EvoGrad: Evolutionary-Weighted Gradient and Hessian Learning for Black-Box Optimization

Yedidya Kfir [1], **Elad Sarafian**[2], **Yoram Louzoun**[1], **Sarit Kraus**[1]

[1]Bar-Ilan University,
[2]Nvidia
kfiry@biu.ac.il, esarafian@nvidia.com, louzouy@math.biu.ac.il, sarit@cs.biu.ac.il

Abstract

Black-box algorithms aim to optimize functions without access to their analytical structure or gradient information, making them essential when gradients are unavailable or computationally expensive to obtain. Traditional methods for black-box optimization (BBO) primarily utilize non-parametric models, but these approaches often struggle to scale effectively in large input spaces. Conversely, parametric approaches, which rely on neural estimators and gradient signals via backpropagation, frequently encounter substantial gradient estimation errors, limiting their reliability. Explicit Gradient Learning (EGL), a recent advancement, directly learns gradients using a first-order Taylor approximation and has demonstrated superior performance compared to both parametric and non-parametric methods. However, EGL inherently remains local and myopic, often faltering on highly non-convex optimization landscapes. In this work, we address this limitation by integrating global statistical insights from the evolutionary algorithm CMA-ES into the gradient learning framework, effectively biasing gradient estimates towards regions with higher optimization potential. Moreover, we enhance the gradient learning process by estimating the Hessian matrix, allowing us to correct the second-order residual of the Taylor series approximation. Our proposed algorithm, EvoGrad2 (Evolutionary Gradient Learning with second-order approximation), achieves state-of-the-art results on the synthetic COCO test suite, exhibiting significant advantages in high-dimensional optimization problems. We further demonstrate EvoGrad2's effectiveness on challenging real-world machine learning tasks, including adversarial training and code generation, highlighting its ability to produce more robust, high-quality solutions. Our results underscore EvoGrad2's potential as a powerful tool for researchers and practitioners facing complex, high-dimensional, and non-linear optimization problems.

Code — https://github.com/yedidyakfir/egl
Extended version — https://arxiv.org/abs/2502.04829

1 Introduction

Black-box optimization (BBO) is the process of searching for optimal solutions within a system's input domain without access to its internal structure or analytical properties (Audet et al. 2017a). Unlike gradient-based optimization methods that rely on the calculation of analytical gradients, BBO algorithms query the system solely through input-output pairs, operating agnostically to the underlying function. This feature distinguishes BBO from traditional ML tasks, such as neural network training, where optimization typically involves backpropagation-based gradient computation.

Many real-world systems naturally fit the BBO framework because their behavior is difficult or impossible to model explicitly. In such cases, BBO algorithms have achieved remarkable success across fields, such as robotic motor control (Gehring et al. 2014; Prabhu et al. 2018), ambulance deployment (Zhen et al. 2014), parameter tuning (Olof 2018; Rimon et al. 2024), and signal processing (Liu et al. 2020), among others (Alarie et al. 2021). BBO applications also go beyond physical systems; many ML problems are black-box when the true gradient is unavailable. Examples include hyperparameter tuning (Bischl et al. 2023), contextual bandit problems (Bouneffouf, Rish, and Aggarwal 2020), large language model training with human feedback (Bai et al. 2022), and optimizing language models for long context inference (Ding et al. 2024), to name a few.

As problem dimensions increase, the cost of evaluating intermediate solutions becomes a critical constraint, especially in real-world settings where interaction with the environment is expensive or in ML, where larger models demand substantial computational power. Therefore, modern BBO algorithms must reduce evaluation steps (Hansen et al. 2010). Achieving this requires algorithms capable of more accurately predicting optimization directions, either through better gradient approximation (Anil et al. 2020; Lesage-Landry, Taylor, and Shames 2020) or momentum-based strategies to handle non-convexity and noise. We propose two novel methods: (1) Evolutionary Gradient Learning (EvoGrad), a weighted gradient estimator that biases toward promising solutions, and (2) Higher-Order Gradient Learning (HGrad), which incorporates Hessian corrections to yield more accurate gradient approximations.

We unify the strengths of EvoGrad and HGrad in EvoGrad^2, which offers four key advantages:

- **Performance**: EvoGrad^2 consistently outperforms baseline algorithms across a diverse range of problems. Including synthetic test suites and real-world ML applications. It can handle noisy and non-convex environments.

- **Strategic Global Gradient Estimation**: Although traditional local search methods can become trapped due to exploitation steps, our algorithm strategically incorporates statistical biases toward promising global regions, enhancing the likelihood of identifying directions leading closer to the global minimum.
- **Success Rate**: Our algorithms are consistent throughout the suite, able to solve more problems than other algorithms on the benchmark.

Related works: Black-box optimization (BBO) algorithms have a long history, with various approaches developed over the years. Some of the foundational techniques include grid search, coordinate search (Audet et al. 2017b), simulated annealing (Busetti 2003), and direct search methods like Generalized Pattern Search and Mesh Adaptive direct search (Audet et al. 2017c), Gradient-less descent (Golovin et al. 2019), and ZOO (Chen et al. 2017). These approaches iteratively evaluate potential solutions and decide whether to continue in the same direction. However, they resample at every step and don't reuse the budget from previous iterations, wasting a lot of budget.

Another prominent family of BBO algorithms is the evolutionary methods (Back 1996). Including methods such as Covariance Matrix Adaptation (CMA) (Hansen 2016) and Particle Swarm Optimization (PSO) (Clerc 2010). These simulate the process of natural evolution, where a population of solutions evolves through mutation and selection (Audet et al. 2017d). They perform well in BBO environments due to their effectiveness in tackling non-convex problems. However, they come with significant drawbacks, particularly the need for extensive fine-tuning of parameters like generation size and mutation rates. CMA, for example, struggles in higher-dimensional environments and requires careful adjustment of hyperparameters and guidance to perform optimally (Loshchilov, Schoenauer, and Sebag 2013; Tang 2021). Recently, (Braun, Lange, and Toussaint 2024) uses CMA as a building block for SVGD (Liu and Wang 2016) in intractable environments, highlighting the diverse ways in which CMA and evolution strategies can be repurposed.

