Optimizing Loss Functions Through Multivariate Taylor Polynomial Parameterization

Santiago Gonzalez[1,2] and **Risto Miikkulainen**[1,2]
[1]Cognizant Technology Solutions, *San Francisco, California, USA*
[2]Department of Computer Science, University of Texas at Austin, *Austin, Texas, USA*
Email: slgonzalez@utexas.edu, risto@cs.utexas.edu

Abstract

Metalearning of deep neural network (DNN) architectures and hyperparameters has become an increasingly important area of research. Loss functions are a type of metaknowledge that is crucial to effective training of DNNs, however, their potential role in metalearning has not yet been fully explored. Whereas early work focused on genetic programming (GP) on tree representations, this paper proposes continuous CMA-ES optimization of multivariate Taylor polynomial parameterizations. This approach, TaylorGLO, makes it possible to represent and search useful loss functions more effectively. In MNIST and CIFAR-10 benchmark tasks, TaylorGLO finds new loss functions that outperform functions previously discovered through GP, as well as the standard cross-entropy loss, in fewer generations. These functions serve to regularize the learning task by discouraging overfitting to the labels, which is particularly useful in tasks where limited training data is available. The results thus demonstrate that loss function optimization is a productive new avenue for metalearning.

1 Introduction

As deep learning systems have become more complex, their architectures and hyperparameters have become increasingly difficult and time-consuming to optimize by hand. In fact, many good designs may be overlooked by humans with prior biases. Therefore, automating this process, known as metalearning, has become an essential part of the modern machine learning toolbox. Metalearning approaches this problem from numerous angles, both by optimizing different aspects of the architecture from hyperparameters to topologies, and by using different approaches from Bayesian optimization to evolutionary computation (35; 6; 31; 28).

Recently, loss-function discovery and optimization has emerged as a new type of metalearning. It aims to tackle neural network's root training goal, by discovering better ways to define what is being optimized. However, loss functions can be challenging to optimize because they have a discrete nested structure as well as continuous coefficients. The first system to do so, Genetic Loss Optimization (GLO) (11) tackled this problem by discovering and optimizing loss functions in two separate steps: (1) representing the structure as trees, and evolving them with Genetic Programming (GP) (2); and (2) optimizing the coefficients using Covariance-Matrix Adaptation Evolutionary Strategy (CMA-ES) (16). Such separate processes make it challenging to find a mutually optimal structure and coefficients. Furthermore, small changes in the tree-based search space do not always result in small changes in the phenotype, and can easily make a function invalid, making the search process ineffective.

In an ideal case, loss functions would be smoothly mapped into arbitrarily long, fixed-length vectors in a Hilbert space. This mapping should be smooth, well-behaved, well-defined, incorporate both a function's structure and coefficients, and should by its very nature make large classes of infeasible loss functions mathematically impossible. This paper introduces such an approach: *Multivariate Taylor*

Preprint. Under review.

expansion-based genetic loss-function optimization (TaylorGLO). With a novel parameterization for loss functions, the key pieces of information that affect a loss function's behavior are compactly represented in a vector. Such vectors are then optimized for a specific task using CMA-ES. Select techniques can be used to narrow down the search space and speed up evolution.

Loss functions discovered by TaylorGLO outperform the standard cross-entropy loss (or log loss) on both the MNIST and CIFAR-10 datasets with several different network architectures. They also outperform the Baikal loss, discovered by the original GLO technique, and do it with significantly fewer function evaluations. The reason for the improved performance is that evolved functions discourage overfitting to the class labels, thereby resulting in automatic regularization. These improvements are particularly pronounced with reduced datasets where such regularization matters the most. TaylorGLO thus further establishes loss-function optimization as a promising new direction for metalearning.

2 Related work

Applying deep neural networks to new tasks often involves significant manual tuning of the network design. The field of metalearning has recently emerged to tackle this issue algorithmically (35; 28; 6; 31). While much of the work has focused on hyperparameter optimization and architecture search, recently other aspects such activation functions and learning algorithms have been found useful targets for optimization (3; 32). Since loss functions are at the core of machine learning, it is compelling to apply metalearning to their design as well.

2.1 Loss function metalearning

Deep neural networks are trained iteratively, by updating model parameters (i.e., weights and biases) using gradients propagated backwards through the network, starting from an error given by a loss function (33). Loss functions represent the primary training objective of a neural network. In many tasks, such as classification and language modeling, the cross-entropy loss (also known as the log loss) has been used almost exclusively. While in some approaches a regularization term (e.g. L^2 weight regularization (40)) is added to the the loss function definition, the core component is still the cross-entropy loss. This loss function is motivated by information theory: It aims to minimize the number of bits needed to identify a message from the true distribution, using a code from the predicted distribution.

In other types of tasks that do not fit neatly into a single-label classification framework different loss functions have been used successfully (10; 8; 24; 42; 5). Indeed, different functions have different properties; for instance the Huber Loss (21) is more resilient to outliers than other loss functions. Still, most of the time one of the standard loss functions is used without a justification, and there is an opportunity to improve through metalearning.

