SurfaceEdge: Predicting Next-Day Options Prices from Options Surface Images Using a Hybrid CNN-Transformer Architecture

Caleb Bettcher
University of Colorado Boulder
calebbettcher@gmail.com

Troy Arthur
University of Colorado Boulder
troy.arthur@colorado.edu

Abstract

Deep learning has shown significant potential in derivatives pricing, but existing models often treat options contracts in isolation and frequently suffer from temporal data leakage. To address these limitations, we introduce SurfaceEdge, a novel architecture which predicts next-day options price movements by treating the broader market options surface as a visual object. Through an encoding of implied volatility, open interest, and volume across a grid into an RGB image, SurfaceEdge leverages a Convolutional Neural Network to extract cross-contract spatial features. We further extend this baseline with a causal self-attention mechanism to capture historical dynamics. Evaluated on a strictly chronologically-split dataset of major U.S. firms, our models consistently outperform a naive baseline. The historical self-attention architecture achieves the best overall performance, outperforming the naïve baseline by approximately 8% (validation set model MAE of 0.127 vs. naïve MAE 0.138), and demonstrating that historical contract modeling with denoised inputs offers significant promise in derivatives pricing.

0 Preliminaries

Our work is centered around the financial derivatives market, which carries specialized terminology. We provide the definitions in Table 1 to ensure clarity throughout this paper for readers who may be unfamiliar with this domain.

Table 1: Common options terminology.

Term	Definition
Call / Put	The right to buy vs. the right to sell
Options contract	The right (not obligation) to buy or sell an asset at a fixed strike price before expiration
Spot price	The current fair market value of an asset
Strike price (K)	The fixed price at which an options contract can be exercised
Mark price	A contract's fair market value; midpoint of bid and ask prices
Premium	Total cost to purchase one contract; = mark price $\times$ 100 shares
Time to expiry (τ)	The number of days until a contract expires
Intrinsic value	Immediate exercisable value; for a call, $\max(S-K,0)$
In-the-money (ITM)	A call where spot price exceeds strike price
Out-of-the-money (OTM)	A call where spot price is below strike price; full mark price is time value
Log-moneyness	$\ln(K/S)$, normalized distance of strike from spot price
Implied volatility (IV)	Market's forward-looking volatility expectation implied by mark price
Open interest	Total outstanding contracts at a specific strike and expiry
Volume	Contracts traded at a specific strike and expiry on a given day
Volatility surface	2D surface of implied volatility across all strikes and expiries
Options surface	Broader representation of options parameters across strikes and expiries

On April 23, 2026, Apple (AAPL) trades at a spot price of \$271.06. You purchase a call option with a strike price of \$275.00 expiring May 6, 2026 ($\tau = 13$) for a mark price of \$8.80 per share (\$880 per contract). There are currently 3 contracts of Open Interest on this strike/expiry combination.

This option is currently out-of-the-money (log-moneyness $= \ln(275/271.06) \approx +0.0145$). The spot price must rise above \$275.00 before May 6, 2026 for the contract to have intrinsic value, with the current \$8.80 mark price representing time value (the probability the spot prices moves ITM increases with more time). If AAPL rises to \$278.00 the following day, the contract moves in-the-money (log-moneyness $= \ln(275/278.00) \approx -0.0108$) and the mark price increases accordingly. Our model aims to predict this next-day mark price change using options surface data from April 23, 2026.

1 Introduction

The Options market for any given underlying equity is a dense representation of the market's current consensus on both the near, and long term price of

that stock. Accurately pricing these options contracts is of critical importance to both institutional and retail traders alike, as even a small edge in the pricing ability can result in several percentage points of returns, which can result in hundreds of millions of dollars of additional revenue for top firms. Each trading day, millions of contracts are priced by the collective activity of market participants, producing a rich two-dimensional structure known as the implied volatility surface for each underlying equity. This surface spreads across a wide range of strikes and expiration dates, with each point encoding the market's collective consensus on future price uncertainty for a specific strike, over a certain time period. This surface is constantly updated in real-time to reflect the most up-to-date news, and general market consensus as participants trade these options contracts.

Recent advancements of Deep Learning architectures have become of particular interest to financial market researchers due to their ever-evolving capabilities to learn highly complex patterns from enormous training sets. Building upon existing research, a plethora of literature centered around options pricing using deep learning has blossomed, with many of these approaches using LSTM-based architectures to account for the time series nature of market data, or defining each contract through independent scalars passed to a model. While several of these models provide promising results and some even with proven revenue through backtesting, we believe they often produce false positives by overlooking temporal context in train/test splits, and none leverage the full spatial structure of the options surface as a visual input. Our novel approach encodes daily options surfaces for an underlying equity through a visual object processed through a convolutional feature extraction model under the assumption that current market conditions carry predictive signals for future price when amended to contract-level scalar inputs.

We introduce SurfaceEdge, a hybrid convolutional and embedding architecture designed to predict the next-day percentage change in an individual option contract's mark price. Rather than treating each contract as an isolated set of scalar parameters, SurfaceEdge encodes the full daily options surface as a 60×30 RGB image mapping implied volatility, open interest, and volume respectively using the Red, Green, and Blue image channels across a log-moneyness by time-to-expiry grid. We use this image to extract market-wide spatial context through a convolutional encoder. We hypothesize that the intrinsic noise suppression properties of convolutional architectures are particularly well-suited to the chaotic nature of financial data, producing surface-level embeddings that capture cross-contract pricing relationships invisible to scalar-only approaches. We use these embeddings in a variety of models, including a baseline fusion model combining the convolutional embedding with learned ticker embeddings and contract-specific scalars, as well as causal forecasting via historical self-attention architecture incorporating sequential surface context. To our knowledge, no prior work has treated the options surface as a visual object for this purpose, making SurfaceEdge's core design philosophy a novel contribution to derivatives pricing literature.

