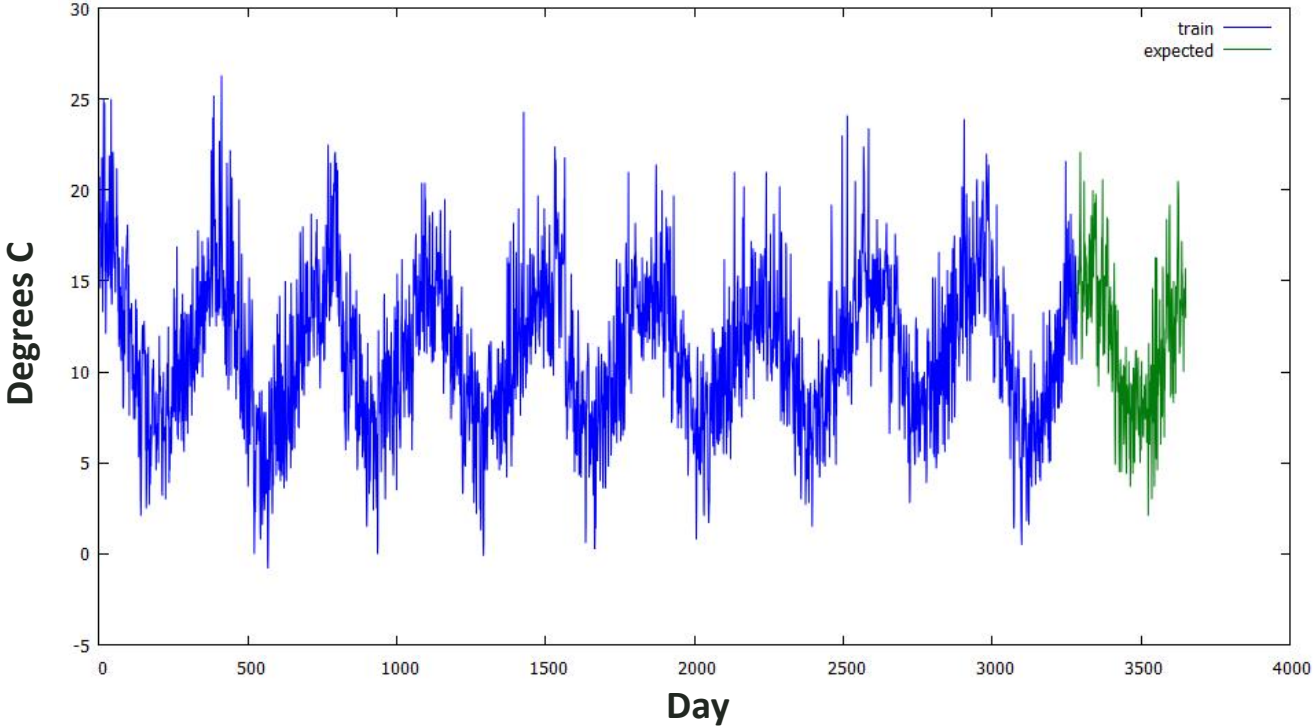
USING GENETIC ALGORITHMS TO OPTIMIZE RECURRENT NEURAL NETWORKS

