

.README Mini Project

Oliver Wood, o.wood0120191@arts.ac.uk, 19019545

Git repo with execution notebooks: <https://github.com/pimentoliver/fungal-futures>

Model results: <https://huggingface.co/pimentooliver/fungi-sd-diffusion>

Final dataset: <https://huggingface.co/datasets/pimentooliver/fungi>

Fungal futures: generating 'new species' of fungi.

One line summary:

This work is an attempt to generate 'new species' of fungi in 3D video and 2D images.

Possible uses:

These fungal objects could then be used in 3D artworks or gamified settings such as a VR fungi forest, or elements in a wider digital artwork concerning AI and ecology.

They can also stand on their own as individual works - potentially, they could be displayed through low-tech hologram projectors, or on transparent screens such as LG OLED.

Methodological exploration:

The initial project premise was to generate 'new species' of fungi.

I initially explored CPPNs - particularly 3D CPPNs. However, the execution of this was somewhat opaque, with no published source code from the original paper (Clune et al, 2013) and available online tutorials and implementations being too computationally expensive or requiring additional, paid for 3D software.

This led me to working with a GAN - I implemented [this PyTorch tutorial to build a DCGAN](#) and started training. During training, while further researching possibilities for 3D output, I stumbled upon [Dreamfusion](#) (Poole et al, 2022). This immediately took my interest - the output quality, creative potential, and convenience of not requiring a 3D dataset seemed ideal for the project.

The text-to-image network used in the original paper, Imagen, is not available to the public. As such, I turned to [Stable Dreamfusion - a PyTorch implementation of Dreamfusion - by ashawkey on GitHub](#). This model uses stable diffusion in place of Imagen.

Optionally, one can use any appropriate model hosted on HuggingFace `(-hf_key arg in main.py)` - so working backwards, I decided to fine-tune a stable diffusion model on my fungi dataset.

During execution of this project, I also looked into [fine-tuning models using LoRA and textual inversion](#). These alternative methods of fine-tuning a stable diffusion model are also valid options. However, textual inversion was not compatible with clustering fungi to create 'new' species - it is possible to rename or insert a pre-existing object, but not to generate an entirely new one. In future, I will explore using LoRA in fine tuning, as it is shown to help prevent catastrophic forgetting.

Execution summary:

The first step was to format my dataset. I decided to cluster the dataset based on feature vectors, hoping for a variety of fungi in each cluster in order to see an output that is truly 'new' in its appearance. I followed [this tutorial](#) by Gabe Flomo, with some minor modifications. The clustered results were aesthetically pleasing and showed some visual similarities, without an overly dominant relationship between visual features (i.e every instance in a cluster being single-color dominant, or single-shape dominant) which was promising.

I then built a simple randomizer to generate species names and text-to-image prompts. Pulling a small dataset of fungi names from [this website](#), I split the data in to 'common' and 'scientific' names. Working with 'common' names only, I removed custom stopwords and built a for-loop to generate randomized, correctly formatted fungi names for each cluster.

I built a similar prompt generator to randomize the words used in the chosen fungi names, and append additional lines such as 'growing in a field'. The additional lines were tested using an initial batch of generated names, and [our base model's hosted inference API](#) on HuggingFace with aesthetically pleasing results achieved.

I opted to randomize the words used because I wanted to see if I could prompt the model to generate subspecies, or mutant species, of the 'new' species it had been trained on.

The dataset was then formatted for use with Stable Diffusion, and HuggingFace's image dataset requirements. I built functions to sort each cluster into a marked folder, and to then generate a metadata.jsonl file using the assigned fungi name as the caption for each image in the cluster. Finally, the dataset was pushed to HuggingFace.

This is where things became difficult - computational expenses began to add up as I repeated the dataset process several times, trying different image sizes (64x64,

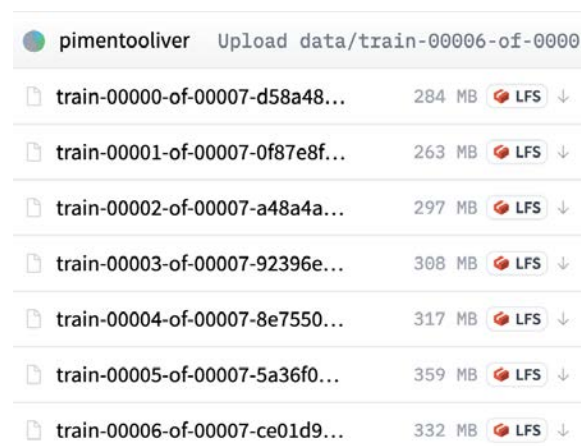
512x512, 256x256, 128x128) to find a balance of best output results and best training time.

