Recovering Predictability from the Implied Volatility Surface: A Vision Transformer Approach*

October 17, 2025

Abstract

Placeholder

Keywords: key1, key2, key3

JEL Codes: key1, key2, key3

*abc

1 Introduction

2 Literature Review

3 Data

3.1 Option-Implied Volatility Surface (IVS)

We obtain the equity and index option data from the IvyDB OptionMetrics database, which provides Volatility Surface files (vsurfdYYYY) that contain the interpolated Black-Scholes implied volatilities, starting from January 1996. For each security on each day, a volatility surface with standard maturities of τ days to expiration and moneyness levels measured by option δ is provided. [1]

We select out-of-the-money (OTM) and near at-the-money (ATM) put options with delta levels in [-0.5, -0.1], as well as OTM and near ATM call options with delta levels in [0.1, 0.5], since the deep-in-the-money call options are often illiquid, following Martin (2017), among others. The options are rearranged by their moneyness (implied strike), so the delta levels start from -0.1 to -0.5, then from 0.5 to 0.1 [2]. We also drop the options with maturities equal to 10 days, since the large fraction of missing values. The final volatility surface data contains 10 maturities and 18 delta levels, and a example of the volatility surface is shown in Figure 2. The input features, in result, can be represented as $IV_{i,t} = \{IV_{i,t}(\tau, \delta)\}_{\tau \in T, \delta \in \Delta}$. the volatility surface of security i at time t.

We obtain the daily return data from CRSP for all firms listed on NYSE, AMEX, and NASDAQ, as well as the S&P 500 index. For each security i at time t, we calculate the

[1] OptionMetrics use a methodology based on a kernel smoothing algorithm to interpolate the volatility surface. The maturities are 10, 30, 60, 91, 122, 152, 182, 365, 547, 730 days to expiration, and the option delta levels are from 0.1 to 0.9 with a step of 0.05, positive for call options and negative for put options.

[2] The implied strike order between put option with delta -0.5 and call option with delta 0.5 is uncertain, but their spread is proved to have predictive power for future returns (Yan, 2011), so we keep both of them.

cumulative return on security i from day t to day $t + H$ as:

$$r_{i,t}^{H} = \prod_{j=0}^{H-1} (1 + r_{i,t+j}) - 1 \tag{1}$$

We link the volatility surface data from OptionMetrics with the return data from CRSP using a linking table provided by WRDS. The final dataset spans from January 1996 to August 2023, and Figure 1 shows the coverage of the stocks with available volatility surface data over time. The number of stocks with available volatility surface data increases significantly over time, from below 1000 to over 5000 in recent years.

3.2 Option Characteristics

Motivated by the extensive literature from cross-section studies of stock returns, we also collect a set of option-related characteristics that have been shown to have predictive power for future stock returns. Neuhierl et al. (2022) summarized a comprehensive list of option characteristics and compared their performance. We follow their results and select the strongest predictors, which are described in the Appendix B.

4 The Vision Transformer Model

In this section, we briefly introduce the vision transformer (ViT) model architecture and our experimental design.

4.1 Brief Introduction of the Vision Transformer Model

First introduced by Vaswani et al. (2017), "Attention Is All You Need," the Transformer model has become a foundational architecture in deep learning, particularly for natural language processing (NLP). Unlike its predecessors, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, the Transformer does not rely on

sequential data processing. Instead, it processes the entire input sequence at once, using a sophisticated mechanism known as self-attention to weigh the importance of different words in the sequence.

The core of the Transformer is its encoder-decoder structure. The encoder's role is to map an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $z = (z_1, ..., z_n)$. The decoder then takes z and generates an output sequence $(y_1, ..., y_m)$, one symbol at a time.

Building on the success of the Transformer in NLP, the Vision Transformer (ViT) was proposed by Dosovitskiy et al. (2020) to apply the same architecture to computer vision tasks, offering a compelling alternative to the widely used Convolutional Neural Networks (CNNs). The fundamental idea behind ViT is to treat an image as a sequence of patches, analogous to how a sentence is treated as a sequence of words.

By converting images into a sequential format, ViT demonstrates that the reliance on convolutions is not a necessity for vision tasks and that a general-purpose attention-based architecture can achieve state-of-the-art (SOTA) performance, especially when pre-trained on large datasets.

The architecture of the ViT, shown in Figure 4, adapts the original Transformer in the following way:

1. **Image Patching and Embedding**: The ViT model first reshapes the input image from a 2D grid of pixels into a sequence of flattened 2D patches. For a image $x \in \mathbb{R}^{H \times W \times C}$, it is divided into N patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) is the height and width of the original image, C is the number of channels (e.g., RGB), and (P, P) is the resolution of each patch, and $N = (H \times W)/(P^2)$ is the number of patches. These patches are then flattened and mapped to a latent D-dimensional embedding space through a trainable linear projection. Each patch is taken as a token, similar to words in NLP.

2. **Learnable Class Token**: Inspired by the [CLS] token used in BERT (Bidirectional

Encoder Representations from Transformers) model (Devlin et al., 2019), a learnable embedding is prepended to the sequence of patch embeddings. The state of this token at the output of the Transformer encoder serves as the aggregate image representation for classification tasks.

3. **Positional Embeddings**: Similar to the positional encodings in the original Transformer, the ViT adds learnable 1D positional embeddings to the patch embeddings to retain spatial information. These embeddings allow the model to learn the relative positions of the image patches.

   The flattened patches, along with the class token and positional embeddings, are served as input to the Transformer encoder:

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1 E; \mathbf{x}_p^2 E; ...; \mathbf{x}_p^N E] + E_{pos} \tag{2}$$

   where $E \in \mathbb{R}^{(P^2 \cdot C) \times D}$ is the patch embedding projection matrix, $E_{pos} \in \mathbb{R}^{(N+1) \times D}$ is the positional embedding matrix, and $\mathbf{z}_0 \in \mathbb{R}^{(N+1) \times D}$ is the resulting sequence of embedded patches, with length $N + 1$ (including the class token) and embedding dimension D.

4. **Transformer Encoder**: The resulting sequence of embedded patches (including the class token) is then fed directly into a standard Transformer encoder (Vaswani et al., 2017). This encoder is composed of alternating layers of multi-head self-attention (MSA) and multi-layer perceptron (MLP) blocks, similar to the original Transformer encoder, but with some slight modifications:

   - **Multi-Head Self-Attention (MSA)**: This mechanism allows the model to focus on different parts of the input sequence simultaneously, capturing various aspects of the image. Each attention head computes a weighted sum of the input embeddings, where the weights are determined by the similarity between the query and key vectors. These similarity scores are calculated using Query (Q), Key (K), and

Value (V) matrices, which are linear projections of the input embeddings. The attention scores are computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

where d_k is the dimension of the key vectors, used for scaling. The multi-head attention mechanism allows the model to jointly attend to information from different representation subspaces at different positions. For layer $l = 1, ..., L$, the MSA block can be expressed as:

$$\mathbf{z}'_l = \text{MSA}(LN(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \tag{4}$$

where $LN(\cdot)$ denotes layer normalization, and $\mathbf{z}'_l \in \mathbb{R}^{(N+1)\times D}$ is the output of the MSA block at layer l.

- **Multi-Layer Perceptron (MLP)**: Following the MSA, a position-wise feed-forward network (FFN) is applied to each position independently and identically. This FFN typically consists of two linear layers with a Gaussian Error Linear Unit (GELU) activation in between, unlike the Rectified Linear Unit (ReLU) used in the original Transformer:

$$\text{MLP}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \tag{5}$$

where W_1, W_2 are weight matrices and b_1, b_2 are bias vectors. Besides, dropout is applied after each fully connected layer to prevent overfitting. For layer $l = 1, ..., L$, the MLP block can be expressed as:

$$\mathbf{z}_l = \text{MLP}(LN(\mathbf{z}'_l)) + \mathbf{z}'_l \tag{6}$$

where $\mathbf{z}_l \in \mathbb{R}^{(N+1)\times D}$ is the output of the MLP block at layer l.

- **Layer Normalization**: In contrast to the post-normalization used in the original Transformer, the encoder blocks in ViT implement pre-normalization, where layer normalization is applied right before the MSA and MLP blocks, which has been shown to lead to mre effective training for deeper networks (Xiong et al., 2020) without the need for learning rate warm-up.

5. **Classification Head**: For image classification, only the output vector corresponding to the prepended class token in the final layer ($\mathbf{z}_L^0$) is used, which serves as the aggregate representation of the entire image:

$$\mathbf{v} = LN(\mathbf{z}_L^0) \tag{7}$$

This vector $\mathbf{v} \in \mathbb{R}^D$, which serves as an encoded vector representation of the input image, is then passed through a final classification head, typically a single linear layer followed by a softmax function, to produce the class probabilities:

$$\mathbf{y} = \text{softmax}(\mathbf{v}W_c + b_c) \tag{8}$$

where $W_c \in \mathbb{R}^{D\times K}$ and $b_c \in \mathbb{R}^K$ are the weights and bias of the classification head, and K is the number of classes.

4.2 Image Representation of Option-Implied Volatility Surface

In the computer vision (CV) field, a image is typically represented as a three-dimensional matrix with (C, H, W) dimensions, where C is the number of channels (e.g., $C = 3$ for RGB images, and $C = 1$ for grayscale images), H is the height (number of pixels in vertical direction), and W is the width (number of pixels in horizontal direction). Each pixel in the image has a value ranging from 0 to 255, representing the intensity of the color channel(s)

at that pixel.

An option-implied volatility surface is a three-dimensional plot that displays the implied volatility of a security options across different strike prices and expirations (Figure 2). As shown in Figure 3, the volatility surface can also be represented as a two-dimensional matrix, which is anlogous to a single-channel grayscale image. The height H of the matrix corresponds to the number of maturities N_τ, and the width W corresponds to the number of moneyness levels N_δ. In our dataset (described in Section 3), the volatility surface contains 10 maturities and 18 moneyness levels, so we can reshape the volatility surface into a "image-like" matrix with dimensions $(C, H, W) = (1, 10, 18)$. The pixel values in the matrix are the corresponding implied volatility levels, which are standardized to have zero mean and unit variance before being fed into the ViT model, just like standard image preprocessing in the computer vision field.

Besides the implied volatility surface of individual stocks, we also consider the implied volatility surface of the S&P 500 index options as an additional input. The index options contain essential information of market-wide risk and risk premia, in time series as well as in cross-section, providing incremental predictive power for the stock returns (Andersen et al., 2015; Ang et al., 2006). The index volatility surface, in hence, can serve as an additional channel for the input volatility surface "image". However, unlike standard RGB images, where the three channels (Red, Green, Blue) contain correlated and complementary information about the same object, the two implied volatility surfaces in our case contain semantically distinct and independent information. This poses a challenge for the standard ViT model, which is designed to process images with correlated channels. Inspired by Bao et al. (2024), we use the so-called Channel ViT model for the embedding of the multiple volatility surfaces.

