Developing an Agent-Based Model of the Banking System with Deep Reinforcement Learning

Georgi Demirev[a]

[a]Bulgarian National Bank Scholarship Recipient

This manuscript was compiled on October 12, 2018

This article presents a macroeconomic model of bank intermediation based on multi agent reinforcement learning. The model consists of heterogeneous households, banks, and firms, each acting to maximize individual utility. Agents acquire their policy autonomously via double Q learning. Simulating the model yields endogenous credit-driven business cycles. Two experiments are carried out in the simulated environment: the first one introduces deposit guarantees, which leads to a lower average number of deposit withdrawals; the second one varies the required reserve ratio for banks, and reveals that at low levels of required reserves banks increase their lending. Several suggestions for future research on reinforcement-learning-based economic models are presented.

Agent Based Models | Reinforcement Learning

Contemporaneous macroeconomic models tend to predominantly be dynamic stochastic general equilibrium (DSGE) models, following the pioneering work of Kydland and Prescott (1). Yet, despite their widespread academic success, DSGE models have recently come under heavy criticism in the face of the perceived failure to predict and explain the 2008 financial crisis (2).

One of the major points of criticism is the over-dependence of DSGE models on exogenous shocks - be it in productivity, preference, capital formation, etc (3). In this view, the reliance on external (unexplained) forces greatly diminishes the models' scientific usefulness, as the major driving forces are often attributed to mere chance.

A second line of criticism points to the critical role that equilibrium solution concepts play in such models. On one hand general equilibrium theory has been largely abandoned by microeconomists, who found it to be neither stable nor unique (4). On the other, as shown by Papadimitriou, finding a Brouwer fixed point (and hence a solution to GE exchange problem) is a likely non-polynomial time task (5). Coupled with the computational complexity of dynamic programming problems with complex multi-dimensional state dynamics, the standard DSGE model imposes rather unrealistic requirements on agents' cognitive prowess. This line of attack is often taken up by proponents of bounded rationality models.

Several alternatives to DSGEs have been put forth, one of the more promising of which being agent-based models (ABM). ABMs shun equilibrium concepts and instead try to simulate the individual behavior of many (possibly heterogeneous) agents and their interactions (6). The modellers then study the resulting simulated economy to arrive at conclusions about the real world.

Despite their promising premise, ABMs too have come under fair amount of criticism. They often rely on ad-hoc rules-of-thumb to explain individual behavior (7). In contrast traditional macro models attempt to base themselves into the "micro foundations" of the individual agent's optimization. Furthermore, a typical agent-based simulation may have myriad of parameters, making it hard to analyze coherently (8). This arbitrary nature of ABMs is one of the main reasons for them to remain outside the academic mainstream.

In this paper I try to unify the agent-based approach of simulating all interactions between agents with the individual optimization assumption of DSGE models. I attempt to do this by employing concepts and techniques from reinforcement learning (RL) - the sub-field of artificial intelligence that deals with adaptive decision making.

In order to do so I develop a model of financial intermediation with three types of agents - households, firms, and banks. All agents in the economy maximize individual utility as in a DSGE model. However instead of assuming that agents behave as per the solution of a dynamic programming problem, they are endowed with a learning mechanism that tries to approximate this optimal solution. This is done by a method known as double Q-learning with a deep neural network value function approximator.

This is not the first paper to employ reinforcement learning in Economics. Famously (9) uses a simple reinforcement learning framework to model repeated games and (10) expand on this by comparing the implications of a reinforcement learning model to a number of experiments on games with mixed strategy Nash equilibria. More recently (11) build a reinforcement learning model of a simple asymmetric information game. There are also plenty of prior work on ABMs of the banking sector - (12), (13), (14), just to name a few.

However this paper is novel, as previous RL economic models, to the best of my knowledge, did not use value function approximation, and previous bank ABMs tend to have little in terms of agent learning mechanisms and rely on rules-of-thumb. Note a clear distinction from what is typically referred as "models of learning" in economics: while the economic literature has most often focused on

[2]For correspondence: georgi.demirevbarcelonagse.eu

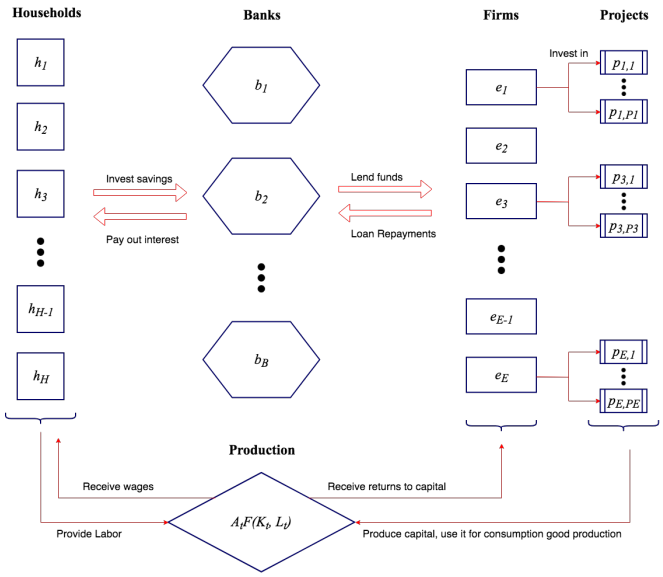

Fig. 1. Circular flow of the economy

beliefs and predictions - i.e. learning to predict (15), reinforcement learning instead focuses on approximate optimization in the face of a high dimensional state space (i.e. learning to optimize).

This paper borrows heavily concepts and algorithms from recent advances in computer science (16), (17). It aims to employ these novel methods in a self-contained agent-based macroeconomic model. The main objective of the paper is to explore the usefulness of deep reinforcement learning in economics. As such the focus is on the structure of the model itself. I do however provide two examples of how economic experiments can be conducted in this framework in order to derive insights. The examples are included mainly to showcase the flexibility and usefulness of the model, and not to serve as basis for actual policy recommendations.

The remainder of this article is structured as follows: part 2 describes the agent based model and the interaction rules of the model economy. Part 3 gives more details about the reinforcement learning mechanism via which agents make decisions. Part 4 examines the emerging properties of the simulated economy. Part 5 details the result of two policy experiments conducted in this model set-up. Part 6 presents some key issues that need to be solved in future research on reinforcement-learning-based economic models. Part 7 concludes.

Model Description

The model economy consists of three types of agents - banks, firms, and households. There are two goods - a consumption good and a capital good. Time passes in discrete periods.

Households. Households provide labor for production and receive wages (in terms of the consumption good). They can either invest their wage or deposit it in a bank in return for interest (households possess no storage technology). Upon consumption households receive utility from a simple logarithmic utility function. Each household is restricted to hold deposits in only a single bank, but at each period households are free to withdraw their deposits and move them to another bank (or consume them).

Thus, the household i's optimization problem at period $t = 0$ is given by:

$$\max_{w_{i,t}, s_{i,t}} \sum_t \beta_H^t \mathbf{E}\left[\log c_h(w_{i,t}, s_{i,t}, \mathbf{A_{-i}}, \mathbf{S_t})\right] \qquad [1]$$

Where $w_{i,t}$ and $s_{i,t}$ are the decision vectors of the household. $w_{i,t}$ denotes whether or not the household withdraws its deposits, and $s_{i,t}$ denoting whether the household consumes it's cash or deposits it at one of the banks. The outcome of the period is determined by the function c_h and depends on the individual household's decision, as well as the actions of all other agents (collected in $\mathbf{A_{-i}}$) and the state of the economy (represented by $\mathbf{S_t}$). The households discount future consumption at β_H.