Genetic Loss Optimization (GLO) (11) provided an initial study into metalearning of loss functions. As described above, GLO is based on tree-based representations with coefficients. Such representations have been dominant in genetic programming because they are flexible and can be applied to a variety of function evolution domains. GLO was able to discover Baikal, a new loss function that outperformed the cross-entropy loss in image classification tasks. However, because the structure and coefficients are optimized separately in GLO, it cannot easily optimize their interactions. Many of the functions created through tree-based search are not useful because they have discontinuities, and mutations can have disproportionate effects on the functions. GLO's search is thus inefficient, requiring large populations that are evolved for many generations.

The technique presented in this paper, TaylorGLO, aims to solve these problems through a novel loss function parameterization based on multivariate Taylor expansions. Furthermore, since such representations are continuous, the approach can take advantage of CMA-ES (16) as the search method, resulting in faster search.

2.2 Multivariate Taylor expansions

Taylor expansions (39) are a well-known function approximator that can represent differentiable functions within the neighborhood of a point using a polynomial series. Below, the common univariate

Taylor expansion formulation is presented, followed by a natural extension to arbitrarily-multivariate functions.

Given a $C^{k_{\text{max}}}$ smooth (i.e., first through k_{max} derivatives are continuous), real-valued function, $f(x) : \mathbb{R} \to \mathbb{R}$, a kth-order Taylor approximation at point $a \in \mathbb{R}$, $\hat{f}_k(x, a)$, where $0 \leq k \leq k_{\text{max}}$, can be constructed as

$$\hat{f}_k(x, a) = \sum_{n=0}^{k} \frac{1}{n!} f^{(n)}(a)(x - a)^n. \tag{1}$$

Conventional, univariate Taylor expansions have a natural extension to arbitrarily high-dimensional inputs of f. Given a $C^{k_{\text{max}}+1}$ smooth, real-valued function, $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, a kth-order Taylor approximation at point $\mathbf{a} \in \mathbb{R}^n$, $\hat{f}_k(\mathbf{x}, \mathbf{a})$, where $0 \leq k \leq k_{\text{max}}$, can be constructed. The stricter smoothness constraint compared to the univariate case allows for the application of Schwarz's theorem on equality of mixed partials, obviating the need to take the order of partial differentiation into account.

Let us define an nth-degree multi-index, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, where $\alpha_i \in \mathbb{N}_0$, $|\alpha| = \sum_{i=1}^{n} \alpha_i$, $\alpha! = \prod_{i=1}^{n} \alpha_i!$. $\mathbf{x}^\alpha = \prod_{i=1}^{n} x_i^{\alpha_i}$, and $\mathbf{x} \in \mathbb{R}^n$. Multivariate partial derivatives can be concisely written using a multi-index

$$\partial^\alpha f = \partial_1^{\alpha_1} \partial_2^{\alpha_2} \cdots \partial_n^{\alpha_n} f = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}. \tag{2}$$

Thus, discounting the remainder term, the multivariate Taylor expansion for $f(\mathbf{x})$ at $\mathbf{a}$ is

$$\hat{f}_k(\mathbf{x}, \mathbf{a}) = \sum_{\forall \alpha, |\alpha| \leq k} \frac{1}{\alpha!} \partial^\alpha f(\mathbf{a})(\mathbf{x} - \mathbf{a})^\alpha. \tag{3}$$

The unique partial derivatives in $\hat{f}_k$ and $\mathbf{a}$ are parameters for a kth order Taylor expansion. Thus, a kth order Taylor expansion of a function in n variables requires n parameters to define the center, $\mathbf{a}$, and one parameter for each unique multi-index α, where $|\alpha| \leq k$. That is: $\#_{\text{parameters}}(n, k) = n + \binom{n+k}{k} = n + \frac{(n+k)!}{n!\,k!}$.

The multivariate Taylor expansion can be leveraged for a novel loss-function parameterization. It enables TaylorGLO, a way to efficiently optimize loss functions, as will be described in subsequent sections.

3 Loss functions as multivariate Taylor expansions

Let an n-class classification loss function be defined as $\mathcal{L}_{\text{Log}} = -\frac{1}{n} \sum_{i=1}^{n} f(x_i, y_i)$. The function $f(x_i, y_i)$ can be replaced by its kth-order, bivariate Taylor expansion, $\hat{f}_k(x, y, a_x, a_y)$. More sophisticated loss functions can be supported by having more input variables, beyond x_i and y_i, such as a time variable or unscaled logits. This approach can be useful, for example, to evolve loss functions that change as training progresses.