Joe Schneible

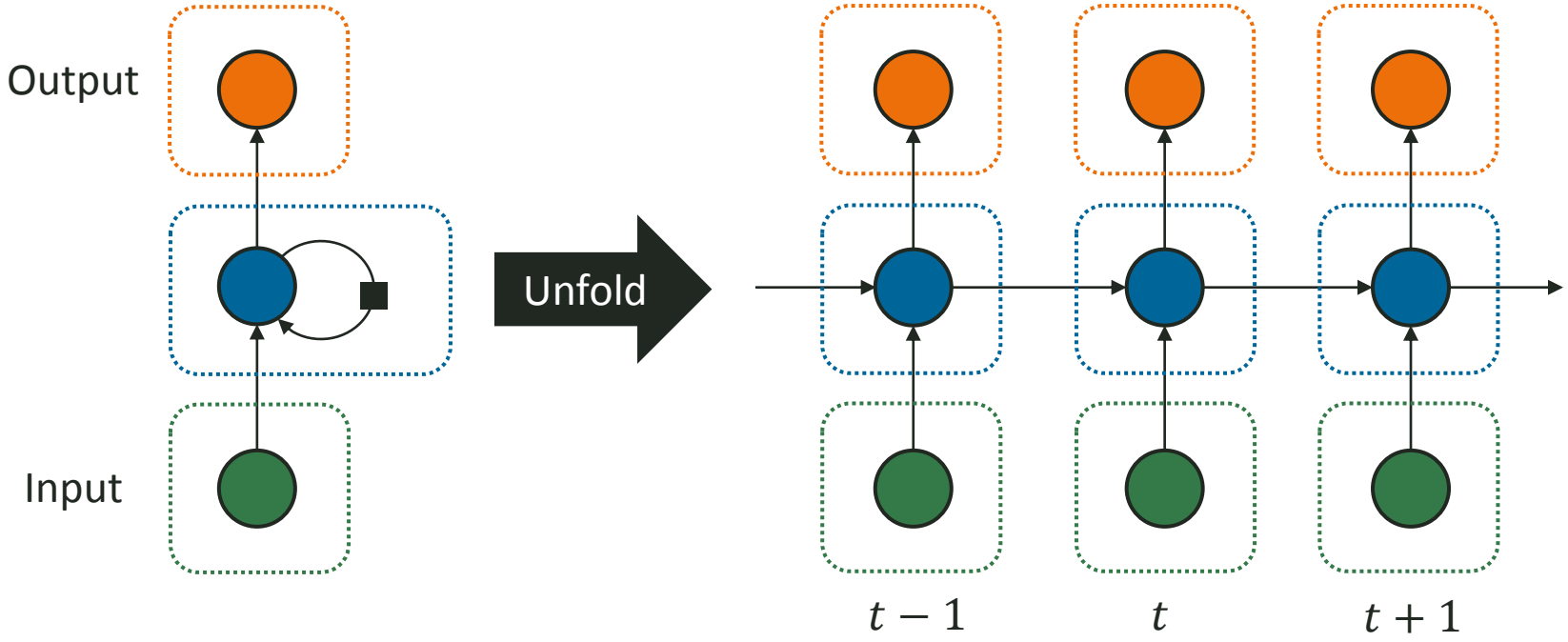
5/10/17

TIME SERIES



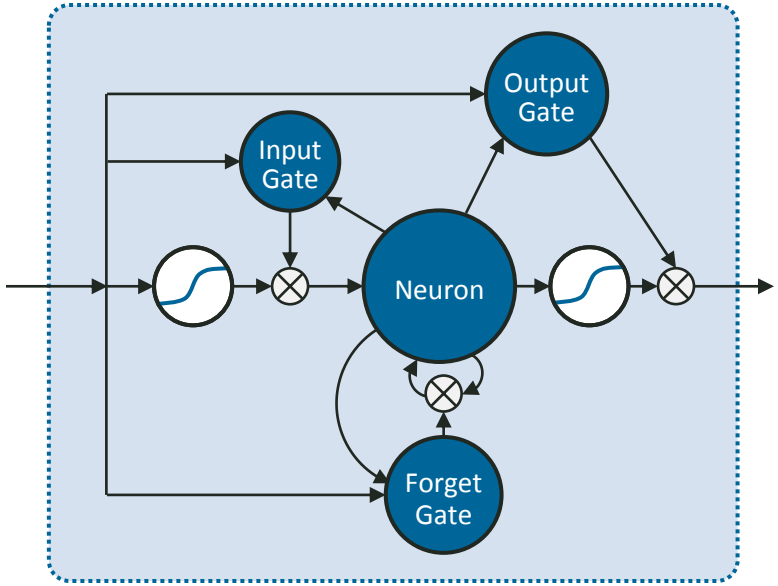
RECURRENT NEURAL NETWORKS

RECURRENT NEURAL NETWORK (RNN)



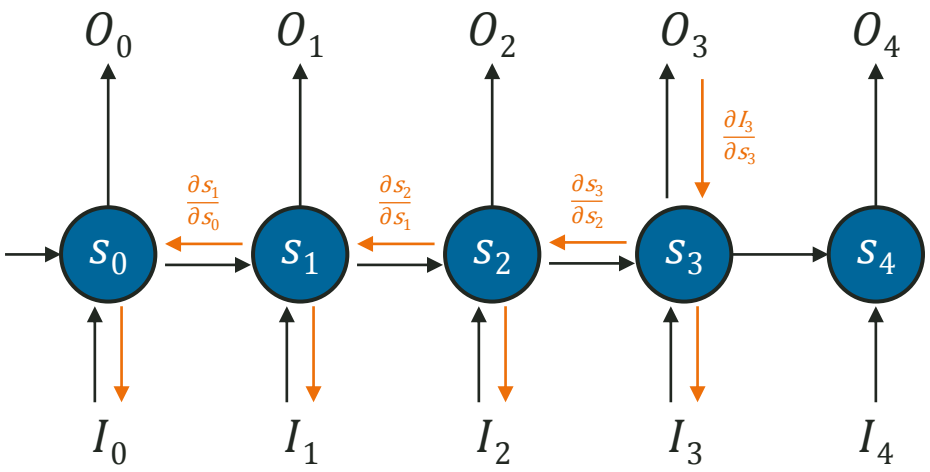
LONG SHORT TERM MEMORY (LSTM)

