STA2536 Project 2: Deep Learning Hedging and Valuation

Ruodan Chen (1005075749)
Wan-ling Renee So (1005472338)
Zhi Ye Luan (1004282867)

November 5, 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 model is a differential equation widely utilized for pricing options or other financial derivatives. Under the Black-Scholes framework, the fundamental idea of delta-hedging is to hedge the option by buying or selling the underlying asset. We relate the two topics of artificial neural networks and hedging with the usage of feed-forward, fully-connected artificial neural networks to determine optimal hedging strategies for bull spread options under Black-Scholes pricing model and determining its performance against classical delta hedging technique. We then observe how different parameters in the forward network affect the resulting strategies. We proceed to find that the network consistently produces strategies that result in P&L distributions centered around -0.5 with its shape of distribution being determined by the neural net architecture.

Keywords— Feed-Forward Fully-connected Artificial Neural Networks, Black-Scholes Model, Discrete Time Daily Delta Hedging

1 Introduction

Over time, the financial market has developed rigorously in various financial instruments and derivatives. Hedging and speculating are the essentials in options trading. Call-and-put options give the option holder the right to sell or purchase a stock at a predetermined price at maturity. More specifically, the holders of call options can potentially have unlimited gain since the stock prices do not have an upper limit. On the other hand, writers of call options experience very high risk. Therefore, instead of writing a plain call option, option writers can build bull spread options to limit their exposure to risk. In general, the bull spread call option is constructed by simultaneously buying a call option with a lower strike price and selling a call option with a higher strike price on the same asset and maturity.

The study aims to hedge the short position of the bull spread option by comparing the differences between the ANN method and the classical Black-Scholes hedge for option valuation and optimal hedging strategy, assuming \$0.005 is charged per asset transaction. To hedge the bull spread option risk, the discrete-time hedging strategy through the Black-Scholes model is examined. Next, the study develops an artificial neural net architecture to hedge the bull spread option, determine the hedging position in the underlying risky asset, and conduct the minimal price to charge for that bull spread option such that the $CVaR_{0.1}$ of the profit and loss is no less than -0.02. Moreover, the study analyses and compares the profit and loss of the bull spread option from the Black-Scholes hedging strategy and the ANN hedging strategy.

2 Methodology

The study explores discrete-time hedging strategies with 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 $\mathbb{P}$ - 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,

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

Furthermore, the study aims to hedge the short position of the bull spread option on a daily basis over 90 trading days (a quarter of a year), where the payoff of the bull spread option is as follows,

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

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

2.1 Black-Scholes Model

2.1.1 Introduction

The theoretical price for a European call option with strike K and maturity T is derived under the Black-Scholes model. By assuming that the stock price S follows a Geometric Brownian with constant drift and volatility coefficients, while the bank account B has a constant interest rate r, we have 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 P-Brownian motion.

Moreover, a claim g written on the asset S with payoff function $G(S)$ satisfies the Black-Scholes PDE:

$$\partial_t g(t,S) + rS\partial_S g(t,S) + \frac{1}{2}\sigma^2 S^2 \partial_{SS} g(t,S) = rg(t,S)$$

with boundary condition $g(T,S) = G(S)$.

Solving the Black-Scholes PDE, the theoretical price of a call option is:

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

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

2.1.2 Discrete time hedging strategy

The study uses delta-hedging strategy to set up a hedging portfolio that offsets the risk of bull-spread option by trading Δ units of stock. This reduces the risk related to the significant price fluctuations of an underlying asset. The value of a hedging portfolio consists of a short position in one unit of bull-spread option, delta units of stock, and a bank account. The Black-Scholes model assumes continuous delta-hedging, however, it is unrealistic in real life. Therefore, the portfolio is re-balanced at each discrete time step. By re-balancing the portfolio, we aim to reach a delta neutral position. The details are discussed below.

The components of the hedging portfolio at time t are defined as:
- S_t: The stock price at time t

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

At t=0, the trader shorts a bull-spread 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 Artificial Neural Networks

2.2.1 ANN Introduction

The ANN, artificial neural nets, maps an input space $\mathcal{X}$ to an output space $\mathcal{Y}$ through multiple hidden layers. The study deals with feed-forward and fully connected neural networks. 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. Figure 1 is a simple two-layer neural net architecture with two inputs x_1 and x_2, and one output y_1. The blue circles in the hidden layers are called nodes. Neural net architectures illustrate how nodes are connected to each other between layers.

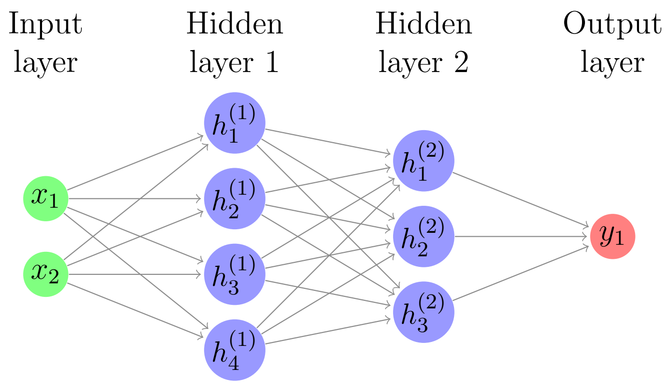

Figure 1: A two-layer neural network with 2 inputs and 1 output

The arrows in neural network architectures represent the propagation, the study uses the affine transformation as the ANN propagation function,

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

where $\boldsymbol{h}$ is a vector that stores the information in a layer, $\boldsymbol{w}$ stands for the **weight** vector corresponding to $\boldsymbol{h}$, and b is the bias. The nodes in neural network architectures refer to activation functions. The study introduces two activation functions, Rectified Linear Unit (ReLU), and Sigmoid.

$$a(x) = max(x, 0),\ a(x) = (1 + e^{-x})^{-1}$$

The study uses ANN to hedge the daily holding position of the underlying stock over 90 trading days. For each day, the input space consists of the current day t in annualized time, and the current underlying stock price S_t. The ANN output space gives the hedging strategy, the desired holding position of the underlying stock on the day t. Moreover, the study explores the impacts of asset price paths (past asset prices) on the ANN hedging strategy by adding yesterday's stock price S_{t-1} in the ANN input space. The study further investigates various neural net architectures with different numbers of layers and nodes.

2.2.2 ANN Training

The goal of ANN training is to learn and update the ANN parameters (the weights and bias) to generate the optimal hedging strategy for the short position of the bull spread option. The study investigates various neural net architectures with different numbers of layers and nodes, using stochastic gradient descent and ADAM optimizer to train the ANN and generate the optimal hedging strategy under each ANN architecture.

Value at Risk and Conditional Value at Risk is often used in risk management to manage tail risk in investment. In this study, Value at Risk VaR_α refers to the α quantile of the profit and loss distribution, where there is $\alpha\%$ P&L data on the left. Conditional Value at Risk $CVaR_\alpha$, which is also known as the expected shortfall ES_α, refers to the conditional expectation as follows,

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

The study wants the profit and loss distribution to distribute to the right as much as possible, meaning that higher CVaR is preferred. In terms of training the ANN, the study trains the ANN to minimize the loss function and maximize the profit and CVaR. Therefore, the study introduces the loss function accordingly,

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

Where $\boldsymbol{\theta}$ stands for the ANN parameters consisting of all weights and biases in the ANN. Moreover, the study further ensures the $CVaR_{0.01}$ of the ANN hedging strategy profit and loss is no less than -0.02

2.2.3 Stochastic Gradient Descent

The Stochastic Gradient Descent is an optimizer technique to minimize the loss function and update the model parameters. The gradient of the loss function refers to the slope of that function, and the gradient descent method guarantees a local extreme of the loss function.

