Learning the Macroeconomic Language

Siddhartha Chib and Fei Tan*

December 25, 2025

Abstract

We show how state-of-the-art large language models (LLMs), seemingly inapplicable to the small samples typical of macroeconomics, can be trained effectively for macroeconomic forecasting. We estimate a dynamic stochastic general equilibrium (DSGE) model on an initial segment of the data to obtain a posterior distribution over structural parameters. We sample from this posterior to generate millions of theory-consistent synthetic panels that, when mixed with actual macroeconomic data, form the training corpus for a time-series transformer with attention. The trained model is then used to forecast out-of-sample through 2025. The results show that this hybrid forecaster, which combines the theoretical coherence of DSGE models with the representational power of modern LLMs, learns key features of the macroeconomic language.

Keywords: Bayesian methods; Deep learning; DSGE models; Economic forecasting; Large language models; Transformers.

JEL Classification: C11, C45, E37

*Chib: Olin Business School, Washington University in St. Louis; Tan: Chaifetz School of Business, Saint Louis University and Center for Economic Behavior and Decision-Making, Zhejiang University of Finance and Economics. Send correspondence to email: tanf@slu.edu (F. Tan).

1 Introduction

Macroeconomic forecasting has long confronted a fundamental challenge, that of learning reliable dynamic relationships from limited historical data. Quarterly macroeconomic time series rarely exceed a few hundred observations, a sample size that is far too small to support the direct application of modern machine learning methods. In contrast, recent advances in sequence modeling, most notably the transformer architecture of Vaswani et al. (2017), have been driven by training on billions of examples. The question we address is whether these powerful sequence learners can be adapted to the inherently small-sample environment of macroeconomics.

A recent illustration of the flexibility of transformer architectures is provided by the Starcloud-1 space station, where an NVIDIA H100 was used to train the nanoGPT model of Karpathy (2022) on Shakespeare's complete works, the first large language model (LLM) trained in orbit. This demonstration shows that transformers, once confined to massive data centers, can be deployed under extreme computational constraints through careful architectural design and training strategies. Adapting these methods to macroeconomic forecasting, however, presents a different challenge, that is the scarcity of data rather than computational capacity.

In this paper, we show how transformer models can be trained effectively for macroeconomic forecasting by combining two complementary sources of information, limited historical observations and abundant theory-consistent synthetic data that are generated from estimated dynamic stochastic general equilibrium (DSGE) models. The key insight is to use DSGE models not as direct forecasting devices, but as structured data generators that encode economic theory. In this respect, our approach is loosely related to the literature where DSGE models are used to inform priors in reduced-form time-series models [see DeJong, Ingram and Whiteman (1993), Ingram and Whiteman (1994), Del Negro and Schorfheide (2004), and Negro et al. (2007)]. Beyond this broad conceptual connection, our method is fundamentally different in its objectives, implementation, and intended application to transformer-based sequence learning.

We proceed in two steps. First, we estimate a Smets and Wouters (2007) style DSGE model augmented with stochastic volatility and Student-t shocks using Bayesian methods on an initial segment of historical data. This estimation yields a posterior distribution over structural parameters that captures uncertainty about the economy's underlying dynamics. We then draw from this posterior to generate millions of synthetic macroeconomic trajectories. Each trajectory represents a plausible counterfactual realization of the economy and incorporates empirically relevant features such as time-varying volatility, fat-tailed shocks, and nonlinear dynamics (Section 2). Second, we train a transformer model using batches that combine synthetic observations with real historical data. This mixed training strategy allows the model to absorb economic structure from DSGE-based simulations while remaining anchored to observed macroeconomic outcomes (Section 3). The synthetic trajectories expose the transformer to a wide range of economic scenarios, including regimes and crisis episodes that are sparsely represented, or entirely absent, in historical samples. In this way, the procedure directly addresses the small-sample problem that limits the applicability of modern machine learning methods in macroeconomics.

Our architectural design incorporates two adaptations that are essential for macroeconomic forecasting. The first concerns the multivariate nature of macroeconomic data. Unlike standard language models, which process a single token at a time, macroeconomic forecasting requires handling several aggregate variables that are observed jointly each period. A naive joint tokenization of all variables would lead to a combinatorial explosion in the vocabulary, rendering the approach infeasible even for modest discretizations. To address this issue, we adopt a *separate-then-concatenate* embedding strategy in which each variable is tokenized and embedded separately, and the resulting embeddings are concatenated before entering the attention layers of the model. This construction reduces the number of parameters by several orders of magnitude while still allowing the attention mechanism to learn cross-variable dependencies in a flexible manner.

