

# The Blessings of Overparameterization: Applications in Solving Economic Models

---

Mahdi Ebrahimi Kahou<sup>1</sup>    Jesús Fernández-Villaverde<sup>2</sup>

<sup>1</sup>Bowdoin College

<sup>2</sup>University of Pennsylvania

# Motivation, Question, and Contribution

---

“I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk.”—*Enrico Fermi*

- Deep learning has proved remarkably capable at solving **high-dimensional problems**.
- Economists are now using it to find accurate approximate solutions to dynamic models once thought intractable.
- These networks are **massively overparameterized**, with far more parameters than “data” points.
- Standard statistical wisdom says: more parameters than “data” points leads to **overfitting** and poor out-of-grid performance.

## Question

---

When solving dynamic models with neural networks

- Does overparameterization lead to poor approximation in out-of-grid performance?
- Is overparameterization a “necessary evil” that comes with modern machine learning methods, or does it give us something extra?

1. **No overfitting:** as the number of network parameters grows, the solutions become more accurate.
2. **Algorithmic stability:** the approximate solutions of dynamic models in economics are unstable; overparameterization increases stability.

## Implication

When in doubt, use a bigger network. Overparameterization is not a source of risk to be managed but a feature to be exploited.

**Background: Economic Models,  
Approximate Solutions, and  
Overparameterization**

---

# An economic model is a system of functional equations

Many theoretical models can be written as functional equations:

- Economic object of interest:  $f$ , where  $f : \mathcal{X} \rightarrow \mathcal{R} \subseteq \mathbb{R}^N$ 
  - e.g., production, investment choice, best-response, etc.
- Domain of  $f$ :  $\mathcal{X}$ 
  - e.g. space of capital, opponents state, etc.
- The “Economics model”:  $\ell(x, f)$ 
  - e.g., Euler and Bellman residuals, equilibrium FOCs.

Then a **solution** is  $f^* \in \mathcal{F}$  where  $\ell(x, f^*) = \mathbf{0}$  for all  $x \in \mathcal{X}$ .

## Approximate solution

1. Sample  $\mathcal{X}$ :  $\mathcal{D} = \{x_1, \dots, x_N\}$
2. Pick a family of parametric functions (e.g., polynomials, Chebyshev polynomials, neural networks)  $f_\theta(\cdot)$ :
  - $\theta = \{\theta_1, \dots, \theta_M\}$ : parameters for optimization (e.g., coefficients, or weights and biases).
3. To find an approximation for  $f$  solve:

$$\min_{\theta} \frac{1}{N} \sum_{x \in \mathcal{D}} \left\| \underbrace{\ell(x, f_\theta)}_{\text{Econ model error}} \right\|_2^2$$

# Neural Networks and Overparameterization

**Neural Networks** form a **highly-overparameterized** ( $M \gg N$ ) class of functions.

- Example: one layer neural network,  $f_\theta : \mathbb{R}^Q \rightarrow \mathbb{R}$ :

$$f_\theta(x) = W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2$$

- $W_1 \in \mathbb{R}^{P \times Q}$ ,  $b_1 \in \mathbb{R}^{P \times 1}$ ,  $W_2 \in \mathbb{R}^{1 \times P}$ , and  $b_2 \in \mathbb{R}$ .
- $\theta \equiv \{b_1, W_1, b_2, W_2\}$  are the coefficients, in this example  $M = PQ + P + P + 1$ .
- $\sigma(\cdot)$  is a nonlinear function applied element-wise (e.g.,  $\max\{\cdot, 0\}$ ).
- Making it “deeper” by adding another “layer”:  $f_\theta(x) \equiv W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2) + b_3$ .

# Applications

---

# Experimental design

- We demonstrate these properties across three canonical models with known benchmarks:

## 1. Linear-Quadratic

Closed-form solution. Sharpest test of approximation error.

## 2. McCall Job Search

1D state, kink at reservation wage. Non-smooth approximation.

## 3. Real Business Cycle

2D stochastic state. No closed form. Benchmarked vs. VFI.

- In all three: **accuracy** and **stability** improve with overparameterization.

## What does “solving” this model mean?

- The problem has a recursive representation in the form of a PDE, the **Hamilton–Jacobi–Bellman equation**:

$$\rho V = \max_c \left\{ u(c) + V_k [e^z k^\alpha - \delta k - c] - \mu z V_z + \frac{1}{2} \sigma^2 V_{zz} \right\}$$

- The unknown is the **value function**  $V(k, z)$ :
  - The first-order condition  $u'(c) = V_k(k, z)$  gives the optimal policy  $c = (u')^{-1}(V_k)$ .
  - From the capital equation, the drift is  $\dot{k} = e^z k^\alpha - \delta k - (u')^{-1}(V_k)$ .
- “Solving the model” means finding  $V(\cdot)$  that satisfies the HJB equation over the state space.
- In general, no closed-form expression exists.
- An alternative is to solve the Euler equation:

$$\frac{d[u'(c_t)]}{u'(c_t)} = [\rho - (\alpha e^{z_t} k_t^{\alpha-1} - \delta)] dt + \sigma \frac{u''(c_t)}{u'(c_t)} dz(k_t, z_t) dW_t$$

## The mathematical structure you already know

- Rewrite the HJB equation as:

$$\mathcal{T}(V) \equiv \rho V - \max_c \{ u(c) + V_k [e^z k^\alpha - \delta k] - \mu z V_z + \frac{1}{2} \sigma^2 V_{zz} \} = 0$$

- Strip away the economics, and you are looking for a function  $V$  such that:

$$\mathcal{T}(V) = \mathbf{0}$$

where  $\mathcal{T}$  is a nonlinear operator between function spaces. Nothing exotic here.