Firms. Firms provide capital goods for production and receive a rental rate in terms of the consumption good. Capital is then decreased via a depreciation rate, which for all results in the paper was set to 1. Firms have access to a production technology in the form of 'projects'. Each project costs a certain amount of the consumption good to set up and then brings in a random stream of the capital good for a fixed period of time. A similar production technology was used to generate endogenous cycles in (18)

In this model, firms have access to a new random project at the beginning of each period. They can either discard the project, invest their own cash in it, or take out a loan to finance it. At each time point there is a probability that the project defaults and stops producing capital (any loans taken for the project however remain outstanding until repaid). Any leftover cash at the end of the period is consumed by the firm's management and utility is received. When firms cannot meet their loan repayment obligations they default - all their projects are terminated at a per-project termination income, and the proceedings go the firm's creditors. The firm then begins next period on a clean slate.

We can write out the objective function of firm j in a similar manner to the household:

$$\max_{e_{j,t}, b_{j,t}} \sum_t \beta_F^t \mathbf{E}\left[\log c_f(e_{j,t}, d_{j,t}, \mathbf{A_{-i}}, \mathbf{S_t})\right] \qquad [2]$$

where the decisions of the firm at each period are whether to pursue a project opportunity or discard it ($e_{j,t}$) and whether to borrow and from which bank ($b_{j,t}$). Firms' utility is also logarithmic. Future utility is discounted at

some rate β_F. Here I have chosen to model the firm's management decision to maximize their own consumption, but alternatively the firm's problem can be re-framed as maximizing profits or equity with slight modifications to the objective function.

Banks. Banks take in deposits and give out loans in exchange for interest. Each period banks set up the interest rates on loans and deposit. It is assumed that interest prices are rigid in the sense that a bank can only move the rate up or down by a fixed increment per period (the reasons for this are explained later on). Banks cannot refuse to take in a deposit if requested by a household, but can refuse to give out a loan. For each loan the expected sum of future capital inflows is calculated and then compared to the initial investment required. This 'capital per dollar' ratio is then compared to the bank's cut-off value and the loan is accepted only if it is above it. The cut-off is also controlled by the bank and can be moved in a similar manner as the interest rate. At the end of each period the bank receives utility equal to an exogenous percentage of the bank's capital (one can think of the manager receiving a bonus, or the shareholders receiving a dividend).

Similar to firms, the model can be modified slightly so the banks problem is stated in terms of profit or equity maximization instead of dividend maximization. Nonetheless in the current form the bank problem takes the form (for some bank k):

$$\max_{l_{k,t},d_{k,t},a_{k,t}} \sum_t \beta_B^t \mathbf{E}\left[\log c_b(l_{k,t}, d_{k,t}, a_{k,t}, \mathbf{A_{-i}}, \mathbf{S_t})\right] \quad [3]$$

Where the control vectors are $l_{k,t}$ - the interest rate on loans at period t, $d_{k,t}$ - the interest rate on deposits, and $a_{k,t}$ - the approval 'capital-per-dollar' cutoff. The reward for the bank is given by the function c_b, and as before depends on the bank's actions, the other agents' actions, and the state of the economy.

Interactions. Each agent begins with an initial endowment of the consumption good (and a small endowment of the capital good for firms). In the case of banks the initial endowment represents their initial reserves. Each banks balance sheet consists of four items - cash reserves, outstanding loans, deposits, and equity. Equity is defined as the difference between assets (reserves and loans) and liabilities (deposits). Banks are subject to a reserve requirement - their reserves are not allowed to fall below a certain ratio of the deposits, and a capital requirement - their capital cannot fall below a certain ratio of the loans. If at any point a bank doesn't meet any of the two requirements it is not allowed to give out additional loans or receive utility from dividends. If at any point a bank's equity becomes negative it goes bankrupt - all depositors lose their savings, and all debtors don't have to pay back their loans. A bankrupt bank begins next period with a reset balance (one can think of a government bailout), but is not allowed to give out dividends/bonuses for a fixed period of time (one can think of the bank being under supervision).

The economy's circular flow is illustrated in figure 1. A given period consists of the following steps (in that order):

1. production of the consumption good takes place. Wages and rental rates are paid out
2. potential bank bankruptcies are resolved
3. banks approve or reject loan applications. Approved projects are initiated
4. banks adjust their interest rate and capital-per-dollar cut-off rate
5. banks give out dividends to shareholders (receive utility)
6. firms receive capital from outstanding projects and draw a new project opportunity
7. firms make loan repayments
8. firms decide whether to invest in the new project (and whether to apply for a loan)
9. firms consume leftover cash
10. households decide whether to keep their deposits and deposit their wage, switch banks, or withdraw and consume their savings.
11. households receive interest on deposits

The production sector in this model is deliberately simplistic. Both factors are supplied inelastically, and are paid per their marginal costs. This is intentional, since the paper focuses on the financial sector.

For simplicity, each firm's projects are drawn from the same distribution over project characteristics. In that case there are no differences in the productive capacity of each firm. However at any point there will be difference in each firm's collection of active projects and hence the firm's risk profile.

In the current setup banks can only approve or reject loans based on the characteristics of the project. More specifically banks calculate the aforementioned "capital-to-dollar" ratio by first calculating the expected stream of the capital good:

$$\sum_{j=1}^{J} k_j(1-p)^j + K(1-p)^J \quad [4]$$

and then dividing it by the initial investment amount. Here J is the total duration of the project, k is the mean

per-period flow of capital, p is the per-period default probability, and K is the terminal income received in the last period. By introducing terminal income K I allow for two types of projects - ones that distribute their capital output throughout the project life period (K relatively small compared to k_js), and ones that give out most of their capital output in a single period (higher K relative to k_j).

Note that in this approach the bank cannot screen riskier firms (it can only set the cut-off capital-to-dollar ratio). This can be improved upon in a future implementation. Note that in principle the expected stream of capital can be discounted by the rate of return and converted into units of the consumption good. This can also be tried in future versions, but in principle, since the bank takes into account the rate of return when deciding its policy, it should be possible to internalize it in the policy function.

Each project has different values for all variables in the capital-to-dollar formula, as well as different per-period standard deviations of capital income, and different liquidation values. An example of a project is given in the appendix.

Notice also that both firms and households are somewhat constrained in their actions. Firms can choose to finance a project either only with cash, or only with loans. Households on the other hand must always have all their savings in a single bank, and can either deposits all their cash, or consume it (but not consume part and deposit the rest). These limitations are somewhat arbitrary and can also be relaxed in future work. Here they are introduced mainly to ease implementation.

For simplicity it is assumed that all agents share the same (logarithmic) utility function, and have the same per-period discount factor (equal to 0.99).

Decision Making

Agents in this model make decisions following a reinforcement learning training strategy. The environment is designed in such a way, that whenever an agent has to make a decision, they must choose one of several discrete options:

Households must choose whether to keep their deposit and consume their cash at hand, whether to deposit their cash at the same bank, or whether to withdraw the existing deposit and put it along with the cash in one of the other (countably many) banks; Firms choose whether to forgo a project, whether to invest cash in it, or whether to apply for funding to one of the (countably many) banks. Banks decide whether to increase by a fixed increment, decrease by a fixed increment, or keep constant the interest rates on loans and deposits, as well as the project profitability cut-off.

In the concrete implementation shown in the text there are ten banks, so this amounts to 13 possible mutually-exclusive decisions for each household per period, 12 for each firm, and 27 (3 variables to adjust with 3 degrees of freedom each) possible decisions for each bank.

