Deep Learning for Implied Volatility: A Comparative Approach

Jonah Baker
Department of Financial Computing
University College London
London WC1E 6BT
zcicjhz@ucl.ac.uk

Abstract

This paper investigates the use of deep learning techniques for modelling the implied volatility of American-style options, with a focus on Apple Inc. options data from 2016 to 2020. Traditional methods such as CRR binomial trees and parametric models such as SABR and SVI, while widely used, face limitations in flexibility, scalability, and potential arbitrage violations. To address these challenges, we implement and compare several machine learning approaches: linear regression, multilayer perceptrons (MLPs), feedforward neural networks (FNNs), recurrent neural networks (RNNs), and a hybrid FNN enhanced with volatility forecasts from LSTM-based RNNs. We evaluate model performance across several statistical metrics and time series cross-validation. Our results demonstrate that deep learning models, particularly the FNN and the hybrid FNN with realized volatility input, significantly outperform alternative approaches in both accuracy and generalization. The findings underscore the potential of neural networks in capturing complex, non-linear dynamics of the implied volatility surface, offering a robust alternative for options pricing and market risk modelling.

1 Introduction

Implied volatility (IV) represents the market's expectation of the future change in the price of an asset over the lifetime of an option. An implied volatility surface (IVS) is a three-dimensional plot of the implied volatility of various listed options on a single underlying asset normally over a fixed time period. IV is often plotted against moneyness and expiry; here, we use days to expiry (DTE) and log-moneyness, which is the log of the strike divided by the underlying.

In quantitative finance, the IVS is used as a key input for calibrating models, assessing relative value across strikes and maturities, and constructing strategies that exploit mispricing or volatility structure. It plays a key role in pricing, hedging and risk management of derivative instruments. As IV is derived from observed option prices, the IVS encodes information about market sentiment, supply-demand imbalances and forward-looking risk assessments. However, traditional IVS models are limited by rigid assumptions, exhibit instability during calibration, are prone to arbitrage violations, and often fail to generalise effectively, particularly in scenarios where market quotes are unavailable.

For European options, IV is calculated using the Black-Scholes equation [Black and Scholes, 1973], typically employing the Newton-Raphson method as a root-finding algorithm [Orlando and Taglialatela, 2018]. This is relatively simple to implement. In contrast, for path-dependent or exotic options, calculating IV requires more sophisticated numerical techniques. In this paper, we focus on modelling the IV for American options. Typically, one uses interpolation to calculate intermediate volatilities for unquoted strikes and maturities [Cboe Global Markets, 2023] .

To manually calculate the IV of an American option, one typically uses the Cox, Ross & Rubinstein (CRR) method [Cox et al., 1979]. This is an application of the binomial pricing model and gives a static result that can be used to interpolate the surface. While CRR is effective for individual option prices, it becomes very computationally intensive if calculating whole curves. Commonly used fundamental surface models include SABR (Stochastic Alpha Beta Rho) [Hagan et al., 2002] and SVI (Stochastic Volatility Inspired) [Gatheral, 2004]. These models are widely employed for modelling the IVS due to their ability to capture key market features such as skew and smile. However, these surface models may exhibit arbitrage opportunities under certain parameter configurations, compromising their reliability.

Given the computational burden associated with both CRR and interpolation methods, we propose a machine learning-based alternative. We compare the application of linear regression, a Multi-Layer Perceptron (MLP), a Feedforward Neural Network (FNN), and a Recurrent Neural Network (RNN). These models will be discussed in detail in the methods section.

2 Literature Review

Since the seminal article by Black and Scholes [1973] on option pricing, the process has evolved dramatically. The original Black-Scholes Equation (BSE) assumed volatility to be a constant in time. Harvey and Whaley [1992] later proposed a regression model to predict implied volatility.

Then, Dumas et al. [1998] showed that modelling volatility as a deterministic function of the underlying asset price and time - as hypothesised by Derman and Kani [1994], Dupire [1994], and Rubinstein [1994] - was no better than ad hoc smoothing of BSE-implied volatilities across strike price and time to expiration. Dumas et al. [1998] examined the empirical behaviour of the IVS, and showed that it is stochastic and neither stationary nor Markovian.

Common fundamental models for IVS include SABR, developed by Hagan et al. [2002], and SVI, introduced by Gatheral [2004]. In SABR both the underlying forward price and its volatility evolve stochastically, allowing for the realistic capture of skew and smile dynamics observed in interest-rate and other derivatives markets. In SVI models, the implied volatility surface is represented through a simple yet flexible parametric form that ensures no static arbitrage, making it a widely used tool for fitting and interpolating market-implied volatilities.