The Stochastic Gradient Descent works as follows. For every iteration, the study grabs a mini-batch of size $N = 100$ price paths from the 10,000 price path simulations. The SGD method updates $\boldsymbol{\theta}$ as follows,

$$\boldsymbol{\theta} - \eta \nabla l(\boldsymbol{\theta}) \rightarrow \boldsymbol{\theta}$$

where $\nabla l(\boldsymbol{\theta})$ is the gradient of the loss function, and η is the learning rate. The study uses the learning rate $\eta = 0.005$. When performing the SGD method, for every iteration, $\boldsymbol{\theta}$ is updated once. The study investigates different numbers of iterations and explores how the loss function and P&L are trained with respect to the number of SGD iterations.

2.2.4 ADAM Optimizer

The Adaptive Moment Estimation, also known as the ADAM optimizer, is a special and efficient type of stochastic gradient descent (SGD). The ADAM optimizer is further developed from SGD to adjust network weights during training. Rather than utilizing a single learning rate, the ADAM optimizer adjusts the learning weight for every network weight. Additionally, it consists of two methodologies, namely the 'gradient descent with momentum' and Root Mean Square Propagation (RMSP) algorithms. These features have enhanced the efficiency of the ADAM optimizer.

Regarding the momentum, the algorithm expedites the SGD by considering the weighted average of the gradients accounted for exponentially. This helps the algorithm to approach the minima at an acceleration pace.

$$w_{t+1} = w_t - \alpha m_t$$

$$\text{where, } m_t = \beta m_{t-1} + (1-\beta)$$

As for the RSMP algorithm that attempts to progress AdaGrad, it considers the exponential moving average.

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t+\varepsilon)^{1/2}} * \left[\frac{\delta L}{\delta w_t}\right]$$

$$\text{where, } v_t = \beta v_{t-1} + (1-\beta) * \left[\frac{\delta L}{\delta w_t}\right]^2$$

- m_t: aggregate of gradients at time t
- m_{t-1}: aggregate of gradients at time t-1
- w_t: weights at time t
- w_{t+1}: weights at time t+1
- α_t: learning rate at time t
- ∂L: derivative of loss function
- ∂W_t: derivative of weights at time t
- β: derivative of weights at time t (constant)
- v_t: sum of square of past gradients
- ϵ : a small positive constant

The mathematical formula of the ADAM optimizer is:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\left[\frac{\delta L}{\delta w_t}\right] v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2$$

The ADAM optimizer has the benefits of both algorithms, providing an enhanced gradient descent. To obtain the global minimum, the rate of gradient descent is manipulated such that the oscillation is minimum as it reaches the global minimum while taking sufficiently large step to arrive local minimum.

2.2.5 Back propagation

The back-propagation in optimization methods allows efficient computation of the gradient of the loss function $\nabla l(\boldsymbol{\theta})$ by using simple chain rules. The back-propagation works as follows, suppose the loss function is given by,

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

where θ_1, θ_2 are arbitrary input parameters and can be stored in a vector $\boldsymbol{\theta} = (\theta_1, \theta_2)$. Then, the sensitivity of the loss function with respect to the θs is computed as follows,

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

The back-propagation has the power to efficiently compute the gradient of the loss function in training Artificial Neural Networks.

3 Results

After implementation in Python, we investigate and compare the results of various architectures of feed-forward neural nets. Then, we further compare the results of the networks to the discrete-time daily hedging strategies of the Black-Scholes model.

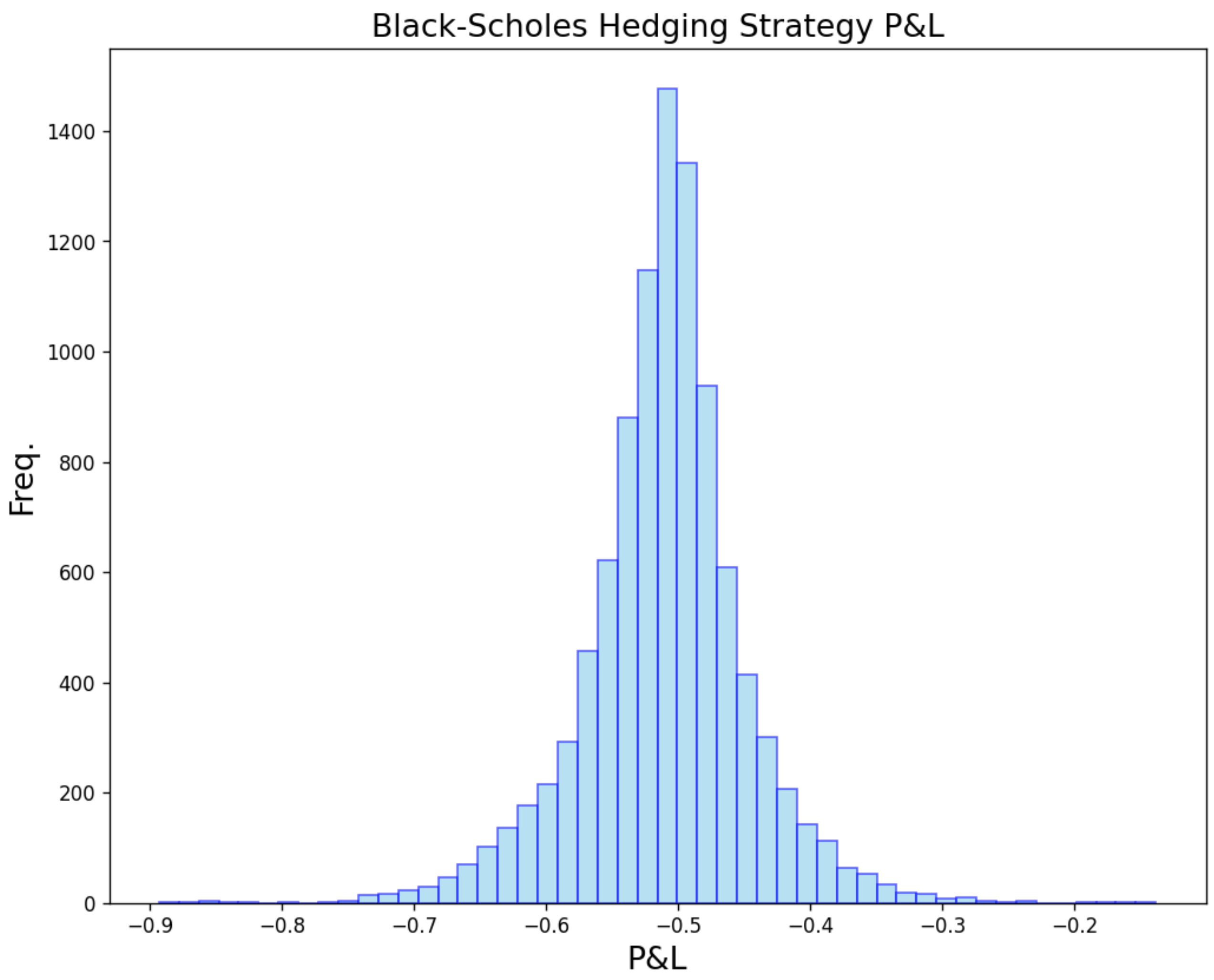

Figure 2: Histogram of profit and loss from daily discrete time Black-Scholes Δ hedging

First, let us observe the resulting profits and loss histogram derived from discrete-time delta-hedging from the Black-Scholes model. We observe that, for 10,000 price paths, the average profit is around -0.5. We can see that the distribution of this histogram admits characteristics comparable to that of a normal distribution with a mean centered around -0.5, with no skewness, and a tail of around 0.15. Due to the inherently random nature of stochastic price paths, the hedging strategy tries its best to correct for the lacking delta for discrete time. This hedging strategy produces better results the more we hedge in discrete time (i.e. the more we re-balance our portfolio to reflect the current delta/rate of change in the underlying risky asset). Of course, in reality, this is an impossible goal as there exist transaction costs per long/short of an asset. In this scenario, we choose transaction costs to be \$0.005 per asset sold regardless of the value of the asset. In more realistic scenarios, the transaction costs are usually a percentage commission of the underlying value of the asset at transaction time t which would imply as the number of times we re-balance goes to infinity, we can expect infinite losses.

