ITESO

Universidad Jesuita
de Guadalajara

004 DL TRADING WITH MLOps EXECUTIVE REPORT

ITESO UNIVERSIDAD JESUITA DE GUADALAJARA

HÉCTOR SEBASTIÁN CASTAÑEDA ARTEAGA

DAVID ROGELIO CAMPOS MURIÁ

MICROSTRUCTURE & TRADING SYSTEMS

MTRO. LUIS FELIPE GÓMEZ ESTRADA

MARTES 28 DE OCTUBRE DEL 2025

Contents

1. Strategy Overview

1.1 Overall Objective of the Strategy

The primary objective of this project is to design and evaluate a systematic trading strategy on the stock NVDA, using deep learning models capable of extracting temporal patterns from 15 years of daily market data. The strategy relies on one-dimensional convolutional neural networks (1D CNNs) that take sequences of engineered technical indicators as input and generate discrete trading signals of three types: Short (−1), Hold (0), and Long (+1). These signals are fed into a backtesting engine that incorporates realistic transaction costs to estimate the net economic viability of the strategy after commissions and borrowing costs associated with short positions.

Functionally, the architecture is designed to cover the full lifecycle of a modern quantitative strategy: from time series feature engineering (momentum, volatility, volume and price-based indicators), through the training and comparison of multiple CNN architectures, experiment tracking with MLFlow, and the deployment of a prediction API, to data drift monitoring and backtesting under realistic assumptions regarding costs and risk management. The goal is not only to produce attractive historical results, but to build a reproducible and extensible workflow aligned with MLOps best practices in algorithmic trading.

The choice of NVDA as the underlying asset is motivated by both economic and methodological reasons. NVDA is a highly liquid technology stock that, over the last decade, has exhibited episodes of elevated volatility, strong upward trends and pronounced regime shifts. These characteristics make it a demanding testbed for deep learning models: on the one hand, they provide opportunities to capture significant directional moves; on the other, they stress-test the robustness of the model in the presence of extreme environments, bubbles and sharp corrections. The ultimate aim is to assess whether, under such conditions, a CNN-based strategy can deliver attractive risk-adjusted returns once operational frictions and practical implementation constraints are considered.

1.2 Deep learning approach to trading signal prediction

The strategy is built around a sequence-based deep learning setup. Instead of treating each trading day in isolation, the model ingests sliding windows of 10 consecutive trading days, where each day is represented by a vector of 28 normalized technical features (momentum, volatility, volume and price-related indicators). Each input sample to the CNN is therefore a small “image” in time: a 10×28 matrix that captures short-term market dynamics. The convolutional layers slide filters across this temporal axis to detect local patterns such as short rallies, volatility clusters, or volume spikes that may anticipate future price moves.

The model is trained in a multi-class classification setting, where the output is a categorical prediction over three possible trading actions: −1 (Short), 0 (Hold), and +1 (Long). For each 10-day window, the target label is derived from the future return of NVDA over a fixed horizon, mapped into one of these three classes according to pre-defined thresholds. The network learns to associate specific temporal configurations of technical indicators with the most likely trading action, producing at each step a probability distribution over the three classes and a final discrete signal given by the most probable one.

To ensure a realistic evaluation, the model is trained under a supervised learning scheme with a chronological 60/20/20 split into training, validation and test sets, strictly preserving the time order of observations to avoid any form of look-ahead bias. All hyperparameter tuning and model selection decisions are based on performance in the validation period, while the test period is reserved exclusively for final out-of-sample assessment. Throughout this process, MLFlow is used as the backbone for experiment tracking: each training run logs the architecture used, hyperparameters, class weights, training and validation metrics (accuracy, loss, F1-score), and associated artefacts such as learning curves and confusion matrices. This MLOps-oriented setup makes the experimentation process transparent, reproducible and easy to extend in future iterations of the strategy.

1.3 Why multiple CNN architecture used

The project deliberately explores three different CNN architectures in order to study how model complexity affects both predictive performance and trading outcomes:

- **SimpleCNN** – a compact baseline model, with a small number of convolutional and dense layers. It provides a reference point for what can be achieved with a relatively low–capacity architecture.
- **DeepCNN** – a deeper network with more convolutional layers and higher filter counts, designed to increase representational capacity and potentially capture more complex temporal patterns in the data.
- **CustomCNN** – a **multi–scale architecture** with three parallel convolutional branches using different kernel sizes. This design aims to capture patterns at short, medium and slightly longer horizons simultaneously, before merging them into a shared representation.

The rationale behind this design is to systematically evaluate the trade-off between expressive power and overfitting risk. In principle, deeper and more sophisticated architectures can approximate more complex relationships between indicators and future returns, but they are also more prone to fitting noise, especially in financial time series where the signal-to-noise ratio is low. By training and evaluating all three models under the same data splits and feature set, the project allows for an empirical model selection based on:

1. **Classification metrics** on the held-out test set (accuracy, precision, recall, F1-score).
2. **Downstream trading performance** in the backtest (Sharpe ratio, drawdowns, total return, stability across periods).

Table 1 summarizes the test-set performance of the three architectures. While SimpleCNN attains the highest test accuracy (≈0.458), the CustomCNN model achieves the best test F1-score (≈0.335), indicating a slightly better balance between precision and recall across the three classes. DeepCNN does not provide a significant improvement over the other two and, in some metrics, underperforms them. Given the importance of handling class imbalance and correctly identifying minority classes (especially Short), the final model selected for backtesting and deployment is CustomCNN, prioritizing F1-score and robustness over marginal gains in raw accuracy.

Table 1 – Test metrics by CNN architecture

Model	Test Accuracy	Test F1-score
SimpleCNN	0.458	0.324
DeepCNN	0.436	0.314
CustomCNN	**0.418**	**0.335**

1.4 Expected advantages and limitations

Expected advantages

From a modelling perspective, the proposed setup offers several potential benefits:

- **Ability to capture local and multi-scale patterns**

  By operating on 10-day sequences of technical features, 1D CNNs are well suited to detect **local temporal structures** such as short rallies, pullbacks, volatility bursts or volume spikes. In the case of the CustomCNN architecture, the use of **multiple convolutional branches with different kernel sizes** further enhances the ability to capture patterns at different time scales within the same model.

- **Learning non-linear relationships between indicators**

  Traditional rule-based strategies often combine indicators like RSI, MACD, ATR or OBV through simple thresholds and linear conditions. In contrast, the CNN learns non-linear interactions between momentum, volatility and volume features that would be difficult to hand–engineer. This opens the door to exploiting subtle configurations of indicators that precede significant price moves but are not obvious from isolated indicator readings.

- **Reproducible, MLOps-oriented framework**

The integration of **MLFlow** for experiment tracking and a modular backtesting engine provides a robust framework for systematic research. Every model run is logged with its hyperparameters, metrics and artefacts, making it straightforward to:

- compare architectures and configurations,
- schedule periodic retraining as new data becomes available, and
- extend the pipeline to other assets or model families while preserving full reproducibility.

In combination, these elements turn the strategy into a research platform rather than a one-off model, which is valuable both academically and for potential production use.

Limitations

At the same time, the approach faces several important limitations that must be acknowledged:

- **Risk of overfitting noisy financial series**

  Financial time series are notoriously noisy, with a low and unstable signal-to-noise ratio. Deeper architectures such as DeepCNN and CustomCNN have enough capacity to memorize idiosyncratic patterns of the training period, leading to optimistic in-sample results that do not fully carry over to validation and test. The contrast between the extremely strong performance in the training backtest and the much weaker performance in validation is a clear symptom of this overfitting risk.

- **Single-asset exposure (lack of diversification)**

  The strategy is designed and evaluated exclusively on **NVDA**, which means that all performance and risk estimates are tied to the specific behavior of this stock. There is no diversification across sectors, factors or instruments. As a result, any structural change affecting NVDA (e.g. company-specific news, regulatory shifts, sector rotations) can directly and heavily impact the strategy.

- **Sensitivity to regime shifts and fat tails**

The data drift analysis and the back testing results both suggest that the model is sensitive to regime changes. Large drawdowns in the back tests (up to approximately −59 % in the training period and around −50 % in the test period) reflect the impact of extreme events and fat-tailed return distributions. When volatility regimes change abruptly or when rare but severe events occur, the model's learned patterns may no longer be valid, exposing the strategy to significant temporary or even permanent losses.

- **Moderate classification accuracy and implications for risk management**

  Across the different architectures, test-set classification accuracy remains in a moderate range (roughly 40–46 %), even for the best-performing models. While this level of accuracy can still be compatible with profitable trading (given asymmetric payoffs and favorable gain/loss profiles), it also implies that a substantial fraction of signals will be wrong. This fact demands a conservative risk management framework: strict position sizing, robust stop-loss rules and realistic expectations about drawdowns and performance variability.

The strategy combines promising modelling capabilities with non-trivial risks typical of data-driven trading systems. The rest of the report quantifies how these advantages and limitations materialize in terms of classification metrics, data drift behavior and, ultimately, trading performance.

2. Feature Engineering

The performance of any deep learning strategy in trading depends critically on the quality and diversity of its input features. In this project, raw OHLCV data for NVDA is transformed into a rich set of technical indicators that seek to summarize price dynamics, volatility regimes and trading activity. All of these features are later normalized and arranged into 10-day sequences to feed the CNN models.

2.1 Momentum, volatility and volume indicators

The feature set is organized around three main dimensions: momentum/trend, volatility and volume/participation. In total, more than twenty indicators are computed, providing a multi–faceted view of recent market behavior.