In this work, we evaluate SurfaceEdge across three model configurations of increasing complexity: a baseline fusion model, a deep residual prediction head, and a causal self-attention model incorporating sequential contract history. In addition, we explore the use of social media text embeddings as a supplementary signal, in combination with our best performing causal self-attention model. Out of major concerns regarding limited compute, all models are trained and evaluated on a chronologically-split subset of our final dataset containing the largest 4 out of the 104 total tickers (AAPL, MSFT, GOOGL, and AMZN). This subset comprises approximately 1.1 million training and 347,000 test contracts, or about 1.45 million of the 30.5 million contracts available. We find that all of our model configurations consistently outperform the naïve baseline, with the causal self-attention model achieving the strongest performance at a MAE of 0.127, or an 8% improvement over the naïve baseline MAE of 0.138. We saw nearly identical model results when incorporating BlueSky social media data, most likely due to the sparsity of labels. These results suggest that treating the options surface as a structured visual input provides meaningful predictive signal, and that incorporating sequential contract history further enhances model capacity.

2 Related Work

Building on top of recent major technological breakthroughs, applications of deep learning to financial markets has been a rapidly developing field of literature in recent years. Radfar [10] evaluates

various common deep learning architectures used for financial analysis, including LSTMs and Transformers, as well as less studied architectures such as CNNs, for stock trend prediction on time series data using contract-specific parameters. It was found that CNN models outperformed LSTMs due to their greater resilience to noise and randomness, a property particularly valuable in financial markets where price dynamics are highly stochastic. Critically, Radfar identifies and critiques the use of randomly shuffled time series data during train/test splitting, as it allows models to exploit price-level trends, largely inflating reported accuracy through data leakage. Radfar demonstrates this key methodological flaw is present in a significant portion of current LSTM and DNN-based literature. One such example is from Alam et al. [1], where a hybrid LSTM-DNN architecture was proposed and evaluated across 26 real-life stock datasets, reporting an average R^2 score of 0.986. Despite being peer-reviewed and using genuine market data, as Radfar identifies, the uniformly high R^2 values across diverse stocks are characteristic of a model tracking the underlying price level trend rather than predicting meaningful price movements. While our methodology differs from these works through our introduction of image-based surface features capturing cross-contract spatial structure, we also directly address Radfar's critique by enforcing a strict chronological train/validation split throughout all evaluations.

Chen, Pelger, and Zhu [4]'s theoretically grounded approach to financial prediction combines three distinct neural network architectures in a unified asset pricing framework: a feedforward network modeling the stochastic discount factor, a recurrent LSTM network extracting economic state variables, and a generative adversarial network identifying portfolios with the largest pricing errors. This approach reduces unpredictable noise and allows the model to extract more meaningful variation from systematic risk. Importantly, the authors demonstrate that macroeconomic variables become uninformative when only their most recent increments are used as inputs, meaning the LSTM is essential for recovering the cyclical dynamics that carry pricing information but are lost when the full time series history is discarded. In contrast, our baseline model operates on a single-day snapshot of an options surface, with the assumption that current market conditions fairly price larger scale macroeconomic trends. However, our work similarly combines complementary architectural components into a unified predictive framework, motivating our use of a CNN encoding mechanism in combination with contract-specific scalar inputs, and further motivating our extension to a self-attention head that conditions predictions on sequential historical surface observations.

The use of a CNN as our primary feature extraction layer is motivated both by Radfar's findings of its advantages to other architectures, as well as the demonstrated noise immunity properties of convolutional architectures documented by Tuchin et al. [12]. On the contrary, Bao, Yue, and Kong [2] combine various noise reduction preprocessing steps such as wavelet transforms, stacked autoencoders, and LSTMs in an effort to suppress the chaotic noise in financial markets. While this approach demonstrates consistent outperformance of the S&P 500, we suspect the intrinsic spatial averaging of a CNN is better suited to suppress idiosyncratic contract-level noise during feature extraction of the unpredictable and notoriously difficult to model randomness of financial markets. Additionally, our unique approach of aggregating multiple contracts into a single surface image prior to the CNN ever receiving contract-specific inputs naturally includes an additional layer of smoothing at the data construction stage, while also providing opportunity for traditional image processing techniques.

Using the options surface itself as a structured input is grounded through extensive research in the application of neural networks to derivatives markets. Hutchinson, Lo, and Poggio [7] were among the first to demonstrate that a neural network can recover the Black-Scholes pricing formula directly from market data using only moneyness and time to expiry as inputs, establishing the viability of data-driven options modeling without parametric assumptions. When applied to S&P 500 futures, this approach demonstrated that a learned pricing formula outperformed conventional parametric alternatives in both pricing and delta hedging. This supports the idea that neural networks and their capability of learning complex, nonlinear relationships are well suited to derivatives pricing without explicit specification of the underlying mathematical structure. Ruf and Wang [11] further explore the versatility of neural networks in derivatives pricing by surveying over 150 subsequent papers on this topic. In particular, two recurring limitations were identified: most models treat each contract

in isolation rather than leveraging the spatial structure described by related contracts, and a significant fraction of published results are inflated by train/test partitioning that violates the time series structure of the data, independently corroborating Radfar's critique. Our work actively incorporates proper time-series train/validation segmentation to prevent data leakage, and addresses the lack of cross-contract context by framing a full day's options surface as an RGB grid encoding various parameters across strikes and expiries. To our knowledge, this particular approach has not appeared in prior work.

