IEOR E4540 Research Project Report: ES Optimization on the MNIST Classification Task

Xingchen (Estella) Ye
xy2527@columbia.edu
May 2, 2023

In this report, we investigate the performance of Evolution Strategies (ES) optimization in comparison to backpropagation for training neural networks on the MNIST classification task. Three studies are conducted to evaluate the impact of various factors on the performance of ES-trained models. The first study compares ES optimization with backpropagation using Stochastic Gradient Descent (SGD) and a fixed learning rate. We find that ES-trained models exhibit better convergence, while backpropagation with TensorFlow built-in optimizer achieves faster training time. The second study explores the effect of different numbers of perturbations ($N = 10, 50, 100$) on ES training, revealing that higher perturbation numbers result in faster and better convergence at the cost of increased training time. The third study compares vanilla ES, antithetic pairs, and FD-style gradient sensing methods, showing that with reward normalization, the three Monte Carlo estimators yield similar convergence curves and downstream accuracies. We further experiment with antithetic rewards using σ_R scaling, which leads to slower convergence but higher validation accuracy.

I. Background

A. Introduction to Evolutionary Strategy

Evolutionary strategy (ES) is a black-box optimization algorithm, belonging to the family of evolutionary algorithms inspired by natural selection. In the context of machine learning, ES and black-box optimizations have been proposed as an alternative to backpropagation. Although backpropagation has been widely used in training neural networks, it relies on the computation of gradients and Hessian matrices. In contrast, ES and black-box optimizations do not depend on gradients, making them particularly suitable for evaluating non-differentiable objective functions. Additionally, ES algorithms benefit from parallel computation, significantly reducing training time.

B. Vanilla Evolutionary Strategy

i) Initialization

Consider the t-th iteration of training and a set of model parameters, denoted by θ_t. Generate N pseudo offspring θ'_t through adding random noises v_i to θ_t. In OpenAI's ES experiment [1], perturbation is applied to all parameters. Sample noise vector $v_i, i = 0, ..., N$ independently from multivariate Gaussian distribution $\mathcal{N}(0, 1)$. ES-related parameters include the number of perturbations N and noise standard deviation σ, which is reflected through multiplications on the Gaussian noise.

ii) Evaluation

For each pseudo-offspring i and the perturbed parameter $\theta + \sigma v_{t,i}$, compute the sample reward vector

$$r_{t,i} = F(\theta_t + \sigma v_{t,i}) \tag{1}$$

Normalize $r_{t,i}$ over all is. Note that, in the MNIST classification task, the reward function is negative cross-entropy.

iii) Update

After combining the rewards for all episodes, we estimate the stochastic gradient by

$$\begin{aligned} g_t^{ES} &= \frac{1}{N\sigma} \sum_{i=1}^{N} v_{t,i} r_{t,i} \\ &= \frac{1}{N\sigma} \sum_{i=1}^{N} v_{t,i} F(\theta_t + \sigma v_{t,i}) \end{aligned} \tag{2}$$

Then, update the model parameters for iteration $t + 1$.

$$\theta_{t+1} = \theta + \alpha g_t^{ES} \tag{3}$$

where α is the learning rate.

C. Gradient sensing as a Monte Carlo estimation

The ES stochastic gradients g_t^{ES} can also be treated using Monte Carlo estimation. In this paper, we also implemented the following two methods for comparison. Similar to i) and ii) in the above section, $v_{t,i} \sim \mathcal{N}(0, 1)$ and gradients are updated using the SGD equation, with the variations in gradient estimation.

ii) **Gradient Estimator with Antithetic Pairs** Salimans et al. [2]:

$$g_t^{anti} = \frac{1}{2N\sigma} \sum_{i=1}^{N} v_{t,i}(F(\theta_t + \sigma v_{t,i}) - F(\theta_t - \sigma v_{t,i})) \tag{4}$$

iii) **FD-style Gradient Estimator** Choromanski et al. [3]:

$$g_t^{FD} = \frac{1}{N\sigma} \sum_{i=1}^{N} v_{t,i}(F(\theta_t + \sigma v_{t,i}) - F(\theta_t)) \tag{5}$$

