Multi-Modal Dynamic Instance Invoker for More Diverse ML Applications

Abubakr Osama Abubakr Osman, C O - 2 0 1 8 - 0 1 1
Final Graduation Thesis, Bachelor of Computer Science, University of Medical Sciences and Technology, Sudan.
+249-11-699-0109 | mrabubakrosama@gmail.com | github.com/SetuBaru/MultiModal-Dynamic-Instance-Invokers

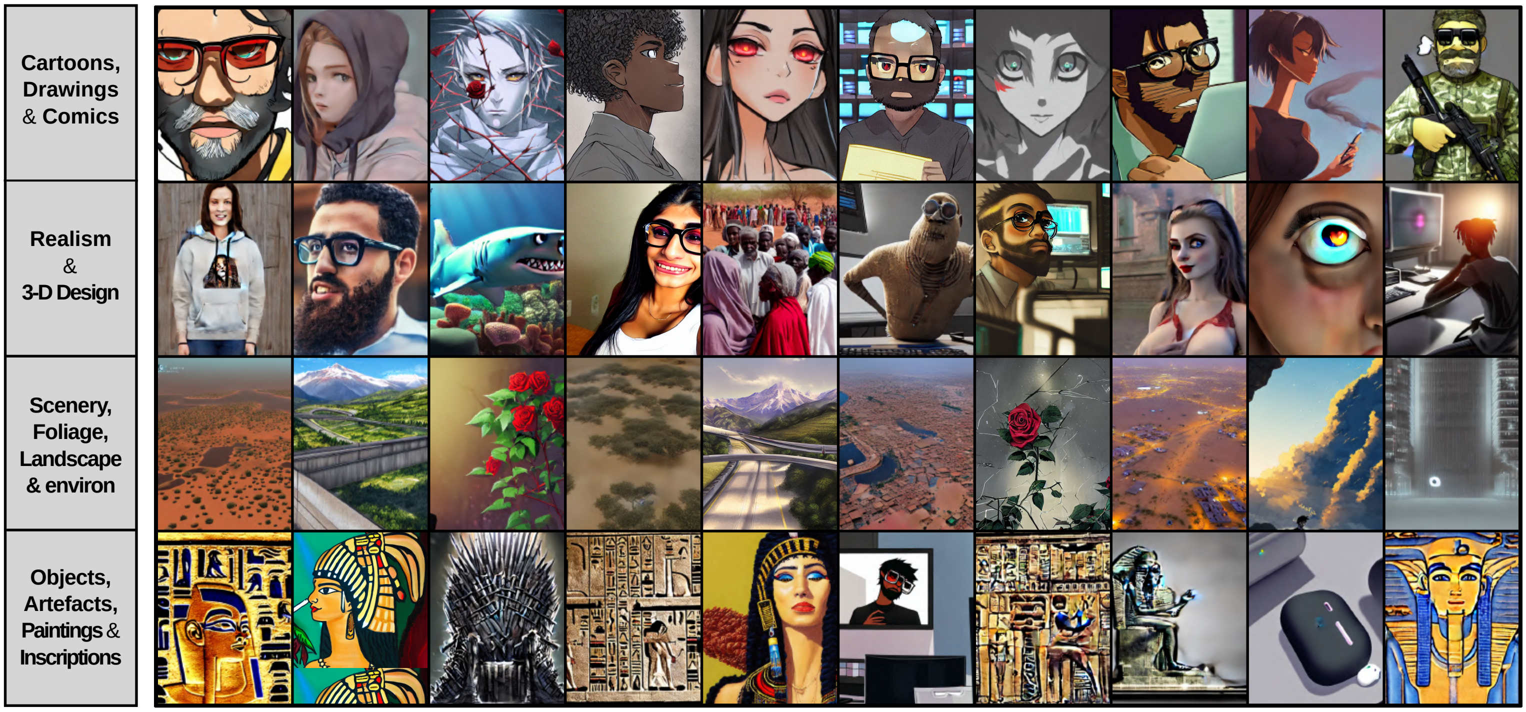


Figure 1 Shows a set of Image generated through our novel approach**[1]**. Our chosen approach technique**[2]** is able to achieve the above results by adding a model localisation attention layer**[3]** layer, which preloads our Meta-Verse Stability Diffusion Model**[5]**. Utilising our novel approach our model is able to outperform the current state of the art model**[6]**, with our model able to outperform in terms of diversity, quality consistency & performance**[7]**.

