Optimized Hybrid Scaled Neural Branch Predictor* ESESC** Implementation

CE 202 Computer Architecture
University of California Santa Cruz

Final Project

David Kooi
Fall 2018

* http://hpca23.cse.tamu.edu/taco/pdfs/iccd2011_dist.pdf DOI:10.1109/ICCD.2011.6081385
** http://masc.soe.ucsc.edu/esesc/

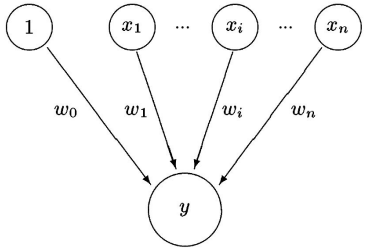
Perceptron Basics

A perceptron is a mathematical function that can "learn" two *linearly separable* data sets.

$$y = w_0 + \sum_{i=1}^{n} x_i w_i.$$

The output Y is run through an activation function to determine the "class" of the input data.

Often the sign of Y is taken where negative values indicate membership of one class and positive values a membership of another class.



Multiple inputs, x0...xn are "weighted" by weights w0...wn. Often w0 is a bias value.

```
if sign(y_out) ≠ t or |y_out| ≤ θ then
        for i := 0 to n do
                w_i := w_i + t x_i
        end for
end if
```

Incorrect perceptron outputs are "trained" by adjusting weights according to the output error.

Perceptrons for Branch Prediction

- Developed into a viable idea by Daniel A. Jiménez in 2001. (DOI: 10.1109/HPCA.2001.903263)
- Optimized by Jiménez in 2011 resulting in the OHSNAP predictor. OHSNAP is the basis for this project.

Perceptrons for Branch Prediction

- Uses branch history to “learn” taken/not taken patterns.

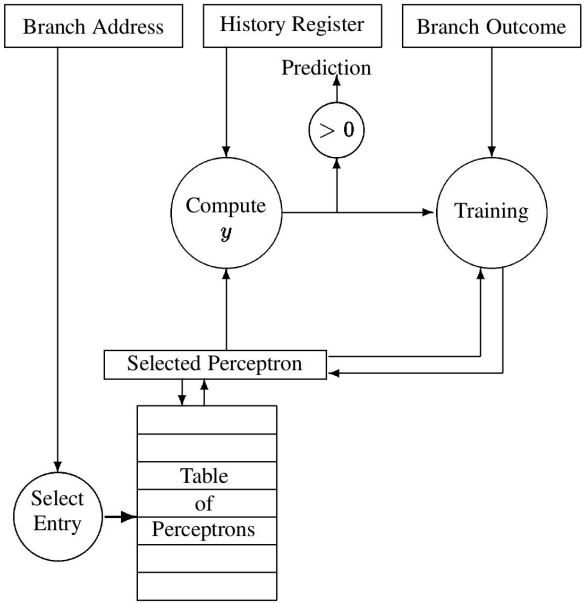


Figure 2: Perceptron Predictor Block Diagram. The branch address is hashed to select a perceptron that is read from the table. Together with the global history register, the output of the perceptron is computed, giving the prediction. The perceptron is updated with the training algorithm, then written back to the table.

OHSNAP Predictor

Optimized Hybrid Scaled Neural Analog Predictor

- A 2011 attempt to make a feasible neural branch predictor with hardware limitations.
- Project uses ESESC simulator so OH-SNP is more a accurate description for this project implementation.

Optimizations

- **Global and Per-Branch History**: GHR table and Per-Branch history table is kept and used for prediction
  - Not used
- **Ragged Array**: W matrix is represented by ragged array, i.e the size of row varies by index of column
  - More recent branches are given more weights to compute. Saves on memory footprint
  - Not used because this is a hardware optimization
- **Training Coefficient Vectors**
  - Not used: A static equation for weight coefficients is used
- **Training Θ**: The adaptive training algorithm from O-GEHL is used to determine the training threshold
  - Used, but improvement not achieved over static Θ
- **Hybrid Predictor**: Perceptrons are accurate with high neural output, but “indecisive” for values around zero
  - OGEHL predictor is used when neural output drops below a certain value
  - Used

Training Θ

Θ is a training coefficient. Correct predictions may be “uncertain”. That is: Their neural output is low. To overcome this, correct prediction magnitudes lower than Θ are trained.

Trivially Θ may be static. However, to adapt to different programs OHSNAP proposes to train Θ using the training method used in OGEHL:

if ((p !=Out) {TC= TC + 1; if (TC == Saturated-Positive) {$\theta = \theta + 1$; TC=0;} }
if ((p == Out) & ($|S| \leq \theta$)) {TC= TC - 1; if (TC == SaturatedNegative){$\theta = \theta - 1$; TC=0;}}

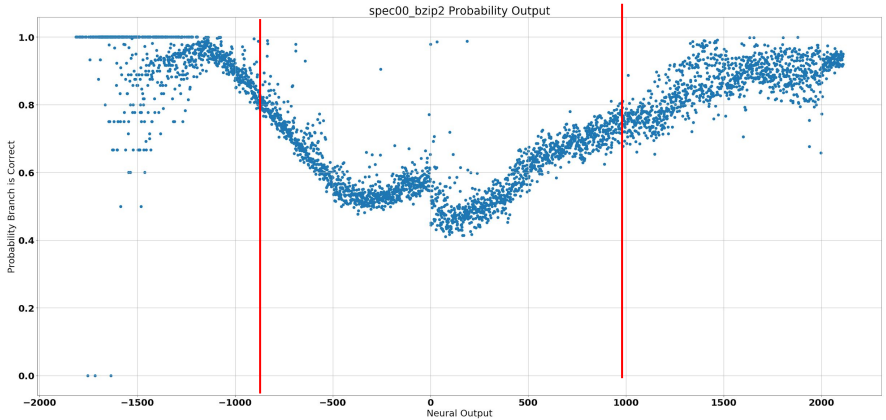
TC is a threshold counter. If TC saturates, Θ is incremented or decremented.

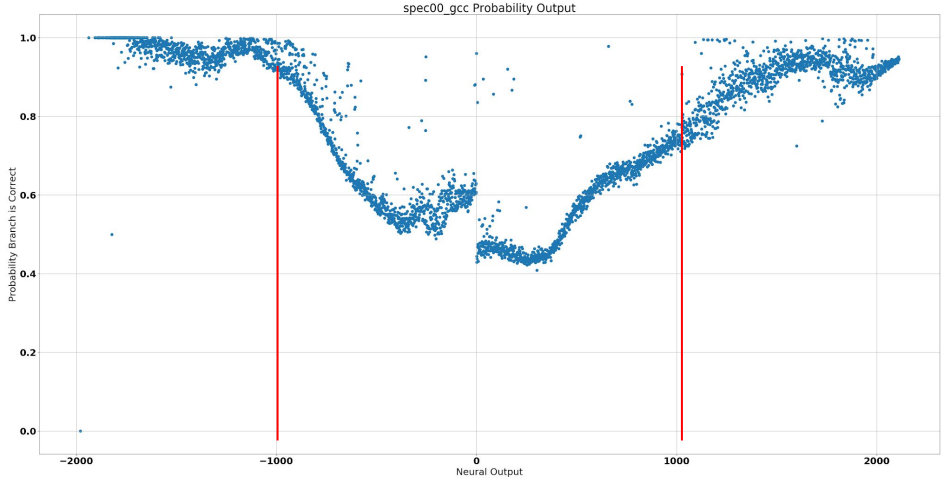
OHSNAP uses a table of Θ values. The simulation also can use a table of Θ values: One Θ per perceptron.