II. Study 1: ES vs Backpropagation

A. Model and Method

Due to the limitations of the Colab environment, we designed a simpler model with a fewer number of parameters compared to the sophisticated neural network with 3,274,634

parameters used by Smith et al. [1]. The model has 233,600 parameters and consists of three fully connected layers with 256, 128, and 10 units respectively. The mini-batch size is 50 and the optimizer is SGD with a learning rate of 0.001. The mini-batch size is set to 50, and the optimizer used is SGD with a learning rate of 0.001. In all experiments, the ES parameter σ is fixed at 0.002, as suggested by Smith et al. [1] to be the most effective through preliminary experiments on the MNIST dataset. In study 1, models are trained for 5 epochs, in a total of 6,000 iterations. Moreover, in our preliminary experiments with ES raw rewards obtained from Equation 1, training ES models always incurs a gradient vanishing problem. Thus, in study 1 and study 2, rewards are normalized before feeding into Equation 2 to calculate gradients, and we will dive into variance reduction techniques in study 3.

B. Results and Analysis

By fixing the optimizer to SGD with a learning rate of 0.001, we compare the baseline performance of the ES-trained model with the number of perturbations $N = 10$ and backpropagation. The results are presented in Table I and Figure 1. The convergence of training and validation accuracy is illustrated in Figure 2.

TABLE I
PERFORMANCE OF BACKPROPAGATION VS ES

Model	Validation Accuracy	Training Time
ES (N=10)	0.7514	24m17s
Backpropagation	0.7058	2m55s

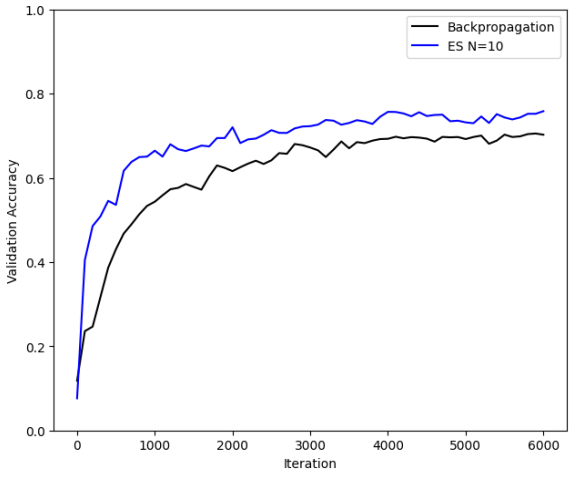

Fig. 1. ES vs Backpropagation

Compared to ES with $N = 10$, backpropagation using TensorFlow 1.x built-in *GradientDescent* optimizer exhibits a faster training time but at the expense of lower validation accuracy. Since ES maintains a population of candidate solutions, it encourages exploration of the search space, reducing the likelihood of premature convergence to suboptimal solutions.

Although ES demonstrates faster convergence and higher validation accuracy, the results are far from ideal as Smith et al. [1] reports ES's validation accuracy of 0.99 on the MNIST classification task. Backpropagation can also achieves higher scores on the MNIST dataset for its low dimensionality and simplicity, especially when employing adaptive learning rates and optimizers such as *Adam* and *AdaGrad*. Furthermore, ES can be more robust to hyperparameter settings compared to backpropagation. In certain cases, this robustness could lead to improved performance on tasks like MNIST classification, especially when optimal hyperparameters for backpropagation are not known. In the next section, we will examine ES-trained model dependence on the number of perturbations N, which will provide insights into tuning ES-trained models for higher validation accuracy.