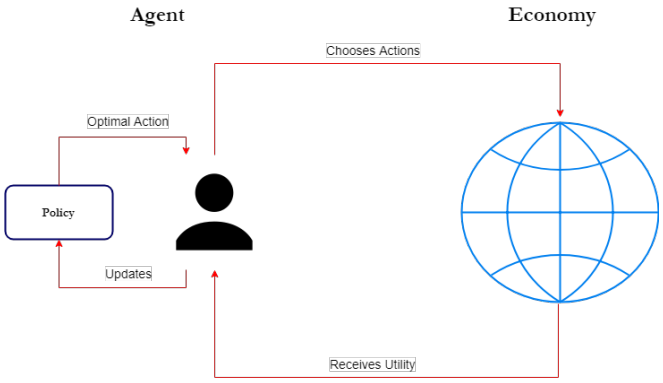

Fig. 2. Agent interacting with the economy

Each period before making a decision, agents observe the economy's (and their own) state. The economy is fully described only by the state variables of all agents included in it (which in the models presented in the text consists of 500 households and 100 firms on top of the 10 banks), only a portion of the state is observable by each agent. This portion includes the balances of all banks (one can think of them as public companies), the individual agent's own balance (be it firm, household, or bank), and some macroeconomic variables (the level of output, the wage, the rate of return on capital). This 'observation set' numbers around a 100 variables per agent. I will denote this observation vector (which represents the state relevant to the decision maker) by s [*].

The goal of each agent in the economy is to learn a mapping between states (observation sets) s and actions a (the action set is discrete), which maximizes their infinite sum of discounted future utilities. Call this mapping (policy) π^*.

To this end the agents in this types of model begin in a 'blank' state, where they don't know anything about optimal behavior. The economy than goes through several 'training' iterations which aim to let agents improve their policies as to resemble a good enough approximation of π^*

Q Learning. The fundamental learning algorithm that is to be employed by the agents in this model is known as Q-Learning. The Q in the name comes from the traditional notation for an action-value function in dynamic programming, where $Q^\pi(s, a)$ usually denotes the expected discounted stream of future rewards (i.e. utility), following state s in which action a is taken, and policy π is followed afterwards. Formally:

[*] $s \in \mathbf{S_t}$ with $\mathbf{S_t}$ being the true aggregate state

$$Q^{\pi}(s,a) = u_0(s,a) + \beta E_{s^t \sim \pi}\left[\sum_{t=1}^{\infty} u_t(s^t, \pi(s^t))\right] = u_0(s,a) + \beta E_{s^t \sim \pi}\left[Q^{\pi}(s', \pi(s'))\right] \quad [5]$$

Where $\pi(s)$ is the action prescribed by the policy π at state s, u_t is the utility at period t, β is the discount factor, $s^t \sim \pi$ denotes the sequence of future states under the policy, and s' denotes the very next state.

Note the similarity with the better known (in Economics) concept of a state value $V^{\pi}(s)$. In fact it holds that:

$$V^{\pi}(s) = Q^{\pi}(s, \pi(s)) \quad [6]$$

Q-learning relates to a general algorithm of finding an approximate solution to the dynamic programming program, given by the Bellman equation:

$$Q^*(s,a) = max_a u(s,a) + \beta E_{s,a} V^*(s') \quad [7]$$

Where

$$V^*(s') = max_{a'} Q(s', a') \quad [8]$$

With a' being any action available at s'.

Whereas in dynamic programming the problem can be solved leveraging the knowledge of system dynamics (e.g. by general policy-value iteration), this is unfeasible in our framework (as it is in most real-world applications), since the dynamics of the system are complex and generally unknown by the agent. Thus the agent has to rely on approximate solutions. A comprehensive reference can be found in (17).

Some of the practical aspects of the implementation are influenced by a seminal paper by Mnih (16), where deep Q networks (a main tool of this paper) were introduced. One of the few papers so far to apply deep Q networks to an economic problem is (19) where they are used to model sequential social dilemmas.

Function Approximation. As already mentioned, the agents in the model are trying to approximate Q^*. To that end define the differentiable function Q_ϕ (parametrized by ϕ) as the agent's current approximation to Q^*. Every time the agent is called to take an action in state s then, they choose $arg\max_a Q_\phi(s,a)$ [†]. As the agent interacts with the environment, they learn how to update the function Q_ϕ, so that it approximates Q^* better and better.

To describe the algorithm to do this, imagine that the agent interacts with the environment for N periods, following some policy. Let s_i, a_i for $i = 1, 2, ..N$ denote the states encountered, and the actions taken. Define the Bellman error (λ) at state s_i, after action a_i as:

$$\lambda_i = Q_\phi(s_i, a_i) - u(s_i, a_i) - \beta V_\phi(s_i', a_i') = Q_\phi(s_i, a_i) - u(s_i, a_i) - \beta \max_{a_i'} Q_\phi(s_i', a_i') \quad [9]$$

Looking back at the definition of Q_ϕ it is evident, that the Bellman error measures the 'surprise' of the agent - the difference between the expected sum of feature utility before the action is taken, and the expected sum of future utility ex-post (when the immediate utility and the new state s_i' are observed). We want to update the parameters such as to minimize this 'surprise' (a.k.a to make the agent better at forecasting the consequences of each action). A simple algorithm to do this is:

1. Collect data $(s_i, a_i, s_i', u(s_i, a_i))$
2. Set $y_i = u(s_i, a_i) + \beta \max_{a'} Q_\phi(s_i', a_i')$
3. $\phi \leftarrow arg\min_\phi \sum_i ||Q_\phi(s_i, a_i) - y_i||^2$

Where step 2. and 3. are repeated K times (think of value function iteration with sampling). This is little more than fitting a regression to y_i as defined above (the $\leftarrow$ operator signifies reassignment of the parameter).

The algorithm described above will achieve its goal, however we want to be able to make updates to the approximate Q function (and therefore to the policy function π) online. That is, after each interaction with the environment, so that our agents are adaptable to changes in the economy. We can do this by taking a gradient descent step for Q_ϕ after each period. Letting the learning rate of the gradient descent be denoted as γ the algorithm becomes:

1. Take an action a_i at state S_i and record $S_i, a_i, S_i', u(S_i, a_i)$
2. Calculate the Bellman error $\lambda_i = Q_\phi(S_i, a_i) - u(S_i, a_i) - \beta \max_{a'} Q_\phi(S_i', a_i')$
3. Set $\phi \leftarrow \phi - \gamma \nabla_\phi Q_\phi(s_i, a_i)\lambda_i$

Approximation with neural networks. So far we haven't mentioned anything about the functional form of Q_ϕ. The simple possible approximator is a linear one $Q(s,a) = \phi' s$. The linear approximator is simple and easy to compute. Furthermore we know comparatively more about its convergence properties (20).

However, the flexibility of a linear function is quite limited, and in many practical applications, a slightly more complex functional form is chosen despite less developed convergence theory (21). This paper does the same. I take Q_ϕ to be a simple feed forward neural network of the form:

[†] to improve learning speed agents actually follow an epsilon-greedy strategy, which means that with probability $1-\epsilon$ they choose $arg\max_a Q_\phi(s,a)$, and with probability ϵ they explore a random action. The model was trained with an exponentially decreasing exploration rate

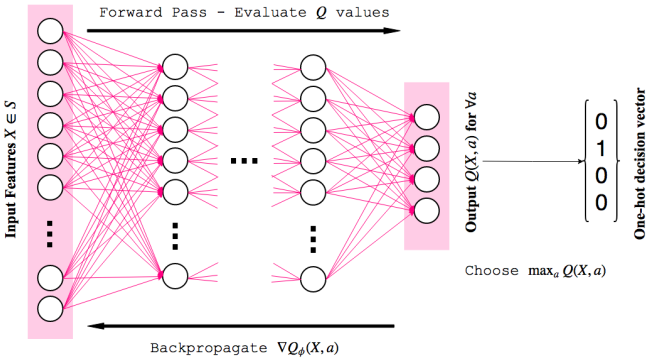

Fig. 3. A representation of a Q-Network

$$Q(s_i, \mathbf{a_i}) = \sum_{l=1}^{L} \delta(\phi_l' s_l) \qquad [10]$$

Where L is the total number of layers, δ is a non-linear function (e.g. a sigmoid, or a rectified linear unit), ϕ_l are the parameters at layer l, and $s_1 = [1, S_i]^T$. This is illustrated in figure 3.

