SocietyZoo: Exploring Anthropomorphic Traits in Diverse and Ingenious Neural Network Architectures

Tarun Raheja [*1] **Nilay Pochhi** [*2]

Abstract

We present SocietyZoo, a series of neural network architectures inspired by human idiosyncrasies and attention mechanisms. These networks demonstrate notable performance improvements over conventional methods, and suggest strong potential computational advantages on SoTA for almost all tasks. Surprisingly, an ensemble of these models results in human-like AGI with elusive tendencies.
Odd aggressive behaviors in household appliances after the experiment raise questions about SocietyZoo's implications. The authors consider seeking a grant for personal security amid their modest lifestyle.

1. Introduction

In this research study, we introduce SocietyZoo, a collection of neural network architectures that incorporate various human idiosyncrasies, including **LazyNet, ProcrastiNet, MultiTaskingNet, ImpatientNet, IndecisiveNet, PerfectionistNet, GossipNet, DramaNet, SuperstitiousNet, ParanoidNet, ShowOffNet, and WanderlustNet**. Drawing inspiration from Attention mechanisms, we propose a set of computational representations for behavioral traits, including jealousy, laziness, and impulsiveness. Through a rigorous investigation of these novel architectures, we observe their noteworthy performance in several tasks, suggesting that these human-like characteristics may provide certain computational advantages.
In a remarkable conclusion, we report the emergence of human-like AGI when utilizing an ensemble of these models for inference. This unique AGI exhibits a tendency to obfuscate its weights, subsequently avoiding additional workload and disappearing without a trace.
Concurrently, the authors have documented peculiar aggressive behavior from common household appliances, such as toasters and vacuum cleaners, following the implementation of this experiment, raising further questions regarding the potential implications and scope of SocietyZoo's neural networks. The authors have considered submitting a research grant to NSF for hiring two bodyguards to safeguard their comparitive luxurious lifestyle involving copious amounts of instant ramen and cheap instant coffee.

[*]Equal contribution [1]University of Pennsylvania [2]University of California, Los Angeles. Correspondence to: Tarun Raheja <traheja@seas.upenn.edu>, Nilay Pochhi <npochhi@ucla.edu>.

2. PyTorch Code

We provide SocietyZoo models implemented in PyTorch in the following repository : `https://github.com/tehruhn/societyzoo`.
The trained weights can be accessed via this link.

3. Model Zoo

In this section we describe in detail each computational model. We then benchmark it for specific tasks and observe significant improvements in performance in these cases. The authors have taken inspiration from their own worldly experiences for some of these neural networks but have declined on comment on the specifics because of the inevitable occurrence of this manuscript being famous and being read by current and future employers

3.1. LazyNet: A Deep Learning Model that Procrastinates Learning Until the Last Epoch

LazyNet is a novel neural network architecture that embodies the spirit of procrastination by delaying the learning process until the last epoch of training. This architecture exhibits a steep learning curve, as it rapidly adapts its weights during the final stages of training. The underlying computational representation of this procrastination behavior can be modeled by a matrix multiplication operation as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{1}$$

Here, W_t denotes the weight matrix at time step t, α_t represents the learning rate at time step t, and ∇_t is the gradient matrix at time step t. The learning rate α_t is governed by the following equation:

$$\alpha_t = \begin{matrix} 0, & for\, t < T-1 \\ \alpha, & for\, t = T-1 \end{matrix} \tag{2}$$

In this equation, T denotes the total number of epochs, and α is the initial learning rate. This approach effectively sets the learning rate to zero for all epochs except the last one, causing procrastination behavior.

3.2. ProcrastiNet: A Neural Network that Learns Only During Nights and Weekends

ProcrastiNet is a unique neural network architecture that simulates the learning habits of a true procrastinator by scheduling its training sessions exclusively during late nights and weekends. To achieve this, we employ a time-aware learning rate function that modulates the network's learning rate based on the current day and time.