Recently, neural networks and machine learning models have been applied to the IVS problem. Bloch and Böök [2021] apply convolutional long short-term memory (LSTM) models to the IVS to form an encoding–forecasting structure, then perform a dimensionality reduction to learn the main volatility risk factors. This model was capable of accurately predicting the movements of the market's smiles.

While Ackerer et al. [2020] apply neural networks (NN) to perform deep smoothing of IVS, this method is designed to be applied to standard IVS models, providing a NN-based correction when traditional models violate no-arbitrage constraints or fail to replicate observed market prices.

Similarly, Kelly et al. [2023] use Convolution Neural Networks (CNN) to extract features of the IVS that respect locality and identify complex non-linear relationships. They then used these insights to generate portfolios with demonstrably high Sharpe ratios, leveraging CNN properties such as translational and rotational invariance, spatial hierarchy, and local connectivity. This work not only demonstrates the potential of IVS-informed machine learning based strategies but also emphasises the need for refined smoothing techniques.

3 Machine Learning Methodology and Data

3.1 Machine Learning Methodology

In machine learning terminology, modelling IVS corresponds to a supervised regression problem. We observe continuous outcomes $y_{t,i}$, the IV from the surface on day t, under financial observation i and aim to predict these outcomes accurately using various models. Below, we outline four distinct modelling approaches (more details can be found in [Scardapane, 2024]).

3.1.1 Linear Regression

Linear regression attempts to fit a linear relationship between the features (e.g., DTE, Volume, log-moneyness) and the target (IV). This model serves as our performance baseline due to its simplicity and interpretability. Mathematically:

$$\hat{y}_{t,i} = \beta_0 + \beta_1 x_{1,t,i} + \beta_2 x_{2,t,i} + \cdots + \beta_p x_{p,t,i} + \epsilon \tag{1}$$

With $\hat{y}_{t,i}$ being the predict IV, $x_{j,t,i}$ the j-th input feature for observation i at time t, β_j the model coefficients and an error term ϵ. The model is optimised by calculating coefficients $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)$ that minimize the loss function for regression, we use the residual sum of the squares (RSS):

$$\text{RSS} = \sum_{t,i} (y_{t,i} - \hat{y}_{t,i})^2 = \sum_{t,i} \left(y_{t,i} - \beta_0 - \mathbf{x}_{t,i}^\top \boldsymbol{\beta}\right)^2 \tag{2}$$

3.1.2 Neural Networks

A Neural Network is a machine learning model designed to mimic the structure and function of biological neural networks in brain or nerve clusters. Neural networks are powerful function approximators capable of learning patterns directly from data, and can model complex, non-linear relationships, making them better suited for capturing the curvature and structure of IVS. Mathematically, at each neuron a neural network functions like this:

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3}$$

Where x_i are the input features generated by the previous layer, w_i are the weights assigned to each connection between nodes, b is the bias a value added at each node, allowing them to activate even when the inputs are zero, and ϕ is the activation function that transforms the output of the node.

For a layer:

$$a = \phi(Wx + b)$$

where ϕ is applied element-wise to the result and Wx is a matrix vector multiplication.

For a network:

$$a^{(l)} = \phi^{(l)}(W^{(l)} a^{(l-1)} + b^{(l)}), \quad \hat{y} = W^{(L)} a^{(L-1)} + b^{(L)}$$

Optimization is performed using back-propagation, a loss function and a learning rule. For all our neural networks, we use mean squared error (MSE) as the loss function:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4}$$

The Adaptive Moment Estimation (Adam) optimizer is used as our learning rule:

$$w_{t+1} = w_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{5}$$

With α as the learning rate, $\hat{m}_t$ the bias-corrected mean gradient and $\hat{v}_t$ the bias-corrected squared gradient.

3.1.3 Feedforward Neural Network (FNN)

A Feedforward Neural Network (FNN) is a type of neural network where data flows strictly forward without cycles or loops. In IVS modelling, FNNs are used to learn the non-linear relationship between implied volatility and input features such as log-moneyness, days to expiry (DTE), and spot price, directly from historical data. For our FNN we use a Rectified Linear Unit (ReLU) activation function:

$$\phi(x) = \max(0, x)$$

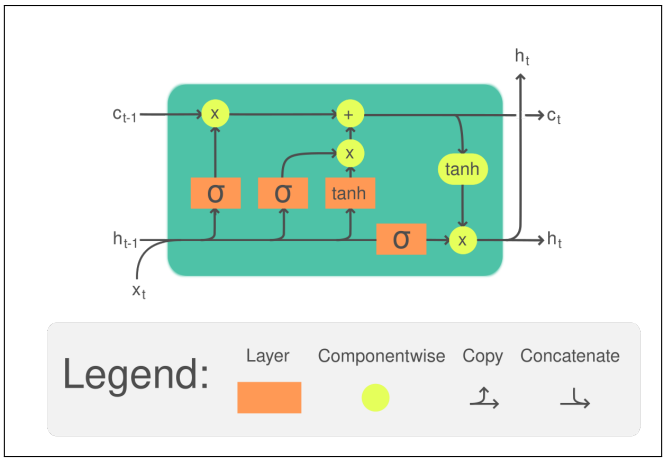

Figure 1: A diagram representing an LSTM cell. See section 3.1.5 for details.

3.1.4 Multilayer Perceptron (MLP)

Multilayer perceptrons (MLPs) are a subset of FNNs that meet two conditions:

- Fully connected, meaning every neuron in a layer is connected to every neuron in the next layer.
- At least one hidden layer (multilayer).

Aside from these conditions they function identically from a mathematical perspective.

3.1.5 Recurrent Neural Network (RNN)

To explore how IVSs evolve over time, we implement a Recurrent Neural Network (RNN) to generate curves that can adapt to daily volatility trends. A RNN is a neural network designed for sequential data. Unlike FNNs (which treat inputs independently), a RNN maintains a hidden state that allows it to retain information from previous time steps. Specifically, we will be using an long short-term memory (LSTM) model. LSTMs are designed to capture long-term dependencies in sequential data, making them a ideal model for IVSs which are commonly influenced by long- and short-term dynamics.

At each time step t, an LSTM processes an input vector $\boldsymbol{x}_t$, the previous hidden state $\boldsymbol{h}_{t-1}$, and the previous cell state $\boldsymbol{c}_{t-1}$ to compute the new hidden state $\boldsymbol{h}_t$ and cell state $\boldsymbol{c}_t$. The update is governed by the following equations:

$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) && \text{(forget gate)} \\
\boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) && \text{(input gate)} \\
\tilde{\boldsymbol{c}}_t &= \tanh\left(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right) && \text{(candidate cell state)} \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t && \text{(updated cell state)} \\
\boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right) && \text{(output gate)} \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) && \text{(updated hidden state)}
\end{aligned}
$$

Here, $\sigma(\cdot)$ denotes the sigmoid activation function, $\tanh(\cdot)$ is the hyperbolic tangent, and $\odot$ represents element-wise multiplication. The matrices $\boldsymbol{W}_*$, $\boldsymbol{U}_*$, and vectors $\boldsymbol{b}_*$ are learnable parameters for each gate. The gates $\boldsymbol{f}_t$, $\boldsymbol{i}_t$, and $\boldsymbol{o}_t$ control how information is passed through the memory cell, enabling the LSTM to learn long- and short-term dependencies in the data.

This system is visualised in Figure 1. The leftmost σ represents the forget gate, the middle σ is the input gate and the candidate cell state is represented by the middle $\tanh$. These three combine with the previous cell state $\boldsymbol{c}_{t-1}$ into the updated cell state by component-wise multiplication and then addition. The rightmost σ represents the output gate, and that feeds directly into the component-wise multiplication with $\tanh(\boldsymbol{c}_t)$ calculating the updated hidden state.

3.1.6 RNN and FNN Hybrid Approach

Finally, we explore a novel approach in which we use an LSTM RNN to model volatility across the time series, we will refer to this model as 'RNN for Vol'. This is based on realized volatility calculated over a 5-day rolling window, derived from log returns:

Log returns:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right) \tag{6}$$

Where P_t is the price of the underlying on day t and r_t is the log return on day t.

5-day sample standard deviation:

$$\sigma_t^{(5)} = \sqrt{\frac{1}{5}\sum_{i=0}^{4}\left(r_{t-i} - \bar{r}\right)^2} \tag{7}$$

Annualized realized volatility:

$$\text{RealizedVol}_t^{(5)} = \sigma_t^{(5)} \cdot \sqrt{252} \tag{8}$$

This realized volatility is then added as an additional input feature to our FNN model, which we will refer to as 'FNN (w/ RNN Vol)'. This allows it to better capture expected changes in future volatility by integrating temporal volatility dynamics into the predictive structure.

3.2 Data

The data set contains Apple option prices from 2016 to 2020, sampled daily at 16:00. With over one million unclean rows, extensive data cleaning was required. First, we separated put and call data from combined rows and normalized them into a standard format with a (P/C) flag. We then implemented log-moneyness as $\log(\text{strike}/\text{underlying})$, which is one of our primary features.