To update the weights of the network, we just use backpropagation, starting with $Q_\phi(s_i, a_i)\lambda_i$ at the outermost layer. As illustrated by the figure, we can convert the output of the network directly to actions as per the definitions of the previous section. This amounts to just setting the element in the action vector, corresponding to the highest Q value to 1, and the others to 0.

Double Q Learning. The above section describes the vanilla version of Deep Q-Learning. However, there are some known problems with this algorithm (22). Namely, the samples of state-utility sequences that the agent encounters while interacting with the environment are highly correlated (since they are taken from the almost the same policy). Furthermore, note that we calculate y_i in the algorithm above **before** taking the gradient of the Q network - in essence we are optimizing to match a moving target. Both these problems hinder the performance of deep Q-networks.

To fix this well-known issues I introduce two ideas, first encountered in (22): the replay buffer, and the target network. The replay buffer is nothing more than a collection of past interactions (think of it as memory). When the network is updated we use not only the last action, but a random sample from the replay buffer - that way the Bellman errors will not be (as) correlated. We call the replay buffer $\mathcal{B}$

The target network fixes the second issue, by creating two networks - one for choosing actions (Q_ϕ), and another for evaluating them ($Q_{\phi'}$). This way when we update the parameters of the network that does the choosing (Q_ϕ), the target is actually fixed (i.e. the network that evaluates them $Q_{\phi'}$). After a certain number of periods the current choosing network becomes the new target.

The algorithm described above is known as "Double Q Learning". For more details on it the reader is referred to (22). To recap the algorithm takes the form:

1. Take action a_i, record $(S_i, a_i, S_i', u(S_i, a_i))$. Add it to $\mathcal{B}$
2. Sample mini-batch $\{S_j, a_j, S_j', u(S_j, a_j)\}_{j=1}^N$ from $\mathcal{B}$
3. Compute the Bellman errors $\lambda_j = Q_\phi(S_j, a_j) - u(S_j, a_j) - \beta Q_{\phi'}(S_j', arg\max_{a_j'} Q_\phi(S_j', a_j'))$
4. Update $\phi \leftarrow \phi - \gamma \sum_j \nabla_\phi Q_\phi(S_j, a_j)\lambda_j$
5. Every T steps set $\phi' \leftarrow \phi$

This final learning mechanism is the one implemented for all agents in the model economy - households, firms, and banks.

In principle the 'purest' agent-based model will have one neural network per each simulated agent. In this model however there is one network per **agent type**. This means there is a single neural network that does the Q function approximation for every household (and another one for banks and another one for firms), but the inputs to the function are different, depending on the current state of the individual simulated agent.

The reason for this choice is mainly ease of computation [‡]. However there is also a nice conceptual justification. Q learning does not distinguish between off-policy and on-policy training (17). This means that agents that employ this algorithm learn equally well from their own experience and from the experience of others. And since in this model we do not assume that there is private information, all agents of a single type can observe the actions and outcomes of all other agents of that type.

All this goes to say is that if I had implemented a version of the model where each single agent has their own independently trained Q-network, I would have to allow all agents of the same type access to the same replay buffer (since everyone also observe others' actions). If we have hundreds of neural networks training from the same data, then the only differences in weights will be due to mere chance in sampling the mini batches and should disappear asymptotically. Thus having a single network per agent type emulates the asymptotic behavior of having one network per individual agent [§].

Implementation. The model as described above was implemented using open source tools - namely the R programming language and Python. The implementation uses an object-oriented paradigm, with the economy being the single most important class, encompassing the objects for the different agent types.

PyTorch was used for implementing the function approximators. Each neural net consists of an input layer,

[‡] the model was trained on a personal laptop with no GPU

[§] This is true for Q-learning but not in general.

two hidden layers, one sigmoid activation layer (after the first hidden layer), one rectified linear unit layer (after the second hidden layer), and an output layer. The input layer and the two hidden layer's outputs were batch-normalized. The learning rate was set to 0.001 and the weight decay to 0.2. A batch size of 512 was chosen and a replay buffer size of 2000. All these values as well as the network topology can in principle be subject to a more sophisticated parameter tuning process.

The (research quality) code is made available in the project's repository [¶].

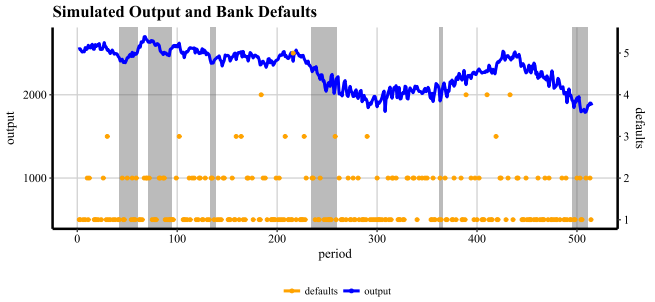

Fig. 4. Simulated Output

Emerging Dynamics

In order to implement the economic environment described above I created a simulated economy, consisting of 10 banks, 100 firms, and 500 households. The exact starting parameters chosen are available in the appendix. The agents in the economy were then left to interact with the environment (and each other) for around 35000 periods. The initial policies of each agent type were random. However as the simulation proceeded they were progressively improved using the Q-Learning algorithm described above. The final policies after this training period were then used in the simulations discussed below.

While in theory agents are optimizing an infinite sum of discounted returns, in practice the training was set up so, that in any given period there was a $1:1000$ chance of the economy resetting to its initial state. In this section I detail the results of the simulated periods following the last such economy reset. This 'iteration' consisted of 615 periods, the first 100 of which were discarded to reduce the dependence on initial conditions.

Output. Figure 4 shows the resulting dynamics of total economic output. Few things are immediately noticeable. First, output seems to oscillate around a region between 2000 and 2500 (with a mean of 2295), sometimes going up or down, but never drifting away completely. This is desirable, since in the absence of technological change, we wouldn't expect to see any long-term trends in output (a regression of output against time actually gives a significant negative estimate, but it's magnitude is less than 0.05 percent of the mean and becomes insignificant if we account for capital and bank variables).

The second noticeable feature is the graph's cyclicality. Output is clearly non-stationary (confirmed with a Ljung-Box test), and has a high-degree of persistence (see the auto-correlation coefficient in table 1).

More importantly, the economy exhibits expansionary and contractionary periods. On top of that, similar to the real world, cycles seem to be quite irregular. The shaded areas on the graph depict periods of recession of the simulated economy (recession starts are defined as 4 consecutive periods of output decline, and recession ends are defined as 4 consecutive periods of growth). The shortest recorded recession lasts only 6 periods, while the longest spans 26. Compare this with NBER data on American recessions and we can see similar discrepancies in length.