- Each cell is comprised of four elements:
  1. **Input Gate** – Determines how a new value is added to the memory
  2. **Forget Gate** – Determines how a value remains in memory
  3. **Output Gate** – Determines how the value in memory affects the output
  4. **Neuron** – with self-recurrent connection



LSTM VS BASIC RNN

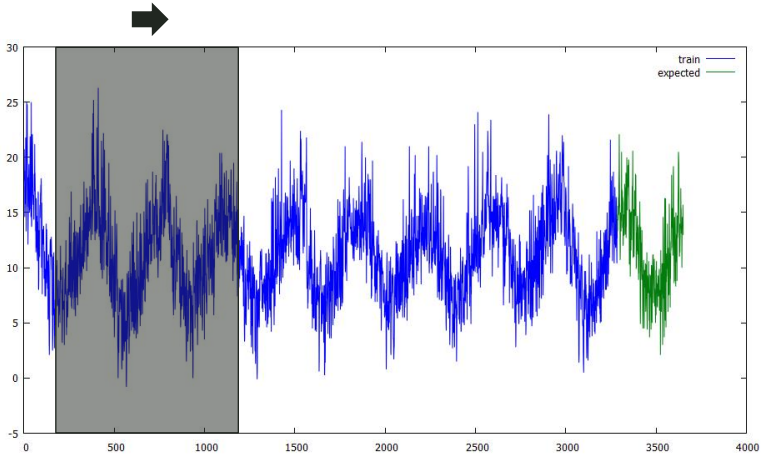
- Both can use standard methods such as back propagation through time
- In basic RNNs, error drops off exponentially
- LSTMs trap error in memory allowing error to be continually fed back until prior cells train well enough to address it



