Hedging Is All You Need

Timothé Dard [1 2] **Marius Pécaut** [1 2] **Christopher Soriano** [1 2]

Abstract

Classical portfolio hedging strategies such as Black–Scholes delta hedging rely on analytical solutions derived under restrictive assumptions, including constant volatility, log-normal price dynamics, and stationary distributions, which limit their applicability in real financial markets characterized by stochastic volatility, heavy tails, and time-varying dynamics. In this work, we propose a deep learning–based framework for learning optimal hedging strategies directly from data. We implement and compare three data-generation approaches and evaluate their impact across two deep hedging pipelines, namely recurrent neural networks (RNNs) and signature-based transformers.

1. Introduction

In recent years, financial markets have experienced a substantial growth in options and derivatives trading, increasing the importance of robust and effective hedging techniques. Classical approaches such as delta hedging within the Black–Scholes framework (Black & Scholes, 1973) offer elegant analytical solutions and theoretical guarantees, but rely on restrictive assumptions such as constant volatility and log-normal price dynamics that fail to capture key empirical features of real financial markets. More recent work has explored deep hedging strategies (Buehler et al., 2019), demonstrating promising performance on synthetic price paths generated under stochastic volatility models such as Heston (Heston, 1993). However, the practical relevance of these results remains limited, as the underlying data-generating processes impose strong structural assumptions and yield guarantees that are largely theoretical. Subsequent studies investigate robustness to distributional shifts and adversarial perturbations (He et al., 2025), yet still rely on Black–Scholes-generated trajectories for both training and evaluation, further constraining their applicability to real-world markets.

In this work, we study deep hedging under more realistic market conditions by combining three data sources: historical market data, Black–Scholes trajectories, and price paths generated via signature diffusion models (Barancikova et al., 2024). We implement a state-of-the-art signature diffusion pipeline that leverages log-signature representations to capture higher-order temporal dependencies, resulting in more expressive and realistic synthetic time series. On the modeling side, we compare two learning-based strategies: a recurrent neural network (RNN) and a signature-based transformer inspired by SigFormer, which integrates attention mechanisms with path-signature features to enhance the representation of complex market dynamics.

2. Setup

We aim to train models that learn to take optimal trading positions on 30-days price paths. In order to achieve this, we feed the network a certain amount of normalized price paths that all start at $S_0 = 100$ and evolve at each discrete timestep $t = 0, \dots, T = 30$. We explore different methods of data generating processes that will be subsequently described. The discrete-time market setup is as follows.

A discrete-time market model is specified on dates $t = 0, 1, \dots, T = 30$ with n tradable assets whose prices form a path $S = (S_0, S_1, \dots, S_T) \in \mathbb{R}^{n \times (T+1)}$.

At time $t = 0$ a single option is sold for an initial premium p_0 (which we would not consider in the training as it is not relevant for the hedging). A trading strategy is a predictable sequence of positions δ_t in the assets where each δ_t is produced by a parameterized decision rule $f_{\theta,t}$ (for instance, a neural network). Along a realized path, the cumulative trading gain is $\sum_{t=1}^{T} \delta_t^\top(\theta)\big(S_{t+1} - S_t\big)$, while at maturity the option payoff $P\big(S_T\big)$ is due. The terminal profit-and-loss of the portfolio consisting of the short option and the dynamic trading strategy is therefore

$$\mathrm{PnL}(\theta) = \sum_{t=1}^{T} \delta_t^\top(\theta)\big(S_{t+1} - S_t\big) - P\big(S_T\big), \qquad (1)$$

i.e cumulative trading gains minus the terminal option payoff.

Rather than maximizing expected profit alone, the strategy is chosen by minimizing a convex risk measure of its terminal PnL. Let $Z = \mathrm{PnL}(\theta)$. A monetary convex risk measure ρ penalizes low realizations of Z so that strategies with higher and less risky profits attain smaller values of ρ. The learning problem is therefore $\min_{\theta \in \Theta} \rho\big(\mathrm{PnL}(\theta)\big)$ which can be interpreted as maximizing a risk-adjusted performance criterion. We focus on risk measures that admit an optimized certainty equivalent (OCE) representation,

$$\rho(Z) = \inf_{\omega \in \mathbb{R}} \Big\{ \omega + \mathbb{E}\big[\ell(-Z - \omega)\big] \Big\},$$

for some convex loss function ℓ.
In all experiments (RNN and SigFormer), we train the hedging strategy by minimizing this entropic OCE risk, i.e., we use $\rho(Z) = \frac{1}{\lambda}\log(\mathbb{E}[\exp(-\lambda Z)]$ as the loss function. For numerical stability, we clip extreme values of $(Z = \mathrm{PnL}(\theta))$ before applying the exponential to prevent overflow and overly large gradients. The parameter λ reflects the risk-aversion level and is chosen at the discretion of the investor.

