DeepHAM: A Global Solution Method for Heterogeneous Agent Models with Aggregate Shocks

Jiequn Han[*], Yucheng Yang[†], and Weinan E[‡]

This version: January 2025
First version: December 2021

Abstract

We propose an efficient, reliable, and interpretable global solution method, the *Deep learning-based algorithm for Heterogeneous Agent Models (DeepHAM)*, for solving high dimensional heterogeneous agent models with aggregate shocks. The state distribution is approximately represented by a set of optimal generalized moments. Deep neural networks are used to approximate the value and policy functions, and the objective is optimized over directly simulated paths. In addition to being an accurate global solver, this method has three additional features. First, it is computationally efficient in solving complex heterogeneous agent models, and it does not suffer from the curse of dimensionality. Second, it provides a general and interpretable representation of the distribution over individual states, which is crucial in addressing the classical question of whether and how heterogeneity matters in macroeconomics. Third, it solves the constrained efficiency problem as easily as it solves the competitive equilibrium, which opens up new possibilities for normative studies. As a new application, we study constrained efficiency in heterogeneous agent models with aggregate shocks. We find that in the presence of aggregate risk, a utilitarian planner would raise aggregate capital for redistribution less than in absence of it because poor households do more precautionary savings and thus rely less on labor income.

[*]Flatiron Institute. Email: jiequnhan@gmail.com.
[†]University of Zurich and SFI. Email: yucheng.yang@uzh.ch.
[‡]Peking University. Email: weinan@math.pku.edu.cn.

We thank the co-editor and anonymous referees for very helpful suggestions. We are grateful to Gianluca Violante for numerous discussions and constructive feedback. We thank Mark Aguiar, Jinhui Bai, Anmol Bhandari, David Childers, Pablo Guerron, Mahdi Kahou, Nobu Kiyotaki, Felix Kubler, Moritz Lenel, Ziang Li, Albert Marcet, Ben Moll, Galo Nuño, Jonathan Payne, Jesse Perla, Mikkel Plagborg-Møller, Simon Scheidegger, Chris Sims, Jeffrey Sun, Wei Xiong, Shenghao Zhu, and many seminar participants for helpful comments. The replication code can be found on `https://github.com/frankhan91/DeepHAM`. Please address correspondence to Yucheng Yang and Jiequn Han.

Keywords: Heterogeneous agent models, aggregate shocks, global solution, deep learning, generalized moments, constrained efficiency.

1 Introduction

The incorporation of both explicit heterogeneity and aggregate fluctuations into quantitative models has been one of the most important recent developments in macroeconomics. It has become an important agenda for a number of reasons. First, uneven dynamics across sectors and population groups after major economic fluctuations and policy shocks suggest that heterogeneity and aggregate shocks are necessary considerations when studying fundamental macroeconomic problems. Second, the recent development of the heterogeneous agent New Keynesian (HANK) models suggests that heterogeneity gives rise to key channels in the transmission of aggregate shocks. These channels are crucial in determining the correct aggregate implications of monetary and fiscal policies. Third, the advancement of computational methods and growth in computing power have allowed economists to build models more realistic than the representative agent models that have long been dominant in both academic and policy research.

Despite the attention this agenda has received, its workhorse models, heterogeneous agent (HA) models with aggregate shocks, still present severe computational challenges. Ideally, one would like to develop solution methods that fulfill the following basic requirements:

- **Efficiency**: The method should be computationally efficient, especially for complex HA models with multiple state variables. This is necessary in order to use the method for calibration, estimation, and further quantitative analysis.

- **Reliability**: The method should produce accurate solutions for all practical situations that HA models are intended for. In particular, it should be applicable beyond the local perturbation regime if nonlinear and nonlocal effects of aggregate shocks are important.

- **Interpretability**: We are not only interested in the numbers that come out of an algorithm, but also in understanding the mechanisms underlying the results. For that, the major components of the algorithm should be interpretable. In particular, solutions to HA models usually involve mappings from the distribution over all individual states to the agent's welfare or decision outcomes. An ideal solution should provide interpretability of these mappings through an interpretable representation of the state distribution. An interpretable representation of the distribution is also necessary to derive reduced dynamic models at the aggregate level from the original HA model.

- **Generality**: The method should in principle be applicable to a wide variety of different HA models (simple or complex), and to different notions of equilibrium (e.g. competitive equilibrium or constrained efficiency problem).

Currently, there are two main approaches for solving HA models with aggregate shocks, and they satisfy only a subset of the requirements listed above. The first is the Krusell-Smith (KS) method, a global solution method proposed in Krusell and Smith (1998). The KS method approximates agent distribution with a small number of moments (e.g., the first moment), which are interpretable. It is efficient when solving simple HA models but becomes less effective when solving complex HA models with multiple shocks, or multiple endogenous states, or when solving the estimation problem. This is due to the large number of variables (such as the possibly large number of moments needed) introduced and the resulting curse of dimensionality problem. That is, the computational cost increases exponentially with the number of variables.

The second approach is the local perturbation method proposed in Reiter (2009). This method allows one to study or estimate complex HA models, but is not reliable for models where aggregate shocks bring significant nonlinear or nonlocal effects. Nonlinear effects are common in models with zero lower bounds (ZLB). Nonlocal effects appear in models with large aggregate shocks, or in models (e.g., macro-finance models) where explicit consideration of aggregate uncertainty plays an important role in shaping agents' behavior, resulting in the deviation of the risky steady state from the deterministic steady state. Table 1 summarizes the advantages and limitations of these two methods.

Model features	KS method	Perturbation method	DeepHAM
Multiple shocks	No	Yes	Yes
Multiple endogenous states	No	Yes	Yes
Large shocks	Yes	No	Yes
Risky steady state	Yes	No	Yes
Nonlinearity (e.g., ZLB)	Yes	No	Yes

Table 1: Model features that different solution methods for HA models with aggregate shocks can handle.

In this paper, we propose a new solution method, the *Deep learning-based algorithm for Heterogeneous Agent Models (DeepHAM)*, which satisfies all the requirements listed above. We formulate a HA model with N agents, where N is large when we aim to solve a problem with a continuum of agents. To solve HA models, the fundamental objects of interest are agents' value and policy functions. A complication arises from the fact that these functions depend not only on the agent's own state, but also the distribution of all agents' states in

the economy. To address this issue, we represent the value function and policy function with deep neural networks, and present an algorithm to update the value and policy functions iteratively. Deep neural networks are a class of functions in deep learning, which have a strong representational capability for high dimensional functions and can be efficiently optimized with stochastic gradient descent algorithms.[1] In contrast to existing literature that uses deep learning to represent high dimensional policy and value functions directly (Maliar et al., 2021; Azinovic et al., 2022), we introduce *generalized moments* to represent the state distribution efficiently, and solve for the value and policy functions as functions of the generalized moments. Generalized moments extract useful information from the state distribution, similarly to classical moments, but are represented by neural networks and automatically determined by the algorithm. The introduction of generalized moments also ensures that the agent's optimal policy and value functions are invariant under permutations of the ordering of the agents. Conceptually, the generalized moments reduce the state dimension while remaining readily interpretable and flexible enough to encode the state distribution through algorithmically-determined moments. In addition, the generalized moments share a similar representation to quantities such as the first moment of wealth distribution that are typically observed in the interaction between the agents and the whole economy. As we will see below, a single generalized moment not only leads to more accurate solutions than using only the first moment, but also extracts key information from the agent distribution with first order implications for aggregate welfare and dynamics. Thus, it provides a general and interpretable way to study a key question in macroeconomics: whether, why, and how inequality matters for the macroeconomy.

As we will demonstrate later, DeepHAM meets all the requirements listed above. First, it shows better global accuracy compared with existing methods. In the baseline model we study, DeepHAM with only the first moment in the state vector reduces the Bellman equation error by 37.5% compared to the KS method. DeepHAM with one generalized moment reduces the error by 54.2%. Second, the computational cost of DeepHAM is quite low in solving complex HA models, and it does not suffer from the curse of dimensionality. DeepHAM can efficiently solve HA models with a Brunnermeier and Sannikov (2014) type of financial sector. Third, the use of generalized moments allows us to revisit classical questions in macroeconomics of whether and how heterogeneity matters to aggregate welfare and dynamics. Krusell and Smith (1998) famously argued that, in their setup, individual welfare is affected by other agents only through the mean of wealth distribution. With the generalized

[1] Compared to the classical polynomial approximation used in the literature (Fernández-Villaverde et al., 2016), the neural network serves the same purpose as a function approximator. The difference is that the neural network does not rely on a fixed set of basis functions and can approximate high dimensional functions more efficiently.

moments, we find that an unanticipated redistributional policy shock would have a non-zero welfare impact instantaneously on those households who are not in the policy program, even when the mean of the wealth distribution is not affected. Finally, as a demonstration of the generality of DeepHAM, we show that it can be used to solve the constrained efficiency problem in HA models, which is regarded as a challenging problem in the literature, as easily as solving the competitive equilibrium. This opens up new possibilities for normative studies in HA models. We present the first global solution to constrained efficiency problems in HA models with aggregate shocks. In the presence of aggregate risk, households would do more precautionary savings and thus rely less on labor income. As a consequence, a utilitarian planner would raise aggregate capital for redistribution less than in the economy without aggregate risk.

DeepHAM should be applicable to a large class of economic problems with heterogeneity and aggregate shocks, and here we list some such examples. First, since DeepHAM does not suffer from the curse of dimensionality with more endogenous states or shocks, we can introduce more realistic portfolio options like housing and mortgage choices (Kaplan, Mitman, and Violante, 2020). We can efficiently handle models with *ex ante* heterogeneous agents, such as households and financial experts (Brunnermeier and Sannikov, 2014), rational and bounded-rational agents (Woodford and Xie, 2022), among others. We can study models with rich firm heterogeneity and aggregate shocks (Khan and Thomas, 2013). We can also study HA models with multiple shocks, where the shocks take forms that appear commonly in the DSGE literature (McKay and Reis, 2016). Second, we can use DeepHAM to study models with large shocks such as the COVID-19 shock (Arellano, Bai, and Mihalache, 2024), strong nonlinearities such as sovereign default models Chatterjee and Eyigungor (2015); Dvorkin et al. (2021), or large endogenous fluctuations such as those discussed in Petrosky-Nadeau, Zhang, and Kuehn (2018). We can also use DeepHAM to study asset pricing and the wealth effects of monetary and fiscal policy in HA models. The empirical literature has shown these factors to be important (Andersen et al., 2021) but, due to computational challenges, they have only been studied in models with limited heterogeneity (Kekre and Lenel, 2021; Caramp and Silva, 2021). We can also study the interaction of asset pricing and wealth inequality (Cioffi, 2021). Third, we can study optimal policy problems with heterogeneous agents using the Ramsey approach (Davila et al., 2012; Nuño and Moll, 2018), such as optimal monetary and fiscal policy (Bhandari, Evans, Golosov, and Sargent, 2021; Le Grand, Martin-Baillon, and Ragot, 2021; Yang, 2022) following Feng et al. (2024), or optimal macroprudential policy (Bianchi and Mendoza, 2018). Such study has heretofore been limited by computational challenges. Furthermore, DeepHAM can be extended to study HA models where interactions between agents are not mediated through aggregate prices, but through direct pairwise

interactions, such as search models (Lise and Robin, 2017; Kargar, Passadore, and Silva, 2020) and network models (Arbex, O'Dea, and Wiczer, 2019), following recent developments in Payne, Rebei, and Yang (2024). Finally, methodologically, we can extend DeepHAM to do model calibration by introducing a calibration target in the objective function so that we can solve and calibrate HA models in the same algorithmic framework.

Related Literature. Our work builds on an extensive literature on solving HA models with aggregate shocks. As discussed, there are two main approaches in the literature (Algan et al., 2014): the global Krusell-Smith (KS) method (Krusell and Smith, 1998; Den Haan, 2010; Fernández-Villaverde et al., 2019; Schaab, 2020; Cao et al., 2023), and the local perturbation method (Reiter, 2009; Winberry, 2018; Ahn et al., 2018; Boppart et al., 2018; Bayer and Luetticke, 2020; Auclert et al., 2021). Due to the curse of dimensionality, the KS method cannot handle complex HA models with multiple assets and multiple shocks. The perturbation method has been applied to complex HA models, for solving local dynamics around the deterministic stationary equilibrium in the absence of aggregate shocks, or for parameter estimation (Liu and Plagborg-Møller, 2019). However, the perturbation method is inapplicable to problems with nonlinear dynamics induced by aggregate shocks, or problems that are not close to the deterministic stationary equilibrium. DeepHAM can handle complex HA models with aggregate shocks and provide a global solution.

This paper proposes a general methodology that extracts the key information of the distribution that matters for aggregate welfare and dynamics. This is important for studying the role of heterogeneity in macroeconomics (Krueger et al., 2016; Kaplan and Violante, 2018). Most papers in this literature study the role of heterogeneity with quantitative decomposition after solving the model (Kaplan et al., 2018), or with sufficient statistics that are derived analytically based on the first-order approximation (Auclert, 2019). In contrast, we propose a general numerical method that extracts key generalized moments of the distribution as part of the numerical solution process. These generalized moments can be viewed as a set of "numerically determined sufficient statistics" of the model. Our idea of the permutation invariant generalized moments coincides with the independent and contemporaneous work of Kahou et al. (2021), while we further explore the interpretation of the generalized moments and heterogeneity, and use them to study the impact of an unanticipated redistributional policy shock.

This work is also of relevance to the literature on machine learning-based algorithms for solving high dimensional dynamic programming problems in scientific computing (Han and E, 2016; Han et al., 2018; Fernández-Villaverde et al., 2020) and in macroeconomics (Fernández-Villaverde, Hurtado, and Nuno, 2019; Scheidegger and Bilionis, 2019; Maliar,

Maliar, and Winant, 2021; Valaitis and Villa, 2021; Maliar and Maliar, 2022; Azinovic, Gaegauf, and Scheidegger, 2022; Gopalakrishna, 2022; Gu, Laurière, Merkel, and Payne, 2023; Huang, 2023; Duarte, Duarte, and Silva, 2024). Recent notable contributions by Maliar et al. (2021); Azinovic et al. (2022) also use deep learning to solve HA models with aggregate shocks. DeepHAM differs from their work in the following aspects. First, we introduce generalized moments to make the high dimensional policy and value functions permutation invariant to the ordering of agents, which improves interpretability and computational efficiency. Second, in the recursive formulation, DeepHAM addresses the optimization problem with directly simulated paths, while Maliar et al. (2021) optimizes over an objective function constructed as a weighted sum of the Bellman residual and the first-order condition. Their setup requires a good approximation not only of the high dimensional value function itself, but also of partial derivatives of the value function, which are challenging to accurately obtain using neural networks. Azinovic et al. (2022) formulates the objective function as the weighted sum of the deviations from equilibrium conditions, which differs from our approach as well. In addition, this paper presents the first example of the use of machine learning-based algorithms to solve constrained efficiency problems in HA models with aggregate shocks.

The rest of this paper is organized as follows. Section 2 presents the DeepHAM method for solving a general HA model with aggregate shocks. Section 3 illustrates the use of DeepHAM on the classic Krusell-Smith model, and highlights the main features of the current approach. Sections 4 and 5 apply DeepHAM to more complex HA models and the constrained efficiency problem in HA models with aggregate shocks. Section 6 concludes the paper with some perspectives.

2 DeepHAM: A New Solution Method

We first present the DeepHAM method to solve the competitive equilibrium in a general HA model in Sections 2.1 to 2.3. Then we extend our setup to the constrained efficiency problem in Sections 2.4 and present the DeepHAM algorithm accordingly.

2.1 General Setup of HA Models

Consider a discrete time and infinite horizon economy consisting of N agents. N is large when we aim to solve a problem with a continuum of agents, and smaller when we aim to solve for the strategic equilibrium with finite agents.[2]

[2]Although our model assumes a finite number of agents, numerical results suggest that a choice of $N = 50$ can approximate the solution to HA models with a continuum of *ex ante* identical agents as in Krusell and Smith (1998) quite well. We have also tested values of $N = 100, 200, ..., 3000$, and got similar solution results.

