Supplementary for Multi-Objective Evolutionary Neural Architecture Search for Liquid State Machine

Sida Xin[1], Renzhi Chen[2], Xun Xiao[3], Yuan Li[3], Lei Wang[1]✉

I. PRELIMINARIES

A. Multi-objective ENAS

Neural architecture search based on multi-objective evolutionary algorithms aims to optimize multiple performance metrics of network architectures simultaneously, such as accuracy and network scale. By simulating the process of natural selection for iterative evolution, this method generates various network architecture candidates, evaluates and selects them based on predetermined multi-objective optimization criteria. Generally, there exists a trade-off between these objectives, meaning that improving one objective may lead to a decrease in performance for another objective. Therefore, the purpose of multi-objective optimization is to find a set of solutions that are considered optimal when all objectives are taken into account. This set of solutions is known as the Pareto optimal set, and the boundary formed by these solutions in the objective space is called the Pareto front [1]. No other solutions can dominate the solutions on the Pareto front. The concept of domination is explained as follows, for any two solutions a and b, solution a dominates solution b under the following conditions.

$$\forall i, f_i(a) \leq f_i(b) \wedge \exists j, f_j(a) < f_j(b) \tag{1}$$

$f_i(\cdot)$ represents the i^{th} optimization objective function, which typically includes accuracy or network scale. By identifying these Pareto optimal solutions, the algorithm gradually finds the optimal or near-optimal neural network architectures that satisfy multi-objective requirements.

We use the Hypervolume (HV) Indicator to quantify the quality of the Pareto front. This metric measures the multidimensional volume enclosed by the solution set in the objective space, which extends from the points in the solution set to a reference point. As the reference point, we select the worst point in the objective space, known as the Nadir Point. A larger HV value indicates better convergence and overall quality of the solution set. The formula for calculating the HV is as follows.

$$HV(S) = \text{Volume}\left(\bigcup_{x \in S} \left[f_1(x), f_1^{\text{ref}}\right] \times \ldots \times \left[f_m(x), f_m^{\text{ref}}\right]\right) \tag{2}$$

[1]National Innovation Institute of Defense Technology, Academy of Military Sciences, Beijing, China sd9.xin@gmail.com, leiwang@nudt.edu.cn

[2]QiYuan Lab, Beijing, China chenrenzhi1989@qiyuanlab.com

[3]College of Computer, National University of Defence Technology, Changsha, China {xiaoxun520, liyuan22}@nudt.edu.cn

S is the solution set, which contains multiple solutions. $f_i(x)$ is the value of the i^{th} objective function for solution x. f_i^{ref} is the reference point value for the i^{th} objective, typically chosen as the worst possible value for that objective. Volume represents the Hypervolume in multidimensional space, formed by the intervals between the solution points and the reference points.

B. Upper Confidence Bound (UCB)

The UCB algorithm based on standard deviation is an effective strategy for solving the multi-armed bandit problem [2], aimed at balancing exploration and exploitation. The core concept of UCB revolves around directing the selection of actions by considering both the uncertainty of rewards associated with each action and the average rewards derived from all actions. The following formula shows how the UCB value for each action is calculated.

$$UCB = \bar{x}_j + c \cdot \sigma_j \tag{3}$$

Here, $\bar{x}_j$ represents the average reward of action j, σ_j represents the standard deviation of the reward for action j, and c is a positive parameter controlling the intensity of exploration. A larger c value tends to encourage more exploration, while a smaller c value emphasizes exploiting the known best action.

In each iteration, the algorithm returns the action with the highest UCB value. As experiments proceed, the average rewards and standard deviations for each action will gradually stabilize, and the algorithm's selections will increasingly lean towards the truly optimal action.

II. RESULTS AND ANALYSIS

A. The Pareto Fronts with Different k

Fig. 1 shows the Pareto fronts obtained under different k after 800 function evaluations.

B. Pareto Solutions and Network Architecture

Table I presents the network structures corresponding to different Pareto solutions after 800 function evaluations with $k = 1$.

REFERENCES

[1] Giagkiozis I, Fleming PJ. Pareto front estimation for decision making. Evolutionary computation. 2014 Dec 1;22(4):651-78.

[2] Kuleshov V, Precup D. Algorithms for multi-armed bandit problems. arXiv preprint arXiv:1402.6028. 2014 Feb 25.

[3] Tian S, Qu L, Wang L, Hu K, Li N, Xu W. A neural architecture search based framework for liquid state machine design. Neurocomputing. 2021 Jul 5;443:174-82.

