Teaching Economics to the Machines

Hui Chen Yuhan Cheng Yanchu Liu Ke Tang*

December 12, 2024

Abstract

Structural models in economics often suffer from a poor fit with the data and demonstrate suboptimal forecasting performances. Machine learning models, in contrast, offer rich flexibility but are prone to overfitting and struggle to generalize beyond the confines of training data. We propose a transfer learning framework that incorporates economic restrictions from a structural model into a machine learning model. Specifically, we first construct a neural network representation of the structural model by training on the synthetic data generated by the structural model and then fine-tune the network using empirical data. When applied to option pricing, the transfer learning model significantly outperforms the structural model, a conventional deep neural network, and several alternative approaches for bringing in economic restrictions. The outperformance is more significant i) when the sample size of empirical data is small, ii) when market conditions change relative to the training data, or iii) when the degree of model misspecification is likely to be low.

Keywords: transfer learning, structural model, deep neural networks, misspecification, option pricing

*Chen: MIT Sloan and NBER. Cheng: School of Management, Shandong University. Liu: Lingnan College, Sun Yat-sen University. Tang: PBC School of Finance and Institute of Economics, Tsinghua University. We thank Lars Hansen, Kewei Hou, Kai Li, Stefan Nagel, Dacheng Xiu, Qunzi Zhang, Guofu Zhou, and participants at the Econometric Society Summer School in Lausanne, CICF, AMES, SIF conference, University of Chicago, Tsinghua University, HKUST, Central University of Finance and Economics, Renmin University of China, and University of Melbourne for insightful comments.

1 Introduction

"All models are wrong, but some are useful." While acknowledging the imperfections of scientific modeling, George Box's famous quote stresses their values in guiding us to learn from data and make informed decisions. However, the combination of big data and advances in machine learning techniques presents data-driven approaches as a potential new way of understanding the world without relying on traditional models and theories. One may even wonder, "Is theory dead?"[1]

Debates about the role of theory have long existed in economics. On the one hand, structural models in economics can convey appealing insights but tend to be scarcely parameterized and offer unsatisfactory fitting for empirical data. On the other hand, reduced-form models offer significantly more flexibility and superior forecasting performances, which increasingly make them the preferred approach in prediction and decision-making tasks. However, a key limitation of the reduced-form approach is that it often suffers from overfitting and lack of generalizability, especially when the sample size of training data is limited or when there is instability in the data-generating process (DGP).

In this paper, we propose a novel and general framework to combine the economic insights from structural models with the flexibility of reduced-form machine learning models. A misspecified structural model can still help guide and regularize the training of a reduced-form model and alleviate the overfitting problem. Moreover, the structural model restrictions can often be readily extended outside the training data boundaries, which could help enhance the generalizability of the reduced-form model.

Our framework is based on transfer learning. Generally speaking, transfer learning aims at transferring knowledge from different but related contexts (referred to as the "source domain") to improve the performance in a new setting (the "target domain"). In our framework, the source domain is generated by the structural model while the empirical data reside in the target domain, and transfer learning allows us to combine the information from the structural model and real data, without treating the potentially misspecified structural restrictions as

[1] See, for example, "The End of Theory: The Data Deluge Makes the Scientific Method Obsolete," Wired Science, 2008, where the author offers an updated version of Box's quote, *"All models are wrong, and increasingly you can succeed without them."*

hard constraints. The implementation is straightforward: we first construct a neural network representation of the structural model by training on the synthetic data generated by the structural model, and then fine-tune the network using empirical data.

The source domain learning step benefits from two features: 1) the training set of synthetic data can be arbitrarily large (limited only by computing budget); 2) the signal-to-noise ratio of synthetic data is typically high. Coupled with the expressivity of neural networks (formally established through the universal approximation theorem; see e.g., Hornik, Stinchcombe, and White, 1989; Hanin, 2019), it means that it is relatively easy to obtain a neural network that accurately inherits the economic restrictions implied by the structural model (Chen, Didisheim, and Scheidegger, 2023). The fine-tuning step in the target domain then allows the neural network to "move away" from theory and incorporate information from empirical data. Specifically, it aims to minimize the empirical loss based on empirical data, beginning the network parameter search with values obtained from the source domain and proceeding with a small learning rate (i.e., updating parameters in small steps).

The sequential nature of transfer learning has two implications. First, the resulting model no longer satisfies the structural restrictions exactly, especially when these restrictions appear inconsistent with empirical data. It is worth noting that our objective in the target domain is to learn the true DGP. This approach differs from those that train a machine learning model by imposing theoretically derived constraints, the most notable example being physics-informed neural networks (Raissi, Perdikaris, and Karniadakis, 2019), which impose governing equations implied by physical laws when training the networks. Since economic models are arguably more susceptible to misspecification than their physical counterparts, imposing these misspecified restrictions as constraints (or equivalently, adding penalties in the loss function) can introduce more biases into the results.

Second, the transfer learning model does not simply average the information embedded in synthetic data and empirical data. Heuristically, one can view training the neural network on the synthetic data as providing an informative prior for the learning on empirical data in the second stage. This resembles Bayesian vector autoregressions (BVAR), which imposes informative priors to help estimate the VAR parameters; in particular, DeJong, Ingram, and Whiteman (1993); Ingram and Whiteman (1994); Del Negro and Schorfheide (2004) propose to derive the prior from a DSGE model. Besides its significantly enhanced ability to capture nonlinearity, the transfer learning model also differs from the Bayesian framework in how it determines the strength of the "prior." While increasing the amount of synthetic data in the source domain can help the neural network capture the structural model restrictions more accurately, this step only provides a starting point for training in the target domain. How much the network parameters move away from this starting point in the target domain depends on the gap between the structural model and the true DGP, the sample size of empirical data, as well as the hyperparameters for target domain training, including the learning rate and epochs. In particular, we show that fine-tuning (with a learning rate that is orders of magnitude smaller than in a standard deep learning model) is key to retaining the information from the structural model. This contrasts with the DSGE-VAR model in Del Negro and Schorfheide (2004), where the ratio of the sample sizes of synthetic to actual observations determines the strength of the prior. This ratio is a hyperparameter that needs to be tuned in the data.

Besides enhancing forecasting power, the transfer learning model can inform us about the limitations of a structural model. The potential features one might want to use for forecasting can go far beyond the relevant state variables in a structural model, which are often kept to a small number to preserve tractability and transparency.[2] To deal with this discrepancy, we draw the complete set of features from a multivariate uniform distribution when generating the synthetic data, but compute the structural model predictions using only the subset of features required by the model. Feature importance analysis can then help identify any features that are ignored in the source domain but useful in the target domain, which would point to directions to improve the structural model.

Finally, the transfer learning framework can be used to compare models in a new way. Traditionally, we often assess a model's performance in isolation, for example, based on the degree of fit with the data or the predictive accuracy. The transfer learning model allows us to compare structural models based on how complementary they are with the empirical data,

[2] For example, in the Black-Scholes option pricing model, the relevant features include the stock price, strike price, time-to-maturity, risk-free rate, dividend yield, and volatility. However, other features that could be empirically relevant include past returns on the option and the underlying asset, put-call ratio, trading volume, bid-ask spreads, etc.

which is a new and relevant perspective in the age of big data. A structural model that is less accurate when making forecasts on its own could be more effective in guiding us to learn from the empirical data.

As an example application, we apply the transfer learning framework to option pricing. We use the Black-Scholes model (Black and Scholes, 1973) to generate synthetic data in the source domain. Despite ample evidence of the empirical limitations of this model, we choose it for its simplicity as well as to illustrate the point that clearly misspecified structural models can still be helpful in the transfer learning framework.

We train a feedforward network with 16 hidden layers, 22 neurons for each layer (with a total of over 7,800 trainable parameters). We use multi-target learning in the source domain to capture the structural restrictions, with the loss function taking into account dollar pricing errors, option delta, and vega. In the target domain, the loss function is based entirely on dollar pricing errors. We train and evaluate the model in the target domain using a rolling window approach (source domain training only needs to be done once). Each iteration employs a training set comprising three months of empirical data, followed by a test set consisting of data from the subsequent three months. We then compare the performance of the transfer learning model (referred to as TL) against that of a deep neural network with identical architecture but does not have information from the structural model (referred to as DL), as well as the Heston model (Heston, 1993), which extends the Black-Scholes model by adding stochastic volatility.

Out of sample, the TL model significantly outperforms both the DL and the Heston model in pricing accuracy from 2001Q1 to 2023Q1. The average of the quarterly median absolute errors in Black-Scholes implied volatility (BSIV-MAE) for the TL model in the entire sample is about 32.4% of that for DL and 9.3% of that for the Heston model. In the cross section, the performance advantage of TL relative to DL is more significant for median and long time-to-maturity, out-of-the-money, more liquid options (lower bid-ask spreads or higher trading volume). In the time series, the performance advantage of TL is larger when market volatility is elevated (measured by average VIX over the past 60 days), when market volatility jumps up, as well as when the the new observation appears more uncommon relative to the training data (measured by the Mahalanobis distance between the input features of the new observation and the training sample). These results are consistent with the interpretation that structural model restrictions become more helpful when standard machine learning methods struggle to generalize beyond the confines of training data or deal with the instability in the underlying DGP.

In addition to higher average accuracy, the TL model also demonstrates greater stability compared to the DL model. This stability is manifested in two aspects. First, the outliers of pricing errors for the TL model are less extreme. For example, the 90th percentile of out-of-sample absolute BSIV errors for the TL model is 34% of that for DL over the full sample. Second, neural networks have embedded randomness stemming from the training method (stochastic gradient descent), initialization of network parameters, as well as the training sample. In this aspect, the results of the TL model demonstrate much smaller variation than the DL model, mainly due to the fact that the TL model uses theory-implied initialization plus a small learning rate in the target domain. As a result, the advantage of the TL model becomes more pronounced when the sample size of empirical data is smaller. For example, when we reduce the training sample to 10% of its original size in each 3-month period, the performance of the DL model deteriorates substantially, while that of the TL model remains relatively stable.

Since the TL model is trained with both synthetic and empirical data, one may wonder whether the DL model can match TL if trained on more data. We raise the size of training data for the DL model in two ways: i) by pooling the synthetic and empirical data; ii) by switching the training set from (3-month) rolling window to expanding window (starting from 2000Q4). We show that data pooling does not meaningfully improve the DL performance. When the two types of data originate from markedly different distributions, adding a large amount synthetic data into the training set is more detrimental to DL's performance due to the biases introduced, which outweigh the benefits of variance reduction. The expanding-window approach does help the DL model, but the gap with the TL model remains significant.[3] This exercise highlights a key challenge with forecasting in the time series. When the DGP can change over time, observations from the distant past become less relevant. Thanks to its

[3] The out-of-sample BSIV-MAE for the DL model under expanding window falls to 62% of that for the original DL under rolling-window training.

ability to achieve stable performance on small datasets, the TL model can be more adaptive to potential structural breaks.

We also examine two alternative approaches for bringing in information from the structural model. In the first approach, we mimic the physics-informed neural networks and impose the structural model restrictions as constraints when training the DL model. This is done by making the average pricing errors relative to the Black-Scholes model a penalty in the loss function, with the weights tuned in the data. In the second approach, we fit a linear model to the pricing errors of the Black-Scholes model and use it boost the structural model's performance. Neither approach can match the TL model's performance. Different from the first approach, the TL model uses the likely misspecified structural model to derive an informative initialization rather than treating them as constraints. The boosting method also starts with the structural model. However, the training on the model errors can suffer from the same over-fitting problem.

Finally, it is worth noting that our transfer learning framework is quite general. It can be applied to any forecasting problem where a suitable structural model is available for generating synthetic data. It is also not tied to neural networks (for example, the two-step procedure can also be applied to VARs), although the benefit of incorporating structural model restrictions is likely more pronounced when the machine learning model has high degrees of freedom relative to the size and representativeness of the training sample.

Related literature There is a fast-growing literature that applies machine learning methods to economics and finance. Among the early contributions are Hutchinson, Lo, and Poggio (1994), Chen and White (1999), and Chen and Ludvigson (2009). Recent works have shown the rich benefits of machine learning techniques in both linear (see e.g., Kelly and Pruitt, 2013; Rapach and Zhou, 2013; Chinco, Clark-Joseph, and Ye, 2019; Kozak, Nagel, and Santosh, 2023) and nonlinear settings (see Gu, Kelly, and Xiu, 2020; Freyberger, Neuhierl, and Weber, 2020; Bianchi, Büchner, and Tamoni, 2021; Cong et al., 2022; Bali et al., 2023; Chen, Pelger, and Zhu, 2024; Campello, Cong, and Zhou, 2024, among others).

It is quite intuitive that one should try to take advantage of domain knowledge when applying these powerful methods. Indeed, it is standard practice to use economically-motivated features and feature engineering when building models for forecasting. Several studies have further exploited theoretically motivated restrictions. For example, Garcia and Gençay (2000) show that the performance of neural networks can be improved by exploiting the homogeneity implied by the option pricing formula. The no-arbitrage condition has been used by Feng et al. (2023), Chen, Pelger, and Zhu (2024), Bryzgalova, Pelger, and Zhu (2023), and Bryzgalova et al. (2024) to help with estimating factor betas, detecting weak factors, or estimating the conditional SDF. Our contribution is to propose a general transfer learning framework that uses potentially misspecified structural model restrictions to guide the learning on empirical data.[4]

There are several alternative approaches for incorporating theoretical restrictions into econometric models. First, a strand of Bayesian Vector Autoregression models integrate prior information based on economic theories, with the goal of reducing overfitting in high-dimensional models through coefficient shrinkage. These prior can be statistically motivated (as in the case of the Minnesota prior, see Doan, Litterman, and Sims, 1984; Litterman, 1986) or derived from a structural model (see DeJong, Ingram, and Whiteman, 1993; Ingram and Whiteman, 1994; Del Negro and Schorfheide, 2004). Second, in physics-informed neural networks (Raissi, Perdikaris, and Karniadakis, 2019), structural model restrictions are treated as hard constraints and incorporated into the loss function. We face bias-variance tradeoffs when these structural restrictions are likely misspecified, which could make transfer learning the more suitable approach in such situations. Third, Almeida et al. (2023) propose to boost parametric option pricing models by fitting a neural network on the model-implied pricing errors. We investigate a simplified version of this boosting approach, replacing the neural network with a linear model.

2 Methodology

At its core, transfer learning is based on the idea that knowledge gained from solving one problem in the source domain can be transferred and applied to solve a different but related

[4] Transfer learning is widely used technique in machine learning. Important applications include computer vision and large language models. For a comprehensive survey on this topic, see Zhuang et al. (2020).

problem in the target domain. The knowledge obtained from the source domain not only makes it easier to train a model in the target domain, where data could be limited but also improves the model's performance. In our setting, we treat the extraction of information from an economic model as the source task. The information from the model is then used to aid the target task, in which we learn directly from actual data.

2.1 The Transfer Learning Framework

Denote by $\mathcal{X}$ and $\mathcal{Y}$ the input and target space, respectively, with unknown probability distribution $\mathbb{P}_{(\mathcal{X},\mathcal{Y})}$ on some σ-algebra of $\mathcal{X} \times \mathcal{Y}$. We want to predict $y \in \mathcal{Y}$ using a function of potential features $f(x)$ for $x \in \mathcal{X}$. The standard data-driven approach is to search for function $\widehat{f}$ from a given function family $\mathcal{H}$ (e.g., the set of neural networks) that minimizes the empirical risk for some loss function $\mathcal{L}(f(x), y)$ over a given training set $S = ((x_i, y_i))_{i=1}^{m}$. The resulting $\widehat{f}$ involves randomness stemming from the training sample as well as the optimization procedure (e.g., random initialization of search and stochastic gradient descent).

Next, consider a theoretical model that imposes restrictions between x and y. In some cases, these restrictions could result in a fully-specified joint distribution for x and y as $\mathbb{Q}_{(\mathcal{X},\mathcal{Y})}$. Alternatively, they could just be about the conditional expectation of y,

$$\mathbb{E}^{\mathbb{Q}}[y|x] = g(x), \tag{1}$$

where the expectation is taken under the model-implied probability measure $\mathbb{Q}$. Notice that the model may only involve a subset of x as its state variables; at the same time, it may involve hidden states h and parameters θ, which are not directly observable. However, as we will explain shortly, Eq. (1) is without loss of generality in that we could accommodate for h and θ through filtering or conditioning down. Finally, the theoretical model could be misspecified in the sense that $\mathbb{Q}$ is inconsistent with $\mathbb{P}$.

We propose a transfer learning framework that uses the theoretical model to guide the training of the machine learning model. Although the framework we propose is more general, to fix ideas, we will focus on neural networks in the remainder of this paper. A neural network with L layers is denoted as $F(L; \sigma_1, \sigma_2, \cdots, \sigma_L; W_1, W_2, \cdots, W_L)$, where σ_i and W_k are the activation function (a non-linear function that is applied element-wise) and weights for the i-th layer, respectively. The training of the neural network results in weights $\widehat{W}^* = [\widehat{W}_1^*, \cdots, \widehat{W}_L^*]$.

The transfer learning framework involves two steps. First, in the source domain, we generate a set of synthetic data $\widetilde{S} = ((\tilde{x}_i, \tilde{y}_i))_{i=1}^M$. Here, we either draw $(\tilde{x}_i, \tilde{y}_i)$ from the model-implied distribution $\mathbb{Q}_{(\mathcal{X},\mathcal{Y})}$, if it is fully specified, or we first draw $\tilde{x}_i$ and then set $\tilde{y}_i = \mathbb{E}^{\mathbb{Q}}[y|\tilde{x}_i]$. We then train the neural network $\widetilde{f}$ on $\widetilde{S}$ by minimizing a certain loss function. Let the resulting network weights from the source domain be $\widetilde{W}^*$. Notice that the empirical data set S is not used in the source domain. In the second step, we move to the target domain and train the network $\widehat{f}$, which has identical architecture as $\widetilde{f}$, on the empirical data S. Crucially, we use $\widetilde{W}^*$, the weights inherited from the source domain, to initialize the training for $\widehat{f}$, and the learning rate used in the target domain is significantly smaller. For this reason, we refer to the training in the target domain as fine-tuning. Next, we discuss the two steps in more detail.

