Machine Learning for Economic Models: A Review and Replication of "Deep Equilibrium Nets"

Annais Gangolf
Haverford College
Haverford, PA 19041
agangolf@haverford.edu

Devansh Goyal
Haverford College
Haverford, PA 19041
dgoyal@haverford.edu

Samuel E. Ross
Haverford College
Haverford, PA 19041
seross@haverford.edu

Abstract

We review and replicate the "Deep Equilibrium Nets" method of Azinovic, Gaegauf, and Scheidegger [1] to numerically solve the analytical overlapping generations model of Krueger and Kubler [5]. We succeed in generating comparable results, finding marginal error between our approximation and the true solution. We additionally find that adding training steps between simulations improves convergence, and that productivity shocks may affect savings less than depreciation shocks.

1 Introduction

Dynamic economic models attempt to simulate the workings of an economy mathematically. At their core, they consist of individuals and firms making decisions to maximize their happiness, or *utility*, over time and subject to a series of constraints. For example, at a given time, an individual may know their current income and past savings, as well as economy-wide parameters like the interest and inflation rates; we call these known quantities *state variables*. Based on this information, our agent must choose how much to spend and how much to save; these are *control variables*. These choices in turn impact the state variables of the next period, creating a recursive complexity. The solution to such a model is a so-called *policy function* that specifies the optimal values of the control variables given any configuration of the state variables. Mathematically, this is known as a dynamic optimization (or programming) problem.

These models can be complicated in various ways, such as adding nonlinearities or complex stochastic processes to the model's parameters. The most common extension, though, is the specification of a lifetime for individuals and processes for death and reproduction; this is known as an *overlapping generations model* (OLG). An OLG model in which agents only live for three or four periods and face lax borrowing constraints is quite simple, and can be solved analytically. But as we consider agents living for more periods or facing more complex constraints, the model can easily become intractable and an analytic solution infeasible. This difficulty has spurred a wide array of research into finding computationally efficient numerical methods for solving these models.

In this report, we consider machine learning-based approaches to numerically approximating the solution (i.e., the optimal policy functions) to such complex models. Specifically, we explain and partially replicate the "Deep Equilibrium Nets" method of Azinovic, Gaegauf, and Scheidegger [1], then apply their approach to several variations of the OLG model presented in Krueger and Kubler [5]. We find similar baseline results to the initial paper, yet find that additional training steps between dataset re-simulations improves model convergence, and that reducing the probability of productivity-based recessions yields marginal decreases in saving.

We proceed as follows. Section 2 surveys the related literature. We explain the economic model used in Azinovic, Gaegauf, and Scheidegger [1] in Section 3. In Section 4, we describe their neural network approximation method. We discuss our partial replication and results in Section 5. Section 6 concludes.

2 Related Literature

Our project lies at the junction of two related literatures: neural networks as tools in numerical approximation, and applications of machine learning to economics. The former was largely founded by the seminal work of Hornik, Stinchcombe, and White [4]. In this paper, the authors prove that a sufficiently deep feed-forward neural network can act as a *universal function approximator*. That is, such a network can approximate any analytic function in any (Borel-measurable) space to an arbitrary precision. This is the theoretical foundation for neural networks' usefulness in solving the complex models that concern our paper. When finding an analytic solution—a policy function on some space—traditional methods suffer from the so-called *curse of dimensionality*: as the number of state variables increases, the number of possible states increases exponentially. By efficiently approximating this function, neural networks allow us to circumvent this issue and find reasonably precise solutions with relatively little computational power.

For the latter literature, machine learning methods have recently become increasingly common in computational economics and econometrics. Mullainathan and Spiess [6] give a high-level overview of these advances and detail a few specific use cases. These include using image-classification models to construct novel datasets, and using machine learning techniques like decision trees and random forests rather than traditional regression to estimate economic relationships.

Combining these two trends, our project focusses on approximating dynamic economic models with neural networks. Leveraging the universal function approximator property discussed above, there is a burgeoning literature on precisely this topic. Fernández-Villaverde, Hurtado, and Nuño [3] focus on estimating the optimal policy functions for financial continuous-time models. Such models consider a continuum of time values (usually $t \in [0, \infty)$) on which the policy function is optimized. Their method leverages a symbolic equation-solver and uses a neural network to approximate an intermediate function, then uses these two results to estimate the policy functions themselves. Fan et al. [2] use physics-informed neural networks to integrate a financial continuous-time model's partial differential equations into its loss function, thus solving the model via its standard training process.

