Multi-objective parametric synthesis of loss functions in neural networks with evolutionary algorithms

Eduard Morozov[1*], and *Vladimir* Stanovov[2]

[1]Reshetnev Siberian State University of Science and Technology 31, Krasnoyarskii rabochii prospekt, Krasnoyarsk, 660037, Russian Federation
[2]Siberian Federal University 79, Svobodnii ave., Krasnoyarsk, 660041, Russian Federation

Abstract. The loss function is a fundamental aspect of neural network training and by choosing a suitable one, better results can be achieved. In classification problems, the cross–entropy loss function is almost exclusively used. In this paper the loss function represented by Taylor's series which are optimized with multi-objective evolutionary algorithm. As results show the new loss function can be better than cross-entropy, however application of multi-objective algorithm does not bring an improvement in comparison with single-objective algorithm.

1 Introduction

Today, neural networks are a quite popular tool for solving various types of problems. A significant increase in computing power made it possible to train models of increasing complexity. However, such models can be extremely complicated: there are no intuitively interacting components in them, and they are difficult for human perception.

Loss functions are a fundamental element for the learning process, serving as the main target according to which the parameters of the model are optimized. The choice of the loss function, i.e. the way in which learning and generalization occurs, implicitly affects the optimization process. However, in modern machine learning, most practitioners rely on several standard loss functions.

Despite the fact that there is now a wide range of work on optimizing many aspects of neural networks, the dynamics of training are still usually set manually without strict justification. However, it is important: the architecture, loss functions, and learning rate all affect the final functionality of the neural network.

In one of the works by Santiago Gonzalez and Risto Mikkulainen [1], the loss function was replaced by a simplified two-dimensional decomposition of the Taylor series of the k-th order. Their work used a single-objective approach, with the help of which they were able to achieve good results.

* Corresponding author: morozoveduardmsd@gmail.com

Thanks to the new parameterization of loss functions, the key pieces of information that affect the behavior of the loss function are compactly represented in the vector. Such vectors are then optimized for a specific task using evolutionary algorithms.

The reason for the performance improvement is that the evolved functions prevent class labels from being overtrained, resulting in automatic regularization. These improvements are especially noticeable when shrinking datasets, where such regularization matters most.

2 Related work

Applying deep neural networks to new tasks often involves significant manual tuning of the network design. Metalearning is relatively recent, and it allows us to solve the problem of parameter tuning algorithmically. Most of the work was focused on the search for architecture or flexible hyperparameter tuning [2]. Other useful aspects for optimization, such as activation functions and learning algorithms, are now highlighted [3].

Since loss functions are a fundamental element of neural network training, it is highly recommended to use metalearning to develop them. Deep neural networks are trained iteratively by updating model parameters using negative gradients.

The learning process starts with an error given by a loss function, thanks to the losses, the algorithm shifts the weights to the side in which the error has the smallest value.

Many problems use the same loss functions, for example, for example, in classification problems, the cross–entropy loss function is almost exclusively used. While some approaches use the regularization [4].

Genetic Loss Optimization (GLO) [5] provided an initial approach to meta-learning of loss functions. As described above, GLO is based on tree representations with coefficients. GLO was able to come up with Baikal, a new loss function that surpassed cross-entropy loss in image classification tasks.

However, because the structure and coefficients are optimized separately in the GLO, their interaction cannot be easily optimized. Many features created by tree search are useless because they have gaps and mutations can have a disproportionate impact on function. As such, GLO are inefficient because it requires large populations that should evolve over many generations.

In this work the new technique presented, which uses genetic loss optimization on Taylor series and aims to solve these problems through a novel loss function parameterization based on multivariate Taylor expansions with multi-objective algorithm.

3 Proposed algorithm

In this paper, the loss function categorical cross-entropy (1) is replaced by the expansion of the Taylor series. To do this, a function was implemented using the formula (2)

$$D(x, y) = -\sum_j x_j \ln y_j, \tag{1}$$

The relationship between x and y can be seen in Figure 1.

$$f(x,y) = -\frac{1}{n}\sum_{i=1}^{n}\left[\begin{array}{l}\theta_2(y_i-\theta_1)^3+\frac{1}{2}\theta_3\left(y_i-\theta_1\right)^2+\frac{1}{6}\theta_4(y_i-\theta_1)^3+\theta_5(x_i-\theta_0)(y_i-\theta_1)+\\ +\frac{1}{2}\theta_6(x_i-\theta_0)(y_i-\theta_1)^2+\frac{1}{2}\theta_7(x_i-\theta_0)^2(y_i-\theta_1)\end{array}\right], \tag{2}$$

where x – the true label of the class, y – predicted class label. And $\theta_0 - \theta_7$ are the coefficients to be found.

