Teaching Economics to the Machines

Hui Chen Yuhan Cheng Yanchu Liu Ke Tang*

September 12, 2024

Abstract

While structural models in economics can offer valuable insights, they 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 tend to struggle to generalize beyond the confines of training data. We propose a novel framework that incorporates economic restrictions from a structural model into a machine learning model through transfer learning. Specifically, we first construct a neural-network representation of the structural model by training it on the synthetic data generated by the structural model, and then fine-tune the network using real data. In an application to option pricing, the transfer learning model significantly outperforms both the structural model and a conventional data-driven deep neural network. The out-performance is more significant when the sample size of real data is small or under volatile market conditions.

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, Jun Tu, Guofu Zhou, Dacheng Xiu, Qunzi Zhang and participants at the Econometric Society Summer School in Lausanne, CICF 2023, AMES 2023, SIF 2024 conferences, Tsinghua University, Hong Kong University of Science and Technology (Guangzhou), Central University of Finance and Economics, and Renmin University of China 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, make predictions, and make informed decisions. However, the combination of big data and advances in machine learning techniques presents data-driven approaches as a new way of understanding the world without relying on traditional models. It makes one wonder, "Is theory dead?"[1]

In fact, the debate about the role of theory has long existed in economics. Structural models in economics can convey appealing insights but tend to be scarcely parameterized and offer unsatisfactory fitting for real data. On the other hand, reduced-form models can offer significantly more flexibility and superior forecasting performances, which often makes 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 issues, especially when the sample size is limited and 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 structural model, despite its imperfections, can still be useful in that it provides structural restrictions among variables which can help regularize the training of a reduced-form model and alleviate the overfitting problem. Moreover, thanks to its causal structure, the restrictions provided by the structural model can be readily extended outside the training data boundaries, which can potentially 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 "source domains") to improve the performance in a target setting (known as "target domain"). In our case, the source domain is generated by the structural model while the real data reside in the target domain, and transfer learning allows us to combine the information from the structural

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

model and real data, without treating the potentially misspecified structural restrictions as hard constraints. The implementation is straightforward: we first construct a neural-network representation of the structural model by training it on synthetic data generated by the structural model, and then fine-tune the network using real data.

Guaranteed by the universal approximation theorem of neural networks (see e.g., Hornik, Stinchcombe, and White, 1989), training in the source domain with an arbitrarily large set of simulated data can result in an accurate neural-network representation of the structural model, which inherits all the economic restrictions implied by the theory (Chen, Didisheim, and Scheidegger, 2022). The fine-tuning step then allows the neural network to "move away" from theory and incorporate information from real data by using a relatively low learning rate (i.e., updating parameters in small steps).

The sequential nature of transfer learning has two desired implications. First, the resulting hybrid model no longer satisfies the structural restrictions exactly, especially when these restrictions appear to be inconsistent with the data. This approach differs from training machine learning models 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. This distinction is crucial. Economic structural models are often more susceptible to misspecification compared to their physical counterparts. Imposing these misspecified restrictions as hard constraints can introduce more biases into the results.

Second, the hybrid model does not simply average the information embedded in synthetic data and real data. One may somehow view the neural network trained on the synthetic data as providing an informative prior for the second stage of learning on real data. This is roughly similar to Bayesian vector autoregressions (BVAR) in a highly conceptual level, which imposes informative priors to help estimate the VAR parameters; the 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). However, substantially different from the Bayesian framework, while increasing the amount of synthetic data in the source domain can help the neural network capture the structural model restrictions

more accurately, the latter only provides a starting point for training in the target domain. In particular, the notion of a "strong prior" does not apply. How much the final model parameters move away from this starting point depends on the gap between the structural model and the true DGP, the size of real training data, as well as the hyperparameters for target domain training, including the learning rate and epochs. Consequently, unlike the previously observed inverse U-shaped relationship between the performance of BVAR models and the relative sample size of synthetic versus real data (see e.g., Del Negro and Schorfheide, 2004), we show that the performance of the transfer learning model tends to increase monotonically with the size of synthetic data.

As an example application, we apply the transfer learning framework to option pricing. We use the Black-Scholes option-pricing model (Black and Scholes, 1973) to generate synthetic data in the source domain. Despite ample evidence on the empirical limitations of this model, we choose it for its simplicity as well as to successfully illustrate the point that even structural models that are clearly misspecified can still be helpful in the transfer learning framework. Empirically, a wide variety of potential features can be used to predict option prices that go far beyond the relevant state variables in a structural model.[2] When generating the synthetic data, we draw the complete set of features from a multivariate uniform distribution but compute the Black-Scholes option prices using only the subset of relevant features. The remaining features are effectively treated as noise in the source domain.

We train a feedforward network with 17 hidden layers, 22 neurons for each layer (with a total of over 8,000 trainable parameters). We use multi-target learning in the source domain to better capture the structural restrictions, with the loss function taking into account not only pricing errors (weighted mean absolute errors for Black-Scholes-implied volatility, or IV-MAE), but also the MAE for option delta and vega. In the target domain, the loss function is entirely based on pricing errors (IV-MAE). We train and evaluate the model in the target domain using rolling windows, each time using 9 months of real data as the training set, the data in the next 3 months as the test set. We then compare the performance of the

[2]In the case of the Black-Scholes model, the only relevant features include the stock price, strike price, time-to-maturity, risk-free rate, and volatility. However, other features that could be empirically relevant include past returns on the option and the underlying asset, option open interest, trading volume, bid-ask spreads, etc.

transfer learning model (TL) against that of a deep neural network with identical architecture but does not have information from the structural model (referred to as DNN), as well as the Heston model (Heston, 1993), which extends the Black-Scholes model by adding stochastic volatility.

In the out of sample setting, the transfer learning model significantly outperforms both DNN and the Heston model in pricing during the period from 2001 to 2021. The average IV-MAE for the TL model in the entire sample is about 23.8% of that for DNN and 14.4% of that for the Heston model. In the cross section, we show that the performance advantage of TL relative to DNN is more significant for options with longer time-to-maturity and lower bid-ask spreads, for which the structural model is arguably closer to the true DGP. In the time series, the performance advantage of TL is larger during periods of market turmoil, measured by high market volatility (VIX) and high left tail volatility (Bollerslev, Todorov, and Xu, 2015), as well as during periods when the distance between the input features and the training set distribution is wider (measured by the Mahalanobis distance). These results are consistent with the interpretation that structural model restrictions become more helpful when instability in the underlying DGP reduces the generalizability of DNN. Moreover, using feature importance analysis, we can readily identify the features that are left out of the structural model but still important for forecasting, which can help guide our efforts to improve the theory.

We perform several additional experiments to isolate the sources of performance improvement for the transfer learning model. Simply expanding the training sample size to the beginning of the full sample (i.e., switching from rolling window to expanding window) or pooling the synthetic data and real data to train the DNN does not meaningfully improve performance. The former is likely due to the instability of the underlying DGP, which requires putting more weight on recent data. The latter highlights the importance of the two-step training used by TL. Next, we mimic the approach used in physics-informed neural networks by directly imposing structural model restrictions in the training of the DNN (we do so by making pricing errors relative to the Black-Scholes model part of the loss function). This does not yield better results, either. We also replace the Black-Scholes model in the TL framework with model-free no-arbitrage restrictions on European options. The resulting

performance improvement over DNN is much smaller. This result suggests that, when the structural restrictions are too weak, they become less effective in regularizing the training of the DNN in real data.

Finally, it is worth emphasizing 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 Our literature review is structured around four key areas: the use of structural economic model constraints in statistical models; constraints within financial machine learning models; the application of machine learning in finance, particularly in option pricing; and the theoretical frameworks that underpin option pricing models. This comprehensive approach allows us to situate our study within the broader scope of current financial econometrics and machine learning advancements.

The classical structural economic model constraints in statistical models are an application of BVAR. Bayesian Vector Autoregression (BVAR) models enhance classical vector autoregression by integrating prior information based on economic theories, thereby improving forecast accuracy and stability. The foundational Minnesota Prior (Doan, Litterman, and Sims, 1984; Litterman, 1986) focuses on reducing overfitting in high-dimensional models through coefficient shrinkage. To make priors more applicable to macroeconomic research, some scholars have considered optimizing the Minnesota Prior, such as using time-domain specifications.Others have opted for different types of prior distributions, such as using DSGE models as prior information for predicting macroeconomic variables.

The transfer learning tool used in this paper is distinct from Bayesian methods. Empirical results show that the performance of Bayesian methods exhibits a U-shaped curve in relation to the weight ratio of prior to posterior data, whereas transfer learning shows a monotonically increasing trend, not suffering performance loss due to excessive prior information. Moreover, while the BVAR method can only enhance VAR, a linear statistical tool, transfer learning

is capable of improving neural networks, which are nonlinear models. This is not merely a parallel effort to BVAR's approach, as neural networks possess the ability to approximate any continuous function, a capability that VAR lacks.

The appeal of taking advantage of "domain knowledge" when applying machine learning methods in economics and finance should be quite obvious. Indeed, it is standard practice to use economically-motivated features and feature engineering when building models for forecasting (e.g., when predicting asset returns or doing credit scoring). For example, Garcia and Gençay (2000) show that the performance of neural networks can be improved by exploiting the homogeneity in the option pricing formula. Our framework goes beyond identifying relevant state variables and utilizes the structural restrictions imposed on their joint dynamics. The no-arbitrage condition is a commonly used benchmark constraint.

Our paper also contributes to the literature on the application of machine learning methods in finance, especially derivative pricing. An early study in this field is the research of Hutchinson, Lo, and Poggio (1994), which highlights the advantages of this data-driven approach while noting that these models necessitate substantial data for effective training. The emphasis on nonlinear machine learning models, especially neural networks and deep learning, represents a significant progression in comprehending complex market dynamics. Research by Gu, Kelly, and Xiu (2020) and others have shown that these models excel at delivering accurate financial forecasts and effectively navigating the intricacies of financial markets. Moreover, AI's role in evaluating market efficiency and effectiveness, as explored by (**?**) and (**?**), offers a fresh lens through which the regulatory mechanisms of financial markets can be understood, providing a comprehensive view of how advanced analytical tools can enhance our understanding of market behaviors. Other works in this area include Anders, Korn, and Schmitt (1998), Amilon (2003), Bennell and Sutcliffe (2004), Andreou, Charalambous, and Martzoukos (2006), Amornwattana, Enke, and Dagli (2007), Andreou, Charalambous, and Martzoukos (2008), Chen and Sutcliffe (2012), Culkin and Das (2017), Buehler et al. (2019) and Cao, Chen, and Hull (2020).

Some of the literature highlights the benefits of machine learning, yet there are also views that stochastic differential equation models can outperform even the best machine learning models. For example, Jang and Lee (2019), who systematically compare deep

learning and structural models[3], conclude that some relatively simple stochastic differential equation models can outperform even the best machine learning models. In the domain of options pricing, machine learning still struggles to completely surpass traditional models. We are still curious to know how to enhance the performance of these models. Although the advantages of incorporating information from stochastic differential equation-based pricing models into machine learning training processes have been recognized, there is currently no general method to address this issue.

The final section of literature relevant to our research pertains to option pricing theories and models. Option pricing theories traditionally utilize stochastic differential equation models, which base their calculations on the dynamics of underlying assets, guided by no-arbitrage conditions. Option pricing theories have evolved significantly since the seminal work of Black and Scholes (1973), which introduce the use of stochastic differential equations to model the dynamics of underlying assets based on no-arbitrage principles. This foundational approach has been extended and refined in numerous ways to better capture the complexities of financial markets. Early expansions, such as the introduction of jump processes, addresses the need to model sudden market movements and discontinuities in asset returns. Subsequent developments recognize the variability of volatility. Heston (1993) pioneers stochastic volatility models to reflect the fluctuating nature of volatility over time.

Our research not only involves certain modifications to option pricing theoretical models from a nonlinear perspective through feature importance analysis, but also provides a new approach for the application of these models. We confirms that these theoretical models can be effectively applied in the framework of transfer learning to provide training sets for the source domain, thus guiding the training of the models effectively.

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

[3]Jang and Lee (2019) examine BS-Cal, BW, KR, LSM, LV, SVJ, and SVM models. They also consider four types of error measurement, including MAE, MAPE, MPE, and MSE.

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 Framework

Denote by $\mathcal{X}$ and $\mathcal{Y}$ the input and target space, respectively, with unknown probability distributoin $\mathbb{P}_{(\mathcal{X},\mathcal{Y})}$ 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 $S = ((x_i, y_i))_{i=1}^m$.

Next, consider a theoretical model that imposes restrictions between x and y. The model could involve hidden states h (in contrast to x and y, which are directly observable) and parameters θ, and we assume that these restrictions can be represented by

$$E[y|x, h] = g(x, h|\theta). \tag{1}$$

Thus, g is the best forecast for y given x and h. The theoretical restrictions could involve only a subset of x and/or y. For example, due to a preference for parsimony , economic models tend to involve a small subset of potential features. Notice that the theoretical model could be misspecified, in the sense that (1) might be inconsistent with $\mathbb{P}_{\mathcal{Y}|\mathcal{X}}$.

When the model f is sufficiently parsimonious, we can combine the information from the theoretical model and data by imposing the theoretical restrictions when estimating the parameters in f with actual data (for example, we typically impose cross-equation restrictions when estimating rational expectations models; see e.g., Hansen and Sargent 1980, 1991). Alternatively, we can adopt a Bayesian approach, using the theoretical restrictions to form an informative prior for the parameters in f and then update it with data. A concrete example of this Bayesian approach is in the context of macro forecasting with vector autoregressions (VAR). Building on the work of DeJong, Ingram, and Whiteman (1993) and Ingram and

Whiteman (1994), Del Negro and Schorfheide (2004) propose to use a DSGE-model-based prior for the VAR estimation, where the prior can also be interpreted as a posterior that is formed based on a sample of observations simulated from the DSGE model.

However, these approaches are only feasible for parsimonious parametric models. For the tractability of the Bayesian approach, additional restrictions are needed to make the theory-implied prior and the likelihood function conjugate. Our method based on transfer learning allows us to work with a significantly more flexible class of models (including neural networks). It involves two steps. First, in the source domain, we simulate data for $\mathbf{x}, \mathbf{y}$ from the theoretical model (e.g., by first drawing $\mathbf{x}$ from some distribution and then drawing $\mathbf{y}$ from $\mathbb{Q}_{\mathcal{Y}|\mathcal{X}}$), and then fit $f(\cdot)$ to the simulated data by minimizing a certain loss function. This is reminiscent of the Bayesian approach above, which uses simulated data from the theoretical model to form an informative prior for the parameters of f.

Next, in the target domain, we take the model trained from the source domain as the starting point and fine-tune it with real data. Again, this step is similar in spirit to the Bayesian approach of updating the informative prior with real data. Like the Bayesian approach, we do not fully embrace the theoretical model. By updating the informative prior with data, the Bayesian approach is effectively using the information from the theoretical model to guide the search for the parameters in the data.We do the same in the target domain by using the parameter values from the source domain as the starting point and then updating them with real data.

The existence of hidden parameters θ in the economic model does not substantially affect this algorithm. In the source domain, we randomly generate θ and consider θ and x together as inputs. During the training and testing in the target domain, we estimate θ in advance using the training set data (for example, through nonlinear least squares), after which normal training and out-of-sample testing can be conducted.

It is easy to see that if the theoretical restrictions between $\mathbf{x}$ and $\mathbf{y}$ are valid, in the sense that $\mathbb{Q}_{\mathcal{Y}|\mathcal{X}}$ is consistent with $\mathbb{P}_{\mathcal{Y}|\mathcal{X}}$, imposing such restrictions should help one learn about $\mathbb{P}_{\mathcal{Y}|\mathcal{X}}$ and make the learned model $\hat{f}$ more accurate out of sample. However, a misspecified theoretical model could also be useful. Although model misspecification will tend to increase the biases in forecasts, imposing such restrictions can also reduce the variance of the learned

model. The out-of-sample performance of $\hat{f}$ depends on the tradeoff between these two forces.