Azinovic, Gaegauf, and Scheidegger [1], the main focus of this report, use an alternating sampling–fitting strategy to solve a complex discrete-time (i.e., $t \in \mathbb{N}$) overlapping generations model. In brief, they initialize a neural network with random parameters and a loss function modelled after an OLG model's constraints and objective. They then generate a training dataset using this model, and re-train the model on this dataset. This alternating process is repeated thousands of times to arrive at a solution to the model. By applying the technique to a model with an analytic solution, they find that their method is arbitrarily precise. We further explain this method and its application to a broader range of models in the following sections.

3 The Economic Model

The economic model that we examine in this paper is that of Krueger and Kubler [5], and is used in Appendix 8 of Azinovic, Gaegauf, and Scheidegger [1]. It is a basic discrete-time overlapping generations model in which agents live for N periods and derive utility from consumption by its natural logarithm; that is, $u(c_s^t) = \log c_t^s$ for an individual of age s at time t. At the beginning of each period $t \in \mathbb{N}$, a single representative agent is born and any at age $N + 1$ die; thus, there are always N agents alive at any given time.

For tractability, we assume that each agent only works during the first period of their life $s = 1$, and is "retired" otherwise; we say that $l_t^1 = 1$ and $l_t^s = 0$ for $s \neq 1$. Since there is only one of each agent with that age at any given time, this means that there is only ever one agent working, so our aggregate labor supply is given by $L_t = 1$ for all $t \in \mathbb{N}$.

At the beginning of each period t, an agent of age s must choose how much to save a_t^s and how much to consume c_t^s in order to maximize their lifetime utility. They face a competitive wage w_t (according to which they earn income) and can save (or borrow) in risky capital k_t^s. They must not carry any debt when they die. Finally, when an agent is born, they begin with no capital; that is, $k_t^1 = 0$ for

all t. We formulate the problem of the agent born at time t mathematically like

$$\max_{\{c_t^s, a_t^s\}_{s=1}^N} E_t \left[\sum_{s=1}^{N} \beta^{s-1} \log c_{t+s}^s \right] \quad \text{s.t.} \quad c_t^s + a_t^s = r_t k_t^s + l_t^s w_t, \quad k_{t+1}^{s+1} = a_t^s, \quad a_t^N \geq 0, \quad (1)$$

where r_t denotes the rate of return on capital, β is a discounting factor, and we take the expectation over random shocks that we will describe below. Notice that the first constraint represents the agent's budget constraint, the second prescribes the movement of capital, and the third enforces no end-of-life debt. We comment further on the second for clarity: when an agent chooses to save money a_t^s, they do so by investing in capital k_t^s (think of this like stocks or bonds). This capital must originate somewhere; in fact, we say that the agent's savings a_t^s in period t determines entirely the capital k_{t+1}^{s+1} in the following period. This is a standard model for capital flow in the economy.

We must now prescribe how the competitive wage w_t and rental rate of capital r_t are set. We assume that there is a single representative firm that models the entire production side of the economy. This firm produces output (the goods or services that the agents consume) according to the function

$$f(K_t, L_t, z_t) = \eta_t K_t^\alpha L_t^{1-\alpha} + K_t(1 - \delta_t), \quad (2)$$

where K_t is aggregate capital (the total amount of capital in the economy) at time t, L_t is aggregate labor (recall that $L_t = 1$ always), α is a parameter that determines the relative proportion of capital and labor needed to produce output, and η_t and δ_t (together represented as z_t) are stochastic shocks to productivity and depreciation (e.g., how fast machines wear out) respectively. The first term of (2) represents the actual production of goods using capital and labor, while the second term reflects the need to repair "wear and tear" on factory machines and the like. When the firm optimizes by differentiating (2), we obtain expressions for w_t and r_t

$$w_t = (1 - \alpha)\eta_t K_t^\alpha L_t^{-\alpha}, \quad (3)$$

$$r_t = \alpha \eta_t K_t^{\alpha-1} L_t^{1-\alpha} + (1 - \delta_t). \quad (4)$$

This completely characterizes the supply side of our economy.