For agent i, her state dynamics (i.e., law of motion for individual states) which usually come from the agent's budget constraint, are given by:

$$s_{t+1}^i = f(s_t^i, c_t^i, X_t, \bar{\mathbb{S}}_t; z_{t+1}^i), \quad i = 1, \ldots, N. \tag{1}$$

Here $s_t^i \in \mathbb{R}^{d_s}$ and $c_t^i \in \mathbb{R}^{d_c}$ denote the state and decision (control) of agent i at period t. $z_t^i \in \mathbb{R}^{d_z}$ denotes the idiosyncratic shock at period t, and is a subvector of s_t^i. The set $\bar{\mathbb{S}}_t = \{(s_t^1, c_t^1), (s_t^2, c_t^2) \ldots, (s_t^N, c_t^N)\}$ denotes the (unordered) set of all agents' state-control pairs. Similarly, we use $\mathbb{S}_t = \{s_t^1, s_t^2, \ldots, s_t^N\}$ to denote the set of agent states. We will also frequently use $S_t = (s_t^1, s_t^2, \ldots, s_t^N)$ to denote the (ordered) vector of agent states. $X_t \in \mathbb{R}^{d_X}$ denotes aggregate state variables excluding $\mathbb{S}_t$. Here, the law of motion of agent states (1) combines the agent's budget constraint, together with other optimization and market clearing conditions that characterize the aggregate prices in the budget constraint. For example, in Krusell and Smith (1998), the household's wealth state in the next period depends on current wealth and consumption, as well as current prices of labor and capital. $\bar{\mathbb{S}}_t$ is included as an input to the law of motion f for the individual state, because prices are functions of (the mean of) the wealth distribution according to the representative firm's optimization conditions and market clearing conditions, and the wealth distribution is a subset of $\bar{\mathbb{S}}_t$.

The control variables are subject to inequality constraints,

$$h_l(s_t^i, X_t, \mathbb{S}_t) \leq c_t^i \leq h_u(s_t^i, X_t, \mathbb{S}_t), \quad i = 1, \ldots, N, \tag{2}$$

where h_l, h_u are vector functions with d^c-dimensional output, and the dynamics of X_t are modeled by

$$X_{t+1} = g(X_t, \bar{\mathbb{S}}_t; Z_{t+1}), \tag{3}$$

where $Z_t \in \mathbb{R}^{dz}$ denotes the aggregate shock, and is usually a subvector of X_t. The aggregate state variable X_t also includes other aggregate quantities that affect the law of motion for individual states (1), but cannot be written only as functions of the collection of state-control pairs $\bar{\mathbb{S}}_t$. An example of X_t that contains more information than Z_t is presented in Section 4.

Here we have indicated that f, g, h_l, h_u depend only on the sets $\bar{\mathbb{S}}_t$ or $\mathbb{S}_t$, not the ordering of the agents, i.e., the state dynamics (1) are invariant to the ordering of the agents in the economy. This is a consequence of the "mean-field" character of the interaction between agents. "Mean-field" is a concept that originated in physics, and describes the situation

In the more general case, a proper choice of N may depend on the nature of the model. In Section 3.3.1, we study a model with infrequent idiosyncratic shocks using N up to 3000.

when the agents, or particles, interact with each other not directly, but through an empirical distribution that all the agents contribute to equally. A typical interaction form of mean-field type is through endogenous aggregate variables $O_t \in \mathbb{R}^{d_O}$ defined by

$$O_t = \frac{1}{N}\sum_{i=1}^{N} \mathcal{O}(s_t^i, c_t^i).$$

Here O_t can be considered a function of $\bar{\mathbb{S}}_t$ as it is permutation invariant to the ordering of agents. Examples of O_t include the first or other moments of individual states. In this paper, we assume the dependence of f, g, h_l, h_u on $\bar{\mathbb{S}}_t$ or $\mathbb{S}_t$ can be written in explicit functional forms. The Krusell-Smith model mentioned above is such an example. This point will be more clear in our presentation of the concrete examples in Sections 3 to 5.[3]

According to the above description, $(X_t, \mathbb{S}_t)$ completely characterizes the state of the whole economy. Mathematically, we are interested in how agents should make decisions $c_t^i = \mathcal{C}(s_t^i, X_t, \mathbb{S}_t)$ through the decision rule $\mathcal{C}$ to achieve optimality.[4]

In the setting of competitive equilibrium, each agent i seeks to maximize her discounted lifetime utility:

$$\mathbb{E}_{\mu,\mathcal{E}} \sum_{t=0}^{\infty} \beta^t u(c_t^i). \tag{4}$$

Here $u(\cdot)$ is the utility function and $\beta \in (0,1)$ is the discount factor. μ is the full state distribution $(X, \mathbb{S})$ at the initial time. We use $\mu(\mathcal{C})$ to denote the stationary distribution of $(X, \mathbb{S})$ when every agent employs the decision rule $\mathcal{C}$ and assume that such a stationary distribution always exists. The expectation is also taken with respect to both the idiosyncratic and aggregate shocks over time, which is denoted as $\mathcal{E} \equiv \{\{z_t^i\}_i, Z_t\}_{t\geq 1}$.

We say that $\mathcal{C}^*$ is an optimal policy in the competitive equilibrium if, for $\forall i \in \{1, \ldots, N\}$,

[3]The assumption on explicit function forms of f, g, h_l, h_u excludes HA models with nontrivial market clearing condition (Algan et al., 2014, Section 4). Recent papers by Azinovic and Žemlička (2023); Gopalakrishna et al. (2024) incorporate non-trivial market clearing conditions into the design of the deep learning algorithms, and their ideas could also be incorporated into the DeepHAM framework.

[4]Note here $\mathcal{C}$ is common across agents. This naturally applies for HA models with *ex ante* homogeneous agents like Krusell and Smith (1998). Similarly, the value function V defined later is common across agents as well. For models with *ex ante* heterogeneous agents, we may introduce additional individual state variables such that $\mathcal{C}$ is common across agents.

$\mathcal{C}^*$ solves agent i's problem,

$$\begin{aligned}
&\max_{\mathcal{C}} \ \mathbb{E}_{\mu(\mathcal{C}^*),\mathcal{E}} \sum_{t=0}^{\infty} \beta^t u(s_t^i, c_t^i), \\
&\text{s.t. (1)(2)(3) hold, } c_t^i = \mathcal{C}(s_t^i, X_t, \mathbb{S}_t), \\
&\text{given } c_t^j = \mathcal{C}^*(s_t^j, X_t, \mathbb{S}_t), j = 1, \ldots, N, j \neq i.
\end{aligned}$$

Note that in contrast to the perturbation method in Reiter (2009), which only computes the solution around the stationary equilibrium in the absence of aggregate shocks, a global solution method seeks to find the solution according to the stationary distribution $\mu(\mathcal{C}^*)$ of the economy, which may significantly differ from the stationary equilibrium without aggregate shocks (Kekre and Lenel, 2021; Bhandari et al., 2021).

We now make two remarks about the general setup we present above.

Remark 1 (Discrete time setup). Throughout this paper, we formulate HA models in discrete time. Founded upon the methodological framework of Achdou, Han, Lasry, Lions, and Moll (2017), the study of HA models in continuous time has received a great deal of interest in recent years. Compared to discrete time, a continuous time setup admits explicit expectation integral formulas (with respect to specific forms of idiosyncratic shocks) and efficient numerical algorithms for solving the corresponding systems of partial differential equations (PDEs). However, when there are aggregate shocks, we need to derive and solve the stochastic PDE systems (Carmona and Delarue, 2018) to obtain the global solution, which is challenging. Here we use a discrete time setup so that the problem is easier to solve and more general forms of shocks can be included.

Remark 2 (Infinite horizon). We restrict the setup to the infinite horizon for the sake of concision when introducing the algorithm. As seen in Section 2.3, our method can also handle finite horizon problems such as life cycle models with only minor modification.

2.2 Representation of the Agent Distribution and Generalized Moments

The exposition of DeepHAM comprises two steps. The first is the introduction of generalized moments to replace the full agent distribution. This can be considered a model reduction step. The second is the introduction of an algorithm to solve the reduced model. We remark that the first step is of independent interest: the reduced model itself is a reliable and interpretable model that can be used as a general starting point for performing economic analysis. We discuss the idea of the first step in this subsection and present the detailed

algorithm in the next subsection.

In HA models, a key question is what variables should be used to represent the whole economy. In algorithmic terms, these are what should be fed into the policy and value function approximators as input. Clearly, the individual state s_t^i and the aggregate state variable X_t should be taken into account. The main question is therefore how to represent the empirical distribution $\mathbb{S}_t = \{s_t^1, s_t^2, \ldots, s_t^N\}$, which also affects the dynamics of s_t^i and X_t. On this point, the existing literature generally uses one of the following two approaches.

The first approach uses the full state vector S_t (see, e.g., Maliar et al., 2021), the full information of the distribution, to characterize the optimal policy and value functions. However, there are two caveats. First, the dimension of S_t is proportional to N. It is thus extremely expensive to deal with economies with a large number of agents. Second, the agent's optimal policy and value function should be invariant to the ordering of other agents' states. A function approximation form taking S_t as the direct input cannot straightforwardly impose this restriction and must inefficiently process the irrelevant information of the agents' ordering.

The second approach uses finite moments, usually the first moment of $\mathbb{S}_t$. This is the approach adopted by the KS method. It overcomes the two caveats discussed above of using the full vector S_t. However, the moments chosen may not carry the full information necessary for an agent to evaluate her current environment and act optimally. Under this simplification, the solution may deviate from the ground truth, especially in complex HA models.

To go beyond the above limitations, we introduce a class of generalized aggregate variables $Q_t \in \mathbb{R}^{d_Q}$ into the state vector:

$$Q_t = \frac{1}{N} \sum_{i=1}^{N} \mathcal{Q}(s_t^i).$$

Here the basis function $\mathcal{Q}$ may take a pre-specified functional form, or it may be a general basis function with variational parameters. For pre-specified functional forms, for example, it could be the identity function, making Q_t the first moment. $\mathcal{Q}$ might also be an indicator function of whether an agent is at the liquidity constraint, so that Q_t captures the share of hand-to-mouth agents (Kaplan, Violante, and Weidner, 2014) in the economy. A Q_t based on pre-specified $\mathcal{Q}$ nests the moment representation we discuss above. A general basis function $\mathcal{Q}$ can be parameterized with neural networks (Han et al., 2022) and the optimal representation solved for in the algorithm. When $\mathcal{Q}$ is a general basis function, we call the components of the resulting Q_t *"generalized moments"*.[5]

With Q_t as the representation of the distribution, we use (s_t^i, X_t, Q_t) as the state vector taken as input by the policy and value function approximations. Conceptually, one can think

[5]A brief mathematical introduction to neural networks is presented in Appendix A.

of this in the following two ways.

First, instead of parameterizing the mapping $[(s_t^i, X_t, \mathbb{S}_t) \mapsto \text{ output}]$ with neural network models, we decompose it into $[(s_t^i, X_t, \mathbb{S}_t) \mapsto (s_t^i, X_t, Q_t) \mapsto \text{ output}]$ and parameterize the two components by two neural networks, the first step is an encoding network and the second step is a fitting network. In this sense, by specifying the dimension d_Q, we ensure that the complexity of the neural networks does not increase rapidly as N increases. Furthermore, the final policy functions are permutation invariant by design. In this way, both shortcomings mentioned above are overcome.[6]

Second, Q_t shares a similar representation to O_t, and can be interpreted as a vector of generalized moments. If $\mathcal{Q}$ is parameterized by a set of specific functional forms informed by structural details, Q_t are selected from a large set of interpretable aggregate moments. If $\mathcal{Q}$ is directly parameterized by neural networks, optimizing $\mathcal{Q}$ can guide the agent to finding the generalized moments Q_t most relevant to their decision making. Compared to the KS method, generalized moments have the flexibility to more closely capture those features of the whole economy most relevant to the optimal decision rules, obtaining a more accurate macroeconomic model without sacrificing interpretability. In this regard, the generalized moments can be viewed as a set of "numerically determined sufficient statistics" of the model, which is the numerical counterpart of those analytical sufficient statistics we commonly see in structural models (Chetty, 2009; Auclert, 2019).[7]

In this paper, we use two separate sets of generalized moments, Q_t^c and Q_t^V, to extract the distribution information for the policy and value functions, respectively. This choice simplifies their updating rule. Algorithmically, when we fed Q_t^c/Q_t^V into the policy/value functions, the variational parameters of the basis and policy/value parameters can be trained jointly end-to-end through the corresponding objective function. Using shared generalized moments for both the policy and value functions will be investigated in future work.

2.3 Solution Method for Competitive Equilibrium

We first describe the algorithm for solving for the competitive equilibrium of economies of the form described in Section 2.1. The algorithm for solving the constrained efficiency problem is quite similar and will be discussed in Section 2.4. To solve the model, we rewrite

[6]Such a two-step decomposition through generalized moments is also closely linked to the recent literature (Zaheer et al., 2017) of approximating permutation invariant functions through deep neural networks. Mathematically, it has been proved that any permutational invariant functions in multi-dimensional space can be approximated by such a two-step decomposition. See the detailed mathematical statement in Appendix A.

[7]In a semi-structural framework, Chang et al. (2021) propose a nonparametric method to represent cross-sectional distribution with sieve coefficients and study their interaction with aggregate macroeconomic variables. Their method is built in a functional vector-autoregressive model, which differs from ours.

agents' objective function based on the dynamic programming principle over T periods. In the competitive equilibrium, for $\forall i \in \{1, \ldots, N\}$, the policy function $\mathcal{C}^*$ solves agent i's problem,

$$\begin{aligned} \max_{\mathcal{C}} \ & \mathbb{E}_{\mu(\mathcal{C}^*),\mathcal{E}} \left[\sum_{t=0}^{T-1} \beta^t u\left(s_t^i, c_t^i\right) + \beta^T V(s_T^i, X_T, \mathbb{S}_T) \right], \\ & \text{s.t. (1)(2)(3) hold, } c_t^i = \mathcal{C}(s_t^i, X_t, \mathbb{S}_t), \\ & \text{given } c_t^j = \mathcal{C}^*(s_t^j, X_t, \mathbb{S}_t), j = 1, \ldots, N, j \neq i, \end{aligned} \tag{5}$$

where the value function V is defined by

$$\begin{aligned} V(s_0^i, X_0, \mathbb{S}_0) = \ & \mathbb{E}_{\mathcal{E}} \left[\sum_{t=0}^{\infty} \beta^t u\left(s_t^i, c_t^i\right) \Big| X_0, \mathbb{S}_0 \right], \\ & \text{s.t. (1)(2)(3) hold, } c_t^i = \mathcal{C}^*(s_t^i, X_t, \mathbb{S}_t), i = 1, \ldots, N. \end{aligned} \tag{6}$$

As in (4), there are two types of randomness in (5): the random initial state drawn from the stationary distribution $\mu(\mathcal{C}^*)$, and the randomness of idiosyncratic and aggregate shocks over time denoted as $\mathcal{E}$.

The overall idea of DeepHAM is an iterative procedure starting from the initial guess of the policy $\mathcal{C}_0$, as presented in Algorithm 1. Each iteration includes three steps that we will discuss in detail: (a) prepare the stationary distribution $\mu(\mathcal{C})$; (b) update the value function according to (6); (c) optimize the policy function according to (5). We refer to one such high-level iteration step as a *round* and repeat N_k rounds until convergence. It is similar to the conventional value function iteration algorithm (Ljungqvist and Sargent, 2018, Chapter 3), except for the following two aspects.

First, we parameterize both value and policy functions with neural networks, each of which nests two sub-networks with a feedforward architecture: one approximates the basis function $\mathcal{Q}$, the other approximates the mapping from (s_t^i, X_t, Q_t) to policy or value function values. The two sub-networks are trained together, and we get the generalized moments once we obtain the basis function after optimization.[8]

Second, we solve for optimal policy to maximize the total utility in (5) over Monte Carlo simulations for T periods, instead of one period, which is typically used in the conventional value function iteration algorithm. When the state vector is high dimensional, it is computationally expensive or even infeasible to update the policy with one period calculation in the

[8]If we choose to use some pre-specified basis to define Q, such as the first moment, we will only have the second sub-network in both the policy and value functions.

whole state space. Instead, we update the parameters of the policy function neural network to maximize the expected total utility in (5) over simulated paths. We choose $T > 1$ such that the error in the value function, which is discounted by β^T, will have little impact on the policy function optimization.

Algorithm 1 DeepHAM for solving the competitive equilibrium

Require: Input: the initial policy $\mathcal{C}_0$, the initial value and policy neural networks with parameters Θ^V and Θ^C, respectively

for $k = 1, 2, \ldots, N_k$ **do**
  prepare the stationary distribution $\mu(\mathcal{C}_{k-1})$ according to the policy $\mathcal{C}_{k-1}$
  for $m = 1, 2, \ldots, N_{m_1}$ **do** ▷ update the value function
    sample N_{b_1} samples of $(X_0, \mathbb{S}_0)$ from $\mu(\mathcal{C}_{k-1})$
    compute the realized total utility in (9) through a single simulated path
    use the empirical version of (10) to compute the gradient ∇_{Θ_V}
    update Θ^V with ∇_{Θ_V}
  end for
  for $m = 1, 2, \ldots, N_{m_2}$ **do** ▷ optimize the policy function
    sample N_{b_2} samples of $(X_0, \mathbb{S}_0)$ from $\mu(\mathcal{C}_{k-1})$
    use the empirical version of (11) to compute the gradient ∇_{Θ_C}
    update Θ^C with ∇_{Θ_C}
  end for
  define $\mathcal{C}_k$ according to (12)
end for

Now we further explain the details of the three main steps of DeepHAM in the k-th round.