In general, an economic model defines the mapping from $g : R^m \to R^k$. The mapping maps m observable economic variables to k observably economic variables or parameters to be estimated. This mapping can be non-linear. This mapping may also not have an analytical solution, the following algorithm allows flexible use of numerical methods. In the framework of artificial intelligence, big data is often applied, so we assume without loss of generality that there are h>m observable economic variables. A neural network defines a mapping $F : R^h \to R^k$. The motivation is to teach the network the knowledge we know before neural network training is with data. A neural network with L layers is denoted as $F(L; \sigma_1, \sigma_2, \sigma_3, ..., \sigma_L; W_1, W_2, W_3, ..., W_L)$ in which σ_i is an activation function. W_k is the weight matrix of layer k. The output layer I of the corresponding neural network is

$$O_i = \sigma_i(w_i I_i + b_i) \tag{2}$$

$$W_i = [w_i; b_i] \tag{3}$$

$$O_{i-1} = I_i \tag{4}$$

Let X_i be the input of the pricing model and y_i be the corresponding output. We hope that $F(L; \sigma_1, \sigma_2, \sigma_3, ..., \sigma_L; W_1, W_2, W_3, ..., W_L)(X_i) \approx y_i$ (However, simply training F on X_i, y_i might face insufficient data problem, overfitting and noisy data). Specifically, the input of the pricing model is: $X_i = (\mathcal{M}_i, \mathcal{Z}_i)$. $\mathcal{M}_i$ is the input of economic model g. We want to constrain our neural network model through the economic model to ensure that the neural network does not seriously deviate from the common sense of economics and at the same time ensure that the neural network can flexibly modify the functional form within a certain range. Intuitively, we restrict $F(\mathcal{M}_i, \mathcal{Z}_i)$ to be not too far from $g(\mathcal{M}_i)$.

2.2 Source Domain

In the source domain we try to obtain a neural network representation of the economic model. This is supported mathematically. A sufficiently deep neural network can approximate arbitrary functions. We first randomly generate training samples within R^m, and calculate $g(M_i)$ through the economic model. It should be noted that the calculation of some economic

models is relatively cumbersome, so generating $g(M_i)$ may be time-consuming. However, the training of the source domain only needs to be performed once, so the method in this paper does not exclude highly computationally intensive economic models. By the same logic, we don't need the model to have an analytical solution.

The following article focuses on explaining how we use neural networks to extract information from stochastic differential equation models. Assume that for a stochastic differential equation model, there is a numerical solution $g(M_i)$ for the option pricing problem.

Here we hope that the neural network can learn the information of the economic model and its derivative level according to the differentiability of the economic model. In practical applications, the input dimension of the economic model may be high, and we can select the derivative of a specific variable of interest to be included in the training.

In the source domain, we train the neural networks to estimate W_i as:

$$\widehat{W_1}, ..., \widehat{W_L} = argmin(\lambda_0 L_0 + \sum_{j=1}^{\#X_i} \lambda_j L_j) \tag{5}$$

in which:

$$L_0 = \sum_{i=1}^{N} |F(L; \sigma_1, \sigma_2, \sigma_3, ..., \sigma_L; W_1, W_2, W_3, ..., W_L)(X_i) - g(\mathcal{M}_i)| w_{i0} \tag{6}$$

$$L_j = \sum_{i=1}^{N} |\frac{\partial F(X_i)}{\partial X_{ij}} - \frac{\partial g(\mathcal{M}_i)}{\partial X_{ij}}| w_{ij}, \forall j > 0 \tag{7}$$

with randomly initialized initial values of matrix W_i.

2.3 Target Domain

The target domain aims to transfer information learned from the source domain to the domain concerning real-world economic modeling. The training of the transfer learning pricing network in the target domain is a fine-tuning method based on the source domain information. Intuitively, neural networks face a noisy data generation process in empirical data, and fully fitting the data will potentially overfit and reduce out-of-sample pricing power. Specifically, the transfer learning pricing network uses a lower learning rate and epoch number

in the training of the target domain than source domain. The training on the target domain uses the weight of the source domain as a starting point.

$$\widetilde{W}_1, ..., \widetilde{W}_L = argmin \sum_{i=1}^{N} |F(L; \sigma_1, ..., \sigma_K, \sigma_{K+1}, ..., \sigma_L; W_1, ..., W_L)(X_i) - y_i| \, v_i \tag{8}$$

with initial value $\widehat{W}_1, \widehat{W}_2, ..., \widehat{W}_L$. The function $F(M; \sigma_1, ..., \sigma_K, \sigma_{K+1}, ..., \sigma_L; \widetilde{W}_1, ..., \widetilde{W}_L)$ is our final (target domain) pricing model. In general, one can choose an L and replace the layers L+1 to K before the source domain training starts with new M-L layers neurons. This approach aims to discard part of the information of the training model in the source domain. That is:

$$\begin{aligned} &\widehat{W}_1, ..., \widehat{W_M} = \\ &argmin \sum_{i=1}^{N} |F(L; \sigma_1, ..., \sigma_K, \widehat{\sigma}_{K+1}, ..., \widehat{\sigma}_M; W_1, ..., W_M)(X_i) - y_i| v_i \end{aligned} \tag{9}$$

with initial values $\widehat{W}_1, \widehat{W}_2, ..., \widehat{W}_K$. $\widehat{\sigma}_{K+1}$ is a different neuron layer from σ_{K+1}.It potentially avoids the transfer learning pricing network from falling into a local optimum near the optimal weight of the source domain, at the cost of information loss for the source domain model. It drops the information from layer K to layer L. If L is not smaller than K, no layer is replaced and in this particular case, the model becomes the previous case. Intuitively, the larger the gap between the source domain task and the target domain task, the more layers we should reset. In the empirical results below, for the judgment that the stochastic differential equation model is closer to the real model, we set $L = K$. Empirical results will reveal that this option is sufficiently good,

This method reveals the occurrence of *fragile co-adaptation*, which means the neurons on neighboring layers co-adapt during training. It should be noticed that a classical frozen

treatment transfer learning algorithm is built as follows

$$\widehat{W_{K+1}}, ..., \widehat{W_M} =$$
$$argmin \sum_{i=1}^{N} w_i |F(L; \sigma_1, ..., \sigma_K, \widetilde{\sigma}_{K+1}, ..., \widetilde{\sigma}_M; W_1, ..., W_M)(X_i) - y_i| \tag{10}$$

The transfer learning framework of the frozen method freezes the first K layer neurons of the source domain model and only updates the weights of the neurons after the K layer. This freezing method is preferred by some industrial application scenarios because it can update parameters less and has a higher training speed. We choose the fine-tuned method other than the frozen method because the literature shows the frozen method performs not as well as f-t for the reason of fragile co-adaption. The neurons should be ca-adapting with each other and the neurons of the last frozen layer and the first non-frozen layer may not ca-adapt well. The function $F(L; \sigma_1, ..., \sigma_K, \widetilde{\sigma}_{K+1}, ..., \widetilde{\sigma}_L; \widetilde{W}_1, ..., \widetilde{W}_L)$ is our final (target domain) model.

3 An Application to Option Pricing

For option pricing, since traditional pricing models (BS, CEV, etc.) have certain pricing power, it is unnecessary to perform pure data-driven deep learning without effectively using the information of traditional models. On the other hand, due to the characteristics of this SDE-like method (such as it is difficult to solve under the assumption of complex underlying asset prices) or unable to grasp the complex, high-dimensional factor, highly non-linear process. Therefore, SDE model needs to be improved by deep learning methods. And transfer learning can effectively transfer information between two related tasks, namely source domain and target domain.

We use LeakyRelu as an activation function in this paper. The case of other activation functions will be discussed in the robustness check. Specifically, the input of the pricing model is: $X_i = (S_i, K_i, T_i, vol_i, d_i, r_i, \mathcal{Z}_i)$, known as stock price, strike price, time to maturity, volatility, dividends, risk-free rate, and other characteristics.Z_i is a flexible vector to denote characteristics concerned.

3.1 Surrogate Models

Figure 6 illustrates the process of applying transfer learning to option pricing, including the training of both the source and target domains. The neural network in the source domain aims to obtain information from a financial model based on stochastic differential equations. This information extraction is not limited to a particular stochastic differential equation, nor does it place specific requirements on how the stochastic differential equation model is solved. This operation allows stochastic differential equation models without analytical solutions. If the model can only be solved numerically, there are no further requirements on the method used to obtain the numerical solution. Whether it is through partial differential equation solving or Monte Carlo for the underlying stochastic process, it can be incorporated into this framework.

In empirical finance's time window rolling calculation, the source domain training still only needs to be performed once. This is because although the parameters of the stochastic volatility model are time-varying, the solution of the stochastic volatility is not. It only needs to be done once, which means that the model of the source domain can consume a huge amount of computing power without affecting the empirical feasibility of the algorithm in this paper. This point has a huge advantage over directly using the stochastic volatility model combined with numerical methods for dynamic fitting. Rolling numerical methods using partial differential equations or Monte Carlo for underlying stochastic processes can be computationally expensive. In practical applications, it is necessary to balance accuracy and calculation time due to the pressure of the transaction cycle. Under the framework of this paper, the target domain of dynamic training is only fine-tuned. Although the source domain consumes computing power, it only needs to be done once. In summary, the main computing power consumption of the framework of this paper is carried out in advance.

The following article focuses on explaining how we use neural networks to extract information from stochastic differential equation models. Assume that for a stochastic differential equation model, there is a numerical solution g(x) for the option pricing problem.

In the source domain, we train the neural networks to estimate W_i as:

$$\widehat{W_1}, ..., \widehat{W_L} = argmin(\lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3), \tag{11}$$

in which:

$$L_1 = \sum_{i=1}^{N} \left| \frac{1}{|\delta_i| + \epsilon_c} \left(F(L; \sigma_1, \sigma_2, \sigma_3, ..., \sigma_L; W_1, W_2, W_3, ..., W_L)(X_i) - g(\mathcal{M}_i) \right) \right| \tag{12}$$

$$L_2 = \sum_{i=1}^{N} \left| \frac{\partial F(X_i)}{\partial S} - \frac{\partial g(\mathcal{M}_i)}{\partial S} \right| \tag{13}$$

$$L_3 = \sum_{i=1}^{N} \left| \frac{\partial F(X_i)}{\partial vol} - \frac{\partial g(\mathcal{M}_i)}{\partial vol} \right| \tag{14}$$

with randomly initialized initial values of matrix W_i. The $g(\mathcal{M}_i)$ is the output of an SDE model with input and parameter vector of $\mathcal{M}_i$.ϵ_c is a small constant number to ensure numerical instability. In deep NN training, the initial value is important because it contains information and dramatically affects the optimum parameters searched and convergence rate. After source domain training, the AI knows the basic idea of pricing and hedging.

The stochastic differential equation pricing model is characterized by inputs that comprise both observable features and unobservable hidden state variables. We denote this mathematical relationship as $\mathcal{M}_i = (\mathcal{MO}_i, \mathcal{MU}_i)$. Observable features can include elements like the exercise price and risk-free interest rates. On the other hand, hidden state variables can incorporate factors like the alpha in the SABR model and the elasticity in the CEV model, among others.In practical implementation, if needed, for every 3-month data, $\mathcal{MU}_i$ is kept constant and is obtained through the least squares method from the preceding 3 months. We utilize the Black-Scholes (BS) model as the source domain for our model. This choice is motivated by two main reasons: firstly, the Black-Scholes model is arguably the most classic framework in the field of option pricing. Secondly, we aim to demonstrate that even a simple structural model can significantly enhance empirical performance.

We weigh the pricing error by $1/\delta$ to ensure the TL and DL pay sufficient attention to OTM options.OTM options are with prices significantly smaller than ITM ones. We hope our models treat every contract equally. ϵ_c is a positive constant to ensure numerical

stability during the training process. Without this term, the contract's pricing error in NN back-propagation with a delta closest to 0 will be almost the entire loss function. Note that NN is initialized randomly, so in the earlier epochs the pricing error of options with prices close to 0 might not be applied with pretty large weight and dominates the loss function. Thus numerical stability is needed.

Another model set-up concern is multi-target learning in the source domain. The first point is we want the NN to be more smooth. We will show the benefit in the empirical part. Another reason is there may be potential benefits when the target domain loss is on pricing alone. In the target domain, the true price is observable while true greeks are not. So in pricing NN we only concentrate on pricing in the target domain. The multi-target training may enroll more information in the source domain.

In the source domain, we know the true model without any noise so we torture our model as soon as possible. The model is trained with numerous epochs and large learning rates without any concern for noise. The model is designed to ignore the information of input that is not the SDE model input.

3.2 Tuning From Real World Option Data

The target domain aims to transfer information learned from the source domain to the domain concerning real-world pricing and hedging.

$$\widetilde{W}_1, ..., \widetilde{W}_L = argmin \sum_{i=1}^{N} \left| \frac{1}{|\delta_i| + \epsilon_c} \left(F(L; \sigma_1, ..., \sigma_K, \widetilde{\sigma}_{K+1}, ..., \widetilde{\sigma}_L; W_1, ..., W_L)(X_i) - P_i \right) \right| \quad (15)$$

with initial values $\widehat{W}_1, ..., \widehat{W}_L$, respectively.

It should be noted that we face a low information-noise ratio market, so we are not as confident as we are in the source domain. We want our model to be robust to noisy data so we cautiously train the model with small epochs and a low learning rate. The method is designed to deny overfitting the noise. In contrast, there is no noise in the source domain data because we know the true DGP.

3.3 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 (2016) 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 $G(a)$ denote 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:

$$b = G(a, \{V_j\}) + a \tag{16}$$

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, 2016; **?**), 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 6. Detailed network parameters are provided in Appendix A.

4 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 and hedging tasks. The model is trained by using 9 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, including learn rate, stopping strategy, train set size, rolling window length, activation function per neural, and structure of the neural network, with our transfer learning model. Moreover, the two models are trained under the same structure and hyper-parameters.

4.1 Data

We sourced daily transaction data on S&P 500 index put options from OptionMetrics, covering the period from January 2, 2001, to December 31, 2021. Table 1 presents the descriptive statistics for this dataset, encompassing a total of 12,857,068 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.

4.2 Pricing Performance

The pricing discrepancy for option i at instance t is articulated as:

$$\tilde{\epsilon_{it}} = |\sigma(P_{it}; K_i, T_{it}, r_i, S_t, d_i) - \sigma(\hat{P_{it}}; K_i, T_{it}, r_i, S_t, d_i)|, \tag{17}$$

Where the function $\sigma(\cdot; K_i, T_{it}, r_i, S_t, d_i)$ maps prices to the implied volatility for day t and contract i. The aggregate pricing deviation of the model is encapsulated by the median absolute error (MAE) across all samples, represented as $\overline{|\epsilon_{it}|}$. It's worth highlighting that we derive σ_{it} utilizing the stochastic gradient descent technique. Concurrently, the Heston model, the deep learning framework, and the transfer learning model are all trained on a uniform dataset spanning from 2001 to 2021.

