ComicCrafter AI Project Report by Ravikumar M

Self Overview

Hello All, I am **Ravikumar M**, an **Electronics and Communication Engineering (ECE) student** at **Sri Sairam Institute of Technology**. My areas of interest include **Artificial Intelligence, Machine Learning, Embedded Systems, and Software Development**. Over time, I have developed strong skills in **Python, Java, AI models, and UI development**, which have helped me work on innovative projects.

I had the **wonderful opportunity** to be selected as an **Intel Intern**, where I worked on the **ComicCrafter AI** project. This internship allowed me to explore **AI-powered content generation**, combining **natural language processing, computer vision, and interactive UI design** to create a unique web-based tool. Through this project, I have gained hands-on experience in **AI model integration, system architecture, and user-friendly application development**.

Project Overview

ComicCrafter AI (also known as "Cosmic Crafter AI") is an Intel Internship project that transforms text prompts into comic book-style visual narratives. This interactive web application uses AI models to generate both the narrative structure and visual elements of a comic strip based on user inputs. Built specifically for Google Colab with GPU acceleration, this tool allows users without programming experience to create personalized comic strips through a simple graphical interface.

Technical Architecture

The project is implemented as a Python application with a Gradio-based web interface, leveraging pretrained AI models for both text and image generation tasks. The system follows a modular architecture with distinct components for story generation, image creation, speech bubble integration, and layout arrangement.

Core Components

1. **Text Generation Model**: A lightweight text model (DistilGPT-2) that structures narrative elements
2. **Image Generation Model**: A Stable Diffusion implementation that visualizes panel scenes
3. **UI Interface**: A Gradio-based interface for user interaction
4. **Image Processing Utilities**: Functions for adding speech bubbles and assembling comic layouts
5. **Error Handling System**: Comprehensive error management with visual fallbacks

Evolution of Model Selection

I. Initial Model Considerations (Implied)

While not explicitly stated in the code, we can infer the project likely began with more complex models that proved problematic:

Text Generation: Initially may have used larger GPT-2 variants or other language models, which would have consumed excessive memory and computing resources.

Image Generation: Likely started with larger Stable Diffusion 2.x variants or models with higher parameter counts.

II. Current Implementation

The project now uses carefully selected lightweight models:

1. **Text Generation**: DistilGPT-2
    a) Significantly smaller than standard GPT-2
    b) Faster inference time
    c) Reduced memory consumption
    d) Provides sufficient capability for basic story structuring

2. **Image Generation**: Stable Diffusion v1.4 (CompVis/stable-diffusion-v1-4)
    a) Smaller and more efficient than SD 2.0 variants
    b) Compatible with memory optimization techniques (xformers, attention slicing)
    c) Provides adequate image quality for comic-style generation
    d) Works with float16 precision for reduced memory footprint

III. Rationale for Model Changes

The code suggests several motivations for shifting to lightweight models:

1. **Google Colab Constraints**:
    a) Limited GPU memory (typically 12-16GB)
    b) Variable resource allocation
    c) Session time limits

2. **Performance Optimization**:
    a) Reduced inference time per panel
    b) Lower memory consumption
    c) More reliable operation without OOM (Out of Memory) errors

3. **User Experience Considerations**:
   a) Faster end-to-end generation time
   b) More responsive interface
   c) Reduced likelihood of timeouts or crashes

Technical Implementation Details

GPU Acceleration

The application leverages CUDA-enabled GPUs for significant performance improvements:

```
device = "cuda" if torch.cuda.is_available() else "cpu"
pipe = pipe.to(device)
```

The code implements fallback to CPU when necessary but prioritizes GPU execution, checking availability before setup:

```
!nvidia-smi  # GPU status check
```

Memory Optimization Techniques

Several strategies are employed to maximize performance within memory constraints:

xformers Integration:

```
if hasattr(pipe, "enable_xformers_memory_efficient_attention"):
    pipe.enable_xformers_memory_efficient_attention()
```

Attention Slicing:

```
pipe.enable_attention_slicing(1)
```

Mixed Precision:

```
torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32
```

Safety Checker Removal:

```
safety_checker=None  # Disabled for speed
```

Optimized Scheduler:

```
scheduler = DPMSolverMultistepScheduler.from_pretrained(...)
```

Error Handling and Resilience

The application includes comprehensive error management:

1. **Exception Catching**: Every major function is wrapped in try-except blocks
2. **Visual Error Feedback**: Errors produce informative images rather than crashing
3. **Fallback Mechanisms**: Alternative approaches when primary methods fail
4. **Detailed Logging**: Comprehensive logging for debugging

User Interface

The application provides a streamlined interface built with Gradio:

Input Controls:

1. Text prompt field
2. Panel count slider (1-4)
3. Art style selection dropdown

Output Display:

1. Comic strip visualization
2. JSON story structure representation
3. Status updates

Progress Indicators:

1. Step-by-step progress feedback
2. Status messages during generation

Workflow Process

Story Structure Generation:

1. Simplified story progression (introduction → conflict → resolution)
2. Character identification
3. Panel scene descriptions
4. Basic dialogue elements

Panel Image Creation:

1. Scene-specific prompt construction
2. Style-consistent visualization
3. Optimized inference parameters (reduced steps: 20)

Speech Bubble Addition:

1. Text overlay creation
2. Simple white bubbles with black text
3. Automatic positioning

Comic Assembly:

1. Grid-based panel arrangement
2. Adaptive layout based on panel count
3. Border addition for visual clarity

Performance Considerations

Several design decisions target performance optimization:

1. **Reduced Panel Count**: Maximum of 4 panels (likely reduced from a higher initial count)
2. **Simplified Prompts**: Streamlined prompt construction
3. **Lower Inference Steps**: 20 steps rather than the typical 50+
4. **Simple Speech Bubbles**: Basic text boxes rather than complex bubble shapes
5. **Default Settings**: UI defaults to 2 panels for faster generation

User Experience Enhancements

Helpful Tips Section:

```
## Tips for Success
- Start with 2 panels for faster generation
- Keep prompts simple and focused
- If you get an error, try a different style or fewer panels
- The model works best with simple character descriptions
```

Progress Updates: Real-time status information during generation

Default Example: Pre-filled prompt to demonstrate capabilities

Multiple Art Styles: Options for different visual aesthetics

Limitations and Constraints

1. **Panel Count**: Limited to 4 panels maximum
2. **Simple Story Structure**: Basic narrative progression
3. **Generic Dialogue**: Limited dialogue sophistication
4. **Basic Panel Layout**: Fixed grid arrangement
5. **Resource Dependency**: Requires Google Colab with GPU access

Future Enhancement Opportunities

1. **Model Upgrades**: Integration of more efficient models as they become available
2. **Advanced Layouts**: Dynamic panel sizing and creative arrangements
3. **Improved Dialogue**: More contextually relevant text generation
4. **Style Consistency**: Better character appearance consistency across panels
5. **User Customization**: Panel editing and story modification tools
6. **Export Options**: PDF or image sequence export functionality
7. **Local Deployment**: Containerized version for local execution

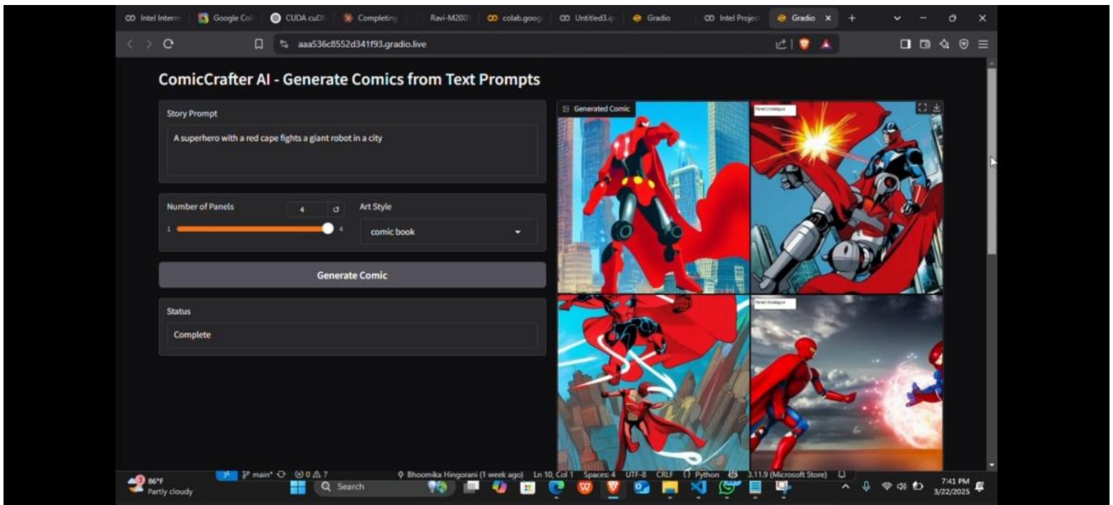
Conclusion

Cosmic Crafter AI, developed as an Intel Internship project, successfully demonstrates the application of optimized AI models for creative content generation within the constraints of cloud-based notebook environments. By carefully balancing model capabilities with performance requirements, the system provides an accessible tool for comic creation without requiring specialized hardware or technical expertise. The modular design allows for future enhancements while maintaining a focus on reliability and user experience.