3. Data Generating Processes

Deep learning algorithms typically require extensive training sets to approximate the underlying data-generating functions well and to avoid overfitting. However, given a hedging horizon of $T = 30$ days, a standard trading year of 252 days yields only 222 price paths per asset. To overcome this limitation, in addition to the historical market price paths, we use two additional data generating processes : the multivariate Black–Scholes–Merton model taking in market regimes factors to accurately take into account of dynamic changes in volatility and drift and a signature-based diffusion model (SigDiffusion).

Figure 1 compares the AAPL return distributions obtained from the three data sources. For each method, the training distribution is shown alongside validation and test distributions computed from market data. While the three approaches exhibit similar central behaviour, they differ in dispersion and tail thickness, with noticeable variations in the frequency of extreme returns across data-generation methods.

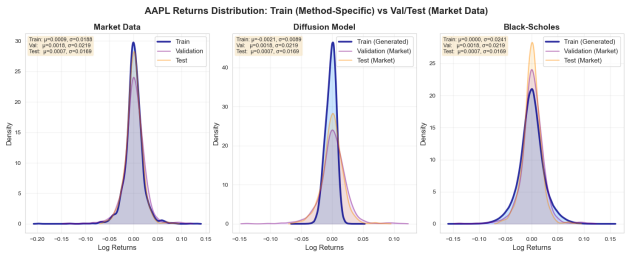

Figure 1. Distribution of the train, test and validation sets over the different Methods

In the following subsections, we provide the description of each data collection and generation process.

3.1. Market Data

To evaluate the effectiveness and fidelity of the synthetic data, we benchmark the performance of the models trained under generated data against empirical market data. It allows one to assess whether the dimensionality expansion improves generalization relative to learning directly from a limited set of historical observations.

We use historical daily adjusted closing prices (close-to-close) for the set of underlying tickers over the period 2008.09.26 to 2024.12.31. To avoid look-ahead bias, we apply a strict chronological split: the first part of the sample is used for model selection and training, while the remaining period serves for out-of-sample evaluation.

Let $\mathbf{S} = \{\mathbf{S_t}\}_{t=0}^{T}$ denote the resulting daily price series. We convert each series into overlapping rolling windows of length $T = 30$ trading days (corresponding to our one-month hedging horizon). Each window is re-based to $S_0 = 100\$$ to ensure comparability across assets and time.

This yields a dataset of normalised price paths (and associated log-returns) of shape $[B, T, N]$ (B is the batch size) that serves as a realistic benchmark against synthetic generators and the source of test trajectories used to evaluate hedging performance under true market dynamics.

3.2. Black–Scholes Regime-Switching

To augment the limited amount of historical trajectories, we generate synthetic training paths using a regime-dependent multivariate Black–Scholes model. We first calibrate market regimes on the training period only. Using a value-weighted market portfolio, we compute a standardized feature vector (returns rolling volatility, rolling mean correlation, and cross-sectional volatility) and fit a Gaussian Mixture Model (GMM). Each day is assigned a regime label $z_t \in \{1, \ldots, K\}$ and for each regime k we estimate an empirical covariance matrix $\hat{\Sigma}_k$ from the asset return observations classified in that regime (with Cholesky factor $\hat{L}_k$ such that $\hat{L}_k\hat{L}_k^\top = \hat{\Sigma}_k$). The drift vector $\hat{\mu}$ is estimated as the overall sample mean of returns. On Fig.(2),

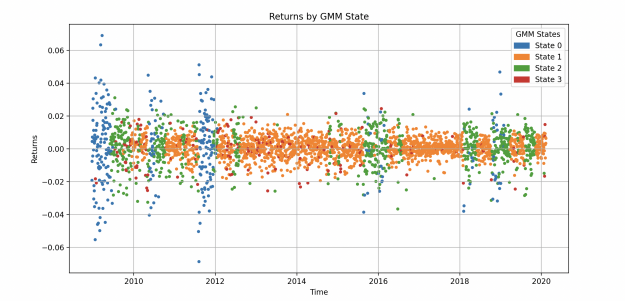

Figure 2. Market regimes on the value weighted portfolio

we observe the different classification of market regimes, effectively capturing the higher volatility during turbulent periods. This allows the generated data to fit the current state of the economy which is classified in four different regimes. By learning regime-specific volatility patterns, the model captures heterogeneous market dynamics.

