JOHANNES KEPLER
UNIVERSITÄT LINZ

Neural Network Weights Visualization

Marius-Constantin Dinu, Yassir Taha

13.12.2017

Motivation

- Deep Neural networks are still black boxes and it is hard to understand their internal training dynamics
- Mostly it is hard to find common structures between pretrained neural networks

Visualization Example I

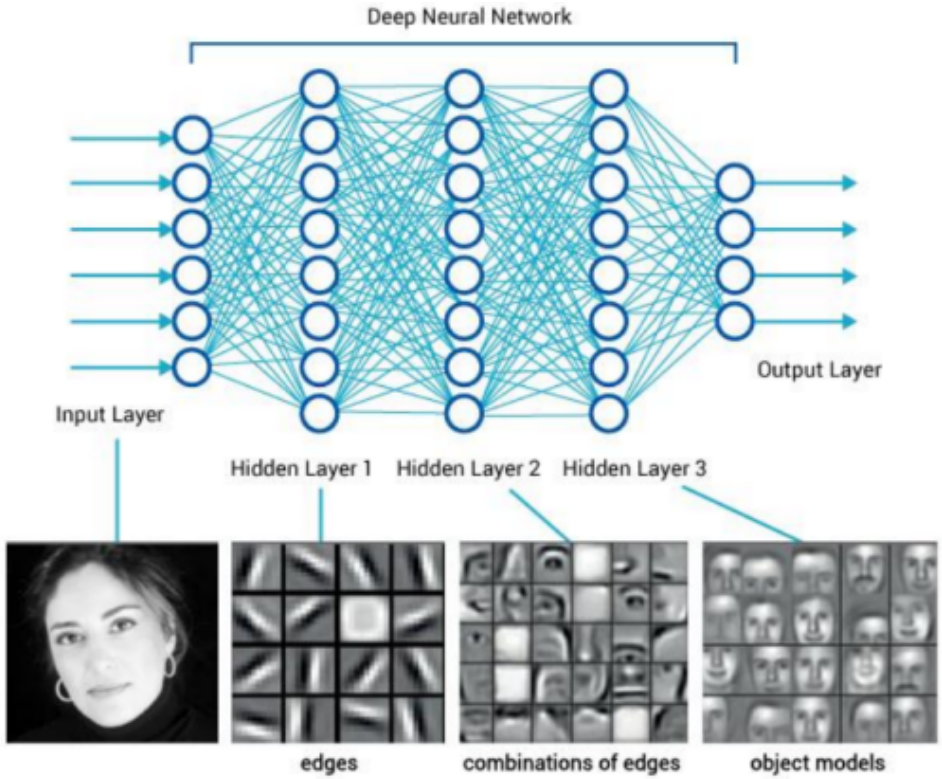


Figure: Visualizing Deep Neural Networks

Visualization Example II

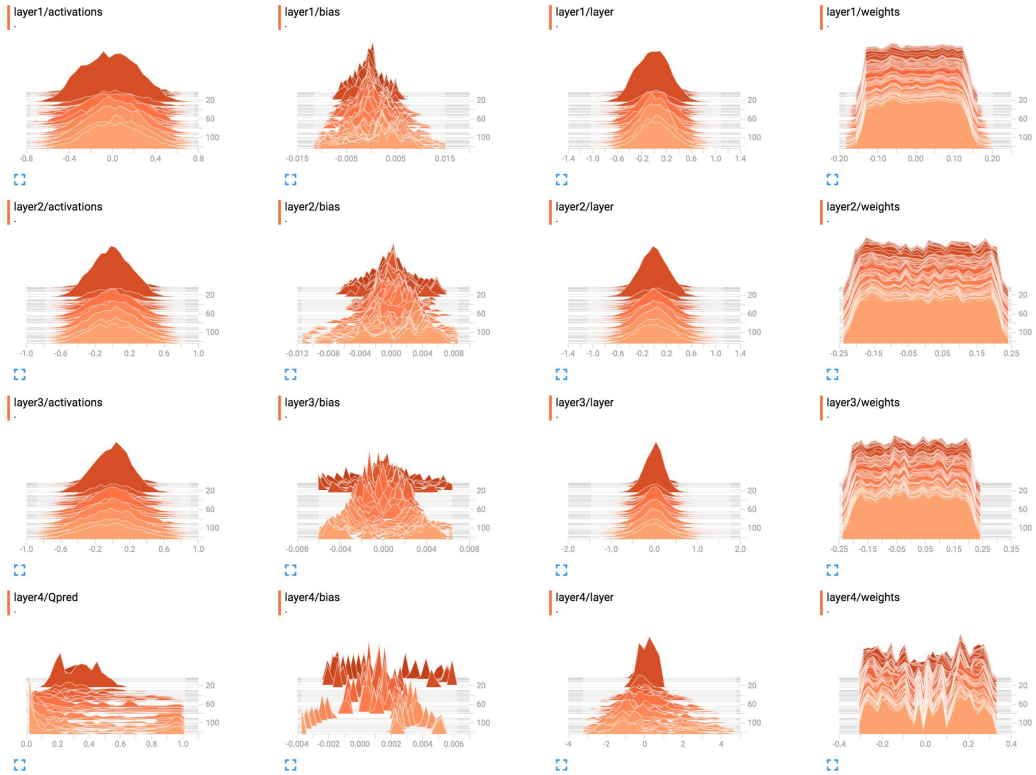


Figure: Tensorboard: Visualizing weight distributions of the layers over time

Visualization Example III

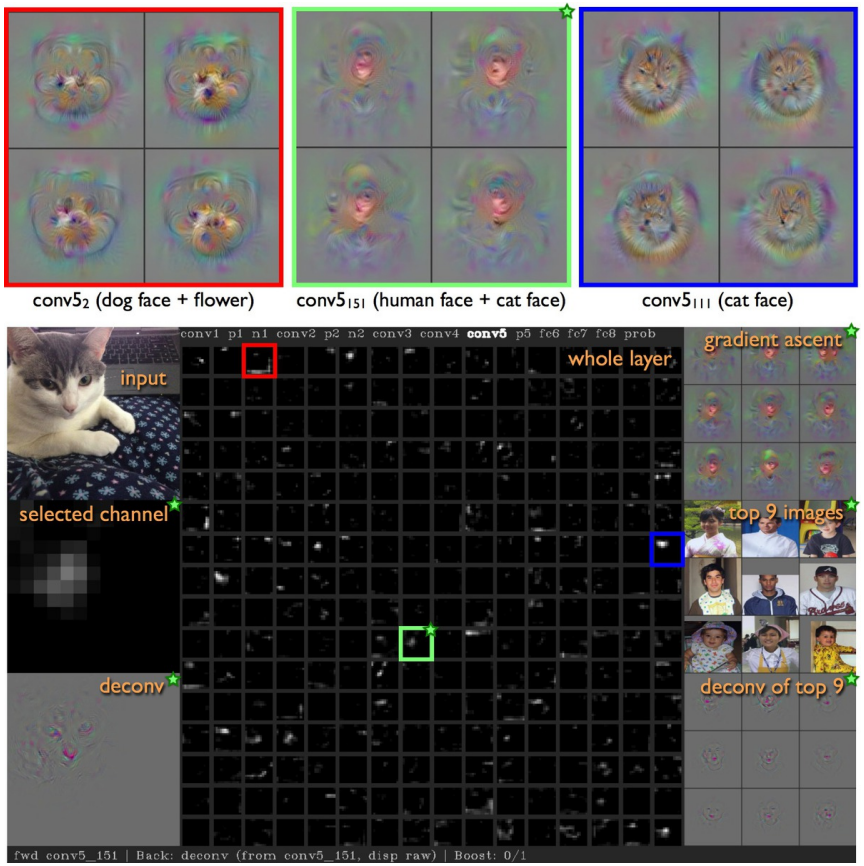


Figure: Visualization Toolbox: Showing neuron activations

Goal

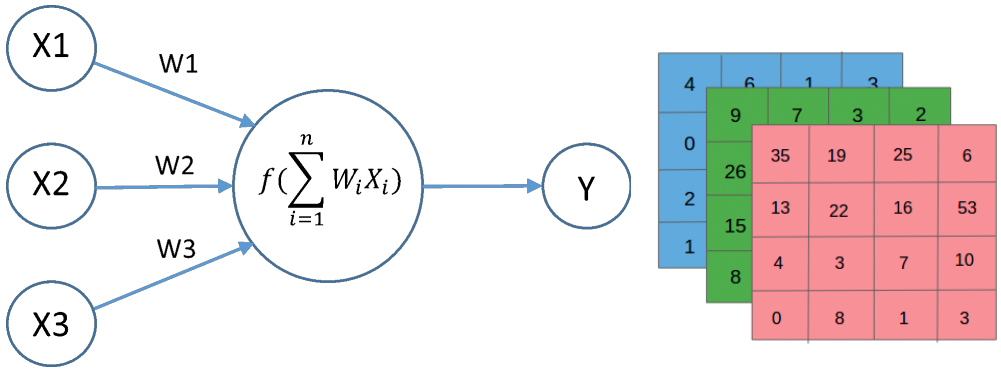
We want to:

- find a good representation to visualize weight distributions after the training process
- extrapolate correlations between weight distributions over time
- compare the dynamics across multiple neural network training processes
- find common patterns between different training results

Dataset Definition

Generating the data:

- given a set of trained neural network models $\mathcal{X}$ on a dataset $\mathcal{D}$
- we want to visualize the weights distribution $\theta_k^{(t)}$ for each model $X_k \in \mathcal{X}$ at each time step t



Dataset Example

- given the MNIST dataset (feature vector of 784 input values)
- a set of 1000 different models with 3 hidden layers and 100 neurons per layer
- we get 784x100 weights for the first layer, 100x100 weights for the second layer and 100x100 weights for the third layer (ignoring biases)
- were we train all models for 30 iterations

$\Rightarrow$ we get a table of:
1000 models | 30 iterations | 3 layers | θ weights
(θ w.r.t. to the corresponding layer)

Dataset Example (cont'd)

Figure: MNIST dataset example.

DeepMind Example

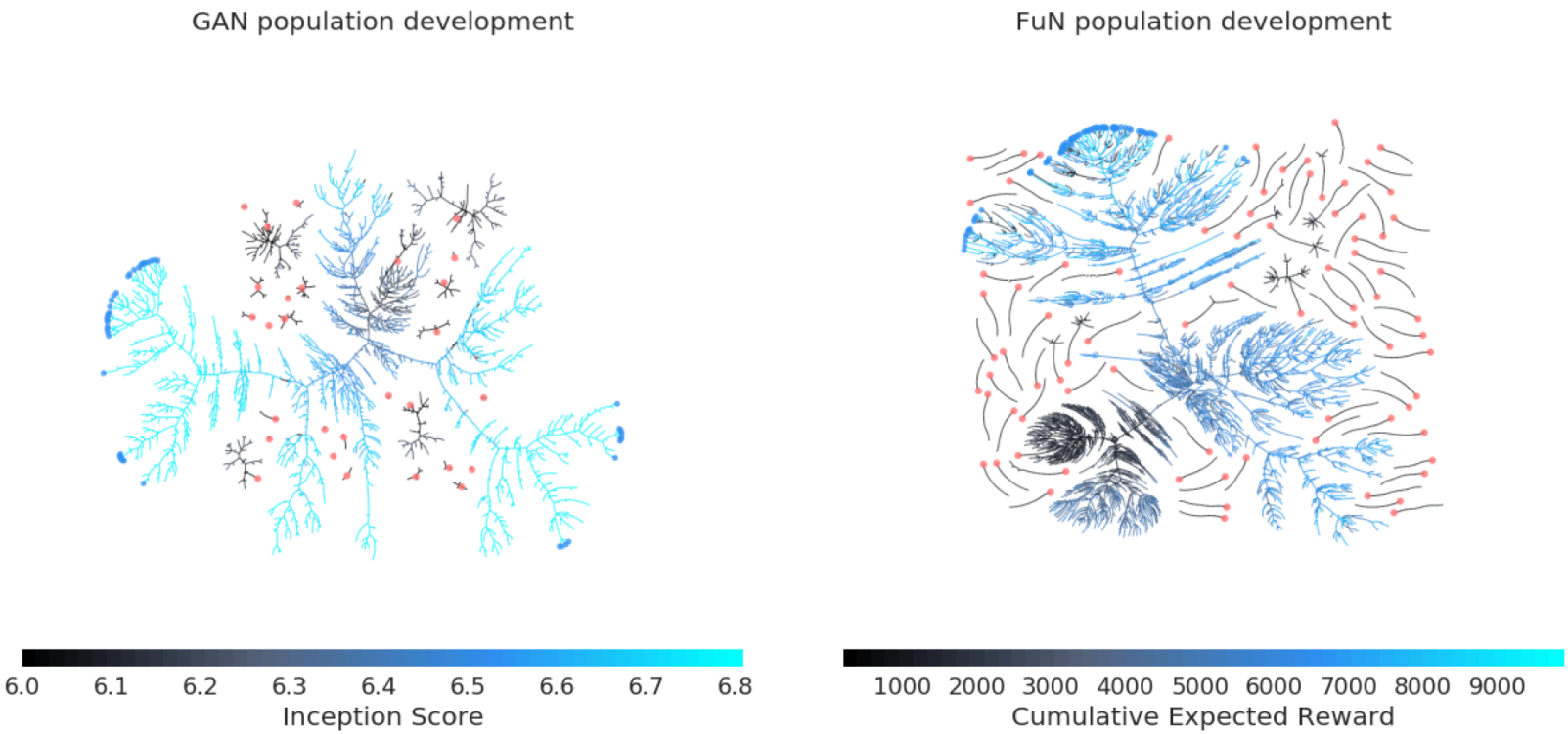


Figure: Neural network weights illustration of training a population.

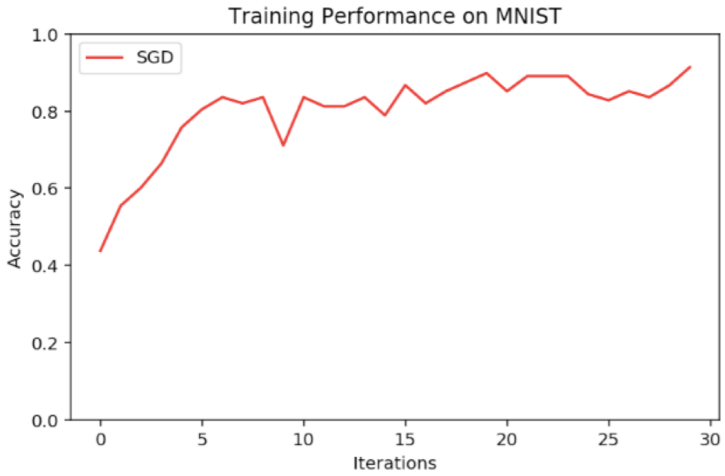
Focus

- Evolution of Individuals
- Visualizing good and bad weights
- Diversity in the population
- Progress during training
- Empirical proof of concept by showing convergence

Concrete Dataset

- given the MNIST dataset (feature vector of 784 input values)
- starting with a set of 20 different models (individuals) with 1 hidden layer (100 neurons) and the output layer (10 neurons)
- we get 784x100 weights for the first layer, 100x10 weights for the output layer (ignoring biases)
- were we train the models either for 20, 30, 40, 50, 100, 200 iterations to measure improvements

Defining Baseline



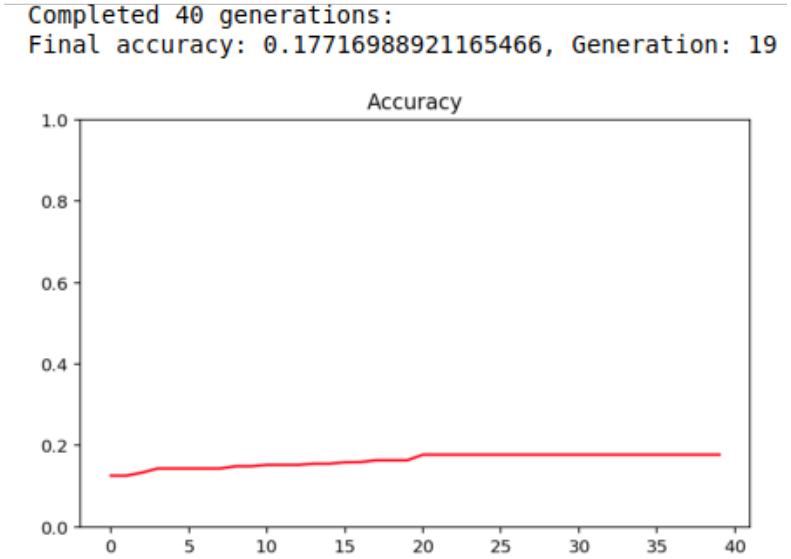


Figure: What happens by chance if you only select the best models.

