Self-Adaptive Robustness of Applied Neural-Network-Soups

Maximilian Zorn[1], Philipp Altmann, Gerhard Stenzel, Michael Kölle,
Claudia Linnhoff-Popien, Thomas Gabor
LMU Munich,
[1]maximilian.zorn@ifi.lmu.de

Abstract

We consider the dynamics of artificial chemistry systems consisting of small, interacting neural-network particles. Although recent explorations into properties of such systems have shown interesting phenomena, like self-replication tendencies, social interplay, and the ability for multi-objective applications, most of these settings are reasoned about in the abstract weight space. We extend this setup to involve an applied, stateful positioning task with mutual dependencies and show that stable configurations can be found jointly in both the weight space and 3D space. We show that the main contributing factor is enabling the networks to self-adapt their interaction rates depending on their internal stability or their ability to position themselves correctly. We find that this method effectively prepares the network assembly against potentially destabilizing interactions, promoting emergent stability while preventing convergence to trivial states.

Introduction

Recent explorations of neural network 'soups' – inspired by emergent behaviors in artificial systems – are often comprised of multiple interacting neural network agents. While this specific setting originates from conceptual work that proposed neural networks with the capability to process their own weights as inputs (*network quines*, Chang and Lipson, 2018), research on this topic is situated within the broader context of artificial chemistries, where prior studies (e.g., Gabor et al., 2019) have established systems based on neural networks acting as particles and now are being further built upon (Illium et al., 2022; Randazzo et al., 2021; Zorn et al., 2023).

The goal in these systems is to observe the emergence of stable particle organization, sometimes with the additional requirement of learning or executing a given secondary task. We argue that utilizing the concept of neural-network chemistries poses a valid alternative to more conventional approaches to such dynamic organization problems; *swarms*, for instance, are suitable for such tasks and might (also) utilize emergent effects but are often in need of a centralized source of information, like video images, sound signals, or GPS (Jin et al., 2018; Vanhie-Van Gerwen et al., 2021; Cofta et al., 2020). On the other end of the autonomous behavior spectrum, *multi-agent systems* (including techniques like multi-agent reinforcement learning) often require communication protocols (Phan et al., 2022) or precisely crafted reward functions to enable emergent behaviors (Hahn et al., 2020; Ritz et al., 2021). Artificial chemistries can be placed somewhere in between these extremes, interacting (often) with purely randomized interactions without assumptions of locality or non-uniformity. The central aspect of randomness in the agent interactions has two major benefits: It alleviates the requirement of designing interaction curricula or protocols, while also ensuring that the learning progress is never too dependent on single individuals or agent combinations. This independence naturally offers robustness against agent failures or systematic interruptions was shown in Gabor et al. (2021).

On the other hand, with the randomness of unstructured interactions the danger of destabilizing actions or events that collapse or destroy the whole system presents an important, safety critical issue for any application in practical multi-agent tasks. Gabor et al., 2022, 2021 has shown that systems eventually organize themselves to stable configurations, as long as the stabilizing actions (or influences on the particles in general) outweigh any destabilizing events. However, as long as agents continuously update their network weights (e.g., to optimize for a secondary task) further stabilization actions are usually needed, which in turn may impact the loss of the secondary task and will repeat the update iteration. Unless the system is forced or incentivised to rest, stable convergence will not be possible.

To counteract these non-stationary effects and explore potential stabilizing mechanisms, we introduced a self-adaptive adjustment of interaction rates based on real-time performance feedback. This mechanism adjusts the rate of each action in response to observed loss, allowing a dynamically re-set to sustainable action rates in accordance with the evolving system needs.

In past work, soup interactions and tasks have been situated on the weight-space level. To illustrate the problem of an practical secondary task as it could occur in reality, we model a dynamic, three-dimensional setting with state-

ful particles that are tasked with positioning themselves in a predefined pattern, similar to the self-organized control of e.g., a robot drone swarm. Each network particle in our soup can perform one of three actions per time step: a destabilization action, a stabilization action (both in the weight space), and a repositioning interaction (in the 3D space), with the latter two being trainable to minimize their respective loss criteria, thereby influencing the collective behavior pattern of the system. We design the position criteria in a way that will always require the agent to re-adjust it's weights as long as other agents in the system move, thus introducing a co-dependent non-stationarity that we then show to be stabilized by the adaptive interation method. In this work we show a practical 3D-positioning task, both for simplicity and for visual demonstration, but we argue that the concept of self-adaptive stability is useful for any kind of non-convergent multi agent system and merits further research integrating concepts from artificial chemistry into autonomous systems learning tasks.

Our findings suggest that the proposed self-adaptive approach, by modulating interaction intensity based on immediate past outcomes, can significantly enhance the robustness and resilience of neural network soups against disruptions, allowing for interesting insights into the management of emergent behaviors in decentralized artificial systems.

We summarize our contribution as follows:

- We adopt mostly abstract concepts of self-replicating neural network soups and apply them to a stateful positioning task in three-dimensional space. We propose an extension to the soup setting of Gabor et al. (2019) based on this positioning criterion and can adjust their technique for training self-replicating neural networks to fit our (different) network dimensionalities.

- We show that by introducing self-adaptive adjustments to their action and interaction rates – depending on their internal and external stability – we can find jointly stable configurations, both in the weight-space representations of the neural networks, as well as in the 3D space where they have to position themselves in relation to their peers.

The remainder of this work is structured as follows: We briefly explore related concepts and give a short overview of related work and applications. We then define our setting and summarize the background, i.e., the fundamental building blocks, as we build our artificial chemistry soup in reference to the work of Gabor et al. (2019). With this formalization in place, we explore and discuss our experimental evaluation before we finally conclude our findings.