For each regime k, we then simulate $M = 1000$ price paths using an Euler discretisation of the multivariate GBM:

$$S_{t+1} = S_t \odot \exp\left(\left(\hat{\mu} - \tfrac{1}{2}\,\mathrm{diag}(\hat{\Sigma}_k)\right)\Delta t + \hat{L}_k Z_{t+1}\sqrt{\Delta t}\right)$$

where $Z_{t+1} \sim \mathcal{N}(0, I_n)$. The resulting synthetic dataset consists of normalised 30-day windows together with their regime label, and is used to train the hedging networks under a wide range of volatility/correlation conditions. Out-of-sample evaluation is performed on the historical test period, where regimes are assigned using the GMM fitted on training data.

A detailed explanation of the different formulas as well as the extended procedure is available in the Appendix

3.3. Signature Diffusion

We also use the score-based diffusion model adapted to log-signatures representations of time series from (Barancikova et al., 2024). Historical price trajectories are first interpeted as continuous paths and segmented into fixed time intervals. We then compute a log-signature embedding (8.2) capturing the temporal structure of the path. A score-based diffusion model is then trained on these log-signature vectors by progressively pertubring them with Gaussian noise and learning the corresponding score function. The synthetic samples are obtained by simulating the reverse diffusion process in log-signature space. Finally the generated log-signatures are mapped back to time series paths using an inversion procedure.

We also tried the generation method from (Yuan & Qiao, 2024), a related diffusion-based approach, that operates directly on raw time-series data rather than on log-signature embeddings. In our experiments, SigDiffusion yielded higher sample quality at a lower computational cost. Consequently, we retain only the data generated using the SigDiffusion approach.

Due to computational resource constraints, the diffusion model is trained independently for each asset, without jointly modeling multiple assets. As a consequence, cross-asset dependencies are not explicitly captured in the generated data and we therefore expect hedging performance to degrade in multi-asset settings compared to single-asset scenarios. For each asset, we generate 20,000 synthetic price paths using the trained diffusion model.

4. Deep-hedging

4.1. RNN Hedger

We use a Recurrent Neural Network given a price path $S = \{S_t\}_{t=0}^{T}$ (possibly multi-asset), we compute log-returns

$$r_t = \log\left(\frac{S_{t+1}}{S_t}\right), \quad t = 0, \dots, T-1,$$

and feed the lagged return features $\tilde{r}_t := \{r_\tau\}_{\tau=t-L}^{t-1}$ with $\tilde{r}_0 = \emptyset$ as input to the network (so that the decision at time t only uses information up to $t-1$). In the regime-switching setting, we additionally provide the regime label z_t as an exogenous input. The network (shared across time) outputs hedge holdings

$$\delta_t = f_\theta(\tilde{r}_t, z_t) \in \mathbb{R}^n, \quad t = 0, \dots, T-1,$$

yielding a trading strategy whose PnL is computed from the gains $\sum_{t=0}^{T-1} \langle \boldsymbol{\delta}_t, \mathbf{S}_{t+1} - \mathbf{S}_t \rangle$ net of the option payoff. Parameters θ are trained by minimising an exponential-utility/OCE risk criterion over simulated or historical paths. Cross validation is used to find optimal batch size, hidden-layers lengths and learning rate. While a depth of two hidden layers may be restrictive, we adopt this choice to ensure a fair comparison with (He et al., 2025).

4.2. SigFormer Hedger

We also implement a causal Transformer hedger using SigFormer (Tong et al., 2023), which combines truncated path signatures with masked self-attention to capture higher-order and long-range paths dependence. From a normalised price path $\mathbf{S} = \{S_t\}_{t=0}^{T}$ we form log-returns $r_t = \log(\mathbf{S}_{t+1}/\mathbf{S}_t)$ and construct a return stream $\mathbf{x}_{0:t}$ available up to time t. We then compute streaming truncated signatures of order m,

$$\mathbf{S}_t = \mathrm{Sig}^{(m)}(\mathbf{x}_{0:t}), \quad t = 0, \dots, T-1,$$

which yield a graded tensor representation encoding iterated-integral features of the path. SigFormer applies, for each signature level, multi-head causal self-attention (lower-triangular mask) followed by a position-wise MLP with residual connections and layer normalisation. A final linear readout maps the resulting representation to hedge positions

$$\boldsymbol{\delta}_t = g_\theta(\mathbf{S}_t) \in \mathbb{R}^n, \quad t = 0, \dots, T-1$$

defining a dynamic strategy evaluated via trading gains $G_T = \sum_{t=0}^{T-1} \langle \boldsymbol{\delta}_t, \mathbf{S}_{t+1} - \mathbf{S}_t \rangle$ and trained end-to-end under the same risk-sensitive hedging objective as the RNN.

