EFFECTIVE AND STABLE NEURON MODEL OPTIMIZATION BASED ON AGGREGATED CMA-ES

Han Xu, Takahiro Shinozaki

Tokyo Institute of Technology,
Kanagawa, Japan

Ryota Kobayashi

National Institute of Informatics,
Tokyo, Japan

ABSTRACT

Computer simulations have facilitated our understanding of the dynamic behavior of the brain and the effect of the medical treatment such as deep brain stimulation. For improving the simulation model, it is essential to develop a method for optimizing parameters of a neuron model from available experimental data. In this paper, we apply Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) to the parameter optimization problem, and compare it with widely used conventional approaches including genetic algorithm (GA) and the Nelder-Mead method. A problem we have observed with CMA-ES is that the performance highly depends on the initial condition. To overcome the problem, we extend CMA-ES by making an aggregation of evolution. We analyze a public dataset recorded from a rat neocortical neuron, which shows that the proposed approach achieves higher performance than the conventional methods.

Index Terms— CMA-ES, Genetic Algorithm, Brain simulation, parameter optimization

1. INTRODUCTION

Computer simulations have facilitated our understanding of the dynamic behavior of the brain and the effect of the medical treatment such as deep brain stimulation [1]. For improving the simulation model, it is essential to develop a method for optimizing parameters of a neuron model from available experimental data [2, 3].

Various approaches for the parameter optimization problem have been proposed. First approach is based on the gradient-based methods, such as gradient descent [4] and the Levenberg-Marquardt method [5]. Though these methods rapidly converge to the minima, they might suffer from local optimum if the fitness function is not concave. The other approaches are based on derivative-free methods that do not require the gradient of the fitness function, including simulated annealing [6] and evolutionary algorithms [7, 8]. Genetic algorithm (GA) and evolution strategy are popular in evolutionary algorithms [9]. Evolution strategy is similar to GA but uses a real-valued vector as a gene.

Among evolution strategy algorithms, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10] is known as its great performance in various black-box optimization tasks [11]. The application of CMA-ES to neuron model optimization has been suggested in [8], but its performance has not been systematically evaluated. Here we investigate the performance of CMA-ES by comparing it with conventional methods. We found that the performance of CMA-ES highly depends on the initial condition, i.e., CMA-ES performs better than GA on average, however it sometimes performs worse than GA. To overcome this drawback, we propose to make an aggregation of CMA-ES for the optimization.

The remainder of this paper is organized as follows. In Section 2, the evolutionary algorithms are briefly reviewed, and the proposed method is described. In Section 3, a spiking neuron model that is used in our optimization task is explained. Experimental setup is described in Section 4, and the results are shown in Section 5. Finally, conclusions and future works are provided in Section 6.

2. EVOLUTIONARY ALGORITHMS

2.1. Genetic algorithm

GA is a search heuristic motivated by the natural evolution process. This algorithm is based on 1) the selection of genes according to their scores, pruning inferior gene vectors for the next iteration (generation); 2) mating pairs of gene vectors to form child gene vectors that mix the properties of the parents, and 3) mutation of a part of a gene vector to produce new gene vectors. Algorithm 1 shows the process of GA.

2.2. Covariance matrix adaptation evolution strategy

Covariance matrix adaptation evolution strategy (CMA-ES) is an evolution strategy, where a closely related method is the natural evolution strategy [12]. Although both of them have several variations, it has been shown that their core part is mathematically equivalent [13]. Here, we follow the derivation of natural ES as the explanation of CMA-ES [14].