This project did not achieve increased optimization using this method.

Hybrid (off)

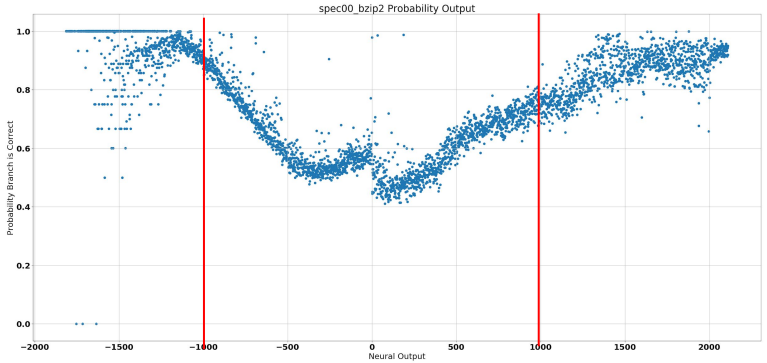
A perceptron based branch predictor is highly accurate when the magnitude of perceptron output is high, but accuracy is less for “undecided values” close to zero.

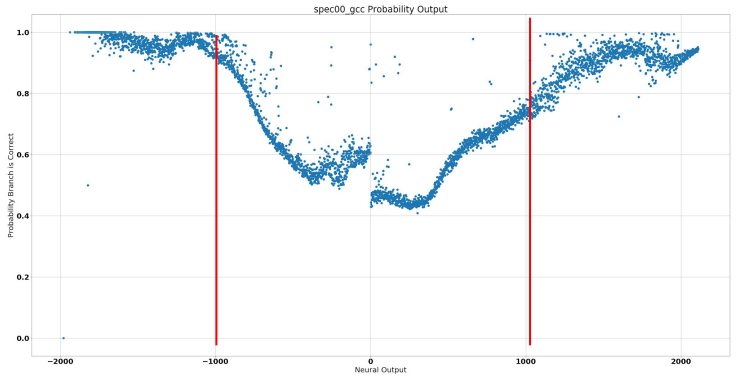




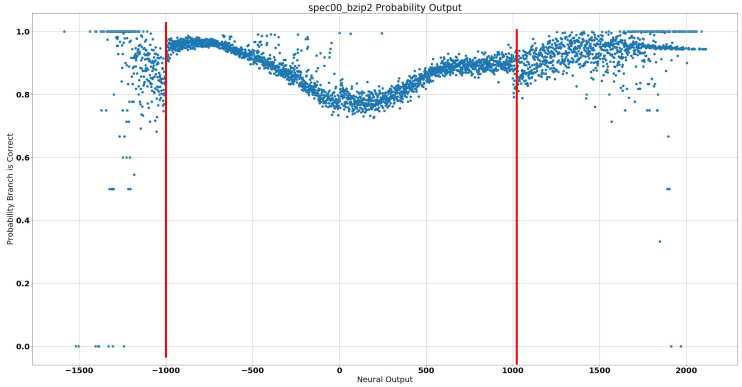
OHSNAP uses a gshare predictor when the magnitude of the perceptron is below a threshold. This simulation uses the OGEHL predictor for neural outputs below a specified magnitude. For example: OGEHL can be used if the neural output magnitude is below 1000.

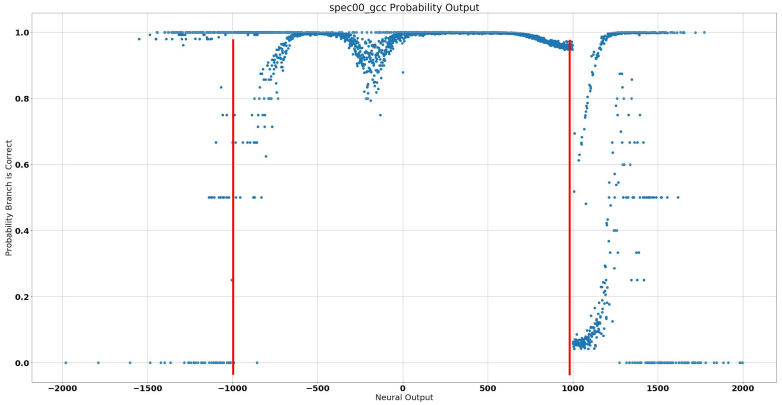
Hybrid (on/off)





Hybrid threshold set to 0 (off)





Hybrid threshold set to 1000

Scaled

Branch history elements are more significant the closer they are. We scale them by the function:

$$f(i) = 1/(A+B\times i)$$

This fits the branch outcome correlation curve:

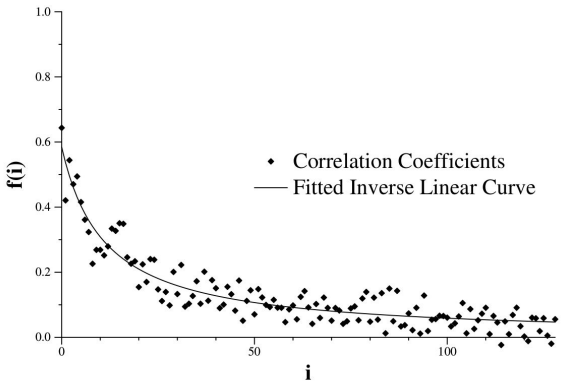


Fig. 2. Weight position and branch outcome correlation. This figure is taken from the original SNAP paper [1].

For the simulation A and B are chosen:

A=0.1111

B=0.037

Following the paper DOI:10.1109/MICRO.2008.4771812

Prediction Detail

```
2037      // Get hash of PC
2038      row = dinst->getPC() % numPerceptrons;
2039      // Make sure perceptron index in range
2040      if(row == numPerceptrons){ row--; }
2041      I(row < numPerceptrons);
2042
2043      // Add all weights
2044      int32_t sum = ptable[row + 0]; // Get bias
2045      for(int i = 0; i < glength; i++){
2046
2047          assert(row + (i+1) <= (glength+1) * numPerceptrons); // Assert array overflow
2048
2049          // Scaling coefficient
2050          float A = 0.1111;
2051          float B = 0.037;
2052          float C = 1;///(A+B*i);
2053
2054          if(ghr[i]){
2055              sum += (int32_t)(C*ptable[row + (i+1)]);
2056          }else{
2057              sum -= (int32_t)(C*ptable[row + (i+1)]);
2058          }
2059
2060      }
2061      ptaken_n = sum > 0 ? 1 : -1;
2062
```

