Educational Image Generation

Group 2, CSE 144, UCSC

Lucas Ellenberger
Computer Science, B.S.
University of California, Santa Cruz
Santa Cruz, CA, USA
lmellenb@ucsc.edu

Thierry Nguyen
Computer Science, B.S.
University of California, Santa Cruz
Santa Cruz, CA, USA
tnguy564@ucsc.edu

Nia Balaji
Computer Engineering, B.S.
University of California, Santa Cruz
Santa Cruz, CA, USA
nbalaji1@ucsc.edu

Christopher Casarez
Computer Science, B.S.
University of California, Santa Cruz
Santa Cruz, CA, USA
chdcasar@ucsc.edu

ABSTRACT

The project aims to address the labor-intensive and time-consuming task of manually searching for relevant images that complement and explain large amounts of textual data. By automatically generating visual representations for textual content, we aim to enhance comprehension and reduce the time required for learning complex material.

We utilized an open-source Hugging Face model for text summarization, which processes user input to generate text summaries. These summaries are used as input to our prompt-tuned Llama-3 model, producing prompts for image generation. The prompts are fed into a fine-tuned stable diffusion model, enhanced using LoRA techniques. Given the unique nature of our task, we created a custom dataset, leveraging a BART model for prompt creation and a web scraper for data collection.

Our project aims to assist users by automating the process of identifying visual aids, facilitating a quicker and more accessible understanding of complex topics.

Text summarization is well-documented and straightforward, but creating prompts is more challenging than it appears. Our progress was limited by a lack of research and quality datasets. Increased research and better access to resources would allow us to fine-tune LLMs more effectively for improved outcomes.

While our image model scores more consistently than the baseline model based on user evaluation metrics, the overall performance remains suboptimal. Creating explanatory images for complex topics demands high accuracy and precise labeling, which our current model struggles to achieve. Despite extensive fine-tuning, the images produced are not yet suitable for practical use. We recommend developing a larger, well-documented dataset and extending training periods to improve image quality within budgetary constraints.

1 Introduction

1.1 Background

When teaching or learning difficult concepts, visual aids can significantly enhance understanding. However, finding appropriate images is often tedious and time consuming. Teachers may spend excessive time searching for images to supplement their lectures, and students might struggle to find visual aids to clarify complex textbook chapters. This effort of finding images detracts from the actual process of teaching and learning.

1.2 Problem Definition

Large amounts of text are challenging to comprehend and even more difficult to find corresponding visual aids. By generating images from lengthy content, we can visualize key points, thereby aiding the teaching and learning process. While effective text summarization models exist, to the best of our knowledge, there are no models that proficiently create images from extensive text.

1.3 Objectives

This project aims to fine-tune existing large language models (LLMs) and stable diffusion models to generate multiple explanatory images from large text inputs. By adapting and enhancing these models for our specific task,

we seek to facilitate educational image generation within a short time frame and limited budget.

1.4 Significance

If successful, our project will provide a flexible tool that generates relevant visuals for any large text, which will substantially reduce the time spent searching for educational images. This capability can support various fields of teaching and learning by offering supplemental images that may not be readily available online. Even if the project does not achieve the intended outcome, it will provide valuable insights into the challenges of this complex task, contributing to the research community.

2 Methodology - BART LLM

2.1 Approach Overview

The primary objective of this study is to process extensive textual data, such as multiple pages from a textbook, eg: a chapter from a biology textbook. Utilizing the Facebook BART model, the aim is to achieve a comprehensive understanding of the complete content, discerning the fundamental concepts and themes. The methodology involves segmenting the text based on key ideas and concepts, subsequently generating concise summaries. Additionally, keywords are extracted from the text to facilitate further analysis. With this information, the intention is to prompt the model to generate contextually relevant and semantically meaningful text-to-image model prompts.

2.2 Algorithm and Models

2.2.1 Bart Model 1. This is a Hugging Face model that was perfect for this purpose as “BART is particularly effective when fine-tuned for text generation (e.g. summarization, translation) but also works well for comprehension tasks (e.g. text classification, question answering).”. This model has been pre-trained by corrupting the textual data using a noise function, then learning a model to reconstruct the text. This model performs well on summarizing textual data.

BART is a model that utilizes a transformer architecture, with a bi-directional encoder, which is similar to BERT, and an autoregressive decoder similar to GPT. I choose to use this model because its capability to generalize text seemed to prove superior to other similar pretrained models. The model was able to understand various texts with relatively similar comprehension levels. We also initially chose to use the DallE Mini model on hugging face which was pre-trained using this model. But we ultimately chose to use the Stable diffusion model for image generation to improve the quality of the images created. We were able to successfully use this model to create prompts to generate our own custom dataset, for testing and validation purposes.

2.2.2 TF-IDF vectorization algorithm. Term Frequency - Inverse Document Frequency (TF-IDF) is a statistical method used in natural language processing, it can help to determine which terms are important within a textual document relative to a collection of documents. This is ideal for the purpose, as we want to be able to identify key ideas from multiple pages of a textbook chapter. This algorithm is able to differentiate between common and rare terms. Although this algorithm does not take into account the semantic meaning of words, using this algorithm in addition to the BART model proved effective.

After obtaining the textual data, the model segmented the given data by relevant topics and created summaries for each segment. We first generate summaries for each statement, and then apply the TF-IDF algorithm to these segment summaries. Next, utilizing the vectors generated by TD-IDF, we employ the BART model. Notably, we fine-tuned this model using the midjourney user prompt dataset on Kaggle. This process enables us to create text prompts efficiently.