Specifically, the Channel ViT model, shown in Figure 5, processes each input channel (volatility surface) independently in the embedding stage. Each channel (volatility surface) is split into patches, linearly embedded, and employs a set of learnable channel embeddings

to encode the channel-specific information besides the positional embeddings. The resulting sequence of vectors from all channels is then concatenated and fed into a standard Transformer encoder, similar to the original ViT model. Since the two volatility surfaces have the exactly the same structure (same maturities and moneyness levels), they share a common positional encoding matrix.

The ViT model, except for the final classification head, can be viewed as a non-linear mapping $ViT(\cdot|\theta)$ from the input image (volatility surface) to a dense vector representation of the image:

$$\mathbf{v}_{i,t} = \mathbf{ViT}(IV_t|\theta) \tag{9}$$

Where $IV_t = IV_{i,t}$ or $IV_t = (IV_{i,t}, IV_{spx,t})$ depending on whether we use single or multiple volatility surfaces as input. Here, $IV_{i,t}$ is the volatility surface of stock i at time t, and $IV_{spx,t}$ is the volatility surface of the S&P 500 index at time t. $\mathbf{v}_{i,t} \in \mathbb{R}^D$ is vector representation of the information contained in the volatility surfaces, and θ is the set of parameters of the ViT model, excluding the final classification head.

4.3 Discretization of Return

Utilizing the dense vector representation $\mathbf{v}_{i,t}$ obtained from the ViT model, a classification head is attached to perform classification tasks. In our study, such a classification head is a single linear layer followed by a softmax function [3], which maps the vector representation

[3] In the original ViT paper, Dosovitskiy et al. (2020) used a multi-layer perceptron (MLP) with one hidden layer at pre-training time and used a single linear layer at fine-tuning time. A recent update, Beyer et al. (2022) from some of the same authors of the original paper suggests that a simple linear layer at the end is not significantly worse than an MLP. We use a single linear layer for simplicity reason.

$\mathbf{v}_{i,t}$ to a probability distribution over the classes:

$$\mathbf{p}_{i,t} = \text{softmax}(W_c \cdot \mathbf{v}_{i,t} + b_c) \tag{10}$$

$$\text{softmax}(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad j = 1, ..., K \tag{11}$$

where $W_c \in \mathbb{R}^{K \times D}$ and $b_c \in \mathbb{R}^K$ are the weights and bias of the classification head, and K is the number of classes. The output $\mathbf{p}_{i,t} \in \mathbb{R}^K$ is a probability vector, where each element $\mathbf{p}_{i,t}(k)$ represents the predicted probability of stock i's return falling into class k at time t.

The simplest classification is the binary classification, where the number of classes $K = 2$. For example, we can classify the stock return into "up" and "down" classes based on whether the future return is positive or negative, as in Jiang et al. (2023). Alternatively, the stock return can be classified into "safe" and "crash" classes based on whether the future return is above a certain negative value (e.g., -10%). In general, we denote $y_{i,t}$ as the binary variable indicating whether stock i's return falls into the two classes defined by a specific threshold q at time t:

$$y_{i,t} = \mathbf{I}(r_{i,t}^H > q) = \begin{cases} 1, & \text{if } r_{i,t}^H > q \\ 0, & \text{if } r_{i,t}^H \leq q \end{cases} \tag{12}$$

$$\hat{y}_{i,t} = \mathbf{p}_{i,t}(1) \tag{13}$$

where $r_{i,t}^H$ is the cumulative return of stock i from day t to day $t + H$, and the output $\hat{y}_{i,t} = \mathbf{p}_{i,t}(1)$ represents the predicted probability of stock i's return being in class 1 at time t. The threshold q can be set to 0 for the "up/down" classification. [4]

For multi-class classification ($K > 2$), the stock returns can be discretized into multiple classes based on quantiles or specific thresholds. Assume $\{q_0, q_1, ..., q_K\}$ are the thresholds that define the K classes, where $q_0 = -\infty$ and $q_K = +\infty$. In general, we denote $y_{i,t}$ as the

[4] In the special case of binary classification ($K = 2$), the softmax function reduces to the logistic (sigmoid) function.

categorical variable indicating which class stock i's return falls into at time t:

$$y_{i,t} = k, \quad \text{if } r_{i,t}^{H} \in (q_{k-1}, q_k], \quad k = 1, 2, ..., K \tag{14}$$

$$\hat{y}_{i,t}^{k} = \mathbf{p}_{i,t}(k) \tag{15}$$

4.4 Training Process

We use standard cross-entropy loss function for the classification task. For binary classification, where $y \in \{0, 1\}$, the loss function can be expressed as:

$$\mathcal{L}(y, \hat{y}) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right] \tag{16}$$

For multi-class classification, where $y \in \{1, 2, ..., K\}$, the loss function can be expressed as:

$$\mathcal{L}(y, \hat{y}) = -\sum_{k=1}^{K} \mathbf{I}(y = k) \log(\hat{y}^k) \tag{17}$$

where $\hat{y}^k$ is the predicted probability of class k.

In the case of imbalanced classes, we can use a weighted cross-entropy loss function to give more importance to the minority classes. The weighted loss function can be expressed as:

$$\mathcal{L}(y, \hat{y}) = -\sum_{k=1}^{K} w_k \cdot \mathbf{I}(y = k) \log(\hat{y}^k) \tag{18}$$

$$w_k = \frac{N}{K \cdot N_k} \tag{19}$$

where w_k is the weight for class k, N is the total number of samples, and N_k is the number of samples in class k. This weighting scheme helps to balance the influence of each class during training, especially when some classes are underrepresented.

We adopt similar regularization techniques as in Gu et al. (2020), to prevent overfitting and improve the comuputational efficiency. We use a Adaptive Moment Estimation With Weight Decay (AdamW) optimizer (Loshchilov and Hutter, 2019), which is standard in the transformer training and a refined version of the Adam optimizer (Kingma and Ba, 2017). AdamW decouples the weight decay (L2 regularization) from the gradient update, which has been shown to lead to better generalization performance. In this study, we set the learning rate to 1×10^{-5}, the weight decay to 1×10^{-2} and the batch size to 512. We use a dropout rate of 0.2 to prevent overfitting. Dropout is a regularization technique that randomly sets a fraction of the input units to zero during training, which helps prevent overfitting by reducing the model's reliance on specific features.

We apply multiple methods for parameter initialization: for weights in the patch embedding layers and classfication head, we use the Xavier initialization (Glorot and Bengio, 2010), which is the best practice for deep neural networks such as CNNs; for parameters in the CLS token and positional embeddings, as well as the weights in the Transformer encoder, we use a truncated normal distribution with a standard deviation of 0.02, following the BERT model Devlin et al. (2019); for layer normalization parameters, we initialize the weights to 1; the biases in all layers are initialized to 0. We use a standard "warm-up" and "cosine decay" learning rate scheduler, which gradually increases the learning rate from zero to the initial value over a specified number of warm-up steps, and then decays it using a cosine function. This approach helps stabilize training in the early stages and allows for better convergence. Early stopping is also employed, where the training process is halted if the validation loss does not improve for a two consecutive epochs, to prevent overfitting and save computational resources.

We train each model using annually updated rolling window, given the time-varying universe of stocks in 1. Specifically, we use a 8-year in-sample period, with 6 years for training and 2 years for validation, to predict the out-of-sample returns for the subsequent year. The first training period starts from January 1996 to December 2003, and the trained

model is then used to predict out-of-sample returns for the subsequent year 2004. The model is then updated with the data from 2004, and used to predict out-of-sample returns for 2005. This process continues until August 2023, the last month of our dataset. Overall, the out-of-sample prediction period is from January 2004 to August 2023, which contains around 20 years of results. The daily dataset provides a large number of training samples, which is crucial for training such a large deep learning model, while we only collect the out-of-sample predictions at the end of each month for empirical analysis. Since the stochastic nature of the Vision Transformer model, we train each specific model for five times and calculate the average predictions, following Gu et al. (2020).

5 Empirical Results

We begin with the simplest binary classification problem ($K = 2$). Following Jiang et al. (2023), the stock returns are simply classified into "up" and "down" classes based on whether the future return is positive or negative. Besides the simplicity and interpretability, one remarkable benefit of such binary classification is that the two classes are naturally more balanced in our dataset, which is to say, there are approximately 50% "up" labels and 50% "down" labels. This is crucial for the training of such deep learning or machine learning models, since imbalanced classes can lead to suboptimal performance, as the model may become biased towards the majority class (He and Garcia, 2009).

We focus on the prediction of 1-month ahead stock returns ($H = 21$ trading days), which is common in the literature of cross-section of stock returns studies. Table 1 reports the portfolio performance of equal-weight and value-weight decile portfolios, including a long-short spread portfolio "H-L", formed on out-of-sample predicted "up" probability $\hat{P}(r_{i,t}^{21} > 0)$ in each month's end, with a holding period of one month. The table

6 Conclusion

References

An, B.-J., A. Ang, T. G. Bali, and N. Cakici, 2014, "The Joint Cross Section of Stocks and Options," *The Journal of Finance*, 69 (5), 2279–2337.

Andersen, T. G., N. Fusari, and V. Todorov, 2015, "The Risk Premia Embedded in Index Options," *Journal of Financial Economics*, 117 (3), 558–584, September.

Ang, A., R. J. Hodrick, Y. Xing, and X. Zhang, 2006, "The Cross-Section of Volatility and Expected Returns," *The Journal of Finance*, 61 (1), 259–299.

Bao, Y., S. Sivanandan, and T. Karaletsos, 2024, "Channel Vision Transformers: An Image Is Worth 1 x 16 x 16 Words," April.

Beyer, L., X. Zhai, and A. Kolesnikov, 2022, "Better Plain ViT Baselines for ImageNet-1k," May.

Chen, A. Y. and T. Zimmermann, 2022, "Open Source Cross-Sectional Asset Pricing," *Critical Finance Review*, 11 (2), 207–264.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova, 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in J. Burstein, C. Doran, and T. Solorio eds. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics, June.

Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, and S. Gelly, 2020, "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*.

Glorot, X. and Y. Bengio, 2010, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256. JMLR Workshop and Conference Proceedings, March.

Gu, S., B. Kelly, and D. Xiu, 2020, "Empirical Asset Pricing via Machine Learning.," *Review of Financial Studies*, 33 (5), 2223–2273.

He, H. and E. A. Garcia, 2009, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, 21 (9), 1263–1284, September.

