Compute-efficient neural network architecture optimization by a genetic algorithm

Sebastian Litzinger   Andreas Klos   Wolfram Schiffmann

Faculty of Mathematics and Computer Science, FernUniversität in Hagen, Germany
contact author: sebastian.litzinger@fernuni-hagen.de

ICANN19

FernUniversität in Hagen

Motivation

- An artificial neural network's (ANN) **topology** greatly influences its ability to **generalize**
- **Neural architecture search** (NAS) seeks to **optimize** an ANN's topology to facilitate high prediction accuracy values and generalization capability
- The **search space** is **huge**, hence favouring **heuristic** approaches
- **Genetic algorithms** (GAs) have been shown to deliver **competitive** results
- Unfortunately, the **computational effort** is still **immense**, calling for endeavors to **raise efficiency** in order to enable NAS when computational **resources** are **scarce**

Contributions

- We present a GA for ANN topology optimization, which can be deployed effectively in low-resource settings
- Optimization aims for classification accuracy as well as compact models
- We provide experimental results based on an implementation of the GA in Python, with the ANN fitness evaluation component utilizing the TensorFlow framework
- We incorporate various techniques to reduce the computational load

Raising Efficiency

- Factor the number of free parameters into fitness evaluation
- Employ and extend early stopping technique to dynamically find adequate duration of training for any architecture
- Python implementation ensures portability and facilitates use of TensorFlow for the evaluation component
- Weight sharing and layer freezing have massive impact on computational demand (reduced by ≈ 2/3)
- Design of genetic operators promotes quick convergence to high quality solutions
- Constraints on architecture possible to force "common" CNN topologies

NAS on the MNIST Dataset

Table 1. Recent results of NAS methods on MNIST

work	test set accuracy
Ma & Xia (2018)	99.72%
Assunção et al. (2018)	99.70%
proposed approach	99.69%
Baldominos et al. (2018)	99.63%
Real et al. (2018)	≈99.50%
Mitschke et al. (2018)	98.67%

