STA2536 Project 3: Deep Hedging and Valuation with GRU's

Zhiye Luan (1004282867)

November 30, 2022

Contents

Abstract

Artificial neural networks methods of estimation are commonly used in many applications throughout the informational landscape ranging from natural language processing, image recognition, and more recently, medical diagnosis. The Black-Scholes (BS) model is a differential equation widely utilized for pricing options or other financial derivatives. We relate the two topics of artificial neural networks and hedging with the usage of recurrent artificial neural networks (RNN's) using gated recurrence units (GRU's) to determine optimal hedging strategies for a short position on a bull-spread option under said BS pricing model to compare its performance in profits and losses against classical delta only hedging strategy. We proceed to find that the networks given more historical data tend to perform better.

1 Introduction

Call/put options are financial derivatives that give the holder the right to purchase/sell an asset at a predetermined price (strike) at some time in the future (maturity). Said holders of call options can potentially have unlimited gain due to the unexpected nature of stock prices. Conversely, issuers of call options experience high risk on the off chance that the price of the asset immensely exceeds the strike at maturity. Therefore, in lieu of writing a simple call option, issuers can construct "*bull-spread*" options to mitigate exposure to risk. In general, bull-spread call options are constructed by simultaneously buying a call option with a relatively lower strike and selling a call option with a relatively higher strike on the same asset and maturity.

In this study, we attempt to emulate the position of an issuer who has just sold an at-the-money bull-spread, and so we hedge the short position of the bull-spread option. We then compare the differences between ANN hedging techniques and the classical discrete time delta hedging technique for option valuation and as optimal hedging strategies. We assume an asset transaction fee of \$0.005 per asset traded.

We develop an RNN to hedge the bull-spread option by determine the hedging position in the underlying risky asset at each time step. We then charge the minimal price for the bull-spread option such that the conditional value at risk of the bottom 10% ($CVaR_{0.1}$) of the losses from the profit and loss distribution after 10,000 simulations is no less than -0.02. Moreover, the study analyses and compares the profit and loss of the bull-spread option from RNN's given varying amounts of historical data.

2 Methodology

For our market, we assume an underlying risky asset $S = (S_t)_{t\geq 0}$ which satisfies the stochastic differential equation below:

$$dS_t = \kappa\left(\theta - \log\left(S_t\right)\right) S_t dt + \sigma S_t dW_t$$

where $W = (W_t)_{t\geq 0}$ is a P - Brownian motion. The study assumes a transaction cost of \$0.005 per asset traded, and the base parameters in annualized values are assumed as follows,

$$S_0 = 10, \kappa = 2, \theta = log(10), \sigma = 40\%, r = 2\%, T = \tfrac{1}{4}$$

We hedge the short position of a bull-spread option on a daily basis over 90 trading days. The payoff of the bull-spread option at maturity is,

$$F(S) = (S - K_1)_+ - (S - K_2)_+ \text{, where } K_1 = 9.50, K_2 = 10.50$$

We see that the bull-spread option is constructed by simultaneously buying a call option with the lower strike price $K_1 = 9.50$ and selling a call option with the higher strike price $K_2 = 10.50$ on the same asset S and maturity $T = \frac{1}{4}$ year.

2.1 Black-Scholes Model

The price for a European call option with strike K and maturity T is derived under our Black-Scholes model assumptions. The price of the asset S_t at a time before maturity t follows a geometric Brownian motion with constant drift and volatility coefficients. The bank account B has a constant interest rate r. The dynamics are represented with the following equations:

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t^{\mathbb{P}}$$
$$\frac{dB_t}{B_t} = r dt$$

where $W_t^{\mathbb{P}}$ is a $\mathbb{P}$-Brownian motion.