You can find more details on the architecture of the SigFormer in the Appendix

5. Results and Analysis

Figure 3 compares the market-trained SigFormer to all alternative approaches on a call option on AAPL. The market SigFormer achieves positive mean P&L across all hedge sets, whereas the market-trained RNN remains essentially flat and the delta hedge is slightly negative in the single-asset benchmark. This shows that performance gains do not come solely from deep learning but from a model with

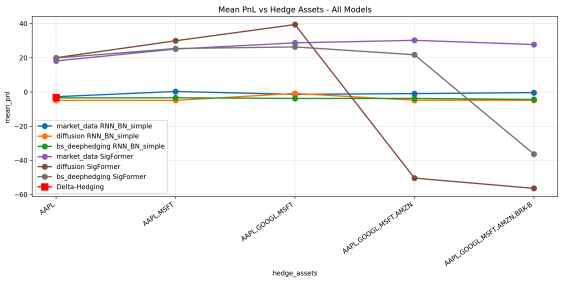

Figure 3. Comparison of Mean Hedging Performance Across Hedge Asset Sets

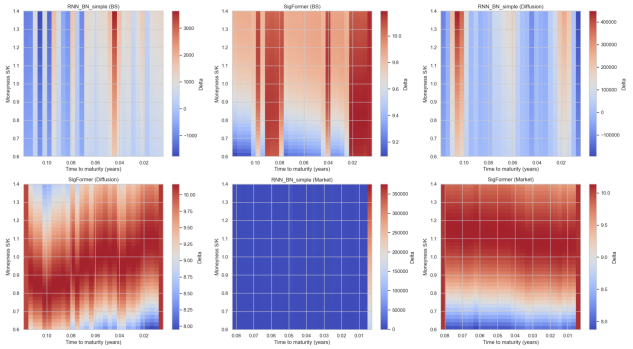

Figure 4. Comparison of learned holding Surfaces across Moneyness, Time to Maturity, and Models

sufficient capacity and an appropriate path representation. The RNN baseline is clearly under-parameterized for this task, which is in line with our expectations.

The Black-Scholes pipeline has slightly lower performance when evaluated on market data, reflecting model misspecification. We suspect that the inclusion of market regimes improves performance by increasing the resemblance between the simulated data and empirical market behaviour. Black-Scholes trained hedgers deteriorate rapidly as the hedge dimension increases, confirming that BS-generated data does not adequately represent real market dynamics, even with regime switching. While the diffusion-trained SigFormer performs well for small hedge sets, performance collapses for four and five hedge assets due to mismatched cross-asset dependence: diffusion models are trained independently per asset and therefore fail to preserve realistic joint dynamics. In contrast, the market-trained SigFormer remains stable across hedge sets,which corresponds to the fact that diffusion data are only a partial representation of actual markets.

The holding surfaces (Figure 4) reveal issues hidden by mean P&L. RNN hedges exhibit economically implausible magnitudes, with extreme leverage under both market- and diffusion-trained settings, indicating unstable policies. We attempted to constrain hedge weights to lie in $[0, 1]$, but this degraded performance. Importantly, under the entropic risk (exponential-utility) objective, the model is not designed to recover a delta-like hedge bounded between 0 and 1, but rather to maximize P&L relative to the option payoff, which naturally leads to unconstrained positions.

These large positions are encouraged by the combination of exponential-utility loss and wealth clipping for numerical stability: once losses exceed the clipping threshold, additional downside is no longer penalized, pushing the optimizer toward mean-seeking and unstable hedges. Since training dynamics under entropic risk are highly non-linear early epochs focus on mitigating rare tail losses, while later epochs emphasize mean P&L we train all models for only a limited number of epochs. Early stopping therefore acts as an implicit regularization mechanism, preserving tail-risk protection and improving robustness under distribution shift (see Appendix 8.4).

Finally, under the entropic risk objective, rare but severe upward price moves dominate the loss, making short positions extremely costly. This induces uniformly positive hedge positions near maturity, largely independent of moneyness, resulting in flat bands in the hedge surfaces.

