Social Learning in Neural Agent-Based Models

Igor Douven

Social Learning in Neural Agent-Based Models

Article *in* Philosophy of Science · August 2024

DOI: 10.1017/psa.2024.33

CITATIONS
4

READS
130

1 author:

Igor Douven
French National Centre for Scientific Research
241 PUBLICATIONS **4,137** CITATIONS

SEE PROFILE

Social Learning in Neural Agent-based Models*

Igor Douven
IHPST / CNRS / Panthéon–Sorbonne University

Abstract

Agent-based models (ABMs) are widely used in examining how interactions at the individual level shape the behaviors of collectives. It has recently been argued that ABMs tend to be too simple and abstract to capture the complexity and variability of real-world actors engaging in social interactions. We address this criticism by integrating artificial neural networks into ABMs, specifically focusing on enhancing the Hegselmann–Krause (HK) model. By replacing standard HK agents with multilayer perceptrons, we obtain a more realistic kind of ABM, more closely capturing the nature of actual agents. The approach yields more than one new model, given that, with multilayer perceptrons as agents, the core elements of the HK model can be defined in a number of ways. Through two computational studies, we compare the resulting models with each other and with a traditional individual-learning paradigm.

1 Introduction

Agent-based models (ABMs) have become a popular tool for studying macro-properties of social systems which, although typically arising from simple micro-level interactions, cannot be fully understood by strictly analytical means. They are used across a range of domains, from economics and political science to epidemiology and urban planning (Crosscombe & Lawry, 2016; Deffuant et al., 2000; Dittmer, 2001; Douven & Hegselmann, 2021; Lorig, Johansson, & Davidsson, 2021; O'Connor & Weatherall, 2019; Schelling, 1971), and philosophers of science have recruited ABMs to argue that social learning is key to the production and acquisition of scientific knowledge (Douven, 2010; Glass & Glass, 2021; Hegselmann et al., 2015; Huang, 2023; Kummerfeld & Zollman, 2016; Olsson, 2013; Olsson & Vallinder, 2013; Rosenstock, O'Connor, & Bruner, 2017; Zollman, 2007, 2010).

Although popular, agent-based modeling has recently come under a cloud. According to various authors, ABMs tend to oversimplify agent behavior, decision-making processes, and environments, which—these authors argue—undermines their ability to adequately capture the complexity and variability of real-world behavior and, thus, to yield accurate predictions when applied to actual social processes (see, e.g., Borg et al., 2019; Cristelli, 2014; Frey & Šešelja, 2018, 2020; Rosenstock, O'Connor, & Bruner, 2017; Šešelja, 2019; Thicke, 2020).

An obvious response to this critique is to make ABMs more realistic, which can be done, for instance, by letting interactions among agents be governed by more complex rules and making the agents' environment more like the real world in relevant respects. Several agent-based COVID-19 models were successful because of this approach. Not only did these models capture relevant population differences (in terms of age, health status, social behavior, mobility patterns, and so on) as well as the resulting heterogeneity of the interactions among agents, they were also able to incorporate data about the evolving pandemic almost in real time, features which made them

*The paper has supplementary materials consisting of an online-only appendix as well as the data and code used for the simulations. The Jupyter notebook containing the code also includes extra analyses of the simulation outcomes and a short tutorial on defining neural networks using the `Flux.jl` package for the Julia language (Bezanson et al., 2017). All materials can be downloaded from this repository: https://osf.io/fs29h/?view_only=71c0534b3bef4651aad8e68f88eb22f5.

valuable tools for policymakers in managing the pandemic (see, e.g., Adam, 2020; Douven, 2024b). A different approach, taken in this paper, is to make the agents *intrinsically* more human-like, by endowing them with some artificial form of intelligence. We aim to accomplish this by integrating ABMs with artificial neural networks (ANNs). The resulting neural agent-based models (NABMs) allow agents to learn from and adapt to their environment and interactions in a manner more akin to how humans learn and adapt.

This is still a very broad proposal, given the number of different ANN architectures on the market as well as the number of different ABMs with which they could be combined. As for ABMs, our focus will be on the Hegselmann–Krause (HK) model (Hegselmann & Krause, 2002, 2006, 2015, 2019), a well-established framework for studying opinion dynamics which enjoys considerable popularity in philosophy and beyond. We will combine this model with multilayer perceptrons (MLPs), which are among the oldest types of ANNs. The HK model captures the process of opinion formation and evolution within a society of agents, where the agents' opinions are influenced both by evidence obtained directly from the world and by the opinions of their peers, which are the agents whose opinions are close to their own opinion. By populating the HK model with MLPs, we aim to simulate the behavior of agents which form their opinions, or more generally update their doxastic states, not on the basis of simple arithmetic operations (as the agents in the HK model do) but rather by leveraging an ability to process complex information in a human-like way. Nevertheless, the dual updating mechanism characteristic of the HK model remains in tact in our NABMs, in that updating will still be one part data-driven, one part based on social interactions, where the latter is achieved through either parameter-averaging or prediction alignment or both (in ways to be explained).

Our primary goal is to present what we believe to be a promising approach to making ABMs more realistic, thereby also addressing the recent critique of such models. A secondary goal is to assess what remains of the seeming support from ABMs for the efficacy of social learning when this issue is considered in more realistic settings. We provide some theoretical background on the two main components of our NABMs (the HK model and MLPs) in Section 2. The HK-based NABMs are then presented in Section 3. Sections 4 and 5 report computational studies conducted using these models, the first study centering on a classification task, the second involving probabilistic updating in the context of medical diagnostics. Both studies address the question of the significance of social learning by comparing forms of such learning with each other as well as with individual learning.

As a preliminary note, we emphasize that the framework to be presented is meant as a blueprint for combining ABMs and ANNs generally, and that our methodology can be adapted to integrate ABMs with ANNs more sophisticated and state-of-the-art than MLPs, including the large language models (LLMs) that have been much in the limelight lately. At present, the requisite adaptations of the framework would encounter practical obstacles, for instance, due to the limited accessibility of cutting-edge LLMs—the most impressive ones being proprietary software—and the substantial computational resources required for training extensive numbers of larger networks. But anyone who has been following developments in the field of artificial intelligence will find it reasonable to expect that such challenges will be overcome sooner rather than later.

2 Theoretical background

2.1 The Hegselmann–Krause model

The Hegselmann–Krause (HK) model is among the most popular frameworks in the domain of agent-based computational modeling. While the model admits of a variety of interpretations (see, e.g., Hegselmann, 2023), it is most commonly interpreted to encapsulate the interplay between two key aspects of human epistemic behavior: the assimilation of information from social peers and the direct acquisition of knowledge from empirical evidence. On this interpretation, it serves as a

mathematical abstraction of opinion dynamics, where agents iteratively adjust their beliefs about the value of some parameter $\tau \in [0, 1]$, whose meaning remains unspecified.

Formally, each agent i starts at time 0 with an estimate $x_i(0)$ of τ and revises this estimate over discrete time steps, where at each time t the revision process is influenced by two primary factors: evidence about τ the agent receives directly from the world at t, and the opinions of its peers at t, which are formally defined to be the agents within its bounded confidence interval (BCI) at t, that is, whose estimates of τ at t differ by no more than some small value ϵ from the agent's own estimate at t. Then agent i's opinion concerning τ after the $(n+1)$-st update (i.e., at time $n+1$) is defined to be

$$x_i(n+1) \;=\; \frac{1-\alpha}{|X_i(n)|} \sum_{j \in X_i(n)} x_j(n) \;+\; \alpha\,\tau,$$

with $x_j(n)$ being the opinion of agent j after update n, and

$$X_i(n) \;:=\; \{j : |x_i(n) - x_j(n)| \leqslant \epsilon\}$$

the set of agents within agent i's BCI after update n. The parameter $\alpha \in [0, 1]$ balances the weight given to social versus evidential information. (For illustrations, see the supplementary materials.)