The uncertainty in the model (reflecting, e.g., the risk of recession) comes from the stochastic terms η_t and δ_t that appear in the firm's production function (2). Each of these terms can take on two values: $\eta_t \in \{0.95, 1.05\}$ and $\delta_t \in \{0.5, 0.9\}$. Their stochasticity is determined by a Markov process in which each state is independent and determined by a probability for the "high" state p_η and p_δ respectively. Together, these variables determine four possible states of the economy $z_t \in \{1, 2, 3, 4\}$ with the first corresponding to $\delta_t = 0.5$ and $\eta_t = 0.95$, the second $\delta_t = 0.5$ and $\eta_t = 1.05$, the third $\delta_t = 0.9$ and $\eta_t = 0.95$, and the fourth with $\delta_t = 0.9$ and $\eta_t = 1.05$. In the case $p_\eta = p_\delta = 0.5$ the transition matrix is given by

$$\Pi_1 = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}. \quad (5)$$

In the case $p_\eta = 0.75$ and $p_\delta = 0.5$, we have

$$\Pi_2 = \begin{bmatrix} 0.125 & 0.375 & 0.125 & 0.375 \\ 0.125 & 0.375 & 0.125 & 0.375 \\ 0.125 & 0.375 & 0.125 & 0.375 \\ 0.125 & 0.375 & 0.125 & 0.375 \end{bmatrix}. \quad (6)$$

We use each of these cases in our later experiments.

Finally, we must define a notion of equilibrium. Generally speaking, an equilibrium given initial conditions z_0 and $\{k_0^s\}_{s=1}^N$ is a set of choices $\{(c_t^s, a_t^s)_{s=1}^N\}_{t=0}^\infty$, aggregate variables $(K_t, L_t)_{t=0}^\infty$, and prices $(r_t, w_t)_{t=0}^\infty$ so that (1) given prices, agents maximize their utility, (2) given prices, the firm maximizes its profits, and (3) markets clear (i.e., supply equals demand). Following Krueger and Kubler [5], we can further define a *functional rational expectations equilibrium* (FREE), which contains a so-called capital investment function $\theta_a : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{N-1}$ that assigns savings choices $a_t = [a_t^1, \ldots, a_t^{N-1}]^T \in \mathbb{R}^{N-1}$ (we exclude the oldest agent since $a_t^N = 0$ optimally) to our current stochastic shock state $z_t \in \mathcal{Z}$ and distribution of capital $k_t = [k_t^1, \ldots, k_t^n]^T \in \mathbb{R}^N$; this is our

policy function. Obtained by differentiating (1), we also must satisfy the *Euler equation* for all $s \in \{1, \ldots, N-1\}$:

$$u'(c^s(x)) = \beta E_z \left[r(x_+) u'(c^{s+1}(x_+)) \right], \tag{7}$$

where $x_+ = [z_+, 0, \theta_a(x)^T]^T$ and $z_+ \in \mathcal{Z}$ is the shock state in the next period. The functions w and r are given by (3) and (4) respectively, and consumption can be calculated as

$$c^s(x) = \begin{cases} w(x) - \theta_a(x)_s, & i = 1 \\ r(x)x_{s+1} - \theta_a(x)_s, & i = 2, \ldots, N-1 \\ r(x)x_{N+1}, & i = N \end{cases}. \tag{8}$$

The exact solution to this model from Krueger and Kubler [5] is given by

$$\theta_a(x) = \beta \left[\frac{1-\beta^{N-1}}{1-\beta^N} w(x), \frac{1-\beta^{N-2}}{1-\beta^{N-1}} r(x)k_2, \ldots, \frac{1-\beta^1}{1-\beta^2} r(x)k_{N-1} \right]^T. \tag{9}$$

We will use this solution later to evaluate the performance of our neural network in approximation.

4 The Neural Network

The goal of Azinovic, Gaegauf, and Scheidegger [1] is to use a neural network find an approximation to the equilibrium policy function $\theta_a : \mathcal{Z} \times \mathbb{R}^N \to \mathbb{R}^{N-1}$ that determines agents' savings (and, implicitly, consumption) given the state of the economy. That is, we would like to find the values of the parameters ρ for a neural network $\mathcal{N}_\rho$ so that

$$\hat{\theta}_a(x) = \mathcal{N}_\rho(x) \approx \theta_a(x). \tag{10}$$

Before discussing the precise architecture of this neural network, we consider its loss function. The equilibrium defined in Section 3 was primarily characterized by the Euler equation (7). Thus, we can measure how far from equilibrium the model is by the absolute errors

$$e^s_{EE}(x) = -u'(\hat{c}(x)) + \beta E_z \left[r(\hat{x}_+) u'(\hat{c}^{s+1}(x_+)) \right] \quad \forall s \in \{1, \ldots, N-1\}, \tag{11}$$

where $\hat{x}_+ = [z_+, 0, \hat{\theta}_a(x)^T]^T$, z_+ is the next period's shock, $r(x)$ and $w(x)$ are as above, and

$$\hat{c}^s(x) = \begin{cases} w(x) - \hat{\theta}_a(x)_s, & i = 1 \\ r(x)x_{s+1} - \hat{\theta}_a(x)_s, & i = 2, \ldots, N-1 \\ r(x)x_{N+1}, & i = N \end{cases}. \tag{12}$$

