Transformer-IVS[*]

Zhenqi Hu[†]

September 7, 2025

Abstract

Placeholder

Keywords: key1, key2, key3

JEL Codes: key1, key2, key3

[*]abc
[†]abc

1 Introduction

2 Literature Review

3 The Vision Transformer Model

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

3.1 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:

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

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:

    $$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

    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:

    $$\mathbf{z}'_l = \text{MSA}(LN(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \tag{3}$$

    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:

  $$\text{MLP}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \tag{4}$$

  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:

  $$\mathbf{z}_l = \text{MLP}(LN(\mathbf{z}'_l)) + \mathbf{z}'_l \tag{5}$$

  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:

$$\mathbf{v} = LN(\mathbf{z}_L^0) \tag{6}$$

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:

$$\mathbf{y} = \text{softmax}(\mathbf{v}W_c + b_c) \tag{7}$$

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.

3.2 Image Representation of Option-Implied Volatility Surface

An option-implied volatility surface is a three-dimensional plot that displays the implied volatility of a stock or index options across different strike prices and expirations, which we take as the input image for the ViT model. For each stock i at time t, we choose the both the volatility surface of stock's options $IV_{i,t}$ and the volatility surface of the S&P 500 index options $IV_{spx,t}$ as the input features, since the index options contain essential information of market risk and risk permia (Andersen et al., 2015).

We can regard the two volatility surfaces as two channels of a informational "image", since they can be represented as a similar matrix with the same height and width (moneyness and maturity). However, unlike a standard RGB images, the two channels/surfaces contain semantically distinct and independent information, which is a challenge for the standard ViT model. Inspired by Bao et al. (2024), we use the so-called Channel ViT model for the embedding of the volatility surfaces.

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

$$v_{i,t} = ViT(IV_{i,t}, IV_{spx,t}|\theta) \tag{8}$$

Where $IV_{i,t}$ 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. $v_{i,t} \in \mathbb{R}^D$ is vector representation of the information contained in the volatility surfaces, and θ is the set of parameters of the ViT model.

3.3 Discretization of Return

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, or classify the stock return into "crash" and "non-crash" classes based on whether the future return is below a certain negative threshold q (e.g., -10%). Denote $y_{i,t}$ as the binary variable (e.g. $y_{i,t} = \mathbb{I}(r_{i,t\to t+H} > 0)$ for up and down classification, or $y_{i,t} = \mathbb{I}(r_{i,t\to t+H} < q)$ for crash and non-crash classification), the output of the classification head can be expressed as:

$$\mathbb{E}(y_{i,t}|IV_{i,t}, IV_{spx,t}) = \sigma(v'_{i,t}W_c + b_c) \tag{9}$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{10}$$

where $\sigma(\cdot)$ is the logistic(sigmoid) function, $W_c \in \mathbb{R}^D$ and $b_c \in \mathbb{R}^1$ are the weights and bias of the classification head.

For multi-class classification

$$\mathbf{p}_{i,t} = \text{softmax}(v'_{i,t}W_c + b_c) \tag{11}$$

$$\text{softmax}(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad j = 1, ..., K \tag{12}$$

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. The output $\mathbf{p}_{i,t} \in \mathbb{R}^K$ is a probability vector, where each element $\mathbf{p}^k_{i,t}$ represents the predicted probability of stock i's return falling into class k at time t.

3.4 Training Process

We use standard cross-entropy loss function for the classification problem, which is defined as:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) \tag{13}$$

where N is the number of samples, C is the number of classes, $y_{i,c}$ is the true label for sample i and class c, and $\hat{y}_{i,c}$ is the predicted probability for sample i and class c. For binary classification ($N = 2$), the loss simplifies to:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{14}$$

We adopt similar regularization techniques as in Gu et al. (2020). We use a Adaptive Moment Estimation With Weight Decay (AdamW) optimizer (Loshchilov and Hutter, 2019), which is standard in the transformer training. We set the learning rate to 1×10^{-5} and set the weight decay to 1×10^{-2} (equivalent to L2 regularization). Besides, 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 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.

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.