Momentum and trend indicators

To capture directional bias and overbought/oversold conditions, the strategy includes:

- **Relative Strength Index (RSI)** at two horizons (e.g. 14 and 21 days), reflecting short- and medium-term momentum and exhaustion points.
- **MACD components**: MACD line, signal line and their difference, summarizing the interaction between faster and slower moving averages and helping identify trend reversals.
- **Rate of Change (ROC)**, measuring the percentage change in price over a recent window to quantify how fast the assets have moved.
- **Stochastic oscillator** (%K and %D), comparing the current close with the recent high–low range, useful for identifying momentum extremes within a trading range.
- **Average Directional Index (ADX)**, which measures the strength of the prevailing trend independently of its direction.

These indicators provide the CNN with a compact description of where the price has been moving, how quickly, and with what trend strength, within each 10-day window.

Volatility indicators

To characterize the risky environment and the stability of price movements, several volatility-related features are engineered:

- Bollinger Bands (upper, lower and middle band), together with:
    - band width (relative distance between the bands), and
    - the position of the closing price inside the band (e.g. closer to the upper or lower band).
- Average True Range (ATR), both in absolute terms and as a percentage of price, capturing typical daily ranges and providing a natural scale for risk and stop levels.

- A rolling measure of historical volatility, computed from recent daily returns and annualized.

These indicators allow the model to distinguish between quiet regimes, volatility expansions and stressed conditions, which are all highly relevant for the profitability and risk of any trading strategy.

Volume and participation indicators

Volume-based features are included to reflect how strongly the market is participating in price moves:

- **On Balance Volume (OBV)**, which accumulates volume signed by the direction of daily returns, approximating whether capital is flowing into or out of the asset.
- **Volume Rate of Change**, comparing current volume to its level several days ago to highlight unusual activity.
- **Moving averages of volume** (e.g. 20-day average) and the ratio of current volume to its moving average, signaling volume spikes or dry-ups relative to normal conditions.
- **Volume-Weighted Average Price (VWAP)** over a rolling window, along with the percentage distance between the current price and VWAP, providing a reference for whether the asset is trading at a premium or discount to its recent volume-weighted fair value.

Taken together, these momentum, volatility and volume indicators produce a highly informative snapshot of NVDA's recent behavior for every day in the dataset. When stacked into 10-day windows, they form the 10×28 matrices that serve as inputs to the CNNs, enabling the models to learn patterns that involve not just price direction, but also the intensity, risk and participation behind each move.

2.2 Normalization and feature scaling

Because the feature set combines variables on very different scales—prices in absolute dollars, indicators bounded between 0 and 100, volatility in percentages and volume in raw units—all inputs are normalized using statistics computed exclusively on the training set. For each technical feature, summary statistics (such as mean and

standard deviation, or equivalent scaling parameters) are estimated on the training period and then used to transform that feature into a scaled, dimensionless version. The same transformation is subsequently applied to the validation and test sets, ensuring that no information from future data leaks into the training process.

This normalization step is critical for two reasons. First, it prevents features with naturally larger magnitudes (for example, raw price or volume) from dominating the gradients during optimization, which would make it harder for the network to learn from subtler but important indicators such as RSI or band position. Second, by bringing all inputs onto a comparable numerical scale, normalization improves numerical stability and typically accelerates convergence, allowing the optimizer to navigate the loss surface more efficiently.

Once normalized, the daily feature vectors are stacked into 10-day sequences, forming matrices of shape 10×28 that serve as inputs to the 1D CNNs. Each such matrix represents a short temporal window of NVDA's recent behavior in a numerically well-conditioned format, facilitating the extraction of meaningful patterns by the convolutional layers.

2.3 Class imbalance strategy

Financial markets do not generate trading opportunities in a balanced way across all possible actions. When labels are defined as Short (−1), Hold (0) and Long (+1) based on future returns and thresholds, the resulting distribution in the training set is naturally imbalanced:

- Long signals tend to be relatively frequent in stocks with a strong long-term uptrend like NVDA.
- Hold signals appear whenever the expected move is small or ambiguous.
- Short signals are comparatively rare, as sustained downward moves are less common and often shorter in duration.

If this imbalance is ignored, a neural network can achieve deceptively high accuracy by simply focusing on the majority class, for example predicting Long most of the time and almost never signaling Short. This behavior is visible in the confusion matrices:

even the better models show a clear tendency to overpredict the Long class and underrepresent Short and, especially, Hold.

To mitigate this problem, the training process incorporates a class weighting scheme. The idea is to assign a larger weight to errors on minority classes and a smaller weight to errors on the majority class. In practice, weights are computed from the class frequencies in the training set so that each class contributes more evenly to the overall loss:

- Misclassifying a rare Short or Hold example incurs a higher penalty in the loss function.
- Misclassifying a frequent Long example is penalized less, relative to its abundance.

By doing so, the optimizer is encouraged to learn decision boundaries that pay attention to all three classes, rather than collapse onto the majority. This typically reduces raw accuracy slightly but improves more informative metrics such as F1-score, especially for the minority classes, which is crucial in trading: missing short signals in adverse regimes or Hold signals when the edge is weak can be costly.

3. Target Variable Definition

3.1 Trading signal labels: Long (1), Hold (0), Short (−1)

The target variable is defined as a three-class trading signal that encodes the directional decision of the strategy at each point in time. For every 10-day input window, the model is asked to predict one of the following actions:

- **Short (−1)** – open or maintain a short position in NVDA, expecting a significant downward move.
- **Hold (0)** – stay out of the market (in cash) or close any open position, when the expected move is not large enough or is too uncertain.
- **Long (1)** – open or maintain a long position in NVDA, expecting a meaningful upward move.

This design turns the problem into a multi-class classification task rather than a regression problem. Instead of predicting a numeric return directly, the model learns to map patterns in the 10-day feature window into one of these three discrete trading actions, which are later translated into positions in the backtesting engine.

3.2 Threshold logic for class assignment

The signals are derived from forward returns over a fixed horizon, using a simple but economically meaningful rule. For each day *t*:

1. A **forward return** is computed over a **5-day horizon**, as the percentage change between the closing price at day *t* and the closing price at day *t+5*.
2. This forward return is compared to a **symmetric threshold of ±2%**:
    - If the forward return is **greater than +2%**, the move is considered a **relevant upward opportunity**, and the target is labelled **Long (1)**.
    - If the forward return is **less than −2%**, the move is considered a **relevant downward opportunity**, and the target is labelled **Short (−1)**.
    - If the forward return lies between **−2% and +2%**, the move is regarded as too small or noisy, and the target is labelled **Hold (0)**.

Conceptually, this rule encodes the idea that the strategy should only commit capital when the expected move is large enough to potentially overcome transaction costs and compensate for risk. Small, anticipated moves are filtered out and treated as Hold situations.

Because the 5-day forward return is used to define the label, the last few rows of the dataset (for which the future price at *t+5* is unknown) are removed to avoid using incomplete information. The resulting labelled dataset is then split chronologically into training, validation and test sets.

3.3 Distribution of signals in the training data

The application of this threshold-based labelling scheme on NVDA's 15-year history produces a naturally imbalanced distribution of classes in the training set:

- **Long (1)** is the **most frequent** class, reflecting the strong upward drift of NVDA over the sample period: there are many 5-day windows where the forward return exceeds +2%.
- **Hold (0)** appears in a substantial fraction of observations, corresponding to periods where price changes remain within the ±2% band and the expected edge is small or ambiguous.
- **Short (−1)** is the **least frequent** class, as extended downward moves of more than −2% over 5 days occur less often than upward swings in this stock.

This empirical distribution is consistent with the economic nature of the asset and directly motivates both:

- the class weighting strategy described previously (to prevent the model from ignoring Short and Hold signals), and
- the interpretation of the confusion matrices later in the report, where a clear tendency to overpredict Long and underrepresent Short/Hold can be observed.

The target construction combines a forward-looking, economically grounded definition of signals (via 5-day returns and ±2% thresholds) with a transparent understanding of how often each type of opportunity appears in the historical data.

4. Model Architecture and Design

4.1 General structure of the CNN models

All three models in the project share a common architectural backbone, tailored to the structure of the input data and the nature of the prediction task.

At each step, the network receives as input a tensor of shape (10 timesteps, 28 features). Each of the 10 rows corresponds to one trading day, and each column to a normalized technical indicator (momentum, volatility, volume or price-based). This 10×28 matrix can be viewed as a short “image in time”, on which the convolutional layers operate.

Across the three architectures (SimpleCNN, DeepCNN and CustomCNN), the core building blocks are:

- **1D convolutional layers with ReLU activation**

  Multiple Conv1D layers slide filters along the temporal axis (the 10 days), learning to detect local temporal patterns such as short rallies, pullbacks, volatility spikes, or combinations of indicators that consistently precede certain types of moves. ReLU activations introduce non-linearity while keeping the computation efficient and stable.

- **Pooling layers to summarize temporal information**

  Depending on the architecture, the convolutional blocks are followed by MaxPooling1D and/or GlobalAveragePooling1D layers.

  - MaxPooling1D progressively reduces the temporal dimension while retaining the strongest activations, focusing the representation on the most salient local patterns.

  - GlobalAveragePooling1D aggregates information over the entire 10-day window into a compact vector, acting as a bridge between temporal feature extraction and the final decision layers.

- **Fully connected (dense) layers for decision making**

  After the convolutional and pooling stages, the resulting feature map is flattened or globally pooled and passed through one or more dense layers. These layers combine the extracted features into a higher-level representation that is then fed to a final output layer with three units and softmax activation, producing a probability distribution over the three classes: Short, Hold and Long.