- The HJB equation is a second-order nonlinear PDE. You have seen these in optimal control and differential games.
- The classical numerical approach is exactly what you would do:
  1. Propose a parameterized approximation  $\widehat{V}_\theta$ .
  2. Evaluate the residual  $\mathcal{R}[\widehat{V}_\theta](X)$  at a collection of points.
  3. Find  $\theta$  that drives the residual to zero.

## So what makes the economics problem different?

- When you solve a typical PDE, you know the equation before you write a single line of code. The coefficients, the forcing term, the boundary conditions: all given.
- You can pick your grid, choose your discretization, and refine your mesh. The equation never changes on you. Better numerics give a better approximation to a **fixed target**.
- In economics, this is not the case. The law of motion for the state variables is the policy function we are trying to find. Agents observe the current state, make a choice, and that choice determines where the system goes next.
- So improving your approximation changes the dynamics, which changes which parts of the state space the system visits, which changes where your approximation needs to be accurate.

### The difference in one sentence

In typical natural science problems, the dynamics are an input. In economics, the dynamics are the output.

## The equilibrium loop

- Return to the HJB equation:

$$\rho V = u((u')^{-1}(V_k)) + V_k [e^z k^\alpha - \delta k - (u')^{-1}(V_k)] - \mu z V_z + \frac{1}{2} \sigma^2 V_{zz}$$

- We already mentioned this is a second-order nonlinear PDE.
- But look at the drift term for  $k$ :  $e^z k^\alpha - \delta k - (u')^{-1}(V_k)$ .
- The law of motion for the state depends on  $V$ , the very function we are solving for!
- The equilibrium loop:** the “coefficients” of the PDE are endogenous to the optimal choices of the agents.

## Why it matters

- Since the drift depends on  $V$ , we need to choose where to evaluate the residual.
- A natural choice: weighting the residual by the **ergodic distribution**  $\xi^*$ :

$$\theta^* = \arg \min_{\theta} \int \mathcal{R}[\widehat{V}_{\theta}](X)^2 d\xi^*(X)$$

- But  $\xi^*$  is generated by the law of motion, which depends on  $V$ , the function we are solving for:

$$V \longrightarrow \text{drift of } k \longrightarrow \xi^* \longrightarrow \text{evaluation points} \longrightarrow \widehat{V}_{\theta} \longrightarrow V$$

- Improving  $\widehat{V}_{\theta}$  changes the drift, changes  $\xi^*$ , and changes where the nodes should have been. Moving the net moves the target.

### Practical fix

Iterate between sampling and optimization. Simulate from the current  $\widehat{V}_{\theta}$ , recompute evaluation points, re-optimize. Convergence is not guaranteed but works well in practice.

## Why traditional methods hit a wall

- When the state space is low-dimensional ( $N \leq 4$ ), everything we just described works fine: lay down a grid or use Chebyshev collocation and solve.
- But many of the questions economists want to answer require large  $N$ :
  - Heterogeneous households differing in wealth, income, age, and portfolio composition.
  - Firms interacting strategically in a dynamic oligopoly.
  - Spatial models with many regions linked by trade and migration.
- Grid-based methods need  $m^N$  points. Chebyshev collocation with tensor products needs  $D^N$  terms. Both become infeasible beyond  $N = 5$  or so.
- Perturbation (Taylor expansion around steady state) sidesteps the grid but is only locally accurate. It fails with kinks, occasionally binding constraints, or large shocks.

## Enter neural networks (the part you know)

- The idea is simple: replace the fixed polynomial basis with a neural network:

$$\hat{d}_\theta(X) = f_\theta(X), \quad \theta^* = \arg \min_{\theta} \frac{1}{L} \sum_{l=1}^L \mathcal{R}[f_\theta](X_l)^2$$

- You already know why this helps: universal approximation, dimension-free rates for the Barron class (Barron, 1993), learned representations. I will take such background as given.
- What is specific to our setting:
  1. The residual  $\mathcal{R}$  is not a supervised-learning loss. It comes from the model's equilibrium conditions (Bellman, Euler, market clearing).
  2. Evaluation points  $\{X_l\}$  are regenerated each epoch from the current approximation (the equilibrium loop we just discussed).
  3. There may be additional constraints: transversality conditions, non-negativity of consumption, budget balance.

# The blessings of overparameterization

---

## The question this paper asks

- Now that you know the setup (finding functions that satisfy operator equations, with the complication that the sampling measure is endogenous), a practical question arises:
- **How large should the network be?**
- Standard statistical wisdom says: more parameters than evaluation points  $\Rightarrow$  overfitting  $\Rightarrow$  poor out-of-grid performance.
- If true here, the practitioner must carefully tune network width, adding a fragile hyperparameter choice to an already demanding computational problem.

### Thesis

This paper argues the opposite. In the context of solving these operator equations, overparameterization is a **blessing**. It improves both accuracy and, more importantly, algorithmic stability.

## Two findings

- **Finding 1, no overfitting:** as the number of network parameters grows, out-of-grid residuals (HJB and Euler) decrease. The approximate value and policy functions converge toward their benchmark counterparts.
- **Finding 2, algorithmic stability:** with small networks, solutions obtained from different random initializations of  $\theta$  disagree substantially.
  - The researcher has **no way of knowing** which initialization produced the better solution. The “answer” depends on the random seed.
  - As network width increases, this disagreement collapses: 50 independent random seeds converge to the same function.

### Practical implication

When in doubt, use a wider network. Overparameterization is not a source of risk to be managed but a feature to be exploited.

## Connection to the machine learning literature

- Our findings connect to a broader phenomenon in modern machine learning.

### Double descent

**Belkin et al. (2019)**: the classical bias–variance tradeoff breaks down for overparameterized models. Test error can decrease again after an initial overfitting peak.

### Benign overfitting

**Bartlett et al. (2020)**: minimum-norm interpolants can generalize well even when parameters vastly exceed observations.