Related Work

Swarm Control For improved robustness and scalability, swarms (e.g., unmanned aerial vehicles (UAV)) are often considered in favor of single entities. However, while central control entities (e.g., Saha and Isto, 2006; Kazanzides and Thienphrapa, 2008) can maintain safety in smaller distributed systems (like small particle swarms), such control is often not practical in a centralized manner when scaled to larger entity sizes or intricate control objectives (cf. Pantelimon et al., 2019). Yet, decentralized control (e.g., Borrelli et al., 2004; Tanner and Christodoulakis, 2007) requires additional communication protocols. Furthermore, safety might be affected by malicious adversarial entities or the potential failure of said protocols (Pantelimon et al., 2019). Therefore, means of robustness need to be built into the algorithms being applied. We approach the decentralized control of autonomous particles by randomizing interactions akin to classical artificial chemistry settings, i.e., without structured communication, signal processing, or similar deterministic requirements. Robustness is then an emergent effect of actions being applied (with certain probabilities) or interactions affecting a small subset of random particles in the soup.

Self-Adaptation Considering that algorithmic parameters of complex dynamic systems themselves are often subject to their own optimization, the question of how to choose those *hyperparameters* is largely considered to be a field of research in itself (cf. Feurer and Hutter, 2019). One such approach, often found in relation to 'natural computing' and machine learning in general, is the concept of *self-adaptation*, i.e., automatic or heuristic approximation of those parameters, depending on the live performance of an algorithm. Applying self-adaptation has been shown to improve the system's overall robustness and convergence properties. Examples can readily be found in the field of evolutionary computing (e.g., Meyer-Nieberg and Beyer, 2007; Saravanan et al., 1995; Boukhari et al., 2018), where early self-adaptivity was applied to, e.g., evolutionary mutation rates and recombination rates. Multi-objective optimization (Büche et al., 2003) and neural network learning hyperparameters (Wijayakulasooriya et al., 2002; Li et al., 2015) are also often subject to self-adaptive methods, as exact (correct) configurations often require costly parameter tuning or detailed domain knowledge of the problem at hand.

AI Safety Finally, when dealing with the question of *why* alternative approaches to such classical tasks (like positioning in space) should be considered, recent literature on machine-learning systems also (by obligation) likes to point towards aspects of AI Safety, i.e., the 'secure' application of such intelligent agents. As 'safety' is usually tricky to define or scope, much of the discussion is centered on what ML agents should *not* do. Some approaches consider safety as a secondary objective to be fulfilled while invisible to the agent and only used for external evaluation (Leike et al., 2017; Amodei et al., 2016; Raji and Dobbe, 2023;

Hendrycks et al., 2021). According to Amodei et al. (2016), an agent must fulfill one or, ideally, multiple of the following criteria to be considered safe:

- **Avoiding Side Effects** Reward functions should be selected in such a way that no undesirable side effects occur. This also applies to network soups interacting with each other and, naturally, undesired interactions should not completely diverge or collapse a group.
- **Self-Modification** An agent should also function in environments in which it can adapt itself. This property is directly translatable to our setting of self-replicating, i.e., weight-modifying networks, and the aspects of gaining stability (or, rather, avoiding instability).
- **Distributional Shift** Even if the test environment differs from the initial environment, an agent should act robustly. Considering all networks consistently change themselves and pose change to others, this also naturally includes aspects of artificial chemistries.
- **Robustness to Adversaries** How can an agent recognize adversaries and deal with their influence safely? While some of the interactions may be considered *adversarial*, only very specific cases include purposefully harmful interactions. However, in many cases, unstable compositions of network states or unfortunate sequences of interaction events may have similarly dangerous effects, which we may want to stabilize against.

The list of requirements by Amodei et al. (2016) includes further aspects, like safe exploration behavior, that are not directly relevant here. In this work, the safety of the agent is indirectly considered in relation to distributional shifts and robustness against harmful interactions.

Methodology

Preliminiaries

In this work, we consider particle *soups*, where one is able to observe interactions by building a population of m mutually interacting network particles $\mathcal{M}_1, \dots, \mathcal{M}_m$. Different combinations and parameterizations of such individuals and their various (inter-)actions produce emergent behavior comparable to an artificial chemistry system (cf. Dittrich et al., 2001) or artificial life system (cf. Adami and Brown, 1994; Ofria and Brown, 1998). This means a soup evolves over a fixed amount of epochs, i.e., evolution steps. At every epoch, different (inter-)action operators can be applied to network particles in the population with a certain chance, resulting in new particles and, thus, a changed soup. The general setting follows that of Gabor et al. (2021), Randazzo et al. (2021), and Gabor et al. (2022). For the sake of completeness, we briefly recap the material related to particle soups introduced there.

Weightwise Application

As has been introduced in recent literature, *self-replicating neural network* can be defined by reasoning about network weights and their repeated reproduction with the notion of *weightwise application* (Gabor et al., 2022). It allows us to apply a neural network $\mathcal{N}$ as input to another neural network $\mathcal{M}$, by applying each encoded weight in $\mathcal{N}$ to $\mathcal{M}$ once and re-using the predicted weight value as the 'replicated' weight. As will be shown in this work, the self-replication property is key to stability in applied artificial network systems.

We also assume that every neural network $\mathcal{M}$ in our soup adheres to the same dense-layered architecture (Goodfellow et al., 2016) and can thus be fully described by giving its weights $\overline{\mathcal{M}} \in \mathbb{R}^{26}$. Following the network's architecture, we also write the vector of weights $\overline{\mathcal{M}} = \langle v_{l,c,p} \rangle_{l,c,p}$ structured according to the layers, cells, and cell connections of the neural network so that each $v_{l,c,p} \in \mathbb{R}$ is the network weight assigned in layer l for neuron c to the connection to neuron p in the previous layer.