Next, we observe the profit and loss distributions from a feed-forward fully collected neural network. First, we begin by observing a feed-forward neural net of three layers with fifty nodes per layer.

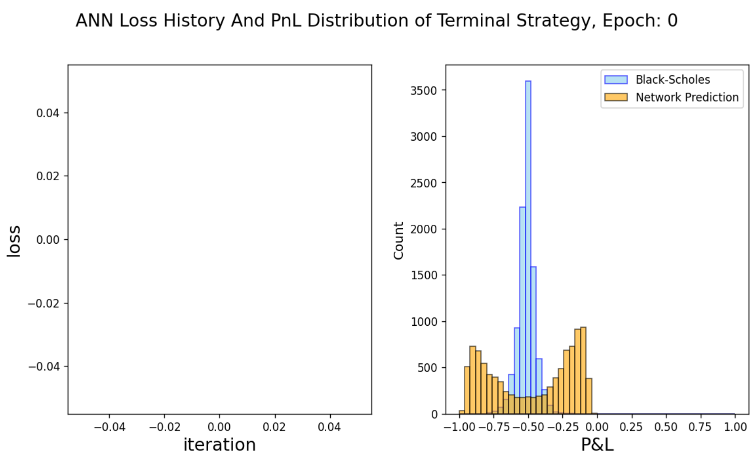

Figure 3: Loss History and PnL Comparison

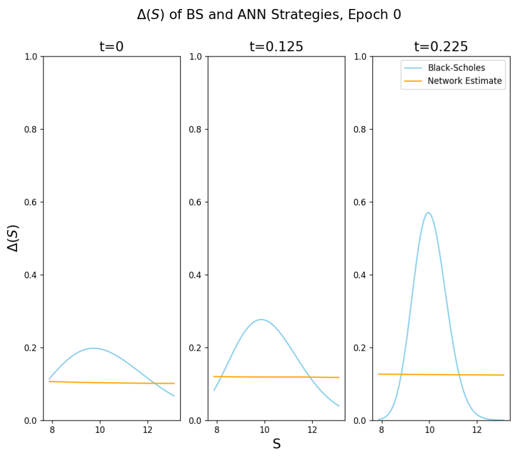

Figure 4: Δ Comparison

We begin at generation zero where we see the results of the neural network's performance in the profit and loss plot on the left. We see that without any training, the profits and loss seem to be bimodal centered around -1 and 0. This would imply that we are consistently over-pricing and under-pricing the option which in reality would imply there would most likely be an overall net loss per contract as the prudent customer would never take the contract to be overpriced.

We also see from the ANN lost history plot that the loss history is empty. This is to be expected as we have not done any training or back-propagation. From the plot on the right, we see that the Δ for the estimated strategy is a straight line at around 0.5 with a slope of 0 at all prices for the three time points $t \in 0, 0.125, 0.225$. These observations are to be expected as these are the results from the base outputs of the neural network without any training. We can expect this type of behavior for all untrained network profits and loss distributions resulting from the linear Δ function.

Let us now observe the evolution of the neural network estimates after 500 generations along with its resulting behavior both in the Δ function and the profits and loss distribution.

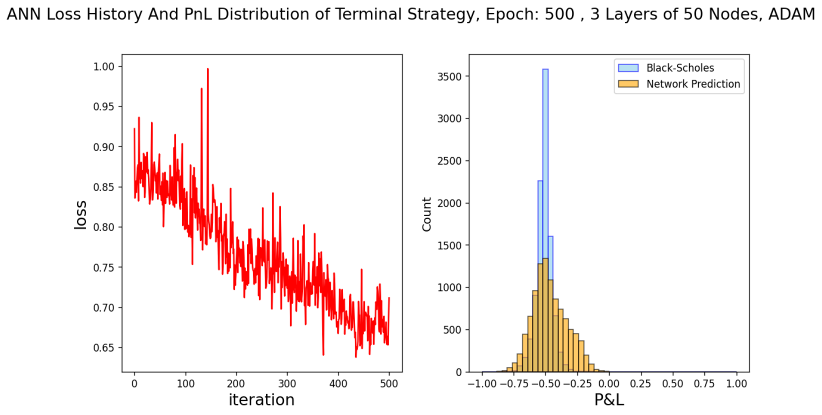

Figure 5: Loss History and PnL Comparison

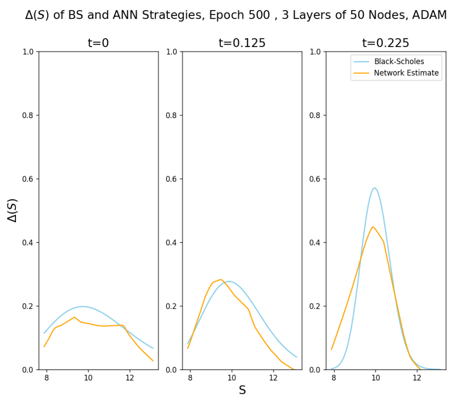

Figure 6: Δ Comparison

We can see that after 500 generations of learning, the neural network progresses in its strategy and produces the loss history, the resulting profits and loss, and the Δ at three different times. We can see that the loss function exhibits the signature stochastic nature due to the stochastic gradient descent with a moving average going down from 0.9 to around 0.65. The profits and loss distribution seems to be shifting away from the bimodal peaks we see at epoch 0 and seems to be congregating towards a -0.5 mean peak similar to the Black-Scholes hedging profits and losses. When we observe the Δ at different times from the plots on the right, we see that the Δ from the network predicted strategy seems to be mimicking the behavior of the Δ function under the Black-Scholes pricing model. As we see from the loss history, the moving average appears to be trending progressively lower as time goes on. However, we notice that the loss function at epoch 500 still shows signs of further descent. Therefore, let us now observe the behavior of the neural network after five-thousand generations of training.

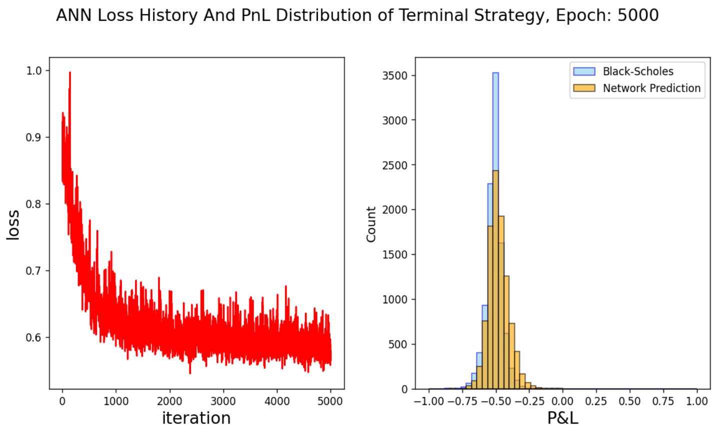

Figure 7: Loss History and PnL Comparison

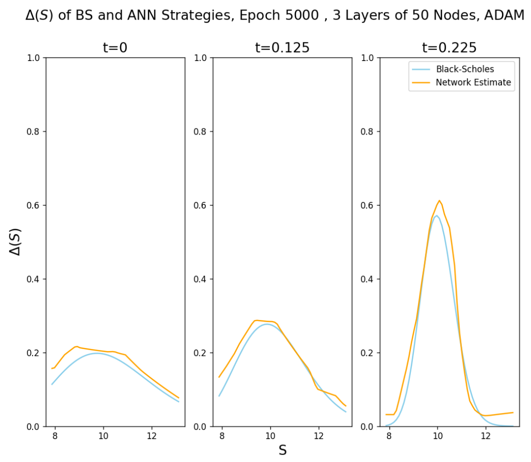

Figure 8: Δ Comparison

From the plots above we can see that after an exhaustive five thousand generations of learning, the neural network progresses similarly to the evolution from generation 0 to generation 500. We see that the average of the loss history seems to stop descending after around generation 1000 and oscillates from the interval of 0.6 ± 0.05. From the loss history, we can see that the profit and loss function from the network estimated strategy seem to now almost coincide with the shape of the Black-Scholes hedging strategy. This similarity also exhibits in the Δ comparisons. We see that the Δ function from the network estimation now almost entirely coincides of the shape with of the Black-Scholes discrete-time hedging strategy.