OUTPUT VIDEO LINK:

Click here - https://drive.google.com/file/d/10eeVmcpX_WhK5hC4d78dLisjpbuWqWwf/view?usp=sharing

OUTPUT IMAGES



aaa536c8552d341f93.gradio.live

ComicCrafter AI - Generate Comics from Text Prompts

Story Prompt

A superhero with a red cape fights a giant robot in a city

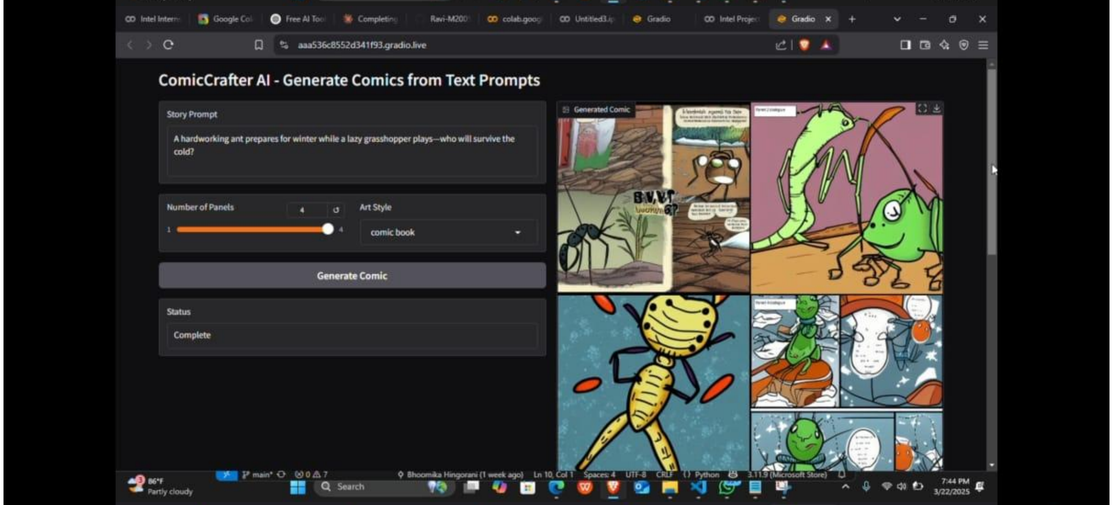
Number of Panels: 4

Art Style: comic book

Generate Comic

Status

Complete



aaa536c8552d341f93.gradio.live

ComicCrafter AI - Generate Comics from Text Prompts

Story Prompt

A hardworking ant prepares for winter while a lazy grasshopper plays—who will survive the cold?

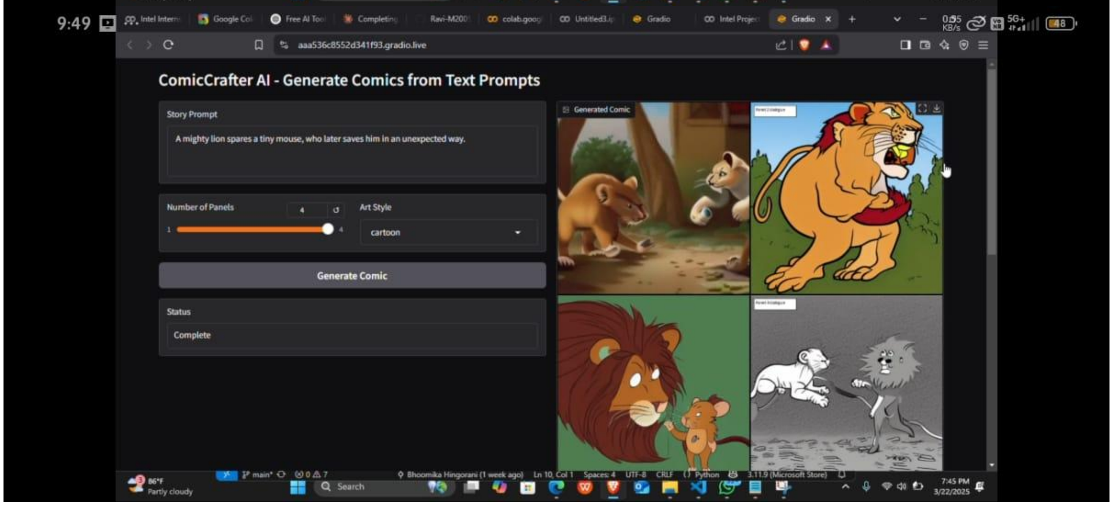
Number of Panels: 4

Art Style: comic book

Generate Comic

Status

Complete



aaa536c8552d341f93.gradio.live

ComicCrafter AI - Generate Comics from Text Prompts

Story Prompt

A mighty lion spares a tiny mouse, who later saves him in an unexpected way.

Number of Panels: 4

Art Style: cartoon

Generate Comic

Status

Complete