The second adaptation concerns the model's output. Drawing inspiration from modular systems in robotics and autonomous driving, rather than estimating a single, monolithic transformer with a high-dimensional output layer, we employ a modular design consisting of seven specialized transformers, one for each target variable. Each transformer conditions on the same multivariate history but produces forecasts for a single macroeconomic series. This modularization avoids the curse of dimensionality in the output space and yields models that are both stable and computationally efficient. In practice, our largest specification contains on the order of 50,000 parameters and can be trained in minutes on standard hardware, while retaining sufficient capacity to capture nonlinear dynamics and cross-variable interactions.

We also depart fundamentally from conventional transfer learning paradigms in which models are pretrained on a large dataset and subsequently fine tuned on a smaller one. In the present setting, such an approach would be ineffective. With only a few hundred quarterly observations relative to the volume of synthetic data, fine-tuning would leave the pretrained representations essentially unchanged, causing the information in the real data to be largely ignored. Instead, we implement *mixed-batch training* wherein each training iteration samples from both real and synthetic data in a controlled mixture (10% real vs. 90% synthetic in our baseline). This ensures that empirical data influences learning from the outset, while synthetic data stabilizes estimation and mitigates overfitting.

The mixing proportion has a interpretation analogous to prior strength in Bayesian inference. Synthetic data encode theoretical restrictions implied by the DSGE model, while real data provide direct empirical evidence. The training process balances these two sources of information in a transparent and controllable manner. More broadly, this approach illustrates how simulation-based models and observational data can be combined within modern deep-learning frameworks, with potential applications beyond macroeconomics to any domain where structural simulators exist but real data are scarce.

Our approach connects to a growing body of work in which simulation-based training is used to overcome data scarcity. In autonomous driving, agents are trained extensively in physics-based simulators such as CARLA (Dosovitskiy et al., 2017) before deployment in real environments. In robotics, synthetic manipulation data generated by simulation engines are routinely used to train control policies, with domain randomization techniques (Tobin et al., 2017) enabling successful transfer from simulated to real-world environments. In each case, the simulator encodes domain-specific structure that allows learning to proceed efficiently despite limited real-world observations. In macroeconomics, DSGE models play an analogous role. They provide theory-consistent simulators that generate economically coherent trajectories, making it possible to train modern transformer-based models even when historical data are scarce.

We apply the method to forecast seven key U.S. macroeconomic variables over an out-of-sample period spanning nearly eight years (Section 4). The results demonstrate strong predictive performance across all variables. When prediction errors occur, they are typically small; the model selects adjacent tokens rather than making large forecasting mistakes, which suggests that the learned representations capture economically meaningful relationships. The model shows superior performance on persistent level variables (hours worked, inflation, interest rate) compared to more volatile growth rate variables (output, consumption, investment, wage), aligning with theoretical expectations that level variables exhibit stronger autocorrelation and mean reversion than period-to-period growth rates.

2 DSGE Prior

The core idea of our approach rests on using a structural DSGE model as a data augmentation mechanism. This section describes how we estimate the DSGE parameters, partition the observed data, and generate synthetic training trajectories that encode economic theory while respecting parameter uncertainty.

2.1 Model Setup

Let $y_t \in \mathbb{R}^K$ denote the vector of endogenous observables at time t, where $K = 7$ in our application. We adopt an enhanced Smets and Wouters (2007) medium-scale DSGE model with two extensions, stochastic volatility in structural shocks and Student-t distributed innovations. The generative model takes the form $y_t \sim p(y_t \mid y_{1:t-1}, \theta)$, where $p(\cdot \mid \cdot)$ is the conditional density implied by the model's equilibrium conditions, and θ encompasses structural parameters (preferences, technologies, nominal rigidities), shock persistence parameters, and stochastic volatility parameters.

The Smets-Wouters framework provides a rich structural environment featuring (i) monopolistic competition in goods and labor markets with Calvo pricing and wage rigidities; (ii) investment adjustment costs and variable capital utilization; (iii) habit formation in consumption; and (iv) seven structural shocks (technology, risk premium, investment-specific technology, government spending, price markup, wage markup, and monetary policy). Our extensions—stochastic volatility and Student-t innovations—enable the model to capture time-varying uncertainty and tail events observed in macroeconomic data, particularly during financial crises and economic recessions. These features are essential for generating realistic synthetic training data that expose the transformer to extreme economic scenarios rarely present in historical samples.

2.2 Real Data

Our quarterly U.S. macroeconomic dataset spans 1947:Q3 to 2025:Q2, comprising 312 observations of seven key variables as constructed in Smets and Wouters (2007): output growth, consumption growth, investment growth, wage growth, hours worked, inflation, and nominal interest rate. We partition the data into three non-overlapping segments.