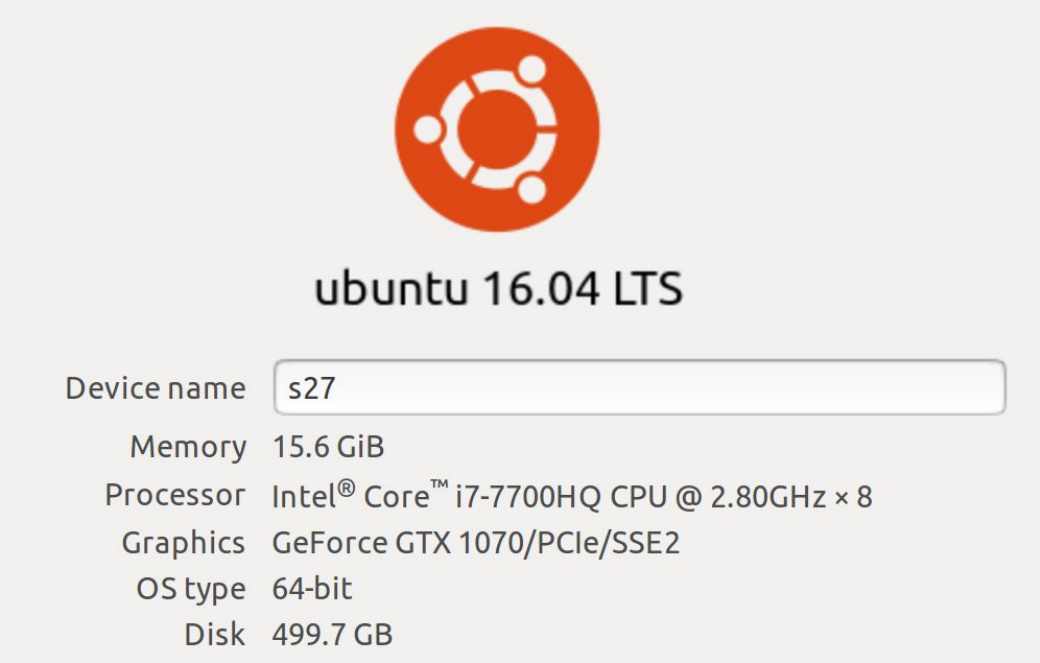
Training Detail(1)

```
2070      // Update perceptron table and ghr
2071      if(doUpdate){
2072
2073          // Update branch/path histories
2074          ghr <<= 1;
2075          ghr[0] = taken ? 1 : 0;
2076
2077          if((ptaken_n != taken_n)){ // Incorrect prediction
2078
2079              int8_t error = taken_n - ptaken_n;
2080
2081              // Adjust training theta
2082              tc_table[row]++;
2083              if(tc_table[row] == 128){ // Saturate as 7bit counter and reset
2084                  theta_table[row]++;
2085                  tc_table[row]= 0;
2086              }
2087
2088              train_weights(error, taken, row);
2089
2090
2091          }else{ // Correct prediction
2092
2093              // Adjust training theta
2094              tc_table[row]--;
2095              if(tc_table[row] == -128){ // Saturate as 7bit counter and reset
2096                  theta_table[row]--;
2097                  tc_table[row]= 0;
2098              }
2099
2100              if(re_abs_sum < static_theta){
2101              //if( re_abs_sum < theta_table[row]){ // If weight is below theta we still need to train
2102                                                  // even if it is correct.
2103                                                  // I.e, we have a low
2104                                                  // confidence correct
2105                                                  // prediction
2106
2107                  train_weights(1, taken, row);
2108
2109              }
2110          }
2111
2112      }
```

Training Detail(2)

```
2157 void OhSnap::train_weights(int8_t error, bool taken, uint16_t row){
2158
2159     // Update bias
2160     if(taken){
2161         ptable[row + 0] += 1;
2162     }else{
2163         ptable[row + 0] -= 1;
2164     }
2165
2166     // Update correlating weights
2167     for(int i=0; i < glength; i++){
2168         if(ghr[i]){
2169             ptable[row + (i+1)] += error;
2170         }else{
2171             ptable[row + (i+1)] -= error;
2172         }
2173
2174         // Saturate the weights
2175         if( ptable[row + (i+1)] > saturation){
2176             ptable[row + (i+1)] = saturation;
2177         }
2178     }
2179
2180 }
```

System Used for Simulation



Results

Trial	# Perceptrons	GHR Length	Training Theta	Scaling Coefficients	Hybrid Threshold	Average % Branch Prediction
1	4095	32	static	f(i)=1	off	67.5
2	4095	32	static	f(i)=1/A+B*i	off	67.75
3	4095	256	static	f(i)=1/A+B*i	off	68.5
4	4095	32	dynamic	f(i)=1/A+B*i	off	63
5	4095	32	static	f(i)=1	100	70
6	4095	32	static	f(i)=1	650	84.5
7	4095	32	static	f(i)=1	1000	94.25

Conclusion and Future Work

- Predictor simulation was performed without memory or hardware constraints in mind
- Efforts to optimize neural prediction did not achieve appreciable results, but more empirical testing may result in better neural prediction.
- Hybrid prediction works well to improve average % correct

References

An optimized scaled neural branch predictor 10.1109/ICCD.2011.6081385

Dynamic branch prediction with perceptrons 10.1109/HPCA.2001.903263

Fast path-based neural branch prediction 10.1109/MICRO.2003.1253199

Low-power, high-performance analog neural branch prediction 10.1109/MICRO.2008.4771812