CMA-ES uses a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})$ having a parameter set $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ to represent a gene distri-

Algorithm 1 Genetic Algorithm

for k =1 to population **do**
  Initialize x_k
end for
while not max_generation **do**
  for k=1 to population **do**
    Evaluate individual x_k to obtain fitness $f(x_k)$
  end for
  Generate next-generation individuals from the current generation and their fitness through selection, crossover, and mutation operations
end while
return Best individual x and fitness $f(x)$

bution, where $\boldsymbol{x}$ is the real-valued vector representing a gene, $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$-dimensional covariance matrix, and D is the gene size. Instead of directly maximizing the fitness $f(\boldsymbol{x})$, CMA-ES maximizes an expected value of the fitness $\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}]$ under the Gaussian distribution as shown in Equations (1) and (2). Higher expectation means the Gaussian distribution generates good genes with high probability.

$$\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}] = \int f(\boldsymbol{x})\,\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})dx, \tag{1}$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\,\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}]. \tag{2}$$

To maximize the expectation, the gradient ascent method is used to iteratively update the current parameter set $\boldsymbol{\theta}_n$ starting from an initial parameter set $\boldsymbol{\theta}_0$, as shown in Equation (3).

$$\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1} + \epsilon \nabla_{\boldsymbol{\theta}} \mathbb{E}\left[f(\boldsymbol{x})|\boldsymbol{\theta}\right] \mid_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{n-1}}, \tag{3}$$

where n is an iteration index and $\epsilon\,(>0)$ is a step size. The iteration corresponds to a generation. By using the relation $\nabla \log f = \frac{\nabla f}{f}$ and approximating the integration by sampling, the gradient is expressed by Equation (6).

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}] \mid_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{n-1}} \tag{4}$$

$$= \int \left(f(\boldsymbol{x})\,\nabla_{\boldsymbol{\theta}} \log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta}_{n-1})\right) \mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta}_{n-1})dx \tag{5}$$

$$\approx \frac{1}{K}\sum_{k}^{K} y_k \nabla_{\boldsymbol{\theta}} \log \mathcal{N}(\boldsymbol{x}_k|\boldsymbol{\theta}_{n-1}), \tag{6}$$

$$\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta}_{n-1}),$$

where $\boldsymbol{x}_k$ is a gene sampled from the previously estimated distribution $\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_{n-1})$, and y_k is the evaluated fitness $y_k = f(\boldsymbol{x}_k)$. The set of K samples at an iteration step corresponds to a set of individuals at a generation in an evolution. If we interpret the fitness as a reward and the gene as an action, CMA-ES may be seen as a type of the policy gradient based reinforcement learning using a Gaussian distribution as the policy function [15, 16].

Algorithm 2 CMA-ES

Initialize $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$
while not max_generation **do**
  for k =1 to population **do**
    Sample $\boldsymbol{x}_k$ from $N(\boldsymbol{x}|\boldsymbol{\mu}_{n-1}, \boldsymbol{\Sigma}_{n-1})$
    Evaluate $f(\boldsymbol{x}_k)$
  end for
  Rank $f(\boldsymbol{x}_k)$
  Update $\boldsymbol{\mu}_n$, $\boldsymbol{\Sigma}_n$
end while
return Best individual $\boldsymbol{x}$ and score $f(\boldsymbol{x})$

Although simple gradient ascent may be directly performed using the obtained gradient, CMA-ES uses the natural gradient $\tilde{\nabla}_{\boldsymbol{\theta}}\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}] = \boldsymbol{F}^{-1}\nabla_{\boldsymbol{\theta}}\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\theta}]$ to improve the convergence speed, where $\boldsymbol{F}$ is a Fisher information matrix defined by Equation (7).

$$\boldsymbol{F}\left(\boldsymbol{\theta}\right) = \int \mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})^T d\boldsymbol{x}. \tag{7}$$

The Fisher information matrix can be analytically evaluated for Gaussian distribution. The final update formula for $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$ are obtained as shown in Equation (8).