1. **DSGE estimation sample** (50 observations, 1947:Q3–1959:Q4): Used exclusively for estimating the DSGE posterior distribution $p(\theta \mid y_{1947:1959})$. This segment provides sufficient variation to identify structural parameters while remaining completely separate from transformer training and testing. We follow the Bayesian estimation procedure detailed in Chib, Shin and Tan (2023) and obtain 10,000 posterior draws after discarding burn-in samples.

2. **Transformer training sample** (231 observations, 1960:Q1–2017:Q3): The real macroeconomic data that, when mixed with synthetic DSGE simulations, form the training corpus for the transformer. All data standardizations (means, standard deviations) are computed exclusively from this segment to prevent look-ahead bias.

3. **Out-of-sample test sample** (31 observations, 2017:Q4–2025:Q2): The final 10% of the sample spanning nearly eight years, completely held out for forecast evaluation. This segment includes major economic events (COVID-19 pandemic, inflation surge of 2021-2023) that stress-test the model's generalization capability.

This partition strategy ensures that (i) DSGE estimation and transformer training use non-overlapping historical periods, providing independent information sources; and (ii) the transformer never sees test-period data during training.

2.3 Synthetic Data Transformer training requires millions of observations, far exceeding the 231 quarters available in our training sample. We address this fundamental constraint by generating synthetic trajectories from the DSGE posterior predictive distribution. Crucially, rather than simulating at a single parameter value $\hat{\theta}$ (such as the posterior mean or mode), we implement proper Bayesian predictive sampling. For each synthetic trajectory $m = 1, \ldots, M$, we (i) draw structural parameters from the DSGE posterior, $\theta^{(m)} \sim p(\theta \mid y_{1947:1959})$; and (ii) simulate the DSGE model forward for S periods using $\theta^{(m)}$, generating trajectory $\{y_t^{(m)}\}_{t=1}^{S}$. We generate $M = 10{,}000$ synthetic trajectories, each of length $S = 1{,}000$ quarters, yielding 10 million synthetic observations. This massive synthetic corpus dwarfs the 231 real training observations, enabling transformer training that would otherwise be infeasible.

The posterior predictive sampling offers several advantages. First, it embeds parameter uncertainty directly into the training data; the transformer experiences diverse economic dynamics corresponding to different plausible parameterizations rather than committing to a single calibration. Second, it generates regime diversity. By sampling across the posterior, we create trajectories exhibiting varying degrees of nominal rigidity, shock persistence, and volatility, exposing the transformer to economic environments spanning the uncertainty reflected in the DSGE posterior. Third, the stochastic volatility and Student-t features ensure synthetic data include tail events and crises, preparing the model for forecasting during turbulent periods.

2.4 Hierarchical Bayesian Interpretation Our procedure admits a clean hierarchical Bayesian interpretation. The DSGE estimation yields a posterior distribution $p(\theta \mid y_{1947:1959})$ that encodes structural beliefs about the economy. The synthetic trajectories drawn from this posterior constitute the prior predictive distribution for future macroeconomic outcomes. The transformer serves as a likelihood function that performs Bayesian updating. It starts with the DSGE prior and refines predictions using the empirical signal from real observations. The mixing ratio between synthetic and real data acts as a hyperparameter controlling the prior-to-likelihood weight, analogous to the effective sample size in Bayesian inference.

This framework contrasts with traditional DSGE forecasting, where predictions are generated via the Kalman filter. Instead, we use the DSGE model to generate a training corpus, then let the transformer learn a flexible forecasting function that need not respect the DSGE's linearized structure. The transformer can discover nonlinear patterns, time-varying relationships, and cross-variable dependencies that the DSGE's equilibrium conditions might misspecify, while still benefiting from the economic structure encoded in synthetic data.

Computationally, DSGE simulation is highly efficient once the posterior is obtained. Generating each 1,000-quarter trajectory requires milliseconds on modern hardware. The synthetic data generation completes in minutes, after which we have a reusable training corpus that can be combined with real data under different mixing ratios. This computational efficiency makes the approach practical for iterative experimentation and hyperparameter tuning.

3 Transformer Likelihood

Transformers learn to forecast by treating time series as a kind of language. Just as LLMs predict the next word given preceding text, our model predicts next period's economic conditions given the most recent history. The analogy runs deeper. Words map to discrete tokens, economic variables map to discrete bins; sentence structure captures semantic relationships, temporal patterns capture dynamic relationships. This section explains how we adapt the transformer architecture to macroeconomic forecasting, translating continuous time series into a learnable vocabulary while respecting the multivariate structure of economic data.