Piecewise Linear Branch Prediction 10.1109/ISCA.2005.40

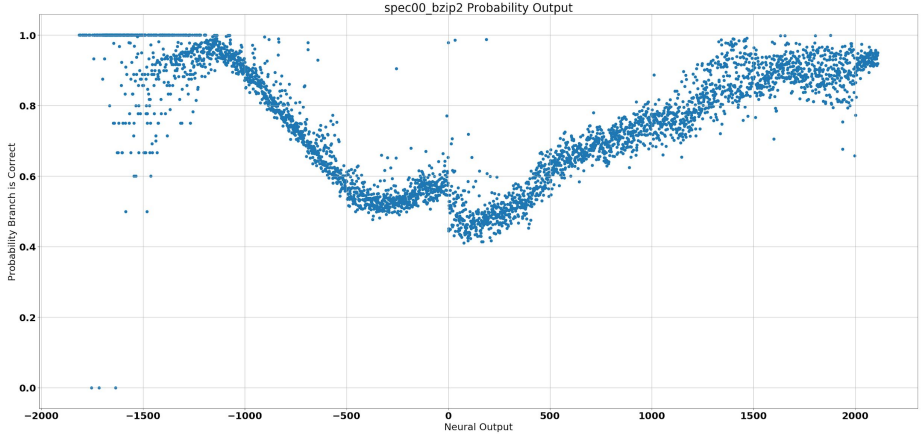
Appendix

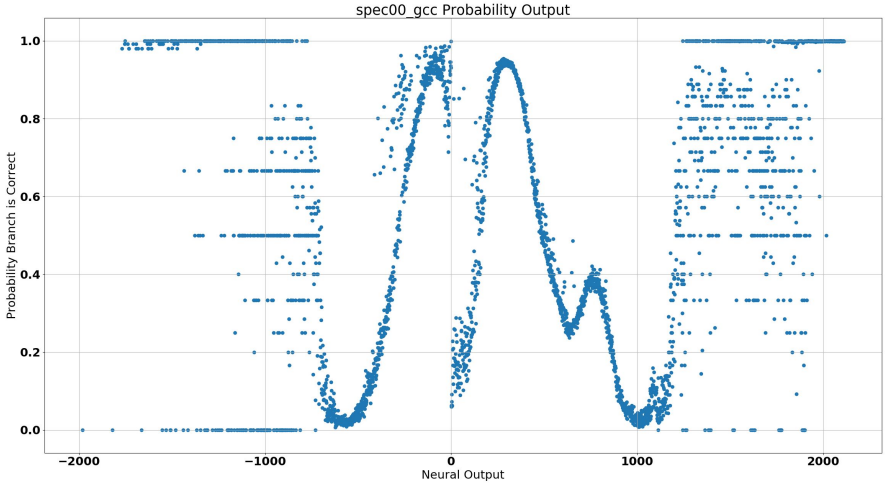
Trial 1

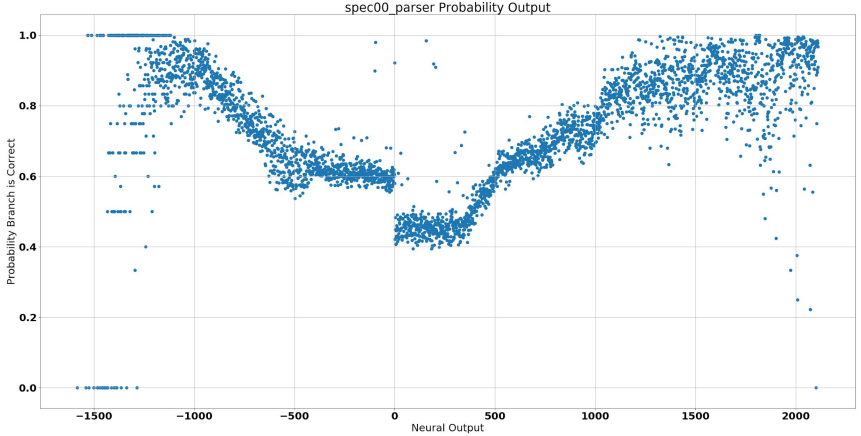
Trial	# Perceptrons	GHR Length	Training Theta	Scaling Coefficients	Hybrid Threshold
1	4095	32	static	f(i)=1	off

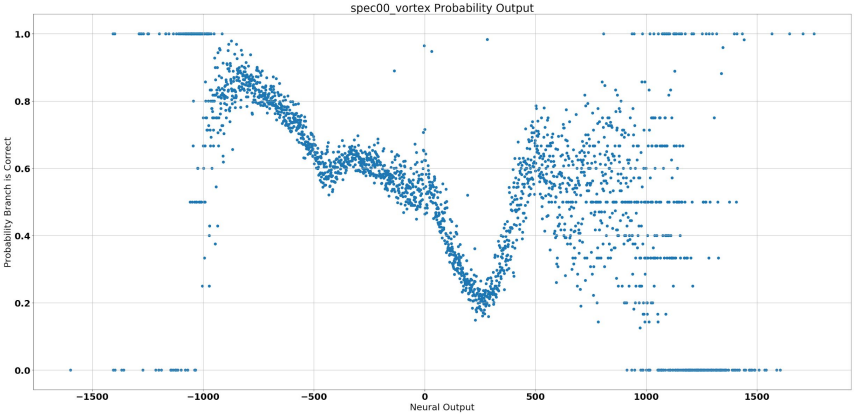
Benchmark	% Correct
spec00_bzip2	73
spec00_gcc	67
spec00_parser	68
spec00_vortex	62

Trial 1


spec00_bzip2 Probability Output
Probability Branch is Correct
Neural Output


spec00_gcc Probability Output
Probability Branch is Correct
Neural Output


spec00_parser Probability Output
Probability Branch is Correct
Neural Output

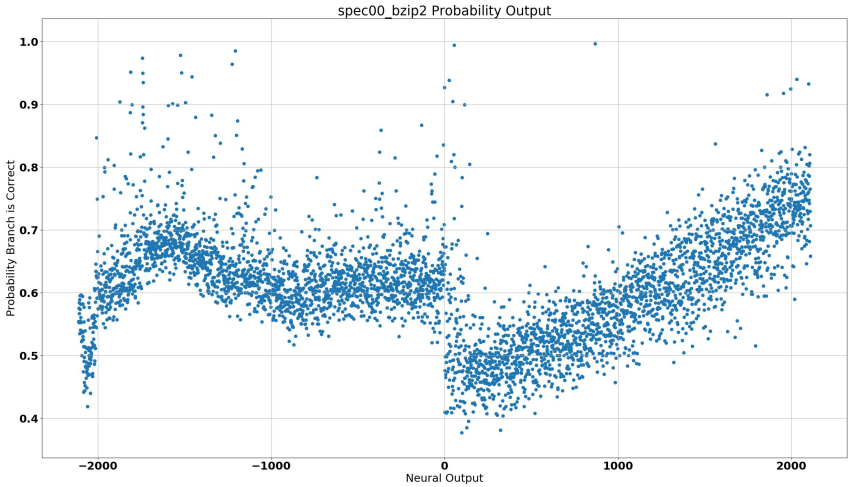

spec00_vortex Probability Output
Probability Branch is Correct
Neural Output

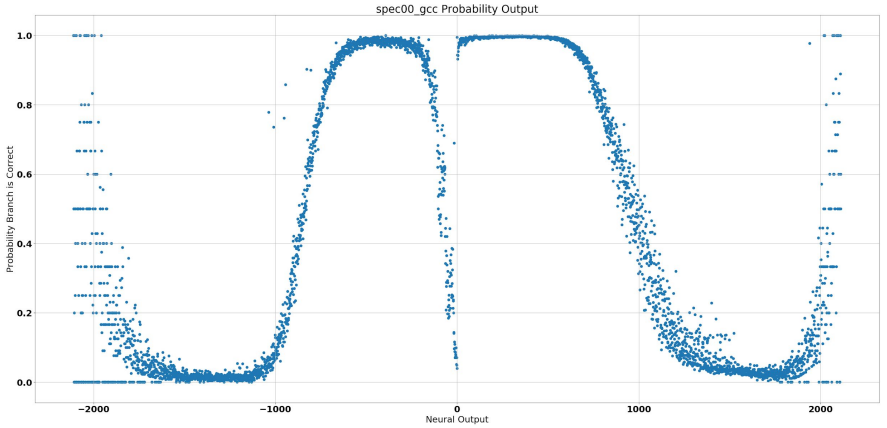
Trial 2

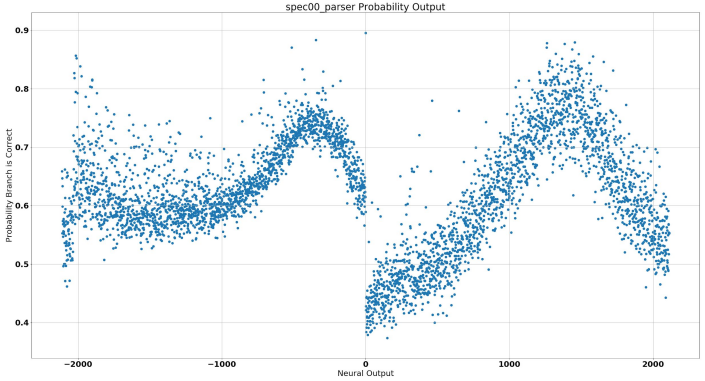
Trial	# Perceptrons	GHR Length	Training Theta	Scaling Coefficients	Hybrid Threshold
2	4095	32	static	f(i)=1/A+B*i	off

