Interpretability of Financial Machine Learning Models

Insights into Model Transparency, Explainability, and Trustworthiness

Alejandro de la Borbolla Monge*, Paul O'Connor*, Prapti Jaydeep Kumar*, Yash Gupta*
aborbo@mit.edu, paulroc@mit.edu, prapti@mit.edu, ygupta@mit.edu
*Massachusetts Institute of Technology, Sloan School of Management

I. Introduction

With the recent advances in machine learning (ML) techniques, improvements in computing speeds, and proliferation of structured and unstructured data, it has become imperative for modern financial institutions to adopt big data technologies and state-of-the-art ML techniques to secure their place at the forefront of the industry. However, the use of these cutting-edge models pose some common challenges: (1) these models typically have a huge number (thousands to millions) of trainable parameters requiring massive amounts of data (or, else are very susceptible to overfit to noise), (2) they typically require considerable inference time, rendering them nonviable for latency-sensitive applications (such as high-frequency market making), and (3) they often are not as transparent as traditional econometric models.

In fact, while deep learning has paved the way for dramatically speeding advances in big data analytics and modeling as well as towards artificial general intelligence (AGI), it has also aggravated concerns around black-box models in the financial industry. Banks are subject to scrutiny by the regulators, who have raised concerns about the lack of explainability and reliability of machine learning models (Alexey Surkov, 2022). Hedge funds and proprietary trading firms similarly find themselves faced with the problem of reconciling highest standards of risk management with lack of feature attribution and interpretability of complex ML approaches (Laurens Swinkels, 2022). Asset and wealth managers need explainability to maintain high levels of investor trust with appropriate economic reasoning. Common questions asked about interpretability of ML models, as seen in the sections that follow, are centered around its reliability (trustworthiness of the learned mappings and the ability to remain robust with changing conditions), explainability (ease of understanding the effect of the different model inputs), and transparency (ability to look into the model mechanics).

The recent computer science literature has connected complex NLP and image models with suitable interpretability techniques (Jacovi and Goldberg, 2020), (Boychev, 2023). However, time series, in particular financial time series, remain a relatively less explored area. Interpretability of financial time series is nuanced by their relative scarcity of data points (such as due to lower reporting frequency of macro variables) and the need to connect with the underlying economic intuition and theoretical equations. The study aims to bridge this gap by examining the applicability of different models from the lens of interpretability (reliability and explainability) for financial time series forecasting through a specific, well-studied instance of this class of problems: volatility forecasting. We analyze models lying on a joint spectrum of transparency and complexity, including the ordinary least squares regression, relevance-based prediction (RBP), gated recurrent units (GRU), and GRU augmented with transfer learning. The models are scrutinized on their predictive accuracy under real and simulated scenarios and on the ease of understanding how different predictions are arrived at and influence by different features .

The goal of this work is threefold. First, it evaluates whether there is a tradeoff between interpretability and predictive accuracy. Second, it examines interpretability tools for inherently interpretable models such as relevance based prediction, and post-hoc interpretability tools for black-box type of models such as GRU networks. Finally, we evaluate, through a simulation analysis, under which circumstances inherently interpretable models exhibit a better performance.

Section II provides a detailed review of the notions of and techniques for interpretability in general as well as finance-focused ML literature, spanning ideas including approximation-based methods, attention mechanisms, hybrid neural networks and econometric models, transfer learning, and interrogation. Section III formally introduces the volatility forecasting problem and the data used for the task along with our lineup of candidate models. Section IV reports the predictive performance of the models on real data for the S&P 500 index. Section V discusses how one may leverage inherently interpretable structure of certain models like RBP or apply post-hoc methods to the black-box models like GRUs to get insights into the model inference and feature sensitivities. Finally, Section VI provides a comprehensive picture of the performance of different models under various linear and non-linear simulated data. Section VII concludes the report by noting the key insights from and limitations of the study as well as avenues for further research.

II. Literature Review

A. Notion of Interpretability

Underlying the notion of interpretability are the ideas of trust, explainability, and transparency, all crucial to a deployable financial model. The researchers and users should be able to trust the predictions of an interpretable model; that is, they must know that the predictions are not based on noise but rather the understanding of the underlying economic rule. They should be able to inspect how the model predictions

are generated, similar to how one can inspect the coefficients of an OLS model.

(Goel and Weber, 2025) outlines three aspects pivotal to the notion of interpretability: robustness, counterfactual quality, and model complexity. Robust decision boundaries, those which require substantial changes to the features on examples for misclassification, are expected to be more interpretable than their less robust counterparts. The higher the quality of counterfactuals, based on probability of factual versus the counterfactual, the more interpretable a model is. Finally, complexity and interpretability are generally regarded as inversely related.

More formally, (Schmidt and Biessmann, 2019) proposes a metric to measure interpretability based on the increase in information transfer rate in an annotation task when model explanations are provided. The authors argue that annotators should be able to reproduce model decisions faster and more accurately when the quality of an interpretability technique is higher. Specifically, they define the information transfer rate, in bits per second, as the ratio of the mutual information between annotations provided by the human labelers and the predictions to the average response time in the annotation task.

B. Interpretability Models

On a very broad level, there are two primary approaches to interpretability, as outlined in (Zhao et al., 2023):

- **Post-hoc Interpretation Methods**: These methods explain the model after the model inference and are generally decoupled from the model building and training process. These are suitable for adding explainability to black-box models after the fact, but require additional computing resources and the user can access the explanations until the inference is completed.
- **Inherently Interpretable Methods**: Another way of making a model explainable and trustworthy is to design it to be robust and transparent. These models are often less complex or use inherently interpretable representations or weights. A trivial example would be OLS regression, wherein the coefficients reflect the reliance of the predictions on the different features.

Some popular examples of post-hoc methods include:

- *Gradient-based methods*: This approach uses backpropagation, involving the computation of partial derivatives, to calculate the relevant information that explains a black-box model's predictions. The gradients or partial derivatives provide an intuitive way to analyze the model's sensitivity to different input features. A specific example would be Class Activation Map which uses gradients in CNNs to identify the most important regions in the input time series in a forecasting problem (Oviedo et al., 2019).
- *Perturbation-based Methods*: These involve estimating the feature importances by erasing, masking, or changing the input and observing the change in the model predictions. A common example is feature ablation, wherein $k \geq 1$ features are removed at a time to identify the set that most effectively affects the model prediction. Other techniques include permuting the input vector of features or erasing parts of the network.
- *Approximation-based Methods*: Local Interpretable Model-agnostic Explanations (LIME) and Shapley Additive Explanations (SHAP) are two primary examples of this class of techniques. These approximate the feature attribution using local linear approximations. LIME (Ribeiro et al., 2016) uses interpretable local surrogate models at any specific point to quantify impact of variations to the input for explaining the individual predictions of a black-box model. SHAP (Lundberg and Lee, 2017), drawing from game theory, estimates feature importance by comparing its effect on model prediction when it is present versus absent, across all possible coalitions of other features.
- (Li et al., 2019) develop an interpretable machine-learning framework that decomposes any predictive model into four economically meaningful components: linear effects, nonlinear effects, pairwise interactions, and higher-order interactions using partial-dependence "model fingerprints".

C. Relevance-Based Prediction

Relevance-Based Prediction (RBP) is a model-free prediction approach designed to provide a transparent alternative to opaque machine learning techniques (Czasonis et al., 2024). The RBP process begins by constructing a prediction grid in which columns represent subsets of predictive variables and rows represent different relevance-censoring thresholds applied to historical observations. This censoring introduces nonlinearity into the prediction process. Each cell in the grid produces a prediction based on a relevance-weighted average of historical outcomes, and the final prediction is computed as a reliability-weighted average across all cells. (Czasonis et al., 2024) show that RBP offers comparable to better predictive performance when compared to Neural Networks for a volatility prediction task.

D. Gated Recurrent Units and Hybrid Models

Introduced in (Cho et al., 2014), the gated recurrent unit (GRU) networks represent a significant milestone in deep time series modeling and neural machine translation: they address the vanishing gradient problem that plagues the standard recurrent neural networks (RNNs) while still offering a more computational efficient alternative to the long short-term memory (LSTM) networks. The GRU is defined by its gating mechanism, which regulates the flow of information inside the unit. Unlike standard RNNs which overwrite their hidden state at every step, GRUs learn when to update and when to reset their memory. The update gate decides how much of the past information (from previous time step) needs to be passed along to the future, while the reset gate decides how much of the past information to forget. Unlike LSTMs, GRU does not maintain a separate cell state and hidden state but merges them into a single hidden state, streamlining the architecture.

GRUs have found widespread application in time series forecasting problems, including those in the financial domain (for example, emerging stock market index prediction (Alsheebah and Al-Fuhaidi, 2024)). However, like other deep black-box models, they are not transparent and are susceptible to overfit, diminishing their utility to critical financial and economic application. The present paper presents some interpretability techniques to unravel the influence of different features on GRU predictions. Alternatively, economic constraints or properties may be tied to the model architecture through hybrid models such as the GARCH-GRU.(Wei et al., 2025) advance the hybrid volatility-forecasting literature by proposing an integrated GARCH-GRU architecture that embeds GARCH(1,1) dynamics directly within the internal computations of a GRU cell. Unlike earlier hybrid models that treat GARCH and neural networks as separate components, either by feeding GARCH residuals into an RNN pipeline or lightly modifying LSTM gates, this new framework creates a fully unified recurrent unit that jointly learns econometric volatility structure and nonlinear temporal dependencies. The model incorporates the conditional variance equation into a dedicated gating mechanism that modulates the GRU's hidden state updates. This allows the network to internalize volatility clustering, persistence, and other stylized facts while retaining the GRU's efficiency and reduced parameterization relative to LSTM-based hybrids. These models are highlighted to outperform the lone GRU, LSTM, Transformer, GARCH(1,1) and GJR-GARCH models on volatility prediction for S&P 500 and NASDAQ on an error (MAE/MSE) basis. Similarly hybrid models for LSTMs ((Kim and Won, 2018)) have yielded successful results in the past. The authors propose a framework that integrates multiple GARCH-type models with a Long Short-Term Memory (LSTM) network to improve stock-index volatility forecasting. Their hybrid approach first uses several traditional GARCH variants—specifically GARCH, EGARCH, TGARCH, and GJR-GARCH—to generate baseline volatility estimates that capture stylized facts such as persistence/clustering, leverage effects, and asymmetry. These GARCH-based volatility series are then used as inputs to an LSTM model, enabling the network to learn nonlinear temporal dependencies and long-range structures that conventional GARCH models cannot fully represent. The improved performance is attributed to the incorporation of economic facts about financial volatility, such as clustering and fat tails of returns, through the GARCH component while nonlinear temporal dependencies are learned by the conventional GRU channel. They report that the combined GARCH-LSTM model yields superior out-of-sample forecasting accuracy compared to individual GARCH models as well as standalone LSTM benchmarks.

E. Transfer Learning

Incorporating physics- or economics-based rules into the model architecture - as seen with GARCH-GRU - is one way to make the model robust to noise and more interpretable. Another way to retain the flexibility of a complex black-box model while still helping it generalize reliably beyond the confines of the training data is to use transfer learning, wherein a model trained on one task is reused as the basis for a second, related task, leveraging the pre-trained knowledge instead of starting from scratch.

