stochastic gradient descent, which is a robust estimate of the classical gradient descent method and thus is only guaranteed to find the global optimal parameters of the model for convex problems (see, e.g., Deisenroth et al., 2020, Section 7.1.3). When the learning rate decreases at an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to local minimum (Bottou, 1998). However, there is theoretical evidence that in the deep learning context where the number of neural network parameters is very large, getting stuck in local minima is not a likely, and saddle points pose a greater challenge (see Dauphin et al., 2014). Adaptive gradient methods, such as the Adam optimizer we use in our work, address this challenge (Staib et al., 2019).

A.7.2 *Passing redundant information to the equilibrium net.* Heuristically, we found it helpful to pass redundant information to the neural network to stabilize the learning process. Since it increases the dimensionality of the input vector considerably, this is typically not done for methods like Smolyak sparse grids, or Gaussian processes. In our case, however, we found it helpful to increase the dimension of the input space by providing redundant information. Concretely, for the example studied in Section 5, we augmented the input state with the TFP shock $\eta(z)$, the depreciation $\delta(z)$, the aggregate capital K and labor L, the return on capital r and the wage for labor w, the aggregate production Y, the distribution of financial wealth across age groups $\mathbf{i}_{\text{fin}} = r \cdot \mathbf{k} + \mathbf{b} \cdot \min\{\kappa r, 1\}$, the distribution of labor income $\mathbf{i}_{\text{labor}} = w \cdot \mathbf{l}(z)$, the distribution of total wealth $\mathbf{i}_{\text{total}} = \mathbf{i}_{\text{labor}} + \mathbf{i}_{\text{fin}}$, and the transition probabilities of the exogenous shock $\Pi[z, :]$. To summarize, instead of passing the sufficient $2N + 1$-dimensional vector $\mathbf{x} = [z, \mathbf{k}^T, \mathbf{b}^T]^T$ to the neural network, we pass a $4N + 12$-dimensional vector $\mathbf{x}^* = [z, \eta(z), \delta(z), K, L, r, w, Y, \mathbf{k}^T, \mathbf{i}_{\text{fin}}^T, \mathbf{i}_{\text{labor}}^T, \mathbf{i}_{\text{total}}^T, \Pi[z, 1:4]]^T$ with redundant information to the neural network.[49]

A.7.3 *Dealing with infeasible predictions at the beginning of training.* At the beginning of the training process, we initialize the parameters of the neural network with random values. Consequently, the approximated policy, which predicts random values, might predict savings that imply that some age groups consume negative amounts or that result in negative aggregate capital. Both of which cause the program to terminate with an error message. To deal with this issue, we replace infeasible values by feasible ones and add a punishment term to the loss function that trains the neural network not to predict policies that imply infeasible values. As learning proceeds and the approximate policies improve, these replacements are no longer necessary. Furthermore, it is important to verify that these replacements only occur at the beginning of the training process and never during simulation, after the training is finished. The following method is rather ad hoc, however, we found it to work sufficiently well:

- Set a punishment parameter: $\epsilon^{\text{punish}} = 10^{-5}$.
- If the predicted consumption is less than ϵ^{punish}, replace by $\epsilon^{\text{punish}} : \hat{c}^i_{\text{adjusted}}(\mathbf{x}) = \max(\hat{c}^i(\mathbf{x}), \epsilon^{\text{punish}})$.
- Add the punishment term $(\frac{1}{\epsilon^{\text{punish}}} \max(-\hat{c}^i(\mathbf{x}), 0))^2$ to the loss function.

That way, if the neural network predicts a policy implying negative consumption, the optimality conditions can still be evaluated. At the same time, the large punishment term will guide the parameter updates so that improved policies do not imply negative consumption.

A.8 *An Example with an Analytical Solution.* In this section, we evaluate the performance of our solution method in a setting in which an exact solution is known. The knowledge of an exact solution allows us to evaluate the precision of our approximate solution beyond the relative errors in the Euler equations, which we have used so far. Furthermore, we use this example to demonstrate that our method is applicable in settings in which approximate aggregation may not hold.