A key virtue of the HK model is that it can be easily extended or adapted for the purpose of addressing specific research questions. For instance, researchers have explored scenarios with "noisy" evidence, where agents receive imperfect signals from the world (Douven, 2010), and have considered agents with interval-valued beliefs to account for vagueness (Crosscombe & Lawry, 2016) as well as agents which can simultaneously hold beliefs about multiple issues (Jacobmeier, 2004; Lorenz, 2008; Pluchino, Latora, & Rapisarda, 2006).

In line with the general critique of ABMs cited in the introduction, one could argue that the HK model, and even the aforementioned extensions of the model, feature agents whose intellectual capacities are, for all we know, unrealistically impoverished. Proponents of the HK model could respond that this does not mean their model cannot be descriptively adequate at the macro-level, for instance, in predicting when a community of agents will reach a consensus and when not. While that is true, we believe a more productive, and independently interesting, response is to consider ways to make the HK model more realistic. One way is to endow the agents in the model with something like a brain, which is capable of learning much in the manner in which we humans learn. This is the approach to be taken here.

2.2 Multilayer perceptrons

The "brains" we are going to equip the agents with are going to be ANNs, specifically multilayer perceptrons (MLPs). Or rather, our agents are going to *be* MLPs, where these MLPs form communities and attend both to worldly evidence and to their peers.

ANNs not only have a brain-inspired architecture (Goodfellow, Bengio, & Courville, 2016, Ch. 1); they also reflect, to some extent, how the human brain operates (Caucheteux & King, 2022; Glorot, Bordes, & Bengio, 2011; Goldstein et al., 2021). More importantly for present purposes, ANNs have been shown to be an adequate tool for simulating various higher-level cognitive processes such as categorization, language learning, and reasoning (Battleday, Peterson, & Griffiths, 2021; Buckner, 2018, 2023; Douven, 2024a; Hoffman, McClelland, & Lambon Ralph, 2018; Hosseini et al., 2023).

MLPs are a specific type of ANNs, belonging to the family of feedforward ANNs, which are characterized by the unidirectional flow of data through the network. The MLP architecture dates back to the late 1950s and early 1960s, but it was not until the 1980s, with the introduction of the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986), that MLPs became able to learn from complex data patterns.

An MLP consists of several densely connected layers: an input layer, one or more hidden layers, and an output layer. Each layer is made up of nodes or neurons, with all neurons in the hidden and output layers being characterized by their weights (one for each neuron in the previous layer) and biases as well as by their activation function. The weights and biases are the adjustable parameters of the network, governing the strength of connections and the threshold for neuron activation, respectively, and the activation function is typically a nonlinear function, such as a sigmoid function or Rectified Linear Unit function (ReLU; this was used in the studies to be reported), which calculates the neuron's output on the basis of its inputs and the weights and bias associated with it.

MLPs are trained using a supervised learning technique (i.e., on the basis of labeled data) and learn through the aforementioned backpropagation algorithm. Specifically, the learning process in an MLP involves two phases, viz., propagation and weight update. During propagation, unlabeled input data is passed through the network, and each neuron processes the incoming data to produce an output, based on its associated weights, bias, and activation function. This output is then passed on to the next layer until it reaches the output layer, producing the network's prediction. Next, this prediction is compared to what the output *should* have been (i.e., the label that was not provided as input). The discrepancy between prediction and label is the "error" or "loss" from the output layer, which serves as input for the backpropagation algorithm. In backpropagation, the network adjusts its weights and biases to minimize the error between its predictions and the actual outcomes. This involves calculating the gradient of the loss function with respect to each parameter (weights and biases). The network then updates its weights and biases, using gradient descent or a similar optimization algorithm, to improve its performance.

MLPs have been used for a variety of tasks (e.g., image and speech recognition, natural language processing, and time-series prediction), and they have found application in a great number of areas, including financial forecasting and medical prognosis, where they aid in uncovering patterns and relationships in data that are not readily apparent to the human eye. In one of our studies, the agents (i.e., MLPs) engage in a multi-class classification task, in the other study, they are employed in the context of medical diagnostics.

3 HK updating for neural networks

The key elements of the HK model are the notion of peerhood, regulated by the ϵ parameter, and the operation of mixing worldly and social information, where the exact mixture depends on the value of the α parameter. We want to retain both elements in our new model, but, given that our agents are going to be MLPs, these elements need to be adapted.

In the original HK model, every agent is, at every point in time, fully characterized by its estimate of τ, making a definition of peerhood in terms of similarity of opinion the only plausible option. Accordingly, the model lets agents i and j be each other's peers at t precisely if $|x_i(t) - x_j(t)| \leqslant \epsilon$. But with MLPs as agents, one agent can be similar to another agent in more than one respect. Most notably, while in the original HK model there is no meaningful distinction between an agent's *state* at a given point in time and its *output* (i.e., its estimate of τ) at that point in time, in the new model there is. At any point in time, an agent is in a certain state, fully characterized by its parameters (its weights and biases) at that time (architecture, including activation functions, will always be the same for all agents in a community), but it can *also* be characterized by its output (i.e., the predictions it would then make, if prompted). As a result, we can distinguish between *state-based similarity* and *output-based similarity*; and of course agents can be similar to each other in both respects at the same time, which would make them *state- and output-based similar*.

Because MLPs can be used for various purposes, the output of an MLP can be many things: a single number, as in the HK model, or a grouping of items of interest into different classes (if the

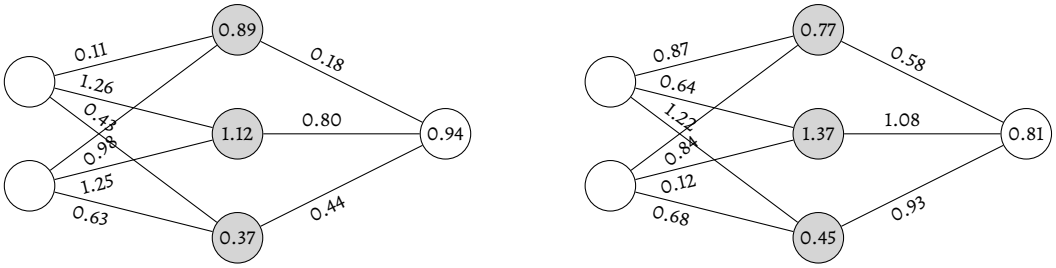

Figure 1: Multilayer perceptrons sharing the same architecture but with different weights and biases. (Weights are annotated on the edges connecting the neurons; biases appear inside the neurons.)

MLP is a classifier), or an assignment of probabilities to a set of competing hypotheses (e.g., if the MLP is used for a multinomial regression task), and so on. How to make the notion of output-based similarity precise will depend on the type of output we are dealing with. If, for instance, it is a single number, output-based similarity could again be defined in terms of absolute difference, as in the HK model; if the MLP is a classifier, there are a number of different metrics of classification similarity one can consider, such as the mutual information index, which we will use in the first study to be reported below; or if the output is a probability distribution, there are again a number of options available, such as the Kullback–Leibler (KL) divergence or the Jensen–Shannon (JS) divergence, the latter of which we will use in the second study; and so on. Two agents will then be said to be each other's *output-based peers* precisely if they are close enough to each other in terms of the appropriate criterion.

The notion of state-based similarity requires more explanation. Given that, in our models, all agents (i.e., MLPs) will have the same architecture—the same number of layers, corresponding layers having the same number of nodes, corresponding nodes having the same activation function—we can measure their similarity by comparing their parameters, node per node. A common metric for this purpose is the cosine similarity, which requires that we vectorize the parameters first.[1] Gathering, in some order, the weights and biases of agent i in a vector params_i and proceeding analogously for the weights and biases of agent j, obtaining params_j, their cosine similarity is calculated as

$$\text{cossim}(i, j) \;=\; \frac{\text{params}_i \cdot \text{params}_j}{\|\text{params}_i\| \times \|\text{params}_j\|},$$

which will be a value between −1 and 1, with 1 indicating maximum similarity and −1 maximum dissimilarity.