Then there are model-based methods (Audet et al. 2017e), which attempt to emulate the behavior of the function using a surrogate model. These models provide important analytical information, such as gradients (Bertsekas 2015), to guide the optimization process and help find a minimum. Within this class, we can further distinguish two sub-classes. To address the issue of dimensionality, Explicit Gradient Learning (EGL) was proposed by (Sarafian et al. 2020). While many model-based methods focus on learning the function's structure to derive analytical insights (e.g., Indirect Gradient Learning or IGL (Lillicrap 2015; Sarafian et al. 2020)), EGL directly learns the gradient information. EGL uses Taylor's theorem to estimate the gradient. The authors also emphasize the importance of utilizing a trust region to handle black-box optimization problems.

Several works have explored hybrid approaches that combine gradient-based methods with evolutionary algorithms, aiming to leverage both global search and local descent capabilities. For example, (Liu, Li, and Qian 2020; Maheswaranathan et al. 2019; Kunpeng, Yugang, and Xiabi 2012) utilize gradient information to construct probabilistic models or distributions, which are then employed to estimate the search direction. On the other hand, (Tang 2021) took a different approach. Ignoring the functions' gradient information, it seeks to train a generative model from the function's distribution and generate better candidate solutions from the model.

The paper is organized as follows: Section 2 covers the algorithm's theoretical background and mathematical foundations. Sections 3 and 4 present our two enhanced variants of the gradient learning algorithm: EvoGrad and HGrad. These are followed by section 5 where we present the full algorithm, EvoGrad^2. Section 6 provides experimental results on the synthetic COCO test suite, and Section 7 highlights 2 real-world high-dimensional applications and potential uses. Finally, section 8 concludes and suggests future research directions. The supplementary material shows our code, experiments, and environment setup.

2 Background

BBO: The goal of black-box optimization (BBO) is to minimize a target function $f(x)$ through a series of evaluations (Audet et al. 2017a), over a predefined domain Ω:

$$\text{find: } x^* = \arg\min_{x\in\Omega} f(x) \tag{1}$$

Explicit Gradient Learning: The Explicit Gradient Learning method, proposed by (Sarafian et al. 2020), leverages the first-order Taylor's expansion: $f(y) = f(x) + \nabla f(x)^\top (y - x) + R_1(x, y)$. Here, $R_1(x, y) = O(\|y - x\|^2)$ is a higher-order residual. By minimizing the residual term with a surrogate neural network model, EGL learns the *mean-gradient*: a smooth approximation of the function's gradient

$$\begin{aligned} g_\varepsilon^{EGL}(x) &= \arg\min_{g_\theta:\mathbb{R}^n\to\mathbb{R}^n} \int_{\tau\in B_\varepsilon(0)} (\mathcal{R}_{g_\theta,x}^{EGL}(\tau))^2 d\tau \\ \mathcal{R}_{g_\theta,x}^{EGL}(\tau) &= f(x) - f(x+\tau) + g_\theta(x)^\top \tau \end{aligned} \tag{2}$$

Here, $B_\varepsilon(0)$ is a ball around 0 defining the acquisition region of new statistics, where usually samples are uniformly sampled from this region. As $\varepsilon \to 0$, the mean-gradient converges to ∇f. EGL thus uses ε to control the accuracy of the mean-gradient. This property enables EGL to explore the entire landscape. Specifically, when ε is sufficiently large, EGL can locate lower regions in the function. Conversely, when $\varepsilon \to 0$, it converges to a local minimum. Leveraging this property, the paper shows it is possible to determine both the algorithm's convergence rate and the estimated gradient's error rate. See App. of (Kfir et al. 2025).

CMA-ES: Covariance Matrix Adaptation Evolution Strategy (CMA-ES) builds and updates a multivariate Gaussian distribution to guide optimization. The key element is the covariance matrix, which captures the shape and orientation of the search distribution based on selected samples.

For each generation g, the algorithm samples $x_1, \ldots, x_\lambda \sim \mathcal{N}(m, \sigma^2 C)$, where m, σ, and C are the evolving mean, standard deviation, and covariance matrix parameters. The sampled points are then sorted

according to their *fitness* values, i.e. $[x_1, \ldots, x_\lambda]$ such that $(f(x_i) \leq f(x_j) : \forall i < j)$, and a set of recombination weights $w_1, \ldots, w_\mu$ is assigned. These weights are positive, decreasing with rank ($w_1 \geq w_2 \geq \cdots \geq w_\mu > 0$), and normalized such that $\sum_{i=1}^{\mu} w_i = 1$. The selected samples are then used to update the parameters m, σ, and C for the next generation $g + 1$. The covariance matrix is updated by first computing an estimate of the generation covariance:

$$C' = \sum_{i=1}^{\mu} w_i \cdot \left(\frac{x_i - m}{\sigma}\right)\left(\frac{x_i - m}{\sigma}\right)^\top$$

Then, applying a weighted update to obtain the new covariance matrix

$$C^g = (1 - c_\mu)C^{g-1} + c_\mu C'$$

. Here C^{g-1} and C^g are the previous and updated matrices, and c_μ is a hyperparameter. Likewise, m and σ are updated to refine the population statistics and guide convergence (usually defined as m or x_1 of the last generation).

Trust Region: A powerful tool for BBO algorithms is a trust region (TR), which restricts the search to a localized area around the current estimate. By constraining the optimization steps and standardizing the input-output statistics. (Sarafian et al. 2020) showed the usefulness of gradient learning with neural-networks. In our work, we applied TR also to CMA-ES to create strong baseline algorithms, see Algorithm in App. (Kfir et al. 2025). This modification, though conceptually simple, leads to significant performance gains, especially in high-dimensional settings.