- **Regularization mechanisms: Dropout, L2 and Batch Normalization**

  To mitigate overfitting, especially in the deeper architectures, the models incorporate several regularization techniques:

  - **Dropout** layers randomly deactivate a fraction of neurons during training, preventing the network from relying on a small subset of features and encouraging more robust, distributed representations.

- **L2 weight regularization** constrains the magnitude of the weights, discouraging overly complex solutions that fit noise in the training data.
- **Batch Normalization** stabilizes the distribution of activations across layers, improving convergence and making training less sensitive to initialization and learning rate choices.

The CustomCNN model extends this general pattern with a multi–scale design: three convolutional branches operate in parallel on the same (10, 28) input, each with a different kernel size, capturing patterns at slightly different temporal resolutions. The outputs of these branches are then merged and passed through additional convolutional, pooling and dense layers before reaching the softmax output. This structure is intended to better exploit the multi-horizon nature of market dynamics, where both very short-term and slightly longer patterns can be informative.

4.2 Training specifications

The three CNN architectures are trained under a **supervised multi-class classification setup** with a consistent training protocol, designed to balance learning capacity and regularization.

- **Loss function – sparse categorical cross-entropy with class weights**

  The optimization objective is sparse categorical cross-entropy, which is appropriate for multi-class problems with integer labels (−1, 0, 1 mapped internally to three classes). To address the class imbalance in the target variable, class weights are applied so that misclassifying minority classes (especially Short) incurs a higher penalty than misclassifying the majority long class. This aligns the optimization process with the economic importance of correctly identifying less frequent but high-impact signals.

- **Optimizer – Adam with fixed learning rate**

  All models are trained using the Adam optimizer with a fixed learning rate in a standard range for deep learning on tabular/sequence data. Adam’s adaptive updates help stabilize training across features with different scales and dynamics, while the fixed rate avoids excessive complexity in tuning. The same

optimizer configuration is used across architectures to ensure a fair comparison.

- **Batch size and number of epochs**

  The batch size and number of epochs are chosen empirically to provide sufficient training without overfitting. In practice, a moderate batch size (e.g. 64) and 15–20 epochs are used as a compromise between convergence speed and generalization. Training is monitored on the validation set at the end of each epoch, and the evolution of metrics is logged for later analysis.

- **Real-time validation and MLFlow logging**

  During training, validation metrics are computed after each epoch on the chronologically separated validation set. At the same time, MLFlow automatically records:

  - the model architecture and hyperparameters,
  - the training and validation accuracy and loss per epoch,
  - and artefacts such as learning curves and confusion matrices. This setup ensures that each training run is fully reproducible and that model selection is based on objective, recorded evidence rather than ad-hoc impressions.

Figure 1 – Training curves for the CustomCNN model

- **Left panel:** *Training vs. validation accuracy* over epochs (CustomCNN_history – accuracy plot).

- **Right panel:** *Training vs. validation loss* over epochs (CustomCNN_history – loss plot).

These curves exhibit a characteristic pattern:

- The training accuracy steadily increases, reaching approximately 55–57 % by the final epochs.
- The validation accuracy improves initially but then stabilizes around 40–43 %, indicating that the model's ability to generalize does not keep growing beyond a certain point.
- In parallel, the validation loss stops decreasing and may start to rise slightly, signaling the onset of overfitting: the model continues to fit the training data more closely without achieving corresponding gains on unseen data.

This behavior is typical in financial time series modelling and underscores the importance of using validation-based model selection, regularization (Dropout, L2, BatchNorm) and, in future extensions, potentially early stopping to halt training once validation performance plateaus.

4.3 Class-weighting scheme and its effect on learning

Given the imbalanced distribution of targets (Long much more frequent than Hold and especially Short), the training loss incorporates a **class-weighting scheme** so that each class contributes more fairly to the optimization:

- Let $n_{\text{short}}, n_{\text{hold}}, n_{\text{long}}$ be the number of training examples in each class.
- Let $N = n_{\text{short}} + n_{\text{hold}} + n_{\text{long}}$ be the total number of training samples.
- For each class c, a weight w_c is defined as inversely proportional to its frequency, so that rarer classes receive larger weights. In practice, weights follow the standard pattern

$$w_c \propto \frac{N}{K \cdot n_c},$$

where K is the number of classes (3 in this project).

When the sparse categorical cross-entropy loss is computed, each example is multiplied by the weight corresponding to its true class. This has two direct consequences:

- Errors on rare Short or Hold examples receive a larger penalty, pushing the model to pay attention to patterns that precede these less frequent but crucial situations.
- Errors on the abundant Long examples are relatively less penalized, preventing the optimizer from converging to a trivial “always Long” solution that would show acceptable accuracy but very poor risk-adjusted performance.

Empirically, the use of class weights shifts the behavior of the models in two ways:

1. **Classification metrics become more balanced.**

   Although overall accuracy remains in a moderate range (around 0.42–0.46 on the test set across architectures), the **F1-scores improve**, particularly for the minority classes, compared to an unweighted baseline.

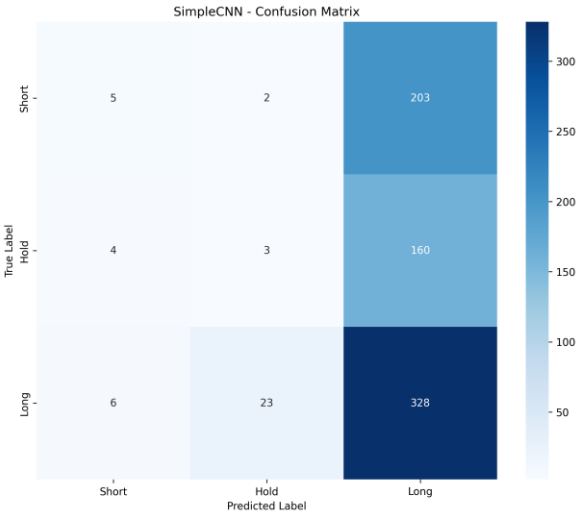
2. **Confusion matrices reveal less extreme bias toward Long.**

   All models still show a clear tendency to predict Long more often than Short or Hold, but the proportion of correctly identified Short and Hold cases increases relative to a naive, unweighted training.

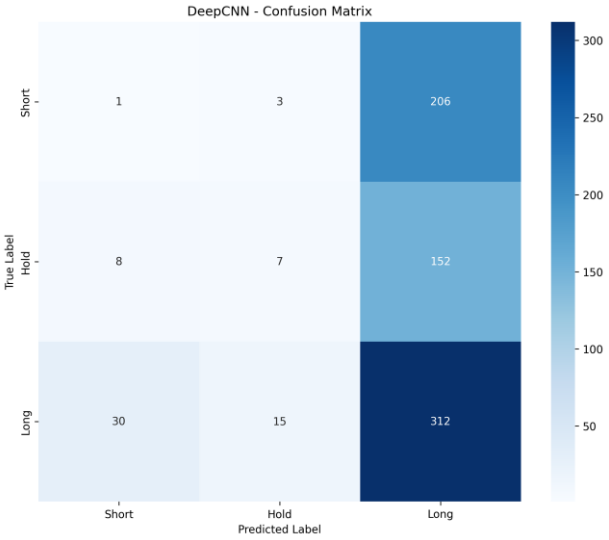
   - These effects can be visually inspected in the confusion-matrix figures:

Figure 2 – Confusion matrices for the three CNN architectures

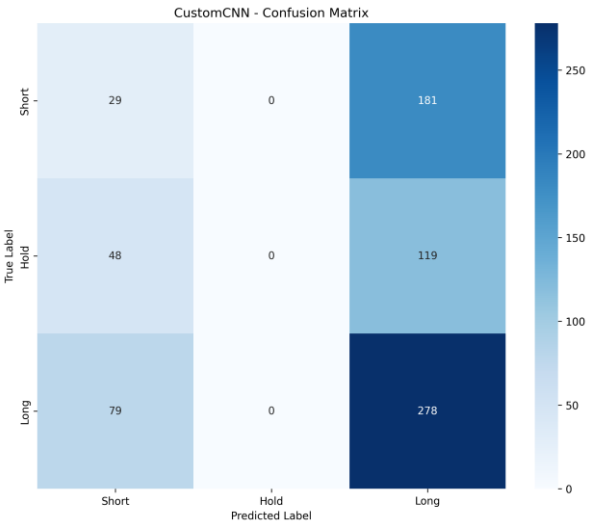
Simple CNN



DeepCNN



CustomCNN



Overall, the class-weighting scheme does not eliminate imbalance in the predictions, but it moves the optimization objective closer to the economic reality of the strategy, where missing rare but important Short or Hold signals can have a disproportionate impact on risk and returns.

5. MLFlow Experiment Tracking

5.1 Experiment setup and logging structure

All training and evaluation runs are organized under a single experiment dedicated to the NVDA CNN strategy. Each combination of architecture and hyperparameters (SimpleCNN, DeepCNN, CustomCNN) is treated as an individual run, with its own configuration and metric history.

For every run, the tracking layer stores:

- the **model configuration** (name of the architecture, number and size of convolutional and dense layers, pooling strategy, regularization settings),
- the **data configuration** (length of the look-back window, number of input features, chronological train/validation/test split),
- the **optimization settings** (optimizer, learning rate, batch size, number of epochs, class weights),
- and a complete set of **training and validation metrics** per epoch, together with the final performance on the test set and the serialized version of the trained model.

This structure makes each experiment fully reproducible and allows systematic comparison of alternative designs.

5.2 Parameters tracked for each model

For transparency and future maintenance, a compact but expressive set of parameters is logged for every run:

- **Model-level parameters**

Architecture name; number of convolutional blocks; filter sizes and kernel widths; type and placement of pooling layers; use of Dropout, L2 regularization and Batch Normalization.