- **Our contribution**: this benign behavior extends to function approximation problems in computational economics, where the loss is defined by **equilibrium conditions** (HJB, Bellman, or Euler equations) rather than by labeled data.

## Contrast with perturbation methods

- A central challenge in solving nonlinear models by perturbation: higher-order Taylor expansions generate explosive sample paths.
- Standard fix: **pruning** (Kim et al., 2008; Andreasen et al., 2018). Cross-products of higher-order terms are deliberately dropped to restore stability.
- Pruning is an exercise in deliberate under-parameterization: parameters are removed to make the approximation well-behaved.
- Our paper delivers the **opposite message**: in the neural network approach, increasing network width is what delivers accuracy and stability.

### Slogan

The perturbation literature prunes to survive; neural network solvers grow to thrive.

- We demonstrate these properties across three canonical models with known benchmarks:

## 1. Linear-Quadratic

Closed-form solution. Sharpest test of approximation error.

## 2. McCall Job Search

1D state, kink at reservation wage. Non-smooth approximation.

## 3. Real Business Cycle

2D stochastic state. No closed form. Benchmarked vs. VFI.

- In all three: accuracy and stability improve with width. No sign of overfitting.
- Theorem? So far, just a conjecture.

# Network architecture and training protocol

- **Hidden width**  $H$  varied systematically.
- One hidden layer:  $f_{\theta}(x) = W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}$ , with  $H(n + 2) + 1$  parameters.
- Two hidden layers:  $f_{\theta}(x) = W^{(3)}\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}) + b^{(3)}$ , with  $H(n + H + 3) + 1$  parameters.
- **Activation functions:** ReLU  $\sigma(z) = \max\{0, z\}$  (baseline), Leaky ReLU  $\sigma(z) = \max\{\alpha z, z\}$  with  $\alpha = 0.01$ , and Sigmoid  $\sigma(z) = 1/(1 + e^{-z})$ .
- **Training:** minimize squared residual loss (Euler or Bellman) using Adam.
- **Key design choice:** 50 independent random seeds per specification. This lets us measure both accuracy (median across seeds) and stability (width of the 10th–90th percentile band).

# Application 1: Linear-Quadratic regulator

---

## The LQ problem

- The general deterministic LQ problem:

$$v(x) = \max_u \{ -x^\top R x - u^\top Q u + \beta v(x') \}$$
$$\text{s.t. } x' = Ax + Bu, \quad x_0 \text{ given}$$

- Closed-form solution:

$$v(x) = -x^\top P x, \quad u(x) = -F x$$

where  $P$  and  $F$  are obtained from the discrete-time algebraic Riccati equation.

- Sharpest test: approximation error can be computed exactly at every point.

## A competitive firm with adjustment costs

- A price-taking firm in a competitive industry.
- Own output  $y$ , aggregate output  $Y$  (exogenous). Inverse demand:  $\alpha_0 - \alpha_1 Y$ .
- The firm chooses investment  $u$  subject to quadratic adjustment cost  $\frac{\gamma}{2}u^2$ :

$$v(y, Y) = \max_u \left\{ (\alpha_0 - \alpha_1 Y)y - \frac{\gamma}{2}u^2 + \beta v(y', Y') \right\}$$
$$\text{s.t. } Y' = h_0 + h_1 Y, \quad y' = y + u$$

- State is two-dimensional:  $(y, Y)$ . With augmented state  $x = (1, y, Y)^\top$ , this maps into the general LQ framework.
- Euler equation:

$$\gamma u(y, Y) = \beta [\gamma u(y', Y') + (\alpha_0 - \alpha_1 Y')]$$

Description	Symbol	Value	
Discount factor	$\beta$	0.9	
Demand intercept	$\alpha_0$	1.0	inverse demand level
Demand slope	$\alpha_1$	1.3	price sensitivity to aggregate
Adjustment cost	$\gamma$	100	convex investment cost
Aggregate drift	$h_0$	0.05	constant in $Y' = h_0 + h_1 Y$
Aggregate persistence	$h_1$	0.9	AR coefficient for $Y$

## LQ: deep learning implementation

- **Architecture:** one-hidden-layer ReLU networks for value and policy functions. Hidden units  $H = 1$  to  $H = 7$  (4 to 22 parameters). 50 seeds per configuration.
- **Training data:** the optimal  $u$  depends only on  $Y$  ( $y$  enters linearly and does not affect the marginal return to investment), so we minimize the mean squared Euler residual over  $n_Y = 6$  grid points:

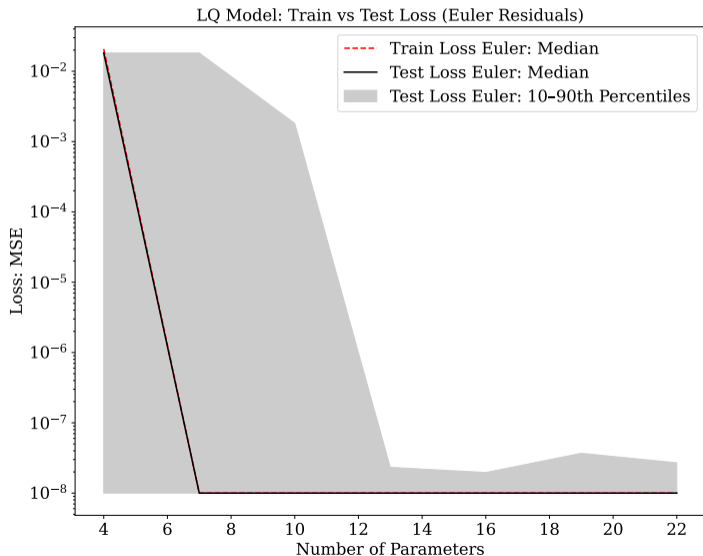
$$\min_{\theta_u} \frac{1}{n_Y} \sum_{i=1}^{n_Y} [\gamma u(Y_i; \theta_u) - \beta(\gamma u(Y'_i; \theta_u) + \alpha_0 - \alpha_1 Y'_i)]^2$$

The value function is then trained on a  $12 \times 12$  Cartesian grid of  $(y, Y)$  pairs. Adam optimizer with early stopping at loss  $< 10^{-8}$ .