By plotting IV against volume and IV against days to expiry (DTE), we identified outliers with extremely high IV occurring mostly at very low DTE and low volume. Thus, we filtered out options with volume less than 5, IV greater than 300%, and IV less than 0.1%. We then removed any rows without IV, STRIKE, or DTE, indexed the data by datetime, and dropped unnecessary columns such as quote and expiry times in Unix format as these values would heavily affect model performance.

After checking the correlations of the variables, the final set of characteristics is:

$$\texttt{UNDERLYING\_LAST}, \texttt{DTE}, \texttt{STRIKE}, \texttt{STRIKE\_DISTANCE}, \texttt{STRIKE\_DISTANCE\_PCT},$$
$$\texttt{VOLUME}, \texttt{MID\_PRICE}, \texttt{LOG\_MONEYNESS}, \texttt{OPTION\_TYPE}(categorical)$$

Engineered features such as log-moneyness, strike distance, and strike distance pct were introduced to enhance model performance. We scaled the numerical features using scikit-learn's Pedregosa et al. [2011] RobustScaler and applied a OneHotEncoder for the categorical OPTION_TYPE. RobustScaler is preferred for financial data as it is more robust to outliers and uses the interquartile range (IQR) instead of the mean and standard deviation, see Eq. 9 & 10 below.

Standard Scaler:

$$x' = \frac{x - \mu}{\sigma} \tag{9}$$

Robust Scaler:

$$x' = \frac{x - Q_2}{Q_3 - Q_1} \tag{10}$$

Here, x is the original value and x' the scaled value, with μ and σ the mean and standard deviation of the feature. Q_1, Q_2 and Q_3 represent the 25th, 50th (median) and 75th percentiles respectively.

During exploratory data analysis, we accounted for Apple's 3:1 stock split on August 28th, 2020, and excluded data prior to the announcement on July 30th, 2020. We used a 80% training, 10% validation, and 10% test split, with implied volatility (IV) as the target variable.

3.3 Model Accuracy

To evaluate the performance of our IVS, we use a combination of four basic metrics on out-of-sample data. First, we apply our loss function, Mean Squared Error (MSE), to measure loss on the test set. We then calculate the Root Mean Squared Error (RMSE), as it is less sensitive to outliers and easier to interpret.

The Mean Absolute Error (MAE) is robust to outliers, as it computes the absolute value first, then takes the average, treating all errors equally, giving a clearer representation of typical absolute error.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{11}$$

where n is the number of data points , $\hat{y}_i$ is the model-implied volatility, and y_i is the market-observed implied volatility.

These measures are computed over a hold-out test set of option prices that were not used in model training. Lower values indicate better fit and generalization.

It is important to note the difference in scale between these metrics as MSE squares the errors, it is expressed in squared implied volatility units and can exaggerate the impact of larger errors. A 1% change in MSE does not translate directly to a 1% change in typical prediction error. In contrast, RMSE and MAE are both expressed in the same units as the target variable (i.e., implied volatility), thus a RMSE of 12% means the model is off by 0.12 'volatility points' on average, making them more interpretable and directly comparable to market spreads or model tolerances.

We also report the Coefficient of Determination, R^2, which measures the proportion of variance in observed IVs explained by our models.

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{12}$$

where y_i is the market IV, $\hat{y}_i$ is the model IV, and $\bar{y}$ is the mean of the market IV. The numerator is the residual sum of squares (model error), and the denominator is the total sum of squares (data variance).

R^2 values range from $-\infty$ to 1. Values closer to 1 indicate that the model explains more variance in the data. For instance, an R^2 of 0.788 means that approximately 79% of the variation in market-implied volatilities is captured by the model. Negative values imply that the model performs worse than simply predicting the mean.

4 Results

In this section, we examine performance metrics using two evaluation approaches. First, we assess out-of-sample performance on the full dataset to simulate an IVS model in production. Then, we apply time series cross-validation (TSCV) to evaluate generalisability to unseen data. Standard cross-validation is unsuitable here due to the temporal structure inherent to time series data.

4.1 Out-of-Sample Testing

We use an 80% training, 10% validation, and 10% test split. Table 1 summarizes performance across all models:

The MLP, FNN, and FNN (w/ RNN Vol) models achieve notably low MSEs and show strong generalization, as evidenced by their low RMSE and MAE values. The standard FNN benefits the most from the MAE's robustness to outliers. The RNN underperforms receiving statistical results on par with basic regression, indicating limited surface learning. The RNN for Volatility performs well when tasked with generating future volatilities for use in the FNN (w/ RNN Vol). The FNN (w/ RNN Vol) performs well, demonstrating the utility of incorporating volatility forecasting.

