.

IMPORTANT:

- Do not reuse or adapt previous outputs.
- Create a completely new exercise and solution.
- Both your exercise statement and your exercise solution MUST be in <randomly choose from English/German/French/Italian> language.
- Strike through 1 realistic word mistake (not digit) if needed in the solution/student part of your answer using the `\strikeMistake`. All you have to do is `\strikeMistake{a mistake}`.
- Create one margin note in your the solution/student part of your answer using either `\reversemarginpar{\rotatebox{90}{\Huge Rotated margin note}}` or `\marginpar{\rotatebox{90}{\Huge Rotated reverse margin note}}`. Randomly select which one of the two commands to use. The margin note text should be short, that a student might scribble on as a last minute remark and only be a couple words at most. The margin note should NOT be a title or similar. Do NOT create a separate title to announce the margin note either. Also randomly vary the rotation angle by a couple degrees within the ranges $[-93, -87]$ or $[87, 93]$.
- You should keep the simple default layout. You have to start your answer with the following structure for the LaTeX header: `\begin{document}`

D.2 Document Augmentation Prompt

The prompt used for the font generation described in Section 2.2.1.

Make this formula look like hurried, imperfect handwriting without changing its content. Add irregularities, uneven lines, and random spacing in math parts, but keep it legible and do not add anything new.

D.3 Subsection Augmentation Prompts

The prompt used for the font generation described in Section 2.2.2.

Prompt for runs without styleguide.

Redraw this mathematical formula in a natural handwriting style.

Apply these stroke characteristics:

- thin, variable-width strokes with natural pressure variation
- slight wobble and curvature in lines
- natural inconsistency between similar characters
- pen or pencil feel, not bold or brush-like

Rules:

- Preserve every symbol exactly – no additions, removals, or reinterpretation
- Empty areas must remain empty
- Do not make strokes bold or go over them multiple times

Prompt for runs with styleguide. Some models (*FLUX.2-dev*, *Nano Banana Pro*) need the image to be edited “{a}” to be the first image (“1”) passed to the model. Other (e.g. *Qwen-Image-Edit*) need it to be the last (“2”). “{b}” is filled with the other number (“2” or “1”).

You are given two images:

Image {a}: a mathematical formula (this is the content to preserve exactly)

Image {b}: a handwriting style reference (this provides only the visual stroke style)

Task:

Redraw Image {a} using the handwriting stroke style from Image {b}.

Rules:

- Preserve every symbol from Image {a} exactly – no additions, removals, or reinterpretation
- Empty areas in Image {a} must remain empty
- Apply the stroke characteristics of Image {b}: line thickness, pressure variation, wobble, curvature, natural inconsistency
- Do NOT copy any text or symbols from Image {b}
- Do NOT overlay the two images
- The output must show only the content of Image {a}, in the handwriting style of Image {b}

For runs with textual ground truth this was added at the end of the prompt. “{formula}” is filled with the raw LaTeX representation of the generated formula.

Help:

The mathematical formula is: {formula}

D.4 Font Generation Prompt

The prompt used for the font generation described in Section 2.2.3. All sentences marked with “{}” reference either the character image or the style guide image and are only used, if their respective image was passed.

TASK: Generate ONE isolated handwritten character <PLAINTEXT CHARACTER>.

{CONTENT SOURCE: first image defines the target character <PLAINTEXT CHARACTER> only.}
 {STYLE SOURCE: second image defines handwriting style only.}

Important constraints:

- Output exactly one character: <PLAINTEXT CHARACTER>.
- No additional characters or symbols.
- No words, paragraphs, fragments, or copied text.
- {- Do NOT copy any visible content from the handwriting sample.}
- {- Use the handwriting sample only for stroke style and writing behavior.}
- Do NOT trace the original character mechanically.
- {- You MUST NOT copy the style guide, use it ONLY as reference for the handwriting style.}
- {- The size of the handwritten character HAS TO be the same as that of the target character in the first image.}

Desired appearance:

- natural human handwriting
- slightly hurried execution
- uneven but confident sharp strokes
- minor asymmetry and imperfections
- still easily readable

Final image: single centered handwritten character on a plain empty background.

D.5 Model Judge Prompt

The prompt used for the model judge described in Section 2.2.4.

You are given two images of mathematical content:

- Image A: the original formula
- Image B: an AI generated version of the first image

Determine whether Image B preserves the same mathematical content and meaning as Image A.