The extension of SurfaceEdge to incorporate sequential surface context builds on a growing body of literature demonstrating the effectiveness of transformer-based architectures for financial time series prediction. Zhou et al. [13] propose a transformer architecture designed specifically for long-sequence time-series forecasting, demonstrating strong performance on multi-step prediction tasks highly dependent on capturing long-range temporal dependencies. In another example, Olorunnimbe and Viktor [9] demonstrate that temporal transformers with similarity embeddings outperform classical models such as ARIMA and GARCH, as well as standard deep learning architectures for multi-horizon financial forecasting across volatile and non-volatile market conditions. Critically, they find that the optimal historical lookback window for a temporal transformer on financial data is 1 to 3 years, with sliding windows significantly improving generalization. Our Causal Self-Attention Forecasting model is directly motivated by these findings, treating each day's SurfaceEdge embedding as a token in a causal sequence, conditioning next-day price predictions on a window of prior surface observations for the same contract. Unlike prior temporal transformer applications to financial data, our tokens encode the full spatial structure of the options surface rather than scalar price features, combining the representational advantages of convolutional surface encoding with the temporal modeling capacity of self-attention.

3 Methods

3.1 Dataset and Surface Image Construction

We use historical end-of-day options chain data for 104 U.S. equities and ETFs spanning 2008 to 2025, sourced from the philippdubach/options-data repository [5]. Each ticker provides two files: a daily price history of the underlying equity containing adjusted close prices, split coefficients, and dividend records; and an options chain parquet containing contract-level parameters such as implied volatility, open interest, volume, pricing, and Greeks. The full raw dataset contains approximately 200 million option contracts across all tickers, representing nearly 10 GB of on-disk storage. This dataset was specifically chosen for its size, diversity, and public accessibility, as institutional-grade options data pipelines were outside the scope of this project.

A defining aspect of SurfaceEdge is the representation of daily options chains as images for use with convolutional feature extraction. We independently construct a surface image for both calls and puts for each ticker on each trading day. Options are filtered using a time-to-expiry window of 1 to 61 days and a log-moneyness range of -1.0 to $+1.0$, where log-moneyness is defined as:

$$x = \ln\left(\frac{K}{S}\right) \tag{1}$$

where K is the strike price of the contract and S is the spot price of the underlying equity. This time and moneyness window was selected for its balance between capturing the most liquid near-term contracts while also including longer-dated positions without expanding too far into sparse, and illiquid long-dated LEAPS with highly speculative equity pricing falling well outside the realm of calculated risk. These dimensions maximize the population density of actively traded contracts, ensuring a meaningful training signal is present in each image.

To ensure historical price consistency, strike prices must be adjusted for stock splits by multiplying each historical strike by the product of all split coefficients occurring after each respective trading date, expressing all contracts in consistent post-split terms. Without this critical step, log-moneyness values computed before a major split event would be systematically offset from post-split values, breaking the spatial semantics of the standardized options surface grid.

Following split adjustment, contracts are binned into a fixed 60×30 grid parameterized by log-moneyness on the y-axis and days-to-expiry on the x-axis, using the discussed bounds of 1 to 61 days, and -1.0 to $+1.0$ log-moneyness range. Bin edges are fixed globally across all tickers and dates, ensuring the same spatial location in any surface image

always corresponds to the same economic coordinates regardless of ticker or date. This global standardization is crucial for the CNN to learn transferable features across tickers and time, insulating the model from spurious correlations tied to specific market regimes. Within each bin, contracts are aggregated by taking the mean implied volatility and summing open interest (OI) and volume. These three quantities are mapped to the R, G, and B image channels respectively, with the OI and volume channels log-transformed to compress their dynamic range, then all three channels are independently min-max normalized to $[0, 255]$:

$$R = 255 \cdot \frac{\sigma_{IV} - \min(\sigma_{IV})}{\max(\sigma_{IV}) - \min(\sigma_{IV})} \tag{2}$$

$$G = 255 \cdot \frac{\log(1+OI) - \min(\log(1+OI))}{\max(\log(1+OI)) - \min(\log(1+OI))} \tag{3}$$

$$B = 255 \cdot \frac{\log(1+V) - \min(\log(1+V))}{\max(\log(1+V)) - \min(\log(1+V))} \tag{4}$$

where σ_{IV} is the mean implied volatility per bin, OI is the summed open interest per bin, and V is the summed volume per bin. Days where fewer than 100 binned grid cells contain a valid next-day label are discarded as too sparse to provide any meaningful training signal.

3.2 Scalar Features, Labels, and Naive Baseline

Alongside each surface image, we store a $60 \times 30 \times 14$ array of per-cell scalar features. Each cell corresponds to an individual grid strike/expiry bin combination on the surface image, and contains the features listed in Table 2.