2.3 Implementation Details

2.3.1 Tools and Libraries

- Google Colaboratory
- Python
- PyTorch
- Diffusers
- Transformers
- PIL
- TfidfVectorizer
- Hugginface_hub: Used to gain access to the Facebook BART CNN model

2.4 Workflow

2.4.1 Research. Did extensive research on different text summarization models to find the optimal choice.

Initially, we started experimenting with using Google’s Pegasus xsum model [1]. However, we noticed that the summaries generated were too simple for this purpose, and looked for other models to test.

An article by Meta AI shows that Facebook BART performs significantly better than the other given models on the same text, for summarization purposes [2]. This evidence solidified the decision to adopt the Facebook BART model for experimentation.

2.4.2 Advantages. A notable advantage of the Facebook BART model — it demonstrated efficiency even when executed on a CPU, thus validating its selection as a lightweight solution. Leveraging this model, the large text was successfully segmented based on key ideas, facilitating a structured approach to summarization.

2.4.3 Process. Following the segmentation of the text into discernible parts based on key ideas, each segment was saved to use in the next steps. To extract essential keywords encapsulating the core themes of each segment, a multi-step approach was employed.

- Initially, the text within each segment underwent natural language processing techniques, including tokenization and part-of-speech tagging, to identify significant terms and phrases.
- Then, a weighting mechanism,TF-IDF (Term Frequency-Inverse Document Frequency), was applied to assign importance scores to these terms based on their frequency of occurrence within the segment and across the entire corpus.
- Advanced techniques, such as word embedding models like Word2Vec, were used to represent these keywords as dense vectors in a high-dimensional semantic space. This transformation facilitated a deeper understanding of the semantic relationships.

2.4.4 Conclusion. The final step involved the synthesis of these extracted keywords and the generated summaries to craft contextually relevant and semantically coherent prompts for each segment. By utilizing both the semantic meaning of the keyword vectors and the summaries, the prompts were relevant and usable for the next process.

3 Methodology - Llama-3 LLM

3.1 Approach Overview

To address the task of generating image prompts from large educational texts, we employed a multi-model approach that would take a large amount of text (mainly educational) as input.. This could be research papers, textbook chapters, etc. We then feed this input into a summarization model. This is fed into another model to create image prompts. This approach was chosen because it was not feasible to have one model that outputs image prompts directly from the input. A lack of datasets and computational resources would have made the latter approach much more difficult. Instead, a layered approach was taken: with 2 models for the LLMs.

3.2 Algorithm and Models

3.2.1 Llama-3 Model 1. A Hugging Face model that excelled at text summarization. It performed well on “lengthy documents, news articles, and textual content,” perfect for our case.

At first, the objective was to train a Llama-3 model from scratch for text summarization, but a consistent failure due to lack of resources on Google Colab and time constraints made us scrap this idea. There is currently a lack of research into producing high quality image prompts and prompt validation would be computationally expensive. Resources were already scarce and focused on training the Stable Diffusion models at hand[3]

Tweaking the max length of the output was also necessary, as we didn’t want too large of a summary. In some cases, too large of a max length caused it to output a summary just as long as the input itself!

3.2.2 Llama-3 Model 2. A Llama-3 model with 8 billion parameters, pulled from Hugging Face. Good for any task and relatively light, compared to the 70B model.
The prompt creator was a pre-trained Llama-3-8B-Instruct model. This was decided as there were not many feasible options to create image prompts directly from a summary. Using a general model that was decent at any task was the best option here.

To increase its capabilities even further, one-shot prompt tuning was utilized for the model to provide the exact formatting and type of output desired.

The model was asked to create 3 prompts for image generation that were simple and educational. It was then given an example of a summary with outputs similar to the prompts that the Stable Diffusion model was trained on.

It was able to successfully create 3 image prompts, which were parsed to be later used.

3.2.3 Algorithm 1. We attempted to use Model 1 to summarize a large amount of text and capture the key features. Then, the summary would be fed into Model 2 (the prompt creator), which was prompt-tuned for image prompt creation. Some input (a large amount of text) is taken in a python script, which is fed into Model 1. Model 1 outputs a small summary, which is then fed into Model 2 to create image prompts.

3.3 Implementation Details

3.3.1 Tools and Libraries

- Google Colaboratory
- Python
- PyTorch
- transformers (Hugging Face): Used to import pre-trained models
- Hugginface_hub: Used to gain access to the Llama-3 model

3.4 Workflow

3.4.1 Researched prompt creation and fine-tuning

1) Experiment with the open-source model provided. 2) Promptist used user prompts from an image generation website [4]. 3) The task at hand is not designed for those types of prompts, meaning data scraping those user prompts was not viable. 4) Initially, we attempted to train a Llama-3 model for text summarization. 5) Then, we realized that open-source models on Hugging face would perform much better than anything we could create in a small amount of time . It was not necessary to recreate the wheel. 6) Researched various models for summarization [5] [6] [7] [8]. 7) The chosen model has the following evaluation statistics [5].

- More lightweight (less parameters)
- Higher user count
- High training stats
- Evaluation Loss: 0.012345678901234567
- Evaluation Rouge Score: 0.95 (F1)
- Evaluation Runtime: 2.3456
- Evaluation Samples per Second: 1234.56
- Evaluation Steps per Second: 45.678

8) Modified the summarization model output length (It was often too long given very large amounts of text). 9) Added a Llama-3 model and was prompt-tuned until sufficient for image prompt creation.