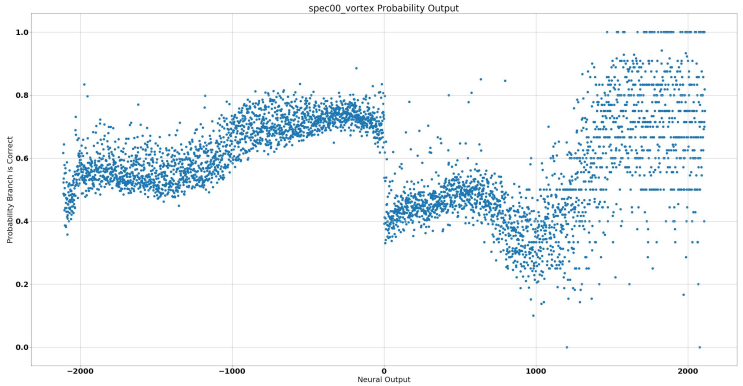
Benchmark	% Correct
spec00_bzip2	74
spec00_gcc	67
spec00_parser	67
spec00_vortex	63

Trial 2


spec00_bzip2 Probability Output
Probability Branch is Correct
Neural Output
1.0
0.9
0.8
0.7
0.6
0.5
0.4
−2000
−1000
0
1000
2000


spec00_gcc Probability Output
Probability Branch is Correct
Neural Output
1.0
0.8
0.6
0.4
0.2
0.0
−2000
−1000
0
1000
2000


spec00_parser Probability Output
Probability Branch is Correct
Neural Output
0.9
0.8
0.7
0.6
0.5
0.4
−2000
−1000
0
1000
2000

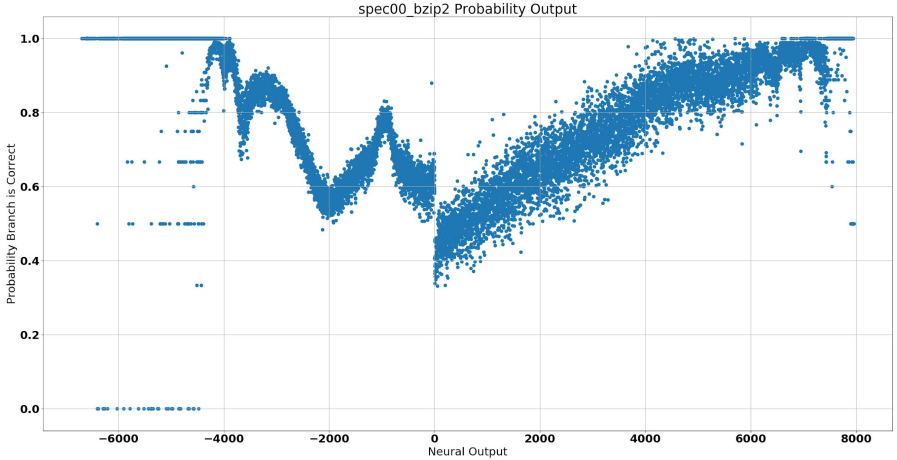

spec00_vortex Probability Output
Probability Branch is Correct
Neural Output
1.0
0.8
0.6
0.4
0.2
0.0
−2000
−1000
0
1000
2000

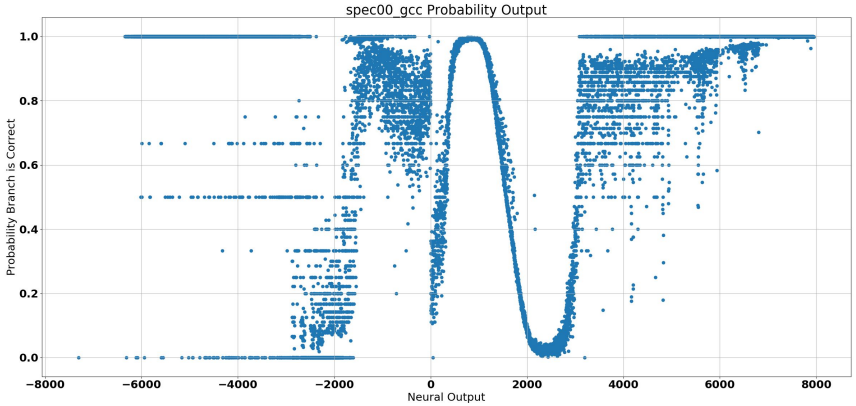
Trial 3

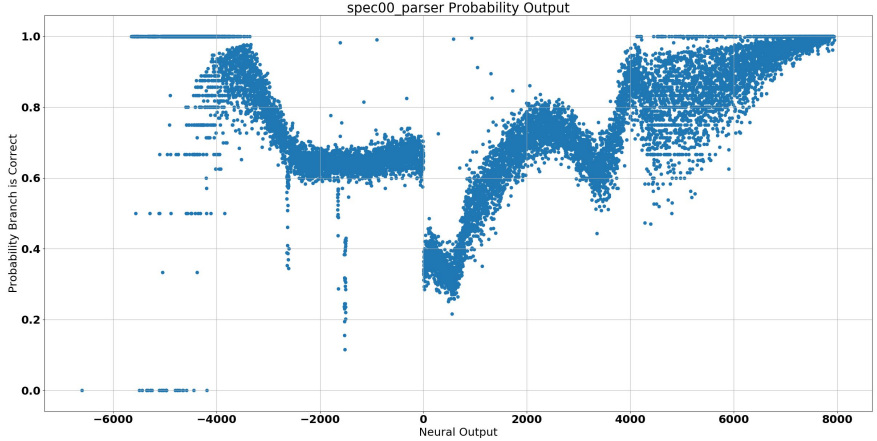
Trial	# Perceptrons	GHR Length	Training Theta	Scaling Coefficients	Hybrid Threshold
3	4095	256	static	f(i)=1/A+B*i	off