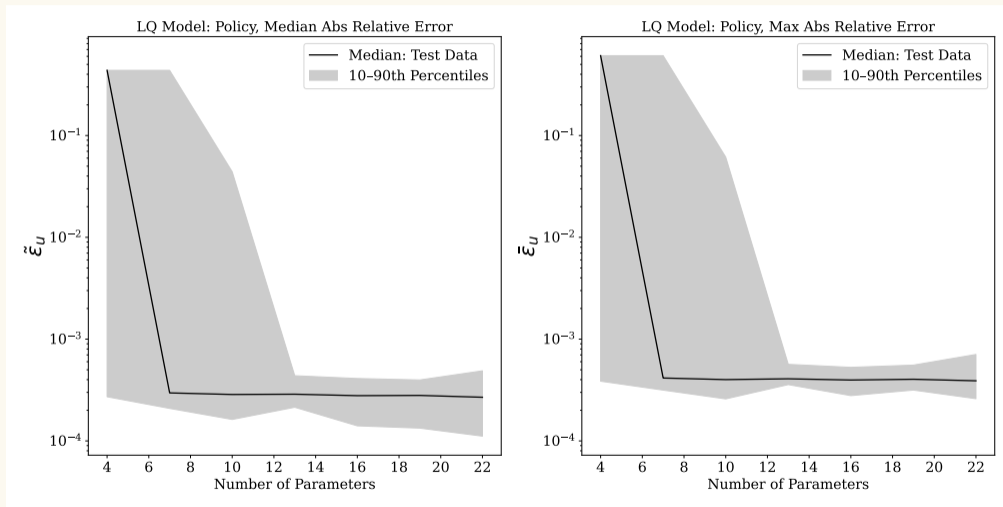
- **Test data:**  $T = 29$  period path simulated from fixed initial conditions. Test points lie in the interior of the state space and never on a training node.

- Red dots:  $12 \times 12$  training grid. Blue crosses:  $T = 29$  test path simulated from fixed initial conditions.
- Test points lie strictly in the interior of the state space and never coincide with a training node.

## LQ: train and test Euler-residual loss

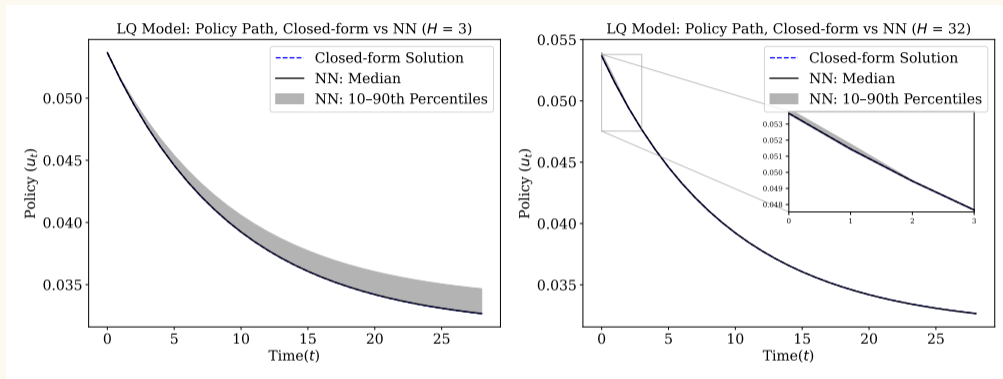


# LQ: absolute relative error vs. closed-form solution



- Median (left) and maximum (right) absolute relative error of the NN policy. Both statistics and the

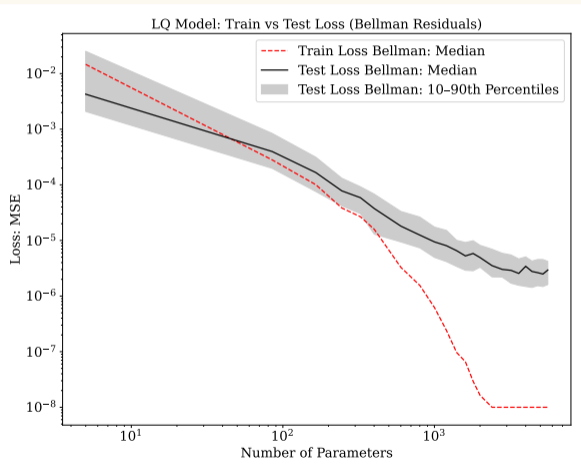
# LQ: policy paths, underparameterized vs. overparameterized



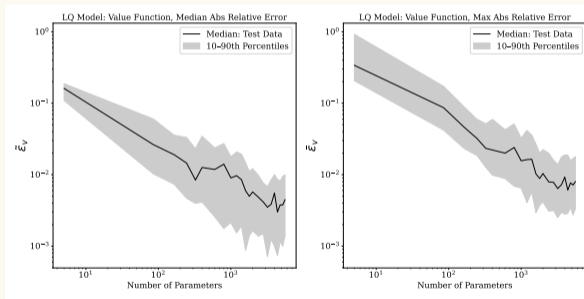
- Left:  $H = 3$  (underparameterized). Median is roughly accurate, but the across-seed band is wide.
- Right:  $H = 32$  (overparameterized). Median tracks the closed-form solution closely and the band collapses.