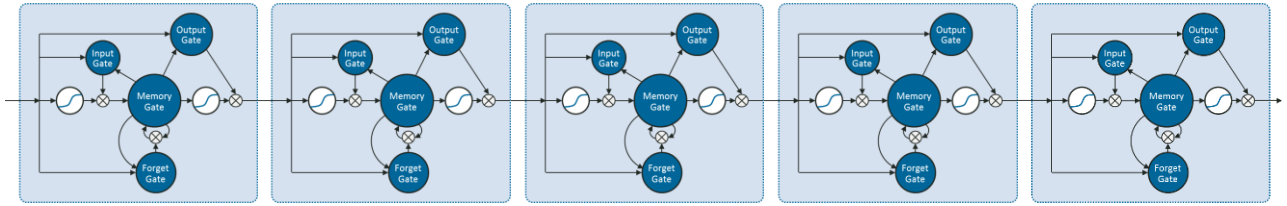
$I = Input$
$O = Output$

PARAMETER SPACE

- Look Back

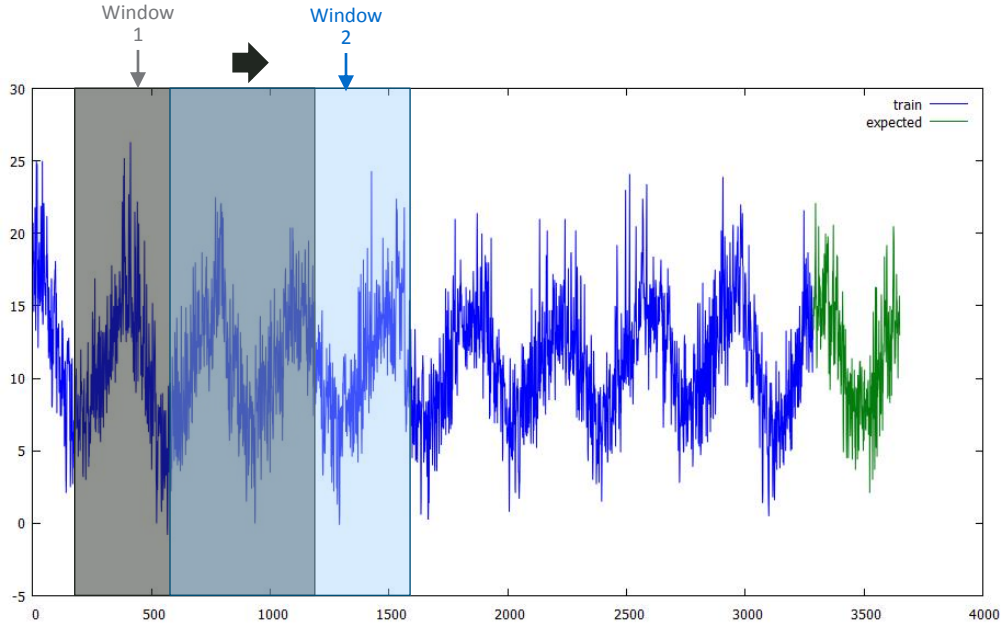


- Number of cells in LSTM layer



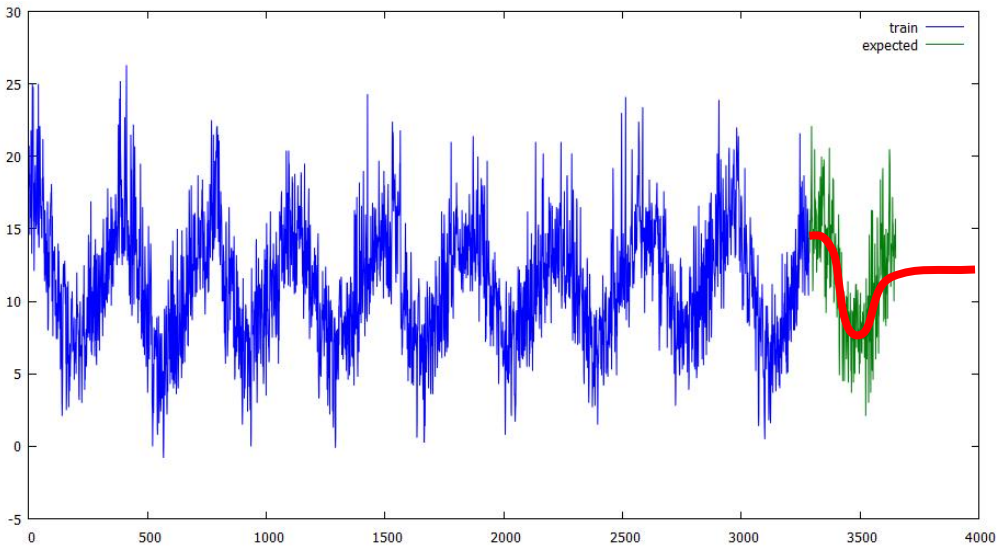
BATCH TRAINING

- Train in mini-batches on independent windows of the training data
- Each window is trained in parallel
- Parameters gained from each sections are averaged to produce a final parameter set
- Takes advantage of parallelism of GPUs



RECURSIVE PREDICTION

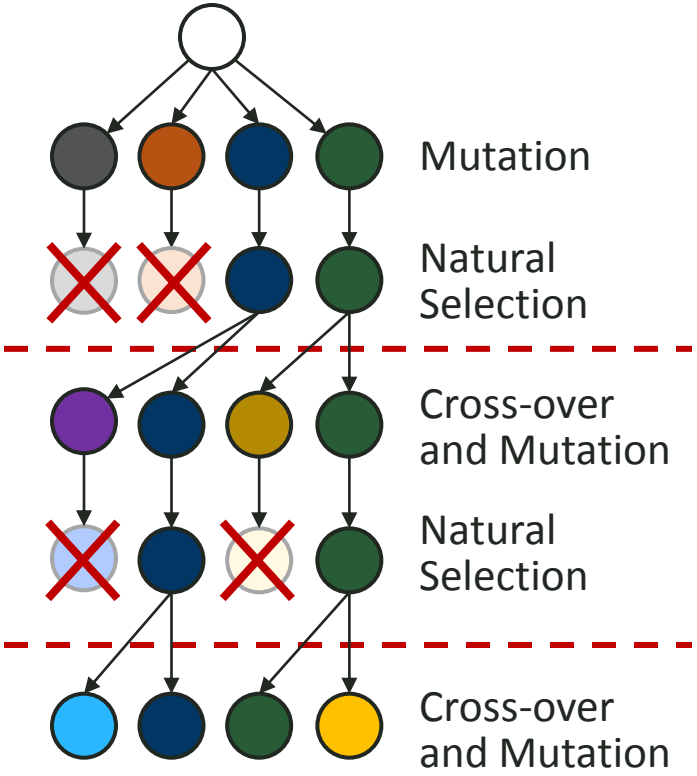
- RNNs predict next time step
- To predict further, one must use resulting output as part of the next input sample
- Error will accumulate as further predictions are made