3 Gradient Learning with Evolutionary Weights

Local search algorithms are based on the notion that the gradient descent path is the optimal search path. However, this assumption often proves suboptimal as it can lead to inefficient sampling and susceptibility to local minima. To address this, we design a gradient learning algorithm that incorporates global statistics measured by statistical algorithms like CMA-ES. To that end, we introduce an importance weighting function W that assigns a higher weight to directions leading to lower values of the objective function. This encourages the algorithm to prioritize descent directions that not only follow the gradient but also align with globally favorable outcomes.

The Evolutionary Gradient Learning objective is defined via importance-weighted integration of Eq. 2

$$g_\varepsilon^{EvoGrad}(x) = \underset{g_\theta:\mathbb{R}^n\to\mathbb{R}^n}{\arg\min} \int\limits_{\tau\in B_\varepsilon(0)} W(x+\tau)\cdot \mathcal{R}_{g_\theta,x}^{EGL}(\tau)^2 d\tau \tag{3}$$

The importance sampling factor W should be chosen s.t. it biases the optimization path towards lower regions regardless of the local curvature around x, i.e. $W(x_1) \geq W(x_2)$ for $f(x_1) \leq f(x_2)$.

In our implementation, we train a CMA-ES with the samples generated by our uniform acquisition function around x_k (where k is the iteration index), and we use the evolved CMA distribution parameters as the integral weights, i.e.

$$W(x) \sim \exp\left(-\frac{1}{2\sigma^2}(x-m)^T C^{-1}(x-m)\right) \tag{4}$$

Where m, σ and C are the CMA-ES evolving statistics as defined in Sec. 2. This choice shifts the focus from the uniform acquisition function around x_k to a higher quality area and biases the gradient towards the current global minimum inside the trust region.

Notice that in practice, the theoretical objective in Eq. 3 is replaced by a sampled Monte-Carlo version (see Sec. 5) s.t. the sum of all weights across the sampled batch is smaller than 1. Additionally, to prevent the loss of gradient information during sampling, the function ensures that no sample is assigned a weight of exactly zero. Therefore, a minimum weight is enforced for each sample. This lower bound guarantees that all sampled directions contribute to the gradient estimate, and it also enables the proof for the *controllable accuracy* (Appendix in (Kfir et al. 2025)) property of EvoGrad, which implies that the mean-gradient converges to the true gradient, still holds for our biased version, s.t. when $\varepsilon \to 0$, $g_\varepsilon^{EvoGrad} \to \nabla f(x)$, this guarantees that the convergence properties of the mean-gradient still hold.

Theorem 3.1. *(Evolutionary Gradient Controllable Accuracy) For any differentiable function f with a continuous gradient, there exists $\kappa^{EvoGrad} > 0$ such that for any $\varepsilon > 0$, $g_\varepsilon^{EvoGrad}(x)$ satisfies*

$$\|g_\varepsilon^{EvoGrad}(x) - \nabla f(x)\| \leq \kappa^{EvoGrad}\varepsilon \quad \textit{for all } x \in \Omega.$$

Corollary 3.2 (Approximate Convergence of EvoGrad)**.** *Let f and $g_\varepsilon^{EvoGrad}$ satisfy the assumptions of Theorem 3.1, and consider the iteration*

$$x_{k+1} = x_k - \alpha\, g_\varepsilon^{EvoGrad}(x_k).$$

For a suitable fixed step size $\alpha > 0$, this update generates a descent sequence and converges to a point $x^\star$ with $\|\nabla f(x^\star)\| = O(\varepsilon)$; in particular, as $\varepsilon \to 0$ the procedure approaches a (local) minimum. A complete statement with explicit conditions and constants is given in the Appendix.

To demonstrate the practical benefits of our evolutionary weighting approach, we plot 4 typical trajectories of EGL and EvoGrad in an environment with multiple local minima in Fig. 1. The trajectory of the weighted gradient is more direct, focusing on the global minimum, and is less easily distracted by nearby local minima. We also find that the biased version has a significantly higher probability of detecting the global minimum across different epsilon sizes, regardless of the starting point. In Fig. 2, we show 2 1D functions and the probability for each algorithm to find the direction to the global minimum (relative to epsilon). For each value of ε, we sampled 500 random points, trained the gradient network $g_\varepsilon^{EvoGrad}(x)$, and evaluated the resulting gradient direction. We can see that EvoGrad rapidly outperforms EGL, which is far more likely to get stuck in the local minimum, achieving substantially higher convergence rates to the global minimum, particularly as ε increases.

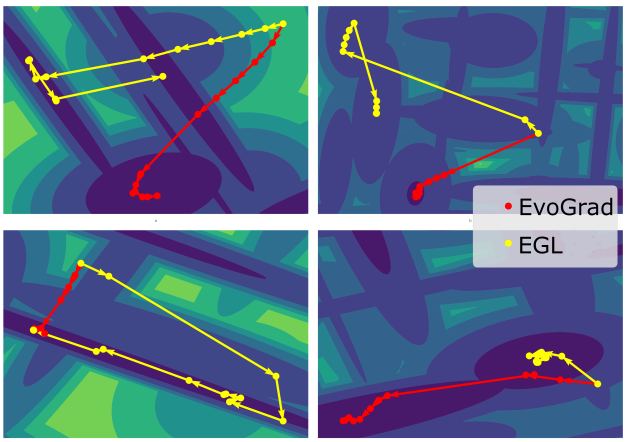


Figure 1: EvoGrad vs EGL trajectories. 1st row: Gallagher's Gaussian 101-me. 2nd row: 21-hi.