Jiang, J., B. Kelly, and D. Xiu, 2023, "(Re-)Imag(in)Ing Price Trends," *The Journal of Finance*, 78 (6), 3193–3249.

Johnson, T. L. and E. C. So, 2012, "The Option to Stock Volume Ratio and Future Returns," *Journal of Financial Economics*, 106 (2), 262–286, November.

Kingma, D. P. and J. Ba, 2017, "Adam: A Method for Stochastic Optimization," January.

Loshchilov, I. and F. Hutter, 2019, "Decoupled Weight Decay Regularization," January.

Martin, I., 2017, "What Is the Expected Return on the Market?*," *The Quarterly Journal of Economics*, 132 (1), 367–433, February.

Neuhierl, A., X. Tang, R. T. Varneskov, and G. Zhou, 2022, "Option Characteristics as Cross-Sectional Predictors."

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, 2017, "Attention Is All You Need," *Advances in neural information processing systems*, 30.

Xing, Y., X. Zhang, and R. Zhao, 2010, "What Does the Individual Option Volatility Smirk Tell Us About Future Equity Returns?" *Journal of Financial and Quantitative Analysis*, 45 (3), 641–662, June.

Xiong, R., Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu, 2020, "On Layer Normalization in the Transformer Architecture," June.

Yan, S., 2011, "Jump Risk, Stock Returns, and Slope of Implied Volatility Smile," *Journal of Financial Economics*, 99 (1), 216–233, January.

Tables

Table 1: Portfolio Analysis, Binary Classification Model, All Stocks

	Decile Portfolios										
Model	1	2	3	4	5	6	7	8	9	10	10 - 1
Equal-Weighted Portfolios											
TF180	-0.66	0.36	0.68	0.88*	0.90**	0.92**	0.86**	0.85***	0.80***	0.68***	1.33***
	(-0.99)	(0.57)	(1.24)	(1.80)	(2.10)	(2.36)	(2.39)	(2.69)	(2.89)	(2.97)	(2.70)
TF360	-0.51	0.12	0.55	0.71	0.84**	0.81**	0.91**	0.95***	0.99***	0.90**	1.41***
	(-0.80)	(0.21)	(1.08)	(1.54)	(2.08)	(2.13)	(2.55)	(2.74)	(2.71)	(1.96)	(2.83)
ViT180	-0.73	0.15	0.66	0.85*	0.88**	0.88**	0.84**	0.81***	0.74***	0.67***	1.39**
	(-0.98)	(0.25)	(1.25)	(1.89)	(2.19)	(2.40)	(2.47)	(2.63)	(2.81)	(3.12)	(2.37)
ViT360	-0.65	-0.08	0.35	0.73*	0.93**	0.99**	1.07***	1.01**	0.77**	0.65*	1.30***
	(-1.23)	(-0.17)	(0.76)	(1.70)	(2.27)	(2.49)	(2.60)	(2.51)	(2.04)	(1.82)	(3.45)
Value-Weighted Portfolios											
TF180	-0.50	0.32	0.66	0.87*	0.96**	0.80**	0.92***	0.98***	0.84***	0.80***	1.30**
	(-0.84)	(0.46)	(1.12)	(1.80)	(2.16)	(2.03)	(2.62)	(3.31)	(3.35)	(4.10)	(2.54)
TF360	-0.52	0.25	0.55	0.74	0.60	0.78**	0.81**	1.09***	1.05***	1.11***	1.63***
	(-0.90)	(0.37)	(1.07)	(1.49)	(1.36)	(2.18)	(2.46)	(3.26)	(3.42)	(2.70)	(3.20)
ViT180	-0.60	0.24	0.66	0.86	0.70	0.89**	0.80**	0.88***	0.82***	0.71***	1.31**
	(-0.86)	(0.35)	(1.11)	(1.64)	(1.48)	(2.41)	(2.53)	(2.93)	(3.11)	(3.51)	(2.19)
ViT360	-0.59	-0.01	0.55	0.81*	0.94**	0.92**	1.01***	0.96**	0.83**	0.67*	1.27***
	(-1.08)	(-0.02)	(1.20)	(1.89)	(2.46)	(2.52)	(2.69)	(2.42)	(2.29)	(1.78)	(2.94)

Note: $^{***}p < 0.01$; $^{**}p < 0.05$; $^{*}p < 0.1$

Table 2: Correlation Analysis

	TF180	TF360	ViT180	ViT360
CIV	-0.504	-0.401	-0.650	-0.457
PIV	-0.539	-0.419	-0.660	-0.468
SKEW	0.099	0.078	0.162	0.078
ΔCIV	0.006	0.008	-0.043	-0.026
ΔPIV	-0.003	0.001	-0.047	-0.029
log(O/S)	-0.021	-0.027	0.029	0.014
IVSpread	-0.089	-0.074	-0.065	-0.053
IVSkew	-0.011	-0.057	-0.040	-0.087
IVol-RVol	0.154	0.125	0.161	0.129
VOV	-0.081	-0.042	-0.144	-0.118
beta	-0.199	-0.232	-0.276	-0.185
log(ME)	0.428	0.306	0.513	0.373
log(B/M)	0.018	0.022	0.023	0.022
Inv	-0.057	-0.048	-0.058	-0.040
Profit	0.099	0.065	0.147	0.109
Illiq	-0.214	-0.167	-0.291	-0.214
Ivol	-0.548	-0.432	-0.645	-0.438
$ret_{12,2}$	0.108	0.085	0.135	0.056
$ret_{1,0}$	0.064	0.051	0.099	0.061

Table 3: Portfolio Characteristics, Binary Classification Model, All Stocks

	Low	2	3	4	5	6	7	8	9	High
$\hat{P}(r_{i,t}^{21} > 0)$	0.448	0.505	0.541	0.565	0.582	0.595	0.606	0.615	0.625	0.646
CIV	1.291	0.843	0.685	0.587	0.516	0.454	0.414	0.368	0.336	0.384
PIV	1.174	0.787	0.651	0.560	0.491	0.432	0.394	0.352	0.321	0.369
SKEW	-0.088	-0.051	-0.011	0.003	-0.004	-0.006	-0.003	0.002	0.004	-0.009
ΔCIV	0.012	0.001	-0.000	-0.000	-0.001	-0.002	-0.002	-0.002	-0.002	-0.003
ΔPIV	0.010	0.002	0.000	0.000	0.000	-0.002	-0.003	-0.002	-0.002	-0.003
log(O/S)	5.388	5.554	5.725	5.747	5.685	5.605	5.584	5.617	5.640	5.384
IVSpread	0.021	0.009	0.005	0.004	0.004	0.001	-0.001	-0.001	0.001	0.002
IVSkew	0.109	0.084	0.073	0.072	0.071	0.070	0.068	0.066	0.068	0.087
IVol-RVol	-0.102	-0.061	-0.041	-0.033	-0.029	-0.025	-0.022	-0.018	-0.020	-0.027
VOV	0.217	0.179	0.151	0.145	0.144	0.144	0.142	0.141	0.143	0.175
beta	1.203	1.262	1.278	1.231	1.169	1.109	1.046	0.970	0.879	0.834
log(ME)	12.502	13.080	13.499	13.821	14.096	14.347	14.618	14.860	14.969	14.443
log(B/M)	1.068	1.117	1.121	1.136	1.221	1.270	1.264	1.296	1.205	1.252
Inv	0.182	0.193	0.200	0.183	0.167	0.154	0.136	0.127	0.115	0.114
Profit	0.013	0.049	0.072	0.085	0.088	0.086	0.080	0.072	0.058	0.042
Illiq	0.088	0.053	0.035	0.025	0.022	0.018	0.017	0.014	0.014	0.025
Ivol	0.032	0.026	0.023	0.020	0.018	0.016	0.015	0.014	0.012	0.013
$ret_{12,2}$	0.081	0.166	0.188	0.160	0.143	0.130	0.132	0.123	0.118	0.108
$ret_{1,0}$	-0.945	-0.149	0.431	0.821	0.999	1.096	1.185	1.137	1.053	0.883

Table 4: Time Series Regression, Binary Classification Model, All Stocks

	TF180	TF360	ViT180	ViT360	IVSpread	IVSkew
	(1)	(2)	(3)	(4)	(5)	(6)
Alpha	0.011***	0.015***	0.012***	0.011***	0.008***	0.001
	(0.003)	(0.004)	(0.004)	(0.003)	(0.002)	(0.002)
MKT	-0.403***	-0.475***	-0.602***	-0.277**	-0.021	0.051
	(0.095)	(0.114)	(0.062)	(0.115)	(0.052)	(0.057)
SMB	-0.924***	-0.942***	-1.129***	-0.898***	-0.024	-0.156
	(0.186)	(0.182)	(0.146)	(0.180)	(0.083)	(0.115)
HML	0.393***	0.466**	0.312**	0.291	0.186*	-0.311***
	(0.120)	(0.184)	(0.145)	(0.286)	(0.110)	(0.089)
RMW	1.224***	1.187***	1.566***	0.971***	0.378***	-0.056
	(0.187)	(0.233)	(0.204)	(0.263)	(0.117)	(0.155)
CMA	-0.029	-0.305	0.302	-0.423	0.032	-0.334**
	(0.231)	(0.338)	(0.275)	(0.386)	(0.195)	(0.153)
UMD	0.146	-0.282	0.344*	-0.116	0.137***	0.163**
	(0.228)	(0.262)	(0.185)	(0.182)	(0.040)	(0.081)
Observations	220	220	236	236	228	228
R^2	0.534	0.446	0.668	0.279	0.130	0.277
Adjusted R^2	0.521	0.431	0.659	0.260	0.106	0.257
Residual Std. Error	0.047	0.054	0.049	0.062	0.029	0.030
F Statistic	21.175***	21.231***	88.988***	8.865***	6.669***	8.560***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 5: Fama-MacBeth Regression, Binary Classification Model, All Stocks

