MATH299W Final Project

Arjun Vedantham

1 Introduction

In this project, I applied the DeepDream machine learning algorithm to produce new music.

2 Background

DeepDream is a machine learning algorithm released by Google in 2015. It consists of a convolutional neural network, where the backpropagation is applied to the inputs, rather than the weight vectors.

2.1 Math Behind Neural Networks

Neural networks consist of weight and bias vectors, used in conjunction with activation functions. In the beginning, an vectorized input is multiplied against a series of weight vectors of the same size. The algorithm then optionally adds a bias term, and feeds the resulting value forward to an activation function. This is an arbitrary function that will output a value for the next sequence of weight and bias vectors, which is called a ”layer”. This process repeats until we get to the final layer of the network, where the resulting value represents the output of the entire network.

For example, a neural network may accept a 28×28 matrix where each entry corresponds to a brightness value in a monochromatic image of a hand-written digit. A neural network would vectorize this input, turning it into a 784-dimensional vector, and then use a sequence of weights, biases, and activation functions to turn this vector into a 10 dimensional vector that represents where the only nonzero entry correponds to the class of that image (e.g. the output class would be 7 for an image that has the number ”7” in it).

Convolutional neural networks are similar, except that they also use pooling to reduce the dimensionality of the image as it goes through the network. This allows it to pick up on distinct features in the image (for example, the top bar on the ”7”).

2.2 Backpropagation

Neural networks are ”trained” using backpropagation, where the output of the network for one training example is compared against the ground truth, and a measure of how incorrect the network is (the ”loss”) is used to update the weight and bias vectors throughout the network. These weight and bias vectors are adjusted by finding the gradient of the loss with respect to each weight or bias term, using the chain rule from calculus.

Normally, backpropagation applies these to the weight and bias vectors with the goal of reducing the loss value, but it doesn't have to - instead, we can apply these corrections to the input vector and aim to increase the loss, thereby enhancing any latent patterns in the image. This is what DeepDream does, and the result are these dreamy, surreal images. We can apply the principles here to any vectorized input, including audio.

3 Applying DeepDream to audio

DeepDream uses the weights from the ImageNet model, which expects a $412 \times 500 \times 3$ matrix, where the 412×500 represents the length and width of the image, and the 3 represents the three channel nature of the image (for the amount of red, green, and blue).

However, audio files are represented as a $2 \times (sampleRate * lengthSeconds)$ matrix, where the sample rate for represents the number of times the audio signal was sampled within a second. For this project, all of the audio files were sampled at 48000 Hz. To convert this into an input compatible for ImageNet, I took three 103000 length samples from the right and left channels of the stereo audio signal and reshaped them to represent the three channels of an RGB image with size 412×500 pixels.

When visualized, this is what the audio looked like:

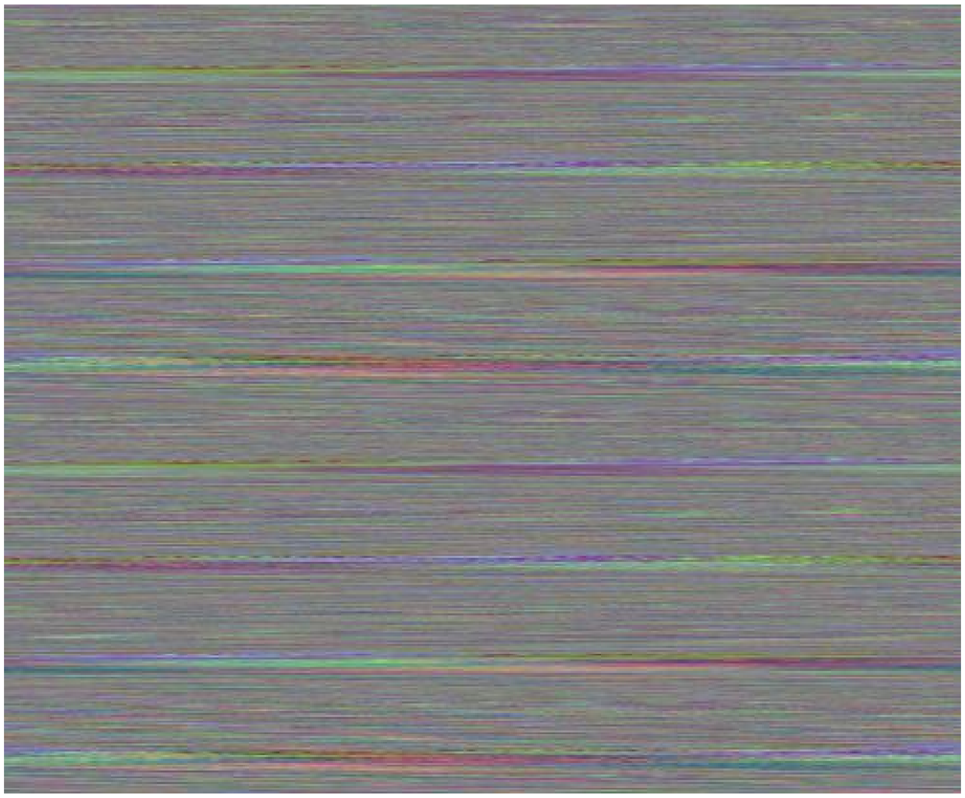

Figure 1: The first 9 seconds of Rickroll, visualized as an RGB image

After using DeepDream, I converted the audio from floating point numbers into 16 bit integers (the standard form for how audio data is represented) and wrote it out to a file. Finally, I performed a little post-processing to reduce the volume of the audio.

4 Result

The result is audio with these loud, surreal, repetitive sounds, with some recognizable features from the input audio. The output audio is best when a listener uses good headphones alone, since it will be more difficult to pick out the recognizable sounds from a noisier speaker output. In the future, I'd like to use a custom neural network that preserves the bit depth of the input signal, since much of the audio detail is being washed out in the conversion between floating point numbers and integers.