Source Domain In the source domain, we are trying to obtain a neural network representation of the structural model restrictions, as summarized by Eq. (1). As Chen, Didisheim, and Scheidegger (2023) show, neural networks are well-suited for this task thanks to their expressivity. Specifically, the universal approximation theorem states that a neural network that is sufficiently wide or deep can approximate a smooth function arbitrarily accurately (see Hornik, Stinchcombe, and White, 1989; Hanin, 2019). Moreover, this task also benefits from the fact that the size of the synthetic dataset can be arbitrarily large (limited only by computing budget), and the signal-to-noise ratio of synthetic data is typically high. These two properties help reduce the risks of over-fitting in the source domain.

To generate the synthetic dataset $\widetilde{S}$, we start by specifying the ranges for the relevant state variables (including both observable and hidden states) and parameters for the model, then draw their values from the multivariate uniform distribution, and finally use Eq. (1) to evaluate the model-implied conditional expectation of y.

Before proceeding any further, we need to ensure that the input vectors in the source domain and target domain, $\tilde{x}$ and x, match each other in dimension. The issue arises from model-irrelevant features and those model states and parameters not directly observable in the actual data. Economic models are typically designed to be parsimonious, with a relatively small number of states. In contrast, a main attraction of machine learning models is that they help us look for information in a large number of potential features empirically, which can go far beyond those considered in an economic model. This can result in a subset of the features in x not being used by the model. For these features deemed by the model as irrelevant, we randomly draw their values and include them as part of the inputs $\tilde{x}$ for the synthetic data.

For those hidden states and the parameters that are required by the economic model to determine the conditional expectation of the target variable y but not directly observable in the empirical data, one possibility is to filter them in the empirical data using the structural model restrictions. We can then augment the input vector x with these filtered values. An alternative solution is conditioning down the expectation of the target variable y by integrating over some hierarchical prior distribution of the parameters and the model-implied distribution of the hidden states. It is worth noting that the neural network trained in the source domain, if trained accurately, can help accelerate the filtering process significantly (see Chen, Didisheim, and Scheidegger, 2023).

In order to train the neural network on the synthetic training set, we need to specify a loss function. Naturally, we would like to minimize the prediction errors of the network relative to target variable, $\tilde{y}$, which can be summarized by the MSE or MAE over the training set. Different from a standard supervised learning setting, where the DGP is unknown, the structural model may provide additional restrictions that we can exploit to further improve the accuracy of the neural network. Consider the example of training a physics-informed neural network for a heat equation. As Raissi, Perdikaris, and Karniadakis (2019) show that, in addition to matching the initial and boundary data, the residuals of the heat equation can be part of the loss function, which is referred to as physics-informed loss. Similarly, we can add economics-informed loss to the source domain training.

Specifically, in the source domain, we train the neural networks by

$$\widetilde{W}^* = \arg\min_{\widetilde{W}} \left(\lambda_0 L_0 + \sum_{j=1}^{K} \lambda_j L_j \right) \tag{2}$$

where

$$L_0 = \sum_{i=1}^{M} w_{0i} \left| F(L; \sigma_1, \cdots, \sigma_L; \widetilde{W}_1, \cdots, \widetilde{W}_L)(\tilde{x}_i) - g(\tilde{x}_i) \right| \tag{3}$$

The first term of the loss function is L_0, the weighted mean absolute error for the network's prediction of $\tilde{y}_i$, where the weights are w_{i0}. The additional economics-informed losses are given by L_i. For example, we can derive from Eq. (1) the gradient of $g(x)$ and use it as part of the loss,

$$L_j = \sum_{i=1}^{M} w_{ji} \left| \frac{\partial F}{\partial x_j}(\tilde{x}_i) - \frac{\partial g}{\partial x_j}(\tilde{x}_i) \right|. \tag{4}$$

The training of the neural network starts with randomly initialized values of matrix $\widetilde{W}$.

Target Domain In the target domain, we fine-tune the network obtained from the source domain using empirical data. This means that we initialize the network with weights $\widetilde{W}^*$ and use a substantially lower learning rate in the target domain than in the source domain. Specifically,

$$\widehat{W}^* = \arg\min_{\widehat{W}} \sum_{i=1}^{m} w_i \left| F(L; \sigma_1, \cdots, \sigma_L; \widehat{W}_1, \cdots, \widehat{W}_L)(x_i) - y_i \right|, \tag{5}$$

with initial weights $\widetilde{W}^*$.

In the transfer learning literature, there are various ad-hoc approaches to design the network architecture with the hope of preserving some of the information from the source domain. For example, the partial fine-tuning method freezes the first $K < L$ layers of the source domain model and only updates the weights of the neurons after the Kth layer. Relative to full fine-tuning method in (5), this method improves computational efficiency because it updates fewer parameters and has a higher training speed. In our empirical exercise, we use the full fine-tuning method. However, it could be interesting to explore the frozen-layer K as a hyperparameter. Intuitively, the larger the gap between the source domain task and the target domain task, the more layers should be allowed to be adjusted.

2.2 An Application to Option Pricing

A large number of parametric option pricing models, starting with the Black-Scholes model, have demonstrated various degrees of empirical success. We use the option pricing example to demonstrate how the transfer learning framework can allow us to incorporate the information from these structural models into a highly flexible deep neural network.

We choose the Black-Scholes model as the structural model for the source domain. This choice is motivated by two main reasons. First, the Black-Scholes model is a seminal option pricing model that is arguably the least exposed to any "look-ahead" biases; the later generations of option pricing models extended the Black-Scholes model by observing its limitations in the actual data. Second, we aim to demonstrate that a simple and misspecified structural model can still be useful in the transfer learning framework.

The inputs of the pricing model are: $x_i = (S_i, K_i, T_i, vol_i, d_i, r_i, \mathcal{Z}_i)$, where the first six, stock price S_i, strike price K_i, time to maturity T_i, volatility vol_i (as proxied by the VIX index lagged by 1 day), dividend yield d_i, and risk-free rate r_i, are features required by the Black-Scholes model; the additional features, as represented by $\mathcal{Z}_i$, include historical volatility of the S&P 500 returns, short and medium-term momentum of both the S&P 500 index return and the specific option return, abnormal trading volume for the option, put-call ratio, and the S&P 500 earnings-price ratio.[5] We use *LeakyRelu* as the activation function in this exercise. Other activation functions will be examined as part of the robustness checks.

The diagram in Figure A.1 illustrates the process of applying transfer learning to option pricing, including the training in both the source and target domains. The source domain training only needs to be performed once. In the target domain, with different training samples (as we retrain the model over time), we simply fine-tune the same source domain model with the new data.

In the source domain, we train the neural networks as:

$$\widetilde{W}^* = \arg\min_{\widetilde{W}}(\lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3), \tag{6}$$

[5] The complete list of features are in Appendix B.

where

$$L_1 = \sum_{i=1}^{N} \left| \frac{1}{|\delta_i| + \epsilon_c} \Big(F(L; \sigma_1, \cdots, \sigma_L; \widetilde{W}_1, \cdots, \widetilde{W}_L)(\tilde{x}_i) - g(\tilde{x}_i) \Big) \right| \tag{7}$$

$$L_2 = \sum_{i=1}^{N} \left| \frac{\partial F(\tilde{x}_i)}{\partial S} - \frac{\partial g(\tilde{x}_i)}{\partial S} \right| \tag{8}$$

$$L_3 = \sum_{i=1}^{N} \left| \frac{\partial F(\tilde{x}_i)}{\partial vol} - \frac{\partial g(\tilde{x}_i)}{\partial vol} \right| \tag{9}$$

with randomly initialized initial values of matrix $\widetilde{W}$. Besides the weighted mean-absolute dollar pricing error, L_1, we also add two economics-informed losses, one on option delta, L_2, and the other on option vega, L_3.

We weigh the pricing error by $1/|\delta|$ to ensure the TL and DL pay sufficient attention to OTM options, whose dollar prices have significantly smaller magnitudes than those of ATM and ITM options. ϵ_c is a small constant to avoid exploding weights for DOTM options. Without this term, the contract's pricing error in NN back-propagation with a delta closest to 0 could dominate the entire loss function. Since the network weights are initialized randomly, the pricing errors could be large during the earlier epochs. Thus, numerical stability is important.

In the source domain, we know the true model without any noise, so we can run a large number of epochs over the training data and choose relatively large learning rates with less concern about overfitting. Furthermore, the model is designed to ignore the information of input that is not the SDE model input.

The target domain aims to transfer information learned from the source domain to the domain concerning empirical data. We set the loss function to be the weighted mean absolute dollar pricing errors, where the weights are again inversely related to option delta:

$$\widehat{W}^* = \arg\min_{\widehat{W}} \sum_{i=1}^{m} \left| \frac{1}{|\delta_i| + \epsilon_c} \Big(F(L; \sigma_1, \cdots, \sigma_L; \widehat{W}_1, \cdots, \widehat{W}_L)(x_i) - P_i \Big) \right| \tag{10}$$

with initial network weights $\widetilde{W}^*$ inherited from the source domain model.

It is worth noting the contrast between the data environment in the source and target domain. Data in the latter case will have much lower information-noise ratio, and the training sample size can be quite limited. We train the model with a low learning rate and use the standard early-stopping criteria to limit the number of epochs. These measures not only guard against over-fitting risks, but also help retain more of the information from the structural model.

2.3 Residual Learning and Skip Connect

We use the residual learning method to avoid the Vanishing Gradient Problem (VGP). The method allows us to make the network deep enough to implement transfer learning algorithms. In the application of deep learning to asset pricing, a natural question is, do deeper networks generalize better? In the research of computer science, the answer to the above question is no. The most immediate problem is the phenomenon of vanishing gradients and exploding gradients: the training of neural networks relies on backpropagation. If the depth of the neural network is too large, the neurons in the earlier layers of the neural network may obtain gradients close to 0 or abnormally large in backpropagation. As revealed by experiments in He and Sun (2015) and shown in Srivastava, Greff, and Schmidhuber (2015) , when the network depth is too deep, the model performance will decrease with the depth of the neural network, which is called the degradation problem. The degradation problem is not caused by overfitting but by the structure of the neural network itself and the characteristics of the training method.

It should be noted that some AI tasks of asset pricing need to fit highly nonlinear equations, such as the fitting of option pricing equations that should be performed in this paper. This means that we need a deeper network with higher expressive power, so the degradation problem of the neural networks is an important problem to be solved. He, Zhang, Ren, and Sun (2016a) proposed a method called the residual learning method to solve the degradation problem. Their network structure allows neural networks to enjoy the benefits of out-of-sample fitting capabilities produced by deeper networks without facing the problems of deep networks. The core idea of this method is to add a never-closed Shortcut Connections to the neural network, which is equivalent to learning residuals.

Mathematically, let a represent the input to a residual block, and $G(a)$ denotes the output from a series of operations applied to a, potentially including processes like convolution, activation, and normalization. A basic residual block can then be reformulated as:

$$b = G(a, \{V_j\}) + a \tag{11}$$

Here, b is the output of the residual block, $G(a, \{V_j\})$ signifies the transformation applied to the input a, with $\{V_j\}$ representing the set of parameters used in the transformation. The a term serves as a "shortcut" or "skip" connection, which is directly added to the transformation's output. The advantage of this structure is that even in very deep networks, residual connections help direct the gradient flow to earlier layers, thus improving the training performance of deep networks. We have drawn inspiration from the concepts of ResNet (He, Zhang, Ren, and Sun, 2016a,b), but unlike prior applications in computer vision, we have adapted these ideas to suit the unique characteristics of our task by incorporating skip connections into our deep network. The specific structure can be found in Figure A.1. Detailed network parameters are provided in Appendix A.

3 Empirical Results

Implementation of the above framework on a large and comprehensive panel of index options in this section clearly exhibits remarkable performance, in both pricing tasks. The model is trained by using 3 months data and tested by using 3 months data right after; it is re-calibrated every 3 months.

To compare our model with the traditional measures, we develop two types of models as benchmarks. In particular, the first model is the stochastic volatility model based on the Heston model. The model's parameters are estimated by minimizing the mean absolute pricing error the day before, which is a standard practice in empirical research.

The second model is a classical deep learning model but without transfer learning technique embedded. Note that the classical deep learning model is trained under the same hyper-parameters with transfer learning, including stopping strategy, train set size, rolling window length, activation function per neural unit, and the structure of the neural network, with our transfer learning model, except for the learning rate, which differs between the two. For fairness, both the transfer learning (TL) and deep learning (DL) methods use their respective optimal learning rates, which are determined through grid search based on prior knowledge.

3.1 Data

We sourced daily transaction data on S&P 500 index put options from OptionMetrics, covering the period from January 2, 2001, to March 31, 2023. Table 1 presents the descriptive statistics for this dataset, encompassing a total of 27,791,709 observations. The distribution of contract-level implied volatility was methodically calculated over time, resulting in various grouped classifications. To effectively categorize the dataset based on expiration dates, we employed 10-day and 60-day benchmarks, segmenting it into three distinct groups. Subsequently, the strike prices were divided into six categories, culminating in 18 specific subcategories of option samples. Notably, nearly half of these options are set to expire within a span of 10 to 60 trading days. The descriptive statistics related to implied volatility align remarkably with the well-recognized volatility smile phenomenon: contracts that are either in-the-money or out-of-the-money exhibit higher levels of implied volatility in comparison to at-the-money contracts.

3.2 Pricing Performance

The Black-Scholes implied volatility (BSIV) pricing error for option i at instance t is articulated as:

$$\tilde{\epsilon_{it}} = |\sigma(P_{it}; K_i, T_{it}, r_{T_{it},t}, S_t, d_t) - \sigma(\hat{P_{it}}; K_i, T_{it}, r_{T_{it},t}, S_t, d_t)|, \tag{12}$$

where the function $\sigma(\cdot; K_i, T_{it}, r_{T_{it},t}, S_t, d_t)$ maps prices to the implied volatility for day t and contract i. The interest rate $r_{T_{it},t}$ is obtained through linear interpolation of the risk-free rate curve corresponding to the respective period.

The aggregate pricing deviation of the model is encapsulated by the median absolute error (MAE) across all samples. The main reason for using the median rather than the mean as the error measure in this study is that the predicted option prices from the model may fall outside

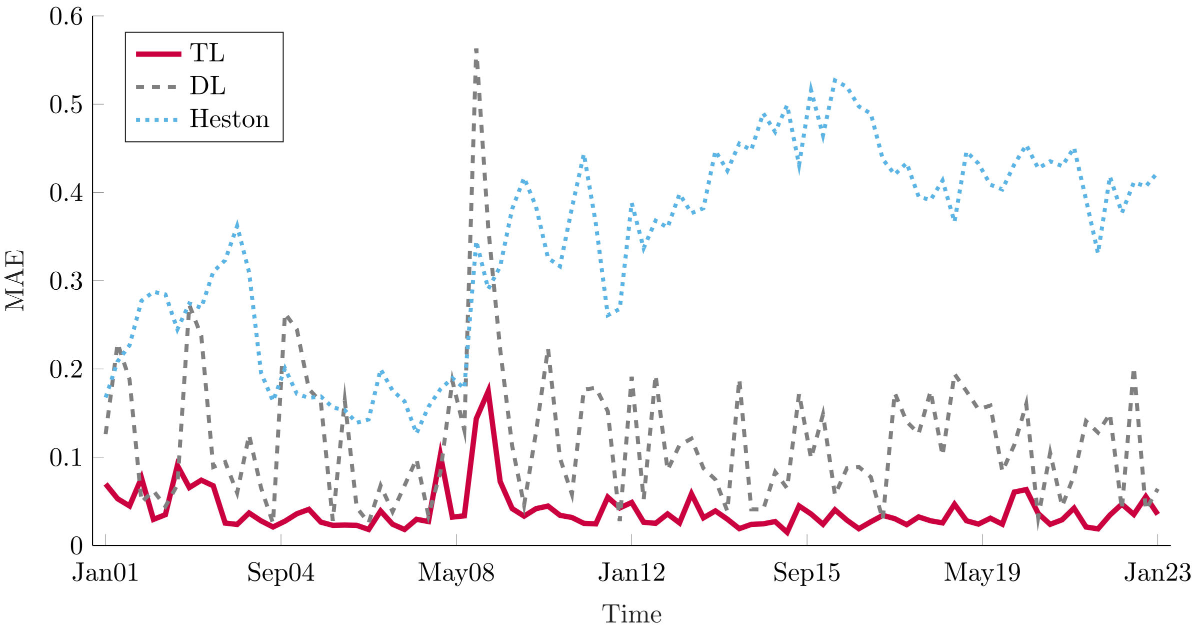

Figure 1: **Out-of-sample pricing errors.** This figure reports the out-of-sample median-absolute pricing errors (in terms of BSIV) for TL, DL, and the Heston model at quarterly frequency from 2001Q1 to 2023Q1. At the end of each quarter, the models are trained using data from the past 3 months, and the pricing errors are computed using data in the following 3 months.

the reasonable range of the Black-Scholes model, making it impossible to convert them into implied volatility. Alternatively, the predicted implied volatilities may be abnormally high, such as those exceeding 100000. These extreme values can disproportionately influence the model's mean error, whereas the median is not affected by such outliers.We derive σ_{it} utilizing the stochastic gradient descent technique. All models are all trained on a uniform dataset spanning from 2001 to 2023.

Instinctively, employing MAE as the performance metric could disproportionately weight in-the-money contracts more heavily than out-of-the-money ones, given that the contractual value of the former might substantially exceed that of the latter. To ensure a balanced representation of options across different moneyness levels, we adopt the discrepancy between implied volatility figures, as projected by models and actual market prices, as a surrogate for pricing errors.