Abstract

Driven by a multitude of groundbreaking developments within the large subfield of Computer Generated Graphics, a model has gained wide-spread traction, quickly becoming the community's goto standard for autonomous image generation. Stable diffusion**[1]** is a fully open-source & community backed research initiative, providing utter-transparency & complete access to all of it's underlying architecture. As a model that comes based off another popular approach known as latent-diffusion**[2],** stable diffusion delivers impressive performance, that competes with other powerful models in the domain of conditional image synthesis.

Results generated through SD have not only competed other top models but have also shown commendable abilities delivering better performance in terms of conditional image generation, overall sample diversity, overall sample quality & resultant aesthetics.

CHAPTER 1

1.1 Introduction

It has long been man's dream to realise his ideas, turning imagination or thoughts into reality.

Modern Machine Augmented Approaches techniques have brought the modern man closer to that dream than ever before, bringing together the pieces that were needed to build his dream.

1000's of years ago it took skill & talent to create a masterpiece, about 30 years ago it took immaculate effort & preserves to create art through electronic machines, something not possible years ago. Today we use prompt driven Machine Learning Techniques to turn Intents into abstract visuals. Something that we only dreamed of in the past, all this was made possible due to the advent and evolution of computational techniques.

1.2 Problem Statement

The Path from Ideation to conceptualisatin is very tedious, resource extensive & time exhaustive. The process of content creation has yet to catch up to pace of the Modern Computational Developments, a fact that remains prominent despite the existence & appropriation of other application ready alternatives. This result is mainly due to the high skill-cap entailed by utilisation of such alternatives and in part due to their limited domain functionality & application scope, current applications of generative model are limited by current computational capabilities and enticed performance trade offs, involved in the production of models that are specialised vs ones that are more generalised.

1.3 Solution

Rather than ask, the traditional question of how many neurons can be fit in a neural network?

We ask the question of how many networks can fit in a neuron.

In an approach aimed towards more powerful ML learning architecture , we propose a dynamic model instancing technique, rather than the tradition static approach strategy.

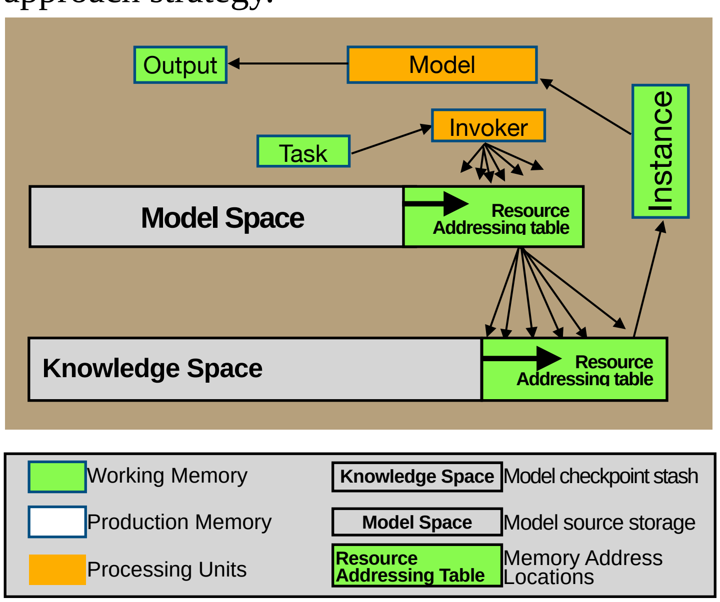


Figure 2 Shows the proposed model architecture pipeline, showing the flow of data through our model, with a legend for the model components at the bottom of the figure

Encapsulating the design from figure 2, would be

- A Developer friendly framework & API.