6. Conclusion

In this work, we investigated deep hedging strategies trained under different data-generating processes and evaluated their performance exclusively on out-of-sample historical market data. While the results remain mixed and do not consistently outperform classical baselines, this outcome should be interpreted in light of the evaluation setting. Real financial markets exhibit strong non-stationarity, heavy tails, and complex cross-asset dependencies, making good hedging performance inherently difficult to achieve. Our results suggest that models trained on synthetic data, even when generated by more expressive mechanisms such as signature diffusions, struggle to fully transfer to true market dynamics. In contrast, market-trained models show more stable behavior. These findings highlight the gap between performance on simulated data and real-world applicability, and underline the importance of evaluating hedging strategies under realistic market conditions rather than relying solely on in-sample or synthetic test sets. Overall, our work brings forward the challenges of learning robust hedging strategies from data and points toward the need for richer joint data generation models, stronger regularization, and evaluation frameworks that better reflect the uncertainty and complexity of financial markets.

7. Ethical Risks

A central ethical risk in this project comes from the use of synthetically generated financial data to derive hedging strategies for risk management purposes. In particular, deep hedging policies learned using RNN and Transformer architectures on signature-diffusion-generated price paths may not fully capture the complexity, non-stationarity, and tail dependencies of real financial markets. The primary stakeholders affected by this risk are financial institutions and practitioners deploying such models. In the specific case of pension funds, this has a broader impact on millions of people. Strategies that appear robust in simulation may fail under real-world conditions, potentially leading to significant financial losses. As highlighted in (Bussmann et al., 2021), the opacity and data-dependence of AI-based risk models can increase existing weaknesses in traditional risk management frameworks, particularly during periods of market stress

This risk was evaluated through both a review of the relevant literature and empirical validation. Prior studies on deep hedging and neural generative models emphasize that learned strategies are sensitive to the assumed data-generating process and may suffer from distributional shift when applied outside their training domain (Buehler et al., 2019). In our project we choose to always test the hedging methods on real-world market data and not on generated data. This allows us to stress-test our hedging strategies. Nonetheless, fully eliminating this risk was constrained by limited historical data availability and computational complexity, which are acknowledged as limitations of the study.

8. Appendix

8.1. Black-Scholes and Market Regimes

We also generate simulated price trajectories using a regime-switching multi-asset framework based on Geometric Brownian Motion (GBM). This approach captures the time-varying volatility and correlation structures inherent in empirical financial data and varying market conditons, including rare ones.

8.1.1. Problem formulation

We consider a market with r risky assets whose price dynamics follow a regime-switching GBM. Let $\mathcal{K} = \{1, 2, ..., K\}$ denote the finite set of latent market regimes. Each regime $k \in \mathcal{K}$ is characterized by a distinct covariance structure $\Sigma_k \in \mathbb{R}^{r\times r}$, where $\Sigma_k \succ 0$ (Σ_k positive definite; required for Cholesky decomposition).

Under regime k, the price vector $\mathbf{S_t} = \left(S_t^{(1)}, ..., S_t^{(r)}\right)^T \in \mathbb{R}_+^r$ evolves according to the multivariate stochastic differential equation :

$$\frac{d\mathbf{S_t}}{\mathbf{S_t}} = \boldsymbol{\mu} dt + \mathbf{L}_k d\mathbf{W_t} \tag{2}$$

where $\boldsymbol{\mu} \in \mathbb{R}^r$ is the drift vector, $\mathbf{W_t}$ is an r -dimensional standard Brownian motion and $\mathbf{L}_k \in \mathbb{R}^{r\times r}$ is the matrix satisfying the Choleksy decomposition : $\mathbf{L}_k\mathbf{L}_k = \Sigma_k$.

8.1.2. Regimes Identification & Parameters Estimation

Let $\mathcal{D}_{train} = \{\mathbf{S_t}\}_{t=1}^{T_{train}}$ denote the historcial training data. To prevent look-ahead bias, we enforce a strict temporal split : $\mathcal{D}_{train}$ and $\mathcal{D}_{test} = \{\mathbf{S_t}\}_{t=T_{train+1}}^{T}$ are disjoint. All regime calibration is performed exclusively on $\mathcal{D}_{train}$.

The historical training prices are analyzed to identify latent market regimes.

Feature engineering : For each time t, we construct a feature vector $x_t \in \mathbb{R}^d$ from a value weighted portfolio, representing the overall US market. This feature vector captures market states, it is composed of : $x_t^{(1)} \equiv$ ann. asset returns, $x_t^{(2)} \equiv$ 30-days rolling window volatility, $x_t^{(3)} \equiv$ 60-days rolling window mean correlation and $x_t^{(4)} \equiv$ cross-sectional volatility. Finally, features are standardized via $\bar{\mathbf{x}}_t = \frac{\mathbf{x}_t - \boldsymbol{\mu}_x}{\boldsymbol{\sigma}_x}$, ensuring all features have the same scale when training the model to find the clusters.