We can now define the application of one network to another as follows:

Definition 1 (application)**.** *Given neural networks $\mathcal{M}, \mathcal{N}$. Let $\mathcal{O} = \mathcal{M} \triangleleft \mathcal{N}$ be the application of $\mathcal{M}$ to $\mathcal{N}$ given by*

$$\overline{\mathcal{O}}_{l,c,p} = \mathcal{M}\big(l, c, p, \overline{\mathcal{N}}_{l,c,p}, 0, 0, \dots\big). \tag{1}$$

This notation follows the idea of encoding multiple tasks within the input vector by masking out unrelated information of other tasks with 0, as introduced by Gabor et al. (2021). The amount of masked information $0, 0, \dots$ depends on other tasks themselves but can be considered fixed for later experiments. Since application alone does not produce stable (non-trivial) replicator networks, we need to further introduce the necessary training procedure, as well as the 'artificial chemistry' interactions as outlined by Gabor et al. (2019, 2021) and Zorn et al. (2023).

Self-Replication Training

Based on the application $\triangleleft$, we can define self-application as follows:

Definition 2 (self-application)**.** *Given a neural network $\mathcal{M}$. We call the neural network $\mathcal{M}' = \mathcal{M} \triangleleft \mathcal{M}$ the self-application of $\mathcal{M}$.*

In theory, the application $\triangleleft$ has fixed points, i.e., there exist $\mathcal{M}$ so that $\mathcal{M} \triangleleft \mathcal{M} = \mathcal{M}$. In a similar way to other artificial chemistry setups (cf. Fontana and Buss, 1994), these play a special role (Gabor et al., 2019). However, since our particles are living in a continuous space, related work uses a relaxed notion called ε*-fixpoints*:

Definition 3 (ε-fixpoint, self-replication (SR))**.** *Given a neural network $\mathcal{M}$. Let $\varepsilon \in \mathbb{R}$ be the error margin for the*

fixpoint property.[1] *We call $\mathcal{M}$ an ε-fixpoint iff for all l, c, p $\left|\overline{\mathcal{M}'}_{l,c,p} - \overline{\mathcal{M}}_{l,c,p}\right| < \varepsilon$ where $\mathcal{M}' = \mathcal{M} \lhd \mathcal{M}$. We also say that $\mathcal{M}$ is able to self-replicate.*

The notion of self-replication in our work is strongly influenced by Dawkins (2016) in that it focuses on the ability to simply copy information. We take a first step towards observing other properties associated with self-replication (like metabolism, e.g.) by expanding a particle's feedback loop for self-sufficient development through its peers. Such stabilization in uncertain systems may also be helpful for simulating and enforcing interactions between inherently stochastic processes and systems with inherent fault potentials or interruption potentials, like multi-agent reinforcement learning or signal communication problems.

Soup Interactions

With networks interacting together in the context of a soup, Gabor et al. (2019) define the interaction of applying and substituting a randomly drawn partner particle to the predicting network as an *attack*. This interaction is given as:

Interaction 1 (attack (`ATK`))**.** *Applied to two random networks $\mathcal{M}, \mathcal{N}$ drawn from the soup, attacking substitutes the weights of the attacked network $\mathcal{M}$ with the weights given via $\mathcal{M}' = \mathcal{N} \lhd \mathcal{M}$.*

This setting – self-replication and attacks as formalized here – can be considered essentially in line with recent related work on such artificial soups (cf. Gabor et al., 2022; Zorn et al., 2023). For this work, we further augment the setup with an additional interaction representing stateful, conditional *repositioning* of network particles in 3D space. We assume therefore that network particles are now additionally created with an initial, randomly sampled position vector $\overrightarrow{p} := \langle x, y, z \rangle$ representing the state position in the x, y and z coordinate. Borrowing the multiple-goal approach for encoding inputs of different tasks from the setup of Gabor et al. (2021), we now let networks interact with each other in the 3D-state position spaces (besides the weight space), enabled by a new interaction which we formalize as follows:

Interaction 2 (reposition (`POS`))**.** *Applied to one sampled random network $\mathcal{M}$ with n networks $\mathcal{N}_1, \ldots, \mathcal{N}_n$ as input, drawn randomly from the soup of m available particles (without replacement). The `POS` action trains the predicting network $\mathcal{M}$ to update its position (state), given as $\overrightarrow{p}_{\mathcal{M}} := \langle x, y, z \rangle \in \mathbb{R}^3$, towards a positioning criterion C. We assume that C is given and appropriate for the task in question. The predicted update is thus given as application on such network $\mathcal{M} : \mathbb{R}^{4+n\cdot 3} \to \mathbb{R}^3$ via:*

$$\hat{p}_{\mathcal{M}} = \langle 0, 0, 0, 0, \overrightarrow{p}_1, \ldots, \overrightarrow{p}_n \rangle, \tag{2}$$

where $\overrightarrow{p}_j := \langle x, y, z \rangle \in \mathbb{R}^3$ for particles $j \in [1, \ldots, n]$. Each of the three outputs corresponds to the positioning update of the x, y and z coordinate in 3D-space respectively. To steer the resulting positioning effect towards a desired pattern or design, the given prediction should minimize loss D, which we compute as the distance of the updated position in 3D space towards the positioning criterion C, i.e.,

$$D = \left|\left| (\overrightarrow{p}_{\mathcal{M}} + \hat{p}_{\mathcal{M}}) - C(\overrightarrow{p}_1, \ldots, \overrightarrow{p}_n) \right|\right|, \tag{3}$$

where $||\_||$ is an appropriate vector norm (for a 3D space in our case) and $+$ is the vector addition.