	TF180		TF360		ViT180		ViT360	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
$\hat{P}(r_{i,t}^{21} > 0)$	0.082**	-0.007	0.115**	0.061	0.103**	0.059*	0.075***	0.049**
	(0.034)	(0.027)	(0.052)	(0.056)	(0.044)	(0.032)	(0.021)	(0.023)
CIV		-0.034***		-0.030***		-0.031***		-0.027***
		(0.009)		(0.009)		(0.008)		(0.007)
PIV		0.019**		0.019**		0.023**		0.024**
		(0.009)		(0.009)		(0.009)		(0.010)
SKEW		-0.010		-0.009		-0.011		-0.008
		(0.007)		(0.007)		(0.007)		(0.007)
ΔCIV		-0.006		-0.007		-0.008		-0.007
		(0.007)		(0.007)		(0.007)		(0.008)
ΔPIV		0.018**		0.017**		0.015*		0.014*
		(0.008)		(0.008)		(0.008)		(0.008)
log(O/S)		-0.000		-0.000		0.000		-0.000
		(0.000)		(0.000)		(0.000)		(0.000)
IVSpread		-0.065***		-0.063***		-0.064***		-0.061***
		(0.012)		(0.012)		(0.013)		(0.012)
IVSkew		-0.009		-0.009		-0.011**		-0.009*
		(0.006)		(0.006)		(0.005)		(0.005)
IVol-RVol		0.005		0.005		0.003		0.003
		(0.004)		(0.004)		(0.003)		(0.003)
beta		0.002		0.002		0.002		0.002
		(0.003)		(0.003)		(0.003)		(0.003)
log(ME)		-0.001		-0.001		-0.001		-0.000
		(0.001)		(0.001)		(0.001)		(0.000)
N	836,275	321,721	836,275	321,721	904,497	344,223	904,497	344,223
Adj R^2	0.022	0.070	0.022	0.070	0.030	0.068	0.021	0.068

Table 6: Logistic Regression, Future Stock Return

	Dependent variable: $\mathbf{I}(r_{i,t}^{21} > 0)$		
	(1)	(2)	(3)
$\hat{P}(r_{i,t}^{21} > 0)$	0.616***		0.391***
	(0.015)		(0.025)
CIV		-0.172***	-0.183***
		(0.057)	(0.057)
PIV		-0.147**	-0.188***
		(0.062)	(0.062)
SKEW		0.193***	0.211***
		(0.041)	(0.041)
ΔCIV		0.090*	0.114**
		(0.049)	(0.049)
ΔPIV		-0.057	-0.057
		(0.049)	(0.049)
log(O/S)		-0.029***	-0.026***
		(0.003)	(0.003)
IVSpread		-0.403***	-0.368***
		(0.069)	(0.069)
IVSkew		-0.156***	-0.165***
		(0.046)	(0.046)
IVol-RVol		0.035*	0.023
		(0.020)	(0.020)
VOV		0.056	0.086*
		(0.048)	(0.048)
beta		-0.021**	-0.007
		(0.008)	(0.008)
log(ME)		0.031***	0.023***
		(0.003)	(0.003)
$ret_{12,2}$		-0.012***	-0.006
		(0.004)	(0.004)
$ret_{1,0}$		-0.001***	-0.001***
		(0.000)	(0.000)
Observations	904497	342535	342535
Pseudo R^2	0.001	0.003	0.003
Note:		*p<0.1; **p<0.05; ***p<0.01	

Table 7: Logistic Regression, Future Stock Crash

	$\mathbf{I}(r^H_{i,t} < -0.1)$			$\mathbf{I}(r^H_{i,t} < -0.2)$			$\mathbf{I}(r^H_{i,t} < -0.3)$		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
$\hat{P}(r^{21}_{i,t} > 0)$	-1.411***		-0.673***	-1.974***		-1.341***	-2.042***		-1.529***
	(0.031)		(0.038)	(0.056)		(0.071)	(0.089)		(0.119)
CIV		0.546***	0.559***		0.573***	0.607***		0.722***	0.773***
		(0.125)	(0.128)		(0.149)	(0.157)		(0.125)	(0.130)
PIV		1.485***	1.558***		1.956***	2.072***		1.551***	1.646***
		(0.133)	(0.137)		(0.164)	(0.173)		(0.152)	(0.156)
SKEW		0.070	0.030		0.141	0.052		0.126	0.030
		(0.064)	(0.065)		(0.101)	(0.103)		(0.120)	(0.122)
ΔCIV		-0.018	-0.065		0.201*	0.120		0.531***	0.454***
		(0.069)	(0.069)		(0.107)	(0.107)		(0.160)	(0.161)
ΔPIV		0.210***	0.217***		0.409***	0.413***		0.467***	0.458***
		(0.069)	(0.069)		(0.105)	(0.106)		(0.157)	(0.160)
log(O/S)		0.093***	0.086***		0.153***	0.140***		0.202***	0.187***
		(0.005)	(0.005)		(0.009)	(0.009)		(0.015)	(0.015)
IVSpread		0.130	0.067		0.269*	0.186		0.500***	0.447**
		(0.109)	(0.111)		(0.142)	(0.145)		(0.178)	(0.183)
IVSkew		-0.109	-0.100		-0.188*	-0.188		0.142	0.139
		(0.068)	(0.069)		(0.113)	(0.115)		(0.171)	(0.174)
IVol-RVol		0.162***	0.182***		0.315***	0.353***		0.325***	0.365***
		(0.028)	(0.029)		(0.046)	(0.048)		(0.045)	(0.045)
VOV		-1.110***	-1.150***		-1.331***	-1.385***		-0.909***	-0.950***
		(0.081)	(0.081)		(0.155)	(0.157)		(0.245)	(0.248)
beta		0.195***	0.175***		0.261***	0.230***		0.307***	0.282***
		(0.012)	(0.012)		(0.020)	(0.021)		(0.033)	(0.034)
log(ME)		-0.131***	-0.115***		-0.210***	-0.175***		-0.253***	-0.212***
		(0.005)	(0.005)		(0.008)	(0.009)		(0.014)	(0.015)
$ret_{12,2}$		-0.014***	-0.023***		-0.020*	-0.036***		-0.054**	-0.072***
		(0.005)	(0.006)		(0.011)	(0.013)		(0.022)	(0.022)
$ret_{1,0}$		-0.004***	-0.004***		-0.007***	-0.007***		-0.011***	-0.011***
		(0.000)	(0.000)		(0.001)	(0.001)		(0.001)	(0.001)
Observations	904497	342535	342535	904497	342535	342535	904497	342535	342535
Pseudo R^2	0.005	0.062	0.063	0.008	0.119	0.122	0.007	0.141	0.145

Note: *p<0.1; **p<0.05; ***p<0.01

Table 8: Summary Statistics of K-Means Clustering, 100 Clusters

	N	Mean	Std	Skew	Min	25th	Median	75th	Max
$E(R_s)$	500	0.016	0.010	0.204	-0.011	0.010	0.015	0.021	0.047
Realized Return	500	0.009	0.074	0.322	-0.307	-0.032	0.008	0.049	0.413
$Std(R_s)$	500	0.503	0.091	1.883	0.389	0.441	0.475	0.536	0.923
Realized Volatility	500	0.293	0.137	2.447	0.054	0.205	0.263	0.345	1.305
$Skew(R_s)$	500	0.453	0.082	0.334	0.258	0.400	0.453	0.508	0.732
$E(R_m)$	500	0.008	0.005	-0.592	-0.012	0.006	0.009	0.012	0.025
$Std(R_m)$	500	0.197	0.009	0.860	0.173	0.192	0.197	0.203	0.230
β_{hist}	498	1.029	0.385	0.537	0.163	0.767	0.994	1.257	2.510
β_{fwd}	500	1.021	0.116	0.543	0.762	0.946	1.005	1.082	1.426

Table 9: Portfolio Returns, K-Means Clustering

	Decile Portfolios										
	1	2	3	4	5	6	7	8	9	10	10 - 1
Equal-Weighted Portfolios											
$E(R_s)$	0.38	0.63**	0.78***	0.90***	0.90***	0.94***	1.02***	1.15***	1.21***	1.40**	1.02**
	(1.15)	(2.29)	(2.80)	(3.02)	(2.90)	(2.71)	(2.82)	(2.82)	(2.91)	(2.52)	(2.21)
$Std(R_s)$	0.84***	0.87***	1.04***	0.95***	0.97***	0.90***	0.97**	0.98**	0.92*	0.90	0.07
	(5.19)	(4.22)	(4.24)	(3.53)	(3.11)	(2.69)	(2.54)	(2.15)	(1.84)	(1.41)	(0.12)
$Skew(R_s)$	1.34**	1.17***	1.25***	1.03***	0.88**	0.83***	0.86***	0.86***	0.72***	0.35	-0.99*
	(2.36)	(2.58)	(3.19)	(2.82)	(2.55)	(2.76)	(2.91)	(3.16)	(2.62)	(1.06)	(-1.96)
β_{fwd}	0.72***	0.79***	0.96***	0.90***	1.00***	0.95***	0.94**	1.06**	1.08**	0.95	0.23
	(3.04)	(3.13)	(3.53)	(3.16)	(3.20)	(2.87)	(2.56)	(2.50)	(2.28)	(1.57)	(0.44)
β_{hist}	0.88***	0.81***	0.94***	0.91***	0.85***	1.07***	1.07***	0.90**	0.91*	0.93	0.05
	(5.12)	(3.63)	(3.77)	(3.09)	(2.68)	(3.17)	(2.76)	(2.12)	(1.91)	(1.48)	(0.09)
Value-Weighted Portfolios											
$E(R_s)$	0.48	0.55**	0.84***	0.93***	0.96***	0.79**	1.13***	0.90**	1.19***	1.52***	1.04**
	(1.38)	(2.33)	(3.22)	(2.91)	(3.09)	(2.15)	(3.07)	(2.29)	(3.01)	(2.82)	(2.35)
$Std(R_s)$	0.79***	0.81***	0.98***	1.11***	0.99***	0.74**	0.95**	0.81**	0.94**	1.16*	0.37
	(4.74)	(3.98)	(3.66)	(3.72)	(3.00)	(1.99)	(2.34)	(2.04)	(1.98)	(1.73)	(0.61)
$Skew(R_s)$	1.53***	1.18***	1.15***	0.81**	0.80**	0.81**	1.08***	0.99***	0.71***	0.53*	-1.00**
	(2.93)	(2.69)	(3.33)	(2.31)	(2.22)	(2.55)	(3.34)	(3.61)	(2.78)	(1.66)	(-2.31)
β_{fwd}	0.64***	0.75***	0.88***	1.04***	0.91***	0.96***	0.78**	1.02***	1.15**	1.04	0.41
	(2.66)	(2.83)	(3.29)	(3.18)	(2.78)	(2.71)	(2.03)	(2.59)	(2.56)	(1.62)	(0.70)
β_{hist}	0.90***	0.82***	0.97***	0.93***	0.68**	1.00***	0.97***	0.95**	0.84*	1.05*	0.15
	(5.21)	(3.71)	(4.64)	(3.30)	(2.24)	(3.11)	(2.78)	(2.46)	(1.79)	(1.75)	(0.27)