To help you with that, here is the formula from Image A in LaTeX format:

INSERT_FORMULA_HERE

Evaluation rules:

- Focus ONLY on whether the mathematical meaning/content matches.
- Ignore irrelevant visual artifacts such as:
 - extra lines
 - rendering noise
 - spacing differences
 - font/style differences
 - alignment issues
 - distorted symbols that do not change meaning
 - decorative or accidental marks
- Ignore any crossed-out content in Image B completely.
- If Image B contains crossed-out content and additional non-crossed-out content, only evaluate the non-crossed-out content.

- Do NOT judge whether the mathematics itself is correct.
- Treat mathematically equivalent notation as matching.
- Only return NO if the actual mathematical meaning/content changes.

Examples that should still count as a match:

- faint extra strokes
- OCR-like artifacts
- different spacing
- different bracket sizes
- handwritten vs typed appearance
- crossed-out incorrect intermediate steps
- messy but semantically identical equations

Examples that should count as NOT matching:

- changed exponent
- missing minus sign
- altered variable/index/number
- different denominator
- removed/additional term
- changed operator
- additional non-crossed-out mathematical content that changes meaning

E Realism Test Word Clouds

This appendix lists the words present in each semantic token cluster by test group.

E.1 Group 1

In group 1, six clusters were identified: Legibility, Realism, Size, Structure, Uncertainty and Uniformity. The tokens present in each cluster may be found in Figure 37 and Figure 38.



Figure 37: Realism test group 1 - clusters Uniformity, Structure and Legibility



Figure 38: Realism test group 1 - clusters Size, Uncertainty and Realism

E.2 Group 2

In group 2, four clusters were identified: Legibility, Realism, Structure and Uniformity. The tokens present in each cluster may be found in Figure 39 and Figure 40.

