Handwriting Data Generation and Augmentation for Mathematical Content

Rosa Mayila, David Schulmeister, Amene Gafsi

École Polytechnique Fédérale de Lausanne (EPFL)

Supervised by:

Imanol Schlág

ETH AI Center, Zurich

A project conducted for ETH Zurich by students from EPFL.

***Abstract*—Building an OCR system is a complex task requiring valuable and meaningful data. This project proposes a method for generating handwritten math exercises using LaTeX and image processing techniques. By simulating handwritten mathematical expressions, the dataset supports the training and evaluation of OCR systems specialized in recognizing mathematical content. The approach focuses on creating realistic, human-like handwriting to ensure authenticity and utility for OCR applications.**

I. Introduction

In the context of Ethel, ETH's virtual teaching assistant, one of the critical challenges is processing handwritten student submissions to provide timely and accurate feedback. Traditional Optical Character Recognition (OCR) systems face significant limitations when applied to student work, particularly in the domains of mathematics and multilingual education. These limitations include difficulty handling handwritten mathematical notation, diverse handwriting styles, and content in multiple languages such as English, German, French, and Italian.

This project focuses on developing a robust dataset that can serve as additional training data for vision-language models designed to process handwritten submissions. By addressing the specific challenges posed by variability in handwriting, mathematical content, and multilingual requirements, the dataset aims to:

- Enhance the transcription accuracy of handwritten text and mathematical expressions.
- Support Ethel's capability to provide meaningful feedback on student submissions.
- Expand the applicability of OCR systems to multilingual and mathematical contexts.

The primary objectives of this project are directly aligned with enhancing Ethel's ability to process handwritten student submissions. First, the goal is to generate a robust dataset using advanced language models to produce a wide range of mathematical exercises. These exercises are designed to reflect the diversity and complexity of real student work, incorporating multilingual content as well as structural and semantic irregularities, such as intentional mistakes and formatting variations. Second, the focus is on augmenting the dataset by converting the generated LaTeX documents into high-quality PDF and PNG formats, followed by applying image processing techniques. These techniques, including noise addition, blurring, and other distortions, simulate the imperfections of scanned handwritten documents. This comprehensive approach ensures that the dataset effectively enhances the accuracy and robustness of vision-language models tailored for Ethel's educational applications.

II. Methodology

To support Ethel's goal of understanding handwritten student submissions, the dataset generation pipeline is designed to address the key requirements of OCR systems for educational contexts. These requirements include:

- **Handwriting Variability:** By using custom fonts and augmentation techniques, a wide range of handwriting styles is simulated to reflect the diversity of ETH's student body. This ensures that the dataset represents the variability in student submissions, from neat handwriting to irregular and imperfect forms.
- **Mathematical Complexity:** Handwritten mathematical content often includes a combination of textual explanations, complex equations, and symbols. The dataset captures these elements with realistic layouts and varying levels of complexity, enabling OCR models to effectively handle both textual and structured mathematical inputs.
- **Multilingual Support:** Recognizing ETH's multilingual environment, the dataset incorporates content in English, German, French, and Italian. This aspect is critical for training OCR models to process multilingual handwritten submissions, including scripts with diacritics, accents, and unique linguistic features.
- **Realism and Imperfections:** To mimic the conditions of scanned documents, the pipeline applies controlled noise, blur, and other distortions to the generated content. This ensures that the dataset reflects real-world imperfections, such as ink smudges, uneven alignments, or faded text.
- **Error Representation:** Students' handwritten submissions often include mistakes, corrections, and formatting inconsistencies. By introducing intentional errors and scribbles into the dataset, authentic student work is simulated and OCR models are prepared for these real-world challenges.

These components make sure that the dataset meets the requirements for training vision-language models capable of accurately processing handwritten submissions in diverse educational scenarios. By focusing on handwriting variability, mathematical complexity, multilingualism, realism, and error representation, the pipeline addresses the key challenges of developing robust OCR systems for Ethel's teaching assistant applications.

