Evolutionary Patterns In Neural Networks Using Gradient Descent

Author, Amandin Chyba Rabeendran

Supervisor, Terry Bridgman

Objective

To intuitively visualize the evolution of a multi-layered perceptron using the gradient descent algorithm. Understanding how the perceptron/neural network identifies patterns in data is crucial towards improving the core model. The network in question operates in 400 dimensions which is hopelessly impossible for humans to visualize or think in without computer assistance. This project will provide meaningful "artworks" that could help bridge this disconnect between man and machine by helping people visualize greater dimensions through color, shapes, and other details.

Background

Simply put, a perceptron takes in several inputs (x_i), in this case 400 inputs, that range from 0 to 1 and multiplies each input by a weight (w_i) before summing all of the values together. This sum of dot products is known as the activation value of the percepton and is then passed through a function that returns a number between 0 and 1 for any real number (e.g. sigmoid function). The resulting number is the output/guess that the perceptron has generated for said inputs.

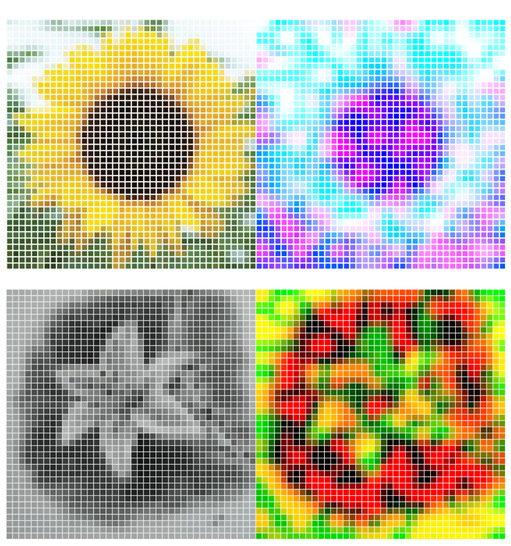
$$\alpha(\sum_{i=1}^{400} w_i \cdot x_i) = output$$

The real magic happens to lie with the weights and the cost function ($cost = (actual - output)^2$). The neural network can learn/improve its output accuracy by taking the derivative of its error in terms of each individual weight. In this fashion, the network can change each weight so that the output is closer to the desired result. This process of taking partial derivatives of the error and slowly adjusting the weights to be more accurate is known as the gradient descent algorithm.

$$w_i = w_i - \frac{\partial cost}{\partial w_i}$$

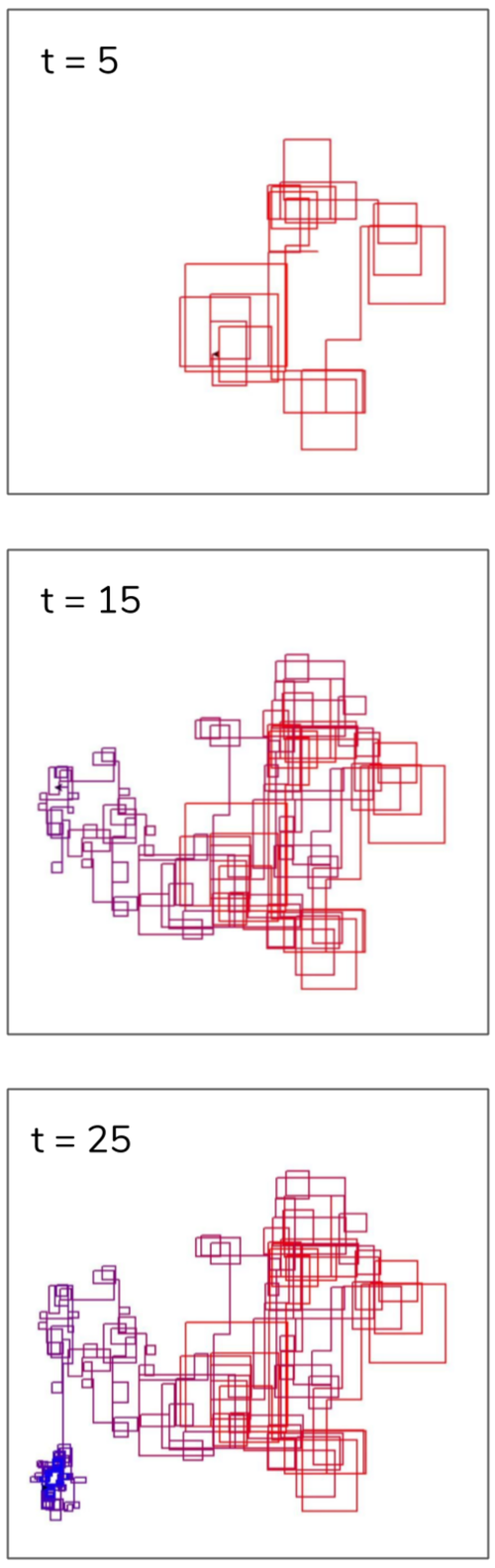
Art

The neural network I made/used was trained to differentiate between sunflowers and lilies. Below are 2 examples where the picture to the left represents the original image and the picture to the right is a generated image based on the networks understanding of it. Each pixel on the generated image is associated with r, g, b values where the red is determined by how important the network thinks that specific pixel is. The green is determined by how bright the pixel was in the original image and the blue is whether it thinks the image is a sunflower or not. Through these images the general shapes are conserved but we can see which pixels the computer cares about the most in determining the type of flower (orange for lilies and white for sunflowers).



The previous images seem chaotic but you might be surprised to know that this network has an accuracy of above 85%. The network was not always this smart and in fact started off pretty random. The network begins very inaccurate because the initial weights are all randomly generated. Therefore, the network has no idea on the visual cues that differentiate a sunflower from a lilie. However, as the network is fed more images and trains itself, the outputs start to look less random as they converge towards some equilibrium.

The images below shows a line that moves forward based on how wrong the network is about an image being a sunflower/lilie. After every image, the line will make a 90 degree turn and repeat. Since the network should improve as we feed it more images, we expect the lines to get smaller as it converges to an equilibrium. The variable $t = 1$ represents the processing of 26 images and the color of the line slowly converts from red to blue after every image to map its progress.



References

3Blue1Brown. "But what is a Neural Network? | Deep learning, chapter 1". October. 5, 2017. Accessed on: April. 2, 2020. Available: https://www.youtube.com/watch?v=aircAruvnKk