Visualizing Evolution

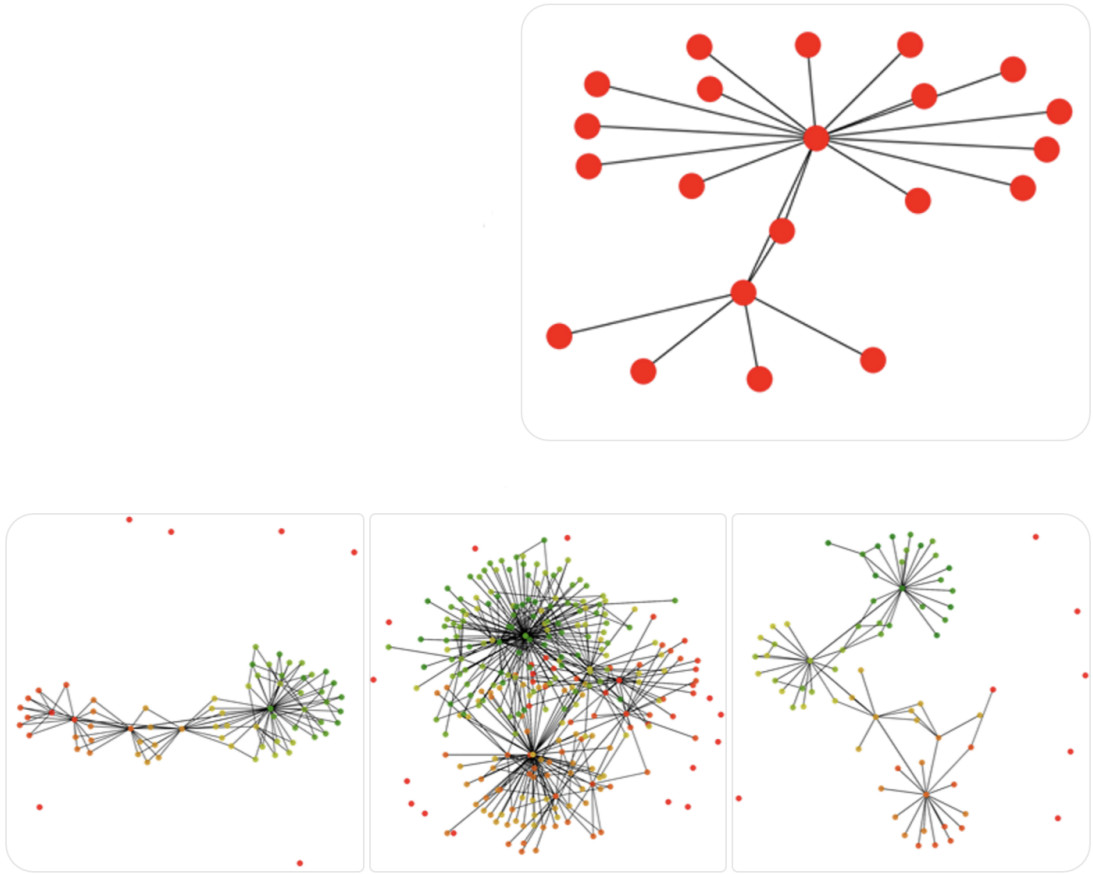


Figure: The story of evolution begins with bugs.

Visualizing Evolution: Interpretations (cont'd)

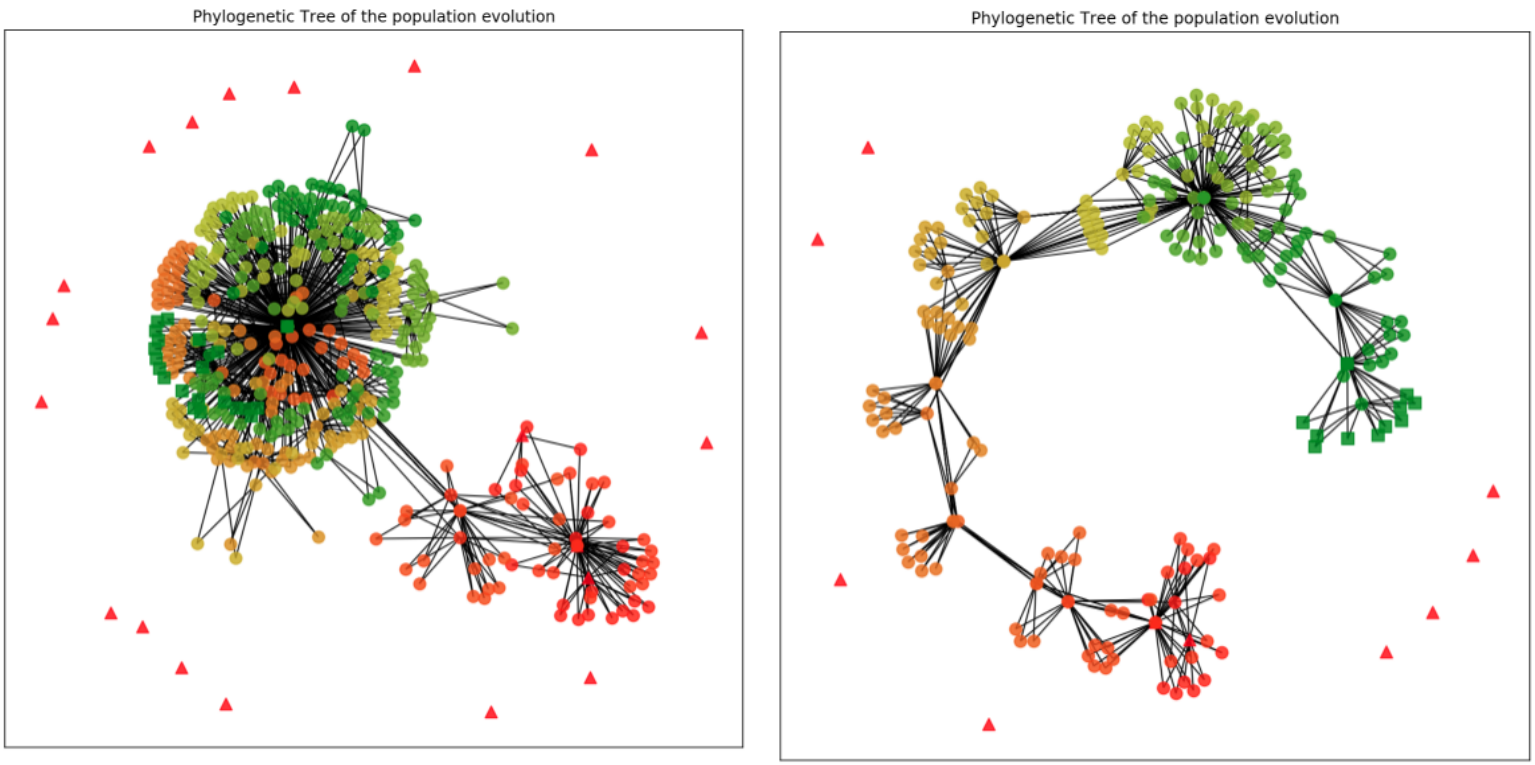


Figure: Diversity collapse and path of evolution.