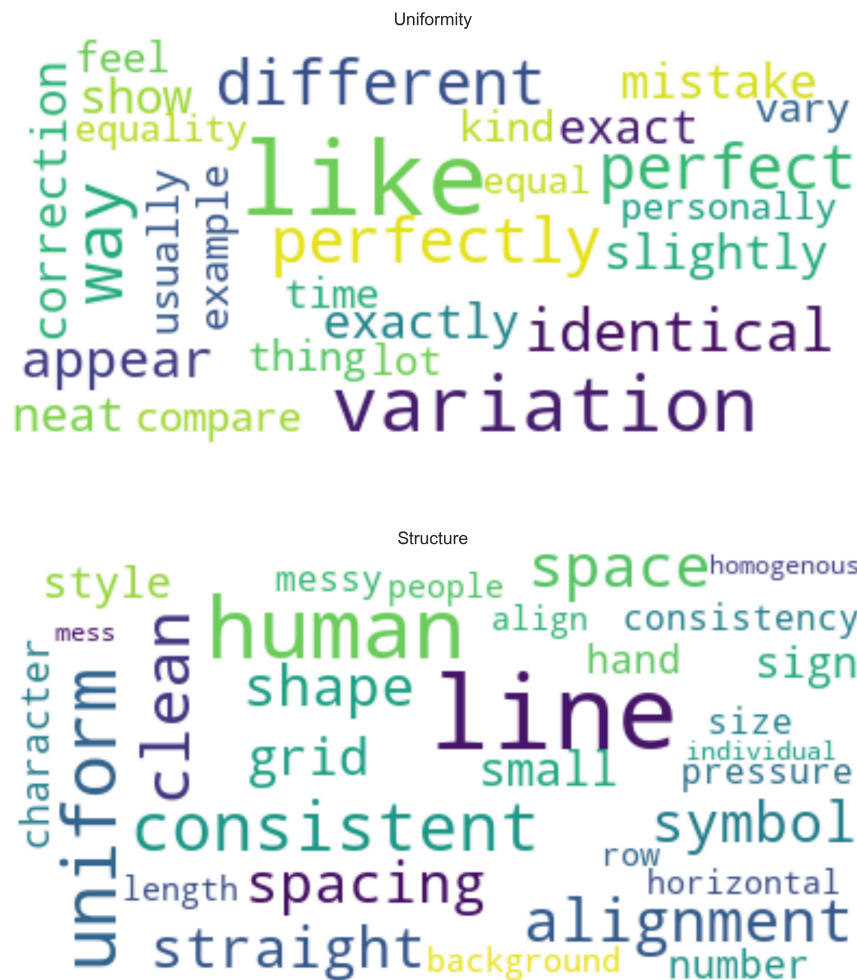


Figure 39: Realism test group 2 - clusters Uniformity and Structure

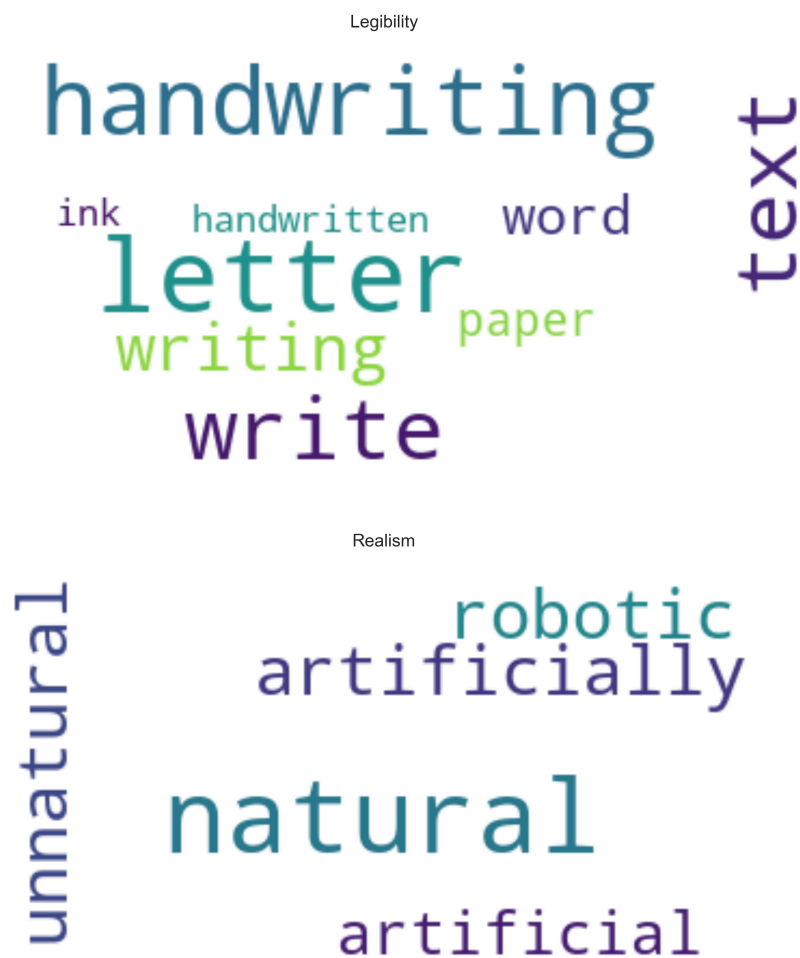


Figure 40: Realism test group 2 - clusters Legibility and Realism

F List of Aids

All aids used throughout this thesis are listed in Table 16, an additional AI Declaration may be found in Appendix F.1.

Task	Aids
Literature review	Google, Google Scholar, ChatGPT
Data analysis and Visualization	Python, Pandas, spaCy, sklearn, sentence_transformers, Matplotlib, Seaborn
Coding	Python incl. dependencies, PyCharm, VSCode, ChatGPT, Claude
Writing	Typst
Collaboration	Outlook, Teams, In-Person
DevOps	GitLab, Docker, Conda environments

Table 16: List of aids

F.1 AI Declaration

AI was used to:

- Help with implementation of experiment Python-scripts
- Brainstorm different realism experiment methods
- Find existing literature

The authors are responsible for the results.

G Code

The code is licensed under *MIT-License* and at the time of this report is hosted on the OST *GitLab* instance at:

<https://gitlab.ost.ch/ba2026-handwriting/code>

G.1 Requirements

To run the full generation pipeline, as well as the demo-client, the following requirements must be installed:

- Python-Packages from `requirements.txt`
- The `xelatex` CLI-tool
- The `pdftoppm` CLI-tool (part of `poppler-utils`)

To run the font creation program, the following are required:

- FontForge
- Potrace

G.2 Usage

For the generation pipeline, refer to the file `_previous/README.md`. Instructions for the demo-client may be found in `demo_client/README.md`. Finally, usage of the font creation program is described in `generate_font/README.md`.

G.3 Replicating Generative Approaches

To replicate the experiments for our generative approaches, some additional information is required. Requirements remain the same as previously stated.