We may also measure these errors in a relative manner, by

$$e^s_{REE}(x) = \frac{u'^{-1}\left(\beta E_z \left[r(\hat{x}) u'(\hat{c}^{s+1}(\hat{x})) \right]\right)}{\hat{c}^s(x)} - 1 \quad \forall s \in \{1, \ldots, N-1\}. \tag{13}$$

We can use this function to find the mean average error across agents and data points and obtain our loss function:

$$\ell_{\mathcal{D}_{\text{train}}}(\rho) = \frac{1}{|\mathcal{D}_{\text{train}}|} \frac{1}{N-1} \sum_{x \in \mathcal{D}_{\text{train}}} \sum_{s=1}^{N-1} e^s_{REE}(x)^2, \tag{14}$$

where $\mathcal{D}_{\text{train}}$ is our training dataset. In the experimental implementation, we also add terms penalizing negative consumption and capital to this cost function; these are omitted here for simplicity.

Now that we have our loss function, we may consider the architecture of the neural network. Hereafter, for tractability, we assume that $N = 6$, following Azinovic, Gaegauf, and Scheidegger [1] and since this value is implemented in our experiments. Since we would like our neural network to approximate the policy function $\theta_a : \mathcal{Z} \times \mathbb{R}^6 \to \mathbb{R}^5$, it might be reasonable to assume that our network should have seven input nodes and five output nodes. However, Azinovic, Gaegauf, and Scheidegger [1] find that adding redundant state information to the input improves the convergence of the model. Thus, our input layer consists of one node for the current shock; six nodes for the capital distribution; two nodes for each of the actual values of the productivity and depreciation shocks; three nodes for aggregate capital, labor, and output; two nodes for the prices—wage and rental rate; and 18 nodes for the respective financial income, labor income, and total income figures. In all, our input layer

has 32 nodes. Following Azinovic, Gaegauf, and Scheidegger [1], we then add two dense layers of 100 and 50 neurons respectively, and finish with five output nodes—one for each non-trivial savings prescription. We use RELU activation for the dense nodes and softplus activation for the output nodes to guarantee satisfaction of positive savings constraints in equilibrium. The neural network weights are created using Xavier initialization.

Training the model consists of two steps, *episodes* and *epochs*. At the beginning of an episode, a new training dataset $\mathcal{D}_{\text{train}}$ is simulated by generating a random sequence of economic shocks the size of the dataset, then passing a random economic state to the neural network as a seed to sequentially generate the dataset. For example, if we were simulating a dataset of size three, we might generate the random shocks 1, 3, and 4 and the random capital distribution (the redundant variables are omitted for simplicity) $x = (0, 0.05, 0.2, 0.1, 0.05, 0.2)$, then feed this back into the neural network to obtain the non-trivial capital distribution for the next period $\mathcal{N}_\rho(3, x)$; this is repeated recursively for the other periods. In an epoch, on the other hand, we perform standard minibatch gradient descent using our loss function (14), the Adam optimizer, and gradient clipping with values of ± 1. One thing is particularly notable about this training technique: the simulated training data actually improves as the model gets better, since it itself is simulated using the network. It is precisely the model's equilibrium conditions (the Euler equation) that allow us to use unsupervised learning and thus make this contraction method feasible.

5 Results

For our project, we use the materials provided by Azinovic, Gaegauf, and Scheidegger [1] to replicate Appendix A.8 of their paper. Specifically, we port their project from an older software library to its new version (Tensorflow 1 to 2), maintaining the plotting portion of their code, as well as several portions of the economic model development. We then use this new code to run three experiments: In one, we replicate the model and training hyperparameter choices used by Azinovic, Gaegauf, and Scheidegger [1] (we call this model EQL-20); we then explore a modification the training regime to improve convergence (EQL-50); and we then use different transition probabilities in the economic model to demonstrate an application of this research (MOD-50).

