Auto-Generation of Graphics Mods
David Blankenship

Pitch:

The application I have developed is specifically meant for gamers looking to apply graphics mods to images in their games and should allow a user to take a set of images in a folder and create a generated image of the original in the style they desire. This is a ready-made application that will allow them to generate a variety of styles as they prefer and the value derived is from player satisfaction with the new graphical style. There are secondary uses as well for general users looking to simply convert a set of images into a style they prefer where, for example, people playing Tabletop Role Playing games often find images like their idea of their character's appearance.

Data source:

The original images are from the game ΔV: Rings of Saturn which can be found on Steam here: https://store.steampowered.com/app/846030/V_Rings_of_Saturn/

I used the following models on huggingface:

BLIP for automated caption generation:

https://huggingface.co/Salesforce/blip-image-captioning-base

Image Models to create based off of the generated prompt:

https://huggingface.co/runwayml/stable-diffusion-v1-5

https://huggingface.co/cagliostrolab/animagine-xl-3.1/tree/main/vae

https://huggingface.co/hakurei/waifu-diffusion

https://huggingface.co/emilianJR/epiCRealism

https://huggingface.co/sd-concepts-library/gta5-artwork

https://huggingface.co/nitrosocke/Ghibli-Diffusion

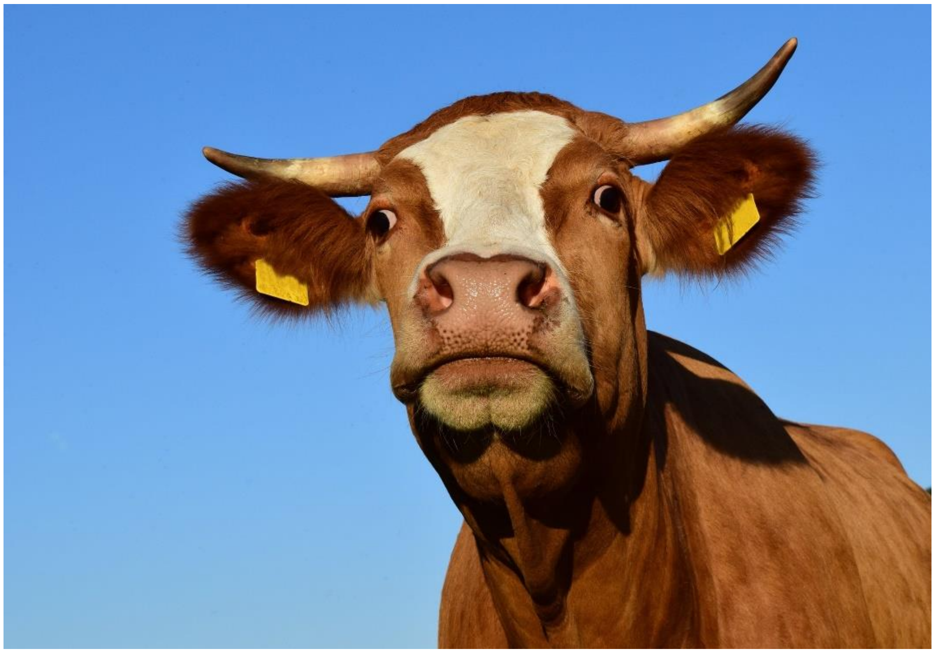
Model and data justification:

I have elected to use these models for the high-quality images and variety of styles that they allow. Additionally, I intend to use this to generate my own graphics mods and so I chose models that I personally am interested in. That being said, my application is generalizable to any model as an unlisted model can be manually entered.