A claim g written on the asset of price S with payoff function $G(S)$ at maturity T satisfies the Black-Scholes PDE system:

$$\begin{cases} \partial_t g(t,S) + rS\partial_S g(t,S) + \frac{1}{2}\sigma^2 S^2 \partial_{SS} g(t,S) = r g(t,S) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad g(T,S) = G(S) \end{cases} \tag{1}$$

Finally, after solving the Black-Scholes PDE, the price of a call option is:

$$f^{call}(S,t) = S_t \Phi(d_+) - K e^{-r(T-t)} \Phi(d_-)$$

Where Φ is the cumulative density function of the standard normal distribution, and

$$d_\pm = \frac{\ln(S_t/K) + (r \pm \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}$$

2.1.1 Discrete time hedging strategy

In our discussion, we use delta-hedging strategy to set up a hedging portfolio that mitigates the risk derived from selling a bull-spread option by trading "Δ" units of stock at time t. This reduces the risk derived from large fluctuations in S_t. The value of our hedging portfolio is composed of a short position in one unit of a bull-spread option, delta units of stock, and a bank account. The Black-Scholes model assumes continuous delta-hedging. Of course we are able to mitigate all risk if we were to match the Δ at every possible time by making infinite transactions. However, not only are we limited by the the impossible task of making infinite transactions, but also the transaction costs which also go to infinity as we make infinite transactions, resulting in infinite loss. Instead, we re-balance the portfolio at each discrete time step. By re-balancing the portfolio, we aim to reach a delta-neutral position where the components of our portfolio at time t are defined by using:

- S_t: The stock price at time t
- B_t: The bank account at time t
- Δ_t=$\dfrac{\partial F}{\partial S}(t, S_t)$, where $F(t, S_t)$ is the value of the bull-spread option
- ϕ: transaction costs

At time zero, we short a bull-spread call option and obtain F(0, S_0). The bank account balance after buying Δ_0 unis of stock and paying the transaction cost is:

$$B_0 = \mathrm{F}(0,\, S_0) \text{ - } \Delta_0 S_0 \text{ -}\phi|\Delta_0|S_0$$

For every time step, with an interval of Δt, the portfolio is re-balanced. At time t, the bank account balance grows to $B_{t-1}e^{r\Delta t}$. Similarly, we adjust the number of stocks required for hedging and pay the transaction cost. The bank account balance at time t is:

$$B_t = B_{t-1}e^{r\Delta t} - (\Delta_k - \Delta_{k-1})\, S_k - \phi\, |\Delta_k - \Delta_{k-1}|\, S_k$$

At the maturity time T, the option trading is settled and profit and loss is determined. If the transaction is financially settled, the bank account balance is as demonstrated:

$$B_T = B_{T-1}e^{r\Delta t} - (\Delta_T - \Delta_{T-1})\, S_T - \phi\, |\Delta_T - \Delta_{T-1}|\, S_T + K\mathbb{I}_{S_T>K}$$

2.2 Feed Forward Artificial Neural Network Architecture

Artificial neural nets, map an input space $\mathcal{X}$ to an output space $\mathcal{Y}$ through multiple hidden layers. The study deal in part with feed-forward and fully connected neural networks in addition to recurrent architecture. This means there are no reverse flows from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$ and each node is connected to every node in the next layer where we use the feed-forward architecture. Figure 1 is a simple two-layer feed-forward architecture with two inputs x_1 and x_2, and one output y_1.

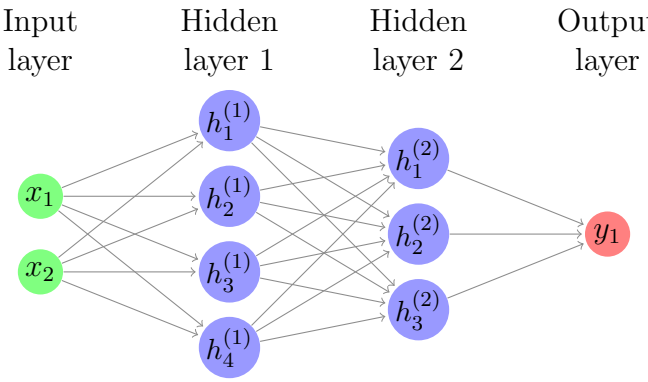

Figure 1: A two-layer neural network with 2 inputs and 1 output

The arrows in neural network architectures represent the propagation of information. We use the following affine transformation as the propagation function,