Figure 1 illustrates that the transfer learning approach consistently exhibits lower MAEs across all periods compared to both the deep learning pricing network and the Heston model.

Undoubtedly, when contrasted with traditional models, the transfer learning paradigm distinctly benefits from amalgamating the economic insights of foundational structural models with the empirical knowledge derived from actual market data. The BSIV-MAE of transfer learning was reduced by 0.082 and 0.387 compared to deep learning and the Heston model, respectively, resulting in improvements of 67.6% and 90.8%. From a temporal perspective, the performance of the Heston model significantly deteriorated after 2008. Over the long term, deep learning models consistently outperform the Heston model; however, during the 2008 financial crisis, the errors exhibit significant peaks.

Fundamentally, the deep learning model operates by internalizing information from historical samples, operating under the presumption that future market behavior mirrors past trends. Consequently, deep learning might falter in periods of market tumult, like during the financial crisis, but excel in more stable conditions, explaining the observed error volatility. Conversely, the transfer learning network, given its source domain information driven by the model, remains relatively insulated from such market perturbations. This nuanced understanding will be further dissected in our subsequent attribution analysis.

3.3 Performance Stability

In addition to achieving higher average accuracy, the transfer learning (TL) model demonstrates significantly greater stability compared to the deep learning (DL) model, making it particularly suitable for complex and data-constrained financial applications.Transfer learning enhances the stability of deep learning in the following three ways: 1) it reduces the discrepancy between predictions for extreme outlier samples and the median level; 2) it is less affected by insufficient sample sizes; and 3) its performance is less influenced by the inherent randomness of the neural network.

The performance advantages of transfer learning (TL) in deep learning for option pricing are evident not only in terms of median accuracy but also in managing outliers effectively. To better understand this distinction, we analyzed the 90th percentile of pricing errors for both models in Figure 2. The results indicate that the TL model significantly reduces extreme errors compared to the deep learning (DL) model. Specifically, the 90th percentile of out-of-sample absolute BSIV errors for the TL model is only 34% of that observed for the

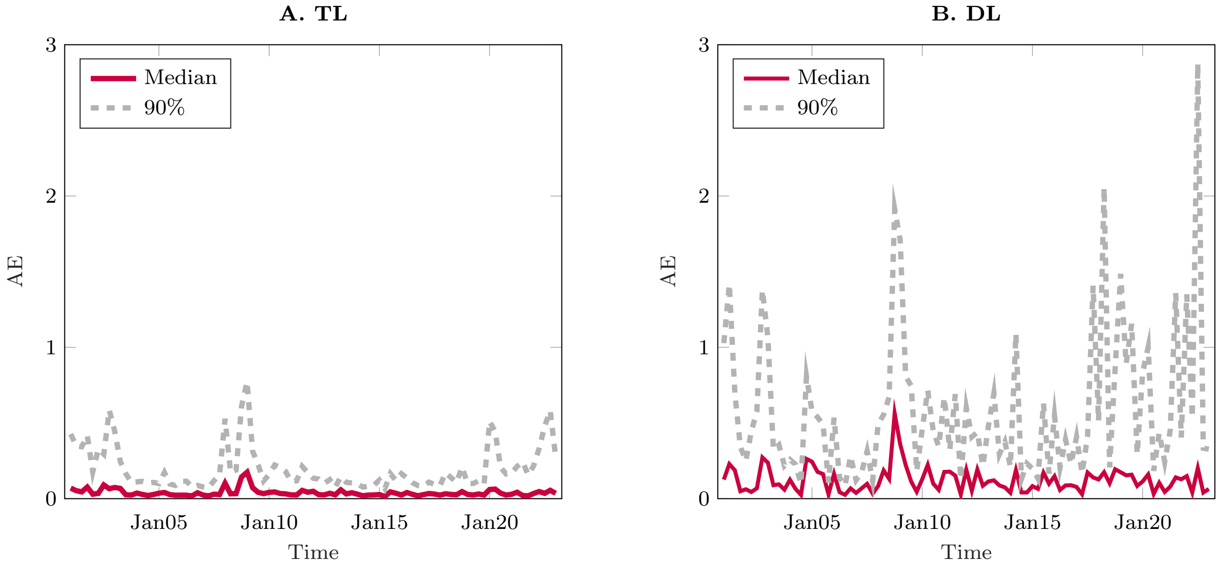

Figure 2: **Outliers for TL vs. DL.** In this figure, we compare the 90th percentiles of out-of-sample pricing errors (absolute errors in BSIV) for the TL and DL model.

DL model across the entire dataset. This highlights the robustness of transfer learning in mitigating the impact of extreme deviations, offering more reliable and consistent predictions even in challenging cases.

A key advantage of the TL model lies in its ability to effectively handle scenarios where data is scarce, a capability that is rooted in its pre-training process. Pre-training equips the model with foundational knowledge from a larger, potentially synthetic dataset, enabling it to perform well even when fine-tuning is conducted on smaller datasets. This is particularly relevant in financial domains such as option pricing, where empirical data may be limited due to market-specific conditions or temporal constraints. For example, when the training sample size is reduced to just 10% of its original volume as shown in Figure 3 for each three-month period, the DL model suffers a significant performance decline, with average IVMAE increasing from 0.122 to 0.234, exhibiting high sensitivity to data availability. In contrast, the TL model shows remarkable resilience, with average IVMAE increasing from 0.0394 to only 0.0629, maintaining relatively stable performance under these restrictive conditions. This small-sample learning capability is vital for real-world financial modeling, as it allows the TL model to deliver consistent results even when faced with fragmented or incomplete data.

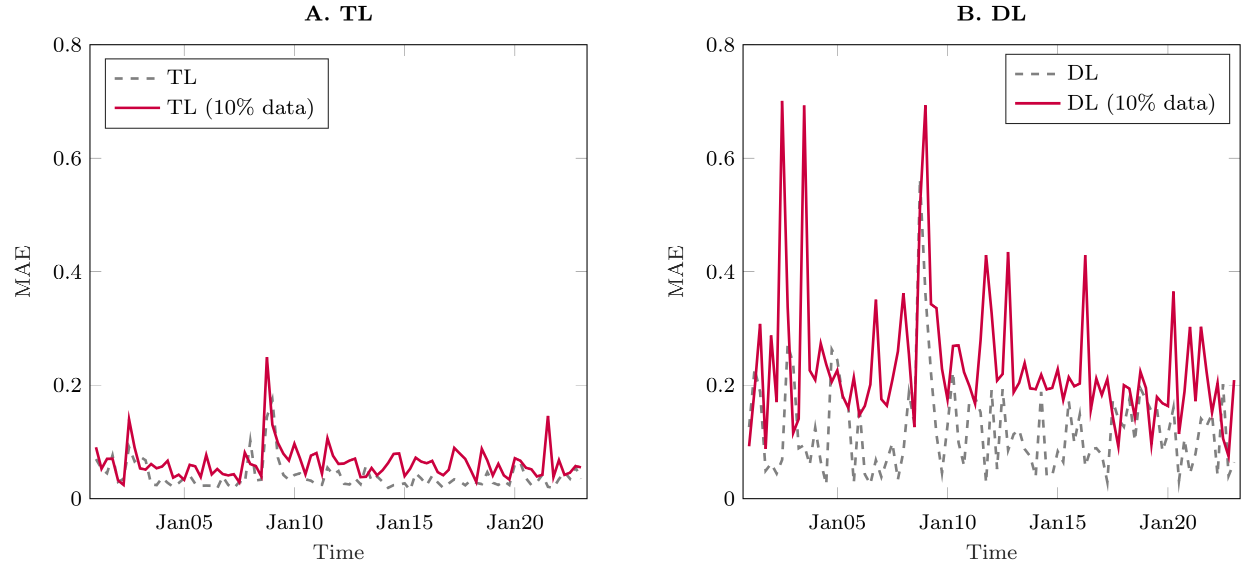

Figure 3: **TL vs. DL with small training sample.** In this figure, we compare the out-of-sample BSIV-MAE for TL against those of DL when the training sample is reduced to 10% of the original size.

Transfer learning is less affected by the inherent randomness of neural networks compared to deep learning. Neural networks are machine learning models that employ stochastic optimization algorithms. The solver for the backpropagation algorithm in neural networks is a variant of the stochastic gradient descent algorithm. From the box plots in Figure 4, it is evident that the confidence interval for the performance of transfer learning is 25% narrower than that of deep learning. This is because transfer learning only requires fine-tuning the parameters rather than restarting the training process. The uncertainty associated with fine-tuning is significantly lower than that of retraining from scratch.

Another crucial factor contributing to the stability of the TL model is its use of theory-implied parameter initialization, combined with a carefully controlled learning rate in the target domain. Neural networks are inherently influenced by randomness arising from the stochastic gradient descent algorithm, the random initialization of parameters, and variability in training data samples, which often leads to fluctuations in model performance. To assess this, we retrained both models using different random seeds. The results in Figure 4 showed that the TL model's errors remained remarkably consistent, demonstrating robust resistance to the effects of training randomness. In contrast, the DL model exhibited considerable

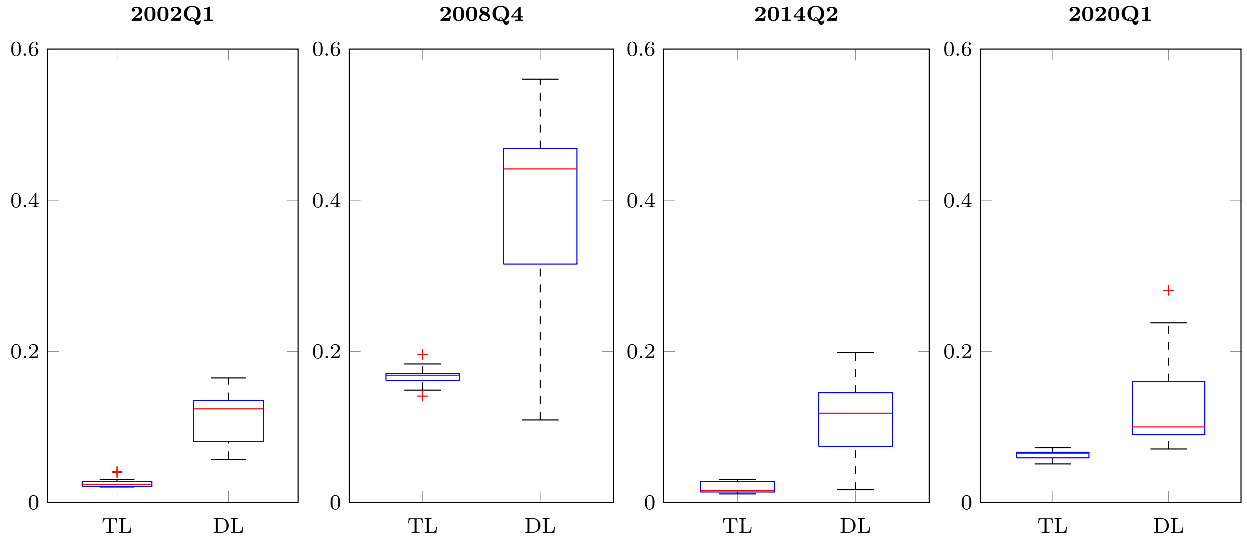

Figure 4: **Stability of TL vs. DL.** In this figure, we examine the distribution of out-of-sample BSIV-MAE for TL and DL at four points in time as the random seeds for network training change.

sensitivity to random seed changes, as evidenced by a substantially wider confidence interval in its performance metrics. This highlights the TL model's superior stability, making it a more reliable choice in applications where consistency is critical.

3.4 Feature Importance

In a neural network, the significance of an input, or its feature importance, is delineated as the incremental rise in the loss function on the training dataset when a specific feature is omitted within the context of transfer learning.To calculate feature importance by setting each feature to its mean and observing the change in loss, first compute the baseline loss, L_{baseline}, with the full feature set. For each feature X_i, replace it with its mean value across the dataset, keeping other features unchanged, and recompute the loss, $L_{\text{mean}(X_i)}$. The importance of X_i is the difference $L_{\text{mean}(X_i)} - L_{\text{baseline}}$. This measures how much the model's performance changes when X_i is replaced, capturing its contribution to predictive accuracy. Through feature importance analysis, our work contributes to the refinement of option pricing theory and offers insights for its theoretical development.We calculate feature importance within each training set, which corresponds to a 3-month data window, and then aggregate the

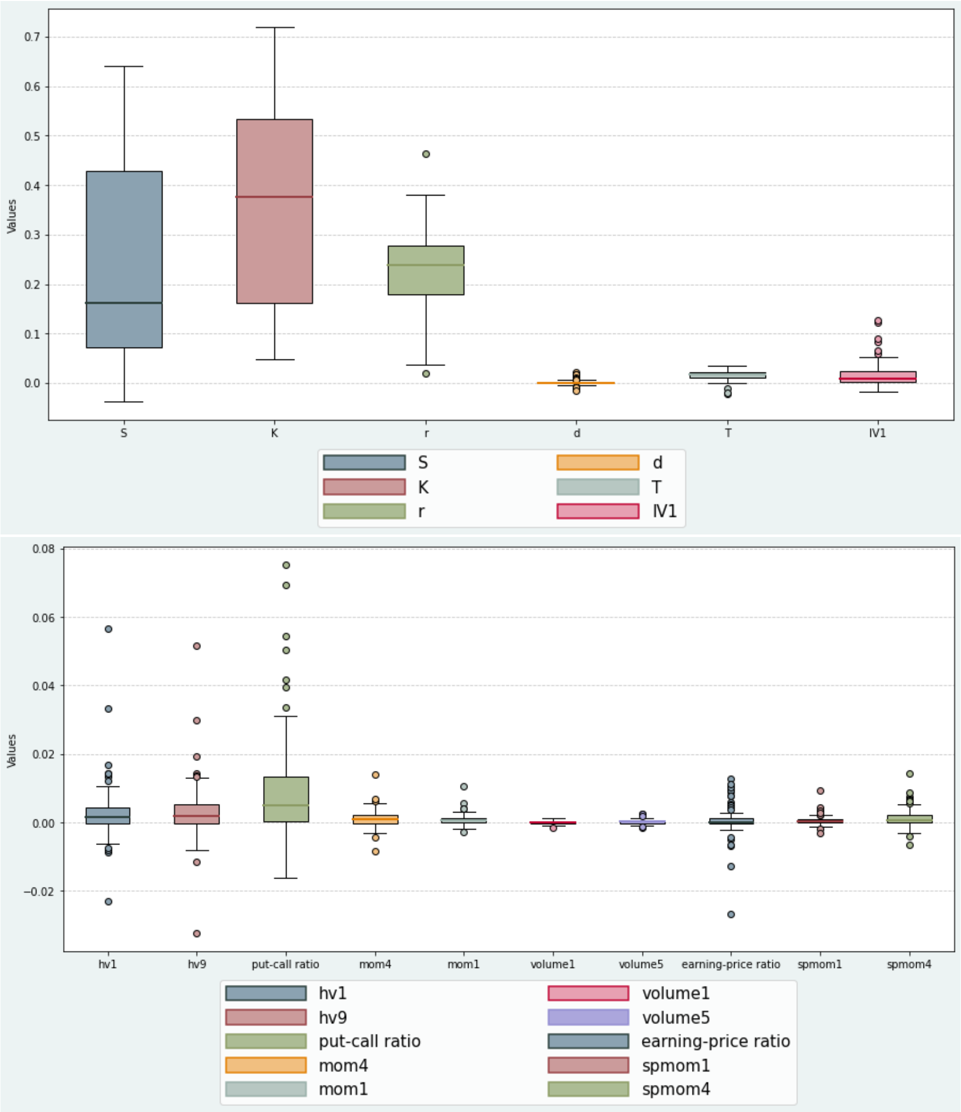

Figure 5: **Feature Importance.** The figure below shows a boxplot of feature importance for a transfer learning model. We have counted the feature importance of the same feature in each training set. This figure aims to visualize the comprehensive features of feature importance on different training sets. The outlier cut-off points of the boxplots were set to Q1-1.5IQR and Q3+1.5IQR. Data falling outside the cutoff point for outliers are marked separately.

results.

Even from the perspective of artificial intelligence and big data, the inputs of the Black-Scholes (B-S) model exhibit significant importance, demonstrating the robustness of classical financial theories and the economic significance of these inputs,according to Figure 5. Feature importance analysis shows that the underlying asset price (0.23), strike price (0.35), and risk-free rate (0.22) are the key variables within the B-S framework, far exceeding other variables in importance. This aligns with the B-S model, as these variables directly determine the present value of an option. The underlying asset price and strike price dictate the intrinsic value of the option, while the risk-free rate affects its time value through discounting.

Variables such as time to maturity (0.015) and implied volatility (IV, 0.017) also exhibit noticeable importance in the transfer learning model. This suggests that these variables retain explanatory power for option pricing even in complex, nonlinear environments. Specifically, time to maturity directly influences the decay of an option's time value. Comparatively, the dividend yield has the lowest importance (0.0012) among B-S variables, yet remains non-negligible when compared to other features. This could be because dividends reduce the upward potential of the underlying asset price, thus influencing the option's valuation.

Beyond verifying the importance of B-S model inputs, transfer learning reveals several non-traditional variables with significant effects on option pricing, offering new directions for extending option pricing theory. In the transfer learning model, the put-call ratio exhibits an importance of 0.009, indicating that market sentiment regarding bullish or bearish positions may play a role in option pricing. Classical option pricing theories primarily focus on equilibrium-based arbitrage-free pricing, while the put-call ratio reflects dynamic changes in market sentiment and trading behavior. A higher put-call ratio often signals increased concern about downside risks in the market, which may lead to a rise in implied volatility—a phenomenon frequently observed in empirical studies. This result suggests that option pricing is influenced not only by fundamental variables but also by market microstructure and investor behavior.