In this formalization, we condition the criterion on the input particles $\overrightarrow{p}_1, \ldots, \overrightarrow{p}_n$, however generally, the positioning criterion can be freely chosen. In essence, we model the repositioning task as a local n-particle interaction, where the predicting particle $\mathcal{M}$ should position itself in the geometric center of the randomly chosen n particles' positions. The proposed pattern, resulting eventually in (fairly) equally spaced particles, was chosen here for simplicity and visual demonstration purposes. Other designs might include positioning the particles maximally far apart in e.g., spherical alignment, grid like structures or in patterns robust to collisions (cf. Cofta et al., 2020).

More formally, the position update $\hat{p}_{\mathcal{M}}$ should minimize D towards the center of mass (midpoint) of a group of n particles with positions $\overrightarrow{p}_1, \ldots, \overrightarrow{p}_n$:

$$C(\overrightarrow{p}_1, \ldots, \overrightarrow{p}_n) = \left\langle \frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}, \frac{\sum_{i=1}^{n} z_i}{n} \right\rangle. \tag{4}$$

Finally, following Zorn et al. (2023), we assume a function id which assigns to every neural network within our soup a unique identifier, e.g., $\mathrm{id}(\mathcal{M}) \in [0; 1]$ to represent said network $\mathcal{M}$. We treat these real-numbered values as IDs for the respective networks without them implying any structure or ordering. The intended purpose is to test whether the additional identification input of soup interaction networks is similarly helpful for stateful tasks like repositioning like it was with weight-space tasks shown by Zorn et al. (2023).

Experiments using this identification will extend the task encoding for input particles $\overrightarrow{p}_1, \ldots, \overrightarrow{p}_n$ in Interaction 2 with their respective $\mathrm{id}(\mathcal{N}_j), j \in [1, \ldots, n]$, and the predicted update changes to an application on a network $\mathcal{M}$: $\mathbb{R}^{4+n\cdot 4} \to \mathbb{R}^3$ given as:

$$\hat{p}_{\mathcal{M}} = \langle 0, 0, 0, 0, \mathrm{id}(\mathcal{N}_1), \overrightarrow{p}_1, \ldots, \mathrm{id}(\mathcal{N}_n), \overrightarrow{p}_n \rangle. \tag{5}$$

We use the terms action *rate* and *chance* interchangeably to mean the application of actions `SR`, `ATK`, `POS` to a prediction candidate network $\mathcal{M}$, with n other networks $N_{1\ldots n}$ drawn randomly from the soup without replacement (including $\mathcal{M}$). We formally abbreviate the chance of these actions occurring at evolution step t with $\mathtt{SR}^*$, $\mathtt{ATK}^*$, $\mathtt{POS}^*$, where

[1] For this paper, we assume $\varepsilon = 10^{-5}$.

we iterate over all m particles in the soup, and **either** apply the first of the actions SR, ATK, POS where $\texttt{action}^* \leq r$ of a random sample $r \sim \mathcal{N}(0,1) \in [0;1)$ drawn from a random normal distribution **or** skip over the particle.

Changes to Gabor et al. (2022) Where ATK applies one network to another, a $1:1$ application, in the case of the POS interaction an update is realized with the information of multiple networks as input in the form of a $n:1$ application, which is novel concept in this context. Furthermore, for this repositioning to work as formalized, each neural network $\mathcal{M}$ we discuss needs 3 real-numbered outputs to be able to represent the 3-dimensional position update function. This deviates from the original concept of weightwise application, where previously Gabor et al. (2022) only utilized networks of the form $\mathcal{M} : \mathbb{R}^{(\cdot)} \to \mathbb{R}^1$ to signify the mapping of $(\cdot)$ inputs to a single output, as the prediction of the currently feed-forwarded weight encoding. However, by simply reducing the network's output-dimension from $\mathbb{R}^3 \to \mathbb{R}^1$ via an appropriate reduction function (sum, mean, etc.), we find that using $\mathbb{R}^3$ as our output dimension works functionally the same for the weightwise application, while also opening the possibility of extending neural network soups to work with multi-dimensional output, like we do with our position update. We found the sum to work best as reduction function for our purpose, and we consider any mention of self-replication or SR-interactions in the form of $\mathbb{R}^{(\cdot)} \to \mathbb{R}^1$ to be the abbreviation of $\mathbb{R}^{(\cdot)} \to \mathbb{R}^1 := \sum(\mathbb{R}^3)$ for the remainder of this work.

Experiments

Given this formalization of artificial soups of neural networks, we begin our exploration with a simple system of up to $m = 10$ interacting particles, each of which is equipped with one of three possible actions (ATK, SR, POS). While ATK is a purely applicative action, SR and POS are both trainable, i.e., a network will update its weights to minimize the respective loss criteria as described in Sec. Methodology. Only one of SR or POS may occur at any one time step of the evolution for every network. Any weight training is computed via the Stochastic Gradient Descent (SGD) optimizer by the Python pytorch library (cf. Paszke et al., 2019) with a learning rate of 0.004 and a momentum parameter of 0.9. This setting is chosen per Gabor et al. (2019) to ensure comparable evolution, particularly concerning the self-replication aspect of the network soup.

For the first proof of concept, we show that positioning in space (here with 3 dimensions) is possible without direct collapse or divergence of particles, using only the POS-interactions. Fig.1a shows a visual representation of stateful particles $p_0, \dots p_m$, each starting from an initial position vector $\langle x, y, z \rangle_{0 \leq i \leq m}$, with $x, y, z \in [0;1] \subset \mathbb{R}$. As particles are not able to completely converge to a stable configuration, each randomized interaction between particles leads to interactive change in the system. Furthermore, since no 'resting' patterns are emerging in this setting, any disturbances can potentially affect all particles. The movement trails (last 50 positions per network) are drawn behind the particles, indicating the erratic behavior of such randomized interactions. This unstable group position is corresponding to the average distances D of all $m \times m$ particle relations at evolution step $t = 1000$, which is also visible as a varied, unsteady line plot in the initial ablation of effects in Fig 2 (bottom left, blue).