For example, a loss function in $\mathbf{x}$ and $\mathbf{y}$ has the following 3rd-order parameterization with parameters θ (where $\mathbf{a} = \langle \theta_0, \theta_1 \rangle$):

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{y}) = -\frac{1}{n} \sum_{i=1}^{n} \Big[& \theta_2 + \theta_3(y_i - \theta_1) + \tfrac{1}{2}\theta_4(y_i - \theta_1)^2 + \tfrac{1}{6}\theta_5(y_i - \theta_1)^3 + \theta_6(x_i - \theta_0) \\ & + \theta_7(x_i - \theta_0)(y_i - \theta_1) + \tfrac{1}{2}\theta_8(x_i - \theta_0)(y_i - \theta_1)^2 + \tfrac{1}{2}\theta_9(x_i - \theta_0)^2 \\ & + \tfrac{1}{2}\theta_{10}(x_i - \theta_0)^2(y_i - \theta_1) + \tfrac{1}{6}\theta_{11}(x_i - \theta_0)^3 \Big] \end{aligned} \tag{4}$$

Notably, the reciprocal-factorial coefficients can be integrated to be a part of the parameter set by direct multiplication if desired.

As will be shown in this paper, the technique makes it possible to train neural networks that are more accurate and learn faster, than those with tree-based loss function representations. Representing loss functions in this manner confers several useful properties:

- Guarantees smooth functions;
- Functions do not have poles (i.e., discontinuities going to infinity or negative infinity) within their relevant domain;

- Can be implemented purely as compositions of addition and multiplication operations;
- Can be trivially differentiated;
- Nearby points in the search space yield similar results (i.e., the search space is locally smooth), making the fitness landscape easier to search;
- Valid loss functions can be found in fewer generations and with higher frequency;
- Loss function discovery is consistent and not dependent on a specific initial population; and
- The search space has a tunable complexity parameter (i.e., the order of the expansion).

These properties are not necessarily held by alternative function approximators. For instance:

Fourier series are well suited for approximating periodic functions (7), therefore, they are not as well suited for loss functions, whose local behavior within a narrow domain is important. Being a composition of waves, Fourier series tend to have many critical points within the domain of interest. Gradients fluctuate around such points, making gradient descent infeasible. Additionally, close approximations require a large number of terms, which in itself can be injurious, causing large, high-frequency fluctuations, known as "ringing", due to Gibb's phenomenon (41).

Padé approximants can be more accurate approximations than Taylor expansions; indeed, Taylor expansions are a special case of Padé approximants where $M = 0$ (12). However, unfortunately, Padé approximants can model functions with one or more poles, which valid loss functions typically should not have. These problems still exist, and are exacerbated, for Chisholm approximants (4) (a bivariate extension) and Canterbury approximants (13) (a multivariate generalization).

Laurent polynomials can represent functions with discontinuities, the simplest being x^{-1}. While Laurent polynomials provide a generalization of Taylor expansions into negative exponents, the extension is not useful because it results in the same issues as Padé approximants.

Polyharmonic splines seem like an excellent fit since they can represent continuous functions within a finite domain. However, the number of parameters is prohibitive in multivariate cases.

The multivariate Taylor expansion is therefore a better choice than the alternatives. It makes it possible to optimize loss functions efficiently in TaylorGLO, as will be described next.

4 The TaylorGLO approach

TaylorGLO (Figure 1) aims to find the optimal parameters for a loss function parameterized as a multivariate Taylor expansion, as described in Section 3. The parameters for a Taylor approximation (i.e., the center point and partial derivatives) are referred to as $\theta_{\hat{f}}$. $\theta_{\hat{f}} \in \Theta$, $\Theta = \mathbb{R}^{\#_{\text{parameters}}}$. TaylorGLO strives to find the vector $\theta^*_{\hat{f}} \in \Theta$ that parameterizes the optimal loss function for a task. Because the values are continuous, as opposed to discrete graphs of the original GLO, it is possible to use continuous optimization methods.

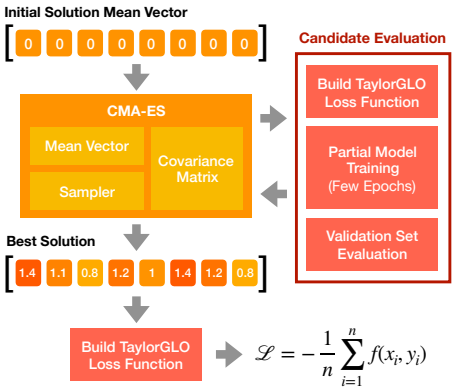


Figure 1: The TaylorGLO approach. Starting with an initial, unbiased solution, a CMA-ES iteratively attempts to maximize on the TaylorGLO loss function parameterization's partial-training validation accuracy. The candidate with the highest fitness is chosen as the final, best solution.

In particular, Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) (16) is a popular population-based, black-box optimization technique for rugged, continuous spaces. CMA-ES functions by maintaining a covariance matrix around a mean point that represents a distribution of solutions. At each generation, CMA-ES adapts the distribution to better fit evaluated objective values from sampled individuals. In this manner, the area in the search space which is being sampled at each step dynamically grows, shrinks, and moves as needed to maximize sampled candidates' fitnesses. The specific variant of CMA-ES that TaylorGLO uses is $(\mu/\mu, \lambda)$-CMA-ES (17), which incorporates weighted rank-μ updates (15) to reduce the number of objective function evaluations that are needed.