The higher importance of $mom4$ (medium-to-long-term momentum in options) and $spmom4$ (medium-to-long-term momentum in the S&P 500) challenges traditional theories. Classical theories suggest that option prices are determined primarily by current market information rather than historical price trends. However, the presence of momentum effects indicates that signals of trend continuation may exist, reflecting adjustments in market expectations of risk and opportunities.

Even with implied volatility included, historical volatility ($hv1$ and $hv9$) still demonstrates significant explanatory power for option prices, with importance scores of 0.0029 and 0.0027, respectively. This finding calls for a reevaluation of the relationship between implied volatility and historical volatility. While implied volatility reflects the market's expectations of future volatility, historical volatility captures the realized risk of the underlying asset over different time horizons, which may significantly influence pricing. For instance, in markets with substantial volatility changes, historical volatility might provide supplementary information to the model.

Transfer learning provides a crucial opportunity to refine and enhance existing structural models. By incorporating transfer learning, models can not only integrate traditional factors (such as the inputs in the Black-Scholes model) but also adapt to the inclusion of non-traditional variables, such as market microstructure and investor behavior, based on market fluctuations. This approach allows theoretical frameworks to dynamically adjust, better reflecting the complexities and uncertainties of real-world markets. Therefore, transfer learning not only contributes to the refinement of classical theoretical models but also offers new perspectives for future theoretical innovation, particularly in addressing the effects of market conditions, sentiment shifts, and investor behavior on pricing.

In summary, the feature importance analysis highlights the significant role of classical inputs, such as those in the Black-Scholes model, while also identifying the potential impact of non-traditional factors on option pricing. These findings reinforce the relevance of traditional theories but also point to the growing need to consider market microstructure and behavioral influences. Future research could focus on examining how these non-traditional features vary with market dynamics and develop new frameworks to account for these effects. Integrating behavioral finance into option pricing models, for instance, could enhance our understanding of the influence of market sentiment, momentum, and risk perception. Further investigation into the interplay between implied and historical volatility also remains a promising area for empirical study.

4 Analyzing Transfer Learning's Performance

4.1 When Does TL Perform Better?

To achieve dimensionless pricing errors, we convert both actual and predicted prices into implied volatility values. We regress the errors from both models on several explanatory variables, consistent with those presented in Table 2:

$$\tilde{\epsilon_{it}}^{DL} - \tilde{\epsilon_{it}}^{TL} = x_{it-1}\beta + u_{it} \tag{13}$$

Table 3 presents results computed across all test-set samples. Specifically, the models undergo training every three months, leveraging data from the preceding nine months to forecast the subsequent three months. We aggregate the data from each test set and proceed with the aforementioned attribution regression.

Andersen, Fusari, and Todorov (2017) argue that short-term options deviate from traditional option pricing models, raising the question of how the relative performance of deep learning (DL) and transfer learning (TL) models changes when applied to short-term options. The results presented in Table 3 indicate that the coefficients for 0-7 days and over 90 days are significantly positive at a 1% level. Specifically, the coefficients for the 0-7 days variable range from -0.0169 to -0.0201 and from -0.0118 to -0.0138 for 7-14. All these coefficients exhibit absolute t-statistics greater than 70. In contrast, for options with maturities longer than 14 days, the performance gap between TL and DL widens in favor of TL.This pattern can be attributed to the unique characteristics of short-term options, which exhibit greater deviations from the assumptions and structural information embedded in the source domain model. Short-term options are highly sensitive to short-lived market dynamics, rapid shifts in volatility, and idiosyncratic factors that are less effectively captured by models pre-trained on broader, long-term data patterns. As a result, the performance gap between TL and DL models narrows on this subset of data.

In transfer learning, the source domain draws from simulated data produced by the benchmark model. Consequently, transfer learning is likely to outperform deep learning in terms of highly nonlinear data. In the regression, we also control the variables about moneyness[6]. The coefficients for assets corresponding to all moneyness levels, except at-the-money (ATM), are positive. This indicates that the performance gap between transfer learning (TL) and deep learning (DL) widens for assets outside the ATM category compared to those at ATM. Notably, this widening is asymmetric: the gap is more pronounced for out-of-the-money (OTM) contracts, which exhibit higher nonlinearity, whereas the gap is smaller for in-the-money (ITM) contracts. This asymmetry highlights TL's superior capability in modeling complex nonlinear functions, making it particularly effective for pricing OTM options, where traditional models and purely data-driven approaches often struggle to capture the intricate dynamics.

Moreover, we find that the coefficient for bid-ask spread is significantly negative, while the coefficient for volume is significantly positive. This indicates that the performance gap between transfer learning (TL) and deep learning (DL) widens for assets with better liquidity. The potential reason for the reduced performance of transfer learning on low-liquidity assets may be that these assets, due to their lower trading activity, are more likely to deviate from the assumptions of the Black-Scholes model. The results suggest that the practical value of TL extends beyond the improvement indicated by its average error reduction compared to DL. Active, highly traded assets are more critical to market participants than illiquid ones, making TL's superior performance on these assets particularly valuable. This enhancement underscores the broader applicability of TL in scenarios where pricing accuracy for liquid and frequently traded assets is crucial, as these assets often drive market dynamics and investment decisions.

The superior performance of transfer learning over traditional deep learning can also be attributed to some other factors. Firstly, the coefficient for market volatility is notably positive with 1% statistical significance, ranging from 0.0719 to 0.117, suggesting that as

[6] We derive our moneyness calculation from Andersen, Fusari, and Todorov (2017). These authors contend that utilizing the simple ratio of the strike price to the underlying price as a proxy for moneyness is inherently biased. Instead, they put forth the following definition for moneyness:

$$m = ln(\frac{K}{H_T})/(\sqrt{T} * IV_{atm,T}) \tag{14}$$

Given that H_T represents the forward strike price with an expiration time of T, and $IV_{atm,T}$ denotes the implied volatility of the option whose strike price is closest to H_T

market volatility escalates, the disparity between the pricing predictions of the transfer learning and deep learning networks widens. In volatile conditions, the capital market might experience increased shocks, potentially influencing short-term derivatives pricing conventions. Being a purely data-driven model, the deep learning pricing network struggles to assimilate the nuances of these pricing adjustments based on limited data. In contrast, transfer learning inherently addresses this challenge by integrating established economic models.

Additionally, the variable $mIVchange$ further illustrates that the performance gap between transfer learning (TL) and deep learning (DL) widens during periods of heightened market volatility. $mIVchange$ is defined as the difference between the current day's market-implied volatility and the average implied volatility in the training dataset. Its coefficient is consistently positive and statistically significant across all relevant regressions, with values exceeding 0.1, highlighting its importance in capturing the dynamics of volatile market conditions.As a purely data-driven model, the deep learning pricing network struggles to adapt to rapidly changing market environments. Such environments often introduce non-linear patterns and structural shifts that exceed the capacity of DL models to generalize effectively from limited data. In contrast, the transfer learning approach inherently addresses this limitation by leveraging pre-trained knowledge rooted in established economic models, enabling it to maintain robust performance even in the face of significant market fluctuations. This suggests that the integration of domain-specific knowledge in TL not only enhances its stability but also improves its capacity to navigate and respond to complex market dynamics.

Envision a hypothetical situation. Suppose that for a specific pricing equation input, such as the risk-free interest rate indicator, its value predominantly ranged between 1%-2% in the years leading up to the testing set. In a standard window partitioning method, this would mean that only data within the 1%-2% range gets included in the training set. Deep learning models, when employed in rolling training, would have adeptly tailored their pricing equations to the prevailing low risk-free interest rates of previous years. However, during the period represented by the test set, an unexpected surge in inflation prompts the Federal Reserve to institute a rate hike, causing the risk-free interest rate to momentarily spike to above 5%. This abrupt shift presents a significant challenge for deep learning models. Under such circumstances, a purely data-driven deep learning model would likely struggle with projecting out-of-sample data. These algorithms rely on backpropagation to persistently modify neural weights based on historical data to model the relationship between inputs and outputs, such as option prices. This implies that the performance of deep learning is significantly influenced by the similarity between the training set and the test set. However, given our example, the deep learning model is conditioned throughout its training to assume that the risk-free interest rate predominantly oscillates between 1% and 2%. It remains largely uninformed about pricing dynamics outside this bracket. This limitation extends to other pivotal inputs, such as volatility, dividend yield, and the price-earnings ratio, which might also be adversely affected by economic structural shifts.

In a broader context, rare or extreme events indeed trigger economic structural shifts, which refer to significant and enduring changes within the macroeconomic system, including changes in market dynamics, policy alterations, or unforeseen events that reshape the economic landscape. It's essential to recognize that these structural shifts can introduce data in the test set that significantly deviates from the training set's input range, thus undermining the predictive prowess of data-driven deep learning. The criticality of this issue in both artificial intelligence and empirical finance domains has hitherto been under-emphasized.

Is the critique of deep learning's challenges, as discussed in the preceding text, merely based on unfounded apprehensions? Is the deviation of the test set from the training set really a reason for transfer learning outperforming deep learning in terms of performance? To quantify this, we introduce the Mahalanobis distance as a proxy variable of the diviation of data generating process (DGP) between the test set and its corresponding train set. The Mahalanobis distance from an n-dimensional vector $\vec{x}$ to an $n \times m$ dimensional matrix Q is defined by the following equation:

$$d_M(\vec{x}, Q) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}, \tag{15}$$

where S is the covariance matrix of Q, and μ is the sample mean of Q. Thus, we can calculate the distance from each set of input variables in the test set to the train set. From Table 3, we can see that the coefficient of distance is 1.93 and significantly positive. The corresponding t-statistic values are all greater than 199. This indicates that an increase in distance leads to a larger gap between deep learning and transfer learning, meaning that the relative performance of transfer learning is improved, which proves our hypothesis.

Deep learning is insufficiently equipped to tackle such unprecedented market scenarios. While this article does not seek to explore the reasons behind these feature distribution shifts, it underscores that structural economic disturbances invariably bring about alterations in the distribution of input features. Anticipating data-driven models to seamlessly adapt to these changes is optimistic, and deep learning's capabilities remain limited in addressing such intricacies.

The transfer learning framework implemented in this paper's AI-based economic model addresses the inherent limitations of deep learning. The training within the source domain utilizes simulated data derived from an economic model, with the data generation process during this simulation being entirely under the researchers' control. This permits researchers to establish a broad data generation spectrum encompassing even the most extreme economic scenarios. For instance, in this context, the preset annual implied volatility in the source domain spans from 0 to 1, while the risk-free interest rates range from 0.0 to 0.1. Such expansiveness ensures that artificial intelligence can accurately interpret pricing rules, even under rare or unprecedented economic conditions. Even profound economic upheavals and black swan events would likely not drive these variables beyond the specified range. A pertinent question that arises is: Does multi-target training in the source domain, characterized by a wide spectrum of preset data generation, necessitate a substantial increment in training data volume? Given that training in the source domain is a one-time process and the rolling training approach solely influences the target domain's design, concerns regarding the time invested in source domain training become moot.

4.2 Is the Outperformance of TL Due to More Training Data?

A natural question is: does the inferior performance of deep learning compared to transfer learning solely result from the latter having access to extra synthetic training data? To examine whether the discrepancy in performance between deep learning and transfer learning is exclusively attributed to the disparity in dataset sizes, we conducted an investigation into the viability of expanding deep neural networks. In each iterative training phase, our training

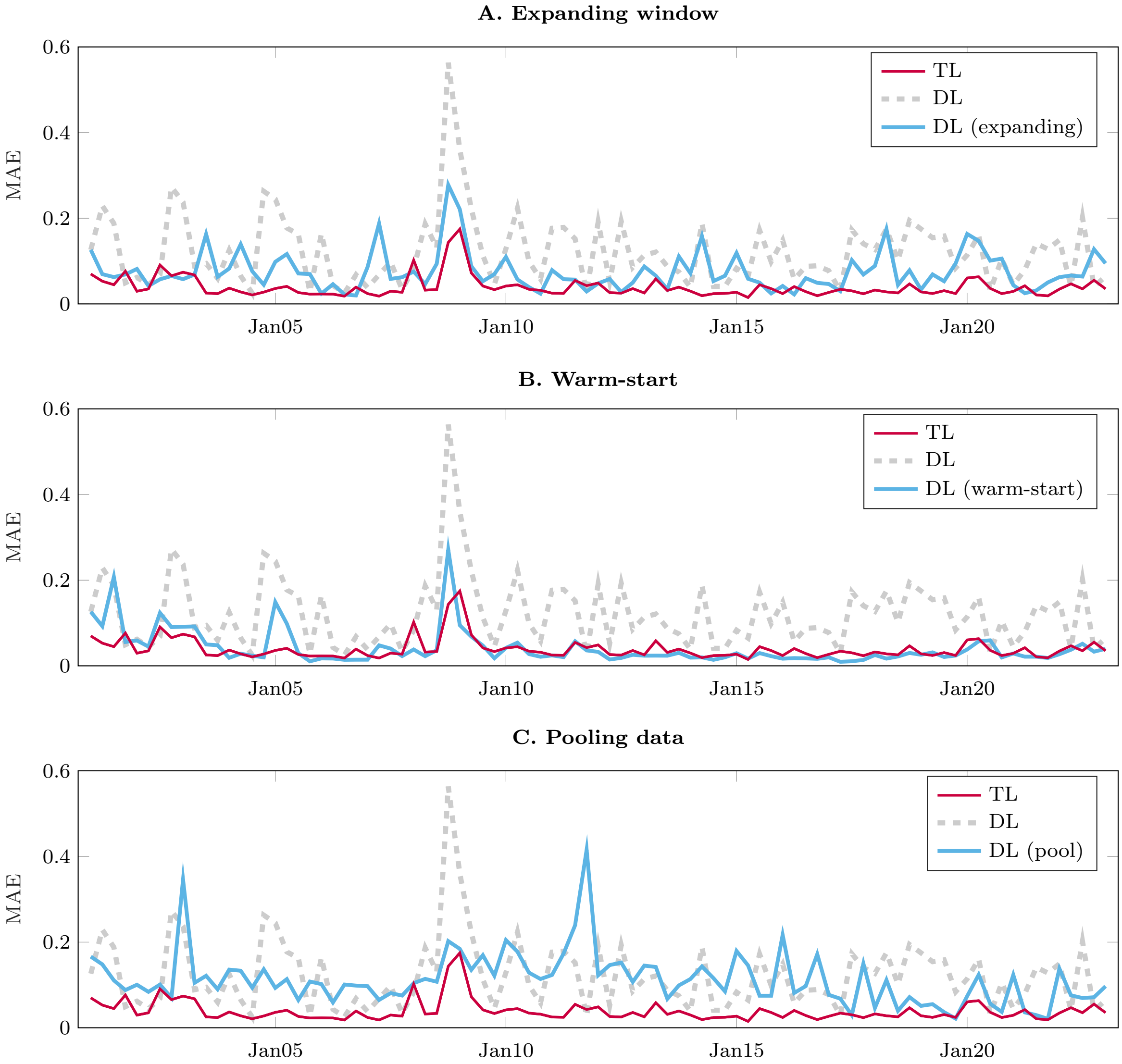

Figure 6: **TL vs. DL under expanding windows and warm-starting.** This figure compares the out-of-sample median-absolute pricing errors for TL and DL under the rolling-window setting against those of two differently-trained DLs. The expanding-window approach retrains the DL in each quarter using all the data available up to that quarter, with network parameters initialized randomly each time. The warm-starting approach follows the rolling-window procedure but trains the DL in each quarter by initializing its parameters using the values from the model trained in the previous quarter.

dataset encompassed all the previously employed training data, including all data preceding the introduction of the test set. Consequently, the training data available to our deep learning model, hereafter referred to as "expanding DL," significantly exceeded the volume of training data accessible to transfer learning models, encompassing both the source and target domains.

Additionally, we incorporated the warm-start method into our analysis. Warm-start refers to the technique where parameters from a previous training cycle are reused and further trained on a new dataset. This approach not only leverages continuity in training but also allows the deep learning models to benefit from a substantial accumulation of training data over time. This iterative enrichment of training data, coupled with the retention of learned parameters, potentially enhances model performance by providing a richer, more continuous learning trajectory as opposed to starting anew with each training session. Thus, by using both expanding datasets and warm-start techniques, our investigation aims to provide a more nuanced understanding of the factors that influence the performance disparities between deep learning and transfer learning approaches.

Our empirical results, as visualized in Figure 6, shed light on the impact of expanding DL and its relative performance in comparison to conventional deep learning. Surprisingly, the results indicated that expanding the training dataset did not yield substantial improvements when contrasted with traditional deep learning. In fact, the mean difference in prediction error between the two approaches averaged only 4.55×10^{-2}. It is worth noting that the augmentation introduced by expanding DL primarily comprised historical data predating the test set. The data generating process underlying this historical data may exhibit significant dissimilarities when compared to the data generating process of the test set. Consequently, the utility of this expanded historical data in enhancing predictions on the test set appears limited. This further underscores that the superior predictive performance observed in transfer learning scenarios is not solely a product of increased dataset size but rather an outcome of the model's inherent structural advantages.

In terms of the warm-start approach, results showed a modest improvement over conventional deep learning, with an average improvement of 0.080. In comparison to the expanding DL approach, which simply enlarges the dataset with historical data, the warm-start approach provides a strategic advantage. While the expanding DL incorporates older data, which might be less relevant due to differences in data generation processes, the warm-start method focuses on continuously improving the model's ability to adapt and refine its parameters based on new and relevant information. This ongoing adjustment and adaptation likely contribute to its superior performance over the expanding DL approach.