A significant period of time was spent attempting to run two scripts: initially, I tried [this HuggingFace tutorial](#), which uses Accelerate. I was hoping Accelerate could help me limit computational costs - however, I spent a long time attempting to debug this script with no results. There appeared to be some opaque issue with running Accelerate inside Google Colab.

I then implemented [this tutorial by Justin Pinkney and Lamda Labs](#), also the original base of the previously attempted HuggingFace script. I implemented this tutorial with several modifications to the original code: removing lines of scripts which were not context relevant, updating parameters and adding additional scripts to format, save, and push model weights to HuggingFace.

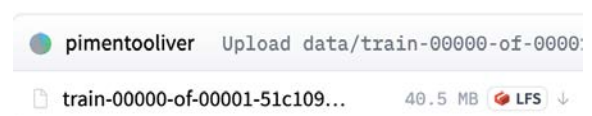
This didn't initially work. It took a significant number of attempts, and several hours debugging my code, to discover the issue: this script only works with one HuggingFace dataset parquet. In other words, my dataset was too big. This issue potentially could have been avoided if I was able to run this code locally and keep my dataset on disc.

Fixing this in script, or running the code locally, was outside the scope of this project and resources available. As such, I significantly modified my dataset - shifting from a resolution of 512x512 to 128x128, removing augmentations and opting to augment in-training using Pytorch Transforms, and removing over 50% of the original images. This allowed the code to run, however it impacted the quality of outputs significantly.



pimentooliver Upload data/train-00006-of-0000		
train-00000-of-00007-d58a48...	284 MB	LFS ↓
train-00001-of-00007-0f87e8f...	263 MB	LFS ↓
train-00002-of-00007-a48a4a...	297 MB	LFS ↓
train-00003-of-00007-92396e...	308 MB	LFS ↓
train-00004-of-00007-8e7550...	317 MB	LFS ↓
train-00005-of-00007-5a36f0...	359 MB	LFS ↓
train-00006-of-00007-ce01d9...	332 MB	LFS ↓

Parquets in original dataset.



pimentooliver Upload data/train-00000-of-0000		
train-00000-of-00001-51c109...	40.5 MB	LFS ↓

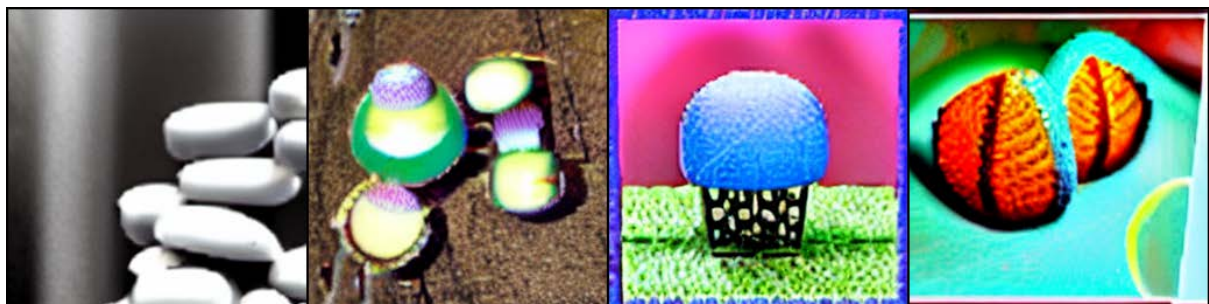
Parquets in final dataset.

I initially trained the model for 22 epochs - a trial run to gage how long training could potentially take and adjust hyperparameters. Using a scaled learning rate, model loss remained between 0.98-1 for >10 epochs. This training cycle was terminated with no results. Result samples from 22 epochs of training are as below: learning rate was not scaled, parameters such as validation prompts and use of Transforms were later adjusted for the next training cycle, as well as batch size and accumulation scales for best use of RAM.

'old bolete mushroom', 22 epochs, 512 x 512



'old bolete mushroom', 22 epochs, 256 x 256



The significant difference in visual features according to requested output size was interesting - where exactly do these differences occur in the model, and why? How could they be utilized to artistic advantage?

Having then updated the parameters, I then trained the model for 100 epochs, which takes approximately 6hrs. I had to do this three times - my Google Colab runtime randomly disconnected twice, taking the model weight files and paid for GPU access with it.

Final results analysis:

'old bolete mushroom', 512



'soapy chanterelle fungus in summer', 512



'berkleys grisette fungi', 512



'honey crab fungus up close'



'swamp agaric fungus', 512



Initially, these results are aesthetically pleasing and overall promising for use with Stable Dreamfusion: the majority of our 'species' have unique, well-defined visual features which could translate well to a 3D model.

However, upon testing more complex prompts, results lacked any significant visual changes. I then shifted to test some control prompts - completely unrelated words to our dataset - and it became apparent that this model has experienced catastrophic forgetting.

