

# Classifying Handwritten Digits with the Artificial Neural Network and Genetic Algorithm in a Distributed Environment

CSI 702 - High Performance Computing

Zach Firth    Stefan Novak

Thursday, May 6th, 2010

## Project Abstract

This study explores the application of the genetic algorithm and artificial neural network in performing classification of handwritten digits in a distributed environment. Message Passing Interface (MPI) is used in the modeling of the environment in which many configurations of the artificial neural network are instantiated while OpenMPI is used to increase the runtime performance of the artificial neural network. Although implementations of the serialized Backpropagation algorithm converge on a solution more rapidly than the distributed implementation, the backpropagation algorithm has a tendency to converge into local minima, requiring additional tuning parameters to avoid such behavior. A superposition of both backpropagation and genetic algorithms are implemented to solve this global optimization problem.

## Project Abstract - In short

- Solve the problem of recognizing handwritten digits.
- Use genetic algorithm (GA) and artificial neural network (ANN) in distributed environment.
- Use MPI to model genetic algorithm dynamics.
- Use OpenMP to speed up ANN processing.

## Project Abstract - In short

- Solve the problem of recognizing handwritten digits.
- Use genetic algorithm (GA) and artificial neural network (ANN) in distributed environment.
- Use MPI to model genetic algorithm dynamics.
- Use OpenMP to speed up ANN processing.

## Project Abstract - In short

- Solve the problem of recognizing handwritten digits.
- Use genetic algorithm (GA) and artificial neural network (ANN) in distributed environment.
- Use MPI to model genetic algorithm dynamics.
- Use OpenMP to speed up ANN processing.

## Project Abstract - In short

- Solve the problem of recognizing handwritten digits.
- Use genetic algorithm (GA) and artificial neural network (ANN) in distributed environment.
- Use MPI to model genetic algorithm dynamics.
- Use OpenMP to speed up ANN processing.

# Project Motivation

- Solve the problem of recognizing handwritten digits.
  - Use local and global optimization techniques.
  - Large data set → HPC!
- Use both MPI and OpenMP to increase accuracy and performance of system.
  - MPI used for genetic algorithm.
  - OpenMP used for artificial neural network.

# Project Motivation

- Solve the problem of recognizing handwritten digits.
  - Use local and global optimization techniques.
  - Large data set → HPC!
- Use both MPI and OpenMP to increase accuracy and performance of system.
  - MPI used for genetic algorithm.
  - OpenMP used for artificial neural network.

# Project Motivation

- Solve the problem of recognizing handwritten digits.
  - Use local and global optimization techniques.
  - Large data set → HPC!
- Use both MPI and OpenMP to increase accuracy and performance of system.
  - MPI used for genetic algorithm.
  - OpenMP used for artificial neural network.

# Project Motivation

- Solve the problem of recognizing handwritten digits.
  - Use local and global optimization techniques.
  - Large data set → HPC!
- Use both MPI and OpenMP to increase accuracy and performance of system.
  - MPI used for genetic algorithm.
  - OpenMP used for artificial neural network.

# Project Motivation

- Solve the problem of recognizing handwritten digits.
  - Use local and global optimization techniques.
  - Large data set → HPC!
- Use both MPI and OpenMP to increase accuracy and performance of system.
  - MPI used for genetic algorithm.
  - OpenMP used for artificial neural network.

# Project Motivation

- Solve the problem of recognizing handwritten digits.
  - Use local and global optimization techniques.
  - Large data set → HPC!
- Use both MPI and OpenMP to increase accuracy and performance of system.
  - MPI used for genetic algorithm.
  - OpenMP used for artificial neural network.

## Increasing GA Performance

- Computing a single generation of the GA could take over 1 minute.
- The more generations a GA is allowed to run, the better potential there is to find a global maxima.
- Distributing the GA allows for a larger population to be used (Since its distributed among multiple nodes).
- Populations can move toward an optimal solution independently leading to a more diverse population.

## Increasing GA Performance

- Computing a single generation of the GA could take over 1 minute.
- The more generations a GA is allowed to run, the better potential there is to find a global maxima.
- Distributing the GA allows for a larger population to be used (Since its distributed among multiple nodes).
- Populations can move toward an optimal solution independently leading to a more diverse population.