$$p(\boldsymbol{h}) = \boldsymbol{w}^T \boldsymbol{h} + b$$

where $\boldsymbol{h}$ is a vector that stores the information in a single layer, $\boldsymbol{w}$ is the **weight** vector corresponding to $\boldsymbol{h}$, and b is the bias. The nodes represent activation functions where in our context, we use the sigmoid function.

$$a(x) = (1 + e^{-x})^{-1}$$

2.3 Recursive Neural Network Architecture With GRU's

To fully encapsulate the sequential nature of consecutive prices, we introduce the concept of recursive neural networks (RNN). The usage of RNN allows us to construct neural network that are able to conceptualize "memory," which enables them to save the states or details of consecutive inputs using three major states. The first state called the *input state* is able to capture the sequential input data for the model. The *recurrent state* is a series of hidden hidden processes that propagate through the consecutively linked hidden states of the RNN architecture. Finally, the *output state* is the output of the RNN which encapsulates both the input states along with the sequential hidden states.
Specifically in this study, we incorporate a popular architecture called *gated recurrence units* (GRU's) sequentially to form a GRU layer. GRU architecture differ slightly from the general RNN architecture in the sense that each GRU that form a layer can output a hidden state h_t that can encapsulate the inherited information from previous GRU's.

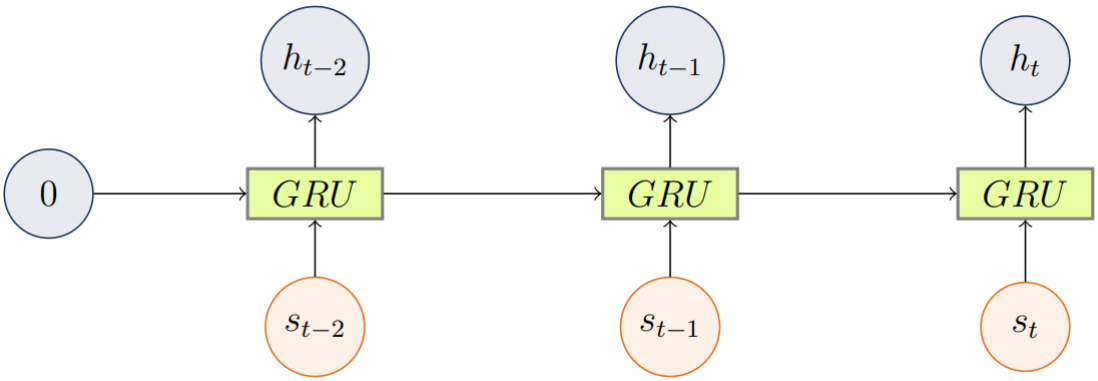

Figure 2: GRU layer of three consecutive input states

Pictured above, the diagram displays a GRU layer that is able to capture three consecutive input states s_{t-2}, s_{t-1}, and s_t. At each GRU node, it is able to emit a hidden state h_t that can describe the flow of sequential information from all previous GRU nodes on top of the corresponding information s_t that is being fed into it. For this study, the first recurrence unit always takes in a zero hidden state.