In the simulated economy we can observe both rapid (period 61), and gradual (periods 300-400) expansions. The same is true for contractions - the recessions starting at periods 71 and 234 are much more steep compared to the gradual decline starting around period 420. In the real world the business cycle is asymmetric with expansions being much more gradual then contractions (23).

Nevertheless, the emergence of endogenous business cycles in the model is an interesting fact by itself. It seems that the business cycle is driven mainly through credit availability (see figure 5).

Finally, figure 4 depicts the number of bank failures per period (the orange dots). We can see that in this economy bank failures are a common fact of life, occurring almost all of the time. In addition it does not seem that bank failures are more or less frequent during recessions than during expansions.

Co-movements. If we move to examine the other time series generated by the model, we would see another key feature of the simulated environment - strong co-movements of economic variables. This is exhibited in figure 5

We notice for example that loans tend be strongly tied to output - a jump in loans is almost always associated by a jump in output. Furthermore it seems that the movement in loans leads the movement in output. A similar feature is also exhibited by the plot of output against investment or deposits, although to a lesser extend (see the correlations in table 1).

Interestingly consumption seems to be almost a constant fraction of output - it varies between 70% and 80% of output in 80 percent of the simulated periods (and is never lower than 62% or higher than 92%). This is somewhat expected, since the utility function is logarithmic, and hence in standard economic models the optimal solution to the consumer problem is to consume a fixed proportion of income. This lends evidence that agents in

[¶] https://github.com/demirev/banks_rl

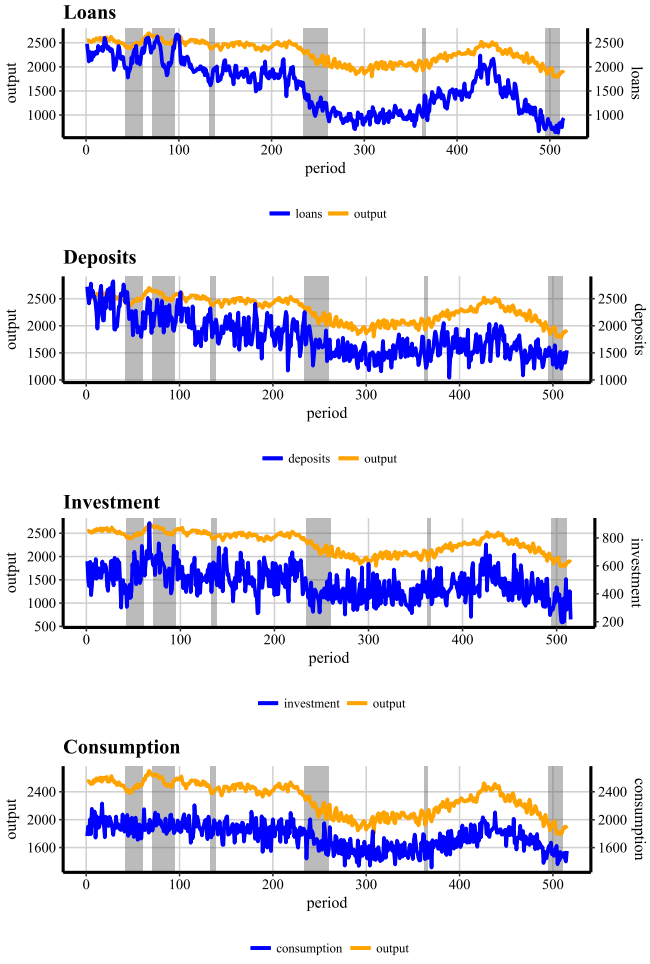

Fig. 5. Co-movements

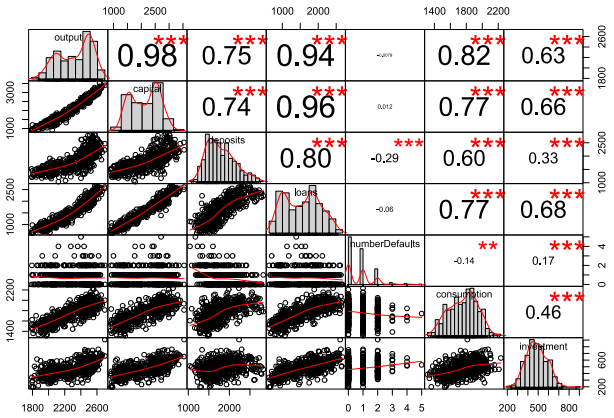

Fig. 6. Pearson correlations and their significance

Table 1. Variation of simulated time series

Series	Coefficient of Variation	Auto-correlation (1 lag)	Contemporaneous correlation with output
output	0.099	0.974	1
capital	0.311	0.977	0.976
deposit interest	0.168	0.965	0.043
loan interest	0.146	0.977	-0.551
approval rate	0.007	0.975	0.523
deposits	0.201	0.824	0.747
loans	0.330	0.969	0.938
consumption	0.103	0.657	0.819
investment	0.233	0.504	0.627

the model are smoothing out their consumption.

Deposits, while varying more than consumption, are on average 77% of output. This is very much in line with the deposit-to-outcome ratios of some developed economies, such as the US (81%), Germany (80%) and Italy (79%) (24).

Capital also shows positive relationship to output and to loans. The rate of return on capital is negatively related to output (through the relationship between the rate of return and the amount of capital), while the wage follows output $1:1$ (fixed inelastic labor supply and fixed labor share of income).

Figure 6 further spells out the relationships between different variables by giving correlation coefficients and scatter plots of various time series pairs.

Variance. We can also examine the variance of different time series and compare it to what we know about real world economic indicators. The first column of table 1 shows the coefficient of variation (standard deviation over mean) of each of the variables derived from the simulation.

We see that investment is more than 2 times as volatile as consumption - a feature also exhibited by real economies (25). On the other hand however it also seems

Table 2. Correlations with lags and leads of output (y)

Series	y_{t-4}	y_{t-3}	y_{t-2}	y_{t-1}	y_t	y_{t+1}	y_{t+2}	y_{t+3}	y_{t+4}
output	0.932	0.940	0.952	0.974	1	0.974	0.952	0.940	0.932
capital	0.927	0.933	0.942	0.955	0.976	0.991	0.968	0.948	0.936
deposit interest	0.0532	0.0498	0.0483	0.0457	0.0430	0.0431	0.0431	0.0417	0.0384
loan interest	-0.508	-0.519	-0.530	-0.541	-0.551	-0.549	-0.547	-0.544	-0.545
approval rate	0.474	0.487	0.497	0.511	0.523	0.523	0.525	0.525	0.523
deposits	0.696	0.707	0.724	0.744	0.747	0.723	0.703	0.704	0.712
loans	0.887	0.894	0.905	0.921	0.938	0.946	0.949	0.932	0.921
consumption	0.771	0.781	0.790	0.812	0.819	0.774	0.762	0.765	0.767
investment	0.576	0.576	0.578	0.581	0.627	0.645	0.651	0.615	0.599

that consumption, albeit much less volatile than investment (which agrees with the data), has about the same coefficient of variation than output (which doesn't).

Of the other variables, loans tend to be more volatile than deposits. The interest rate on deposits however tends to vary somewhat more than the one on loans. Finally, the approval cut-off looks like the most stable variable of them all (it stays around its initial value of 1 throughout the simulation). This indicates that bank agents haven't learned a good use of the approval rate in order to increase profits, and have instead decided to not use this control instrument and rely on the interest rate adjustments. Two variables that are not shown in the table are the wage and the rate of return to capital, which vary almost $1:1$ with output (since the factor markets are competitive).

Serial Correlation. As mentioned above, output exhibits a strong serial correlation. In fact the same is true for all derived variables. In addition most of them are correlated with output at various lags and leads. These correlations are presented in table 2.