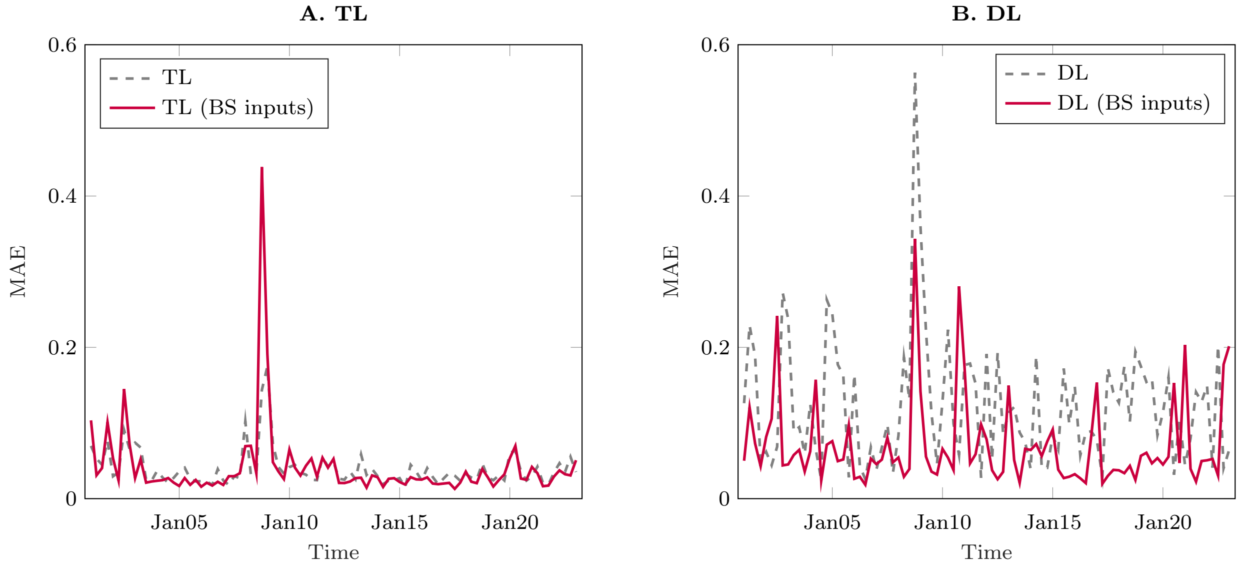

Figure 7: **TL and DL with model-implied inputs.** TL (BS inputs) refers to that within the transfer learning framework, we retained solely the inputs utilized in the Black-Scholes model. Similarly, DL(BS inputs) means that within the deep learning framework, we retained solely the inputs utilized in the Black-Scholes model.

However, this still lags behind the performance seen with transfer learning, where the average error was 0.002 lower than that of the warm-start deep learning models. Transfer learning involves pre-training a model on a large, well-curated dataset where the model learns a wide array of features that are generalizable, followed by fine-tuning on a smaller, specific target domain to adjust these features to new nuances. This method produces robust and adaptable models. In contrast, the warm-start approach enhances performance through continuous training and iterative refinement within the same framework, retaining knowledge across iterations. However, it lacks the broad exposure to diverse datasets and tasks during pre-training, which enriches transfer learning models, thereby limiting the diversity and richness of the feature set it can develop.

4.3 Is the Structural Model Simply Identifying Relevant Features?

In the primary results, we utilize 16 input features. Further exploration involved reducing the number of features, retaining only those inputs from the Black-Scholes model for the Transfer Learning (TL) and Deep Neural Network (DL) models, as presented in Figure 7. It is observed that using a larger number of features does not significantly enhance predictive accuracy compared to using only BS-inputs. As indicated in the figure, additional features only effectively reduced the implied volatility error at the peak of TL; they substantially mitigated the impact of extreme volatility fluctuations, yet at other test points, additional features did not result in a notable decrease in the Median Absolute Error, only improved 9.48×10^{-4} on average. In the DL model, the incorporation of more features even resulted in poorer prediction outcomes, with an average increase of 0.051 in BSIV-MAE. This contrast further underscores the superiority of Transfer Learning, which is able to significantly improve prediction accuracy through an advanced model structure rather than solely relying on feature complexity.This result reveals the contrast between theoretical modeling and the effectiveness of feature engineering. Using only variables related to the theoretical model as inputs for the neural network is a common feature engineering technique. After incorporating information from the theoretical model, the marginal improvements brought by feature engineering become relatively minor, highlighting the role of economic theories in the era of artificial intelligence.

These analyses highlight a significant advantage of transfer learning over deep learning: it can harness the predictive power of weak predictors without being adversely affected by their noise. This reduces concerns about the "garbage in, garbage out" problem, meaning we are less worried about the inclusion of noisy inputs. In contrast, deep learning is more prone to a modeling dilemma: while adding more features can provide additional information, it also increases the risk of being compromised by low signal-to-noise ratio inputs. When using deep learning tools, effective feature engineering becomes more critical, and achieving optimal performance may require sacrificing weaker covariates. Transfer learning, however, avoids such challenges entirely.

4.4 TL Performance under Different Structural Models

This section aims to demonstrate that transfer learning methods can also be employed to incorporate weak information, an approach of general significance in economics. In economics, there are certain key mappings whose mathematical properties we understand with reasonable certainty, yet the exact mathematical forms are elusive. For example, while we generally agree on the characteristics of the utility function regarding consumption as being monotonically

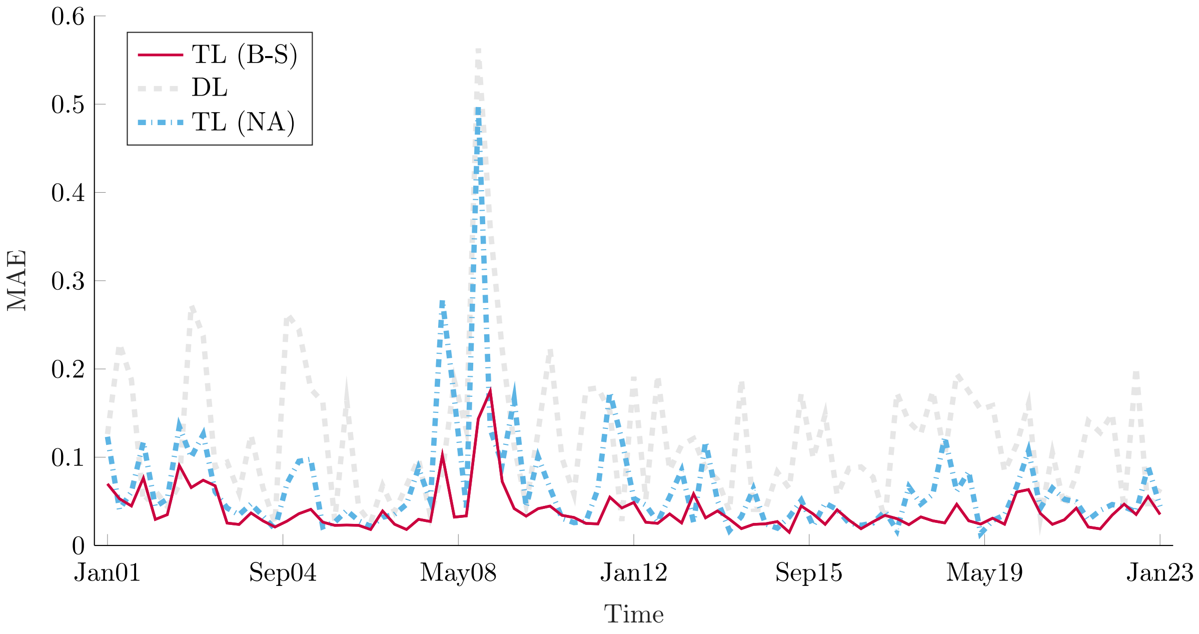

Figure 8: **TL with strong vs. weak structural restrictions.** In this figure, we compare the out-of-sample median-absolute pricing errors for TL that uses the Black-Scholes model in the source domain against another TL that imposes only the model-free no-arbitrage restrictions in the source domain. Specifically, the synthetic put option prices are randomly drawn from between the lower and upper bounds under the no-arbitrage conditions ($\max(Ke^{-rT} - S, 0)$ and Ke^{-rT}).

increasing and concave, it is challenging to articulate a universally accepted exact form.

In general, the current literature in the field of economics has seen neural networks aimed at flexibly fitting nonlinear functions, but the means of incorporating characteristics of these functions still require a general discussion. Transfer learning methods offer a novel and general way to address this issue. Specifically in the domain of option pricing, discussions on how to incorporate no-arbitrage information within the context of artificial intelligence technology, such as in the work by Cao, Liu, and Zhai (2021), have informed our design of an entirely new neural network architecture designed for option pricing and incorporating no-arbitrage information, which has significant potential for generalization. In the source domain, we first sample features such as strike prices, underlying asset prices, and risk-free interest rates according to a uniform distribution. Subsequently, we determine the upper and lower bounds of the no-arbitrage conditions for each observation, and finally, we generate random numbers as prices within the range of the no-arbitrage conditions to construct the

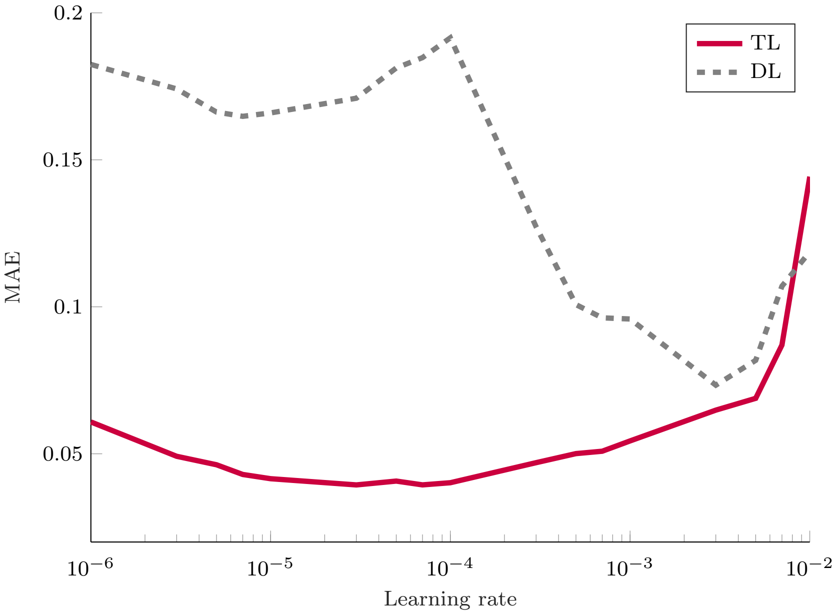

Figure 9: **Optimal Learning Rate in Target Domain.** The figure demonstrates the relationship between the learning rate and the out-of-sample pricing errors. We retrain the TL and DL model under different learning rates. The out-of-sample pricing errors (BSIV-MAE) are then averaged over the full sample.

training set for the source domain. The neural network then learns from this random dataset under no-arbitrage conditions as the source domain, followed by learning from real data. Our results (Figure 8) are consistent with Cao, Liu, and Zhai (2021), indicating that the no-arbitrage conditions offer significant improvement to deep learning, improving 0.022 when measured by BSIV-MAE error. This approach adeptly facilitates the infusion of intricate insights that are not from model dimensions of economic theories into the fabric of neural networks.

4.5 Optimal Learning Rate

The learning rate is a crucial hyperparameter in training machine learning models, impacting the convergence and performance of the model. To investigate the effect of learning rate on our target domain, we conducted a series of experiments and visualized the results in Figure 9. The figure clearly demonstrates a pronounced U-shaped curve, indicating that both excessively small and excessively large learning rates are detrimental to model performance.

Transfer learning, compared to traditional deep learning, is more suitable for training with a smaller learning rate. This is because transfer learning typically starts from a pre-trained model with weights already optimized on a related task, requiring only fine-tuning on the target task. Using a smaller learning rate helps preserve the knowledge embedded in the pre-trained model while allowing gradual adaptation to the new data, thereby avoiding significant deviations from the learned representations.

The performance curve of transfer learning around the optimal learning rate is noticeably flatter compared to deep learning. This indicates that transfer learning is more robust, as its performance does not degrade sharply when the learning rate deviates from the optimal value. Such robustness can be attributed to the pre-trained model's ability to stabilize training dynamics, making it less sensitive to hyperparameter variations and improving generalization on the target task

5 Alternative Ways to Bring in Structural Information

5.1 PINN and Boosting

In this section, we compare TL with other approaches for integrating information from theoretical models into the neural networks. For instance, the most direct approach seems to be the estimation with a combination of data conforming to pricing models and actual data, a concept similarly employed in the estimation of DSGE-VAR models in macroeconomics (Litterman (1986)). Specifically, we devise two schemes to examine this issue.

In the Pooled Data scheme, the training set for the neural network consists of a mixed dataset of real and simulated data. The simulated data is identical to the training set used in the Source Domain for the neural network. From Figure 6, on average, the Pooled Data improves the BSIV-MAE error by only 9.28×10^{-3}, which is significantly less than the level achieved by transfer learning. While this scheme seems similar to the transfer learning approach, the order of data input leads to a marked discrepancy in results. This is due to the Pooled Data scheme's issue of biased targets. The True Model is embedded within the actual data, not the theoretical model. As such, the Pooled Data scheme's concurrent consideration of errors from both theoretical and actual data in effect makes the neural network's objective diverge from our real target. Transfer learning, on the other hand, better manages the asymmetrical relationship between theoretical models and actual data by fine-tuning with real data, rather than assigning equal status to both.

Similarly, we empirically tested Physics-Informed Neural Networks (PINNs), which refer to a class of models that incorporate physical model constraints into the learning process, enabling multi-objective learning by combining different loss terms. In this work, we applied a PINN approach to integrate the Black-Scholes model with actual data. Specifically, we tested the Mixed Target scheme, which does not expand the training set but instead uses a weighted average of the BS model Error and the actual data error as the loss function during the neural network's training, which is

$$Loss_{total} = \lambda_{BS} Loss_{BS} + \lambda_{real} Loss_{real}. \tag{16}$$

To ensure comparability, within each training set, the weights were set according to the ratio of the volume of data from the source domain in the TL scheme to the sample size of the actual data training set used in the original deep learning approach. The empirical results of this scheme (Figure 10), similar to the Pooled Data, suffer from the biased target issue, and its performance is significantly inferior to transfer learning, worsened the BSIV-MAE error by 0.077 on average compared with deep learning.

Another penalty for deep learning can be related to no-arbitrage theory. Unlike the Black-Scholes model, which inherently carries a bias, no-arbitrage conditions are considered to be unbiased. In this way, the total loss function is composed of two components: the first measures the primary error between predictions and real values, while the second penalizes cases where samples fall outside the defined range. Specifically, the total loss $Loss_{total}$ is calculated as:

$$\begin{aligned} Loss_{total} = {} & \lambda_{real} Loss_{real} \\ & + \lambda_{NA} \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\{X_i \notin range_i\} \times min\{|X_i - Ke^{-rt}|, |X_i - max(Ke^{-rT} - S, 0)|\}. \end{aligned} \tag{17}$$

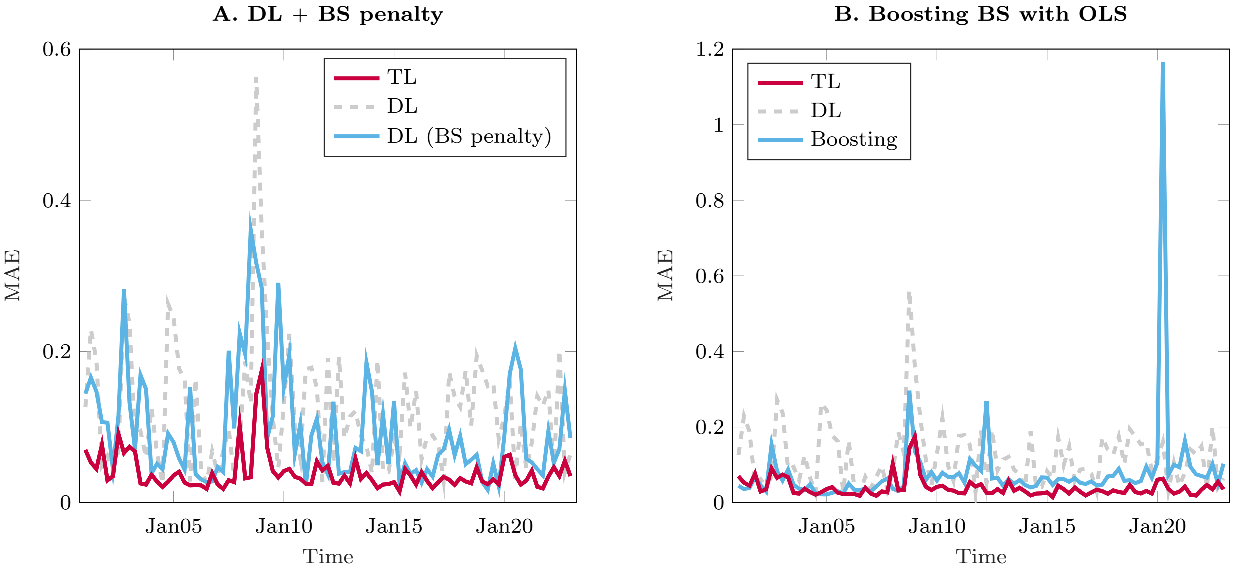

Figure 10: **Comparison with alternative ways to incorporate structural model information.** In this figure, we compare the out-of-sample BSIV-MAE for TL against those of a DL that treats the Black-Scholes model restrictions as constraints (Panel A) and a Black-Scholes boosting model (Panel B).

Here, N is the number of the sample. The indicator function $\mathbb{I}\{X_i \notin range_i\}$ identifies whether a sample X_i lies outside the no-arbitrage range $range_i$: it equals 1 if X_i is outside $range_i$ (the complement of the range), and 0 otherwise. The upper bound of $range_i$ is defined as Ke^{-rt}, while the lower bound is $max(Ke^{-rT} - S, 0)$, where r represented the risk-free rate, S denoted the current price of the underlying asset, K was the option's strike price, r stood for the risk-free rate, and T represented the remaining time to expiration. The second term, $\sum_{i=1}^{N} \mathbb{I}\{X_i \notin range_i\} \times min\{|X_i - Ke^{-rt}|, |X_i - max(Ke^{-rT} - S, 0)|\}$, the degree to which the predicted values deviate from the no-arbitrage condition, weighted by λ_{no_arb}. This ensures that the model not only minimizes prediction error but also adheres to predefined economic constraints.