[49] Note that $\mathbf{b}$ is implicitly passed to the neural network, because it is given by $\mathbf{b} = \frac{\mathbf{i}_{\text{fin}} - r \cdot \mathbf{k}}{\min\{\kappa r, 1\}}$.

We chose the same model as Krueger and Kubler (2004), which is an adapted version of the model in Huffman (1987). Although analytically solvable, this model is closely related to those we solve in this article.

A.8.1 *The model.* Next, for convenience and completeness, we outline this analytical test case. For readability, our notation here omits the dependence of variables on the node of the event tree, z^t, and we simply index them with time t, when there is no risk of confusion.

Households. Agents live for N periods and have log utility. Following Krueger and Kubler (2004), we assume that agents only work in the first period of their life: $l_t^s = 1$ for $s = 1$ and $l_t^s = 0$ otherwise. This implies that the aggregate labor supply is constant and equal to one: $L_t = 1$. Agents receive a competitive wage and can save in risky capital. Households cannot die with debt. Otherwise, there are no constraints or adjustment costs. The household's problem is given by

$$\max_{\{c_{t+i}^i, a_{t+i}^i\}_{i=0}^{N-1}} \mathrm{E}_t \left[\sum_{i=0}^{N-1} \log(c_{t+i}^i) \right] \tag{A.44}$$

subject to:

$$c_t^h + a_t^h = r_t k_t^h + l_t^h w_t, \tag{A.45}$$

$$k_{t+1}^{h+1} = a_t^h, \tag{A.46}$$

$$a_t^N \geq 0, \tag{A.47}$$

where c_t^h denotes the consumption of age group h at time t, a_t^h denotes the saving, k_t^h denotes the available capital in the beginning of the period, and r_t denotes the price of capital.

Firms. There is a single representative firm with Cobb–Douglas production function. The production function is given by

$$f(K_t, L_t, z_t) = \eta_t K_t^{\alpha} L_t^{1-\alpha} + K_t(1 - \delta_t), \tag{A.48}$$

where K_t denotes aggregate capital, L_t denotes the aggregate labor supply, α denotes the capital share in production, η_t denotes the stochastic TFP, and δ_t denotes the stochastic depreciation. The firm's optimization problem implies that the return on capital and the wage are given by $r_t = \alpha \eta_t K_t^{\alpha-1} L_t^{1-\alpha} + (1 - \delta_t)$ and $w_t = (1 - \alpha)\eta_t K_t^{\alpha} L_t^{-\alpha}$. Here, we study the special case where $L_t = l_t^1 = 1$.

A.8.2 *Equilibrium.* For completeness and convenience, we define the equilibrium we want to compute.

Competitive equilibrium.

Definition A.3 (competitive equilibrium). A competitive equilibrium, given initial conditions z_0, $\{k_0^s\}_{s=1}^N$, is a collection of choices for households $\{(c_t^s, a_t^s)_{s=1}^N\}_{t=0}^{\infty}$ and for the representative firm $(K_t, L_t)_{t=0}^{\infty}$ as well as prices $(r_t, w_t)_{t=0}^{\infty}$, such that

1. given prices, households optimize,
2. given prices, the firm maximizes profits,
3. markets clear.

Functional rational expectations equilibrium. Here, a FREE is defined as follows:

DEFINITION A.4 (FUNCTIONAL RATIONAL EXPECTATIONS EQUILIBRIUM). A FREE consists of equilibrium functions $\boldsymbol{\theta}_a : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{N-1}$, where $\boldsymbol{\theta}_a : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{N-1}$ denotes the capital investment functions, such that for all states $\mathbf{x} := [z, \mathbf{k}^T]^T \in \mathcal{Z} \times \mathbb{R}^N$, where $z \in \mathcal{Z}$ denotes the exogenous shock and $\mathbf{k} = [k_1, \dots, k_N]^T$ denotes the endogenous state (i.e., the distribution of capital) with $k_1 = 0$, we have for all $i = 1, \dots N-1$:

$$u'(c^i(\mathbf{x})) = \beta \mathrm{E}_z\big[r(\mathbf{x}_+)u'(c^{i+1}(\mathbf{x}_+))\big], \tag{A.49}$$