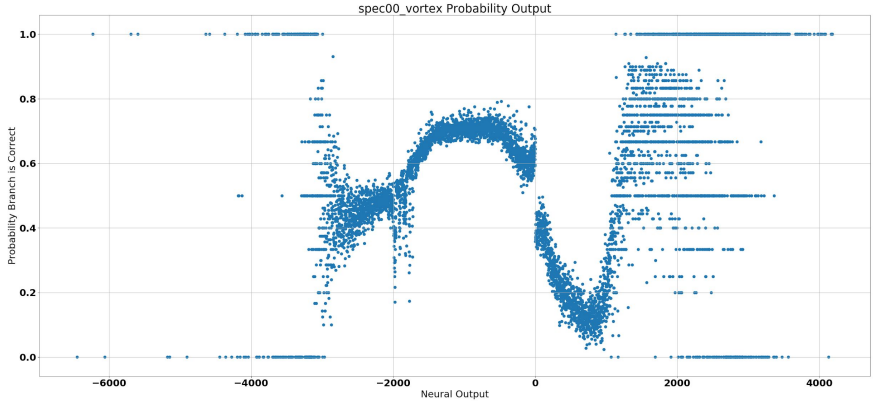
Benchmark	% Correct
spec00_bzip2	74
spec00_gcc	72
spec00_parser	67
spec00_vortex	61

Trial 3


spec00_bzip2 Probability Output
Probability Branch is Correct
Neural Output


spec00_gcc Probability Output
Probability Branch is Correct
Neural Output


spec00_parser Probability Output
Probability Branch is Correct
Neural Output

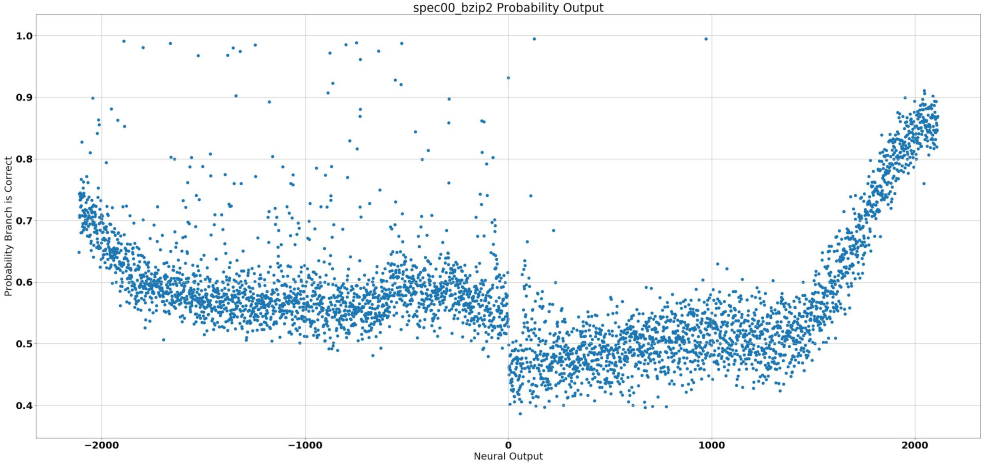

spec00_vortex Probability Output
Probability Branch is Correct
Neural Output

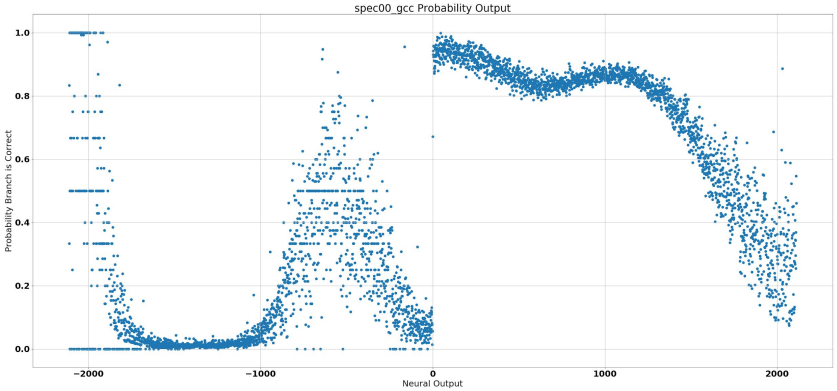
Trial 4

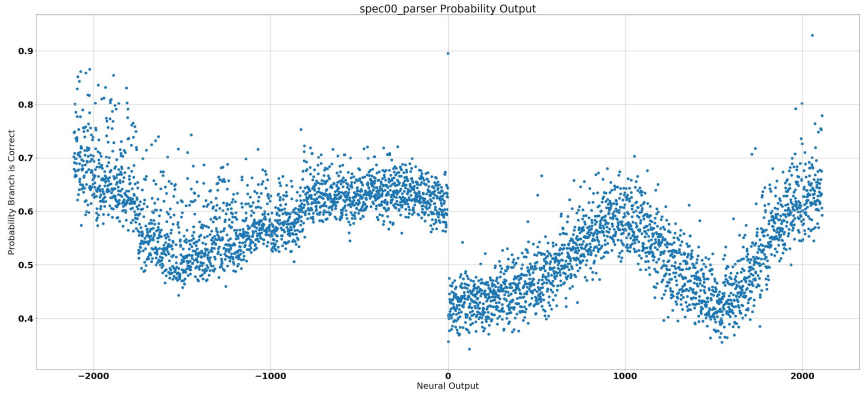
Trial	# Perceptrons	GHR Length	Training Theta	Scaling Coefficients	Hybrid Threshold
4	4095	32	dynamic	f(i)=1/A+B*i	off

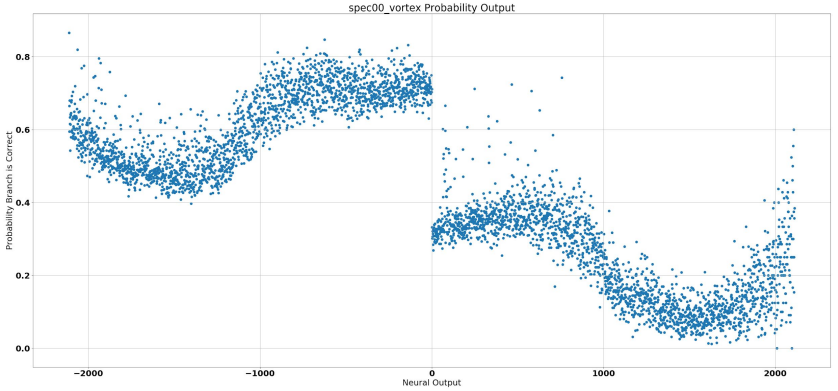
Benchmark	% Correct
spec00_bzip2	71
spec00_gcc	63
spec00_parser	60
spec00_vortex	58