High R^2 scores for MLP, FNN, and FNN (w/ RNN Vol) suggest that they closely replicate the underlying structure of the IVS, as demonstrated in Figure 2. These models exhibit a clear volatility

Table 1: Out-of-sample testing results by model and metric

Metric	Regression	MLP	FNN	RNN	RNN Vol	FNN (w/ RNN Vol)
Test MSE	0.0606	0.0181	0.0127	0.0548	0.0071	0.0104
Test RMSE	0.2462	0.1347	0.1126	0.2342	0.0844	0.1019
Test R^2	0.3022	0.7911	0.8541	0.3685	0.4995	0.8804
Test MAE	0.1328	0.0783	0.0539	0.1202	0.0653	0.0665

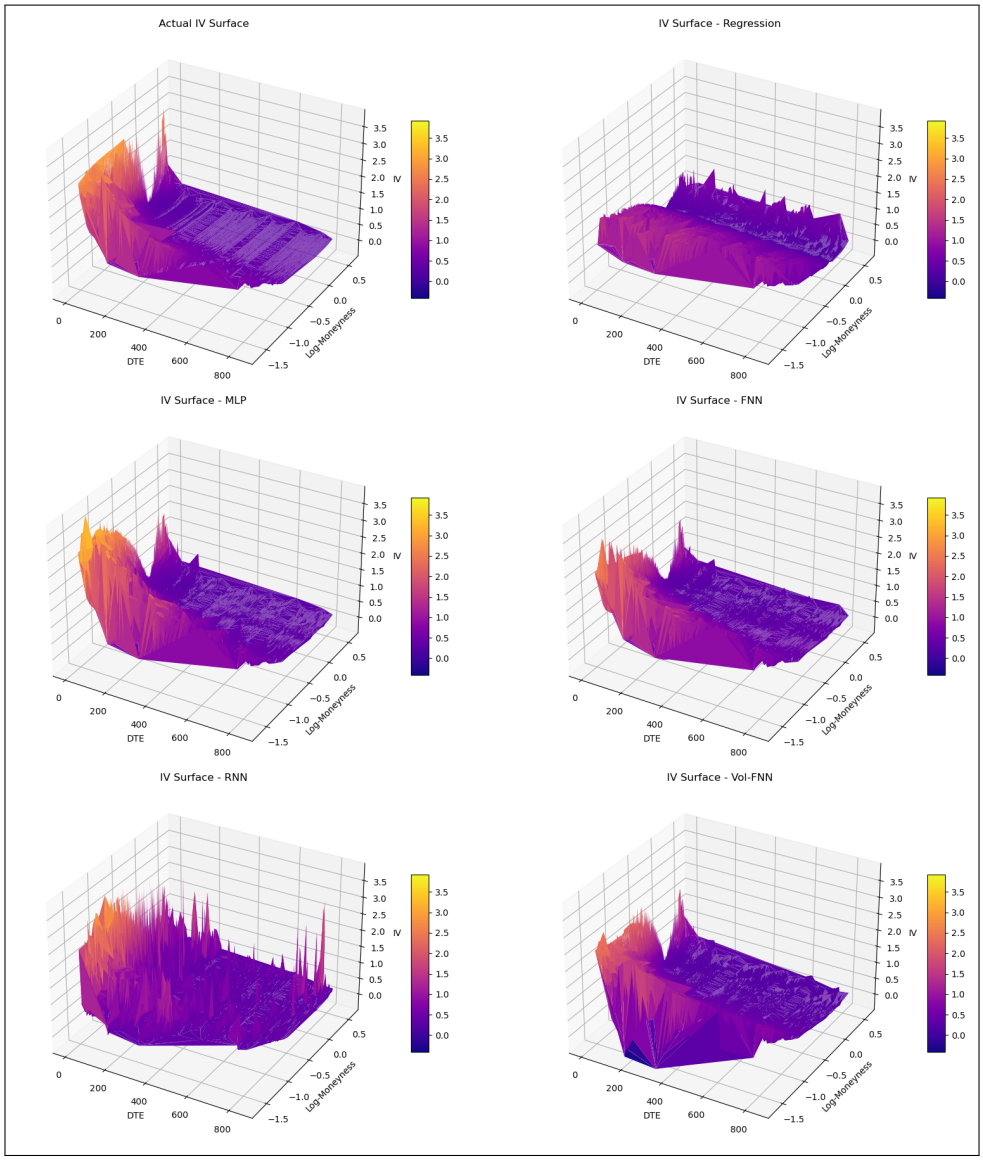

Figure 2: Implied volatility surface across the test set and models. With higher peaks and warmer colours representing greater implied volatility. The Z-scale is uniform across all graphs for ease of comparison.