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

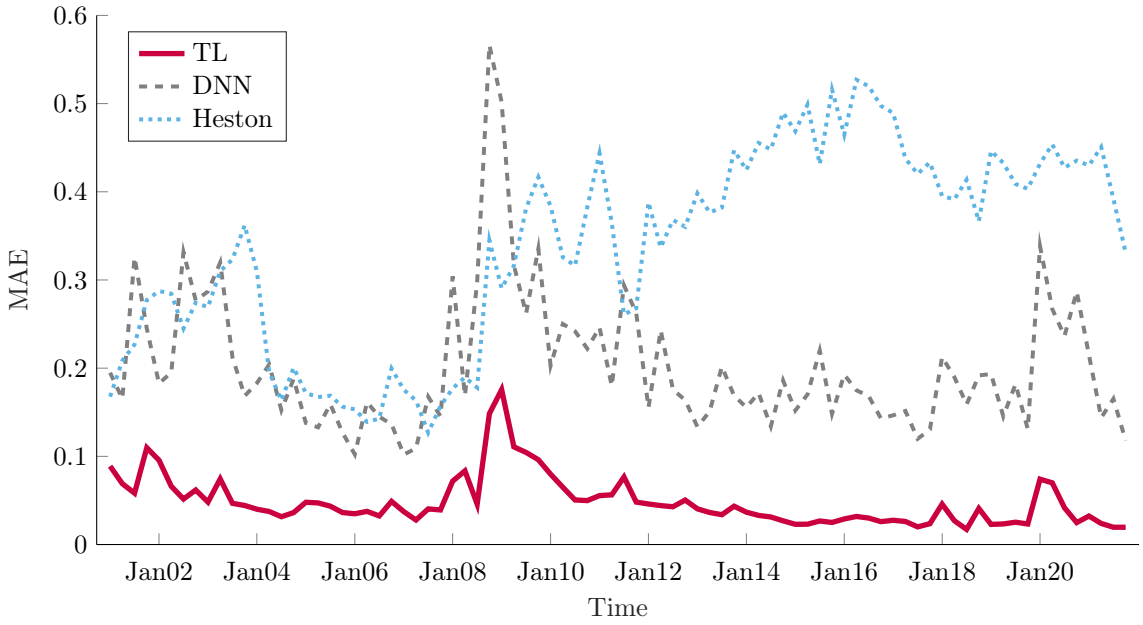


Figure 1: **Out-of-sample pricing errors.** This figure reports the out-of-sample mean-absolute pricing errors (in terms of BSIV) for TL, DNN, and the Heston model at quarterly frequency from 2001/01 to 2021/12. At the end of each quarter, the models are trained using data from the past 9 months, and the pricing errors are computed using data in the following 3 months. The average IV-MAE for TL, DNN, and the Heston model are 0.04834, 0.20287, and 0.33636, respectively.

distinctly benefits from amalgamating the economic insights of foundational structural models with the empirical knowledge derived from actual market data. The IV-MAE of transfer learning was reduced by 0.155 and 0.288 compared to deep learning and the Heston model, respectively, resulting in improvements of 76.2% and 85.6%.

It's noteworthy that when observing the implied volatility across various time spans, the pricing errors of all models manifested a pronounced peak during 2008 and 2020, aligning with the onset of the financial crisis and the epidemic. Insights from our model underline the significant ripple effect that both the financial crisis and the epidemic have on the derivatives market, leading to these errors. This observation dovetails with existing literature documenting the repercussions of the financial crisis and the pandemic on capital markets, whereby pricing mechanisms are notably influenced. Throughout our analysis period, the transfer learning pricing network consistently exhibited minimal error, in stark contrast to the deep learning model, which displayed substantial volatility in its error rates. Fundamentally,

the deep learning pricing network 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 and the epidemic, but excel in more stable conditions, explaining the observed error volatility. Conversely, the transfer learning network, given its foundational layers driven by the model, remains relatively insulated from such market perturbations. This nuanced understanding will be further dissected in our subsequent attribution analysis.

A natural question is: does the Inferior Performance of Deep Learning Compared to Transfer Learning Solely Result from the Smaller Dataset Size? To ascertain 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 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 DNN," 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 2, shed light on the impact of expanding DNN 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 1.28×10^{-3}. It is worth noting that the augmentation introduced by expanding DNN 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 0.073. In comparison to the expanding DNN approach, which simply enlarges the dataset with historical data, the warm-start approach provides a strategic advantage. While the expanding DNN 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 DNN approach.

However, this still lags behind the performance seen with transfer learning, where the average error was 0.081 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.

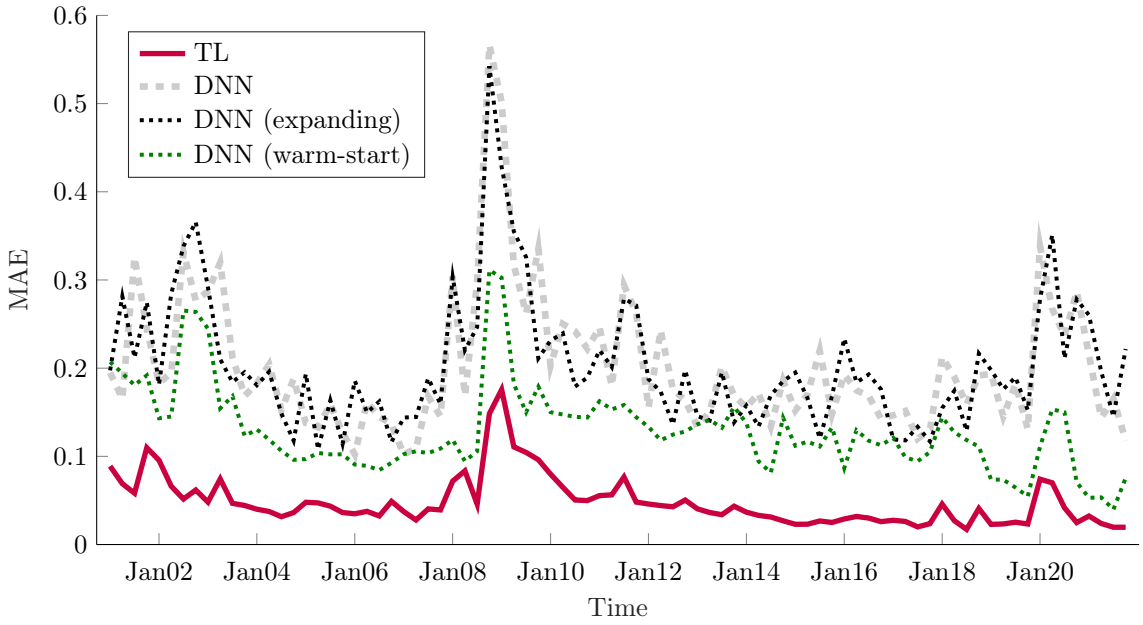


Figure 2: **TL vs. DNN under expanding windows and warm-starting.** This figure compares the out-of-sample mean-absolute pricing errors for TL and DNN under the rolling-window setting against those of two differently-trained DNNs. The expanding-window approach retrains the DNN in each quarter using all 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 DNN in each quarter by initializing its parameters using the values from the model trained in the previous quarter. The average IV-MAE for TL, DNN, DNN(expanding) and DNN(warm-start) are 0.04834, 0.20287, 0.20416, and 0.12982, respectively.

4.3 Feature Importance of Option Pricing

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. Through feature importance analysis, our work contributes to the refinement of option pricing theory and offers insights for its theoretical development.

Initially, we conducted a comprehensive examination of input feature importance within the framework of the Black-Scholes model. As detailed in Table 3, transfer learning suggests that the salient features of the option contract, specifically the strike price (K), the underlying asset's price (S), and the risk-free interest rate (r), hold significant prominence. This assertion

is bolstered by their feature importance values which notably surpass those of other features. For example, the average feature importance for K is approximately 0.255, and its median value hovers around 0.282. Excluding S, K, and r, the importance of other features is notably under 0.1. Volatility, a pivotal variable in option pricing, has profound economic relevance and is often seen as the cornerstone of the Black-Scholes equation. While the feature importance of volatility ($IV1$) lags behind S, K, and T, it remains superior to other features. The expiry time, T, slightly trails volatility in importance, ranking fifth. In this study, we found that the feature importance of dividend rates is relatively low, with a median value of -3.5×10^{-5} and an average value of 1.65×10^{-4}. This result significantly deviates from the expectations set by traditional option pricing theories regarding the impact of dividend rates. Although these numbers may seem statistically insignificant, they actually reveal an important finding: in the current market conditions, the impact of dividend rates on option values may be much lower than predicted by classical theories. This discovery challenges the conventional views on option pricing and suggests a need to revisit and understand the role of dividend rates in determining option values .

Beyond the variables encompassed by the Black-Scholes model, the Put-Call Ratio emerges as a significant feature. This ratio reflects the interplay between the demand for put and call options, traditionally seen as a barometer of market sentiment. A higher Put-Call Ratio is often indicative of a bearish outlook among investors, particularly when they prefer purchasing more put options for hedging. Conversely, a lower ratio might signal bullish sentiments. The feature importance of this ratio boasts a mean value of 3.64×10^{-3} and a median of 1.67×10^{-3}. This empirical evidence places it sixth in the hierarchy of feature significance, suggesting that in certain market scenarios, prevailing sentiments can significantly influence option prices. While the impacts of both long-term and short-term volatilities on option prices seem comparably significant, the latter's influence is slightly more pronounced. Notably, both the Put-Call Ratio and short-term volatility (hv1) exhibit outlier values in feature importance, hinting at potential drastic fluctuations in option prices under various temporal or market conditions. In comparison, although the short-term momentum's feature importance does not surpass its long-term counterpart, it exceeds 10^{-4}, thus outstripping the impact of dividend rates. The momentum effect on option prices remains somewhat perplexing within the existing

no-arbitrage pricing models. Additionally, the long-term momentum of the underlying asset, denoted as ($spmom4$), records mean and median feature importance values of 4.89×10^{-4} and 2.40×10^{-4}, respectively, once again surpassing the influence of dividend rates. This long-term momentum predominantly has a bullish predictive influence on option prices, likely reflecting the market's optimistic outlook on extended trends. The explanatory power of the Put-Call Ratio and momentum on option prices challenges traditional theories and stimulates our exploration of new theories.

5 Why is Transfer Learning 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:

$$\tilde{\epsilon_{it}}^{DL} - \tilde{\epsilon_{it}}^{TL} = x_{it-1}\beta + u_{it} \tag{18}$$

Table 4 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.

5.1 Structural Shocks and Rare Event Issues

"Rare or extreme events" denote occurrences characterized by an exceptionally low probability. Typically, these refer to infrequent incidents that exert a profound influence on a model system, potentially jeopardizing its stability. In the realm of derivatives pricing, the challenge posed by rare events is a longstanding issue that deep learning models often grapple with. However, the transfer learning algorithm highlighted in this paper proves adept at navigating this challenge.

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:

$$d_M(\vec{x}, Q) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}, \tag{19}$$

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 4, we can see that the coefficient of distance is significantly positive, ranging from 3.66×10^{-4} to 8.43×10^{-4}, and the corresponding t-statistic values are all greater than 35. 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.

5.2 The Similarity of DGPs between True Data and the Source Domain Model

Transfer learning, enhanced through pre-training, exhibits varying performance levels in predictive tasks. This variability largely hinges on the chosen model for the pre-trained data. Specifically, the extent of similarity between the data generation processes in the real and source domain models plays a crucial role. In essence, the selection of the model in the source domain, which is a human-driven decision, significantly influences transfer learning outcomes.

Our pre-training approach involves generating random data as a preliminary step. Following this, we employ the Black-Scholes model to derive prices from this random data, serving as the training benchmark for the source domain. However, this methodology reveals a notable decline in the relative advantage of transfer learning over deep learning when real-world data markedly deviates from the Black-Scholes model assumptions. This phenomenon is evident in our attribution analysis.

Referencing Table 4, it's observed that the coefficients related to 'vol20' (the 20-day standard deviation of volatility) are consistently negative, ranging between -0.177 and -0.134. Furthermore, the absolute values of their corresponding t-statistics exceed 60. This trend indicates a decrease in transfer learning performance as the standard deviation of volatility increases. The rationale behind this pattern is that a higher volatility standard deviation suggests real data aligning more closely with a stochastic volatility model rather than the Black-Scholes model. Given the significant disparity between these two models, the relative effectiveness of transfer learning diminishes, constrained by the choice of the pre-trained model.

5.3 Relative Performance and Time to Maturaty

Is the relative performance of transfer learning affected by the expiration time of an option? To explore this question, we classified options into four categories based on their expiration

times and incorporated three time-dichotomous variables: 0-7 days, 7-14 days, and over 90 days, in our attribution analysis. The results presented in Table 4 indicate that the coefficients for these time variables are significantly positive at a 1% level. Specifically, the coefficients for the 0-7 days variable range from 0.0202 to 0.0332, for 7-14 days from 0.00972 to 0.0159, and for over 90 days from 0.0224 to 0.0416. All these coefficients exhibit t-statistics greater than 40, with those for the over 90 days category exceeding 120. This demonstrates the substantial relative performance of transfer learning in predicting both ultra-long-term and ultra-short-term options.

Both ultra-long-term and ultra-short-term options are challenging to predict due to distinct factors. Ultra-long-term options exhibit higher price uncertainty because of their extended time to expiration, resulting in increased volatility and unpredictability. Similarly, ultra-short-term options are difficult to forecast accurately due to their proximity to the exercise date, which leads to different pricing dynamics, as explored in Andersen, Fusari, and Todorov (2017), compared to other options. While transfer learning can effectively lessen the impact of volatility on price predictions by utilizing broad and diverse pricing data acquired during training in the source domain, deep learning is less adept at handling these challenges. This is further evidenced by the relative magnitudes of the coefficients for the 0-7 and 7-14 days variables. As the expiration date draws closer, the volatility of option prices escalates, and deep learning shows a reduced performance compared to transfer learning. The results indicate that the coefficients for the 0-7 days variable are twice as large as those for the 7-14 days variable. This suggests that the relative performance of transfer learning improves as the expiration time shortens.

5.4 Left Tail Volatility

In the realm of financial markets, understanding the nuances of risk and volatility is paramount. Among the various forms of market risk, left tail volatility holds a particular significance, as discussed in Atilgan et al. (2020). This term refers to the heightened volatility associated with extreme negative returns in an asset's price distribution – the "left tail" of the distribution curve. These are not just routine market downturns; rather, they represent severe, often unexpected drops in asset values. In essence, left tail volatility captures the essence of market

fears and worst-case scenarios, where the stakes are high, and the consequences are significant.

This type of volatility is especially critical in times of financial crises or major economic shocks, where traditional models of market behavior, predicated on normal distribution assumptions, fail to accurately predict or explain the extreme outliers. By focusing on these outliers, left tail volatility provides a more realistic and comprehensive view of market risk, especially in terms of understanding and preparing for rare but catastrophic events.

In our attribution analysis, we employed left tail volatility as a critical measure, and the results revealed a noteworthy phenomenon. The coefficient of left tail volatility was found to be significantly positive, ranging from 1.17×10^{-3} to 1.77×10^{-3}, with all of the t-statistic exceeding 35. This intriguing outcome implies that higher left tail volatility correlates with better relative performance of transfer learning compared to traditional deep learning models.

To understand this phenomenon, we need to consider the nature of transfer learning and the challenges posed by left tail events. Transfer learning, by design, is adept at adapting pre-existing models (trained on large, diverse datasets) to specific, often unique scenarios. In the context of left tail volatility, where extreme market downturns provide limited, albeit impactful data, transfer learning demonstrates its strength. It leverages the foundational knowledge gained from broader datasets to make more accurate and robust predictions about these rare events.

In contrast, deep learning models, when developed from scratch, require extensive data to learn and make predictions. The scarcity and the extreme nature of left tail events pose a significant challenge to these models. They struggle to generalize from limited data and often fail to capture the nuanced patterns that are crucial for predicting extreme market movements.