where $\mathbf{x}_+ = [z_+, 0, \boldsymbol{\theta}_a(\mathbf{x})^T]^T$, where z_+ denotes the random exogenous shock in the next period, and where

$$r(\mathbf{x}) = f_K\left(\sum_{i=1}^{N} x_{1+i}, 1, x_1\right), \tag{A.50}$$

$$w(\mathbf{x}) = f_L\left(\sum_{i=1}^{N} x_{1+i}, 1, x_1\right), \tag{A.51}$$

$$c^i(\mathbf{x}) = \begin{cases} w(\mathbf{x}) - \theta_a(\mathbf{x})_i, & \text{for } i = 1, \\ r(\mathbf{x})x_{1+i} - \theta_a(\mathbf{x})_i, & \text{for } i = 2, \dots, N-1, \\ r(\mathbf{x})x_{1+N} & \text{for } i = N. \end{cases} \tag{A.52}$$

A.8.3 *Computing the equilibrium.* We chose to present this model because it is closely related to the models we study in this article while an exact solution is available. In this section, we provide the exact solution and briefly reiterate how the equilibrium would be approximated using DEQNs.

Exact solution. Following Krueger and Kubler (2004), a FREE is given by

$$\boldsymbol{\theta}_a(\mathbf{x}) = \beta\left[\frac{1-\beta^{N-1}}{1-\beta^N}w(\mathbf{x}), \frac{1-\beta^{N-2}}{1-\beta^{N-1}}r(\mathbf{x})k_2, \frac{1-\beta^{N-3}}{1-\beta^{N-2}}r(\mathbf{x})k_3, \dots, \frac{1-\beta^1}{1-\beta^2}r(\mathbf{x})k_{N-1}\right]^T. \tag{A.53}$$

This solution implies that aggregate capital evolves according to

$$K_t = \gamma_1 w(\mathbf{x}_{t-1}) + \gamma_2 w(\mathbf{x}_{t-2})r(\mathbf{x}_{t-1}) + \cdots + \gamma_{N-1} w(\mathbf{x}_{t-(N-1)})r(\mathbf{x}_{t-1})r(\mathbf{x}_{t-2}) \dots r(\mathbf{x}_{t-(N-2)}), \tag{A.54}$$

where $\gamma_i := (\beta^i - \beta^N)/(1-\beta^N)$.

Approximating the solution with DEQNs. We seek to find parameters $\boldsymbol{\rho}$ of a deep neural network $\mathcal{N}_{\boldsymbol{\rho}}$, such that it approximates the true equilibrium policies:

$$\hat{\boldsymbol{\theta}}_a(\mathbf{x}) := \mathcal{N}_{\boldsymbol{\rho}}(\mathbf{x}) \approx \boldsymbol{\theta}_a(\mathbf{x}). \tag{A.55}$$

Since, in general, the true policy $\boldsymbol{\theta}_a(\mathbf{x})$ is unknown, we follow the algorithm described in this article and find parameters $\boldsymbol{\rho}$ by minimizing the errors in the equilibrium conditions with variants of mini-batch gradient descent. We can plug market clearing and the firms' optimization into the Euler equations so that the Euler equations characterize the recursive equilibrium. Given neural network parameters $\boldsymbol{\rho}$, which imply approximate policy functions $\hat{\boldsymbol{\theta}}_a(\mathbf{x})$, the errors in the equilibrium conditions at state $\mathbf{x}_j$ are given by

$$e^i_{\mathrm{EE}}(\mathbf{x}_j) := -u'(\hat{c}^i(\mathbf{x}_j)) + \beta \mathrm{E}_z\big[r(\hat{\mathbf{x}}_+)u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_+))\big], \text{ for } i = 1 \dots N-1, \tag{A.56}$$

where

$$\hat{\mathbf{x}}_+ = \left[z_+, 0, \hat{\boldsymbol{\theta}}_a(\mathbf{x})^T\right]^T, \tag{A.57}$$

