Generating 3d Sculptures Using a Recurrent Neural Network with Long Short-Term Memory

Nick Downey
Kenyon College Digital Humanities

Goal

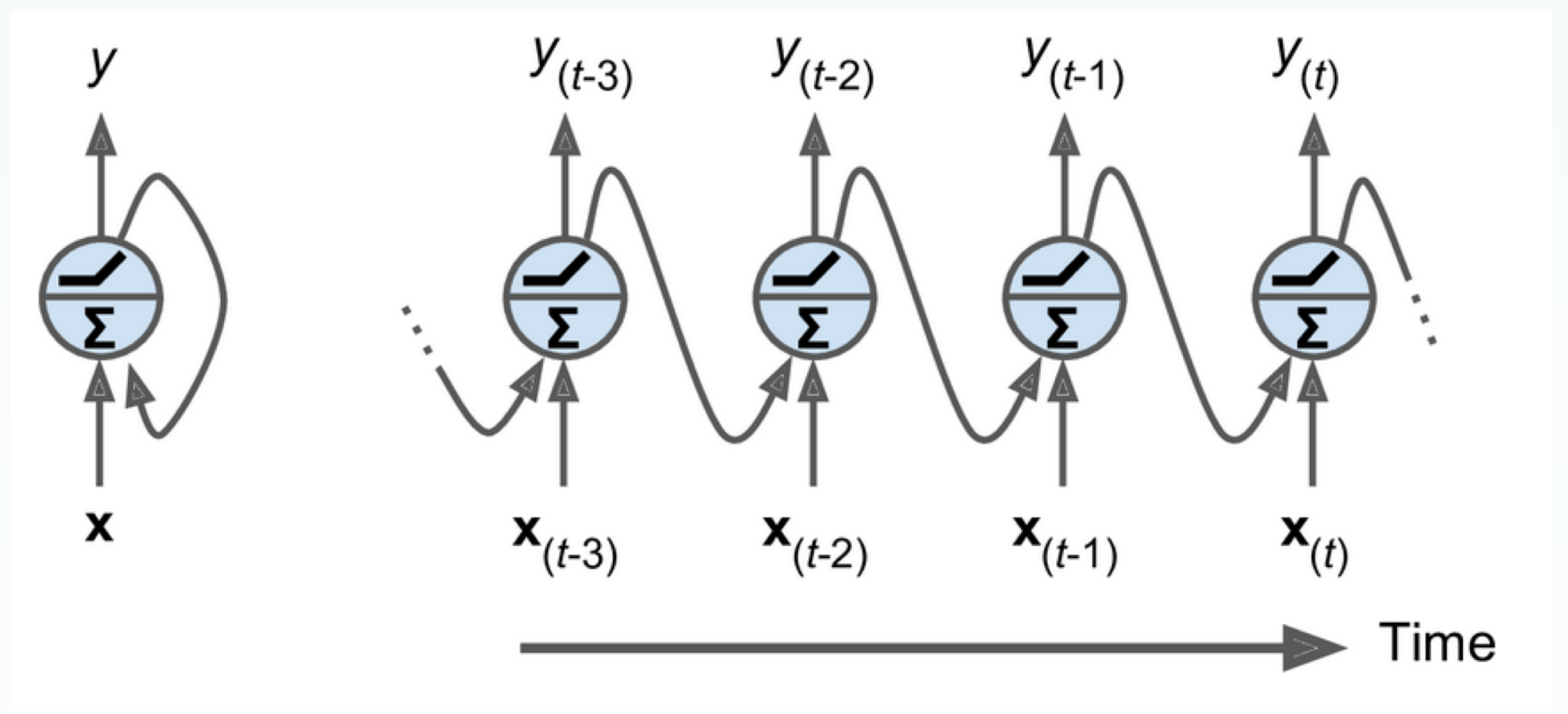
Sophisticated generative models have been some of the most obviously impressive achievements of AI in the past decade. Indeed, AI has approached or achieved human-level performance in a variety of domains, with a particular focus placed on artistic disciplines. Because writing a value function for this kind of model is effectively impossible, these models are notoriously difficult to evaluate. My aim was to create such a model, and in doing so elucidate the characteristics of successful generative AI and explore how generative models can be adapted successfully to new domains. And more broadly, I hoped to create a model capable of producing good sculpture, although I have no intention of evaluating what it means for a sculpture to be good.