## Increasing GA Performance

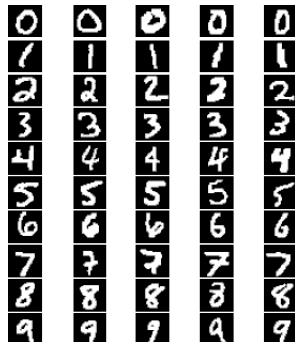
- Computing a single generation of the GA could take over 1 minute.
- The more generations a GA is allowed to run, the better potential there is to find a global maxima.
- Distributing the GA allows for a larger population to be used (Since its distributed among multiple nodes).
- Populations can move toward an optimal solution independently leading to a more diverse population.

## Increasing GA Performance

- Computing a single generation of the GA could take over 1 minute.
- The more generations a GA is allowed to run, the better potential there is to find a global maxima.
- Distributing the GA allows for a larger population to be used (Since its distributed among multiple nodes).
- Populations can move toward an optimal solution independently leading to a more diverse population.

# Overview of the MNIST Handwritten Digit Database

- Handwritten digits with  $28 \times 28$  resolution with 256 bit-depth.
- 60,000 training examples and 10,000 test examples.
- Standard data set to test new computational learning algorithms.



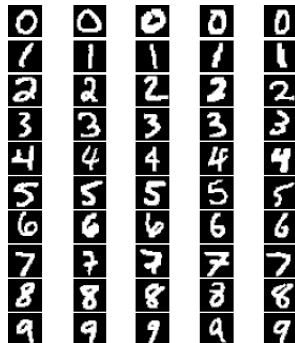
- Image source:

[http://www.delas.it/public/projects/hdr/relazione/img/example\\_mnist.gif](http://www.delas.it/public/projects/hdr/relazione/img/example_mnist.gif)

# Overview of the MNIST Handwritten Digit Database

- Handwritten digits with  $28 \times 28$  resolution with 256 bit-depth.
- 60,000 training examples and 10,000 test examples.

- Standard data set to test new computational learning algorithms.

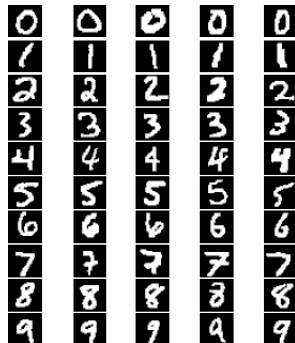


- Image source:

[http://www.delas.it/public/projects/hdr/relazione/img/example\\_mnist.gif](http://www.delas.it/public/projects/hdr/relazione/img/example_mnist.gif)

# Overview of the MNIST Handwritten Digit Database

- Handwritten digits with  $28 \times 28$  resolution with 256 bit-depth.
- 60,000 training examples and 10,000 test examples.
- Standard data set to test new computational learning algorithms.

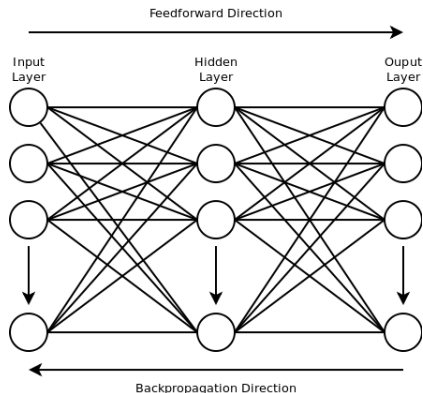


- Image source:

[http://www.delas.it/public/projects/hdr/relazione/img/example\\_mnist.gif](http://www.delas.it/public/projects/hdr/relazione/img/example_mnist.gif)

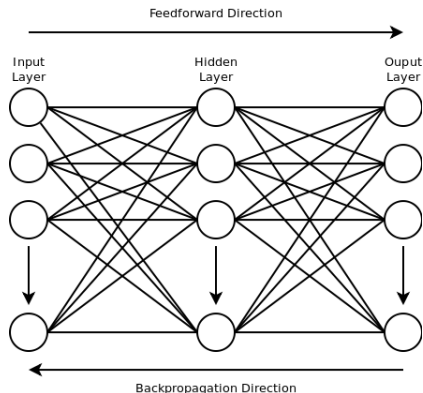
# Overview of the Artificial Neural Networks

- General algorithm to approximate complex functions.
- Build complex network of linear combinations of relationships to translate input to output.
- Each connection (weight) is optimized to represent target function.