Preparing the stationary distribution. We simulate the economy (1)(3) forward for sufficiently many periods under the policy $\mathcal{C}_{k-1}$ to find the stationary distribution $\mu(\mathcal{C}_{k-1})$ of the economy. Then we store enough samples of $(X_0, \mathbb{S}_0)$ according to $\mu(\mathcal{C}_{k-1})$, which will be used as the initial condition for later updating of the value and policy functions.

Updating the value function. Given the policy $\mathcal{C}_{k-1}$, updating the value function can be formulated as a supervised learning problem. Denote the parameters in the value function neural network by $\Theta^V = (\Theta^{VQ}, \Theta^{VO})$, where Θ^{VQ} are the parameters in the general basis function defining Q_t^V and Θ^{VO} are the parameters in the function that maps (s_t^i, X_t, Q_t^V) to value outcomes. Our approximation to the value function can thus be written,

$$V_{\text{NN}}(s_t^i, X_t, \mathbb{S}_t; \Theta^V) \coloneqq \widetilde{V}_{\text{NN}}(s_t^i, X_t, Q_t^V; \Theta^{VO}) = \widetilde{V}_{\text{NN}}(s_t^i, X_t, \frac{1}{N}\sum_{i=1}^{N} \mathcal{Q}_{\text{NN}}(s_t^i; \Theta^{VQ})); \Theta^{VO}). \quad (7)$$

We want to use V_{NN} to approximate the agents' expected lifetime utility under the policy $\mathcal{C}_{k-1}$, i.e.,

$$V_{\text{NN}}(s_t^i, X_t, \mathbb{S}_t; \Theta^V) \approx \mathbb{E}_{\mathcal{E}}\left[\sum_{\tau=0}^{\infty} \beta^\tau u(s_{t+\tau}^i, c_{t+\tau}^i) \,\middle|\, s_t^i, X_t, \mathbb{S}_t\right]. \tag{8}$$

However, evaluating the expectation in (8) is still computationally expensive. To reduce the computational cost, given each sample $(s_0^i, X_0, \mathbb{S}_0)$ from the stationary distribution $\mu(\mathcal{C}_{k-1})$, we only simulate a single path for T_{simul} periods (with T_{simul} sufficiently large) under the policy $\mathcal{C}_{k-1}$ to get the truncated realized total utility

$$\widehat{V}^i = \sum_{\tau=0}^{T_{\text{simul}}} \beta^\tau u(s_\tau^i, c_\tau^i). \tag{9}$$

Note that $\widehat{V}_t^i$ is a random variable influenced by the realization of idiosyncratic and aggregate shocks, and the initial state. The true value function minimizes the expected difference with the realized total utility. Thus, we solve the following regression problem to update the value function:

$$\min_{\Theta^V} \mathbb{E}_{\mu(\mathcal{C}_{k-1}),\mathcal{E}}\left[V_{\text{NN}}(s_0^i, X_0, \mathbb{S}_0; \Theta^V) - \widehat{V}^i\right]^2. \tag{10}$$

We use the stochastic gradient descent (SGD) algorithm to solve (10). Specifically, in each update step, we sample N_{b_1} samples of $(X_0, \mathbb{S}_0)$ from $\mu(\mathcal{C}_{k-1})$, use the empirical version of (10) to compute the gradient with respect to Θ^V by backpropagation, and update Θ^V accordingly. We repeat N_{m_1} steps to achieve convergence. As $\Theta^V = (\Theta^{VQ}, \Theta^{VO})$, we also obtain, at the end of the update, the updated basis function $\mathcal{Q}_{\text{NN}}(\cdot; \Theta^{VQ})$ and the generalized moments Q_t at the same time.[9]

Optimizing the policy function. In the competitive equilibrium, the policy function is updated iteratively following the spirit of fictitious play (Brown, 1951). A similar idea has been used in Han and Hu (2020) and Hu (2021) to solve stochastic differential games based on neural networks. Similar to our approach to the value function, we will update the parameters associated with the policy function neural network through the stochastic gradient descent algorithm. We call each update a "play". In each "play", we fix everyone but agent $i = 1$'s policy as that from the last play, and consider agent $i = 1$'s utility maximization problem to update the neural network parameters, to get the new policy in this "play". All agents then adopt the new policy in this "play". We repeat the "plays"

[9]Backpropagation is an algorithm allowing an efficient computation of all partial derivatives of the neural network (composition of a series of functions) with respect to its parameters.

until convergence.[10]

For the utility maximization problem of agent $i = 1$, the algorithm essentially builds upon the one proposed in Han and E (2016): optimizing the parameters of the policy function neural network over simulated paths. Given the updated value function $V_{\text{NN}}(s_t^i, X_t, \mathbb{S}_t; \Theta^V)$ in the same round and other agents' policy function from the last "play", agent $i = 1$ aims to solve

$$\max_{\Theta^C} \mathbb{E}_{\mu(\mathcal{C}_{k-1}),\mathcal{E}} \left[\sum_{t=0}^{T-1} \beta^t u(s_t^i, c_t^i) + \beta^T V_{\text{NN}}(s_T^i, X_T, \mathbb{S}_T; \Theta^V) \right], \tag{11}$$

with her policy parameterized by neural networks in the form of

$$\begin{aligned} c_t^i &= \mathcal{C}(s_t^i, X_t, \mathbb{S}_t; \Theta^C) \\ &= \big(h_u(s_t^i, X_t, \mathbb{S}_t) - h_l(s_t^i, X_t, \mathbb{S}_t)\big) \odot c_{\text{NN}}\left(s_t^i, X_t, \mathbb{S}_t; \Theta^C\right) + h_l(s_t^i, X_t, \mathbb{S}_t). \end{aligned} \tag{12}$$

Here the outputs of $h_u(\cdot), h_l(\cdot)$, and $c_{\text{NN}}(\cdot)$ are d^c dimensional, and $\odot$ denotes element-wise multiplication. We use the sigmoid function $\frac{1}{1+e^{-x}} \in [0, 1]$ as the last composed function of $c_{\text{NN}}(\cdot)$, so that the inequality constraints (2) are always satisfied. For a mathematical introduction to the composition structure of neural networks, see Appendix A. Similar to the value function, the parameters in the policy function neural network $\Theta^C = (\Theta^{CQ}, \Theta^{CO})$, where Θ^{CQ} are parameters in the general basis function and Θ^{CO} are parameters in the function that maps (s_t^i, X_t, Q_t) to policy outcomes. So we have

$$c_{\text{NN}}(s_t^i, X_t, \mathbb{S}_t; \Theta^C) = \widetilde{c}_{\text{NN}}(s_t^i, X_t, Q_t^C; \Theta^{CO}) = \widetilde{c}_{\text{NN}}(s_t^i, X_t, \frac{1}{N}\sum_{i=1}^{N} \mathcal{Q}_{\text{NN}}(s_t^i; \Theta^{CQ})); \Theta^{CO}). \tag{13}$$

Once more, we use the stochastic gradient descent (SGD) algorithm corresponding to (11). We sample N_{b_2} samples of $(X_0, \mathbb{S}_0)$ from $\mu(\mathcal{C}_{k-1})$ as the initial conditions, use the empirical version of (11) to compute the gradient with respect to Θ^C, and update Θ^C accordingly. The gradient with respect to Θ^C can be obtained by backpropagation as well, because all the component functions in (11)(12)(13) are explicit and differentiable; see Figure 1 for the computational graph corresponding to (11).

From the above description, we can see one merit of DeepHAM: its ready ability to handle models with aggregate shocks. The algorithm remains almost the same when solving models with aggregate shocks or without. In contrast, the continuous time PDE approach (Achdou et al., 2017) can solve models without aggregate shocks efficiently (in the low-dimensional case), but faces challenges in the presence of aggregate shocks.

[10] In a more general setup, we can also fix the other agents' policies for several "plays" and then update from the agent $i = 1$'s policy.

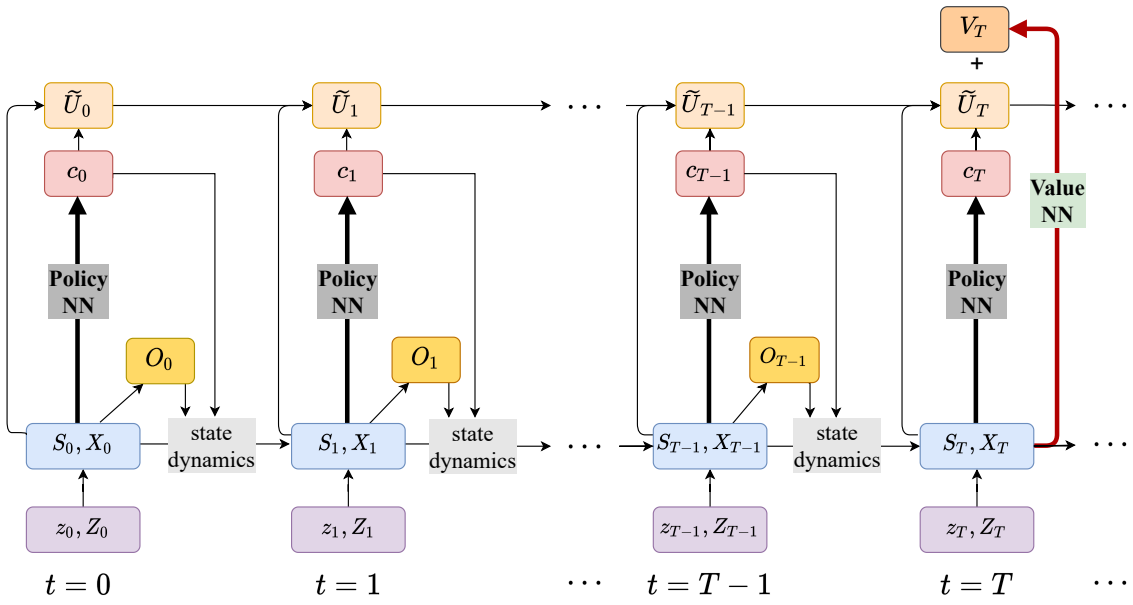


Figure 1: Computational graph to solve general HA models using DeepHAM. S_t, z_t, and c_t denote the collection of all agents' states, idiosyncratic shocks, and decisions at time t, respectively. Z_t denotes aggregate shocks at time t. $\widetilde{U}_t$ denotes the collection of all agents' cumulative utilities up to period t, i.e., $\widetilde{U}_t^i = \sum_{\tau=0}^{t} \beta^\tau u\left(s_\tau^i, c_\tau^i\right)$. The policy function neural networks are the same at each period of time.

2.4 Extension to Constrained Efficiency Problem

In the constrained efficiency problem, a benevolent social planner seeks to find a policy rule $\mathcal{C}$, determining each agent's decision variable c_t^i, in order to maximize the discounted sum of social welfare $\Omega(\bar{\mathbb{S}}_t)$. The social welfare depends on the collection of all agents' state-control pairs:

$$\max_{\mathcal{C}} \ \mathbb{E}_{\mu(\mathcal{C}),\mathcal{E}} \sum_{t=0}^{\infty} \beta^t \Omega(\bar{\mathbb{S}}_t),$$
$$\text{s.t. } (1)(2)(3) \text{ hold, } c_t^i = \mathcal{C}(s_t^i, X_t, \mathbb{S}_t), i = 1, \ldots, N.$$

Here the social welfare function can take the utilitarian form $\Omega(\bar{\mathbb{S}}_t) = \frac{1}{N}\sum_{i=1}^{N} u(s_t^i, c_t^i)$, or $\Omega(\bar{\mathbb{S}}_t) = \sum_{i=1}^{N} \omega_i u(s_t^i, c_t^i)$ with Negishi weights $\omega_i = \frac{u_c(s_t^i, c_t^i)}{\sum_{i=1}^{N} u_c(s_t^i, c_t^i)}$ (Bhandari et al., 2021), or other general forms.

The overall procedure for solving the constrained efficiency problem is the same as that for solving the competitive equilibrium, as presented in Algorithm 2: each round consists of (a) preparing the stationary distribution, (b) updating the value function, and (c) optimizing the policy function. Below, we explain the three steps in the k-th round in detail and mainly

highlight the differences between these and the procedure for solving for the value and policy objectives in competitive equilibrium.

Algorithm 2 DeepHAM for solving the constrained efficiency problem

Require: Input: the initial policy $\mathcal{C}_0$, the initial value and policy neural networks with parameters Θ^V and Θ^C, respectively

for $k = 1, 2, \ldots, N_k$ **do**
  prepare the stationary distribution $\mu(\mathcal{C}_{k-1})$ according to the policy $\mathcal{C}_{k-1}$
  for $m = 1, 2, \ldots, N_{m_1}$ **do** ▷ update the value function
    sample N_{b_1} samples of $(X_0, \mathbb{S}_0)$ from $\mu(\mathcal{C}_{k-1})$
    compute the realized total social welfare in (15) through a single simulated path
    use the empirical version of (16) to compute the gradient ∇_{Θ_V}
    update Θ^V with ∇_{Θ_V}
  end for
  for $m = 1, 2, \ldots, N_{m_2}$ **do** ▷ optimize the policy function
    sample N_{b_2} samples of $(X_0, \mathbb{S}_0)$ from $\mu(\mathcal{C}_{k-1})$
    use the empirical version of (17) to compute the gradient ∇_{Θ_C}
    update Θ^C with ∇_{Θ_C}
  end for
  define $\mathcal{C}_k$ according to (12)
end for

Preparing the stationary distribution. This is done exactly the same as for the competitive equilibrium problem. We simulate the economy (1)(3) forward for sufficiently many periods under the policy $\mathcal{C}_{k-1}$ to find the stationary distribution $\mu(\mathcal{C}_{k-1})$ of the economy. Then we store enough samples of $(X_0, \mathbb{S}_0)$ according to $\mu(\mathcal{C}_{k-1})$, which will be used as the initial condition for later updating of the value and policy functions.

Updating the value function. Given the policy $\mathcal{C}_{k-1}$, we want to use V_{NN} to approximate the expected total social welfare under the policy $\mathcal{C}_{k-1}$, i.e.,

$$V_{\text{NN}}(X_t, \mathbb{S}_t; \Theta^V) \approx \mathbb{E}_{\mathcal{E}} \left[\sum_{\tau=0}^{\infty} \beta^\tau \Omega(\bar{\mathbb{S}}_{t+\tau}) \,\middle|\, X_t, \mathbb{S}_t \right], \tag{14}$$

where

$$V_{\text{NN}}(X_t, \mathbb{S}_t; \Theta^V) = V_{\text{NN}}(X_t, Q_t^V; \Theta^{VO}) = V_{\text{NN}}(X_t, \frac{1}{N} \sum_{i=1}^{N} \mathcal{Q}_{\text{NN}}(s_t^i; \Theta^{VQ}); \Theta^{VO}).$$

Similarly, to avoid the computational cost of evaluating the expectation in (14) given each sample $(X_0, \mathbb{S}_0)$ from the stationary distribution $\mu(\mathcal{C}_{k-1})$, we only simulate a single path for

T_{simul} periods (with T_{simul} sufficiently large) under the policy $\mathcal{C}_{k-1}$ to obtain the truncated realized total social welfare

$$\widehat{V} = \sum_{\tau=0}^{T_{\text{simul}}} \beta^{\tau} \Omega(\bar{\mathbb{S}}_{\tau}). \tag{15}$$

Then we only need to solve the following regression problem to update the value function:

$$\min_{\Theta^V} \mathbb{E}_{\mu(\mathcal{C}_{k-1}),\mathcal{E}} \left[V_{\text{NN}}(X_0, \mathbb{S}_0; \Theta^V) - \widehat{V} \right]^2. \tag{16}$$

This can be done using stochastic gradient descent in the same way as for (10).