The result of the artificial network trained with the loss function $-CVaR_{0.10}$ seems to be that after sufficient training, the estimated strategy profits and losses converges to the visibly normal distribution around -0.5 for the purchasing and selling of assets. Furthermore, we are interested in pricing the option so that the $CVaR_{0.10}$ is no more than -0.02. To achieve this, we simply add -0.02 to our bull spread option price at time zero calculated by our network and then discounted by e^{-rT} . In other words, the neural network has predicted the optimal price to be:

$$\begin{aligned} V = (-0.02 - C_0)e^{-rT} &\approx 0.5417 \\ C_0 &\approx -0.56447 \end{aligned}$$

3.1 Layer/Node Modulation

The question arises where one might ask "*are the current hyper-parameters of the neural network sufficient?*" or "*are there simpler ways to train the network to arrive at a similar conclusion?*" In other words, how does changing the architecture of the neural network affect the ANN optimal trading strategy?

Consider a new construction for a feed-forward fully-connected neural network, this time with a single layer of fifty nodes. In other words, we still have two inputs feeding into a hidden layer with the same activation functions, but this time there is only one hidden layer that outputs directly to the hedging strategy output. Let us observe the behavior of this neural network at generation zero.

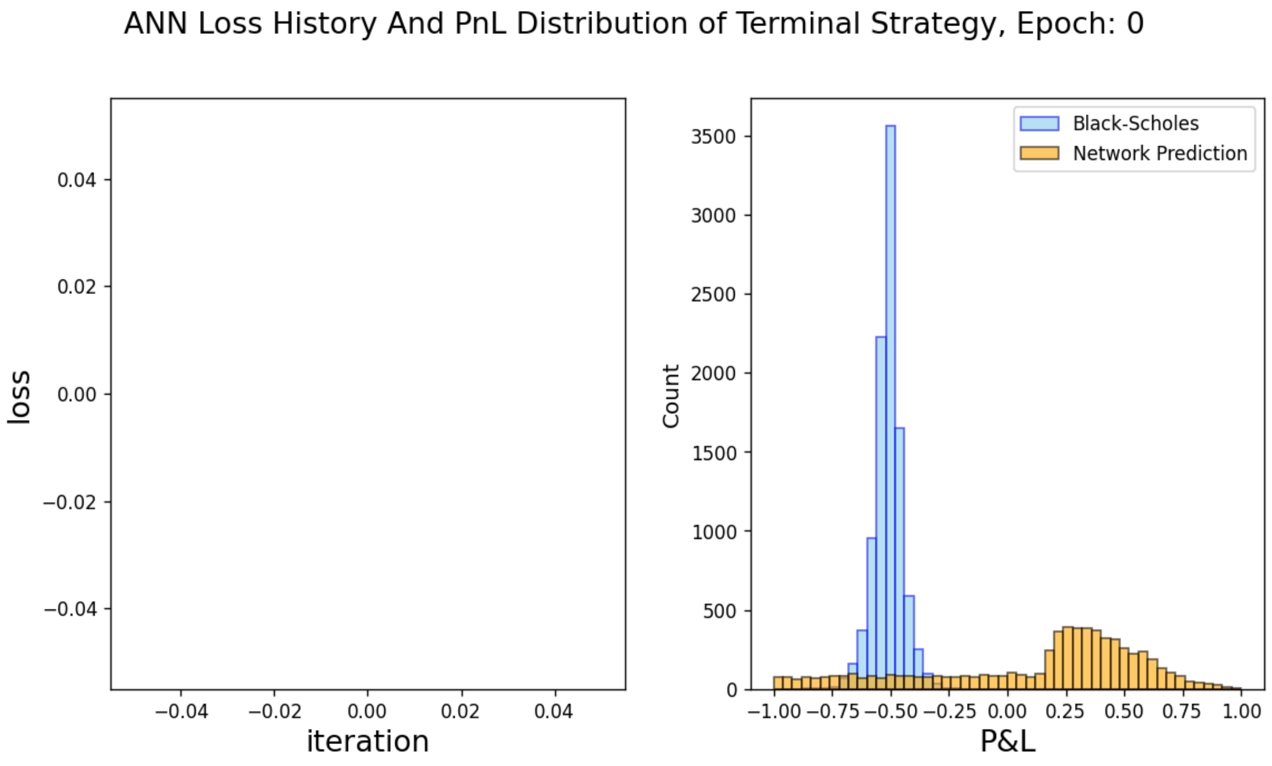

Figure 9: Histogram of profit and loss from daily discrete time Black-Scholes Δ hedging

The plot above reminds us of the results of our first architecture at generation zero where again, the history plot is empty on the account of we have not yet trained our network. Additionally, we can see the profits and loss distribution from the resulting ANN hedging strategy does not resemble that of the discrete-time Black-Scholes Δ hedging strategy at all. Let us now observe the performance of the network after five hundred training iterations.

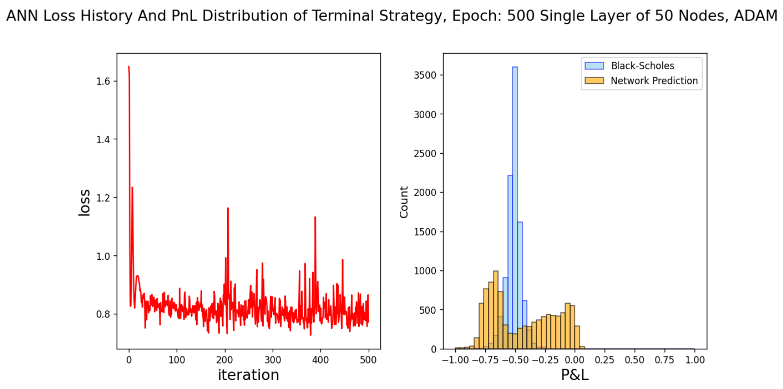

Figure 10: Loss History and PnL Comparison

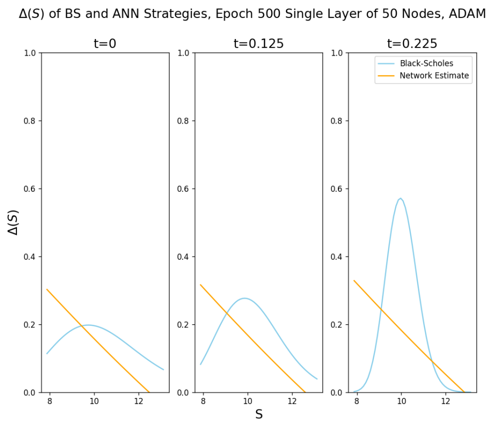

Figure 11: Δ Comparison

After five hundred generations of training, we arrive at the results above. We can see the loss history has appeared to begin to plateau after 100 generations with large jumps in losses at various iterations due to the random nature of stochastic gradient descent. We can also see that the resulting profit and loss distribution begins to converge on the Δ hedging distribution from the Black-Scholes model. Our new model behaves similarly to the previous architecture. We see this similarity again in the Δ function for different underlying prices at various times where the neural network has begun to attempt to mimic the Δ function of the Black-Scholes model. Let us now observe this architecture after five thousand epochs.