Dataset

This project utilizes the Thingi10K dataset. This dataset consists of 10,000 3d models sourced from Thingiverse, an open forum for 3d modeling and printing. Thingi10K features a wide range of content, including tools, mechanical pieces, art, and other structures. The variety of its content was the trait that made it the most desirable, as I sought to have as little input as to the content of the model's output as possible. After converting the dataset to .obj files and concatenating them as plain text, it totaled just over 25 gb. While this dataset was too large to train on in its entirety with the resources I had access too, the size of the dataset provides clear room for growth of the model with more powerful training resources.

Model Architecture

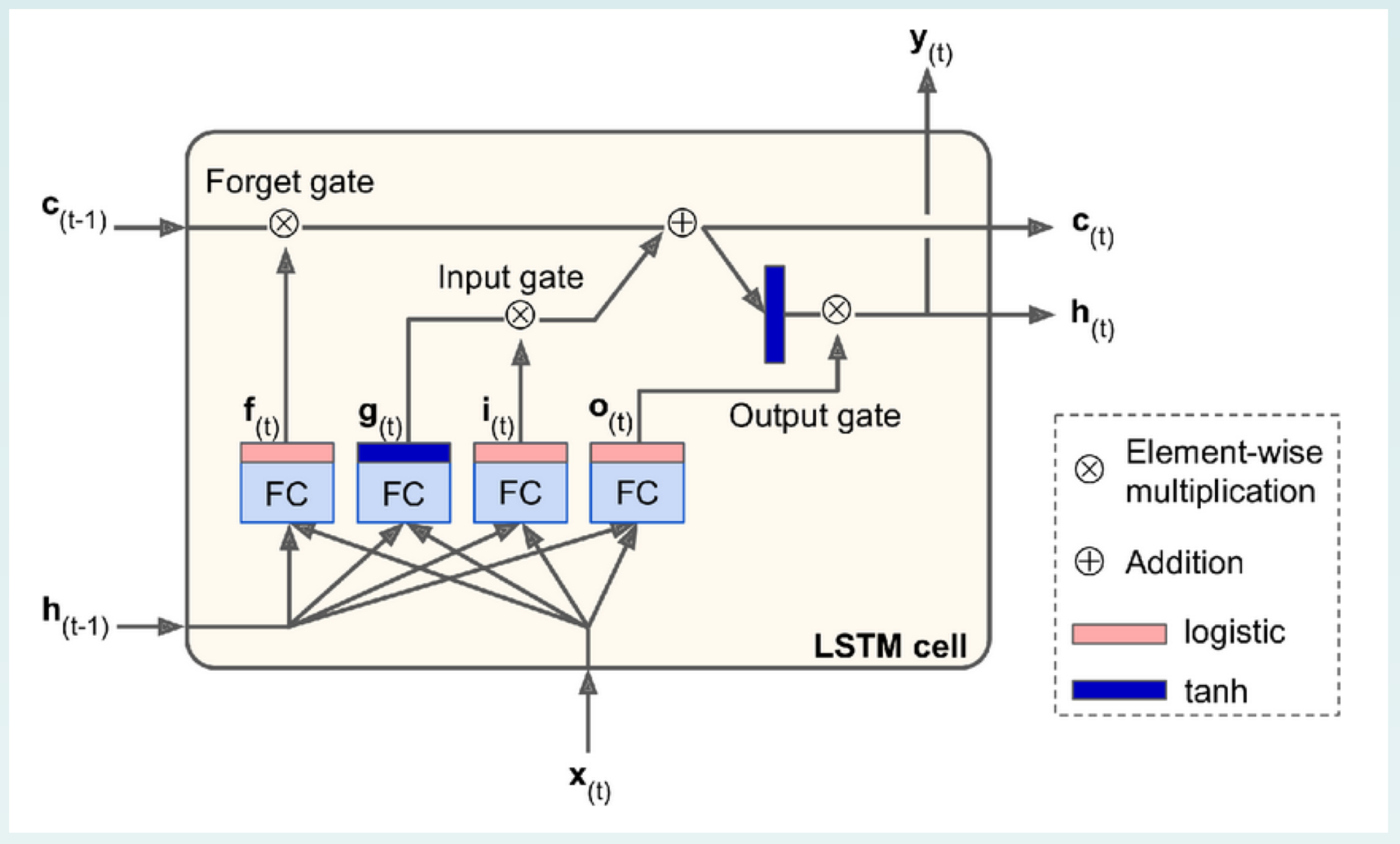
Recurrent neural networks (RNN) are, unlike more traditional dense neural networks, informed in part by the output of previous timesteps, they are particularly adept at handling sequence data. As a result, RNN's have become increasingly popular for the processing and generation of text, whose contextual nature has long been a challenge for rule-based systems as well as less sophisticated neural networks. By converting the Thingi10K dataset into .obj files (which are plain text representations of 3d objects), a RNN optimized for text generation should be capable of generating 3d structures