Furthermore, the positive correlation between left tail volatility and the efficacy of transfer learning suggests that as the market experiences higher uncertainty and more pronounced negative swings, the adaptability and pre-trained knowledge inherent in transfer learning become increasingly valuable. This adaptability is less about predicting the exact nature of the market downturn and more about effectively managing the uncertainty and rarity of data in these scenarios.

5.5 Other Factors

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, suggesting that as 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.

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. Concerning moneyness, [4] the results of regression shown in Table 4 are consistent with this intuition.

The positive coefficients for Out-of-the-Money (OTM) and Deep Out-of-the-Money (DOTM) options, which are statistically significant at the 1% level and often exhibit t-statistics above 200, highlight an intriguing aspect of options behavior. OTM and DOTM options represent scenarios where the underlying asset's price is above the strike price, making these options less likely to be exercised. This characteristic imbues them with a high level of uncertainty and nonlinearity in their pricing. In such contexts, deep learning, while proficient at handling nonlinear patterns, is outperformed by transfer learning. The advantage of transfer learning lies in its ability to leverage insights from nonlinear models trained in different but related domains (source domains). This results in a more nuanced understanding of the complex dynamics at play in the pricing of OTM and DOTM options.

[4] We derive our moneyness calculation from Anderson, 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:

$$m = ln(\frac{K}{H_T})/(\sqrt{T} * IV_{atm,T}) \tag{20}$$

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

On the flip side, In-the-Money (ITM) and Deep In-the-Money (DITM) options exhibit significant negative coefficients, ranging from -0.0235 to -0.0816. The nature of ITM and DITM options is inherently more linear when compared to OTM and DOTM options. This linearity stems from the fact that the strike prices of ITM and DITM options are usually above the price of the underlying asset, making them more likely to be exercised. Pre-training with the Black-Scholes model might imply a relatively high level of model complexity, especially when dealing with options pricing and financial derivatives. This complexity can sometimes lead to overfitting, where the model performs well on pre-training data but its effectiveness diminishes when processing broader or simpler linear data.

In essence, the performance of transfer learning relative to deep learning appears to be influenced by the degree of nonlinearity inherent in the option types. While transfer learning excels in environments characterized by high uncertainty and nonlinearity (as in OTM and DOTM options), its advantages may be less pronounced in more linear and predictable scenarios (as with ITM and DITM options).

6 Extensions

6.1 Alternative Ways: Pooled Data and Mixed Targets

An inherent question arises as to whether the enhanced performance of transfer learning is merely a result of the incorporation of more data. A related query is the potential effect of integrating information from theoretical models into neural networks using other, more straightforward methods. 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 Figure3, on average, the Pooled Data improves the IV-MAE error by only 0.019061, which is significantly less than the level achieved

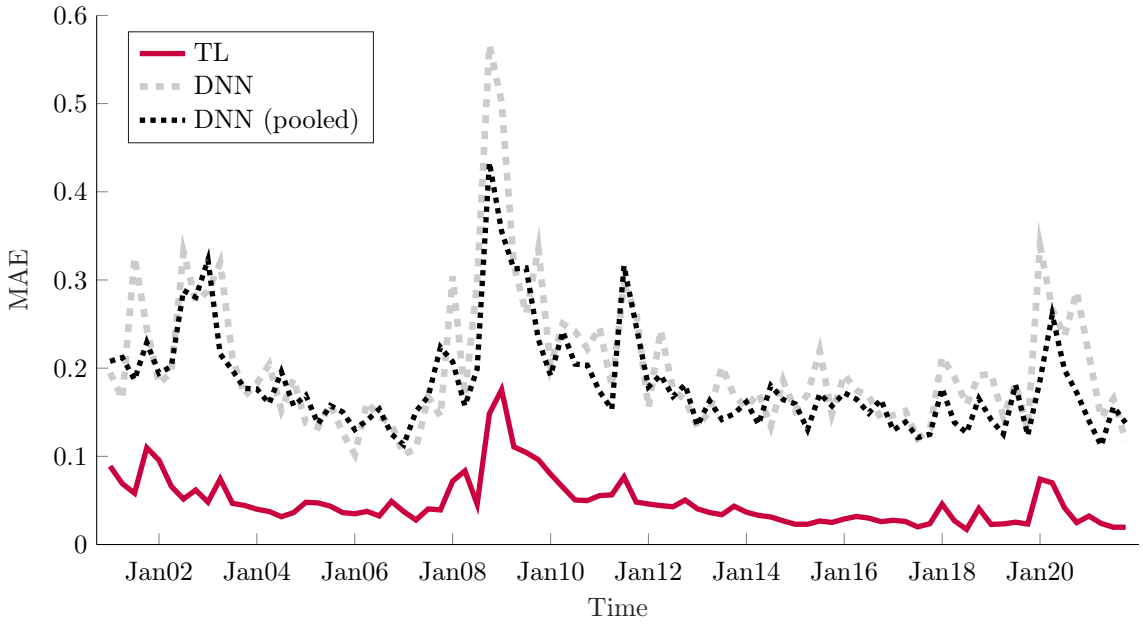


Figure 3: **TL vs. DNN with pooled data.** In this figure, we compare the out-of-sample mean-absolute pricing errors for TL against those of a DNN that is trained by pooling the synthetic data with real data in each training window. The average IV-MAE for TL, DNN, and DNN(pooled) are 0.04834, 0.20287, and 0.18381, respectively.

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 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

$$Loss_{total} = \lambda_{BS} loss_{BS} + \lambda_{real} loss_{real}. \tag{21}$$

To ensure comparability, within each training set, the weights were set according to the ratio

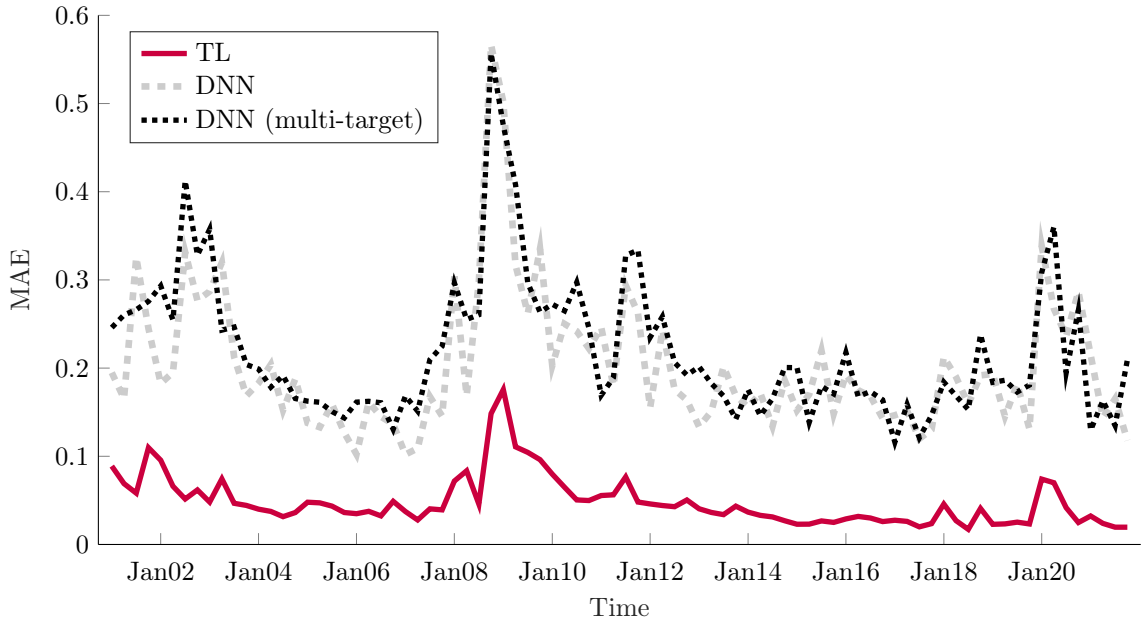


Figure 4: **TL vs. DNN under Black-Scholes constraints.** In this figure, we compare the out-of-sample mean-absolute pricing errors for TL against those of a DNN that directly includes the Black-Scholes model restrictions as part of its loss function. The average IV-MAE for TL, DNN, and DNN(multi-target) are 0.04834, 0.20287, and 0.22003, respectively.

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 4), similar to the Pooled Data, suffer from the biased target issue, and its performance is significantly inferior to transfer learning, worsened the IV-MAE error by 0.017155 on average compared with deep learning.

6.2 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. **?** has already employed 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 10, 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.

6.3 Impact of Learning Rate on Target Domain

The learning rate is a crucial hyperparameter in training machine learning models, significantly 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 11. The figure clearly demonstrates a pronounced U-shaped curve, indicating that both excessively small and excessively large learning rates are detrimental to model performance.

When the learning rate is too small, the model fails to update its parameters effectively. This situation leads to suboptimal performance as the model struggles to reach the optimal solution within a limited number of training iterations. Essentially, the model makes only marginal adjustments to the weights, which are insufficient for achieving convergence in a reasonable timeframe. As a result, the model remains stuck in a state far from the optimum, unable to learn meaningful patterns from the data.

Conversely, when the learning rate is too large, the model updates its parameters too aggressively. This behavior can cause the model to overshoot the optimal solution, leading to oscillations or even divergence. Moreover, a large learning rate can render the training information from the source domain ineffective. The model may disregard the valuable knowledge transferred from the source domain, resulting in poor generalization to the target domain.

In our experiments, the intermediate range of learning rates yielded the best performance, striking a balance between convergence speed and stability. This optimal learning rate allows the model to make significant progress towards the optimal solution while maintaining the beneficial influence of the source domain training information. This indicates that training on the source domain plays a significant role in enhancing the model's performance. However, fine-tuning on the target domain is essential, as it allows the model to adjust from the strict assumptions of theoretical models and better adapt to real-world scenarios. Selecting an appropriate learning rate is thus essential for maximizing the efficacy of the training process and ensuring robust performance in the target domain.

6.4 Features v.s. Theory

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 (DNN) models, as presented in Figure 9. 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 0.0045210 on average. In the DNN model, the incorporation of more features even resulted in poorer prediction outcomes, with an average increase of 0.036158 in IV-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. 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.

6.5 Weak Information

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 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 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 scheme for 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 training set for the source domain. The neural network then learns from this random dataset under no-arbitrage conditions as the source domain, followed

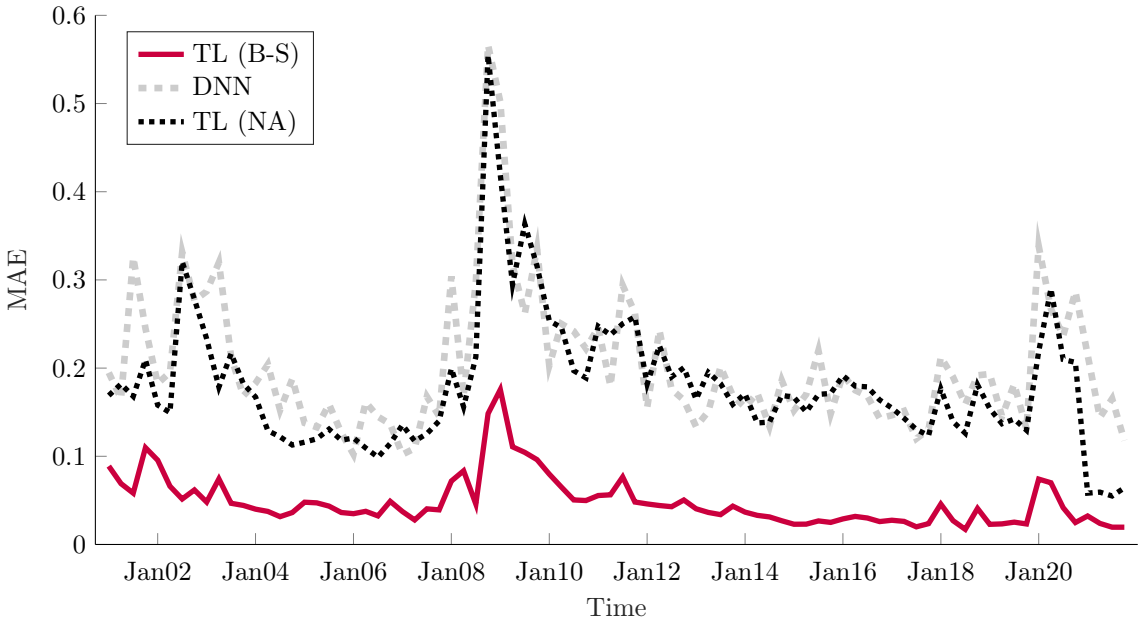


Figure 5: **TL with strong vs. weak structural restrictions.** In this figure, we compare the out-of-sample mean-absolute pricing errors for TL that uses the Black-Scholes model in the source domain against another TL that imposes only the model-free no-arbitrage restrictions in the source domain. Specifically, 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}). The average IV-MAE for TL(B-S), DNN, and TL(NA) are 0.04834, 0.20287, and 0.18101, respectively.

by learning from real data. Our results(Figure 5) differ from those of Cao, Liu, and Zhai (2021), indicating that the no-arbitrage conditions offer no significant improvement to deep learning, only improving 0.021864 when measured by IV-MAE error. This may be because the no-arbitrage conditions are already implicitly contained within the real data; almost every piece of real data adheres to the no-arbitrage conditions, making the inclusion of this information unlikely to enhance deep learning significantly.

Although the integration of non-model information did not yield substantial enhancements in deep learning within the realm of option pricing, the underlying methodology embodies profound applicability and versatility. This approach adeptly facilitates the infusion of intricate insights that are not from model dimensions of economic theories into the fabric of neural networks.

6.6 Robustness Checks

Furthermore, 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 table 7, 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. For all test sets, adjustments in the weight hyper-parameters result in an error deviation of less than 0.04. 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 5.0039^{-3}. With the weights adjusted to 10:2:0, the average deviation between the error curve and the principal result is 5.9807^{-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 5.9807^{-4} 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 also delve into the regression structure for attribution analysis, particularly within the context of different multi-objective learning across varied source domains, referencing Table 5 and 6.

For the 10:4:2 weight configuration, the derived coefficient value for market volatility marginally decreases. Within the quintet of regressions, this coefficient fluctuates between 0.114 and 0.340. Its significance is pronounced at the 0.001 level, with t-values for all six sets surpassing 100. Such a finding intimates that the transfer learning neural network, structured around this specific weighting, might accord a tad less emphasis on implied volatility compared to the principal result. The quintet of 0-7, 0-14 and 90+ coefficients are estimated as 6.42×10^{-3} to 4.31×10^{-2}, all distinctly significant at the 0.001 level. This underscores that, particularly for ultra-short-term and ultra-long-term options, the transfer learning network employing $10 : 4 : 2$ weights demonstrates a marked superiority over conventional deep learning. Notably, the coefficients for both distance and left_tail_vol are still significantly positive, with the absolute values of the t-statistic greater than 90 and 40, respectively. The coefficients of vol20 aligns closely with the core findings, clocking in at -0.135 to -0.0937. In addition, the sign and statistical significance of the coefficients of the moneyness-related variables are also consistent with the main results. For the weight configuration of 10:2:0, the sign and statistical significance of the key variables mentioned above also remain highly consistent with the primary results, although there are some numerical variations. For instance, the absolute value of the coefficient for 'vol20' has increased, while the coefficient for 'left_tail_vol' has

decreased.