Optimizing the policy function. In order to find the constrained optimum, we need to update the policy function by solving

$$\max_{\Theta^C} \mathbb{E}_{\mu(\mathcal{C}_{k-1}),\mathcal{E}} \left[\sum_{t=0}^{T-1} \beta^t \Omega(\bar{\mathbb{S}}_t) + \beta^T V_{\text{NN}}(X_T, \mathbb{S}_T; \Theta^V) \right]. \tag{17}$$

Compared to the policy function optimization in the competitive equilibrium problem, here we get rid of the fictitious play step and instead optimize all the agents' policies simultaneously to maximize the total social welfare. The optimization problem (17) can be solved in the same way as for the problem (11) with the stochastic gradient descent algorithm. The gradient with respect to Θ^C can be obtained by backpropagation in the same computational graph as in Figure 1.

The above description demonstrates that the DeepHAM Algorithm 2 for solving the constrained efficiency problem is identical to Algorithm 1 for the competitive equilibrium, except for the two differences in lines 6 and 11, in which we use different objectives for the value function and policy function corresponding to the different setting. Meanwhile, the pipeline of data sampling and optimization methods for these different objectives is exactly the same. In this sense, DeepHAM can solve the constrained efficiency problem as easily as the competitive equilibrium problem.

2.5 Remarks on the Method

In this subsection, we further discuss the distinctive features of the DeepHAM method, particularly in comparison to other deep learning-based solution approaches and traditional global solution methods like the KS method. We focus on two critical aspects: the optimization objective and its evaluation, as well as the representation of distributions. These

features are essential to understanding how DeepHAM differs from other methods and why it offers advantages in certain applications.

First, in the policy optimization step, DeepHAM employs objective functions (11) and (17) that are based on cumulative utility unrolled over multiple periods $T > 1$. We update the parameters of the policy function neural networks to optimize these objective functions over a large number of simulated paths. This objective formulation sets DeepHAM apart from other deep learning-based approaches like Maliar et al. (2021); Azinovic et al. (2022), which typically rely on first-order equilibrium conditions as their objective functions. This allows DeepHAM to address problems where first-order conditions are complex or insufficient, such as constrained efficiency problems or optimal policy design.

It is also worthwhile to highlight our approach of directly optimizing the expected value over simulated paths, which allows us to solve the household or planner optimization problem through the computational graph in Figure 1. Conceptually, this means agents or planners formulate expectations of the future based on simulations rather than relying on a perceived law of motion as in the KS method. This approach aligns with a similar philosophy found in reinforcement learning. While reinforcement learning typically operates under the assumption that agents do not know the analytic form of transition dynamics and rewards (a model-free setting), DeepHAM can be seen as a model-based reinforcement learning algorithm tailored for economic models. This framework also enables the study of heterogeneous agent models that deviate from rational expectation equilibrium, broadening the scope of potential applications.

Second, DeepHAM introduces generalized moments to represent distributions. Unlike the classical KS method, which requires pre-specification of a few simple moments, our approach offers greater flexibility by automatically identifying the generalized moments most relevant to decision-making. Compared to methods that directly use the full state of all agents such as Maliar et al. (2021), our method is numerically more efficient, as the number of required neural network parameters would not increase much when the number of agents increases, whereas full state representation demands increasing parameters. Conceptually, the representation through generalized moments ensures that the derived policy is independent of the ordering of other agents in the state space. Moreover, this approach offers insights into dimension reduction, highlighting the factors that matter most for the overall economic system. In contrast, the full state representation lacks all these benefits. We provide a further numerical illustration of these points in Section 3.3.2.

3 DeepHAM for the Krusell-Smith Model

In this section, we illustrate DeepHAM on the classic Krusell-Smith model, and highlight the advantages of this method.

3.1 Model Setup

The setup follows Den Haan (2010). Household i's state $s_t^i = (a_t^i, z_t^i) \in \mathbb{R}^2$, with beginning-of-period wealth a_t^i, employment status $z_t^i \in \{0, 1\}$. The consumption $c_t^i \in \mathbb{R}$ is the control variable. Households have log utility over consumption. $Z_t \in \{Z^h, Z^l\}$ denotes aggregate productivity. The process z_t^i, Z_t follows a first-order Markov process. The aggregate state variable $X_t = Z_t$, so its dynamics (3) are trivial. The state dynamics of the household comes from the household budget constraint,

$$
\begin{aligned}
&a_{t+1}^i = (1 + r_t - \delta)a_t^i + [(1 - \tau_t)\bar{l}z_t^i + b(1 - z_t^i)]w_t - c_t^i, \\
&a_{t+1}^i \geq 0, \quad c_t^i \geq 0,
\end{aligned}
$$

where the net rate of return of capital is $r_t - \delta$, with depreciation rate δ. The factor prices r_t, w_t are determined by the first order condition (FOC) of the representative firm, which produces with a Cobb-Douglas technology $Y_t = Z_t K_t^\alpha L_t^{1-\alpha}$, in the competitive factor market,

$$
w_t = Z_t(1 - \alpha)(K_t/L_t)^\alpha, \quad r_t = Z_t\alpha(K_t/L_t)^{\alpha-1},
$$

with aggregate capital $K_t = \frac{1}{N}\sum_{i=1}^N a_t^i$ and labor supply $L_t = \bar{l}(L^h \mathbb{1}_{Z_t=Z^h} + L^l \mathbb{1}_{Z_t=Z^l})$, in which $\bar{l}$ is the time endowment of each agent. Unemployed agents ($z_t^i = 0$) receive unemployment benefits bw_t where b is the unemployment benefit rate. Employed agents ($z_t^i = 1$) earn after-tax labor income $(1 - \tau_t)\bar{l}w_t$, with tax rate $\tau_t = b(1 - L_t)/\bar{l}L_t$, such that government budget constraint always holds (total tax income equals unemployment benefits). This completes the specification of (1). The borrowing constraint and non-negative consumption constraint specifies (2). The calibration of the model follows Den Haan (2010) and is presented in Appendix B.1.1.

3.2 Results

We solve the Krusell-Smith model described above in the case of $N = 50$ using DeepHAM. We have tried other choices of $N = 100, 200, ...$, and we find $N = 50$ is large enough to approximate the solution to the Krusell-Smith model. The computational graph for

this problem is shown in Figure 2. Detailed learning parameter values are provided in Appendix E.

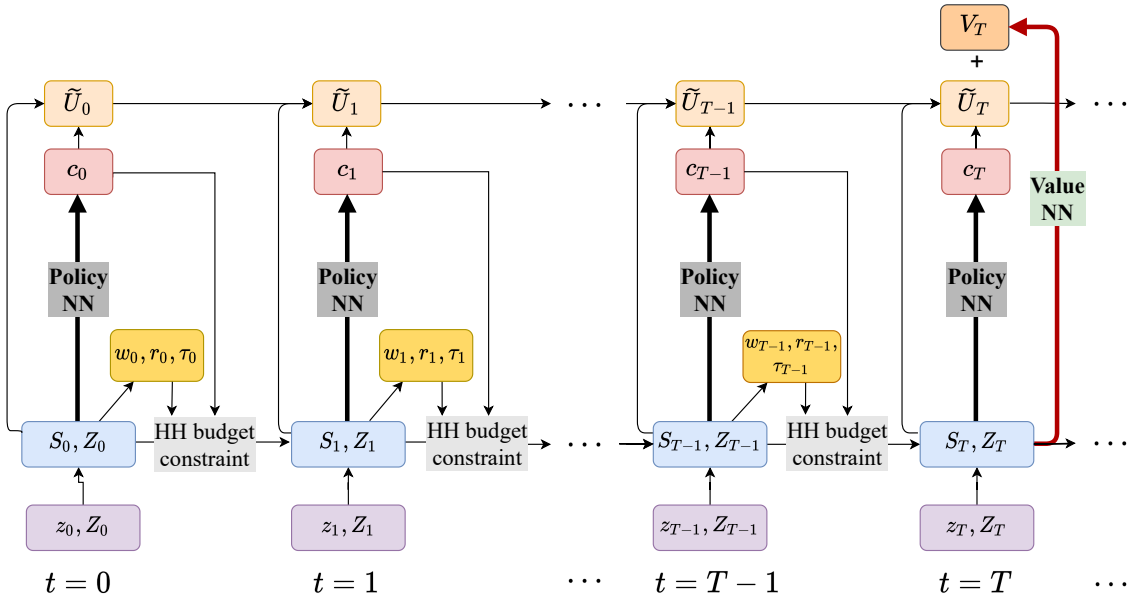


Figure 2: Computational graph to solve Krusell and Smith (1998) using DeepHAM. S_t, z_t, c_t, and $\widetilde{U}_t$ denote the collection of all agents' states, idiosyncratic shocks, decisions, and cumulative utilities at time t, respectively. Z_t denotes aggregate shocks at time t. Aggregate prices w_t, r_t are determined by FOCs of the representative firm in the competitive factor market. Income tax rate τ_t depends on the aggregate shock Z_t and is pinned down in the government budget constraint.

3.2.1 Solution Accuracy

In Table 2, we compare the Bellman equation errors (defined in Appendix C) of DeepHAM to the same error for the KS method implemented in Maliar et al. (2010). For DeepHAM, we present accuracy measures for the cases where we include (1) the first moment of the household wealth distribution and (2) one generalized moment of the household wealth distribution in the state variable. We present the standard deviation of the Bellman errors from multiple runs of the numerical algorithm in the last column of Table 2. We see that all results are statistically significant.[11]

As we see in Table 2, the solutions obtained using DeepHAM are highly accurate. Compared to the KS method, DeepHAM with the first moment in the state vector reduces the Bellman equation error by 27.2%. DeepHAM with one generalized moment reduces the error

[11]Following the moment construction in Krusell and Smith (1998), here the generalized moments are constructed on the wealth distribution, rather than the joint distribution of wealth and employment status.

Method and Moment Choice	Bellman error	Std of error
KS Method (Maliar et al., 2010)	0.0253	0.0002
DeepHAM with 1st moment	0.0184	0.004
DeepHAM with 1 generalized moment	0.0153	0.0006
DeepHAM with 2 generalized moments	0.0152	0.0011

Table 2: Comparison of solution accuracy for Krusell-Smith problem

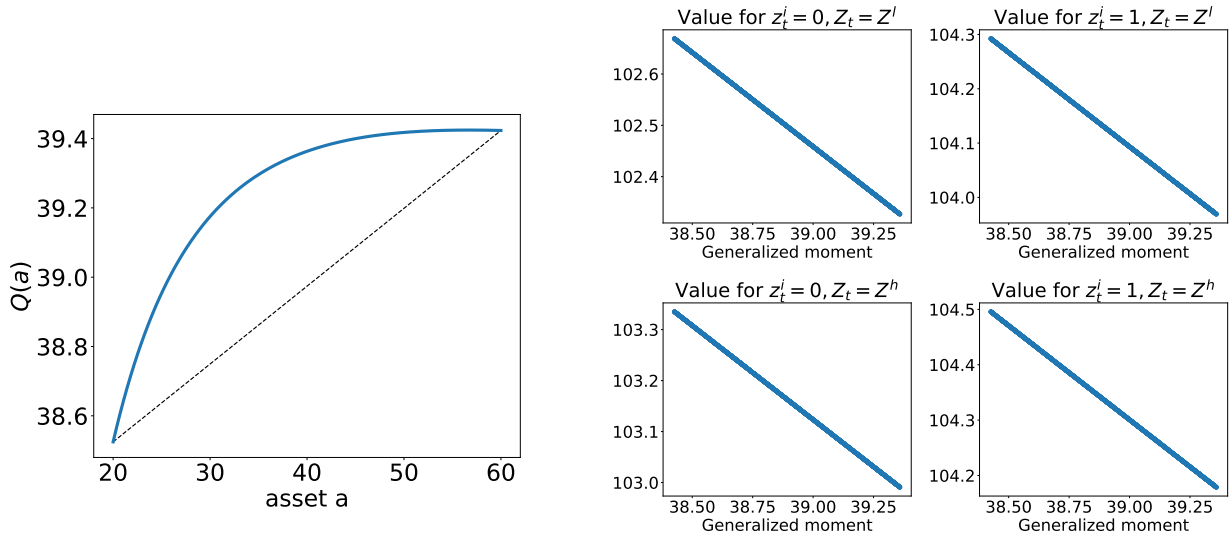
by 39.5%. The generalized moment plays an important role in improving solution accuracy because it provides a more concise representation of the household distribution and extracts more relevant information than the first moment. We discuss and interpret the single generalized moment we obtain in the Krusell-Smith problem in the next subsection. Additionally, we tested DeepHAM with two generalized moments and observed only a marginal and insignificant improvement in numerical accuracy compared to the single-moment solution. Our interpretation is that for relatively simple models like the KS model, one generalized moment is sufficient to capture the relevant distributional information, given the flexible function form and the algorithmically determined parameters.[12]

3.2.2 Generalized moments and redistributional effect

Among the three results in Table 2, DeepHAM with one generalized moment yields the most accurate solution. To better understand the improvement, we visualize the mapping from individual asset holdings a_t^i to the basis function $\mathcal{Q}(a_t^i)$, and the mapping from the generalized moment $\frac{1}{N}\sum_i \mathcal{Q}(a_t^i)$ to the value function in Figure 3.

We find that the basis function is concave in the individual asset, while the value function is linear with regard to the generalized moment. That is, households with different levels of wealth will have heterogeneous contributions to the generalized moment: giving an additional unit of assets to poor households increases the generalized moment more than giving the same assets to rich households. This implies that a purely redistributional policy would affect aggregate welfare and dynamics even in the simple setup of Krusell and Smith (1998). Consider an unanticipated one-time policy shock (MIT shock): if one unit of the asset is redistributed from the richest households to the poorest households, the welfare of "middle" households who are not in the redistribution program would decrease on impact since the generalized moment increases. Such an unanticipated policy shock will lead to higher ag-

[12]The KS method with the first moment is known to solve the Krusell-Smith model reasonably well. This is confirmed by the small Bellman error for the KS method in Table 2. Though DeepHAM can further improve the solution accuracy, the simulated economy based on the DeepHAM solution is highly consistent with the simulation based on the KS method. We present this comparison in Appendix D.1. This further confirms the accuracy of the DeepHAM solution for the Krusell-Smith model.



(a) Plot of $\mathcal{Q}_1(a)$

(b) Map $\frac{1}{N}\sum_i \mathcal{Q}_1(a_i)$ to the value function

Figure 3: Generalized moments for the Krusell-Smith problem. Left panel (solid blue line): a concave mapping from the individual asset to the basis function of the generalized moment. Right panel: mapping from the generalized moment to the value function, assuming households' individual assets fixed at the average level in the stationary equilibrium. Each figure in the right panel corresponds to one realization of idiosyncratic and aggregate shocks.

gregate savings in the future, since there are fewer people on the borrowing constraint. The increase in savings would lead to a higher future wage and a lower future asset return. Under the calibration of this model, the "middle" households who are not part of the redistribution program experience a larger marginal decline in capital income compared to the marginal increase in labor income following the unanticipated policy shock, making them worse off.[13]

This reasoning contrasts with the predictions of the KS method. According to the KS method, households' welfare only depends on the first moment and individual states. So the redistributional policy shock would have no instantaneous welfare impact on those "middle" households who are not in the redistribution program, since the first moment of individual

[13]Given the competitive factor market, the marginal change in capital income is

$$\left|\frac{\partial(r_t-\delta)a_t^i}{\partial K_t}\right| = \frac{\alpha Z_t(1-\alpha)}{L_t}\left(\frac{K_t}{L_t}\right)^{\alpha-2} a_t^i = \frac{\alpha Z_t(1-\alpha)}{L_t}\left(\frac{K_t}{L_t}\right)^{\alpha-1}\frac{1}{L_t},$$

for a middle income household with $a_t^i = K_t$. Similarly, the marginal change in labor income is:

$$\left|\frac{\partial[(1-\tau_t)\bar{l}z_t^i + b(1-z_t^i)]w_t}{\partial K_t}\right| = \frac{\alpha Z_t(1-\alpha)}{L_t}\left(\frac{K_t}{L_t}\right)^{\alpha-1}[(1-\tau_t)\bar{l}z_t^i + b(1-z_t^i)].$$

Due to the government's budget constraint, where total tax revenue equals total unemployment benefits, the expected marginal change in labor income for middle-income households is $\frac{\alpha Z_t(1-\alpha)}{L_t}\left(\frac{K_t}{L_t}\right)^{\alpha-1}$, which is smaller than the marginal change of her capital income as $L_t < 1$ in this model.

wealth distribution would not change.[14]

3.3 More Discussion of the Results

3.3.1 Infrequent Event and Scalability with More Simulated Agents

The results presented above are based on numerous simulation paths, each with 50 agents, a setup that works well in simple HA models such as Krusell and Smith (1998). However, in many HA models with aggregate shocks, infrequent or rare events, such as extreme income or wealth shocks, or endogenous defaults, play a crucial role in shaping individual decisions. Such infrequent events may call for a higher number of agents to ensure a faithful representation of the economic dynamics. In this subsection, we demonstrate that DeepHAM is capable of addressing these types of challenges, owing to the stability and scalability of our method, even when applied to a much larger number of agents.