and where z_+ denotes the random exogenous shock in the next period, and

$$r(\mathbf{x}) = f_K\left(\sum_{i=1}^{N} x_{1+i}, 1, x_1\right), \tag{A.58}$$

$$w(\mathbf{x}) = f_L\left(\sum_{i=1}^{N} x_{1+i}, 1, x_1\right), \tag{A.59}$$

$$\hat{c}^i(\mathbf{x}) = \begin{cases} w(\mathbf{x}) - \hat{\theta}_a(\mathbf{x})_i, & \text{for } i = 1, \\ r(\mathbf{x})x_{1+i} - \hat{\theta}_a(\mathbf{x})_i, & \text{for } i = 2, \dots, N-1, \\ r(\mathbf{x})x_{1+N} & \text{for } i = N. \end{cases} \tag{A.60}$$

The relative errors in the Euler equations are given by

$$e^i_{\text{REE}}(\mathbf{x}_j) := \frac{u'^{-1}\left(\beta \mathrm{E}_{z_j}\left[r(\hat{\mathbf{x}}_{j,+})u'(\hat{c}^{i+1}(\hat{\mathbf{x}}_{j,+}))\right]\right)}{\hat{c}^i(\mathbf{x}_j)} - 1 \tag{A.61}$$

and the loss function for the DEQN is given by

$$\ell_{\mathcal{D}_{\text{train}}}(\boldsymbol{\rho}) := \frac{1}{|\mathcal{D}_{\text{train}}|}\frac{1}{N-1}\sum_{\mathbf{x}_j \in \mathcal{D}_{\text{train}}}\sum_{i=1}^{N-1} e^i_{\text{REE}}(\mathbf{x}_j)^2, \tag{A.62}$$

where $\mathcal{D}_{\text{train}}$ denotes a set of states collected by simulating the economy using the current approximation of the policy functions.

A.8.4 *Parameterization and hyperparameters.* We solve the model for $N = 6$ agents and base our parameterization on Krueger and Kubler (2004). The chosen economic parameters as well as the chosen hyperparameters are given in Table A.8.

A.8.5 *Assessing the quality of the solution.* We train the neural network for a total of 10,000 episodes. We first evaluate the quality of the obtained policy functions by looking at the relative errors in the Euler equations, since this is the measure we focus on in the settings we are interested in and where no exact solution is available. Table A.9 shows the relative errors in the Euler equations on 15,000 randomly simulated states.[50]

The mean errors are in the order of $\sim 10^{-4}$, whereas the 99.9th percentile is below 1%. Since this is a much simpler model than our benchmark model, the errors are slightly lower than the errors we obtained for the models discussed in the previous sections.

Since the exact solution is available in this model, we now compare the approximate policy learned by our method to the precise solution. Figure A.8 shows the savings decisions learned by the DEQN (shown as stars) as well as the precise solution (shown as circles) as a function of the agents' capital at the beginning of each period in a scatter plot. The figure shows only 200 simulated states for better readability. The colors indicate the four exogenous shocks. Visually, the policies are indistinguishable. Table A.10 shows statistics of the errors in the learned savings decisions.

[50] We simulate 16,000 periods and throw away the first 1,000.

TABLE A.8
PARAMETERIZATION OF THE ECONOMIC PARAMETERS (TOP) AND THE HYPERPARAMETERS OF THE NEURAL NETWORK (BOTTOM)

