Optimizing Loss Functions Through Multi-Variate Taylor Polynomial Parameterization

Santiago Gonzalez*
University of Texas at Austin
Austin, Texas, USA
Cognizant AI Labs
San Francisco, California, USA
slgonzalez@utexas.edu

Risto Miikkulainen
University of Texas at Austin
Austin, Texas, USA
Cognizant AI Labs
San Francisco, California, USA
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, CIFAR-10, and SVHN benchmark tasks, TaylorGLO finds new loss functions that outperform the standard cross-entropy loss as well as novel loss functions previously discovered through GP, 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.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; Distributed algorithms; *Computer vision*; **Genetic algorithms**; **Neural networks**.

KEYWORDS

Loss Functions, Metalearning, Evolutionary Strategies

ACM Reference Format:
Santiago Gonzalez and Risto Miikkulainen. 2021. Optimizing Loss Functions Through Multi-Variate Taylor Polynomial Parameterization. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3449639.3459277

*Current affiliation: Apple Inc.

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8350-9/21/07…$15.00
https://doi.org/10.1145/3449639.3459277

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 aims to solve this problem through a variety of approaches, including optimizing different aspects of the architecture from hyperparameters to topologies, and by using different methods from Bayesian optimization to evolutionary computation [7, 32, 35, 42].

Recently, loss-function discovery and optimization has emerged as a new type of metalearning. Focusing on neural network's root training goal it aims to discover 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) [13] 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) [19]. While the approach was successful, 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 mapped into 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 exclude large classes of infeasible loss functions. This paper introduces such an approach: *Multivariate Taylor 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. Special techniques can be developed 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 the MNIST, CIFAR-10, CIFAR-100, and SVHN 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 [7, 32, 35, 42]. 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, 38]. Since loss functions are at the core of machine learning, it is compelling to apply metalearning to their design as well.

Deep neural networks are trained iteratively, by updating model parameters (i.e., weights and biases) using gradients propagated backward through the network [39]. The process starts from an error given by a loss function, which represents the primary training objective of the 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 [47]) 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 [6, 9, 12, 27, 53]. Indeed, different functions have different properties; for instance the Huber Loss [24] 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; therefore, there is an opportunity to improve through metalearning.

Genetic Loss Optimization (GLO) [13] provided an initial approach 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. Thus, GLO does not scale to the large models and datasets that are typical in modern deep learning.

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 [19] as the search method, resulting in faster search.

3 LOSS FUNCTIONS AS MULTIVARIATE TAYLOR EXPANSIONS

Taylor expansions [46] 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 $C^{k_{\max}}$ 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 $0 \le k \le k_{\max}$, 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 $C^{k_{\max+1}}$ 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 $0 \le k \le k_{\max}$, 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

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

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 $|\alpha| \le k$. 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. 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 third-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:

- It guarantees smooth functions;
- Functions do not have poles (i.e., discontinuities going to infinity or negative infinity) within their relevant domain;
- They can be implemented purely as compositions of addition and multiplication operations;
- They 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 [8]. Consequently, 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 [50].

Padé approximants can be more accurate approximations than Taylor expansions; indeed, Taylor expansions are a special case of Padé approximants where $M = 0$ [14]. 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 (a bivariate extension) [4]) and Canterbury approximants (a multivariate generalization) [15].

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 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 METHOD

TaylorGLO (Figure 1) aims to find the optimal parameters for a loss function represented as a multivariate Taylor expansion. The

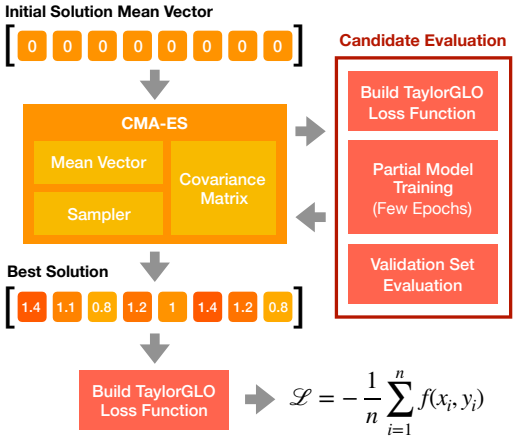


Figure 1: The TaylorGLO method. Loss functions are represented by fixed-size vectors whose elements parameterize modified Taylor polynomials. Starting with a population of initially unbiased loss functions (i.e., vectors around the origin), CMA-ES optimizes their Taylor expansion parameters in order to maximize validation accuracy after partial training. The candidate with the highest accuracy is chosen as the final, best solution.