A singular RNN node represented in time

While RNN's are a powerful tool for handling sequence data, their key insight is really just that previous output and current output are somehow connected. Their drawbacks are readily apparent when they're tasked with handling large bodies of text in which and element's relevance may exceed the scope of an RNN's timestep. For this reason, I chose to utilize long short-term memory (LSTM) cells in conjunction with traditional recurrent cells. LSTM cells weight input from previous time steps by their relative importance, then perform element-wise multiplication upon the cell's output. The hope of LSTM is that it acts both as an attention mechanism, as well as a more sophisticated model of context. A well-trained LSTM cell should be able to determine both whether a previous piece of information is relevant to current output, as well as the magnitude of its importance.

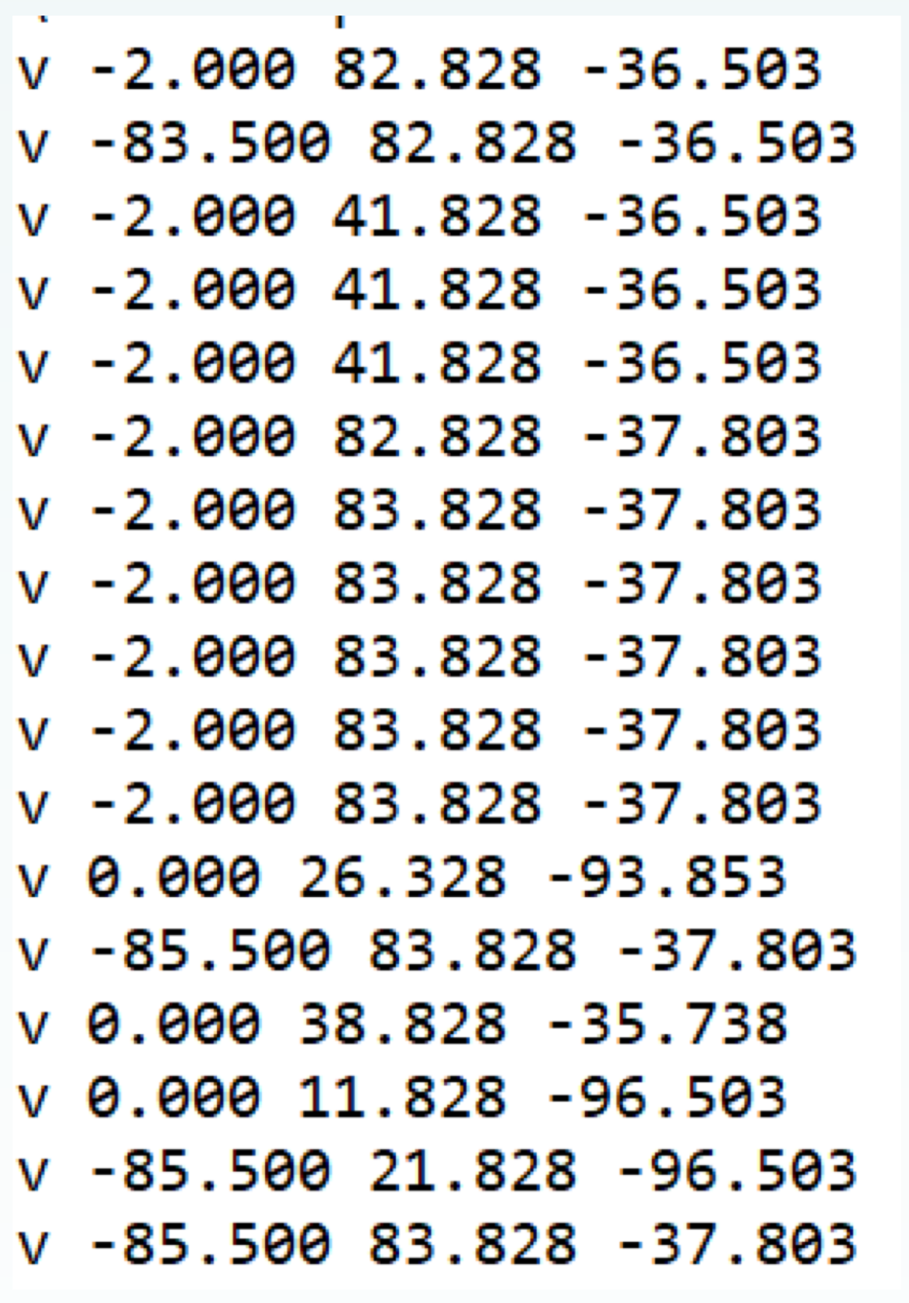
Model Architecture cont.