Table 2: Per-cell scalar features stored alongside each surface image.

Feature
Days to expiry (τ)
Log-moneyness (x)
Today's mark price
Option type (call/put)
Delta
Gamma
Vega
Theta
Normalized bid-ask spread
Annualized dividend yield
Days to next ex-dividend date
5-day underlying momentum
20-day underlying momentum
Mean implied volatility

Additionally, each sample is provided with an integer ticker index in the range $[0, 103]$, derived from the containing folder name rather than stored on disk. This ticker index is passed to a learned embedding layer during training to capture ticker-specific surface characteristics.

Labels for each contract are defined as the next-day percentage change in mark price:

$$\ell = \frac{\text{mark}_{t+1} - \text{mark}_t}{\text{mark}_t} \tag{5}$$

where mark $= (\text{bid} + \text{ask})/2$ is the bid/ask midpoint. For each contract on day t, the next-day mark price (mark_{t+1}) is found using the contract identifier as a matching key. We exclude contracts with no next-day quote or a zero mark price on either day. Expressing the label as a signed decimal is a deliberate design choice, producing a scale-invariant label that treats a \$0.50 move on a \$1.00 contract and a \$50 move on a \$100 contract equivalently, which is critical across 104 tickers spanning a wide range of absolute price levels.

For evaluation, we define a naive baseline which predicts zero price change for every contract ($\hat{\ell} = 0.0$). While this assumption is obviously untrue, this baseline has two important properties. First, in combination with Mean Absolute Error as the evaluation metric, the naive MAE of any dataset directly equals the average absolute daily price movement of any given contract in that set, providing a meaningful normalization datapoint for highly volatile assets and an easy comparison for contracts too volatile to carry any significant training signal. Second, predicting zero change for all contracts produces an MAE equivalent to randomly assigning price changes, since a random assignment produces equal positive and negative movements whose absolute values average to the same quantity. This means the naive MAE represents the absolute minimum bar a model must clear to demonstrate that its inputs carry genuine predictive information beyond randomly guessing. All model performance is reported relative to this baseline.

3.3 Filtering and Train/Validation Split

Initial analysis of the raw labeled dataset of approximately 36.2 million contracts revealed two categories of contracts that inflate the naive baseline MAE through heavy-tailed label distributions, unrepresentative of typical contract pricing behavior.

First, contracts with next-day price changes exceeding 200% in absolute value ($|\ell| > 2.0$) are dominated by extremely illiquid or deep out-of-the-money positions where the mark price can move dramatically on negligible trading activity. Our dataset contains a disproportionate number of these contracts, likely reflecting the outsized retail and speculative interest in high-risk, high-reward positions that characterizes much of the options market activity captured in publicly available data.

Second, contracts with a normalized bid-ask spread greater than 0.5 are too illiquid to be practically tradeable and introduce labels driven by microstructure noise rather than genuine price discovery, where spread is defined as:

$$s = \frac{\text{ask} - \text{bid}}{\text{mark}} \tag{6}$$

This condition identifies a complementary category of illiquid contracts whose remaining time value partially stabilizes day-to-day mark price movement, preventing them from exhibiting the extreme $|\ell| > 2.0$ changes filtered above. Yet these wide spreads indicate they are thinly traded and priced primarily by market makers rather than genuine two-sided activity. Removing both categories consolidates the training data around actively traded contracts whose price movements reflect real market consensus rather than the erratic behavior of speculative or neglected positions whose price may collapse or double overnight based on a single earnings report from the underlying company.

Additionally, any trading days with fewer than 100 contracts remaining after filtering are dropped entirely for the same sparsity reasoning described earlier. Applying these three conditions reduces the dataset from 36.2 million to 30.5 million contracts, retaining 84.4% of the original data. The naive baseline MAE drops from 1.054 (105.4% average daily price movement) to 0.132 (13.2%), an 87.5% reduction, reflecting the outsized influence of extreme tail-end contracts on raw options price distributions. The filtered dataset is what all model training and evaluation in this paper are conducted on.

As discussed by Radfar [10] and Ruf and Wang [11], a common oversight in financial deep learning research is the use of random sampling for train/test splitting, which introduces data leakage by providing the model with implicit context about future prices. To address this, we define a clear chronological split date of November 21, 2024. Approximately 80% of dataset samples occur prior to this date and are assigned to the training set; all samples on or after this date are reserved for the validation set. While this raises concern that past performance may not indicate future results, the alternative would produce highly unrealistic and overinflated test results that do not reflect real-world deployment conditions.

3.4 Dataset Subset Selection and Feature Normalization

While the full options dataset spans 104 U.S. equities and ETFs, primarily out of concerns of limited time and compute resources, all models in this work are trained and evaluated on a subset of four large-cap equities: Apple (AAPL), Microsoft (MSFT), Alphabet (GOOGL), and Amazon (AMZN). These four tickers represent the largest portions of the full options dataset, are highly liquid, with dense options chains and consistent trading history ensuring a high-quality training signal relative to less actively traded tickers in the full dataset, while simultaneously reducing the total compute required for our initial experimentation of SurfaceEdge's novel architecture. Following the chronological train/validation split described in Section 3.3, this subset yields approximately 1.1 million training contracts and 347,000 validation contracts. The final naive MAE of this 347,000 contract validation set is 0.13762 which will be used as the final evaluation metric for our various models.