$$\begin{cases} \hat{\boldsymbol{\mu}}_n = \hat{\boldsymbol{\mu}}_{n-1} + \epsilon_{\boldsymbol{\mu}} \sum_{k=1}^{K} w(y_k)(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1}), \\ \hat{\boldsymbol{\Sigma}}_n = \hat{\boldsymbol{\Sigma}}_{n-1} + \epsilon_{\boldsymbol{\Sigma}} \sum_{k=1}^{K} w(y_k) \\ \qquad \cdot\big((\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_{n-1})^{\intercal} - \hat{\boldsymbol{\Sigma}}_{n-1}\big), \end{cases} \tag{8}$$

where $^{\intercal}$ is the matrix transpose. Note that, as in [10], y_k in Eq. (6) is approximated in Eq. (8) as a weight function $w(y_k)$, which is defined as:

$$w(y_k) = \frac{\max\{0, \log(K/2+1) - \log(\mathrm{R}(y_k))\}}{\sum_{k'=1}^{K} \max\{0, \log(K/2+1) - \log(\mathrm{R}(y_{k'}))\}} - \frac{1}{K}, \tag{9}$$

where $\mathrm{R}(y_k)$ is a ranking function that returns the descending order of y_k among $y_{1:K}$ (i.e., $\mathrm{R}(y_k) = 1$ for the highest y_k, $\mathrm{R}(y_k) = K$ for the smallest y_k, and so forth). This equation only considers the order of y, which makes the updates less sensitive to the evaluation measurements. Algorithm 2 summarizes the CMA-ES optimization procedure.

2.3. Aggregated CMA-ES

While CMA-ES has proven to be efficient, the results tend to largely vary for different trials, that is, the performance highly depends on the initialization. This is maybe because the landscape of the objective function in the gene space is fitted by a Gaussian distribution, which only describes a symmetric distribution with a single peak. To improve CMA-ES for achieving near global optimum result, we propose an approach that runs CMA-ES for multiple trials with different initialization and finds the best individual among all the trials. We refer to

Table 1: The range of initial condition for the model parameters. s_1 and s_2 are defined by $\tau_1 = 20/(1+e^{-s_1})$ and $\tau_2 = 200/(1+e^{-s_2})$, respectively.

Parameter	α_1	α_2	ω	s_1	s_2
Min	50	3	-62	-1	-1
Max	80	7	-32	1	1

this strategy as aggregated CMA-ES, and the number of trials as aggregation size. The original CMA-ES is a special case of the aggregated CMA-ES whose aggregation size is 1. In this way, we have multiple Gaussian distributions in an aggregated evolution process, and can represent complex divergence as a whole.

3. NEURON MODEL

3.1. Model organization

We considered the parameter optimization problem of Multitimescale Adaptive Threshold (MAT) model [17]. The neuron model generates spikes when the potential u exceed the spike threshold θ as shown in Equation (10).

$$\begin{aligned} &\text{If } u(t) \geq \theta(t) \rightarrow \text{Emit a spike at time } t, \\ &C_m \frac{du}{dt} = -g_L(u - E_L) + I_{ex}(t), \\ &\theta(t) = \omega + \sum_{k:k_t<t} \alpha_1 e^{-\frac{t-t_k}{\tau_1}} + \alpha_2 e^{-\frac{t-t_k}{\tau_2}}, \end{aligned} \tag{10}$$

where $C_m = 160$ (pF) is the membrane capacitance, $g_L = 16$ (nS) is the leak conductance, $E_L = -71.5$ (mV) is the leak potential, and $I_{ex}(t)$ (pA) is the injected current. The threshold parameters $\{\omega, \alpha_1, \alpha_2, \tau_1, \tau_2\}$ need a black-box optimization and we apply GA and CMA-ES.

3.2. Performance evaluation

We evaluate the model performance by the coincidence factor Γ [17], which is defined by Equation (11).

$$\Gamma = \frac{N_c - 2f_m N_d \Delta}{N_d + N_m} \times \frac{2}{1 - 2f_m \Delta}, \tag{11}$$

where N_c is the number of coincident spikes with precision $\Delta = 4$ (ms), N_d (N_m) is the number of spikes of the real (model) neuron, and f_m is the spike frequency of the model neuron. The maximum value of $\Gamma = 1$ is achieved only if all the spikes coincide with precision Δ.