Auto-Generation of Graphics Mods
David Blankenship

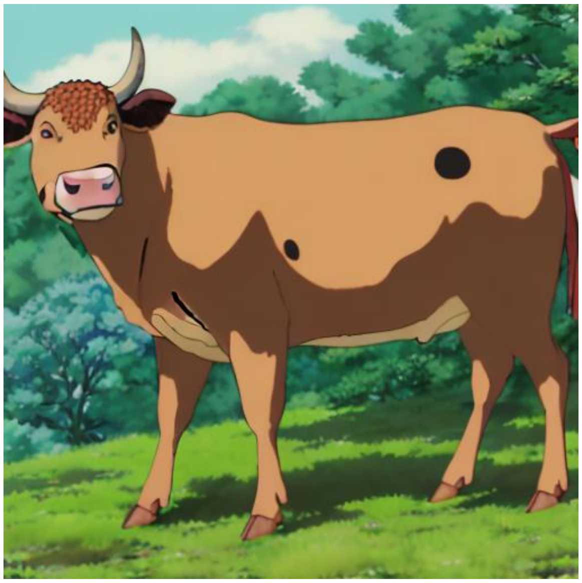
Commented examples:

For the single image test runs my original image is the following:



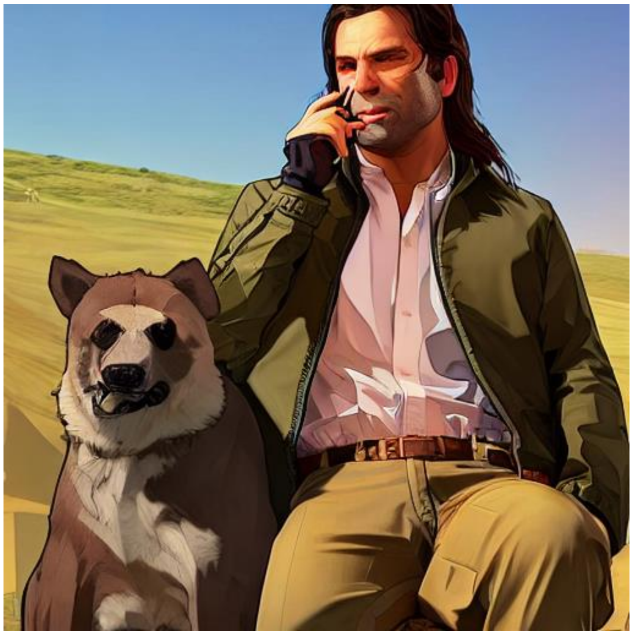
The prompt I received when running through BLIP is as follows: “a brown cow with a yellow tag on its ear”. It seems quite accurate. I tested several different styles and below are a few examples.

Ghibli Style



Auto-Generation of Graphics Mods
David Blankenship

GTA5 Style

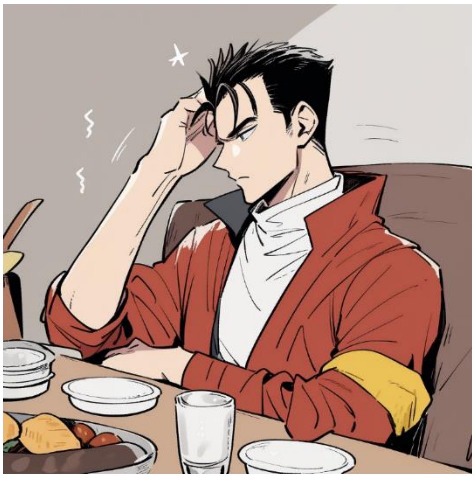


This image was run with 200 inference steps (most of the others range from 20-40). Looking at the dog I suspect any cow in the early steps may have morphed into the dog and a generic GTA5-esque image was created. I suspect this can be almost seen as an overfitting of the gta5 token that this model uses that seems to have washed away all of the other features in the prompt.

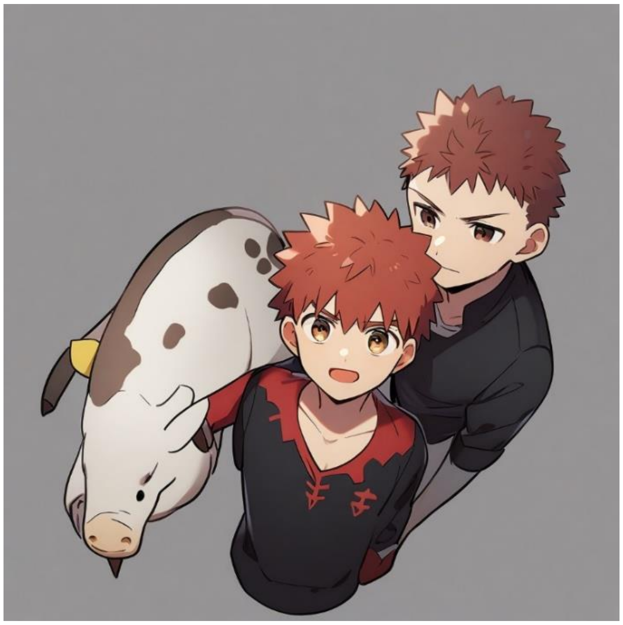
Animagine 3.1 Anime Style

Of note with the animagine 3.1 model is that it's prompt input is supposed to have the following format:

"1girl/1boy, character name, from what series, everything else in any order."

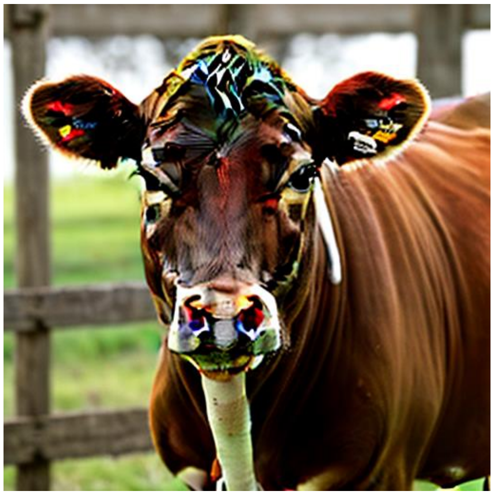


The input for this image was 1boy, Vash, Trigun followed by the original cow prompt. Of note, while this is not the actual character Vash, he does indeed have an iconic red coat and the art style is impressively similar to the actual Trigun series. Additionally, the ‘yellow eartag’ appears to have morphed into a yellow armband.



This is a previous run where the input prompt was “1boy, Emiya Shirou, fate stay night” again followed by the cow prompt. Hilariously, the cow has morphed into the character’s arm. The major takeaway for me is that this model is really meant to specifically create character portraits/images and should be used as such. I suspect that many other models may be better used for certain roles

Realism Style

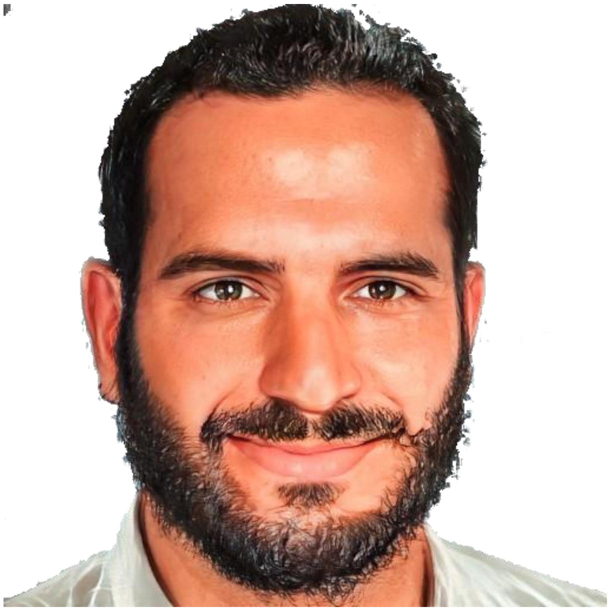


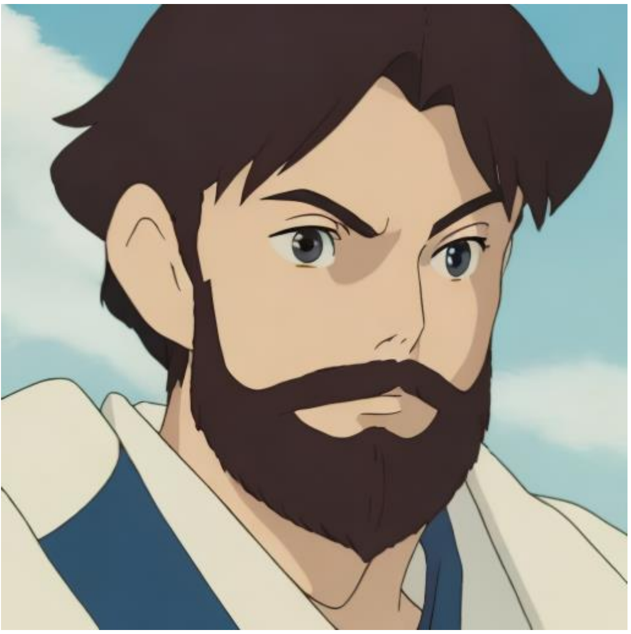
This was a style meant to create realistic images. I suspect that I need to increase the number of inference steps for this image for finer tuning of the detail. I used 30 inference steps in this image. Regardless, I am pleased to see it seemed to be moving in the right direction.