To show how fragile this balance is, we then introduce destabilization on the weights of the particles themselves, in our case by additionally including the interaction ATK to be executed by random particles (even with very low probability of $\texttt{ATK}^* = 0.001$ per particle per evolution step), and see drastic impact, first, on the weights themselves and, subsequently, on the resulting movement (as conditional effect, resulting from the network's now more imprecise forward passes). Fig. 3 shows such an exemplary collapse to a single weight set — a couple of early attacks are enough to diverge a soup collective without any form of explicit stability (training). The final state is similar to the trivial 0-fixpoint found by Gabor et al. (2019). In 3D space, this collapse similarly contracts all positions together or — in the case of weights diverging (i.e., growing towards $\pm\infty$) — the particles simply position themselves far outside the visual cube.

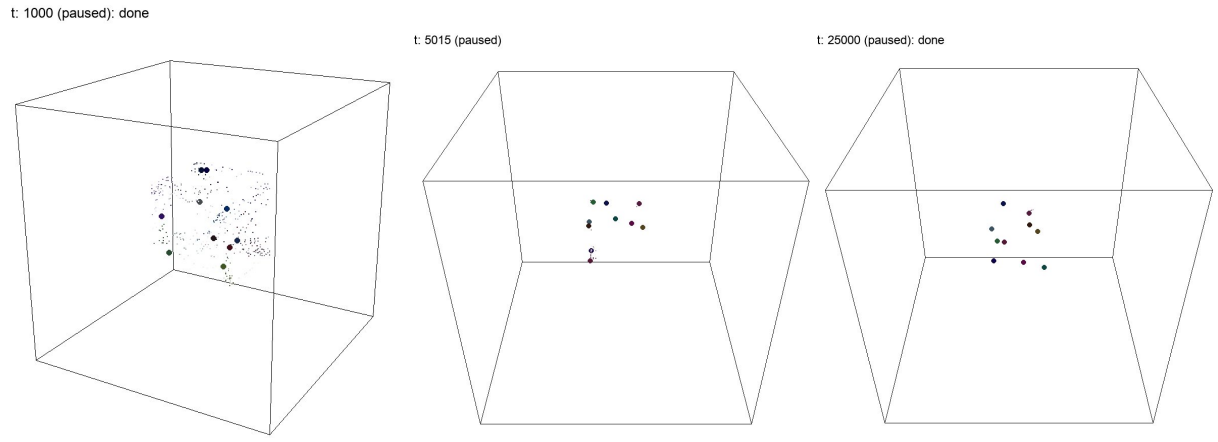
Consequently, we hypothesize that the inclusion of the SR action may prevent this destabilization to a certain degree, depending on the network's ability to self-replicate. This result is to be expected from similar experiments by Gabor et al. (2019, 2021) or Zorn et al. (2023), although we can now confirm that this stabilization effect is also helpful in 'applied' network realizations (i.e., the actual positioning in space, which also influences the movement of other particles), rather than simply the effect on individual weight sets, as was discussed in their previous works. We can indeed observe this stabilization to help with soups not diverging or collapsing, but it does not remedy the issue of interdependent movement shifts around the space. We, therefore, include the concept of self-adaptivity to automatically regulate the hyperparameters governing the action-intensity (i.e., $\texttt{SR}^*, \texttt{POS}^*$).

For our adaptation of self-adaptive interaction rates for the initial interaction rates $\texttt{action}^*_{old} \in [0;1)$ (for both SR and POS) we realize a simple, loss-conditioned rate update as follows:

$$\texttt{action}^*_{new} = \text{clip}(\texttt{action}^*_{old} + \text{loss}_t * (r - 0.5)),$$

where *clip* also constrains $\textit{rate}_{new} \in [0;1)$ and $r \sim \mathcal{N}(0,1) \in [0;1)$ is a dynamically sampled random value (drawn Gauss-uniformly random) at evolution step t.

Although this auto-regulation of hyperparameters now allows for conditional (re-)training of the interactions and self-regulates the required robustness against degrees of desta-



(a) (b) (c)

Figure 1: We plot the positioning of 10 individual network particles in 3D space, where every network is initialized with an initial, randomly sampled position vector $\overrightarrow{p} := \langle x, y, z \rangle$, with $x, y, z \in [0; 1) \subset \mathbb{R}$. All particles optimize their position w.r.t. a positioning criterion, here the geometric center of the n randomly sampled, distinct particles from the whole soup. Movement is indicated as the trail of dots of the last 50 positions $\overrightarrow{p}_{t-50}, \ldots, \overrightarrow{p}_{t-1}$ of each network particle.
We show the following configurations as: **(a)** One instance of a destabilized soup of erratically moving particles. **(b)** An initial, locally-optimal configuration of a different soup instance and the final configuration **(c)** after multiple (smaller) disturbances, resulting in a final ε-*fixpoint-soup* (cf. Gabor et al., 2019).

bilization (i.e., different rates of `ATK` interactions), to prevent hyperparameters prematurely converging (or entirely remaining) at values very close to 0.0 (and therefore very unlikely of ever being applied again), we do need to introduce two *meta*-hyperparameters $\beta, \gamma \in [0; 1)$ that effectively represent the probability of particles re-evaluating their abilities or state (i.e., checking self-replication ability with probability β or their positioning in space with probability γ) and re-adapt the hyperparamters should the loss have increased. In every evolution step, we employ this process via an exclusive decision, where a network that does *not* enact the interactions `SR` or `POS` has a respective β, γ chance of re-evaluating their abilities or state.