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 $\theta^*_{\hat{f}}$ 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.

In particular, Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [19] 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 that is being sampled at each step grows, shrinks, and moves dynamically as needed to maximize sampled candidates' fitnesses. TaylorGLO uses the $(\mu/\mu, \lambda)$ variant of CMA-ES [20], which incorporates weighted rank-μ updates [18] to reduce the number of objective function evaluations needed.

In order to find $\theta^*_{\hat{f}}$, at each generation CMA-ES samples points in Θ. Their fitness is determined 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.

Fully training a model can be prohibitively expensive in many problems. However, performance near the beginning of training is usually correlated with performance at the end of training, and therefore it is enough to train the models only partially to identify the most promising candidates. This type of approximate evaluation

is common in metalearning [16, 25]. 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 step implies that any term t within $f(x_i, y_i)$ with $\frac{\partial}{\partial y_i}t = 0$ can be replaced with 0. For example, this refinement simplifies Equation 4, providing a reduction in the number of parameters from twelve to eight:

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

5 EXPERIMENTAL SETUP

This section presents the experimental setup that was used to evaluate the TaylorGLO technique.

Domains: MNIST [31] was included as simple domain to illustrate the method and to provide a backward comparison with GLO; CIFAR-10 [28], CIFAR-100 [28], and SVHN [37] were included as more modern benchmarks. Improvements were measured in comparison to 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.

Evaluated architectures: A variety of architectures were used to evaluate TaylorGLO: the basic CNN architecture evaluated in the GLO study [13], AlexNet [29], AllCNN-C [44], Preactivation ResNet-20 [22], which is an improved variant of the ubiquitous ResNet architecture [21], and Wide ResNets of different morphologies [52]. Networks with Cutout [5] and CutMix [51] were also evaluated, to show that TaylorGLO provides a different, complementary approach to regularization.

TaylorGLO setup: CMA-ES was instantiated with population size $\lambda = 28$ on MNIST and $\lambda = 20$ on all other datasets, 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 6.4—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.

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—using StudioML [36]. 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. The implementation made use of the open-source SwiftCMA [11] library for CMA-ES. These components run centrally on one machine and asynchronously dispatch work to the cluster. Training for each candidate was aborted and retried up to two additional times if validation accuracy was below 0.15 at the tenth epoch. This method helped reduce computation costs.

Further dataset and experimental setup details are provided in Appendix A.

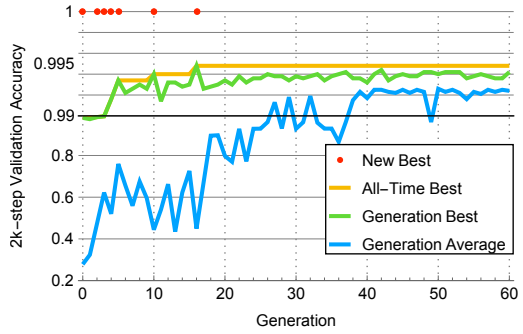


Figure 2: The process of discovering loss functions in MNIST. Red dots mark generations where new improved loss functions were found. TaylorGLO discovers good functions in very few generations. The best had a 2000-step validation accuracy of 0.9948, compared to 0.9903 with the cross-entropy loss, averaged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

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 three datasets across a variety of models are shown in Table 1.

6.1 The TaylorGLO discovery process

Figure 2 illustrates the evolution process over 60 generations, which is sufficient to reach convergence on the MNIST dataset. 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 3 shows the shapes and parameters of each generation's highest-scoring loss function. In Figure 3*a* the functions are plotted as if they were being used for binary classification, i.e. the loss for an incorrect label on the left and for a correct one on the right [13]. The functions have a distinct pattern through the evolution process. Early generations include a wider variety of shapes, but they later converge towards curves with a shallow minimum around $y_0 = 0.8$.

