**GANimals: A Study in Art using Artificial Intelligence**

*CS 455 – Spring 2022*

*Vivian Dang, Daniel Khalil, Luke Crump*

**INTRODUCTION**

The challenge for this project was that there was a need to create monstrous animal faces based on a mixture of other animal faces. Doing this traditionally, either by hand-drawing or photoshop, proved to be too time consuming, so we decided to utilize Artificial Intelligence (specifically GANs) to do it for us. This is how GANimals was conceived.

By utilizing Generative Adversarial Networks, otherwise known as GANs, we were able to seamlessly mix a bunch of land-dwelling mammal faces to produce a new, never-before seen creature of horrific appearance.

A visual dataset, provided by Kaggle, comprising of 16,130 high quality images was the basis for which we blended to create our outputs. All of them were capped at 512 x 512 resolution (since uniformity greatly improves the ability of Generative Adversarial Networks) and split into three categories: cats, dogs, and wildlife (which included a range of animals varying from foxes to tigers to lions; though strangely enough, there were no bear faces). Due to these limitations, the project was limited to creating only land-dwelling mammal-mixtures.

Though there was barely any useful application for this product besides it being a creative project, we aimed to have fun and achieve our goal of animal face creation, which we deemed a success.

**APPROACH**

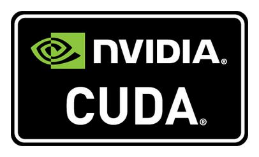
***Data Analysis***

There was honestly not much analysis that was done for this project. We had an idea and had already decided that we were going to work with GANs. The only thing we were looking for was a suitable dataset, which was easily provided by a Google search.

***Resources Used***

Graphical user interface, application, Teams

Description automatically generatedAs mentioned before, this project utilized a visual dataset comprised of 16,130 high quality images at 512 x 512 resolution. The dataset was resourced from Kaggle, an online community of data scientists and machine learning practitioners that is licensed under Google. It was split into three folders: “Cats”, “Dogs”, and “Wildlife”, with each folder including roughly one-third of the photos.

We also used CUDA, which is a parallel computing platform and application programming interface that allows software to use certain types of graphics processing units for general purpose processing. Though this was strictly necessary for our project to run, it *was* needed if we wanted our project to run *fast*.

***Software Design***

We only had one source code file for this project: main.py. Pictured is our general code structure. It contains two models: a Generator Neural Network and a Discriminator Binary Classifier that is based on a Convolutional Neural Network. To start out, the Generator NN is given a random input vector, which will act as a kind of seed for a picture. To update weights on the generator it generates a loss function via tricking the Discriminator NN and receiving a score it as well.

Diagram

Description automatically generatedThe Discriminator NN’s role, however, is to try to classify real versus generated (“fake”) images. To do this the Discriminator NN labels the dataset of faces as real and the generated images as fake. These two models train against each other, with the Generator NN generating fake images to trick the Discriminator NN, and the Discriminator NN classifying fake images versus real images. We run this for fifty epochs and save images per epoch.

**RESULTS**

Due to the nature of this project, the definition of “success” is a bit fluid. Perhaps due to the images being scaled down so much (64 x 64 vs 512 x 512), the generated outputs were a bit more granulated than our liking. If we had had more time on this project, we would have liked to figure out a way to optimize our code to output images at the original 512 x 512 resolution, or at least at a better resolution than what was produced.

Overall, I would say that this project was a success. We were able to successfully train our own Generative Adversarial Network to produce what we wanted, albeit at a bit of a quality loss. Given our time constraints and general workload from our other courses, we managed to output some semblance of a creature. Next time, it would be cool to see what would occur if we had limited certain training sessions to only one category of animal. I am most interested in seeing what would happen if we had only used the “Wildlife” category, since these animals are usually seen as more predatory than your average domesticated dog/cat.

**CONCLUSION**

I would say that to the best of our abilities, that these results were overall considered a success. Technical lessons that were learned include the implementation of Generative Adversarial Networks. We as a team had no idea just how labor intensive for our hardware training and outputting our epochs would be. We went through multiple hardware changes and tweaks before we were able to somewhat reasonably generate the data we wanted to see.

As mentioned before, if we were to continue this project, we would be attempting to generate images with much higher resolutions than we did already. 512 x 512 resolution being scaled down to 64 x 64 resolution did not do us any favors. We would also like to add the option of generating images based on specific categories only, or be able to adjust which categories have more weight in the outputs.

**REFERENCES**

**APPENDIX**

***Daniel Khalil***

The Generative Adversarial Networks for Animal Face generation was a successful project. I am proud of all my teammates and all that we have accomplished. I learned a lot about how GANS work. It is interesting how it uses two different networks to compete against each other which in-turn makes the generation of images more appealing to the problem space. I worked on creating two different networks that played off each other called the generator and the discriminator. The generator was a model that would generate images based off of a random vector seed (that I programmed). And the discriminator was a binary classifier that would determine whether images where real or fake. To do this I created a Convolutional Neural network. I learned about convolutional neural networks in the class CS455. This process was terrible to run on our CPUs, so when me and Luke and Vivian were programming this on our laptops, it took way too much time to compute 1% of 1 epoch. This forced me to learn how to utilize my GPU using CUDA. CUDA allowed us to (relatively) quickly compute the animal faces over 50 epochs.

My role in this project was to create the initial draft of the code and get a semi working representation of our project. Luke worked with me to debug and reformat my mistakes and add some functionality. Vivian is a god tier PowerPoint and document writer, so she did all the documentation and logistics of the project. All together we compensate for each others flaws in very helpful ways, Luke is neater at programming, I’m fast and can apply functionality very quickly, and Vivian is the only one on our team who can read and write in English.

***Luke Crump***

I learned that Generative Adversarial Networks are very computer intensive. Going into the project I did not expect them to take much time to train, maybe an hour at most. The GANs took eight minutes to train the first epoch. When we were doing the project all our laptops stopped responding so we switched to our desktops. My desktop was still super slow, so Daniel used his desktop to run all the testing. GANs were also a lot more interesting than I originally thought. GANs uses two different neural networks to create better images. The first neural network is a generator which creates the images. Its goal is to create real images that can trick the discriminator neural network. The discriminator neural network is supposed to figure out which pictures are fake. So, every epoch the first neural network will create a set of images and the second neural network will go through and pick which ones are fake. They do this over and over until the algorithm is stopped. This form of training is very affective because as soon as one gets better so does the other network. This creates a form of constant improvement since the two algorithms are continuously learning.

My role in this project was to assist in coding and debugging. I also reformatted the code to be more readable. Our team met over the weekend at the beginning of April to finish the project. Me and Daniel did all the coding while Vivian focused on the documentation and the presentation. This created a good dynamic because while we were coding, we were also finishing the report part of the project.

***Vivian Dang***

***Source Code***