Unsupervised Market Regime Classification : An unsupervised Gaussian Mixture Model (GMM) is fitted to the standardized feature vectors, the distribution of feature vec-

tors are modeled as :

$$p(\bar{\mathbf{x}}_t|\mathbf{\Theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\bar{\mathbf{x}}_t|\boldsymbol{\mu}_k, \Sigma_k)$$

where $\mathbf{\Theta} = \{(\pi_k, \boldsymbol{\mu}_k = \boldsymbol{\mu}, \Sigma_k)\}_{k=1}^{K}$, are the mixture parameters, $\pi_k \geq 0, \sum_{k=1}^{K} \pi_k = 1$, and $\Sigma_k \in \mathbb{R}^{d\times d}$ are the feature-space covariances. Since the GMM is fitted on standardized log-return–based features and assumes Gaussian regime-wise distributions, the regime detection aligns with the distributional assumptions used for path generation. The number of regimes K is selected using the Bayesian Information Criterion: $\mathrm{BIC}(K) = -2\log\hat{\mathcal{L}}_K + p_K \log(T)$, which balances goodness of fit and model complexity, preventing overfitting as the likelihood increases with the number of mixture components.

Regime Parameterization : Each observation t of the historical training data is assigned to regime $z_t = arg\max_k q_{k,t}$ where $q_{k,t}$ denotes the posterior regime probability given the parameters $\mathbf{\Theta} = \{\pi_k, \mu_k, \Sigma_k\}_{k\in[K]}$:

$$q_k(\bar{\mathbf{x}}_t) = p(k|\bar{\mathbf{x}}_t, \theta) = \frac{\pi_k \mathcal{N}(\bar{\mathbf{x}}_t|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K^*} \pi_j \mathcal{N}(\bar{\mathbf{x}}_t|\boldsymbol{\mu}_j, \Sigma_j)}$$

For each detected regime $k \in \{1, ..., K^*\}$, a regime-specific covariance matrix ($\hat{\Sigma}_k$) is calculated from the subset of historical returns classified under $regime_k$. Additionally, the drift vector ($\hat{\boldsymbol{\mu}}$) is estimated as the overall historical mean for each asset.

8.1.3. Multi-Asset Path Simulation

Synthetic price paths are generated for each of the N assets under the control of the regime-specific parameters.
For each regime $k \in K^*$, we generate $M = 20'000$ paths. The paths consist of the 30 days evolution of the stock price with dynamics described in the equation below. All stock price paths start at $S_0 = 100$ and are updated based on the Euler discretization of GBM :

$$\mathbf{S}_t = \mathbf{S}_{t-1} \odot \exp\left(\left(\hat{\boldsymbol{\mu}} - \frac{1}{2}\operatorname{diag}(\hat{\Sigma}_{k_m})\right)\Delta t + \mathbf{L}_{k_m}\mathbf{Z}_t\sqrt{\Delta t}\right)$$

where $\mathbf{Z}_t \sim \mathcal{N}(0, \mathbf{I}_N)$, $\Delta t = 1/365$ (1 day) and $\hat{\Sigma}_{k_m}, \mathbf{L}_{k_m}$ denote the lower triangular Cholesky matrix and the empirical covariance matrix of regime k_m.

8.1.4. Test data

In order to evaluate the out-of-sample performance of the deep hedging strategy, we construct test trajectories from the held-out portion of historical data $\mathcal{D}_{test}$. For each date t in the test period, we compute the feature vector $\mathbf{x}_t$ and assign a regime label $z_t \in \{1, \ldots, K^*\}$ using the trained GMM model and scaler from the training phase. We then construct overlapping 30-day price windows, normalizing each window to $S_0 = 100$ to match the training price paths. Thies yields a test set of real price trajectories $\{\mathbf{S}_{test}^{(m)}\}_{m=1}^{M_{test}}$ paired with their corresponding regime classifications. These serve as features to the network, effectively using the learned strategies under different regimes to make hedging decisions.

8.2. Log-Signature of a Path

Let $X : [0, T] \to \mathbb{R}^d$ be a continuous path of bounded variation. The *signature* of X over $[0, T]$ is defined as the sequence of iterated integrals

$$S(X)_{0,T} = \left(1,\ \int_0^T dX_t,\ \int_{0<t_1<t_2<T} dX_{t_1} \otimes dX_{t_2},\ \ldots\right) \in T((\mathbb{R}^d)), \tag{3}$$

where $T((\mathbb{R}^d))$ denotes the tensor algebra over $\mathbb{R}^d$.