To illustrate, consider the MLPs shown in Figure 1. We lay out sequentially the parameters of each network from top to bottom and from left to right, and we calculate the dot product of the resulting vectors:

$$0.11 \cdot 0.87 \;+\; 1.26 \cdot 0.64 \;+\; 0.43 \cdot 1.22 \;+\; \cdots \;+\; 0.94 \cdot 0.81 \;\approx\; 7.35.$$

We further calculate that the norm of the first vector equals

$$\sqrt{0.11^2 + 1.26^2 + 0.43^2 + \cdots + 0.94^2} \;\approx\; 2.94$$

and that of the second vector equals

$$\sqrt{0.87^2 + 0.64^2 + 1.22^2 + \cdots + 0.81^2} \;\approx\; 3.09.$$

[1] To compare networks with different architectures, other metrics than the cosine similarity are recommended; see Chen et al. (2021).

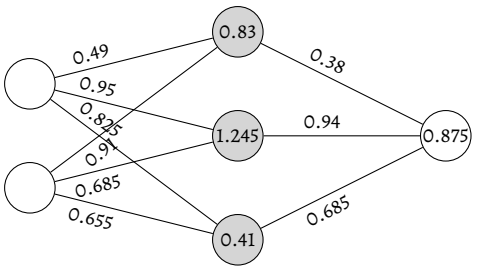

Figure 2: Multilayer perceptron with weights and biases resulting from averaging the corresponding weights and biases from the multilayer perceptrons shown in Figure 1.

Thus, the cosine similarity for the above MLPs equals (approximately) $7.35/(2.94 \times 3.09) \approx 0.81$.

We will say that agents i and j are *state-based peers* precisely if $\text{cossim}(i, j) \geqslant 1 - \epsilon$, for the chosen $\epsilon \in [0, 1]$; so, for instance, if $\epsilon = .2$, then the agents in the above illustration are each other's peers. Note that, as in the original HK model, a larger value of ϵ means a more liberal or inclusive notion of peerhood, which does not require agents to be as similar with respect to their parameters to qualify as peers; conversely, the smaller the value of ϵ, the more similar the agents have to be, with the limiting case of $\epsilon = 0$ meaning that the agents must be maximally similar, also analogous to the original HK model.

It merits emphasis that being able to differentiate types of peerhood—based on state, outcome, or a combination of the two—is already an enrichment compared to the original HK model. For, as social scientists have shown (e.g., Eysenbach et al., 2004; Laninga-Wijnen & Veenstra, 2023), in real life peer selection is influenced by a multitude of criteria: we may want to team up with people who share our views, but also with people who look like us or have the same educational background or socio-economic status. State-based peers could be regarded as corresponding somewhat to peers who "look like us," output-based peers as corresponding to peers who "have views like ours."

The averaging operation can take different forms as well, again due to the fact that, with MLPs as agents, we can make a state–output distinction. Supposing we have determined an agent's peers at a given point in time (be these state-based, output-based, or state- *and* output-based peers), one plausible option is to average the parameters of those peers and calculate the output of the network with the resulting averages as parameters, given the input at the point in time; another, equally plausible option is to calculate the outputs of all peers at the point in time and average those outputs. In general, the results will be different. Suppose, for instance, the two MLPs depicted in Figure 1 are both given as input the vector $(2/3, 1/3)$. Then it is an easy (if somewhat tedious) exercise to calculate that the left MLP will give as output (approximately) 3.70 and the right one will give as output (approximately) 5.33, yielding an average of (approximately) 4.52. But applying the procedure of the first option to the same MLPs results in the network shown in Figure 2. And this network yields (approximately) 4.46 when given $(2/3, 1/3)$ as input.[2]

The different definitions of peerhood and averaging can be combined in a variety of ways to obtain NABMs whose agents update in a HK-like fashion. We will make no attempt to be exhaustive here and confine ourselves to studying three models that could all be plausibly regarded as extensions of the HK model, the main difference in all three cases being that the traditional HK agents have been replaced by MLPs. Roughly, the first model assumes a state-based notion of peerhood and also averages agent parameters instead of outputs. The second model assumes an output-based notion of peerhood and averages outputs. And the third model combines the first and second, which means that it proceeds by averaging parameters of state-based peers but also by averaging outputs

[2]We are assuming ReLU activation functions here.

of output-based peers. In the remainder of this section, we describe each of the models in more detail, and in the next two sections we use computer simulations to compare their performance on standard machine learning benchmarks.

All three models require as input a community of agents, which will be MLPs but could also be different types of networks; input data, split into a training and a test set; and values for parameters regulating peerhood and the mixing of worldly and social factors in updating. There is no restriction on the exact architecture of the MLPs, except that (i) it must be the same for all MLPs in a given community, meaning that they must have the same number of layers and that corresponding layers must have the same number of nodes as well as the same activation function, and (ii) the input and output layers must (of course) fit the data and task, respectively. In the first two models to be considered, the parameter ϵ regulates the criterion for peerhood (which, however, means different things in the two models) and the parameter α regulates the weighting of the worldly versus the social factor in updating (the weighing operation also means different things in the two models). The third model, which as said combines the first two, has two parameters regulating peerhood—one regulating state-based peerhood (ϵ_1), the other output-based peerhood (ϵ_2)—as well as two weight parameters, one pertaining to the weighing of states (α_1), the other pertaining to the weighing of outputs (α_2).

The first model consists of three main parts. The first part calculates a cosine similarity matrix for all agents in the community and, on that basis, selects peers for each agent (i.e., the agents which are ϵ-similar to it). It then calculates the averages of those parameters, in the way illustrated previously, and it stores these parameter averages. The second part, which can be thought of as the worldly part of the updating process, trains for one training round every agent (i.e., MLP) on the data it received, where it is left open at this point whether all agents receive the same data or receive different (possibly partly overlapping) subsets of the data. The third part, finally, takes a weighted average of the parameters of the agent that resulted from the training process in the second part and the parameter averages of the agent's peers that were calculated in the first part, the weighing depending on the value of α. The parameters that result from this weighted averaging are then set as the new parameters of the agent. Algorithm A.1 in Appendix A.2 presents pseudo-code for the updating method defined by this model. In that presentation, the procedure outputs both the updated agents and the results from evaluating the updated agents on the relevant data (the training set, or the test set, or both, whichever is most useful for one's purposes).[3]

The building blocks of the second model are basically the same as those of the first model, but they appear in a different order: First, all agents are trained on whatever the relevant data are (where it is again left open whether all agents are trained on the same data or whether each agent receives its own data set); then they make predictions, whether for their training data or their test data or both (e.g., if the task at hand is one of classification, they predict, after being trained, how each data point will be classified); in a next step, the peers of each agent are determined on the basis of how similar their predictions are (the similarity cut-off depending on ϵ); and finally, some α-weighted average of the agent's predictions after the worldly update and the averaged predictions of its peers is calculated and then evaluated. Algorithm A.2 in Appendix A.2 presents the pseudo-code for the second model. As presented there, the procedure gives the result of the final step (i.e., of the weighted averaging) as output, together with the updated agents.

The notion of averaging, as it is used in the second model, requires a comment. Parameters are always numbers and we know what it means to average numbers. So, in the first model, averaging always means taking the arithmetic average of whatever the relevant numbers are. But as already explained, given the many kinds of tasks MLPs can fulfill, the outputs in the case of the second model

[3] For a still better understanding of the computational details, readers are invited to consult the Jupyter notebook in the supplementary materials, which contains the Julia code of the simulations reported in the following sections.

need not be numeric. As a result, the operations of averaging and weighted averaging, as carried out in the model, may differ, depending on the nature of the data or the task at hand. For instance, in the first study below averaging consists in determining the modal responses for the various data points, in a way to be detailed. Nevertheless, the intended meaning of averaging in this algorithm should be clear: the average is always some kind of best compromise of whatever different responses are under consideration.