4 Empirical Results

4.1 Data

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 a vector $(IV_{i,t}, IV_{spx,t})$, where $IV_{i,t} = \{IV_{i,t}(\tau, \delta)\}_{\tau \in T, \delta \in \Delta}$ is the volatility surface of stock i at time t, and $IV_{spx,t} = \{IV_{i,t}(\tau, \delta)\}_{\tau \in T, \delta \in \Delta}$ is the volatility surface of the S&P 500 index at time t.

We obtain the daily return data from CRSP for all firms listed on NYSE, AMEX, and

[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.

NASDAQ, as well as the S&P 500 index. For each security i at time t, we calculate the cumulative return on security i from day t to day $t + H$ as:

$$r_{i,t\to t+H} = \prod_{j=0}^{H-1} (1 + r_{i,t+j}) - 1 \tag{15}$$

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 number of stocks with volatility surface data over time.

4.2 Directional Prediction of Stock Returns

4.3 Crash Prediction

5 Conclusion

References

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.

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

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*.

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

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.

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.

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

	Portfolio Sorted on $\hat{P}(r_{i,t\to t+21} > 0)$					
	Low	2	3	4	High	H-L
Panel A: Equal-Weighted Portfolios						
Excess Return (%)	−0.150	0.506	0.659	0.765	0.787	0.937
	(−0.283)	(1.197)	(1.803)	(2.212)	(2.261)	(2.653)
Sharpe Ratio	−0.068	0.272	0.390	0.479	0.528	0.683
CAPM α (%)	−1.149	−0.418	−0.181	−0.024	0.082	1.231
	(−3.612)	(−1.937)	(−0.884)	(−0.125)	(0.387)	(4.088)
FF6 α (%)	−0.647	−0.186	−0.014	0.081	0.165	0.813
	(−3.892)	(−1.347)	(−0.080)	(0.481)	(0.863)	(3.738)
Q5 α (%)	−0.367	0.018	0.080	0.136	0.248	0.615
	(−1.628)	(0.099)	(0.390)	(0.711)	(1.201)	(2.670)
Panel B: Value-Weighted Portfolios						
Excess Return (%)	−0.148	0.730	0.621	0.826	0.705	0.853
	(−0.285)	(1.601)	(1.789)	(2.796)	(2.988)	(2.206)
Sharpe Ratio	−0.068	0.373	0.376	0.563	0.565	0.536
CAPM α (%)	−1.119	−0.202	−0.209	0.087	0.086	1.205
	(−3.649)	(−0.801)	(−1.192)	(0.481)	(0.607)	(3.586)
FF6 α (%)	−0.689	0.018	−0.047	0.069	−0.029	0.660
	(−3.187)	(0.081)	(−0.280)	(0.412)	(−0.236)	(2.943)
Q5 α (%)	−0.406	0.185	−0.038	0.111	0.104	0.510
	(−1.564)	(0.745)	(−0.189)	(0.546)	(0.689)	(1.906)

Table 1: Portfolio Analysis of the Binary Classification Model (Up and Down), All Stocks

	Portfolio Sorted on $\hat{P}(r_{i,t\to t+21} > 0)$					
	Low	2	3	4	High	H-L
Panel A: Equal-Weighted Portfolios						
Excess Return (%)	0.035	0.601	0.684	0.813	0.825	0.790
	(0.069)	(1.453)	(1.880)	(2.361)	(2.411)	(2.354)
Sharpe Ratio	0.016	0.326	0.405	0.505	0.551	0.597
CAPM α (%)	-0.971	-0.319	-0.158	0.018	0.112	1.083
	(-3.138)	(-1.512)	(-0.766)	(0.090)	(0.555)	(3.725)
FF6 α (%)	-0.514	-0.108	0.000	0.107	0.188	0.702
	(-3.145)	(-0.744)	(-0.002)	(0.625)	(1.021)	(3.358)
Q5 α (%)	-0.282	0.077	0.082	0.168	0.270	0.552
	(-1.273)	(0.424)	(0.404)	(0.840)	(1.373)	(2.447)
Panel B: Value-Weighted Portfolios						
Excess Return (%)	-0.037	0.802	0.596	0.826	0.719	0.756
	(-0.072)	(1.823)	(1.730)	(2.754)	(3.091)	(1.993)
Sharpe Ratio	-0.017	0.418	0.364	0.562	0.575	0.480
CAPM α (%)	-1.020	-0.116	-0.230	0.084	0.100	1.120
	(-3.375)	(-0.492)	(-1.326)	(0.458)	(0.706)	(3.423)
FF6 α (%)	-0.638	0.105	-0.076	0.074	-0.022	0.616
	(-2.829)	(0.497)	(-0.453)	(0.450)	(-0.180)	(2.692)
Q5 α (%)	-0.354	0.238	-0.066	0.119	0.105	0.459
	(-1.352)	(1.020)	(-0.331)	(0.582)	(0.695)	(1.739)

