UNIVERSITY OF CALIFORNIA
Los Angeles

Building an Options Portfolio with Deep Learning

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Dylan Jorling

2023

ABSTRACT OF THE THESIS

Building an Options Portfolio with Deep Learning

by

Dylan Jorling
Master of Applied Statistics
University of California, Los Angeles, 2023
Professor Ying Nian Wu, Chair

Recent applications of powerful machine learning models within the field of portfolio optimization have shown promising results. This paper explores the application of similar methods to create an actively managed options portfolio, rather than a traditional portfolio containing stocks and ETFs. Although financial options are popular assets amongst both retail and institutional investors, they are almost exclusively traded for one of two purposes: hedging or intra-asset speculation. Little, if any, literature exists on the topic of inter-asset options strategies. Using percent change in implied volatility data as a proxy for single-day straddle returns, various machine learning models are trained by directly optimizing the Sharpe Ratio–a risk-adjusted return metric. The models output daily volatility positions in 315 underlying assets thereby creating the Options Portfolio. Results show significant potential in such a strategy, with the Attention Transformer model yielding a before-cost average annual Sharpe Ratio of 10.76 compared to the GRU model of 6.41, the LSTM model of 5.07, and the equal-weighted baseline of 0.53.

Nicolas Christou

George Michailidis

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2023

To my parents...

for their endless love and support, through thick and thin.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Background

A financial portfolio is a collection of assets owned by a single entity, containing a mixture of stocks, bonds, exchange-traded funds (ETFs), mutual funds, options, real estate, etc.[23]. Choosing specific assets to be included in a portfolio, along with the percent of total funds to allocate to each asset is a process called portfolio selection. Modern Portfolio Theory theoretically optimizes selection using the idea of mean-variance trade-off: for a given level of return, an investor will choose the portfolio that minimizes risk, while, for a given level of risk, an investor will choose the portfolio that maximizes expected return [20]. Portfolios that satisfy either of these conditions are said to lie on the efficient frontier and are the only portfolios rational investors consider constructing.

Reducing the risk of a portfolio can be achieved through diversification, which is defined as investing in multiple, ideally minimally correlated, assets. Given a portfolio containing n assets with equal weighting, the variance of the portfolio is calculated as [5]:

$$\sigma_P^2 = \frac{1}{n}\bar{\sigma_i^2} + \frac{n-1}{n}\bar{\sigma_{ij}} \tag{1.1}$$

where $\bar{\sigma_i^2}$ represents the average variance of each stock in the portfolio and $\bar{\sigma_{ij}}$ represents the average covariance amongst all pairs i, j in the portfolio such that $i \neq j$. When n is large, the variance of the portfolio is approximated as the average covariance of each asset pair. Clearly then, diversification reduces portfolio variance.

Traditionally, portfolio optimization is a two-step process [20]. In the first step, expected

returns and the variance-covariance matrix for all potential assets must be predicted, typically by using historical data. In the second step, a constrained optimization problem is solved such that either portfolio variance is minimized, given a targeted expected return, or expected return is maximized, given a target variance. The solution to the optimization problem is a set of assets and weights that lie on the estimated efficient frontier.

While the optimization step is theoretically sound, its reliance on the accuracy of the prediction step is problematic [35]. In practice, accurately predicting future returns and covariances is extremely difficult due to the stochastic and non-stationary nature of financial time series data. Estimation of the variance-covariance matrix is especially difficult as inter-asset correlations are constantly changing in not only magnitude, but direction. GARCH-based models developed by Engle (2002), [7] and later expanded upon by Fiszeder et al (2019) [8], as well as an SVR approach by Fiszeder et al (2021)[9], have proven useful in predicting future variance-covariance matrices for currency assets. Applying the same models to stock and ETF assets, however, has yielded sub-par results.

Recently a new approach to portfolio optimization has emerged that utilizes an end-to-end architecture where a performance metric replaces a traditional loss function and is directly maximized. The Sharpe Ratio [27] is a popular risk-adjusted return metric used to measure portfolio performance and is calculated as:

$$Sharpe = \frac{\mathbb{E}(R_p)}{\sigma_{R_p}} \tag{1.2}$$

where $\mathbb{E}(R_p)$ is the expected portfolio return, and σ_{R_p} is the estimated standard deviation of portfolio returns. The original idea of an end-to-end model using the Sharpe Ratio as the objective function was proposed by Moody et al (1998) [22] where the authors directly maximize the risk-adjusted return of a single asset. Zhang et al (2020) expand this idea to a portfolio containing 4 minimally correlated ETFs, and utilize deep learning models [35]. The authors' LSTM-based model produces a portfolio that dominates the mean-variance (MV) portfolio in nearly every financial metric. The model also proves robust against costs and outperforms an equal-weighted baseline up to a cost of 10 basis points. Zhang et al. (2021)

later use an end-to-end approach on a significantly larger dataset containing 730 stocks over a 37-year period [34]. The authors test various constraints other than the standard long-only constraint, including allowing for short selling, cardinality, leverage, and a combination of all three. Results again show end-to-end portfolios substantially outperforming MV portfolios, though increasing cost assumptions quickly cause performance to deteriorate compared to baseline, passively managed portfolios.

Kisiel and Gorse (2022) implement the current state of the art deep learning Attention Transformer (AT) model in a similar end-to-end framework on three datasets including ETFs, commodities, and stocks [17]. The AT portfolio significantly outperforms the previous state-of-the-art LSTM in nearly every metric. Critically, the authors integrate costs directly into the Sharpe loss function and returns prove to be more resilient to realistic costs.

The goal of this paper is to expand on the work of Kisiel et al. by implementing an end-to-end AT model for a proof-of-concept of an actively managed options portfolio. Financial options are complex assets derived from notoriously noisy underlying asset distributions. Therefore, full understanding of the mechanics of an options portfolio requires a base level of options knowledge, particularly regarding 1) contract features, 2) use-cases 3) strategies and 4) profit-and-loss (PNL). All four of these areas are discussed in the next section, before the chapter concludes with an overview of the proposed options portfolio.

1.2 Options Overview

An options contract gives the contract holder the right to buy (call) or sell (put) an underlying asset at a pre-specified price, called the strike. The components of an options contract include:

- Type: A call option gives the holder the right to buy an underlying asset at the strike price while a put option gives the holder the right to sell an underlying asset at the strike price. If the right to buy (sell) is exercised by the contract buyer, the contract seller is then obligated to sell (buy) the asset at the given strike price.

- Strike: The pre-specified price that an option buyer has the right to buy (call) or sell (put) the underlying asset at
- Expiration: The date the options contract expires on
- Multiplier: The quantity of the underlying asset that 1 contract gives the holder the right to buy (call) or sell (put); for stocks, the multiplier is typically 100, meaning 1 call contract give the buyer the right to buy 100 shares at the strike price. For futures-based contracts the option typically has the same multiplier as the future.
- Exercise Style: Two types of exercises exist: European and American. European options can only be exercised when the contract expires, while American options can be exercised at any time. A vast majority of stocks and ETFs are American-style options [19].
- Underlying Asset: The asset that the contract references

Options contracts are almost exclusively traded as either a risk-reducing "hedge" trade, or a high-risk, high-reward speculation trade. Hedging is a financial strategy that involves preventing loss, guaranteeing gain, or a combination of both. Hedging with options is often done by either commodity producers to protect against falling prices, or stock-market participants to protect against adverse market returns, though many other examples exist.