The third model combines the previous two. Specifically, it proceeds as follows: (i) for each agent, select its state-based peers (depending on ϵ_1) and take the averages of their parameters; (ii) train all agents on their training data; (iii) for each agent, set its parameters equal to a weighted average of its parameters after the training and the averaged parameters of their state-based peers (before training; the weighting of the average is determined by α_1); (iv) let all agents make predictions on the relevant data; (v) for each agent, select its output-based peers (based on ϵ_2), in light of the predictions obtained in the previous step; and finally (vi) for each agent, take an α_2-weighted average of its own predictions and the averaged predictions of its output-based peers.

While in the description of the models we have not explicitly referred to an equivalent of the parameter τ in the HK model, the references to agents' making predictions and being evaluated all refer *implicitly* to such an equivalent, that is, a target that the agents are aiming at and can get right to differing degrees; also, the data will, ideally, be informative of that equivalent, meaning that they will help the agent approximate the target, or even hit it. But precisely because MLPs can be used for a variety of purposes, it is impossible to characterize the target generally. If the MLPs are trained on a classification task, the aim is to classify correctly whatever data they are given as input; their predictions concern the classification of those data—they are their best guess of how the data are classified in reality—and they are evaluated in light of how closely their predictions match the correct classification. Similarly if the MLPs are trained to assign probabilities to a set of rival theories, or to predict time series, or to encrypt data. In all those cases, there is a target that they are trying to come as close as possible to and with respect to which they can be evaluated, but the nature of the target is different each time, unlike in the HK model, where it is always a number (or set of numbers, in some extensions).

It is again to be noted that we are not aiming at exploring all possibilities of integrating the HK model with ANNs. We do believe, however, that the three models defined in the foregoing are all natural extensions of that model as originally conceived, the new characteristic element being that the communities of agents are constituted by neural networks. While it has been shown that, depending on an agent's environment and its goals (epistemic or otherwise) in that environment, social updating in the manner of the HK model can have notable epistemic advantages (Crosscombe & Lawry, 2016; Douven, 2010, 2019; Douven & Hegselmann, 2021, 2022; Glass & Glass, 2021), it remains to be seen whether there is any merit to HK updating for agents conceived as neural networks. To find out, the next two sections test the three models on tasks neural networks have been commonly used for, and we compare the performance of networks in the models with that of neural networks carrying out the same tasks in a strictly individual fashion.

4 Study I: Classifying colors

The first study considers communities of agents (i.e., MLPs) that are trained to classify colors on the basis of their coordinates in color similarity space, specifically CIELUV space (see the left panel of Fig. A.1 in App. A.3; also Fairchild, 2013, for theoretical background). Both the training and the testing materials come from the 320 chromatic Munsell chips which served as the materials for the World Color Survey (WCS; Cook, Kay, & Regier, 2005), a large catalogue of color-naming systems

from across the globe; the 320 chips are highlighted in the right panel of Figure A.1 in Appendix A.3 and shown in a chart in the way they were presented in the WCS in Figure A.2 in the same appendix.

Because a significant number of participants in color-naming studies for both English and French used only ten of the eleven basic color terms ("green," "blue," and so on) in describing the colors of the WCS chips, leaving out "gray" (Berlin & Kay, 1969; Claidière, Jraissati, & Chevallier, 2008), we take as the target classification which the agents should try to learn—our τ, so to speak—a clustering of the WCS chips into ten categories. Also, because in the same color-naming studies there was considerable interpersonal variability in how these chips were named, we use the k-means clustering algorithm to provide a kind of objective approximation of the natural color concepts.[4] The result, which is the classification the agents should try to learn, is shown in the top row of Figure A.3 in Appendix A.3.

The MLPs that populate the models are not much more complicated than the ones used in our earlier illustration. They also have only one hidden layer, now consisting of nine nodes and integrating the Rectified Linear Unit (ReLU) activation function for each node.[5] Given that the task at hand is to categorize colors as belonging to one of ten classes on the basis of their CIELUV coordinates, the input layers of the MLPs have three nodes—one for each coordinate—and the output layers ten, each representing one of the basic colors minus gray.

We ran three sets of simulations, one for each of the models defined in the previous section, where the communities always consisted of 50 MLPs with the architecture described above. Each simulation involved training the agents over 100 epochs, where an epoch is a single application of the given model, with the agents being returned after epoch n serving as input for the model in epoch $n+1$, for $n \in \{1, \ldots, 99\}$.[6] The training used the Adam optimization algorithm (with a learning rate set to 0.001) and the Multiclass Cross-entropy Loss, which computes the loss by measuring the difference between the predicted classification probabilities (i.e., the probability that a chip should be classified as green, the probability that it should be classified as blue, and so on) and the true class labels.

Per epoch, the agents received a fresh batch of training data, each time sampled randomly and for each agent individually from the WCS chips in such a way that the number of chips from each category according to the target classification was greater than 0 but otherwise random. Thus, every agent was assigned at the beginning of each epoch a set $\left\{\left\langle\langle L^*_c, u^*_c, v^*_c\rangle, C_c\right\rangle\right\}_{c \in s}$ of pairs as training data, with each pair comprised by the CIELUV coordinates of some WCS chip c in sample s as well as its label C_c indicating the color it has according to the target classification.[7] The test data, on which the agents were evaluated after each epoch, were always the same for all agents and consisted of the coordinates of all 320 color chips together with their labeling according to the target classification.

The evaluation used the mutual information index, which measures the similarity of different classifications (see Pfitzer, Leibbrandt, & Powers, 2009, for why this measure is preferable to alternative measures, such as the Rand index).[8] To be more precise, after each epoch we measured the

[4] See Douven (2017, 2024c) for more on this; how close the approximation is is unimportant for present purposes.

[5] In light of recent work on ANNs, this is an exceedingly simple and shallow architecture, which, by today's standards, does not even qualify as *deep* (see, e.g., Buckner, 2023, p. 50; Buckner and Garson, 2018). But everything said in this paper generalizes to MLPs with any number of hidden layers and even, with some qualifications, to more recent architectures; see Section 6.

[6] At start time (i.e., epoch 1), the layers of the agents were initialized using the Xavier method introduced in Glorot and Bengio (2010).

[7] For a more detailed description of the procedure, see Douven (2024c). As noted in that paper, there is no fixed sample size in this procedure, given that a *random* number of chips is sampled from each color category. The average sample size was empirically determined to be 165.02 (± 31.98).

[8] We used the normalized version of this measure, so that mutual information values were always between 0 and 1, with 0 indicating that the classifications are maximally dissimilar and 1 indicating that the classifications are maximally similar (i.e., identical). For a formal definition of this measure, and for formal definitions of all other technical notions to be used in the following, see Appendix A.1.

accuracy of each agent by calculating the mutual information between how it classified the 320 chips in our materials and how these chips ought to be classified according to the target classification.

For the first two models, which have only one ϵ and one α parameter, we used a grid search strategy to approximate optimal combinations of parameters. For each combination resulting from letting ϵ and α range independently over the unit interval in steps of .025, we ran 100 simulations as described above. For state-based social updating, the parameter setting $\alpha = \epsilon = .9$ yielded, on average, the highest mutual information at the end of the training process. For output-based social updating, the combination of $\alpha = .1$ and $\epsilon = .3$ did best. For the combined social updating method, which has four parameters, a grid search would have been computationally too costly and therefore we ran a random search procedure to approximate the best setting (or *a* best setting; uniqueness is not guaranteed). Specifically, we ran 100 simulations for 500 combinations of random choices (all uniformly sampled from the unit interval) for the two α and the two ϵ parameters, finding that the best score (i.e., the highest average mutual information) after 100 epochs was obtained for the setting $\alpha_1 = .81$, $\epsilon_1 = .9$, $\alpha_2 = .14$, and $\epsilon_2 = .07$.[9]

To get a first impression of the accuracy that can be achieved using the different updating mechanisms with their optimal parameter setting, we trained a community of 50 agents for 1,000 epochs for each of these mechanisms and compared the resulting modal classifications with the target classification. (A modal classification is the classification which gives, for each chip in our materials, the modal—i.e., most frequent—response for that chip in the given community.) For completeness, we included in the comparison the modal classification obtained from a community of 50 agents (MLPs) that do not engage in any social updating but are individually trained in the exact same way as the agents in the communities of social updaters are. It turned out that, first, the different updating methods led to modal classifications that looked almost the same, and second, that those classifications were almost identical with the target classification (see Fig. A.3 in App. A.3). Indeed, a comparison with the target classification yielded the same high mutual information of .97 for each modal classification.