# LQ: value function results

(A) Train and test Bellman MSE



(B) Median and max relative error



- Value function network (trained conditional on estimated policy) displays the same pattern: accuracy

## **Application 2: McCall job-search model**

---

## From a linear to a non-smooth operator equation

- The LQ model is linear, which makes it the cleanest test but also the easiest.
- Does the blessing survive when the operator  $\mathcal{T}$  is nonlinear and the solution is non-smooth?
- The McCall model (McCall, 1970) provides exactly this test: the value function has a **kink** at the reservation wage, so the network must approximate a non-differentiable function.
- An unemployed worker receives i.i.d. wage offers  $w \sim f$  on  $[0, B]$  each period.
- Accept: permanent income stream  $w/(1 - \beta)$ . Reject: unemployment compensation  $c$  plus continuation value.
- Bellman equation (a nonlinear integral equation with a max operator):

$$v(w) = \max \left\{ \frac{w}{1 - \beta}, c + \beta \int_0^B v(w') f(w') dw' \right\}$$

## McCall: the kink and why it matters

- Optimal policy: threshold form. There exists a reservation wage  $\bar{w}$  such that accept iff  $w > \bar{w}$ .
- Closed-form value function:

$$v(w) = \begin{cases} \bar{w}/(1 - \beta) & \text{if } w \leq \bar{w}, \\ w/(1 - \beta) & \text{if } w > \bar{w}. \end{cases}$$

- The reservation wage  $\bar{w}$  solves:

$$\bar{w} - c = \frac{\beta}{1 - \beta} \int_{\bar{w}}^B (w' - \bar{w}) f(w') dw'$$

LHS: net gain from rejecting the marginal offer. RHS: expected gain from continued search.

- The kink at  $\bar{w}$  is a demanding test: the network must learn a non-smooth function from only  $N = 12$  training points.

# Parameterization

Description	Symbol	Value	
Discount factor	$\beta$	0.9	
Wage support	$B$	1.0	upper bound of $U[0, B]$
Unemployment benefit	$c$	0.1	flow payoff while searching

## McCall: deep learning implementation

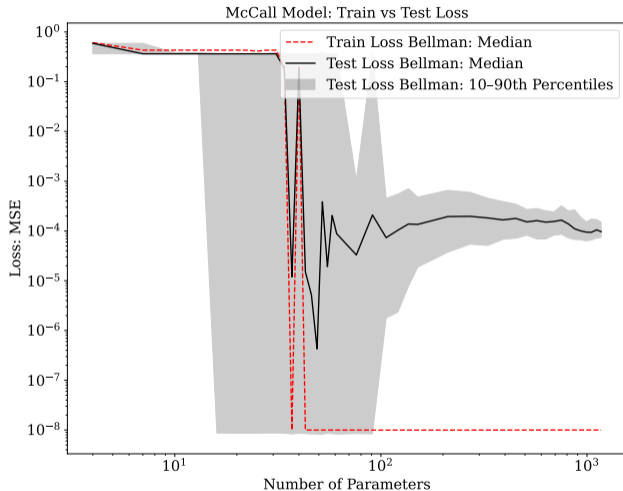
- **Architecture:** one-hidden-layer ReLU network  $v(w; \theta_v)$ , linear output. 50 random seeds per width.
- **Training:**  $N = 12$  uniform grid on  $[0, 1]$ . Minimize mean squared Bellman residual:

$$\min_{\theta_v} \frac{1}{N} \sum_{w_i \in \mathcal{D}} \left[ v(w_i; \theta_v) - \max \left\{ \frac{w_i}{1 - \beta}, c + \beta \int_0^B v(w'; \theta_v) f(w') dw' \right\} \right]^2$$

Integral by quadrature. Adam with early stopping at  $10^{-8}$ .

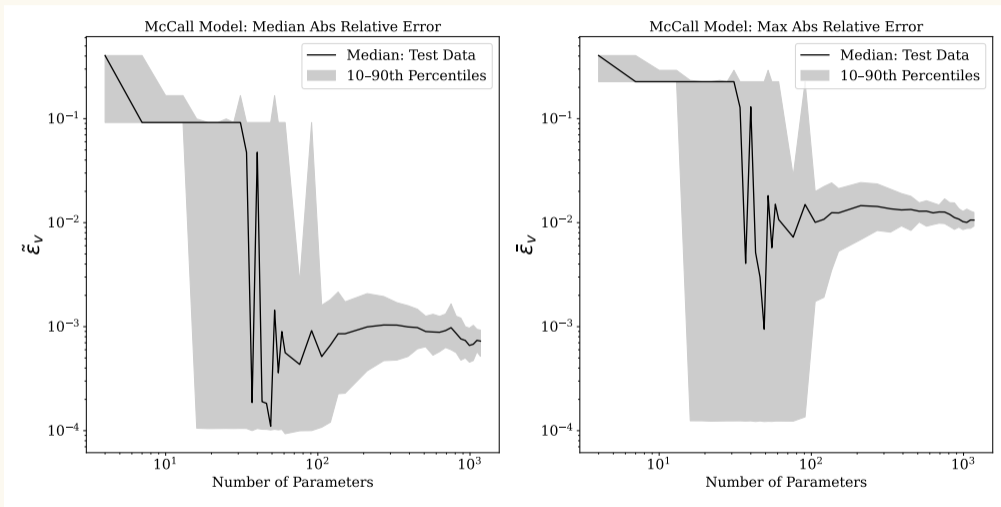
- **Test:** finer  $N = 100$  uniform grid, strictly outside the training set.
- **Accuracy metric:** absolute relative error  $\varepsilon_v(w) = |v(w; \theta_v^*) - v(w)| / |v(w)|$ . Summarized by max and median across test grid, then 10th/50th/90th percentiles across 50 seeds.