Improved, but still crap

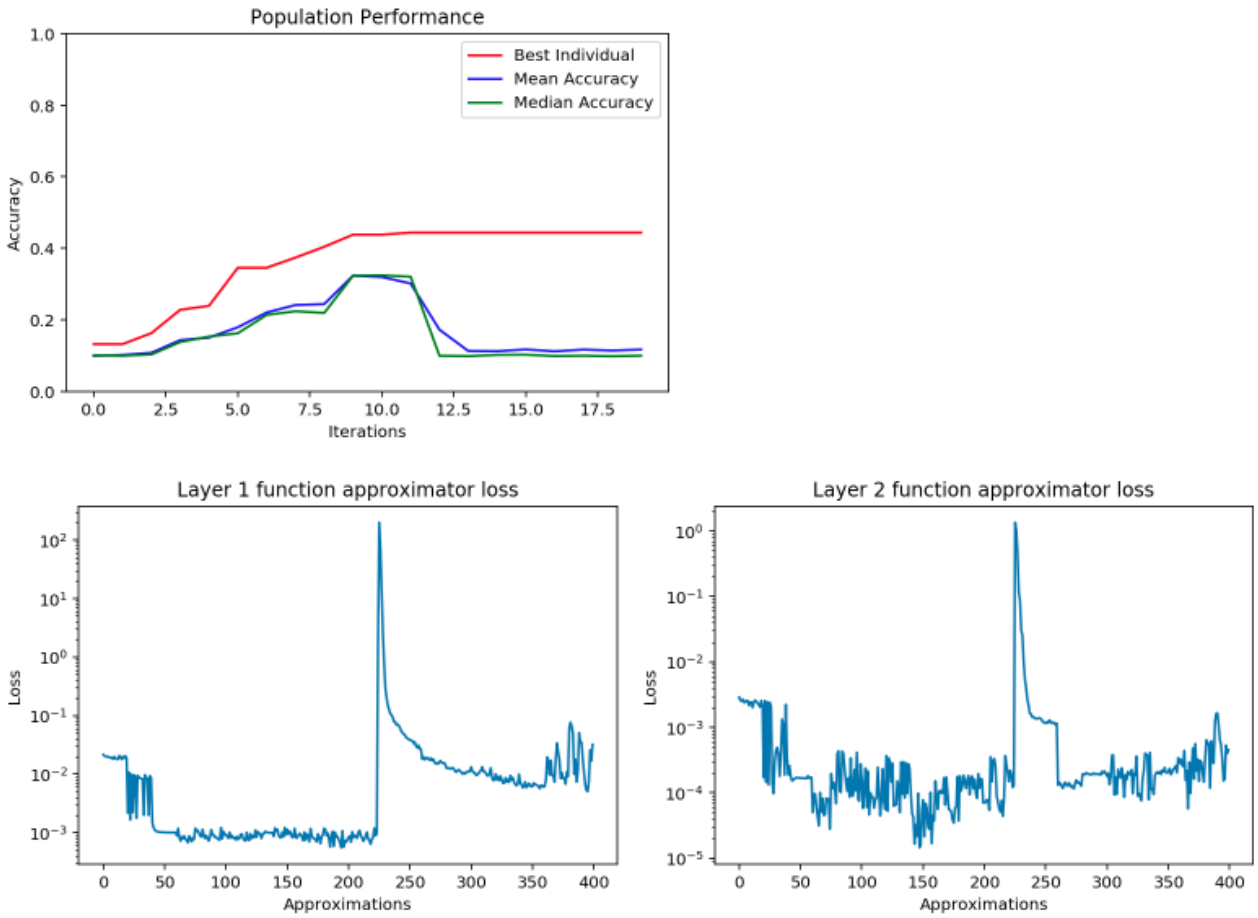


Figure: Results after a few trials.

Improved, but still crap (cont'd)

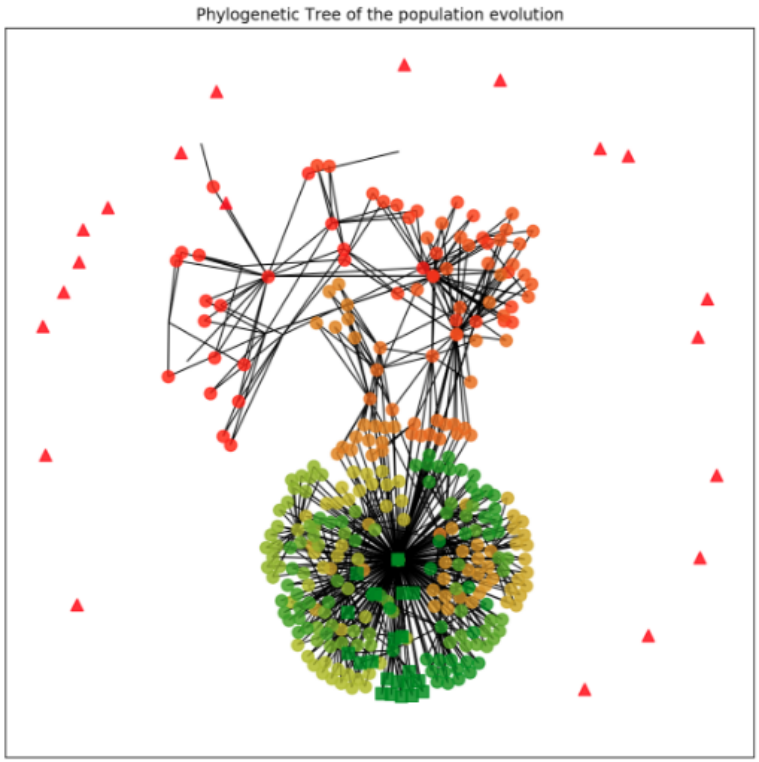


Figure: The function approximator learning rate and suboptimal kernel filters messed up the training.

Visualizing Neurons

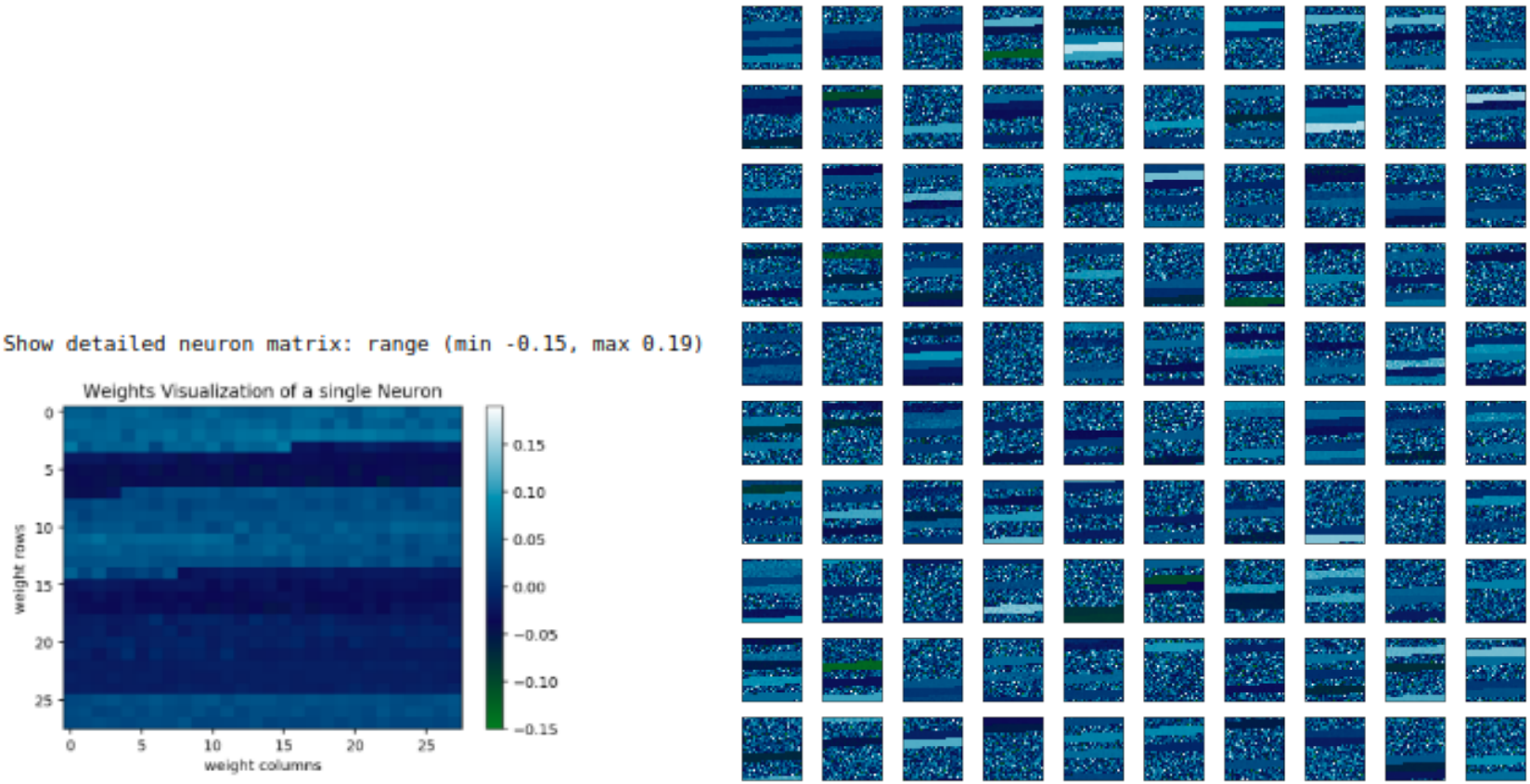


Figure: Neurons of a very good model.