Should we conclude that the various forms of social updating are equally good but also that social updating, in whichever form, is not worth the extra effort of averaging (whether parameters or predictions, let alone both)? That would be rash, because the modal classifications tell a very incomplete story, for two reasons. First, note that modal responses can be the same even if, for one algorithm, only a small fraction of agents got the label right at the end (but wrong responses were all over the place), while for another, all, or almost all, agents got it right. Second, we will want to look at more than the end state of the training process and will also be interested in how *fast* the agents were able to learn. Perhaps all updating methods led to an excellent classification eventually, but if one already got the classification more or less right quite early on in the training process, while the other updating methods did not, then for many practical purposes that will make the former preferable.

On these issues, Figure 3 offers some helpful insights. For each of the four communities of agents under consideration (i.e., the community of nonsocial updaters, and the three communities of social updaters, each using a different updating method with their optimal setting) and for each epoch, the figure shows the mean mutual information obtained by the agents, together with 95 percent confidence bands. We see that the combined state-based and output-based procedure swiftly surpasses the others, maintaining its lead throughout the training process.

We conducted one-way ANOVAs for the mutual information scores of the four groups after each epoch. The ω^2-values for each ANOVA are plotted on the alternative y-axis of Figure 3, a green marker indicating that the ANOVA showed group means to be significantly different, a red marker that they were *not* significantly different. An ω^2-value greater than 0.14 is conventionally taken

[9] See the supplementary materials for details and additional analyses.

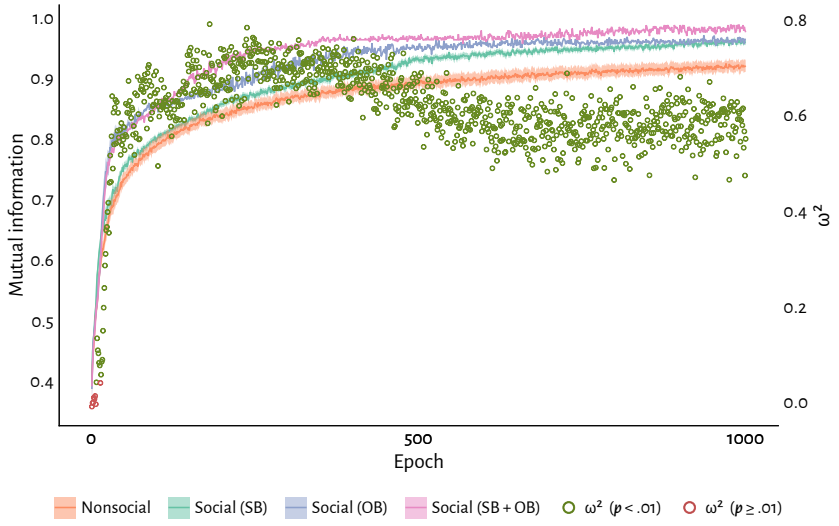

Figure 3: Per-epoch average mutual information (with 95 percent bootstrap confidence intervals) for the four communities of agents (social updating always with optimal settings; see the text). Effect sizes (ω^2) for the ANOVAs that were run for each epoch are shown on the alternative y-axis. (Here and elsewhere: SB = state-based; OB = output-based.)

to indicate a large effect size, meaning in our case that, although the *modal* classifications of all communities were equally good at the end, even at the end there were large differences among the communities in terms of average mutual information scores (i.e., how well, on average, members of the communities did with respect to approximating the target classification).

Results of the per-epoch follow-up tests with pairwise comparisons, which are contained in the supplementary materials, further reveal that not only does the combined updating method top all of its rivals after virtually all epochs, but at almost every epoch a choice of the former over any of the alternatives would largely impact the achieved accuracy (where the effect size was measured using Cohen's d). The only method that at times comes close and sporadically even does better is the output-based social updating method. The pairwise comparisons also confirm what could already be guessed on the basis of Figure 3, viz., that all social updating methods outperform individual updating by far.

We can also measure the total accuracy achieved by the agents over the 1,000 epochs by using the area under the learning curve (AULC; see, e.g., Bouckaert, 2006; Tsai, Ho, & Lin, 2010), which plots the learning curve of a neural network and measures the area under that curve. Networks that learn faster and achieve greater accuracy sooner will have a larger area under the learning curve, while models that learn more slowly or achieve a lower level of accuracy will have a smaller area, assuming the same number of epochs. Thus, the AULC can be interpreted as a measure of the overall performance of the network throughout the training process, with larger values for this metric indicating better average performance of a network throughout the training process.

The AULC values obtained for the agents in the community using the combined social updating method were significantly larger than those obtained for the agents in the other communities. Specifically, a one-way ANOVA showed that type of updating had a significant and very large effect on AULC values over 1,000 epochs; $F(3, 196) = 200.58$, $p < .0001$, $\omega^2 = 0.75$. Pairwise t-tests showed that the AULC values for the agents using combined social updating significantly exceeded those for the agents using any other method of updating (smallest $t = 52.04$, all $ps < .0001$), with a mean AULC value for the combined method of 938.40 (± 0.56), for the output-based method of 921.54 (± 2.22), for the state-based method of 897.00 (± 0.90), and for the nonsocial method of 866.95

(± 31.44). A Cohen's d test showed that using the best method (combined updating) instead of its closest competitor (output-based updating) still has a large impact on overall accuracy (d = 10.41).

5 Study II: Staging hypertension

The setup of the second study is broadly the same as that of the first. We look at communities of agents which aim at a target and receive evidence relevant to that target. In this study, too, the communities use different update methods, with one community consisting of nonsocial updaters, and three communities consisting of social updaters, one for each of the three models from Section 3.

The target is different in this study. For ease of interpretation, imagine the agents to be medical interns tasked to predict the stages of hypertension in patients based on a variety of demographic and lifestyle data, intentionally excluding direct blood pressure readings. For the training process, we use data sourced from the National Health and Nutrition Examination Survey (NHANES), which is a yearly survey conducted by the National Center for Health Statistics.[10] From an initial cohort of 613 patients, we focus on the 587 adults aged 20 and above. As key variables for analysis, we include age, gender, body mass index (BMI), diabetic status, physical activity, alcohol consumption, and smoking behavior. These variables present a mix of continuous (such as age, BMI, and physical activity), binary (gender), and ordinal (diabetic status, alcohol use, smoking) types. On the basis of these variables, the interns are to predict class probabilities for hypertension stages, ranging from normotensive (i.e., normal blood pressure) via pre-hypertensive, stage 1 and stage 2 hypertensive, to hypertensive crisis, thus encompassing five distinct categories. Hypertension stages were determined using the systolic and diastolic blood pressure readings included in the NHANES data set.

The agents are again modeled as MLPs, now comprising two hidden layers with 32 and 16 nodes, respectively, employing the ReLU activation function. The input layer is designed to match the seven input variables (age, gender, etc.), while the output layer consists of five nodes corresponding to the hypertension stages. We use the softmax function in the output layer to model the output as a probability distribution, ensuring that the nodes' outputs all lie between 0 and 1 (inclusive) and that their sum equals 1.

In the training process, an agent processes data from one patient at a time and assigns, on the basis of this data, probabilities to each of the relevant hypotheses (i.e., that the patient is normotensive, that she is pre-hypertensive, etc.). As in the first study, the training uses Multiclass Cross-entropy Loss and the Adam optimization algorithm (with a learning rate of .005). This process represents the worldly part of the update, which for one community of agents is all the updating they engage in. Three other communities of agents also participate in social updating, each using a distinct one of the update methods introduced in Section 3.