The neural network can be initialized by training on synthetic data generated by a structural economic model and then fine-tuned to the real empirical data. (Chen et al., 2025) proposes this framework in the context of option pricing to incorporate financial restrictions while learning the non-linear dependencies from the historical data. The transfer learning approach yields superior performance than both the conventional deep neural network (DNN) as well as the structural Heston model. Furthermore, the outperformance is significant when:

- the empirical dataset is small so that the conventional DNN is much more susceptible to overfit,
- market conditions change relative to the in-sample, making the fundamental underlying financial principles more important than the in-sample datapoints, and
- the structural model is not very misspecified, as is the case with the Black-Scholes model which (Chen et al., 2025) uses.

In the context of financial time series forecasting, transfer learning approaches may be used to boost the generalizability of the model - and hence its trustworthiness or reliability and, in turn, its interpretability - if a suitable structural and correctly specified model is identified to generate the initial synthetic data. This presents another avenue, in addition to GARCH-GRU to provide interpretability to a black-box model like GRU.

(Song et al., 2025) extend the model-fingerprint logic and use it not for economic interpretation, but for model diagnostics, proposing "interrogation" as a universal, model-agnostic test for underfitting and overfitting. Rather than assessing performance through cross-validation, which provides little insight into why a model underfits or overfits, interrogation decomposes a prediction function into the same four interpretable components. Their work highlights that interpretability need not be sacrificed when adopting sophisticated ML models; instead, systematically decomposing predictive components can provide transparent insight into a model's behavior, its economic intuition, and the sources of its performance. We use this technique to assess interpretability in the present paper.

III. Methodology: Models Studied

A. Forecasting Problem, Features, and Models

We will assess the ability of different models to forecast stock return volatility. Volatility forecasting is a widely studied problem in finance. Unlike stock or index returns which constitute a sequence of martingale differences (and hence remain impossible to model per EMH), volatility has been modeled with several classical and empirical models, including GARCH(p, q), GARCH variants like AGARCH and IGARCH, LSTMs, and transformers. Furthermore, volatility prediction

has profound implications for hedging, volatility trading, and investment strategies. The temporal nature of the problem, the applicability of several classes of models, and the high financial interest make volatility prediction an ideal problem to study from an interpretability standpoint.

We calculate volatility as the standard deviation of the S&P 500 daily returns over a one-month period. Table I shows the variables used as predictors and the sources . These variables are similar to the ones used by (Czasonis et al., 2024), but we dropped some of the features used in that study, such as implied volatility, to extend the sample back to 1980.

Variable	Source
Short-term interest rate level	FRED
Short-term interest rate change	FRED
Long-term interest rate change	FRED
Credit spread (corporate minus Treasury yield)	FRED
GDP growth	FRED
Nonfarm payroll employment	FRED
Inflation (consumer price index)	FRED
Trailing 1-month realized daily volatility of S&P 500	Bloomberg
Trailing 3-month realized daily volatility of S&P 500	Bloomberg
Trailing 1-month equity return of S&P 500	Bloomberg
Trailing 3-month equity return of S&P 500	Bloomberg

Table I: Macroeconomic and Financial Predictors

We evaluate the performance of models that have varying degrees of interpretability models. We analyze Relevance Based Prediction (RBP), Gated Recurrent Unit Model (GRU), and GRU with Transfer Learning. They are benchmarked against OLS which is a traditionally interpretable model.

B. Relevance-Based Prediction

The core statistical concept in RBP is relevance, which quantifies the contribution of each historical observation x_i to a given prediction task x_t. Relevance is decomposed into two components:*similarity* (closeness to the task) and *informativeness* (uniqueness relative to the sample mean $\bar{x}$). Both components are based on the Mahalanobis distance, which adjusts for scale and correlation in the predictive variables.

$$r_{it} = \text{sim}(x_i, x_t) + \frac{1}{2}\left(\text{info}(x_i, \bar{x}) + \text{info}(x_t, \bar{x})\right)$$

$$\text{sim}(x_i, x_t) = -\frac{1}{2}(x_i - x_t)'\Omega^{-1}(x_i - x_t)$$

$$\text{info}(x_i, \bar{x}) = (x_i - \bar{x})'\Omega^{-1}(x_i - \bar{x})$$

Here, x_i denotes the vector of predictive variables for observation i, x_t is the vector for the prediction task, and Ω is the covariance matrix of the predictive variables. Higher r_{it} indicates greater relevance of observation i for predicting task t.

RBP selectively retains observations with relevance above a threshold r^*. Define the censoring indicator:

$$\delta(r_{it}) = \begin{cases} 1, & r_{it} \geq r^*, \\ 0, & r_{it} < r^*. \end{cases}$$

$$n = \sum_{i=1}^{N} \delta(r_{it}), \qquad \psi = \frac{n}{N}$$

$$\bar{r}_{\text{sub}} = \frac{1}{n}\sum_{i=1}^{N} \delta(r_{it})\, r_{it}$$

The scaling factor λ^2 preserves the distribution of relevance after censoring:

$$\lambda^2 = \frac{\frac{1}{N-1}\sum_{i=1}^{N} r_{it}^2}{\frac{1}{n-1}\sum_{i=1}^{N} \delta(r_{it}) r_{it}^2}.$$

For each cell θ (a combination of variables and a censoring threshold), prediction weights are:

$$w_{it\theta} = \frac{1}{N} + \frac{\lambda^2}{n-1}\left(\delta(r_{it}) r_{it} - \psi \bar{r}_{\text{sub}}\right).$$

These weights tilt the uniform average $1/N$ toward more relevant observations. When censoring is high ($N \gg n$), the weights revert towards $1/N$, limiting overfitting.

The prediction for each cell is given by:

$$\hat{y}_{t\theta} = \sum_{i=1}^{N} w_{it\theta} y_i$$

Each cell receives a reliability score based on adjusted fit:

$$\text{adjusted fit}_{t\theta} = K\left(\text{fit}_{t\theta} + \text{asymmetry}_{t\theta}\right),$$

where K is the number of variables used in the cell and

$$\text{fit}_{t\theta} = \rho(w_{t\theta}, y)^2$$

$$\text{asymmetry}_{t\theta} = \frac{1}{2}\left(\rho(w_t^{(+)}, y) - \rho(w_t^{(-)}, y)\right)^2.$$

Here, $w_t^{(+)}$ uses retained observations and $w_t^{(-)}$ uses censored ones. Fit penalizes small samples; asymmetry rewards cases in which retained observations differ meaningfully from censored ones.

To compute the final grid prediction, each cell receives a reliability weight

$$\psi_\theta = \frac{\text{adjusted fit}_{t\theta}}{\sum_{\tilde{\theta}} \text{adjusted fit}_{t\tilde{\theta}}}.$$

Total prediction weights are:

$$w_{it,\text{grid}} = \sum_{\theta} \psi_\theta\, w_{it\theta}.$$

The final prediction is:

$$\hat{y}_{t,\text{grid}} = \sum_{i=1}^{N} w_{it,\text{grid}} y_i.$$

RBP is grounded in three key theoretical results:

1) **Information Theory and Mahalanobis Distance.** The information in an observation is proportional to the

negative log-likelihood under a multivariate normal distribution, implying information is proportional to Mahalanobis distance.

2) **Equivalence to Linear Regression.** When all variables and all observations are used (no censoring), RBP predictions coincide with those of linear regression.
3) **Fit Converges to R^2.** For a full-sample linear regression, informativeness-weighted average fit equals the model's R^2.

In the following analysis, we will evaluate two versions of RBP. First, RBP single cell corresponds to the prediction of the cell that contains all the features and has a relevance threshold of 50% of the observations. Second, RBP multiple cells corresponds to the weighted average of the predictions of all the single feature cells and the full feature cells, both with a relevance threshold of 0% and of 50%. RBP multiple cells is similar to the approach used by (Czasonis et al., 2024), but with a fewer number of cells to improve computational speed. Computing speed is an important constraint on the practical use of RBP, as discussed in Section VI.

C. Gated Recurrent Units

Given the temporal nature of the volatility forecasting problem, we consider deep sequential models, such as RNNs, GRUs, and LSTMs. As mentioned in Section II, GRUs handle the vanishing and exploding gradient problems observed in RNNs while still being computationally cheaper than LSTMs. Hence, they are our model of choice, representative of black-box sequential models, for the present study.

The GRU accepts a multivariate time series (MVTS) as its input, unlike the OLS and RBP models. Each datapoint is an MVTS of length 18 (corresponding to 18 months for real data) and contains all the macro features and appropriate lagged values at each of those timestamps. The target to be learned is still a floating-point number: the volatility for the next period (or month in case of real data). The inputs are normalized using standard scaling $x' = \frac{x-\bar{x}}{\hat{SD}(x)}$, based on parameters learned from the training data, before entering the network.

Figure 1 shows the model architecture followed: the two GRU layers (64 and 32 units respectively) allow for learning temporal dependencies, while the subsequent dense layers (32, 16, and 1 units respectively) allow for mapping the learned representation of the time series to a single number. It may be noted that the second GRU layer is dropped for experiments with simulated data to reduce computational burden as well as to reduce the tendency to overfit. Dropout layers with dropout probability of 10% along with batch normalization provide additional measure against overfitting and for stability.

For optimizing the MSE loss function, an Adam optimizer with mini-batch gradient descent is used owing to its computational efficiency, low memory requirements, and suitability for problems with a large number of parameters (Kingma and Ba, 2015). The learning rate is initialized at 0.01 and is continuously decayed to allow for a more stable descent. Gradients are clipped to be within a norm of 1.0 to avoid the problem of exploding gradients and unstable updates.

D. Transfer Learning Framework

Despite the overfitting measures listed above, using GRU directly still poses the following challenges:

- The number of trainable parameters (around 26,000) far surpass the number of available datapoints (around 300) for real data, still making the model very susceptible to overfit.
- The model relies on data without regard to underlying linear relationships or economic fundamentals.

One potential solution could be to employ hybrid models, such as the aforementioned GRU-GARCH. While capable of incorporating economic ideas (for example, volatility clustering in this case), these models still do not mitigate the risk of overfit. In fact, the performance and behavior of GRU-GARCH in the present study were found to be virtually the same as GRU, hence prompting us to drop it from further consideration.

The present study uses a transfer learning (TL) framework, which closely parallels that proposed in (Chen et al., 2025). A key difference is the availability of an economically sound model for option pricing in (Chen et al., 2025): the Black-Scholes-Merton model. In contrast, there is no generally agreed upon model for volatility forecasting based on theoretical rather than empirical considerations. Therefore, we use a modified framework suggested in Figure 2.

The TL process starts with fitting an ordinary least squares regression model on the training data. Additionally, distributions for each of the exogenous variables (macro variables, in this case) are also fitted based on the training data. For this purpose common distributions - gaussian, beta, gamma, uniform, lognormal, exponential, and weibull - are considered with trainable parameters. Figure 3 shows the fitted Weibull minimum extreme value distribution (red) for Credit Spread along with the empirical frequency chart observed.

Features are randomly sampled using the fitted distributions and fed into the regression model, which generates the y-values, or next-period volatility in our case, to accumulate a synthetic store of time series data. It may be noted that since lag-1 volatility is also used as a feature in the OLS model, the model predictions are concatenated with the incoming stream of features for several subsequent predictions (with the initial seed value of volatility coming from the learned distribution through random sampling).