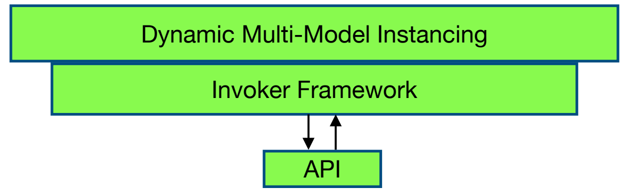


Figure 4 Shows the proposed back-end / core model structure

- A user-friendly Cross-Platform UI .

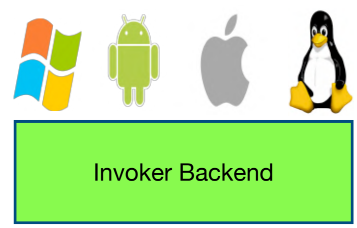


Figure 5 Shows our approach towards a multi-platform Invoker Backend, should be accessible as both a CLI tool and a user friendly GUI.

1.4 Objectives

Our aim from this project, is the development of an optimised approach for model inference, one that implements our proposed solution for dynamic multi-modal instancing.

- Model-oriented domain expansion & diversification.

- Improved Overall Performance.

- Development of a Community Driven ML Framework, Developer toolset and pipeline.

- Releasing a stable & multi-platform Easy to use UI.

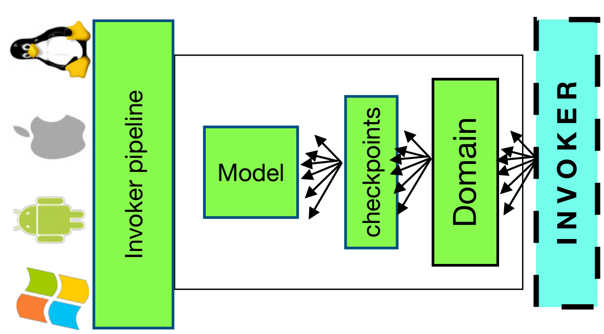


Figure 6 Shows a high level overview of the invoker pipeline and its composite modules.

1.5 Methodology

First By allowing pre-trained models to do what they were trained to do, harness their individual specialisation and narrow domain expertise, our approach is centred around dynamic model instancing, through a higher faculty of task allocation, we present our dynamic multi-modal agent system, enhancing model specific generalisations and appropriation based on model variation.