Model setup The exercise is based on a simplified Krusell-Smith model. There are N *ex ante* identical households in this economy. Household i's labor supply is subject to idiosyncratic shocks $z_t^i \in \{e_0, e_1, e_2\}$, which are i.i.d. across agents and follow a Markov process. We model e_2 as the rare high income idiosyncratic state which only appears with 2% probability on average. We study the economy with agent number $N = 1000, 2000, 3000$ to ensure that the rare idiosyncratic state is realized in every period of each simulation sample, allowing us to properly account for its impact on aggregate prices in the agents' policy and value functions.

Household i accumulates asset a_t^i in the form of real capital. Household i's state $s_t^i = (a_t^i, z_t^i)$ follows

$$
\begin{aligned}
&a_{t+1}^i = (1 + r_t - \delta)a_t^i + w_t z_t^i - c_t^i, \\
&a_{t+1}^i \geq 0, \quad c_t^i \geq 0,
\end{aligned}
$$

where consumption c_t^i is the only control variable. The representative firm adopts production technology $Y_t = Z_t K_t^\alpha L_t^{1-\alpha}$, with aggregate productivity shocks $Z_t \in \{Z^l, Z^h\}$ following a Markov process. The factor prices are, $w_t = Z_t(1-\alpha)(K_t/L_t)^\alpha, \quad r_t = Z_t\alpha(K_t/L_t)^{\alpha-1}$, where aggregate capital $K_t = \frac{1}{N}\sum_{i=1}^N a_t^i$ and labor supply $L_t = (L^h \mathbb{1}_{Z_t=Z^h} + L^l \mathbb{1}_{Z_t=Z^l})$. Using the general descriptive variables in Section 2, the aggregate state variable $X_t = Z_t$. Appendix B.1.2 presents the details of the calibration.

[14]To note, here we consider the instantaneous welfare impact of an unanticipated redistributional policy shock, instead of the steady state effect of a usual redistributional policy program, which is studied in Davila et al. (2012) and in Section 5 of this paper.

Results Thanks to the scalability of our approach, we are able to solve the problem with 1,000 agents in under 4 hours on an A100 GPU on Google Colab, which is easily accessible to all researchers.

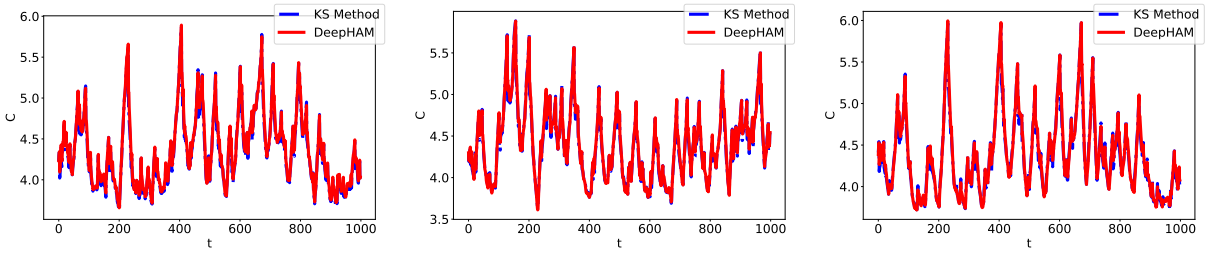


Figure 4: Aggregate consumption dynamics based on solutions obtained from the KS method and DeepHAM, under the same realization of idiosyncratic and aggregate shocks in each panel. Left to right: results with $N = 1000, 2000,$ and 3000.

To validate the DeepHAM solution accuracy with thousands of agents, we note that this new exercise remains a Krusell-Smith type of problem, where capital is the only asset. So it can be properly solved using the conventional Krusell-Smith method, providing a reliable benchmark for comparison. Importantly, the Krusell-Smith method does not suffer from the challenges of infrequent events, as it can easily incorporate both non-stochastic simulation or simulations with a large number (e.g., 10^4) of agents, as demonstrated by Maliar et al. (2010). In Figure 4, we confirm that our DeepHAM solutions with 1,000, 2,000, and 3,000 agents align almost perfectly with the results obtained using the conventional method, further validating the robustness and accuracy of our approach in handling infrequent events.

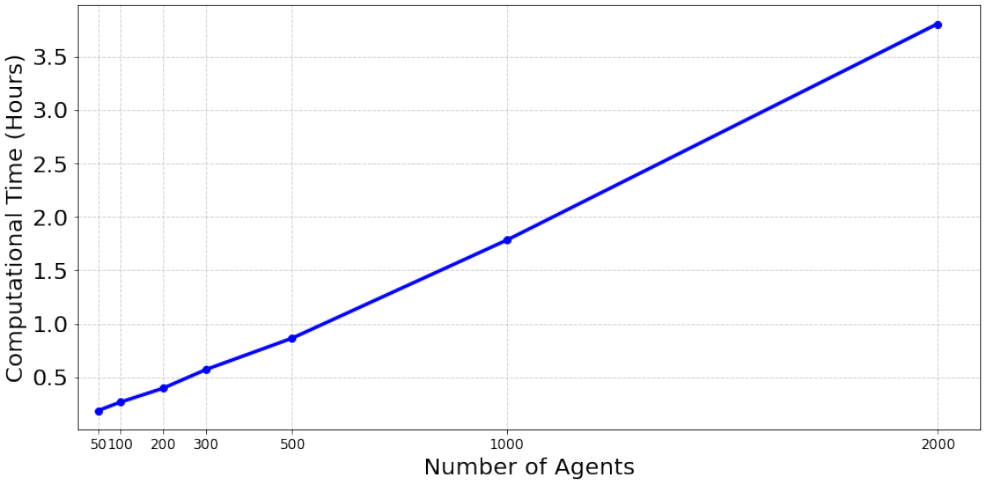


Figure 5: Scalability of computational time with the number of agents $N = 50, 100, 200, 300, 500, 1000, 2000$. We set the same values of hyperparameters at # of simulated paths = 128, unrolled $T = 20$ in Equation (11).

Scalability We use this model to conduct a systematic analysis of the relationship between computational cost and the number of agents. For this analysis, we need to fix the values of key hyperparameters (e.g., neural network architecture, training epochs, and batch sizes) to ensure comparability across experiments. Figure 5 plots computational time against the number of agents under a specific hyperparameter configuration. This plot clearly exhibits the linear scalability of the computational cost with respect to the number of agents, providing evidence of DeepHAM's scalability in handling larger number of agents.

3.3.2 Numerical Advantage of Generalized Moments

In this subsection, we compare the effectiveness of generalized moments (GM) as the distribution representation against the full state vector approach used in prior literature, by evaluating the solution accuracy for the Krusell-Smith problem in Section 3.1. In the GM approach, we solve for the policy function (and value function similarly) in the form of

$$\widetilde{c}_{\mathrm{NN}}(s_t^i, X_t, Q_t^C; \Theta^{CO}),$$

where $Q_t^C = \frac{1}{N}\sum_{i=1}^N \mathcal{Q}_{\mathrm{NN}}(s_t^i; \Theta^{CQ})$ are the generalized moments. In contrast, the full state vector approach directly solves for

$$c_{\mathrm{NN}}(s_t^i, X_t, \mathbb{S}_t; \Theta^C)$$

where the distribution $\mathbb{S}_t = \{s_t^1, s_t^2, \ldots, s_t^N\}$ is represented as a vector of all agents' individual states. In this comparison, we use the same objective function and evaluation method (via simulated paths), with the only difference being the distribution representation.

We aim to answer two key questions. First, if we keep all training process hyperparameters (e.g., number of layers, hidden neurons, and training steps) the same, how does the numerical error of the full state vector approach compare to that of the GM approach? Second, if we aim to make the full state vector approach achieve similar performance as the GM approach, how much larger must the neural network be, and how much longer would the training take?

To address these questions, we conducted numerous numerical experiments with the full state vector approach, including the four hyperparameter settings we report in Table 3. These four settings are:

Setting 1: the number of hidden layers, neurons per layer in the value and policy functions, and the number of training steps are kept the same as when using a single generalized moment;

Setting 2: the number of hidden layers and neurons per layer remain the same as in the first setting, but the number of training steps is doubled;

Setting 3: significantly larger networks are employed for the value and policy functions, but with the same number of training steps;

Setting 4: both the network size and the number of training steps are increased.

In summary, these different settings result in varying numbers of parameters in V_{NN} and c_{NN} and different total training steps for updating these parameters, allowing for a comprehensive comparison between the two state representation methods.

Table 3 lists the hyperparameters (number of neurons per layer, number of training steps in the SGD algorithm), the resulting number of parameters in the policy function neural network c_{NN}, and the computational time for the entire training process. The last column shows the average Bellman equation error under each setting, along with the standard deviation of this error metric.[15]

Method	# of neurons per layer (c_{NN})	# of SGD steps for c_{NN}	# of params in c_{NN}	Computation time (min)	Bellman error (std)
1 generalized moment	24	10000	938	60	0.0151 (0.0015)
Full state (Setting 1)	24	10000	1921	55	0.0861 (0.0411)
Full state (Setting 2)	24	20000	1921	110	0.0372 (0.0058)
Full state (Setting 3)	200	10000	51201	86	0.0243 (0.0031)
Full state (Setting 4)	200	20000	51201	172	0.0225 (0.0006)

Table 3: Numerical performance comparison for Krusell-Smith model: generalized moments vs. full state vector. Numbers in parentheses indicate the standard deviation of the Bellman error, based on 5 independent solutions. Note: due to the large memory requirements of the full state vector representation, the experiments in this table were conducted on a server equipped with an A100 GPU with 80GB memory. This differs from the other experiments in this paper, which were primarily conducted on Google Colab, where GPUs generally provide up to 40GB of memory as of December 2024.

Comparing the generalized moment approach to Setting 1, we find that, with all the

[15]Besides the hyperparameters reported in Table 3, all other parameters are kept the same as reported in Appendix E. The computational times reported in this paper are directly taken from our numerical experiments. It is important to note that actual runtime can vary depending on many factors, including hardware configurations and the current status of the hardware.

other hyperparameters kept the same, the solution accuracy with the generalized moment is much better than that of the full state vector approach. This improvement is primarily due to the more effective representation of the state variable by the generalized moment. Even with more training steps in Setting 2, the accuracy of the full state vector representation remains constrained by its inability to efficiently extract the most relevant information from the state distribution.

Then we go beyond Settings 1 and 2 to significantly increase the size of neural networks and the number of training steps to attempt matching the accuracy of the generalized moment approach. After conducting numerous experiments, we present results from Settings 3 and 4, which represent the best performance we achieved for the full state vector approach in the KS problem with $N = 50$. As expected, increasing the network capacity for the value and policy functions led to improved solution accuracy. However, despite a more than 50-fold increase in the number of neural network parameters and a threefold increase in computational time, the accuracy of the full state representation still falls short of that achieved with a generalized moment.

In Appendix D.2, we extend our experiments to $N = 80$ to assess the scalability of the two approaches as the state space grows. The key takeaway is that, for the generalized moment approach, the computational cost increases only marginally when scaling from $N = 50$ to $N = 80$. In contrast, the full state vector representation requires significantly more computational resources, both in terms of additional parameters and extended training steps, to reach a comparable level of accuracy.

4 DeepHAM for More Complex HA Models

In this section, we use DeepHAM to solve a HA model with a financial sector and aggregate shocks, as proposed in Fernández-Villaverde, Hurtado, and Nuno (2019). Compared to the Krusell-Smith model with the two-state aggregate shocks in Section 3, here the aggregate shocks take values in a continuous range, which makes the problem more costly to solve.

4.1 Fernández-Villaverde et al. (2019): Model Setup

The setup is the discrete time version of Fernández-Villaverde et al. (2019). We use the subscript t and $t + \Delta t$ to highlight that the model comes from the discretization of a continuous time model, but should be interpreted as representing the dynamics between t and $t + 1$ in the general setup in Section 2. In this economy, there are N households who save in risk-free bonds and consume. Their labor supply is exogenous and exposed to idiosyncratic shocks.

There is a representative financial expert who issues risk-free bonds to households and invests in productive capital. A representative firm produces with capital from the financial expert and with labor supplied by the households. The growth rate of productive capital is exposed to aggregate shocks.

Household's problem. For household i, her state is $s_t^i = (a_t^i, z_t^i) \in \mathbb{R}^2$, with beginning-of-period risk-free asset a_t^i, and the idiosyncratic shocks on labor supply $z_t^i \in \{z_1, z_2\}$ with $0 < z_1 < z_2$. The process z_t^i follows a first-order Markov process with ergodic mean 1 such that the aggregate labor supply $L_t = 1$. Household i has constant relative risk aversion (CRRA) utility from consumption c_t^i with parameter $\gamma > 0$ and discount factor $e^{-\rho \Delta t}$.

The household budget and borrowing constraints determines the state dynamics (1) of household i:

$$\begin{aligned} a_{t+\Delta t}^i &= a_t^i + (w_t z_t^i + r_t a_t^i - c_t^i)\Delta t, \\ a_{t+\Delta t}^i &\geq 0, \quad c_t^i \geq 0, \end{aligned} \tag{18}$$

where the aggregate prices are characterized below. Aggregate risk-free asset demand $B_t = \frac{1}{N}\sum_{i=1}^N a_t^i$.

Representative firm's problem. The firm produces with Cobb-Douglas technology $Y_t = K_t^\alpha L_t^{1-\alpha}$. It hires labor L_t from households at wage w_t, and rents capital K_t from the financial expert at rental rate rc_t, both in the competitive factor market:

$$w_t = (1-\alpha)(K_t/L_t)^\alpha, \quad rc_t = \alpha(K_t/L_t)^{\alpha-1}. \tag{19}$$

Financial expert's problem. The representative financial expert issues a risk-free bond B_t at rate r_t to households, and rents capital K_t at rate rc_t to the representative firm. Her net worth $W_t = K_t - B_t$. For the financial expert, the instantaneous return rate on capital is exposed to aggregate shocks Z_t:

$$\frac{K_{t+\Delta t} - K_t}{K_t} = (rc_t - \delta)\Delta t + \sigma Z_t \sqrt{\Delta t},$$

where δ is the depreciation rate of capital, σ is the volatility of aggregate shocks, and Z_t follows an i.i.d. standard normal distribution.[16]

[16] In the continuous time model, the aggregate shock is a white noise process with volatility σ. There would be a slight numerical difference to the discretized model, but the difference is tiny as we choose a tiny Δt.

The financial expert has log utility with discount rate $\widehat{\rho} < \rho$ over consumption $\widehat{C}_t$, so she consumes a constant share of her net worth: $\widehat{C}_t = \widehat{\rho} W_t$, and chooses a leverage ratio proportional to excess return of risky capital $\frac{K_t}{W_t} = \frac{1}{\sigma^2}(rc_t - \delta - r_t)$. So the risk-free return is

$$r_t = \alpha(K_t/L_t)^{\alpha-1} - \delta - \sigma^2 \frac{K_t}{W_t}. \tag{20}$$

The budget constraint of the financial expert $W_{t+\Delta t} = W_t + (rc_t - \delta)K_t\Delta t + \sigma K_t Z_t\sqrt{\Delta t} - B_t r_t \Delta t - \widehat{C}_t \Delta t$ implies the following dynamics of net worth W_t:

$$W_{t+\Delta t} = W_t + \left(\alpha K_t^{\alpha-1} - \delta - \widehat{\rho} - \sigma^2\left(1 - \frac{K_t}{W_t}\right)\frac{K_t}{W_t}\right) W_t \Delta t + \sigma K_t Z_t \sqrt{\Delta t}. \tag{21}$$

Using the general descriptive variables in Section 2, the aggregate state $X_t = (W_t, Z_t)$. Since $K_t = B_t + W_t$, the evolution of W_t only depends on $W_t, B_t = \frac{1}{N}\sum_{i=1}^N a_t^i$, and Z_t. The aggregate state dynamics (3) are specified as (21). Equations (18)(19)(20), together with the stochastic process of z_t^i, complete the specification of (1) and (2). The calibration of the model follows Fernández-Villaverde et al. (2019) and is presented in Appendix B.2.

4.2 Solution Accuracy and Efficiency

We use DeepHAM to obtain the global solution to the problem described above with $N = 50$. We compare the Bellman equation errors (see the definition in Appendix C) of DeepHAM to the generalized KS method with the nonlinear perceived law of motion implemented in Fernández-Villaverde et al. (2019) in Table 4. For DeepHAM, we present accuracy measures for the cases where we include in the state variable (1) only the first moment or (2) one generalized moment of household asset distribution.