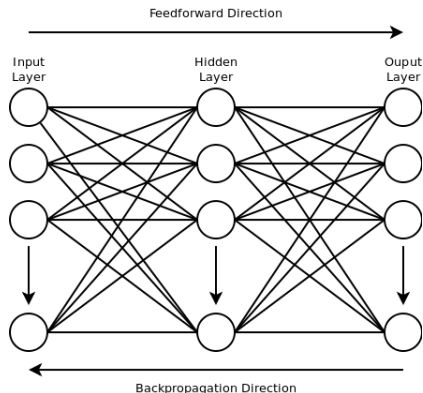
# Overview of the Artificial Neural Networks

- General algorithm to approximate complex functions.
- Build complex network of linear combinations of relationships to translate input to output.
- Each connection (weight) is optimized to represent target function.



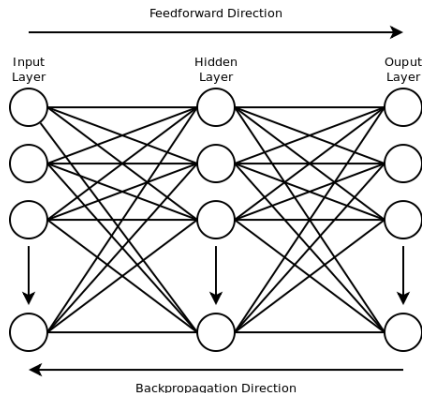
# Overview of the Artificial Neural Networks

- General algorithm to approximate complex functions.
- Build complex network of linear combinations of relationships to translate input to output.
- Each connection (weight) is optimized to represent target function.



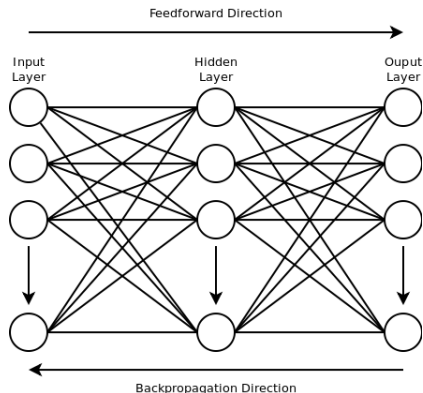
## Overview of the Artificial Neural Networks, Cont.

- Backpropagation algorithm uses gradient descent to solve for optimal weight configuration.
- Often gets stuck in local minima.
- GA can provide opportunity for global optimization.



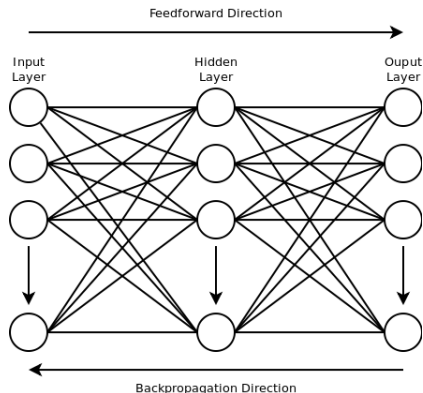
## Overview of the Artificial Neural Networks, Cont.

- Backpropagation algorithm uses gradient descent to solve for optimal weight configuration.
- Often gets stuck in local minima.
- GA can provide opportunity for global optimization.



## Overview of the Artificial Neural Networks, Cont.

- Backpropagation algorithm uses gradient descent to solve for optimal weight configuration.
- Often gets stuck in local minima.
- GA can provide opportunity for global optimization.