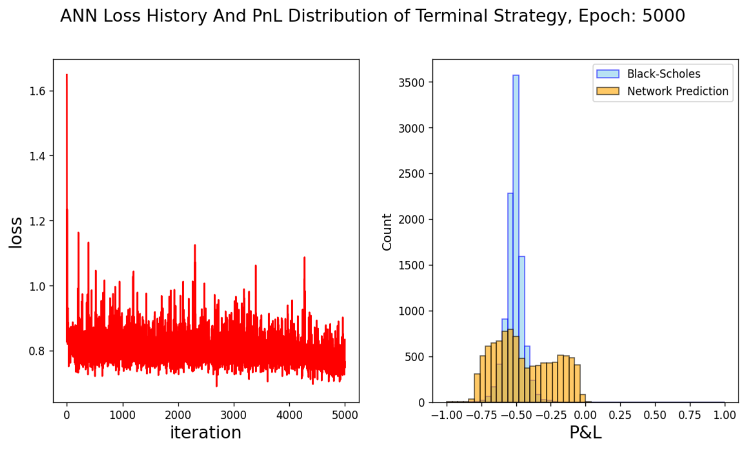

Figure 12: Loss History and PnL Comparison

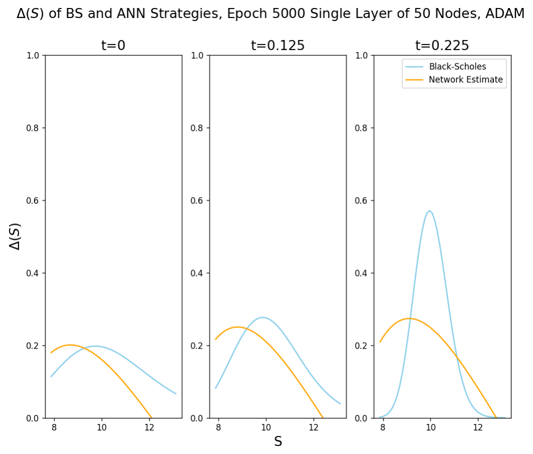

Figure 13: Δ Comparison

The performance of the single-layered network of 50 nodes per layer can be observed from the plots above. We can see that after 5000 generations, the network does not change significantly than the same network after 500 generations. This lack of change is also reflected in the profits and loss histogram where there does not seem to be much of a change in distribution from that of the strategy from epoch 500 aside from smoothing of a few peaks. However, we can see from the resulting Δ distribution that the network estimated function now adapts to a curve that is not presented in the 500 generations version. This suggests that perhaps with more training, the network could adapt its strategy to better converge onto a normal distribution around -0.5. However, we know by the universal approximation theorem that this limited architecture can only mimic the underlying to a certain extent. Perhaps after five thousand generations, it would be wise to choose a more sophisticated architecture. Let us consider a more sophisticated architecture; say 10 layers of 50 nodes each.

We have observed enough strategies at time zero to understand the implications of the loss function and the resulting Δ performance of network estimates to be able to generalize the behavior of the network estimation to be nowhere near the normal distribution centered around zero. So let us observe the performance of the heavy architecture after 500 generations.

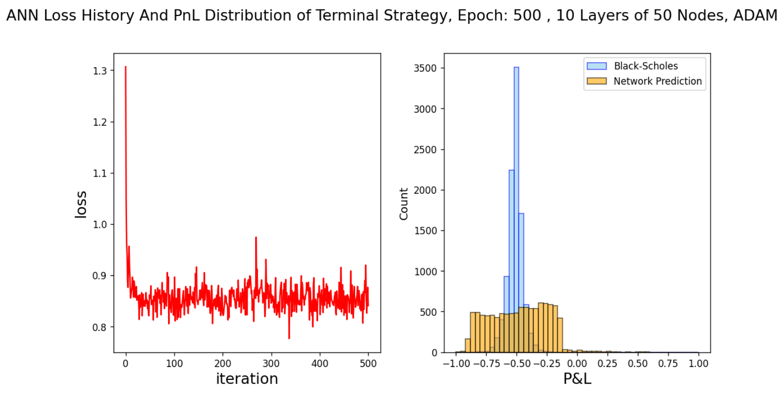

Figure 14: Loss History and PnL Comparison

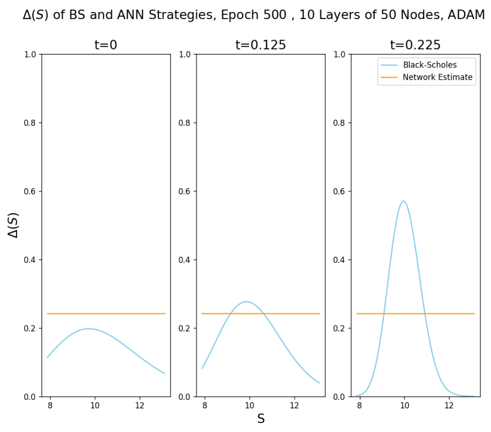

Figure 15: Δ Comparison

Interestingly, having more layers in our architecture results in less adherence to a normal distribution. We can see in the loss history that after 100 generations the losses have stopped decreasing and oscillates between 0.8 and 1. Furthermore, while the mean of the profits and loss distribution seems to match that of the Black-Scholes hedging strategy, the distribution seem to be more spread out over the interval instead of being concentrated at around -0.5. Finally, we see that the estimate ANN hedging function seem to be the same over time. Perhaps this is due to the off chance that each of our chosen mini batch for gradient descent was consistently hindering the loss. To confirm, let us observe the results of the network after five thousand generations.

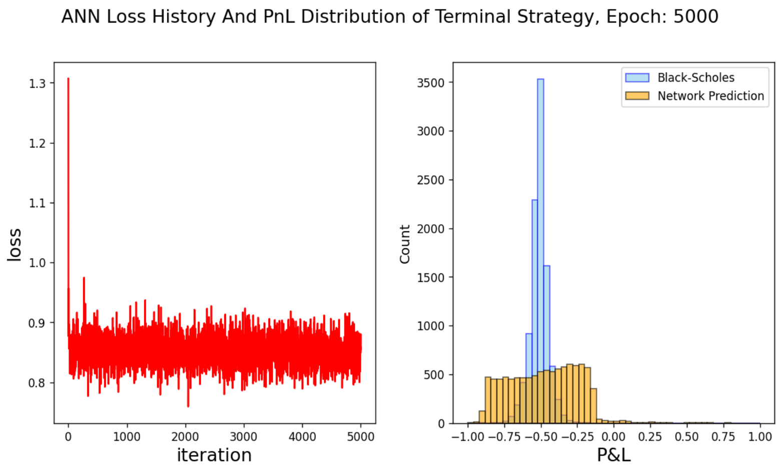

Figure 16: Loss History and PnL Comparison

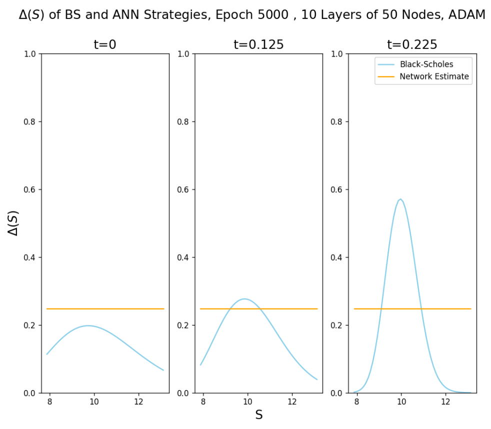

Figure 17: Δ Comparison

From the results above, we can say that the loss history remains oscillating around 0.8 and 1 with the same profits and loss distribution. Additionally, we see that the estimated Δ function remains the same. Evidently, having too many layers causes the strategy to not converge to a normal distribution centered around -0.5. A proposed reason for this could be that the gradient diminishes fast in deeper architectures, so the usual style of back propagation does not work very well. Back-propagated errors become very small after a few layers, which may render learning ineffective.