Several observations can be made. First, investment tends to lead output (having higher correlation with future output than past output). The same goes for total loan amount, and the interest rate on loans. Consumption and deposits on the other hand seem to have higher correlations with past output than future output, indicating that they are lagging the cycle.

Running Economic Experiments

Probably the main advantage of an economic model based on reinforcement learning is its flexibility. Due to the ability to find approximate solutions to dynamical programming problems inherent to RL algorithms, the modeler needs to worry a lot less about the dimensionality of the agents' optimization problems and the behavior of their objective function. This means that it is relatively easy to experiment with changing the model in some way and letting the agents adapt to the new setting.

In order to demonstrate the usefulness of the model I conduct two "experiments" using the simulated economy. Both of them use as a baseline the economy described in the previous section, but change one of the rules of interaction, keeping everything else fixed. I then examine the differences in the resulting behavior.

Introducing deposit guarantees. In this experiment every aspect of the economy is identical to the benchmark, except now deposits of households are 'guaranteed'. This means that whenever a bank defaults, instead of getting all their deposits wiped out, households receive them back as cash.

In order to make up for that, in the next turn the wage is decreased equally (for both those households who had deposits in the defaulted bank and those who did not) by an amount, such that the total wage decrease equals the sum of paid back deposits (think of a government providing the guarantee, but then raising taxes to pay back for it).

Other than that, the banks can still default, losing their ability to pay back dividends for the next few periods and resetting their balances. Firms are also unaffected by the change, as a bank default still means that they don't have to repay any loans to that bank.

The simulation was ran for $8,000$ periods, for which agents adapted their policies from the one learned in the baseline case. All parameters were kept the same. After training, three separate simulations of 500 periods each for the baseline case, and three simulations of the deposit guarantee case were carried out and the results were compared.

The simulated economies with deposit guarantee share a lot of features with the baseline case - cyclical behavior, high autocorrelations, and the overall variance structure. However some key differences emerged.

For one, the average number of deposit withdrawals (either for consumption or for reinvesting in another bank) significantly decreases - from an average of 210 withdrawals per period across the three simulations of the baseline case, to just above 175 for the simulations of the deposit guarantee case. The difference is significant according to a simple t test.

This indicates that with the guarantee, households' propensity to keep their deposits increases. As a result the average level of deposits on the banks' books almost twice as high in the alternative scenario as in the baseline scenario (4254 units of the consumption good compared

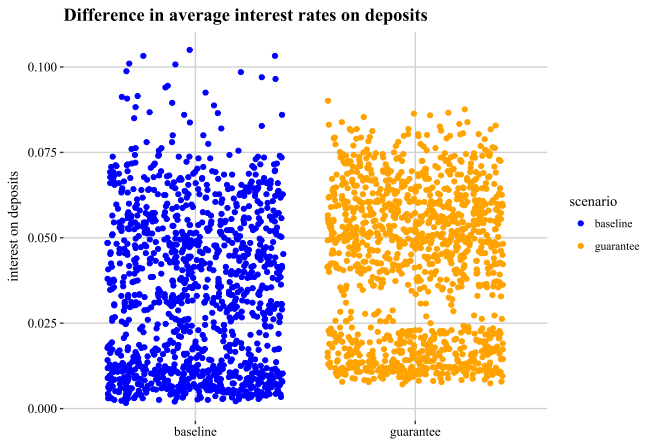

Fig. 7. Average interest rate on deposits

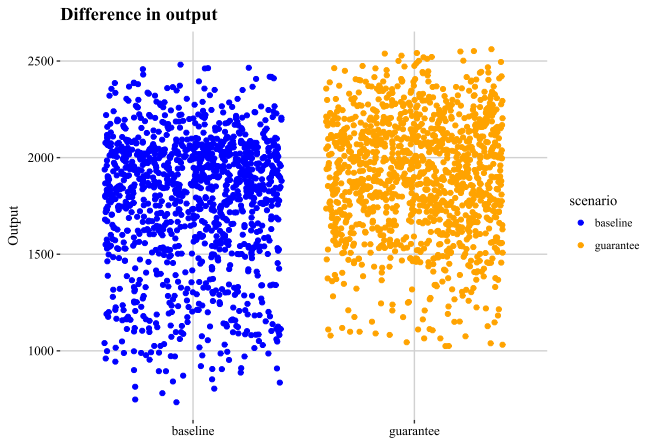

Fig. 8. Average per-period output

to 2294 units of the consumption good).

Interestingly, under the alternative scenario both the interest rate on deposits, and the interest rate on loans increase (the first of which is shown in figure 7). The change in the deposit interest rate (from an average of 0.034 across periods to an average of 0.042) is somewhat surprising, as one can guess that the higher supply of savings will allow banks to offer lower interest on them. The change in the loan interest rate (from 0.041 to 0.05) can be explained with the need to service the new deposits. Both differences are significant.

Despite the higher interest rates however, it seems that the increased levels of spending ushered by the deposit guarantee have lead to an overall increase in average output across periods - from 1774.5 in the simulations without deposit guarantee to 1901.7 in the ones where the guarantee was in effect. This is shown in figure 8. As with the other figures cited above, these differences are highly significant when compared using a t test.

Changing the required reserve ratio. The next experiment involves only changing the required reserves-to-deposits ratio of each bank. Five different economies were created, with reserve ratios of 0.04, 0.08, 0.12, 0.16 and 0.20 respectively.

The networks of the economy with a 4% reserve ratio were initialized by transferring the weights of the benchmark environment (where the required reserve ratio was 0.10). The agents were then trained for an additional 4098 periods. The weights of this model were transferred to the economy with an 8% reserve ratio, where training resumed for another 2048 periods (the agents should need less training to adjust to a smaller parameter difference). This training transfer was carried on until all five scenarios were covered.

Each of the cases for the different reserve requirements were then simulated 3 times for 500 periods. Table 3 shows a summary of the resulting economic time series after aggregating the three simulations of each scenario.

Table 3. Mean levels of various aggregates under different reserve requirements

Series	4%	8%	12%	16%	20%
output	2220	2108	2153	2116	2083
deposits	4323	4145	6172	8651	5134
loans	1596	1314	1425	1368	1265
consumption	1557	1573	1326	1500	1256
investment	364	340	341	341	307
deposit interest	0.027	0.042	0.034	0.028	0.043
loan interest	0.030	0.061	0.049	0.064	0.052

Several key differences are immediately obvious. First, average per-period output seems to be declining as the required reserve ratio increases - from a mean of 2220 to a mean of 2083. However, if we account for standard errors we see that the difference is not significant (see figure 9.

The level of deposits seem to be lower at laxer reserve requirements compared to the amount of deposits under the stricter regulations. The loan level on the other hand seems to be exhibiting a downward trend across the reserve requirement ratios. This is also reflected to a lesser degree in the overall investment levels. This observation is in line with an interpretation that as banks have to keep less reserves they manage to give out more loans. However these differences are also insignificant (the standard deviation of the loan level is around 500 for all five environments).

