Evolving Autoencoders

Christopher Ross   Jonathan Brant   Zak Roessler

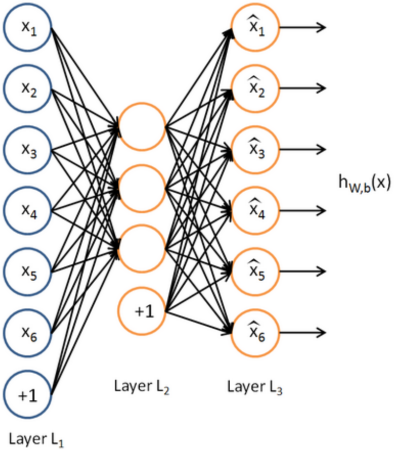
November 1, 2015

Introduction

- Image Processing/Classification
    1. Each pixel is a separate input
    2. ANN inputs grows exponentially as resolution increases
    3. Method of dimensionality reduction needed
- Autoencoders learns compressed representation
- Traditionally, autoencoder hidden layer topology has been hand-crafted
- Our approach evolves autoencoder structure that can then be optimized
    1. Connection weights between input/hidden and hidden/output substrate evolved using HyperNEAT
    2. Backpropagation optimizes evolved autoencoder

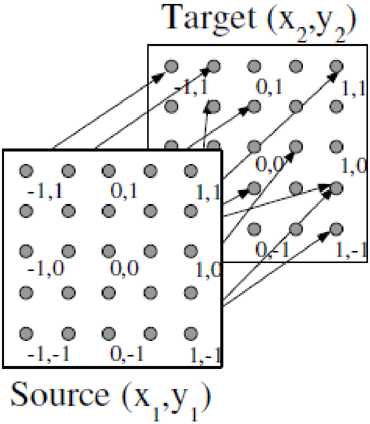
Autoencoders

- Autoencoders attempt to extract most salient features from visible layer
- Typical layers include:
  1. Input (visible) layer where uncompressed features are received
  2. Hidden layer where compressed features are stored
  3. Output layer where reconstruction error is calculated
- Most often trained using backpropagation with the intent of minimizing reconstruction error



HyperNEAT

- Indirect encoding substantially reduces search space
  - We're no longer evolving connection weight for each pixel individually
- Substrate configuration hand-designed to capture domain geometry
  - State-space sandwich substrate ideal for visual mapping
- Ability to scale with little-to-no loss of functionality
  - Ideal for image processing at varying resolutions

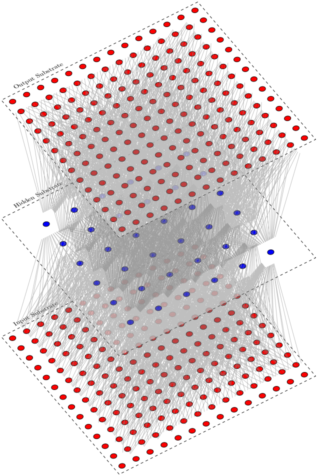


Evolution of Autoencoders

- Two-phase process:
  1. HyperNEAT evolves initial autoencoder connection weights
  2. Backpropagation fine tunes autoencoder weights
- Results will be analyzed:
  1. Quantitatively - reduction of reconstruction error
  2. Qualitatively - how closely does reconstructed image resemble the original
- MNIST handwritten digit dataset will be used for training/validation

Substrate Configuration

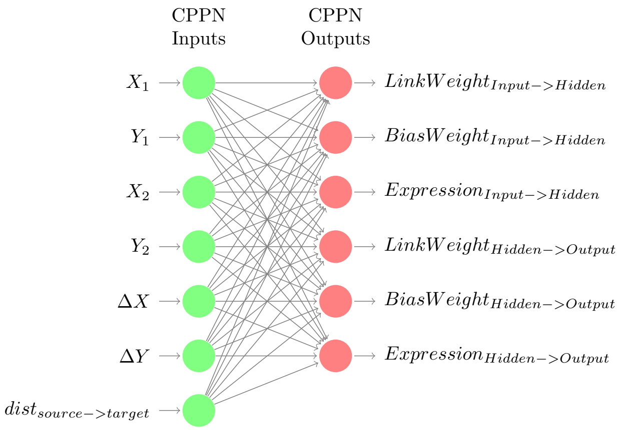
- Three-tiered state-space sandwich substrate
  1. Input sheet - all points in 2-dimensional image (size varies based on image resolution)
  2. Hidden sheet - compressed feature vector
  3. Output sheet - reconstructed image at original resolution



Minimal CPPN

- CPPN inputs are cartesian coordinates between source and target sheet
  - X_1 and Y_1 is point on source sheet
  - X_2 and Y_2 is point on target sheet
  - ΔX and ΔY are difference between X and Y components on source and target sheet
  - $dist_{source->target}$ is euclidean distance between source and target point
- CPPN queries the substrate twice:
  - Once for connection weight, bias weight, and expression threshold between input and hidden sheet
  - Once for connection weight, bias weight, and expression threshold between hidden and output sheet

Minimal CPPN

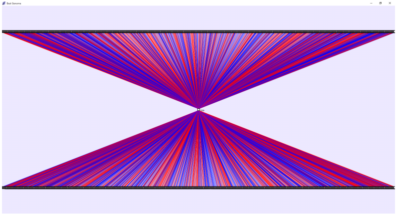


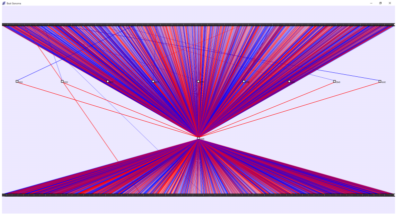
Experiment Configuration

Parameter	Value	Description
Training Sample Proportion	80%	Percentage of sample images used for training
Number of Backpropagation Iterations	100	Number of backpropagation iterations performed during training
Learning Rate	1	Learning Rate (to control backpropagation convergence speed)
Image Resolution Reduction Factor	2	Reduction factor of source image (i.e. a 28x28 image becomes 14x14 with a reduction factor of 2)
MNIST Digit	1	MNIST handwritten digit dataset image used

Current State

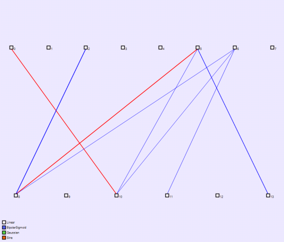
- Standard NEAT initially attempted
  - Slow evolution due to massive search space
  - Quantitative reconstruction accuracy rarely exceeded 60%
  - Prone to getting stuck in local optima, yielding to minimal improvements

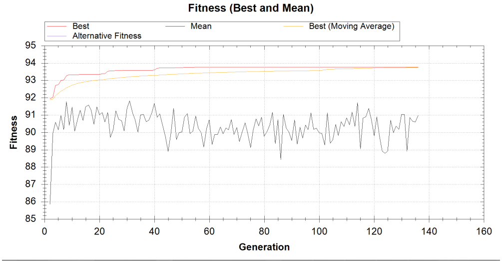




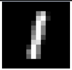
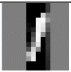
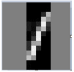
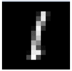
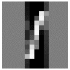
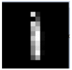
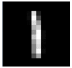
Current State - HyperNEAT

- HyperNEAT achieved much better reconstruction accuracy
  - Attained over 90% accuracy in a couple of generations
- Matched source images quite closely from a qualitative standpoint





Current State - HyperNEAT

Source	Reconstructed	Set	Index
		Training	30
		Training	60
		Validation	85
		Validation	95