Symposium MATHEMATICS AND APPLICATION, Faculty of Mathematics, University of Belgrade, 2019 ,Vol. X (1)

Modified hybrid genetic algorithm for training convolutional neural networks

Milan M. Čugurović

Faculty of Mathematics, University of Belgrade, Studentski trg 16, 11000 Belgrade
e-mail: milan_cugurovic@matf.bg.ac.rs

Nikola Dimitrijević

Microsoft Development Center Serbia, Španskih boraca 3, 11000 Belgrade
e-mail: nikoladim95@gmail.com

Stefan Mišković

Faculty of Mathematics, University of Belgrade, Studentski trg 16, 11000 Belgrade
e-mail: stefan@matf.bg.ac.rs

Abstract.

This paper presents a modified variant of genetic algorithm for training convolutional architectures which reduces the execution time of the algorithm. Modification is based on changing the evolutional segment of the algorithm by focusing on limiting the training time of each individual and incorporating the learnt knowledge of neuron parameters from the previous generations into each new one. By doing so the evolution is made more efficient, thus reducing the time needed to find the desired architecture.

Additional contribution of this paper is creating new dataset *DoubledMNIST*, which represents a successor of the popular MNIST dataset. Created dataset is doubled with respect to the MNIST dataset both in terms of the number of instances and in terms of the resolution of each individual isntance. Results shown in the paper were obtained using the presented improved method on the created dataset. The paper also shows classification results on the said dataset.

Keywords: genetic algorithm; CNN arhitectures; MNIST dataset; DoubledMNIST dataset

1. Introduction

2. Related work

da spomenemo izvor ideje?

This section provides background about offline handwritten character datasets and about incorporating genetic algorithms with the learning of CNN architectures, and their training.

2.1. Offline Handwritting Datasets

2.2. Genetic CNN

With the rise in number of layers in the CNNs, deeper neural networks are more difficult to train, and using skip connections proved invaluable and allowed for easier training of substantially deeper networks. [6]. Those skip connections enable identity functions to be learned easily where needed. Those skip connections can be manually selected, but since the number of possible skip connections grows exponentially, and because evaluating each model can take a long time, in practice its impossible to try each possible architecture. Many handcrafted CNN architectures exist, but since manually searching the space of all architectures is impractical, it gives a great incentive for automatic search for a good architecture.

A possible solution for finding a good architecutre automatically is by using some kind of metaheuristic. One paper proposes an encoding method of representing each network structure as a fixed length binary string [4]. They define genetic operations: selection, mutation and crossover to generate better suited individuals and eliminate weak ones. The fitness of an individual is defined through as their recognition accuracy, which is gathered through evaluation on a given reference dataset. The learnt structures are transferrable to other datasets image-baseed datasets.

Another approach focuses on automatically constructing CNN architectures for an image classification task based based on Cartesian genetic programming (CGP). They use convolutional blocks and tensor concatenation as the node functions in CGP. Their results are comparable with handcrafted state-of-the-art models [5].

3. DoubledMNIST Dataset

4. Network representation

Many network architectures, such as [6, 7], can be partitioned into stages. In each stage the width, height and depth of the data cube remain unchanged. Adjacent stages are connected with pooling layers which reduce both width and height of the data. The polling usually performs $2x2$ max polling operation, which halfs the width and height by transforming each $2x2$ pixel group into one pixel whose value is the maximum of the pixel group. The number of channels (filters) within convolutions of the same stage is the same.

That observation is leveraged to define a set of networks which can be encoded into a binary string of fixed length. In this scenario, a CNN is composed of S stages, each containing K_i nodes ($i = 1, ..., S$). The nodes within a stage are ordered, and connections are only allowed from a lower number towards a higher numbered node. Each node represents a convolutional operation. If a node has multiple inputs, then they are all summed up element-wise. After convolution the ReLU activation is performed. The fully connected end of the network is not encoded, as its hyperparameters are static.

Since each node can be an input only to higher nubmered nodes, that means that $(K_i - 1) + (K_i - 2) + ... + 2 + 1 = \frac{K_i(K_i-1)}{2}$ bits are needed to encode a stage containing K_i nodes.

Some encodings would represent invalid networks, thus default nodes are introduced in each stage. They are the first and the last nodes in the stage, and since they're always present, they aren't encoded in the binary string.

There are a few special cases:

- If a node does not have any input or output, then it is removed.
- If a node does not have an input, then we connect the first node in the stage to it.
- If a node does not have an output, then we connect it to the last node in the stage.

5. Method

The core idea of the genetic algorithm is to get a good solution to a problem by generating increasingly better solutions through the process of evolution. Network architecture is encoded as a binary string of fixed length. That string represents a gene of an individual. Individuals from the population have a higher chance to pass on their genes to the next generation if they are more fit for a given task. Through many generations its expected to arrive to a population that has many good individuals and the best individual out of that last generation represents the solution.

The evolution process consists of selection, mutation and crossover. The selection process allows stronger individuals to be perserved, and for weaker individuals to be eliminated. Crossover process combines genes of two individuals to create individuals for the new generation. Mutation randomly changes a gene of an individual, thus introducing more variety within a population.