GENETIC ALGORITHMS

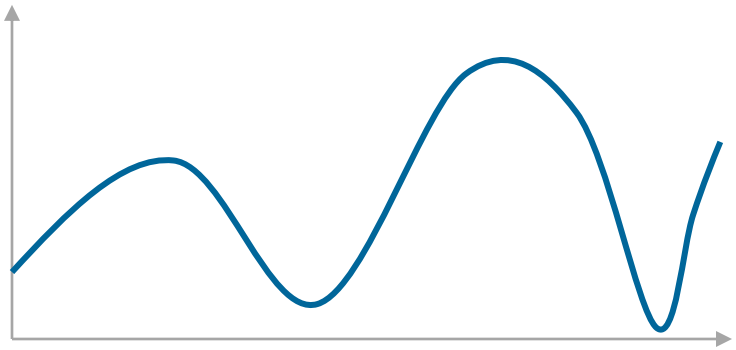
GENETIC ALGORITHMS (GA)

- Heuristic approach to searching a parameter space for a (near) optimal solution
- Modeled on evolution
  - Create a set of solutions called a generation
  - Test all elements of the generation to determine the best solutions
  - Create a new generation through cross-over and mutation of best solutions
  - Repeat



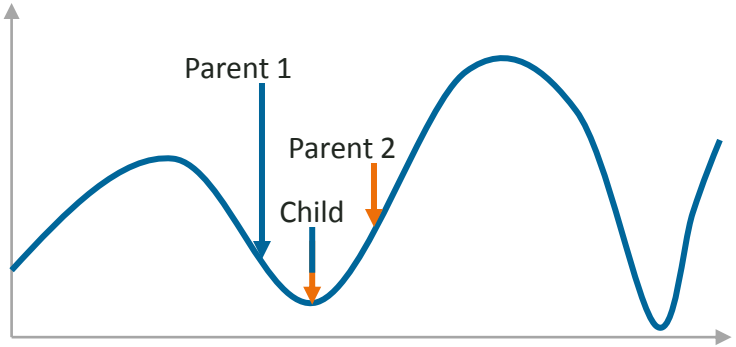
FITNESS FUNCTION

- Determines which candidate solutions are best
- Root Mean Square Error (RMSE) $\sqrt{\mathrm{E}((\hat{\theta} - \theta)^2)}$
- Penalty for larger networks



CROSS-OVER

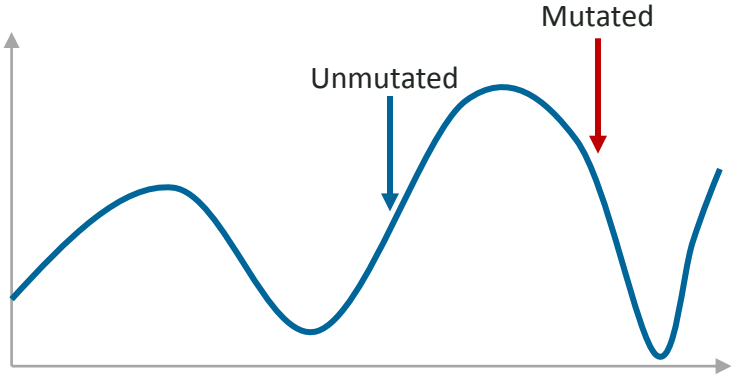
- Combines the genes from multiple parents
- Randomly selects genes from second parent to overwrite those of the first parent as a given rate



Parent 1	1	1	1	0	1	0	0	1	0	1
Parent 2	0	1	0	0	0	1	0	1	1	0
Child	1	1	0	0	1	1	0	1	0	1

MUTATION

- Genes are randomly altered at a small rate
- Enables movement away from local minima



1	1	0	0	1	1	0	1	0	1
1	1	0	**1**	1	1	0	1	**1**	1

IMPLEMENTATION

IMPLEMENTATION

- Framework
- LSTM Model
- GA Gene Representation
- GA Fitness Function

KERAS

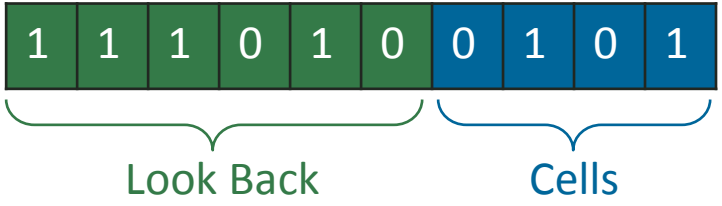
- Python package for neural networks
- Simple and easy usage enables rapid prototyping
- Used Theano backend

MODEL CREATION

- Tested various model architectures
- Used stateless LSTM as opposed to stateful LSTM to take advantage of GPU via mini-batches
- Achieve better results due to efficiency and the ability to perform more training