In TaylorGLO, CMA-ES is used to find try to find $\theta^*_{\hat{f}}$. At each generation, CMA-ES samples points in Θ whose fitness is determined; this is accomplished by training a model with the corresponding loss function and evaluating the model on a validation dataset. Fitness evaluations may be distributed across multiple machines in parallel and retried a limited number of times upon failure. An initial vector of $\theta_{\hat{f}} = \mathbf{0}$ is chosen as a starting point in the search space to avoid bias.

Note that fully training a model can prove to be prohibitively expensive for many problems. Fundamentally, there is a positive correlation between performance near the beginning of training and at the end of training. In order to identify the most promising candidates, it is enough to train the models only partially. This type of approximate evaluation is widely done in the field (14; 22). An additional positive effect is that evaluation then favors loss functions that learn more quickly.

For a loss function to be useful, it must have a derivative that depends on the prediction. Therefore, internal terms that do not contribute to $\frac{\partial}{\partial \mathbf{y}} \mathcal{L}_f(\mathbf{x}, \mathbf{y})$ can be trimmed away. This implies that any term, t within $f(x_i, y_i)$, where $\frac{\partial}{\partial y_i} t = 0$, can be replaced with 0.

For example, this refinement simplifies Equation 4, providing a reduction in the number of parameters from twelve to eight:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\frac{1}{n} \sum_{i=1}^{n} \Big[\theta_2(y_i - \theta_1) + \tfrac{1}{2}\theta_3(y_i - \theta_1)^2 + \tfrac{1}{6}\theta_4(y_i - \theta_1)^3 + \theta_5(x_i - \theta_0)(y_i - \theta_1) + \tfrac{1}{2}\theta_6(x_i - \theta_0)(y_i - \theta_1)^2 + \tfrac{1}{2}\theta_7(x_i - \theta_0)^2(y_i - \theta_1) \Big] . \tag{5}$$

5 Experimental setup

This section presents the experimental setup that was used to evaluate the TaylorGLO technique. The standard MNIST (27) digit classification and CIFAR-10 (25) natural image classification benchmark tasks were used as domains to measure the technique's efficacy, and provide a point of comparison against GLO and the standard cross-entropy loss function $\mathcal{L}_{\text{Log}} = -\frac{1}{n} \sum_{i=1}^{n} x_i \log(y_i)$ (where x is sampled from the true distribution, y is from the predicted distribution, and n is the number of classes).

MNIST domain The MNIST task is relatively simple, which makes it possible to illustrate the TaylorGLO process in several ways. The basic CNN architecture evaluated in the GLO study (11) can also be used to provide a direct point of comparison with prior work. Importantly, this architecture includes a dropout layer (20) for explicit regularization. As in GLO, training is based on stochastic gradient descent (SGD) with a batch size of 100, a learning rate of 0.01, and, unless otherwise specified, occurred over 20,000 steps.

CIFAR-10 domain Several different architectures were evaluated in the CIFAR-10 domain, including AlexNet (26), AllCNN-C (37), and Preactivation ResNet-20 (19), which is an improved variant of the ubiquitous ResNet architecture (18). Models were trained with their respective hyperparameters from the literature. Inputs were normalized by subtracting their mean pixel value and dividing by their pixel standard deviation. Standard data augmentation techniques consisting of random, horizontal flips and croppings with two pixel padding were applied during training.

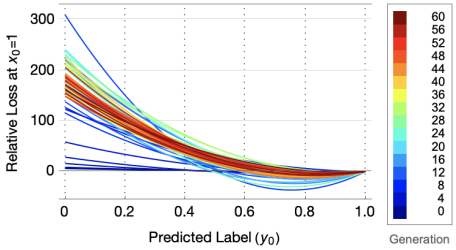
TaylorGLO CMA-ES was instantiated with population size $\lambda = 28$ on MNIST and $\lambda = 20$ on CIFAR-10, and an initial step size $\sigma = 1.2$. These values were found to work well in preliminary experiments. The candidates were third-order (i.e., $k = 3$) TaylorGLO loss functions (Equation 5). Such functions were found experimentally to have a better trade-off between evolution time and performance compared to second- and fourth-order TaylorGLO loss functions (although the differences were relatively small).

Candidate evaluation During candidate evaluation, models were trained for 10% of a full training run on MNIST, equal to 2,000 steps (i.e., four epochs). An in-depth analysis on the technique's sensitivity to training steps during candidate evaluation is provided in Appendix D—overall, the technique is robust even with few training steps. However, on more complex models with abrupt learning rate decay schedules, greater numbers of steps provide better fitness estimates.

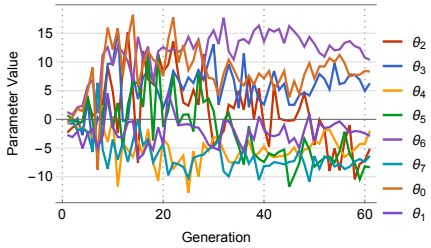
Further experimental setup and implementation details are provided in Appendix A

Table 1: Test-set accuracy of loss functions discovered by TaylorGLO compared with that of the cross-entropy loss baseline. The TaylorGLO results are based on the loss function with the highest validation accuracy during evolution. All averages are from ten separately trained models and p-values are from one-tailed Welch's t-Tests. Standard deviations are shown in parentheses. TaylorGLO discovers loss functions that perform significantly better than cross-entropy loss in all architectures with both datasets.

