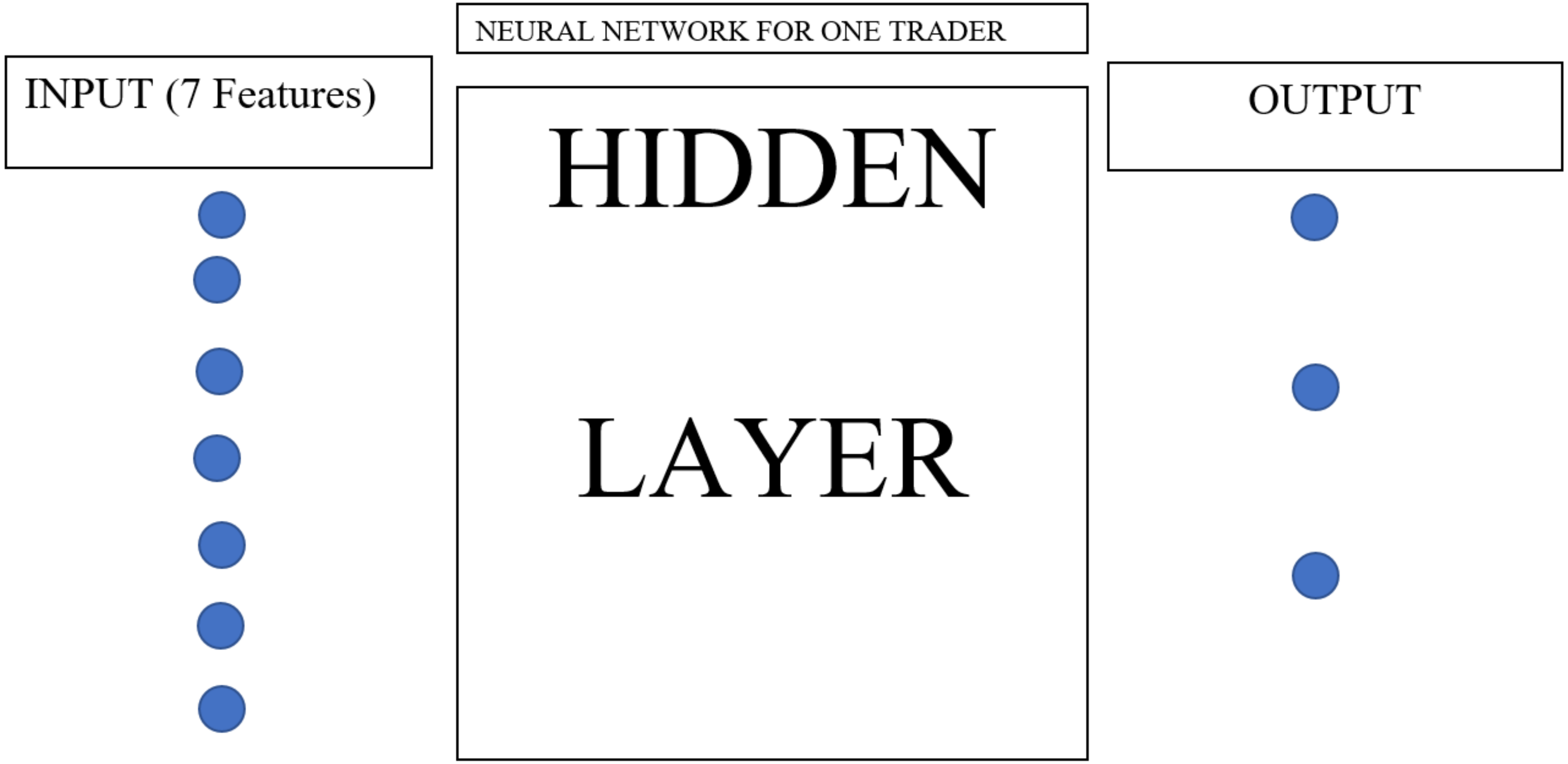
Selecting the best trading strategies through the combination of Genetic algorithm and Neural Network

Implementation

- Run the genetic algorithm for 50 generations
- Each generations has 50 traders
- Each has a budget of 1 million dollars
- Each traders implements their own neural network
- Each neural network has 100 learning epoch to adjust the weights associated with it to improve its performance.