GENE REPRESENTATION

- Initially started with a decimal representation of parameters where mutations and crossovers would lead to discovering non-global minima



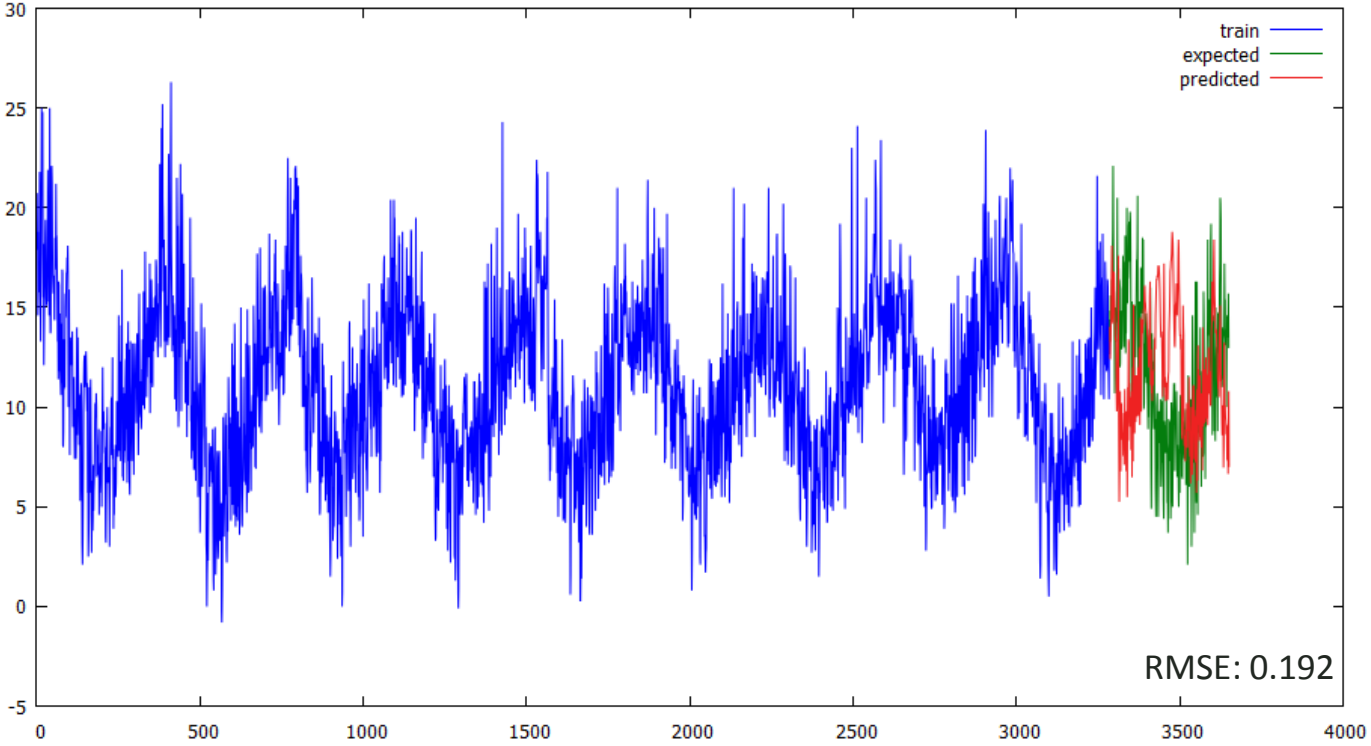
- Changed to binary representation to help solve this issue
- Integer parameters are represented as sum of 0-1 binary sequence
- Length of sequence determined by user-selected theoretical maximum