In unreported results, we find that even for the no-arbitrage pricing constraint that does not introduce any bias, the effect of transfer learning is superior to that of multi-objective learning with an added penalty term. Multi-objective learning moderately improves the performance of DL, with the improvement brought by the introduction of no-arbitrage conditions through transfer learning being more pronounced.

5.2 Comparison between Transfer Learning and Bayesian Methods

This section aims to highlight the structural advantages of transfer learning. The use of simulated data to enhance predictive capabilities has been demonstrated in the application of Bayesian methods. Del Negro and Schorfheide (2004) use the Bayesian approach to apply constructed simulated data to macroeconomic forecasting. Since Bayesian methods combine data from prior distributions with real data to form a posterior distribution, the inclusion of early simulation data incrementally improves prediction accuracy. However, once the data from the prior distribution reaches a certain volume, the excess of non-real data can lead to a diminished significance of real data in the model, decreasing the grasp on reality and thus prediction accuracy. Therefore, in Bayesian methods, the relationship between prediction error and the ratio of simulated data volume to real data volume typically forms a U-shaped curve. Transfer Learning effectively avoids this problem by distinctly separating simulated data from real data, training within the source domain and target domain separately; it first captures the general direction of prediction in the source domain training, followed by adjustment of parameters in the target domain using real data as the training set. In such a structure, no matter how much the pre-training data volume increases, it only functions within the source domain. In the target domain, regardless of the volume of simulated data, further training with real data is essential, implying that Transfer Learning does not suffer from reduced importance of real data due to excessive volumes of simulated data, thereby maintaining accurate prediction targets and achieving better predictive performance.

To validate this, we selected four representative forecast points for testing: the first quarter of 2002, the fourth quarter of 2008, the second quarter of 2014, and the first quarter of 2020. Our test results are presented in Figure 11, displaying the ratio of the MAE of implied volatility predictions from Deep Learning to that from Transfer Learning. It can be observed that the predictive performance does not show a trend of initial improvement followed by decline as the ratio of pre-training training set samples to real data training set samples increases. The training sample sizes of the target domains corresponding to these four points vary significantly, yet the model demonstrates high stability at each point when the volume of data is sufficient to reduce the impact of randomness.

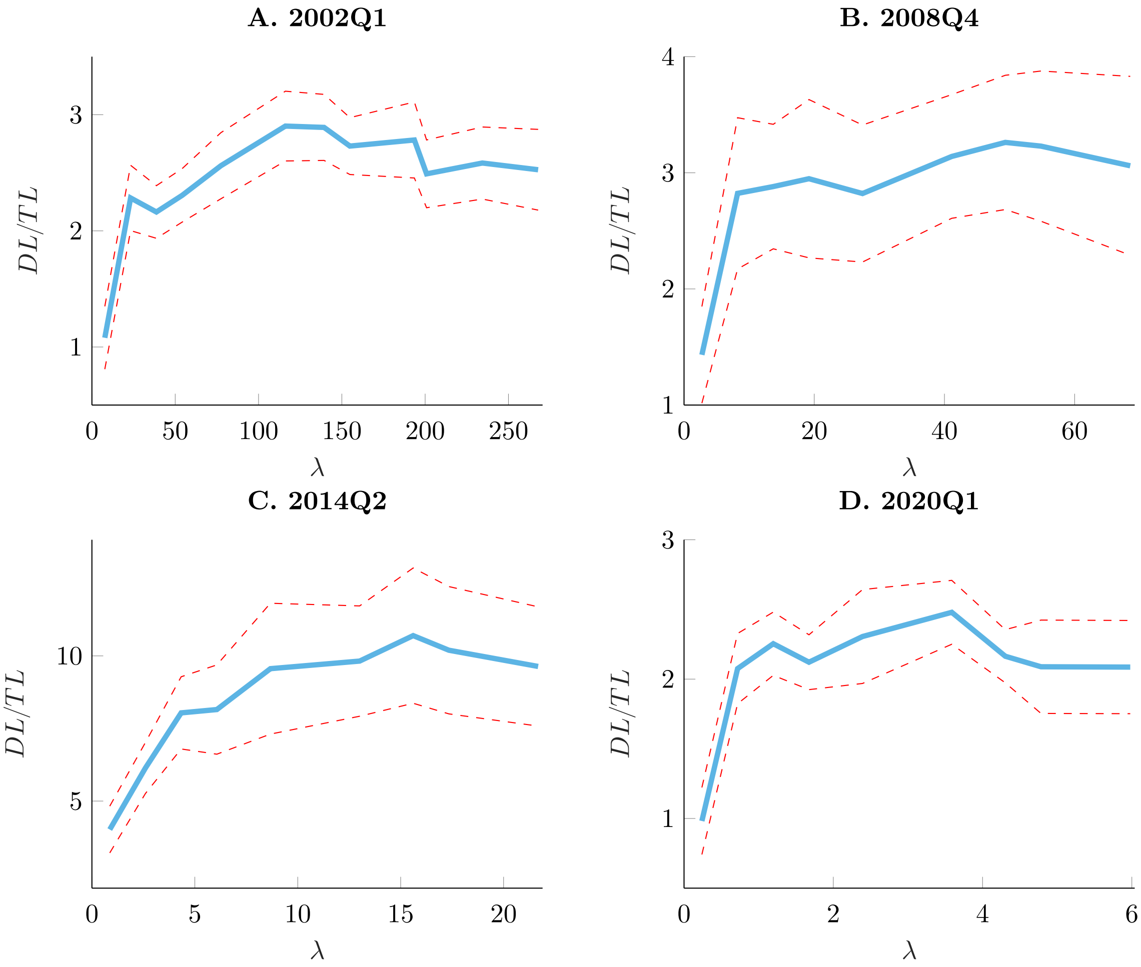

Figure 11: **Changing the Relative Quantity between the source domain and the target domain.** The figure demonstrates the effect of the ratio of training data between the source domain and the target domain on prediction error. In the graph, the y variable is the error of deep learning divided by the error of transfer learning. We have chosen four representative quarters. Prediction performance is measured by calculating the Median Absolute Error of deep learning in estimating implied volatility at that point, divided by the MAE of transfer learning in estimating implied volatility at that point. We vary the ratio by changing the number of training samples in the source domain.

The monotonicity of the lambda pair in transfer learning provides a distinct advantage over Bayesian optimization. This pattern means that transfer learning does not require the deliberate search for the optimal lambda to enhance performance; instead, it only needs a sufficiently large pre-training sample to allow the model to automatically determine the best lambda. In this sense, transfer learning can be seen as capable of discovering the

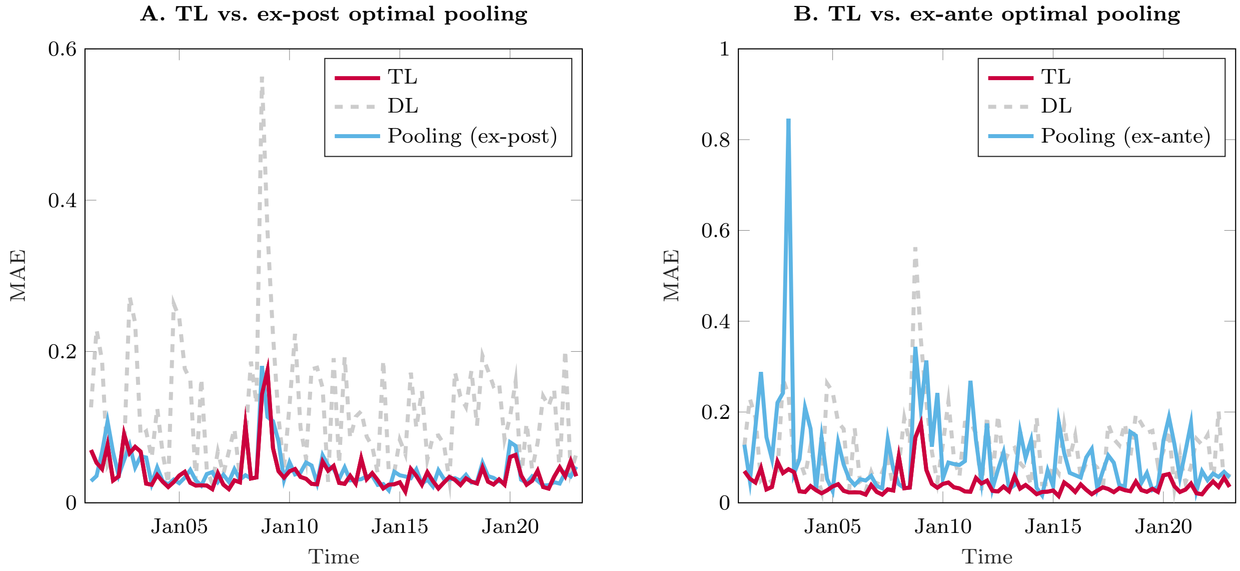

Figure 12: **Optimal Performance of Bayesian Method.** This figure presents the error curves achieved under the posterior optimal approximation using a Bayesian method, compared with those obtained from transfer learning and deep learning. At each point, we train multiple weights with synthetic and real data, then select the weight that performs best on the test set.

optimal lambda through data. On the other hand, Bayesian optimization requires a numerical optimization search to find the best lambda, which introduces additional complexities, such as addressing the potential inconsistency between pre-optimization (prior) and post-optimization (posterior) results. Thus, transfer learning simplifies the process by automatically adjusting lambda, avoiding the technical challenges inherent in Bayesian optimization, and improving efficiency.

Now, in the Bayesian method, we incorporate synthetic data into the training set with varying weights. Specifically, the ratio of synthetic data to real data is set as 0.1 through 0.9 at the increment of 0.1, and then 1 through 9 at the increment of 1. In the case of pre-optimization, we select the optimal weight from the previous test set as the weight for the next test set. In the case of post-optimization, we evaluate the performance of all weight combinations on the test set and select the weight configuration that minimizes the error. The significance of this work lies in providing an explanation of transfer learning (TL) through a mixed data approach. The mixed data method is quite similar to the approach used in Bayesian Vector Autoregression (BVAR).

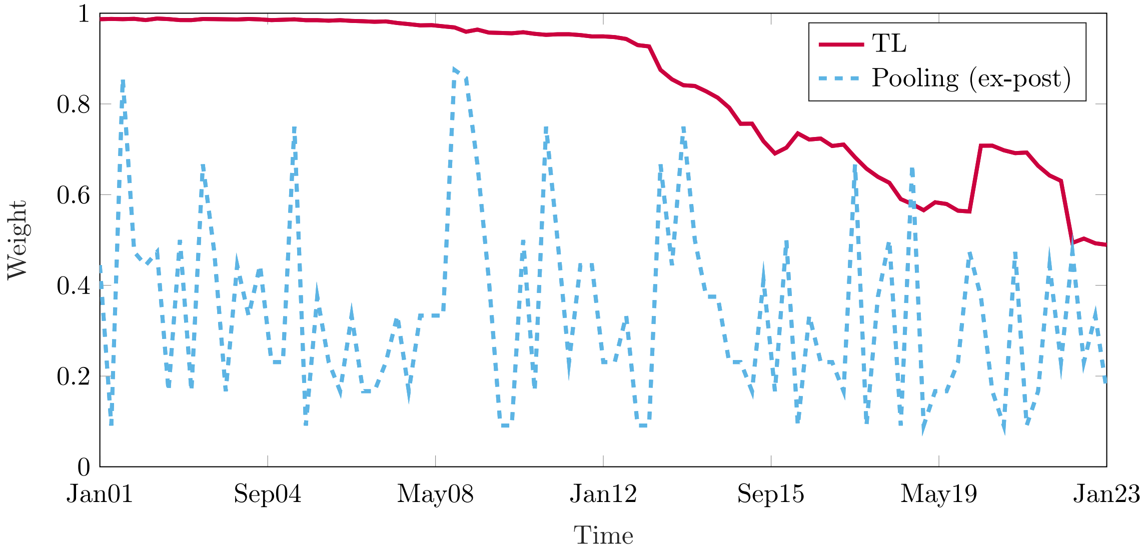

Figure 13: **Weights of Synthetic Data.** The figure illustrates the weights of synthetic data in transfer learning and the optimal weights derived using a deep learning-based Bayesian method. At each time point, we extract the weights corresponding to the minimum value from multiple sets of weights to determine the optimal weights.

Figure 12 reveals that Bayesian optimization with dynamic optimization (pre-optimization) cannot match the performance of Transfer Learning. This limitation could stem from the lack of sufficient prior information or the model's inability to capture the complexities inherent in the data. Bayesian methods rely on prior distributions for inference, but if the prior information is insufficient or poorly chosen, the model's predictive power may be constrained, resulting in suboptimal performance compared to TL. When future information is incorporated, the mixed data method can effectively mimic the performance of transfer learning. However, in practice, using future data in the optimization process is not possible, as this contradicts the core principle of predictive modeling, where future data cannot be known at the time of prediction.

The weight assigned to future information helps us better understand the underlying mechanisms of transfer learning, as this shows transfer learning framework can consider the optimal weights in its training process without using future information. This can be further reflected in the changes in the optimal weights.

Figure 13 presents two curves that offer intriguing insights. One curve represents the proportion of synthetic data within the total dataset, i.e., the ratio of the real data training set to synthetic data. This curve is nearly monotonically decreasing, dropping from close to 1 to approximately 50 percent. The reason for this is straightforward: as the market develops, an increasing amount of option data becomes available. The second curve, which oscillates, represents the relative weight of model information determined by future data. 1) This oscillation clearly highlights a significant difference between transfer learning and Bayesian methods. The proportion of synthetic data in the sample does not dictate how much synthetic data is utilized. Instead, artificial intelligence dynamically searches for the optimal usage level of synthetic data. 2) The optimal proportion of synthetic data fluctuates dramatically over time, making it difficult to search for a stable value. This reveals a weakness in the Bayesian approach. When a researcher attempts to use a sliding window to search for the optimal lambda in the Bayesian method, they will likely struggle to find the lambda that would be considered optimal in retrospect. This issue illustrates a key limitation of the Bayesian method, particularly in dynamic settings, and emphasizes the advantages of more flexible, adaptive approaches like transfer learning.

6 Conclusion

This article presents a transfer learning-based methodology that integrates economic model information into the neural network framework, demonstrating its broad applicability across various areas of economics and finance, beyond just asset pricing. While the model is applied to option pricing in this study, its flexible structure can be extended to other domains where traditional economic models can be enriched by machine learning techniques. The transfer learning framework outperforms stochastic volatility models and deep learning approaches in terms of pricing accuracy, and provides substantial improvements in implied volatility prediction. This methodology offers a more general, versatile approach that could be applied to a wide range of economic and financial challenges, highlighting its potential to enhance both theoretical and empirical research across multiple fields.

Our attribution analysis results indicate that the new transfer learning framework can overcome the inherent drawbacks of data-driven methods compared to deep learning. It is more suitable for high-volatility market environments, characterized by small-sample learning, and is more robust to various shocks like regime shifts. Additionally, the transfer learning framework performs better for traditionally challenging models like short-term options.

Theoretically, this paper addresses two important asset pricing questions. The first is the relationship between structured financial models and data-driven artificial intelligence. For a long time, the academic community has accumulated a wealth of traditional financial models. Can these models improve AI models for researchers using machine learning in asset pricing? This paper provides a comprehensive and flexible framework that can integrate information from structured models in a more general sense.

The second point provides a solution for handling time-varying rules and can robustly deal with rare risk shocks. In our attribution analysis, we reveal that when the test set's distribution significantly differs from the training set's, the performance of transfer learning is much more robust than deep learning. This is because deep learning purely driven by data struggles to identify rules that are rare or never occurred in history, while transfer learning, with the help of the source domain, remains robust during periods of drastic distribution changes.

The third issue is the training of AI models for asset pricing in the case of key variables in the data with low discrepancy. The construction of asset pricing models using direct deep learning techniques requires big data support and is thus difficult to implement in this scenario. Since economic and financial data are not always abundant, whether AI can serve asset pricing under such circumstances is a sufficiently important question. The transfer learning framework in this article provides a clear approach to small-sample learning. This issue can be extended to general economic and financial problems.

The fourth point is how to avoid neural networks mistaking noise as rules in noisy financial datasets. This article provides a solution to this problem. Transfer learning using economic model information as a regularization method is an important solution.

Finally, transfer learning reveals the crucial role of theoretical models in the age of artificial intelligence. We find that even with considerable effort in feature engineering, its effect improvement is hard to match the help of theoretical models in the source domain for neural networks. This responds to the initial question of whether theory is dead in the age of AI. Even as data volume increases, AI will still face challenges of time-varying rules, signal-to-noise ratio, and other issues. The economic models created by human economists are of crucial importance for AI to overcome its inherent flaws.

7 Tables and Figures

Table 1: **Descriptive statistics**

We considered all SP500/SPX European put option samples on CRSP from 2001 to 2023. Due to the existence of bid-ask parity, we only care about call options. The table divides the options in the whole market into 18 categories according to the expiry time and the logarithmic moneyness of the simplified algorithm and calculates the mean and standard deviation of the implied volatility of the option within each category.