4 Methodology - Stable Diffusion

4.1 Approach Overview

Due to constraints on time and computational resources, we needed to find techniques that would provide the most impact on the output of the model with minimal computational work. We opted to use a pre-trained stable diffusion model to significantly reduce the training time. However, we couldn't fine-tune the entire pre-trained model due to resource limitations. Therefore, we implemented Parameter Efficient Fine Tuning (PEFT) methods, specifically Low-Rank Adaptation (LoRA), to reduce the number of trainable parameters dramatically. This approach, while limiting some flexibility and expressive power, was necessary given our constraints.

LoRA, originally developed for fine-tuning Large Language Models (LLMs) [9], has been adapted for stable diffusion, notably by Simo Ryu [11]. Their GitHub project [10] provides examples and insightful discussions. Using Hugging Face's PEFT library [11], we incorporated LoRA techniques, which freezes the original model weights and injects a rank decomposition matrix into the cross-attention layers of the UNet. This greatly reduces the number of learnable parameters—down to 1/1000 of the original in some cases—making the fine-tuned model easier to manage and deploy.

All models were based on the CompVis/stable-diffusion-v1-4 model from Hugging Face, chosen for its open-source availability and compatibility with Google Colab's Tesla T4 GPU. Training and testing were conducted on Google Colab with a Python 3 Google Compute Engine backend (GPU), equipped with a T4 GPU (12.7 GB system RAM, 15 GB GPU RAM, and 201 GB Disk).

Every model we trained uses the same python training program [12] to learn the LoRA weights. The different models augmented the training dataset and hyperparameters of the training script [13] to produce different models. A list of requirements for these scripts and my invocation of one of them can be found in the following program [14] on our github.

4.2 Dataset Creation

4.2.1 Web scraping Script Creation. The implementation was done by utilizing the given Google Cloud credits provided by Professor Yi Zhang. I created a Google API key and a Google Custom Search Engine Dataset ID. Using the Google Cloud Credits we were given 10,000 queries to the Google API.

Initial stages of script development involved the creation of an Excel workbook to accommodate the incorporation of images into the dataset, recognizing the inherent limitations of CSV files in this regard. Leveraging Python libraries such as Pandas and Openpyxl, images were seamlessly integrated into the workbook. Furthermore, image manipulation techniques including resizing and cropping were applied to ensure conformity with predefined specifications. However, subsequent considerations for dataset publication on Hugging Face platform necessitated the conversion of images to .png format, prompting a transition from Excel workbooks to CSV files.

In an effort to improve functionality, I downloaded the queried images to a file named 'train', with the corresponding file names appended to the custom dataset. Initially pursued strategies utilizing BeautifulSoup4 and Scrapy proved challenging due to recurrent errors in image retrieval.

The final approach was created using ChromeDriver Selenium2Driver as these are more powerful web processing tools. Robust error handling mechanisms were implemented to validate image downloads and format compliance, which lowered the need for manual post-processing of the data. This transition enabled the deployment of enhanced web scraping techniques, resulting

in the creation of a dataset without prior impediments and inconsistencies.

4.3 Algorithms and Models

4.3.1 Model 1 [15]

- Dataset [16]: Hand-curated set of 20 images and labels.
- Training: 10 epochs with a checkpoint at 5 epochs.
- Purpose: Verify the results of the training script on a small dataset.

4.3.2 Model 2 [17]

- Dataset [16]: Same hand-curated set of 20 images.
- Training: 100 epochs with checkpoints every 25 epochs.
- Purpose: Extended verification with more training steps.

4.3.3 Model 3 [18]

- Dataset [19]: Caltech-UCSD's CUB-200 dataset (approx. 6,000 bird images).
- Training: 15,000 epochs with checkpoints every 500 epochs.
- Key Differences: Included random horizontal flips to reduce overfitting.
- Purpose: Validate methodology on a larger, well-labeled dataset.

4.3.4 Model 4 [20]

- Dataset [21]: Web-scraped dataset of approximately 250 images.
- Training: 1,000 epochs with checkpoints every 100 epochs.
- Purpose: Validate on a larger, custom dataset while mitigating overfitting risks.

4.3.5 Model 5 [22]

- Dataset [16]: Same hand-curated set of 20 images.
- Training: 1,000 epochs with checkpoints every 100 epochs.
- Purpose: Final validation with a well-understood small dataset.

4.4 Implementation Details

All of our models [15] [17] [18] [20] [22], used the following hyperparameters, unless stated otherwise. We list our rationale for choosing the particular hyperparameter, but there are other hyperparameters that could have performed better.

4.4.1 Hyperparameters

3.4.1.1 Precision. Mixed precision fp16 to reduce training time and memory usage, allowing for larger batches of data to be loaded into GPU RAM.

4.4.1.2 Image Processing. Standardized resolution to 512x512 and applied center cropping for normalization. Note that we did not randomly flip our images horizontally, which is standard practice to reduce overfitting. Our dataset contains diagrams and English text which is orientation specific, so we could not include random flipping.

4.4.1.3 Data Loading. Used 8 workers to speed up the data pipeline.

4.4.1.4 Training Duration and Learning Rate. Varied by model, with larger models trained for up to 15,000 epochs. Common learning rate of 1e-04, suitable for LoRA fine-tuning.

4.4.1.5 Gradient Clipping. Applied at a norm of 1 to prevent gradient explosion.

4.4.1.6 Learning Rate Scheduling. We used a cosine learning rate scheduler with no warm-up steps. This scheduler is suitable for fine-tuning due to its improved convergence near the end of training. Since the model is pre-trained, we do not need gradual adaptation to the training process.