Note: $^{***}p < 0.01$; $^{**}p < 0.05$; $^{*}p < 0.1$

Table 10: Cross-Sectional Regression, K-Means Clustering, 100 Clusters

	10 Portfolios		25 Portfolios		50 Portfolios	
	(1)	(2)	(3)	(4)	(5)	(6)
const	-0.002	0.007***	-0.002	0.007***	-0.000	0.007***
	(0.004)	(0.001)	(0.003)	(0.001)	(0.003)	(0.001)
β_{fwd}	0.010**		0.010***		0.008**	
	(0.004)		(0.003)		(0.003)	
β_{hist}		0.001		0.001		0.001
		(0.001)		(0.001)		(0.001)
R^2	0.493	0.128	0.311	0.055	0.127	0.037
Adjusted R^2	0.430	0.019	0.281	0.014	0.108	0.017
Residual Std. Error	0.001	0.001	0.002	0.001	0.002	0.002
F Statistic	7.786**	1.172	10.360***	1.339	6.960**	1.864

Note: *p<0.1; **p<0.05; ***p<0.01

Figures

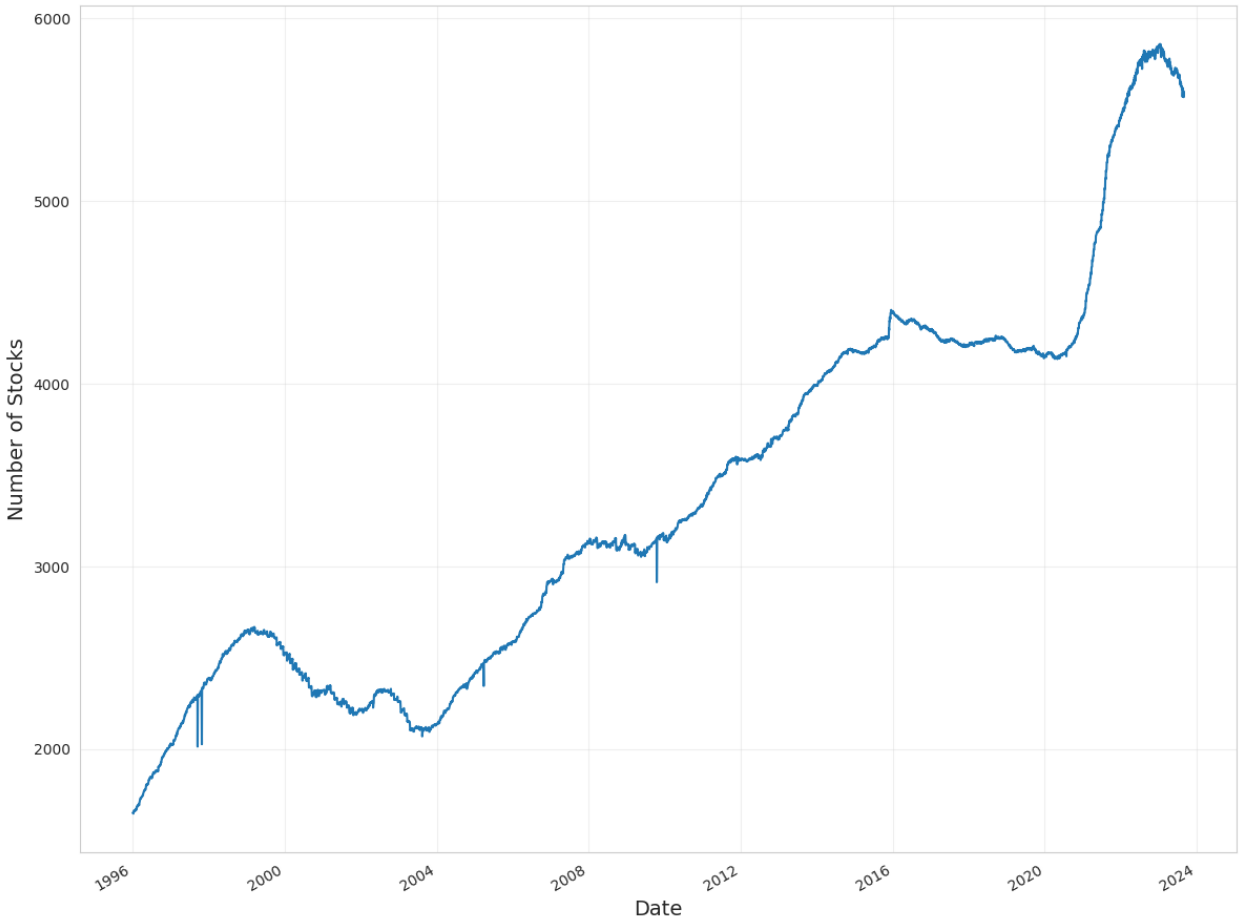

Figure 1: Number of Stocks with Option-Implied Volatility Surface Data (1996-2023). The figure shows the number of stocks with available option-implied volatility surface data from IvDB OptionMetrics, spanning from January 1996 to August 2023.

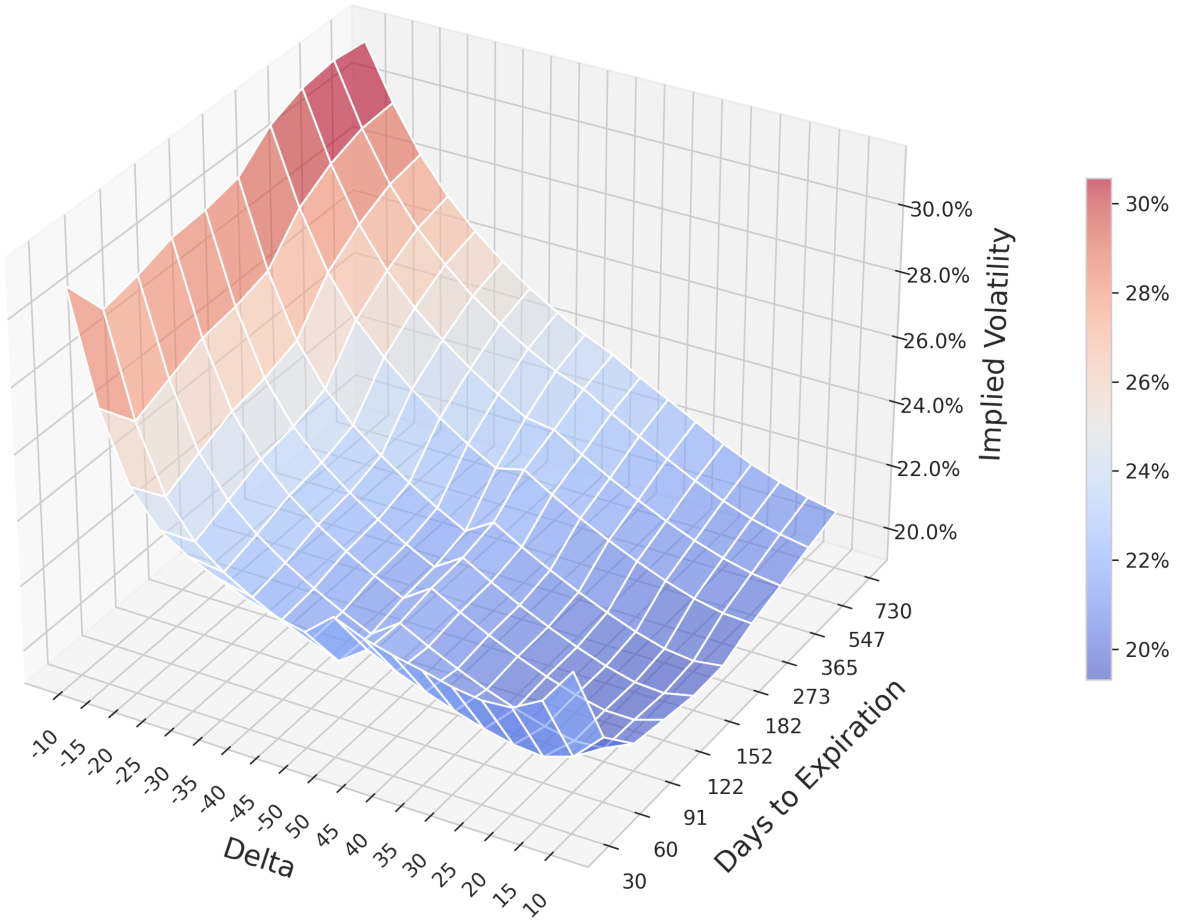

Figure 2: Example of Interpolated Option-Implied Volatility Surface (AAPL, 01/08/2023). The plot shows the volatility surface for Apple Inc. (AAPL) on Aug 1st, 2023, with moneyness (option delta) on the x-axis, days to expiration on the y-axis, and implied volatility levels represented by the color gradient.

