GPNN Memory

Theory: A neural network uses a one dimensional gradient array of three dimensional color vectors to store and fetch information. Having each information be assigned to a specific color vector. It then can separate it in 2 thresholds specifying 2 regions (warm region and cold region).

Practical Example:

Let's assume a Neural Network named “**X**”

Input: 3D Color Vector [e.g: (1.0, 0.5, 0.62)]
Output: 3D Color Vector [e.g: (1.0, 0.65, 1.0)]
Access Memory (AM): 1D Gradient (Blank at start)
Comparable Memory (CM) : 1D Gradient (Color Spectrum)
Example Input Data: {
orange: (1.0, 1.0, 0.0),
rottenOrange: (0.0, 0.0, 1.0)
}

Input: orange **OR** rottenOrange
Logged Output: accessMemory: [(1.0, 1.0, 0.0), (0.0, 0.0, 0.0), …] **OR** [(0.0, 0.0, 1.0), (0.0, 0.0, 0.0), …]
Practical Output: May Vary

Logic: On first start it'll have it's **Access Memory** Gradient be blank and have an extra **Comparable Memory** Gradient which it'll compare it's input with.
For each input it'll compare it with **CM** and append it in **AM** with the same or near index of matched color in **CM**.
This **AM** Gradient can now be used to execute specific tasks. For example the model could avoid any item with a color value near the colder region of **CM** (like **rottenOrange**). It can learn and decide which object is more likely to be selected or what action is more logical to take based on the color vector assigned to the input variable. The logic is totally on the developer to select what would the model do.