Task	Avg. TaylorGLO Acc.	Avg. Baseline Acc.	p-value
MNIST on Basic CNN (11)	**0.9951 (0.0005)**	0.9899 (0.0003)	2.95×10^{-15}
CIFAR-10 on AlexNet (26)	**0.7901 (0.0026)**	0.7638 (0.0046)	1.76×10^{-10}
CIFAR-10 on AllCNN-C (37)	**0.9271 (0.0013)**	0.8965 (0.0021)	0.42×10^{-17}
CIFAR-10 on PreResNet-20 (19)	**0.9169 (0.0014)**	0.9153 (0.0021)	0.0400



(a) Best discovered functions over time



(a) Best function parameters over time

Figure 3: The best loss functions (a) and their respective parameters (b)) from each generation of TaylorGLO on MNIST. The functions are plotted in a binary classification modality, showing loss for different values of the network output (y_0 in the horizontal axis) when the correct label is 1.0. The functions are colored according to their generation from blue to red, and vertically shifted such that their loss at $y_0 = 1$ is zero (the raw value of a loss function is not relevant; the derivative, however, is). TaylorGLO explores varying shapes of solutions before narrowing down on functions in the red band; this process can also be seen in (b), where parameters become more consistent over time, and in the population plot of Appendix B. The final functions decrease from left to right, but have a slight increase in the end. This shape is likely to prevent overfitting during learning, which leads to the observed improved accuracy.

6 Results

This section illustrates the TaylorGLO process and demonstrates how the evolved loss functions can improve performance over the standard cross-entropy loss function, especially on reduced datasets. A summary of results on MNIST and CIFAR-10 across a variety of models are shown in Table 1.

6.1 The TaylorGLO discovery process

Figure 2 gives an overview of the evolution process over 60 generations on the MNIST dataset, which is sufficient to reach convergence. TaylorGLO is able to discover highly-performing loss functions quickly, i.e. within 20 generations. Generations' average validation accuracy approaches generations' best accuracy as evolution progresses, indicating that population as a whole is improving. Whereas GLO's unbounded search space often results in pathological functions, every TaylorGLO training session completed successfully without any instabilities.

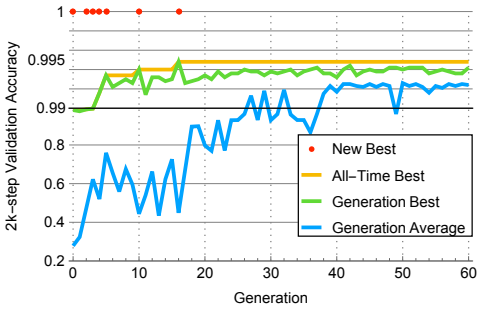


Figure 2: The process of discovering loss functions for MNIST. Red dots mark generations with new improved loss functions. TaylorGLO discovers good functions in very few generations. The best one had a 2000-step validation accuracy of 0.9948, compared to 0.9903 of the cross-entropy loss, averaged over ten runs. This difference translates to a similar improvement on the test set, as shown in Table 1.

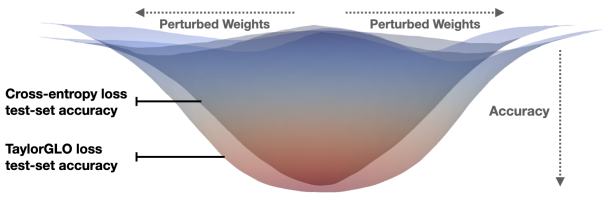


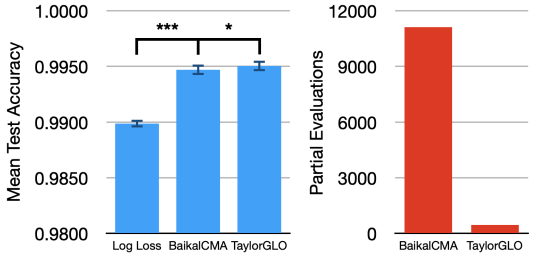
Figure 5: Accuracy basins for AllCNN-C models trained with both cross-entropy and TaylorGLO loss functions. The TaylorGLO basins are both flatter and lower, indicating that they are more robust and generalize better (23), which results in higher accuracy.

Figure 3 shows the shapes and parameters of each generation's highest-scoring loss function. They are plotted as if they were being used for binary classification, using the same procedure as in the GLO study (11). The functions have a distinct pattern through the evolution process, where early generations show a wider variety of shapes that converge in later generations towards curves with a shallow minimum around $y_0 = 0.8$ (the best loss function found on MNIST—described below—had a minimum at $y_0 = 0.8238$). In other words, the loss increases near the correct output, which is counterintuitive. It is also strikingly different from the cross-entropy loss, which decreases monotonically from left to right, as one might expect all loss functions to do. The evolved shape is effective most likely because can provide an implicit regularization effect (11): it discourages the model from outputting unnecessarily extreme values for the correct class, and therefore makes overfitting less likely. This is a surprising finding, and demonstrates the power of machine learning to create innovations beyond human design.