Table 1: Test-set accuracy of loss functions discovered by TaylorGLO compared with that of the cross-entropy loss. 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 the cross-entropy loss in almost all cases, including those that include Cutout, suggesting that it provides a different form of regularization.

Task	Model	Avg. TaylorGLO Acc.	Avg. Baseline Acc.	p-value
MNIST	Basic CNN [1]	**0.9951 (0.0005)**	0.9899 (0.0003)	2.95×10^{-15}
CIFAR-10	AlexNet [2]	**0.7901 (0.0026)**	0.7638 (0.0046)	1.76×10^{-10}
	AlexNet + Cutout [6]	**0.7786 (0.0022)**	0.7741 (0.0040)	0.0049
	AlexNet + CutMix [7]	**0.7928 (0.0027)**	0.7856 (0.0026)	8.13×10^{-6}
	PreResNet-20 [4]	**0.9169 (0.0014)**	0.9153 (0.0021)	0.0400
	AllCNN-C [3]	**0.9271 (0.0013)**	0.8965 (0.0021)	0.42×10^{-17}
	AllCNN-C [3] + Cutout [6]	**0.9329 (0.0022)**	0.8911 (0.0037)	1.60×10^{-14}
	AllCNN-C [3] + CutMix [7]	**0.9327 (0.0014)**	0.8749 (0.0042)	1.89×10^{-13}
	Wide ResNet 16-8 [5]	**0.9558 (0.0011)**	0.9528 (0.0012)	1.77×10^{-5}
	Wide ResNet 16-8 [5] + Cutout [6]	**0.9618 (0.0010)**	0.9582 (0.0011)	2.55×10^{-7}
	Wide ResNet 28-5 [5]	0.9548 (0.0015)	0.9556 (0.0011)	0.0984
	Wide ResNet 28-5 [5] + Cutout [6]	0.9621 (0.0013)	0.9616 (0.0011)	0.1882
CIFAR-100	PyramidNet 110a48 [8]	0.7409 (0.0040)	**0.7523 (0.0037)**	3.87×10^{-6}
	PyramidNet 110a48 [8] + Cutout [6]	**0.7708 (0.0029)**	0.7674 (0.0036)	0.0189
SVHN	Wide ResNet 16-8 [5]	**0.9658 (0.0007)**	0.9597 (0.0006)	1.94×10^{-13}
	Wide ResNet 16-8 [5] + Cutout [6]	**0.9714 (0.0010)**	0.9673 (0.0008)	9.10×10^{-9}
	Wide ResNet 28-5 [5]	**0.9657 (0.0009)**	0.9634 (0.0006)	6.62×10^{-6}
	Wide ResNet 28-5 [5] + Cutout [6]	**0.9727 (0.0006)**	0.9709 (0.0006)	2.96×10^{-6}

Network architecture references: [1] Gonzalez and Miikkulainen [13] [2] Krizhevsky et al. [29] [3] Springenberg et al. [44] [4] He et al. [22] [5] Zagoruyko and Komodakis [52] [6] DeVries and Taylor [5] [7] Yun et al. [51] [8] Han et al. [17]

In other words, the loss increases near the correct output—which is counterintuitive. This shape 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: it discourages the model from outputting unnecessarily extreme values for the correct class, and therefore makes overfitting less likely [13]. This is a surprising finding, and demonstrates the power of machine learning to create innovations beyond human design.

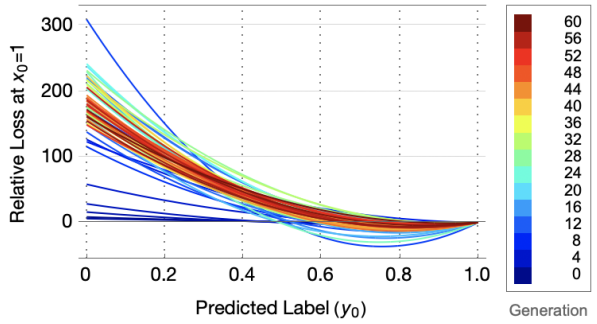
6.2 Performance comparisons

Over 10 fully-trained models, the best TaylorGLO loss function achieved a mean testing accuracy of **0.9951** (stddev 0.0005) in MNIST. 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) [13]; both differences are statistically significant (Figure 5). Notably, TaylorGLO achieved this result with significantly fewer generations. GLO required 11,120 partial evaluations (i.e., 100 individuals over 100 GP generations plus 32 individuals over 35 CMA-ES generations), while the top TaylorGLO loss function only required **448** partial evaluations, i.e. 4.03% as many. Thus, TaylorGLO achieves improved results with significantly fewer evaluations than GLO.