4. EXPERIMENTAL SETUP

We analyzed a public dataset from the International Competition on Quantitative Single-Neuron Modeling 2009 [2, 18] (Challenge A). The data consists of the stimulus and the voltage recorded from a rat neocortex neuron.

Table 2: Model performance optimized by grid search and Nelder-Mead method.

Strategy	Grid Search	Nelder-Mead
Coincidence factor (Γ)	0.580	0.618

Five threshold parameters $\{\omega, \alpha_1, \alpha_2, \tau_1, \tau_2\}$ are optimized by applying GA and CMA-ES to the dataset between 17.5 sec and 39 sec. For the initialization, GA needs a set of initial genes of the population size whereas CMA-ES needs a single individual as the mean. We generated these initial genes by random sampling from a multidimensional uniform distribution over an interval shown in Table 1. We used Gaussian mutation and two-point cross-over for GA. We initialized the covariance matrix for CMA-ES by $0.4\boldsymbol{I}$, where $\boldsymbol{I}$ is an identity matrix. The fitness is evaluated by the model performance Γ. We used GA library, DEAP [19][1] and CMA-ES library in Hansen's webpape [2] for implementation.

Table 2 shows the model performance Γ obtained by the grid search and the Nelder-Mead method [20]. Γ obtained by the grid search and the Nelder-Mead method were 0.580 and 0.618, respectively. The Nelder-Mead method is one of the state-of-art methods for neuron model optimization [2, 21]. We first examine the following range by using a grid search: $\alpha_1 \in [5, 10, \cdots, 75]$, $\alpha_2 \in [0.5, 1.5, \cdots, 9.5]$, and $\omega \in [-65, -62, \cdots, -38]$. Then, the Nelder-Mead method is applied from each grid point.

5. RESULTS

We compared the average of the model performance Γ achieved by GA with CMA-ES (Fig. 1). Because the performance depends on the initialization, we calculated the average of 50 trials. Fig. 1A shows how the performance of GA improves during the evolution. Γ increases with the population size and does not converge when the population size is 50. At a population size of 50 (100), Γ was 0.595 (0.599). While the performance increases quickly at the early stage of the evolution (e.g., less than 30 generations), it does not converge at the later stage. For example, Γ is 0.598 (0.599) at 100-th (120-th) generation for a population size of 100. Thus, GA requires a high computational cost to achieve good performance. Fig. 1B shows how the performance of CMA-ES improves during the evolution. Although the performance of CMA-ES increases more slowly than that of GA at the early stage, it converges faster than GA. Before the 100th generation, the elements of the covariance matrix become very small and CMA-ES converges in most of the cases. The improvement resulting from changing the population size

[1] https://github.com/deap/deap

[2] https://www.lri.fr/~hansen/cmaes_inmatlab.html

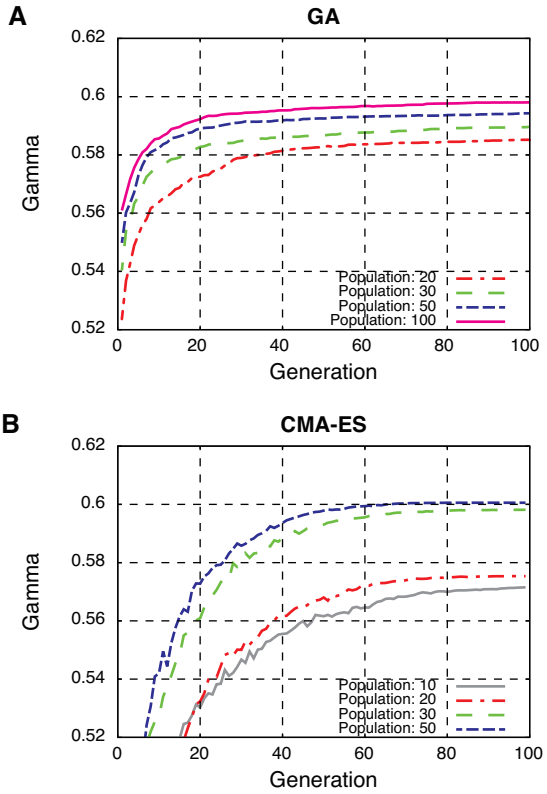

Fig. 1: Comparison of the model performance Γ obtained by GA (A) and CMA-ES (B). The average performance was compared for different population sizes.