Genetic Algorithm

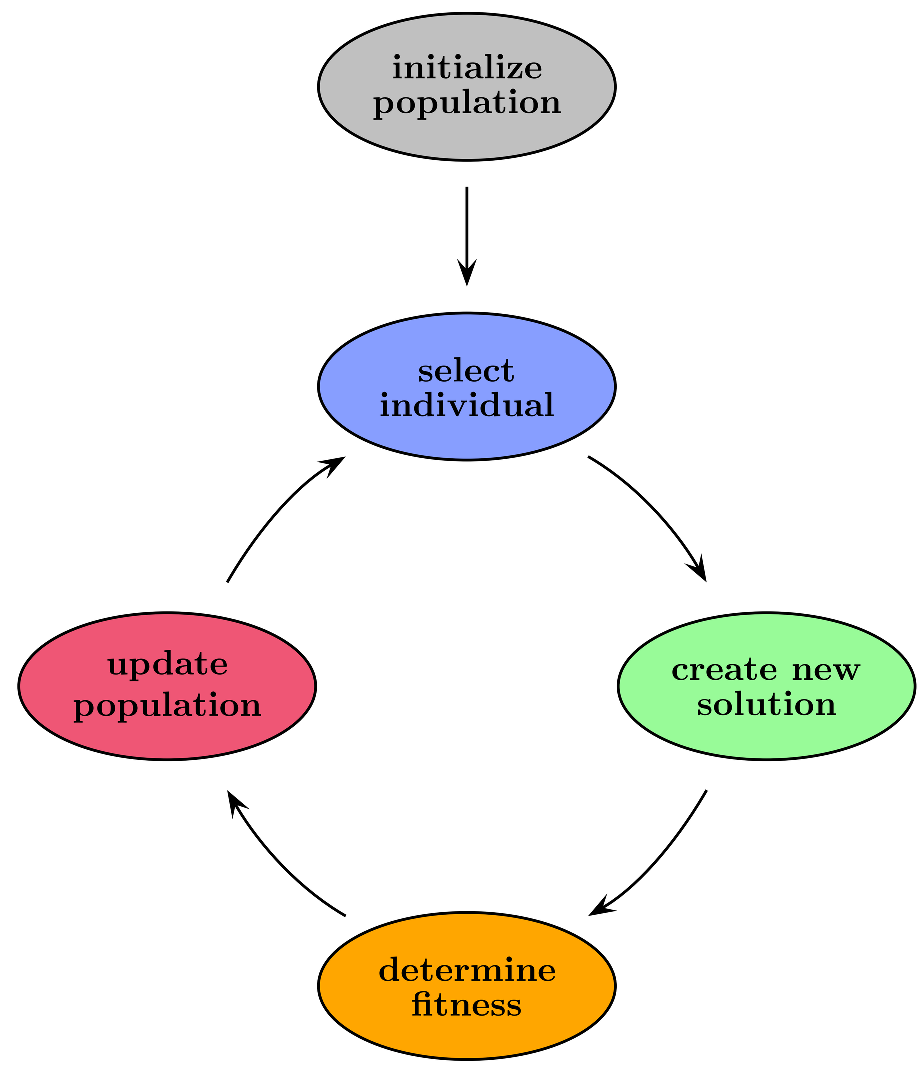

Figure 1. Generic genetic algorithm

- Fitness criteria: prediction accuracy, number of free parameters
- Evolution by mutation: insert/delete/modify layer, switch two layers
- Evolution by recombination: crossover at configurable number of crossover points
- Layer modifications: in-/decrement of values decaying over time, thus focus shifts from quick exploration of solution space to refining the best solutions
- Selection step: roulette selection, two types of tournament selection
- Repeated training advisable for precise evaluation (cf. Figure 2), configurable and adaptable to accuracy and number of free parameters

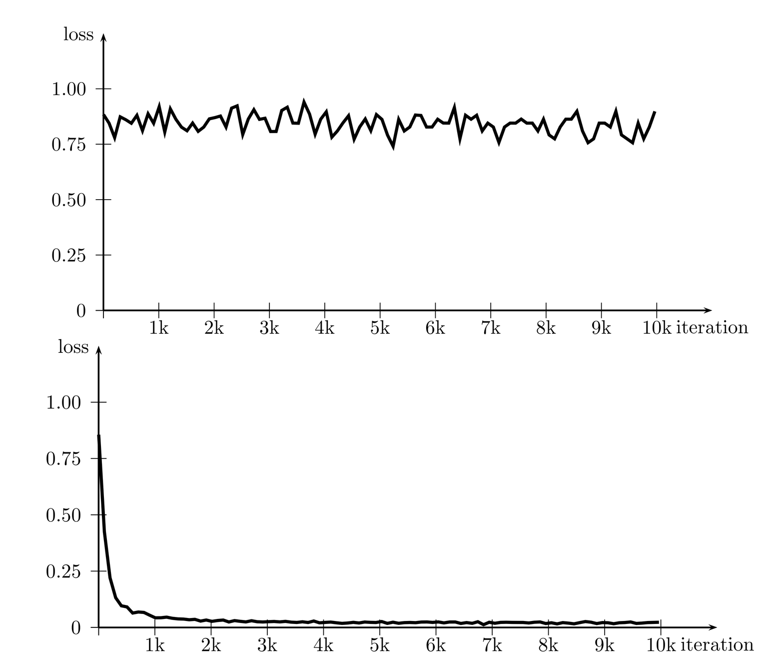

Figure 2. Different progression of training loss under equal configuration of training

Experiments

- Classification of the MNIST dataset
- Determine parametrization for network training and GA
- **Scenario 1**: achieve same test set accuracy as ANN from TensorFlow documentation with minimal free parameter count
- **Scenario 2**: achieve maximal test set accuracy with fewer free parameters than ANN from TensorFlow documentation

Results

Parametrization of training for MNIST problem:

- AdaGrad training algorithm
- He or Xavier initialization
- Learning rate scheduling
- Batch normalization for convolutional layers
- Dropout regularization for dense layers

Results for **scenario 1**:

- Test error rate: 0.60%
- 61,787 free parameters
- 1.9% the size of reference network
- 47 hours runtime on Nvidia GeForce GTX 1050 Ti for 2000 iterations of GA
- GA can reveal small yet well-performing architectures – essential for ANNs operating under firm real-time constraints

Table 2. Best architecture found for scenario 1

layer type	dimensions	kernel	stride	feature maps
input	28 × 28			
convolutional	14 × 28	5 × 5	2 × 1	52
avg. pooling	14 × 28	2 × 3	1 × 1	
convolutional	14 × 28	3 × 2	1 × 1	50
convolutional	7 × 14	5 × 3	2 × 2	24
convolutional	4 × 14	2 × 5	2 × 1	33
dense	10			

Results for **scenario 2**:

- Competitive test error rate of 0.31%
- ≈ 2.5m free parameters
- Runtime: days on Nvidia Pascal consumer card
- Endeavor to reduce resource consumption does not impair quality of the results

Table 3. Best architecture found for scenario 2

layer type	dimensions	kernel	stride	feature maps
input	28 × 28			
residual	14 × 28	3 × 7	2 × 1	254
convolutional	14 × 28	2 × 3	1 × 1	102
residual	14 × 28	6 × 7	1 × 1	86
residual	4 × 7	5 × 4	4 × 4	16
residual	2 × 4	4 × 3	3 × 2	114
convolutional	1 × 4	1 × 2	3 × 1	170
dense	172			
dense	10			

Conclusions & Outlook

- NAS via GA with focus on **computational efficiency** can deliver **competitive results**
- MNIST dataset: **single** Nvidia Pascal **consumer card** provides sufficient performance
- For larger datasets: run GA **on multiple machines** independently, occasionally **exchange information**