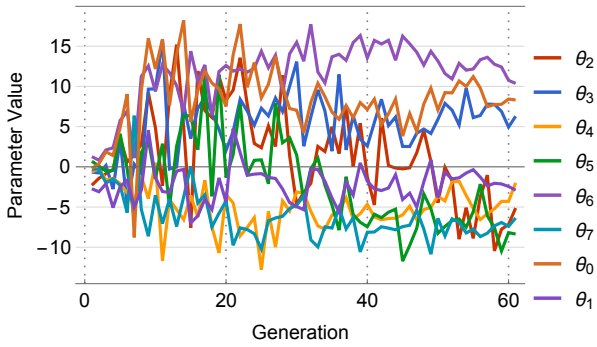
Due to the very large number evaluations required by GLO, TaylorGLO is only compared to GLO on MNIST. GLO is not practically applicable to deeper models with longer training times. For example, even a relatively small deep network, PreResNet-20 [22], would require over 171 GPU days of computation, assuming the same number of evaluations as above on MNIST.

The large reduction in evaluations during evolution compared to GLO allows TaylorGLO to tackle harder problems, including models that have millions of parameters. On CIFAR-10, CIFAR-100, and SVHN, TaylorGLO was able to outperform cross-entropy baselines consistently on a variety models, as shown in Table 1. These increases in accuracy are greater than what is possible through implicit learning rate adjustment alone (detailed in Appendix 6.5). TaylorGLO also provides further improvement on architectures that use Cutout [5], suggesting that its mechanism of avoiding overfitting is different from other regularization techniques.

In addition, TaylorGLO loss functions result in more robust trained models. In Figure 4, accuracy basins for two AllCNN-C models, one trained with the TaylorGLO loss function and another with the cross-entropy loss, are plotted along a two-dimensional slice $[-1, 1]$ of the weight space (a technique due to [33]). 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 [26].



(a) Best discovered functions over time



(b) 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 significant increase in the end. This shape is likely to prevent overfitting during learning, which leads to the observed improved accuracy.

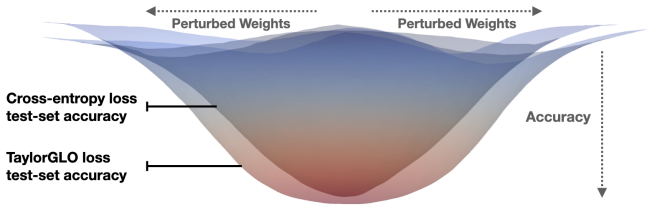
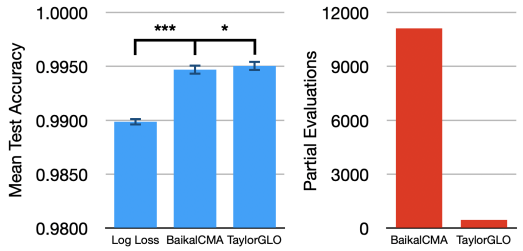


Figure 4: Accuracy basins for AllCNN-C models trained with both cross-entropy and TaylorGLO loss functions. The TaylorGLO basins are both flatter and lower, indicating that they are more robust and generalize better [26], which results in higher accuracy.



(a) Accuracy (b) Evaluations

Figure 5: (a) Mean test accuracy across ten runs on MNIST. The TaylorGLO loss function with the highest validation score significantly outperforms the cross-entropy loss ($p = 2.95\times10^{-15}$ in a one-tailed Welch's t-test) and the BaikalCMA loss [13] ($p = 0.0313$). (b) Required partial training evaluations for GLO and TaylorGLO on MNIST. The TaylorGLO loss function was discovered with 4% of the evaluations that GLO required to discover BaikalCMA.