We delve deeper into the selection of the model's evaluation metrics. Table 9 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 5.3427 less than that of deep learning. Considering the three outcomes for weighted mean absolute error, transfer learning experience error reductions ranging from 0.281 to 0.315, while the prediction error in deep learning decreases from 3.415 to 3.751. The relative improvements in pricing error for transfer learning vis-à-vis deep learning are recorded as 6.9910^{-2}, 6.6110^{-2}, and 7.3110^{-2}. Proportionally speaking, when compared to deep learning, the transfer learning's pricing error, as measured by average absolute error, witnesses a reduction by 89.98%, 87.40%, 87.19%, and 87.58% 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 99.27%, 98.78%, 98.79%, and 98.78% 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 91.02% 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. Using MAE as an illustration, the enhancement in transfer learning pricing error—when compared to deep learning—for the minimal stability constant at 0.8×10^{-2} and the maximal at 1.2×10^{-2} stands at 87.19% and 87.58% respectively, with a mere discrepancy of approximately 0.39%. 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.

6.7 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 8, ELU-Transfer learning exhibits similarities to the primary result's error curve. The mean deviation between ELU and LeakyReLU's error curves is 7.48×10^{-3}, economically negligible. In the attribution analysis in table 8, critical explanatory variables like MarketIV, distance, left_tail_vol, and the three binary variables associated with time to maturity remain significantly positive at the 0.001 significance level, and vol20 remains notably negative. Specifically, the regression coefficients associated with market-implied volatility are within the range of 0.154 to 0.295 and are statistically significant at a 0.001 level. The distance's coefficient lies between 2.65×10^{-4} and 5.09×10^{-4}, also significant at the 0.001 level, and those coefficients of the three binary variables related to time range from 4.28×10^{-3} to 3.86×10^{-2}. This underscores that, compared to directly employing neural networks, transfer learning exhibits pronounced benefits in contracts characterized by high implied volatility and impacted by structural shocks. Additionally, transfer learning can better deal with the uncertainty and volatility related to time to maturity. The coefficients for left_tail_vol are between 2.70×10^{-3} and 3.11×10^{-3}, and for vol20, they range from -0.19 to -0.153, both statistically significant at the 1% level. This indicates that, the transfer learning framework offers clear advantages over traditional deep learning when the left tail volatility increases, while when real data closely resembles a random volatility model, the performance of transfer learning tends to decline, albeit still significantly better than deep learning.

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 direct connections were omitted. In the absence of all direct channels, the network reverts to the classical multi-layer perceptron design. The outcomes of this configuration are detailed in Table 7 and Figure 8. 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 3.67×10^{-4} 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 employed across various network architectures,

including LSTM, traditional multi-layer perceptrons, GRU, and several others pertinent to option pricing. 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. While GPT has achieved notable success in the realms of natural language processing and image recognition, boasting hundreds of billions of neuronal parameters, it's yet to be determined whether such computation-intensive methodologies can enhance option pricing capabilities within the framework established by this paper.

6.8 Number of Epochs

To further emphasize the robustness of the results, we varied the number of epochs in the training of Deep Learning and Transfer Learning within the target domain. We again selected the same four forecast points for training, with the outcomes presented in Figure 12. The results clearly show that with increasing epochs, once the number exceeded six, there was no significant improvement in the predictive performance of Deep Learning; the trend was predominantly fluctuating. While there was a slow downward trend in the error of Deep Learning predictions at the points of the second quarter of 2014 and the first quarter of 2020, Transfer Learning exhibited the same trend, and continuing to increase the epochs did not narrow the gap between Deep Learning and Transfer Learning.

A more refined approach to substantiate this perspective is to train deep learning models using early stopping. This method halts training when two consecutive observations show an increase rather than a decrease in training errors. Figure Figure 13 presents a distribution chart comparing the pricing errors of deep learning with early stopping, traditional deep learning, and transfer learning. We conducted tests at the fifth point (2002Q1) and the sixtieth point (2015Q4), with sample sizes of 45,731 and 698,300, respectively, covering both small and large training sets. The results demonstrate that incorporating early stopping indeed improves the performance of deep learning to some extent. However, when examining the distribution of errors, the pricing errors from transfer learning are significantly smaller than those from deep learning with early stopping for both types of samples. From this, we may conclude that our findings are robust; in the problem of option pricing, Transfer

Learning is indeed significantly superior to Deep Learning.

6.9 Final Remarks

First and foremost, while we've showcased the BS model as a representative of the source domain, the transfer learning pricing technique presented here can be broadly extended to SDE models. Even in cases where the SDE model lacks an analytical solution, one can still derive its numerical solution through computational techniques and subsequently utilize it as a source domain model for training, adhering to the rationale delineated in this paper. An intriguing avenue for future research lies in examining the impact of the choice of the source domain's stochastic differential equation model, its parameter estimation, and the adopted numerical simulation method on transfer learning's pricing and hedging efficacy. The academic literature is replete with a plethora of stochastic differential equation models tailored for option pricing, complemented by an array of numerical methodologies like Markov Chain Monte Carlo and Quasi-Monte Carlo. Scrutinizing the influence of these models and computational methods on the ultimate option pricing model's pricing and hedging capabilities warrants further exploration.

The second noteworthy aspect is the overarching applicability of this paper's framework to derivatives in a comprehensive sense. Its utility isn't solely confined to European stock index options as discussed herein; it possesses the versatility to be seamlessly extended to other derivative arenas, encompassing American options, futures, exotic options, and digital currency derivatives. To be more precise, the model can be tailored to assimilate a stochastic differential equation model associated with American options, futures, exotic options, or digital currency derivatives in the source domain while acquainting itself with the pertinent market transaction data in the target domain. Given the fragmented nature of data in the derivatives market, the refinement and expansion of this approach offer promising avenues for subsequent scholars.

Thirdly, the framework delineated in this article, with judicious modifications, can be adeptly tailored for application in other financial markets. For instance, this adaptation could entail the utilization of a stock market factor model within the source domain, supplanting the stochastic differential equation model. In parallel, the target domain ought to be meticulously

aligned with real market transaction data. Implementing such a strategy could efficaciously tackle the challenges engendered by the low signal-to-noise ratio, a characteristic often encountered in the modeling of various financial markets. This refined approach not only broadens the framework's applicability but also enhances its potential to provide nuanced insights into complex market dynamics.

7 Conclusion

This article introduces a class of transfer learning-based models that incorporate economic model information into the neural network framework, and applies this model to the field of option pricing. This framework exhibits strong flexibility. Compared to stochastic volatility models and direct use of deep learning, this transfer learning framework yields lower pricing and hedging errors. It also empirically outperforms derivatives pricing models based on stochastic differential equations. In terms of implied volatility - median absolute error (IV-MAE) levels, the transfer learning model significantly enhances performance compared to deep learning and stochastic volatility models. This improvement has substantial economic significance.

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.

References

Amilon, H. 2003. A neural network versus black–scholes: a comparison of pricing and hedging performances. *Journal of Forecasting* 22:317–35.

Amornwattana, S., D. Enke, and C. H. Dagli. 2007. A hybrid option pricing model using a neural network for estimating volatility. *International Journal of General Systems* 36:558–73.

Anders, U., O. Korn, and C. Schmitt. 1998. Improving the pricing of options: A neural network approach. *Journal of forecasting* 17:369–88.

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

Anderson, T. G., N. Fusari, and V. Todorov. 2017. Short-term market risks implied by weekly options. *Journal of Finance* 72:1335–85.

Andreou, P. C., C. Charalambous, and S. H. Martzoukos. 2006. Robust artificial neural networks for pricing of european options. *Computational Economics* 27:329–51.

———. 2008. Pricing and trading european options by combining artificial neural networks and parametric models with implied parameters. *European Journal of Operational Research* 185:1415–33.

Atilgan, Y., T. G. Bali, K. O. Demirtas, and A. D. Gunaydin. 2020. Left-tail momentum: Underreaction to bad news, costly arbitrage and equity returns. *Journal of Financial Economics* 135:725–53.

Bennell, J., and C. Sutcliffe. 2004. Black–scholes versus artificial neural networks in pricing ftse 100 options. *Intelligent Systems in Accounting, Finance & Management: International Journal* 12:243–60.

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

Bollerslev, T., V. Todorov, and L. Xu. 2015. Tail risk premia and return predictability. *Journal of Financial Economics* 118:113–34.

Buehler, H., L. Gonon, J. Teichmann, and B. Wood. 2019. Deep hedging. *Quantitative Finance* 19:1271–91.

Cao, J., J. Chen, and J. Hull. 2020. A neural network approach to understanding implied volatility movements. *Quantitative Finance* 20:1405–13.

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, F., and C. Sutcliffe. 2012. Pricing and hedging short sterling options using neural networks. *Intelligent Systems in Accounting, Finance and Management* 19:128–49.

Chen, H., A. Didisheim, and S. Scheidegger. 2022. Deep surrogates for finance: With an application to option pricing. Working paper.

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. *The Journal of Finance* 74:449–92.

Culkin, R., and S. R. Das. 2017. Machine learning in finance: the case of deep learning for option pricing. *Journal of Investment Management* 15:92–100.

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.

Dimitroff, G., D. Roeder, and C. P. Fries. 2018. Volatility model calibration with convolutional neural networks. *Available at SSRN 3252432* .

Doan, J. O., R. Litterman, and C. A. Sims. 1984. Forecasting and conditional projection using realistic prior distributions. *Econometric Review* 3:1–100.

Freyberger, J., A. Neuhierl, and M. Weber. 2020. Dissecting characteristics nonparametrically. *The 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. *The Review of Financial Studies* 33:2223–73.

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. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–8.

Henry-Labordere, P. 2017. Deep primal-dual algorithm for bsdes: Applications of machine learning to cva and im. *Available at SSRN 3071506* .

Heston, S. L. 1993. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The 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. *The 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.

Jacquier, A. J., and M. Oumgari. 2019. Deep ppdes for rough local stochastic volatility. *Available at SSRN 3400035* .

Jang, H., and J. Lee. 2019. Machine learning versus econometric jump models in predictability and domain adaptability of index options. *Physica A: Statistical Mechanics and its Applications* 513:74–86.

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

Kong, L., J. Sun, and C. Zhang. 2020. Sde-net: Equipping deep neural networks with uncertainty estimates. *arXiv preprint arXiv:2008.10546* .

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.

8 Tables and Figures

Table 1: **Descriptive statistics**

We considered all SP500/SPX European call option samples on CRSP from 2001 to 2021. 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				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	97192	491564	931335	<-0.5	1.6749	0.8503	0.4757	<-0.5	0.5614	0.3721	0.1909
[-0.5,-0.25]	233509	949378	875477	[-0.5,-0.25]	0.9912	0.4465	0.3170	[-0.5,-0.25]	0.4639	0.1091	0.0659
[-0.25,0]	1096668	3236739	1677220	[-0.25,0]	0.3819	0.2464	0.2307	[-0.25,0]	0.2285	0.0916	0.0611
[0,0.25]	384935	1471336	1160310	[0,0.25]	0.2986	0.1641	0.1581	[0,0.25]	0.2770	0.0968	0.0564
[0.25,0.5]	17997	61036	139070	[0.25,0.5]	1.2350	0.5254	0.2314	[0.25,0.5]	0.5436	0.1775	0.0923
>0.5	507	4696	28099	>0.5	1.8918	0.7783	0.3448	>0.5	0.4540	0.2397	0.1204
Total	1830808	6214749	4811511	mean	1.0789	0.5018	0.2930	mean	0.4214	0.1811	0.0978

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
market IV	Daily	market implied volatility
BASpread	Contract-daily	Bid-ask spread
Distance	Contract-daily	The Mahalanobis distance between train set and test set
vol20	Contract-daily	Standard deviation of implied volatility in 20 days
left_tail_vol	Contract-daily	Left tail volatility
mIV_change	daily	Change of market implied volatility
ITM	Contract-daily	In-The-Money
OTM	Contract-daily	Out-The-Money
DOTM	Contract-daily	Deep-Out-The-Money
DITM	Contract-daily	Deep-Out-The-Money
OTM%	Contract-daily	The percentage of OTM
DOTM%	Contract-daily	The percentage of DOTM
ITM%	Contract-daily	The percentage of ITM
DITM%	Contract-daily	The percentage of DITM
IV_T	Contract-daily	interactive term
IV_ITM	Contract-daily	interactive term
IV_OTM	Contract-daily	interactive term
IV_DITM	Contract-daily	interactive term
IV_DOTM	Contract-daily	interactive term
IV_T_OTM	Contract-daily	interactive term
IV_T_ITM	Contract-daily	interactive term
IV_T_DOTM	Contract-daily	interactive term
IV_T_DITM	Contract-daily	interactive term

Table 3: **Statistics of Feature Importance**

The table shows the descriptive statistics of the transfer learning feature importance results. The table reports the mean, median, standard deviation, upper 25% quantile, and lower 25% quantile of feature importance on different training sets for each feature. The evaluation method of feature importance is the improvement value of the loss function after removing the feature on the training set. The loss function is the weighted average absolute loss.

Feature	**Mean**	**Median**	**Std**	$Q_{25\%}$	$Q_{75\%}$
S	0.080802	0.065811	0.053689	0.046090	0.102313
K	0.254765	0.281755	0.125342	0.152587	0.338298
T	0.008463	0.008559	0.006065	0.005045	0.011809
r	0.163964	0.189252	0.067358	0.108908	0.221215
d	0.000165	-0.000035	0.001459	-0.000547	0.000336
IV1	0.015858	0.010775	0.015948	0.004021	0.022601
mom1	0.000232	0.000170	0.000491	-0.000098	0.000386
mom4	0.001008	0.000723	0.001664	-0.000079	0.001711
hv1	0.002691	0.000421	0.007501	-0.000077	0.001748
hv9	0.000717	0.000350	0.002214	-0.000284	0.001217
volume1	0.000003	0.000020	0.000195	-0.000064	0.000073
volume5	0.000080	0.000124	0.000267	-0.000005	0.000233
Put-Call Ratio	0.003640	0.001666	0.007027	0.000419	0.003318
Earnings-Price Ratio	-0.000080	0.000025	0.000776	-0.000248	0.000297
spmom1	0.000114	0.000064	0.000399	-0.000108	0.000275
spmom4	0.000489	0.000240	0.001012	-0.000001	0.000539

Table 4: **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 variable X_Z represents the interaction term of the variables X and Z. The superscripts ***, **, and * indicate statistical significance at the 1%, 5%, and 10% levels, respectively.