# Genetic Algorithm Definitions

- 1 Generation
- 2 Population
- 3 Chromosomes
- 4 Mutations
- 5 Crossover

# Genetic Algorithm Definitions

- 1 Generation
- 2 Population
- 3 Chromosomes
- 4 Mutations
- 5 Crossover

# Genetic Algorithm Definitions

- 1 Generation
- 2 Population
- 3 Chromosomes
- 4 Mutations
- 5 Crossover

# Genetic Algorithm Definitions

- 1 Generation
- 2 Population
- 3 Chromosomes
- 4 Mutations
- 5 Crossover

# Genetic Algorithm Definitions

- 1 Generation
- 2 Population
- 3 Chromosomes
- 4 Mutations
- 5 Crossover

# Stages of a serial GA

- 1 Population creation
- 2 Fitness Evaluation
- 3 Selection
- 4 Crossover and Mutation
- 5 Repeat steps 2-4

# Stages of a serial GA

- 1 Population creation
- 2 Fitness Evaluation
- 3 Selection
- 4 Crossover and Mutation
- 5 Repeat steps 2-4

# Stages of a serial GA

- 1 Population creation
- 2 Fitness Evaluation
- 3 Selection
- 4 Crossover and Mutation
- 5 Repeat steps 2-4

# Stages of a serial GA

- 1 Population creation
- 2 Fitness Evaluation
- 3 Selection
- 4 Crossover and Mutation
- 5 Repeat steps 2-4

# Stages of a serial GA

- 1 Population creation
- 2 Fitness Evaluation
- 3 Selection
- 4 Crossover and Mutation
- 5 Repeat steps 2-4

## Translation to a parallel algorithm

- 1 Multiple populations created.
- 2 Fitness independently computed.
- 3 Independent selection, crossover and mutation.
- 4 Periodic "Genetic Transfer" between nodes.
- 5 Repeat steps 2-4.

## Translation to a parallel algorithm

- 1 Multiple populations created.
- 2 Fitness independently computed.
- 3 Independent selection, crossover and mutation.
- 4 Periodic "Genetic Transfer" between nodes.
- 5 Repeat steps 2-4.

## Translation to a parallel algorithm

- 1 Multiple populations created.
- 2 Fitness independently computed.
- 3 Independent selection, crossover and mutation.
- 4 Periodic "Genetic Transfer" between nodes.
- 5 Repeat steps 2-4.

## Translation to a parallel algorithm

- 1 Multiple populations created.
- 2 Fitness independently computed.
- 3 Independent selection, crossover and mutation.
- 4 Periodic "Genetic Transfer" between nodes.
- 5 Repeat steps 2-4.

## Translation to a parallel algorithm

- 1 Multiple populations created.
- 2 Fitness independently computed.
- 3 Independent selection, crossover and mutation.
- 4 Periodic "Genetic Transfer" between nodes.
- 5 Repeat steps 2-4.

## Integrating MNIST Database with ANN

- ANN modeled as structure in C.
  - Inputs, Outputs, and Weights are pointers to global data.
  - Allows for more flexible data manipulation.
- Example:

### Stepping through training data

```
for(i=0; i<10; i++)  
{  
    network.systemInput = trainingData[i];  
    network.targetOutput = trainingLabels[i];  
    PropagateInputsForward(&network);  
    PropagateErrorBackwards(&network);  
}
```

## Integrating MNIST Database with ANN

- ANN modeled as structure in C.
  - Inputs, Outputs, and Weights are pointers to global data.
  - Allows for more flexible data manipulation.
- Example:

### Stepping through training data

```
for(i=0; i<10; i++)  
{  
    network.systemInput = trainingData[i];  
    network.targetOutput = trainingLabels[i];  
    PropagateInputsForward(&network);  
    PropagateErrorBackwards(&network);  
}
```

## Integrating ANN with GA

- Same flexibility is applied to interactions between ANN and GA.
  - Single ANN instance can be used to represent multiple instance in population.
- Example:

### Stepping through members of population

```
for(i = 0;i<(ga.popSize);i++)  
{  
    network.weights = &myNewPopulation[i*(g.numWeights)];  
    network.systemInput = trainingData[trainingDataID];  
    network.targetOutput = trainingLabels[trainingDataID];  
    PropagateInputsForward(&network);  
    PropagateErrorBackwards(&network);  
}
```

## Integrating ANN with GA

- Same flexibility is applied to interactions between ANN and GA.
  - Single ANN instance can be used to represent multiple instance in population.
- Example:

### Stepping through members of population

```
for(i = 0; i < (ga.popSize); i++)  
{  
    network.weights = &myNewPopulation[i*(g.numWeights)];  
    network.systemInput = trainingData[trainingDataID];  
    network.targetOutput = trainingLabels[trainingDataID];  
    PropagateInputsForward(&network);  
    PropagateErrorBackwards(&network);  
}
```

# Integrating GA With MPI

- MPI used to facilitate merging of chromosome.
- Chromosomes are shared in round-robin fashion.
- Book-keeping methods prevent duplicate chromosome distributions.

# Integrating GA With MPI

- MPI used to facilitate merging of chromosome.
- Chromosomes are shared in round-robin fashion.
- Book-keeping methods prevent duplicate chromosome distributions.

# Integrating GA With MPI

- MPI used to facilitate merging of chromosome.
- Chromosomes are shared in round-robin fashion.
- Book-keeping methods prevent duplicate chromosome distributions.

## Speeding Up ANN Processing With OpenMP

- GA requires fitness function after each iteration.
- ANN can verify current weight configuration again 10,000 test cases.
  - Biggest bottleneck in code!
- Can use OpenMP to parallelize success rate calculation via summation reduction operator.

## Speeding Up ANN Processing With OpenMP

- GA requires fitness function after each iteration.
- ANN can verify current weight configuration again 10,000 test cases.
  - Biggest bottleneck in code!
- Can use OpenMP to parallelize success rate calculation via summation reduction operator.

## Speeding Up ANN Processing With OpenMP

- GA requires fitness function after each iteration.
- ANN can verify current weight configuration again 10,000 test cases.
  - Biggest bottleneck in code!
- Can use OpenMP to parallelize success rate calculation via summation reduction operator.

## Speeding Up ANN Processing With OpenMP

- GA requires fitness function after each iteration.
- ANN can verify current weight configuration again 10,000 test cases.
  - Biggest bottleneck in code!
- Can use OpenMP to parallelize success rate calculation via summation reduction operator.

# Results

- A Python script was written to run a parameter scan over:
  - Number of MPI nodes
  - Number of OpenMP threads
  - Probability of Crossover
  - Probability of Mutation
  - Backpropagation Learning Rate

# Results

- A Python script was written to run a parameter scan over:
  - Number of MPI nodes
  - Number of OpenMP threads
  - Probability of Crossover
  - Probability of Mutation
  - Backpropagation Learning Rate

# Results

- A Python script was written to run a parameter scan over:
  - Number of MPI nodes
  - Number of OpenMP threads
  - Probability of Crossover
  - Probability of Mutation
  - Backpropagation Learning Rate

# Results

- A Python script was written to run a parameter scan over:
  - Number of MPI nodes
  - Number of OpenMP threads
  - Probability of Crossover
  - Probability of Mutation
  - Backpropagation Learning Rate

# Results

- A Python script was written to run a parameter scan over:
  - Number of MPI nodes
  - Number of OpenMP threads
  - Probability of Crossover
  - Probability of Mutation
  - Backpropagation Learning Rate

# Results

- A Python script was written to run a parameter scan over:
  - Number of MPI nodes
  - Number of OpenMP threads
  - Probability of Crossover
  - Probability of Mutation
  - Backpropagation Learning Rate

# Results

- Preliminary results show:
  - Number of generations computed increased linearly with number of nodes.
  - When running OpenMP locally, significant improvements.
  - When running OpenMP on cluster, performance decreased.
  - Need to make additional submissions to the cluster to have a definitive understanding of the dynamics.

# Results

- Preliminary results show:
  - Number of generations computed increased linearly with number of nodes.
  - When running OpenMP locally, significant improvements.
  - When running OpenMP on cluster, performance decreased.
  - Need to make additional submissions to the cluster to have a definitive understanding of the dynamics.

# Results

- Preliminary results show:
  - Number of generations computed increased linearly with number of nodes.
  - When running OpenMP locally, significant improvements.
  - When running OpenMP on cluster, performance decreased.
  - Need to make additional submissions to the cluster to have a definitive understanding of the dynamics.