An oil producer who will produce 100,000 barrels of oil over the next month may want to hedge against a sudden drop in oil prices. If oil futures are currently priced at \$100, and the producer wants protection against any downturn greater than 5% over the next month, she could accomplish this goal by buying 100 contracts (each with a 1,000-barrel multiplier) of the \$95 strike put. Given a contract price of \$1, if oil drops to \$90 over the next month, the producer will lose \$1M from the initial expected \$10M value of the barrels at the beginning of the month, but gain $(95 - 90 - 1) * 100 * 1000 = 400000$ dollars from the put position, thus reducing her total losses.

Options speculation, in essence, is a highly leveraged bet on the direction and/or magnitude of an underlying asset move. Leverage here refers to the ability to multiply buying or

selling power and is proportionally related to risk [12]. An investor who believes that Apple stock, currently valued at $100, will double within the next 6 months, could make one of two hypothetical trades: buy 1 share of Apple stock for $100, or buy 20 call options with a strike price of $150 for $5 each (assume no multiplier for simplicity). If Apple is valued at $200 in 6 months, the options position would gain a total of $((200-150)*20-100 = \$900$ for a total gain of 900% compared to "just" a 100% gain on the stock position. On the other hand, if Apple is valued at $150 in 6 months, the options position expires worthless for a total loss of 100% compared to a total gain of 50% for the stock position. This scenario emphasizes the high-risk, high-reward nature of directional options speculation.

Speculating on the magnitude–but not direction–of an underlying asset move is called volatility speculation. A common instrument used in volatility speculation is the at-the-money straddle. In general, a straddle involves buying both a call and put at the same strike price., and at-the-money (ATM) refers to the strike price of the options relative to the current underlying asset price–in this case meaning the strike price and current underlying price are roughly equivalent. An option is considered "in-the-money" if, given an immediate exercise the option would hold value and "out-of-the-money" if not.

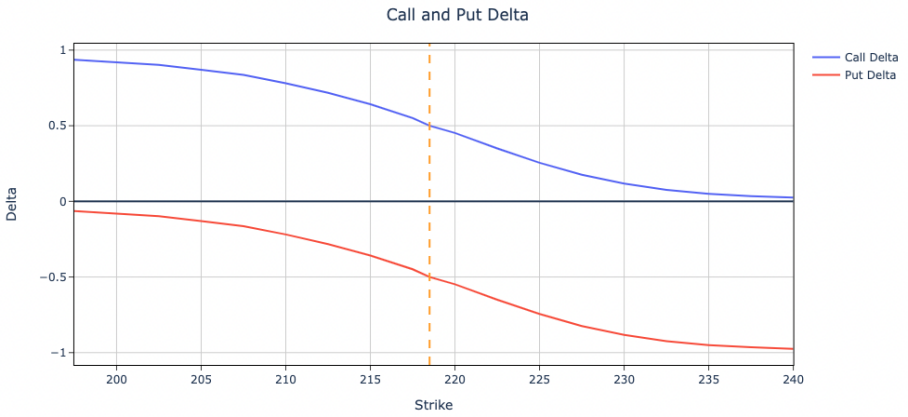

Figure 1.1: Option Strike Price vs. Delta

The reason the ATM straddle price carries very little underlying directional risk, is based on the concept of delta. Delta is defined as the change in option price given a $1 change in the underlying asset price. The absolute value of delta can also be thought of as the

percent probability of an option finishing "in-the-money" and ranges from 0 to 1. Figure 1.1 plots option deltas for different strike prices of both calls and puts for an underlying asset currently priced at $ 218.50 (orange dotted line).

Note that all calls have positive delta, meaning they increase in price when the underlying asset increases in price, and all puts have negative delta, meaning they increase in price when the underlying asset decreases in price. Another important property of delta is that ATM options have an absolute delta value of 0.50, meaning the ATM straddle has a delta value near zero. Therefore, its price is largely unaffected by moves in the underlying asset–so long as its strike price remains close to the current underlying asset price.

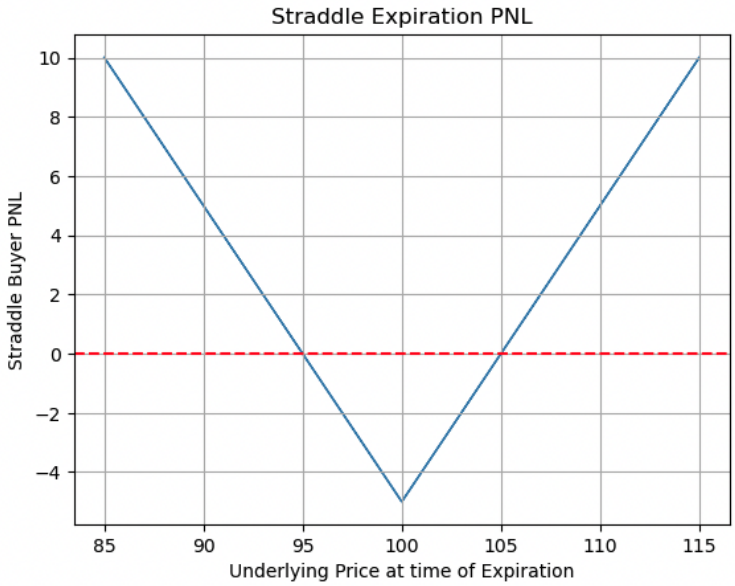

Figure 1.2: Straddle Expiration PNL

Figure 1.2 represents the PNL at time of expiration of an at-the-money straddle with a strike price $100, and a cost of $5. If the underlying asset moves $5 in either direction, then both the buyer and seller of the straddle make $0 and are said to break-even. If the underlying asset moves less than $5, the buyer loses (seller gains) $(|100 - price_{exp}| - 5)$ dollars, with a max loss (gain) of $5. If the underlying asset moves greater than $5, the buyer gains (seller loses) $(|100 - price_{exp}| - 5)$ dollars, and has hypothetically unlimited upside (downside). The figure reiterates that the sole determinant of the ATM straddle

PNL at the time of expiration is the size of the underlying price move.

The proposed Options Portfolio attempts to generate excess risk-adjusted returns through volatility speculation, using daily changes in implied volatility as a proxy for daily straddle returns. Volatility is synonymous with the annualized standard deviation of underlying asset returns, and implied volatility therefore refers to the expected underlying standard deviation of returns through the length of the options contract. To illustrate the relationship between percent change in implied volatility and percent change in straddle price, the components that determine the straddle price for non-expiring straddles must be examined. A straddle price approximation formula derived from the Black-Scholes presents these components clearly [11][33]:

$$Straddle_{ATM} = 0.8S\sigma\sqrt{T} \tag{1.3}$$

Here S represents the current asset price, called the "spot" price, σ represents annualized implied volatility, and T represents time until expiration, in years. Intuitively, each of these three relationships is logical: 1) higher priced assets have higher priced options, holding implied volatility and time until expiration equal 2) higher implied volatility means a higher probability of large underlying asset moves and 3) more time until expiration means more time for a large underlying asset move to occur.