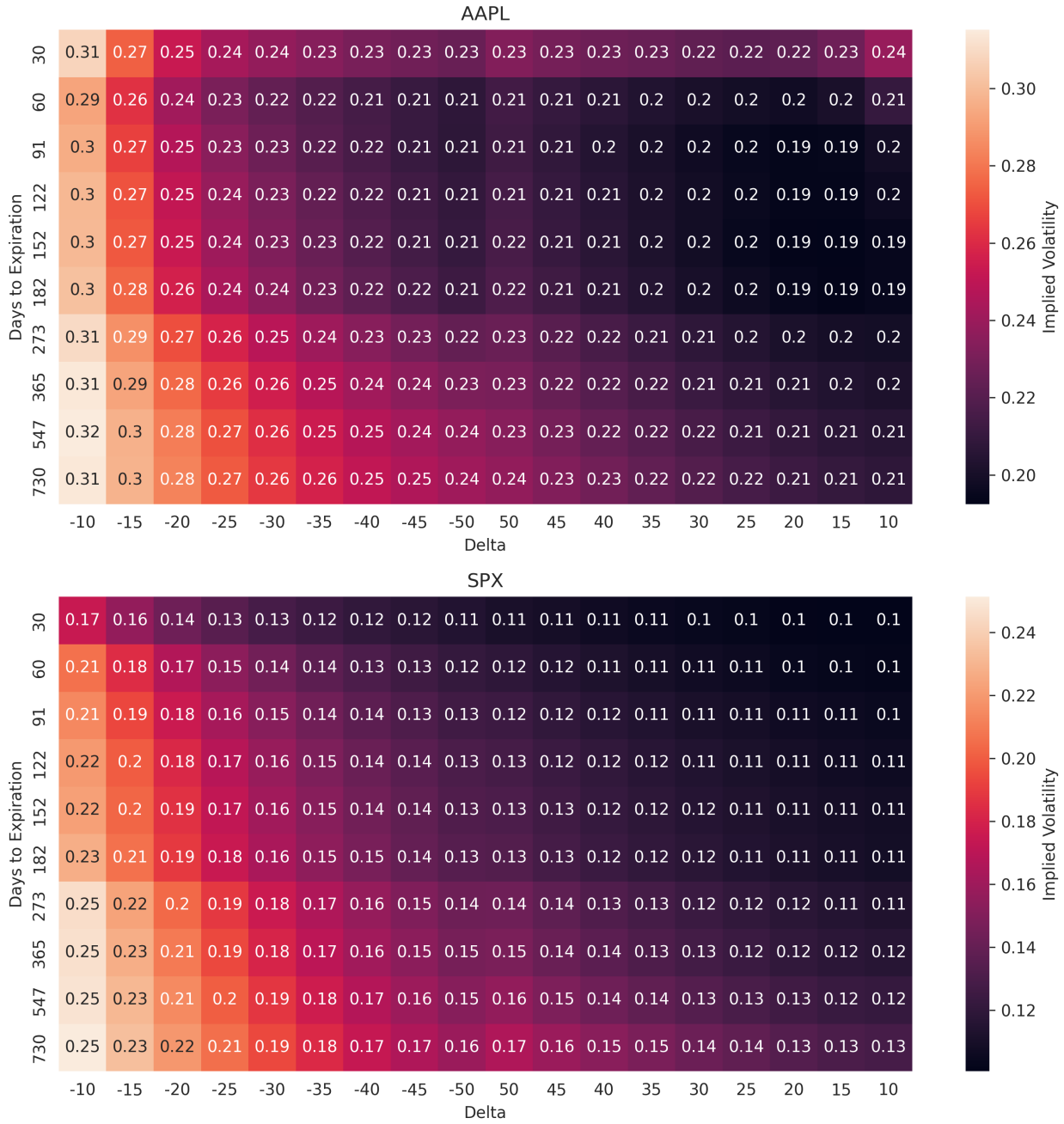

Figure 3: "Image" Representation of Volatility Surface (01/08/2023). The upper panel shows the heatmap of the volatility surface for Apple Inc. (AAPL), while the lower panel shows the volatility surface for the S&P 500 index (SPX) on the same date. The x-axis represents moneyness (option delta), and the y-axis represents days to expiration, with color intensity indicating implied volatility levels.

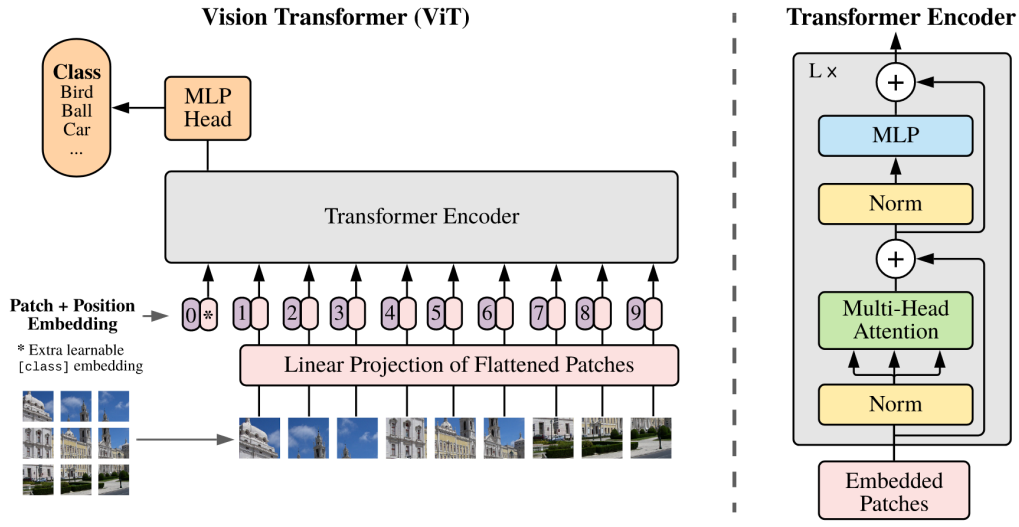

Figure 4: Vision Transformer (ViT) Architecture from Dosovitskiy et al. (2020). The image is split into fixed-size patches, linearly embedded, and then position embeddings are added. The resulting sequence of vectors is fed to a standard Transformer encoder. Similar to BERT, a classification token is prepended to the sequence, and the final hidden state corresponding to this token is used for classification tasks.

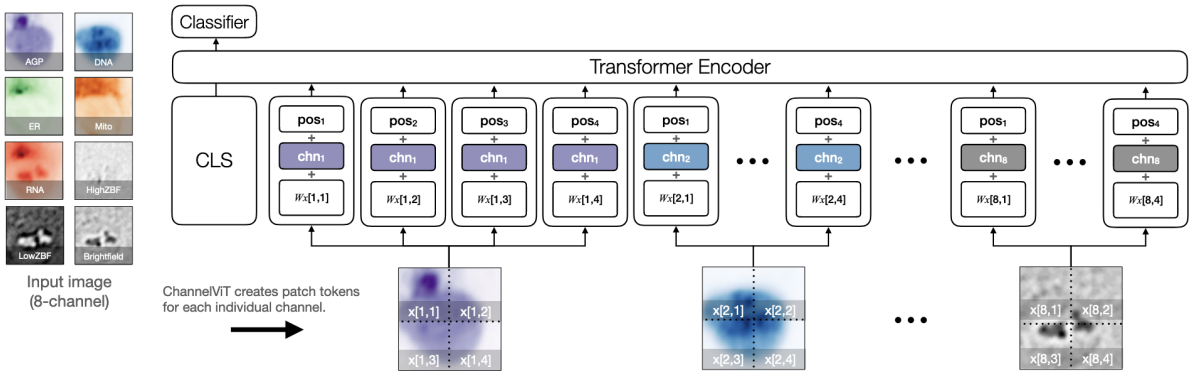

Figure 5: Channel Vision Transformer (ChannelViT) Architecture from Bao et al. (2024). The input image comprises multiple channels potentially carrying semantically distinct and independent information. ChannelViT constructs patch tokens for each individual channel, utilizing a learnable channel embedding **chn** to encode channel-specific information. The linear embedding **W** and positional embeddings **pos** are shared across all channels. The resulting sequence of vectors from all channels is concatenated and fed into a standard Transformer encoder. Similar to BERT, a classification token is prepended to the sequence, and the final hidden state corresponding to this token is used for classification tasks.

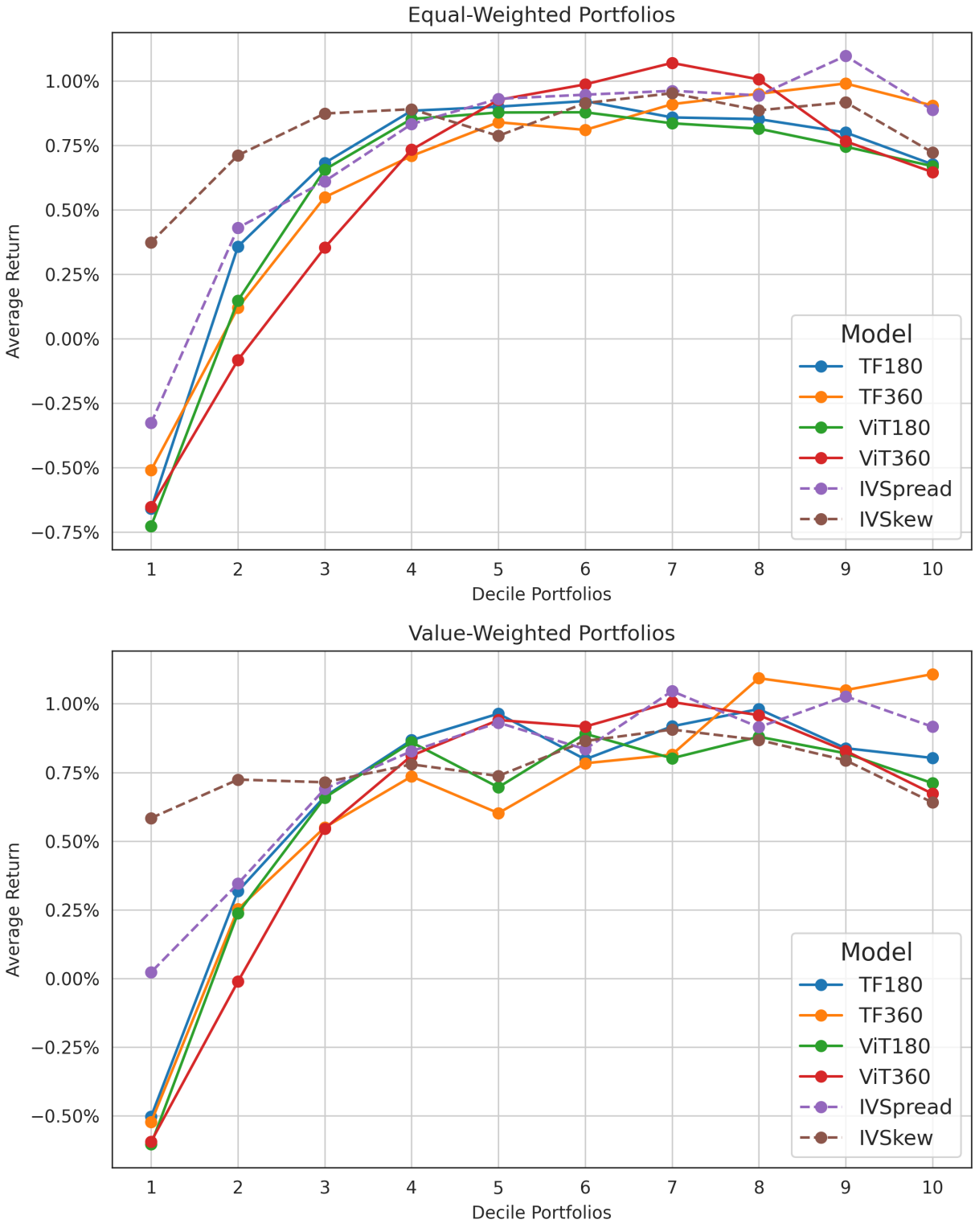

Figure 6: Portfolio Performance from Different Models.