- **Training configuration**

  Loss function (sparse categorical cross-entropy with class weights); optimizer (Adam) and learning rate; batch size; number of epochs; exact class-weight values used during optimization.

- **Data configuration**

  Window length of 10 trading days; 28 engineered and normalized features per day; indices and sizes of the three chronological segments (training, validation, test).

These parameters are sufficient to reconstruct the full training environment of any run, which is essential both from an academic and from an operational perspective.

5.3 Metrics and diagnostic artefacts

The experiment tracking captures both numerical metrics and visual diagnostics:

- **Per-epoch metrics**
  - Training and validation accuracy.
  - Training and validation loss.
- **Final test-set metrics**
  - Overall accuracy.
  - Macro-averaged precision, recall and F1-score for the three classes (Short, Hold, Long).
- **Diagnostic figures**
  - Learning-curve plots showing the evolution of accuracy and loss on training and validation.

- Confusion-matrix heatmaps for each architecture, where rows correspond to true labels and columns to predicted labels.

- **Backtesting performance metrics**

  For the selected model, profitability and risk indicators are also tracked on each segment (train, validation, test), including total return, number of trades, win rate, Sharpe ratio, Sortino ratio, Calmar ratio and maximum drawdown.

Table 2 – Test-set classification metrics by architecture

Model	Test Accuracy	Test Precision	Test Recall	Test F1-score
SimpleCNN	0.458	0.351	0.458	0.324
DeepCNN	0.436	0.298	0.436	0.314
CustomCNN	0.418	0.287	0.418	0.335

5.4 Model comparison and selection

The tracking interface allows **side-by-side comparison** of all runs. Model selection follows a two-step logic:

1. **Statistical performance.**

   The three architectures are first compared on the basis of their out-of-sample classification metrics. SimpleCNN achieves the highest test accuracy, but CustomCNN exhibits the highest F1-score, indicating that it handles the minority classes (Short and Hold) more effectively. The confusion matrices confirm that CustomCNN, while still biased toward Long, commits fewer extreme misclassifications than the other two models.

2. **Economic performance.**

In a second step, the models are evaluated through the backtesting framework. For the CustomCNN-based strategy, the test period shows a substantial increase in equity relative to the initial capital, a moderate win rate with positively skewed trade distribution, and attractive risk-adjusted indicators, albeit at the cost of deep drawdowns.

Balancing these two perspectives, **CustomCNN is selected as the final model**. It offers the best compromise between statistical quality and trading profitability, while maintaining a level of complexity that is still interpretable and operationally manageable.

6. Data Drift Monitoring

6.1 Drift detection methodology

To assess the stability of the feature space over time, the project includes a data drift monitoring component that evaluates how the distribution of each input feature changes across the three chronological segments: training, validation and test. The objective is to detect regime shifts in the behavior of NVDA and its technical indicators that may degrade model performance if not addressed through retraining or model adaptation.

For continuous features—which include most of the engineered technical indicators (momentum, volatility and volume measures)—the methodology relies on the Kolmogorov–Smirnov (KS) test. For a given feature, its empirical distribution in the training period is compared with its distribution in a later period (validation or test). The KS statistic measures the maximum distance between the two cumulative distribution functions, and the associated p-value indicates whether the observed difference is likely to have arisen by chance under the null hypothesis of “no distributional change”.

For features that are discrete or effectively categorical (for example, if any indicator is discretized into bins or if auxiliary categorical variables were to be included), the framework allows for the use of a Chi-squared test on frequency counts across categories. This test evaluates whether the distribution of categories has shifted significantly between periods.

In both cases, the project adopts a simple and transparent decision rule:

- A feature is flagged as exhibiting significant drift if its test yields a p-value < 0.05, indicating that the probability of observing such a discrepancy under the assumption of stability is below 5%.

By applying these tests feature by feature, the monitoring component builds an aggregated view of which indicators remain stable over time and which ones experience substantial shifts as the market evolves from the training regime into validation and test regimes. This information is later used to interpret changes in model performance and to support decisions regarding retraining frequency and model robustness.

6.2 Data drift dashboard in Streamlit

Data drift is monitored through an interactive Streamlit dashboard that provides both a graphical and a tabular view of how the feature distributions evolve from the training period into validation and test. The dashboard is organized into three main components.

First, a time-series panel displays, for each selected feature, the evolution of its mean and standard deviation over time, segmented by the three regimes (train, validation, test). This allows the user to visually check whether an indicator remains stable around a given level or whether it undergoes clear shifts in location or dispersion as the market changes. Sudden jumps in the mean or persistent increases in the standard deviation are early signs of potential drift.

Second, a drift summary table lists all features together with their statistical drift diagnostics. For each feature, the table reports:

- the feature name,
- the p-value of the KS test comparing its distribution in training versus a later period, and
- a binary flag indicating whether drift has been detected according to the chosen threshold (p-value < 0.05).

This tabular view makes it straightforward to sort or filter features by severity of drift and to identify which indicators are most affected by regime shifts.

Finally, the dashboard includes a summary section that highlights the top five features with the strongest evidence of drift, i.e. those with the smallest p-values. For each of these features, a brief qualitative interpretation is provided, typically linking the detected drift to changes in market volatility, trading activity or trend structure. For example, strong drift in volatility-related indicators may be associated with periods of heightened uncertainty, while drift in volume-based features may reflect structural changes in market participation or liquidity conditions.

6.3 Features with significant drift and market context

The drift analysis reveals that not all features remain stable as the market evolves from the training period into the validation and test periods. Several volatility- and momentum-related indicators tend to exhibit the lowest p-values in the KS tests, signaling the strongest distributional shifts. Examples include:

- Volatility measures such as ATR and rolling historical volatility, whose level and dispersion change substantially between calm and stressed market phases.
- Momentum indicators such as ROC and some RSI configurations, which behave very differently in prolonged rallies compared to sideways or corrective markets.
- In some cases, volume-based features, especially those capturing relative volume (current volume vs. moving average), also show drift as market participation patterns evolve over time.

These patterns are consistent with the economic history of NVDA over the last decade. The stock has gone through extended bullish trends, abrupt corrections and post-2020 volatility spikes linked to macroeconomic uncertainty and changing expectations about the technology sector. In such environments, it is natural for the distribution of volatility, momentum and volume indicators to shift markedly between the early part of the sample (used for training) and the later periods (validation and test).

From the modelling perspective, these results have two main implications:

1. They confirm that the model is operating in a non-stationary environment, where the statistical properties of key features change over time. Even a well-

calibrated CNN trained on historical data will see its effective input distribution drift, which can degrade performance if left unaddressed.

2. They highlight the importance of regular retraining or adaptation. When the drift monitoring detects persistent and significant shifts in core features—especially those related to volatility regimes or structural changes in trading activity, it is advisable to update the model using more recent data, or at least to reassess its parameters and thresholds.

In summary, the drift analysis does not invalidate the strategy, but it emphasizes that any deep learning model used in live trading on NVDA should be treated as a time-sensitive component, requiring periodic review and retraining as the market transitions through different regimes.

7. Backtesting Methodology

7.1 Signal generation from model predictions

In the backtesting framework, trading decisions are driven directly by the probabilistic outputs of the selected CustomCNN model. For each trading day within the backtest period, the model receives the most recent 10-day window of features and produces a probability distribution over the three possible actions:

- **Short (−1)**
- **Hold (0)**
- **Long (+1)**

The operational signal for that day is obtained by taking the class with the highest predicted probability (argmax). This converts the model output into a single discrete recommendation: short, stay flat, or go long.

To translate these daily signals into actual positions, the strategy uses a simple state-transition rule based on the current position:

- If the portfolio is in Hold (no position) and the model issues a Long signal, the strategy opens a long position.

- If the portfolio is in Hold and the model issues a Short signal, the strategy opens a short position.
- If the portfolio is currently Long and the model's signal switches to Hold or Short, the long position is closed; if the new signal is Short, a short position is opened immediately.
- Symmetrically, if the portfolio is Short and the model's signal switches to Hold or Long, the short position is closed; if the new signal is Long, a long position is opened.

In other words, the model's prediction each day defines a target position (Short, Hold or Long), and the back testing engine adjusts the portfolio to match that target, closing and opening positions as needed. This design creates a direct and transparent link between the model's classification output and the realized trades, making it possible to evaluate how the sequence of predicted signals translates into actual profit and loss once transaction costs and borrowing costs are considered.

7.2 Position sizing rules and trade management

The backtesting engine assumes a **single-asset, fully invested strategy** with simple and transparent position-sizing rules:

- The **initial capital** is set to **USD 100,000**.
- Whenever the model issues a **Long** or **Short** signal, the strategy adopts an **all-in position**:
    - 100 % of the available capital is deployed on the long side when the signal is Long.
    - 100 % of the available capital is deployed on the short side when the signal is Short, subject to the constraint of effective capital and the mark-to-market evolution of existing positions.
- When the signal changes to **Hold**, any open position is **fully closed**, and the portfolio remains in cash until a new Long or Short signal is generated.

To approximate real-world trading conditions, the backtest incorporates explicit transaction and financing costs:

- A commission of 0.125 % is charged on the notional value of each transaction, both on entry and on exit. Every round trip therefore incurs two commission charges.
- For short positions, an additional annualized borrow rate of 0.25 % is applied, prorated by the number of days the short is held. This represents the cost of borrowing shares to sell them short.