	(1)	(2)	(3)	(4)	(5)	(6)
T	0.0222***	0.116***	0.123***			
	(112.86)	(191.94)	(169.05)			
0-7				0.0332***	0.0222***	0.0202***
				(121.22)	(80.22)	(71.96)
7-14				0.0159***	0.0101***	0.00972***
				(72.82)	(45.52)	(43.91)
90+				0.0224***	0.0416***	0.0367***
				(121.44)	(145.59)	(127.98)
marketIV	0.149***	0.345***	0.368***	0.154***	0.257***	0.243***
	(150.35)	(252.29)	(254.94)	(155.14)	(191.32)	(182.81)
BAspread	-0.00056***	-0.00057***	-0.00039***	-0.000154***	0.000182***	0.000178***
	(-19.73)	(-18.93)	(-14.19)	(-6.58)	(7.51)	(7.20)
distance	0.000843***	0.000435***	0.000366***	0.000746***	0.000525***	0.000523***
	(82.72)	(42.13)	(35.97)	(74.49)	(50.79)	(50.78)
vol20	-0.139***	-0.137***	-0.134***	-0.164***	-0.177***	-0.162***
	(-65.73)	(-65.12)	(-63.96)	(-78.01)	(-82.90)	(-75.25)
left_tail_vol	0.00271***	0.00317***	0.00317***	0.00264***	0.00297***	0.00295***
	(85.61)	(101.13)	(101.14)	(83.49)	(95.15)	(94.38)
OTM	0.0216***	0.13***	0.134***	0.0158***	0.116***	0.117***
	(88.55)	(245.19)	(254.31)	(63.53)	(216.04)	(217.33)
DOTM	0.108***	1.01***	0.994***	0.0941***	0.988***	0.946***
	(240.47)	(150.16)	(146.06)	(203.09)	(149.82)	(142.40)
ITM	-0.0583***	-0.0652***	-0.0652***	-0.0648***	-0.0794***	-0.0816***
	(-101.99)	(-77.90)	(-78.19)	(-113.88)	(-94.53)	(-97.67)
DITM	-0.0706***	-0.0235***	-0.0283***	-0.076***	-0.0285***	-0.0433***
	(-83.60)	(-20.91)	(-24.69)	(-90.02)	(-25.06)	(-37.67)
IV_ITM		0.000107	-0.0209***		0.0397***	0.0682***
		(0.04)	(-7.47)		(14.33)	(24.41)
IV_DITM		-0.16***	-0.142***		-0.164***	-0.0454***
		(-49.17)	(-39.06)		(-48.71)	(-12.53)
IV_DOTM		-1.79***	-1.78***		-1.75***	-1.64***
		(-140.38)	(-136.50)		(-139.22)	(-129.01)
IV_OTM		-0.315***	-0.362***		-0.282***	-0.28***
		(-192.23)	(-212.90)		(-168.75)	(-165.54)
IV_T		-0.354***	-0.41***		-0.0971***	-0.064***
		(-162.40)	(-134.70)		(-83.62)	(-46.68)
IV_T_OTM			0.13***			0.00703***
			(72.08)			(5.07)
IV_T_ITM			0.0488***			-0.133***
			(6.34)			(-17.46)
IV_T_DOTM			0.0104***			-0.142***
			(2.59)			(-38.23)
IV_T_DITM			-0.0496***			-0.293***
			(-7.76)			(-48.30)
const	0.0748***	0.0269***	0.0245***	0.0738***	0.054***	0.0555***
	(266.53)	(78.97)	(69.55)	(261.84)	(168.33)	(173.77)
R^2	0.04053	0.05706	0.05815	0.04131	0.04963	0.05139

Table 5: **Attribution Analysis of Transfer Learning with Source Domain Weight 5:2:1**

The table shows the results of the attribution analysis. The response variable is the difference between pricing errors for deep learning and transfer learning. Pricing error is measured by the median absolute error at the level of implied volatility. The variable X_Z represents the interaction term of the variables X and Z. The superscripts ***, **, and * indicate statistical significance at the 1%, 5%, and 10% levels, respectively.

	(1)	(2)	(3)	(4)	(5)	(6)
T	0.0139***	0.119***	0.119***			
	(73.42)	(193.00)	(160.81)			
0-7				0.0342***	0.0207***	0.0168***
				(125.27)	(75.06)	(60.50)
7-14				0.0152***	0.00769***	0.00642***
				(70.23)	(35.29)	(29.42)
90+				0.0163***	0.0431***	0.0374***
				(89.85)	(152.93)	(132.47)
marketIV	0.114***	0.336***	0.34***	0.118***	0.247***	0.221***
	(116.11)	(246.96)	(238.96)	(120.41)	(184.34)	(167.19)
BAspread	-0.000136***	-0.000151***	-7.34e-05***	0.000142***	0.000613***	0.000463***
	(-5.65)	(-6.27)	(-3.11)	(5.88)	(20.24)	(16.24)
distance	0.00142***	0.000956***	0.000935***	0.00133***	0.00105***	0.00109***
	(140.28)	(94.71)	(93.03)	(131.87)	(99.79)	(104.56)
vol20	-0.0954***	-0.096***	-0.0937***	-0.117***	-0.135***	-0.119***
	(-46.95)	(-47.32)	(-46.07)	(-57.15)	(-64.22)	(-56.23)
left_tail_vol	0.00123***	0.00177***	0.00176***	0.00117***	0.00157***	0.00155***
	(39.05)	(56.80)	(56.36)	(37.06)	(50.49)	(49.82)
OTM	0.00103***	0.132***	0.133***	-0.00439***	0.119***	0.118***
	(4.28)	(251.35)	(254.45)	(-17.78)	(221.35)	(219.36)
DOTM	0.0705***	0.99***	0.967***	0.057***	0.977***	0.926***
	(161.79)	(159.80)	(152.26)	(127.17)	(158.25)	(148.42)
ITM	-0.0462***	-0.0524***	-0.053***	-0.052***	-0.0666***	-0.0683***
	(-82.92)	(-64.82)	(-65.77)	(-93.73)	(-81.98)	(-84.56)
DITM	-0.0595***	-0.00969***	-0.0135***	-0.0644***	-0.0145***	-0.0276***
	(-72.30)	(-8.88)	(-12.22)	(-78.18)	(-13.19)	(-24.83)
IV_ITM		-0.00484	-0.00918***		0.0359***	0.0771***
		(-1.82)	(-3.34)		(13.20)	(28.16)
IV_DITM		-0.171***	-0.147***		-0.173***	-0.0543***
		(-52.70)	(-40.89)		(-52.22)	(-15.12)
IV_DOTM		-1.85***	-1.8***		-1.81***	-1.67***
		(-156.45)	(-147.36)		(-153.97)	(-139.54)
IV_OTM		-0.378***	-0.394***		-0.345***	-0.316***
		(-233.35)	(-233.93)		(-208.03)	(-187.47)
IV_T		-0.394***	-0.404***		-0.135***	-0.0765***
		(-176.27)	(-130.53)		(-115.47)	(-55.51)
IV_T_OTM			0.0484***			-0.0713***
			(26.38)			(-50.50)
IV_T_ITM			0.00943			-0.166***
			(1.21)			(-21.78)
IV_T_DOTM			-0.0496***			-0.2***
			(-13.46)			(-59.77)
IV_T_DITM			-0.0593***			-0.291***
			(-8.94)			(-46.24)
const	0.0894***	0.035***	0.035***	0.0871***	0.0628***	0.0648***
	(321.07)	(103.06)	(100.15)	(311.28)	(197.30)	(204.45)
R^2	0.02265	0.04427	0.04454	0.02400	0.03613	0.03807

Table 6: **Attribution Analysis of Transfer Learning with Source Domain weight 10:2:0**

The table shows the results of the attribution analysis. The response variable is the difference between pricing errors for deep learning and transfer learning. Pricing error is measured by the median absolute error at the level of implied volatility. The variable X_Z represents the interaction term of the variables X and Z. The superscripts ***, **, and * indicate statistical significance at the 1%, 5%, and 10% levels, respectively.

	(1)	(2)	(3)	(4)	(5)	(6)
T	0.0167***	0.116***	0.121***			
	(88.74)	(194.25)	(167.20)			
0-7				0.0246***	0.0125***	0.00953***
				(89.66)	(45.08)	(34.02)
7-14				0.0128***	0.00616***	0.00531***
				(59.31)	(28.20)	(24.27)
90+				0.0168***	0.0409***	0.0359***
				(91.25)	(144.69)	(126.50)
marketIV	0.179***	0.382***	0.397***	0.182***	0.293***	0.273***
	(181.03)	(281.72)	(276.87)	(184.44)	(220.06)	(206.62)
BAspread	-1.47e-06	-1.07e-05	0.00011***	0.000308***	0.000736***	0.000649***
	(-0.06)	(-0.45)	(4.64)	(11.76)	(22.12)	(19.95)
distance	0.000858***	0.000435***	0.000389***	0.000785***	0.000544***	0.000565***
	(85.09)	(43.34)	(38.78)	(77.44)	(50.94)	(53.22)
vol20	-0.168***	-0.166***	-0.164***	-0.187***	-0.202***	-0.187***
	(-81.85)	(-81.35)	(-80.06)	(-90.24)	(-94.66)	(-87.09)
left_tail_vol	0.00173***	0.00219***	0.00219***	0.00168***	0.00201***	0.00199***
	(55.47)	(71.17)	(71.20)	(53.73)	(65.32)	(64.79)
OTM	0.00609***	0.113***	0.116***	0.0017***	0.102***	0.102***
	(25.17)	(216.89)	(222.65)	(6.84)	(191.03)	(190.35)
DOTM	0.107***	0.984***	0.976***	0.0964***	0.985***	0.944***
	(242.33)	(163.30)	(156.79)	(211.25)	(163.27)	(153.20)
ITM	-0.0502***	-0.0548***	-0.0548***	-0.0552***	-0.0684***	-0.0702***
	(-89.40)	(-66.59)	(-66.76)	(-98.41)	(-82.55)	(-85.24)
DITM	-0.0615***	-0.0161***	-0.0191***	-0.0656***	-0.0204***	-0.0331***
	(-74.43)	(-14.63)	(-17.04)	(-79.34)	(-18.34)	(-29.52)
IV_ITM		-0.00929***	-0.0225***		0.0325***	0.0692***
		(-3.42)	(-8.00)		(11.75)	(24.64)
IV_DITM		-0.155***	-0.144***		-0.157***	-0.0467***
		(-47.64)	(-39.80)		(-46.97)	(-12.89)
IV_DOTM		-1.76***	-1.75***		-1.74***	-1.63***
		(-153.05)	(-146.19)		(-151.75)	(-137.96)
IV_OTM		-0.313***	-0.345***		-0.282***	-0.265***
		(-194.31)	(-204.88)		(-171.50)	(-158.20)
IV_T		-0.375***	-0.414***		-0.122***	-0.0763***
		(-172.51)	(-135.92)		(-104.16)	(-54.84)
IV_T_OTM			0.0889***			-0.0371***
			(49.19)			(-26.25)
IV_T_ITM			0.0262***			-0.16***
			(3.43)			(-21.33)
IV_T_DOTM			0.00558			-0.159***
			(1.50)			(-46.50)
IV_T_DITM			-0.0289***			-0.271***
			(-4.57)			(-45.05)
const	0.0768***	0.0272***	0.0255***	0.076***	0.0552***	0.0569***
	(277.10)	(80.72)	(73.13)	(273.02)	(174.89)	(180.91)
R^2	0.04127	0.05912	0.05962	0.04171	0.05100	0.05256

Table 7: **Attribution Analysis of Transfer Learning Non-residual Learning Neural Network**

The table shows the results of the attribution analysis. The response variable is the difference between pricing errors for deep learning and transfer learning. Pricing error is measured by the median absolute error at the level of implied volatility. The variable X_Z represents the interaction term of the variables X and Z. The superscripts ***, **, and * indicate statistical significance at the 1%, 5%, and 10% levels, respectively.

	(1)	(2)	(3)	(4)	(5)	(6)
T	0.00553***	0.144***	0.128***			
	(21.44)	(225.82)	(180.10)			
0-7				0.0689***	0.0531***	0.0433***
				(270.60)	(210.01)	(170.58)
7-14				0.0232***	0.0146***	0.00969***
				(123.49)	(77.62)	(51.76)
90+				0.0117***	0.0505***	0.0439***
				(54.10)	(204.86)	(188.35)
marketIV	0.349***	0.427***	0.375***	0.356***	0.316***	0.25***
	(400.53)	(366.47)	(308.69)	(409.70)	(266.77)	(213.15)
BAspread	0.0016***	0.00154***	0.00112***	0.00187***	0.0025***	0.00173***
	(27.92)	(31.07)	(28.21)	(30.38)	(31.91)	(29.16)
distance	0.00104***	0.000993***	0.00116***	0.000887***	0.00106***	0.00129***
	(99.74)	(100.84)	(125.55)	(82.01)	(84.68)	(116.62)
vol20	-0.264***	-0.261***	-0.269***	-0.298***	-0.318***	-0.305***
	(-126.03)	(-127.65)	(-138.05)	(-138.84)	(-136.21)	(-142.93)
left_tail_vol	-0.000548***	7.07e-05***	1.55e-05	-0.000664***	-0.000238***	-0.000257***
	(-21.21)	(2.81)	(0.62)	(-25.66)	(-9.27)	(-10.18)
OTM	-0.0316***	-0.0359***	-0.0458***	-0.0415***	-0.0584***	-0.0671***
	(-145.63)	(-74.94)	(-96.49)	(-186.83)	(-120.39)	(-139.62)
DOTM	0.284***	1.2***	1.15***	0.259***	1.13***	1.07***
	(783.93)	(258.31)	(229.56)	(685.49)	(242.97)	(209.70)
ITM	-0.0797***	-0.142***	-0.141***	-0.0898***	-0.161***	-0.159***
	(-160.52)	(-199.89)	(-201.39)	(-180.36)	(-225.44)	(-225.00)
DITM	-0.0885***	-0.15***	-0.14***	-0.0969***	-0.158***	-0.156***
	(-122.26)	(-155.86)	(-144.97)	(-133.31)	(-163.25)	(-160.94)
IV_ITM		0.228***	0.248***		0.272***	0.331***
		(98.61)	(105.94)		(118.45)	(141.59)
IV_DITM		0.236***	0.204***		0.228***	0.296***
		(80.04)	(65.89)		(78.95)	(96.26)
IV_DOTM		-1.74***	-1.59***		-1.61***	-1.41***
		(-197.98)	(-166.15)		(-183.19)	(-144.42)
IV_OTM		0.0374***	0.148***		0.0843***	0.233***
		(25.44)	(98.00)		(57.22)	(153.47)
IV_T		-0.518***	-0.387***		-0.187***	-0.0364***
		(-230.26)	(-132.98)		(-163.41)	(-28.65)
IV_T_OTM			-0.305***			-0.423***
			(-180.78)			(-315.78)
IV_T_ITM			0.0667***			-0.102***
			(10.10)			(-15.84)
IV_T_DOTM			-0.219***			-0.346***
			(-69.28)			(-123.30)
IV_T_DITM			0.0887***			-0.149***
			(16.47)			(-29.96)
const	0.00683***	-0.0219***	-0.0161***	0.000912***	0.0108***	0.0139***
	(30.04)	(-76.39)	(-54.76)	(3.99)	(40.68)	(53.39)
R^2	0.195804	0.232078	0.239786	0.203352	0.220106	0.23326

Table 8: **Attribution Analysis of Transfer Learning with ELU activation function**

The table shows the results of the attribution analysis. The response variable is the difference between pricing errors for deep learning and transfer learning. Pricing error is measured by the median absolute error at the level of implied volatility. The variable X_Z represents the interaction term of the variables X and Z. The activation function is replaced with ELU activation function. The superscripts ***, **, and * indicate statistical significance at the 1%, 5%, and 10% levels, respectively.