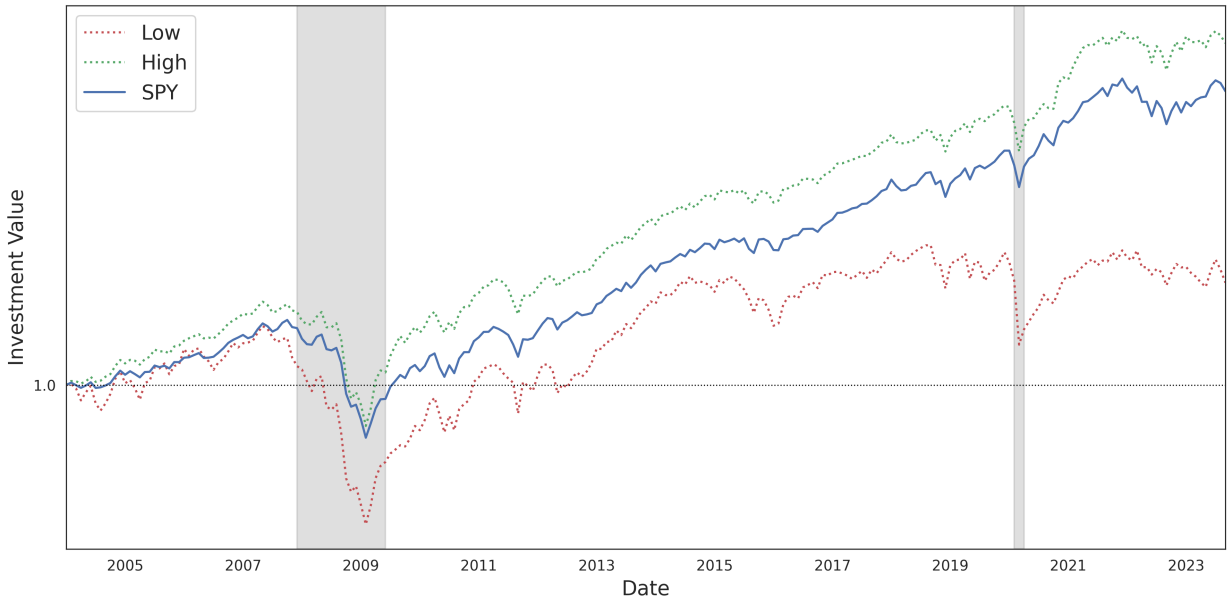

Figure 7: Cumulative Returns, S&P 500 Stocks. This figure shows the cumulative log return of equal-weight decile portfolios formed on out-of-sample predicted "up" probability $\hat{P}(r_{i,t}^{21} > 0)$ from January 2004 to August 2023. The sample includes only stocks that are part of the S&P 500 index. The "Low" series represents the 1/3 stocks with the lowest predicted "up" probabilities, while the "High" series represents the 2/3 stocks with the highest predicted "up" probabilities. The "SPY" series represents the cumulative log return of the S&P 500 index ETF (SPY) over the same period.

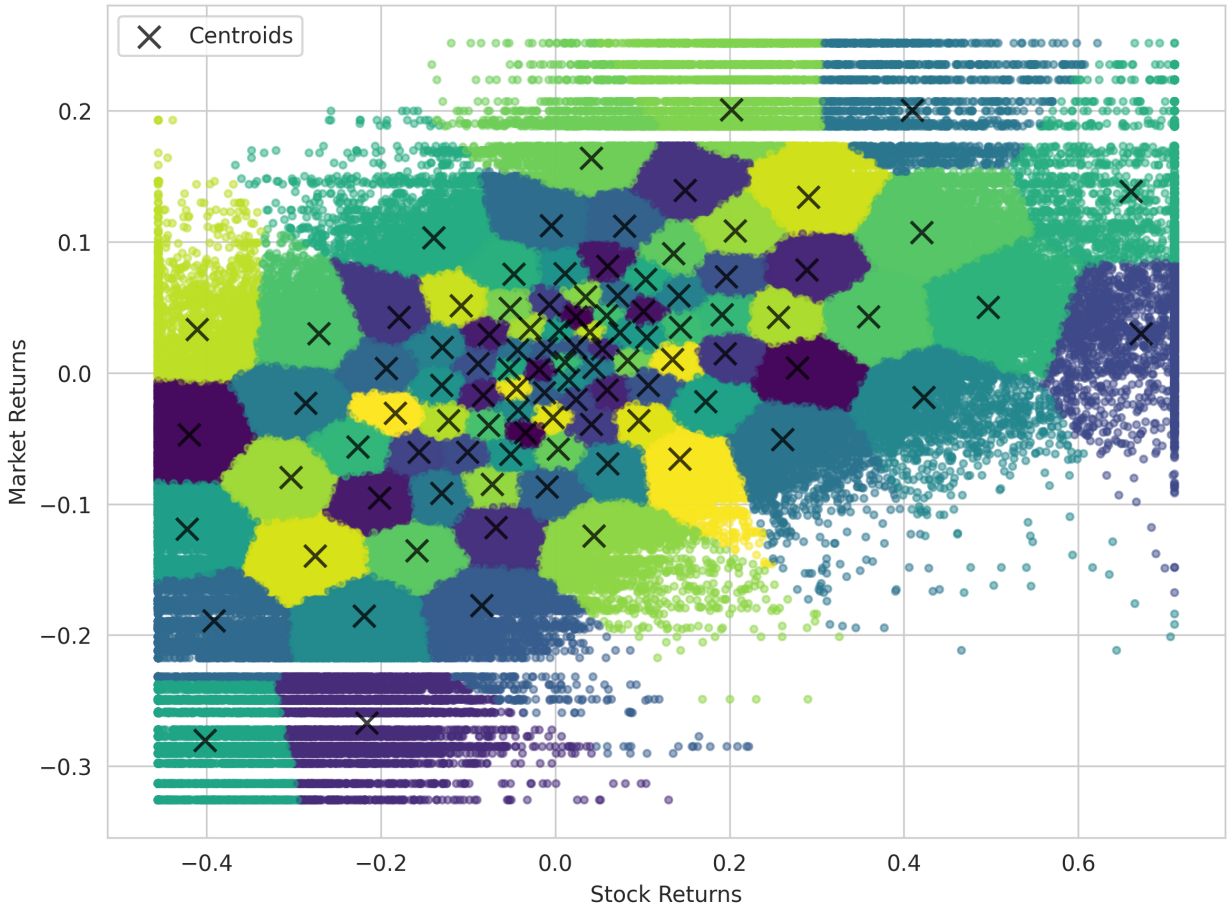

Figure 8: Classification Visualization (K-Means Clustering)

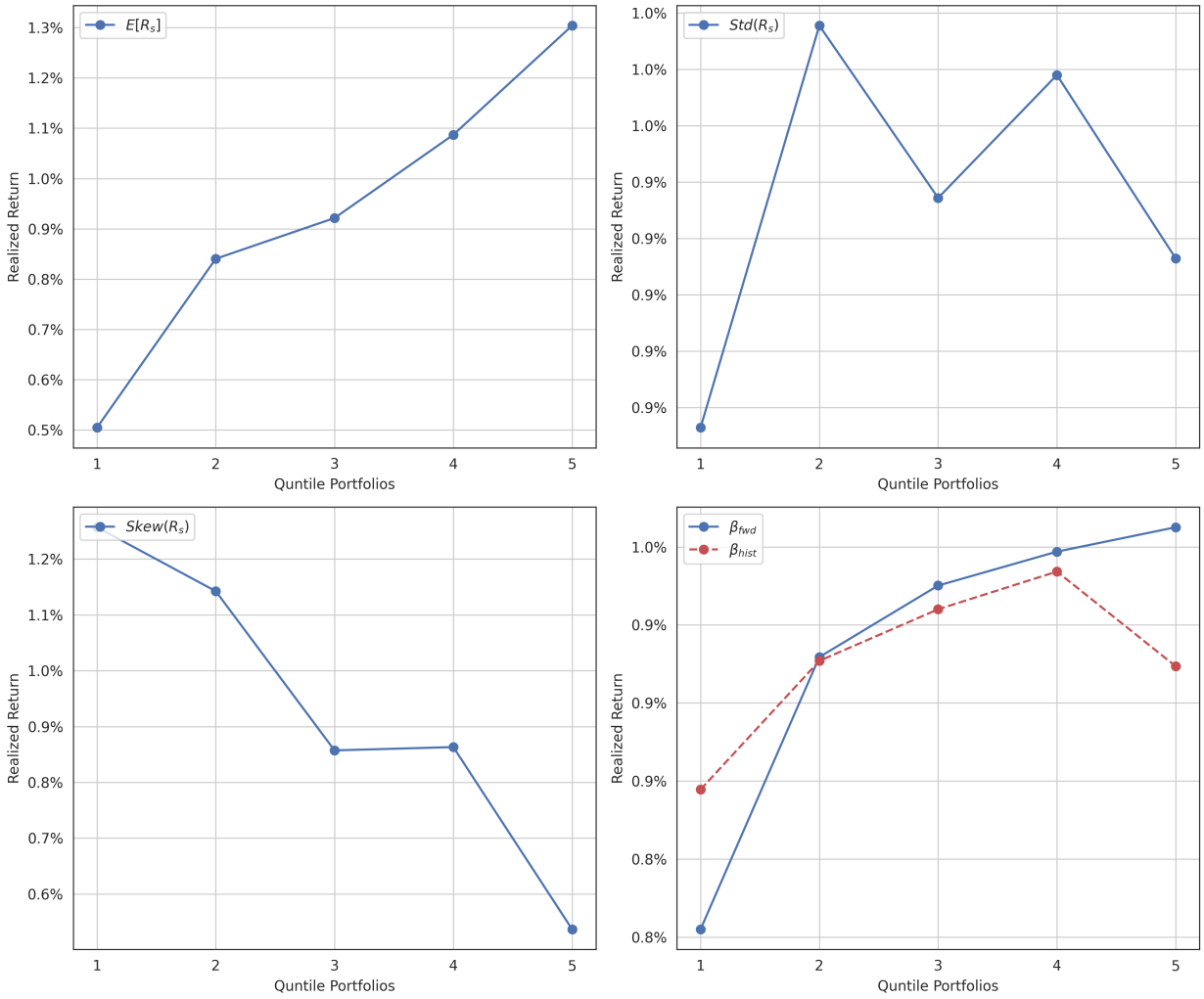

Figure 9: Portfolio Performance from K-Means Clustering (100 Clusters)