# McCall: train and test Bellman-residual loss



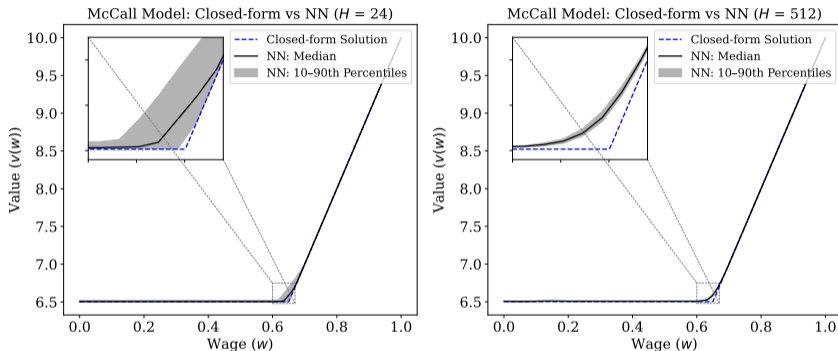
- Train loss hits the early-stopping floor at the interpolation threshold ( $\sim 12$  parameters, matching  $N = 12$  training points).
- Test loss descends again in the overparameterized regime: the **double-descent pattern**.
- The across-seed band collapses.

# McCall: absolute relative error



- Median (left) and max (right) relative error fall from  $\sim 30\%$  to  $\sim 2\%$  as the network enters the

# McCall: value function, underparameterized vs. overparameterized



- Left:  $H = 24$  (underparameterized). Median tracks the closed-form solution well, but the across-seed band is wide.
- Right:  $H = 512$  (overparameterized). Band is nearly invisible: 50 random initializations converge to the same function.

## **Application 3: Real Business Cycle model**

---

## From deterministic to stochastic, from closed form to none

- The LQ model is linear with a closed-form solution. The McCall model is nonlinear with a kink but still one-dimensional and still admits a closed form.
- The Real Business Cycle model (Kydlan and Prescott, 1982) is the most demanding test: a two-dimensional stochastic state space  $(k_t, z_t)$  and no analytical solution.
- Social planner, Cobb–Douglas production  $e^{z_t} k_t^\alpha$ , stochastic TFP:

$$\max_{\{c_t, k_{t+1}\}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \quad \text{s.t.} \quad c_t + k_{t+1} = e^{z_t} k_t^\alpha + (1 - \delta)k_t$$
$$z_t = \rho z_{t-1} + \sigma \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, 1)$$

- Key object: policy function  $k'(k, z)$ . Benchmark: value function iteration (VFI) on a fine grid.

## RBC: the Euler equation

- Euler equation:

$$u'(c_t) = \beta \mathbb{E}_t [u'(c_{t+1}) (\alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} + 1 - \delta)]$$

- **LHS**: marginal utility cost of saving one unit today.
- **RHS**: expected discounted marginal utility gain from extra capital tomorrow (marginal product plus undepreciated capital).
- Nonlinearity plus stochastic two-dimensional state means no analytical solution.
- This is the operator equation from slide 1, now in its full stochastic form.

Description	Symbol	Value	
Discount factor	$\beta$	0.96	
Capital share	$\alpha$	1/3	Cobb–Douglas exponent
Depreciation rate	$\delta$	0.1	per-period capital depreciation
TFP persistence	$\rho$	0.9	AR(1) coefficient for $z_t$
TFP volatility	$\sigma$	0.01	std. dev. of $\varepsilon_t$
Utility function	$u(c)$	$\ln c$	log utility

## RBC: deep learning implementation

- **Architecture:** one-hidden-layer ReLU network  $k'(k, z; \theta)$ . Output layer: Softplus activation  $\log(1 + e^x)$  to enforce  $k' > 0$ . 50 seeds per width.
- **Training:**  $20 \times 20$  Cartesian grid ( $N = 400$ ). Euler residual:

$$\ell_{\text{Euler}}(k, z; \theta) = \frac{1}{c(k, z; \theta)} - \beta \sum_{i=1}^M \frac{w_i}{\sqrt{\pi}} \left[ \frac{\alpha e^{\rho z + \sqrt{2}\sigma\zeta_i} k'(k, z; \theta)^{\alpha-1} + 1 - \delta}{c(k'(k, z; \theta), \rho z + \sqrt{2}\sigma\zeta_i; \theta)} \right]$$

with  $M = 15$  Gauss–Hermite nodes. Adam with StepLR scheduler.

- **Penalty term:**  $\lambda(k'(k^*, 0; \theta) - k^*)^2$  with  $\lambda = 0.01$ , enforcing a necessary implication of the transversality condition at the steady state.

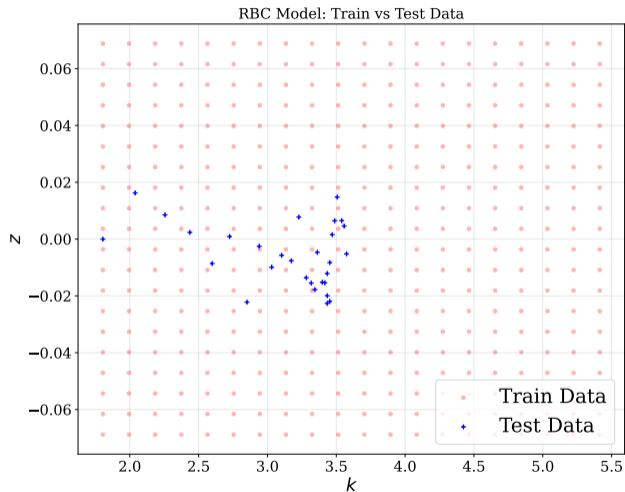
## RBC: why the penalty term?