Table 2: Portfolio Analysis of the Binary Classification Model (Up and Down), Non-Micro Stocks

	Portfolio Sorted on $\hat{P}(r_{i,t\to t+21} > 0)$					
	Low	2	3	4	High	H-L
Panel A: Equal-Weighted Portfolios						
Excess Return (%)	−1.140	−0.649	−0.422	−0.213	0.243	1.382
	(−1.411)	(−0.948)	(−0.694)	(−0.496)	(0.500)	(2.139)
Sharpe Ratio	−0.385	−0.255	−0.195	−0.115	0.146	0.559
CAPM α (%)	−2.140	−1.563	−1.251	−0.855	−0.307	1.832
	(−3.866)	(−3.634)	(−3.151)	(−2.397)	(−0.844)	(3.246)
FF6 α (%)	−1.375	−0.916	−0.819	−0.562	−0.050	1.326
	(−3.092)	(−2.714)	(−2.303)	(−1.611)	(−0.135)	(2.651)
Q5 α (%)	−0.955	−0.519	−0.573	−0.411	−0.034	0.921
	(−2.185)	(−1.577)	(−1.545)	(−1.104)	(−0.092)	(1.702)
Panel B: Value-Weighted Portfolios						
Excess Return (%)	−1.140	−0.360	−0.096	−0.027	0.402	1.542
	(−1.368)	(−0.499)	(−0.144)	(−0.057)	(0.739)	(2.392)
Sharpe Ratio	−0.376	−0.136	−0.042	−0.014	0.228	0.613
CAPM α (%)	−2.197	−1.341	−1.011	−0.741	−0.199	1.997
	(−3.867)	(−3.057)	(−2.325)	(−1.952)	(−0.489)	(3.536)
FF6 α (%)	−1.433	−0.668	−0.593	−0.407	0.096	1.528
	(−3.237)	(−2.036)	(−1.503)	(−1.087)	(0.231)	(3.087)
Q5 α (%)	−1.037	−0.261	−0.350	−0.232	0.086	1.123
	(−2.301)	(−0.832)	(−0.873)	(−0.583)	(0.213)	(2.091)

Table 3: Portfolio Analysis of the Binary Classification Model (Up and Down), Micro Stocks

	Portfolio Sorted on $\hat{P}(r_{i,t\to t+21} > 0)$					
	Low	2	3	4	High	H-L
Panel A: Equal-Weighted Portfolios						
Excess Return (%)	0.584	0.684	0.917	0.949	0.924	0.340
	(1.206)	(1.847)	(2.714)	(3.283)	(3.199)	(0.989)
Sharpe Ratio	0.290	0.415	0.593	0.655	0.672	0.241
CAPM α (%)	−0.409	−0.155	0.122	0.220	0.316	0.725
	(−1.457)	(−0.899)	(0.717)	(1.529)	(1.589)	(2.729)
FF6 α (%)	−0.182	−0.085	0.148	0.168	0.181	0.363
	(−0.894)	(−0.609)	(1.149)	(1.210)	(0.856)	(1.370)
Q5 α (%)	0.031	−0.031	0.176	0.197	0.253	0.221
	(0.134)	(−0.204)	(1.160)	(1.305)	(1.090)	(0.758)
Panel B: Value-Weighted Portfolios						
Excess Return (%)	0.714	0.642	0.802	0.940	0.782	0.068
	(1.617)	(1.976)	(2.475)	(3.089)	(3.502)	(0.189)
Sharpe Ratio	0.376	0.405	0.551	0.666	0.654	0.047
CAPM α (%)	−0.220	−0.160	0.048	0.214	0.235	0.455
	(−0.907)	(−0.924)	(0.314)	(1.501)	(1.271)	(1.574)
FF6 α (%)	−0.023	−0.104	0.044	0.170	0.024	0.047
	(−0.102)	(−0.646)	(0.341)	(1.199)	(0.146)	(0.173)
Q5 α (%)	0.017	−0.172	−0.008	0.218	0.149	0.131
	(0.071)	(−0.972)	(−0.052)	(1.465)	(0.748)	(0.427)