As noted in Section 3, state-based peers are always selected on the basis of the same criterion—viz., similarity of weights and biases—but the criterion on whose basis output-based peers are selected depends on the type of output generated. In the present case, the output consists of probability functions, and so we need a similarity measure for such functions. A prominent one is the Kullback–Leibler (KL) divergence, but here we use the Jensen–Shannon (JS) divergence, D_{JS}, which is based on the KL divergence but which, unlike the latter, is symmetric, bounded, and normalized. That not only makes it easier to interpret (0 indicates that the probability functions are identical, 1 that they are maximally different), it can also be bounded by an ϵ parameter whose value lies in the unit interval. Thus, where one agent's predicted probabilities at a given point in time are

[10] We used the presently most recent batch of data available on the NHANES website, namely, the data collected from the beginning of 2017 until March of 2020. The data can be downloaded from this address: https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?Cycle=2017-2020.

represented by p and another's by q, they will be said to be each other's peers at that time precisely if $D_{\mathrm{JS}}(p \parallel q) < \epsilon$, for some specified $\epsilon \in [0, 1]$.

Relatedly, and as also previously explained, averaging of outputs will mean different things depending on the nature of the outputs. Here, they are probability functions, and we use the best known method for averaging such functions, which is linear pooling. Given probability functions $\{f_i\}_{i=1}^{n}$, the weighted linear average of these functions is defined to be $\sum_{i=1}^{n} \omega_i f_i$, with $\omega_i \geqslant 0$ for all i and $\sum_{i=1}^{n} \omega_i = 1$ (see, e.g., Dietrich & List, 2016). In our model, peers are always weighted equally, meaning that we always take the straight average of their probability functions.

After each update, the agents are evaluated on the patients in the test set. Because they are making probabilistic predictions about these patients, and given that the hypotheses the probabilities get assigned to are *ordered* (e.g., stage 1 hypertension is closer to stage 2 hypertension than the pre-hypertensive stage is), we evaluate the agents using the ranked probability score (RPS; see Epstein, 1969). This scoring rule is particularly suited for the kind of case at hand, given that it penalizes predictions not only on the basis of how much they differ from the objective probabilities but also on the basis of the "distance" between the hypotheses in terms of their order. For example, if an agent incorrectly assigns a high probability to a stage that is adjacent to the true stage (e.g., assigning a high probability to the hypothesis that the patient has stage 1 hypertension when the patient actually has stage 2 hypertension), this is considered a less severe error than assigning a high probability to a more distant stage (e.g., assigning a high probability that the patient is normotensive, in the same scenario). After each update, we calculate the RPS for each agent and each patient, and then average over all patients in the test set to obtain the overall score for the given agent after the given update. Note that lower RPS values indicate better predictive performance, with 0 being the ideal score, indicating perfect predictions.

We are interested in ascertaining whether social learning enhances the predictive accuracy of the agents and, if so, which of the social learning methods introduced in Section 3 proves most effective. To optimize the parameters for the social methods, we proceed as in the previous study, performing grid searches for the state-based and output-based methods and a random search for the combined method. This yields a best setting of $\alpha = .55$ and $\epsilon = .98$ for the state-based method, of $\alpha = .1$ and $\epsilon = .9$ for the output-based method, and of $\alpha_1 = .99$, $\epsilon_1 = .14$, $\alpha_2 = .05$, and $\epsilon_2 = .74$ for the combined method. (See the supplementary materials for details.)

We use computer simulations to compare the social methods with optimal parameter settings both with each other and with individual updating. More specifically, we run 50 simulations, each of which starts by randomly splitting the selected NHANES data 70–30 into a training set of 410 patients and a test set of 177 patients. The 410 patients in the training set are further randomly partitioned into 10 equally sized parts of 41 patients. Each of these parts then serves as the training set of one of the interns in each of four communities of 10 interns, where each community uses a different one of the four update methods we are interested in (i.e., either individual updating or one of the three social methods).

Figure 4 shows, for the four types of communities and for each update, the average (averaged over the 50 simulations) of the average (averaged over the 10 agents in the given community) RPS scored at the given update. As is already clear from the graphs, the individual updaters do, on average, worst, even by a wide margin (certainly when compared with the output-based and combined social updaters). It is equally clear that the output-based and combined social methods do better than the state-based social method. Although less clear, it seems that the output-based method does, at least for most of the updates, slightly better than the combined method.

All of this is confirmed by the ANOVAs with post hoc t-tests that we conducted for the simulation results per update. The outcomes are reported in the supplementary materials, which show, among other things, that the ANOVAs were all highly significant and that they all had ω^2-values well above 0.16. (As mentioned earlier, values for this statistic above 0.14 indicate a large effect

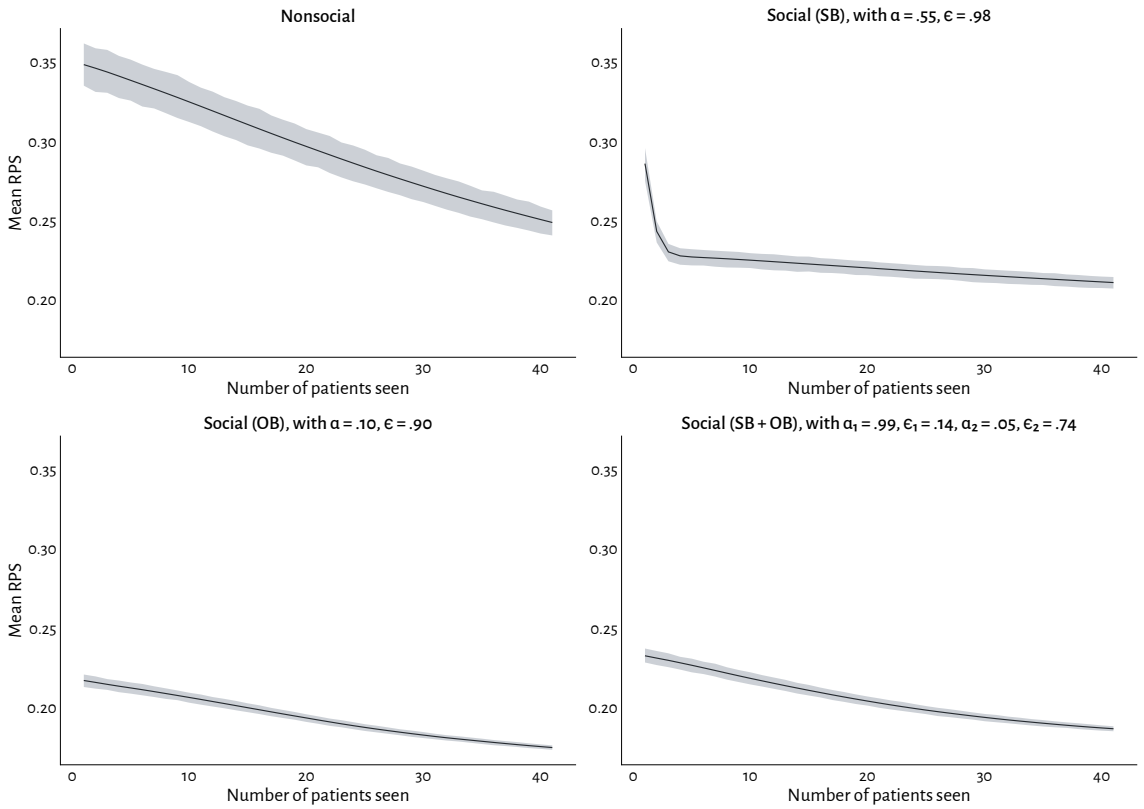

Figure 4: Per-update average (with 95 percent bootstrap confidence intervals) over 50 simulations of mean RPS achieved by agents, shown separately for the four communities of 10 agents. (See the text for further explanation.)

size.) These outcomes were to be expected in light of Figure 4, given the notable differences between, on the one hand, the nonsocial and, on the other, all of the social updating methods. The results from the Cohen's d-tests that were also part of the follow-up tests are more informative and show that choosing a social method over nonsocial updating mostly has a large ($d > 0.8$), and always at least a medium ($d > 0.5$), impact on accuracy. And choosing the output-based method over either of the other social methods has at least ultimately a medium impact on accuracy. (See again the supplementary materials for further details.)