- The Euler equation is a necessary but not sufficient condition for optimality.
- Both the stable and explosive solution manifolds through the steady state satisfy it identically.
- What rules out explosive solutions is the transversality condition (TVC):

$$\lim_{t \rightarrow \infty} \beta^t u'(c_t) k_{t+1} = 0$$

- The Euler residual loss does not enforce the TVC directly.
- The penalty resolves this multiplicity by imposing  $k'(k^*, 0) = k^*$ : the steady state is a fixed point of the policy function.
- Without this, gradient-based training can converge to the explosive manifold: small Euler residuals but divergent simulated paths.

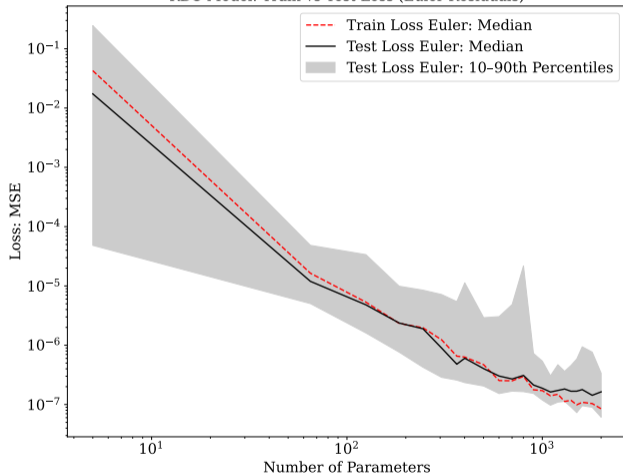
# RBC: training and test data



- Red dots:  $20 \times 20$  training grid.
- Blue crosses:  $T = 29$  path simulated under the VFI policy starting at  $k_0 = 0.5 k^*$ .
- Test path visits the interior of the state space and is never on a training node.

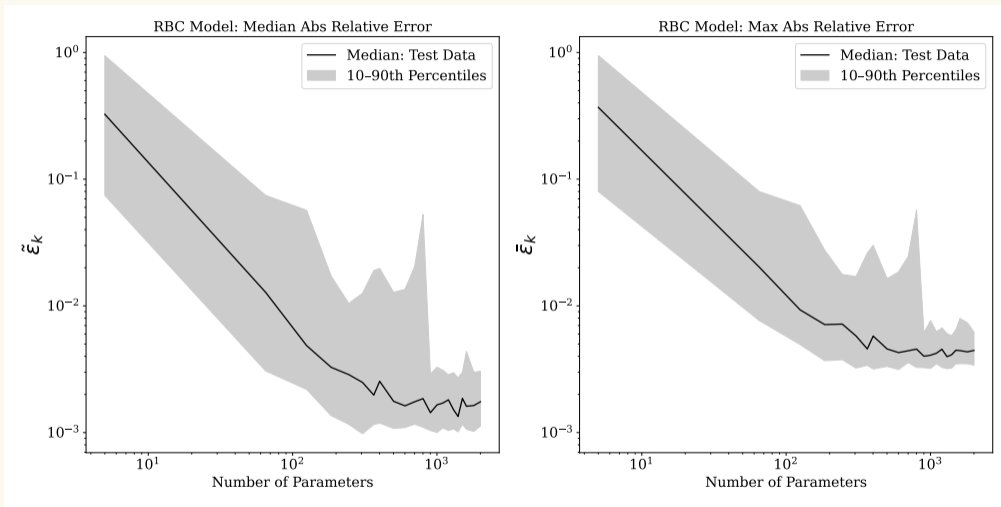
# RBC: train and test Euler-residual loss

RBC Model: Train vs Test Loss (Euler Residuals)



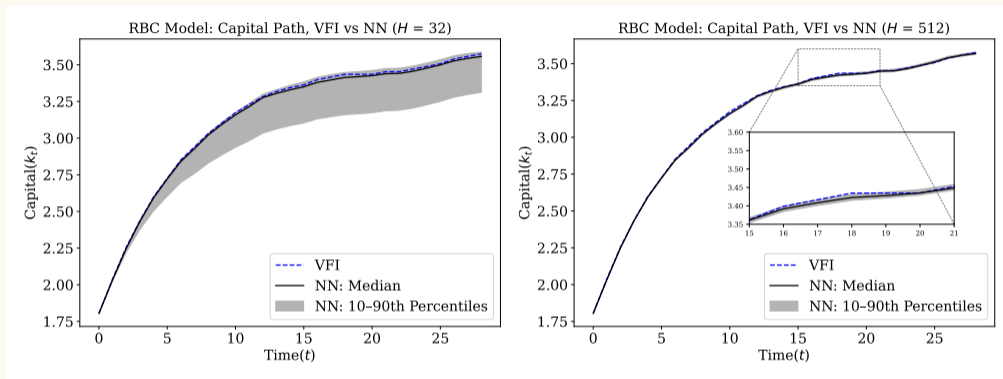
- Both losses decline as the network grows wider, with no **sign of overfitting**.
- The across-seed band narrows sharply: overparameterization simultaneously improves accuracy and reduces initialization sensitivity.

# RBC: absolute relative error vs. VFI



- Median (left) and maximum (right) absolute relative error of the NN capital policy versus VFI.

# RBC: capital paths, underparameterized vs. overparameterized



- Left:  $H = 32$ . Median path close to VFI, but across-seed band is wide. Accurate on average but unreliable.
- Right:  $H = 512$ . Median tracks VFI almost exactly over the full horizon and the band collapses to near zero.