Table 4: Portfolio Analysis of the Binary Classification Model (Up and Down), S&P Constituents

Figures

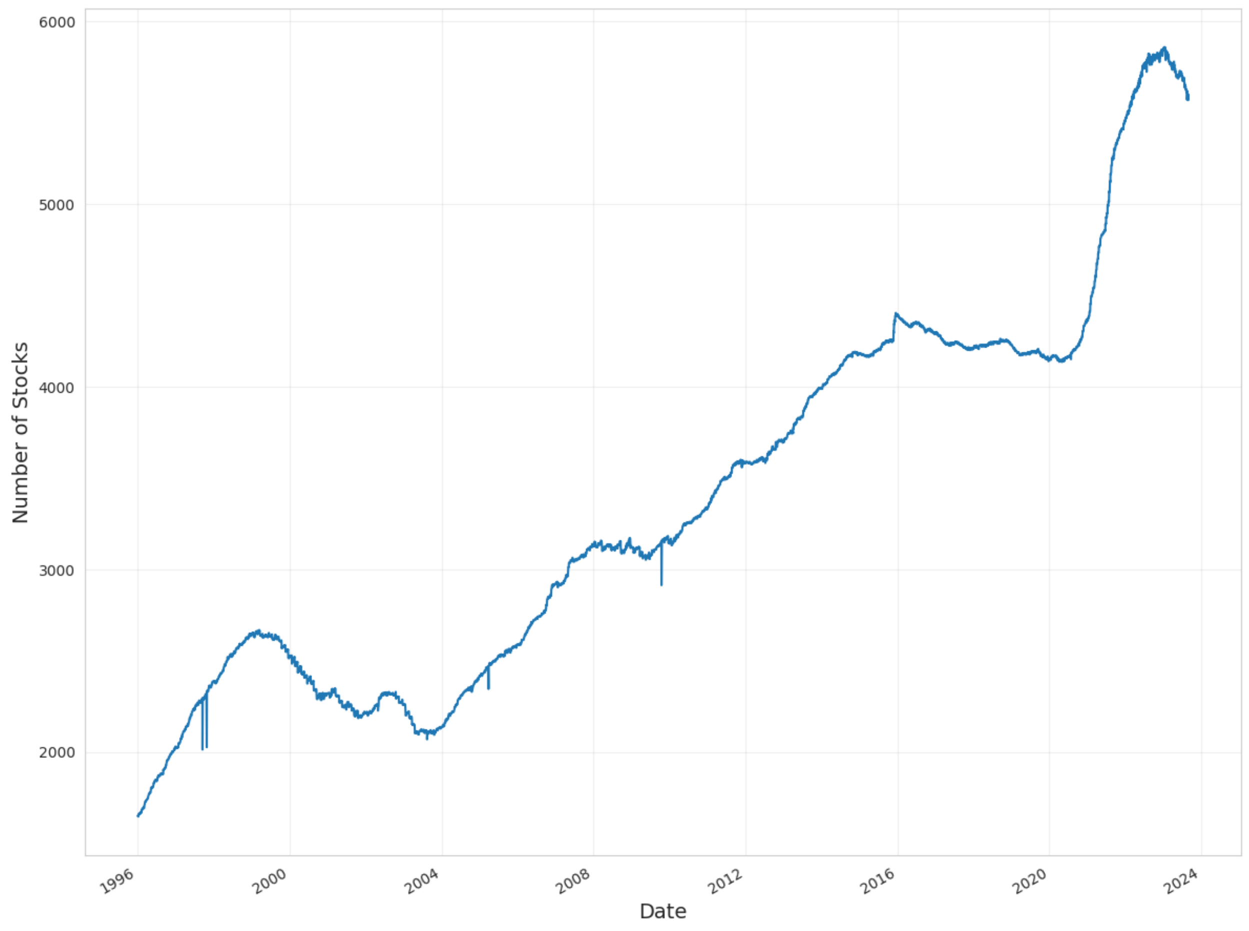

Figure 1: Number of Stocks with Option-Implied Volatility Surface Data (1996-2023).