To find coefficients, you can use the 0th order method, evolutionary algorithms fall into this category.

For the operation of the algorithm, was chosen NSGA-II [6] which use operators from Differential Evolution [7]. This algorithm is used to solve multi-criteria problems. The NSGA-II method uses an elitist strategy, eliminates the need to adjust the parameter responsible for maintaining diversity in the population, and uses a more efficient sorting mechanism.

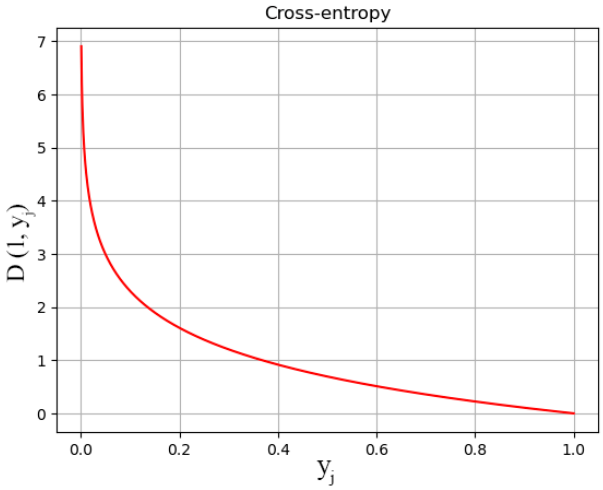

Fig. 1. Graph of cross-entropy, the smaller the error, the greater the accuracy and vice versa.

In the first stage of the algorithm, the population is initialized in the same way as in differential evolution. With the help of a special operator, each individual receives his own rank. An individual's rank is obtained according to how many other individuals dominate him, or in case he is not dominated by anyone, he gets the lowest rank.

A group of individuals belonging to the same front is called a front, all the individuals on the front are not dominating each other. Subsequently, during the first pass of the operator, the existence of group of dominate individuals is erased and the next front is formed. That is, individuals of the 2nd rank are dominated by the 1st rank, as shown in Figure 2.

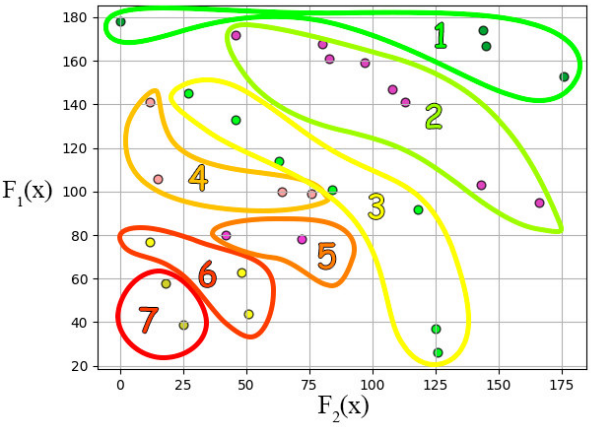

Fig. 2. Distribution of solutions on Pareto fronts. An abstract integer value was taken as the objective functions. Solutions on the 1st front are better (dominant) than those on the other fronts.

At the beginning of the algorithm, a starting population of *x_massive* coefficients is initialized, such as the one in Figure 3. Each number corresponds to a coefficient $[\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7]$ in loss function.

$$[5.9, 3.7, 1, 6.2, 2.5, 8.2, 1.3, 4.1]$$

Fig. 3. Randomly generated solution of the population, denoted as *x_massive*.

After that, we train a neural network with each of the solutions in the population using a loss function based on the Taylor series. As a result, we will obtain accuracy for each solution in the training set and in the validation set. These values will be the values of the objective functions of the genetic algorithm. Next, the work of mutation, crossover, and selection operators takes place. We put the resulting population in the place of the parent population and the process is repeated until the number of generations specified at the beginning as a hyperparameter ends. The whole cycle is illustrated in Figure 4.