Now that the relationship between straddle price and implied volatility has been established, the relationship between the one-day straddle return and the one-day percent change in implied volatility must be considered. The one aspect of the straddle price that is guaranteed to change is time until expiration T. Given one day passing, $T_1 = T_0 - \frac{1}{365}$, and the one-day percentage straddle return due to time can be expressed as:

$$R_{Time} = \frac{\sqrt{T_1} - \sqrt{T_0}}{\sqrt{T_0}} = \sqrt{\frac{T_1}{T_0}} - 1 \tag{1.4}$$

Therefore, the one-day straddle return due to time is always negative. This is the basis of what is known as theta, or time-decay of an option. The counterbalance to theta is gamma,

defined as the change in option delta given a $1 move in the underlying asset. Long gamma positions become longer (shorter) delta on an upward (downward) underlying move, and thus the straddle return due to gamma is always positive. While the straddle approximation formula does not allow for direct calculation of the straddle return from gamma, theoretically, the gamma-theta trade-off is a zero-sum-game, and empirical evidence has done little to reject this notion [29]. Given the large sample used in this study, an assumption of negligible average PNL from gamma and theta is made. Under this assumption, the average one-day percent return of a long ATM straddle position can be approximated by the one-day percent change in implied volatility:

$$R(Straddle) \approx R(Gamma) + R(Theta) + \%\Delta(IV) \approx \%\Delta(IV) \tag{1.5}$$

1.3 The Options Portfolio

Combining portfolio optimization and options theory is at the heart of the proof-of-concept Options Portfolio. The literature on inter-asset options strategies is virtually non-existent, with an extensive search yielding very little discussion on the topic. In industry, inter-asset strategies are certainly being used, though the extent and complexity of such strategies is uncertain. The options portfolio can best be understood through a stock portfolio where short selling is allowed and the sum of the absolute values of individual asset weights equals one.

The long-short stock portfolio generates return in two ways: net exposure and individual asset selection. Net exposure refers to the current net position of the portfolio i.e., whether the sum of all portfolio positions is greater than (long exposure) or less than (short exposure) zero.

Let n represent the total number of assets contained in the long-short portfolio and $w_{i,t}$ represent the portfolio weight of asset i at time t. Exposure is then simply calculated by summing over all asset weights:

$$Exposure_t = \Sigma_{i=1}^{n} w_{i,t} \tag{1.6}$$

Subject to constraint:

$$\Sigma_{i=1}^{n} |w_{i,t}| = 1$$

Positive return from exposure is generated when either exposure is positive and the average price of portfolio stocks goes up, or when exposure is negative, and the average price of portfolio stocks goes down. This idea can be easily translated to the Options Portfolio, where exposure refers to the net implied volatility position rather than the net stock position. Positive return from exposure for the Options Portfolio is generated when either implied volatility exposure is positive and average IV goes up, or IV exposure is negative and average IV goes down. Therefore, being able to time long exposure prior to volatility rising, and short exposure prior to volatility falling is critical to overall portfolio performance.

The second source of return for both the long-short portfolio and the Options Portfolio is individual asset selection, which is the process of determining which assets are selected, and in what proportions, to achieve the targeted portfolio exposure. For the Options Portfolio, the goal of asset selection is to enter short straddle positions in assets where future implied volatility is deemed likely to fall and long straddle positions in assets where future implied volatility is deemed likely to rise. While exposure is bounded between -1 and 1, the range of combinations in individual asset selection is much more expansive. To profit from individual asset selection, long-term dependencies between portfolio assets must be uncovered and shifts in these dependencies must be quickly adapted.

The recent ascent of deep learning models stems from their ability to uncover such complex relationships. Deep learning models using Sharpe Maximization have proved highly capable when applied to various stock and ETF portfolios, including those allowing short selling. These models will therefore be used to create the Options Portfolio. Of particular

interest is the Attention Transformer model which has shown drastically improved capability over LSTM and GRU models in time series applications.

CHAPTER 2

Deep Learning Overview

2.1 Introduction to Deep Learning

Deep learning is a branch of machine learning where data is passed through various interconnected algorithmic layers that together are designed to roughly imitate human thinking. Deep learning models can extract complex insights from raw data and are prominently featured in facial recognition software, self-driving cars, and the recently launched ChatGPT, which has sent shockwaves throughout the technological landscape.

A deep learning model is typically comprised of an input layer, multiple hidden layers and an output layer. Input layer data can be of any shape or type and is typically fed to the model in batches containing several observations. Each input batch passes through multiple hidden layers until the model generates an output that could be as simple as an economic forecast or as complex as a self-driving car decision. The flow of data from input to output is called the forward process, and in basic feed-forward networks, the output of hidden layer h_l can be computed as [25]:

$$h_l = Y_l(s_l), \quad where \;\; s_l = w_l h_{l-1} + b_l \tag{2.1}$$

The score vector s_l is generated by multiplying the layer weight matrix w_l, with the vector output from the previous layer h_{l-1}, and summing the result with a bias term b_l. Y_l represents a non-linear activation function that is applied uniformly to each element in the score vector and provides added flexibility to the model. Input data can be denoted as the first model layer h_0, while the output layer can be denoted as h_L. Often a transformation

or series of transformations is applied to the raw output-layer score vector s_L to determine a final predicted value $\hat{y}$. Every aspect of a deep-learning model is pre-specified except for its weights and biases, which are typically randomly initialized. Models therefore must be trained by adjusting layer weights and biases in a way that minimizes model error.

Model error, or loss, is defined by the criterium of a selected loss function, and typically involves comparing predicted values $\hat{y}$ to actual target values y. For example, given a training batch of images in a classification problem, error would be attributed to any images that are misclassified. The most common loss functions include mean squared error for regression problems, and cross-entropy loss for classification problems.

Back propagation is an algorithm that uses chain rule differentiation to calculate the slope, or gradient, of the loss function with respect to the weight and bias parameters in each layer. Parameters are then iteratively adjusted against their computed gradients with the goal of minimizing model loss. Beginning with total mini batch loss, back propagation moves backwards through model layers while calculating parameter gradients at each layer. For a given layer, the back propagation algorithm first calculates the loss gradient with respect to the hidden layer output [26]:

$$\frac{\partial L}{\partial h_{l-1}^T} = \frac{\partial L}{\partial h_l^T} Y_l' \odot w_l \tag{2.2}$$

Here Y_l' represents the derivative of the layer l activation function and is applied to each element of weight matrix w_l. The top-layer loss gradient, $\frac{\partial L}{\partial h_L^T}$ is equal to the prior mini batch error, e and is the first input in the backward process. Once the loss gradient with respect to layer h_{l-1} is calculated, the loss gradient with respect to layer weights is derived via chain rule as:

$$\frac{\partial L}{\partial w_{l-1}} = Y_{l-1}' \odot \frac{\partial L}{\partial h_{l-1}} h_{l-2} \tag{2.3}$$

while the loss gradient with respect to the layer bias term is derived as:

$$\frac{\partial L}{\partial b_{l-1}} = Y'_{l-1} \odot \frac{\partial L}{\partial h_{l-1}} \tag{2.4}$$

Once every gradient in the mini batch is computed, parameters are updated in a process called stochastic gradient descent (SGD). In SGD, previous parameters are updated by moving down their respective gradients, and a learning rate is utilized to determine the magnitude of the adjustment at each iteration [15].

$$\theta_{t+1} = \theta_t - \eta_t \frac{1}{m} \Sigma_{i=1}^{m} \frac{\partial Loss_i}{\partial \theta_t}, \quad \theta = (w_l, b_l, l = 1, \ldots, L) \tag{2.5}$$

Here, η_t represents the learning rate at time t and m represents the size of each mini batch of data. After every parameter is updated, the next batch of inputs are run through the forward process using the updated weights and biases, and the forward-backward loop is repeated until average loss is no longer being significantly reduced.