Visualizing Neurons: Bad vs. Good Neurons (cont'd)

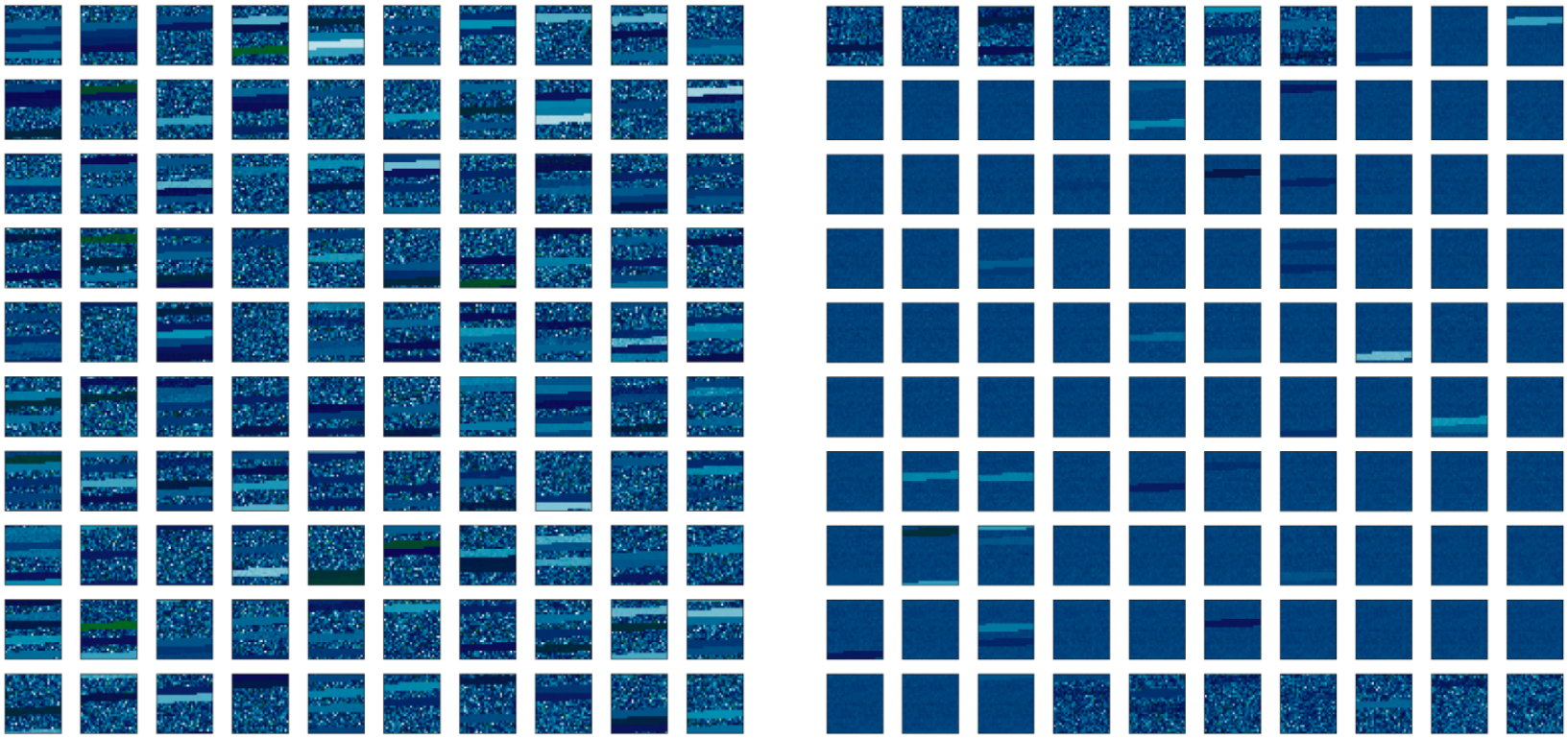


Figure: Comparing neurons of trained models. Good: left, bad: right

Visualizing Neurons: Improving the Kernel Filter (cont'd)

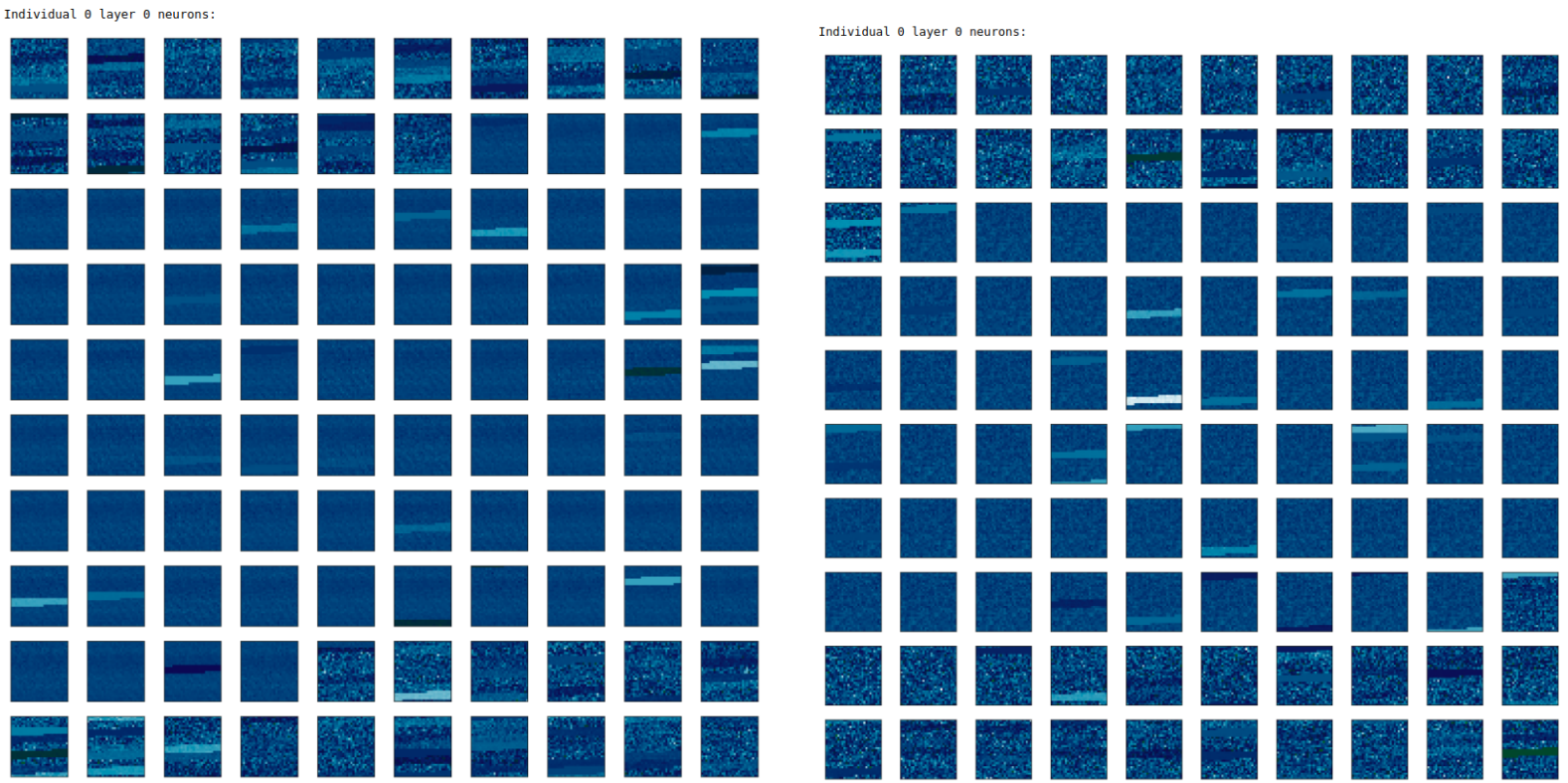


Figure: Comparing models after improving the kernel filter. Before: left, after: right