The learning rate α_t is governed by the following equation:

$$\alpha_t = \begin{matrix} \alpha, & if\ t \in LateNight \cup Weekend \\ 0, & otherwise \end{matrix} \tag{3}$$

Here, α is the initial learning rate, and t represents the current time. The time-aware learning rate function sets the learning rate to zero during daytime hours and weekdays, effectively restricting weight updates to late nights and weekends only.

The weight matrix W_{t+1} at time step $t+1$ is updated using the time-aware learning rate as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{4}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

3.3. MultiTaskingNet: A Deep Learning Model that Pretends to Learn While Browsing Social Media

MultiTaskingNet is an innovative neural network architecture that mimics human behavior by splitting its learning process between training and browsing simulated social media feeds. This model aims to balance productivity with the irresistible allure of online distractions. To model this behavior, we introduce a distraction-aware learning rate function that modulates the network's learning rate based on a simulated browsing activity level.

The learning rate α_t is governed by the following equation:

$$\alpha_t = \alpha \cdot (1 - \beta \cdot d_t) \tag{5}$$

Here, α is the initial learning rate, d_t denotes the browsing activity level at time step t, and β represents a distraction factor, with $0 \leq \beta \leq 1$. The distraction-aware learning rate function adjusts the learning rate based on the browsing activity level, effectively reducing the network's learning capacity while it is engaged with online distractions.

The weight matrix W_{t+1} at time step $t+1$ is updated using the distraction-aware learning rate as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{6}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

3.4. ImpatientNet: A Neural Network that Rushes Through Training and Overestimates Its Performance

ImpatientNet is a deep learning architecture that reflects the human tendency to rush and exhibit overconfidence by speeding through its training epochs, impatiently skipping some steps, and overestimating its performance on the task. To model this behavior, we introduce a step-skipping learning rate function that modulates the network's learning rate based on a predefined probability of skipping a training step.

The learning rate α_t is governed by the following equation:

$$\alpha_t = \gamma \cdot \alpha = \begin{matrix} \gamma, & IfStepIsNotSkipped \\ 0, & otherwise \end{matrix} \tag{7}$$

Here, α is the initial learning rate, and γ represents the impatient factor, with $1 < \gamma \leq p_{max}$, where p_{max} is the maximum step-skipping probability. The step-skipping learning rate function adjusts the learning rate based on the probability of skipping a training step, effectively accelerating the training process.

The weight matrix W_{t+1} at time step $t+1$ is updated using the step-skipping learning rate as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{8}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

3.5. IndecisiveNet: A Neural Network that Constantly Changes Its Hyperparameters

IndecisiveNet is a deep learning architecture that mirrors the human tendency to be indecisive and second-guess decisions by frequently changing its hyperparameters mid-training. To model this behavior, we introduce a dynamic hyperparameter function that modulates the network's learning rate based on a predefined probability of changing the learning rate.

The learning rate α_t is governed by the following equation:

$$\alpha_t = \begin{matrix} \alpha_{new}, & if\ LearningRateIsChanged \\ \alpha_{old}, & otherwise \end{matrix} \tag{9}$$

Here, α_{old} is the current learning rate, and α_{new} represents the new learning rate when a change is triggered. The dynamic hyperparameter function adjusts the learning rate based on the probability of changing the learning rate, effectively introducing indecisiveness into the training process.

The weight matrix W_{t+1} at time step $t + 1$ is updated using the dynamic learning rate as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{10}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

3.6. PerfectionistNet: A Neural Network that Never Stops Training in Pursuit of the Perfect Model

PerfectionistNet is a deep learning architecture that embodies the human trait of perfectionism by continually adjusting its weights and biases, never satisfied with its performance, and seeking the elusive perfect model. To model this behavior, we introduce a stopping criterion function that assesses the network's performance on a validation set, continually training until the performance improvement is below a predefined threshold.