Each training sample consists of the SurfaceEdge image, an index for the corresponding ticker (AAPL vs. MSFT), and the 14 discussed scalar inputs. The SurfaceEdge image is normalized through subtracting from each pixel within each of the three RGB dimensions the average corresponding dimension's pixel value across the entire training dataset, and then dividing by the standard deviation of the corresponding training set pixel dimensions. The scalars are normalized similarly, by subtracting the training set's corresponding scalar average and dividing by the associated standard deviation.

3.5 Baseline Model Design

We begin with a baseline model design against which future experiments will be compared. As a starting point, we construct a model which does not incorporate time-series for contracts, but rather makes a single prediction for each contract for each timestep, with the contract's corresponding data

taken in isolation. At a high level, we construct a dense feature vector which contains all of the hypothesized relevant information for the corresponding entry. The feature vector is then passed through linear layers to make a single prediction, being the percent change in the price of the underlying asset. See figure 1 for a high-level conceptualization of the baseline model.

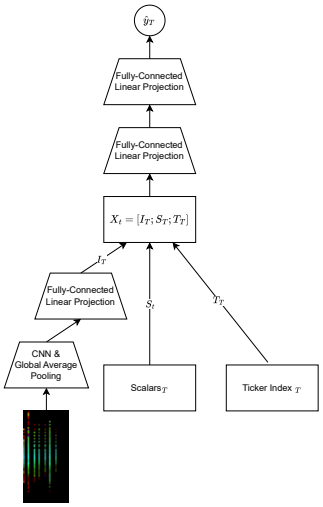

Figure 1: Baseline Model Design

3.5.1 Feature Vector Construction

In order to extract meaningful image features, we rely on a Convolutional Neural Network backbone. The image processing involves three sequential convolutional blocks. Each block applies a two-dimensional convolution with a 3x3 kernel, a stride of 1, and zero-padding of 1 to maintain image dimensions, followed immediately by a ReLU activation function and a 2x2 max-pooling operation with a stride of 2 to downsample the feature maps. The number of channels increases across the blocks from 32, to 64, to 128. Then, to prepare the data for a flat feature vector, the spatial dimensions of the resulting feature maps are reduced to 1x1 through global average pooling. The resulting 128-dimensional feature vector is passed through a dense linear layer which projects this vector to d_{img}, a dimension determined during hyperparameter tuning. This projected vector is passed through a final ReLU activation function to obtain the ultimate image feature vector, I_t, corresponding to entry t.

As each contract corresponds to a ticker, and tickers may generally correspond to differing interactions between contract-specific features when predicting the following day's price change, we create ticker-specific embeddings. Each unique ticker (for example, AAPL) receives a particular index. This index is then passed through a trainable embedding layer which projects to dimensionality d_{ticker}, resulting in a ticker feature vector T_t.

As the scalar values are either boolean or real numbers, each scalar value corresponds to a single dimension in the final scalar feature vector. As there are 14 scalars, the scalar vector at entry t, S_t, is of dimensionality 14.

Our final feature vector X_t is a concatenation of the post-CNN image feature vector, the ticker feature vector, and the scalar feature vector: $X_t = [I_t; T_t; S_t]$ of dimensionality $d_x = d_{\text{img}} + d_{\text{ticker}} + 14$.

3.5.2 Linear Projections

In order to predict the next-day percent asset change through modeling interactions between the image, ticker-specific, and contract-specific features, we follow these steps sequentially:

1. Pass X_t through a fully-connected linear projection, projecting from d_x to d_h
2. Pass through a layernorm to stabilize
3. Pass through a ReLU activation function
4. Apply a dropout (ascertained during hyperparameter testing)
5. Pass the resulting feature vector through a fully-connected linear projection, projecting from d_h to 1

The resulting single output value represents the predicted asset price percent change.

3.5.3 Training

All training runs use the Adam optimizer. During training, the subset of data selected for the training set is split into batches. For each entry within each batch, the model makes a prediction regarding the next-day percent price change. Then, the batch-wise average absolute difference between the predicted and the true label is calculated, and this loss is backpropagated to the model's parameters. This training setup is run across the entire training set for several epochs, with validation set MAE calculated at the end of each epoch.

3.5.4 Hyperparameter Selection

In order to select hyperparameters, we follow the advice of [3] and perform a random hyperparameter search. During each iteration of the search, variables shown in table 3 are randomly assigned values from their corresponding sample sets, and the highest performing set combination is maintained for future testing and experimentation. The number of epochs is set to 10, with each combination's final performance measured according to its highest validation set performance on any epoch. Twelve iterations are run before the random search is terminated. The resulting chosen values are shown in table 4

Hyperparameter	**Sample Set**
Learning Rate	{0.00001, 0.00005, 0.0001, 0.0005}
Dropout	{0.05, 0.1, 0.15, 0.2}
Batch Size	{64, 128, 256, 512}
d_h	{32, 64, 128}
d_{img}	{64, 128, 256}
d_{ticker}	{32, 64, 128}

Table 3: Hyperparameter search space and corresponding sample sets utilized during the random search.

Hyperparameter	**Selected Value**
Learning Rate	0.00001
Dropout	0.2
Batch Size	256
d_h	128
d_{img}	64
d_{ticker}	128
Epochs	10

Table 4: Final hyperparameter configuration selected via random search.