2.2 RNN

A substantial development occurred in 1982 when Hopfield built the first recurrent neural network (RNN), which introduced a memory-like component into deep learning models that could be used for time- or sequence-dependent problems [3]. In RNNs, hidden layer state h_t takes input not only from current data input x_t, but also from its previous layer state, h_{t-1} [14]:

$$h_t = Y_t \left(w \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \tag{2.6}$$

Crucially, hidden layer state h_t is not equivalent to hidden layer h_l used in the feed-forward networks above. While h_{l-1} represents the previous hidden layer in a feed-forward network, h_{t-1} represents a prior state of h_t and in fact, can represent any number of hidden

layers occurring at time t-1. Since each hidden layer state takes input from its previous state, back propagation for RNNs must be calculated not only through each hidden layer, but also through each prior time period [18]. The many repetitive weight matrix multiplications that occur during RNN back propagation tend to cause the computed gradients to either approach infinity and "explode", or more commonly, approach 0 and "vanish". When this happens, gradient descent is no longer viable, and the model cannot be efficiently trained. The development of the LSTM model–a sub-class of RNN–in 1997 was a substantial innovation in deep learning research, as it solved the vanishing gradient problem [10].

2.3 LSTM

LSTM uses various gate and update vectors and the concept of selective memory to stabilize parameter gradients in back propagation. At each time t, the vanilla LSTM initially computes four vectors: forget gate (f_t), input gate (i_t), output gate (o_t), and candidate memory (Δc_t), that are each calculated as [13]:

$$g_t = Y_t \left(w \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \quad for\ g_t = [f_t, i_t, o_t, \Delta c_t] \tag{2.7}$$

Here, weight w is unique for each of the four vectors, the sigmoid activation function is used to compute the three gate vectors, and the tanh activation function is used to compute Δc_t. These vectors are then used to calculate the memory cell vector c_t, that stores the long-term memory of the model, as well as the current hidden layer state vector h_t:

$$c_t = f_t \odot c_{t-1} + i_t \odot \Delta c_t, \quad h_t = tanh(o_t \odot c_t) \tag{2.8}$$

The key to solving the vanishing gradient problem lies within the structure of the memory cell gradient, which can be derived as [1]:

$$\begin{aligned}\frac{\partial c_t}{\partial c_{t-1}} &= \frac{\partial c_t}{\partial c_{t-1}}[c_{t-1} \odot f_t + i_t \odot \Delta c_t] \\ &= f_t + \frac{\partial c_t}{\partial c_{t-1}}[i_t \odot \Delta c_t] \\ &= f_t + A_t\end{aligned}$$

This implies that so long as the forget gate f_t, which decides whether to discard or keep long-term memory, is non-zero, a gradient can be calculated [3].

2.4 GRU

Gated Recurrent Unit (GRU) was designed by Cho, et al. in 2014 as a computationally faster adaptation of LSTM [28]. While GRU is technically less complex than LSTM, there is debate over which model performs better. GRU uses two gates r_t and z_t which are computed similarly to the LSTM gate vectors, and one update vector $\tilde{h_t}$ that is calculated as: [4]

$$\tilde{h_t} = r_t \odot h_{t-1} \tag{2.9}$$

Instead of using an intermediary memory cell vector, GRU uses the update vector to directly compute the hidden layer state vector h_t, while z_t acts similarly to the forget gate by determining whether or not to replace the previous layer state h_{t-1}:

$$h_t = z_t \odot \tilde{h_t} + (1 - z_t) \odot h_{t-1} \tag{2.10}$$

Although LSTM and GRU are powerful models that are still widely prevalent, they are limited by the fact that they can only pass information through hidden layer states sequentially. The current state of the art Attention Transformer model on the other hand, does not suffer from this constraint.

2.5 Attention Transformer

In 2014, Cho et al. developed an encoder-decoder model architecture featuring two jointly trained RNNs used in a language translation problem [4]. In their model, the encoder RNN takes input from each word sequentially, and utilizes information from the previous hidden layer state h_{t-1}. The decoder RNN then takes the encoder output as its initial input, and sequentially outputs the phrase, word-by-word, in another language. Although this model was state of the art at the time, the sequential restriction of LSTM limited its ability to contextualize phrases.

Bahdanau et al. introduced the first attention mechanism in language translation problems with a bi-directional RNN encoder that output distinct context vectors fed directly to each decoder layer [2]. Subsequent translation models began including attention mechanisms within various RNN architectures, resulting in iteratively improved performance. In 2017 however, Vaswani et al. completed discarded the RNN, and built an attention-only Transformer model that dominated prior models in terms of both speed and performance [31].

The distinguishing feature of the Attention Transformer is its use of multi-head self-attention, rather than recurrence. In a self-attention framework, each hidden layer state h_m interacts directly with every other layer regardless of sequential position. For each vector h_m, distinct learned weight parameters are used to compute query (q_m), key (k_m), and value vectors (v_m), which are then used to calculate affinity scores for each m-t vector pair [31]:

$$Affinity(h_m, h_t) = \frac{< q_m, k_t >}{\sqrt{d_k}} \tag{2.11}$$

Note that each query and key vector share the same dimension d_k, which is used to scale the dot product so that a near-zero gradient is avoided. A softmax activation function is applied to affinity values which yields the attention value $A_{m,t}$:

$$A_{m,t} = \frac{e^{Affinity(h_m, h_t)}}{\Sigma_{t'=1}^{M} e^{Affinity(h_m, h_{t'})}}, \quad \Sigma_{t=1}^{M} A_{m,t} = 1 \tag{2.12}$$

The set of $A_{m,t}$ can be interpreted as weights for each vector pair over which a weighted average vector Δm is then computed:

$$\Delta m = \Sigma_{t=1}^{m} A_{m,t} v_t \tag{2.13}$$

The Transformer model uses what is known as multi-head attention in its encoder and decoder layers. The idea behind using multiple attention "heads" is that each head focuses on a different contextual aspect of the sequential data. The single-head mechanism described above can easily be expanded to a multi-head mechanism. In multi-head attention, query, key and value vectors are generated for each state vector h_m, and each specific $head_j$, and are denoted as $q_m^{(j)}, k_m^{(j)}, v_m^{(j)}$. Importantly, these vectors only interact with vectors generated from the the same $head_j$ of all other hidden layer states h_t. Therefore multi-head attention featuring n heads generates n different update vectors, $\Delta m^{(j)}$, which are then concatenated together as follows:

$$\Delta m = \begin{pmatrix} \Delta m^{(1)} \\ \vdots \\ \Delta m^{(j)} \\ \vdots \\ \Delta m^{(n)} \end{pmatrix} \tag{2.14}$$