We can also notice that the interest rates on both deposits and loans do not exhibit a clear trend across the five scenarios. Even though the two interest rates are the lowest in the scenario with the lowest reserve requirements, we see that the deposit rate is 4% both in the 8% required reserve environment and in the 20%

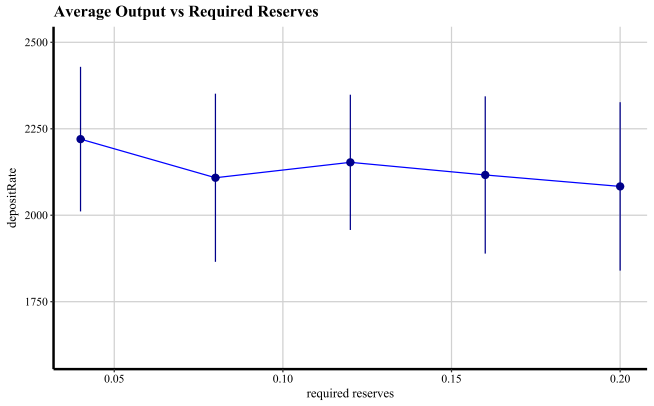

Fig. 9. Average output against required reserves

environment. Similarly the interest on loans is almost identical in the 8% and the 16% scenarios - around 0.06. The standard deviations of all interest rate means are around 0.01.

Finally, we see that the mean level of consumption also doesn't exhibit a clear trend across the scenarios, although it is markedly lower in the 12% and 20% environments. The differences are again insignificant, with standard deviations in the 700 to 1000 range for all simulations.

Overall it seems that lowering the reserve requirements tend to encourage banks to lend more, which leads to more projects being under taken and higher economic output in results. However the differences are two small to be conclusive. Possibly doing more simulations (compared to 3 per case here), adjusting the economy's parameters, or examining the learned policy functions of the agents may shed more light and give firmer conclusions.

Future Research

While implementing this model I have encountered several minor and major difficulties, some of which have been hinted at above.

Policy Convergence. In this paper agents interact with the environment and with each other. This creates several problems. First, the environment becomes dynamic, as the optimal policy of each agent changes with the current policies of all other agents. This means that convergences will be slower and likely non-monotone.

Second, since the three agent types have different population sizes, their replay buffers' size differ. Namely, because there are only 10 banks they accumulate experience to learn from much slower than the 500 households. The same holds for the training accuracy of the Q function approximator.

As a result it becomes hard to know how long to train the agents and to evaluate whether a certain network topology preforms better than another (26). In general the concept of 'good' performance of the reinforcement learning algorithm becomes blurry in a multi-agent setting.

In order for RL methods to be successfully implemented for economic modelling we need to develop clear objective measures of agent performance in any given environment.

Agent Sophistication. The model presented here is an example of what is known as 'model-free reinforcement learning' (17). This means that agents don't try to actively learn a model of the environment in which they operate. They change the propensity to choose certain actions based on utility feedback, but don't try to build a model of state transitions.

It is possible to introduce such a model in the learning scheme. One possibility to directly extend Q Learning by allowing each agent to 'predict' the next state s' given the current state and their actions (estimating $\hat{\mathbf{S}}_\mathbf{t}(a, \mathbf{S_t})$). This will allow to 'imagine' how different actions play out and hence learn a better policy (27). Doing so of course means another set of parameters for the prediction function that need to be learned by the agent.

Another way to increase the sophistication of the agents in the simulation is to allow them to model the actions of other agents. In the currently presented framework agents don't realize that they are strategically interacting with each other, and instead treat other agents as part of the environment. This is an obvious flaw, and can be improved by introducing some game theoretic concepts, such as Nash equilibrium (28). This however means that each agent has to learn some model of $\mathbf{A_{-i}}$, which significantly increases implementation complexity.

Choosing hyper-parameters. The output of the model created in this paper is quite sensitive to the values of the chosen hyper parameters. Changing productivity for example may lead to much higher or much lower landing overall, and to much less stable output.

Figuring out how each parameter affects the conclusions is not a straight-forward task and careful techniques that balance plausibility with sensible results must be devised.

A further open question is how to take the outputs of models such as these to the data. As mentioned above some of the features of this model do not conform with real world facts. Adjusting hyper parameters may change that. Criteria for deciding whether a simulated economy behaves like the data should be put forth, and methods to calibrate the model's parameters in order to meet these criteria while avoiding overfitting should be invented ‖

Continuous actions spaces. The reader would have noticed that the action space of all agent types in the model is discrete. This was intentional, since the DQN algorithm works with discrete actions. DQN is relatively easy to implement, and yet has a solid track record in achieving

‖ DSGE-style calibration is one option to do that

good performance in complex tasks (16), therefore it was deemed most appropriate for the current paper.

However, many economic decisions are better suited to be modelled as continuous. For example, it is more natural for the banks to directly choose the interest rate, rather than only move it up or down by a fixed increment.

There are plenty of reinforcement learning algorithms suited for continuous actions spaces (see for example (29)). Their application to economic models can be explored in the future.

Concurrent simulation. Another defining feature of the model was that time was divided into discrete chunks called periods. Furthermore within each period actors acted in 'turns' - first the banks, then the firms, then the households.

This is unrealistic and in principle unneeded. The model can be implemented concurrently, with each agent acting and interacting with the environment 'when ready' and without waiting for the others. This of course requires considerable development effort.

Policy interpretation. In this model agents learn a policy in a state space with around a hundred dimensions. The ability to deal with this high dimensionality is the main advantage of reinforcement learning, but it also presents the main challenge in drawing conclusions from the model.

It is sometimes hard to infer what the agents learned in a given scenario and how it differs from their policy in another one. Answering this questions is important for understanding the driving forces of the model.

Since the policy in this case amounts to choosing the highest predicted Q value, and since Q values are predicted using a deep neural network, the interpretation problem comes down to deciphering the inner workings of this neural network. This is not an easy task (30), but is an active area of research and some promising methods based on local interpretability may be useful (31).

Causal Links. Finally, possibly the biggest difficulty with the model (which is shared with other ABMs) is discovering the cause and effect link between different components of the system.

While the simulations clearly show that the economy exhibits cyclical fluctuations, that deposit guarantees lead to more savings, and that increasing required minimal reserves leads to more lending, explaining *why* these phenomena occur is not a straightforward task. I have tried to provide narrative explanations whenever possible, but scientific usefulness would require a much more rigorous approach.

Conclusion

This paper has presented a model of the financial sector of the economy in which heterogeneous agents learn how to maximize their utility in a bounded rational way using deep reinforcement learning. This approach has allowed for a high degree of flexibility on the side of the modeller in specifying the interactions and aggregate processes in the economy, with the effective number of state space dimensions for each agent being in the hundreds.

The key feature of the resulting dynamics of macroeconomic aggregates is the presence of endogenous business cycles. The cycles are driven by expansions and contractions in the firm loan market. Each credit crunch leads to less investment from firms and less outputs. Subsequent expansions of credit on the other hand tend to lead expansions in production.

Having and such an adaptive agent based model with learning agents allows to run a large number of economic experiments. Agents will then use their learning capabilities to autonomously find an approximate optimal policy in the new environment. Two such experiments were shown in the main text, however the number of possible questions that can be answered with such a model is much larger.

As far as the baseline model is considered an accurate enough representation of reality, the results of such experiments can then be used to inform monetary or fiscal policy by the government, or business decisions by the financial industry. While I make no claim that this particular model can serve this purpose, I strongly believe that it is possible to build a realistic enough model by employing tools similar to those presented here.

In order to do so however several major challenges need to be solved. Some of these challenges lay in the domain of multi-agent reinforcement learning - e.g. how to deal with multiple agents optimizing at the same time, or how to interpret learned policies. Others lay in the field of agent-based economics - how to design rich simulation environments that also allow us to easily draw causal conclusions, or how to make the model interactions more like real world ones while balancing computational complexity.

Despite these challenges however, agent based models based on reinforcement learning hold tremendous potential for Economics. With enough focus and dedication they may as well help shape the future of the field.