6.2 Performance comparisons

Over 10 fully-trained models, the best TaylorGLO loss function achieved a mean testing accuracy of **0.9951** (stddev 0.0005). In comparison, the cross-entropy loss only reached 0.9899 (stddev 0.0003), and the BaikalCMA loss function discovered by GLO 0.9947 (stddev 0.0003) (11); both differences are statistically significant (Figure 4). Notably, TaylorGLO achieves this result with significantly fewer generations and partial training sessions. BaikalCMA required 11,120 partial evaluations (i.e., 100 individuals over 100 GP generations plus 32 individuals over 35 CMA-ES generations, ignoring evaluations subsequent to the discovery of BaikalCMA), while the top TaylorGLO loss function only required **448** partial evaluations (that is, 4.03% as many). Thus, TaylorGLO achieves improved results with significantly fewer evaluations than GLO.

Such a large reduction in evaluations during evolution allows TaylorGLO to tackle harder problems, including models that have millions of parameters. On the CIFAR-10 dataset, TaylorGLO is able to consistently outperform cross-entropy baselines on a variety models, as shown in Table 1. Interestingly, TaylorGLO also provides more consistent results on CIFAR-10, with accuracy standard deviations nearly half of that of the baselines'. In addition, TaylorGLO loss functions also result in more robust trained models. Using a recent visualization technique (29), accuracy basins for a model can be plotted along a two-dimensional slice (in $[-1, 1]$) of the network's weight space. Figure 5 provides such a visualization on an AllCNN-C model. The TaylorGLO loss function results in a flatter, lower basin. This result suggests that the model is more robust, i.e. its performance is less sensitive to small perturbations in the weight space, and it also generalizes better (23).



(a) Accuracy (b) Evaluations

Figure 4: (a) Mean test accuracy across ten runs on MNIST. The TaylorGLO loss function with the highest validation score significantly outperforms the cross-entropy loss (p-value of 2.95×10^{-15} in a one-tailed Welch's t-test) and BaikalCMA loss from (11) ($p = 0.0313$ in the same test). (b) Required sample partial training evaluations for GLO and TaylorGLO on MNIST. The TaylorGLO loss function was discovered with 96% fewer evaluations than the BaikalCMA loss function with GLO.

6.3 Performance on reduced datasets

The performance improvements that TaylorGLO provides are especially pronounced with reduced datasets. For example, Figure 6 compares accuracies of models trained for 20,000 steps on different portions of the MNIST dataset. Overall, TaylorGLO significantly outperforms the cross-entropy loss. Similar results were obtained with the AllCNN-C model on CIFAR-10. When evolving a TaylorGLO loss function and training against 10% of the training dataset, with 225 epoch evaluations, TaylorGLO reached an average accuracy across ten models of **0.7595** (stddev 0.0062). In contrast, only four out of ten cross-entropy loss models trained successfully, with those reaching a lower average accuracy of 0.6521. Thus, customized loss functions can be especially useful in applications where only limited data is available to train the models, presumably because they are less likely to overfit to the small number of examples.

7 Discussion and future work

TaylorGLO was applied to CIFAR-10 using various standard architectures with standard hyperparameters. These setups have been heavily engineered and manually tuned by the research community, yet TaylorGLO was able to improve them. Interestingly, the improvements were more substantial with wide architectures and smaller with narrow and deep architectures such as the Preactivation ResNet. While it may be possible to further improve upon this result, it is also possible that loss function optimization is more effective with architectures where the gradient information travels through fewer connections, or is otherwise better preserved throughout the network. An important direction of future work is therefore to evolve both loss functions and architectures together, taking advantage of possible synergies between them.

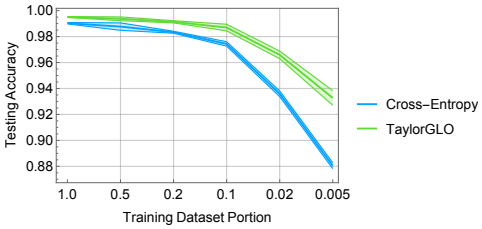


Figure 6: Accuracy with reduced portions of the MNIST dataset. Progressively smaller portions of the dataset were used to train the models (averaging over ten runs). The TaylorGLO loss function provides significantly better performance than the cross-entropy loss on all training dataset sizes, and particularly on the smaller datasets. Thus, its ability to discourage overfitting is particularly useful in applications where only limited data is available.

Another important direction is to leverage additional input variables in TaylorGLO loss functions, such as the percentage of training steps completed. TaylorGLO may then find loss functions that are best suited for different points in training, where, for example, different kinds of regularization work best (9). Unintuitive changes to the training process, such as cycling learning rates (36), have been able to improve model performance; evolution could be a useful way to discover similar techniques. Additionally, the technique may be adapted to models with auxiliary classifiers (38) as a means to touch deeper parts of the network.