From an econometric perspective, the transformer can be interpreted as a nonlinear function estimating dynamic conditional expectations $\mathbb{E}[y_{t+1} \mid y_{1:t}]$ through stacked attention and feed-forward layers. Each layer expands the functional space over which expectations are formed, analogous to progressively richer lag polynomials or state-transition expansions in classical time-series models. The architecture is thus a high-dimensional analogue of recursive forecasting models, but with data-driven lag selection via attention.

3.1 Tokenization Strategy

Continuous macroeconomic time series present fundamental challenges to direct application of transformer models, which traditionally operate on discrete tokens from finite vocabularies. Standard approaches like uniform binning or simple discretization fail to capture the complex, time-varying distributions characteristic of economic data. We address this through a percentile-based tokenization scheme that preserves distributional information while enabling efficient sequence modeling.

Beyond enabling transformer architectures, tokenization serves as a principled noise reduction mechanism, which is of critical importance in noise-heavy social science applications. In practice, one does not require the precision implied by continuous measurements. A central bank deciding on interest rates cares whether inflation is broadly low, moderate, or high, not whether it is precisely 2.3% versus 2.35%. Our tokenization formalizes this intuition by discretizing continuous observations into economically meaningful bins, effectively filtering out noise while retaining the signal relevant for forecasting and decision-making.

We implement the variable-specific tokenization as follows. For each economic variable $k \in \{1, \ldots, K\}$, we compute data-driven bin boundaries from the training data (both real and synthetic) using empirical percentiles $\{p_0, p_{10}, \ldots, p_{90}, p_{100}\}$, which define $J = 10$ bins in our baseline specification. Each continuous observation $y_t^{(k)}$ is then mapped to a discrete bin index $x_t^{(k)}$. This yields tokens $x_t^{(k)} \in \{0, 1, \ldots, J-1\}$ for each variable, where token 0 represents values in the bottom percentile and token $J-1$ represents the top percentile. This percentile-based discretization shares conceptual similarities with approaches in financial time series modeling such as Shi et al. (2025), though our multivariate embedding strategy discussed below differs from the univariate setting.

This approach has several advantages over uniform binning. First, it ensures balanced token frequencies, as each bin contains approximately 10% of observations by construction. Second, it automatically handles different scales and distributions across variables. Inflation rates and output gaps, for example, have vastly different ranges. Third, it preserves information about tail events, which are particularly important for macroeconomic forecasting during crisis periods. Lastly, the coarse discretization (ten bins) acts as a regularizer, preventing the model from overfitting to measurement errors or spurious precision in the data, which is particularly valuable given the small sample sizes typical in social science datasets. Our tokenization ensures that tokens carry meaningful economic information about relative magnitudes and tail events while suppressing irrelevant high-frequency variation.

3.2 Model Architecture

Our architectural design incorporates two key adaptations for macroeconomic forecasting. First, we train seven specialized transformers rather than one monolithic model, inspired by modular systems in autonomous driving where separate networks handle distinct tasks (e.g., Tesla's Full Self-Driving uses over 100 specialized transformers). Each transformer sees the latest multivariate history but predicts just one target variable in the next period. This design avoids the curse of dimensionality. Jointly predicting seven variables would require modeling the combinatorial space of all possible outcome combinations.

The second architectural adaptation addresses multivariate embedding. Transformers operate on continuous vector representations rather than discrete tokens directly. The embedding table converts each discrete token into a continuous vector in high-dimensional space, allowing the model to learn relationships and patterns through numerical operations. This is analogous to how economists use vector autoregressions (VARs) to capture multivariate dynamics. However, naively creating one embedding table for all possible token combinations across seven variables would require $10^7 = 10$ million entries, which is difficult to estimate even with ample synthetic data. Instead, we use separate embedding tables for each variable (70 embeddings total). Specifically, each token $x_t^{(k)}$ is mapped to a dense embedding vector $e_t^{(k)} \in \mathbb{R}^d$, where $d = 8$ is the embedding dimension in our baseline specification. These variable-specific embeddings are then concatenated to form the multivariate state representation $e_t = [e_t^{(1)}, \ldots, e_t^{(K)}] \in \mathbb{R}^E$ (Figure 1). This strategy reduces the number of parameters by several orders of magnitude while respecting that token "5" means different things for inflation versus unemployment.