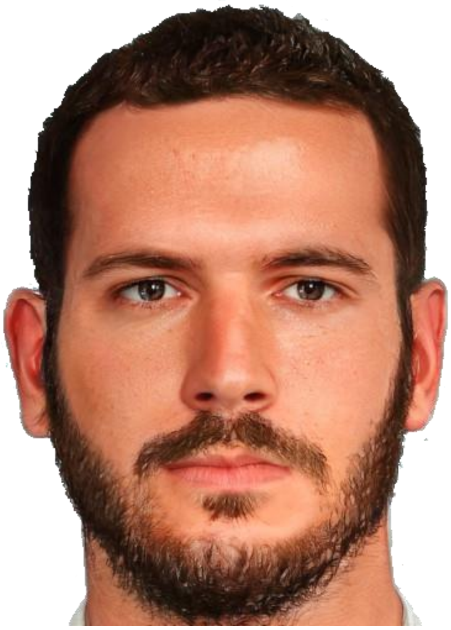
Full Run

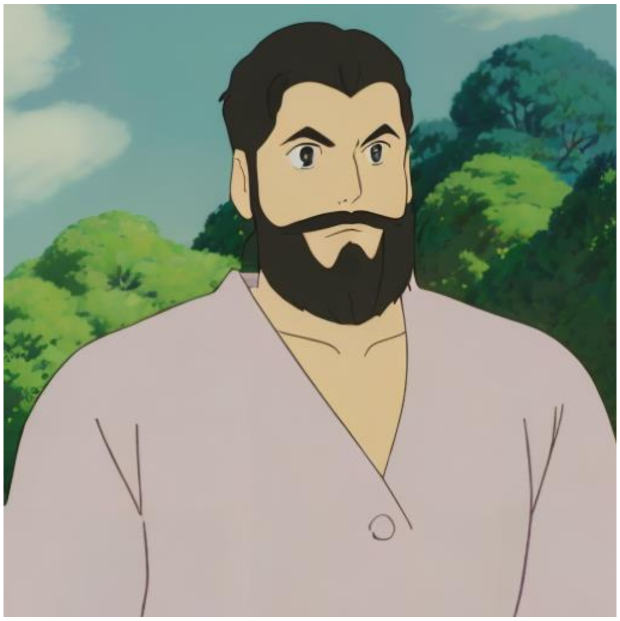
For the full run using the actual game images I used the Ghibli style. I will compare the original images and the generated images side by side.

First a few of the better ones.