2.4 Model Architecture

We measure performance as the profit and loss of the resulting bank account after 90 days of hedging with 10,000 price path simulations. We aim to quantify how the incorporation of price history affects the performance of the hedging efficacy of ANN's. To do so, we use a combination of recurrent, and feed-forward architecture in tandem to capture the effect of price history. We use the output of the ANN to determine the daily held position of the underlying stock over 90 trading days. For each day t, the input space consists of the price of the asset (S), the valuation of our bank account (M), and the position that we held at the previous day (Δ) which is the output of the previous day. We refer to this triple as

$$s_t = (S_t, M_t, \Delta_{t-1})$$

To explore the difference that historical prices has on network performance, we vary the size of the input to be directly tied with the number of historical data we would like to consider. In total, we look at the performance of the net on top of being given current s_t, given two points of historical data, one point of historical data, and no points of historical data. Of course to accompany the varying data, we vary the structure of the GRU layer by having the number of recurrence units to match the number of days of data we are considering. For 0 lags, we use one recurrence unit in our GRU layer, for 1 lag we use two recurrence units in our GRU layer, and for 2 lags, we use three recurrence units in our GRU layer.

Along with the GRU layer, we pass in the final hidden state h_t along with the triple s_t into a fully-connected, feed-forward network of 16 layers with 6 nodes per layer. Finally, from this feed-forward network we obtain our network estimated hedging position Δ_t.

2.5 Training

2.5.1 Conditional Values at Risk as Loss

The goal of training is to update network parameters consisting of weights and bias to generate the optimal hedging strategy for a short position of our bull-spread call option. We do so by using stochastic gradient descent to generate the optimal hedging strategy under each network.

Values at risk and *conditional values at risk* is often used in risk management to manage tail risk in investment. Values at risk VaR_α in our context refers to the α quantile of the profit and loss distribution after 10,000 simulations where there is $\alpha\%$ of our losses on the left. Conditional values at risk $CVaR_\alpha$, refers to the conditional expectation as follows,

$$CVaR_\alpha = E[P\&L | P\&L \leq VaR_\alpha]$$

We aim to maintain good profit and loss distribution by distributing to the right as much as possible to produce a desirable $CVaR_{0.1}$ while at the same time minimizing the cost for potential customers hence the at-the-money price. In terms of training the RNN, we train it to minimize the loss function and maximize the profit and $CVaR$. Therefore, the study introduces the loss function ℓ accordingly,

$$\ell(\boldsymbol{\theta}) = -CVaR_{0.1}(\boldsymbol{\theta})$$

Where $\boldsymbol{\theta}$ are the network's parameters. Moreover, we further ensure the $CVaR_{0.01}$ of profit and loss from deep hedging is no less than -0.02 after simulation.

2.5.2 Parameter Optimization

Stochastic gradient descent (SGD) is an optimizer technique to minimize the loss function and update the model parameters. The gradient of the loss function refers to the slope, and so, gradient descent method guarantees a local extreme of the convex loss function. For every iteration, we grab a mini-batch of 100 price paths from the 10,000 price path simulations and updates $\boldsymbol{\theta}$ as follows,

$$\boldsymbol{\theta}_{old} - \eta \nabla \ell(\boldsymbol{\theta}_{old}) \rightarrow \boldsymbol{\theta_{new}}$$

where $\nabla \ell(\boldsymbol{\theta_{old}})$ is the gradient of the loss function given the old parameters, and η is the learning rate. The study uses the learning rate $\eta = 0.005$. When performing the SGD method, for every iteration, $\boldsymbol{\theta}_{old}$ is updated to $\boldsymbol{\theta}_{new}$.

2.5.3 Back propagation

Back-propagation in optimization methods allows us to compute of the gradient of the loss function $\nabla \ell(\boldsymbol{\theta})$ by using derivation chain rules as follows. Suppose the loss function is given by,

$$(\theta_1, \theta_2) \rightarrow z := f(\theta_1, \theta_2) \rightarrow l(z)$$

where θ_1, θ_2 are input parameters stored in a vector $\boldsymbol{\theta} = (\theta_1, \theta_2)$. Then, the sensitivity of the loss function with respect to the θ is:

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \theta_i} = \frac{\partial l(z)}{\partial z} \times \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i}$$

3 Results

For each time lag we consider and its corresponding network, we perform 1000 steps of back-propagation to and then determine the conditional values at risk of the bottom 10% from the network output on 10,000 simulations of a 90 day price path. Let us first observe the results for no lags. For the sake of comparison to the Black-Scholes hedging strategy, we have added to the profits and losses of all of the networks by the Black-Scholes valuation of the bull-spread option which is around \$0.468.