For each completed trade, the net profit or loss (P&L) is computed by:

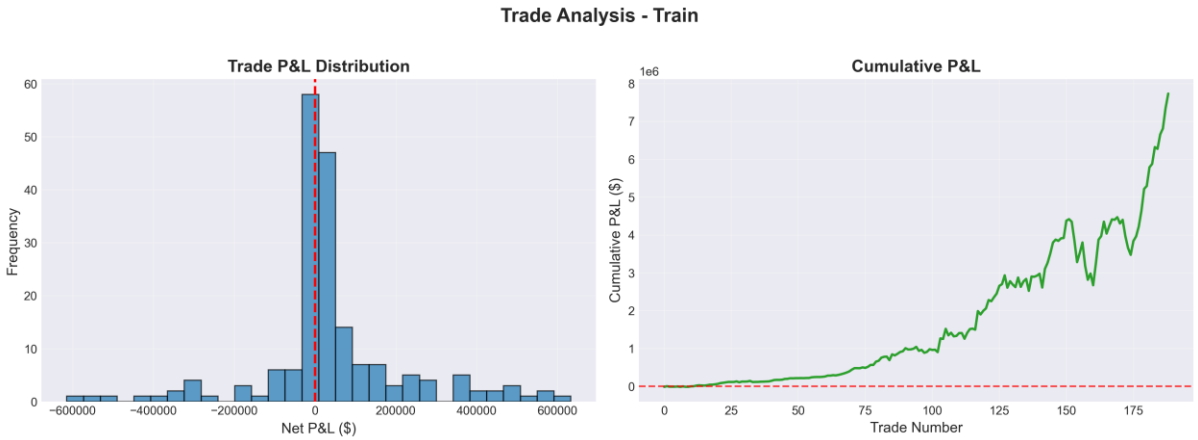
1. Marking the position to market between entry and exit prices.
2. Subtracting all associated transaction costs (commissions on entry and exit).
3. For shorts, subtracting the accrued borrowing cost over the holding period.

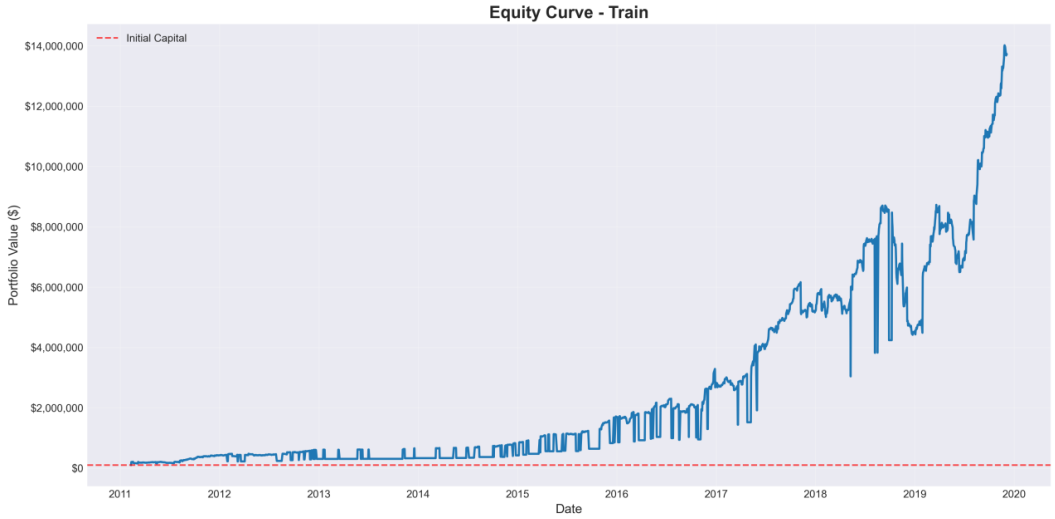
These trade-level P&L values are then aggregated to obtain:

- **Cumulative P&L curves** over time, showing how the equity of the strategy evolves during the train, validation and test periods.
- **Histograms of P&L per trade**, which illustrate the distribution of individual outcomes (small frequent losses vs. occasional large gains, symmetry or skewness, tail behavior).

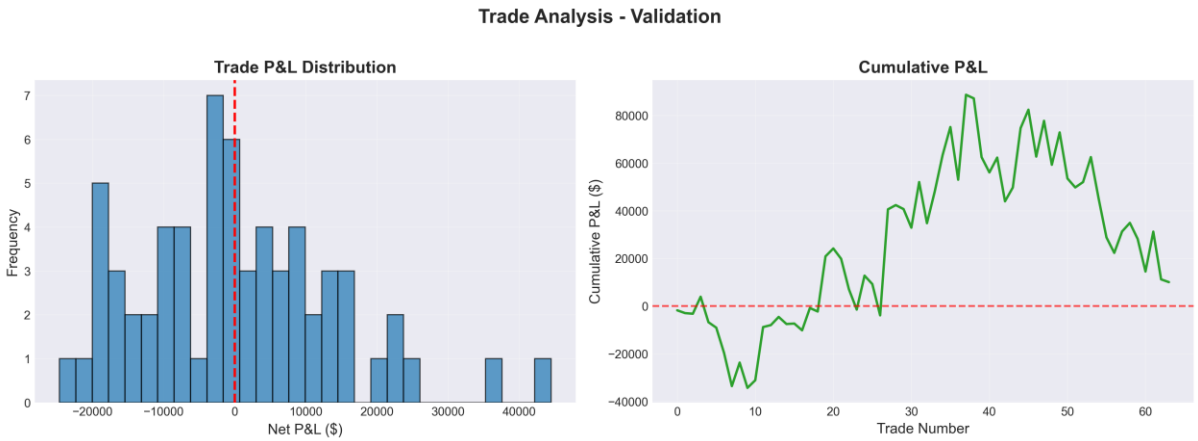
Figure 3 – Trade-level P&L distribution and cumulative P&L

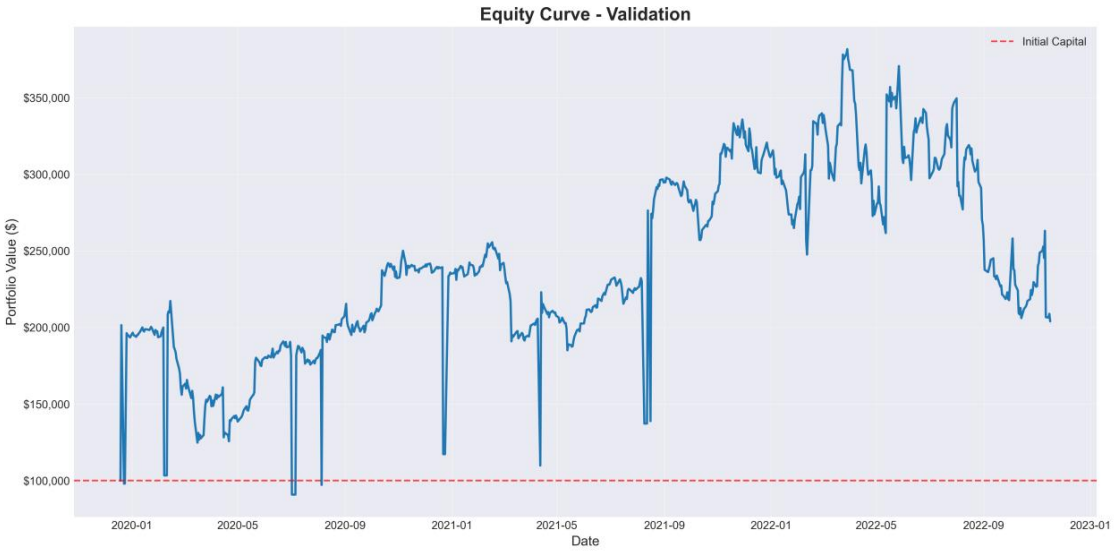
Train


Trade Analysis - Train
Trade P&L Distribution
Frequency
Net P&L ($)
Cumulative P&L
1e6
Cumulative P&L ($)
Trade Number