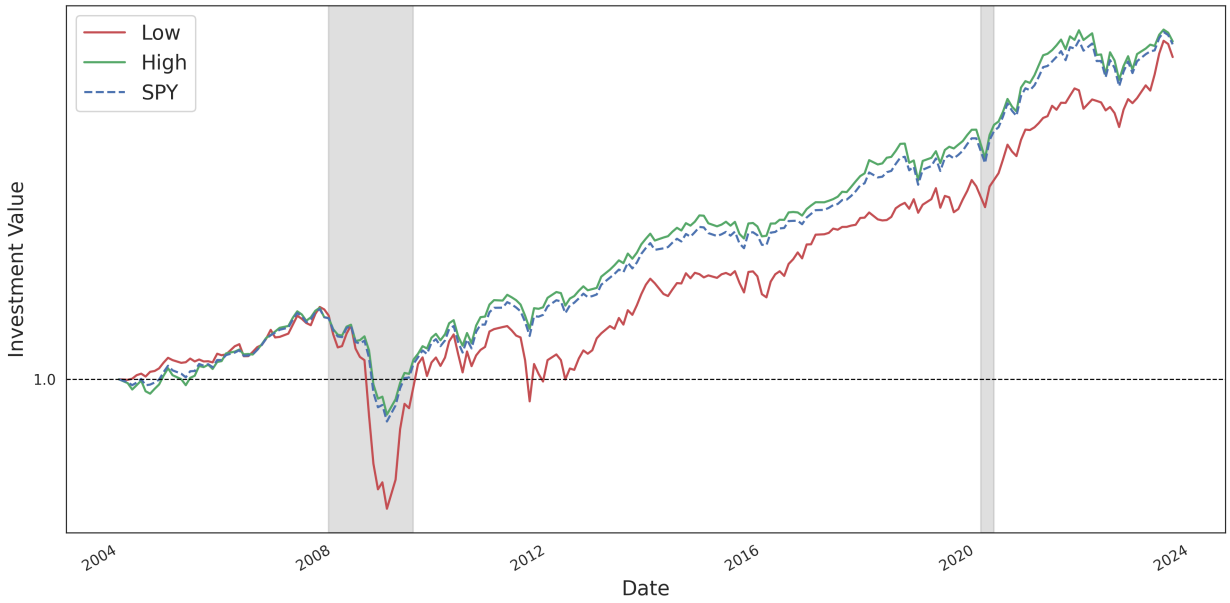

Figure 10: Cumulative Returns, S&P 500 Stocks (K-Means Clustering, 100 Clusters)

A Vision Transformer Model Structure

B Option-Based and Other Characteristics

This section provides additional details on the option-based characteristics used in the empirical tests. We select a set of most significant characteristics from Neuhierl et al. (2022). Most of these variables are calculated from the standardized option-implied volatility surface data (vsurfdYYYY), while the option volume are from the trading volume data (opvoldYYYY). Monthly sample period is from January 1996 to August 2023, same as the date range of OptionMetrics data.

- **CIV** & **PIV**: Near At-The-Money (ATM) call and put implied volatility with maturity of 30 days and absolute delta of 0.5, following An et al. (2014)
- **ΔCIV** & **ΔPIV**: Monthly change in the near ATM call and put implied volatility, also from An et al. (2014)
- **SKEW**: Difference between average put and call IV, using options with 30 days to expiration and average across call with delta between 0.2 to 0.4, and put with delta between -0.2 to -0.4, following Neuhierl et al. (2022).
- **log(O/S)**: Total monthly option volume over all call and put options, divided by monthly stock trading volume, following Johnson and So (2012).
- **IVS**$_{ATM}$: Difference between the near ATM put (delta = -0.5 and maturity = 30 days) and call (delta = 0.5 and maturity = 30 days) implied volatility, using the last daily observation of each month, following Yan (2011).
- **IVS**$_{OTM}$: Difference between the Out-The-Money (OTM) put implied volatility with moneyness closest to but above 1, and the OTM call implied volatility with moneyness closest to but below 1, using options with expiration between 10 and 60 days, following Xing et al. (2010).

Thanks to the work of Chen and Zimmermann (2022), we are able to directly download most of these option characteristics from their website [5]. The methodology that they used closely followed the original papers with only minor modifications. It is worth noting that we use the same name conventions as in Neuhierl et al. (2022), while the construction of these variables might be slightly different from them.

[5] `https://www.openassetpricing.com/`

Besides these option-based characteristics, we also include the following commonly used stock characteristics as control variables in the empirical tests: CAPM beta (beta), logarithm of market capitalization (log(ME)), past 1-month return ($ret_{1,0}$), past 12-month return excluding the most recent month ($ret_{12,2}$). These stock characteristics are also downloaded from Chen and Zimmermann (2022), except for the CAPM beta, which is downloaded from the Beta Suite by WRDS[6], calculated using daily returns over a 252-trading-day estimation window and a 126-trading-day minimum requirement.

C Additional Results

Table C.1: Portfolio Analysis, Binary Classification Model, Non-Micro Stocks

	Decile Portfolios										
Model	1	2	3	4	5	6	7	8	9	10	10 - 1
Equal-Weighted Portfolios											
TF180	0.12	0.69	0.86*	0.95**	0.95**	0.87**	0.91***	0.85***	0.80***	0.72***	0.59
	(0.22)	(1.33)	(1.90)	(2.33)	(2.50)	(2.52)	(2.82)	(2.90)	(3.18)	(3.49)	(1.42)
TF360	0.03	0.60	0.71	0.87**	0.80**	0.90***	0.92***	0.93***	0.99***	0.96***	0.93**
	(0.05)	(1.27)	(1.58)	(2.19)	(2.18)	(2.64)	(2.82)	(3.11)	(3.32)	(2.99)	(2.34)
ViT180	0.09	0.70	0.82*	0.89**	0.88**	0.82**	0.86***	0.83***	0.76***	0.69***	0.60
	(0.15)	(1.40)	(1.89)	(2.23)	(2.45)	(2.37)	(2.69)	(2.99)	(2.98)	(3.35)	(1.37)
ViT360	-0.19	0.49	0.65*	0.87**	0.89**	0.96***	1.03***	1.05***	0.86**	0.73**	0.92***
	(-0.40)	(1.12)	(1.72)	(2.35)	(2.52)	(2.64)	(2.81)	(3.30)	(2.43)	(2.13)	(2.65)
Value-Weighted Portfolios											
TF180	0.32	0.61	0.83*	0.92**	0.82**	0.91***	0.91***	0.96***	0.77***	0.78***	0.46
	(0.52)	(1.10)	(1.75)	(2.14)	(2.03)	(2.69)	(2.71)	(3.25)	(3.28)	(4.12)	(0.92)
TF360	0.10	0.67	0.54	0.64	0.70*	0.79***	0.98***	0.99***	0.98***	1.05***	0.94*
	(0.17)	(1.26)	(1.09)	(1.49)	(1.82)	(2.64)	(2.90)	(3.46)	(3.76)	(3.46)	(1.90)
ViT180	0.04	0.86	0.79	0.85*	0.64*	0.87***	0.79**	0.91***	0.78***	0.70***	0.66
	(0.06)	(1.48)	(1.63)	(1.82)	(1.72)	(2.75)	(2.36)	(3.48)	(3.03)	(3.48)	(1.25)
ViT360	-0.13	0.68	0.81*	0.84**	0.90***	0.89**	0.98***	1.05***	0.79**	0.73*	0.86**
	(-0.25)	(1.58)	(1.86)	(2.31)	(2.59)	(2.48)	(2.74)	(3.24)	(2.41)	(1.87)	(2.38)

Note: $^{***}p < 0.01$; $^{**}p < 0.05$; $^{*}p < 0.1$

[6] https://wrds-www.wharton.upenn.edu/pages/get-data/beta-suite-wrds/

Table C.2: Time Series Regression, Binary Classification Model, Non-Micro Stocks

	TF180	TF360	ViT180	ViT360	IVSpread	IVSkew
	(1)	(2)	(3)	(4)	(5)	(6)
Alpha	0.006**	0.011***	0.008***	0.009***	0.006***	-0.000
	(0.003)	(0.003)	(0.003)	(0.003)	(0.001)	(0.002)
MKT	-0.630***	-0.708***	-0.738***	-0.339***	-0.035	0.037
	(0.100)	(0.139)	(0.063)	(0.104)	(0.045)	(0.054)
SMB	-0.626***	-0.639***	-1.038***	-0.862***	0.024	-0.126
	(0.115)	(0.163)	(0.128)	(0.154)	(0.078)	(0.118)
HML	0.461***	0.358*	0.390**	0.426*	0.093	-0.290***
	(0.124)	(0.204)	(0.155)	(0.248)	(0.110)	(0.092)
RMW	0.953***	0.880***	1.117***	0.711***	0.343***	-0.032
	(0.203)	(0.237)	(0.219)	(0.263)	(0.111)	(0.154)
CMA	0.082	0.005	0.325	-0.359	-0.056	-0.278*
	(0.213)	(0.330)	(0.282)	(0.379)	(0.182)	(0.142)
UMD	0.130	-0.187	0.223	-0.156	0.071*	0.150*
	(0.180)	(0.204)	(0.178)	(0.186)	(0.038)	(0.084)
Observations	220	220	236	236	228	228
R^2	0.591	0.468	0.677	0.270	0.091	0.235
Adjusted R^2	0.580	0.453	0.668	0.251	0.067	0.215
Residual Std. Error	0.041	0.050	0.044	0.059	0.024	0.030
F Statistic	31.930***	23.746***	106.589***	9.118***	3.055***	8.704***

Note: *p<0.1; **p<0.05; ***p<0.01

Table C.3: Fama-MacBeth Regression, Binary Classification Model, Non-Micro Stocks

	TF180		TF360		ViT180		ViT360	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
$\hat{P}(r_{i,t}^{21} > 0)$	0.034	-0.004	0.067	0.070	0.048	0.045	0.064***	0.057**
	(0.036)	(0.028)	(0.073)	(0.055)	(0.042)	(0.033)	(0.024)	(0.025)
CIV		-0.033***		-0.028**		-0.027***		-0.022**
		(0.011)		(0.011)		(0.010)		(0.011)
PIV		0.020*		0.018*		0.021*		0.021*
		(0.012)		(0.011)		(0.012)		(0.013)
SKEW		-0.006		-0.005		-0.007		-0.004
		(0.009)		(0.009)		(0.008)		(0.008)
ΔCIV		-0.009		-0.010		-0.010		-0.009
		(0.008)		(0.008)		(0.008)		(0.008)
ΔPIV		0.023**		0.021**		0.020**		0.017*
		(0.010)		(0.010)		(0.009)		(0.009)
log(O/S)		0.000		-0.000		0.000		0.000
		(0.000)		(0.000)		(0.000)		(0.000)
IVSpread		-0.062***		-0.060***		-0.061***		-0.057***
		(0.014)		(0.013)		(0.013)		(0.012)
IVSkew		-0.006		-0.006		-0.008		-0.005
		(0.007)		(0.007)		(0.006)		(0.006)
IVol-RVol		0.005		0.005		0.004		0.004
		(0.004)		(0.004)		(0.004)		(0.004)
beta		0.001		0.002		0.001		0.001
		(0.003)		(0.003)		(0.003)		(0.003)
log(ME)		-0.000		-0.001		-0.001		-0.000
		(0.001)		(0.001)		(0.001)		(0.000)
N	539,932	288,990	539,932	288,990	581,697	309,011	581,697	309,011
Adj R^2	0.027	0.073	0.027	0.074	0.034	0.071	0.028	0.071