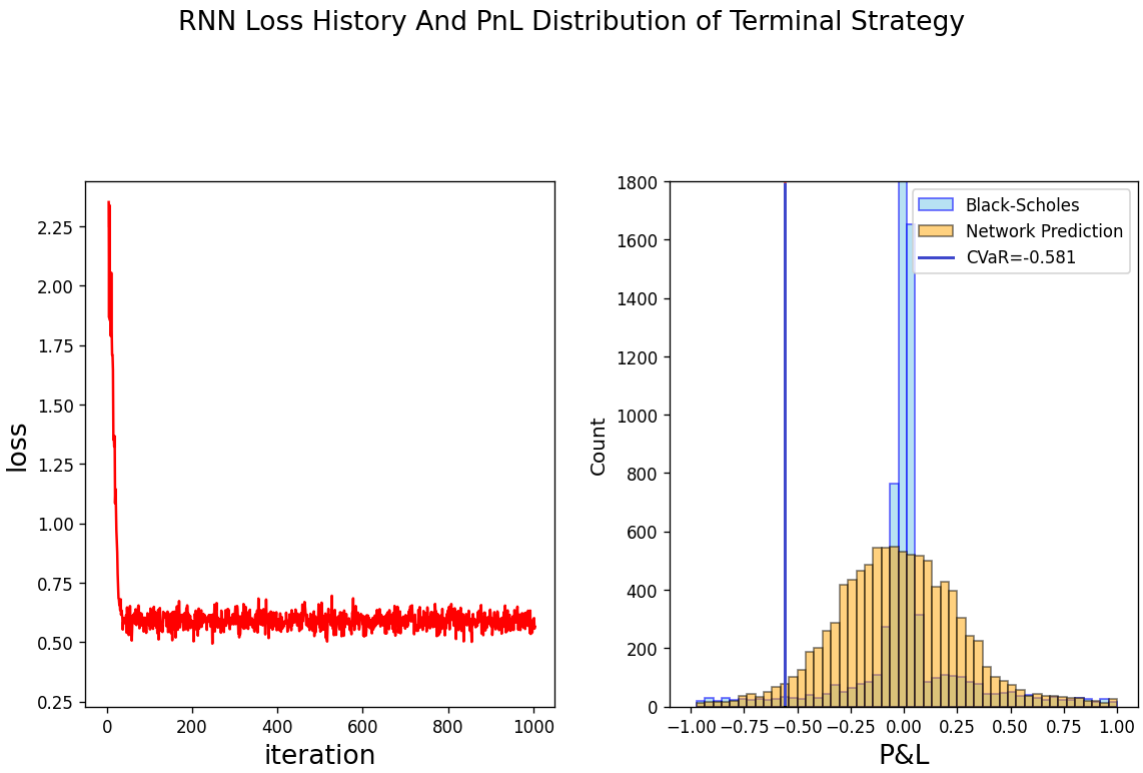

Figure 3: Network results using only current day data

From the plot on the left, we observe the loss history of the network. We can see that within 200 iterations, it managed to enter a local minimum and continued to stay there for the remaining iterations while oscillating at a loss of around 0.6. For the performance of the network after training, we can see for 10,000 iterations, we find that the profits and loss distribution seems to assume a normal distribution centered around 0 with the shape of the being much more widely distributed in comparison to the Black-Scholes. We see that the tail 10% conditional values at risk is around -0.581. We would like for this value to be at -0.02. This means means to charge the optimal price to ensure $CVaR_{0.1} = -0.02$, we have to discount the difference between the current conditional values at risk plus the Black-Scholes valuation and -0.02 back to the signing of the contract in addition to the \$0.468 already charged at time 0. In other words, the optimal price is:

$$(|CVaR_{0.1}| - 0.02) * e^{-rT} + 0.468 \approx 1.027$$

Next, let us look at the results of the network given current day, and lagged one day of data.