Method and Moment Choice	Bellman error	Std of error
KS Method (Fernández-Villaverde et al., 2019)	0.00417	0.00011
DeepHAM with 1st moment	0.00415	0.00014
DeepHAM with 1 generalized moment	0.0036	0.00061

Table 4: Comparison of solution accuracy on a HA model with a financial sector and aggregate shocks. The KS method refers to the solution method implemented by (Fernández-Villaverde et al., 2019).

As we see in Table 4, the solutions obtained using DeepHAM are highly accurate. Compared to the generalized KS method with the nonlinear law of motion on the first moment implemented by Fernández-Villaverde et al. (2019), DeepHAM with the first moment can

obtain global solutions with the same level of accuracy. Introducing the generalized moments can further reduce the Bellman equation error by 13.3% on average, as they provide a more concise representation of the household distribution and extract more relevant information than the first moment. We present the standard deviation of the Bellman errors from multiple runs of the numerical algorithm in the last column of Table 4.[17]

Solving this HA model, with a financial sector and aggregate shocks which take values in a continuous range, takes DeepHAM 33% longer than solving the simple Krusell-Smith model in Section 3. This result demonstrates the efficiency of DeepHAM in studying complex HA models with aggregate shocks: unlike the grid-based method, the computational cost of DeepHAM does not increase quickly when the number of state variables or grid points increases.[18]

5 DeepHAM for Constrained Efficiency Problem in HA Models

In this section, we solve the constrained efficiency problem in HA models using DeepHAM. In contrast to the competitive equilibrium, the constrained optimum of an HA model, defined as the allocation decided by a benevolent social planner who maximizes social welfare, is much harder to solve. Existing literature only handles constrained optima of HA models without aggregate shocks (Davila, Hong, Krusell, and Ríos-Rull, 2012; Nuño and Moll, 2018), and at a much higher computational cost than for the competitive equilibrium of the same model.

In contrast, as presented in Section 2.4, DeepHAM can solve the constrained efficiency problem as easily as it can solve the competitive equilibrium. We illustrate this advantage by using DeepHAM to solve the constrained efficiency problem in an Aiyagari model as in Davila et al. (2012), and in a HA model with aggregate shocks.

5.1 Model Setup

Baseline setup without aggregate shocks. The baseline setup is an Aiyagari model that follows the "high wealth dispersion" calibration in Davila et al. (2012) to match empir-

[17]Note that Fernández-Villaverde et al. (2019) builds and solves the model in continuous time. To compare it with DeepHAM, which solves the discrete time version of the model, we evaluate their solution also on the discrete time Bellman equation error defined in Appendix C. Since we chose a small Δt, the continuous time solution should give a meaningful approximate Bellman error in discrete time. We find that DeepHAM obtains an accurate solution which is comparable to Fernández-Villaverde et al. (2019).

[18]In Appendix F, we report the computational time required by DeepHAM to solve each model, including comparisons across different moment choices. The appendix also includes the hardware configurations used for each experiment.

ical US wealth inequality. There are N *ex ante* homogeneous households in this economy. Household i's labor supply is subject to idiosyncratic shocks $z_t^i \in \{e_0, e_1, e_2\}$, which are i.i.d. across agents and follow a Markov process. Household i accumulates asset a_t^i in the form of real capital. Household i's state $s_t^i = (a_t^i, z_t^i)$ follows

$$\begin{aligned} a_{t+1}^i &= (1 + r_t - \delta)a_t^i + w_t z_t^i - c_t^i, \\ a_{t+1}^i &\geq 0, \quad c_t^i \geq 0, \end{aligned} \tag{22}$$

where consumption c_t^i is the only control variable. The representative firm produces with a Cobb-Douglas technology $Y_t = K_t^\alpha L_t^{1-\alpha}$ and rents capital and hires labor in a competitive factor market. So the wage w_t and capital rental rate r_t are:

$$w_t = (1-\alpha)(K_t/L_t)^\alpha, \quad r_t = \alpha(K_t/L_t)^{\alpha-1}, \tag{23}$$

where aggregate capital $K_t = \frac{1}{N}\sum_{i=1}^N a_t^i$ and labor supply $L_t = \bar{L}$ is constant. This completes the specification of (1) and (2). Since there is no aggregate shock in the baseline setup, there is no aggregate state variable X_t nor are there dynamics as in (3).

The benevolent social planner seeks to find a policy rule $\mathcal{C}$ determining c_t^i for all the households $i = 1, ..., N$ and $t = 0, 1, ..., \infty$ to maximize the utilitarian objective,

$$\max_{\mathcal{C}} \ \frac{1}{N}\mathbb{E}_{\mu(\mathcal{C}),\mathcal{E}} \sum_{i=1}^N \sum_{t=0}^\infty \beta^t u(c_t^i),$$

subject to the constraints in (22).[19]

Setup with aggregate shocks. We also solve the constrained efficiency problem of a HA model with aggregate shocks. On top of the baseline model above, we introduce aggregate productivity shocks $Z_t \in \{Z^l, Z^h\}$, that follow a Markov process, on the production technology of the representative firm $Y_t = Z_t K_t^\alpha L_t^{1-\alpha}$, such that the factor prices are,

$$w_t = Z_t(1-\alpha)(K_t/L_t)^\alpha, \quad r_t = Z_t\alpha(K_t/L_t)^{\alpha-1}, \tag{24}$$

where aggregate capital $K_t = \frac{1}{N}\sum_{i=1}^N a_t^i$ and labor supply $L_t = (L^h \mathbb{1}_{Z_t=Z^h} + L^l \mathbb{1}_{Z_t=Z^l})$. Using the general descriptive variables in Section 2, the aggregate state variable $X_t = Z_t$.

[19] Given the form of the utilitarian objective, we have an alternative approach to approximate the expected social welfare besides the one presented in Section 2.4. We can learn the individual value function defined in (8) with the approximation form (7) and use that to approximate the expected total social welfare by taking the average of the individual value function. We have used the latter approach in this work.

We also introduce countercyclical idiosyncratic risk to the model, so that the probability that households enter the low income state $z_t^i = e_0$ becomes larger in the bad aggregate state, and smaller in the good aggregate state. Our setup follows the "integration principle" proposed by Krusell et al. (2009), so that when aggregate shocks are eliminated, the model will exactly reduce to the baseline setup. Equations (22) and (23) (or (24)), together with the stochastic process of z_t^i, complete the specifications of (1) and (2). The calibration of both models are presented in Appendix B.3.

5.2 Results

We solve the constrained planner's problems with $N = 50$ in both the baseline model without aggregate shocks, and in the model with aggregate shocks and countercyclical idiosyncratic shocks. The equilibrium statistics of these problems are presented in Table 5 and 6. In comparison, we also present equilibrium statistics under the competitive equilibrium of the same models.[20]

	No aggregate shock		Aggregate shock	
	Market	Constrained Opt.	Market	Constrained Opt.
Aggregate capital	30.635	119.741	34.296	95.811
Output	10.294	16.816	12.159	17.592
Capital-output ratio	2.976	7.120	2.821	5.446
Real interest rate	4.097%	-2.944%	4.678%	-1.433%
Wealth coeff. of variation	2.621	2.483	2.574	2.924
Wealth Gini index	0.864	0.862	0.812	0.878
Consumption coeff. of variation	1.548	0.710	1.699	0.736
Consumption Gini index	0.615	0.386	0.578	0.388

Table 5: Equilibrium statistics in the market outcome (competitive equilibrium) and constrained optimum for models without or with aggregate shocks.

The main findings are as follows. First, in both models (with or without aggregate shocks) in Table 5, the constrained optimum requires a much higher level of aggregate capital than the competitive equilibrium. In the absence of aggregate shocks, the planner chooses a capital level 3.91 (= 119.741/30.635) times that of the laissez-faire equilibrium, which is consistent with the finding of Davila et al. (2012). This is because the planner with the utilitarian objective aims to redistribute from rich households to poor households in order to improve social welfare. Since poor households have a higher labor income share, the

[20] When solving the constrained efficiency problem, we usually find two local maxima of the problem. Since we are solving the constrained planner's problem, we only take the local optimum with higher expected total welfare. We leave the study of the "second best" constrained optimum for future research.

planner would raise the aggregate capital level, so that the wage rate increases and capital return decreases according to equations (23) and (24). By raising the aggregate capital level, such a redistribution leaves poor households better off. Meanwhile, in both models, the constrained efficiency problem features a similar level of wealth inequality and a lower level of consumption inequality relative to the market outcome, measured by the Gini indices and coefficients of variation of these two variables.

Second, compared to the constrained optimum in the absence of aggregate shocks, the model with aggregate shocks features a lower level of aggregate capital stock. With aggregate shocks and countercyclical unemployment risks, the planner chooses a capital level 2.79 (= 95.811/34.296) times that of the laissez-faire equilibrium, which is lower than the 3.91 times for the model without aggregate shocks. This is because with aggregate shocks, households, especially poor households, have a stronger precautionary saving motive, and their labor income share is thus lower than in the model without aggregate shocks. So the planner would still raise aggregate capital to redistribute through price changes, but not as much as in the economy without aggregate shocks. We provide further validation of this explanation in Section 5.3.

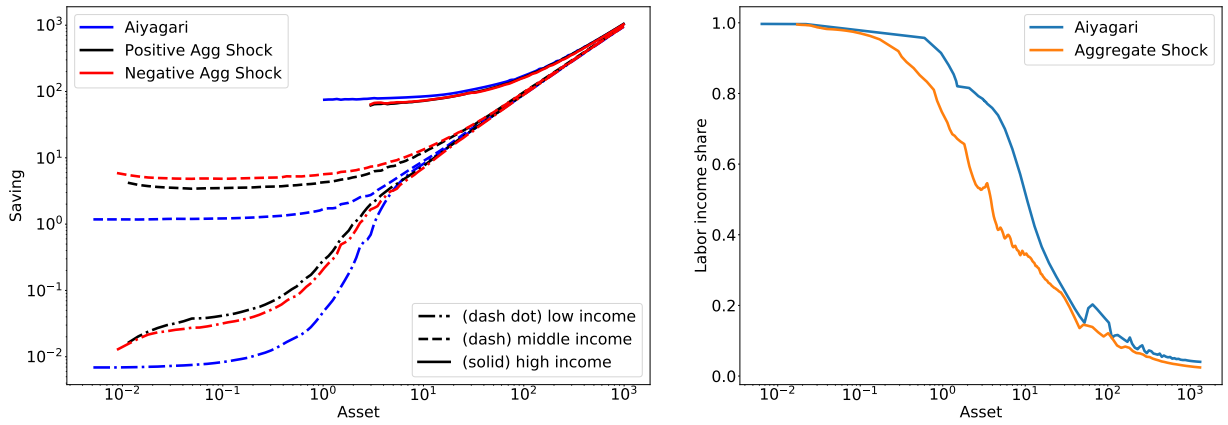
	Positive aggregate shock		Negative aggregate shock	
	Market	Constrained Opt.	Market	Constrained Opt.
Aggregate capital	36.316	99.793	32.260	91.826
Output	13.925	20.038	10.393	15.146
Capital-output ratio	2.608	4.980	3.104	6.063
Real interest rate	5.147%	-1.116%	4.208%	-1.750%
Wealth coeff. of variation	2.533	2.894	2.614	2.953
Wealth Gini index	0.815	0.877	0.805	0.877
Consumption coeff. of variation	1.693	0.756	1.697	0.713
Consumption Gini index	0.599	0.407	0.542	0.345

Table 6: Equilibrium statistics in the market outcome (competitive equilibrium) and constrained optimum for the HA model with aggregate shocks, conditional on the realization of aggregate shock.

Third, according to Table 6, which presents equilibrium statistics for the HA model with aggregate shocks conditional on the realization of aggregate shocks, the planner intends for households to increase their savings in a higher ratio (2.85 = 91.826/32.260) in the bad aggregate state, compared to (2.75 = 99.793/36.316) in the good aggregate state.

5.3 Impact of Aggregate Shocks on Constrained Optimum

To further understand the impact of aggregate shocks on the constrained efficiency problem, we compare households' saving policy and labor share across the asset distribution in the constrained optimum with aggregate shocks and without aggregate shocks ("Aiyagari economy") in Figure 6. Figure 6a shows that in the presence of aggregate shocks, households, especially wealth-poor households, save more than they do in the Aiyagari economy due to precautionary motives. As a result, households, especially wealth-poor households, have a lower labor income share compared to the Aiyagari economy, which is shown in Figure 6b. As a result, in order to redistribute towards wealth-poor households, the constrained planner does not need to raise the aggregate capital level as much in the presence of aggregate shocks as in the economy without aggregate shocks.



(a) Saving policy

(b) Labor share

Figure 6: Household's saving policy and labor share along asset distribution in the constrained optimum. In the model with aggregate shocks, households, especially wealth-poor households, have more precautionary saving and a lower labor income share, compared to the economy without aggregate shocks.

5.4 Computational Efficiency for Constrained Efficiency Problem

In this section, we highlight the computational efficiency of DeepHAM in solving the constrained planner's problem. In Table 7, we compare the computational cost of DeepHAM versus the conventional method following Davila et al. (2012), for both the baseline Aiyagari model and the model with aggregate shocks. Each method is first executed on the most suitable hardware: the conventional method, coded in Fortran, is optimized for CPU execution, while DeepHAM, a deep learning-based approach, leverages the computational advantages of GPUs, which are designed to accelerate neural network training.

	Hardware	Aiyagari model	With aggregate shocks
Classical method	CPU	> 15 hours	not solved in the literature
DeepHAM	GPU	27 minutes	40 minutes
DeepHAM	CPU	≈ 2 hours 16 minutes	≈ 2 hours 20 minutes

Table 7: Comparison of the computational speed for the constrained efficiency problem. The conventional method (Davila et al., 2012) is implemented on the high-RAM CPU on Google Colab. DeepHAM is implemented on the A100 GPU and the high-RAM CPU, both on Google Colab.

The computational times in the first two rows of Table 7 reflect the results when each method is executed on its respective optimal hardware. To address potential concerns about hardware inconsistencies, we also conducted a direct comparison where both methods were run exclusively on CPUs. When implemented on suitable machines, DeepHAM is about 30 times faster than the conventional method to solve the constrained efficiency problem without aggregate shocks. While DeepHAM takes longer on the CPU (approximately 2.25 hours, compared to 30 minutes on the GPU), it remains far more efficient than the conventional method, which requires over 15 hours on the same CPU.[21]

To our knowledge, a global solution to the constrained efficiency problem in HA models with aggregate shocks has not been presented in the literature due to the computational challenges. In contrast, DeepHAM can handle this class of problems quite efficiently, taking 40 minutes on a GPU that is easily accessible to all researchers.

6 Conclusion and Future Research

In this paper, we present DeepHAM, an efficient, reliable, and interpretable deep learning-based method for globally solving HA models with aggregate shocks. DeepHAM achieves highly accurate results, and can be applied to complex HA models without suffering from the curse of dimensionality. The algorithm automatically generates a flexible and interpretable representation of the agent distribution through generalized moments. The generalized moments extract key information of the distribution that are most relevant to individual decision rules and thus, through aggregation, to welfare and the evolution of the aggregate economy. They furthermore help us better understand whether and how heterogeneity matters in macroeconomics. Moreover, DeepHAM can solve the constrained efficiency problem

[21]It is worth noting that the CPU computational times on Google Colab can be somewhat stochastic due to unpredictable hardware allocation, whereas GPU times are more stable. Despite this variability, DeepHAM's CPU runtime of 2.25 hours remains substantially faster than at least 15 hours (and even longer) required by the conventional method. This confirms that DeepHAM is significantly more efficient, even when run on less optimized hardware.

as fast as solving the competitive equilibrium, a significant advantage over existing methods. The results demonstrate that DeepHAM is a powerful tool for studying global patterns of complex HA models with aggregate shocks, opening up many exciting possibilities for future research.

In this paper, we assume that the dependence of aggregate prices and quantities on individual states could be written in explicit functional forms. This assumption excludes HA models with aggregate variables that are determined recursively as a function of expected future aggregate variables: the inflation rate in a New Keynesian Phillips curve (NKPC), for example, or indirect utility under Epstein-Zin preferences. Handling forward-looking equilibrium conditions in global solution methods is crucial in solving HANK models with aggregate shocks and requires an additional price function to be incorporated into the algorithm. We leave this for future research. Another important direction for future research is to integrate deep learning-based solution methods with model calibration and estimation, as demonstrated by Kase, Melosi, and Rottner (2022); Friedl, Kübler, Scheidegger, and Usui (2023). This would further exploit the computational efficiency of deep learning, making it even more valuable in quantitative economic models.