4.4.2 Checkpointing and Validation. Checkpoints are evenly distributed across training steps. For large-scale training (15,000 steps), checkpoints every 500 steps provided valuable insights into model behavior. Validation prompts are used to monitor generative quality, but are omitted for small datasets to reduce training time.

4.4.3 Reproducibility. We included a seed to ensure reproducible results, enhancing reliability and accuracy in an academic setting. All of our datasets are available and publicly usable on Hugging Face. All of our models are available for serverless inference to verify results. For detailed scripts and parameter lists, refer to our GitHub repository [23].

4.4.4 Tools and Libraries. To implement and fine-tune our stable diffusion models, we used the below tools and libraries.

4.4.4.1 Data Handling and Numerical Operations

- **Numpy**: A Python package, supporting large, multi-dimensional arrays and matrices.
- **Datasets**: A library by Hugging Face for loading and processing datasets.
- **Datasets.load_dataset**: A function to load datasets from Hugging Face's datasets library.

4.4.4.2 Machine Learning and Deep Learning

- **Torch**: PyTorch, the recommended package for handling tensors and neural networks in Python due to its ubiquity in the community and strong GPU acceleration.
- **Transformers**: Hugging Face's library for state-of-the-art NLP models, providing pre-trained models and tools to work with them.
- **Accelerate**: A library to easily handle the process of running PyTorch models on any distributed setup and speed up computational processes.

4.4.4.3 Model Configuration and Utilities

- **Huggingface_hub**: Tools to interact with the Hugging Face Hub, including creating repositories and uploading models.
- **Peft**: Parameter Efficient Fine Tuning library, which includes the LoRA configuration tools.
- **Torchvision.transforms**: A module that provides common image transformations.

4.4.4.4 Diffusion Models

- **Diffusers**: A library from Hugging Face specifically designed for diffusion models, providing a wide range of tools and utilities.
- **AutoencoderKL**: A class for the KL-regularized autoencoder.
- **DDPMScheduler**: A scheduler for Denoising Diffusion Probabilistic Models.
- **DiffusionPipeline**, **StableDiffusionPipeline**, **UNet2DConditionModel**: Various pipelines and models for diffusion-based tasks.
- **Diffusers.optimization.get_scheduler**: Utility for obtaining learning rate schedulers.
- **Diffusers.training_utils**: Includes functions for casting training parameters and computing signal-to-noise ratio.
- **Diffusers.utils**: Utility functions for version checking, model card management, and more.

By using these tools and libraries, we trained our stable diffusion models efficiently, despite limited computational resources. Each library assisted in different aspects of our project, from data handling and preprocessing to model configuration and optimization, ensuring a smooth and effective workflow. To see the full training process, view this file on our GitHub.

4.5 Workflow

Once a dataset is created and uploaded to Hugging Face, we simply modify the hyperparameters via the training script [13]. The script invokes the training program [12] with the specified dataset and hyperparameters. Training can be monitored while the training program runs on the Google Colab instance. Upon completion, the resulting LoRA model and all checkpoints are automatically uploaded to Hugging Face.

5 Results - Stable Diffusion

5.1 Results of Model 3 [18]

To summarize the qualitative evaluation of our third model, we present the following prompt and associated images. These images were generated by the baseline model and the third model at different checkpoints of the training cycle. We chose this example because it demonstrates the overall trends clearly, but other prompts will likely show similar results. Since all of these models are publicly available via Hugging Face's serverless API, everyone can easily validate these findings.

5.1.1 Prompt. A photo of a chestnut sided warbler.

Figure 1: **Baseline Image**

Figure 2: **Early Training**

Figure 3: **Mid Training**

Figure 4: **Late Training**

Our third model underwent the longest training period. By manually evaluating the images, several trends emerged:

- **Early Training (500-1,000 epochs)**: Images were very similar to the baseline model, as seen in Figure 1 and Figure 2.
- **Mid Training (10,000 epochs)**: Image quality declined significantly, suggesting the model had unlearned some of the original model's capabilities, as seen in Figure 3.
- **Late Training (15,000 epochs)**: The model began to converge to a better fit, producing images that closely mimicked the training dataset. However, while the final model produced birds that resembled the prompts, the pixel quality deteriorated compared to the original model, as seen in Figure 4.

Despite some prompts outperforming the baseline, overall, the third model did not improve upon the original. This training cycle offers valuable insights into LoRA training, particularly the double descent phenomenon and the potential for generating images aligned with the fine-tuning dataset. These findings can be applied to a high-quality dataset of educational images and prompts, potentially addressing the problem of finding relevant visual aids.

5.2 Results of Model 4 [20]

5.2.1 Prompt. Image of the exchange of gasses, nutrients, and waste in a closed circulatory system.

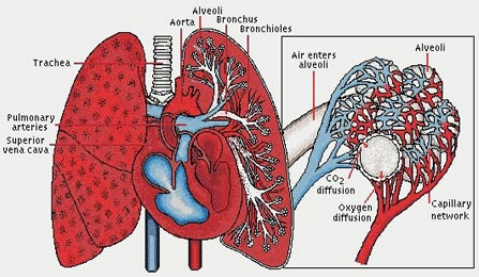

Figure 5: **Real Diagram [24]**

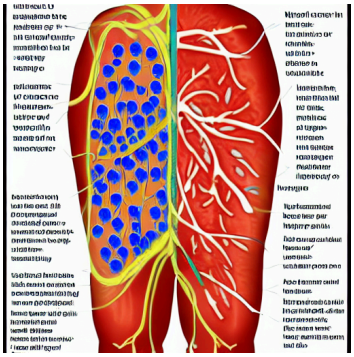

Figure 6: **LoRA Image**

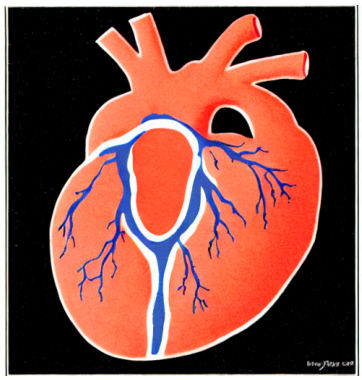

Figure 7: **Baseline Image**

Our fourth model was trained on our custom web-scraped dataset [21], despite concerns about the quality of prompts and images:

- **Image Complexity**: The generated images were more complex than those from the baseline model. We can see the added complexity of the image in Figure 6 over that of Figure 7.
- **Text in Diagrams**: The model attempted to write text to describe diagrams, but the text was illegible and diagrams often incomplete, as seen in Figure 6.

While the results were disappointing, they highlighted the complexity of the task. The fourth model might be in the middle of a double descent, similar to the third model. Improving the dataset quality and extending training time could significantly enhance results. These findings emphasize the challenge of creating explanatory images from complex text, underscoring the need for high-quality datasets.

5.3 Results of Model 5 [22]

Due to dataset concerns, our fifth model reverted to the smaller, hand curated dataset [16]. We used 18 prompts to generate images from both the baseline CompVis/stable-diffusion-v1-4 and our final fine-tuned model. A user evaluation survey was conducted, with reviewers rating images on a scale of 1 to 5 across three metrics: image quality, relevance to prompt, and usability. The results are shown in the following figures.

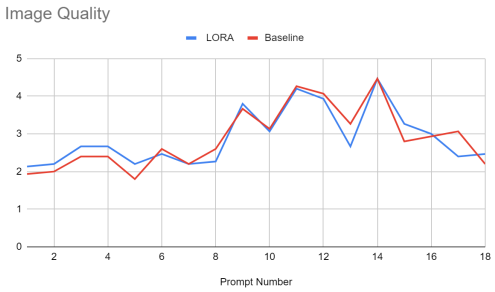

Figure 8: **Image Quality**

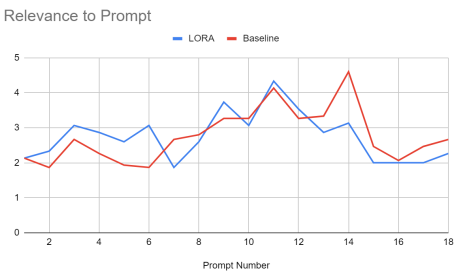

Figure 9: **Relevance to Prompt**

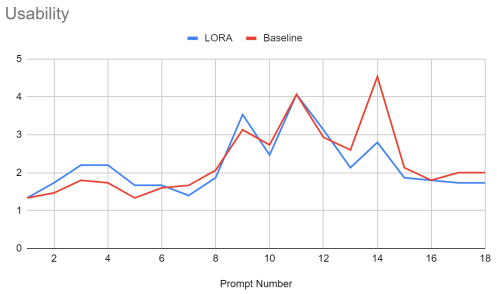

Figure 10: **Usability**

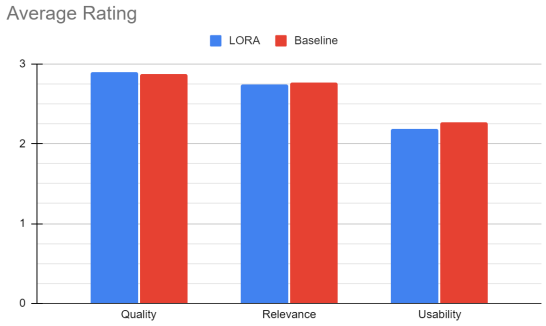

Figure 11: **Average Rating**

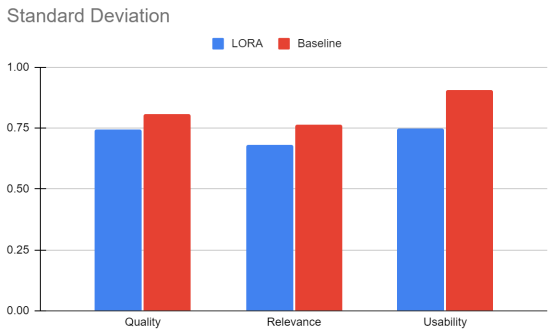

Figure 12: **Standard Deviation**

5.3.1 Evaluation Metrics. Trends for each metric can be seen in Figures 8, 9, and 10, with the red line being the baseline model and the blue line being the LoRA model.

- **Image Quality**: Fine-tuned model scored 2.89; Baseline model scored 2.88.
- **Relevance to Prompt**: Fine-tuned model scored 2.75; Baseline model scored 2.76.
- **Usability**: Fine-tuned model scored 2.19; Baseline model scored 2.27.

The average standard deviation for the fine-tuned model was 0.725, compared to 0.83 for the baseline model, as shown across the three metrics in Figure 11. These findings indicate no meaningful difference in scores between the fine-tuned and baseline models. However, the lower standard deviation for the fine-tuned model suggests greater consistency in image generation, as seen in Figure 12. This consistency is promising for generating more reliable visual aids, potentially making learning more efficient as images become more uniformly relevant and useful.