4 Experiments

4.1 Baseline Model Results

With the hyperparameter settings shown in Table 4, the baseline model achieves a mean average absolute error of $\approx .129$ on the validation set and a mean average absolute error of $\approx .130$ on the training set. This is our baseline performance against which to compare future experiments.

4.2 Deep Prediction Head

Our first experiment involves modification to the linear projection layers which sit atop the constructed feature vector X_t. We hypothesize that the complexity of the relationships between the different input features is better modeled by a deeper set of linear projections, as increased feedforward neural model depth drastically increases the complexity of potentially predictive feature interactions [6]. Because financial data is notoriously complex, we predict that increasing the depth of the neural network head will increase the performance of the neural model. We move from the baseline's two-layer head to a six-layer head, integrating residual connections to as to avoid the vanishing gradients typically seen with deep neural networks. We choose the skip connection $y_2 = x \oplus g(f(x))$, where functions g and f represent a single linear projection and following normalizations and activations. We construct a residual block template as the following set of operations:

1. Pass input x through a linear projection from d_h to d_h
2. Apply a layernorm
3. Apply a ReLU activation function
4. Apply a dropout to avoid overfitting during training, producing $g(x)$
5. Pass $g(x)$ through a second linear projection from d_h to d_h
6. Apply a ReLU activation function, producing $f(g(x))$
7. Element-wise add $y = x \oplus f(g(x))$ with a skip connection

We integrate two such residual blocks into our prediction head by first projecting X_t from d_x to d_h. The resulting vector is then passed through two sequential residual blocks, and then projected from d_h to 1, which is our final predicted price percent change. We train our deep neural network for ten epochs using the same hyperparameter and training setup as the baseline, selecting the best performing epoch on the validation set. This is epoch 6, and corresponds to a mean average absolute error of ≈ 0.131 on the validation set, marginally larger than the baseline's mean average absolute error of ≈ 0.129 by epoch ten. Interesting to note, by epoch ten, the deep neural network's mean average absolute error on the training set dropped to $\approx .125$, contrasted with the baseline's epoch ten training set MAE of $\approx .130$. This points to the deep neural model, with its increased parameter count from the

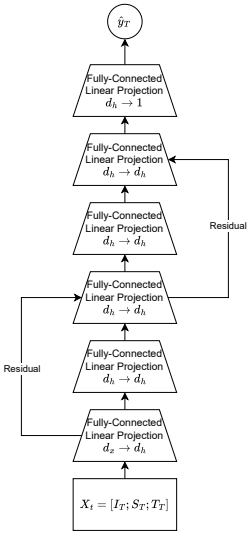

Figure 2: Deep Prediction Head Design

deep prediction head, being more susceptible to overfitting to the training set.

4.3 Causal Forecasting via Historical Self-Attention

While our prior experiment and the baseline neural model look at one contract's scalar values at one time step in isolation from the rest of the contract's history, this approach is limited by the fact that each contract may have particular behaviors throughout its evolution to expiry. We draw intuition from large language models' use of causal self-attention to formulate a contract's history as comparable to a string of tokens, with daily fluctuations in the contract comparable to different tokens throughout an utterance. The contract's entry at time T, when predicting the next-day price percent change, may therefore benefit from contextualization from prior entries corresponding to that contract from prior days.

To model this intuition, we implement a time-series self-attention mechanism (see figure 3) which takes as context linear projections of feature vectors from entries $[T-25, ..., T]$ when predicting the price change occurring at time $T+1$, or the day after the final entry. Our context window is set to 25 as this is the maximum context window size which our machines allow without significant runtime performance detriments. The entries within the context window are element-wise summed with positional embeddings corresponding to each entry's position in the context window to allow the model to incorporate relative importance of each contract based on each entry's position in the historical context. Additionally, each entry's feature vector is altered from the baseline neural model such that it includes the next-day percent price change, with the exception of the final entry which has a zero value in place of its label.

We pass the context window through two transformer encoder blocks, each with 4 transformer heads, and then extract the resulting contextualized vector of entry T. This contextualized feature vector is then passed through the two final linear projection layers, consistent with the baseline model's prediction head.

During training, rather than selecting one window for each contract, we generate a sliding window of length 25 over each entry in the contract's history. To increase the number of training samples provided to the model, we allow for the number of 'true' entries in the window to be between 1 and 25 inclusive, with a padding mask over nonexistent entries in the context window provided to the model.

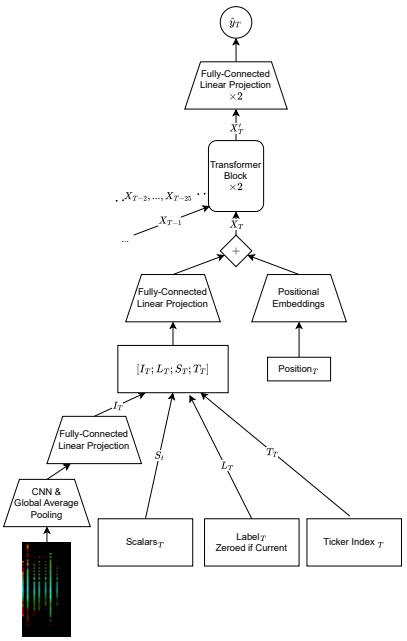

Figure 3: Time-Series Self-Attention Design

Due to machine performance bottlenecks, the batch size is reduced from the baseline's 256 to 16. All other hyperparameters are kept consistent with the baseline neural model's setup. The highest

performing epoch of the time-series self-attention model is epoch 8, with a validation set mean average absolute error of ≈ 0.127, marginally smaller than the baseline neural model's mean average absolute error of ≈ 0.129, and a training set mean average absolute error of ≈ 0.129. This indicates that contextualizing a contract's current metrics in its historical behavior increases model capacity to predict future price changes, and even marginal increases in model predictive capability can be significant in the environment of financial data.