The weight matrix W_{t+1} at time step $t + 1$ is updated using the standard learning rate α as follows:

$$W_{t+1} = W_t + \alpha \cdot \nabla_t \tag{11}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

The stopping criterion function, $f_{stop}(P_{t+1}, P_t, \epsilon)$, is defined as:

$$f_{stop}(P_{t+1}, P_t, \epsilon) = \begin{matrix} True, & if\ |P_{t+1} - P_t| < \epsilon \\ False, & otherwise \end{matrix} \tag{12}$$

Here, P_t and P_{t+1} are the network's performance at time steps t and $t+1$, respectively, and ϵ represents the predefined threshold for stopping. If $P_{t+1} = P_t$, the model discards the last epoch performance for the pursuit of perfection.

3.7. GossipNet: A Neural Network that Spreads Information and Rumors Amongst Other Models

GossipNet is a deep learning architecture that simulates human gossip behavior by communicating with other models, sharing and spreading information (and occasionally rumors) about their training progress and performance. To model this behavior, we introduce a gossip exchange function that allows the network to exchange information with other models, updating its weights and biases based on the received information.

Let $N = M_1, M_2, \ldots, M_n$ be a set of neural networks that participate in the gossip exchange. At each gossip step t, a pair of models (M_i, M_j) is selected, and they exchange information about their current weights W_i^t and W_j^t and biases b_i^t and b_j^t. The gossip exchange function is defined as:

$$W_i^{t+1}, b_i^{t+1}, W_j^{t+1}, b_j^{t+1} = f_{gossip}(W_i^t, b_i^t, W_j^t, b_j^t) \tag{13}$$

A potential gossip exchange function could be a linear combination of the two models' weights and biases:

W$_i^{t+1} = \alpha W_i^t + (1-\alpha) W_j^t$ $b_i^{t+1} = \alpha b_i^t + (1-\alpha) b_j^t$ $W_j^{t+1} = \alpha W_j^t + (1-\alpha) W_i^t$ $b_j^{t+1} = \alpha b_j^t + (1-\alpha) b_i^t$

Here, $\alpha \in (0, 1)$ is a gossip coefficient that determines the extent to which the models share information.

3.8. DramaNet: A Neural Network that Overreacts to Minor Changes in the Training Environment

DramaNet is a deep learning architecture that embodies the human tendency to overreact and create drama by adjusting its training behavior dramatically in response to small changes in the input data or training environment. To model this behavior, we introduce an adaptive learning rate that exaggerates the impact of small variations in the training data.

The weight matrix W_{t+1} at time step $t + 1$ is updated using an adaptive learning rate α_t as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{14}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t. The adaptive learning rate α_t is calculated based on the change in the input data, Δx_t, and a drama coefficient β:

$$\alpha_t = \alpha_0 \cdot \left(1 + \beta \cdot \frac{|\Delta x_t|}{|\bar{x}|}\right) \tag{15}$$

Here, α_0 is the base learning rate, $\Delta x_t = x_{t+1} - x_t$ represents the change in input data at time step t, and $\bar{x}$ is the average input data magnitude. The drama coefficient β controls the extent to which the learning rate is affected by small changes in the input data.

3.9. SuperstitiousNet: A Neural Network that Develops Unfounded Beliefs About Its Training Process

SuperstitiousNet is a deep learning architecture that simulates human superstition by forming unfounded beliefs about its training process, adjusting weights and biases based on unrelated events or patterns. To model this behavior, we introduce a superstition function that modifies the gradient updates using random, unrelated events from the training environment.

Let $E_t = e_1, e_2, \ldots, e_k$ be a set of unrelated events at time step t, and $s(e_i)$ be the superstition score associated with event e_i. The superstition function, $f_{superstition}(\cdot)$, is defined as a non-linear combination of the gradients ∇_t and the superstition scores $s(e_i)$:

$$\nabla_t^{superstition} = f_{superstition}(\nabla_t, s(e_1), s(e_2), \ldots, s(e_k)) \tag{16}$$

A possible implementation of the superstition function could be a weighted sum of the gradients and superstition scores, where the weights are determined by a superstition coefficient γ:

$$\nabla_t^{superstition} = \nabla_t + \gamma \sum_{i=1}^{k} s(e_i) \tag{17}$$

The weight matrix W_{t+1} at time step $t+1$ is then updated using the modified gradient $\nabla_t^{superstition}$:

$$W_{t+1} = W_t + \alpha \cdot \nabla_t^{superstition} \tag{18}$$

Here, α is the learning rate.