Algorithm 1: Genetic algorithm for generating the appropriate network architecture

Input: the testing dataset T, number of generations G, number of individuals in each generation N, mutation and crossover probabilities p_M and p_C, mutation parameter q_M and crossover parameter q_C.
Initialization: generate randomized individuals, train them and compute their fitness (evaluate classification accuracy);
for t = *1,...,G* **do**
  Selection: perform selection using a rulet method to p;
  Crossover: perform crossover with probability p_C and crossover parameter q_c;
  Mutation: perform mutation on individuals which have not had crossover with probability p_M and mutation parameter q_M;
  Construction: construct CNN from the gene encoding it;
  Inheritance: inherit the stage weights from the most similar individual of the last generation;
  Training: train the constructed networks, where number of epochs depends on number of inherited stages;
  Evaluation: evaluate all individuals to get their fitness;
end
Output: individuals of the final generation and their classification accuracy.

We denote the n-th individual in generation t as $M_{t,n}$ and fitness of that individual as $f_{t,i}$.

5.1. Initialization phase

The initial generation of individuals is generated by assigning each bit in the binary string a random value from Bernoulli distribution $\mathcal{B}(0.5)$. Then all individuals from the initial generation are fully trained on the training dataset and evaluated on a testing dataset to get their fitness. After that, the genetic evolution process is repeated for a set number of generations.

5.2. Selection phase

The most common selection methods are roulet selection and tournament selection. Here we use roulet selection where the least fit individual is always eliminated. In roulet selection, each individual has a chance to pass on its genes to the next generation, where the probability of that event is proportional to the individuals fitness in comparison to the fitness of all other individuals. The sampling is performed N times (number of individuals in each generation) with replacement, thus each individual may be selected multiple times. The probability of individual $M_{t,n}$ passing the selection is equal to $f_{t,i}/\sum_{i=1}^{N}(f_{t,i} - f_{t,min})$, where $f_{t,min} = \min_{i=1}^{N} f_{t,i}$.

5.3. Crossover phase

The crossover process combines the genes of two individuals to create one or two new individuals. Here crossover of two individuals always results in two new individuals. When two individuals are in a crossover they swap a whole stage. That way the learned connections within a stage are kept through the generations, while still introducing more variaty in individuals. Candidates for crossover are pairs of individuals $(M_{t,2i-1}, M_{t,2i}), \forall i = 1, .., \lfloor N/2 \rfloor$. The probability of crossover between two individuals is p_C, and the probability of two stages being swapped is q_C.

5.4. Mutation phase

Mutation can occur only if an individual didn't go through the crossover process. In that case the individual starts going through mutation with probability p_M. Then each bit in the individuals string representation has a low chance of being inverted, defined in q_M. Mutation ensures additional variaty in individuals and allows for exploring different architectures within each stage.

5.5. Construction phase

The binary encoded string of each individual is parsed and the graph is constructed, where graph nodes represent the convolution operations. Connections withing graph nodes signal that there is a connection between those two nodes. The whole CNN is constructed by parsing the binary string according

to the representation discussed in 4. The end of the network always consists of a flatten layer, followed by a dense (fully connected) layer with 32 units, and finally a dense layer with softmax activation and number of units equal to the number of possible classes.

5.6. Inheritance phase

The novelty in this approach stems from the observation that there isn't always a need to train each network from scratch. In the most trivial case, the network in the new generation is identical to the network in the last generation and there is no need to train it again. In that situation, the weights of the new individual can be inherited (copied) from the individual of the last generation. Though the probability of the trivial case is small, we can extend the idea to other cases.

In case the new individual has the first n out of m (where $n < m$) stages the same as some individual from the last generation, we can still leverage that similarity by inheriting the stages from an individual of the last generation that are unchanged in the individual of the new generation. In this instance, the network already has learnt the lower level features, and only the last $m - n$ stages need to be trained. The first n stages are not frozen so that they can freely adapt to the new architecture.

Individuals which have many identical segments as an individual in previous generation, but do not have the first segment identical to that individual do not inherit any weights. This is motivated by the fact that the latter layers in the network are adapted to outputs of previous layers, and if those outputs of previous layers were to change, the learnt parameters of the latter layers wouldn't be of any use. For that reason we only allow inheriting the stages which were unchanged from the first stage.

Since the lower stages are already trained, the network needs less time to converge, thus we can reduce the number of epochs in training the network. That observation leads to an implementation that allows us to double the execution time of the algorithm that performs architecture search. The results are shown in section 6.

5.7. Training phase

Training is performed on a constructed CNN model for a set amount of epochs. The more stages were inherited in inheritance phase, the less number of epochs is needed to train the model. Training is done on the training dataset which is the same for all individuals.

5.8. Evaluation phase

Evaluation phase is performed to get the fitness of an individual. The trained model is evaluated on a testing dataset to get its classification accuracy, which is used as its fitness.

6. Evaluation

7. Conclusion

References

[1] **Cohen, G., Afshar, S., Tapson, J., van Schaik, A.** EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373.*, 2017.

[2] **Floreano, D., Dürr, P., Mattiussi, C.** Neuroevolution: from architectures to learning. *Evolutionary intelligence*, 1(1), 47-62, 2008.

[3] **Voß, S., Martello, S., Osman, I. H., Roucairol, C. (Eds.).** Meta-heuristics: Advances and trends in local search paradigms for optimization. *Springer Science and Business Media.*, 2012.

[4] **Xie, L., Yuille, A.** Genetic cnn. *In Proceedings of the IEEE International Conference on Computer Vision (pp. 1379-1388).*, 2017.

[5] **Suganuma, M., Shirakawa, S., Nagao T.** A genetic programming approach to designing convolutional neural network architectures. *In Proceedings of the Genetic and Evolutionary Computation Conference (pp. 497-504).*, 2017.

[6] **He, K., Zhang, X., Ren S., Sun J.** Deep Residual Learning for Image Recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] **Simonyan K., Zisserman A.** Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, 2014.