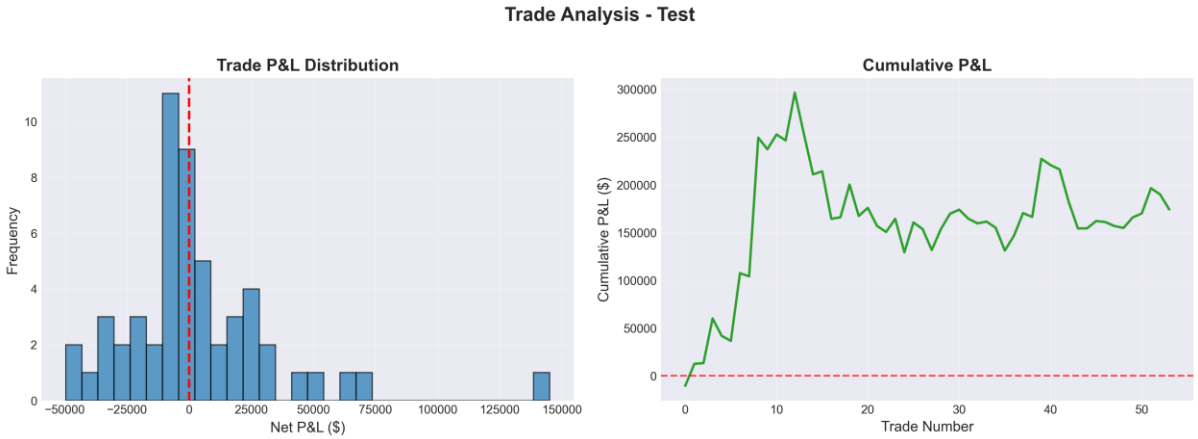

Equity Curve - Train
Initial Capital
Portfolio Value ($)
Date

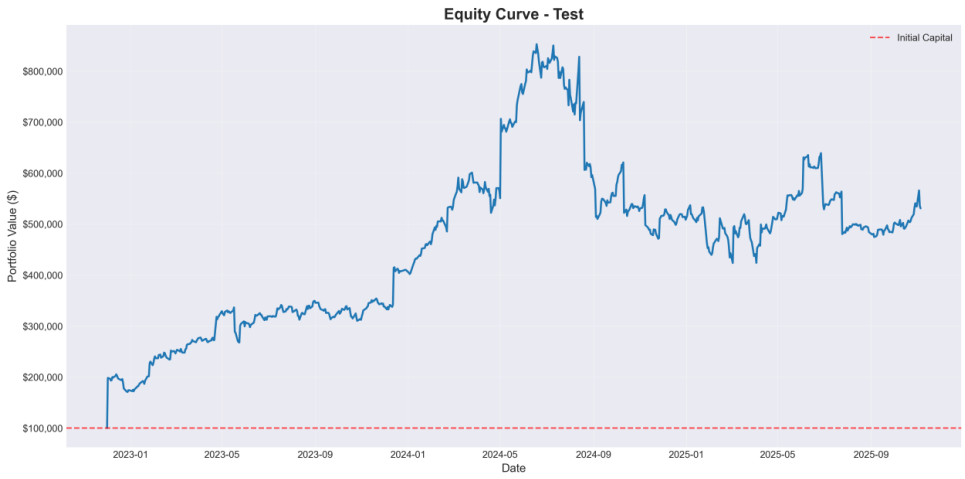
Validation


Trade Analysis - Validation
Trade P&L Distribution
Frequency
Net P&L ($)
Cumulative P&L
Cumulative P&L ($)
Trade Number


Equity Curve - Validation
Initial Capital
$350,000
$300,000
$250,000
$200,000
$150,000
$100,000
Portfolio Value ($)
2020-01
2020-05
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01
Date

Test


Trade Analysis - Test
Trade P&L Distribution
Frequency
Net P&L ($)
Cumulative P&L
Cumulative P&L ($)
Trade Number


Equity Curve - Test
Initial Capital
$800,000
$700,000
$600,000
$500,000
$400,000
$300,000
$200,000
$100,000
Portfolio Value ($)
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
2025-05
2025-09
Date

This combination of sizing rules and realistic cost modelling provides a conservative but clear view of how the model's signals would have translated into actual economic outcomes under a straightforward, fully invested trading policy.

7.3 Walk-forward evaluation scheme

The evaluation of the strategy follows a chronological walk-forward scheme that mimics how the model would be used in practice. The full 15-year history of NVDA is divided into three non-overlapping, time-ordered blocks:

- **Training period (~60 % of the data)**

  Approximately from **2010 to 2019**, used exclusively to fit the model parameters and compute feature-scaling statistics.

- **Validation period (~20 % of the data)**

  Roughly from 2019 to 2022, it was used for model selection and hyperparameter tuning. No retraining occurs within this block; instead, different architectures and configurations trained on the training period are compared based on their performance here.

- **Test period (~20 % of the data)**

  Approximately from **2022 to 2025**, reserved as a **strict out-of-sample segment**. The chosen model (CustomCNN) is evaluated on this block without any further adjustment.

At each stage, the model only has access to data prior to the period being evaluated, strictly preserving time order. This avoids any form of look-ahead bias and reflects a realistic scenario in which the model is trained on historical data and then applied to future, unseen observations.

Although there is no rolling retraining in the current implementation (the model is trained once on the initial 60 % of the data), the three-block structure can be interpreted as a coarse walk-forward evaluation: performance is examined sequentially on a validation regime and then on a later test regime, each with potentially different market conditions. In future extensions, this framework could be

refined into a true rolling or expanding-window walk-forward procedure, with periodic retraining as new data become available.

7.4 Assumptions and limitations of the backtest

While the backtesting setup is designed to be realistic, it inevitably relies on several simplifying assumptions that must be kept in mind when interpreting the results:

- **No explicit modelling of slippage or liquidity constraints**

  Trades are assumed to be executed at the **daily closing price**, with no additional slippage beyond the fixed commission. Given NVDA's high liquidity, this assumption is reasonable for moderate position sizes, but it may underestimate execution costs under stressed conditions or for very large orders.

- **All-in position sizing and deep drawdowns**

  The strategy allocates 100 % of capital to each Long or Short position. This all-in approach amplifies both gains and losses and leads to very deep drawdowns (around −59 % in the training period and close to −50 % in the test period). In any realistic implementation, such drawdown levels would be unacceptable and would require more conservative position sizing, diversification, or explicit risk constraints.

- **Single-asset focus and lack of diversification**

  The backtest is conducted on **NVDA only**, with no diversification across assets, sectors or strategies. As a result, the performance is heavily exposed to idiosyncratic risk and to structural changes specific to this stock. In practice, a model like this would likely be deployed as part of a **multi-asset or multi-strategy portfolio**, not in isolation.

- **No interaction with other risk-management or overlay systems**

  The strategy does not incorporate additional overlays such as volatility targeting, regime filters, stop-loss frameworks or capital allocation rules across strategies. This keeps the analysis focused on the raw behavior of the model-

driven signals but understates the type of safeguards that would be present in a production environment.

Given these assumptions, the backtest should be interpreted primarily as proof of concept of the integration between deep learning, MLOps (MLFlow) and a modular backtesting engine, rather than as a fully production-ready trading system. The results demonstrate the potential of the approach and highlight its risks, providing a foundation on which more sophisticated risk management and portfolio construction layers can be built.

8. Results and Performance Analysis

8.1 Equity curves and drawdowns

Training period

Figure 4 shows the equity curve of the strategy during the training period. Starting from an initial capital of USD 100,000, the equity grows explosively to approximately USD 7.43 million, implying an extremely high in-sample return. This reflects the model's ability to exploit patterns present in the historical data on which it was fitted but also raises concerns about overfitting.

Figure 4 - Equity curve, Training period

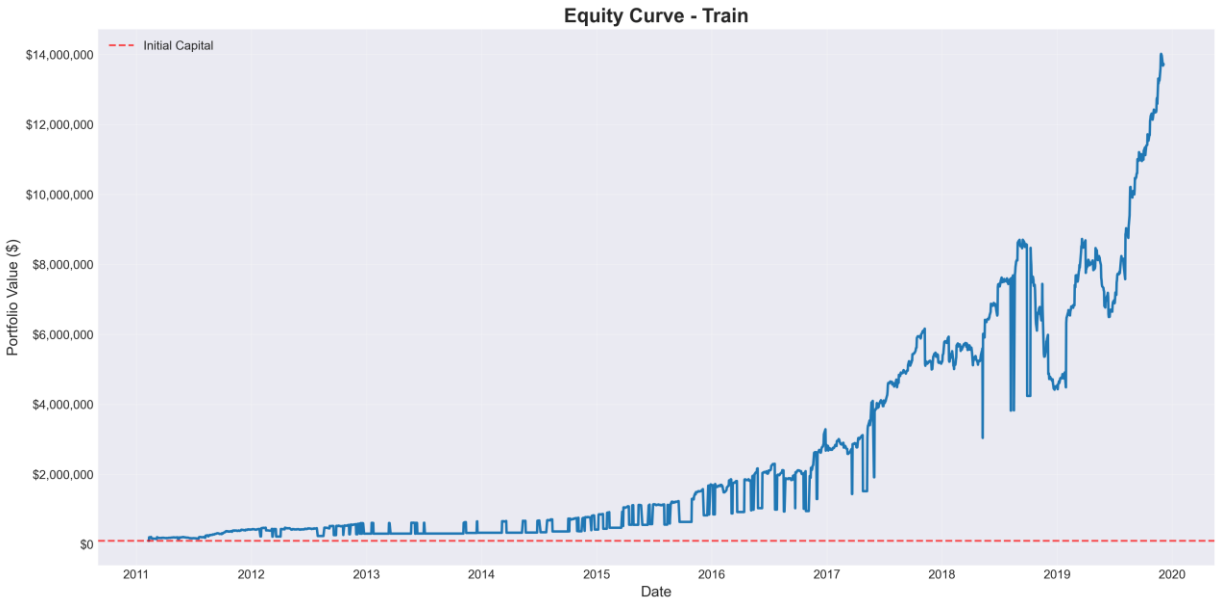
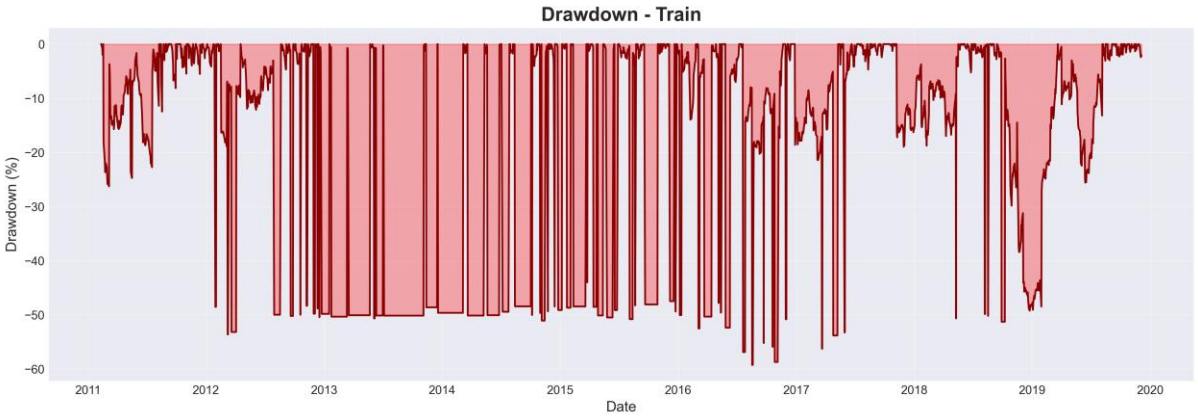


Figure 5 presents the corresponding drawdown curve. Despite the strong overall upward trajectory, the strategy experiences **deep interim losses**, with a **maximum drawdown close to −59%**. At several points, the portfolio loses more than half of its value from peak to trough before recovering. This combination of very high returns and very high drawdowns is a strong indication of **aggressive risk-taking and in-sample overfitting**.