Our compact transformer processes sequences of length $T = 4$ (analogous to four lags in

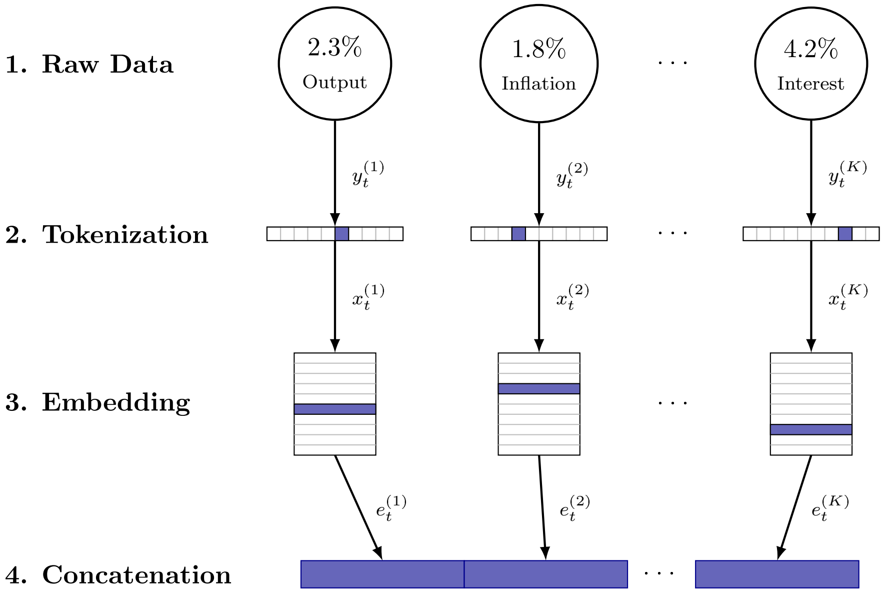

Figure 1: From continuous data to concatenated embeddings. The process transforms raw economic indicators (top) into a unified vector representation (bottom) suitable for the transformer. Each variable is independently discretized into tokens, mapped to a dense vector via its own embedding table, and finally concatenated to form the state representation.

quarterly VARs) of concatenated embeddings $\{e_{t-T+1}, \ldots, e_t\}$ to predict the next-period token for the target variable. The architecture uses $L = 2$ transformer layers with $H = 2$ attention heads per layer—totaling approximately 52,000 parameters (Table 1). The attention mechanism learns to weight past states adaptively. During recessions, the model attends to previous crisis episodes and leading indicators like investment; during expansions, attention focuses on recent dynamics. This resembles how economists perform historical decompositions in DSGE models, but the attention weights are learned from data rather than imposed by theoretical restrictions.

We optimize using cross-entropy loss on next-token prediction, the discrete analogue of maximizing one-step-ahead likelihood. This training objective directly parallels maximum likelihood estimation in standard econometrics. Just as ordered probit or logit models estimate conditional probabilities for discrete outcomes, our transformer learns the probability distribution over the $J = 10$ ordered tokens for the target variable. The key difference lies in the functional form. Rather than assuming parametric link functions (probit, logit) with linear indices, the transformer uses stacked attention and deep feedforward layers to approximate the conditional distribution nonparametrically. This flexibility allows the model to capture complex, high-dimensional dependencies that would be difficult to specify in traditional discrete choice frameworks.

3.3 Mixed Training

How should we combine 231 historical observations with millions of synthetic trajectories when training the transformer? The synthetic data vastly outnumber real

Table 1: Model Configuration

Hyperparameter	**Notation**	**Value**
Embedding dimension	E	56
Number of layers	L	2
Number of attention heads	H	2
Sequence length (quarters)	T	4
Number of variables	K	7
Tokens per variable	J	10
Batch size	B	256
Real data mixing ratio	α	0.1
Total parameters		**52,538**

observations, yet both contain essential information. Simulations encode theoretical structure while historical data capture actual patterns.

Standard transfer learning strategies prove inadequate in this asymmetric data regime. The conventional approach of pretraining on abundant data, then fine-tuning on limited samples, assumes that the abundant data provide general representations that require only minor adjustments. But with overwhelming synthetic abundance versus scarce real data, fine-tuning would barely shift pretrained representations. The model effectively memorizes DSGE dynamics while ignoring empirical signals, analogous to an overconfident prior that resists updating despite empirical evidence.

We instead implement *mixed-batch training*, a protocol that treats theory and data as complements rather than substitutes. Each training iteration constructs mini-batches of size $B = 256$ by randomly sampling observations from both sources: 10% from real data and 90% from synthetic DSGE simulations in our baseline specification. This ensures real observations continuously influence learning from the first iteration, while synthetic data provide stable gradients, prevent overfitting to the small historical sample, and expose the model to rare events (crises, regime changes, tail realizations) underrepresented in 77 years of the U.S. quarterly data.

The mixing ratio $\alpha \in [0, 1]$ serves as an interpretable hyperparameter controlling the theory-data balance. From a Bayesian perspective, α functions analogously to prior precision or effective sample size. It determines the relative weight assigned to structural beliefs encoded in DSGE simulations versus empirical evidence in historical observations. At $\alpha = 0$, the model learns pure DSGE dynamics, effectively imposing the structural model as a dogmatic prior. At $\alpha = 1$, the transformer sees only real data and risks overfitting, similar to flat-prior estimation with minimal regularization. Our baseline $\alpha = 0.1$ strikes a middle ground. The DSGE model provides a theory-informed prior through synthetic trajectories, while historical observations serve as the likelihood that updates beliefs.