	(1)	(2)	(3)	(4)	(5)	(6)
T	0.00386***	0.121***	0.136***			
	(12.96)	(178.92)	(165.45)			
0-7				0.0302***	0.0188***	0.016***
				(98.73)	(61.13)	(51.10)
7-14				0.012***	0.00513***	0.00428***
				(52.19)	(22.26)	(18.45)
90+				0.0045***	0.0386***	0.0358***
				(17.97)	(131.44)	(124.91)
marketIV	0.154***	0.259***	0.295***	0.158***	0.166***	0.154***
	(157.39)	(196.43)	(206.34)	(161.48)	(123.43)	(111.17)
BAspread	0.0019***	0.0019***	0.002***	0.00208***	0.00269***	0.00262***
	(28.91)	(31.49)	(31.85)	(31.03)	(32.12)	(31.24)
distance	0.000509***	0.000333***	0.000265***	0.00043***	0.000435***	0.000453***
	(41.51)	(27.97)	(21.87)	(34.54)	(30.72)	(31.55)
vol20	-0.161***	-0.153***	-0.157***	-0.178***	-0.192***	-0.184***
	(-68.93)	(-67.31)	(-68.43)	(-75.87)	(-76.04)	(-72.59)
left_tail_vol	0.00276***	0.0031***	0.00311***	0.0027***	0.00289***	0.00286***
	(84.33)	(95.69)	(95.68)	(82.49)	(88.30)	(87.22)
OTM	0.0904***	0.0893***	0.094***	0.0855***	0.0759***	0.0763***
	(355.83)	(156.03)	(164.51)	(326.67)	(131.34)	(131.25)
DOTM	0.1***	0.506***	0.426***	0.0879***	0.497***	0.384***
	(205.67)	(75.26)	(55.93)	(175.26)	(76.23)	(50.57)
ITM	-0.0609***	-0.0952***	-0.0926***	-0.0659***	-0.11***	-0.11***
	(-106.05)	(-116.01)	(-112.93)	(-115.20)	(-133.05)	(-134.00)
DITM	-0.0735***	-0.089***	-0.0843***	-0.0777***	-0.0938***	-0.101***
	(-89.34)	(-81.93)	(-76.50)	(-94.43)	(-85.66)	(-90.97)
IV_ITM		0.115***	0.0663***		0.158***	0.169***
		(44.34)	(24.31)		(60.24)	(61.97)
IV_DITM		0.0688***	0.0186***		0.0657***	0.129***
		(21.78)	(5.32)		(20.52)	(37.05)
IV_DOTM		-0.799***	-0.651***		-0.767***	-0.508***
		(-62.57)	(-44.44)		(-62.05)	(-34.94)
IV_OTM		0.00761***	-0.0429***		0.0412***	0.0485***
		(4.47)	(-24.09)		(24.06)	(27.16)
IV_T		-0.439***	-0.53***		-0.168***	-0.141***
		(-189.60)	(-159.81)		(-127.87)	(-88.32)
IV_T_OTM			0.127***			-0.0145***
			(62.57)			(-8.46)
IV_T_ITM			0.184***			-0.0256***
			(23.99)			(-3.41)
IV_T_DOTM			-0.145***			-0.325***
			(-33.69)			(-79.66)
IV_T_DITM			0.121***			-0.155***
			(19.37)			(-26.61)
const	0.0398***	0.00969***	0.00449***	0.0378***	0.0397***	0.0409***
	(143.40)	(28.39)	(12.55)	(134.76)	(123.78)	(127.57)
R^2	0.062899	0.078742	0.079965	0.063984	0.070375	0.071341

Table 9: **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	0.01564	2.13896	2.05151
	weighted-mse1	0.00397	0.32653	4.83115
	weighted-mse2	0.00373	0.30835	5.19495
	weighted-mse3	0.00423	0.34773	4.55007
mae	unweighted	0.59468	5.93737	8.17972
	weighted-mae1	0.29801	2.36583	9.79846
	weighted-mae2	0.28014	2.18657	9.95153
	weighted-mae3	0.31320	2.52161	9.67452
mape	unweighted	102.0684283	1136.349854	38169.6716

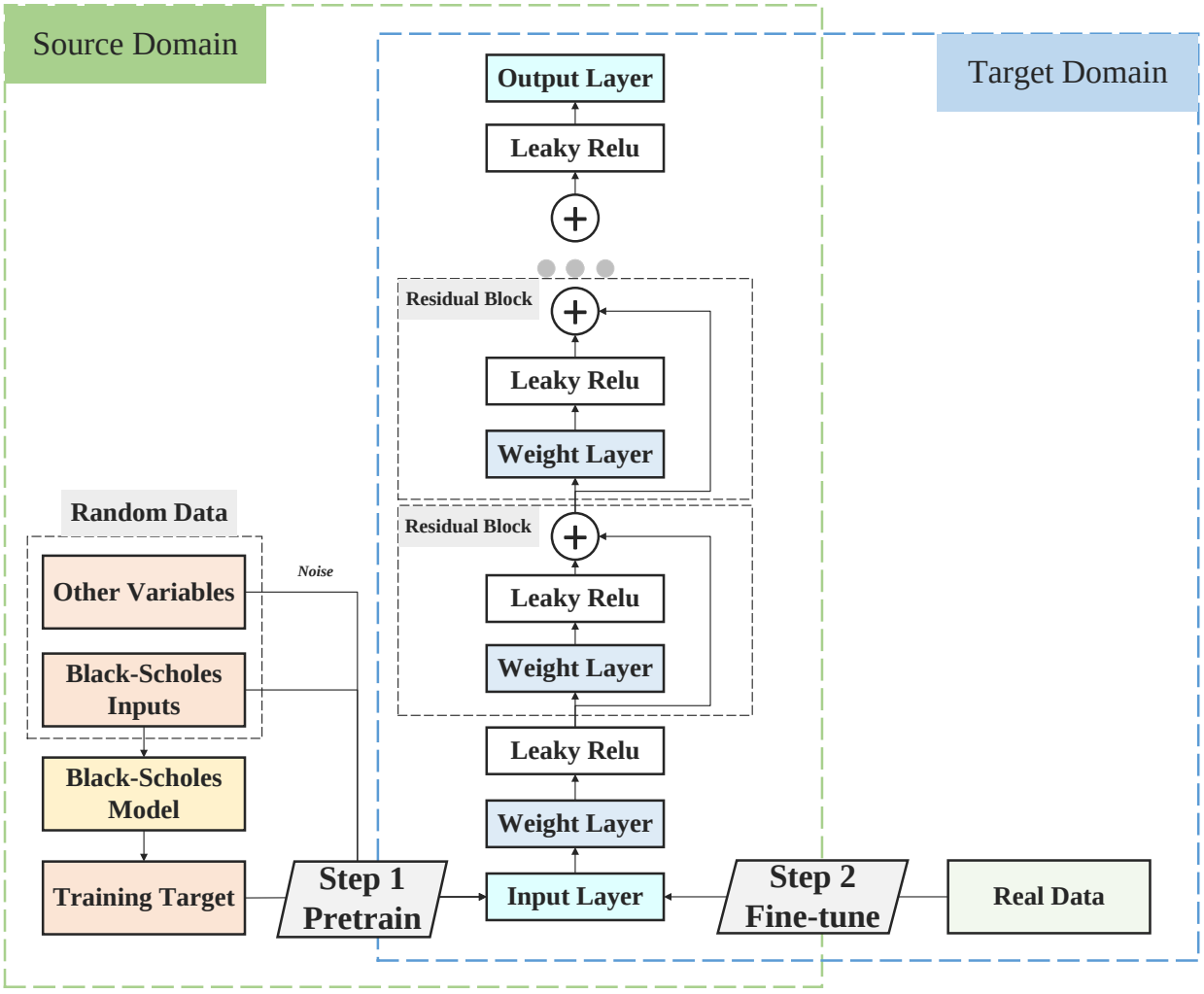


Figure 6: **The Transfer Learning Framework.** This figure illustrates the application of our adopted transfer learning framework in the context of option pricing.

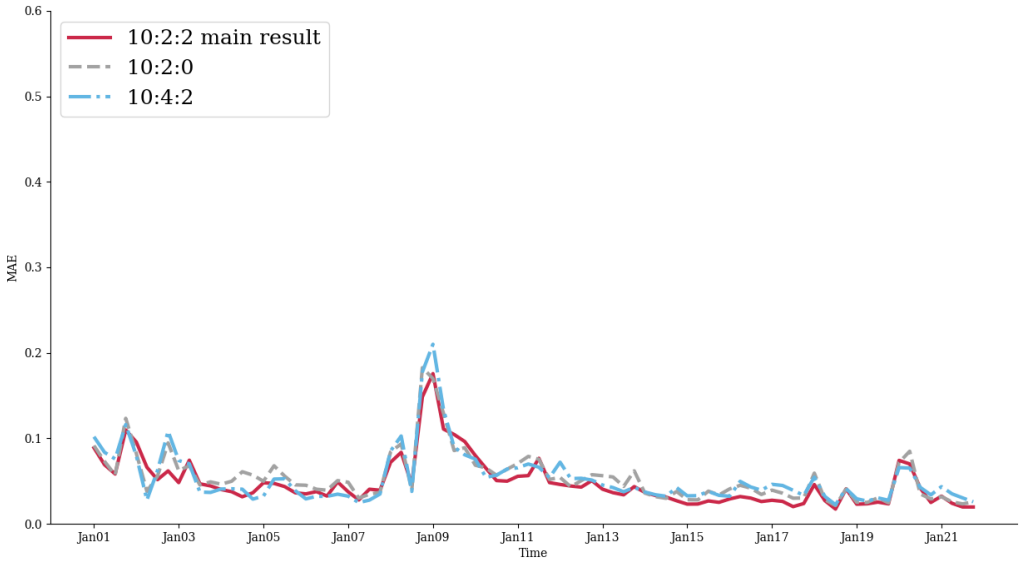


Figure 7: **Robustness Check: Weight of Source Domain Multi-target Training.** We change the Weight of Source Domain Multi-target Training. The main results in the main text report the pricing, delta hedging, and vega hedging weights set to 10:2:2. We set the weights to 5:2:1 and 10:2:0 to observe whether 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. The average IV-MAE for TL(10:2:2), TL(10:2:0), and TL(10:4:2) are 0.04834, 0.05432, and 0.05334, respectively.

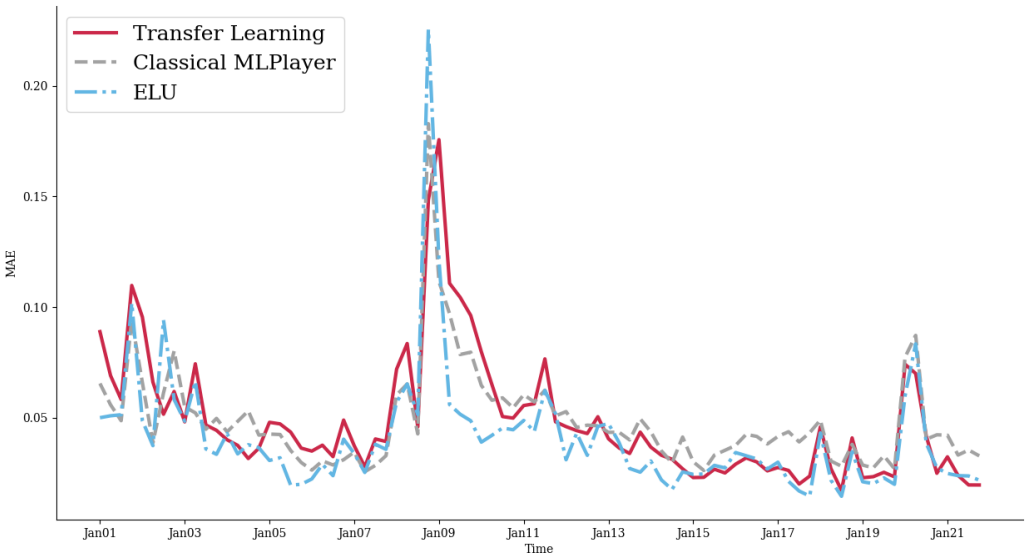


Figure 8: **Robustness Check: NN structure and activation function.** We change the activation function. The frequency of error evaluation is the same as the frequency of model retraining, that is, model retraining and model evaluation every three months. The loss function of the model is chosen as IV-MAE. 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. The average IV-MAE for TL, TL(Classical MLPlayer), and TL(ELU) are 0.04834, 0.04870, and 0.04086, respectively.

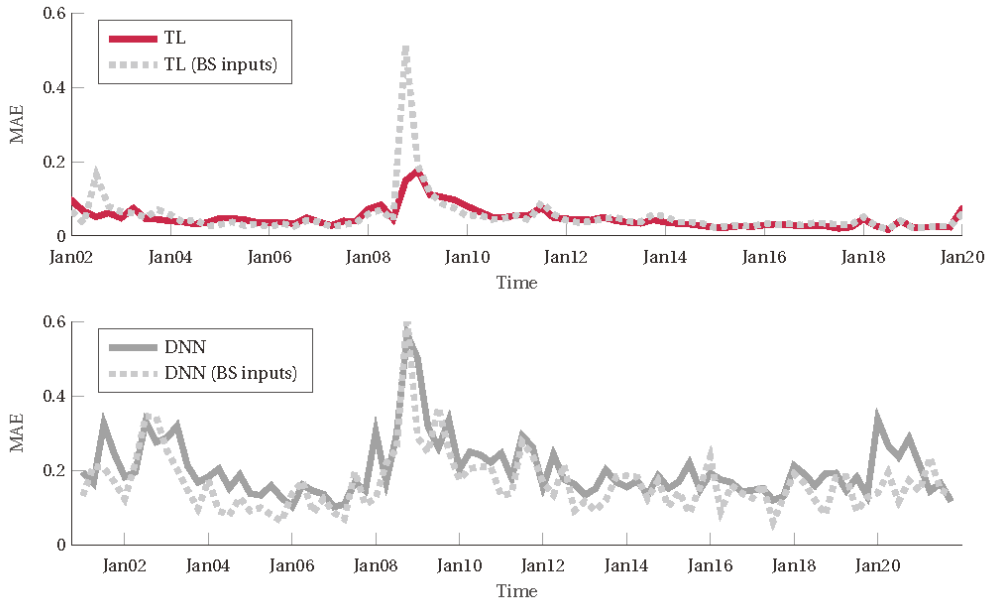


Figure 9: **Using only BS-iputs as features.** TL(BS-inputs) refers to that within the transfer learning framework, we retained solely the inputs utilized in the Black-Scholes model. Similarly, DNN(BS inputs) means that within the deep learning framework, we retained solely the inputs utilized in the Black-Scholes model. The average IV-MAE for TL, TL(BS inputs), DL, and DL(BS inputs) are 0.04834, 0.05286, 0.20287, and 0.16671, respectively.

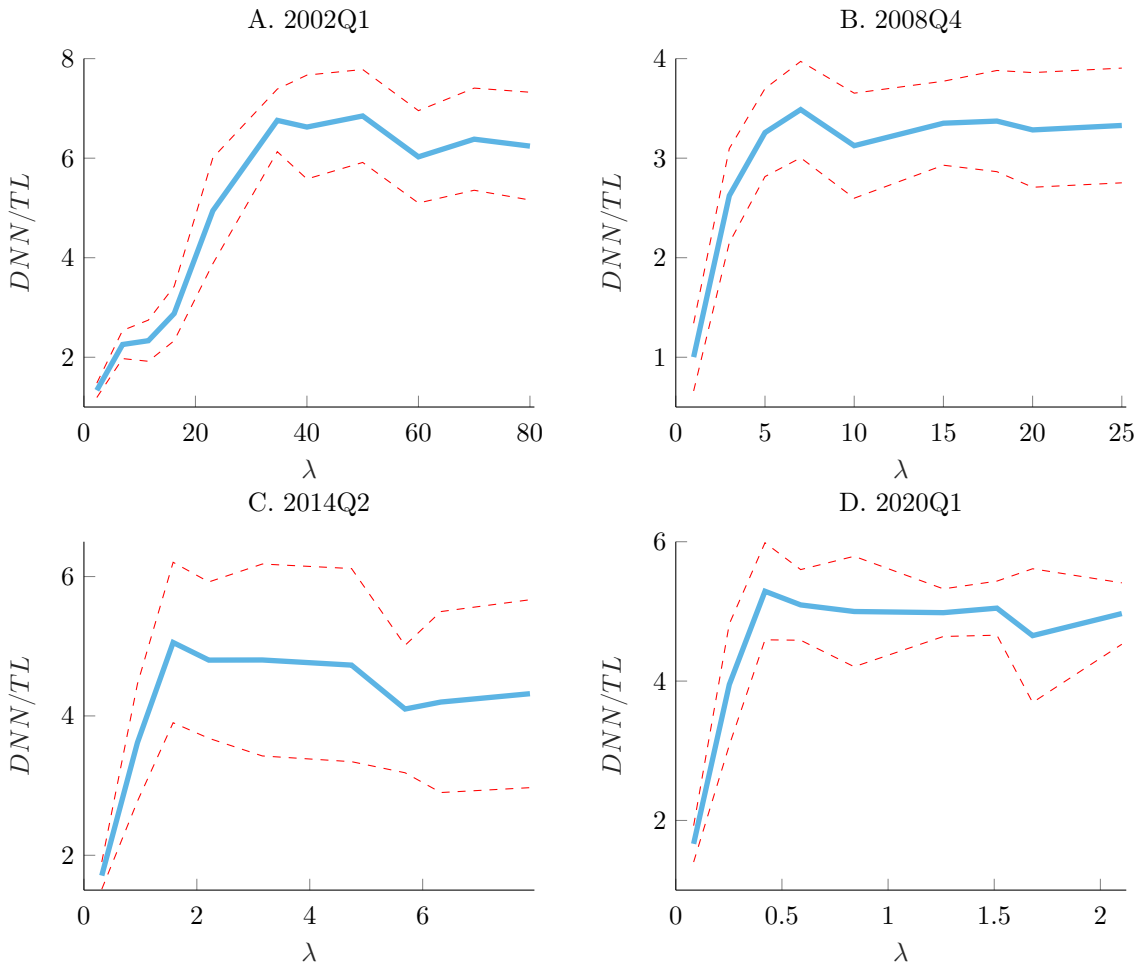


Figure 10: **Changing the Relative Quantity between the source domain and the target domain.** The figure demonstrates the effect of the ratio of training data between the source domain and the target domain on prediction error. In the graph, the y variable is the error of deep learning divided by the 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.