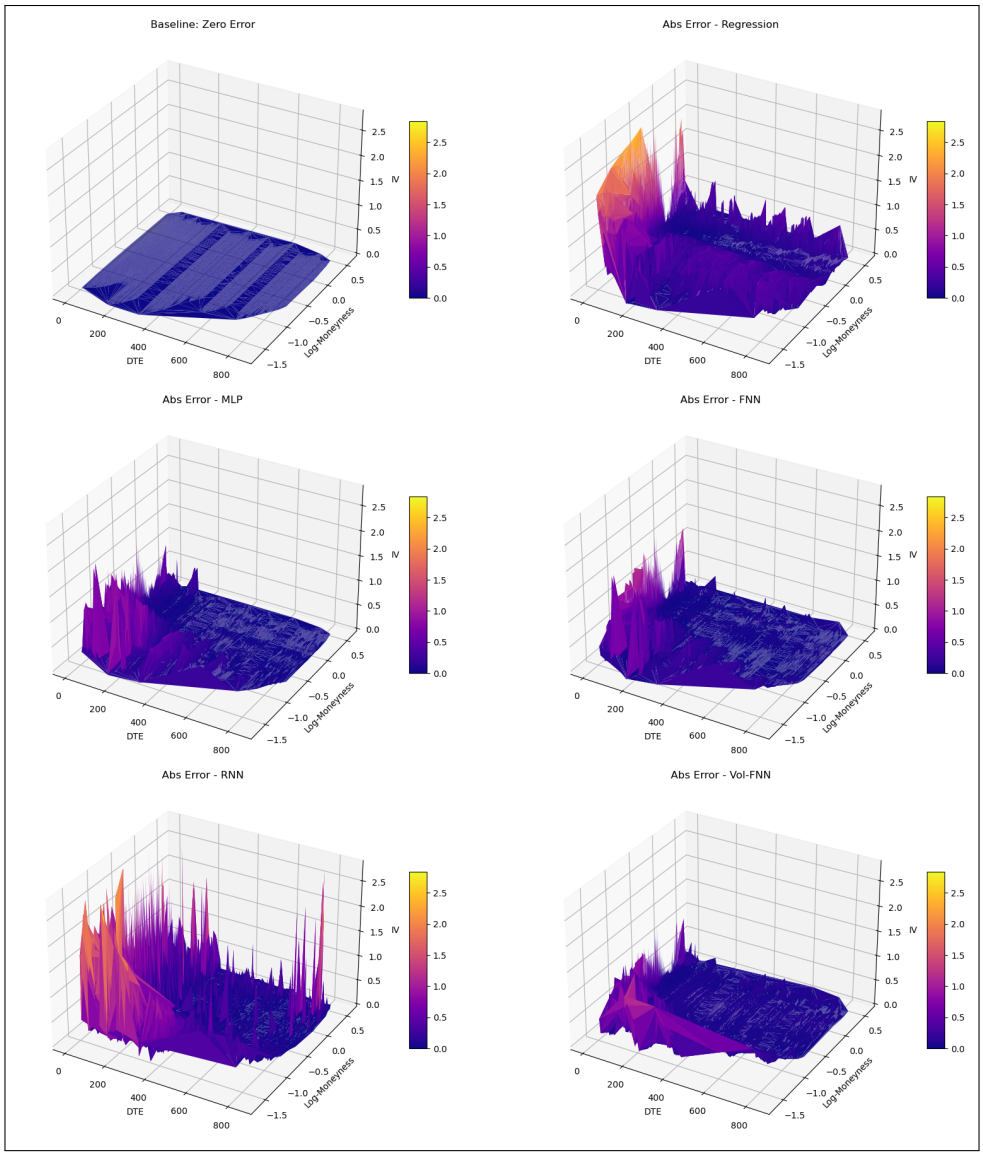

Figure 3: The absolute error $|\hat{y}_i - y_i|$ of implied volatility predictions across the test set and models. With higher peaks and warmer colours representing greater error. The Z-scale is uniform across all graphs for ease of comparison.

'smile' and are skew-consistent within option pricing theory. Although RNN and regression models yield similar statistical performance, their implied surfaces differ noticeably. Regression captures the 'smile' of the surface well but is not complex enough to model the non-linear relations of the surface. Whereas, the RNN struggles to effectively capture the characteristics of the surface in any way, this may be due to over parametrization of the model.