It may be noted that one major limitation of this sampling scheme is that it implicitly assumes the features to be approximately independent. The distribution of any feature is assumed to be invariant across time, and past values are assumed to not impact the future values. Similarly, no correlation is assumed between any two features. A more comprehensive scheme would consider the full-rank covariance matrix between the features but is deemed to be too computationally expensive for the present study.

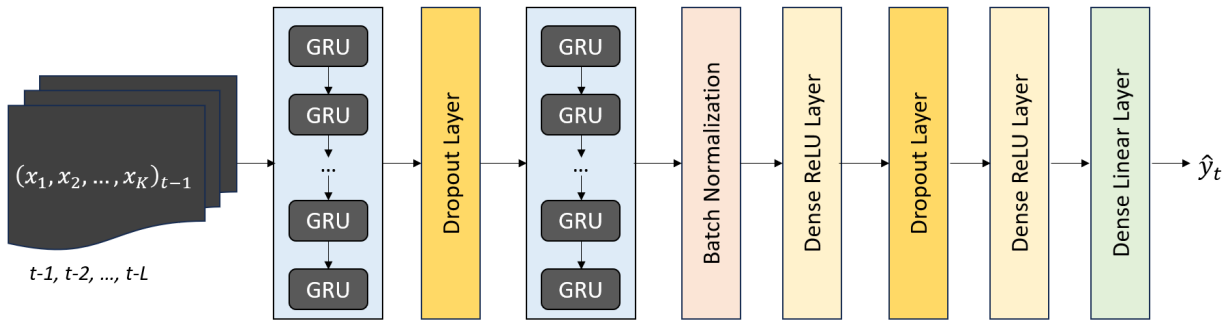

Fig. 1: GRU Model Architecture

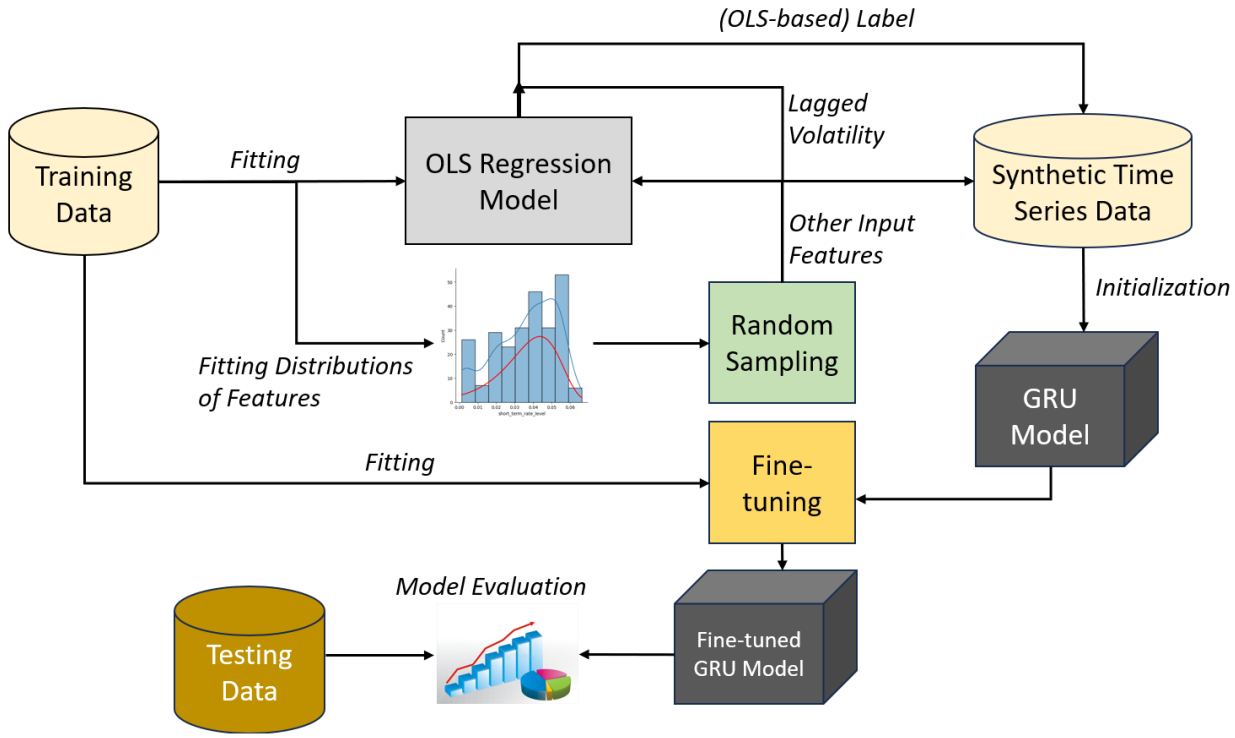

Fig. 2: Transfer Learning Design

The synthetic data store (data size kept comparable to the number of trainable parameters) is used to initialize a GRU model through training with early stopping (using a validation partition on the synthetic data). The initialized model is fine-tuned against the training data with a learning rate $\frac{1}{10}$ of the initial learning rate to minimize the MSE.

IV. Forecasting Performance with Real Data

This section assesses performance of the models to forecast next month's realized volatility of the S&P 500. We compare the predictive ability of OLS, RBP using a single cell, RBP using multiple cells, GRU, and GRU with transfer learning.

First, we assess the models' performance splitting the data into an in-sample training period from 1980 to 2013 and an out-of-sample evaluation period from 2014 to 2025. Figure 4 shows the time series of realized volatility and the predictions of the models. Table II shows the out-of-sample mean square error, mean average error, and R-squared for each of the models.

The results show that the best performing model is RBP with multiple cells, followed by OLS. The rest of the models perform worse than a prediction equal to the out-of-sample mean, as shown by their negative R-squares. This shows that in this modest sample size, the more interpretable models perform better than the ones based on GRUs. The high level of complexity of GRUs makes them prone to overfitting in this context. The transfer learning approach that was intended to alleviate the small sample problem did not improve performance relative to the standard GRU. Finally, only the RBP with multiple cells outperformed OLS, while RBP with a single cell performed worse. This suggests that the ability of RBP to capture more complex relationships relies more on making

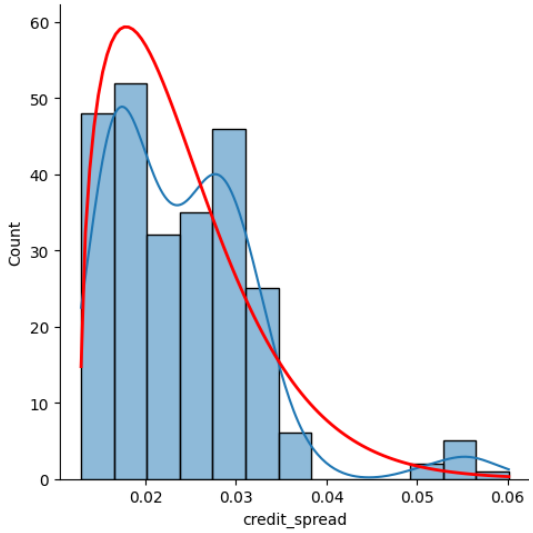

Fig. 3: Observed and Fitted Distributions of the Credit Spread

predictions with different sets of features than on censoring observations with low relevance.

Further, we perform the analysis using a rolling 15-year window for the training sample. We reestimate the models each year and forecast the volatility for each of the 12 months in that year. GRU with transfer learning is initialized only in the first window using synthetic data and subsequently fine tuned with the real data from each rolling window. Figure 5 and Table III show the out-of-sample results.

With rolling window training samples, GRU has a notoriously bad performance, highlighting the risk of using complex models in small datasets. The transfer learning approach mitigates the instability of the GRU without any initialization, but still performs poorly. RBP with multiple cells is still the best performing model. One interesting point to pay attention to is whether the models trained on rolling windows perform better given that they are trained on more recent data that might be more informative to predict future volatility. The second panel of Table III allows to make a fair comparison by showing the performance metrics on the same sample. The table shows that RBP with multiple cells performs slightly better with rolling training, but the rest of the models perform worse.

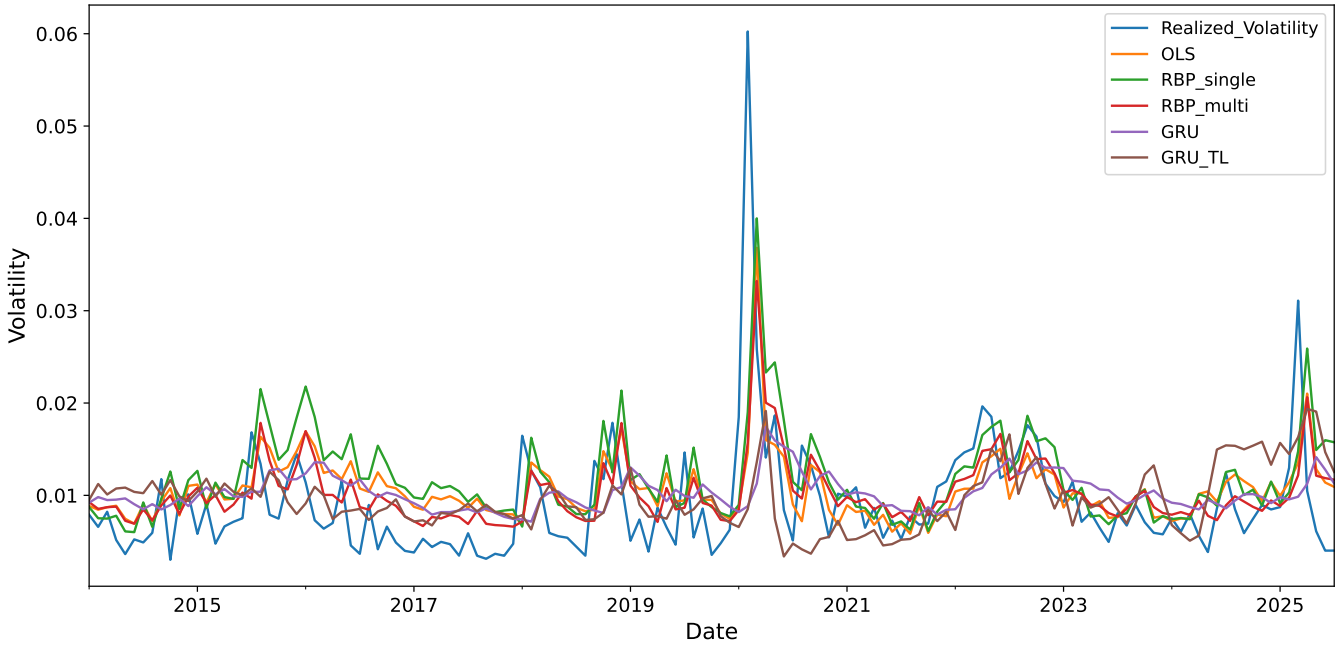

Fig. 4: Realized volatility and model predictions with a fixed training sample.
The figure reports out-of-sample predictions after training on data from 1980 to 2013.