The *log-signature* of X is defined as the logarithm of the signature in the tensor algebra,

$$\mathrm{LogSig}(X)_{0,T} = \log(S(X)_{0,T}) \in \mathcal{L}((\mathbb{R}^d)), \tag{4}$$

where $\mathcal{L}((\mathbb{R}^d))$ is the free Lie algebra generated by $\mathbb{R}^d$.

In practical applications, the log-signature is truncated at level m,

$$\mathrm{LogSig}^{(m)}(X)_{0,T} = \left(\ell^{(1)}, \ell^{(2)}, \ldots, \ell^{(m)}\right), \tag{5}$$

where the first-order term corresponds to the path increment,

$$\ell^{(1)} = \int_0^T dX_t, \tag{6}$$

and the second-order term is given by the iterated Lie bracket

$$\ell^{(2)} = \int_{0<t_1<t_2<T} [dX_{t_1}, dX_{t_2}]. \tag{7}$$

Higher-order terms involve nested Lie brackets of iterated integrals.

8.3. SigFormer

8.3.1. input representation:

At each trading time $t \in 0, \ldots, T$ we construct a state vector $x_t \in \mathbb{R}^{d_{\text{in}}}$ containing the returns available to the hedger. we then get a path of returns $\mathbf{x}0 = (x_0, \ldots, x_T)$. SigFormer outputs a sequence of hedge positions $h_t \in \mathbb{R}^{d\text{out}}$ (one per time step), which we interpret as the number of shares held in each hedging instrument.

$$\left(\mathcal{S}_t^{(1)}, \ldots, \mathcal{S}_t^{(m)}\right) = \mathrm{Sig}^{(m)}\big(\mathrm{Proj}(\mathbf{x}_{0:t})\big), \qquad t = 0, \ldots, T. \tag{8}$$

This provides a graded, tensor-valued representation at each time step, with total feature size $\sum k = 1^m d^k$.

8.3.2. SIGFORMER ARCHITECTURE

SigFormer processes the signature tensors with B repeated tensor-attention blocks. Concretely:

1. **Linear projection:** each $x_t \in \mathbb{R}^{d_{\text{in}}}$ is mapped to $\tilde{x}_t \in \mathbb{R}^d$.
2. **Signature layer:** compute the streaming signatures $(\mathcal{S}_t^{(1)}, \ldots, \mathcal{S}t^{(m)})$ for all t.
3. **Tensor causal self-attention:** for each level $k = 1, \ldots, m$, we apply multi-head causal self-attention with a lower-triangular mask (no look-ahead). Attention is applied separately per signature level to respect the tensor structure, following (Tong et al., 2023).
4. **Tensor MLP + residuals:** each attention block is followed by a position-wise MLP (again per level), with residual connections, dropout, and layer normalisation.
5. **Flatten + readout:** concatenate all signature levels into a single vector and apply a linear readout to obtain hedge positions:

$$\sum_{t=0}^{T-1} \left\langle h_t^\theta, ; S_{t+1} - S_t \right\rangle . \tag{9}$$

8.3.3. HYPERPARAMETERS

Due to computational constraints, we did not perform cross-validation or extensive hyperparameter tuning; instead, we deliberately chose a compact SigFormer configuration to limit model size and reduce overfitting risk. We use the following SigFormer configuration in our experiments: model dimension $d =$ `model_dim`, signature truncation order $m =$ `order`, number of attention heads `n_heads`, number of blocks `n_blocks`, learning rate `lr`, and risk aversion $\lambda =$ `risk_aversion`. In our main setup we set `model_dim` $= 8$, `order` $= 3$, `n_heads` $= 2$, `n_blocks` $= 2$, `lr` $= 10^{-3}$, and $\lambda = 1.3$.

Remark. Compared to RNN-based hedgers, SigFormer benefits from (i) signature features that encode higher-order path information and (ii) attention mechanisms that can model long-range temporal dependencies, which is particularly relevant when hedging under non-stationary or regime-dependent dynamics (Tong et al., 2023).