Computationally, the approach is remarkably efficient. Training completes in minutes on standard hardware with parallel processing (MacOS with Apple M2 Silicon chip), enabling rapid experimentation with different transformer architectures and DSGE specifications. This accessibility contrasts sharply with LLM training, which typically requires industrial-scale computational resources. The efficiency stems from our compact architecture (approximately 50,000 parameters) and the highly structured nature of macroeconomic data, where theoretical constraints dramatically reduce the effective dimensionality of the learning problem.

4 Posterior Results

Figure 2 displays three complementary views of the model's forecasting performance. The grayscale heatmap shows the full predictive token distribution at each time step for each variable. This probabilistic output distinguishes our approach from traditional point forecasting methods. The model naturally quantifies its uncertainty by assigning probabilities across the entire support of possible outcomes. During periods of high confidence, the distribution concentrates sharply on a narrow range of tokens (appearing as dark vertical bands), while during uncertain periods, the probability mass spreads more evenly across multiple tokens. The blue line with circles represents the actual observed values, mapped to their corresponding positions on the token scale. Horizontal bars indicate the model's point predictions, green bars denote correct predictions where the predicted token matches the actual token, while red bars indicate prediction errors. The vertical axis shows the percentile bin boundaries estimated from the training data.

Several patterns emerge from the forecasting results. First, the model performs better on persistent level variables (hours worked, inflation, interest rate) than on volatile growth rate variables (output, consumption, investment). This matches with theoretical expectations. Level variables exhibit stronger autocorrelation and mean reversion, providing more predictable dynamics. Growth rates, by contrast, reflect period-to-period changes that are inherently noisier and harder to forecast. The transformer appears to internalize this hierarchy of persistence, similar to the eigenvalue structure embedded in linearized DSGE models.

Second, rather than making large forecasting mistakes that would indicate random guessing or complete model failure, errors typically involve confusing neighboring states, for instance, predicting moderate inflation when actual inflation is slightly high. This ordinal structure suggests the model has learned economically sensible rankings of macroeconomic conditions, even when precise point predictions deviate from realized values. From a policy perspective, this pattern is encouraging since decision-makers care more about correctly identifying the broad state of the economy (such as expansion versus recession or low versus high inflation) than about predicting