As we did in the first study, we end by looking at the total accuracy the agents achieved during the training process, using again the AULC. The measure of accuracy in the second study is a scoring rule, which assigns penalties to agents. So, while in the first study we were interested in which updating method achieved the *largest* AULC, in this study better performance is indicated by a *smaller* area under the learning curve. A one-way ANOVA reveals a significant and substantial effect of the updating method on the accuracy of predictions; $F(3, 1996) = 189.73$, $p < .0001$, $\omega^2 = 0.22$. Follow-up t-tests confirm that all types of social updaters achieved significantly greater accuracy than individual updaters, which achieved a mean AULC of 11.61 ($\pm$ 5.02; smallest $t = 11.71$, all ps $< .0001$, smallest $d = 0.68$). Furthermore, the state-based method users, which achieved a mean AULC of 8.74 ($\pm$ 2.14), did significantly worse than both the output-based method users, with a mean AULC of 7.83 ($\pm$ 1.09; $t = 8.63$, $p < .0001$, $d = 0.96$) and the combined method users, with a mean AULC of 8.75 ($\pm$ 2.14; $t = 6.56$, $p < .0001$, $d = 0.39$). Finally, the output-based method users did significantly better than the combined method users, though the size of the effect is small in this case ($t = 3.28$, $p < .005$, $d = 0.18$).

6 Conclusion

In this paper, we introduced three neural agent-based models (NABMs) that extend the traditional Hegselmann–Krause (HK) model by integrating multilayer perceptrons (MLPs). Our models go beyond the scalar opinion representation in the HK model, enabling agents to perform complex learning tasks. The agents of the new type have not just enhanced learning capabilities individually but are also capable of richer social interactions, which were seen to further improve learning.

Our computational studies, focusing on the classification of Munsell color chips and probabilistic predictions about hypertension stages, demonstrated the effectiveness of these new extensions of the HK model. Agents employing social updating consistently outperformed individual learners, underscoring the value of social learning. The results also suggested task-specific nuances in the efficacy of different updating strategies, highlighting the importance of context in social learning.

The results from our computational studies not only validate our models but also help to address the criticisms directed at agent-based modeling in, for instance, Cristelli (2014), Frey and Šešelja (2018, 2020), and Borg et al. (2019). For many ABMs, it may well be true that, as these critics allege, they are too simplistic and idealized for real-world applications. We hope to have shown, however, that this *need* not be the case and that, by equipping ABMs with ANNs, we can model realistic forms of learning and adaptation, far beyond the limitations imposed in traditional models like the HK model. Not only that: using the new models, we obtained results showing the efficacy of social learning, in line with previous studies, which however relied on models whose validity had been called into question by the aforementioned critique.

We have limited our attention to extending one specific ABM by populating it with one specific type of artificial neural network (ANN). It would be wrong to state that our proposal generalizes swiftly to *any* kind of ABM and *any* kind of ANN. However, there are many ABMs close enough to the HK model (e.g., Deffuant et al., 2000; Friedkin & Johnsen, 1990; Olsson, 2013) that combining them with ANNs in the manner of this paper should be straightforward. As for other network architectures, the key operations of the models proposed in this paper—judging similarity on the basis of state and on the basis of output, and averaging states and outputs—apply as readily to, for instance, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) as they do to MLPs. Thus, an obvious avenue for future research would be to study the HK model and similar ABMs with either CNNs or RNNs as agents and see how well they do in solving tasks appropriate for the type of network used (e.g., image recognition if the agents are CNNs, or predicting time series data if the agents are RNNs).

More challenging follow-up research would focus on advancing the complexity of ANNs integrated within ABMs beyond that of the ones just mentioned. Recent work has shown how large language models (LLMs) can be made to communicate, in that one LLM's output serves as the prompt for one or more other LLMs, and so on, recursively (Du et al., 2023). That could be the basis for developing NABMs structurally similar to, but much more powerful than, the ones studied in this paper. Naturally, comparing the internal states of LLMs is not nearly as straightforward as comparing the internal states of MLPs, and measuring the similarities between the outputs of LLMs may also be harder. But there is some work on measuring the similarity of LLMs (Chen et al., 2021), and how to compare outputs is something that will have to be decided on a case-by-case basis anyhow, as we saw already for the simple MLPs we used. Supposing these hurdles can be overcome, NABMs featuring LLMs as agents may hold the potential to enhance our understanding of complex social behaviors by enabling the study of the interplay between social learning and advanced forms of reasoning (see, e.g., Liu, Neubig, & Andreas, 2024, on inductive and abductive reasoning in LLMs) within agent-based simulations.[11]

[11] I am greatly indebted to Rainer Hegselmann, Christopher von Bülow, and two anonymous referees for valuable comments on previous versions of this paper.

References

Adam, D. (2020). The simulations driving the world's response to COVID-19. *Nature*, *580*, 316–318.

Battleday, R. M., Peterson, J. C., & Griffiths, T. L. (2021). From convolutional neural networks to models of higher-level cognition (and back again). *Annals of the New York Academy of Sciences*, *1505*, 55–78.

Berlin, B., & Kay, P. (1969). *Basic color terms*. CSLI Publications.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*, 65–98.

Borg, A., Frey, D., Šešelja, D., & Straßer, C. (2019). Theory-choice, transient diversity and the efficiency of scientific inquiry. *European Journal for Philosophy of Science*, *9*, 26. https://doi.org/10.1007/s13194-019-0249-5

Bouckaert, R. R. (2006). Efficient AUC learning curve calculation. In A. Sattar & B.-H. Kang (Eds.), *AI 2006: Advances in artificial intelligence* (pp. 181–191). Springer.

Buckner, C. (2018). Empiricism without magic: Transformational abstraction in deep convolutional neural networks. *Synthese*, *195*, 5339–5372.

Buckner, C. (2023). *From deep learning to rational machines*. Oxford University Press.

Buckner, C., & Garson, J. (2018). Connectionism and post-connectionist models. In M. Sprevak & M. Colombo (Eds.), *The Routledge handbook of the computational mind* (pp. 76–91). Routledge.

Caucheteux, C., & King, J. R. (2022). Brains and algorithms partially converge in natural language processing. *Communications Biology*, *5*, 134. https://doi.org/10.1038/s42003-022-03036-1

Chen, Z., Lu, Y., Yang, W., Xuan, Q., & Yang, X. (2021). Graph-based similarity of neural network representations. *arXiv*. https://arxiv.org/abs/2111.11165

Claidière, N., Jraissati, Y., & Chevallier, C. (2008). A colour sorting task reveals the limits of the universalist/relativist dichotomy: Colour categories can be both language specific and perceptual. *Journal of Cognition and Culture*, *8*, 211–233.

Cook, R. S., Kay, P., & Regier, T. (2005). The world color survey database: History and use. In H. Cohen & C. Lefebvre (Eds.), *Handbook of categorization in cognitive science* (pp. 223–242). Elsevier.

Cristelli, M. (2014). *Complexity in financial markets*. Springer.

Crosscombe, M., & Lawry, J. (2016). A model of multi-agent consensus for vague and uncertain beliefs. *Adaptive Behavior*, *24*, 249–260.

Deffuant, G., Neau, D., Amblard, F., & Weisbuch, G. (2000). Mixing beliefs among interacting agents. *Advances in Complex Systems*, *3*, 87–98.

Dietrich, F., & List, C. (2016). Probabilistic Opinion Pooling. In A. Hájek & C. Hitchcock (Eds.), *The Oxford handbook of probability and philosophy* (pp. 519–541). Oxford University Press.

Dittmer, J. C. (2001). Consensus formation under bounded confidence. *Nonlinear Analysis*, *7*, 4615–21.

Douven, I. (2010). Simulating peer disagreements. *Studies in History and Philosophy of Science Part A*, *41*(2), 148–57.