NEURAL NETWORK FOR ONE TRADER

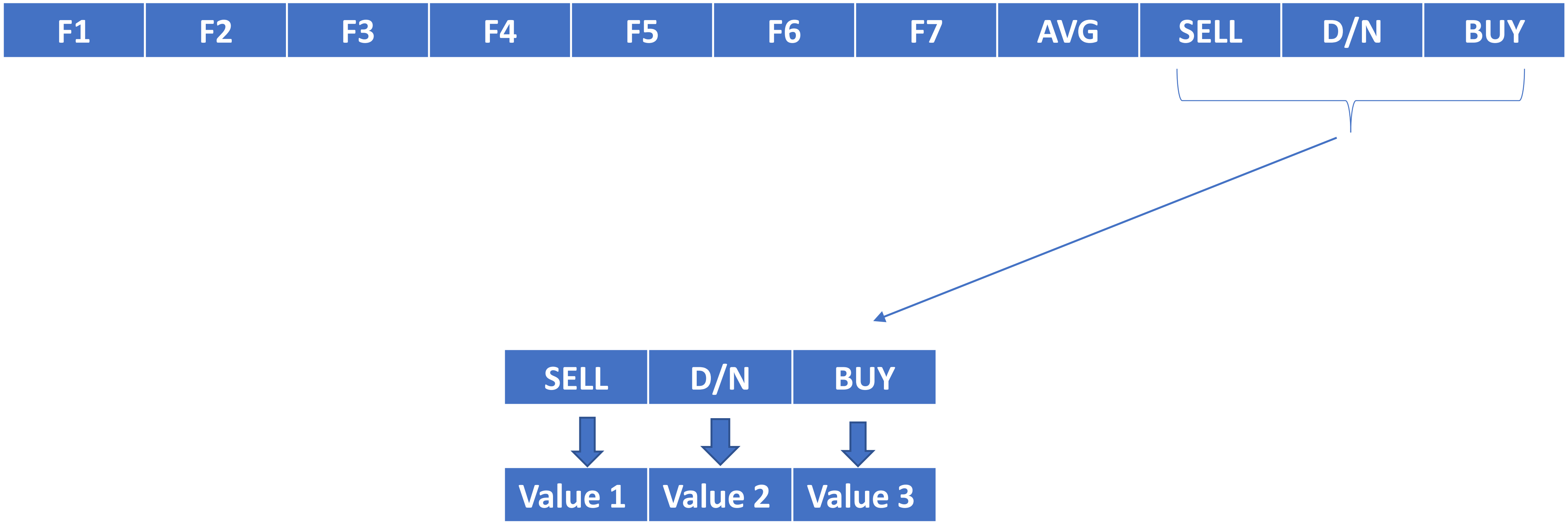
INPUT (7 Features)

HIDDEN

LAYER

OUTPUT

Features
On Balance Volume
Accumulation Distribution Line
Simple Moving Average
Aroon Indicator
Exponential Moving Average
Relative Strength Index
Stochastic Oscillator



F1 F2 F3 F4 F5 F6 F7 AVG SELL D/N BUY

SELL D/N BUY

Value 1 Value 2 Value 3

- If value 2 is the largest among all three values then do nothing
- If value 1 is the larges among all three values then… $\frac{value\ 1}{value\ 3} > threshold$ → sell
- If value 1 is the larges among all three values then… $\frac{value\ 3}{value\ 1} > threshold$ → buy

Data Used

- The data we used for training and testing of our algorithm was from the S&P 500 index and it's time series over the period from April 1st, 2010- May 20th 2016.
- The data was broken into minute data but we converted it into 15 min data in order to produce better results since there was greater variation in price between them.

Training and Testing data

- Our data breakdown was 80% training then 20% for testing, so approximately 2200 training days and 450 testing days.
- The S&P 500 data that was used in our study had a generally increasing trend and is important to note before consulting our results.