A single LSTM cell

Results

By far the biggest drawback of generating 3d Objects using a text generator rather than a more specialized model is the fragility of the obj format. Not all text is a valid object, inf fact, most is not. A model of this architecture cannot guarantee that its output will be a manifold object at all, let alone that it can generate them with reasonable consistency. Indeed, even the most successful iterations of the model were unable to generate manifold objects with even 50% consistency. Notably, this variety of RNN will generate as much text as it asks for, so it's non trivial to measure the model's true hit rate.



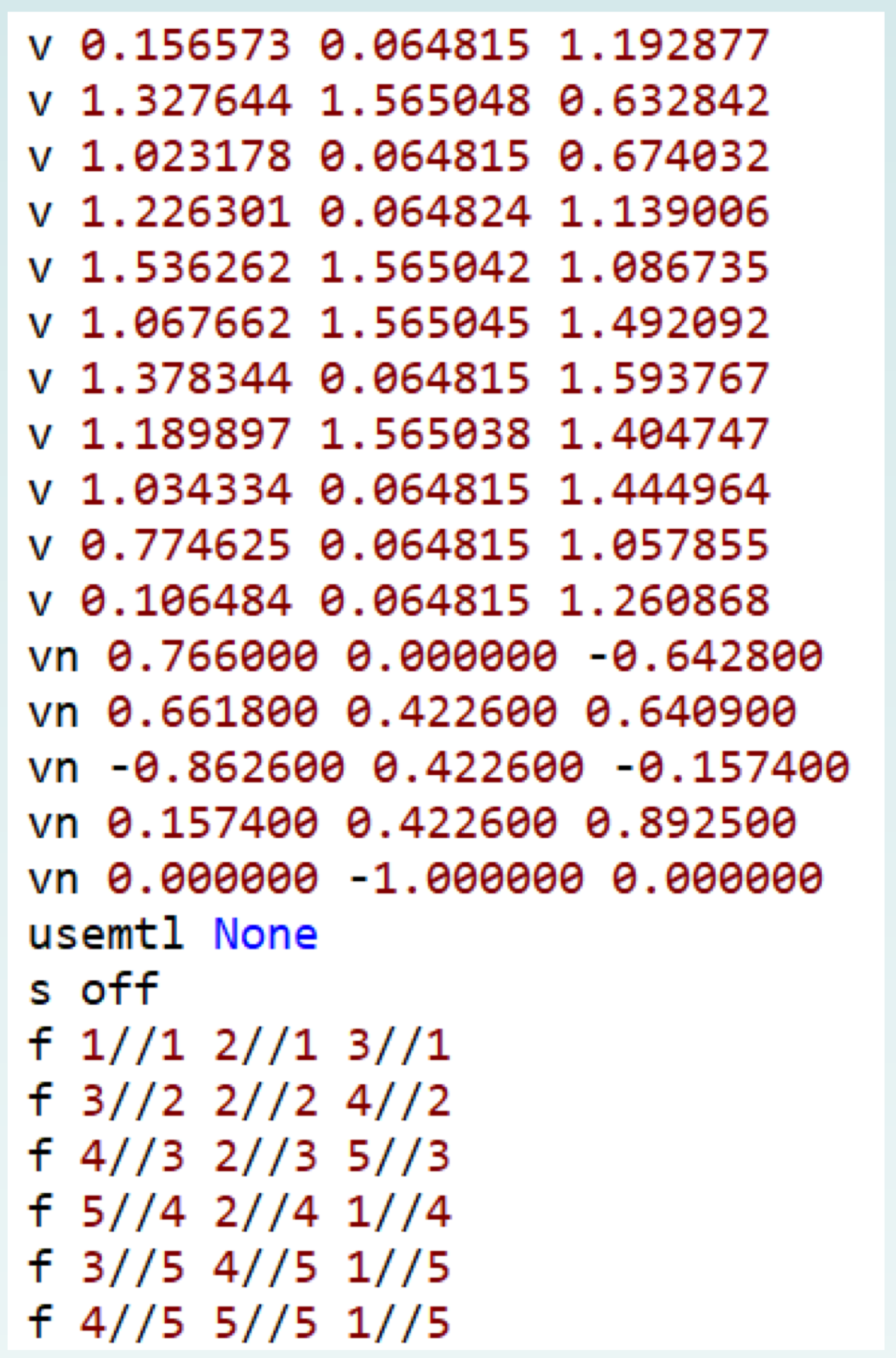
```
v -2.000 82.828 -36.503
v -83.500 82.828 -36.503
v -2.000 41.828 -36.503
v -2.000 41.828 -36.503
v -2.000 41.828 -36.503
v -2.000 82.828 -37.803
v -2.000 83.828 -37.803
v -2.000 83.828 -37.803
v -2.000 83.828 -37.803
v -2.000 83.828 -37.803
v -2.000 83.828 -37.803
v 0.000 26.328 -93.853
v -85.500 83.828 -37.803
v 0.000 38.828 -35.738
v 0.000 11.828 -96.503
v -85.500 21.828 -96.503
v -85.500 83.828 -37.803
```