III. Implementation

The implementation of the handwriting synthesis project is divided into two major components: the synthetic dataset generation pipeline and the custom font generation subproject. Together, these components form a complete system capable of generating, processing, and refining handwritten mathematical exercises. Python is used for:

A. Synthetic Dataset Generation Pipeline

***Latex Generation**:* A large language model (LLM) is prompted to generate solutions for random math exercises in a language randomly selected from a predefined set specified

in the configuration file (`main.py`). The structure of the output is explicitly defined in the prompt: it must start with `\begin{document}` and end with `\end{document}`.

However, the model sometimes omits these delimiters or includes extraneous text outside them. To ensure consistency, the generated response undergoes a cleaning process with the following steps:

- Ensuring Required Commands: Verify that the response includes both the `\begin{document}` and `\end{document}` commands. If these are missing, they are added to encapsulate the content properly.
- Removing Extraneous Text: Any text appearing outside the `\begin{document}` and `\end{document}` commands is removed to maintain the integrity of the LaTeX structure.

Human Errors Simulation: To achieve a human-like "strike-through" effect for simulated errors, the pipeline introduces intentional mistakes into the LaTeX file, which are delimited for further processing. The task of inserting these mistakes is handled by the LLM, which is explicitly instructed in the prompt to include humanistic mistakes and encapsulate them using the custom `\strikeMistake{}` command.

During LaTeX compilation, the header is randomly augmented with one of several predefined `\strikeMistake` designs. These designs, created using the TikZ package, provide diverse strike-through effects, enhancing the visual variety and realism of the dataset.

Figure 1. Example of mistake.

Handwriting Simulation: To achieve a handwriting effect in the LaTeX file, handwriting fonts are used as the primary method. Fonts such as JaneAusten and Augie produce good results; however, these existing fonts lack support for mathematical symbols, which is crucial for our use case. To address this, we implemented a custom font generation pipeline III-B that creates several personal fonts, including "ML4Science" (which is supported by the model). This font includes mathematical symbols, such as $\sqrt{}$ or $\int$, necessary for the applications.

However, a challenge arises when adjusting the size of certain math symbols, which needs to be dynamic depending on the content. For example, the fraction symbol must be adjusted in both height and length based on the size of its content. This issue is handled by explicitly mapping these symbols to corresponding ones in the custom font using the `\renewcommand{}` command. A transformation is then applied to the mathematical symbols according to their content size, ensuring proper rendering in the final output.

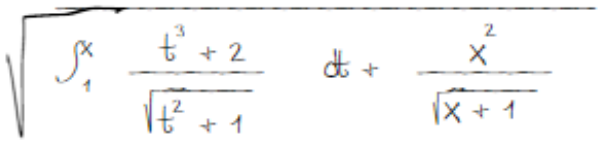

Figure 2. Example of equation generated using the ML4Science font.

To create more irregular sentences that better mimic handwriting, a `\preprocessText` command is used to encapsulate all raw text in the LaTeX file. The definition of this command is added to the LaTeX header. The command works by separating the text into individual words, with each word being shifted along the y-axis and rotated. Both the amount of shift and the degree of rotation are randomly selected from predefined ranges. This randomness in the positioning and rotation of words introduces variability, making the text appear more natural and handwritten.

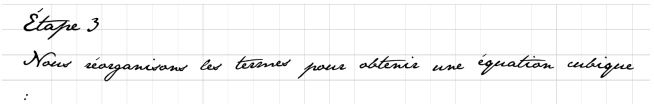

Figure 3. Example of irregular sentence using JaneAusten font.

Image generation: To generate PNG images from the LaTeX files, a custom header is added to the content generated by the LLM. This header includes all the necessary components discussed earlier, such as font specifications and strike-through designs for the mistakes. The LaTeX files use a standalone class, rather than the typical article class, as this allows the page dimensions to be dynamically adjusted based on the content, eliminating the need for cropping in the generated PNG images.

Once the headers are added, the LaTeX files are compiled into PDFs using the XeLaTeX engine. For each LaTeX document, all possible combinations of parameters specified in the configuration file (main.py) are generated, including different text colors, page colors, and fonts. Additionally, for each file, two versions are generated: one with a clear background and another with a grid-like background that resembles notebook pages.