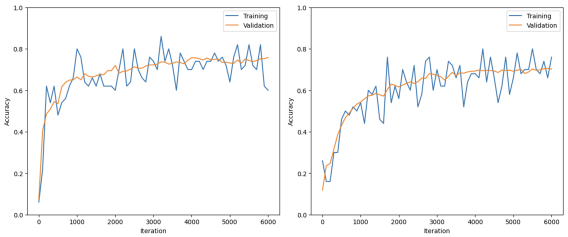

Fig. 2. Validation Accuracy (Orange) vs Training Accuracy (Blue) (Left: ES (N=10), Right: Backpropagation)

III. STUDY 2: ES AND NUMBER OF PERTURBATIONS

A. Model and Method

The number of perturbations N refers to the number of candidate directions generated in each iteration of the optimization process. A larger number of perturbations lead to more exploration of the search space. Also, the generated noise table V for each iteration has dimensions *Number of Model Parameters* $\times$ *Number of perturbations*, so increasing the number of perturbations also results in increased computational costs.

To investigate the effects of the number of perturbations N on accuracy and training time, experiments with $N = 10, 50, 100$ are conducted. The rest of the training hyperparameters are kept the same, and reward vector $F(\theta + \sigma v_i)$ is normalized in each iteration.

B. Results and Analysis

Training with 2 epochs and in total 2,400 iterations, we obtain the accuracy in Table II and convergence curves in Figure 3 and Figure 4. As expected, higher numbers of perturbations result in higher validation accuracy. Salimans, et al. [2] demonstrates that increasing the number of perturbation to 50k can obtain 0.99 validation accuracy on the MNIST task. However, the trade-off in computational cost is considerable, as time complexity increases approximately proportionally with N in our experiment.

Plotting the three validation accuracy curves against the iteration number, we observe that a higher number of perturbations leads to a more rapid increase in the initial 250 iterations. The rapid increase in the first hundreds of iterations could be attributed to more exploration conducted in the search space of model weights. Thus, increasing N would achieve a decent validation accuracy in fewer iterations.

TABLE II
PERFORMANCE ON DIFFERENT NUMBER OF PERTURBATIONS

N	Validation Accuracy	Training Time
10	0.7058	9m52s
50	0.7983	32m50s
100	0.8099	60m11s

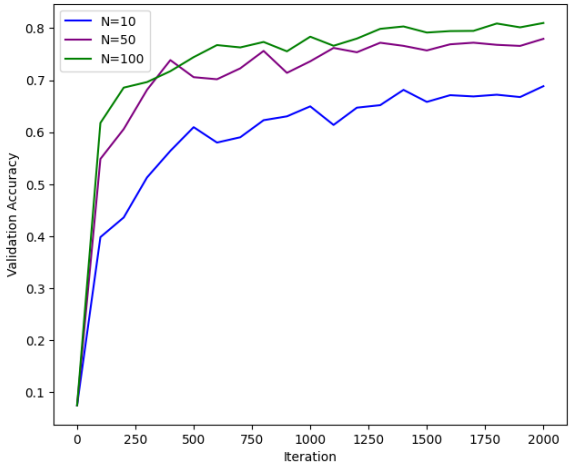

Fig. 3. Validation Accuracy vs Iteration for Different N

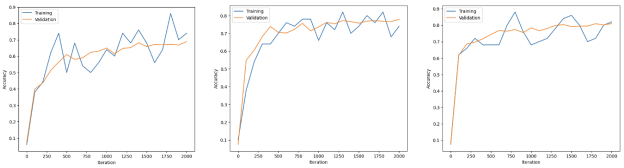

Fig. 4. Validation Accuracy (Orange) vs Training Accuracy (Blue) for Different Ns (Left to Right: N=10, 50, 100)

IV. STUDY 3: MONTE CARLO ESTIMATOR

A. Model and Method

In the previous sections, experiments are conducted with vanilla ES gradient from Equation 2. We also tested the three Monte Carlo estimators on the MNIST classification test. The model architecture remains the same, with σ fixed at 0.002 and the number of perturbations N set to 50. The only variation occurs in the reward collection and gradient calculation equations. The gradient estimator with antithetic pairs collects $2N$ rewards from two opposite blurring directions, and the FD-style estimator subtracts the reward calculated from the model parameters for each sampled reward. Then, gradients are calculated using Equation 4 and Equation 5. Additionally, there are large variations in the observed raw rewards and the calculated reward differences across iterations. Scaling methods, including normalization and σ_R scaling approaches, are applied for variance reduction.