CHAPTER 3

Methodology

3.1 Model Architecture

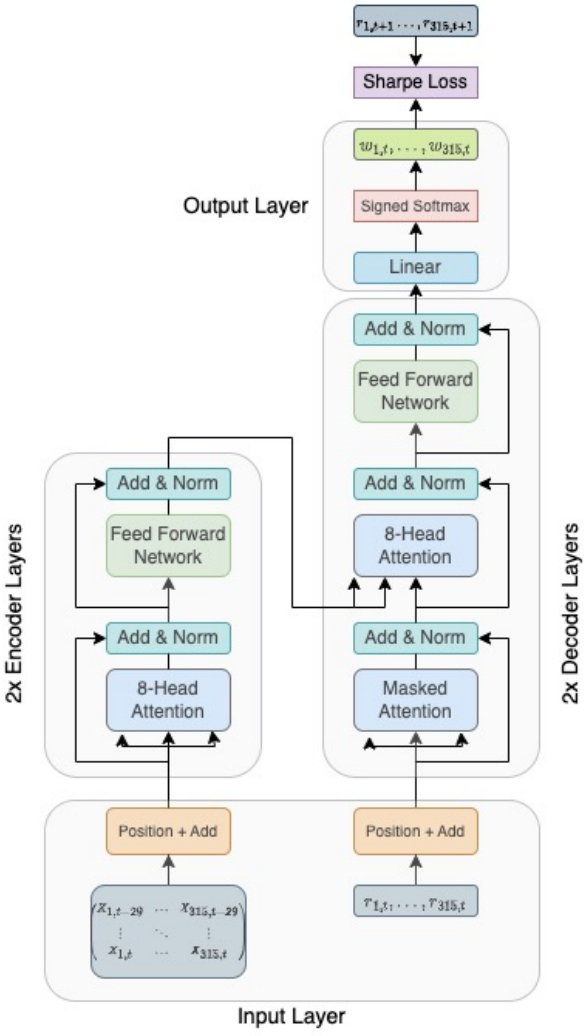

Figure 3.1: Attention Transformer Options Portfolio Architecture

Despite being developed for language translation problems, the Attention Transformer

has since been successfully applied to many sequence-dependent tasks including financial time series [32][17]. Therefore, the model architecture of the Attention Transformer Options Portfolio, displayed in Figure 3.1, is very similar to the one used in the original Transformer paper.

3.1.1 Input Layer

Daily input data consists of 10 unique features including end-of-day implied options volatility, implied options skew, and various underlying asset metrics. A look-back period of 30 trading days is used such that each individual observation is a tensor of dimension (30, 3150). Since Inputs are fed through the model in mini-batches, and given a chosen batch size of 32, the total input dimension is (32, 30, 3150).

One critical element that self-attention does not innately account for is the position of each hidden state vector h_m. Therefore, positions must be encoded prior to entering multi-head attention. The options portfolio follows the positional encoding methodology outlined in [31], which utilizes sine and cosine functions of different frequencies. The positional embedding is added to the original embedding and the resulting vector is passed to the encoder layer.

3.1.2 Encoder

The encoder contains two identical layers, each with two sub-layers: 8-head attention, and a feed forward network. The feed forward network used here is a single-layer fully connected network with 2048 neurons, relu activation and a dropout rate of 0.2. Dropout is a form of regularization where dropout rate*N different neurons are randomly deactivated for each pass through the network.

Each sub-layer output is summed with the original layer vector h_m and then normalized to mean 0 and standard deviation 1 before being passed through the next step of the forward pass. This "add and norm" process can be shown as [31]:

$$h_m = LayerNorm(h_m + w\Delta m) \tag{3.1}$$

For the attention sub-layer, Δm refers to the concatenated update vector output by the 8 attention heads as described in 2.14. For the feed-forward network, Δm simply refers to the output layer of the network, while w is a unique learned parameter for each sub-layer. After passing through both the attention and feed forward layers twice each, the resulting vectors move to the decoder layer.

3.1.3 Decoder

The decoder also contains two identical layers, with the addition of a third sub-layer called masked attention. The decoder takes input from the prior-day implied volatility returns, denoted as $r_{i,t}$ for each observation in the mini batch, and positionally encodes them. The embedded vectors then pass through to the masked attention sub-layer. Unlike the encoder layer, the decoder layer is autoregressive meaning it must output predicted returns, and subsequently weights, sequentially. The idea of masked attention is that input vectors can interact with previous return vectors, but not with any future return vectors.

After passing through masked attention and an add & norm layer, the vectors pass to a multi-head attention layer where they attend with the vectors output by the encoder, before entering another normalize layer. Finally, the resulting vectors pass through a feed-forward network identical to the one used in the encoder and go through one last add & norm layer. The full decoder layer is then repeated, and the final decoder output passes to the output layer.

3.1.4 Output Layer

The output layer is comprised of a fully connected linear layer and a final transformation layer used to implement portfolio constraints. The linear layer takes the final decoder output and transforms it into 315 portfolio scores, $s_{i,t}$. As previously stated, the absolute value of

the Option Portfolio asset weights must always sum to 1. To implement this constraint into the model, a variation of the softmax function, originally devised by Zhang et al [34] is applied to the scores $s_{i,t}$ to compute the final portfolio weights for each asset, $w_{i,t}$:

$$w_{i,t} = sign(s_{i,t}) * \frac{e^{s_{i,t}}}{\Sigma_{j=1}^{N} e^{s_{j,t}}} \tag{3.2}$$

3.2 Loss Function

Deriving the Sharpe Loss function begins by defining portfolio return. The portfolio return over time t, $R_{P,t}$, is calculated by multiplying the asset weights from the previous period with the asset returns from the current period t, and summing over n assets:

$$R_{P,t} = \Sigma_{i=1}^{n} w_{i,t-1} r_{i,t} \tag{3.3}$$

From this equation, the expected portfolio return over time period T can be calculated as:

$$\mathbb{E}(R_{P,T}) = \frac{1}{T} \Sigma_{i=1}^{T} R_{P,t} \tag{3.4}$$

and the Sharpe Ratio, introduced in 1.2, can be parameterized over time T as follows [35]:

$$S_T = \frac{\mathbb{E}(R_{P,T})}{\sqrt{\mathbb{E}(R_{P,T}^2) - (\mathbb{E}(R_{P,T}))^2}} \tag{3.5}$$

Although alternative performance metrics such as the Sortino Ratio, which only includes downside standard deviations, or the Calmar ratio, which measures return over maximum peak-to-trough portfolio decline, are often preferred to the Sharpe Ratio in industry, the

Sharpe Ratio is the only one of the three that is easily differentiated. Starting from the result of 3.5, Molina provides a thorough walkthrough of Sharpe Ratio maximization using backward propagation and gradient ascent [21]. Once the gradient of the Sharpe loss function with respect to model parameters θ is computed, model parameters are updated as follows:

$$\theta_{t_1} = \theta_{t_0} + \alpha * \frac{\partial S_{t_1}}{\partial \theta} \tag{3.6}$$

3.3 Training Methodology

Training the model is initialized by splitting the data into training, validation and testing sets, with the initial training set spanning five years (2008-2012). Each data observation contains the past 30 trading days' worth of information and is used to select optimal portfolio weights for the one-day-forward time period. An expanding window training approach is implemented where the model is re-trained yearly on an iteratively growing training set. Model parameters are tuned using a validation set comprised of the first year of data following the end of each training set, before the tuned model is tested on a subsequent year of unseen test data.

Training is done using the Adam Optimizer, which utilizes both adaption and momentum to quickly and efficiently implement stochastic gradient descent to minimize the loss function within each mini batch of data [16]. Final hyper-parameters include a mini batch size of 32 and learning rate of 10E-5, while using 100 epochs to train each unique training set. Additionally, attention-specific parameters are tuned, yielding 8 attention heads and a dimension value of 2048. In total, the testing period includes nine years of data from 2014 to 2022. Models are developed in python's PyTorch module and trained using a NVIDIA P100-SXM 16GB GPU with 40 GB of RAM memory.