4 Gradient Learning with Hessian Corrections

To learn the mean-gradient, EGL minimizes the first-order Taylor residual (Sec. 2). Higher-order approximations yield more accurate models, the second-order Taylor expansion is:

$$f(x+\tau) = f(x)+\nabla f(x)^\top \tau+\frac{1}{2}\tau^\top \nabla^2 f(x)\tau+R_2(x,x+\tau)$$

Here $R_2(x,y) = O(\|x-y\|^3)$ is the second order residual. By replacing ∇f with a surrogate model g_θ and minimizing the resulting surrogate residual, we obtain our Higher-order Gradient Learning (HGrad) variant

$$g_\varepsilon^{HGrad}(x) = \underset{g_\theta:\mathbb{R}^n\to\mathbb{R}^n}{\arg\min} \int_{\tau\in B_\varepsilon(0)} \mathcal{R}_{g_\theta,x}^{HGrad}(\tau)^2 d\tau \qquad (5)$$

$$\mathcal{R}_{g_\theta,x}^{HGrad}(\tau) = f(x) - f(x+\tau) + g_\theta(x)^\top \tau + \frac{1}{2}\tau^\top J_{g_\theta}(x)\tau$$

The new higher-order term $J_{g_\theta}(x)$ is the Jacobian of $g_\theta(x)$, evaluated at x which approximates the function's Hessian matrix in the vicinity of our current solution, i.e., $J_{g_\theta}(x) \approx \nabla^2 f(x)$. Next, we show theoretically that, as expected, HGrad converges faster to the true gradient, which amounts to lower gradient error in practice.[1]

Theorem 4.1. *(Improved Controllable Accuracy): For any twice differentiable function $f \in \mathcal{C}^2$, there exists $\kappa_{HGrad} > 0$ such that for any $\varepsilon > 0$, the second-order mean-gradient $g_\varepsilon^{HGrad}(x)$ satisfies*

$$\|g_\varepsilon^{HGrad}(x) - \nabla f(x)\| \le \kappa_{HGrad}\varepsilon^2 \quad \textit{for all } x \in \Omega.$$

In other words, in HGrad, the model's error is of an order of magnitude ε^2 instead of ε in EGL and EvoGrad.

[1]Notice that, while HGrad incorporates Hessian corrections during the gradient learning phase, unlike Newton's methods, it is not used for scaling the gradient step size. Scaling the gradient requires calculation of the inverse Hessian, which is prone to numerical challenges and instabilities, and in our experiments was found less effective than the HGrad approach of minimizing the Taylor residual term.

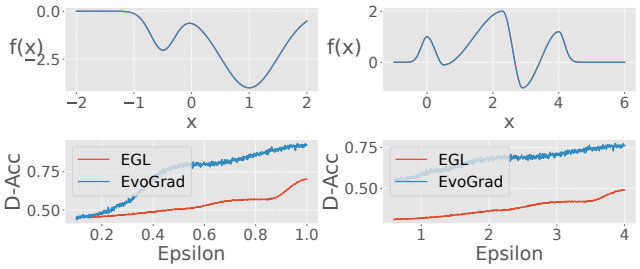


Figure 2: D(irectional) Acc(uracy) - Probability for each algorithm to find the correct direction to the global minimum at a randomly selected starting point, based on the epsilon.

Learning the gradient with the Jacobian corrections introduces a computational challenge, as double backpropagation can be expensive. This overhead can hinder the scalability and practical application of the method. A swift remedy is to detach the Jacobian matrix from the computation graph. While this step slightly changes the objective's gradient (i.e., the gradient through ($\mathcal{R}^{HGrad}$), it removes the second-order derivative, and in practice, we found that it achieves similar results compared to the full backpropagation through the residual $\mathcal{R}^{HGrad}$, see Tab. 1.

5 EvoGrad^2 - Combining Evolutionary and Higher-Order Gradient Learning

Finally, we combine the Evolutionary Weighting method (Sec. 3) with the Hessian corrections (Sec. 4), presenting a practical implementation where integrals are replaced by Monte-Carlo sums over sampled pairs. The resulting loss function defines the Evolutionary Higher-order Gradient (EvoGrad^2) model

$$\mathcal{L}_\varepsilon^{EvoGrad^2}(\theta) = \sum_{\|x_i-x_j\|\le\varepsilon} W(x_j)\mathcal{R}_{g_\theta,x_i}^{HGrad}(x_i-x_j)^2 \quad (6)$$

The summation is over sampled pairs which satisfy $\|x_i - x_j\| \le \varepsilon$. As explained in Sec. 4, we detach the Jacobian of g_θ from the computational graph to avoid second-order derivatives. Our final algorithm, including the technical details on CMA integration with gradient learning, is outlined in the appendix (See Alg. in (Kfir et al. 2025) appendix).

6 Experiments in the COCO test suite

We evaluated the EvoGrad, HGrad, and EvoGrad^2 algorithms on the COCO framework (Hansen et al. 2021) and compared them to EGL and other strong baselines: (1) CMA and its trust region variant CMA-TR, (2) Implicit Gradient Learning (IGL), where we train a model for the objective function and obtain the gradient estimation by backpropagation as in DDPG (Lillicrap 2015), and (3) a variety of known algorithms (BFGS (Nocedal and Wright 2018), Nelder-Mead (Nelder and Mead 1965), Powell (Powell 1964), SLSQP (Kraft 1988)), Using Scipy[2] python package's implementation. We also adjusted EGL hyperparameters (See App. in (Kfir et al. 2025)) and improved the trust

[2]https://scipy.org/

Metric	**EvoGrad^2**	**EvoGrad**	EvoGrad-0.1	HGrad	HGrad-Attached	EGL	IGL
Budget to solve	**44,712**	51,859	54,972	61,147	61,221	58,488	75,031
Mean	**0.003(0.02)**	0.007(0.04)	0.072(0.1)	0.014(0.07)	0.013(0.07)	0.020(0.07)	0.044(0.10)
Solved Functions	**0.949**	0.917	0.71	0.858	0.858	0.821	0.674
Metric	CMA-TR	CMA	BFGS	SLSQP	Nelder-Mead	Powell	
Budget to solve	53,569	71,855	89,667	117,172	102,783	69,205	
Mean	0.047(0.16)	0.090(0.22)	0.159(0.31)	0.363(0.42)	0.216(0.36)	0.070(0.19)	
Solved Functions	0.740	0.612	0.502	0.296	0.435	0.631	

Table 1: Comparison of different metrics: Budget used to find value of 0.01 or lower (↓), the mean normalized results (↓), std (↓); and percentage of solved problems (↑).

region (See App. in (Kfir et al. 2025)) to reduce the budget usage.