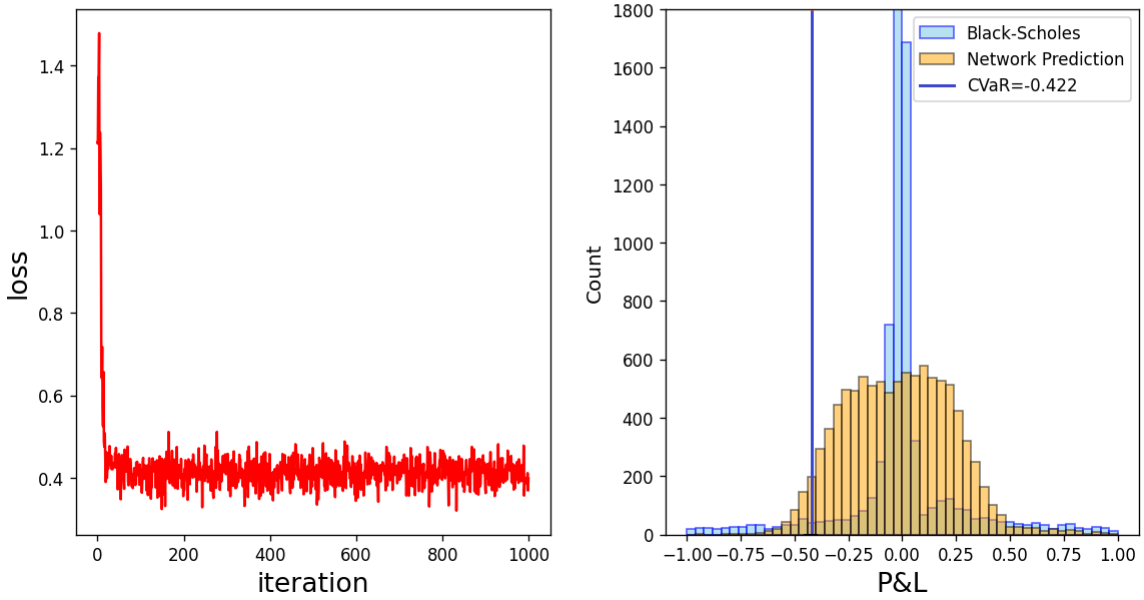

Figure 4: Network results using current day data and previous day data.

For the plot on the left, we observe the loss history of the network. We can see that within the first few iterations, it managed to enter a local minimum and oscillated around 0.4. For the performance of the network after training, we can see for 10,000 iterations, we find that the profits and loss distribution seems to assume a normal distribution centered around 0. Again, we see the shape of the distribution being much wider than that of the Black-Scholes, but still being narrower than the profits and losses distribution of the network trained with no lags. This change is reflected in the tail where we see the tail 10% conditional values at risk is around -0.422 which is considerably better than the previous output. Using the same method as above, our valuation not becomes around \$0.8689 to satisfy $CVaR_{0.1} = -0.02$. This pricing is better than the previous model trained with no lags.

Finally, we observe the model with 2 lags on top of the current day's data.