We train each model for 1000 episodes. We additionally give our choices of hyperparameters for each of the economic models in Table 1 and for each neural network in Table 2. For the most part, we follow the parameters given in Azinovic, Gaegauf, and Scheidegger [1]. However, for the EQL-50 model we increase the number of epochs within each episode from 20 to 50. With this change, we hope to investigate how modifying the modelling regime impacts training performance. For the MOD-50 model, we also use 50 epochs per episode and additionally change the probability of the high productivity state from 0.5 to 0.75. Note that the first two models use the transition matrix given in (5), while the third uses (6). We hope here to explore how changing the economic model might allow this approximation method to give us concrete economic insights.

Table 1: We give the choices of hyperparameters for our economic model from Krueger and Kubler [5] for each of our experiments. All of these values follow from Azinovic, Gaegauf, and Scheidegger [1], being chosen by them due to empirical evidence and convention. The only divergence in these parameters is the use of a higher probability of the high productivity state in the MOD-50 experiment.

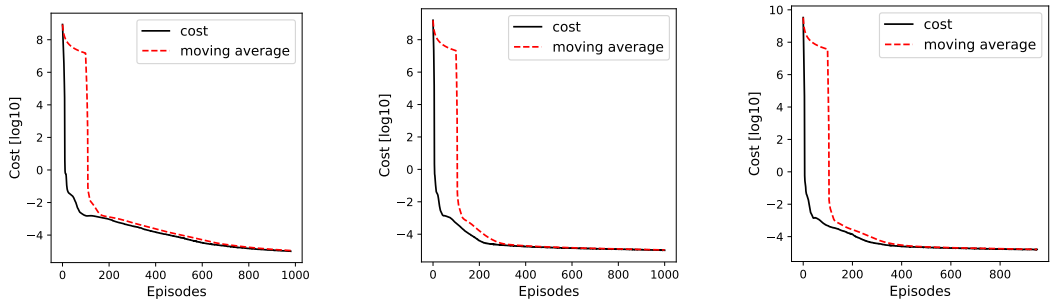
Model	Age Groups (N)	Discount Factor (β)	Capital Share (α)	p_η	p_δ
EQL-20	6	0.7	0.3	0.5	0.5
EQL-50	6	0.7	0.3	0.5	0.5
MOD-50	6	0.7	0.3	0.75	0.5

In Figure 1, we plot the value of the loss function (14) during training for each of our models. For each, we see strong convergence throughout, with a initial precipitous drop followed by a soft decrease towards an asymptote at zero; each model ends training with a cost value near 10^{-5} (the specific value for, e.g., EQL-50 is $10^{-5.33}$). These results are comparable to Azinovic, Gaegauf, and Scheidegger [1]. Between models, we observe one striking difference: the models that use 50 epochs per episode rather than 20 (EQL-50 and MOD-50) converge much faster than their counterpart (EQL-20). On one hand, this is quite expected—we are training our model for 150% more epochs overall. But we see that, for example, EQL-50 reaches a cost value near $10^{-4.5}$ after only 200 episodes, while

Table 2: We give the choices of hyperparameters for our neural network for each of our experiments. Most of these values are the same as those used by Azinovic, Gaegauf, and Scheidegger [1], however in the latter two models we increase the epochs per episode from 20 to 50, which we find significantly improves model convergence.

Model	Learning Rate	Periods per Episode	Epochs per Episode	Minibatch Size
EQL-20	10^{-5}	10240	20	512
EQL-50	10^{-5}	10240	50	512
MOD-50	10^{-5}	10240	50	512

it takes nearly 800 for EQL-20 to reach this same value. This is much more of a difference than linear variation with the additional epochs alone could explain. Moreover, one epoch is significantly less computationally expensive than the data simulation at the beginning of each episode; so, with this change, we increase our number of training steps by 150% while only increasing training time marginally (roughly 50%). Thus, by adding additional training steps between dataset re-simulations, we seem to improve model converge significantly.