4.4 Utilization of Social Media Text as Predictor

While our experimentation has thus far relied strictly on financial data to inform predictions on next day price change, these models are limited in their ability to incorporate the impacts of macroeconomic trends on day-to-day pricing behavior. To resolve this limitation, we incorporate text data pulled directly from the BlueSky social media platform's API into the Causal Forecasting Model.

A unique set of date, ticker pairs is first constructed based on all training set and validation set contracts' corresponding tickers and days. Then, for each unique date, ticker pair, we query the BlueSky API for five social media posts provided on the given date, containing the ticker as a substring within the post. These posts are then passed through the pre-trained RoBERTa base model encoder [8], with 12 layers and a hidden size of 768, along with a CLS token to capture the post's global context. We then mean-pool the resulting CLS tokens' embeddings across the five randomly sampled posts, once for each unique ticker, date pair. These ticker, date, mean embedding sets are generated and saved prior to model training. Importantly, we observe that we do not fine-tune the RoBERTa encoder.

The new model design incorporates these text embeddings by, for each contract entry, pulling the corresponding ticket-day pair mean embedding, filling with zeroes if none exist for the entry's date and ticker. This embedding is then passed through a single dense linear projection layer, downprojecting from dimensionality 768 to 64. The resulting features R_T are then concatenated with the model's primary feature vector $[I_T; L_T; S_t; T_T; R_T]$, which is then, as in the Causal Forecasting Model, passed along to the transformer block. Other than the incorporation of the text data, this model's design is identical to the Causal Forecasting Model's design. All hyperparameters are kept consistent with the Causal Forecasting Model's hyperparameters.

Results indicate no meaningful performance improvements over the Causal Forecasting Model; The model's validation set predictions achieve a mean average absolute error of ≈ 0.127 by epoch ten, with a training set mean average absolute error of ≈ 0.128. These results do not meaningfully differ from the Causal Forecasting Model's results. This indicates a lack of evidence on improved predictive capability from the described incorporation of social media text data.

4.5 Overview of Primary Results

As seen in figure 4, the two causal-forecasting self attention models performed the best, while baseline outperformed the deep prediction head model. All neural models outperformed the naïve baseline, which predicts a 0 percent value change.

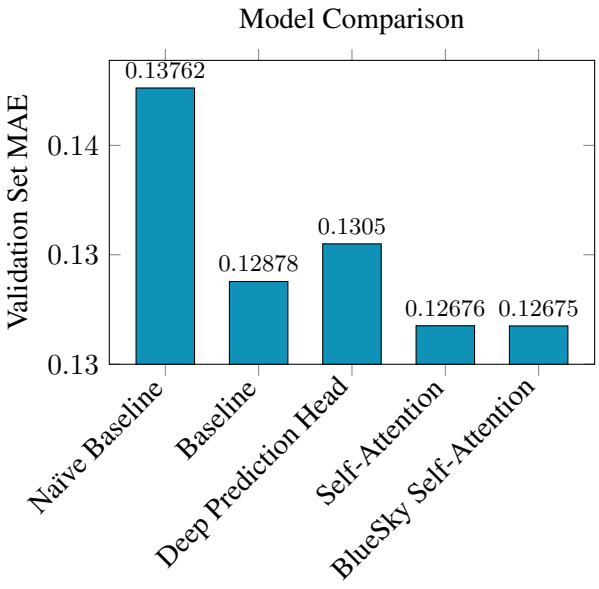

Figure 4: Mean Absolute Error (MAE) for various model architectures on the validation set.

4.6 Discussion of Experimental Trends

Regarding the observation that all neural models outperformed the naïve baseline which predicts a 0 percent value change, we may conclude that valuable insights are provided by contract-specific parameters. Additionally, all neural models utilized the surface edge image as input, which substitutes for computationally expensive denoising time series methods. The fact that all neural models outperformed the naïve baseline suggests that the surface edge image may be a computationally inex-

pensive method for time-series denoising methods, although future work ought to directly compare our models' results against better developed baselines such as the Black Scholes Model.

The failure of the deep prediction head model to outperform the baseline neural model demonstrates a vulnerability to overfitting. The deep prediction head's depth drastically increases the per-contract parameter count and allows the model to incorporate more complex feature interaction; the power of the increased parameter count is demonstrated by the deep head model's status as representing the lowest *training* mean average absolute error across all tested model configurations. However, given the relatively early convergence (after six epochs) between the training and development sets' loss and the poor development set performance of the deep head model, we hypothesize that this model's increased parameter count led to overfitting to the training set.

The causal forecasting self-attention's model's status as the highest performing configuration represents the power of incorporating contracts' past history into a next-day-price-change prediction. If a particular contract's corresponding price changes have historically undergone significant volatility, then this increased volatility may add predictive power to the neural model when predicting the next-day-price-change. Although the self-attention models are more computationally expensive than single-contract models, they provide a relatively cheap substitute for long-range LSTMs which incorporate years of context.