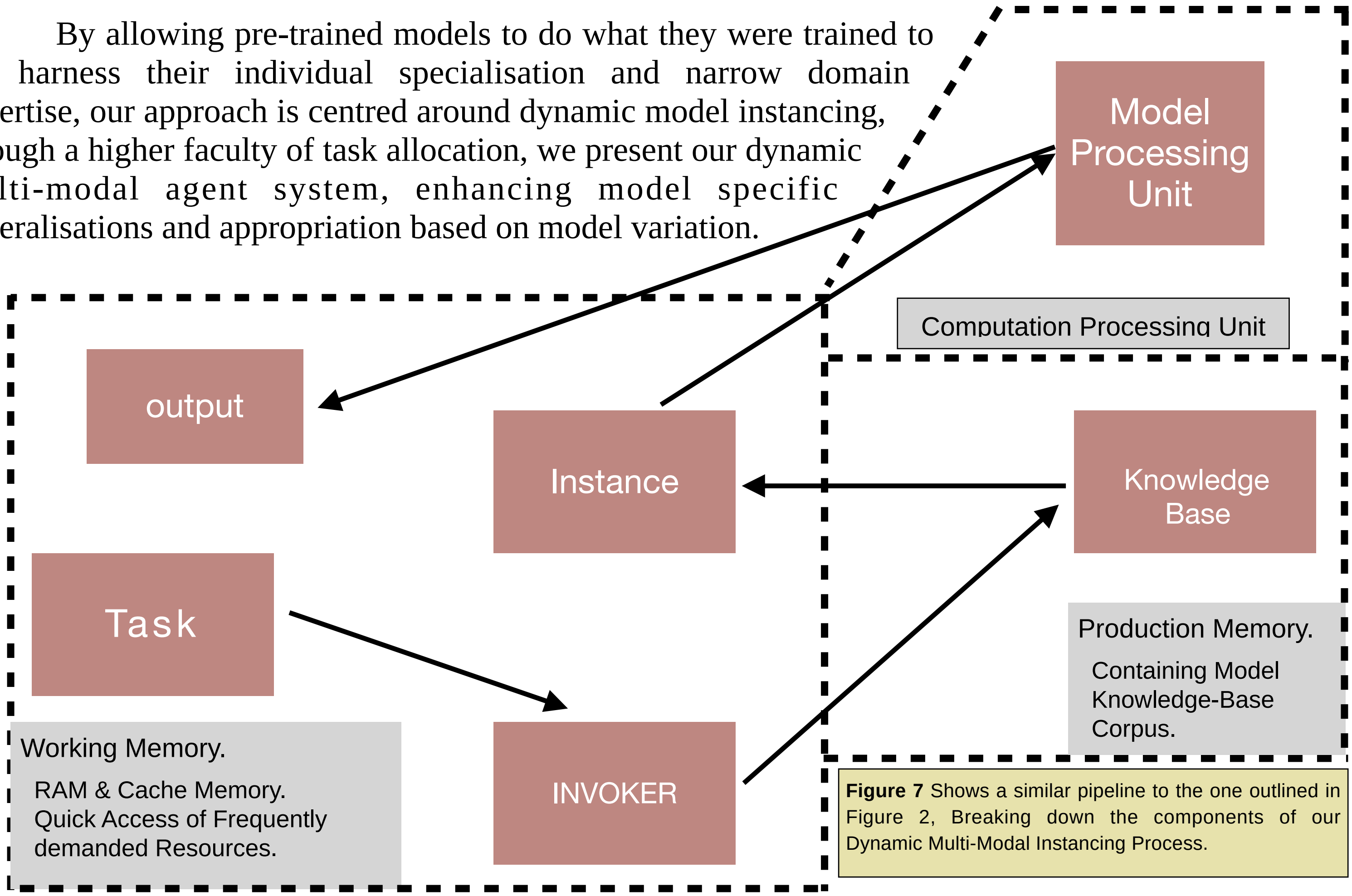


Figure 7 Shows a similar pipeline to the one outlined in Figure 2, Breaking down the components of our Dynamic Multi-Modal Instancing Process.

Figure 3 Shows the results we were able to generate using our approach. in the above figure we were able to take advantage of both the contributions of stable diffusion and its predecessor Waifu Diffusion, which allowed us to create the above samples.

We begin our proposed approach by conducting a model conditioning process for an open source model called Stable Diffusion. While the model does come with predefined set of weights, we were able to get our hands on pre-trained variant called Waifu diffusion, I then implemented my dynamic multi-modal instancing model to effectively and seamlessly switch between the original model instance and the variant instance, it came to our notice while running this model that it was able to produce a much more diverse results, without adding to computational requirement.

We Initialised our approach to implementing our model through resource pooling.

We then create create a working environment to serve as a container for the gathered resources.

Setting up & building all of the model dependencies & ROM file data. ROM files here consisting of Model setups & variations.

assets	17 days ago	
CompVis	6 days ago	
configs	21 days ago	
data	21 days ago	
diffusers	20 days ago	
ldm	21 days ago	
models	24 days ago	
outputs	21 days ago	
pytorch-apple-silicon-benchmarks	19 days ago	
scripts	17 days ago	
unfiltered-diffusers	15 days ago	
variants	11 days ago	
venv	20 days ago	
MyDemo.ipynb	18 hours ago	1.38 MB
convert_original_stable_diffusion_to_diffusers.py	20 days ago	29.5 kB
diffusion.png	19 days ago	311 kB
environment-mac.yaml	24 days ago	736 B
environment.yaml	24 days ago	734 B
LICENSE	24 days ago	14.4 kB
main.py	24 days ago	28.2 kB
notebook_helpers.py	24 days ago	10.3 kB
pygui-1-2-5.zip	15 days ago	10.4 kB
README.md	24 days ago	21.7 kB
requirements.txt	24 days ago	550 B
setup.py	24 days ago	233 B
Stable_Diffusion_v1_Model_Card.md	24 days ago	9.34 kB
v1-inference.yaml	20 days ago	1.87 kB