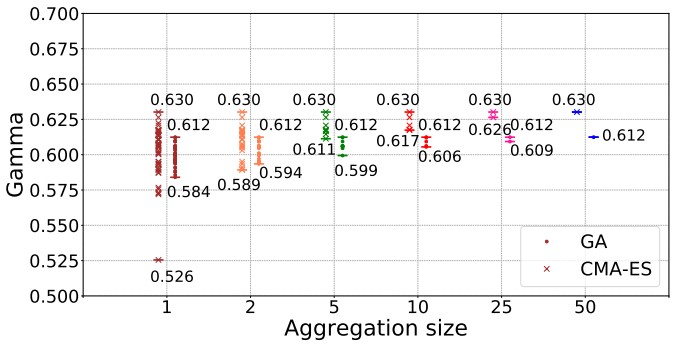

Fig. 2: Distribution of Γ when aggregated GA and CMA-ES were used. The population size and the number of generations were 100 and 150 for GA, and 50 and 100 for CMA-ES.

Table 3: The performance and the parameters of the neuron model optimized by aggregated GA (A-GA) and aggregated CMA-ES (A-CMA-ES).

	A-GA	A-CMA-ES
Coincidence factor (Γ)	0.612	0.630
α_1	70.2	63.4
α_2	7.18	9.10
ω	-47.9	-49.4
τ_1	9.62	9.71
τ_2	90.3	85.6

from 30 to 50 is small, that is, Γ was 0.598 (0.601) for a population size of 30 (50). A population size of 50 seems to be sufficient for CMA-ES. The results suggest that CMA-ES is more computationally efficient than GA.

We examined the effect of the initial condition on the performance (Fig. 2 at aggregation size 1). The performance of CMA-ES is sensitive to the initial condition compared to GA. While the average and the best performance by CMA-ES are better than those by GA, GA sometimes performs better than CMA-ES due to this sensitivity.

To improve the robustness against the initialization, we propose the aggregated CMA-ES (Section 2.3) and examine the effect of the aggregation sizes on the performance (Figure 2). We divided the results of 50 trials, e.g., we have 25 dots for aggregation size 2, and 10 dots for 5. The aggregated CMA-ES provides higher Γ than the aggregated version of GA when we choose the aggregation size larger than 10. The best Γ was 0.612 by GA and 0.630 by CMA-ES when the aggregation size was 50. Finally, as a supplemental experiment, we run GA for 10,000 generations with aggregation size 1 and population size 30. The obtained Γ was 0.617. There is a small improvement compared to 0.612 of the aggregated GA result with 150 generations, but it was smaller than 0.630 obtained by aggregated CMA-ES with 100 generations.

Table 3 shows the model parameter values optimized by GA and CMA-ES. Aggregation size, population size, and number of generations were 50, 100, and 150 for GA, and 50, 50, and 100 for CMA-ES, respectively. Compared to the Nelder-Mead method [2, 17], GA yielded no improvement but CMA-ES achieved 2.0% relative improvement.

6. CONCLUSION

We have applied CMA-ES to a neuron model optimization problem for a public dataset recorded from rat neocortex, and have demonstrated that its averaged performance outperforms conventional optimization approaches based on GA and Nelder-Mead methods. However, we have found that the fitness obtained by CMA-ES largely depends on the initial condition, and sometimes it gives worse results than the conventional methods. To address the problem, we tried a simple strategy that makes the aggregation of CMA-ES for the optimization, which we referred to as aggregated CMA-ES. Experimental results show that aggregated CMA-ES is robust against the initial condition, and stably achieves better performance than conventional methods. Future work is to apply the proposed method to more complex neuron models and to extend it for a better performance with smaller computational cost.