	OLS	RBP Single Cell	RBP Multiple Cells	GRU	GRU TL
MSE	0.000037	0.000043	0.000032	0.000042	0.000045
MAE	0.0041	0.0046	0.0036	0.0043	0.0046
R2	0.091	-0.068	0.213	-0.050	-0.123

Table II: Rolling-window model performance metrics

V. Explainability of Models with Real Data

A. Explaining Relevance-based Prediction

RBP provides two complementary interpretability tools: observation-level relevance and feature-level importance. The first is illustrated in Figure 6, which plots how each in-sample observation contributes to the predictions for the lowest- and highest-volatility test points. This allows us to diagnose whether the weighting scheme behaves as expected. For example, observations from the mid-1980s receive uniformly low relevance (blue), even for extreme volatility events, suggesting a potential area where the current RBP specification could be improved.

The second tool concerns feature importance. Figure 7 reports an importance measure defined as the difference between the average fit of grid cells that include a given variable and those that do not. In contrast to many post-hoc interpretability methods, the importance of each feature can be computed separately for every test observation. A notable pattern appears in March 2020, where the fit contributions of most features, particularly the trailing volatility measures, increase sharply.

B. Explaining GRUs

This section builds on ideas from (Li et al., 2019) about "model fingerprints" for black-box neural networks, adapting them to the specific case of time series data. Unlike attention mechanisms, GRUs are not inherently interpretable; therefore, we need post-hoc methods to understand what is going internally as well as to diagnose any problems (like overfitting to noise).

A unifying theme here is to measure sensitivities to different changes, such as perturbations in input features or shuffling, for example. This is akin to comparing coefficients on different features in an ordinary least-squares regression model. If the features are standardized to the same scale and there is little to no multicollinearity, the magnitude of the features is a measure of model's sensitivity to changes in that feature, and hence a measure of the feature importance.

1) Cell-wise Sensitivities: Unlike the case of a cross-section regression model or the neural networks discussed in (Li et al., 2019), the input to GRU encapsulates both cross-sectional and temporal information: there are multiple features and each feature has multiple timestamps or lags.

One approach to measure sensitivities is to perturb only a particular "cell" (that is, one lag of a particular feature) by a small amount and observe the resultant change in output. Specifically, we choose to perturb the jth lag of the ith feature (x_{ij}) by a small multiple α of its standard deviation ($\sigma_{x_{ij}}$). Let the output change from y_0 to y_1. The operation is performed

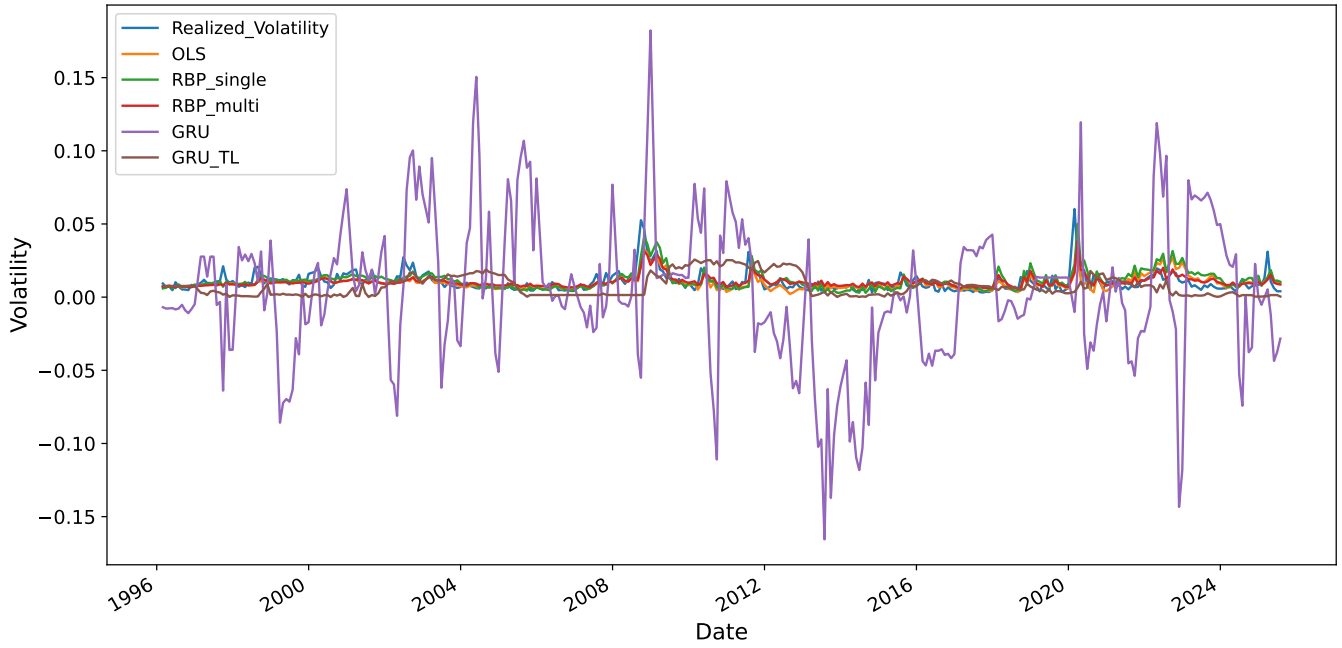

Fig. 5: Realized volatility and rolling-window model predictions.
The figure reports out-of-sample predictions from a ten-year rolling window. OLS and RBP models are re-estimated in each window. The GRU is trained without initialization in every window, while the GRU-TL model is initialized only in the first window using synthetic data and subsequently fine-tuned on real data.

	OLS	RBP Single Cell	RBP Multiple Cells	GRU	GRU TL
			1996–2025		
MSE	0.000030	0.000036	0.000027	0.002435	0.000104
MAE	0.0033	0.0041	0.0033	0.0372	0.0078
R2	0.325	0.189	0.388	-54.363	-1.354
			2014–2025		
MSE	0.000039	0.000054	0.000031	0.002327	0.000075
MAE	0.0039	0.0055	0.0036	0.0377	0.0060
R2	0.072	-0.296	0.255	-54.325	-0.787

Table III: Rolling-window model performance metrics
The first panel computes the performance metrics on the full test sample, while the second panel only computes the metrics on the test sample after 2014 to allow for comparisons with the fixed training sample results.

over all data points ($n = 1, 2, ..., N$) in the testing sample. Then, the cell-wise sensitivity is

$$sensitivity(feat = i, t = k) = \frac{1}{N} \sum_{n=1}^{N} \frac{y_1^{(n)} - y_0^{(n)}}{\alpha \cdot \sigma_{x_{ij}}}$$

This closely parallels the notion of a partial derivative. The averaging serves to minimize the error since the partial derivative is based on empirical data and approximations here. Note that the amount by which the input is perturbed could be positive ($\alpha > 0$, therefore, increase) or negative ($\alpha < 0$, therefore, negative).

Figure 8 plots the cell-wise sensitivities of the fine-tuned GRU based on the real volatility and macro data using $\alpha = +0.1$ for all 18 lags of the features used. It is easy to note that the predictions are most sensitive to the most recent values of the trailing 21-day volatility. In fact, we observe an almost regularly decreasing trend in sensitivity to past volatility with increasing lags. This aligns well with the intuitive argument that the most recent values of volatility are most representative of the current market uncertainty and hence affect the future volatility the most. A similar observation may be made for the trailing index returns. The sensitivities of most of the macro features peak after a few lags, which aligns well with the fact that the macro movements are typically much slower than the actively traded financial markets and hence effects take time to manifest. Finally, the fact that the sensitivity curves remain very similar across training (red) and test (blue) partitions paints a favorable picture about the generalization capabilities of the transfer-learning based GRU.

Figure 9 contrasts the sensitivities of the vanilla GRU (trained directly on the real data) against those of the fine-tuned GRU (trained using transfer learning). It is immediately

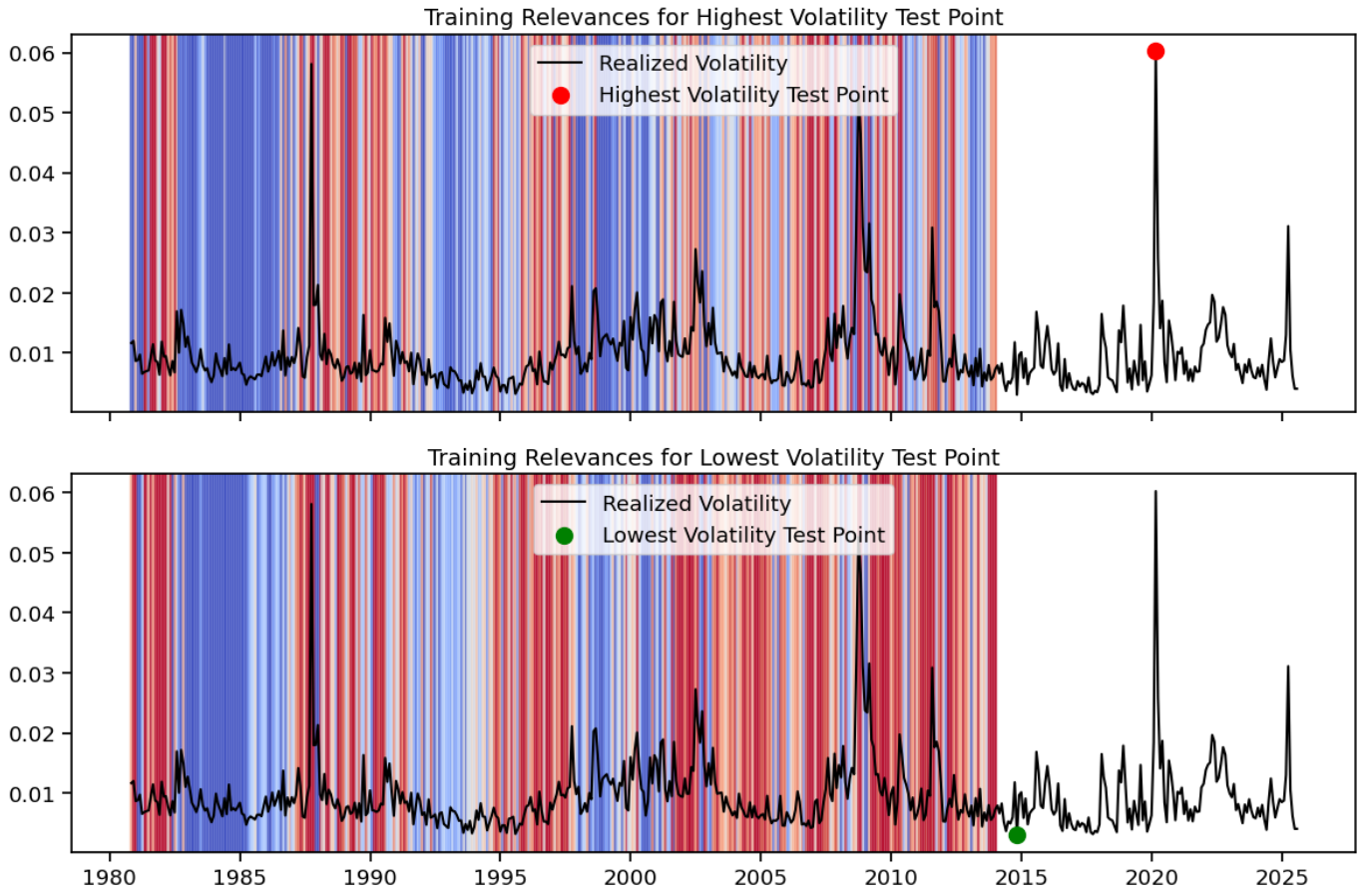

Fig. 6: In-Sample Relevances for Highest and Lowest Volatility Dates in Out-of-Sample data