# Robustness

---

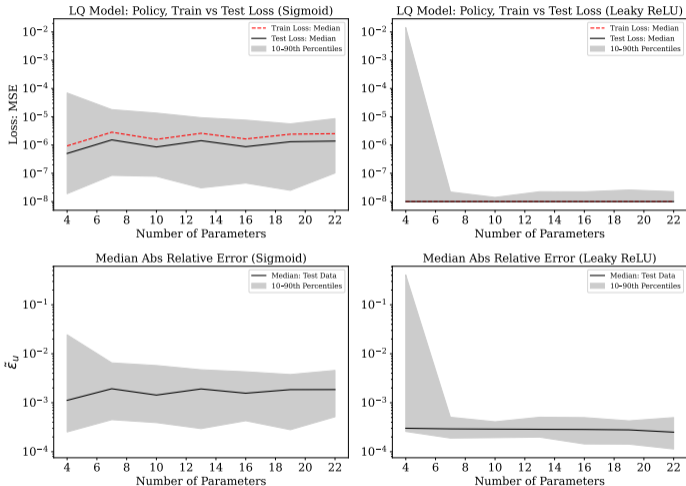
## Are these results specific to ReLU and one hidden layer?

- A natural concern: the blessing might depend on the particular activation function or the single-layer architecture.
- We test robustness along two dimensions:
  - **Activation functions:** Sigmoid and Leaky ReLU in addition to the baseline ReLU.
  - **Network depth:** two-hidden-layer architectures in addition to the baseline one-hidden-layer.

### Result

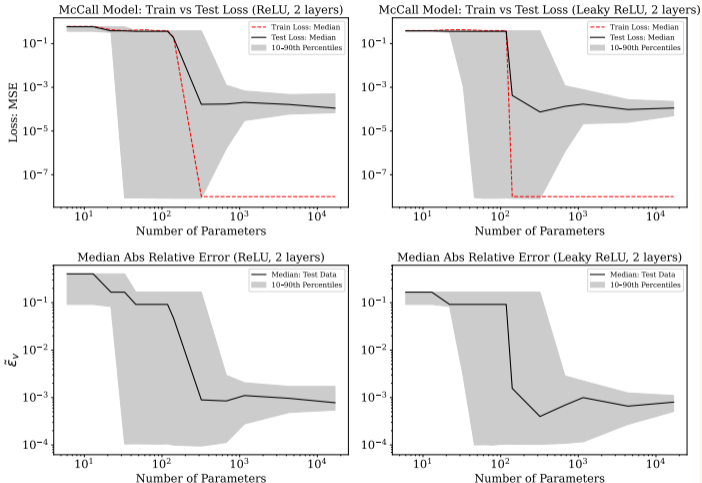
The blessing holds in every configuration we tried. The next slides document this for each model.

# Robustness: alternative activations for the LQ model



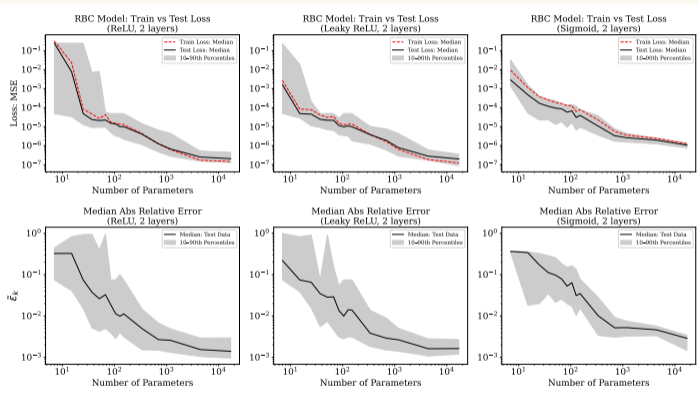
- Sigmoid (left) and Leaky ReLU (right).
- Top row: train/test Euler MSE. Bottom row: median relative error.
- Both activations replicate the main finding: accuracy improves with width, across-seed dispersion collapses.

# Robustness: deeper networks for the McCall model



- Two-hidden-layer architecture with ReLU (left) and Leaky ReLU (right).
- Quantitatively similar results: the blessing is not specific to one-hidden-layer networks.

# Robustness: deeper networks for the RBC model



- Two-hidden-layer architecture with ReLU (left), Leaky ReLU (center), Sigmoid (right).
- All three activations replicate the main finding for a stochastic model with no closed-form solution.

- The blessing of overparameterization is robust across all dimensions most relevant to applied practice:
  - **Activation functions:** ReLU, Leaky ReLU, Sigmoid.
  - **Network depth:** one and two hidden layers.
  - **Operator type:** Euler residual (LQ, RBC), Bellman residual (McCall).
  - **Solution regularity:** smooth (LQ), non-smooth with kink (McCall), stochastic without closed form (RBC).
  - **State dimensionality:** 1D (McCall), 2D (LQ, RBC).

## Conclusion

---

# Takeaways

- **Finding 1:** no overfitting. As network width increases, out-of-sample Euler and Bellman residuals decline. Value and policy functions converge toward their benchmarks.
- **Finding 2:** algorithmic stability. Small networks produce solutions that depend on the random seed. Wide networks concentrate across seeds, making the algorithm reliable and initialization-independent.
- Documented consistently across three operator equations: linear with closed form (LQ), nonlinear with kink (McCall), stochastic with no analytical solution (RBC).
- Robust across activations (ReLU, Leaky ReLU, Sigmoid) and depths (one and two hidden layers).

## Bottom line

When in doubt, use a wider network. Overparameterization is a blessing, not a curse.

# Appendix

---

## Two PDEs, one key difference

- The time-dependent **Schrödinger equation** for a single particle in one dimension:

$$i\hbar \frac{\partial \psi}{\partial t} = -\frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2} + V(x) \psi$$

Unknown:  $\psi(x, t)$ . The potential  $V(x)$  is given.

- The **Hamilton–Jacobi–Bellman equation** for the stochastic neoclassical growth model:

$$\rho V = \max_c \left\{ u(c) + V_k [e^z k^\alpha - \delta k - c] - \mu z V_z + \frac{1}{2} \sigma^2 V_{zz} \right\}$$

Unknown:  $V(k, z)$ . The drift of  $k$  depends on  $(u')^{-1}(V_k)$ , a derivative of the unknown.