We use the following evaluation metrics:

- **Performance**: The lowest objective value found within the given budget.
- **Success Rate**: Percentage of problems solved within a fixed budget.
- **Robustness**: Performance stability across different hyperparameter settings.

Performance was normalized against the best-known solutions to minimize bias: $\texttt{normalized\_value} = \frac{y - y_{min}}{y_{max} - y_{min}}$. A function was considered solved if the normalized value was below 0.01.

Success Rate and Performance

For the first experiment, observe Figure 3, where we compare the success rates of the benchmark algorithms by dimension. The dimensional analysis shows that as dimension grows, traditional optimization methods suffer. While the trust region improved the results of the CMA variants, it still shows a performance drop in dimensions above 40. This performance degradation reflects a fundamental challenge for sample-based BBO methods: the curse of dimensionality demands exponentially more evaluations as problem size grows. Traditional approaches that work effectively in lower-dimensional spaces become increasingly sample-inefficient in higher dimensions. On the other hand, using gradient learning, we found methods to reduce the samples required for an effective search, as EvoGrad and EvoGrad^2 maintain a steady success rate across dimensions.

Budget-Constrained Applications

In our second experiment, we analyze how different algorithms perform under varying application constraints to identify the most suitable method for specific use cases. Figure 4 presents an analysis of critical considerations that directly impact real-world deployment decisions. Fig. 4(a) illustrates algorithm performance under different budget constraints. While the gradient-based method starts slower (due to warm-up training that consumes early budget), both EvoGrad and EvoGrad^2 quickly pick up pace, outperforming competitors around 8K samples. While CMA has an advantage in the low-budget regime, our algorithm excels when high-accuracy solutions are required, as shown in Table 1; EvoGrad and EvoGrad^2 show a clear advantage in solving the function with minimal budget. For low-budget applications where function evaluations are expensive, results show that although CMA variants initially improve performance efficiently, they plateau earlier. In contrast, EvoGrad and EvoGrad^2 demonstrate superior long-term performance, making them suitable when the total budget allows extended optimization runs.

High-Accuracy Requirements

Figure 4(b) examines success rates as a function of optimality gap, which is crucial for high-precision applications. The results demonstrate that EvoGrad and EvoGrad^2 consistently maintain higher success rates across all precision thresholds. This is particularly important for applications such as adversarial attack generation or hyperparameter tuning, where finding near-optimal solutions is critical.

Finally, Fig. 4(c) and our t-test analysis (See App. in full paper (Kfir et al. 2025)) confirm that our gains are statistically significant. For all baseline, the p-value is below 10^{-6}, indicating that both EvoGrad and EvoGrad^2 outperform alternative methods well beyond the α = 0.05 threshold.

Ablation

For EvoGrad algorithm (Sec. 5), we explored different weighting schemes for biased sampling. Our final approach uses CMA's covariance matrix to create the importance sampler, which adapts to the local geometry. We compared this against simpler alternatives: EvoGrad-0.1 uses the softmax function to create W. Assigning probabilities for each sample $x \in \mathcal{D}_k$ based on $f(x)$. Table 1 shows that the adaptive covariance matrix significantly outperforms simpler weighting schemes, particularly in higher dimensions.

For HGrad (Sec. 4), computing second-order derivatives can be memory-intensive. We found that detaching the Jacobian from the computational graph (HGrad vs HGrad-Attached in Tab. 1) maintains nearly identical performance while reducing memory overhead, to improve scalability.

Hyperparameter Tolerance

We assess the algorithm's robustness to hyperparameter tuning. Our objective is to show which hyperparameters have an impact on the algorithm's performance and how to set those parameters. We conducted systematic experiments to

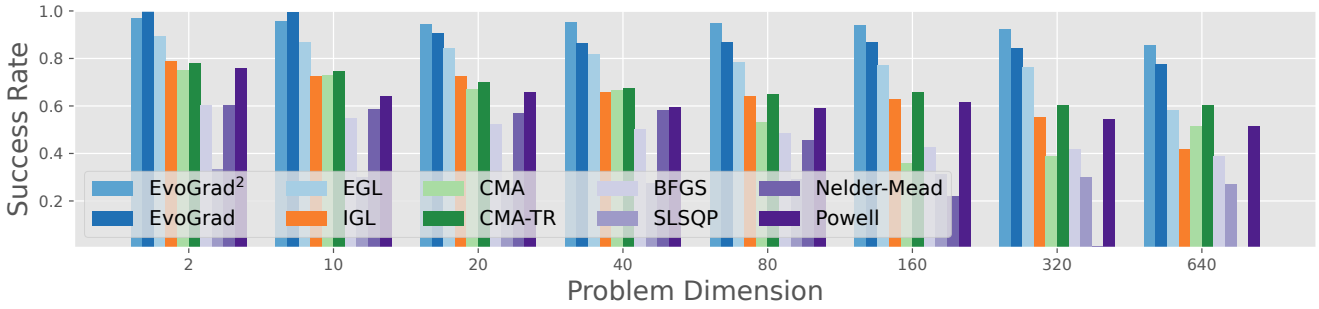


Figure 3: The probability of each algorithm to solve a problem in each dimension

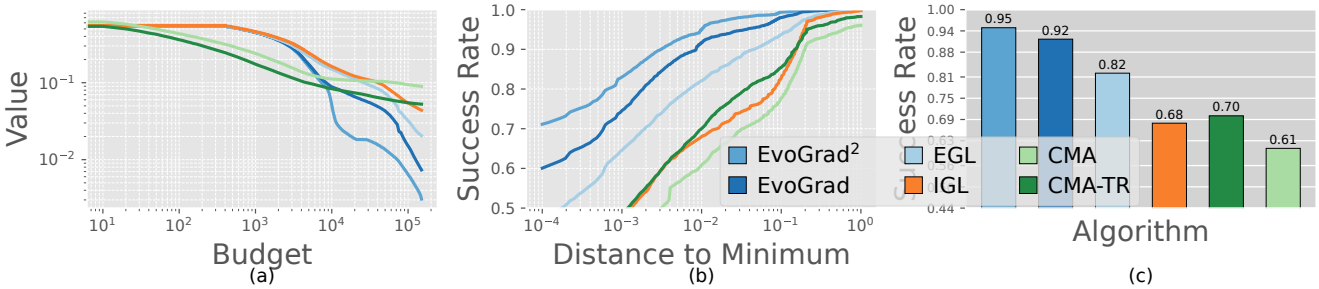


Figure 4: Experiment results against the baseline: (a) Convergence for all our algorithms against baseline algorithms, (b) Success rate as a function of the normalized distance from the best-known solution, (c) Percentage of solved algorithms when the distance from the best point is 0.01