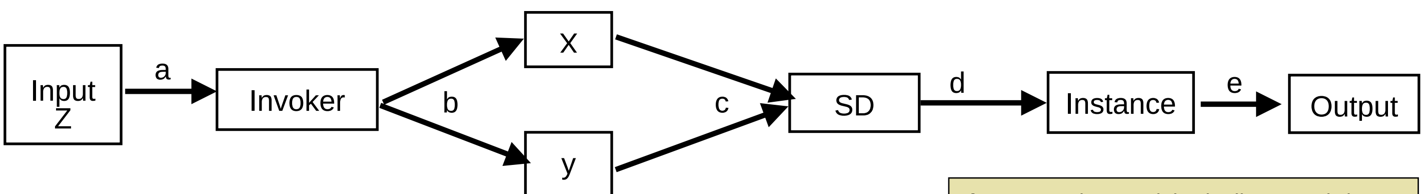
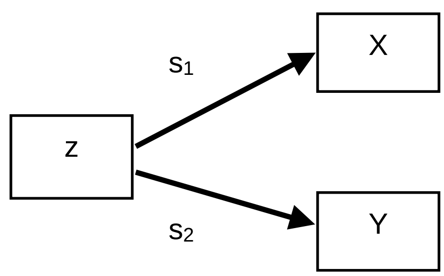


Figure 8 The model pipeline used in our current implementation.

Following the model approach in Figure 8, our steps were as follows.

step a - step b We began our approach by implementing a simple on hot encoding vector for our stable diffusion model domain scope, those are “Waifu Diffusion” and “Stable-Diffusion-v1-4”.

step c Our Input Z is now a 1 hot encoding vector of model X and Y. This process takes place at the invoker which then calculates the likelihood that Z is either of X or Y.



Where s_1 & s_2, are a measure of how close vector Z is to X and Y respectively.

In a sense we’re a fuzzy logic layer for simple probabilistic inference, it could also optionally be a rule-base layer depending on the use-case and task flexibility.

step a - steps c Allow us to run dynamic optimisations based on z, this allows us to pick the most adept model implementation (ROM) to undertake the task.

step d creates an instance of the decision and ships it for processing.

Running the model most adept to achieving the best performance a forward pass through our network architecture looks like this