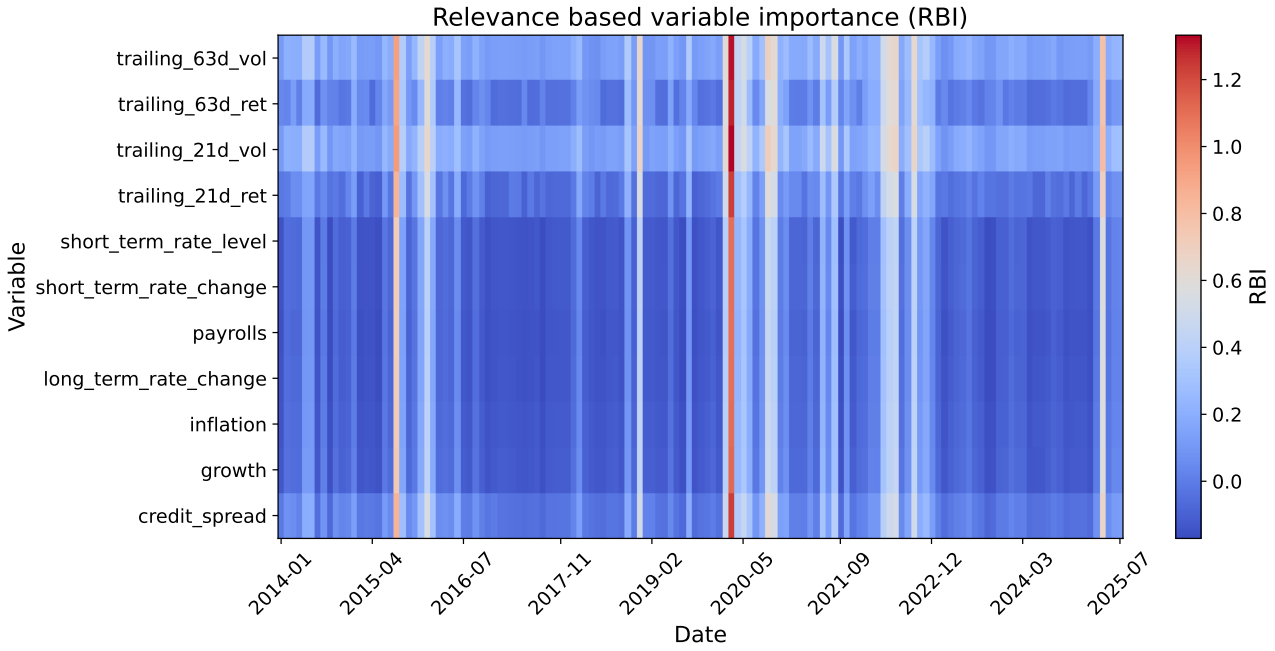

Fig. 7: Feature Importance across time

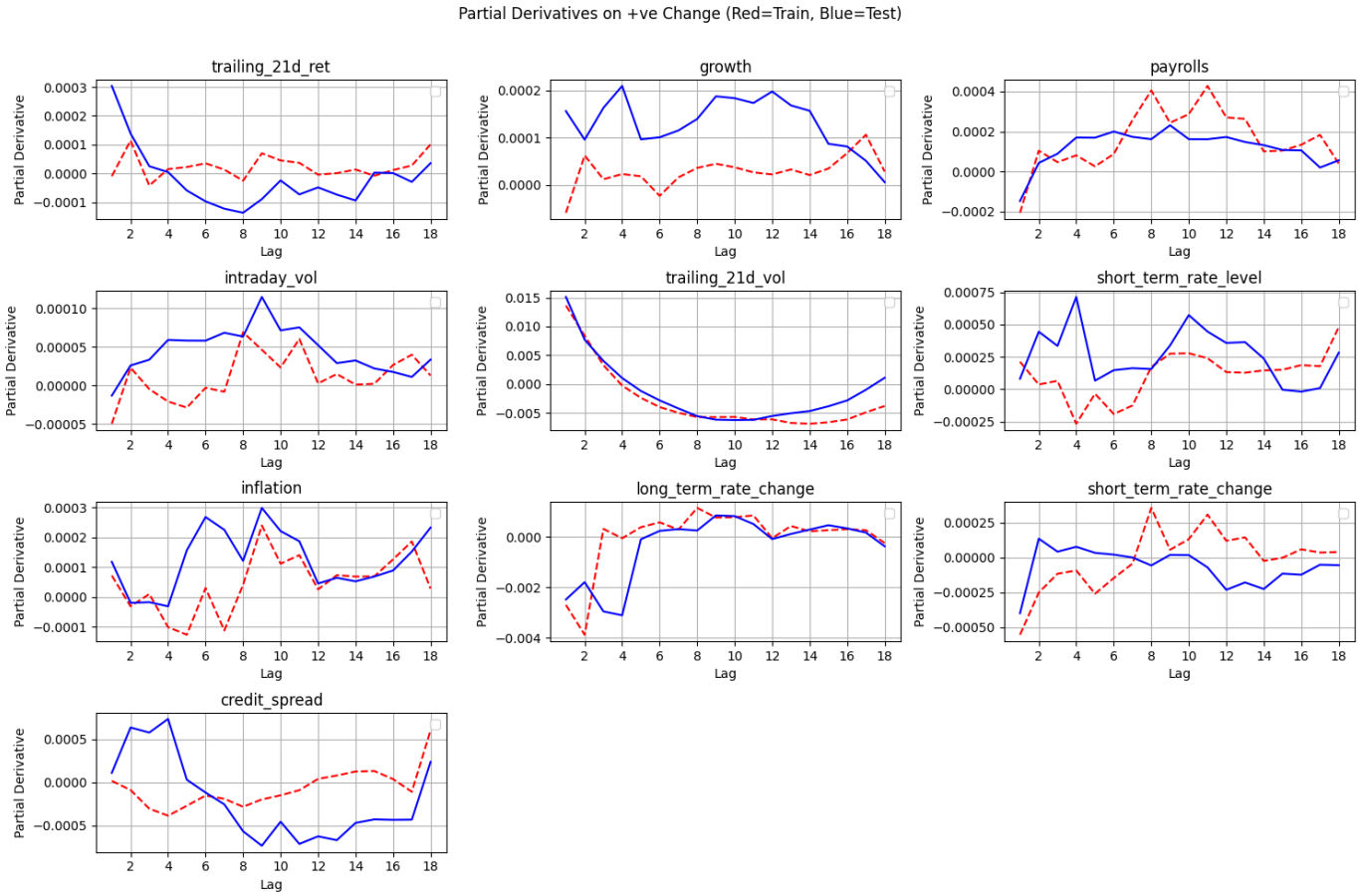

Fig. 8: Cell-wise Sensitivities for the Fine-tuned GRU on Training and Testing Data

obvious that the sensitivities for the vanilla GRU are much larger in magnitude and they show large oscillations across lags. This indicates a very poor fit. Parallels can be drawn to an ordinary least-squares regression model plagued by severe multicollinearity. Additionally, it also serves to explain Figure 5, wherein the predictions of the vanilla GRU show large movements compared to the TL-based GRU and the other models.

2) Feature-wise and Pair-wise Sensitivities: Another approach to measure sensitivity on a less granular level is to apply the same change across all timestamps for a particular feature and measure the change in the predicted output. This provides an idea about the sensitivity of the prediction towards the feature as a whole. Formally, if for the ith feature (with standard deviation standardized to 1), if we apply a perturbation of α, changing the output from y_0 to y_1, the sensitivity is

$$sensitivity(feat = i) = \frac{1}{N}\sum_{n=1}^{N}\frac{|y_1^{(n)} - y_0^{(n)}|}{\alpha}$$

This notion can be extended to account for pairwise interaction, similar to (Li et al., 2019). We can perturb two features x_i and x_j by the same amount α (provided they are both standardized to have the unit standard deviation):

$$sensitivity(feats = (i,j)) = \frac{1}{N}\sum_{n=1}^{N}\frac{|y_1^{(n)} - y_0^{(n)}|}{\sqrt{2}\alpha}$$

The factor of $\sqrt{2}$ enables direct comparison with the case we perturb only one feature, making the assumption that the changes in the features are not very highly correlated.

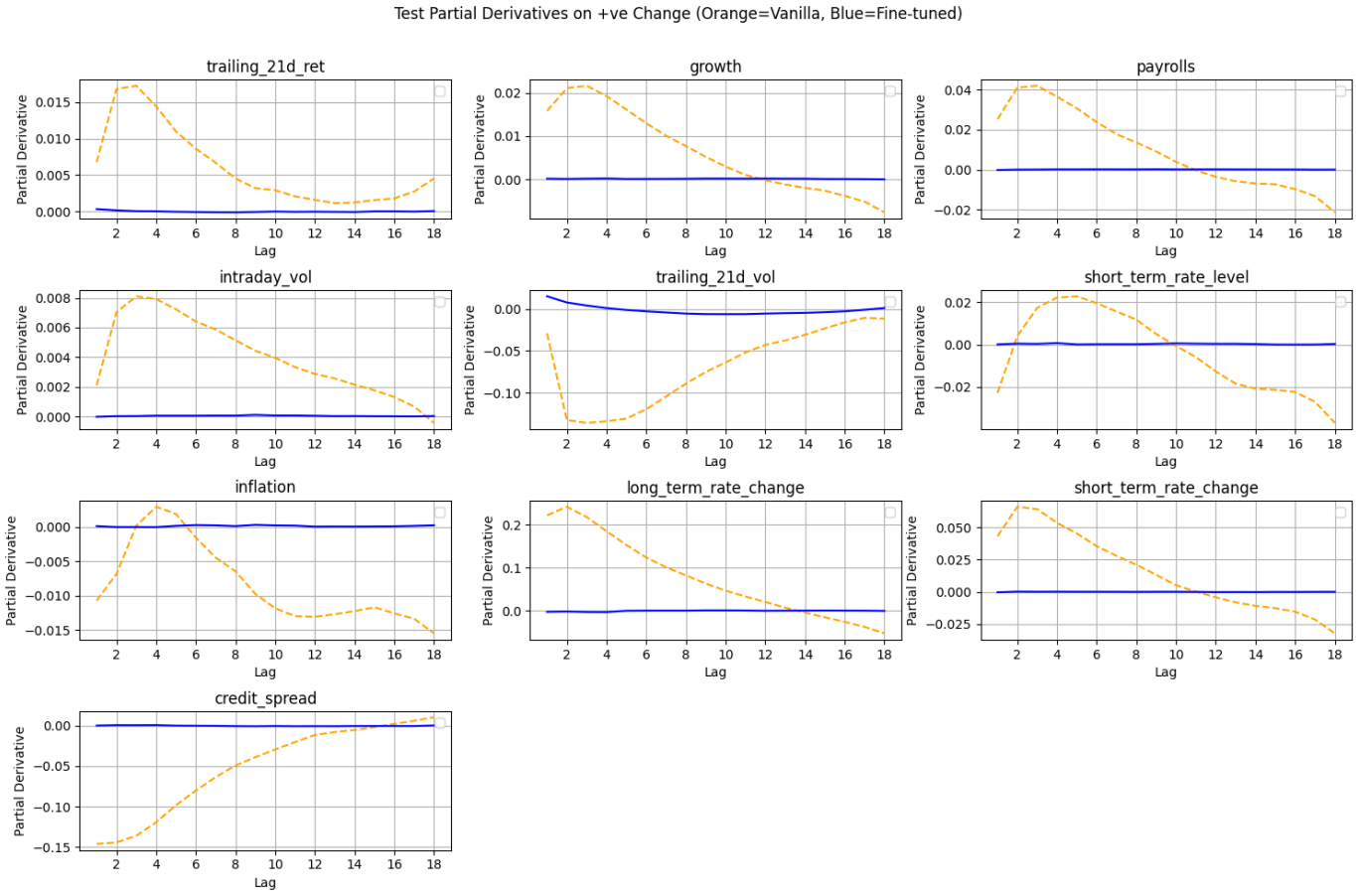

Fig. 9: Cell-wise Sensitivities for the vanilla and TL-based GRU on the Test Data

Figure 10 displays the absolute sensitivities towards individual features as well as pairs of features on a heatmap. Based on the diagonal entry, the trailing 21-day volatility emerges as the most important feature, followed by the long-term rate change and short-term rate level. The fact that not all off-diagonal entries are zero indicates that it might be reasonable to consider interaction terms in a polynomial regression than just considering only individual effects.