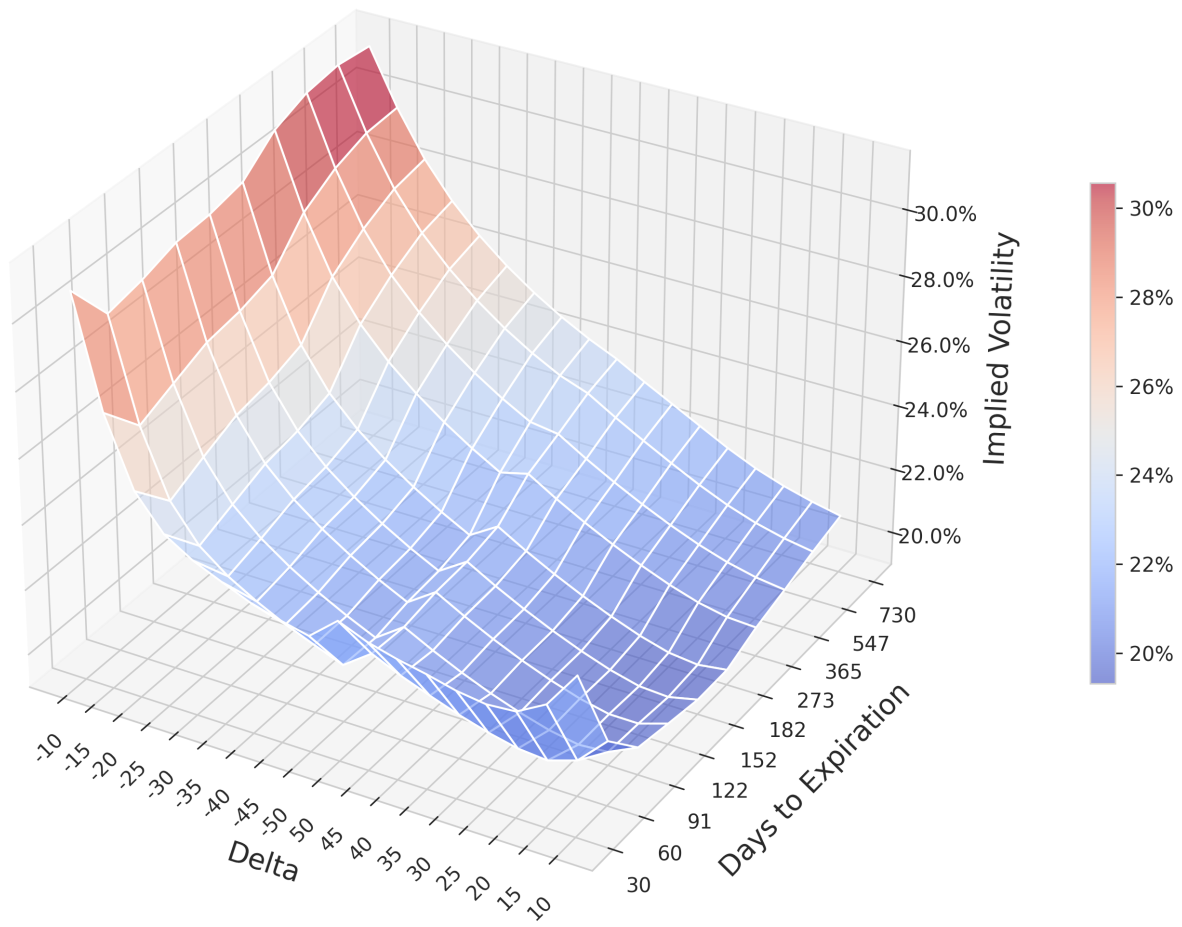

Figure 2: Example of Interpolated Option-Implied Volatility Surface (AAPL, 01/08/2023). The plot shows the volatility surface for Apple Inc. (AAPL) on Aug 1st, 2023, with moneyness (option delta) on the x-axis, days to expiration on the y-axis, and implied volatility levels represented by the color gradient.