The proper choice of loss function may depend on other types of state as well. For example, batch statistics could help evolve loss functions that are more well-tuned to each batch; intermediate network activations could expose information that may help tune the function for deeper networks like ResNet; deeper information about the characteristics of a model's weights and gradients, such as that from spectral decomposition of the Hessian matrix (34), could assist the evolution of loss functions that are able to adapt to the current fitness landscape.

8 Conclusion

This paper proposes TaylorGLO as a promising new technique for loss-function metalearning. TaylorGLO leverages a novel parameterization for loss functions, allowing the use of continuous optimization rather than genetic programming for the search, thus making it more efficient and more reliable. TaylorGLO loss functions serve to regularize the learning task, significantly outperforming the standard cross-entropy loss on both MNIST and CIFAR-10 benchmark tasks with a variety of network architectures. They also outperform previously discovered loss functions while requiring many fewer candidates to be evaluated during search. Thus, TaylorGLO is a mature metalearning technique that results in higher testing accuracies, better data utilization, and more robust models.

Broader Impact

This paper presents TaylorGLO, a new technique for automatically discovering new, more optimal loss functions for neural networks. Models trained with TaylorGLO loss functions achieve higher accuracies and are more robust. These properties allow TaylorGLO to be used as a general technique that can help humans building machine learning systems to train better models with their finite efforts.

TaylorGLO uses a population-based search to optimize loss functions that can require many models to be trained. This methodology requires a large amount of compute power that scales with the time needed to train individual models. In addition to requiring greater amounts of electricity, these high compute costs can put this technique out of reach of those who do not have access to such resources. TaylorGLO attempts to ameliorate this with a parameter that can tune a trade-off between computation and loss function performance estimate accuracy.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, 2016. USENIX Association.

[2] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic programming: An introduction*, volume 1. Morgan Kaufmann San Francisco, 1998.

[3] G. Bingham, W. Macke, and R. Miikkulainen. Evolutionary optimization of deep learning activation functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020.

[4] J. Chisholm. Rational approximants defined from double power series. *Mathematics of Computation*, 27(124):841–848, 1973.

[5] H. Dong, S. Yu, C. Wu, and Y. Guo. Semantic image synthesis via adversarial learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5706–5714, 2017.

[6] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[7] J. B. Fourier. La théorie analytique de la chaleur. *Mémoires de l'Académie Royale des Sciences de l'Institut de France*, 8:581–622, 1829.

[8] R. Gao and K. Grauman. 2.5D visual sound. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 324–333, 2019.

[9] A. S. Golatkar, A. Achille, and S. Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Advances in Neural Information Processing Systems 32*, pages 10677–10687, 2019.

[10] S. Gonzalez, J. Landgraf, and R. Miikkulainen. Faster training by selecting samples using embeddings. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.

[11] S. Gonzalez and R. Miikkulainen. Improved training speed, accuracy, and data utilization through loss function optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, 2020.

[12] P. Graves-Morris. The numerical calculation of Padé approximants. In *Padé approximation and its applications*, pages 231–245. Springer, 1979.

[13] P. Graves-Morris and D. Roberts. Calculation of Canterbury approximants. *Computer Physics Communications*, 10(4):234–244, 1975.

[14] J. J. Grefenstette and J. M. Fitzpatrick. Genetic search with approximate function evaluations. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 112–120, 1985.

[15] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In *International Conference on Parallel Problem Solving from Nature*, pages 282–291. Springer, 2004.

[16] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE international conference on evolutionary computation*, pages 312–317. IEEE, 1996.

[17] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[20] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[21] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, pages 73–101, 1964.

[22] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1:61–70, 06 2011.

[23] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proceedings of the Fifth International Conference on Learning Representations (ICLR)*, 2017.

[24] D. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR)*, 12 2014.

[25] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[27] Y. LeCun, C. Cortes, and C. Burges. The MNIST dataset of handwritten digits, 1998.

[28] C. Lemke, M. Budka, and B. Gabrys. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44(1):117–130, 2015.

[29] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc., 2018.

[30] L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[31] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier, 2019.

[32] E. Real, C. Liang, D. R. So, and Q. V. Le. Automl-zero: Evolving machine learning algorithms from scratch. *arXiv:2003.03384*, 2020.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[34] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou. Empirical analysis of the Hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[35] J. Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[36] L. N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

[37] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2015.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[39] B. Taylor. *Methodus incrementorum directa & inversa. Auctore Brook Taylor, LL. D. & Regiae Societatis Secretario*. typis Pearsonianis: prostant apud Gul. Innys ad Insignia Principis in . . . , 1715.

[40] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. In *Proceedings of the USSR Academy of Sciences*, volume 4, pages 1035–1038, 1963.

[41] H. Wilbraham. On a certain periodic function. *The Cambridge and Dublin Mathematical Journal*, 3:198–201, 1848.