Trial 4


spec00_bzip2 Probability Output
Probability Branch is Correct
Neural Output


spec00_gcc Probability Output
Probability Branch is Correct
Neural Output


spec00_parser Probability Output
Probability Branch is Correct
Neural Output

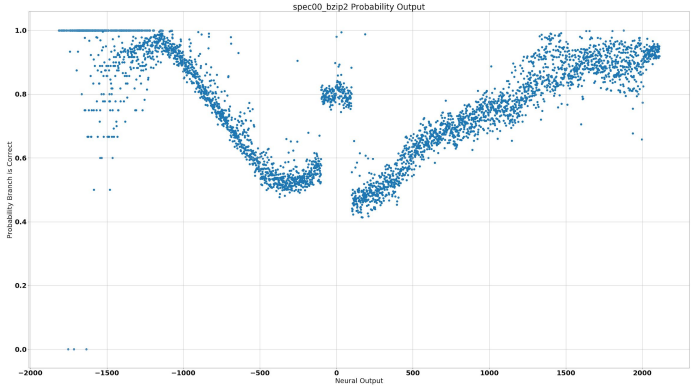

spec00_vortex Probability Output
Probability Branch is Correct
Neural Output

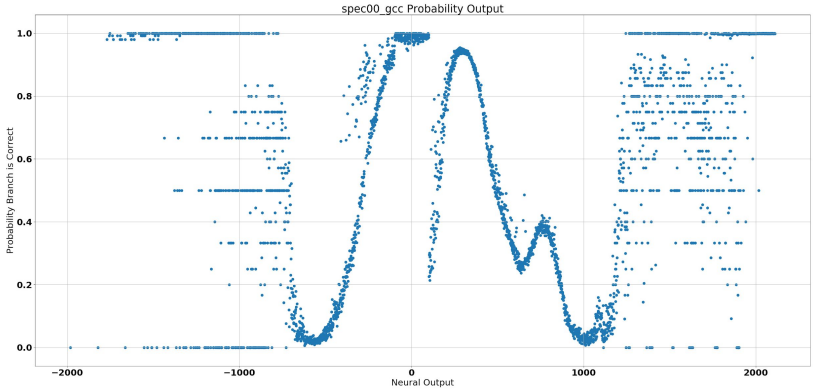
Trial 5

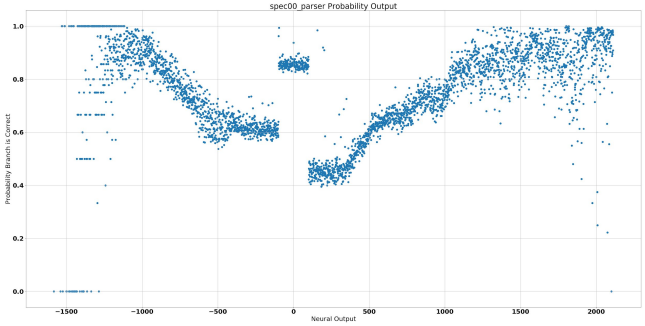
Trial	# Perceptrons	GHR Length	Training Theta	Scaling Coefficients	Hybrid Threshold
5	4095	32	static	f(i)=1	100

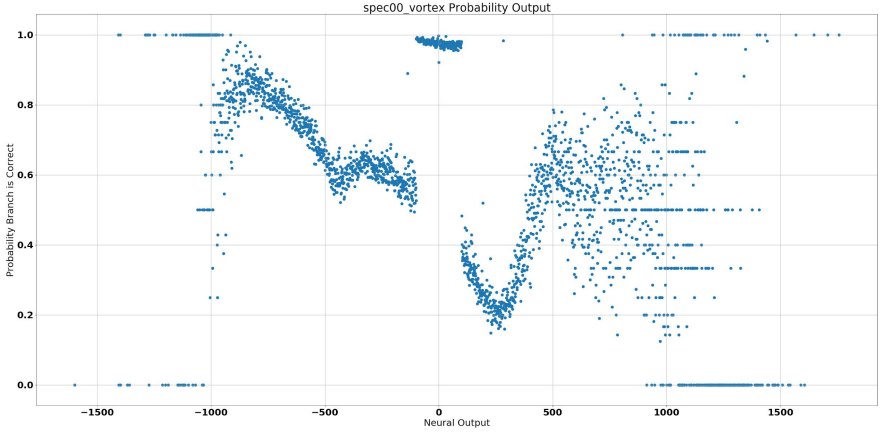
Benchmark	% Correct
spec00_bzip2	75
spec00_gcc	68
spec00_parser	70
spec00_vortex	67