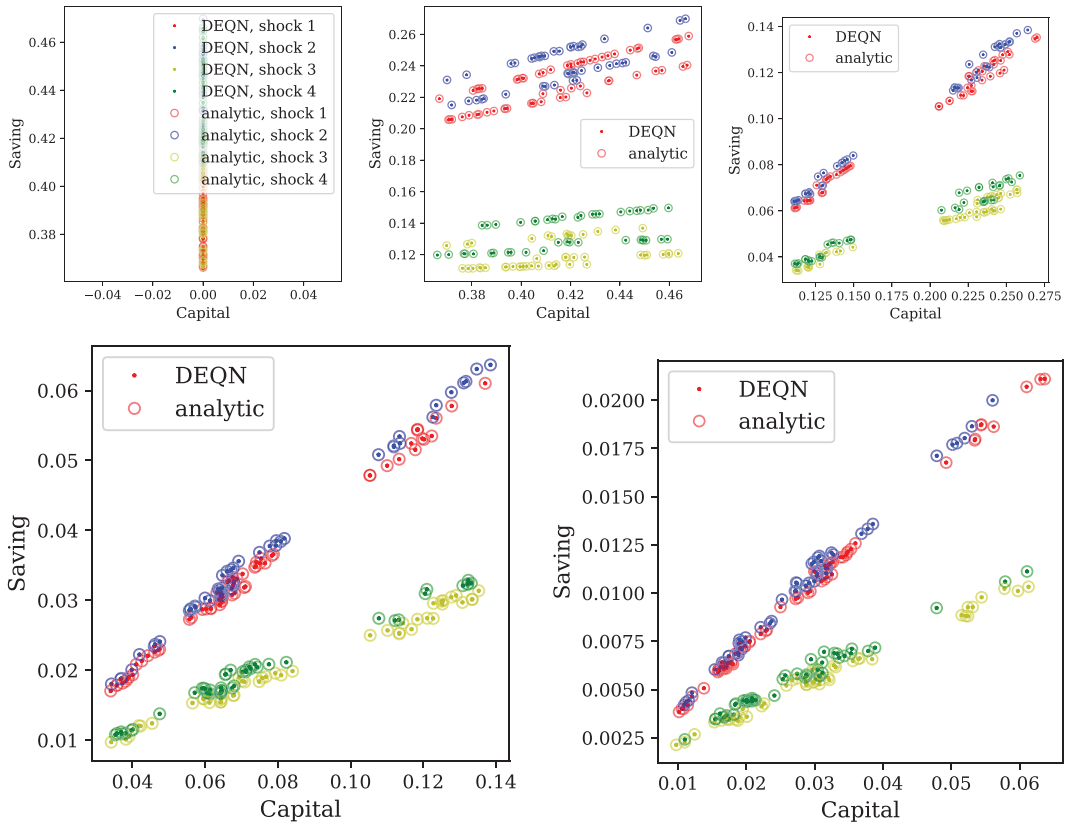
Number Age Groups N	Discount Factor β	Relative Risk Aversion γ	Capital Share α	TFP η	Depreciation δ	Persistence TFP $P(\eta_{t+1} = 1.05 \mid \eta_t = 1.05)$ $P(\eta_{t+1} = 0.95 \mid \eta_t = 0.95)$	Persistence Depreciation $P(\delta_{t+1} = 0.5 \mid \delta_t = 0.5)$ $P(\delta_{t+1} = 0.9 \mid \delta_t = 0.9)$
6	0.7	1	0.3	{0.95, 1.05}	{0.5, 0.9}	0.5 0.5	0.5 0.5

Learning Rate α^{learn}	Periods per Episode T	Epochs per Episode N^{epochs}	Mini-Batch Size m	Nodes Hidden Layers	Activations Hidden Layers
0.00001	12,800	20	640	100 50	relu relu

TABLE A.9
RELATIVE EULER EQUATION ERRORS ON 15,000 SIMULATED PERIODS

	Mean	Max	0.1	10	50	90	99.9
Rel Ee capital [log10]	−3.4	−2.4	−6.4	−4.4	−3.6	−3.0	−2.5

NOTE: The columns show the mean and the max errors as well as the 0.1st, 10th, 50th, 90th, and 99.9th percentile.



NOTES: Exact and approximate saving decisions plotted against capital holding at the beginning of the period for 200 simulated states for each of the 6 age groups. The circles show the exact solution, and the stars show the solutions learned by the deep equilibrium net after training for 10,000 episodes with parameters shown in Table A.8. The first row shows the policies for age groups 1, 2, and 3, the second row shows the policies for age groups 4 and 5. The policy of the sixth age group is trivial (consume everything and save nothing). Shock 1 is denoted in red, shock 2 in blue, shock 3 in yellow, and shock 4 in green. The TFP and deprecation (η, δ) of shock 1, shock 2, shock 3, and shock 4 are (0.95, 0.5), (1.05, 0.5), (0.95, 0.9), and (1.05, 0.9), respectively.