CHAPTER 4

Data and Results

4.1 Dataset Description

Data used in this study is primarily sourced from the Nasdaq U.S. Equity Historical & Option Implied Volatilities (VOL) data feed, which contains end-of-day volatility data for publicly traded stocks and ETFs [24]. The data includes implied volatility for at-the-money options contracts that expire nearest to various specified days until expiration. For example, the 60-day implied volatility data refers to the IV of the listed options contracts that expire closest to 60 days from the date of observation. Equity options contracts are listed monthly, implying that data used for the 60-day timeframe comes from options that span between 45 and 75 calendar days until expiration. Therefore, on one day each month the IV data "rolls" from the 45-day-until-expiration option to the 75-day-until-expiration option, and the measured return from an implied volatility position will incorrectly reference two separate assets. While this discrepancy must be acknowledged, these datapoints comprise of less than five percent of the total data. Additionally, over the entire 15-year dataset, the mean 60-day and 90-day volatilities are within 0.009 percentage points of each other, making it difficult for a model to take significant advantage of.

In addition to the at-the-money data, the dataset includes "skew" data, which measures the IV ratio of out-of-the-money puts to out-of-the-money calls, and realized volatility data, which measures the actual standard deviations of underlying asset returns. The data was augmented with additional underlying asset data from the AlphaVantage Daily Time Series API including daily returns, volume and range [30]. The 60-day expiring data was preferred to the 30-day expiring data due to a lower price impact from gamma-theta and preferred

to the 90-day period due to higher liquidity. Assets with high liquidity are those with high trading volumes and tight bid-ask spreads, and therefore have lower execution costs.

The process of selecting the asset pool began with all current components of both the S&P 500 and Nasdaq 100 indices, which contain highly liquid options markets. This initial set of stocks was expanded with a variety of ETFs including index, commodity, volatility, and real-estate related funds, to create a more diverse pool. The final criterium for inclusion was complete, or near-complete options data dating back to at least January 1, 2008. The 15-year span of data ensured a lengthy testing period while the inclusion of 2008 in model training was needed to expose models to the most adverse of market conditions [6]. In total, model portfolios took daily implied volatility positions in nearest-to-60-day-expiring ATM options referencing 315 different underlying assets.

4.2 Baselines

Three models were used as baselines to compare the Attention Transformer Options Portfolio to. In addition to the previously mentioned LSTM and GRU models, an equal-weighted IV portfolio was also included, which continuously holds implied volatility positions of size $\frac{1}{315}$ in each asset. In an actual trading environment, the major advantage of the equal-weighted portfolio is that its positions are only adjusted monthly, on the roll date, as opposed to daily updates for the other three models. Options trading costs are both substantial and difficult to estimate, particularly with indirect costs such as execution costs. Since the purpose of this study is to showcase the predictive power of the tested models on the dataset rather than to create an actual profitable trading strategy, trading costs are ignored.

4.3 Results

The AT Options Portfolio results are compared to the three baseline models using various annualized financial metrics. These metrics include the Sharpe Ratio and its components, the previously mentioned Sortino Ratio, calculated as $Sortino = \frac{R_p}{\sigma_{R_{downside}}}$, Max-Drawdown

(MDD), calculated as the mean yearly maximum portfolio peak-to-trough decline, and the Calmar ratio, calculated as $Calmar = \frac{R_p}{MDD}$.

	Return	Annualized SD	Sharpe Ratio	Sortino Ratio	Max DD	Calmar Ratio
EW Baseline	0.301	0.499	0.530	1.027	-0.335	0.851
LSTM	0.389	**0.082**	5.069	7.321	-0.049	15.359
GRU	0.514	0.083	6.413	9.452	-0.049	18.634
Attn Transformer	**0.865**	0.092	**10.755**	**15.514**	**-0.044**	**37.731**

Table 4.1: Model Performance

The AT Options Portfolio outperforms the three baseline models in nearly every category. While the GRU and LSTM portfolios have slightly lower variance, this is more than offset by the substantially higher returns produced by the Options Portfolio. Impressively, the AT Options Portfolio has both the lowest average max-drawdown and the highest annual return, to yield a Calmar Ratio–the preferred metric for hedge-fund performance–more than double that of the GRU model, and 44 times higher than the equal-weighted portfolio.

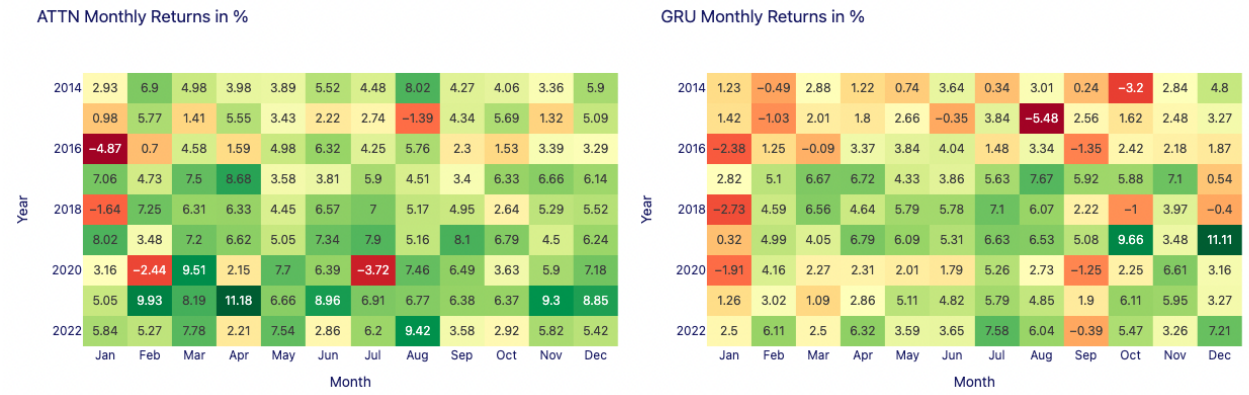

Figure 4.1: AT vs GRU Monthly Returns

Figures 4.1 highlights the robustness of the AT Options Portfolio to changing market regimes. Over 108 months, including numerous periods of substantial market turmoil, the AT Options Portfolio generates a positive monthly return in all but 5, as compared to the next-best GRU model which has 14 months of negative performance. In 2020, when the Options Portfolio performs its relative worse, it still produces a very-respectable Sharpe Ratio of 3.47, compared to a worst-case 1.67 for the GRU model and 0.73 for the LSTM

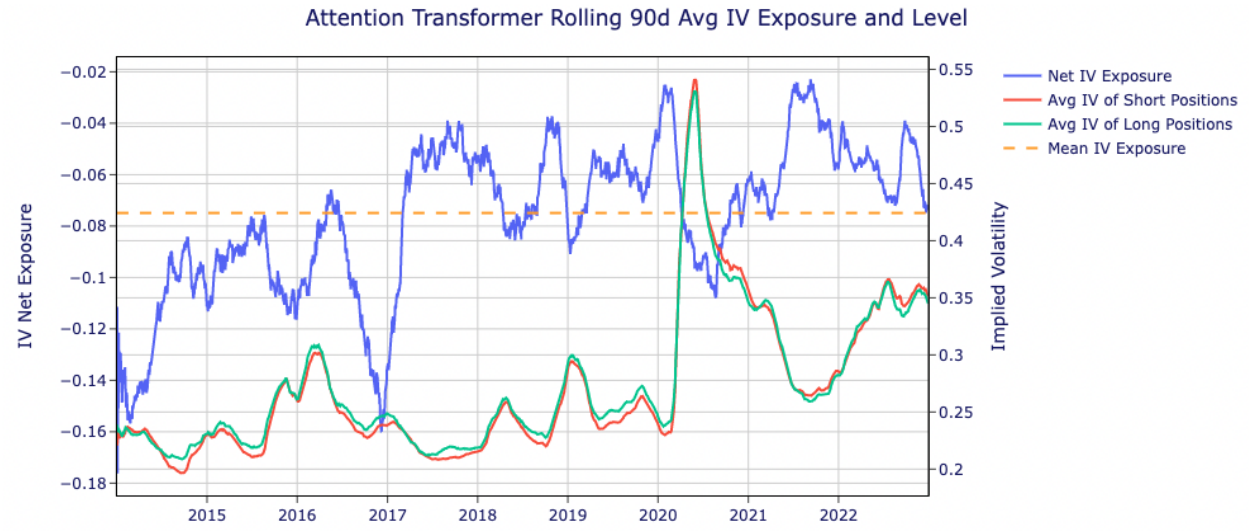

Figure 4.2: IV Exposure vs IV Level

model.