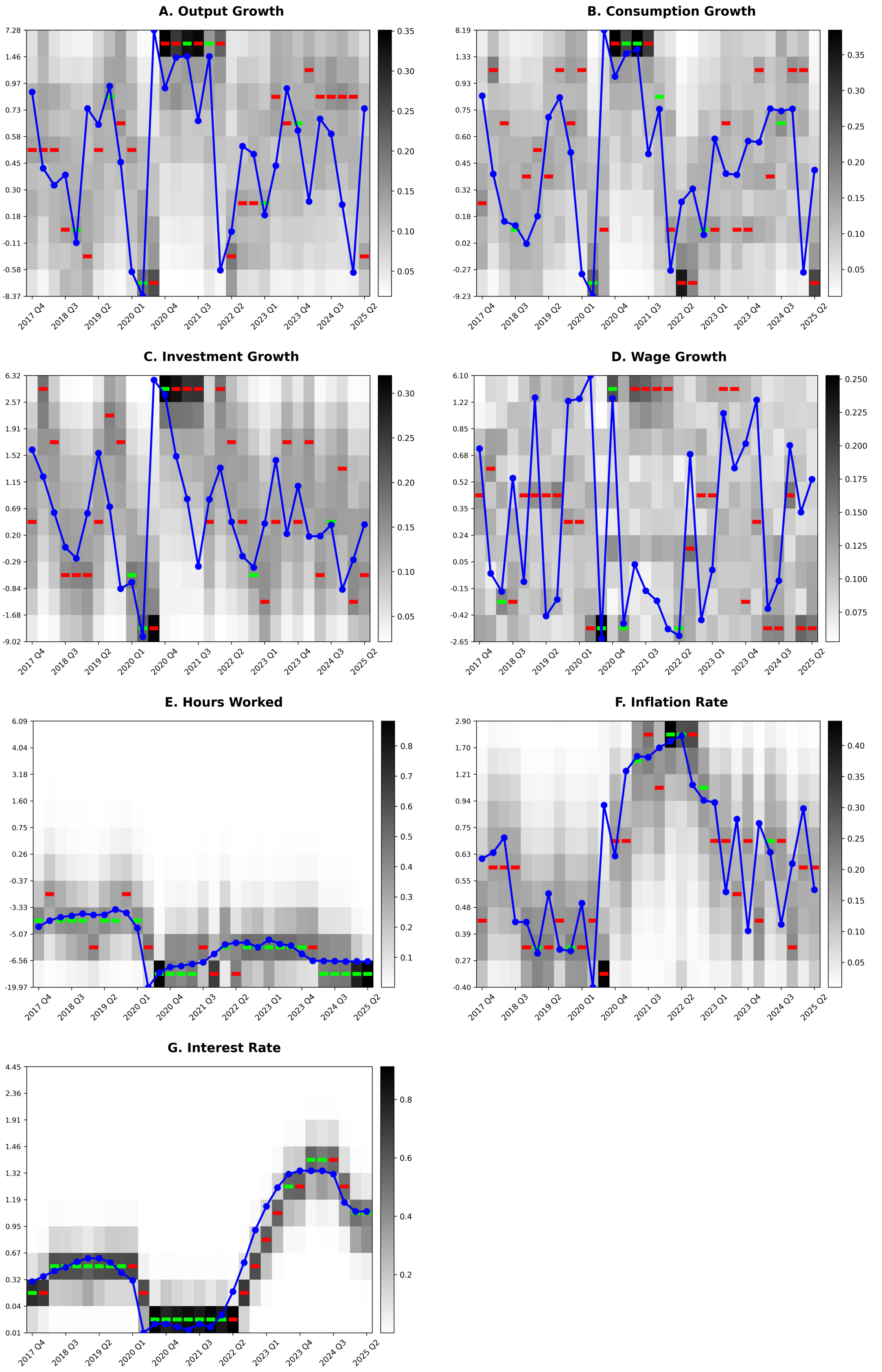

Figure 2: Out-of-sample predictions for seven macroeconomic variables, 2017:Q4–2025:Q2.

values with spurious precision.

These results validate our mixed training approach. The synthetic DSGE simulations provide abundant training signal about economic co-movements and dynamic relationships, including crisis scenarios and tail events rarely observed in historical data, while the real observations anchor predictions to actual patterns. The 10% vs. 90% mixture proves effective. The model generalizes well to out-of-sample periods, suggesting it has learned structured economic relationships from simulations that transfer to real forecasting tasks.

Our findings carry several implications for economic forecasting practice. First, transformer architectures, appropriately adapted, can deliver strong predictive performance despite small samples. The key innovation is treating DSGE models not as competing forecasting methods but as complementary data augmentation mechanisms that encode economic theory.

Second, the strong performance with compact architecture (two network layers, 56-dimensional embeddings, approximately 50,000 parameters) suggests macroeconomic forecasting may not require the massive models common in transformer models. Economic data are highly structured with well-understood theoretical constraints. This structure allows smaller models to learn effectively, particularly when training incorporates theory-consistent simulations. The computational accessibility facilitates reproducibility and broader adoption in academic and policy settings.

Third, the framework admits natural extensions. Multi-horizon forecasting could be implemented by training separate models for different forecast horizons or by extending the architecture to predict multiple steps ahead. Scenario analysis under alternative policy rules could leverage the DSGE simulator to generate counterfactual training data. Ensemble approaches combining multiple DSGE specifications could provide robustness to model misspecification. These extensions could yield general-purpose “macro-foundation models”, compact, theory-aligned forecasting systems that serve as economic counterparts to language-foundation architectures.

Finally, the approach enables novel applications beyond pure forecasting. The framework can be interpreted as an *amortized likelihood learner*. Once trained, the transformer approximates $p(y_{t+1} \mid y_{1:t}, \theta)$ for draws from the DSGE parameter posterior, effectively learning a fast surrogate for the computationally expensive DSGE likelihood. Future work could embed this within a full Bayesian workflow, using the transformer as a differentiable likelihood approximation for simulation-based inference, merging DSGE estimation and deep forecasting in a unified framework.

5 Conclusion

We show that a transformer can learn important features of the macroeconomic language when trained using theory-consistent simulation. By tokenizing economic variables and training on mixed batches of DSGE-generated synthetic data and real observations, we develop a forecaster that combines structural discipline with predictive flexibility. We view our contribution as methodological and therefore refrain from direct comparisons with VAR or DSGE forecasts. Instead, we demonstrate a new approach to training time-series transformers for macroeconomic forecasting that combines theory-consistent synthetic data from DSGE models with limited empirical observations. The same strategy applies more broadly to economic settings where structural models are available but data are scarce.