Promising results

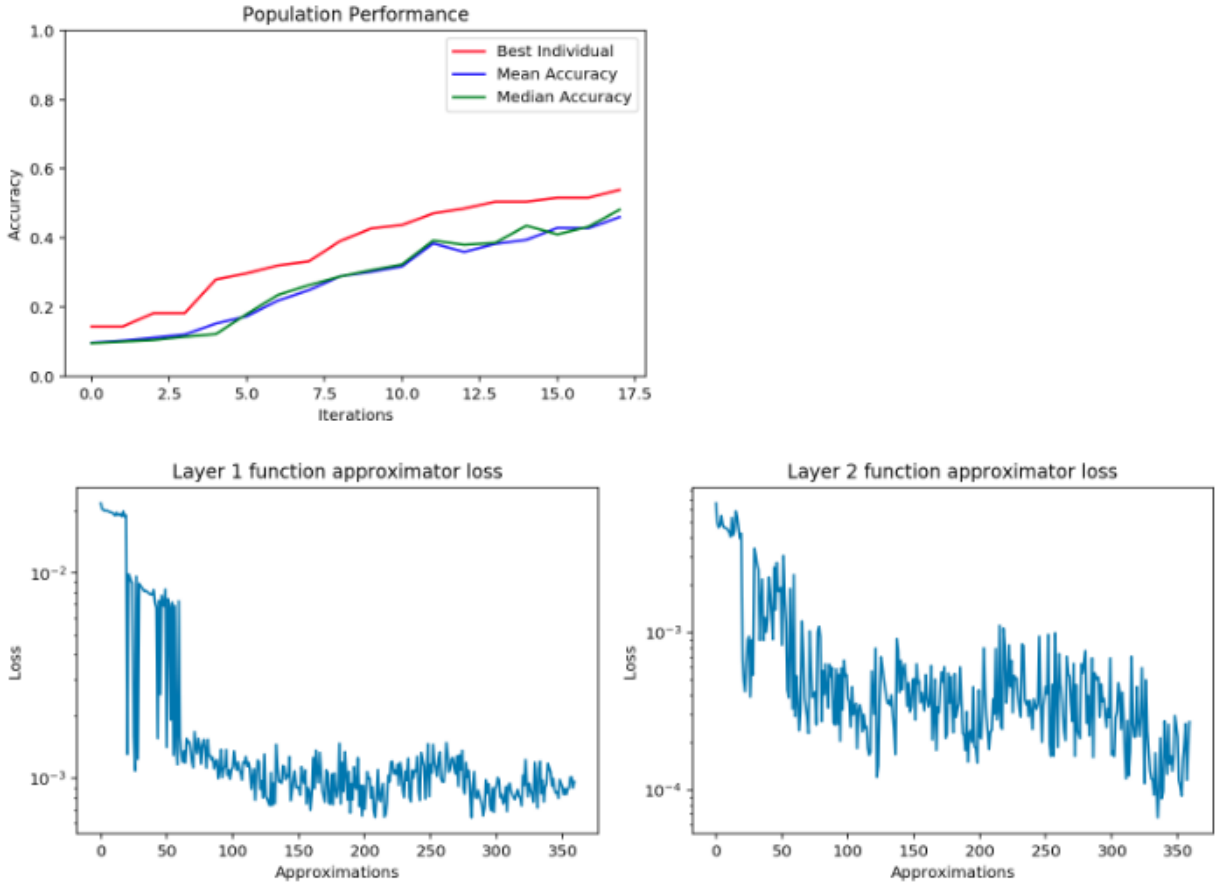
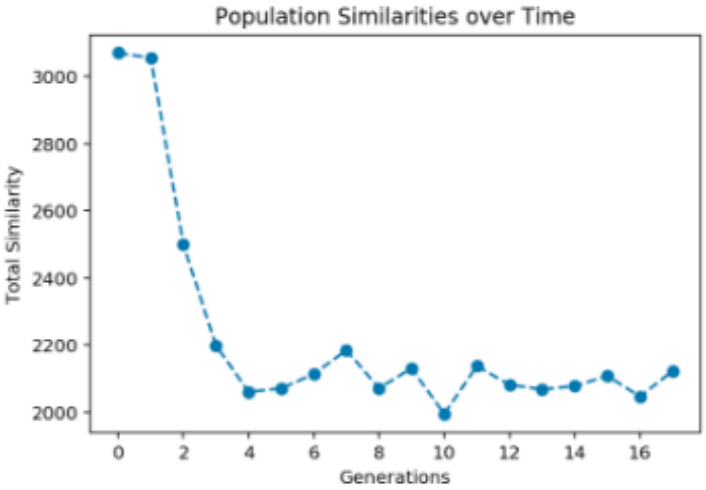


Figure: Good training evolution.

Promising results: Measuring Population Diversity (cont'd)



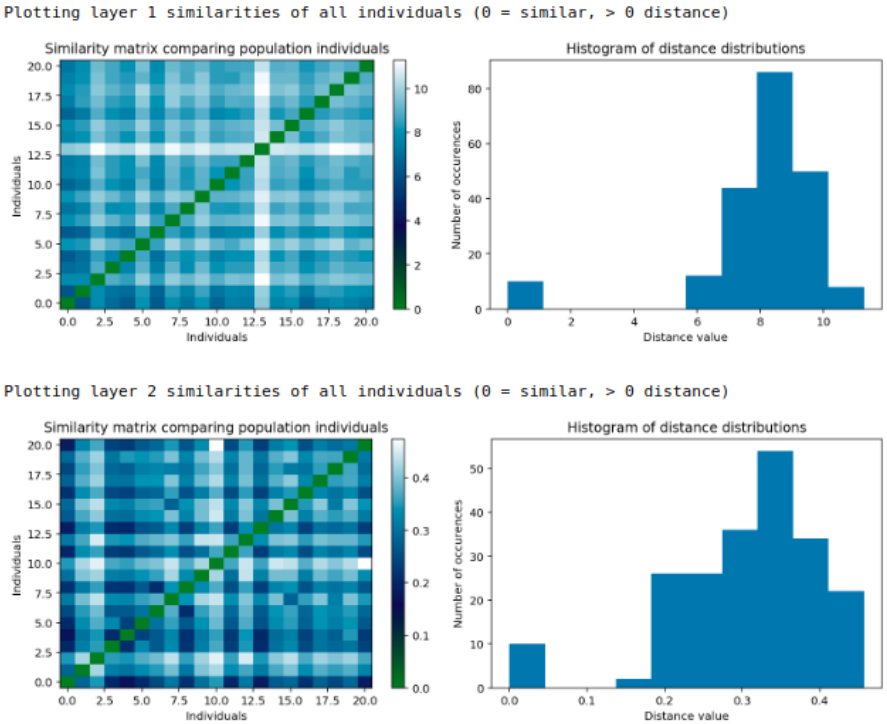


Figure: Diversity of the population.

Found the sweet spot

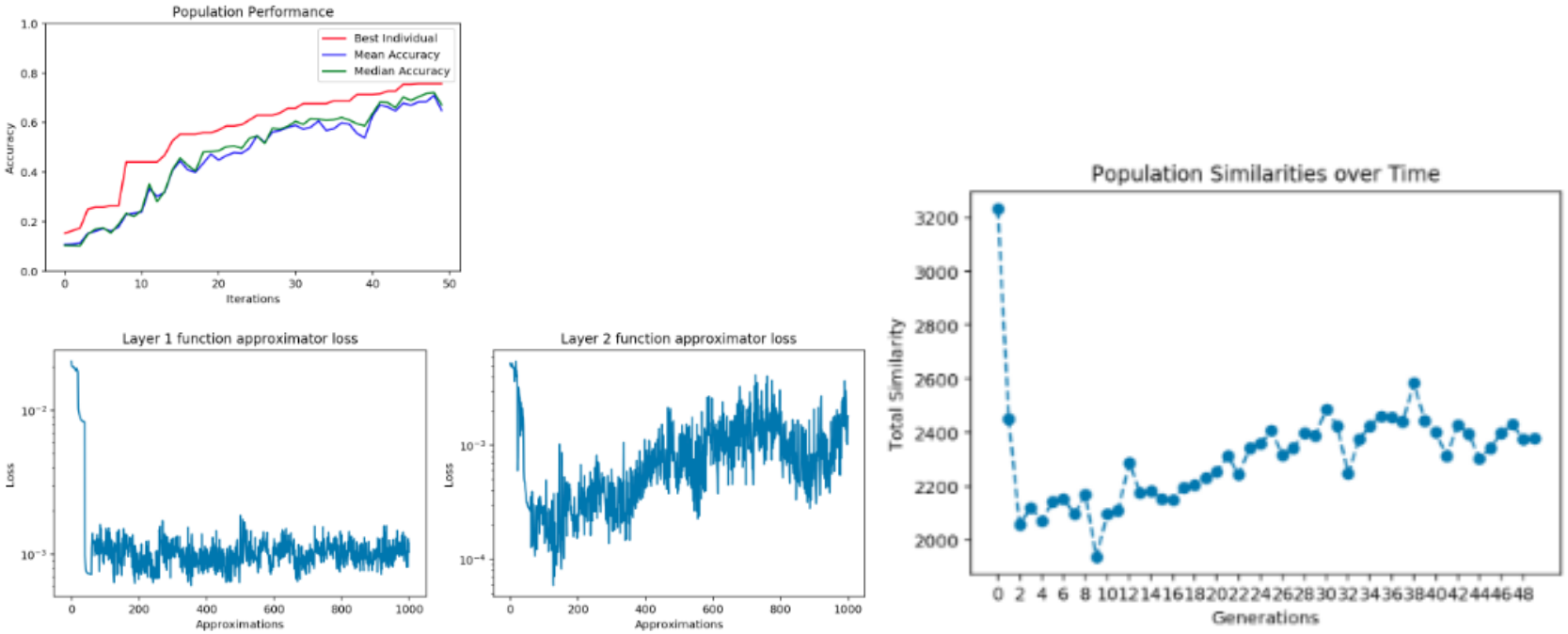


Figure: Good training results.