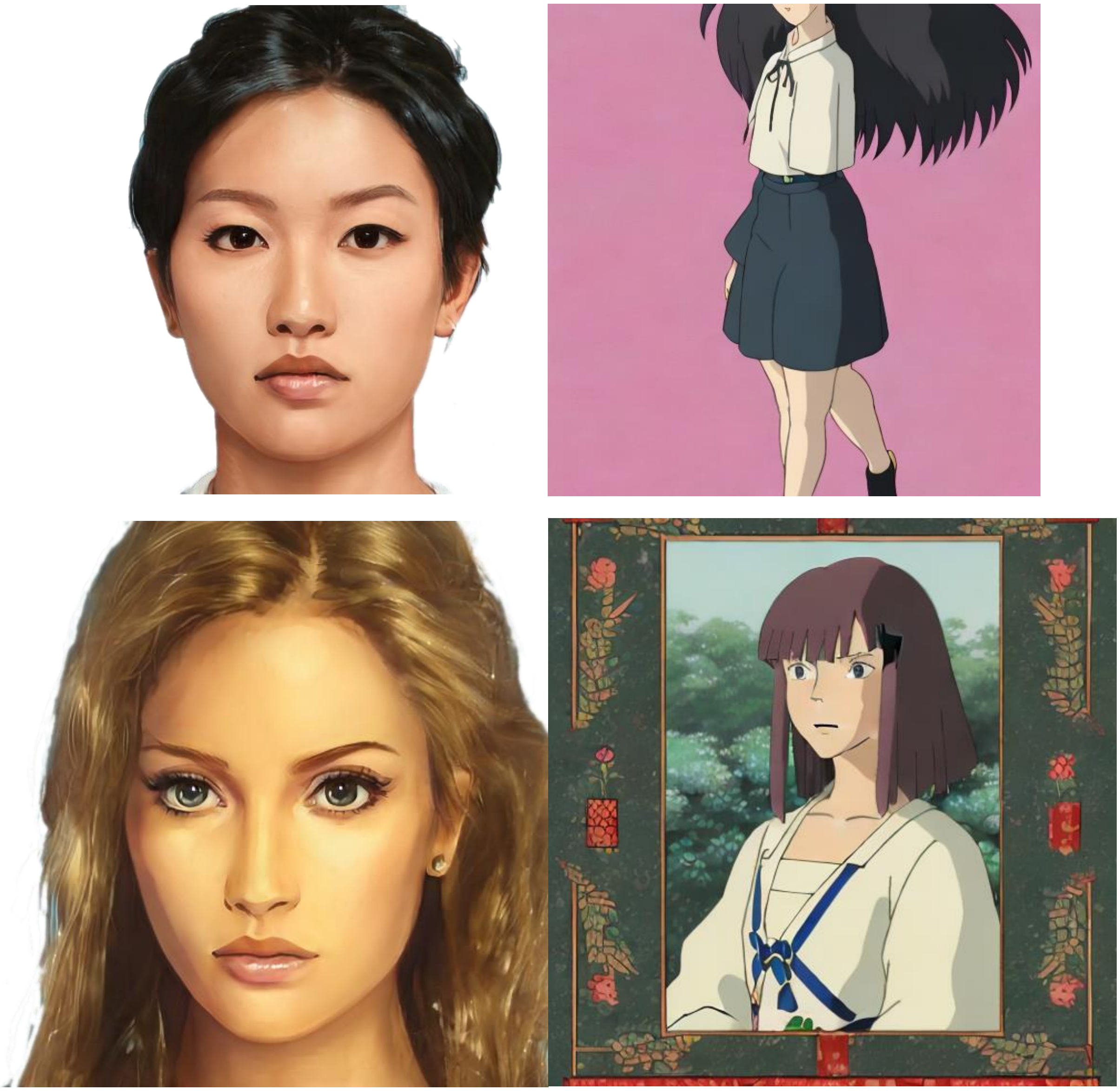






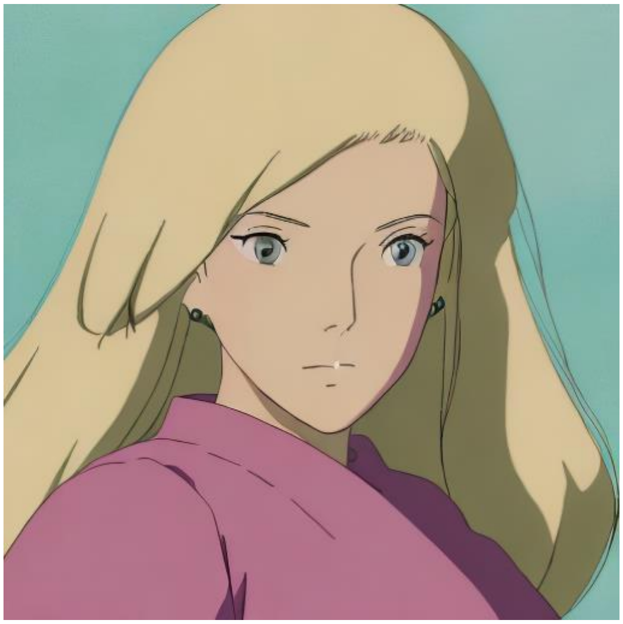


Now some of the worst examples.

