Project Report: ComicCrafter AI

Submitted By:
Himanshu Kumar and Team
Date: March 27, 2025
Purpose: Industrial Training Project

1. Introduction

ComicCrafter AI is an AI-powered tool designed to generate 4-panel comic strips from user-provided text prompts. The project leverages advanced machine learning models to create engaging storylines and visually appealing comic panels, combining natural language processing (NLP) and image generation techniques. This tool aims to simplify the process of comic creation for enthusiasts, artists, and storytellers by automating the story generation and illustration process.

The project was developed as part of my industrial training to explore the application of AI in creative domains, specifically focusing on the integration of language models and diffusion-based image generation models.

2. Objective

The primary objectives of this project are:

- To develop a system that generates a 4-panel comic strip based on a user-provided text prompt.
- To integrate NLP for story generation and diffusion models for image creation.
- To provide a user-friendly interface for interacting with the system.
- To explore optimization techniques for efficient model performance on GPU/CPU.

3. Technologies Used

- **Programming Language**: Python 3.10
- **Libraries and Frameworks**:
  - **DistilGPT-2**: For generating comic storylines.
  - **Stable Diffusion 2.1**: For creating high-quality comic panel images.
  - **ControlNet (Canny)**: For integrating user sketches into the image generation process.
  - **Gradio**: For building a web-based user interface.
  - **PyTorch**: Backend for model handling and GPU/CPU optimization.
  - **OpenCV**: For image processing (Canny edge detection).
  - **Pillow, NumPy**: For image and array operations.
  - **Accelerate**: For model loading optimization.

Hardware:

- GPU: CUDA-enabled GPU (NVIDIA, CUDA 12.1)
- CPU: Fallback for systems without GPU support

4. System Architecture

The system is divided into the following components:

1. **Story Generation**:
    1. Uses DistilGPT-2 to generate a 4-part storyline (Introduction, Development, Climax, Conclusion) based on the user prompt.
    2. Each part is summarized and used to extract visual elements for image generation.
2. **Image Generation**:

    1. Stable Diffusion 2.1, combined with ControlNet, generates comic panels.
    2. ControlNet uses Canny edge detection on user-uploaded sketches to guide the image generation process.

3. **User Interface**:

    1. Built with Gradio, providing an interactive web interface to input prompts, upload sketches, and adjust generation settings.

4. **Optimization**:

    1. GPU/CPU detection for model execution.
    2. Memory optimizations like enable_model_cpu_offload() to reduce memory usage.

5. Methodology

1. **Setup**:
    1. Pre-trained models (DistilGPT-2, Stable Diffusion, ControlNet) were downloaded and stored locally.
    2. A virtual environment (comic_env) was created to manage dependencies.
2. **Story Generation**:

    1. The user provides a text prompt (e.g., "A superhero adventure").
    2. DistilGPT-2 generates a 4-part story, with each part focusing on a specific segment of the narrative.
    3. Visual elements are extracted from each part for image generation.

3. **Image Generation**:

1. Stable Diffusion generates images for each story part.
2. ControlNet uses user sketches (if provided) to guide the style and structure of the generated images.

4. **UI Development**:

    1. Gradio was used to create a web interface with input fields for prompts, sketches, and settings (e.g., inference steps, guidance scale).
    2. The interface displays the generated 4-panel comic strip.

5. **Testing:**

    1. The system was tested with various prompts (e.g., "A magical forest journey", "Space adventure").

    2. Performance was evaluated on both GPU and CPU setups.

6. Challenges Faced

- **Model Loading Issues**: Large models like Stable Diffusion required significant disk space and memory, which was mitigated using enable_model_cpu_offload().
- **Environment Setup**: Virtual environment issues (path mismatches) were resolved by recreating the environment.
- **Image Quality**: Initial images had artifacts, which were reduced by tuning the guidance scale and ControlNet conditioning scale.
- **Processing Time**: Image generation took 60-120 seconds per comic, which is acceptable but can be improved with further optimization.

7. Results

- The system successfully generates 4-panel comic strips from text prompts.
- Storylines are coherent and follow a logical narrative structure (Introduction → Development → Climax → Conclusion).
- Generated images are vibrant and align with the comic book style, with user sketches effectively guiding the output when provided.
- The Gradio interface is user-friendly, allowing easy interaction with the system.

Example Output:

- **Prompt**: "A young boy who gains the power of flight and protects his city"
- **Output**: A 4-panel comic strip showing:

    1. The boy discovering his powers.
    2. Training in the sky.
    3. Facing a villain in a dramatic showdown.
    4. Saving the city and being celebrated.

8. Future Scope

- **Faster Processing**: Integrate more optimization techniques (e.g., xformers for memory-efficient attention) to reduce generation time.
- **Enhanced Customization**: Add support for more user inputs (e.g., style selection, color themes).
- **Text in Comics**: Include speech bubbles and text overlays in the generated panels.
- **Cloud Deployment**: Deploy the system on a cloud platform for wider accessibility.

---

9. Conclusion

ComicCrafter AI demonstrates the potential of AI in creative applications like comic generation. By combining NLP and image generation, the project provides a seamless way to create comic strips, making it a valuable tool for storytellers and artists. The industrial training experience helped me gain hands-on knowledge of AI model integration, optimization, and UI development.

---

10. Acknowledgments

I would like to thank my mentors and peers for their guidance during this industrial training. Special thanks to the open-source community for providing the tools and libraries that made this project possible.