6 Limitations

6.1 LLM Limitations

6.1.1 Lack of Research and Datasets. There was a scarcity of research indicating what constitutes a "good" prompt—i.e., a prompt that would generate the highest quality image. We adopted two approaches: focusing on the most referenced sentences and creating short prompts based on training data. Existing models trained on user prompts from image generation sites were not compatible with our specific needs.

6.1.2 Difficulty in Validating Prompts. Validating the effectiveness of prompts required running a substantial number through our image model. Given our limited computational resources and tight time constraints, this was not feasible.

6.1.3 Necessity for Multiple Models. Creating effective prompts required using two models due to the complexity of the task. This approach was ambitious and less efficient than using a single model, leading to increased resource consumption and potential inefficiencies.

6.1.4 Computational Power. Training and running models frequently caused crashes in Colab due to insufficient RAM and usage limits. This significantly hindered our progress and necessitated frequent interruptions and troubleshooting.

6.2 Stable Diffusion Limitations

6.2.1 Lack of Data. The available data for training the image model was insufficient for our needs. To mitigate this, we created a custom dataset by scraping images from Google using prompts generated from a BART model. This process was time-consuming and required meticulous curation to ensure the images were both high quality and relevant.

6.2.2 Computational Power. Running complex models on a CPU was slow and impractical. The use of GPUs was necessary but limited due to their high cost and scarcity, which constrained our ability to train models efficiently.

6.2.3 API Query Limits. The Google API imposed a limit on the number of queries, restricting our capacity to scrape a

large volume of images. While Google Cloud credits helped to some extent, careful management of queries was essential to stay within these limits, often resulting in suboptimal data collection.

6.2.4 Limited Information and Dependencies Management. Detailed information about the models, particularly regarding dependencies, was lacking. Ensuring compatibility between different versions of libraries such as PyTorch and Python was critical but challenging. Upgrading to newer versions often led to errors, necessitating a trial-and-error approach to identify and use specific compatible versions.

7 Potential Improvements and Future Work

7.1 LLM Improvements

7.1.1 Single Model for Prompt Generation. Train a singular model that takes a large amount of text and outputs multiple prompts. This would streamline the LLM pipeline by eliminating one step, though it would require more research into effective prompt creation. To train this single LLM, a dataset mapping large texts to specific prompts would be necessary.

7.2 Dataset Improvements

7.2.1 Curate a High-Quality Dataset. Develop a comprehensive dataset with simplified versions of concepts and high-quality labels to improve training accuracy.

7.2.2 Feedback Loop for Graph Generation. Implement a feedback loop using LLM-generated Matplotlib code to create graphs, which can then be used to train the stable diffusion model.

7.3 Web Scraping Improvements

7.3.1 Multi-Site Image Scraping. Scrape images from multiple websites to enhance the relevance and quality of the collected images.

7.3.2 Automated Relevance Comparison. Utilize image-to-text models to automatically compare and validate the relevance of scraped images to the text.

7.3.3 Improved Error Handling. Enhance the web scraping script's error handling to ensure more robust data collection.

7.4 Usability Improvements

7.4.1 GUI Development. Create a graphical user interface (GUI) using tools such as Tkinter to improve the model's usability, making it more accessible for users without technical expertise.

7.5 Future Work

7.5.1 Support for Mathematical and Statistical Information. Extend the project's scope to include support for mathematical and statistical information by identifying texts that would benefit from specific charts and graphs. For such texts, prompt the LLM to generate Matplotlib code directly, rather than relying on the stable diffusion model to create these types of images.

8 Conclusion

In this project, we tackled the challenging task of automating the generation of explanatory visual aids from large textual content, aiming to enhance the teaching and learning experience. Our approach involved fine-tuning large language models (LLMs) and stable diffusion models to create relevant images that complement complex text. Despite encountering significant challenges, including limited datasets, computational constraints, difficulties in prompt validation, and poor evaluation metrics, our efforts provide valuable insights and offer areas for future research.

8.1 Key Contributions

8.1.1 Model Development and Training. We explored various models and training cycles, providing a detailed analysis of their performance and limitations. Our experiments with multiple checkpoints and extended training epochs shed light on the behavior of models during fine-tuning, particularly the double descent phenomenon.

8.1.2 Custom Dataset Creation. Due to the lack of existing datasets suitable for our task, we developed a custom dataset by scraping images and generating prompts. This dataset facilitated our experiments and provided a foundation for future work.

8.1.3 Evaluation Metrics and User Feedback. Through user evaluations, we assessed the quality, relevance, and usability of the generated images. Although the fine-tuned models did not consistently outperform the baseline, the reduced standard deviation in scores indicated improved consistency in image generation.

8.2 Limitations and Future Work

Our work faced several limitations, including the limited computational resources, the complexity of images, and the challenges in evaluating results. To address these issues, we propose several potential improvements and directions for future research:

8.2.1 Unified Model for Prompt Generation. Developing a single model that can generate multiple prompts from large texts would streamline the process and reduce inefficiencies.

8.2.2 Enhanced Datasets. Curating high-quality datasets with well-labeled concepts and creating a feedback loop for graph generation could significantly improve model performance.

8.2.3 Advanced Web Scraping Techniques. Implementing multi-site scraping, automated relevance comparison, and robust error handling would enhance the quality and relevance of collected images.

8.2.4 User-Friendly Interfaces. Developing a graphical user interface (GUI) would make the model more accessible to non-technical users, broadening its applicability.

8.2.5 Support for Specialized Content. Extending the project to include mathematical and statistical information by generating Matplotlib code for specific visual aids could provide targeted support for these fields.