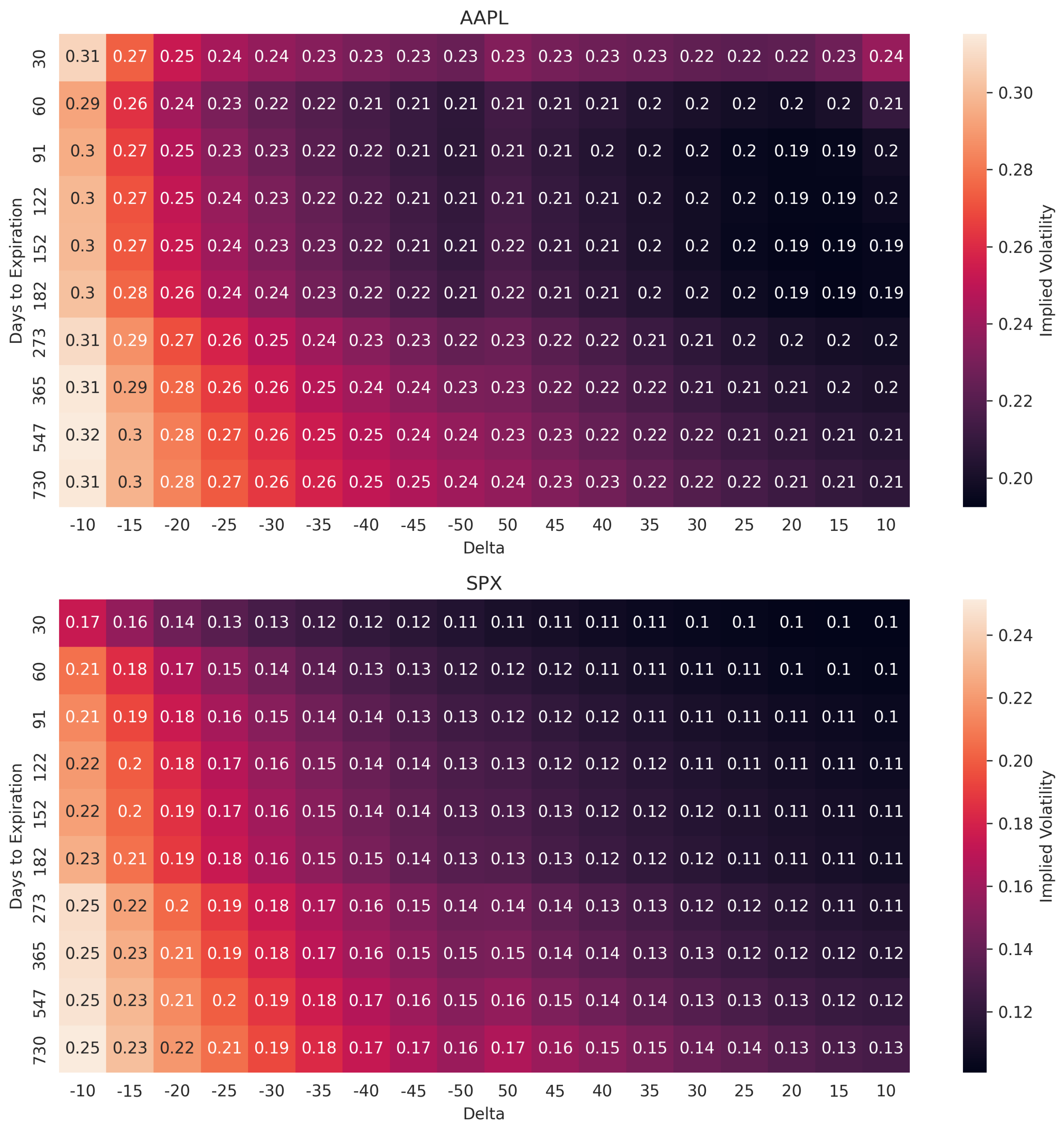

Figure 3: "Image" Representation of Volatility Surface (01/08/2023). The upper panel shows the heatmap of the volatility surface for Apple Inc. (AAPL), while the lower panel shows the volatility surface for the S&P 500 index (SPX) on the same date. The x-axis represents moneyness (option delta), and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