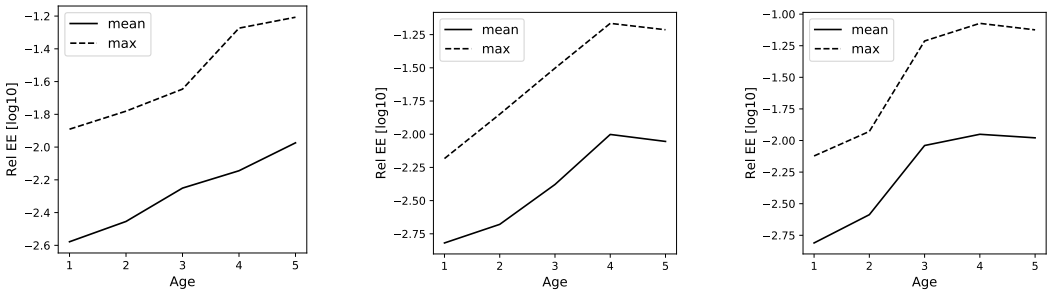


(a) EQL-20 (b) EQL-50 (c) MOD-50

Figure 1: We record each model's cost for our experiments across 1000 episodes of training. Note that none of the plots show exactly 1000 time periods due to modulus indexing issues. The cost shown is a measure—derived from the economic model's equilibrium conditions—of how incorrect the neural network's predictions are. It can be thus used to compute gradients for training the models. Each episode consists of 20–50 training epochs (20 for EQL-20, 50 otherwise) and the training dataset is re-simulated using the neural network at the beginning of each episode. Observe that the 50-epoch models train significantly faster than those with 20-epochs, even beyond a per-epoch basis. Our plotting code is taken from Azinovic, Gaegauf, and Scheidegger [1].

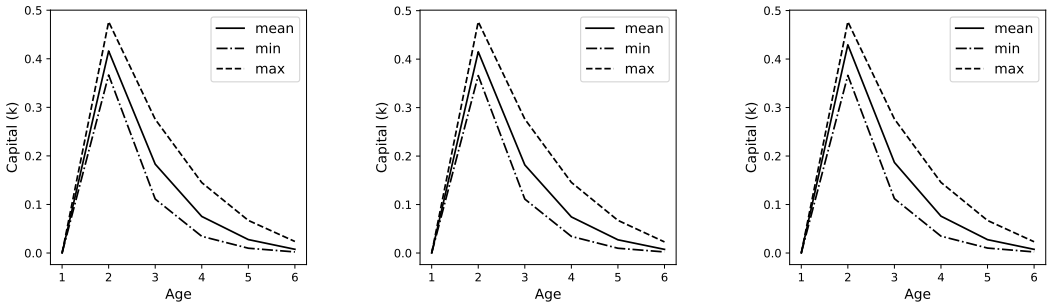
Figure 2 gives the relative Euler equation errors (the main component of the neural network loss) across age groups for each of our models. In each plot, we see an overall upward trend in error (albeit with small values throughout). This makes sense since the complex of policy decisions increases with age; that is, an agent just born is guaranteed to have no capital and can only see the current economic shock level, while an agent at age four has a saving decision contingent on past economic shocks and their own past savings choices. This explanation is in fact evidenced by the slight decrease in error we see for the agent of age five in EQL-50 and MOD-50. If we examine a similar plot for these models at only 200 episodes of training (available upon request), we see a similar monotone trend as EQL-20. But as training continues, a consistent dip for the oldest agent appears. We suspect that this is because the oldest agent's decision is less complex than the second oldest: They know that they must die with no debt and should consume everything in the last period. Thus, as the model converges, it seems to learn the relative complexity of decisions at each agent's age.

We plot the capital distribution predicted by each trained model in Figure 3. Each plot is comparable, and aligns with our expectations: Agents are born with no capital, then aggressively save in their first year since it is the only one in which they are employed. They then gradually decrease their savings to zero over the next four periods, exhausting their accumulated resources. This result has a satisfying economic and intuitive appeal, since agents appear to rationally curtail their consumption during their highest income period, so that they can enjoy a more comfortable retirement later in life.