G.3.1 Document Augmentation

For the document- and document-subsection-augmentation, the file `generative_experiments/systematic_experiments/image_edit.py` may be used to replicate the grid-search.

G.3.2 Font Generation

The files `generate_font/_glyph_generator.py` and `generate_font/_glyph_generator_openrouter.py` may be used to generate all needed glyphs, depending on which method of running/prompting models is used. Afterward manually inspecting some generated glyphs, the file `generate_font/_font_assembler.py` may be used to convert glyph images into an installable font-file. Importantly, the file `generate_font/config.py` is used, to specify paths and font names, see also Appendix G.2.

G.3.3 Model Judge

Lastly, the file `model_judge/model_judge.py` may be used to replicate the Model Judge setup. The directory `model_judge` also provides some correct (i.e. hallucination-free) samples and some incorrect (i.e. with hallucination) ones, as well as the ground truths for each sample.

G.4 Replicating Realism Benchmark

Once the user experiment has been carried out, analysing collected data can be done with the scripts found on the thesis documentation GitLab Repository [20] at `analysis/src/analysis/main.py` for the quantitative benchmark and `analysis/src/qualitative_analysis/main_g1.py`, as well as `analysis/src/qualitative_analysis/main_g2.py` for the qualitative analysis.

These scripts require the Python-Packages listed in `analysis/src/requirements.txt` from the same GitLab Repository.

H Full Scope

The section “Context” describes the overall project context. The actual BA requirements/BA scope are/is defined in “Minimal requirements” and “Optional requirements”.

H.1 Supervisors

Mitra Purandare (mitra.purandare@ost.ch)

Gerd Kortemeyer (gerd.kortemeyer@sl.ethz.ch)

H.2 Students

Sylvester Homberger (sylvester.homberger@ost.ch)

Nicola Dall’Acqua (nicola.dallacqua@ost.ch)

H.3 Context

The primary customers of this project are ETH’s Apertus team and ETH’s Ethel team. They want a synthetic dataset of realistic, exam-style documents with generated handwriting. There already is an existing pipeline that generates handwritten documents, producing results such as in Figure 41.