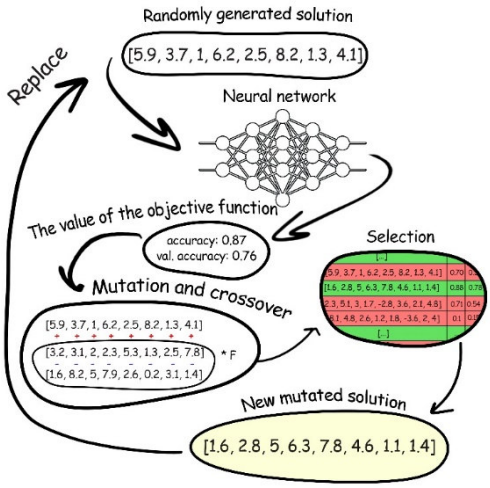

Fig. 4. The process of the algorithm.

This algorithm has been implemented and tested on classification problems such as MNIST [8], CIFAR10 [9], and SVHN [10]. The results of the studies are presented in Table 1.

Table 1. The accuracy of the test sets of loss functions detected by Taylor GLO is compared with the accuracy of cross-entropy losses. TaylorGLO's results are based on a loss function with the highest verification accuracy during evolution. All average values are taken from ten separately trained models.

Task and Model	Avg. TaylorGLO Acc., %	Avg. Baseline Acc., %	Student t-test
MNIST, fully connected	98.00 (0.06)	97.64 (0.26)	+
CIFAR10, CNN	77.81 (0.8)	77.26 (0.35)	=
SVHN, CNN	95.86 (0.07)	95.6 (0.26)	+

The coefficients used to obtain the result are presented in Table 1 and Table 2. When comparing the use of the trained loss function and cross-entropy, graphs for the CIFAR10 problems were obtained (Figure 5, Figure 6).

Table 2. The coefficients obtained during training on the CIFAR10 task were trained in 100 iterations of the algorithm with a population of 15 on a GeForce GTX 2060 graphics card, using the Tensorflow 2.10 and Keras libraries. The NSGA-II algorithm was implemented in Python 3.9.

θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7
-1.694	3.338	0.340	-0.162	4.553	0.056	-2.396	1.635

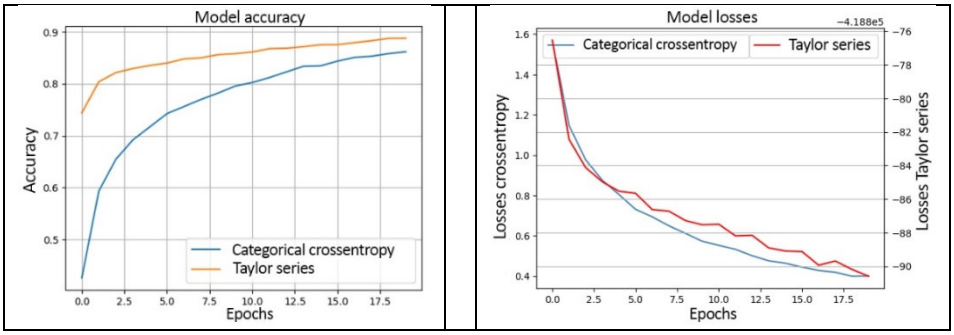

Fig. 5. Results on the training dataset.

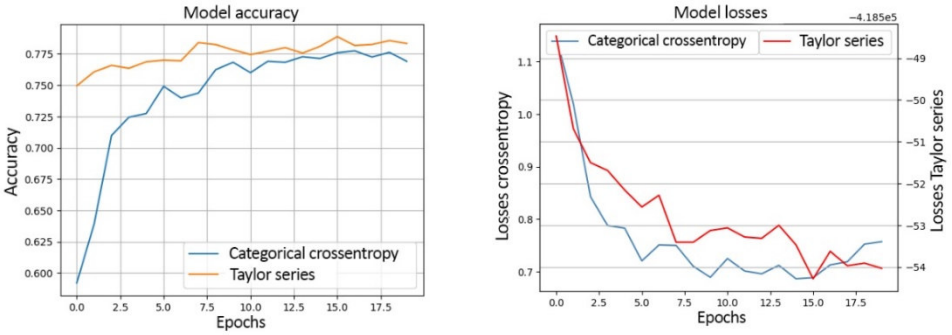

Fig. 6. Results on the validation dataset.