Figure 5 - Drawdown curve, Training period



Validation period

In contrast, Figure 6 shows a much **flatter equity curve** for the validation period. The final equity ends around **USD 100,254**, i.e. only a marginal gain relative to the initial capital. Over this segment, the strategy essentially oscillates around break-even, with gains and losses largely offsetting each other.

Figure 6– Equity curve, Validation period

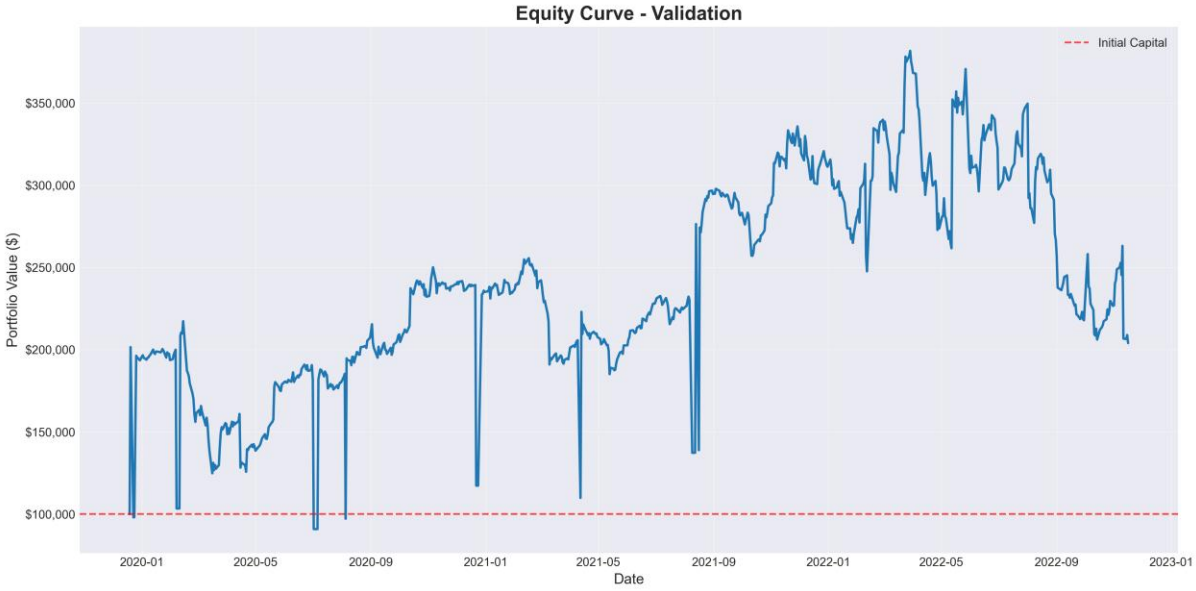
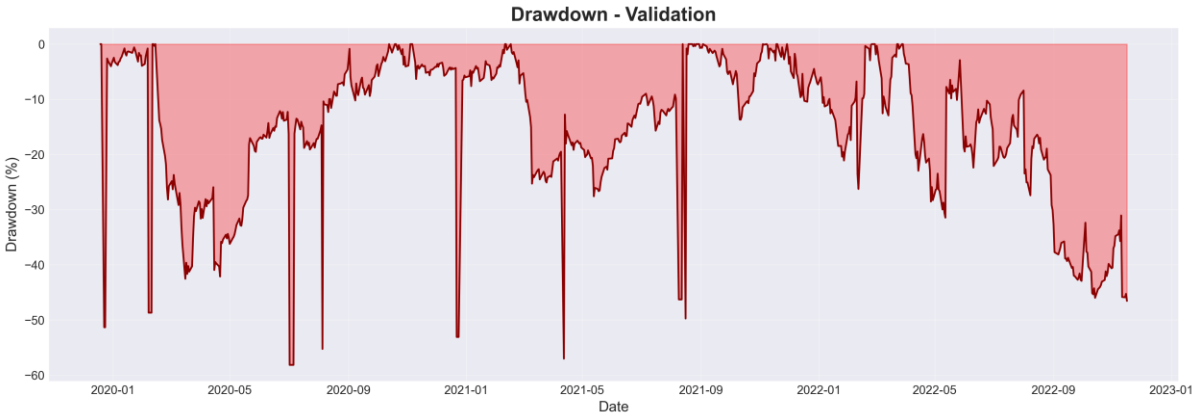


Figure 7 shows that, despite the flat net result, the **risk profile remains severe**. The maximum drawdown during validation reaches approximately **−58 %**, which is very similar to the training period. This means that even when the strategy does not generate meaningful profits, it still exposes the portfolio to **very large temporary losses**, which is unattractive from a practical risk–return perspective.

Figure 7 – Drawdown curve, Validation period



Test period

Figure 8 displays the equity curve in the test period, which is reserved as a strict out-of-sample segment. Here, the strategy achieves a **substantial positive result**: the equity grows from **USD 100,000** to approximately **USD 257,285**, corresponding to a total return of about **157 %** after transaction and borrowing costs. The curve shows several strong upward phases interspersed with periods of stagnation and partial

reversals, consistent with the volatile nature of NVDA and the model's directional signals.

Figure 8 – Equity curve, Test period

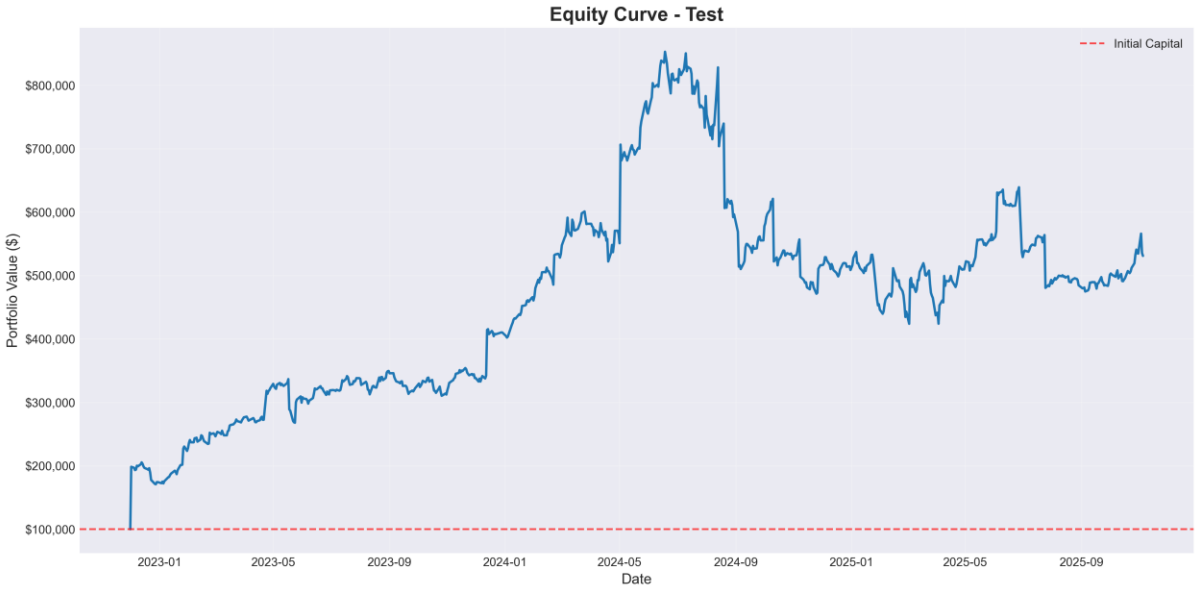
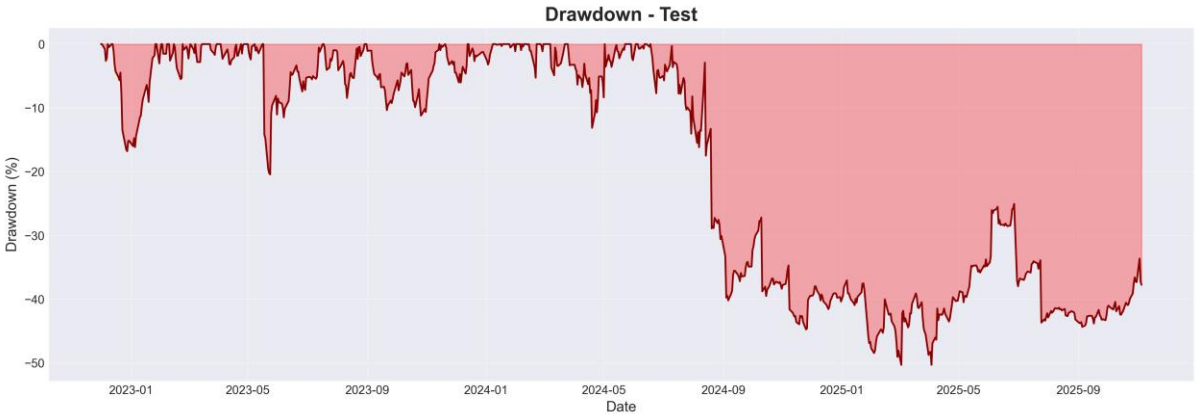


Figure 9 presents the drawdown curve for the test period. The maximum drawdown is close to **−50 %**, indicating that at some point the portfolio loses about half of its value relative to a prior peak. This pattern—strong but uneven growth combined with deep drawdowns—suggests that, while the strategy can be profitable under certain regimes, it does so at the cost of **high return volatility and substantial interim losses**.