After the PDFs are generated, they are converted into high-resolution PNG images using the `pdftoppm` utility. To further augment the data, for each generated PNG image, three additional versions are created: one with noise 4, one with blur 10, and one that is both noisy and blurred. This is done using the `Pillow` library. By simulating scanning artifacts and other visual distortions, a set of final images is produced that closely resemble genuine handwritten work.

These images can serve as valuable training data for OCR models focused on handwriting recognition. [1]

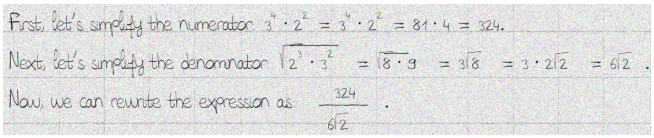

Figure 4. Example of noisy text generated.

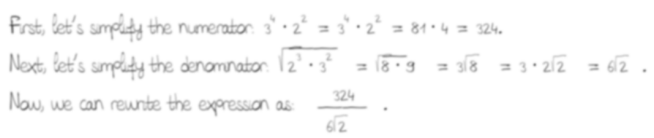

Figure 5. Example of blurred text generated.

B. Custom Font Generation Sub-project

The custom font generation sub-project is a critical component of our pipeline, supporting Ethel's ability to process handwritten student submissions by providing diverse and realistic handwriting styles. This sub-project begins with the creation of character templates using the `Pillow` library to produce grids that define precise placement for each glyph. These templates are distributed in PNG and PDF formats, allowing contributors to fill them with handwritten characters.

Once these templates are filled, they are processed to extract individual glyphs. Borders and backgrounds are removed dynamically, and the glyphs are up-scaled and binarized. The `Potrace` tool is then used to convert the bitmap images into scalable vector graphics (SVG) format. Each extracted glyph is mapped to a corresponding Unicode slot based on a predefined mapping in a CSV file.

All these glyphs are integrated into an OpenType font using `FontForge` [2]. Special care is taken to normalize the glyphs, ensuring consistent visual alignment, proper handling of ascenders and descenders, and accurate placement of mathematical symbols and accents. The resulting fonts can be seamlessly used within the LaTeX compilation process, thereby introducing a genuine handwritten appearance to the final dataset.

C. System Tools and Libraries

The system relies on several tools and libraries. FontForge is used to construct OpenType fonts from extracted glyphs, while Potrace converts bitmap glyphs into scalable vectors. The Pillow library supports image processing tasks, including scaling, border removal, and noise addition, and the `pdf2image` utility facilitates conversion from PDF to PNG. Finally, XeLaTeX serves as the LaTeX engine that compiles documents into PDFs. Together, these tools form a robust and flexible tool-chain for generating, processing, and refining realistic handwritten datasets.

D. Challenges and Solutions

During implementation, several difficulties were encountered that required careful attention. Issues included ensuring that syntactic and semantic mistakes introduced by the language model were correctly represented, achieving consistent visual scaling and positioning for glyphs during font creation, and producing realistic noise and blur artifacts that convincingly emulated scanned handwriting. These challenges were addressed by iteratively refining the prompts for the language model, improving the glyph extraction and normalization steps, and experimenting with a range of image processing parameters until the resulting dataset closely matched the appearance of authentic handwritten work.

IV. Results

The results of the methodology yield a dataset of synthetic handwritten math exercises specifically tailored to address Ethel's challenges in processing student submissions. By leveraging advanced language models, LaTeX formatting, custom fonts, and controlled image degradation techniques, a dataset has been created that closely mimics authentic scanned documents. This dataset is designed to enhance OCR systems' performance, enabling Ethel to more accurately and robustly recognize multilingual handwritten content, complex mathematical expressions, and step-by-step solutions. Figure 4 shows the noisy version of generated exercise solution and Figure 6 shows the results of the custom font metrics right after its generation. Additional results images can be found in the Appendix.