8.4. Training dynamics under the entropic risk objective

Let $\phi = (\phi_t)_{t=0}^{N-1}$ denote the hedging strategy and define the terminal wealth

$$X(\phi) = \sum_{t=0}^{N-1} \phi_t \Delta S_{t+1} - (S_T - K)^+, \tag{10}$$

where $\Delta S_{t+1} = S_{t+1} - S_t$. The model is trained by minimizing the entropic risk functional

$$\mathcal{L}(\phi) = \frac{1}{\lambda} \log \mathbb{E}\left[\exp\left(\lambda X(\phi)\right)\right], \tag{11}$$

which is equivalent (up to monotone transformations) to exponential utility maximization.

Gradient structure: The gradient of $\mathcal{L}$ with respect to the terminal wealth satisfies

$$\frac{\partial \mathcal{L}}{\partial X} = \frac{\exp(\lambda X)}{\mathbb{E}[\exp(\lambda X)]}, \tag{12}$$

so that paths with extreme values of X receive exponentially larger weight during optimization. Consequently, the training dynamics are initially dominated by tail outcomes of the terminal P&L distribution.

Early training regime: tail-risk domination: At initialization, the hedge positions ϕ_t are close to zero, and terminal wealth is dominated by the option payoff. For a short call position, the worst outcomes correspond to large upward price moves, where $S_T \gg K$ and

$$X(\phi) \approx -(S_T - K).$$

In these scenarios, X is strongly negative, causing the exponential weighting to concentrate on paths with $\Delta S > 0$. As a result, the gradient of $\mathcal{L}$ with respect to ϕ_t is positive across time steps,

$$\nabla_{\phi_t} \mathcal{L} \;\propto\; \mathbb{E}[\Delta S_{t+1} \exp(\lambda X)] > 0, \tag{13}$$

which drives the optimizer toward uniformly positive hedge positions. This explains the emergence of "long-everywhere" hedging strategies after very few training epochs.

Later training regime: mean-dominated optimization: After several epochs, the model has substantially reduced extreme tail losses, and the distribution of $X(\phi)$ becomes more concentrated. In this regime, the entropic risk admits the second-order expansion

$$\mathcal{L}(\phi) \approx \mathbb{E}[X(\phi)] + \frac{\lambda}{2} \mathrm{Var}(X(\phi)), \tag{14}$$

so that optimization increasingly emphasizes improvements in expected terminal wealth. For a short call option, this favors strategies that exploit the convex option payoff by shorting the underlying asset, which improves the mean P&L once tail risk is controlled. Consequently, the gradient reverses sign and the learned hedge converges toward uniformly negative positions.

Abrupt regime transition: The sharp transition between the two regimes is a consequence of the strong nonlinearity of the log-sum-exp objective and the high expressive power of the SigFormer architecture. Once extreme losses are sufficiently mitigated, the optimization rapidly shifts from tail-risk minimization to mean P&L optimization, resulting in a qualitative change in the learned hedging strategy.

8.5. Delta Hedging

We compare our hedging strategies to the well known Delta Hedging. In this approach one calculates the sensitiy of the option price $P(t, S_t)$ to a change in the underlying asset price. This sensitivity is called the Delta : $\Delta = \frac{\partial P}{\partial S}$ and tells one how many shares of the underlying it should acquire to mitigate the risks of the option position. The hedging strategy requires continuous rebalancing of the portfolio by holding Δ_t shares of the underlying asset at each time t+.

References

Barancikova, B., Huang, Z., and Salvi, C. SigDiffusions: Score-Based Diffusion Models for Time Series via Log-Signature Embeddings. *arXiv preprint arXiv:2406.10354*, 2024.

Black, F. and Scholes, M. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81 (3):637–654, 1973. URL http://www.jstor.org/stable/1831029.

Buehler, H., Gonon, L., Teichmann, J., and Wood, B. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

Bussmann, N., Giudici, P., Marinelli, D., and Papenbrock, J. Artificial intelligence and machine learning in risk management—current challenges and ethical considerations in the financial sector. *Risk Management*, 23:1–16, 2021.

He, G., Sutter, T., and Gonon, L. Distributional Adversarial Attacks and Training in Deep Hedging. *arXiv preprint arXiv:2508.14757*, 2025.

Heston, S. L. A closed-form solution for options with stochastic volatility, with applications to bond and currency options. *The Review of Financial Studies*, 6(2): 327–343, 1993. doi: 10.1093/rfs/6.2.327.

Tong, A., Nguyen-Tang, T., Lee, D., Tran, T., and Choi, J. SigFormer: Signature Transformers for Deep Hedging. *arXiv preprint arXiv:2310.13369*, 2023. URL https://arxiv.org/abs/2310.13369.

Yuan, X. and Qiao, Y. Diffusion-ts: Interpretable diffusion for general time series generation. 2024. URL https://arxiv.org/abs/2403.01742.