[42] Y. Zhou, C. Liu, and Y. Pan. Modelling sentence pairs with tree-structured attentive encoder. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING), Technical Papers*, pages 2912–2922, 2016.

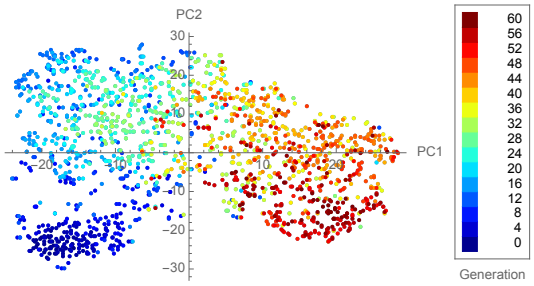


Figure 7: A visualization of all TaylorGLO loss function candidates using t-SNE (30) on MNIST. Colors map to each candidate's generation. Loss function populations show an evolutionary path and focus over time towards functions that perform well, consistent with the convergence and settling in Figure 3.

A Experimental setup

A.1 MNIST

The first domain used for evaluation was MNIST Handwritten Digits, a widely used dataset where the goal is to classify 28×28 pixel images as one of ten digits. MNIST has 55,000 training samples, 5,000 validation samples, and 10,000 testing samples. The dataset is well understood and relatively quick to train, and forms a good foundation for understanding how TaylorGLO evolves loss functions.

A.2 CIFAR-10

To validate TaylorGLO in a more challenging context, the CIFAR-10 (25) dataset was used. It consists of small color photographs of objects in ten classes. CIFAR-10 traditionally consists of 50,000 training samples, and 10,000 testing samples; however 5,000 samples from the training dataset were used for validation of candidates, resulting in 45,000 training samples.

A.3 Implementation details

Due to the number of partial training sessions that are needed to evaluate TaylorGLO loss function candidates, training was distributed across the network to a cluster, composed of dedicated machines with NVIDIA GeForce GTX 1080Ti GPUs. Training itself was implemented with TensorFlow (1) in Python. The primary components of TaylorGLO (i.e., the genetic algorithm and CMA-ES) were implemented in the Swift programming language which allows for easy parallelization. These components run centrally on one machine and asynchronously dispatch work to the cluster.

B Illustrating the evolutionary process

The TaylorGLO search process can be illustrated with t-SNE dimensionality reduction (30) on *every* candidate loss function within a run (Figure 7). The initial points (i.e. loss functions) are initially widespread on the left side, but quickly migrate and spread to the right as CMA-ES explores the parameter space, and eventually concentrate in a smaller region of dark red points. This pattern is consistent with the convergence and settling in Figure 3.

C Top MNIST loss function

The best loss function obtained from running TaylorGLO on MNIST was found in generation 74. This function, with parameters $\theta = \langle 11.9039, -4.0240, 6.9796, 8.5834, -1.6677, 11.6064, 12.6684, -3.4674 \rangle$ (rounded to four decimal-places), achieved a 2k-step validation accuracy of 0.9950 on its single evaluation, higher than 0.9903 for the cross entropy loss. This loss function was a modest improvement over the previous best loss function from generation 16, which had a validation accuracy of 0.9958.

D MNIST evaluation length sensitivity

200-step TaylorGLO is surprisingly resilient when evaluations during evolution are shortened to 200 steps (i.e., 0.4 epochs) of training. With so little training, returned accuracies are noisy and dependent on each

individual network's particular random initialization. On a 60-generation run with 200-step evaluations, the best evolved loss function had a mean testing accuracy of 0.9946 across ten samples, with a standard deviation of 0.0016. While slightly lower, and significantly more variable, than the accuracy for the best loss function that was found on the main 2,000-step run, the accuracy is still significantly higher than that of the cross-entropy baseline, with a p-value of 6.3×10^{-6}. This loss function was discovered in generation 31, requiring 1,388.8 2,000-step-equivalent partial evaluations. That is, evolution with 200-step partial evaluations is over three-times less sample sample efficient than evolution with 2,000-step partial evaluations.

20,000-step On the other extreme, where evaluations consist of the same number of steps as a full training session, one would expect better loss functions to be discovered, and more reliably, because the fitness estimates are less noisy. Surprisingly, that is not the case: The best loss function had a mean testing accuracy of 0.9945 across ten samples, with a standard deviation of 0.0015. While also slightly lower, and also significantly more variable, than the accuracy for the best loss function that was found on the main 2,000-step run, the accuracy is significantly higher than the cross-entropy baseline, with a p-value of 5.1×10^{-6}. This loss function was discovered in generation 45, requiring 12,600 2,000-step-equivalent partial evaluations; over 28-times less sample sample efficient as evolution with 2,000-step partial evaluations.

These results thus suggest that there is an optimal way to evaluate candidates during evolution, resulting in lower computational cost and better loss functions. Notably, the best evolved loss functions from all three runs (i.e., 200-, 2,000-, and 20,000-step) have similar shapes, reinforcing the idea that partial-evaluations can provide useful performance estimates.