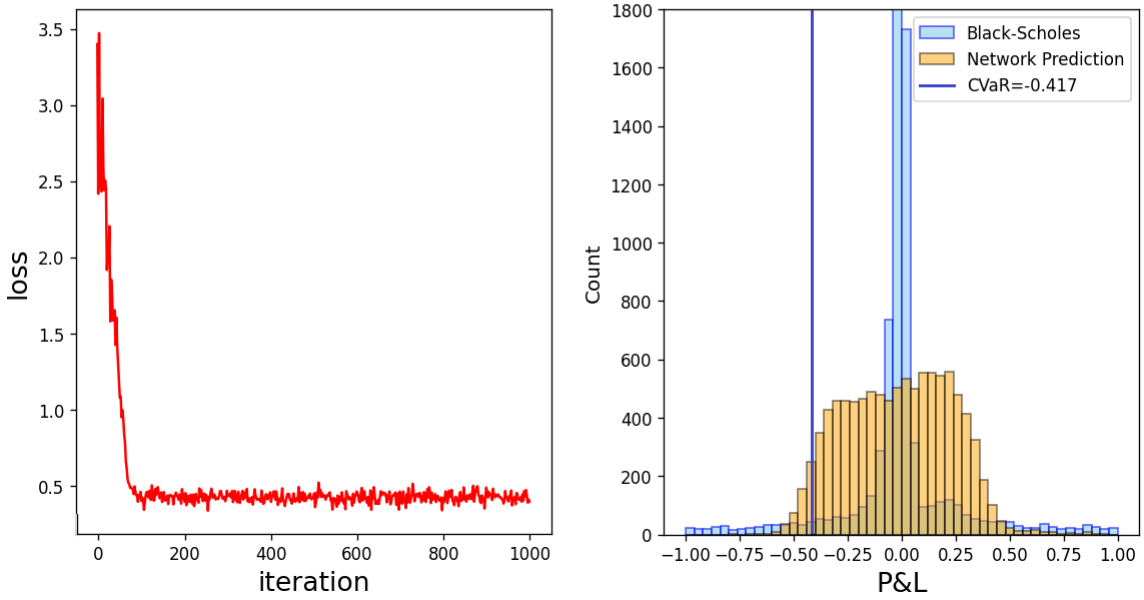

Figure 5: Network results using data from three days

From the loss history, we can see that the model took a bit longer to reach a local minimum taking around 100 iterations where it then starts oscillating about 0.4. From the profits and loss distribution on the right, we see the shape being not much different than that of the results from 1 lag plus the current day's data. This is further reinforced when we see the $CVaR_{0.1}$ to be around -0.417 which is very close to -0.422 from before. To accompany this, the optimum price to satisfy $CVaR_{0.1} = -0.02$ is around \$0.864.
We can see that the jump from 0 lags to 1 lags has a sharper improvement in performance than the jump from 1 to 2 lags.

4 Conclusion

In our discussion, we develop a deep learning approach for the valuation and hedging bull-spread options using a combination of recurrent, and feed forward neural network architecture of one GRU layer that accommodates 0, 1, and 2 lags connected to a fully-connected, feed-forward neural network of 16 layers with 6 layers each. This was chosen due to the heuristic of having deeper nets with shallow layer size performing well with less computational demand. We find that in comparison, the network performs better when given historical data alongside the current day data. This is observed in the lower price required to ensure the $CVaR_{0.1}$ of the profits and loss distribution after 90 days of hedging displayed by the networks given 1 and 2 days of historical data compared to the higher price needed to do the same for the network that was only given one day of data.

We find that the profits and losses distribution of all of the networks seems to perform poorly in comparison to that of the Black-Scholes Δ hedging strategy where most of the shapes of the profits and losses distribution being far away from zero. The lack in performance may be attributed to our choice of network architecture where 16 layers chosen may have been too much due to vanishing gradients. Another potential reason could be the choice of starting the bank account with the Black-Scholes pricing of a bull-spread option as that could have influenced the initial conditions of the network. A possible yet unlikely reason could be that in all scenarios, all models are stuck in some local minima and failed to converge further. This is extremely unlikely as not only are we using stochastic gradient descent, we are also iterating for a thousand generation for each network. Perhaps another reason could be that the sizes of the inputs are not normalized where the difference in the scale for all the input parameters prove to be too large. However, despite all this, the results suggest that with recurrent neural network structure, it is found that giving the network more sequential historical information as time lags can be beneficial in hedging risk compared to only giving the network one day of data.

5 Reference

[1] Cartea, Àlvaro, and Sebastian Jaimungal. Machine Learning and Algorithm Trading.

[2] Jaimungal, Sebastian, and Ali Al-Aradi. Pricing Theory.