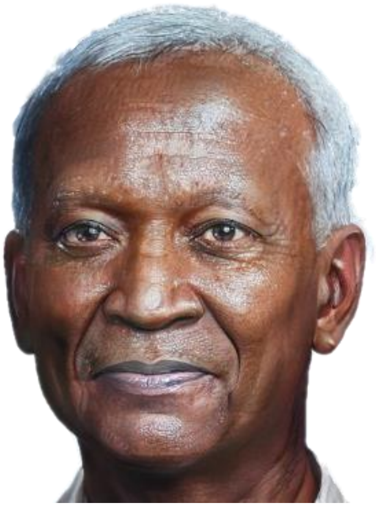


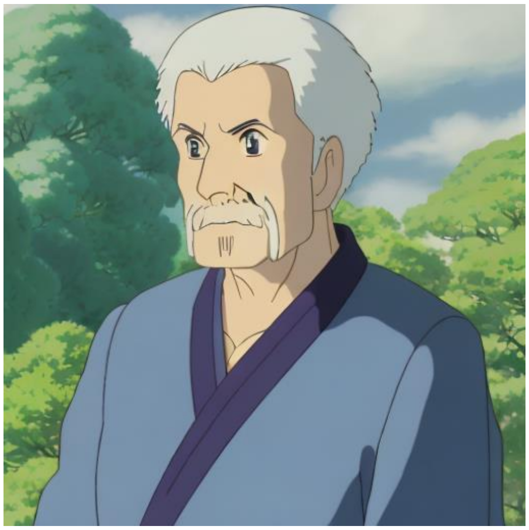
One other thing to note is that occasionally an image will trip the NSFW filter and be returned as a blacked-out image. My satisfaction check in the application is made to combat this, though it would require occasional user input.

We also see several pictures that approach being quite excellent but have random artifacts. For example, in the image below we see a picture of a woman that has a random white spot on her lips.



Of note, the original images contained pictures of people of a variety of different races, whereas the output was exclusively featured Asian and White looking characters. This could be a factor of the fact that Ghibli movies tend to feature Asian and White characters and the prompt generation by BLIP does not seem to mention ethnicity when it generates. For example, the image below generated the text "an older man with white hair and a white shirt." Interestingly, the new image also loses the white shirt. Since no ethnicity is specified, we can essentially expect to get back the image we receive back will be an ethnicity that is a random sampling of ethnicities of characters/people in the training data.





Setting aside the specific matter, we could generalize this idea as a method of testing a model's bias towards any characteristic. In theory when we have a feature not explicitly specified and if we were to run the model enough times, we could approach the ratios for different outcomes given the various choices a feature may have. Sort of like approaching a limit in calculus or approaching a 50/50 split for coins flips given infinite coin flips.

I would like to try generating an image where I explicitly prompt for a generated characters ethnicity to determine if this was a result of the particular model I used as I suspect but was unable to do so before the deadline.

Testing:

Unfortunately, as has been mentioned in class there is no real way to provide testing metrics as the results are ultimately subjective satisfaction with the results. That said, I verified that my application works by running it in my notebook.

Code and instructions to run it:

Provide a link to the code and any required instructions to run it. Please include some testing examples so we can quickly experience what you experienced with the model.

The application can be found at my github here:

https://github.com/General-Cow/AutogenGraphicsMods

The code can be run by either using the image_generator function for a single image or using the generate_graphic_mods function for a set of images in a folder. Be aware, that it will download the model to your cache if it isn't already there, I intend to develop a keep_model variable that could be specified to remove the model after generating the image however it is not implemented at this time as I elected to focus on the generative AI aspect of our assignment and was wary to test a feature that would be deleting things.

My recommendation at this time if you intend to run the model is to use the image_generator function. I recommend leaving autogen_prompt=True, keep_model=True, and predownloaded=False for the first run to download the model and set it to true once you have the model downloaded. Huggingface's connection was extremely unstable for me and cause failures constantly. This forced me to constantly rerun the line until I could connect. I also have a variable processor_type which you can use to set whether you use cuda, cpu, or mps as you prefer. The default is cpu as that's what I used.