EvoCNN: Evolving Deep Convolutional Neural Networks Using Backpropagation-Assisted Mutations

Eli (Omid) David(✉) and Nathan S. Netanyahu

Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
mail@elidavid.com, nathan@cs.biu.ac.il

Abstract. In this abstract we present our initial results with a novel genetic algorithms based method for evolving convolutional neural networks (CNN). Currently, standard backpropagation is the primary training method for neural networks (NN) in general, including CNNs. In the past several methods proposed using genetic algorithms (GA) for training neural networks. These methods involve representing the weights of the NN as a chromosome, creating a randomly initialized population of such chromosomes (each chromosome represents one NN), and then evolving the population by performing the steps (1) measure the fitness of each chromosome (the lower the average loss over the training set, the better), (2) select the fitter chromosomes for breeding, (3) perform crossover between the parents (randomly choose weights from the parents to create the offspring), and (4) mutate the offspring. While in smaller NNs these methods obtained results comparable with backpropagation, their results deteriorate as the size of NN grows, and are impractical for training deep neural nets. Nowadays these methods have largely been abandoned due to this inefficiency.

We propose a combination of GA-based evolution and backpropagation for evolving CNN as follows. Similar to the abovementioned methods we create an initial population of N chromosomes, each representing the weights of one CNN, and then evolve the chromosomes by applying fitness evaluation, crossover, and mutations, but with several key differences: **(1) During crossover, instead of randomly selecting weights from each of the two parents, randomly select entire filters from each parent** (this ensures that a useful filter is copied in its entirety, rather than being disrupted), **(2) During mutation, modify weights by performing standard backpropagation, instead of random changes; and then randomly set a small portion of weights to zero** (these steps allow for a more goal-oriented evolution, and zeroing some weights encourages sparsity in the network and has a regularizing effect). We refer to this method as EvoCNN.

To measure the performance of our method, we ran several experiments on the MNIST handwritten digit recognition dataset. A standard CNN architecture was used containing the following layers: [Input size 28×28]–[convolution with 128 filters of size 5×5]–[max-pooling]–[convolution with 256 filters of size 3×3]–[max-pooling]–[fully connected layer of size 1000]–[softmax layer of size 10].

A. Lintas et al. (Eds.): ICANN 2017, Part II, LNCS 10614, pp. 725–726, 2017.
https://doi.org/10.1007/978-3-319-68612-7

For a baseline to compare against, the CNN with backpropagation alone resulted in test error of **0.82%**. Training 20 separate CNNs and then performing model averaging reduced the test error to **0.75%**. Using the EvoCNN method described above, we trained a population of 20 CNNs, with a crossover rate of 0.75 and mutation rate of 0.005. The test result in this case was **0.51%**, a new state-of-the-art for MNIST without preprocessing, dataset augmentation (e.g., by distortions), and without pretraining. Note that the entire time required for training a population of 20 CNNs is similar to the time required for training 20 separate CNNs, so the comparison shows a substantial improvement due to our method.