We now examine the absolute error of the surface shown in Figure 3. The performance in these plots should correspond to the MAE scores. The FNN (w/ RNN Vol) is most effective on low-DTE options but struggles with low log-moneyness options. This is likely due to the 5-day realization horizon in the volatility model. Although MLP and FNN also perform well, they underperform in short-dated options compared to the hybrid model. The regression model has high absolute error in areas of high surface volatility due to its simplistic shape. The RNN has an unpredictable surface and unpredictable error.

Table 2: Time series cross-validation results by model and metric

Model	Metric	K = 1	K = 2	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8	Mean
Regression	MSE	0.0130	0.0147	0.0242	0.0307	0.0660	0.0218	0.0455	0.0600	0.0345
	RMSE	0.1141	0.1214	0.1554	0.1753	0.2568	0.1477	0.2133	0.2450	0.1786
	R^2	0.2220	0.2491	0.3259	0.4473	0.2649	0.3121	0.4692	0.3090	0.3249
	MAE	0.0693	0.0684	0.0804	0.0752	0.1342	0.0794	0.1206	0.1303	0.0947
MLP	MSE	0.0065	0.1196	0.0080	0.0072	0.0144	0.0036	0.0087	0.0118	0.0225
	RMSE	0.0805	0.3458	0.0896	0.0850	0.1198	0.0601	0.0933	0.1088	0.1229
	R^2	0.6127	-5.0896	0.7758	0.8701	0.8400	0.8861	0.8986	0.8638	0.0822
	MAE	0.0385	0.2181	0.0375	0.0342	0.0604	0.0282	0.0450	0.0524	0.0643
FNN	MSE	0.0050	0.0278	0.0073	0.0047	0.0111	0.0029	0.0118	0.0104	0.0101
	RMSE	0.0705	0.1667	0.0854	0.0682	0.1056	0.0539	0.1086	0.1022	0.0951
	R^2	0.7032	-0.4156	0.7963	0.9163	0.8758	0.9084	0.8624	0.8797	0.6908
	MAE	0.0249	0.1136	0.0354	0.0274	0.0464	0.0225	0.0509	0.0474	0.0461
RNN	MSE	0.0181	0.0195	0.0275	0.0400	0.0588	0.0203	0.0488	0.0485	0.0352
	RMSE	0.1345	0.1395	0.1659	0.2000	0.2426	0.1424	0.2209	0.2203	0.1832
	R^2	-0.0804	0.0083	0.2321	0.2808	0.3442	0.3614	0.4309	0.4412	0.2523
	MAE	0.0887	0.0793	0.0675	0.1005	0.1230	0.0696	0.1006	0.1127	0.0927
RNN Vol	MSE	0.0347	0.0074	0.0016	0.0044	0.0982	0.0106	0.0194	0.0149	0.0239
	RMSE	0.1864	0.0858	0.0401	0.0667	0.3134	0.1031	0.1392	0.1221	0.1321
	R^2	-5.0424	-0.0806	0.0112	-0.5206	-1.6478	-0.0252	-4.4776	-0.0475	-1.4788
	MAE	0.1736	0.0706	0.0304	0.0565	0.2468	0.0747	0.1283	0.1040	0.1106
FNN (w/ RNN Vol)	MSE	0.0136	0.0076	0.0091	0.0103	0.0168	0.0047	0.0156	0.0108	0.0111
	RMSE	0.1166	0.0871	0.0955	0.1013	0.1298	0.0685	0.1251	0.1040	0.1035
	R^2	0.1869	0.6132	0.7453	0.8155	0.8123	0.8521	0.8176	0.8755	0.7148
	MAE	0.0816	0.0496	0.0447	0.0475	0.0782	0.0327	0.0639	0.0706	0.0586

4.2 Time Series Cross-Validation (TSCV)

For TSCV, the data set is divided into nine six-month windows. For each window K, models are trained in all preceding periods and tested on a 3-month validation and 3-month test set in window $K + 1$. This results in eight validation rounds. The 8-fold cross-validation result is the mean of all these trials.

The results in Table 2 show that model performance improves as training history increases. The RNN shows a negative R^2 in the initial rounds, with improvement as more data becomes available, suggesting overparameterization and sensitivity to data size.

Both MLP and FNN models produce negative R^2 scores in the second fold ($K = 2$), probably due to outliers. This indicates the need for additional exploratory data analysis. Interestingly, although RNN for Vol improves the accuracy of the hybrid model in earlier tests, it delivers inconsistent performance in TSCV with multiple negative R^2 scores, raising concerns about its practicality in real-world deployment.

5 Discussion

Our results clearly show that for general IV surface modelling, the Feedforward Neural Network approach is best. The MLP, FNN, and FNN (w/ RNN Vol) models all perform well statistically and accurately capture the market observed surface structure. The integration of temporal volatility dynamics gives FNN (w/ RNN Vol) the strongest performance overall.