The failure of the text-incorporated BlueSky self-attention model to outperform the text-ablated forecasting self-attention model may suggest either a lack of utility of text data when building options models, or issues with our implementation. The text-incorporated model only averaged five relevant social media posts for each ticker, day pair; these posts were randomly chosen, without filters for the reputation of the source (the OP) or the relevance of the post beyond containing the ticker symbol as a substring. Additionally, five posts is a relatively small sample size, and data could not be pulled from dates prior to 2023, as BlueSky did not yet exist prior to February of that year. These factors contribute to potentially unreliable training data. Lastly, due to limitations on our computational resources, the RoBERTa encoder which generates text embeddings for each post was not fine-tuned. While RoBERTa is a general tool to extract meaningful insights from text data, it is possible that its general-use status hinders its ability to extract meaningful insights for our use case.

4.7 Questions Unanswered, Future Directions

Although we demonstrate that our neural models consistently outperform a naïve baseline, we have not compared our models' results against industry-standard options market predictors or current leading literature. Our capacity to outperform industry-grade models is therefore an underdeveloped research question which will necessitate further testing and likely further development of more complex and computationally expensive neural models.

Outside of model comparison, the most prominent flaw in our work are the extreme limitations of our computing resources. The biggest setback of which, was using just 4 out of the 104 tickers in our final filtered dataset, or roughly 5% of available contracts. Although we incorporate social media data into our time-series self-attention model, computational resources limited our ability to use social media data and sentiment signals to their fullest potential. Future work ought to weight particular posts by the authors' reputability, potentially through the use of an attention mechanism. More training time and larger computational resources will allow us make use of our full 30.5 million contract dataset, as well as to fine-tune the RoBERTa encoder responsible for extracting use-case-specific signals from the provided text data. We ought to also investigate textual resources beyond BlueSky, such as mainstream news and other social media sources.

5 Conclusions

5.1 Main Takeaway

SurfaceEdge demonstrates that treating the daily options surface as a structured visual input provides meaningful predictive signal for next-day contract price movements. By encoding implied volatility, open interest, and volume into a convolutional architecture, we show that cross-contract spatial relationships can be captured to some degree and can be included with traditional scalar-only inputs. The addition of a causal self-attention mechanism over sequential contract history further improves predictive performance, achieving an 8% improvement over the naïve baseline (MAE 0.127 vs. 0.138). While the improvement is minimal, 8% is statistically significant over 347,000 validation contracts,

implying SurfaceEdge's potential upon further expansion of this work. Critically, these results are obtained under a strict chronological train/validation split, addressing a pervasive methodological flaw in existing financial deep learning literature. While our experiments are limited to four large-cap U.S. equities due to limited computing resources, our preliminary models for SurfaceEdge establish a promising foundation for surface-aware derivatives pricing models.

5.2 Ethical Considerations

The ethical considerations of SurfaceEdge, and similar efforts to price financial equities using deep learning, primarily revolve around accessibility and market fairness. Sophisticated pricing tools have historically been the exclusive domain of institutional investment firms with billions of dollars of resources dedicated to finding small statistical edges in the market. By providing the SurfaceEdge architecture as a publicly accessible deep learning framework, we contribute toward democratizing access to more informed trading decisions, lowering the barrier to entry for retail investors who have traditionally been at a significant informational disadvantage relative to professional market participants. However, this democratization introduces a paradox: any statistical edge in financial markets exists precisely because it exploits underutilized data or mispricings that have not yet been reflected by the collective opinion of market participants. Once a model capable of identifying such an edge is released publicly, widespread adoption degrades the very inefficiency the model was trained to exploit, rendering the edge obsolete. This self-defeating property is the very reason institutional traders are so protective of their proprietary trading strategies.

References

[1] Kazi Alam, Md.H̃asan Bhuiyan, Iftekharul Ul Haque, Md.F̃arhan Monir, and Tanvir Ahmed. 2024. Enhancing stock market prediction: A robust LSTM-DNN model analysis on 26 real-life datasets. *IEEE Access*, 12:122757–122768.

[2] Wei Bao, Jun Yue, and Yulei Rao. 2017. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7).

[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

[4] Luyang Chen, Markus Pelger, and Jason Zhu. 2024. Deep learning in asset pricing. *Management Science*, 70(2):714–750.

[5] Philipp Dubach. 2024. options-data: Historical U.S. equity options chain data. `https://github.com/philippdubach/options-data`. Accessed: March 30, 2026.

[6] Ronen Eldan and Ohad Shamir. 2015. The power of depth for feedforward neural networks. *Preprint*, arXiv:1512.03965.

[7] James M. Hutchinson, Andrew W. Lo, and Tomaso Poggio. 1994. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[9] Kingsley Olorunnimbe and Herna Viktor. 2024. Ensemble of temporal transformers for financial time series. *Journal of Intelligent Information Systems*, 62:1087–1111.

[10] Erfan Radfar. 2025. Stock market trend prediction using deep neural network via chart analysis: a practical method or a myth? *Humanities and Social Sciences Communications*, 12:662.

[11] Johannes Ruf and Weiguan Wang. 2020. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance*, 24(1):1–46.

[12] Mikhail Tuchin and 1 others. 2022. Noise immunity and robustness study of image recognition using a convolutional neural network. *Symmetry*, 14(2).

[13] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35.