FIGURE A.8
EXACT AND APPROXIMATE SAVING DECISIONS

The mean errors in the policy functions are of order $\sim 10^{-4}$ and the maximum errors stay below 0.2%. In the next section, we study the extent to which an accumulation of errors could influence aggregates in this model.

A.8.6 *Approximate aggregation.* This section serves two purposes. First, we want a measure of the accumulation of errors with time as we simulate the economy forward. Second, we want to see to what extent approximate aggregation, as in the seminal work of Krusell and Smith (1998), holds in this model.

Following Krueger and Kubler (2004), we only use the log of aggregate capital to forecast next-period's capital contingent on the exogenous shock when assessing the approximate

TABLE A.10
STATISTICS OF THE RELATIVE ERRORS IN THE LEARNED POLICY FUNCTIONS ON 15,000 SIMULATED PERIODS

	Mean	Max	0.1	50	99.9
Relative policy errors age group 1 [%]	0.03	0.14	0.00	0.03	0.13
Relative policy errors age group 2 [%]	0.02	0.09	0.00	0.01	0.08
Relative policy errors age group 3 [%]	0.02	0.10	0.00	0.02	0.09
Relative policy errors age group 4 [%]	0.01	0.05	0.00	0.01	0.14
Relative policy errors age group 5 [%]	0.01	0.06	0.00	0.01	0.04
Relative policy errors age group 1 [log10]	−3.47	−2.85	−6.08	−3.54	−2.88
Relative policy errors age group 2 [log10]	−3.82	−3.06	−6.72	−3.91	−3.10
Relative policy errors age group 3 [log10]	−3.69	−2.99	−6.40	−3.75	−3.04
Relative policy errors age group 4 [log10]	−4.09	−3.29	−7.11	−4.26	−3.37
Relative policy errors age group 5 [log10]	−3.92	−3.33	−6.70	−4.00	−3.35

NOTE: The columns show the mean and the max errors as well as the 0.1st, 50th, and 99.9th percentile.

TABLE A.11
STATISTICS OF THE RELATIVE ERRORS IN THE SEQUENCE OF AGGREGATE CAPITAL COMPARED TO THE EXACT SOLUTION WHEN SIMULATING 15,000 TIME PERIODS

	Mean	Max
DEQN [%]	0.019	0.13
Log-linear forecast [%]	0.24	1.3
DEQN [log10]	−3.72	−2.88
Log-linear forecast [log10]	−2.62	−1.87

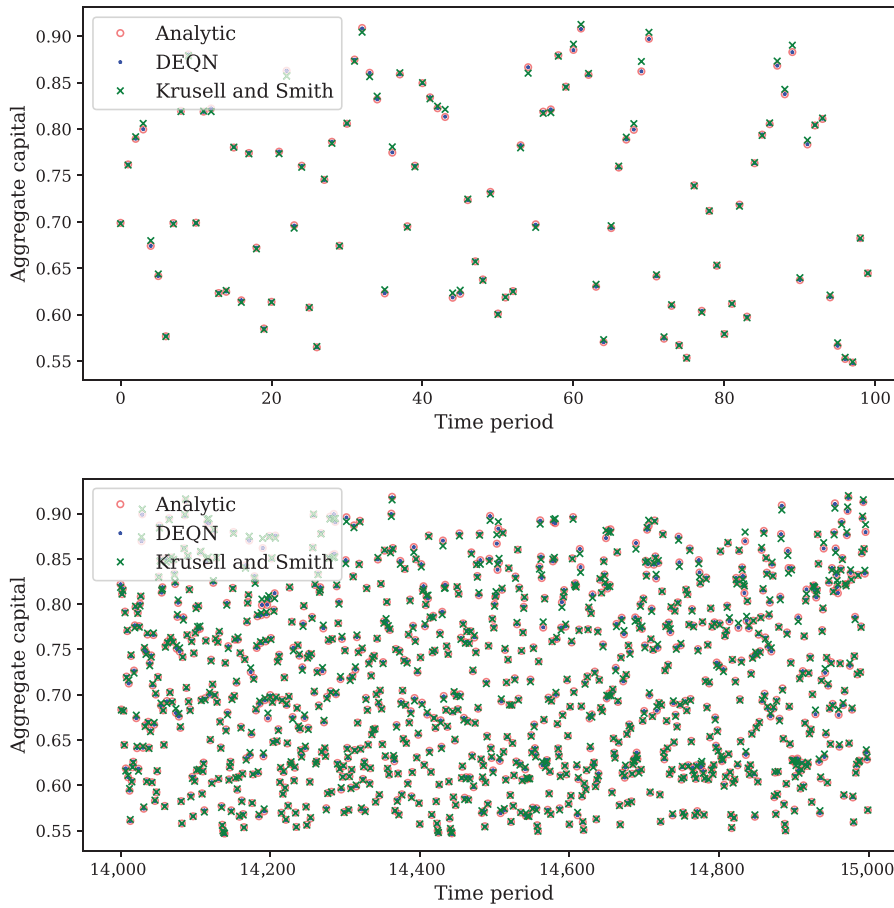
aggregation in this model.[51] More precisely, the functional form of the forecasting rule is given by