3) Sensitivity to Time Series Structure: Figure 8 indicates that the model has different sensitivities towards different lags of the same features. Furthermore, the recurrent nature of GRUs is known to encapsulate temporal interactions and dependencies. Therefore, it stands to reason that destroying the temporal structure of the model (by shuffling the inputs across time, for example) should result in a deterioration in model's performance. To measure the sensitivity of the model towards the time series structure of different features, we calculate the percentage change in MSE of the model with that feature permuted in time against the original model.

As indicated in Figure 11, the out-of-sample MSE always increases when the features are shuffled across time. Furthermore, the more important the feature (per the previous subsection on feature-wise importance), the higher the impact of destroying the time series structure. It also makes intuitive sense that the impact of shuffling the trailing 21-day volatility is the highest by a significant margin given the clear decreasing pattern in sensitivity across lags in Figure 8.

4) Partial Functional Form: While partial derivatives discussed previously provide information about the sensitivity towards a feature for small changes around a datapoint, they fail to capture the monotonic or non-monotonic behavior of the function encapsulated by the model across a wider range of feature values. Figure 12 illustrates the variation of the predicted values when any particular input feature is changed (across all timestamps) keeping the other features constant. Unlike small perturbations, the behavior is likely to change a lot depending on other feature values; therefore, the other features are held at their median value for consistent comparison. It is easy to note that the variations are non-monotonic but largely smooth for most features.

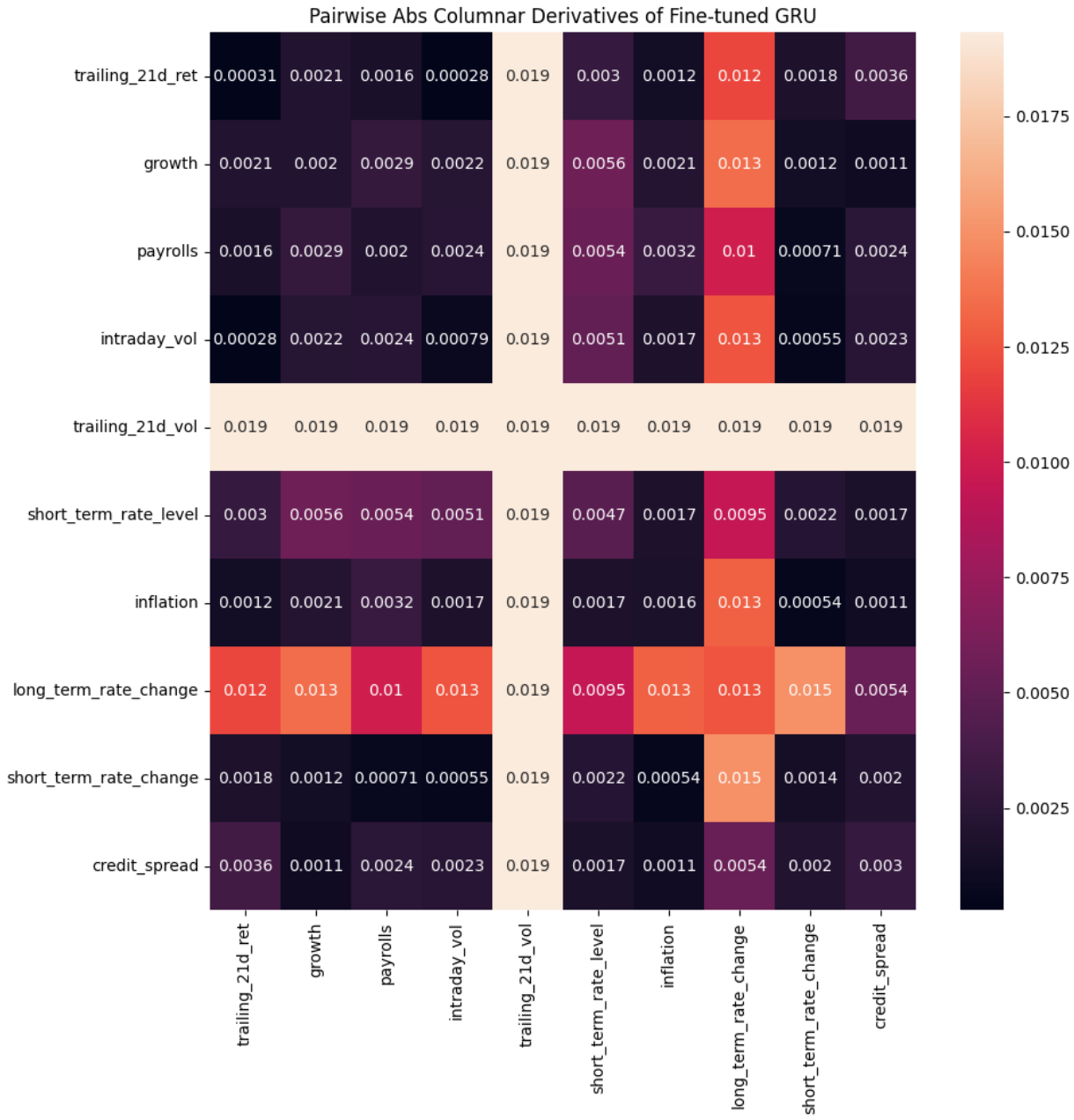

Fig. 10: Feature- and pair-wise sensitivities for TL-based GRU

VI. Forecasting Performance under Simulated Scenarios

This section describes the simulation framework used to evaluate the performance of OLS, Relevance-Based Prediction (RBP), and GRU neural networks. The simulations consist of three components: (i) a latent Markov-switching regime, (ii) a high-dimensional vector of predictors following a regime-dependent VAR(1) process, and (iii) a scenario-specific outcome equation. We adapt eight simulation scenarios (A–H) from (Song et al., 2025) to a time series context. The scenarios are constructed to vary the degree of linearity, nonlinearity, interactions, and noise.

A. *Latent Regime Process*

We introduce stochastic structural shifts that have time-series dependence through a two-state Markov chain:

$$S_t \in \{0, 1\},$$

where state 0 denotes a *calm* regime and state 1 denotes a *stress* regime. The transition probabilities are governed by

$$P = \begin{bmatrix} p_{CC} & 1 - p_{CC} \\ 2(1 - p_{CC}) & 1 - 2(1 - p_{CC}), \end{bmatrix} \qquad p_{CC} = 0.99.$$

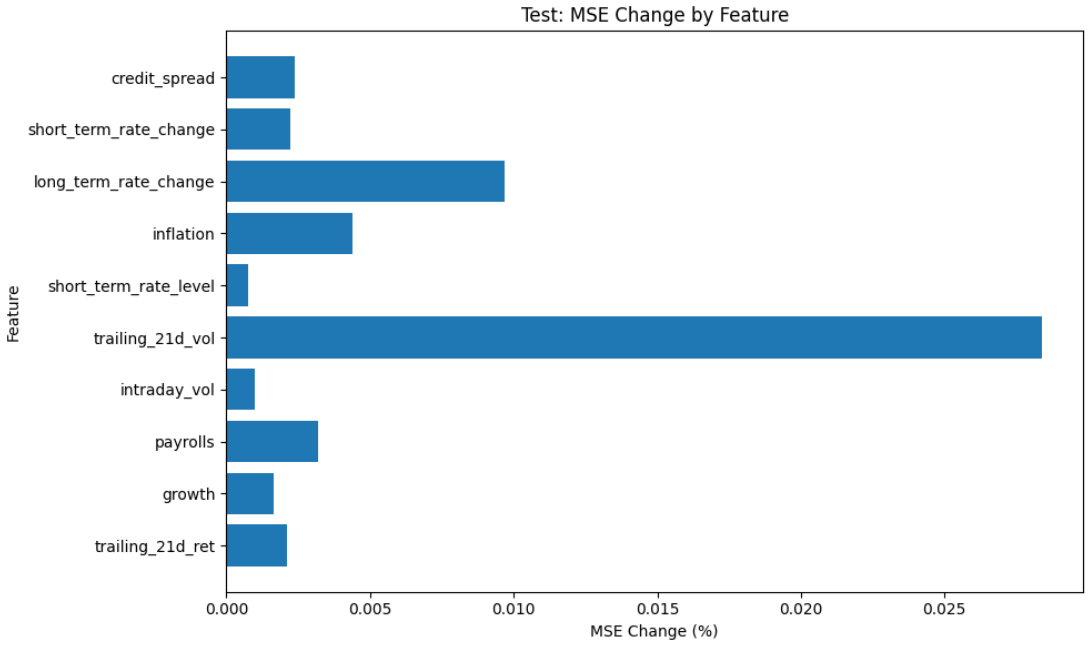

Fig. 11: Feature-wise impact of temporal shuffling on out-of-sample MSE (fractional change)

This matrix is constructed so that the chain has a fixed stationary distribution

$$\pi = \left(\frac{2}{3}, \frac{1}{3}\right),$$

implying that calm periods occur twice as frequently as stress periods in the long run. Additionally, the process is constructed in such a way that states are highly persistent. The chain is initialized from this stationary distribution and simulated forward.

B. Predictor Dynamics: Regime-Dependent VAR(1)

Predictors are given by a d-dimensional process

$$X_t = (x_{t,1}, \ldots, x_{t,d})^\top, \qquad d = 10,$$

that evolves as a VAR(1):

$$X_t = \Phi X_{t-1} + u_t.$$

The autoregressive matrix Φ is diagonal,

$$\Phi = \mathrm{diag}(\phi_1, \ldots, \phi_d), \qquad \phi_j \sim \mathrm{Uniform}(-0.1, 0.9),$$

so predictors exhibit heterogeneous persistence ranging from weakly mean-reverting to near-unit-root behavior.