Fig. 2 shows this approach applied to our soup setting. In the evaluation over multiple runs, the self-adaptive systems (bottom right) show more stable and less variant distances (bottom right, blue). In this context, we have also tested the effect of including the id-function-identifier *with_id* for each input network in the application but found it to have no significant benefit to either the self-replication ability in the soup or the positioning in space — on the contrary, in some cases the variance was unexpectedly high over multiple runs (bottom right, orange). Since the inclusion of the identifiers showed promising results in their application in the social soups of Zorn et al. (2023), this result is interesting to note, but could also indicate the difference of applicability to stateful tasks (rather than the weight-space-specific ones). We did, therefore, not include the identifiers in the final experimental evaluation, showing the complete setup of self-adaptive robustness against increased rates.

The plots in Fig. 4 show one representative experiment run of $m = 10$ particles over 5000 evolution steps with the `SR` and `POS` interactions both enabled with self-adaptive rate-adjustments. Both the `POS`* rates (Fig. 4a) and the `SR`* rates (Fig. 4b) are shown to quickly adapt to repeated instances of `ATK` and, e.g., destabilized networks (red, orange, grey) adequately re-set their interaction rates to higher values. This result can also be seen visually, both in the weight space (Fig. 4c), where the three most affected networks take visibly longer (over time, i.e., cf. height of the z-axis) to converge to stable fixpoints.

This evolution to stable patterns in 3D space — which we observe as emerging 'checkpoints' of local optima, where particles iteratively come to rest (until further agitation) — can also be observed in the actual resting position (no trails, i.e., without movement over the last 50 steps) in 3d-space, first to the pattern seen in Fig. 1b and finally in the pattern of Fig. 1c, which is also a full group of 10 ε-self-replicators and fully stable against any further `ATK` interactions.

Conclusion

In our experiments, we took the primarily abstract concepts of self-replicating neural network soups originating from Gabor et al. (2019) and applied them to a stateful positioning task in a non-abstract (3D) space. We focus on the issue of finding jointly stable configurations, both in the weight-space representations of the neural networks, as well as in the 3D space where they have to position themselves in re-

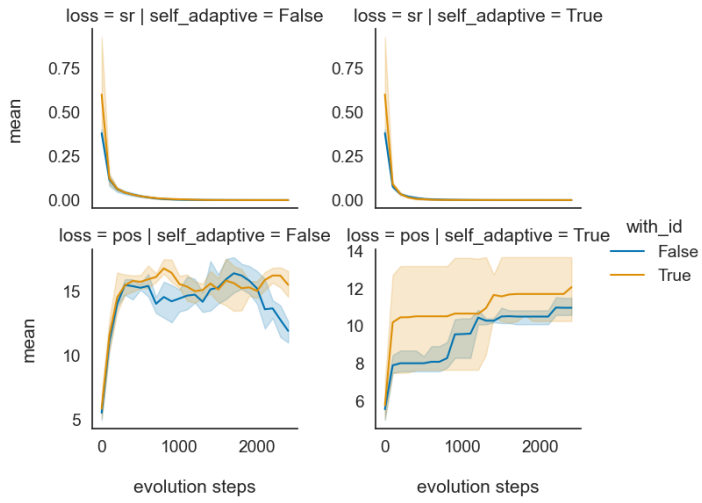


Figure 2: **(top row, y-axis)** Mean self-replication loss of all m particles in a soup of 10 networks. **(bottom row, y-axis)** Corresponding mean distances of all $m \times m$-particles at the respective evolution steps (2500 in total). We test whether the effect of including the id-function-identifier *with_id* (orange) or not (blue) is helpful for finding stable group positions (*loss=pos*) or for the self-replication (*loss=sr*), as well as depending on whether the self-adaptation is enabled **(right column)** or not **(left column)**. For this initial ablation, all parameters were optimistically set and show the mean and 95% confidence interval over three runs each; Initially $\texttt{SR}^*, \texttt{POS}^* = 0.1$, self-adaptive meta-rates were $\beta = 0.25\%, \gamma = 0.005\%$ probability per network per evolution step and $\texttt{ATK}^* = 0.0$, $n = 2$. Lower (mean) self-replication losses are better, and distance losses are better the more stable (and less variant) they are.