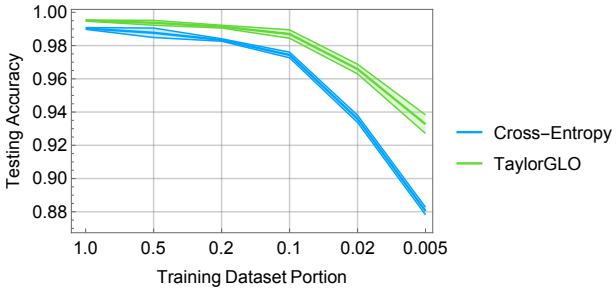


Figure 6: Accuracy with reduced portions of the MNIST dataset. Progressively smaller portions of the dataset were used to train the models (averaging over ten runs). The TaylorGLO loss function provides significantly better performance than the cross-entropy loss on all training dataset sizes, and particularly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

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 (similar results were obtained with other datasets and architectures). Overall, TaylorGLO significantly outperforms the cross-entropy loss. 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.

6.4 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 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. That is, evolution with 20,000-step full evaluations is over 28-times less sample efficient than 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.

6.5 Learning rate sensitivity

Loss functions can embody different learning rates implicitly. This section shows that TaylorGLO loss functions' benefits come from more than just metalearning such learning rates. Increases in performance that result from altering the base learning rate with cross-entropy loss are significantly smaller than those that TaylorGLO provides.

More specifically, Figure 7 quantifies the effect of varying learning rates on the final testing accuracy of AllCNN-C models trained on CIFAR-10. AllCNN-C was chosen for this analysis since it exhibits the largest variations in performance, making this effect more clear. While learning rates larger than 0.01 (the standard learning rate for AllCNN-C) reach slightly higher accuracies, this effect comes at the cost of less stable training. The majority of models trained with these higher learning rates failed to train. Thus, the standard choice of learning rate for AllCNN-C is appropriate for the cross-entropy loss, and TaylorGLO loss functions are able to improve upon it.

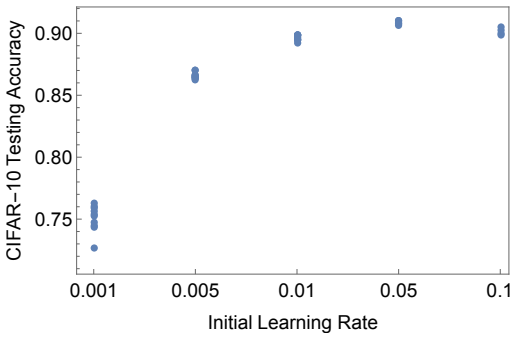


Figure 7: Effect of varying learning rates in AllCNN-C when trained with the cross-entropy loss on CIFAR-10. For each learning rate, ten models were trained, with up to ten retries if training failed. The majority of training attempts failed for learning rates larger than 0.01. The 0.01 learning rate used in the experiments in this paper results in best stable performance. Overall, the small performance differences that can result from adjusting the learning rate, regardless of stability, are much smaller than those that result from training with TaylorGLO. Thus, TaylorGLO provides a mechanism for improvement beyond implicit adjustments of the learning rate.

Table 2: Performance of Taylor approximations of the cross-entropy loss function on AllCNN-C with CIFAR-10. Approximations of different orders, with $\mathbf{a} = \langle 0.5, 0.5 \rangle$, are presented. Presented accuracies are the mean from ten runs. Higher-order approximations are better, suggesting a potential (although computationally expensive) opportunity for improvement in the future.

Loss Function	Mean Accuracy (stddev)
$k = 2$	0.1034 (0.0101)
$k = 3$	0.8451 (0.0043)
$k = 4$	0.8592 (0.0032)
$k = 5$	0.8649 (0.0042)
Cross-Entropy	**0.8965 (0.0021)**

7 TAYLOR APPROXIMATIONS OF THE CROSS-ENTROPY LOSS

While TaylorGLO's performance originates primarily from discovering better loss functions, it is informative to analyze what role the accuracy of the Taylor approximation plays in it. One way to characterize this effect is to analyze the performance of various Taylor approximations of the cross-entropy loss.