Shocks are Gaussian,

$$u_t \sim N(0, \Sigma_{S_t}),$$

with covariance matrices:

$\Sigma_0 =$ Equicorrelated with variance 1 and correlation $\rho = 0.2$,

$\Sigma_1 =$ Equicorrelated with variance 1.2 and correlation $\rho = 0.4$.

Thus, in the stress regime the predictors exhibit both higher variance and stronger cross-sectional comovement. A burn-in period ensures convergence to the stationary distribution of the VAR.

C. Scenario Parameters and Outcome Equations

To introduce heterogeneity across the eight scenarios, we draw independently:

- linear coefficients $b_1, b_2 \in \mathbb{R}^d$ from a normal distribution,
- symmetric quadratic matrices $C_1, C_2 \in \mathbb{R}^{d \times d}$ from a normal distribution,
- subsets of indices $Z_1, Z_2 \subset \{1, \ldots, d\}$ used for higher-order interactions.

Noise terms follow

$$\varepsilon_t \sim N\left(0, \sigma^2_{S_t} \cdot\right),$$

Allowing $\sigma^2_{S_t}$ to change depending on the scenario.

The target variable is generated according to

$$y_{t+1} = \mu_{S_t} + k_{S_t} \cdot f_{\text{scenario}}(X_t, S_t) + \varepsilon_{t+1},$$

where f_{scenario} differs across simulations A–H. k_{S_t} is a scaling constant that ensures that the variance of $k_{S_t} \cdot f_{\text{scenario}}(X_t, S_t)$ is equal to 1. We set $\mu_0 = 0$ and $\mu_1 = 1$, such that the calm regime has a lower mean. The regime dependent DGP has the following form:

a) Simulation A: Linear in both regimes.:

$$f_{\text{A}}(X_t, S_t) = \begin{cases} X_t^\top b_1, & S_t = 0, \\ X_t^\top b_2, & S_t = 1. \end{cases}$$

b) Simulation B: Linear in calm, nonlinear in stress.:

$$f_{\text{B}}(X_t, S_t) = \begin{cases} X_t^\top b_1, & S_t = 0, \\ \sin(X_t)^\top b_2, & S_t = 1. \end{cases}$$

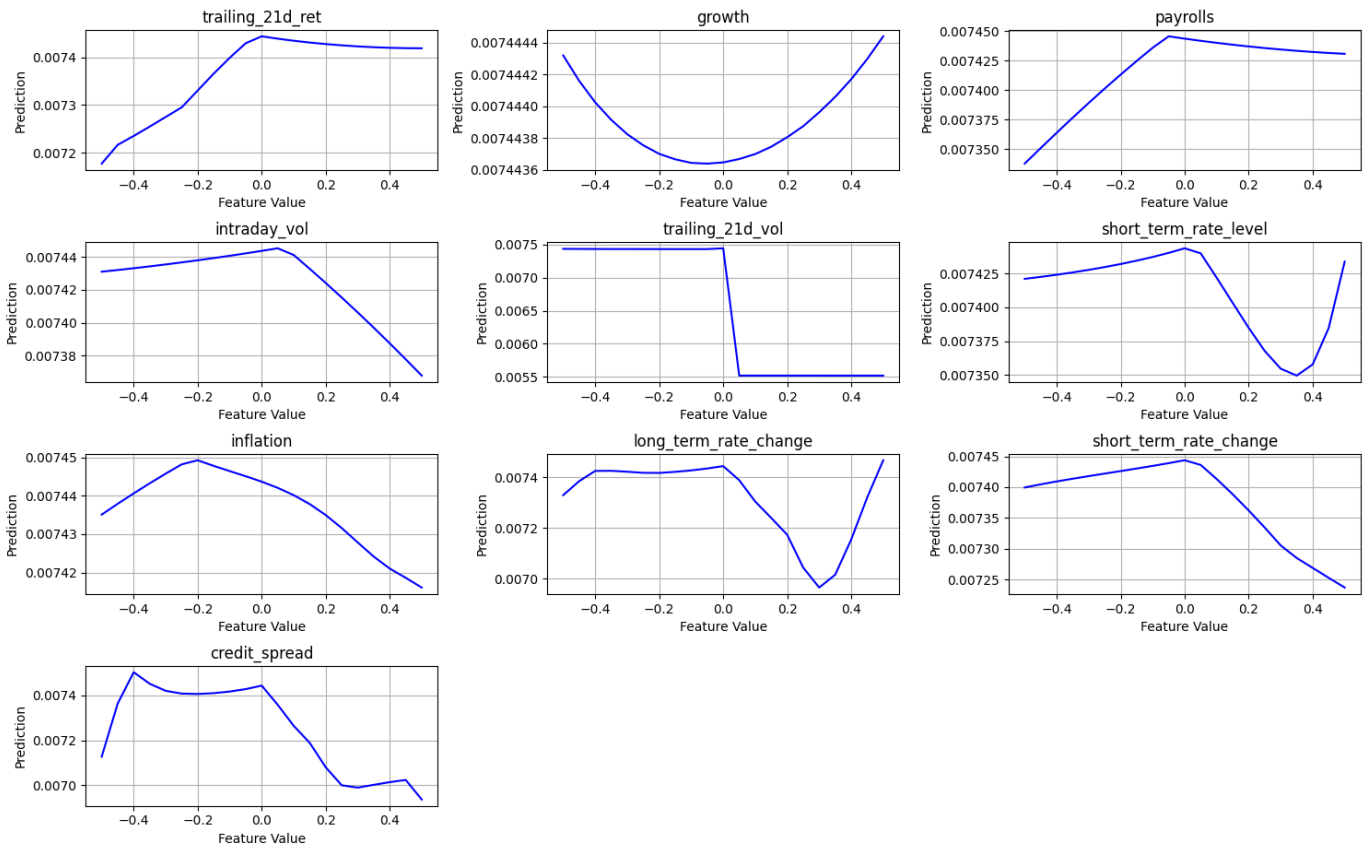

Fig. 12: Variation of predictions with individual features holding other inputs constant

c) Simulation C: Nonlinear in calm, linear in stress.: Reverse of scenario B.

d) Simulation D: Quadratic nonlinearities in both regimes.:

$$f_{\mathrm{D}}(X_t, S_t) = \begin{cases} X_t^\top b_1 + X_t^\top C_1 X_t, & S_t = 0, \\ \sin(X_t)^\top b_2 + \sin(X_t)^\top C_2 \sin(X_t), & S_t = 1. \end{cases}$$

e) Simulation E: Same structure as D but higher noise variance.:

f) Simulation F: Reverse regime roles from D.:

g) Simulation G: High-order interaction terms.:

$$f_{\mathrm{G}}(X_t, S_t) = \begin{cases} X_t^\top b_1 + X_t^\top C_1 X_t \\ \quad + \prod_{k \in Z_1} X_{t,k} + \prod_{k \in Z_2} X_{t,k}, & \text{if } S_t = 0, \\ \sin(X_t)^\top b_2 \\ \quad + \sin(X_t)^\top C_2 \sin(X_t) \\ \quad + \prod_{k \in Z_1} \sin(X_{t,k}) \\ \quad + \prod_{k \in Z_2} \sin(X_{t,k}), & \text{if } S_t = 1. \end{cases}$$

where Z_1 and Z_2 represent disjoint sets with 3 to 5 elements.

h) Simulation H: : Reverse regime roles from G.

Table IV summarizes the degree of nonlinearity, interactions, and noise across scenarios.

Scenario	Nonlinearity	Pairwise Interactions	High-Order Interactions
A	None	No	No
B–C	Yes	No	No
D–F	Yes	Yes	No
G–H	Yes	Yes	Yes

Table IV: Structural complexity of simulation scenarios A–H.

D. Relevance-based prediction runtime

Figure 13 shows how long it takes for RBP single cell to compute forecasts for all the training sample as the number of observation increases. The figure is made with simulated data from scenario A and a 70-30 train-test sample. The results of this exercise show that the runtime of RBP increases exponentially with the number of observations. For a sample of only 1,000 observations, it takes RBP around 8 minutes to perform the forecasts. The long runtime makes it infeasible to compute RBP for many cells or for large datasets. Therefore, we exclude RBP multiple cells from the simulation analysis

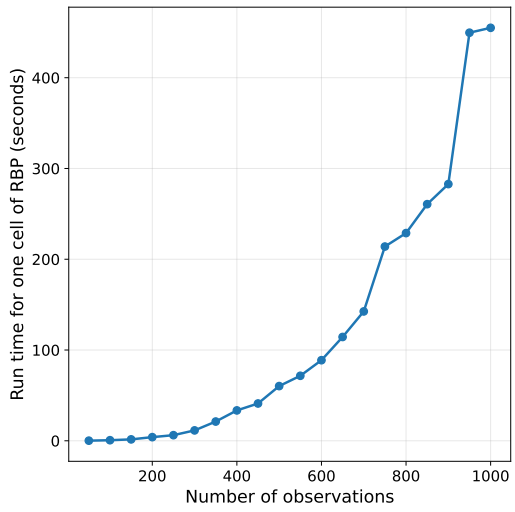

Fig. 13: Relevance based prediction runtime

and only include RBP single cell in simulations with a low number of observations.

E. Model Estimation and Performance

After simulating $\{S_t\}$, $\{X_t\}$, and $\{y_t\}$, the dataset is then used to train and evaluate OLS, RBP, and GRU forecasting models. In all models, we use a 30-70 train-test split, using a dataset with 600 observations, which corresponds to 50 years of monthly data The results are reported in Tables V, VI, and VII. Additionally, Figure 14 shows a boxplot of the MSE across simulated paths for scenario A and H. The results show that RBP performs slightly better than OLS, especially in the more complex scenarios. However, the boxplot suggests this might just be due to randomness. The inherently interpretable models perform better than GRU across all scenarios, although the gap narrows down in the scenarios with more complexities.

Table V: Mean squared error (MSE) 600 observations.

model scenario	OLS	RBP single cell	GRU	GRU-TL
A	0.543010	0.519400	1.064532	1.015869
B	0.579139	0.583523	1.166882	1.075451
C	0.673947	0.653527	1.088774	1.070044
D	1.047784	0.986106	1.267059	1.175112
E	1.117995	1.051933	1.327884	1.255447
F	0.873379	0.833979	1.044387	1.021356
G	1.116381	1.018640	1.246788	1.211247
H	0.973995	0.916341	1.063757	1.020792

Table VI: Mean absolute error (MAE). 600 observations.

model scenario	OLS	RBP single cell	GRU	GRU-TL
A	0.535064	0.518050	0.806168	0.792278
B	0.557949	0.553805	0.852881	0.816421
C	0.616204	0.614839	0.805015	0.800387
D	0.752255	0.740449	0.828698	0.790464
E	0.785132	0.771313	0.855633	0.827431
F	0.665155	0.667988	0.722758	0.708338
G	0.753936	0.736066	0.798843	0.780662
H	0.637812	0.646748	0.663850	0.644976

Table VII: Coefficient of determination (R^2) 600 observations.

model scenario	OLS	RBP single cell	GRU	GRU-TL
A	0.543219	0.538063	0.070454	0.120873
B	0.527783	0.497612	0.037142	0.113724
C	0.367489	0.351457	-0.015487	0.016800
D	0.146088	0.178892	-0.027386	0.046591
E	0.132068	0.173714	-0.027806	0.031428
F	0.147518	0.161236	-0.027287	0.016265
G	0.111923	0.144790	0.006988	0.035307
H	-0.029606	-0.038881	-0.086504	-0.001331

We also perform the analysis with a dataset of 25,000 observations, which corresponds to roughly 100 years of daily data. The results are reported in Tables VIII, IX, and X. Figure 15 shows the boxplot for scenario A and H. OLS has a better performance in the more simple scenarios, but is slightly less accurate in the presence of interactions. This highlights that non-inherently interpretable models might be useful to forecast in highly non-linear settings, but it requires to have a large-enough training sample that might not be available in many contexts.