After exploring the effects of the number of layers in the neural network on the resulting ANN hedging strategy, let us now explore the effects that changing the number of nodes affects the training of the neural network and the subsequent performance of the estimate hedging strategy. Consider a neural network consisting of five layers with each layer containing five nodes. Let us skip observing the behaviour of the estimate strategy without anything training at generation zero and proceed to observe the behavior after five hundred generations of training.

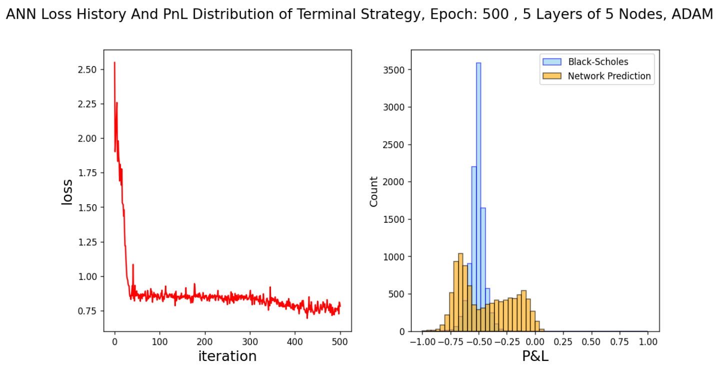

Figure 18: Loss History and PnL Comparison

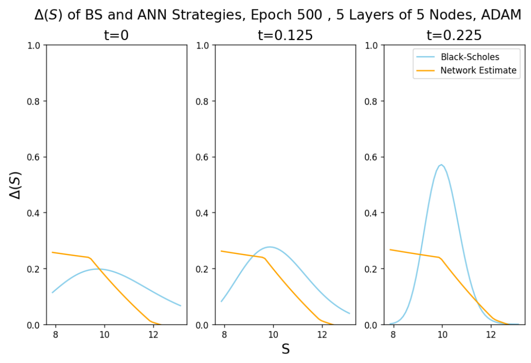

Figure 19: Δ Comparison

Starting from the loss history, we can see that the network has descended rather quickly and maintained an average loss of around 0.8 from epoch 50 to epoch 350 where then after we see the loss function starting to descend even further. The profit and loss distribution at this time seems to be bimodal around -0.75 and 0.15. The Δ function seems to be similar throughout all times t and appears to be a crude piece-wise function. Let us now observe this network after five thousand epochs.

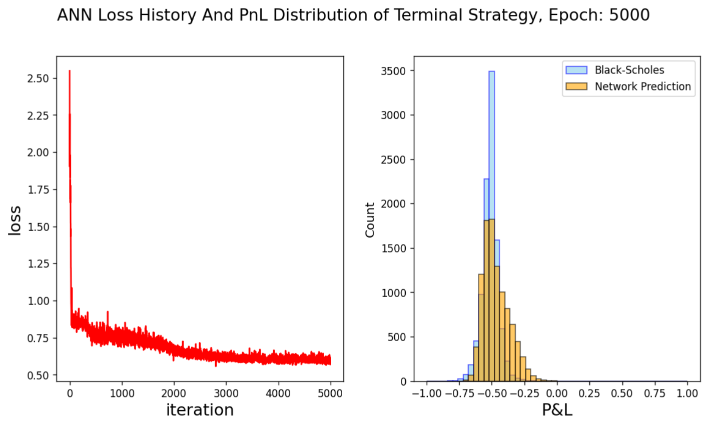

Figure 20: Loss History and PnL Comparison

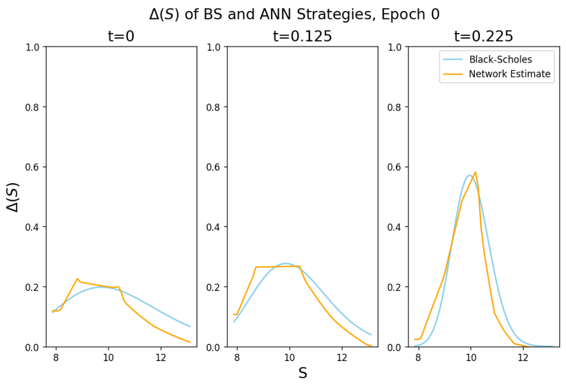

Figure 21: Δ Comparison

After five thousand iterations of training, we arrive at the above results. We see that the loss history after five hundred generations seems to maintain an average of 0.75 until generation 1500 where it then proceeds to descend even further to around 0.6. The profits and loss distribution appear to also converges to -0.5 in a similar manner to our first network of three layers of 50 nodes per layer. The similarities also reflect in the Δ function, the ANN estimated delta mimics the delta function of the Black-Scholes model very closely.