assess this, modifying several hyperparameters and analyzing their effects. Table 2 (Also, see App. in full paper (Kfir et al. 2025)) reports the coefficient of variation (CV), defined as $CV = \frac{\sigma}{\mu}$, across different hyperparameter sweeps, highlighting the algorithm's stability under varying conditions.

Our findings show that hyperparameters such as the epsilon factor, shrink factor, and learning rate (LR) have cumulative effects: small variations may seem negligible but can significantly impact performance over time. In the App. (See full paper (Kfir et al. 2025)), we establish the step size–epsilon relationship needed for progress, while the shrink factor should align with the budget to maximize explored sub-problems. Thus, fine-tuning these parameters is critical. By contrast, network structure (layers and size) had little effect, suggesting that the Taylor loss enables effective learning even with simple architectures, and greater complexity does not necessarily improve results.

Metric	Networks	ε-Factor	LR	Gradient LR
CV	0.0429	0.1676	0.2328	0.1263

Table 2: Coefficient Variation ($CV = \frac{\sigma}{\mu}$) over Hyperparameter sweep experiment. TR SF = trust region shrink factor.

7 High Dimensional Applications

Adversarial Attacks

As powerful vision models like ResNet (Targ, Almeida, and Lyman 2016) and Vision Transformers (ViT) (Han et al. 2022) grow in prominence, adversarial attacks have become a significant concern. These attacks subtly modify inputs, causing models to misclassify them, while the perturbation remains imperceptible to both human vision and other classifiers (Tang et al. 2019). We can define it formally:

$$x_a^* = \arg\min_x d(x, x_a) \quad \text{s.t.} f(x) \neq f(x_a) \tag{7}$$

f is the classifier and d a distance metric between elements.

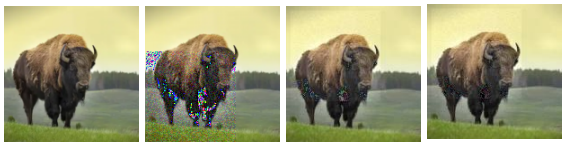
Recent studies have extended adversarial attacks to domains like AI-text detection (Sadasivan et al. 2024) and automotive sensors (Mahima et al. 2024). These attacks prevent tracking and detection, posing risks to both users and pedestrians. Adversarial attacks are classified into black-box and white-box methods. Black-box attacks only require query access, while white-box methods use model gradients to craft perturbations (Machado, Silva, and Goldschmidt 2021; Cao et al. 2019). Despite some black-box methods relying on surrogate models (Dong et al. 2018; Xiao et al. 2018; Madry et al. 2017; Goodfellow, Shlens, and Szegedy 2014), approaches like (Tu et al. 2019) generate random samples to approximate gradient estimation, though they are

computationally expensive. Other methods use GAN networks to search latent spaces for adversarial examples (Liu et al. 2021; Sarkar et al. 2017). Still, they depend on existing GANs and their latent space diversity.

Our EvoGrad^2 method offers a true black-box approach with precise perturbation control, avoiding gradient backpropagation. EvoGrad^2 directly optimizes perturbations to maintain low distortion while fooling the model, handling high-dimensional spaces with over 30K parameters.

Metric	CMA	EvoGrad	EvoGrad^2	CMA+EvoGrad
Accuracy	0.02	0.02	0.02	0.02
MSE	0.05	**0.001**	**0.001**	0.001
Time	**20m**	6H	7H	3H

Table 3: Methods' comparison on Accuracy, MSE, Convergence time.



(a) Original (b) CMA (c) EvoGrad (d) EvoGrad^2

Figure 5: Adversarial examples generated by EvoGrad and CMA against the ImageNet model.

Methodology: We tested our algorithms against classifiers trained on MNIST, CIFAR-10, and ImageNet, aiming to minimize Eq. (7). To generate adversarial images with minimal distortion, we developed a penalty that jointly minimizes MSE and CE-loss (See appendix in full paper (Kfir et al. 2025)). This approach successfully fooled the model, evading the top 5 classifications.

Results: We evaluated five different configurations: CMA, EvoGrad, HGrad, EvoGrad^2, and a combination of both (CMA+EvoGrad), where the CMA run provides the initial guess for an EvoGrad run. While CMA alone was not able to converge to a satisfying adversarial example, the combined CMA+EvoGrad enjoyed the rapid start of CMA with the robustness of EvoGrad, s.t. it was able to find a satisfying adversarial example half the computation time of EvoGrad and EvoGrad^2.