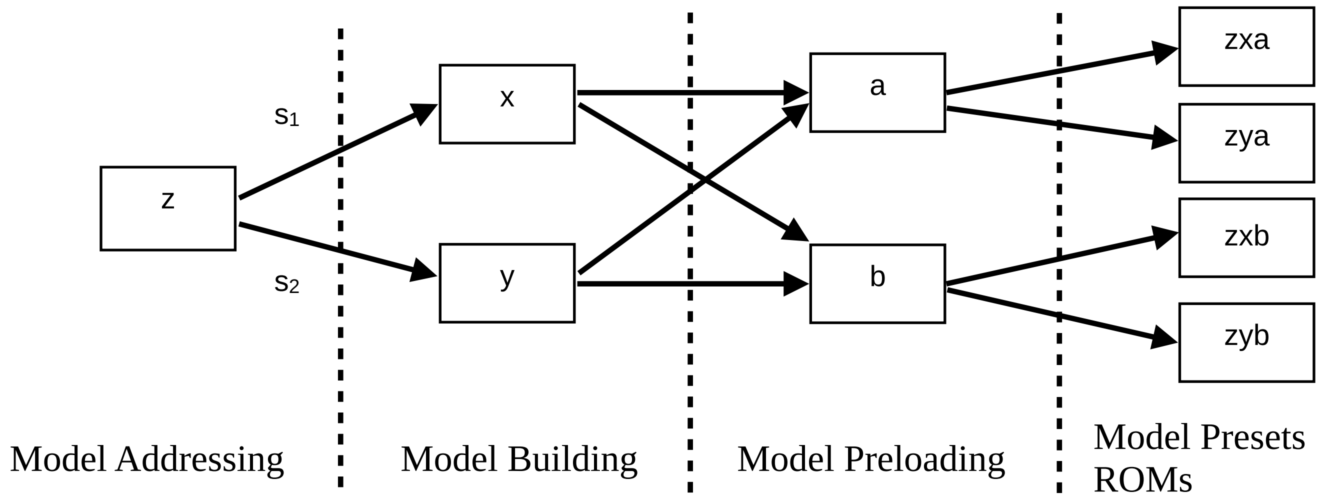


Figure 9 The above figure outlines the steps in our model instancing process.

This instancing process takes place using the runtime machine's processing unit, here our experiments within this paper were conducted within a localised instance on our WorkStation's integrated cpu.

In my case this was a 3.2 GHz 8-Core Intel Xeon W performing at an average of about 7.34 s/it.

1.6 Implementation

Here we import the necessary dependancies to run our experiment, on a jupyter notebook

```
In [1]: import os
        import torch
        import cv2
        import numpy as np
        import uuid
        import matplotlib.pyplot as plt
        %matplotlib inline

        print(torch.backends.mps.is_available())
        print(torch.backends.mps.is_built())

        True
        True
```

Here we build a function to save and show images on the jupyter notebook

```
In [2]: def show_img(img_path):
            if os.path.isfile(img_path) is False:
                print('Invalid Choice! Please Choose a different Entry and try again.')
                return False
            img = cv2.imread(img_path)
            plt.imshow(img)

        def save_img(img_handle, location=None):
            if img_handle == None:
                print("Error Img Handle Object Must be off string type! please make sure the img_obj is
                return False
            elif location == None:
                _loc = "/Users/mohammedabbarroh/Desktop"
            else:
                _loc = location
            temp_filename=str(uuid.uuid4().hex)
            img_path = f"{_loc}/{temp_filename}.png"
            image.save(img_path)
            return img_path
```

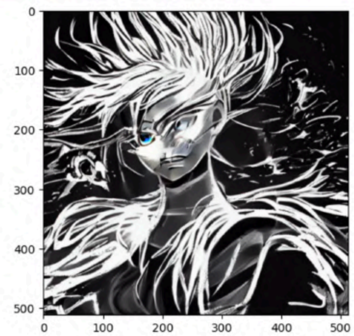
Where our instancing happens, instance is piped to computational unit

```
In [3]: # make sure you're logged in with `huggingface-cli login`
        from diffusers import StableDiffusionPipeline

        DEVICE1 = 'mps'
        DEVICE2 = 'cpu'
        DEVICE3 = 'mkldnn'
        DEVICE4 = 'vulkan'
        DEVICE5 = 'ipu'
        DEVICE6 = 'xpu'
        DEVICE7 = 'opencl'
        DEVICE8 = 'hpu'
        DEVICE9 = 'gpu'

        pipe = StableDiffusionPipeline.from_pretrained("CompVis/stable-diffusion-v1-4")
        pipe = pipe.to(DEVICE2)
```

Prompt input & segmentation

```
In [4]: prompts = "1girl, flaming eyes, fiery red hair, frozen, between a black and white path, anime, t
```

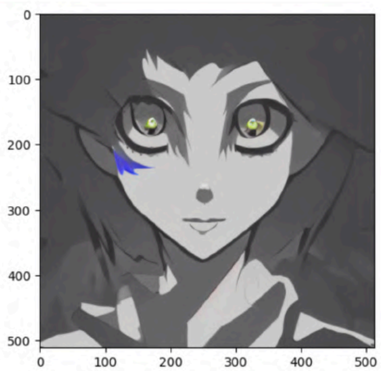
```
In [6]: image = pipe(prompts, num_inference_steps=10).images[0]
        100% 11/11 [01:20<00:00, 7.36s/it]

In [7]: x = save_img(image)

In [8]: show_img(x)
```



