Evolutionary Algorithms in Convolutional Neural Network Design

Author: Filip Badáň,
badan.filip@gmail.com Supervisor: prof. Ing. Lukáš Sekanina, Ph.D.

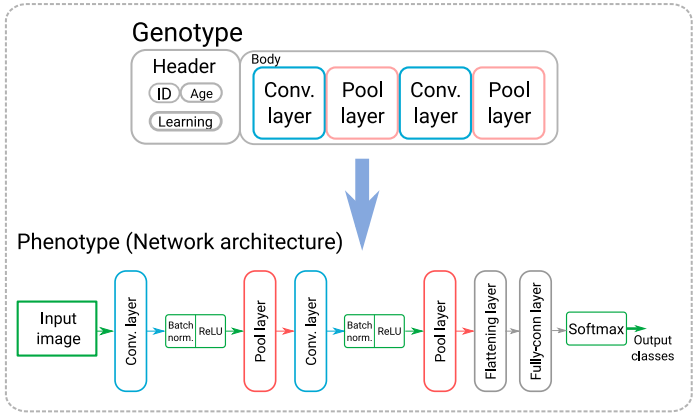
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Motivation

The design of high-quality DNNs is a hard task even for experienced designers because the state of the art DNNs have large and complex structures with millions of tunable parameters. The purpose of this framework is to automatize this process with respect to the classification error and model complexity. The framework focuses on the design of convolutional neural networks.

Principle

The framework uses the evolutionary algorithm specially designed to search through the state space of different CNN structures with the goal of finding the ones with the optimal architecture and parameters. The framework uses the concept of weight inheritence to make the evolutionary process more effective. The simple speciation mechanism based on the training age is used to give more innovative, but less trained candidate solutions equal chance to train.

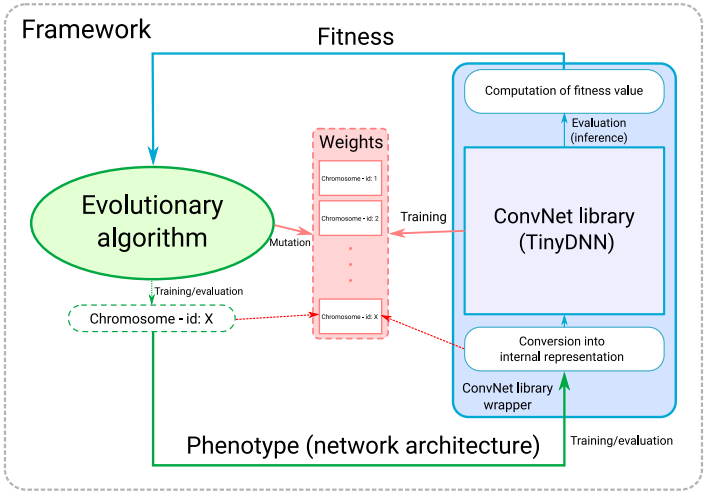


Mutations

1. **Weight reset** - all weights of a given layer are randomly generated.
2. **Add a new layer** - a randomly generated layer (with randomly generated hyperparameters) is inserted on a randomly chosen position in CNN.
3. **Remove layer** - one layer is removed from a randomly chosen position.
4. **Modify layer** - some parameters of a randomly selected layer are randomly modied.
5. **Modify hyperparameters of the fully connected layer** - the number of connections in the last fully connected layer is increased or decreased.
6. **Modify the learning rate (randomly).**

Crossover

Simple one-point crossover operator on each pair of parents.



Algorithm

1: P = Create Initial Population; // randomly or using existing CNN
2: Evaluate(P, D_{test}) using TinyDNN; i = 0;
3: **while** $(i < G_{max})$ **do**
4: $Q = \emptyset$; // a set of offspring
5: **while** $(|P| \neq |Q|)$ **do**
6: a, b = Tournament Selection (P);
7: a', b' = Crossover(a, b, p_{cross});
8: a'' = Mutation(a', p_{mut}); b'' = Mutation(b', p_{mut});
9: $Q = Q \cup \{a''\} \cup \{b''\}$;
10: **end while**
11: Run TinyDNN's Training Algorithm for all NNs in Q with D_{train};
12: Update the Age counter for all NNs.
13: Evaluate(Q, D_{test}) using TinyDNN;
14: P = Replacement With Speciation (P, Q);
15: $i = i + 1$;
16: **end while**

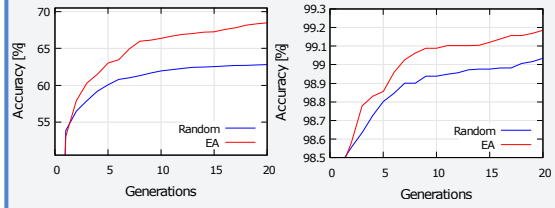
Fitness function

The fitness function is based on the CNN accuracy *a* but it also incorporates complexity of an architecture *p* (number of parameters) with the goal of finding not only the most accurate, but also compact solutions. The coefficient *k* can be used to affect how much the network complexity contributes to the final fitness score.

$$f_x = a * (k * \frac{1}{\log(p) + 1})$$

Results

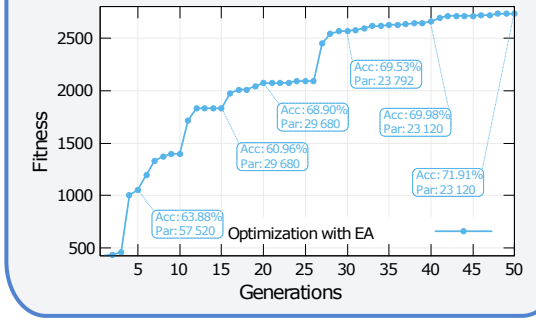
Experimental results on datasets MNIST and CIFAR10 showed great potential of the proposed framework. The best network found by the EA on dataset MNIST had the accuracy of 99.36%, while the accuracy of 73.05% was obtained for CIFAR10 in a relatively short time. In both cases the proposed EA clearly outperformed better then control random experiment.



Optimization through EA

The framework can also be used to optimize an existing CNN architecture by reducing the number of parameters (weights and biases) while keeping the test accuracy highest as possible.

CNN	Parameters	Accuracy	Layers	E_{mult} reduction
Baseline CNN (FP)	360,264	75.80 %	7	1.0
Evolved with EA4CNN				
CNN1	8 480	64.33 %	9	42.9×
CNN2	12 784	67.50 %	7	28.1×
CNN3	15 728	68.92 %	8	22.9×
CNN4	23 120	70.97 %	9	15.6×
CNN5	0.17 M	72.96 %	6	2.1×



Acknowledgments

This work was supported by the Ministry of Education, Youth and Sports, under the INTER-COST project LTC 18053, NPU II project IT4Innovations excellence in science LQ1602 and by Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center - LM2015070".