Code Generation

The development of large language models (LLMs) such as Transformers (Vaswani 2017) have advanced code generation (Dehaerne et al. 2022). Despite these strides, fine-tuning outputs based on parameters measured post-generation remains challenging. Recent algorithms have been developed to generate code tailored for specific tasks using LLMs. For instance, FunSearch (Romera-Paredes et al. 2024) generates new code solutions for complex tasks, while Chain of Code (Li et al. 2023) incorporates reasoning to detect and correct errors in the output code. Similarly, our method uses black-box optimization to guide code generation for runtime efficiency. Building on (Zhang et al. 2024), which links LLM

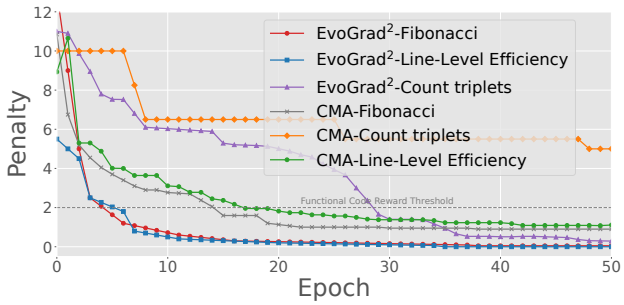


Figure 6: Code generation experiments: penalty over time.

expertise to a small parameter set, we fine-tuned the embedding layer to reduce Python code runtime. Using LoRA (Hu et al. 2021), we optimized the generated code based on execution time, scaling up to $\sim$ 200k parameters.

Fibonacci: We tested this approach by having the model generate a Fibonacci function. Initial results were incorrect, but optimization guided the model to a correct and efficient solution. In the appendix (See (Kfir et al. 2025)), we illustrate this progression, with the 25th step showing an optimized version.

Line-Level Efficiency Enhancements: We tested the model's ability to implement small code efficiencies, such as replacing traditional for-loops with list comprehensions. The algorithm optimized the order of four functions—'initialize', 'start', 'activate', and 'stop'—each with eight variants, minimizing overall runtime by optimizing the function order.

Code Force: For a more complex problem, we used the Count Triplets challenge from Codeforces(https://codeforces.com/). While the model initially struggled, once it found a correct solution, the algorithm further optimized it for runtime performance (See Appendix in the full paper (Kfir et al. 2025)).

Discussion: Our method generates correct solutions while applying micro-optimizations. In simple tasks like Fibonacci, EvoGrad^2 converged on an optimal solution (see Fig. 6), and in more complex problems, it improved the initial solutions. However, for harder tasks, the LLM may generate incorrect code, hindering run-time optimization. To address this, stronger models or methods that prioritize solution correctness are needed. This ensures valid solutions are generated first and then optimized for performance.

8 Conclusion

We introduced EvoGrad^2, a novel black-box optimizer that integrates local gradient learning with global evolutionary statistics, making it well-suited for high-dimensional, non-convex problems. By incorporating 2nd-order Taylor approximation with Hessian correction, it improves gradient estimation and, to our knowledge, is the first method to combine neural gradient learning with evolutionary algorithms. EvoGrad^2 also opens promising avenues for core learning problems, such as RLVR tuning of LLMs and accelerating gradient-based generation in diffusion models.

References

Alarie, S.; Audet, C.; Gheribi, A. E.; Kokkolaras, M.; and Le Digabel, S. 2021. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization*, 9: 100011.

Anil, R.; Gupta, V.; Koren, T.; Regan, K.; and Singer, Y. 2020. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*.

Audet, C.; Hare, W.; Audet, C.; and Hare, W. 2017a. Chapter 1: The begining of DFO algorithms in derivative-free and blackbox optimization. *Derivative-Free and Blackbox Optimization*, 11–15.

Audet, C.; Hare, W.; Audet, C.; and Hare, W. 2017b. Chapter 3: The begining of DFO algorithms in derivative-free and blackbox optimization. *Derivative-Free and Blackbox Optimization*, 33–47.

Audet, C.; Hare, W.; Audet, C.; and Hare, W. 2017c. DIRECT search in derivative-free and blackbox optimization. *Derivative-Free and Blackbox Optimization*, 93–156.

Audet, C.; Hare, W.; Audet, C.; and Hare, W. 2017d. Genetic methods in derivative-free and blackbox optimization. *Derivative-Free and Blackbox Optimization*, 57–73.

Audet, C.; Hare, W.; Audet, C.; and Hare, W. 2017e. Model-based methods in derivative-free and blackbox optimization. *Derivative-Free and Blackbox Optimization*, 156–218.

Back, T. 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Bertsekas, D. 2015. *Convex optimization algorithms*. Athena Scientific.

Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.-L.; et al. 2023. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2): e1484.

Bouneffouf, D.; Rish, I.; and Aggarwal, C. 2020. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. IEEE.

Braun, C. V.; Lange, R. T.; and Toussaint, M. 2024. Stein Variational Evolution Strategies. *arXiv preprint arXiv:2410.10390*.

Busetti, F. 2003. Simulated annealing overview. *World Wide Web URL www. geocities. com/francorbusetti/saweb. pdf*, 4.

Cao, Y.; Xiao, C.; Yang, D.; Fang, J.; Yang, R.; Liu, M.; and Li, B. 2019. Adversarial objects against lidar-based autonomous driving systems. *arXiv preprint arXiv:1907.05418*.

Chen, P.-Y.; Zhang, H.; Sharma, Y.; Yi, J.; and Hsieh, C.-J. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 15–26.

Clerc, M. 2010. *Particle swarm optimization*, volume 93. John Wiley & Sons.

Dehaerne, E.; Dey, B.; Halder, S.; De Gendt, S.; and Meert, W. 2022. Code generation using machine learning: A systematic review. *Ieee Access*, 10: 82434–82455.

Ding, Y.; Zhang, L. L.; Zhang, C.; Xu, Y.; Shang, N.; Xu, J.; Yang, F.; and Yang, M. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 9185–9193.

Gehring, C.; Coros, S.; Hutter, M.; Bloesch, M.; Fankhauser, P.; Hoepflinger, M. A.; and Siegwart, R. 2014. Towards automatic discovery of agile gaits for quadrupedal robots. In *2014 IEEE international conference on robotics and automation (ICRA)*, 4243–4248. IEEE.

Golovin, D.; Karro, J.; Kochanski, G.; Lee, C.; Song, X.; and Zhang, Q. 2019. Gradientless descent: High-dimensional zeroth-order optimization. *arXiv preprint arXiv:1911.06317*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. 2022. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1): 87–110.

Hansen, N. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

Hansen, N.; Auger, A.; Ros, R.; Finck, S.; and Pošík, P. 2010. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, 1689–1696.