ACKNOWLEDGMENTS. I am grateful to the Bulgarian National Bank (and in particular the Student Scholarship Committee) for giving me the opportunity pursue the ideas presented in this paper. I am also thankful to Svilen Pachedzhiev for his guidance and advice.

References

1. Kydland F, Prescott E (1982) Time to build and aggregate fluctuations. *Econometrica* 50(6):1345–70.
2. Stiglitz JE (2017) Where modern macroeconomics went wrong, (National Bureau of Economic Research), Working Paper 23795.
3. Romer P (2016) The trouble with macroeconomics, (Stern School of Business New York University), Working paper.
4. Ackerman F (2002) Still dead after all these years: Interpreting the failure of general equilibrium theory. 9:119–139.
5. Daskalakis C, Goldberg PW, Papadimitriou CH (2009) On the complexity of computing nash equilibrium. *SIAM Journal of Computing* 39:195–259.
6. Epstein JM, Axtell R (1996) *Growing Artificial Societies: Social Science from the Bottom Up*. (The MIT Press) Vol. 1, 1 edition.
7. Richiardi M (2004) The promises and perils of agent-based computational economics, (University Library of Munich, Germany), Computational economics.
8. Buchanan MB (2009) Meltdown modelling. could agent-based computer models prevent another financial crisis? 460(7256):680–68.
9. Roth A, Erev I (1995) Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior* 8(1):164–212.
10. Erev I, Roth A (1998) Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review* 88(4):848–81.
11. Fudenberg D, Levine DK (2016) Whither game theory? towards a theory of learning in games. *Journal of Economic Perspectives* 30(4):151–70.
12. Chan-Lau J (2017) Abba: An agent-based model of the banking system. 17:1.
13. Ismail OR (2012) An agentbased computational model for bank formation and interbank networks. *McMaster University PhD Thesis*.
14. Grilli R, Tedeschi G, Gallegati M (2014) Bank interlinkages and macroeconomic stability. *International Review of Economics Finance* 34(C):72–88.
15. Fudenberg D, Peysakhovich A (2014) Recency, Records and Recaps: Learning and Non-Equilibrium Behavior in a Simple Decision Problem, (Harvard University OpenScholar), Working Paper 167691.
16. Mnih V, et al. (2013) Playing atari with deep reinforcement learning.
17. Sutton RS, Barto AG (2018) *Reinforcement learning - an introduction, 2nd Edition*. (Final Draft).
18. Bernanke B, Gertler M (1989) Agency costs, net worth, and business fluctuations. *The American Economic Review* 79(1):14–31.
19. Leibo JZ, Zambaldi V, Lanctot M, Marecki J, Graepel T (2017) Multi-agent reinforcement learning in sequential social dilemmas.
20. Melo FS, Ribeiro MI (2007) Q-learning with linear function approximation in *Learning Theory*, eds. Bshouty NH, Gentile C. (Springer Berlin Heidelberg, Berlin, Heidelberg), pp. 308–322.
21. Thrun S, Schwartz A (1993) Issues in using function approximation for reinforcement learning in *Proceedings of the 1993 Connectionist Models Summer School*, eds. M. Mozer, P. Smolensky DTJE, Weigend A. (Erlbaum Associates).
22. van Hasselt H, Guez A, Silver D (2015) Deep reinforcement learning with double q-learning. *CoRR* abs/1509.06461.
23. Acemoglu D, Scott A (1997) Asymmetric business cycles: Theory and time-series evidence. *Journal of Monetary Economics* 40(3):501 – 533.
24. TheWorldBank (2015) Global financial development database, Technical report.
25. Stock JH, Watson MW (1998) Business cycle fluctuations in u.s. macroeconomic time series, (National Bureau of Economic Research), Working Paper 6528.
26. Zawadzki E, Lipson A, Leyton-Brown K (2014) Empirically evaluating multiagent learning algorithms. *CoRR* abs/1401.8074.
27. Sutton RS (1990) Integrated architectures for learning, planning, and reacting based on approximating dynamic programming in *Machine Learning Proceedings 1990*, eds. Porter B, Mooney R. (Morgan Kaufmann, San Francisco (CA)), pp. 216 – 224.
28. Hu J, Wellman MP (2003) Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research* 4:1039–1069.
29. Sutton RS, McAllester D, Singh S, Mansour Y (1999) Policy gradient methods for reinforcement learning with function approximation in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99. (MIT Press, Cambridge, MA, USA), pp. 1057–1063.
30. Olden JD, Jackson DA (2002) Illuminating the black box: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling* 154(1):135 – 150.
31. Ribeiro MT, Singh S, Guestrin C (2016) "why should I trust you?": Explaining the predictions of any classifier in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. pp. 1135–1144.

Appendix

A. Firm Projects. As mentioned in the main text, each firm in the model draws a new "project" opportunity at the beginning of each period.

An example of such a project would have the following characteristics:

- duration: 13
- amount: 15.3
- income: 4.1
- income standard deviation: 0.23
- terminal income: 3.9
- terminal income standard deviation: 0.61
- default probability: 0.023
- liquidation multiplier: 0.34

This values are interpreted as follow: The project will be active for 13 periods. It costs 15.3 units of the consumption good to initiate the project. At the beginning of each period there is a 0.023 probability that the project will default and stop producing capital. Otherwise it will produce a random amount of capital, drawn from a normal distribution with mean 4.1 and standard deviation 0.23 (the distribution is truncated at 0). On the 12-th period the project will produce an additional terminal income, drawn form a normal distribution with mean 4.6 and standard deviation 0.61. If the firm defaults prior to the project completion, the project is liquidated, yielding a liquidation payment in terms of the consumption good equal to 0.34 times the amount needed to setup the project.

When the firm draws a new project, all these parameters are selected independently from uniform distributions. The support of these distributions can be seen in the project repository, but the example values given above are chosen to be close to the midpoints of the support in each case.

As mentioned in the main text, all firms have access to essentially the same 'project pool', meaning that the parameters above are drawn from the same distribution for all firms. One can experiment by allowing for different distributions - in effect creating inherently more or less productive firms - and studying the resulting firm policies.

B. Simulation Parameters. The following parameters were used to setup the simulation:

Households. There were 500 households, each possessing 1 unit of labor across all periods. Each household begins each iteration of the simulation with 100 units of the consumption good.

Firms. There were 100 firms. Each firm is initially endowed with 15 units of the consumption good and 1 unit of the capital good. Each firm has access to the same project pool.

Banks. There were 10 banks. Each was initially endowed with 400 units of the consumption good as reserves. The initial deposit and loan interest rates were set to 0.03 and 0.04 respectively. The initial approval rate was set to 1. The capital adequacy ratio was set to 0.08 and the required reserve ratio to 0.1 for the baseline simulation. The banks were forbidden to issue out dividends for the first 18 periods after a default. The deposit rate, the loan rate, and the approval ratio can all be adjusted by the banks up or down in increments of 0.0025.

Aggregate Parameters. The depreciation rate was set to 1 (full depreciation). The production function was Cobb-Douglas:

$$Y_t = A_t K_t^{\alpha} L_t^{1-\alpha} \quad [11]$$

Where K_t is the sum of the capital holdings of each firm, L_t is the sum of labor endowments of each household (in practice fixed across periods). A_t is total factor productivity, and was set to 3 for all periods of all simulations (one can experiment with shocking this variable). The capital share of output α was set to 0.3.

Training Parameters. The replay buffer size was 2000 for all agents. Weights were initialized uniformly. The batch size was set to 512. The learning rate was fixed at 0.001. The exploration rate used for the epsilon-greedy action was exponentially decreased from 1 at the beginning of training to 0.005.