Douven, I. (2017). Clustering colors. *Cognitive Systems Research*, *45*, 70–81.

Douven, I. (2019). Optimizing group learning: An evolutionary computing approach. *Artificial Intelligence*, *275*, 235–51.

Douven, I. (2024a). The learnability of natural concepts. *Mind and Language*, in press.

Douven, I. (2024b). Pandemics and flexible lockdowns: In praise of agent-based modeling. *European Journal for Philosophy of Science*, in press.

Douven, I. (2024c). The role of naturalness in concept learning: A computational study. *Minds and Machines*, in press. https://doi.org/10.1007/s11023-023-09652-y

Douven, I., & Hegselmann, R. (2021). Mis- and disinformation in a bounded confidence model. *Artificial Intelligence*, *291*, 103415.

Douven, I., & Hegselmann, R. (2022). Network effects in a bounded confidence model. *Studies in History and Philosophy of Science Part A*, *94*, 56–71.

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., & Mordatch, I. (2023). Improving factuality and reasoning in language models through multiagent debate. *arXiv*. https://doi.org/10.48550/arXiv.2305.14325

Epstein, E. S. (1969). A scoring system for probability forecasts of ranked categories. *Journal of Applied Meteorology*, *8*, 985–987.

Eysenbach, G., Powell, J., Englesakis, M. F., Rizo, C., & Stern, A. (2004). Health related virtual communities and electronic support groups: Systematic review of the effects of online peer to peer interactions. *British Medical Journal*, *328*, 1166.

Fairchild, M. D. (2013). *Color appearance models*. Wiley.

Frey, D., & Šešelja, D. (2018). What is the epistemic function of highly idealized agent-based models of scientific inquiry? *Philosophy of the Social Sciences*, *48*, 407–33.

Frey, D., & Šešelja, D. (2020). Robustness and idealizations in agent-based models of scientific interaction. *British Journal for the Philosophy of Science*, *71*, 1411–37.

Friedkin, N. E., & Johnsen, E. C. (1990). Social influence and opinions. *Journal of Mathematical Sociology*, *15*, 193–206.

Glass, C., & Glass, D. H. (2021). Opinion dynamics of social learning with a conflicting source. *Physica A: Statistical Mechanics and its Applications*, *563*, 125480. https://doi.org/10.1016/j.physa.2020.125480

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323.

Goldstein, A., Zada, Z., Buchnik, E., Schain, M., Price, A., Aubrey, B., Nastase, S. A., Feder, A., Emanuel, D., Cohen, A., Jansen, A., Gazula, H., Choe, G., Rao, A., Kim, S. C., Casto, C., Fanda, L., Doyle, W., Friedman, D., … Hasson, U. (2021). Thinking ahead: Spontaneous prediction in context as a keystone of language in humans and machines. *bioRxiv*. https://doi.org/10.1101/2020.12.02.403477

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hegselmann, R. (2023). Bounded confidence revisited: What we overlooked, underestimated, and got wrong. *Journal of Artificial Societies and Social Simulation*, *26*, 11. https://doi.org/10.18564/jasss.5257

Hegselmann, R., König, S., Kurz, S., Niemann, C., & Rambau, J. (2015). Optimal opinion control: The campaign problem. *Journal of Artificial Societies and Social Simulation*, *18*. http://jasss.soc.surrey.ac.uk/18/3/18.html

Hegselmann, R., & Krause, U. (2002). Opinion dynamics and bounded confidence: Models, analysis, and simulations. *Journal of Artificial Societies and Social Simulation*, *5*. http://jasss.soc.surrey.ac.uk/5/3/2.html

Hegselmann, R., & Krause, U. (2006). Truth and cognitive division of labor: First steps towards a computer aided social epistemology. *Journal of Artificial Societies and Social Simulation*, *9*. http://jasss.soc.surrey.ac.uk/9/3/10.html

Hegselmann, R., & Krause, U. (2015). Opinion dynamics under the influence of radical groups, charismatic leaders, and other constant signals: A simple unifying model. *Networks and Heterogeneous Media*, *10*, 477–509.

Hegselmann, R., & Krause, U. (2019). Consensus and fragmentation of opinions with a focus on bounded confidence. *American Mathematical Monthly*, *126*, 700–716.

Hoffman, P., McClelland, J. L., & Lambon Ralph, M. A. (2018). Concepts, control, and context: A connectionist account of normal and disordered semantic cognition. *Psychological Review*, *125*, 293–328.

Hosseini, E. A., Schrimpf, M., Zhang, Y., Bowman, S., Zaslavsky, N., & Fedorenko, E. (2023). Artificial neural network language models predict human brain responses to language even after a developmentally realistic amount of training. *bioRxiv*. https://doi.org/10.1101/2022.10.04.510681

Huang, A. C. W. (2023). Track records: A cautionary tale. *British Journal for the Philosophy of Science*, in press. https://doi.org/10.1086/728459

Jacobmeier, D. (2004). Multidimensional consensus model on a Barabási–Albert network. *International Journal of Modern Physics C*, *16*, 633–646.

Kummerfeld, E., & Zollman, K. J. S. (2016). Conservatism and the scientific state of nature. *British Journal for the Philosophy of Science*, *82*, 956–968.

Laninga-Wijnen, L., & Veenstra, R. (2023). Peer similarity in adolescent social networks: Types of selection and influence, and factors contributing to openness to peer influence. In B. Halpern-Felsher (Ed.), *Encyclopedia of child and adolescent health* (pp. 196–206). Academic Press.

Liu, E., Neubig, G., & Andreas, J. (2024). An incomplete loop: Deductive, inductive, and abductive learning in large language models. https://doi.org/10.48550/arXiv.2404.03028

Lorenz, J. (2008). Fostering consensus in multidimensional continuous opinion dynamics under bounded confidence. In D. Helbing (Ed.), *Managing complexity: Insights, concepts, applications* (pp. 321–334). Springer.

Lorig, F., Johansson, E., & Davidsson, P. (2021). Agent-based social simulation of the Covid-19 pandemic: A systematic review. *Journal of Artificial Societies and Social Simulation*, *24*. https://doi.org/http://jasss.soc.surrey.ac.uk/24/3/5.html

O'Connor, C., & Weatherall, J. O. (2019). *The misinformation age: How false beliefs spread*. Yale University Press.

Olsson, E. J. (2013). A Bayesian simulation model of group deliberation and polarization. In F. Zenker (Ed.), *Bayesian argumentation* (pp. 113–133). Springer.

Olsson, E. J., & Vallinder, A. (2013). Norms of assertion and communication in social networks. *Synthese*, *190*, 2557–2571.

Pfitzer, D., Leibbrandt, R., & Powers, D. (2009). Characterization and evaluation of similarity measures of pairs of clusterings. *Knowledge and Information Systems*, *19*, 361–394.

Pluchino, A., Latora, V., & Rapisarda, A. (2006). Compromise and synchronization in opinion dynamics. *European Physical Journal B*, *50*, 169–176.

Rosenstock, S., O'Connor, C., & Bruner, J. (2017). In epistemic networks, is less really more? *Philosophy of Science*, *84*, 234–252.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Schelling, T. C. (1971). Dynamic models of segregation. *Journal of Mathematical Sociology*, *1*, 143–186.

Šešelja, D. (2019). Some lessons from simulations of scientific disagreements. *Synthese*, *198*, 6143–6158.

Thicke, M. (2020). Evaluating formal models of science. *Journal for General Philosophy of Science*, *51*, 315–335.

Tsai, M.-H., Ho, C.-H., & Lin, C.-J. (2010). Active learning strategies using SVMs. *The 2010 International Joint Conference on Neural Networks*, 1–8. https://doi.org/10.1109/IJCNN.2010.5596668

Zollman, K. J. S. (2007). The communication structure of epistemic communities. *Philosophy of Science*, *74*, 574–587.

Zollman, K. J. S. (2010). The epistemic benefit of transient diversity. *Erkenntnis*, *72*, 17–35.