Found the sweet spot: Evolution Tree (cont'd)

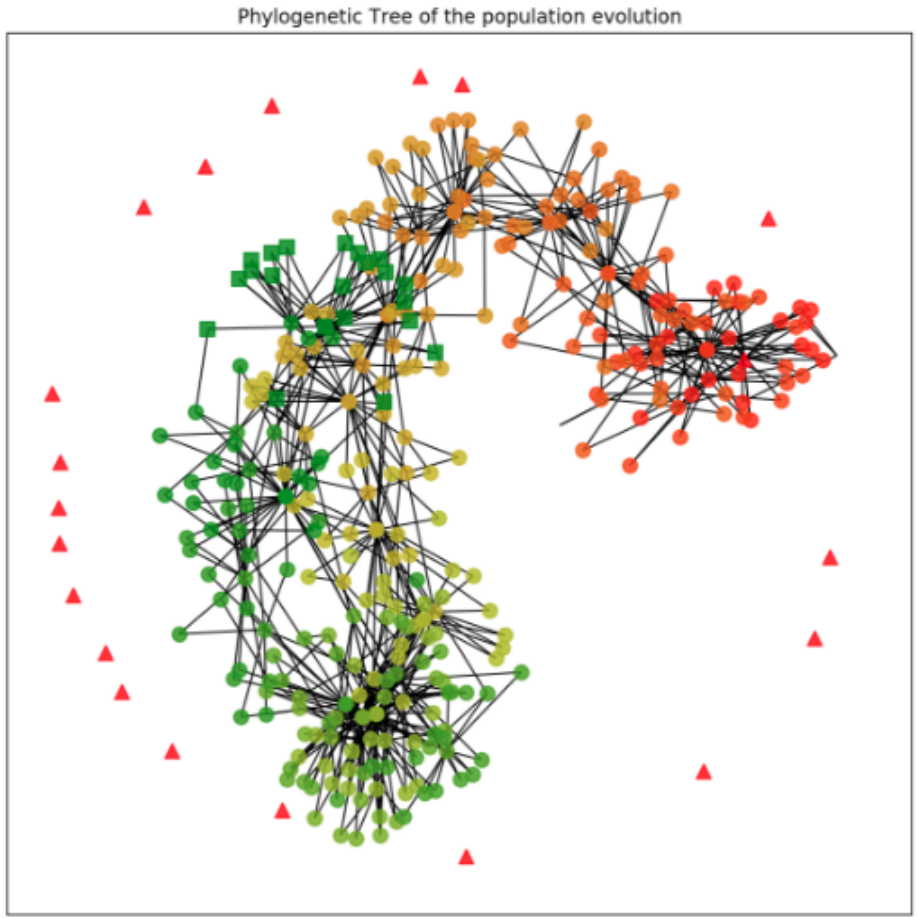


Figure: Evolution of the population over time.

Found the sweet spot: Weights Visualization (cont'd)

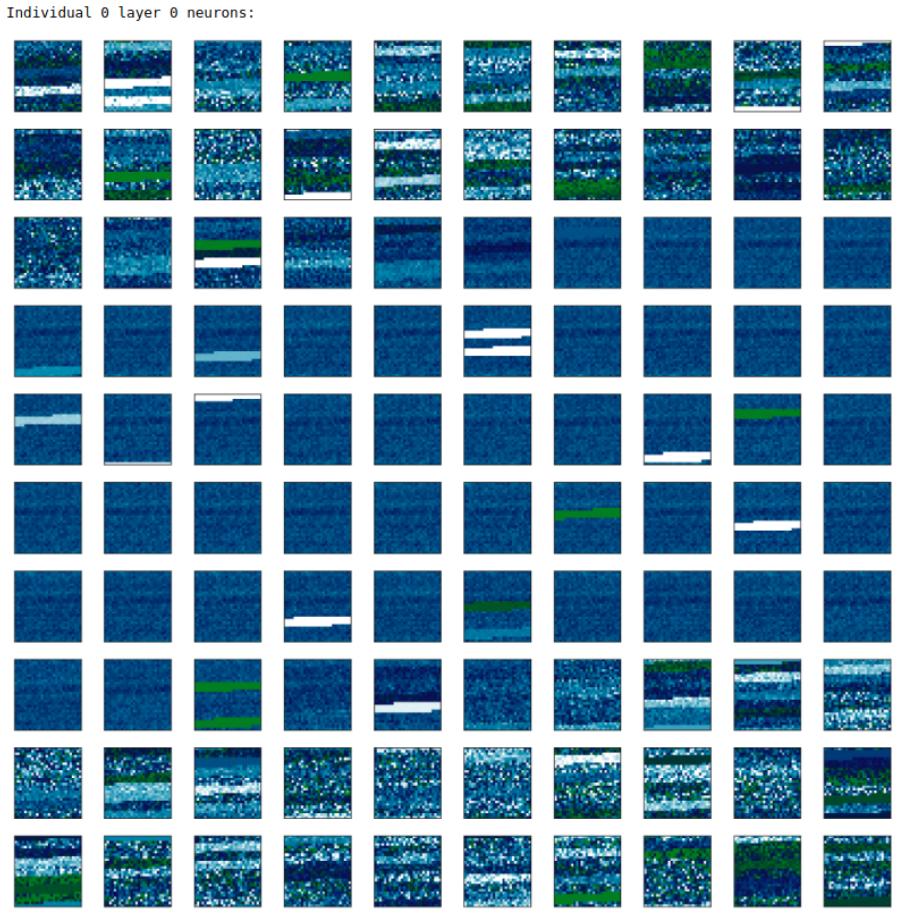


Figure: Weights of the last training step.

Bootstrapping Knowledge

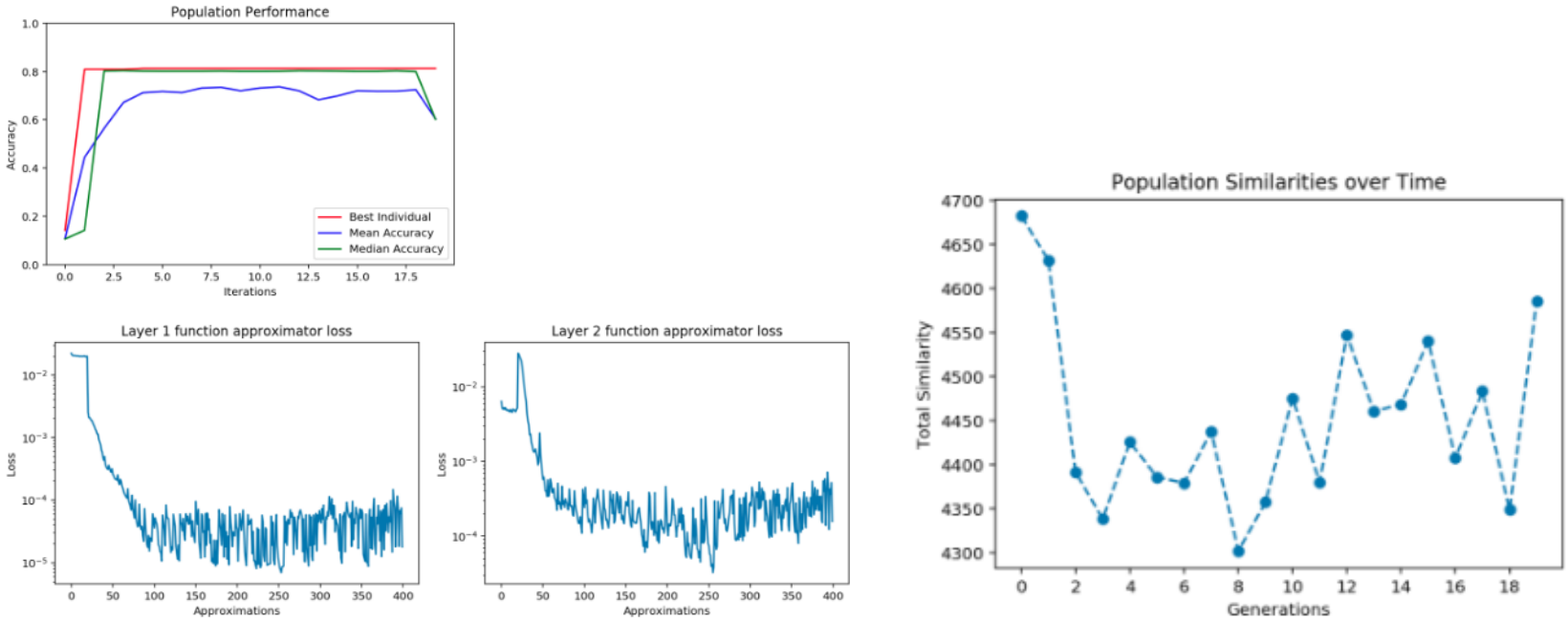


Figure: Using a pretrained function approximator to train another.