Figure 4.2 provides significant insight into how the AT Options Portfolio generates outsized risk-adjusted return. Rolling average IV exposure is displayed in blue, referencing the left-hand y-axis, while average raw IV of both long (green) and short (red) positions reference the right-hand y-axis. Crucially, the model holds an average net IV exposure of just -0.075 (orange dotted line) over the span of the nine-year testing period, meaning it is generally only minimally exposed to market-wide implied volatility shifts. For reference, given a benchmark equal-weight portfolio constantly holding this same -0.075 net exposure, and the fact that IV on average went up 30%, said benchmark would lose about 2.3% annually.

Despite average volatility positioning in the "wrong direction", the AT Options Portfolio still returns nearly triple that of the equal-weighted portfolio on average. The first thing this indicates is that the model performs exceptionally well from an asset-selection standpoint. The model is clearly able to consistently identify positions with high expected value of return in both long and short positions. Interestingly, the asset-selection is far more complex than taking long positions in assets that have relatively low implied volatility, and short positions in assets that have relatively high implied volatility. In fact, on average, the mean implied volatility of portfolio long positions is actually *higher* than the mean implied volatility of

portfolio short positions. The difference between the red and green lines in Figure 4.2 emphasizes this point, with the green line being, on average, above the red line. The AT Options Portfolio is therefore able to identify much more complex relationships between asset implied volatilities.

The second thing this indicates is that the Options Portfolio is incredibly adaptive and excellent at getting ahead of market volatility trends. This is most clearly demonstrated between the years 2020-2021. During the global pandemic in 2020, volatility began to sky-rocket, as shown in Figure 4.2. Although the AT Option Portfolio's average exposure is still short, the portfolio reduces average exposure to -0.03 which is its smallest rolling average exposure up to that point in time. While the portfolio still loses money from the net short exposure, the decision to get close to net-zero exposure considerably protects the portfolio value. Then, as IV continues to rise towards its peak, the AT Options Portfolio increases short exposure to -0.11, and substantially profits when volatility craters back down to 0.35. In 2021, with volatility at its relative low point, the AT Options Portfolio again takes a near-flat IV exposure position and is protected when average IV rises from 0.27 to 0.37. Overall, the AT Options Portfolio leverages its ability to both identify complex inter-asset relationships and quickly adapt to unforeseen market conditions to generate huge returns with relatively low risk exposure.

CHAPTER 5

Further Research and Conclusion

5.1 Further Research

The Attention Transformer Options Portfolio clearly demonstrates the power of applying state of the art deep learning models to inter-asset options strategies, but there still exists a gap from proof-of-concept to production. This section discusses potential future research approaches aimed at closing this gap.

The first issue involves testing the theoretical assumption that change in implied volatility is an adequate proxy for daily straddle returns. The primary reason for training the model using option implied volatility data instead of option price data is that the cost of acquiring price data that spans the timeframe needed to train a deep learning model is substantial. Testing the implied volatility assumption is fortunately more manageable, as options price data spanning a short period of time is more easily acquired. If back tests using actual price data and model weights yield high-performance results, then the model can confidently use implied volatility data for future positional recommendations. Otherwise, the performance discrepancy must be investigated, and more data may need to be obtained.

The second issue involves implementing realistic trading costs into the model. Trading costs can be divided into two categories: direct costs and execution costs. Direct costs, such as brokerage and exchange costs, are typically predictable and are relatively stable over time. Execution costs are related to liquidity and can be thought of as the difference between the fair price of an asset and the price a market participant can execute a trade at. Estimating execution costs is difficult due to variance in both single-asset and market-wide liquidity. For example, in normal market conditions, a bid-ask spread for an Apple straddle may be

50 cents, meaning a market participant wanting to execute a trade can do so at a price just 25 cents from the middle of the market. In a high-volatility, low-liquidity environment, that same straddle may have a bid-ask spread of $5, meaning a market participant would now incur an execution cost of $2.50, which is ten times higher than normal.

Portfolio costs stem largely from asset turnover which accounts for both the quantity and magnitude of daily portfolio asset weight adjustments. Single-period portfolio turnover can be expressed as:

$$Turnover = |(1 + R_{p_1})\Sigma_{i=1}^{n} w_{i,1} - \Sigma_{i=1}^{n}(1 + r_{i1})w_{i,0}| \tag{5.1}$$

Here, R_{p_1} represents total portfolio return from $t = 0$ to $t = 1$, r_{i1} represents return of asset i over the same time period, and $w_{i,t}$ represents the portfolio position weight in asset i at time t. The AT Options Portfolio itself has an average asset turnover of approximately 1.1 meaning that over 100% of the entire portfolio value is being reallocated each day. Despite impressive zero-cost performance, the Options Portfolio as-is would likely not produce sustainable returns in a real trading environment due to significant average turnover. To solve this issue, various cost-reducing approaches can be examined using back testing and direct model integration.

One cost-reducing strategy that is easily back tested is cardinality, which involves restricting the number of total assets the portfolio can invest in each day. Keeping daily net IV exposure the same as the full model, back tests were run where portfolios took positions in the $\frac{n}{2}$ most heavily weighted long and short assets of the full model for each day.

Figure 5.1 shows back test results with Sharpe Ratio (blue) plotted against both return (orange) and annualized standard deviation (green), for various max number of daily options positions. Results are promising in that a portfolio with less than 20 daily positions achieves a higher Sharpe Ratio than the full portfolio. Any costs assessed on a per-transaction basis would be substantially reduced using this approach, and asset turnover would likely fall as well, given the restriction on possible portfolio weight combinations.