# Results

- Preliminary results show:
  - Number of generations computed increased linearly with number of nodes.
  - When running OpenMP locally, significant improvements.
  - When running OpenMP on cluster, performance decreased.
  - Need to make additional submissions to the cluster to have a definitive understanding of the dynamics.

# Results

- Preliminary results show:
  - Number of generations computed increased linearly with number of nodes.
  - When running OpenMP locally, significant improvements.
  - When running OpenMP on cluster, performance decreased.
  - Need to make additional submissions to the cluster to have a definitive understanding of the dynamics.

## Technical Challenges

- At the beginning of production runs, GMICE cluster was unavailable due to maintenance.
- As a potential fall back option we explored using Amazon E2C for production runs.
  - ElasticWulf Project: <http://code.google.com/p/elasticwulf/>
  - Unfortunately lack of current documentation and time constraints made implementation impractical.
  - Difficulty in automatically syncing up data across EC2 instances via S3 buckets.

## Technical Challenges

- At the beginning of production runs, GMICE cluster was unavailable due to maintenance.
- As a potential fall back option we explored using Amazon E2C for production runs.
  - ElasticWulf Project: <http://code.google.com/p/elasticwulf/>
  - Unfortunately lack of current documentation and time constraints made implementation impractical.
  - Difficulty in automatically syncing up data across EC2 instances via S3 buckets.

## Technical Challenges

- At the beginning of production runs, GMICE cluster was unavailable due to maintenance.
- As a potential fall back option we explored using Amazon E2C for production runs.
  - ElasticWulf Project: <http://code.google.com/p/elasticwulf/>
  - Unfortunately lack of current documentation and time constraints made implementation impractical.
  - Difficulty in automatically syncing up data across EC2 instances via S3 buckets.

## Technical Challenges

- At the beginning of production runs, GMICE cluster was unavailable due to maintenance.
- As a potential fall back option we explored using Amazon E2C for production runs.
  - ElasticWulf Project: <http://code.google.com/p/elasticwulf/>
  - Unfortunately lack of current documentation and time constraints made implementation impractical.
  - Difficulty in automatically syncing up data across EC2 instances via S3 buckets.

## Technical Challenges

- At the beginning of production runs, GMICE cluster was unavailable due to maintenance.
- As a potential fall back option we explored using Amazon E2C for production runs.
  - ElasticWulf Project: <http://code.google.com/p/elasticwulf/>
  - Unfortunately lack of current documentation and time constraints made implementation impractical.
  - Difficulty in automatically syncing up data across EC2 instances via S3 buckets.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Improve fidelity of GA.
  - Use of tabu list.
  - Use of dynamic parameters.
- Improve complexity of ANN.
  - Increase hidden layer node count.
  - Increase number of layers.
- Investigate scalability of problem.
  - Increase data set size by interpolating MNIST data.
  - Increase population size in GA.

## Future Work

- Explore additional parameter configuration on GMICE.
- Explore large scale deployment on Amazon EC2.
- Stress test memory requirements by increase data set resolution ( $100 \times 100$  grid)

## Future Work

- Explore additional parameter configuration on GMICE.
- Explore large scale deployment on Amazon EC2.
- Stress test memory requirements by increase data set resolution ( $100 \times 100$  grid)

## Future Work

- Explore additional parameter configuration on GMICE.
- Explore large scale deployment on Amazon EC2.
- Stress test memory requirements by increase data set resolution ( $100 \times 100$  grid)

# Conclusion

- Ambitious project to explore combination of local and global optimization techniques.
- MPI can be used to model genetic algorithm dynamics in a distributed environment.
- OpenMP is used successfully to increase performance runtime for artificial neural networks.

## Conclusion

- Ambitious project to explore combination of local and global optimization techniques.
- MPI can be used to model genetic algorithm dynamics in a distributed environment.
- OpenMP is used successfully to increase performance runtime for artificial neural networks.

## Conclusion

- Ambitious project to explore combination of local and global optimization techniques.
- MPI can be used to model genetic algorithm dynamics in a distributed environment.
- OpenMP is used successfully to increase performance runtime for artificial neural networks.