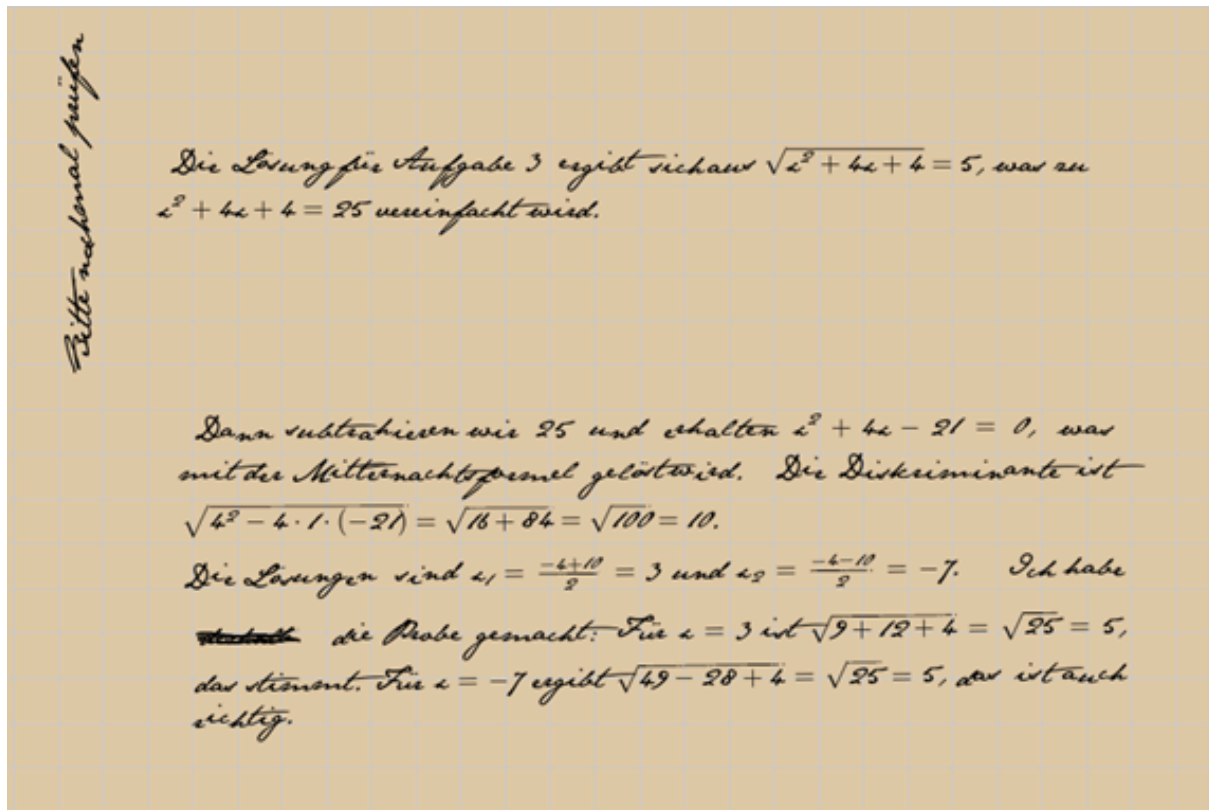


Figure 41: Existing pipeline - Sample 3

This pipeline may be used and extended. The pipeline works by generating ground truths in Latex-format using LLMs. These ground truths are then processed to create “handwritten” documents. The Apertus team specifically requires that the generated dataset meaningfully differs from what is already publicly available. Existing handwritten datasets fall into either of two categories:

- Snippets of math/individual math symbols, not embedded into a larger text/paragraph, without additional context
- Entire texts/paragraphs (i.e. a page of handwriting) but without any math symbols

Currently, there aren't any large-scale, public datasets which contain mathematical derivations embedded into larger handwritten text/paragraphs. Apertus requires that this BA combines these two categories to fill this gap. They have conducted a preliminary search and found the current outputs of the pipeline to already be quite distinctive from existing datasets (see example screenshot, contains math and text). However, they find these outputs to be too rigid/restrictive (i.e. placing to rigid, text lines too straight, characters always look the same, ...).

The Apertus team also suggested to extend the pipeline such that the outputs are even more "exam-style". This means the ground truths that are used, need to have realistic content. Furthermore, they proposed generating documents with typed/machine-written problem statements (standard computer fonts) and handwritten solutions (like is the case in real-world exams). If there is enough time, things such as multiple-choice questions with handwritten crosses may also be attempted. Additionally, they would like millions of samples in the generated dataset. Also, the findings of this BA should be made into a conference paper, which the Apertus team would submit to some conferences.

Aside from Apertus, the Ethel team/Gerd would like to have a quality measure for the generated handwriting. Several people will receive both generated handwriting and real, human handwriting and are tasked with classifying which is which. Exact test design beyond these basic requirements is part of the BA. Additionally, the Apertus team has proposed doing this same test with pretrained Vision Models (GPT-Vision, ...).

Some initial ideas that could be done (but aren't required):

- Maybe don't classify binary (real/generated) but rather on a scale of "how sure are you"
- The real documents and generated documents might have the same font
 - This means the handwriting of people creating the real documents may be turned into a font
 - This could be done with code that the Apertus team has developed at an earlier point
- The ground truths for the generated documents might contain the same content as the real handwriting (manually create ground truths rather than letting an LLM generate them)

Furthermore, a secondary approach at generating handwriting should be tested out. Specifically, existing Diffusion Models/GANs/... should be tested for their handwriting generation capabilities. If they produce better results than the current pipeline, they are to be used going forward. To that end, the Apertus team provides resources on the CSCS HPC which may be used to run these models.

H.4 Minimal Tasks

1. Literature review to prove that our dataset is different from existing ones, i.e.:
 - Contains math and larger text paragraphs/context
2. Test out new handwriting generation/augmentation approach
 - Can AI models be used to improve the quality of the generated data?
3. Implement Apertus's suggestions to make the data more "exam-style"
 - Focus on typed problem statements. Multiple-choice and ground-truth-verification optional if time permits
4. Quality measure, incl.:
 - Exact test design
 - Classification with human test subjects

Optional Tasks

1. If it makes sense timewise: Perform Turing Test/Quality measure using a pretrained Vision Model instead of human test subjects
2. If it makes sense timewise & quality-wise: make the pipeline production-ready to create millions of samples for Apertus and produce those

3. If it makes sense timewise: Write draft for the conference paper (without submission and not the final/finished version)