Experiment Results

- Project implemented in Python
- Neural Networks implemented using Tensorflow
- Promising results

Experiment Results

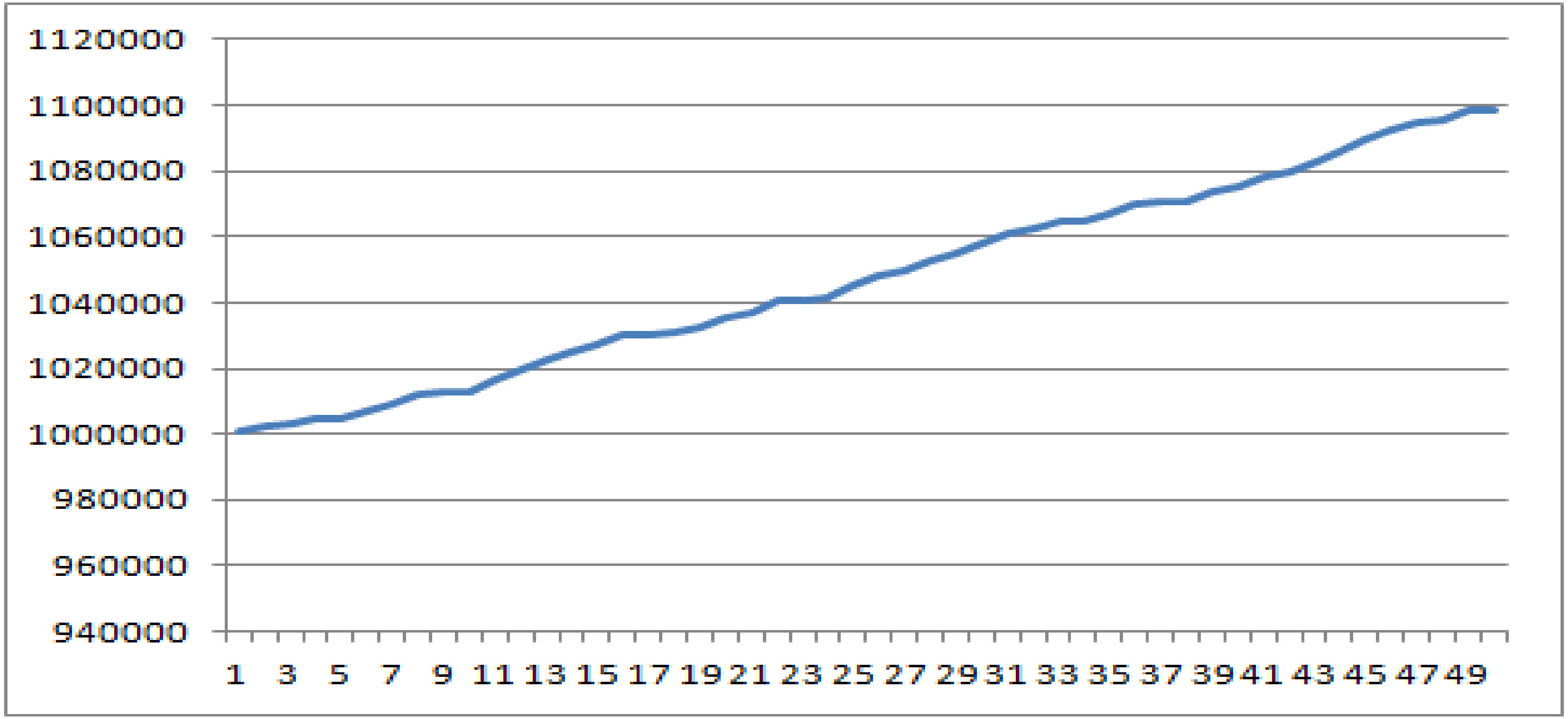
- The running time seemed to be a challenge
- Even though we set the total number of population to be 50 per generation, the model was taking **more than 20 minutes** to run one generation on a computer with a i5 2.9GHz processor.
- To overcome this issue we implemented some techniques:
  - We saved each Neural Network configuration locally, within specific binary files.
  - Implemented batch-training: dividing the training set in smaller parts called batches and modify the weights for each batch
  - The number of nodes in the hidden layer bounded within (1-100)

Experiment Results

The winning "DNA": [1,0,1,0,1,0,0,60,0.36548919802, 1098519.37182342]

- The "On Balance Volume", the "Simple Moving Average" and the "Exponential Moving Average" generated the best predicting results (accuracy over training dataset : **78.24%;** accuracy over test set: **50.8%**)
- 60 nodes in Hidden Layer
- Returns: **$98519.37**

Best profits over 50 generations



We can assume that the return rate will keep improving given more generations.

Further Improvements

- More generations
- More population
- More features
- More Hidden layers
- More Nodes per layer

=

**More
Computing
Power**