Other cost-reducing methods that could be investigated include shrinking the size of the

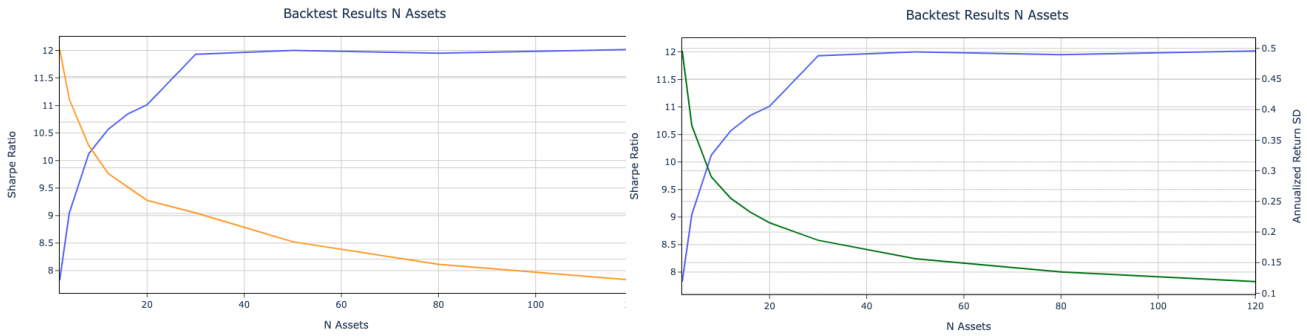

Figure 5.1: N-asset Backtest Results

overall asset pool and implementing a longer holding period. Shrinking the potential asset pool lowers turnover by reducing the combination of possible portfolio positions and can best be accomplished by identifying and eliminating highly correlated assets. One drawback of this approach is that, given the non-stationary nature of asset correlations, it risks elimination of an asset that would provide valuable diversification in the future. A longer positional holding period guarantees substantial reduction in turnover, but efficient implementation of such a constraint is complicated. Multi-day holding periods are considerably riskier than one-day holding periods, and therefore profit-taking and stop-loss decision rules would need to be built into the model.

After using back tests to identify promising cost-reducing constraints, these constraints can be layered into the AT Options Portfolio model structure, and the model can be re-trained. Estimated trading costs can also be directly applied within the model structure using a similar method as Kisiel et al., but with the added caveat of making them variable instead of constant [17]. One possible way to incorporate non-constant costs is to train a second model that predicts costs based on various market- and asset-level conditions. Once costs can be accurately estimated, the primary model itself can select the optimal way to maximize after-cost performance. If this model is still able to generate out-sized returns, then its recommendations can be used to make actual trading decisions.

5.2 Conclusion

This work expands on prior implementations of end-to-end and attention transformer models within stock portfolio optimization, to develop a novel inter-asset option strategy. Little, if any public research exists on even simplistic inter-asset options strategies, and although such strategies are being executed to some degree in industry, it is unlikely that any are as robust as the Attention Transformer Options Portfolio.

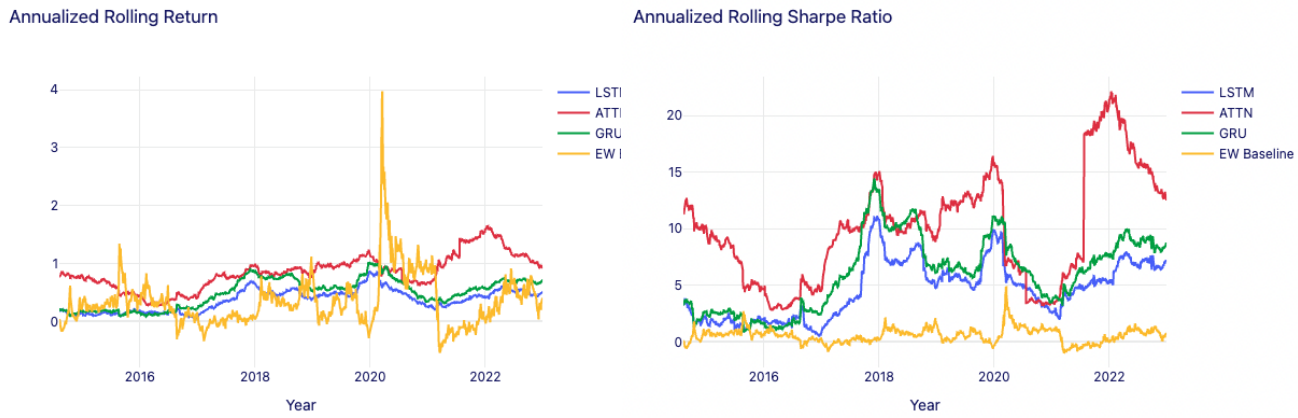

Figure 5.2: Portfolio Performances Through Time

Before-cost risk-adjusted returns are significantly higher for each deep learning options portfolio than for any known prior end-to-end deep learning model used in traditional portfolio optimization. While the best performing end-to-end stock portfolios yield Sharpe Ratios between 2.5 and 3.0, the lowest performing LSTM options portfolio yields an average annual Sharpe Ratio of over 5.0. Further, the dominance of the AT Options Portfolio over the prior state-of-the-art GRU and LSTM portfolios show promise for its application to other financial problems. The AT Options Portfolio is an exciting exploration of complex inter-asset options strategies, and if cost reducing methods are effectively integrated into the model, there exists real potential for generating excess risk-adjusted returns.

REFERENCES

[1] Nir Arbel. How lstm networks solve the problem of vanishing gradients. *Medium*, December 2018.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[3] Pablo Caceres. The recurrent neural network.

[4] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[5] Nicolas Christou. Portfolio expected return and risk.

[6] Matthew DiLallo. The biggest stock market crashes in history. *The Motley Fool*, January 2023.

[7] Robert Engle. Dynamic conditional correlation. *Journal of Business & Economic Statistics*, 20(3):339–350, 2002.

[8] Piotr Fiszeder, Marcin Fałdziński, and Peter Molnár. Range-based dcc models for covariance and value-at-risk forecasting. *Journal of Empirical Finance*, 54:58–76, 2019.

[9] Piotr Fiszeder and Witold Orzeszko. Covariance matrix forecasting using support vector regression. *Applied Intelligence*, 51(10):7029–7042, 2021.

[10] Keith D. Foote. A brief history of deep learning. *Dataversity*, 2022.

[11] Adam Hayes. Black-scholes model: What it is, how it works, options formula. *Investopedia*, October 2022.

[12] Adam Hayes. What is financial leverage, and why is it important? *Investopedia*, July 2022.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[14] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[15] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[17] Damian Kisiel and Denise Gorse. Portfolio transformer for attention-based asset allocation, 2022.

[18] Neeraj Krishna. Backpropagation in rnn explained. *Towards Data Science*, 2022.

[19] Christina Majaski. American vs. european options: What's the difference? *Investopedia*, July 2022.

[20] Harry Markowitz. Portfolio selection*. *The Journal of Finance*, 7(1):77–91, 1952.

[21] Gabriel Molina. Stock trading with recurrent reinforcement learning (rrl. Stanford University, 2016.

[22] John E. Moody and Matthew Saffell. Reinforcement learning for trading systems and portfolios. In *Knowledge Discovery and Data Mining*, 1998.

[23] Brian O'Connel. What is a portfolio? *Forbes*, 2022.

[24] quantcha. Us equity historical and option implied volatilities, 2022.

[25] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[26] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[27] William F. Sharpe. The sharpe ratio. *The Journal of Portfolio Management*, 21(1):49–58, 1994.

[28] Gaurav Singhal. Lstm versus gru units in rnn. *Pluralsight*, September 2020.

[29] Midas Technologies. Gamma-theta tradeoff and compensation. *Midas Tech*, 2021.

[30] Alpha Vantage. Time series daily adjusted.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[32] Kieran Wood, Sven Giegerich, Stephen Roberts, and Stefan Zohren. Trading with the momentum transformer: An intelligent and interpretable architecture, 2021.

[33] Yang Yang. Straddle approximation formula. *Brilliant.org*.

[34] Chao Zhang, Zihao Zhang, Mihai Cucuringu, and Stefan Zohren. A universal end-to-end approach to portfolio optimization via deep learning, 2021.

[35] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep learning for portfolio optimization. *The Journal of Financial Data Science*, 2(4):8–20, aug 2020.