Figure 6. Custom font right after its creation

V. Future Work

Future work will focus on further aligning the dataset and pipeline with Ethel's requirements by enhancing its realism, flexibility, and applicability:

A. Layout and Content Variability

Future work should focus on introducing diverse layouts to create more realistic and varied document structures. This includes irregular placements of equations and text, which better simulate the layout variability typically found in real-world handwritten submissions. Additional elements such as handwritten annotations, diagrams, and common page imperfections—like stains or incomplete scans—will further enhance the authenticity of the dataset. Furthermore, Handwriting recognition from a blackboard should be considered, as this could enable automatic transcription of blackboard contents from lectures.

Moreover, multiple handwritten solutions for the same problem should be generated to provide a more comprehensive dataset. These solutions will include correct answers representing ideal submissions, incorrect answers containing intentional errors to mimic real student mistakes, and partially corrected submissions featuring scribbled annotations and iterative fixes. These enhancements will ensure that the dataset reflects the diversity and complexity of real student work, making it more applicable for training and evaluating OCR systems.

B. Improved Formula Rendering

Future work should involve the development of a custom handwriting synthesizer to improve the rendering of mathematical formulas. This will provide enhanced control over formula aesthetics by addressing stroke imperfections, irregular spacing, and alignment variability. Additionally, the synthesizer could simulate handwritten mathematical symbols and equations with greater diversity and accuracy, ensuring a more natural and realistic representation of handwritten content for training OCR systems.

C. Handwriting Style Diversity

Expanding the diversity of handwriting styles could be another key focus for future work. This would involve incorporating glyph samples from varied cultural and linguistic backgrounds to ensure the dataset represents a wide spectrum of writing styles. Additionally, personal writing traits such as slant, stroke pressure, and unique patterns—like cursive and irregular scripts should be simulated to enhance the realism and authenticity of the generated content. These improvements would ensure the dataset captures the full range of variability found in real handwritten documents.

VI. Ethical Risks

The development and use of synthetic handwritten datasets raise ethical concerns that require careful mitigation. To prevent misuse for fraudulent purposes, access should be restricted, and watermarks or metadata embedded in synthetic content. Privacy concerns can be addressed by anonymizing handwriting samples, obtaining explicit consent, and mixing fonts to avoid resembling any individual's handwriting. To safeguard authenticity, synthetic handwriting must be clearly distinguished from real handwriting and not used in fields like forensics. Finally, to avoid unintended OCR biases, the dataset should include diverse handwriting styles and maintain human oversight in grading systems.

By addressing these risks, the project can responsibly support educational technology while minimizing misuse.

VII. Summary

In summary, a pipeline has been developed for synthesizing realistic handwritten math exercises using language models, LaTeX formatting, custom fonts, and image augmentation techniques. The resulting dataset captures the complexity and variability of genuine student solutions and serves as a valuable resource for training handwriting recognition models and automated grading systems. Through continued refinement and expansion, this approach holds promise for significantly advancing the capabilities of educational technology tools.

Acknowledgments

We would like to thank Imanol Schlag for his supervision and support throughout the project.

References

[1] F. Ullah, K. Muhammad, and M. Sajid, "A comprehensive review of optical character recognition systems for handwritten text," *Journal of Machine Learning and Applications*, vol. 15, pp. 45–67, 2022.

[2] F. Developers, *FontForge API Reference*, FontForge, 2024, available at https://fontforge.org/docs/scripting/python.html.

Appendix

Here, additional results images are provided.

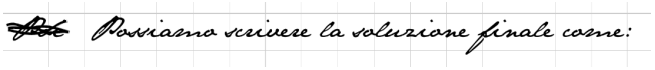

Figure 7. Example of humanoid strike-through design.

Demonstration of Strikes

This is a demonstration of different strike-through styles in a single paragraph. Sometimes, simple strikes are enough to cross out certain words or phrases, but other times you may want a wiggly line for a more dynamic effect. For a bold, artistic touch, the brush-like style adds a creative feel. Finally, when everything needs to connect seamlessly, the connected line strike provides a polished and professional look. We do not have a strike command yet, but we can create one. 52562 46152 + 4 = 0

Figure 8. Example of irregular text generated.