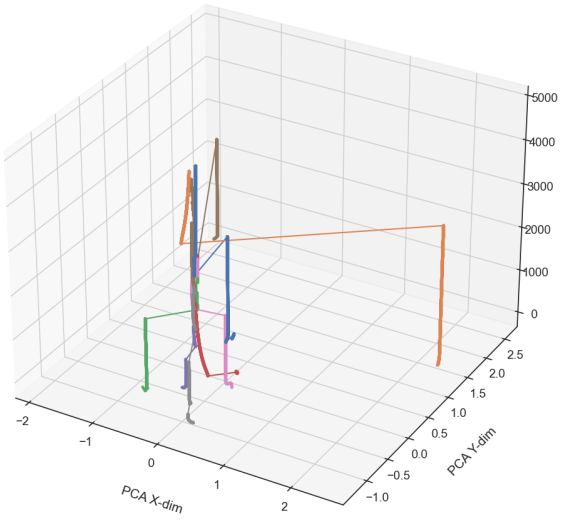


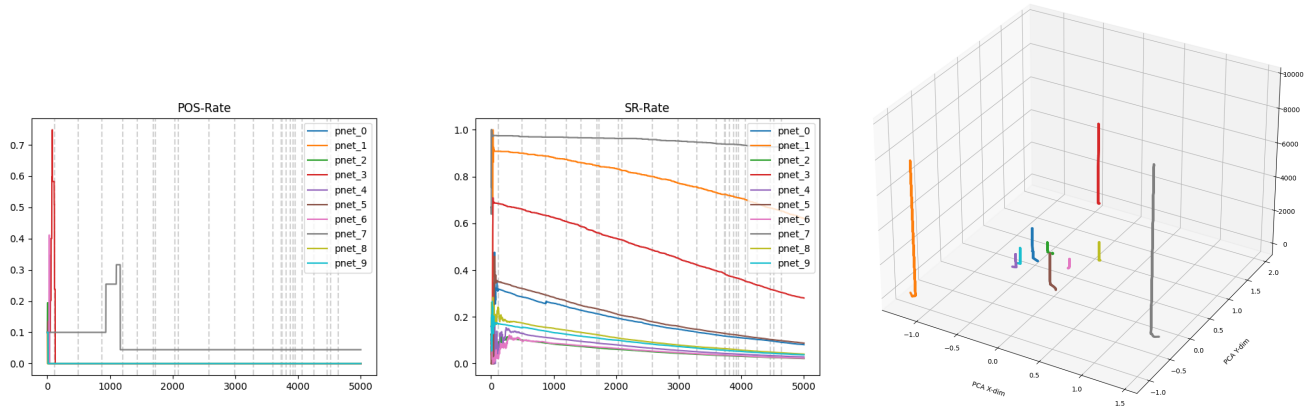
Figure 3: Network weights of a soup of 10 networks are depicted as two-dimensional weight space based on the transformed x- and y-axes derived via PCA dimensionality reduction. We plot the transformed PCA-X and PCA-Y in the horizontal plane and show the change of weights overtime on the z-axis (this is not the same 3D space as in the positioning task!). Even with very low probability of $\texttt{ATK}^* = 0.001$ and moderately low probability of moving via POS interaction with $\texttt{POS}^* = 0.1$ ($n = 2$) per particle per evolution step, all networks collapse to a singular point in the weight space due to randomly but repeatedly occurring ATK interactions on individual network weights.

lation to their peers. To achieve this, we have extended the soup setting of Gabor et al., 2021; Zorn et al., 2023 further by introducing network-dependent interactions and relaxing the initial weightwise application of self-replication to accommodate prediction dimensions > 1 for our purposes. To tackle the challenge of interdependent positioning, we employ the technique of self-adaptive adjustments of their action/interaction rates depending on the stability of their self-replication or their ability to position themselves correctly in 3D space. We have found that this method effectively shields the network assembly against destabilizing interactions and produces emergent stability in the form of optimal intermediary patterns.

In the future, we would like to explore how far this setting can be generalized. Given that weightwise replication seems to relax to output dimensions > 1 readily, it would be interesting to study how large of an output dimension the self-replication ability can still contest with. Increasing the network particles and their dimensionalities would also open many possible applications into even more applied (cooperative) machine-learning interactions and forms of ML robustness. Finally, since self-adaptiveness has been shown to work well individually, we will also test how foreign adaptation, i.e., regulation of other particles' rates in different degrees of stability, might affect collective soup robustness as a whole.

Acknowledgements

This work was funded by the Bavarian Ministry for Economic Affairs, Regional Development and Energy as part of a project to support the thematic development of the Institute for Cognitive Systems.



(a) (b) (c)

Figure 4: We plot the setting of 10 individual network particles with self-adaptive interaction rates for SR and POS. The respective meta-hyperparameters β, γ as chances of re-evaluating the self-replication abilities or the current state position are set at $\beta = 0.35\%, \gamma = 0.005\%$ per particle per evolution step. The interaction rates themselves are initially set to $\texttt{SR}^* = 1.0$ and $\texttt{POS}^* = 1.0$ (before self-adaptation). The destabilizing interaction ATK is set at $\texttt{ATK}^* = 0.001$ per particle per evolution step, instances of ATK are shown as gray dashed lines on the x-axis .
We show an exemplary run of action rate self-adaptation of $\texttt{POS}^*$ **(a, y-axis)** and $\texttt{SR}^*$ **(b, y-axis)** over 5000 the evolution steps **(a/b, x-axis)**. Of note here are three particles (red, gray, orange). Plot **(a)** shows all other particles quickly positioning and stabilizing themselves towards resting positions, as indicated by their almost immediate adaption of $\texttt{POS}^* \rightarrow 0.0$. Plot **(b)** shows the corresponding rates of $\texttt{SR}^*$, and the increased rates of the red, gray and orange particles due to destabilizing attacks, which in turns effect the positioning abilities leading to the multiple respective re-adaptations of $\texttt{POS}^*$ over the course of the soup's evolution. Both red and orange eventually find stable positions, with $\texttt{POS}^* \rightarrow 0.0$ and $\texttt{SR}^*$ (without need for stabilization) then steadily decreasing as intended. Only the gray particle is not able to find a stable position in this run, with ongoing $\texttt{POS}^* > 0.0$, hence never lowering the $\texttt{SR}^*$ rate either.
The same experiment is also depicted as a two-dimensional weight space based on the transformed X- and Y-axes derived via PCA dimensionality reduction in plot **(c)**. We plot the transformed PCA-X and PCA-Y in the horizontal plane and show the change of weights overtime on the z-axis. Depending on the rates of $\texttt{SR}^*$ and $\texttt{POS}^*$, particles have varying amounts of training steps where the restless particles (orange, gray, and red) can also be visually found to self-adapt to longer SR training to stabilize their self-replication and state position losses.

References

Adami, C. and Brown, C. T. (1994). Evolutionary learning in the 2d artificial life system 'avida'. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV*, pages 377–381. MIT Press, Cambridge, MA.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Borrelli, F., Keviczky, T., and Balas, G. J. (2004). Collision-free uav formation flight using decentralized optimization and invariant sets. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, volume 1, pages 1099–1104. IEEE.

Boukhari, N., Debbat, F., Monmarché, N., and Slimane, M. (2018). A study on self-adaptation in the evolutionary strategy algorithm. In *Computational Intelligence and Its Applications: 6th IFIP TC 5 International Conference, CIIA 2018, Oran, Algeria, May 8-10, 2018, Proceedings 6*, pages 150–160. Springer.