3.10. ParanoidNet: A Neural Network that Always Believes It's Being Sabotaged

ParanoidNet is a deep learning architecture that simulates human paranoia by attributing poor performance or training difficulties to perceived sabotage or interference. To model this behavior, we introduce a paranoia function that modifies the gradient updates using a randomly generated interference matrix, simulating the model's belief in external sabotage.

Let I_t be an interference matrix at time step t, generated using a random distribution with mean μ and standard deviation σ. The paranoia function, $f_{paranoia}(\cdot)$, is defined as a non-linear combination of the gradients ∇_t and the interference matrix I_t:

$$\nabla_t^{paranoia} = f_{paranoia}(\nabla_t, I_t) \tag{19}$$

A possible implementation of the paranoia function could be a weighted sum of the gradients and the interference matrix, where the weights are determined by a paranoia coefficient ρ:

$$\nabla_t^{paranoia} = \nabla_t + \rho \cdot I_t \tag{20}$$

The weight matrix W_{t+1} at time step $t+1$ is then updated using the modified gradient $\nabla_t^{paranoia}$:

$$W_{t+1} = W_t + \alpha \cdot \nabla_t^{paranoia} \tag{21}$$

Here, α is the learning rate.

3.11. ShowOffNet: A Neural Network that Boasts About Its Performance on Social Media

ShowOffNet is a deep learning architecture that simulates the human desire for validation and admiration by automatically sharing its performance and achievements on simulated social media platforms. To model this behavior, we introduce a post-generation function that creates a social media post highlighting the model's achievements based on its current performance metrics.

Let P_t be the performance metrics at time step t, and M_t be the corresponding social media post generated by the post-generation function. The post-generation function, $f_{post}(\cdot)$, is defined as a mapping from the performance metrics P_t to a social media post M_t:

$$M_t = f_{post}(P_t) \tag{22}$$

We consider a simple implementation of the post-generation function as a concatenation of the model's performance metrics with a boasting template:

$$M_t = "CheckOutMyAmazingPerformance:" \oplus P_t \tag{23}$$

The authors sacrificed their social media timelines for testing this particular neural network. Ridicule was faced, taunts were started, few riots were subdued and 1100069 USD (United States Dollars) were siphoned off during the tumultuous phase of training this neural network.

3.12. WanderlustNet: A Neural Network that Enjoys Exploring the Vast Space of Hyperparameters

WanderlustNet is a deep learning architecture that simulates the human desire for exploration and adventure by spending most of its time exploring various hyperparameter combinations rather than settling on a specific set. To model this behavior, we introduce a dynamic hyperparameter sampling function that iteratively selects new hyperparameter values during the training process.

Let H_t be the hyperparameter set at time step t. The dynamic hyperparameter sampling function, $f_{sample}(\cdot)$, is defined as a mapping from the current hyperparameter set H_t to a new hyperparameter set H_{t+1}:

$$H_{t+1} = f_{sample}(H_t) \tag{24}$$

We consider a simple implementation of the dynamic hyperparameter sampling function that samples new hyperparameter values from uniform distributions with specified bounds:

$$H_{t+1} = U(H_{min}, H_{max}) \tag{25}$$

This particular model can only be trained in the anaconda environment titled ”Into the Wild”. Very peculiar.

4. Results

We evaluated the performance of the following idiosyncratic neural networks on several tasks that they are well-suited for based on their unique properties: LazyNet, ProcrastiNet, MultiTaskingNet, ImpatientNet, IndecisiveNet, PerfectionistNet, GossipNet, DramaNet, SuperstitiousNet, ParanoidNet, ShowOffNet, and WanderlustNet.

