Evolving Deep Neural Networks for Text Prediction and Image Classification

Checkpoint Demo

Gabriel Meyer-Lee
Harsha Uppili
Alan Zhuolun Zhao

Introduction & Motivation

- Traditional DNN have few rules to guide topology design and hyperparameter choices
- NEAT does not require these choices but is limited in the depth and complexity of the network it can produce.
    - K. Stanley → HyperNEAT
    - R. Mikkulainen → CoDeepNEAT
- Our goal is to extend the capacities of evolutionary algorithms such as NEAT and its variations (CoDeepNEAT specifically)
    - Many methods have emerged recently for optimizing topology and hyperparameters but tend to be very computationally expensive
    - Our intent is to preserve high levels of accuracy while reducing computational costs

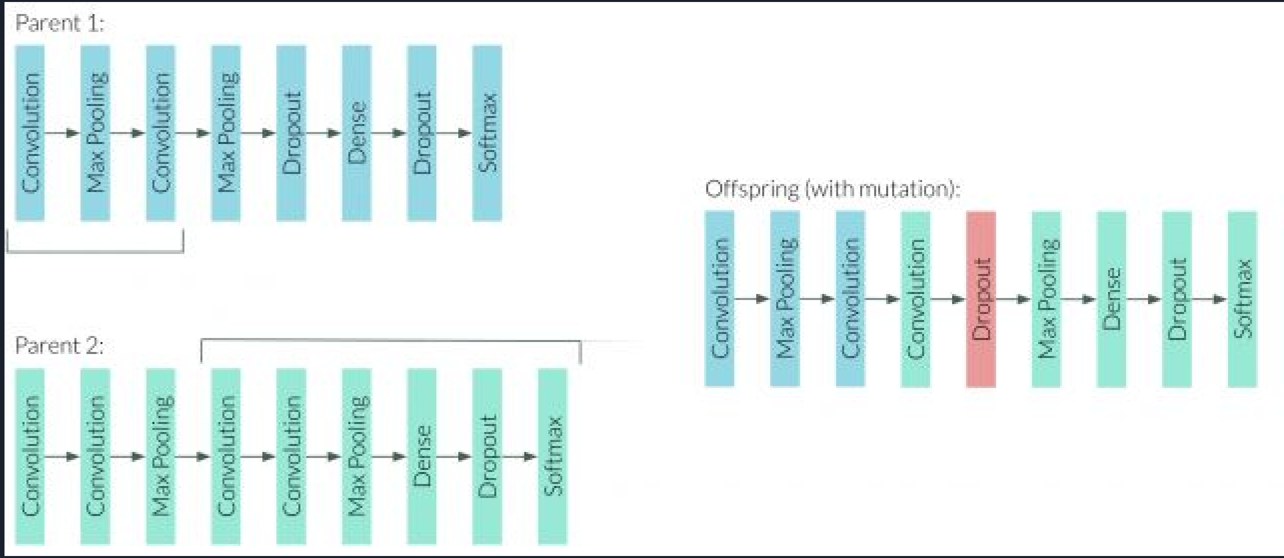
CoDeepNEAT

- DeepNEAT
    - Layers instead of individual neurons
    - Connections and weights aren’t evolved any more, just trained
- CoDeepNEAT
    - Implements hierarchical design
        - Hierarchical SANE
    - Evolve two populations simultaneously
        - Blueprint chromosomes encode overall network structure
        - Module chromosomes encode the makeup of small sets of layers which can be stacked to create a DNN
    - Fitness of modules is based on the average fitness of the networks they’re included in