7. REFERENCES

[1] C.C Mcintyre and T.J. Foutz, "Computational modeling of deep brain stimulation," in *Handbook of clinical neurology*, vol. 116, pp. 55–61. Elsevier, 2013.

[2] W. Gerstner and R. Naud, "How good are neuron models?," *Science*, vol. 326, no. 5951, pp. 379–380, 2009.

[3] R. Migliore, C.A. Lupascu, L.L. Bologna, A. Romani, J-D. Courcol, and S. et al. Antonel, "The physiological variability of channel density in hippocampal ca1 pyramidal cells and interneurons explored using a unified data-driven modeling workflow," *PLoS computational biology*, vol. 14, no. 9, e1006423, 2018.

[4] S. Mensi, R. Naud, and W. Gerstner, "From stochastic nonlinear integrate-and-fire to generalized linear models," in *Proc. Neural Information Processing Systems (NIPS)*, pp. 1377–1385. 2011.

[5] R. Kobayashi and K. Kitano, "Impact of network topology on inference of synaptic connectivity from multi-neuronal spike data simulated by a large-scale cortical network model," *Journal of Computational Neuroscience*, vol. 35, no. 1, pp. 109–124, 2013.

[6] M.C.Vanier and J.M. Bower, "A comparative survey of automated parameter-search methods for compartmental neural models," *Journal of Computational Neuroscience*, vol. 7, no. 2, pp. 149–171, Sep 1999.

[7] S. Druckmann, Y. Banitt, A. Gidon, F. Schürmann, H. Markram, and I. Segev, "A novel multiple objective optimization framework for constraining conductance-based neuron models by experimental data," *Frontiers in Neuroscience*, vol. 1, pp. 1, 2007.

[8] C. Rossant, D.F. Goodman, B. Fontaine, J. Platkiewicz, A. Magnusson, and R. Brette, "Fitting neuron models to spike trains," *Frontiers in Neuroscience*, vol. 5, pp. 9, 2011.

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

[10] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[11] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009," in *Proc. the 12th annual conference companion on Genetic and evolutionary computation (GECCO)*, 2010, pp. 1689–1696.

[12] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 949–980, 2014.

[13] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi, "Bidirectional relation between CMA evolution strategies and natural evolution strategies," in *Proc. Parallel Problem Solving from Nature (PPSN)*, 2010, pp. 154–163.

[14] Takafumi Moriya, Tomohiro Tanaka, Takahiro Shinozaki, Shinji Watanabe, and Kevin Duh, "Evolution-strategy-based automation of system development for high-performance speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 77–88, 2019.

[15] R.S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Neural Information Processing Systems (NIPS)*, pp. 1057–1063. 1999.

[16] V. Mnih, Adria P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. The 33rd International Conference on Machine Learning (ICML)*, pp. 1928–1937. 2016.

[17] R. Kobayashi, Y. Tsubo, and S. Shinomoto, "Made-to-order spiking neuron model equipped with a multi-timescale adaptive threshold," *Frontiers in Computational Neuroscience*, vol. 3,9, 2009.

[18] W. Gerstner, R. Naud, M. Carandini, L.C. Sincich, J.C. Horton, D.L. Adams, and J.R. Economides, "Quantitative single-neuron modeling competition for year 2009, developed by researchers at epfl; data from wistar rat cortical neurons and rhesus monkey lateral geniculate nucleus and retinal ganglion cells," *CRCNS.org.*, 2009.

[19] F. Fortin, F. De Rainville, M. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

[20] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[21] R. Jolivet, R. Kobayashi, A. Rauch, R. Naud, S. Shinomoto, and W. Gerstner, "A benchmark test for a quantitative assessment of simple neuron models," *Journal of neuroscience methods*, vol. 169, no. 2, pp. 417–424, 2008.