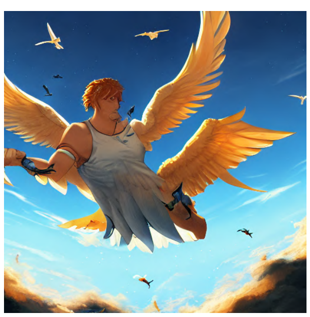
Model optimisations happen based on pipeline meta-data, this affects variables such as num_inference_steps & guidance.

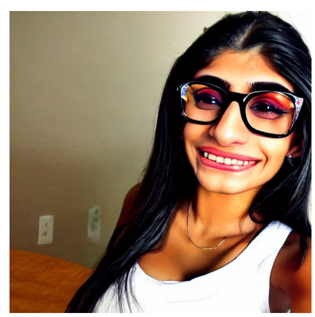
1.7 Results

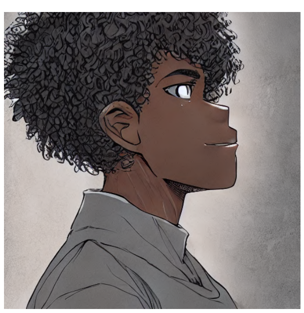
```
100% 24/24 [02:58<00:00, 7.35s/it]

In [16]: x = save_img(x)
         show_img(x)
```

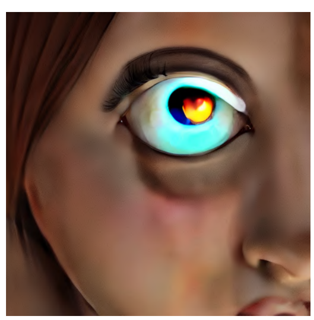


2 minutes 58 seconds to run 24 inference steps using this approach.