(a) EQL-20 (b) EQL-50 (c) MOD-50

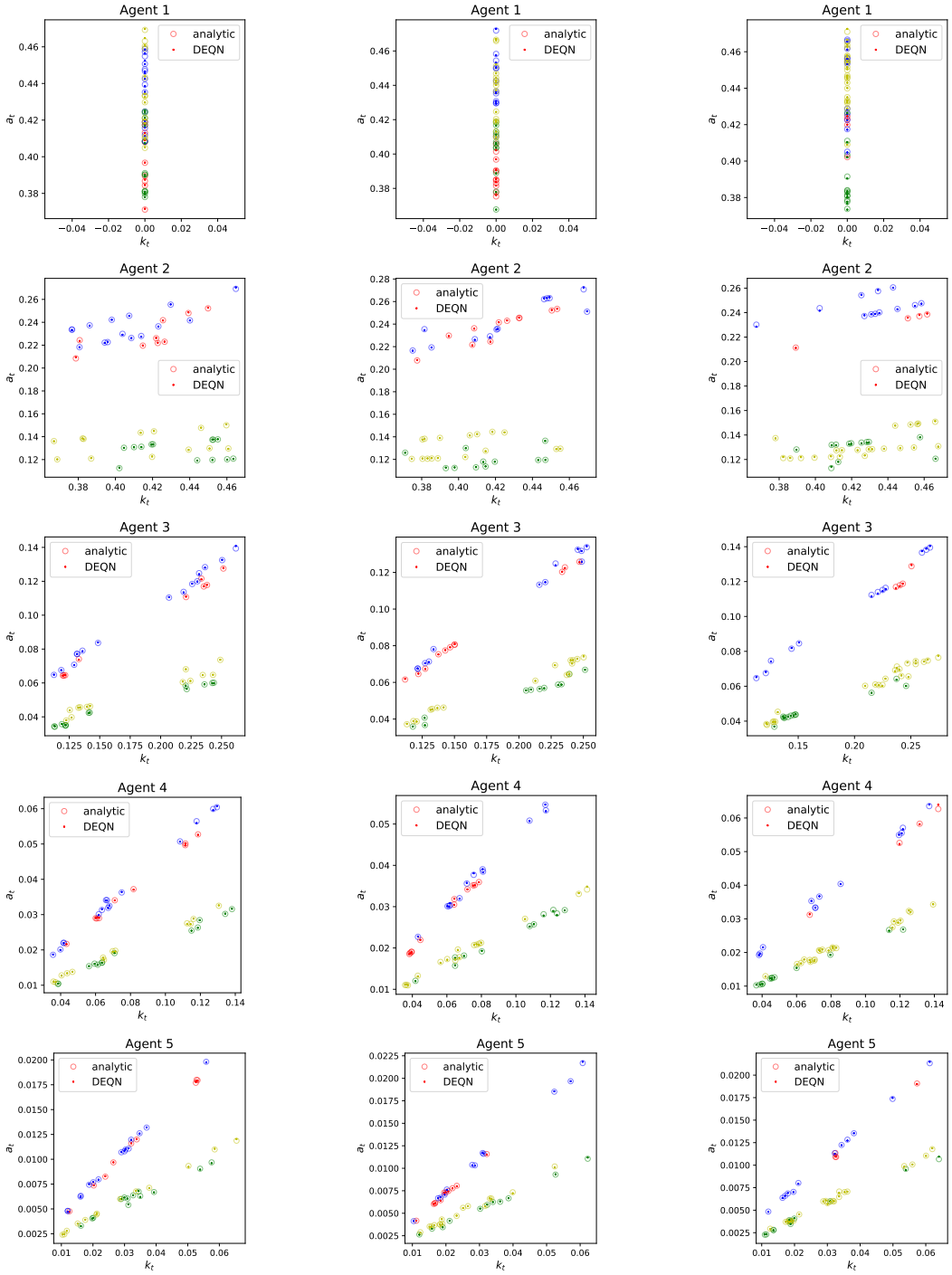
Figure 2: We report the Relative Euler Equation Error (REE) across age groups for each of our models. REE is the main component of our neural networks' cost. It essentially indicates how far the network's proposed solution is from true equilibrium (i.e., the analytic solution). We see that the model does best with younger agents, worsening with age until the oldest, which it may find slightly easier. This aligns with expectations, since younger agents have less complex shock histories. We take our plotting code from Azinovic, Gaegauf, and Scheidegger [1].



(a) EQL-20 (b) EQL-50 (c) MOD-50

Figure 3: We plot the distribution of capital (i.e., how much each agent saves in the previous period) for each trained model across age groups by sampling economic states. We find that agents save most during the beginning of their lives then gradually decrease saving as they age. This is expected, since agents only work to earn income during their first year of age. We see little difference between the models. Plotting code is from Azinovic, Gaegauf, and Scheidegger [1].

Finally, we evaluate the performance of each of our models against the analytic solution given by Krueger and Kubler [5] in Figure 4. To do this, we randomly sample 50 economic states from the final simulated dataset, then plot for each agent their current capital and recommended savings by the model and the true solution. In every plot, the analytic policies are nearly identical to the approximations, indicating that all our models converged properly to the true solution. For an economic takeaway, observe that, on the whole, the plots indicate that agents save the least during the least severe shock ($z = 4$), then slightly more when depreciation is good but productivity is bad ($z = 3$), significantly more when depreciation is bad but productivity is good ($z = 2$), and the most when both are bad ($z = 1$). This yields both the basic insight that agents save more during recessions due to the impact on lifetime earnings, and that shocks to depreciation seem to have a more severe impact of the economy than shocks to productivity (our depreciation shock is absolutely larger than the productivity shock, but this difference seems larger than that alone could explain). We also see the difference in shock probability between EQL-50 and MOD-50 in two ways: In the plots for MOD-50, there are significantly more blue points than red and more yellow than green, reflecting the greater probability of the higher productivity state. Additionally, although slight, we seem to see better separation between between points that have the same depreciation shock but different productivity shocks in MOD-50 (that is, the red points are more consistently below the blue, and likewise), indicating potentially that the greater probability of good productivity induces agents to save less during low productivity times, relying more on the reappearance of the good shock in later periods.