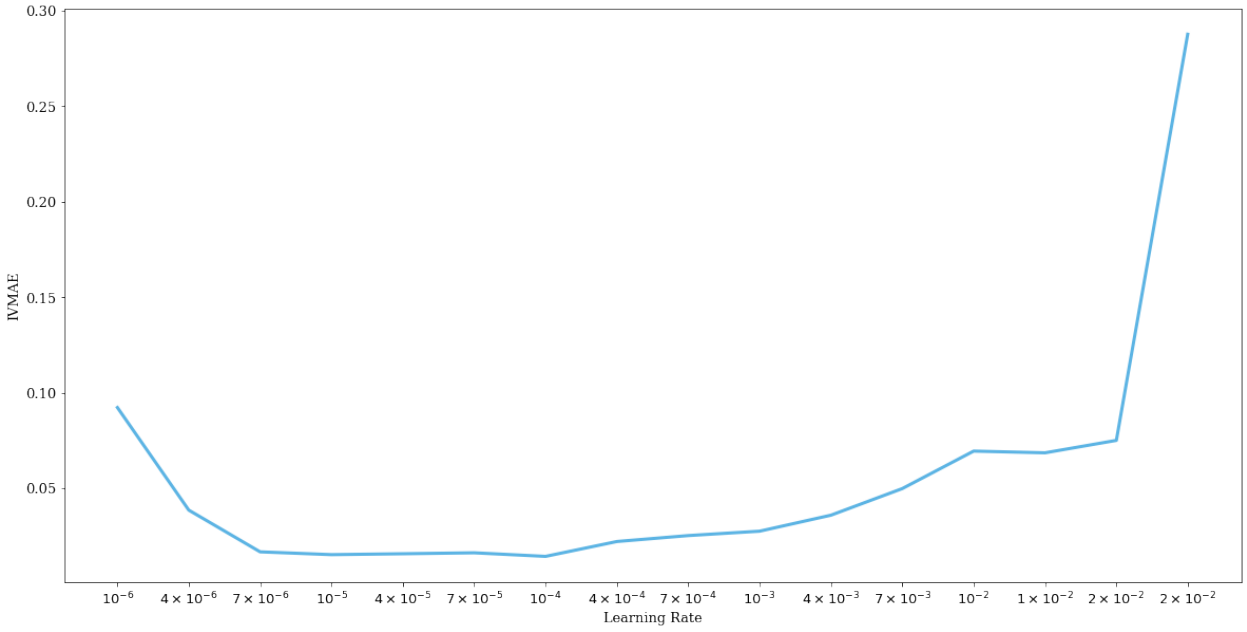


Figure 11: **Changing Learning Rate in Target Domain.** The graph illustrates the impact of different learning rates in the target domain on prediction performance. We select 2014Q2 as example, varying the learning rates in the target domain, to observe changes in the Median Absolute Error of implied volatility for both models.

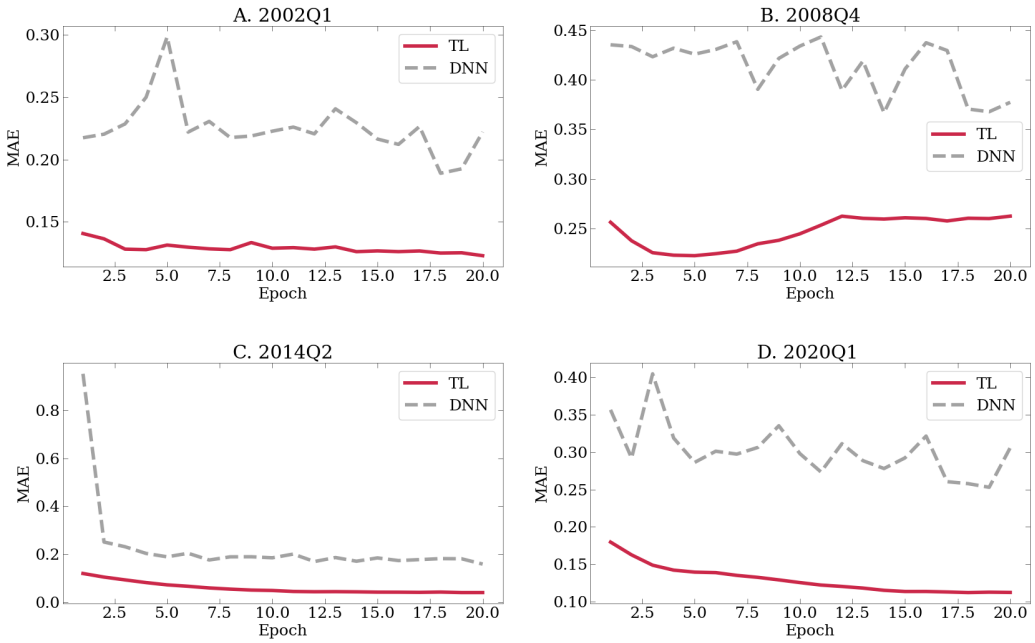
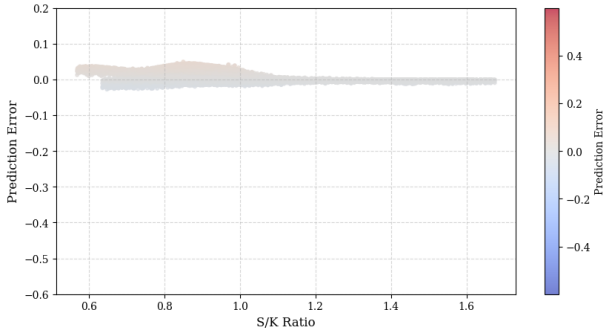
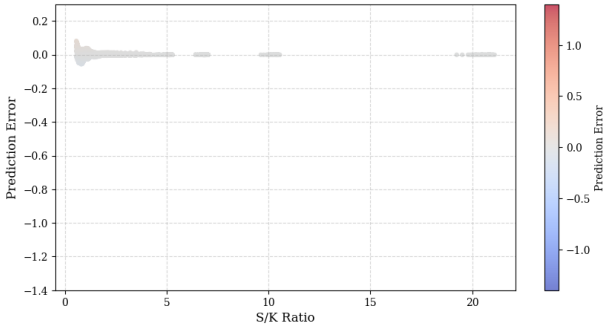


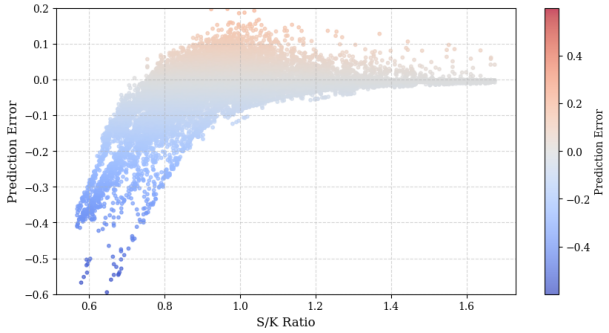
Figure 12: **Changing epochs.** The graph illustrates the impact of different epochs on prediction performance. We select four representative points, varying the number of epochs for deep learning and the number of epochs for transfer learning in the target domain, to observe changes in the Median Absolute Error of implied volatility for both models.



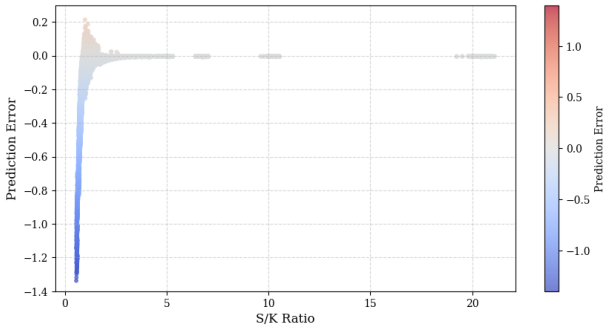
(a) 2002Q1 TL



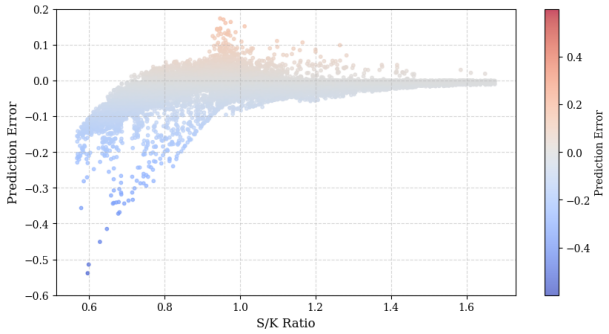
(b) 2015Q4 TL



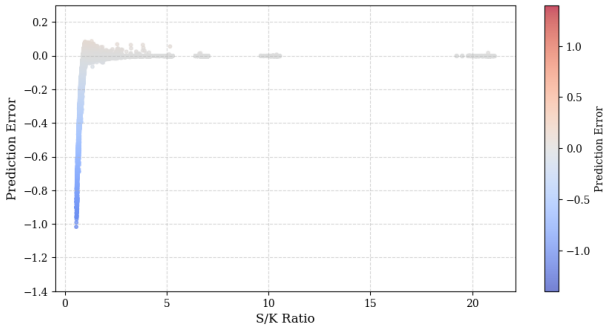
(c) 2002Q1 DL



(d) 2015Q4 DL



(e) 2002Q1 DL with Early Stopping



(f) 2015Q4 DL with Early Stopping

Figure 13: **Comparison of Pricing Errors Using Early Stopping in Deep Learning.** The scatter plot compares the pricing errors of deep learning with early stopping to those of traditional deep learning and transfer learning. The adopted early stopping method stops training when the training error increases consecutively twice during deep learning training. DL refers to training with the same number of epochs as TL on real data. We selected two specific points for testing: the fifth point, corresponding to the test sample of 2002Q1, and the sixtieth point, corresponding to the test sample of 2015Q4. The errors shown in the plot are calculated as the predicted values from the trained network minus the actual prices.

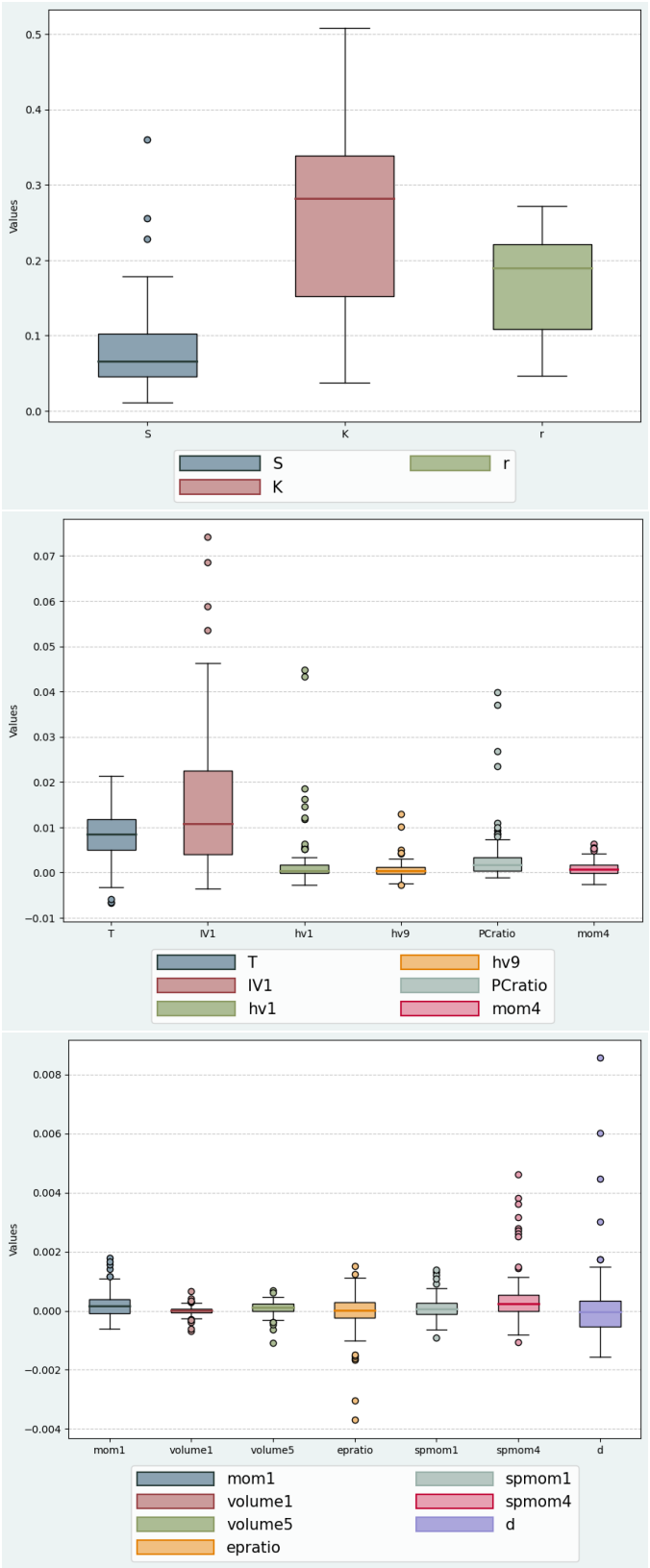


Figure 14: **Feature Importance.** The figure below shows a boxplot of feature importance for a transfer learning model. We have counted the feature importance of the same feature in each training set. This figure aims to visualize the comprehensive features of feature importance on different training 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.

Appendix A. Details of Neural Network Architecture and Training

The employed neural architecture in this article encompasses 17 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 size 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 set encompassed 9 months of data, with a testing set comprising 3 months of data, employing a rolling training approach. The fine-tuning stage consisted of 6 epochs, a learning rate of 0.000006, and a batch size 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.

Deep Learning: We exclusively employed real data and conducted training following the methodology established in the Transfer Learning paradigm for the target domain.

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 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 to 20 to ensure network convergence.

Deep Learning with Mixed-Target: 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 the ratio of the training set's quantity to the sample size of source domain train set (700,000).

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.

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

Appendix B. Description of the Variables used as inputs of Deep Learning and Transfer Learning

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-period 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 period ago to the real price six periods 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 period ago to the real price 21 periods 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 period:

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

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 period:

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

11. **volume1**: A transformed volume metric, measuring the normalized deviation of the current volume from the rolling mean of the past 20 periods, lagged by one period.
12. **volume5**: A 5-period moving average of the normalized volume deviations, lagged by one period.
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 period ago to six periods 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 period ago to 21 periods ago.

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

Appendix 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:

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

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