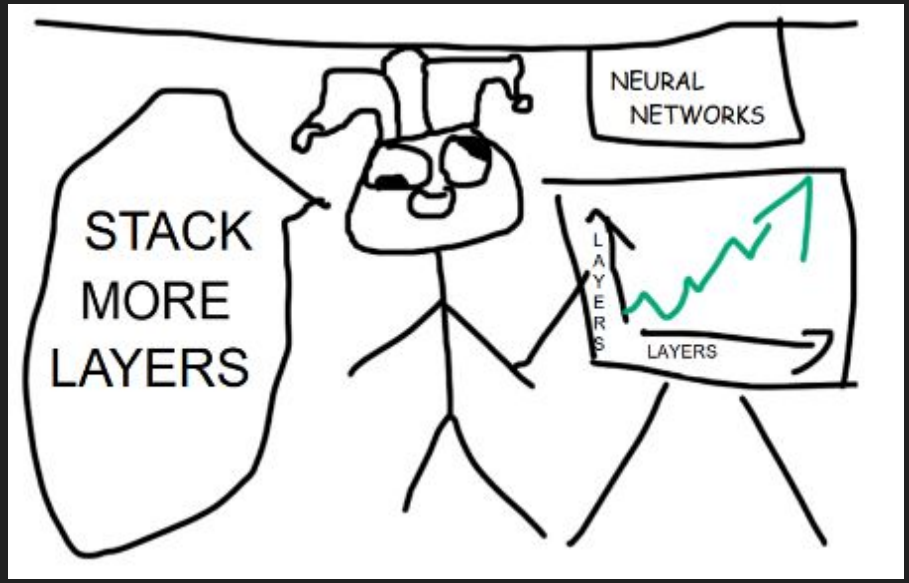
Motivation

- Gradient Flow issues with Deep Neural Networks.
- Neural Networks often get stuck up at local optimas.
- Neural Network Training requires a lot of memory to speed up Backprop by storing activations.
- Current Multi-GPU Training strategies are ineffective with considerable amount of synchronization.
- Needs Differentiable Activation Functions.

What We have been Doing !



The Local Optima Issue

Solution

- Replacing Gradient Descent with Particle Swarm Optimization
- Coupling Gradient Descent with Particle Swarm Optimization(Hybrid Approach)

Particle Swarm Optimization.

- A swarm based artificial intelligence algorithm consisting of a number of particles exploring a multi-dimensional space to optimize a function.
- Each particle has its own position and velocity.
- Both position and velocity are randomly initialized.
- Position is updated using velocity.
- Velocity updates take place depending upon the fitness of the position

Positions in PSO

- Particle Best : Best known position for each particle
- Local Best : Best Known position between the particle and its immediate neighbours.
- Global Best : Best known position among all the particles.
- All variants of PSO use Particle best.
- Global Best & Local Best are used by the basic variant & local best variant of the PSO.

