

Aim: Will volatility over the next 5 trading days be high or low?

Justification: If we can reliably detect high-vol regimes, we can decide when it is safer to hold positions for 3–5 days, and when we should be more cautious.

Formal Aim: Given daily OHLCV and engineered features for US stocks, learn a model that, at each day t , predicts whether the realised volatility of log returns over days $t+1 \dots t+5$ will lie above or below the median ($\text{HIGH_VOL_Q} = 0.5$). Equivalently, we classify each (ticker, date) into:

- Class 1 (high-vol regime): $\text{realized_vol_5}(t) \geq \text{median}(\text{realized_vol_5} \text{ on train})$
- Class 0 (low-vol regime): otherwise.

Modelling Method: CNN with pipeline

Phase 1: Data & Label Construction

- **Base Data:** combined_cleaned.csv: Contains daily data for many US stocks from 2009–2017: Open, High, Low, Close, Volume, Ticker, Date, plus other precomputed features.
- **Engineered Data:** Variables representing log returns & realised volatility, such as:
 - $\text{log_return_t} = \log(\text{Close_t}) - \log(\text{Close_}\{t-1\})$ (per ticker).
 - $\text{realized_vol_5}(t) = \sqrt{(r_{\{t+1\}}^2 + \dots + r_{\{t+5\}}^2) / 5}$. This is the future 5-day realised volatility used to define the label.
- **Label Classification:** High-vol & low-vol label:
 - On train data only, compute the median of realized_vol_5. Afterwards, we apply labelling using the following definition:
 - 1 if $\text{realized_vol_5} \geq \text{median}$
 - 0 otherwise.

Phase 2: Features & Sequences Building

For each (Ticker, Date) set:

- We build a 10-day sliding window ($\text{LOOKBACK} = 10$).
- Input features include:
 - Trend & momentum: log_return, lag1, lag3, lag5, MA5, MA20, MA_diff
 - Volatility & risk: vol_5, return_over_vol
 - Volume & liquidity: Volume, rel_vol, volume_spike

- Cross-sectional info (per day across tickers): log_return_z_cs, MA5_z_cs, MA20_z_cs, vol_5_z_cs, Volume_z_cs, rel_vol_z_cs, log_return_rank_cs, vol_5_rank_cs, Volume_rank_cs
 - Market regime: regime_high_vol.
- After performing another set of data cleaning, we obtain 27,379 samples. Corresponding to each sample having a shape of (10, 22), equivalent to 10 days × 22 features.
- As such, we define the model inputs as: $X_i \in \mathbb{R}^{10 \times 22}$, $y_i \in \{0,1\}$

Phase 3: CNN Building

The chosen config for the run is as follows:

- Temporal window: LOOKBACK = 10 (past 10 trading days).
- Extractor: Two stacked Conv1D blocks, each containing:
 - Conv1D(64 filters, kernel size = 5, ReLU, padding="same")
 - BatchNorm → MaxPooling1D(2) → Dropout(0.3)
- Classifier head:
 - Flatten → Dense(64, ReLU) → Dropout(0.4) → Dense(1, sigmoid)

Training setup:

- Loss: binary_crossentropy
- Optimiser: Adam (LR = 1e-3)
- Batch size: 512
- Early stopping: patience 10 epochs on val_loss.

Rationale:

- Kernel size 5 approximates “one week pattern”; 2 conv layers with pooling let the network detect multi-day build-ups and compress them.
- Moderate capacity (64 filters, 64-unit dense layer) is enough since we don’t need huge models to learn this regime.

Phase 4: Evaluation Protocol Using Walk-forward

We use a 3-fold walk-forward time-series cross-validation scheme, where:

- Fold 1:
 - Train: 2009–2013
 - Val: 2014
 - Test: 2015
- Fold 2:
 - Train: 2009–2014
 - Val: 2015
 - Test: 2016
- Fold 3:
 - Train: 2009–2015
 - Val: 2016
 - Test: 2017

For each fold, we:

1. Train the CNN with early stopping.
1. Predict probabilities on validation set; tune threshold τ over the discrete set $\{0.40, 0.45, 0.50, 0.55, 0.60\}$ to maximise validation accuracy.
2. Apply the best τ to test probabilities and then compute test accuracy.

We obtain the following results:

Per-fold test accuracy:

fold	test year	accuracy	baseline_acc	tuned τ	pos_ratio_train	n_train	n_test
fold1	2015	0.6620	0.4233	0.45	0.50	5,884	1,512
fold2	2016	0.7560	0.3618	0.50	0.50	7,306	1,512
fold3	2017	0.8491	0.2358	0.45	0.50	8,818	1,272

Average across folds:

- CNN mean test accuracy: 0.7557
- Baseline mean accuracy: 0.3403

We can see from the above that CNN roughly adds +0.41 absolute accuracy over the baseline, more than double the naive classifier.

With regards to the training curves:

- Train loss falls steadily from 0.8 to around 0.45 - 0.55.
- Val loss drops early and then plateaus, sometimes slightly rises. Early stopping kicks in after 20–30 epochs, which prevents the model from continuing its trend of overfitting.

In conclusion, we can obtain some Insights from these results:

- The 5-day risk regime is genuinely predictable.
 - Baseline (majority class) accuracy sits at around 0.34 because train labels are balanced, but test distributions vary.
 - CNN achieves an accuracy of around 0.66 to 0.85 across years, averaging at 0.75, showing that:
 - There are stable temporal patterns in returns/volume that precede high-volatility weeks.
 - These patterns generalise across different calendar years (2015–2017).
- More training data results in better performance in later years.
 - Accuracy by fold increases from 0.66 to 0.76, and finally to 0.85, while `n_train` increases from 5,884 to 7,306, and finally to 8,818.
 - As we include more history in training, the CNN sees more examples of different volatility regimes.
 - This is most evident in fold3, which has the longest training window using data from 2009 to 2015, and performs best on the test data of 2017.
 - This pattern mirrors a realistic deployment setting, where we would train on all available past years and then apply the model to the upcoming year.
- A 10-day context & weekly kernel is a good scale for predicting 5-day volatility.
 - With `LOOKBACK = 10` (around two weeks of history) and `kernel_size=5`, we reach 0.75 mean accuracy.
 - Suggesting that the 5-day regime is mostly driven by recent 1–2 week patterns, such as sequences of large/small returns, rising realised volatility, volume spikes or cross-sectional extremes.
 - We can thus conclude that volatility tends to cluster over short horizons.
- Tuning the classification threshold matters.

- Each fold's best τ is as follows:
 - Fold1: 0.45
 - Fold2: 0.50
 - Fold3: 0.55
- They all lay around 0.5, but not exactly equal.
- It also shows that we're not overfitting the model by picking extreme thresholds (e.g., 0.1 or 0.9).