Bootstrapping Knowledge: Evolution Tree (cont'd)

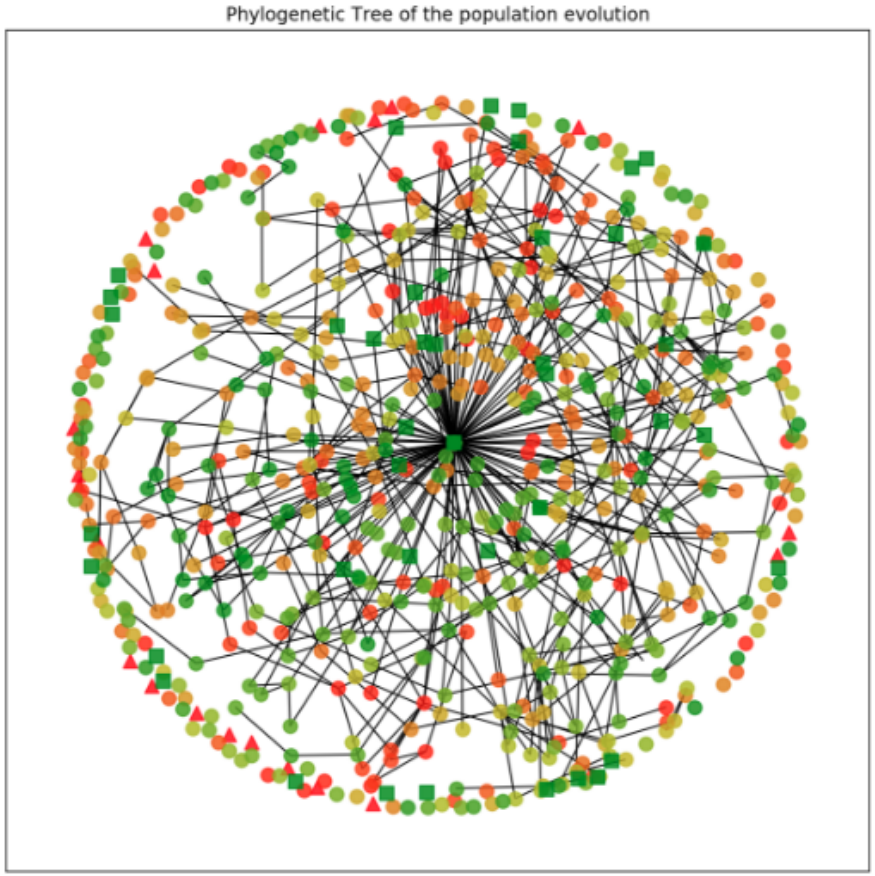


Figure: Evolution of the population with knowledge transfer.

Bootstrapping Knowledge: Population diversity (cont'd)

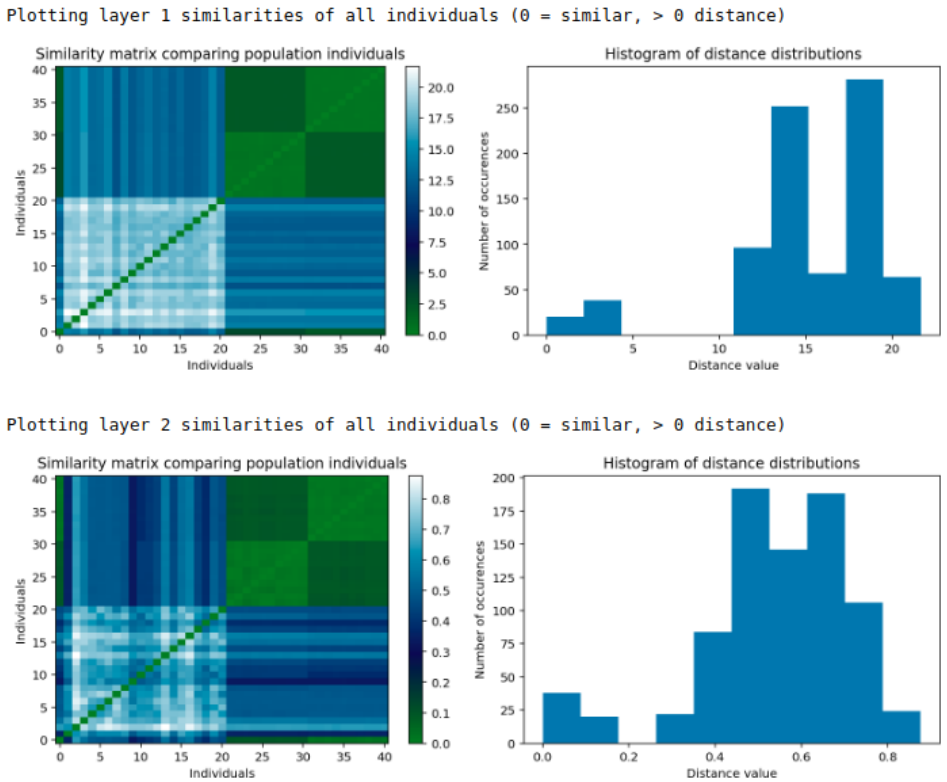


Figure: Distance matrix with knowledge transfer.

Bootstrapping Knowledge: Weights Visualization (cont'd)

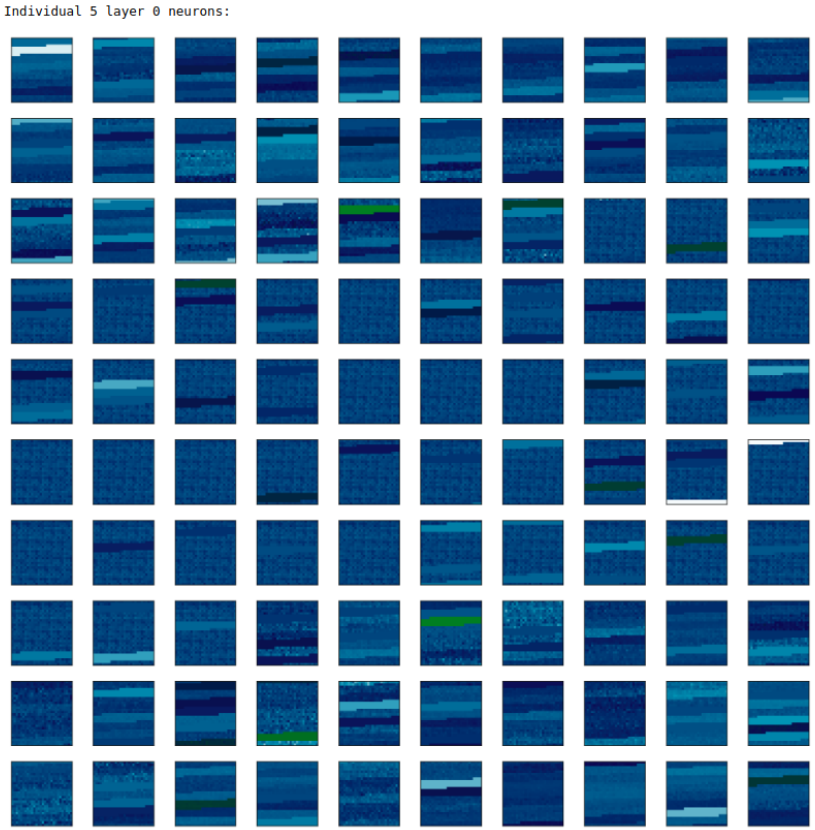


Figure: Weights visualization of trained function approximator with knowledge transfer.