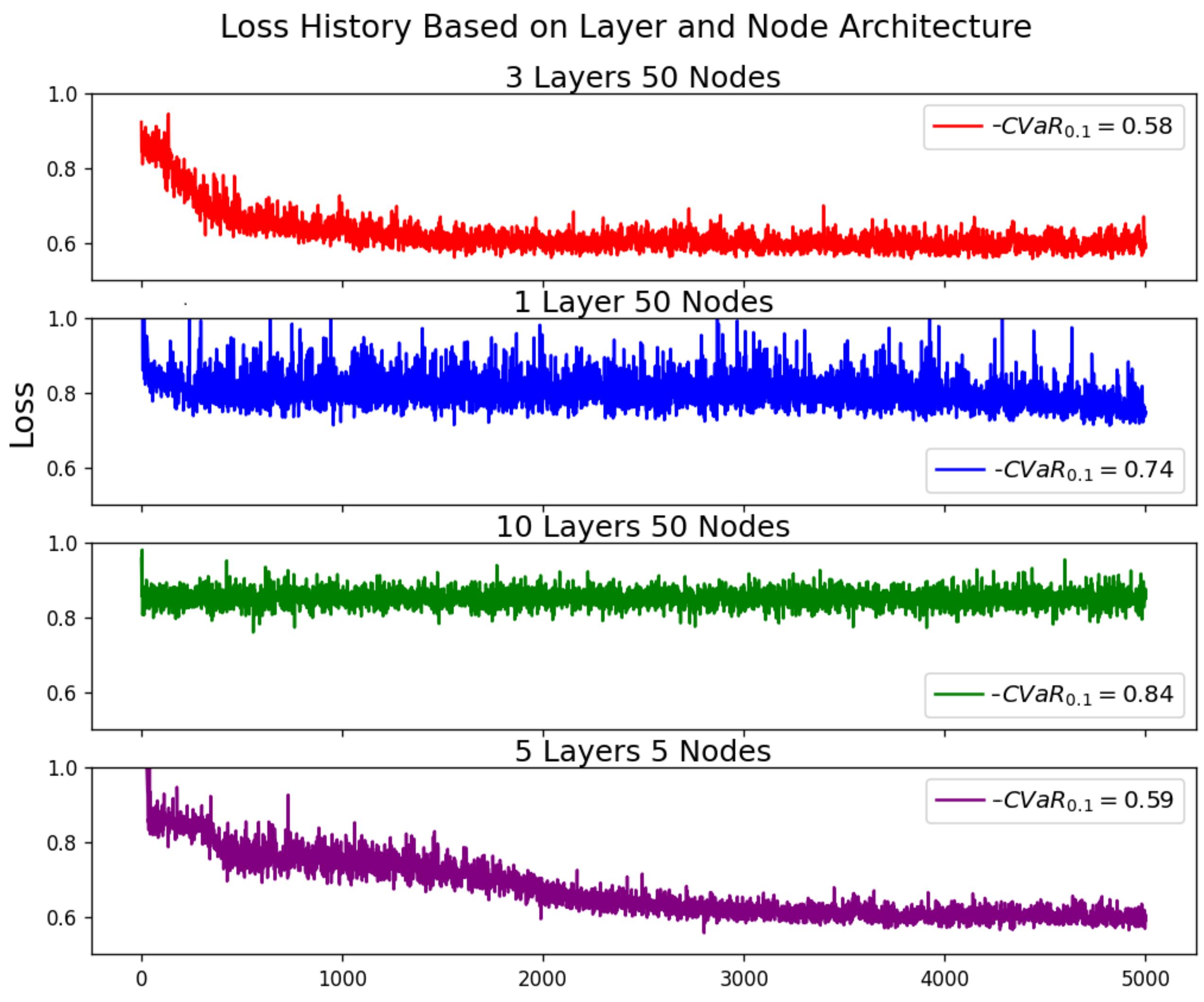

Figure 22: Histogram of profit and loss from daily discrete time Black-Scholes Δ hedging

Above, we can see the comparison in loss histories of all the different network architectures with their associated $CVaR_{0.1}$ after five thousand generations. We see that for neural networks with too many or too little layers, it could either take too long to converge to the familiar normal distribution around -0.5 or unable to converge due to physical limitations of the hidden layers. We also observe that given sufficient layers, just a few nodes per layer is sufficient for convergence to the normal distribution centered around -0.5 albeit taking a longer time to do so. However, this time it takes to converge could be based entirely on chance that specific mini batches consistently kept the losses from descending even further.

3.2 Stochastic Gradient Descent V.S. ADAM

For all of the different architecture we have explored so far, we have been using the ADAM optimizer. Does using stochastic gradient descent change anything? We further compare the two optimizers.

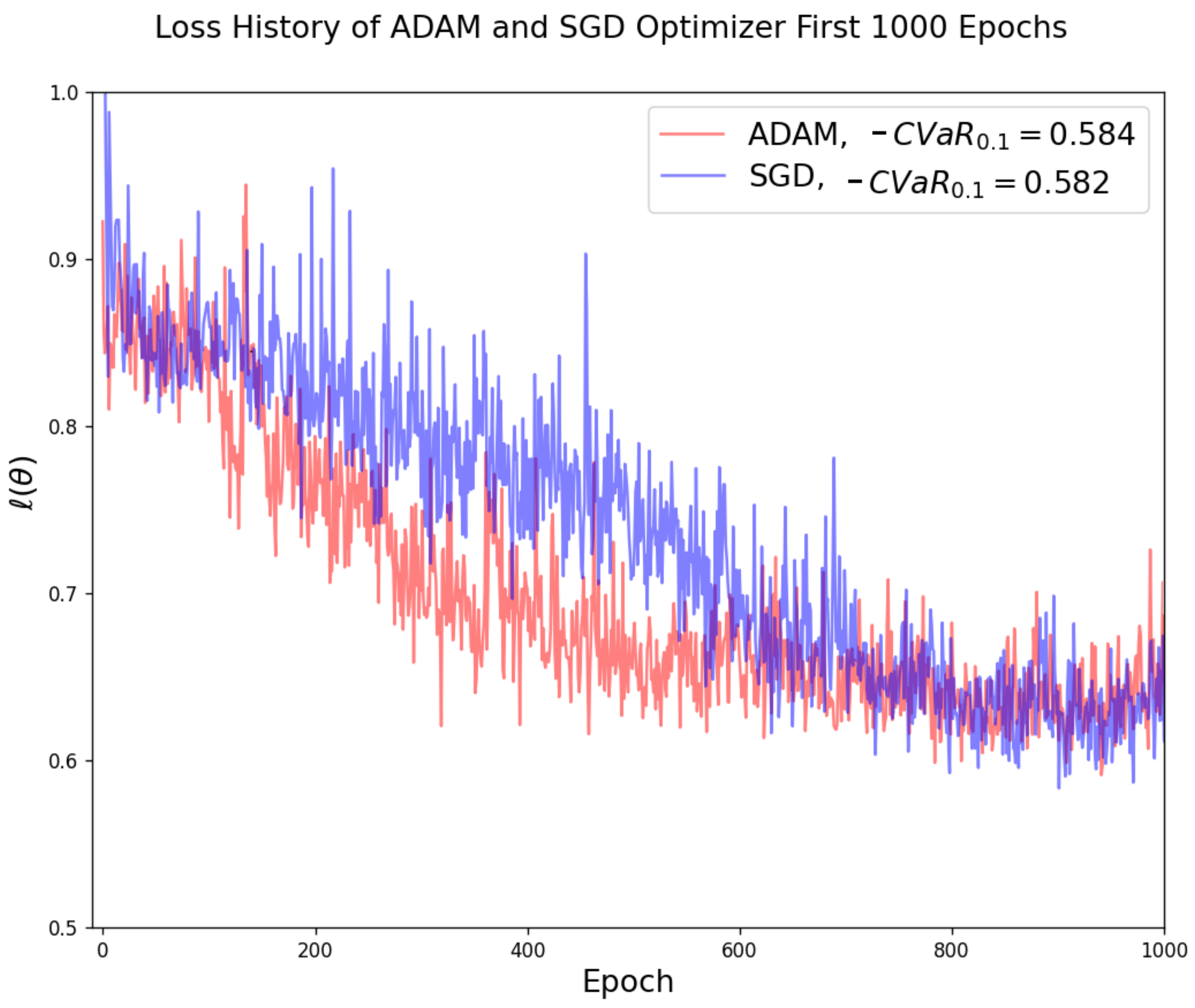

Figure 23: Loss History Comparison of ADAM V.S. SGD

As we can observe from above, the resulting loss function $-CVaR_{0.1}$ for both optimizers are similar within the window of $\pm - 0.002$. By comparison, we see that the network using ADAM reached the final conditional values at risk around 200 generations before the network using pure stochastic gradient descent. However, this difference in the time the networks took to descend could just be due to pure luck where the mini batch sampled from the data just so happened to lead to the illusion of faster loss descent. The main observation to note here remains that both optimizers appear to reach the same stationary average of around 0.58 after five thousand generations of training.

3.3 Feed-Forward Neural Networks With Historical Data

Another way we can alter the architecture is by changing the number of inputs we relay into the hidden layers. For all of our networks so far, we have only used two inputs; time t and the price of the underlying asset S_t at time t. What would happen if were to use not only the price of the current day? Let us observe the results of a neural network trained on three inputs; time t, price of the underlying asset S_t at time t, and the price of the underlying asset S_{t-1} at time $t-1$ somewhat akin to a time series with a time lag of 1. For time zero, we simply have the input vector $(0, S_0, S_0)$.