Figure 9 – Drawdown curve, Test period



Taken together, these curves reveal a consistent picture:

- **spectacular but likely overfitted gains in training**,

- **almost flat performance with large drawdowns in validation**, and
- **profitable yet highly volatile behavior in test**.

This highlights both the **potential** of the CNN-based approach to generate significant returns and its **limitations in terms of risk and robustness**, reinforcing the need for stricter risk management and more conservative capital allocation in any realistic implementation.

8.2 Aggregated performance metrics by period

Table 3 summarizes the **aggregated performance** of the strategy in the three periods. It reports not only total return and trade count, but also risk-adjusted measures such as Sharpe, Sortino, maximum drawdown and Calmar ratio.

Table 3 – Performance summary by period

Period	Initial Capital (USD)	Final Equity (USD)	Total Return %	Total Trades	Win Rate %	Sharpe	Sortino	Max Drawdown %	Calmar
Train	100,000	7,430,000	7,333.0	189	70.9	1.36	1.90	−59.3	123.8
Validation	100,000	100,254	0.25	64	45.3	0.91	1.51	−58.2	0.004
Test	100,000	257,285	157.3	54	46.3	1.06	2.09	−50.3	3.13

From this table, three key patterns emerge:

- In the training period, the strategy delivers extraordinary in-sample performance (over 7,300 % total return), with a high win rate and strong Sharpe and Calmar ratios, but at the cost of very large drawdowns (around −59 %), signaling aggressive risk-taking and likely overfitting.
- In the validation period, performance almost collapses to flat, with virtually no net gain, while deep drawdowns persist. This combination of negligible returns and extreme risk highlights the fragility of the strategy when exposed to new market conditions.

- In the test period, the strategy recovers and achieves a solid positive return (about 157 %), with acceptable risk-adjusted metrics (Sharpe above 1 and Sortino above 2), but still with maximum drawdowns around −50 %, indicating that the path to those returns is highly volatile and psychologically demanding.

8.3 Trade statistics

Beyond aggregate performance metrics, the distribution of trade outcomes provides important insight into how the strategy generates profits and losses.

The histograms of trade-level P&L for the three periods (Train, Validation, Test) show highly skewed distributions. In all cases, there is a dense cluster of relatively small gains and losses around zero, together with a long right tail corresponding to a limited number of very profitable trades. This pattern indicates that the strategy's overall result is heavily influenced by a small subset of "outlier" winning trades, while many other trades contribute modestly or even negatively.

Across periods, several common features can be observed:

- There is a clear difference between average gain and average loss. Even when the win rate is only moderate in Validation and Test (around 45–46 %), the strategy tends to earn more per winning trade than it loses per losing trade, which is crucial for profitability in the presence of noisy predictions.
- The number of trades decreases from Train (almost 190 trades) to Validation and Test (around 60 and 50 trades respectively), reflecting both the shorter length of these periods and changes in the frequency of signals generated by the model under different regimes.
- The typical duration of trades is relatively short, often spanning only a few days, consistent with the 5-day labelling horizon and the daily re-evaluation of signals. Positions rarely remain open for very long, which keeps exposure aligned with recent model predictions but also increases turnover and transaction costs.
- There is a systematic predominance of long trades over short trades, consistent with the underlying upward bias of NVDA and with the class distribution of the

labels. Short trades are fewer but often concentrated on specific stress episodes, where they can materially improve performance if timed correctly.

Overall, the trade-level analysis reinforces the idea that the strategy does not generate smooth, incremental gains. Instead, it relies on frequent, small fluctuations around zero and a minority of large positive trades that dominate the outcome—particularly in the training and test periods.

8.4 Relationship between model accuracy and profitability

An important aspect of the evaluation is the relationship between classification accuracy and trading performance. On the test set, the CNN architectures achieve moderate accuracy levels, typically in the range of 41–46 %, even for the best-performing model. At first glance, such figures may appear low compared to standard machine learning benchmarks.

However, the backtest shows that, despite this modest accuracy, the CustomCNN-based strategy still achieves a cumulative net return of approximately 157 % in the test period, after accounting for transaction and borrowing costs. This apparent discrepancy highlights a key point:

The link between classification accuracy and trading P&L is not linear.

What matters for profitability is not simply the fraction of correct predictions, but:

- The asymmetry between gains and losses: if correct signals tend to coincide with large price moves (big winners) and incorrect signals occur in periods of small or mean-reverting moves (small losers), the strategy can be profitable even with modest accuracy.
- The risk management framework and position sizing: an all-in strategy amplifies both the impact of good and bad signals. In more conservative settings, the same accuracy might lead to smaller but more stable returns.
- The presence of a few “outlier” winning trades as seen in the trade histograms, a limited number of exceptional trades can account for a large share of total profits, compensating for many small or moderate losses.

In this context, the results obtained in the test period illustrate the following idea:

“A model with only moderate classification accuracy can still be profitable if its correct predictions tend to coincide with large market moves, while its errors are concentrated in periods with limited price action.”

At the same time, the deep drawdowns observed across all periods show the downside of this configuration: when a sequence of misclassified or poorly timed signals occurs during a volatile regime, losses can accumulate rapidly. Accuracy alone is therefore an incomplete measure of quality; it must be interpreted jointly with payoff asymmetry, drawdown behavior and the robustness of risk management.

9. Conclusions

9.1 Key findings and strategy viability

The project demonstrates that a CNN-based, MLOps-oriented trading framework can be successfully implemented end-to-end: from feature engineering and deep learning modelling, through experiment tracking with MLFlow, to a realistic backtesting engine with transaction and borrowing costs. Within this framework, the CustomCNN model provides evidence of potential profitability in the test period, delivering a cumulative net return of around 157 % after costs.

At the same time, the results clearly show that this profitability comes with a very high-risk profile. The validation segment reveals that the strategy is highly sensitive to regime changes and that there is substantial overfitting to the training period: performance in validation is almost flat, yet deep drawdowns persist. Across all periods, the strategy experiences large peak-to-trough losses, often exceeding 50%, which would be difficult to tolerate in a real trading environment.

Taken together, these findings suggest that, in its current form, the strategy is better viewed as a research module—a proof of concept for integrating deep learning, drift monitoring and backtesting—rather than as a fully deployable stand-alone trading system.

9.2 Model selection summary

Three CNN architectures were evaluated: SimpleCNN, DeepCNN and CustomCNN. The final choice of model is CustomCNN, for three main reasons:

- It achieves the highest F1-score on the test set, indicating a more balanced treatment of the three classes (Short, Hold, Long) than the other architectures, even if its raw accuracy is slightly lower than that of SimpleCNN.
- Its multi-scale architecture, with parallel convolutional branches operating at different temporal resolutions, offers a more flexible representation of short- and medium-term patterns in technical indicators.
- It provides a reasonable trade-off between capacity and regularization, especially when combined with Dropout, L2 penalties and Batch Normalization.

When embedded in the back testing framework, the CustomCNN-based strategy delivers, in the test period, a net return of approximately 157 %, with a Sharpe ratio above 1 and a Calmar ratio around 3.1, even after including realistic transaction and borrowing costs. These figures support the choice of CustomCNN as the most promising architecture within the current experimental set-up, while still acknowledging its risk limitations.

9.3 Profitability after costs and recommendations for future work

Profitability after costs

The strategy explicitly incorporates a 0.125 % commission per transaction and a 0.25 % annualized borrow rate for short positions, prorated over the holding period. Even after these frictions, the CustomCNN model remains profitable in the test segment, confirming that its edge is not entirely consumed by trading costs. However, the path to this profitability is characterized by large and prolonged drawdowns, which would pose serious challenges in live trading from both risk management and behavioral standpoints.

Recommendations for extensions and improvements

To move from a proof of concept to a more robust and implementable strategy, several avenues for improvement are recommended:

- **Reduce effective position size**

  Replace the all-in approach with **more conservative capital allocation**—for example, limiting each position to **20–30 % of available capital**. This would

likely reduce drawdowns substantially, at the cost of lower nominal returns, and would make the strategy more compatible with realistic risk budgets.

- **Introducing periodic retraining and rolling schemes**

  Exploit the information provided by data drift monitoring: when significant and persistent drift is detected in key features, the model should be retrained on more recent data using an expanding or rolling window. This would help the strategy adapt to new market regimes rather than relying indefinitely on patterns learned from older periods.

- **Test across additional assets and markets**

  Extend the current framework to other large-cap technology stocks or sector indices. This would allow an assessment of the generalisation capacity of the approach and open the possibility of constructing multi-asset portfolios, which can reduce idiosyncratic risk.

- **Incorporate temporal regularisation of signals**

  Introduce mechanisms that penalise excessive day-to-day signal changes, for example by adding constraints or costs linked to frequent position flips. This could reduce unnecessary trades, lower transaction costs and smooth the equity curve.

- **Combine CNN signals with market risk indicators**

  Augment the model-driven signals with macro or market-level risk measures, such as volatility indices (e.g. VIX), sector indices or simple regime filters. These additional signals could be used to filter out adverse environments (high stress, low liquidity) or to modulate position size depending on the prevailing risk regime.

The current implementation establishes a solid methodological foundation—deep learning, MLOps tracking, drift analysis and backtesting—but also makes clear that strong risk controls, diversification and adaptive mechanisms are essential for turning this type of model into a viable component of a production trading system.