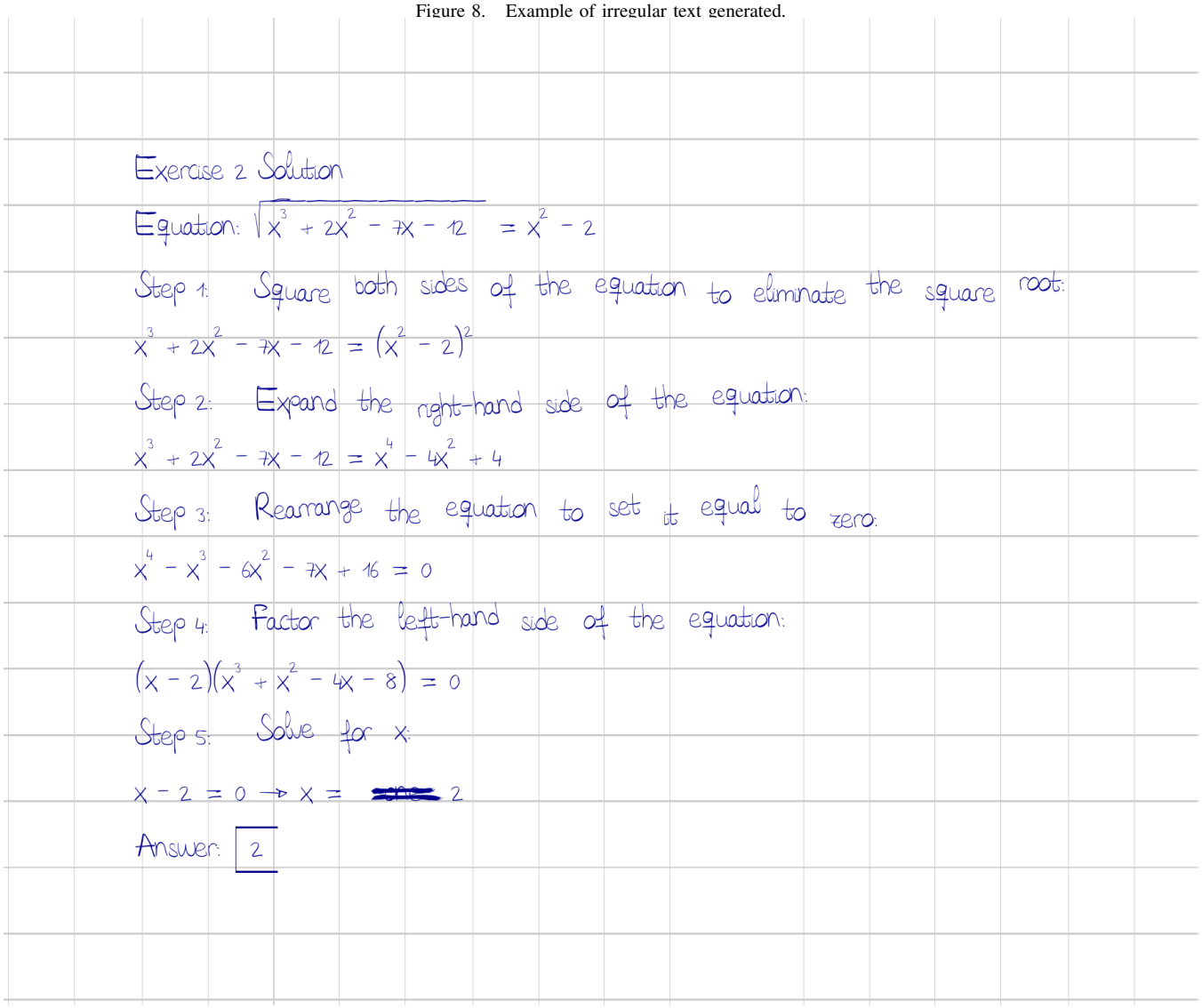

Figure 9. Dark blue text generated with the ML4Science font, displayed on a grid page.