8.3 Importance of the Work

This project contributes to the field of deep learning by addressing a practical and impactful problem in education. By automating the generation of explanatory images, we aim to reduce the time and effort required for educators and learners to find relevant visual aids, thereby enhancing the efficiency and effectiveness of the learning process. Our findings underscore the complexity of the task and highlight the potential for significant advancements through improved models, datasets, and methodologies.

In conclusion, while our project faced several challenges, it provided a solid foundation for future research and development in the area of automated visual aid generation. By building on our work and addressing the identified limitations, future efforts can make substantial strides toward creating effective, user-friendly tools that support education across various domains.

Member Contributions

Lucas Ellenberger:

- Researched possible baseline stable diffusion models. Tested several models that ultimately failed due to colab GPU RAM limits. Found the CompVis/stable-diffusion-v1-4 baseline model that we used as our baseline model for all testing and fine-tuning.
- Created the LoRA training script [13] and python file that performed the training [12].
- Trained all 5 stable diffusion LoRA models [15] [17] [18] [20] [22] that were described in this report using the training script and python file [12] [13].
- Created the small, hand curated dataset [16], which was used to train 3 of our LoRA models [15] [17] [22].
- Found the Caltech-UCSD cod-200 dataset [19] that was used to train one of our LoRA models [18].
- Generated baseline images for comparison, which can be seen in Figure 1 and Figure 7. Other baseline images were used for the user evaluation survey for our final model [22].
- Generated fine tuned images for all models for evaluation, which can be seen in Figure 2, Figure 3, Figure 4, and Figure 6. Other images from our final model [22] were used in the user evaluation survey.
- Created a google form to create our user evaluation survey, using the images generated from the baseline and final model [22].
- Performed manual evaluation of two models [18] [20], as described in section 5.1 and section 5.2 of this report.
- Aggregated the user evaluation survey results for our final model [22] into tangible metrics, as shown in Figure 8, Figure 9, Figure 10, Figure 11, and Figure 12. These results are analyzed in section 5.3 of this report.
- Wrote the majority of the following sections of this report: abstract, section 1, section 4.1, section 4.3, section 4.4, section 4.5, section 5, section 6.2, sections 7.2-7.5, and section 8.
- Revised and formatted this entire report, apart from section 2 and section 3 to follow the ACM guidelines.
- For both the mid quarter and final project presentation, cleaned slides to make them more readable in a presentation format.
- All GitHub contributions [25] can be seen on our group's repository [23].

Nia Balaji:

- All GitHub contributions [25] can be seen on our group's repository [23] and edit history on all submitted google docs.
- Researched the various LLM and Image generation models that we could potentially use, found the DalleE mini model that we used for our initial efforts.
- Researched the different datasets that we could potentially use for both the prompt generation and the image generation. (All research contributions can be seen in the project proposal document as well as the middle project presentation).
- Researched the methods of using Google Images to create a custom dataset, and initially tried to do this using BeautifulSoup4 and Scrapy. This approach had some issues since BeautifulSoup can only parse files in HTML/XNL files, and as we were trying to parse multiple images, to find the one that was most related to the prompt(using object detection, tagging, and embeddings), so I looked into other methods. Did extensive research on using ChromeDriver for web scraping.
- Created a web scraping script using Google API, Google Cloud Console, Selenium, ChromeDriver by accessing the Google API to collect and store images

from Google Images into a HuggingFace dataset [21] [26] [27].
- Researched how to create good prompts for text to image models by reading papers published online, including using prompt-tuning.
- Created the model to create text summarization and to generate prompts using the Facebook BART model (on HuggingFace) which was used to create validation sets for the Stable Diffusion model. Used TF-IDF vectorization to identify keywords.
- Completed the bulk of the Middle Project Presentation and Final Presentation, with assistance primarily from Lucas.
- Authored and finalized the entire Course Project Proposal, which encompassed a significant portion of the needed research, with assistance primarily from Lucas.

Christopher Casarez:

- Researched and attempted training LLM Llama-3 model for text Summarization [6]
- Researched and implemented open-source Hugging Face model Fine-Tuned T5 Small for Text Summarization [5]
- Implemented Prompt Tuning with both models
- Researched into fine tuning and Prompt Engineering [28]
- Worked on LLM Methodologies in the final project presentation.
- Worked on LLM Methodologies in this final report
- Researched Image Prompt Generation and and Prompt Validation [3]

Thierry Nguyen:

- Researched the possible LLM models that could be used for text summarization and prompt generation [5] [7] [8]
- Researched how to create better prompts by reading papers and experimenting with open-source models.
- Including Microsoft Promptist [4]
- Attempted to train a Llama-3 model for text summarization, which kept crashing due to resource limitations (even optimized with LoRA)
- Found and implemented the Hugging Face model for text summarization.
- Set up and prompt-tuned the Llama-3 model for prompt generation.
- Worked on the LLM portion for the final presentation slides
- Worked on LLM methodology, introduction, and limitations on the final report.

REFERENCES

[1] Z. Sun, “google/pegasus-xsum · Hugging Face,” huggingface.co, 2023. https://huggingface.co/google/pegasus-xsum.

[2] “Papers with Code - cnn_dailymail Benchmark (Summarization),” paperswithcode.com. https://paperswithcode.com/sota/summarization-on-cnn-dailymail.

[3] J. Guo, C. Wang, Y. Wu, E. Zhang, K. Wang, X. Xu, S. Song, H. Shi, G. Huang *Zero-Shot Generative Model Adaptation via Image-Specific Prompt Learning*. 1 June 2023, https://arxiv.org/abs/2304.03119

[4] Y. Hao, Z. Chi, D. Liu, and F. Wei, “Optimizing Prompts for Text-to-Image Generation,” arXiv (Cornell University), Dec. 2022, doi: https://doi.org/10.48550/arxiv.2212.09611.