GENETIC ALGORITHM FITNESS FUNCTION

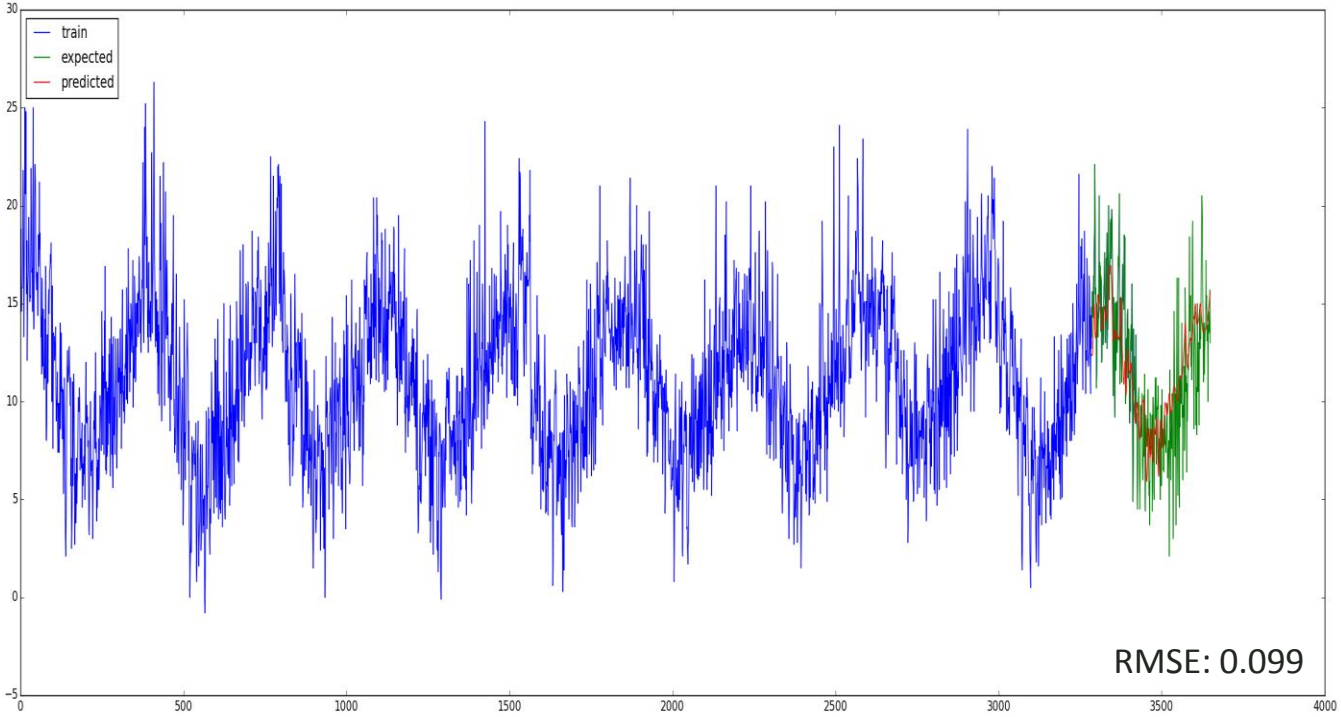
- Low training loss did not lead to being able to predict recursively
- Testing RMSE produced better results