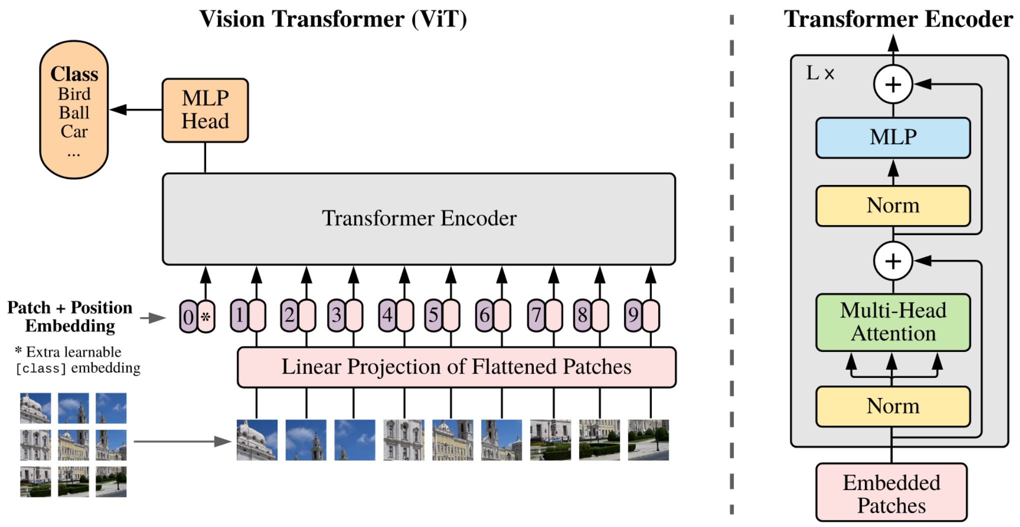

Figure 4: Vision Transformer (ViT) Architecture from Dosovitskiy et al. (2020). The image is split into fixed-size patches, linearly embedded, and then position embeddings are added. The resulting sequence of vectors is fed to a standard Transformer encoder. Similar to BERT, a classification token is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

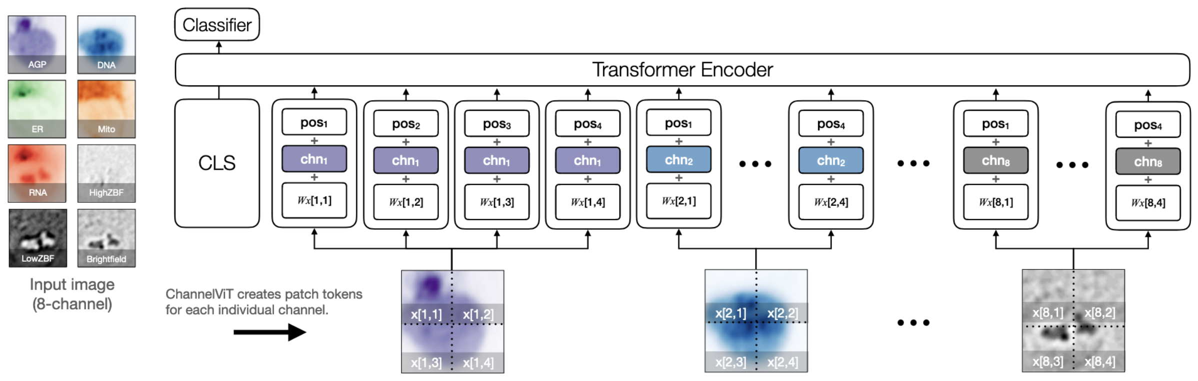

Figure 5: Channel Vision Transformer (Channel ViT) Architecture from Bao et al. (2024).

Appendix A. Placeholder