B. Results and Analysis

We notice that for all three estimators, there are significant variations in the observed raw rewards and the calculated reward differences across iterations. In our experiments, directly using the raw values calculated from Equation 1 and 2 (Vanilla ES), 4 (Antithetic), and 5 (FD-style) into the update equation result in gradient vanishing. Therefore, normalization is also applied to the reward difference vectors of the antithetic pairs $F(\theta_t + \sigma v_{t,i}) - F(\theta_t - \sigma v_{t,i})$ and FD-style gradient $F(\theta_t + \sigma v_{t,i}) - F(\theta_t)$. With normalization, the model performances are shown in Table III and Figure 5.

TABLE III
PERFORMANCE ON DIFFERENT GRADIENT SENSING METHODS

Estimator	Validation Accuracy	Training Time
Vanilla	0.7983	32m50s
Antithetic	0.7901	45m14s
FD-style	0.7821	38m13s

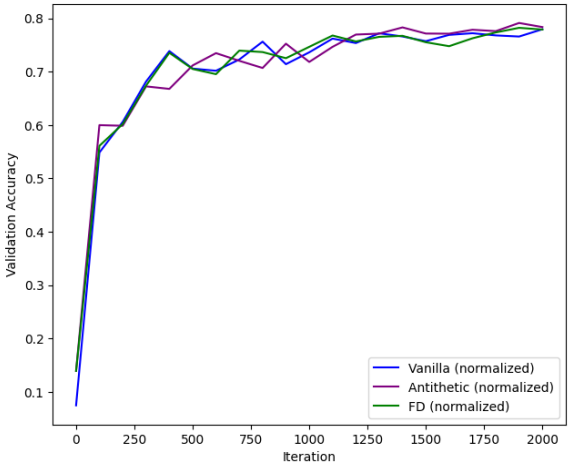

Fig. 5. Comparison of the 3 Gradient Sensing Methods with Normalized Rewards

In this environment, the three models do outperform each other in terms of validation accuracy. The rates of convergence are very similar across the three models after normalization, as illustrated in Figure 5.

The performance of the normalized FD-style gradient estimator is explainable: Since the FD-style gradients differ from the vanilla ES gradients only by subtracting a constant vector on the rewards, the normalized FD-style rewards are the same as the vanilla ES-style rewards, and it reduces to the ES-style vanilla gradient sensing with normalization.

The variation in training time shows that vanilla ES gradient sensing is the most efficient as we expected. Collecting $2N$ rewards for the antithetic pairs of perturbations result in the highest computational costs, followed by calculating one reward $F(\theta_t)$ per each iteration for the FD-style gradient sensing.

C. Variance Reduction Techniques

Since normalization fails to show the improvements induced by the antithetic and FD-style gradients from vanilla ES gradients, we focus on antithetic pairs to experiment with variance reduction techniques.

Antithetic pairs compute $2N$ rewards which result a larger variations of the differences $F(\theta_t + \sigma v_{t,i}) - F(\theta_t - \sigma v_{t,i})$. To reduce the variance, H Mania et al. [4] suggest scaling the

update step by the standard deviation σ_R of the $2N$ rewards collected. That is, find the standard deviation σ_R of $F(\theta_t - \sigma v_{t,i})$ and $F(\theta_t + \sigma v_{t,i})$ and scale the update step size. To express it in an equation,

$$g_t^{anti,\sigma_R scaled} = \frac{1}{N\sigma_R} \sum_{i=1}^{N} v_{t,i}(F(\theta_t + v_{t,i}) - F(\theta_t)) \quad (6)$$

In the experiment, we trained two antithetic models with $N = 10$ and $N = 50$. The effect of scaling by σ_R is revealed through more iterations, so the $N = 10$ model is trained with 2 epochs, while $N = 50$ is trained with 3 epochs. The results are shown in Table IV and Figure 6.