While the RNN for Vol effectively models the volatility time series, the base RNN is not well-suited for IV modelling, as shown by both its metrics and surface shape. TSCV results confirm that most models generalize well to new data, consistent with their test set performance—except the RNN for Vol, which may require further development.

Since we treated the dataset as accurate, we neglected to perform our own feature engineering and initial IV calculations, potentially introducing data errors. We mitigated this risk with thorough preprocessing and EDA. We also assumed daily evolution of IVS for the LSTM, though in reality, intraday trends may be more revealing.

As shown by Kelly et al. [2023], deep learning offers powerful tools for identifying and exploiting market inefficiencies to generate alpha—returns in excess of a benchmark. As investors embrace

and develop these models, the process of option pricing must also evolve. The results show here are particularly relevant for pricing options or training models to spot option mispricing. Our results suggest that deep learning models can improve upon traditional parametric approaches, especially when dealing with large, multi-dimensional options datasets like the one used here.

However, our models are limited to AAPL-specific data and may not generalize across assets. We did not explicitly model structural breaks such as the COVID-19 crash. Our model may perform better if we directly integrated traditional approaches, or if we enforced no-arbitrage conditions such as including checks for calendar spread arbitrage (monotonicity in expiry), butterfly arbitrage (convexity in strike), and vertical spread arbitrage when training our models.

Further studies could explore the applications of custom loss functions to enforce no-arbitrage in the training of models or develop a FNN for smoothing surfaces under no-arbitrage conditions trained on traditional methods, and then apply that FNN to a RRN as explored here for a hybrid approach. Finally, a Conditional Generative Adversarial Network (cGAN) could be used to generate surfaces from market like conditions. This cGAN method could be expanded by training the discriminator to detect no-arbitrage conditions on the surface generated. Such a framework would enable a single party to train a model capable of both exploiting market inefficiencies and accurately pricing options for sale.

References

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973. doi: 10.1086/260062.

Giuseppe Orlando and Giovanni Taglialatela. A review on implied volatility calculation. *Journal of Computational and Applied Mathematics*, 329:203–221, 2018. doi: 10.1016/j.cam.2017.05.016.

Cboe Global Markets. Cboe american-style options implied volatility calculations methodology, 2023. Accessed: 2024-03-31.

John C. Cox, Stephen A. Ross, and Mark Rubinstein. Option pricing: A simplified approach. *Journal of Financial Economics*, 7(3):229–263, 1979. doi: 10.1016/0304-405X(79)90015-1.

Patrick S. Hagan, Deep Kumar, Andrew S. Lesniewski, and Diana E. Woodward. Managing smile risk. *Wilmott Magazine*, 2002(September):84–108, 2002.

Jim Gatheral. A parsimonious arbitrage-free implied volatility parameterization with application to the valuation of volatility derivatives. Presented at Global Derivatives and Risk Management Conference, 2004.

Campbell R. Harvey and Robert E. Whaley. Market volatility prediction and the efficiency of the s&p 100 index option market. *Journal of Financial Economics*, 31(1):43–73, 1992. doi: 10.1016/0304-405X(92)90015-E.

Bernard Dumas, Jeff Fleming, and Robert E. Whaley. Implied volatility functions: Empirical tests. *The Journal of Finance*, 53(6):2059–2106, 1998. doi: 10.1111/0022-1082.00083.

Emanuel Derman and Iraj Kani. Riding on a smile. *Risk*, 7(2):32–39, 1994.

Bruno Dupire. Pricing with a smile. *Risk*, 7(1):18–20, 1994.

Mark Rubinstein. Implied binomial trees. *The Journal of Finance*, 49(3):771–818, 1994. doi: 10.1111/j.1540-6261.1994.tb00079.x.

Daniel Alexandre Bloch and Arthur Böök. Deep learning based dynamic implied volatility surface. *SSRN Electronic Journal*, 2021. doi: 10.2139/ssrn.3952842.

Damien Ackerer, Natasa Tagasovska, and Tobias Vatter. Deep smoothing of implied volatility surfaces, 2020.

Bryan T. Kelly, Boris Kuznetsov, Semyon Malamud, and Teng Andrea Xu. Deep learning from implied volatility surfaces. *SSRN Electronic Journal*, 2023. doi: 10.2139/ssrn.4531181. Swiss Finance Institute Research Paper No. 23-60.

Simone Scardapane. *Alice's Adventures in Differentiable Wonderland: Designing Neural Networks with Intuition*. Independently published, 2024.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.