Table 2 provides results from such a study. Bivariate approximations to the cross-entropy loss, centered at $\mathbf{a} = \langle 0.5, 0.5 \rangle$, with different orders k were used to train AllCNN-C models on CIFAR-10. Third-order approximations and above are trainable. Approximations' performance is within a few percentage points of the cross-entropy loss, with higher-order approximations yielding progressively better accuracies, as expected.

Table 3: Estimated TaylorGLO experiment durations and total emissions. The estimates assume populations of 20 concurrent candidates and 50 generation runs. Emission values are upper bounds reported in equivalent kilograms of carbon dioxide, thus accounting for other gases of interest. Overall, experiments are short enough that they can each be run over a few days.

TaylorGLO Experiment	Duration (hrs)	Total Emissions ($kgCO_2eq$)
AlexNet on CIFAR-10	3.60	4.07
ResNet-20 on CIFAR-10	10.24	11.58
Pre ResNet-20 on CIFAR-10	9.26	10.48
AllCNN-C on CIFAR-10	17.06	19.30
PyramidNet 110a48 on CIFAR-10	73.86	83.53
Wide ResNet 28-5 on CIFAR-10	42.90	48.52
Wide ResNet 16-8 on CIFAR-10	36.08	40.80
Wide ResNet 28-10 on CIFAR-10	105.30	119.10

The results thus show that third-order TaylorGLO loss functions cannot represent the cross-entropy baseline loss accurately. One possibility for improving TaylorGLO is thus to utilize higher order approximations. However, it is remarkable that TaylorGLO can still find loss functions that outperform the cross-entropy loss. Also, the increase in the number of parameters—and the corresponding increase in computational requirements—may in practice outweigh the benefits from a finer-grained representation. This effect was seen in preliminary experiments, and the third-order approximations (used in this paper) deemed to strike a good balance.

8 EXPERIMENT DURATIONS AND ENVIRONMENTAL IMPACT

Understanding the computational costs and broader impacts of modern deep learning systems is crucial as they become more complex and computationally intensive over time.

The infrastructure that ran the experiments in this paper is located in California, which is estimated to have had an estimated carbon dioxide equivalent total output emission rate of 226.21 $kgCO_2eq/kWh$ in 2018 [48]. This quantity can be used to calculate the climate impact of compute-intensive experiments.

Table 3 provides estimates of durations and total emissions for various TaylorGLO experiments. Emissions were calculated using the Machine Learning Impact calculator [30], assuming that no candidates failed evaluation (which would result in slightly lower estimates). Presented values can thus be thought of as being an upper bound.

Overall, experiment durations are short enough that TaylorGLO can be practically applied to different tasks to find customized loss functions.

9 DISCUSSION AND FUTURE WORK

TaylorGLO was applied to the benchmark tasks 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.

As illustrated in Figure 3*a*, the most significant effect of evolved loss functions is to discourage extreme output values, thereby avoiding overfitting. It is interesting that this mechanism is apparently different from other regularization techniques such as dropout (as shown by [13]) and data augmentation with Cutout (as seen in Table 1). Dropout and Cutout improve performance over the baseline, and loss function optimization improves it further. This result suggests that regularization is a multifaceted process, and further work is necessary to understand how to best take advantage of it.

Another important direction is to incorporate state information into 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 [10]. Unintuitive changes to the training process, such as cycling learning rates [43], have been found to improve performance; evolution could be used to find other such opportunities automatically. 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 [41], could assist the evolution of loss functions that adapt to the current fitness landscape. The technique could also be adapted to models with auxiliary classifiers [45] as a means to touch deeper parts of the network.

10 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, outperforming the standard cross-entropy loss significantly on MNIST, CIFAR-10, CIFAR-100, and SVHN benchmark tasks with a variety of network architectures. They also outperform previously loss functions discovered in prior work, while requiring many fewer candidates to be evaluated during search. Thus, TaylorGLO results in higher testing accuracies, better data utilization, and more robust models, and is a promising new avenue for metalearning.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, Savannah, GA, 265–283. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[2] Wolfgang Banzhaf, Peter Nordin, Robert E Keller, and Frank D Francone. 1998. *Genetic programming: An introduction*. Vol. 1. Morgan Kaufmann San Francisco.

[3] Garrett Bingham, William Macke, and Risto Miikkulainen. 2020. Evolutionary Optimization of Deep Learning Activation Functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

[4] JSR Chisholm. 1973. Rational approximants defined from double power series. *Math. Comp.* 27, 124 (1973), 841–848.