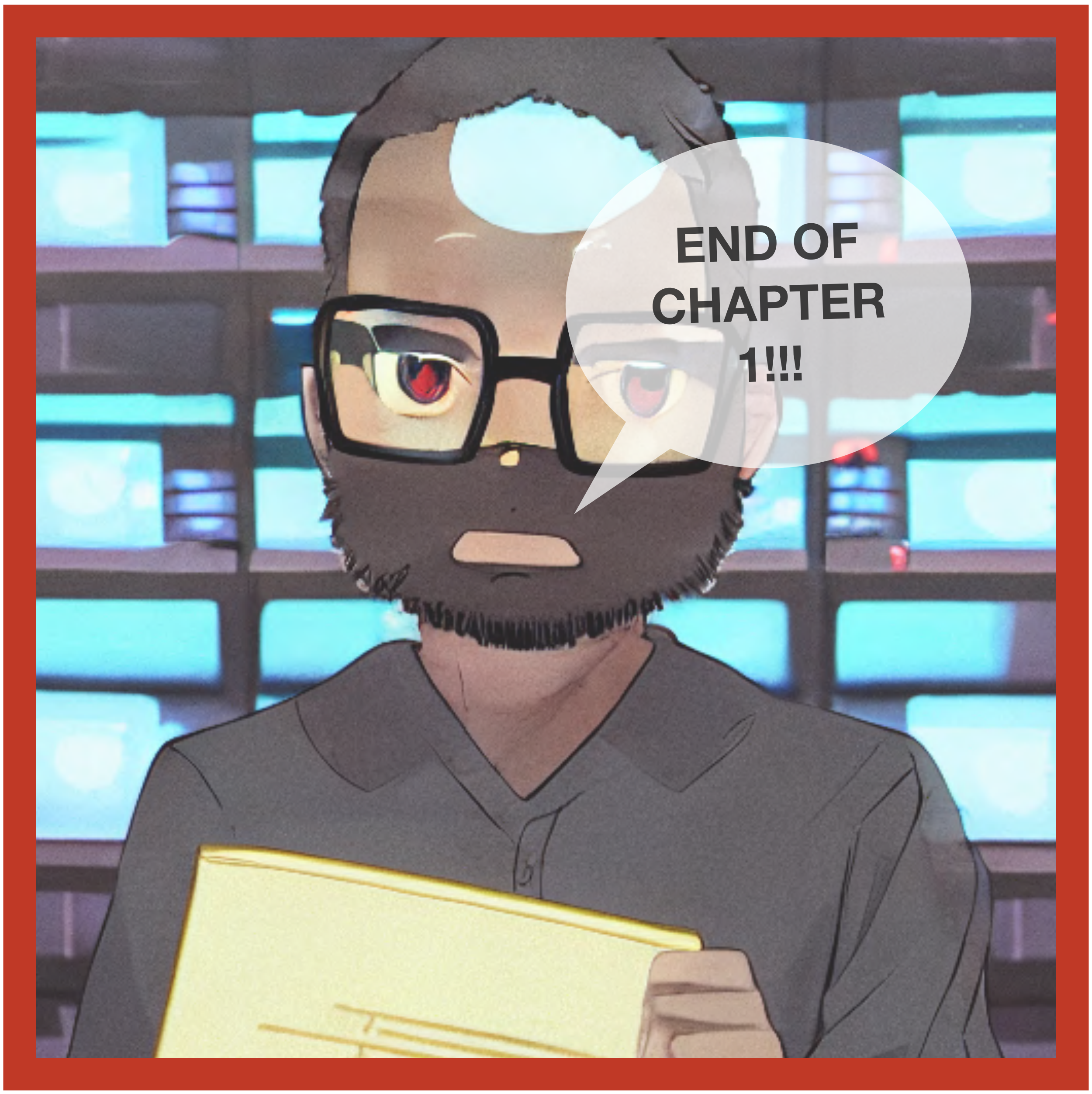








1.7 Tools & Frameworks



CHAPTER 2

2.1 Goals

- Development of User Friendly Interface built upon our Art Invoker Technique.
- Development of an improve Invoker Backend.
- Development of An API wrapper for the Invoker Backend.
- Creation of our ROMs (Models, Initialisations, Presets, Instance Blueprints, etc).

2.2 Requirements

- Acquisition of Resources & Building ROM files that promote improved synchronised throughput. *(i.e if user uses this model, then this model will add this functionality or this variation on output)*
- *Building Invoker Synchronisation Modelling Approach / Mechanism.*
- *Creating a cross-platform CLI-interface.*
- *Create a cross-platform GUI demo.*
- *Conduct Closed Alpha Testing Round.*
- *Release Relevant Documentation & Refine Approach.*
- *Drop The Open Alpha Release*
  ETA: LATE 2023

2.3 Related Work

HuggingFace Transformers API
https://huggingface.co/docs/transformers/index

Transformers

Replika AI Companion
https://replika.com/

Replika

Open AI's DALE-2
https://openai.com/dall-e-2/

Google Imagen
https://imagen.research.google/

Google

Midjourney
https://www.midjourney.com/home/

Ebsynth
https://ebsynth.com/

EBSYNTH . COM

figure 9 Shows a vintage ROM DISC GAME Cartridge.

2.4 References

[1] CompVis. (n.d.). *COMPVIS/stable-diffusion: A latent text-to-image diffusion model*. GitHub. Retrieved November 3, 2022, from https://github.com/CompVis/stable-diffusion

[2] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022, April 13). *High-resolution image synthesis with Latent Diffusion Models*. arXiv.org. Retrieved November 3, 2022, from https://arxiv.org/abs/2112.10752

[3] Dhariwal, P., & Nichol, A. (2021, June 1). *Diffusion models beat gans on image synthesis*. arXiv.org. Retrieved November 3, 2022, from https://arxiv.org/abs/2105.05233

[4] Nichol, A., & Dhariwal, P. (2021, February 18). *Improved denoising diffusion probabilistic models*. arXiv.org. Retrieved November 3, 2022, from https://arxiv.org/abs/2102.09672

END OF CHAPTER **TWO**