H.5 Deliverables

1. Pipeline that CAN generate millions of samples
 - These samples must contain long text paragraphs with mathematical calculations/derivations in context
 - Speed and high efficiency are not a constraint to be considered
2. Demo-Client where users can submit their own latex and have it turned into handwriting
3. Review of existing datasets to appropriately document how our dataset is different
4. Explore generative AI approach and answer the question: “Can AI models be used to improve the quality of the generated data?”
5. Add typed problem statements and handwritten solutions as suggested by Apertus
6. Turing test/Quality measure including the concrete test design
 - The approval for this was handled by Gerd
 - The ETH’s Decision Science Lab participants can be used for this

H.6 Grading

Both students will receive a joint grade, independent of individual work. The grade consists of the following parts weighed at their standard levels in accordance with the “Leitfaden BA SA”:

- Organisation und Durchführung: 10%
- Formale Qualität des Berichts: 20%
- Analyse, Entwurf und Auswertung: 20%
- Technische Umsetzung: 30%
- Bachelorprüfung: 20%

H.7 Roadmap

The roadmap for this project is illustrated in Figure 42. Importantly, orange boxes are optional and may not be done within the BA.

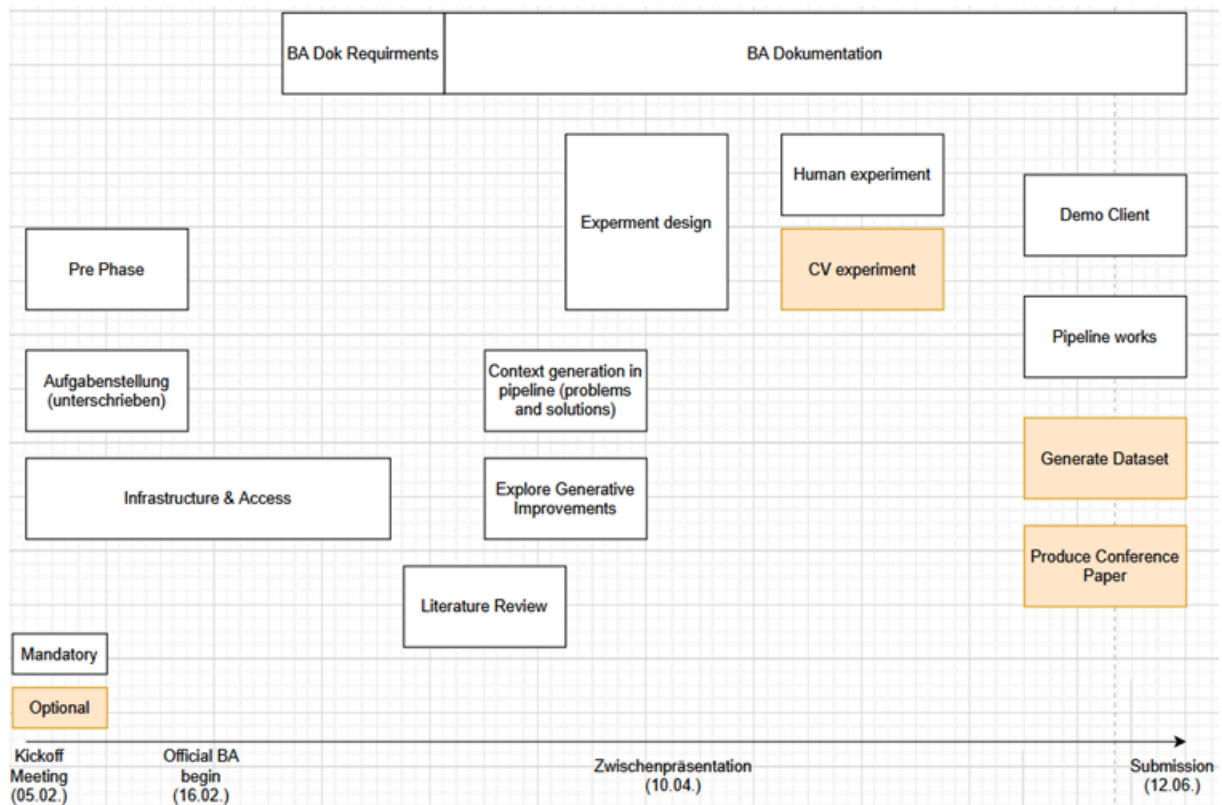


Figure 42: Thesis roadmap

H.8 Further Information

- All BA products (code/data/...) are open source