RESULTS

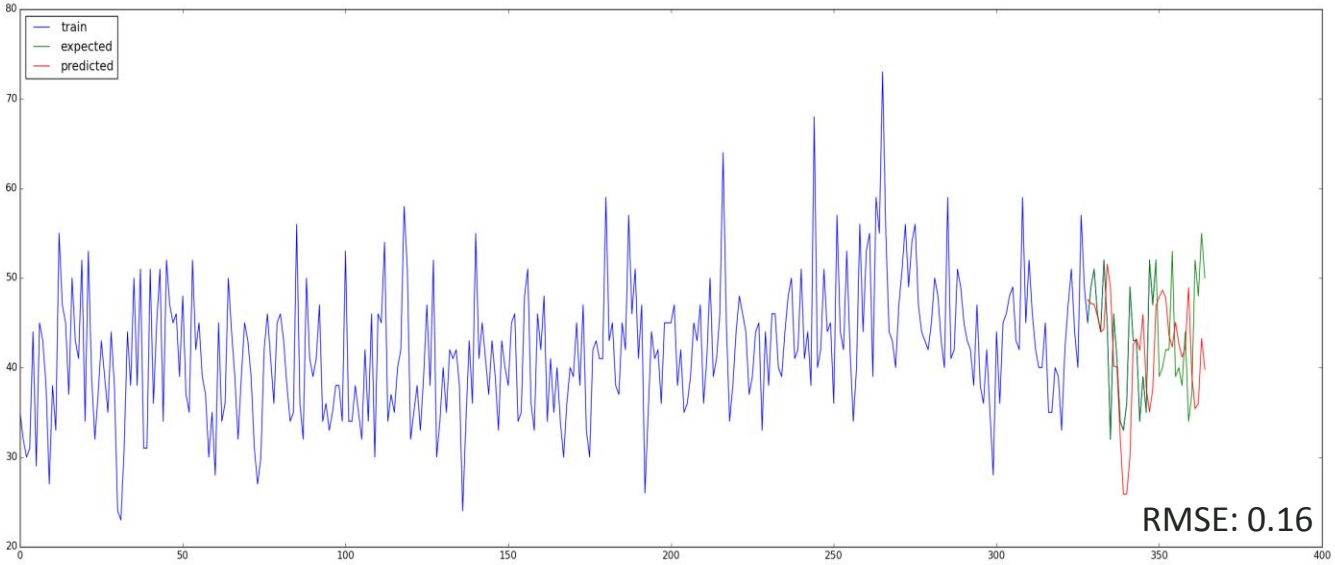
HAND TUNED PARAMETERS



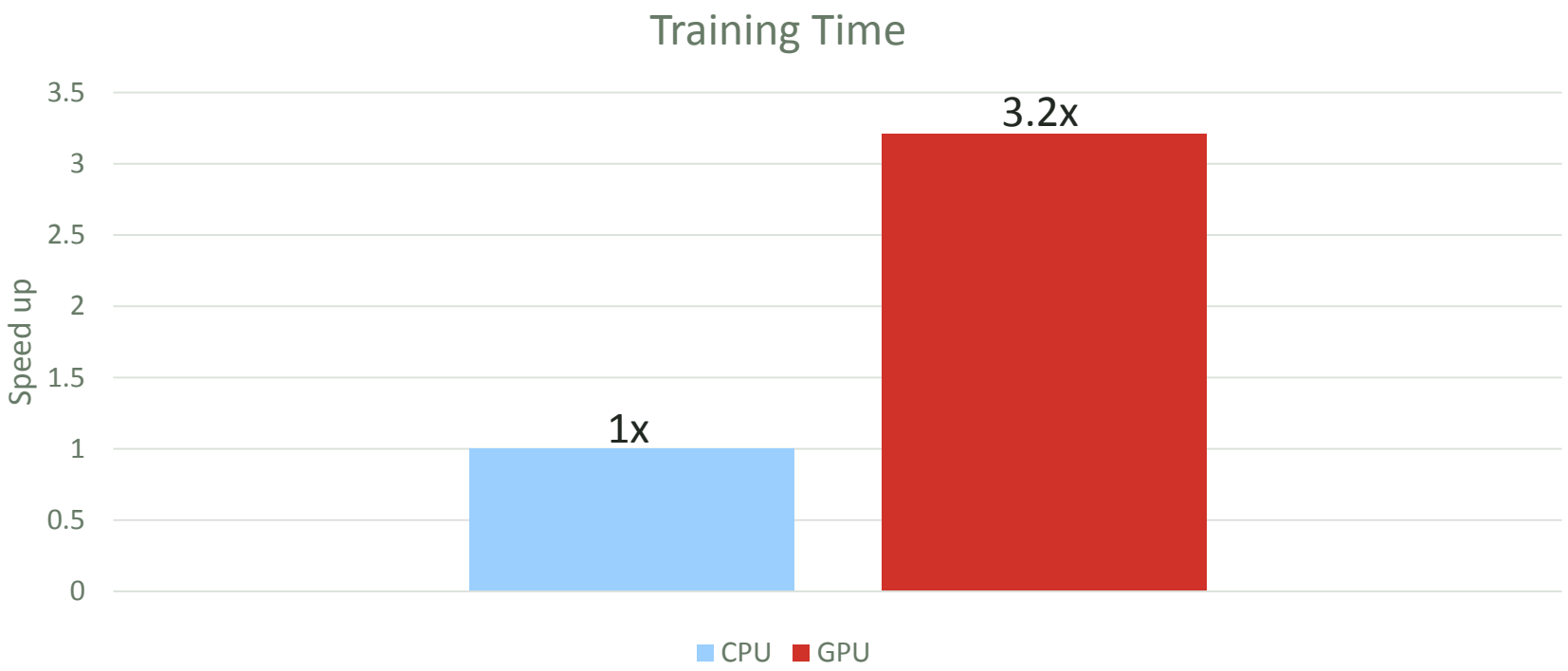
GENETIC ALGORITHM PARAMETERS



FEMALE BIRTHS



PERFORMANCE



FUTURE WORK

- Multi-GPU Parallelism
  - Each solution in the population can be trained independently
  - Training time is dependent on parameters, so speed up will be sub-linear
- Apply to General Neural Networks

CONCLUSION

- Genetic algorithm chose parameters for our LTSM network
- Produced better results than our hand tuning
- Would be useful for individuals that lack experience selecting parameters
- Requires further parallelization to be feasible for larger network parameter spaces

SPECIAL THANKS

Alex Lu (Junior Software Engineer)

CONTACT INFORMATION

Joe Schneible
Research Scientist
jschneible@technicacorp.com

For more information:

Technica Corporation
22970 Indian Creek Dr.
Suite 500
Dulles, VA 20166
703.662.2000
technicacorp.com

http://technicacorp.com/innovation