TABLE IV
PERFORMANCE ON DIFFERENT NUMBER OF PERTURBATIONS

Model	Validation Accuracy	Training Time
N=10, Norm	0.7269	10m22s
N=10, Scaled	0.7558	11m37s
N=50, Norm	0.8099	60m55s
N=50, Scaled	0.8150	67m50s

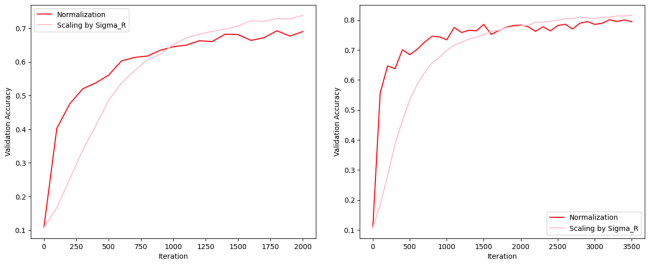

Fig. 6. Normalization vs Scaling by σ_R for Antithetic Pairs Estimator (Left: N=10, Epoch=2; Right: N=50, Epoch=3)

For both sets of parameters, the validation accuracy of scaling by σ_R initially falls behind in the first thousand iterations before converging to a more optimal solution. The initial delay can be attributed to the inherently smaller standard deviation of N rewards in the same direction compared to the standard deviation of $2N$ rewards in opposite directions, leading to a larger step size for normalization. On the other hand, normalization converges to suboptimality due to the smaller step size and reduced exploration caused by subtracting the mean from the rewards, which can result in an under-fitting model.

V. CONCLUSION

In this paper, we conducted three studies. In the first study, we compared ES optimization with backpropagation on the MNIST classification task, and we showed that with SGD and a fixed learning rate of 0.001, the ES-trained model yields better convergence, whereas backpropagation with TensorFlow 1.x built-in optimizer has a faster training time. In the second study, we experimented with ES training using different numbers of perturbations. Testing $N = 10, 50, 100$ reveals that the higher the number of perturbations, the faster and better convergence of the ES-trained model. The trade-off is the roughly proportionally increased time complexity. In the third study, we compared vanilla ES, antithetic pairs, and FD-style gradient sensing methods in training. We demonstrated that with reward normalization, the three Monte Carlo estimators yield similar convergence curves and downstream accuracy. Considering the dependence of Monte Carlo estimators on the variance reduction technique applied to the collected rewards, we conducted a further experiment on antithetic rewards using σ_R scaling, which exhibits slower convergence but higher validation accuracy.

REFERENCES

[1] Smith, L. A., Williams, C. K. (2017). On the relationship between the OpenAI evolution strategy and stochastic gradient descent. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 2837-2846). JMLR.

[2] Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I. (2017). Evolution Strategies as a Scalable Alternative to Reinforcement Learning. arXiv preprint arXiv:1703.03864.

[3] Choromanski, K., Xu, H., Sohl-Dickstein, J., Sutskever, I. (2018). Structured Evolution with Compact Architectures for Scalable Policy Optimization. arXiv preprint arXiv:1804.02409.

[4] Mania, H., Guy, A., Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. arXiv preprint arXiv:1803.07055.

[5] "Evolution Strategies: A Review": Weng, L. (2019, September 5). Evolution Strategies: A Review. Retrieved from https://lilianweng.github.io/posts/2019-09-05-evolution-strategies/