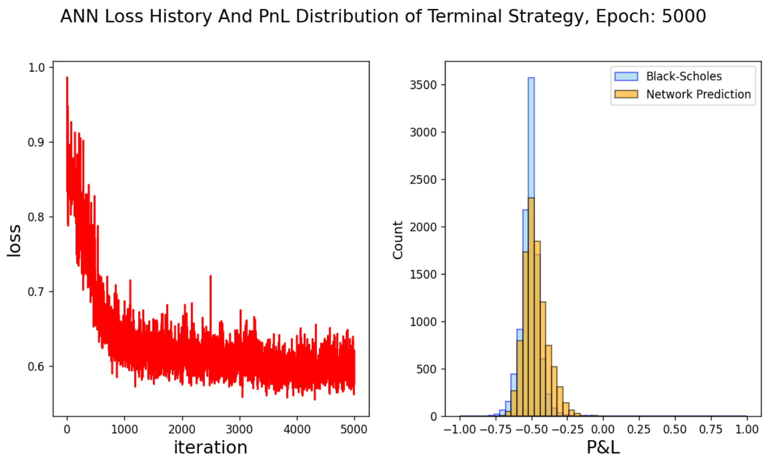

Figure 24: Loss History and PnL Comparison

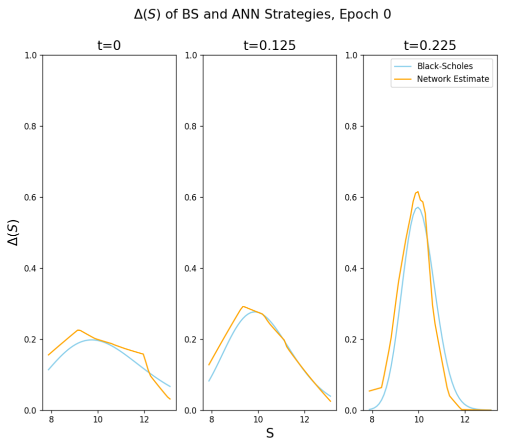

Figure 25: Δ Comparison

The loss history of the 3-input network seems to behave in a similar fashion to that of the 2-input network from before along with similar profits and loss distributions, and Δ function.

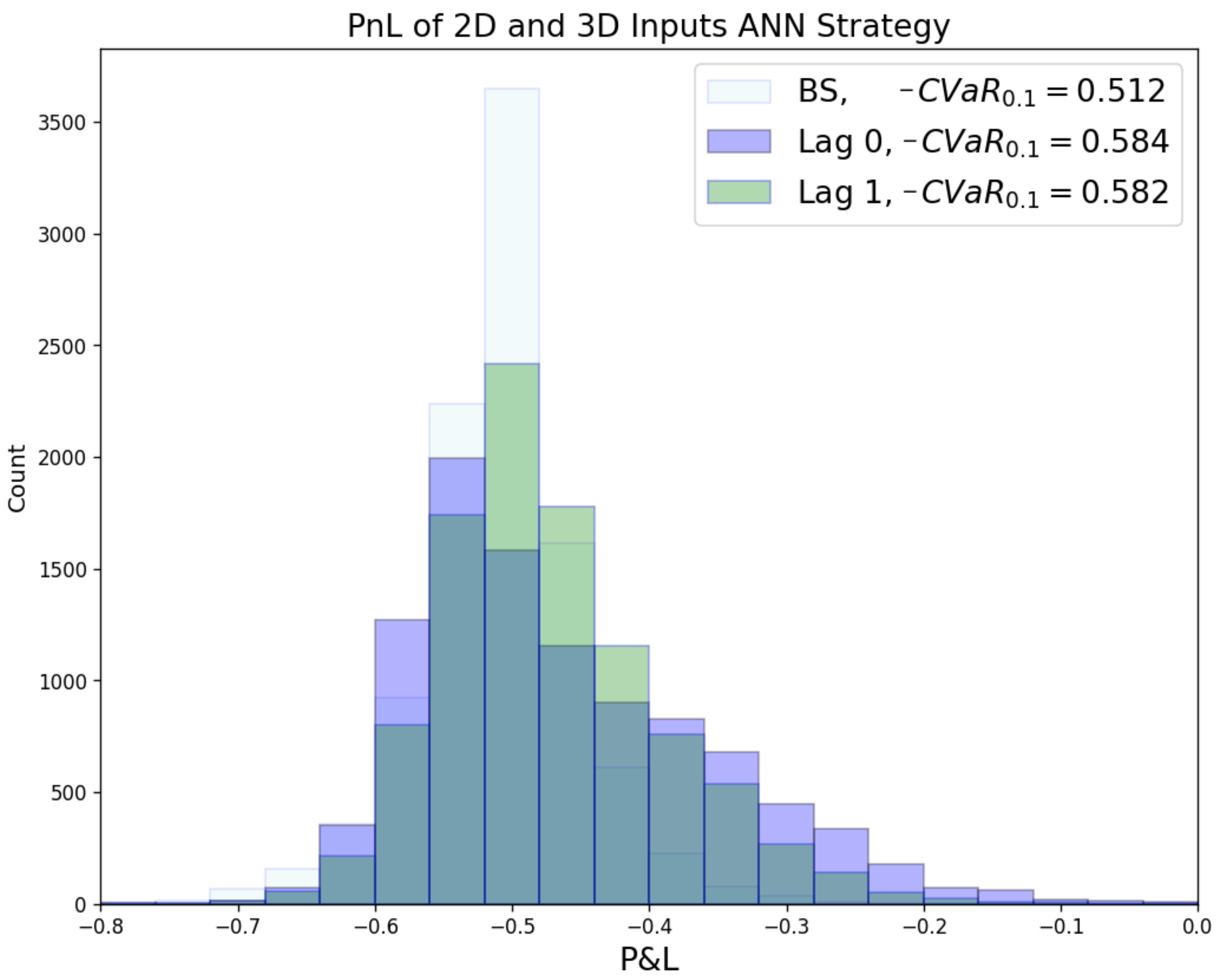

Figure 26: Profit and loss distributions of BS, 2-input, and 3-input network strategies.

As we see above, we compare the profits and loss histograms from the Black-Scholes hedging strategy, the estimated ANN strategy with two inputs (lag 0), and the estimated ANN strategy with three inputs (lag 1). The final conditional values at risk for the 2-input network and the 3-input network seem to be almost the same value. However, we see that the profits and loss distribution from the network with lag 1 seem to be tighter in distribution with a slightly smaller mean. These findings suggest that the hedging strategy estimated by the neural network could further decrease the profits and losses when given more price history.

We would like to further examine the effects of further time-based asset price path with feed-forward networks, however, it becomes evident that there exist some logical inconsistencies with looking at chronological time-based data with a feed-forward network. The proper investigations of true price process based hedging strategies require more sophisticated neural network architectures such as recurrent neural networks beyond the scope of this discussion.

Without any concrete chronological flow of information, the usage of any more price histories as inputs would prove to be redundant. Our naive approach of simply adding more input data is flawed as a way of implementing price histories into a simple feed-forward neural network.

4 Conclusion

In this report, we develop a deep learning approach for the valuing and hedging of bull spread options by using $-CVaR_{0.1}$ as a loss function in feed-forward neural networks. The study finds that while neural networks can mimic the shape of the Black-Scholes hedging strategy being visibly normally distributed losses centered at around -0.5. We find that changing the number of layers in the architecture to be either too simple or too deep can lead to strategies where the profits and losses appear to be uniformly distributed between -1 and 0. Additionally, we find that even with simpler models, with sufficient training time we can come to a similar strategy exhibited by more complex architectures.

We find that with different optimizer functions, the performance of the loss history may be impacted. We find that the ADAM optimizer seems to minimize loss slightly faster than classic stochastic gradient descent. However, this notion is inconclusive as there is always a possibility that faster descent can be attributed to being "lucky" by choosing a mini-batch that avoids any small radius local extremes. We also explore the concept of using price paths in feed-forward networks where we simply add the asset price of the previous day as an additional input into the network which produces a PnL distribution with slightly higher losses. We then quickly reject this idea when we realize that feed-forward fully-connected networks do not have the capabilities to fully conceptualize the time aspect of a price path.

5 Reference

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

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

[3] Gupta, Ayush. A Comprehensive Guide on Deep Learning Optimizers. 7 Oct. 2021, www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/.

[4] "Intuition of Adam Optimizer." GeeksforGeeks, 24 Oct. 2020, www.geeksforgeeks.org/intuition-of-adam-optimizer/.