[5] A. Khoumane, “Falconsai/text_summarization · Hugging Face,” huggingface.co, 2024. https://huggingface.co/Falconsai/text_summarization.

[6] J. Spisak, “meta-llama/llama3,” GitHub, Apr. 2024. https://github.com/meta-llama/llama3/tree/main (accessed Jun. 12, 2024).

[7] P. Szemraj, “pszemraj/led-large-book-summary · Hugging Face,” huggingface.co, Nov. 30, 2022. https://huggingface.co/pszemraj/led-large-book-summary.

[8] P. Szemraj, “pszemraj/long-t5-tglobal-base-16384-book-summary · Hugging Face,” huggingface.co, 2024. https://huggingface.co/pszemraj/long-t5-tglobal-base-16384-book-summary.

[9] E. J. Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models,” arXiv:2106.09685 [cs], Oct. 2021, Accessed: Jun. 11, 2024. [Online]. Available: https://arxiv.org/abs/2106.09685.

[10] S. Ryu, “Low-rank Adaptation for Fast Text-to-Image Diffusion Fine-tuning,” GitHub, Apr. 14, 2023. https://github.com/cloneofsimo/lora.

[11] “PEFT,” huggingface.co, Dec. 08, 2022. https://huggingface.co/docs/peft/en/index (accessed Jun. 12, 2024).

[12] L. Ellenberger, “CSE144/train_text_to_image_lora.py at main · Lucas-Ellenberger/CSE144,” GitHub, May 2024. https://github.com/Lucas-Ellenberger/CSE144/blob/main/train_text_to_image_lora.py (accessed Jun. 12, 2024).

[13] L. Ellenberger, “CSE144/script_sd.sh at main · Lucas-Ellenberger/CSE144,” GitHub, May 2024. https://github.com/Lucas-Ellenberger/CSE144/blob/main/script_sd.sh (accessed Jun. 12, 2024).

[14] L. Ellenberger, “CSE144/Lora_Trainer.ipynb at main · Lucas-Ellenberger/CSE144,” GitHub, May 2024. https://github.com/Lucas-Ellenberger/CSE144/blob/main/Lora_Trainer.ipynb (accessed Jun. 12, 2024).

[15] L. Ellenberger, “LucasEllenberger/CSE144-small-lora · Hugging Face,” huggingface.co, May 2024. https://huggingface.co/LucasEllenberger/CSE144-small-lora (accessed Jun. 12, 2024).

[16] L. Ellenberger, “LucasEllenberger/ChemistryImages · Datasets at Hugging Face,” huggingface.co, May 2024. https://huggingface.co/datasets/LucasEllenberger/ChemistryImages (accessed Jun. 12, 2024).

[17] L. Ellenberger, “LucasEllenberger/CSE144-small-lora-2 · Hugging Face,” huggingface.co, May 2024. https://huggingface.co/LucasEllenberger/CSE144-small-lora-2 (accessed Jun. 12, 2024).

[18] L. Ellenberger, “LucasEllenberger/CSE144-bird-lora · Hugging Face,” huggingface.co, May 2024. https://huggingface.co/LucasEllenberger/CSE144-bird-lora (accessed Jun. 12, 2024).

[19] C. Wah, S. Branson, P. Welinder, P. Peronaand S. Belongie, “The Caltech-UCSD Birds-200-2011 Dataset”, California Institute of Technology, Jul. 2011.

[20] L. Ellenberger, “LucasEllenberger/CSE144-science · Hugging Face,” huggingface.co, May 2024. https://huggingface.co/LucasEllenberger/CSE144-science (accessed Jun. 12, 2024).

[21] N. Balaji, “niabalaji123/googlewebscrapedataset · Datasets at Hugging Face,” huggingface.co, May 2024. https://huggingface.co/datasets/niabalaji123/googlewebscrapedataset (accessed Jun. 12, 2024).

[22] L. Ellenberger, “LucasEllenberger/CSE144-chemistry-mini at main,” huggingface.co, May 2024.

https://huggingface.co/LucasEllenberger/CSE144-chemistry-mini/tree/main (accessed Jun. 12, 2024).

[23] L. Ellenberger, "Lucas-Ellenberger/CSE144," GitHub, Apr. 2024. https://github.com/Lucas-Ellenberger/CSE144 (accessed Jun. 12, 2024).

[24] "Gas Exchange: What we know," PEER Program. https://vetmed.tamu.edu/peer/what-we-know/.

[25] L. Ellenberger, "Contributors to Lucas-Ellenberger/CSE144," GitHub, May 2024. https://github.com/Lucas-Ellenberger/CSE144/graphs/contributors (accessed Jun. 12, 2024).

[26] N. Balaji, "niabalaji123/google_png · Datasets at Hugging Face," huggingface.co, May 2024. https://huggingface.co/datasets/niabalaji123/google_png (accessed Jun. 12, 2024).

[27] N. Balaji, "niabalaji123/googlesearchtrain · Datasets at Hugging Face," huggingface.co, May 2024. https://huggingface.co/datasets/niabalaji123/googlesearchtrain (accessed Jun. 12, 2024)

.[28] Trad F., Chehab A.. "Prompt Engineering or Fine-Tuning? A Case Study on Phishing Detection with Large Language Models." *MDPI*, Multidisciplinary Digital Publishing Institute, 6 Feb. 2024, www.mdpi.com/2504-4990/6/1/18.