Our approach also speaks to a broader limitation emerging in contemporary large language model development. Scaling laws appear to be reaching diminishing returns despite massive increases in parameters and training data. In contrast, humans learn efficiently by integrating new information with prior structure in a manner resembling Bayesian updating. This suggests that progress may depend less on brute-force scaling and more on domain-specific adaptations. Just as autonomous vehicles and robotic systems learn from physics-based simulators, time series transformers can learn from DSGE-generated synthetic trajectories, opening the door to analogous applications of theory generated synthetic data in other disciplines.

Appendix

We provide open-source Python implementation of our forecasting framework, adapting the codebase from Karpathy (2022), to facilitate replication and extension by researchers. The codebase comprises three modular components designed for clarity and flexibility: `model.py`, `train.py`, and `config.py`. This architecture separates model specification, training logic, and hyperparameter configuration, enabling rapid experimentation without modifying the core functionality.

- **Model architecture (`model.py`).** This module implements the transformer architecture using PyTorch, a popular deep learning framework widely adopted in machine learning research. The implementation includes: (i) variable-specific embedding tables that map discrete tokens to continuous vectors; (ii) multi-head self-attention layers with causal masking to prevent information leakage from future time steps; (iii) feedforward networks with activation functions and layer normalization; and (iv) an output head producing probability distributions over tokens via softmax activation. The modular design allows researchers to experiment with architectural variations by modifying isolated components rather than rewriting the entire model.

- **Training pipeline (`train.py`).** This module orchestrates the mixed-batch training protocol. Core functionality includes: (i) data loading routines that read real and synthetic datasets, apply standardization transformations, and implement the percentile-based tokenization scheme; (ii) mixed-batch sampling that constructs training batches by drawing $\alpha \cdot B$ observations from real data and $(1-\alpha) \cdot B$ from synthetic simulations; (iii) optimization loop using Adam optimizer with cross-entropy loss, computing gradients and updating model parameters; and (iv) evaluation routines that compute forecast accuracy metrics on held-out test data and save trained model checkpoints. The implementation handles edge cases such as small real data samples through careful batch construction and supports early stopping based on validation performance.

- **Configuration (`config.py`).** This module centralizes all hyperparameters and data paths in a single location. Key settings include: (i) architectural parameters (E, L, H, T) controlling model capacity; (ii) training parameters (batch size B, mixing ratio α, learning rate, number of epochs); (iii) tokenization settings (number of bins J, percentile computation method); and (iv) file paths for real and synthetic datasets. Researchers can modify these settings without touching the model or training code, which facilitates systematic hyperparameter searches and sensitivity analysis.

Practitioners can apply the framework to their forecasting problems by preparing two datasets: (i) real data in CSV format with columns for each variable and rows for time periods, and

(ii) synthetic data from estimated structural models in the same format. The training script automatically handles data preprocessing, standardization, and tokenization. After training, the saved model can generate predictive token distributions for each target variable. The complete implementation, including detailed documentation, usage examples, and the exact configurations used in this paper, is available upon request.

References

Chib, S., M. Shin, and F. Tan (2023): "DSGE-SVt: An Econometric Toolkit for High-Dimensional DSGE Models with SV and t Errors," *Computational Economics*, 61(1), 69–111.

DeJong, D., B. Ingram, and C. Whiteman (1993): "Analyzing VARs with Monetary Business Cycle Model Priors," in *Proceedings of the American Statistical Association, Bayesian Statistical Science Section*, pp. 160–169. American Statistical Association.

Del Negro, M., and F. Schorfheide (2004): "Priors from General Equilibrium Models for VARS," *International Economic Review*, 45(2), 643–673.

Dosovitskiy, A., G. Ros, F. Codevilla, A. Lopez, and V. Koltun (2017): "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*.

Ingram, B. F., and C. H. Whiteman (1994): "Supplanting the Minnesota Prior: Forecasting Macroeconomic Time Series Using Real Business Cycle Model Priors," *Journal of Monetary Economics*, 34(3), 497–510.

Karpathy, A. (2022): "nanoGPT: The simplest, fastest repository for training/finetuning medium-sized GPTs," GitHub repository.

Negro, M. D., F. Schorfheide, F. Smets, and R. Wouters (2007): "On the Fit of New Keynesian Models," *Journal of Business & Economic Statistics*, 25(2), 123–143.

Shi, Y., Z. Fu, S. Chen, et al. (2025): "Kronos: A Foundation Model for the Language of Financial Markets," arXiv:2508.02739.

Smets, F., and R. Wouters (2007): "Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach," *American Economic Review*, 97(3), 586–606.

Tobin, J., R. Fong, A. Ray, et al. (2017): "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30.

Vaswani, A., N. Shazeer, N. Parmar, et al. (2017): “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, ed. by I. Guyon, U. V. Luxburg, S. Bengio, et al., vol. 30. Curran Associates, Inc.