Trial 5


spec00_bzip2 Probability Output
Probability Branch is Correct
Neural Output


spec00_gcc Probability Output
Probability Branch is Correct
Neural Output


spec00_parser Probability Output
Probability Branch is Correct
Neural Output

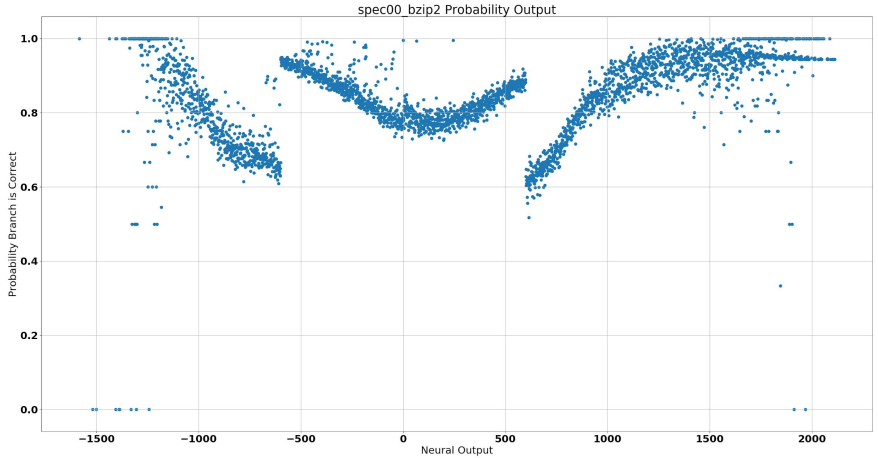

spec00_vortex Probability Output
Probability Branch is Correct
Neural Output

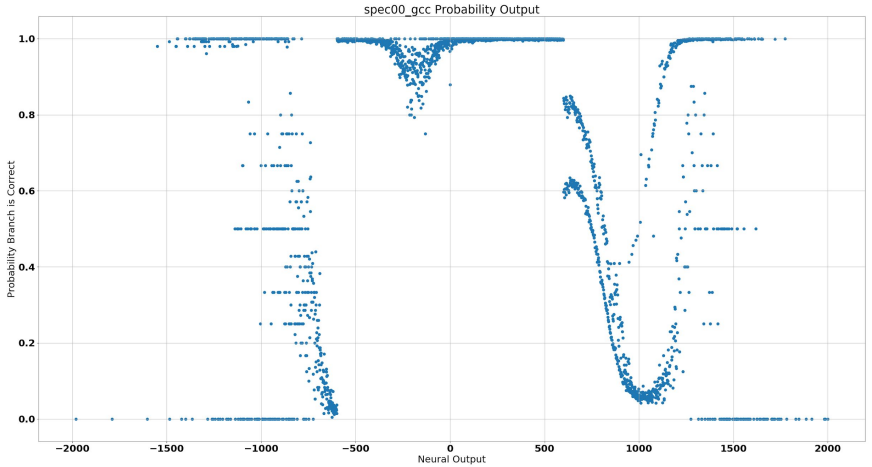
Trial 6

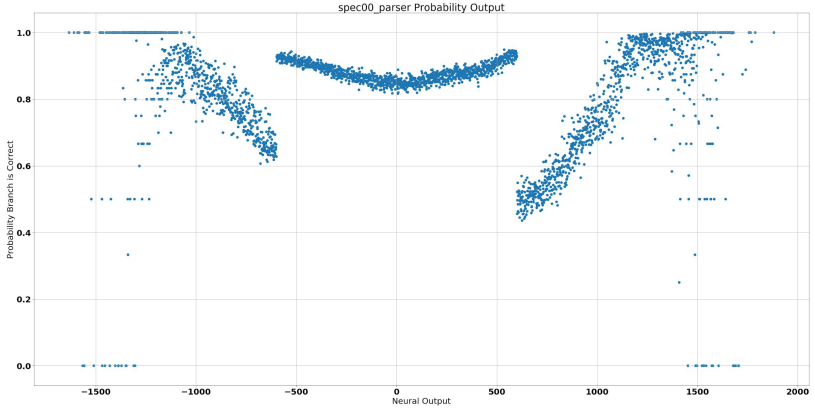
Trial	# Perceptrons	GHR Length	Training Theta	Scaling Coefficients	Hybrid Threshold
6	4095	32	static	f(i)=1	650

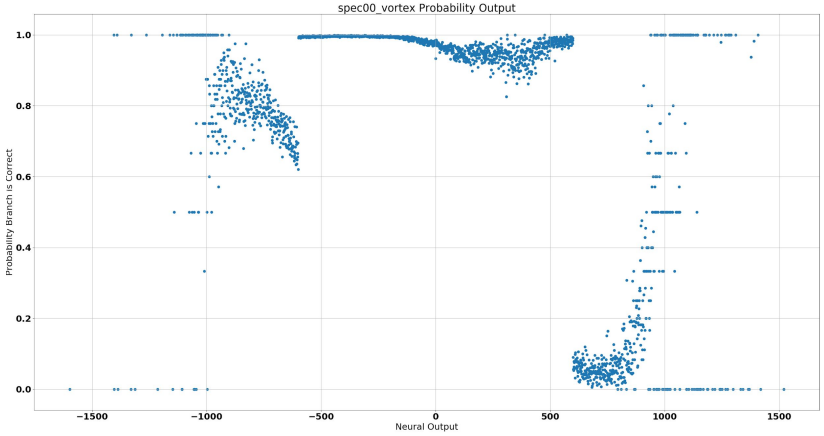
Benchmark	% Correct
spec00_bzip2	85
spec00_gcc	74
spec00_parser	84
spec00_vortex	95

Trial 6


spec00_bzip2 Probability Output
Probability Branch is Correct
Neural Output


spec00_gcc Probability Output
Probability Branch is Correct
Neural Output


spec00_parser Probability Output
Probability Branch is Correct
Neural Output

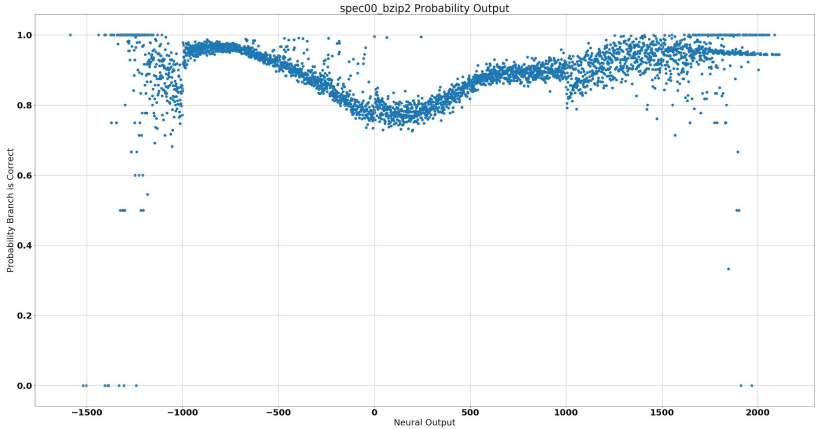

spec00_vortex Probability Output
Probability Branch is Correct
Neural Output

Trial 7

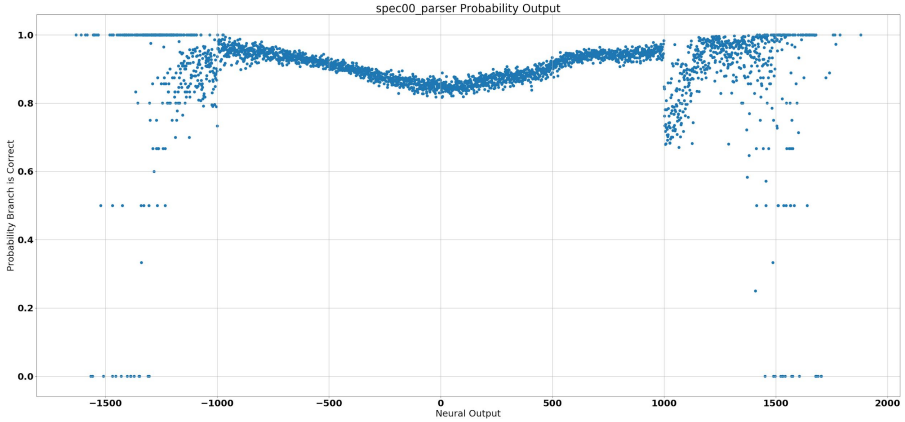
Trial	# Perceptrons	GHR Length	Training Theta	Scaling Coefficients	Hybrid Threshold
7	4095	32	static	f(i)=1	1000

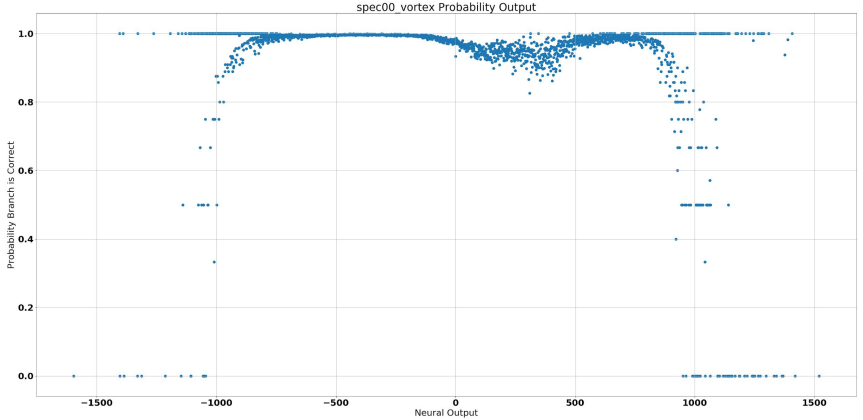
Benchmark	% Correct
spec00_bzip2	90
spec00_gcc	97
spec00_parser	91
spec00_vortex	99

Trial 6


spec00_bzip2 Probability Output
Probability Branch is Correct
Neural Output


spec00_gcc Probability Output
Probability Branch is Correct
Neural Output


spec00_parser Probability Output
Probability Branch is Correct
Neural Output


spec00_vortex Probability Output
Probability Branch is Correct
Neural Output