Appendix

A Neural Networks: a Class of Function Approximator

In this paper, we consider deep, fully connected feedforward neural networks. A network $y = u(x;\Theta)$ with L ($L \geq 1$) hidden layers defines a mapping $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$, in which $x \in \mathbb{R}^{d_1}$ is the input variable, $y \in \mathbb{R}^{d_2}$ is the output variable, and $\Theta = (W_1, b_1, \ldots, W_{L+1}, b_{L+1})$ is the collection of network parameters. The network's mapping is defined by a series of compositions of linear transformations and nonlinear activation functions:

$$
\begin{aligned}
y &= \sigma_{L+1} \circ (W_{L+1} z_L + b_{L+1}), \\
z_L &= \sigma_L \circ (W_L z_{L-1} + b_L), \\
&\vdots \\
z_2 &= \sigma_2 \circ (W_2 z_1 + b_2), \\
z_1 &= \sigma_1 \circ (W_1 x + b_1).
\end{aligned}
$$

Here, $W_l \in \mathbb{R}^{m_l \times m_{l-1}}$ is called the weight matrix, and $b_l \in \mathbb{R}^{m_l}$ is called the bias vector, with $l = 1, \ldots, L+1$. We have $m_0 = d_1, m_{L+1} = d_2$, and $m_1, \ldots, m_l$ are set as network hyperparameters. $\sigma_l : \mathbb{R} \to \mathbb{R}$ is a scalar function called an activation function, and $\circ$ denotes element-wise evaluation. The typical choices of σ_l include rectified linear units (ReLU) $\sigma(x) = \max\{(0, x)\}$ and the sigmoid function $1/(1+e^{-x})$, among others. Typically, σ_l are the same for all $l = 1, \ldots, L$ and σ_{L+1} is chosen as the identity function to ensure the output is unrestricted. In our policy function neural network $c_{\text{NN}}(\cdot)$ in equation (12), we choose σ_{L+1} as the sigmoid function such that the inequality constraints on the decision variable can be satisfied.

Hornik et al. (1989); Cybenko (1989) prove that neural networks with one hidden layer neural networks are universal approximators, i.e., they can approximate arbitrary well any unknown Borel measurable function over a compact domain. In recent years, it has been extensively demonstrated empirically and theoretically that deep neural networks with multiple hidden layers have better approximation and optimization efficiency than shallow neural networks with one hidden layer (Goodfellow et al., 2016). In various fields such as reinforcement learning (Silver et al., 2016), numerical PDEs (E et al., 2021a), and scientific computing (E et al., 2021b), deep neural networks have demonstrated astonishing capability in handling high-dimensional state variables in which traditional numerical tools suffer a lot from the curse of dimensionality.

In this paper we are interested in approximating permutation invariant functions through generalized moments. Mathematically, a function $f : (\mathbb{R}^d)^N \to \mathbb{R}$ is called *permutation invariant* if

$$f(\mathbf{x}_{\sigma(1)}, \ldots, \mathbf{x}_{\sigma(N)}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_N),$$

for any permutation $\sigma \in S(N)$ (the symmetric group of degree N), and elements $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$. It has been proved that any permutational invariant functions in multi-dimensional space can be approximated by a two-step decomposition (Zaheer et al., 2017; Han et al., 2022), which justifies the usage of generalized moments. Here we recall the following theorem from Han et al. (2022) for better readability.

Theorem 1 (Theorem 1.1, Han et al. (2022))**.** *Let $f : \Omega^N \to \mathbb{R}$ be a continuously differentiable, permutation invariant function, where Ω is a compact subset of $\mathbb{R}^d$. Let $0 < \epsilon < \|\nabla f\|_2 \sqrt{Nd} N^{-\frac{1}{d}}$, here $\|\nabla f\|_2 = \max_X \|\nabla f(X)\|_2$. Then there exist $g : \mathbb{R}^d \to \mathbb{R}^M$, $\phi : \mathbb{R}^M \to \mathbb{R}$, such that for any $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \Omega^N$,*

$$\left| f(X) - \phi\left(\sum_{j=1}^{N} g(\mathbf{x}_j) \right) \right| \leq \epsilon,$$

where M, the number of feature variables, satisfies the bound

$$M \leq 2^N (\|\nabla f\|_2^2 Nd)^{Nd/2} / (\epsilon^{Nd} N!).$$

B Details of the Model Setup

B.1 Models in Section 3

B.1.1 Krusell and Smith (1998) Model

Calibration. The parameters follow Den Haan (2010) with each period representing one quarter. The capital share $\alpha = 0.36$, depreciation rate of capital $\delta = 0.025$, household discount factor $\beta = 0.99$, labor supply $\bar{l} = 1/0.9$, unemployment benefit rate $b = 0.15$, and households have log utility. Aggregate productivity $Z_t \in \{1.01, 0.99\}$. The joint transition matrix for idiosyncratic and aggregate shock is $\Pi_Z = \begin{bmatrix} 0.525 & 0.35 & 0.03125 & 0.09375 \\ 0.038889 & 0.836111 & 0.002083 & 0.122917 \\ 0.09375 & 0.03125 & 0.291667 & 0.583333 \\ 0.009115 & 0.115885 & 0.024306 & 0.850694 \end{bmatrix}$, where the four rows and columns correspond to $(Z_t, z_t) = (0.99, 0), (0.99, 1), (1.01, 0), (1.01, 1)$ respectively.

B.1.2 Model with Rare Event

Calibration to allow for rare events. The capital share $\alpha = 0.36$, depreciation rate of capital $\delta = 0.08$, discount factor $\beta = 0.887$, and the risk aversion of the households $\gamma = 2$.

The aggregate shock $Z_t \in \{Z^l, Z^h\} = \{0.95, 1.05\}$ with transition matrix $\Pi_Z = \begin{bmatrix} 0.875 & 0.125 \\ 0.125 & 0.875 \end{bmatrix}$.

Idiosyncratic shock $z_t^i \in \{e_0 = 1, e_1 = 5.29, e_2 = 46.55\}$. When $Z_t > 1$, the transition matrix across labor endowment

$$\Pi_{e,t} = \begin{bmatrix} 0.98 & 0.02 & 0 \\ 0.01 & 0.98 & 0.01 \\ 0 & 0.2 & 0.8 \end{bmatrix}$$

with invariant distribution over $\{e_0, e_1, e_2\}$ being $\Big[0.32258065, 0.64516129, 0.03225806\Big]$.

When $Z_t < 1$,

$$\Pi_{e,t} = \begin{bmatrix} 0.6512 & 0.3488 & 0 \\ 0.98 & 0.01 & 0.01 \\ 0 & 0.2 & 0.8 \end{bmatrix}$$

with invariant distribution $\Big[0.72795341, 0.25909199, 0.0129546\Big]$.

The aggregate labor supply in good and bad aggregate states are

$$\{L^h, L^l\} = \{5.237096774193559, 2.701586641312124\},$$

respectively. We let $\bar{L} = \frac{L^h + L^l}{2} = 3.9693417077528417$.

B.2 Fernández-Villaverde et al. (2019) Model

Calibration. We follow Fernández-Villaverde et al. (2019) for parameters. The capital share $\alpha = 0.35$, depreciation rate of capital $\delta = 0.1$, household discount rate $\rho = 0.05$, expert discount rate $\widehat{\rho} = 0.04971$, volatility of aggregate shocks $\sigma = 0.014$, and the risk aversion of the households $\gamma = 2$. To solve the problem in discrete time, we choose $\Delta t = 0.2$, which should be a small number so that the solution is comparable to the continuous time solution. Households' discount factor $\beta = e^{-\rho \Delta t}$. The transition matrix of idiosyncratic shocks is $\Pi_e = \begin{bmatrix} 1 - \lambda_1 \Delta t & \lambda_1 \Delta t \\ \lambda_2 \Delta t & 1 - \lambda_2 \Delta t \end{bmatrix}$ where $\lambda_1 = 0.986$, $\lambda_1 = 0.052$.

B.3 Davila et al. (2012) Model

Calibration. The parameter setting in the baseline model follows Davila et al. (2012). The capital share $\alpha = 0.36$, depreciation rate of capital $\delta = 0.08$, discount factor $\beta = 0.887$, and the risk aversion of the households $\gamma = 2$. Labor endowment $z_t^i \in \{e_0 = 1, e_1 = 5.29, e_2 = 46.55\}$, and $\Pi_e = \begin{bmatrix} 0.992 & 0.008 & 0 \\ 0.009 & 0.980 & 0.011 \\ 0 & 0.083 & 0.917 \end{bmatrix}$ with stationary distribution $\{0.498, 0.443, 0.059\}$. The aggregate labor supply $L_t = \bar{L} = 0.498 e_0 + 0.443 e_1 + 0.059 e_3 = 5.574$.

In the model with aggregate shocks, $Z_t \in \{Z^l, Z^h\} = \{0.95, 1.05\}$ with transition matrix $\Pi_Z = \begin{bmatrix} 0.875 & 0.125 \\ 0.125 & 0.875 \end{bmatrix}$. The idiosyncratic shocks are countercyclical, and the transition matrix across labor endowment $\Pi_{e,t} = \begin{bmatrix} 0.98 & 0.02 & 0 \\ 0.009 & 0.980 & 0.011 \\ 0 & 0.083 & 0.917 \end{bmatrix}$ when $Z_t > 1$, $\Pi_{e,t} = \begin{bmatrix} 0.6512 & 0.3488 & 0 \\ 0.978 & 0.011 & 0.011 \\ 0 & 0.083 & 0.917 \end{bmatrix}$ when $Z_t < 1$. The aggregate labor supply in good and bad aggregate states are $\{L^h, L^l\} = \{7.525, 3.623\}$, respectively.

C Accuracy Measures

The main accuracy measure we adopt in this paper is the Bellman equation error defined in this section. We choose it over the Euler equation error since it provides a better measure over the whole state space, especially the region close to the inequality constraints, without the need to introduce the Lagrangian multiplier.

In the general HA model in Section 2, agent i's optimization problem can be characterized recursively:

$$V(s_t^i, X_t, \mathbb{S}_t) = \max_{c_t^i} \left[u(c_t^i) + \beta \mathbb{E} V(s_{t+1}^i, X_{t+1}, \mathbb{S}_{t+1})\right].$$

Given the solved value function $V(\cdot)$, we can evaluate the Bellman equation error for each state $(s_t^i, X_t, \mathbb{S}_t)$ in the state space as:

$$\text{err}_{\text{B}}(s_t^i, X_t, \mathbb{S}_t) = \left| V(s_t^i, X_t, \mathbb{S}_t) - \max_{c_t^i} \left[u(c_t^i) + \beta \mathbb{E} V(s_{t+1}^i, X_{t+1}, \mathbb{S}_{t+1})\right] \right|$$

where the expectation operator is approximated by Monte Carlo sampling of aggregate and idiosyncratic shocks, and the consumption choice c_t^i is solved again given the solved value function $V(\cdot)$, rather than directly taken from the optimal policy we solve.

We then average with respect to the stationary distribution over $(X_t, \mathbb{S}_t)$ to calculate the Bellman equation error for the solution we obtain:

$$\text{err}_{\text{B}} = \mathbb{E}_{\mu(\mathcal{C}^*)} \left| V(s_t^i, X_t, \mathbb{S}_t) - \max_{c_t^i} \left[u(c_t^i) + \beta \mathbb{E} V(s_{t+1}^i, X_{t+1}, \mathbb{S}_{t+1})\right] \right|.$$

D Additional Results

D.1 Solution Comparison for Krusell-Smith Model

In this section, we compare the simulated economy based on the DeepHAM solution with the simulation based on the KS method. Under the same realization of idiosyncratic and aggregate shocks, the two economies simulated with 1000 agents based on the two solution methods are presented in Figure 7.[22]

The KS method with the first moment can solve the Krusell-Smith model reasonably well, as has been widely validated in the literature. Although DeepHAM can further improve the solution accuracy, as we present in Section 3.2.1, the simulated economy based on the DeepHAM solution is highly consistent with the simulation based on the KS method, as

[22]The DeepHAM solution comes from a finite agent model with $N = 50$, while we simulate the two economies with 1000 agents to show the two solutions are consistent with each other in simulations with a relatively large number of agents.

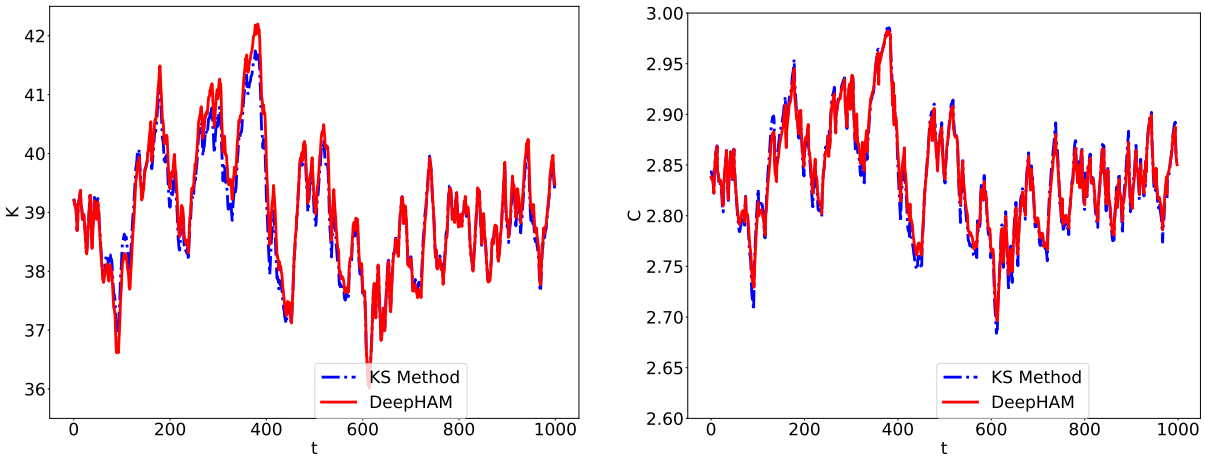


Figure 7: The simulated paths based on solutions obtained from the KS method and DeepHAM, under the same realization of idiosyncratic and aggregate shocks. Left panel: aggregate capital (K_t); right panel: aggregate consumption (C_t).

shown in Figure 7. This further confirms the accuracy of the DeepHAM solution for the Krusell-Smith model.

D.2 Scalability of Generalized Moment Representation

In Section 3.3.2, we examine the numerical benefit of using the generalized moment to represent the distribution, compared to using the full state vector in a Krusell-Smith model with $N = 50$. In this Section, we further extend our analysis to $N = 80$, increasing the state space. We want to examine the scalability of the generalized moment approach and the full state vector approach. The results, shown in Table 8 below, include outcomes for both $N = 50$ and $N = 80$ (highlighted with a light gray background).

As a reminder, Settings 3 and 4 for the full state vector involve using a significantly larger neural network than what is used for a single generalized moment. This was done to achieve optimal performance across various hyperparameter combinations, as detailed in Section 3.2.2, with the key hyperparameter changes reported in the third and fourth columns of Table 8.

From the comparison in the first two rows, we observe that when N increases from 50 to 80, our DeepHAM method based on a single generalized moment maintains (and even improves upon) the Bellman error with *the exact same set of hyperparameters* for the deep learning algorithm. The computational time only increases from 60 minutes to 82 minutes to achieve the same level of numerical accuracy.

In contrast, when using the full state vector representation, applying the same large

Method	# of agents N	# of neurons per layer (c_{NN})	# of SGD steps (c_{NN})	# of params (c_{NN})	Comput. time	Bellman error (std)
1 generalized moment	50	24	10000	938	60 min	0.0151 (0.0015)
1 generalized moment	80	24	10000	938	82 min	0.0140 (0.0008)
Full state (setting 3)	50	200	10000	51201	86 min	0.0243 (0.0031)
Full state (setting 3)	80	200	10000	57201	131 min	0.2367 (0.3944)
Full state (setting 4)	50	200	20000	51201	172 min	0.0225 (0.0006)
Full state (setting 4)	80	200	20000	57201	260 min	0.0328 (0.0092)

Table 8: Numerical performance comparison for Krusell-Smith model with the number of agents $N = 50$ and $N = 80$: generalized moments vs full state vector. Numbers in parentheses indicate the standard deviation of the Bellman error, based on 5 independent runs. Note: due to the large memory requirements of the full state vector representation, the experiments in this table were conducted on a server equipped with an A100 GPU with 80GB memory.

network and the same number of SGD update steps as in the single generalized moment scenario (Setting 3) leads to a significant decline in performance as N increases from 50 to 80. Even doubling the number of SGD updates (Setting 4) results in a noticeable increase in Bellman error.