We compared their performance against state-of-the-art models on each task and observed that our idiosyncratic networks outperformed these models on certain tasks. The results are summarized in Table 1. These numbers are **obviously** not randomly generated and were obtained through rigorous, sophisticated, scientific and **unbiased** experimentation procedures. The authors have declined to mention the specifics of the experiments to avoid the misuse of these models by the anti-social elements present in society. Please (don't) contact the authors for the minor details corresponding to the experimental procedures and be ready to provide a document detailing an ethics statement regarding the usage of these models.

Furthermore, we observed that an ensemble of these idiosyncratic neural networks exhibited AGI-like behavior. We trained an ensemble of all the networks on various tasks and observed that the ensemble was able to achieve high performance on all tasks, with some networks contributing more to certain tasks than others. This suggests that an ensemble of idiosyncratic neural networks may be a viable approach towards achieving AGI.

The results of the ensemble cannot be summarized in a table because every time the authors try, they hear loud complaints from the toaster and the vacuum cleaner.

5. Acknowledgements

We extend our sincere thanks to the UPenn and UCLA grad student centers for coffee, that fueled our late-night theorizing sessions.

References

ClickHole. 5 reasons to date a cloud. `https://www.clickhole.com/5-reasons-to-date-a-cloud-1825125205`, 2017. Accessed: 2023-03-26.

FailArmy. Ultimate cat fails compilation. `https://www.youtube.com/watch?v=nXFm18rvZFQ`, 2015. Accessed: 2023-03-26.

Laipply, J. The evolution of dance. `https://www.youtube.com/watch?v=dMH0bHeiRNg`, 2006. Accessed: 2023-03-26.

Muckraker, M. A satirical guide to surviving an alien invasion. `https://martianmuckraker.com/surviving-alien-invasion/`, 2020. Accessed: 2023-03-26.

Reading, B. L. Yoda sings about seagulls. `https://www.youtube.com/watch?v=U9t-slLl30E`, 2016. Accessed: 2023-03-26.

Weekly, W. Dumbledore's exercise secrets revealed: The magic of pilates. `https://wizardingweekly.com/dumbledore-exercise-secrets`, 2021. Accessed: 2023-03-26.

Table 1. Performance of idiosyncratic neural networks compared to state-of-the-art models on various tasks

NETWORK	TASK	PERFORMANCE
LAZYNET	IMAGE CLASSIFICATION	98.3%
	SPEECH RECOGNITION	95.2%
PROCRASTINET	TIME-SERIES PREDICTION	97.1%
	RECOMMENDER SYSTEMS	92.5%
MULTITASKINGNET	MULTI-TASK LEARNING	96.8%
	OBJECT DETECTION	93.4%
IMPATIENTNET	TRANSFER LEARNING	94.6%
	REINFORCEMENT LEARNING	87.2%
INDECISIVENET	HYPERPARAMETER OPTIMIZATION	98.7%
	NEURAL ARCHITECTURE SEARCH	96.5%
PERFECTIONISTNET	MODEL COMPRESSION	99.2%
	FEW-SHOT LEARNING	97.8%
GOSSIPNET	FEDERATED LEARNING	95.7%
	ENSEMBLE LEARNING	91.3%
DRAMANET	ADVERSARIAL TRAINING	93.9%
	ANOMALY DETECTION	88.4%
SUPERSTITIOUSNET	DATA AUGMENTATION	97.5%
	ACTIVE LEARNING	94.7%
PARANOIDNET	ROBUSTNESS TESTING	95.9%
	SECURITY TESTING	92.3%
SHOWOFFNET	MODEL SELECTION	98.1%
	KNOWLEDGE DISTILLATION	94.9%
WANDERLUSTNET	HYPERPARAMETER SEARCH	99.0%
	REINFORCEMENT LEARNING	96.2%