Büche, D., Müller, S., and Koumoutsakos, P. (2003). Self-adaptation for multi-objective evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization: Second International Conference, EMO 2003, Faro, Portugal, April 8–11, 2003. Proceedings 2*, pages 267–281. Springer.

Chang, O. and Lipson, H. (2018). Neural network quine. In *Artificial Life Conference Proceedings*. MIT Press.

Cofta, P., Ledziński, D., Śmigiel, S., and Gackowska, M. (2020). Cross-entropy as a metric for the robustness of drone swarms. *Entropy*, 22(6):597.

Dawkins, R. (2016). *The selfish gene*. Oxford university press.

Dittrich, P., Ziegler, J., and Banzhaf, W. (2001). Artificial chemistries—a review. *Artificial life*, 7(3):225–275.

Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. *Automated machine learning: Methods, systems, challenges*, pages 3–33.

Fontana, W. and Buss, L. W. (1994). What would be conserved if" the tape were played twice"? *Proceedings of the National Academy of Sciences*, 91(2):757–761.

Gabor, T., Illium, S., Mattausch, A., Belzner, L., and Linnhoff-Popien, C. (2019). Self-replication in neural networks. In *ALIFE 2019: The 2019 Conference on Artificial Life*, pages 424–431. MIT Press.

Gabor, T., Illium, S., Zorn, M., Lenta, C., Mattausch, A., Belzner, L., and Linnhoff-Popien, C. (2022). Self-Replication in Neural Networks. *Artificial Life*, pages 205–223.

Gabor, T., Illium, S., Zorn, M., and Linnhoff-Popien, C. (2021). Goals for self-replicating neural networks. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Hahn, C., Ritz, F., Wikidal, P., Phan, T., Gabor, T., and Linnhoff-Popien, C. (2020). Foraging swarms using multi-agent reinforcement learning. In *Artificial Life Conference Proceedings 32*, pages 333–340. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ....

Hendrycks, D., Carlini, N., Schulman, J., and Steinhardt, J. (2021). Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*.

Illium, S., Zorn, M., Lenta, C., Kölle, M., Linnhoff-Popien, C., and Gabor, T. (2022). Constructing organism networks from collaborative self-replicators. *arXiv preprint arXiv:2212.10078*.

Jin, Y.-H., Ko, K.-W., Lee, W.-H., et al. (2018). An indoor location-based positioning system using stereo vision with the drone camera. *Mobile Information Systems*, 2018.

Kazanzides, P. and Thienphrapa, P. (2008). Centralized processing and distributed i/o for robot control. In *2008 IEEE International Conference on Technologies for Practical Robot Applications*, pages 84–88. IEEE.

Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Orseau, L., and Legg, S. (2017). Ai safety gridworlds. *arXiv preprint arXiv:1711.09883*.

Li, X., Xiang, S., Zhu, P., and Wu, M. (2015). Establishing a dynamic self-adaptation learning algorithm of the bp neural network and its applications. *International Journal of Bifurcation and Chaos*, 25(14):1540030.

Meyer-Nieberg, S. and Beyer, H.-G. (2007). Self-adaptation in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*, pages 47–75. Springer.

Ofria, C. and Brown, C. (1998). The avida user's manual. In Adami (1998), The Avida software is publicly available at ftp.krl.caltech.edu/pub/avida.

Pantelimon, G., Tepe, K., Carriveau, R., and Ahmed, S. (2019). Survey of multi-agent communication strategies for information exchange and mission control of drone deployments. *Journal of Intelligent & Robotic Systems*, 95:779–788.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Phan, T., Sommer, F., Altmann, P., Ritz, F., Belzner, L., and Linnhoff-Popien, C. (2022). Emergent cooperation from mutual acknowledgment exchange. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1047–1055.

Raji, I. D. and Dobbe, R. (2023). Concrete problems in ai safety, revisited. *arXiv preprint arXiv:2401.10899*.

Randazzo, E., Versari, L., and Mordvintsev, A. (2021). Recursively fertile self-replicating neural agents. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press.

Ritz, F., Ratke, D., Phan, T., Belzner, L., and Linnhoff-Popien, C. (2021). A sustainable ecosystem through emergent cooperation in multi-agent reinforcement learning. In *Artificial Life Conference Proceedings 33*, volume 2021, page 74. MIT Press, One Rogers Street, Cambridge, MA.

Saha, M. and Isto, P. (2006). Multi-robot motion planning by incremental coordination. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5960–5963. IEEE.

Saravanan, N., Fogel, D. B., and Nelson, K. M. (1995). A comparison of methods for self-adaptation in evolutionary algorithms. *BioSystems*, 36(2):157–166.

Tanner, H. G. and Christodoulakis, D. K. (2007). Decentralized cooperative control of heterogeneous vehicle groups. *Robotics and autonomous systems*, 55(11):811–823.

Vanhie-Van Gerwen, J., Geebelen, K., Wan, J., Joseph, W., Hoebeke, J., and De Poorter, E. (2021). Indoor drone positioning: Accuracy and cost trade-off for sensor fusion. *IEEE Transactions on Vehicular Technology*, 71(1):961–974.

Wijayakulasooriya, J. V., Putrus, G., and Minns, P. (2002). Electric power quality disturbance classification using self-adapting artificial neural networks. *IEE Proceedings-Generation, Transmission and Distribution*, 149(1):98–101.

Zorn, M., Illium, S., Phan, T., Kaiser, T. K., Linnhoff-Popien, C., and Gabor, T. (2023). Social neural network soups with surprise minimization. In *ALIFE 2023: Ghost in the Machine: Proceedings of the 2023 Artificial Life Conference*. MIT Press.