Local Best Variant vs Basic Variant

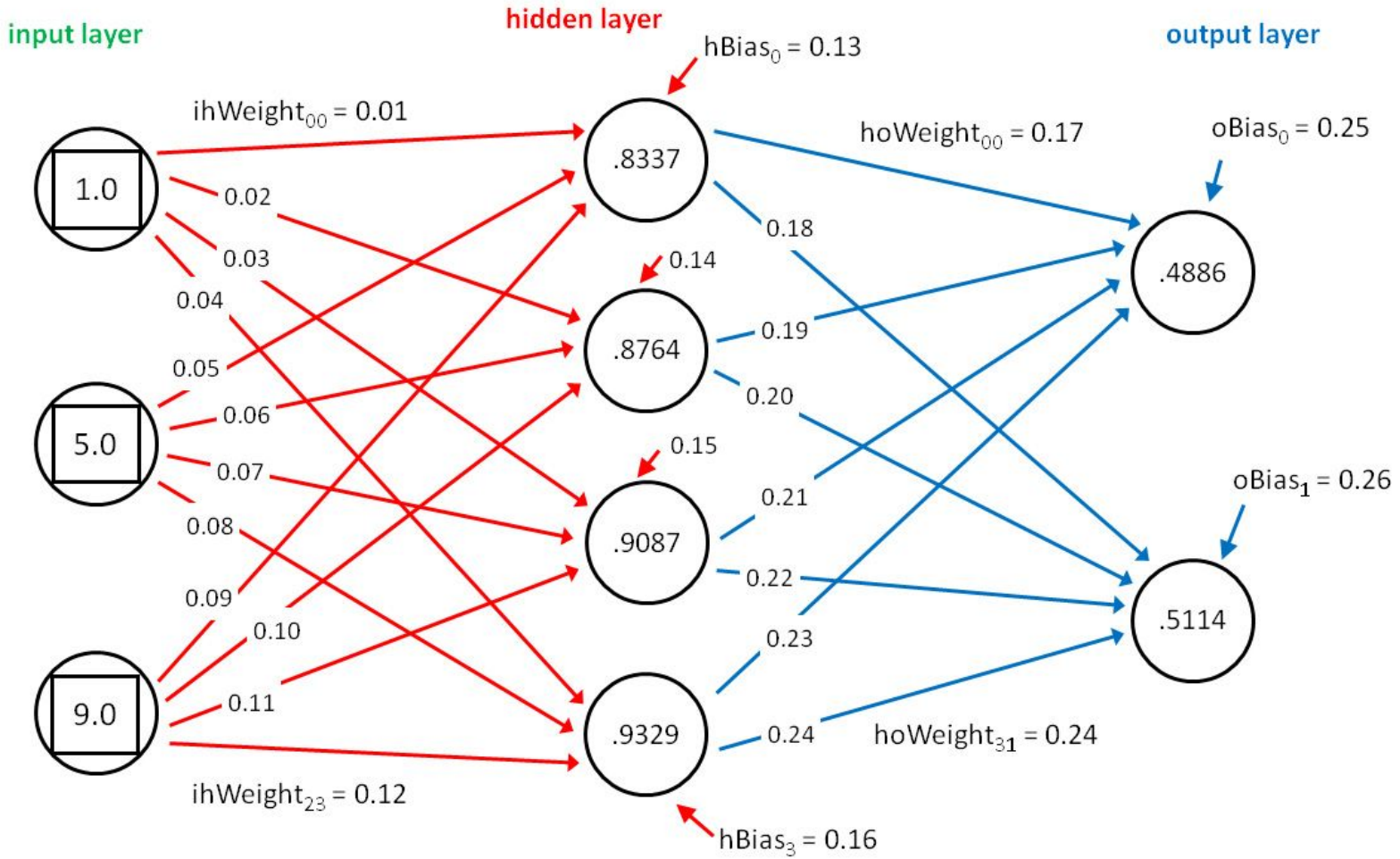
- Basic Variant has faster convergence
- Local Best Variant is more robust against Local Optimas
- In the local best version small clusters of particles are formed.
- In the global best version a single cluster of particles is formed

Neural Networks

- Neural Networks consists of Matrix Multiplications along with Non Linearities(Activation Functions).
- Weights are numbers used in the Matrix Multiplication.
- Biases are numbers added post Matrix Multiplication.
- Weights , Biases, and Parameters in Activation Functions(Prelu) form the Dimensional Space in a Neural Network.
- Training a Neural Network is finding the best position in this dimensional space

Backpropagation and Memory

- Backpropagation is used to train Neural Networks.
- It is a series of chained derivatives on the basis of which updates can be made to train the neural network.
- The complexity of Backpropagation can lie between 0(n2) to 0(1) depending upon whether or not activation at each layer are stored.
- Most Frameworks store activations at each Layer.


input layer
hidden layer
output layer
hBias0 = 0.13
ihWeight00 = 0.01
hoWeight00 = 0.17
oBias0 = 0.25
1.0
0.02
0.03
0.04
.8337
0.18
0.14
.4886
0.05
0.06
0.07
.8764
0.19
0.20
5.0
0.15
oBias1 = 0.26
0.08
0.09
0.10
0.11
.9087
0.21
0.22
0.23
.5114
9.0
.9329
0.24
hoWeight31 = 0.24
ihWeight23 = 0.12
hBias3 = 0.16

Application of PSO to Neural Network

- Each Particle is a Neural Network
- The dimensions are the weights, biases and parameters for activation functions.
- The fitness for a position is the value given by the loss function at that position.

Implementation Details

- Libraries : Tensorflow (>=1.4)
- Language : Python 3

Conclusions

- Use of PSO for Initializing Neural Networks before Training with Gradient Descent is worth exploring
- The Hybrid Approach seems to be robust against local optimas and converges faster when compared to only gradient descent.
- Use of PSO seems to a viable alternative to current multi-GPU training approaches.