Table VIII: Mean squared error (MSE) 25,000 observations.

model scenario	OLS	GRU	GRU-TL
A	0.492645	0.777518	0.791403
B	0.488020	0.803391	0.821476
C	0.636441	0.859459	0.875114
D	0.911964	0.915982	0.927117
E	0.981092	0.977917	0.991867
F	0.890844	0.897176	0.908729
G	0.994798	0.971887	0.984777
H	1.107968	1.037427	1.036870

Table IX: Mean absolute error (MAE) 25,000 observations.

model scenario	OLS	GRU	GRU-TL
A	0.533203	0.683326	0.688663
B	0.530593	0.702834	0.709991
C	0.613786	0.698788	0.702414
D	0.704083	0.686229	0.691474
E	0.740446	0.724094	0.731245
F	0.608476	0.597463	0.601377
G	0.690514	0.657039	0.663257
H	0.539728	0.481362	0.481988

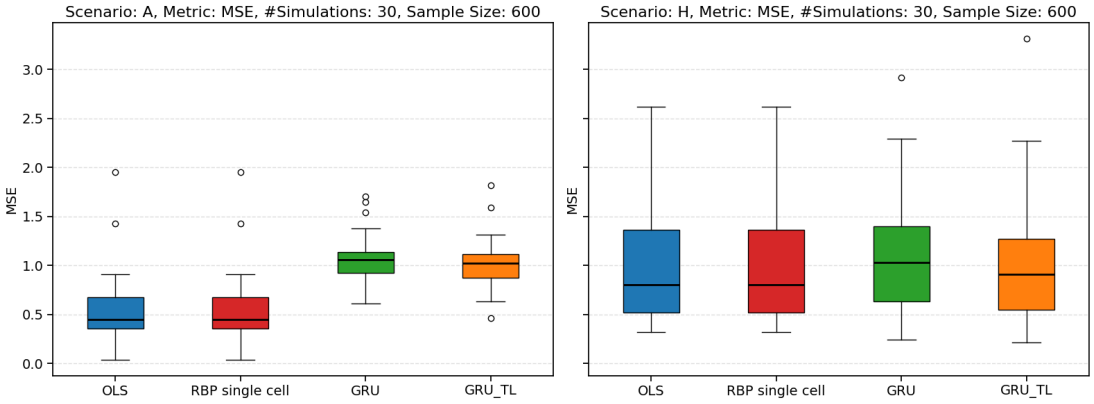

Fig. 14: Box plot for MSE across simulated paths of scenarios A and H, 600 observations

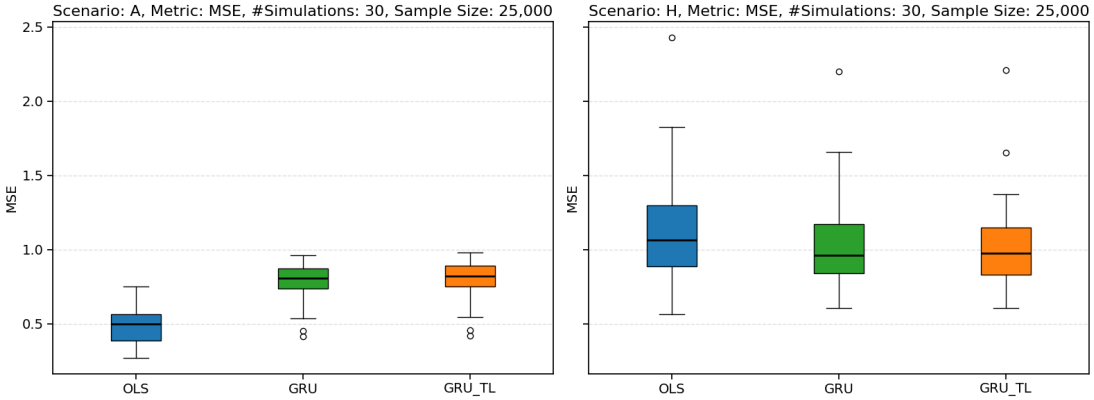

Fig. 15: Box plot for MSE across simulated paths of scenarios A and H, 25000 observations

Table X: Coefficient of determination (R^2) 25,000 observations.

model scenario	OLS	GRU	GRU-TL
A	0.599843	0.367583	0.356343
B	0.603231	0.346609	0.331874
C	0.482471	0.302270	0.289842
D	0.284524	0.277654	0.268630
E	0.264186	0.262819	0.252583
F	0.297601	0.287143	0.277330
G	0.231335	0.245654	0.235417
H	0.195187	0.237857	0.238148

VII. Conclusion and Future Work

Predictive performance alone does not justify the use of an opaque model, especially in trust-heavy and regulated industries like finance. A notion of interpretability around reliable performance and explainability is needed to scrutinize a potential model before deployment in such applications. We examined a series of increasingly complex modeling techniques under these lens: OLS regression, RBP, GRUs, and GRUs under a transfer learning framework. A noteworthy observation made was that there need not always be a tradeoff between performance and model transparency or complexity. Relevance based prediction delivered the best fits for the real data, followed by OLS. Inherently interpretable models maintained high performance under different simulated scenarios. Relevance-based prediction showed ability to handle complex relationships beyond OLS, but has the drawback of high computational requirements, which makes it infeasible for problems with large data samples.

GRUs, the most opaque of the models studied, struggled with small amounts of data, common in financial and macroeconomic settings. They added value over OLS when the data was moderately high and the scenario was simulated to include high non-linear terms. Transfer learning helped stabilize GRUs for the case of very small training samples as shown in the rolling window training, but did not produce any improvement beyond a certain sample size. The diminishing utility of transfer learning and worsened performance in case of an ill-specific teacher model (as in simulated scenarios) are consistent with the insights from (Chen et al., 2025).

We note the following limitations for the present study:

- We drop RBP with multiple cells and large data samples from the experiments with simulated paths due to high runtimes.
- The transfer learning uses independent distributions for the different features during data generation through

random sampling. A more thorough investigation, with higher compute resources, would consider the full-rank covariance matrix between the macro-economic variables while drawing from the joint distribution. Furthermore, OLS may not be the best teacher model due to lack of theoretical justification (unlike the case of Black-Scholes model in (Chen et al., 2025))

- We consider the model performance in aggregate while not considering if the errors have significantly different economic costs (such as errors made in predicting volatility spikes versus relatively calmer periods).

The study paves several opportunities for further research:

- A different, higher-frequency financial time series forecasting problem, such as hourly trading volume prediction or market impact estimation, might help shed light on whether models with a high number of parameters (like the GRU) are better in other financial settings.
- Transfer learning may result in better results for more complex models, such as LSTMs, CNNs, transformers, and fully-connected neural networks. TL may also perform better even in time series settings when there is a teacher model based on economic or financial theory (similar to Black-Scholes model in (Chen et al., 2025)).
- Given RBP's remarkable balance between interpretability and predictive performance, efforts to improve its runtime efficiency are warranted.
- While the current study investigates sensitivity of the final outputs to the model inputs for GRUs, further light on how predictions are generated can be shed by considering the effect on intermediate hidden layer distributions.

VIII. Acknowledgments

The authors extend their sincere gratitude to Prof. Hui Chen for his continuous guidance and supervision throughout this project, as well as for organizing insightful guest lectures on state-of-the-art machine learning research in finance. We are also grateful to Prof. David Thesmar, Prof. Leonid Kogan, and the course TA, Xin Xiong, for their valuable feedback and support. Finally, we would like to express our appreciation to David Turkington (State Street) for generously taking the time to deliver a lecture on interpretability in financial machine learning.

References

Val Srinivas Alexey Surkov, Jill Gregorie. Unleashing the power of machine learning models in banking through explainable artificial intelligence (xai). *Deloitte Insights*, 2022.

Fikry Mansour M Alsheebah and Belal A Al-Fuhaidi. Emerging stock market prediction using gru algorithm: Incorporating endogenous and exogenous variables. *IEEE Access*, 2024.

Deylan Boychev. Interpretable computer vision models through adversarial training: Unveiling the robustness-interpretability connection. *Computer Vision and Pattern Recognition*, 2023.

Hui Chen, Yuhan Cheng, Yanchu Liu, and Ke Tang. Teaching economics to the machines. 2025. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4642167.

Kyunghyun Cho, Bert van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics, and Structure in Statistical Translation*, 2014.

Megan Czasonis, Mark Kritzman, and David Turkington. A transparent alternative to neural networks with an application to predicting volatility. 2024. Available online.

Prateek Goel and Rosina Weber. Measuring interpretability: A systematic literature review of interpretability measures in artificial intelligence. *The Florida AI Research Society*, 2025.

Alan Jacovi and Yaov Goldberg. Towards faithful (nlp) systems: How should we define and evaluate faithfulness? *Proceedings of the Meeting of the Association of Computational Linguistics*, 2020.

Ha Young Kim and Chang Hyun Won. Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Internation Conference for Learning Representations*, 2015.

Tobias Hoogteijling Laurens Swinkels. Forecasting stock crash risk with machine learning. *Robeco: The Investment Engineers*, 2022. White Paper.

Yimou Li, David Turkington, and Alireza Yazdani. Beyond the black box: An intuitive approach to investment prediction with machine learning. (SSRN 3475538), 2019. URL `https://ssrn.com/abstract=3475538`. Posted November 5, 2019; Date Written October 23, 2019.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 2017.

Felipe Oviedo, Zekun Ren, Shijing Sun, Charles Settens, Liu Zhe, Noor Hartono, Savitha Ramasamy, Brian DeCost, Giuseppe Romano, Aaron Kusne, and Tonio Buonassisi. Fast and interpretable classification of small x-ray diffraction datasets using data augmentation and deep neural networks. *npj Computational Methods*, 2019.

Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Philipp Schmidt and Felix Biessmann. Quantifying interpretability and trust in machine learning systems. *AAAI Workshop on Network Interpretability for Deep Learning*, 2019.

Huili Song, Megan Czasonis, David Turkington, and Yin Li. Replacing cross-validation with interrogation: A universal test for underfitting and overfitting. April 23 2025. doi: 10.2139/ssrn.5229414. Available at SSRN: `https://ssrn.com/abstract=5229414`.

Jingyi Wei, Steve Y Yang, and Zhenyu Cui. Unified garch-recurrent neural network in financial volatility forecasting. 2025. Available online: https://ssrn.com/abstract=5215228.

Ziqi Zhao, Yucheng Shi, Shushan Wu, Fan Yang, Wenzhan Song, and Ningaho Liu. Interpretation of time-series deep models: A survey. 2023. Available online: https://arxiv.org/pdf/2305.14582.