$$\log(K_{t+1}) = \phi_{0,z} + \phi_{1,z} \log(K_t), \tag{A.63}$$

where we estimate the parameters $\phi_{0,z}$ and $\phi_{1,z}$ with an ordinary least squares regression for each of the four shocks. The R^2 values of the corresponding regressions are between 0.9978 and 0.9979. The corresponding mean forecasting errors are between 0.16% and 0.24%. The corresponding maximum forecasting errors are between 0.61% and 0.98%.

Following Den Haan (2010) and Winberry (2018), we also analyze the extent to which approximate aggregation holds by simulating the linear forecast of aggregate capital forward without updating the value of aggregate capital. We do the same using the solution learned by the DEQN and compare the sequences of aggregate capital on 15,000 simulated states. Table A.11 reports the mean and maximum of the relative error of the sequence of aggregate capital compared to the exact solution. The mean relative errors in the sequence of aggregate capital are more than one order of magnitude lower using the policies learned by the DEQN than when using a log-linear approximation of the law of motion. To illustrate this point, Figure A.9 shows the simulated values of aggregate capital for the exact solution (red circles), using the policies learned by the DEQNs (blue dots), and when using a log-linear forecast (green crosses). For readability, we show the first 100 and the last 1,000 periods. The figure shows that the evolution of aggregate capital obtained from the policies learned by the DEQN is visually indistinguishable from the exact solution, whereas the evolution using a log-linear forecast shows visible differences. This further illustrates the point that our method applies to settings in which approximate aggregation does not obtain. For both approximate solutions, we find no evidence that the quality of the solution deteriorates when simulating many periods.

[51] The regressions use 15,000 simulated states in total, approximately 3,750 states for each shock.



NOTES: The red circles show the exact evolution, the blue stars show the evolution implied by the policies learned by the deep equilibrium net, and the green crosses show the evolution when using a log-linear forecast. The upper figure shows the first 100 and the lower figure the last 1,000 periods of 15,000 simulated periods.

FIGURE A.9

EXACT AND APPROXIMATED EVOLUTION OF AGGREGATE CAPITAL

As noted in Krueger and Kubler (2004), in this simple model, errors in forecasting the aggregate capital stock do not translate to welfare losses for the agents. This is because, in the simple model presented here, the optimal decision does not depend on interest rates.

A.9 *Parallelization Scheme.* To speed up the solution process of the computational method introduced in this article by orders of magnitude, we present a generic and highly-scalable parallelization scheme that allows us to make efficient use of state-of-the-art high-performance computing (HPC) facilities, whose performance nowadays can reach up to hundreds of Petaflop/s (see, e.g., https://www.top500.org). The use of modern HPC hardware may enable us to solve hard problems with many state variables in a relatively short time and to tackle problems that have, thus far, been out of reach.

Our algorithm can be described as iterating on two phases: the simulation phase, where new training data are generated using the previous iteration's neural network, and the learning phase, where the neural network is trained on the new data. We make use of contemporary frameworks to parallelize both phases. More precisely, our method is parallelized by combining TensorFlow (Abadi et al., 2015) with Horovod (Sergeev and Del Balso, 2018).