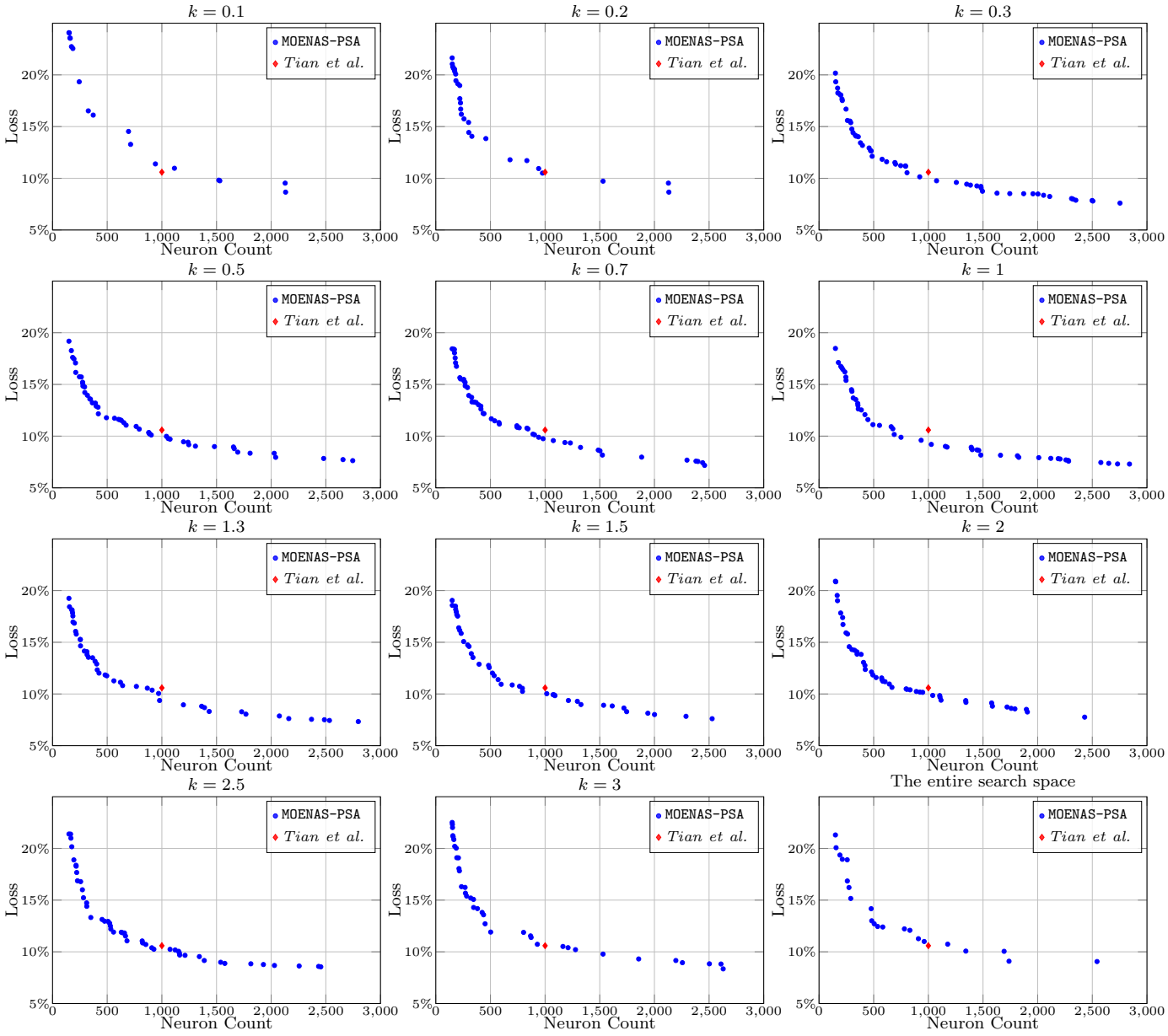

Fig. 1. The Pareto fronts obtained under different k after 800 function evaluations, including the unreduced search space.

TABLE I

THE NETWORK STRUCTURES CORRESPONDING TO DIFFERENT PARETO SOLUTIONS AFTER 800 FUNCTION EVALUATIONS WITH $k = 1$. ALL LIQUID STATE MACHINE MODELS ARE SINGLE-LAYER WITH THREE RESERVOIRS, WITH EACH ROW REPRESENTING THE PARAMETERS OF ONE RESERVOIR.

			Loss	**Neuron Count**	**Network Architecture**					
					N	$Ratio$	$P_{E\to E}$	$P_{E\to I}$	$P_{I\to E}$	$P_{I\to I}$
Tian et al. [3]			10.6%	1000	75	0.9	0.5	0.5	0.5	0.1
					505	0.8	0.4	0.3	0.4	0.0
					420	0.8	0.2	0.4	0.3	0.0
MOENAS-PSA	Extreme Points	N_E	18.5%	**150**	50	0.89	0.0015	0.83	0.67	0.90
					50	0.81	0.0023	0.35	0.35	0.00
					50	0.90	0.0004	0.65	0.89	0.00
		L_E	**7.3%**	2840	924	0.88	0.0134	0.76	0.54	0.81
					991	0.78	0.0006	0.38	0.37	0.00
					925	0.90	0.0164	0.61	0.71	0.00
	Trade-off Points	L_T	9.2%	1027	419	0.88	0.060	0.84	0.54	0.22
					319	0.78	0.032	0.39	0.37	0.00
					289	0.90	0.002	0.61	0.77	0.00
		N_T	10.7%	675	51	0.89	0.00517	0.83	0.69	0.90
					336	0.81	0.00004	0.35	0.35	0.00
					288	0.90	0.00130	0.65	0.86	0.00
	Knee Point	K	9.9%	750	183	0.88	0.17	0.84	0.54	0.22
					331	0.78	0.03	0.42	0.37	0.00
					236	0.83	0.04	0.31	0.77	0.00