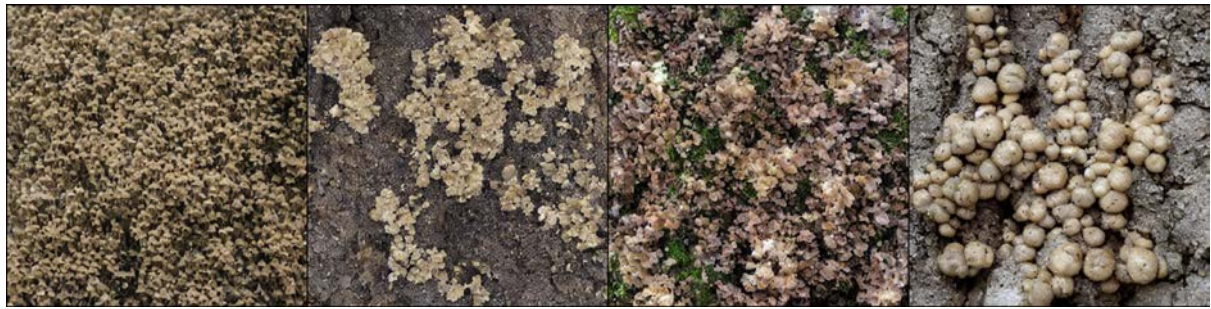
This is common as fine tuning stable diffusion is quite experimental - the risk is significantly lowered by using LoRA, which I will definitely do in future. This also could have been avoided by improving image metadata with additional captions - such as 'in a field', 'large', 'small' etc., rather than only using the fungi name tag.

This means that unfortunately, our engineer prompts will not work, and the model may not work well with Stable Dreamfusion.

'hamburger', 512x512



'girl holding a basket', 512x512



'flower', 512x512



Further exploring our results, let's examine our outputs in 256x256.

'flower', 256



'hamburger', 256



These outputs, while diverse from outputs in 512x512, have very similar visual features, as occurred with our control prompts in 512x512.

Let's explore our original prompts in a smaller output size.

'old bolete mushroom', 256x256



'honey crab fungus', 256x256



'swamp agaric fungus', 256x256



'soapy chanterelle fungus in summer',256x256



'berkleys grisette fungi', 256x256



Overall, these outputs are aesthetically interesting and suggest promising results in future with an execution more conscious of catastrophic forgetting.



40th epoch training output

Across sizes - 512x512 and 256x256 - there is a significant amount of pixelation. This pixelation is not present in early training output images - such as on the left. It has most likely been adapted from our dataset, in which instances are 128x128, resized to 256x256 during training, due to memory constraints.

Regardless of this setback, and due to time and resource constraints, I decided to continue with the project and explore 3D outputs.

I used additional scripts from [Justin Pinkney](#) to prune the model weights of excess information, and push the model to HuggingFace, for use with Stable Dreamfusion.

Shifting to 3D outputs, I implemented a [notebook from ashawkey](#), with some relevant modifications, in order to test the compatibility of the model with Stable Dreamfusion. It is compatible - however, output is unrecognizable as fungi and not aesthetically pleasing. It does not resemble the 2D output previously generated for this prompt.

This is most likely due to the catastrophic forgetting within our model - because we cannot use engineered prompts, and prompts relevant to the dataset return limited results, it is unsurprising that the final result is not as expected.

'honey crab fungus'

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2a3ad5d3-d99b-4f0f-87c1-1a825cb668d9/df_ep0050_rgb.mp4

Final summary and future steps:

In summary, our model experienced catastrophic forgetting and as such, results are below expectation or non-functional for purpose.

This is reparable: in future, I will explore the use of LoRA to lower the risk of catastrophic forgetting. I will also adjust the metadata construction to include clear captions for a diverse range of output.

Ideally, in future, I will access suitable hardware in order to run scripts locally and use datasets of appropriate quality to produce 2D images of higher resolution, in line with industry standard standard stable diffusion outputs.

This project has shown that the base idea is possible - to fine tune a stable diffusion model on fungi, and implement it with Stable Dreamfusion. With appropriate adjustments following this experiment, I hope to continue the work and produce aesthetically pleasing, fit for purpose results.

Citations:

Clune, J., Chen, A. and Lipson, H. (2013). *Upload any object and evolve it: Injecting complex geometric patterns into CPPNS for further evolution*. [online] IEEE Xplore. doi:<https://doi.org/10.1109/CEC.2013.6557986>.

Poole, B., Jain, A., Barron, J.T. and Mildenhall, B. (2022). DreamFusion: Text-to-3D using 2D Diffusion. *arXiv:2209.14988 [cs, stat]*. [online] Available at: <https://arxiv.org/abs/2209.14988> [Accessed 3 Oct. 2022].