Improving upon CoDeepNEAT

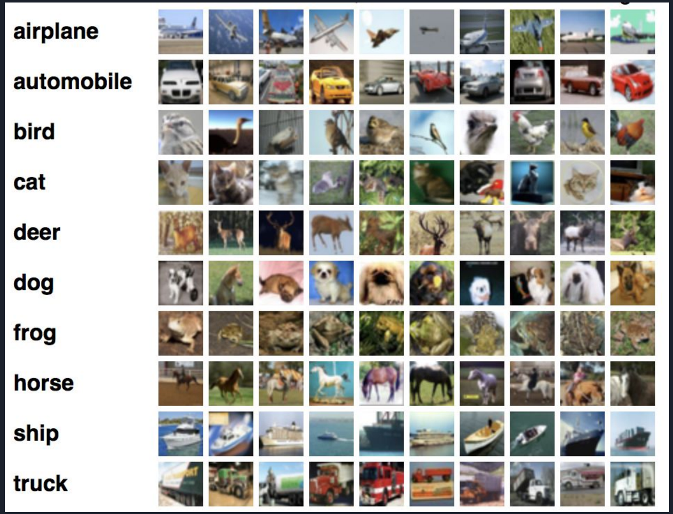
- Lamarckian Evolution/Epigenetics
    - A simple way to speed up the time it takes to evolve a Deep Neural Network is to not train the DNNs from scratch every generation but to preserve the connection weights between generations, passing them from parent to child
- Fitness function
    - Assigning of the lower evolutionary level (modules) based on the average fitness of the DNNs they participated is a fairly crude approximation of that module's fitness
    - We seek to improve this approximation through Multi-Objective optimization (InfoMAX) and more sophisticated statistical use of the available data on the module's performance (Bayesian Optimization)

Starting Point - DEvol



Experiment - Image Classification

- CIFAR-10
  - Dataset of 50,000 32x32 color training images, labeled over 10 categories, and 10,000 test images.
- Relatively expensive
  - ~20s per epoch on a GTX 1080
  - ~10 epochs per model
  - ~20 models per generation
  - ~20 generations -> 80000s -> 22 hours
- Ways to make it less expensive
  - Parallelize the training tasks
  - Inherited weights



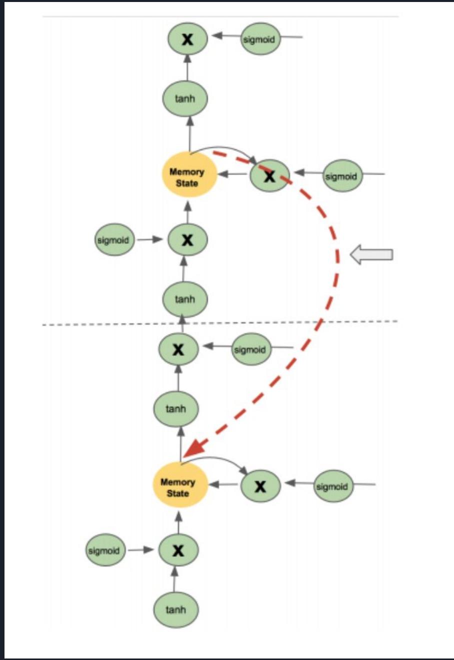
Training DEvol on CIFAR-10

Experiment - Language Modeling

- “The dog ran after the “ - what comes next?
- Penn Tree Bank
  - Corpora of a variety of articles consists of 929k training words, 73k validation words, and 82k test words - overall, it contains 10k unique words
  - Pre-tagged
- Literature Experimentation
  - 50 LSTM networks instantiated with uniform initial weights; parameters included up to two recurrent layers and up to 650 hidden nodes in each layer
  - Each network was trained for 39 epochs; training a single network took >3 hours → inefficient and infeasible
  - Evolved over 25 generations

LSTMs and CoDeepNEAT

- Two approaches taken: varying LSTM units, and finding novel connections between LSTM layers
- CoDeepNEAT → Allowed Mutations
  - Enable/disable connections between layers
  - Skip connections between LSTM nodes
- Blueprints: network graphs
- Modules: LSTM variants



Our Intended Approach

- Deviates from that of CoDeepNEAT
- Two steps:
  - 1. Unsupervised Learning: optimize the LSTM nodes by an objective function
  - 2. Train the networks by optimizing the fitness (NEAT)
- BINGO: Binary Information Gain Optimization algorithm
  - Maximizes the information gained from observing the output of a single layer network
  - Each network's output is treated as a vector of independent binary variables

Where we go from here

- Continue improving our CoDeepNEAT implementation for image classification
- Continue studying the math behind the BINGO algorithm and work on implementing it