(a) EQL-20 (b) EQL-50 (c) MOD-50

Figure 4: For each of our experiments (EQL-20, EQL-50, MOD-50), we sample 50 states of the economy randomly and compare the trained models' policy (i.e., their choice of savings a_t given current capital k_t) with that of the analytical solution given by Krueger and Kubler [5]. We denote shock $z = 1$ by red points, shock 2 by blue, shock 3 by green, and shock 4 by yellow. Each row of the figure represents the policy for an individual of the listed age. The results for the agent with one year of age ("agent 1") appear as a line because every agent is born with no capital. We omit the results for the agent with six years of age since they are trivial (the oldest agent always optimally saves nothing, since they know they will die thereafter). We take our plotting code from Azinovic, Gaegauf, and Scheidegger [1].

6 Conclusion

In this report, we analyze the "Deep Equilibrium Nets" method of Azinovic, Gaegauf, and Scheidegger [1] for finding approximate solutions to economic models. Specifically, we replicate Appendix 8 of their paper, in which they use the overlapping generations model of Krueger and Kubler [5] to assess their approximation's performance against an analytic solution.

We find similar results to Azinovic, Gaegauf, and Scheidegger [1] within our baseline specification, indicating the success of our replication. We also find for all our models that the analytic solution aligns almost perfectly with our approximations. We additionally investigate a modification of the given training regime by increasing the number of epochs within each episode, finding a significantly increased convergence rate beyond that predicted by the increase in overall epochs alone. This suggests a synergistic relationship between training steps and dataset re-simulation, even when the simulated dataset has significant noise (i.e., the model is yet unconverged). To demonstrate an application, we also found several economic results; namely, that modifying the probability of productivity-based recessions has negative yet relatively marginal impacts on savings, suggesting that precautionary saving may originate more from depreciation shocks than productivity shocks alone.

These results reflect the promising development of machine learning-based approaches to solving economic problems. With this method in hand, a wide array of complex economic models—hitherto unfeasible to solve analytically or approximately—could be used to gain insights about concrete phenomena from recessions to growth. Indeed, economic modelling and development could be another field soon revolutionized by recent advancements in deep neural network design.

References

[1] Marlon Azinovic, Luca Gaegauf, and Simon Scheidegger. "Deep Equilibrium Nets". In: *International Economic Review* 63.4 (Nov. 1, 2022), pp. 1471–1525. DOI: 10.1111/iere.12575. URL: https://doi.org/10.1111/iere.12575 (visited on 04/23/2023).

[2] Benjamin Fan et al. "Deep Learning for Solving and Estimating Dynamic Macro-Finance Models". Cambridge, MA, 2022. URL: https://math.mit.edu/research/highschool/primes/materials/2022/Fan-Qiao.pdf (visited on 04/23/2023).

[3] Jesús Fernández-Villaverde, Samuel Hurtado, and Galo Nuño. "Financial Frictions and the Wealth Distribution". In: *National Bureau of Economic Research Working Paper Series* No. 26302 (2019). DOI: 10.3386/w26302. URL: http://www.nber.org/papers/w26302.

[4] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Networks* 2.5 (Jan. 1, 1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[5] Dirk Krueger and Felix Kubler. "Computing Equilibrium in OLG Models with Stochastic Production". In: *Journal of Economic Dynamics and Control* 28.7 (Apr. 1, 2004), pp. 1411–1436. DOI: 10.1016/S0165-1889(03)00111-8. URL: https://www.sciencedirect.com/science/article/pii/S0165188903001118.

[6] Sendhil Mullainathan and Jann Spiess. "Machine Learning: An Applied Econometric Approach". In: *Journal of Economic Perspectives* 31.2 (2017), pp. 87–106. DOI: 10.1257/jep.31.2.87. URL: https://www.aeaweb.org/articles?id=10.1257/jep.31.2.87.

Supplementary Materials

The code we wrote, along with instructions for our replication of Azinovic, Gaegauf, and Scheidegger [1], is available online at https://github.com/devanshgoyal25/DeepEquilibriumNets.