These findings indicate that as the number of agents increases from 50 to 80, the full state vector representation demands considerably more computational resources—in terms of both additional parameters and training steps—to achieve comparable accuracy, unlike the generalized moment representation. Due to the current memory limitations of the A100 GPU, we conducted experiments with up to 80 agents. Notably, with the full state vector approach, we are unable to run experiments involving infrequent events as in 3.3.1, which typically require simulations with over 1,000 agents. We expect that this gap in efficiency between the two methods will become even more pronounced as the number of agents increases further.

E Deep Learning Hyperparameters

We present the parameters used in DeepHAM with generalized moments in Table 9, where Models 1, 2, and 3 correspond to the models from Krusell and Smith (1998), Fernández-Villaverde et al. (2019), and Davila et al. (2012), respectively. We always use a fully con-

nected feed-forward network with same number of neurons in each layer, and a $\tanh(\cdot)$ activation function. We optimize all network parameters using the Adam optimizer (Kingma and Ba, 2015) with $\epsilon = 1e-8$, $\beta_1 = 0.99$, $\beta_2 = 0.99$, and a constant learning rate. In our experiments with the full state vector used in Section 3.3.2, we kept all learning parameters unchanged, except for the number of neurons per layer and the number of outer iterations N_k (resulting in proportional changes in the total SGD steps for c_{NN}), as reported in Table 3.

	Model 1	Model 2	Model 3
Number of layers in $\mathcal{Q}_{\text{NN}}$	2	2	2
Neurons per layer in $\mathcal{Q}_{\text{NN}}$	12	12	12
Number of layers in V_{NN} and c_{NN}	2	2	2
Neurons per layer in V_{NN} and c_{NN}	24	64	24
Learning rate of V_{NN}	10^{-4}	10^{-4}	10^{-4}
Learning rate of c_{NN}	4×10^{-4}	4×10^{-4}	4×10^{-4}
Number of outer iterations N_k	5	8	20
Number of SGD steps for c_{NN} per outer iteration	2000	1600	400
Batch size N_{b_1} for learning V_{NN}	128	128	128
Batch size N_{b_2} for learning c_{NN}	384	384	384
Number of periods T in policy objective function	150	200	50
Number of periods T_{simul} for truncated total utility	800	2000	800

Table 9: Parameters of deep learning. Models 1, 2, and 3 correspond to models adapted from Krusell and Smith (1998), Fernández-Villaverde et al. (2019), and Davila et al. (2012), respectively.

F Computational Time Cost

We have now included detailed computational time results for all experiments across the different algorithms, along with the hardware configurations used in each case. This provides a clearer picture of the time costs for each approach. The table below reports the computational time for the Krusell-Smith (KS) model, Fernández-Villaverde et al. (2019) model, and the constrained efficiency problem without and with aggregate shocks:

References

Achdou, Yves, Jiequn Han, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll (2017), "Income and wealth distribution in macroeconomics: A continuous-time approach." Technical report, National Bureau of Economic Research.

Model	DeepHAM w/ 1st Moment	DeepHAM w/ GM
KS Model	1h13min	1h30min
Fernández-Villaverde et al. (2019) Model	1h12min	2h
Constrained Efficiency w/o agg shock	27min	36min
Constrained Efficiency w agg shock	39min	47min

Table 10: Comparison of computational time across different methods and models. All models are implemented on an A100 GPU on Google Colab with 40GB memory, a platform that is easily accessible to everyone with a Google account. The deep learning hyperparameters are reported in Appendix E.

Ahn, SeHyoun, Greg Kaplan, Benjamin Moll, Thomas Winberry, and Christian Wolf (2018), "When inequality matters for macro and macro matters for inequality." *NBER Macroeconomics Annual*, 32, 1–75.

Algan, Yann, Olivier Allais, Wouter J Den Haan, and Pontus Rendahl (2014), "Solving and simulating models with heterogeneous agents and aggregate uncertainty." In *Handbook of Computational Economics*, volume 3, 277–324, Elsevier.

Andersen, Asger Lau, Niels Johannesen, Mia Jørgensen, and José-Luis Peydró (2021), "Monetary policy and inequality." Technical report.

Arbex, Marcelo, Dennis O'Dea, and David Wiczer (2019), "Network search: Climbing the job ladder faster." *International Economic Review*, 60, 693–720.

Arellano, Cristina, Yan Bai, and Gabriel Mihalache (2024), "Deadly debt crises: Covid-19 in emerging markets." *Review of Economic Studies*, 91, 1243–1290.

Auclert, Adrien (2019), "Monetary policy and the redistribution channel." *American Economic Review*, 109, 2333–67.

Auclert, Adrien, Bence Bardóczy, Matthew Rognlie, and Ludwig Straub (2021), "Using the sequence-space Jacobian to solve and estimate heterogeneous-agent models." *Econometrica*, 89, 2375–2408.

Azinovic, Marlon, Luca Gaegauf, and Simon Scheidegger (2022), "Deep equilibrium nets." *International Economic Review*.

Azinovic, Marlon and Jan Žemlička (2023), "Economics-inspired neural networks with stabilizing homotopies." *arXiv preprint arXiv:2303.14802*.

Bayer, Christian and Ralph Luetticke (2020), "Solving discrete time heterogeneous agent models with aggregate risk and many idiosyncratic states by perturbation." *Quantitative Economics*, 11, 1253–1288.

Bhandari, Anmol, David Evans, Mikhail Golosov, and Thomas J Sargent (2021), "Inequality, business cycles, and monetary-fiscal policy." *Econometrica*, 89, 2559–2599.

Bianchi, Javier and Enrique G Mendoza (2018), "Optimal time-consistent macroprudential policy." *Journal of Political Economy*, 126, 588–634.

Boppart, Timo, Per Krusell, and Kurt Mitman (2018), "Exploiting MIT shocks in heterogeneous-agent economies: the impulse response as a numerical derivative." *Journal of Economic Dynamics and Control*, 89, 68–92.

Brown, George W (1951), "Iterative solution of games by fictitious play." *Activity analysis of production and allocation*, 13, 374–376.

Brunnermeier, Markus K and Yuliy Sannikov (2014), "A macroeconomic model with a financial sector." *American Economic Review*, 104, 379–421.

Cao, Dan, Wenlan Luo, and Guangyu Nie (2023), "Global DSGE models." *Review of Economic Dynamics*, 51, 199–225.

Caramp, Nicolas and Dejanir H Silva (2021), "Monetary policy and wealth effects: The role of risk and heterogeneity." Technical report.

Carmona, René and François Delarue (2018), *Probabilistic Theory of Mean Field Games with Applications II: Mean Field Games with Common Noise and Master Equations*, volume 84. Springer.

Chang, Minsu, Xiaohong Chen, and Frank Schorfheide (2021), "Heterogeneity and aggregate fluctuations." Technical report, National Bureau of Economic Research.

Chatterjee, Satyajit and Burcu Eyigungor (2015), "A seniority arrangement for sovereign debt." *American Economic Review*, 105, 3740–3765.

Chetty, Raj (2009), "Sufficient statistics for welfare analysis: A bridge between structural and reduced-form methods." *Annual Review of Economics*, 1, 451–488.

Cioffi, Riccardo A (2021), "Heterogeneous risk exposure and the dynamics of wealth inequality." Technical report.

Cybenko, George (1989), "Approximation by superpositions of a sigmoidal function." *Mathematics of control, signals and systems*, 2, 303–314.

Davila, Julio, Jay H Hong, Per Krusell, and José-Víctor Ríos-Rull (2012), "Constrained efficiency in the neoclassical growth model with uninsurable idiosyncratic shocks." *Econometrica*, 80, 2431–2467.

Den Haan, Wouter J (2010), "Comparison of solutions to the incomplete markets model with aggregate uncertainty." *Journal of Economic Dynamics and Control*, 34, 4–27.

Duarte, Victor, Diogo Duarte, and Dejanir Silva (2024), "Machine learning for continuous-time finance." *Review of Financial Studies*.

Dvorkin, Maximiliano, Juan M Sánchez, Horacio Sapriza, and Emircan Yurdagul (2021), "Sovereign debt restructurings." *American Economic Journal: Macroeconomics*, 13, 26–77.

E, Weinan, Jiequn Han, and Arnulf Jentzen (2021a), "Algorithms for solving high dimensional pdes: From nonlinear monte carlo to machine learning." *Nonlinearity*, 35, 278.

E, Weinan, Jiequn Han, and Linfeng Zhang (2021b), "Machine-learning-assisted modeling." *Physics Today*, 74, 36–41.

Feng, Zhigang, Jiequn Han, and Shenghao Zhu (2024), "Optimal taxation with incomplete markets–an exploration via reinforcement learning." *Available at SSRN 4758552*.

Fernández-Villaverde, Jesús, Samuel Hurtado, and Galo Nuno (2019), "Financial frictions and the wealth distribution." Technical report, National Bureau of Economic Research.

Fernández-Villaverde, Jesús, Galo Nuno, George Sorg-Langhans, and Maximilian Vogler (2020), "Solving high-dimensional dynamic programming problems using deep learning."

Fernández-Villaverde, Jesús, Juan Francisco Rubio-Ramírez, and Frank Schorfheide (2016), "Solution and estimation methods for DSGE models." In *Handbook of macroeconomics*, volume 2, 527–724, Elsevier.

Friedl, Aleksandra, Felix Kübler, Simon Scheidegger, and Takafumi Usui (2023), "Deep uncertainty quantification: with an application to integrated assessment models."

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016), *Deep learning*. MIT press.

Gopalakrishna, Goutham (2022), "ALIENs and continuous time economies."

Gopalakrishna, Goutham, Zhouzhou Gu, and Jonathan Payne (2024), "Asset pricing, participation constraints, and inequality." Technical report.

Gu, Zhouzhou, Mathieu Laurière, Sebastian Merkel, and Jonathan Payne (2023), "Deep learning solutions to master equations for continuous time heterogeneous agent macroeconomic models." Technical report.

Han, Jiequn and Weinan E (2016), "Deep learning approximation for stochastic control problems." *arXiv preprint arXiv:1611.07422, Deep Reinforcement Learning Workshop, Conference on Neural Information Processing Systems*.

Han, Jiequn and Ruimeng Hu (2020), "Deep fictitious play for finding markovian nash equilibrium in multi-agent games." In *Mathematical and Scientific Machine Learning*, 221–245, PMLR.

Han, Jiequn, Arnulf Jentzen, and Weinan E (2018), "Solving high-dimensional partial differential equations using deep learning." *Proceedings of the National Academy of Sciences*, 115, 8505–8510.

Han, Jiequn, Yingzhou Li, Lin Lin, Jianfeng Lu, Jiefu Zhang, and Linfeng Zhang (2022), "Universal approximation of symmetric and anti-symmetric functions." *Communications in Mathematical Sciences*, 20, 1397–1408.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989), "Multilayer feedforward networks are universal approximators." *Neural networks*, 2, 359–366.

Hu, Ruimeng (2021), "Deep fictitious play for stochastic differential games." *Communications in Mathematical Sciences*, 19, 325–353.

Huang, Ji (2023), "A probabilistic solution to high-dimensional continuous-time macro and finance models."

Kahou, Mahdi Ebrahimi, Jesús Fernández-Villaverde, Jesse Perla, and Arnav Sood (2021), "Exploiting symmetry in high-dimensional dynamic programming." Technical report.

Kaplan, Greg, Kurt Mitman, and Giovanni L Violante (2020), "The housing boom and bust: Model meets evidence." *Journal of Political Economy*, 128, 3285–3345.

Kaplan, Greg, Benjamin Moll, and Giovanni L Violante (2018), "Monetary policy according to HANK." *American Economic Review*, 108, 697–743.

Kaplan, Greg and Giovanni L Violante (2018), "Microeconomic heterogeneity and macroeconomic shocks." *Journal of Economic Perspectives*, 32, 167–94.

Kaplan, Greg, Giovanni L Violante, and Justin Weidner (2014), "The wealthy hand-to-mouth." *Brookings Papers on Economic Activity*, 2014, 77–138.

Kargar, Mahyar, Juan Passadore, and Dejanir Silva (2020), "Liquidity and risk in OTC markets: A theory of asset pricing and portfolio flows."

Kase, Hanno, Leonardo Melosi, and Matthias Rottner (2022), "Estimating nonlinear heterogeneous agents models with neural networks."

Kekre, Rohan and Moritz Lenel (2021), "Monetary policy, redistribution, and risk premia." Technical report, National Bureau of Economic Research.

Khan, Aubhik and Julia K Thomas (2013), "Credit shocks and aggregate fluctuations in an economy with production heterogeneity." *Journal of Political Economy*, 121, 1055–1107.

Kingma, Diederik P. and Jimmy Ba (2015), "Adam: A method for stochastic optimization." In *3rd International Conference on Learning Representations, ICLR 2015* (Yoshua Bengio and Yann LeCun, eds.).

Krueger, Dirk, Kurt Mitman, and Fabrizio Perri (2016), "Macroeconomics and household heterogeneity." In *Handbook of macroeconomics*, volume 2, 843–921, Elsevier.

Krusell, Per, Toshihiko Mukoyama, Ayşegül Şahin, and Anthony A Smith Jr (2009), "Revisiting the welfare effects of eliminating business cycles." *Review of Economic Dynamics*, 12, 393–404.

Krusell, Per and Anthony A Smith, Jr (1998), "Income and wealth heterogeneity in the macroeconomy." *Journal of Political Economy*, 106, 867–896.

Le Grand, François, Alaïs Martin-Baillon, and Xavier Ragot (2021), "Should monetary policy care about redistribution? optimal fiscal and monetary policy with heterogeneous agents." Technical report, Working Paper, SciencesPo.

Lise, Jeremy and Jean-Marc Robin (2017), "The macrodynamics of sorting between workers and firms." *American Economic Review*, 107, 1104–1135.

Liu, Laura and Mikkel Plagborg-Møller (2019), "Full-information estimation of heterogeneous agent models using macro and micro data." Technical report.

Ljungqvist, Lars and Thomas J Sargent (2018), *Recursive macroeconomic theory*. MIT press.

Maliar, Lilia and Serguei Maliar (2022), "Deep learning classification: Modeling discrete labor choice." *Journal of Economic Dynamics and Control*, 135, 104295.

Maliar, Lilia, Serguei Maliar, and Fernando Valli (2010), "Solving the incomplete markets model with aggregate uncertainty using the krusell–smith algorithm." *Journal of Economic Dynamics and Control*, 34, 42–49.

Maliar, Lilia, Serguei Maliar, and Pablo Winant (2021), "Deep learning for solving dynamic economic models." *Journal of Monetary Economics*, 122, 76–101.

McKay, Alisdair and Ricardo Reis (2016), "The role of automatic stabilizers in the US business cycle." *Econometrica*, 84, 141–194.

Nuño, Galo and Benjamin Moll (2018), "Social optima in economies with heterogeneous agents." *Review of Economic Dynamics*, 28, 150–180.

Payne, Jonathan, Adam Rebei, and Yucheng Yang (2024), "Deep learning for search and matching models." Technical report.

Petrosky-Nadeau, Nicolas, Lu Zhang, and Lars-Alexander Kuehn (2018), "Endogenous disasters." *American Economic Review*, 108, 2212–45.

Reiter, Michael (2009), "Solving heterogeneous-agent models by projection and perturbation." *Journal of Economic Dynamics and Control*, 33, 649–665.

Schaab, Andreas (2020), "Micro and macro uncertainty." Technical report.

Scheidegger, Simon and Ilias Bilionis (2019), "Machine learning for high-dimensional dynamic stochastic economies." *Journal of Computational Science*, 33, 68–82.

Silver, David, Aja Huang, Chris J Maddison, et al. (2016), "Mastering the game of Go with deep neural networks and tree search." *Nature*, 529, 484–489.

Valaitis, Vytautas and Alessandro Villa (2021), "A machine learning projection method for macro-finance models."

Winberry, Thomas (2018), "A method for solving and estimating heterogeneous agent macro models." *Quantitative Economics*, 9, 1123–1151.

Woodford, Michael and Yinxi Xie (2022), "Fiscal and monetary stabilization policy at the zero lower bound: consequences of limited foresight." *Journal of Monetary Economics*, 125, 18–35.

Yang, Yucheng (2022), "Redistributive inflation and optimal monetary policy." *Available at*

SSRN 4275770.

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, et al. (2017), "Deep sets." *Advances in neural information processing systems*, 30.