Num (million)				IV-mean				IV-Std			
T	<10	[10,60]	>60	T	<10	[10,60]	>60	T	<10	[10,60]	>60
log(K/S)	Nums			log(K/S)	Mean			log(K/S)	Std		
< -0.5	0.256	0.920	1.710	< -0.5	0.700	0.794	0.734	< -0.5	0.374	0.319	0.378
[-0.5, -0.25]	0.469	1.710	1.560	[-0.5, -0.25]	0.725	0.727	0.628	[-0.5, -0.25]	0.377	0.381	0.416
[-0.25, 0]	2.510	6.490	3.240	[-0.25, 0]	0.538	0.504	0.401	[-0.25, 0]	0.433	0.396	0.301
[0, 0.25]	1.510	3.930	2.540	[0, 0.25]	0.144	0.121	0.175	[0, 0.25]	0.213	0.156	0.236
[0.25, 0.5]	0.073	0.234	0.500	[0.25, 0.5]	0.515	0.309	0.268	[0.25, 0.5]	0.452	0.324	0.389
> 0.5	0.005	0.029	0.108	> 0.5	0.547	0.356	0.333	> 0.5	0.510	0.403	0.432
Total	4.820	13.300	9.650	Mean	0.528	0.469	0.423	Mean	0.393	0.330	0.359

Table 2: **Explanatory variable description table for attribution analysis**

The table shows the frequency and brief description of the explanatory variables in the regression.

Variable	Frequency	Des.
T	Contract-daily	Time to maturity
marketIV60	Daily	60-day average of VIX
BASpread	Contract-daily	Bid-ask spread
Distance	Contract-daily	The Mahalanobis distance between each contract observation and the distribution of the training set.
volume5daily	Daily	5-day average of the variable "volume5", a 5-day moving average of the normalized volume deviations, lagged by one day.
mIVchange	Daily	Change of market implied volatility between train set and test set
ITM	Contract-daily	In-The-Money
OTM	Contract-daily	Out-of-The-Money
DOTM	Contract-daily	Deep-Out-of-The-Money
DITM	Contract-daily	Deep-Out-of-The-Money

Table 3: **Performance Attribution Analysis**

The table shows the results of the attribution analysis.The dependent variable is the difference between the absolute pricing error (in terms of BSIV) for deep learning and transfer learning for contract i on day t.The superscripts ***, **, and * indicate statistical significance at the 1%, 5%, and 10% levels, respectively.

	(1)	(2)	(3)	(4)	(5)	(6)
0-7	-0.0201***	-0.0216***	-0.0211***	-0.0205***	-0.0221***	-0.0169***
	(-99.18)	(-108.29)	(-104.00)	(-100.85)	(-107.76)	(-83.07)
7-14	-0.0118***	-0.0129***	-0.0124***	-0.0121***	-0.0138***	-0.0124***
	(-76.75)	(-85.70)	(-80.37)	(-78.43)	(-89.53)	(-80.92)
90+	0.0170***	-0.0136***	0.0143***	0.0108***	0.0251***	0.0176***
	(15.20)	(-12.16)	(12.79)	(9.69)	(21.44)	(15.15)
marketIV60			0.0719***	0.1170***	0.0907***	0.1060***
			(95.23)	(143.62)	(108.49)	(126.67)
mIVchange			0.1810***	0.1790***	0.1100***	0.1180***
			(219.74)	(216.67)	(119.97)	(130.34)
volume5daily				0.0136***	0.0200***	0.0180***
				(129.83)	(75.60)	(72.86)
distance					1.9300***	1.9300***
					(200.96)	(199.15)
BAspread					-0.6880***	-0.6280***
					(-18.78)	(-18.62)
OTM	0.0318***	0.0330***	0.0355***	0.0342***	0.0324***	0.11***
	(266.00)	(277.04)	(293.61)	(279.75)	(220.21)	(333.76)
DOTM	0.0720***	0.0637***	0.0743***	0.0728***	0.0713***	0.194***
	(193.08)	(174.19)	(199.07)	(194.31)	(187.59)	(239.74)
ITM	0.04470***	0.0333***	0.0425***	0.0427***	0.0437***	0.0466***
	(104.94)	(78.20)	(99.64)	(100.22)	(102.41)	(78.76)
DITM	0.0140***	0.0122***	0.0130***	0.0131***	0.0131***	0.00625***
	(22.17)	(18.05)	(20.52)	(20.64)	(20.65)	(9.38)
IV × ITM						-0.026***
						(-11.25)
IV × DITM						0.1***
						(16.68)
IV × DOTM						-0.262***
						(-144.41)
IV × OTM						-0.185***
						(-231.62)
Const	0.0541***	0.1790	0.0399***	0.0335***	0.0354***	0.0341***
	(916.22)	(0.98)	(254.81)	(206.75)	(176.84)	(174.93)
Time FE	×	✓	×	×	×	×
R^2	0.006931	0.058859	0.010283	0.011084	0.014409	0.023819

Appendix

A Details of Neural Network Architecture and Training

The employed neural architecture in this article encompasses 16 hidden layers with 16 input variables each. Each hidden layer consists of 22 neurons and employs Leaky ReLU as the activation function. These hidden layers are linear transformations, constituting fully connected layers. To address the gradient vanishing problem, we introduced the ResNet structure. Detailed below are the specific parameters and structures utilized for all our results.

Transfer Learning: For the source domain, we generated a series of random factor values and computed the prices produced by the Black-Scholes model for each data point. Employing these factor values as input variables and the prices derived from the Black-Scholes model as output variables, we conducted training using a total of 700,000 data instances. Training encompassed 200 epochs, with a learning rate of 0.0002 and a batch number of 600. We computed three distinct loss functions: Mean Absolute Error (MAE) of Vega, MAE of Delta, and a weighted MAE that utilizes Delta as weights between predicted and actual prices. These three loss functions were combined with a weight ratio of 0.2:0.2:1, yielding a novel loss function for backpropagation. For the target domain, we fine-tuned the model using real data's factor values as input variables and actual prices as output variables. The training and validation sets together covered 3 months of data, with a 7:3 ratio between them, while the testing set consisted of an additional 3 months of data. The fine-tuning stage employed an early stopping strategy, terminating training early if the validation loss increased consecutively over two epochs. The learning rate, determined through a sparse grid search on data preceding the first train-test pair, was set at 0.0001, and the batch size was calculated by dividing the training set's sample size by 600. Backpropagation utilized the weighted MAE of predicted prices and actual prices with Delta as weights as the loss function.

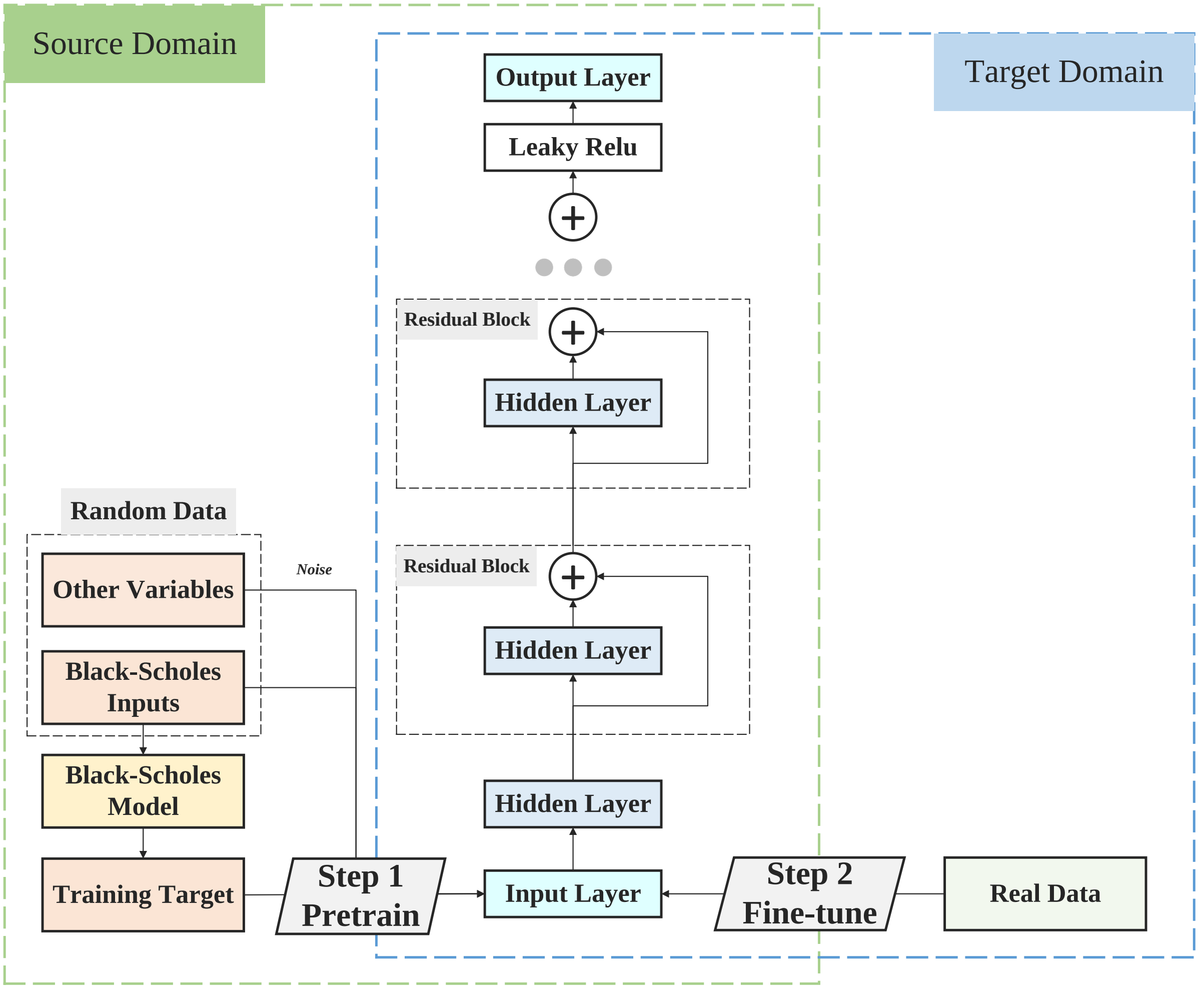

Figure A.1: **The Transfer Learning Framework.** This figure illustrates the application of our adopted transfer learning framework in the context of option pricing.

Deep Learning: We exclusively employed real data and conducted training following the methodology established in the Transfer Learning paradigm for the target domain.It is important to note that the learning rate for the deep learning model was also determined in advance through a sparse grid search, resulting in a value of 0.001.

Deep Learning - Warm Start: Exclusively utilizing real data, we followed the training approach of the target domain as outlined in the Transfer Learning methodology. In contrast to regular Deep Learning, the network was initialized only before the initial training; subsequently, parameters from the previous training were retained throughout the rolling training without

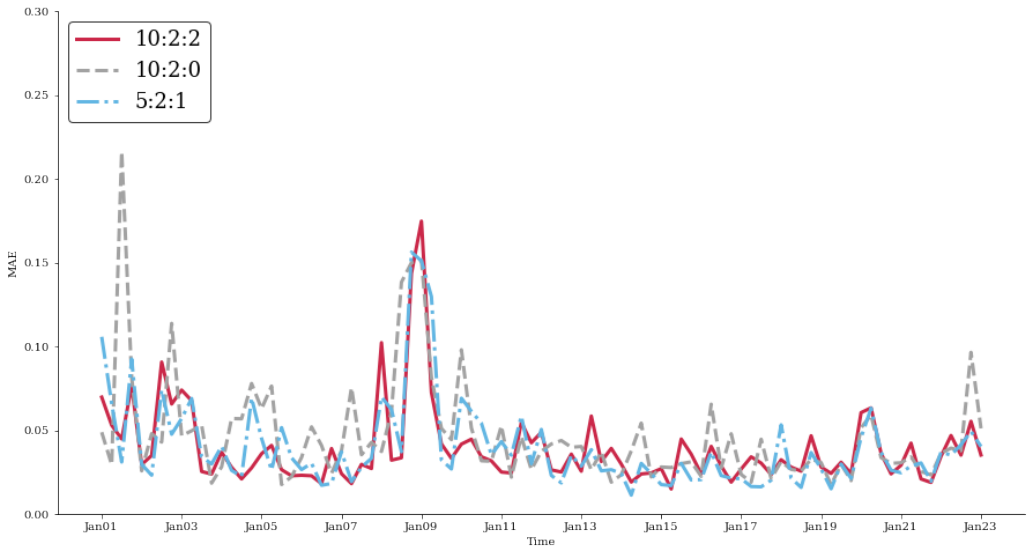

Figure A.2: **Robustness Check: Weight of Source Domain Multi-target Training.** We change the Weight of Source Domain Multi-target Training. The main results in the main text report the pricing, delta hedging, and vega hedging weights set to 10:2:2. We set the weights to 5:2:1 and 10:2:0 to observe whether there is a large change in the pricing error. The frequency of error evaluation is the same as the frequency of model retraining, that is, model retraining and model evaluation every three months. The loss function of the model is chosen as IV-MAE. Specifically, we first convert the model-predicted price and the real price into implied volatility and then calculate the average absolute error between the two implied volatility. This scheme is designed to ensure that the model pays sufficient attention to out-of-the-money options.

further initialization.

Deep Learning - Expanding Window: The Expanding Window method refers to a strategy for incrementally increasing the size of the dataset used for training a model.As time progresses, new data is continuously incorporated into the training set, expanding the window of historical data the model learns from.

Pooled Data Method: We merged the randomly sampled training set from the source domain in Transfer Learning with the real data set. Subsequently, training was executed following the target domain's methodology without pretraining. We set the number of epochs

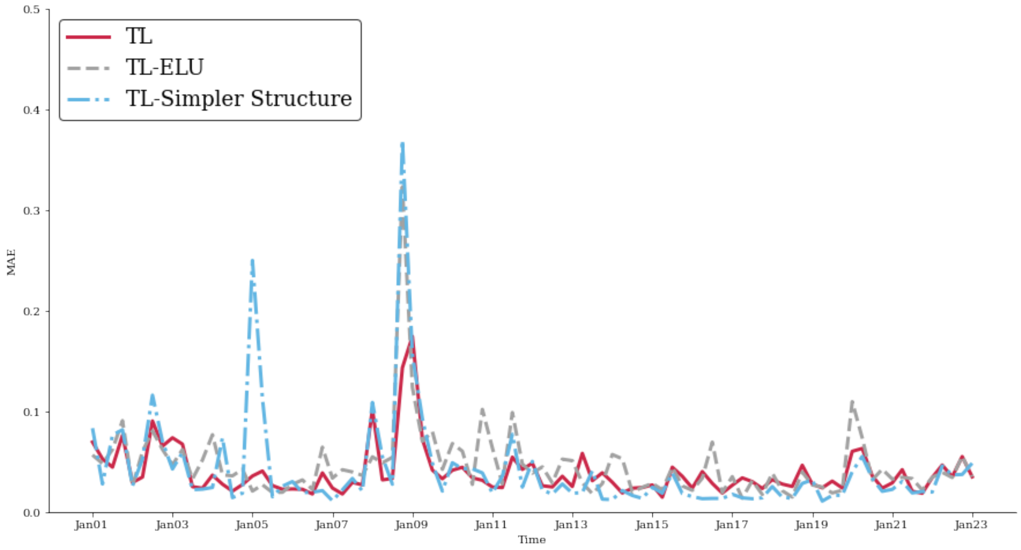

Figure A.3: **Robustness Check: NN structure and activation function.** We change the activation function. The frequency of error evaluation is the same as the frequency of model retraining, that is, model retraining and model evaluation every three months. The loss function of the model is chosen as IV-MAE. Specifically, we first convert the model-predicted price and the real price into implied volatility and then calculate the average absolute error between the two implied volatility. This scheme is designed to ensure that the model pays sufficient attention to out-of-the-money options.

to 20 to ensure network convergence.

Deep Learning with Mixed-Target Loss Function: In Deep Learning, we modified the loss function during training. Firstly, we separately computed the weighted MAE between predicted prices and actual prices, as well as the MAE between predicted prices and prices derived from the Black-Scholes model. Subsequently, weights were allocated based on in-sample grid search.

Only Impose No-Arb Constraint: We adopted the training pattern of Transfer Learning, with the distinction that the output variable was no longer the price derived from the Black-Scholes model. Instead, it ranged between randomly generated upper and lower bounds under the no-arbitrage conditions. The upper bound was defined as Ke^{-rt}, while the lower bound was $max(Ke^{-rT} - S, 0)$, where r represented the risk-free rate, S denoted the current price of the underlying asset, K was the option's strike price, r stood for the risk-free rate, and T represented the remaining time to expiration.

Deep Learning with Only Input Variables from the Black-Scholes Model: Within the Transfer Learning framework, we retained solely the inputs utilized in the Black-Scholes model.This means that during the training in both the source domain and the target domain, only the inputs of the Black-Scholes model were considered as inputs for the neural network.

Deep Learning with Only Input Variables from the Black-Scholes Model: In the context of Deep Learning, only the parameters used in the Black-Scholes model were retained.

Table A.1: **Alternative Loss of Pricing Models**

The following formula reports the full sample results of various loss functions. The following Loss reports the summary of pricing error results on all test sets. MAE is the mean absolute error of simple statistics, and MSE is the mean square error. The weighted error is the result of Delta weighting. In the three ways of weighting 1, weighting 2, and weighting 3, the constant δ_c to ensure the stability of the value is set to 0.1, 0.08, and 0.12 respectively. MAPE is the mean absolute percentage error.

		TL	***DL***	***Heston***
mse	unweighted	0.0174	0.2910	3.28
	weighted-mse1	0.0054	0.0463	7.42
	weighted-mse2	0.0052	0.0408	8.04
	weighted-mse3	0.0057	0.0515	6.96
mae	unweighted	0.594	2.55	10.4
	weighted-mae1	0.333	1.15	11.6
	weighted-mae2	0.314	1.07	11.7
	weighted-mae3	81.400	393.00	36500.0
mape	unweighted	102	1136	38170

B Description of Features for DL and TL

1. **S**: Represents the price of the S&P 500 index divided by 1000.

2. **K**: The strike price of an option divided by 1000, which is the price at which the holder of the option can buy (call) or sell (put) the underlying stock or index.