Using the coefficients from Table 2, it is possible to plot the loss function for two classes instead of 10 as in problem CIFAR10, using formula 3.

$$f(x,y) = -\frac{1}{2} * \left[\begin{array}{l} (\theta_2(y-\theta_1)^3 + \frac{1}{2}\theta_3(y-\theta_1)^2 + \frac{1}{6}\theta_4(y-\theta_1)^3 + \theta_5(x-\theta_0)(y-\theta_1) + \frac{1}{2}\theta_6(x-\theta_0)(y-\theta_1)^2 + \\ + \frac{1}{2}\theta_7(x-\theta_0)^2(y-\theta_1)) + (\theta_2(1-y-\theta_1)^3 + \frac{1}{2}\theta_3(1-y-\theta_1)^2 + \frac{1}{6}\theta_4(1-y-\theta_1)^3 + \\ + \theta_5(1-x-\theta_0)(1-y-\theta_1) + \frac{1}{2}\theta_6(1-x-\theta_0)(1-y-\theta_1)^2 + \frac{1}{2}\theta_7(1-x-\theta_0)^2(1-y-\theta_1)) \end{array} \right] \quad (3)$$

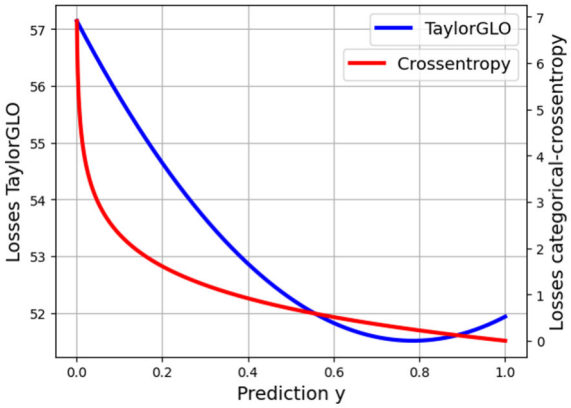

Fig. 7. Comparison of cross entropy and TaylorGLO plots at $x_0 = 1$.

The observed values of the optimized loss function are significantly greater than those of cross-entropy.

4 Conclusion

Thus, the studies have shown that the implemented approach is capable of parametric synthesis of effective loss functions. Moreover, it can be seen that when cross-entropy is used, the graph of losses begins to fluctuate after several epochs, after which it begins to grow, which means that the model begins to overfit. While in the graph of the parametric synthesis model, the loss function continues to decrease, which means that the network is less susceptible to overfitting. The disadvantage of this method is the high requirements for computing resources. The advantage is greater accuracy compared to cross-entropy.

Based on the results, we can say that it was possible to implement the approach, but the chosen algorithm does not give a result higher than CMA-ES, which means that the use of multi-objective optimization in the meta-training of the loss function is inefficient.

The efficiency turned out to be the same due to the fact that one of the criteria was not sufficiently consistent, namely accuracy on the training sample, since this criterion is less important relative to accuracy on the validation sample. In some moments, it may prevent you from getting a better result.

This work was supported by the Ministry of Science and Higher Education of the Russian Federation (Grant No.075-15-2022-1121).

References

1. S. Gonzalez, R. Miikkulainen, *Optimizing loss functions through multi-variate taylor polynomial parameterization,* Proceedings of the Genetic and Evolutionary Computation Conference (2021)
2. J.T. Springenberg, A. Dosovitskiy, T. Brox, M.A. Riedmiller, Striving for Simplicity: The All Convolutional Net. CoRR, abs/1412.6806 (2014)
3. G. Bingham, W. Macke, R. Miikkulainen, *Evolutionary optimization of deep learning activation functions,* Proceedings of the 2020 Genetic and Evolutionary Computation Conference (2020)

4. A.I. Tikhonov. Solution of Incorrectly Formulated Problems and the Regularization Method (1963)
5. S. Gonzalez, R. Miikkulainen, *Improved Training Speed, Accuracy, and Data Utilization Through Loss Function Optimization,* 2020 IEEE Congress on Evolutionary Computation (CEC), 1-8 (2019)
6. Deb, Kalyanmoy et al., *A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation: NSGA-II*, Parallel Problem Solving from Nature (2000)
7. Storn Rainer, Kenneth V. Price, Journal of Global Optimization **11**, 341-359 (1997)
8. L. Deng, IEEE Signal Processing Magazine **29(6)**, 141-142 (2012)
9. Alex Krizhevsky, Vinod Nair, Geoffrey Hinton "The CIFAR-10 dataset " (2014). http://www. cs. toronto. edu/kriz/cifar. html 55.5.
10. Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)