Sample model output after 20,000 samples

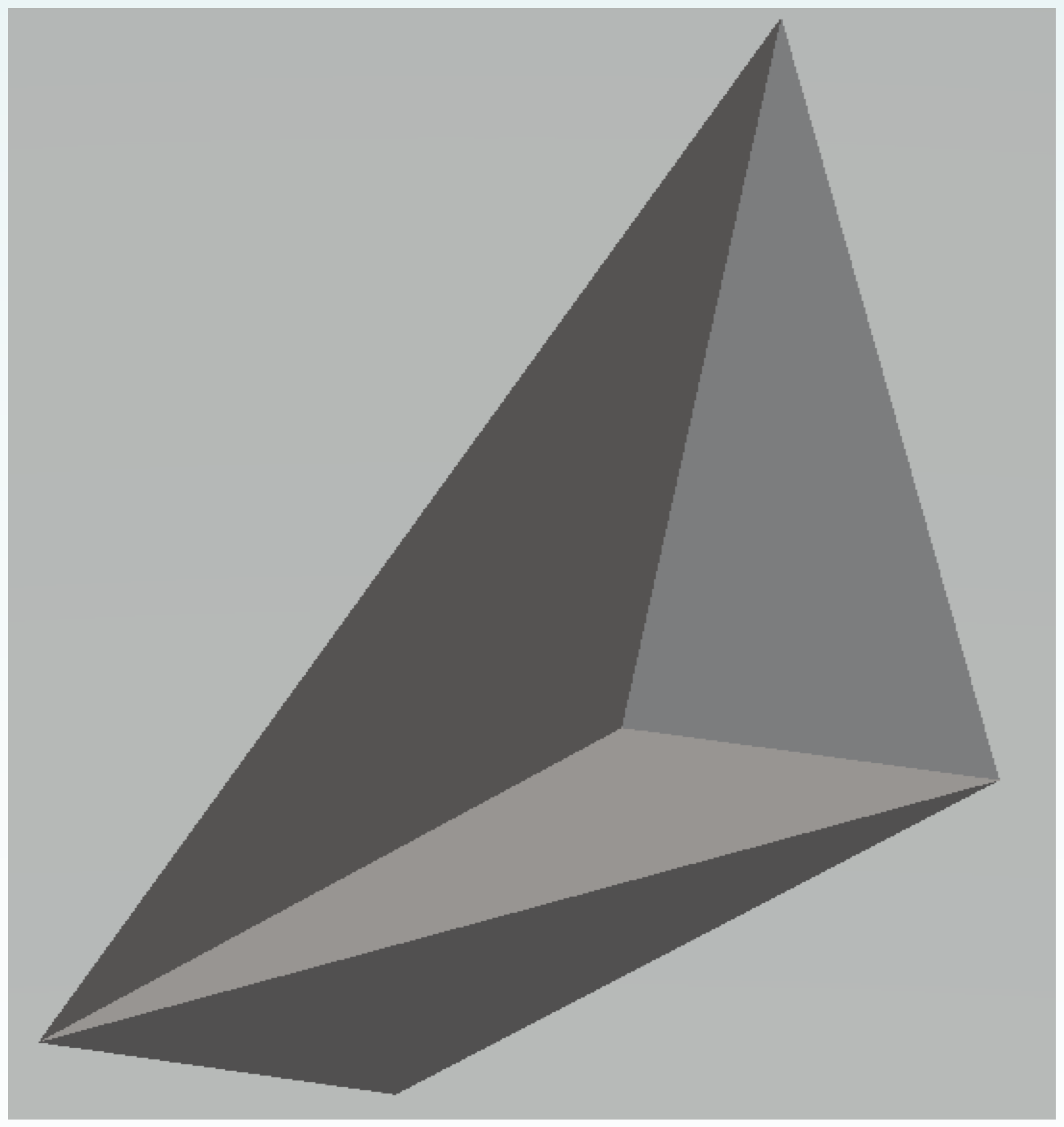
Above is sample output from the model which is not manifold. Each line beginning with 'v' gives the x, y, and z coordinates of a vertex. Given that this output is simply a list of vertices, it's obviously not a valid object. However, our output does look remarkably like valid obj form, a good indication for future models. WIth a dataset of just over 1 Mb of a potential 25 Gb, and just 20,000 samples, our models shows huge room for improvement

Results cont.

Pictured is sample output after 200,000 samples. This is the first manifold object generated by the model, which at this point was generating manifold objects roughly 1/20 times. While simple and relatively uninteresting, its ability to be generated at all is an important benchmark for our model



```
v 0.156573 0.064815 1.192877
v 1.327644 1.565048 0.632842
v 1.023178 0.064815 0.674032
v 1.226301 0.064824 1.139006
v 1.536262 1.565042 1.086735
v 1.067662 1.565045 1.492092
v 1.378344 0.064815 1.593767
v 1.189897 1.565038 1.404747
v 1.034334 0.064815 1.444964
v 0.774625 0.064815 1.057855
v 0.106484 0.064815 1.260868
vn 0.766000 0.000000 -0.642800
vn 0.661800 0.422600 0.640900
vn -0.862600 0.422600 -0.157400
vn 0.157400 0.422600 0.892500
vn 0.000000 -1.000000 0.000000
usemtl None
s off
f 1//1 2//1 3//1
f 3//2 2//2 4//2
f 4//3 2//3 5//3
f 5//4 2//4 1//4
f 3//5 4//5 1//5
f 4//5 5//5 1//5
```



The object rendered in 3D space

Conclusion

Given the limit on resources, I'm happy that this model was able to generate manifold objects at all. Indeed, with greater computational resources, this same model training on a far larger dataset would presumably be able to generate objects for more interesting in appearance. However, the drawbacks of this model have made themselves readily available. If we really want a model which produces manifold objects with a 100% hit rate, then a matrix representation makes far more sense, although datasets would be harder to come by, and training would be several orders of magnitude more expensive.