[5] Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (2017).

[6] Hao Dong, Simiao Yu, Chao Wu, and Yike Guo. 2017. Semantic image synthesis via adversarial learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 5706–5714.

[7] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research* 20, 55 (2019), 1–21.

[8] Joseph BJ Fourier. 1829. La théorie analytique de la chaleur. *Mémoires de l'Académie Royale des Sciences de l'Institut de France* 8 (1829), 581–622.

[9] Ruohan Gao and Kristen Grauman. 2019. 2.5D visual sound. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 324–333.

[10] Aditya Sharad Golatkar, Alessandro Achille, and Stefano Soatto. 2019. 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*. 10677–10687.

[11] Santiago Gonzalez. 2019. SwiftCMA. https://github.com/sgonzalez/SwiftCMA.

[12] Santiago Gonzalez, Joshua Landgraf, and Risto Miikkulainen. 2019. Faster Training by Selecting Samples Using Embeddings. In *2019 International Joint Conference on Neural Networks (IJCNN)*.

[13] Santiago Gonzalez and Risto Miikkulainen. 2020. Improved Training Speed, Accuracy, and Data Utilization Through Loss Function Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*.

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

[15] PR Graves-Morris and DE Roberts. 1975. Calculation of Canterbury approximants. *Computer Physics Communications* 10, 4 (1975), 234–244.

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

[17] Dongyoon Han, Jiwhan Kim, and Junmo Kim. 2017. Deep pyramidal residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5927–5935.

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

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

[20] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*. Springer, 630–645.

[23] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[24] Peter J Huber. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics* (1964), 73–101.

[25] Yaochu Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* 1 (06 2011), 61–70. https://doi.org/10.1016/j.swevo.2011.05.001

[26] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proceedings of the Fifth International Conference on Learning Representations (ICLR)*.

[27] Diederik Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR)*.

[28] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[30] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the Carbon Emissions of Machine Learning. *arXiv preprint arXiv:1910.09700* (2019).

[31] Yann LeCun, Corinna Cortes, and CJC Burges. 1998. The MNIST dataset of handwritten digits.

[32] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. 2015. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review* 44, 1 (2015), 117–130.

[33] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the Loss Landscape of Neural Nets. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 6389–6399. http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

[34] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[35] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. 2019. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 293–312.

[36] Karl Mutch. 2017 - 2021. *Studio Go Runner*. https://github.com/leaf-ai/studio-go-runner/tree/0.13.1

[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature learning. *Neural Information Processing Systems, Workshop on Deep Learning and Unsupervised Feature Learning* (2011).

[38] Esteban Real, Chen Liang, David R. So, and Quoc V. Le. 2020. AutoML-Zero: Evolving Machine Learning Algorithms From Scratch. *arXiv:2003.03384* (2020).

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation*. Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.

[40] Graeme D Ruxton. 2006. The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test. *Behavioral Ecology* 17, 4 (2006), 688–690.

[41] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. 2017. Empirical analysis of the Hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454* (2017).

[42] Jürgen Schmidhuber. 1987. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. Ph.D. Dissertation. Technische Universität München.

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

[44] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2015. Striving for Simplicity: The All Convolutional Net. *CoRR* abs/1412.6806 (2015).

[45] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

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

[47] Andrey N. Tikhonov. 1963. Solution of incorrectly formulated problems and the regularization method. In *Proceedings of the USSR Academy of Sciences*, Vol. 4. 1035–1038.

[48] United States Environmental Protection Agency. 2020. EPA eGRID2018. https://www.epa.gov/egrid.

[49] Bernard L Welch. 1947. The generalization of Student's problem when several different population variances are involved. *Biometrika* 34, 1/2 (1947), 28–35.

[50] Henry Wilbraham. 1848. On a certain periodic function. *The Cambridge and Dublin Mathematical Journal* 3 (1848), 198–201.

[51] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 6023–6032.

[52] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* (2016).

[53] Yao Zhou, Cong Liu, and Yan Pan. 2016. Modelling Sentence Pairs with Tree-structured Attentive Encoder. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING), Technical Papers*. 2912–2922.