Hansen, N.; Auger, A.; Ros, R.; Mersmann, O.; Tušar, T.; and Brockhoff, D. 2021. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optimization Methods and Software*, 36: 114–144.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Kfir, Y.; Sarafian, E.; Kraus, S.; and Louzoun, Y. 2025. EvoGrad: Evolutionary-Weighted Gradient and Hessian Learning. arXiv:2502.04829.

Kraft, D. 1988. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*.

Kunpeng, P.; Yugang, L.; and Xiabi, L. 2012. Descent search with mean direction evolution strategies based on GPU with

CUDA. In *2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 298–304. IEEE.

Lesage-Landry, A.; Taylor, J. A.; and Shames, I. 2020. Second-order online nonconvex optimization. *IEEE Transactions on Automatic Control*, 66(10): 4866–4872.

Li, C.; Liang, J.; Zeng, A.; Chen, X.; Hausman, K.; Sadigh, D.; Levine, S.; Fei-Fei, L.; Xia, F.; and Ichter, B. 2023. Chain of code: Reasoning with a language model-augmented code emulator. *arXiv preprint arXiv:2312.04474*.

Lillicrap, T. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, B.; Guo, Y.; Jiang, J.; Tang, J.; and Deng, W. 2021. Multi-view correlation based black-box adversarial attack for 3D object detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1036–1044.

Liu, F.-Y.; Li, Z.-N.; and Qian, C. 2020. Self-Guided Evolution Strategies with Historical Estimated Gradients. In *IJCAI*, 1474–1480.

Liu, Q.; and Wang, D. 2016. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29.

Liu, S.; Chen, P.-Y.; Kailkhura, B.; Zhang, G.; Hero III, A. O.; and Varshney, P. K. 2020. A Primer on Zeroth-Order Optimization in Signal Processing and Machine Learning: Principals, Recent Advances, and Applications. *IEEE Signal Processing Magazine*, 37(5): 43–54.

Loshchilov, I.; Schoenauer, M.; and Sebag, M. 2013. Bi-population CMA-ES agorithms with surrogate models and line searches. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, 1177–1184.

Machado, G. R.; Silva, E.; and Goldschmidt, R. R. 2021. Adversarial machine learning in image classification: A survey toward the defender's perspective. *ACM Computing Surveys (CSUR)*, 55(1): 1–38.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Maheswaranathan, N.; Metz, L.; Tucker, G.; Choi, D.; and Sohl-Dickstein, J. 2019. Guided evolutionary strategies: Augmenting random search with surrogate gradients. In *International Conference on Machine Learning*, 4264–4273. PMLR.

Mahima, K. T. Y.; Perera, A. G.; Anavatti, S.; and Garratt, M. 2024. Toward Robust 3D Perception for Autonomous Vehicles: A Review of Adversarial Attacks and Countermeasures. *IEEE Transactions on Intelligent Transportation Systems*, 1–27.

Nelder, J. A.; and Mead, R. 1965. A Simplex Method for Function Minimization. *Comput. J.*, 7: 308–313.

Nocedal, J.; and Wright, S. J. 2018. Numerical Optimization. In *Fundamental Statistical Inference*.

Olof, S. S. 2018. A comparative study of black-box optimization algorithms for tuning of hyper-parameters in deep neural networks.

Powell, M. J. D. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.*, 7: 155–162.

Prabhu, S. G. R.; Seals, R. C.; Kyberd, P. J.; and Wetherall, J. C. 2018. A survey on evolutionary-aided design in robotics. *Robotica*, 36(12): 1804–1821.

Rimon, Z.; Jurgenson, T.; Krupnik, O.; Adler, G.; and Tamar, A. 2024. Mamba: an effective world model approach for meta-reinforcement learning. *arXiv preprint arXiv:2403.09859*.

Romera-Paredes, B.; Barekatain, M.; Novikov, A.; Balog, M.; Kumar, M. P.; Dupont, E.; Ruiz, F. J.; Ellenberg, J. S.; Wang, P.; Fawzi, O.; et al. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995): 468–475.

Sadasivan, V. S.; Kumar, A.; Balasubramanian, S.; Wang, W.; and Feizi, S. 2024. Can AI-Generated Text be Reliably Detected? arXiv:2303.11156.

Sarafian, E.; Sinay, M.; Louzoun, Y.; Agmon, N.; and Kraus, S. 2020. Explicit Gradient Learning for Black-Box Optimization. In *ICML*, 8480–8490.

Sarkar, S.; Bansal, A.; Mahbub, U.; and Chellappa, R. 2017. UPSET and ANGRI: Breaking high performance image classifiers. *arXiv preprint arXiv:1707.01159*.

Tang, S.; Huang, X.; Chen, M.; Sun, C.; and Yang, J. 2019. Adversarial attack type I: Cheat classifiers by significant changes. *IEEE transactions on pattern analysis and machine intelligence*, 43(3): 1100–1109.

Tang, Y. 2021. Guiding Evolutionary Strategies with Off-Policy Actor-Critic. In *AAMAS*, 1317–1325.

Targ, S.; Almeida, D.; and Lyman, K. 2016. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*.

Tu, C.-C.; Ting, P.; Chen, P.-Y.; Liu, S.; Zhang, H.; Yi, J.; Hsieh, C.-J.; and Cheng, S.-M. 2019. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 742–749.

Vaswani, A. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Xiao, C.; Li, B.; Zhu, J.-Y.; He, W.; Liu, M.; and Song, D. 2018. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*.

Zhang, W.; Wan, C.; Zhang, Y.; Cheung, Y.-m.; Tian, X.; Shen, X.; and Ye, J. 2024. Interpreting and Improving Large Language Models in Arithmetic Calculation. *arXiv preprint arXiv:2409.01659*.

Zhen, L.; Wang, K.; Hu, H.; and Chang, D. 2014. A simulation optimization framework for ambulance deployment and relocation problems. *Computers & Industrial Engineering*, 72: 12–23.