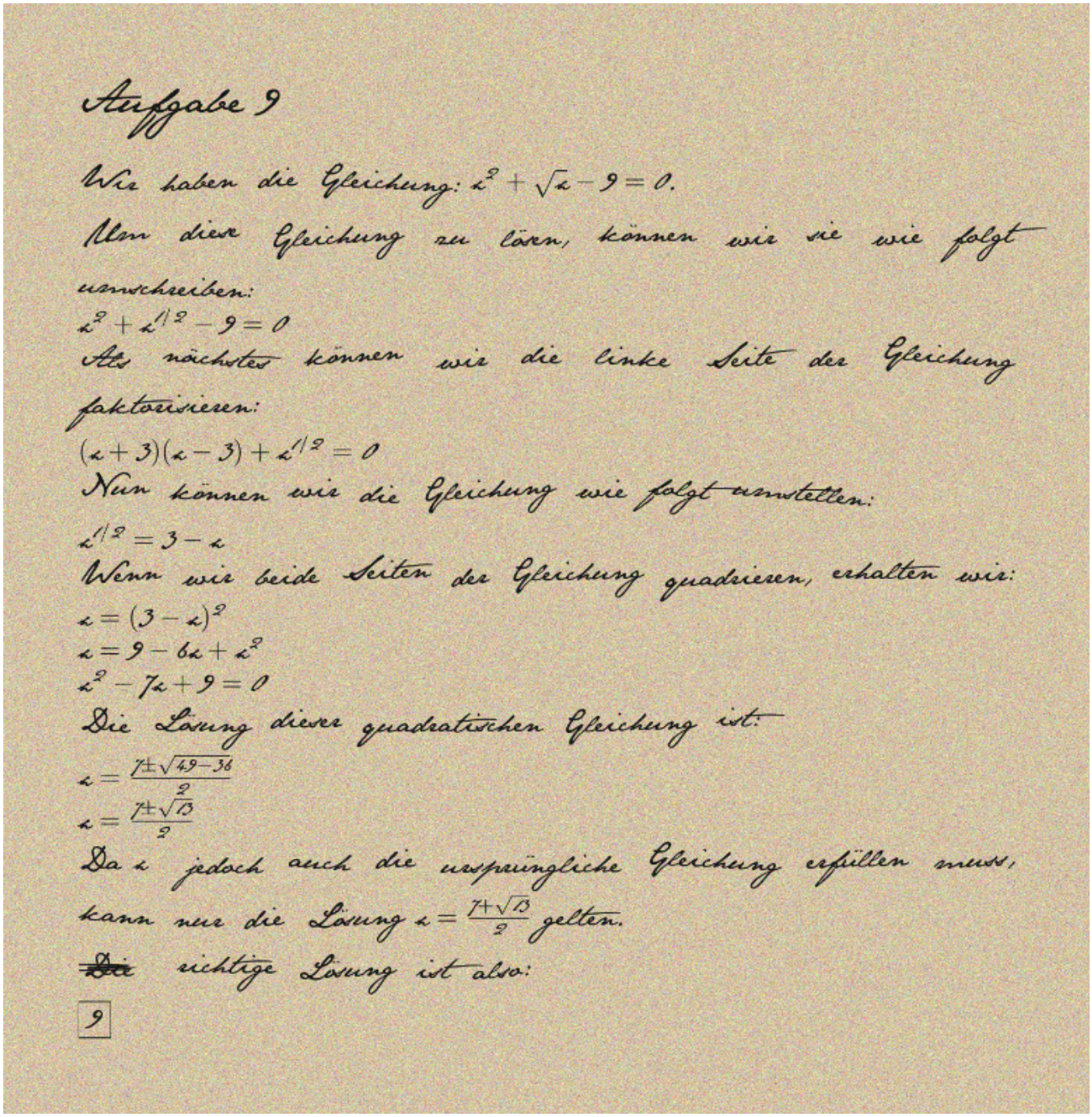

Figure 10. Noisy German text generated using JaneAusten font.