3. **T**: Time to expiration, expressed in years. This is the remaining time until the option expires.

4. **r**: The risk-free interest rate.

5. **d**: The dividend yield, which is the rate of dividends paid out by the underlying stock or index relative to its price.

6. **IV1**: Lagged 1-day of the VIX index. The VIX is referred to as the market's "fear gauge" and measures the market's expectation of volatility over the upcoming 30 days.

7. **mom1**: Short-term momentum of the option price calculated as the logarithm of the ratio of the real price one day ago to the real price six days ago.

$$\text{mom1} = \log\left(\frac{\text{real\_price}_{t-1}}{\text{real\_price}_{t-6}}\right)$$

8. **mom4**: Medium-term momentum of the option price calculated as the logarithm of the ratio of the real price one day ago to the real price 21 days ago.

$$\text{mom4} = \log\left(\frac{\text{real\_price}_{t-1}}{\text{real\_price}_{t-21}}\right)$$

9. **hv1**: This variable represents the annualized standard deviation of the daily log returns of the S&P 500 index, calculated over a 20-day rolling window. The calculation involves taking the square root of the sum of squared log returns over the past 20 days, multiplying by the annualization factor (252 trading days in a year), and then dividing by the window size (20 days). This value is lagged by one day:

$$\text{hv1} = \sqrt{\left(\frac{\text{Sum of squared log returns over 20 days} \times 252}{20}\right)}$$

10. **hv9**: Similarly, this variable calculates the annualized standard deviation of daily log

returns over a 180-day rolling window. The procedure is the same as for hv1, but using 180 days for the window size. This value is also lagged by one day:

$$\text{hv9} = \sqrt{\left(\frac{\text{Sum of squared log returns over 180 days} \times 252}{180}\right)}$$

11. **volume1**: A transformed volume metric, measuring the normalized deviation of the current volume from the rolling mean of the past 20 days, lagged by one day.
12. **volume5**: A 5-day moving average of the normalized volume deviations, lagged by one day.
13. **Put-Call Ratio (PCratio)**: Calculated as the ratio of total open interest of puts to calls, providing a measure of market sentiment. A higher ratio indicates more bearish sentiment.

$$\text{PCratio} = \frac{\text{Open Interest of Puts}}{\text{Open Interest of Calls} + 1}$$

14. **Earnings-Price Ratio**: This ratio is calculated as the earnings (E) divided by the price (P) of S&P 500.
15. **spmom1**: The logarithm of the short-term momentum of the S&P 500 index, measured as the ratio of the index value one day ago to six days ago.

$$\text{spmom1} = \log\left(\frac{\text{price}_{t-1}}{\text{price}_{t-6}}\right)$$

16. **spmom4**: The logarithm of the medium-term momentum of the S&P 500 index, measured as the ratio of the index value one day ago to 21 days ago.

$$\text{spmom4} = \log\left(\frac{\text{price}_{t-1}}{\text{price}_{t-21}}\right)$$

C Generation of Synthesized Source Domain Data

The following table details the features used for generating synthetic data, their distribution ranges, and the formulae for derived metrics:

For each data point, the price is calculated according to the Black-Scholes (BS) model

Feature	**Distribution**	**Range**
S	Uniform	0 to 5
K	Uniform	0 to 5
T	Uniform	0 to 4
r	Uniform	0 to 0.1
d	Uniform	0 to 0.1
IV1	Uniform	0 to 1
mom1, mom4	Uniform	-10 to 10
hv1, hv9	Uniform	0 to 0.8
volume1, volume5	Uniform	-4 to 4
PCratio (Put-Call Ratio)	Uniform	0 to 2.5
epratio (Earnings-Price Ratio)	Uniform	0 to 0.1
spmom1, spmom4	Uniform	-0.3 to 0.2

to serve as the training target, and both Delta and Vega are computed. Inputs not within the structural model will be random variables independent of the source domain prediction target.

D Robustness Checks

D.1 Network Hyperparameters

Within the source domain, we define the hyperparameters for multi-objective training. The weights assigned to pricing, delta hedging, and Vega hedging are set at 10, 2, and 2, respectively. The guiding principle behind these settings is to prioritize minimizing pricing errors. When evaluating the trade-off between Delta hedging and Vega hedging, we recognize that the price of the underlying asset is more readily observable compared to volatility. Consequently, Delta hedging is assigned a higher weight than Vega hedging.

An immediate inquiry that arises is the sensitivity of the results, as derived from the transfer learning algorithm presented in this study, to the weight settings of the multi-objective optimization. To address this, we conducted a robustness test. Initially, we diminished the weight for pricing, adjusting the weights to a 10:4:2 ratio, and then assessed any notable shifts in the implied volatility-median absolute error curve. Moreover, we augmented the weight for Vega hedging, calibrating the weights to a 10:2:0 ratio, and subsequently examined the implied volatility-median absolute error curve.

From the observations in Figure A.2, variations in the weight assignments of the source domain exert minimal impact on the error curve, confirming the robustness of our findings. The error curves for different weight configurations display remarkable overlap at several time nodes. The mean disparity between the error curve, when the source domain multi-objective learning weights are set at 10:4:2, and the primary outcome's error curve stands at a mere 2.51×10^{-4}. With the weights adjusted to 10:2:0, the average deviation between the error curve and the principal result is 7.37×10^{-3}. Neither of these values carry significant weight either economically or statistically. Thus, the outcomes of the transfer learning derivative pricing model display resilience against alterations in the source domain multi-objective learning weight settings. Although hedging and pricing might ostensibly appear as divergent tasks, each inherently informs the other. The gradient of the pricing objective function essentially forms the hedging objective function. Consequently, the weight parameter for multi-objective learning merely necessitates that the neural network duly emphasizes both aspects. Intriguingly, augmenting the emphasis on pricing errors in the source domain doesn't necessarily correspond to a reduction in out-of-sample pricing errors in the target domain. At first glance, this might seem paradoxical. How can the pricing error increase by 7.37×10^{-3} upon lessening the weight assigned to the vega hedging error? Conventionally, one might posit that if a neural network's primary aim is pricing, its loss function should exclusively account for pricing errors. Yet, our empirical observations underscore that the model's efficacy in the target domain is enhanced by engaging in multi-task learning within the source domain. Overlooking hedging factors in the source domain might inadvertently diminish the precision in pricing. This intertwined relationship can be attributed to the inherent nexus between pricing and hedging. The insights a neural network garners from the structured hedging model invariably aid its pricing endeavors.

We delve deeper into the selection of the model's evaluation metrics. Table A.1 systematically evaluates the influence of error assessment indicators and error weighting methodologies on the robustness of our findings.Both deep learning and transfer learning were trained using the original WMAE (Weighted Mean Absolute Error) as the loss function, but with a modification in the evaluation approach. This modification is intended to test the predictive performance of the models from multiple perspectives. As gleaned from the table, regardless of whether the metric is mean absolute error, percentage error, or mean square error, and irrespective of the presence or absence of error weighting, the efficacy of the transfer learning pricing network in enhancing pricing remains consistent. In terms of unweighted mean absolute error, transfer learning exhibits an error that is 1.95 less than that of deep learning. Considering the three outcomes for weighted mean absolute error, transfer learning experience error reductions ranging from 0.245 to 0.280, while the prediction error in deep learning decreases from 1.328 to 1.478. The relative improvements in pricing error for transfer learning to deep learning are recorded as 0.816, 0.754, and 0.869. Proportionally speaking, when compared to deep learning, the transfer learning's pricing error, as measured by average absolute error, witnesses a reduction by 76.7%, 71.0%, 70.6%, and 71.3% respectively.

During model training, the loss function employed in the neural network backpropagation is the weighted mean absolute error. Nevertheless, it's fundamentally feasible to use the mean square error (MSE) as an evaluation metric to gauge the model's robustness. Conceptually, utilizing a loss function that displays a more pronounced discrepancy relative to the backpropagation loss function can magnify potential overfitting issues in the model. In terms of MSE, the relative improvements in pricing error for transfer learning in comparison to deep learning are 94.0%, 88.2%, 87.3%, and 88.9% respectively.

Within the context of MSE, the transfer learning pricing network exhibits a pronounced superiority over its deep learning counterpart. This is attributable to the heightened overfitting issues inherent in deep learning, making alternative evaluation metrics beyond backpropagation more challenging for transfer learning. In terms of MAPE (Mean Absolute Percentage Error), the merits of transfer learning relative to deep learning are also readily discernible when considering mean absolute error. The MAPE for transfer learning is 91.0% lower compared to deep learning. The rationale behind this observation mirrors that of the MAPE context. MAPE, essentially an MAE normalized with the actual price, deviates from the primary backpropagation error which employs Delta normalization. Hence, transfer learning's performance remains more consistent when switching error assessment metrics.

Furthermore, in the main results, we advocate for the utilization of weighted mean absolute error for training, supplemented by a stability constant, to ensure backpropagation error's numerical stability and to accentuate the significance of OTM options. The stability constants in this context are deliberately designated. The findings presented in this table underscore that the configuration of these stability constants exerts a minimal influence on the outcomes. Moreover, our analyses indicate that, on a broader scale, both deep learning and transfer learning offer tangible advancements over stochastic volatility models. While deep learning might underperform in certain market scenarios, such as pronounced market volatilities or the repercussions of significant events like the new crown epidemic, its aggregate pricing error remains lower than that of the stochastic volatility model in the long-term perspective.

D.2 Alternative Neural Networks

The structure presented in this paper offers significant flexibility in the choice of neural network architecture, and its academic relevance remains undiminished irrespective of advancements in deep learning technology.

In exploring the types of neural networks, we've delved into two main aspects: adjusting the neural network's activation function and altering its structure. Initially, this study substitutes the LeakyReLU activation function with the ELU function in the primary results. Both ELU and LeakyReLU are variations of the ReLU function. While LeakyReLU enhances the gradient when the neuron input's linear sum is negative, ELU primarily aims to produce a more robust learning process by combining the properties of linear units for positive inputs and exponential units for negative inputs. The ReLU function is non-differentiable at x=0. In contrast, ELU closely resembles ReLU but helps to mitigate the vanishing gradient problem by ensuring that the function is differentiable at all points and smoothly transitions between the linear and exponential segments. Our objective is to discern how this smoother activation function impacts outcomes. In the subsequent figure, we examine the model's error outcomes and the core regression results from the attribution analysis after replacing all instances of LeakyReLU with the ELU function. In the error analysis presented in Figure A.3, ELU-Transfer learning exhibits similarities to the primary result's error curve. The mean deviation between ELU and LeakyReLU's error curves is 7.34×10^{-3}, economically negligible.

These findings suggest that the error curve remains robust amidst activation function alterations. Modifying the neural network's computational unit type doesn't influence the algorithm's economic rationale or empirical outcomes, aligning with our foundational hypothesis. This paper's core algorithmic design accentuates neural network performance by leveraging the inherent information of the economic model, which serves as an anchoring mechanism. Thus, the methodology here should be broadly applicable across diverse neural networks. However, this paper's discourse on neural network activation function selection is not exhaustive. Hypothetically, each neural network layer could feature distinct activation functions. Given the myriad activation function choices, a comprehensive discussion would entail evaluating K^N scenarios, where K represents the activation function candidate set and N signifies the neural network layers count. If we consider varying the layer count, the scenarios become theoretically infinite. Our dialogue here is not to enumerate all activation function permutations but to underline this paper's methodological universality.

The activation function within the neural network can be perceived as its micro-level attributes, while the overarching design of the network represents its macro-level characteristics. The realm of computer science presents a plethora of structural designs for this aspect. In our primary study, we employed a residual learning structure. Distinct from traditional deep learning algorithms, this structure incorporates direct channels between layers to mitigate the vanishing gradient dilemma. The network we discussed ensures a direct connection between any two consecutive layers. We also explored scenarios where some direct connections were omitted. In the absence of some direct channels, the network reverts to the simpler multi-layer perceptron design.

The outcomes of this configuration are detailed in Figure A.3. Empirical findings suggest model robustness irrespective of the neural network's architecture. Notably, a non-residual learning structure appears to underperform in time series analyses when juxtaposed against its residual learning counterpart, but by only 2.11×10^{-3} on average.

This uptick can be attributed both to the trimmed network layers and the intrinsic issues non-residual learning might introduce, such as vanishing or exploding gradients. However, even in scenarios where all layers don't incorporate residual learning, our algorithm consistently outperforms conventional techniques. This superior efficacy stems from a synergistic blend of knowledge- and data-driven methodologies, resulting in impressive pricing capabilities. Concurrently, it's pivotal to recognize the invaluable contributions from the computer science domain. Superior network architectures also tend to exhibit enhanced performance under the umbrella of transfer learning. This insight further accentuates that our transfer learning-based derivatives pricing algorithm benefits from the integration of advanced artificial neural network techniques.

Parallel strategies can be seamlessly applied across a variety of network architectures, including LSTM, traditional multi-layer perceptrons (MLP), GRU, convolutional neural networks (CNN), self-attention mechanisms, Transformers, and other architectures. These strategies enable effective implementation across diverse neural network structures to address the needs of option pricing and hedging. Even as the computer science community introduces more sophisticated neural network structures, future scholars can leverage the methodology articulated in this paper to devise transfer learning models for pricing and hedging based on these novel architectures, potentially achieving superior results.

References

Almeida, C., R. Fan, R. Freire, and X. Tang. 2023. Can a machine correct option pricing models? *Journal of Business & Economic Statistics* 41:123–37.

Andersen, T. G., N. Fusari, and V. Todorov. 2017. Short-term market risks implied by weekly options. *Journal of Finance* 72:1335–86.

Bali, T. G., H. Beckmeyer, M. Moerke, and F. Weigert. 2023. Option return predictability with machine learning and big data. *Review of Financial Studies* 36:3548–602.

Bianchi, D., M. Büchner, and A. Tamoni. 2021. Bond risk premiums with machine learning. *Review of Financial Studies* 34:1046–89.

Black, F., and M. Scholes. 1973. The pricing of options and corporate liabilities. *Journal of Political Economy* 81:637–54.

Bryzgalova, S., V. DeMiguel, S. Li, and M. Pelger. 2024. Asset-pricing factors with economic targets. Working paper, London Business School.

Bryzgalova, S., M. Pelger, and J. Zhu. 2023. Forest through the trees: Building cross-sections of stock returns. Working paper, London Business School.

Campello, M., L. W. Cong, and L. Zhou. 2024. A data-driven-robust-control approach to corporate finance and ai-guided managerial actions. Working paper.

Cao, Y., X. Liu, and J. Zhai. 2021. Option valuation under no-arbitrage constraints with neural networks. *European Journal of Operational Research* 293:361–74.

Chen, H., A. Didisheim, and S. Scheidegger. 2023. Deep surrogates for finance: With an application to option pricing. Working paper.

Chen, L., M. Pelger, and J. Zhu. 2024. Deep learning in asset pricing. *Management Science* 70:714–50.

Chen, X., and S. C. Ludvigson. 2009. Land of addicts? an empirical investigation of habit-based asset pricing models. *Journal of Applied Econometrics* 24:1057–93.

Chen, X., and H. White. 1999. Improved rates and asymptotic normality for nonparametric neural network estimators. *IEEE Transactions on Information Theory* 45:682–91.

Chinco, A., A. D. Clark-Joseph, and M. Ye. 2019. Sparse signals in the cross-section of returns. *Journal of Finance* 74:449–92.

Cong, L. W., K. Tang, J. Wang, and Y. Zhang. 2022. Alphaportfolio: Direct construction through deep reinforcement learning and interpretable ai. Working paper, Cornell University.

DeJong, D., B. F. Ingram, and C. Whiteman. 1993. Analyzing vars with monetary business cycle model priors. In *Proceedings of the American Statistical Association, Bayesian Statistics Section*, vol. 160, 69.

Del Negro, M., and F. Schorfheide. 2004. Priors from general equilibrium models for vars. *International Economic Review* 45:643–73.

Doan, T., R. Litterman, and C. Sims. 1984. Forecasting and conditional projection using realistic prior distributions. *Econometric reviews* 3:1–100.

Feng, G., Z. He, N. G. Polson, and J. Xu. 2023. Deep learning in characteristics-sorted factor models. *Journal of Financial Economics* 148:601–22.

Freyberger, J., A. Neuhierl, and M. Weber. 2020. Dissecting characteristics nonparametrically. *Review of Financial Studies* 33:2326–77.

Garcia, R., and R. Gençay. 2000. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics* 94:93–115.

Gu, S., B. Kelly, and D. Xiu. 2020. Empirical asset pricing via machine learning. *Review of Financial Studies* 33:2223–73.

Hanin, B. 2019. Universal function approximation by deep neural nets with bounded width and ReLU activations. *Mathematics* 7:992–.

He, K., and J. Sun. 2015. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5353–60.

He, K., X. Zhang, S. Ren, and J. Sun. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–8.

———. 2016b. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, 630–45. Springer.

Heston, S. L. 1993. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies* 6:327–43.

Hornik, K., M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2:359–66.

Hutchinson, J. M., A. W. Lo, and T. Poggio. 1994. A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance* 49:851–89.

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* 34:497–510.

Kelly, B., and S. Pruitt. 2013. Market expectations in the cross-section of present values. *Journal of Finance* 68:1721–56.

Kozak, S., S. Nagel, and S. Santosh. 2023. Shrinking the cross-section. *Journal of Financial Economics* 147:335–62.

Litterman, R. B. 1986. Forecasting with bayesian vector autoregressions—five years of experience. *Journal of Business & Economic Statistics* 4:25–38.

Raissi, M., P. Perdikaris, and G. E. Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* 378:686–707.

Rapach, D., and G. Zhou. 2013. Forecasting stock returns. In *Handbook of economic forecasting*, vol. 2, 328–83. Elsevier.

Srivastava, R. K., K. Greff, and J. Schmidhuber. 2015. Training very deep networks. *Advances in Neural Information Processing Systems* 28.

Zhuang, F., Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE* 109:43–76.